2013 IEEE 27th International Symposium on Parallel & Distributed Processing Workshops and PhD Forum

BPS: A Performance Metric of I/O System

Shuibing He, Xian-He Sun, Yanlong Yin
Department of Computer Science
Illinois Institute of Technology
Chicago, IL, USA 60616
{she11, sun, yyin2}@iit.edu

Abstract—It is known that I/O system rather than CPU and memory is the performance killer of many of the newly emerged data intensive applications. Evaluating and understanding I/O system performance has become a timely issue facing the high performance computing community. Conventional I/O performance metrics, such as Input/Output Operations Per Second (IOPS), bandwidth, response time, etc., are effective for traditional I/O environments. However, as I/O systems become more and more complex, existing I/O metrics become less and less able to catch the characteristic of I/O systems performance. In this study, we reveal the limitations of existing metrics, and introduce a novel I/O metric, Blocks Per Second (BPS), to measure the performance of the I/O systems. A unique merit of BPS is that it provides an overall I/O system performance, not the file system performance or disk performance. This is very important; since with concurrency and optimization at the I/O stacks, file system performance and disk performance no long represent the data access performance. In fact, they are often misleading. A methodology is designed to measure BPS, and experiments are conducted with various I/O access patterns and storage configurations. Experimental results show that BPS is significantly more appropriate than existing metrics in I/O performance evaluation.

Keywords-I/O performance evaluation, I/O metrics, parallel I/O system

I. INTRODUCTION

During the last two decades, the advance of VLSI technology has made a dramatically improvement in CPU and memory performance. However, I/O performance has not gained as much improvement as have CPU and memory. For example, in modern computers [1], the CPU cycles are in the range of several nanoseconds or even less and memory access latency ranges from tens to hundreds of nanoseconds; I/O latency of an ordinary disk is still up to several milliseconds, which is three orders of magnitude larger than CPU cycles. As a result, for many data intensive applications in the high performance computing (HPC) community, I/O rather than CPU and memory is the performance bottleneck. Furthermore, the performance gap between the I/O system and the rest of the computer system is widening rapidly, hence I/O systems are becoming the dominant performance factors for data intensive applications.

Because I/O systems are the performance bottleneck, evaluating and measuring I/O systems has become an important issue facing HPC community. Existing I/O metrics, such as Input/Output Operations Per Second (IOPS), bandwidth, response time, etc., are designed for traditional storage systems or devices. However, current I/O systems are rapidly evolving and become more and more complex. On the one hand, they include multiple layers in the I/O stacks, such as I/O middleware, file system, and the underlying storage system. On the other hand, I/O systems often adopt multiple optimization techniques to improve efficiency, such as application optimizations, I/O middleware optimizations, and storage layer optimizations. This complexity often is in terms of concurrency or utilization of concurrency. While conventional I/O metrics are effective for traditional I/O environments, they are not designed to handle the more and more sophisticated I/O systems, and become less and less able to catch the characteristic of current I/O system's overall performance. In general, single component's improvement does not necessarily lead to an improvement in terms of overall computer performance or overall I/O performance. In fact, they are often misleading. For example, a higher bandwidth value for file systems or storage systems does not necessarily means a better overall I/O performance. Like shown in Section IV.C.2, when IOPS increases, the application execution time may increase. In another example in Section IV.C.3, with parallel I/O optimization applied, average response time also shows misleading information on application execution time.

There are two major reasons that existing I/O metrics cannot directly characterize the overall I/O systems performance. First, most I/O operations in modern I/O systems are performed in parallel. Multiprocessing and multithreading techniques [2, 3] increase the parallelism of CPU execution and the I/O accesses. Parallel file systems, such as Lustre [4], GPFS [5], PVFS [6], and PanFS [7], allow multiple I/O servers to service I/O requests concurrently. These techniques make the relationship between I/O access and computer performance more complicated. A single I/O access performance is no longer able to represent the overall I/O system performance. In these cases, evaluating I/O systems from a single access or on a single component does not reflect the complexity of modern I/O systems. A more appropriate I/O metric should consider all the I/O concurrency to measure the overall performance of an I/O system.

Second, because of the utilization of advanced I/O optimizations, the amount of data required by applications

978-0-7695-4979-8/13 $26.00 © 2013 IEEE
DOI 10.1109/IPDPSW.2013.64

may be largely different with the amount of data actually moved through the I/O system. For instance, data sieving [8, 9], a widely used optimization for small, noncontiguous I/O accesses [10-12], will access some extra data regions (holes) required by the applications. Data prefetching [13, 14] may also prefetch data more than required. However, an additional data movement may not always be useful for the data access performance as seen by applications. Thus, the influence of the improvement of one particular component becomes increasingly tangled and elusive. In general, I/O metrics which only reveal the I/O performance of particular component are not suitable to reflect the overall I/O performance in modern computing systems.

In summary, we need a new I/O metric to measure the performance of modern I/O systems. To reach this goal, the new I/O metric—Blocks Per Second (BPS) metric is proposed in this study. A unique merit of BPS is that it provides an overall I/O system performance, not the file system performance or storage device performance. Component-width performance evaluation certainly has its usefulness, but with the preeminence increase of I/O systems, measurement and understanding of the overall I/O system performance is in high demand and is timely important in assisting I/O performance optimization.

The introduction of BPS is threefold. First, this study introduces the definition of BPS; then it presents the measurement methods of determining the number of blocks and the time of I/O accesses; finally, numerous experiments are conducted to confirm that BPS is significantly more appropriate than the existing I/O performance metrics. The statistical variable correlation coefficient is used to evaluate the effectiveness and accuracy of the metrics. BPS has a 0.91 correlation coefficient value with the overall system performance, whereas conventional metrics only have good or acceptable correlation in certain cases.

