# Introduction

## Motivation

Reducing energy consumption and using biodegradable materials has become the main purpose in all the existent industry fields. In the last few decades, researchers have been focusing on improving performance in all possible fields, neglecting overall efficiency considering energy consumption. With the growth of impact of industry and pollution on the environment, the direction of evolution has changed from a performance directed one to an efficiency based approach.

Green computing is a new trend in computer science, encouraging the development of environmentally sustainable computing. Basically, it regards the study and practice of using computing resources efficiently. Studies have been focused on designing computing resources which maximize the energy efficiency during product’s lifetime, are biodegradable and don’t use hazardous materials. Extensive research is being undertaken in design of algorithms and systems for environmentally-friendly computer technologies and efficient use of computers.

On 2007, the report to congress on server and data energy efficiency [1] gave a definite warning in what regards the energy consumption of servers and data centers. From 2000 to 2006 the energy consumption of datacenters has doubled in United States, rising to about 61 billion kilowatt-hours (kWh) and being similar to the amount of electricity consumed by approximately 5.8 million average U.S. households. This is why a great part of the effort of painting the computing industry in green has to be directed towards decreasing datacenters’ and servers’ overall energy consumption, including the cooling infrastructure that supports IT equipment in datacenters.

Reducing the energy consumption in datacenters can be approached by improving the efficiency of computer resources or by software solutions of intelligently using those resources. Several solutions have been approached for decreasing datacenters’ overall energy consumption. These can be categorized in two main directions: hardware solutions and software solutions. The hardware solutions intend to decrease energy consumption by designing low power components thereby reducing the entire’s machine energy consumption. The software solutions are at their beginnings but there have been researches focused on decreasing power consumption depending on the existing load or on distributing efficiently tasks in the datacenter in order to have optimal power consumption.

## Objectives and contributions

The objectives of this thesis project are the following:

* A **description of the datacenter context** composed from servers which have certain resources as CPU, Storage, Memory and this **context’s mapping on the <R,A,P>**(Resources, Actors, Policies) model presented in [2].

The datacenter is composed of a number of servers with different characteristics for different resources like memory, storage and CPU. These are being mapped on a context model composed of Resources, Actors and Policies. The role of Resource is being played by any server in the datacenter and the task plays the Actor role. Considering the previously mentioned elements, policies are being defined using SWRL describing the situations in which the GPI ( Green Performance Indicators) for the datacenter, and the KPI (Key Performance Indicators) for the task are being respected.

* **Developing a self-\* algorithm** with the purpose of optimally distributing the workload, while using as less energy as possible.

Using the above context, a self-adapting algorithm is being designed, for the getting the described model to the lowest possible energy consumption. This algorithm involves a reinforcement learning approach by adapting the self-healing algorithm presented in [3], and adding few improvements. The self-healing algorithm is also being used for controlling the temperature and humidity in the datacenter room.

* **Negotiating the Service Level Agreement** of the task for the case in which the task doesn’t fit in any server on the datacenter, to modify existing policies for the datacenter and make it possible for the tasks to be properly distributed.

Because the user is giving us a requested range for the resources of a server, there might be a situation in which there is no place for that task due to GPI Policies. In this case, a negotiation should be made in order to have a higher maximum accepted value and the task to fit. There is also the case where the task could get more than the minimum requested value, case in which a redistribution of available resources among existing tasks on that server will be made.

## Publications

* + 1. **A Reinforcement Learning based Self-healing Algorithm for Managing Context Adaptation** [3]

This paper presents a reinforcement learning approach for finding the optimum sequence of actions for healing a broken context. For this algorithm to be able to function weights are added to resources and policies and concepts like entropy and inter-independent resources group are being introduced.

* + 1. **An Autonomic Algorithm for Energy Efficiency in Service Centers** [4]

This paper presents an improvement of the previous self-healing algorithm, used for the self-adaption of the energy-efficient datacenter. The self-adapting approach features a closed feedback loop with four MAPE phases: Monitoring, Analysis, Planning and Execution.

## Overview of the report

In the next parts of this report, I will present a theoretical background needed in order to present this project, with available software and technologies needed for implementing it on a real datacenter. The next chapter, chapter 3 gives a view on the existing work in the domain. The fourth chapter describes the algorithms and models and from here on the architecture, the design and implementation is presented ending with testing and conclusion.

# Theoretical Background

## Context-Aware Computing

Awareness is one of the main problems arising in nowadays computing systems. Building systems which are aware about what happens and about their awareness becomes crucial in hospitals, modern buildings and even personal houses. To handle contexts in all of these environments the system must have sensors for each monitored element and a way to control that element, through different actuators. In order for the system to know when the context is not in the state desired by the user, policies are being described for all elements composing the context. For example, in a smart laboratory, we need to know that if no one is inside the light should be off. Generally the system takes the sensor information and enforces actions on actuators through web services, this way having a low coupled architecture.

## Energy-Aware Computing

Energy-awareness is a subset of context awareness, improving the systems towards green use of the computing infrastructure. The green use of a datacenter implies reducing the energy consumption of computers and other information systems as well as using them in an environmentally sound manner. For the real servers and datacenters to have this behavior there are several options like programmatically assign loads to resources or programmatically assigning loads to servers. The load assignment for datacenters is easily accomplished through virtualization. By this, a system administrator could combine several physical systems into virtual machines on one single, powerful system, thereby unplugging the original hardware and reducing power and cooling consumption.

### Virtualization

According to Wikipedia, the term "virtualization" was coined in the 1960s, to refer to a virtual machine (sometimes called pseudo machine), a term which itself dates from the experimental IBM M44/44X system [5].Virtualization is a new technique reproducing computer hardware through software. In a typical server environment there exist different servers each hosting only one task, for example a web server and a file server. By using server virtualization, both the previously mentioned servers will be running on the same machine, one independently of the other, therefore reducing the costs and energy consumption of a second machine. The center of the entire virtualization process is the virtual machine, it being defined as a software implementation of a machine that executes programs like a physical machine. There are two types of virtual machines: system virtual machines and process virtual machines. From the process virtual machines, the JVM (Java Virtual Machine) and the .NET Framework, which runs on a VM called the Common Language Runtime, are the most known one. Process virtual machines run as a single application inside the operating system, and support one single process. They are created when that process is started, and destroyed when it exits. We are interested with the system virtual machines which allow the sharing of the underlying physical machine resources between different virtual machines, each running its own operating system [5]. There are several advantages coming with the use of system virtual machines, like the fact that multiple operating system environments can run on the same computer, in strong isolation from each other or that the virtual machine can provide an instruction set architecture different from that of the real virtual machine. The software layer providing virtualization for system virtual machines is called a virtual machine monitor or hypervisor. Due to the fact that it is an important part of datacenter administration having virtual machines in the role of tasks, hypervisor description will be detailed in the followings.

#### Hypervisors

The hypervisor, also called Virtual Machine Monitor (VMM) provides the guest operating system a virtual platform and monitors their execution. Despite the fact that the virtual machines can commonly used resources, the failure of one virtual machine won’t produce the failure of all the other virtual servers running on that machine. The isolation ensured by the hypervisor is one of the main features of virtualization, which brings it to the top of technologies to be used in modern datacenters. The hypervisors are split into two categories: software and bare-metal. Software hypervisors need a host operating system to run on and have lower I/O performance due to the overhead resulting from the hypervisor-host OS communication. Bare-metal hypervisors received their name from the fact that they run on “bare metal”, needing no host operating system. Actually the server must be formatted before this hypervisors are installed. This close connection to the underlying hardware brings better I/O performance and is also faster due to the removal of the layer introduced by the operating system.

Microsoft Hyper-V is the hypervisor which is the most present one in the data centers all over the world. It can be run on an x64 version of Windows Server 2008, the R2 version having the live migration feature enabled. In order for Hyper-V role to be enabled for a windows server, the processor needs to have hardware assisted virtualization. This is available for processors that include a virtualization option (Intel VT or AMD Virtualization). Live migration is supported with the use of cluster shared volumes (CSV). This feature is extremely important for enabling the movement of a virtual machine from a server to another, from efficiency reasons. Live migration of virtual machines from a server to another is done automatically for situations in which the node (the Hyper-V server) fails. In this situation, each virtual machine running on the failed node may migrate to other live nodes independently of other virtual machines. Due to the fact that this diploma project is undertaken within a larger group with different missions, we have chosen Hyper-V as a hypervisor, it providing both a high-level and a low level API. In there are presented on short two other hypervisors, one which offers full virtualization and one offering paravirtualization, together with a description of paravirtualization and comparison to full virtualization.

