Evaluating I/O Scheduling Techniques at the Forwarding Layer and Coordinating Data Server Accesses

Jean Luca Bez, Philippe O. A. Navaux

[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil

{jean.bez,navaux}@inf.ufrgs.br

Abstract. *In this dissertation, we evaluate I/O scheduling techniques for the I/O forwarding layer of supercomputers. We demonstrate that existing algorithms that work to improve spatiality and request size of the access patterns are only partially effective. We propose TWINS, a new scheduling algorithm to coordinate the access of intermediate I/O nodes to the data servers. Our solution decreases concurrency at the latter, a factor proved to negatively affect performance. We are able to improve read performance from shared files by up to* 28% *over state-of-the-art scheduling algorithms and by up to* 50% *over not forwarding I/O. Our collaborations to the HPC field accounted for 16 papers and also motivated interactions with researchers from Argonne, BSC, INRIA, and LNCC.*

1. Introduction

Scientific applications such as climate, seismic, and biology simulations fill the High-Performance Computing (HPC) field with rising performance requirements in order to provide knowledge and help understand complex phenomena. "The Opportunities and Challenges of Exascale Computing" report presented by the U.S. Department of Energy [DOE 2010] stated that the Exascale problem is more than just a matter of scale. Applications' behavior and performance will be determined by a complex interplay of the program code, processor, memory, interconnection network, and I/O operation. Consequently, achieving good performance on scale requires an optimized orchestration of those components and a whole system view in order to understand root causes of inefficiencies. Since I/O is a bottleneck for an increasing number of applications, due to the historical gap between processing and data accesses speeds, it has the potential of critically impacting applications' performance on the next generation of supercomputers.

These large-scale cluster and supercomputers rely on Parallel File Systems (PFS) to provide a persistent shared storage infrastructure. To achieve high performance these systems harness parallelism by distributing data across multiple storage nodes. However, as the number of processing nodes starts to grow, so does the existing bottleneck on the PFS servers. To alleviate the contention when accessing the shared servers, a technique known as I/O forwarding can be applied. It introduces a new layer in the I/O software stack to decrease contention in the access to a file system. Present in several of today's supercomputers, this additional layer presents great potential to apply optimizations on I/O requests. These include, but are not limited to, I/O request scheduling, reordering, aggregation, and compression.

Applications issue their I/O requests in different ways, depending on how they were designed and coded. Several characteristics such as how many requests are issued,

the requests' sizes and their spatial location in the file compose what we call the application's access pattern. This pattern has a direct impact on performance, hence a lot of research effort is put into optimizing data access. Despite the fact that some applications try to locally optimize their access patterns, interferences generated by multiple applications accessing the shared storage infrastructure might also break or compromise the efficiency of the optimizations performed on the client side. In these cases, the I/O scheduling technique can be applied to improve access to the file system data servers by organizing and reordering requests, taking into account multiple competing applications.

This optimization technique has already been successfully applied to the forwarding layer [Vishwanath et al. 2010, Ohta et al. 2010] to adjust the applications' access patterns. In this work, we evaluate a distinct set of schedulers including algorithms that were proven to bring performance improvements when applied directly to the data servers. We demonstrate that the FIFO (*First In, First Out*), HBRR (*Handle-Based Round-Robin*), TO (*Time Order*), SJF (*Shortest Job First*), and MLF (*Multilevel Feedback*) schedulers[1] are only partially effective because the access pattern is not the main factor that influences the performance of requests through the I/O forwarding nodes. Concurrency and contention when accessing the data servers must also be taken into account.

The main objective of our research is to **evaluate I/O scheduling in the forwarding layer**, detecting algorithms that help improve performance and those whose overhead prevent them from being used in this context. Considering these goals, our main contributions are two-folded. We **evaluate** a total of five **scheduling algorithms** in the **I/O forwarding layer**. We demonstrate that existing schedulers, that provide improvements when used in the file system servers, do not provide similar results when applied to the forwarding layer. We propose a **new scheduling algorithm** to coordinate accesses to the PFS servers, aiming at **reducing contention** and increasing performance. As far as we know, this is the first work to propose a scheduling technique such as this to the I/O forwarding layer. We also conducted an extensive evaluation of this new scheduler considering different access patterns and forwarding scenarios.

The reminder of this document is organized as follows. Section 2 describes the scheduler proposed in the dissertation. Section 3 presents the experiments and main results. Section 4 summarizes our findings. Finally, in Section 5 we list our main contributions and publications.

