Scaling and Self-repair of Linux Based Services Using a Novel Distributed Computing Model Exploiting Parallelism

Giovanni Morana
DIEEI
University of Catania
Catania, Italy
giovanni.morana@dieei.unict.it

Rao Mikkilineni
IEEE Member
Kawa Objects Inc.,
Los Altos, California, USA
giovanni.morana@dieei.unict.it

Abstract—This paper describes a prototype implementing a high degree of fault tolerance, reliability and resilience in distributed software systems. The prototype incorporates fault, configuration, accounting, performance and security (FCAPS) management using a signaling network overlay and allows the dynamic control of a set of nodes called Distributed Intelligent Managed Elements (DIMEs) in a network. Each DIME is a computing entity (implemented in Linux and in the future will be ported to Windows) endowed with self-management and signaling capabilities to collaborate with other DIMEs in a network. The prototype incorporates a new computing model (Mikkilineni, 2010) with signaling network overlay over the computing network and allows parallelism in resource monitoring, analysis and reconfiguration. A workflow is implemented as a set of tasks, arranged or organized in a directed acyclic graph (DAG) and executed by a managed network of DIMEs. Distributed DIME networks provide a network computing model to create distributed computing clouds and execute distributed managed workflows with high degree of agility, availability, reliability, performance and security.

Keywords - Communication System Signaling; Distributed computing; Distributed Software Systems; Workflow Automation; Network Centric Computing Model

I. INTRODUCTION

The introduction of virtualization technologies has brought great benefits in reducing the costs and making easier the management of the infrastructure. Virtualization represents an important step, at least, for two reasons: it guarantees the isolation of tenants and improves scalability. The isolation of one tenant (utilizing the computing, network and storage resources) from another assures fault, configuration, accounting, performance and security (FCAPS) management is at the server level in a physical server farm and is at a virtual server level in a virtual server farm. Moreover, with Virtual Machine (VM) density reaching up to ten (the number of guest VMs that can be effectively consolidated on a hypervisor without disruptive performance impact within a physical server) a ratio of 1 to 10 is possible to accommodate multiple tenants. Any further scaling can only be obtained with an architectural transformation to provide isolation and FCAPS management at much finer granularity at an object level providing a service in an individual (physical or virtual) server or at a transaction (end-to-end customer interaction including execution and persistence) spanning across multiple virtual servers.

The same scalability problem exists in many-core servers because the current generation operating systems, such as Linux and Windows, can support only few tens of CPUs in a single instance and are inadequate to manage servers that contain hundreds of CPUs, each with multiple cores. The solutions currently proposed for solving the scalability issue in these systems, i.e. the use of SSI [1] or the introduction of multiple instances of the OS in a single enclosure with high-speed connectivity (e.g. [2]), are inefficient because they increase the management complexity. In this paper, we propose to fill this operating system gap (defined as the difference between the number of cores available in an enclosure and the number of cores visible to a single image instance of an OS) using a new computing model called the Distributed Intelligent Managed Element (DIME) network computing model.

This model allows the encapsulation of each computing engine (a universal Turing Machine) into a self-managed computing element that collaborates with a network of other such elements to execute a workflow implemented as a set of tasks, arranged or organized in a directed acyclic graph (DAG). A signaling network overlay over the computing network of DIMEs allows parallelism in orchestrating the DIME network management to dynamically monitor, control and optimize the workflow execution based on business priorities, workload fluctuations and latency constraints. The self-management of each DIME incorporates a parallel monitoring, control and optimization of fault, configuration, accounting (of utilization), performance (FCAPS) and security of the services provided by each DIME utilizing its allocated resources. Using the signaling network the FCAPS management is extended to the DIME network. In Section II, we define the DIME network architecture (DNA). Sections III, and IV describe an implementation of DNA using Linux operating system to show a managed workflow that demonstrates (1) scaling of itself to adapt to workload and performance changes and (2) how the system tackles faults both in signaling and in execution layers. The DIME prototype demonstrates “live migration” of managed Linux processes without the use of hypervisors or virtualization as practiced today. Section V discusses related work and Section VI draws some conclusions from the implementation.
II. THE DIME COMPUTING MODEL

Each DIME is an autonomous computing entity endowed with self FCAPS management capabilities along with computing capabilities. It communicates with the other DIMEs via two parallel channels:

1. One for signaling each other to implement DIME network management which configures, secures, monitors, repairs and optimizes the workflow based on workload characteristics, latency constraints and business priorities specified as service management profiles\(^1\) (we call this profile the Service Regulator SR) and

2. Another for communicating with each other to implement a distributed business workflow as a DAG (in the von Neumann Stored Program Control Computing model). The tasks implemented by the DIME node is specified as a DAG (called the Service Package SP).

