# Trace-based Visualization as a Tool to Understand I/O Performance of Application and File System

Rodrigo Kassick, Francieli Zanon, Norton Barbieri, Philippe Navaux
Grupo de Processamento Paralelo
Instituto de Informática - UFRGS

## Abstract

*For a long time, data I/O and storage has been a source of performance contention for HPC applications: the limited bandwidth to transfer data in and out of processing nodes and storage systems counters the increasing computing power provided by bigger clusters with faster processors and accelerators. Scaling applications to a large number of processors in this context is no longer only a matter of good domain decomposition: data organization and access must be an ever-present consideration to enable scalability.*

*Understanding the behavior of applications and file systems (how much data is necessary to it's execution, when it is accessed, how it is distributed) is an important step to provide indications as of what optimizations are necessary – and at which points of the applications' code or the infrastructure.*

*In this paper, we present the use of trace-based visualization on two-levels – application and file system – as a tool to help understand the performance issues of a parallel application.*

## 1. Introduction

Cluster architectures and parallel programing are nowadays the standard solution to develop high performance applications. These application are usually associated with large data-sets used either as input or generated as result of their execution. Due to the volume and characteristics of these data-sets, there is need for a *permanent storage facility* accessible by all the instances of the application on the cluster.

The performance of this infrastructure is limited by two main factors: the speed of the underlying storage devices is orders of magnitude lower than the speed of the processing units; and the bandwidth available to transfer data from processing nodes to the storage system is limited both in the clients and at the storage servers. These factors combined make the storage infrastructure a significant bottleneck for parallel applications [5].

Making applications scale up to thousands of processing units requires developers, system administrators and designers to understand the behavior of the application and the I/O infrastructure.

One way of understanding this behavior is through visualization of events in the application generated through trace libraries. In the case of I/O, these events occur on a variety of places:

**the application code** , managed by the developer

**the VFS layer** of the system's kernels and other system processes (e.g. FUSE)

**the network stack** of clients and servers

**the processes of the storage servers** , distributed on several machines

In such scenario, studying the behavior of the application by looking at only one of these levels may provide incomplete information and lead to false assumptions on the performance issues of the system. Thus, it's necessary to make a comprehensive instrumentation of all these layers involved on the storage of data and study them in a combined fashion.

In this paper, we describe the use of trace-based visualization to study the performance issues of OLAM on the PVFS file system. The remaining of the paper is divided as follows: in Section 2 we describe the Ocean-Land-Atmosphere Model and it's I/O issues. Section 3 we detail the tracing and visualization tools used in this work. In Section 4 we present the tracing strategy and events organization to allow the intended performance study. Finally, we present our conclusions on Section 5

## 2. The Ocean-Land-Atmosphere Model and PVFS

The Ocean-Land-Atmosphere Model (OLAM) [8], developed at Duke University, aims to represent both global and local scale phenomena, as well as bi-directional iterations among regions on different scales. This is achieved by a global grid that can be locally (statically) refined on given points of interest. OLAM was developed to extend features of the Regional Atmospheric Modeling System (RAMS) [9] to the global domain. It uses many of RAMS' functions, such as physical parametrization and I/O formats.

The model consists of a global geodesic grid with triangular mesh cells with local refinement capability, the full comprehensible non-hydrostatic Navier-Stokes equations, a finite volume formulation of conservation laws for mass, momentum and potential temperature, and others. While the global domain expands the range of atmospheric systems and scale interactions that can be represented in the model, local refinements can be specified to cover areas with more resolution through recursion. OLAM is developed in FORTRAN 90 and parallelized with Message Passing Interface (MPI).

OLAM is an iterative model, where each timestep may result in the output of data as defined in its parameters. The amount of written data varies with the number of processes running the simulation (the number of independent output files increases with the processes count). This leads to an access pattern of "large amount of small files" that comes with a great cost in terms of I/O performance: the overhead of file creation and the small size of the writes causes the file system to perform poorly.

PVFS [4] is a good choice of parallel file system for this situation since it includes optimizations target on small files [3]. In recent works [6] we have shown that despite such optimizations, OLAM's I/O performance on PVFS still suffers greatly.

To mitigate this problem, we evaluated a hybrid MPI+OpenMP version of OLAM on top of PVFS [2]. The hybrid version outperformed the original MPI implementation by one order of magnitude due to the reduced number of files. This led to the current approach of studying the interaction of file system and application to better understand what optimizations can be used to improve performance.

## 3. Trace Libraries and Visualization Tools

To study the behavior of OLAM and the underlying file system we choose to use a tracing library to register events on the systems of interest and use them for later study. The tool used to capture these events is LibRastro [10].

