Simulation Modelling Practice and Theory 34 (2013) 48–63

Contents lists available at SciVerse ScienceDirect

Simulation Modelling Practice and Theory

journal homepage: www.elsevier.com/locate/simpat

# A novel black-box simulation model methodology for predicting performance and energy consumption in commodity storage devices

Laura Prada, Javier Garcia, Alejandro Calderon, J. Daniel Garcia [*], Jesus Carretero

*Computer Architecture Group, University Carlos III of Madrid, Av. Universidad Carlos III, 22, 28270 Colmenarejo, Madrid, Spain*

A R T I C L E  I N F O

*Article history:*
Received 27 August 2012
Received in revised form 17 January 2013
Accepted 19 January 2013
Available online 28 February 2013

*Keywords:*
Storage devices simulation
Black-box probabilistic models
I/O systems performance
Scalable performance evaluation
I/O energy consumption
Scalable simulation

A B S T R A C T

Traditional approaches for storage devices simulation have been based on detailed and analytic models. However, analytic models are difficult to obtain and detailed models require a high computational cost which may be not affordable for large scale simulations (e.g. detailed data center simulations). In current systems like large clusters, grids, or clouds, performance and energy studies are critical, and fast simulations take an important role on them.

A different approach is the black-box statistical modeling, where the storage device, its interface, and the interconnection mechanisms are modeled as a single stochastic process, defining the request response time as a random variable with an unknown distribution. A random variate generator can be built and integrated into a bigger simulation model. This approach allows to generate a simulation model for both real and synthetic complex workloads.

This article describes a novel methodology that aims to build fast simulation models for storage devices. Our method uses as starting point a workload and produces a random variate generator which can be easily integrated into large scale simulation models. A comparison between our variate generator and the widely known simulation tool DiskSim, shows that our variate generator is faster, and can be as accurate as DiskSim for both performance and energy consumption predictions.

## 1. Introduction

As demonstrated by the successful emergence of the Green500 [8] list, which provides a ranking of the most energy-efficient supercomputers in the world, energy has become as significant as performance. Consequently, the performance-per-watt has been established as a new metric to evaluate systems. Many researchers have shown interest in identifying in which cases there is room for improvement in the context of power efficiency. As a result, a lot of inefficiencies in relation to energy have been identified. Research works show that a CPU resting in an idle state reaches about 50% of peak power consumption [10]. Storage subsystems alone represent roughly 10–25% of the power consumed by the data center [14]. High power consumption in storage systems is expected, as an idle machine with one processor and two disks can easily spend as much power on disks as on processors [22]. Storage consumption can become a greater problem in storage subsystems where the average number of disks per machine is in the dozens.

* Corresponding author. Tel.: +34 918561316. Fax: +34 918561270.
*E-mail address:* josedaniel.garcia@uc3m.es (J.D. Garcia).

1569-190X/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.simpat.2013.01.006

In this way, simulation techniques are commonly used for both performance and energy consumption evaluation of many applications and systems. In order to obtain realistic results, data access to either a file system or database management system cannot be ignored. However, a key element in the data access studies is the storage device simulation model. A storage device simulation model accepts as input the parameters of an application workload (as a flow of device requests) and generates a performance metric prediction. The output performance measurement may be a general performance metric such as average bandwidth, throughput, and latency. Such metrics give an idea of the global performance of the device. However, if the device simulation model is integrated on top of a large system model, such as file systems or database managers, a detailed metric as request's response time is needed. Even more, disk's manufactures provide response time as an average value however, these metrics are not suitable for detailed simulations.

Scalability is one the most critical issues for current system's wide simulators. This is the case of large clusters, peer to peer or volunteer computing models, grid storage infrastructures [7], and content delivery networks. In all these cases, expensive realistic simulations spend a large amount of computational resources and computation time.

Traditionally, storage devices have been modeled by means of detailed or analytic models based on devices geometry [26], zone splitting [35,5], or the use of read-ahead caches [9] and request reordering [30], reaching the emulation level in many cases. An alternate approach is given by the black-box simulation models, where almost no knowledge of the storage device is required. The main advantage of the black-box simulation models is that access patterns of the storage devices can be modeled using a sequence of random variables, which models the time required to service disk's requests. This sequence of random variables is a stochastic process. The goal of these simulation models are the generation of values which fits this stochastic process. Experimental data must be obtained and analyzed to fit the distribution behind.

In a previous work [11], we presented initial results using black-box simulation models. The model introduced in the previous work was not based on probabilistic distributions but on histogram representations. In this way, many experimental data must be stored to get enough accuracy, requiring high memory costs. Additionally, other issues such as effects of queuing, sequentiality, and inactivity periods were not considered.

We propose a novel black-box simulation model methodology, named Back-Box Model based on Probability distributions (BBMP). Our solution is based on probability distributions, which are specially fast for generating values and providing response times predictions from disk drives. Effects of queuing, sequentiality, caching, and inactivity periods are considered as well.

The main contributions of this work are the following. First, we present BBMP that aims to predict disk response times in a very fast way, saving time in large simulation process. This solution is also easily extensible to other storage devices such as SSDs (Solid State Disks). Second, our method incorporates the usage of a new response time measurement tool, which can be used in any kind of disk, with any kind of interface. Third, unlike other works [20], our evaluations use real workloads, or synthetic with characteristics typical of real workloads. Also, we compare our built models with the ones generated for the widely known simulation tool DiskSim [5]. Finally, as we will show in our evaluation section, the proposed method can be used not only to predict performance, but also to obtain energy consumption estimations from disk drives.

The rest of this article is organized as follows. In Section 2, we briefly discuss about related work. Section 3 gives an overview of the method used to build the simulation model. Section 4 describes the procedure used to obtain disk response times. Section 5 explains how to construct models and how to implement random variate generators. Next, Section 6 includes a description of an energy consumption model. The experimental results are presented in Section 7. Finally, Section 8 presents our conclusions.

## 2. Related work

Preliminary work has been performed by proposing simulation models that fall into three main model categories: Analytic, Detailed, and Back-box. Table 1 shows an advantage/drawback comparison of different simulation models for storage devices.

Analytic models use mathematical equations that summarize disk behaviors. Thus, predictions are usually fast. However, analytic models are difficult to obtain as it is required to know the hard disk drive internals, and for some of its parts it is not easy to reach to an equation that describes them. Examples of such models include single disk models [30,35] or array disk models [37,17,42]. Detailed models are usually very accurate, as they emulate the hard disk drive behavior. One widely known detailed simulator is DiskSim [5]. The DiskSim simulation tool is able to model commodity hard drives by using parameters which are extracted from disks, using semi-automated algorithms [40] or by means of the DIXtrac disk charac-

**Table 1**
Advantages and drawbacks of different disk model categories.

| | Advantage | Drawback |
|---|---|---|
| Analytic | Fast | Difficult to obtain |
| Detailed | Accurate | High cost |
| Back-box | Easy to obtain | Accurate? |

terization tool [28,29]. To give an idea of the complexity of the simulation model employed, DIXtract extracts almost 100 performance critical configuration parameters. However, this models is inappropriate when the number of disks are very high due to the high computational cost as a counterpart.

Back-box simulation models are easier to obtain because the internal disk characteristics is not required (as in the previous models). Here, disks are considered as black-boxes. However, for some of the existing approaches, it is not easy to obtain accurate models which are cheap in resources as well.

There are black-box models based on tables [2] and on Classification And Regression Trees (CARTs) [39,21]. Back-box simulation models based on tables store in memory, entries of input/output data from workloads and predictions. Missing data entries in the tables are interpolated from the most similar characteristics on the tables. These models are the most accurate of all. Accuracy can be even improved by adding new information to the tables. However, these models are not scalable, because large tables involve in slower searches, and therefore, a higher prediction time. CART models are an enhancement of black-box table-based models. In CART models goal/output data entries are organized in binary search trees whose leaves are accessed by using information from input. This generates faster predictions, but these models suffer from the same lack, scalability. Unlike other black-box simulation models, our proposed methodology predicts service times by choosing, at most, among five different probability distributions, which makes it faster when predicting. This approach reduces the simulation time significantly, allowing run more complex and exhaustive simulations (as we proof in Section 7).

Many researchers rely on traces that have been captured on production systems with real workloads. These traces are replayed through simulators in order to estimate power consumption and evaluate novel power-aware storage strategies. For example, a team of Microsoft researchers gathers live traces that have been previously captured from a production environments such as Hotmail and Messenger, and replays them in simulations in order to predict energy savings. Power consumption is simply estimated by the percentage of servers of the baseline systems that is powered on [34].

