Massive GIS Database System with Autonomic Resource Management

Yun Lu, Ming Zhao, Guangqiang Zhao, Lixi Wang, Naphtali Rishe
NSF Industry-University Cooperative Research Centers and
School of Computing and Information Sciences
Miami, Florida 33199
{yun, ming, gzhao002, lwang007, rishe}@cs.fiu.edu

Abstract:* GIS application hosts are becoming more and more complicated. Thus, their management is more time consuming, and reliability decreases with the complexity of GIS applications increasing. We have designed, implemented, and evaluated, a virtualized whole Large Scale Distributed Spatial Data Visualization System for optimizing maintainability and performance when handling large amount of GIS data. We employ the virtual machines (VMs) technique, load balance cluster techniques, and autonomic resource management to improve the system's performance. The proposed system was prototyped on TerraFly [1], a production web map service, and evaluated using actual TerraFly workloads. The results show that the virtual TerraFly system has both good performance and much better maintainability. Our experiments show that the proposed Virtual TerraFly Geo-database system has doubled the reliability, and saved 20-30% computing resources cost compared to current static peak-load physical machine node allocations.

***Keywords:* Database Systems, GIS, maintainability, performance**

I. Introduction

With the exponential growth of the World Wide Web, there are more domains open to GIS applications. The Internet can provide information to a multitude of users, making GIS available to a wider range of users than ever before.

This change of domain users from GIS experts to the general public requires the re-evaluation of design issues and creations of new features for GIS. A major objective is to make a GIS system accessible to the general public, who has little knowledge of spatial data, and allow them to interact with the system to manipulate and retrieve the information they need. In order to address this issue, we have adopted systematic methodologies to solve several distributed digital library system problems. This research utilizes the TerraFly platform, a web-based spatial data access system that handles many remote sensed data sets.

Virtualization techniques can improve the performance and manageability of web-enabled system, while handing large scale GIS data. For a web based system with very large amounts of data, like TerraFly, the response time of search requests is critical, as are the maintainability and manageability. By varying the configuration of Virtual Machines, the whole system performance may improve.

VMs are powerful platforms for hosting this Geo-database system. VMs support flexible resource allocation to meet both database system demands, and share resources with other applications. Virtualization is also an enabling technology for the emerging cloud computing paradigm, which further allows highly scalable and cost-effective database hosting, leveraging its elastic resource availability, and pay-as-you-go economic model [2]. This allows solution of major geospatial data processing problems.

Due to the highly complex and dynamic nature of GIS database systems, it is challenging to efficiently host them using virtualized resources. Typical database systems and applications have to serve dynamically changing workloads, consisting of a variety of queries, and consuming various types (and amounts) of resources. This makes it difficult to host databases on shared resources without compromising performance, or wasting resources.

We present solutions to problems of autonomic resource allocation of virtualized spatial data visualization systems, to improve the system performance, and reduce the system computing resources cost. The system should be able to automatically learn the current system requirement of resources, and to automatically allocate computing resources. Combined with other kinds of database system optimization, this greatly benefits the TerraFly geo spatial database system.

In order to realize and verify the improvement, we study the Virtualized TerraFly System. By employing a map partition algorithm, cache technique, load balance cluster techniques combined with virtualization technology, the proposed Virtualized TerraFly System gains improvement in both performance and maintainability, and this kind of configuration works on most web-enabled large data GIS systems. This paper proposes novel offline and online methods to achieve VM nodes allocation.

This proposed system is realized on Hyper-V VM environments, and evaluated via experiments using actual workloads collected from the production TerraFly system. The results showed that the system has a 20-30% general performance improvement and a 116% improvement in very important performance parameters. It also saves substantial resources, compared to static peak-load machine node allocation.

In summary, this article's main contribution is novel autonomic resource allocation to a virtualized geo-database that has a changing geo query-workload. The rest of this paper is organized as follows: Section 2 presents the background of the

TerraFly system. Sections 3 and 4 aspects of proposed Virtual TerraFly system. Section 5 presents results from quantitative performance analyses. Section 6 examines related work, and finally, Section 7 concludes the paper.