VMWare ESXi is a “bare metal” hypervisor, meaning that it doesn’t need to run on top of other operating systems. This implies a lower overhead and a better control and granularity for allocating resources (CPU time, disk bandwidth, etc.) and a considerable increase of security. VMWare ESX and VMWare ESXi offers advanced resource management capabilities to improve performance and increases consolidation ratios. Both Hyper-V and VMWare ESXi offer a full virtualization approach which allows datacenters to run an unmodified guest operating system, thus maintaining the existing investments in operating systems and applications and providing a nondisruptive migration to virtualized environments. On the other hand, the paravirtualization approach modifies the guest operating system to eliminate the need for binary translation. Therefore it offers potential performance advantages for certain workloads but requires using specially modified operating system kernels [6]. The Xen open source project was designed initially to support paravirtualized operating systems. While it is possible to modify open source operating systems, such as Linux and OpenBSD, it is not possible to modify “closed” source operating systems such as Microsoft Windows. It is also not practical to modify older versions of open source operating systems that are already in use. As it turns out, Microsoft Windows is the most widely deployed operating system in enterprise datacenters. For such unmodified guest operating systems, a virtualization hypervisor must either adopt the full virtualization approach or rely on hardware virtualization in the processor architecture [7].

#### Virtual appliances

A software appliance is a full application stack containing the operating system, the application software and any required dependencies, and the configuration and data files required to operate. Everything is preinstalled, preintegrated and ready to run. Software appliances come in the form of a file which can be a virtual machine image, an ISO, a USB key image, or an Amazon EC2 AMI. They run a JeOS(Just Enough Operating System) is a customized operating system that precisely fits the needs of a particular application (Ubuntu JeOS, OEL JeOS, SUSE JeOS, OpenSolaris JeOS, OpenSolaris JeOS, Orange JeOS, and Windows Server Core).The virtual appliances, a sub-class of software appliances, add to the advantages of software appliances the benefits of virtualization.

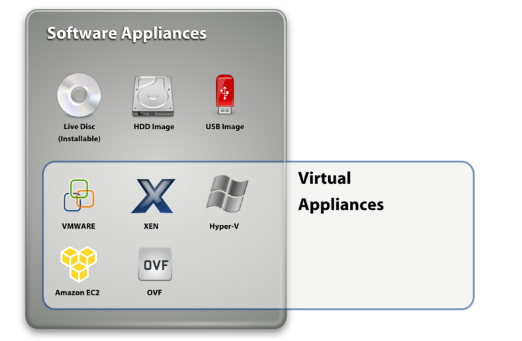


Figure 2.1: Software Appliances Taxonomy- © Novell Inc

The biggest advantage of virtual appliances is that there is a considerable energy economy because we don’t have any more tasks waiting one for another and redirection of conflicting tasks, being an alternative to massive changes and increased complexity on the software stack. Virtual appliances allow for rapid movement of virtual appliances between physical execution environments, provide an improved isolation between several appliances sunning on the same server and improve fault tolerance .For tasks that need to be sent to a data center and run on a server, a virtual appliance can be created with all the needed applications and only the needed part of the operating system and sent to be run secure and isolated from other tasks.

In terms of solutions to energy efficient datacenters, software appliances are a feasible alternative to virtual machines, considering that by having all the needed software and no more than that on top of a lightweight JeOS the time for programmatically creating virtual machines with different characteristics is reduced to 0 and the dimension of the virtual machine is reduced. Not all the hypervisors are able yet to host virtual appliances, therefore this is a subject for implementation after virtual appliances leave the research state and people get acquainted with them.

### Hyper-V WMI Provider

For monitoring virtual machines in a network of servers running Hyper-V, the Hyper-V WMI Provider is being used. It is a high level API, giving information about virtual machines and their status. It enables developers and scripters to build custom tools, utilities and enhancements for the virtualization platform, managing all aspects of the Hyper-V Services [8]. Most functions in this API are available in Basic, PowerShell, C# and C++, therefore the limitations from the programming language point are not too high.

## Autonomic Features

In a manifesto in 2001, IBM invites the world, their customers, competitors and colleagues to accept the Grand Challenge of building and deploying computing systems that regulate themselves and remove complexity from the lives of administrators and users. On short, they consider that the new Grand Challenge in computing industry is the overgrowing software complexity both in terms of management and in terms of maintenance. They believe that the growing complexity of the I/T infrastructure threatens to undetermine the very benefits that information technology aims to provide. Human intervention and administration to manage software complexity is starting to be overwhelmed. It is estimated that at current rates of expansion, there will not be enough skilled IT people to keep the world’s computing systems running.

Considering the fact that “in the evolution of humans and human society, automation has always been the foundation for progress” [9], IBM states that it’s time to design and build computing systems capable of running themselves, adjusting to varying circumstances and preparing their resources to handle most efficiently the workloads we put upon them. These autonomic systems must anticipate the needs and allow users to concentrate on what they want to accomplish rather than figuring how to rig the computer systems to get them there [10].

They give four directions of approach in terms of self-management, described bellow: self-configuration, self-healing, self-optimization and self-protection. Systems having all these capabilities are also called CHOP systems or self-\* systems. This thesis presents a self-healing approach for the datacenter room, and a self-optimizing approach for having a datacenter which consumes the optimum amount of energy.

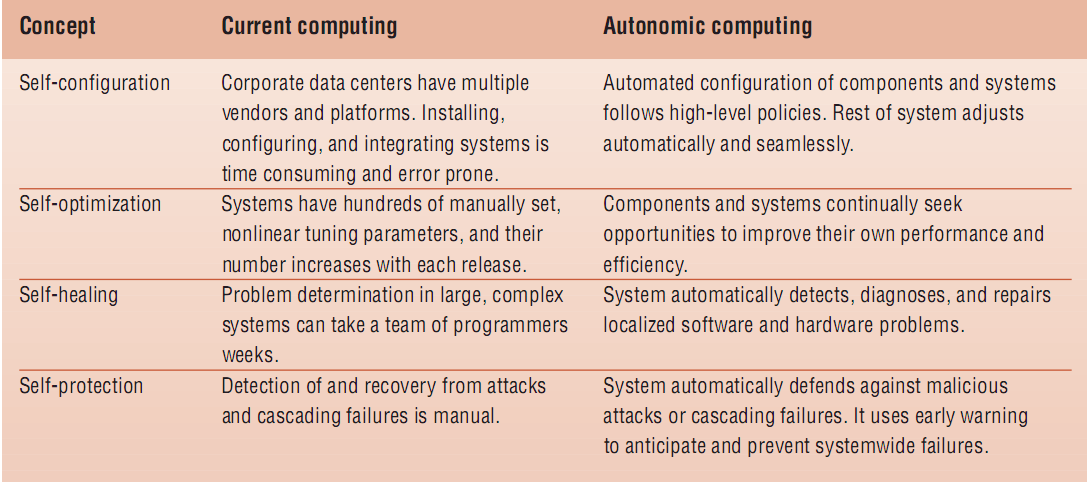


Figure 2.2: Four aspects of self-management © IBM

## Context Model

The most relevant context modeling approaches are presented in [11]. They are classified by the scheme of the data structures which are used to extract information: key-value models, markup scheme models, graphical models, object oriented models, logic based models and ontology based models. Out of these, the logic based model and ontology based model are the ones that store information on which it can easily be reasoned upon. The logic based model is a very formal model, defining conditions on which concluding expression or facts may be derived from a set of other expression or facts. Ontologies are of the most promising instruments to specify concepts and interrelations [11]. They are suitable both for projecting parts of the information used in our everyday life onto a data structure utilizable by computers, and for reasoning in that information with the help of a rule language layered on top of RDF. This is why the chosen context model for our context-aware system is an ontology model.!!!!!!!! OWL , RDF :D prescurtari The ancient Greek philosopher Parmenides of Elea was among the first to propose an ontological characterization of the fundamental nature of reality. An ontology together with a set of individual instances of classes constitutes a knowledge base. In reality, there is a fine line where the ontology ends and the knowledge base begins.For creating and modifying the ontology, the Protégé ontology editor and knowledge-base framework is being used.

## Mobile Agents – JADE

Java Agent DEvelopment(JADE) Framework is a mobile agents framework which simplifies the implementation of multi-agent systems. A mobile agent is a software agent with features like autonomy, social ability, learning and mobility. Mobile agents are extremely important in context-aware applications, each agent taking a responsibility and acting independently of the other agents. They can communicate and can be distributed on different machines, or can move even at run-time from one machine to another and resume the execution on the new machine.

