Applications of Parallel I/O

David Kotz

Technical Report PCS-TR96-297
Release 1

Department of Computer Science
Dartmouth College
Hanover, NH 03755-3510
dfk@cs.dartmouth.edu

October 14, 1996

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, I list and describe many papers and web pages that describe scientific applications that use parallel I/O. While I do not go into depth about the characteristics of each application, I 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. See [Kot96] for a complete bibliography of parallel I/O.

This research was funded by NSF under grant number CCR-9404919, by NASA Ames under agreement number NCC 2-849.

I 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 me 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. I do not include papers about scientific kernels (LU factorization, matrix multiplication, sorting, FFT, and so forth).

- [HLDS95] They discuss a **weather code** that runs on the CM-5. The code writes a history file, dumping some data every timestep, and periodically a restart file. They found that CM-5 Fortran met their needs, although required huge buffers to get much scalability. They want to see a single, shared file-system image from all processors, have the file format be independent of processor count, use portable conventional interface, and have throughput scale with the number of computation processors.

- [JKH95] This paper is about a **weather code**. There's a bit about the parallel I/O issues. They periodically write a restart file, and they write out several types of data files. They write out the data in any order, with a little mini-header in each chunk that describes the chunk. I/O was not a significant percentage of their run time on either the CM5 or C90.

- [RHH95] **weather simulation code**

- [RW93, Rya91] A paper, and corresponding I/O-template code, about **aircraft simulation and computational fluid dynamics (Navier-Stokes flowfields)**. They describe their parallel implementation of the ARC3D code on the iPSC/860. A section of the paper considers I/O, which is to write out a large multidimensional matrix at each timestep. They found that it was actually faster to write to separate files because of congestion in the I/O nodes was hurting performance. They never got more than 2 MB/s, even so, on a system that should obtain 7-10 MB/s peak. The sample code tries to behave like a parallel ARC3D in terms of its output. It writes two files, one containing three three-dimensional matrices X, Y, and Z,

[1] `http://www.cs.dartmouth.edu/reports/abstracts/TR96-297.html`.

and the other containing the four-dimensional matrix Q. The matrices are spread over all the nodes, and each file is written in parallel by the processors.

- [CSWM95] They "discuss data production rates and their impact on the performance of scientific applications using parallel computers." They "present performance data for a **biomolecular simulation** of the enzyme, acetylcholinesterase, which uses the parallel molecular dynamics program EulerGROMOS. The actual production rates are compared against a typical time frame for results analysis where we show that the rate limiting step is the simulation, and that to overcome this will require improved output rates."

- [FAJL+95] "**Remotely sensed imagery** has been used for developing and validating various studies regarding land cover dynamics." They "develop a parallel version of [their] algorithm that is scalable in terms of both computation and I/O. Experimental results obtained show that a Thematic Mapper (TM) image (36 MB per band, 5 bands need to be corrected) can be handled in less than 4.3 minutes on a 32-node CM-5 machine, including I/O time."

- [MPP+95] They discuss "parallel processing of **Synthetic Aperture Radar (SAR) data**...," which is image data collected by satellite. They "parallelized the most compute-intensive SAR correlator phase of the Spaceborne Shuttle Imaging Radar-C/X-Band SAR (SIR-C/X-SAR) code, for the Intel Paragon." They "describe the data decomposition, the scalable high-performance I/O model, and the node-level optimizations which enable us to obtain efficient processing throughput."

- [OOVW96] This paper is about "**imaging of complex, oil-bearing geologies**, such as overthrusts and salt domes.... ...highly accurate techniques involve the solution of the wave equation and are characterized by large data sets and large computational demands. The portable code, Salvo, performs a wave-equation-based, 3-D, prestack, depth imaging and currently runs on the Intel Paragon, the Cray T3D and SGI Challenge series. It uses MPI for portability, and has sustained 22 Mflops/sec/proc (compiled FORTRAN) on the Intel Paragon." There are two pages about their I/O scheme, mostly regarding a calculation of the optimal balance between compute nodes and I/O nodes to achieve perfect overlap.

- [HRW+95] They discuss a **climate modeling** code, which does some out-of-core work to communicate data between time steps. They also dump a 'history' file every simulated day, and periodic checkpoint files. They are flexible about the layout of the history file, assuming postprocessing will clean it up. The I/O is not too much trouble on the Cray C90, where they get 350 MBps to the SSD for the out-of-core data. The history I/O is no problem. On distributed-memory machines with no SSD, out-of-core was impractical and the history file was only written once per simulated month. "The most significant weakness in the distributed-memory implementation is the treatment of I/O, [due to] file system maturity...."

