Journal of Computer Science

Original Research Paper

Model for Automated and Improved Utilization of Existing Computer Resources on an Example of Web Servers

Dino Alagić and Ivan Magdalenić

Faculty of Organization and Informatics, University of Zagreb, Varazdin, Croatia

Article history
Received: 28-11-2017
Revised: 26-01-2018
Accepted: 28-02-2018

Corresponding Author:
Dino Alagić
University of Zagreb, Faculty of organization and informatics, Varazdin, Croatia
Email: dialagic@gmail.com

Abstract: Information technology is under constant innovation pressure to provide the highest level of data availability – i.e., the continuous functioning of operating systems. This is the very reason for an accelerated development of complex systems encompassed by the term cloud computing. Among other things, such solutions are aimed to ensure high-level availability of complex systems and architecture. Numerous studies indicate that the main cause of high cost of data centers are procurement and maintenance costs of servers. Furthermore, this paper presents concrete practice examples which confirm that there are systems whose resources are not sufficiently exploited, but still have to be provided due to their importance. The high costs and inadequate utilization of the existing computer resources constitute the main motivation for the present research. In this study, we present a new model for automated and improved utilization of the existing computing resources. The model is verified by using an application for automated management of computer resources that was developed for this research and by several tests conducted on Web clusters.

Keywords: Cloud Computing, Web Cluster, Web Farm, HTTP, Algorithm, Model

Introduction

In the last decade, the concept of cloud computing has been increasingly used, becoming an important part of any modern business system. Cloud computing is a type of computing which relies on sharing of computing resources starting with applications and including a variety of related services (Islam *et al*., 2012). Such systems are very complex and require an IT infrastructure – i.e., data centers for proper functioning. This is primarily caused by globalization and liberalization of markets in which it is not acceptable to have an information system without a high level of availability. However, such data centers are not cheap and in order for one such center to survive in today's market, continuous investment in the growth and improvement of the system is required. This means that capital expenditure (CAPEX) as well as operating expenses (OPEX) (Gruber, 2009; Li *et al*., 2013; Wiboonrat, 2014) will be incurred. Numerous studies confirm that servers account for the high costs of data centers (Sampson and Tullsen, 2012).

Providers of cloud computing strive for multiple use of existing resources – i.e., they wish to achieve system automation and optimization. The question addressed in our research is how to ensure multiple use of such an infrastructure or servers in order to minimize costs. More than a decade ago, this problem was approached through virtualization of computing resources by dividing the resources of a single physical device or server into several smaller virtual environments (Soundararajan and Herndon, 2014). Today, virtualization is common in almost all data centers and as such no longer presents an innovation, but a necessary standard for achieving market competitiveness. To ensure a high level of availability, data centers make use of virtualized machines which are powered on demand. In such a scenario, data centers have reserved resources such as CPUs and RAM. Our field of interest is Web clusters for small and medium-sized enterprises because they have limited resources that need to be utilized to the greatest possible extent. On the basis of our preliminary research we concluded that services have different demands on resources at different times. We believe that is reasonable to take resources from a poorly loaded server and transfer them to a server under heavy load and vice versa, if needed. This can be done and is actually done in practice by powering off virtualized machines and increasing or decreasing their resources. However, this process is time-demanding and system response is

usually measured in minutes. The process is also hard to automate because of the lack of an appropriate tool. In this study, we propose a new model for automated and improved utilization of the existing computing resources, mainly CPUs and RAM, without the need to power off virtualized machines.

Research Problem

Since cloud computing is a very broad term, we have narrowed our focus on Web servers, where computer resources are mostly inadequately utilized but are still necessary to ensure a high level of system availability. Following the development of technologies and increasingly complex user demands, the services provided through Web servers and their architecture are also becoming more complex. One example of such a solution are Web farms, which may consist of one or more Web clusters containing multiple Web servers, the architecture and design of which depend on the technology and the type of content (static or dynamic).

In addition to issues of complexity and required amount of resources of such systems, the fact that one Web farm is usually not sufficient also needs to be taken into consideration (Islam *et al*., 2012; Celesti *et al*., 2011; Teodoro *et al*., 2003; Yanmaz *et al*., 2005). Some of the reasons for this are: Compatibility (technologies and programming languages), type of content (static or dynamic), Web servers (Apache, nginx, IIS, GWS etc.), type of business (security and privacy issues), resources (dedicated or shared), performance, availability etc.

For the above reasons, it is usually necessary to have several Web farms in order to ensure higher competitiveness in the market. However, a growing number of Web farms requires a larger number of servers as well as a more complex infrastructure for their proper functioning, such as a larger space, better air-conditioning, more electricity etc. All of this eventually results in higher OPEX and CAPEX costs.

One of the main reasons for the greater complexity of the Web farm architecture model is that it strives to provide the highest possible level of system availability with maximum performance. In other words, in addition to the Web server as the main component of the system, there are many other supporting components such as: Supporting tools, system cache cluster databases etc. Consequently, such a complex architecture can be divided into the following four categories or levels:

- ***Network*** – this category also includes the load balancer for HTTP/HTTPS requests and traffic. Systems are usually installed in pairs with the aim of higher availability and are characterized by: *Types of algorithms, business category, technology types (solutions)* etc
- ***Web*** – This level entails two components: *Web farms*, that is, clusters divided by means of the HTTP/HTTPS load balancer system and *system cache*, which serves to faster access and process data. The main purpose of these systems is to save system memory in order to decrease the time necessary for accessing and processing the data
- ***Databases*** – these systems no longer represent a single database, but an entire cluster, the so-called farm database, wherein the main objective is to achieve optimal results regarding processing and system availability through a number of separate servers
- ***Supporting tools and services*** – in order for such a complex system to function properly, a number of supporting and monitoring tools are required, some of which are: *Logging, back-up, management system, safety system*, *monitoring, virtualization system* etc

To provide a better understanding of the entire solution, a system containing several Web farms with all the other components necessary for their correct operation is shown in Fig. 1.

The figure shows that proper functioning of Web farms necessitates a large number of components. Consequently, the validation of a new model for automated and improved utilization of the existing computing resources proposed in this study will be performed on an example of Web server.

As mentioned earlier, numerous analyses and surveys confirm that servers are the most expensive component of cloud computing systems (Islam *et al*., 2012). The reason for this lies in large capital and operating costs that servers entail as well as in a high level of amortization. That is why computer resources need to be used more efficiently. However, there are concrete practice examples (Web farms) which confirm that there are systems whose resources are not sufficiently exploited, but still have to be provided due to their importance. In order to validate this research topic, we conducted a preliminary study by using the data from several IT companies that provide Web hosting services. To enable us to conduct the research in its entirety, the companies secured access to all their relevant data and indicators related to Web farms, including, among others, the data concerning the infrastructure and the number of visits or user requests for individual Web sites. The values, including the number of user requests and load on computer resources, were collected by server monitoring tools (*Observium* and *Munin*). In our preliminary research four different Web farms were used as a reference test sample, encompassing between 50 and 100 Web pages. Web farms differed according to the type of content: Business, video games, adult content and information. The following chart (Fig. 2) shows the ratio of requests and resource utilization in the course of 24 h for all the four Web farms.

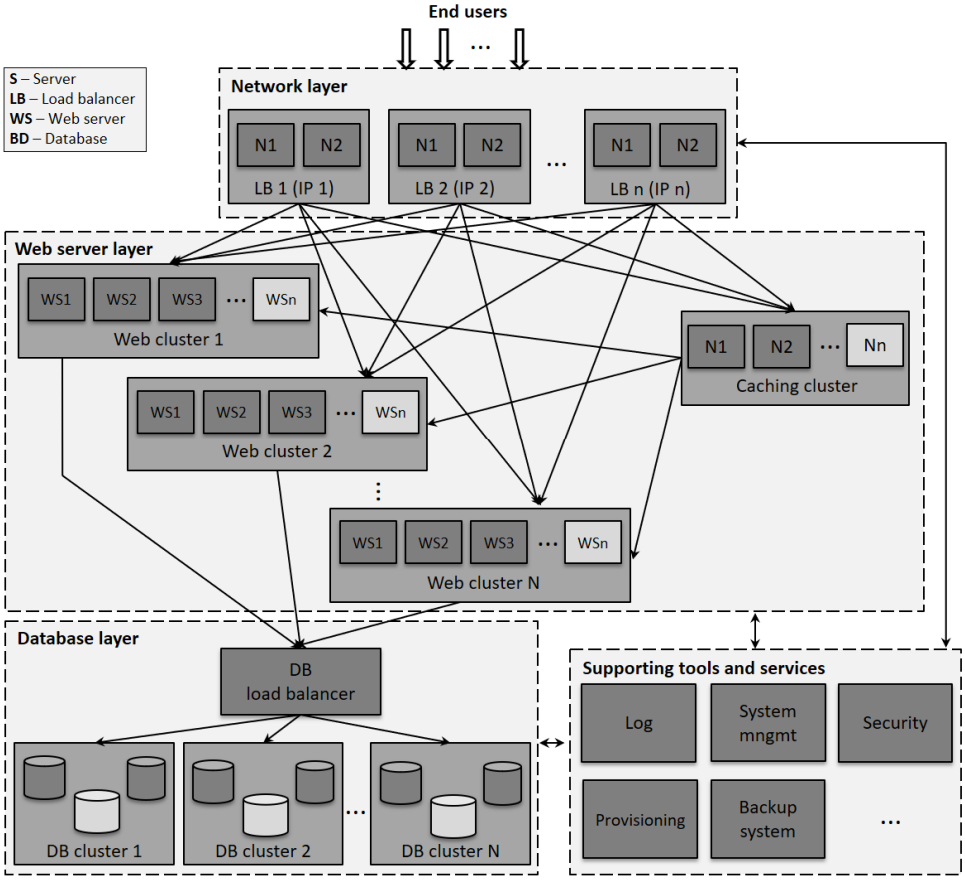

Fig. 1: Detailed schematic representation of the information flow in a Web farm environment

