Modeling I/O System in HPC: An ABMS Approach Work-in-Progress Paper

Diego Encinas, Marcelo Naiouf and Armando De Giusti

Informatics Research Institute LIDI
National University of La Plata
La Plata, Buenos Aires, Argentina
Email: {dencinas, mnaiouf, degiusti}@lidi.info.unlp.edu.ar

Sandra Méndez

High Performance Systems Division
Leibniz Supercomputing Centre (LRZ)
Garching, Munich, Germany
Email: sandra.mendez@lrz.de

Dolores Rexachs and Emilio Luque

Computer Architecture and Operating Systems Department
University Autonoma of Barcelona
Bellaterra, Barcelona, Spain
Email: {dolores.rexachs, emilio.luque}@uab.es

***Abstract*—Data-Intensive scientific applications use parallel Input/Output (I/O) software to access files. A tool capable of predicting the behavior of this kind of applications in High Performance Computing (HPC) would be very useful for parallel application developers. On the other hand, Agent-Based Modeling and Simulation (ABMS) has been used to model and simulate problems and complex systems for various science fields. This work presents a preliminary analysis to model a component of the I/O system in HPC using ABMS.**

Keywords–Agent-Based Modeling and Simulation (ABMS); HPC-I/O System; Parallel File System.

I. INTRODUCTION

In HPC, increasing demands in I/O system can cause bottlenecks more often. Parallel I/O plays a fundamental role to balance the fast increase in computational power and the progress of processor architectures. Although hierarchical memory architecture helps to mitigate performance penalizations due to disk accesses, its capacity is limited. In critical research areas, like nanotechnology, astrophysics, weather prediction and physical energy, several scientific and engineer simulations are processing more and more data. This fact also reflects the development of new fault-tolerance tools through periodical process state savings (checkpointing). Therefore, it is necessary to identify the factors that affect the performance and to propose new solutions that are capable of reducing performance gap between Central Processing Unit (CPU) and I/O.

The I/O system becomes more complex to provide an appropriate performance and to fulfill the requirements of the parallel applications on the HPC systems. Currently, an I/O system is a multilayer system where each layer has its responsibility and a behavior depending on the system workload. This layered structure hinders to observe the system global behavior, identify which layer is a bottleneck and whether it is doing the I/O inefficiently.

Running the same application with different I/O configurations gives the possibility to tune the I/O system according to the application access pattern. However, tuning the I/O system configuration on the production environment, it is only possible in some layers. Therefore, analyzing application requirements before configuring the real system could be an advantage. One way to predict application performance in HPC systems with different I/O configurations is using modelling and simulation techniques.

In HPC field, several I/O system simulators have been developed for parallel I/O. Simulator Framework for Computer Architectures and Storage Networks (SIMCAN) [1] is aimed at optimization of communications and I/O algorithms. Parallel I/O Simulator of Hierarchical Data (PIOsimHD) [2] was developed to analyze Message Passing Interface-Input/Output (MPI-I/O) performance and cluster hardware configurations. Co-design of Exascale Storage System (CODES) [3] is a framework for evaluating exascale storage system design points. High-Performance Simulator for Hybrid Parallel I/O and Storage Systems (HPIS3) [4] models the application workloads, parallel file system and storage system of the HPC environment. SIMCAN is based on OMNET++ and PIOsimHD is written in Java. CODES and HPIS3 are built on Rensselaer's Optimistic Simulation System [5], a parallel simulation platform. All these simulators are based on Discrete Event Simulation (DES).

ABMS is an approach to modelling complex systems composed of autonomous agents. Agents have behaviours, often described by simple rules, and interactions with other agents, which in turn influence their behaviours. DES and ABMS work mostly in discrete time but DES is used at low to middle abstraction. Compared to DES, in ABMS the behavior is defined at individual level, and the global behavior emerges as a result of many individuals, each following its own behavior rules, living together in some environment and communicating with each other and with the environment. ABMS is also typically easier to maintain because the model refinements normally result in very local, not global changes [6]. ABMS has been used in various fields, such as economics [7], health [8], biology [9] and sensor networks [10].

Due to the complexity of the modern HPC I/O systems, we consider ABMS appropriate to model and simulate it. We propose a model that represents the global behavior of the I/O system through the interaction of the different agents identified in each layer of the system. For tuning the I/O system, the agent attributes can be modified to observe how impact on the global behavior. When a performance problem is observed, this will be evaluated focusing on a layer, its agents and interactions with environment to detect the source of the problem.

