energies

MDPI

Article

High-Performance Algorithms for Soft X-Ray Diagnostics Towards Future Fusion Reactors and Power Generation

Rafał Krawczyk [1,*], Tomasz Czarski [2] and Maryna Chernyshova [3]

[1] Faculty of Electronics and Information Technology, Warsaw University of Technology, Nowowiejska 15/19, 00-665 Warsaw, Poland
[2] Institute of Plasma Physics and Laser Microfusion, Hery 23, 01-497 Warszaw, Poland
[3] National Centre For Nuclear Research, Andrzeja Sołtana 7/3, 05-400 Otwock, Poland
* Correspondence: rafal.krawczyk@pw.edu.pl

Abstract

Nuclear fusion represents a transformative solution for global energy systems, offering a carbon-free, inherently safe, and virtually inexhaustible power source. As the field transitions from experimental reactors like ITER to demonstration power plants (DEMO) capable of delivering net electricity to the grid (300–500 MW), the computational demands for plasma control have escalated. Modern fusion diagnostics, particularly soft X-ray (SXR) systems, generate massive data volumes that require high-throughput processing to ensure plasma stability and optimize energy gain. Recent breakthroughs in record-breaking plasma durations have further exposed the critical latency bottlenecks in traditional analytical workflows. This work addresses these challenges by introducing advanced computational strategies optimized towards next-generation reactors. Firstly, we present new data-processing algorithms in C++ and CUDA, achieving significant reductions in computation time. This allowed for more efficient analysis of collected experimental data for plasma confinement studies. Secondly, we discuss hardware architectures that will allow, in the future, up-scaling and parallel runtime processing of data with a feedback signal to the reactor control systems. We present a detailed analysis of the computational workflows underlying soft X-ray diagnostics, followed by a presentation of the proposed optimized algorithms. Their impact on prospective hardware system designs is then evaluated in terms of scalability, latency, and throughput. Performance evaluations demonstrated substantial speedups of both the sequential CPU-based and the parallel GPU-based algorithms, highlighting the potential of these methods for future real-time plasma control for energetically stable and efficient fusion power generation. The sequential and parallel algorithms were 18.8 and 89.1 times faster, respectively, versus the baseline implementation. The processing rate was increased from 31.8 MiB/s to 4.32 GiB/s. The results show the effectiveness of massively parallel computation for plasma diagnostics and pave the way towards further research to produce a cluster-based distributed system. The demand for such high-performance, real-time data processing methodologies extends beyond the plasma confinement domain and is expected to grow across energy systems as they become increasingly complex and data-driven.

Keywords: energy projects; electrotechnologies; energy efficiency optimization; fusion reactors; plasma diagnostics; measurements; high-performance computing; architecture; X-ray detectors; real-time feedback; algorithms; challenges; hardware

Academic Editor: Kan Liu

Received: 5 May 2026
Revised: 22 June 2026
Accepted: 24 June 2026
Published: 29 June 2026

1. Introduction

Annual demand growth for electricity over the next five years is set to be 50% higher on average than in the previous decade, with an average increase of 3.5% per year [1]. This is driven by rising industrial requirements, the electrification of transport, increased cooling needs, and the rapid expansion of data centers, largely fueled by AI infrastructure. This trend is observable across emerging, developing, and advanced economies. The rapidly growing demand is also coupled with an urgent necessity to replace traditional fossil sources with efficient and safe alternatives. Such a transition is essential to mitigate environmental impact, to reduce the effect of global warming, and to stop relying on shrinking conventional sources of energy.

However, the magnetic confinement of plasma in thermal fusion reactors remains one of the most significant and technically demanding challenges in contemporary research and engineering [2–7]. Achieving a fusion energy surplus requires satisfying the Lawson criterion [8] where plasma temperature, density, and confinement time are the critical factors. In order to achieve this in the most promising magnetic fusion reactors so far (tokamaks), plasma must be magnetically confined and maintained at temperatures exceeding 150 million degrees Celcius. Most crucially, plasma must be perfectly insulated from the reactor's first wall. Any interaction between plasma and the tokamak vessel results in the influx of metallic impurities, which "poison" the plasma, leading to an immediate and drastic efficiency drop in the reaction.

The long-term objective of plasma research is the construction of an ITER (International Thermonuclear Experimental Reactor), a 500 MW device designed to achieve a ten-fold energy gain [6]. The subsequent step is the development and construction of DEMO (the DEMOnstration power plant), which aims to bridge the gap between experimental research and commercial energy production. DEMO is envisioned as the first fusion facility capable of delivering net electricity to the grid, with an expected power output in the range of 300 to 500 MW [7]. Unlike ITER, which is primarily an experimental device, DEMO will focus on demonstrating steady-state operation, high plant availability, and the efficient conversion of fusion energy into electricity. The successful realization of DEMO will mark a decisive transition from proof-of-concept experiments to practical fusion power generation, paving the way for the integration of commercial fusion reactors into the global power grids in the subsequent decades. From this long-term perspective, real-time control, sustained operation, and system reliability are of key importance.

Consequently, intensive research is being conducted through global cooperation to understand plasma behavior and develop robust control mechanisms in existing devices. Within this framework, diagnostics and control systems play a critical role in harnessing high-temperature tokamak plasma. Maintaining stable operation requires over 50 distinct diagnostic systems, varying in complexity and operational importance [4]. Studies conducted at WEST (Tungsten Environment in Steady-state Tokamak) in the Cadarache facility, as well as in other magnetic fusion devices worldwide, aim to understand complex physical phenomena and mitigate factors that inhibit sustained reaction.

The recent breakthroughs in plasma confinement durations [9], and the related research [10–12] at WEST, have experimentally demonstrated an urgent need for the development of significantly more computationally efficient and high-throughput analytical methods.

In particular, the scope of this work focuses on optimizing the computational workloads of the soft X-ray (SXR) diagnostics system, which is being commissioned at WEST. The SXR diagnostics are designed to provide, among other insights, a deeper understanding of the interplay between MHD activity and metallic impurities, originating from the plasma-facing components of the device. Currently, the system plays an important role in

advancing the understanding of heavy impurity transport and plasma–wall interaction physics within tokamaks.

Measurements at the WEST tokamak have successfully demonstrated the ability to acquire spatially, temporally, and spectrally resolved SXR data. This capability provides a robust solution for monitoring tungsten and fast electron generation in high-temperature, high-radiation environments [10,11]. Looking forward, it is expected to facilitate stable plasma operation by producing feedback for the plasma control systems [10,11]. The objective is to enable real-time detection of metallic impurities that would otherwise degrade or quench the reaction.

The SXR diagnostics exhibit a unique combination of features. Firstly, they have a distinguishably large amount of unique computations at large throughput. This is to an extent that the runtime computing requires more than a single server. Secondly, its final purpose is providing critical control throughout the experiment. In this context, a long-term evolution of the system has been planned. In its current stage, the purpose of the system is to collect data for offline analysis to better understand the plasma interactions. In this form, the system follows the same principles of its *soft real-time* counterparts in high-energy physics data acquisition systems at CERN [13–21]. These hybrid FPGA and server clusters allow for continuous collection and selection of physics events, with a *soft real-time* requirement that the processing should not delay to an extent that the buffers are filled and the physics data is lost. Successful long-term operation has been documented with Commercial-Off-The-Shelf (COTS) servers with CPUs and GPUs, with offline analysis that allowed for post-experimental deep understanding of physics [13–18]. This approach has also a reported prospective use for the future runs at CERN [19–21]. SRX diagnostics pertains to this pattern, with its unique use case and custom set of algorithms.