II. Background

A. TerraFly Spatial Data Visualization System

TerraFly is a web-enabled system designed to aid in the visualization of spatial and remote sensed imagery. TerraFly users visualize aerial photography, satellite imagery, and various overlays, such as street names, roads, restaurants, services and demographic data. TerraFly systems can manage large amounts of spatial data, and provide advanced functionalities. Users virtually "fly" over imagery via a web browser, without any software to install or plug in. TerraFly's tools include user-friendly geospatial querying, data drill-down, interfaces with real-time data suppliers, demographic analysis, annotation, route dissemination via autopilots, customizable applications, production of aerial atlases, and application programming interface (API) for web sites [3].

Because the number of users is increasing, in addition to the increasing amount of integrated data of TerraFly systems, the performance of whole systems is becoming more important. Distributing the system is a good way to expand the capacity, processing and storage. However, there might be a better way to realize the distributing VMs.

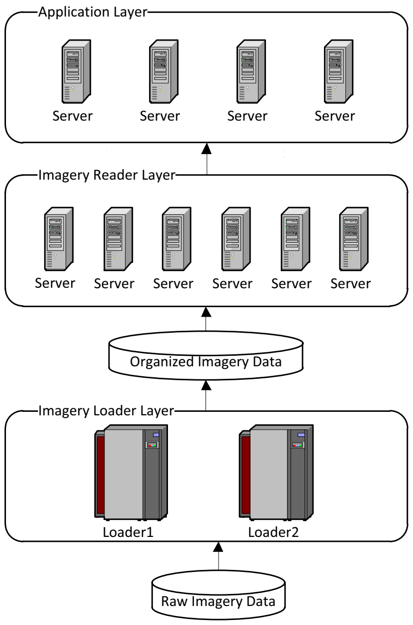

Figure 1: Physical TerraFly system

Browsers have a limited number of download connections per hostname. For example, IE 8 increased their number of downloads per hostname from 2 to 6. The default settings in other browsers have been tested, for example, Firefox 3: 2 connections, Firefox 5: 6 connections, and Safari 3.0.4 Mac/Windows: 4 connections. Users can modify their connection number themselves, for example, "network.http.max-persistent-connections-per-server" in Firefox.

The TerraFly system has three layers of components as seen in Figure 1. The image data, which TerraFly offers, is dynamic data, which means the data is calculated when the request comes. This may offer the latest image, or customized image; it is very flexible, and the tradeoff is more calculations on the server side, especially in the image reader layer [4].

We have analyzed client requests covering over 125 countries and regions worldwide, as well as all of the United States. In order to improve the overall performance, we have implemented a cache that balances computing resources. Since the majority of our requests are based in Florida, Texas, California, and New York, we have allocated additional VMs to handle map request over these states, conversely reducing and combining the cache and VMs resources on the states or countries for which we seldom receive request, in an effort to balance the computing resources system-wide [5]. This information can be used to provide a baseline template for placement and deployment of VMs, and when the VM resources of these seldom used areas are insufficient, the Dynamic VM migration can move it to the host computer, which has available resources.

B. Virtualized Database Hosting

Traditionally, geo-databases are hosted on dedicated physical servers that have sufficient hardware resources to satisfy their expected peak workloads in a desired response time.

VM consolidation has become increasingly important for improving efficiencies of resource usage and power consumption in data centers. It enables one physical server to host multiple independent virtual machines, and the transparent movement of workloads from one physical server to another. Previous work has addressed the problem of placing and replacing virtual machines in servers, in order to optimize the management of data center resources from various perspectives, including performance, power, and thermal management. For application clusters, we can create one VM per application, or a group of similar applications, to facilitate the management of applications.

However, this is often inefficient for the real-world situations in many application domains where the workloads are intrinsically dynamic in terms of their busty arrival patterns, and ever-changing unit processing costs. Even in domains where static workload exists, the database can dynamically switch from one workload to another at runtime. Consequently, peak-load based resource provision often leads to underutilization of resources for normal state workloads, and causes substantial overhead.

Using VMs to host databases can effectively address this limitation, because virtualized resources, including CPU and memory, are decoupled from their physical infrastructure, and can be flexibly allocated to the databases as needed. This approach allows a database to transparently share the consolidated resources with other applications, with strong isolation between their dedicated VMs. It also allows a database's resource usage to elastically grow and shrink based on the dynamic demand of its workload. Such benefits are important to the efficiency of database hosting in both typical data centers, and emerging cloud systems. On one hand, users only need to pay for the resources that their databases actually consume. On the other hand, resource providers only need to allocate resources as required by the database VMs, while saving valuable resources for hosting other applications.

