A Static Analysis of I/O Characteristics of a Broad Class of Scientific Applications

Barbara K. Pasquale
George C. Polyzos

SEQUOIA 2000
Technical Report #92/19

November 1992

Computer Systems Laboratory
Department of Computer Science and Engineering
University of California, San Diego
La Jolla, CA 92093-0114, USA

A Static Analysis of I/O Characteristics of a Broad Class of Scientific Applications

Barbara K. Pasquale and *George C. Polyzos*

Computer Systems Laboratory
Department of Computer Science and Engineering
University of California, San Diego
La Jolla, CA 92093-0114, USA

ABSTRACT

Past research on high performance computers for scientific applications has concentrated on CPU speed and exploitation of parallelism, but has, until very recently, neglected I/O considerations. This paper presents a study of the production workload at the San Diego Supercomputer Center from an I/O requirements and characteristics perspective. Results of our analyses support our hypothesis that a significant proportion of scientific applications with intensive I/O demands have predictable I/O requirements.

1. Introduction

Past efforts to develop high performance computer systems have been primarily focused on the computational speeds of processors, often ignoring other important system components, such as the I/O subsystem and the operating system [3, 8, 17, 19]. Resulting progress in the areas of raw processor speed and parallelism, both in hardware and software, has produced GFLOPS machines, but has done little to close the ever widening gap between CPU performance and that of the attached I/O subsystem [1, 8, 9]. It is now apparent, however, that the successful future of high performance computing relies on our ability to design well-balanced systems which can easily provide the functionality and performance required by current and future scientific applications.

The traditional viewpoint that supercomputer applications are number-crunching, compute-bound codes, has led to little concern over the growing system imbalance between CPU and I/O performance. Now however, scientific research based on computational approaches has intensified. This mode of computing is now exerting much greater demands on the I/O system. Applications involving simulation based modeling require massive amounts of diverse data to be stored, organized, accessed, distributed, analyzed, and visualized. Relying on these large-scale computations and data analysis techniques, progress in many scientific disciplines is limited only by the available capacity of high performance computing [4]. In particular, computational Global Change research in fields such as oceanography, atmospheric science, and geology, relies on and produces atmospheric and oceanographic data sets ranging from hundreds of megabytes up to tens of gigabytes per execution. The sheer volume of this data intensifies I/O demands within the local system and communication requirements across networks [16].

Continuing to increase CPU speeds and to further exploit parallelism without improving the I/O system will create more I/O bound jobs which can become a bottleneck to system performance [1, 8, 9, 11, 12, 18]. In a recent study of the San Diego Supercomputer Center (SDSC), a monthly CPU idle time increase of 66% was directly attributed to I/O blocking [13]. In order to maintain well-balanced systems, we must establish a thorough understanding of the I/O behavior of scientific applications, and from this knowledge, design the mechanisms and implement the policies that are needed to improve the I/O subsystem and provide these I/O intensive applications with their needed resources without degrading overall system performance.

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).

Based on the need to design well-balanced supercomputer systems that deliver the high performance expected of them, we have undertaken a comprehensive long-term study of I/O requirements of scientific applications, in collaboration with other CSL and SDSC researchers. The main goals of this research are:

(1) to characterize the I/O behavior of a wide spectrum of scientific applications,

(2) to develop models for well-defined classes of scientific application I/O activity,

(3) to construct a set of executable I/O benchmarks for the purpose of locating existing or potential I/O bottlenecks, and

(4) to recommend operating system enhancements to improve supercomputer I/O performance.

The research presented in this paper corresponds to the first of the above goals and is a prerequisite for attaining the other goals. In order to determine a wide spectrum of interesting applications we undertook a workload characterization study of the Cray Y-MP at SDSC. Our goal here is to present some of the observations of that study and to investigate the hypothesis that a high proportion of I/O intensive applications exhibit regular behavior. The I/O metrics considered in this study include the number of total bytes transferred by an application, the virtual I/O rate, i.e., the number of bytes transferred per CPU second, and the number and average virtual rate of logical I/O requests. We are interested in identifying a set of I/O intensive applications and in summarizing their I/O resource usage in a way that describes the relationship between the applications, their resource usage, and their contribution to the entire system workload.

The remainder of this paper is organized as follows. In section 2 we discuss the motivation for investigating I/O behavior, and particularly, the roles of static and dynamic characterizations. We also discuss two recent scientific application I/O studies, and present our hypothesis. In section 3 we provide a description of the observed environment at the SDSC. In sections 4 and 5 we progressively present our selection of interesting subsets of the workload and we describe the analyses performed and results obtained. Our conclusions are presented in section 6.

2. Motivation

In the past, little attention has been given to the study and analysis of I/O behavior on high performance systems. Therefore, our understanding of the effects of current application I/O and projected increases in application I/O is not well established. Scientific applications form an interesting set of programs worthy of study because they absolutely need high performance supercomputers and therefore, systems should be designed to meet their requirements. Before we can do the latter, we must understand these requirements. We believe that it is necessary to undertake a thorough study of production workloads in order to extract the characteristic behavior of I/O intensive applications.

2.1. Related Research

The analysis of application I/O has been seriously addressed in two recent studies by Miller and Katz [12] and Lim and Condry [10]. By recording detailed information about each I/O request (e.g., type of I/O, request size, number of bytes transferred, file accessed, and file position), they were able to construct run-time profiles depicting an application's I/O behavior over time.

Miller and Katz characterized and modeled the behavior of 7 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. Applications analyzed had high I/O rates and used massive data files. The resulting run-time profiles revealed very interesting characteristics about scientific application I/O. Three types of I/O requests were observed: compulsory (I/O accesses to read an initial state and to write interim and final results), checkpointing, and data staging. After the brief initial phase of compulsory I/O, the applications produced bursts of high-volume I/O activity at regular intervals which created a cyclical pattern and ended in a burst of final phase compulsory I/O. They also observed that file accesses were sequential and that request sizes remained constant during the cyclical phase of I/O. They attributed this regularity to the iterative nature of the underlying algorithms.

