Agent Based Simulation as an efficient way for HPC I/O system tuning

Diego Encinas[1,2], Marcelo Naiouf[1], Armando De Giusti[1], Román Bond[2], Sandra Mendez[3], Dolores Rexachs[3], Emilio Luque[3]

[1] Informatics Research Institute LIDI, CIC's Associated Research Center, National University of La Plata, 50 y 120, La Plata, 1900, ARGENTINA
[2] SimHPC-TICAPPS, Universidad Nacional Arturo Jauretche, Florencio Varela, 1888, ARGENTINA
[3] Computer Architecture and Operating Systems Department, Escola d'Enginyeria. Universitat Autònoma de Barcelona, Edifici Q Campus UAB, Bellaterra, 08193, SPAIN

Abstract. Generally, evaluating the performance offered by an HPC I/O system with different configurations and the same application allows selecting the best settings. This paper proposes to use agent-based modeling and simulation (ABMS) to evaluate the performance of the I/O software stack to allow researchers to choose the best possible configuration without testing the real system. Testing configurations in a simulated environment minimizes the risk of disrupting real systems.
In particular, this paper analyzes the communication layer of an HPC I/O system, more specifically the communication layer of the parallel file system (PVFS2). ABMS has been selected because it enables a rapid change in the level of analysis for modeling and implementing a simulator. It can focus on both macro- and micro-levels (how the aggregate behavior of system agents is born and analyzing their individual behavior).

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

1 Introduction

Improving the processing and storage of large amounts of data has become a challenge in parallel and distributed systems. Generally, evaluating the performance offered by an I/O system with different configurations and the same application allows selecting the best settings. However, to make changes in the configuration, analyzing the performance that the applications will be able to achieve before tuning the system (hardware and software) could be advantageous. Since at the hardware level a correct configuration encompasses the arrangement of computing and input/output nodes, and at the software level, it includes selecting the configurable parameters for the type of application used (intensive writing or

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63759-9_16

reading, method for performing I/O, and so forth). On top of this there can be a user level both for hardware and software such as an administrator, to generate the appropriate configurations. In other words, both users and researchers oftentimes want to make changes to these shares systems to analyze how this affects their applications, but they are unable to do so because these systems often run 24/7 or administration permissions are required. For this reason, one of the methods used to predict different configurations and the same application behavior in a computer system is using modeling and simulation techniques [1].

In this paper, we propose using Agent-Based Modeling and Simulation (ABMS) to develop a modeling of the Parallel Virtual File System 2 (PVFS2) communications layer called BMI (Buffered Message Interface) that will allow evaluating much of the performance of the I/O software stack. ABMS allows an easy change at the analysis level: it can focus on both macro- and micro-levels (how the aggregate behavior of system agents is born and, also, analyzing their individual behavior). The agent paradigm is used in various scientific fields and is of special interest in Artificial Intelligence (AI); it allows successfully solving complex problems compared with other classic techniques [2] like event-based simulation paradigm (Discrete Event Simulation, DES).

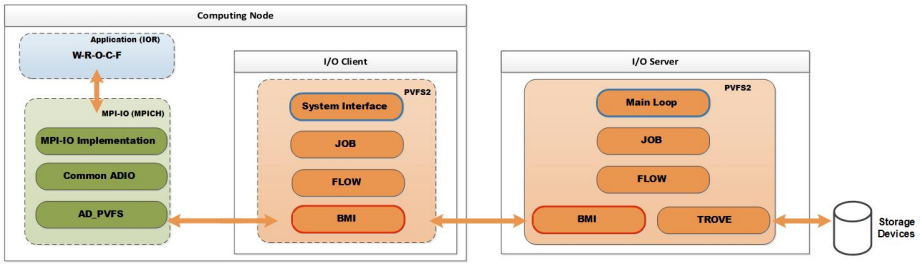

Fig. 1. At the left box, the layers in computer node and the right box represents the layers in the I/O server.

Generally, both paradigms operate in discrete time, but DES is used for low to medium abstraction levels. In ABMS, system behavior is defined at an individual level, and global behavior appears when the communication and interaction activities among the agents in an environment start. In fact, ABMS is easier to modify, since model debugging is usually done locally rather than globally. With the idea of using bottom-up modeling [3] and thus working in parts of the system, the agent paradigm was chosen.

To model the system and to have data to calibrate and validate, the test platforms developed have been previously explained [4] and add the appropriate monitoring tasks to the I/O software stack layers (Figure 1), as well as the different functional and temporal models achieved [5] and their implementation to check the different stages of the model. Furthermore, storage nodes can be classified into data servers and metadata servers. The file system divides the files

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63759-9_16

to be stored into fragments or stripes that are distributed in the data servers. The fragments are stored in the aforementioned buffers, both in client nodes and in server nodes.

