energies

MDPI

Review

High-Performance and Parallel Computing Techniques Review: Applications, Challenges and Potentials to Support Net-Zero Transition of Future Grids

Ahmed Al-Shafei [1,*], Hamidreza Zareipour [1,*] and Yankai Cao [2]

[1] Department of Electrical and Software Engineering, University of Calgary, Calgary, AB T2N 1N4, Canada
[2] Department of Chemical and Biological Engineering, University of British Colombia, Vancouver, BC V6T 1Z3, Canada
* Correspondence: ahmed.alshafei@ucalgary.ca (A.A.-S.); hzareipo@ucalgary.ca (H.Z.)

Abstract: The transition towards net-zero emissions is inevitable for humanity's future. Of all the sectors, electrical energy systems emit the most emissions. This urgently requires the witnessed accelerating technological landscape to transition towards an emission-free smart grid. It involves massive integration of intermittent wind and solar-powered resources into future power grids. Additionally, new paradigms such as large-scale integration of distributed resources into the grid, proliferation of Internet of Things (IoT) technologies, and electrification of different sectors are envisioned as essential enablers for a net-zero future. However, these changes will lead to unprecedented size, complexity and data of the planning and operation problems of future grids. It is thus important to discuss and consider High Performance Computing (HPC), parallel computing, and cloud computing prospects in any future electrical energy studies. This article recounts the dawn of parallel computation in power system studies, providing a thorough history and paradigm background for the reader, leading to the most impactful recent contributions. The reviews are split into Central Processing Unit (CPU) based, Graphical Processing Unit (GPU) based, and Cloud-based studies and smart grid applications. The state-of-the-art is also discussed, highlighting the issue of standardization and the future of the field. The reviewed papers are predominantly focused on classical imperishable electrical system problems. This indicates the need for further research on parallel and HPC approaches applied to future smarter grid challenges, particularly to the integration of renewable energy into the smart grid.

Keywords: parallel computing; optimization; power system studies; grid transition; renewable energy; IoT; smart grid; cloud computing; high-performance computing

Citation: Al-Shafei, A.; Zareipour, H.; Cao, Y. High-Performance and Parallel Computing Techniques Review: Applications, Challenges and Potentials to Support Net-Zero Transition of Future Grids. *Energies* **2022**, *15*, 8668. https://doi.org/10.3390/en15228668

Academic Editor: Tek Tjing Lie

Received: 28 September 2022
Accepted: 14 November 2022
Published: 18 November 2022

1. Introduction

To date, the global mean temperature continues to rise, and emissions continue to grow, creating great risk to humanity. Efforts and pathways are drawn by many jurisdictions to limit warming to 2 °C and reach net zero CO_2 emissions [1]. This is largely due to the outdated electrical energy system operation and infrastructure, which causes the largest share of emissions of all sectors. Electrical energy systems are, however, witnessing a transition, consequently causing a growth in the scale and complexity of their planning and operation problems. The changing grid topology, decarbonization, electricity market decentralization, and grid modernization mean innovations and new elements are continuously introduced to the inventory of factors considered in grid operation and planning. Moreover, with the accelerating technological landscape and policy changes, the number of potential future paths to Net-Zero increases, and finding the optimal transition plan becomes an inconceivable task.

The use of parallel techniques becomes inescapable, and defaulting HPC competence by the electrical energy and power system community is inevitable in the face of the presumed future and its upcoming challenges. With some algorithmic modifications, parallel

computing unlocks the potential to solve huge power system problems that are conventionally intractable. This helps in the reduction of cost and CO_2 emissions indirectly through detailed models that help us find less conservative operational solutions—which reduce thermal generation commitment and dispatch—and plan the transition to a net-zero grid with optimal placement of the continually growing inventory of Renewable Energy (RE) resources and smart component investment—which help us achieve the Net-Zero goal and future proof the grid. Moreover, parallel processing on multiple units is inherently more efficient and reduces energy use. Using multi-threading causes the energy consumption of multi-processors to increase drastically [2]. Thus, in the case of resource abundance, it is more efficient to distribute work on separate hardware. Resource sharing is more effective than resource distribution, which reduces the demand for hardware investment and larger servers. All of these factors make it increasingly important for electrical engineering scientists to familiarize themselves with efficient resource allocation and parallel computation strategies.

North America [3], the EU [4], and many other countries [5] set a target to completely retire coal plants earlier than 2035 and decarbonize the power system by 2050. In addition, the development of Carbon Capture and Storage Facilities is growing [6]. Renewable energy penetration targets are set, with evidence of fast-growing proliferation across the globe, including both transmission-connected Variable Renewable Energy (VRE) [7] and behind the meter distributed resources [8]. The demand profile is changing with increased electrification of various industrial sectors [9] and the transportation sector [10] building electrification, energy efficiency [11,12], and the venture into a Sharing Economy [13].

The emerging IoT, facilitated by low latency, low-cost next-generation 5G communication networks, helps roll out advanced control technologies and Advanced Metering Infrastructure [14,15]. This gives more options for contingency remedial operational actions to increase the grid reliability, and cost-effectiveness, such as Transmission Switching [16], Demand Response [17], adding more micro-grids, and other Transmission–Distribution coordination mechanisms [18]. Additionally, they allow lower investment in transmission lines and look for other future planning solutions, such as flow management devices and FACTs [19], Distributed Variable Renewable Energy [20], and Bulk Energy Storage [21].

Moreover, the future grid faces non-parametric uncertainties in the form of new policies such different as carbon taxation, pricing and certifications schemes [22], feed-in-tariffs [23] and time of use tariffs [24], and other incentives. More uncertainties are introduced in smart grid visions and network topological and economic model conceptual transformations. These include the Internet of Energy [25], Power-to-X integration [26], and the transactive grid through Peer-to-Peer energy trading facilitated by distributed ledgers or Blockchain Energy [27]. Such disruptions create access to cheap renewable energy and potential revenue streams for prosumers and encourage load shifting and dynamic pricing. Many of these concepts already have pilot projects in various locations in the world [28]. Due to all the factors mentioned above, cost-effective real-time operations and decision-making while maintaining reliability becomes extremely difficult, as does planning the network transition to sustain such a dynamic nature and stochastic future.

Many current electrical system operational models are non-convex, mixed-integer, and non-linear programming problems [29] and incorporate stochastic framework accounting for weather, load, and contingency planning [30]. Operators must solve such problems for day-ahead and real-time electricity markets and ensure reliability standards are met. In NERC, for example, the reliability standards require transmission operators to perform a real-time reliability assessment every 30 min [31]. The computational burden to solve these decision-making problems, even with our current grid topology and standards, forces the recourse to cutting-edge computational technology and high-performance computing strategies for online real-time applications and offline calculations to achieve tractability. This is part of the reason for the increased funding for High-Performance algorithms for complex energy systems and processes [32].

Operators already use high-performance computation facilities or services in areas such as Transmission and Generation Investment Planning, Grid Control, Cost Optimiza-

tion, Losses Reduction, Power System Simulation, and Analysis and Control Room Visualization, as seen in [33,34]. However, for operational purposes where problems need to be solved on a short time horizon, system models are usually simplified, and heuristic methods are used, relying on the experience of operators, such as in [35]. As a consequence, these models tend to be conservative in fashion, reaching a reliable solution at the expense of reduced efficiency [36]. According to The Federal Energy Regulatory Commission, finding more accurate solution techniques for complex problems such as AC Optimal Power Flow (ACOPF) could save billions of dollars [37]. This motivates the search for methods to produce high-quality solutions in a reasonable time, and one of the ways to push the wall of speed up is through parallel techniques and HPC. Finding appropriate techniques, formulations, and proper parallel implementation of HPC for electrical system studies has been a research area of interest. Progress has been made to make solving complex, accurate power system models for real-time decisions favorable.

The first work loosely related to parallelism on a high-level task in a power system might have been that of Harvey H. Happ in 1969 [38], in which a hierarchical decentralized multi-computer configuration was suggested, targeting Unit Commitment (UC) and Economic Dispatch (ED). Other similar work in multi-level multi-computer frameworks followed, soon targeting Security and voltage control in the 1970s [39–41]. P. M. Anderson from the Electrical Power Research Institute created a special report in 1977, collecting various studies and simulations performed at the time, which explored the potential applications for power system analysis on parallel processing machines [42]. Additionally, several papers came out suggesting new hardware to accommodate power-system-related calculations [43].

In 1992, C. Tylasky et al. made what might be the first comprehensive review of the state-of-the-art of parallel processing for power systems studies [44]. They discussed challenges still relevant today, such as different machine architectures, transparency, and portability of the codes used. A few parallel power system study reviews have been conducted throughout the development of computational hardware. Some had similar goals to this paper reviewing HPC applications for power systems [45,46] and on distributed computing for online real-time applications [47]. Computational paradigms changed exponentially, reducing those reviews to pieces of history. The latest relevant, comprehensive reviews on the topic were by R. C. Green et al. for general power system applications [48], and again focused on innovative smart grid applications [49]. These two handle a variety of topics in power systems. This work adds to existing reviews, providing a fresh overview of the state-of-the-art. It distinguishes itself by providing the full context and history of parallel computation and HPC in electrical system optimization and its development up to the latest work in the field. It also provides a thorough base and background for newcomers to the field of power system optimization in terms of both computational paradigms and applied algorithms. It highlights the importance of defaulting HPC utilization in a net-zero future grid. Finally, it brings to light the necessity of integrating HPC in future studies amidst the energy transition and suggests a framework that encourages future collaboration to accelerate HPC deployment. Table 1 shows a side-by-side comparison of related reviews.

Most of the existing work in the literature applies to classical problems that are still relevant to the future grid. For example, the electrical system stability problems only become bigger and more complex with the new grid paradigms, especially with the introduction of synthetic inertia coming from massive wind and solar plants at many data points. However, the amount of HPC and cloud computing consolidation to the future smart grid vision is small, so only one section of this work introduces frameworks of cloud-smart grid integration with different network architectures and topologies. Accompanying cloud usage in future studies is important, as it introduces new considerations and complications, such as optimal resource allocation and cloud security issues.

This paper is organized as follows: Sections 2 and 3 identify the main HPC components and parallel computation paradigms. The next six sections review parallel algorithms under both Multiple Instruction Multiple Data (MIMD) and Single Instruction Multiple Data (SIMD) paradigms split into their early development and state-of-the-art for each study,

starting with Section 4 PF. Section 5 Optimal Power Flow (OPF). Section 6 UC. Section 7 reviews power system stability studies. Section 8 System State Estimation (SSE). Section 9 reviews unique formulations and studies. Section 10 reviews gird and cloud computation applications of classical problems. Section 11 reviews smart grid cloud-based applications. In these studies, novel approaches and algorithms and modifications to existing complex models are made and parallelized to achieve tractability, or their processing time is reduced below that needed for real-time applications. Many studies showed promising outcomes and made a strong case for further opportunities in using complex system models on HPC and Cloud for operational applications. Section 12 highlights and discusses the present challenges in the studies, re-projects the future of HPC in power systems and energy markets, and recommends a framework for future studies. Section 13 concludes the review.

Table 1. Previous work and related reviews.

Paper	**[44]**	**[45]**	**[46]**	**[47]**	**[48]**	**[49]**	**This Work**
Year	1992	1993	1996	1996	2011	2013	2022
Focus	AR [1]	DS [2]	AR	OA [3]	AR	SG [4]	AR
Historical Overview	•	•	•				•
Comprehensive Review	•		•		•	•	•
Leading edge (2013–2022)							•
Critical Review							•
HPC Tutorial			•	•		•	•
Cloud/Smart Grid						•	•
RE and Net-Zero							•
Guideline and Recommendations							•

[1] All-round; [2] Dynamic Simulation; [3] Online Applications; [4] Smart Grid.

This review does not include machine learning or meta-heuristic parallel applications such as particle swarm and genetic algorithms. Furthermore, while it does include some studies related to the co-optimization of Transmission and Distribution systems, the study excludes parallel analysis and simulations of distribution systems. It is also important to note that this review does not include Transmission and generation system planning problems or models related to grid transitioning because of the lack of parallel or HPC application in the literature on such models, which is addressed in the discussion section.

2. Parallel Hardware

Parallel computation involves several tasks being performed simultaneously by multiple workers on a parallel machine. Here, a worker is a loose term and could refer to different processing hardware (e.g., a core, Central Processing Unit (CPU), or a compute node). Predominantly, parallel machines can be placed under two categories based on The Von Neumann architecture [50], MIMD, and SIMD machines, as shown in Figure 1.

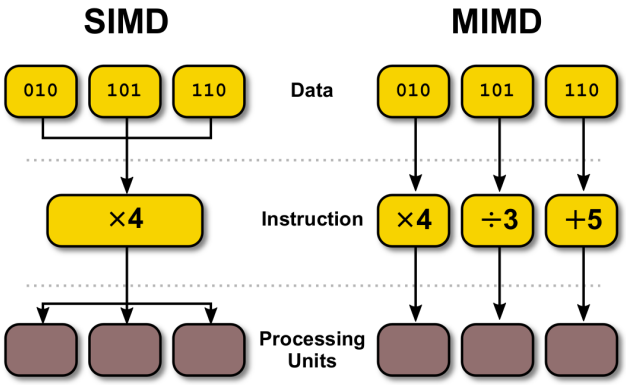

Figure 1. MIMD is versatile, allowing multiple heterogeneous instructions to be carried out on a pool of data. SIMD is simple and performs the same operation on a pool of data. Instructions are not limited to arithmetics and can take different forms (logical, transfer, etc.).

SIMD architecture dominated supercomputing with vector processors, but that changed soon after general-purpose processors became scalable in the 1990s [51,52]. Followed by transputers and microprocessors designed specifically for aggregation and scalability [53]. The effect of this shift in architecture on the algorithm design can be observed in the coinciding studies mentioned in this paper. The following subsection details the function of the present default computer hardware.

2.1. CPUs

CPUs were initially optimized for scalar programming and executing complex logical tasks effectively until they hit a power wall, leading to multicore architecture [54]. Today, they function as miniature superscalar computers that enable pipelining, task-level parallelism, and multi-threading [55]. They employ a variety of self-optimizing techniques, such as "Speculation", "Hyperthreading or "Simultaneous Multi-threading", "Auto vectorization", and "task dependency detection" [55]. They contain an extra SIMD layer that supports data-level parallelism, vectorization, and fused multiply-add with high register capacity [56,57]. Furthermore, CPUs use a hierarchy of memory and caches, as shown in Figure 2, which allows complex operations without Random Access Memory (RAM) fetching, from high-speed low-capacity (L1) to lower-speed, higher-capacity caches (L2 then L3). They give the CPU a distinct functional advantage over GPUs.

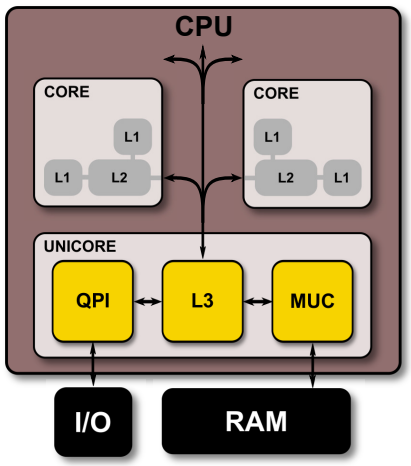

Figure 2. Architecture illustration resembles a generic CPU from Intel Xeon series [58] showing the interconnection of Quick Path Interconnect (QPI), Memory Control Unit (MCU), with caches and RAM.

A Workstation CPU can have up to 16 processing cores, and server-level CPUs can have up to 128 cores in certain products [59]. Multi-threading is carried out on Application Programming Interfaces (API) such as Cilk or OpenMP, allowing parallelism of functions and loops. Using several server-level CPUs in multi-processing to solve massive decomposed problems is facilitated by APIs such as MPI.

2.2. GPUs

GPUs function very similarly to Vector Processing Units or Array processors, which used to dominate supercomputer design. They are additional components to a "Host" machine that sends kernels, which is essentially the CPU. GPUs were originally designed to render 3D graphics. They are especially good at vector calculations. The representation of 3D graphics has a "grid" nature and requires the same process for a vast number of input data points. This execution has been extended to many applications in scientific computing and machine learning, solving massive symmetrical problems or performing symmetrical tasks. Unlike CPUs, achieving efficiency in GPUs parallelism is a more tedious task due to

the fine-grained SIMD nature and rigid architecture. Figure 3 shows a simple breakdown of the GPU instruction cycle.

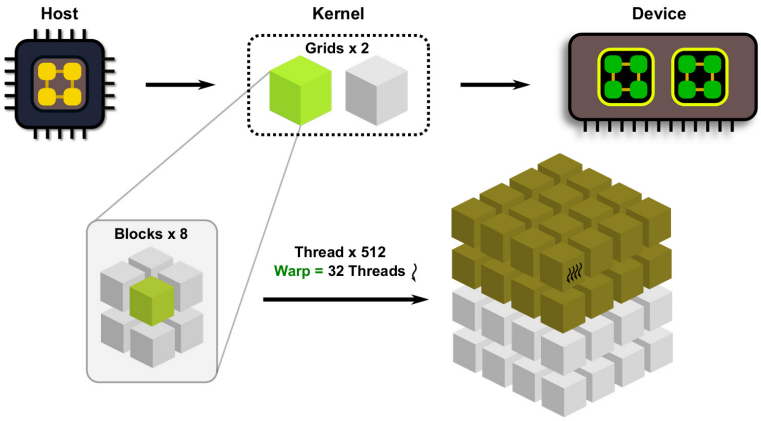

Figure 3. GPU instruction cycle and Compute Unified Device Architecture (CUDA) abstractions.

The GPU (Device) interfaces with the CPU (Host) through PCI express bus from which it receives instruction "Kernels". In each cycle, a Kernel function is sent and processed by vast amounts of GPU threads with limited communication between them. Thus, the symmetry of the parallelized task is a requisite, and the number of parallel threads has to be of specific multiple factors to avoid the sequential execution of tasks. Specifically, they need to be executed in multiples of 32 threads (a warp) and multiples of two streaming processors per block for the highest efficiencies.

GPUs can be programmed in C or C++. However, many APIs exist to program GPUs, such as OpenCL, HIP, C++ AMP, DirectCompute, and OpenACC. These APIs provide high-level functions, instructions, and hardware abstractions, making GPU utilization more accessible. The most relevant interface is the CUDA by NVIDIA since it dominates the GPU market in desktop and HPC/Cloud [60]. CUDAs libraries make NVIDIAs GPU's power much more accessible to the scientific and engineering communities.

GPU's different architecture may cause discrepancy and lower accuracy in results, as floating points are often rounded in a different manner and precision than in CPUs [61]. Nevertheless, these challenges can be worked around with CUDA and sparse techniques that reduce the number of ALUs required to achieve a massive speedup. Finally, GPUs can offer a huge advantage over CPUs in terms of energy efficiency and cost if their resources are used effectively and appropriately.

2.3. Other Hardware

There are two more notable parallel devices to mention. One is the Field Programmable Gate Arrays (FPGA). This chip consists of configurable logic blocks, allowing the user to have complete flexibility in programming the internal hardware and architecture of the chip itself. They are attractive as they are parallel, and their logic can be optimized for desired parallel processes. However, they consume a considerable amount of power compared to other devices, such as the Advanced RISC Machine. Those are processors that consume very little energy due to their reduced instruction set, making them suitable for portable devices and applications [62].

3. Aggregation and Paradigms

In the late 1970s, project ARPANET took place [63] UNIX was developed [64], and advancement in networking and communication hardware was achieved. The first commercial LAN clustering system/adaptor, ARCNET, was released in 1977 [65], and hardware abstraction sprung in the form of virtual memory, such as OpenVM, which was adopted by operating systems and supercomputers [66]. Around that same time, the concept of computer clusters was forming. Many research facilities and customers of commercial

supercomputers started developing their in-house clusters of more than one supercomputer. Today's HPC facilities are highly scalable and are comprised of specialized aggregate hardware, as displayed in Figure 4. The communication between processes through aggregate hardware is aided by high-level software such as MPI, which is available in various implementations and packages such as Mpi4py in python or Apache, Slurm, and mrjob, to aid in data management, job scheduling, and other routines.

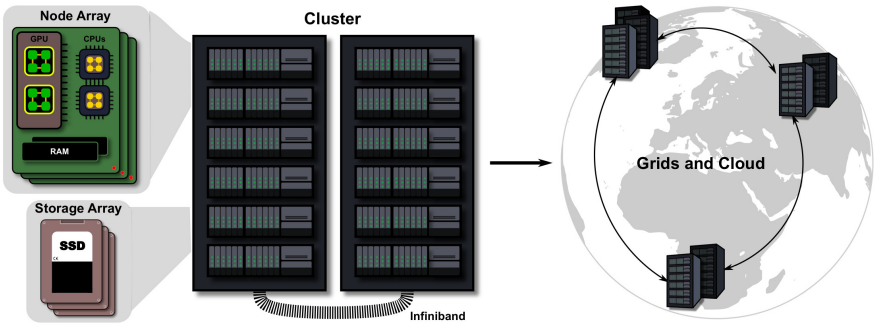

Figure 4. Powerful chips to Cloud computing.

Specific clusters might be designed or equipped with components geared more toward specific computing needs or paradigms. HPC usually includes tasks with rigid time constraints (minutes to days or maybe weeks) that require a large amount of computation. The High-Throughput Computing (HTC) paradigm involves long-term tasks that require a large amount of computation (months to years) [67]. The Many Task Computing (MTC) paradigm involves computing various distinct HPC tasks and revolves around applications that require a high level of communication and data management [68]. Grid or Cloud facilities provide the flexibility to adopt all the mentioned paradigms.

3.1. Grid Computing

The information age spurring in the 1990s set off the trend of wide-area distributed computing and "Grid Computing", the predecessor of the Cloud. Ian Foster coined the term with Carl Kesselman and Steve Tuecke, who developed the Globus toolkit that provides grid computing solutions [69]. Many Grid organizations exist today, such as Organizations such as NASA 3-EGEE and Open Science Grid. Grid computing shaped the field of "Metacomputing", which revolves around the decentralized management and coordination of Grid resources, often carried out by virtual organizations with malleable boundaries and responsibilities. The infrastructure of grids tends to be very secure and reliable, with an exclusive network of users (usually scientists and experts), discouraging virtualization and interactive applications. Hardware is not available on demand; thus, it is only suitable for sensitive, close-ended, non-urgent applications. Grid computing features provenance performance monitoring and is mainly adopted by research organizations.

3.2. Cloud Computing

Cloud computing is essentially the commercialization and effective scaling of Grid Computing driven by demand, and it is all about the scalability of computational resources for the masses. It mainly started with Amazon's demand for computational resources for its e-commerce activities, which precipitated Amazon to start the first successful infrastructure as a service-providing platform with Elastic Compute Cloud [70] for other businesses that conduct similar activities.

The distinction between Cloud and Grid is an implication of their business model. Cloud computing is way more flexible and versatile than Grid when it comes to accommodating different customers and applications. It relies heavily on virtualization and resource sharing. This makes Cloud inherently less secure, less efficient in performance than Grid, and more challenging to manage, yet way more scalable, on-demand, and

overall more resource efficient. It achieves a delicate balance between efficiency and the cost of computation.

Today, AWS, Microsoft Azure, Oracle Cloud, Google Cloud, and many other cloud commercial services provide massive computational resources for various companies such as Netflix, Airbnb, ESPN, HSBC, GE, Shell, and the NSA. It only makes sense that the electrical industry will adopt the Cloud.

3.2.1. Virtualization

The appearance of virtualization caused a considerable leap in massive parallel computing, especially after the software tool Parallel Virtual Machines (PVM) [71] was created in 1989. Since then, tens and hundreds of virtualization platforms have been developed, and are used today on the smallest devices with processing power [72]. Virtualization allows resources to be shared in a pool, where multiple instances of different types of hardware can be emulated on the same metal. This means less hardware can be allocated or invested in Cloud computing for a more extensive user base. Often, the percentage of hardware used is low compared to the requested hardware. Idle hardware is reallocated to other user processes that need it. The instances initiated by users float on the hardware, such as clouds shifting and moving or shrinking and expanding depending on the actual need of the process.

3.2.2. Containers

While virtualization makes hardware processes portable, containers make software portable. Developing applications, software, or programs in containers allows them to be used on any Operating System (OS) as long as it supports container engines. That means one can develop a Linux-based software (e.g., that works on Ubuntu 20.04) in a container and run that same application on a machine with Windows OS or iOS installed. This flexibility applies to service-based applications that utilize HPC facilities. An application can be developed on containers, and clients can use it on their cluster or a cloud service. Figure 5 compares virtual machines with container layers. While this shows that a host OS is required, they can also run on bare metal, removing latency and development complexity. Docker and Apptainer (formerly known as singularity) are commonly used containerization engines in Cloud and Grid, respectively [73,74].

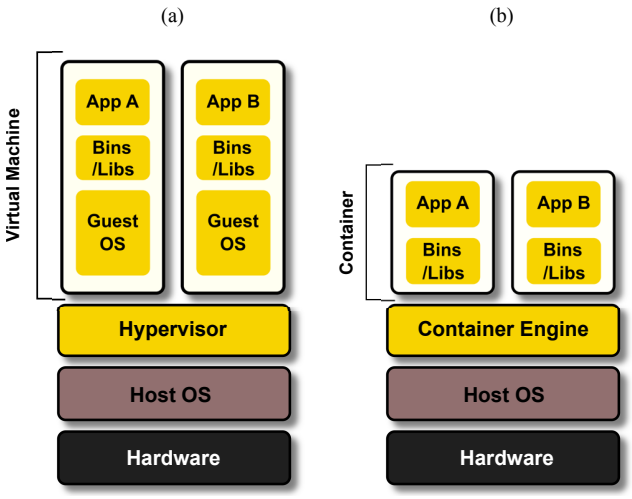

Figure 5. A comparison between Virtual Machine layers (**a**), Container Layers (**b**).

3.2.3. Fog Computing

Cloudlets, edge nodes, and edge computing are all related to an emerging IoT trend, Fog Computing. Fogs are computed nodes associated with a cloud that are geographically closer to the end-user or control devices. Fogs mediate between extensive data or cloud

computing centers and users. This topology aims to achieve data locality, offering several advantages, such as low latency, higher efficiency, and decentralized computation.

3.3. Volunteer Computing

Volunteer computing is an interesting distributed computing model that originated in 1996 via a Great Internet Mersenne Prime Search [75], allowing individuals connected to the internet to donate their personal computer's idle resources for a scientific research computing task. Volunteer computing remains active today, with many users and various middleware and projects, both scientific and non-scientific, primarily based on BOINC [76], and in commercial services such as macOS Server Resources [77].

3.4. Granularity

Fine-grained parallelism appears in algorithms that frequently repeat a simple homogeneous operation on a vast dataset. They are often associated with embarrassingly parallel problems. The problems can be divided into many highly, if not wholly symmetrical simple tasks, providing high throughput. Fine-grained algorithms are also often associated with multi-threading and shared memory resources. Coarse-grained algorithms imply moderate or low task parallelism that sometimes involves heterogeneous operations. Today, coarse-grained algorithms are almost synonymous with Multi-Processing, where the algorithms use distributed memory resources to divide tasks into different processors or logical CPU cores.

3.5. Centralized vs. Decentralized

Centralized algorithms refer to problems with a single task or objective function, solved by a single processor, with data stored at a single location. When a centralized problem is decomposed into N subproblems, sent to N number of processors to be solved, and retrieved by the central controller to update variables, re-iterate, and verify convergence, the algorithm becomes a "Distributed" algorithm. The terms distributed and decentralized are often used interchangeably and are often confused in the literature. There is an important distinction to make between them. A Decentralized Algorithm is one in which the decomposed subproblems do not have a central coordinator or a master problem. Instead, the processes responsible for the subproblems communicate with neighboring processes to reach a solution consensus (several local subproblems with coupling variables where subproblems communicate without a central coordinator). The value of each type is not only determined by computational performance but the decision-making policy.

In large-scale complex problems, distributed algorithms sometimes outperform centralized algorithms. The speedup keeps growing with the problem size if the problem has "strong scalability". Distributed algorithms' subproblems share many global variables. This means a higher communication frequency, as all the variables need to be communicated back and forth to the central coordinator. Moreover, in some real-life problems, central coordination of distributed computation might not be possible. Fully decentralized algorithms solve this problem as their processes communicate laterally, and only neighboring processes have shared variables. Figure 6 illustrates the three schemes.

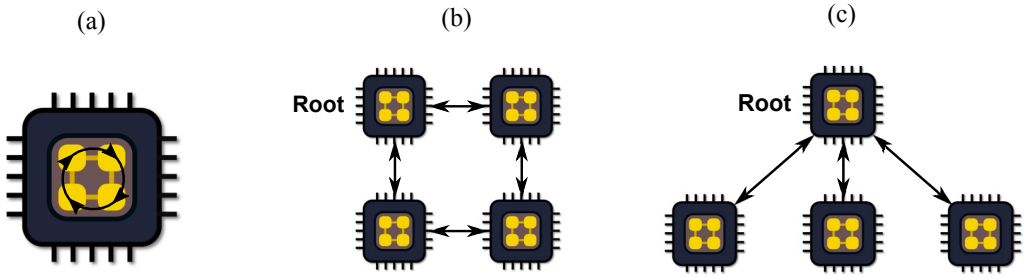

Figure 6. Information exchange between processors in different parallel schemes, (**a**) Centralized, (**b**) Decentralized, (**c**) Distributed. The arrow points in the direction of communication.

