A Case Study of Scientific Application I/O Behavior

Barbara K. Pasquale
George C. Polyzos

Sequoia 2000 Technical Report #93/36

December 1993

Computer Systems Laboratory
Department of Computer Science and Engineering
University of California, San Diego
La Jolla, CA 90293-0114, USA

A Case Study of Scientific Application I/O Behavior

Barbara K. Pasquale and George C. Polyzos

Computer Systems Laboratory
Department of Computer Science and Engineering
University of California, San Diego
La Jolla, CA 90293-0114, USA

ABSTRACT

The growing number of I/O intensive, scientific applications is focusing attention on the need to improve the support for and design of supercomputer I/O systems. Very little is known about the individual I/O requirements of these applications, but the combined effects of an I/O intensive workload can create bottlenecks, which seriously degrade overall system performance. Understanding the dynamic I/O demands of scientific applications can provide insight into the policies and mechanisms needed to prevent potential I/O bottlenecks in future high performance systems. This paper is a case study of one scientific application's dynamic I/O behavior. Results from this study show that with appropriate measurement tools, dynamic I/O behavior can be characterized and regular, recurring patterns of I/O activity can be isolated.

Key Words: empirical I/O behavior, supercomputer applications

Barbara Pasquale can be reached at bittel@cs.ucsd.edu, +1-619-534-8604, +1-619-534-7029 (Fax).
George Polyzos can be reached at polyzos@cs.ucsd.edu, +1-619-534-3508, +1-619-534-7029 (Fax).

1. Introduction

Today's high performance computer systems provide users with immense computational power to facilitate research across a wide spectrum of scientific disciplines. However, coupled with the ability to perform GFLOPS computations, is the need to access and produce massive amounts of data. The sheer volume of this data and the need to store, organize, access, distribute and visualize this data, intensifies I/O demands within the local system and communication requirements across networks [1]. Considering the disparity between the speed of the CPU and the speed of its attached I/O devices and the increase in the number of I/O intensive, scientific applications, I/O performance may well be the limiting factor in high performance computing [2,3]. In order to maintain well-balanced systems, it is necessary to establish a thorough understanding of the I/O behavior of scientific applications, and use this knowledge to improve application I/O support and provide requirements for the design of new supercomputer I/O systems.

The research presented in this paper describes our preliminary work in analyzing scientific application I/O behavior. We obtained a common climate research analysis application and collected detailed information on the application's run-time I/O activity. From the resulting trace log of I/O events, we produced detailed profiles characterizing the I/O activity over time. In the sections that follow, we describe our methodology for analyzing scientific application I/O, provide background information on the application and the measurement tool, and then present the results of our I/O analysis of the application on the San Diego Supercomputer Center's Cray Y-MP8/864.

2. Prior Research

Emerging research in the area of scientific application I/O indicates that scientific applications have a high degree of regularity in I/O resource usage and that this regularity can be exploited to improve I/O performance. Miller and Katz [4] characterized the I/O behavior of seven computational fluid dynamics and climate applications to study the effects of proposed software buffer sizes and policies to mitigate the effects of peak I/O bandwidth. Lim and Condry [5] analyzed the I/O behavior of four chemistry applications and the Perfect Club benchmark suite to determine how these applications could better exploit gigabit networks. Both studies revealed that applications exhibit specific phases of I/O activity and within each phase, there are distinct, cyclical patterns of I/O. They also observed that file accesses were sequential and that data request sizes remained constant within the phases of I/O activity.

A previous workload characterization study of the San Diego Supercomputer Center's Cray Y-MP8/864 [6], concluded that a small number of private user applications consumed a large portion of I/O resources. Of the 3,366 distinctly named private user applications identified, the individual executions of 50 distinct applications were responsible for 71% of total system bytes transferred and only 9.5% of total system CPU time during a one-month period. Average I/O rates for these 50 I/O intensive applications ranged from 7MB/sec. up to 132MB/sec. Detailed analysis of half of these applications (i.e., those executed more than ten times during the one-month observation period), revealed a high degree of regularity in the resource usage of the individual applications. In particular, I/O demands, in terms of the total number of bytes transferred and the total number of logical I/O requests, showed considerable correlation with total CPU time consumed. We also observed that individual applications exhibited (1) both "short" (i.e., on the order of seconds) and "long" (i.e., on the order of minutes) CPU execution times, (2) fairly stable, average logical I/O rates across different execution time periods, and (3) highly consistent in-core memory usage.

