From Personal Computers to Personal Computing Networks

A New Paradigm for Computation

Mark Burgin
UCLA
Los Angeles, USA
mburgin@math.ucla.edu

Rao Mikkilineni and Giovanni Morana
C³ DNA
Santa Clara, USA
{rao,giovanni}@c3dna.com

Abstract— **Transition from mainframe computers to personal computers marked a new important step in computer technology. Here, we suggest a new transition from personal computers to personal computing networks that, as proven in scientific literature, they can be more powerful and efficient in computation. An efficient tool for personal computing networks is the Distributed Intelligent Managed Element (DIME) network architecture, which extends the conventional computational model of information processing networks, allowing improvement of the efficiency and resiliency of computational processes. This approach is based on organizing the process dynamics under the supervision of intelligent agents that, knowing the intent of the underlying process, is able to optimize its execution. In this paper, we will discuss about main ideas, structural features and tentative applications of personal computing networks and will explain why the DIME network architecture is suitable to build them.**

Keywords- personal computer; personal computing network; Oracle; DIME network architecture; structural operation; connectivity; modularity.

I. INTRODUCTION

After their creation, electronic computers existed in the form of a mainframe computer where the end user's requests are filtered through operating staff, or a time sharing system in which one large processor is shared by many individuals. A new important step in computer technology followed with the transition from mainframe computers to personal computers intended for interactive individual use, as opposed to time sharing access to mainframe computers.

Efficiency of personal computers has grown very fast and contemporary personal computers are more powerful than gigantic mainframe computers, which existed in the past. However, people need to solve more and more complex computational problems. To do this, engineers build more and more efficient and fast computers, develop new architectures such as Grid and Cloud Computing [1] [2] [3] while programmers write more and more complex distributed software systems [4] [5] [6] [7].

Here we suggest a new computational paradigm, which presuppose the transition from personal computers to personal computing networks (PCN), a set of distributed resources provisioned on demand (often provided by different service providers) that may have global reach using private and public clouds. The goal of PCN is to solve computational problems of the user with a specific goal with appropriate resources throughout the computation life-cycle to maintain optimal or desired availability, performance, security, compliance and cost.

As it is proved in [8], computation using networks of computers can be more powerful and efficient than computation using an individual computer. Being an efficient environment for concurrent computations, this approach will unleash a revolution of new possibilities in computer technology and applications of computers.

An efficient tool for PCNs is the distributed intelligent managed element (DIME) network architecture [9] [10] [11] [12] [13], which extends the conventional computational model of information processing networks, allowing improvement of the efficiency and resiliency of computational processes. This approach is based on organizing the process dynamics under the supervision of intelligent agents that, knowing the intent of the underlying process, is able to optimize its execution.

The DIME network architecture (DNA) utilizes the DIME computing model with non-von Neumann parallel implementation of managed Turing machines with a signaling network overlay adding cognitive elements to evolve super recursive information processing, for which it is proved that they improve efficiency and power of computational processes.

The aim of this paper is to explain why the DIME network architecture is suitable to build PCNs. The paper is organized as follows. Section II of this paper, describes the main ideas, structural features and tentative applications of PCNs. Section III describes how it is possible to build PCNs with assured distributed resources throughout the computation life-cycle based on the DIME network architecture. In Section IV, we apply theoretical models to study properties of PCNs, while in Section V, some conclusions are considered and directions for future work are suggested.

II. PERSONAL COMPUTING NETWORKS

Here we review the main ideas, structural features and tentative applications of PCNs.

There are different approaches to the PCN architecture. The goal of a PCN is to allow the user to work with a network of computers in a similar way to working with a single computer. The simplest solution is to form a

Copyright (c) IARIA, 2015. ISBN: 978-1-61208-389-6

computing network and to provide access to each of its computers for the user.

However, to interact with a personal distributed network of computers at the same time is an unmanageable task. That is why to achieve this goal, it is necessary to have a special machine, which on one hand, provides an efficient interface for the user, while, on the other hand, manages functioning of all computers, called *basic network computers*, in the PCN. We call such a machine the *network Oracle* because it has to be more powerful and have more information that the basic computers in the PCN.

There are different modes how the network Oracle can manage the basic network computers. Here we consider three prime modes of such a management:

1. Information supply
2. Control
3. Supervision

Definition 1. *Information supply* of the network Oracle is the function of the Oracle in Turing machines where the computing machine from time to time goes to the Oracle to get information in the form of data that the Oracle already has.