2. TWINS: an I/O Scheduler to Coordinate Server Access

In this section, we present a new scheduling algorithm for the I/O forwarding layer called *Server Time WINdows* (TWINS). TWINS was conceived as an attempt to coordinate the accesses from the nodes of the forwarding layer to the PFS servers, in order to further reduce concurrency and contention. An extensive evaluation of existing I/O request schedulers demonstrated that, when applied to this layer, they were not able to yield similar improvements as when applied directly to the data servers. As some of them were indeed able to improve aggregation, these solutions demonstrated to be only partially effective because they do not take into consideration resource contention and data distribution. The main idea behind TWINS is coordinating intermediate I/O nodes' accesses to the file system so that, at any given moment:

[1] Details and particularities of each scheduling algorithm can be found in Section 3.2.1 of the dissertation.

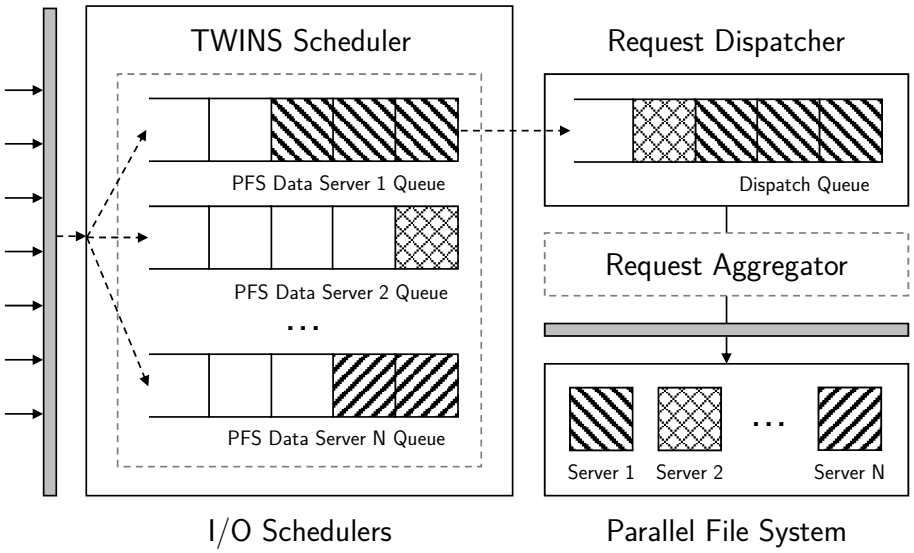

Figure 1. Flow of requests through the I/O forwarding node of the IOFSL framework daemon with the new TWINS scheduling solution.

I. an I/O node is focusing its accesses to only one of the PFS data servers;
II. the different I/O nodes are focusing on different servers.

To achieve this, TWINS keeps multiple request queues, one per data server. During the execution, TWINS iterates the different queues following a round-robin scheme, respecting a time window that must be dedicated to each server. We integrated the AGIOS scheduling library [Boito et al. 2015] in IOFSL [Ali et al. 2009], an open source forwarding solution, as a scheduling option. AGIOS can be used by I/O services to manage incoming requests at the file level and provides a simple API to develop new scheduling algorithms. We implemented TWINS through the AGIOS API as it makes our solution more generic, allowing it to be used by other I/O services, or by other I/O forwarding frameworks.

The new organization of the IOFSL daemon is illustrated in Figure 1. Using TWINS, requests are added to the per-server queues upon arrival at IOFSL. When the algorithm decides to process a request, a callback function written inside IOFSL simply adds it to the dispatch queue. This ensures requests will be processed in the order dictated by the active scheduler, in this case, TWINS.

TWINS requires some additional information on the file distribution that can be requested only once, with a minimum overhead ($54.3ms$, the average of 124 observations). This time can be expected to be diluted by longer read and write times. Using the file distribution information, the starting server for a request is obtained as a function of its starting offset and stripe size. With this information and an extra translation step (as detailed in Section 4.2 of the dissertation), TWINS is able to achieve the desired coordination effect. This translation is done according to the I/O node identifier. The N_{th} I/O node will use the N_{th} permutation of the list of servers as a translation rule. If the number of intermediate nodes is larger than the number of servers, more than one node may access the same server at the same time, but these concurrent accesses are minimized.

3. Results

All experiments conducted in the dissertation were carried out in two clusters from the French testbed Grid'5000 [Balouek et al. 2013]. Each I/O node runs on a separate machine, not shared with clients or PFS servers. Furthermore, the forwarding nodes are placed in the same cluster, close to the clients, to mimic a real life-like scenario.

