Performance Extrapolation of IO Intensive Workloads

[Work in Progress]

Dheeraj Chahal Rupinder Virk Manoj Nambiar

{d.chahal|rupinder.virk|m.nambiar}@tcs.com
Performance Engineering Research centre
TCS Innovation Labs
Mumbai, India

ABSTRACT

Performance prediction of an application before migrating from a source system and deploying on the target system is a challenging but important task.

In this paper, we present a method for predicting the performance of an IO intensive multithreaded enterprise application workload on target systems connected to advanced storage devices. Our approach is an extension of well-known trace and replay method. We extract traces of IO intensive enterprise workloads representing temporal and spatial characteristics (e.g. read and write requests) on the source system where application is currently deployed. These traces are replayed on the system of interest called target system. The experimental results presented demonstrate the effectiveness and accuracy of this method.

Keywords

Performance prediction; IO traces; extrapolation

1. INTRODUCTION

Many organizations are showing interest in migrating their applications from the existing low-end hard disk based systems to a high-end hard disk or a local flash memory based Solid State Device (SSD) systems for a dual purpose of saving energy and enhancing the performance of database servers. Migrating the application to a new system and testing the performance is a non-trivial and daunting task. It requires lots of efforts to set it up and subsequently fine tune. One solution is running the synthetic workloads generated by the IO subsystem and characterization tools. The synthetic workloads have access pattern very similar to that of real application. Though this approach is relatively easier to implement but may not reproduce the characteristics of the application or the workload accurately.

IO trace replay is another popular technique that can be used for reproducing the application characteristics on a target platform. Trace replay is a commonly used technique for debugging and benchmarking I/O systems.

Traces are portable such that the trace generated under one environment can be run on the other with minimal efforts without changes in the code. Trace replay can mimic the behavior of the application with very high accuracy by capturing its characteristics without revealing the sensitive information. Moreover, trace and replay is a preferred technique for migration studies since it does not require copying the actual data on the target system because data access pattern is important than the actual data itself. Traces are deterministic and prove to be better than other methods like modeling techniques in some situations.

Unfortunately, trace capturing tools like *strace* and *blktrace* slowdown the execution of the application and cause software overhead at larger workloads. Hence capturing trace for a large workload at the source system results in time dilation and replaying the same on the target systems might not provide the correct performance estimation. One solution is to capture the traces at low concurrency levels and replay on the target system and then extrapolate the results.

Using a performance prediction method before actually migrating an IO intensive application to an advanced hard disk drive (HDD) or SSD would be helpful in capacity planning. We used trace and replay technique for predicting the performance of enterprise applications on a target system with storage systems like high-end HDD and SSD. Our approach consists of the following steps:
1) Systematically generating the IO traces of the application on the database server of the source system for varying concurrencies (no. of users).
2) Replaying the traces on the database layer of the target system and collecting the performance statistics like utilization, throughput and response time.
3) Extrapolating the data collected on the target system using an extrapolation tool.

Eventually, we should be able to answer the questions like "What is the maximum number of users that we can serve if we upgrade to a new system with an advanced storage device ?" or "What would be the performance of my application with a new storage system under different workloads ?". Also these queries shall be answered without actually going through the painful process of deploying the application on target systems with different types of storage devices.

This short paper introduces a trace and replay procedure for predicting the performance of an enterprise application

ICPE'16, March 12-18, 2016, Delft, Netherlands

DOI: http://dx.doi.org/10.1145/2851553.2858665

in a new environment where it is to be migrated. Our contribution is a method that can be followed for pre-deployment performance estimation of mutlithreaded IO intensive applications on different target systems with advanced storage devices. We also present efficacy of this methods by testing on multiple target systems with different storage devices.

The rest of this paper is structured as follows: Section 2 describes the related work. In section 3 we discuss our approach to implement our methodology. Experimental set up is discussed in section 4. Section 5 provides the analysis of experimental results to evaluate the efficiency of our method. Conclusion and future scope is discussed in section 6.

2. RELATED WORK

The use of IO traces for application profiling has been there for many years now but its usage has found traction in recent years for predicting the performance in the cloud environment. Recently Tak et. al. developed a technique called PseudoApp[10]. Contrary to the approach followed in PseudoApp which builds an artificial workload , we run the actual trace on the target system. The former approach does not require replicating the database on target architecture but replaying the artificial workload is a complex process. Another similar work in this area is development of ROOT[12]. We have further extended the work of PseudoApp and ROOT to predict the performance of multithreaded applications for higher concurrency using our extrapolation tool called PerfExt [4].

