Resource Management and Fault Tolerance Principles for Supporting Distributed Real-time and Embedded Systems in the Cloud *

Kyoungho An
Institute for Software Integrated Systems (ISIS)
Department of Electrical Engineering and Computer Science
Vanderbilt University, Nashville, TN 37235, USA
kyoungho.an@vanderbilt.edu

ABSTRACT

Cloud computing provides an attractive solution to host enterprise applications due to its cost effectiveness, and its ability to seamlessly adjust to changing application workloads while providing the desired performance assurances using elastic and dynamic resource management. These benefits, however, do not yet readily carry over to distributed, real-time and embedded (DRE) systems, which are a class of systems that require stringent assurances on quality of service (QoS) properties including timeliness, reliability and security all at once. This doctoral research is investigating the sources of these limitations that make it hard to host DRE systems in the cloud, and developing solutions to overcome them. This paper makes three contributions in this regard. First, it outlines the key challenges that must be resolved in supporting DRE systems in the cloud and surveys related literature. Second, it presents ongoing work that addresses one key challenge stemming from the need for real-time and scalable resource monitoring in the cloud. Third, it outlines our proposed ideas on resolving the remainder of the challenges.

Categories and Subject Descriptors

C.4 [**Computer Systems Organization**]: Performance of Systems—*real-time, fault-tolerance, availability*

General Terms

Monitoring, Reliability, Performance

*This work is supported in part by NSF Award CNS 0915976. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Middleware 2012 Doctoral Symposium, December 3, 2012, Montreal, Quebec, Canada.

Keywords

Cloud, DRE systems, Resource Management

1. INTRODUCTION

Cloud computing is a large-scale distributed computing paradigm based on the principles of utility computing that offers various resources such as CPU and storage, systems software, and applications as services over the Internet [1]. The primary driving force behind the success of cloud computing is economy of scale. As the Internet technology evolves, everything is being provided as a service (XaaS) [2], including the well-understood patterns like SaaS (Software as a Service), PaaS (Platform as a Service), IaaS (Infrastructure as a Service), and Database as a Service (DBaaS).

Traditionally although the cloud has been used to support enterprise applications, lately a class of systems called distributed, real-time and embedded (DRE) systems that are mission-critical and require stringent quality of service (QoS) assurances are moving towards being hosted in the cloud [3]. For example, [4] presents how DRE Pub/Sub systems leverage cloud computing. As a motivating example, DRE applications for search and rescue (SAR) operations in a cloud computing environment are used. In addition, [5] introduces an approach to enhancing the SIP/SDP based framework to support DDS-based DRE systems in the cloud including air traffic management, online stock trading, and weather monitoring over QoS-enabled WANs. However, current algorithms and mechanisms used to support applications in the cloud are tailored to meet the performance and reliability requirements of enterprise applications. To address the more stringent QoS requirements of DRE systems, new algorithms and techniques will need to be designed to manage the different cloud platform entities, such as service architecture, data center network architecture, and virtualized resources.

The remainder of this paper is organized as follows: Section 2 discusses the key challenges addressed in this doctoral research and surveys existing literature in this area; Section 3 describes our initial contributions in addressing the resource monitoring requirements for hosting DRE systems in the cloud; Section 4 outlines our proposed ideas to address the unresolved challenges; and finally Section 5 offers concluding remarks.

2. CHALLENGES SUPPORTING DRE SYSTEMS IN THE CLOUD

This section describes the key challenges in supporting DRE systems in the cloud and surveys related literature. By no means are these the only challenges, however, we list only the key challenges we have identified based on limitations in prior work and those that we will resolve as part of the doctoral research.

2.1 Real-time and Scalable Resource Monitoring

Context and Problem: Providing scalable and QoS-enabled (*i.e.*, real-time and reliable) monitoring of resources (both virtual and physical) in the cloud is essential to supporting application QoS properties in the cloud as well as identifying security threats. Existing approaches to resource monitoring in the cloud are based on web interfaces, such as RESTful APIs and SOAP, which cannot provide real-time information efficiently and scalably because of a lack of support for fine-grained and differentiated monitoring capabilities. Moreover, their implementation overhead results in a distinct loss in performance, incurs latency jitter, and degrades reliable delivery of time-sensitive information.

