Simulating the ATLAS Distributed Data Management System

Martin Barisits\(^1,2\), Vincent Garonne\(^2\), Mario Lassnig\(^2\), Angelos Molfetas\(^2,3\) on behalf of the ATLAS Collaboration

\(^1\) Vienna University of Technology, Vienna, Austria
\(^2\) CERN, Geneva, Switzerland
\(^3\) University of Sydney, Sydney, Australia
E-mail: martin.barisits@cern.ch

Abstract. The ATLAS Distributed Data Management system organizes more than 90PB of physics data across more than 100 sites globally. Over 5 million files are transferred daily with strongly varying usage patterns. For performance and scalability reasons it is imperative to adapt and improve the data management system continuously. Therefore future system modifications in hardware, software, as well as policy, need to be evaluated to accomplish the intended results and to avoid unwanted side effects. Due to the complexity of large-scale distributed systems this evaluation process is primarily based on expert knowledge, as conventional evaluation methods are inadequate. However, this error-prone process lacks quantitative estimations and leads to inaccuracy as well as incorrect conclusions.

In this work we present a novel, full-scale simulation framework. This modular simulator is able to accurately model the ATLAS Distributed Data Management system. The design and architecture of the component-based software is presented and discussed. The evaluation is based on the comparison with historical workloads and concentrates on the accuracy of the simulation framework. Our results show that we can accurately model the distributed data management system within 80%.

1. Introduction

The ATLAS experiment\(^1\) stores and processes vast amounts of physics data. The experiment’s distributed data management system Don Quijote 2 (DQ2)\(^2\) is responsible to manage Petabytes of experiment data on over 750 storage end points worldwide. DQ2 is build atop the EGEE/EMI gLite\(^3\), NorduGrid/EMI ARC\(^4\) and Open Science Grid (OSG)\(^5\) middlewares. The DQ2 system provides a common interface layer to allow users to interact with the data without requiring knowledge of the specific structure of the ATLAS grid sub-systems. The daily workload averages at 5 million file transfers with transfer sizes ranging from several megabytes to hundreds of gigabytes. The usage patterns vary from physicists accessing output data from jobs submitted to the grid, over physics groups storing and accessing large amounts of data to fully automated analysis systems.

Modifications to the data management system, both in software and underlying hardware, need to be evaluated continuously to ensure the quality and correctness of the alterations. Conventional software systems are usually evaluated in an experimental testbed. However, for large-scale distributed systems an adequate testbed requires an impracticable amount of
resources. Also test deployments in the production system are considered too risky by experts, as they could result in bad performance or even unforeseen side effects.

The above reasoning motivated the investigation of a full-scale simulation framework for the ATLAS distributed data management system. The simulator is able to predict the outcome of system modifications without actually deploying the modifications to the production system. This paper describes this simulation framework.

The paper continues as follows: Section 2 gives an overview of the DQ2 architecture, the tracing system, and discusses the worldwide ATLAS grid topology. Section 3 describes the simulation framework. In section 4 we validate the simulator by replaying historic workload traces from the DQ2 system and evaluate its results. We finally conclude in section 5 and give an outlook on future work.

2. DQ2 architecture

The DQ2 system architecture, as displayed in figure 1, consists of a set of loosely coupled components, responsible for globally managing all ATLAS data. These components directly interact with a set of grid middleware systems. The DQ2 clients represent the direct interface to end-users and ATLAS services, for example the production system[6], analysis system[7] or data export system[8]. The central services components are responsible for the global book-keeping and logical organization of the data. The site services are an agent-based framework in charge of remote data transfers.

