Int j simul model 10 (2011) 1, 27-37
ISSN 1726-4529 Original scientific paper

DISK ARRAY SIMULATION MODEL DEVELOPMENT

Vickovic, L.; Celar, S. & Mudnic, E.
University of Split, Faculty of Electrical Engineering, Mechanical Engineering and Naval Architecture, R. Boskovica 32, 21000 Split, Croatia
E-Mail: linda.vickovic@cern.hr, stipe.celar@fesb.hr, eugen.mudnic@fesb.hr

Abstract
This paper presents a detailed development process of dynamic, discrete event simulation model for disk array. It combines a hierarchical decomposition with the "bottom up" approach. This way, at the beginning, the focus is set on the elementary storage component – a single disk drive. Once when functional simulation model for disk drive has been achieved it is used as a basic storage element for disk array model development. Further on, it is explored how to simulate different interfaces inside disk array, toward underlying disks. The difference in throughput produced by developed model and measurements is from 1.5-3.16 % for writing and from 2.5-2.8 % for reading, depending on interface type. However, such results are limited on workload imposed by the requirements of the ALICE transient storage system, or more precisely, sequential storing and reading of large data files.
(Received in August 2010, accepted in December 2010. This paper was with the authors 2 months for 1 revision.)

Key Words: Discrete-Event Simulation, Disk Array Simulation, Disk Simulation, Disk Cache Model

1. INTRODUCTION

Many modern, data intensive applications, like scientific data logging in high-energy physics, are aggressively pushing demands for data storage. Consequently, it becomes an interesting issue to analyze and improve performance of storage media. In order to perform that, several approaches are possible. One of them, described in [1], is to perform a set of measurements on the existing storage system. It is advisable, if storage system really exists, but in many cases it can be too expensive in the terms of required hardware and measurements software and even too disruptive toward system itself. The other approach is to develop a model of the system and study it, as a surrogate for system optimization. Although model development is time-consuming process, once when it is developed optimal system configuration can be chosen simply by changing system components, their organization or input parameters without any additional expenses.

For storage systems, especially complex, data intensive storage systems, model development primarily means building up the model of its main storage element - a disk array or Redundant Array of Independent/Inexpensive Disks – RAID.

Its analytical models have been under study for more than decade. The earliest models discussed in [2], [3] approximated access latencies. These were followed by various contributions, each focused on a different aspect. For example, [4] considered the effect of the execution mode on performance, [5] investigated fault tolerance, [6] analyzed the effects of caching. Even so, the analytical model of arrays are still under study like in [7] where the performance of a multi-level array architecture is described, or in [8] where model is used for an analyses of disk-oriented rebuild processing in RAID5 arrays.

On contrary, simulation models for arrays are very rare. Till now, only one functional simulator has been developed - a RAIDframe [9], aimed for redundant array software development. Other ones simulate just a part of the array under study like in [10] where

simulator is used to study performance of the various striping methods and buffer caching scheme, or in [7] where it is used to validate the analytical model results.

If a hierarchical decomposition is combined with the "bottom up" approach in the array model development, at the beginning the focus is set on the elementary storage component - a single disk drive.

Although many analytical models of disk drive have been developed and presented in literature, according to [11] because of their nonlinear, state-dependent behaviour, they cannot be modelled analytically with any accuracy so most work in this area uses simulation. Those simulators can be used for various purposes like: generation of disk drive service time distribution described in [12] or for investigation of object-based storage devices [13]. Also, two functional simulation environments for a disk sub-system have been developed: DiskSim [14] and Pantheon [15]. Both of them include disk modules which have been carefully validated against real disks.

This paper presents a development process for discrete event (DE) simulation model of a single disk drive and disk array, created in Ptolemy [16] an environment for simulation and prototyping of heterogeneous systems developed at Berkeley, the University of California.

Both models, verified and validated should be used as an elementary storage module for complex storage system simulation with the aim to explore a performance of different storage media for the requirements of the ALICE experiment at CERN. The ALICE, one of the four experiments at the Large Hadron Collider (LHC) at CERN is characterized by the most extensive data flow toward data storage. As such it set up very high demands on storage system performances, and a model can be used to test and optimize possible storage architectures.

The remainder of the document is organized as follows. The next section describes workload specification imposed by the ALICE experiment. The system architecture for test measurements, used for model verification and validation is given in section 3. Section 4 contains model development process for disk drive. While Section 5 describes model development process for disk array. Finally, conclusions and plans for future development are given in section 6.

2. WORKLOAD SPECIFICATIONS

The ability of any model to identify optimal configuration parameters does not just depend on the accuracy of the model. It also depends on the representativeness of the workload [17]. The workload definition is a bit more complex, and even harder to be properly described than a system itself because it should exactly describe the process of the request arrival to the system components, for given simulation.