Related Research: Contemporary compute clusters and grids have provided special capabilities to monitor the distributed systems via frameworks, such as Ganglia [9] and Nagios [10]. Additionally, NWS (Network Weather Service) [11] provides a forecasting service for dynamically changing performance of distributed resources. However, these frameworks are structured to monitor physical resources only, and not a mix of virtualized and physical resources. Even though some of these tools have been enhanced to work in the cloud, *e.g.*, virtual machine monitoring in Nagios[1] and customized scripts used in Ganglia, they still do not focus on the timeliness and reliability of the dissemination of monitored data that are essential to support DRE systems in the cloud.

In recent works, [12] presents a virtual resource monitoring model while [13] discusses cloud monitoring architecture for private clouds. Although these prior works describe cloud monitoring systems and architectures, they do not provide experimental performance results of the models, such as overhead and response time. Consequently, we are unable to determine their relevance to host mission-critical applications in the Cloud. Latency results using RESTful services are described in [14], however, they are not able to support diverse and differentiated service levels for cloud clients.

Unresolved Challenges in Prior Work: Prior research illustrates a general lack of resource monitoring capability in the cloud infrastructure that is suitable for hosting mission-critical, real-time applications. For example, the performance of RESTful services described in [14] and [15] do not show promise in using RESTful APIs for the fine-grained and timely monitoring, and dissemination of resource usage information needed to support mission-critical applications in the cloud. Thus, we observe a significant limitation in today's state-of-the-art for cloud resource monitoring, which is the problem we address in this paper. To date our research has identified and developed a solution to address these requirements, which is described briefly in Section 3.

[1] `http://people.redhat.com/~rjones/nagios-virt`

2.2 Time-critical Applications in Data Center Networks

Context and Problem: At the heart of a cloud platform are data centers that provide a large collection of networked resources to host the applications. Assuring timeliness of network flows in data center networks is crucial to complete requested application tasks within expected deadlines. Prior efforts to satisfy deadlines of network flows in data centers can be categorized into two classes: (1) packet scheduling using the Earliest Deadline First (EDF) scheduling policy, and (2) bandwidth reservations [16]. EDF scheduling and rate reservations approaches perform relatively well towards data center networks for time-critical flows, but still incur three challenges. First, deadlines in DRE systems are associated with applications flows, not packets. EDF is packet based, and works on per-hop packet deadlines while applications have end-to-end flow deadlines. Second, data centers today have a diverse mix of flows with widely varying deadlines.

Related Research: Network protocols for data centers is an active area of research. For example, DCTCP (Data Center TCP), which is TCP modified for data center networks [17]. DCTCP realizes better throughput than TCP, reducing queuing delays and congestive packet drops via Explicit Congestion Notification (ECN) to notify feedback to the hosts. However, DCTCP does not work well for deadline sensitive applications as deadlines of network flows are not regarded in the protocol.

Hence, Wilson et al. [6] suggest D^3, a deadline-aware control protocol customized for the data center environment, as a solution to achieve real-time data center networks. D^3 strives to maximize the number of flows that satisfy their deadlines, accommodating burst application workflows, and amplifying network throughput for flows without deadlines. The key insight guiding D^3 design is the following: given a flow's size and deadline, the rate needed to satisfy the flow deadline are determined.

Although D^3 enhances DCTCP by providing deadline-awareness feature, there are two main drawbacks to D^3. First, 24% to 33% of priority of requests are inverted which increases deadline miss ratio. Second, customized hardware is required to use D^3. This shortcoming makes its use hard with commodity TCP and switching hardware used in data centers without using hardware for D^3. Therefore, D^2TCP is suggested to overcome these flaws [18]. D^2TCP adopts a reactive approach for bandwidth allocation. Additionally, ECN and deadlines are used to control congestion. D^2TCP reduces deadline miss ratio of DCTCP and D^3 by 75% and 50%, respectively.

Unresolved Challenges in Prior Work: The recent research on data center networks has been addressing throughput and deadline issues through adjusting protocols between physical server machines and network switches. However, as cloud data center employs virtualization technology, network I/O resources in a single physical machine need to be scheduled properly to realize high throughput and low latency because several virtual machines share I/O resources of a physical machine. Moreover, since a DRE system is likely to be distributed across multiple virtual machines, such assurances must be provided holistically.