3.6. Synchronous vs. Asynchronous

Synchronous algorithms are ones in which the algorithm does not move forward until the parallel tasks at a certain step or iteration are executed. Synchronous algorithms are more accurate and efficient for tasks with symmetrical data and complexity. However, that is usually not the case in power system optimization studies. The efficiency of these algorithms suffers, however, when the tasks are not symmetrical. Asynchronous algorithms allow idling workers to take on new tasks, even if not all the adjacent processes are complete. This is possible only at the cost of accuracy when there are dependencies between parallel tasks. To achieve better accuracy in asynchronous algorithms, "Formation" needs to be ensured, meaning that while subproblems may have a deviation in the direction of convergence, they should keep a global tendency toward the solution.

3.7. Problem Splitting and Task Scheduling

Large emphasis must be placed on task scheduling when designing parallel algorithms. In multi-threading, synchronization of tasks is required to avoid "Race Conditions" that cause numerical errors due to multiple threads accessing the same memory location simultaneously. Hence, synchronization does not necessarily imply that processes will execute every instruction simultaneously but rather in a coordinated manner. Coordination mechanisms involve pipe-lining or task overlapping, which can increase efficiency and reduce the latency of parallel performance. For example, sub-tasks that take the longest time in synchronous algorithms can utilize idle workers of completed sub-tasks if no dependencies prevent such allocation. Dependency analysis is occasionally carried out when splitting tasks. In an elaborate parallel framework, such as in multi-domain simulations or smart grid applications, task scheduling becomes its own complex optimization problem, which is often solved heuristically. However, there exist packages such as DASK [78], which can help with optimal task planning and parallel task scheduling, as shown in Figure 7. DASK is a python library for parallel computing which allows easy task planning and parallel task scheduling optimization. The boxes in Figure 7 represent data and circles represent operations.

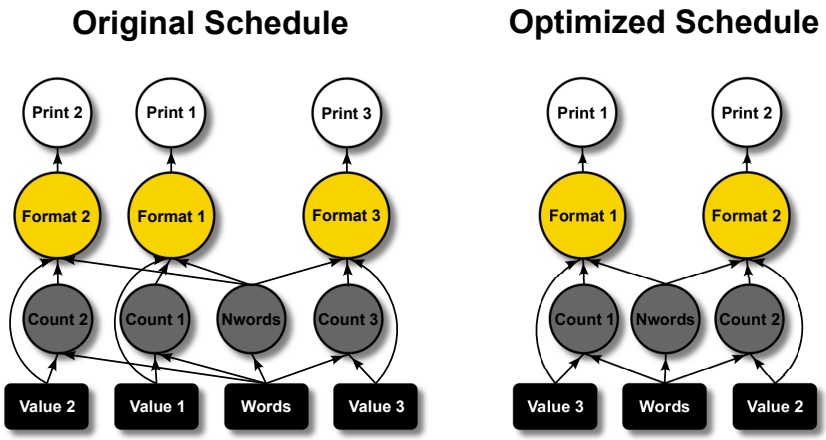

Figure 7. Typical task graph generated and optimized by DASK [78].

3.8. Parallel Performance Metrics

Solution time and accuracy are the main measures of the success of the parallel algorithm. According to the Amhals law of strong scaling, there is an upper limit to the speedup achieved for a fixed-size problem. Dividing a specific fixed-size problem into more subproblems does not result in a linear speedup. However, if the parallel portion of the algorithm increases, then proportionally increasing the subproblems or the number of processors could continuously increase the speedup, according to Gustafson's Law of strong scaling. The good news is that Gustafson's law applies to large decomposed power system problems.

There are three types of metrics most frequently used in the literature, as shown in Equation (1)–(3).

$$\text{Speedup} = \frac{\text{Serial Latency}}{\text{Parallel Latency}} \tag{1}$$

$$\text{Efficiency} = \frac{\text{Speedup}}{\text{No. of Workers}} \tag{2}$$

$$\text{Scalability} = \frac{\text{N worker latency}}{\text{1 workers latency}} \tag{3}$$

A Linear Speedup is considered optimal, while a sublinear speedup is a norm because there is always a serial portion in a parallel code. Efficiency and scalability are vaguely related. Efficiency is mostly used to compare a specific parallel setup to a sequential one, whereas scalability is used to see how the parallel algorithm scales with increased hardware. A scalable algorithm is not necessarily an efficient one. When creating a parallel algorithm, emphasis on the quality of the serial portion of the algorithm must be ensured. A parallel algorithm, after all, is contained and executed by a serial code. Both sequential and parallel programs are vulnerable to major random errors caused by the Cosmic Ray Effect [79], which has been known to cause terrestrial computational problems [80]. Soft errors might be of some concern regarding real-time power system applications. However, in parallel programming, especially in multi-processing, reordering floating-point computations is unavoidable; thus, a tiny deviation in accuracy from the sequential counterpart is expected and should be tolerated, given that the speedup achieved justifies it. All the computing paradigms mentioned above are points to tweak and consider when creating and applying any parallel algorithm.

4. Power Flow

Power Flow (PF) studies are central to all power system studies involving network constraints. The principal goal of PF is to solve the network's bus voltages and angles to calculate the flows of the network. For some applications, PF is solved using DC power flow equations, approximations based on realistic assumptions. Solving these equations is easy and relatively fast and results in an excellent approximation of the network PF [29]. On the other hand, non-linear full AC Power flow equations need to be solved to obtain an accurate solution, and these require numerical approximation methods. The most popular ones in power system analysis are the Newton Raphson (NR) method, and the Interior Point Method (IPM) [37]. However, these methods are computationally expensive and too slow for real-time applications, making them a target for parallel execution.

4.1. MIMD Based Studies

Table 2 at the end of this section summarizes contributions of MIMD-based PF studies at the end of it. Parallelism in PF in some earlier studies was achieved by restructuring the hardware to suit the algorithm. In what might be the first parallel power flow implementation, Taoka H. et al. designed their multi-processor system around the Gauss Iterative method in 1981, such that each i8085 processor in the system would solve the power flow for each bus [81]. Instead of modifying the algorithm, the hardware was tailored to it, achieving a linear speedup compared to the sequential implementation in [82]. Similarly, one year later, S. Chan and J. Faizo [83] reprogrammed the CDC supercomputer to solve the Fast-Decoupled Power Flow (FDPF) algorithm in parallel. FPGAs were also used to parallelize LU decomposition for PF [84,85]. The hardware modification approach, while effective, is, for the most part, impractical, and it is common sense to modify algorithms to fit existing scalable hardware.

The other way to parallelize PF (or OPF) is through network partitioning. While network partitioning usually occurs at the problem level in OPF, in PF, the partitioning often happens at the solution/matrix level. Such partitioning methods for PF use sparsity techniques involving LU decomposition, forward/backward substitution, and diakoptics

that trace back to the late 1960s, predominantly by H. H. Happ [86,87] for load flow on typical systems [88,89], and dense systems [41]. Parallel implementation of PF using this method started in the 1980s on array processors such as the VAX11 [90] and later in the 1990s on the iPSC hypercube [91]. Techniques such as FDPF were also parallelized on the iPSC using Successive Over-relaxation (SOR) on Gauss-Sidel (GS) [92], and on vector computers such as the Cray, X/MP using Newtons FDPF [93]. PF can also be treated as an unconstrained non-linear minimization problem, which is precisely what E. Housos and O. Wing [94] did to solve it using a parallelizable modified conjugate directions method.

When general processors started dominating parallel computers, their architecture was homogenized, and the enhancements achieved by parallel algorithms became comparable and easier to experiment with. This enabled a new target of optimizing the parallel techniques themselves. Chen and Chen used transputer-based clusters to test the best workload/node distribution on clusters, and [95] and a novel BBDF approach for power flow analysis [96]. The advent of Message Passing Interface (MPI) allowed the exploration of scalability with the Generalized Minimal Residual Method (GMRES) in [97] and the multi-port inversed matrix method [98] as opposed to the direct LU method. Beyond this point, parallel PF shifted heavily towards using SIMD hardware (GPUs particularly), except for a few studies involving elaborate schemes. Some examples include transmission/distribution, smart grid PF calculation [99], or Optimal network partitioning for fast PF calculation [100].

Table 2. Power flow MIMD-based state-of-the-art studies.

Paper	Contribution
[81]	Gauss Iterative method on a specialized machine
[82]	Tailored parallel hardware for FDPF
[83]	Reprogrammed CDC supercomputer to parallelize FDPF
[84,85]	Reprogrammed FPGA to parallelize LU method
[90]	LU method on VAX11
[91]	LU method on iPSC hypercube
[92]	FDPF using SOR on Gauss–Siedel on IPSC hypercube
[93]	Newton FDPF on Cray X/MP
[94]	Parallel conjugate directions method
[95]	Novel BBDF on transputer-based cluster
[97]	MPI scheme parallelizing GMRES
[98]	MPI scheme parallelizing multi-port inverse matrix method

4.2. SIMD Based Studies

4.2.1. Development

GPU dominates recent parallel power system studies. The first power flow study implementation might have been achieved by using a preconditioner Biconjugate Gradient Algorithm and sparsity techniques to implement the NR method on a NVIDIA Tesla C870 GPU [101]. Some elementary approaches parallelized the computation of connection matrices for networks where more than one generator could exist on a bus on a NVIDIA GPU [102]. CPUs were also used in SIMD-based power flow studies since modern CPUs exhibit multiple cores; hence, multi-threading with OpenMP can be used to vectorize NR with LU factorization [103]. Some resorted to GPUs to solve massive batches of PF for Probabilistic Power Flow (PPF) or contingency analysis, thread per scenario, such as in [104]. Others modified the power flow equations to improve the suitability and performance on GPU [105,106].

While many papers limit their applications to NVIDIA GPUs by using CUDA, OpenCL, a general parallel hardware API, has also been used occasionally [107]. Some experimented with and compared the performance of different CUDA routines on different NIVIDIA GPU models [108]. Similar experimentation on routines was conducted to solve ACOPF using FDPF [109]. In [110], NR, Gauss Sidel, and Decoupled PF were tested and compared against each other on GPU. Improvement on the Newtons Method and parallelizing different steps

of it were performed previously [111]. Asynchronous PF algorithms were applied on GPU, which sounds difficult, as the efficiency of GPUs depends on synchronicity and hegemony [112]. Even with the existence of CUDA, many still venture into creating their routines with OpenCL [113,114] or direct C coding [115] of GPU hardware to fit their needs for PF. Very recently, a few authors made thorough overviews for parallel power flow algorithms on GPU covering general trends [116,117] and specifically AC power flow GPU algorithms [118]. In the State-of-the-Art subsection, the most impactful work is covered.

4.2.2. State-of-the-Art

Table 3 summarizes the latest SIMD-based PF studies contributions. A lot of the recent work in this area focuses on pre-conditioning and fixing ill-conditioning issues in iterative algorithms to solve the created Sparse Linear Systems (SLS). An ill-conditioned problem exhibits a massive change in output for a minimal change in input, making the solution to the problem hard to find iteratively. Most sequential algorithms are LU-based direct solvers, as they do not suffer from ill-conditioning. However, Iterative solvers such as the Conjugate Gradient method, which have been around since the 1990s [119], are regaining traction for their scalability and parallel computing advancement.

The DC Power Flow (DCPF) problem was solved using the Chebyshev preconditioner and conjugate gradient iterative method in a GPU (448 cores Tesla M2070) implementation in [120,121]. The vector processes involved are easily parallelizable in the most efficient way with CUDA libraries such as CUBLAS and CUSPARSE, which are Basic Linear Algebra Subroutine and Sparse Matrix Operation Libraries. This work used comparisons of sparse matrix storage formats, such as the Compressed Sparse Row (CSR) and Block Compressed Sparse Row (BSR). On the largest system size, a speedup of 46× for the pre-conditioning step and 10.79× for the conjugate gradient step was achieved compared to a sequential Matlab implementation (8-core CPU).

Later, the same author went on to Parallelize the FDPF using the same hardware and pre-conditioning steps [122]. Two natural systems were used, the Polish system, which had groups of locally connected systems, and the Pan-European system, which consisted of several large coupled systems. This topology difference results in a difference in the sparsity patterns of the SLS matrix, which offers a unique perspective. Their proposed GPU-based FDPF was implemented with Matlab on top of MatPower v4.1. In their algorithm, the CPU regularly corresponds with the GPU, sending information back and forth over one iteration. Their tests showed that the FDPF performed better on the Pan-European system because its connections were more ordered than the Polish system. CPU-GPU communication frequently occurred in their algorithm steps, most likely bottlenecking the speedup of their algorithm (less than 3× achieved compared to CPU only).

Instead of adding pre-conditioning steps, M. Wang et al. [123] focus on improving the continuous Newtons method such that a stable solution is found even for an ill-conditioned power flow problem. For example, if any load or generator power exceeds 3.2 p.u. in the IEEE-118 test case, the NR method fails to converge; even if the value is realized in any iteration, their algorithm will still converge to the solution with their method. This was achieved using different-order numerical integration methods. The CPU loads data into GPU and extracts the results upon convergence only, making the algorithm very efficient. The approach substantially improved over the previous work by removing the pre-conditioning step and reducing CPU–GPU communication (speedup of 11× compared to CPU-only implementation).

Sometimes, dividing the bulk of computational load between the CPU and GPU (a hybrid approach) can be more effective depending on the distribution of processes. In one hybrid CPU–GPU approach, a heavy emphasis on the sparsity analysis of PF-generated matrices was made in [124]. When using a sparse technique, the matrices operated on are reduced to ignore the zero terms. For example, the matrix is turned into a vector of indices referring to the non-zero values to confine operations to these values. Seven parallelization schemes were compared, varying the techniques used (Dense vs. Sparse

treatment), the majoring type (row vs. column), and the threading strategy. Row/column-major signifies whether the matrix's same row/column data are stored consecutively. The thread invocation strategies varied in splitting or combining the calculation of P and Q of the mismatch vectors. In this work, two sparsity techniques were experimented with, showing a reduction in operations down to 0.1% of the original number and two or even three orders of magnitude performance enhancement for power mismatch vector operations. In 100 trials, their best scheme converged within six iterations on a four-core host and a GeForce GTX 950M GPU, with a small deviation in solution time between trials. CPU–GPU communication took about 7.79–10.6% of the time, a fairly low frequency. However, the proposed approach did not consistently outperform a CPU-based solution with all of these reductions. The authors suggested that this was due to using higher-grade CPU hardware than the GPU.

Zhou et al. might have conducted the most extensive research in GPU-accelerated batch solvers in a series between 2014 and 2020. They fine-tune the process of solving PPF for GPU architecture in [125,126]. The strategies used include Jacobian matrix packaging, contiguous memory addresses, and thread block assignment to avoid divergence of the solution. Subsequently, they use the LU-factorization solver from previous work to finally create a batch-DPF algorithm [127]. They test their batch-DPF algorithm on three cases: 1354-bus, 3375-bus, and 9241-bus systems. For 10,000 scenarios, they solved the largest case within less than 13 s, showing the potential for online application.

Most of the previous studies solve the PF problem in a bare and limited setup when compared to the work by J. Kardos et al. [128] that involves similar techniques in a massive HPC framework. Namely, preventative Security Constrained Optimal Power Flow (SCOPF) is solved by building on an already existing suite called BELTISTOS [129]. BELTISTOS specifically includes SCOPF solvers and has an established Schurs Complement Algorithm that factorizes the Karush–Kuhn–Tucker (KKT) conditions, allowing for a great degree of parallelism in using IPM to solve general-purpose Non-Linear Programming (NLP) problems. Thus, the main contribution of this work is in removing some bottlenecks and ill-conditioning that exist in Schur's Complement steps introducing a modified framework (BELTISTOS-SC). The parallel Schur algorithm is bottlenecked by a dense matrix associated with the solution's global part. This matrix is solved in a single process. Since GPUs are meant to be used for dense systems, they factorize the system and apply forward–backward substitution, solving it using cuSolve, a GPU-accelerated library to solve dense linear algebraic systems.

Table 3. Power flow SIMD-based state-of-the-art studies.

Paper	Contribution
[120,121]	Chebyshev pre-conditioner and conjugate gradient iterative method
[122]	Parallel FDPF
[123]	Improving the continuous Newtons method
[124]	7 parallelization strategies with sparsity analysis
[125,126]	Probabalistic Power Flow parallelization
[127]	Batch decoupled power flow
[128]	Security Constrained OPF (BELTISTOS software)

They performed their experiments using a multicore Cray XC40 computer at the Swiss National Supercomputing Centre. They used 18 2.1 GHz cores, NVIDIA Tesla P100 with 16 GB memory, and many other BELTISTOS and hardware-associated libraries. They tested their modification on several system sizes from PEGASE1354 to PEGASE13659. Their approach sped up the solution of the Dense Schur Complement System by 30× for the largest system over CPU solution of that step, achieving notable speed up in all systems sizes tested. They later performed a large-scale performance study, where they increased the number of computing cores used from 16 to 1024 on the cluster. The BELTISTOS-SC augmented approach achieved up to 500× speedup for the PEGASE1354 system and 4200×

for the PEGASE9241 when 1024 cores are used, demonstrating strong scalability up to 512 cores.

5. Optimal Power Flow

Like PF, OPF studies are the basis of many operational assessments such as System Stability Analysis (SSA), UC, ED, and other market decisions [29]. Variations of these assessments include Security Constrained Economic Dispatch (SCED) and SCOPF, both involving contingencies. OPF ensures the satisfaction of network constraints over cost or power-loss minimization objectives. The full ACOPF version has non-linear, non-convex constraints, making it computationally complex and making it difficult to reach a global optimum. DC Optimal Power Flow (DCOPF) and other methods, such as decoupled OPF, linearize and simplify the problem, and when solved, they produce a fast but sub-optimal solution. Because DCOPF makes voltage and reactive power assumptions, it becomes less reliable with increased RE penetration. RE deviates voltages and reactive powers of the network significantly. This is one of the main drivers behind speeding up ACOPF in real-time applications for all algorithms involving it. The first formulation of OPF was achieved by J. Carpenter in 1962 [130], followed by an enormous volume of OPF formulations and studies, as surveyed in [131].

5.1. MIMD Based Studies

5.1.1. Development

OPF and SCOPF decomposition approaches started appearing in the early 1980s using P-Q decomposition [132,133] and including corrective rescheduling [134]. The first introduction to parallel OPF algorithms might have been by Garng M Huang, and Shih-Chieh Hsieh in 1992 [135], who proposed a "textured" OPF algorithm that involved network partitioning. In a different work, they proved that their algorithm would converge to a stationary point and that with certain conditions, optimality is guaranteed. Later, they implemented the algorithm on the nCUBE2 machine [136], showing that both their sequential and parallel textured algorithm is superior to non-textured algorithms. It was atypical for studies at the time to highlight portability, which makes Huang's work in [92] special. It contributed another OPF algorithm using Successive Overrelaxation by making it "Adaptive", reducing the number of iterations. The code was applied on the nCUBE2 and ported to Intel iPSC/860 hypercube, demonstrating its portability.

In 1990, M.Teixeria et al. [137] demonstrated what might be the first parallel SCOPF on a 16-CPU system developed by the Brazilian Telecom R&D center. The implementation was somewhat "makeshift" and coarse to the level where each CPU was installed with a whole MS/DOS OS for the multi-area reliability simulation. Nevertheless, it outperformed a VAX 11/780 implementation and scaled perfectly, was still 2.5 times faster than running on, and exhibited strong scalability.

Distributed OPF algorithms started appearing in the late 1990s with a coarse-grained multi-region coordination algorithm using the Auxiliary Problem Principle (APP) [138,139]. This approach was broadened much later by [140] using Semi-Definite Programming and Alternating Direction Method of Multiplier (ADMM). Prior to that, ADMM was also compared against the method of partial duality in [141]. The convergence properties of the previously mentioned techniques and more were compared comprehensively in [142].

The asynchronous parallelization of OPF first appeared on preventative [143], and corrective SCOPF [144] targeting online applications [47] motivated by the heterogeneity of solution time of different scenarios. Both SIMD and MIMD machines were used, emphasizing portability as "Getsub and Fifo" routines were carried out. On the same token, MPI protocols were used to distribute and solve SCOPF, decomposing the problem with GRMES and solving it with the non-linear IPM method varying the number of processors [145]. Real-time application potential was later demonstrated by using Benders decomposition instead for distributed SCOPF [146]. Benders decomposition is one of the most commonly used techniques to create parallel structures in power system optimization problems, and

it shows up in different variations in the present literature. As Figure 8 shows, Benders Decomposition is applied by fixing the complicating variables of the objective function to a different value in every iteration and constructing a profile with benders cuts to find the minimum objective function value with respect to the complicating variables. If the profile is non-convex, then optimality is not guaranteed since the value is changed in steps descending the slope of the cuts.

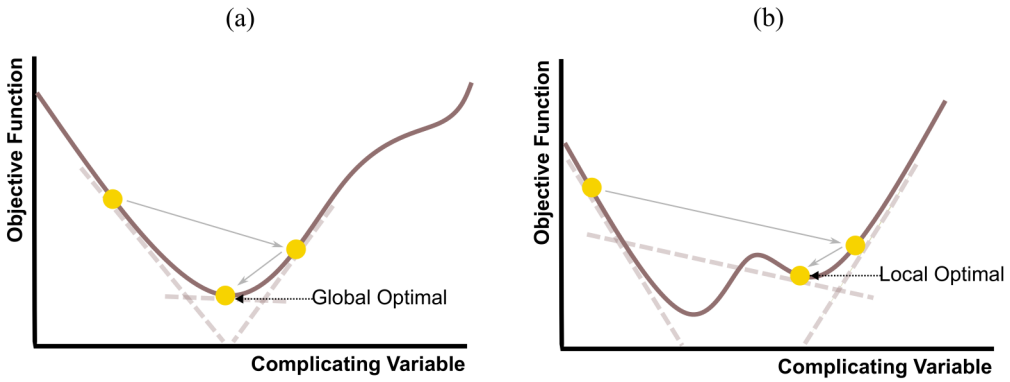

Figure 8. The conversion steps of the complicated variable when it is Convex (**a**) and Non-Convex (**b**) with respect to the objective function.

5.1.2. State-of-the-Art

Table 4 summarizes the contributions of MIMD-based OPF studies of this section. As mentioned, involving AC equations in large-scale power system studies is crucial and might soon become the standard. The difficulty of achieving this feat varies depending on the application. For example, precise nodal price estimation is attained by solving ACOPF multiple times under different possible system states (Probabilistic ACOPF). This can be effortlessly scaled as each proposed scenario can be solved independently, but a large number of scenarios can exhaust available resources. This is when researchers resort to scenario reduction techniques such as Two-Point Estimation [147]. In this study, it was applied to 10k Monte Carlo (MC) scenarios, and the reduced set was used to solve a conically relaxed ACOPF model following the approach in [148]. Using 40 cores from the HPC cluster of KTH Royal Inst of Tech, the approach resulted in an almost linear speedup with high accuracy on all test cases.

In contrast, parallelizing a monolithic ACOPF problem itself is much more complicated. However, the same authors did this readily since their model was already decomposable due to the conic relaxation [149]. Here, the choice of network partitions is treated as an optimization problem to realize the least number of lines between sub-networks. A graph partitioning algorithm and a modified benders decomposition approach were used, providing analytical and numerical proof that they converge to the same value as the original benders. This approach achieved a lower–upper bound gap of around 0–2%, demonstrating scalability. A maximum number of eight partitions (eight subproblems) were divided on a four-core 2.4 GHz workstation. Beyond four partitions, hyperthreading or sequential execution must have occurred. This is a shortcoming, as only four threads can genuinely run in parallel at each time. Hyper-threading only allows a core to alternate between two tasks. Their algorithm might have even more potential if distributed on an HPC platform.

The ACOPF formulation is further coupled and complicated when considering Optimal Transmission Switching (OTS). The addition of binary variables ensures the non-convexity of the problem, turning it from an NLP to an Mixed Integer Non-Linear Programming (MINLP). In Lan et al. [150], this formulation is parallelized for battery storage embedded systems, where temporal decomposition was performed, recording the State of Charge (SoC) at the end of each 6 h (four subproblems). They employed a two-stage scheme with an NLP first stage to find the ACOPF of a 24 h time horizon and transmission switching in the second stage. The recorded SOCs of the first stage are added as constraints

to the corresponding subproblems, which are entirely separable. They tested the algorithm IEEE-188 test case and solved it with Bonmin with GAMS on a four-core workstation. While the coupled ACOPF-OTS formulation achieved a 4.6% Optimality gap at a 16 h 41 min time limit, their scheme converged to a similar gap within 24 m. The result is impressive, considering the granularity of the decomposition. This is yet another example in which a better test platform could have shown more exciting results, as the authors were limited to parallelizing four subproblems. Algorithm-wise, an asynchronous approach or better partition strategy is needed, as one of the subproblems took double the time of all the others to solve.

The inclusion of voltage and reactive power predicate the benefits gained by ACOPF. However, it is their effect on the optimal solution that matters, and there are ways to preserve that while linearizing the ACOPF. The DCOPF model is turned into a Mixed Integer Linear Programming (MILP) in [151] by adding on/off series reactance controllers (RCs) to the model. The effect of the reactance is implied by approximating its value and adding it to the DC power flow term as a constant without actually modeling reactive power. The binary variables are relaxed using the Big M approach to linearize the problem, derive the first-order KKT conditions, and solve it using the decentralized iterative approach. Each node solved its subproblem, making this a fine-grained algorithm, and each subproblem had coupling variables with adjacent buses only. The approach promised scalability, and its convergence was proven in [152]. However, it was not implemented in parallel, and the simulation-based assumptions are debatable.

Decentralization, in that manner, reduces the number of coupling variables and communication overhead. However, this also depends on the topology of the network, as shown in [111]. In this work, a stochastic DCOPF formulation incorporating demand response was introduced. The model network was decomposed using ADMM and different partitioning strategies where limited information exchange occurs between adjacent subsystems. The strategies were implemented using MATLAB and CPLEX on a six-bus to verify solution accuracy and later on larger systems. The ADMMBased Fully Decentralized DCOPF and Accelerated ADMM for Distributed DCOPF were compared. Recent surveys on Distributed OPF algorithms showed that in OPF decomposition and parallelization, ADMM and APP are preferred in most of the studies as a decomposition technique [149,153]. The distributed version converged faster, while the decentralized version exhibited better communication efficiency. More importantly, a separate test showed that decentralized algorithms work better on subsystems that exhibit less coupling (are less interconnected) and vice versa. This breeds the idea that decentralized algorithms are better suited for ring or radial network topologies while distributed algorithms are better for meshed networks [153,154].

Table 4. Optimal Power Flow MIMD based state-of-the-art studies.

Paper	Contribution
[147]	Two-Point estimation for scenario reduction (Monte Carlo Solves)
[148]	Conically Relaxed ACOPF (Monte Carlo Solves)
[149]	Monolithic ACOPF model parallelization using conic relaxation
[150]	Battery storage and Transmission Switching with Temporal Decomposition
[111]	Comparing decentralized and distributed algorithms on different network topologies
[155]	Transmission-Distribution network co-optimization,
[99,156]	Two-stage stochastic algorithm accounting for DER

Distribution networks tend to be radial. A ring topology is rare, except in microgrids. Aside from their topology, they have many differences compared to transmission networks causing the division of their studies and OPF formulations. OPF for Transmission–Distribution co-optimization makes a great case for HPC use in power system studies, as co-optimizing the two together is considered peak complexity. S. Tu et al. [155] decomposed a very large-scale ACOPF problem in the Transmission–Distribution network co-optimization attempt. They devised a previously used approach where the whole net-

work was divided by its feeders, where each distribution network had a subproblem. The novelty in their approach lies in a smoothing technique that allows gradient-based non-linear solvers to be used, particularly the Primal-Dual-Interior Point Method, which is the most commonly used method for solving ACOPF. Similar two-stage stochastic algorithms were implemented to account for the uncertainties in Distributed Energy Resources (DER) at a simpler level [99,156].

S. Tu et al. used an augmented IEEE-118 network, adding distribution systems to all busses, resulting in 9206 buses. Their most extreme test produced 11,632,758 bus solutions (1280 scenarios). Compared to a generic sequential Primal Dual Interior Primal Method (PDIPM), the speedup of their parallelized approach increased linearly with the number of scenarios and scaled strongly by increasing the number of cores used in their cluster. In contrast, the serial solution time increased superlinearly and failed to converge within a reasonable time in a relatively trivial number of scenarios. While their approach proved to solve large-scale ACOPF much faster than a serial approach, it falls short in addressing Transmission–Distribution co-optimization because it merely considered the distribution network as sub-networks with the same objective as the transmission, which is unrealistic.

