SimMatrix: SIMulator for MAny-Task computing execution fabRIc at eXascale

Ke Wang
Department of Computer Science
Illinois Institute of Technology
Chicago, IL, USA
kwang22@hawk.iit.edu

Kevin Brandstatter
Department of Computer Science
Illinois Institute of Technology
Chicago, IL, USA
kbrandst@hawk.iit.edu

Ioan Raicu
Department of Computer Science
Illinois Institute of Technology
Chicago, IL, USA
iraicu@cs.iit.edu

ABSTRACT

Exascale computing have challenges, most of which can be potentially addressed by Many-task computing paradigm through efficient task execution frameworks that are several orders of magnitude beyond current batch schedulers. This paper proposes a light-weight discrete event simulator, SimMatrix, which simulates distributed job scheduler comprising of millions of nodes and billions of cores/tasks. We validated SimMatrix against MATRIX up to 4K-cores, running on an IBM Blue Gene/P system, and compared SimMatrix with SimGrid and GridSim in terms of resource consumption at scale. Results show that SimMatrix consumes up to two-orders of magnitude lower memory per task, and at least one-order of magnitude (and up to four-orders of magnitude) lower time per task overheads.

1. INTRODUCTION

With exascale computing, we expect that applications running on exascale machines would be decomposed with large number of tasks with finer granularity in size and duration, along with large volume of data. Driven by these expectations, Many-Task Computing (MTC) was proposed [1] to bridge the gap between HPC and HTC. Many MTC applications are structured as graphs of discrete tasks, with input and output dependencies forming the graph edges. MTC applications often demand a short time to solution, may be communication intensive or data intensive [2]. For many applications, a graph of distinct tasks is a natural way to conceptualize the computation.

Task execution framework for exascale systems will have to be much more scalable and flexible to handle both HPC and MTC. We fall back to simulations to study various scheduling architectures and algorithms at exascale. In Discrete event simulation (DES), the operations of a system are represented as a chronological sequence of events. We propose SimMatrix that simulates job scheduler comprising of millions of nodes and billions of cores/tasks. SimMatrix supports centralized (FIFO) and distributed (work stealing) scheduling at node/core level.

2. RELATED WORK

The earliest batch job schedulers are Condor [3], Slurm [4]. All these systems target as the HPC or HTC applications, and lack the granularity of scheduling jobs at node/core level, making them hard to be applied to the MTC applications. What's more, the centralized dispatcher in these systems suffers scalability and reliability issues. Falkon [5] is a light-weight task execution framework with both centralized and hierarchical architectures for MTC workload, and although it scaled and performed several orders magnitude better than the traditional batch schedulers, it even cannot scale to petascale systems [6]. Sparrow [7] is another hierarchical task execution framework targeting at sub-second tasks. However the Java-based framework is very hard to be deployed on supercomputers.

Popular simulators for distributed systems are SimGrid [8], GridSim [9]. SimGrid uses PDES and claims to have 2M nodes' scalability. However, it is neither suitable to run exascale MTC applications, due to the complex parallelism. GridSim uses multi-threading (one thread per simulated element), making it impossible to reach exascale on a single shared-memory system.

3. SIMMATRIX

SimMatrix is a DES for MTC execution fabric at exascale. The architectures of SimMatrix are shown in Figure 1.

