proceedings

MDPI

Proceedings

State of the Art of Information Technology Computing Models for Autonomic Cloud Computing †

Eugene Eberbach

Department of Computer Science and Robotics Engineering Program, Worcester Polytechnic Institute, 100 Institute Road, Worcester, MA 01609-2280, USA; eeberbach@gmail.com; Tel.: +1-508-831-4937

† Presented at the IS4SI 2017 Summit DIGITALISATION FOR A SUSTAINABLE SOCIETY, Gothenburg, Sweden, 12–16 June 2017.

Published: 9 June 2017

Abstract: In the paper we present several models of computation for autonomic cloud computing. In particular, we justify that for autonomic cloud computing if we require perfect self-reflection, we need models of computation going beyond Turing machines. To approximate self-reflection, models below Turing machines are sufficient. The above claims are illustrated using as an example the DIME Network Architecture.

Keywords: autonomic cloud computing; Turing machine; incompleteness; self-reflection; superTuring machines; DIME network architecture

1. Introduction

A **distributed system** is a collection of independent computers that appears to its users as a single coherent system with basic characteristics:

1. *concurrency of components*, *lack of a global clock, scalability* and *independent failures of components*,
2. *underlying transparency* (inherited by cloud computing), hiding unnecessary details of management, networking, multi-computer implementation or services,
3. *providing a uniform way of looking at the whole distributed system*.

We will consider two subclasses of distributed systems represented by cloud and autonomic computing.

Cloud Computing is a relatively new concept in computing that emerged in the late 1990s and the 2000s (but having roots from 1960s—IBM VMs for System360/370/390), first under the name *software as a service*—the details of computing are hidden transparently in the networking "cloud"

1. *Cloud computing* really is accessing resources and services needed to perform functions with **dynamically changing needs**.
2. The cloud is a **virtualization of resources that maintains and manages itself**.
3. Cloud computing is mostly used to sell hosted remote services as ***SaaS*** (*Software as a Service*—specific software, e.g., MS Office 365, Google Apps, games), ***PaaS*** (*Platform as a Service*—specific OS, DB, web server, development tools), and ***IaaS*** (*Infrastructure as a Service*—generic VMs, storage, servers, network).
4. Although the data are owned by one organization (the client) and are part of one unified distributed database, the underlying computers are owned and operated by another organization (the service vendor).
5. Computers are remote from the client's locations and are accessed over the Internet.

The concept of the cloud is really itself an extension of virtualization using virtual machines, containers (lightweight virtual machines) and meta-containers. Prior to being able to rent a virtual

server a client either had to accept noisy neighbors in a truly shared host, or pay a large amount of money to either rent a full machine or pay someone else to watch yours for you.

Instead of buying or renting physical servers, customers can buy or rent virtual ones. In addition to general computing resources, resources for specific use like databases and webservers are also available. This space is dominated by Amazon's AWS, though Google's Cloud and Microsoft's Azure are also well known providers. A large number of data-storage systems on the cloud have been built in recent years, in response to data storage needs of extremely large-scale web applications, including Big Data, Deep Machine Learning and Cloud Robotics.

Autonomic computing refers to self-managing characteristics of distributed computing resources, including self-reconfiguration, self-healing, self-optimization, self-protection, and adapting to unpredictable changes while hiding intrinsic complexity to operators and users.

There is a fundamental reason why current Turing, von Neumann stored program computing models cannot address large scale distributed computing with fluctuations both in resources and in computation workloads without increasing complexity and cost. It is a theorem of Kurt Gödel [7] that the description of an object is one class higher than the object. An important implication of Gödel's incompleteness theorem is that it is not possible to have a finite description with the description itself as the proper part. In other words, it is not possible to read yourself or process yourself as a process. In short, Gödel's theorem prohibits "self-reflection" in TMs. This self-reflection causes exactly that we have so many TM unsolvable problems. In particular, a famous UTM halting problem becomes unsolvable when UTM tries to put as its input itself.

If we want to have a perfect self-managing, self-reflection, or self-healing in autonomic cloud computing, we have to use models more powerful than TM, known as hypercomputation, superTuring machines, or superrecursive algorithms [1,4].

On the other hand, if we want to investigate only approximate self-reconfiguration or self-repair in autonomic cloud computing, we can use TMs or even their less powerful subsets like finite automata or pushdown automata.