In a nutshell, the simulator developed allows configuring the selected scenario (computing nodes, data servers and metadata servers), the file size for read and write operations, the access pattern (shared file or one file per process), and the number of iterations for each experiment. The output will allow quantifying I/O operations (immediate and non-immediate), bandwidth, operation time for the analyzed layer in the software stack. At a functional level, node-layer interactions can be viewed on the dashboard. To achieve a better control the test platform, each client node was configured to run a single application process.

2 Functional Analysis

The difficulty in the analysis of the I/O system lies in hardware heterogeneity and software stack complexity. In particular, work has been going on analyzing the PVFS file system as well as its OrangeFS equivalent, monitoring the different functions that make up each of the software layers. In the last publication [6] both client and server system interface and main loop layers, respectively, have been modeled in detail, and functional and temporal modeling and simulation were achieved. In the Modeling section below, the development achieved is explained.

To obtain data sets that allow carrying out a spatial modeling of the I/O system, monitoring tools were used on the Job, Flow and BMI layers of both the client and the server. These tools were used in previous works [6] but only focusing on obtaining temporary metrics. To further describe spatial behavior, the functions on the BMI layer were taken into account. BMI is a high-performance communications library used in PVFS2 that facilitates data transfer between the nodes of a computing system transparently. BMI will affect the model achieved through an agent that represents it and is explained in the Modeling section below.

Both `BMI_tcp_post_send_list()` and `BMI_tcp_post_recv_list()` are functions used to send and receive data asynchronously. Both functions are called by the PVFS2 client process to send a list of data buffers (BMI buffer) over the network to the dataservers.

By default, the size of the buffers at this level is 256 KiB. As regards the number of elements in the buffer, this strictly depends on the size of the selected stripe defined by the file system in the system interface layer. Generally:

$$stripesPerBuffer = \frac{BMIBufferSize}{StripeSize} \tag{1}$$

The number of stripes used by each client can be calculated as the size divided by the size of stripe. To know the number of stripes handled by each client, divide the previous result by the number of clients. Therefore:

$$StripesPerClient = \frac{FileSize \div StripeSize}{amountOfClients} = \frac{workload_i}{StripeSize} \tag{2}$$

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63759-9_16

Where $workload_i$ represents a portion of the total file size. Thus, the sum of all workload_i would be the size of the entire file. As mentioned in the previous section, to achieve greater control of the test platform, it is configured so that each client node executes a single application process. Thus, the size of the file is divided equally for each client node.

Another issue to consider is the number of BMI_tcp_post_send_list operations that are called to send a certain workload in bytes. Generally:

$$BMIClientOps = \frac{bytesPerClient}{BMIBufferSize} = \frac{FileSize \div amountOfClients}{BMIBufferSize} \quad (3)$$

If the number of dataservers is odd, the BMIclientOps relation must be modified by adding the file size as a new term.

All of this regarding `tcp_post_send/recv_generic` and `work_on_recv/send_op` applies to both client and server. The server differs from the client in terms of the function responsible for sending data. On the client side, the function `BMI_tcp_post_send/recv_lists` used, while the equivalent server-side function is `BMI_tcp_post_send/recv`. It should be noted that, in the context of the server, a write operation corresponds to the reception of data and a read operation corresponds to the sending of data.

For operations in each dataserver, this involves taking into account that the total number of client operations must reach all dataservers. That is, the number of BMI client operations are sent to the different dataservers equally, using a round robin method [7].

3 Modeling

In previous works, the functional and temporal analysis has been carried out by monitoring the interactions of the I/O system at the software layer level in the I/O nodes as well as between the functions that make up each of these layers. Also, the use of state machines and sequence diagrams allowed obtaining a more detailed model to describe the interactions that are triggered throughout the system when write, read or close files operations are performed.

In the area of modeling, the concepts of macro- and micro-modeling have been introduced to describe other systems [8]. It could be said that the modeling carried out in the parallel I/O system is a macro-modeling, since the interactions found are due to the communications necessary to represent the behavior of the system without individualizing spatial parameters such as I/O operations, stripes and number of bytes per second. A micro-modeling would imply modeling the behavior of the fragments to be distributed in each of the I/O nodes.

3.1 Macro-Modeling

In a previous work has been [6] detailed client-server interactions for the PVFS2 file system, the interactions between their system interface and main loop layers,

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63759-9_16

to analyze the spatial parameters for the BMI layers. In the BMI layer, immediate message quantification had to be added to the model; these messages are sent immediately to the receiver with the certainty that they will be received [9]. Next, the sequences of events that occur to send messages between client and server through the BMI layer are explained.