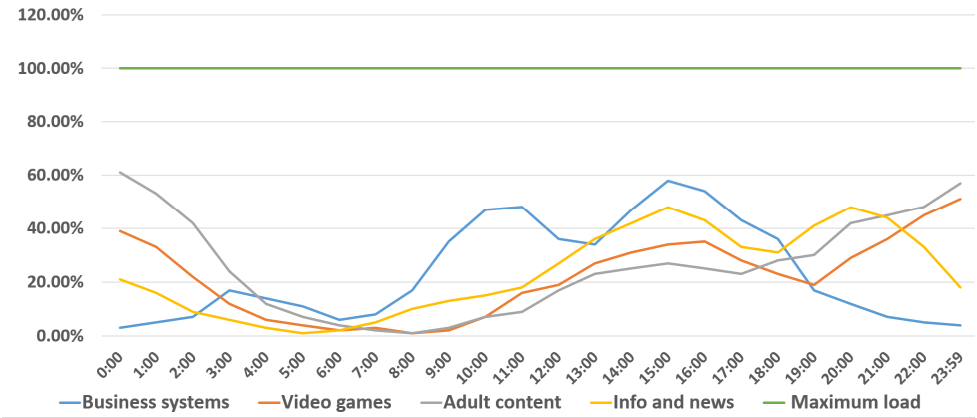

Fig. 2: System load by Web farms for 24 h

It is important to note that the results in a one-year period are very similar to the results for a 24 h period – i.e., there are no major deviations. The analysis of the results revealed that none of the Web farms was overload, in other words, that computer resources utilization did not exceed 65%, which is an argument for ensuring the capacity that will be available during peaks.

In Fig. 2 the load on computer resources is expressed in percentages. The ideal state of a server would be its 100% utilization, but in practice this is very difficult to achieve and is usually not good for the business system.

This would mean that all resources are being fully utilized and that each subsequent request is in queue for execution. In other words, any load exceeding 100%

would represent a situation in which system overload is likely to occur that may result in system shutdown, which is unacceptable in today's business. In this case, system load depends on the number of user requests, which is an external (and thus unpredictable) variable. For this reason, security mechanisms need to be implemented to prevent system overload and enable as maximum possible resources utilization not exceeding 100%. The analysis of the preliminary research results also showed that all the four Web hosts had a different number of visits within the 24 h period, with some of them requiring a greater amount of resources during the day (information and business) and others being more active at night (video games and adult content). It is therefore evident that the resource load is not always linear and each farm has extra resources for situations when an increased number of requests is being made. Consequently, the question arises of how to utilize resources that are free, that is, how to distribute or transfer them to a part of the system where they are needed. Therefore, the problem to be addressed does not concern increasing the resources or grouping Web pages within a Web portal regardless of the type of content, but allowing the existing free resources to be better utilized, which is an issue still to be resolved.

Existing Solutions and Related Works

During our investigation of scientific literature, we found a large body of research in this field proposing numerous methods, concepts and approaches as well as solutions to the problem of insufficient utilization of the existing computing resources (CPU and memory). Most studies that focus on better utilization of computing resources use a combination of their own algorithms and mechanisms for controlling Quality of Service (QoS). Such mechanisms enable prioritization of resources or services between different applications, users or data flows. However, the existing solutions focus solely on CPU resources and not on working memory, which is another important aspect of computer resources (You *et al*., 2011; Zhanjun *et al*., 2000). This issue can be resolved by using a variety of network algorithms and architectures for the distribution of traffic across the network (Pham *et al*., 2010). This approach does not represent the entire solution since resources remain assigned to specific servers and in the case of an increase in requests, they are immediately forwarded to another system, without increasing or optimizing the utilization of the existing systems (Pham *et al*., 2010). Most research on the topic of multiple utilization of computer resources has a completely different goal – achieving higher system availability so as to perform the virtual migration of servers from one physical server to another. This approach creates an increase in indirect costs (electric energy, larger number of servers, network complexity, maintenance etc.) because the main objective is only to keep the system working, without making a better use of resources (Ma *et al*., 2012; Marrone and Nardone, 2015; Ichikawa and Komoda, 2016). Similarly, this issue can be approached by a combination of various methods and algorithms that enable the scaling of the system in order to ensure its availability (Corsava and Getov, 2003; Frachtenberg *et al*., 2002). The opposite approach to scaling and increasing of resources is the grouping of virtual servers with the aim of minimizing physical servers so that the remaining (currently free) physical servers could shut down, enabling less power and other resources to be spent (Beloglazov *et al*., 2012). However, combining methods and algorithms for system scaling is not adequate, because even when they are shut down they occupy an expensive rack space in data centers. In addition to its major limitations, such an approach instigates other issues like potential unavailability of the entire system in cases of a sudden increase in the number of requests. In that case, there a sudden need for computer resources would occur that are not available at a given moment because of the time it takes the physical server to start up. This problem can be solved through prioritization of CPU resources by employing one of numerous mechanisms and algorithms that allow individual virtual servers to gain higher priority over the physical server's CPU resources. However, by establishing process execution priority, the number of processor cores still does not increase (Song *et al*., 2013; Guan *et al*., 2014). During our study of the literature, several problems related to the existing research were identified as follows:

- *Partial analyses* – most studies are not fully elaborated – i.e., they do not present any way of evaluating the proposed models and solutions. Their authors mostly only state their assumptions and introduce concepts that should solve the research problem
- *Solutions applicability* – proposed solutions are usually not applicable in practice and are developed only at the conceptual level. Solutions that *are* applicable in practice are not used widely due to various technological constraints
- *Testing repeatability* – in most research the environment in which the tests and measurements were performed are described poorly or not at all, whereas test samples are quite specific, meaning that it is not possible to do further research drawing on the existing results
- *Solutions availability* – even if certain solutions and suggestions on how to solve the research problem are provided, they are usually realized through modification or expansion of the existing models and tools that are commercial or not accessible to everyone, making their verification, implementation and further improvements difficult

There are several solutions for the distribution of computing resources that are used in practice and are described in the current professional literature. A common characteristic of all these solutions is that they are based on virtualization platforms (KVM, VMware, Hyper-V etc.) since virtualization is a prerequisite for the sharing of resources. Today there are numerous platforms that combine these solutions and technologies, the best known of which are *OpenStack* and *Eucalyptus*. They are free open source platforms that allow for easier administration and distribution of computing resources in cloud computing (OpenStack, 2016; HPED, 2016; HPE Helion Eucalyptus, 2015). These platforms as such are not an independent solution but represent a set of various technologies that complement the existing solutions, including the aforementioned virtualization platforms. On the other hand, the disadvantages and limitations regarding the distribution of computer resources that are originally found in virtualization platforms themselves are transferred to a higher level – i.e., to solutions such as *OpenStack* and *Eucalyptus*.

Several ***commercial solutions*** are being used, whose advantages and disadvantages are as follows.

VMware

One of the most popular virtualization platforms that offers several models of computing resources distribution. The most common way is to start new instances of Web servers by monitoring the resources of the existing servers. Such a solution depends on the inventory of computing resources that are not being used but must be reserved to achieve a high level of system availability. In other words, when the system is not overload, these resources are not utilized, resulting in non-optimal utilization of total resources (VMware, 2015; 2009). This virtualization platform also allows you to add resources to existing Web servers, but under two conditions: This must be defined before starting the server and must be allowed by the operating system.Working memory may be allocated and taken away by defining the maximum and minimum level of resources. In this case, the working memory is either added or subtracted, depending on the system load. As for the CPU, one can only add resources and only for specific operating systems, while seizing CPU resources is not included as a possibility on the VMware platform. The reason why this has not been developed yet is that almost none of the operating systems supports the possibility of seizing a number of cores of a system that is in working condition (Boche, 2009; Lowe, 2013; VMware, 2011). This again means that sub-optimal utilization of the system will occur when the system load is no longer present, because once the CPU resources are allocated (if that is possible for a given operating system) they can no longer be deducted without restarting the Web server.

Hyper-V

This Microsoft virtualization platform works on a similar principle as VMware. When loaded, the system creates new instances of the Web server as long as there are computing resources available (Coughlin, 2016; Halbe, 2015). Maximum and minimum levels within which it is possible to add or subtract RAM can also be defined. The manipulation of CPU resources is resolved in such a way that Web servers are assigned prioritization of CPU resources of the physical server (e.g., if there are two Web servers which have been allocated 400 and 100% of CPU resources respectively, this means that four requests from the first Web server will be resolved prior to one request from the other) (Larson, 2016 n.d.) (Microsoft, 2014). This solution is not optimal because it is limited to prioritizing of the number of cores in a Web server, meaning that it is not possible to add or subtract CPU resources (i.e., – increase or decrease the capacity).

