Towards Loosely Coupled Programming on Petascale System

CS554 – Data Intensive Computing CWID – A20402686

Summary of the paper

This paper describes the use of Petascale system for tightly and loosely coupled parallel computations. Now a day, more scientists rely on scripts for automating the processes with provenance tracking, task coordination and bookkeeping. This paper focuses on problem in which data size is modest and number of tasks are increased i.e. loosely coupled application with many tasks (e.g. Falkon/Swift), on supercomputers with thousands of processors. The application consists of many tasks is called Many Task Computing (MTC). MTC is different than High Performance Computing (HPC) in using large number of resources for small time to complete many computational tasks. Generally, metrics of MTC are in seconds and for HPC, it is months and years. Most MTC applications are performance oriented. MTC applications varies from simple to complex and communication intensive. There are many point of supporting MTC applications on Petascale HPC systems.

Petascale system can provide unique capabilities for MTC applications like high bandwidth and low latency interconnects. Also cost to run on Petascale system is less than that of cluster and grid. It is desirable to run loosely coupled applications on idle portions of petascale systems because Large-scale systems have utilization issues. Also, there are some applications that only petascale systems have enough compute power to get output in desired timeframe. Large scale and loosely coupled applications are executed on petascale systems using multi-level scheduling, efficient task dispatch and extensive use of caching to minimize shared infrastructure. Swift and Falkon have been used in various environments like cluster, multi-site grid, supercomputers. To work Falkon on Blue Gene/P, static resource provisioning is used in which application requests a number of processors for a fixed duration directly from the Cobalt LRM. Performance also depends on task dispatch mechanism. Some functionality of Falkon is implemented in Java and some in C. TCP protocol is used between dispatcher and executor.

Initially, Falkon architecture consist of single dispatcher to manage many executors. Overload of login nodes, I/O nodes and compute nodes led to the offloading of dispatcher from one login node to many I/O nodes. Single dispatcher can handle thousands of tasks per second and tens of thousands of executors with 4 to 8 cores at 2GHz+ and 2GB+ of memory. Then processors are increased to 160K and centralized design comes with limitation of scaling a single dispatcher to 160K executors. To reduce the load of login node, number of dispatchers are increased. Also, system admin of Blue Gene/P did not allow high utilization of login node for more time period for larger MTC applications. There are also some reliability issues for large scale problems. There is 10 days of mean-time-to-failure (MTBF). However, Falkon has mechanism to identify errors and act on them with actions.

How is this work different than the related work?

Cope system integrated their work in Cobalt scheduling system instead of Falkon. Implementation of Cope was on Blue Gene/L using HTC model instead of MTC which is used in Falkon. There was performance difference of one order of magnitude worse at small scale and performance gap increases with large scale because they used approach of high overheads. Peter's et al. also shows same performance difference between HTC mode of Blue Gene/L and Falkon MTC on Blue Gene/P. This paper focuses on systems with single highly parallel machines but systems like Condor, MapReduce, Hadoop, and BOINC have used highly distributed pools of processors. MapReduce is processed at programming language level and applied to name/value pairs data model, it involves development of filtering scripts and it is less capable of utilization. The work mentioned in this paper focuses on efficiency and performance, but Condor focuses more on robustness and recoverability which causes less efficiency for large scale MTC application. Task farming is evaluated on Blue Gene/L instead of Blue Gene/P, but it does not offer performance evaluation for comparison.

Identify the top 3 technical things this paper does well.

- **Many-Task Computing (MTC)** – MTC is used instead of High Throughput Computing (HTC) for using large number of computing resources for short period to complete tasks in metrics of seconds.
- **Task Dispatch Performance –** Utilization of large scale systems is achieved using great task dispatch performance.
- **Number of Dispatcher –** Number of dispatchers are increased from one login node to many I/O nodes for performance improvement.

Identify 3 things the paper could do better.

- Paper could implement the mechanism for trouble in scaling a single dispatcher to 160K executors.
- There should be mechanism to improve efficiency at small scale and short tasks while distributing the Falkon dispatcher from one login node to many I/O nodes.
- Falkon should be more dynamic, reliable and provide natural flow as compared to Swift.

If you were to be an author of a follow up paper to this paper, what extensions would you make to improve on this paper?

If I was an author of this paper, I would have included the method to increase the efficiency at small scale and short tasks while distributing the Falkon dispatcher from one login node to many I/O nodes.