3. Methodology

System accounting data, like that used in [6], is inadequate for characterizing dynamic I/O behavior. With only aggregate resource usage totals and average I/O rates, it is impossible to draw any conclusions as to how I/O resources are utilized over time. Therefore, it was first necessary to obtain an appropriate measurement tool, develop programs to reduce collected data, formulate procedures for data analysis, and experiment with a suitable application. Below we provide a brief overview of our experimental methodology.

The performance measurement tool we used is IPS-2. IPS-2 is a software performance measurement system developed at the University of Wisconsin-Madison [7] for the purpose of monitoring and analyzing parallel and distributed programs. IPS-2 allows for the collection of performance data at five levels of program abstraction: program, machine, process, procedure, and primitive activity. Performance data includes execution times (CPU, wall clock), file I/O activity (operations, bytes), wait times (I/O, barrier, semaphore), etc. Since performance data is collected from special software instrumentation within the

operating system kernel and run-time I/O library, the application's source code does not have to be modified in order to obtain measurement data. However, to enable this performance measurement, the source code must be recompiled with the special IPS option. Then as the program executes, trace records are automatically collected, detailing the type and time of program events.

The application selected for analysis is called R. R is an analysis application used in the Climate Research Division at Scripps Institution of Oceanography. The purpose of R is to compare small observational data sets from the National Meteorological Center's (NMC) large-scale, global climate data and weather station site data with massive, model output data sets from the Regional Atmospheric Modeling System (RAMS)[8]. The R analysis application is currently used for the verification of the fine-grained RAMS climate model over the western United States.

R represents a common type of analysis application that utilizes large input files, has significant I/O demands, and produces large output files. The manageable size and execution time of R allowed us to consider the application as a "black box," until the final stages of analysis. This was a conscious decision on our part because we did not want to be unnecessarily biased when interpreting R's dynamic I/O behavior. In the end, this early decision yielded two important results: (1) we developed a hierarchical set of data reduction and analysis tools to extract varying levels of I/O detail and (2) we now have an understanding of what application specific questions need to be answered to focus and direct our analysis efforts on key I/O activities. (This will save immeasurable time and resources when analyzing large Cray applications which execute for hours.)

After recompiling and executing R with IPS-2, we obtain large trace files containing records detailing the time and type of individual I/O events, I/O transfer sizes, and file descriptor numbers. For each recorded I/O event, two microsecond resolution timestamps are given. One timestamp represents the wallclock event time (real time) and the other represents the process event time (CPU or virtual time). We restrict our analysis to process event time or virtual time I/O activity. Considering I/O activity with respect to virtual or CPU time reveals the I/O activity as embodied in the source code, eliminating the effects of the system workload. Integrating I/O events over one second intervals of virtual or CPU time, we construct time series of the total number of I/O transfers performed and the total number of bytes transferred per unit time. At this resolution we can observe the general patterns of recurring I/O activity. We also produce microsecond resolution time series of I/O activity. At this resolution we can observe detailed I/O activity in terms of individual file accesses, transfers sizes, and times between I/O events. We call these time series the dynamic I/O profile of the application. Together, these profiles provide a detailed characterization of the application's I/O activity: distinct phases of I/O processing, cyclical patterns of file accesses, and interevent times of I/O transfers. Thus, without any prior knowledge of code structure and the computational behavior of R, we can gain a basic understanding of this application and detail its characteristic I/O behavior.

4. I/O Analysis of R

Our analysis of R involved both a dynamic and static I/O characterization. A dynamic characterization is a detailed description of the times and types of program events which occur during application execution. A static characterization is a high-level description of resource usage in terms of aggregate totals for the entire execution period. For example, static characteristics may include total time executing in system (or user) mode, total number of I/O requests performed, total number of bytes physically transferred, and total number of main memory bytes used. In the discussion that follows, we first present the static characterization for R and then describe the dynamic analysis.