The subsequent step of the system evolution is its incorporation into *hard real-time* control to provide feedback to the tokamak, as these systems are to be incorporated into the Plasma Control System in ITER [22] or an equivalent for tokamaks such as WEST [23], HL-2M [24], DIII-D [25], ASDEX-U [26], and MAST-U [27]. In this context, there has been documented successful use of COTS HPC servers and infrastructure for *hard real-time* control with real-time Linux distributions. Additionally, GPU-based solutions have been proposed in different types of diagnostics for the HBT-EP [28] and the EAST tokamaks [29].

In either of these approaches, it is critical to understand how distributed and massively parallel computations can be implemented to efficiently process the large workloads of plasma physics data.

The recent record plasma durations achieved in the WEST tokamak have highlighted an urgent need for more efficient processing of diagnostic data. In the current iteration of the SXR diagnostic system [10,11], *events* of interest are selected in real-time by an FPGA-based (Field Programmable Gate Array) front-end, transmitted to a central server via PCIe links, and subsequently saved to local storage.

To date, the analysis of the collected data has been performed post-experimentally using MATLAB R2024b. However, recent long-duration plasma pulses in WEST have generated massive volumes of event data. The resulting computational workloads have exceeded the capabilities of existing analytical workloads, proving that such processing can no longer be efficiently handled without dedicated, high-performance software.

Additionally, the next generation of the system currently under development is intended to utilize multiple servers, each handling significantly larger data volumes. The anticipated roadmap involves upscaling the computing infrastructure to handle workloads within a High-Performance Computing (HPC) cluster. This transition necessitates a substantial reduction in data volumes at the server level during runtime to prevent network and storage bottlenecks.

Therefore, the primary objective of this work was to address the two urgent requirements. Firstly, we developed algorithms to significantly accelerate computationally intensive post-experimental data analysis. Secondly, we proposed future server architectures designed for massively parallel runtime processing and data reduction in the new revision of the system.

In this regard, we present novel data processing algorithms implemented in C++ and in CUDA that achieve substantial reductions in computation time. These results, alongside their impact on prospective HPC system designs, are evaluated in terms of scalability, latency, and throughput. The demonstrated speedups of both the sequential CPU-based implementation and the parallel GPU-based approach highlight the potential of these methods for future real-time plasma control.

The remainder of the paper is structured as follows. Section 2 outlines the core computing concepts of SXR diagnostics. Section 3 discusses potential hardware architectures for high-performance runtime data analysis. Section 4 details the optimized sequential C++ algorithm for CPUs and the parallel CUDA algorithm for GPUs. Section 5 presents the experimental results and evaluates their impact on the proposed architectures, while Section 6 provides concluding remarks.

2. The Processing Pipeline in Soft X-Ray Tokamak Diagnostics

The current version of the SXR diagnostic system involves multiple steps, including analog signal shaping, digital signal processing with data selection in FPGA front-end cards, and data collection in a server. After the experiment, the results are collected and analyzed offline. The processing stages of the system are illustrated in Figure 1. The main hardware modules are shown in gray rectangles, whereas the associated processing stages are labeled on the right. Data transmissions across modules are marked with arrows.

There is a clear demarcation between the two types of processing. The early stages are highly customized and conducted on dedicated hardware. The final step is the transmission over the PCIe links to the server. The later stages rely on the Commercial-Off-The-Shelf (COTS) server. This heterogeneous architecture allows for an up-scaling of infrastructure using the methods known in HPC cluster computing. This is motivated by the planned increase in detector resolution and data rates, as well as the possibility of incorporating additional computations at the server level, such as tomographic reconstruction of the radiation distribution [30,31].

The first step of the SXR detection consists of a Gaseous Electron Multiplier (GEM) coupled with an analog front-end. The GEM detector operates by converting incident photons into electron clouds, which are subsequently detected as collections of *events* corresponding to signals recorded on the detector's *pixels* [32,33]. A single electron cloud typically produces multiple events occurring in close time vicinity, usually in the form of a Gaussian distribution of electric charge across multiple adjacent channels of the detector.

The second step involves signal processing and selection in the FPGA cards [34]. Each of the cards processes a fragment of all the detector's channels. The FPGAs are point-to-point connected to a server through PCIe cards. The server collects multiple streams of events, where a single event is represented by 11 Bytes containing the channel number, the charge value, and the system timestamp. The data from FPGAs are sorted by timestamp in each of the streams, and each of the streams is saved to persistent storage for the duration of the experiment.

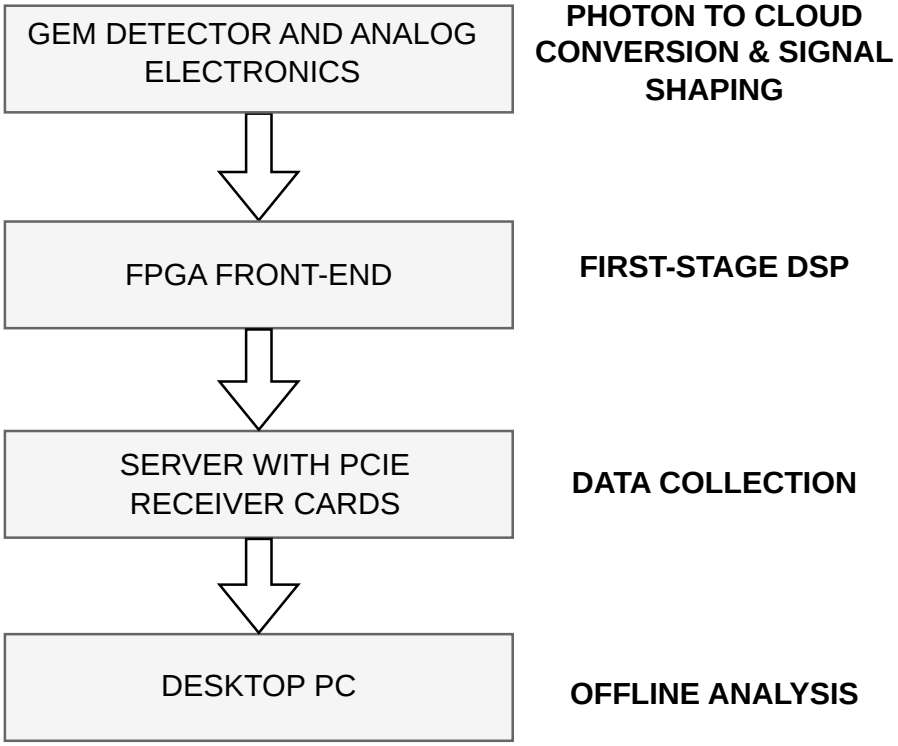

Figure 1. The main processing stages of the SXR diagnostic system. Rectangles indicate main hardware modules, right labels indicate associated processing stages, and arrows indicate data exchange.

After the experiment, the data are processed offline. The post-experimental analysis involves multiple steps:

- As an initial step, all data streams are merged into a single sequence of events sorted in time.
- Then, events in time proximity are grouped together into sets known as *packs*. The *pack finding* process is conducted with predefined criteria. Two consecutive events are qualified into the same pack if the timestamp difference of their occurrence is less than or equal to 3 on-FPGA clock cycles.
- The *cluster finding* algorithm is conducted. Its purpose is to identify groups of events in the pack that are in spatial proximity, corresponding to the electron clouds. This step involves reordering the events in the packs, cluster identification, and the extraction of their parameters. Any irregular clusters (that is, exhibiting excessive size, too large time span, or atypical charge distribution) are discarded during this process.
- The final step involves producing an output three-dimensional histogram of the spatial, charge, and time distribution of electron clouds reconstructed from the clusters.

The output histogram allows for further analysis of the SXR distribution in time in the detector. This result is intended for future runtime control of plasma stability, shape, and impurity content. One of the assumed future analyses involves tomographic reconstruction [30,31]. Additionally, although currently established experimentally, criteria for cluster selection can also evolve throughout the campaigns in tokamaks. This likelihood of change strongly reinforces the necessity of using modular software-defined solutions.