The rest of this paper is organized as follows. We discuss the advantages and disadvantages of the existing I/O metrics in Section II. Section III gives the definition of BPS and presents the measurement methodology. Section IV evaluates BPS by a series of experiments. Finally, we conclude the paper in Section V.

II. EXISTING I/O METRICS

As the same as the performance measurement of CPU and memory system, I/O system's evaluation also involves several different metrics. We present an overview of these metrics in this section.

One of the most commonly used I/O performance metrics is throughput. Throughput describes the rate at which the I/O system transfers data, and is usually measured in two ways: I/O rate and data rate. I/O rate is denoted by Input/Output Operations Per Second (IOPS) and data rate is denoted by bandwidth [15]. For a given period, IOPS indicates the number of I/O accesses per second, and bandwidth indicates the amount of data accessed per second. These two metrics are used to evaluate the performance of traditional storage systems or storage devices. IOPS is usually used for approximate performance evaluation in cases where the size of each request is small, such as online transaction processing [16]. Bandwidth is generally used for applications where the size of each request is large [17, 18].

While IOPS works well to evaluate I/O performance for fixed-size I/O requests, it is not practical asking all general HPC applications to issue I/O requests with the same size in real systems. If IOPS is used to evaluate performance in I/O systems where a lot of varied-size I/O requests exist, it may fail badly in correctness. For example, Figure 1(a) describes two I/O access cases where two requests R1 and R2 are served by an I/O system. In the left case, R1 and R2 are served with a small size of S and an I/O time of T1 and T2. In the right case, the two requests are served together with a large size of 2S, and a less I/O time of T1. Assuming T1=T2=T, according to the definition of IOPS, the left case has a value of (2)/(2T)=1/T, just as the same as that of the right one. However, in terms of overall computer performance or overall I/O performance as seen by the application, the right case performs better than the left one, because of the shorter execution time or I/O time. The mismatch between IOPS and the overall I/O performance shows that it is inefficient to evaluate performance in general HPC applications. In contrast, our BPS can measure the I/O performance accurately for variably-sized I/O requests.

Bandwidth is a metric similar to BPS. It uses the amount of data moved into file systems or storage systems to evaluate the I/O performance. The main difference is that bandwidth measures the performance of the underlying file systems but BPS measures the performance of the I/O systems. The argument of BPS is that many optimizations/managements are conducted in I/O stacks, before sending the requests through the network. I/O systems performance is no long equal to the file system performance in modern computer systems. For example, in the two I/O access cases in Figure 1(b), though bandwidth in the right part is higher than that of the left one, their overall I/O access time is the same from an application point of view, which means the overall performance remains the same for their data accesses. It shows that bandwidth is not a good metric to evaluate I/O systems of HPC. By measuring the I/O performance using data required by application, BPS is proved a good performance metrics.

Another performance metric for I/O systems is response time (or latency). Response time measures how long it takes a system to finish an I/O operation. As the response time of each I/O request in the I/O systems may be different with one another, average response time (ARPT), which is the arithmetic mean of all the I/O request response times, is often used to measure the I/O performance. Figure 1(c) demonstrates two I/O access cases with sequential and concurrent I/O requests respectively. These two cases have the same APRT value T, but obviously the concurrent I/O access case has higher overall I/O performance. As ARRT does not consider the I/O access concurrency, it is also not suitable to measure

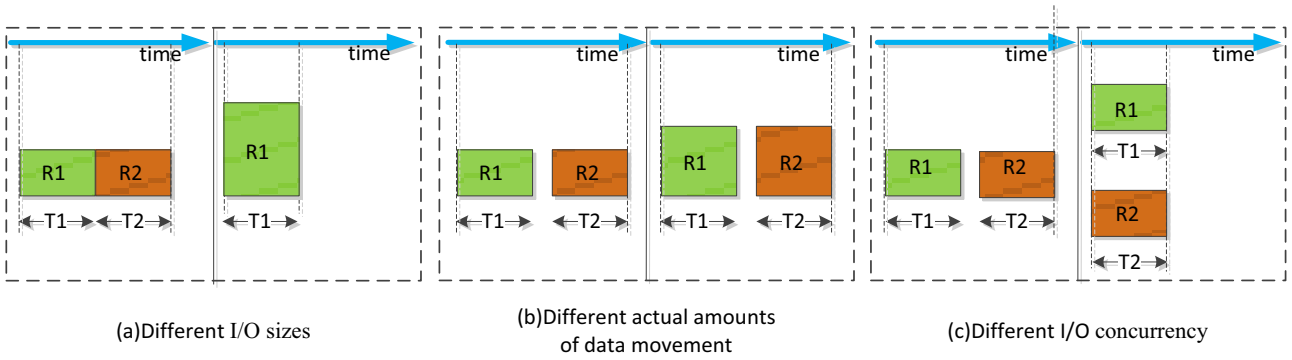

Figure 1. Two I/O requests R1 and R2 are served by an I/O system in 6 different cases. In each case, each request is represented by a rectangle, of which the length means the I/O completion time and the height means the I/O request size. Each subfigure presents two I/O access cases for comparison.

the performance of the overall I/O systems. Contrastively, BPS evaluates the performance of I/O systems using the overlapped I/O time, and thus is effective for both sequential and concurrent requests.

