proceedings

Proceedings

Self-managing distributed systems and globally interoperable network of clouds.†

Eng. Giovanni Morana, PhD [1,*]

[1] C3DNA Inc, 7533 Kingsburry Ct, Cupertino, CA 95014, USA
* Correspondence: giovanni@c3dna.com; Tel.: +39-349-094-1356
† Presented at the IS4SI 2017 Summit DIGITALISATION FOR A SUSTAINABLE SOCIETY, Gothenburg, Sweden, 12-16 June 2017.

Published: date: 9 June 2017

Abstract: This paper describes a new approach to autonomic computing that brings self-managing properties to applications and workflows of applications using a new computing model inspired by the functioning of the Turing Oracle Machine discussed in Alan Turing's Thesis. This approach, based on an effective management of the knowledge about the function, the structure and the fluctuations of the managed workflow, makes it possible to create an interoperable global network of clouds that can be used to support self-provisioning workloads with auto-scaling, auto-failover and live migration, guaranteeing the required level of QoS without disrupting user experience.

Keywords: Cognitive Distributed Computing; Distributed Reasoning; Signaling Layer; Common Knowledge

1. Introduction

Every IT workflow, either in scientific computing or business process management, is made up of several distributed, autonomous, interacting software components cooperating with each other to reach a common goal. The performance of each workflow, as well as its availability, robustness, resiliency, security and cost are the result of the combination of the workflow design, how it is deployed and the hardware characteristics of the machines hosting its components.

For many years workflows have been designed with a static context: each component is replicated multiple times to guarantee high availability and it runs on machines that are able to handle the highest estimated workload peaks without performance degradation. This robust but expensive and inefficient way to deploy workflows have been made outdated by the advent of Cloud Computing [1] and the concept of "* as-a-service" solutions (Infrastructure, Network, Storage and so on). Clouds and, more recently, SDN [2] and virtual networking technologies [3] have completely modified the way workflows are designed and, more important, managed.

The components of workflow are no more tightly coupled with each other or to the hosting hardware: in cloud-native workflow each component has to be designed stateless, addressable by logical name (location transparency) and able to interact with other components using asynchronous messages. Those characteristics, associated with the flexibility offered by the API of Cloud providers (which allows to build any type of hardware infrastructure with any configuration in terms of CPU, memory, network bandwidth and latency, storage capacity, IOPs and throughput), make possible a more effective management of the workflow that can dynamically balance between performance and cost.

Current state of the art allows creating virtual machine images of servers or containers on demand by using the services of the cloud provider. Cloud providers also support self-provisioning, auto-scaling and high availability zones. Other third party tools and point solutions also allow these virtual machine images or containers also to be deployed on different clouds. However these approaches suffer from two fundamental limitations: 1) vendor lock-in makes difficult their integration and 2) the lack of common, shared knowledge makes impossible an effective coordination of them.

This paper briefly introduces the DIME computing model [4-6], an innovative approach that uses a computing model inspired by the Turing Oracle Machine to demonstrate that any workload can be endowed with self-provisioning, auto-scaling, self-repair and live migration to any computing infrastructure without disrupting user experience.

2. Toward cognitive computing

According to Prof. Mark Burgin "efficiency of an algorithm depends on two parameters: power of the algorithm and the resources that are used in the process of solution. If the algorithm does not have necessary resources, it cannot solve the problem under consideration." The Turing computing model addresses how the computation is executed but it does not address the allocation and the management of resources needed to complete the computation. In the datacenter, the computing resources required for the computation depend both on their availability in terms of CPU, memory, network bandwidth, latency, storage capacity, IOPs and throughput characteristics of the hardware and also on the nature of the algorithm (the software). The efficiency of execution (i.e. the function) of the computation depends upon managing the dynamic relationship of the hardware and the software (i.e. the structure) to monitor the fluctuations and adjusting the resources to execute the computation. Often, based on the function of the computation, the structure of the computing resources may be altered (scaling, migration etc.) to meet the fluctuating demands for resources dictated by the computation. Thus function, structure and fluctuations dictate the evolution of computation.

In order to improve the efficiency of computation, the knowledge of how to manage the dynamics that is outside the purview of the computation is necessary, as well as an external model that integrates the computer (the hardware resources), the computed (the software), and the monitoring and management of their evolution in a consistent way. The DIME computing model implements a knowledge-driven, agent-based architecture (with a modified Turing Oracle design [7]) able to configure the components of a workflow and their hosting machines, monitor them and control their evolution to deliver their intent.

3. Cognitive Distributed Computing and Application Area Network

This section will describe the workload implemented as an Application Area Network (AAN) and its related hardware resources implemented as Distributed Computing Cluster (DCC) of resources.