Other commercial solutions are mostly even more limited. In addition to the virtualization platforms above, there are many cloud computing solutions which partially solve this problem. The best known among them is *Amazon Web Services (AWS)* – one of Amazon's cloud computing solutions where optimization of computer resources is performed by automatically or manually triggering Web server instances on and off, depending on the load of the system which must be defined in advance (Amazon Web Services, 2017; Rodge *et al*., 2015). This solution has two shortcomings. The first one is that server profiles are predefined and it is therefore not possible to have a linear increase or decrease in resources, only exceptionally, according to predefined specifications. Thus, the system is not flexible, so users, in spite of the availability of profiles that are optimal for their needs, often use profile servers with more computing resources which ultimately results in non-optimal utilization of resources. Another disadvantage of this solution is system scaling – i.e., optimization of resources which is performed by turning new instances of Web servers on or off, which results in dedicated resources for each new server-every operating system requires specific computer resources (CPU, memories, disk, etc.) in order to function properly, which prevents those resources from being allocated to the Web server. Needless spending of other resources – each new instance of the Web server requires additional resources such as an IP address, consumes additional drive IOPS (Input/Output Operations Per Second), which ultimately results in higher CPU load on physical servers owing to instruction execution queues, operating system licenses etc. Higher maintenance costs – a larger number of Web servers requires greater monitoring and control by the professional staff as well as additional supporting tools that are usually limited (depending on the number of servers) etc. System complexity – a larger number of Web servers increases the complexity of the system, which can ultimately result in a longer period of repair.

Another commercial solution is *DigitalOcean* – one of the better known companies that deals with cloud computing and an alternative to Amazon. However, compared to Amazon's, their solution is even more limited because it does not provide the automatic scaling of resources, only the modification of existing, for which it is necessary to first turn off the Web server (Mitchell Anicas n.d.).

IBM PowerVM – a very expensive solution which allows for the distribution of computer resources, but only in IBM processors or microchips (IMB, 2016).

Xen VMM a virtualization platform that uses *Credit Scheduler*. It represents a mechanism for prioritizing CPU resources, but not for their increase and the decrease in the number of processors (Cherkasova *et al*., 2007).

Non-commercial solutions are mostly even more limited. One such option is *Docker* – unlike previously described solutions, this one is free and open source. It ensures resources for Web servers by reserving scalable computing resources on all Web servers in form of containers which are later used as required. Therefore, a high level of availability of the system is achieved by an advanced use of existing computer resources (Docker Inc, 2015; Ismail and Sheikh, 2016). However, resources are still not exploited in an optimal manner because they are assigned and reserved for a certain Web server and cannot be transferred to other Web servers when not used. Research on other non-commercial solutions has established that there are many approaches to solving the problem of optimizing resources utilization by combining various scripts and programs, which still results in a lesser optimization of resources compared to commercial solutions (Ivan, 2008).

The currently available solutions, whether commercial or non-commercial, do not solve the problem of resources utilization optimization in its entirety. This is mainly due to the fact that these solutions do not allow for multiple utilization of the existing resources which are already allocated to a specific server, but most often allocate new resources reserved for such extraordinary situations (Herrmann *et al*., 2015; VMware, 2012). New resources are often allocated through an excessive increase in the existing resources or by creating additional replicas of servers that are overload.

Furthermore, the lack of a solution that would enable a multiple use of the existing computer resources can also be accounted for by the fact that most virtualization platforms are commercial solutions whose producers (VMware, Hyper-V etc.) wish to use as many servers as possible since software licenses are usually charged by the number of servers. On the other hand, even in non-commercial solutions, which do not require licenses and take a different approach, this problem has not yet been fully resolved.

Proposing a model for automated and improved utilization of existing computer resources on an example of Web servers.

The previous chapter focused on the issue of insufficient utilization of the existing computing resources. To solve this problem, in this chapter we introduce a model that would enable automated and improved utilization of the existing computing resources.

The metamodel of the automatic control system in Fig. 3 shows the functioning of the system at the system level. System generalization is needed to allow for a wide application of the model, independently of the virtualization platform and programming language.

The system consists of two input components – requests (applications or user requirements) and resources (CPU and memory). The central part of the system are the servers that process requests and convert them into output components: Operations (resulting from the processing of requests) and load (during the processing of requests).

The main component of this automated control system is a negative backlink in the form of a closed loop that performs a constant system load check and allocates the resources based on special parameters (configuration, constraints and methods). The leading value in this case is system load. The next step in designing the model is the elaboration of the model at a higher level, where the main components of the system are presented with their parameters and interconnections (Fig. 4).

This approach allows for the application of the model independently of the virtualization platform and coding program. The layered architecture of the new model consists of two levels (physical and virtual) and a two-step integrative process that is being continually executed (Fig. 5).

Fig. 5 shows a single physical server with a virtualization platform (the same principle applies to multiple servers) including an agent as an important component of the new model. The new model consists of two types of agents:

- ***HOST agents*** – located on physical servers; their main task is to check and allocate computer resources across the entire physical server
- ***VM agents*** – located on virtual servers; their main task is sending reports on the state of their own computer resources

In this way communication between all virtual servers and their respective physical servers is made that makes it possible to determine if one of the servers has a shortage or surplus of computer resources. This information exchange results in the last step – the distribution of computer resources between the servers. For better understanding of the entire process, a diagram of activity is provided in Fig. 6.

The entire process shown in Fig. 6 is executed according to pre-defined time iterations and consists of two main steps – Initialization and Resource reallocation.