6. Unit Commitment

The UC problem goes back to the 1960s [157]. In restructured electricity markets, Security Constrained Unit Commitment (SCUC) is used to determine the generation schedule at each time point at the lowest cost possible while maintaining system security. A typical formulation used in today's industry can be found in [158]. Often, implementations use immensely detailed stochastic models involving N-1 or N-2 contingencies [159], ACOPF constraints, and incorporating RE resources DER [160] and distribution networks [161]. This leads to a large number of scenarios and a tremendously complex problem. Decomposition of the problem using Lagrangian Relaxation (LR) methods is very common [36,162] and many formulations are ready to segue for HPC parallel implementation. This includes global optimal solution methods for AC- SCUC, as in [163]. Recent literature on parallel UC is abundant, making this section the largest in this review.

6.1. MIMD-Based Studies

6.1.1. Development

Simulations of parallel environments to implement parallel UC algorithms started appearing around 1994, modeling hydrothermal systems [164] and stochasticity [165] on supercomputers [166] workstation networks [167]. The earliest UC implementations on parallel hardware used embarrassingly parallel metaheuristics, such as simulated annealing [165] and other genetic algorithms [168,169]. However, the first mathematical programming approach might have been conducted by Misra in 1994 [166] using dynamic programming and vector processors. Three years later, K.K. Lau and M.J. Kumar also used dynamic programming to create a decomposable master–slave structure of the problem. It was then distributed over a network of workstations using PVM libraries, with each subproblem solved asynchronously [167]. These, however, were not network-constrained problems.

In 2000, Murillo-Sanchez and Robert J. Thomas [170] attempted full non-linear AC-SCUC in parallel by decomposing the problem using APP but failed to produce any results, upholding the problem's complexity. In quite an interesting case, volunteer computing with the BOINC system was used to parallelize the MC simulations of stochastic load demand in UC problems with hydrothermal operation [171]. However, that did not include network constraints either. There has been some work conducted on decomposition algorithms for network-constrained UC, but this has rarely been applied in a practical parallel setup. Most parallel implementations happened in the last decade.

6.1.2. State-of-the-Art

Table 5 summarizes the latest contributions of MIMD-based UC studies of this section. Papavasillio et al. [172], set up the framework of a scenario-based two-stage stochastic framework of UC with reserve requirement for wind power integration, emphasizing wind forecasting and scenario selection methodology. In later work, Papavasiliou compared a benders decomposition approach that removes the second-stage bottlenecking and a Lagrangian Relaxation (LR) algorithm based on [162], where the impact of contingencies on decisions was implied in the constraints [160]. The LR approach proved to be more scalable for that formulation. As a result, Papavasillo chose the LR approach to solving the same formulation by adding DC network constraints [173]. Even though the wind scenarios were carefully selected, there existed instances in which specific subproblems took about double the time of the following most complex subproblem. In a follow-up work, Aravena and Papavasillio resorted to an asynchronous approach in [174] that allows time sharing. This work showed that the synchronous approach to solving subproblems could be highly inefficient, as the idle time of computational resources can reach up to 84% compared to the asynchronous algorithm.

While many SCUC parallel formulations use DC networks, the real challenge is using ACOPF, as exhibited in earlier failed attempts [170]. In [175], the conic relaxation approach mentioned earlier [148] was used to turn the AC-SCUC into a Mixed Integer Second-Order Conic (MISOC) program. It allowed the decomposition of the problem to a master problem, where UC is determined, and a subproblem in which the ACOPF is solved iteratively. In their approach, the active power is variable if the master problem is fixed, and only the reactive power is solved to check if the commitment is feasible. Fixing the active power allows for time decomposition since ramping constraints no longer apply in the subproblems. They compared the computational efficiency of their approach against a coupled DC- SCUC and AC- SCUC, solved using commercial solvers such as GAMS, DICOPT, and SBB. Their approach took only 3.3% of the time taken by previous similar work [176] to find a solution at a 0.56% gap. However, their approach faced accuracy and feasibility issues, and the parallelism strategy was unclear since they created eight subproblems while using three threads.

Temporal decomposition was also used on a unique formulation, Voltage Stability Constrained SCUC (VSC- SCUC) [177]. The problem is an MINLP with AC power flow constraints and added voltage stability constraints that use an index borrowed from [178]. APP decomposition was used to decompose the model into 24 subproblems, and it was compared against conventional AC- SCUC on several test cases. It converged after 55 iterations compared to 44 by the AC-SCUC solution on the IEEE-118 case. The structure and goals achieved by VSC- SCUC make tractability challenging, deeming the approach itself promising. However, the study fell short of mentioning any details about the claimed parallel routine used.

Nevertheless, SCUC decentralization is valued for more than performance enhancement. It can help achieve privacy and security, and it fits the general future of the smart grid, IoT, and market decentralization. In a market in which the ISO sets prices of energy and generators are merely price takers, a decentralization framework called "Self-Commitment" can be created from the UC formulation [179]. Inspired by self-commitment, Feizollahi et al. decentralize the SCUC problem relevant to bidding markets, including temporal constraints [180]. They implement a "Release-and-Fix" process which consists of three ADMM stages of decomposing the network. The first stage finds a good starting point by solving a relaxed model. The second and third stages are iterative; a feasible binary solution is found, followed by a refinement of the continuous variables. They used two test cases (3012 and 3375-bus) and applied different partitions from 20 to 200 sub-networks with different root node (coordinator node) combinations. They also varied the level of constraints in three cases, from no network up to AC network and temporal constraints. A sub 1% gap was achieved in all cases, outperforming the centralized solution in the most complicated cases and showing scalability where the centralized solution was intractable. The scalability

saturated, however, at 100 partitions, and one of the key conclusions was that the choice of the root node and partition topologies are crucial to achieving gains.

Multi-Area formulations often involve ED, but rarely UC, as seen in [181]. The UC formulation in this study includes wind generation. The wind uncertainty is incorporated using Adjustable Interval Robust Scheduling of wind power, a novel extension of Interval Optimization Algorithms, a chance-constrained algorithm similar to Robust Optimization. The resulting Mixed-Integer Quadratic Programming (MIQP) model is decentralized using an asynchronous consensus ADMM approach. They verified the solution quality on a three-area six-bus system (achieving a 0.06% gap) and then compared their model against Robust Optimization and Stochastic Optimization models on a three-area system composed of one IEEE-118 bus system each. For a lower CPU time, their model achieved a much higher level of security than the other mentioned models. The study mentions that the parallel procedure took half the time the sequential implementation did, promising scalability. However, no details were given regarding the parallel scheme used and implementation.

Similarly, Ramanan et al. employed an asynchronous consensus ADMM approach to achieve multi-area decentralization slightly differently, as their formulation is not a consensus problem [182]. Here, the algorithm is truly decentralized, as the balance in coupling variables needs only to be achieved between a region and its neighboring one. The solution approach is similar to that of the one from [172], where UC and ED are solved iteratively. They divided the IEEE 118 bus into ten regions; each region (subproblem) was solved by one intel Xeon 2.8 GHz processor. They ensure asynchronous operation by adding 0.2 s of delay for some subproblems. The mean results of the 50 runs demonstrated the time-saving and scaling potential of the asynchronous approach that was not evident in other similar studies [183]. However, the solution quality significantly varied and deteriorated, with an optimality gap reaching 10% for some runs, and no comparison was drawn against a centralized algorithm.

In later work, the authors improved the asynchronous approach by adding some mechanisms, such as solving local convex relaxations of subproblems while consensus is being established. This allowed the subproblem to move to the next solution phase if the binary solution was found to be consistent over several iterations [184]. In addition, they introduced a globally redundant constraint based on production and demand to improve privacy further. Moreover, they used point-to-point communication without compromising the decentralized structure. They implemented their improved approach on an IEEE-3012 bus divided into 75, 100, and 120 regions. A 2.8 GHz core was assigned to solve each region and controller subproblem. They compared their approach, this time against Feizollahis implementation from 2015 [180] and a centralized approach. The idle time of the synchronous approach was higher than the computation time of the Asynchronous approach, doubling the scalability for higher region subdivisions. The gap achieved in all cases was larger than that of the Centralized solution by around 1.5%, which is a huge improvement, considering the previous work and the 18x speedup achieved.

Consensus ADMM methods typically do not converge for MILP problems such as UC without a step size diminishing property [185]. Lagrangian methods, in general, are known to suffer from a zigzagging problem. To overcome the issue, the Surrogate Lagrangian Relaxation (SLR) algorithm was used in [186] to create a distributed asynchronous UC. In later work, their approach was compared against a Distributed Synchronous SLR and a sequential SLR [187] using four threads to parallelize the subproblems. With that, better scalability against the synchronous approach was demonstrated, and a significant speedup was achieved (12× speedup to achieve a 0.03% duality gap in one instance).

To avoid the same zigzagging issue, but for Multi-Area SCUC, Kargarian et al. opted for Analytical Target Cascading (ATC) since multiple options exist for choosing the penalty function in ATC [188]. They take the model from [189] and apply ATC from [190] to decompose the problem into a distributed bi-level formulation, with a central coordinator being the leader and subproblems followers. In this work, they switched the hierarchy by putting the coordinator first, making it the follower instead of the leader. This convexified

the followers' problem, allowing the use of KKT conditions, turning it into a Mixed Complementarity Problem (MCP). Those steps turned the formulation into a decentralized one, as only neighboring subproblems became coupled. They numerically demonstrated that with their reformulation, the convergence properties of ATC were still upheld virtually and that the convex quadratic penalty functions act as local convexifiers of the subproblems. Moreover, they demonstrated numerically how the decentralized algorithm is less vulnerable to cyber attacks. Unfortunately, the approach was not implemented practically in parallel; rather, the parallel solution time was estimated based on the sequential execution of the longest subproblem.

In a similar work tackling Multi-Area SCUC, a variation of ACT is used [191] in which the master problem determines the daily transmission plan, and each area becomes an isolated SCUC subproblem. This problem is much more complicated, as it involves AC power flow equations, HVDC tie-lines, and wind generation. The power injection of the tie lines is treated as a pseudo generator with generation constraints that encapsulate the line flow constraints. This approach removes the need for consistency constraints used in traditional ATC-based distributed SCUC, such as the ones used in [190]. In the case study, they subdivide several systems into three-area networks and split their work into three threads. Comparisons were drawn against a centralized implementation, the traditional distributed form, and four different tie-line modes of operation with varying load and wind generation. Their approach consistently converged at lower times than the traditional ATC algorithm. It slightly surpassed the centralized formulation on the most extensive network of 354-bus, which means the approach has the potential for scalability.

Finding the solution of a UC formulation that involves the transmission network, active distribution network, microgrids, and DER is quite a leap in total network coordination. This challenge was assumed by [161] in a multi-level interactive UC (MLI-UC). The objective function of this problem contained three parts: the cost of UC at the transmission level, the cost of dispatch at the distribution level, and the microgrid level. The three levels' network constraints were decomposed using the ATC algorithm, turning it into a multi-level problem. A few reasonable assumptions were made to aid in the tractability of the problem. The scheme creates a fine-grained structure at the microgrid level and a coarser structure at the distribution level, both of which were parallelized. The distribution of calculation and information exchange between the three levels provides more information regarding costs at each level and the Distribution of Locational Marginal Pricing (DLMP).

Most of the previous work in power system problems—apart from UC—focuses on the solution process rather than the database operations involved. In [192], a parallel SCUC formulation for hydrothermal power systems is proposed, incorporating pumped hydro. This paper uses graph computing technologies and graph databases (NoSQL) rather than relational databases to parallelize the formulation of their MIP framework. Their framework involves Convex Hull Reformulation, a Special Ordered Set method to reduce the number of variables of the model, constrained relaxation techniques [193], and LU decomposition. The graph-based approach showed significant enhancement in speedup over a conventional MIP solution method on a Tigergraph v2.51 database. Similar applications of NoSQL were explored in other power system studies [194–196].

In real industrial applications, there is a lower emphasis on the accuracy of the solution, and a high-speed "good enough" policy is adopted, often using heuristics extensively. Midwest ISO (MISO) published a few papers showing the development of their Day-Ahead DC UC decision-making in 2016 [197], 2020 [35], and 2021 [198]. Their HPC parallel approaches introduce novel strategies as part of the HIPPO project [199], focusing on finding smart heuristics to speed up SCUC decomposition and distributed methods. MISO has been using CPLEX to solve day Ahead SCUC and SCED for 50,000 binary variables and 15,000 transmission constraints over 36 hourly intervals, and they limit the day ahead of SCUC to 1200 s. Figure 9 illustrates the processes run by the HIPPO system in parallel. They run several algorithms in parallel to ensure continuity. If the most accurate fails to converge within its time limit, the solution of the next most accurate algorithm is taken

instead, such as SCED. The convergence criterion of the optimality gap is 0.1% which amounts to about $24,000. They use surrogate ADMM, Lazy transmission constraints from experience, and a Polishing-E method, which reduces the set of possible generators and an Uplift function to choose a good set of generators.

In their latest work [198], they introduce a neighboring search algorithm that improves their E-polishing algorithm and the selection of a set of lazy constraints. This HIPPO system uses 18 nodes and 24 2.3 GHz 64 GB RAM on the Pacific Northwest National Laboratory (PNNL) HPC cluster and uses Gurboi to solve their problems. In their study, they compared the time it takes to sequentially solve 110 different SCUC problems of different complexities against using HIPPO. They showed that problems that take a long time sequentially experience a more drastic speedup with HIPPO, meaning their system scales efficiently. One problem that took 2000 s in full sequential MIP was solved in 200 s with HIPPO. For all the tested cases, the highest solution time on the MISO network using HIPPO was 633 s, under the required standard time limit for finding solutions. In addition, they explore the possibility of solving at 15 min rather than hourly intervals, as this is more appropriate for RE generation. They solve the MIP for one hour and then feed that solution to the 15-min interval problem as an initial point. Compared to using root relaxation to solve 15-min intervals, a huge jump in speed is achieved once again for the more complex problems.

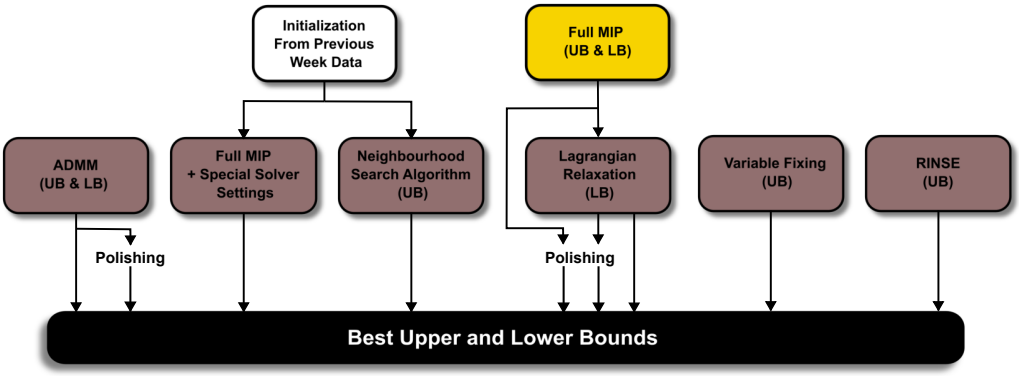

Figure 9. HIPPO Concurrent Optimizer (vertically aligned processes run in parallel).

Table 5. Unit Commitment MIMD based state-of-the-art studies.

Paper	Contribution
[174]	84% idle time reduction with an Asynchronous approach and Time-Sharing
[175]	AC- SCUC converted to Mixed Integer Second Order Conic program
[177]	Voltage Stability Constrained SCUC with temporal decomposition using APP
[180]	Decentralized Market SCUC using ADMM for temporal decomposition
[181]	Multi-Area UC, a novel extension of Interval Optimization & Asynchronous ADMM
[182]	Non-consensus multi-area decentralization using iterative ED and UC
[184]	Asynchronus decentralized UC by convex relaxations and other techniques
[186]	Achieving faster convergence in distributed asynchronous UC by SLR
[188]	Multi-Area SCUC avoiding the zigzagging issue at convergence by using ATC
[191]	Multi-Area AC-SCUC distribution using a variation of ATC.
[161]	Multi-level interactive UC Involving ADN, micro-grids, and DER
[192]	Data parallelism focusing on graph databases (NoSQL)
[35,197]	Commercial HPC UC HIPPO project novel strategies
[198]	Neighboring search algorithm improving HIPPO from [35]

7. Power System Stability

Power system stability studies in this section include Static, Transient, and Dynamic Stability. A power system is considered stable if it can regain operating equilibrium with the entire system intact after being subjected to a disturbance. Depending on the type of

study, the stability metric could be the line flows, the generator rotor angle, or bus voltage and frequency [200,201].

Static stability analysis involves solving the power flow or optimal power flow under contingency conditions. This type of study could involve either contingency screening (solving various PF problems in parallel) or solving the OPF or other power system problems monolithically under various contingencies when it comes to transient and dynamic simulations. These studies solve a set of differential and algebraic equations (DAE), which can be represented as:

$$\begin{aligned} 0 &= \Psi(x, V) \\ \Gamma x &= \phi(x, V) \\ x(t_0) &= x_0, V(t_0) = V_0 \end{aligned} \tag{4}$$

where x is the state vector of differential-algebraic variables besides voltages, V is the network's voltages vector, and Γ is the diagonal matrix where its elements are zero, where the equation is algebraic, and one where the equation is differential. Parallelism of those types of power system studies has been one of the most abundant and earliest to investigate, especially in transient stability. Most of these studies are performed offline, and the goal is to speed them up to solve them in real time.

7.1. MIMD Based Studies

7.1.1. Development

Power system operators need to detect system states or schedules that carry the risk of steady-state emergency if single or multiple equipment failures occur. This is to alleviate that risk, either by changing the system state to avoid such an emergency (preventative measure) or by employing a form of control strategy that would mitigate that emergency if it occurs (corrective measure) [200]. Contingency screening, in particular, is an embarrassingly parallel task and one of the easiest to parallelize, as it involves splitting several conventional power flow problems equivalent to the number of possible contingencies over multiple job arrays.

In earlier studies, the parallel steady-state analysis involved parallelizing the different contingency cases and not the power flow algorithm. The first successful attempt at distributed contingency screening might have been in [202], where the process was distributed over four DN425 processors. By adding pre-filtering schemes and strategies to reduce the computational burden, real-time static security assessment was already achieved in the early 2000s using multi-processing [203,204]. Further studies in enhancing the allocation of resources and dynamic load balancing in order to reduce the idling time of processors were performed by PNNL on their local cluster with the aid of MPI [205,206]. The same research team followed up the work by applying parallel betweenness centrality to identify the high-impact transmission lines in the screening process.

Some work in the area emphasized processor load balancing to task scheduling for contingency screening, from master–slave scheduling to proactive task scheduling. Incorporating both multi-processing and multi-threading on various systems and using various concurrent programming languages such as D [207] and X10 [208]. Most of the work today for static security involves improvement in whole EMS systems and software, and most of the modern work involves the efficient allocation of cloud-based resources. However, more elaborate schemes are appearing that involve parallelizing the OPF within the contingency analysis, which offers room for improvement, especially when considering more complicated OPF and post-contingency network models.

Dynamic stability is another form of steady-state analysis that evaluates the system condition and oscillatory behavior after very small signals and disturbances that last for up to 30 s due to fluctuations in generation and load levels or controllers. It is an obvious parallelism candidate, since it is the most intensive computational task in power system studies, as it models the electromechanical interaction between system compo-

nents and their controllers. It is also one of the most important and is directly related to secure operations.

Figure 10 illustrates the two goal-posts of dynamic simulation during its development. Real-time simulation means the computational time matches the duration of the simulated interaction. Faster than real-time simulation means the computational time is lower than the duration of the simulated interaction. The goal for achieving practical real-time parallel dynamic simulations was already being set in the 1990s [209]. Pioneering applications used the Conjugate Gradient method on the Cray Y-MP, iPSC/860, and IBM 3090 mainframe [210,211]. Parallel dynamic simulation studies quickly moved to large system implementations emphasizing balanced network partitioning and computational load and creating parallel software tools in the 2000s [212]. Faster-than-real-time simulations became the default, and the first faster-than-real-time parallel dynamic simulation was achieved on the WECC system for the first time in 2013 [213].

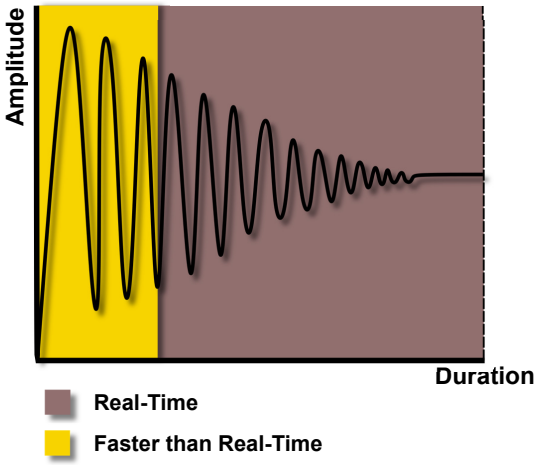

Figure 10. Real-time simulation vs. Faster than Real-Time simulation. The signal here represents the voltage oscillation due to a transient action.

Transient stability studies are made to ensure that after large disturbances such as circuit breaker trips or load loss, the system remains synchronized and can return to normal conditions. Parallel transient stability was explored in abundance, as they are critical, and the trapezoidal integration method makes them disposed to parallelism. Fernando L. Alvarado, in an impactful paper [214], demonstrated analytically that for time T and with some $\frac{T}{2}$ processors, transient stability differential equations could be solved in time order of $log_2 T$ using the trapezoidal method with potentially better convergence properties than serial implementation. The Electric Power Research Institute made a report in 1977 exhibiting various works exploring potential parallel applications for power system analysis [42].

Most of the work until that point discussed potential parallel methods to apply on parallel machines such as Cray-1, CDC STAR-100, IBM 370-195, and ILLIAC. Moreover, much like power flow development, many proposed parallel architectures that would exploit the parallelism in using methods such as Chaotic Relaxation, BBDF Gauss–Seidel, and Newton's method [215,216]. Some simulated the performance of new microcomputer and processor architectures using networks of computers or existing supercomputers, such as the CDC 6600 [217,218].

The first parallel implementation of transient stability might have been the first parallel power system study implementation by F. Orem and W. It was solved using a CDC 6500 equipped with an AP-120B array processor hosted on a computer. Comparisons on different hardware, such as vector processors vs. array processors or Cray-1 vs. IBM-3081D, were made using the trapezoidal integration method with linear and non-linear loads [219,220]. Different decomposition methods started to be introduced in the literature, including parallel-in-space and parallel-in-time approaches. The combination of the two was achieved while using "Nested Iteration" and time-windowing to enhance convergence in [221–223].

The variety of algorithms used and the discovery of different parallelism paradigms in transient stability simulation were promising for parallelism in power system studies in general. SOR and The Maclaurin–Newton Method (MNM) were used on the Alient and IPSC machines in [224,225]. The waveform relaxation method (WRM) was proposed by [226] to decompose the non-linear system into several dynamical subsystems to be solved in parallel, followed by [221,227]. A Parallel-in-Frequency paradigm was introduced with a demonstration of the possibility of vector processing coarse-grained algorithm [228]. More complicated fine/coarse-grained frameworks combining general processing (CRAY Y-MP9/464) and vector processing [229].

The possibility for real-time simulation was first demonstrated on the NCube2 HypeCube [230]. HyperCube machines were popular in the early 1990s for transient studies exhibiting various techniques such as SOR [231], and LU factorization path trees with different communication scheduling techniques [91,232].

Message passing in heterogeneous clusters started appearing in the late 1990s. This included Real-time Contingency and Transient stability with PMV [233], and other formulations with MPI [234,235]. Faster than real-time transient stability simulation was achieved by combining MPI and multi-threading techniques with network and time-domain decomposition in [212,236,237]. Later algorithms were used [238]. A full rundown of most of the techniques used until that point for large-scale transient stability studies can be found in [239].

Electromagnetic Transient studies (EMT) are transient studies that assess systems' overcurrents and overvoltages due to fault conditions or large disturbances. EMT simulations are the most accurate tools used to describe fast dynamics of power systems, and hence are the most computationally intensive. Software such as PSCAD uses an EMTP-type program, or EMtp, which is the most widely used algorithm for this type of study. Parallel EMT studies were first proposed in the early 1990s on MIMD hardware [240] and implemented using network partitioning techniques on a hypercube machine with care to load balancing [241]. Since then, various parallel approaches have been used, parallelizing the problem in space and time. The Very Dishonest Newton method, as implemented on multi-computer setup [242], on FPGAs real-time simulation of EMT was aimed for [243]. Techniques such as system partitioning and solving short time-steps for more dynamic parts and long time-steps for others in combination with multi-processing have been proposed in [244], but with no actual implementation.

Most of the recent work today in power system stability studies, particularly transient stability, takes place on SIMD architecture (GPUs specifically). The traditional way to perform the study is similar to that of regular transient studies. Thus, in-space-in-time decomposition can be achieved with similar techniques such as LU factorization [235], forward and backward substitutions [242], and graph theory [245] to be solved in parallel.

7.1.2. State-of-the-Art

Table 6 summarizes the critical work of this section. When it comes to probabilistic studies performed in contingency analysis, not many studies apply the N-2 criterion. Duan et al. performed N-2 contingency analysis generating with PPF for each contingency case [246]. With an IEEE-300 test case, using AC power flow equations and the NR method, and 1000 MC system state scenarios, the solution was distributed on the Danzek HPC cluster at Manchester. The solution time was 168 h. The only thing that this work offers is a testimony to the complexity of full AC Power flow equations. N-2 criterion merely increases the number of embarrassingly parallel tasks. Any real improvement needs to be made either with a finer decomposition of the embedded power flow problems, or by incorporating complexity formulations used.

In another N-2 reliability contingency analysis, transmission switching was incorporated for corrective action [247]. This work performed a dynamic stability analysis of the corrective transmission switching action to ensure its viability. AC power flow equations were used, and the analysis was conducted on the PJM interconnections (15.5 k bus system).

Both transmission and generation contingencies were considered, creating around 1.4 m contingencies. Heuristics were used to reduce the additional computation. The approach was very effective, as the solution time was reduced to 96 s using 128 threads (compared to 999 s with eight threads). This contrast between this result and the previous study demonstrates the importance of having a good parallel scheme.

One of the major challenges that face current implementations is the lack of standardization, as reflected in studies such as S. Jin et al. [248], which compared four different parallel implementations of the dynamic model. Some implementations were run on the IEEE-288 bus system and others on the WECC system, using different supercomputers with different hardware allocations for each implementation. This variety resulted in valuable but difficult-to-compare observations due to the unequal testing grounds. The work concludes that the direct integration method with a fast direct LU solver is the recommended approach for HPC dynamic simulation, as it enables faster than a real-time solver. Nevertheless, the recommendation is based on trials that are hard to compare fairly.

In an impactful series of works, P. Aristidou et al. implemented a parallel dynamic simulation on a transmission-distribution network using the Very Dishonest Newtons Method (VDHN) and Schur-Complement-based decomposition in 2014 [249] 2015 [250] and 2016 [251]. Their most significant test case included 15,226 buses, 21,765 branches, and detailed 3483 generator models, including their excitation systems and governors, voltage regulators, and power-system stabilizers. The software was written using standard Fortran language with OpenMP directives. The authors were very thorough in analyzing the performance, as their approach was compared against several, including fast and widely implemented sequential algorithms in terms of speedup and scalability. They tested their algorithm on various platforms, with the highest speedup achieved being $\times$4.7 on 24-core AMD processor-based systems. Faster than real-time dynamic simulation of the entire WECC system, interconnection (17,000 buses) was achieved in [109]. The simulation included machine models, exciter, governor, relay, and network models. With 16 core cores, they simulated a 0.05 s three-phase fault lasting for 20 s, with 0.005 s time steps, within 15.47 s. They achieved this feat using an open source HPC framework called GridPack, which they developed. GridPack is further explored in the discussion section of this review.

Table 6. Power System Stability MIMD-based state-of-the-art studies.

Paper	Contribution
[246]	N-2 contingency analysis with Probabilistic Power Flow
[247]	N-2 contingency analysis with corrective transmission switching
[248]	Comparison of four parallel dynamic simulation models
[251]	Dynamic simulation transmission-distribution network using VDHN and Schur-Complement.
[109]	Faster than the real-time dynamic simulation of the entire WECC system

7.2. SIMD-Based Studies

7.2.1. Development

SIMD architecture has long been used for transient and dynamic simulations to solve the non-linear differential-algebraic equations (DAE) and parallelize the trapezoidal rule, and GPUs have been used in power system studies as early as 2007 [252].

The first paper using GPU for static stability was published the same year CUDA was released, where DC Power flow contingency analysis was performed to solve the Gauss-Jacobi algorithm [252]. The paper used a NIVIDIA 7800 GTX card and direct instructions as opposed to a CUDA interface. GPU-based contingency analysis studies have recently been enhanced with pre-conditioning methods such as the Krylov theory [253]. Pre-conditioned Conjugate Gradient method [254] and compensation method and FDPF to parallelize AC Power Flow (ACPF) within contingency analysis [255].

