Study of High Throughput Computing for CPU utilization using Condor

Raman Sehgal*, P. Shukla, R. K. Choudhury, and S. Kailas
Nuclear Physics Division, Bhabha Atomic Research Centre, Mumbai - 400085, INDIA
** email: sc.ramansehgal@gmail.com*

Introduction

An event in high energy physics is the collection of all detected particles produced due to the collision between two highly energetic particles. The Large Hadrons Collider (LHC) will store data corresponding to millions of these events per second. A job usually is a reconstruction or analysis program running on an event. Such a job can be parallelized on many computing nodes where same program is running on different sets of events. High throughput computing [1] or distributed computing thus plays a great role in high energy physics.

The key to High-Throughput Computing (HTC) is effective management and exploitation of all available computing resources in a fair manner especially if the computing environment is distributed.

These challenges are addressed by the Condor. Condor enables scientists and engineers to simultaneously and transparently exploit the capacity of idle workstations they might not even know exists. It can be used to manage workload on a dedicated computing resource (rack mount cluster), and/or to farm out work to idle non dedicated resource (desktop computers) called cycle scavenging.

Prototype condor pool

The prototype setup is designed to execute jobs remotely at idle workstations. Condor is installed and configured on a Local Area Network (LAN) of 4 Desktop computers. File System used is Network File System (NFS). These machines were connected via 100 Mbps LAN. Each of these machines are having quad core CPU @ 3.0 Ghz. Later on the setup will be replicated onto a computing cluster having 16 worker nodes and additional 20 TB of storage. Each or these nodes is having dual 6 core CPUs and gives a performance of 120 Gflops on HPL Linpack benchmark. On our prototype setup during normal working days applications like SSH, web browsing, file transfer, ROOT macro etc are mainly used which consumes very less amount of CPU. As shown in figure 1 the machine are underutilized and on an average only 2-10% of CPU is utilized. By making use of cycle scavenging capability of condor this unutilized computation power can be utilized to execute more jobs during idle times. The work of choosing the machine to submit and execute a job is done by condor itself by making use of its checkpointing [2][3], remote system calls, and ClassAds [3] mechanisms. These mechanisms are used by different universes [4] provided by condor. In this work Vanilla universe is used to run the jobs.

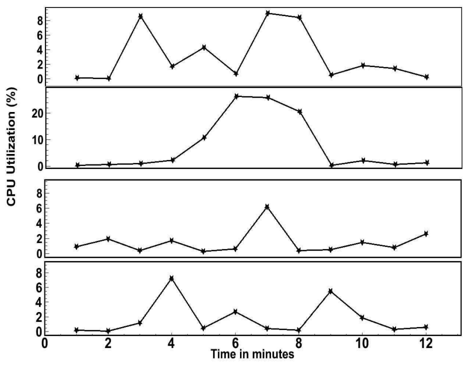

Fig. 1 Utilization profile of four machines during normal working time

ClassAd mechanism in condor

In the condor environment different machines plays different roles which can be Central manager, execute or submit. All the machines advertise their information and status periodically in classAds, Central manager periodically enters the negotiation cycle where it matches the resource request classAds of submit machine against resource offer classAds of execute machine. If a match is found, the requesting and offering machines are paired together to run the job.

Test results

The performance results reported here were collected by running pythia job using ROOT analysis framework developed by CERN. The job has to generate 4000 events. It is very computation intensive as it utilize full single core of CPU, which is observed by running the job on a quad core machine and CPU utilization results showed that only 25% of total CPU power is used.

These machines allows user to run four jobs in parallel to achieve 100% CPU utilization. In condor pool if user machine is fully utilized then also he can submit more jobs without worrying about where the job would execute. While submitting the job, condor also take care of load balancing among the machines in condor pool. Figure 2 shows CPU utilization of desktop machines in the condor pool during the execution of remote jobs on them when the machines were idle.

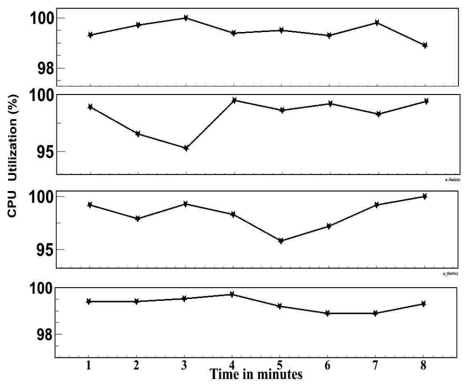

Fig. 2 Utilization profile of four machines in condor pool

Figure 3 shows the graph which depicts the reduction in execution time of job when the same job is divided in to multiple jobs each is having less number of events to be generated but total number of events are same i.e. 4000. Execution time results are shown for 1 job having 4000 events, 4 jobs each is having 1000 events, 8 jobs each is having 500 events, and 16 jobs each is having 250 events. These multiple instances of jobs were run simultaneously using Condor.

In our prototype condor pool we have 16 cores in total. It is observed that when we divide the job in 16 subjobs where each job has to generate only 250 events, then we achive 95-100% CPU utilization as shown by figure 2. Now in order to generate 4000 event it takes only 8 minutes which previously takes around 120 minutes to generate 4000 event in a single job. Hence we achieve great reduction in execution time.

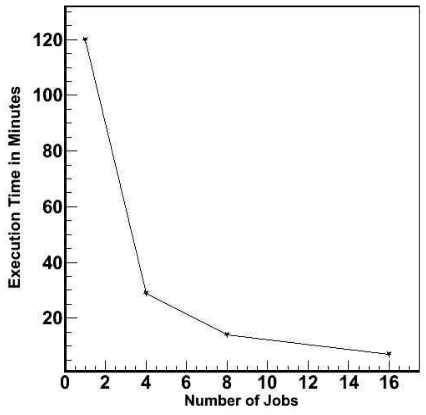

Fig. 3 Execution time v/s Number of jobs

In future this setup will be done on high end computing cluster having 16 worker node and additional storage capacity of 20 TB. This cluster will give a performance of about 1.9 Tera Flops

References

[1] http://www.cs.wisc.edu/condor/htc.html
[2] http://www.cs.wisc.edu/ condor/ manual/v7.0 / 4_2Condor_s_Checkpoint.html
[3] M. Livny, J. Basney, R. Raman and T. Tannenbaum. Mechanism for High Throughput Computing, Department of Computer Sciences, University of Wisconsin-Madison, May9,1997
[4] http:// www.cs.wisc.edu/ condor/manual/v6.4/ 2_4Road_map_Running.html