To determine how Grand Challenge applications could better exploit gigabit networks, Lim and Condry analyzed the I/O behavior of four chemistry applications and the Perfect Club benchmark suite. The dynamic profiles and resulting analysis revealed that in their present form, these applications were incapable of utilizing the full potential of a high-speed network because over 50% of the data accesses resulted in small data transfers, namely less than 16 KB. However, the individual executions of each chemistry application analyzed exhibited a high degree of consistency with respect to the number of read/write accesses, total bytes read or written, disk space usage, data request sizes, and sequential access patterns. Looking at these I/O activities over time, revealed one general pattern of dynamic I/O behavior.

The two studies presented above are extremely valuable sources for instrumentation and analysis methodology. They also demonstrate that characterizing application I/O is possible. Their initial results indicate that for the classes of scientific applications considered, I/O activity has, in general, a cyclical pattern, possibly augmented with distinct initial and final phases. However, these studies have dealt with relatively few applications chosen from very specific disciplines, and extrapolating from these results to the I/O behavior problem in general might be questionable. In particular, these studies do not address the question of how prevalent are the selected applications in the system workload, and how intense is their I/O activity in comparison with the remainder of the workload.

2.2. Our Working Hypothesis

The hypothesis we investigate here is that scientific application I/O is predictable. This hypothesis is based on characteristic I/O behaviors discovered in previous studies (described above, but also in our own preliminary investigations of the dynamic behavior of scientific applications), and the following two key observations:

(1) scientific applications are highly structured, regular codes, and

(2) system configuration and policies influence how resources are used.

Although sophisticated and complex, most scientific applications are based on components that are implementations of well-known algorithms. For example, sparse linear system solvers, FFTs, rapid elliptic problem solvers, multigrid schemes, integral transforms, etc. Given the underlying algorithms of the applications and some specifics of their implementations (e.g., number and type of operations performed, size of data set), an estimate of the required CPU time and memory requirements can usually be obtained.

However, even supercomputer resources are finite, and therefore, the coded application must conform to the physical limitations of the system. The best example of this conformity deals with the use of in-core memory. On a real memory machine like the Cray Y-MP, the entire application and its data set must be resident in memory during execution. Since data sets often eclipse the maximum available memory partition, the technique of data overlaying or data staging is used [12]. At regular intervals in the computation or at each iteration of a looping construct, processing is temporarily suspended while the entire in-core data set is replaced with a new data set. As a result of this data swapping, a cyclic pattern of same size I/O requests is generated by the application. The same is also true when checkpointing is performed [12]. Checkpointing involves writing a portion of the in-core data set to disk to save the state of the computation should the system fail. If failure does occur, the application can then be restarted from the last checkpoint, with minimal recomputation.

Based on the inherent structure of scientific 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.

2.3. Our Approach

We focus on the characterization of a broad set of I/O intensive applications rather than overall system behavior because our intent is to investigate current I/O demands as well as future ones, which are not limited to what is observable in current systems. Specifically, we are interested in understanding the issues involved around a potential I/O bottleneck in future supercomputers and to provide tools to prevent it. This scenario is associated with supercomputers that are computationally much more powerful than those of today and also with workloads that are much more I/O intensive. The latter is driven by user expectations about performance of the future machines. Based on current application I/O characterizations one can extrapolate future I/O application requirements, and then through synthesis determine overall I/O workload intensities and patterns.

The first issue that needs to be resolved is how to determine which applications are most worthy of study. The applications selected should span a variety of disciplines and cover the major scientific codes used in these disciplines, as well as represent a frequently executed, I/O intensive component of the workload. In general, this would require a survey of a large number of supercomputer centers, where it would be necessary to analyze the composition of their workloads. Once the major scientific applications of each center are recognized, the results would have to be combined. This is a major undertaking, and outside the scope of our study. However, we conducted such a survey at SDSC, which already supports a diverse workload of many scientific applications, and isolated a set of frequently executed, I/O intensive applications based on measured volumes and rates of I/O activity. The methodology developed for this study can be used to conduct similar studies at other centers.

The second question revolves around how to characterize an application’s I/O, in particular, what are the appropriate descriptors. In general, an answer to this question depends on the purpose of the characterization, and characterizations are expected to be objects of study themselves. However, there is one basic dichotomy: static or dynamic characterization. Dynamic characterizations (particularly if they can be made independent of the system and the remainder of the executing workload) are the most powerful, and thus, the most desirable ones. However, they are also expensive to obtain and cumbersome to work with. Therefore, only a limited number of applications can be followed closely in order to provide dynamic profiles of them. On the other hand, static analysis can be applied to a much wider set of scientific applications and should therefore be the first and guiding step for the dynamic analysis.

3. The San Diego Supercomputer Center Environment

The system and workload observed for this study was that of the San Diego Supercomputer Center (SDSC) Cray Y-MP8/864. SDSC is one of the four U.S. National Science Foundation sponsored Supercomputer Centers. The SDSC workload spans all major scientific disciplines, including materials science, physics, biochemistry, atmospheric science, chemistry, chemical engineering, astronomy, mechanics, oceanography, and electrical engineering. The user community consists of 2500 researchers from over 150 academic institutions. The amount of data manipulated daily exceeds 1 TB. The broad base of this system's workload and the scientific significance of its applications, makes it an important and appealing environment to study.

In addition to the Cray, SDSC also operates two parallel machines, a 64-node Intel iPSC/860 (soon to be upgraded to an Intel Paragon), and a 128-node nCUBE Mark 2. We have considered focusing our study on one of these machines, but have found many drawbacks in that approach. Even though both of these machines are now in heavy use, there are severe problems that limit the feasibility and usefulness of such a study at the present time. First, there exist no good facilities for collection of statistics and system tuning is not mature enough. In addition I/O capabilities are much more restricted than those of the Cray. Finally, probably because of these limitations, the workload on these machines tends to be specialized and does not reflect well the requirements of the broad scientific computation community.