2. Computing Models for Autonomic Cloud Computing

In [6] several models of computation and their limitations as potential models for autonomic cloud computing were presented. Generally, they were split into two classes: models not more expressive than Turing machines, and more expressive superTuring models.

We paid a particular attention to TM [8,10], because it is well known and repeated in many textbooks on computability theory (see e.g., [8]) that real computers and Turing machines are the same entities, i.e., every real computer can be encoded as an instance of Turing machine and every Turing machine can be simulated on scalable real computers (scalable to allow to approximate a TM infinite tape). It is also claimed that parallel and distributed networking computing can be reduced to a single Turing machine. If so, we would need only one single-tape deterministic Turing machine to express all autonomic cloud computing.

If the above was true what about non-recursively enumerable problems, non-computable (and not expressible) by Turing machines, and thus by implication by distributed networking computers? What about problems recursive enumerable but not recursive? Are interconnected computers equivalent to a large supercomputer, or interconnected nodes/cells provide a new quality, emerging property—not present in a single cell?

There are the following problems with using TM/UTM to model autonomic clouds: TM requires knowing in advance all possible inputs, whereas current computers interact during computations (this may require an infinite or uncountable (for continuous, real-value environments) tape memory). Distributed systems do not have a global clock, whereas TM is a totally synchronous model. TM is static and incomplete—you cannot express (even using simulating capabilities of UTM), self-reconfiguration, self-repair from autonomic computing. To model reactive systems like OS and client/server computing, extensions of TM allowing infinite time or memory are needed. But for that we need more powerful models of computation going beyond TM, i.e., superTuring models of computation. We need such models to express the self-reference of autonomic clouds.

Many scientists are convinced that computations going beyond Turing machines are possible, and current and future computers compute already and will compute beyond Turing limit (e.g., reactive programs like operating systems or client/server Internet computing). Many *superTuring models* have been proposed (also called *hypercomputation* or *superrecursive models*, and some of them, surprisingly, are equally as old as Turing machine), including Alan Turing's o-machines (TM with Oracle), c-machines and u-machines, John von Neumann's cellular automata, discrete and analog neural networks, Interaction Machines, Persistent Turing Machines, Site and Internet Machines, the π-calculus, the \$-calculus, Inductive Turing Machines, Infinite Time Turing Machines, Accelerating Turing Machines, Evolutionary Turing Machines and Evolutionary Automata [1,2,4,5].

SuperTuring models derive their higher than the TM expressiveness using three principles: *interaction*, *evolution*, or *infinity*. In the *interaction principle* the model becomes open and the agent interacts with either a more expressive component or with an infinite many components. In the *evolution principle*, the model can evolve to a more expressive one using non-recursive variation operators or by pure chance. In the *infinity principle*, models can use unbounded resources: time, memory, the number of computational elements, an unbounded initial configuration, an infinite alphabet, etc. The details can be found in [4].

3. An Example of Autonomic Cloud: DIME Network Architecture and Its Modeling

A DIME Network Architecture (DNA), pioneered by Rao Mikkilineni and his group from C3DNA, is a cloud architecture with meta-containers providing services for autonomic clouds and grids and consisting of Distributed Intelligent Managed Element (DIME) nodes connected through signaling/controlling and input/output communication channels [3,6,9]:

1. *DIME node,* called also *Cognitive Meta-Container,* consists of the *Managed Intelligent Computing Element* (*MICE*) performing computations controlled by meta-level controlling self-management of *Faults, reConfiguration, Accounting, Performance* and *Security* (*FCAPS*). The *MICE* can be in the form of simple (atomic) worker or hierarchically defined *DNA* subnetwork.
2. *Communication channels* interconnecting the *DIME* nodes are of two types: the *signaling channels* for meta-level FCAPS management and the *input-output channels* for input/output of managed MICE computing elements/workers. Signaling channels connect meta-layers for reconfiguration, performance monitoring, fault-tolerance, security and accounting, whereas i/o channels connect MICE workers to perform their message-passing for the regular work.

The Gödel's incompleteness theorem prohibits "self-reflection" in Turing machines. UTM was able to decide halting problem until it was given as an input itself. This would be equivalent to the requirement that FCAPS meta-layer can manage itself, i.e., can recover from errors in fault-manager (in the style of self-correcting codes able to correct all bits in words), or configuration manager that configures itself, or security manager able to detect malignant software in itself.