Signaling, used to monitor and control the business workflow implementation using the parallelism, provides a powerful approach in extending the current Service Oriented Architectures (SOA). By integrating computation and its management at the computing element level, and at the network level, this model exploits distributed resources, parallelism and networking to provide real-time telecom grade availability, reliability, performance, and security management of distributed workflow services execution. This is in total contrast to current approaches where the Operating Systems and management systems, which were designed in the days of CPU and memory resource constraints, and labor dependent administration paradigms are used to implement both computation and management workflows.

Each DIME is implemented as group of multi-process, multi-thread components, as shown in Figure 1.

Each one of the components constituting the DIME performs a specific function and, based on its configuration given to it by an external configuration file or by commands; each DIME can assume several roles (see Section III) in the management of workflows.

The DIME local Manager (DLM) sets up the other DIME components; it monitors their status and manages their execution based on local policies. Upon a request to instantiate a DIME, the DLM, based on the role assumed by the DIME, sets up and starts three independent threads to provide the Signaling Manager (SM), the Managed Intelligent Computing Element (MICE) Manager (MM) and the FCAPS Manager (FM) functions.

The SM is in charge of the “signaling channel”. It sends or receives commands related to the management and setting up of DIME to guarantee a scalable, secure, robust and reliable workflow execution. It also provides inter-DIME switching and routing\(^2\) functions.

The MM is a passive component which starts an independent process, the MICE, which in turn executes the task (or tasks), assigned to that DIME and, on completion, notifies the event to the SM. All the actions related to the task execution, which are performed by the MICE including memory, network, and storage I/O, are parameterized and can be configured and managed dynamically by the SM through the FM via the MM. This enables both the ability to set up the execution environment on the basis of the user requirements and, overall, the ability to reconfigure this environment, at runtime, in order to adapt it to new, foreseen or unforeseen, conditions (e.g. faults, performance and security conditions).

The FM is the “connection point” between the channels for workflow management and workflow execution. It processes the events received from the SM or from the MM and configures the MICE appropriately to load and execute specific tasks (by loading program and data modules available from specified locations). The main task of FM is the provisioning of FCAPS management for each task loaded and executed in the local DIME. This makes the FM a key component of the entire system:

1. It handles autonomously all the issues regarding the management of faults, resources utilization, performance monitoring and security,
2. It provides a “separation of concerns” which decouples the management layer from the execution layer
3. It simplifies the configuration of several environments on the same DIME to provide appropriate FCAPS management of each task that is assigned to the MICE which in turn, performs the processing of the task based on an associated profile.

Not all the components seen above have to be active, at the same time: the DLM will start only the components that are required to accomplish the functionalities specified by the role of each DIME in the network.

\(^1\) Profile based resource management is borrowed from Plain Old Telephone Service (POTS). As soon as a telephone goes off-hook, the network knows the service characteristics and requirements of the initiating party and as soon as the destination number is dialed, the network knows the service characteristics and requirements of the dialed party to allocate the appropriate resources and initiate the connection. In addition, the network monitors the connections, bills for the duration and assures the connection reliability, availability, performance and security. The DIME computing model performs the same function for the services that utilize computing, network and storage resources. Similar approaches are also used in PANS (Pretty Amazing New Services) on the Internet

\(^2\) Each DIME is globally addressable and supports network connectivity for both signaling and computing workflows using inter-DIME routing and switching.
III. IMPLEMENTING DIME NETWORKS TO EXECUTE A MANAGED WORKFLOW

The execution of a managed workflow needs the creation of an ad hoc network of DIMEs. In each network, in order to have two separated and parallel layers for managing and executing the workflow, it is necessary to foresee two different classes of DIMEs.

An ad hoc DIME network with parallel signaling and computing workflows is implemented using two classes of DIMEs:

1. Signaling DIMEs responsible for the management layer at the network level are of the type Supervisor and the Mediator. The Supervisor sets up and controls the functioning of the sub network of DIMEs where the workflow is executed. It coordinates and orchestrates the DIMEs through the use of the Mediators. A Mediator is a specialized DIME for providing predefined roles such as fault or configuration or accounting or performance or security management. The Configuration Manager performs network-level configuration management and provides directory services. These services include registration, indexing, discovery, address management and communication management with other DIME networks. The Fault Manager guarantees the availability and reliability in the sub network by coordinating the “Fault” components of the FM of all the DIMEs involved in the workflow provisioning. The Fault Manager detects and manages the faults in order to assure the correct completion of the workflow. The Performance Manager coordinates performance management at the network level and coordinates the performance using the information received through the signaling channel from each node. The Security Manager assures network level security by coordinating with the individual DIME component security. The Account Manager tracks the utilization of the network wide resources by communicating with the individual DIMEs.