# Related Work

Self-\* and context-aware systems have become leading domains in terms of research. The self-\* initiative has been started by IBM, in the manifesto from 2001, stating that developing autonomic systems is the new challenge in IT [9]. For a system to be self-\* it has to be aware of the context in which it has to function. Therefore, for creating a self-adapting datacenter, it has to be aware of the surrounding context (waiting tasks, temperature, and humidity) and of the energy context (servers’ power consumption, resources used by each deployed task). It will also need to have policies for it to know how to adapt in new situations. On the other hand, considering that the user will tell the datacenter how many resources he would need, and that the user tends to ask too much therefore the datacenter can be in a situation where it can’t fit the task on none of the servers, a negotiation process is needed for SLA. In the following subchapters, related work is presented for all the above mentioned problems.

## Context Aware Systems

Ubiquitous Computing is a term coined by Mark Weiser around 1988, and refers to the seamless integration of devices into the users’ everyday life. The appliances should be hidden into the background, to make the user aware of his tasks and priorities rather than computing devices and technical issues. One field of pervasive of ubiquitous computing is the context-aware (sentient) computing. Context aware systems operate without necessary user intervention, by taking environmental context into account. Smart rooms, which can be seen as intense ubiquitous computing environments, are a step toward Weiser’s [12] vision. Currently, there are already some smart rooms functioning at different organizations, like MIT’s Intellingent Room[13] or Stanford’s iRoom Project [14].

What seems to be a problem in context-aware computing is the privacy of people whose activity is monitored by pervasive computing systems and on whose behalf actions are being undertaken. This is why, in [15] Hong and Landay describe an architecture for privacy sensitive ubiquitous computing which addresses the issue generated by people’s concerns about the strong potential for abuse over a potential lack of control. Confab is a toolkit for facilitating the development of privacy-sensitive ubiquitous computing applications. Confab provides a framework for ubiquitous computing applications, where information is collected and processed as much is possible on the end-user’s computer. In this way, the user can specify how much of the information on his computer is available, and for what purposes. Confab therefore gives “an extendable design that provides a software architecture support for building privacy-sensitive ubicomp applications that are optimistic, pessimistic, and mixed-initiative” [15].

Currently most attempts to use context-awareness within ubiquitous computing have been focusing on the physical elements of the environment, or the user’s device. In contrast with that, a new context-awareness direction towards capturing the cognitive elements of a user’s context is explored by Prekop and Burnett with a conceptual model of Activity Centric Context [16]. The focus of the activity-centric view of the context is on the information surrounding the performance of an activity undertaken by an agent. The activity-centric view has, as its name states, the agents and activities in the centre of the entire perspective on the context. By monitoring and storing agent’s activities, each of the activity belonging to a higher lever activity together with its context, Prekop and Burnett’s paper provides a model for supporting complex context-aware applications after capturing agent’s behavior.

For supporting context-aware systems different ontologies and languages have been described, with the purpose of easing the pervasive frameworks.

The COBRA-ONT ontology for COntext BRoker Architecture (COBRA) is “a collection of ontologies for describing places, agents, events and their associated properties in an intelligent meeting room domain” [17]. The design of COBRA addresses important issues like supporting resource-limited mobile computing and addressing concerns for user privacy, and contains four essential components: a context knowledge base, a context reasoning engine, a context acquisition module and a policy management module.

In [18] Kagal et al. present a policy language specialized for pervasive systems. Rei is a policy language based on deontic concepts, which can be used to describe several kinds of policies. With its help, security policies can be described, restricting access to resources of organizations. It can be used to create actions on resources and describe restrictions and availabilities for users in an organization or defining conversation policies very important in autonomous environments. All of the above possibilities makes Rei a versatile and expressive policy language especially for context-aware computing systems, it being a branch of pervasive computing.

## Energy Aware Systems

For decreasing datacenters’ energy consumption, many solutions have been approached. Brogetto and Stolf describe in [19] an autonomous system which through virtualization and consolidation, manages to decrease energy consumption. They use TUNe as an autonomic job scheduler, and find the best consolidation solutions for reducing energy consumption. What differs from our approach is that we employ a learning algorithm for finding the solution, and that we also use as consolidation action the move actions, for moving virtual machines from one server to another.

In [20] it is used a machine learning approach with dynamic backfilling to optimally distribute tasks in a datacenter. Their approach applies some scheduling policies that reduce the number of unused machines according to the workload needs in each moment, and decide task placing and reallocation in order to compact jobs in the lowest number of machines with-out degrading their service level agreements (SLA). Tasks are regarded as normal applications, and no virtualization is undertaken. Also, the service level agreements aren’t being negotiated. The way in which one can break the SLA agreement is seen as a policy constraint.

HP scientists present in [21] an energy aware grid that is intended to provide global utility infrastructure, managing both energy efficiency matters and thermal issues in datacenters. Workload placement decisions are being taken considering energy coefficients and depending on them the resource co-allocator chooses Globus Resource Allocation Manager (GRAM) as destination for the workload. In addition to that, HP scientists have also designed a physical infrastructure for supporting a low energy consumption of the entire grid.

Srinivasan et al. present in [22] an approach which uses a Swarm Intelligence based approach (SITA) for Task Allocation and scheduling in a dynamically reconfigurable environment such as the computational Grid. An ant colony optimization technique is being used for optimal resource discovery in the Grid, with great adaptation on extraordinary situations like node failure, link failure, and congestion. With the role of ants some distributed agents are being used, which are working in parallel and independently of each other. They take into account both cost and time minimization, for obtaining a balance between these two conflicting items, using a constraint satisfaction based approach to task allocation.

In [23] an energy aware design of service-based applications evaluates KPI (key performance indicators) and GPI (Green Performance Indicators), trying to obtain the best tradeoff between the two of them. The energy efficiency of a service is defined depending on the service’s normal and idle mode, price and energy consumption. Their goal is to maximize the aggregated quality and energy values by considering all possible execution paths and find the best solution at least for the most frequent ones. Considering that both this paper and the thesis which is being presented now are part of the GAMES (Green Active Management of Energy in Service centers) project, there are several similarities between what Ferreira et al. present and the project being currently presented. They both use KPI and GPI for estimating energy efficiency and virtualization for a better encapsulation of tasks, but with different approaches. In [29], Ferreira et al. emphasize that the energy consumption is also a service quality problem. Through this approach, they tackle the energy efficiency problem at the service level as a nonlinear Service Concretization (SC) problem, considering both infrastructure characteristics and quality of service requirements. For a better description of the context, a the energy efficiency GPI metric is defined, its computation being based on attributes like execution time and energy measures like energy consumption.

## Self-\* Systems

Computing systems have reached a level of complexity where the human efforts necessary for maintenance and development are getting out of hand. Just as in 1920, when human operators weren’t enough to work with the switchboards, the solution is automation [24]. Autonomic computing tries to simulate the behavior of the human body, in which the autonomic nervous system takes care unconsciously of reflexes, the digestion, rate and depth of the respiration and other such processes. The term autonomic computing was first used by IBM in 2001 to describe computing systems that are self-managing, just as the human body auto-manages itself without us even thinking about it. Huebscher et al. present of review of the autonomic computing domain, and approaches that have been taken in this domain. Self-management is described as being implemented with the help of utilities for estimation of system’s behavior, through reinforcement learning or Bayesian techniques.

Carzaniga et al. propose in [25] the idea of automatic workarounds as a self-healing method for software automation. Giving a failure event, the system can automatically execute one or more alternative sequences that are known to have an equivalent behavior. A method of self-healing is described by giving formalization to the workaround problem, and finding solutions for it. A general architecture for automatic workarounds is proposed, and a mean of workarounds representation and run-time usage is described.

In [26] a solution for an integrated datacenter using power management is presented, employing server management tools, appropriate sensors and monitors and an agent based approach for achieving power and performance objectives. By intelligently turning off servers when the workload is small, there can be achieved over 25% power savings over the unmanaged case without incurring SLA penalties. By using a testbed containing IBM BladeCenter

So far, proposals have been made for self-healing computing systems, but without an actual proof-of-concept tool for creating self-\* software. To overcome this problem, IBM has developed a demo toolkit which addresses the problem of self-healing and self-optimizing software [27]. IBM Emerging Technologies Toolkit (ETTK) explores these two key aspects of autonomic computing both in a visual and in a practical way. It demonstrates that we already have almost all the necessary knowledge for creating such tools and that this technology could be a reality today.

