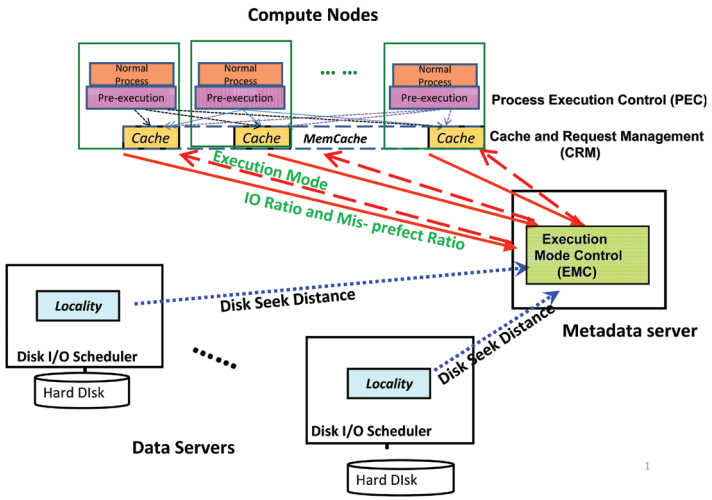
Opportunistic Data-driven Execution of Parallel Programs for Efficient I/O Services

Kei Davis, CCS-7; Song Jiang, Xuechen Zhang, Wayne State University

The work described here contributed to research into common runtime elements for programming models for increasingly parallel scientific applications and computing platforms. A parallel computing system relies on both process scheduling and input/output (I/O) scheduling to efficiently use resources and a program's performance hinges on the resource on which it is bottlenecked. Existing process schedulers and I/O schedulers are independent, but when the bottleneck is I/O there is an opportunity to alleviate it via cooperation between the I/O and process schedulers: the service efficiency of I/O requests can be highly dependent on their issuance order, which in turn is heavily influenced by process scheduling. We conceived a data-driven program execution model in which process scheduling and request issuance are coordinated to enable high I/O efficiency. Our design, DualPar, was implemented on the CCS-7 experimental cluster Darwin, and our experiments showed that DualPar can significantly increase system I/O throughput for relevant benchmarks [1].

Scientific computing is becoming increasingly data-intensive, with the consequence that the input/output (I/O) system is an increasingly severe performance bottleneck. In general, when a system resource becomes a parallel program's performance bottleneck, a better scheduling policy is sought to alleviate it. If the resource is the processors, this can be a process scheduling strategy for load balancing or co-scheduling. If the resource is storage, this could be an optimized I/O scheduler. In today's systems, when I/O service on the storage system becomes a program's bottleneck, the process scheduler becomes less relevant. This is especially true when I/O requests are mostly synchronous, most of the time the processes are idle waiting for the completion of their I/O requests and their scheduling is essentially a passive reaction to the progress of I/O operations.

Fig. 1. The architecture of DualPar.



Over decades the I/O stack, through which I/O requests pass and are serviced, has been significantly optimized, such as by forming larger sequential requests, hiding I/O time behind compute time with conservative prefetching, or increasing the parallelism of data access with parallel file systems. However, in these scenarios the way in which processes are scheduled for execution is not considered for its effect on I/O efficiency. I/O requests are issued by processes and the requested data are consumed by processes. Therefore, the order in which the requests are issued and served can be significantly influenced by process scheduling. When a process is driven by its computation, the computation determines the request issuance order, which can directly affect the request service order and I/O efficiency. The throughput of a hard disk for serving sequential requests can be more than an order of magnitude higher than that for serving random requests, so I/O request issuance order is a critical factor in I/O efficiency.

We proposed and implemented a new data-driven execution mode for parallel programs that is enabled when I/O becomes a bottleneck and I/O efficiency is being compromised by request issuance order. In this mode, a process is scheduled to resume its execution not when the request on which it is currently blocked is completed, but when the data that it and its peer processes are anticipated to read has been prefetched into the buffer cache, or the data to write have been buffered in the cache. This then allows the processes to run longer before they block on a new I/O request. In the data-driven mode we not only require that requests of a process be served in a batch, but also that the serving of requests from different processes be coordinated. This is because requests from different processes may disruptively compete for disk service and degrade disk efficiency. Furthermore, this coordination

For more information contact Kei Davis at Kei@lanl.gov

creates an opportunity to further improve the request issuance order to increase access sequentiality and to reduce the number of requests. Figure 1 illustrates the DualPar architecture.