There is a large body of work for IO trace replay mechanisms particularly for storage system evaluations for different purposes [2][3][7]. //Trace is another popular approach for parallel application. Yet another interesting research work for predicting the performance of web application on cloud is CloudProphet [9]. Though CloudProphet is capable of predicting the end-to-end performance of multiple resources with high accuracy but it is based on simple scaling which may not hold good for high concurrency.

3. OUR APPROACH

Our systems is a multi-tier system consisting of the load generator, application layer an the database layer. The http scripts of the application of interest are captured using the TCPProxy[1] plug-in. We perform randomization and parameterization of http scripts to simulate the random access pattern of the web application by different users. These test scripts are replayed with Grinder http plug-in [8] for arbitrary number of users and thinktime. Since the primary focus of this research is to study IO intensive applications, we captured and replayed traces only on the database layer. Detailed discussion on our approach is provided in our previous work [11].

3.1 IO trace recording

There are numerous methods and tools to trace the IO calls of an applications depending upon the layer they operate on e.g. kernel, user space or a combination of both. The layer at which these tools and methods execute also defines the performance overhead and the complexity involved. We preferred user mode contrary to the perception that kernel mode reduces error. The user mode requires no modification in the application or the kernel and profiling information can be captured easily. The I/O profile trace of application of interest is captured using the *strace* utility in the linux system. To reduce *strace* overhead and the size of trace file, we captured only IO related system calls: *read()*, *write()*, *pread()*, *pwrite()*, *lseek()*, *fsync()*, *open()*, *close()*. Each row in the captured trace consists of process ID, timestamp value, offset and the IO system call. To capture the trace, we first find all the thread IDs that are spawned by the MySQL and then *strace* is attached to each of these thread IDs. Thus multiple trace output files are generated. In order to maintain the same order of the execution on the target system, we merge all these files in to a single file and then sort system calls according to their timestamp value.

3.2 Trace replay

As a next step, the database files of the application are copied to a temporary directory on the target system. Any access to the database file in the trace is replaced by a path to the temporary directory. We used ioreplay [6] to replay the I/O trace captured on the test system. The replay tool executes the IO operations as recorded in the trace file. The ioreplay studies have shown that the it scales within a difference of few percent when compared with the original application [5]. One of the drawback with ioreplay is that it is single threaded. Hence replaying the trace for high concurrency is a challenge. We modified the ioreplay to support multlithreading.

One of the challenges associated with the trace-replay method is maintaining the realism of the workload when load-profile is replayed on a target system. We capture the IO system calls along with its timestamp. When the trace is replayed on the target system we ensure that IO calls are executed at the same time interval as in the original system so that workload is replicated correctly.

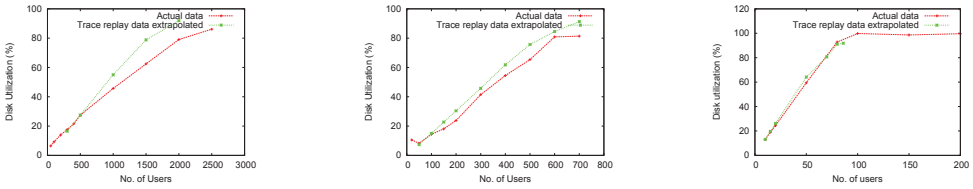
3.3 Extrapolation

To extrapolate the performance data of an application from a small number to a large number of users on the target system, we used PerfExt. PerfExt is a tool developed in our lab. The tool takes load testing results as input from for a small number of users in terms of throughput and resource utilization. To extrapolate throughput, PerfExt first estimates the maximum throughput based on the resource utilization information. Linear regression is used to predict the performance until throughput reaches the half of the maximum throughput and beyond that point sigmoid curve is fit in till the throughput reaches 90% of the maximum value. It uses a combination of linear regression and another statistical technique called sigmoid curve (or S curve) to predict the performance until the application encounters the first bottleneck. PerfExt has been tested successfully with a number of sample multi-tier applications and is able to provide accuracy of about 90% in the throughput and utilization metrics. However, it makes an assumptions that there in no software bottleneck in the application of interest.

For extrapolation using PerfExt, user performance data obtained by running traces for two concurrency levels on the target system is sufficient. The resource utilization for these multiple concurrencies is used in the PerfExt as input and extrapolated for higher concurrencies to obtain performance metrics like resource utilization, throughput and response time.

