Extreme Scale Cloud Computing

NSF Cloud Workshop Position Paper

Justin Y. Shi | shi@temple.edu

October 28, 2014

Abstract

Virtualized computers can deliver higher resource efficiency than traditional dedicated servers via resource sharing by leveraging applications' different resource usage patterns. This efficiency comes with a cost – increased resource volatility. The volatility increases when the cloud applications grow in size or the number of application increases. This position paper critically examines the traditional programming paradigms under the lens of extreme scale cloud computing. We report that the use of virtual circuit (VC) in programming paradigms is the cause of application scalability difficulties. Consequently, the reproducibility of computational results cannot be guaranteed for extreme scale cloud applications. In other words, although VC-based programming paradigms are widely practiced today, they are fundamentally unfit for extreme scale computing due to its inherent scalability limits. This position paper argues for a next generation data-parallel implicit parallel programming paradigm from theoretical and practical aspects. Preliminary computational results are included in support of the proposed paradigm.

1. Introduction

Under the lens of extreme scale computing, the virtual circuit (VC) concept is the first exposed flaw in programming paradigms. This is not a new discovery. Since 1994, Peter Deutsch and others called it "distributed programming fallacies" [28]. Although the harmful effects of the overly optimistic programming practice have been reiterated numerous times, paradigm designers are tempted by the coding ease and small probability of transient failures. However, all designers knew that there was a debt to be paid by the over assumptions somehow in the future. In 2012, the over assumptions were first linked to the program scalability limitations [20].

Due to the programming ease and the small probability of transient failures in small applications, the VC concept has managed to permeate all main stream programming paradigms, such as message passing interface (MPI), share memory (OpenMP), remote procedure call (RPC) and remote method invocation (RMI). Under the lens of extreme scale cloud computing, however, one finds that every VC can fail on any transient software and hardware failure even after the data packets are transmitted correctly. A close look into the communication protocols reveal that the single-thread buffer management layer is unprotected after the packets are received. The lack of retransmission discipline in the application programming paradigms ensures that every transient software/hardware failure on either the sending or receiving host will hang the entire application. The probability of application failure increases exponentially as we upscale the processing infrastructure. The cloud resource volatility worsens the situation.

This discovery explains the mysterious application crashes without obvious reasons. It also inspired the promotion of a data parallel statistic multiplexed computing concept (SMC) [2,6,7,8,11 and 20]. Since VC

is widely used in all applications, including storage systems and data intensive parallel applications, the SMC concept has a broad impact on all applications.

2. Statistic Multiplexed Computing (SMC)

The application-level retransmission discipline has two basic requirements: a) every retransmission must traverse a different path than the last, and b) redundant transmission must be eliminated when necessary. These requirements constitute the essence of statistic multiplexing as found in the packet switching networks. To this date, only implicit data parallel programming paradigm was found to meet these requirements Unlike explicit parallel programs, an implicit parallel programming (and processing) paradigm allows the runtime layer to exploit resources incrementally by statistic multiplexing user data (tuples) – a missing discipline in existing application programming paradigms. The implicit data parallel API decouples the applications from the processing components. It gives the runtime layer a fighting chance to deliver dynamically optimized performance and reliability at the same time. Consequently, not only application reproducibility could be improved, energy efficiency could also be improved and performance scalability limit may be lifted altogether.

As technology developments will inevitably change the future processing, communication and storage methods [3], the explicit parallel paradigms have become counter-productive to scientific research and big data processing applications, especially in light of extreme scale cloud computing and reproducibility of large scale computer applications [1].

3. Proposed Extreme Scale Cloud Computing Research

Comparative performance and reliability studies between the proposed implicit parallel paradigm against explicit paradigms are proposed for large scale cloud environments. The proposed studies promise to deliver a better definition of "scalability" for all applications since only SMC applications can deliver incrementally better performance and reliability at the same time. The declared architecture inflection point [2] seems only applicable to VC-based systems.

It is also interesting to note that only SMC applications can meet the true definition of "mission critical" application since every processing component is exploited for the betterment in performance and reliability of the application.

