A Statistical Approach to Virtual Server Resource Management

Dimitris Kontoudis
Corresponding Author
University of Macedonia
Department of Applied Informatics
156 Egnatia str., 54006, Thessaloniki, Greece
tel. +30 210-3579274
fax. +30 210-3574230
kontoudis@uom.gr

Panayotis Fouliras
University of Macedonia
Department of Applied Informatics
156 Egnatia str., 54006, Thessaloniki, Greece
tel. +30 2310-891843
pfoul@uom.gr

Abstract

Resource management is of key importance in a wide array of computer and network environments. Failure to effectively allocate resources necessary for smooth infrastructure operation may result in impediments in successful service provisioning. This is especially prominent in environments whose workloads present fluctuating resources demand, such as in Cloud-based infrastructures. The mathematical science of Statistics offers an extensive theoretical and practical toolbox that can be of potential use in tackling such problems in the Information Technology domain. We propose a novel approach to resource management of virtualized datacenter components. We have worked towards creating a mechanism that allows for dynamic resource allocation, adapting to the changing demand patterns. Our system incorporates a resource controller, based on Statistical Process Control, which permits the online management of a virtual machine's processor and memory capacity through the real-time analysis of its observed performance. We successfully demonstrate our approach, with real data, on an architecture running the core banking environment of a financial institution. The controller successfully manages the resource allocation of the virtual machine and stabilizes its performance while business workloads are being processed. Our approach can be extended to realize different workload models and to manage other types of hypervisor provisioned resources.

Keywords: Resource Management, Statistical Process Control, Cloud, Elasticity, Autoscaling

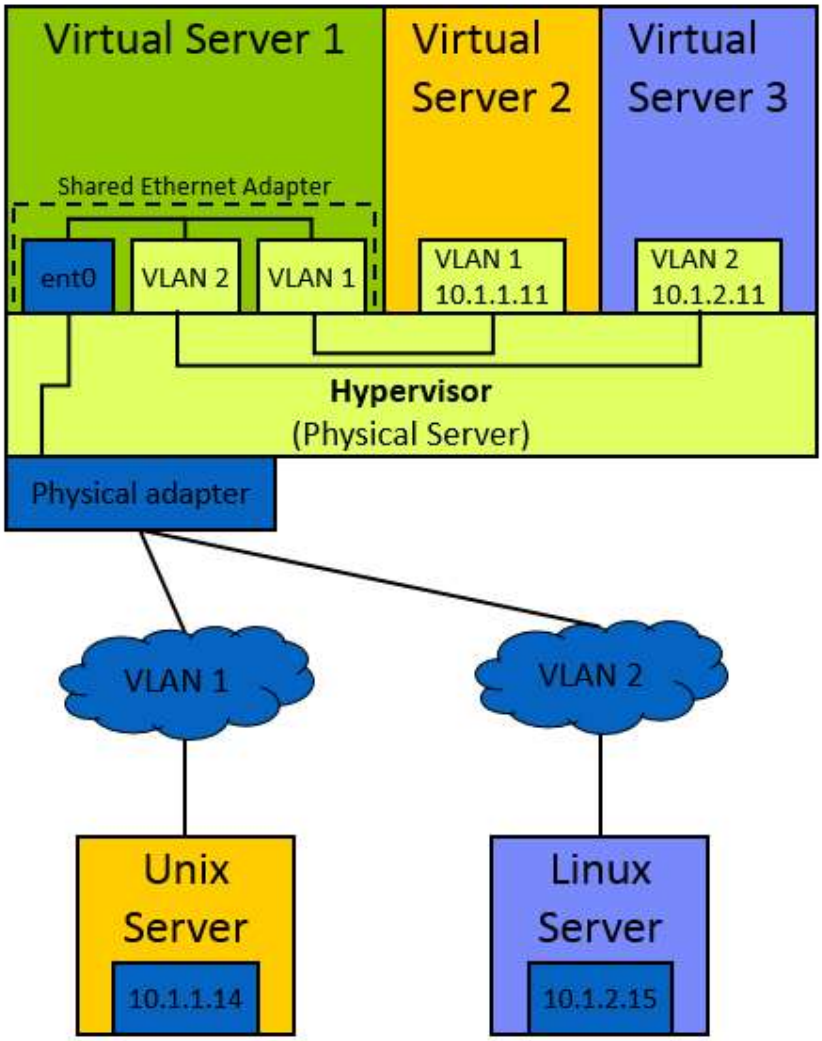
1. Introduction

Resource management is of key importance in a wide array of computer and network environments. Failure to effectively allocate the resources necessary for smooth infrastructure operation may result in impediments in successful service provisioning. This condition is more prominent in commercial infrastructures due to the very fact that over- or under-provisioning of computing and network resources can incur negative financial impact. To mention but a few examples, a major challenge faced in successfully administering commercial Cloud environments is the processing of the enormous amount of data, thus, in effect, the dynamic management of the infrastructure's resources - mainly, processing power and memory capacity, virtual networks and network bandwidth, data storage and virtual machines [1]. This allows for *cloud elasticity*, facilitating infrastructure adaption to the varying customer demand (dynamic leasing and releasing of resources). In a related example, data centers employ virtualization at several layers in order to host cloud, online, distributed and other enterprise, commercial or research applications. This is facilitated via heterogeneous networked computing servers, storage arrays and active network elements. Proper planning of resources, accurate provisioning along with targeted monitoring ensures the performance and guaranteed operation of application and services hosted by the data center [2].

Modern computer and network implementations in data centers exhibit a high degree of convergence to a common design and operational entity [3]. The introduction of computing servers acting as active network nodes (Fig.1) allows for flexible management of the underlying infrastructure and of the provisioned protocols and services [4]. This has been made possible due to advances in server virtualization (also referenced to as system or machine virtualization) and the introduction of the hypervisor (implemented by a specific thin software layer also known as the virtual machine monitor).

It becomes apparent that resource management is a, multi-factored, important challenge apparent in different infrastructure configurations, ranging from stand-alone application provisioning to diverse networking architectures and commercial cloud environments. It is, however, a crucial element in avoiding service level violations while operating in a, ubiquitous, cost-saving context. Understanding the workload characteristics and achieving predictable infrastructure performance for the available resources pool is crucial in achieving set goals. The mathematical science of Statistics is considered as a potential source of powerful tools for computer performance evaluation and related applications [5] and can be used in the server resource management context, enabling data-driven decision making.

Dimitris Kontoudis
Figure 1
Top of Figure



In this paper we present a novel auto-scaling resource management approach based on Statistics. The scaling mechanism utilizes Statistical Process Control (SPC) with the managed infrastructure being continuously monitored, having its performance patterns analyzed by applying the SPC methods. The involved resources are modified in a reactive manner, based on the outcome of the analysis, for bringing server performance within acceptable levels. Our approach is facilitated via a resource controller (titled *V-SPC*), which incorporates both the monitoring and the scaling components involved, and allows for the dynamic reconfiguration of hypervisor-provisioned Cloud infrastructure computing resources. We have implemented V-SPC on IBM pSeries AIX Unix infrastructure [6] using own developed code and executed a series of tests using real-world workloads based on the Temenos T24 Core Banking software [7], serving batch processes for a commercial financial institution. A thorough performance evaluation study based on the test results shows that the proposed approach has a ground foundation and practical applicability for actual environments.

The remainder of this paper is organized as follows: section 2 provides an overview of related work. In section 3 we describe the V-SPC architecture and the underlying theoretical foundation. Section 4 presents the design of the experimental implementation platform and the technologies used. The results of the approach are discussed in section 5. We conclude in section 6 by summarizing the findings of our research and outlining relevant future directions.

2. Related Work

The research introduced in this paper applies to local scheduling of virtualized resources [8] and fits under the autonomic resource management methods [9]. These approaches are of high importance as they deal with one of the very key problems [2] in the Cloud Computing context; they result in infrastructures that can have their resources dynamically modified as per the varying workload applied on them. Successful methods can have a significant positive financial and service impact on the involved infrastructure. Several such approaches to resource management in Cloud environments, based on different foundations, have been proposed. Control theory [10][11] has been used in single-tier application environments for adjustment of web server performance and guaranteed response [12], server cache management [13], virtual machine CPU and memory management [14][15]. In multi-tier application environments, [16] proposes an adaptive control system that can dynamically adjust the virtualized resources used in utility computing virtualized infrastructure. Resource allocation in cluster environments has been addressed in [17] as a constrained optimization problem focusing on performance isolation of resource allocations in individual machines and high utilization of server resources. In [18] a pre-computed scheduling graph is proposed for reserving CPU allocation and to implement guaranteed time constraints with accurate a priori feasibility analysis. Authors of [19] argue that the behavior of application workloads is an essential characteristic driving the resource allocation of datacenter resources. To model the resulting system performance patterns the authors propose a time–domain description of a generalized processor sharing server. A constrained optimization technique is, then, used to dynamically allocate the required resources based on the patterns observed. Work in [20] allows for dynamic allocation of the provisioned data center core cluster resources, based on the infrastructure's continuous energy consumption and relevant footprint. In the cloud computing and data center contexts, efforts are oriented primarily on CPU capacity management (as the core infrastructure resource), so as to meet QoS metrics and application related SLAs in effect. Statistical approaches, where present, are limited to virtual machine migrations and their suitable placement so as for taking advantage of overlapping/coexistence of workloads. This, in essence, is not an application of a statistical method - rather a best effort based on workload distribution on the infrastructure in question. In that direction, certain proposals rely on statistical multiplexing for consolidating and provisioning multiple virtual machines [21] or, in the same context, for modeling and improving the performance of the consolidated environments, based on the bandwidth resources managed by the hypervisor [22]. Statistical forecasting techniques have been proposed in the context of predicting future infrastructure workloads. These approaches allow for load demand profiling particularly in web applications. Time series analysis is used in [19] for estimating the workload characteristics based on requests flow. Queuing networks have been proposed in [23] and [24] for workload modeling of datacenter-hosted single and multi-tier applications. Proposals in [25] and [26] employ controllers in order to estimate the anticipated infrastructure load, under the generalization of the dynamic resource provisions problem as the cloud elasticity concept. An analytic survey detailing the aforementioned efforts is presented in [9], outlining relevant research and challenges faced.

SPC techniques have met application outside of the manufacturing domain and it has been observed that service and infrastructure related problems can benefit from such approaches [27]. Some efforts, of diverse focus, have been proposed in Computer Science and Information Technology. Authors of [28] have introduced a hybrid intelligent tool, *IntelliSPC*, which is an expert system based on neural networks. The tool was designed to allow for on-line data collection and processing, providing automated SPC in a controlled environment - demonstrating the use of hybrid artificial intelligence techniques in conjunction with SPC algorithms. The application of artificial neural networks has been introduced in [29] for pattern recognition in control charts, for automatically detecting non-random patterns. A hybrid system has been built and experiment results on the prototype implementation have demonstrated the effectiveness of the method. SPC has received considerable attention from the software engineering world, with a focus on software process monitoring and related metrics [30]. In the same domain, work in [31] introduces control charts for analyzing performance counters across test runs, in order to facilitate automatic performance regression testing in a software engineering environment. Results have been promising, showing that SPC can accurately identify performance regression in software systems. Proactive fault detection in computing infrastructures, using SPC, is investigated in [32]. Authors of [33] propose an intrusion detection system based on SPC. The system processes and analyzes audit events employing two detection techniques, one for anomalies and one for misuses. The application of SPC in the autonomic computing context, for application service quality, has been investigated in [34]. In [35] authors propose an expert system for the online construction of control charts. The proposed system does not require the human intervention of an SPC-skilled person. Finally, in computer networks, SPC has been proposed in [36] for determining the wireless channel behavior in IEEE 802.11 networks in order to facilitate better user centric management. In [37] authors apply multi-scale control charts in monitoring and analyzing communication on IEEE 802.11 and Bluetooth wireless systems, for effecting higher levels of quality of service (QoS).