Besides the above three widely used I/O metrics, there are other performance metrics for some special storage device. With rotating drives, the seek time measures the time it takes the disk head assembly on the actuator arm to travel to the track of the disk where the data will be read or written. In addition, rotational latency is the delay waiting for the rotation of the disk to bring the required disk sector under the read-write head. It depends on the rotational speed of a disk (or spindle motor), measured in revolutions per minute (RPM). For most magnetic media-based drives, the average rotational latency is based on the empirical relation that the average latency is half of the rotational period.

In summary, existing I/O performance metrics, such as IOPS, bandwidth, and response time, are inadequate to evaluate the overall performance of modern I/O systems. In other words, with concurrency and optimization at the I/O stacks, a better value in one of existing I/O metrics may not mean a better performance of the overall I/O system. Recently, Sun and Wang have proposed a new memory performance metric—Access Per Cycle (APC) to measure memory system [19]. APC separates memory system performance from overall computer performance but in the meantime correlates with overall computer performance. Considering the different characteristics between memory system and I/O system, and inspired by the thought of APC, we propose the new metric of BPS to measure the overall I/O system performance.

III. BPS Definition and Measurement

In this section, we describe the formal definition of BPS, and give the method of calculating the number of blocks and the time of I/O accesses used in the BPS equation in a real I/O system.

A. BPS Definition

BPS is proposed to evaluate the performance of overall I/O system and is measured as the number of I/O blocks per second. We use the term "block" because I/O systems usually read/write data from/to a block device. Definitively, BPS is the number of I/O blocks (e.g., 512bytes) required from application divided by the time of I/O accesses. We use the amount of data required by applications to reveal the overall I/O system performance rather than a component performance. Letting B denote the number of I/O blocks (Read/Write), and T denote the total time consumed by these accesses, BPS is defined as equation (1).

$$BPS = \frac{B}{T} \tag{1}$$

The definition is simple. However, because modern CPU and I/O systems use a large number of advanced techniques to improve application performance, several I/O accesses may co-exist in the system at the same time. Measuring time T is not as simple as it may seem.

- T should only include the time when I/O operation is performing, which means the inactive time is not included in T when there are no I/O accesses in the system during an observed time period.
- In addition, T should be measured based on the overlapping mode. That is to say, for several concurrent I/O accesses overlapped together, T is the wall time of the overlapping I/O time.

Figure 2 gives an example of how to measure T for four concurrent I/O requests (R1, R2, R3, and R4) in an I/O system. Among them, R1, R2, and R3 overlap with each other partially. Instead of using (T1+T2+T3), we use $\Delta t1$ as T—the total time of I/O access in equation (1) for the first three requests. For R4, T is equal to $\Delta t2$ (or T4). The idle I/O period between t6 and t7 is not included. In

this scenario, the overall T for these four requests is equal to Δt1+ Δt2.

For the other parameter B in equation (1), all the I/O blocks issued from the application are counted, including all successful accesses, non-successful ones, and all concurrent ones. For example, in terms of the requests in Figure 2, B will be the sum of all the I/O blocks of R1 to R4, which is equal to the total amount of data of these requests divided by the block unit size.

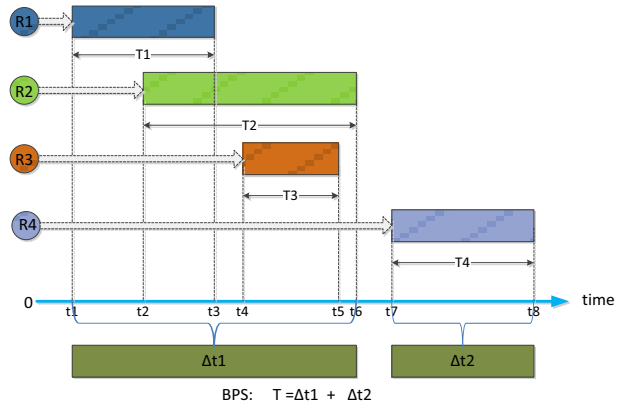

Figure 2. Measurement of time T in BPS equation

B. BPS Measurement Methodology

Monitoring computer performance is a long-standing topic of concern. In order to measure the CPU performance, modern computer cores, such as Intel Core [20] and IBM POWER3 [21], have provided a number of hardware performance counters to describe the detailed information of the internal hardware components. Some high-level programming interface for accessing hardware performance counters from application, such as PAPI [22], is available today. However, in terms of I/O performance, there are no hardware counter supports in existing computer cores. As a result, we measure BPS with a software approach.

Calculating I/O access time in a real environment is indirect for two reasons. First, I/O access does not occur in every clock cycle. Secondly, many different I/O accesses can overlap with each other. Ideally, the I/O access time should be counted only once in the total I/O access time even if there are several different I/O accesses occurring at the same time as shown in Figure 2.

In this study we measure BPS via the following steps.

Step 1: Recording I/O access information of each process

We use one record to capture the information of each I/O access of a process. Each record includes process ID, I/O size (blocks), I/O start time, and I/O end time. Multiple I/O accesses of a process lead to multiple records. We get this information in the I/O middleware layer for MPI-IO applications, or I/O function libraries for ordinary POSIX interface applications, to avoid the modification of applications. After the execution of the application, the detailed information related to each I/O access is obtained. If the application has multiple processes, the information of each process is recorded. If the I/O system services more than one application concurrently, we record the I/O access information of all the applications.

Step 2: Gathering the information of all processes into a global collection

We collect the I/O access information of all processes to have a comprehensive knowledge of the performance of the overall I/O system. First, we accumulate the number of I/O blocks of each process into B—to get the total number of I/O blocks in BPS definition. Second, we gather the I/O time information of all processes into one time collection (col_time) to prepare for the total I/O time in BPS definition. The time information collection contains the value pairs of beginning and ending time of all I/O accesses.