It should be noted that the size of the output histogram is notably reduced in comparison with the input feed. The server receives multiple streams of data at an order of Gigabytes per second per PCIe interface of an FPGA card [34]. The output histogram is expected to be at most tens of MiB in size. This observation is a strong premise to move workloads from offline computing to runtime analysis on servers, and to use the relatively small volumes of output for runtime tokamak control.

3. Hardware Architecture for High-Performance Runtime Data Analysis

The most urgent objective of the presented developed software has been the speedup of the analysis of post-experimental data. We additionally evaluated whether the processing can be sufficiently accelerated so that it can be moved to real-time analysis. This mode of operation allows for runtime data reduction, thus minimizing the amount of output sent to fast tokamak control.

The present baseline form of the server processing pipeline of the SXR diagnostic system is shown in Figure 2. The server is marked in grey, and the transmission across the devices over the PCIe interfaces is marked with white arrows.

We propose two new approaches that can potentially benefit from massively parallel GPU processing. The first proposed architecture is presented in Figure 3. The main modification involves adding a GPU through a PCIe link that will process data from the CPU, then sending it back to the CPU. The CPU can then store the data or send the processed output. The advantage of this approach is profiting from parallel GPU processing, but it introduces additional latencies for transmitting data to and out of the GPU over the PCIe interface.

The second proposed upgrade is presented in Figure 4. It relies on the Remote Direct Memory Access protocol (RDMA) [35], where the CPU is only responsible for dataflow control. In this approach, data are transferred directly from FPGA to GPU using the GPUDirect protocol [36]. GPU also has to support RDMA [37]. Additionally, after the data are processed in the GPU, one can further profit from RDMA by sending the system output from the GPU directly, provided that the data will be fed through an RDMA-supporting network card. In this approach, the GPU completely bypasses data transfers through the CPU. This reduces the number of data transmissions, latencies, and memory footprint of the system (by mitigating redundant copying of data).

If additional up-scaling is needed, more servers will be used in each of the three approaches. That said, an architecture exploiting RDMA can be further profitable, because with such HPC frameworks as CUDA-aware MPI [38], a direct GPU-to-GPU data exchange can be made over CUDA-supporting RDMA NICs [39]. The main disadvantage of this approach is the additional effort in configuring RDMA-compliant devices and then configuring RDMA in the data flow.

The performance of either of the proposed architectures strongly depends on the speedup and scalability of the GPU processing algorithms. The development of optimized algorithms was, therefore, our next step.

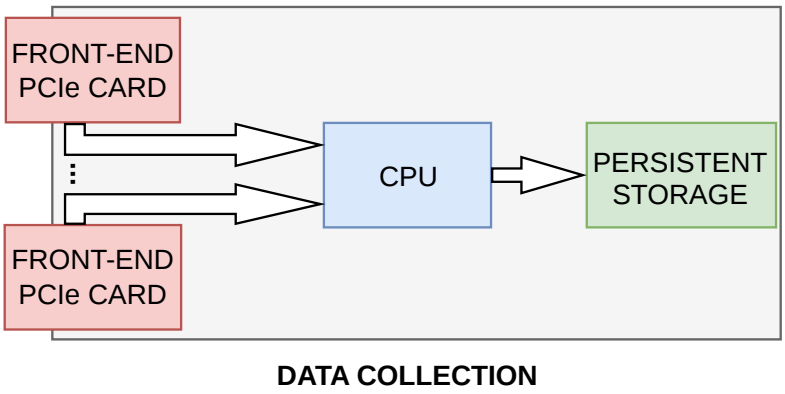

Figure 2. Data flow in the present hardware architecture for the server in the SXR diagnostic system. White arrows indicate data exchange over the PCIe. Data streams are sent to a CPU and written to persistent storage after processing.

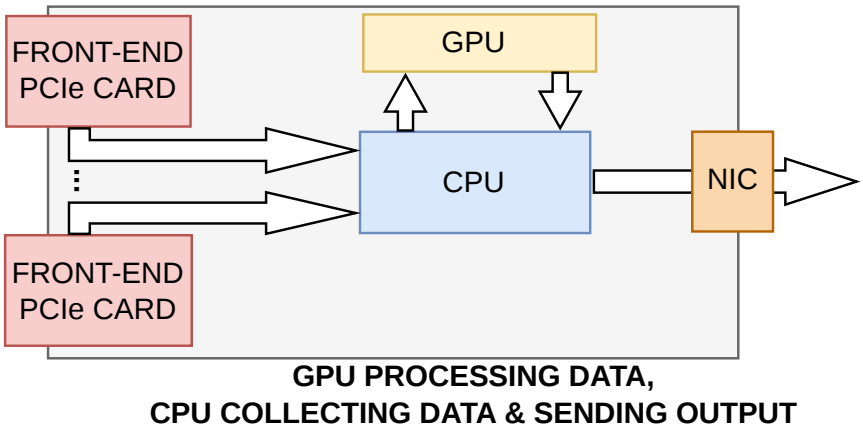

Figure 3. The overview of the proposed data flow in a server with a GPU. Computations are offloaded to a massively parallel device.

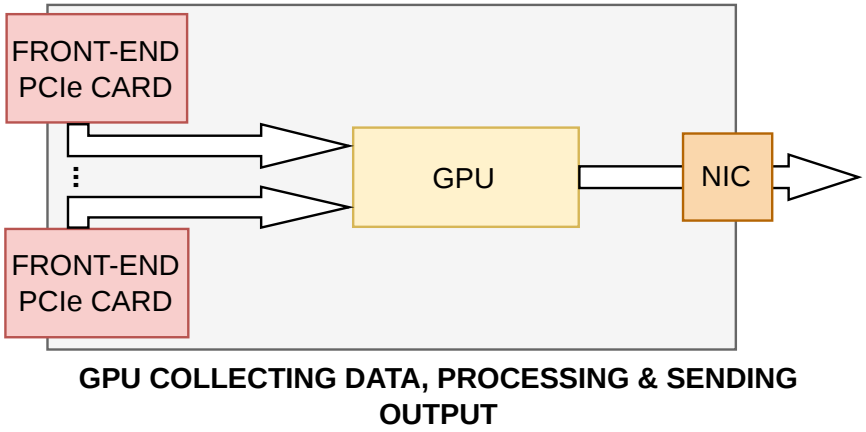

Figure 4. The overview of the proposed data flow with a GPU and benefiting from RDMA. Data exchange in a CPU is bypassed through the use of zero-copy RDMA and GPUDirect protocols.

4. Optimized Algorithms for SXR Diagnostics

The initial post-experimental data analysis was written in MATLAB scripts [40]. Although the application was optimized for matrix operations, it proved insufficient in terms of performance for large data volumes. Therefore, we have designed standalone applications that accelerate these computations. The initial MATLAB data access pattern is presented in Figure 5. The data structures are marked as gray rectangles, whereas the computations are marked as red arrows.

The initial processing had several disadvantages originating from MATLAB limitations:

- It relied on a time-consuming, MATLAB built-in sort algorithm.
- The data localization (and therefore profiting from cache) was limited, as a large number of intermediate data structures were produced in each step based on the full input data set. This imposed a large memory footprint.
- On top of that, MATLAB runtime added latencies and consumed additional resources.

We have therefore addressed these issues by designing a standalone sequential algorithm in C++. This has been an intermediate step towards computing on a GPU with CUDA. C++ and CUDA have been deliberately selected for performance reasons.