2.1 Rationale of the proposed approach

SPC as a theoretical and practical foundation for application in managing infrastructure resources is driven by the fact that the technology is well-established in industrial application, which presents specific advantages in the computing server context (discussed below) *not found in other methods* [27][5].

The key differentiator of our proposal, as compared to other related work discussed, is the effective handling of virtualized managed elements of the target infrastructure via *leveraging the SPC advantages*. While several proposals exist in the resource management context, as presented in [9], the application of SPC to meet the same objectives offers substantial advantages, not found in related work. The key advantage of SPC is that *it focuses on a measurable process as a whole and not on any particular attributes of it*: the behavior of the process is the collective outcome of variation in any of the processes' associated attributes. Thus, monitoring and controlling the process helps to minimize this variability. Determining threshold limits in SPC is a robust, well-established procedure with a solid theoretical foundation and, most importantly, verified by decades of actual industrial application [38][39][30][27]. Related work presents a diverse array of techniques for scheduling the virtualized resources – no proposal, however, treats *as a single measurable attribute* the

impact that *different factors* have on process. Local allocation of resources is addressed as a result of a primary factor (CPU utilization, network latency etc.) and the techniques can be quite successful for handling *that factor only* [9]. The aforementioned key differentiator leads to additional individual positive repercussions. When SPC is applied properly, using the data gathered makes it easier to analyze the process in question – which attributes affect it and in what manner. Therefore, better process understanding is achieved – something not possible when one is focusing on a particular attribute. This is of the utmost importance in actual data center infrastructures as identifying root causes can be a challenging task. Applying this technology makes it possible to investigate a process in a top-down approach and, gradually, rule out causes of process variability – it is even possible, in cases, to detect problematic factors that were considered as normal and excluded from the focus of the analysis. These advantages have been leveraged for years in the manufacturing context, where SPC was first introduced, and are considerably mature and accepted approaches. Although properly designed experiments, such as those described in related proposals [9], can quickly resolve some of the relevant issues, an initial thorough examination of the SPC data can be valuable.

Another important advantage in applying SPC in a threshold-based rule reactive approach is that the latter, at their present form, cannot handle unexpected changes in the workload pattern – this being the main disadvantage of the techniques. The inherent analytical capabilities of SPC, though, provide indication on when a process is deviating from the normal down to a problematic state. It is, therefore, possible to eliminate or reduce infrastructure operation and quality issues before they appear. Variation in the measured process is reflected in the calculated thresholds and change in the workload patterns can be handled dynamically as long as the process pattern remains symmetrical. This is reflected in the normal distribution of the measured data. Related proposals cannot support such *foregoing* handling of the managed infrastructure unless they are combined with time series based methods applied for yielding predictive results [40] for the *anticipated* workload level. In the case of asymmetric process patterns SPC offers specific tools that can handle such cases. Although these tools are widely used in industrial applications, their applicability in the resource management context and in our proposed approach has not been verified – it constitutes an area of further research.

Overall, SPC can be considered as an embracing approach which allows for rapid assessment and maintenance of a process' quality without the need of investigating and regulating each individual process attribute. These attributes do exist and influence the final outcome, as properly suggested in related proposals – however, the statistical analysis of the process as a whole remains consistent. Based on the aforementioned facts, and in light of the fact that SPC has not been applied before in this direction, we argue that this technology can provide the basis for a new resource management approach in the datacenter context. The results of our research indicate that SPC has solid application ground.

2.2 Comparison to existing related proposals

Our proposal as compared to the aforementioned resource management work differs, apart from the underlying mathematical foundation, in several other aspects. From a managed infrastructure element perspective, related work can be grouped into proposals targeted on the application/middleware layer ([10][11][12][13][24]), on single virtual machines (VMs) ([15][16][18]), on clustered VMs ([17][19][21]), on specific technical characteristics such as energy profiling [20] or rack-wide network bandwidth [22], and on data center scale VM

mobility and provisioning ([23][25][26]). All these proposals have a key goal: that of achieving the performance requirements expected from the managed infrastructure. To meet the set performance goals each proposal employs different metrics for monitoring the infrastructure and for triggering resource provisioning actions. Our work focuses on *self-adaptation of single or clustered VMs* and is mostly concerned with the *CPU and memory utilization* along with the virtualized infrastructure's dynamic resource allocation *adaptation time*, as the key metrics involved in the calculations (detailed in section 5.1). Our technique can complement others relevant to local resource management, especially those focusing on broader infrastructure behavior, such as Cloud local and geographical elasticity ([22][23][25][26]) as well as energy considerations when related to the performance of VMs ([20]). It is possible for our work to be incorporated in the aforementioned frameworks in order to support and facilitate local management decisions. However, we do not focus on Cloud services and their auto-scaling when provisioned locally or when dispersed over different Clouds (e.g. as in [41] and [42] - this is a different area of research [9]).

Approach	Foundation	Target Layer	Validation Data	Metrics
[10]	Control theory	Application	Synthetic	Transactions/s, no. of users
[11]	Control theory	Application	Synthetic	HTTP requests, TCP connection delay
[12]	Control theory	Application	Synthetic	Web server request rate, system resources utilization
[13]	Control theory	Application	Synthetic, Empirical	Proxy server request rate, cache storage utilization
[15]	Control theory	Single VM	Synthetic	VM virtual clock time
[16]	Control theory	Single VM	Synthetic	CPU {utilization, cycles}
[17]	Constrained optimization	Clustered VMs	Synthetic	Web server request rate, cluster capacity utilization
[18]	Graph theory	Single VM	Synthetic	Execution duration, CPU reservation time
[19]	Time-series analysis, Non-linear optimization	Clustered VMs	Synthetic, Trace	No./rate of HTTP requests
[20]	Greedy algorithm	Energy	Synthetic	TCP connections, Server power draw
[21]	Statistical multiplexing, Time-series analysis	Clustered VMs	Real	VM capacity and utilization
[22]	Statistical multiplexing, Stochastic Bin Packaging	Network	Synthetic	Network bandwidth
[23]	Queuing theory, Statistical forecasting	Data Center	Synthetic	Server {response time, throughput}
[24]	Queuing theory	Application	Synthetic	Application request rate
[25]	Queuing theory	Data Center	Trace	Service request rate
[26]	Wavelet functions	Data Center	Trace	Service request {rate, type}
Our solution	*Statistics (Statistical Process Control)*	*Single VMs*	*Real*	*CPU, memory utilization, adaptation time*

Table 1. Overview of related resource management proposals

Existing works that have *a direct relevance to our proposal*, pertaining to *local (vertical) scheduling of virtualized resources*, are outlined in Table 1. Control theory-based approaches

([10], [11], [12], [13], [15], [16]) allow for systematic design of the adaptive infrastructure using analytical methods. This is in contrast with other ad-hoc resource management that requires extensive, sometimes iterative, tuning of the controller environment. There are challenges in applying control theory to computing infrastructure, such as developing effective resource models, handling sensor delays, and addressing lead times in effector actions. However, the major advantage of these approaches is that they offer performance guarantees, derived from established and well-understood theoretical foundations, in environments with varying, not a-priori known, behavior. Our SPC-based approach, before deployment, requires tuning as per the target infrastructure environment and does not offer performance guarantees. The method, however, does not require accurate or available historical resource usage measurements. The controller, post-deployment, is fully automated and aware of application SLOs, requiring recalibration only in cases of major (architectural) infrastructure modifications. Statistical multiplexing techniques are employed in [21] and [22] whereby resource allocation is commensurate with incoming demand. Contrary to other methods where VMs are treated in isolation, statistical techniques allow for joint-VM provisioning and resource management. In this situation the workload patterns of each managed entirety do not, necessarily, coincide. The benefit of these methods is that capacity can be saved by consolidating workloads. In normal usage scenarios the provisioned resources are not exploited to their entirety thus, by workload multiplexing, resource provisioning is optimized. Other approaches to dealing with this problem present scalability limitations and cannot easily address the problem of workloads with different utilization characteristics. Queuing theory [24], [25] and wavelet functions [26] particularly address multi-tier infrastructure architectures and the analytical modeling of the behavior of applications therein. These approaches are important as they allow, apart from capacity provisioning, for performance prediction in tiers with varying workload characteristics. The main benefactor of such use is horizontal elasticity in cloud environments via the addition or removal of virtual machines. Our proposed approach does not target horizontal elasticity but intra-VM resource management, which can be employed in general elasticity frameworks. In our work, hypervisor-provisioned resources are monitored and managed and this functionality can be one of the available options to the end-to-end process-centric resource control systems in cloud environments. Our controller can neither deal with resource contention between multiple VMs nor achieve a desired level of performance differentiation when not in isolation in a particular VM.

The technique proposed in this paper is a threshold-based rule reactive approach, employing static thresholds that form the basis for the decisions for the initiation of elasticity actions. The thresholds are calculated via Statistical Process Control and each infrastructure performance metric (discussed in 5.1), as measured in real time, uses three thresholds for comparison purposes (consecutive off-limit outliers, above or below the control limits, for CPU and/or memory utilization). Similar approaches, using static thresholds, are used by Rightscale [82], Amazon [88] and other works [81][85][86][87]. An exhaustive list of these approaches can be found in [70]. These approaches, with the Rightscale algorithm being a prominent example, allow for direct manipulation of the hardware resources in order to scale the infrastructure horizontally, should certain conditions are triggered. Our proposed method is similar in concept, however, it is used for local (vertical) scaling, allowing for elasticity on a virtual machine's core resources (CPU, memory, etc.) and not for increasing or decreasing the number of available VMs in a collective cluster (spawning or decommissioning VMs). The high-level approach we follow is similar to Rightscale's auto-scaling in that the method allows for provisioning or decommissioning resources automatically based upon different observed predefined conditions. As an example, in Rightscale, additional servers (virtual

machines) can be spawned in the infrastructure when the current active servers are over-utilized (e.g. idle CPU utilization is less than 30% for 15 minutes or longer). In reverse, provisioned servers can be removed when utilization decreases. Likewise, our approach allows for adding or removing resources *in* a server when similar conditions are triggered (e.g. CPU utilization higher than a certain threshold, for more than three consecutive observations). Rightscale's approach uses a voting process [84] for deciding whether to expand or shrink *an array* of virtual machines. The array is monitored for certain metrics (voting tags [83]) and, based on the number of occurrences it is determined whether a particular scaling action, *affecting the entire array*, is needed. Rightscale's approach which uses a combination of a voting proposal along with basic resource utilization rules, extends the traditional reactive auto-scaling logic (simple rule usage) and has been adopted in a few proposals ([43]). These proposals incorporate extended conditions for scaling decisions, based on the voting logic, such as number of active sessions and different scaling policies based on workload characteristics. Our approach uses Statistical Process Control for determining whether events monitored in a particular server justify the scaling action *for that server only*. Additionally, both Rightscale's algorithm and our proposal suffer from the (common in rule-based approaches [40]) problem of threshold definition. The former requires user-defined input whereas our approach relies on the SPC generated thresholds. Finally, both Rightscale and our approach account for avoiding oscillations (instability due to infrastructure reconfigurations) in the system operation. To that end, Rightscale introduces, after each scaling action, a time period ("*resize calm time*" – recommended at 15 minutes), during which no other reconfiguration action is performed. Our approach makes use of a "*decision threshhold*" metric (discussed in 5.2), which determines the number of consecutive outliers, before a scaling action is started. Both approaches aim at maintaining infrastructure stability.

