CloudKon: CKHPC

Extending CloudKon to support HPC jobs scheduling

By:- Karthik Belgodu
Pankaj Purandare
Isha Kapur

Contents

- Goal of the Project
- Motivation
- Background
- Terms Used
- Implementation
- Architecture
- Minimum Messages Required
- Throughput Evaluation

Contents continued...

- Comparison Results
- Latency Evaluation
- Highlights
- Future Work
- Conclusion
- References

Goal of the Project

- Create a tightly coupled implementation of CloudKon for HPC environments, using Amazon Services, which is highly scalable and gives high performance for both MTC and HPC jobs.
- Make sure that current performance of the system for MTC tasks is not hampered to a large extent by our code.
- Extend CloudKon to act as a distributed job management system for HPC that can support millions of tasks from multiple users delivering over 2X the performance compared to other systems like Slurm in terms of throughput.

Motivation

- Eliminate the need to have high performance private grids or clusters when the system can be implemented on the “Cloud”
- Moving away from age old paradigms like master/slave architectures that have single point of failure, bottleneck and scalability issues.
- Stop using techniques like random sampling, resource/work stealing and hierarchical system and use distributed queues and NoSQL database services.

Background

- **CloudKon**: A job management system tailored to run MTC jobs on AWS.
- **Amazon Web Services**
  - SQS: Highly Scalable Distributed Queue
  - EC2:Resizable, pay-as-you-go compute capacity
  - DynamoDB: High performance NoSQL Database configured for high read and write throughputs.
- **MTC**: Using many computing resources over short periods of time
- **HPC**: Aggregating computing power in a way that delivers much higher performance.
- **Google Protocol Buffer**: Language-neutral, platform-neutral, extensible mechanism for serializing structured data
- **RMI**: Java API that performs the object-oriented equivalent of remote procedure calls (RPC)