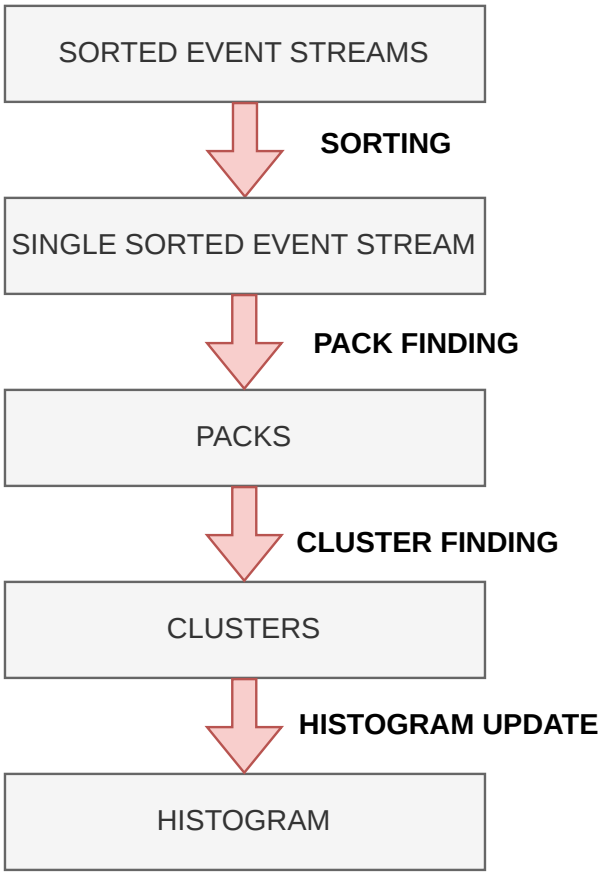

Figure 5. The data flow and the processing stages of the MATLAB post-experimental SXR data analysis. Gray rectangles indicate intermediate data structures; red arrows indicate computing steps.

4.1. Standalone Sequential Algorithm for a CPU

The sequential C++ algorithm introduced the following changes to address the poor performance:

- The sort algorithm has been replaced with a merging algorithm, thus reducing the complexity from O(Nlog(N)) to O(N).
- The pack finding, cluster finding, and histogram update have been fused to process small chunks of data. When a pack is found, it is processed as long as all its clusters are found and all histogram updates are made. This has significantly increased the data reuse and the number of localized data accesses.
- The number of intermediate data structures has been minimized, thereby reducing the memory footprint.
- On top of that, the standalone version is not negatively impacted by the overhead of MATLAB runtime.

The outline of the algorithm is shown in Figure 6. The differences from a reference presented in Figure 5 are notable. The sequential algorithm for a CPU has been used as a baseline for the subsequent algorithm exploiting massively parallel GPU processing.

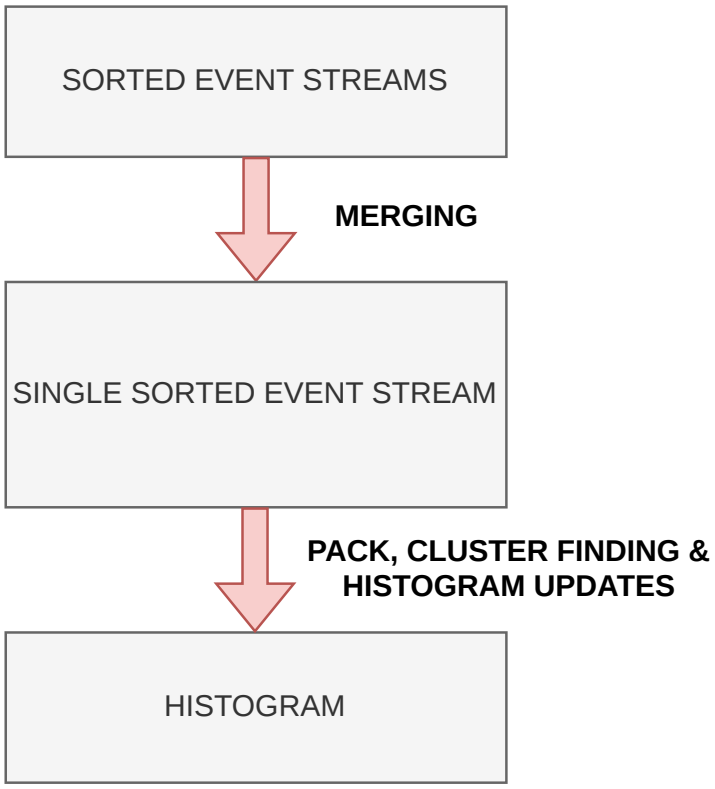

Figure 6. The data flow and the processing stages of the standalone sequential algorithm for a CPU. Pack finding, cluster finding and histogram update are fused to process localized operations for small chunks of data.

4.2. Standalone Parallel Algorithm for a GPU

Having developed a baseline sequential C++ implementation, we have further redesigned the algorithm into a parallel version for a GPU. The primary objective was to transform the algorithm so that it could profit from massive parallelism of a GPU. The outline of the algorithm is presented in Figure 7. The algorithm consists of three stages, each of which is run in a separate CUDA kernel. The main premises for its design were as follows:

- Merging of streams can be efficiently parallelized on a GPU.
- Each pack of events can be processed independently, and in parallel.
- The only additional workload versus the sequential algorithm is a lightweight definition of ranges of data for each of the threads. This is to prevent the threads from repeating the computations on the same packs.
- The sole shared structure across the threads is the updated output histogram.

The first modification was introducing the well-balanced and GPU-efficient parallel merge path algorithm [41].

The subsequent step was implementing the decomposition of data into the separate ranges. In comparison with the sequential algorithm, this is an additional step. This is a critical part, where data are divided into collections of packs. The range finding allows for further independent parallel processing of packs in GPU threads. The decomposition itself is also parallelizable. As the further steps involve reorganizing the data in the pack, the range finding had to be done separately first. However, this is a relatively lightweight operation which involves the following steps:

- Initially, the single merged stream is divided into ranges of equal size.
- Each thread checks in parallel if its *start* of a range is not in the middle of a pack. This can be checked by comparing the timestamp difference across the consecutive events.

If in the affirmative, the thread moves the start of its range forward, thereby reducing the total size of its range.

- Subsequently, each thread checks if its *end* of a range is not in the middle of a pack. If in the affirmative, the thread moves the end of the range forward, thereby extending the total size of its range.
- In the end, all of the threads analyze data sets that are aligned to full packs.

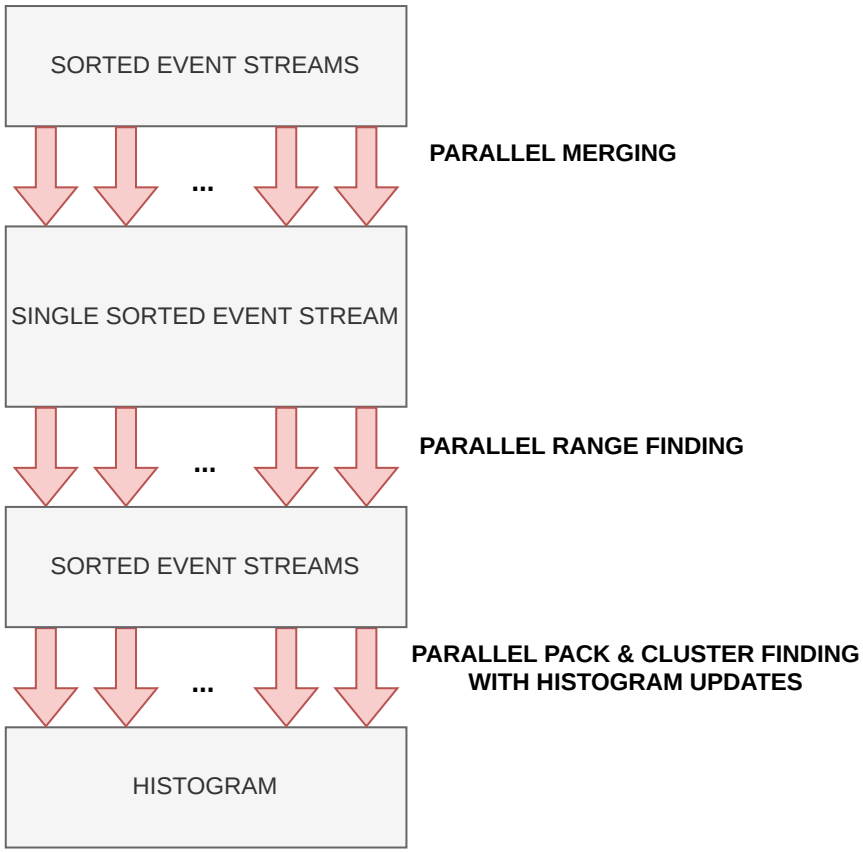

Figure 7. The data flow and the processing stages of the standalone parallel algorithm for a GPU. CUDA kernels are marked with red arrows. Additional range finding step is distinguishable.