Zhu and Zhou [44] gather their energy usage from DiskSim [3], which they have extended with a storage cache simulator called CacheSim. Energy estimations are done by analyzing cache misses and the power management scheme. Also, both idle and active times are taken into account in order to estimate energy through simulations with traces as input data. Power consumption deviations are reported to be under 2%.

In order to simulate energy consumption of GreenHDFS, the authors feed their trace-driven simulator with data extracted from the data sheet of system components [16]. Similarly, the approach described by Narayanan et al. [22] is based on feeding simulators and testbeds with power data stemming from production manuals and traces gathered in production systems. Disk-level power consumption is measured in this way as well.

Allalouf et al. [1] contribute a power modeling framework called STAMP. The practical use of this framework is threefold. First, it offers a way to estimate online power consumption for storage systems. Second, the framework can be used as an online power-aware capacity planning tool. Statistical performance information represents the host storage workload which is used as input for the storage models. Finally, the use of statistical information allows the framework to be used for power and performance estimation on non-alive systems still in the design stage as well. Experimental results show how this modeling method achieves a reasonable error deviation of 2–9%, depending on the workload transfer size.

Knobloch et al. [18] analyze power performance of MPI applications. Their study is specially focused on energy saving opportunities that originate from busy-wait states that are shown to be highly power-inefficient. They extend the Scalasca [12] (scalable analysis of large-scale applications) tool-set, which is designed to detect wait states, focusing on energy efficiency. It is examined which power-states could be assigned to each wait-state in order to study potential power savings. Although their work focuses on benchmarking CPU power performance, exploitation of idle states is something which every system component can take advantage of, including and especially disks in storage systems.

In a previous work [11] we presented initial results using black-box models based on histogram representations. Using histogram representations, many experimental data must be stored to get enough accuracy, requiring a lot of computational resources. In this paper, the proposed black-box model is based on probabilistic distributions. Unlike the previous models, information is gathered into one or several probabilistic distributions. Thus, the only information to host are the parameters that characterize them, avoiding the previously mentioned scalability problems.

## 3. BBMP overview

Our modeling methodology is composed of four main step or tasks as described in Fig. 1.

BBMP starts with a sequence of disk requests (from a synthetic workload or real traces repositories) and generates a variate generator capable of producing several instances of simulated service time traces. The first task to accomplish consists on obtain access patterns that include service time for each single disk operation. Those workloads come from two different sources: real I/O traces from a specific system or synthetic workloads modeling a set of I/O traces. The selection of the source for the I/O requests depends on the purpose of the simulator to be produced. Our method allows to work with real I/O traces as well as with synthetic workloads, as soon as a common representation is used. It is important to remark that the simulation module is dependent on the I/O requests data sets. As an example, if we use I/O requests data sets coming from a Web

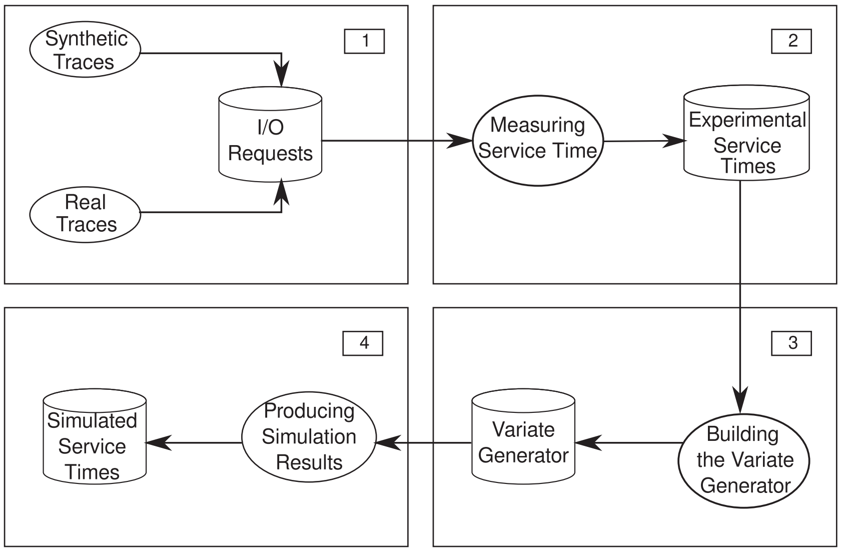

**Fig. 1.** Steps in black-box simulation modeling methodology.

server, the generated simulation module will be accurate to simulate disk service time under those load conditions and not under the load given by a database server.

Next step (step 2) consists on obtaining samples of experimental traces with disk service time measurements. We perform different realizations of the measurement procedure. In each realization, requests are sent to the disk and the service time for each individual request is measured, recording all the information in an experimental traces repository. Section 4 describes in detail this procedure.

After experimental measurement, obtained data are analyzed in order to build a variate generator (step 3). Here, distributions behind datasets are searched. Finally, the variate generator is included in a simulation module (step 4) to produce simulation results. After several realizations of the simulation are run with the initial workload used in step 2, simulation results are compared with experimental data to determine accuracy.

## 4. Service times measurement tool

It is noticeable to remark that the goal here is not to get an overall metric, such as response time mean or throughput. In contrast, our first objective focus on measure every request in order to build a simulation model. Once a workload is available, the next step consist of measuring the response time for every request. A workload dataset will be represented as a file which has as many lines as I/O requests have the application or system. Each line contains details about a single request such as the address of the disk at which is targeted, its size, the type of operation, and the timestamp when the request is performed. A performance evaluator takes as input a previously obtained workload and generates response times for this characterized workload.

To the best of our knowledge, *dxreplay* is the only tool that aims to achieve this goal. *dxreplay* is included in the package *DIXtrac* [28,29]. Having a certain workload, *dxreplay* performs response time measurements on SCSI disks in order to compare them with the measurement results taken from the DiskSim simulator. This tool is used for validating the correct extraction of the parameters that conform the detailed simulation model of a real disk in DiskSim. It is composed of a main stream, which creates three threads: *lbnreader*, *issuer*, and *collector*. *lbnreader* reads requests from an workload input file; *issuer* launches the previously read requests to the disk at its specific timestamp; and *collector* waits for the launched requests to be finished. The three threads are executed by turns using synchronization primitives. However, the main lack of *dxreplay* is that only works with SCSI disks in a non-generic way.

In order to deal with the previous lacks, we propose *play*, a performance evaluation tool that performs response time measurements on any kind of disk or interface. Unlike *dxreplay*, *play* obtains measurements by using standard POSIX calls and not specific SCSI commands. It is composed of a main stream (as shown in Algorithm 1). *play* reads every request from the workload (line 3) and executes at the specified timestamp (line 5). This timestamp is recorded for a later calculation of the request's response time (line 4). It respects arrival request's timestamps by waiting the required period until the next request (line 6). *play* also contains a signal handler (as shown in Algorithm 2), which turns on when a specific request has finished and its response time can be calculated (line 2), and reported directly to the output file (when making debugging tasks), or recorded on an in-memory data structure (as shown in line 3).

**Algorithm 1.** play

```
1: activate(signal_handler, wake_up_when_a_request_finishes)
2: loadRequestsInMemoryFromWorkloadFile()
3: for $i \Leftarrow 1$ to maxNumberOfRequests do
4:     recordStartTimeOfRequest(i)
5:     lauchRequestToDiskAtItsTimeStamp(i)
6:     waitForNextRequest(i)
7: end for
8: writeResponseTimesToOutputFile()
```

**Algorithm 2.** signal_handler

```
1: recordEndTimeOfRequest(finished_request)
2: calculateResponseTime(finished_request)
3: reportResponseTimeOnDataStructure(finished_request)
```

To avoid any impact in the evaluation procedure from the file system layer, and to ensure that every I/O request in the workload is physically sent to the evaluated disk, we define the disk as a raw character device and associate it to the disk that is going to be evaluated. In that way, the device can be accessed directly by using generic POSIX calls. We also remove the effect of the operating system I/O scheduler for it not to reschedule the requests and they can keep their arrival times. *play* works similar to *dxreplay* (disk level) but using a higher lever interface which makes it more versatile. *play* also incorporates a characteristic that allows the user to decide whether response times include disk queuing times effects. Thus, the maximum number of enqueued requests is restricted to be 1 when queuing times effects are required to be removed. When that restriction is not applied, queuing times become a part of response times.