Step 3: Calculating the overlapped I/O access time

Input: I/O access time collection (col_time)
Output: Overlapped I/O access time (T)

```
/* sort all records in col_time according to the start time of each record */
sort()

tempRecord =first Record of col_time
while col_time has next do
        if tempRecord.endtime < nextRecord.starttime then
          T =tempRecord.endtime-tempRecord.starttime
        endif
        else
          nextRecord.starttime =tempRecord.starttime
          if nextRecord.endtime <tempRecord.endtime
                nextRecord.endtime = tempRecord.endtime
          endif
        endelse
 tempRecord=nextRecord
endwhile
T =tempRecord.endtime - tempRecord.starttime
```

Figure 3. BPS time calculating algorithm

Figure 3 gives the algorithm to calculate the I/O access time. It takes the record sequence col_time as input, and then gives an output of the total I/O access time. The algorithm is straightforward. It includes two parts. One is a sort function, which orders the records according to the start time of each record. This part is very similar to an ordinary fast sort function, and it is used to speed up the process of I/O time calculating. The other is actual time calculating, which produces the final total I/O time in BPS definition through a step-by-step record comparison.

After these three steps, we can get the BPS value with equation (1).

C. Overhead Analysis

Because our BPS measurement is based on a software approach, it does not need extra hardware to evaluate the I/O performance. However, there are two overheads need to be considered in our measurement.

One is the space overhead in recording the I/O access information. In our method, the number of record is proportional to the number of I/O requests. All these records can be located on available media, such as memory or disk space, according to a configuration file defined by users. As the size of each record is 32 bytes, even for 65535 I/O operations, all the records need about 3 megabytes, which is ignorable in modern computer system.

The other is the time cost in calculating the overlapped I/O time. The complexity of the algorithm is $O(n\log_2 n)$, where n is the number of records. Since this calculation can be overlapped with data accesses, the computing overhead of this algorithm is very affordable for modern CPUs. Please notice, while I/O performance has received more and more attention in recent years, hardware counter for I/O performance is expected to be available in the near future. Therefore, the correctness and effectiveness of BPS is the focus of this study, not its overhead.

IV. Experimental Evaluation

This section first introduces the mathematical concept of correlation coefficient. With correlation coefficient as the proximity measurement, a series of experiments are conducted with different I/O access cases to confirm our BPS is a better metric than the existing ones.

A. Correlation Coefficient

The motivation of I/O evaluation is due to the fact that the final overall computer performance is heavily influenced by I/O system performance for data-intensive applications. Therefore, an appropriate I/O metric should correlate the system performance. The mathematical statistic variable correlation coefficient (CC) is used in this study to determine which I/O metric most closely trends with the overall computer performance variation. Correlation coefficient describes the proximity between two variables' changing trends from a statistics viewpoint. It measures how well two variables match with each other. Please notice, correlation does not mean proportional. For instance, car's sale correlates with the quality of the car. But, the quality and sale are two totally different concepts and there is not direct measurement comparison.

Given two variables X and Y, CC can be calculated as equation (2). CC has a value range from -1 to 1. Two points about the correlation between X and Y can be derived from the CC value. (1) Correlation direction. A positive CC value means positive correlation, which means if X increases, Y also increases. Otherwise, it is a negative correlation direction between the two. (2) Correlation strength. The higher the absolute CC value is, the stronger relation the two variables would have. If it is 1 or -1, the two variables' trends are perfectly matched to each other; if it is zero, there is no correlation between the two variables.

$$\mathrm{CC} = \frac{\sum\left(X-\overline{X}\right)\left(Y-\overline{Y}\right)}{\sqrt{\sum(X-\bar{X})^2}\cdot\sqrt{\sum(Y-\bar{Y})^2}} \quad (2)$$

We use CC to reflect the relationship between the overall computer performance and I/O system performance. For a given application, we use the application execution time to denote the overall computer performance, and different I/O metrics to denote I/O system performance. Table 1 lists the expected correlation directions (shown by CC value) when different metrics are used. We take the I/O metric as the most appropriate one if its variety trend has the closest match with that of the overall performance variation. Obviously, this closest match means a stable correlation direction and strongest correlation strength in different I/O access cases.

Table 1. Expected correlation directions of each I/O metric

I/O metrics	CC value
IOPS	negative
Bandwidth	negative
Average response time	positive
BPS	negative

B. Experiment Settings

The experiments were conducted on a 65-node SUN Fire Linux cluster, in which there are 64 computing nodes and one head node. Each computing node has two Quad-Core AMD Opteron(tm) processors, 8GB memory and a 250GB 7200RPM SATA-II disk (HDD). All nodes are equipped with Gigabit Ethernet interconnection, and 17 nodes are equipped with an additional PCI-E X4 100GB SSD. The operating system is Ubuntu 9.04, Linux kernel 2.6.28.10. The parallel file system is PVFS2 version 2.8.1.

We conducted 4 sets of experiments to confirm that BPS is better than traditional I/O metrics. Table 2 lists the detailed I/O access cases in our experiments.

Table 2. I/O acess cases

Experiments	Descriptions
Set1	various storage device
Set2	various I/O request size
Set3	various I/O concurrency
Set4	various additional data movement

We conducted the evaluation with three popular I/O benchmark tools, IOzone, IOR, and Hpio.