## SLA Negotiation

Katsuhide Fujita et al. propose in [30] a Distributed Mediator Protocol for securely finding Pareto optimal solutions. They employ “approximated fairness”, using deviation for measuring the difference of utilities achieved by agents and a Nash bargaining solution maximizing the product of each agent’s utilities in our model. By defining a non-linear utility function in a complex utility space for multi-issue negotiation the estimation of agents’ satisfaction is quantified.

In [31], Chen et al. describe an automated negotiation mechanism which combines a game theory approach and a co-evolutionary approach for finding a Pareto optimal solution. In spite of the fact that the agents don’t know other agents’ strategies and payoffs, solutions that complies Nash equilibrium and Pareto efficiency concepts are discovered. The proposed method adopts genetic algorithms to optimize agent’s negotiation strategies generating payoff matrices, and after that finding the optimized point through a game-theory approach.

Brandic et al. describe in [32] a way of establishing adaptable, versatile and dynamic services by undertaking a service mediation process, considering negotiation bootstrapping. This work is being achieved in the context of Foundation of Self-Governing ICT Infrastructures (FoSII) project. They approach the gap between QoS methods and Grid/Cloud services through describing an architecture for service management, with components for meta-negotiations and SLA-mappings. The negotiation is being described in a document, having pre-requisites with negotiation terms and authentication data, and an agreement which concludes the negotiation, therefore having the term meta-negotiation.

The trust negotiation Strategy based on Negotiation Petri Net (SNPN) is described in [33], modeling the policies participating a trust negotiation. In automated trust negotiation, the access to a resource is given based on attributes rather than identification, policies participating negotiation being modeled as Negotiation Petri Nets in this paper. The SNPN is a complete negotiation strategy, having constructed both a Negotiation Petri Net and a Reverse Negotiation Petri Net and thereby being a good strategy for automated trust negotiation. Credential exchanges in this solution are being considered until both parties know that there exists a successful negotiation, and at the end only credentials in the safe disclosure sequence are disclosed, this way avoiding the disclosure of any credentials that are not needed for successful negotiation.

Multi-issue negotiation is an important problem in everyday life, existing numerous situations in which, for example, not only the price is to be negotiated, but also quantity, delivery time or other issues. For multi-issue negotiation the solution space is n-dimensional instead of the normal one-dimensional space which single-issue negotiation relies on. Most of the existing work has been focused on independent issues on multi-issue negotiation, and on sequential negotiation for multi-issue problems. In the followings, there are presented several solutions to these problems.

Guoming Lai et al. present in [34] a decentralized model for multi-attribute negotiation, being able to be applied to the situations where agents are self-interested. Agents can negotiate multiple attributes simultaneously, and always consider the Pareto optimal outcome. The model allows a proposer to make several offers in each round. The responder can choose an offer, or can reject all of them, exchange roles with the negotiator and proceed to make offers to the former negotiator. This model can be applied to incomplete information scenarios, where the agents have information only about themselves and not about other agents. Despite the fact that they don’t know the others’ agents utility functions or negotiation strategies, the generalization property is essential in this model, it being applied successfully in systems with complex utility functions in continuous negotiation domains.

In [35] an analysis of feasible solutions for multi-issue negotiation is presented. Fatima et al. present a comparison between a Package Deal Procedure (PDP) and a Simultaneous Procedure (SP). They define -Nash equilibrium strategies for both the PDP and SP, adding rules for reaching an outcome that maximizes social welfare for the simultaneous procedure since it results in multiple equilibria. From comparing the PDP and SP with the formulated strategies, it results that package deal procedure always generates optimal outcomes, while the simultaneous procedure may be better for one of the two agents, and also increases social welfare. In [36], they consider the case where there are time constraints, uncertainty about the opponent’s negotiation parameters and interdependence between issues, and compare several approaches for the considered negotiation strategies, in terms of time of agreement, time to compute equilibrium, Pareto optimality and uniqueness of the equilibrium. This paper is thoroughly documented giving also a review on possible negotiation types, and presents several theorems and conclusions regarding package deal procedure, simultaneous procedure and sequential procedure.

# Analysis and design

This part describes the algorithms used for handling SLA negotiation and optimal task distribution in a datacenter having as components servers, a storage server and a global loop which coordinates the entire task flow.



Figure .: GAMES datacenter infrastructure

The server cluster contains several servers, on which we can operate actions like move virtual machine, deploy virtual machine or remove a virtual machine from specific server in the cluster to another one. The environment is also controlled through an algorithm providing self-healing capabilities, which enforces some temperature and humidity policies on the datacenter environment. I will describe the concepts involved in the design of the self-adapting datacenter in the followings.

## The Self-Healing Behavior

The self-healing behavior described in [2] finds an action plan for bringing the context to a healthy state. Enforcing these series of actions on the broken context brings it to an optimal state. There are several concepts defined in this section which ease the algorithm description.

The context is described under a <R, A, P> (Resources, Actors and Policies) form, the entire description being kept in an ontology. The Resources refer to both sensors and actuators associated to those sensors, for applying actions which will modify resource’s value. The actors are in this case the actions associated to each resource, while Policies are conjunctions of sensor’s states, which describe the optimal state for that sensors’ association.

For evaluating the closeness of the context to the optimal context described through the policies, the entropy concept is being defined. When the entropy is above the defined threshold, the self-healing algorithm will start.

### Algorithm Description

Our approach for the self-healing process consists of the following steps (Figure 4.2):

1. **Check if the current context has been encountered so far.**

If in the memory file we have a context equal to the current context, we take add the associated sequence of actions to the current sequence of action (at the beginning, NULL). If after the sum of ***recheck times*** for the actions, the system is not healed we ***rollback*** the sequence of actions and go to step 2.

If we don’t find any sequence learned, we go to step 2.

1. **Select the broken policy with the largest entropy contribution**

Search through the policies having entropy contribution larger than zero, and select the ***policy with the largest contribution to the entropy*** (this means that it is either the most important, or the most broken, or a combination between them). Go to step 3.

1. **Take the next Inter-Independent Resources Group (IIRG)**

Select the next IIRG for which we find a best sequence of actions through a reinforcement learning process, taking as reward a function inversely proportional to the entropy. Go to step 4.

1. **Add the found sequence of actions to the actions sequence list**

We take the found sequence of actions and add it to the sequence of actions list. If there are no more resources to be brought to their optimal states for this policy, go to step 5, otherwise go to step 3.

1. **Add the found actions to the current sequence of actions**

Take the list of sequences of actions found for the current policy and add it to the current action sequence associated to the entire context. If the entropy is still larger than the threshold, go to step 1, otherwise go to step 6.

1. **Memorize the (sequence of actions, Step1 context) association**

Associate the current list of sequences of actions to the initial broken context.

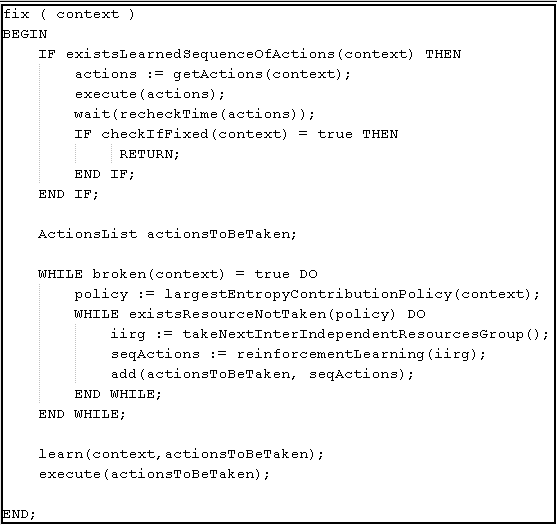


Figure 4.2: Self-healing algorithm

If while searching for the sequence of actions which would bring the context to an acceptable one, there appears a new broken policy, with a contribution to the entropy much larger than the one that we are fixing, it goes first in the queue with broken policies, the current process of healing the policy which is supposed to have had the greatest contribution to the entropy is suspended and the process of finding the sequence of actions is being started for the new policy.

The self-healing process is described as a sequence of steps, emphasizing on few concepts which are the baseline of this algorithm.

### Recheck Time

The recheck time is a physical property which defines each resource. It describes the needed time for the consequences of an action to be spotted on the current action. For example, we need some time for the temperature to be lowered after we have started the air conditioning. In this context, we have the temperature sensor with the role of sensor, the air conditioning unit with the role of actuator and the time that we need to wait with the role of recheck time.