 ISBN: 978-1-61208-442-8

In this context, our aim is to define a preliminary model of the HPC I/O system using ABMS. The proposed model will allow to design different configurations of the I/O system to assess their impact on the system performance. By using the model, we will be able to identify possible bottlenecks and predict the adjustments that will maximize overall system performance.

The rest of this paper is organized as follows. Section II presents the proposed methodology. Section III briefly describes I/O system in HPC. Section IV addresses the development of the initial model. Finally, Section V presents our conclusions and future work.

II. METHODOLOGY

Methodology applied to develop the model is an iterative spiral process. Each iteration involves 5 phases:

1) Collecting information of the general functionality of the I/O system in HPC.
2) Modeling I/O components and their interactions.
3) Implementation of the Model in a Simulator.
4) Validation of the simulator considering the information of the general functionality.
5) Identifying those components that affect application performance.

Modeling Agent-Based allow represent I/O components that may be either Hardware modules or Software and to refine or update them more easily. Compute nodes, I/O nodes, storage nodes and the network in a computer cluster exhibit a behavior depending on their roles. The processes and the components that form these systems can be modeled with interactions and attributes. The generated models can be used to model a particular cluster. The models must be adjusted with the characteristics of interest and combine them to represent the system behavior.

In the HPC scenario, modeling the behavior of hardware and software requires an appropriate level of abstraction to avoid a degree of detail that will cause a too complex and/or costly simulation. Therefore, the use of agents to model system components and attributes would provide a useful tool for analyzing the I/O behavior of the applications while each component is refined.

III. COMPONENTS OF THE I/O SYSTEM IN HPC

In this section, we present the results of the first phase of the methodology and section III-A details the Parallel Virtual File System (PVFS) selected for the first proposed model.

The HPC I/O systems have different configurations in each computer cluster. Usually, the parallel I/O system is represented as is shown in Figure 1. In this case, it is organized hierarchically according to the path that data follow in a computer cluster.

The parallel I/O system can be observed from three points of view:

- I/O architecture is related to storage and interconnection network features, data management, storage devices location, buffering/caching and storage node availability [11].
- I/O software stack comprises Scientific Data Library (like HDF5 and NetCDF), I/O Middleware (like MPI-IO) and Parallel File System (like GPFS, Lustre and PVFS2) [12].
- I/O access patterns of parallel applications. A parallel application I/O behavior model based on I/O patterns can be used to support the evaluation, design and selection of different I/O configurations [13].

Hardware components can have different roles depending on the data management that is performed by the parallel file system. Therefore, we have selected a parallel file system and analyze its functionality.

I/O libraries can be present in the compute nodes and their configurations affect the performance of the parallel applications.

I/O access patterns of parallel application could obtain different performance depending on the configuration of the I/O libraries, parallel file system or network interconnections. For this reason, the global behavior of the I/O system must consider the software and the configuration of the hardware components.

A. Parallel File System

In HPC, data management is carried out by a parallel file system that supports concurrent access to partitioned and distributed data in the storage system.

PVFS2 is an open source file system developed to support efficient read and write operations of large amounts of data. Due to its modular architecture, it is easy to include new hardware media and new algorithms. This fact demonstrates the PVFS suitability for parallel file system research.

Figure 2 illustrates the modular architecture of PVFS2 [14]. It is designed as a client-server architecture where the server provides the storage and the client contains the access logic to the distributed storage. Servers can be classified into data servers and metadata servers. The formers keep parts of logical files while the latters keep attributes of the logical file system objects (files and directories in PVFS).

IV. PROPOSED MODEL

Our approach for Modeling I/O System in HPC considers:

- I/O components (hardware and/or software) will be individually modelled defining its internal behavior (functional level) and the interactions (interfaces and communication actions) with the rest of the components.
- The functional level allows modelling each one of the specific components through the use of:
    - "State variables" characterizing the main "features" of the specific module/component.
    - "State transition maps" modelling, in a simple and powerful way, the dynamic internal behavior of each component based on the different interactions with other modules guided by the communication actions.