IOzone is a file system benchmark used to a broad I/O performance analysis with a number of I/O access patterns on different platforms [23]. It supports a bunch of file operations, such as read, write, re-read, re-write, and read backwards, small/large file sizes, small/large record sizes, and single/multiple process I/O tests. We use IOzone to produce the I/O access cases of set 1 to 3 in Table 2.

IOR is a program in the ASCI Purple Benchmark Suite developed at Lawrence Livermore National Laboratory [25], it is a software used to test random and sequential I/O performance of parallel file systems. We use IOR to produce the concurrent I/O access cases in Table 2.

Hpio is a program designed by Northwestern University and Sandia National Laboratories to evaluate noncontiguous I/O performance for MPI-IO [24]. This benchmark program can generate various data access patterns by changing three parameters: region count, region spacing, and region size. In our experiment, we use Hpio to simulate the I/O access cases of set 4 in Table 2. Specially, we fix region count and region size, and vary region spacing from 8 bytes to 4096 bytes. When data sieving optimization is used, each process will access additional data required by the application.

In order to ensure that all data were accessed from storage devices, the system caches of all computing nodes and I/O servers were flushed prior to each run. We ran each set of experiments 5 times, and the average was used as the results. In addition, in order to show a clear comparison we normalized the CC values in the following way: If the value for each I/O metric showed a consistent correlation direction with the expected one listed in Table 1, we recorded it with a positive value; otherwise, we recorded it with a negative value. This applied to all the results related with CC values in this paper.

C. Experiment Results

1) Results of various storage devices

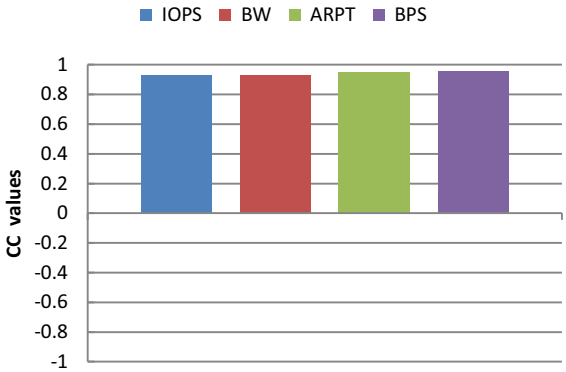

Figure 4. The normalized CC values of IOPS, bandwidth (BW), average response time (ARPT), and BPS.

We first watched the performance of IOPS, BW, ARPT, and BPS when the storage devices are changed, in number and in media. We ran IOzone in single process mode to read a 64GB file sequentially in different storage device configurations. In this case, the data was accessed through local file systems mounted on HDD, SSD, and a PVFS2 file system. In order to simulate different storage performance, here the PVFS2 was also built on different storage nodes, from 1 I/O server to 8 I/O servers, respectively. When all runs were finished, we correlated the values of IOPS, BW, ARPT, and BPS against the values of application execution time to calculate the final CC values.

Figure 4 reports the normalized results between each I/O metric and application execution time. All of the four metrics perform well, because not only they have right correlation directions, but also they have strong correlation strengths, with an absolute average value nearly 0.93. This strong correlation means that all of the traditional IOPS, BW, ARPT, and our BPS, work well for traditional storage device improvement.

2) Results of various I/O sizes

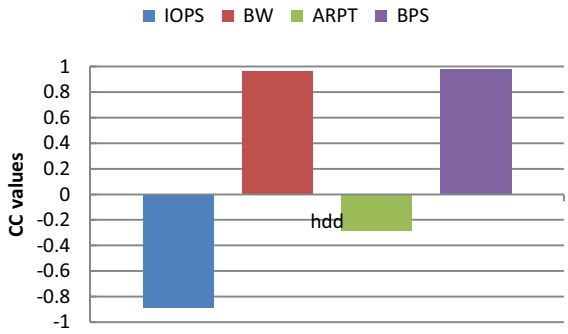

Figure 5. The normalized CC values of IOPS, bandwidth (BW), average response time (ARPT), and BPS in HDD environment

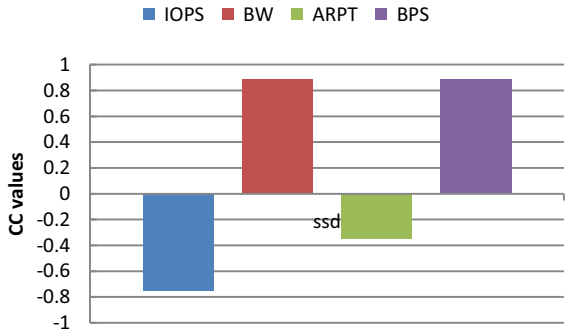

Figure 6. The normalized CC values of IOPS, bandwidth (BW), average response time (ARPT), and BPS in SDD environment.

For different I/O size testing, we ran IOzone to read a 16GB file from the local file system with the record size from 4KB to 8MB. When each run is finished, we collected IOPS, BW, ARPT, BPS, and application execution time. After all the runs were finished, we got the final CC values. We conducted the experiments with two storage configurations: one uses a slow HDD device; and the other a faster SSD device.

Figure 5 and Figure 6 show the normalized CC values in HDD and SSD environment respectively. The first observation is that both BW and BPS have correct correlation directions, but it is not true for IOPS and ARPT. The second observation is that both BW and BPS have

stronger correlation strengths, with average CC values nearly 0.90. These strong relations also reflect the fact that the overall system performance of I/O intensive applications largely depends on the I/O performance.