Virtualization also offers a new paradigm for database deployments. Modern databases are sophisticated software systems, where their installation and configuration require substantial domain knowledge and experience, as well as considerable efforts from database administrators. VM based database hosting allows carefully installed databases to be distributed as simply as copying the data that represents the database VMs. In addition, this approach allows databases to be quickly replicated and distributed for performance and reliability improvements [6].

III. VIRTUALIZING TERRAFLY SYSTEM

A. *Motivation*

There are two challenges in this hosting system:

1. The dynamics in the realistic workload, which causes CPU consumption, changes over time.
2. The specific objective in TerraFly Geo-Date system. On one hand, you want to assign more VMs to reader layers in order to optimize response time. On the other hand, you also need to reserve some VM resources for the loader node to keep the data fresh.

B. *Virtual Machines on TerraFly*

The plan of virtualizing the system is to virtualize all three layers: image loader layer, image reader layer and application layer, to make the whole system easier to manage and to increase the utilization of resources.

If we create VMs on all three layers with the same configuration of previous physical machines, we can use those VMs to replace the previously used physical machines.

We can deploy TerraFly to several virtual machines to break the limitations of browsers. Browsers have a limited number of downloads per hostname. Typically, a full screen of maps is composed of less than 12 tiles (4 x 3); therefore, deploying the reader cluster as 12 VMs with dedicated IP addresses is good for the situation. This article will discuss later why, and how, to deploy VMs.

The load process is not always in service; it only starts when new image data comes in. Therefore, it is a good idea if the loader cluster and reader cluster share resources. As shown in Figure 2, we can activate a VM once there is an image source needed to be loaded, and assign to the loader VMs much more resources than to the reader VMs.

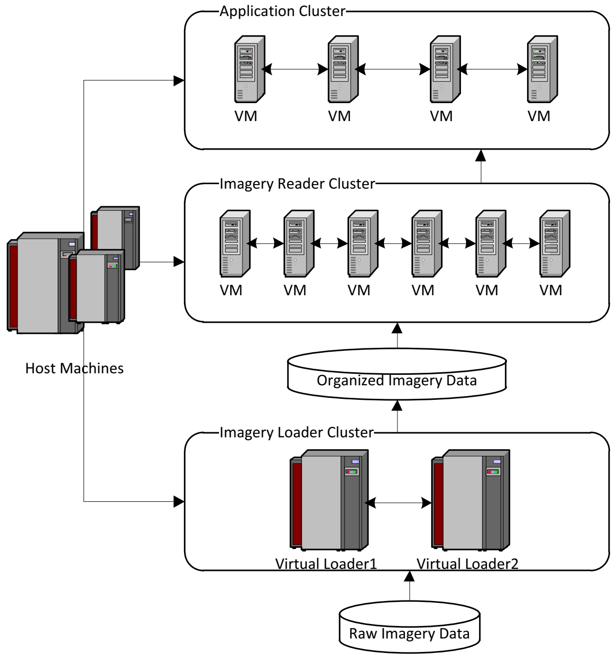

Figure 2: Virtualized TerraFly System

C. *Virtual Load balance cluster*

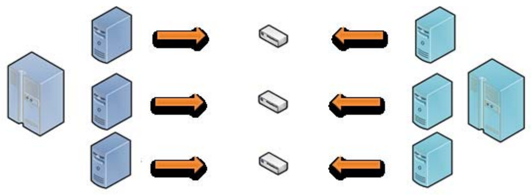

Figure 3: Virtual load balance cluster

Network Load Balancing can provide high availability and reliability, as well as high scalability. Web applications are stateless applications, and every client request to a stateless application is a separate transaction, so it is possible to distribute the requests among multiple servers to balance the load. One attractive feature of Network Load Balancing is that all servers in a cluster monitor each other with a heartbeat signal, so there is no single point of failure.

Use of a virtual machine will facilitate the build of load balancing clusters. Two host servers build pairs of VMs, and then the pairs construct different load balance clusters (Figure 3). For application layers, each server will have a paired server, to respond to requests together. This is an implementation of dual-server auto fail-over, offering better reliability.

