Performance Modeling of a Parallel I/O System: An Application Driven Approach *†

Evgenia Smirni   Christopher L. Elford   Daniel A. Reed   Andrew A. Chien

Abstract

The broadening disparity between the performance of I/O devices and the performance of processors and communication links on parallel platforms is a major obstacle to achieving high performance in many parallel application domains. We believe that understanding the interactions among application I/O access patterns, parallel file systems, and I/O hardware configurations is a prerequisite to identifying levels of I/O parallelism (i.e., the number of disks across which files should be distributed) that maximize application performance.

To validate this conjecture, we constructed a series of I/O benchmarks that encapsulate the behavior of a class of I/O intensive access patterns. Performance measurements on the Intel Paragon XP/S demonstrated that the ideal distribution of data across storage devices is a strong function of the I/O access pattern. Based on this experience, we propose a simple, product form queuing network model that effectively predicts the performance of both I/O benchmarks and I/O intensive scientific applications as a function of I/O hardware configuration.

1 Introduction

The I/O demands of large-scale parallel applications continue to increase, while the performance disparity between individual processor and disks continues to widen. Given these trends, effectively distributing data across multiple storage devices is key to achieving desired I/O performance levels. In turn, we believe that identifying effective operating points requires an understanding of the interplay among application I/O access patterns, data partitioning alternatives, and hardware and software I/O configurations.

Given the plethora of possible optimizations, determining preferred policy parameters by exhaustive exploration of the I/O parameter space is prohibitively expensive. Moreover, application developers need simple, qualitative models for choosing I/O parallelization strategies. Such models should encapsulate the performance implications of using either a smaller or larger number of disk, the effects of file access size, the granularity of data distribution across the disks, and the file access pattern. The goal of this paper is creation of such a model for parallel I/O.

Modeling disk arrays and parallel I/O systems has been an active research area for several years. Approximate analytical models of disk arrays [1, 2] using synthetic workloads assist the development of simple rules for preferred striping configuration in disk arrays. Our work complements these efforts by capturing the interaction of I/O requirements of scientific applications with *both* file system software and hardware. We construct a simple,

*This work was supported in part by the Advanced Research Projects Agency under DARPA contracts DABT63-94-C0049 (SIO Initiative), DAVT63-91-C-0029 and DABT63-93-C-0040, by the National Science Foundation under grant NSF ASC 92-12369, and by the Aeronautics and Space Administration under NASA Contracts NAG-1-613, NGT-51023 and USRA 5555-22.

†Department of Computer Science, University of Illinois, Urbana, Illinois 61801.

product form queuing network model that accurately models the basic performance trends of interleaved access patterns on the Intel Paragon XP/S Parallel File System (PFS). This model is appropriate for use by both application and file system developers.

The remainder of this paper is organized as follows. In §2, we describe QCRD, a large scientific code for solving quantum chemical reaction dynamics problems. This is followed in §3 by a description of synthetic benchmarks that drive a simple, product form queuing network model. In §4, we characterize the performance of the QCRD code and validate the model for several disk striping configurations. Finally, §5 summarizes our findings.

2 Quantum Chemical Reaction Dynamics (QCRD)

Understanding the interactions among application I/O access patterns, parallel file systems, and I/O hardware configurations is a prerequisite to identifying levels of I/O parallelism that maximize application performance. Thus, a major objective of the multi-agency Scalable I/O Initiative (SIO) is to assemble a suite of I/O intensive, national challenge applications, to collect detailed performance data on application characteristics and access patterns, and use this information to design and evaluate parallel file system policies. Below, we characterize the I/O performance of QCRD, one application from the SIO code suite, using an extended version of the Pablo performance analysis environment [3].

The QCRD application [5] uses the method of symmetrical hyperspherical coordinates and local hyperspherical surface functions to solve the Schrödinger equation for the differential and integral cross-section of the scattering of an atom by a diatomic molecule. Code parallelism is achieved by data decomposition (i.e., all processors execute the same code on different data portions of the global matrices). Via this data decomposition, the load is equally balanced across the processors, and code execution progresses in five logical phases (programs) that operate as a logical pipeline.