Infrastructure management (resource allocations and re-allocations) embody latency and hypervisor/virtualization vendors seek to optimize (minimize) the delays inherent in such operations. This latency, also referred to as "*virtualization overhead*" depends on the type and characteristics of the virtualization and its implementation specifics [71]. A main drawback of reactive auto-scaling approaches is their inability to account for future, therefore unexpected, changes in the workloads. The outcome of this is further hindered due to the virtualization overhead in resource operations - leading delays in infrastructure adaptation and possible instability in the system (oscillations, poor performance etc.) From a research perspective, there are efforts in providing efficient prediction algorithms that could complement the latency issue, such as those based on control theory ([10][11][12][13][15][16]), queuing theory ([24][25]) and, typically, time-series analysis as manifested via machine learning [73][74][75][76][77], moving average (MA) [78][79][80], auto regression (AR) [60][63][64][65][66], ARMA (combination of MA and AR) [60][61][62] and other methods ([19][21][22][23]). Time-series analysis methods focus on the change of a measurement of a quantity over time and are used either for prediction of future values or for the identification of patterns in the measured data. Both applications of the methods enable the handling of future conditions based on forecasting [40]. Effectiveness of the methods heavily depends on the quality of the forecasts, with each method (regression, neural networks etc.) yielding different levels of accuracy. Additionally, neural network based methods, apart from the prediction accuracy of the model, will require an effective training period based on fed data from past infrastructure utilization [69]. Predictive analysis of resource usage is of major importance not only in the discussed context but, also, in other design and operational decisions such as capacity planning, infrastructure sizing etc.

As compared to predictive approaches, our technique is based on online workload characterization and does not provide prediction mechanisms. It can, however, be combined with a predictive approach, as part of a general management solution, in order to facilitate forecasting *and* actions when the resource capacity of the managed infrastructure will be nearing its limits, as per the *anticipated* workload pattern and/or utilization – therefore, this option can be used as a failsafe mechanism in order to avoid resource exhaustion, something that would result in failure of any resource management operation. In this direction, the auto-scaling problem can be approached as a two-phased situation: the first phase involves the prediction of future infrastructure state, based on the time-series (or other forecasting methods) analysis of current measurements [67][72]. The forecasting results are, consequently, used in the second phase which involves making the scaling decisions, modifying the infrastructure accordingly. Our solution will benefit by forecasting techniques as scaling the infrastructure as per the incoming demand is key to meeting SLAs. If scaling does not happen within specific response time limits (not prematurely nor late) then it is possible that the adaptation will fail, breaking SLAs in effect. Such hybrid solutions constitute an ongoing research effort and challenge as, any solution, will have to account for the advantages as well as for the drawbacks of both approaches [68][77]. Example applications of hybrid solutions, as applied to CPU scheduling, are addressed in [17][18][19][20][25] in the context of real-time resource reservation and negotiation for single or clustered VMs, combining on-line measurements with prediction and allocation techniques. Time series analysis and optimization approaches are used for predicting expected workload patterns from measured infrastructure metrics. The advantage of these techniques is that they capture the transient behavior of applications while incorporating nonlinearity in the system model. The proposed SPC method provides a balance between static and/or unspecified workload demand and the automated allocation of resources after the demand profiling (tuning) for the controller environment.

Threshold-based rule reactive approaches lack a formal evaluation methodology of their effectiveness (including the absence of a commonly accepted scoring metric) and, indeed, appear simple in their design with the most difficult part being that of determining the proper threshold limits as per the distinct characteristics of individual workloads [40]. This lack of verification formality hardens the task of objectively assessing our proposal's effectiveness, as compared to other approaches. One overall factor, yet hard to quantify, is the level of difficulty inherent in applying the approach. This fact has been reported in literature in the context of resource management in cloud environments [43][8]. Rule-based approaches, as compared to control theoretic ones, are easier to implement, simple per se and convenient provided that the user can correctly determine the scaling indicators and the required performance goals. In essence, if the workload characteristics are known then a rule-based method is easier to adopt. We have verified this fact given that a well-known feature of SPC is its ability to facilitate rapid prototyping while maintaining the robustness of the mathematical formulae for the determination of the involved thresholds. As part of the verification approach, we proceeded via application with real data and data center virtualized infrastructure running a commercial batch workload, in an attempt to yield actual results. All related work discussed were validated through experiment and simulation and, except for proposal [21], did not use real data in the experimental implementation; proposals [13][19][25] and [26], in part, used trace data from past Internet portals. We have, thus, provided a first proof of suitability of our proposal for application on production infrastructures. Further comparison between our and the related proposals, based on specific technical or other attributes, is not feasible. The lack of a formal testing and comparison framework along with the fact that each author focused on different Cloud architecture

directions and used totally diverse infrastructures, metrics, testing scenarios, workloads etc. harshen direct comparative assessments; a problem already faced in the auto-scaling in the Cloud context [40].

3. The V-SPC Architecture

Resource management in the Cloud Computing datacenter context refers to the process of allocating particular computing resources in order to meet the performance requirements of hosted application workloads. The underline mechanism for determining the proper resource allocation strategy may rely on varying foundations and can be deployed for different resource types. Our proposed approach employs SPC and focuses on the computing system CPU and memory resources.

3.1 Theoretic Framework

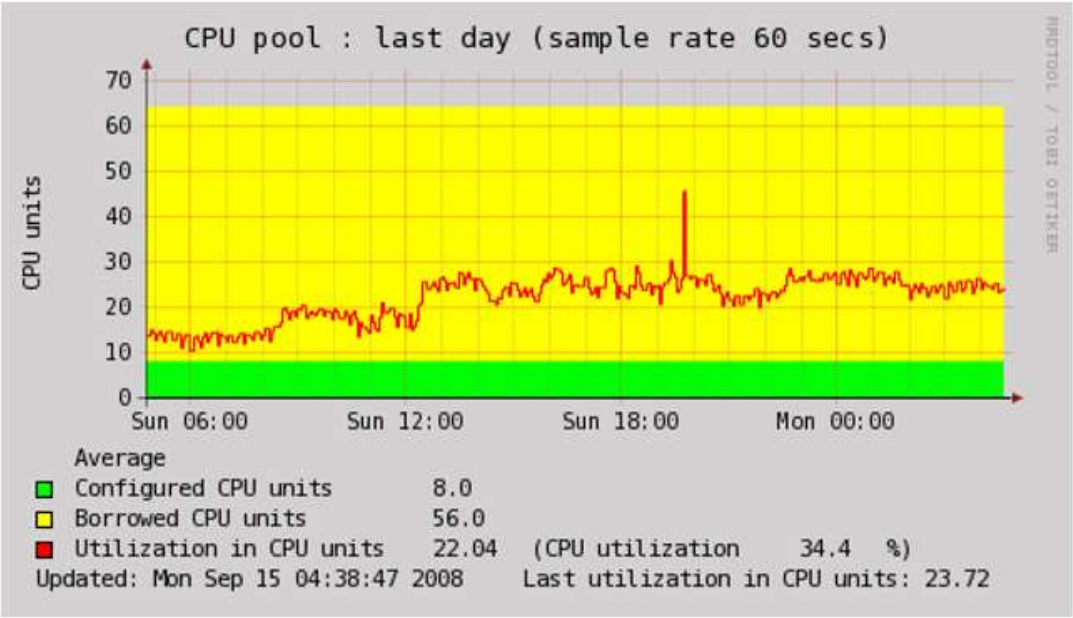
SPC is a term, collectively used, for specific analytical methods which employ statistical techniques to the monitoring and control of a process, in order to ensure that it operates within designed limits for acceptable output. Under SPC, a process behaves predictably to produce as much conforming product as possible with the least possible waste. While SPC has been applied most frequently to controlling product manufacturing lines, it applies equally well to any process with a measurable output. SPC indicates when an action should be taken in a process, but it also indicates when no action should be taken – the latter in the case when the process is behaving as expected. The reader is referred to [38] for introduction and detailed discussion on SPC. The term process, as used in the context of this paper, refers to the processor or memory utilization of a virtual machine (VM) – hence, undesired process levels may result in impediments in the service offered by the infrastructure. In this direction the problem of using SPC to manage the VM's performance can be described, in general terms, as follows. Let **X** represent the VM's resource utilization while executing under a particular payload. The distribution function of **X** is indexed by the feature vector **V**, collectively reflecting a number of different factors (type of workload, processor parameters, system tuning or other workloads running on the same infrastructure etc.) In SPC terminology, the process **X** is operating at a stable state when $\mathbf{V}=\mathbf{V_0}$ and is referred to as being in-control. $\mathbf{V_0}$ is the baseline for successful VM operation against which comparisons can be made. Obviously, there is no ubiquitous accepted value for $\mathbf{V_0}$ – this is case-based and will depend on the particular problem in question. Therefore, variations in **V** against the baseline $\mathbf{V_0}$ will be reflected as variations in **X** and will allow us to determine when an action should be taken in order to bring the process back in-control.

It has become apparent in recent data center implementations that, since most modern system and network infrastructure implementations are virtualized, the virtualization overhead imposed by the hypervisor layer (present in all parts of the infrastructure, i.e. server, network, storage etc.) needs to be assessed and evaluated [44][45]. The very use of virtualization technology introduces certain facts that need to be addressed as traditional approaches to resource management may not apply, or, not be optimal for use in the virtualized landscape. From the resource management perspective it is crucial that, apart from the virtualization overhead, two other information facts are taken into account in any management foundation: i) the available (free) resources, and ii) the resource utilization patterns imposed by the workloads running on the virtualized infrastructure. These points of interest, along with any other that affect the VM's performance, are all reflected in the vector **V** where, by

reciprocation, the $\mathbf{V_0}$ state is the accepted state of all resources, indicated by VM resource utilization within the defined control limits. Our proposed approach allows adaptation to changes in the infrastructure as these are, ultimately, reflected in the resource utilization. By employing control charting analysis on the VM resource utilization patterns it is possible to identify variation in the data and react appropriately.

Key tools in SPC are the Shewhart Control Charts [46] (also referred to as $\bar{\bar{X}}$–S or $\bar{\bar{X}}$– R charts) which can be used for on-line monitoring of processes to show how they are performing over time and how their capabilities are affected by changes introduced to the process. This information is then used to make quality improvements. Control charts are also used to determine the capacity of the process by helping identify special or assignable causes for factors that impede peak performance. Control charts rely on well set control limits. These must be defined in such way that the value of $\mathbf{X}$ will be unlikely to fall outside of them when the VM is performing normally, at acceptable service levels (when $\mathbf{V}=\mathbf{V_0}$). VM operation can be managed by the ongoing control of the process $\mathbf{X}$, essentially by monitoring resource utilization variation over time and detecting anomalies. These will occur as a result of unwanted deviations of V from the established baseline $\mathbf{V_0}$. When this occurs management action is justified and corrective procedures can be applied. As a result, VM operation quality, resulting cost and effective capacity can be optimized. Additionally, retrospective assessment or prospective capacity planning by extrapolation are possible based on the analysis of the monitoring data. Evaluating the control limits is a procedure that involves determining certain statistical attributes in the recorded data, hence, of the observed workload. Daily server operation varies considerably and resource utilization graphs of a business or cloud virtualized server running a typical workload over a 24 hour period usually exhibit a spiked spread, where the server's resource is consumed at lower or higher levels running the time-varying workload (Fig. 2 indicatively showing VM CPU utilization over time) [47]. In heavily consolidated servers, where each hosts several workloads, the utilization graph may appear smoother with a flat, sustained, utilization; still, spikes do exist and can be revealed if a smaller monitoring sampling interval is used.

