8-19-1998

Applications of Parallel I/O

Ron Oldfield
Dartmouth College

David Kotz
Dartmouth College

Applications of Parallel I/O

Ron Oldfield and David Kotz

Technical Report PCS-TR98-337
Supplement to PCS-TR96-297

Department of Computer Science
Dartmouth College
Hanover, NH 03755-3510

August 19, 1998

Abstract

Scientific applications are increasingly being implemented on massively parallel supercomputers. Many of these applications have intense I/O demands, as well as massive computational requirements. This paper is essentially an annotated bibliography of papers and other sources of information about scientific applications using parallel I/O. It will be updated periodically.

1 Introduction

Scientific applications are increasingly being implemented on massively parallel supercomputers. Many of these applications have intense I/O demands, as well as massive computational requirements.

In this paper, we list and describe many papers and web pages that describe scientific applications that use parallel I/O. While we do not go into depth about the characteristics of each application, it is our hope that this paper helps researchers and application programmers to locate information that will help them to better understand the issues behind parallel I/O. This paper is meant to be used as a supplement to the previously written paper of the same name [Kot96]. The earlier version contains many applications not listed in this paper as well as a section on workload characterizations. For a complete parallel I/O bibliography, see [Kot97].

We intend to update this technical report periodically; check its web page for updated versions.[1] At that page you can also find a link to an on-line copy of this bibliography, with links to many of the cited papers.

Please feel free to send us additional references that you may find.

2 Papers about specific applications

These papers discuss specific applications, from the scientific point of view, but discuss their use of parallel I/O at some point. We do not include papers about scientific kernels (LU factorization, matrix multiplication, sorting, FFT, and so forth).

- [CDZ+97] They discuss the parallelization on message-passing computers of the *DNAml* algorithm, a tool used to construct phylogenetic trees from DNA sequences. By performing a run-time analysis of the behavior of the algorithm they came up with an efficient parallel implementation based on dynamic scheduling strategies, speculative run-time execution decisions and I/O buffering. They use I/O buffering (prefetching) to fetch tasks that need to be processed. The parallel code was written in C using PVM for message passing and is available via anonymous FTP.[2]

- [CMA+97] This paper is about a parallel database *Titan*, designed for handling remote-sensing data. Remotely-sensed data is acquired from satellite-based sensors and is commonly used for geographical, meteorological and environmental studies. The Titan system consists of a single front-end host and a multiprocessor back-end. All of the data is stored on the local disks of the back-end processing nodes. The data set is partitioned into coarse-grained data blocks and indexed from the front-end.

[1] http://www.cs.dartmouth.edu/reports/abstracts/TR98-337/.

When the front-end receives a query, it makes a list of all data blocks that intersect with the query. The front-end then distributes the data block requests to the back-end nodes. Each back-end node then computes a schedule for retrieving and processing the data blocks from its disks. When the data blocks have been processed, the image is sent back to the front-end. They attempt to minimize the I/O in several ways. They use a data-placement algorithm that accounts for common query patterns to achieve good disk bandwidth. They try to maximize the disk parallelism by using graph-based algorithms to efficiently decluster the data-set. They also try to minimize disk seeks by cleverly arranging disk blocks assigned to a single disk. Finally, the back-end nodes overlap computation, I/O and communication by issuing multiple asynchronous requests for data blocks from both the network and the disk. As requests are pending, the back-end node processes requests that have already arrived.

- [DLY+98] This paper describes a climate-modeling application that in a single day can generate approximately 60 Tbytes of raw data. The authors argue that only reasonable way to keep data-sets of this size manageable is to use data compression. They developed a run-length-encoding compression algorithm that uses the gather/scatter hardware available on the Cray parallel vector machines. The compression algorithm efficiently exploits multiple processors and ensures that the basic operations within the inner loops of the algorithm are vectorizable.

- [FMH+97] This paper describes a client/server application that emulates a high-power light microscope. They use wavelet compression to reduce the size of each of the electronic slides and they use a parallel data server much like the parallel database server used for satellite images [CMA+97] to service I/O requests.