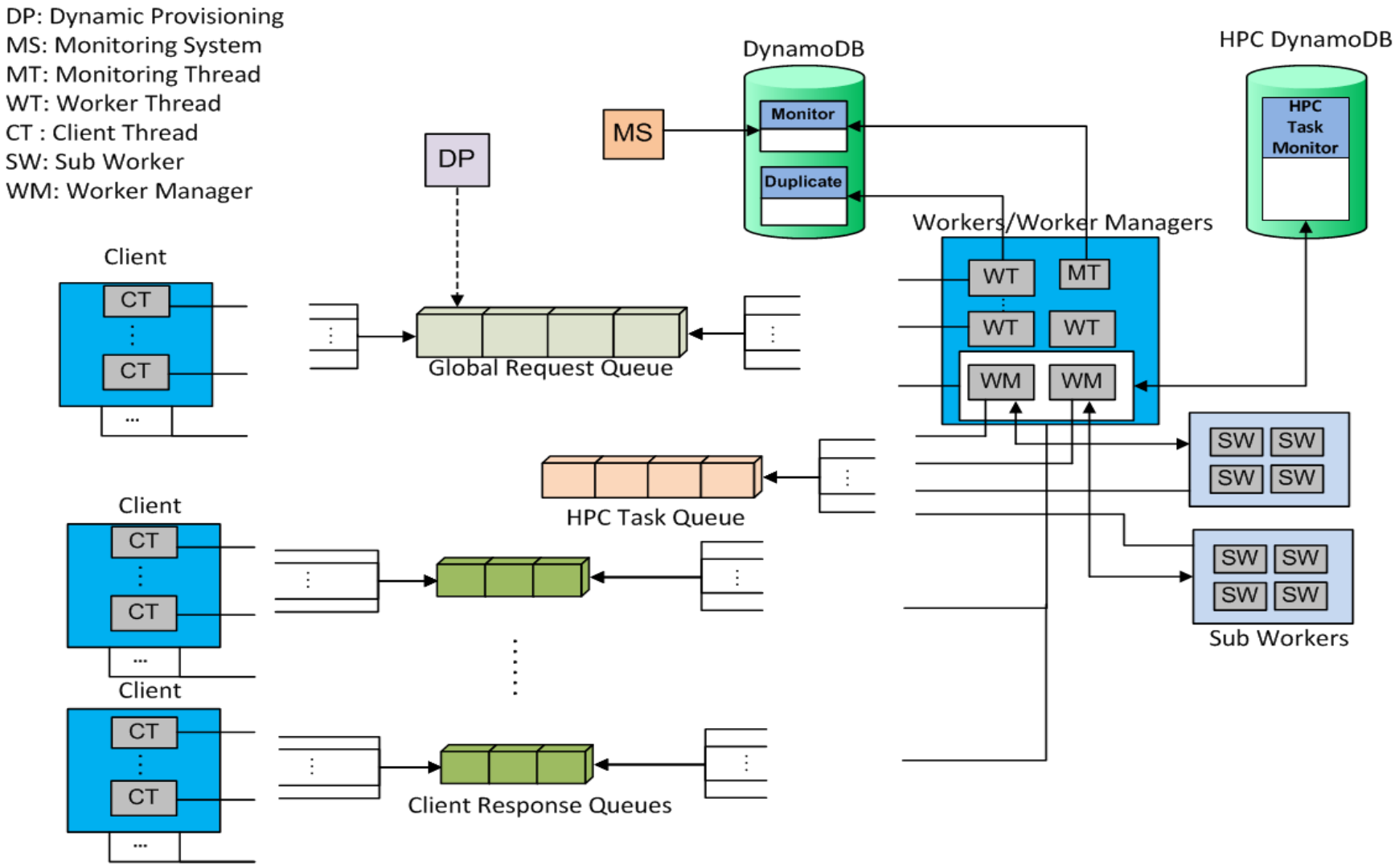
Terms Used

- Global Request Queue
- HPC Queue
- Client Nodes
- Worker Nodes
  - Managers
  - Subworkers
- DynamoDB – for HPC support
- Client Response Queue
- Job
- Task

Implementation

- Dispatching jobs
- Pick up of jobs by workers
- Workers becoming managers and subworkers
- DynamoDB for deadlock prevention of worker availability
- RMI used for workers communications
- Deletion of tasks from queues
- Sending response back to client

Architecture



Minimum Messages Required

- Minimum number of internal messages used for the HPC implementation = **m(6n + 4)**