Dimitris Kontoudis
Figure 2
Top of Figure



Whichever the case, the overall spread follows a random pattern with a standard normal distribution characterized by specific statistical attributes expressed by standard Statistics or SPC textbook formulae [48]:

a. the Mean, $\bar{\boldsymbol{X}}$, refers to the average resource utilization measured granularly over a time period, formulated by

$$\bar{\boldsymbol{X}} = \frac{\boldsymbol{x_1 + x_2 + \ldots + x_n}}{\boldsymbol{n}} = \frac{\sum_{i=1}^{n} \boldsymbol{x_i}}{\boldsymbol{n}} \tag{1}$$

where x_n denotes each resource utilization measurement (sample). Similarly, the average of all sample means, $\bar{\bar{\boldsymbol{X}}}$, for several measurement time periods, is the average of each period's calculated mean value $\bar{\boldsymbol{X}}$, formulated by

$$\bar{\bar{\boldsymbol{X}}} = \frac{\bar{\boldsymbol{X}}_1 + \bar{\boldsymbol{X}}_2 + \ldots + \bar{\boldsymbol{X}}_k}{\boldsymbol{k}} = \frac{\sum_{j=1}^{k} \bar{\boldsymbol{X}}_j}{\boldsymbol{k}} \tag{2}$$

b. the Range, $\boldsymbol{R}$, is the difference between the highest and lowest recorded resource utilization samples across the time period under measurement. The Mean Range, $\bar{\boldsymbol{R}}$, is the average of the ranges calculated for several distinct measurement time periods, formulated by

$$\bar{\boldsymbol{R}} = \frac{\boldsymbol{R_1 + R_2 + \ldots + R_k}}{\boldsymbol{k}} = \frac{\sum_{i=1}^{k} \boldsymbol{R_i}}{\boldsymbol{k}} \tag{3}$$

c. the Standard Deviation, $\boldsymbol{\sigma}$, is a measure of how widely the intervals differ from the mean $\bar{\boldsymbol{X}}$, formulated by

$$\boldsymbol{\sigma} = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{X})^2}{n-1}} \quad (4)$$

The average sample standard deviation, $\bar{\boldsymbol{s}}$, is the average of the respective values calculated for several measurement time periods:

$$\bar{\boldsymbol{s}} = \frac{\boldsymbol{\sigma_1 + \sigma_2 + \ldots + \sigma_n}}{\boldsymbol{n}} = \frac{\sum_{i=1}^{n}\boldsymbol{\sigma_i}}{\boldsymbol{n}} \quad (5)$$

The normal distribution for the resource utilization of a virtual server running a specific workload will be consistent across time frames, as long as the workload pattern does not change nor there are parameters that affect the server's normal operation. Should the latter occurs, a different distribution will be observed and the root causes will be manifested as different utilization measurements and variances. In the absence of extraordinary conditions the distribution pattern of the observed utilization data will exhibit a majority 95% population of recorded values at less than a $\boldsymbol{3\times\sigma}$ difference from the calculated mean $\bar{\boldsymbol{X}}$ (the condition often referred to as the 68%-95%-99% rule [47]). Hence, the appearance of resource utilization at levels greater than $\boldsymbol{3\times\sigma}$ might indicate a possible error state in the infrastructure. Before any SPC method can be applied, it is necessary to setup the proper control limits. Given that the number of recorded measurements and data points produced when monitoring VM resource utilization is, usually, quite large (in the range of hundreds or thousands per measurement period) the $\bar{X}$–S chart is the appropriate tool for this data processing as, for measurement data sets of more than 12 values, the $\bar{X}$–R chart is inefficient in properly handling the range ($\boldsymbol{R}$ and $\bar{\boldsymbol{R}}$) [48]. The control limits for the $\bar{X}$–S chart are calculated as follows:

d. the Center Line, ***CL***, the Upper Control Limit, ***UCL***, and the Lower Control Limit, ***LCL***, respectively, for the $\bar{X}$ and the *S* charts, by

$$\boldsymbol{CL} = \bar{\bar{\boldsymbol{X}}}\,,\ \boldsymbol{CL} = \bar{\boldsymbol{s}} \quad (6)$$

$$\boldsymbol{UCL} = \bar{\bar{\boldsymbol{X}}} + \boldsymbol{A_3}\bar{\boldsymbol{s}}\,,\ \boldsymbol{UCL} = \boldsymbol{B_3}\bar{\boldsymbol{s}} \quad (7)$$

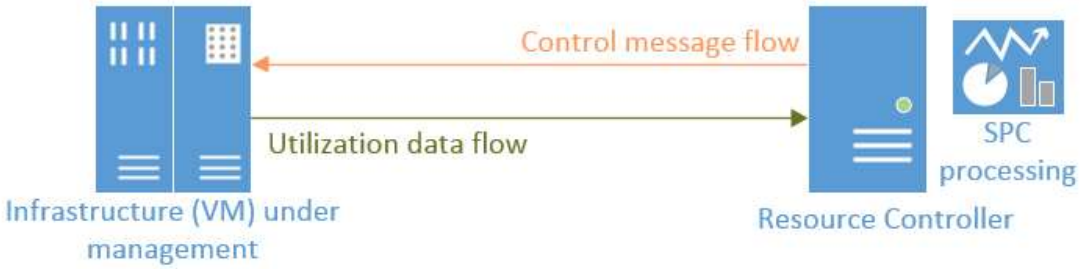
$$\boldsymbol{LCL} = \bar{\bar{\boldsymbol{X}}} - \boldsymbol{A_3}\bar{\boldsymbol{s}}\,,\ \boldsymbol{LCL} = \boldsymbol{B_4}\bar{\boldsymbol{s}} \quad (8)$$

where $\boldsymbol{A_3}, \boldsymbol{B_3}, \boldsymbol{B_4}$ are constants used for the construction of the control chart.

3.2 Design

We propose a resource controller (Fig. 3) that incorporates the logic for applying SPC on VM-based datacenter infrastructure. The controller processes the resource utilization data and determines whether corrective actions are needed. Should this is the case, the latter are applied on the VM in order to bring the environment back in-control, under accepted performance parameters. Operation is performed in an online mode with the controller receiving and processing utilization data at a continuous rate during the desired management period.

Dimitris Kontoudis
Figure 3
Top of Figure



Managing the infrastructure using the V-SPC controller involves three distinct phases: *i)* preprocessing, *ii)* base-lining, and *iii)* online management. The purpose of the first phase is to collect the required initial utilization data sets from the infrastructure in question and prepare this data so as to be suitable for statistical processing. Any extraordinary performance data which can be attributed to known causes needs to be removed. The second phase performs the data analysis and produces a baseline, a depiction of the infrastructure's normal state of operation. During this phase the statistical attributes of the supplied data sets are determined (the population mean, the standard deviation and the associated upper and lower control limits). Once the performance baseline has been obtained, as this is reflected by the statistical attributes, the management phase can commence. During this phase, a continuous flow (Fig. 3) of utilization data is streamed to the resource controller which, in turn, applies the SPC logic and reverts back with the appropriate infrastructure reconfiguration commands. The control loop receives the current VM processor or memory utilization and compares it with the actual boundaries computed by the SPC processing. Based on the analysis of the observed behavior the controller dynamically adjusts the VM's resource allocation in response to changes in the workload. The control loop executes continuously for the entire duration of the desired management period. The resource controller implements an "average of all data set means" based approach ($\bar{X}$–S) but can be easily modified or extended to realize other workload models. The high-level algorithm for the facilitation of the three phases is presented in Table 2.

Step	Phase
1. Collect performance utilization data sets for a number of normal infrastructure usage periods (to be used for calculating each individual sample period mean $\bar{\boldsymbol{X}}$ and the average of all sample means $\bar{\bar{\boldsymbol{X}}}$)	preprocessing
2. Transfer recorded data sets to the resource controller system	

3. Normalize performance data
For each data set *(1..n):*
If data points attributed to known causes exist, then
remove those data points
End If	

4. Baseline normal VM operation by calculating the required statistical attributes in the collected data ***For*** *each data set (1..n):* Compute the mean, $\bar{\boldsymbol{X}}$, using equation (1) Compute the standard deviation, $\boldsymbol{\sigma}$, using equation (4)	base-lining

End For	
5. Compute the average of all sample means, $\bar{\bar{X}}$, using equation (2) and the average sample standard deviation, $\bar{s}$, using equation (5)	
6. Construct the $\bar{X}$**–*S*** control chart and compute the Upper Control Limit (**UCL**) and the Lower Control Limit (**LCL**), using equations (7) and (8) respectively	
7. Monitor (VM CPU/memory utilization) and manage the infrastructure resource	online management
8. *Occurrences_Counter_High* = 0	
9. *Occurrences_Counter_Low* = 0	
10. *In_control = True*	
11. ***Repeat*** until end of management period	
Retrieve CPU/memory resource utilization from the VM	
While *In_control = True*	
If *resource utilization* > *UCL*, **then**	
Increase *Occurrences_Counter_High* by one	
End If	
If *resource utilization* is < *LCL*, **then**	
Increase *Occurrences_Counter _Low* by one	
End If	
If {*Occurrences_Counter_High* OR *Occurrences_Counter_Low****}*** > 3 , **then**	
In_control = False	
End If	
End While	
If *Occurrences_Counter_High = 3*, **then**	
Send control message to increase VM resource allocation by one unit	
Occurrences_Counter_High = 0	
End If	
If *Occurrences_Counter_Low* = 3, **then**	
Send control message to decrease VM resource allocation by one unit	
Occurrences_Counter_Low = 0	
End If	
In_control=True	
End Repeat	

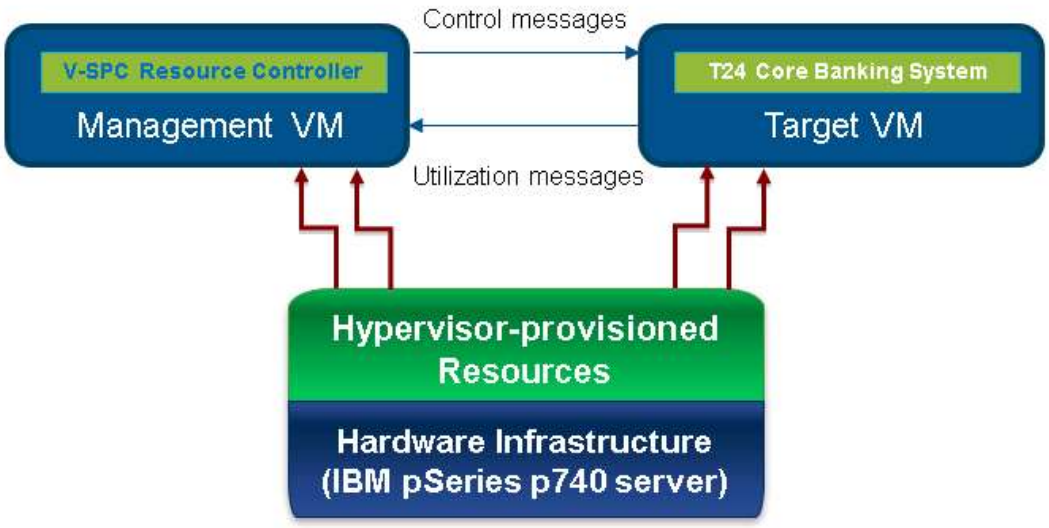
Table 2. Algorithm for the V-SPC infrastructure management

4. Experimental Implementation

We implemented an experimental technical architecture for showcasing the proposed concept, focusing on a realistic assessment based on real-life and not synthetic data. To that end we used the Temenos T24 Core Banking System software [7] with actual business data as the application environment running on the managed VM. The test setup consists of an IBM pSeries 740 UNIX server divided into two logical partitions (LPARs - VMs), each VM running IBM's AIX operating system version 7.1 [6]. Employing the hypervisor capabilities, each VM was assigned a certain amount of p7 CPUs and GBs of RAM memory along with a virtual Ethernet adapter realized via the in-memory IEEE 802.1Q virtual switch [49]. One of the VMs was selected as the target system to be managed by V-SPC. On this system we installed a copy of the core banking software. On the second VM we developed and installed appropriate code (KornShell93 scripts and C binaries) that implements the resource controller functionality. This VM, additionally, serves as an aggregation system, collecting the performance information from the managed VM, gathering utilization patterns and producing consolidated statistics of the core resources provisioned by the hypervisor. Operating system specific monitoring tools gather the performance related information (*sar* and *vmstat* commands) and SSH DLPAR operations, via the p740 system's Hardware Management

Console (HMC), provide for the dynamic reconfiguration of resources, as per the control messages sent by the controller. A high-level illustration of the experimental implementation infrastructure is shown in Fig. 4.