2. Worker DIMEs constituting the “execution” layer of the network perform domain specific tasks that are assigned to them. A worker DIME, in practice, provides a highly configurable execution environment built on the basis of the requirements/constraints expressed by the developers and conveyed by the Service Regulator.

The deployment of DIMEs in the network, the number of signaling DIMEs involved in the management level, the number of available worker DIMEs and the division of the roles are established on the basis of the number and the type of tasks constituting the workflow and, overall, on the basis of the management profiles related to each task. The profiles play a fundamental role in the proposed solution; each profile, in fact, contains the indication about the control and the configuration of both the signaling layer and execution environment for setting up the DIME that will handle the related task.

The signaling DIMEs and the worker DIMEs in the current computing model have parallels in cellular biology where regulatory genes control the actions of other genes which allow them the ability to turn on or turn of specific behaviors. As affirmed by Philip Stanier and Gudrun Moore, [3] “In essence, genes work in hierarchies, with regulatory genes controlling the expression of ‘downstream genes’ and with the elements of ‘cross-talk’ between the regulatory genes themselves.” The same parallel, furthermore, exists between the task profile and the concept of gene expression. Gene expression is the process by which information from a gene is used in the synthesis of a functional gene product. With the same mechanism, the task profile (SR) is used to set up the environments in a DIME and execute the task (SP). As mentioned in Section I, each workflow assigned to the Supervisor DIME consists of a set of tasks arranged in a DAG. Each node of this DAG contains both the task executables (which itself could be another DAG) and the profile DAG as a tuple < task (SP), profile (SR) >: in this way, it is possible not only to specify what a DIME has to do or execute but also its management (how this has to be done and under what constraints). These constraints allow the control of FCAPS management both at the node level and the sub-network level. In essence, at each level in the DAG, the tuple gives the blueprint for both management and execution of the down-stream graph. Under these considerations, it is easy to understand the power of the proposed solution in designing self-configuring, self-monitoring, self-protecting, self-healing and self-optimizing distributed service networks.

The supervisor DIME, upon receiving the workflow, identifies the number of tasks and their associated profiles. It instantiates other DIMEs based on the information provided, by selecting the resources among the ones available, both the management and the execution layers. In particular, the number of tasks is used to determine the number of needed DIMEs while the information within the profiles becomes instrumental to define 1) the signaling sub-network, 2) the type of relationship between the mediator DIMEs composing the signaling sub-network and the FM of each worker DIME and, finally, 3) the configuration of all the MICEs of each worker DIME to build the most suitable environment for the execution of the workflow. In this way, the Supervisor is able to create a sub-network that implements specific workflows that are FCAPS managed both at management layer (through the mediators) and at execution layer (through the FM of each worker DIME).

IV. WORKFLOW IMPLEMENTATION USING DIME NETWORK

In this section a case study is presented for underlining the flexibility of the proposed architecture. In particular, the attention will be focused on creation of the DIME (sub)network that will execute the workflow showing, in particular, (1) how the system is able to scale up for adapting itself to workload and to performance changes and (2) how the system tackles faults both in signaling and in execution layers. The considered case study takes into account a workflow composed by four tasks, arranged in a simple DAG as shown in the figure 2.

Each one of the considered tasks is composed of the Service Regulator (SR) and the Service Package (SP). The first one contains all the information for setting up the management using the signaling layer of the DIME; the second one contains
all the information for setting up of the MICE, i.e. the execution layer.