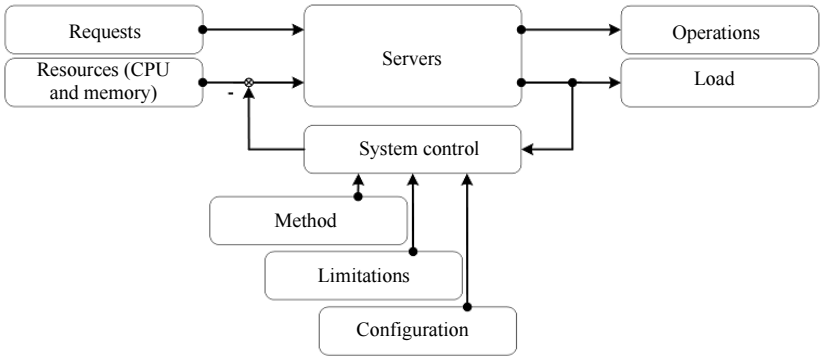

Fig. 3: Automatic control system metamodel

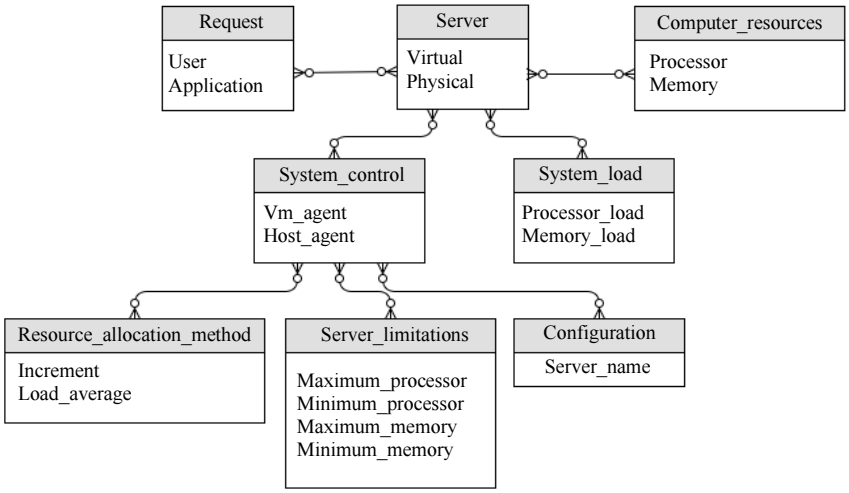

Fig. 4: ERA model of the new model

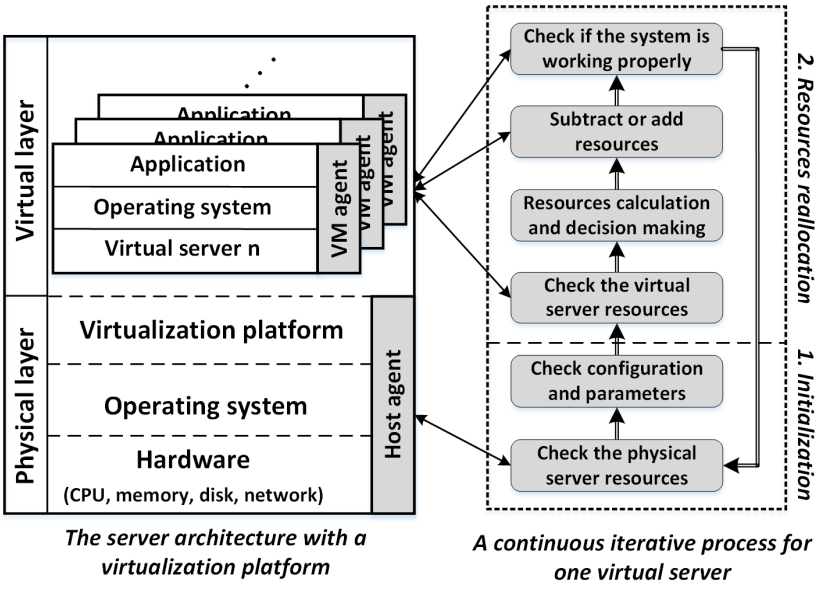

Fig. 5: The layered architecture of the new model

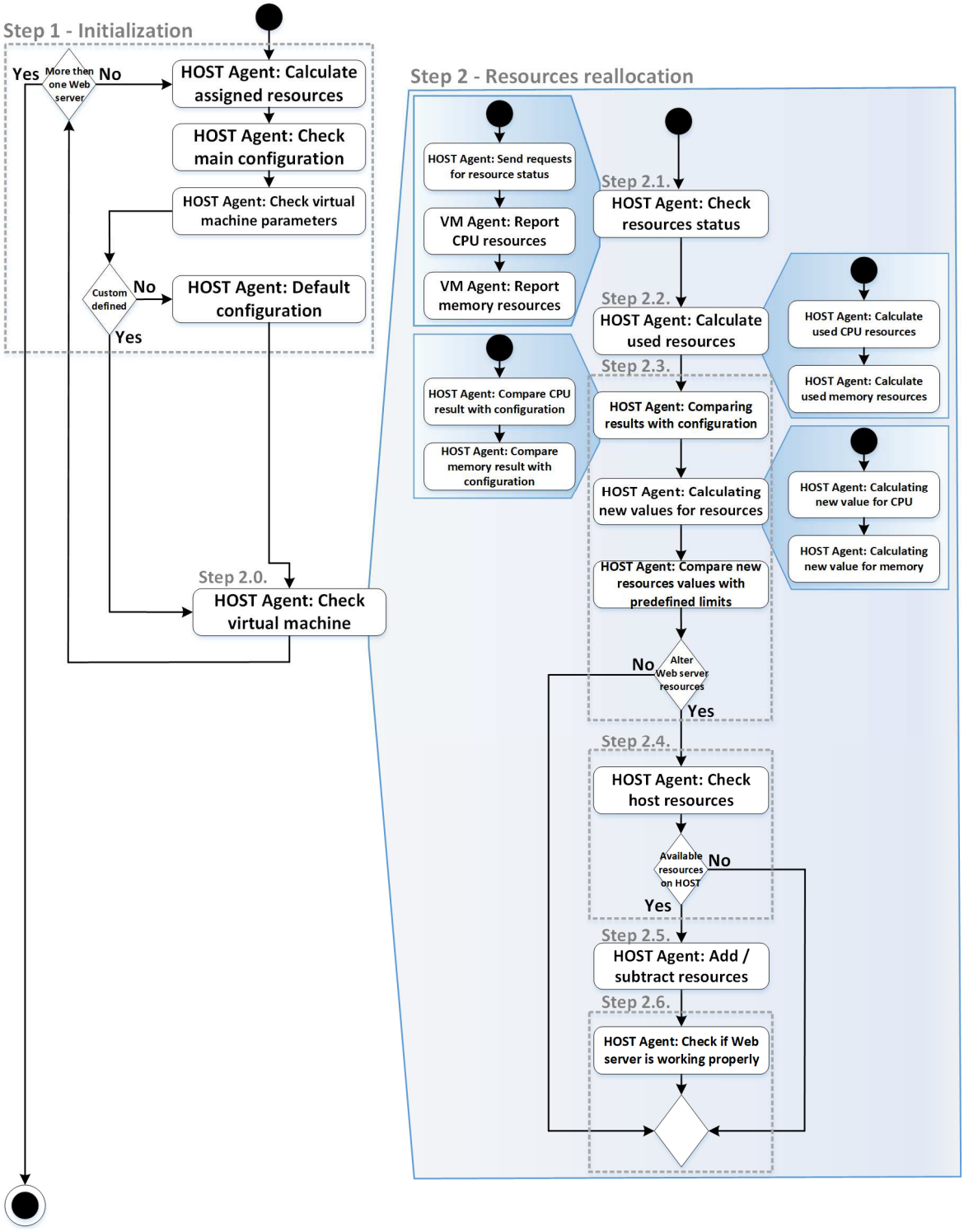

Fig. 6: Activity diagram of the new model

Initialization begins by verifying the total amount of resources allocated to virtual servers. After that, the HOST agent, or the main script, checks the main configuration file and based on it determines: A list of all virtual servers, default parameters for maximum and minimum levels of CPU and memory limits (if defined) and profiles of resources for CPU and memory (if defined). In this step, the HOST agent checks each virtual

server separately by capturing its name and corresponding configuration. The HOST agent first checks if the virtual server has specifically defined parameters in the main configuration and uses them accordingly. Otherwise, the HOST agent will use the parameters of the main scripts which are the same for all virtual servers.

Resources reallocation – after determining the configuration parameters, a six-step process ensues that involves only one virtual server, which means that there will be no interruptions in the functioning of other virtual servers, enabling their continuous operation.

The first step starts with the process of *checking resources on the virtual server*. Computing resources that are checked are: Random Access Memory (RAM), which is expressed in gigabytes (GB) and Central Processing Unit (CPU), which is expressed in the number of its cores. In this step, the main script captures the information on resources calculated by the script from the virtual servers (VM agent). By using a server daemon (xinetd), the VM agent calculates the average load on the CPU and the percentage of free memory that, together with the cache, make up the unused memory. Once the VM agent retrieves the information on the resources, it forwards it to the HTTP port (in this case, arbitrary port number 9299) to enable the HOST agent to reach it.

In the second step, the process of *calculating free resources on the virtual server* is performed. In this step, the HOST agent compares the retrieved information on the status of resources on virtual servers with the resources allocated to those virtual servers. Based on that, it calculates the CPU and memory usage.

The third step starts with the *comparison of the obtained results with the initial configuration and deciding whether resources will be subtracted or added.* In this step, checking and comparison of the maximum and minimum limits of the CPU and memory defined in the configuration occurs. These values are arbitrary and can be changed depending on the system needs. The values that were used in this research are: Maximum CPU limit, minimum CPU limit, maximum memory limit and minimum memory limit. These limits will vary depending on whether the virtual server is allocated less or more resources. For example, if a virtual server is allocated 2 GB of memory, its limits are 0.8 GB or 1.6 GB. In other words, a virtual server has 0.8 GB of 'workspace' and these resources are allocated to it. However, if a virtual server is allocated 10 GB, its limit is 4 GB or 8 GB, which means that its 'workspace' is 4 GB. Consequently, in spite of a minimum/maximum limit, there are still too many allocated resources. Therefore, additional measure called minimum free memory is applied to further increase the utilization of computing resources. It also controls the distance between the minimum and the maximum limit, while providing resources for the proper functioning of the virtual server. This step also includes making a decision on whether to add or subtract resources and if so, how many of them. There are two ways of adding or subtracting resources are implemented: *LA (Load Average)* and *INCREMENT*.

LA (Load Average) – advanced allocation of resources (only possible for the CPU) expressed as an average load, which enables their precise and rapid growth of resources. For instance, if the average load is 500%, it is necessary to add five cores to enable the system to operate normally. The number by which we increase the number of cores (multiplier) is also a variable that allows for a better allocation of resources. For example, if the multiplier is 1.2 and the load is 500%, it will be allocated to six cores (1.2×5=6). This allows for the allocation of an additional reserve of resources when the system is under unexpected load. In addition to the multiplier, it is possible to choose more than one way of rounding the result (the number of required cores): HALF UP – standard rounding (e.g., 0.5 is rounded up to 1), UP – rounding (e.g., 0.1 is 1) DOWN – rounding down (e.g., 1.7 is 1).

INCREMENT – default rules used to define the way of allocating and seizing of resources for all virtual servers. They apply to both the CPU and memory. The system is designed to first check the main configuration in which the following values are set: *vm* – name of the virtual server; *cpu_min* – minimum CPU value (number of cores); *cpu_max* – maximum CPU value (number of cores); *mem_min* – minimum memory value (in GB); *mem_max* – maximum memory value (in GB); *cpu_balance_logic* – increase/decrease CPU resources profile (X:Y, where X represents system load and Y represents increase/decrease in resources); *mem_balance_logic* – increase/decrease memory profile (calculated in the same manner as the previous values).

The purpose of the main configuration is that all values can be individually manipulated, in accordance with the needs of particular virtual servers. If the values of virtual servers are defined in the main configuration, they are determined from the main scripts and these parameters are thus valid for all virtual servers. Here are some examples of memory profiles (the same rules apply to the CPU): 2:1 – if the memory is larger than 2GB, increment (or decrease) by 1; 4:2 – if the memory is larger than 4GB, increment (or decrease) by 2; 16:4 – if the memory is larger than 16GB, increment (or decrease) by 4. This can be illustrated by the following example: At the moment of verification, the virtual server has GB of memory and if additional resources are necessary, it will increment by 2GB. If the virtual server initially has 18GB of memory and additional resources are necessary, it will increment by 4GB. If the CPU for the LA resources allocation is not defined, it will be calculated according to the INCREMENT principle. It is important to note that these calculations comply with the minimum and maximum values allowed for virtual servers, as defined in the configuration.

The fourth step starts with the process of *verification of physical server resources*. Before subtracting or adding resources, physical servers are verified so as to

prevent the load of the entire system. If it is necessary to add resources and if they are available on the physical server, the procedure continues to the following step. Otherwise, the verification of the next virtual server takes place, meaning that the whole process starts from the beginning.

The fifth step starts with *adding or subtracting of resources*. In this step, the resources are added or subtracted depending on the load of the virtual server. The procedure is done simultaneously for the CPU and the memory. In other words, it might be possible to add resources to the CPU and decrease them from memory and vice versa.

The last step starts with the process of *verifying whether the virtual server is working*. While the process of adding or subtracting resources is carried out, it is necessary to make sure that there are no interruptions or errors in the operation of the main service. Since the validation of the model is to be performed on an example of Web servers, the verification will be done using the HTTP status code. This means that after each resource operation (addition or subtraction) a Web server check will be performed. In other words, if the system control receives HTTP 200 code from the Web server, the system operation is not compromised, for instance, returns HTTP 500 code, there is an error in the functioning of the Web server.

After the second step, verification of the main configuration is done by checking whether more virtual servers exist. If they do, the entire procedure is repeated from the beginning. Otherwise, the process ends and waits for a predefined time interval until it is restarted. In this research, the whole process on one physical server with three virtual servers was always completed within ten seconds. The proposed model was developed in such a way that it is possible to define how many times the process is to be performed.