References

1. Yale Law School Roundtable on Data and Code Sharing, "Reproducibility Research," 2009 http://www.stanford.edu/~vcs/papers/RoundtableDeclaration2010.pdf
2. Justin Y. Shi, "Statistic Multiplexed Computing – A Neglected Path to Unlimited Scalability," 2nd Software Engineering Assembly, National Center for Atmosphere Research (NCAR), April 2, 2013.
3. CCC Community White Paper, "21st Century Computer Architecture," 2012 http://csl.stanford.edu/~christos/publications/2012.21stcenturyarchitecture.whitepaper.pdf
4. Eric Brewer, "Towards Robust Distributed Systems," (CAP Conjecture Keynote) Symposium on Principles of Distributed Computing, 2000.
5. Nancy Lynch and Seth Gilbert, "Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services", *ACM SIGACT News*, Volume 33 Issue 2 (2002), pg. 51-59.
6. Justin Y. Shi, "On CAP, API Design and Big Data Processing Architecture," private communication, 2014.

7. Yuan Shi, “System for High-Level Virtual Computer with Heterogeneous Operating Systems,” US Patent #5,381,534, 1995.
8. Yuan Shi, “A Distributed Programming Model and Its Applications to Computation Intensive Problems for Heterogeneous Environments,” AIP Conference, Earth and Space Science Information Systems, Pasadena, CA., 1992.
9. Justin Y. Shi (Temple), Davide Del Vento (NCAR). “Statistic Multiplexed Computing – A Blueprint for Exascale Computing,” Research Forum Presentation, Supercomputing 2013, Denver, CO., 2013.
10. Justin Y. Shi. “MPI Application Wrapping – A Blueprint for Practical Exascale Computing,” in review for Supercomputing 2014.
11. Moussa Taifi. “Stateless Parallel Processing Architecture for Exascale and Auction-based HPC Clouds,” Ph.D. Dissertation, CIS Department, Temple University. 2013.
12. Justin Y. Shi and Suntian Song, “Apparatus and Method of Optimizing Database Clustering with Zero Transaction Loss”, U.S. Patent Application #US20080046400 A1, February 2008
13. Alan Fekete, Nancy A. Lynch, Yishay Mansour, John Spinelli, “The Impossibility of Implementing Reliable Communication in the Face of Crashes,” Journal of the ACM, 1993. http://65.54.113.26/Publication/787440/the-impossibility-of-implementing-reliable-communication-in-the-face-of-crashes
14. Justin Y. Shi, Moussa Taifi, Abdallah Khreishah, Jie Wu, “Tuple Switching Network – When Slower Maybe Better,” Journal of Parallel and Distributed Computing, 2012. http://www.sciencedirect.com/science/article/pii/S0743731512000263
15. Hadoop, http://hadoop.apache.org/ , 2014
16. ampLab. “Spark - Lighting Fast Cluster Computing,” 2014 https://amplab.cs.berkeley.edu/projects/spark-lightning-fast-cluster-computing/
17. Fanfan Xiong, “High Performance Fault Tolerant Multi-Database System – An Implementation of K-Order Shift Mirrored VLDB”, Ph.D. Dissertation, CIS Department, Temple University, May 2009
18. QNX, “Software Optimization for Multicore Processors,” 2008. http://www.qnx.com/download/feature.html?programid=17599
19. Justin Y. Shi. “Statistic Multiplexed Computing (SMC) -- A Blueprint of Exascale Computing,” Research Exhibit Forum Presentation, Supercomputing 2013, Denver, CO. [online] ***http://cis-linux2.temple.edu/~shi/SC13/SMC-SC13.pdf***
20. Justin Y. Shi. “Program Scalability Analysis for HPC Cloud – Applying Amdahl’s Law to NAS Benchmarks”, High Performance Computing, Networking, Storage and Analysis (SCC) 2012 Supercomputing Conference Companion, pp. 1215-1225, ISBN: 978-1-4673-6218-4, Salt Lake City, Utah, November 2012
21. Tuple Space, [online] http://en.wikipedia.org/wiki/Tuple_space , 2014
22. Tanenbaum and Steen 2006, Distributed Systems: Principles and Paradigms, 2nd Edition, ISBN-10: 0132392275 | ISBN-13: 978-0132392273, Pearson/Prentice Hall.
23. Justin Y. Shi. Chapter 19: Fundamentals of cloud application architectures, “Cloud Computing: Methodology, System, and Applications.” CRC, Taylor & Francis Group, 2011.
24. Byzantine Failure Tolerance. [online] http://en.wikipedia.org/wiki/Byzantine_fault_tolerance
25. Diminishing Returns. [online] http://en.wikipedia.org/wiki/Diminishing_returns
26. Statistical Time Division Multiplexing. [online] http://en.wikipedia.org/wiki/Statistical_time_division_multiplexing

27. Infiniband. [online] http://en.wikipedia.org/wiki/InfiniBand
28. Fallacies of Distributed Computing, [online] http://en.wikipedia.org/wiki/Fallacies_of_distributed_computing