The workflow described here is a real-word e-commerce application composed of 3 basic components:

- A web page, i.e. a virtual host on Apache HTTP Server, acting as front-end or "presentation layer": it collects all the input coming from users and forward it to the business layer.
- A web application, i.e. a java program hosted by an Apache Tomcat, acting as back-end or "business layer": it elaborates all the requests coming from the users, reading and writing on database when needed. It should be noted that the business layer could be decomposed in several java applications, each one executing a specific backend operation.
- A database, hosted by MySQL DBMS, representing the "data layer": it stores all the data and represents the status of the application. Being the only component storing data, it should be managed in a consistent way if replicated.

In order to improve scalability, the virtual host and the java application have to be stateless, interacting via asynchronous message-passing solution and their sessions have to be stored on clientside. All the three components including the MySQL database, have to be addressable by logical names.

Figure 1 shows current implementation of the Distributed Computing Cluster as a globally interoperable cloud network (with 10 different cloud providers).

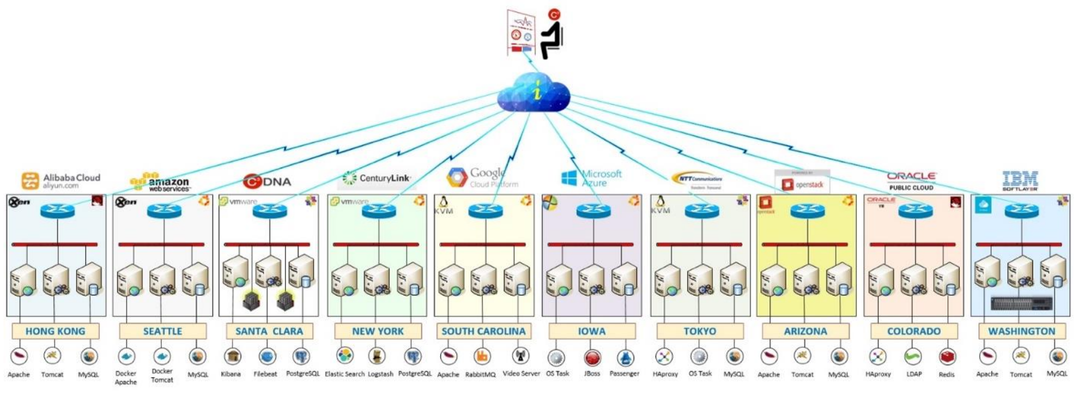


Figure 1. Globally Interoperable Cloud Network

First, a blueprint defines a static view of the workload, defining it in terms of its components (AAN). It contains details about where to find the needed software (i.e. the executable and/or specific configuration files) or how the components have to be configured (e.g. if they need to be directly exposed to the web, to be behind a load balancer or firewall or to be the master or the slave in a replication schema). It is important to note two fundamental aspects on this type of description:

- All this information is free of any hardware choice.
- It assumes that all aspects that require knowledge about the dynamic relationship of hardware and software, such as location transparency (DNS management, Load Balancing operations), security (firewall, IDS/IPS) and consistency (HA/DR), are automatically managed by the cognitive layer of the DIME Network Architecture, that represents the mediation point between the hardware and the software. It is able to act "as an oracle (of Delphi mentioned by Alan Turing in his thesis)" thanks to its ability to reasoning, in a decentralized fashion, on a global knowledge about the environment.

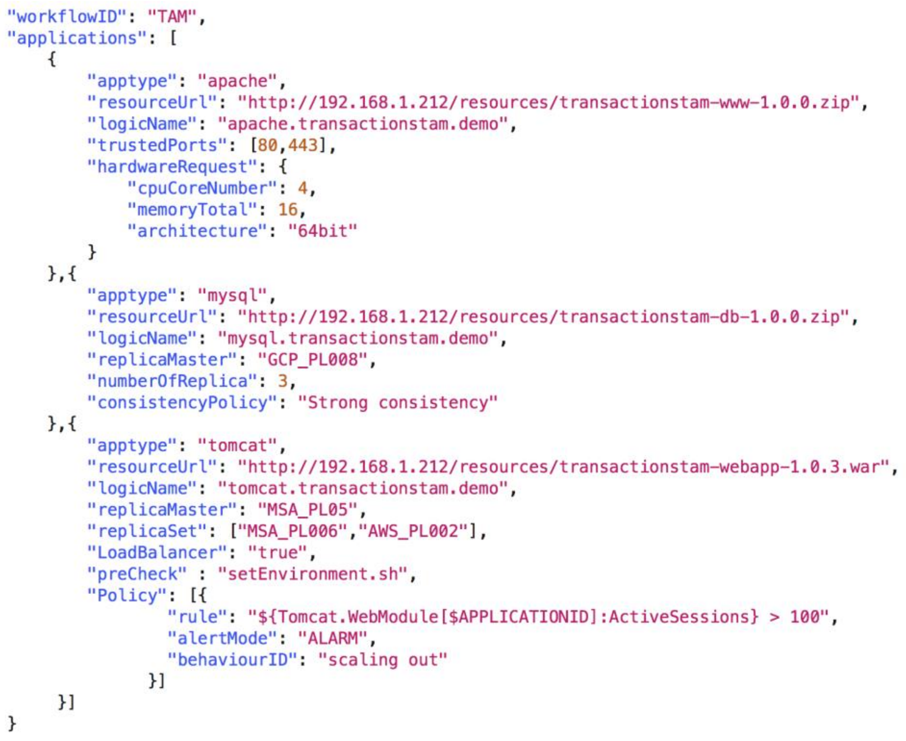
The last aspect is very important because it allows to overcome the vendor lock-in and enables the creation and dynamic management of a globally, interoperable cloud network on top of autonomous, independent and competitive infrastructure provided by myriad cloud providers.