All our experiments were conducted on the Intel Paragon XP/S at the Caltech Center for Advanced Computing Research. As a platform for I/O research, this system supports multiple I/O configurations, including a set of older RAID-3 disk arrays and a group of newer Seagate disks. On all these configurations, the Paragon XP/S parallel file system (PFS) stripes file across multiple disks in default units of 64 KB.

As a baseline for our I/O performance analysis of the QCRD code, we considered a representative, though modest, data set and measured the performance and behavior of QCRD on the Caltech system when using 80 Seagate disks. The first four phases of the QCRD were executed on 64 processors, while the fifth phase was executed on 16 processors.

To compare the application I/O timings with an older disk configuration of 16 RAID-3 disk arrays, we then striped the data files across only 16 of the 80 Seagate disks and repeated the experiment. Not surprisingly, the newer 16 disks were faster than the older RAID-3 configuration. However, they also reduced application execution time by roughly ten percent compared to use of all 80 disks.

Finally, observing that increased I/O parallelism need not increase performance, we restricted the files to a single Seagate disk and repeated the experiment once more. Figure 1 shows the sum of time spent on I/O by all processors for the five QCRD phases. With the exception of phase one that achieves the best performance with only one disk, the cumulative I/O time is minimized when the data is striped across more than one, but less than all of the disks.

Table 1 illustrates the aggregate performance summaries for phase one and phase two of QCRD (the performance of the other phases are qualitatively similar to that of phase two and are not reported here for brevity's sake). The table shows clearly that the I/O

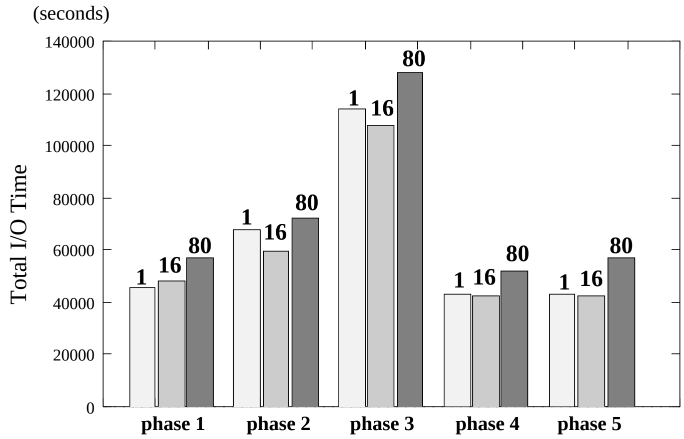

FIG. 1. *Cumulative QCRD I/O times (1, 16, and 80 disks).*

Phase One		Total I/O Time (seconds)		
Operation	**Count**	**1 Disk**	**16 Disks**	**80 Disks**
`open`	832	421.82	457.72	570.61
`seek`	35,724	45,084.12	46,279.36	54,441.27
`write`	31,252	2,068.41	1,509.17	1,759.71
`close`	768	6.55	1.12	1.12
Phase Two		**Total I/O Time (seconds)**		
Operation	**Count**	**1 Disk**	**16 Disks**	**80 Disks**
`open`	1,600	836.35	857.53	1,186.73
`read`	60,672	2,952.19	2,123.51	3,327.15
`seek`	65,664	62,150.16	55,708.28	65.847.36
`write`	3,600	1,672.38	586.23	1,679.12
`close`	1,536	101.20	168.42	429.10

TABLE 1
QCRD I/O operation frequencies and overheads.

times are dominated by seeks — the application developers chose to use the PFS UNIX file access mode because it is the most direct and portable analog to sequential UNIX I/O. Using this file mode, each processor repeatedly seeks to its designated part of the shared file before performing any read or write operations. In fact, the total time spent on seeks, usually negligible on sequential machines, dominates the total code execution time. Seeks represent roughly 10 percent of phase one's execution time, 50–60 percent of the execution time for phases two, three, and four, and almost 90 percent of phase five's execution time. The following section explores the reasons of this behavior in greater detail.