Definition 2. When the network Oracle *A controls* functioning of the basic computers from its network, *A* monitors these computers all the time changing, if necessary, functioning of any of these computers.

Definition 3. When the group Oracle *A supervises* functioning of the basic computers from its network, *A* interacts with these computers from time to time, verifying if the functioning of any of them is correct, providing necessary data and allocating instructions for forthcoming work.

Possible applications are:

A. *Information search.*

Each basic network computer from a PCN uses a specific search engine for its task. For instance, one computer uses Google, another utilizes Yahoo while the third one applies Microsoft's Bing. After each search cycle, the network Oracle collects the most relevant results from all basic computers, excluding repetitions and reorganizing data. Then it transmits the results to the user and assigns new tasks to the basic network computers. In such a way, the user receives more relevant and organized information.

When it is necessary to find information about different objects, e.g., different terms, each basic network computer can explore only one object. As a result, the PCN performs search in a parallel mode decreasing time of the search.

B. *Computer simulation.*

Each basic network computer from a PCN uses a specific simulation algorithm or/and simulation technique for its task. After each simulation cycle, the network Oracle collects the obtained results from all basic computers, excluding repetitions, eliminating irrelevant information and reorganizing data. Then, it transmits the results to the user and assigns new tasks to the basic network computers. In such a way, the user receives more relevant and organized information.

When it is necessary to simulate different systems, e.g., air currents and ocean currents, each basic network computer can simulate only one system. As a result, the PCN performs simulation in a parallel mode decreasing time of the simulation.

III. DIME NETWORK ARCHITECTURE AS A BASIS FOR PERSONAL COMPUTING NETWORKS

The DIME network architecture introduces three key functional constructs to enable process design, execution and management to improve both resiliency and efficiency of computing networks:

1. Machines with an Oracle
2. Blue-print or policy managed fault, configuration, accounting, performance and security monitoring and control
3. DIME network management control overlay over the managed Turing Oracle machines

A. *Machines with an Oracle*

Executing an algorithm, the DIME basic processor *P* performs the {read ⇒ compute ⇒ write} instruction cycle or its modified version the {interact with external agent ⇒ read ⇒ compute ⇒ interact with external agent ⇒ write} instruction cycle. This allows the external agent to influence the further evolution of computation, while the computation is still in progress. We consider three types of agents:

(a) A DIME agent.
(b) A human agent.
(c) An external computing agent.

In a PCN, we use DIME with several basic processors The DIME agent plays the role of the network Oracle in its PCN, while the basic DIME basic processors function as the basic computers from this network.

It is assumed that a DIME agent knows the goal and intent of the algorithm (along with the context, constraints, communications and control of the algorithm) the DIME basic processor is executing and has the visibility of available resources and the needs of the basic processor as it executes its tasks. In addition, the DIME agent also has the knowledge about alternate courses of action available to facilitate the evolution of the computation to achieve its goal and realize its intent. Thus, every algorithm is associated with a blueprint (analogous to a genetic specification in biology), which provides the knowledge required by the DIME agent to manage the process evolution. An external computing agent is any computing node in the network with which the DIME unit interacts.

