PerfExt^{++}: Performance Extrapolation of IO Intensive Workloads

Dheeraj Chahal Mukund Kumar Manoj Nambiar

{d.chahal|mukund.k@tcs.com|m.nambiar}@tcs.com
TCS Innovation Labs
Mumbai, India

ABSTRACT

We present a tool, PerfExt^{++}, for cross platform performance extrapolation of IO intensive workloads. The tool is based on *trace and replay* mechanism and has the capability to record and replay the temporal and spatial characteristics of IO workloads. We show the design and implementation of PerfExt^{++}, which requires minimal intervention from the user to predict and extrapolate performance metrics across platforms.

Keywords

Performance prediction; IO traces; extrapolation

1. INTRODUCTION

Application migration is inevitable when it outgrows the storage resources allocated to it. The migration process involves moving the application from source system containing low-end storage devices having high utilization and latency to a newer target system with better storage devices like high-end hard disk drives (HDDs) or solid state drives (SSDs). The application migration is a non-trivial and time consuming process as it requires performance characterization of the application on potential target systems.

Trace and replay is a preferred technique as it does not require deploying the actual application or database on the target system as performance is dependent on the access pattern than actual data. Traces are portable and deterministic in nature and hence prove to be a better technique than modeling techniques in some situations [1]. However, trace collection tools like *strace* or *blktrace* slowdown the execution of application hence result in overhead at large workloads or concurrencies (no. of users). One solution is to collect application traces only at low concurrencies and then extrapolate for larger concurrencies.

Our contribution is a tool called PerfExt^{++} that can be used for performance prediction of IO intensive applications across multiple target systems containing advanced storage devices. The tool captures traces at low concurrencies on the source system and replays on target systems. The performance data of traces on target system is extrapolated for higher concurrencies. Moreover, the approach followed in the tool does not require deploying applications or databases on target systems.

The extrapolation using PerfExt^{++} is based on the assumption that there are no software bottlenecks in the application of interest and the first bottleneck arises due to the hardware resource limitations.

ICPE'17 April 22-26, 2017, L'Aquila, Italy

ACM ISBN 978-1-4503-4404-3/17/04.

DOI: http://dx.doi.org/10.1145/3030207.3053669

2. OUR APPROACH

Our approach consists of three important steps :

1) Systematically collecting the traces of an IO intensive application for varying concurrencies on the source system.

2) Replaying the collected straces on the target systems and collecting the performance data.

3) Extrapolating the performance data for larger concurrencies on the target system.

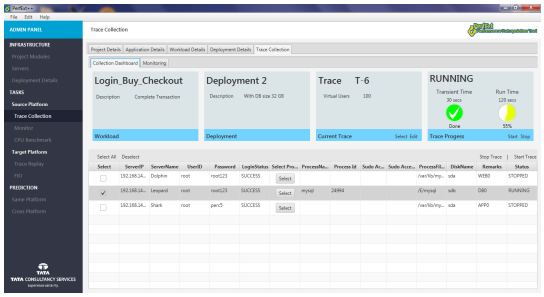
These three steps are implemented in PerfExt^{++} as below:

2.1 Trace Collection

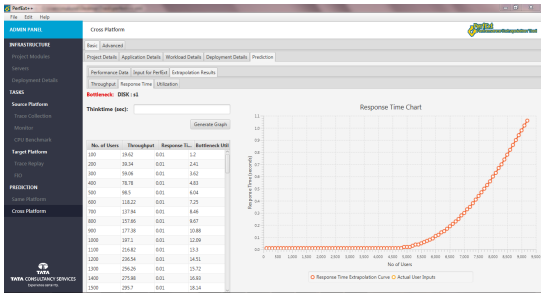
The IO trace of the application is captured using the *strace* utility available in linux systems. Trace contains all IO system calls: *read(), write(), pread(), pwrite(), open(), close(), fsync(), lseek()*. To capture the trace of an application on database server, our tool filters all the thread IDs initiated by MySQL and *strace* is attached to each of these thread IDs. A trace file is generated corresponding to each thread ID. All the trace files are merged and sorted based on timestamp value of each system call to maintain the same order of execution while replaying. IO traces are collected for multiple concurrency levels.