For the new model to function flawlessly, several control and monitoring mechanisms have been implemented: Continuous execution system, resource monitoring and possibility of warning.

Continuous execution system – the system (new solution) is executed periodically and it is possible to modify the time when it will start. Before the new start (repetition of the two main steps above), the verification of the existing process takes place. If the process is still running, the next launch is delayed until the process is completed.

Resource monitoring – there are two levels of resources verification and monitoring. At the level of a physical server, prior to allocating resources to a virtual server, the HOST always verifies available resources of the physical server to ensure proper functioning of the system. At the level of a virtual server, the HOST agent ensures that the virtual server has continuously available resources, using minimal value parameters (CPU min and mem min).

Possibility of warning – there are two levels of verification systems that send alerts (e-mails) if there is an error or an interruption in the operation of the system. One of them is activated during the execution of the process and the other in case of congestion or errors that occur on a single virtual server during the process of adding or subtracting computer resources.

One of the motives for this research is the inefficient use of computing resources in Web servers. Consequently, they were selected as a concrete case for the application and validation of the new model. Figure 7 contains a graphical representation of the entire solution or model that should be applicable to *n* Web servers or *n* Web clusters. The model was designed in a way that supports the existing requirements and Web Farm architecture. In other words, it is possible to scale all system components if necessary.

As can be seen from Fig. 7, each one of the physical servers has a new model component called the HOST agent, whose main task is to verify and allocate computer resources to the entire physical server. To do this, it continually checks the other component of the model called the VM agent. It is located on all virtual servers and its task is to send reports about the status of its own computer resources.

Evaluation

The environment on which the initial testing and measurement of resource consumption was carried out is shown in Figure 7. The main focus of the new model will be on Web servers, in which, according to our previous research research, the highest amount of allocated resources whose consumption or occupancy is not optimally utilized will be found. This is why the other components of Web farms (classifiers of databases, tools and supporting services) are marked in a lighter shade (Fig. 8).

As already mentioned, one of the main goals of this research is to provide an open source solution as well as a detailed description of the overall environment in which the research was done. The testing environment encompasses three physical servers (HP ProLiant DL360 G7). All the three physical hosts feature the same characteristics: 2 CPUs, both of them quad-core, 32GB of RAM and a 140GB local hard drive.

Physical servers contain a total of nine virtual servers with the following specifications: Two load balancers (single-core CPU, 1 GB of RAM, 10GB local hard drive), six web servers (single-core CPU, 0.6 GB of RAM, 10GB local hard drive) and one database (quad-core CPU, 4 GB of RAM, 20GB local hard drive).

Having described the testing environment, we will describe the process of model validation. For the purpose of validation, an application was developed based on the model. Also, tests were carried out to simulate user requests which account for the computer server resources load.

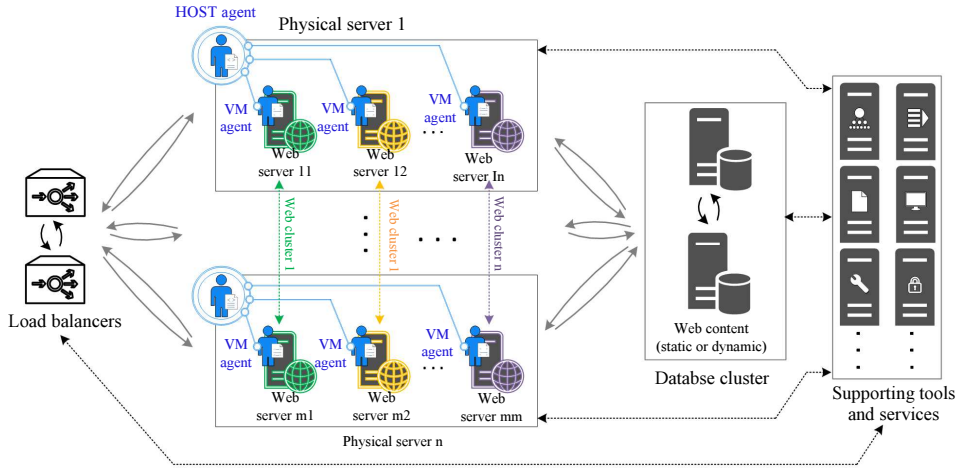

Fig. 7: Representation of the new solution on an example of Web servers

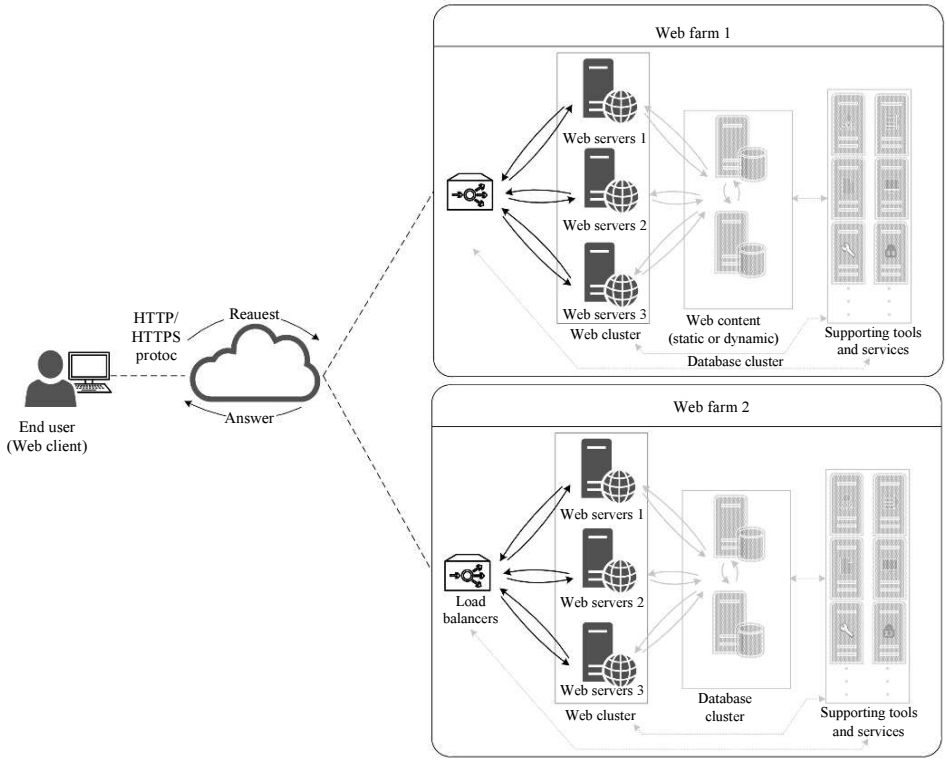

Fig. 8: An overview of the environment on which the testing and building of the new model was performed

The purpose of these tests was to verify whether it is possible to add and subtract computer resources (CPU and memory) without the need to restart the server. Today, almost all operating systems have implemented mechanisms to determine how much of the system is active – i.e., when the system was last booted. This mechanism was used to establish whether the server would be resumed after the allocation of resources. The experiment was performed by means of the following tools for the simulation of requests:

- *ApacheBench* – allows for simple and rapid tests without an advanced configuration. The tool was used in the first version of the model
- *Apache JMeter* – allows for advanced (parallel) testing between multiple Web sites located on more Web clusters. This tool was applied in the three main evaluation tests

The main tests were developed on the basis of the aforementioned preliminary research in which data provided by several IT companies were used. Drawing on

the data such as the number of simultaneous users at a given time, we defined tests that would yield results on computer resources load (CPU and memory). Next, we developed several groups of tests to verify the following three options: Adding and subtracting computer resources, advanced addition and subtraction of computer resources and application on multiple Web clusters.

It should be noted that almost all system monitoring tools gather data every five minutes, registering the average value for the said period. This value is predefined and fixed. Consequently, it is possible that at particular times the intervals of values are even larger, but are not displayed.

During the tests, the same algorithm was used in the HTTP/HTTPS traffic load balancer. This algorithm, entitled Weighted Least Connections, is mostly used when the number of requests that a system can take is known. Based on this information, the co-called focus is defined, that is the maximum number of requests (connections) that a single node (Web server) can process. The process takes place cyclically, wherein are first taken to the node with the highest number of free resources available for processing. This approach ensures an equal load on all Web servers, a single representative sample of which will be shown.

In the following subsection, tests will first be performed without the usage of the new application that was developed on the basis of the new model for automated and improved utilization of the existing computing resources. Subsequently, the same tests will be conducted again, this time using the developed application. In both cases, the computer server resources will be monitored. After the tests have been completed, a comparison of the results will be made to establish if the use of the existing computer resources has been improved. If the new model proves satisfactory, improved utilization of the existing computing resources, when compared to the original solution (CPU – i.e., the number of cores and memory) should be reflected in the figures that represent the experiment results.

Conducting the Experiment

Following the analysis of real-world systems (taking the number of user requests and resource load in a given time as parameters), a more comprehensive experiment consisting of several scenarios drawing on examples from practice was defined, encompassing: A linear increase or decrease in user requirements, sudden increase or decrease in user requirements and sleep (when there are few or no user requirements).

Figure 9 shows the parameters of the experiment in the *Jmeter* tool that involved a number of simultaneous users (user requests) alternating within a sixty-minute period. As already mentioned, the time unit (or the duration of the experiment) was not important, since in this case value was system load. The main goal of this experiment was to check the three aforementioned scenarios identified in the real-world practice in a given period (in this case, sixty minutes).