Recheck time is a constant associated with each resource, referring to the time needed for an action to have the expected effect. If a resource will have a number of actions to be completed, the recheck moment will be after (nrActions+1)\* recheckTime. The default for the recheck time is zero, being valid for resources on which actions have an immediate effect.

### Rollback

After we enforce the found sequence of actions list on the current context, we check they produced the expected effect. If this is not the case, each action will be rewinded in the reverse order with regard to the order in which they were taken. Then the system continues by searching for a sequence of actions that will have the desired effect.

### Inter-Independent Resources Group

#### Dependency between Resources

The dependency of a resource to another resource is visible when the actions taken upon a resource affect the other resource to. The dependency relation can be seen as a dependency function:

0, if no path exists between i and j

, if k is the length of the shortest path from resource i to resource j

The dependency is a **transitive** relation:

(4.1)

Therefore if resource i is dependent on resource j, and resource j is dependent of resource k, resource i will be dependent of resource k. For example, in a smart laboratory context, if the air conditioning affects the temperature, and the temperature affects the humidity, the temperature will indirectly affect the humidity, so the resource humidity is dependent on air conditioning with a degree two of dependency.

There are two dependency types: direct dependency and indirect dependency. Direct dependency means that a change of the first resource would lead to a change in the other resource. On the other hand, indirect dependency describes a sequence of resources in which each resource depends on the next resource, which means that the first resource indirectly depends on the last resource.

(4.2)

The *dependency graph* is a direct weighted graph, having as nodes the resources and as arcs dependency relations and as weights the dependency degrees. The *dependency degree* between the resources is the inverse of the length of the shortest path between the nodes.

#### Inter-Independent Resources Group

Two resources are *inter-independent* with each other iff D(i,)0D(j,i)0. That means that doesn’t depend on and j doesn’t depend on i.

The *Inter-Independent Resources Group* is a set S of resources for which the following properties are true:

* S,
* S,–resource in current policy such that weight( weight(
* The set S is maximal.

In other words, any resources belonging to a current IIRG are inter-independent, have the highest weights and form a maximal set.

!!!!!!!!!!!Check daca vreau sa adaug iirg extraction

### Mathematical Description of the Context

Each resource and each policy in our context have an associated *weight* describing the importance of the resource or policy in the current context, in relation with the other existing resources or policies.The policy weights are represented as a (1, n) matrix, which has on each i-th row the weight of the policy i.

The weight of a policy reflects how much we need for it to be respected:

W = . W- Weight Vector

. – weight of the policy i

On the other hand, resources importance in a policy depends both on the resource and on the policy. This is why, for resources weights we have a two-dimensional matrix:

…

…

…

R = … … … … … R-the resource importance matrix

… … … … … R [i,j] = the importance of resource i in policy j

… … … … …

…

In the context of a policy, each resource will have a value which indicates how much the resource respects that policy. The value of a resource is formalized as follows:

…

…

V= … … … … … V-the matrix of values

… … … … … V [i,j]=the value of resource j in policy i

… … … … …

…

One element of this matrix, V [ij], will be a function dependent on resource j and policy i, evaluating how much does the resource j respect the policy i. This function will be defined as:

v:



0, rj has discrete states, state(rj)=desiredstate(rij)

v (i,j) k ,rj has discrete states,state(rj)!= desiredstate(rij)

|state(rj)-desiredstate(rij)| , otherwise

Having these matrices, the state of the context at a given time is defined by P\*R\*V.

#### Entropy

According to Wikipedia, “the entropy is a measure of how disorganized a system is” [5]. Extrapolating on the second laws of physics, the entropy has become an important concept in cosmology and even everyday life. As the universe is supposed to be a closed system, by the Second Law of Thermodynamics, its total entropy is constantly increasing. This has lead to speculations that the universe is fated to a heat death. In the study “Natural selection for least action” [37] Ville Kaila and Arto Anilla describe how the second law of thermodynamics can be written as an equation of motion to describe evolution, showing how natural selection and the principle of least action can be connected, evolution exploring possible paths to level differences in energy density and increasing entropy most rapidly.

Extrapolating on the problem of thermodynamics and entropy, the same concept is defined for our environment. The entropy is the measure for how disorganized our system is. It describes the way in which the environment policies are fulfilled. For our context described above, the entropy will be:

E= ∑ \*. (4.3)

Where:

* **-** the weight of i-th policy
* - the importance of resource j in policy i
* **-** the value of resource j in policy i.

The entropy can also be expressed as: E=∑P\*R\*V (4.4). In this relation, the product P\*V describes the respectability degree of the policy, while the product P\*R gives the total weight of a resource within the environment, considering all the described policies.

#### Policy Contribution to the Entropy

The entropy contribution of a policy to the entire system entropy describes the way in which the incomplete fulfillment of a policy affects the entire context.

The policy contribution to the entropy will be defined as:

=\*(4.5)

Where represents the contribution to the entropy of the ith policy, is the weight of the policy i, and\* is the respectability degree of the policy i in the current context.

#### Entropy Threshold

For describing an acceptance limit for the context, we introduce the concept of *entropy threshold.* This threshold is a numeric value describing a context which isn’t acceptable anymore, and therefore we need to start the self-healing algorithm. The self-healing algorithm will start each time the entropy will pass the entropy threshold.

Depending on how strict we are about the way in which the system should behave, we have a few possibilities for deciding when the system should consider itself broken or “disturbed”. There are several variants:

* At its stricter version, the system is only healthy when all the policies are perfectly respected. This is the case when T=0.
* For a more complex system, we can attach to each policy a value which denotes how much of it should be respected, and compute the threshold as :
  + - (4.6), where is the minimum achievable entropy
* The threshold can be dependent also on a weighted average of the policy contribution to the entropy, and add it to the minimal entropy. This solution would also improve computational time and assure that the system can go out of its way as long as an important policy is not broken
  + - (4.7)

### Algorithm Complexity

The algorithm complexity is a breath first search complexity, therefore having a worst case complexity of, where b is the branching factor and d is the depth factor. The branching factor is described by the following relation:

(4.8)

Where defines the number of resources existent in the context, defines the maximum number of actions associated to each resource while is the branching level at a moment of time. From each branch level, we are recursively defining more and more branching points, and therefore the branching factor grows with the reached.

The depth factor is:

(4.9)

Where defines the number of actions associated to each resources. The depth factor is described by the total number of actions associated to each resource. The context is guaranteed to reach a solution before taking all the possible actions, each action being taken a maximum number of times defined by.

The above branching and depth factors describe the worse case situation, which will rarely appear due to the fact that policies and resources are weighted, and there exists an entropy variable describing the degree of disturbance in the system, and the way in which the system progresses throughout the breath first search path.

## Datacenter Self-Adapting Behavior

An autonomic datacenter which can lower the amount of energy consumed without disregarding the service level agreement needs to adapt itself to all kinds of new situations, to find solutions to new problems and keep them in mind for the future.

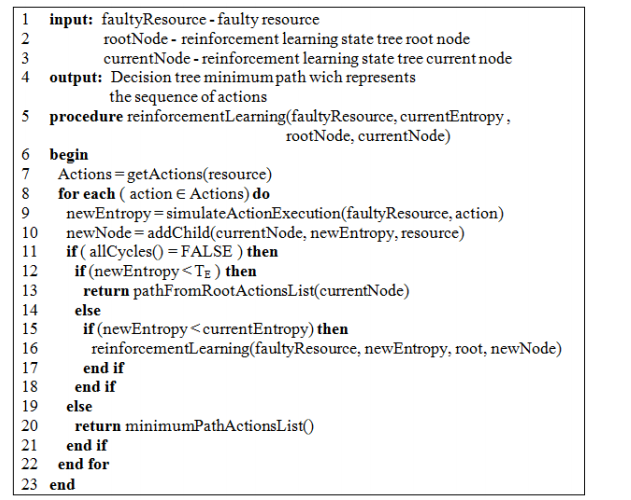


Figure .: Self-adapting Behavior

The self-adapting behavior is implemented through a reinforcement learning approach. The algorithm used for enforcing the environment self-healing behavior is adapted for creating the self-adapting the self-healing behavior of the datacenter.

At each step, the algorithm simulates the execution of each of the possible actions, and evaluates the expected reward and entropy for the resulting states. From all the resulting states, and the states that haven’t previously been expanded, the system chooses the one with the largest reward. From this new current state, the system is retrying all the possible actions and expanding them.