- [KBCH95] This paper describes the architecture and high-level design of the data management system for the Earth Observing System Data and Information System (EOSDIS). They do not discuss implementation details, but they do discuss the tremendous I/O requirements of the project. The goal of EOSDIS is to maintain a large archive (petabytes) of scientific data that will quickly and easily be available to a wide variety of users ranging from "K-12 schools, to graduate schools, scientists, policy makers and public officials."

- [LSH98] This paper describes the implementation of a 3D simulation code for "turbulent flow and combustion processes in full-scale utility boilers on an Intel XP/S computer." They briefly discuss the I/O performance during the computations.

- [LEG+97] This paper is about a NASA project GEOS-DAS (Goddard Earth Observing System-Data Assimilation System). The goal of the project is to produce "accurate gridded datasets of atmospheric fields". The data will be used by meteorologists for weather analysis and forecasts as well as being a tool for climate research. This paper discusses their plans to parallelize the core code of the system. They include a section on parallel I/O.

- [SW97] They describe a parallel, out-of-core treecode library used for N-body simulations. Their approach targets machines in which secondary storage is attached to each processor. The library manually pages temporary data to the local disk to improve spatial and temporal locality. They showed results from a 16 node cluster of 200MHz Pentium Pro systems with 128MB of memory running Linux. They used a single 100baseT ethernet switch with bi-directional bandwidth of 80MB/s and latency of 150us. The entire system cost less than $60,000! They showed overall performance and paging behavior of an 80 million body model, a 5 million body model and a 500000 body model.

- [TSF+97] This paper is about the Fast Ocean-Atmosphere Model (FOAM), a climate model that uses "a combination of new model formulation and parallel computing to expand the time horizon that may be addressed by explicit fluid dynamical representations of the climate system." Their model uses message passing on massively parallel distributed-memory computer systems. They are in the process of investigating parallel I/O to further increase their efficiency.

3 Characterizations of parallel applications

These papers are detailed characterizations of the I/O access pattern of one or more parallel applications.

[2] ftp://ftp.ac.uma.es/pub/ots/pDNAml/

- [AVV98] This paper discusses algorithms to reduce the number of I/O requests in out-of-core geographical information systems (GIS) applications.

- [CHKM96] The authors presents a study of eight scientific applications on three types of parallel architectures. They have one paragraph on I/O performance and a nice table showing total I/O as well as I/O per Megaflop for each of the applications.

- [DIS96] This paper uses timing models to analyze the performance of the proposed tertiary storage system of the Earth Observing System Distributed Information System (EOSDIS). They examine tertiary storage and network performance of typical user scenarios from climate-modeling applications to identify potential bottlenecks in the system. The timing models include the I/O time from tertiary storage to disk cache and the time to send the data across the network. The modeling also accounts for device contention (network or tertiary storage).

- [KKCB97, KKCB98] They describe the I/O performance of a parallel computational chemistry package using the Hartree-Fock (HF) method and the Passion I/O library [TCB+96]. Before using any I/O optimizations, the I/O phase of the HF method accounted for up to 62% of their total execution time. They studied the effect of replacing the FORTRAN I/O calls with calls to the Passion I/O library. They then classified the factors that affect the I/O performance into application-related factors and system-related factors and examined the impact of each category on the I/O behavior of the application. They also ranked the optimizations based on the performance impact of the I/O phase of the HF method. All experiments were performed on an Intel Paragon.

- [LPJ98] This paper describes the I/O requirements of a parallel application that models shelf sea regions. The authors developed high-level routines to hide the details of the parallel I/O from the application code. They present analytical models of the I/O costs and show results on a Cray T3D.

- [MMD98] They use a parallel finite-element groundwater-transport code to analyze and compare three different strategies for parallel I/O. Each node in the application performs many writes of only a few kilobytes in length. The three strategies they use for I/O are to let a single processor

collect data and perform sequential I/O, use Intel's parallel file system (pfs) to collect data striped across an array of disks, and use their own Extended Distributed Object Network I/O library (EDONIO) [DR95]. EDONIO provides a "fast direct access random I/O operation to a global shared file by providing a large multi-gigabyte disk cache using the aggregate distributed memory." As expected, EDONIO showed significant improvement over the first two techniques.