As can be seen in Fig. 9, the design of the experiment allowed for all the three aforementioned scenarios to be tested, namely: A linear increase or decrease in user requirements, sudden increase or decrease in user requirements and sleep.

Figure 10 shows the results of the conducted experiment – i.e., the distribution of requests for computing resources in the conducted experiment.

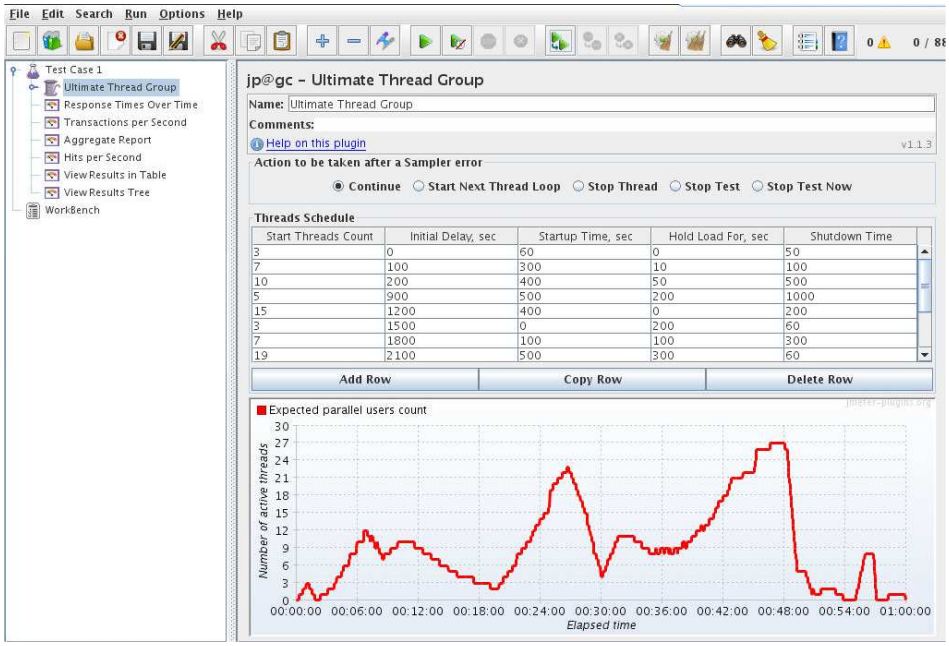

Fig. 9: Parameters of the experiment in *Jmeter* tool

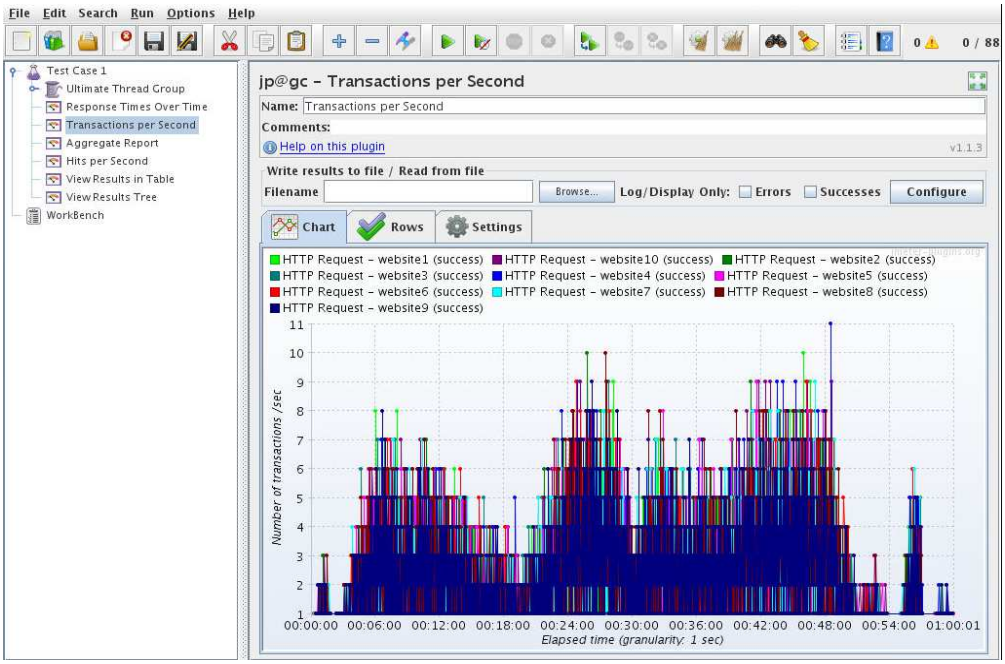

Fig. 10: Number of requests in the conducted experiment

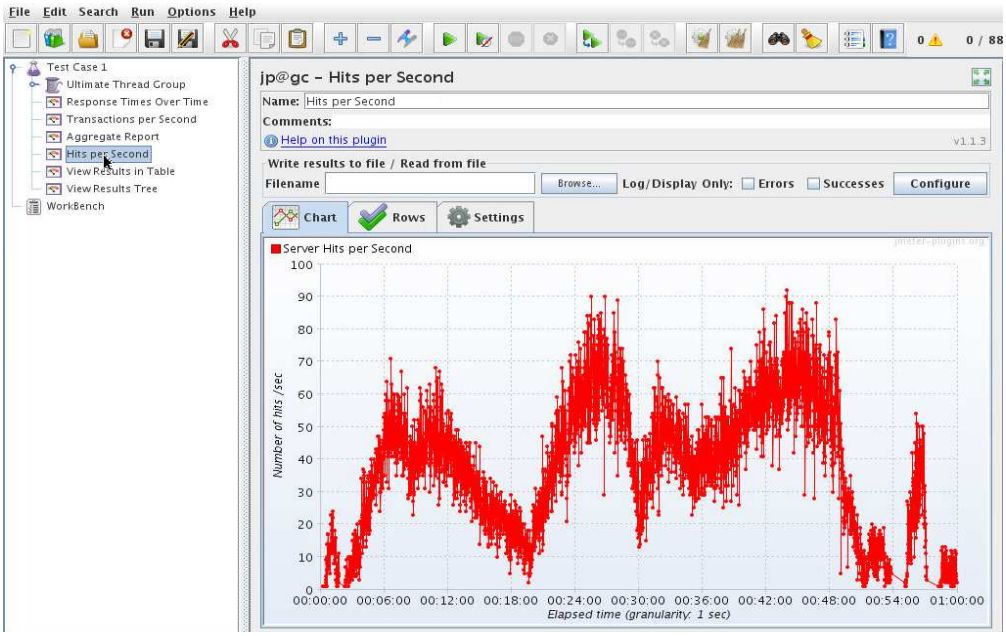

Fig. 11: Number of loadings per second in the conducted experiment

As can be seen in Fig. 10, the defined experiment parameters match the figures showing the executed requests, which is in accordance with our expectations. The results of the executed requests (or the number of loads per second) are shown in Fig. 11.

The results of the experiment performed with and without the application that was developed on the basis of the new model are presented in the following subsection.

Results

In order to verify whether the implementation of the new model would allow for better utilization of the existing computer resources (CPU and memory), two tests were performed (with and without the usage of the application developed on the basis of the new model, respectively).

In continuation, the load of only one virtual Web server will be observed, since the Web farm architecture, which

consists of the HTTP/HTTPS system for the distribution of requests, which ensures that each Web server receives the same number of requests based on an algorithm.

The load of the processor and the number of cores allocated for the test conducted without the usage of the application developed on the basis of the new model is shown in Fig. 12. The observed Web server features four cores.

Figure 12 confirms the main premise of this research, which is that there are times when the existing allocated computer resources (in this case, the processor) are not sufficiently used. It is also evident that there are two periods when the system was overloaded, i.e., when the allocated number of cores was insufficient to process the number of user requests, which ultimately resulted in the system running more slowly.

The memory load and memory allocation for the experiment conducted without the usage of the application developed on the basis of the new model can be seen in Fig. 13.

Figure 13, the memory load is given in GB to show the portion of the resources that were never used. The reason why oscillations in memory load are not visible in the figure is that their amplitudes are expressed in MB. The memory load is fairly low because generic Web pages were used to make the test repeatable for the purpose of further research and improvement of the model.

From the results in Fig. 13 it can be concluded that the system is never overload. However, such a state is not efficient because there are resources (in this case, the memory) that are under-utilized, which also confirms the main premise of this research.

We further show the results of the test in which the application developed on the basis of the new model *was* used.

It is important to note that the same experiment parameters (number of simultaneous users in a sixty-minute period) were used here as in the first test.

The model was developed in such a way that it is adaptable to particular systems – i.e., the administrator himself decides on the system input parameters (global parameters, initial parameters etc.). Since the system administrator has the freedom to define the input parameters, they are configured to utilize the processor more efficiently as well as to increase or decrease the memory at a given moment. These features of the model again support the claim made in the first phase of the experiment – i.e., that it is possible to add or decrease not only the processors (number of cores), but also the memory. Figure 14 represents the processor load and the number of cores allocated for the conducted experiment with the usage of the application developed on the basis of the new model.

Figure 14 shows that a sufficient number of cores were allocated throughout the experiment to execute user requests, i.e., there were no periods in which a large number of unused processors or overloads occurred. Fig. 15 represents the memory load and the memory allocated to the conducted experiment using the application developed on the basis of the new model.

Although it was already mentioned that the observed system – i.e., Web sites, does not require a large amount of memory, it is evident that in this case it was used far more efficiently.

Namely, the deviations were below 2 GB (as defined by the administrator), which is much better than the case when the application developed on the basis of the new model was not used. The periods in which a memory increase or decrease occurred were intentionally adjusted by setting the input parameters (upper and lower permissible load limits) to once again prove that it is possible to add or decrease memory *during* server operation.