Dimitris Kontoudis
Figure 4
Top of Figure



All data flow between different components is based on messages structured in XML, a mature and widely accepted standard [50]. Apart from the well-established benefits of the language, XML representation increases the possibility of using the created messages in a wider array of management (or other) applications. The target VM is continuously monitored and its utilization status is reported to the resource controller via suitable messages (space delimited values of primary statistics). The information received is statistically processed by the controller and, if proved necessary, infrastructure-related decisions are made. These are instantiated via control messages and infrastructure reconfiguration commands issued by the controller. Table 3 shows a brief example of such events sequence: the controller receives a resource utilization message containing several attributes of resources state at a particular point in time. This data is routed to the SPC algorithms and processed against the previously calculated baselines. A decision is made to increase the resource capacity of the managed VM by one unit. The necessary infrastructure reconfiguration command (essentially, KornShell93 or C executable code) is constructed and executed by the resource controller. The utilization messages as well as the reconfiguration commands are platform specific and depend on the deployed hypervisor and operating system used.

Resource utilization message	Infrastructure reconfiguration command

<MESSAGE ID="2368"> <NAME="ProcUtil"> </NAME> <VALUE=” 13:37:08 %user %sys %wait %idle physc %entc lbusy app vcsw phint 47.1 20.6 2.3 29.9 8.15 108.7 24.2 1.32 11562 826 ”> </VALUE> </MESSAGE>	FRAME=Server1-SN06C0121 HMC=hmc1 PROC_TO_ADD=1 VPROC_TO_ADD=1 NODE_TO_ADD=SRVCBP1 ssh hscroot@${HMC} "chhwres -r proc -o a -p $NODE_TO_ADD -m $FRAME --procunits $PROC_TO_ADD --procs $VPROC_TO_ADD -w 1"

Table 3. Example of V-SPC infrastructure management messages and commands

In order to provide description standardization of the V-SPC architecture we modelled it using the Common Information Model (CIM) proposed by the Distributed Management Task Force (DMTF) [51] and the KF model [52]. We followed an object-oriented approach for describing the management entities in our test environment. These models are not bound to any particular implementation and, thus, enable the platform-independent and technology-neutral exchange of management information, providing a consistent definition and structure of data. Given the standardization followed it becomes easier to port the IBM AIX system specific implementation to other operating system platforms.

The choice of the workload and the virtualization architecture is typical of those found in commercial public and private Cloud datacenters (Internet Service Providers, Cloud Operators and/or commercial environments [53][8][54]). The workload used in the experiment consisted of T24 core banking jobs processing a data set, as structured for a typical nightly batch processing procedure [7]. The total data volume was sized at 500GB of storage area network provisioned capacity, of which 240GB were consumed by the business data. Each test run included the same, resource-intensive, jobs executed sequentially, indicatively:

- SYSTEM.END.OF.DAY5-EOD.UPDATE.CATEG.ENT.MONTH
- FILE.TIDY.UP-EB.CLEAR.FILES
- FT.START.OF.DAY-FT.LOCAL.DATA.PURGE
- REPORT.PRINT.REPORTING-EB.EOD.REPORT.PRINT
- UPD.DEL.CYCLES-EXTRACT.WRONG.BKTS
- RE.UPDATE.SLC.LINE.BAL-RE.UPDT.STAT.LINE.CONT

Each of the involved jobs executed during the test runs created a different stress level on the allocated processors and resulted in varying CPU and memory utilization. Given that the data under processing remained static for each test run, the overall execution time for the entire set of jobs depended solely on the number of the allocated resources – in our case, on the number of allocated processors and gigabytes of RAM memory.

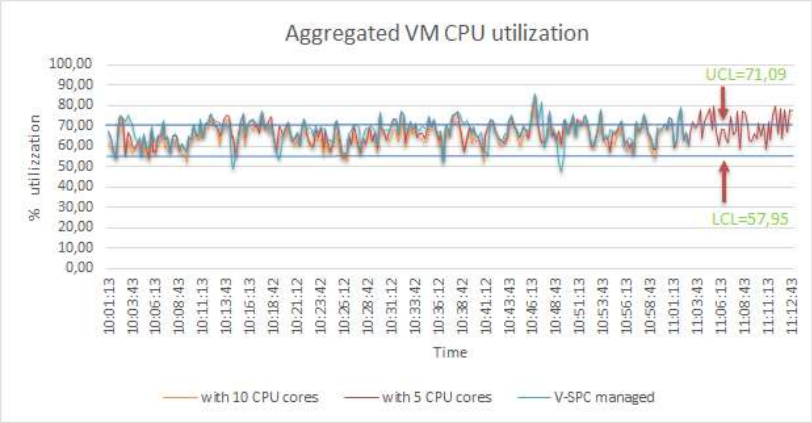
4.1 Test runs

During the preprocessing phase we run five times a T24 batch processing procedure on the target VM, using typical business data of a core banking environment, resulting in similar and anticipated *workload patterns* for each run. The VM’s processor allocation was initially capped at 10 CPU units and 20 GBs of RAM memory, which resulted in a run window of one

hour for servicing the particular workload. This is considered as the acceptable process output (baseline duration) for the given experiment, under a static resource allocation environment. The VM was monitored during each run and detailed performance information was collected, resulting in the creation of five distinct performance data sets. The performance data was normalized and the data points that could be attributed to known causes were removed. To that end data from, approximately, one minute duration at the start and finish times of each run (regions 10:00:00-10:00:58 and 11:00:13-11:01:00) was deleted as it corresponds to VM utilization at stages where no workload-related processing occurred.

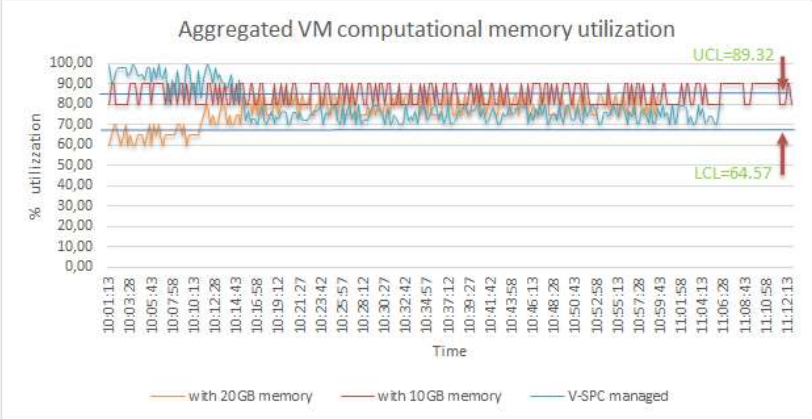
Following data collection and preprocessing, the statistical representation of the accepted baseline was established, by calculating certain statistical attributes of the collected data sets ($\bar{X}$, $\bar{\bar{X}}$, σ, $\bar{s}$, CL, UCL, LCL). Control charting for CPU utilization yielded the values of UCL=71.09 and LCL=57.95. Respectively, for memory utilization the values were UCL=89.32 and LCL=64.57. These values were consecutively used for the next stages of the experiment. The VM's processor and memory allocation was decreased, by 50% (5 CPU units and 10GB RAM), so as to lead to processor and memory starvation for the applied workload and to force triggering of corrective procedures. Two new test runs, for each resource type (hence four runs in total) were initiated on the decreased configuration VM: one without any external management action and one with the V-SPC resource controller actively managing the VM, providing a dynamic resource allocation environment. Performance data was collected for each run. For the CPU resource, the run on the unmanaged VM, using half of the original CPU capacity resulted in an extended duration of one hour and 12 minutes, therefore increased by 20% as compared to the acceptable process duration of one hour. The run on the V-SPC managed VM resulted in a duration of one hour and 2.5 minutes, nearly similar to the acceptable one, with the VM's CPU allocation stabilized at eight units. The aggregated graphical results of the experiment are plotted in Fig. 5.

Dimitris Kontoudis
Figure 5
Top of figure

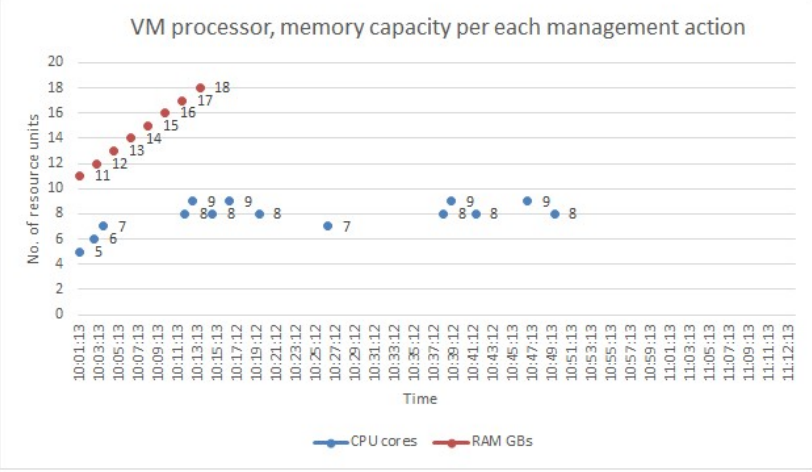


In memory test runs it is the computational memory that is of importance (i.e. the actual memory amount that the workload consumes – further discussed in section 5.1) and not the total amount of physical RAM available to the VM. In the case of the unmanaged VM workload, using half of the original RAM allocation, execution resulted in a duration overhead of 28 minutes, therefore increased by 46.6% as compared to the acceptable process duration. With the resource controller active managing the VM, the resulting duration exceeded the anticipated one by 6.5 minutes, whereby the system memory reached a sustained capacity of 18 GB. The resulting utilization patterns are shown in Fig. 6. The graphs across different runs show common patterns due to the fact that the exact source application data set was used, unaltered, for each run. The resulting quantities of CPU and memory capacity modifications, as performed by the V-SPC controller, are shown in Fig. 7.