Once the workflow composition is defined, its intent/goal must be added to the blueprint. This is usually defined in terms of performance, robustness and cost. The expected performance is given in terms of number of machines and in terms of constraints about the amount of their resources (CPU, memory, network bandwidth and latency and storage capacity, IOPs and throughput) and the threshold values when it is needed to address fluctuations. A trade-off between robustness and cost is used to decide how to address fluctuations both in workload demand and the available resource pools.

The definition of the expected performance translates the workflow blueprint in an infrastructure blueprint, which specifies the appropriate servers, storage and network (the DCC) requirements and represents the input for the workflow deployment. It is important to note that it is possible both to specify a direct mapping between a workflow component and a specific machine or define the list of wanted hardware characteristics (in terms of list of hardware resources or in terms of flavor, i.e., predefined set of given amount of resources).

Also in this case, it is important to note that the mapping between the components and the machines, i.e. the deployment of the workflow on top of the resources available on the multi-cloud network, is automatically managed (self-provisioning) by the cognitive layer of the DIME Network Architecture. This happens because some of the information needed to satisfy the workflow performance constraints (e.g. estimation for Recovery Time Objective/Recovery Point Objective in a scaled database or Disaster Recovery solution, which involves knowledge about two independent and autonomous machines belonging to two different providers) can be obtained only by a higher-level workflow manager (i.e., the DIME manager itself , which is aware of the two machines) , not by any of the actors involved in the workflow (neither software components nor resources provider) due to their "lack of global knowledge" (in this case, the IP addresses of all the machines).

The last step before the workflow deployment consists of defining all the corrective actions that have to be undertaken to effectively react to a deviation from an expected behavior. These corrective actions are defined in the workflow blueprint as policies (following the schema < IF deviation THEN action >) and represent a fundamental extension for enabling a *cognitive-aware, proactive* management. Each policy, which can be specified also at execution-time, can have a local or a global scope, can involve only one or multiple components of the workflow, can be execute only one time or can be persistent, can be predefined or specified by the user. Figure 2 shows a blueprint with different details for each component of the workflow.



```
"workflowID": "TAM",
"applications": [
    {
        "apptype": "apache",
        "resourceUrl": "http://192.168.1.212/resources/transactionstam-www-1.0.0.zip",
        "logicName": "apache.transactionstam.demo",
        "trustedPorts": [80,443],
        "hardwareRequest": {
            "cpuCoreNumber": 4,
            "memoryTotal": 16,
            "architecture": "64bit"
        }
    },{
        "apptype": "mysql",
        "resourceUrl": "http://192.168.1.212/resources/transactionstam-db-1.0.0.zip",
        "logicName": "mysql.transactionstam.demo",
        "replicaMaster": "GCP_PL008",
        "numberOfReplica": 3,
        "consistencyPolicy": "Strong consistency"
    },{
        "apptype": "tomcat",
        "resourceUrl": "http://192.168.1.212/resources/transactionstam-webapp-1.0.3.war",
        "logicName": "tomcat.transactionstam.demo",
        "replicaMaster": "MSA_PL05",
        "replicaSet": ["MSA_PL006","AWS_PL002"],
        "LoadBalancer": "true",
        "preCheck" : "setEnvironment.sh",
        "Policy": [{
                "rule": "${Tomcat.WebModule[$APPLICATIONID]:ActiveSessions} > 100",
                "alertMode": "ALARM",
                "behaviourID": "scaling out"
              }]
     }]
}
```