## 5. Variate generator module

In this section we describe the third step of our methodology. We show how to build a variate generator with the previously obtained in the previous step. We start with a sample of response times and we end up with a variate generator capable of producing simulated response times. Our method consists of obtaining response times, detecting statistic distributions, fitting to statistic distributions, and finally constructing a simulation model.

Our first data set is a sample of response times obtained from replaying a trace on a real disk. This sample was obtained by running *play*, our response time measurement program presented in Section 4. Next task performed consist of detecting statistic distributions from the previously obtained response times sample. Usually, samples present several different distributions that cover different ranges of the response time domains. Each distribution can be the result of different causes that generate response times from one distribution. As causes, we initially identify that a specific request may be serviced from the platters of the disk or from its buffers, may have to wait for the previous requests to be serviced (scheduling), etc. After detecting possible distributions, the identified distributions must be fitted to some known distributions. BBMP takes advantage of the *R* statistical analysis environment [41] in order to fit the samples to know theoretical distributions. This method has been described in a previous work [25]. Finally, the fitted distributions are used to construct the model.

### 5.1. Constructing simulation models

The construction model step focus on deduce the causes and major parameters, for the previously detected, and fitted-to-theoretical-distributions samples. For causes and parameters, we mean the workload characteristics that produce response times from a certain distribution. Major parameters addressed in this work are the following:

- *Long inactivity periods.* When the disk has been idle for a certain amount of time, the next request to be serviced may last longer than as usual.
- *Queuing times.* When a request to be serviced arrives when one or several previous requests have not finished yet, it must wait until the end of the previous ones, making its response time bigger.
- *Sequentiality*. When several requests are sequential, the difference in LBNs (logical block number) between them is small. On the contrary, when the difference in LBNs is big enough, sequentiality may not be identified and response times are bigger.
- *Caching effects.* Disk drives have internal buffers. When their usage is activated, servicing requests from the disk drive buffers is faster than servicing requests directly from the platters.

Once we have identified the causes for generating response times from each specific fitted distribution, we construct a model in which, depending on the input data, we choose theoretical distributions in order to generate accurate response times.

In an example of a Seagate Cheetah 10K.7 disk under a Financial trace [36], we identified two distributions and three causes for them. Distribution 1 is a mixture of two Normal distributions (as shown in Table 2). Response times from part 1 of Distribution 1 are generated when requests must be serviced from the platters. Response times from part 2 of Distribution 1 are generated when inactivity periods are longer than 1 s and when requests serviced from the platters, must wait until the end of the previous ones. Finally, response times from Distribution 2 are generated when inactivity periods are around half a second. For this specific model, we constructed the algorithm as shown in Algorithm 3.

**Algorithm 3.** Simulation model example for a Seagate Cheetah 10K.7 disk under the Financial trace

```
1: **if** $((simTime()\text{-}ini\_disk[actReq-1]) > 520) AND ((simTime()\text{-}ini\_disk[actReq-1]) < 700)$ **then**
2:    $choose \Leftarrow bernoulli(0.92208)$
3:    **if** $choose = 1$ **then**
4:       $response\_time \Leftarrow normal(47.04, 6.385)$
5:    **else**
6:       $response\_time \Leftarrow normal(29.42, 3.993)$
7:    **end if**
8: **else**
9:    **if** $((simTime()\text{-}ini\_disk[actReq-1]) > 1000)$ OR $(simTime() < end\_disk[actReq-1])$ **then**
10:      $response\_time \Leftarrow normal(10.819, 5.706)$
11:   **else**
12:      $response\_time \Leftarrow normal(3.727, 1.965)$
13:   **end if**
14: **end if**
```

### 5.2. Models based on real traces

For each real trace we distinguish among several reasons to generate response times from one fitted distribution, if there exist more than one distribution. If there exist only one distribution, all response times are generated from it.

Finally, we count with a frequent simulation model repository from several real traces with different characteristics. Predicting from this simulation model involves choosing one of several previously modeled traces. The criteria of selection is based on the mean request size and the mean queuing time of the trace to model. Sequentiality is also taken into account. On the basis of the mentioned criteria, a previously modeled trace is selected and used to predict response times from an input workload. The criteria is checked again, and if a substantial characteristic has changed, another most similar previously modeled trace is chosen. In order to guarantee the accurate ratio, we select at least one of the previously modeled traces with similar characteristics as the trace to be predicted. Algorithm 4 shows how the prediction in this way works.

Before the first request arrives, *contLastReqs* is initialized. *lastReqs* is a constant which determines the maximum number of arrived requests, before checking whether another distribution has to be selected. If that is the case, the method *calculateDistr*() is executed (line 22). This method determines, on the basis of the previously mentioned criteria, which distribution is the most appropriate for the input trace to predict. According to the selected distribution, which value is in variable *distr*, response times are generated from it (lines 2–16). Every time a response time must be generated for a specific request, the statistics for the input workload must be also updated. Those statistics include mean request size (*addReqSizeToStatistics*(), line 18), mean queuing time (line 19), and sequentiality (line 20).

**Table 2**
Parameters of the probabilistic distributions that model response times obtained from a Seagate Cheetah 10K.7 disk under a Financial trace.

| | Type | $\pi$ | $\mu$ | $\sigma$ |
|---|---|---|---|---|
| Distribution 1 | Normal | 0.89 | 3.73 | 1.96 |
| | Normal | 0.11 | 10.82 | 5.71 |
| Distribution 2 | Normal | 0.92 | 47.04 | 6.38 |
| | Normal | 0.08 | 29.42 | 3.99 |

**Algorithm 4.** Model based on several real traces for a Seagate Cheetah 10K.7 disk

```
if ($contLastReqs < lastReqs$) then
  if ($distr = 0$) then
    $response\_time \Leftarrow generate\_S3D()$
  end if
  if ($distr = 1$) then
    $response\_time \Leftarrow generate\_BTIO()$
  end if
  if ($distr = 2$) then
    $response\_time \Leftarrow generate\_MadBench()$
  end if
  if ($distr = 3$) then
    $response\_time \Leftarrow generate\_Financial()$
  end if
  if ($distr = 4$) then
    $response\_time \Leftarrow generate\_Cello99()$
  end if
  $contLastReqs$++
  $addReqSizeToStatistics()$
  $addQueuingTimeToStatistics()$
  $addSequentialityToStatistics()$
else
  $calculateDistr()$
  $contLastReqs \Leftarrow 0$
end if
```

## 5.3. Models based on synthetic traces

The big difference with regard to build models based on real traces is that queuing times are not included in response times samples. Besides, it uses only one trace. We remove queuing times from response time samples by using one of the features of *play*. As previously discussed, it allows users extract response times by avoiding queuing times. It can be done by restricting the maximum number of pending I/O requests that can be enqueued to the disk to one request. Thus, queuing times are modeled on-line in the predictions. As queuing times are one of the most significant and distinguishing effects in response times, if every input trace predicts them on its own, models can be more versatile and general. The purpose of this is finally to have a more versatile model by simulating the queuing times on the fly. In Algorithm 5 we show, for the previously constructed model of the Seagate Cheetah 10K.7 disk, under the Financial trace [36], how to calculate queuing times on-the-fly. Although the chosen trace is not synthetic, our goal here is to show the difference between generating queuing times from a previously modeled distribution and when generating them on-the-fly.

**Algorithm 5.** Model for a Seagate Cheetah 10K.7 disk, under the Financial trace. Queuing times are calculated on the fly

```
if $((simTime()\text{-}ini\_disk[actReq-1]) > 520)\&((simTime()\text{-}ini\_disk[actReq-1]) < 700)$ then
  $choose \Leftarrow bernoulli(0.92208)$
  if $choose = 1$ then
    $response\_time \Leftarrow normal(47.04, 6.385)$
  else
    $response\_time \Leftarrow normal(29.42, 3.993)$
  end if
else
  if $((simTime()\text{-}ini\_disk[actReq-1]) > 1000)$ then
    $response\_time \Leftarrow normal(10.819, 5.706)$
  else
    $response\_time \Leftarrow normal(3.727, 1.965)$
  end if
end if
if $(simTime() < end\_disk[actReq-1])$ then
  $response\_time \Leftarrow response\_time + (end\_disk[actReq-1] - simTime())$
end if
```

As previously said, the generation of response times from Distribution 2 is described in lines 1–8. Every time the disk has been idle for more than 520 ms and less than 700 ms, the actual request, *actReq*, generates its response time from one of the parts of Distribution 2. We determine that idleness by doing a subtraction between the timestamp, when *actReq* arrives (*simTime*()), and the time stamp when the previous request arrives ($ini\_disk[actReq - 1]$). Probability of choosing part 1 of Distribution 2 is higher (0.92208) than choosing part 2. We determine which part to use by using a Bernoulli distribution (line 2).