Dimitris Kontoudis
Figure 6
Top of figure



Dimitris Kontoudis
Figure 7
Top of figure



5. Discussion

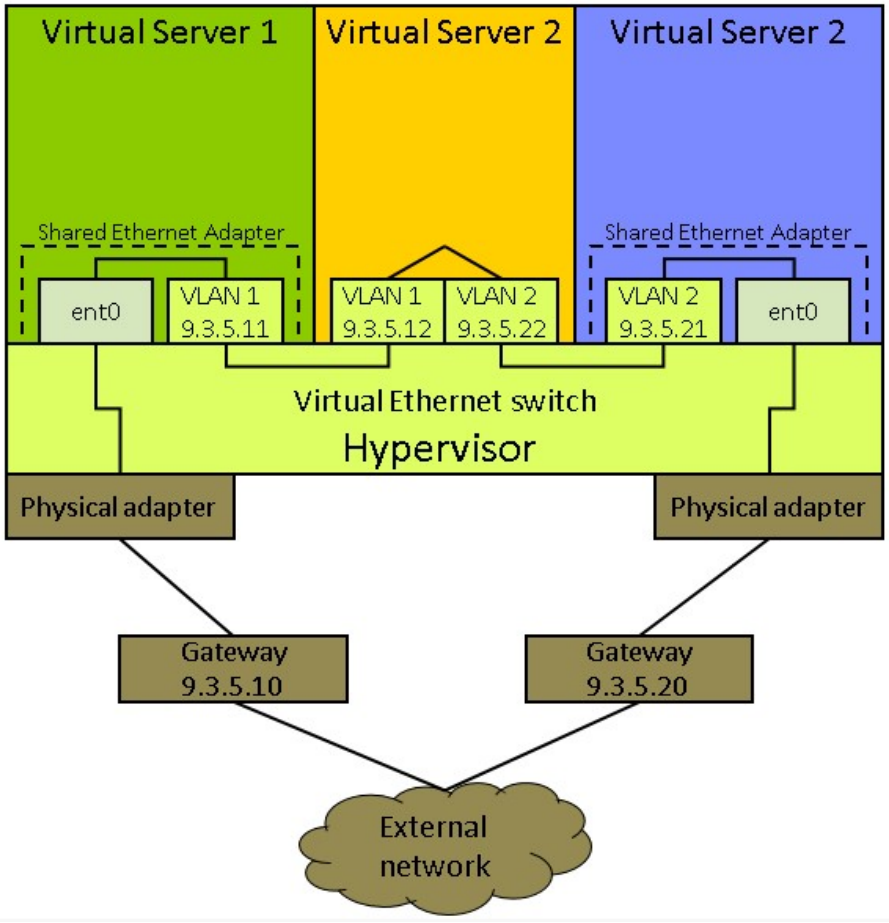
The experimental implementation provided a positive proof of concept of the suitability of Statistical Process Control as the basis for taking local management decisions and making online performance adjustments on a running VM. During the V-SPC controlled operation the VM's CPU and memory allocations were modified fourteen and eight times respectively as a result of management actions based on the interpretation of the live feed of performance data. Special importance must be given to determining the characteristics of the workload and of the virtualization platform, as these are used in the execution logic applied in our design. SPC control charting is depended on the handling of the control points produced during process monitoring. There are various reasons that lead to control points that fall outside of the calculated control limits – not all cases, however, indicate a problem in the monitored process. What is of importance is the *sustained* appearance of such points. This is reflected via the use of a *decision threshold* which determines, for the workload and infrastructure used, which control point repetition frequency is deemed as problematic (discussed in 5.1). In the case of the CPU resource, had we initiated a management action for every control point that was observed outside the control limits during the test run, eighty-nine actions would be needed, resulting in a hypothetical configuration of a VM of sixty-two CPUs (not to mention the duration overhead due to the enormous number of resource allocations [55]). This would, clearly, be an exaggeration and a failure of the proposed method, given that the acceptable process output is obtained with only ten CPUs. Dynamic memory management also exhibited similar problems with low decision thresholds leading to excessive of allocation actions and continuous oscillations in performance patterns.

Memory utilization and allocation, as opposed to CPU, poses special challenges. Memory is managed both by the hypervisor and by the running operating system. In the case of UNIX systems the hypervisor maps a range of physical memory to the VM's logical memory. This range of physical memory is dedicated to the VM in the case of a dedicated memory system. In certain configurations the physical memory assigned to one VM at one time can be assigned to another VM at another time. Therefore memory is not necessarily dedicated to a single VM; it can be mapped to any logical address of any shared memory VM of the same physical system [49]. Furthermore, the total logical memory of all shared memory VMs in a system is allowed to exceed the real physical memory (RAM) allocated to a shared memory pool in that system. This results in memory oversubscription, a condition that needs special design and handling as it can result in performance bottlenecks. Memory management becomes even more complicated as the operating system (*UNIX*, in the architecture discussed in this paper) caches file system data in order to enhance I/O performance. This results in two issues that need to be addressed: memory usage and processing due to file system caching and optimum monitoring of the actual workload impact on the memory resources. The latter is referred to as computational memory and this metric is of interested in the SPC approach discussed.

For our approach to have an effective end result, as with other dynamic resource reconfiguration methods, it is necessary that the application running on the managed system be able to handle changes in the hardware. The application must be able to adapt to changes in the configuration otherwise it will either remain unaffected or suffer performance degradation, failing to notice resources that are added or removed and scale accordingly. The platform's operating system and hypervisor play a crucial role in this direction by properly managing the allocated resources. We used a UNIX infrastructure in our experimental implementation [6] along with a well-parallelized multi-threaded application [7], where the dispatching of active processes was distributed relatively uniformly amongst all configured

processors. Despite the fact that our approach focused on CPU and memory as the target core resources it is possible for the V-SPC controller to manage any resource provisioned by the hypervisor, as long as, dynamic operations are supported on the resource. In modern hypervisors [49] networking and I/O attributes (such as fiber and virtual SCSI connection characteristics) can, also, be managed dynamically. Given the convergence of communications and computing to a common operational entity, the core networking support in server virtualization environments is based on the IEEE 802.1Q VLAN implementation, where virtual network segments are established on top of physical switches – the latter being provided by the server's hardware features (the hypervisor works as a virtual Ethernet switch and supports queues for each VLAN in the system's memory). In this way it is possible to establish network communication across different virtual servers, implementing virtual Ethernet adapters without routing network traffic outside the physical system that hosts them (Fig. 8) and performs the virtualization [49]. Depending on the managed resource and the platform used, a different accumulative threshold might be necessary for indicating sustained bad operation. Determining the best value will require per-case experimentation and controller tuning.

Dimitris Kontoudis
Figure 8
Top of figure



During the test runs the VM resource utilization was monitored four times every minute and the performance data were transmitted to the resource controller. There is no ubiquitous rule on the sampling frequency. What is of importance is to ensure that the sampling used does

capture any particular characteristics of the workload running on the VM. If there exist, for example, special processing stages that overcommit the system resources or influence the process output in any way, then these must be included in the collected data. Hence, the sampling frequency is workload-dependent and should be determined in such way so as not to introduce any bias [48]. Generally, during the initial stages of process control, sampling occurs more often and is decreased once stability of the process has been ensured. Managing the VM with the V-SPC controller did produce a related outcome: that of bringing the system's performance within acceptable limits with a CPU and memory capacity *lower than that of the initial VM setup*. It became apparent that the initial configuration was greater than that required in order to service the workload within the requested time frame. SPC can, therefore, also be used with a potential for capacity planning and profiling for any given workload, resulting in more efficient and cost-effective use of the infrastructure resources.

5.1 Metrics

For defining the auto-scaling triggering we concentrated on hardware level metrics, in particular, on system *processor and computational memory utilization* as these are of the key metrics involved in systems performance investigation as well as in related cost/benefit decisions in modern datacenters [55][56][21]. In the experimental platform the CPU utilization was manifested as an aggregated percentage figure for the total online virtual processors allocated to the logical partitions by the IBM Power hypervisor. SPC processing of aggregated CPU utilization determines the thresholds (upper and lower control limits) that dictate when a resource allocation must be considered. Finer control of the environment or specialized case handling can be achieved by monitoring specific hardware utilization factors, such as the number of processor context switches, the number of threads awaiting on the processor's run queue, the memory or network interface usage and other fine-grain details [43].

In the case of memory, we monitor and process the utilization of the computational pages (working memory segments) used by the actual workload running on the system. The size of the computational memory directly relates to the characteristics of the workload demand and comprises of the executable files along with their data. The diverse characteristics that affect the resource allocation (as manifested both by the hypervisor and by the operating system) are collectively represented as an overall utilization level. This includes, indicatively, reading and releasing pages of physical memory from/to the swap area, page faulting resulting from processes trying to load data from the hard disk as well as the overhead incurred by memory sharing between processes and/or intra-VM operations. It is possible to perform fine-grain dynamic resource allocation based on any specific attribute, by monitoring that very attribute and reacting based on the recorded utilization patterns [59]. This, however, will not necessarily provide an advantage as it is only the computational memory that collectively represents the information pertaining to the resource usage while the system is working on actual computation.

The use of the metric was combined with a *decision threshold* variable indicating the number of consecutive occurrences the upper or lower control limits were exceeded. Determining the suitable *decision threshold* involves examining the workload and the virtualization platform characteristics. The ability to dynamically allocate hardware resources is entirely platform-depended, in particular, on the hypervisor used. Whichever the virtualization platform, a certain amount of hypervisor inertia is always present, affecting the end-to-end time needed for hardware reconfigurations to take effect [55][49]. This delay may, also, include execution

time consumed by intermediate systems that enable the hypervisor operations, such as the Hardware Management Console system in the technologies used in the experimental implementation. We refer to the time taken for a hardware reconfiguration task to complete as the *adaptation time*. Usually, memory operations take longer time than the respective ones on the platforms' CPUs. Furthermore, each reconfiguration operation itself needs processing time and the higher the load imposed on the processors at that instant, the longer it takes for reconfigurations to complete. An optimum threshold must, therefore, be determined in order to avoid a negative effect due to overcommitting reconfiguration actions. In our infrastructure, the *adaptation time* for each CPU reconfiguration action was approximately 10 seconds. The duration for dynamic memory operations was in the range of 20 to 30 seconds. During the experiments, the decision to initiate a CPU reconfiguration action after three consecutive utilization readings (*decision threshold*) was proved to be the optimum for the infrastructure and workload used in our implementation. The respective threshold for memory reconfigurations was pinpointed at seven sequential out-of-bound readings.

5.2 Limitations of the proposed approach

SPC methods require the monitored data to be identically distributed. The standard Shewhart analysis of individual measurements produces control charts based on the Central Limit Theorem, which presupposes that the sampling distribution follows the normal distribution. This is workload-dependent and can be applied to VM utilizations with symmetric population distribution. Therefore, in the approach presented in this paper, the VM's performance is assumed to exhibit a constant mean and the workload used in the experimental implementation did produce such a performance pattern. CPU (or other resource) utilization that does not present these characteristics may not be able to be handled by the standard Shewhart approach. In these cases, other types of SPC tools could be applicable, such as the IX-MR [46], MCUSUM, MEWMA and multivariate Shewhart control charts [57]. Control charts, regardless of their type, can very quickly indicate when the quality of the measured process has changed. They do not, however, indicate the direction and the magnitude of required corrective measures. Hence, additional effort needs to be put on determining the nature of the latter and these measures are workload and infrastructure dependent. Finally, control charts are not entirely error-free and may, in cases, indicate that the process is out of control as a result of a special cause variation, whereas no issue actually exists. It is possible, therefore, for false alarms to be issued resulting in unnecessary management actions and possible unwanted overhead.

Threshold-based rule reactive approaches, regardless of the application focus (global or local scheduling of virtualized resources) exhibit the general characteristic of possible delays in infrastructure status detection and subsequent resource adaptation [9][42]. Our approach, falling under the same class of methods, is affected in the *reactivity* characteristics, i.e. in adjusting resource allocation responding to changes in the monitored workload. Although detection is near instantaneous, given the processing power of modern infrastructure and since the required statistical processing is completed *before* the monitoring period (during the SPC preprocessing phase), the resource adaptation heavily relies on the characteristics of the virtualization platform. Delays in adaptation may lead to instability in the infrastructure's operation (oscillations) as, by the time the resource schema has been reconfigured, the workload may have changed demanding different handling. Evaluation results show that the practical applicability of the method is affected if the underlying hardware virtualization cannot provide realistic dynamic reconfiguration response times.

