Scalable Middleware and Tools

Dong Ahn¹, Dorian Arnold¹, Bronis De Supinski², Joshua Goehner¹, Taylor Groves³, Samuel Gutierrez¹,³, Greg Lee², Matthew Legendre², Barton Miller⁵, Philip Roth³, Martin Schulz²

¹University of New Mexico • ²Lawrence Livermore National Laboratory • ³Los Alamos National Laboratory • ⁴Oak Ridge National Laboratory • ⁵Roguewave Software, Inc. • ⁶University of Wisconsin

Contact: Dorian Arnold, darnold@cs.unm.edu

Tree-based Overlay Networks

Tree-based Overlay Networks (TBÕNs) are hierarchically organized process networks used for scalable, high-performance data communication and aggregation. Our open-source TBÕN prototype, MRNet, has an easy-to-use C++ API and supports features like flexible network topologies, customizable data aggregation filters, multiple concurrent data streams and scalable, responsive failure recovery mechanisms. MRNet runs on UNIX/Linux clusters and supercomputers like the IBM BlueGene and Cray XT. As a result of its scalability characteristics, flexibility and wide availability, MRNet has been integrated into a variety of tools and applications and is being used at research organizations throughout the world, including the Lawrence Livermore, Los Alamos and Sandia National Laboratories.

Reliable TBÕNs

Many applications of data aggregation require high degrees of robustness. For example, data analysis tools that use schemes based on anomaly detection risk missing outlier behavior if only partial data are aggregated. However, the expected failure rates of existing and imminent petascale systems have raised serious concerns over current recovery models, which are based on explicit state replication mechanisms like checkpointing or message logging.

In response to these concerns, we have developed a technique called state compensation for robust, high-performance data aggregation in the face of transient or permanent fail-stop failures, detectable failures that cause processes to cease output production. Our central observation is that many TBÕN-based computations naturally maintain redundant state amongst the processes in the system. Intuitively, as information is propagated from the TBÕN leaves to its root, aggregation state, which generally encapsulates the history of processed information, is replicated at successive levels in the tree. State compensation uses this redundant state from processes that survive failures to compensate for information lost due to failures. This approach completely avoids explicit data replication and, generally, only requires that aggregation operations be associative and commutative. We have extended MRNet with an implementation of one of our compensation mechanisms and used this framework to show that for TBÕNs supporting up to millions of application processes, state compensation can yield millisecond failure recovery latencies with unnoticeable application perturbation. Current work in this area includes studying aggregations comprised of heterogeneous filters, implementing other compensation mechanisms and studying the applicability of the state compensation approach for more general computations.

The MRNet-based STAT tool uses the TBON model to execute efficiently at very extreme scales.

Recovery latencies when a TBON process fails.
With depth of 3, 128-way fan-out supports 2
Autonomous Middleware

In developing highly-scalable, generic middleware services for tools and applications, this work seeks to answer two fundamental questions: (1) what are the performance limits of application-agnostic middleware infrastructure? and (2) how efficiently can a developer use middleware infrastructure without any performance considerations or knowledge of the system’s underlying mechanisms. In other words, how well can an infrastructure perform on behalf of an application if there is no application-specific knowledge embedded in the infrastructure, and the application developer is not a computer systems expert.

Our approach to this problem is to use lightweight mechanisms for autonomous middleware infrastructure. Such infrastructure entails monitoring components that compile events that are analyzed to detect functional and performance deficiencies. Decision processes must then be used to determine corrective actions and, in the case of performance failures, evaluate the costs and benefits of system reconfigurations. An important observation is that the definition of a performance failure is dataflow dependent; i.e. the rate of data transmission, the complexity of the aggregation operation, and the topology of the middleware infrastructure determine whether a deficiency exists -- generally defined as an under or over subscription of resources. Accordingly, corrective actions need to be considered on a per dataflow basis.

Large Scale Debugging

The Stack Trace Analysis Tool (STAT), a 2011 R&D 100 Award winner, is a highly scalable, lightweight debugging tool that identifies groups of processes in a parallel application that exhibit similar behavior. STAT gathers and merges stack traces from a parallel application’s processes. The tool produces 3D spatial-temporal callgraph prefix tree profiles based on series of snapshots from the application taken over time.

In 2012, STAT demonstrated successful debugging of a program running over one million MPI processes on the IBM Blue Gene/Q (BGQ)-based Sequoia supercomputer. In this significant accomplishment, STAT has helped both early access users and system integrators quickly isolate a wide range of errors, including particularly perplexing issues that only manifested at extremely large scales up to 1,179,648 compute cores. The STAT team continues to investigate new research as well as convenience features that promise to make the tool even more useful and impactful.

Bootstrapping Large Scale Tools and Applications

The lightweight infrastructure-bootstrapping infrastructure (LIBI) project targets a uniform and scalable bootstrapping process for extreme-scale-software systems. This involves launching processes on a requested set of nodes and propagating relevant initialization information to the launched processes. The LIBI API presents a consistent interface to the programmer while leveraging the native HPC services (like SLURM, ALPS or OpenRTE) when available. This enables application portability while maintaining the speed of the native services. We also developed a novel algorithm (based on our performance model) that determines an optimal bootstrapping strategy. Our algorithm can decrease bootstrap time by up to 50%.

References

http://www.cs.unm.edu/~darnold
http://www.paradyn.org/mrnet
http://www.paradyn.org/STAT

/IPDPS ’10/ Dorian Arnold and Barton Miller, “Scalable Failure Recovery for High-performance Data Aggregation”.
/[SC ’08] Gregory Lee, Dorian Arnold, Dong Ahn, Bronis de Supinski, Matthew Legendre, Barton Miller, Martin Schulz, and Ben Liblit, “Lessons Learned at 208K: Towards Debugging Millions of Cores”.
/[ICPP ’08] Dong Ahn, Dorian Arnold, Bronis de Supinski, Gregory Lee, Barton Miller and Martin Schulz, “Overcoming Scalability Challenges for Tool Daemon Launching”.
/[IPDPS ’07] Dorian Arnold, Dong Ahn, Bronis de Supinski, Gregory Lee, Barton Miller, and Martin Schulz, "Stack Trace Analysis for Large Scale Applications”.
/[HIPS ’06] Dorian Arnold, Gary Pack and Barton Miller, ”Tree-based Overlay Networks for Scalable Applications”.
/[SC ’03] Roth, Arnold, and Miller, “MRNet: A Software-Based Multicast/Reduction Network for Scalable Tools”.

STAT identifies bug at 1 Million MPI Tasks on Sequoia!