For the model presented in this paper the workload is defined by the requirements of the ALICE Transient Data Storage (TDS) system.

The data flow in the ALICE, presented in Fig. 1 starts from detectors, continues through Data Acquisition System (DAQ) and High Level Trigger and finishes with storage in Mass Storage System (MSS).

The MSS is organized on two levels. The first one is responsible for the real-time data storage and is called the Transient Data Storage (TDS). It consists of arrays, and ensures the required input throughput from DAQ, but has small capacity. The second level, the so-called Permanent Data Storage (PDS) cannot assure required input bandwidth but it provides enough storage capacity for all data acquired during the lifetime of the experiment [18]. Such requirements imposed relatively simple workload model for the simulation: a sequential storage or reading of large data files, where all files have same size.

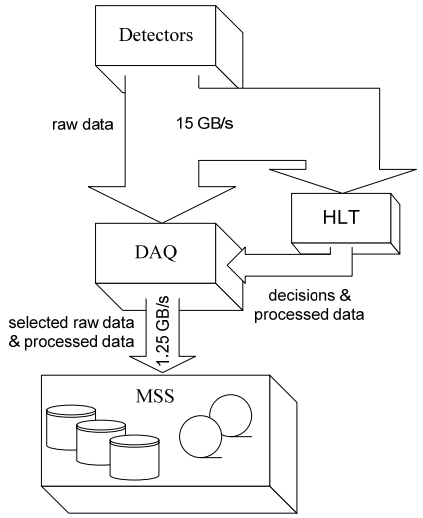

Figure 1: The ALICE data flow.

3. SYSTEM ARCHITECTURE FOR MODEL VERIFICATION

System architecture for test measurements was organized like it is presented in Fig. 2. It contained following storage media:

- Maxtor MaXLine Plus II disk;
- two identical Infortrend arrays (IFT-7250F) each with 5 Maxtor MaXLine Plus II disks;
- two identical DotHill arrays (SANnet II 200 FC Array) each with 5 Cheetah X15 36LP FC disks.

Each storage element was dedicated just for tests without any operating system installed. As a result, the whole data transfer from and to the storage media was only produced by the performance measurements program installed on server. The program was a standalone client running in the background and it constantly wrote or read fixed length records (8, 32, 128, 512, 2048, 8192, 32768, 131072 kB) filled with random data until the predefined output file size (2 GB) was reached. In the same time, it monitored the exact duration of each transfer. Also, to get as precise results as possible each measurement for different file - record combination was repeated 15 times.

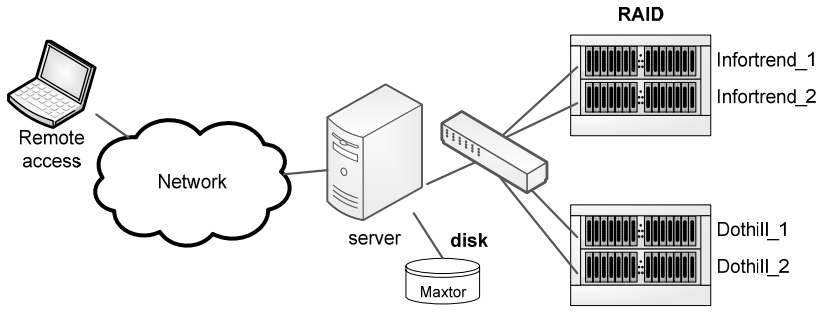

Figure 2: System architecture for test measurements.

The idea behind measurements was to use produced results for model validation, but even same measurements done on same array at different time, did not always produced the same results. Like it is shown in Fig. 3, writing to the disk array, is not a constant process. Actually, results for different measurements on DotHill array were very similar, but individual measurements for Infortrend array differed considerably. Because of those differences, the average value for each array was counted and used as reference value for validation.