![DQ2 Architecture Diagram](image)

**Figure 1. DQ2 Architecture**

2.1. DQ2 tracer system

All transfer operations on the ATLAS grid are reported to a central tracing system[9]. Each file transfer is logged by the tracing system individually and several attributes, describing the transfer, are stored as well. These traces act as input to the simulation framework. A list of tracing attributes used in the simulator is described in table 1.
Table 1. DQ2 tracer attributes used in the simulation framework

| Attribute   | Description                                                                 |
|-------------|-----------------------------------------------------------------------------|
| Remote Site | Site where data is being downloaded/uploaded to                              |
| Local Site  | Site where data is being downloaded/uploaded from                            |
| Event Type  | Type of application initiating data transfer, e.g. an analysis job            |
| Time Start  | Time operation started (Subsequently referred to as $t_{\text{submit}}$)      |
| Transfer Time| Time stamp the first byte arrives (Subsequently referred to as $t_{\text{start}}$) |
| Validation Start| Time stamp the validation starts (Subsequently referred to as $t_{\text{validation}}$) |
| Time End    | Time operation ended (Subsequently referred to as $t_{\text{end}}$)          |
| File Size   | Size of file                                                                 |
| Hostname    | Name of host                                                                 |
| IP          | Hosts IP address                                                             |

2.2. ATLAS computing grid topology
The various ATLAS computing centers are categorized into four, hierarchical organized, tiers. Tier-0 is CERN itself and stores safe-guarded copies of the detectors raw data as well as first pass output data. The ten Tier-1 centers store a second copy of the raw data as well as provide access to processed output data. The Tier-2 facilities are associated to a Tier-1 site and provide access to specific processed data for physics analysis. Tier-3 centers are less well defined and mostly provide resources on a temporary basis.

Tier-0 to Tier-1 and Tier-1 to Tier-1 facility interconnections are provided by the LHC Optical Private Network, a dedicated network exclusively used for experiments data. Tier-1, Tier-2 and Tier-3 facilities are usually connected over conventional internet routes or dedicated national science networks.

3. DDM Simulator
Modeling a system always means a trade-off between simulation accuracy and simulation time\[10\]. A more detailed model will result in better accuracy but also in higher simulation time and vice-versa. The level of detail put into a system model basically depends on the system behavior one wants to simulate, evaluate and optimize. Our primary goal is to use the model to evaluate the systems performance from a user point of view, which focuses on the single question:

*How long does it take to store and retrieve data?*

This means, given an operations submit time $t_{\text{submit}}$ the simulator has to be able to accurately predict the operations end time $t_{\text{end}}$. Considering above’s simulation-accuracy principle and our simulation objective, we have to identify which DQ2 system components have to be represented in the simulator. There are three ways of representing a system component in a simulator.

*Performance models* aim to create realistic output as well as accurately model the performance characteristics of a system component. This means, the model can realistically predict how long the component will take to process its input.

*Output models* will create a realistic output which is equivalent to the real-world component, but no performance prediction is made. From a simulator point of view, components represented by an output model execute their tasks instantaneously.

Lastly, it is also possible not to represent a component by a model in the simulator at all. This is a valid approach if the component in question is not significantly involved in the workflow.
one wants to model. Thus, not representing the component would only yield a minimal or no simulation error at all.

The primary interaction with data in DQ2 is based on datasets. A dataset is a logical aggregation of files that are usually processed together and serve collectively as input or output of a computation or data acquisition process. The most fundamental function of DQ2 is to transfer datasets between two locations. A transfer workflow (See figure 2) in DQ2 consists of the following steps:

(i) The catalog is queried for available replicas for the requested dataset
(ii) Based on a policy replicas are selected
(iii) Permissions needed for the transfers are checked
(iv) The actual network transfers are executed
(v) Validation of the transferred data

![Transfer workflow in DQ2](image)

