On The Reproducibility of Large Scale Computer Applications

reproducibility@XSEDE: An XSEDE14 Workshop

Justin Y. Shi | shi@temple.edu

April 29, 2014 r1.0 June 11, 2014

A Position Paper

Abstract

For dedicated small scale environments, having replicated code and data are sufficient to reproduce prior results if the processing platform is also compatible with both the code and the data. Experiences indicate that these dependencies impede reproducibility due to rapid technology advances. The current main-stream explicit parallel programming paradigm not only makes the parallel programs closely coupled with the processing platform at the time of programming but also builds growing instabilities into the parallel programs making reproducibility impossible in larger scales even if the processing platform only grows in size. This impossibility is also linked to our inability to quantify parallel application's scalability.

As technology developments will inevitably change the future data processing methods, the explicit parallel paradigm should be considered counter-productive to scientific research, especially in light of reproducibility of scientific results.

This position paper details why and how this long standing problem could be overcome. Preliminary computational results are available in support of the proposed position.

1. Why Large Scale Reproducible Computation Research is Harder?

The 2009 Reproducible Research paper [1] raised the question for needing data and code sharing in order to address the reproducible research challenges for computational sciences. For small scale applications, code and data sharing are considered sufficient for research reproducibility at least for compatible processing platforms. We report that there are practical and theoretical reasons why results of explicit parallel programs are fundamentally not reproducible in large scale even if all codes and data are identically replicated as the original published research.

The first problem is rooted in how we put the parallel programs together. In practice, the packet switching (statistic time multiplexing) principle is responsible for the basic program-to-program communications for all computer applications. All existing application programming interfaces (APIs) rely on "virtual circuits" implemented by the TCP/IP protocol or similar managed buffer protocols, such as SDP (Sockets Direct Protocol), for order preserved, lossless data communication between the communicating programs. A deeper inspection reveals that the "virtual circuit" can break upon any transient software and hardware error [2] in the single-thread of packet (or buffer) management process. According to the impossibility theory of perfect communication [13], using virtual circuit-based APIs makes the resulting applications vulnerable due to its dependency on the rigid network of application-dependent virtual circuits. The entire application will halt on any transient software and hardware failures. For nondeterministic applications, these failures can trigger different logical paths generating different results using the same dataset. The deterministic applications will crash without

obvious cause. These applications suffer performance reproducibility problem and scalability limits for large scale deployments. Bigger scales bring bigger problems. In general, the user can only expect to gain, non-deterministically, either performance or reliability when up scaling the processing infrastructure; a computer architecture inflection point was claimed in 2012 [3]. Reproducibility is harder for larger scale computer applications.

The second problem is how we predict the parallel program we put together would behave. In theory, Amdahl's Law [4] is commonly accepted as parallel application performance scalability measure. However, the lack of communication measure in the commonly accepted Amdahl's Law formulation makes it impossible to reliably predict or justify a parallel application's scalability. Thus, even if all codes and data are identically replicated, the original research results may still not be reproducible if the processing infrastructure is changed. Historically, the law has been claimed "broken" multiple times since the definition of "sequential-to-parallel ratio" is too abstract to map to practical computational experiments [5 and 6].

2. How to Create Reproducible Large Scale Computer Applications?

Since component volatility is unavoidable in large scale computer applications, large scale parallel programming API must not assume static reliable communications. This calls for a new API to allow the application and its processing architecture to statistic multiplex all vulnerable components rendering asymptotically close approximation to the ideal theoretical application desirable states, thus the reproducibility. We call this Statistic Multiplexed Computing or SMC [2].

There are two levels of optimization in order to eliminate nondeterministic factors [14]. The first is to reinstate the retransmission discipline in applications -- a common missing feature in virtual circuit-based applications. Without it, every virtual circuit is a single-point-failure susceptible to transient software or hardware errors that each can halt the entire application. In comparison, there is no single-point failure in a SMC application, since it uses a <key, value> pair API [14] that is statistic multiplexed to harness massively many not-so-reliable processing and networking components. Data parallel semantics are assumed in the <key, value> API and the SMC architecture [7 and 2]. Thus, without NP-hard optimization algorithms, a SMC application runtime can automatically form SIMD, SIMD and pipeline processor clusters at runtime. The second level optimization occurs at runtime by tuning processing granularity. This allows finding the optimal computing/communication overlap experimentally by tuning a single parameter for the entire application [8]. The granularity tuning capability gives each application the reproducibility for arbitrary processing scale. Like the applications using raw packets delivered by a packet switching network, the tuple switching network based SMC application architecture is also inductive. The inductive SMC application architecture is defined on the number of processing and networking components in terms of application's performance and reliability. Adding processing or networking components can deliver better application performance and reliability at the same time incrementally. Thus the scalability limit or the architecture inflection point [3] can indeed be eliminated [9]. Understandably, this unlimited scalability claim must also work under the rule of "diminishing of returns" which is defined by the sizes of computation problems.