Copyright (c) IARIA, 2015. ISBN: 978-1-61208-389-6

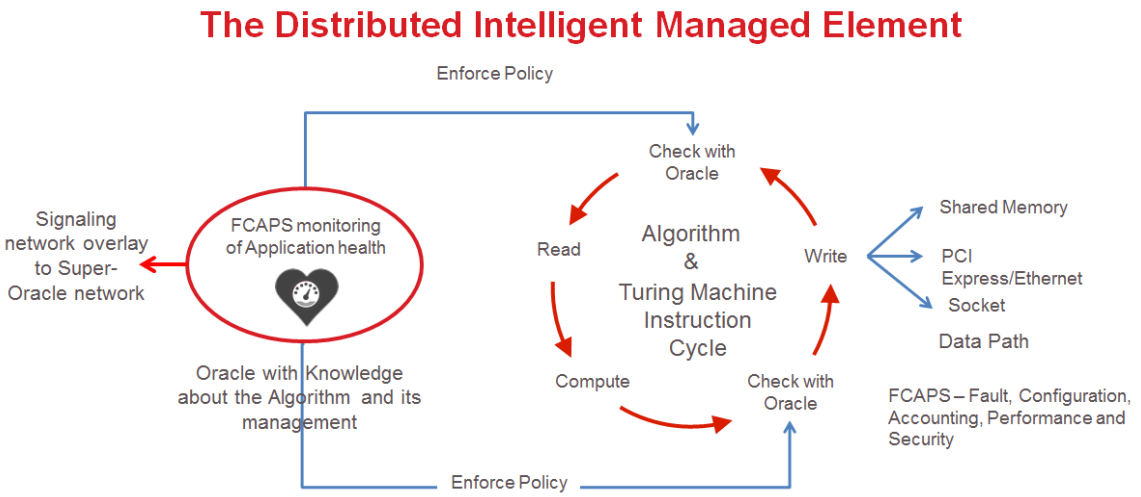

Figure 1. The distributed intelligent managed element (DIME) is a managed Turing Oracle Machine endowed with a signaling network overlay for super recursive policy based DIME network management

B. *Blue-print or policy managed fault, configuration, accounting, performance and security monitoring and control*

The DIME agent, which uses the blueprint to configure, instantiate, and manage the DIME basic processor executing the algorithm uses concurrent DIME basic processors with their own blueprints specifying their evolution to monitor the vital signs of the DIME basic processor.

It also implements various policies to assure non-functional requirements such as availability, performance, security and cost management while the managed DIME basic processor is executing its intent. Figure 1 shows the DIME basic processor (executing its "Algorithm & Turing Machine Instruction Cycle") and its DIME agent (performing "FCAPS monitoring of Application health). The DIME agent extends the basic processor capability adding the "Check with Oracle" operation: this allows the DIME agent to infuse to the underlying basic processor new knowledge, stored as the blueprint [15] and coming from both local information, collected by the DIME agent itself, and global information, received from the network of oracles through the signaling channel.

C. *DIME network management control overlay over the managed Turing Oracle machines*

In addition to read/write communication of the DIME basic processor (the data channel), other DIME basic processors communicate with each other using a parallel signaling channel. This allows the external DIME agents to influence the computation of any managed DIME basic processor in progress based on the context and constraints. The external DIME agents are DIMEs themselves. As a result, changes in one computing element could influence the evolution of another computing element at run time without halting its Turing machine executing the algorithm. The signaling channel and the network of DIME agents can be programmed to execute a process, the intent of which can be specified in a blueprint. Each DIME basic processor can have its own Oracle managing its intent, and groups of managed DIME basic processors can have their own domain managers implementing the domain's intent to execute a process. The management DIME agents specify, configure, and manage the sub-network of DIME units by monitoring and executing policies to optimize the resources while delivering the intent.

Figure 2 shows the DIME network architecture implementation for a process with different hardware, functions and an evolving structure used to attaining the intent of the process. This architecture has following benefits from current architectures deploying virtual machines to provide cloud services such as self-provisioning, self-repair, auto-scaling and live-migration:

1. Using DNA same cloud services can be provided at application and workflow group level across physical or virtual servers. The mobility of applications comes from utilization of the policies implemented to manage the intent through the signaling network overlay over the managed computing network. Applications are moved into static Virtual Machines or physical servers with given service levels provisioned.

2. Scheduling, monitoring, and managing distributed components and groups with policies at various levels de-couple the application/workflow management from underlying distributed infrastructure management systems. The vital signs (CPU, memory, bandwidth, latency, storage IOPs, throughput and capacity) are monitored and managed by DIMEs, which are functioning similar to the Turing Oracle Machines.