For transient stability, A parallel program was developed and used by engineers in Hydro-Quebec in 1995, which simulates transient stability by parallelizing the Very Dishonest Newtons Method (VDHN) with LU Decomposition on the shared memory machine Sequent Symmetry S81 [256]. Waveform-Relaxation method was used later on the

same machine [257]. The first re-purposing of non-GPU image processing hardware for power system studies might have been in 2003 in [245], where PULSE IV image processor was used to achieve real-time transient stability simulation on WSCC 9 bus test case.

The first huge leap in performance was by Jalili-Marandi et al., in a hybrid approach achieving a speedup of up to ×344 for a 1248-bus system compared to CPU only approach [258]. Later Jalili-Marandi et al. refined the algorithm to perform all the calculations on GPU on an almost purely SIMD-based approach. Which proved to be more effective beyond 500-bus size systems [259]. In their last work, they showed the potential of GPU augmentation to enhance the inner solution performance. An instantaneous relaxation technique involving full newton iteration, implicit integration, and a sparse LU linear solver was tailored to run simultaneously on four T10 GPUs. A 9985 bus system generates a 22,272 × 22,272 matrix and 99.69% sparsity, which was solved within 5 min [260].

Yu et al. [261] performed another hybrid-based transient stability study by constructing a Jacobian Free Newton Generalized Minimal Residual method, which approximates the Jacobian vector products using the finite difference technique. While it eliminates a jacobian matrix step, it still requires heavy matrix-vector multiplication for a pre-conditioning step, making it suitable for GPU. This approach proved scalability starting from a 216-bus sized system, and it outperformed the newton based transient simulation solver PSAT. This approach showed better consistency performance enhancements for various sized systems compared to a similar work, where a pre-conditioning step was parallelized prior to the GMRES method [262]. Yet, using pre-conditioned GRMES in a combination of in-time coarse-grained schemes and in-space fine-grained schemes with GPU acceleration as in [263] showed universal scalability whether the number of GPUs used or the problem size increased.

There have been a few GPU-based parallel EMTP-type simulators for EMT studies. Some integrating wind farms to the models [264]. Others added complicated controls to large-scale systems with PV, transformers, and reactive components [265]. GPUs were used primarily to solve the linear algebraic equations associated with the algorithm, while the CPU performed most of the other parts. However, "Full" GPU-based parallel solvers that parallelize numerous other steps of the algorithm were developed recently [266].

7.2.2. State-of-the-Art

The most relevant recent work in power system stability mainly involved SSA and EMT and it is summarized in Table 7. In [267], Zhou et al. continue their work in N-1 SSA, this time treating the DC power flow contingency screening part, where only branch thermal violation is accounted for. This study tries to apply the Critical Contingency list contingency screening on GPU. DCPF-based Contingency screening involves dense vector operations. This paper claims to be the first of its kind to present a novel GPU-accelerated algorithm for DC contingency screening, where they optimize the data transmission, parallelization, memory access, and CUDA streams in their algorithm. The presented algorithm was tested on 300-bus, 3012-bus, and 8503-bus systems. The hardware used was A Tesla K20C GPU, and the host was Intel Xeon E5-2620 2 GHz CPU. They compared the performance against a single-threaded CPU-based algorithm implemented on a higher-end Intel Core i7-3520M 2.90 GHz CPU and 4 GB memory notebook. Their largest test case exhibited a speedup of 47× over the sequential case, demonstrating the scalability of their approach. They achieved this by reducing the data transmission by ×50, further optimizing task allocation and thread/block redistribution, and using memory coalescing to enhance memory access. The last improvement is particularly important, as memory handling is often ignored in the field, yet it is very crucial. Figure 11 demonstrates the impact category of the coalescing strategy used. Single threads often access different chunks of memory locations at the same time; when GPUs are most efficient when multiple threads access contiguous memory locations at the same time, this is called coalesced memory access.

To tackle the same problem, Chen et al. [268] use a slightly different GPU implementation, which exhibits a pipelined fashion. They employ a two-layered parallel method.

In the first layer, they apply a hierarchical path tree parallel LU decomposition for all contingency cases. In the second layer, they solve the decomposed problems for every contingency in parallel. The first layer process is repeated for every contingency case in parallel, sending groups of threads for each contingency. This means that the same process will run for several contingencies simultaneously, subject to the GPU's number of thread blocks allowed to run simultaneously. They employed an asynchronous scheme in which the CPU performs convergence checks. It receives the output convergence of three cases simultaneously; if one contingency calculation converges, the next one in line is sent to that available thread block. In their work, they pay attention to data management and utilize the cache architecture of GPU to improve their process. They compare their GPU approach against a KLU-based commercial suit that solves the problem on the CPU. In the largest case and using 32 GPU thread groups, their algorithm shows a 9.22× speedup over a single-thread CPU solution and a 3× speedup on a four-threaded CPU solution. In this study, the importance of matching thread number to warp number is demonstrated because using 32-thread groups did not make a massive difference to the speedup against 16-thread groups.

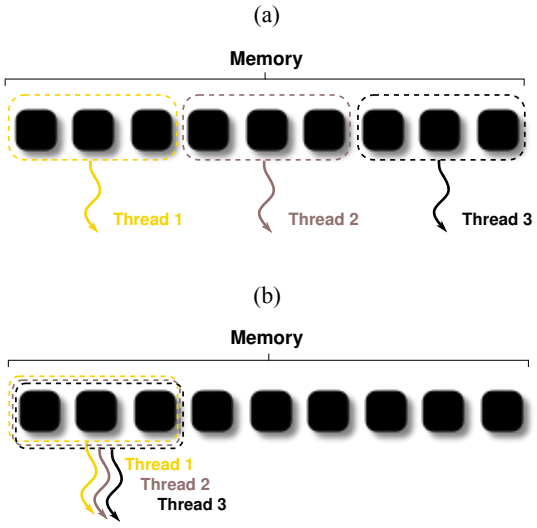

Figure 11. Illustration of strided memory access (**a**) and Coalesced memory access (**b**).

In [269], Zhou et al. expand on their previous work and attempt to skip the screening step, directly solving ACPF for all cases in a batch-ACPF solver. Their framework packages the batch-ACPF into a new problem through which a high degree of parallelism can be achieved. They created dependency graphs and used a QR left-looking algorithm for its numerical stability compared to LU decomposition. They compared their solver against commercial multi-threaded CPU-based SLS solvers KLU and PARADISO in several test cases. On an 8503-bus system, their solver took 2.5 s vs. 9.9 s (4× speedup) on a 12-threaded KLU solution and 144.8 s (57.6× speedup) on a sequential single-threaded solution. At face value, a GPU approach is superior. However, due to memory bandwidth limitations, adding more than ten CPU solvers would not have enhanced the commercial solver performance. Thus, performance-wise (core to core), this conclusion cannot be drawn. Economically speaking, the GPU is far superior, as adding a single GPU is much cheaper than several equivalents compute nodes.

Following the trend of resurrecting iterative solvers for PF applications mentioned earlier, Zhou et al. designed a GPU-accelerated Preconditioned Conjugate Gradient solver to solve the same ACPF for N-1 SSA in 2020. They tested their GPU-designed algorithm on 118, 1354, 2869, and 10,828 bus systems, the last system being the East China power grid. The number of contingencies was, respectively, 177, 1728, 4204, and 11,062. They compared their approach to two solution algorithms: 1—Complete LU SSA solution, which was implemented on a single core. 2—Rank-one update-based solution implemented on

a single, four, and eight cores supported by a multi-threaded solver. The hardware specs were similar to previous studies. With their GPU implementation of this algorithm on the East China power grid, the solution speedup was 4.9× compared to the eight-core multi-threaded CPU implementation. Zhou's works show that solvers for power system-related problems tailored for GPUs have considerable potential.

When it comes to EMT, the first record of GPU use is in [270], where co-processing of vector operations for 117 bus networks on GPU had double the speed of PSCAD, a CPU-based software. In later work, they reduced the communication between the CPU and GPU and derived a GPU-specific algorithm to achieve close to ×40 speedup on a 3861 bus network [271]. A similar implementation that aimed to reduce communication time and fully and efficiently exploit the SIMD architecture in EMT was also conducted in [272].

Earlier work by Zhou et al. was a GPU implementation of Electromagnetic Transient Simulation [273]. They utilize both SIMD and scalar operations and emphasize the importance of avoiding simultaneous memory access when parallel processing. The homogeneity of tasks was ensured where elements that are modeled similarly were grouped; for example, all RLC elements were processed in the same kernel with a unified lumped model. Separate Kernels were made for the Universal Line Model (ULM) in four stages, where in every stage, kernels were executed simultaneously. The Unified Machine Model (UMM) simulation, the third ubiquitous task, was also divided and managed in detail. The level of parallelism in this work is massive and attempts to squeeze every parallel structure in the problem and an ounce of performance from the GPU. The algorithm was tested on eight test cases from 40 buses up to 2458 bus systems. They used a NIVIDIA Tesla C2050 GPU with 448 cores and 3 GB memory and an AMD Phenom 4 core 3.2 GHz CPU. The total simulation time was 100 milliseconds at 20 microseconds. On that setup, a speedup of 5.63× was achieved compared to an optimized commercial software EMTP-RV.

Finally, in one of the most impactful papers by the same authors, a huge test case containing 240,000 buses was decomposed with propagation delay [274]. The system was divided into linear, non-linear, and control subsystems. Additionally, the Jacobian domain and the voltage calculations were parallelized, creating another decomposition layer, resulting in a highly fine-grained problem. Two GP104 GPUs were used in which all the iterations were processed, and convergence was checked. Using a single GPU against the EMTP-RV resulted in a 15× speedup. Moreover, GPU linear scaling was demonstrated by achieving double the speedup (30×) by adding the second GPU. However, it is important to note that the 240,000 bus system was an augmentation of the IEEE-39 bus system.

Table 7. Power System Stability SIMD-based state-of-the-art studies.

Paper	Contribution
[267]	N-1 SSA with DC power flow branch contingency screening
[268]	A two-layered parallel method which exhibits a pipelined fashion
[269]	Direct ACPF solve in a batch-ACPF skipping screening
[270]	First GPU parallel EMT
[271]	Improvement over EMT with reduced CPU-GPU communication
[272]	Fully and efficiently exploit the SIMD architecture in EMT
[273]	Utilize SIMD and scalar operations avoiding simultaneous memory access.
[274]	Fine-grained algorithm by parallelizing the Jacobian domain and the voltage calculations

8. System State Estimation

Power SSE is a centerpiece of control centers. Thousands of voltage measurements are collected from SCADA systems and Phasor Measurement Unit (PMU) and processed to understand the conditions of the grid better. SSE studies provide accurate and reliable estimates of the phase angle and bus voltages from incomplete system measurements.

8.1. MIMD-Based Studies

8.1.1. Development

Parallelizing SSE, much like other studies, started with simulations. Y Wallach and E. Handschin proposed a distributed master–slave topology [275], showing that merely partitioning the network would achieve speed gains. Later in 1982, C. W. Brice and R. K. Cavin Simulated the potential performance of distributed and decentralized algorithms on parallel hardware for state estimation, where one is communication intensive, and the other is computationally intensive [276]. Different SSE decomposition techniques and parallel simulations were carried out over the 1980s and 1990s. Those include block partitioning to decompose the state estimation problem by the network simulating the performance on a MIMD machine [277], parallel forward–backward substitution [278], recursive quadratic programming [279], the Dantzig–Wolfe Decomposition Algorithm [280], and other simulations [281]. The first practical implementation was in the year 2000 using APP [282,283].

The Weighted Least Square (WLS) algorithm is the most commonly used in SSE. WLS contains a matrix inversion step, which can be solved using LU decomposition. The first parallel WLS solver implementation might have been in [284], where shared memory vs. MPI schemes solving the linear system of equations were compared. The exploitation of parallelism in the Khan Filtering Method only showed up in 2009 [285], although it has been in use since the 1970s to improve the prediction aspect of SSE.

8.1.2. State-of-the-Art

The most relevant work made in this area on MIMD is by Korres et al. They used an efficient distributed WLS algorithm to perform multi-area state estimation in parallel using MPI [286]. They tested the algorithm with several processors from 1 up to 60, solving problems using the scientific toolkit PETSc, which contains parallel optimization linear and non-linear solvers. They employed different communication/coordination strategies and control area partition numbers and sizes to estimate the state of an 1180 bus system (10× IEEE-118 system) in two cases, where case 2 exhibited more interconnections between "slave" areas. The algorithm was implemented on the National Technical University of Athens cluster consisting of 11 Intel Core 2 Duo E8200 PC nodes. However, in this work, scalability was demonstrated, but it lacked a speedup comparison against the fastest sequential algorithm.

8.2. SIMD-Based Studies

8.2.1. Development

Most of the development of state estimation occurred over MIMD studies. The earliest significant SIMD application occurred in [245], where the PULSE IV, a scalable SIMD Chip, was designed to help achieve faster-than-real-time application in 2003.

8.2.2. State-of-the-Art

Table 8 summarizes the critical work of this section. Several uncommon SSE techniques were related to GPU, such as the Fast Decoupled State Estimation [287] and selective column identification in the numerical differentiation [288,289]. The WLS algorithm is the most commonly used for electrical SSE. WLS contains a matrix inversion step, which can be solved using LU decomposition. That is what Karimipour et al. did in [290], in which they implemented all the steps of the solution algorithm from the Admittance matrix formation to the convergence check on GPU. The GPU used was a 512-core NIVIDIA with double-precision peak performance and the Intel Xeon E5-2610 2GHz CPU as a host. The test system used was an IEEE-39-bus system, duplicated and interconnected to create larger systems of up to 4992 buses. They achieved a speedup of up to 38 for the largest system compared to a sequential CPU implementation. The algorithm exhibited strong scalability, and they estimated that the maximum theoretical speedup achievable by that GPU is 312× for this algorithm. Notably, they also address the issue of solution discrepancy due to

the hardware architectural difference. They did this by considering both Correlated and Uncorrelated Gaussian Noise in the measurement samples (to consider bias) in small test cases, which is supposed to lead to larger errors in the final result. The errors that occurred in the GPU solution matched those in the CPU solution, which confirmed the robustness of their algorithm.

Later Karimipour et al. produced a GPU parallelized Dynamic State Estimation based on Kalman Filters [291] on a Tesla S2050 GPU. Compared to a quad-core CPU, the approach achieved a speedup of $\times$10 for a 19,968 bus system with 5120 generators exhibiting close to zero error of estimation. They extended and refined the approach by increasing the GPU portion of work and utilizing PMUs, and SCADA measurements [292]. For a smaller system (4992-bus), they achieved a higher speedup (15$\times$) with high voltage precision (0.002 p.u. and 0.05 rad error). Finally, they made the algorithm robust against coordinated false data injection, enabling its detection through the parallel algorithm and optimized, secure PMU installation [293].

In [294], the Dishonest Gauss Method in the WLS algorithm is used, where the Jacobian update is not executed at every iteration. The algorithm was implemented on a Tesla K20c GPU, fragmenting the original by vectorizing multiplication and multi-threading addition processes. They investigate the method's accuracy and show that to get an accuracy of 100%, the Jacobian needs to be updated every seven iterations. They also demonstrate the algorithm's robustness by applying different noise levels to determine its effect on accuracy, showing the method ranges between 98–100% at different levels of noise. A complete mathematical analysis of the convergence of their method was conducted in [295]. Real-Time Digital Simulators measurements were used while adding errors that vary from 1–15%, and 30 samples per second were taken, a typical PMU sampling Rate. On an IEEE 118 system and a three-second duration, their algorithm achieved a 15$\times$ speedup over what is claimed to be the best existing CPU implementation.

Table 8. State Estimation SIMD-based state-of-the-art studies.

Paper	Contribution
[287]	Parallel fast decoupled state estimation
[288,289]	Parallel selective column identification in the numerical differentiation
[290]	Implemented all the steps of WLS algorithm on GPU
[291]	GPU parallelized Dynamic State Estimation based on Kalman Filters
[292]	Increasing the GPU portion of work and utilizing PMUs, and SCADA measurements
[293]	Enable PMU false data detection through a parallel algorithm
[294]	Fragmenting Dishonest Gauss method by parallelizing multiplication and addition
[295]	Mathematical analysis of the convergence of their parallel method [294]

9. Unique Formulations and Other Studies

9.1. SIMD-Based Studies

Almost all of the previous studies involve classical power system optimization problems, the ones showcased in Figure 12. These can be augmented and combined in various ways to achieve various objectives. Thus, a few unique and computationally expensive formulations in the literature sprouted with parallel implementation as summarized in Tables 9 and 10. A common one is an OPF problem that ensures the voltage stability in the solution, the Transient Stability Constrained OPF (TSCOPF). Adding such constraints complicates the problem, but it was shown that by using techniques such as Benders decomposition [296] and reduced space IPM [297], a remarkably faster solution can be achieved prior to parallelizing the subproblems.

In one formulation by [298], the TSCOPF is combined with UC to create a Transient Stability Constrained UC (TSCUC). The criterion of transient stability is given by the following inequality constraints:

$$\begin{aligned} \delta^{min} \leq \delta^{i,0} - \delta^{0}_{COI} \leq \delta^{m}ax \\ \delta^{min} \leq \delta^{i,t} - \delta^{t}_{COI} \leq \delta^{m}ax \end{aligned} \tag{5}$$

where COI refers to the moment of inertia of the rotor. The previous equations, along with others in the UC formulation, add temporal complexity. Thus, the TSCUC formulation was decomposed temporally using APP, and 24 cores were used to solve it on the IEEE-300 system. The model showed notable scalability; the solution time was reduced from 16 h to 1 h. Furthermore, using the same pre-defined contingency, the transient stability of the first-hour interval of the solution was examined and compared to a standard SCUC solution. The TSCUC solution maintained the whole system's stability without any alteration, whereas the SCUC solution failed to regain stability even by modifying the power output post-solution. This means that the proposed model guarantees stability as opposed to the conventional one.

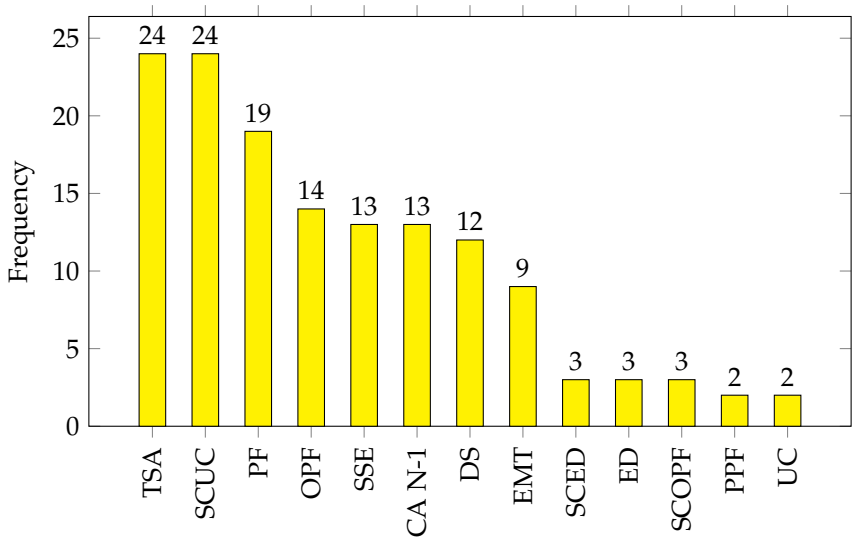

Figure 12. The occurrences of parallel classical power system studies.

Table 9. Unique Formulations and Other SIMD-based state-of-the-art studies.

Paper	Contribution
[298]	TSCUC (TSCOPF + UC) temporal decomposition using APP
[299]	ACOPF mitigating electromagnetic storm disturbances using APP

Deviating slightly to the security side, interesting work by [299] investigates mitigating the disturbance effect of geomagnetic storms on power systems. Those manifest as low-frequency, quasi-DC, high-impact extreme events. Tools for large and complex power systems to mitigate this problem do not exist; thus, this paper proposes a parallel solution approach to conduct optimal secure operation planning considering geomagnetic storm disturbances (GMD). They include Geomagnetic Induced Current (GIC) into an ACOPF model and turn it into an MINLP by modeling GIC blocking components holding three states, bypass, resistor, and capacitor. APP is used to decompose the problem into two problems, an NLP, solving for ACOPF, and a MILP which determines the state of switchable network components and calculates the GIC flow. The two subproblems can be solved independently and in parallel, coupled by a third model. Tests were performed on a multi-core workstation showing the utility of this approach in mitigating GIC for a 150-bus network. However, the authors did address that their extensive experience with the method guarantees that their APP approach results in an effective solution.

9.2. MIMD-Based Studies

Table 10 summarizes the critical work of this section. There have been a few venturesome studies, such as in [300], which used GPUs to accelerate a stability analysis involving both transmission and distribution systems into the network, which achieved some performance improvement, but the speedup factor might not justify the use of power. In other formulations, the gas and thermal systems are modeled and coupled into the electrical system to calculate the energy flow using the Inexact Newtons method and GMRES pre-conditioning [301]. Contrary to previous studies on GPUs, this study showed that the larger the system, the more time is required. This is a testimony that care must be taken in routines and schemes that would scale with the hardware. Finally, the last notable work is in the AC TSC-OPF using GPU acceleration [302]. TSC-OPF is an extension to ACOPF with EMT dynamics and stability constraints. Below is the compact formulation of the problem:

$$\begin{aligned} min_{u,v_0,v_i(t)} \quad & \phi(u) \\ s.t. \quad & \Omega(u, v_0) = 0 \\ & \underline{\Gamma} \leq \Gamma(u, v_0) \leq \overline{\Gamma} \\ & v_0 = v_i(0) \qquad\qquad i \in [1, ..., n_C] \\ & 0 = F_i(v_i(t), v_i(t), u) \\ & \underline{\Psi}_i \leq \Psi_i(v_i(t)) \leq \overline{Psi}_i \end{aligned} \tag{6}$$

where u is the control variable, including active and reactive generation outputs. v_0 and $v_i(t)$ refer to steady and transient state variables of the ith contingency in the differential equation F_i. n_C is the number of contingencies. The constraints in the vector expressions above include the steady state and transient-state constraints, DAE, and initial conditions. The elaborate model can be found in [303]. The reduced-space IPM was used in this study, and it was the portion that was decomposed and parallelized on GPU using Schur's complement. They used a 12,951 bus system in the largest case and compared their GPU-accelerated algorithm to a single sequential core and parallel 16-core CPU implementations. Their algorithm achieved a speedup of $\times$24 and $\times$6.5, respectively.

Table 10. Unique Formulations and Other MIMD-based state-of-the-art studies.

Paper	Contribution
[300]	Transmission and distribution system stability analysis on GPU
[301]	Thermal & electrical system coupling using Inexact NR and GMRES
[302]	AC TSC-OPF using GPU acceleration

10. Grid and Cloud-Computing-Based Studies

Table 11 summarizes the latest contributions in Grid and Cloud-based studies. The use of HPC facilities in the electrical power industry is not uncommon in various offline (and some online) applications, especially ones related to smart grids and microgrid planning [46]. For example, California Independent System Operator (CAISO) uses HPC to perform various real-time assessments of the network, such as reliability assessments [34]. Facing the huge computational load, ISO-NE installed an on-premise computer cluster in 2007 using EnFuzion as a job manager [304]. They later faced challenges in choosing the optimal size for clusters and investment in computational power, since the peak computing jobs and average ones were very different. Hence, it made sense to move some non-emergent applications to Cloud. In fact, ISO-NE had already initiated a project to adopt cloud computing with emphasis on achieving privacy and security [305,306]. When facilities begin to struggle to meet the increasing requirement of deployed power system applications, it makes sense to resort to cloud services. Cloud computing really expands the realm in which algorithms and systems can be parallelized and exhausted. Regulators and players will not have to worry about the availability of resources any more. Instead, they

will squeeze out every inch of performance and manage the "rented" resources. Unneeded capital investment can be avoided, and real-time data can be shared with third parties.

When it comes to Grid and Cloud computing studies, performance enhancement is often sought through scalability and resource availability rather than optimizing for specific hardware. While this works very well, the combination of fine optimization would be much more powerful. However, this might be only possible through the Grid rather than the Cloud model, since it is more controllable. Most of the work in this area is very recent, but it starts with a few studies on Grid Computing. In an application that is very similar to the Cloud computing paradigm, the work by Morante et al. [307] might have been the first modular and hardware scalable implementation of parallel contingency analysis on a grid of eight heterogeneous computers. A middleware called Hierarchical Metacomputers (HiMM) was used to allocate resources economically based on resource adequacy and a given budget value. By increasing the budget value, their middleware managed to lower the execution time by exploiting more expensive, more powerful resources. Other papers from the time explored the idea of monitoring and control of the power system using decentralized schemes on grid computing [308]. A few more studies explored Grid-based frameworks and applications, such as Huang et al. [309,310] and Ali M. [311], load flow on Grid by Al-Khannak et al. [312], and dynamic security assessment by Xingzhi Wang et al. [313].

There were a few recent studies that showcased large-scale cloud implementations. One study was part of a diverse paper showcasing the challenges and experiences gained by ISO-Newengland in moving to cloud services. In their move, they used Axceleon CloudFuzion [314] job balancer, which provides high failure tolerance and job monitoring. The work involves heuristics and operational decisions, providing a great insight into the methodologies and equations used to choose the number of instances and squeeze every bit out of the rented hour. An N-2 contingency analysis was performed on a test case that takes 470 h on a regular workstation; the case jobs were carried out in less than an hour with their scheme. CloudFuzaion was not flawless, however, as its workflow was often interrupted by manual steps (which meant it had to be monitored). Thus, ISO-NE started a project with Axceleon to develop an independent power system simulation platform for cloud computing that addresses that issue, fully automating processes after receiving the user input. In a 2019 study [304], they demonstrated that their platform managed to run multiple instances reaching near 100% CPU utilization of the instances launched for certain jobs and capable of many task computing and co-simulations.

Security becomes a major issue when using cloud facilities, and while the work above used a service-level security mechanism, Sarker and Wang [315] wanted to ensure security, assuming that the cloud in-house security infrastructure is compromised. They transform the ED problem into a Confidentiality-Preserving Linear Programming (CPLP) formulation [316,317] to achieve holistic security, such that all sensitive information remains unknown by competitors. The approach protects against attacks from passive and active entities on the Cloud (administrators and customers). It works by converting inequality to equality constraints and multiplying the coefficients by randomly generated positive real numbers twice (a mononomial matrix U then H), which are held privately. The resultant constraint matrix is sent to the Cloud, and the equipment information implied in the constraint coefficients remains obscure to any attackers. This work enhances the security matrix reduction of CPLP. Since the feasible region of the CPLP that is produced after those operations is the same as the original LP problem, solving for those new constraints, (the CPLP) yields the same solution as the LP. A test of the algorithm was performed on a 2383-bus Polish system, including 327 generators solved using CPLEX, comparing its performance on four different cloud instances. The method showed scalability, but it was not tested against a regular ED algorithm.

Another paradigm that Cloud facilitates is the Many Task Computing paradigm. It facilitates Co-Simulations, which involve solving many optimization problems and performing many studies apart and then connecting them. In [318], they perform a large co-simulation by decomposing a network into heterogeneous partitions that are unique

to each other, creating different problems for each partition (e.g., generators, passive components, loads, etc.). The dynamic resource allocation ability fits well with large-scale co-simulations because 1—different components have different transient reactions. 2—They might require different timesteps depending on transient status. 3—Each problem could have a different formulation (NLP, MILP, etc.) and require different solution times. The paper demonstrates the achievable co-simulation performance and interfacing on the Cloud using existing commercial tools. For example, in one instance, the network was divided into multiple Simulink models, launching Matlab script simulations in different processes. In another trial, a compiled MPI C code was used, and Simulink executables were to run the simulation.

In [319], a fine scope was taken on task management of massive parallel contingency analysis using the Hadoop Distributed File System on the Cloud. They applied an N-1 transmission line contingency analysis and used the NR method to solve the power flows. First, the system distributes the contingency and other parameters to separate nodes such that each node solves a contingency case. What is unique about their job management scheme is that when the number of cores increases, the network bandwidth automatically increases as well, further increasing the performance. With this approach, they could perform a full AC contingency analysis for a real network in less than 40 s.

Table 11. Grid & Cloud Computing state-of-the-art studies.

Paper	Contribution
[307]	First modular and hardware scalable implementation of parallel contingency analysis
[314]	Showcase the challenges and experiences by ISO-Newengland in moving to cloud
[304]	Many task computing management, near 100% utilization of CPU instances
[315]	ED specific cloud security algorithmic re-reinforcement
[318]	Large EMT co-simulation decomposing a network into heterogeneous partitions
[319]	Parallel contingency analysis using Cloud Hadoop Distributed File System

11. Smart Grid and Renewable Integration Applications

A summary of the contributions of studies covered in this section is availalbe in Table 12. In 2010, the literature almost completely shifted towards cloud computing and particularly an integrated framework combining smart grids with the Cloud, given the advent of AMI and big data at the dawn of that year. Several frameworks and models for smart grid co-ordination [320,321], and power system assessment [206] using Cloud appeared that year and later [322]. Ideas such as cloud-based demand response were being explored [323], and many papers suggested network architecture and control topologies that are realizable with cloud [324]. Concepts such as Cloud assisted IoT could help us achieve a much more efficient network of sensors for future power systems. An example of such a system is an architecture for RPL-based IoT application, which specifies the application of RPL focusing on reforming industrial operations through cutting-edge technologies [325]. The versatility of cloud infrastructure is an excellent complement to the smart grid future, and its applications are covered in this review as it involves distributed work, which is the core of parallel computation. The diagram in Figure 13 shows a typical multi-layer vision shared across the Cloud integrated smart grid literature.