There are two classes of models for autonomic cloud computing (including DNA) possible: *allowing self-reflection* or only *approximating it*.

So far, two models for the DIME network architecture have been proposed with perfect self-reflection:

The first DNA implementation has been deployed over Linux Ubuntu and Redhat and it approximates self-reflection only. The same applies to DNA models of computation using finite number of nodes of the class of Turing Machine and below.

4. Conclusions

In the paper, we investigated several models of computation for autonomic cloud computing with or without self-reflection. Obviously, if we take hypercomputational models (e.g., TM with Oracle, or Inductive TM) to represent nodes in the cloud we get model needed self-reflection. However, we proved recently, both for Interaction Machines with simple nodes of the class of finite automata [12], and for Evolutionary Finite Automata [2] that they can decide both recursive enumerable but not recursive languages as well as non-recursive enumerable languages. Both above hypercomputational

models consist of infinite linear chains of simple nodes. Their tremendous expressiveness is due to either of interaction or evolution of an infinite number of simple nodes (of the class of finite automata). If we replace nodes in Evolutionary Finite Automata by a full blown TM, we get as a surprise that such collection, i.e., Evolutionary TM [5] is not more expressive than Evolutionary Finite Automaton [2]. Of course, in the case of the DIME network architecture, in implementation, we can approximate only an infinite number of nodes or channels by their finite subsets.

Conflicts of Interest: The author declares no conflict of interest.

References

1. Burgin, M. *Super-Recursive Algorithms*; Springer: New York, NY, USA, 2005.
2. Burgin, M.; Eberbach, E. Evolutionary automata: Expressiveness and convergence of evolutionary computation. *Comput. J.* **2012**, *55*, 1023–1029, doi:10.1093/comjnl/bxr099.
3. Burgin, M.; Mikkilineni, R. Agent technology, superrecursive algorithms and DNA as a tool for distributed clouds ang grids. In Proceedings of the 2016 IEEE 25th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), Paris, France, 13–15 June 2016; pp. 89–94.
4. Eberbach, E.; Goldin, D.; Wegner, P. Turing's Ideas and Models of Computation. In *Alan Turing: Life and Legacy of A Great Thinker*; Teuscher, Ch., Ed.; Springer-Verlag: Berlin, Germany, 2004; pp. 159–194.
5. Eberbach, E. Toward a theory of evolutionary computation. *BioSystems* **2005**, *82*, 1–19.
6. Eberbach, E.; Mikkilineni, R.; Morana, G. Computing models for distributed autonomic clouds and grids in the context of the DIME Network Architecture. In Proceedings of the 21st IEEE International Conference on Collaboration Technologies and Infrastructures WETICE 2012, Track on Convergence of Distributed Clouds, Grids and Their Management, Toulouse, France, 25–27 June 2012; pp. 125–130.
7. Gödel, K. Über formal unentscheidbare Sätze der Principia Mathematica und verwander Systeme. *Monatschefte für Mathematik und Physik* **1931**, *38*, 173–198.
8. Hopcroft, J.E.; Motwani, R.; Ullman, J.D. *Introduction to Automata Theory, Languages and Computation*, 3rd ed.; Addison-Wesley: Boston, MA, USA, 2007.
9. Mikkilineni, R.; Comparini, A.; Morana, G. The Turing O-Machine and the DIME network architecture: Injecting the architecture resiliency into distributed computing. In Proceedings of the Turing Centenary Conference, Turing-100, Alan Turing Centenary, EasyChair Proc. in Computing, EPiC, Manchester, UK, June 2012; Volume 10, pp. 239–251.
10. Turing, A. On Computable Numbers, with an Application to the Entscheidungsproblem. *Proc. Lond. Math. Soc*. **1936**, *42*, 230–265; A correction, ibid, **1937**, *43*, 544–546.
11. Turing, A. Systems of logic based on ordinals. *Proc. Lond. Math. Soc.* **1939**, *45*, 161–228.
12. Wegner, P.; Eberbach, E.; Burgin, M. Computational Completeness of Interaction Machines and Turing Machines. In Proceedings of the Turing Centenary Conference, Turing-100, Alan Turing Centenary, EasyChair Proc. in Computing, EPiC, Manchester, UK, 23 June 2012; Volume 10, pp. 405–414.