Copyright (c) IARIA, 2015. ISBN: 978-1-61208-389-6

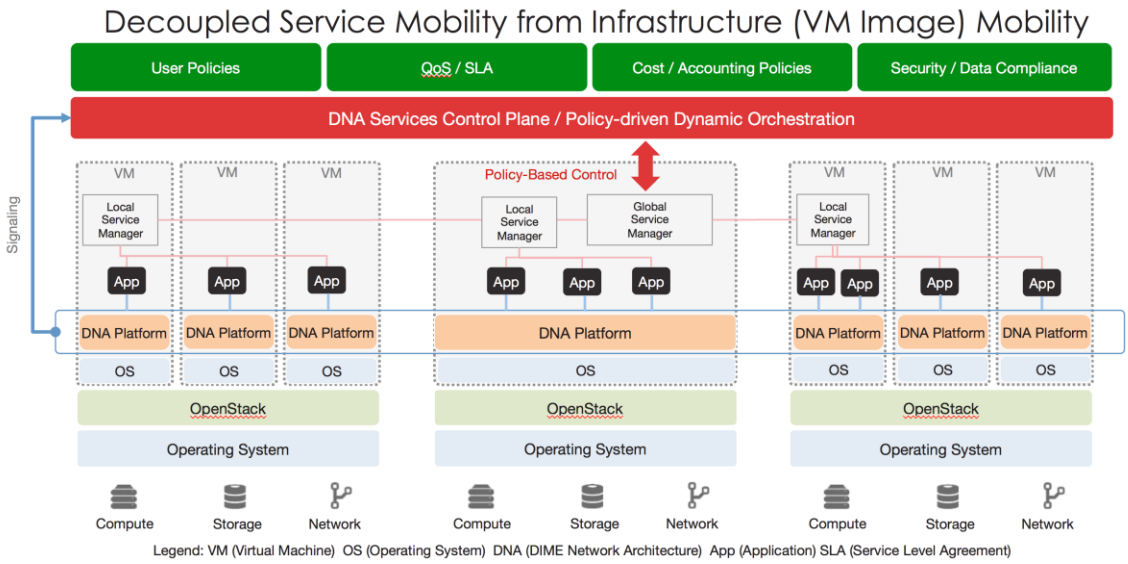

Figure 2. A managed application network with super recursive global, local and process level policy management with a signaling control network overlay

While implementing the monitoring and management of the DIME agent, the DIME network monitors and manages its own vital signs and executing various policies to assure availability, performance and security. At each level in the hierarchy, a domain specific task or workflow is executed to implement a distributed process with a specific intent. In Figure 2, each web component has its own policies and the group has the service level policies that define its availability, performance and security. Based on policies, the elements are replicated or reconfigured to meet the resource requirements based on monitored behavior.

In essence, the DIME computing model infuses sensors and actuators connecting the DIME basic processor with the DIME agent to manage the DIME basic processor and its resources based on the intent, interactions and available resources. Higher level policy managers are used to configure, monitor and manage the intent of a network of lower level managed basic processors.

The DIME network architecture has been successfully implemented using both Linux and Parallax [9][10][14], an operating system designed to support natively the DIME-based approach. More recently, a product based on DIME network architecture was used to implement auto-failover, auto-scaling, and live-migration of a web based application deployed on distributed servers with or without virtualization [15].

The mobility of the applications is provided by using the Oracle interrupt to control the down-stream processes. The global knowledge of down-stream process activity allows the higher level Oracles to reason and affect changes to the down-stream process dynamics. In addition to read/write communication of the basic automaton (the data channel), the Oracles manage different basic automata communicating with each other using a parallel signaling channel. This allows the external Oracles to influence the computation of any managed basic automaton in progress based on the context and constraints just as a Turing Oracle is expected to do.

The Oracle uses the blueprint to configure, instantiate, and manage the automaton and executing the algorithm. Utilization of concurrent automata in the network with their own blueprints specifying their evolution to monitor the vital signs of the DIME basic automaton and to implement various policies allows the Oracle to assure non-functional requirements such as availability, performance, security and cost management, while the managed DIME basic automaton is executing its task to achieve its goal and realize its intent.

The external Oracles represent DIME agents, allowing changes in one computing element influence the evolution of another computing element at run time without stopping its basic automaton executing the algorithm. The signaling channel and the network of the Oracles can be programmed to execute a process whose intent itself can be specified in a blueprint. Each basic automaton can have its own Oracle managing its intent, and groups of managed basic automata can have their own domain managers implementing the domain's intent to execute a process. The management Oracles specify, configure and manage the sub-network of DIMEs by monitoring and executing policies to optimize the resources while delivering the intent. The DIME network implementing the Oracles is itself managed by monitoring its own vital signs and executing various FCAPS (i.e. Fault, Control, Accounting, Performance and Security) policies to assure availability, performance and security.

An Oracle is modeled by an abstract automaton that has higher computational power and/or lower computational complexity than the basic automaton it manages. For instance, the Oracle can be an inductive Turing machine, while the basic automaton is a conventional Turing machine.