![Workflow Diagram](image)

**Fig 2 The workflow considered in case study**

The workflow is submitted to Global Supervisor DIME that, once received, analyses it in order to understand the requirements and the constraints expressed by the user. The Global Supervisor, then, creates the sub-network where the workflow will be executed. As explained in the previous section, a DIMEs sub-network consists of a set of worker DIMEs and a set of signaling DIMEs. The number of DIMEs constituting the set of workers is equal to the number of tasks in the workflow. The DIMEs composing the signaling layer are known a priori but their number can vary based on the requirements and on the constraints of the workflow. The Global Supervisor, in fact, can decide to use six different DIMEs (one acting as local Supervisor and five as the different managers - Fault, Configuration, Account, Performance and Security managers - ) or a single DIME playing all the foreseen roles or, in addition, to use some dedicated DIMEs only for specific roles. Moreover, the Global Supervisor can decide to use new DIMEs for the signaling layer or to share the managers with other sub-networks. These decisions are taken based on the requirements of the workflow and based on the current workload managed by the Global supervisor. For example, if the workflow management requires a great number of messages for its coordination, the Global Supervisor can use a DIME for each manager. Instead, if the coordination of workflow requires only few signaling messages, the Global Supervisor can decide to use only a DIME for all the managers. The decisions taken by the Global Supervisor, however, can be modified at run-time. This ability, which represents one of the major advantages provided by the signaling based network-aware DIME organization, is fundamental for tackling the workload fluctuation typical of systems providing services on-demand. Simply activating or halting some components in DIMEs and/or modifying the used addressing schema, the Global Supervisor can decrease or increase the number of DIMEs dedicated for the management of signaling layer; in this way using it, we can tune optimally the behavior and, as a consequence, the performance of the workflow in execution. The features become very important especially, in those system where there can be several workflows working in parallel; based on workflow profile, on QoS parameters, on resources required and on costs paid by the users, the Global Supervisor can be used to organize the DIME network (and sub-networks) for optimizing given parameters (for example, for maximizing the resources utilization, for maximizing the system gain or for improve robustness of entire system).

Similar considerations can be used for choosing the position/deployment of worker DIMEs. The physical location of the DIMEs in the sub-network, in fact, can influence strongly the performance of the workflow.

If, for example, the workflow needs low-latency communications, the Global Supervisor is used to deploy the worker DIMEs as close as possible in term of communications delay. Based on the available resources and on the performance requirements, the Global Supervisor can choose to deploy the DIMEs on the same PC (shared memory-based or PIPE-based communication), on the same rack/server/blade (dedicated-bus communication, gigabit-Ethernet or Infiniband) or to distribute them over the internet (socket-based communications). As is the case with the signaling layer, the DIMEs belonging to the computing workflow layer are not also configured in a static way but can be re-arranged at run-time to adapt the workflow performance to the workload fluctuation or to react to unforeseen events. The MICE of worker DIMEs, in fact, can be dynamically re-configured for redirecting I/O at run-time. This ability, obtained using a proxy-like communication management system in both layers (signaling and execution), allows the Supervisor to modify the sub-network based on running workflow and on its workload needs. Once the best solution for organizing the sub-network of DIMEs is defined statically to execute the workflow, the Global Supervisor, starts a Local Supervisor and all the managers. Each manager, based on its offered functionalities and on workflow requirements, executes a list of controls: for example, they check the "heartbeat" of the system for fault prevention, authorization and authentication for security, or the timers for accounting and performance evaluation. If all the checks are successful, the Local Supervisor sets up the MICEs (through the Cmanager) of the entire worker DIME network for starting the execution of the workflow. If no problems occur, the workflow completes its execution correctly and the output is sent to the user; the sub-network is de-allocated and the DIMEs are made free.

Instead, if an error occurs during the execution of workflow, for example if the worker DIME executing the task#2 becomes unreachable (figure 3), the system starts the recovery of the errors and restores the correct functioning of the entire sub-network.

![Workflow Diagram](image)

**Fig 3DIME executing Task#2 becomes unreachable**

The fault manager, which monitors the "heartbeat" signal of each worker DIME, identifies immediately that a DIME has become unreachable, deduces that an error has occurred and
asks the Local Supervisor for a new worker DIME. The Local Supervisor, then, obtains the reference of a new DIME from the Configuration Manager; the Local Supervisor then configures the new DIME with the same SR and SC of task#2 and sends the reference of this new DIME to the worker DIME executing the task#1. The CManager of this worker DIME, then, modifies at run-time the I/O ports of the MICE in order to redirect the output of TASK#1 toward the new worker DIME (figure 4). All these actions, done autonomously and in a transparent way in relation with the other DIMEs of the sub-network, can be collected in both standard and ad hoc procedures. In the latter case, the user can specify to system a given behavior for reacting to specific faults.