- [BC93] A substantial part of this **structural-analysis** application was involved in I/O, moving substructures in and out of RAM.
- [Joh84] A paper about **three-dimensional wave-equation computations in seismic modeling.** They describe their need for large memory and fast paging and I/O in out-of-core solutions. They used 4-way parallel I/O. They needed to transfer a 3-d matrix in and out of memory by rows, columns, and vertical columns. Stored in block-structured form to improve locality on the disk.
- [Die90] An **out-of-core backpropagation** application that reads large files, sequentially, on CM2 with DataVault.

3 Characterizations of specific applications

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

- [AUB+96] They discuss four application programs from the areas of **satellite-data processing** and **linear algebra**. They tune each one of them by restructuring the program.
- [CACR95] A detailed characterization of three applications: **electron scattering, terrain rendering, and quantum chemistry**. They look at the volume of data moved, the timing of I/O, and the periodic nature of I/O. They do a little bit with the access patterns of data within each file. They found a *huge* variation in request sizes, amount of I/O, number of files, and so forth. Their primary conclusion is thus that file systems should be adaptable to different access patterns, preferably under control of the application.
- [SACR96] They study two applications (**electron scattering and computational fluid dynamics**) over several versions, using Pablo to capture the I/O activity. They thus watch as application developers improve the applications use of I/O modes and request sizes. Both

applications move through three phases: initialization, computation (with out-of-core I/O or checkpointing I/O), and output. They found it necessary to tune the I/O request sizes to match the parameters of the I/O system. In the initial versions, the code used small read and write requests, which were (according to the developers) the "easiest and most natural implementation for their I/O." They restructured the I/O to make bigger requests, which better matched the capabilities of Intel PFS. They conclude that asynchronous and collective operations are imperative. They would like to see a file system that can adapt dynamically to adjust its policies to the apparent access patterns. Automatic request aggregation of some kind seems like a good idea; of course, that is one feature of a buffer cache.

- [Are91] They use a **genome-sequence comparison** program to study the performance of Intel CFS. The application reads in a huge file of records, each a genome sequence, and compares each sequence against a given sequence.

- [BW96] They characterize four parallel applications: sort, matrix multiply, **seismic migration**, and video server, in terms of their I/O activity. They found results that are consistent with [KN95], in that they also found lots of small data requests, some large data requests, significant file sharing and interprocess locality.

- [Bel88] They describe a specialized database system for **particle physics** codes. The paper is valuable for its description of access patterns and subsequent file access requirements. Particle-in-cell codes iterate over timesteps, updating the position of each particle, and then the characteristics of each cell in the grid. Particles may move from cell to cell. Each particle update needs itself and nearby gridcell data. The whole dataset is too big for memory, and each timestep must be stored on disk for later analysis anyway. Regular file systems are inadequate: a specialized DBMS is more appropriate. Characteristics are needed by their application class: multidimensional access (by particle type or by location, i.e., multiple views of the data), coordination between grid and particle data, coordination between processors, coordinated access to meta-data, inverted files, horizontal clustering, large blocking of data, asynchronous I/O, array data, complicated joins, and prefetching according to user-prespecified order. Note that many of these things can be provided by a file system, but that most are hard to come by in typical file systems, if not impossible. Many of these features are generalizable to other applications.

- [CHKM93] They look at many parallel applications, although there is little mention of I/O. They average 1207 Bytes/MFlop. Some of the applications do I/O throughout their run (2400 Bytes/MFlop avg.), while others only do I/O at the beginning or end (14 Bytes/MFlop avg). But I/O is bursty, so larger bandwidths are suggested. The applications are parallel programs running on Intel Delta, nCUBE/1, or nCUBE/2; and are in C, FORTRAN, or both.

- [KGF93, KFG94] They store a sparse, multidimensional **radio-astronomy** data set as a set of tagged data values, i.e., as a set of tuples, each with several keys and a data value. They use a "PLOP" format to partition each dimension into slices, so that each intersection of the slices forms a bucket. They decide on the splits based on a preliminary statistical survey of the data. Bucket overflow is handled by chaining. Then, they evaluate various kinds of queries, i.e., multidimensional range queries, for their performance. In this workload queries (reads) are much more common than updates (writes).