Of all the supercomputing resources at SDSC, the Cray Y-MP supports the most diverse I/O workload and contains a wide variety of hardware resources. The Y-MP has a 6 ns clock, 8 CPUs, and 512 MB (or 64 M 64-bit words) of memory. For this installation, the Cray Y-MP has been tuned to maximize utilization, with the supercomputer idle only 1% of the available wall clock time. A job mix scheduler attempts to dynamically select an optimal job mix to execute within the real-memory architecture of the Y-MP. Keeping 12-15 jobs in memory typically allows a context switch to be made whenever a job is waiting for I/O completion. This tends to maximize the amount of generated I/O in addition to maximizing CPU utilization. The sustained average file system I/O rate of the workload is on the order of 10 MB/s.

To support the I/O requirements of a combined interactive and production workload, the storage system for SDSC's Cray Y-MP is a 5-level buffer and cache hierarchy. Table 1 details the levels in this hierarchy and their characteristics [13].

Table 1: SDSC Storage Hierarchy

Caching Level	Capacity (GB)	Max. Transfer Rate (MB/s)	Daily I/O Volume (GB)	Residency Period
SSD	1	1250	1500	minutes
Local Disks	65	10	1700	days
Archival Disks	70	2 (avg.)	5	weeks
Tape Robot	1200	-	8	months
Shelf Tape	2000	-	3	years

The critical levels in this hierarchy are the SSD and the local disks. Together these devices must meet the demands of the two major sources of I/O in the system, namely job swapping and application disk I/O. To service the interactive component of the workload, 388 MB of the SSD is allocated as a data cache. This SSD data cache provides for high-speed access to the root file system as well as for interactive swap space for jobs less than 8 MB. The remainder of the SSD is allocated as a data buffer for the temporary file system. Data from this 42 GB file system effectively streams through the SSD with minimal reuse. The local disks serve as a data cache for the temporary and scratch file systems and for swap space for large jobs (i.e., greater than 8 MB).

Large, long running jobs, namely those that require more than 20 minutes of CPU time, 6 MWords of memory, and/or 60 MWords of local disk space, may not be run interactively. Instead, these jobs must be submitted for execution through the NQS (Network Queueing System) Unicos batch facility. Based on the required resources and run-time priority level, users submit their jobs to one of the 27 batch queues described in Table 2 below. The individual queues are distinguished by their type, memory space partition, and priority level. For example, of the 20 standard queues, there is a separate low, medium, high, and preferred priority queue for each of the 5 available memory space partitions. By default, the ‘‘large disk’’ queues have high priority. Within each queue, jobs are handled on a first-come first-served basis and run when the required resources are available.

Table 2: Batch Queues Parameters

Type	Number	CPU Limit (minutes)	Memory Space (MWords)	Local Disk Space (MWords)
Standard	20	6000	2, 4, 8, 16, or 32	85
Large Disk	4	6000	8 or 32	625 or 1000
Test	1	8	32	85
Express	1	10	6	85
Standby	1	360	6	85

Considering the resource constraints in the NQS environment, users carefully structure their codes and data sets to balance the trade-offs between run-time priority level, job turnaround time, and main memory availability. The breakdown of the 4,336 batch jobs serviced in February 1992 by memory partition size is as follows: 39.1% for the 2 MWord, 23.15% for the 4 MWord, 25.07% for the 8 MWord, 7.7% for the 16 MWord, and 4.98% for the 32MWord.

4. I/O Workload Characterization

We first conducted a workload characterization study to isolate the I/O intensive component of the workload. This component represents the set of scientific applications which have extensive demands in terms of I/O volume, i.e., total number of bytes transferred, and average virtual I/O rate, i.e., number of bytes transferred per CPU second. By analyzing this component, we can then identify these applications by name and determine their combined resource usage with respect to the entire system workload.

Using the virtual I/O rate, rather than real I/O rates, provides us with some basic isolation from the details of the system architecture and the workload executing simultaneously with our target application (and the resource contention it generates). Of course, many other factors cannot be completely controlled, such as idiosyncrasies of the accounting software and the way it charges CPU time to processes. However, the degree of approximation achieved is adequate for the purpose of this stage of the research, where we are more interested in general trends and behavior rather than the details of application dynamics.

4.1. Data Collection

Measurements of application resource usage were obtained from the Cray System Accounting (CSA) utility and collected over a one-month period during February 1992. On a per job-basis, the CSA utility automatically captures an application’s resource usage through the use of kernel probes and creates a process account record that summarizes the total resource usage. The recorded resource usage includes: application name, process, user, and job identification numbers, start and end times, total CPU time (execution time in either system or user mode), total number of bytes transferred, number of logical I/O requests, number of physical I/O requests, memory high water mark, CPU connect times for multitasked jobs, and I/O wait times.

Although the CSA data only provides aggregate resource usage, it enabled us to analyze the workload at two important levels, the functional level and the physical resource level [3, 5, 7]. Serazzi [15] showed that reliable workload characterizations can capture the functionality of the real system workload while still preserving the underlying physical resource usage. By spanning these two levels, we can describe both the higher level applications and the lower level resource usage simultaneously, so that the relationship between I/O intensive applications and the resources they consume can be understood.

4.2. The Workload

The collected data was separated into two categories: *system utilities* and *user applications.* System utilities represent programs that are available to all users whereas user applications are limited to a single user or a small group of users. Table 3 shows how each component contributes to the overall workload resource usage.

Table 3: February 1992 Resource Usage								
	Total Number of Applications Executed		Number of Distinctly Named Applications Executed		Cumulative CPU Time (s)		Cumulative Bytes Transferred (MB)	
System Utilities	5,651,460	95%	445	12%	1,566,303	8%	5,477,771	12%
User Applications	311,408	5%	3,366	88%	16,822,314	92%	41,552,252	88%
Total Workload	5,962,868	100%	3,811	100%	18,388,617	100%	47,030,023	100%

As can be noted from the figures of Table 3, the *user* component of the workload has a tremendous impact on total resource usage. Even though it represents an extremely small portion of all executed jobs (5%), it accounts for 92% of total CPU time and 88% of total bytes transferred. The results of this workload characterization study are consistent with an earlier study of the SDSC workload [14] where it was found that *user* applications represented 7% of the workload and consumed 90% of total CPU time and 75% of I/O channel time.