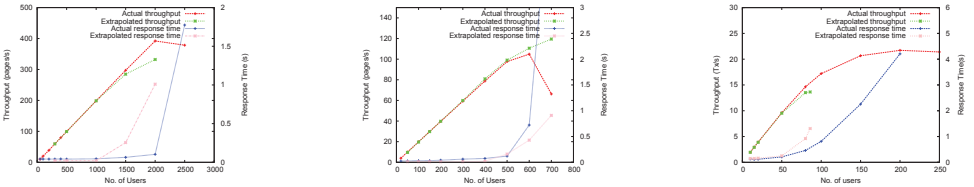
Storage device	Disk Model	RPM	No. of Disks	IO Schedular	File System	Interface	System Config	Linux Kernel
Low-end HDD	Caviar SE Serial ATA drive	7200	1	CFQ	ext4	300 Mb/s Serial ATA 2.0	8 Core Xeon CPU @ 2.6 GHz,,6MB L2 cache	CentOS 6.5,2.6.32
High-end HDD	HP-GEN7	10000	1	CGQ	ext4	Dual Port,SAS 6GB/s	16 Core Xeon CPU @ 2.4 GHz,12MB L2 cache	CentOS 6.6,2.6.32
High-end HDD (VM)	HP-GEN9	10000	1	CGQ	ext4	Dual Port,SAS 6GB/s	16 Core Xeon CPU @ 2.4 GHz,12MB L2 cache	CentOS 6.6,2.6.32
SSD	Virident Systems Inc. FlashMAX Drive Micron-slc-32	-	1PCIe	Default	ext3	-	16 Core Xeon CPU @ 2.4 GHz,,12MB L2 cache	CentOS 6.6,2.6.32

Table 1: Storage systems used in our study



(a) (b) (c)

Figure 1: Disk utilization predictions for high-end HDD for (a)JPetStore (b) equiz(c) TPC-C



(a) (b) (c)

Figure 2: Throughput and response time prediction for high-end HDD for (a) JPetStore (b) equiz (c) TPC-C

4. EXPERIMENTAL SETUP

We have validated our methodology using industrial benchmark TPC-C and two web based applications. TPC-C is an online transaction processing (OLTP) benchmark . TPC-C is considered a popular benchmark for comparing performance across different softwares and hardware configurations. Being an IO intensive benchmark, it is an appropriate test case for us.

Other two applications used in this study are equiz and JPetStore. The equiz application is implemented with java servlets, stored procedures and includes an automatic code evaluation (ACE) framework. JPetStore is an eCommerce J2EE application benchmark which allows users to browse and search for different types of pets in five top level categories.

TPC-C is executed from the command prompt while JPetStore and equiz are deployed on apache tomcat server. MySQL 5.6 is used as backend for all the applications. The think time between the application pages is fixed at 5 sec. All the performance data metrics are mesured in the steady state of the run.

The storage system configurations that we used in our studies are are given in table 1. We used low-end HDD with source system and high-end HDD or SSD with target system. The virtual machine used in the experiment had 48 Core Xeon CPU (2.5 GHz),30MB L2 cache and CentOS 6.4.

5. RESULTS

We predicted the performance of these applications on a high end HDD and SSD storage systems using the trace generated on low-end HDD.

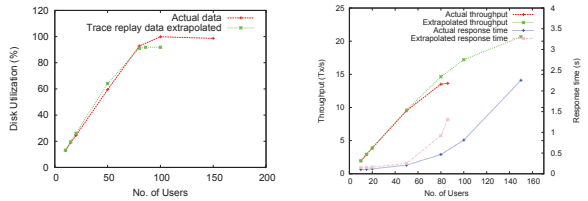
5.1 Low-end HDD to High-end HDD migration

Trace files were generated on the source system by running JPetStore, equiz application and TPC-C benchmark on the test system for varying workloads. JPetStore application was run on the source system with low-end HDD for 50, 100, 200, 300 users, equiz for 50,100 and 150 users and TPC-C for 10, 15, 20 users. All these trace files were replayed on the target system and performance metrics were observed. Performance data was extrapolated for higher concurrencies using PerfExt as shown in the Figure 1 and Figure 2. Extrapolation tool is modeled to predict until any of the resources (CPU, disk or memory) is 90% utilized while the application is run till average disk utilization is 98% and hence actual data trend lines are extended for larger concurrencies as compared to the extrapolated. Disk utilization (Figure 1), throughput and response time (Figure 2) are predicted accurately until 90% of the resource utilization (CPU or disk) on the database server when compared with the actual performance data. Some inaccuracy is observed in the response time prediction particularly for TPC-C at higher concurrencies. We observed that for TPC-C disk attains 90% utilization around 120 user workload but throughput and response time increases until utilization is 98% for 200 users beyond which throughput starts falling gradually.

