CloudKon: DTS

Distributed Task Scheduling with Amazon STACK

Contents

[Table of Figures 1](#_Toc372407868)

[Abstract 2](#_Toc372407869)

[Introduction 2](#_Toc372407870)

[Background 2](#_Toc372407871)

[References 2](#_Toc372407872)

# Table of Figures

[Figure 1 : Architecture Diagram 4](#_Toc370105379)

# Abstract

Predictions are that by the end of this decade, we will have exascale system with millions of nodes and billions of threads of execution[1]. Task Scheduling and execution of tasks over these large scale, distributed systems plays an important role in achieving good performance and high system utilization. Many Task computing paradigm[2] aims to bridge the gap between High Performane Computing and High Throughput Computing . Tasks may be small or large, uniprocessor or multiprocessor, compute-intensive or [data-intensive](http://en.wikipedia.org/wiki/Data-intensive_computing) but MTC tasks include loosely coupled communication intensive tasks. Today’s jobschedulers have centralized Master/Slaves architecture (e.g. Slurm, Condor, PBS,SGE), where a centralized server is in charge of the resource provisioning and job execution. This architecture has worked well in modest scales and coarse granular workloads, but it has poor scalability at the extreme scales of petascale systems with fine granular MTC workloads. The goal of this project is to provide a efficient light weight and scalable distributed execution framework built on built on open source stack[HazleCast,Cassandra] to address MTC workloads deployed over Amazon Ec2 instance in cloud environment

# Introduction

The goal of an execution fabric is to effectively utilize the execution system aiming towards high throughput and also provide efficient results for executed tasks. Today’s workload involves a lot of fine granular workloads with execution times in seconds. Centralized schedulers are optimized towards high computational massive tasks where the complex decision policy and architecture of the schedulers play a major role. But they tend to add considerable overhead while scheduling these lots of small tasks. Moreover the centralized architecture tends to be a bottleneck in scheduling and execution. The solution to this problem is to have a decentralized and simple architecture. A decentralized architecture avoids the single point of failure, while a simple architecture reduces the considerable overhead involved in decision making for scheduling.

The execution fabric requires lot of computing resources to execute the ever growing workload of today’s world. Cloud computing seems to be a viable solution to this problem. Our solution is to build a loosely coupled compact and distributed execution fabric over public cloud (Amazon Ec2 instance) with distributed building blocks such as Cassandra [3] and HazleCast [4]. The motivation behind not making extensive use of AWS [Amazon Web Services] is to decouple the fabric from AWS and provide easier transition to private cloud environment.

Recent studies suggest that clouds were not suitable candidates for scientific HPC computing [5]. The problems listed were largely due to following the same approach involved in traditional clusters and grids. Our work involves running applications which are optimized for cloud environment. Traditional workloads can also be run on our execution fabric but with suitable decomposition of the workload.

# Background

# References

1. P. Kogge, et. al., “Exascale computing study: Technology challenges in achieving exascale systems,” 2008
2. IEEE Workshop on Many-Task Computing on Grids and Supercomputers (MTAGS08)
3. HazleCast <http://www.hazelcast.com/>
4. Cassandra <http://cassandra.apache.org/>
5. Q. He, S. Zhou, B. Kobler, D. Duffy, and T.McGlynn. “Case study for running HPC applications in public clouds,” In Proc. of ACM Symposium on High Performance Distributed

Computing, 2010.

1. D. Thain, T. Tannenbaum, M. Livny, “Distributed Computing in Practice: The Condor Experience” Concurrency and Computation: Practice and Experience 17 (2-4), pp. 323-356, 2005.
2. M. A. Jette et. al, Slurm: Simple linux utility for resource management. In Lecture Notes in Computer Sicence: Proceedings of Job Scheduling Strategies for Prarallel Procesing (JSSPP) 2003 (2002), Springer-Verlag, pp. 44-60
3. B. Bode et. al. “The Portable Batch Scheduler and the Maui Scheduler on Linux Clusters,” Usenix, 4th Annual Linux Showcase & Conference, 2000.
4. I. Raicu, et. al. “Falkon: A Fast and Light-weight tasK executiON Framework,” IEEE/ACM SC 2007
5. CloudKon <http://www.cs.iit.edu/~iraicu/teaching/CS554-F13/proj/CloudKon-DTS.pdf>
6. ZHT: A Light-weight Reliable Persistent Dynamic Scalable Zero-hop Distributed Hash Table, IPDPS 2013