Although the <key, value> pair API is universally acceptable to all data processing applications, most current parallel applications are not programed this way. Thus they cannot take advantage of the unlimited scalability and reproducibility benefits directly.

3. SMC Wrapper for Legacy Applications

We have recently completed a SMC wrapper experiment that compares legacy (MPI) parallel performance against wrapped SMC parallel performance for a textbook parallel dense matrix multiplication example [10]. The SMC wrapper was designed to a) eliminate the application downtimes due to transient component failures without cost prohibitive checkpoints, and b) enable tunable processing granularity for optimized parallel processing performance for heterogeneous processors.

We have demonstrated [10] that SMC wrapped MPI program can produce more than 10% better performance after granularity tuning in a small-scale experiment (16 homogeneous processing cores). Larger experiments with higher level of resource volatility will produce bigger performance differences. The inductive nature of SMC application architecture ensures the delivery of unlimited scalability.

Another recent study has shown that most SMC applications can qualify for checkpoint-free parallel statistic multiplexing [11] while still producing the correct results. Thus the SMC wrapped MPI programs are the future exascale parallel applications that are structurally resistant to massive transient failures while delivering incrementally better performance and reliability at the same time. The SMC wrapped applications are theoretically reproducible given the same code and data under any processing scale.

A more practical benefit is that the wrapped MPI programs are capable of exploiting diverse cloud resources under extreme volatile conditions, such as auction-based HPC clouds [11]. This capability was considered impractical before.

The structure of the MPI program wrapper is applicable to OpenMP and other legacy parallel programs. The actual implementation will vary depending upon the involved APIs. The SMC prototype is available on https://github.com/jys673/Synergy30 .

4. Summary

This position paper reports the practical and theoretical reasons for the impossibility of reproducible large scale explicit parallel computer applications. A new paradigm, called Statistic Multiplexed Computing or SMC, is presented to address the reproducibility and scalability problems at the same time. SMC is a paraphrase of the time statistic multiplexed packet switching network. Like network applications using raw packets on a packet switching network are capable of unlimited architectural scalability, SMC applications using <key, value> tuple switching network can also deliver unlimited architectural scalability suitable for exascale computing. A scalable architecture must be inductive on the number of computing and communication elements in terms of deliverable application performance and reliability.

Since most applications today are not coded in <key, value> API, we developed a SMC wrapper for a textbook parallel dense matrix multiplication program. We demonstrated that the SMC wrapped MPI program can produce 10% better performance than the unwrapped version in small scale. The application reliability is exponentially better than the original MPI version. Since <key, value> pair API can be easily supported by any future data processing platforms regardless technologies deployed, the compatibility difficulty is significantly reduced. This improves the long term reproducibility of scientific research.

Improved reproducibility should also inspire quantitative program scalability models. Qualitative performance laws have caused more confusion than being helpful. Quantitative interpretations of qualitative terms are often the culprits.

Today's Internet is a result of a paradigm shift from circuit-switching only network to a combined packet switching and circuit-switching network. Future's extreme scale scientific research needs a similar paradigm shift: from application level virtual circuit networks to application level statistic multiplexed computing or SMC. Explicit parallel programming paradigm has served us well in the past in learning how to program parallel machines. But it is unrealistic to expect a statically optimized program for a dedicated processing environment to run well in different environments. A paradigm shift seems necessary now.

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. Wikipedia, "Amdahl's Law," http://en.wikipedia.org/wiki/Amdahl's_law
5. Wikipedia, "Gustafson's Law," http://en.wikipedia.org/wiki/Gustafson's_law
6. Justin Y. Shi, "Reevaluating Amdahl's Law and Gustafson's Law," 1996. http://spartan.cis.temple.edu/shi/public_html/docs/amdahl/amdahl.html
7. Yuan Shi, "System for High-Level Virtual Computer with Heterogenous Operating Systems," US Patent #5,381,534, 1995.
8. Yuan Shi, "A Distributed Programming Model and Its Applications to Computation Intensive Problesm 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 Applicatoin 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. Yuan Shi, "Program Scalability Analysis," IASTED International Conference on Parallel and Distributed Computing, 1997. http://spartan.cis.temple.edu/shi/public_html/super96/timing/timing.html
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