4.3. I/O Intensive Applications

To extract the set of I/O intensive applications from the set of all *user* applications, we produced an average virtual I/O rate ordering of all distinctly named *user* applications and calculated resource statistics with respect to the total workload resource usage. Focusing more on I/O rates rather than total I/O volume is dictated by our desire to investigate applications that might stress the I/O system to the point of affecting their response time or interfere with other applications. Table 4 shows an abbreviated version of this ordering.

Table 4: Average I/O Rate Ordering

Top N Ranked Applications	Percentage of Distinctly Named User Applications	Percentage of Total Workload CPU time	Percentage of Total Workload Bytes Transferred	Minimum I/O Rate of Top N Grouping (MB/s)
1	0.03	0.001	0.10	230.56
10	0.3	0.30	14.59	107.86
20	0.6	0.60	24.54	80.13
30	0.9	0.60	24.60	71.71
50	1.5	0.66	26.27	49.87
80	2.4	0.69	26.85	35.48
160	4.8	3.16	48.89	18.45
300	8.9	8.23	70.33	6.89
360	10.7	9.93	74.64	4.78
580	17.2	19.90	85.98	1.58

Given that the percentage of CPU time was increasing at the same rate as the number of applications considered, we investigated the average CPU time for these applications. A distribution of the average CPU times for the top 580 I/O rate ordered applications revealed that 472 of these applications (i.e., 82%) had an average CPU time of approximately 106 seconds or less. To filter out the effects of these short, bursty jobs and focus attention on those jobs which exert a longer, sustained demand on system resources, we produced a new I/O rate ordering which included only those applications whose average CPU time exceeded 100 seconds. The statistics of this new set are given below in Table 5.

Table 5: Average I/O Rate Ordering with Average CPU Time > 100 sec

Top N Ranked Applications	Percentage of Distinctly Named User Applications	Percentage of Total Workload CPU Time	Percentage of Total Workload Bytes Transferred	Minimum I/O Rate of Top N Grouping (MB/s)
1	0.03	0.13	6.73	131.70
10	0.3	1.29	33.63	29.78
20	0.6	3.35	49.26	18.07
30	0.9	4.90	57.63	11.08
50	1.5	9.42	71.08	4.70
80	2.4	12.86	75.97	3.31

Thus, using only 1.5% (i.e., 50) of the *user* applications we were able to identify an I/O intensive set of scientific applications, in terms of both volume and rate, whose combined resource usage accounts for 71% of total bytes transferred and only about 9.5% of total CPU time (see Appendix A). The combined effects of applications like these can exert peaks of I/O load that might considerably stress the I/O subsystem. Therefore, these are the applications that we have selected as candidates for further study.

5. Regularity in I/O Intensive Applications

From the set of 50 I/O intensive applications, 24 were selected for individual analysis (see Appendix A). Our selection criterion was based on the frequency of execution over the month-long observation period. This is important for two reasons. First, in this initial phase of analysis, it is necessary to uncover the I/O intensive applications which are continually present in the workload. Second, several observations are required to make judgements about typical resource usage. For these reasons, we set the execution frequency threshold to 10 executions over the one month period considered. Although 10 executions might at first seem a low number, one must also consider that many of these scientific applications use hours of Cray CPU time, and therefore, their execution frequency is often limited by the turnaround time of the NQS batch system. Note that with respect to the entire workload, these 24 applications consumed 6% of the total CPU time and were responsible for 55% of the total number of bytes transferred.

5.1. Cluster Analysis

To find natural partitions or patterns within the resource usage records for a given application, the multidimensional analysis technique of *K*-means clustering was used [6, 7]. A notable feature of the clustering algorithm was that it used weighted Euclidean distance as a dissimilarity measure, which allowed the size of each cluster to vary in inverse proportion to its variance. The resource usage parameters used in the clustering were: CPU time, memory high water mark, and number of logical I/O requests. An average logical I/O request rate was also calculated for each resulting cluster. Although clustering algorithms are non-trivial because they must recognize ‘‘nearness’’ among the characteristic parameters selected, Calzarossa and Ferrari [2] found that the non-hierarchical *K*-means algorithm produces reliable results for workload characterization and modeling.

Based on the clusters found for each application, three general patterns of resource usage were observed and are described below.

(1) Logical I/O Rate and CPU Time: The logical I/O rate remained fairly stable across marked divisions in CPU time for approximately 1/3 of the applications. For other applications like `trans.im`, `timteb.x`, and `xm901`, the logical I/O rate decreased across increasing divisions in CPU time. In some sense, this behavior can be expected. If an application has intense phases of initial and/or final I/O activity, the average logical I/O rate will naturally decrease for longer-running executions.

(2) CPU Time: Several applications showed a small range or fairly constant CPU time across all cluster groupings. However, for more than half of the applications, cluster groupings were separated by marked divisions in CPU time. These applications had both short executions on the order of seconds and long executions on the order of minutes.

(3) Memory High Water Mark: The memory high water mark (or memory size), represents the maximum number of main memory words allocated to a program during execution. For virtually all of the applications, this average memory size was consistent across all cluster groups. Looking at the exact, unscaled counts for all executions of a given application, revealed the reasons for this consistency. Maximum memory sizes for the applications fell into one of three categories: (a) exactly one size, (b) one small range of sizes, or (c) one distinct size (or possibly two) and a small range of sizes. It is also interesting to note that the applications were roughly equally divided among these memory size categories.