**Figure 2.** Transfer workflow in DQ2

The pre-steps (i-iii) of the transfer workflow include interactions with the DQ2 central services and site services components. Table 2 shows that the pre-transfer step in the workflow only takes a fraction of the total transfer duration. Also, the duration of these steps is much smaller than the expected simulation error of the simulator. Furthermore building a performance model for these database-dominated components is a very complex task. However, the output of these steps has a highly significant impact on the performance of the subsequent steps, as the decision of data source or destination is made by them. For these reasons we decided to represent these steps as an output model in the simulation framework. The network transfer step (iv) represents the actual (wide-area) network transfers. Included in this step is also the interaction with the grid’s mass storage systems, though these interactions are not considered separately in this work. With over 70% of the total operation duration for all transfers, and up to 97% for very large transfers, the network transfer corresponds the most significant step of the transfer workflow. The post-step (v) represents the validation of the data and is directly executed by the DQ2 clients. All transferred data is validated with the adler-32[11] checksum algorithm. The statistics show that this step takes 27% of the workflow duration and therefore is also significant for the overall performance evaluation. Consequently, the validation is represented as a performance model in the simulator.

3.1. Framework architecture

The framework consists of two parts: the simulator and a preparation tool. The preparation tool is responsible for generating the simulator-readable ATLAS grid topology files as well as
### Table 2. DQ2 transfer workflow statistics (Based on 90 days of traces)

| Size (Gb) | n    | \( \bar{t} \) (s) | % of total | \( \bar{t} \) (s) | % of total | \( \bar{t} \) (s) | % of total |
|----------|------|-------------------|------------|-------------------|------------|-------------------|------------|
| [0, 1)   | 62892722 | 0.05 | 0.4 | 8.04 | 74.5 | 2.7 | 25 |
| [1, 10)  | 6330402 | 6.3 | 1.6 | 362 | 95.3 | 11.3 | 2.9 |
| [10, 100) | 198234 | 7.1 | 0.3 | 2460 | 96.8 | 81.8 | 3.2 |
| [100, \( \infty \)) | 58088 | 0.06 | 0 | 53371 | 97.1 | 1567 | 2.8 |
| All      | 69547643 | 0.05 | 0.4 | 8.6 | 72.8 | 3.2 | 27.1 |

preparing the input trace, bookkeeping data and validation model. The full simulation workflow is displayed in figure 3.

![Figure 3. Simulation workflow](image1.png)

Figure 3. Simulation workflow

The simulator utilities are implemented in Python[12] while the simulator itself is based on C[13]. We chose a modular approach for the architecture design (See figure 4) to guarantee as much flexibility as possible. The user interaction layer is responsible for the command-line user interactions as well as reading/streaming the input files to the respective modules (e.g. streaming the input trace to the modules). The core layer interacts with all modules and represents the high-level operations of the simulated DDM system. The network model (See section 3.2) is responsible for performance predictions of network transfers. This module is tightly coupled with the core-layer, as the entire grid topology is represented by the network model. The replica selection module represents the source/destination replica selection policy. The catalog is responsible for dataset/file book-keeping. The validation model (See section 3.3) makes performance predictions of the validation process of files. All modules are interchangeable and can be replaced by modified (experimental) modules.

![Figure 4. Simulator architecture](image2.png)

Figure 4. Simulator architecture

3.2. Modeling the performance of network communication

A network performance model has to be able to accurately characterize the network topology that define links and their attributes (latency and bandwidth). It has to include traffic models that specify sender and receiver locations and handle concurrent data transfers based on a transfer protocol. The model has to realistically capture the characteristics of this transfer protocol (e.g. TCP fairness). In literature, there are several studies of techniques to model the described behavior. These techniques can be categorized into packet-level and flow-level
Packet-level simulators represent network transfers as sequence of events, such as packet departures and arrivals. Flow-level simulators use fluid-models, which were first proposed by Anick et al. [14], to model network transfers. Packet-level simulators provide a very detailed model of the internal operations of networks, as they simulate the movement on TCP packet level. Popular packets include ns-2 [15], GTNetS [16], and SSFNet [17]. However, simulation time increases with the number of packets. For large-scale systems, like the DQ2 system, this results in high simulation time due to very large data transfers [18]. For this reason specific grid-scale network models have been introduced. These models relax the notion of a packet, thus simulation packets do not correspond to real network packets anymore. Prominent simulators include Bricks [19], GridSim [20], OptorSim [21] and GangSim [22] (see [23] for a survey).