The principle of operation of the data decomposition in the range finding kernel is also illustrated in Figure 8. Atomic packs of data are marked as white rectangles. The purpose of the kernel is to avoid sharing packs across two ranges. Yellow and violet translucent rectangles represent two ranges. Either of them is gradually aligned by a separate CUDA thread. In the first step, each of the threads have ranges of equal size. Ranges can have misaligned end and start indices and the pack can be incorrectly split. Each thread independently moves its own start and end range index, thereby creating independent ranges covering the full merged data set.

This method of workload decomposition is well balanced, as it is experimentally proven that the packs have at most dozens of events. The unevenness is negligible in practice. Given that typically at least dozens of millions of events are grouped into ranges for tens of thousands of threads, the offsets of dozens of events do not notably affect load balancing. The thread's range typically has at least tens of thousands of events.

As a following step, each of the ranges is processed independently and in parallel, with operations involving pack and cluster finding. These fused operations have been implemented in the same localized fashion as for the sequential C++ algorithm. This step is embarrassingly parallel and can be done fully independently for each CUDA thread.

The final step involves an update of a histogram with the produced data. The histogram structure is shared across threads, and therefore its updates must be atomic. However, with a large enough histogram, the likelihood of forced sequential data access is low. Expected output histogram dimensions of $200 \times 5 \times 1500$ mean there are typically tens of millions of bins for tens of thousands of parallel running threads. Additionally, parallel computations made before the histogram update are a more computationally intensive dominant factor. In conclusion, based on the presented principle of operation, the algorithm is very well scalable.

However, it should be expected that the transmission of input data (whose volume is several orders of magnitude higher than the output) will be notable in the overall processing time. Special attention should therefore be paid to how much of the processing time is spent on the transmission to and from the CPU and the actual computing. This has been investigated in the subsequent experimental part.

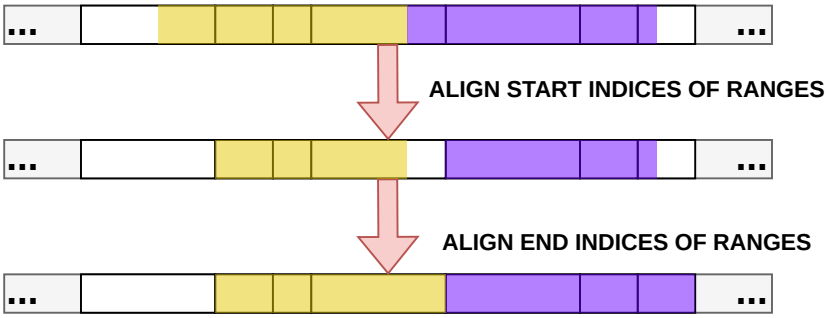

Figure 8. The three decomposition steps producing independent ranges. White rectangles represent atomic packs. Yellow and violet translucent rectangles represent two constructed ranges that are gradually aligned by separate CUDA threads.

5. Experimental Results

Our sequential CPU and parallel GPU algorithms have been implemented in the form of two standalone applications. Their functionality can be summarized as follows:

- They can read the experimental data file.
- The applications run the algorithms presented in Figures 6 and 7 respectively.
- Execution times of interest are measured.
- After computations, the results can be compared with the output from MATLAB for the corresponding experimental data.

Our first step was to perform a sanity check of our application by running unit tests designed to compare the application output with the result produced in MATLAB. The tests proved the full compliance of the results with the initial MATLAB implementation. MATLAB R2024b was used for all of the tests.

Our subsequent tests focused on the performance and scalability evaluation. We have used a reference 570.6 MiB test data set (with 54,393,812 events) for the benchmarking. We then measured the performance of MATLAB scripts (which were run on a CPU), the standalone C++ application on a CPU, and the standalone CUDA application on a GPU. File operations were not accounted for in the performance measurements.

As an output, a 90.2 MiB histogram with 23,647,000 bins was produced. Its dimensions were as follows: 85 for channel, 200 for charge, 1391 for time.

The performance test on the GPU executed CUDA kernels for a fixed number of 128 threads per block. This was the experimentally tuned best. The number of CUDA blocks was the quotient of the total number of events and the number of threads per block. There were 424,952 CUDA blocks in total.

For a CPU, we used Intel Core i5-11400H with 32 GB of DDR4. For a GPU, we used the NVIDIA GeForce RTX 3050 GPU with eight lanes of PCIe 4.0 and 4096 MB of GDDR6. The GPU has 16 Streaming Multiprocessors, each with 128 CUDA cores. Applications were launched in Ubuntu 24.04 LTS with power mode set to maximize the performance of computations. The MATLAB scripts were modified to make time measurements for parts of processing presented in Figure 5. The performance results are presented in Table 1, showing the average execution time and the speedups versus the MATLAB scripts. The sequential C++ application was 18.8 times faster than the MATLAB processing. Additionally, the GPU algorithm was 4.8 times faster than the sequential C++ code and 89.1 times faster than the initial MATLAB scripts. The test proved that the used optimization techniques were valid.

Table 1. Comparison of average execution times of post-experimental SXR analysis for a reference data set of the initial MATLAB script, the standalone sequential in C++, and the parallel CUDA application.

	MATLAB Script	**C++ Sequential Code**	**GPU Parallel Code**
Execution time	17.918 s	0.955 s	0.201 s
Speedup vs. MATLAB	1	18.8x	89.1x

In the presented test, we further measured how the data transmission and the GPU computations contribute to the overall execution time. The breakdown of execution times is presented in Table 2. The table presents how much time the most important parts of the processing consumed, and what their overall contribution was to the total processing time.

Table 2. Average timing and percentage breakdown of execution steps for the standalone parallel CUDA post-experimental SXR analysis.

Type of Operation	**Duration (ms)**	**Percentage (%)**
Input transmission time to the GPU	51.0	25.4
GPU merging kernel	42.2	21.0
Range finding kernel	7.3	3.6
Joint pack, cluster, and histogram kernel	79.9	39.7
Output transmission time from the GPU	9.2	4.6
Data allocation and deallocation	10.8	5.4
Total processing time	**201.1**	**100.00**

Several conclusions can be drawn from these results. In the application, computations accounted for 64.3% of the entire duration of the processing. The data transmission time had a notable impact on the overall processing time, and it contributed to 30% of the total time, which was almost half as much as the duration of CUDA kernels. A non-negligible 5.4% of processing has been attributed to resource allocation and deallocation.

We assume that the deployed application will rely on the *thread pool* pattern. Unlike in the test, buffers will be allocated once at startup, deallocated on termination, and reused throughout the program execution for the following portions of data during the experiment runtime.