Using LibRastro requires the developer to manually add the events of interest along the application code (as well as all the data necessary to describe the event). Each process involved in the target application generates a binary trace file. These files need to be merged and converted to a higher level language. This conversion must be made by an application-specific code, since the semantic of the events may change from one application to another.

One high-level event description language is Pajé [1]. Pajé allows the developer to describe *events*, *states* and *arrows* between distinct *containers* (a container being a process, a computing node, a cluster, a file or any element that may have states, events or be source or target of an arrow). The developer is free to create containers and the associated events in whatever fashion better describes his code.

The visualization of the events can be done either via the Pajé Visualization Tool or Triva [7], built on top of Pajé infrastructure. Pajé allows for a*gantt-chart* style, time-based visualization of the events and states of the containers. Triva, on the other hand, allows for visualization of other kinds of relations, like analysis of the time spent on each kind of state.

## 4. Application and File System Trace Extraction

To provide traces of the events in OLAM and PVFS, it's necessary to add LibRastro function calls around each code block that represents a state (like function calls). This implies inserting code in several points of OLAM and PVFS client and server. So far, we have instrumented these systems to trace the following kinds of states:

- OLAM's MPI and other utility functions
- OLAM's physical simulation functions
- HDF5 I/O functions in OLAM
- PVFS Server's state machine functions and states for the following tasks:
  - read
  - write
  - open
  - close
  - create
  - meta-data operations
  - actual I/O on disk

The traces generated by these processes are later merged and converted to the Pajé language. This is done by a specially developed converter that transforms the events from these different layers into more descriptive units of Pajé.

Figure 1 presents the proposed Pajé visualization of the traces. $P_0$ and $P_1$ represent MPI Processes for the application, while $Server_0$ and $Server_1$ represent instances of the file system's servers. The rectangles represent the states of the application and the file system, indicating the time spent on each given function. The arrows between the elements in this visualization can represent either a point-to-point communication (between processes), an I/O operation (between a process and a file) or the origin of a given I/O state – processes write to a file; a write state on a server represents a write operation on one specific file, etc.

The temporal event visualization of Figure 1 is the standard visualization obtained with the Pajé Visualization Tool. Triva will for other kinds of visualization, like temporal analysis of the time spent on each state of the application and file system, like studying what kind of I/O operation represented a bottleneck on the file system or in the application.

## 5. Conclusions

Studying I/O performance of distributed applications executing on top of parallel file systems is an important task to identify points of contention and propose optimizations that allow for better scalability. On the other hand, one-sided studies may lead to wrong assumptions on the causes of contention.

Trace visualization of both application and storage infrastructure's events will allow us to get a bigger picture of how these two independent systems interact and what are the real sources of contention. To achieve the level of detail intended in the beginning of the project, it's still necessary to finish instrumentation of PVFS's client and the conclusion of the converter code.

The objective of the project is to identify possible optimizations in OLAM and PVFS and implement and evaluate such changes, targeting a conference paper.

## References


[1] P. Augerat, C. Martin, and B. Stein. Scalable monitoring and configuration tools for grids and clusters. In *Parallel, Distributed and Network-based Processing, 2002. Proceedings. 10th Euromicro Workshop on*, pages 147–153. IEEE, 2002.

[2] F. Boito, R. Kassick, L. Pilla, N. Barbieri, C. Schepke, P. Navaux, N. Maillard, Y. Denneulin, C. Osthoff, P. Grunmann, P. Dias, and J. Panetta. I/O performance of a large atmospheric model using PVFS. In *Rencontres francophones du Parallélisme (RenPar'20)*, 2011.

[3] P. Carns, S. Lang, R. Ross, M. Vilayannur, J. Kunkel, and T. Ludwig. Small-file access in parallel file systems. *Parallel and Distributed Processing Symposium, International*, 0:1–11, 2009.

[4] P. H. Carns, I. I. I. Walter B. Ligon, R. B. Ross, and R. Thakur. Pvfs: a parallel file system for linux clusters. In *Proceedings of the 4th conference on 4th Annual Linux Showcase and Conference (ALS'00)*, pages 28–28, Berkeley, CA, USA, 2000. USENIX Association.

[5] J. Michalakes, J. Hacker, R. Loft, M. O. McCracken, A. Snavely, N. J. Wright, T. Spelce, B. Gorda, and R. Walkup. WRF Nature Run. *Journal of Physics: Conference Series*, 125(1):012022, 2008.

[6] C. Osthoff, P. J. Grunmann, F. Boito, R. Kassick, L. Pilla, P. O. Navaux, C. Schepke, J. Panetta, N. B. Maillard, P. L. S. Dias, and R. Walko. Improving performance on atmospheric models through a hybrid OpenMP/MPI implementation. In *Proceedings of the 9th IEEE International Symposium on Parallel and Distributed Processing with Applications*, 2011.