In brief, we made the following contributions.

- We proposed a new program execution mode in which the scheduling of processes can be explicitly adapted for I/O efficiency. To this end, we use pre-execution to predict data to be requested for prefetching and a client-side buffer to hold written data for efficient writeback. In this way the computation of the program can be decoupled from the issuance of requests for its needed data, such that the I/O bottleneck can be alleviated, which cannot be achieved by conventional disk schedulers.
- We designed algorithms, which comprise DualPar, to detect the condition for enabling and disabling the data-driven mode and to coordinate data access and process executions.
- We implemented these algorithms in the MPICH2 MPI-IO library for message-passing interface (MPI) programs. We evaluated it with representative benchmarks, including mpi-io-test, ior-mpi-io, BTIO, and S3aSim. Experimental measurements on the CCS-7 large-scale experimental cluster Darwin show that DualPar can significantly improve I/O performance and, equally importantly, does not significantly degrade performance in other scenarios.

Figure 2 gives an example of relative I/O throughputs of standard MPI-IO, collective I/O, and DualPar for three concurrent instances of the BTIO benchmark. As shown, the advantage conferred by DualPar increases with the degree of parallelism.

Our basic premise was that, in the context of data-intensive computing, the independence of process scheduling and I/O scheduling, with their differing priorities, exacerbates the I/O bottleneck when using hard-disk-based storage. Preliminary experiments supported this premise, suggesting an opportunity to mitigate the I/O bottleneck via cooperation between the process and I/O schedulers. A new data-driven execution mode was conceived wherein processes would be scheduled according to the immediate (in-memory) availability of their I/O requests, this in turn being enabled by the pre-execution of processes. This scheme, named DualPar, was implemented as part of the MPICH2 MPI-IO library. Extensive testing showed that DualPar could significantly improve the performance of I/O-bound parallel benchmarks, whether or not they used collective I/O. DualPar is transparent to the user, requiring no changes to application source code.

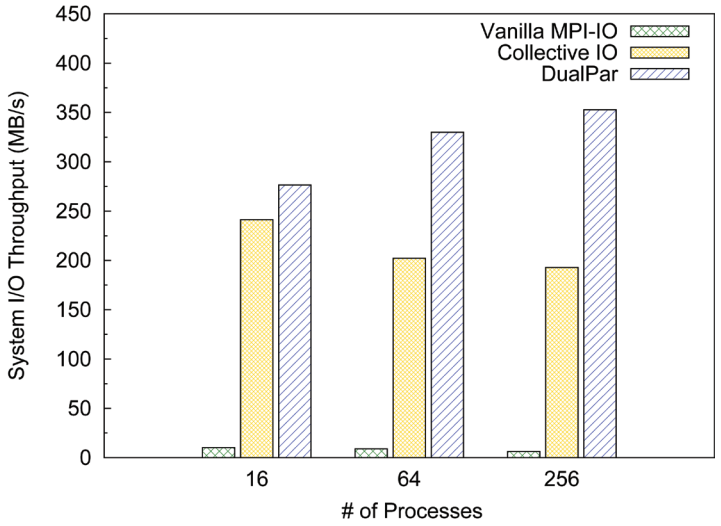


Fig. 2. System I/O throughput with three concurrent instances of BTIO benchmark. We compare throughput with DualPar to those with vanilla MPI-IO and collective IO, as we increase process parallelism from 16, 64, to 256.

[1] Zhang, X. et al., "iTransformer: Using SSD to Improve Disk Scheduling for High-performance I/O," *26th IEEE Int Parallel Distr Process Symp,* to appear (2012).

Funding Acknowledgments
DOE NNSA, Advanced Simulation and Computing Program; DOE Office of Science, Faculty and Student Program