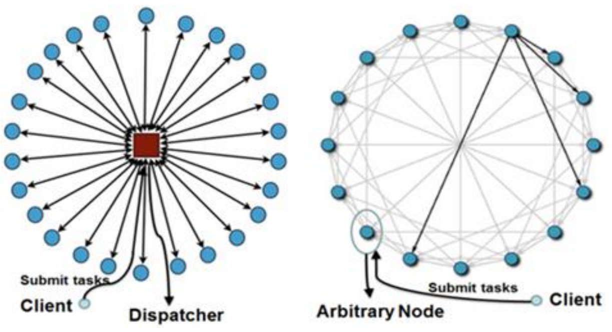


Figure 1: SimMATRIX architectures

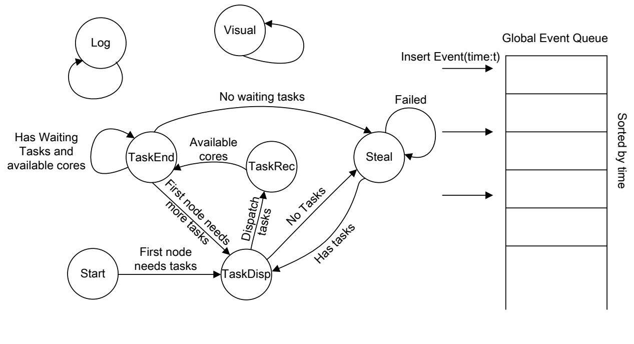


Figure 2: State transition diagram of SimMatrix

For simplicity, we assign consecutive integer numbers as the node ids, ranging from 0 to the number of node N-1. The system could be centralized (Figure 1 left part), where a single dispatcher maintains a task queue and manages the task submission, task assignment, and task execution state updates. It could also be distributed (Figure 1 right part), where each computing node maintains a task execution framework, and they cooperate with each other to achieve load balancing through work stealing technique. The global event queue is the core part of SimMatrix. Any behavior is converted to an event and put in the queue that is sorted based on the occurrence time. We advance the simulation time to the occurrence time of the first event removed from the queue. The state transition diagram of all the events are shown in Figure 2, where each state is an event that is executed, the next state is the event to be inserted in the event queue signaled after finishing the current event. For example, if the current event is "TaskEnd", meaning that a node finishes a task and has one more available core. It will insert another "TaskEnd" event for the available core, or steal tasks from neighbors.

4. PERFORMANCE EVALUATION

We validate SimMatrix against MATRIX [10], a real task execution framework for MTC, up to 4K-core scale running on an IBM Blue Gene/P machine. The result is shown in Figure 3. For SimMatrix, each node is configured to have 4 cores; the number of tasks is 10 times of the number of cores, and we use "sleep 0" tasks. The simulation matched the real performance data with average 5.85% normalized difference (abs(SimMatrix - MATRIX) / SimMatrix), a relatively small amount of error.

We also compare SimMatrix with SimGrid and GridSim in terms of memory and time consumption. The results are depicted in Figure 4. Notice that for GridSim, we just scaled up to 256 nodes, as it took significant time to run larger scales. The time per task of GridSim is increasing as the system scales up, while SimMatrix and SimGrid experienced decreasing or constant time per task. SimGrid could scale up to 65K nodes, however, after which point it ran out of memory (256GB). SimMatrix scales up to 1M nodes without any problems, and it is likely to simulate even greater scales with moderate resource requirement.

5. CONCLUSION & FUTURE WORK

Efficient task execution frameworks are needed to address the challenges of exascale computing with MTC. We developed a light-weight and scalable DES of JMS, SimMatrix, at exascale. The scalability and resource consumption of SimMatrix are significantly better. In the future, we plan to explore more complex network topologies for exascale systems, such as Fat Tree, 3D/4D/nD Torus, and InfiniBand. We believe SimMatrix could also be developed to simulate workflow systems, and it would allow us to study job dependency and data aware scheduling with more realistic constraints.

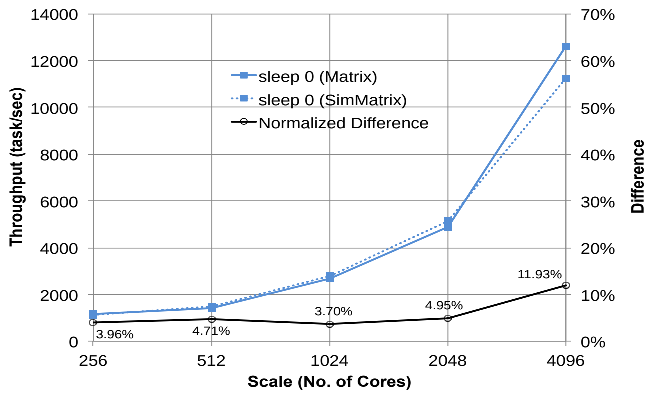


Figure 3: Validation of SimMatrix against MATRIX

6. REFERENCES

[1] I. Raicu, Y. Zhao, I. Foster, "Many-Task Computing for Grids and Supercomputers," 1st IEEE Workshop on Many-Task Computing on Grids and Supercomputers (MTAGS) 2008.