2.3 Real-time Scheduling in Hypervisors

Context and Problem: Resource virtualization is a key technology that improves the utilization of resources in the

data center and provides isolation among applications. Virtualization allows physical machine resources to be shared among different virtual machines that have their own operating systems by using a software layer called a hypervisor or a virtual machine monitor (VMM). The hypervisor (VMM) virtualizes the physical resources such as CPUs, memory, networks, and other devices for guest domains, and the guest domains are isolated and scheduled by the hypervisor. As tasks from virtual CPUs are scheduled by the hypervisor, execution and completion time of applications in guest domains are dependent on a scheduling policy selected by the hypervisor, which may not be suitable for real-time tasks.

Related Research: Prior research [19, 20, 21, 22] has focused on achieving real-time computation in virtualized environments. In [19], the Xen hypervisor's credit scheduler is modified to support real-time tasks. In the modified scheduler, deadlines, called laxity in the paper, of domains are used to insert real-time tasks into the scheduler's run queue and the tasks can be scheduled in desired deadlines. To determine the positions of the real-time tasks in the run queue, the expected wait times of all the tasks in the queue need to be maintained, and it is calculated by the amount of CPU time utilized in previous run cycles gained from virtual CPUs. The modified scheduler, however, does not change the credit distribution mechanism of Xen's credit scheduler to prevent starvation.

RT-Xen [22] implements four fixed priority real-time schedulers (Deferrable Server, Periodic Server, Polling Server, and Sporadic Server) in Xen. Experimental results comparing real-time schedulers to the traditional Xen schedulers in terms of overhead and deadline miss ratio are presented in the paper. In the experiments, scheduling overhead including context switch of the suggested schedulers (4 fixed priority real-time schedulers) are about 0.21% which is acceptable for soft real-time systems, but still worse than the general schedulers (credit and SEDF schedulers) which are less than 0.1%. In contrast, deadline ratios of the suggested schedulers are better in both normal and overloaded situations. Specifically, the credit scheduler performs poorly in terms of capacity, missing almost all deadlines even under normal load, while the SEDF scheduler maintains a good capacity with the normal case but comparatively worse than the fixed priority schedulers in most overloaded cases.

Unresolved Challenges in Prior Work: Similar to the shortcomings in prior work on data center networks, related research in real-time scheduling in hypervisors also need to examine performance of network intensive applications with hypervisors where real-time scheduling policies are applied.

2.4 High Availability and Tunable Adaptive Consistency

Context and Problem: Hardware failure in data centers occur frequently, which requires elegant mechanisms to survive the failure to deliver high availability of services demanded by mission critical systems. Special-purpose hardware or re-engineering software to include complicated recovery logic is generally used for unceasing services, but they are expensive and not trivial to be accomplished for the different services with different QoS requirements deployed in the cloud. Therefore, mechanisms involving efficiently replicating virtual machines are needed withing the cloud infrastructure in a general and transparent way.

A commercial product for fail-over protection against virtual machine failures in virtualized environment already exists [26] to provide highly available services in the cloud. However, in the currently available products, only the results written to disk prior to the crash are preserved without the state of CPU and main memory [27]. Consequently, the entire active states, network connections of applications are lost and initiated again. Moreover, recovering a virtual machine does not appear instantly due to the virtual machine's booting time on another host.

Related Research: The solutions presented in [27] address the challenge by making checkpoints of a running virtual machine very frequently, typically tens of times per second. Likewise, [28] presents Remus, a software system that provides high availability via efficient virtual machine replications with extending the technique to make snapshots used for live migrations. Remus achieves it by disseminating frequent checkpoints of an active virtual machine to a backup physical host. On the backup, the image of virtual machine is resident in memory and may immediately begin execution if failure of the active system is detected. Because the backup is only periodically consistent with the primary, all network output must be buffered until state is synchronized on the backup. When a consistent image of the active virtual machine has been received, the network buffer is released to external clients to achieve strong consistency between active and host machines. The virtual machine on the backup host is not actually executed until a failure occurs. Therefore, this consumes a relatively small amount of the backup host's resources.

Kemari [29] is another approach which takes advantage of both lock-stepping and continuous check-pointing approaches. It synchronizes primary and secondary VMs just before the primary VM has to send an event to devices such as storage and networks. At this point, the primary VM pauses and Kemari updates the state of secondary VM to the current state of primary VM. Thus, VMs are synchronized with less complexity compared to lock-stepping and output latency of continuous check-pointing due to external buffering mechanism is also avoided.