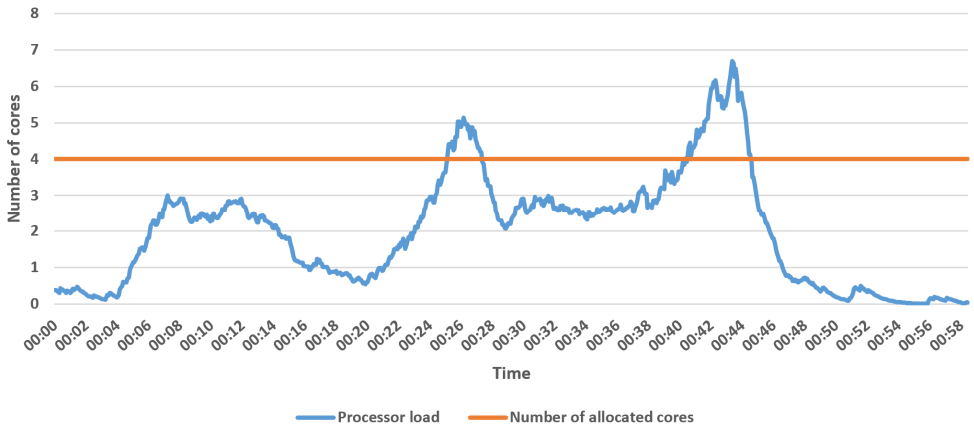

Fig. 12: Processor load and the number of allocated cores for the experiment conducted without the usage of the application developed on the basis of the new model

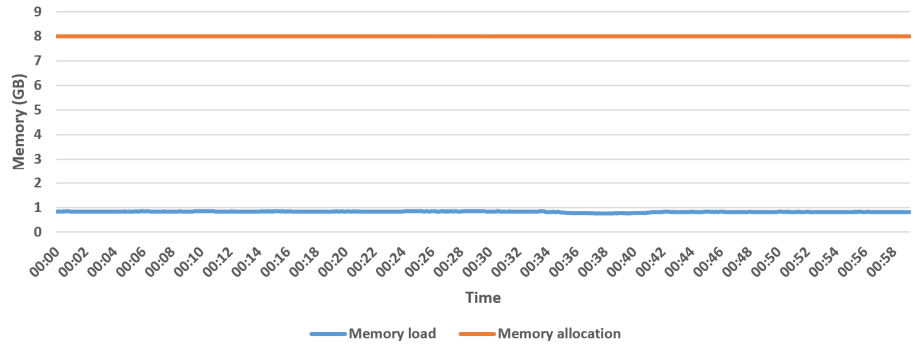

Fig. 13: Memory load and memory allocation for the experiment conducted without the usage of the application developed on the basis of the new model

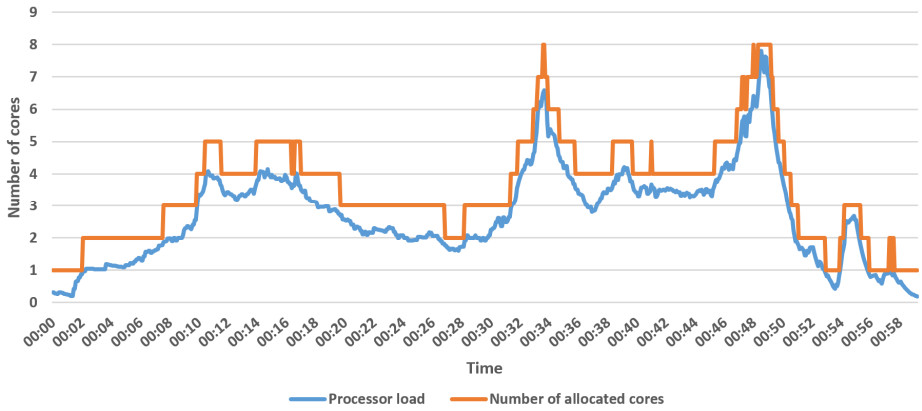

Fig. 14: Processor load and the number of allocated cores for the experiment conducted using the application developed on the basis of the new model

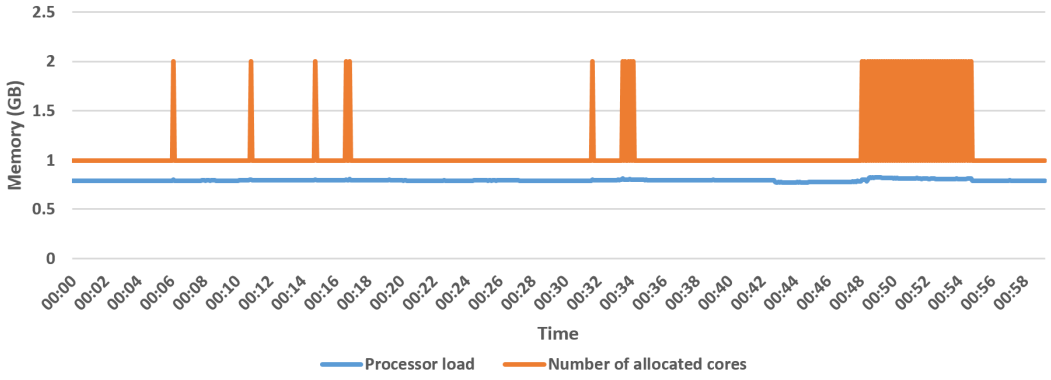

Fig. 15: Memory load and memory allocation for the experiment conducted using the application developed on the basis of the new model

From all of the above, we can conclude that the usage of the proposed model in virtualized systems allows for better utilization of the existing computing resources (CPU and memory) with respect to the original state.

Discussion

In comparison with the existing solutions, the new model proposed in this study should provide: More efficient use of the existing resources, higher system availability, advanced resource allocation, open source solution, higher financial profitability, simplicity of use and environment friendliness.

More Efficient use of the Existing Resources

The model enables higher resource utilization with a smaller number of virtual servers because it is possible to add and subtract resources (CPU and working

memory) without the need to restart the virtual server. In other words, the system always uses as much resources as needed for proper operation and does not need to start another instance of the virtual server that would cause additional energy consumption.

Higher System Availability

the system has a faster response – i.e., a faster increase in resources in the case of system load. This is because it is not necessary to restart the virtual server in order to change the resources.

Advanced Resource Allocation

Within the existing solutions, resources are allocated linearly to the maximum permissible limit. Based on the speed of seizing free resources, the new model should be able to assess the level of resources necessary for the optimal functioning of the system. This method of allocation of resources is intended to allow virtual servers to more easily manage emergency situations when a sudden increase in requests occurs.

Open Source Solution

Today there are many commercial solutions that do not only fail to solve the problem, but are very expensive and demanding in terms of computing resources needed for their proper functioning. For the sake of social contribution, the entire solution will be open source, which is also one of the main goals of this research. The overall test environment in which our research will be conducted will be described in detail so that the application and model verification can be repeated as easily as possible for the purpose of further improvement and research by other authors that may be interested in this topic.

Higher Financial Profitability

Multiple use of the existing resources enables a lower number of required servers, both virtual and physical. A smaller number of servers in turn results in: Simpler infrastructure, lower maintenance and hardware replacement costs, lower monthly expenses of the data center (e.g., electricity and air conditioning) etc. All of these are ultimately reflected in lower OPEX costs.

Simplicity of Use

The new solution applies only to automated and optimized allocation of computing resources, as opposed to existing solutions that are usually expensive, complex and unspecialized. For their proper use and configuration, experts with years of experience in particular aforementioned areas are often needed, or it is necessary to provide additional staff training and certification. Such complex systems and platforms usually consume additional system resources to function normally, unlike the new solution which does not feature restrictions such as the minimum necessary computer resources in order to work.

Environment Friendliness

Last but not least, an indirect effect of the new solution is its environment friendliness, owing to a smaller number of servers. Electricity which is used in most data centers does not comply with ecological norms and standards as it does not come from renewable sources of energy. The new solution operates with fewer servers, enabling greater efficiency and better results, which ultimately results in lower power consumption.

From the above, it can be seen that the new model should be able to address many of the disadvantages of the existing solutions, since it allows for automated and improved distribution of the existing computing resources.

Conclusion

This paper presents a new model for automated and improved utilization of existing computer resources on an example of Web servers. In the literature review at the beginning of this paper, it was established that servers are the main cause of high costs of data centers, which was subsequently argumented. Additional analysis revealed that there are systems such as Web farms where the existing computer resources are underused. Ultimately, such a solution should not only enable easier maintenance of the system, but also result in large financial savings.

A detailed study of previous scientific research and solutions from practice led us to conclude that the problem of insufficient utilization of existing computer resources has so far not been effectively resolved, which also motivated us to conduct this research. One of the disadvantages of the existing solutions is that they do not address the issue of using the existing resources more efficiently. Instead, they usually add new servers or migrate virtual servers to other physical servers in critical situations, for which even more computer resources are needed. The second common approach to solving this problem is the process prioritization, whereby servers that require resources are given the highest priority in executing the process. The disadvantage of this approach is that resources cannot be increased or reduced, only prioritized, which still results in inclusion of resources that are not being used. Another shortcoming of the existing solutions is that it is not possible to add or decrease computer resources (CPU and memory) without restarting the server. Furthermore, a large number of the existing solutions focuses only on CPU or memory, but not on both. Finally, the existing solutions do not provide an advanced resource allocation option that would enable resources to be decreased or added faster during critical moments.