In total, 35.4% of processing time can be potentially reduced by reusing the buffers and RDMA to avoid redundant data transmissions. In other words, the tested implementation can potentially profit from architecture proposed in Figure 6. This is under the assumption that the transmissions with GPUDirect will have comparable throughput and latencies to

the CPU-GPU transmissions made in the tests over PCIe. Evaluating this will be the subject of future studies.

That said, we managed to process 570.6 MiB of data in 129.4 ms, thus processing the data at an average throughput of 4.32 GiB/s (as opposed to 31.8 MiB/s for MATLAB). That said, we achieved a 138-fold increase in throughput in computations. A further increase is expected when moving to more efficient GPUs and to a multiple-node cluster.

We have further investigated the scalability of the CUDA algorithm on a single GPU. The reason was to understand how much the speedup of computations presented in Tables 1 and 2 has been attributed to parallelism, and if the solution can potentially be further scalable, provided that more GPUs are used. For this purpose, a test has been conducted with the GPU algorithm launched for different numbers of threads per block.

This approach allowed us to adjust the number of parallel threads per block. The purpose was to evaluate whether the problem is memory-bound or inherently sequential. Streaming Multiprocessors in the used GPU architecture execute blocks in warps of 32 threads. Warp executes the kernel code in a Single-Instruction Multiple-Threads (SIMT) paradigm. If fewer than 32 threads per block are used, then the remainder of the warp is idling [42]. We selectively measured the time for computations (that is, for CUDA kernels doing merging, range finding, and joint pack and cluster finding with histogram generation). The results are presented in Figure 9.

The used GPU had 16 Streaming Multiprocessors, so for 1 thread per block, 16 threads were running in parallel. As can be seen in Figure 9, the performance improved until the level of 128 threads per block (and thus 128 × 16 = 2048 threads running in parallel on the GPU), where the computation time reached the minimum of 129.4 ms. The best result for 2048 threads running in parallel corresponded to a total of 2048 CUDA cores on the GPU. Above that value, no further speedup was observed. This indicated that the problem had become memory-bound, as the oversubscription of CUDA cores (in which CUDA threads were waiting until memory operations were completed for further computations) did not reduce the computing time. This result has proven that the computations profited from parallelism, and that it was scalable on the used GPU until it reached the number of CUDA cores.

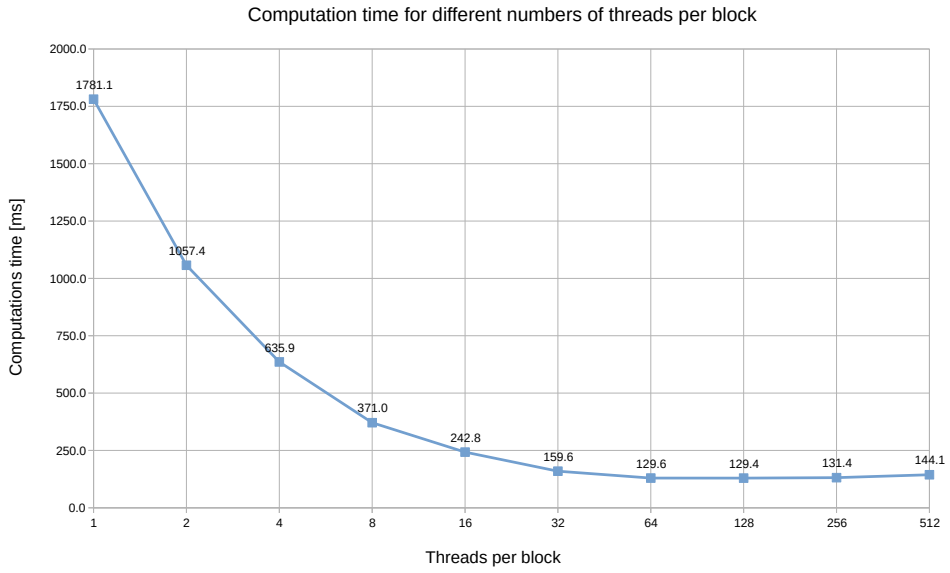

Figure 9. The average computation time, including and excluding the transmission time, for the GPU algorithm for different numbers of parallel threads per block in the CUDA kernel.

Our last test was conducted on the GPU with 128 threads per block for different input sizes. This was to assess the impact of data volume on the processing throughput.

Throughput was defined as a quotient of input data and the processing time. This is to access the maximal possible data rate if the input streams were continuously fed to the system and processed in real-time during the run of the experiment. To this end, the original input data set was progressively reduced to 10%, 20%, and so on, up to 90% and 100% of the total input file size. The results are presented in Figure 10. A peak throughput of 6.23 GiB/s was reached for 171 MiB of data, and 4.32 GiB/s for the full data set. The differences in processing time also depend on the statistical properties of data (like the number of invalid clusters), and the size of the chunk of data for processing will be tuned and optimized in the final system.

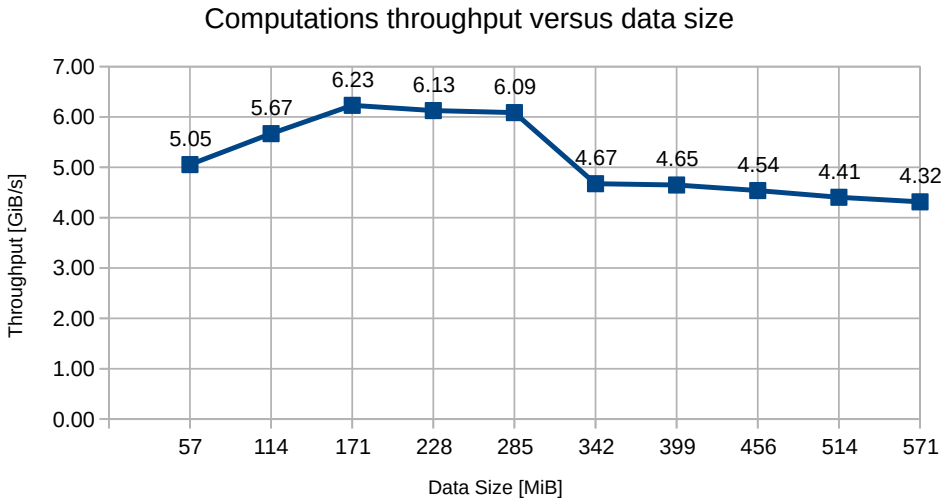

Figure 10. Average computation throughput versus input data size.

6. Conclusions

The presented study demonstrated significant acceleration of post-experimental soft X-ray (SXR) diagnostic data processing through the development of optimized algorithms. Based on the reference MATLAB scripts, we have successfully accelerated computations using our new sequential C++ and parallel CUDA algorithms for the soft X-ray diagnostic system. For the reference experimental data, the introduction of a standalone C++ algorithm resulted in an 18.8 speedup versus the MATLAB implementation. The implementation in CUDA further resulted in a speedup of 89.1 times versus the MATLAB implementation and a speedup of 4.8 times versus the sequential C++ application. When analyzing the breakdown of processing into computations and data transmission, we have demonstrated that the throughput of the computations for the reference data set has been increased by a factor of 138, from 31.8 MiB/s to 4.32 GiB/s.

As demonstrated in proposed server architectures supporting RDMA, this throughput increase can be reached by the proposed configurations supporting RDMA zero-copy protocols.

We have proven that with the proposed architectural solutions presented in Section 3, the performance of offline analysis notably benefits from using the presented standalone C++ and CUDA algorithms. In the tests, the latencies from CPU and GPU transmissions were notable and can be mitigated through the use of RDMA. This means that when moving from offline to online processing, the processing pipeline can potentially benefit from RDMA.

Subsequently, implementing the system with the presented architectures will be next steps of our research. Performance will be further evaluated for continuous data streams using RDMA. Additionally, a distributed algorithm for data exchange in a multi-node cluster is under development. The presented algorithms and results have been a successful

first step towards a proof of concept for exploiting massive parallelism in future real-time tokamak control with the soft X-ray diagnostics.