Copyright (c) IARIA, 2015. ISBN: 978-1-61208-389-6

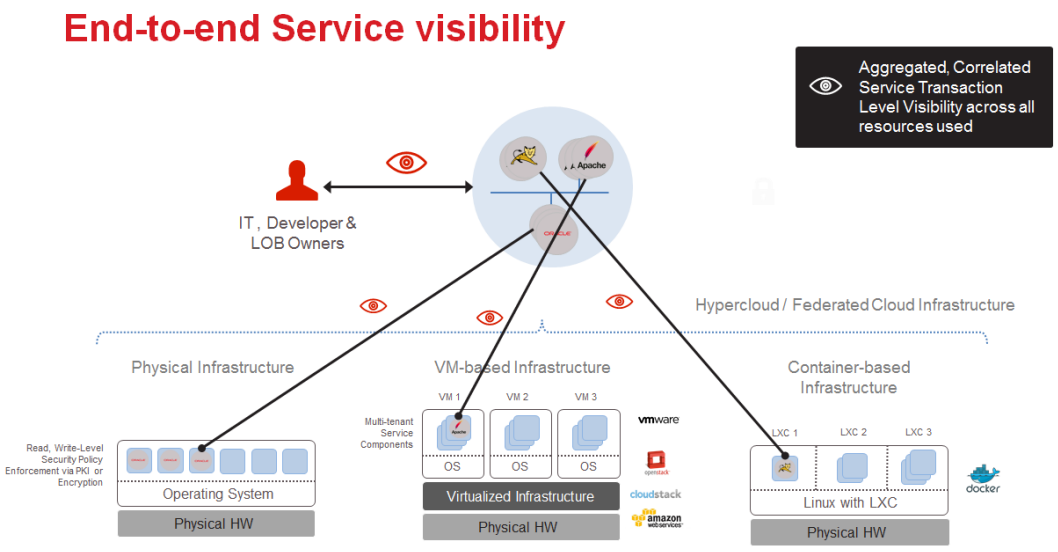

Figure 3. A Web application suite deployed using DIME network architecture with policies to monitor and manage the service availability, performance and security using auto-scaling, auto-fail-over and live migration

It is proved that inductive Turing machines have much higher computational power and lower complexity than conventional Turing machine [16][17][18]19].

DIME agents possess a possibility to infer new data and knowledge from the given information. Inference is one of the driving principles of the Semantic Web, because it will allow us to create software applications quite easily. For the Semantic Web applications, DIME agents need high expressive power to help users in a wide range of situations.

To achieve this, they employ powerful logical tools for making inferences. Inference abilities of DIME agents are developed based on mathematical models of these agents in the form of inductive Turing machines, limit Turing machines [16] and evolutionary Turing machines [18][20][21][22].

Figure 3 shows a DNA workflow of a web application running on a physical infrastructure that has policies to manage auto-failover by moving the components when the vital signs being monitored at various levels are affected. For example if the virtual machine in the middle server fails, the service manager at higher level detects it and replicates the components in another server on the right and synchronizes the states of the components based on consistency policies. A similar schema, described in details in another work [15], is adopted for handling (live) migration and (auto) scaling-out.

Figure 4 shows, in details, how a network of Oracles can be deployed in order to allow the user/developer of the application to have full control over an application. A user (a developer of a web service in this case) can provision resources by defining the intent of the global Oracle and use multiple Oracles at various levels to implement, monitor and manage the life-cycle of each component of the web application The DIME network architecture allows components to be developed and composed into services, deploy them, monitor the resources and their behavior and take corrective actions to fulfill the intent.

In conclusion, the PCNs distinguish themselves with the following properties:

1. On-demand service provisioning with required resources,
2. Auto-failover based on policies,
3. Stateful or stateless application migration and
4. End-to-end service visibility and control to assure availability, performance and security.

Users can create application components and compose them into service workflows and execute them on available distributed resources with dynamic service assurance.

IV. CAPABILITIES OF PRIVATE COMPUTING NETWORKS

Let us assume that it is possible to model the group Oracle and each computer from its group by a Turing machine. Results from [16] allow us to prove the following result.

Theorem 1. If the network Oracle *A* works in the recursive mode and controls functioning of basic network computers in the recursive mode, then it is possible to model the network functioning by a Turing machine.

