Strategies for Reliable, Cloud-based Distributed Real-time and Embedded Systems

Kyoungho An
Department of Electrical Engineering and Computer Science
Vanderbilt University
Nashville, TN 37235, USA
kyoungho.an@vanderbilt.edu

***Abstract*—Cloud computing enables elastic and dynamic resource provisioning while providing cost-effective computing solutions. However, while cloud computing provides customers access to scalable and elastic resources, it does not guarantee the user's expectations of Quality of Service (QoS). This is because a number of customers share resources in the cloud infrastructure simultaneously: compute-intensive processes and network traffic associated with one customer often impact the performance of other applications operated on the same infrastructure in unexpected ways. The inability of the cloud to enforce QoS and provide execution guarantees prevents cloud computing from becoming useful for distributed, real-time and embedded (DRE) systems [1].**

Providing the required levels of service to support DRE systems in the cloud is complicated for a variety of reasons: (1) lack of effective monitoring that prevents timely auto-scaling needed for DRE systems, (2) hypervisors and data-center networks that do not support real-time scheduling of resources, and (3) absence of efficient and predictable fault tolerant mechanisms with acceptable overhead and consistency. This paper describes ongoing and proposed doctoral research to address these challenges.

***Keywords*-Cloud Computing, Quality of Service**

I. Ongoing Research

Our ongoing research has focused on real-time and fault-tolerant cloud infrastructures for distributed real-time and embedded (DRE) systems. To overcome Challenge 1 in the abstract, a scalable and QoS-enabled cloud monitoring system (SQRT-C) was developed using OMG Data Distribution Service (DDS) real-time publish/subscribe (pub/sub) middleware. As real-time applications are deployed in the cloud, a finer-grained resource monitoring service is desired for better auto-scaling and fault-tolerant mechanisms. The SQRT-C design comprises two primary software artifacts as shown in Figure 1: DDS-based pub/sub communication and a Monitoring Manager. The DDS-based pub/sub communication is needed to disseminate monitoring 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. The Monitoring Manager serves as the orchestrator for the deployment of data-writers and data-readers of the DDS pub/sub mechanism. To evaluate the performance of our monitoring infrastruture, we compared the response time and jitter between RESTful services that are generally used as the communication middleware in the cloud and our DDS-based SQRT-C. Our results indicate that SQRT-C outperforms RESTful services in terms of response time and jitter for real-time applications.

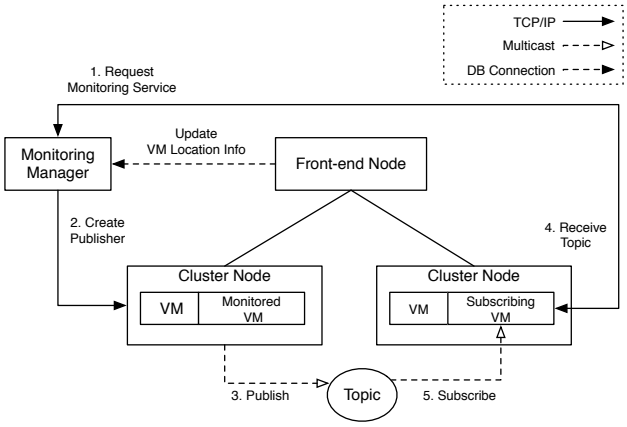

Figure 1. SQRT-C System Architecture

II. Proposed Research

Developing solutions to Challenges 2 and 3 in the abstract require a deep understanding of the problem space. Our survey of past research in this area reveal that these have focused on: 1) timeliness in data-center networks and hypervisors, and 2) high availability via replications of virtual machines. For example, Wilson et al. [2] suggests D^3, which is a deadline-aware control protocol customized for the data-center environment, as a solution to achieve real-time data-center networks with solutions to address EDF (Earliest Deadline First) scheduling and rate reservations in the data center. For timeliness in hypervisors, [3] and [4] propose modifying hypervisor schedulers for real-time computing. For example, [5] presents Remus, which is a software system that provides high availability via efficient virtual machine replications with support for extending the technique to make snapshots used for live migrations.

However, resolving 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 is a trade-off 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 [6].

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 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, since different DRE systems may have different requirements, we need a solution that can be strategized.

Redundancy-based fault recovery mechanisms for DRE systems have been researched in the past. The primary characteristics of the fault recovery mechanisms are the following:

1) **Replication using primary-backup replication** is attractive because it consumes fewer resources in comparison to using active replication while delivering comparable performance in optimized conditions.
2) **A proactive, resource-aware fail-over strategy** attempts to maximally meet response times of applications by dynamically ordering the fail-over targets based on measured resource utilization [7].
3) **A resource-aware allocation based on backup resource overbooking** leverages the properties of the primary-backup scheme, wherein the fact that 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 [8].

Our proposed research related to the fault recovery mechanisms is investigating how these mechanisms can be adapted for virtualized environments to support high-availability of cloud services while optimizing resource usage and satisfying service level agreement (SLA). Consequently, to achieve reliable cloud-based DRE systems, the following steps are needed, which will form the basis of the doctoral research.

- Implementation of fault-tolerant cloud architecture applying redundancy-based fault recovery mechanisms
- Performance analysis regarding the trade-off between strict timeliness and strong consistency
- Integration of real-time hypervisors and deadline-aware data-center networks

ACKNOWLEDGMENTS

I would like to thank my adviser Dr. Aniruddha Gokhale for his valuable comments on this paper. This work was supported in part by the National Science Foundation NSF SHF/CNS Award CNS 0915976, NSF Award CNS 0720736, and Vanderbilt IDEAS grant. 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 or Vanderbilt University.

REFERENCES

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

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

[7] 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.

[8] 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.