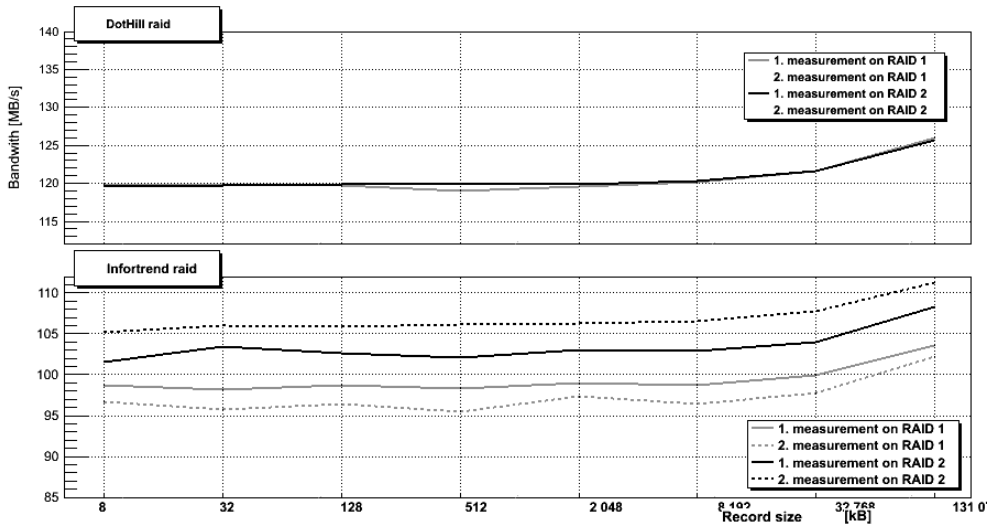

Figure 3: Disk array writing performance for 2GB files.

4. DISK DRIVE SIMULATION MODEL

Computer systems are usually modelled as discrete event models because their states are generally discrete, like the number of jobs in a queue, available buffer size, and so on. Therefore, the disk drive model simulation elaborated in this paper is modelled as discrete event. Its structure in Ptolemy, in detail described in [19], is shown in Fig. 4. The model is built from two discrete event components: server and disk module. Server module is modelled like a simple delay server. It produces sequential write or read requests toward Disk module and presents throughput results when a given task is accomplished. Actually, it simulates the behaviour of the performance measurements program.

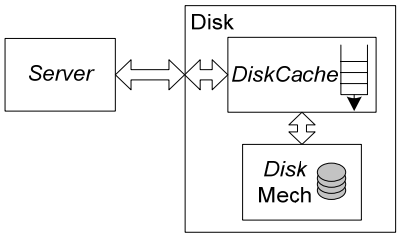

Figure 4: The general scheme of a disk drive model.

Disk module simulates the behaviour of disk drive and it comprises two modules *diskCache* and *diskMech*. The first of them represents the behaviour of disk cache and it was initially modelled like simple First In First Out (FIFO) queue. *diskMech* module models the behaviour of disk drive mechanical components.

Data flow in this model starts from *Server* module which sends request to *Disk*. If it is a write request then it is stored in *diskCache* and the time required for storage is defined by the disk interface speed. When the request is stored, the amount of free space on *diskCache* is decreased and it is pushed on cache queue. From that point *diskCache* leads two independent communications. The former is toward *Server*, which sends another request and their communication proceeds until all requests are sent, or until there is no more free space in *diskCache*. The latter communication is between *diskCache* and *diskMech*. According to FIFO principle, one by one request from cache is sent to *diskMech*. Once when the record is actually stored, the amount of free space in *diskCache* is increased. The communication proceeds until all records are stored.

If *Server* sends a read request they are just forwarded without delay to *diskCache*, which forwards them further to *diskMech*. This component counts the time required to read the record from disk drive platters and media transfer time and sends it to *diskCache* with counted delay. From the *diskCache*, the record is forward to the *Server*. This model does not apply any read ahead strategy because it is going to be used for sequential reading of very large files.

To simplify this approach following assumptions are taken:

- the simulation starts, on the empty disk, from the outer most track with the maximal internal transfer rate, and it is decreased linearly toward minimal value, each time the track on disk is changed;
- all tracks have the same size, an average track size provided by manufacturer.

Also, the disk drive is presented like a two dimensional array where the number of rows is equal to the number of cylinders, and the number of columns to the number of disk surfaces. In this case whenever it is necessary to switch the disk head (change a column) or switch to another cylinder (change a raw) an additional delay is counted. Those delays, produced by switching the disk head or changing to the subsequent disk cylinder, are given in disk's technical specifications. Also, like in real measurements all simulated tests were repeated 15 times.

With these general guidelines, model was developed in several subsequent phases, where the results from each phase were compared with the measurements performed on a real system, until satisfactory results are achieved. The more detailed preview of disk drive model development is presented in [20].

4.1. Model development for write workload

The initial model of a hard disk drive was very straightforward. Cache was modelled as simple FIFO queue, while the behaviour of its mechanical components was completely based on the equation:

$$average\_service\_time = \frac{f}{2} + \frac{r}{2} + media\_transfer\_time + \frac{l}{i} + h \qquad (1)$$

where: f is full-stroke seek time [s], r is full-disk rotation time [s], l is mean request size [B], t is average size of disk tracks [B], i [B/s] is interface transfer speed, h a controller overhead. media_transfer_time parameter from (1) was not counted analytically but from the linear change of the internal transfer rate. According to the data sheets given by manufacturers, the disk internal transfer rate is maximal at the outer most track and minimal at the inner most track. So, assumed that disk is empty, the storage can start from the outer most track with maximal speed and then decreased by some Δspeed value, each time the track on disk is changed. Minimal value is reached at the inner most track when the disk is full.