Flow-level models are orders of magnitude faster [18] than packet-level models but make several assumptions which do not hold in real networks (e.g. ignoring TCP slow-start), thus decreasing simulation accuracy [24]. However, Nicol et al. [25] showed that the error of estimated measures obtained by fluid-based simulations is very small compared to packet-based simulations and significantly only affects small transfers. While flow-models are not able to simulate certain characteristics (e.g. packet-level metrics) they are able to accurately model the performance of TCP flows. Prominent solutions based on fluid-based models in the literature include SimGrid [13], FSIM [26] and, [27]. Also hybrid models [28], combining fluid and packet techniques have been introduced.

However, the scope of this work lies in the area of modeling the global systems performance, from a DDM operations standpoint of view. Detailed packet-level models are not required to achieve this goal. For this reason we decided to use a fluid-based model, as they result in lower simulation time and the loss of accuracy is negligible small [29] compared to packet-level models. We chose to use the MSG stack of SimGrid [13], a C based API for the description of network models.

3.3. Modeling the performance of data validation

The validation process is based on the adler-32 checksum algorithm. As adler-32 is simply calculating checksums based on the integer-interpretation of each byte of the checked data, the process is mainly dependent on the size of the data. We therefore chose the ordinary least square regression [30] approach, which minimizes the sum of squared residuals to create an estimator. Consequently we used historic traces to create this validation model.

4. Evaluation

To evaluate the simulation framework we re-played historical DQ2 traces with the simulator. This means the simulation framework uses an existing dataset/file catalog dump and uses the same replica selection decisions as specified in the input traces. The traces included transfers of one day. However, only remote transfers (From one site to another) were considered, as site internal transfers are, due to their dependency on mass storage systems, not well enough covered by the model (See section 5). For the evaluation we use the absolute distance of the estimator \( t \) from the historic value and normalize it by the transfer duration, as explained in equation 1. We use the absolute distance of the estimator to review the simulation error in a critical way. Analysis of the real value of the distance of the estimator, e.g. to look for biases in any direction, will be part of future research.

\[
\text{error} = \frac{|\hat{t}_{\text{end}} - t_{\text{end}}|}{t_{\text{end}} - t_{\text{start}}}
\]  

(1)

The result of the evaluation is presented in figure 5. The overall simulation error is 20%. The box plot shows that small transfers are simulated more accurately than large transfers. Our interpretations for this effect go into multiple directions. The performance of the storage systems
providing and receiving the data is not accounted for in the simulation. Small transfers are most likely directly handled by a storage buffer and eventually written to disk or tape, thus hiding the actual storage performance from the data management system. For large transfers however this buffering effect becomes negligible. Furthermore the impact of network congestion and other network effects is most likely more significant for larger, long-lasting transfers. Also the available information on the network topology is only very limited and estimations for certain links had to be made. Lastly, the transfer systems of the middleware have the possibility to enforce quality of service properties on the transfers which could result in penalties of large transfers, which are not covered by the network model.

5. Conclusion
This paper demonstrated a simulation framework for DQ2, the distributed data management system of the ATLAS collaboration. This simulation framework can be used to evaluate modifications to the data management system. We showed which components of the DQ2 system have to be represented as performance or output models to achieve high accuracy during simulation. The simulation framework is a modular software package including both performance and output models for the network communication, the validation process, the book-keeping and replica management. The individual modeling-techniques were presented and the evaluation resulted in a simulation error of 20%.

Future work involves the modeling of mass storage systems and the generation of a failure-model, to simulate the impact of failures on the data management system.