Based on the average value for each parameter (e.g., CPU time, number of logical I/O requests, memory space), the individual clusters of each application showed distinct resource usages. For the 24 applictaions, 101 such clusters were generated. However, not all of the clusters seemed to provide useful information about the characteristic resource usage of the application. In particular, many applications had singleton clusters with CPU times of less than 1 second and number of logical I/O requests less than 50, while other clusters had significantly longer execution times and larger numbers of logical I/O requests. Therefore, to better understand the relationship between the individual clusters and the application, we calculated the percentage of total application resource usage attributed to each of the individual application clusters. From the percentages obtained, we could determine which individual clusters represent the application as a whole in terms of total resource usage and define its ‘‘characteristic’’ resource usage. Considering the percentage of individual cluster resource usage, as well as cluster size, it was possible to characterize 15 of the applications by only 1 of its clusters and 6 of the applications by only 2. For the 15 applications with 1 ‘‘characteristic’’ cluster, approximately 95% of CPU time and 97% of logical I/O requests were attributed to the 1 cluster containing 67% of total executions on the average. For the 6 applications with 2 ‘‘characteristic’’ clusters, approximately 95% of CPU time and 95% of logical I/O requests were attributed to the 2 clusters which jointly contained 84% of total executions on the average. The remaining 3 applications (`cpmd.x`, `timteb.x`, `dir.cpx`), however, were each characterized by 3 of their clusters. For these applications, large sized clusters made a small contribution to total resource usage while small sized clusters made a significant contribution. The ‘‘characteristic’’ clusters for each application are described further in Appendix B.

5.2. Regression Analysis

Given that several applications showed consistent logical I/O rates across individual cluster groupings, we decided to investigate statistical correlations between the measured values of CPU time, characters transferred, and logical I/O request count. Clustering algorithms, such as the one we used in the previous section, can detect natural groupings in data based on a specific parameter set, but do not describe statistical relationships between the variables in the selected parameter set. Therefore, we used regression analysis to explore such potential relationships.

Table 6: CSA Resource Statistics for `griz.exe`

CPU Time (sec.)	Elapsed Time (CPU sec.)	Bytes Transferred	Logical I/O Count	High Water Memory	Logical I/Os per CPU sec.	Bytes Transferred per CPU sec.
207.6	828.488	4971823104	25415	595968	122.38	23939800
177.53	617.30	4971823104	25415	595968	143.16	28005500
183.73	488.80	4885053440	25003	595968	136.09	26582200
177.50	553.93	4885053440	25003	595968	140.86	27521400
174.79	465.79	4885053440	25003	595968	143.05	27948100
191.64	477.31	4884267008	24965	595968	130.27	25486700
178.99	582.65	4884267008	24965	595968	139.48	27287900
177.36	465.11	4884267008	24965	595968	140.76	27538700
100.70	208.68	2694578176	13790	595968	136.91	26758500
94.74	222.43	2694578176	13790	595968	145.56	28441800
93.80	169.25	2694578176	13790	595968	147.02	28726800
93.69	160.52	2694578176	13790	595968	147.19	28760600

As an example of the analysis performed to detect underlying resource usage relationships, we present the data and statistics for application `griz.exe`. This application has all the characteristics of an application with highly regular resource usage: a consistent logical I/O rate, exactly one memory high water mark, and a range of CPU times. Considering the measured values from the process account records and the calculated logical I/O request and bytes transferred rates given in Table 6, it is not difficult to see this regularity. Even though there is no particular size for logical I/O requests imposed by the system, one can expect a high degree of correlation between the logical I/O count and the number of characters transferred in any application. For `griz.exe` these two measures are near perfectly correlated.

Next we investigated the relationships between CPU time and both logical I/O count and the total number of bytes transferred. Linear regression analysis showed the major influence CPU time has on the total number of bytes transferred as well as on the number of logical I/O requests. Table 8 provides the actual values from the analysis.

Table 8: Regression Analysis for `griz.exe`

Dependent Variable	Independent Variable	Coefficient of Determination	*t*-statistic
Bytes Transferred	CPU Time	.964	16.255
Logical I/Os	CPU Time	.963	16.189

The important result of this analysis is the high value of the coefficient of determination (which describes the goodness of fit of the model), for both models. This strong correlation between CPU time and these I/O measures is compatible with the hypothesis that `griz.exe` contains a regular, repetitive I/O processing phase, like those observed in [10, 12].

Regression results for all selected applications are provided in Appendix C. Table C.1 shows results for the standard linear model

$$\text{Number\_of\_Bytes\_Transferred} = c + a \times \text{CPU\_Time}$$

We have grouped the 24 applications into three categories based on the regression results. The first group with 8 applications has all regression coefficients positive and relatively high coefficients of determination. This group fits well the hypothesis of regular behavior during the main phase, with I/O volume highly correlated with CPU time, and additional initial and final I/O phases (with positive I/O amounts, independent of the CPU time).

The second group contains only two applications. The coefficients of determination, are very low, suggesting that this model does not fit these applications.

The last group contains 14 applications. Even though R^2 is high (0.866 and above), suggesting a good fit for a linear model with positive correlation between CPU time and number of bytes transferred, the values of the constant of the regression obtained are negative, which does not fit our hypothesis of I/O during an initial or final phase. The *t* statistic for the constant is low for 10 of the 14 applications in this group, however, it is typically not advisable to drop the constant of the regression under any circumstances. The *t* statistic for the coefficient of the CPU time is always very high, except for the 2 applications of the second group.

In response to the results for this last group and in order to investigate non-linearities in the variables, we attempted to fit the following model,

$$\text{Number\_of\_Bytes\_Transferred} = e^{c} \times \text{CPU\_Time}^{a}$$

which is linear after a logarithmic transformation of the data. Table C.2 in Appendix C shows the results for this model. All applications fit well the above model, except three. The lowest coefficient of determination is obtained for `stone.exe` which also had an unacceptable low value in the case of the first model. Inspecting the data for this application reveals that there is essentially one run profile for this application, with small, seemingly random, variations in CPU time and number of characters transferred. The power of CPU time estimated from the regression is roughly between one and two (or close to these values), which is interesting in itself. The estimated constant, *c*, has values between 8 and 20.