The reverse correlation direction of IOPS means that if IOPS increases, the overall computer performance may decrease, just as we mentioned in Section I. IOPS may be misleading because of the ignorance of I/O sizes. Given the same fixed I/O sizes, the larger the IOPS shows the higher the I/O performance. Otherwise, the conclusion is uncertain. Figure 7 gives the detailed results about IOPS and application execution time in the HDD environment when the I/O size is varied from 4KB to 8MB. When I/O size is 4KB, the IOPS is 5156.39 and the application execution time is 809.6 seconds; when I/O size is 64KB, the IOPS is 732.3 and the application execution time is 358.1 seconds. Obviously, the IOPS is largely decreased, but the overall computer performance is largely increased. This example shows that IOPS is ill-suited to evaluate I/O performance.

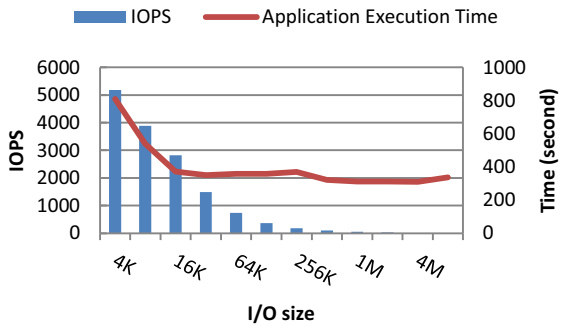

Figure 7. IOPS and application execution time with various I/O sizes in HDD environment.

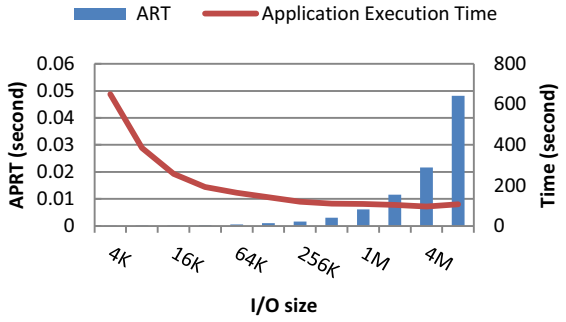

Figure 8. ARPT and application execution time with various I/O sizes in SSD environment.

Similar to IOPS, ARPT may be also confusing. Figure 8 shows the results of ARPT and application execution time when the I/O size was varied from 4KB to 8MB in the SSD environment. When I/O size is changed from 4KB to 4MB, ARPT is increased from 0.00014 to 0.02235 second, meaning a decreased I/O performance. However, the overall computer performance is largely increased because the I/O operations are performed with larger I/O sizes. From this example, we can see that ARPT has some limits to reveal the relation between the I/O performance and the overall computer performance accurately.

Using our car sale example, CC pointing to a wrong direction means improving the quality of car will reduce car sale. That is terribly wrong. The only explanation is that the measurement of quality has some problem. IOPS and ARPT are designed for a specific group of applications that issue fixed-size I/O requests. They are not feasible to be used to evaluate general HPC I/O systems.

3) *Results of various I/O concurrency*

In these experiments, we evaluated the performance of BPS in two kinds of environments where concurrent I/O requests exist.

We first ran IOzone in its throughput test mode to simulate a relative "pure" concurrent I/O environment, where the disk contention — a side effect led by I/O concurrency would be avoided. To this end, each process of IOzone accessed its own PVFS2 file, and each file is hosted on an individual I/O server. We limited each file to locate on one I/O server by setting the file stripe layout attributes when it was created. There were eight I/O servers in our experiments; the file system was accessed through the POSIX interface, and the total data amount of file accesses is 32GB.

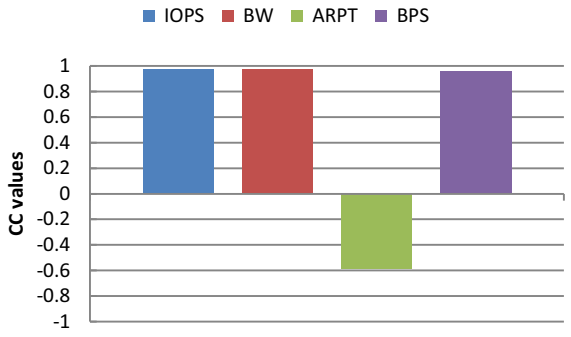

Figure 9. The normalized CC values of IOPS, bandwidth (BW), average response time (ARPT), and BPS.

Figure 9 demonstrates the results of each I/O metric when the number of processes was varied from 1 to 8. Among the four metrics, IOPS, BW, and BPS perform well. They not only show right correlation directions, but also have strong correlation strengths, with an absolute average value nearly 0.96. For ARPT, it has a wrong correlation direction, with a lower CC value nearly 0.58. The incorrect correlation direction shows that ARPT is not a good metric for I/O performance evaluation in this situation. When there are many I/O requests in the system,

the average response time is not able to determine the wall time of all these requests. So only considering the average response time is not appropriate to evaluate the overall computer performance. Figure 10 demonstrates the values of ARPT and application execution time when the I/O concurrency was changed. From the detailed results, we can observe that compared with the variation of application execution time, ARPT has a smaller variation, so it is not able to reflect the overall computer performance accurately. However, as seen from Figure 4, if I/O requests arrive at the I/O systems sequentially, APRT works as well as BPS.

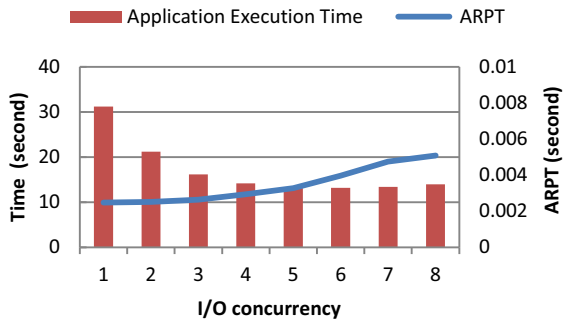

Figure 10. ARPT and application execution time with various I/O concurrency.