4.1 Static Characterization

Table 1 below describes the data set we used with the R application. The data set consisted of five input files, totaling 373 MBytes, and produces two output files, totaling 18 MBytes. The small ASCII input files are used to set initial conditions for the analysis process and the large binary input files contain the climate data to be analyzed

Table 1: Data Files

File Type	Cray Y-MP File Number	Cray Y-MP Size in Bytes
ASCII Input	3	70,000
	9	47,288
Binary Input	4	324,677,632
	5	15,122,432
	6	33,140,736
Binary Output	7	13,299,712
	8	4,788,244

We monitored R as it executed on San Diego Supercomputer Center's Cray Y-MP8/864. The YMP has a 6 ns clock, 8 CPUs, 512 MB (or 64 M 64-bit words) of memory, and a peak execution rate of 2.67 GFLOPS. Since both CPU time and main memory requirements of R exceed the maximum usage for interactive jobs (20 CPU minutes and 4MWords memory), the application was run in batch mode. In batch mode, submitted jobs are organized into queues based on needed CPU time, main memory space, local disk space, and run-time priority level, and then scheduled to run based on the current job mix and availability of system resources. R was submitted to a standard batch queue, providing a maximum of 6000 CPU minutes, 8 MWords of main memory, 85 MWords of local disk space, and a high run-time priority level. Table 2 below gives the corresponding execution times for R.

Total turn-around time for R was approximately 32 minutes. This turn-around time includes the time spent waiting in the batch queue prior to initial service. The wall clock time or time between initial service and program conclusion was approximately 21 minutes. Wall clock time includes the time during which the program was running as well as the time during which the program was in a "non-runnable" state (i.e., swapped out of memory, waiting for I/O to complete). Process CPU time or the time during which the program was actively using the CPU was approximately 14 minutes. It is not surprising to see a large difference between CPU time and wallclock time on the Cray. During its execution R was one of the approximately 12 to 15 jobs kept in main memory for the purpose of efficiently time-sharing the CPUs.

Table 2: Execution Times

Execution Times	Cray Y-MP (seconds)	Cray Y-MP (minutes)
Turnaround	1,934.00	32
Wallclock	1,258.57	21
Process CPU	846.47	14

Table 3 below gives the I/O resource usage statistics for R. In 14 CPU time minutes, 3,919 I/Os occurred, transferring a total of 768 MBytes of data. Considering the relative sizes of the data files given in Table 1, it is not surprising to note that file reads are the main component of I/O activity. In fact, read I/O represents 97.6% of all bytes transferred and of all I/O transfers performed. It is interesting to note, however, that the total number of bytes transferred for read I/O purposes is approximately twice the size of the input data set and the total number of bytes transferred for write I/O purposes is equal to the size of the output data set. This gives a ratio of approximately 42 bytes read for every 1 byte written. Based on the measured values for I/O transfers performed and bytes transferred, we calculated the average I/O rates appearing in Table 3. In the absence of more detailed data, these average rates can be used to assess the relative intensity of I/O. However, it is more than likely that there are large peaks of I/O load which fluctuate over time.

Table 3: I/O Resource Usage Statistics

I/O Usage	Cray Y-MP		
	Total I/O	Read I/O	Write I/O
Number of I/O Transfers	3,919	3,826	93
Number of Bytes Transferred	768,875,048	750,787,112	18,087,936
Average Number of I/O Transfers per CPU Second	5	5	<1
Average Number of Bytes Transferred per CPU Second	908,323	886,955	21,368
Average Number of Bytes per I/O Transfer	196,192	196,233	194,494

4.2 Dynamic Characterization

Monitoring R with IPS-2 at the program level, we obtained a trace file of records describing each I/O event that occurred. The format of an I/O event record is: wall clock time (usecs.), process time (usecs.), I/O event (e.g., read, write), file descriptor number, and bytes transferred. The trace file also included records for I/O delay and preread events, however, this data was not used since our focus is on virtual time I/O behavior. Read and write I/O events and their corresponding bytes transferred values were integrated over one second intervals and one microsecond intervals of wall clock and process CPU time. We then constructed time series of the total number of I/O transfers performed and the total number of bytes transferred per unit time to obtain the dynamic I/O profiles.