In this work, we begin modeling the I/O system for the use case request attributes of a file to the parallel file system. For example, a parallel application to execute an I/O operation

 ISBN: 978-1-61208-442-8

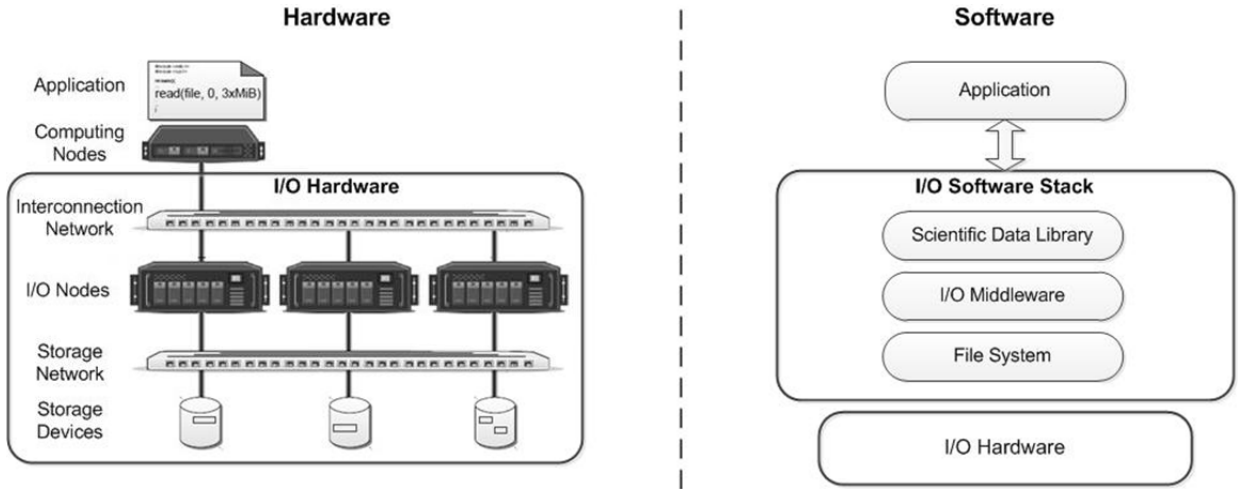


Figure 1. Components of I/O system in a computer cluster. The left picture shows the I/O architecture and the right picture depicts the I/O software stack.

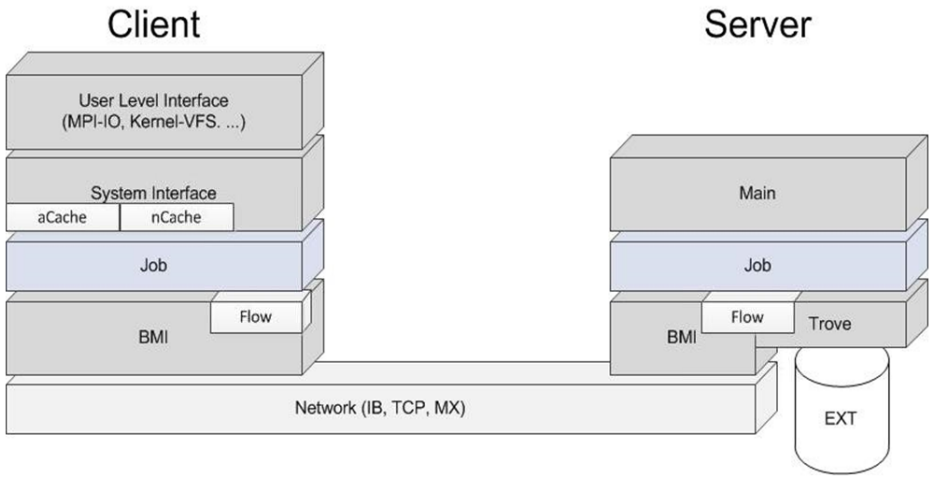


Figure 2. Software architecture of PVFS-2.

(`MPI_File_write`) will generally need request attributes. For this operation, in Figure 3(a) are shown the involved hardware and software components. The agents Application, Client, Network, Server are identified at high abstraction level. These agents are present in all the I/O systems. The agent Application starts the operation through the Client agent which communicates with the network interface to request attributes to the agent I/O server. The environment for the agents Application and Client is represented by the compute nodes, for the agent I/O server by I/O nodes and storage nodes; and for the agent network interface by the compute nodes, storage nodes and I/O nodes.

The main components for an I/O system that use PVFS2 is shown in Figure 3(b). The interactions of the agents Client, Network and I/O server were only pointed out at low abstraction level. However, data have went through several components that were not pointed out in this design stage, since only control signals were used as restriction of the initial modeling. One of the modelled agents is the Trove component. Trove has been included in the model because it is the only component that communicates with the storage.