The resource allocations (Fig. 7) were the results of the management actions following the primary metric analysis. One characteristic that needs to be taken into account is the time taken for the hypervisor to complete a reconfiguration request. This depends on a number of reasons that pertain to the hardware characteristics of the virtualization platform. On the deployed platform the dispatch latency of a virtual processor depends on the partition CPU entitlement and the number of virtual processors that are online in the virtual partition. The capacity entitlement is equally divided among these online virtual processors, therefore the number of online virtual processors impacts the length of each virtual processor's dispatch [58]. For the hypervisor used, the worst dispatch latency is 18 milliseconds, since the minimum dispatch cycle that is supported at the virtual processor level is one millisecond. This latency is based on the minimum partition entitlement of 1/10 of a physical processor and the 10 millisecond rotation period of the hypervisor's dispatch wheel [59][49]. Similar logic applies to any other hypervisor and CPU offered by other vendors and, depending on the workload or Cloud service characteristics, this overall CPU allocation hypervisor overhead has to be taken into account. Furthermore, as discussed in section 5, this *hypervisor inertia* requires an initial learning effort and overhead before the method is tuned for application on the platform of choice. It was found out during the experiments that controller tuning for memory operations required a greater effort given the complexity of memory management on the systems. Overall, it becomes apparent that the controller needs to be tuned to reflect the characteristics of the architecture it is deployed in. A configuration optimized for a specific hardware platform will not necessarily be appropriate for a different platform.

6. Conclusion

It is widely accepted in the IT industry that effective infrastructure resource management is of vital importance in successfully delivering services to users and other interested parties alike. Infrastructure resources need to be efficiently allocated in order to support the varying application workload demands. Failing to do so may result in erratic infrastructure operation, hence improper service, or excessive overhead in financial costs. This is more prominent in modern datacenters where the complexity of the architecture is increased by the use of server virtualization to support building and operating computer networks.

In this paper we introduced a new approach to dynamic resource management, based on a set of statistical concepts and tools, namely Statistical Process Control. Our approach allows for the online management, via auto-scaling, of processor and memory capacity of a virtual machine, taking into account the particular characteristics of the application workload running on it. We demonstrated the method on an experimental setup using real and not synthetic data, so as to provide a proof of concept of the method for use in actual datacenter environments. The involved resource controller, *V-SPC*, efficiently manages the virtualized resources of a virtual machine, adapting to changes in the workload patterns and providing for capacity planning and optimization of the initial resource allocation. The method can be extended to manage other type of resources such as hypervisor-provisioned virtual switch network bandwidth. Part of our ongoing work focuses on extending the resource controller to manage these other type of resources. Finally, we work towards enhancing the controller in order to effectively parse asymmetric resource performance patterns and to be able to correlate information resulting from different resource types. This will allow for the creation of a basis for the end-to-end management of a datacenter computing resource adhering to actual SLA-specified constraints regardless of the workload type.

References

[1] S. Sakr, A. Liu, D. M. Batista, and M. Alomari, "A survey of Large Scale Data Management Approaches in Cloud Environments," *IEEE Commun. Surv. Tutor.*, vol. 13, no. 3, pp. 311–336, 2011.

[2] Z. Qi, C. Lu, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," *J. Internet Serv. Appl.*, vol. 1, no. 1, pp. 7–18, 2010.

[3] M. F. Bari, R. Boutaba, R. Esteves, and L. Z. Granville, "Data Center Network Virtualization: A Survey," *IEEE Commun. Surv. Tutor.*, vol. 15, no. 2, pp. 909–928, 2013.

[4] P. Rygielski and S. Kounev, "Network Virtualization for QoS-Aware Resource Management in Cloud Data Centers: A Survey," *PIK - Prax. Informationsverarbeitung Kommun.*, vol. 36, no. 1, pp. 55–64, 2013.

[5] S. Patil and D. Lilja, "Statistical methods for computer performance evaluation," *WIREs Comput. Stat.*, vol. 4, no. 1, pp. 98–106, 2012.

[6] IBM, "IBM AIX operating system," *IBM AIX, UNIX software for Power Systems*, 2013. [Online]. Available: http://www-03.ibm.com/systems/power/software/aix/index.html. [Accessed: 04-Oct-2013].

[7] "Temenos T24 Core Banking Software," 2014. [Online]. Available: http://www.temenos.com/en/products-and-services/front-and-middle-office/t24-core-banking/.

[8] S. S. Manvi and G. K. Shyam, "Resource management for Infrastructure as a Service (IaaS) in cloud computing: A survey," *J. Netw. Comput. Appl.*, vol. 41, pp. 424–440, 2014.

[9] J. Brendan and R. Stadler, "Resource management in clouds: Survey and research challenges," *J. Netw. Syst. Manag.*, pp. 1–53, 2014.

[10] Y. Diao, J. L. Hellerstein, S. Parekh, and R. Griffith, "A control theory foundation for self-managing computing systems," *IEEE J. Sel. Areas Commun.*, vol. 23, no. 12, pp. 2213–2222, 2005.

[11] L. Chenyang, T. F. Abdelzaher, J. A. Stankovic, and S. H. Son, "A feedback control approach for guaranteeing relative delays in web servers," in *Proceedings of the 7th IEEE Real-Time Technology and Applications Symposium*, Taipei, 2001, pp. 51–62.

[12] T. F. Abdelzaher, K. G. Shin, and N. Bhatti, "Performance guarantees for web server end-systems: A control-theoretical approach," *IEEE Trans. Parallel Distrib. Syst.*, vol. 13, no. 1, pp. 80–96, 2002.

[13] L. Ying and T. F. Abdelzaher, "Design, implementation, and evaluation of differentiated caching services," *IEEE Trans. Parallel Distrib. Syst.*, vol. 15, no. 5, pp. 440–452, 2004.

[14] N. Gandhi and D. M. Tilbury, "MIMO control of an apache web server: Modeling and controller design," in *Proceedings of the 2002 American Control Conference*, Anchorage, USA, 2002, vol. 6, pp. 4922–4927.

[15] Y. Zhang, A. Bestavros, M. Guirguis, I. Matta, and R. West, "Friendly virtual machines: leveraging a feedback-control model for application adaptation," in *Proceedings of the 1st ACM/USENIX international conference on Virtual execution environments*, Chicago, USA, 2005, pp. 2–12.

[16] P. Padala *et al.*, "Adaptive control of virtualized resources in utility computing environments," in *Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems*, 2007, pp. 289–302.

[17] A. Mohit, P. Druschel, and W. Zwaenepoel, "Cluster reserves: a mechanism for resource management in cluster-based network servers," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 28, no. 1, pp. 90–101, 2000.

[18] M. Jones, D. Rosu, and M. C. Rosu, “CPU Reservations and Time Constraints: Efficient, Predictable Scheduling of Independent Activities,” *ACM SIGOPS Oper. Syst. Rev.*, vol. 31, no. 5, pp. 198–211, 1997.
[19] A. Chandra, W. Gong, and P. Shenoy, “Dynamic Resource Allocation for Shared Data Centers Using Online Measurements,” in *Proceedings of IWQoS 2003*, Monterey, CA, 2003, pp. 381–398.
[20] J. Chase, D. Anderson, P. Thakar, A. Vahdat, and R. Doyle, “Managing energy and server resources in hosting centers,” *ACM SIGOPS Oper. Syst. Rev.*, vol. 35, no. 5, pp. 103–116, 2001.
[21] X. Meng, C. Isci, J. Kephart, L. Zhang, E. Bouillet, and D. Pendarakis, “Efficient Resource Provisioning in Compute Clouds via VM Multiplexing,” in *Proceedings of the 7th International Conference on Autonomic Computing*, 2010, pp. 11–20.
[22] D. Breitgand and A. Epstein, “Improving consolidation of virtual machines with risk-aware bandwidth oversubscription in compute clouds,” in *Proceedings of the 2012 IEEE International Conference on Computer Communications*, 2012, pp. 2861–2865.
[23] M. Bennani and D. Menasce, “Resource allocation for autonomic data centers using analytic performance models,” in *2nd International Conference on Autonomic Computing*, 2005, pp. 229–240.
[24] B. Urgaonkar, G. Pacifici, P. Shenoy, M. Spreitzer, and A. Tantawi, “An analytical model for multi-tier internet services and its applications,” *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 33, no. 1, pp. 291–302, 2005.
[25] A. Ali-Eldin, J. Tordsson, and E. Elmroth, “An adaptive hybrid elasticity controller for cloud infrastructures,” in *13th IEEE Network Operations and Management Symposium*, 2012, pp. 204–2012.
[26] “AGILE: elastic distributed resource scaling for infrastructure-as-a-service,” in *10th International Conference on Autonomic Computing*, San Jose, CA, 2013, pp. 69–82.
[27] A. Scordaki and S. Psarakis, “Statistical Process Control in Service Industry - an Application with Real Data in a Commercial Company,” in *Proc. 7th Hellenic European Conference on Computer Mathematics and Its Applications*, Athens, Greece, 2005.
[28] R. S. Guh, J. D. T. Tannock, and C. O’Brien, “IntelliSPC: a hybrid intelligent tool for on-line economical statistical process control,” *Expert Syst. Appl.*, vol. 17, no. 3, pp. 195–212, 1999.
[29] Z. Chen, S. Lu, and S. Lam, “A hybrid system for SPC concurrent pattern recognition,” *Adv. Eng. Inform.*, vol. 21, no. 3, pp. 303–310, 2007.
[30] N. Boffoli, G. Bruno, D. Caivano, and G. Mastelloni, “Statistical process control for software: a systematic approach,” in *Proc. of the 2nd ACM-IEEE international symposium on Empirical software engineering and measurement*, Kaiserslautern, Germany, 2008, pp. 327–329.
[31] T. H. Nguyen, B. Adams, Z. M. Jiang, A. E. Hassan, M. Nasser, and P. Flora, “Automated Detection of Performance Regressions Using Statistical Process Control Techniques,” in *Proc. of the 3rd ACM/SPEC International Conference on Performance Engineering*, Boston, MA, USA, 2012, pp. 299–310.
[32] C. Lee, D. Lee, J. Koo, and J. Chung, “Proactive Fault Detection Schema for Enterprise Information System Using Statistical Process Control,” in *Proc. of the Symposium on Human Interface 2009 on ConferenceUniversal Access in Human-Computer Interaction*, San Diego, Ca, USA, 2009, pp. 113–122.
[33] Y. Nong, S. M. Emran, L. Xiangyang, and C. Qiang, “Statistical process control for computer intrusion detection,” in *Proc. of the DARPA Information Survivability Conference & Exposition*, Anaheim, CA, USA, 2001, vol. 1, pp. 3–14.