3 Microbenchmarks and Performance Models

In §2, we saw that understanding the interactions among application request patterns, file system semantics, and disk hardware configurations is critical to identifying effective operating points. Determining such points by exhaustively running the applications across the entire range of file system configurations is prohibitively expensive. An alternative, cost-effective method is to create an analytic model of parallel I/O performance that can predict efficient disk striping parameters for given request access characteristics. To identify the key elements of an effective I/O model and parameterize it accordingly, we first constructed a series of microbenchmarks. These benchmarks were designed to highlight

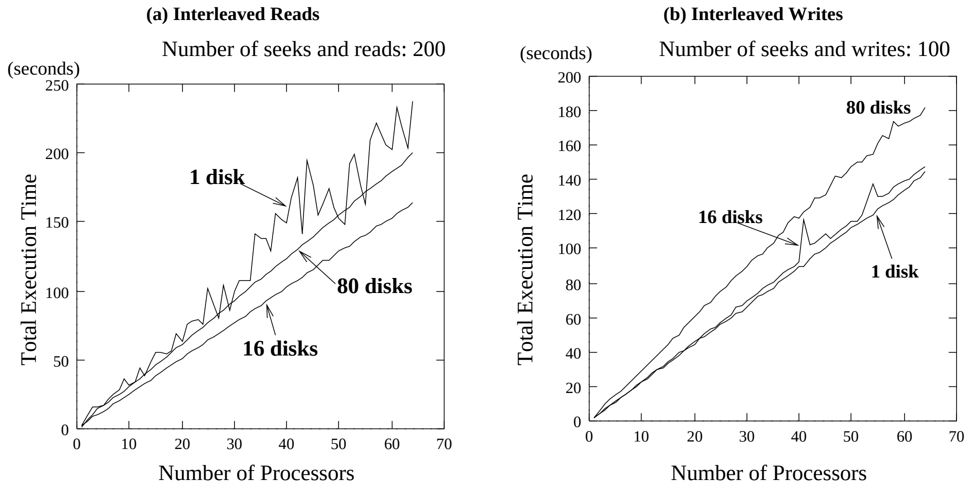

FIG. 2. *Microbenchmark execution times.*

system bottlenecks and to reflect the I/O behavior of actual applications.

As we noted earlier, application developers tend to use the UNIX I/O API because it is portable and because it is most familiar. Unfortunately, this does not exploit all available parallel file system features [4]. However, given the frequent use of the UNIX I/O API, we focus on modeling the Intel Paragon XP/S PFS performance characteristics using the default UNIX file access mode and the default 64 KB PFS stripe size.

As a first step, we constructed a synthetic workload that mimics the global interleaved access patterns found in the QCRD code. Each processor sequentially accesses its interleaved portion of the file, issuing a predefined number of synchronous I/O requests of the same size. We then parameterized this synthetic workload to control the load imposed on the I/O system. These parameters include the number of processors that simultaneously perform interleaved operations on the file, the request access size, the stripe group size, and the type of I/O operation (reads or writes). By varying different parameters, we incrementally increase the stress on the I/O system and identify performance bottlenecks.

Figure 2 shows the results of two experiments, one of interleaved reads (workload A) and one with interleaved writes (workload B) with files striped across 1, 16, and 80 disks. For workload A, there is a clear performance benefit if the file is striped across 16 disks — striping across the maximum number of disks is slower by twenty percent. For workload B, using a single disk is the most desirable alternative.

Figure 3 illustrates the average `seek`, `read`, and `write` durations for the two workloads as a function of the number of disks. For interleaved reads, there is a clear reduction in the average `seek` time from one to eight disks. Beyond this point, however, the mean seek time increases for all processor count combinations.

The seek operations for interleaved writes, shown in the lower portion of Figure 3, are much more expensive than those for reads and the costs increase rapidly with the number of disks. In turn, this suggests an optimal operating point for this interleaved write workload with the file striped across only 16 disks.