D. Log research and VM resource control

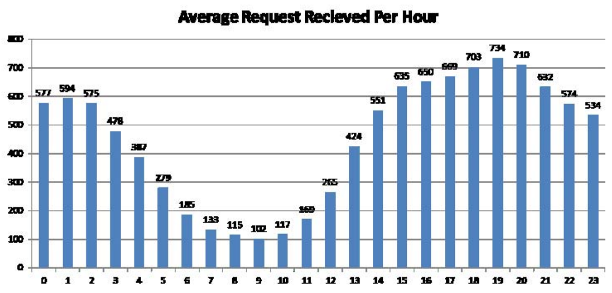

Figure 4: Workload daily pattern

Another goal is create an active mechanism of VM resource allocation, to enable the VMs to allocate resources automatically based on the real-time requirement, in order to improve the system, and reduce the system cost. For example, let a group of VMs be complementary to each other. Once one of the VM runs out of computing resources, other VMs will share their idle resources to the VM by the dynamic VM resource control technology.

Using machine learning technique to analyze the usage data to predict future patterns is a good way to produce a proper configuration of VMs, and improve the entire system performance.

TerraFly systems have a large numbers of users, and it is very important that we analyze the real usage data of a web GIS system. We analyzed the usage data of one of the most important image reader servers. There were 3 Million request information collected in the logs over a 304 day period. There were, on average, 450 visits per hour. In Figure 4, we can see that the overall request is lowest between the hours of 4 am to 1 pm; specifically between 7 am to 10 am. During this time, the request is approximately 1 percent of the daily requests received. Using this information to configure the initial VM placement, it should be possible to turn off some servers, and save those resources for other uses, or power the servers down, or have them placed into sleep states in order to conserve power during this low point in activity. Should the amount of requests being received suddenly increase, then via the use of Dynamic VM resource allocation, more host computers can be brought back online to handle the additional processing needed by the influx of new requests.

IV. AUTONOMIC RESOURCE MANAGEMENT

In this article, in order to describe how many resources are used, we use nodes×hours as the unit to measure the resource cost. One node×hours means we use one core CPU and a 2G memory machine node to work per hour. Both physical and virtual machines are using the same measure.

A. On-demand offline resource allocation for different workloads

We use varying quantities of map reader nodes on all kinds of request rates, and then monitor the response time and throughput. This way, we find the lowest number of VM nodes needed by varying the request rate, to adjust the final plan of optimization.

The desired system page time is 0.9 seconds. If we want to achieve this goal by the traditional deployment plan, we need 10 node readers, but we may need much less by dynamic deployment of VMs.

For each VMs configuration, we perform an experiment. The number of users per VM is from 40 to 260, and we collect the page time. As a result, we obtain the minimal number of VMs to achieve the 0.9 second page time.

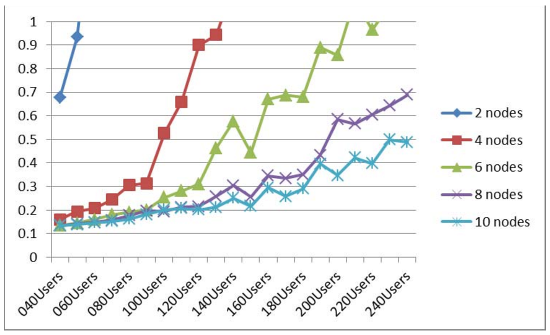

Figure 5: Offline Profile

40 Users Per VM	Need 2 nodes
110 Users Per VM	Need 4 nodes
180 Users Per VM	Need 6 nodes
230 Users Per VM	Need 8 nodes
260 Users Per VM	Need 10 nodes

B. On-demand dynamic deployment and CPU usage

We vary quantities of map reader nodes on all kinds of request rates shown in Figure 5, and then monitor the response time and CPU usage. This way, we find the lowest CPU amount needed by different request rate to adjust the final plan of optimization.