Beyond their immediate relevance to magnetic confinement fusion, the solutions proposed in this work are likely to find widespread applicability across multiple domains. A general trend of introducing HPC clusters for runtime data selection and analysis can be observed in high-energy physics [13–18], radio astronomy [43], and radar systems [44]. As supported by recent research in the field of energies [45–48], the demand for such high-performance systems is likely to grow as energy systems become more complex. Therefore, the methodologies developed herein, including algorithmic optimization, heterogeneous computing, and runtime data reduction, possess significant cross-domain potential for addressing modern challenges in energetics.

Author Contributions: HPC research and literature analysis: R.K.; HPC software development, implementation and evaluation: R.K.; detector design: M.C.; design of the analysis algorithms for the detector: T.C.; original draft preparation and figures: R.K.; grant supervision: M.C.; project administration: M.C.; funding acquisition: M.C. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been carried out within research project No. 2020/39/B/ST2/02681 financed by the National Science Center.

Data Availability Statement: The data presented in this study are available on request from the corresponding author. (The publication has also been made in cooperation with CEA, which is partially a military institution. For this reason there are strict restrictions concerning code and data sharing).

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

Abbreviations

The following abbreviations are used in this paper:

COTS	Commercial off-the-shelf
DEMO	DEMOnstration power plant
FPGA	Field Programmable gate array
GiB	gibibyte (2^{30} bytes)
HPC	High-Performance Computing
ITER	International Thermonuclear Experimental Reactor
MHD	Magnetohydrodynamic
MiB	mebibyte (2^{20} bytes)
RDMA	Remote Direct Memory Access
SIMT	Single-Instruction Multiple-Threads
SXR	soft X-ray
Tokamak	Toroidalnaya Kamera s Magnitnymi Katushkami (magnetic fusion reactor)
WEST	Tungsten (W) Environment in Steady-state Tokamak

References

1. IEA. Electricity 2026 Report. Analysis and Forecast to 2030. Available online: https://iea.blob.core.windows.net/assets/b73798cb-e452-42b9-9d8a-07542de7a041/Electricity_2026.pdf (accessed on 26 May 2026).
2. U.S. Department of Energy, Fusion Science & Technology Roadmap. Available online: https://www.energy.gov/sites/default/files/2025-10/fusion-s%26t-roadmap-101625.pdf (accessed on 26 May 2026).
3. European Research Roadmap to the Realisation of Fusion Energy. Available online: https://euro-fusion.org/wp-content/uploads/2022/10/2018_Research_roadmap_long_version_01.pdf (accessed on 26 May 2026).
4. Mazon, D.; Fenzi, C.; Sabot, R. As hot as it gets. *Nat. Phys.* **2016**, *12*, 14–17. [CrossRef]
5. Meade, D. 50 years of fusion research. *Nucl. Fusion* **2009**, *50*, 014004. [CrossRef]
6. What Will ITER Do? Available online: https://www.iter.org/fusion-energy/what-will-iter-do (accessed on 26 May 2026).

7. Joffrin, E.; Wischmeier, M.; Baruzzo, M.; Hakola, A.; Kappatou, A.; Keeling, D.; Labit, B.; Tsitrone, E.; Vianello, N.; the ASDEX Upgrade Team; et al. Overview of the EUROfusion Tokamak Exploitation programme in support of ITER and DEMO. *Nucl. Fusion* **2024**, *64*, 112019. [CrossRef]
8. Wurzel, S.E.; Hsu, S.C. Progress toward fusion energy breakeven and gain as measured against the Lawson criterion. *Phys. Plasmas* **2022**, *29*, 062103. Available online: https://pubs.aip.org/aip/pop/article/29/6/062103/2847827/Progress-toward-fusion-energy-breakeven-and-gain (accessed on 26 May 2026). [CrossRef]
9. Nuclear Fusion: WEST Beats the World Record for Plasma Duration. Available online: https://www.cea.fr/english/Pages/News/nuclear-fusion-west-beats-the-world-record-for-plasma-duration.aspx (accessed on 26 May 2026).
10. Chernyshova, M.; Mazon, D.; Malinowski, K.; Czarski, T.; Ivanova-Stanik, I.; Jabłoński, S.; Wojeński, A.; Kowalska-Strzęciwilk, E.; Poźniak, K.T.; Malard, P.; et al. First exploitation results of recently developed SXR GEM-based diagnostics at the WEST project. *Nucl. Mater. Energy* **2020**, *25*, 100850. [CrossRef]
11. Mazon, D.; Chernyshova, M.; Jardin, A.; Peysson, Y.; Król, K.; Malard, P.; Czarski, T.; Wojeński, A.; Malinowski, K.; Colette, D.; et al. First GEM measurements at WEST and perspectives for fast electrons and heavy impurities transport studies in tokamaks. *J. Instrum.* **2022**, *17*, C01073. [CrossRef]
12. Bucalossi, J.; Ekedahl, A.; the WEST Team; Achard, J.; Afonin, K.; Agullo, O.; Alarcon, T.; Allegretti, L.; Almuhisen, F.; Ancher, H.; et al. WEST full tungsten operation with an ITER grade divertor. *Nucl. Fusion* **2024**, *64*, 112022. [CrossRef]
13. Pisani, F.; Colombo, T.; Durante, P.; Frank, M.; Gaspar, C.; Cardoso, L.G.; Neufeld, N.; Perro, A. Design and Commissioning of the First 32-Tbit/s Event-Builder. *IEEE Trans. Nucl. Sci.* **2023**, *70*, 906–913. [CrossRef]
14. Krawczyk, R.D.; Colombo, T.; Neufeld, N.; Pisani, F.; Valat, S. Ethernet for High-Throughput Computing at CERN. *IEEE Trans. Parallel Distrib. Syst.* **2022**, *33*, 3640–3650. [CrossRef]
15. Hayrapetyan, A.; Tumasyan, A.; Adam, W.; Andrejkovic, J.; Arnold, B.; Bergauer, H.; Bergauer, T.; Chatterjee, S.; Damanakis, K.; Dragicevic, M.; et al. Development of the CMS detector for the CERN LHC Run 3. *J. Instrum.* **2024**, *19*, P05064. [CrossRef]
16. LHCb Trigger and Online Upgrade Technical Design Report. Available online: https://cds.cern.ch/record/1701361/files/LHCB-TDR-016.pdf (accessed on 26 May 2026).
17. LHCb Upgrade GPU High Level Trigger Technical Design Report. Available online: https://cds.cern.ch/record/2717938/files/LHCB-TDR-021.pdf (accessed on 26 May 2026).
18. Available online: https://cds.cern.ch/record/2011297/files/ALICE-TDR-019.pdf (accessed on 26 May 2026).
19. Amoiridis, V.; Behrens, U.; Bocci, A.; Branson, J.; Brummer, P.; Cano, E.; Cittolin, S.; Quintanilha, J.D.S.A.D.; Darlea, G.-L.; Deldicque, C.; et al. The CMS Orbit Builder for the HL-LHC at CERN. *EPJ Web Conf.* **2024**, *295*, 02011. [CrossRef]
20. LHCb Data Acquisition Enhancement TDR. Available online: https://cds.cern.ch/record/2886764/files/LHCB-TDR-025.pdf (accessed on 26 May 2026).
21. The Phase-2 Upgrade of the CMS Data Acquisition and High Level Trigger. Available online: https://cds.cern.ch/record/2759072/files/CMS-TDR-022.pdf (accessed on 26 May 2026).
22. Snipes, J.; de Vries, P.; Gribov, Y.; Henderson, M.; Hunt, R.; Loarte, A.; Nunes, I.; Pitts, R.; Sinha, J.; Zabeo, L.; et al. ITER plasma control system final design and preparation for first plasma. *Nucl. Fusion* **2021**, *61*, 106036. [CrossRef]
23. Nouailletas, R.; Moreau, P.; Santraine, B.; Colnel, J.; Reux, C.; Moudden, Y.; Darche, S.; Amiot, N. WEST plasma control system status. *Fusion Eng. Des.* **2023**, *192*, 113582. [CrossRef]
24. Huang, Z.; Li, B.; Zheng, G.; Yuan, Q.; Ji, X.; Xiao, B.; Zhou, J.; Huang, J.; Zhang, R.; Lu, T.; et al. A new scheme of plasma control system based on real-time Linux cluster for HL-2M. *Fusion Eng. Des.* **2023**, *192*, 113763. [CrossRef]
25. Margo, M.; Penaflor, B.; Shen, H.; Ferron, J.; Piglowski, D.; Nguyen, P.; Rauch, J.; Clement, M.; Battey, A.; Rea, C. Current State of DIII-D Plasma Control System. *Fusion Eng. Des.* **2020**, *150*, 111368. [CrossRef]
26. Treutterer, W.; Cole, R.; Lüddecke, K.; Neu, G.; Rapson, C.; Raupp, G.; Zasche, D.; Zehetbauer, T. ASDEX upgrade discharge control system—A real-time plasma control framework. *Fusion Eng. Des.* **2014**, *89*, 146–154. [CrossRef]
27. McArdle, G.; Kochan, M.; Vincent, C.; Swaidani, L.; Anand, H.; Eldon, D.; Wehner, W.; Derks, G.; Elmore, S.; Kool, B.; et al. Integrated Real-Time Control on the MAST Upgrade Tokamak. Available online: https://scientific-publications.ukaea.uk/wp-content/uploads/UKAEA-CCFE-CP2363.PDF (accessed on 26 May 2026).
28. Rath, N.; Bialek, J.; Byrne, P.; DeBono, B.; Levesque, J.; Li, B.; Mauel, M.; Maurer, D.; Navratil, G.; Shiraki, D. High-speed, multi-input, multi-output control using GPU processing in the HBT-EP tokamak. *Fusion Eng. Des.* **2012**, *87*, 1895–1899. [CrossRef]
29. Huang, Y.; Xiao, B.; Luo, Z.; Yuan, Q.; Pei, X.; Yue, X. Implementation of GPU parallel equilibrium reconstruction for plasma control in EAST. *Fusion Eng. Des.* **2016**, *112*, 1019–1024. [CrossRef]
30. Jabłoński, S.; Chernyshova, M.; Malinowski, K.; McCarthy, K. X-ray tomography system concept for stellarator TJ-II. *Phys. Plasmas* **2024**, *31*, 073301. [CrossRef]
31. Jardin, A.; Mazon, D.; Bielecki, J.; Dworak, D.; Guibert, D.; Król, K.; Savoye-Peysson, Y.; Scholz, M.; Walkowiak, J.; the WEST Team. Validating and speeding up x-ray tomographic inversions in tokamak plasmas. *Plasma Phys. Control. Fusion* **2024**, *66*, 085010. [CrossRef]