By construction, the interleaved read and write behavior of these two benchmarks is similar to that found in QCRD; see Figure 1. Moreover, the reasons for the convexity of both performance curves as a function of the number of disks used is the same.[1] For both

[1]The same qualitative behavior was detected for larger requests of 32 KB requests (i.e., half the disk

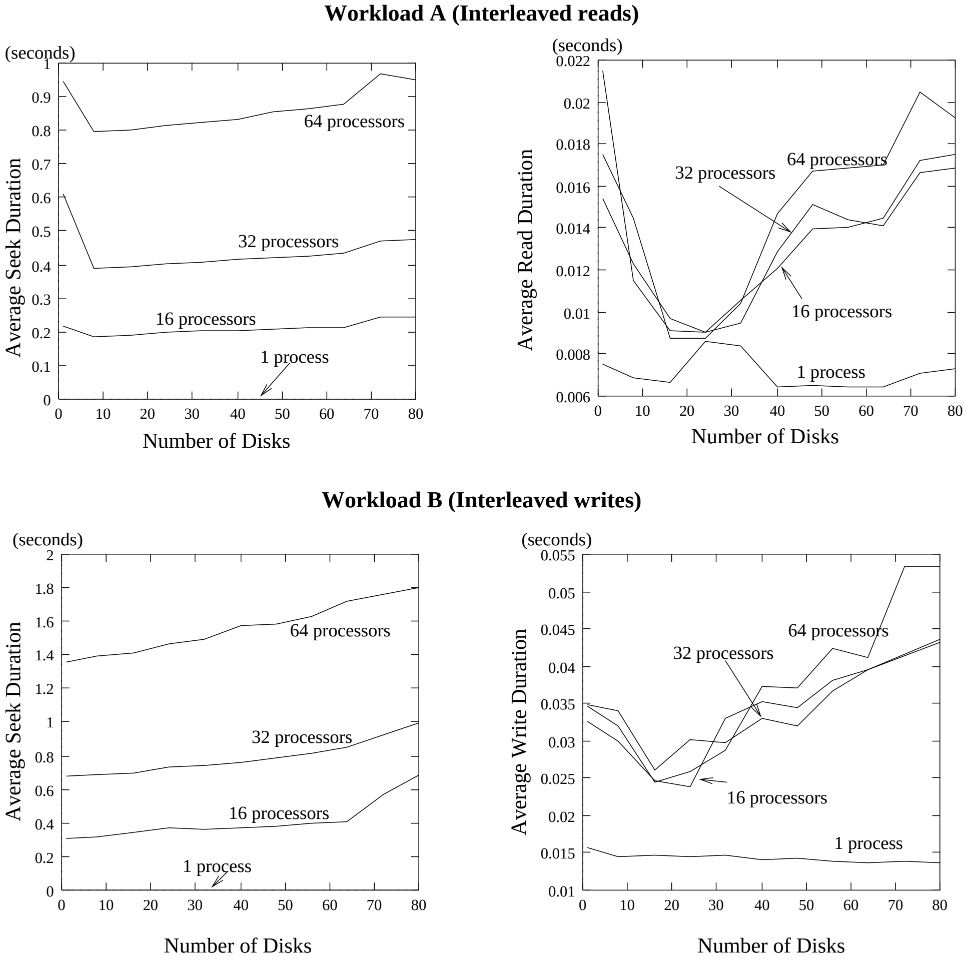

FIG. 3. *Average microbenchmark operation durations (2,400 byte requests).*

the benchmarks and QCRD, the average `seek` times are at least an order of magnitude more expensive than the associated `read` or `write` operations. The primary reason for these high seek costs lies in the Intel PFS implementation of UNIX file system semantics. PFS maintains sequential consistency for shared file pointers even when the file is opened read-only.