Although our focus is on the virtual or CPU time dynamic I/O activity, we had a general interest in observing R's I/O activity in real time. The complete wallclock time profile is shown in Figure 1. We observe that R's I/O activity is not constant over time, but occurs in bursts, reaching a maximum of 50 I/O transfers per second. Another interesting observation is that the frequency and intensity of these bursts vary over time, showing two different patterns of I/O behavior. This wallclock I/O profile can be considered the workload dependent I/O profile and can only capture dynamic I/O behavior with respect to the job mix and available system resources at the time of execution. The characteristic I/O behavior of the application, as embodied in the source code, must be derived from the virtual or CPU time profile.

The virtual or CPU time profile with respect to I/O transfers for R is shown in Figure 2. At the second resolution, this profile clearly depicts the characteristic I/O behavior of R. R has two distinct phases of processing: (1) a short, initial phase with frequently occurring, repeating bursts of I/O activity and (2) a longer, main phase, during which I/O activity repeats at larger intervals and is composed of two individual periods of I/O. Although both phases show a regular and repeating pattern of I/O activity, there is some variation between I/O bursts in each phase. Between theses two phases, there is a unique spike of I/O activity. The corresponding dynamic profiles with respect to bytes transferred is shown in Figure 3. These two profiles highlight the strong correlation between the number of I/Os performed and bytes the number of transferred during each period of I/O activity. Inspection of the individual trace records revealed that 196,608 bytes was the standard unit of data transfer. On the Cray Y-MP this is equivalent to 48 blocks of 512 64-bit words.

A breakdown of the I/O resource usage for each of these two phases is given in Table 4 below. The first three minutes of R's execution, or the initial phase, account for 50% of total application I/O activity and represents 21.5% of total execution time.Write I/O activity during this phase only represents 1% of all initial phase I/O. The main phase accounts for 74% of total application write I/O activity and 50% of total application read I/O activity. However, within the main phase itself, write I/O still represents only 3% of main phase I/O activity. It is also interesting to note that the total read I/O activity in this longer, main phase is approximately equal to that in the short, initial phase.

Table 4: I/O Resource Usage Statistics

	Initial Phase		Main Phase	
	Read I/O	Write I/O	Read I/O	Write I/O
Number of I/O Transfers	1,937	25	1,889	68
Number of Bytes Transferred	374,827,560	4,788,224	371,171,328	13,299,712

To determine the individual contribution of each data file to total application I/O activity, we analyzed file I/O activity by processing phase. Table 5 below describes the active files in terms of read and write accesses for both the initial and main processing phases. Of the seven data files used, five are read only, one is write only, and one is both read and write. Only the input files 4, 5, and 6 are active throughout program execution and in each phase they appear to be read once. It is also interesting to note that the two output files correspond to a particular phase of processing: file 8 for the initial phase and file 7 for the main phase. The two ASCII input files are used exclusively in the initial phase.

Table 5: File I/O Statistics

	Initial Phase		Main Phase	
	Read I/O	Write I/O	Read I/O	Write I/O
Input File 3 Bytes Transferred	70,000	--	--	--
Input File 9 Bytes Transferred	47,288	--	--	--
Input File 4 Bytes Transferred	324,677,632	--	324,677,632	--
Input File 5 Bytes Transferred	15,712,256	--	14,532,608	--
Input File 6 Bytes Transferred	34,320,384	--	31,961,088	--
Output File 7 Bytes Transferred	--	--	--	13,299,712
Output File 8 Bytes Transferred	4,788,224	4,788,224	--	--