The generation of response times from Distribution 1 is described in lines 9–14. Every time the disk has been idle for more than 1000 ms, the actual request, *actReq*, generates its response time from part 2 of Distribution 1 (line 10). In any other case, response times are generated from part 1 of Distribution 1 (line 12). If *actReq* arrives (*simTime*()), and the previous request *actReq*–1 has not been serviced yet, *actReq* is enqueued. To simulate that it has been enqueued, we calculate its response time, by adding the time that it stays in the queue ($end\_disk[actReq - 1] - simTime()$, line 16) up to its previously calculated service time (*response_time*).

## 6. Energy consumption model

In a computer system, disk drives are one of the most power consuming elements. This is mainly due to their mechanical nature. Disk drives usually have several platters in which data are stored. To access those data, several heads, moved by an arm, are used. Energy consumed by disk drives affects both laptop/desktop and data centers. In the case of laptop/desktop environments, energy consumed by disk drives reduces the amount of power available in batteries and power supplies. In the case of large data centers, energy consumption of disk drives increases, mostly, expenses in electricity bills, and also $CO_2$ dissipation. That is mainly because in data centers many disk drives are used in order to improve performance, by servicing I/O requests in a parallel fashion [24].

With an aim toward saving power, current disk drives have several states of energy: *active*, *idle*, and *standby*. A disk is in active state when it is reading or writing data. When a disk is not doing anything, but is still spinning, its state is idle. When a disk is not doing anything and its platters do not spin either, its state is standby. The highest power consumption occurs at the active state. Idle state consumes less power than active, and standby state consumes much less power than the previous two states. The disk spins down to the standby state when it has been idle for a certain period and it is predicted that a state change is worth it. When an I/O request needs to be serviced by a disk that is in the standby state, it has to spin up before anything, and then service the I/O request.

As commented before, the main goal of the BBMP methodology is to simulate disk response times in a fast way. In order to calculate a disk energy estimation it is required to know its response times. Thus, BBMP can also be used to obtain energy consumption estimations. We propose a power consumption simulation model which works with BBMP. The power consumption simulation model employs an extension of the 2-Parameter Model described in [43]. This model applies the following formula to calculate disk energy consumption estimations:

$$E_{disk} = E_{activeDisk} + E_{idleDisk} + E_{standbyDisk} \tag{1}$$

$$E_{idleDisk} = P_{idleDisk} x T_{idleDisk} \tag{2}$$

where $E_{activeDisk}$ is calculated as $P_{activeDisk} x T_{activeDisk}, P_{activeDisk}$ is the power consumption when the disk is active, and $T_{activeDisk}$ is the time spent by the disk while satisfying disk requests. $E_{idleDisk}$ is calculated in Eq. (2) where $P_{idleDisk}$ is the power consumption in the idle mode and $T_{idleDisk}$ is the length of the idle period. $E_{standbyDisk}$ is calculated as $P_{standbyDisk} x T_{standbyDisk}, P_{standbyDisk}$ is the power consumption in the standby mode and $T_{standbyDisk}$ is the length of the standby period. Eq. (1) is applied to the execution time line, in which response times from the disk model are included, and energy is obtained in terms of Joules.

Integrating the energy consumption model in BBMP, we are able to predict the energy consumption of I/O intensive workloads commonly used in computers and data centers as we show in Section 7.

## 7. Evaluation

We have evaluated our solution on a SCSI disk. The chosen disk is a Seagate Cheetah 10K.7 and its main features are summarized in Table 3. We constructed our models on top of the OMNeT++ discrete event simulation environment [38].

We use several different block-level traces, common in data-intensive I/O systems. Financial [36] is the I/O core of an OLTP application gathered at a huge Financial organization. It performs about 5 million requests over 24 disks. Cello99 [15] belongs to a shared compute/mail server from HP Labs. It performs about 6 million requests over 25 disks.

Other research works have investigated several I/O intensive parallel scientific applications, such as MadBench2 [19], S3D, and BTIO [23], to mention a few. We have evaluated our methodology, using real traces of the previous cited high performance applications. BTIO and Madbench traces were obtained from a beowulf cluster with 4 I/O nodes, which uses PVFS2 [6] as a high performance parallel file system. The storage system consist of four magnetic disks and file blocks (64 KBytes) are mapped round-robin over all disks. S3D traces were obtained from Red Storm [27] trace repository. They were extracted from a Cray XT3+ class machine, which uses Lustre as file system. The block size used was 1 MB and the storage system counted with 320 magnetic disks, in which file blocks were mapped round-robin over all disks.

**Table 3**
Manufacturer specifications for a Seagate Cheetah 10K.7 model ST373207LC.

| Specification | Metric |
|---|---|
| Heads | 2 |
| Discs | 1 |
| Bytes per sector | 512 |
| Bytes per track | 556 KBytes |
| Default read/write heads | 2 |
| Spindle speed | 10,000 rpm |
| I/O data transfer rate | 320 MBytes/s |
| Formatted capacity | 73.4 GBytes |
| Guaranteed sectors | 143,374,744 |
| Cache buffer | 8 MBytes |
| Average latency | 3 ms |

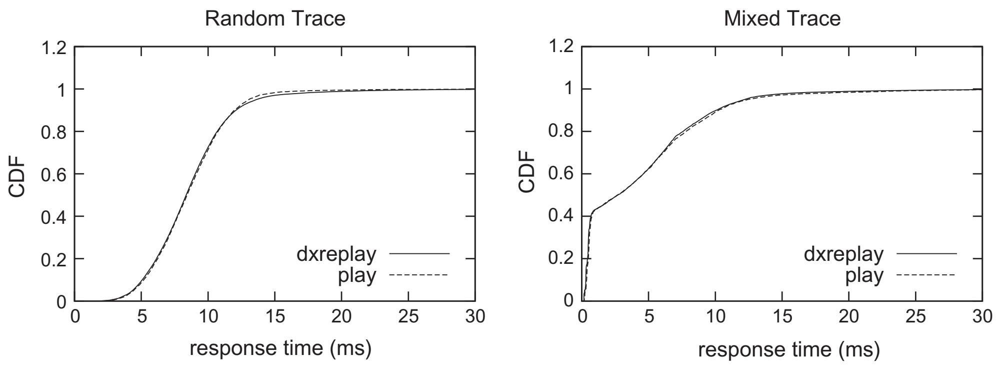

**Fig. 2.** Response times CDFs superimposition from a Seagate Cheetah 10K.7 disk and two synthetic traces.

Finally, we present results for two synthetic traces, *Random* and *Mixed*. *Random* has 10,000 requests, of which 66.6% are reads and 33.3% are writes. The LBNs are random and are distributed across the entire disk. Request sizes range from 1 KB to 8 KB. *Mixed* has 5000 requests, in which 66.6% are reads and the rest are writes. 20% of the requests are sequential and 30% are local. The remaining 50% have random LBNs. Request sizes also range between 1 KBytes and 8 KBytes. *Random* was evaluated by disabling the disk cache. *Mixed* was executed after enabling again the cache, and hence, the read-ahead and immediate write reporting.

## 7.1. Service time measurement

Fig. 2 shows CDFs (Cumulative Distribution Functions) superimposition results of running two synthetic traces on the Seagate Cheetah 10K.7 disk by using *dxreplay* (included int the *DIXtrac* tool), compared to executions by using *play*, our service time measurement tool.

We have used the demerit error metric [26] in order to validate our service time measurement tool. Demerit is defined as the root mean square of the horizontal distances between *dxreplay* and *play*. We presented the demerit results in absolute terms (as a difference in milliseconds), for comparison with other *DIXtrac* prediction models [4].

Demerits obtained for *Random* and *Mixed* are 0.59 ms and 0.47 ms respectively. This is a quite good match, so consider *play* as validated and we use it in the rest of the evaluations of ths work.