  where ,

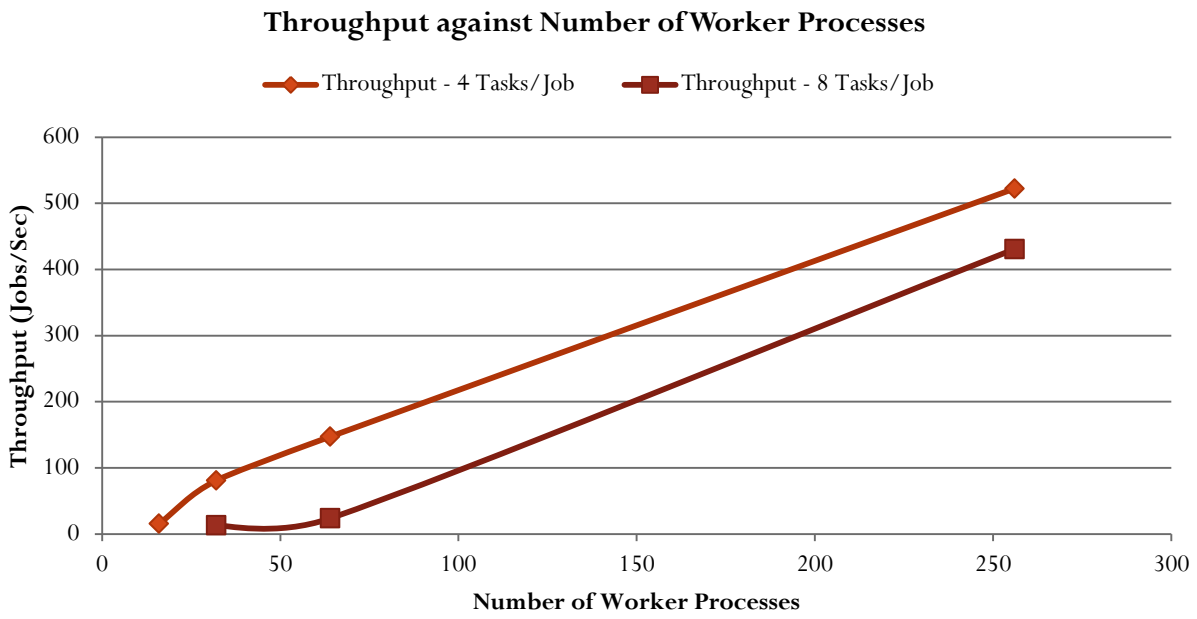
  n- number of tasks per job

  m- number of HPC jobs

Throughput Evaluation

Throughput for HPC jobs with different number of worker processes with 4 tasks and 8 tasks

Number of worker processes vs Throughput

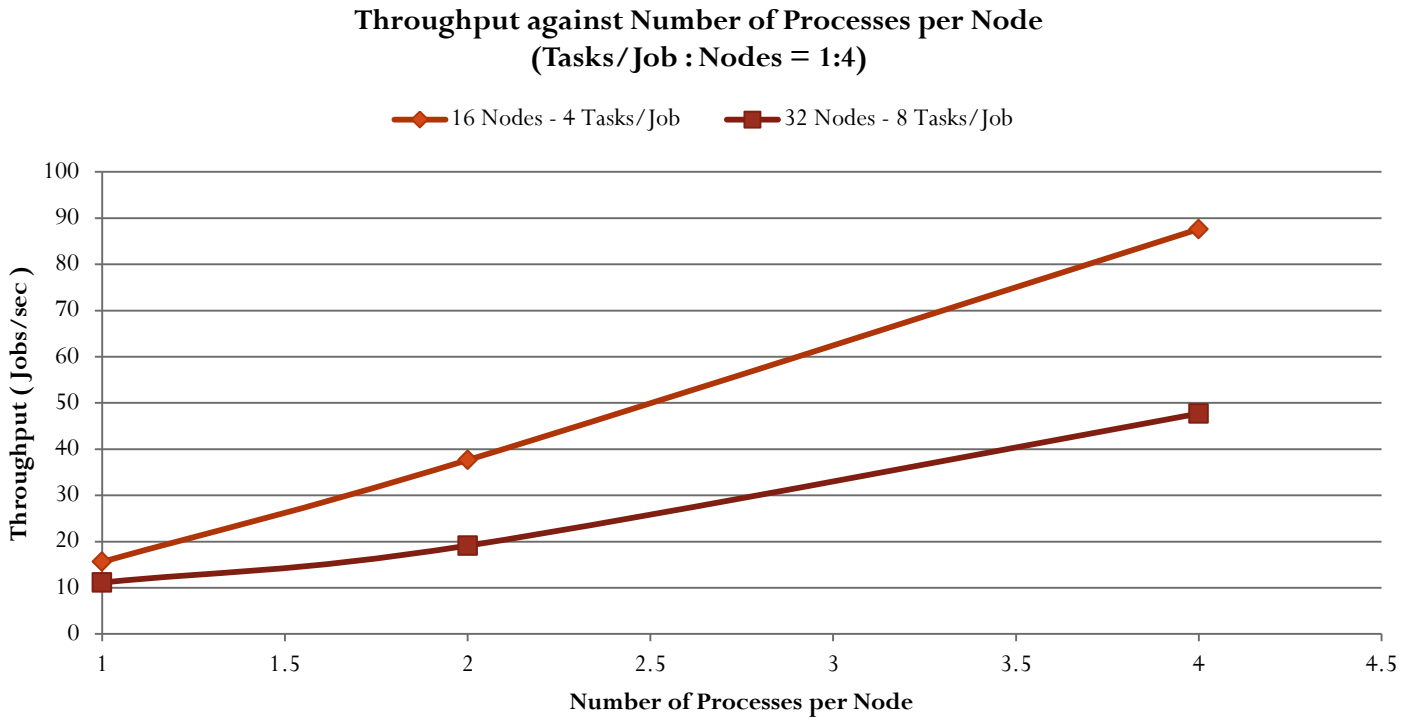


Throughput Evaluation

Throughput with multiple invocations on a single node.