- [OWO98] They describe the I/O performance for a seismic-imaging application called *Salvo* [OOVW96]. Salvo uses an I/O partition consisting of a portion of the compute nodes to perform all of the I/O. The I/O partition is used to perform asynchronous I/O requests, collective I/O, and data distribution. They derived an analytical model for estimating the I/O, computation and communication times for each of the operations and use the model to estimate the optimal ratio of compute nodes to I/O nodes for their application. Performance results are presented for the Intel Paragon.

- [SR98a] This paper compares logical I/O performed by the application with the corresponding physical I/O that takes place at the disk. By instrumenting the SCSI device drivers of the Intel Paragon OSF/1 operating system to record key physical I/O activities, they can correlate I/O patterns of the scientific application with physical activity of the file system. The authors performed experiments on a computational chemistry application called *MESSKIT*. They concluded, "physical input/output patterns induced by application requests are strongly affected by data striping mechanisms, file system policies, and disk hardware attributes."

- [SR97, SR98b] They compared the I/O performance of five scientific applications from the Scalable I/O Initiative (SIO) suite of applications running on a 512-node Intel Paragon XP/S. Their goals were to collect detailed performance data on application characteristics and access patterns and to use that information to design and evaluate parallel file system policies and parallel file system APIs. The related work section gives a nice overview of recent I/O characterization studies. They used the Pablo [RAN+93] performance analysis environment to analyze the performance of their five applications. The applications they chose to evaluate include: *MESSKIT* and *NWChem*, two implementations of the Hartree-Fock method for computational chemistry applications; *QCRD*, a quantum

chemical reaction dynamics application; *PRISM*, a parallel 3D numerical simulation of the Navier-Stokes equations that models high speed turbulent flow that is periodic in one direction; and *ECAT*, a parallel implementation of the Schwinger multichannel method used to calculate low-energy electron molecule collisions. The results showed that the applications used a combination of both sequential and interleaved access patterns, which shows that there is a clear need for a more complex API than what is given by the standard UNIX API. In addition, when the applications required concurrent accesses, they commonly channeled all I/O requests through a single node. Some form of collective I/O would have helped in these cases. They also made an observation that despite the existence of several parallel I/O APIs, programmers of scientific applications preferred to use standard Unix. They argued that this is mostly due to the lack of an established portable standard. They mention that their study was "instrumental in the design and implementation of MPI-IO". Their section on emerging I/O APIs is particularly interesting. They comment that "the diversity of I/O request sizes and patterns suggests that achieving high performance is unlikely with a single file system policy." They argue that we need a file system in which the user can give "hints" to the file system expressing expected access patterns or to have a file system that automatically classifies access patterns. The file system can then chose policies to deal with the access patterns.

- [SCM+98] They describe the design of a database application to "facilitate efficient access to and preprocessing of large volumes of satellite data". First, experiments were performed on a prototype parallel implementation that used a 16 node IBM SP2 with 12 GB of disk storage on each node. The experimental database used 16 indices that cover a portion of the west coast of North America. They measured general performance, scalability, I/O performance and preprocessing performance of their prototype database server. The results showed that I/O accounted for, on average, approximately 8% of the total processing time for a query; however, I/O accounted for over one quarter of the processing time for larger number of processors. They then discuss the design of a new system that focuses on three aspects of the image database system: "data placement on the disk farm, query partitioning and coordination of data retrieval, computation and communication over the entire machine." Their design is much like

that of the Titan system [CMA+97] .

4 Other papers on applications using parallel I/O

There are a few other papers that do not discuss specific applications, but still discuss issues relating to parallel I/O for scientific applications.