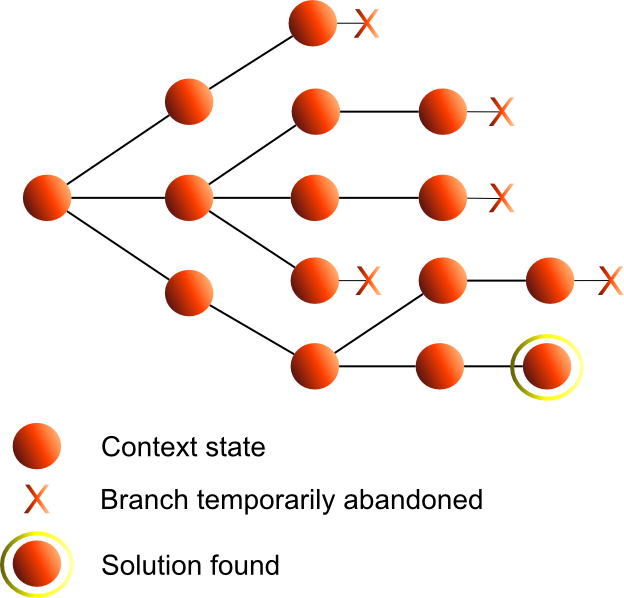


Figure 4.4: Reinforcement Learning Algorithm- States with small rewards get expanded later

By always expanding the leaf with the largest reward (figure 4.3), the system is guaranteed to find the best sequence of actions which would bring the system as close as possible to the 0-entropy state.

### Datacenter Context Mapping

Any autonomy can happen only when the system is aware of what is happening around it. This is why, before being a self-adapting system it needs to have the form of a context-aware model, and be able to understand and reason through this information. The model presented in [4] is being used for creating the context-aware part of the system. In order for us to be able to use this model, we need to map the datacenter elements on the context-aware model.

The <R, A, P> model describes a context-aware system, having resources, actors and policies which interact with each-other. Resources are described as physical characteristics of the environment or physical objects with which we can interact. The actors interact with resources and can change the entire context, while policies describe the best state for the context.

Table 4.1: Mapping between <R,A,P> and datacenter context

|  |  |  |
| --- | --- | --- |
| <R,A,P> entity | Self-Adapting context entity | Details |
| Resources | **Servers** | * All datacenter servers |
| Actors | **Actions** | * Deploy task * Move task * Send server to hibernate * Wake up server |
| Policies | **Energy policies** | * Green performance indicators |
| **QoS policies** | * Key performance indicators |

Therefore, as shown in the Table 4.1, in our datacenter the servers will play the role of resources, the actions applied on those servers will play the role of actors and the QoS and energy policies will be the policies of the context-aware system.

### Entropy

The entropy describes the degree of disturbance in the environment. For the green datacenter context, the entropy reflects both the happiness of the customer, and the happiness of the environment.

The entropy needs to be held as its lowest value, and therefore the definition of the function describing the entropy is extremely important. Since we have defined policies both for Key Performance Indicators (KPI) and Green Performance Indicators (GPI), the entropy will evaluate the respectability degree of each policy.

(4.10)

Where:

* + - represents the weight of the policy (QoS or energy policy)
    - is the weight of the system resource i in the policy j. The resource weight reflects the system resource importance for the policy. If a policy imposes no restrictions to a system resource then the resource weight of the resource for that policy is zero.
    - is the deviation between the current value of the resource, and the accepted range for that resource

For the deviation above between the desired resource value and the existent one, if the optimum values have the form of a range, there exists a deviation when current value is either smaller than the lower limit of the range, or larger than the upper limit of the range.

### Reward in Reinforcement Learning

In any reinforcement learning algorithm, the context’s state is defined by the reward the systems gets for arriving or for being in that state. In this approach, the following function is defined for describing the expected reward:

(4.11)

In the relation 4.9, defines the expected reward for the next state of the context. It is an expression of the reward of the system for getting in the current state, , and a gamma factor multiplied with the transition reward for getting from state i to state i+1. The gamma factor is called a discount factor, enabling the system to try to take the best actions in the early phase of the algorithm. and are numerical values for current entropy and respectively the future entropy, for the case in which the system takes an action with the cost . The variable describes the number of actions taken so far, and is included in the function describing the expected reward because we need to take as few actions as possible, considering that they are time consuming.

### Task History

George Santayana, a Spanish philosopher affirmed in his book [37] that “Those who cannot remember the past are condemned to repeat it.” This is why, besides remembering what actions we have found for bringing the context to an acceptable state, we also need to store the type of tasks that we have received in the datacenter so far, and try to reason on this information.

The obvious reasoning on this information would be a load prediction, for us to know approximately how many servers and what type of servers we need for a time period in the future. The task history component would ease a lot the time wasted on waiting for servers to start, because knowing what will come next we can anticipate the workload and take the right attitude when considering it.

### Tail Server

Using the history component, we are able to predict the task flow and therefore we will be able to wake up servers when anticipating many tasks arriving to the datacenter, and there are not enough tasks available.

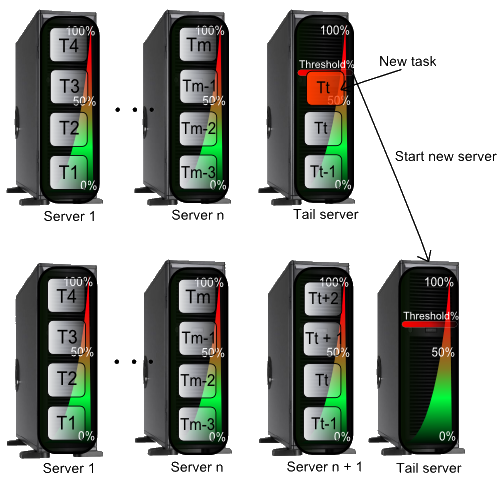


Figure 4.5: Tail Server

Also, when the workload of the tail server grows above its defined threshold, we need to wake up a new server (figure 4.4). The decision on the type of the server to be woken up will be made depending on the task flow predicted by the history component.

### Algorithm Complexity

For estimating the complexity of the algorithm, we need to find the branching factor and the depth factor, knowing that the complexity of a depth first search algorithm is , and that the reinforcement learning algorithm is based on a depth first generation of nodes.

The branching factor is equal to the number of actions that could bring the context into a new state. The number of wake up actions is equal to the number of sleeping servers, the number of sleep actions is equal to the number of online servers, the number of deploy actions is equal to the number of undeployed tasks multiplied with the number of available servers, while the number of move actions is equal to the number of tasks multiplied with the number of available servers.

(4.12)

The depth factor will also be equal to the number of actions, since by doing all the actions available it is impossible not to find the solution.

(4.13)

The worst case complexity of the algorithm will be:

(4.14)

Even though the complexity seems considerable, the worst case situation will rarely be reached due to the reward function which enables the algorithm to rapidly converge to a solution, intelligently, without searching blindly through the entire space.

## Sla Negotiation between Client and Datacenter

Negotiation is a dialogue intended to resolve disputes, to produce an agreement upon courses of action, to bargain for individual or collective advantage, or to craft outcomes to satisfy various interests [5].

In service centers, the negotiation is an essential part since a client will always ask for the most resources possible for the money that he pays. Therefore the system will always get to a situation where the new task won’t fit in any of the existent servers. In these situations, a negotiation process needs to be undertaken in order to lower the requirements of the client in an admissible range, and be able to deploy the task on a server.

Negotiation in this system needs to have under consideration multiple issues, because we have multiple resources which define a server or some task requirements. It also needs to be taken between the entire datacenter and the client represented by the task. There are several possibilities for doing that:

* **“Choose first” solution**

Consider the server which is the closest to the imaginary one described by task requirements, and negotiate between the task and the found server. This solution doesn’t guarantee that there exists a solution to that negotiation and we don’t need to take the next closest server.

* **“Choose last” solution**

Create a possible server from existent servers in the datacenter by taking extremities of the admissible resource ranges, and negotiate between this server and the task. After a solution is found to the negotiation, the returned server will be the one closest to the negotiation solution.

* **Sequential elimination solution**

For a better accuracy, we can sequentially reduce the available servers, after having done negotiations on each resource. This ensures a solution closer to the optimal. The disadvantage of this solution is that we need to make evaluations after each negotiation in order to get the remained servers.

For the first two solutions, we need a mean of evaluating the distance between two servers. The distance between the two servers is visually described bellow, in figure 4.5.