- Tasks per HPC Job : Total instances = 1:4

Number of processes on a single node vs Throughput

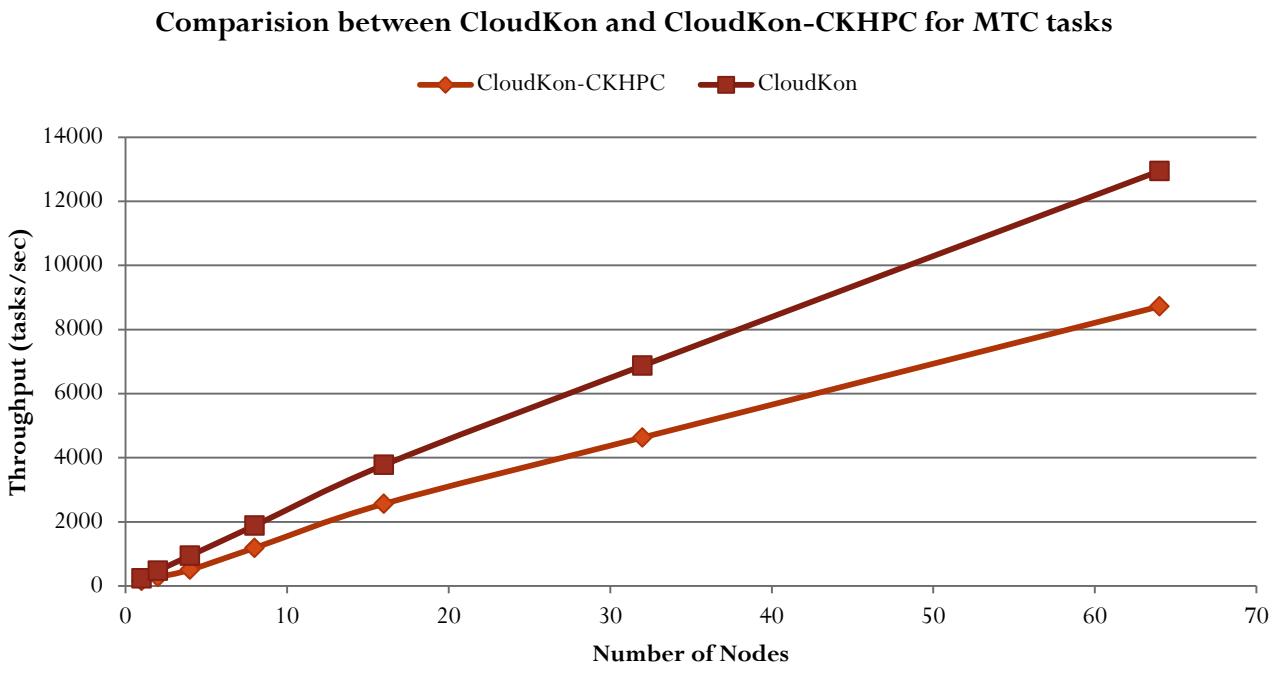


Throughput Evaluation

Comparison of CloudKon-CKHPC with CloudKon (MTC tasks)