- [BCD97] The authors present techniques for implementing large scale irregular out-of-core applications. The techniques they describe can either be used by a parallel compiler (e.g., HPF and its extensions) or the programmer using message passing. The objectives of the proposed techniques are "to minimize I/O accesses in all steps while maintaining load balance and minimal communication." They demonstrate the effectiveness of their techniques by showing results from a Computational Fluid Dynamics (CFD) code.

- [NFK98] They describe an I/O project *ChemIO*, which defines an interface designed specifically for parallel out-of-core applications in computational chemistry. The ChemIO API supports three models: disk resident arrays, exclusive access files and shared files. Disk resident arrays support the transfer of data between global memory and secondary storage. This allows the programmer to read and write array data structures to and from local memory, remote memory and disk storage. They optimize the transfer of these data structures by supporting collective I/O. Exclusive access files allow each node in the computation to write to individually owned "scratch files". These files are primarily for out-of-core computations. A shared file allows multiple nodes to share access to a file. The application must handle mutual exclusion.

- [PSS96] They advocate the use of traditional demand-paged virtual memory systems in supporting out-of-core applications. They are implementing an operating system for the NEC Cenju-3/DE, a shared-nothing MIMD multiprocessor with a multistage interconnection network and disks on every node. The operating system is based on Mach, and they have extended Mach to allow user-provided [local] replacement policies. Basically, they argue that you can get good performance as long as you write your own replacement policy (even OPT is possible in certain applications),

and that this is easier than (re)writing the application with explicit out-of-core file I/O calls. They measure the performance of two applications on their system, with OPT, FIFO, and a new replacement algorithm customized to one of the applications. They show that they can get much better performance with some replacement policies than with others, but despite the paper's title they do not compare with the performance of an equivalent program using file I/O.

- [TLG98] This paper is an introduction to the IJSA Special issue on I/O in parallel applications. They argue the importance of the application program interface (API) in obtaining efficient parallel I/O and why the standard UNIX API is ineffective. They explain that an appropriate API should be explicitely parallel with support for collective I/O. Then they discuss MPI-IO, an API designed to address the I/O needs of high performance parallel applications.

5 Discussion

One of the goals of this paper was to show that scientific applications with large I/O requirements span many disciplines. The types of applications presented in this paper include medical applications, seismic imaging, climate modeling, computational chemistry and computational biology. This bibliography is by no means complete, but it does show that the demand for efficient I/O in scientific computing is abundant.

The techniques used by developers to relieve the intense I/O demands of scientific applications varied from improving the I/O interface to implementing improved out-of-core techniques. As shown in [MMD98, OWO98, SR98b, NFK98, TLG98], scientific applications clearly benefit from using an API that enables advanced parallel I/O techniques such as collective I/O, prefetching and data sieving; however, most application developers still prefer to use the standard inefficient UNIX API. This is partly because until recently (MPI-IO), no commonly used standard API for parallel I/O existed. We also suspect that some scientists are just starting to become aware of the importance of efficient I/O. Unless technology trends change, I/O will become a bottleneck for many more scientific applications.

We were also surprised by the number of papers about out-of-core applications [AVV98, PSS96, SW97, BCD97, NFK98]. An interesting point made by

[SW97] is that "the ratio of DRAM to disk pricing suggests the use of out-of-core techniques to overcome memory capacity limitations." Some believe that advances in memory technology will result in memory capacities so large that there will no longer be a need for out-of-core applications and that any reasonable application should be able to fit in-core. This type of attitude is naive. Even if memories do become large enough to fit most applications in-core, the memory architecture is often heirerarchical. Out-of-core techniques used between disk and processor memory can also be applied to the multiple layers within the memory heierarchy. In addition, we suspect that memory requirements for scientific applications will increase at least at the same pace as the technology for increasing memory capacity. The combination of market trends, memory architecture and the growing size of scientific applications suggests that the demand for efficient out-of-core techniques will increase rather than decrease.

References

[AVV98] Lars Arge, Darren Erik Vengroff, and Jeffrey Scott Vitter. External-memory algorithms for processing line segments in geographic information systems. *Algorithmica*, 1998.