32. Chernyshova, M.; Malinowski, K.; Jabłoński, S.; Casiraghi, I.; Demchenko, I.N.; Melikhov, Y. Development of 2D GEM-based SXR plasma imaging for DTT device: Focus on readout structure. *Fusion Eng. Des.* **2021**, *169*, 112443. [CrossRef]
33. Chernyshova, M.; Malinowski, K.; Jabłoński, S.; Melikhov, Y.; Wojeński, A.; Kasprowicz, G.; Fornal, T.; Imríšek, M.; Jaulmes, F.; Weinzettl, V. 2D GEM-based SXR imaging diagnostics for plasma radiation: Preliminary design and simulations. *Nucl. Mater. Energy* **2022**, *33*, 101306. [CrossRef]
34. Wojeński, A.; Kasprowicz, G.; Chernyshova, M. GEM3k: Architecture and Design of a Novel 3rd Generation High Channel Density soft X-ray Diagnostic System Towards Commercial Fusion Power Plants. *Energies* **2026**, *19*, 918. [CrossRef]
35. RDMA Aware Networks Programming User Manual. Available online: https://docs.nvidia.com/networking/display/rdmaawareprogrammingv17 (accessed on 26 May 2026).
36. NVIDIA GPUDirect RDMA Release 13.2 Documentation. Available online: https://docs.nvidia.com/cuda/pdf/GPUDirect_RDMA.pdf (accessed on 26 May 2026).
37. NVIDIA AI Enterprise Release Notes. Available online: https://docs.nvidia.com/ai-enterprise/4.0/pdf/nvidia-ai-enterprise-release-notes.pdf (accessed on 26 May 2026).
38. Open MPI v5.0.x Documentation, Chapter 11.2.6. CUDA. Available online: https://docs.open-mpi.org/en/v5.0.x/tuning-apps/networking/cuda.html (accessed on 26 May 2026).
39. NVIDIA ConnectX NICs. Available online: https://www.nvidia.com/en-sg/networking/ethernet-adapters/ (accessed on 26 May 2026).
40. Czarski, T.; Chernyshova, M.; Malinowski, K.; Pozniak, K.T.; Kasprowicz, G.; Kolasinski, P.; Krawczyk, R.; Wojenski, A.; Linczuk, P.; Zabolotny, W.; et al. Measuring issues in the GEM detector system for fusion plasma imaging. *J. Instrum.* **2018**, *13*, C08001. [CrossRef]
41. Green, O.; McColl, R.; Bader, D.A. GPU Merge Path—A GPU Merging Algorithm. In *Proceedings of the International Conference on Supercomputing*; Association for Computing Machinery: New York, NY, USA, 2012. [CrossRef]
42. CUDA Programming Guide, Chapter 1.2 Programming Model. Available online: https://docs.nvidia.com/cuda/cuda-programming-guide/01-introduction/programming-model.html (accessed on 26 May 2026).
43. Wang, R.; Tobar, R.; Dolensky, M.; An, T.; Wicenec, A.; Wu, C.; Dulwich, F.; Podhorszki, N.; Anantharaj, V.; Suchyta, E.; et al. Processing Full-Scale Square Kilometre Array Data on the Summit Supercomputer. In *SC '20: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*; IEEE: New York, NY, USA, 2020. [CrossRef]
44. Imperatore, P.; Russo, M.; Nisar, M.; Lapegna, M.; Romano, D.; Pauciullo, A. Parallel GLRT-Based SAR Tomographic Processing on Distributed Multi-GPU Platforms. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2026**, *19*, 9379–9392. [CrossRef]
45. Shaik, M.B.; Vishwanathan, R. Verifiable Computation in Smart Grids Using Dynamic Slicing. In *2025 IEEE International Conference on Pervasive Computing and Communications Workshops and Other Affiliated Events (PerCom Workshops)*; IEEE: New York, NY, USA, 2025. [CrossRef]
46. Liu, G.; Liu, K.; Shi, D.; Zhu, W.; Wang, Z.; Chen, X. Graph Computation and Its Applications in Smart Grid. In *2017 IEEE International Congress on Big Data (BigData Congress)*; IEEE: New York, NY, USA, 2017. [CrossRef]
47. Xie, L.; Zheng, X.; Sun, Y.; Huang, T.; Bruton, T. Massively Digitized Power Grid: Opportunities and Challenges of Use-inspired AI. *Proc. IEEE* **2022**, *111*, 762–787. [CrossRef]
48. Liu, X.; David, I. Introduction to Digital Twins for the Smart Grid. In *Digital Twin Technology and Smart Grid*; Elsevier: Amsterdam, The Netherlands, 2026.