Another important work on high availability is HydraVM [30]. It is storage based, memory efficient high availability solution which does not need a passive memory reservation for backups. It uses incremental check-pointing like Remus, but it maintains a complete recent image of VM in shared storage instead of memory replication. Thus, it reduces hardware costs for providing high availability support and provides greater flexibility as recovery can happen on any physical host having access to shared storage.

Unresolved Challenges in Prior Work: The solutions suggested above employ the active replication approach rather than passive primary-backup replication. In systems that use primary-backup replication for fault-tolerance, maintaining system availability after failures refers not just to ensuring the liveness of application functionality at a backup replica but also to ensuring that the state of the promoted backup matches that of the failed primary. DRE systems may demand different levels of availability and state consistency requirements. Consequently, as single scheme as proposed in prior research will not suffice. New algorithms and mechanisms are needed that can tune the replica consistency algorithms at runtime in accordance with the workloads, resource availabilities, and QoS requirements. Additionally, in the current state-of-the-art, there does not exist a flexible

and practical framework, which provides both high availability and acceptable response times to DRE applications while optimizing resource consumption in data centers.

3. REAL-TIME AND SCALABLE RESOURCE MONITORING

Scalable and QoS-enabled monitoring of cloud resources is required to support mission critical applications in the cloud. Contemporary compute clusters and grids have provided special capabilities to monitor the distributed systems via frameworks, such as Ganglia [9] and Nagios [10]. Additionally, [**?**] provides a comparative study of pub/sub middleware for real-time grid monitoring in terms of real-time performance and scalability. According to [**?**], one of the distinctions between grids and clouds is that cloud resources are more abstracted and virtualized compared to grid resources. However, these frameworks are structured primarily to monitor physical resources only, and not a mix of virtualized and physical resources.

Moreover, existing web-based approaches to resource monitoring in the current cloud cannot provide resource information efficiently and scalably because of a lack of support for fine-grained and QoS capabilities. We surmise that publish/subscribe (pub/sub) paradigm can overcome the limitations with existing monitoring and dissemination mechanisms that use RESTful APIs. To that end we have designed an OMG Data Distribution Service (DDS)-based solution called *Scalable and QoS-enabled virtual resource monitoring system for Real-Time applications in Clouds* (SQRT-C) [**?**]. SQRT-C is our solution to address the challenge of real-time monitoring described in Section 2.1.

3.1 Architecture of SQRT-C

Figure 1 illustrates the SQRT-C architecture to describe how the solution interacts with a cloud platform and clients. We have borrowed terminology, such as Cluster Node and Front-end Node, from the OpenNebula [31] open source cloud platform to represent the physical computing entities inside the cloud. SQRT-C uses the DDS-based pub/sub technology to disseminate resource usage information for virtual resources from the source (*i.e.*, publishers) to the sinks (*i.e.*, the subscribers) while also supporting the QoS requirements on the dissemination of the monitored information.

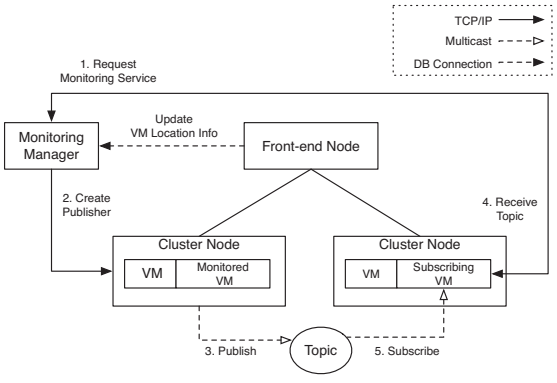

Figure 1: SQRT-C System Architecture

The building blocks of the SQRT-C architecture comprises Publisher, Subscriber, Monitoring Manager, and clients residing in different virtual machines. We refer to clients as the cloud users who will be hosting their QoS-sensitive applications in the cloud and hence will be interested in obtaining timely resource usage information at the specified QoS levels and intervals of time. Each client consists of command line interface APIs to subscribe to the monitoring data from Publishers residing in Cluster Nodes.

Each Cluster Node has a Publisher, which disseminates resource information of virtual machine instances to a Subscriber. A Subscriber is deployed in a client machine (usually a virtual machine) which does auto-scaling and/or provides fault-tolerance for its applications hosted in the cloud. To isolate computation overhead on monitored virtual machines, a Publisher is hosted in a physical Cluster Node and not a deployed virtual machine.

The Monitoring Manager, which is located in the Front-end Node or on an individual physical node (if a database connection is established remotely), serves as an orchestrator to manage DDS connections between Publishers and Subscribers, receiving requests from clients and sending commands to Cluster Nodes.