[BCD97] P. Brezany, A. Choudhary, and M. Dang. Parallelization of irregular out-of-core applications for distributed-memory systems. *High-Performance Computing and Networking*, 1225:811–820, 1997.

[CDZ+97] C. Ceron, J. Dopazo, E. L. Zapata, J.M. Carazo, and O. Trelles. Parallel implementation of DNAml program on message-passing architectures. *Parallel Computing*, 24(5–6):701–716, June 1997.

[CHKM96] Robert Cypher, Alex Ho, Smaragda Konstantinidou, and Paul Messina. A quantitative study of parallel scientific applications with explicit communication. *Journal of Supercomputing*, 10(1):5–24, March 1996.

[CMA+97] Chialin Chang, Bongki Moon, Anurag Acharya, Carter Shock, Alan Sussman, and Joel Saltz. Titan: a high-performance remote-sensing database. In *Proceedings of the Thirteenth International Conference on Data Engineering*, pages 375–384, Birmingham, U.K., April 1997.

[DIS96] James Demmel, Melody Y. Ivory, and Sharon L. Smith. Modeling and identifying bottlenecks in EOSDIS. In *Proceedings of the Sixth Symposium on the Frontiers of Massively Parallel Computation*, pages 300–308. IEEE Computer Society Press, October 1996.

[DLY+98] G. Davis, L. Lau, R. Young, F. Duncalfe, and L. Brebber. Parallel run-length encoding (RLE) compression—reducing I/O in dynamic environmental simulations. *The International Journal of High Performance Computing Applications*, 12(4), Winter 1998. To appear in a Special Issue on I/O in Parallel Applications.

[DR95] E. F. D'Azevedo and C. H. Romine. EDONIO: Extended distributed object network I/O library. Technical Report ORNL/TM-12934, Oak Ridge National Laboratory, 1995.

[FMH+97] Renato Ferreira, Bongki Moon, Jim Humphries, Alan Sussman, Joel Saltz, Robert Miller, and Angelo Demarzo. The virtual microscope. In *American Medical Informatics Association, 1997 Annual Fall Symposium*, pages 449–453, Nashville, TN, October 1997.

[KBCH95] Ben Kobler, John Berbert, Parris Caulk, and P. C. Hariharan. Architecture and design of storage and data management for the NASA Earth Observing System Data and Information System (EOSDIS). In *Proceedings of the Fourteenth IEEE Symposium on Mass Storage Systems*, pages 65–76. IEEE Computer Society Press, September 1995.

[KKCB97] Meenakshi A. Kandaswamy, Mahmut T. Kandemir, Alok N. Choudhary, and David E. Bernholdt. Optimization and evaluation of Hartree-Fock application's I/O with PASSION. In *Proceedings of SC97: High Performance Networking and Computing*, San Jose, CA, November 1997. IEEE Computer Society Press.

[KKCB98] Meenakshi Kandaswamy, Mahmut Kandemir, Alok Choudhary, and David Bernholdt. An experimental study to analyze and optimize Hartree-Fock application's I/O with PASSION. *The International Journal of High Performance Computing Applications*, 12(4), Winter 1998. To appear in a Special Issue on I/O in Parallel Applications.

[Kot96] David Kotz. Applications of parallel I/O. Technical Report PCS-TR96-297, Dept. of Computer Science, Dartmouth College, October 1996. see suplement in PCS-TR98-XXX.

[Kot97] David Kotz. BibTeX bibliography file: Parallel I/O. Available on the WWW at `http://www.cs.dartmouth.edu/pario/bib/`, February 1997. Ninth Edition.

[LEG+97] P.M. Lyster, K. Ekers, J. Guo, M. Harber, D. Lamich, J.W. Larson, R. Lucchesi, R. Rood, S. Schubert, W. Sawyer, M. Sienkiewicz, A. da Silva, J. Stobie, L.L. Takacs, R. Todling, and J. Zero. Parallel computing at the NASA data assimilation office (DAO). In *Proceedings of SC97: High Performance Networking and Computing*, San Jose, CA, November 1997. IEEE Computer Society Press.