## 7.2. Disk drive modeling comparison

Fig. 3 plots CDFs superimposition results of modeling and executing the five previously mentioned traces on BBMP, DiskSim, and a real Seagate Cheetah 10K.7 disk, with no cache and cache effects. In order to clarify the results, we summarize total demerits of the previous mentioned traces in Table 4. The DiskSim simulation model parameters were obtained by running DIXtrac on the Seagate Cheetah 10K.7 disk. Fig. 3 plots the CDF curves for both real disk and models outputs and used the demerit error as metric to validate the models against the real disk. Demerit figures are presented in relative terms as a percentage of the mean response time. We have shown results for both enabling and disabling disk caches (left and right column respectively) in order to show the disk cache effects over the curves, and also because it is a common practice used to disable disk caches on individual disks used for databases or similar environments due to possible data integrity problems.

On the left column of Fig. 3, we plot demerits for the previously mentioned traces when the disk cache is disabled. For some traces, demerits are better in our approach than in DiskSim, and vice versa. In any case, most of them, keep roughly

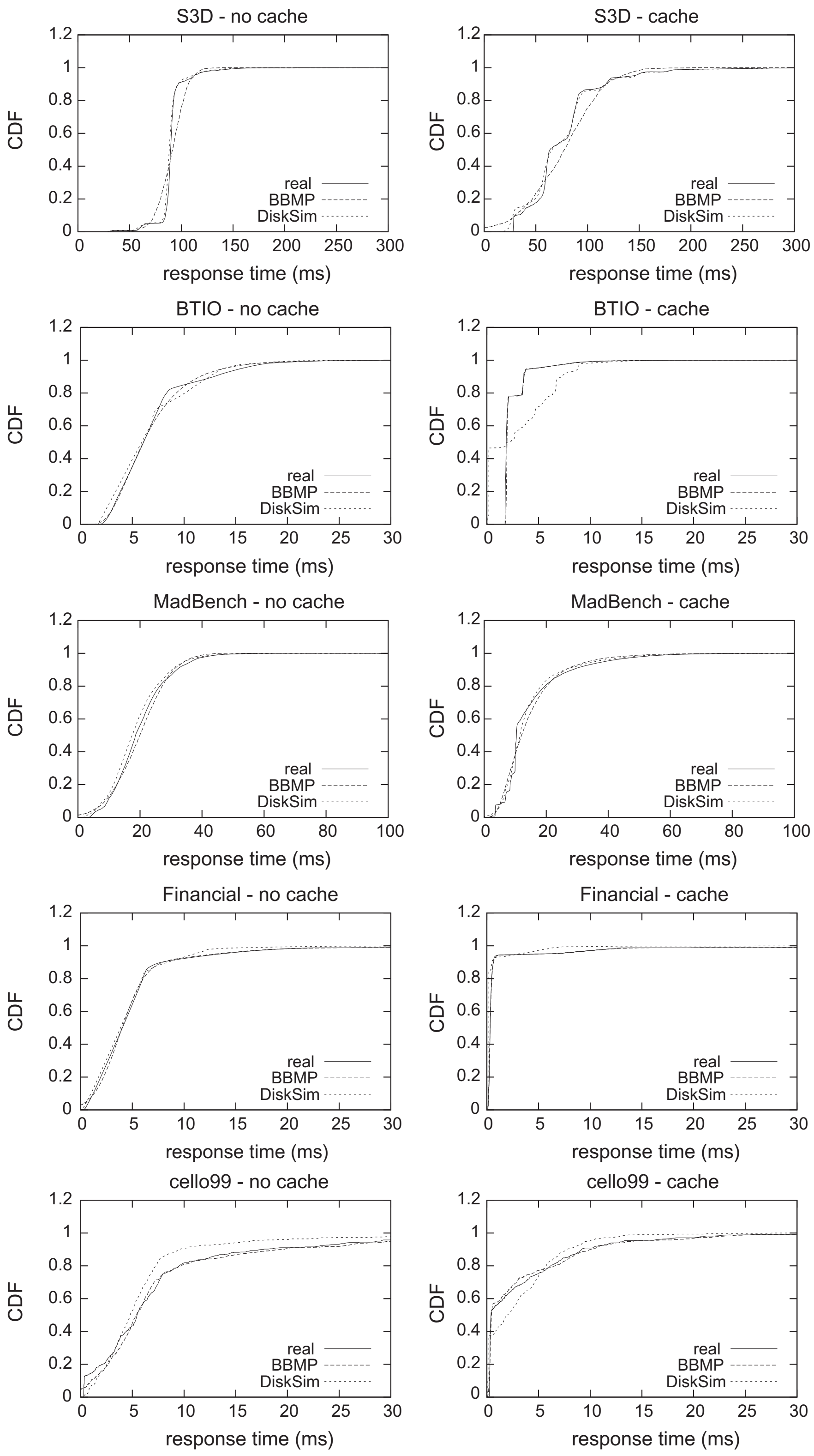

**Fig. 3.** Response times CDFs superimposition from a Seagate Cheetah 10K.7 disk and several block traces. The left column shows result when cache effects are removed from the device. In the right, cache effects are presented.

**Table 4**
Comparison of demerits in relative terms for BBMP and DiskSim.

| | Cache | | No cache | |
|---|---|---|---|---|
| | DiskSim (%) | BBMP (%) | DiskSim (%) | BBMP (%) |
| S3D | 4.7 | 20.0 | 5.6 | 8.0 |
| BTIO | 94.0 | 4.9 | 13.0 | 8.3 |
| MadBench | 8.7 | 14.3 | 6.7 | 8.4 |
| Financial | 286.0 | 10.2 | 54.0 | 7.4 |
| Cello99 | 51.0 | 12.0 | 45.0 | 9.0 |

in the same range. The highest demerits turn out to be in the DiskSim case, for Financial (54%) and Cello99 (45%) traces. In both cases, DiskSim did not take into account idle time effects on response times. Other traces, like S3D and MadBench produced higher demerits in BBMP (8% and 8.40%, respectively) than in DiskSim (5.6% and 6.70%, respectively) due to errors in prediction.

As we show in the right column of Fig. 3, demerits are slightly better for some traces in the BBMP approach than in DiskSim when the cache is activated. However, in the DiskSim approach demerits are out of range for some traces (BTIO, Financial, and Cello99). This is because DIXtrac did not perfectly identify caching parameters or policies for the evaluated disk. However, other traces, like S3D and MadBench, produced higher demerits in the BBMP trace (20% and 14.30%, respectively) than in DiskSim (4.70% and 8.70%, respectively). That is because of errors produced when predicting. In the DiskSim cases, caching did not affect response times for S3D and MadBench because both traces are very bursty and also their requests sizes are big. This makes queuing times to outshine possible hits on disk cache, and also demerits to be lower.

Finally, Fig. 4 shows how much BBMP is faster in comparison with DiskSim in terms of speedup. Speedups are compared when the disk buffer cache is activated and it is not. Execution times for S3D are almost 600 times faster than for DiskSim. This is primarily due to the number of blocks that are demanded in the same request, which are higher than in other traces. The other traces are still faster in the BBMP executions (two orders of magnitude), which translates into the fact that our model is highly efficient.

### 7.3. Modeling non-trained traces

In this subsection we present the evaluation of two non-trained traces by using one of the previous constructed black-box simulation models. A non-trained trace is a trace that has not been previously used in the simulation model construction. We have chosen *WebUsers* and *Webmail* from the SNIA IOTTA repository [31]. *WebUsers* represents a web server hosting faculty, which includes both staff and graduate students web sites. In turn, *WebMail* represents the web interface to the department web server.

Fig. 5 shows the CDF results of executing both traces on our variate generator (simulated) and the Seagate Cheetah 10K.7 disk (real). For both of them, predictions are made from the model constructed for the Financial trace, which is similar in characteristics of size, queuing times, and sequentiality. Demerits of those distributions for cache and non-cache approaches are shown in Table 5.