[7] L. Schnorr, G. Huard, and P. Navaux. Triva: Interactive 3d visualization for performance analysis of parallel applications. *Future Generation Computer Systems*, 26(3):348–358, 2010.

[8] R. Walko and R. Avissar. The Ocean–Land–Atmosphere Model (OLAM). Part I: Shallow-Water Tests. *Monthly Weather Review*, 136:4033–4044, 2008.

[9] R. Walko, C. Tremback, and R. Hertenstein. *RAMS - The Regional Atmospheric Modeling System - Version 3b - Users Guide*. ASTER Division, Fort Collins, CO, 1995.

[10] G. Wiedenhoft, P. da Silva, B. Stein, and L. de Computaçao. librastro: Uma biblioteca para geraç ao e leitura de rastros de aplicaç oes.

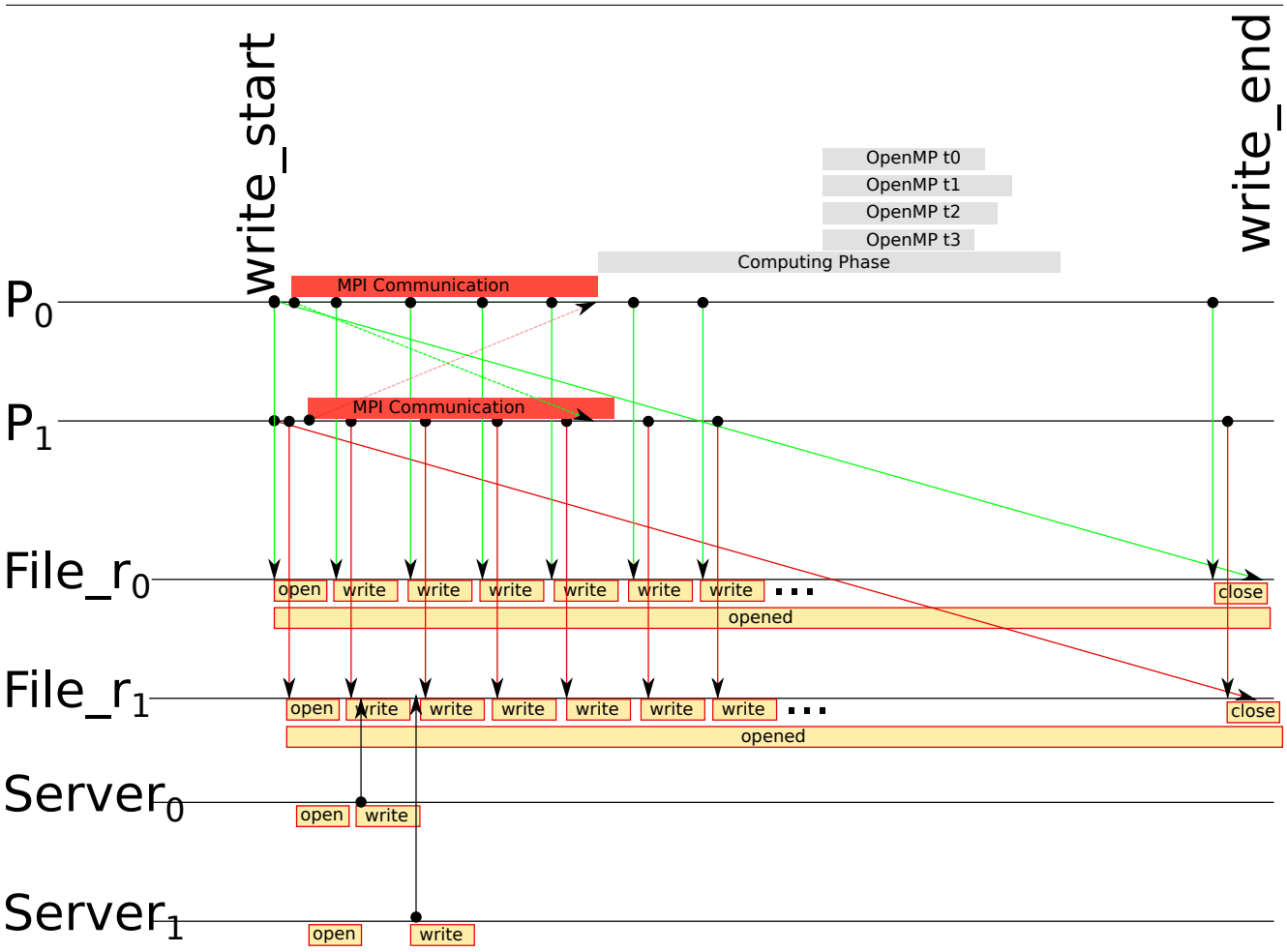


Figure 1: Proposed Components in Pajé