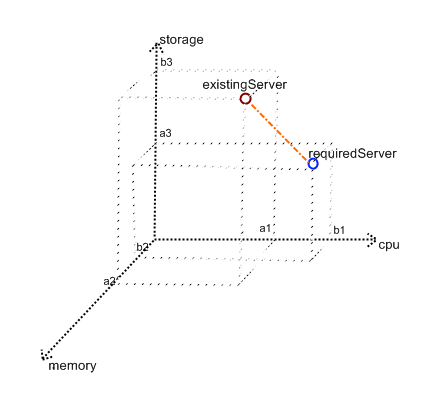


Figure 4.6: Distance between two servers

The server is defined by CPU, memory, storage and other resources. We will denote them a1, a2, a3, and we will have for the existing server the defining list of resources (a1,a2,a3,…) and for the required server the list (b1 ,b2 ,b3,…). For finding the server which differs the least from the required server, we need the server having a minimum distance to the required server (figure 4.5). For this, we can use Gaussian (4.13) Manhattan (4.14) or Chebyshev(4.15) distances :

(4.15)

(4.16)

(4.17)

In this project, there are two approaches for negotiating: with the help of fuzzy logic or through finding the Nash equilibrium.

There are several possibilities for negotiating multiple issues at once:

* **Considering the issues independent one of another**

This solution is applicable to situations where the issues are not interdependent with each other. Each issue is negotiated in the same time, and after that the result is formed of all the results of the negotiations. This is a fast process, but the negotiation result is guaranteed to function only for independent issues.

* **Sequential Negotiation**

This solution is applicable to any situations, since negotiation is made for each issue. The result of each negotiation will be the input for the negotiation for the next issue. After negotiating all issues, the result will be the output of the last negotiation. The issues need to be weighted, because we need to know which the priority for each resource, considering that we have to negotiate for a single issue at a time.

* **Package Deal Procedure**

The package deal procedure is the one giving the optimal negotiation solution. Issues are negotiated as a package, and offers are made in terms of collection of interrelated issues. The offer is being made as a proposal for each issue under negotiation. Agents are allowed to either accept an offer, or reject all of it. Despite the fact that the Package Deal Procedure (PDP) gives always the optimal solution, the time spent in finding the solution is the highest from the three described.

### Fuzzy Logic Negotiation

Fuzzy logic is a form of multi-valued logic derived from fuzzy set theory, to deal with reasoning that is approximate rather than precise [5]. For negotiating, we define two ranges: one is the range requested by the client, another is the range needed by the server to have optimum power consumption together with their satisfaction value represented by the Y axis. Through defuzzification by center of gravity we get a value for the current resource which is acceptable both for the client and the datacenter.

### Nash Negotiation

John Forbes Nash put the basis of the domain called game theory. He proposed a solution concept of a game involving two or more players, in which each player is assumed to know the equilibrium strategies of the other players and no player has anything to gain by changing their options unilaterally. If each player has chosen a strategy and no player can benefit by changing his or her strategy while other players keep theirs unchanged, then the current set of strategy choices constitutes a Nash equilibrium [5].

Nash Theorem revolutionized and basically put the foundation of Game Theory. If we see the current negotiation like a cooperation game between the energy and the QoS, we can also see the Nash Equilibrium as the point in which both the client and the data center are satisfied. There are two kinds of approaches for Nash negotiation: the straight forward technique and the bargaining technique. The two of them will be described as follows.

#### Straight Forward Nash Negotiation

For a straight forward Nash negotiation, we build a matrix having as elements pairs of utility values for the task and respectively for the datacenter, for one resource. The utility functions are:

(4.18)

(4.19)

(4.20)

(4.21)

The negotiated point will be the point with both the maximum utility of the task with report to the value of the server’s attribute and the maximum utility of the server with report to the value of the task’s attribute.

For each resource (CPU, memory, storage) we find the equilibrium value, while we have as ranges the extremity points of what we have in the entire data center (ex. If we have two servers with a CPU desired range of (400, 2000) and one of (200, 1600), the range for the datacenter will be (200, 2000). After finding the equilibrium points for all resources, we take the server which is the closest to the found values.

#### Nash Bargaining Solutions

Another approach to finding Nash equilibrium involves two agents, and offers made from one agent to another. This is also called a **bargaining** process. In Nash bargaining game each of the two players demand a part of some good, and the two players get their demand if the two proposals sum to no more than the total good, otherwise both players getting nothing.

The Nash bargaining solution is guaranteed to reach a Parento optimal solution. In Nash bargaining solution each player offers a deal based on backward reasoning (what he expects from the other to offer at the next move). At each step if the first player refuses the offer made by the second player, a new offer will be made from the second player towards the first player. In this state, their roles will interchange. By this technique and by defining acceptable offers at each step, the negotiation is expected to rapidly converge to a Pareto Optimal Solution.

### Dynamic Policy Concept

After negotiating, the policies for datacenter need to be modified in order to fit the negotiated values. This involves the existence of dynamic policies which will reset to the old values when a new task appears in the task queue.

## Isomorphism of Self-\*

By defining a context-aware model described by <R, A, P> (Resources, Actors, Policies). This model generalizes any environments, from rooms to datacenters and therefore we can use on it the same algorithm searching for actions to enable for self-\* behavior. Therefore, the reinforcement learning algorithm with or without improvements can be used for self-healing and self-adapting capabilities implementation. From this we can generalize and say that by applying reinforcement learning on the <R, A, P> model we can implement any kind of self-\* behavior.

# Design and Implementation

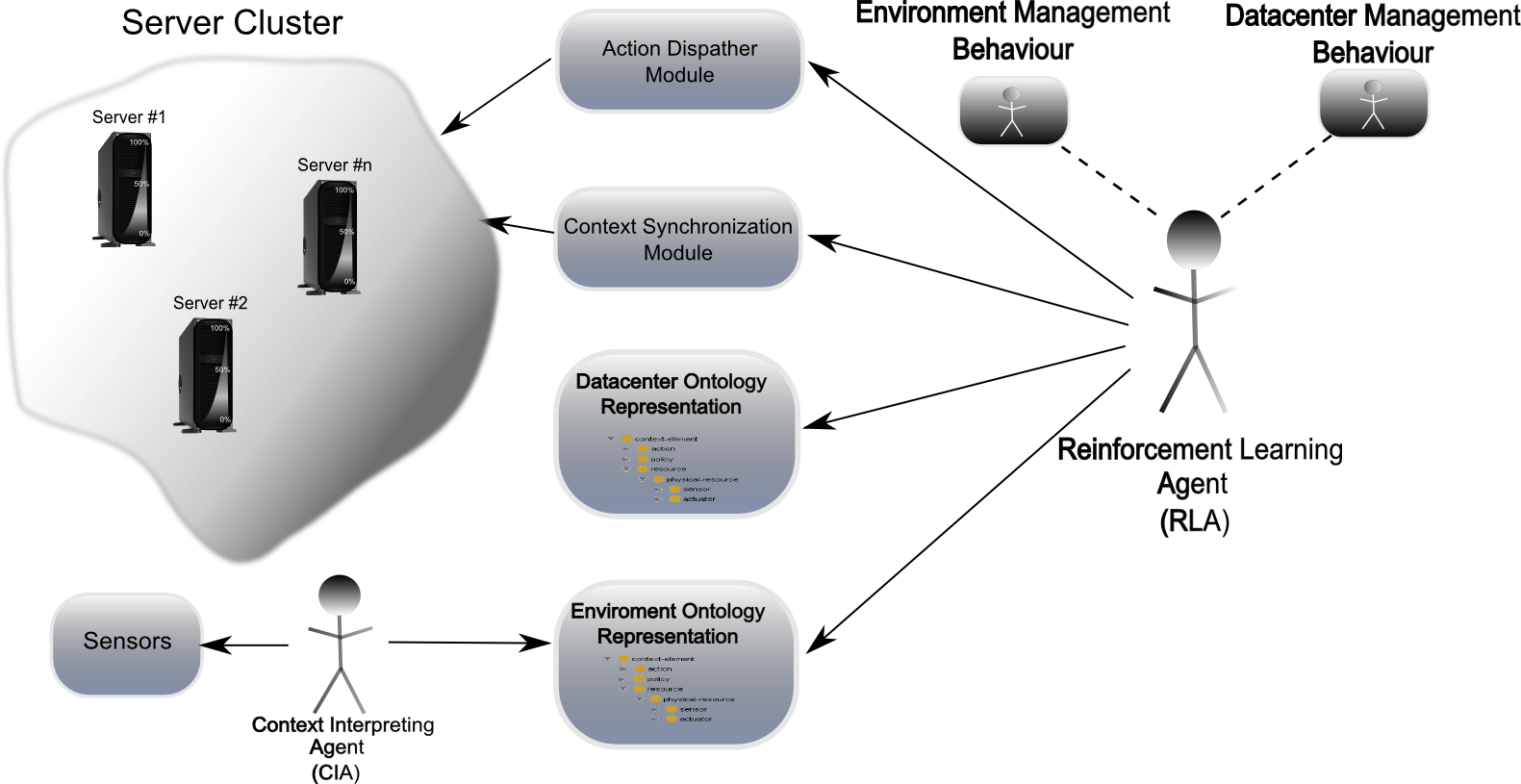


Figure .: View on the System

## Conceptual architecture

# References

[1] **U.S. Environmental Protection Agency, ENERGY STAR Program**, “Report to Congress on Server and Data Center Energy Efficiency”, Public Law 109-431, 2 august 2007

