Falkon

Summary

The papers comprehensively discuss about the Falkon and how it is extended to make loosely coupled programming a reality on petascale system such as IBM Blue Gene/P supercomputer, a practical and useful programming model. Generally, we use batch schedulers in clusters to receive individual tasks, dispatch them to processors notify clients when execution is done. There are many limitations associated, as the most important one is the time required to dispatch a task can be as large as 30 sec or more. To cope up with these limitations Falkon (Fast and light weight task execution framework) is proposed. It uses a dispatcher to receive, enqueue, and dispatches tasks; a task executor to receive and executes task; a provisioner to allocate and deallocate executors. Microbenchmarks show that Falkon can process millions of task and scale to 54000 executors.

The paper mentions the problems we face when we keep are data size modest but increase the number of tasks which falls into the category of loosely coupled applications involving many tasks. This leads to the new class of applications that is Many-Task Computing(MTC). MTC are composed of many tasks both dependent and independent that can be schedule individually on different nodes over the network. MTC is somewhat like High Throughput Computing (HTC), only difference is MTC emphasizes on using large number of computing resources over a shorter period of time. The paper provides a hypothesis that MTC applications can be executed efficiently on emerging petascale computers with evidence and proof. Some of the problems like local resource manager scalability and granularity, efficient utilization of the raw hardware, shared file system contention, and application scalability must be overcome to make it all work. The paper has validated the hypothesis on two systems Swift and Falkon which are known for large-scale loosely coupled applications on clusters and Grids. Microbenchmarks and real application is executed on Blue Gene/P which shows that it can be scaled up to 160k processors core. The application has excellent scalability, speedup and efficiency as they scale to 128K cores when tested on different domains: economic energy modelling and molecular dynamics. The paper also presented challenges of I/O performance while implementing the concept. Lastly the paper discusses about the implementation of loosely coupled application on Blue Gene/P using Falkon distributed architecture. Using Falkon multilevel scheduling and light weight executor much of the insignificant overhead is avoided. Falkon extensive caching technique also allows the application to scale by minimizing the use of shared file systems.

How is this work different than the related work

There are some full featured Local Resource Managers (LRMs) such as Condor, Portable Batch System (PBS), Load Sharing Facility, and SGE. These LRMs focuses on process migration, client requirements, check pointing, accounting and demon fault recovery. On the other hand, Falkon is not full featured LRM as it focuses more on task dispatching and executing but it can rely on LRMs for certain features like accounting, client specification. BOINC's task dispatcher can dispatch 8.8M tasks per day to 400K workers, whereas Falkon can execute 2M (trivial) tasks in two hours, and has scaled to 54K managed executors with similarly high throughput. Frey et al works on Candor which is a batch scheduler while MyCluster creates personal cluster on Candor or SGE. Singh et al. measures that the reduction in queue wait time can reduce completion time up to 50%. Ramakrishnan et al address adaptive resource provisioning with a focus primarily on resource sharing and container level resource management. Bresnahan et al. states a multi-level scheduling architecture specialized for the dynamic allocation of compute cluster bandwidth.

There has been some advancement in the field of loosely coupled programming focusing on HTC on the IBM Blue Gene /L. Cope et al. focuses on incorporating the solution in the Cobalt scheduling system instead of another system such as Falkon. They implemented the solution on the Blue Gene/L using the HTC-mode support in Cobalt, result of the performance study was done at a small scale (64 nodes, 128 processors). The results were at least one order of magnitude worse at small scales than the results of Falkon support for MTC workloads on the Blue Gene/P and the performance gap would only increase with larger-scale tests as their approach has higher overheads (i.e., nodes reboot after each task, in contrast with simply forking another process). Peter's et al. from IBM shows similar results which when tested on the HTC- mode. Mapreduce is another system which is like what we used in this paper. Mapreduce can spread the large datasets to thousands of processors but is not easy to utilize and it often involves development of custom filtering. The paper has put forward the points on performance and efficiency. As discussed earlier Candor focuses on robustness and recoverability which limits its efficiency for MTC applications in large- scale systems. Reid mentions "task framing" approach at the programming language level, evaluated on the Blue Gene/L as a proof of concept, but offered no performance evaluation for comparison, and required that applications be modified to run over the proposed middleware.

Identify the top 3 technical things this paper does well

The three things this paper does well are as follows:

- Implementation of multilevel scheduling on petascale system using Falkon and Swift.
- Detail explanation of the architecture of the Falkon framework and its components like dispatcher, executor and provisioner and working of each component.
- Evaluation of GPFS throughput using Falkon on various file sizes.

Identify 3 things the paper could do better

The three things this paper could do better are as follows:

- Performance of HTC on IBM Blue Gene/L and MTC on Blue Gene/P could have been compared using linear graph.

- Paper could have discussed how cost of read/write operations in petascale class system like Blue Gene/P supercomputers can be affected if tasks are scheduled concurrently.
- Paper could have discussed the implementation of collective I/O operations to use specialized high bandwidth and low latency interconnect like Torus network.

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?

MTC applications uses files for intercrosses communication and files are transferred from one node to another. We can use data diffusion, TCP pipes or MPI messages directly between the compute nodes to remove this bottleneck.