For both traces, queuing times are more common than in Financial, and real response times turn out to be longer than the ones generated from the based-on-Financial-model. That happens because in the model, if a current request has been long, its length does not affect subsequent requests' response times. This is so because generation of response times are indepen-

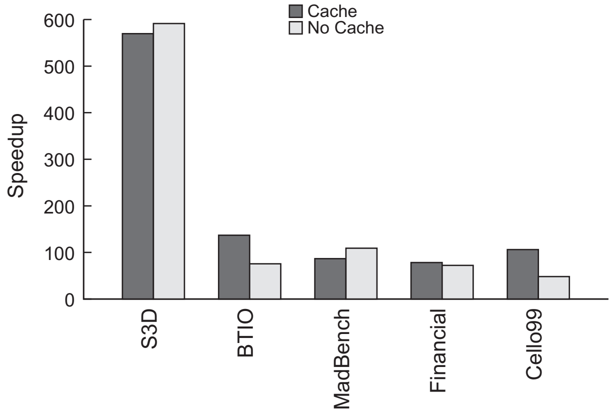

**Fig. 4.** Speed-up comparative of S3D trace. We plot how much BBMP is faster than DiskSim in terms of simulation time.

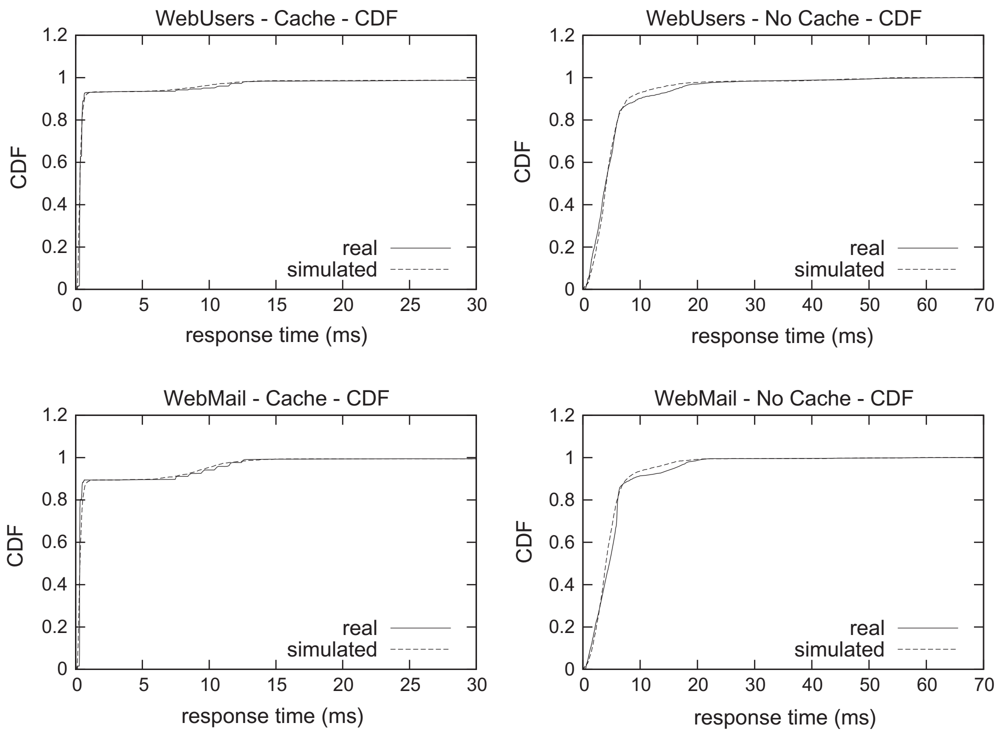

**Fig. 5.** Response times CDFs superimposition for non-trained traces.

dent, and if a current response time was long, the subsequent request does not have to be so. This makes certain response times may be shorter than real queuing times, as it happens. The use of caching reduces queuing effects, and as it is shown in the figures, real and simulated results present better similarities.

## *7.4. Simulation models based on synthetic traces*

In this subsection, we present and evaluate a model based on a synthetic trace. Among different available options, we chose the SPC Benchmark v1.10.1 [32] defined by the Storage Performance Council [33]. As discussed in [13], traces generated by the benchmark simulate a real environment, commonly used in business applications such as OLTP, database, and mail server systems.

In order to evaluate the model, we replay several real traces on it and plot their response times CDFs superimposition. We chose two of the bunch of the previously described traces. We selected the ones that were more similar in the range of request sizes to SPC-1 workloads. Those were Financial [36] and Cello99 [15]. We also used another trace, namely WebSearch [36], which belongs to a famous search engine server, which executes about 4 million read requests over six disks during 4 h. For simplicity, we did not deactivated disk buffers. We constructed the model and predicted from it, by using read-ahead and immediate reporting by default.

Fig. 6 depicts CDFs comparisons of four workloads by using the simulation model constructed for a synthetic trace. Demerits of those distributions are shown in Table 6. For the four workloads, predictions are made from a model based on a SPC1 trace. When predicting, queuing times are calculated on the fly, by checking if the previous requests have finished.

SPC1 is the same trace as the one used to build the simulation model. However, when constructing the model, queuing times were removed from the response time samples. For all traces, both the modeled and real distributions follow the same trend. However, prediction errors may lead to shorter queuing times, as in SPC1 trace, or larger queuing times, as in WebSearch, Finnancial, and Cello99.

**Table 5**
Demerits in relative terms for non-trained traces.

| Trace | Cache (%) | No cache (%) |
|---|---|---|
| WebUsers | 39 | 21 |
| Webmail | 25 | 21 |

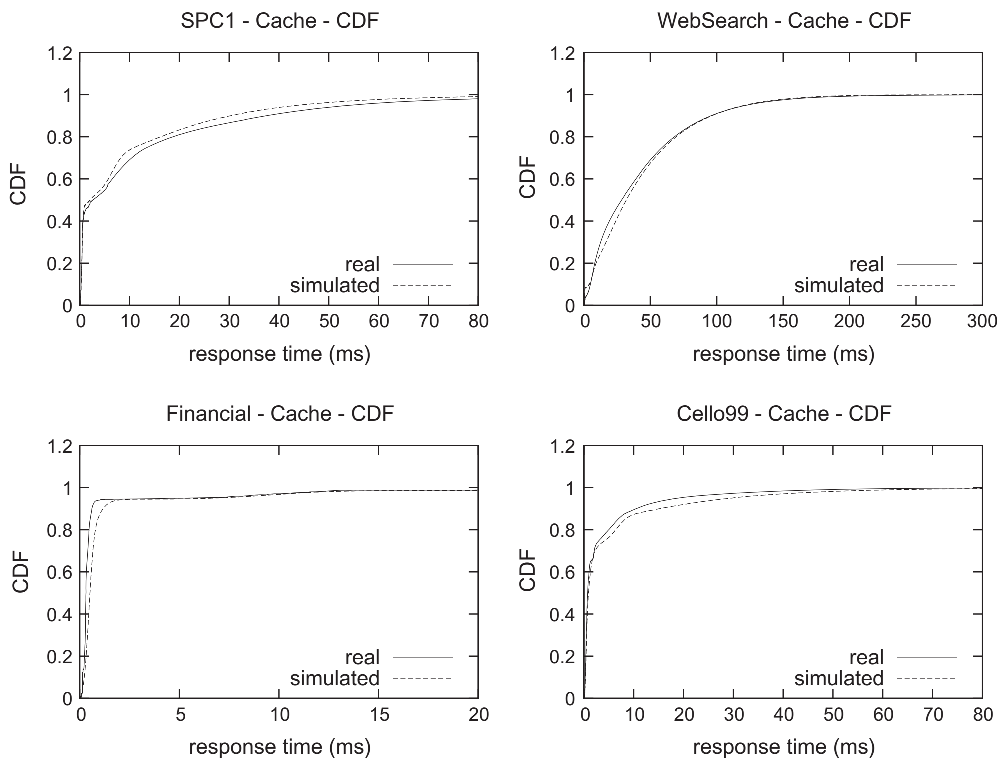

**Fig. 6.** Response times CDFs superimposition from a Seagate Cheetah 10K.7 disk and a BBMP based on a synthetic trace. Non-trained real traces are executed. Caching is active.

**Table 6**
Demerits in relative terms for our black-box model approach (BBMP). Non-trained traces are executed.

| Workload | Demerit (%) |
|---|---|
| SPC1 | 59.6 |
| WebSearch | 28.9 |
| Financial | 48.3 |
| Cello99 | 87.0 |

**Table 7**
Demerits, values of energy, and energy errors, in relative terms, for our black-box model approach (BBMP) and DiskSim.

| Trace | Demerit (%) | Energy DiskSim (J) | Energy BBMP (J) | Prediction error (%) |
|---|---|---|---|---|
| BTIO | 7.25 | 33290.29 | 33306.20 | 0.04 |
| MadBench | 11.63 | 68933.28 | 68958.55 | 0.04 |
| S3D | 7.36 | 10629.27 | 10680.77 | 0.48 |
| Financial | 15.58 | 522704.00 | 524189.60 | 0.28 |

In the specific case of Financial, although queuing times are not very common, response times from the model keep being bigger than from the real trace. This is because in Financial mean request sizes are smaller than in SPC1, which covers a wide range of sizes, generally bigger than Financial.