Initial Phase

As the dynamic profile shows (Figure 2), the repeating pattern of I/O activity in this phase is a single, large burst of I/O. There are 30 such cycles of I/O activity. Each cycle repeats approximately every 6 seconds. The detailed view of the initial phase I/O cycle is shown in Figure 4. After four seconds of computation, there is a two-second burst of 65 read I/Os, transferring 12.5 MB of data. In the first second, 55 read I/Os are performed, transferring 10.8 MB. In the subsequent second, 9 read I/Os are performed, transferring 1.7 MB. After four seconds of computation the pattern repeats. The last period of I/O activity in this initial phase is shown in Figure 5. Recall from the CPU time profile in Figure 2 that this I/O activity is unique and seems to mark the transition between processing phases. After four seconds of computation, there is again a two-second burst of I/O activity. However, this burst contains both read and write I/Os, totaling 114 I/Os and transferring 22 MB of data. In the first second, 16 read I/Os are performed, transferring 3.1 MB, and in the following second, both read and write I/O activity is performed. At this time, 73 read I/Os are performed, transferring 14.1 MB of data and 25 write I/Os are performed, transferring 4.7 MB of data

The microsecond resolution of the initial phase I/O cycle is shown in Figure 6. After 0.6 seconds, the first period of I/O begins. During the next 2,097 microseconds, 3 read I/Os to file 5 are performed followed by 6 read I/Os to file 6. The number of bytes transferred in each read I/O is exactly 196,608. Computation resumes for 5.21 seconds and then the second period of I/O begins. During this 279,389 microsecond period, all 55 read I/Os are to file 4, each transferring 196,608 bytes of data. The entire cycle then repeats.

Considering all 30 initial phase cycles at this level, we observed the following characteristics. In the period of read I/O to files 5 and 6, only one transfer completed within a given microsecond and time between successive transfers averaged 277.19 microseconds. The first computation phase follows the reads to files 5 and 6 and lasts for an average of 647431 microseconds. During the second period of read I/O to file 4, only one transfer completed within a given microsecond. As seen in Figure 6 (b and c), the time between successive transfers averaged 5082.01 microseconds. Following the file 4 reads, the second computation phase begins and lasts for an average of 5.13 seconds. Statistics summarizing interevent times and computation periods are given in Appendix A, Table A1.

Main Phase

The dynamic profile in Figure 2, shows that the repeating pattern of I/O activity in this phase consists of two separate periods: a short burst of low I/O activity and a short burst of high I/O activity. There are 30 such distinct cycles which repeat approximately every 22 seconds. The details of the main phase I/O cycle is shown in Figure 7. After 16 seconds of computation, 8 read I/Os, transferring 1.5 MB of data and 3 write I/Os, transferring 0.5 MB data are performed in the same one-second period of time. Then, following 4 seconds of computation, there is one second of read I/O, with 55 I/Os transferring 10.08 MB of data. After 16 seconds of computation the pattern repeats.

The microsecond resolution of the initial phase I/O cycle is shown in Figure 8. After 16.68 seconds of computation, two write I/Os, each transferring 196,608 bytes of data, are performed to file 7 in 183 microseconds. Following a period of 4,217 microseconds, 3 read I/Os to file 5 are performed followed by 6 read I/Os to file 6 in 2095 microseconds. The number of bytes transferred in each read I/O is also 196,608. Computation resumes for 5.13 seconds and then the second period of I/O begins. During this 273,200 microsecond period, all 55 read I/Os are to file 4, each transferring 196,608 bytes of data. The cycle then repeats.

Considering all 30 main phase cycles, we observed many characteristics similar to those in the initial phase. In the periods of write I/O to file 7 and read I/O to files 5 and 6, only one transfer completed within a given microsecond. The average interevent time for the write I/Os is 186.20 microseconds and for the file 5 and 6 read I/Os it is 279.15 microseconds. There is a brief period of computation after the write I/O to file 7 lasting an average of 4277 microsecond. The next computation phase (i.e, the first computation phase) follows the read I/O to files 5 and 6 and lasts an average of 5.16 seconds. The second period of read I/O to file 4 is consistent with the corresponding period in the initial phase, with an average interevent time of 5107.55 microseconds. The second computation phase follows the read I/O to file 4 and lasts for an average of 16.70 seconds. Statistics summarizing interevent times and computation periods are given in Appendix A, Table A2.