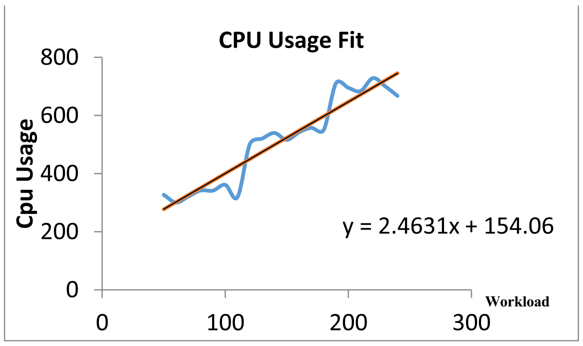

Figure 6: Workload to CPU Usage model

The requirement of CPU is almost linear with request rates. We composed a trend line: *c(t)* is the CPU usage of the whole system at time *t*, and *w(t)* is the number of users per VM at *t*. $c(t) = 2.4631*w(t) + 154.06$, as shown in Figure 6.

C. Online Dynamic VMs deployment by prediction of workload

The major difficulty of online resource management for a virtualized system lies in how to model its intrinsically dynamic and complex behavior in an accurate and efficient way. Commonly used linear modeling methods are no longer sufficient for modeling a system whose workload consists of different queries with diverse usage of multiple types of resources. We combined all previous work results, to propose an optimal deployment of VMs based on both the history of request rates and of the current request predictions. The following is the architecture of our proposed resource management for virtualized databases based on the aforementioned VM deployment approach:

1) Predict algorithm

Because user requests have a time pattern, as mentioned before, we perform request rate predictions by exponential smoothing along with weighing of history data.

Time series analysis techniques are widely applied in economic data analysis to provide statistical prediction and to guide business decisions. A variety of time series prediction methods is available, such as the Moving Averages, Linear Regression, and Exponential Smoothing [9][10][11]. *V(t)* is the last hour request value, horizontally second exponential smoothing predicted value is *h(t)*, vertically second exponential smoothing predicted value is *v(t)*, and the history average value is *s(t)*. vSpan is the vertical Span: in this pattern, we predict hourly, the vertical is the same hour on different days, the span is 24 hours. *p(t)* is the final prediction of the work load.

$$A_1(t) = \alpha \times V(t) + (1-\alpha) \times A_1(t-1)$$

$$A_2(t) = \beta \times A_1(t) + (1-\beta) \times A_2(t-1)$$

$$h(t) = 2 \times A_1(t) - A_2(t)$$

$$B_1(t) = \rho \times V(t) + (1-\rho) \times B_1(t-vSpan)$$

$$B_2(t) = \gamma \times B_1(t) + (1-\gamma) \times B_2(t-vSpan)$$

$$v(t) = 2 \times B_1(t) - B_2(t)$$

$$p(t) = a \times h(t) + b \times v(t) + (1-a-b) \times h(t)$$

2) Parameter training

The parameters are *α, β, ρ, γ, a, and b.* All six parameters were recalculated each hour, to achieve the smallest standard errors *e. k* is the accuracy of the error calculating.,

$$e_h = \sum_t \left(\sum_{i=0}^{k} (V(t-k) - h(t-1-k))^2 \right) \div k$$

$$e_v = \sum_t \left(\sum_{i=0}^{k} (V(t-k) - v(t-1-k))^2 \right) \div k$$

$$e = \sum_t \left(\sum_{i=0}^{k} (V(t-k) - p(t-1-k))^2 \right) \div k$$

We use e_h to train the combination of *α, β*. The range of *α, β* is from 0 to 1. We use e_v to train the combination of *ρ, γ*. The range of *ρ, γ* is from 0 to 1. We use e_h to train the combination of *a, b*. The range of *a, b* is from 0 to 1.

As figure 7 shows, the forecast *p(t)* is a combination of *h(t)*, *v(t)* and *h(t)*, and reveals that both predictions of previous work loads, and historical workloads, are a very good estimate of the next time period.

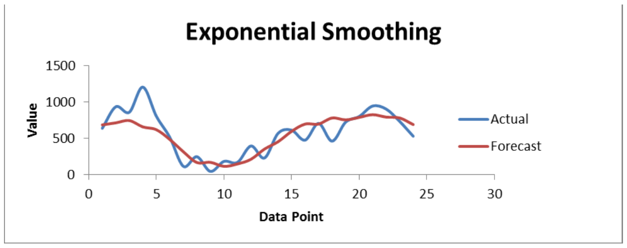

Figure 7: Workload prediction

V. EVALUATION

We have evaluated by comparing the virtual TerraFly system with the current physical TerraFly system. The first aspect is performance, including response time, throughput, and power consumptions. We have run both systems to handle different kinds of test cases, and then analyzed the resulting data. The second aspect is maintainability and manageability, including the ability to confront system errors and collapse, and the service management cost. We have examined whether virtual TerraFly systems can offer further benefits other than performance improvement.