The `BMI_tcp_post_recv()` function is called to receive a 256-KiB buffer. Then, the `tcp_post_recv_generic()` function is started, but it fails to receive the entire buffer. At this point, the server receives only a few bytes or nothing at all. If some bytes of data are received, it is considered as an immediate operation (with the amount of bytes received). Therefore, `BMI_tcp_post_recv()` ends with errors. Next, the function `work_on_recv_op()` is started, which is responsible for receiving the remaining bytes that `tcp_post_recv_generic()` could not receive immediately. Several operations are necessary to receive the data buffer. Finally, `work_on_recv_op()` manages to receive the entire buffer and the `BMI_tcp_testcontext()` function is called to confirm the correct reception of the data. By means of the above-mentioned functions, the state machine corresponding to the BMI layer has been generated.

3.2 Micro-Modeling

To find spatial parameters data sets, the elements that make up the data buffers, the operations and the times in which they are transmitted have been monitored, but with certain configuration conditions in the real system. The deployed test scenarios used PVFS2 as parallel file system and MPICH distribution as middleware. File and transfer sizes were 1, 2, 3, and 4 GiB using the IOR (Interleaved-or-Random) benchmark (application layer) [10] on a 6-node physical cluster. Each cluster node has a specific and dedicated role on different physical machines: 2 client nodes, 3 dataserver nodes and 1 metadataserver node.

An important feature of PVFS2 is that it has an event logging system called Gossip that generates a text file with all the events ordered chronologically with relevant information such as timestamp for each operation and bytes involved. In this stage, the number of total operations, IOPS and Bps are obtained.

Finally, the data sets obtained are analyzed to find the algorithms that are shown below and model the behavior of the spatial parameters, and implement them in the simulator. That is, the verification and calibration stages of the simulation are carried out considering state machines, state variables and agents as in previous works [5,6].

It is observed that, to send data, the client uses immediate and non-immediate operations. In the case of reception by the client, only non-immediate operations are observed. Table 1 summarizes client node send operations.

With this information, equations are proposed to model total times, immediate times, number of total operations, number of immediate operations and bytes. The development of the modeling equations was carried out using the regression technique. Based on the above, the file size is used as the independent variable; in the following equations it is represented by the parameter x.

On the client side, for data send operations:

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63759-9_16

Table 1. Results for sending data by client.

`BMI_tcp_post_send_list()`	File Size (GiB)			
	1	2	3	4
BMI Client Ops	2049	4098	6147	8196
Immediate operations	235	1023	1635	2690
Non-immediate operations	1814	3075	4512	5506
Total time; immediate ops. (s)	0.015	0.063	0.1	0.16
Total time; immediate and non-immediate ops. (s)	130.54	254.39	378.03	493.21
bps (MB/seg)	3.92	4.02	4.06	4.15
BMI Client Total IOPS	15.69	16.11	16.26	16.62
Immediate IOPS	15187.25	16095.15	16190.76	16284.94

- $inmmediateOperations = 104.553571x^2 + 259.8107142x - 30.19285714$
- $totalTime = 123.39x + 4.46$
- $immediateOpsTotalTime = 0.00336x^4 - 0.00274x^2 + 0.07493x^2 - 0.0354x - (1.365E - 16)$
- $totalOperations = \frac{x \div amountOfClients}{BMIBufferSize} - 1 + 2x$
- $bytes = \frac{x}{amountOfClients}$

On the client side, for data receive operations:

- $totalTime = 131.61x + 0.097$
- *bytes* and *totaloperations* same as send operations.

As regards the implementation of the simulator, it was developed using the NetLogo framework. This tool is used especially in the context of complex system simulation. Once the equations explained in the previous section have been found, they are implemented in the simulator for their corresponding tuning and validation.

4 Results and Discussion

The scenario generated for the micro-modeling is configured in the simulator, and the following metrics are obtained to represent the spatial parameters related to operations and bytes over time. Besides, simulation total times are minimal.

Table 2 shows the difference between the values obtained from the real system and the values generated by the simulation model of the system. Table 3 show predictions/estimates for 5 and 6 GiB for client.

Based on the network equipment and the interfaces used, the amount of main memory available to the nodes that make up the cluster, the type of protocol and the defined file size, it was expected to find a low percentage of operations completed immediately and a high percentage of non-immediate operations.

As regards buffers, it was observed on the server side that many small operations are needed to fill a 262,144 KiB buffer. Data reception on the server refers

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63759-9_16

Table 2. Simulated vs. physical with absolute error.

	Executed				Simulated				Absolute error			
`BMI_tcp_post_recv()`	File Size (GiB)				File Size (GiB)				File Size (GiB)			
	1	2	3	4	1	2	3	4	1	2	3	4
BMI Server Ops	1366	2732	4098	5464	1366	2732	4098	5464	0	0	0	0
Bytes (GiB)	0.33	0.66	1	1.33	0.33	0.66	1	1.33	0	0	0	0
Total time (s)	87.14	169.87	252.39	329.35	86.22	168.88	250.95	333.62	0.9133	0.9985	1.4472	4.2624
BMI Server Total IOPS	15.67	16.08	16.24	16.59	15.84	16.18	16.33	16.38	0.166	0.095	0.0936	0.2119
Bps (MB/s)	3.88	3.98	4.06	4.13	3.92	4	4.08	4.08	0.0411	0.0235	0.0234	0.0528

Table 3. Prediction of 5 and 6 GiB for sending data to the client.