Not only that average difference between the measured and simulated results for this model was around 17 %, but it also showed descending trend for large record sizes (32768, 131072 kB) that was not a case for measurements.

As the described model of disk drive had not produced satisfactory results, a next step was to introduce some “random” behaviour in the simulation. The most common parameters with random behaviour in disk drive modelling are: workload generation, modelling of disk seek and rotation time and service time delay simulation ([11], [21], [22]). Although stochastic elements cannot be used for workload generation, because all real measurements were done with just one sequential data stream, they can be used for service time delay in two components *Server* and *diskCache*. For service time delay generation three general distributions: normal, uniform and exponential were individually incorporated in the model and tested.

Random service time delay implied in *Server* module referred delay between successive requests sent to the disk drive, but it proved to be immune on service time delay change. Obviously, the delay produced by storing a file is much bigger than a delay between successive requests and it does not influence the system behaviour.

However, *diskCache* module was sensitive to the changes of service time delay and because of that some random service time delay from the interval of standard DRAM cache delay was presented to its outputs. The resulting model's behaviour is presented in Fig. 5. It can be easily concluded that uniform distribution should be avoided, while the normal and exponential distribution showed similar results. In average the difference between measured and simulated results for normal and exponential distribution was 2.22 % (0.98 % without 8 kB records) and 2.01 % (1.03 % without 8 kB records), respectively. As the largest record sizes are most likely to be used in practice, discrepancy for 8 kB records can be neglected and the normal distribution is chosen.

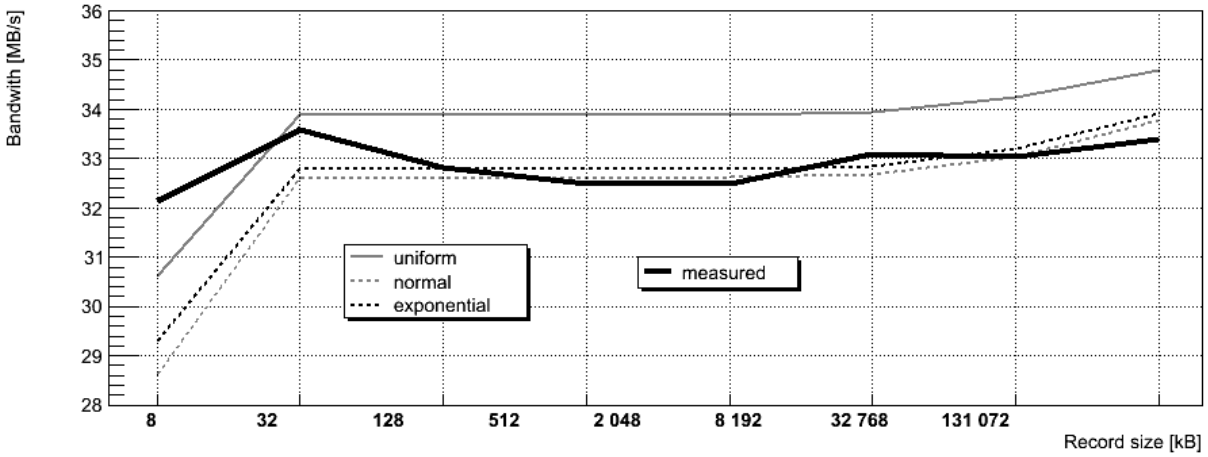

Figure 5: Simulation output for a different random service time delays in *diskCache* module.

4.2. Model development for read workload

With the functional model for write workload, the next step in the simulation development was to accommodate that model for the read workload. Although most of the functionalities from the write workload model can be used, different direction of data flow required some changes.

Once when functional version was achieved, it was firstly tested with the stochastic parameters that produced the best results for write workload. Simulation output, shown in Fig. 6, has the same increasing trend like one produced by write workload, but difference between measured and simulated results (18.02 % difference for 8 kB record size and 7.7 % difference in average for the other record sizes) pointed out that it was not possible to describe read and write cache behaviour with the same stochastic parameters.

Such outcome was expected because of two reasons, one related with the disk drive mechanism, and other related with the disk cache. For writing, the disk drive mechanism should position read/write head exactly to avoid damage of neighbouring bytes, while for reading such precision is not necessary and because of that reading from disk is a bit faster than writing. Besides, the cache model applied in this simulation is very simple and it does not take into account any throughput increase produced by read ahead cache policy. With the intention to keep a model as simple as possible those two factors are presented by some smaller delay generated by *diskCache* module. The resulting model's performances for normal, exponential and uniform distribution are also presented in Fig. 6.