5. R's Dynamic Profile and Functional Characterization

From the physical resource usage point of view, our analysis successfully provided us with a detailed characterization of R's dynamic I/O behavior; a summary of this behavior is given below in Table 6. Comparing the initial and main phases of R, the similarities and differences can be readily noted. Excluding the final I/O activity in the initial phase, there is an equal number of I/O cycles. Read I/O activity and the first computation period are identical for both phases. Each phase produces a separate output file; in the initial phase, the output file is written once, at the end of the phase whereas in the main phase writes to the output file occur in every cycle. To understand the reasons behind this behavior, we reviewed the source code and consulted the application designer. This provided us with the corresponding functional characterization of the R application.

Table 6: Dynamic I/O Characterization

Phase	Number of Cycles	Cycle Activities	Cray YMP Time	I/Os	Bytes
Initial	30	Read File 5		5	983,040
		Read File 6	.002097 secs	6	1,179,648
		1st Computation	5.130000 secs	---	---
		Read File 4	.279389 secs	55	10,813,440
		2nd Computation	.640000 secs	---	---
	1	Write File 8	.004322 secs	25	4,915,200
		Read File 8	.005519 secs	25	4,915,200
Main	30	Read File 5		5	983,040
		Read File 6	.002095 secs	6	1,179,648
		1st Computation	5.160000 secs	---	---
		Read File 4	.273200 secs	55	10,813,440
		2nd Computation	16.70 0000secs	---	---
		Write File 7	.000183 secs	2	393,216

The three predominant data files used for the analysis are the weather station site data (file 5), the large-scale NMC global data (file 6), and the fine-grained RAMS climate model output data (file 4). Each file contains 30 days of climate measurements made at 0Z and 12Z GMT. The data space for the analysis is represented by a three-dimensional matrix with dimensions 61 x 51 x 31. Each point in the data space is described by 8 variables. The RAMS model output data has data values for each point in the analysis data space, whereas the NMC and weather station site data only have data values for a subset of the data points in the analysis data space. R performs two separate computations in the analysis and in each computation the three data files are read sequentially, processing each of the 30 days one at a time (i.e., the initial and main phases with 30 cycles each).

There is no overlap between computation and I/O.

In the first computation, daily average values are computed for each point in the data space. One iteration of this computation is as follows. One day's worth of NMC and weather station site data is read (i.e., read I/O files 5 & 6). Since these data sets represent only a subset of the values needed for the analysis data space, preprocessing of the data is necessary (i.e., 1st computation period). Next, one day's worth of RAMS model output data is read (i.e., read I/O file 4) and subsequently average values are computed for each point in the data space (i.e., 2nd computation). When average values for all days have been computed, they are written out to one file (i.e., write I/O file 8).

In the second computation, covariances between various variables in the data space are computed. This computation begins by reading the daily average values for the one month (i.e., read I/O file 8). One iteration of the remainder of the computation is as follows. One day's worth of NMC and weather station site data is read (i.e., read I/O files 5 & 6) and preprocessed as in the first computation (i.e., 1st computation period). Next one day's worth of RAMS model output data is read (i.e., read I/O file 4) and then covariances are computed (i.e, 2nd computation). At the end of the computation, the covariance values for the given day are written out (i.e., write I/O file 7). When all 30 days have been processed, the covariance data file is closed and the application terminates.

6. Conclusions

Characterizing the dynamic I/O behavior of scientific applications and isolating regular, recurring patterns of I/O activity, requires detailed measurement and analysis of run-time I/O activity. We monitored a climate research analysis application on a Cray Y-MP8/864 supercomputer and produced dynamic I/O profiles with respect to virtual or CPU time depicting the type, volume, and frequency of I/O resource usage over time. These virtual time I/O profiles enabled use to observe the application's I/O activity as embodied in the source code, eliminating the effects the system workload. From these profiles, we characterized R's dynamic I/O behavior as two distinct and repeating cycles of I/O activity. The accuracy of the dynamic profiles and characterization were verified by inspecting the actual source code of R and discussing its functionality with the application designer.

An interesting feature of the dynamic I/O behavior of R is that it displays a different type of I/O profile than what had been observed in the past for some other applications. In previous studies [4,5] and in our own preliminary investigations with different codes, I/O profiles which have three phases of processing, initial, main, and final, have been observed. The initial phase is usually short in duration and has high-volume, bursty read I/O activity. The main phase represents the major part of total execution time and exhibits recurring cycles of both read and write activity. In this phase, read and write activity appear to be equal in frequency and intensity. The final phase is also short in duration and has high volume, bursty, write I/O activity.