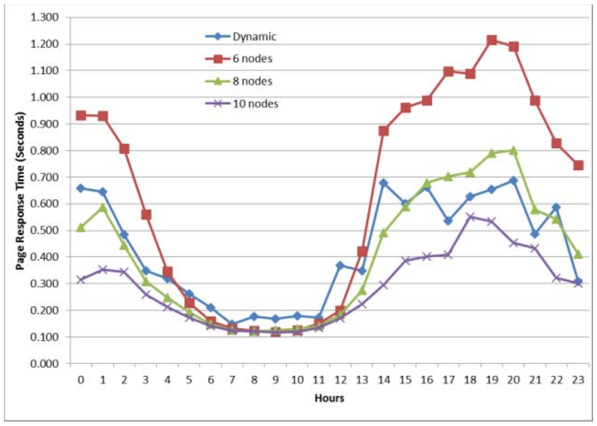

Figure 8: Result: page response time

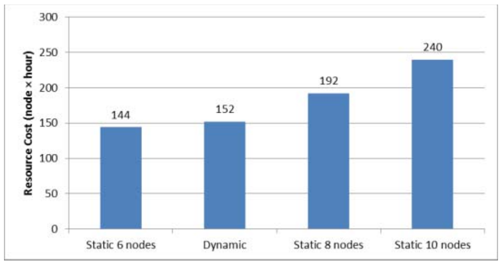

Figure 9: Result: total VM nodes cost

Showed at Figure 8 and Figure 9, the Online dynamic deployment plan used 152 nodes × hours a day, less than the static 8 nodes plan of 192 nodes × hours, and 10 nodes plan of 240 nodes × hours. The plan exceeded the static 6 nodes plan by only 5.56% of resource cost, but offered a much better performance.

Compared with the static 6-node plan with the 0.634 seconds of average page time, the online dynamic plan provided a very good average page time of 0.430 seconds, which is a 32.18% of average page time improvement, only using 152-144=8 nodes × hours. There were 13 points with page time exceeding 0.7 seconds (the expected value) in the static 6-node plan, which is, with 54.17% of the results not reaching the expected value; there were no points exceeding 0.7 seconds in the dynamic plan, which is a great improvement on the worst page time points. The slowest page time was 0.687, which is a good and reasonable response time in a web application. Most of the points of the static 6-node plan were worse than the points of the dynamic plan.

Compared with the static 8-node plan, the dynamic plan saved 192-152=40 nodes × hours, which saved 20.83% of resources. There were 3 points with page times exceeding 0.7 seconds (the expected value) in the static 8-node plan, while no points exceeded 0.7 seconds in the dynamic plan. The performance of the dynamic plan was better than the static 8-node plan, and at the same time, saved 20.83% of resources.

Compared with the static 10-node plan, the dynamic plan saved 240-152=88 nodes × hours, which saved 36.67% of resources. No points exceeded 0.7 seconds (the expected value) in both the static 10-node plan and the dynamic plan. The performance of the dynamic plan was the same as of the static 10-node plan, although the static 10-node plan had better page times.

VI. Conclusion

Virtualization can greatly facilitate the deployment of database systems, and substantially improve their resource utilization. To fulfill this potential, resource management is the key; it should be able to automatically allocate resources to geo-database VMs based on their performance targets. In this article, we have proposed a large scale distributed spatial data visualization system with autonomic resource management to improve performance and maintainability of Web-based large geo-database systems. Specifically, our provisioning approach is based on the use of virtual machine autonomic resource allocation, cache, and load balance cluster techniques to improve the whole system. Our preliminary evaluations showed that our approach achieves significant improvements in both performance, and maintainability.