[2] I. Raicu et. al. "Towards Data Intensive Many-Task Computing", book chapter in Data Intensive Distributed Computing: Challenges and Solutions for Large-Scale Information Management, IGI Global Publishers, 2011.

[3] J. Frey et. al. "Condor-G: A Computation Management Agent for Multi-Institutional Grids," Cluster Computing, 2002.

[4] M. A. Jette et. al, Slurm: Simple linux utility for resource management. Proceedings of Job Scheduling Strategies for Prarallel Procesing (JSSPP) 2003 (2002), Springer-Verlag, pp. 44-60.

[5] I. Raicu, et. al. "Falkon: A Fast and Light-weight tasK executiON Framework," IEEE/ACM SC 2007.

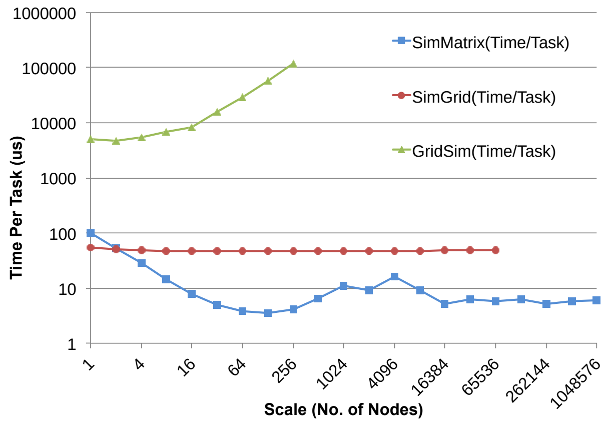
[6] I. Raicu, et. al. "Toward Loosely Coupled Programming on Petascale Systems," IEEE SC 2008.

[7] K. Ousterhout et. al. "Batch Sampling: Low Overhead Scheduling for Sub-Second Prallel Jobs." University of California, Berkeley, 2012

[8] M. Quinson et. al "Parallel Simulation of Peer-to-Peer Systems." In Proceedings of the 12th IEEE International Symposium on Cluster Computing and the Grid (CCGrid'12), May 2012.

[9] R. Buyya and M. Murshed. "GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing," The Journal of Concurrency and Computation: Practice and Experience (CCPE), Volume 14, Issue 13-15, Wiley Press, Nov.-Dec., 2002.

[10] K. Wang et. al. MATRIX: MAny-Task computing execution fabRIc at eXascales. http://datasys.cs.iit.edu/projects/MATRIX/index.html, 2013.



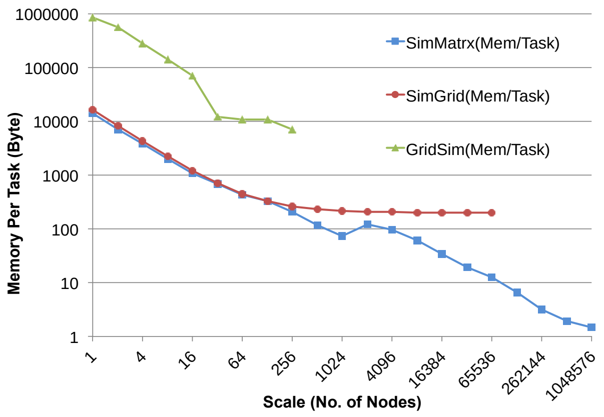


Figure 4: Resouce consumption comparions among SimMatrix, SimGrid, and GridSim