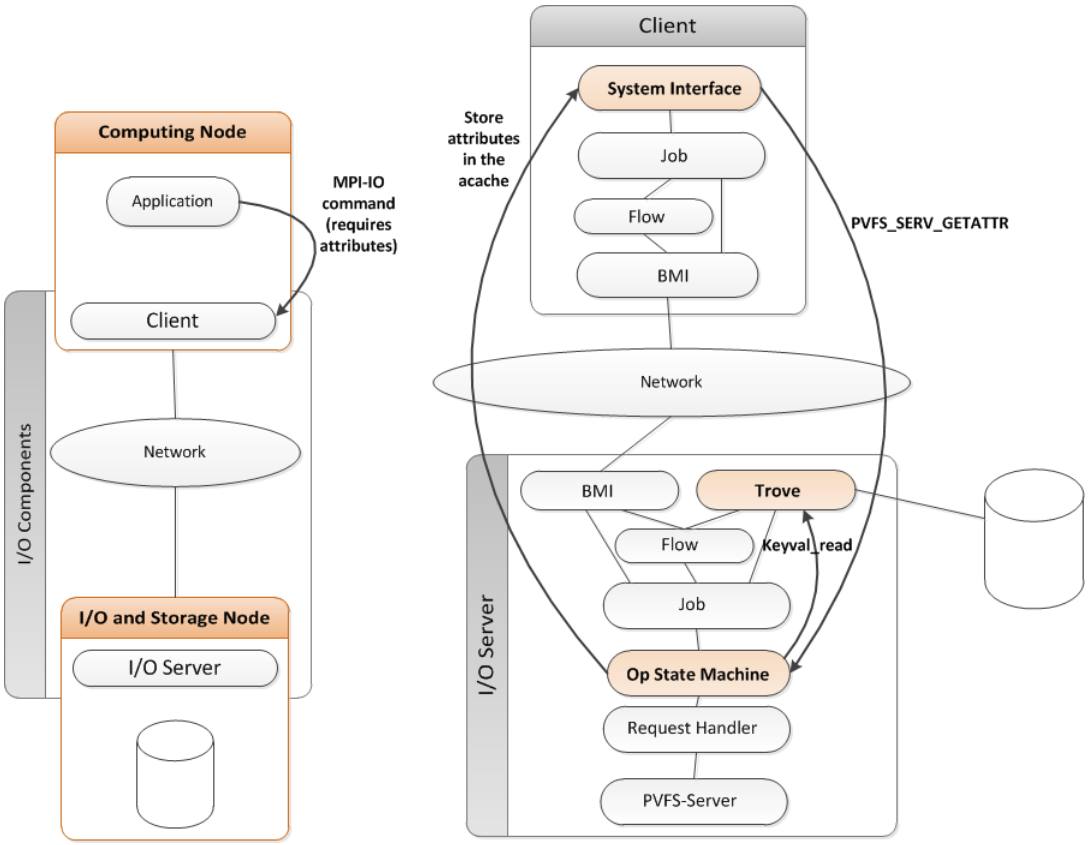
TABLE I. TROVE AGENT

Agent	Variables	Concept	Possible values
Trove	Directories	Object attributes	Attributes, entries (handles, keys)
	Metafile	Object metadata	Datafile handle (keyval), metafile distribution
	Datafile	Logical files data	Data

Trove is a software layer that is responsible of providing and managing persistence in the storage space for the implemented objects. To simulate the significant component behavior, a state machine can be used. The agent Trove state is defined by a collection of state variables that have different possible values. Each state will be associated with a specific combination of state variable values.

Table I shows an initial set of state variables that are defined with basic information to model Trove [15]. This component is selected in first place because it is interesting to model the disk write operation (HD or SSD) in the server node and Trove is the interface that manages it.

 ISBN: 978-1-61208-442-8



(a) Preliminary agents of I/O System in a request attribute operation.

(b) Components in an operation to request attributes and preliminary agents in PVFS-2.

Figure 3. Data flow for requesting attributes in PVFS2

In this example, the interactions needed to move from one state to another will be defined by the agent Trove inputs and outputs that correspond to categories provided by Trove interface [16]. These interactions include the management of storage, collections and dataspace and the accesses to Key/values and bytestream.

V. Conclusion

This paper presents a preliminary analysis to model a software component of the I/O system in HPC using ABMS. It has been shown by means an example that it is feasible to model PVFS-2 components, its state variables and the interactions with the different agents.