Copyright (c) IARIA, 2015. ISBN: 978-1-61208-389-6

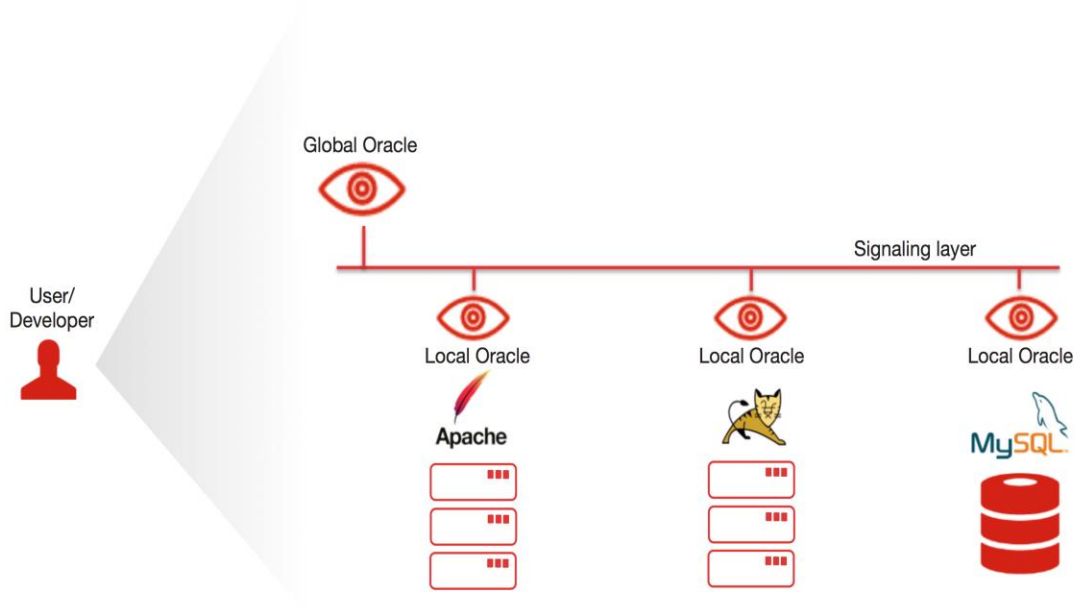

Figure 4. PCN for a Web Application

Results from [16] allow us to prove the following results.

Theorem 2. If the network Oracle *A* working in the recursive mode supervises functioning of basic network computers, then the supervised network can be more powerful than any Turing machine even if all basic network computers function in the recursive mode and *A* does not have noncomputable information.

Theorem 3. If the network Oracle *A* has recursively non-computable information, then the supervised network can be more powerful than any Turing machine.

V. CONCLUSION

Three innovations are introduced, namely, the parallel monitoring of vital signs (CPU, memory, bandwidth, latency, storage IOPs, throughput and capacity) in the DIME, signaling network overlay to provide run-time service management and machines with Oracles in the form of

DIME agents. This allows interruption for policy management at read/write in a file/device allow self-repair, auto-scaling, live-migration and end-to-end service transaction security with private key mechanism independent of infrastructure management systems controlling the resources and thus, provide freedom from infrastructure and architecture lock-in. The DIME network architecture puts the safety and survival of applications and groups of applications delivering a service transaction first using secure mobility across physical or virtual servers. It provides information for sectionalizing, isolating, diagnosing and fixing the infrastructure at leisure. The DIME network architecture therefore makes possible reliable services to be delivered on even not-so-reliable infrastructure. Modeling this architecture by grid automata allows researchers to study properties and critical parameters of semantic networks and provides means for optimizing these parameters. Future work will investigate specific predictions that can be made from the theory for a specific DIME network execution and compare the resiliency and efficiency using both recursive and super-recursive implementations.

ACKNOWLEDGMENT

Rao Mikkilineni and Giovanni Morana thank the team from C^3 DNA Inc., for their continued support in implementing the DIME network architecture.

REFERENCES

[1] R. Buyya, C. S. Yeoa, S. Venugopala, J. Broberga and I. Brandic, "Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility," Future Geeneration Computer Systems, Vol. 25, no. 6, pp. 599-616, 2009.

[2] J. Varia, Cloud Computing: Principles and Paradigms, Wiley Press, 2011, Best Practices in Architecture.

[3] M. Rahman, R. Ranjan, R. Buyya and. B Benatallah, A Taxonomy and Survey on Autonomic Management of Applications in Grid Computing Environments, Concurrency and Computation: Practice and Experience, Volume 23, Number 16, Pages: 1990-2019,, ISSN: 1532-0626, Wiley Press, New York, USA, November 2011.