Various types of solutions have been studied in the literature to address the problem of autonomic VM resource management. Various machine learning algorithms have been considered to model VM resource usages [15][16][17]. Although this article focused on virtualized geo-databases, we believe that our proposed VM resource allocation approach is generally applicable to the virtual resource management for other types of Big Data handling applications. The application-specific part of this approach is geo-database system applications with large numbers of users. Compared to the traditional approach, which treats a VM as a black box, such a gray-box resource management approach can be more accurate for modeling virtualized applications that have dynamic and complex resource usage behaviors.

Acknowledgment

This material is based in part upon work supported by the National Science Foundation under Grant Nos. MRI CNS-0821345, MRI CNS-1126619, CREST HRD-0833093, I/UCRC IIP-1338922, I/UCRC IIP-0829576, RAPID CNS-1057661, RAPID IIS-1052625, MRI CNS-0959985, AIR IIP-1237818, SBIR IIP-1330943, FRP IIP-1230661, III-Large IIS-1213026, SBIR IIP-1058428, SBIR IIP-1026265, SBIR IIP-1058606, SBIR IIP-1127251, SBIR IIP-1127412, SBIR IIP-1118610, SBIR IIP-1230265, SBIR IIP-1256641, HECURA CCF-0938045, CAREER CNS-1253944. Includes material licensed by TerraFly (http://terrafly.com) and the NSF CAKE Center (http://cake.fiu.edu

References

[1] Rishe, N., Chen, S. C., Prabakar, N., Weiss, M. A., Sun, W., Selivonenko, A., & Davis-Chu, D. (2001, April). TERRAFLY: A High-Performance Web-based Digital Library System for Spatial Data Access. In ICDE Demo Sessions (pp. 17-19).

[2] Lixi Wang, Jing Xu, Ming Zhao, Fuzzy Modeling Based Resource Management for Virtualized Database Systems, 19th Annual IEEE International Symposium on Modelling, Analysis, and Simulation of Computer and Telecommunication Systems

[3] Lu, Y., Zhang, M., Tao Li, Y. G., & Rishe, N. (2013). Online Spatial Data Analysis and Visualization System. ACM KDD IDEA 2013, pp.73-79

[4] Robust Database Structures with Dynamic Query Patterns", EJOR, 2006.

[5] T. E. Anderson, L. L. Peterson, S. Shenker, and J. S. Turner, "Overcoming the internet impasse through visualization." IEEE Computer, vol. 38, no. 4, pp. 34-41, 2005.

[6] Bennani, M. N., & Menasce, D. A. (2005, June). Resource allocation for autonomic data centers using analytic performance models. In Autonomic Computing, 2005. ICAC 2005. Proceedings. Second International Conference on (pp. 229-240). IEEE.

[7] Network reconfiguration in distribution systems for loss reduction and load balancing, ME Baran, FF Wu - Power Delivery, IEEE Transactions on, 1989 - ieeexplore.ieee.org

[8] T. Wood, L. Cherkasova, K. Ozonat and P. Shenoy, "Profiling and Modeling Resource Usage of Virtualized Applications", Middleware, 2008.

[9] Forecasting with Exponential Smoothing: The State Space Approach, Hyndman, R., Koehler, A.B., Ord, J.K., Snyder, R.D. 2008, XIII, 362 p

[10] A state space framework for automatic forecasting using exponential smoothing methods International Journal of Forecasting, 18 (2002), pp. 439–454

[11] The fundamental theorem of exponential smoothing, RG Brown, RF Meyer - Operations Research, 1961 or.journal.informs.org

[12] Exponential smoothing: The state of the art, ES Gardner Jr - International Journal of Forecasting, 1985 - Elsevier

[13] Xiaoyan Li, Sharing geoscience algorithms in a Web service-oriented environment, Computers & Geosciences Volume 36, Issue 8, August 2010

[14] Peng Yue, Semantics-based automatic composition of geospatial Web service chains, Computers & Geosciences Volume 33, Issue 5, May 2007

[15] J. Wildstrom, P. Stone and E. Witchel, "CARVE: A Cognitive Agent for Resource Value Estimation", ICAC, 2008.

[16] J. Rao, X. Bu, C. Xu, L. Wang and G. Yin, "VCONF: A Reinforcement Learning Approach to Virtual Machines Auto-configuration", ICAC, 2009.

[17] L. Wang, J. Xu, M. Zhao, Y. Tu and J. A.B. Fortes, "Fuzzy Modeling Based Resource Management for Virtualized Database Systems", MASCOTS. 2011