In order to validate the new model, an application was developed to be applied to Web servers, where the problem of inefficient use of computer resources had been recognized. Several tests were run that showed highly satisfactory results because it was possible to subtract and add resources (CPU and memory) without

the need to restart the server. This new application, unlike the currently available solutions solutions, uses the existing computer resources more efficiently and does not depend on the number of user requests. The only limitation in this case are the resources of the physical server itself, which was to be expected. Although the main objectives of this research were achieved, it is possible to further improve and upgrade the system. This may be done by developing a Graphical User Interface (GUI) that would make it easier to manage parameters such as the maximum and minimum limits of resources per Web server, selectable models of resource exchange (LA or INCREMENT), minimum of free memory etc.

The target audience for this research includes small and mid-sized IT companies, which cannot afford a variety of commercial solutions, often spending a great deal of money on computer resources. The other target group is the academic community with regards to further research on this topic. In our future work we therefore intend to describe the entire solution in detail, from its infrastructure to the description of the system and the test environment. It is important to note that the entire solution will be open source in order to be more accessible for further improvements and enhancements.

Author's Contributions

Dino Alagić: The main responsible author for the literature review. Contributed in preparation and writing.

Ivan Magdalenić: The author responsible for review preparation and improvements. Contributed in preparation, organization and supervision.

Ethics

This article is original and contains unpublished material. The corresponding author confirms that all of the other authors have read and approved the manuscript and no ethical issues involved.

References

Amazon Web Services, 2017. Amazon EMR developer guide. Amazon Web Services.

Beloglazov, A., J. Abawajy and R. Buyya, 2012. Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing. Future Generation Comput. Syst., 28: 755-768. DOI: 1016/j.future.2011.04.017

Boche, J., 2009. vSphere Memory Hot Add/CPU Hot Plug. *Boche*.

Celesti, A., F. Tusa, M. Villari and A. Puliafito, 2011. An approach to enable cloud service providers to arrange IaaS, PaaS and SaaS using external virtualization infrastructures. Proceedings of the 2011 IEEE World Congress on Services, Jul. 4-9, IEEE Xplore Press, Washington, pp: 607-611. DOI: 10.1109/SERVICES.2011.92

Cherkasova, L., D. Gupta and A. Vahdat, 2007. Comparison of the three CPU schedulers in Xen. ACM SIGMETRICS Performance Evaluation Rev., 35: 42-51. DOI: 10.1145/1330555.1330556

Corsava, S. and V. Getov, 2003. Intelligent architecture for automatic resource allocation in computer clusters. Proceedings of the International Parallel and Distributed Processing Symposium, Apri. 22-26, IEEE Xplore Press, Nice, France. DOI: 10.1109/IPDPS.2003.1213369

Coughlin, F., 2016. Onboarding Microsoft Hyper-V resources. BMC Software, Inc.

Docker Inc., 2015. Building a Continuous Integration Pipeline with Docker. Docker Inc.

Frachtenberg, E., F. Petrini, J. Fernandez and S. Coll, 2002. Scalable resource management in high performance computers. Proceedings of the IEEE International Conference on Cluster Computing, Sept. 26-26, IEEE Xplore Press, Chicago, pp: 305-314. DOI: 10.1109/CLUSTR.2002.1137759

Gruber, C.G., 2009. CAPEX and OPEX in Aggregation and Core Networks. Proceedings of the Conference on Optical Fiber Communication - Incudes Post Deadline Papers, Mar. 22-26, IEEE Xplore Press, San Diego, pp: 9-11. DOI: 10.1364/OFC.2009.OThQ1

Guan, H., R. Ma and J. Li, 2014. Workload-Aware Credit Scheduler for Improving Network I/O Performance in Virtualization Environment. IEEE Trans. Cloud Comput., 2: 130-142. DOI: 10.1109/TCC.2014.2314649

Halbe, S., 2015. Hyper-V. BMC Software, Inc.

HPED, 2016. Hewlett packard enterprise development LP, 2016. Eucalyptus 4.2.2 Administration Guide. Build 3221.

HPE Helion Eucalyptus, 2015. General purpose reference architecture: HPE helion eucalyptus. HPE Helion Eucalyptus.

Ichikawa, Y. and N. Komoda, 2016. Scenario-Based Task Executor for IT Resource Management. Proceedings of the 5th IIAI International Congress on Advanced Applied Informatics, Jul. 10-14, IEEE Xplore Press, Kumamoto, pp: 888-893. DOI: 10.1109/IIAI-AAI.2016.250

IMB, 2016. Server virtualization with IBM PowerVM. IBM, Inc.

Islam, S.S., M.B. Mollah, I Huq and A. Ullah, 2012. Cloud computing for future generation of computing technology. Proceedings of the IEEE International Conference on Cyber Technology in Automation, Control and Intelligent Systems, May 27-31, IEEE Xplore Press, Bangkok, pp: 129-134. DOI: 10.1109/CYBER.2012.6392539

Ismail, U. and B. Sheikh, 2016. Continuous Integration and Deployment with Docker and Rancher. 1st Edn., Rancher Labs.

Ivan, B., 2008. Samooblikujuća arhitektura sustava zasnovanih na uslugama. PhD Thesis, University of Zagreb.

Herrmann, J., D. Parker and S. Radvan, 2015. Red Hat Enterprise Linux 7 Virtualization Tuning and Optimization Guide. 1st Edn., Red Hat, Inc., North Carolina.

Larson, R., 2016. Controlling processor resources in hyper-v guests. VirtualizationAdmin.

Li, Y., H. Wang, J. Dong, J. Li and S. Cheng, 2013. Operating cost reduction for distributed Internet data centers. Proceedings of the 13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, May 13-16, IEEE Xplore Press, Delft, pp: 589-596. DOI: 10.1109/CCGrid.2013.106

Lowe, S., 2013. vSphere 5.1: Hot add RAM and CPU. *VirtualizationAdmin.*

Ma, F., F. Liu and Z. Liu, 2012. Distributed load balancing allocation of virtual machine in cloud data center. Proceedings of the IEEE 3rd International Conference on Software Engineering and Service Science (ESS' 12), pp: 20-23.

Marrone, S. and R. Nardone, 2015. Automatic resource allocation for high availability cloud services. Procedia Comput. Sci., 52: 980-987. DOI: 10.1016/j.procs.2015.05.176

Microsoft, 2014. Hyper-V dynamic memory overview. *Microsoft*.

Mitchell Anicas, How To Resize Your Droplets on DigitalOcean. *DigitalOcean.*

OpenStack, 2016. OpenStack documentation.

Pham, V., E. Larsen, Ø. Kure and P.E. Engelstad, 2010. Gateway load balancing in future tactical networks. Proceedings of the IEEE Military Communications Conference MILCOM, Oct. 31-Nov. 3, IEEE Xplore Press, San Jose, pp: 1844-1850. DOI: 10.1109/MILCOM.2010.5679555

Rodge, A.S., C. Pramanik, J. Bose and S.K. Soni, 2015. Multicast routing with load balancing using amazon web service. Proceedings of the Annual IEEE India Conference, Dec. 11-13, IEEE Xplore Press, Pune. DOI: 10.1109/INDICON.2014.7030543

Sampson, J. and D.M. Tullsen, 2012. Battery provisioning and associated costs for data center power capping. UC San Diego.

Song, X., J. Shi, H. Chen and B. Zang, 2013. Schedule processes, not VCPUs. Proceedings of the 4th Asia-Pacific Workshop on Systems, Jul. 29-30, Singapore, pp: 1-7. DOI: 10.1145/2500727.2500736

Soundararajan, V. and B. Herndon, 2014. Benchmarking a Virtualization Platform. Proceedings of the IEEE International Symposium on Workload Characterization, Oct. 26-28, IEEE Xplore Press, Raleigh, pp: 99-109. DOI: 10.1109/IISWC.2014.6983049

Teodoro, G., T. Tavares, B. Coutinho, W. Meira and D. Guedes, 2003. Load balancing on stateful clustered Web servers. Proceedings of the 15th Symposium on Computer Architecture and High Performance Computing, Nov. 12-12, IEEE Xplore Press, Sao Paulo. DOI: 10.1109/CAHPC.2003.1250340

VMware, 2015. Performance best practices for VMware vSphere.

VMware, 2009. Understanding memory resource management in VMware® ESXTM server. VMware, Inc.

VMware, 2011. vSphere 5 documentation center. VMware, Inc.

VMware, 2012. vSphere resource management. EN-000793-00.

Wiboonrat, M., 2014. Life cycle cost analysis of data center project. Proceedings of the 9th International Conference on Ecological Vehicles and Renewable Energies, Mar. 25-27, IEEE Xplore Press, Monte-Carlo. DOI: 10.1109/EVER.2014.6844139

Yanmaz, E., O.K. Tonguz and R. Rajkumar, 2005. Is there an optimum dynamic load balancing scheme? Proceedings of the GLOBECOM - IEEE Global Telecommunications Conference, Nov. 28-Dec. 2, IEEE Xplore Press, St. Louis, pp: 598-602. DOI: 10.1109/GLOCOM.2005.1577694

You, X., J. Wan, X. Xu and J. Zhang, 2011. ARAS-M: Automatic resource allocation strategy based on market mechanism in cloud computing. J. Comput., 6: 1287-1296. DOI: 10.4304/jcp.6.7.1287-1296

Zhanjun, Z., Y. Xueliang and H. Chengde, 2000. Key issues of resource management in distributed multimedia computer systems. Proceedings of the International Conference on Communication Technology, Aug. 21-25, IEEE Xplore Press, Beijing, pp: 1628-1632. DOI: 10.1109/ICCT.2000.890972