We used the MPI-IO Test benchmark tool [LANL 2006], from Los Alamos National Laboratory, to simulate different access patterns, common to HPC applications. TWINS is able to provide read performance improvements of up to 28% over the baseline scheduling algorithms and of up to 50% over not using forwarding. The best results were obtained for the shared file 1D strided access pattern, and gains for the shared-file contiguous pattern were also observed – up to 20% over the baseline. Furthermore, TWINS performance for small (32KB) 1D strided requests is comparable to the use of collective operations. However, it is paramount to notice that TWINS represents a more transparent and generic solution than MPI-IO collective operations. As a result of being applied to the I/O nodes, TWINS is completely transparent to applications and I/O library generic. Therefore applications using any method to perform I/O operations, such as POSIX, can benefit from this optimization, without modifications to the source code.

Besides these common patterns, we have explored TWINS in a multitude of scenarios. We considered a multi-application scenario where we simulate two applications accessing the shared PFS under contention. In this situation, TWINS is able to improve performance by up to 16% over state-of-the-art schedulers, and up to 45% over not using forwarding. Interference, *i.e.* the ratio between concurrent and standalone execution time, is also decreased by up to 12% over the baseline algorithms and up to 31% over not using I/O nodes. We have also confirmed that the size of aggregated requests is larger when using TWINS, and by measuring the TCP congestion window size, that congestion is significantly reduced, hence improving performance.

We also explored the impact of distinct configurations of the forwarding layer, varying the ratio of clients connected per I/O node and of I/O nodes connected to the PFS data servers. We have observed that TWINS was able to provide improvements by tuning its window size to each scenario. Further details can be found in Section 5.6 of the dissertation. An automatic mechanism to tune this parameter is currently being discussed in collaboration with researchers from the Barcelona Supercomputing Center, Spain.

In summary, as no single scheduler or configuration is suited for all patterns and configurations, TWINS results have demonstrated that it is a sound contribution to the field. It further improves I/O performance for various scenarios, without source-code modifications, and without harming performance in situations it cannot improve it.

4. Conclusion

In this dissertation, we proposed a new scheduling algorithm for the I/O forwarding layer of supercomputers. Our solution aims at closing a gap observed in our evaluation on existing schedulers for this layer. We examined five algorithms that have not shown significant improvements, despite the fact that some of them increased the aggregated request sizes. This demonstrates that they are only partially effective and other factors were responsible for harming performance. Our hypothesis was that the lack of a coordination mechanism

between the I/O nodes, when accessing the data servers, might still be affecting performance. To confirm it, we proposed a new scheduling algorithm for the I/O forwarding layer named *Server Time WINdows* (TWINS). Our solution uses multiple requests queues and fixed time windows to coordinate the I/O nodes' accesses and decrease contention. To the best of our knowledge, ours is the first work to apply such a technique to the forwarding layer of the HPC I/O stack.

TWINS results have shown performance improvements for shared-file read access patterns of up to 28% over the state-of-the-art algorithms. Compared to not using I/O forwarding nodes, the gains were of up to 50%. Improvements were also shown for a multi-application scenario, accompanied by a decrease in interference. Moreover, even for situations where TWINS is not able to improve performance, it does not necessarily harm it. Additionally, the performance obtained by TWINS for the 1D strided read access pattern was comparable to what can be achieved by making the application use collective operations, a popular alternative applied to this scenario. Nevertheless, opposed to it, our proposal is completely transparent to applications and library independent.

Due to the historical differences in processing speed and data access speeds, and to the contiguous increase in the volume of data required and produced by scientific applications, every contribution in the field of parallel I/O for HPC has a great impact on how we make science today. Thus applications from multiple domains can benefit from faster I/O operations, yielding faster and more detailed results.

5. Publications and Contributions

In this section, we present the **main contributions** produced throughout the master's degree. Papers were submitted to a journal (ACM Computing Surveys); and to national (WSCAD, WPerformance, and ERAD) and international (PDP) conferences. We list here those related to this work and to the parallel I/O research field. Additional contributions to the HPC research field are detailed in the dissertation Section 6.2.