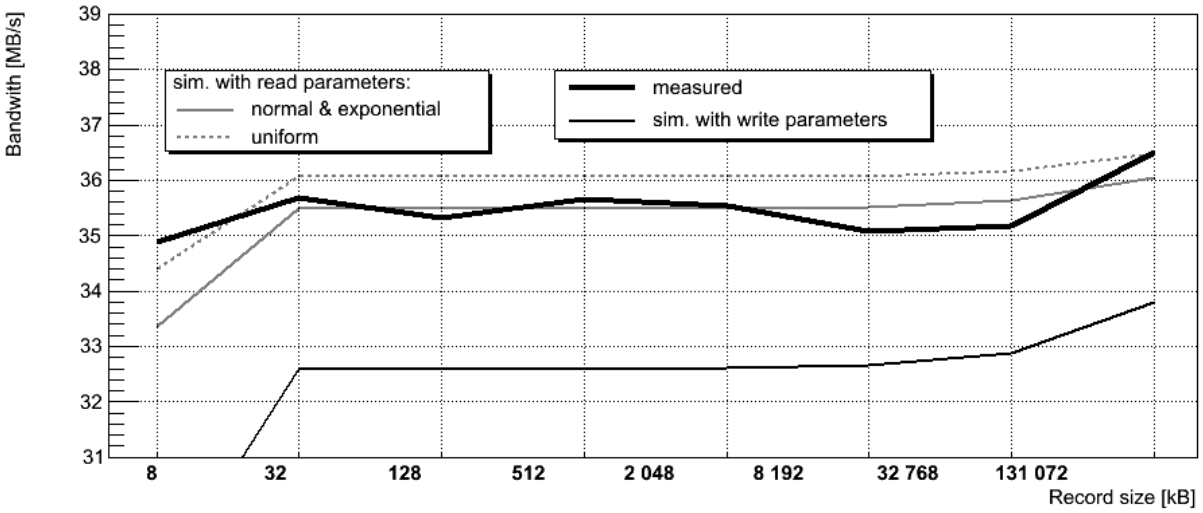

Figure 6: Simulation output for a read workload.

Although the greater part of distributions and parameters produced satisfactory results, to describe the service delay produced by the disk cache normal distribution is chosen because it produced the lowest average difference 1.29 % for all record sizes and 0.8 % if 8 kB records are omitted.

5. DISK ARRAY SIMULATION MODEL

The basic approach to the modelling of disk array is the same one used for modelling of disk drive. Because of its modularity this model uses a disk drive model described in previous section as a basis. Also it is important to notice that all models described in this section refer to the RAID5 architecture with 5 disks inside the array, although the number of disks in the model is not fixed and can be changed easily. Again, it was firstly customised for write workload, and then for read workload.

The array model developed for the ALICE experiment is presented by a modular model, shown in Fig. 7. Like the single disk drive model, on the most general level, it consists of two discrete event components: *Server* and *Array*.

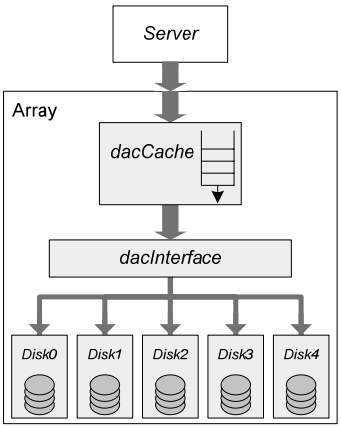

Figure 7: The general scheme of a disk array model with five disks.

The *Server* module is the same one used for the single disk drive model and it simulates the behaviour of the performance measurements program. The model of the *Array* module is derived from the closed queuing network model [6], [17] linking the three components, the

array cache (*dacCache*), the array controller interface (*dacInterface*), and disk modules (*Disk*). Even if both, the array cache and the array controller interface, are a part of the array controller, they are modelled as separate components to detach communication between controller and underlying disks form the cache logic.

Data flow in this model is very similar to the one in the disk drive model. The *dacInterface* module just coordinates a communication between the *dacCache* and underlying disks. The only significant difference is in the logic of the *dacCache* since it should provide transparent access to underlying disks on the RAID5 manner. As a result, it is modelled by a single queuing server with the fork-join queue. The incoming request, if it is bigger or equal to the stripe size, is forked to the stripe size chunks, the parity information is counted and everything is pushed on the queue. If the request is smaller than the stripe size, several requests are combined then until the stripe size is reached, and then the parity information is counted and they are pushed on the queue. According to the FIFO principle the requests are popped from the queue and sent to disks.