TPC-C performance prediction was also done for VM with high-end HDD (figure 3) and results similar to physical machine are obtained upto 90% of disk utilization.

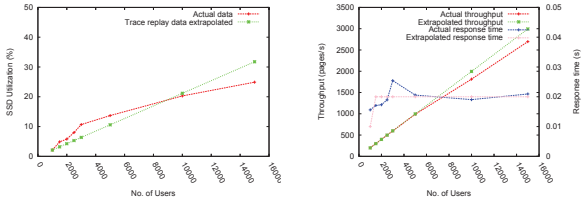
5.2 Low-end HDD to SSD migration

IO traces of JPetStore and equiz from the source system were also tested on target systems with SSD. As shown in the



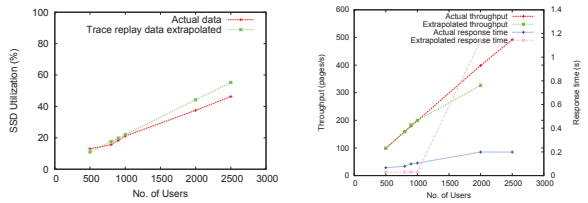
(a) (b)

Figure 3: TPC-C throughput and response time prediction for VM with high-end HDD (a) Disk utilization (b) Throughput and response time



(a) (b)

Figure 4: JPetStore application predictions for SSD (a) Device utilization (b)throughput and response time



(a) (b)

Figure 5: Equiz application prediction for SSD (a) Device utilization (b) Throughput and response time

figure 4 and figure 5, maximum device utilization for JPetStore and equiz is 25% for 15000 and 45% for 2500 users respectively. Beyond this workload, CPU at application server becomes a bottleneck. SSD utilization, throughput and response time predictions are close to actual performance data for JPetStore (Figure 4). Utilization and throughput are predicted correctly for equiz as well (Figure 5). The incorrect response time prediction for equiz beyond 1200 users is due to our extrapolation tool considering only disk service demand for predictions while CPU service demand is dominant in this case.

6. CONCLUSION

The primary objective of our research is cross platform application performance prediction when it is migrated from one storage system to another. We predicted the performance with a high accuracy when application database is migrated from a low-end HDD to a high-end HDD (VM and physical systems) and SSD. The experimental results show that the prediction accuracy is within 10% error bound until 90% of the storage device utilization is reached.

Our future work aims to evaluate the performance predictions for CPU intensive applications using IO, network and memory traces. We also plan to conduct extensive tests for performance estimation of virtual machines running within cloud data centers.

7. REFERENCES

[1] http://grinder.sourceforge.net/g3/tcpproxy.html.

[2] E. Anderson, M. Kallahalla, M. Uysal, and R. Swaminathan. Buttress: A toolkit for flexible and high fidelity i/o benchmarking. In *Proceedings of the 3rd USENIX Conference on File and Storage Technologies*, pages 4–4. USENIX Association, 2004.

[3] A. Aranya, C. P. Wright, and E. Zadok. Tracefs: A file system to trace them all. In *FAST*, pages 129–145, 2004.

[4] 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.

[5] J. Horkỳ and R. Santinelli. From detailed analysis of io pattern of the hep applications to benchmark of new storage solutions. In *Journal of Physics: Conference Series*, volume 331, page 052008. IOP Publishing, 2011.

[6] H. J. Ioapps toolkit - ioprofiler and ioreplay tools, 2010.

[7] N. Joukov, T. Wong, and E. Zadok. Accurate and efficient replaying of file system traces. In *FAST*, volume 5, pages 25–25, 2005.

[8] J. KrizÌŇanicÌĄ, A. GrguricÌĄ, M. MosÌŇmondor, and P. Lazarevski. Load testing and performance monitoring tools in use with ajax based web applications. In *MIPRO, 2010 Proceedings of the 33rd International Convention*, pages 428–434, May 2010.

[9] A. Li, X. Zong, M. Zhang, S. Kandula, and X. Yang. Cloudprophet: predicting web application performance in the cloud. *ACM SIGCOMM Poster*, 2011.

[10] B. C. Tak, C. Tang, H. Huang, and L. Wang. Pseudoapp: Performance prediction for application migration to cloud. In *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*, pages 303–310, May 2013.

[11] R. Virk and D. Chahal. Trace replay based i/o performance studies for enterprise workload migration. In *2nd Annual Conference of CMG India*, page Online, Nov. 2015.

[12] Z. Weiss, T. Harter, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau. Root: Replaying multithreaded traces with resource-oriented ordering. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, SOSP '13, pages 373–387, New York, NY, USA, 2013. ACM.