PROWL: Towards Predicting the Runtime of Batch Workloads

Dheeraj Chahal
Tata Consultancy Services
Mumbai, India
d.chahal@tcs.com

Benny Mathew
Tata Consultancy Services
Mumbai, India
benny1.m@tcs.com

ABSTRACT

Many applications in the enterprise domain require batch processing to perform business critical operations. Batch jobs perform automated, complex processing of large volumes of data without human intervention. Parallel processing allows multiple batch jobs to run concurrently to minimize the total completion time. However, this may result in one or more jobs exceeding their individual completion deadline due to resource sharing.

The objective of this work is to predict the completion time of a batch job when it is running in conjunction with other batch jobs. Batch jobs may be multi-threaded and threads can have distinct CPU requirements. Our predictions are based on a simulation model using the service demand (total CPU time required) of each thread in the job. Moreover, for multi-threaded jobs, we simulate the server with instantaneous CPU utilization of each job in the small intervals instead of aggregate value while predicting the completion time.

In this paper, a simulation based method is presented to predict the completion time of each batch job in a concurrent run of multiple jobs. A validation study with synthetic benchmark FIO shows that the job completion time prediction error is less than 15% in the worst case.

KEYWORDS

Batch jobs, Service demand, Hyper-Threading

ACM Reference Format:
Dheeraj Chahal and Benny Mathew. 2018. PROWL: Towards Predicting the Runtime of Batch Workloads. In *ICPE '18: ACM/SPEC International Conference on Performance Engineering Companion , April 9–13, 2018, Berlin, Germany.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3185768.3186407

1 INTRODUCTION

Batch workloads are usually compute intensive with repetitive transactions. One interesting challenge in batch workloads is to study the impact of one job on other concurrently running jobs in the batch. Although batch jobs are scheduled to run during off-hours, known as batch window, but these jobs can over run long enough in the presence of other jobs to impact the critical transactions during business hours. Hence it is very important to estimate the completion time of a job *a priori* in the presence of other jobs.

Generally multiprocessor compute resources used for batch jobs are managed using time sharing. The completion time of each individual batch job, when it runs concurrently with other jobs, can be derived simply from its clock time in isolation. This naive way of execution time computation generates correct results when number of cores available is more than the cores required by concurrent jobs. If in a case when the number of cores is less than the total requirement of the jobs, cores are shared between jobs or threads according to the operating system policy. This requires an advanced job execution model for completion time predictions.

Moreover, predicting the completion time of batch jobs becomes more challenging if these jobs are multi-threaded with distinct service demands of threads. Different threads constituting a job may have different service demands which directly affects the job completion time. This is due to the fact that early finishing threads do not compete for the resources and those unused computing resources are available for remaining threads. This results in faster overall execution of the remaining threads in the system.

We present a method to predict the completion time of concurrent batch jobs. Our method considers distinct service demands of threads in jobs. We consider CPU utilization of a job as a function of time or a distribution while simulating a thread on a server.

2 OUR APPROACH

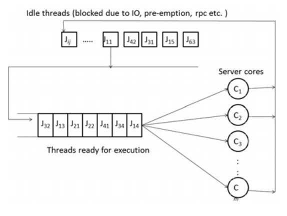
We first characterize individual batch job. We observe the service demand of each thread in the job and also measure the CPU utilization of the job at small intervals of time during the execution. A job execution model is designed based on the statistical analysis of the job characterization data. The job execution model is simulated using a discrete event simulation tool called PROWL developed by us.

2.1 Service Demand Measurement

Threads in a job may be assigned unique work and hence their service demands are non-identical within a job. This variation in service demand among threads in a job affects the completion time of all the jobs running concurrently. Hence it is very important to measure the service demand of each thread in a job for accurate simulation and prediction of job completion time of parallel running jobs in a batch. We measure the service demand of each thread in a job in isolation.

2.2 CPU Utilization Measurement

The varying CPU consumption by a job during the different stages of execution affects the completion time of other jobs. The aberrations in the completion time prediction of a batch job can be mitigated by simulating the thread execution behavior for smaller intervals of time. This requires capturing the CPU utilization value


(a)

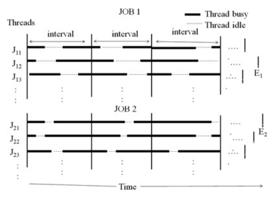

(b)

Figure 1: (a) Job execution model (b) Thread execution in small intervals

of the job at small intervals in isolation and fitting into a regression (linear or non-linear, exponential, polynomial of degree n etc.) function or distribution. Thus CPU utilization of a job can be represented with the help of a appropriate function of time or some distribution. Instead of using the constant CPU utilization for the entire run of a thread, instantaneous values of CPU utilization can be derived with the distribution or function of time for each of the smaller intervals during simulations.