- [RB90] Using five applications from the **Perfect benchmark suite**, they studied both implicit (paging) and explicit (file) I/O activity. They found that the paging activity was relatively small and that sequential access to VM was common. All access to files was sequential, though this may be due to the programmer's belief that the file system is sequential. Buffered I/O would help to make transfers bigger and more efficient, but there wasn't enough rereferencing to make caching useful.

- [TLG95] They describe an **astrophysics** application, which "focuses on the study of the highly turbulent convective layers of late-type stars, like the sun, in which turbulent mixing plays a fundamental role in the redistribution of many physical ingredients of the star...." Its I/O usage is straightforward: it just writes its matrices every few timesteps. The application writes whole matrices; the OS sees request sizes that are more a factor of the Chameleon library than of the application. Most of the I/O itself is not implemented in parallel, because they used UniTree on the SP, and because the Chameleon library sequentializes this kind of I/O through one node.

- [TGL96a, TGL96b] They use an **astrophysics** application, which simulates the gravitational collapse of self-gravitating gaseous clouds, to compare the file systems of the Intel Paragon and the IBM SP-2.

- [Woo93] This paper is interesting for its impressive usage of RAIDs and parallel networks to support **scientific visualization**. In particular, the proposed Gigawall (a 10-foot by 6-foot gigapixel-per-second display) is run by 24 SGI processors and 32 9-disk RAIDs, connected to an MPP of some kind through an ATM switch. They propose 512 GBytes of storage, playable at 450 MBytes per second, for 19 minutes of animation.

4 Characterizations of the workload

These papers characterize the I/O workload of a production parallel computer, but do not discuss specific applications.

The CHARISMA project[2] traced the workload of two parallel machines running production scientific applications, and then characterized the workload in detail.

- [KN94a, KN94b, KN95] Intel iPSC/860 at NASA Ames
- [PEK+94, PEK+95] Thinking Machines CM-5 at NCSA
- [NKP+95, NKP+96] both

5 Other surveys of applications using parallel I/O

There are a few other papers that have an extensive survey of several parallel scientific applications using parallel I/O.

- [dC94] A nice summary of grand-challenge and other applications, and their I/O needs.
- [Poo94] This paper from the Scalable I/O Initiative describes several applications and their I/O needs. They focus on four categories of I/O needs: input, output, checkpoint, and virtual memory ("out-of-core" scratch space). Not all types are significant in all applications. (Two groups mention databases and the need to perform computationally complex queries.) Large input is typically raw data (seismic soundings, astronomical observations, satellite remote sensing, weather information). Sometimes there are real-time constraints. Output is often periodic, e.g., the state of the system every few timesteps; typically the volume would increase along with I/O capacity and bandwidth. Checkpointing is a common request; preferably allowing application to choose what and when to checkpoint, and definitely including the

[2]http://www.cs.dartmouth.edu/research/charisma.html.

state of files. Many kinds of out-of-core: 1) temp files between passes (often written and read sequentially), 2) regular patterns like FFT, matrix transpose, solvers, and single-pass read/compute/write, 3) random access, e.g., to precomputed tables of integrals. Distinct differences in the ways people choose to divide data into files; sometimes all in one huge file, sometimes many "small" files (e.g., one per processor, one per timestep, one per region, etc.). Important: overlap of computation and I/O, independent access by individual processors. Not always important: ordering of records read or written by different processors, exposing the I/O model to the application writer. Units of I/O seem to be either (sub)matrices (1–5 dimensions) or items in a collection of objects (100–10000 bytes each). Data set sizes varied up to 1 TB; bandwidth needs varied up to 1 GB/s.

- [GGL93] They give a useful overview of the I/O requirements of many applications codes, in terms of input, output, scratch files, debugging, and checkpointing.
- [Moo95] They briefly describe the I/O requirements for four production **oceanography** programs running at Oregon State University. The applications all rely exclusively on array-oriented, sequential file operations. Persistent files are used for checkpointing and movie making, while temporary files are used to store out-of-core data.
- I have collected a few anecdotes.[3]
- I have collected several example codes.[4]

References

[Are91] James W. Arendt. Parallel genome sequence comparison using a concurrent file system. Technical Report UIUCDCS-R-91-1674, University of Illinois at Urbana-Champaign, 1991.

[AUB+96] Anurag Acharya, Mustafa Uysal, Robert Bennett, Assaf Mendelson, Michael Beynon, Jeffrey K. Hollingsworth, Joel Saltz, and Alan Sussman. Tuning the performance of I/O intensive parallel applications. In *Fourth Workshop on Input/Output in Parallel and Distributed Systems*, pages 15–27, Philadelphia, May 1996.

