Cloud Computing for Cyber Physical Systems: Reliability and Security Challenges and Solutions

Kyoungho An, Subhav Pradhan, Faruk Caglar, Prithviraj Patil, Shashank Shekhar, Aniruddha Gokhale
Department of Electrical Engineering and Computer Science
Vanderbilt University, Nashville, TN 37235, USA
Email: {kyoungho.an, subhav.m.pradhan, faruk.caglar, prithviraj.p.patil, shashank.shekhar, a.gokhale}@vanderbilt.edu

***Abstract*—Contemporary cloud computing solutions that have proven beneficial in supporting the needs of enterprise applications do not readily carry over to supporting cyber physical system workflows. For example, existing cloud platforms do not yet provide the stringent assurances on multiple simultaneous quality of service (QoS) properties expected by CPS applications including timeliness, reliability and security. This position paper outlines key challenges we have discovered and promising solutions we are designing to address these needs.**

***Index Terms*—cloud computing, CPS, fault tolerance, security, resource management.**

I. Problem Statement

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]. Traditionally although the cloud has been used to support enterprise applications, lately a class of systems called cyber-physical systems that are mission-critical, tightly integrate physics and the cyber artifacts, and require stringent quality of service (QoS) assurances are moving towards being hosted in the cloud [2].

This section describes the key challenges in supporting CPS 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 are addressing as part of our ongoing research.

A. 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 [3] and Nagios [4]. Additionally, NWS (Network Weather Service) [5] 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, [6] presents a virtual resource monitoring model while [7] 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 [8], 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 [8] and [9] 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.

B. 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 [10]. EDF scheduling and rate

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

reservations approaches perform relatively well towards data center networks for time-critical flows, but still incur two 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 [11]. 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. [12] 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 [13]. 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.

C. 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 [14], [15], [16] has focused on achieving real-time computation in virtualized environments. In [14], 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 [16] 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.

D. 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 within 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 [17] 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 [18]. 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 [18] address the challenge by making checkpoints of a running virtual machine very frequently, typically tens of times per second. Likewise, [19] 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 [20] 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 [21]. 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.

II. Ongoing Research

In current work we have addressed two of the four challenges. For example, in [22] we presented a scalable cloud resource monitoring system called SQRT-C that leverages the OMG's Data Distribution Service (DDS) publish/subscribe technology for efficient resource monitoring and and support QoS. We observed through experimental results that DDS in the cloud as a monitoring service is more appropriate for hosting real-time applications that need fine-grained auto-scaling decisions than widely used technologies like RESTful services. There were a few challenges we uncovered in using DDS for a cloud monitoring service because in the cloud it is preferred that resource information be obtained as a service rather than directly by accessing physical machines or virtual machines by clients. Also, proper configurations of DDS services according to service levels are not trivial to be defined by clients. The Monitoring Manager in SQRT-C is hence indispensable as a public access point and an orchestrator to furnish services appropriately and automate most of the activities.

In another recent work [23], we presented our preliminary work on a high availability framework we are developing. The paper presented the architectural details of a framework for a fault-tolerant cloud computing infrastructure that can automatically deploy replicas of VMs according to flexible algorithms defined by users. Finding an optimal placement of VM replicas in data centers is an important problem to be resolved because it determines the QoS delivered to performance-sensitive applications running in the cloud. To that end the paper presented an instance of an online VM replica placement algorithm we have formulated as an ILP problem.

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] 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

[3] 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.

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

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

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

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

[19] 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.

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

[21] 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.

[22] K. An, S. Pradhan, F. Caglar, and A. Gokhale, "A Publish/Subscribe Middleware for Dependable and Real-time Resource Monitoring in the Cloud," in *To Appear in the Secure and Dependable Middleware for Cloud Monitoring and Management (SDMCMM '12) Workshop at ACM/USENIX/IFIP Middleware 2012*. Montreal, Canada: ACM, Dec. 2012.

[23] K. An, F. Caglar, S. Shekhar, and A. Gokhale, "Automated Placement of Virtual Machine Replicas to Support Reliable Distributed Real-time and Embedded Systems in the Cloud," in *To Appear in the International Workshop on Real-time and Distributed Computing in Emerging Applications (REACTION), 33rd IEEE Real-time Systems Symposium (RTSS '12)*. San Juan, Puerto Rico, USA: IEEE, Dec. 2012.