In general, parity information could be placed on different levels within array model like it is described in [19] and [23], but this paper describes an approach when it is integrated within disk array cache.

Fig. 8 presents measured and simulated results for write workload on both test arrays. Used simulation model was very straightforward. *Disk* module was the same one used for single disk simulation and two other components of *Array* module (*dacCache* and *dacInterface*) preformed they functionality without any additional service time delay. The average difference for such scenario was about 1.85 % for Infortrend and 7.81 % for DotHill array.

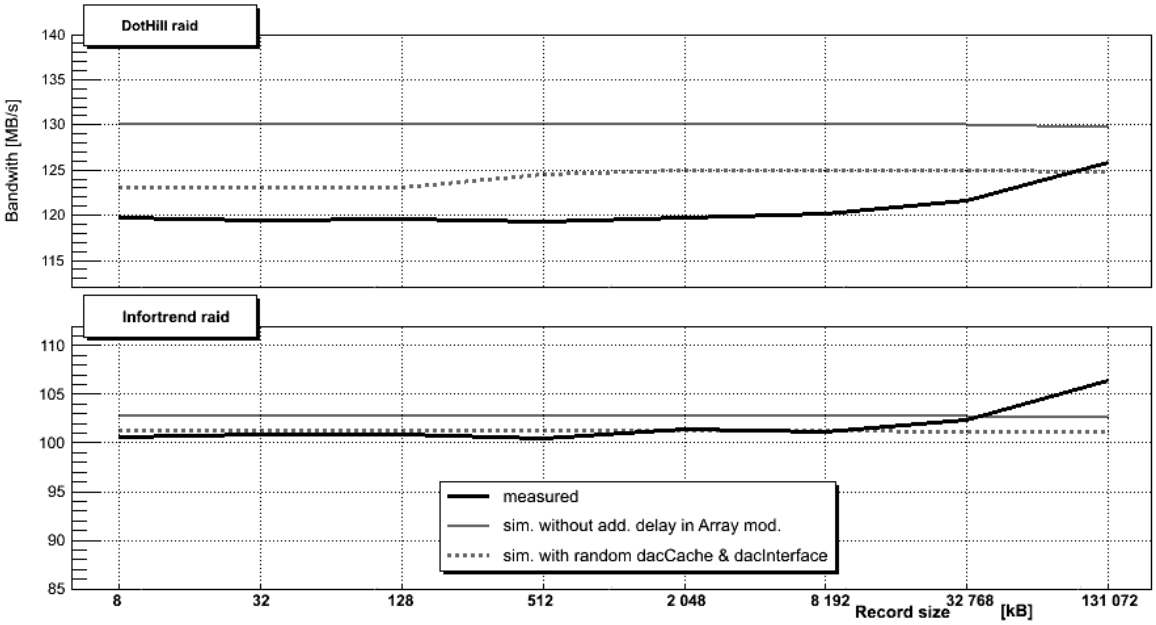

Figure 8: Array simulation model for write workload.

To explore the impact of additional delays on other components in *Array* module, some constant delay was added separately on *dacCache* and *dacInterface*. A constant delay on *dacCache* lowered the average difference on 1.1 % and 5.15 %, for Infortrend and DotHill array respectively. However, when a constant delay was applied on both, *dacCache* and *dacInterface* the average difference for Infortrend was increased on 2.27 %, while for the DotHill it was lowered up to 3.57 %.

Such results can be explained by different array architectures. In the Infortrend array underlying disks are connected with array controller through an Ultra ATA interface. It does not produce any additional delay and *dacInterface* component can be modelled without any additional delay. On contrary, disks in the DotHill array are connected through FC-AL (fibre channel arbitrated loop) that introduces some additional delay in the system through arbitration overhead. To mimic the behaviour of the real system as good as possible, the FC-AL delay is going to be presented by some additional delay in *dacInterface*. The resulting model behaviour is also presented in Fig. 8 and it produced the difference about 1.05 % and 3.16 % for Infortrend and DotHill array respectively.

Following step in disk array model development was to accommodate existing model for read workload. The resulting model behaviour is presented in Fig. 9, and it produced difference about 3.5 % for all file sizes on Infortrend array and about 4 % on DotHill array. However if the smallest record size is omitted from the counted results, like in single disk simulation, the results are much better and about 2.5 % and 2.8 %, for Infortrend and DotHill array respectively.