## *7.5. Modeling storage energy consumption*

In this subsection, we show how BBMP can be used to measure energy consumption in hard disk drives. We present results of models constructed for the previous traces. The different energy-aware states of the simulated disk are summarized in Table 8.

**Table 8**
Seagate Cheetah 15K.5 power specifications.

| State | Consumption (W) |
|---|---|
| Idle | 12 |
| Active | 17 |
| Standby | 2.6 |

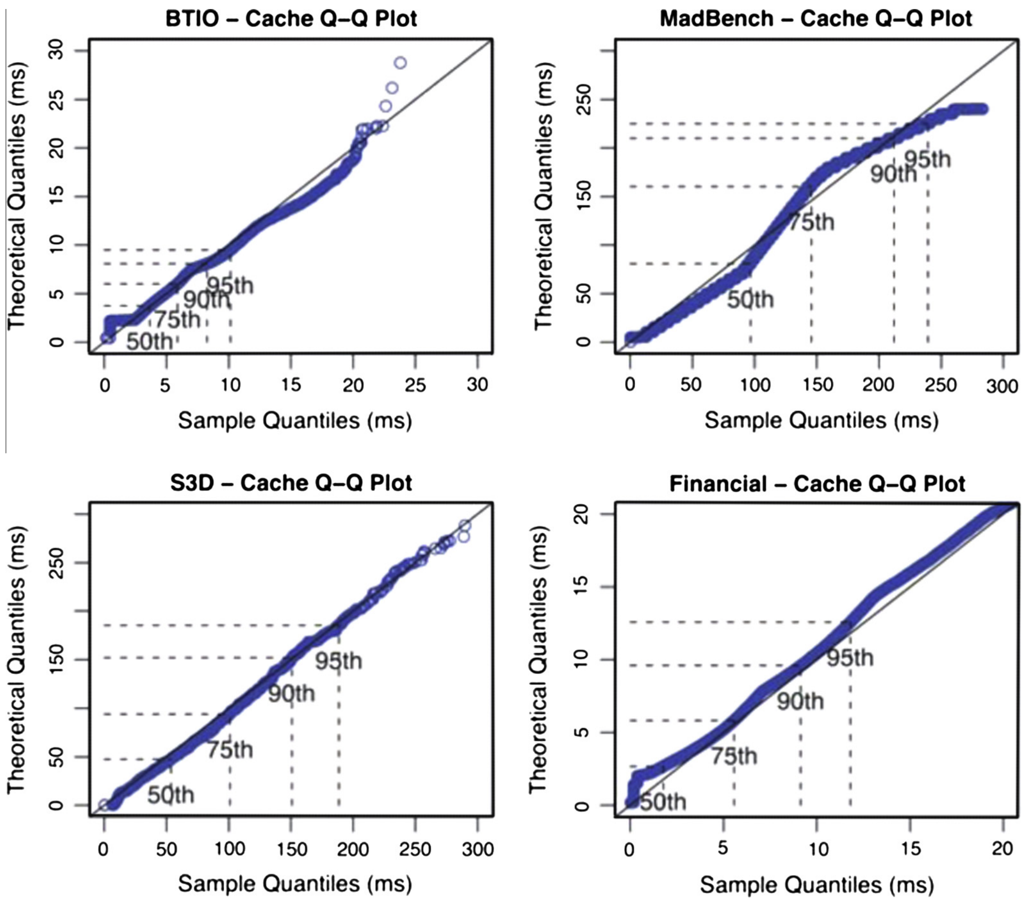

**Fig. 7.** Q–Q plots for simulated response times of a Seagate Cheetah 15 K.5 disk and four different workloads.

**Table 9**
Breakdown of black-box model generation time.

| Trace | Recording, $t_r$ (min) | Detection, $t_d$ (s) | Identification, $t_i$ (s) |
|---|---|---|---|
| S3D | 15.0 | 0.79 | – |
| BTIO | 45.0 | 6.40 | 22.40 |
| MadBench | 95.4 | 0.34 | – |
| Financial | 720.0 | 13.44 | 113.92 |
| Cello99 | 1440.0 | 40.00 | 240.00 |

Fig. 7 shows Q–Q plots for BTIO, MadBench, S3D, and Financial. The four ones allow to judge the goodness-of-fit of each particular model to response times obtained by using DiskSim. The energy consumption model described in Section 6 was applied to both model categories. As we show in Table 7, demerits are not very high due to values of energy are very similar, both in DiskSim and in BBMP. This lead us to conclude that BBMP can also be used for predicting energy consumption of I/O intensive applications.

### 7.6. Generation time of black-box model

In this subsection we analyze the time needed to generate a black-box statistical model. That time consists of three terms (see Fig. 1):

- $t_r$: Time required to record requests response times from the trace (step 2).
- $t_d$: Time required to detect possible distributions from the response times sample (step 3).
- $t_i$: Time required to identify possible causes for the detected distributions (step 3).

Table 9 shows times $t_r, t_d$, and $t_i$ for each trace used in this study.

Used traces are long enough to ensure that the process is in a steady state, so that using longer traces would not add any additional value. Distribution detection time ($t_d$) takes in our traces, at the worst case, 1 min. Actual time depends on the response times sample shape and trace duration.

Identification time ($t_i$) may be longer, although it is under acceptable limits. This time is highly dependent on the number of detected distributions, and also on the trace length. The parameters used to generate response times from a certain distribution are cache size, sequentiality, queueing effects, and idle times. A genetic algorithm identifies those parameters by using the sample, and previously detected distributions.

For S3D and MadBench traces we do not provide identification time as there is no need to perform any identification. The reason behind that, is that for those traces, a single distribution can be used, and thus all response times can be generated from that distribution.

## 8. Conclusion

Providing efficient and power-aware storage solutions is currently an active research field. There is a growing number of optimizations and solutions which deal with the problem of achieving energy efficiency and performance, however there is a lack of tools which allow evaluate then. In this paper we have described a methodology used to build fast and accurate black-box simulation models for hard disk drives. The presented solution is based on probability distributions and can be used for both synthetic and real traces. The methodology includes a generic measuring service time tool.

Our approach is based on a multiple real trace repository and may be accurate when the traces to predict have similar characteristics as one of the traces, for which the model has been constructed. The model can also be based on a single synthetic trace. This trace is supposed to cover a wide range of characteristics from other traces, including several access patterns, common in real disks. Additionally, this approach allows to calculate queuing times on the fly on prediction stage, making it more versatile.

Our solution has been compared with DiskSim in terms of accuracy and speed-up, showing that BBMP is two orders of magnitude faster than DiskSim for most of the experiments. It has also been compared with DiskSim in terms of energy consumption estimations, and we concluded that it keeps being accurate in that respect.

## Acknowledgements

This work has been partially funded by the Spanish Ministry of Science and Technology under the Grant TIN2010-16497 *Técnicas Escalables de Entrada/Salida en Entornos Distribuidos y de Computación de Altas Prestaciones* (Input/Output Scalable Techniques in Distributed and High Performance Computing Environments).