Secondly, we evaluated each metric's behavior in the I/O environment of general HPC systems. We ran IOR with the MPI-IO interface to access a shared PVFS2 file, which is striped across the underlying 8 I/O servers with a default stripe layout. Each of n MPI processes is responsible for reading its own 1/n of a 32 GB file. Each process continuously issues requests of fixed transfer size (64KB) with sequential offsets.

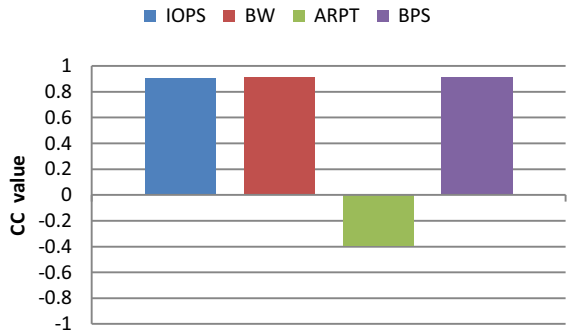

Figure 11. The normalized CC values of IOPS, bandwidth (BW), average response time (ARPT), and BPS.

Figure 11 demonstrates the results of each I/O metric when the number of process was varied from 1 to 32. We observed that in a real I/O environment, though IOPS, BW, and BPS perform a little worse than do they in a perfect environment, they still have good performance, with an absolute average value nearly 0.91. As expected, ARPT still has a poor performance, with a wrong correlation direction and a lower CC value nearly 0.39.

4) Results of various additional data movement

We compared BPS with the existing I/O metrics when additional data movement was existed in I/O system. We ran Hpio to simulate these I/O access cases. We tested the noncontiguous file read operation on PVFS2 file system configured with 4 I/O servers. Data sieving was enabled, so that I/O middleware (MPI-IO library) would read a bunch of additional file holes located between the adjacent file regions. The region count was set to 4096000, and the region size was set to 256 bytes. We varied the region spacing from 8 bytes to 4096 bytes to produce various additional data movement.

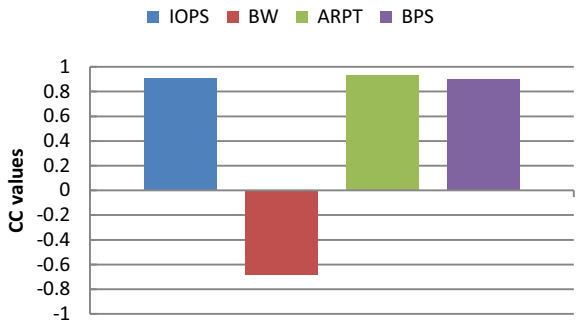

Figure 12. The normalized CC values of IOPS, bandwidth (BW), average response time (ARPT), and BPS.

Figure 12 gives the results of each metrics. We observed that IOPS, ARPT, and BPS have correct correlation directions with application execution times as we expect. At the same time, they have strong correlation strengths, with the absolute value of CC nearly 0.92. However, BW has a wrong correlation direction, which will mislead people. This result shows that BW is not a good I/O performance metric when additional data movement exists in I/O system. In other word, it cannot catch the I/O stack optimizations. In this case, file system performance does not represent I/O system performance. Using file system performance to represent I/O system performance is incorrect.

5) Summary of experiment results

Through the above experiments and analyses, via different I/O access scenarios, we have demonstrated that BPS is the only metric that works well for all the scenarios. BPS correctly correlates with the overall computer performance in all the tests, and achieves high CC values. In contrast, other I/O system metrics cannot accurately reflect the system performance, and all mislead in some testing scenario.

V. CONCLUSIONS

I/O is the identified performance bottleneck for many data intensive applications. Evaluating and understanding I/O system performance is a critical issue facing the HPC community. Existing I/O metrics work well for traditional I/O systems, but have some inherent flaws in catching the complexity of modern parallel I/O systems. In order to improve I/O performance, modern I/O systems have adapted various advanced optimization techniques in applications, I/O middleware, and storage systems and devices. Faced with these complicated I/O systems where concurrency and optimization at lower storage device level and higher I/O stack level, traditional I/O metrics are no longer effective.

In this paper, we propose a new I/O metric called BPS, describe its measurement methodology, and demonstrate its unique ability to measure the overall performance of modern I/O systems. Intensive experiments were conducted to verify the potential of BPS and to compare it with existing I/O performance metrics. Experimental results show that BPS is a significantly more effective I/O metric than other existing ones on reflecting the overall performance of I/O systems.

In the near future, we will conduct more performance measurements using BPS. We will make BPS an easy-to-use toolkit and release it to the public. With the better understanding of I/O system performance, we will adopt and evaluate different I/O optimization mechanisms and their combinations in terms of overall I/O system performance.

REFERENCE

[1] M. E. Thomadakis, "The Architecture of the Nehalem Processor and Nehalem-Ep Smp Platforms," A research report of Texas A&M university, Mar 2011.

[2] P. Kongetira, K. Aingaran, and K. Olukotun, "Niagara: A 32-Way Multithreaded Sparc Processor," *Micro, IEEE,* vol. 25, pp. 21-29, 2005.