- Throughput with HPC support/Throughput without HPC support = 2/3

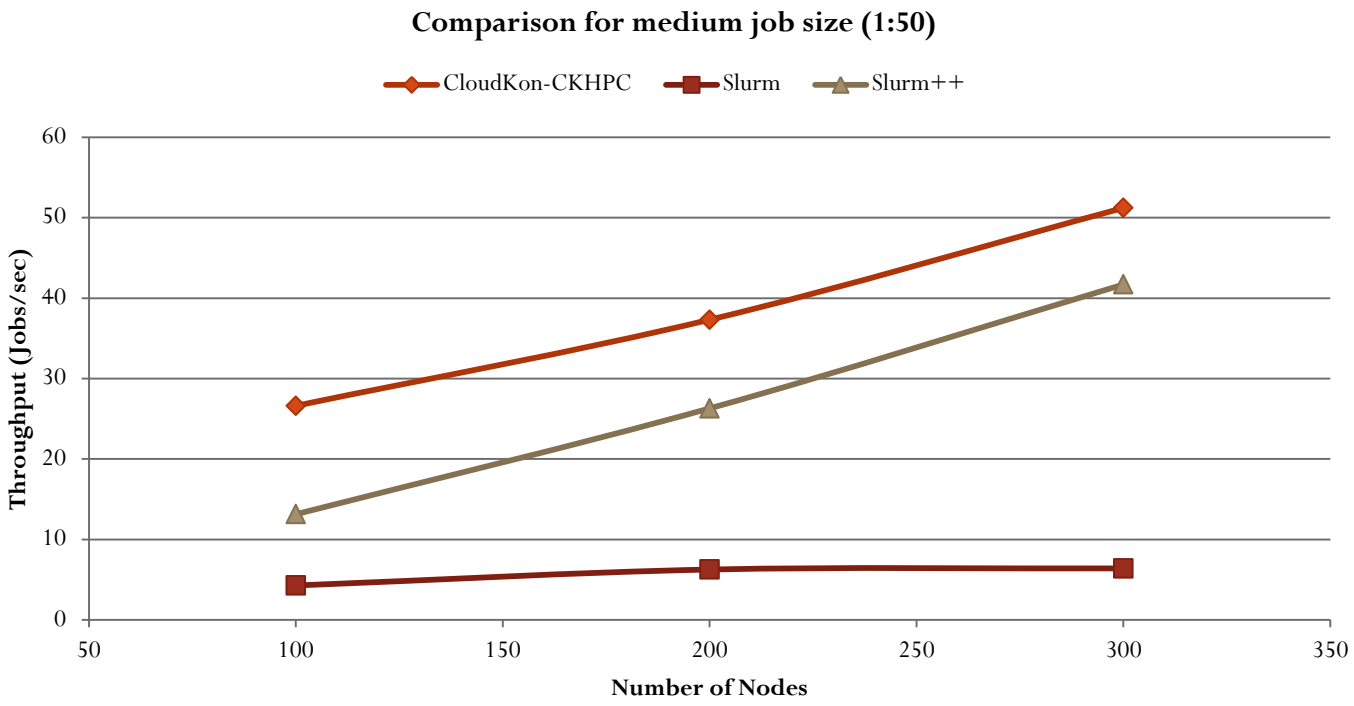
Number of nodes vs Throughput



Comparison Results

- Comparison to Slurm and Slurm++

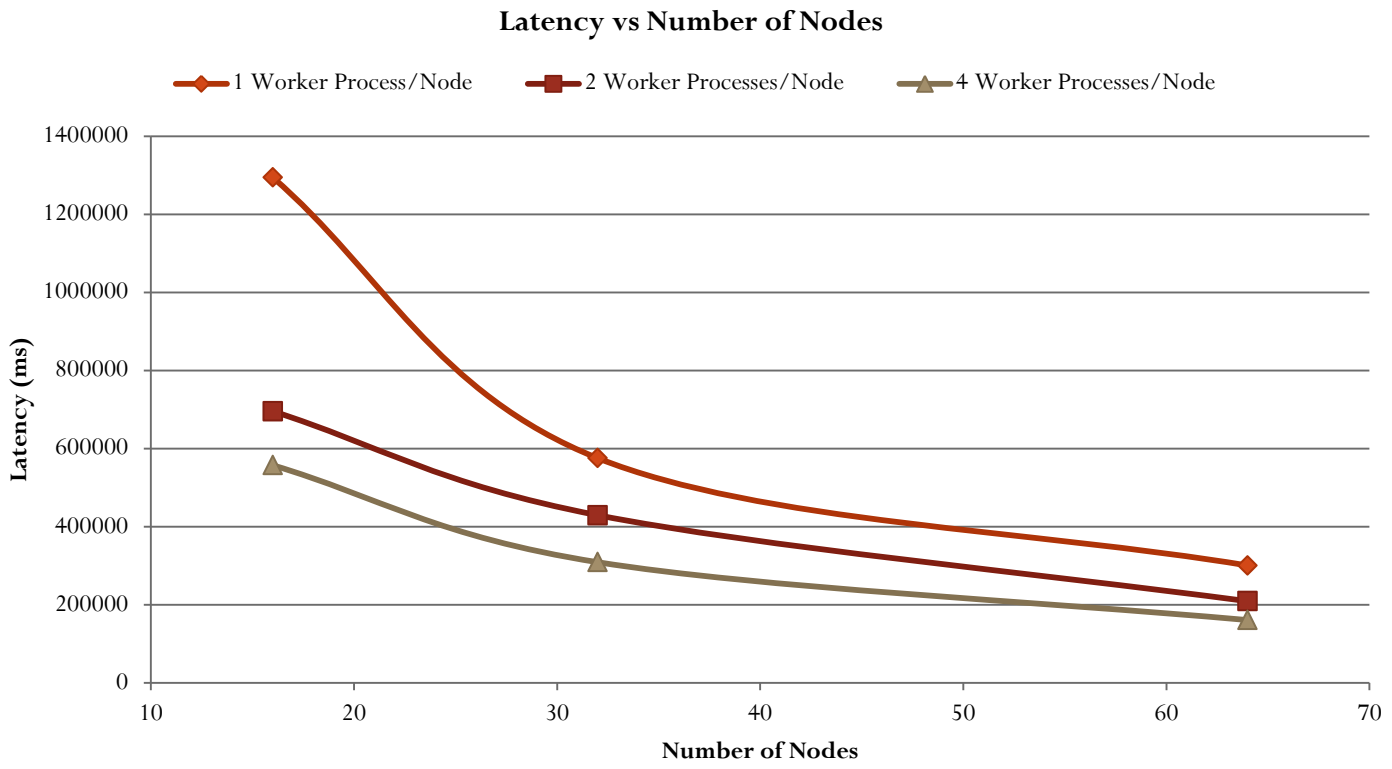
Number of nodes vs HPC jobs/sec, where HPC jobs are of varying length.



Latency Evaluation

- Latency calculated for different worker processes per node

Number of nodes vs Latency



Highlights

- Invoking 4 processes per node gives better performance than single process per node.

  - Results for 4 worker processes per node:
    - 32 nodes, 4 Tasks per job, Throughput was **2101 tasks/sec**
    - 64 nodes, 8 Tasks per job, Throughput was **3446 tasks/sec**

- 300 nodes having 40 tasks each giving a throughput of **119 tasks/sec**.
- Contributed to the CCGrid CloudKon paper under the guidance of Iman Sadooghi.

Future Work

- To reduce the centralized dependency on DynamoDB for getting free workers.
- Make it run it for real-time tasks, instead of sleep tasks.
- Extend support for MPI applications.

Conclusion

- Deadlock avoidance better than deadlock recovery - increases the resource utilization and has little effect on throughput.
- The evaluation of CloudKon CKHPC shows that it is able to provide a very high throughput outperforming other scheduling systems like Slurm.
- The throughput for CloudKon with HPC support is almost 2/3rd of that for CloudKon standalone, hence not affecting much the current support for MTC jobs.
- Since we have been able to test it upto the scale of 512 worker processes, it should be able to scale up higher as well.
- Invoking 4 processes per node gives better performance than single process per node.

References

- I. Sadooghi and I. Raicu, "CloudKon: a Cloud enabled Distributed tasK executiON framework", 2013. Available from http://www.cs.iit.edu/~iraicu/research/publications/2013_Qual-IIT_CloudKon.pdf
- High Performance Computing, Techopedia, [online] 2013, http://www.techopedia.com/definition/4595/high-performance-computing-hpc
- Amazon Elastic Compute Cloud (Amazon EC2), Amazon Web Services, [online] 2013, http://aws.amazon.com/ec2/
- Amazon DynamoDB (beta), Amazon Web Services, [online] 2013, http://aws.amazon.com/dynamodb

Thank You