Using the microbenchmark data from Figures 2–3, we focused on modeling the effects of the PFS `open`, `seek`, `read`, and `write` primitives using UNIX access semantics. However, the models could be extended to include other PFS I/O access modes (e.g., `M_RECORD`, `M_ASYNC`) that relax consistency constraints.

To simply analysis, we assume that access times for all I/O operations are exponentially distributed, that requests are served by the I/O system in first-come-first-served (FCFS) order, and that all read and write requests are of the same size. Because files can be striped across a variable number of disks, a natural way to capture the effects of disk striping is via a fork-join system.

Unfortunately, the complexity of fork-join systems prohibits exact models that can be solved easily using analytical methods. Thus, we opted to use an approximate, single class,

---

striping unit) and 128 KB requests (i.e., twice the disk striping unit).

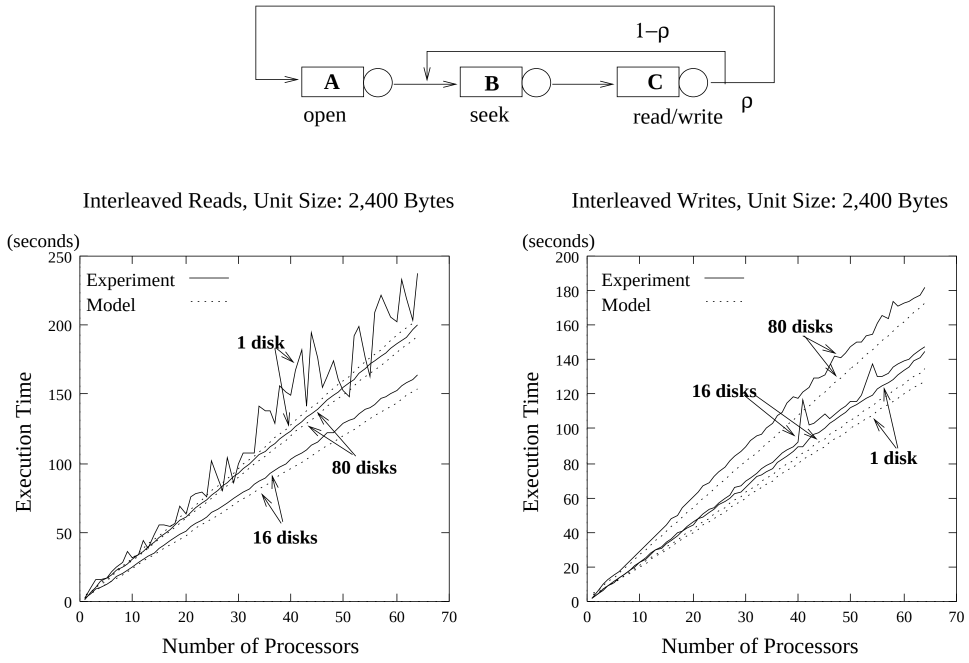

FIG. 4. *Model prediction for interleaved accesses (2,400 byte operations).*

closed queuing network that models N tasks that all execute the same sequence of I/O operations: they first open a common file, then perform a series of synchronous interleaved `read` or `write` requests. At each moment, we assume that N customers (i.e., tasks) circulate in the network.

To model the interleaved disk access pattern we used a closed queuing network with three devices; see Figure 4. The time consumed for `open`, `seek`, and `read/write` operations is modeled by servers A, B, and C, respectively. The resource demand on each server is a function of the server rate and the branching probability ρ shown in Figure 4. The server rates used as input to the model were taken from the microbenchmarks described earlier.

Figure 4 illustrates the model prediction for interleaved accesses for `read` and `write` operations. The model accurately captures the relative ranking of the experiments' execution time with respect to the number of disks. By analyzing the queue lengths at the various devices, we see that as the number of tasks in the network increases, a larger percentage of the workload's execution time is attributed to queueing delay at the seek server, just as shown by the microbenchmarks. For both reads and writes, the model accuracy is within ten percent (with the exception of interleaved reads and stripe group equal to one disk).