As future work, the agents required to complete the PVFS-2 model will be defined. Next, the model will be extended to include components of the I/O system. In addition, a simulation tool will be defined to implement the simulation model and thus, to be able to reproduce the behavior of the entire system.

Acknowledgment

This research has been supported by the MINECO (MICINN) Spain under contract TIN2011-24384 and TIN2014-53172-P.

References

[1] A. Núñez, J. Fernández, J. D. Garcia, F. Garcia, and J. Carretero, "New techniques for simulating high performance mpi applications on large storage networks," J. Supercomput., vol. 51, no. 1, Jan. 2010, pp. 40–57.

[2] J. Kunkel, "Using Simulation to Validate Performance of MPI(-IO) Implementations," in Supercomputing, ser. Lecture Notes in Computer Science, J. M. Kunkel, T. Ludwig, and H. W. Meuer, Eds., no. 7905. Berlin, Heidelberg: Springer, 06 2013, pp. 181–195.

[3] N. Liu et al., "Modeling a leadership-scale storage system." in PPAM (1), ser. Lecture Notes in Computer Science, R. Wyrzykowski, J. Dongarra, K. Karczewski, and J. Wasniewski, Eds., vol. 7203. Springer, 2011, pp. 10–19.

[4] B. Feng, N. Liu, S. He, and X.-H. Sun, "HPIS3: Towards a High-performance Simulator for Hybrid Parallel I/O and Storage Systems," in Proceedings of the 9th Parallel Data Storage Workshop, ser. PDSW '14. Piscataway, NJ, USA: IEEE Press, 2014, pp. 37–42.

[5] C. Carothers, D. Bauer, and S. Pearce, "ROSS: a high-performance, low memory, modular time warp system," in Parallel and Distributed Simulation, 2000. PADS 2000. Proceedings. Fourteenth Workshop on, 2000, pp. 53–60.

[6] A. Borshchev and A. Filippov, "From System Dynamics and Discrete Event to Practical Agent Based Modeling: Reasons, Techniques, Tools," in The 22nd International Conference of the System Dynamics Society, 2004.

[7] L. Tesfatsion, "Introduction to the special issue on agent-based computational economics," Iowa State University, Department of Economics, Staff General Research Papers, 2001.

[8] M. Taboada, E. Cabrera, F. Epelde, M. Iglesias, and E. Luque, "Using an agent-based simulation for predicting the effects of patients derivation policies in emergency departments," Procedia Computer Science, vol. 18, 2013, pp. 641 – 650.

[9] A. Siddiqa et al., "A new hybrid agent-based modeling & simulation decision support system for breast cancer data analysis," in Information and Communication Technologies, 2009. ICICT '09. International

 ISBN: 978-1-61208-442-8

Conference on, Aug 2009, pp. 134–139.

[10] M. Niazi and A. Hussain, “Agent-based tools for modeling and simulation of self-organization in peer-to-peer, ad hoc, and other complex networks,” Communications Magazine, IEEE, vol. 47, no. 3, March 2009, pp. 166–173.

[11] D. Kotz, “Introduction to multiprocessor I/O architecture,” in Input/Output in Parallel and Distributed Computer Systems, ser. The Kluwer International Series in Engineering and Computer Science, R. Jain, J. Werth, and J. C. Browne, Eds. Kluwer Academic Publishers, 1996, vol. 362, ch. 4, pp. 97–123.

[12] R. Ross, R. Thakur, and A. Choudhary, “Achievements and challenges for I/O in computational science,” Journal of Physics: Conference Series, vol. 16, no. 1, 2005, pp. 501+.

[13] S. Méndez, D. Rexachs, and E. Luque, “Modeling parallel scientific applications through their input/output phases.” in CLUSTER Workshops. IEEE, 2012, pp. 7–15.

[14] W. Ligon and B. Wilson, “Orangefs,” in High Performance Parallel I/O, ser. Chapman & Hall/CRC Computational Science, Q. K. Prabhat, Ed. Chapman and Hall/CRC, 2014, ch. 10, pp. 119–134.

[15] T. PVFS2, “Trove and PVFS2,” PVFS Development Team, Tech. Rep., 2015.

[16] S. Lang, “Trove: The PVFS2 Storage Interface,” PVFS Development Team, Tech. Rep., 2015.

 ISBN: 978-1-61208-442-8