## References

[1] M. Allalouf, Y. Arbitman, M. Factor, R. Kat, K. Meth, D. Naor, Storage modeling for power estimation, in: Proceedings of SYSTOR 2009: The Israeli Experimental Systems Conference, 2009.
[2] E. Anderson, Simple Table-based Modeling of Storage Devices, Technical Report HPL-SSP-2001-4, HP Laboratories, 2001.
[3] J.S. Bucy, J. Schindler, G.R. Schlosser, Steven W. Ganger, Contributors, The DiskSim Simulation Environment Version 4. 0 Reference Manual, Technical Report CMU-PDL-08-101, Carnegie Mellon University Parallel Data Lab, 2008.
[4] Carnegie Mellon University Parallel Data Lab, DIXtrac: Automated Disk Drive Characterization, 2008. <http://www.pdl.cmu.edu/Dixtrac/index.shtml> (last visited January 2012).
[5] Carnegie Mellon University Parallel Data Lab, The DiskSim Simulation Environment (V4.0), 2008. <http://www.pdl.cmu.edu/DiskSim> (last visited on January 2012).
[6] A. Ching, A. Choudhary, W.K. Liao, R. Ross, W. Gropp, Noncontiguous I/O through PVFS, in: Proceedings of the IEEE International Conference on Cluster Computing, CLUSTER'02, IEEE Computer Society, Washington, DC, USA, 2002, p. 405.
[7] A. Chouder, S. Silvestre, N. Sadaoui, L. Rahmani, Modeling and simulation of a grid connected PV system based on the evaluation of main PV module parameters, Simulation Modelling Practice and Theory 20 (2012) 46–58.
[8] CompuGreen LLC. The Green 500 List, 2007. <http://www.green500.org> (last visited June 2012).
[9] Y. Deng, J. Zhou, X. Meng, Deconstructing on-board disk cache by using block-level real traces, Simulation Modelling Practice and Theory 20 (2012) 33–45.
[10] X. Fan, W.D. Weber, L.A. Barroso, Power provisioning for a warehouse-sized computer, in: Proceedings of the 34th Annual International Symposium on Computer Architecture, ISCA'07, ACM, New York, NY, USA, 2007, pp. 13–23.
[11] J.D. Garcia, L. Prada, J. Fernandez, A. Nunez, J. Carretero, Using black-box modeling techniques for modern disk drives service time simulation, in: Proceedings of the 41st Annual Simulation Symposium, ANSS'08, IEEE Computer Society, Washington, DC, USA, 2008, pp. 139–145.

[12] M. Geimer, F. Wolf, B. Wylie, E. Ábrahám, D. Becker, B. Mohr, The Scalasca performance toolset architecture, Concurrency and Computation: Practice and Experience – Scalable Tools for High-End Computing 22 (2010) 702–719.
[13] B.S. Gill, D.S. Modha, SARC: sequential prefetching in adaptive replacement cache, in: Proceedings of the Annual Conference on USENIX Annual Technical Conference, ATEC '05, USENIX Association, Berkeley, CA, USA, 2005, p. 33.
[14] U. Hoelzle, L.A. Barroso, The datacenter as a computer: an introduction to the design of warehouse-scale machines, Synthesis Lectures on Computer Architecture, Morgan and Claypool Publishers, 2009.
[15] HP Labs, The Cello99 Trace, 1999. <http://tesla.hpl.hp.com/opensource/cello99> (last visited June 2012).
[16] R.T. Kaushik, M. Bhandarkar, GreenHDFS: towards an energy-conserving, storage-efficient, hybrid Hadoop compute cluster, in: Proceedings of the 2010 International Conference on Power Aware Computing and Systems, HotPower'10, USENIX Association, Berkeley, CA, USA, 2010, pp. 1–9.
[17] M.Y. Kim, A.N. Tantawi, Asynchronous disk interleaving: approximating access delays, IEEE Transactions on Computers 40 (1991) 801–810.
[18] M. Knobloch, B. Mohr, T. Minartz, Determine energy-saving potential in wait-states of large-scale parallel programs, Computer Science – Research and Development 27 (2012) 255–263.
[19] Lawrence Berkeley National Laboratory, MADbench2 Benchmark, 2008. <http://crd.lbl.gov/borrill/MADbench2> (last visited 2011).
[20] S. Li, H.H. Huang, Black-box performance modeling for solid-state drives, in: Proceedings of the 2010 IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, MASCOTS '10, IEEE Computer Society, Washington, DC, USA, 2010, pp. 391–393.
[21] M.P. Mesnier, M. Wachs, R.R. Sambasivan, A.X. Zheng, G.R. Ganger, Modeling the relative fitness of storage, SIGMETRICS Performance Evaluation Review 35 (2007) 37–48.
[22] D. Narayanan, A. Donnelly, A. Rowstron, Write off-loading: practical power management for enterprise storage, ACM Transactions on Storage 4 (2008) 10:1–10:23.
[23] NASA Advanced Supercomputing Division, NAS BTIO Benchmark, 2011. <http://www.nas.nasa.gov/Resources/Software/npb.html> (last visited 2011).
[24] M. Poess, R.O. Nambiar, Energy cost, the key challenge of today's data centers: a power consumption analysis of TPC-C results, Proceedings of the VLDB Endowment 1 (2008) 1229–1240.
[25] L. Prada, A. Calderon, J. Garcia, J.D. Garcia, J. Carretero, A black box model for storage devices based on probability distributions, in: Proceedings of the 10th IEEE International Symposium on Parallel and Distributed Processing with Applications, ISPA, IEEE Computer Society, Los Alamitos, CA, USA, 2012, pp. 355–362.
[26] C. Ruemmler, J. Wilkes, An introduction to disk drive modeling, IEEE Computer 27 (1994) 17–28.
[27] Sandia National Laboratories, S3d I/O Traces, 2009. <http://www.cs.sandia.gov/Scalable_IO/SNL_Trace_Data> (last visited June 2012).
[28] J. Schindler, G.R. Ganger, Automated disk drive characterization, in: Proceedings of the 2000 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, SIGMETRICS'00, ACM, New York, NY, USA, 2000, pp. 112–113.
[29] J. Schindler, J.L. Griffin, C.R. Lumb, G.R. Ganger, Track-aligned extents: matching access patterns to disk drive characteristics, in: Proceedings of the 1st USENIX Conference on File and Storage Technologies, FAST'02, USENIX Association, Berkeley, CA, USA, 2002, pp. 259–274.
[30] E. Shriver, A. Merchant, J. Wilkes, An analytic behavior model for disk drives with readahead caches and request reordering, ACM SIGMETRICS Performance Evaluation Review 26 (1998) 182–191.
[31] SNIA, Snia IOTTA Trace Repository, 2011. <http://iotta.snia.org/traces> (last visited June 2012).
[32] Storage Performance Council, SPC Benchmark-1 (SPC-1). Official Specification, 2006. V1.10.1.
[33] Storage Performance Council, SPC Benchmarks, 2006. <http://www.storageperformance.org> (last visited June 2012).
[34] E. Thereska, A. Donnelly, D. Narayanan, Sierra: practical power-proportionality for data center storage, in: Proceedings of the Sixth Conference on Computer systems, EuroSys'11, ACM, New York, NY, USA, 2011, pp. 169–182.
[35] P. Triantafillou, S. Christodoulakis, C.A. Georgiadis, A comprehensive analytical performance model for disk devices under random workloads, IEEE Transactions on Knowledge and Data Engineering 14 (2002) 140–155.
[36] University of Massachusetts Amherst Laboratory for Advanced System Software, UMass Trace Repository, 2007. <http://traces.cs.umass.edu> (last visited June 2012).
[37] M. Uysal, G.A. Alvarez, A. Merchant, A. modular, analytical throughput model for modern disk arrays, in: Proceedings of the 9th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, MASCOTS'01, IEEE Computer Society, Washington, DC, USA, 2001, pp. 183–192.
[38] A. Varga, OMNeT++ Community Site, 2012. <http://www.omnetpp.org> (last visited January 2012).
[39] M. Wang, K. Au, A. Ailamaki, A. Brockwell, C. Faloutsos, G.R. Ganger, Storage device performance prediction with CART models, in: Proceedings of the IEEE Computer Society's 12th Annual International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems, MASCOTS'04, IEEE Computer Society, Washington, DC, USA, 2004, pp. 588–595.
[40] B.L. Worthington, G.R. Ganger, Y.N. Patt, J. Wilkes, On-line extraction of SCSI disk drive parameters, ACM SIGMETRICS Performance Evaluation Review 23 (1995) 146–156.
[41] WU Wien Institute for Statistics and Mathematics, The R Project for Statistical Computing, 2012. <http://www.r-project.org/> (last visited January 2012).
[42] P.S. Yu, A. Merchant, Analytic modeling and comparisons of striping strategies for replicated disk arrays, IEEE Transactions on Computers 44 (1995) 419–433.
[43] J. Zedlewski, S. Sobti, N. Garg, F. Zheng, A. Krishnamurthy, R. Wang, Modeling hard-disk power consumption, in: Proceedings of the 2nd USENIX Conference on File and Storage Technologies, FAST'03, USENIX Association, Berkeley, CA, USA, 2003, pp. 217–230.
[44] Q. Zhu, Y. Zhou, Power-aware storage cache management, IEEE Transactions on Computers 54 (2005) 587–602.