Interesting network paradigms could be created given that computational resources can be flexible and scaled as cloud services provide. Sheikhi [326] explores the idea of an "Energy Hub" where customers can be active in Demand Response Management by reducing their direct electricity consumption and using the output of the combined heat and power from the energy hub that the gas supplier supplies. This does not change customers' electricity consumption level, but the demand has reduced from the electrical supplier's point of view. This Energy Hub + Smart meter is now a Smart Energy Hub, and single or multiple customers could share it. The problem was formulated in a game theory approach where the Smart Energy hub is a price anticipator, which tries to predict the consequences of its own action on the price and chooses the optimal load-shifting schedule to reduce

the cost on customers on those bases. In a similar fashion to [327], the smart energy hubs read and control the outputs and send data to the Cloud to be aggregated and computed for decision-making, solving the game according to the cost function. They simulated their approach and showed that it resulted in a decrease in energy price compared to no Demand Side Management (DSM) game. They also compared the communication cost of direct messaging configuration vs. cloud configuration, where the Cloud showed lower cost, making the platform more suitable for such applications.

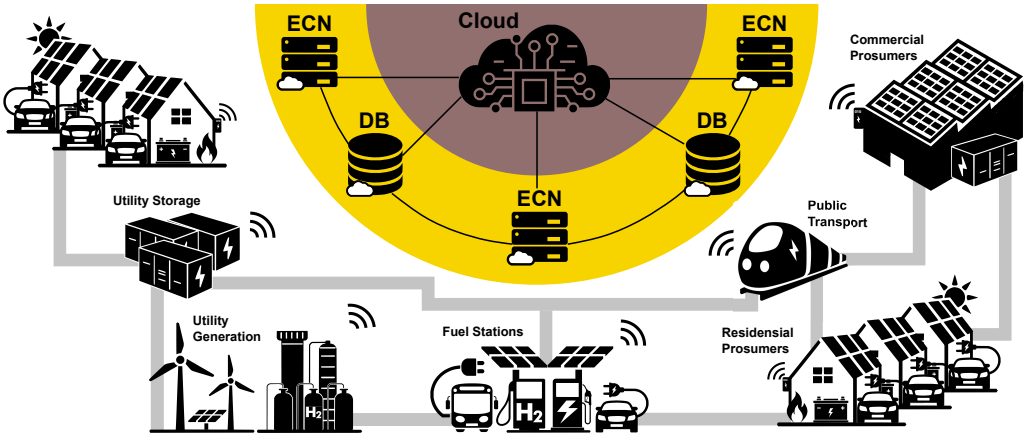

Figure 13. The general idea is to have a network of databases and compute nodes (which includes smart meters) to keep essential local data and computation locally where quick responses and decisions are needed, while the cloud is overseas and sends signals coordinating the system at a higher level. DB: Remote Data Base, ECN: Edge Compute Node.

For certain applications, such as DSM, fixed resources become an even greater issue as the amount of information processing and the computational requirement fluctuates based on the availability/flexibility of demand-side resources, which dictates the complexity of the problem at every instance. One of the options that are becoming more attractive is using cloud computing services, which could be much cheaper than expanding existing facilities. Using such services means an optimal allocation of the computational resource becomes much more crucial as cloud services are often billed based on the consumed resources, and pay-as-you-go terms [328]. In 2016 Z. Cao et al. [329] handled this issue with a source allocation algorithm that finds the optimal cloud computing resources for DSM instances. Commercial cloud computing resources differ from regular HPC clusters in the sense that there exists a greater variety in architecture, and the performance compactness might be lower than that of specialized HPCs used in research.

On a much more refined and more local scale, Wang et al. [330] attempt to decentralize the problem of Dynamic Economic Dispatch (DED), i.e., energy management in real time, by using inverter Digital Signal Processor (DSP) chips and cloud computing. The paper solves a multi-parametric quadratic programming optimization problem, which has been highly applied in the area of coordinated power system ED and TSO-DSO network coordinated dispatch. The solution involves two parts and is decomposed into two subproblems: 1—An offline calculation that Cloud carries out. 2—Real-time decision making that the DSP carries out. In the cloud computing part, distributed renewable generation and loads are forecasted to create piecewise expressions. Every 4 h, the expressions and information are sent to inverters so that the DSP chip can solve and optimize the output based on the real-time input of load and RE. The Cloud provides flexibility and handles the highest computation burden while simplifying the subproblem solved by the inverter.

Using a 14-node test case with PV, wind, Grid, diesel, and battery systems, the authors drew a comparison between their approach and a traditional implementation on an i7 regular laptop. Amazon Web Services (AWS) instances were created, and a real DSP chip was used. Their test showed that by moving offline computations to Cloud, it was solved within 34 µs compared to the traditional algorithm (372 ms). This is a colossal speedup,

but it might not be fair, since the traditional algorithm creates a whole new deterministic problem every time it collects new values. The main gist is that it achieves the needed solution time for using DSP, since the calculations on the inverter must be lower than 100 µs so as not to cause issues and interruptions. Sharing the inverter's chip rather than adding local controllers lowers investment and maintenance costs, and the distributed nature makes it robust against single-point failure. However, care needs to be taken in the job timing such that the control functionality of the inverter is not interrupted.

Addressing the security concerns of the Cloud, F. Ma et al. also proposed a cost-oriented model to optimize computing resource allocation, specifically for demand-side management problems using simulated annealing and modified priority list algorithms [33]. The objective function parameters are based on actual Amazon cloud service pricing. This cost-oriented model was compared to a traditional O2O model, which allocates resources based on the peak computational load for the renting period. The proposed optimization method showed a significant cost reduction over the traditional source allocation method. There is a security concern that comes with outsourcing sensitive processes. For other players in a free market, there is an economic benefit to engaging in cyberattacks and accessing information from competitor rs' processes, such as ED, as it could help with their bidding strategies. In their study, they explain the use of the Virtual Private Cloud (VPC) scheme, which isolates their portion of the Cloud such that their resources are not shared with other organizations or applications, even if idle. It is supposed to increase the security of the outsourcing process. Yet, one can see how the spread of such a strategy would create an impediment to the scalability and efficiency of the Cloud.

Table 12. SmartGrid & Renewable Integration state-of-the-art studies.

Paper	Contribution
[326]	Explores the idea of an "Energy Hub" with active demand response
[327]	Energy hub cloud-based aggregated computation
[329]	Optimal cloud resource allocation based on available demand response
[330]	Decentralization of Dynamic Economic Dispatch using Digital Signal Processors on AMI
[33]	Demand side management using simulated annealing and modified priority list

12. Discussion

Parallel Applications for power systems started showing up around the late 1960s and early 1970s, around the same time when a commercial market for supercomputers and clusters was sprouting. At that stage, parallel computers were still experimental in nature, and their design often targeted a specific problem type or structure. Very few computers were suitable for power system studies, as most had low arithmetic precision that is equal to or less than 32-bit, which has been shown to be inadequate for direct solution methods [331].

At an abstract level, computer hardware architecture and its uses in power system studies are still the same. What used to be a "computer" or "host" is today's CPU, and SIMDs such as array processors were used just like today's GPUs would be used for power system studies. Algorithms that include diakoptics/tearing and tree graphs used to be a common theme at the start of vectorization and fine-grained parallelism, and it is still used in current GPU power system studies. Another example of similarity is that one of the issues faced at the time was that transient simulation timestep iterations sometimes required substantial logic and data to model for each node [332]. This means it would create a burden on the computer that hosts the array processes and could cause communication bottlenecks. This is very analogous to what happens today in GPU-CPU optimization algorithms. Ironically, S. Jose argued in 1982 [332] for the need for a general-purpose processor to tackle the previous issues since vector/array computers pose software hurdles and challenges that are too great to justify the enhancements achieved. Yet the same challenges are faced today, just at a different scale and magnitude (i.e., GPU-CPU interfacing/Cloud Implementations). A major shift in the field occurred around the 1990s; around the same

time, general-purpose processors experienced significant innovation and cost reduction, and more parallel optimization algorithms started appearing. Studies in power system stability became abundant, and UC algorithms debuted with most papers using meta-heuristics to solve the problem. While implementation would have been arguably doable, simulations of parallel hardware still existed because more care was placed on implementation optimization and ensuring the practicality/portability of the parallel algorithms. The meaning or extent of what is considered coarse-grained and fine-grained algorithms shifted over time.

The main direction for HPC incorporation in power system studies applications is moving towards real-time applications, much more so than offline applications. From the literature, it seems that renewable energy generation is the urgent driver for resorting to using AC formulations in real-time applications, followed by annual cost savings of replacing DC formulations. Benders decomposition and Lagrangian relaxation seem to be the most common combination in decomposing stochastic full AC problems. In larger systems, the application of parallel computation is clearly more advantageous, while in smaller systems, serial programming performs better or at least matches parallel computational approaches, mainly due to the communication overhead, as an increased number of processes means longer and more communication time between them. The extent of this effect depends highly on the strategies used in parallelization, as well as the cluster architecture and hardware used. GPUs, for example, exhibit extreme parallelism in processing architecture, yet have superior performances over more coarse CPU implementation shown previously [333]. Many organizations and research teams are developing public tools and frameworks to help incorporate HPC into power system studies. PNNL developed HIPPO [334], a tool to help grid operators tackle SCUC by leveraging optimization algorithms for HPC deployment. PNNL also initialized the development of another framework for power system simulations called GridPack, which falls under a larger suite called GridOPTICS [335]. While such tools facilitate the use of multi-processor parallelization, others such as Nividia CUDA [336] evolved GPUs—which have an immensely parallel architecture—to become easily programmable and sprouted the trend of using GPUs for scientific calculations showing a promising future. This trend can be observed in Figure 14.

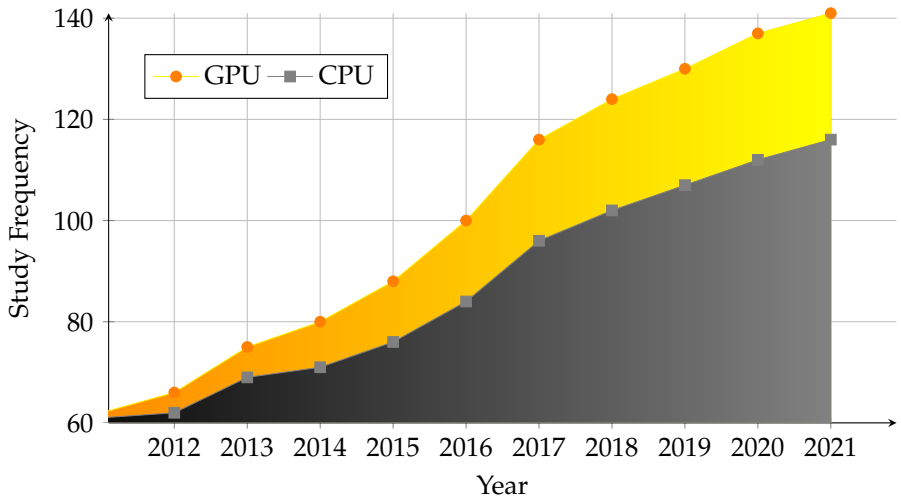

Figure 14. The emergence and increase in use of GPUs based on the power system studies covered in this work. Grey and yellow areas represent the cumulative volume of CPU and GPU based studies respectively.

On-premise HPC is not future proof, as the grid organism keeps on evolving. A power system with n components, with each component having m states, can have m^n over all possible states. The Grid is quickly adding more components in terms of quantity and variety, AMI, EVs, IoT, etc. All power studies will keep on growing, and control rooms

and operators will also need immediate visualizations for easy information analysis. This means that power system operators will inevitably resort to Cloud services. However, cloud computing has many of its own challenges related to policy, security, and cooperation before any solid adaptation is made. The Optimal placement of data centers depends on various stochastic factors, and the lack of interoperability between providers of cloud services does not make this problem any easier. Regulatory compliance in terms of security and access is extremely hard to ensure. Data and process locations are unknown, and it becomes hard to investigate any dysfunction or intrusion. An efficient recovery mechanism needs to always be in place, and even if the host company's structure or ownership changes, long-term data viability must be in place. Moreover, the business case for moving to cloud computing needs to be established first, which is different for every entity, and it is difficult to predict the future costs of the services. Lots of preparation and tools need to be created locally to ensure stable operation and inseparability and security, such as handling software licensing issues and data coordination/processing.

Computation aggregation evolved from a single processor to a processor and accelerator to a multi-processor system, Beowulf clusters, and multi-core processors, then a grid. Even at a small level, much like the way vector arrays and ALUs were added to processors, future CPUs and GPUs will be integrated into the same device, and the cycle continues. In the future, the Cloud will be an integral part of all operational entities, including the electrical industry. The future electrical Grid and Cloud will look very different from today; both will be dynamic and transactive and will have a reciprocal relationship in which the Cloud acts as the brain of the electrical network, and both will probably be driven by similar forces.

12.1. Software and Solvers

Commercial solver use can be traced back to the sixties with solvers such as the LP/90/94 [337] in conjunction with the development of the field of mathematical programming. Thus, today, there is an abundance of open-source and commercial solvers that race to employ the best techniques to solve standard problem formulations. This is evident in Figure 15, showing the variety of solvers used.

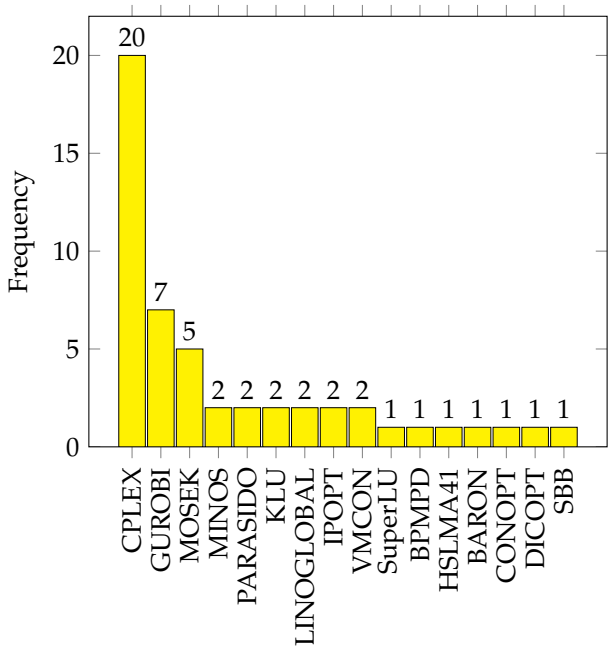

Figure 15. The occurrences of solvers in the reviewed literature.

To a large degree, commercial solvers simplified optimization for engineers, allowing them to focus on modeling, leading to the subfield of model decomposition. Nevertheless,

a few challenges arise when using commercial solvers instead of employing a specific solution algorithm to the problem. The heuristics involved in solver design could create a vast disparity in performance, even for solvers within the same caliber solving the same type of problem. Additionally, the ability of a solver to identify and exploit the structure of the model heavily determines whether the model can be solved within a reasonable time. If the solver fails to accomplish this step, it might exhibit exponential growth in running time, as indicated by complexity analysis. Moreover, hidden bugs and issues with the source code of the solvers could exist. This is particularly true for commercial solvers.

Established commercial solvers with full-time development teams, such as CPLEX and Gurobi, exhibit a more comprehensive dictionary of identifiable problem structures to accommodate the large user base. They are robust, scalable, and capable of handling large search spaces with multithreading and HPC exploitation capabilities. Moreover, they are easy to install and interface with many programming languages. Figure 15 shows the hierarchy of occurrences of different solvers in the surveyed literature, and it can be observed that the previously mentioned solvers dominate the literature for the previously mentioned reasons. However, this should not deter us from experimenting with non-commercial solvers, as they may be superior for specific problems. It is also worth noting that all the well-established general solvers in the tier of CPLEX and Gurobi are CPU based, and none exploit GPUs in their processes, which is an area worth exploring [338].

Compiled and procedural languages, such as C and Fortran, dominate the literature due to their superior performance, as shown in Figure 16a. However, other multi-paradigm multi-paradigm and object-oriented languages (Matlab and python) started to infiltrate the literature due to their simplicity and convenient libraries. Other concurrent programming-oriented languages that might be of interest include Charm, Chapel, Cython, and Julia. Chapel has more advanced parallelism than Julia, while Julia has gained huge popularity since its recent release. Julia is expected to populate future literature due to its heavy emphasis on optimization and C-like performance. In terms of Parallel APIs, the fast adaptation of CUDA as shown in Figure 16b testifies the thirst for massive throughput and suggests that in terms of GPU-based power system optimization studies, there is more to come.

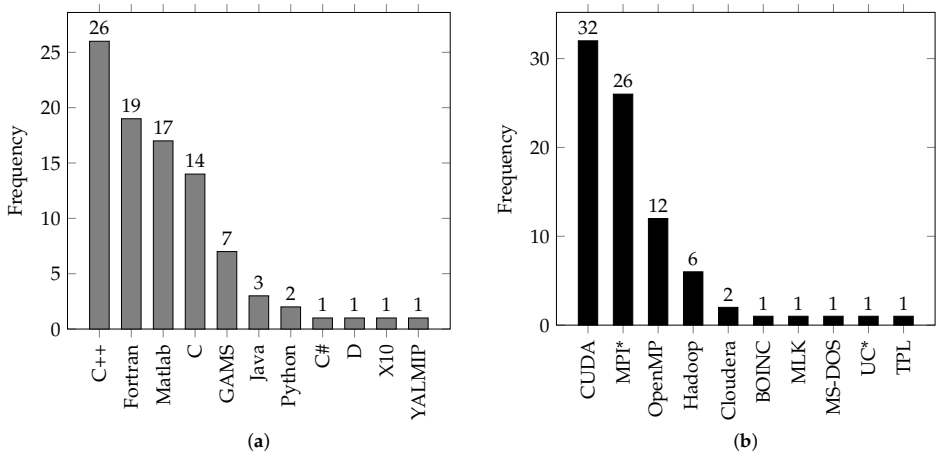

Figure 16. The occurrences of (**a**) programming languages and (**b**) APIs in the reviewed literature. * MPI includes MPICH, mpi4py, MultiMATLAB. UC: Unix Command.

There exist some integrated high-level frameworks designed to scale certain power system studies on HPC, such as BELTISTOS [129], which solves multi-period, security-constrained, and stochastic OPF problems incorporating a multilevel solution strategy implemented in PARADISO. However, when compared to GridPack, this framework seems quite limited. As part of the HIPPO project mentioned earlier [199], PNNL developed the software framework GridPack™ that lowers the barrier for power system research and analysis in creating parallel models for HPC implementation [339].

Grid pack automates processes such as determining the Y-Bus of the network and solving PF equations, integrating algebraic differential equations, coupling simulation components, distributing network and matrix representations of the network, and employing linear and non-linear solvers. GridPack has a partitioner that partitions the network module buses into several processors, where it maximizes the interconnections between buses within the same processor and minimizes the ones between separate processors. It is based on "Parmetis" partitioning software, which achieves graph mesh partitioning, matrix reordering, etc. The matrices of the distributed matrices of the partitioned network are then distributed by mappers, which determine the contribution buses and branches from each processor by getting the dimensions and locations of elements. The math module generates those matrices and supplies linear and non-linear solvers built on the PETSc library. GridPack also has libraries of already developed, ready-to-use parallel applications. This includes different types of contingency analysis, initialization of dynamic simulation, power flow, and voltage stability analysis.

12.2. Challenges in the Literature

This review did not delve into deep comparisons between the different approaches due to the lack of standardization in various aspects of the studies, making it hard to draw meaningful comparisons. These challenges start with network topologies, sizes, a difference in hardware, and a mere lack of information. This is further discussed later in this subsection and can be observed in Table S1. One common point to note is the strong focus on simplifying power flow equations which the dominance of Gauss–Newton and LU-decomposition methods in the literature entails, as shown in Figure 17.

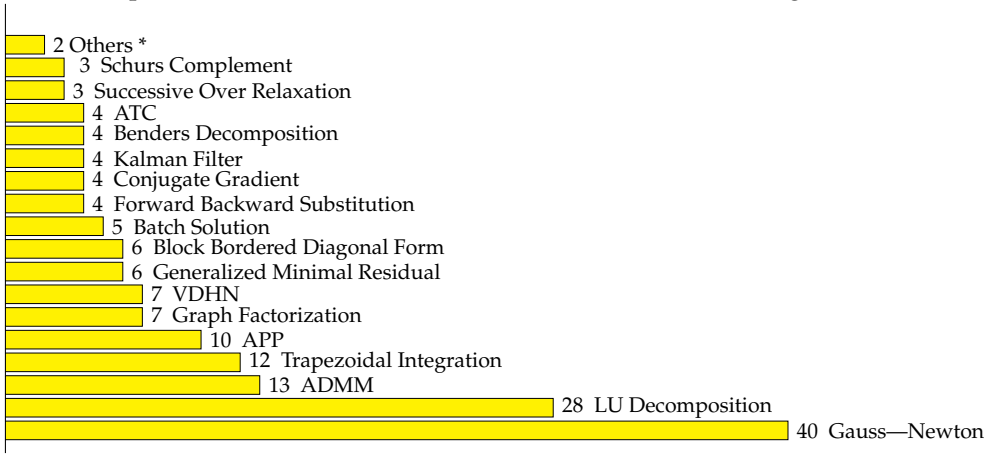

Figure 17. Frequency of decomposition methods and parallelized techniques in the literature. * (Compensation Network Decomposition, Propagation Delay Partitioning, PDIPM, WLS, Barrier, KKT, Waveform Relaxation, W-Matrix, Textured Decomposition, Binary Reduction, Cholesky Factorization, Lagrangian Relaxation, Monte Carlo).

First, the variety of test cases between studies causes a solution universality issue. Many solution approaches exploit the structure and properties of the problem, such as sparsity and asymmetry, which vary with different network topologies and the number of bus interconnections. The effect of that was evident in [122] where the parallel FDPF scheme performed better on the Pan-European system than the Polish system because it had a more orderly topology. Some studies boast remarkable results on massive synthetic bus systems that are an augmentation of the same small bus system connected with tie lines, such as in [274], where the IEEE-39 case was copied and connected over 6000 times. This creates a level of symmetry that does not exist in natural systems, one that certainly affects the performance rendition. Moreover, some SIMD-based studies use made-up or modified power systems that are very dense, which suits what the hardware is designed for but presents a false or exaggerated sense of performance accomplishment, since real systems are generally sparse. Moreover, the speedup metric is often used unfairly. For example,

in decomposition algorithms, solving subproblems is parallelized, then the performance comparison is drawn against the same serial algorithm (i.e., the scalability equation used but referred to as speedup). The parallel algorithm must be compared to the best serial algorithm that can achieve the same task to have a truly fair comparison. This alludes to the fact that there are several approaches for the speedup metric, which should be discussed in parallel algorithm studies for higher transparency, as shown in [256]. Additionally, there are cases in which superlinear speedup is achieved in some studies. This is often due to a playfield change and deep modifications in algorithms' features, leading to unfair comparisons. For example, when the parallel algorithm's cache memory usage is optimized or distributed, memory is used, allowing faster access to memory than serial processes.

The second challenge is the lack of details in the experimentation setup essential for replication. Some papers provide the model of the hardware used without the number of threads or processes used and vice versa. Others claim a parallel application without mentioning the communication scheme used or the number of subproblems created. Furthermore, some papers use or compare iterative algorithms such as "Traditional ADMM/LR" or other generic algorithms without providing the user-adjusted parameters/heuristics involved in tweaking such algorithms, making it impossible to replicate and verify the results. Additionally, barely any of the studies explicitly mention the number and type of constraints and variables generated by the formulation and test cases used. There is also no standardization in the metrics used to evaluate performance. Some studies use absolute speedup; some use relative speedups. Some compare their parallel approach to a different parallel approach, which is weak because the comparison loses its meaning if the proposed parallel approach is inferior to a sequential one. Some suffice by comparing their own parallelized approach to itself applied sequentially (scalability metric), which is problematic because a decomposed task could perform worse than a coupled one when applied sequentially.

The third challenge is the lack of parallel implementation of long-term grid planning models akin to Transmission System Planning or Generation System Planning or their combination. This type of study that helps us plan the transition to the future network struggles with a very small number of factors, accuracy, and uncertainty, and on small test cases that many studies started resorting to decomposition algorithms [340], mainly benders decomposition to decouple investment variables in the models. Yet, it seems that almost none of the studies use or resort to parallel and high-performance computing, which is a huge lost opportunity, as we need to add as many factors as possible to find the real optimal path of transitioning and investment given all the future policies technologies and scenarios that we can speculate at the moment.

The final significant challenge is the lack of standardization in the software and hardware used in the studies. The main issue with software is the variety of solvers used in papers that employ model decomposition schemes. Commercial solvers operate as black boxes that use different techniques, some of which are trade secrets. They are coded with different efficacies and have their bugs and problems, amplifying the confusion in interpretation.

The lack of hardware standardization in the literature has been highlighted since the 1990s [341]. Even very recently, within supposedly comparative work where four different parallel schemes were compared, each scheme was performed on a different supercomputer, and a different test case [248]. The single study that provided a meaningful cross-hardware comparison was [108]. They experimented with different CUDA routines on different but closely related NIVIDIA GPU models, showing that their approach was not superior on every model and proving the importance of hardware normalization.

One way to help tackle the challenge of hardware standardization is by using cloud service instances, such as AWS, as a benchmark, as they are easily accessible globally. Especially since Virtual CPUs (vCPUs) handle the standardization of the heterogeneous hardware and usage (an Elastic Compute Unit is equivalent to the computing power of a 1.0–1.2 GHz 2007 Opteron or 2007 Xeon Processor [342]. Moreover, it fits the industry's

trend of shifting computational power to the Cloud. Additionally, these services come with metric tools that allow the user to look into the actual hardware usage and CPU and memory efficiency of their algorithm. This leads us to the last point: most previous studies merely glance over memory and treat memory resources as a bottleneck rather than a shared and finite resource. More focus on data and memory efficiency is needed. Future studies need to mention the maximum amount of data that needs to be processed and the actual memory usage of their approaches.

12.3. Future and Recommendations

At this point, the role cloud computation would play in power system HPC applications, and the future Grid is almost unquestionable due to its sheer scale and versatility. The Cloud acting as the brain of the Grid fits the notion of the living organism envisioned by many for the Future Grid. In a few studies, the Cloud has been recognized as the centerpiece for distributed computing paradigms such as fog computing and AMI resource leveraging. With that said, the increasing dependence on centralized cloud computing services is antithetical to the goal of energy decentralization/independence. Yet the trend could not be more natural, manifesting a cycle that almost recreates the onset of electrical generation monopolies. The decentralization of computational resources for power systems over micro users, however, does not seem to be that far-fetched of an idea, especially with currently existing applications such as blockchains and volunteer computing.

A lot of the earlier parallel computing studies for power systems modified or created the hardware around algorithms used [81]. This hardware manipulation to suit limited computational purposes is making a comeback due to moors law and other limitations. Analog computing hardware is making a comeback, as it is way more efficient in matrix multiplications. Rather than turning on and off, analog transistors encode a range of numbers based on the conductance magnitude, which is dictated by the gate. Their level of precision, however, makes them mainly suitable for AI chips and algorithms. Provided that their future precision becomes comparable to digital computers, they might be a contender for GPUs in mathematical optimization matrix operations.

The advancement in Quantum Computing research is creating a creeping disruptor of classical computation and algorithms as we know them. It has been shown that current Quantum Computers can solve combinatorial optimization problems that resemble ones related to energy system problems. Namely on the IBMs D-WAVE solving facility location problem [343]. The potential of applying quantum computation for dynamic stability simulations, OPF, and UC was discussed in the early 2000s [344]. In fact, the mixed-integer quadratic UC problem can be transformed into a Quadratic Unconstrained Binary Optimization (QUBO) by discretizing the problem space, a form that can be turned into a quantum program. In a merely experimental effort, this was actually implemented by [343]. The test systems were very small, from 3 to 12 units, and the solutions of the D-Wave were accurate for a smaller number of units, but quickly started deviating. The DC power flow was also implemented on the D-Wave with an HHL algorithm process on a 3-bus test case showing accuracy [345]. These might be the first experimental efforts employing quantum computation for operational power system studies.

Looking back at the studies, one can observe that our current parallel studies do not come anywhere near covering the potential variables of the future Grid. The models are highly simplified and filled with assumptions. The amount of detail, planning factors, and uncertainties are not close to what needs to be considered in grid modernization and future transition. Yet the accuracy and computational performance of the solutions are sometimes unimpressive. Even when decomposition techniques are used, and the created parallel structures are exploited with HPC, we are often faced with not-so-impressive outcomes, probably due to the lack of understanding and ingenuity in employing the parallel and decomposition and parallel techniques. The hardware that is used in many of the studies is often limited to a multi-core processor limiting the potential throughput. A complicated brain is needed to operate the complicated organism that is the future Grid. In the face of

the Grid transformational changes, the power system community needs to start heavily adapting HPC techniques and utilization, incorporating them into future operational and planning studies. Moreover, a high level of transparency and collaboration is needed to accelerate the adaptation of parallel techniques, making such knowledge the norm in power system studies for the Future Grid.