`BMI_tcp_post_send_list()`	File Size (GiB)	
	5	6
BMI Client Ops	10249	12299
Immediate operations	3888	5299
Non-immediate operations	6331	7000
Total time; immediate ops. (s)	0.37	0.93
Total time; immediate and non-immediate ops. (s)	622.022	745.61
Bps (MB/s)	4.11	4.12
BMI Client Total IOPS	16.47	16.49
Immediate IOPS	10384.98	5715.14

to write operations, so this behavior is reasonable/expected. On the client side, a similar behavior is observed when receiving (reading) data; to fill reception buffers, a considerable number of operations are necessary.

The times in the different functions scale reasonably as file size increases. As indicated by the predictions, it is expected that the times will skyrocket exponentially if the physical performance of the cluster does not change. Some of the improvements in physical features may include solid state drives in data servers or a change in the speed of the local network (to 1Gbps, for example). With the proposed improvements, the number of immediate operations is expected to increase considerably, consequently increasing IOPS.

5 Conclusions

This article presents an analysis and conceptual modeling of the PVFS2 communications layer, using the agent paradigm with the objective of better tuning a costly part of HPC such as communications I/O. When creating the model, a distinction is made to achieve a macro- and micro-model.

For macro-modeling, the interactions between client and server of PVFS2 are taken into account. Finite state machines allow modeling the exchanges and

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63759-9_16

defining agent implementation in the simulator. Additionally, the relations that represent spatial parameters functionality are obtained.

For micro-modeling, the mathematical equations are obtained to quantify the different parameters, such as number of operations, bytes, bandwidth and IOPS. Time is the only temporal parameter used. Consequently, the agents are BMI_client and BMI_server along with functions that help to the communication process between layers.

Finally, with the purpose of making system predictions, the simulator is run with input parameters of 5 and 6 GiB. As regards future work, different analysis, monitoring and tuning strategies will continue to be evaluated, such as different input and output stack elements in the system; and communications in metadata servers or at the metadata level. Another possibility is to include several processes on different nodes.

Acknowledgements This research has been supported by the Agencia Estatal de Investigacion (AEI), Spain and the Fondo Europeo de Desarrollo Regional (FEDER) UE, under contract PID2020-112496GB-I00 and partially funded by the Fundacion Escuelas Universitarias Gimbernat (EUG).

References

1. Boito, Francieli Zanon, et al. A checkpoint of research on parallel i/o for high-performance computing. ACM Computing Surveys (CSUR) 51.2 (2018): 1-35.
2. Paradiso, Martín, et al. An approach to parallelization of respiratory disease spread simulations in emergency rooms, 2022 International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, USA, 2022, pp. 1359-1365, doi: 10.1109/CSCI58124.2022.00244.
3. Siebers, P., Macal, C., Garnett, J. et al. Discrete-event simulation is dead, long live agent-based simulation!. J Simulation 4, 204–210 (2010). https://doi.org/10.1057/jos.2010.14
4. Gomez-Sanchez, Pilar, et al. Using AWS EC2 as Test-Bed infrastructure in theI/O system configuration for HPC applications. Journal of Computer Science and Technology 16.02 (2016): 65-75.
5. Encinas, Diego, et al. On the Calibration, Verification and Validation of an Agent-Based Model of the HPC Input/Output System. Proceedings from The Eleventh International Conference on Advances in System Simulation (SIMUL 2019). 2019.
6. Encinas, Diego, et al. An Agent-Based Model for Analyzing the HPC Input/Output System. International journal on advances in systems and measurements 13.3: 192-202. ISSN: 1942-261x. 2020.
7. Koziol, Quincey, ed. High performance parallel I/O. CRC Press, 2014.
8. Helbing, Dirk, et al. Micro-and macro-simulation of freeway traffic. Mathematical and computer modelling 35.5-6 (2002): 517-547.
9. Amerson, Gregory, and Amy Apon. Implementation and design analysis of a network messaging module using virtual interface architecture. 2004 IEEE International Conference on Cluster Computing (IEEE Cat. No. 04EX935). IEEE, 2004.
10. Shan, Hongzhang, and John Shalf. Using IOR to analyze the I/O performance for HPC platforms. No. LBNL-62647. Lawrence Berkeley National Lab.(LBNL), Berkeley, CA (United States), 2007.

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63759-9_16