Figure 5. Box plot of the normalized simulation error partitioned by transfer size
References
[1] Jones R and Barberis D 2008 Journal of Physics: Conference Series 119 072020
[2] Branco M, Cameron D, Gaidioz B, Garonne V, Koblitz B, Lassnig M, Rocha R, Salgado P and Wenaus T 2008 Journal of Physics: Conference Series 119 062017
[3] Laure E et al. 2004 Journal of Physics: Conference Series
[4] Ellert M, Grønager M, Konstantinov A, Könya B, Lindemann J, Livenson I, Nielsen J, Niininäki M, Smirnova O and Wääränen A 2007 Future Generation computer systems 23 219–240
[5] Pordes R et al. 2007 Journal of Physics: Conference Series 78 012057
[6] Klimentov A, Nevski P, Strandberg J, Tojo J, Vaniachine A and Walker R 2011 IEEE Nuclear Science Symposium and Medical Imaging Conference
[7] Maeno T 2008 Journal of Physics: Conference Series 119 062036
[8] Elsing M, Goossens L, Nairz A and Negri G 2010 Journal of Physics: Conference Series 219 072011
[9] Lassnig M, Fahrlinger T, Garonne V, Molfetas A and Branco M 2009 e-Science’09. Fifth IEEE International Conference on e-Science (IEEE) pp 156–163
[10] Law Averill M and David K 1991 Simulation modeling and analysis (McGraw-Hill)
[11] Deutsch P and Gailly J 1996 Zlib compressed data format specification version 3.3 (RFC Editor)
[12] Van Rossum G and Drake Jr F 1996 Python reference manual (Stichting Mathematisch Centrum, Amsterdam)
[13] Casanova H 2001 First IEEE/ACM International Symposium on Cluster Computing and the Grid, 2001. Proceedings. pp 430–437
[14] Anick D, Mitra D and Sondhi M 1980 ICC’80; International Conference on Communications p 13
[15] Breslau L et al. 2000 IEEE Computer 33 59–67
[16] Riley G F 2003 Proceedings of the ACM SIGCOMM workshop on Models, methods and tools for reproducible network research pp 5–12 MoMeTools ’03
[17] Cowie J, Nicol D and Ogielski A 1999 Computing in Science Engineering 1 42 –50
[18] Liu B, Figueiredo D, Guo Y, Kurose J and Towsley D 2001 INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies vol 3 pp 1244 –1253
[19] Takefusa A, Matsuoka S, Nakada H, Aida K and Nagashima U 1999 The Eighth International Symposium onHigh Performance Distributed Computing, 1999. Proceedings pp 97 –104
[20] Buyya R and Murshed M 2002 Concurrency and Computation: Practice and Experience 14 1175–1220
[21] Bell W, Cameron D, Millar A, Capozza L, Stockinger K and Zini F 2003 International Journal of High Performance Computing Applications 17 403
[22] Dumitrescu C and Foster I 2005 International Symposium on Cluster Computing and the Grid, 2005. CCGrid 2005 vol 2 pp 1151–1158
[23] Quetier B and Cappello F 2005 17th IMACS World Congress
[24] Fujiwara K and Casanova H 2007 Proceedings of the 2nd international conference on Performance evaluation methodologies and tools pp 12:1–12:10 ValueTools ’07
[25] Nicol D, Goldsby M and Johnson M 1999 Proceedings of the 1999 European Simulation Symposium vol 2
[26] Sakumoto Y, Asai R, Ohsaki H and Imase M 2007 International Symposium on Modeling, Analysis, and Simulation of Computer Systems pp 166–172
[27] Liu Y, Lo Presti F, Misra V, Towsley D and Gu Y 2003 Proceedings of the 2003 ACM SIGMETRICS international conference on Measurement and modeling of computer systems pp 91–101 SIGMETRICS’03
[28] Gu Y, Liu Y and Towsley D 2004 INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies vol 4 pp 2856–2866
[29] Velho P and Legrand A 2009 Simultools ’09: Proceedings of the 2nd International Conference on Simulation Tools and Techniques pp 1–10
[30] Goldberger A et al. 1964 Econometric theory.