The lack of standardization makes it very hard to replicate techniques from different works. Therefore, it is very important to have a standard framework and minimum information requirements in future power system study publications. This is especially important to ensure published models and techniques' validity, given the scientific reproducibility crisis [346–348]. The following points should serve as a guideline for future parallel studies in the field:

1. A small validation test case, including any modifications, should be presented with all of the parameters and results.
2. All the model expressions must be fully indexed without brevity or detail omissions. This includes both the model pre and post-decomposition. If possible, the full extended model specific to the validation test case should be provided.
3. The pseudocode of the algorithm and flow chart demonstrating the parallel task splitting and synchronization should be included. The values of any tuning factors or heuristic parameters used should be provided.
4. All the platforms, software tools used, parallel strategies, and metrics should be specified. This includes:
    - Operating system (e.g., Windows 10).
    - Coding language (e.g., Python or Julia).
    - Commercial solvers & version (e.g., Gurobi 9.0.1).
    - Parallelization API or pacakge (e.g., mpi4py).
    - Processes communication protocol (e.g., point-to-point or collective, etc.).
    - Machine used (e.g., local university cluster, personal laptop).
    - Type of worker allocated and all its model specs (e.g., 8-core 2.1 GHz 4MB intel i-7400).
    - Memory allocated and technology (e.g., 10GB DDR5 RAM).
    - Number of processes, threads & allocation per worker (e.g., the 100 subproblems were divided on 5CPUs (20 sub-problem/processes per CPU, each subproblem was solved using six threads (12 hyper-threads) automatically allocated by the solver.
    - Average and Peak efficiency of memory and CPU usage. (e.g., CPUs efficiency: 100% peak and 91% average. Memory utilization: 80% peak and 40% average.
5. A test should be carried out on test cases incrementally increasing in size with a variety of network topologies to demonstrate the scalability and universality of the proposed method. An effort should be made to compare the speedup to the fastest known algorithm.

13. Conclusions

In this article, the beginnings of parallel computation and its appearances in power system studies were recounted, and the recent research and literature were reviewed. Several past reviews were cited, distinguishing this work from previously conducted research. The significance of hardware, paradigms and the history of parallel computing was then discussed. Studies of parallel power systems were summarized later, starting by reciting the development of studies up until the 21st century, with emphasis on the most impactful papers from the last decade. Studies included analyses of the stability of the power systems, state estimation and operation of the power systems, and market optimization. The state-of-the-art was also discussed, highlighting the need for standardization in the literature and showcasing the future of computation in power system studies. Given the grid modernization and transition towards net-zero emission, power systems are becoming

increasingly complex, and resorting to high-performance and parallel computing and cloud computing has never been more important.

Supplementary Materials: The following supporting information can be downloaded at: www.mdpi.com/xxx/s1, Table S1: HPC in Power System Literature Data.

Author Contributions: Conceptualization, A.A.-S.; validation, H.Z. and Y.C. formal analysis, A.A.-S.; investigation, A.A.-S.; resources, Y.C.; data curation, A.A.-S.; writing—original draft preparation, A.A.-S.; writing—review and editing, H.Z.; visualization, A.A.-S.; supervision, H.Z. and Y.C.; project administration, H.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: We thank Manuel Zamudio Lopez (University of Calgary) for his comments on the manuscript.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ACOPF	AC Optimal Power Flow
ACPF	AC Power Flow
ADMM	Alternating Direction Method of Multiplier
API	Application Programming Interfaces
APP	Auxiliary Problem Principle
ATC	Analytical Target Cascading
AWS	Amazon Web Services
CA	Contingency Analysis
CPLP	Confidentiality-Preserving Linear Programming
CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
DCOPF	DC Optimal Power Flow
DCPF	DC Power Flow
DER	Distributed Energy Sources
DS	Dynamic Stability
DSM	Demand Side Management
ED	Economic Dispatch
EMT	Electromagnetic Transient
FDPF	Fast-Decoupled Power Flow
FPGA	Field Programmable Gate Arrays
GMRES	Generalized Minimal Residual Method
GPU	Graphical Processing Unit
HPC	High Performance Computing
IoT	Internet of Things
IPM	Interior Point Method
KKT	Karush–Kuhn–Tucker
MC	Monte Carlo
MIMD	Multiple Instruction Multiple Data
MILP	Mixed Integer Linear Programming
MINLP	Mixed Integer Non-Linear Programming
MPI	Message Passing Interface
NLP	Non-Linear Programming
NR	Newton Raphson
OPF	Optimal Power Flow

OS	Operating System
PDIPM	Primal Dual Interior Primal Method
PF	Power Flow
PMU	Phasor Measurement Unit
PNNL	Pacific Northwest National Laboratory
PPF	Probabilistic Power Flow
PVM	Parallel Virtual Machines
RAM	Random Access Memory
RE	Renewable Energy
SCOPF	Security Constrained Optimal Power Flow
SCUC	Security Constrained Unit Commitment
SIMD	Single Instruction Multiple Data
SLR	Surrogate Lagrangian Relaxation
SLS	Sparse Linear Systems
SOR	Successive Over-relaxation
SSA	System Stability Analysis
SSE	System State Estimation
TSCOPF	Transient Stability Constrained OPF
TSCUC	Transient Stability Constrained UC
TSA	Transient Stability Analysis
UC	Unit Commitment
VO	Virtual Organization
VDHN	Very Dishonest Newton Method
WLS	Weighted Least Square

References

1. Climate Change 2022: Impacts, Adaptation and Vulnerability. Available online: https://www.ipcc.ch/report/ar6/wg2/ (accessed on 13 November 2022).
2. Mccool, M.D. *Parallel Programming, Chapter 2*; Number March in 1; Morgan Kaufmann: Burlington, MA, USA, 2012; pp. 39–76. [CrossRef]
3. The White House. Building on Past U.S. Leadership, Including Efforts by States, Cities, Tribes, and Territories, the New Target Aims at 50–52 Percent Reduction in U.S. 2021. Available online: https://bit.ly/3UIWaeK (accessed on 15 November 2021).
4. Coal Exit. 2022. Available online: https://beyond-coal.eu/coal-exit-timeline/ (accessed on 25 May 2022).
5. Carbon Brief. Online. 2020. Available online: bit.ly/3UHwzD4 (accessed on 15 November 2021).
6. IEA. Online. 2022. Available online: https://bit.ly/3QB9fFh (accessed on 25 May 2022).
7. Ritchie, H.; Roser, M.; Rosado, P. Energy. In *Our World in Data*; 2020. Available online: https://ourworldindata.org/renewable-energy (accessed on 13 November 2022).
8. IEA. 2022. Available online: https://bit.ly/3njRMEd (accessed on 25 May 2022).
9. Deloitte Electrification in Industrials. Available online: https://bit.ly/3UIpe62 (accessed on 25 May 2022).
10. IEA. 2021. Available online: https://www.iea.org (accessed on 25 May 2022).
11. González-Torres, M.; Pérez-Lombard, L.; Coronel, J.F.; Maestre, I.R.; Yan, D. A review on buildings energy information: Trends, end-uses, fuels and drivers. *Energy Rep.* **2022**, *8*, 626–637. [CrossRef]
12. B2E Resources. 2021. Available online: https://bit.ly/3HFyceH (accessed on 25 May 2022).
13. Zhu, X.; Liu, K. A systematic review and future directions of the sharing economy: Business models, operational insights and environment-based utilities. *J. Clean. Prod.* **2021**, *290*, 125209. [CrossRef]
14. EIA. *Annual Electric Power Industry Report*; Technical Report; 2021. Available online: https://www.eia.gov/electricity/annual/ (accessed on 13 November 2022).
15. Borenius, S.; Hämmäinen, H.; Lehtonen, M.; Ahokangas, P. Smart grid evolution and mobile communications—Scenarios on the finnish power grid. *Electr. Pow. Syst. Res.* **2021**, *199*, 107367. [CrossRef]
16. Hua, H.; Liu, T.; He, C.; Nan, L.; Zeng, H.; Hu, X.; Che, B. Day-ahead scheduling of power system with short-circuit current constraints considering transmission switching and wind generation. *IEEE Access* **2021**, *9*, 110735–110745. [CrossRef]
17. Daly, P.; Qazi, H.W.; Flynn, D. Rocof-constrained scheduling incorporating non-synchronous residential demand response. *IEEE Trans. Power Syst.* **2019**, *34*, 3372–3383. [CrossRef]
18. Nawaz, A.; Wang, H. Distributed stochastic security constrained unit commitment for coordinated operation of transmission and distribution system. *CSEE J. Power Energy Syst.* **2021**, *7*, 708–718. [CrossRef]
19. Luburić, Z.; Pandžić, H.; Carrión, M. Transmission expansion planning model considering battery energy storage, tcsc and lines using ac opf. *IEEE Access* **2020**, *8*, 203429–203439. [CrossRef]
20. Zhuo, Z.; Zhang, N.; Yang, J.; Kang, C.; Smith, C.; O'Malley, M.J.; Kroposki, B. Transmission expansion planning test system for ac/dc hybrid grid with high variable renewable energy penetration. *IEEE Trans. Power Syst.* **2020**, *35*, 2597–2608. [CrossRef]

21. Gonzalez-Romero, I.C.; Wogrin, S.; Gomez, T. Proactive transmission expansion planning with storage considerations. *Energy Strategy Rev.* **2019**, *24*, 154–165. [CrossRef]
22. Basics, C.T. *State and Trends of Carbon Pricing 2021*; Technical Report; 2021. Available online: https://www.c2es.org/content/carbon-tax-basics/ (accessed on 13 November 2022).
23. Prahastono, I.; Sinisuka, N.I.; Nurdin, M.; Nugraha, H. A review of feed-in tariff model (fit) for photovoltaic (pv). In Proceedings of the 2019 2nd International Conference on High Voltage Engineering and Power Systems (ICHVEPS), Denpasar, Indonesia, 1–4 October 2019; pp. 76–79. [CrossRef]
24. Oprea, S.V.; Bâra, A. Setting the time-of-use tariff rates with nosql and machine learning to a sustainable environment. *IEEE Access* **2020**, *8*, 25521–25530. [CrossRef]
25. Shahzad, Y.; Javed, H.; Farman, H.; Ahmad, J.; Jan, B.; Zubair, M. Internet of energy: Opportunities, applications, architectures and challenges in smart industries. *Comput. Electr. Eng.* **2020**, *86*, 106739. [CrossRef]
26. Kaya, O.; van der Roest, E.; Vries, D.; Keviczky, T. Hierarchical model predictive control for energy management of power-to-x systems. In Proceedings of the 2020 IEEE PES Innovative Smart Grid Technologies Europe (ISGT-Europe), Delft, The Netherlands, 26–28 October 2020; pp. 1094–1098. [CrossRef]
27. Bokkisam, H.R.; Singh, S.; Acharya, R.M.; Selvan, M.P. Blockchain-based peer-to-peer transactive energy system for community microgrid with demand response management. *CSEE J. Power Energy Syst.* **2022**, *8*, 198–211. [CrossRef]
28. Ledger, P. *Utility Tokens and erc-20 icos: Where Are We Now?* Technical Report; Global Blockchain Business Council: Geneva, Switzerland, 2022.
29. Conejo, A.J.; Baringo, L. *Power Electronics and Power Systems Power System Operations*; Springer: Cham, Switzerland, 2019; pp. 21–22.
30. Morales, J.M.; Conejo, A.J.; Madsen, H.; Pinson, P.; Zugno, M. *Integrating Renewables in Electricity Markets*; Springer: New York, NY, USA, 2014; Volume 205, p. 67. [CrossRef]
31. *Standard BAL-001-2*; Reliability Standards for the Bulk Electric Systems of North America. NERC: Atlanta, GA, USA, 2019; pp. 1–1044.
32. Office of Science. *Department of Energy Announces $8.5 Million in High-Performance Algorithms Research for Complex Energy Systems and Processes*; U.S. Department of Energy: Washington, DC, USA, 2022.
33. Ma, F.; Luo, X.; Litvinov, E. Cloud computing for power system simulations at iso new england—Experiences and challenges. *IEEE Trans. Smart Grid* **2016**, *7*, 2596–2603. [CrossRef]
34. Alam, A.; Gopinathan, G.; Shrestha, B.; Zhao, R.; Wu, J.; Xu, R. High preformance computing for operations and trnasmission planning at caiso. In Proceedings of the 2018 IEEE/PES Transmission and Distribution Conference and Exposition, Denver, CO, USA, 16–19 April 2018; pp. 3–7.
35. Chen, Y.; Wang, F.; Ma, Y.; Yao, Y. A distributed framework for solving and benchmarking security constrained unit commitment with warm start. *IEEE Trans. Power Syst.* **2020**, *35*, 711–720. [CrossRef]
36. Constantinescu, E.M.; Zavala, V.M.; Rocklin, S.L.M.; Anitescu, M. A computational framework for uncertainty quantification and stochastic optimization in unit commitment with wind power generation. *IEEE Trans. Power Syst.* **2011**, *26*, 431–441. [CrossRef]
37. O'Neill, R.; Castillo, A.; Cain, B. *The IV Formulation and Linearizations of the AC Optimal Power Flow Problem*; Technical Report; Federal Energy Regulatory Commission: Washington, DC, USA, 2013. Available online: https://www.ferc.gov/sites/default/files/2020-04/acopf-2-iv-linearization.pdf (accessed on 13 November 2022).
38. Happ, H.H.; Undrill, J.M. Multicomputer configurations and diakoptics: Real power flow in power pools. *IEEE Trans. Power Appl. Syst.* **1969**, *88*, 789–796. [CrossRef]
39. Ewart, D.N.; Kirchmayer, L.K. Automation and utility system security. *IEEE Spectr.* **1971**, *8*, 37–42. [CrossRef]
40. Narita, S.; Hammam, M.S.A.A. Multicomputer control of system voltage and reactive power on real-time basis. *IEEE Trans. Power Appl. Syst.* **1973**, *92*, 278–286. [CrossRef]
41. Wu, F. Solution of large-scale networks by tearing. *IEEE Trans. Circuits Syst.* **1976**, *23*, 706–713. [CrossRef]
42. Anderson, P.M. *Exploring Applications of Parallel Processing to Power System Analysis Problems*; Seminar Proceeding Special Report; Electric Power Research Institute: Palo Alto, CA, USA, 1977.
43. Podmore, R. Application of an array processor for power system network computations. *Build. Environ.* **1982**, *17*, 95–105.
44. Tylavsky, C.c.D.J.; Bose, A.; Alvarado, M.F.; Betancourt, R.; Clements, K.; Heydt, G.T.; Huang, G.; Ilic, M.; Scala, M.L.; Pai, M.A.; et al. Parallel processing in power systems computation. *IEEE Trans. Power Syst.* **1992**, *7*, 629–638. [CrossRef]
45. Bose, A. Parallel processing in dynamic simulation of power systems. *Sadhana* **1993**, *18*, 815–841. [CrossRef]
46. Falcao, D.M. High Perfromance Computing in Power System Applications. In *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*; Springer: Berlin/Heidelberg, Germany, 1996; Volume 1215. [CrossRef]
47. Ramesh, V.C. On distributed computing for on-line power system applications. *Int. J. Electr. Power Energy Syst.* **1996**, *18*, 527–533. [CrossRef]
48. Ii, R.C.G.; Wang, L.; Alam, M. High performance computing for electric power systems: Applications and trends. In Proceedings of the 2011 IEEE Power and Energy Society General Meeting, Detroit, MI, USA, 24–28 July 2011; pp. 1–8.
49. Green, R.C.; Wang, L.; Alam, M. Applications and trends of high performance computing for electric power systems: Focusing on smart grid. *IEEE Trans. Smart Grid* **2013**, *4*, 922–931. [CrossRef]

50. Tan, L.; Jiang, J. Chapter 14—Hardware and software for digital signal processors. In *Digital Signal Processing*, 3rd ed.; Tan, L., Jiang, J., Eds.; Academic Press: Cambridge, MA, USA, 2019; pp. 727–784. [CrossRef]
51. Aspray, W. The intel 4004 microprocessor: What constituted invention? *IEEE Ann. Hist. Comput.* **1997**, *19*, 4–15. [CrossRef]
52. Stringer, L. Vectors: How the Old Became New again in Supercomputing. 2016. Available online: https://www.hpcwire.com/2016/09/26/vectors-old-became-new-supercomputing/ (accessed on 13 November 2022).
53. Hey, A.J.G. Supercomputing with transputers—Past, present and future. In Proceedings of the 4th International Conference on Supercomputing (ICS '90), Amsterdam, The Netherlands, 11–15 June 1990; Association for Computing Machinery: New York, NY, USA, 1990; pp. 479–489. [CrossRef]
54. Bose, P. Encyclopedia of parallel computing. In *Encyclopedia of Parallel Computing*; Padua, D., Ed.; Springer: Boston, MA, USA, 2011; Chapter Power Wall; pp. 1593–1608. [CrossRef]
55. Intel. *8th and 9th Generation Intel Core Processor Families and Intel Xeon E Processor Families*; Technical Report; Intel Coorperation: Mountain View, CA, USA, 2020.
56. Intel. *Intel oneAPI DPC++/C++ Compiler Developer Guide and Reference*; Intel Coorperation: Mountain View, CA, USA, 2022. Available online: https://bit.ly/3X69OKu (accessed on 13 November 2022).
57. Intel. *Intel AVX-512—Instruction Set for Packet Processing*; Technical Report; Intel Corporation: Mountain View, CA, USA, 2021. Available online: https://intel.ly/3GiWooH (accessed on 13 November 2022).
58. Intel. *Intel Xeon Silver 4110 Processor*; Technical Report; Inter Coorperation: Mountain View, CA, USA, 2018.
59. Computing, A. *Ampere Altra Max 64-bit Multi-Core Processor Features*; Technical Report; Ampere Computing: Santa Clara, CA, USA, 2021; Available online: https://bit.ly/3hNqTZG (accessed on 13 November 2022).
60. Teich, P. Nvidia Dominates the Market for Cloud AI Accelerators More Than You Think. 2021. Available online: https://bit.ly/3QIvdqa (accessed on 13 November 2022).
61. Navarro, C.A.; Hitschfeld-Kahler, N.; Mateu, L. A survey on parallel computing and its applications in data-parallel problems using gpu architectures. *Commun. Comput. Phys.* **2014**, *15*, 285–329. [CrossRef]
62. ARM. ARM Technology Is Defining the Future of Computing: Record Royalties Highlight Increasing Diversity of Products and Market Segment Growth. 2022. Available online: https://www.arm.com/company/news/2022/11/arm-achieves-record-royalties-q2-fy-2022 (accessed on 13 November 2022).
63. Leiner, B.M.; Cerf, V.G.; Clark, D.D.; Kahn, R.E.; Kleinrock, L.; Lynch, D.C.; Postel, J.; Roberts, L.G.; Wolff, S. *Brief History of the Internet*; Technical Report; Internet Society: Reston, VA, USA, 1997.
64. Spinellis, D. A repository of unix history and evolution. *Empir. Softw. Engg.* **2017**, *22*, 1372–1404. [CrossRef]
65. Stott, M. *ARCNETworks*; Technical Report; Arcnet Trade Association: Downers Grove, IL, USA, 1998.
66. Majidha Fathima, K.M.; Santhiyakumari, N. A survey on evolution of cloud technology and virtualization. In Proceedings of the 2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV), Tirunelveli, India, 4–6 February 2021; pp. 428–433. [CrossRef]
67. Beck, A. High Throughput Computing: An Interview with Miron Livny. 2021. Available online: https://bit.ly/3y2Tuje (accessed on 13 November 2022).
68. Raicu, I.; Foster, I.T.; Yong, Z. Many-task computing for grids and supercomputers. In Proceedings of the 2008 Workshop on Many-Task Computing on Grids and Supercomputers, Austin, TX, USA, 17 November 2008; pp. 1–11. [CrossRef]
69. Globus. Globus Toolkit. 2021. Available online: https://toolkit.globus.org/ (accessed on 13 November 2022).
70. Amazon. *Overview of Amazon Web Services: Aws Whitepaper*; Technical Report; Amazon Web Services: Seattle, WA, USA, 2022; Available online: https://docs.aws.amazon.com/whitepapers/latest/aws-overview/introduction.html (accessed on 13 November 2022).
71. Geist, G.A.; Sunderam, V.S. Network-based concurrent computing on the pvm system. *Concurr. Pract. Exper.* **1992**, *4*, 293–311. [CrossRef]
72. UTM. What is UTM? 2022. Available online: https://docs.getutm.app/ (accessed on 13 November 2022).
73. Apptainer. The Container System for Secure High Performance Computing. 2022. Available online: https://apptainer.org/ (accessed on 13 November 2022).
74. Docker. Docker Engine Overview. 2022. Available online: https://docs.docker.com/engine/ (accessed on 13 November 2022).
75. Mersenne. Great Internet Mersenne Prime Search—Primenet. 2022. Available online: https://www.mersenne.org/ (accessed on 15 March 2012).
76. BOINC. News from Boinc Projects. 2022. Available online: https://boinc.berkeley.edu/ (accessed on 15 March 2012).
77. Apple. Macos Server. 2022. Available online: https://www.apple.com/macos/server/ (accessed on 15 March 2012).
78. DASK. Task Graph Optimization. 2022. Available online: https://docs.dask.org/en/stable/optimize.html (accessed on 15 March 2012).
79. Adams, L. Cosmic ray effects in microelectronics. *Microelectron J.* **1985**, *16*, 17–29. [CrossRef]
80. Government, A. *Aviation Safety Investigations & Reports: In-Flight Upset—Airbus A330-303, VH-QPA, 154 km West of Learmonth, WA, 7 October 2008*; Technical Report; Australian Transport Safety Bureau: Canberra, Australia, 2021.
81. Taoka, H.; Abe, S.; Takeda, S. Multiprocessor system for power system analysis. *Annu. Rev. Autom. Program.* **1981**, *11*, 101–106. [CrossRef]

82. Ward, J.B.; Hale, H.W. Digital computer solution of power-flow problems [includes discussion]. *Trans. Am. Inst. Electr. Eng. Part III Power Appar. Syst.* **1956**, *75*, 398–404. [CrossRef]
83. Hulskamp, J.P.; Chan, S.; Fazio, J.F. Power flow outage studies using an array processor. *IEEE Trans. Power Appl. Syst.* **1982**, *101*, 254–261. [CrossRef]
84. Foertsch, J.; Johnson, J.; Nagvajara, P. Jacobi load flow accelerator using fpga. In Proceedings of the 37th Annual North American Power Symposium, Ames, IA, USA, 25 October 2005; pp. 448–454. [CrossRef]
85. Wang, X.; Ziavras, S.G.; Nwankpa, C.; Johnson, J.; Nagvajara, P. Parallel solution of newton's power flow equations on configurable chips. *Int. J. Electr. Power Energy Syst.* **2007**, *29*, 422–431. [CrossRef]
86. Happ, H.H. Special cases of orthogonal networks—Tree and link. *IEEE Trans. Power Appl. Syst.* **1966**, *85*, 880–891. [CrossRef]
87. Happ, H.H. Z diakoptics—Torn subdivisions radially attached. *IEEE Trans. Power Appl. Syst.* **1967**, *86*, 751–769. [CrossRef]
88. Carre, B.A. Solution of load-flow problems by partitioning systems into trees. *IEEE Trans. Power Appl. Syst.* **1968**, *87*, 1931–1938. [CrossRef]
89. Andretich, R.G.; Brown, H.E.; Happ, H.H.; Person, C.E. The piecewise solution of the impedance matrix load flow. *IEEE Trans. Power Appl. Syst.* **1968**, *87*, 1877–1882. [CrossRef]
90. Takatoo, M.; Abe, S.; Bando, T.; Hirasawa, K.; Goto, M.; Kato, T.; Kanke, T. Floating vector processor for power system simulation. *IEEE Power Eng. Rev.* **1985**, *5*, 29–30. [CrossRef]
91. Lau, K.; Tylavsky, D.J.; Bose, A. Coarse grain scheduling in parallel triangular factorization and solution of power system matrices. *IEEE Trans. Power Syst.* **1991**, *6*, 708–714. [CrossRef]
92. Huang, G.; Ongsakul, W. An adaptive sor algorithm and its parallel implementation for power system applications. In Proceedings of the 1994 6th IEEE Symposium on Parallel and Distributed Processing, Dallas, TX, USA, 26–29 October 1994; pp. 84–91. [CrossRef]
93. Gomez, A.; Betancourt, R. Implementation of the fast decoupled load flow on a vector computer. *IEEE Trans. Power Syst.* **1990**, *5*, 977–983. [CrossRef]
94. Housos, E.C.; Wing, O. Parallel optimization with applications to power systems. *IEEE Trans. Power Appl. Syst.* **1982**, *101*, 244–248. [CrossRef]
95. Chen, S.D.; Chen, J.F. Fast load flow using multiprocessors. *Int. J. Electr. Power Energy Syst.* **2000**, *22*, 231–236. [CrossRef]
96. Chen, S.D.; Chen, J.F. A novel approach based on global positioning system for parallel load flow analysis. *Int. J. Electr. Power Energy Syst.* **2005**, *27*, 53–59. [CrossRef]
97. Feng, T.; Flueck, A.J. A message-passing distributed-memory newton-gmres parallel power flow algorithm. In Proceedings of the IEEE Power Engineering Society Summer Meeting, Chicago, IL, USA, 21–25 July 2002; Volume 3, pp. 1477–1482. [CrossRef]
98. Li, Y.; Li, F.; Li, W. Parallel power flow calculation based on multi-port inversed matrix method. In Proceedings of the 2010 International Conference on Power System Technology, Hangzhou, China, 24–28 October 2010; pp. 1–6. [CrossRef]
99. Sun, H.; Guo, Q.; Zhang, B.; Guo, Y.; Li, Z.; Wang, J. Master–slave-splitting based distributed global power flow method for integrated transmission and distribution analysis. *IEEE Trans. Smart Grid* **2015**, *6*, 1484–1492. [CrossRef]
100. Su, X.; Liu, T.; Wu, L. Fine-grained fully parallel power flow calculation by incorporating bbdf method into a multistep nr algorithm. *IEEE Trans. Power Syst.* **2018**, *33*, 7204–7214. [CrossRef]
101. Garcia, N. Parallel power flow solutions using a biconjugate gradient algorithm and a newton method: A gpu-based approach. In Proceedings of the IEEE PES General Meeting, Detroit, MI, USA, 24–29 July 2010; pp. 1–4. [CrossRef]
102. Singh, J.; Aruni, I. Accelerating power flow studies on graphics processing unit. In Proceedings of the 2010 Annual IEEE India Conference (INDICON), Kolkata, India, 17–19 December 2010; pp. 1–5. [CrossRef]
103. Dağ, H.; Soykan, G. Power flow using thread programming. In Proceedings of the 2011 IEEE Trondheim PowerTech, Trondheim, Norway, 19–23 June 2011; pp. 1–5. [CrossRef]
104. Vilachá, C.; Moreira, J.C.; Míguez, E.; Otero, A.F. Massive jacobi power flow based on simd-processor. In Proceedings of the 2011 10th International Conference on Environment and Electrical Engineering, Rome, Italy, 1–7 May 2011; pp. 1–4. [CrossRef]
105. Yang, M.; Sun, C.; Li, Z.; Cao, D. An improved sparse matrix-vector multiplication kernel for solving modified equation in large scale power flow calculation on cuda. In Proceedings of the 7th International Power Electronics and Motion Control Conference, Harbin, China, 2–5 June 2012; Volume 3, pp. 2028–2031. [CrossRef]
106. Xue, L.; Fangxing, L.; Clark, J.M. Exploration of multi-frontal method with gpu in power flow computation. In Proceedings of the 2013 IEEE Power Energy Society General Meeting, Vancouver, BC, Canada, 21-25 July 2013; pp. 1–5. [CrossRef]
107. Ablakovic, D.; Dzafic, I.; Kecici, S. Parallelization of radial three-phase distribution power flow using gpu. In Proceedings of the 2012 3rd IEEE PES Innovative Smart Grid Technologies Europe (ISGT Europe), Berlin, Germany, 14–17 October 2012; pp. 1–7. [CrossRef]
108. Blaskiewicz, P.; Zawada, M.; Balcerek, P.; Dawidowski, P. An application of gpu parallel computing to power flow calculation in hvdc networks. In Proceedings of the 2015 23rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, Turku, Finland, 4–6 March 2015; pp. 635–641. [CrossRef]
109. Huang, R.H.; Jin, S.; Chen, Y.; Diao, R.; Palmer, B.; Qiuhua. Faster than real-time dynamic simulation for large-size power system with detailed dynamic models using high-performance computing platform. In Proceedings of the 2017 IEEE Power & Energy Society General Meeting, Chicago, IL, USA, 16–20 July 2017.