4 I/O Characterization and Model Prediction of QCRD

Because the five QCRD code phases are structured similarly, we concentrate on analysis of phase two, which is representative of all but phase one. Except for phase one, which performs a set of interleaved writes, the remaining of the QCRD phases contain both interleaved reads and writes.

Phase two executes the following sequence of steps. First, all processors synchronize, then open two *basis* files created by phase one. The two files are accessed in sequence with each processor seeking to its designated portion and performing 38 interleaved reads of 2,400 bytes each. After all processors have finished a two-dimensional quadrature, they

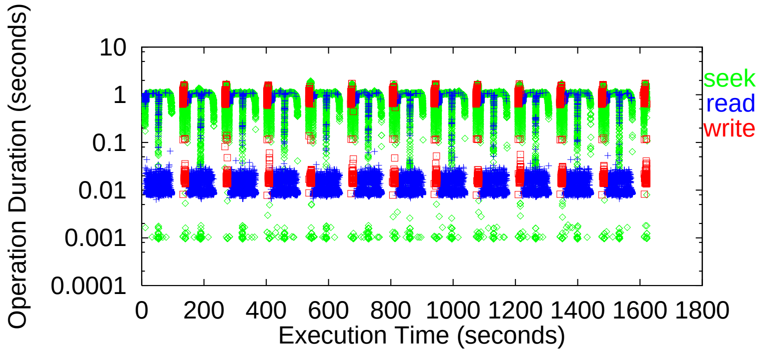

FIG. 5. *QCRD operation durations (phase two with 80 disks).*

open the same *overlap* file. Each processor then seeks to its designated portion and performs five interleaved writes of 2,400 bytes each. All of the steps above are then repeated twelve times.

Using our Pablo I/O instrumentation software, we captured a timestamped event trace of all I/O operations in QCRD phase two. Figure 5 illustrates the temporal spacing and duration of the `seek`, `read`, and `write` operations when files are striped across 80 disks. Figure 5 illustrates the twelve alternating bursts of I/O and computation activity. As previously discussed, file seeks are the largest source of I/O overhead.

Figure 6 shows a detailed view of `seek` durations for the first of the twelve I/O–compute cycles for three disk configurations. At the beginning of each I/O interval, seek durations increase rapidly until the system reaches a steady state. At the end of each I/O interval, the `seek` durations decline as the number of competing processors declines.

Using the simple queueing network model of §3, we predicted the I/O scalability of QCRD as a function of disk configuration. We parameterized the model's transition probability ρ using the operation I/O frequencies from the measured data and the service rates for servers B and C suggested by the experimental measurements of §3. Figure 7 illustrates the experimental and predicted I/O execution times of each interleaved operation portion for the first cycle of phase two. The model effectively captures the performance trends across the three disk configurations.

5 Conclusions

We demonstrated that a single distribution of data across I/O devices is unlikely to perform optimally for all file access patterns. Using a series of simple I/O benchmarks that encapsulate common access patterns, we measured the cost of I/O primitives with respect to request size, interaccess time, and operation interaction across various disk striping configurations We then constructed and parameterized a single class queueing network model that predicts benchmark and application behavior as a function of disk striping configuration. The major advantage of the model is its simplicity.

References

[1] CHEN, P. M., AND PATTERSON, D. A. Maximizing Performance in a Striped Disk Array. In *Proceedings of the 17th Annual International Symposium on Computer Architecture* (1990), pp. 322–331.

[2] LEE, E., AND KATZ, R. An Analytic Performance Model of Disk Arrays. In *ACM SIGMETRICS* (May 1993), pp. 98–109.