3.2 Experimental Results

In Figure 2, we demonstrate the average message latency (note the logarithmic scale) comparison between by SQRT-C and RESTful services. We compared the performance with only a RESTful service since it is a conventional communication service used in current cloud platforms for monitoring resources, but comparing with other communication middleware technologies such as RMI, and SOAP would be needed to strengthen our hypothesis in the future.

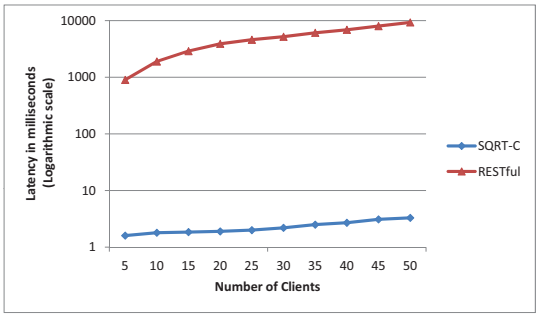

Figure 2: Average Message Latency Comparison of SQRT-C and RESTful by Number of VMs

Since SQRT-C uses a pub/sub model of communication, which is inherently asynchronous and one way, computing average message latency is tricky. We choose an approach where it is calculated by increasing the number of publishers for a single subscriber. This is because in a pub/sub model of communication, publishers and subscribers are decoupled from each other, and thus increasing the number of subscribers does not affect the latency as a publisher does not care about how many subscribers are interested in what it is publishing. However, the number of messages received by a subscriber can be increased by increasing the number of publishers for that subscriber. Therefore, increasing publishers for a single subscriber will cause increase in latency for messages received by the subscriber. We therefore create

a scenario where a single subscriber is subscribing for resource information from up to 50 different virtual instances.

Unlike SQRT-C, RESTful services use a client-server model of communication which is "pull"-based. In case of SQRT-C, the clients or subscribers, and servers or publishers are completely unaware of each other. However, in case of RESTful service, clients and servers have to be aware of each other since clients request for information from the server on a per-requirement basis. Therefore, to calculate average message latency for RESTful service, we increased the number of clients (subscribers) for a single server (publisher) and measured the round-trip latency.

From this figure it is clear that latency increases with increase in number of clients. This is because we have a single physical server which is serving requests from all of the clients. If we compare this result with the results for SQRT-C average message latency, we can see that both of them show linear increase, however, the latencies observed for RESTful services are orders of magnitude larger than those for SQRT-C. For example, the average latency for RESTful service starts from just less than 1,000 milliseconds (compared to the order of just a few milliseconds in the SQRT-C case) for 5 clients and increases significantly to around 9,200 milliseconds for 50 clients. These initial results provide an idea of the significant scalability advantage of SQRT-C over RESTful approaches.

4. PROPOSED IDEAS TO ADDRESSING UNRESOLVED CHALLENGES

Our survey of related research in QoS support in the cloud has focused on 1) timeliness in data-center networks and hypervisors, and 2) high availability via replications of virtual machines. However, resolving the challenges associated with a property such as timeliness and high availability does not mean that the architecture with the suggested solutions can be applied in a straightforward fashion for DRE systems.

In fact, there needs to be a trade-off made between timeliness and high availability with strong consistency. The compromise characteristics between response time and consistency was introduced in the comparison between BASE (Basically Available replicated Soft state with Eventual consistency) and ACID (atomicity, consistency, isolation, and durability) database models. Additionally, in the context of the CAP (Consistency, Availability, and Partition tolerance) theorem, support for extremely rapid responses and fault-tolerance make consistency to be optional for developers, and a justification was made for cloud services with weak consistency or assurance properties [32].

The ACID model has a pessimistic behavior. It will fail if it cannot reach consistency guarantees, and response time is less important than consistency. On the other hand, response time is the most important factor for BASE systems, and consistency may be sacrificed to ensure it. For DRE systems hosted in the cloud, both availability with low latency and strong consistency are significant, and therefore will need a solution that will make effective trade-offs between the conflicting properties depending on service requirements. As a result, realizing fault-tolerant cloud computing architecture satisfying strict timeliness is a challenging research topic. Moreover, in the current cloud computing, there does not exist a flexible and practical framework, which provides both high availability and acceptable response times optimizing resource consumption in data centers.