110. Guo, C.; Jiang, B.; Yuan, H.; Yang, Z.; Wang, L.; Ren, S. Performance comparisons of parallel power flow solvers on gpu system. In Proceedings of the 2012 IEEE International Conference on Embedded and Real-Time Computing Systems and Applications Performance 2012, Seoul, Republic of Korea, 19–22 August 2012; pp. 232–239. [CrossRef]
111. Wang, Y.; Wu, L.; Wang, S. A fully-decentralized consensus-based admm approach for dc-opf with demand response. *IEEE Trans. Smart Grid* **2017**, *8*, 2637–2647. [CrossRef]
112. Marin, M.; Defour, D.; Milano, F. Asynchronous power flow on graphic processing units. In Proceedings of the 2017 25th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP), St. Petersburg, Russia, 6–8 March 2017; pp. 255–261. [CrossRef]
113. Gnanavignesh, R.; Shenoy, U.J. Gpu-accelerated sparse lu factorization for power system simulation. In Proceedings of the 2019 IEEE PES Innovative Smart Grid Technologies Europe (ISGT-Europe), Bucharest, Romania, 29 September–2 October 2019; pp. 1–5. [CrossRef]
114. Tang, K.; Fang, R.; Wang, X.; Dong, S.; Song, Y. Mass expression evaluation parallel algorithm based on 'expression forest' and its application in power system calculation. In Proceedings of the 2019 IEEE Power Energy Society General Meeting (PESGM), Atlanta, GA, USA, 4–8 August 2019; pp. 1–5. [CrossRef]
115. Araújo, I.; Tadaiesky, V.; Cardoso, D.; Fukuyama, Y.; Santana, Á. Simultaneous parallel power flow calculations using hybrid cpu-gpu approach. *Int. J. Electr. Power Energy Syst.* **2019**, *105*, 229–236. [CrossRef]
116. Yoon, D.H.; Han, Y. Parallel power flow computation trends and applications: A review focusing on gpu. *Energies* **2020**, *13*, 2147. [CrossRef]
117. Daher Daibes, J.V.; Brown Do Coutto Filho, M.; Stacchini de Souza, J.C.; Gonzalez Clua, E.W.; Zanghi, R. Experience of using graphical processing unit in power flow computation. *Concurr. Comput. Pract. Exp.* **2022**, *34*, e6762. [CrossRef]
118. Abhyankar, S.; Peles, S.; Rutherford, R.; Mancinelli, A. Evaluation of ac optimal power flow on graphical processing units. In Proceedings of the 2021 IEEE Power Energy Society General Meeting (PESGM), Washington, DC, USA, 25–29 July 2021; pp. 1–5. [CrossRef]
119. Dag, H.; Alvarado, F.L. Computation-free preconditioners for the parallel solution of power system problems. *IEEE Trans. Power Syst.* **1997**, *12*, 585–591. [CrossRef]
120. Li, X.; Li, F. Gpu-based power flow analysis with chebyshev preconditioner and conjugate gradient method. *Electr. Pow. Syst. Res.* **2014**, *116*, 87–93. [CrossRef]
121. Li, X.; Li, F. Gpu-based two-step preconditioning for conjugate gradient method in power flow. In Proceedings of the 2015 IEEE Power Energy Society General Meeting, Denver, CO, USA, 26–30 July 2015; pp. 1–5. [CrossRef]
122. Li, X.; Li, F.; Yuan, H.; Cui, H.; Hu, Q. Gpu-based fast decoupled power flow with preconditioned iterative solver and inexact newton method. *IEEE Trans. Power Syst.* **2017**, *32*, 2695–2703. [CrossRef]
123. Wang, M.; Chen, Y.; Huang, S. Gpu-based power flow analysis with continuous newton's method. In Proceedings of the IEEE Conference on Energy Internet and Energy System Integration (EI2), Beijing, China, 20–22 October 2018.
124. Su, X.; He, C.; Liu, T.; Wu, L. Full parallel power flow solution: A gpu-cpu-based vectorization parallelization and sparse techniques for newton-raphson implementation. *IEEE Trans. Smart Grid* **2020**, *11*, 1833–1844. [CrossRef]
125. Zhou, G.; Feng, Y.; Bo, R.; Zhang, T. Gpu-accelerated sparse matrices parallel inversion algorithm for large-scale power systems. *Int. J. Electr. Power Energy Syst.* **2019**, *111*, 34–43. [CrossRef]
126. Zhou, G.; Bo, R.; Chien, L.; Zhang, X.; Yang, S.; Su, D. Gpu-accelerated algorithm for online probabilistic power flow. *IEEE Trans. Power Syst.* **2018**, *33*, 1132–1135. [CrossRef]
127. Zhou, G.; Bo, R. Gpu-based batch lu-factorization solver for concurrent analysis of massive power flows. *IEEE Trans. Smart Grid* **2017**, *32*, 4975–4977. [CrossRef]
128. Kardoš, J.; Kourounis, D.; Schenk, O. Two-level parallel augmented schur complement interior-point algorithms for the solution of security constrained optimal power flow problems. *IEEE Trans. Power Syst.* **2020**, *35*, 1340–1350. [CrossRef]
129. Beltistos. Beltistos. 2022. Available online: http://www.beltistos.com/ (accessed on 13 November 2022).
130. Carpentier, J. Optimal power flows. *Int. J. Electr. Power Energy Syst.* **1979**, *1*, 3–15. [CrossRef]
131. Huneault, M.; Galiana, F.D. A survey of the optimal power flow literature. *IEEE Trans. Power Syst.* **1991**, *6*, 762–770. [CrossRef]
132. Shoults, R.R.; Sun, D.T. Optimal power flow based upon p-q decomposition. *IEEE Trans. Power Appl. Syst.* **1982**, *101*, 397–405. [CrossRef]
133. Talukdar, S.N.; Giras, T.C.; Kalyan, V.K. Decompositions for optimal power flows. *IEEE Trans. Power Appl. Syst.* **1983**, *PAS-102*, 3877–3884. [CrossRef]
134. Monticelli, A.; Pereira, M.V.F.; Granville, S. Security-constrained optimal power flow with post-contingency corrective rescheduling. *IEEE Power Eng. Rev.* **1987**, *7*, 43–44. [CrossRef]
135. Huang, G.M.; Hsieh, S.C. Exact convergence of a parallel textured algorithm for constrained economic dispatch control problems. In Proceedings of the 31st IEEE Conference on Decision and Control, Tucson, AZ, USA, 16–18 December 1992; pp. 570–575. [CrossRef]
136. Huang, G.; Hsieh, S.C. A parallel had-textured algorithm for constrained economic dispatch control problems. *IEEE Trans. Power Syst.* **1995**, *10*, 1553–1558. [CrossRef]
137. Teixeira, M.J.; Pinto, H.J.C.P.; Pereira, M.V.F.; McCoy, M.F. Developing concurrent processing applications to power system planning and operations. *IEEE Trans. Power Syst.* **1990**, *5*, 659–664. [CrossRef]

138. Kim, B.H.; Baldick, R. Coarse-grained distributed optimal power flow. *IEEE Trans. Power Syst.* **1997**, *12*, 932–939. [CrossRef]
139. Baldick, R.; Kim, B.H.; Chase, C.; Luo, Y. A fast distributed implementation of optimal power flow. *IEEE Trans. Power Syst.* **1999**, *14*, 858–864. [CrossRef]
140. Anese, E.D.; Zhu, H.; Giannakis, G.B. Distributed optimal power flow for smart microgrids. *IEEE Trans. Smart Grid* **2013**, *4*, 1464–1475. [CrossRef]
141. Liu, K.; Li, Y.; Sheng, W. The decomposition and computation method for distributed optimal power flow based on message passing interface (mpi). *Int. J. Electr. Power Energy Syst.* **2011**, *33*, 1185–1193. [CrossRef]
142. Kim, B.H.; Baldick, R. A comparison of distributed optimal power flow algorithms. *IEEE Trans. Power Syst.* **2000**, *15*, 599–604. [CrossRef]
143. Talukdar, S.; Ramesh, V.C. A multi-agent technique for contingency constrained optimal power flows. *IEEE Trans. Power Syst.* **1994**, *9*, 855–861. [CrossRef]
144. Rodrigues, M.; Saavedra, O.R.; Monticelli, A. Asynchronous programming model for the concurrent solution of the security constrained optimal power flow problem. *IEEE Trans. Power Syst.* **1994**, *9*, 2021–2027. [CrossRef]
145. Wei, Q.; Flueck, A.J.; Feng, T. A new parallel algorithm for security constrained optimal power flow with a nonlinear interior point method. In Proceedings of the IEEE Power Engineering Society General Meeting, San Francisco, CA, USA, 12–16 June 2005; Volume 1, pp. 447–453. [CrossRef]
146. Borges, C.L.T.; Alves, J.M.T. Power system real time operation based on security constrained optimal power flow and distributed processing. In Proceedings of the 2007 IEEE Lausanne Power Tech, Lausanne, Switzerland, 1–5 July 2007; pp. 960–965. [CrossRef]
147. Yuan, Z.; Hesamzadeh, M.R.; Cui, Y.; Bertling Tjernberg, L. Applying high performance computing to probabilistic convex optimal power flow. In Proceedings of the 2016 International Conference on Probabilistic Methods Applied to Power Systems, PMAPS 2016—Proceedings, Beijing, China, 16–20 October 2016. [CrossRef]
148. Farivar, M.; Low, S.H. Branch flow model: Relaxations and convexification—Part I. *IEEE Trans. Power Syst.* **2013**, *28*, 2554–2564. [CrossRef]
149. Yuan, Z.; Hesamzadeh, M.R. A modified benders decomposition algorithm to solve second-order cone ac optimal power flow. *IEEE Trans. Smart Grid* **2019**, *10*, 1713–1724. [CrossRef]
150. Lan, T.; Huang, G.M. An intelligent parallel scheduling method for optimal transmission switching in power systems with batteries. In Proceedings of the 19th International Conference on Intelligent System Application to Power Systems (ISAP), San Antonio, TX, USA, 17–20 September 2017.
151. Zhang, Q.; Sahraei-ardakani, M. Distributed dcopf with flexible transmission. *Electr. Power Syst. Res.* **2018**, *154*, 37–47. [CrossRef]
152. Mohammadi, J.; Zhang, J.; Kar, S.; Hug, G.; Moura, J.M.F. Multilevel distributed approach for dc optimal power flow. In Proceedings of the 2015 IEEE Global Conference on Signal and Information Processing (GlobalSIP), Orlando, FL, USA, 14–16 December 2015; pp. 1121–1125. [CrossRef]
153. Sadnan, R.; Dubey, A. Distributed optimization using reduced network equivalents for radial power distribution systems. *IEEE Trans. Power Syst.* **2021**, *36*, 3645–3656. [CrossRef]
154. Molzahn, D.K.; Dorfler, F.; Sandberg, H.; Low, S.H.; Chakrabarti, S.; Baldick, R.; Lavaei, J. A survey of distributed optimization and control algorithms for electric power systems. *IEEE Trans. Smart Grid* **2017**, *8*, 2941–2962. [CrossRef]
155. Tu, S.; Wächter, A.; Wei, E. A two-stage decomposition approach for ac optimal power flow. *arXiv* **2020**, arXiv:2002.08003.
156. DeMiguel, V.; Murray, W. A local convergence analysis of bilevel decomposition algorithms. *Optim. Eng.* **2006**, *7*, 99–133. [CrossRef]
157. Kerr, R.H.; Scheidt, J.L.; Fontanna, A.J.; Wiley, J.K. Unit commitment. *IEEE Trans. Power Appl. Syst.* **1966**, *85*, 417–421. [CrossRef]
158. Ma, X.; Song, H.; Hong, M.; Wan, J.; Chen, Y.; Zak, E. The security-constrained commitment and dispatch for midwest iso day-ahead co-optimized energy and ancillary service market. In Proceedings of the 2009 IEEE Power Energy Society General Meeting, Minneapolis, MN, USA, 25–29 July 2009; pp. 1–8. [CrossRef]
159. Shiina, T.; Birge, J.R. Stochastic unit commitment problem. *Int. Trans. Oper. Res.* **2003**, *11*, 19–32. [CrossRef]
160. Papavasiliou, A.; Oren, S.S. A comparative study of stochastic unit commitment and security-constrained unit commitment using high performance computing. In Proceedings of the 2013 European Control Conference (ECC 2013), Zurich, Switzerland, 17–19 July 2013; pp. 2507–2512. [CrossRef]
161. Ji, X.; Zhang, Y.; Han, X.; Ye, P.; Xu, B.; Yu, Y. Electrical power and energy systems multi-level interactive unit commitment of regional power system. *Int. J. Electr. Power Energy Syst.* **2021**, *125*, 106464. [CrossRef]
162. Wu, L.; Shahidehpour, M.; Li, T. Stochastic security-constrained unit commitment. *IEEE Trans. Power Syst.* **2007**, *2*, 800–811. [CrossRef]
163. Liu, J.; Laird, C.D.; Scott, J.K.; Watson, J.P.; Castillo, A. Global solution strategies for the network-constrained unit commitment problem with ac transmission constraints. *IEEE Trans. Power Syst.* **2019**, *34*, 1139–1150. [CrossRef]
164. Wong, K.P.; Wong, Y.W. Short-term hydrothermal scheduling with reservoir volume constraints. Ii. Parallel simulated annealing approach. In Proceedings of the 1993 2nd International Conference on Advances in Power System Control, Operation and Management (APSCOM-93), Hong Kong, China, 7–10 December 1993; Volume 2, pp. 565–570.
165. Numnonda, T.; Annakkage, U.D.; Pahalawaththa, N.C. Unit commitment using stochastic optimization. In Proceedings of the International Conference on Intelligent Systems Applications to Power Systems (ISAP), Orlando, FL, USA, 28 January–2 February 1996; Volume 1, pp. 428–433. [CrossRef]

166. Misra, N.; Baghzouz, Y. Implementation of the unit commitment problem on supercomputers. *IEEE Trans. Power Syst.* **1994**, *9*, 305–310. [CrossRef]
167. Lau, K.K.; Kumar, M.J. Parallel implementation of the unit commitment problem on nows. In Proceedings of the High Performance Computing on the Information Superhighway (HPC Asia '97), Seoul, Republic of Korea, 28 April–2 May 1997; pp. 128–133. [CrossRef]
168. Yang, H.T.; Yang, P.C.; Huang, C.L. Optimization of unit commitment using parallel structures of genetic algorithm. In Proceedings of the 1995 International Conference on Energy Management and Power Delivery (EMPD '95), Singapore, 21–23 November 1995; Volume 2, pp. 577–582. [CrossRef]
169. Yang, H.T.; Yang, P.C.; Huang, C.L. A parallel genetic algorithm approach to solving the unit commitment problem: Implementation on the transputer networks. *IEEE Power Eng. Rev.* **1997**, *17*, 58–59.
170. Murillo-s, C.E.; Thomas, R.J. Parallel processing implementation of the unit commitment problem with full ac power flow constraints. In Proceedings of the 33rd Annual Hawaii International Conference on System Sciences, Maui, HI, USA, 4–7 January 2000, pp. 1–9.
171. Baslis, C.G.; Papadakis, S.E.; Bakirtzis, A.G. Simulation of optimal medium-term hydro-thermal system operation by grid computing. *IEEE Trans. Power Syst.* **2009**, *24*, 1208–1217. [CrossRef]
172. Papavasiliou, A.; Oren, S.S.; O'Neill, R.P. No titlereserve requirements for wind power integration: A scenario-based stochastic programming framework. *IEEE Trans. Power Syst.* **2011**, *26*, 2197–2206. [CrossRef]
173. Papavasiliou, A.; Oren, S.S.; Rountree, B. Applying high performance computing to transmission-constrained stochastic unit commitment for renewable energy integration. *IEEE Trans. Power Syst.* **2015**, *30*, 1109–1120. [CrossRef]
174. Aravena, I.; Papavasiliou, A. A distributed asynchronous algorithm for the two-stage stochastic unit commitment problem. In Proceedings of the 2015 IEEE Power Energy Society General Meeting, Denver, CO, USA, 26–30 July 2015; pp. 1–5. [CrossRef]
175. Bai, Y.; Zhong, H.; Xia, Q.; Kang, C.; Xie, L. A decomposition method for network-constrained unit commitment with ac power flow constraints. *Energy* **2015**, *88*, 595–603. [CrossRef]
176. Bai, X.; Wei, H.; Fujisawa, K.; Wang, Y. Semidefinite programming for optimal power flow problems. *Int. J. Electr. Power Energy Syst.* **2008**, *30*, 383–392. [CrossRef]
177. Khanabadi, M.; Wang, C. Security-constrained unit commitment considering voltage stability: A parallel solution. In Proceedings of the 2016 North American Power Symposium (NAPS), Denver, CO, USA, 18–20 September 2016.
178. Kargarian, A.; Raoofat, M.; Mohammadi, M. Reactive power market management considering voltage control area reserve and system security. *Appl. Energy* **2011**, *88*, 3832–3840. [CrossRef]
179. Sioshansi, R.; Oren, S.; O'neill, R. Chapter 6—The cost of anarchy in self-commitment-based electricity markets. In *Competitive Electricity Markets*; Sioshansi, F.P., Ed.; Elsevier Global Energy Policy and Economics Series; Elsevier: Oxford, UK, 2008; pp. 245–266. [CrossRef]
180. Feizollahi, M.J.; Costley, M.; Ahmed, S.; Grijalva, S. Large-scale decentralized unit commitment. *Int. J. Electr. Power Energy Syst.* **2015**, *73*, 97–106. [CrossRef]
181. Doostizadeh, M.; Aminifar, F.; Lesani, H.; Ghasemi, H. Multi-area market clearing in wind-integrated interconnected power systems: A fast parallel decentralized method. *Energy Convers. Manag.* **2016**, *113*, 131–142. [CrossRef]
182. Ramanan, P.; Yildirim, M.; Gebraeel, N.; Ramanan, P.; Gebraeel, N. Asynchronous decentralized framework for unit commitment in power systems. *IEEE Trans. Power Syst.* **2017**, *108*, 665–674. [CrossRef]
183. Shi, W.; Ling, Q.; Yuan, K.; Wu, G.; Yin, W. On the linear convergence of the admm in decentralized consensus optimization. *IEEE Trans. Signal Process.* **2014**, *62*, 1750–1761. [CrossRef]
184. Ramanan, P.; Yildirim, M.; Chow, E.; Gebraeel, N. An asynchronous, decentralized solution framework for the large scale unit. *IEEE Trans. Power Syst.* **2019**, *34*, 3677–3686. [CrossRef]
185. Boyd, S.; Parikh, N.; Chu, E.; Peleato, B.; Eckstein, J. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.* **2011**, *3*, 1–122. [CrossRef]
186. Bragin, M.A.S.; Luh, P.B. Distributed and asynchronous unit commitment and economic dispatch. In Proceedings of the 2017 IEEE Power& Energy Society General Meeting, Chicago, IL, USA, 16–20 July 2017.
187. Bragin, M.A.; Yan, B.; Luh, P.B. Distributed and asynchronous coordination of a mixed-integer linear system via surrogate lagrangian relaxation. *IEEE Trans. Autom. Sci. Eng.* **2021**, *18*, 1191–1205. [CrossRef]
188. Kargarian, A.; Mehrtash, M.; Falahati, B. Decentralized implementation of unit commitment with analytical target cascading. *IEEE Trans. Power Syst.* **2018**, *33*, 3981–3993. [CrossRef]
189. Fu, Y.; Li, Z.; Wu, L. Modeling and solution of the large-scale security-constrained unit commitment. *IEEE Trans. Power Syst.* **2013**, *28*, 3524–3533. [CrossRef]
190. Kargarian, A.; Fu, Y.; Li, Z. Distributed security-constrained unit commitment for large-scale power systems. *IEEE Trans. Power Syst.* **2015**, *30*, 1925–1936. [CrossRef]
191. Ming, Z.; Junyi, Z.; Gengyin, L.; Jianwen, R. Distributed dispatch approach for bulk ac/dc hybrid systems with high wind power penetration. *IEEE Trans. Power Syst.* **2018**, *33*, 3325–3336.
192. Wei, L.; Liu, G.; Yan, S. Graph computing based security constrained unit commitment in hydro-thermal power systems incorporating pumped hydro storage. *CSEE J. Power Energy Syst.* **2021**, *7*, 485–496. [CrossRef]
193. Fu, Y.; Shahidehpour, M. Fast scuc for large-scale power systems. *IEEE Trans. Power Syst.* **2007**, *22*, 2144–2151. [CrossRef]

194. Feng, W.; Yuan, C.; Dai, R.; Liu, G.; Li, F. Graph computation based power flow for large-scale ac/dc system. In Proceedings of the 2018 International Conference on Power System Technology (POWERCON), Guangzhou, China, 6–8 November 2018; pp. 468–473. [CrossRef]
195. Yuan, C.; Zhou, Y.; Liu, G.; Dai, R.; Lu, Y.; Wang, Z. Graph computing-based wls fast decoupled state estimation. *IEEE Trans. Smart Grid* **2020**, *11*, 2440–2451. [CrossRef]
196. Shi, Q.; Yuan, C.; Feng, W.; Liu, G.; Dai, R.; Wang, Z.; Li, F. Enabling model-based lti for large-scale power system security monitoring and enhancement with graph-computing-based power flow calculation. *IEEE Access* **2019**, *7*, 167010–167018. [CrossRef]
197. Chen, Y.; Casto, A.; Wang, F.; Wang, Q.; Wang, X.; Wan, J.; Prices, D.S.; Indices, A. Improving large scale day-ahead security constrained unit commitment performance. *IEEE Trans. Power Syst.* **2016**, *31*, 4732–4743. [CrossRef]
198. Chen, Y.; Pan, F.; Holzer, J.; Rothberg, E. A high performance computing based market economics driven neighborhood search and polishing algorithm for security constrained unit commitment. *IEEE Trans. Power Syst.* **2021**, *36*, 292–302. [CrossRef]
199. PNNL. High-Performance Computing Helps Grid Operators Manage Increasing Complexity. 2020. Available online: https://bit.ly/3tLmUA6 (accessed on 13 November 2022).
200. Stott, B.; Alsac, O.; Monticelli, A.J. Security analysis and optimization. *Proc. IEEE* **1987**, *75*, 1623–1644. [CrossRef]
201. Balu, N.; Bertram, T.; Bose, A.; Brandwajn, V.; Cauley, G.; Curtice, D.; Fouad, A.; Fink, L.; Lauby, M.G.; Wollenberg, B.F.; et al. On-line power system security analysis. *Proc. IEEE* **1992**, *80*, 262–282. [CrossRef]
202. Hao, S.; Papalexopoulos, A.; Peng, T. Distributed processing for contingency screening applications. *IEEE Trans. Power Syst.* **1995**, *10*, 838–844. [CrossRef]
203. Mendes, J.C.; Saavedra, O.R.; Feitosa, S.A. A parallel complete method for real-time security analysis in power systems. *IEEE Trans. Power Syst.* **2000**, *56*, 27–34. [CrossRef]
204. Balduino, L.; Alves, A.C.B. Parallel processing in a cluster of microcomputers with application in contingency analysis. In Proceedings of the 2004 IEEE/PES Transmision and Distribution Conference and Exposition: Latin America (IEEE Cat. No. 04EX956), Sao Paulo, Brazil, 8–11 November 2004; pp. 285–290. [CrossRef]
205. Huang, Z.; Chen, Y.; Nieplocha, J. Massive contingency analysis with high performance computing. In Proceedings of the 2009 IEEE Power & Energy Society General Meeting, Calgary, AB, Canada, 26–30 July 2009; pp. 1–8.
206. Huang, Q.; Zhou, M.; Zhang, Y.; Wu, Z. Exploiting cloud computing for power system. In Proceedings of the International Conference on Power System Technology, Hangzhou, China, 24–28 October 2010; pp. 1–6.
207. Khaitan, S.K.; McCalley, J.D. Parallelizing power system contingency analysis using d programming language. In Proceedings of the 2013 IEEE Power Energy Society General Meeting, Vancouver, BC, Canada, 21–25 July 2013; pp. 1–5. [CrossRef]
208. Khaitan, S.K.; McCalley, J.D. Scale: A hybrid mpi and multithreading based work stealing approach for massive contingency analysis in power systems. *Electr. Pow. Syst. Res.* **2014**, *114*, 118–125. [CrossRef]
209. Sekine, Y.; Takahashi, K.; Sakaguchi, T. Real-time simulation of power system dynamics. *Int. J. Electr. Power Energy Syst.* **1994**, *16*, 145–156. [CrossRef]
210. Pai, M.A.; Sauer, P.W.; Kulkarni, A.Y. Conjugate gradient approach to parallel processing in dynamic simulation of power systems. In Proceedings of the 1992 American Control Conference, Chicago, IL, USA, 24–26 June 1992; pp. 1644–1647. [CrossRef]
211. Decker, I.C.; Falcao, D.M.; Kaszkurewicz, E. Conjugate gradient methods for power system dynamic simulation on parallel computers. *IEEE Trans. Power Syst.* **1996**, *11*, 1218–1227. [CrossRef]
212. Shu, J.; Xue, W.; Zheng, W. A parallel transient stability simulation for power systems. *IEEE Trans. Power Syst.* **2005**, *20*, 1709–1717. [CrossRef]
213. Jin, S.; Huang, Z.; Diao, R.; Wu, D.; Chen, Y. Parallel implementation of power system dynamic simulation. In Proceedings of the IEEE Power and Energy Society General Meeting, Vancouver, BC, Canada, 21–25 July 2013. [CrossRef]
214. Alvarado, F.L. Parallel solution of transient problems by trapezoidal integration. *IEEE Trans. Power Appl. Syst.* **1979**, *98*, 1080–1090. [CrossRef]
215. Hatcher, W.L.; Brasch, F.M.; Van Ness, J.E. A feasibility study for the solution of transient stability problems by multiprocessor structures. *IEEE Trans. Power Appl. Syst.* **1977**, *96*, 1789–1797. [CrossRef]
216. Hatcher, W.L. *A Special Purpose Multiprocessor for the Simulation of Dynamic Systems*; Northwestern University: Evanston, IL, USA, 1976.
217. Brasch, F.M.; Van Ness, J.E.; Sang-Chul, K. The use of a multiprocessor network for the transient stability problem. In Proceedings of the IEEE Conference Proceedings Power Industry Computer Applications Conference, Cleveland, OH, USA, 15–19 May 1979; pp. 337–344. [CrossRef]
218. Brasch, F.M.; Van Ness, J.E.; Kang, S.C. Simulation of a multiprocessor network for power system problems. *IEEE Trans. Power Appl. Syst.* **1982**, *101*, 295–301. [CrossRef]
219. Taoka, H.; Abe, S.; Takeda, S. Fast transient stability solution using an array processor. *IEEE Trans. Power Appl. Syst.* **1983**, *102*, 3835–3841. [CrossRef]
220. Taoka, H.; Abe, S. Fast transient stability solution adapted for an array processor. *IEEJ Trans. Power Energy* **1984**, *104*, 297–304. [CrossRef]
221. La Scala, M.; Bose, A.; Tylavsky, D.J.; Chai, J.S. A highly parallel method for transient stability analysis. *IEEE Trans. Power Syst.* **1990**, *5*, 1439–1446. [CrossRef]

