SimMatrix

Exascale computers will enable the unravelling of significant scientific mysteries. Predictions are that by 2019, supercomputers will reach exascales with millions of nodes and billions of threads of execution. Many-task computing (MTC) is a new viable distributed paradigm for extreme-scale supercomputing. The MTC paradigm can address four of the five major challenges of exascale computing, namely concurrency, resilience, heterogeneity, and I/O and memory; this work specifically addresses the first three major challenges.

SimMatrix is a new light-weight and scalable discrete event simulator that enables the exploration of distributed scheduling for MTC workloads at exascale levels with up to 1 million nodes and 1 billion cores. SimMatrix is validated against real MTC workloads executed under Falkon at peta-scale levels, with 40K nodes and 160K-cores.

Centralized scheduling is compared and contrasted to distributed scheduling; this work adopts work stealing, as an efficient and scalable approach to distributed load balancing. It explores a wide range of parameters important to understand work stealing at exascale levels, such as number of tasks to steal, number of neighbors of a node, static or dynamic neighbors, and different workloads. Experiment results show that the centralized scheduling saturates at small number of nodes, while the distributed scheduler configured with optimal parameters could scale up to 1 million nodes and 1 billion cores without any explicit upper bound. SimMatrix is light-weight and scalable, having been tested up to 1 billion cores and 10 billion tasks with modest resources (e.g. 200GB of memory and 256-core hours).

Work Stealing

Work stealing refers to a distributed load balancing approach in which processors needing work steal computational tasks from other loaded processors. There are several parameters which could affect the performance of work stealing to achieve load balancing, such as steal tasks from global space or just some neighbours, how to select neighbours, how many number of neighbours a node could have, how many tasks to steal, and the length of waiting time if a node fails to steal takes from others. Work stealing has been used at small scales successfully in parallel languages such as Cilk, to load balance threads on shared memory parallel machines. Theoretical work has proved that a work-stealing scheduler can achieve execution space, time, and communication bounds all within a constant factor of optimal. However, the scalability of work stealing has not been well explored on modern large-scale systems. In particular, concerns exist that the randomized nature of work stealing can lead to long idle times and poor scalability on large-scale clusters. The largest studies to date of work stealing have been at thousands of cores scales, showing good to excellent efficiency depending on the workloads. Our work from the simulation perspective shows that work stealing with optimal parameters we found works great for even exascale systems at millions of nodes and billions of cores.

Contributions

(1) Developed a scalable light-weight discrete event simulator that enables distributed scheduling for exascale MTC workloads.
(2) Provided evidence that work stealing is a scalable method to achieve load balance, even at exascales.
(3) Identified optimal parameters affecting the performance of work stealing; at the largest scales, in order to achieve the best work stealing performance, we found the number of tasks to steal is half and there must be a squared root number of dynamic neighbors (e.g. at 1M nodes, we would need 1K neighbors).