- Boito, F. Z.; Inacio, E. C.; **Bez, J. L.**; Navaux, P. O. A.; Dantas, M. A. R; Denneulin, Y. *A Checkpoint of Research on Parallel I/O for High-Performance Computing*. In: ACM Computing Surveys, v. 51, p. 1-35, 2018. **(Qualis A1)**.
- Carneiro, A. R.; **Bez, J. L**; Boito, F. Z.; Fagundes, B. A.; Osthoff, C.; Navaux, P. O. A. *Collective I/O Performance on the Santos Dumont Supercomputer*. In: Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, 2018. v. 1. p. 45-52. **(Qualis A2)**.
- **Bez, J. L**; Boito, F. Z.; Schnorr, L. M.; Navaux, P. O. A.; Mehaut, J. *TWINS: Server Access Coordination in the I/O Forwarding Layer*. In: Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, 2017. v. 1. p. 1-8. **(Qualis A2)**.
- Machado, V. R.; Rampn, N. G.; Braga, A. B.; **Bez, J. L.**; Boito, F. Z.; Kassick, R. V.; Padoin, E. L.; Diaz, J.; Mehaut, J.; Navaux, P. O. A.; *Towards Energy-Efficient Storage Servers*. In: The 32nd ACM Symposium On Applied Computing, 2017. p. 1554-1559. **(Qualis A1)**.
- Boito, F. Z.; **Bez, J. L**; Durpos, F.; Dantas, M.; Navaux, P. O. A.; Aochi, Hideo. *High Performance I/O for Seismic Wave Propagation Simulations*. In: Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, 2017 **(Qualis A2)**.

- Pavan, P. J.; Lorenzoni, R. K.; **Bez, J. L.**; Boito, F. Z.; Padoin, E. L.; Navaux, P. O. A.; Mehaut, J. *Eficiência Energética e Desempenho de E/S com Arquiteturas de Baixa Potência*. In: WSCAD 2016 - XVII Simpósio em Sistemas Computacionais de Alto Desempenho, 2016, Aracaju. **(Qualis B3)**.
- **Bez, J. L.**; Boito, F. Z.; Schnorr, L. M.; Navaux, P. O. A. *Escalonamento de I/O em Servidores de Encaminhamento*. In: XVI Escola Regional de Alto Desempenho (ERAD/RS), 2016, São Leopoldo. p. 173-174.

This dissertation was also submitted to the contest of Thesis and Dissertations (CTD) of the *XVIII Simpósio em Sistemas Computacionais de Alto Desempenho (WSCAD 2017)*, one of the most relevant HPC conferences in Brazil, held with the *29th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD 2017)*. Among the **16 submitted works** from fellow Brazilian researchers, this dissertation was awarded the **first place**. This highlights the relevance of the contributions herein for the HPC research field.

During the master's degree, the candidate participated in collaborations of his research group with researchers from LNCC - **Laboratório Nacional de Computação Científica** (Carla Osthoff), Brazil; **INRIA Grenoble** through the joint laboratory LICIA (Bruno Raffin, Jean-François Méhaut), France; and BSC - **Barcelona Super Computer Center** (Toni Cortes, Ramon Nou), Spain. During the research the candidate also interacted with researchers from ANL - **Argonne National Laboratory** (Robert Ross, Matthieu Dorier), USA.

References

Ali, N., Carns, P., Iskra, K., Kimpe, D., Lang, S., Latham, R., Ross, R., Ward, L., and Sadayappan, P. (2009). Scalable I/O Forwarding Framework for High-Performance Computing Systems. In *Proceedings...*, pages 1–10. IEEE International Conference on Cluster Computing and Workshops, IEEE.

Balouek, D., Amarie, A. C., Charrier, G., and Desprez, F. (2013). *Adding Virtualization Capabilities to the Grid'5000 Testbed*, pages 3–20. Communications in Computer and Information Science. Springer International Publishing.

Boito, F. Z., Kassick, R. V., Navaux, P. O. A., and Denneulin, Y. (2015). Automatic I/O scheduling algorithm selection for parallel file systems. *Concurrency and Computation: Practice and Experience*.

DOE (2010). The opportunities and challenges of exascale computing. Technical report.

LANL (2006). Los Alamos National Lab MPI-IO Test, User's Guide.

Ohta, K., Kimpe, D., Cope, J., Iskra, K., Ross, R., and Ishikawa, Y. (2010). Optimization techniques at the I/O forwarding layer. In *Proceedings...*, pages 312–321. IEEE International Conference on Cluster Computing, IEEE.

Vishwanath, V., Hereld, M., Iskra, K., Kimpe, D., Morozov, V., Papka, M. E., Ross, R., and Yoshii, K. (2010). Accelerating I/O Forwarding in IBM Blue Gene/P Systems. In *Proceedings...*, SC'10, pages 1–10. 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis, IEEE.