6. Conclusions

We performed a study of the main workload of a supercomputer installation with a very diverse clientele, concentrating on scientific application I/O characteristics. Based on our investigation, we found that there is a high degree of regularity in resource usage for scientific applications. Cluster and regression analysis of system accounting data yielded results which support the hypothesis that scientific applications have a high degree of regularity in their functional operation and resource usage. I/O demands, both in terms of the number of characters transferred and the number of logical requests, show considerable correlation with CPU time consumed.

The three general resource usage patterns observed show that even though more than half of the applications exhibit both ‘‘short’’ (i.e., on the order of seconds) and ‘‘long’’ (i.e., on the order of minutes) executions (in terms of CPU time), logical I/O rates for about one-third of the applications remain fairly stable across the different execution time periods. For all applications, in-core memory usage was highly consistent in that either a single size or a small range of sizes were used. Considering the percentage of resource usage attributed to the individual clusters of an application, as well as cluster size, it was possible to use only a small set of individual clusters to the describe the characteristic resource usage of the entire application. Of the 24 applications characterized, 15 could be described with only 1 cluster, 6 with 2 clusters, and the remaining with 3 clusters.

Speculating on the underlying causes for the observed general patterns of resource usage and the individual application characteristics, we offer the following explanations. The consistency in memory size can be attributed to fixed sized data sets obtained from external data collection devices or to carefully partitioned data sets designed to take advantage of the Y-MP’s fixed partition memory scheme. The "short" and "long" divisions in CPU time can be attributed to the size of the data set used or to the degree of resolution required in the computation (i.e., number of repetitions over a single-sized data set). The stability in logical I/O rates across different execution times may represent the dominanting effects of a highly regular main processing phase within the application. Through dynamic profiling, Miller and Katz [12] observed a class of scientific applications whose main processing phase consisted of a period of CPU processing followed by a burst of intense I/O activity, which repeated at regular intervals. If one were to assume negligible I/O during the initial and final phases and that the number of I/O requests are constant within each iteration, an application in this class would have a consistent, logical I/O rate across different execution times.

In the end, a regular behavior can only be confirmed through dynamic analysis on a per application basis. To obtain such evidence, it will be necessary to monitor and measure the run-time resource behavior of these applications. We intend to proceed in that direction and the present analysis will guide us in selecting the applications to consider and also to generalize the results obtained by dynamic analysis to classes of applications determined by this study.

References

[1] Bell, G., ''The Future of High Performance Computers in Science and Engineering,'' *Communications of the ACM,* Vol. 32, No. 9, pp. 1091-1101, September 1988.

[2] Calzarossa, M., and Ferrari, D., ''A Sensitivity Study of the Clustering Approach to Workload Modeling,'' *Performance Evaluation,* Vol. 6, pp. 25-33, North-Holland, 1986.

[3] Calzarossa, M., and Serazzi, G., ''Workload Characterization for Supercomputers,'' *Performance Evaluation of Supercomputers,* Ed. J. L. Martin, pp. 283-315, North-Holland, 1988.

[4] Denning, P. J., and Adams III, G. B., ''Research Questions for Performance Analysis of Supercomputers,'' *Performance Evaluation of Supercomputers,* J. L. Martin, ed., pp. 403-419, North-Holland, 1988.

[5] Ferrari, D., ''Workload Characterization and Selection in Computer Performance Measurement,'' *Computer*, pp. 18-24, July/August 1972.

[6] Hartigan, J. A., *Clustering Algorithms,* J. Wiley, New York, 1975.

[7] Heidelberger, P., and Lavenberg, S. S., ''Computer Performance Evaluation Methodology,'' *IEEE Transactions on Computers*, Vol. 33, No. 12, pp. 1195-1220, December 1984.

[8] Hennessy, J. L., and Patterson, D. A., ''Computer Architecture: A Quantitative Approach,'' Morgan Kaufmann Publishers, Inc., 1990.

[9] 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.

[10] 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.

[11] Martin, J. L., ''Supercomputer Performance Evaluation: The Comparative Analysis of High-Speed Architectures Against Their Applications,'' *Performance Evaluation of Supercomputers,* J. L. Martin, ed., pp. 3-19, North-Holland, 1988.

[12] Miller, E. L. and Katz, R. H., ''Input/Output Behavior of Supercomputing Applications,'' *Proceedings of Supercomputing '91,* November 1991.

[13] Moore, R., ''File Servers, Supercomputers, and Networking,'' *Proceedings of the NSSDC Conference on Mass Storage Systems and Technologies for Space and Earth Science Applications,* 1991.

[14] Pasquale, J. C., Bittel, B. K., and Kraiman, D. J., ''A Static and Dynamic Workload Characterization Study of the San Diego Supercomputer Center CRAY X-MP,'' *Proceedings of the 1991 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems,* pp. 218-219, May 1991.

[15] Serazzi, G., ''A Functional and Resource-Oriented Procedure for Workload Modeling,'' *Performance '81,* pp. 345-361, North-Holland, 1981.

[16] Stonebraker, M., and Dozier, J., ''Large Capacity Object Servers to Support Global Change Research,'' Project Sequoia 2000 Report #91/1, University of California, Berkeley, July 23, 1991.

[17] Williams, E., ''The Effects of Operating Systems on Supercomputer Performance,'' *Performance Evaluation of Supercomputers,* J. L. Martin, ed., pp. 69-81, North-Holland, 1988.

[18] Lazowska, E. D., and Sevcik, K. C., co-chairs, ''Report of the Workshop on Scientific Computing Performance Analysis,'' Division of Advanced Scientific Computing, NSF, Boulder, Colorado, August 29-31, 1898.

[19] Committee on Supercomputer Performance and Development, ''An Agenda for Improved Evaluation of Supercomputer Performance,'' National Research Council, Washington, D.C., 1986.