Figure 2. Example of Blueprint

An example of default and predefined policy is the self-repair of a component or a workflow: if a specific action is not defined, each time a fault occurs the DIME Workflow Manager tries to restart the component and, if it is not possible, restarts it in an another (or new, if needed) machine to guarantee the defined degree of availability. A more complex example of self-management capability provided by the policy-based solution is the auto-scaling of a database as a reaction to an unexpected increment of workload. The scaling, in fact, requires the knowledge about the workload, the ability to modify the workflow structure of the workflow to add a new copy of a component, the ability to modify the hardware infrastructure to spin up of a new machines (including the ability to interact with several resource providers, both private and public), the ability to save, move and restore date on different machines, the ability to modify the content of some services (such as DNS or load balancer) to guarantee the location transparency and more important, a consistent data synchronization across all the replicas.

Once deployed, the workflow starts its execution and the DIME Workflow manager will guarantee that it executes with the expected QoS, managing effectively any type of fluctuation that could deviate it from the optimal behavior.

A video example can be find at https://youtu.be/tu7EpD_bbyk .

4. Conclusion

This paper has described an implementation of a globally interoperable cloud network and demonstrated how workloads can be endowed with self-managing properties to cope with large scale and fluctuations both in workload demands and available resource pools. This approach provides the enterprise business owner end-to-end service visibility and control and the ability to leverage disparate heterogeneous cloud infrastructures without either the dreaded vendor lock-in or the complexity of myriad point solutions and tools. It provides a new level application layer availability, security and compliance across the cloud network using private or public network connectivity.

Theoretical results [8-10] demonstrate that Dime Computing Model extends the current computing model by infusing sensors and actuators into the conventional models of computation, such as Turing machine, as well as in more advanced models, such as inductive Turing machine, supplying cognitive behavior with super recursive computing. It is also proved that the DNA model is more efficient, powerful and expressive than the current Turing model based recursive computations. Consequently, the Dime Computing Model can essentially increase power and possibilities of the contemporary information processing technology

References

1. Buyya R.; Yeo C. S.; Venugopal S.; Broberg J.; Brandic I. Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility, FGCS, Volume 25, 2009, pp. 599-616, ISSN: 0167-739X
2. Kreutz D.; Ramos F. M. V.; Veríssimo P. E.; Rothenberg C. E.; Azodolmolky S.; Uhlig S. Software-Defined Networking: A Comprehensive Survey. In Proceedings of the IEEE , vol. 103, 2015, pp. 14-76 DOI: 10.1109/JPROC.2014.2371999
3. ETSI Portal. Network Functions Virtualization (NFV) White Paper #3 Available online: https://portal.etsi.org/Portals/0/TBpages/NFV/Docs/NFV_White_Paper3.pdf (accessed on 15 May 2017).
4. Mikkilineni R.; Morana G. Infusing Cognition into Distributed Computing: A New Approach to Distributed Datacenters with Self-Managing Services on Commodity Hardware (Virtualized or Not). In Enabling Technologies: Infrastructure for Collaborative Enterprises, Proceedings of the International IEEE WETICE Conference, Parma, Italy, 2014, pp. 131-136.
5. Mikkilineni R; Morana G.; Zito D. Cognitive Application Area Networks: A New Paradigm for Distributed Computing and Intelligent Service Orchestration In Enabling Technologies: Infrastructure for Collaborative Enterprises, Proceedings of the International IEEE WETICE Conference, Larnaca, Cyprus, 2015, pp. 51-56.
6. Mikkilineni R.; *Designing a New Class of Distributed Systems*. Springer, New York, USA, 2011.
7. Mikkilineni, R.; Comparini, A.; Morana, G. The Turing O-Machine and the DIME Network Architecture: Injecting the Architecture Resiliency into Distributed Computing. In The Turing Centenary Conference, EasyChair Proceedings in Computing, Manchester, UK, 2012 , pp. 239-251.
8. Eberbach E.; Mikkilineni R.; Morana G. Computing Models for Distributed Autonomic Clouds and Grids in the Context of the DIME Network Architecture. In Enabling Technologies: Infrastructure for Collaborative Enterprises, Proceedings of the International IEEE WETICE Conference, Toulouse, France, 2012, pp. 125-130.
9. Burgin, M. *Super-recursive Algorithm;* Springer, New York, USA, 2005
10. Burgin M.; Mikkilineni R.; Morana G. Intelligent Organization of Semantic Networks, DIME Network Architecture and Grid Automata. *IJES* **2016**, Vol. 8, pp. 352-366, DOI: http://dx.doi.org/10.1504/IJES.2016.077796.