This paper presents a specific implementation using Linux operating system to demonstrate the feasibility, assess its usefulness in real world service development and to propose further research. Current distributed computing practices have their origin from the server-centric von Neumann Stored Program Control (SPC) architecture that has evolved over the last five decades. The SPC von Neumann architecture prevalent today in our IT data centers is different from the von Neumann architecture for self-replicating system [7]. Various network computing models are discussed in the literature [8] starting from timeshared applications and Client server computing to Peer to Peer computing models [9, 10]. The Peer to Peer computing model called JP2P proposed by Junzhou Luo and Xiaoqiang Tong [10] comes closest in identifying some of the abstractions in the form of security layer, messaging layer and service layer in every computing node. It also identifies the need for two channels of communication one for control and management and another for content (data) transfer. Another close computing model to DIME computing model is Aneka [11, 12], a multi-model & SLA-Oriented platform for cloud/grid applications. Aneka supports container architecture for executing an application. The container supports directory, heartbeat, resource allocation, task execution, security, and communications services. While Aneka implements various FCAPS services and creates a distributing software systems architecture, the DIME computing model by introducing the out of band signaling channel and parallel FCAPS management architecture both in the node and at the network level, provides dynamic reconfiguration capability for implementing the workflows.

We believe that the signaling abstractions and their implementation to provide FCAPS management of a network of DIMEs (each DIME providing managed services) are critical to provide full hierarchical scalability and global interoperability very similar to telecommunication networks. The signaling allows the organization of task execution and helps the implementation of workflows with appropriate policies. The separation of business workflows and the management workflows that govern distributed software system creation, configuration and management, we believe, provides an order of magnitude simplicity and productivity compared to systems where these two functions are mixed and implemented in same application architecture. In addition, the model takes full advantage of parallelism, composition and scalability of the network abstractions.

V. RELATED WORK

The DIME computing model is a new approach that separates control flow and the computing workflow at a service level and allows the composition of services using the network computing model. A similar approach was first proposed by Goyal [4, 5]. The DIME computing model was formalized by Mikkilineni [6] in 2010. The services are encapsulated and an associated service profile defines how the services are used and managed so that they can be connected to other services to create a service network.
VI. CONCLUSION

In this paper, we present a new approach where self-management of computing nodes and their management as a group using a parallel signaling network are introduced to execute managed computational workflows. The DIME computing model does not replace any of the computational models in use today. Instead, it complements these approaches by decoupling the management workflow from computing workflow and providing dynamic reconfiguration, Fault Management, Utilization Management, Performance Management, and Security Management. We have implemented the DIME network computing model using Linux operating system and exploited the parallelism to demonstrate the dynamic Fault, configuration, performance, and security management. Two major enhancements from current computing models are demonstrated:

1. The parallel implementation of FCAPS management and computing integrates workflow execution and workflow management provides a new degree of reliability, availability, performance and security management in allocating resources at the node level.

2. The signaling scheme allows a new degree of agility though configuration management of each node at run time.

There are two ways in which the workflow can be dynamically changed. The first one is through the control of the MICE component to load and execute a specific program at run time. The second, instead, is through the reconfiguration of the I/O channels of the DIME to compose and decompose new services.

We believe that the DIME network computing model represents a generalization of other distributed computing models. The capability to set up the management layer, in fact, allows replicating the functioning of the other models such as, Clusters, P2P or Grid computing; one of the next steps of our research will be focused on the definition of well-defined profiles, used as templates for Signaling DIMEs, for setting up of DIME computing networks that are able to offer the same functionalities of those computing models with an added benefit of dynamic reconfiguration and real-time end-to-end integrated FCAPS management. The separation of service regulator executables from the service executables and their implementation in parallel with dynamic FCAPS management facilitated by signaling perhaps provides an object one class type higher, which von Neumann was talking about in his lecture at the Hixon Symposium [13]. "There is a good deal in formal logic which indicates that when an automaton is not very complicated the description of the function of the automaton is simpler than the description of the automaton itself but that situation is reversed with respect to complicated automata. It is a theorem of Gödel that the description of an object is one class type higher than the object and is therefore asymptotically infinitely longer to describe." It is for the mathematicians to decide. The DIME computing model, as we have shown in this proof of concept prototype, supports the genetic transactions of replication, repair, recombination and reconfiguration [14]. In conclusion, we summarize three potential uses this proof of concept points to:

1. Supporting dynamically reconfigurable communication mechanism (shared memory, or PCIexpress or Socket communication) under Linux operating system, we can improve the communication efficiency between multiple Linux images based on context that will make many-core servers more useful with less complexity of cluster management.

2. By encapsulating Linux based processes with FCAPS management and signaling, we can provide live migration, fault management and performance management of workflows without the need for a Hypervisor-based server virtualization.

3. Incorporating signaling in hardware provides a new level of agility to distributed services creation, delivery and assurance.

References