[LPJ98] P. Lockey, R. Proctor, and I. D. James. Characterization of I/O requirements in a massively parallel shelf sea model. *The International Journal of High Performance Computing Applications*, 12(3):320–332, Fall 1998.

[LSH98] J. Lepper, U. Schnell, and K.R.G. Hein. Parallelization of a simulation code for reactive flows on the Intel Paragon. *Computers and Mathematics with Applications*, 35(7):101–109, April 1998.

[MMD98] David Mackay, G. Mahinthakumar, and Ed D'Azevedo. A study of I/O in a parallel finite element groundwater transport code. *The International Journal of High Performance Computing Applications*, 12(3):307–319, Fall 1998.

[NFK98] Jarek Nieplocha, Ian Foster, and Rick Kendall. ChemIO: High-performance parallel I/O for computational chemistry applications. *The International Journal of High Performance Computing Applications*, 12(3):345–363, Fall 1998.

[OOVW96] Curtis Ober, Ron Oldfield, John VanDyke, and David Womble. Seismic imaging on massively parallel computers. Technical Report SAND96-1112, Sandia National Laboratories, April 1996.

[OWO98] Ron A. Oldfield, David E. Womble, and Curtis C. Ober. Efficient parallel I/O in seismic imaging. *The International Journal of High Performance Computing Applications*, 12(3):333–344, Fall 1998.

[PSS96] Yoonho Park, Ridgway Scott, and Stuart Sechrest. Virtual memory versus file interfaces for large, memory-intensive scientific applications. In *Proceedings of Supercomputing '96*. ACM Press and IEEE Computer Society Press, November 1996. Also available as UH Department of Computer Science Research Report UH-CH-96-7.

[RAN+93] D. A. Reed, R. A. Aydt, R. J. Noe, P. C. Roth, K. A. Shields, B. W. Schwartz, and L. F. Tavera. Scalable performance analysis: The Pablo performance analysis environment. In A. Skjellum, editor, *Proceedings of the Scalable Parallel Libraries Conference*, pages 104–113, Silver Spring, 1993.

[SCM+98] Carter T. Shock, Chialin Chang, Bongki Moon, Anurag Acharya, Larry Davis, Joel Saltz, and Alan Sussman. The design and evaluation of a high-performance earth science database. *Parallel Computing*, 24(1):65–89, January 1998.

[SR97] E. Smirni and D.A. Reed. Workload characterization of input/output intensive parallel applications. In *Proceedings of the Conference on Modelling Techniques and Tools for Computer Performance Evaluation*, volume 1245 of *Lecture Notes in Computer Science*, pages 169–180. Springer-Verlag, June 1997.

[SR98a] Huseyin Simitci and Daniel Reed. A comparison of logical and physical parallel I/O patterns. *The International Journal of High Performance Computing Applications*, 12(3):364–380, Fall 1998.

[SR98b] E. Smirni and D.A. Reed. Lessons from characterizing the input/output behavior of parallel scientific applications. *Performance Evaluation: An International Journal*, 33(1):27–44, June 1998.

[SW97] John Salmon and Michael Warren. Parallel out-of-core methods for N-body simulation. In *Proceedings of the Eighth SIAM Conference on Parallel Processing for Scientific Computing*, March 1997.

[TCB+96] Rajeev Thakur, Alok Choudhary, Rajesh Bordawekar, Sachin More, and Sivaramakrishna Kuditipudi. Passion: Optimized I/O for parallel applications. *IEEE Computer*, 29(6):70–78, June 1996.

[TLG98] Rajeev Thakur, Ewing Lusk, and William Gropp. I/O in parallel applications: The weakest link. *The International Journal of*

High Performance Computing Applications, 12(4), Winter 1998. To appear in a Special Issue on I/O in Parallel Applications.

[TSF+97] Michael Tobis, Chad Schafer, Ian Foster, Robert Jacob, and John Anderson. FOAM: Expanding the horizons of climate modeling. In *Proceedings of SC97: High Performance Networking and Computing*. IEEE Computer Society Press, November 1997.