To realized reliable cloud-based DRE systems, the following will form the doctoral research.

- Experimental analysis to identify the possible tradeoffs will be our first focus. This step will include:
  1. Performance analysis regarding the trade-off between strict timeliness and strong consistency.
  2. Performance analysis of deadline-aware data center networks in virtualized environment.
- Designing and developing a framework for fault-tolerance for cloud-based real-time applications will include:
  1. Investigating the use of replication using primary-backup replication, which is attractive because it consumes fewer resources in comparison to using active replication while delivering comparable performance in optimized conditions.
  2. Investigating the use of a proactive, resource-aware fail-over strategy, which attempts to maximally meet response times of applications by dynamically ordering the fail-over targets based on measured resource utilization [33].
  3. Developing a resource-aware allocation based on backup resource overbooking, which leverages the properties of the primary-backup scheme. In this scheme, a backup replica does not impose the same load on a resource as the primary is exploited to pack more backup replicas of different applications on the available resources [34].

5. CONCLUDING REMARKS

Recent trends in cloud computing reveal an increased push towards supporting DRE systems in the cloud. However, the existing algorithms and mechanisms in the cloud are not suitable to host DRE systems. In this paper we have surveyed the literature that attempt to address one or more of these challenges and outlined a number of open challenges that this doctoral research is attempting to address. Our preliminary results on real-time and scalable monitoring of resources in the cloud are encouraging.

6. REFERENCES

[1] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A View of Cloud Computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[2] B. Rimal, E. Choi, and I. Lumb, "A taxonomy and survey of cloud computing systems," in *INC, IMS and IDC, 2009. NCM'09. Fifth International Joint Conference on.* Ieee, 2009, pp. 44–51.

[3] T. M. Takai, "Cloud Computing Strategy," Department of Defense Office of the Chief Information Officer, Tech. Rep., Jul. 2012. [Online]. Available: http://www.defense.gov/news/DoDCloudComputingStrategy.pdf

[4] J. Hoffert, D. Schmidt, and A. Gokhale, "Adapting distributed real-time and embedded pub/sub middleware for cloud computing environments," *Middleware 2010*, pp. 21–41, 2010.

[5] A. Hakiri, A. Gokhale, D. Schmidt, B. Pascal, J. Hoffert, and G. Thierry, "A sip-based network qos provisioning framework for cloud-hosted dds applications," *On the Move to Meaningful Internet Systems: OTM 2011*, pp. 507–524, 2011.

[6] C. Wilson, H. Ballani, T. Karagiannis, and A. Rowstron, "Better never than late: Meeting deadlines in datacenter networks," in *Proceedings of the ACM SIGCOMM 2011 conference on SIGCOMM*. ACM, 2011, pp. 50–61.

[7] K. Keahey, I. Foster, T. Freeman, and X. Zhang, "Virtual workspaces: Achieving quality of service and quality of life in the grid," *Scientific Programming*, vol. 13, no. 4, pp. 265–275, 2005.

[8] L. Vaquero, L. Rodero-Merino, and R. Buyya, "Dynamically scaling applications in the cloud," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 1, pp. 45–52, 2011.

[9] M. Massie, B. Chun, and D. Culler, "The ganglia distributed monitoring system: design, implementation, and experience," *Parallel Computing*, vol. 30, no. 7, pp. 817–840, 2004.

[10] W. Barth, *Nagios: System and network monitoring*. No Starch Pr, 2008.

[11] R. Wolski, N. Spring, and J. Hayes, "The network weather service: a distributed resource performance forecasting service for metacomputing," *Future Generation Computer Systems*, vol. 15, no. 5, pp. 757–768, 1999.

[12] F. Han, J. Peng, W. Zhang, Q. Li, J. Li, Q. Jiang, and Q. Yuan, "Virtual resource monitoring in cloud computing," *Journal of Shanghai University (English Edition)*, vol. 15, no. 5, pp. 381–385, 2011.

[13] S. De Chaves, R. Uriarte, and C. Westphall, "Toward an architecture for monitoring private clouds," *Communications Magazine, IEEE*, vol. 49, no. 12, pp. 130–137, 2011.

[14] D. Guinard, V. Trifa, and E. Wilde, "A resource oriented architecture for the web of things," in *Internet of Things (IOT), 2010*. IEEE, 2010, pp. 1–8.