[4] S. Jha, et al. "Distributed computing practice for large ‐ scale science and engineering applications." Concurrency and Computation: Practice and Experience 25.11 (2013): 1559-1585.

[5] D. Thain, T. Tannenbaum, and M. Livny. "Distributed computing in practice: The Condor experience." Concurrency and computation: practice and experience 17.2 ‐ 4 (2005): 323-356.

[6] A.D. Kshemkalyani, M. Singhal, Distributed Computing: Principles, Algorithms, and Systems, ISBN-13: 9780521876346, Cambridge University Press (2008).

[7] O. Babaoglu, K. Marzullo, Consistent Global States of Distributed Systems: Fundamental Concepts and Mechanisms, Distributed Systems, ACM Press (editor S.J. Mullender), Chapter 4, 1993.

[8] M. Burgin, Algorithmic Control in Concurrent Computations, in Proceedings of the 2006 International Conference on Foundations of Computer Science, CSREA Press, Las Vegas, June, 2006, pp. 17-23 .

[9] R. Mikkilineni, Designing a New Class of Distributed Systems. New York: Springer, 2011.

[10] R. Mikkilineni, G. Morana and I. Seyler, "Implementing Distributed, Self-Managing Computing Services Infrastructure using a Scalable, Parallel and Network-Centric Computing Model." In Achieving Federated and Self-Manageable Cloud Infrastructures: Theory and Practice, ed. M. Villari, I. Brandic and F. Tusa, pp. 57-78 2012.

[11] R. Mikkilineni, "Architectural Resiliency in Distributed Computing," International Journal of Grid and High Performance Computing (IJGHPC) 4. 2012 doi:10.4018/jghpc.2012100103. [retrieved: February, 2015]

[12] R. Mikkilineni, G. Morana, D. Zito, and M. Di Sano, "Service Virtualization Using a Non-von Neumann Parallel, Distributed, and Scalable Computing Model," Journal of Computer Networks and Communications, vol. 2012, Article ID 604018, 10 pages, 2012. doi:10.1155/2012/604018 [retrieved: February, 2015].

[13] R. Mikkilineni, "Going beyond Computation and Its Limits: Injecting Cognition into Computing." Applied Mathematics 3 (2012): 1826.

[14] R. Mikkilineni, A. Comparini and G. Morana, The Turing O-Machine and the DIME Network Architecture: Injecting the Architectural Resiliency into Distributed Computing, In Turing-100. The Alan Turing Centenary, (Ed.) Andrei Voronkov, EasyChair Proceedings in Computing, Volume 10, 2012.

[15] R. Mikkilineni and G. Morana, "Infusing Cognition into Distributed Computing: A new approach to distributed datacenters with self-managing services" Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), 2014 23rd IEEE International Conference, June 2014, pp.131-136.

[16] M. Burgin, Super-recursive Algorithms, New York: Springer, 2005.

[17] M. Burgin, Interactive Hypercomputation, in Proceedings of the 2007 International Conference on Foundations of Computer Science (FCS'07), CSREA Press, Las Vegas, Nevada, USA, 2007, pp. 328-333.

[18] M. Burgin, "Reflexive Calculi and Logic of Expert Systems", in Creative processes modeling by means of knowledge bases, Sofia, (1992) pp. 139-160 .

[19] J. P. Crutchfield and M. Mitchell, "Evolution of Emergent Computation" Computer Science Faculty Publications and Presentations. Paper 3. 1995
http://pdxscholar.library.pdx.edu/compsci_fac/3 [retrieved: February, 2015].

[20] M. Burgin, "Inductive Turing machines," Notices of the Academy of Sciences of the USSR, 270 N6 (1983) pp. 1289-1293 (translated from Russian).

[21] M. Burgin and E. Eberbach, "On Foundations of Evolutionary Computation: An Evolutionary Automata Approach," in Handbook of Research on Artificial Immune Systems and Natural Computing: Applying Complex Adaptive Technologies (Hongwei Mo, Ed.), IGI Global, Hershey, Pennsylvania, 2009, pp. 342-360 .

[22] M. Burgin and E. Eberbach, "Evolutionary Automata: Expressiveness and Convergence of Evolutionary Computation," Computer Journal, v. 55, No. 9 (2012) pp. 1023-1029.