Appendix A: I/O Intensive Applications

Name	Frequency	Cumulative CPU Time (s)	Average CPU Time (s)	Cumulative Bytes Transferred (MB)	Average Bytes Transferred (MB)	I/O Rate Rate (MB/s)
trans.im*	30	24031.04	801.03	3164989.70	105499.66	131.70
resm*	13	19862.59	1527.89	2505657.02	192742.85	126.15
xxpd3	4	9931.87	2482.97	1071274.75	267818.69	107.86
timteb.x*	60	12569.74	209.50	1215216.92	20253.62	96.68
fdsol*	18	42196.74	2344.26	3381554.48	187864.14	80.14
crayfeat*	13	3879.31	298.41	276885.85	21298.91	71.38
feat	7	6951.01	993.00	490432.51	70061.79	70.56
ocean*	11	92899.19	8445.38	2950222.36	268202.03	31.76
Analyze*	19	7828.71	412.04	241259.24	12697.85	30.82
xm901*	21	17483.59	832.55	520723.20	24796.34	29.78
momlw.x*	29	27719.75	955.85	792660.41	27333.12	28.60
griz.exe*	12	1852.15	154.35	48857.34	4071.45	26.38
d2us5.ms	4	967.00	241.75	22861.02	5715.25	23.64
d2us4.ms	5	1472.96	294.59	34369.50	6873.90	23.33
eigen.ss*	20	8675.21	433.76	178681.92	8934.10	20.60
xmain	228	165250.74	724.78	3049029.25	13372.94	18.45
xm824*	32	5765.54	180.17	105682.45	3302.58	18.33
d2us4a.m	2	762.52	381.26	13518.34	6759.17	17.73
stress*	140	26198.42	187.13	432072.68	3086.23	16.49
tub*	20	68214.42	3410.72	1066686.37	53334.32	15.64
cgcm*	25	259833.80	10393.35	3969227.85	158769.12	15.28
forsinc3	1	220.48	220.48	3327.23	3327.23	15.09
invt.x	1	417.31	417.31	5882.37	5882.37	14.10
cxi.x*	11	4019.79	365.44	54458.16	4950.74	13.55
cfsconc*	41	119838.88	2922.90	1587134.06	38710.59	13.24
l913.exe	6	59597.73	9932.95	716211.71	119368.62	12.02
rot	8	993.19	124.15	11023.70	1377.96	11.10
ddam	4	4227.27	1056.82	46832.96	11708.24	11.08
cas2.exe*	116	144544.22	1246.07	1459567.52	12582.48	10.10
scf.exe*	34	8607.28	253.16	76458.08	2248.77	8.88
stone.exe*	13	1326.85	102.06	11240.45	864.65	8.47
dnsavg4.*	13	5664.27	435.71	43224.88	3324.99	7.63
cpmd.x*	114	55702.78	488.62	403798.65	3542.09	7.25
NAST66B	951	225568.59	237.19	1555168.13	1635.30	6.89
dir.cpx*	141	93347.47	662.04	634664.30	4501.17	6.80
em3d1.x	7	4232.06	604.58	28609.56	4087.08	6.76
m927c	2	17294.83	8647.42	115937.28	57968.64	6.70
NAST67*	343	78403.61	228.58	524921.26	1530.38	6.70
df0008.x	5	3825.57	765.11	25552.38	5110.48	6.68
l508.exe	8	16394.23	2049.28	108183.87	13522.98	6.60
cadpac4l	6	33568.18	5594.70	218562.87	36427.14	6.51
df0010.x	5	5958.71	1191.74	35748.61	7149.72	6.00
df8000.x	5	5023.10	1004.62	28982.78	5796.56	5.77
l311.exe	6	12464.29	2077.38	71682.69	11947.11	5.75
l202.exe	21	19907.45	947.97	102387.07	4875.57	5.14
df0014.x	5	7152.52	1430.50	35923.46	7184.69	5.02
rotm2	4	652.81	163.20	3065.97	766.49	4.70
l705.exe	2	5998.82	2999.41	27973.89	13986.94	4.66
% of Total Workload Resources		9.5 %		71 %		

* Applications selected for individual analysis.