2.2 Trace Replay

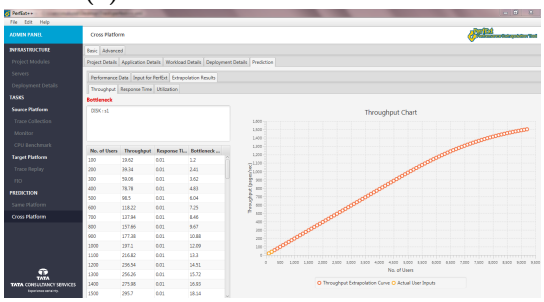
Once the trace collection is over, the tool copies all the database and trace files to a temporary directory on the target system. Any reference to a database file in the trace is replaced with the path to temporary directory on the target system. Traces collected on the source system are replayed on the target system using IOreplay [3] and performance data is recorded for each trace replay. Using the timestamp value associated with system call in the trace, we ensure that IO calls are executed at the same time interval as in the original system so that workload is replicated correctly on the target system.



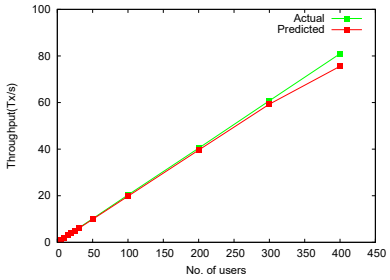
(a) Trace collection view of GUI



(b) Response time and bottleneck resource prediction



(c) Throughput and bottleneck resource prediction



(d) Throughput prediction on SSD for TPCC

Figure 1: PerfExt^{++} GUI showing trace collection view, extrapolated performance metrics and comparison of predicted and actual throughput on SSD for TPCC application.

2.3 Performance Extrapolation

For extrapolating the performance data of an application from small number of users to a large number of users, we use previously developed tool called PerfExt [2]. PerfExt takes perfromance data from trace runs on the target system as input for a small number of users in terms of throughput and resource utilization. Linear regression and S-curve technique is used to predict the throughput. PerfExt has been tested successfully with a number of sample multi-tier applications and is able to provide accuracy of about 90% in throughput and utilization metrics. PerfExt has been integrated with PerfExt^{++}, which has cross platform prediction capabilities.

3. PerfExt^{++}: SALIENT FEATURES

PerfExt^{++} GUI (Figure 1a) allows user to create a project and add or create applications for the project.The workload details which are essentially the transactions or mix of transactions can be mapped to the applications created for a project. Current deployment infrastructure details and target infrastructure details like server IPs (web, application, database etc.), user credentials, storage sub systems of the source and the target system being used by the application of interest are requested from the user. User can map different workloads to the physical servers. PerfExt^{++} can be used to collect traces of a particular workload by attaching to processes running on the source system (Figure 1a). Traces are archived in the repository on the source system. Collected IO profile traces are then replayed on target systems using trace replay feature and performance data is collected. The tool also provides an interface to monitor resource utilization in real-time while traces are running. Both cross-platform and same platform predictions are possible using embedded extrapolation tool PerfExt. GUI displays extrapolated performance metrics including response time, throughput, resource utilization (CPU, Disk) on multiple servers in a tabular as well as graphical representation (Figure 1b and 1c).

4. EVALUATION

PerfExt^{++} has been evaluated successfully for application migration studies of TPCC[1], JPetStore[2] and TCS client applications from low-end HDD to high-end HDD, high-end HDD to SSDs etc. Figures 1b and 1c shows prediction of response time for JPetStore application respectively on high-end HDD using trace from low-end HDD. Figure 1d shows comparison of predicted and actual throughput on SSD using trace from HDD.

5. REFERENCES

[1] D. Chahal, R. Virk, and M. Nambiar. Performance extrapolation of io intensive workloads: Work in progress. In *Proceedings of the 7th ACM/SPEC on International Conference on Performance Engineering*, ICPE '16, pages 105–108, New York, NY, USA, 2016. ACM.

[2] S. Duttagupta and R. Mansharamani. Extrapolation tool for load testing results. In *Performance Evaluation of Computer Telecommunication Systems (SPECTS) 2011 International Symposium on*, pages 69–76, June 2011.

[3] ioreplay. http://code.google.com/p/ioapps/wiki/ioreplay,2016.{Online;accessed-2016-08-06}.

[1]http://www.tpc.org/tpcc/

[2]https://github.com/mybatis/jpetstore-6