222. Scala, M.L.; Brucoli, M.; Torelli, F.; Trovato, M. A gauss-jacobi-block-newton method for parallel transient stability analysis. *IEEE Trans. Power Syst.* **1990**, *5*, 1168–1177. [CrossRef]
223. Scala, M.L.; Sbrizzai, R.; Torclli, F. A pipelined-in-time parallel algoiiitiim for transient stability analysis. *Test* **1991**, *6*, 715–722.
224. Zhu, N.; Bose, A. A dynamic partitioning scheme for parallel transient stability analysis. *IEEE Trans. Power Syst.* **1992**, *7*, 940–946. [CrossRef]
225. Chat, J.S.; Bose, A. Bottlenecks in parallel algorithms for power system stability analysis. *IEEE Trans. Power Syst.* **1993**, *8*, 9–15. [CrossRef]
226. Ilic'-Spong, M.; Crow, M.L.; Pai, M.A. Transient stability simulation by waveform relaxation methods. *IEEE Trans. Power Syst.* **1987**, *2*, 943–949. [CrossRef]
227. Crow, M.L.; Ilic, M. The parallel implementation of the waveform relaxation method for the simulation of structure-preserved power systems. In Proceedings of the 1990 IEEE International Symposium on Circuits and Systems (ISCAS), New Orleans, LA, USA, 1–3 May 1990; Volume 2, pp. 1285–1288. [CrossRef]
228. Tylavsky, D.J.; Nagaraj, S.; Crouch, P.E.; Gaye, A.A.; Jarriel, L.F. Parallel-vector processing synergy and frequency domain transient stability simulations. *Electr. Pow. Syst. Res.* **1993**, *28*, 89–97. [CrossRef]
229. Granelli, G.P.; Montagna, M.; La Scala, M.; Torelli, F. Relaxation-newton methods for transient stability analysis on a vector/parallel computer. *IEEE Trans. Power Syst.* **1994**, *9*, 637–643. [CrossRef]
230. Taoka, H.; Iyoda, I.; Noguchi, H.; Sato, N.; Nakazawa, T. Real-time digital simulator for power system analysis on a hypercube computer. *IEEE Trans. Power Syst.* **1992**, *7*, 1–10. [CrossRef]
231. Chai, J.S.; Zhu, N.; Bose, A.; Tylavsky, D.J. Parallel newton type methods for power system stability analysis using local and shared memory multiprocessors. *IEEE Trans. Power Syst.* **1991**, *6*, 1539–1545. [CrossRef]
232. Lee, S.Y.; Chiang, H.D.; Lee, K.G.; Ku, B.Y. Parallel power system transient stability analysis on hypercube multiprocessors. *IEEE Trans. Power Syst.* **1991**, *6*, 1337–1343. [CrossRef]
233. Aloisio, G.; Bochicchio, M.A.; La Scala, M.; Sbrizzai, R. A distributed computing approach for real-time transient stability analysis. *IEEE Trans. Power Syst.* **1997**, *12*, 981–987. [CrossRef]
234. Hong, C.; Shen, X.M. Parallel transient stability analysis on distributed memory message passing multiprocessors. In Proceedings of the 1997 Fourth International Conference on Advances in Power System Control, Operation and Management (APSCOM-97) (Conf. Publ. No. 450), Hong Kong, China, 11–14 November 1997; pp. 304–309. [CrossRef]
235. Hong, C. Implementation of parallel algorithms for transient stability analysis on a message passing multicomputer. In Proceedings of the 2000 IEEE Power Engineering Society Winter Meeting. Conference Proceedings, Singapore, 23–27 January 2000; pp. 1410–1415.
236. Xue, W.; Shu, J.; Zheng, W. Parallel transient stability simulation for national power grid of china. In Proceedings of the ISPA'04: Second International Conference on Parallel and Distributed Processing and Applications, Hong Kong, China, 13–15 December 2004; Springer: Berlin/Heidelberg, Germany; pp. 765–776. [CrossRef]
237. Jikeng, L.; Xinyu, T.; Xudong, W.; Weicheng, W. Parallel simulation for the transient stability of power system. In Proceedings of the 2008 Third International Conference on Electric Utility Deregulation and Restructuring and Power Technologies, Nanjing, China, 6–9 April 2008; pp. 1325–1329. [CrossRef]
238. Lin Jikeng, L.; Xudong, W.; Xinyu, T. Asynchronous parallel simulation of transient stability based on equivalence. In Proceedings of the 2009 International Conference on Sustainable Power Generation and Supply, Nanjing, China, 6–7 April 2009; pp. 1–5. [CrossRef]
239. Jalili-Marandi, V. *Acceleration of Transient Stability Simulation for Large-Scale Power Systems on Parallel and Distributed Hardware*; Technical Report; University of Alberta: Edmonton, AB, Canada, 2010.
240. Werlen, K.; Glavitsch, H. Computation of transients by parallel processing. *IEEE Trans. Power Deliv.* **1993**, *8*, 1579–1585. [CrossRef]
241. Falcao, D.M.; Kaszkurewicz, E.; Almeida, H.L.S. Application of parallel processing techniques to the simulation of power system electromagnetic transients. *IEEE Trans. Power Syst.* **1993**, *8*, 90–96. [CrossRef]
242. Morales, F.; Rudnick, H.; Cipriano, A. Electromechanical transients simulation on a multicomputer via the vdhn—Maclaurin method. *IEEE Trans. Power Syst.* **2001**, *16*, 418–426. [CrossRef]
243. Dufour, C.; Jalili-Marandi, V.; Bélanger, J. Real-time simulation using transient stability, electromagnetic transient and fpga-based high-resolution solvers. In Proceedings of the 2012 SC Companion: High Performance Computing, Networking Storage and Analysis, Salt Lake City, UT, USA, 24–29 June 2012; pp. 283–288. [CrossRef]
244. Peng, Z.; Shu-jun, Y.; Hai-lin, Z.; Shuo, Z. Multi-rate electromagnetic transient simulation of large-scale power system based on multi-core. *J. Eng.* **2017**, *2017*, 1106–1112. [CrossRef]
245. Beaudin, S.; Marceau, R.J.; Bois, G.; Savaria, Y.; Kandil, N. An economic parallel processing technology for faster than real-time transient stability simulation. *Eur. Trans. Electr. Power* **2003**, *13*, 105–112. [CrossRef]
246. Duan, P.; Xu, S.; Chen, H.; Yang, X.; Wang, S.; Hu, E. High performance computing (hpc)for advanced power system studies. In Proceedings of the 2nd IEEE Conference on Energy Internet and Energy System Integration (EI2 2018), Beijing, China, 20–22 October 2018. [CrossRef]
247. Li, X.; Balasubramanian, P.; Sahraei-Ardakani, M.; Abdi-Khorsand, M.; Hedman, K.W.; Podmore, R. Real-Time Contingency Analysis with Corrective Transmission Switching solver and inexact newton method. *IEEE Trans. Power Syst.* **2017**, *32*, 2604–2617. [CrossRef]

248. Jin, S.; Huang, Z.; Diao, R.; Wu, D.; Chen, Y. Comparative implementation of high performance computing for power system dynamic simulations. *IEEE Trans. Smart Grid* **2017**, *8*, 1387–1395. [CrossRef]
249. Aristidou, P.; Fabozzi, D.; Van Cutsem, T. Dynamic simulation of large-scale power systems using a parallel schur-complement-based decomposition method. *IEEE Trans. Parallel Distrib. Syst.* **2014**, *25*, 2561–2570. [CrossRef]
250. Aristidou, P.; Van Cutsem, T. A parallel processing approach to dynamic simulations of combined transmission and distribution systems. *Int. J. Electr. Power Energy Syst.* **2015**, *72*, 58–65. [CrossRef]
251. Aristidou, P.; Lebeau, S.; Van Cutsem, T. Power system dynamic simulations using a parallel two-level schur-complement decomposition. *IEEE Trans. Power Syst.* **2016**, *31*, 3984–3995. [CrossRef]
252. Gopal, A.; Niebur, D.; Venkatasubramanian, S. DC power flow based contingency analysis using graphics processing units. In Proceedings of the 2007 IEEE Lausanne Power Tech, Lausanne, Switzerland, 1–5 July 2007; pp. 731–736.
253. Tang, K.; Dong, S.; Zhu, B.; Ni, Q.; Song, Y. Gpu-based real-time n-1 ac power flow algorithm with preconditioned iterative method. In Proceedings of the 2018 IEEE Power Energy Society General Meeting (PESGM), Portland, OR, USA, 5–10 August 2018; pp. 1–5.
254. Fu, M.; Zhou, G.; Zhao, J.; Feng, Y.; He, H.; Liang, K. Gpu-based n-1 static security analysis algorithm with preconditioned conjugate gradient method. *IEEE Access* **2020**, *8*, 124066–124075. [CrossRef]
255. Huang, S.; Dinavahi, V. Real-time contingency analysis on massively parallel architectures with compensation method. *IEEE Access* 2018, *6*, 44519–44530. [CrossRef]
256. Wu, J.Q.; Bose, A.; Huang, J.A.; Valette, A.; Lafrance, F. Parallel implementation of power system transient stability analysis. *IEEE Trans. Power Syst.* **1995**, *10*, 1226–1233. [CrossRef]
257. Hou, L.; Bose, A. Implementation of the waveform relaxation algorithm on a shared memory computer for the transient stability problem. *IEEE Trans. Power Syst.* **1997**, *12*, 1053–1060. [CrossRef]
258. Jalili-Marandi, V.; Dinavahi, V. Large-scale transient stability simulation on graphics processing units. In Proceedings of the 2009 IEEE Power & Energy Society General Meeting, Calgary, AB, Canada, 26–30 July 2009; pp. 1–6.
259. Jalili-Marandi, V.; Dinavahi, V. Simd-based large-scale transient stability simulation on the graphics processing unit. *IEEE Trans. Power Syst.* **2010**, *25*, 1589–1599. [CrossRef]
260. Jalili-Marandi, V.; Zhou, Z. Large-scale transient stability simulation of electrical power systems on parallel GPUs. *IEEE Trans. Parallel Distrib. Syst.* **2010**, *23*, 1255–1266. [CrossRef]
261. Yu, Z.; Huang, S.; Shi, L.; Chen, Y. GPU-based JFNG method for power system transient dynamic simulation. In Proceedings of the 2014 International Conference On Power System Technology, Chengdu, China, 20–22 October 2014; Volume 1, pp. 20–22.
262. Wen, B.; Guo, W.; Hu, J.; Wang, F.; Ye, J. GPU based parallel simulation of transient stability using symplectic gauss algorithm and preconditioned GMRES method. In Proceedings of the 2012 Power Engineering And Automation Conference, Wuhan, China, 18–20 September 2012.
263. Liao, X.; Wang, F. Parallel computation of transient stability using symplectic gauss method and GPU. *IET Gener. Transm. Distrib.* **2016**, *10*, 3727–3735. [CrossRef]
264. Gao, H.; Chen, Y.; Xu, Y.; Yu, Z.; Chen, L. A GPU-based parallel simulation platform for large-scale wind farm integration. In Proceedings of the 2014 IEEE PES T&D Conference and Exposition, Chicago, IL, USA, 14–17 April 2014; Volume 1, pp. 1–5.
265. Song, Y.; Chen, Y.; Huang, S.; Xu, Y.; Yu, Z.; Marti, J. Fully GPU-based electromagnetic transient simulation considering large-scale control systems for system-level studies. *IET Gener. Transm. Distrib.* **2017**, *11*, 2840–2851. [CrossRef]
266. Song, Y.; Chen, Y. Efficient GPU-based electromagnetic transient simulation for power systems with thread-oriented transformation and automatic code generation. *IEEE Access* **2018**, *6*, 25724–25736. [CrossRef]
267. Zhou, G.; Zhang, X.; Lang, Y.; Bo, R.; Jia, Y.; Lin, J.; Feng, Y. Electrical power and energy systems a novel gpu-accelerated strategy for contingency screening of static security analysis. *Int. J. Electr. Power Energy Syst.* **2016**, *83*, 33–39. [CrossRef]
268. Chen, D.; Jiang, H.; Li, Y.; Xu, D. A two-layered parallel static security assessment for large-scale grids based on gpu. *IEEE Trans. Smart Grid* **2017**, *8*, 1396–1405. [CrossRef]
269. Zhou, G.; Feng, Y.; Bo, R.; Chien, L.; Zhang, X.; Lang, Y.; Jia, Y.; Chen, Z. GPU-accelerated batch-ACPF solution for n-1 static security analysis. *IEEE Trans. Smart Grid* **2017**, *8*, 1406–1416. [CrossRef]
270. Debnath, J.K.; Fung, W.K.; Gole, A.M.; Filizadeh, S. Simulation of large-scale electrical power networks on graphics processing units. In Proceedings of the 2011 IEEE Electrical Power and Energy Conference, Winnipeg, MB, Canada, 3–5 October 2011; pp. 199–204. [CrossRef]
271. Debnath, J.K.; Gole, A.M.; Fung, W.K. Graphics-processing-unit-based acceleration of electromagnetic transients simulation. *IEEE Trans. Power Deliv.* **2016**, *31*, 2036–2044. [CrossRef]
272. Song, Y.; Chen, Y.; Yu, Z.; Huang, S.; Chen, L. A fine-grained parallel emtp algorithm compatible to graphic processing units. In Proceedings of the 2014 IEEE PES General Meeting | Conference Exposition, Denver, CO, USA, 26–30 July 2014; pp. 1–6. [CrossRef]
273. Zhou, Z.; Dinavahi, V. Parallel massive-thread electromagnetic transient simulation on gpu. *IEEE Power Energy Soc. Gen. Meet.* **2014**, *29*, 1045–1053. [CrossRef]
274. Zhou, Z.; Dinavahi, V. Fine-grained network decomposition for massively parallel electromagnetic transient simulation of large power systems. *IEEE Power Energy Technol. Syst. J.* **2017**, *4*, 51–64. [CrossRef]

275. Wallach, Y.; Handschin, E.; Bongers, C. An efficient parallel processing method for power system state estimation. *IEEE Trans. Power Appar. Syst.* **1981**, *100*, 4402–4406. [CrossRef]
276. Cavin, R.K. Multiprocessor static state estimation. *IEEE Trans. Power Appl. Syst.* **1982**, *101*, 302–308.
277. Aoki, K. A parallel computation algorithm for static state estimation by means of matrix inversion lemma. *IEEE Power Eng. Rev.* **1987**, *2*, 624–631.
278. Abur, A.; Tapadiya, P. Parallel state estimation using multiprocessors. *Electr. Power Syst. Res.* **1990**, *18*, 67–73. [CrossRef]
279. Lin, S.Y. A distributed state estimator for electric power systems. *IEEE Trans. Power Syst.* **1992**, *7*, 551–557. [CrossRef]
280. El-Keib, A.A.; Nieplocha, J.; Singh, H.; Maratukulam, D.J. A decomposed state estimation technique suitable for parallel processor implementation. *IEEE Trans. Power Syst.* **1992**, *7*, 1088–1097. [CrossRef]
281. Falcao, D.M.; Wu, F.F.; Murphy, L. Parallel and distributed state estimation. *IEEE Trans. Power Syst.* **1995**, *10*, 724–730. [CrossRef]
282. Ebrahimian, R.; Baldick, R. State estimation distributed processing [for power systems]. *IEEE Trans. Power Syst.* **2000**, *15*, 1240–1246. [CrossRef]
283. Carvalho, J.B.; Barbosa, F.M. Distributed processing in power system state estimation. In Proceedings of the 2000 10th Mediterranean Electrotechnical Conference. Information Technology and Electrotechnology for the Mediterranean Countries. Proceedings. MeleCon 2000 (Cat. No.00CH37099), Lemesos, Cyprus, 29–31 May 2000; Volume 3, pp. 1128–1131. [CrossRef]
284. Nieplocha, J.; Marquez, A.; Tipparaju, V.; Chavarria-Miranda, D.; Guttromson, R.; Huang, H. Towards efficient power system state estimators on shared memory computers. In Proceedings of the 2006 IEEE Power Engineering Society General Meeting, Montreal, QC, Canada, 18–22 June 2006; p. 5. [CrossRef]
285. Schneider, K.P.; Huang, Z.; Yang, B.; Hauer, M.; Nieplocha, Y. Dynamic state estimation utilizing high performance computing methods. In Proceedings of the 2009 IEEE/PES Power Systems Conference and Exposition, Seattle, WA, USA, 15–18 March 2009; pp. 1–6. [CrossRef]
286. Korres, G.N.; Tzavellas, A.; Galinas, E. A distributed implementation of multi-area power system state estimation on a cluster of computers. *Electr. Power Syst. Res.* **2013**, *102*, 20–32. [CrossRef]
287. Xia, Y.; Chen, Y.; Ren, Z.; Huang, S.; Wang, M.; Lin, M. State estimation for large-scale power systems based on hybrid cpu-gpu platform. In Proceedings of the 2017 IEEE Conference on Energy Internet and Energy System Integration (EI2), Beijing, China, 26–28 November 2017; pp. 1–6. [CrossRef]
288. Magaña-Lemus, E.; Medina-Ríos, A.; Ramos-Paz, A.; Montesinos-González, V.H. Periodic steady state determination of power systems using graphics processing units. In Proceedings of the 2013 10th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE), Mexico City, Mexico, 30 September–4 October 2013; pp. 274–279. [CrossRef]
289. Magaña-Lemus, E.; Medina, A.; Ramos-Paz, A. Periodic steady state solution of power systems by selective transition matrix identification, lu decomposition and graphic processing units. In Proceedings of the 2015 IEEE Power Energy Society General Meeting, Denver, CO, USA, 26–30 July 2015; pp. 1–5. [CrossRef]
290. Karimipour, H. Accelerated parallel wls state estimation for large-scale power systems on gpu. In Proceedings of the North American Power Symposium (NAPS), Manhattan, KS, USA, 22–24 September 2013; Volume 1.
291. Karimipour, H.; Dinavahi, V. On detailed synchronous generator modeling for massively parallel dynamic state estimation. In Proceedings of the 2014 North American Power Symposium (NAPS), Pullman, WA, USA, 7–9 September 2014; pp. 1–6. [CrossRef]
292. Karimipour, H.; Dinavahi, V. Extended kalman filter-based parallel dynamic state estimation. *IEEE Trans. Smart Grid* **2015**, *6*, 1539–1549. [CrossRef]
293. Karimipour, H.; Dinavahi, V. On false data injection attack against dynamic state estimation on smart power grids. In Proceedings of the 2017 IEEE International Conference on Smart Energy Grid Engineering (SEGE), Oshawa, ON, Canada, 14–17 August 2017; pp. 388–393. [CrossRef]
294. Rahman, A.; Venayagamoorthy, G.K. Dishonest gauss newton method based power system state estimation on a gpu. In Proceedings of the Clemson University Power Systems Conference (PSC), Clemson, SC, USA, 8–11 March 2016.
295. Rahman, M.A.; Venayagamoorthy, G.K. Convergence of the fast state estimation for power systems. *SAIEE Afr. Res. J.* **2017**, *108*, 117–127. [CrossRef]
296. Kim, M.K.; Park, J.K.; Nam, Y.W. Market-clearing for pricing system security based on voltage stability criteria. *Energy* **2011**, *36*, 1255–1264. [CrossRef]
297. Geng, G.; Jiang, Q. A two-level parallel decomposition approach for transient stability constrained optimal power flow. *IEEE Trans. Power Syst.* **2012**, *27*, 2063–2073. [CrossRef]
298. Jiang, Q.; Zhou, B.; Zhang, M. Parallel augment lagrangian relaxation method for transient stability constrained unit commitment. *IEEE Trans. Power Syst.* **2013**, *28*, 1140–1148. [CrossRef]
299. Gong, L.; Fu, Y.; Shahidehpour, M. A parallel solution for the resilient operation of power systems in geomagnetic storms. *IEEE Trans. Smart Grid* **2020**, *11*, 3483–3495. [CrossRef]
300. Vasquez, A.D.; Sousa, T. A parallel processing approach to stability analysis considering transmission and distribution systems. In Proceedings of the 2019 IEEE Milan PowerTech, Milano, Italy, 23–27 June 2019; pp. 1–6. [CrossRef]
301. Qi, S.; Li, G.; Bie, Z. Hybrid energy flow calculation for electric-thermal coupling system based on inexact newton method. In Proceedings of the 2019 IEEE Sustainable Power and Energy Conference (iSPEC), Beijing, China, 21–23 November 2019; pp. 2443–2449. [CrossRef]

302. Geng, G.; Jiang, Q.; Sun, Y. Parallel transient stability-constrained optimal power flow using gpu as coprocessor. *IEEE Trans. Smart Grid* **2017**, *8*, 1436–1445. [CrossRef]
303. Geng, G.; Ajjarapu, V.; Jiang, Q. A Hybrid Dynamic Optimization Approach for Stability Constrained Optimal Power Flow. *IEEE Trans. Power Syst.* **2014**, *29*, 2138–2149. [CrossRef]
304. Luo, X.; Zhang, S.; Litvinov, E. Practical design and implementation of cloud computing for power system planning studies. *IEEE Trans. Smart Grid* **2019**, *10*, 2301–2311. [CrossRef]
305. Feng, M.; Xiaochuan, L.; Qiang, Z.; Litvinov, E. Cloud computing: An innovative it paradigm to facilitate power system operations. In Proceedings of the 2015 IEEE Power Energy Society General Meeting, Denver, CO, USA, 26–30 July 2015; pp. 1–5. [CrossRef]
306. ISO England. Working Toward a Smarter, Greener Grid. 2021. Available online: https://bit.ly/3Oh1xhM (accessed on 15 November 2022).
307. Morante, Q.; Ranaldo, N.; Vaccaro, A.; Zimeo, E. Pervasive grid for large-scale power systems contingency analysis. *IEEE Trans. Ind. Inf.* **2006**, *2*, 165–175. [CrossRef]
308. Taylor, G.A.; Irving, M.R.; Hobson, P.R.; Huang, C.; Kyberd, P.; Taylor, R.J. Distributed monitoring and control of future power systems via grid computing. In Proceedings of the 2006 IEEE Power Engineering Society General Meeting, Montreal, QC, Canada, 18–22 June 2006; p. 5. [CrossRef]
309. Huang, Q.; Qin, K.; Wang, W. Development of a grid computing platform for electric power system applications. In Proceedings of the IEEE Power Engineering Society General Meeting, Montreal, QC, Canada, 18–22 June 2006; pp. 1–7.
310. Huang, Z.; Nieplocha, J. Transforming power grid operations via high performance computing. In Proceedings of the 2008 IEEE Power and Energy Society General Meeting—Conversion and Delivery of Electrical Energy in the 21st Century, Pittsburgh, PA, USA, 20–24 July 2008; pp. 1–8. [CrossRef]
311. Ali, M.; Dong, Z.Y.; Li, X.; Zhang, P. Rsa-grid: A grid computing based framework for power system reliability and security analysis. In Proceedings of the 2006 IEEE Power Engineering Society General Meeting, Montreal, QC, Canada, 18–22 June 2006; p. 7. [CrossRef]
312. Al-Khannak, R.; Bitzer, B. Load balancing for distributed and integrated power systems using grid computing. In Proceedings of the 2007 International Conference on Clean Electrical Power, Capri, Italy, 21–23 May 2007; pp. 123–127. [CrossRef]
313. Wang, X.; Yan, Z.; Li, L. A grid computing based approach for the power system dynamic security assessment. *Comput. Electr. Eng.* **2010**, *36*, 553–564. [CrossRef]
314. Axceleon. Enfuzion—High Performance Parallel Computing Software. 2021. Available online: http://www.axceleon.com/prod-cloudfuzion/ (accessed on 13 November 2022).
315. Sarker, M.R.; Wang, J. Security and cloud outsourcing framework for economic dispatch. *IEEE Trans. Smart Grid* **2018**, *9*, 5810–5819. [CrossRef]
316. Mangasarian, O.L. Privacy-preserving linear programming. *Optim. Lett.* **2010**, *5*, 165–172. [CrossRef]
317. Li, W.; Li, H.; Deng, C. Privacy-preserving horizontally partitioned linear programs with inequality constraints. *Optim. Lett.* **2011**, *7*, 137–144. [CrossRef]
318. Overlin, M.; Smith, C. High performance computing techniques with power systems simulations. In Proceedings of the 2018 IEEE High Performance Extreme Computing Conference, HPEC 2018, Waltham, MA, USA, 25–27 September 2018. [CrossRef]
319. Yoon, D.H.; Kang, S.K.; Kim, M.; Han, Y. Exploiting coarse-grained parallelism using cloud computing in massive power flow computation. *Energies* **2018**, *11*, 2268. [CrossRef]
320. Rusitschka, S.; Eger, K.; Gerdes, C. Smart grid data cloud: A model for utilizing cloud computing in the smart grid domain. In Proceedings of the First IEEE International Conference on Smart Grid Communications, Gaithersburg, MD, USA, 4–6 October 2010; pp. 483–488.
321. Mohsenian-Rad, A.H.; Leon-Garcia, A. Coordination of cloud computing and smart power grids. In Proceedings of the 2010 First IEEE International Conference on Smart Grid Communications, Gaithersburg, MD, USA, 4–6 October 2010; pp. 368–372. [CrossRef]
322. Markovic, D.S.; Zivkovic, D.; Branovic, I.; Popovic, R.; Cvetkovic, D. Smart power grid and cloud computing. *Renew. Sustain. Energy Rev.* **2013**, *24*, 566–577. [CrossRef]
323. Hongseok, K.; Kim, Y.J.; Yang, K.; Thottan, M. Cloud-based demand response for smart grid: Architecture and distributed algorithms. In Proceedings of the 2011 IEEE International Conference on Smart Grid Communications (SmartGridComm), Brussels, Belgium, 17–20 October 2011; pp. 398–403. [CrossRef]
324. Bo, Z.Q.; Wang, L.; Zhou, F.; Luo, K.; Han, M.; Yin, W.; Liu, J.Y. Substation cloud computing for secondary auxiliary equipment. In Proceedings of the 2014 International Conference on Power System Technology, Chengdu, China, 20–22 October 2014; pp. 1853–1858. [CrossRef]
325. Bhandari, K.S.; Cho, G.H. An Energy Efficient Routing Approach for Cloud-Assisted Green Industrial IoT Networks. *Sustainability* **2020**, *12*, 7358. [CrossRef]
326. Sheikhi, A.; Rayati, M.; Bahrami, S.; Ranjbar, A.M.; Sattari, S. A cloud computing framework on demand side management game in smart energy hubs. *Int. J. Electr. Power Energy Syst.* **2015**, *64*, 1007–1016. [CrossRef]
327. Rajeev, T.; Ashok, S. Dynamic load-shifting program based on a cloud computing framework to support the integration of renewable energy sources. *Appl. Energy* **2015**, *146*, 141–149. [CrossRef]

328. Amazon. Amazon Web Services. 2020. Available online: https://aws.amazon.com/ (accessed on 13 November 2022).
329. Cao, Z.; Lin, J.; Wan, C.; Song, Y.; Zhang, Y.; Wang, X. Optimal cloud computing resource allocation for demand side management in smart grid. *IEEE Trans. Smart Grid* **2017**, *8*, 1943–1955. [CrossRef]
330. Wang, S.; Wang, X.; Wu, W. Cloud computing and local chip-based dynamic economic dispatch for microgrids. *IEEE Trans. Smart Grid* **2020**, *11*, 3774–3784. [CrossRef]
331. Happ, H.H.; Pottle, C.; Wirgau, K.A. Future computer technology for large power system simulation. *Automatica* **1979**, *15*, 621–629. [CrossRef]
332. Jose, S. Effects of special purpose hardware in scientific computation ww with emphasis on power system applications. *IEEE Trans. Power Appl. Syst.* **1982**, *101*, 265–270.
333. Roberge, V.; Tarbouchi, M.; Okou, F. Parallel power flow on graphics processing units for concurrent evaluation of many networks. *IEEE Trans. Smart Grid* **2017**, *8*, 1639–1648. [CrossRef]
334. Pan, F.; Northwest, P. *Hippo: A Computation Tool for Planning Tomorrow'S Electricity*; Pacific Northwest National Laboratory: Richland, WA, USA, 2017.
335. Palmer, B.; Perkins, W.; Chen, Y.; Jin, S.; Callahan, D.; Glass, K.; Diao, R.; Rice, M.; Elbert, S.; Vallem, M.; et al. Gridpacktm: A framework for developing power grid simulations on high-performance computing platforms. *Int. J. High Perform. Comput. Appl.* **2016**, *30*, 223–240. [CrossRef]
336. Nividia. Nividia on Demand. 2020. Available online: https://developer.nvidia.com/cuda-zone (accessed on 13 November 2022).
337. Grötschel, M. Special issue: Optimization stories. Selected papers based on the presentations at the 21st international symposium on mathematical programming, ISMP, Berlin, Germany, August 19–24. 2012. In *Documenta Mathematica, Extra Vol., Optimization Stories*; FIZ Karlsruhe GmbH: Eggenstein-Leopoldshafen, Germany, 2012; 460p.
338. Glockner, G. Does Gurobi Support Gpus? 2021. Available online: https://bit.ly/3bbqjBA (accessed on 13 November 2022).
339. Jin, S.; Chen, Y.; Diao, R.; Huang, Z.H.; Perkins, W.; Palmer, B. Power grid simulation applications developed using the gridpack™ high performance computing framework. *Electr. Pow. Syst. Res.* **2016**, *141*, 22–30. [CrossRef]
340. Koltsaklis, N.E.; Dagoumas, A.S. State-of-the-art generation expansion planning: A review. *Appl. Energy* **2018**, *230*, 563–589. [CrossRef]
341. Ng, C.P.; Jabbour, K.; Meyer, W. Loadflow analysis on parallel computers. In Proceedings of the 32nd Midwest Symposium on Circuits and Systems, Champaign, IL, USA, 14–16 August 1989; Volume 1, pp. 10–15. [CrossRef]
342. Services, A.W. *Amazon Elastic Compute Cloud: User Guide for Linux Instances*; Amazon: Bellevue, WA, USA, 2022.
343. Ajagekar, A.; You, F. Quantum computing for energy systems optimization: Challenges and opportunities. *Energy* **2019**, *179*, 76–89. [CrossRef]
344. Tylavsky, D.J.; Heydt, G.T. Quantum computing in power system simulation. In Proceedings of the 2003 IEEE Power Engineering Society General Meeting (IEEE Cat. No.03CH37491), Toronto, ON, Canada, 13–17 July 2003; Volume 2, pp. 950–956. [CrossRef]
345. Eskandarpour, R.; Ghosh, K.; Khodaei, A.; Zhang, L.; Paaso, A.; Bahramirad, S. Quantum computing solution of dc power flow. *arXiv* 2020, arXiv:2010.02442.
346. Baker, M. 1500 scientists lift the lid on reproducibility. *Nature* **2016**, *533*, 452–454. [CrossRef]
347. Serra-Garcia, M.; Gneezy, U. Nonreplicable publications are cited more than replicable ones. *Sci. Adv.* **2021**, *7*, eabd1705. [CrossRef]
348. Cockburn, A.; Dragicevic, P.; Besançon, L.; Gutwin, C. Threats of a replication crisis in empirical computer science. *Commun. ACM* **2020**, *63*, 70–79. [CrossRef]