We believe that the regularity and consistency exhibited in R's dynamic I/O behavior is not uncharacteristic of scientific applications in general. Although sophisticated and complex, scientific applications for supercomputers are highly structured, regular codes which utilize carefully formatted data sets. Based on the strict design of these applications and the physical constraints of the system, it is likely that the resource usage of scientific applications, especially I/O, follows a regular and predictable pattern. The case study analysis presented here and the methods followed to obtain dynamic I/O profiles are the first steps in establishing an understanding of the dynamic I/O demands of scientific applications. We intend to pursue further studies to investigate how prevalent is this regular I/O activity among supercomputer applications with intense I/o demands.

7. Acknowledgments

The authors are grateful to Dr. John O. Roads of Scripps Institution of Oceanography for providing the R application and discussing details of R's analysis procedures. They would also like to thank Jon Cargille of the University of Wisconsin-Madison for his assistance in using the newly released Cray version of IPS-2.

8. References

[1] Stonebraker, M., "Overview of the Sequoia 2000 Project," Proceedings of COMPCON '92, San Francisco, California, February 1992.

[2] Hennessy, J. L., and Patterson, D. A., "Computer Architecture: A Quantitative Approach," Morgan Kaufmann Publishers,

Inc., 1990.

[3] Katz, R. H., Gibson, G.A., and Patterson, D. A., “Disk System Architectures for High Performance Computing,” Proceedings of the IEEE, Vol. 77, No. 12, pp. 1842-1858, December 1989.

[4] Miller, E. L. and Katz, R. H., “Input/Output Behavior of Supercomputing Applications,” Proceedings of Supercomputing ‘91, November 1991.

[5] Lim, S. B. and Condry, M. W., “Supercomputing Application Access Characteristics”, Technical Report No. UTUCDCS-R-91-1708, University of Illinois at Urbana-Champaign, October 1991.]

[6] Pasquale, B. K. and Polyzos, G. C., “A Static Analysis of I/O Characteristics of a Broad Class of Scientific Applications”, UCSD Technical Report No. CS92-270, November 1992.

[7] Miller, B. P., et. al., “IPS-2: The Second Generation of a Parallel Program Measurement System”, IEEE Transactions on Parallel and Distributed Systems, Vol. 1, No. 2, April 1990.

[8] Roads, J. O., et. al., “A Preliminary Description of the Western U.S. Climatology”, Proceedings of the Ninth Annual Pacific Climate (PAClim) Workshop, September 8, 1992.

Appendix A

Cray YMP Analysis

Table A1: Initial Phase I/O Cycle Statistics

Event	Number of Events (per cycle)	Minimum Interevent Time (microseconds)	Maximum Interevent Time (microseconds)	Mean (microseconds)	Standard Deviation (microseconds)
Files 5 & 6 Read I/O	9	226	589	277.19	55.57
File 4 Read I/O	55	5040	5152	5082.01	16.42
Event	Number of Events (per Initial Phase)	Minimum Time (microseconds)	Maximum Time (microseconds)	Mean (microseconds)	Standard Deviation (microseconds)
First Computation	30	643832	665343	647431	3852.015
Second Computation	30	5113249	5148545	5130149.567	9203.746

Table A2: Main Phase I/O Cycle Statistics

Event	Number of Events (per cycle)	Minimum Interevent Time (microseconds)	Maximum Interevent Time (microseconds)	Mean (microseconds)	Standard Deviation (microseconds)
File 7 Write I/O	2	182	209	186.20	11.314
Files 5 & 6 Read I/O	9	229	526	279.15	54.341
File 4 Read I/O	55	5015	5386	5107.55	51.360
Event	Number of Events (per Main Phase)	Minimum Time (microseconds)	Maximum Time (microseconds)	Mean (microseconds)	Standard Deviation (microseconds)
After Write I/O	30	4212	4378	4277	52.00
First Computation	30	5124249	5220935	5165872	24335.69
Second Computation	30	.166801e08	.167357e08	.167042e08	15327.10