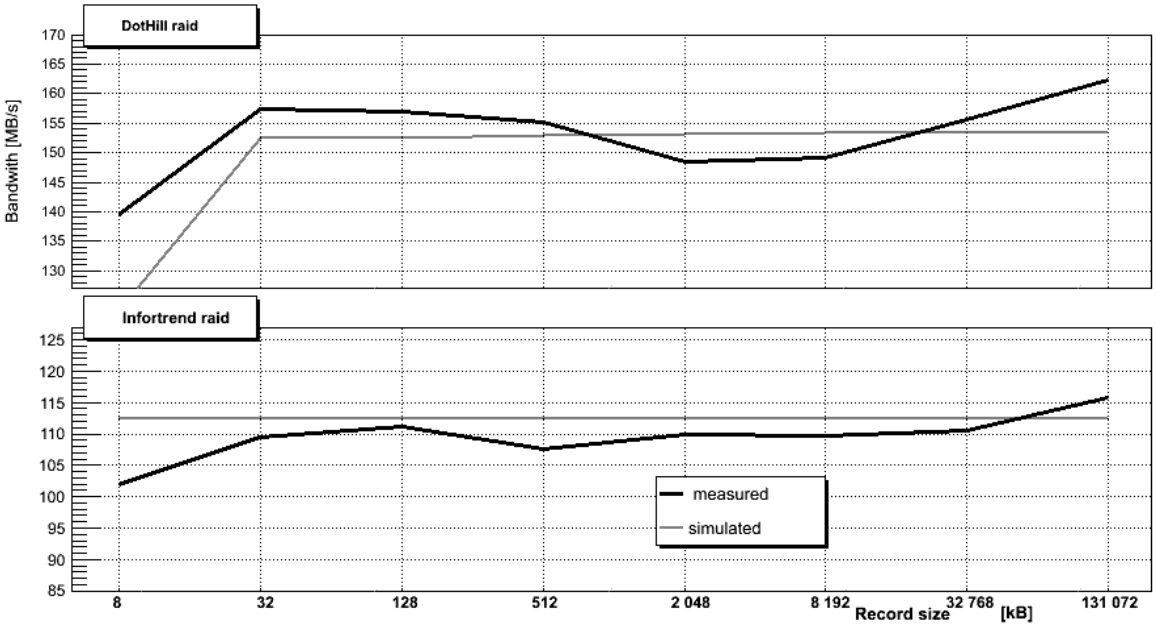

Figure 9: Array simulation model results for read workload.

Also, as it can be seen on both Figs. 8 and 9 there is no ascending trend for large record sizes that is the main characteristic of all measurements. The reason for that is the striping, or more precisely the way the developed model handles striping. In this model all incoming records were forked on 128 kB chunks (128 kB was used because it is defined as the stripe size for the Infortrend and DotHill array), and then no matter what was the original record size 128 kB chunks were sent. So, each record size greater than 128 kB, in all components beyond dacCache was treated as 128 kB record size. That logic even produced a slightly degraded throughput for largest record sizes, because all communication delays were more immanent for a huge number of repeats.

6. CONCLUSION

This paper gives a preview of disk array and disk drive model development. Although, such models exist for more than decade, they are still under study today as a part of more complex storage systems like Storage Area Networks (SAN).

Presented model differs from ones described in literature in several aspects. One of them is analytically counted positioning time for disk drive mechanism (seek time and rotational latency) in the simulation. As a result it can be classified as hybrid model, while all others are strictly analytical or simulation ones.

Further on, it describes how different storage devices interfaces (like Ultra ATA or FC) between array controller and underlying storage elements can be modelled by simple, randomly generated service time delay in *dacInterface* module.

Finally, it provides more than satisfactory results compared with test measurements. The differences between simulated and measured throughput is summarized in Table I.

Table I: Differences between measured and simulated results.

	Difference [%]		
	Single disk	Ultra ATA Array	FC-AL Array
Write workload	0.98	1.5	3.16
Read workload	0.8	2.5	2.8

However, such results are not without limitation. Discussed model is customized for the requirements of the ALICE transient storage system, which means it is going to be used for a storage system simulation with sequential reading and writing of large files. So, those results are produced only for system simulation with this specific kind of workload.

The first step in future development should be to extend current model to represent behaviour of the ALICE TDR – a bunch of arrays connected through switch. That model should be used to explore the influence of different parameters, like data file size, record size, stripe unit size and disk number in array, on the system performance. When that goal is achieved, the next step is to use existing model as a basis for simulation of more complex storage architectures like SANs with wealth of storage devices.