[15] J. Meng, S. Mei, and Z. Yan, "Restful web services: A solution for distributed data integration," in *Computational Intelligence and Software Engineering, 2009. CiSE 2009. International Conference on*. IEEE, 2009, pp. 1–4.

[16] C. Aras, J. Kurose, D. Reeves, and H. Schulzrinne, "Real-time communication in packet-switched networks," *Proceedings of the IEEE*, vol. 82, no. 1, pp. 122–139, 1994.

[17] M. Alizadeh, A. Greenberg, D. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, M. Sridharan, C. Faster, and D. Maltz, "Dctcp: Efficient packet transport for the commoditized data center," in *Proc. SIGCOMM*, 2010.

[18] B. Vamanan, J. Hasan, and T. Vijaykumar, "Deadline-aware datacenter tcp (d2tcp)," in *Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication*. ACM, 2012, pp. 115–126.

[19] M. Lee, A. Krishnakumar, P. Krishnan, N. Singh, and S. Yajnik, "Supporting soft real-time tasks in the xen hypervisor," in *ACM SIGPLAN Notices*, vol. 45, no. 7. ACM, 2010, pp. 97–108.

[20] T. Cucinotta, D. Giani, D. Faggioli, and F. Checconi, "Providing performance guarantees to virtual machines using real-time scheduling," in *Euro-Par 2010 Parallel Processing Workshops*. Springer, 2011, pp. 657–664.

[21] A. Masrur, S. Drossler, T. Pfeuffer, and S. Chakraborty, "Vm-based real-time services for automotive control applications," in *Embedded and Real-Time Computing Systems and Applications (RTCSA), 2010 IEEE 16th International Conference on*. IEEE, 2010, pp. 218–223.

[22] S. Xi, J. Wilson, C. Lu, and C. Gill, "Rt-xen: towards real-time hypervisor scheduling in xen," in *Proceedings of the ninth ACM international conference on Embedded software*. ACM, 2011, pp. 39–48.

[23] J. Strosnider, J. Lehoczky, and L. Sha, "The deferrable server algorithm for enhanced aperiodic responsiveness in hard real-time environments," *Computers, IEEE Transactions on*, vol. 44, no. 1, pp. 73–91, 1995.

[24] L. Sha, J. Lehoczky, and R. Rajkumar, "Solutions for some practical problems in prioritized preemptive scheduling," in *IEEE Real-Time Systems Symposium*, 1986, pp. 181–191.

[25] B. Sprunt, "Aperiodic task scheduling for real-time systems," Ph.D. dissertation, Citeseer, 1990.

[26] Vmware high availability. [Online]. Available: http://www.vmware.com/products/high-availability/

[27] D. Petrovic, "Virtual machine replication."

[28] B. Cully, G. Lefebvre, D. Meyer, M. Feeley, N. Hutchinson, and A. Warfield, "Remus: High availability via asynchronous virtual machine replication," in *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*. USENIX Association, 2008, pp. 161–174.

[29] Y. Tamura, K. Sato, S. Kihara, and S. Moriai, "Kemari: Virtual machine synchronization for fault tolerance," *In USENIX ï£¡08 Poster Session*, 2008.

[30] K.-Y. Hou, M. Uysal, A. Merchant, K. G. Shin, and S. Singhal, "Hydravm: Low-cost, transparent high availability for virtual machines," HP Laboratories, Tech. Rep., 2011.

[31] J. Fontán, T. Vázquez, L. Gonzalez, R. Montero, and I. Llorente, "Opennebula: The open source virtual machine manager for cluster computing," in *Open Source Grid and Cluster Software Conference*, 2008.

[32] K. Birman, D. Freedman, Q. Huang, and P. Dowell, "Overcoming cap with consistent soft-state replication," *Computer*, no. 99, pp. 1–1, 2011.

[33] J. Balasubramanian, S. Tambe, C. Lu, A. Gokhale, C. Gill, and D. Schmidt, "Adaptive failover for real-time middleware with passive replication," in *Real-Time and Embedded Technology and Applications Symposium, 2009. RTAS 2009. 15th IEEE*. IEEE, 2009, pp. 118–127.

[34] J. Balasubramanian, A. Gokhale, A. Dubey, F. Wolf, D. Schmidt, C. Lu, and C. Gill, "Middleware for resource-aware deployment and configuration of fault-tolerant real-time systems," in *Real-Time and Embedded Technology and Applications Symposium (RTAS), 2010 16th IEEE*. IEEE, 2010, pp. 69–78.