[BC93] P. E. Bjørstad and J. Cook. Large scale structural analysis on massively parallel computers. In *Linear Algebra for Large Scale and Real-Time Applications*, pages 3–11. Kluwer Academic Publishers, 1993. ftp from ftp.ii.uib.no in `pub/tech_reports/mpp_sestra.ps.Z`.

[3] `http://www.cs.dartmouth.edu/cs_archive/pario/anecdotes.html`

[4] `http://www.cs.dartmouth.edu/cs_archive/pario/examples.html`

[Bel88] Jean L. Bell. A specialized data management system for parallel execution of particle physics codes. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 277–285, 1988.

[BW96] Sandra Johnson Baylor and C. Eric Wu. Parallel I/O workload characteristics using Vesta. In Ravi Jain, John Werth, and James C. Browne, editors, *Input/Output in Parallel and Distributed Computer Systems*, volume 362 of *The Kluwer International Series in Engineering and Computer Science*, chapter 7, pages 167–185. Kluwer Academic Publishers, 1996.

[CACR95] Phyllis E. Crandall, Ruth A. Aydt, Andrew A. Chien, and Daniel A. Reed. Input/output characteristics of scalable parallel applications. In *Proceedings of Supercomputing '95*, December 1995.

[CHKM93] R. Cypher, A. Ho, S. Konstantinidou, and P. Messina. Architectural requirements of parallel scientific applications with explicit communication. In *Proceedings of the 20th Annual International Symposium on Computer Architecture*, pages 2–13, 1993.

[CSWM95] Terry W. Clark, L. Ridgway Scott, Stanislaw Wlodek, and J. Andrew McCammon. I/O limitations in parallel molecular dynamics. In *Proceedings of Supercomputing '95*, 1995.

[dC94] Juan Miguel del Rosario and Alok Choudhary. High performance I/O for parallel computers: Problems and prospects. *IEEE Computer*, 27(3):59–68, March 1994.

[Die90] Carl Diegert. Out-of-core backpropagation. In *International Joint Conference on Neural Networks*, volume 2, pages 97–103, 1990.

[FAJL+95] Hassan Fallah-Adl, Joseph JáJá, Shunlin Liang, Yoram J. Kaufman, and John Townshend. Efficient algorithms for atmospheric correction of remotely sensed data. In *Proceedings of Supercomputing '95*, 1995.

[GGL93] N. Galbreath, W. Gropp, and D. Levine. Applications-driven parallel I/O. In *Proceedings of Supercomputing '93*, pages 462–471, 1993.

[HLDS95] Steven W. Hammond, Richard D. Loft, John M. Dennis, and Rochard K. Sato. Implementation and performance issues of a massively parallel atmospheric model. *Parallel Computing*, 21:1593–1619, 1995.

[HRW+95] James J. Hack, James M. Rosinski, David L. Williamson, Byron A. Boville, and John E. Truesdale. Computational design of the NCAR community climate model. *Parallel Computing*, 21:1545–1569, 1995.

[JKH95] Philip W. Jones, Christopher L. Kerr, and Richard S. Hemler. Practical considerations in development of a parallel SKYHI general circulation model. *Parallel Computing*, 21:1677–1694, 1995.

[Joh84] Olin G. Johnson. Three-dimensional wave equation computations on vector computers. *Proceedings of the IEEE*, 72(1):90–95, January 1984.

[KFG94] John F. Karpovich, James C. French, and Andrew S. Grimshaw. High performance access to radio astronomy data: A case study. In *Proceedings of the 7th International*

Working Conference on Scientific and Statistical Database Management, September 1994. Also available as Univ. of Virginia TR CS-94-25.

[KGF93] John F. Karpovich, Andrew S. Grimshaw, and James C. French. Breaking the I/O bottleneck at the National Radio Astronomy Observatory (NRAO). Technical Report CS-94-37, University of Virginia, August 1993.

[KN94a] David Kotz and Nils Nieuwejaar. Dynamic file-access characteristics of a production parallel scientific workload. Technical Report PCS-TR94-211, Dept. of Math and Computer Science, Dartmouth College, April 1994. Revised May 11, 1994.

[KN94b] David Kotz and Nils Nieuwejaar. Dynamic file-access characteristics of a production parallel scientific workload. In *Proceedings of Supercomputing '94*, pages 640–649, November 1994.

[KN95] David Kotz and Nils Nieuwejaar. File-system workload on a scientific multiprocessor. *IEEE Parallel and Distributed Technology*, pages 51–60, Spring 1995.