REFERENCES

[1] Bokhari, S.; Rutt, B.; Wyckoff, P. (2006). Experimental analysis of a mass storage system, *Concurrency and Computation: Practice and Experience*, Vol. 18, No. 15, 1929-1950, doi:10.1002/cpe.1038

[2] Chen, S.; Towsley, D. (1993). The design and evaluation of RAID5 and parity striping disk array architecture, *Journal of Parallel and Distributed Computing*, Vol. 17, No. 1-2, 58-74, doi:10.1108/03321640910991994

[3] Hock, N. C. (1996). *Queuing Modeling Fundamentals*, John Wiley, West Sussex, England

[4] Merchant, A.; Yu, P. S. (1996). Analytic modeling of clustered raid with mapping based on nearly random permutation, *IEEE Transactions on Computers*, Vol. 45, No. 3, 367-373, doi:10.1109/12.485575

[5] Bachmat, E.; Schindler, J. (2002). Analysis of methods for scheduling low priority disk drive tasks, *Proceedings of the 2002 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, 55-65

[6] Varki, E.; Merchant, A.; Xu, J.; Qiu, X. (2003). An integrated performance model of disk arrays, *Proceedings of the IEEE International Symposium on Modelling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS'03)*, 296-305

[7] Harrison, P.; Zertal, S. (2007). Queuing models of RAID systems with maxima of waiting times, *Performance evaluation*, Vol. 64, No. 7-8, 664-689, doi:10.1016/j.peva.2006.11.002

[8] Thomasian, A.; Fu, G.; Ng, S. W. (2007). Analysis of Rebuild Processing in a RAID5 Disk Arrays, *The Computer Journal*, Vol. 50, No. 2, 217-231, doi:10.1093/comjnl/bxl064

[9] Courtright, W. et al. (1996). RAIDframe: A Rapid Prototyping Tool for RAID Systems, Carnegie Mellon University

[10] Kim, J.-H.; Eom, S.; Noh, S. H.; Won, Y.-H.; Joo, B.-G. (2002). Studies on striping and buffer caching issues for the software RAID file system, *Journal of System Architecture: the EUROMICRO Journal*, Vol. 47, No. 1, 923-936, doi:10.1016/S1383-7621(02)00052-8
[11] Ruemmler, C.; Wilkes, J. (1994). An introduction to disk drive modelling, *IEEE Computer*, Vol. 27, No. 3, 17-29, ISSN 0018-9162
[12] Garcia, J. D.; Prada, L.; Fernandez, J.; Nunez, A.; Carretero, J. (2008). Using Black-Box Modeling Techniques for Modern Disk Drives Service Time Simulation, *Proceedings of 41st Annual Simulation Symposium (ANSS 2008)*, 139-145
[13] Xi, W.; For, W.; Wang, D.; Kanagavelu, R.; Goh, W. (2006). OSDsim - a Simulation and Design Platform of an Object-based Storage Device, *Proceedings of the 23rd IEEE Conference on Mass Storage Systems and Technologies (MSST2006)*, 97-102
[14] Ganger, G. R.; Patt, Y. N. (1998). Using system-level models to evaluate I/O subsystem designs, *IEEE Transactions on Computers*, Vol. 47, No. 6, 667-78, doi:10.1109/12.689646
[15] Wilkes, J. (1995). The Pantheon storage-system simulator, *Technical Report HPL-SSP-95-14*, HP Laboratories, Palo Alto, CA, USA
[16] Bhattacharyya, S.; Buck, J. T. et al. (1997). The *Almagest – Ptolemy 0.7 User's Manual*, University of California at Berkeley, CA, USA
[17] Simitci, H. (2003). *Storage Network Performance Analysis*. Wiley Publishing, Inc, Indianapolis, USA
[18] ALICE Collaboration (2005). *ALICE Technical Design Report of the Computing*, ALICE-TDR-012, CERN-LHCC-2005-018, CERN, Geneva, Switzerland
[19] Vicković, L. (2007). *Management and optimization of mass data storage system for ALICE experiment*, PhD thesis, University of Split, Split, Croatia
[20] Vickovic, L.; Celar, S. & Mudnic, E. (2010). Disk drive model development, *DAAAM International Scientific Book 2010*, Vol. 9, 535-548, doi:10.2507/daaam.scibook.2010.47
[21] Shriver, E. (1997). *Performance modelling for realistic storage devices*, PhD Thesis, New York University, New York, USA
[22] Shriver, E.; Hillyer, B.; Silberschatz, A. (2000). Performance Analysis of Storage system. Published as a section in *Performance Evaluation: Origins and Directions*, Lecture Notes in Computer Science in New York University Computer Science, 33-50, Springer Verlag
[23] Vickovic, L.; Mudnic, E.; Gotovac, S. (2009). Parity information placement in the disk array model, *COMPEL: The International Journal for Computation and Mathematics in Electrical and Electronic Engineering*, Vol. 28, No. 6, 1428-1441, doi:10.1108/03321640910991994