2.3 Job execution model

The job execution model that we use is as shown in the Figure 1a. In our approach we model the execution time of a thread in small intervals of time based on time spent in CPU and its idle time outside CPU. Total execution time of each individual thread i is divided into small intervals n of size T such that $n * T = E_i$ (Figure 1b). Within each interval we determine the idle time t_d and execution time t_e of the thread i as below.

$$t_e = T * C_t$$
$$t_d = T * (1 - C_t)$$

Where C_t is the CPU utilization of the job defined at time t of execution. In order to consider the fluctuations in idle time and executions as in a real operating system, we do not want all threads to start execution or go to idle state at the same time. Hence for each interval we choose the idle time and execution from the uniform distribution with average t_d and t_e.

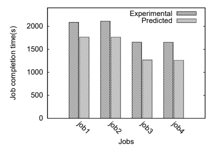
2.4 Simulation tool and algorithm

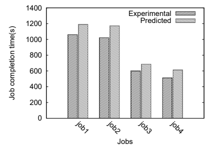
We conducted simulations using our simulation tool called PROWL. This tool is an extension of our previously developed tool DESiDE [2]. PROWL is a discrete event simulation environment and it has capabilities to perform *what-if* scenarios for capacity planning purpose.

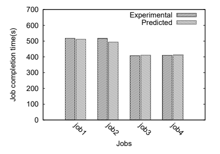
As discussed above, unique feature of PROWL is its capability to address varying CPU utilization with time. CPU utilization in PROWL can be defined as a function of time (e.g. polynomial regression) or some distribution and its instantaneous value can be determined for each interval. We use processor sharing queuing model to simulate the concurrent execution of threads in jobs.

3 EVALUATION

The experiments were performed using synthetic benchmark called FIO [1]. The tests was run with HT enabled and with HT disabled on one Intel Xeon server with 8 physical cores and also on another server with 28 physical cores. With HT enabled, this results in 16 and 58 logical cores respectively.


(a)


(b)


(c)

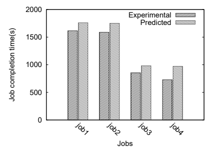

(d)

Figure 2: Experimental vs predicted job completion time of 4 concurrent FIO jobs (a) 8 core machine (No-HT) (b) 28 core machine (No-HT) (c) 16 core machine (HT) (d) 56 core machine (HT)

First we ran FIO jobs in isolation on two different servers. We observed per-thread service demand of each job and its CPU utilization with HT enabled and HT disabled. This data was fed into our simulator PROWL and we predicted the completion time of each job as if they were running concurrently. The average of five simulation runs with different seeds was taken as the predicted completion time. The predicted time was compared with an actual concurrent run on the two servers. We ran experiments with different workloads. In one experiment, the workload comprised of four FIO jobs running concurrently. We introduced some thinktime in the IO operations of each job such that each job while running in isolation results in some CPU idle time (Figure 2a and Figure 2b). This experiment was repeated with HT as well (Figure 2c and Figure 2d). We observed small difference between actual and predicted completion time of the jobs. We attribute this prediction error to the difference in service demand of some FIO jobs when running in isolation vis-á-vis running concurrently with other batch jobs.

4 CONCLUSION

We have proposed a technique to predict the execution time of individual batch job running concurrently with other batch jobs and sharing resources. We take into account per-thread demand variation in each job. We also break this per-thread demand into time slices using random variate distribution. This method ensures the accurate prediction of completion time of each job with the maximum observed error being 15%. We have also built a server simulator called PROWL. We believe that the modeling technique we have proposed can be used to predict and optimize any system involving workflows.

REFERENCES

[1] J Axboe. 2014. FIO-Flexible I/O Tester. http://freshmeat.net/projects/fio/. (2014).

[2] Benny Mathew and Dheeraj Chahal. 2017. DESiDE: Discrete Event Simulation Developers Environment. In *Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering (ICPE '17)*. ACM, New York, NY, USA, 173–174. https://doi.org/10.1145/3030207.3053672