[Kot96] David Kotz. BibTeX bibliography file: Parallel I/O. Available for ftp from cs.dartmouth.edu in `pub/pario/pario.bib`, and on the WWW at `http://www.cs.dartmouth.edu/cs_archive/pario/bib.html`, May 1996. Eighth Edition.

[Moo95] Jason A. Moore. Parallel I/O requirements of four oceanography applications. Technical Report 95-80-1, Oregon State University, January 1995.

[MPP+95] Craig Miller, David G. Payne, Thanh N. Phung, Herb Siegel, and Roy Williams. Parallel processing of spaceborne imaging radar data. In *Proceedings of Supercomputing '95*, 1995.

[NKP+95] Nils Nieuwejaar, David Kotz, Apratim Purakayastha, Carla Schlatter Ellis, and Michael Best. File-access characteristics of parallel scientific workloads. Technical Report PCS-TR95-263, Dept. of Computer Science, Dartmouth College, August 1995. To appear in IEEE TPDS.

[NKP+96] Nils Nieuwejaar, David Kotz, Apratim Purakayastha, Carla Schlatter Ellis, and Michael Best. File-access characteristics of parallel scientific workloads. *IEEE Transactions on Parallel and Distributed Systems*, 7(10):1075–1089, October 1996.

[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.

[PEK+94] Apratim Purakayastha, Carla Schlatter Ellis, David Kotz, Nils Nieuwejaar, and Michael Best. Characterizing parallel file-access patterns on a large-scale multiprocessor. Technical Report CS-1994-33, Dept. of Computer Science, Duke University, October 1994.

[PEK+95] Apratim Purakayastha, Carla Schlatter Ellis, David Kotz, Nils Nieuwejaar, and Michael Best. Characterizing parallel file-access patterns on a large-scale multiprocessor. In *Proceedings of the Ninth International Parallel Processing Symposium*, pages 165–172, April 1995.

[Poo94] James T. Poole. Preliminary survey of I/O intensive applications. Technical Report CCSF-38, Scalable I/O Initiative, Caltech Concurrent Supercomputing Facilities, Caltech, 1994.

[RB90] A. L. Narasimha Reddy and Prithviraj Banerjee. A study of I/O behavior of Perfect benchmarks on a multiprocessor. In *Proceedings of the 17th Annual International Symposium on Computer Architecture*, pages 312–321, 1990.

[RHH95] Bernardo Rodriguez, Leslie Hart, and Tom Henderson. Programming regular grid-based weather simulation models for portable and fast execution. In *Proceedings of the 1995 International Conference on Parallel Processing*, pages III:51–59, August 1995.

[RW93] J. S. Ryan and S. K. Weeratunga. Parallel computation of 3-D Navier-Stokes flowfields for supersonic vehicles. In *31st Aerospace Sciences Meeting and Exhibit*, Reno, NV, 1993. AIAA Paper 93-0064.

[Rya91] Steve Ryan. CFS workload demonstration code. WWW ftp://ftp.cs.dartmouth.edu/pub/pario/examples/CFS3D.tar.Z, July 1991. A simple program demonstrating CFS usage for ARC3D-like applications.

[SACR96] Evgenia Smirni, Ruth A. Aydt, Andrew A. Chien, and Daniel A. Reed. I/O requirements of scientific applications: An evolutionary view. In *Proceedings of the Fifth IEEE International Symposium on High Performance Distributed Computing*, pages 49–59, 1996.

[TGL96a] Rajeev Thakur, William Gropp, and Ewing Lusk. An experimental evaluation of the parallel I/O systems of the IBM SP and Intel Paragon using a production application. Technical Report MCS-P569–0296, Argonne National Laboratory, February 1996.

[TGL96b] Rajeev Thakur, William Gropp, and Ewing Lusk. An experimental evaluation of the parallel I/O systems of the IBM SP and Intel Paragon using a production application. In *Proceedings of the Third International Conference of the Austrian Center for Parallel Computation (ACPC)*, volume 1127 of *Lecture Notes in Computer Science*, pages 24–35. Springer-Verlag, September 1996.

[TLG95] Rajeev Thakur, Ewing Lusk, and William Gropp. I/O characterization of a portable astrophysics application on the IBM SP and Intel Paragon. Technical Report MCS-P534-0895, Argonne National Laboratory, August 1995. Revised October 1995.

[Woo93] Paul R. Woodward. Interactive scientific visualization of fluid flow. *IEEE Computer*, 26(10):13–25, October 1993.