[2] **Tudor Cioară, Ionuţ Anghel, Ioan Salomie**

[3] **Ionuţ Anghel, Tudor Cioară, Ioan Salomie, Mihaela Dînşoreanu, *Georgiana Copil*, Daniel Moldovan**, “A Reinforcement Learning based Self-healing Algorithm for Managing Context Adaptation”, 8th Annual IEEE Int. Conf. on Perv. Comp. and Communications, 2010.

[4] **Tudor Cioară, Ionuţ Anghel, Ioan Salomie, Mihaela Dînşoreanu, *Georgiana Copil*, Daniel Moldovan**, “A Self-Adapting Algorithm for Context Adaptation”, !!!! roedunet?

[5]**Wikipedia**, Web Site: [www.en.wikipedia.org](http://www.en.wikipedia.org)

[6] **James Carter**, “**A look inside IBM's green data center research**”, Proceedings of the 14th ACM/IEEE international symposium on Low power electronics and design.

[7] **VMWare Team**, “A performance Comparison of Hypervisors”, Jan 31, 2007

[8] **Hyper-V WMI Provider**, Web Site: <http://msdn.microsoft.com/en-us/library/cc136992%28v=VS.85%29.aspx>

[9] **IBM manifesto,** Web Site:<http://www.research.ibm.com/autonomic/manifesto/>

[10] **Jeffery Kephart, David Chess**,”The vision of autonomic computing”,Computer Magazine, IEEE, 2003

[11] **Thomas Strang, Claudia Linnhoff-Popien**, “A context modeling survey”, Workshop on Advanced Context Modelling, Reasoning and Management, UbiComp 2004 - The Sixth International Conference on Ubiquitous Computing, Nottingham/England

[12] **M. Weiser**, “The Computer for the 21st Century,” in Scientific American, 1991.

[13] **MIT's Intelligent Room**, Web Site: [www.ai.mit.edu/projects/iroom](http://www.ai.mit.edu/projects/iroom)

[14] **Stanford’s iRoom project**, Web Site: <http://graphics.stanford.edu/projects/iwork>

[15] **Jason I. Hong, James A. Landay**, “An Architecture for Privacy-Sensitivity Ubiquitous Computing”, International Conference On Mobile Systems, Applications And Services,  
Proceedings of the 2nd international conference on Mobile systems, applications, and services

[16] **Paul Prekop, Mark Burnett,** “Activities, Context and Ubiquitous Computing”, Computer Communications, Volume 26, Number 11, 1 July 2003 , pp. 1168-1176(9)

[17] **Harry Chen, Tim Finin, Anupam Joshi,** “An Ontology for Context-Aware Pervasive Computing Environments”, The Knowledge Engineering Review, Volume 18 ,  Issue 3  (September 2003)

[18] **Lalana Kagal, Tim Finin, Anupam Joshi,** “A Policy Language for a Pervasive Computing Environment”, Proceedings of the 4th IEEE International Workshop on Policies for Distributed Systems and Networks

[19] **Damien Borgetto, Patricia Stolf,** “Energy Aware Autonomic Manager”

[20] **Josep Lluís Berral, Íñigo Goiri,Ramon Nou,Ferran Julià, Jordi Guitart, RicardGavaldà, Jordi Torres, “Towards energy-aware scheduling in data center using machine learning“,** 1st Int’l Conference on Energy-Efficient Computing and Networking (e-Energy 2010).  University of Passau, Germany. April 13-15, 2010.

[21] **Chandrakant Patel, Ratnesh Sharma, Cullen Bash, Sven Graupner, “**Energy Aware Grid: Global Workload Placement based on Energy Efficiency”, HP Technical Report HPL-2002-329

[22] **T.Srinivasan, J.B.Siddharth Jonathan, Jayesh Seshadri and Arvind Chandrasekhar**, “A Comprehensive Architectural Framework for Task Management in Scalable Computational Grids”- Annual National level technical symposium conducted by the Computer Science and Engineering Association of the Department of Computer Science and Engineering (DCSE), College of Engineering, Anna University, March 14-15,2005

[23] **Alexandre Mello Ferreira, Kyriakos Kritikos, Barbara Pernici**, “Energy-Aware Design of Service-Based Applications Source”, Proceedings of the 7th International Joint Conference on Service-Oriented Computing, 2009

[24] **Markus Huebscher, Julie McCann,** “A survey of Autonomic Computing – Degree, Models and Applications”, ACM Computing Survey, 40, 3, Article 7, August 2008

[25] **Antonia Carzaniaga, Alessandra Gorla, Mauro Pezzè,** “Self-Healing by Means of Automatic Workarounds”, Proceedings of the 2008 international workshop on Software engineering for adaptive and self-managing systems

[26] **Rajarshi Das, Jeffery Kephart, Charles Lefurgy,** “Autonomic Multi-Agent Management of Power and Performance in Data Centers”, Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems: industrial track, 2008

[27] **IBM ETTK**, Web Site: <http://www.ibm.com/developerworks/webservices/library/ws-heal/>

[28] **Mazeiar Salehie, Ladan Tahvildari,** “Self-Adaptive Software: Landscape and Research Challenges”, ACM Transactions on Autonomous and Adaptive Systems (TAAS), May 2009

[29] **Alexandre Mello Ferreira, Kyriakos Kritikos, Barbara Pernici,**“Energy-aware Design of Service Based Applications”, Proceedings of the 7th International Joint Conference on Service-Oriented Computing, 2009

[30] **Katsuhide Fujita, Takayuki Ito, Mark Klein,** “Finding Nash Bargaining Solutions for Multi-issue Negotiations: A Preliminary Result”, Proceedings of the 1st International Working Conference on Human Factors and Computational Models in Negotiation, 2008

[31] **Jen-Hsiang Chen, Kuo-Ming Chao, Nick Godwin, Colin Reeves, Peter Smith,** “An Automated Negotiation Mechanism Based on Co-Evolution and Game Theory”, Proceedings of the 2002 ACM symposium on Applied computing, 2002

[32] **Ivona Brandic, Dejan Music, Schahram Dustdar**, “Service Mediation and Negotiation Bootstrapping as First Achievements Towards Self Adaptable Grid and Cloud Services”, Proceedings of the 6th international conference industry session on Grids meets autonomic computing, 2009

[33] **Yan He, Miaoliang Zhu,** “A Complete and Efficient Strategy Based on Petri Net in Automated Trust Negotiation”, Proceedings of the 2nd international conference on Scalable information systems, 2007

[34] **Guoming Lai, Katia Sycara, Cuihong Li,** “A Decentralized Model for Multi-attribute Negotiations”, Proceedings of the 8th international conference on Electronic commerce: The new e-commerce: innovations for conquering current barriers, obstacles and limitations to conducting successful business on the internet,2006

[35] **Shaheen Fatima, Michael Woolbridge, Nicholas R. Jennings,** “An Analysis of Feasible Solutions for Multi-issue Negotiation involving Nonlinear Utility Functions”, Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 2, 2009

[36] **Shaheen Fatima, Michael Woolbridge, Nicholas R. Jennings,** “Multi-issue Negotiation with Deadlines”, Journal of Artificial Intelligence Research,  
Volume 27 ,  Issue 1  (September 2006)

[37] **Ville Kaila, Arto Annila** “Natural selection for least action.” Proceedings of The Royal Society A. 2008 PhysOrg.com.

[38]**George Santayana,** “The life of reason-The phases of human progress”, 1905-1906