Appendix B: Cluster Analysis Results

	Characteristic Application Clusters							
	Cluster		CPU Time		Logical I/O Requests		Logical I/O	Memory
Application	Size	% of Total Executions	Average (s)	% of App. Cum. CPU	Average (#)	% of App. Cum. Log. I/O	Rate Avg. (#/s)	Space Avg. (MWords)
tub	18	90.0	3788.75	99.9	309671.22	99.9	81.73	1.103
cas2.exe	102	87.9	1416.95	99.9	413907.76	99.9	292.11	0.658
cfsconc	15	75.0	7989.04	99.9	12727206.40	99.9	1593.08	0.258
eigen.ss	15	75.0	578.31	99.9	18011.00	99.9	31.14	6.532
ocean	8	72.7	11420.39	98.3	14182056.0	99.9	1241.82	0.451
resm	11	84.6	1802.84	99.8	2133282.91	99.7	1183.29	0.413
momlw.x	19	65.5	1436.79	98.4	599779.37	99.3	417.44	2.910
scf.exe	18	52.9	417.83	87.3	123979.44	98.7	296.72	0.329
Analyze	9	47.3	745.71	85.7	932004.44	98.7	1249.82	0.935
cxi.x	7	63.6	562.92	98.0	39547.57	97.4	70.25	0.776
trans.im	13	43.3	1847.13	99.9	38175.00	96.8	20.67	5.883
stress	65	46.4	381.36	94.6	40587.63	96.3	106.43	5.520
xm824	21	65.6	225.15	82.0	61893.62	96.2	274.90	2.424
fdsol	11	61.1	3356.04	87.4	1248264.73	90.8	371.95	8.074
xm901	17	80.9	947.41	92.12	358000.47	90.4	377.87	2.439
griz.exe	8	66.7	183.65	79.3	25091.75	78.4	136.63	0.595
	4	33.3	95.73	20.7	13790.00	21.6	144.05	0.595
		(100.0)		(100.0)		(100.0)		
crayfeat	6	46.1	504.97	78.1	1152473.33	79.5	2282.25	3.263
	5	38.4	169.85	21.8	354803.20	20.4	2088.92	3.243
		(84.5)		(99.9)		(99.9)		
cgcm	16	64.0	11381.39	70.0	863017.50	21.6	75.83	2.844
	3	12.0	24873.73	28.7	16575317.33	78.0	666.38	2.110
		(76.0)		(98.7)		(99.6)		
dnsavg4	2	15.3	2692.36	95.0	7035.00	72.3	2.61	19.867
	9	69.2	30.95	4.9	568.00	26.2	18.35	19.872
		(84.5)		(99.9)		(98.5)		
NAST67	145	42.2	481.46	89.0	33771.75	87.2	70.14	3.812
	143	41.6	55.61	10.1	4232.97	10.7	76.12	3.809
		(83.8)		(99.1)		(97.9)		
stone.ex	2	15.3	95.18	14.3	35090.00	15.4	368.65	0.284
	8	61.5	97.46	58.7	34837.75	61.4	357.45	0.284
		(76.8)		(73.0)		(76.8)		
cpmd.x	1	0.8	39315.13	70.5	9866752.00	43.7	250.97	3.809
	80	70.1	46.12	6.6	77188.57	27.3	1673.66	0.633
	16	14.0	793.51	22.7	405925.56	28.8	511.56	5.004
		(84.9)		(99.8)		(99.8)		
timteb.x	3	5.0	1161.60	27.7	183205.67	22.3	157.72	5.804
	22	36.6	355.58	62.2	62502.05	55.9	175.78	6.797
	24	40.0	52.36	9.9	22080.29	21.5	421.70	5.797
		(81.6)		(99.8)		(99.7)		
dir.cpx	11	7.8	5246.17	61.8	17395.36	27.7	3.32	7.403
	47	33.3	673.88	33.9	8242.87	56.2	12.23	6.631
	60	42.0	62.89	4.0	1589.13	13.8	25.27	6.945
		(83.1)		(99.7)		(97.7)		

Appendix C: Regression Analysis Results

Table C.1: Regression Analysis

Number_of_Bytes_Transferred = c + a × CPU_Time

Application	Frequency	c (MB)	*t*-stat	a (MB/s)	*t*-stat	R^2
dnsavg4	13	33.723	0.991	7.736	240.152	1.00
resm	13	5749.9	0.591	125.414	24.571	0.982
trans.im	30	1126.41	0.181	133.459	27.527	0.964
griz.exe	12	435.691	1.829	24.189	16.255	0.964
eigen.ss	20	154.74	0.234	20.734	16.535	0.938
stress	140	179.472	1.543	15.929	41.915	0.927
cpmd.x	113	1769.42	3.080	3.833	24.935	0.849
cxi.x	11	1353.4	1.554	10.169	5.678	0.782
xm824	32	2337.52	4.557	5.796	2.835	0.211
stone.exe	13	876.524	132.063	0.086	1.345	0.141
cgcm	25	-24058	-1.279	17.957	12.919	0.879
timteb.x	60	-12064.5	-2.682	156.586	23.197	0.903
fdsol	18	-17945.6	-1.246	89.716	20.051	0.962
crayfeat	13	-564.258	-0.405	74.978	22.711	0.979
ocean	11	-10027.8	-1.348	33.706	57.382	0.997
Analyze	19	-1824.62	-0.755	35.985	11.951	0.894
xm901	21	-1669.81	-0.887	32.504	15.238	0.924
momlw.x	29	1785.1	-1.914	31.149	41.911	0.985
tub	20	-1186.13	-0.189	16.36	10.77	0.866
cfsconc	21	-1672.11	-0.30	13.84	21.371	0.962
cas2.exe	116	-684.313	-5.049	10.889	142.464	0.994
scf.exe	34	-110.348	-0.748	9.532	22.317	0.940
dir.cpx	141	-1249.09	-4.726	8.848	55.377	0.957
NAST67	343	-87.988	-1.739	7.240	52.579	0.890

Table C.2: Regression Analysis – Logarithmic Transformation

Number_of_Bytes_Transferred = e^c × CPU_Time ^{a}

Application	Frequency	c	*t*-stat	a	*t*-stat	R^2
resm	13	19.202	133.020	0.931	44.018	0.994
eigen.ss	20	9.641	25.604	2.143	30.876	0.981
griz.exe	12	17.572	83.477	0.908	21.616	0.979
cas2.exe	116	12.740	88.369	1.480	66.961	0.975
momlw.x	29	9.402	22.184	2.112	30.376	0.972
cgcm	25	12.457	26.299	1.434	25.935	0.967
trans.im	30	12.315	35.575	1.860	26.530	0.962
crayfeat	13	11.285	13.615	2.205	13.478	0.943
tub	20	8.346	8.889	2.022	16.538	0.938
cfsconc	20	11.402	15.640	1.577	16.229	0.936
scf.exe	34	12.930	28.831	1.542	17.591	0.906
fdsol	18	10.147	9.460	1.941	12.357	0.905
ocean	11	9.936	5.751	1.767	8.408	0.887
cxi.x	11	12.670	12.019	1.650	8.162	0.881
dir.cpx	141	12.257	48.485	1.538	31.419	0.877
timteb.x	60	11.860	27.195	2.074	19.563	0.868
xm901	21	17.556	30.746	0.950	11.071	0.866
cpmd.x	113	12.933	44.882	1.832	24.818	0.847
xm824	32	17.371	51.965	0.880	12.437	0.838
Analyze	19	14.014	18.855	1.464	9.223	0.833
stress	140	12.399	42.245	1.782	26.165	0.832
NAST67	343	14.963	59.944	1.151	21.416	0.574
dnsavg4	13	12.158	6.189	1.759	3.786	0.566
stone.exe	13	20.553	544.015	0.011	1.297	0.133