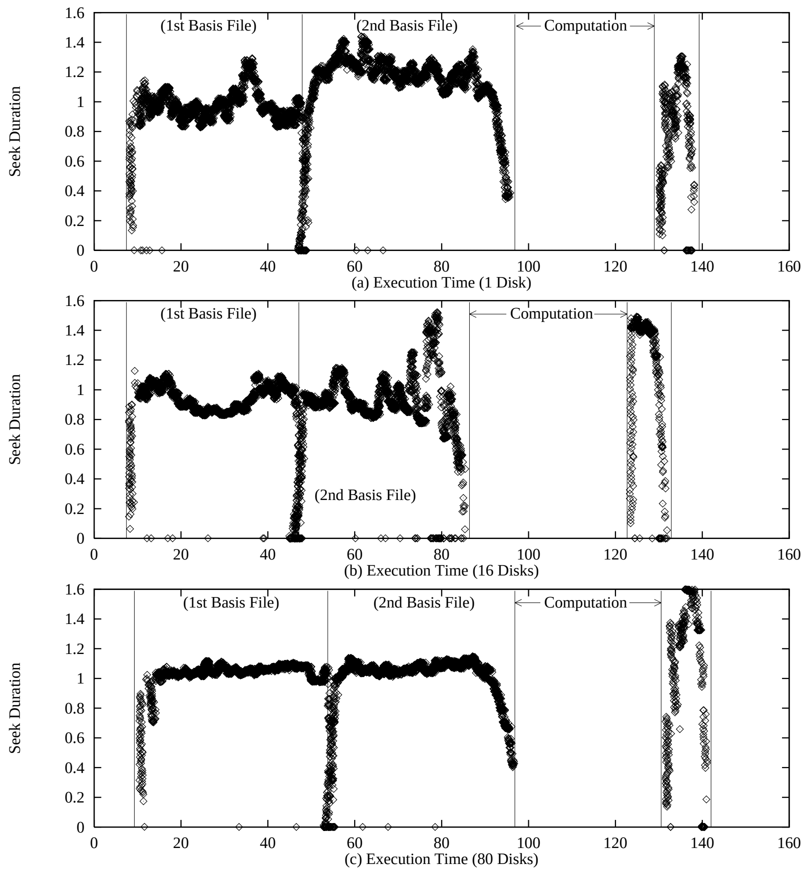

FIG. 6. *QCRD phase two seek durations (in seconds) for 1, 16, and 80 disks.*

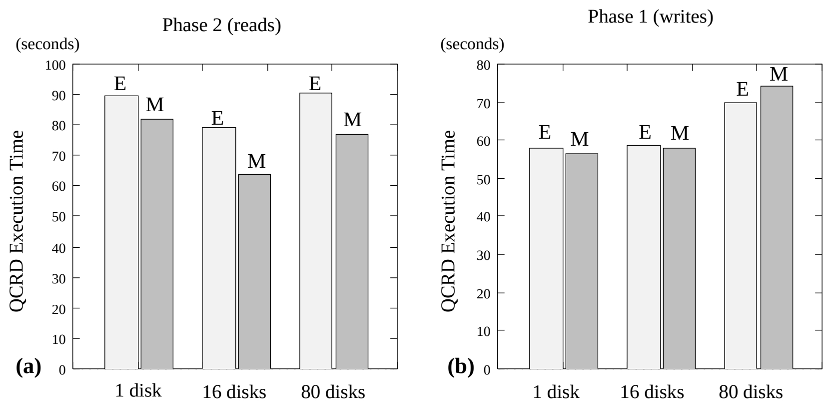

FIG. 7. *Model prediction for QCRD.*

[3] REED, D. A., ELFORD, C. L., MADHYASTHA, T., SCULLIN, W. H., AYDT, R. A., AND SMIRNI, E. I/O, Performance Analysis, and Performance Data Immersion. In *MASCOTS '96* (Feb. 1996), pp. 1–12.

[4] SMIRNI, E., AYDT, R. A., CHIEN, A. A., AND REED, D. A. I/O Requirements of Scientific Applications: An Evolutionary View. In *High Performance Distributed Computing* (1996).

[5] WU, Y.-S. M., CUCCARO, S. A., HIPES, P. G., AND KUPPERMANN, A. Quantum Chemical Reaction Dynamics on a Highly Parallel Supercomputer. *Theoretica Chimica Acta 79* (1991), 225–239.