[34] Q. L. Zhang and J. Gao, “Applying SPC to autonomic computing,” in *Proc. of the 2004 International Conference on Machine Learning and Cybernetics*, Shanghai, China, 2004, vol. 2, pp. 744–749.
[35] D. T. Pham and E. Oztemel, “XPC: an on-line expert system for statistical process control,” *Int. J. Prod. Res.*, vol. 30, no. 12, pp. 2857–2872, 1992.
[36] R. R. Oliveira, A. A. Loureiro, and A. C. Frery, “A Multi-Scale Statistical Control Process for Mobility and Interference Identification in IEEE 802.11,” *Mob. Netw. Appl.*, vol. 14, no. 6, pp. 725–743, 2009.
[37] R. R. Oliveira, R. R. Pereira, and A. A. Loureiro, “Adaptive configuration of wpans and wlans communications using multi-scale statistical process control,” in *Proc. of the 10th ACM Symposium on Modeling, analysis, and simulation of wireless and mobile systems*, Chania, Crete, Greece, 2007, pp. 138–142.
[38] B. Wetherill and W. Brown, *Statistical Process Control: Theory and Practice*. Chapman and Hall, 1991.
[39] W. H. Woodall, “Controversies and Contradictions in Statistical Process Control,” *J. Qual. Technol.*, vol. 32, no. 4, pp. 341–350, 2000.
[40] T. Lorido-Botran, J. Miguel-Alonso, and J. Lozano A., “Auto-scaling Techniques for Elastic Applications in Cloud Environments,” University of the Basque Country, Technical Report EHU-KAT-IK-09-12, Sep. 2012.
[41] N. M. Calcavecchia, B. A. Caprarescu, E. Di Nitto, D. J. Dubois, and D. Petcu, “DEPAS: A Decentralized Probabilistic Algorithm for Auto-Scaling,” *Computing*, vol. 94, no. 8–10, pp. 701–730, 2012.
[42] F. Wuhib, E. Yanggratoke, and R. Stadler, “Allocating compute and network resources under management objectives in large-scale Clouds,” *J. Netw. Syst. Manag.*, vol. 23, no. 1, pp. 122–136, 2015.
[43] H. Ghanbari, B. Simmons, M. Litoiu, and G. Iszlai, “Exploring Alternative Approaches to Implement an Elasticity Policy,” in *Proceedings of the 2011 IEEE International Conference on Cloud Computing*, 2011, pp. 716–723.
[44] B. Zhang, X. Wang, R. Lai, L. Yang, Y. Luo, and X. Li, “Evaluating and optimizing I/O virtualization in kernel-based virtual machine (KVM),” in *Network and Parallel Computing*, vol. 6289, Springer Berlin Heidelberg, 2010, pp. 220–231.
[45] Y. Dong, Y. Xiaowei, L. Jianhui, L. Guangdeng, T. Kun, and G. Haibing, “High performance network virtualization with SR-IOV,” *J. Parallel Distrib. Comput.*, vol. 72, no. 11, pp. 1471–1480, 2012.
[46] D. J. Wheeler and D. S. Chambers, *Understanding Statistical Process Control*. Knoxville, TN.: SPC Press, 2010.
[47] S. Bain *et al.*, “A Benchmark Study on Virtualization Platforms for Private Clouds,” IBM SWG Competitive Project Office, ZSW03125-USEN-01, 2009.
[48] J. S. Oakland, *Statistical Process Control*. Butterworth-Heinemann, 2003.
[49] W. Armstrong *et al.*, “Advanced virtualization capabilities of POWER5 systems,” *IBM J. Res. Dev.*, vol. 49, no. 4, pp. 523–532, 2005.
[50] T. Bray, J. Paoli, C. M. Sperberg-McQueen, F. Yergeau, and J. Cowan, “Extensible Markup Language (XML),” W3C, W3C Technical Report REC-xml11-20060816, Aug. 2006.
[51] D. Kontoudis and P. Fouliras, “A survey of models for computer networks management,” *Int. J. Comput. Netw. Commun.*, vol. 6, no. 3, pp. 157–176, 2014.
[52] D. Kontoudis and P. Fouliras, “Modeling and managing virtual network environments,” in *Proceedings of the 17th PanHellenic Conference on Informatics (PCI ‘13)*, Thessaloniki, Greece, 2013, pp. 39–46.

[53] R. Jain and S. Paul, “Network virtualization and software defined networking for cloud computing: a survey,” *IEEE Commun. Mag.*, vol. 51, no. 11, pp. 24–31, 2013.
[54] Q. Duan, Y. Yan, and A. Vasilakos, “A Survey on Service-Oriented Network Virtualization Toward Convergence of Networking and Cloud Computing,” *IEEE Trans. Netw. Serv. Manag.*, vol. 9, no. 4, pp. 373–392, 2012.
[55] N. Huber, M. Von Quast, M. Hauck, and S. Kounev, “Evaluating and modeling virtualization performance overhead for cloud environments,” in *Proc. of the 1st International Conference on Cloud Computing and Services Science*, Noordwijkerhout, The Netherlands, 2011, pp. 563–573.
[56] M. Fenn, M. Murphy, J. Martin, and S. Goasguen, “An evaluation of KVM for use in cloud computing,” in *Proceedings of the 2nd International Conference on the Virtual Computing Initiative (ICVCI ’08)*, Durham, NC, 2008.
[57] S. Bersimis, S. Psarakis, and J. Panaretos, “Multivariate statistical process control charts,” *Qual. Reliab. Eng. Int.*, vol. 23, pp. 517–543, 2007.
[58] IBM, “Server virtualization with IBM PowerVM,” *Server virtualization with IBM PowerVM*, 2013. [Online]. Available: http://www-03.ibm.com/systems/power/software/virtualization/. [Accessed: 03-Oct-2013].
[59] S. Vetter, M. Cordero, L. Correia, H. Lin, V. Thatikonda, and R. Xavier, *IBM PowerVM Virtualization Introduction and Configuration*. IBM, 2015.
[60] H. Fernandez, G. Pierre, and T. Kielmann, “Autoscaling Web Applications in Heterogeneous Cloud Infrastructures.” in Proceedings of the IEEE International Conference on Cloud Engineering (IC2E), March 2014, pp. 195–204.
[61] N. Roy, A. Dubey, and A. Gokhale, “Efficient autoscaling in the cloud using predictive models for workload forecasting,” in Proceedings of the IEEE International Conference on Cloud Computing (CLOUD), 2011, pp. 500–507.
[62] C. Kan, “DoCloud: An elastic cloud platform for Web applications based on Docker,” in Proceedings of the 18th International Conference on Advanced Communication Technology (ICACT), Jan 2016, pp. 478–483.
[63] S. M.-K. Gueye, N. D. Palma, E. Rutten, A. Tchana, N. Berthier, “Coordinating self-sizing and self-repair managers for multi-tier systems,” Future Gener. Comp. Sy., vol. 35, pp. 14 – 26, 2014.
[64] W. Iqbal, M. N. Dailey, D. Carrera, P. Janecek, “Adaptive resource provisioning for read intensive multi-tier applications in the cloud,” Future Gener. Comp. Sy., vol. 27, no. 6, pp. 871 – 879, 2011.
[65] Z. Gong, X. Gu, J. Wilkes, “Press: Predictive elastic resource scaling for cloud systems,” in Proceedings of the International Conference on Network and Service Management (CNSM), Oct 2010, pp. 9–16.
[66] S. Khatua, A. Ghosh, N. Mukherjee, “Optimizing the utilization of virtual resources in Cloud environment,” in Proceedings of the IEEE International Conference on Virtual Environments Human-Computer Interfaces and Measurement Systems (VECIMS), Sept 2010, pp. 82–87.
[67] Laura R. Moore, Kathryn Bean, Tariq Ellahi, “A Coordinated Reactive and Predictive Approach to Cloud Elasticity,” in Proceedings of the Fourth International Conference on Cloud Computing, GRIDs, and Virtualization, May 27 - June 1 2013, Valencia, Spain
[68] Daniel Gmach Jerry Rolia, Ludmila Cherkasova, Alfons Kemper, “Resource pool management: Reactive versus proactive or let’s be friends,” Comput. Netw., vol. 53, pp. 2905–2922, 2009.
[69] Carlos Vazquez, Ram Krishnan, Eugene John, “Time Series Forecasting of Cloud Data Center Workloads for Dynamic Resource Provisioning,” J. of Wireless Mobile

Networks, Ubiquitous Computing, and Dependable Applications, vol. 6, no. 3, pp. 87-110, 2016.
[70] Yahya Al-Dhuraibi, Fawaz Paraiso, Nabil Djarallah, Philippe Merle, “Elasticity in Cloud Computing: State of the Art and Research Challenges,” IEEE T. Serv. Compu., vol.: PP, no. 99, pp. 1-1, 2017.
[71] Athanasios Naskos, Anastasios Gounaris, Spyros Sioutas, “Cloud Elasticity: A Survey,” in Proceedings of the First International Workshop on Algorithmic Aspects of Cloud Computing, 2016, Patras, Greece, pp 151-167.
[72] Hummaida R. Abdul, Norman W. Paton, Rizos Sakellariou, “Adaptation in cloud resource configuration: a survey,” J. of Cloud Computing: Advances, Systems and Applications, vol.5, no. 7, 2016.
[73] N. Vasic, D. Novakovic, S. Miucin, D. Kostic, R. Bianchini, “DejaVu: Accelerating Resource Allocation in Virtualized Environments,” in Proceedings of the 17th International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS XVII, NY, USA, 2012, pp. 423–436.
[74] S. Islam, J. Keung, K. Lee, A. Liu, “Empirical Prediction Models for Adaptive Resource Provisioning in the Cloud,” Future Gener. Comp. Sy., vol. 28, no. 1, pp. 155–162, 2012.
[75] E. Kassela, C. Boumpouka, I. Konstantinou, N. Koziris, “Automated workload-aware elasticity of NoSQL clusters in the Cloud,” in Proceedings of the IEEE International Conference on Big Data, Oct 2014, pp. 195–200.
[76] L. R. Moore, K. Bean, T. Ellahi, “A coordinated reactive and predictive approach to cloud elasticity,” in Proceedings of the 4th International Conference on Cloud Computing, GRIDs, and Virtualization, 2013, pp. 87–92.
[77] Moore, Laura R. and Bean, Kathryn, Ellahi, Tariq, “Transforming Reactive Auto-scaling into Proactive Auto-scaling,” in Proceedings of the 3rd International Workshop on Cloud Data and Platforms, NY, USA, 2013, pp. 7–12.
[78] P. D. Kaur, I. Chana, “A resource elasticity framework for qos-aware execution of cloud applications,” Future Gener. Comp. Sy., vol. 37, pp. 14–25, 2014.
[79] H. Mi, H. Wang, G. Yin, Y. Zhou, D. Shi, L. Yuan, “Online Self-Reconfiguration with Performance Guarantee for Energy Efficient Large-Scale Cloud Computing Data Centers,” in Proceedings of the IEEE International Conference on Services Computing, July 2010, pp. 514–521.
[80] J. Huang, C. Li, and J. Yu, “Resource prediction based on double exponential smoothing in cloud computing,” in Proceedings of the 2nd International Conference on Consumer Electronics Communications and Networks, 2012, pp. 2056–2060.
[81] F. Paraiso, P. Merle, L. Seinturier, “soCloud: A Service Oriented Component-Based PaaS for Managing Portability, Provisioning, Elasticity, and High Availability across Multiple Clouds,” Computing, pp. 1–27, 2014.
[82] Rightscale, "Set up Autoscaling using Alert Escalations," 2017. [Online]. Available: http://support.rightscale.com/03-Tutorials/02-AWS/02-Website_Edition/How_do_I_set_up_Autoscaling%3F/ [Accessed: 12-May-2017].
[83] Rightscale, "Set up Autoscaling using Voting Tags," 2017. [Online]. Available: http://support.rightscale.com/12-Guides/Dashboard_Users_Guide/Manage/Arrays/Actions/Set_up_Autoscaling_using_Voting_Tags/index.html [Accessed: 13-May-2017].
[84] Rightscale, "Understanding the Voting Process," 2017. [Online]. Available: http://docs.rightscale.com/cm/rs101/understanding_the_voting_process.html [Accessed: 13-May-2017].

[85] C. Kan, “DoCloud: An elastic cloud platform for Web applications based on Docker,” in Proceedings of the 18th International Conference on Advanced Communication Technology, 2016, pp. 478–483.
[86] F. Cruz, F. Maia, M. Matos, R. Oliveira, J. A. Paulo, J. Pereira, R. Vilac¸ “MeT: Workload Aware Elasticity for NoSQL,” in Proceedings of the 8th ACM European Conference on Computer Systems, NY, USA, 2013, pp. 183–196.
[87] Moore, L. R., Bean, K., Ellahi, T., “Transforming Reactive Auto-scaling into Proactive Auto-scaling,” in Proceedings of the 3rd International Workshop on Cloud Data and Platforms, NY, USA, 2013, pp. 7–12.
[88] Amazon “Autoscaling,” 2017. [Online]. Available: https://aws.amazon.com/autoscaling/ [Accessed: 05-June-2017].