[3] D. M. Tullsen, S. J. Eggers, J. S. Emer, H. M. Levy, J. L. Lo, and R. L. Stamm, "Exploiting Choice: Instruction Fetch and Issue on an Implementable Simultaneous Multithreading Processor," in *Proceedings of the 23rd Annual International Symposium on Computer Architecture*, 1996, pp. 191-202.

[4] S. Microsystems, "Lustre File System: High-Performance Storage Architecture and Scalable Cluster File System," Lustre File System White Paper, Dec 2007.

[5] F. Schmuck and R. Haskin, "Gpfs: A Shared-Disk File System for Large Computing Clusters," in *Proceedings of the 1st USENIX Conference on File and Storage Technologies(FAST'02)*, 2002, pp. 231-244.

[6] P. H. Carns, I. Walter B. Ligon, R. B. Ross, and R. Thakur, "Pvfs: A Parallel Virtual File System for Linux Clusters," in *Proceedings of the 4th Annual Linux Showcase and Conference*, 2000, pp. 317-327.

[7] D. Nagle, D. Serenyi, and D. Serenyi, "The Panasas Activescale Storage Cluster: Delivering Scalable High Bandwidth Storage," in *Proceedings of the 2004 ACM/IEEE conference on Supercomputing*, 2004.

[8] R. Thakur, W. Gropp, and E. Lusk, "Data Sieving and Collective I/O in Romio," in *The Seventh Symposium on the Frontiers of Massively Parallel Computation* 1999, pp. 182-189.

[9] R. Thakur, W. Gropp, and E. Lusk, "A Case for Using Mpi's Derived Datatypes to Improve I/O Performance," in *Proceedings of the 1998 ACM/IEEE conference on Supercomputing*, San Jose, CA, 1998, pp. 1-10.

[10] F. Wang, Q. Xin, B. Hong, S. A. Brandt, E. L. Miller, D. D. E. Long, and T. T. McLarty, "File System Workload Analysis for Large Scale Scientific Computing Applications," in *Proceedings of the 21st IEEE / 12th NASA Goddard Conference on Mass Storage Systems and Technologies*, 2004, pp. 139-152.

[11] N. Nieuwejaar, D. Kotz, A. Purakayastha, C. Sclatter Ellis, and M. L. Best, "File-Access Characteristics of Parallel Scientific Workloads," *IEEE Transactions on Parallel and Distributed Systems* vol. 7, pp. 1075-1089, 1996.

[12] E. Molina-Estolano, M. Gokhale, C. Maltzahn, J. May, J. Bent, and S. Brandt, "Mixing Hadoop and Hpc Workloads on Parallel Filesystems," in *Proceedings of the 4th Annual Workshop on Petascale Data Storage*, Portland, Oregon, 2009, pp. 1-5.

[13] C. Yong, S. Byna, S. Xian-He, R. Thakur, and W. Gropp, "Hiding I/O Latency with Pre-Execution Prefetching for Parallel Applications," in *High Performance Computing, Networking, Storage and Analysis, 2008. SC 2008. International Conference for*, 2008, pp. 1-10.

[14] S. Byna, C. Yong, S. Xian-He, R. Thakur, and W. Gropp, "Parallel I/O Prefetching Using Mpi File Caching and I/O Signatures," in *Proceedings of the International Conference forHigh Performance Computing, Networking, Storage and Analysis (SC2008)* 2008, pp. 1-12.

[15] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*: Morgan Kaufmann Pub, 4th edition, 2011.

[16] D. Bitton, M. Brown, R. Catell, S. Ceri, T. Chou, D. DeWitt, D. Gawlick, H. Garcia-Molina, B. Good, and J. Gray, "A Measure of Transaction Processing Power," *Datamation,* vol. 31, pp. 112-118, 1985.

[17] J. S. Vetter and A. Yoo, "An Empirical Performance Evaluation of Scalable Scientific Applications," in *Proceedings of the 2002 ACM/IEEE conference on Supercomputing*, 2002, pp. 1-18.

[18] H. Song, Y. Yin, Y. Chen, and X.-H. Sun, "A Cost-Intelligent Application-Specific Data Layout Scheme for Parallel File Systems," in *Proceedings of the 20th international symposium on High performance distributed computing*, San Jose, California, USA, 2011, pp. 37-48.

[19] X.-H. Sun and D. Wang, "Apc: A Performance Metric for Memory Systems," *ACM SIGMETRICS Performance Evaluation Review,* vol. 40, 2012.

[20] J. Doweck, "Inside Intel Core Microarchitecture and Smart Memory Access," http://www.iuma.ulpgc.es/~nunez/procesadoresILP/Intel64-Core-smartmemoryaccess-sma.pdf, Intel White Paper, 2011.

[21] S. Andersson, R. Bell, J. Hague, H. Holthoff, P. Mayes, J. Nakano, D. Shieh, and J. Tuccillo, "Rs/6000 Scientific and Technical Computing: Power3 Introduction and Tuning Guide," http://www.redbooks.ibm.com/redbooks/pdfs/sg245155.pdf, April 2011.

[22] "Performance Application Programming Interface," http://icl.cs.utk.edu/papi/, 2012.

[23] *Iozone Filesystem Benchmark*. Available: http://www.iozone.org/

[24] A. Ching, A. Choudhary, L. Wei-keng, L. Ward, and N. Pundit, "Evaluating I/O Characteristics and Methods for Storing Structured Scientific Data," in *Proceedings of the 20th International Parallel and Distributed Processing Symposium*, 2006.

[25] Interleaved or random (IOR) benchmarks, 2012. http://sourceforge.net/projects/ior-sio.