Performance Forecasting: Characterization of Applications on Current and Future Architectures

Brian Armstrong Rudolf Eigenmann
Purdue University

January 20, 1997

Abstract

A common approach to studying future computer systems is to build simulators that accurately model the behavior of applications and architectures. While this approach can lead to quantitative information about the systems performance it has one serious drawback: simulating large applications with large data sets can be prohibitively time-consuming. In this paper we propose an alternative approach. We model the performance of an application with "Resource Usage Equations," verify these equations with acual program runs on current systems and current data sets, and then extrapolate the system behavior, scaling up architectural components and data set parameters.

1 Introduction

Issues in Characterizing the Performance of Current and Future Computer Systems

Loads placed on the system's resources must be quantified in order to determine the performance at which a given computer architecture executes an application program. Such loads include the demands on the disk IO, the communication network, the CPUs, etc.. Expressing such loads in terms of the applications' input and architecture's parameters enables us to describe the performance scalability with respect to these parameters. Of specific interest in our work is the study of very large data sets, very large numbers of processors and the impact of various interconnection topologies. This will allow us to look ahead at architectures and application problems of future computer systems.

Our approach is related to other work that provides tools and methodologies for the analysis of current and future computer systems through simulation and performance characterization. A wide range of contributions deal with issues of performance analysis and prediction for individual aspects and components of computer architecture and software. Several approaches model overall system performance. In both [SSRV95] and [AD96], the communication of realistic applications and systems is characterized, which is a basis for the performance predictability of future systems. [EP94] uses templates that reflect the control structure of parallel algorithms. Combined with hardware characteristics of synchronization and other critical operations, one can estimate the performance on an application. [GS95] uses a hardware description and a software "signature" that can be combined to describe the behavior of an application on a particular system. Our work differs in that we analyze and extract equations that describe resource

usage directly from the given applications. Both communication and disk IO are included. We put emphasis on the use of realistic applications, large data sets, and processor configurations as they may become available in the medium-term future.

An important issue is the choice of applications for such a study. The use of realistic applications is crucial for any study of software and architectures but is of even greater importance for extrapolating from currently measurable systems onto future ones. our study will consider applications that are established as industrially significant representatives of a high-performance computer workload. In doing so, we will leverage off of an effort that has already identified a suite of industrially relevant applications. This suite is called SPEChpc96 and is maintained by the SPEC/High-Performance Group [EH96] for both machine benchmarking and scientific study.

The Seismic Application

In this paper we concentrate on the seismic processing suite SPECseis96, or short "Seismic." It is a code used in the seismic processing industry for the search of oil and gas. The code consists of four applications, referred to as four "phases," which perform the seismic processes: "Data generation," "Stacking of data," "Time Migration," and "Depth migration." The entire code contains 20,000 lines of Fortran and C and includes intensive communication as well as intensive disk IO.

Several data sets are available, ranging from a very small test set to sizes that are larger than current machines can handle. The data space consists of seismic samples in traces along a number of so-called lines [MH93]. Temporary disk space is required for the file of traces, which are records of signal amplitudes at specific 3D coordinates. When executed on more than one processor, the lines are allocated into groups of traces for each processor. Thus, the data size depends on the number of samples in a trace, the number of traces in a group, the number of groups in a line, and the number of lines.

Each of the four phases of the overall application have distinct computation, communication, and disk IO characteristics. The phases are designed to execute sequentially, though the third and fourth phases can be executed simultaneously. Within each phase, a main-loop performs a series of functions on each trace. Phases 2, 3, and 4 require data to be processed and passed across the processors. However, Phase 1 requires communication only to decompose at the start of the phase and join the data at the phase's completion. Each processor writes a disjoint segment of the data file allowing for non-blocking writing.

Communication is implemented using a PVM layer and is easily captured by time-stamping the send and receive functions. Likewise, disk IO is captured using time-stamps at the beginning and end of the read and write functions. Instrumentation as described above gives the information we use to produce the runtime profiles below for the different data sizes and varying number of processors used. All time spent outside of the disk IO and communication routines is considered time spent in computation.

2 Characterization of Seismic on Current Machines

In a first step of our project we have studied the application in-depth on several architectures. In this paper we report the results obtained on a Silicon Graphics Power Challenge machine.

# P	Total	Phase			
		1	2	3	4
4	795	156	28.8	0.640	609
8	410	76.6	16.7	0.638	316
16	215	41.3	12.2	0.632	161

Table 1: Execution Time (in seconds) for the *Small* Data Set on 4, 8, & 16 Processors.

This machine includes MIPS R10000 processors and runs under the IRIX 6.2 operating system. The R10000 processors each have a peak performance of 390 MFLOPs. Up to 16 of these processors are used simultaneously in single-user mode.

We have time-stamped every message and every read to and write from the disk, thereby tracing all messages and disk IO operations that the code explicitly implements. The result is a quantitative description of the loads placed on the communication network, disk IO, and CPUs, which is the basis for accurately describing the measured performance and locating bottlenecks in the application and the architecture.

Communication Scaling with the Number of Processors

The first figure (Figure 1) displays the communication profiles for the *small* data set as the number of processors is increased from 4 to 16 processors. The profiles indicate the number of active processors, that is, processors that are *not* doing communication at each point in the program's runtime. (Disk IO was accounted for in later runs.)

The phases are distinguished from each other using different shadings. The x-axis time scale was chosen as 960 seconds for the 4-processor case, 480 seconds for 8 processors, and 240 for 16 processors. In each profile, the smallest tick-mark on the x-axis indicates 10 seconds. Numeric values for the execution times of each phase as the number of processors is increased can be found in Table 1.

The speedups of the 8 and 16-processor versions over the 4-processor execution are 1.92 and 3.7, respectively. The cause for the slightly less-than-ideal speedups becomes more evident when observing each phase separately. The speedups of the phase 1 alone are 2 and 3.84 for the 8 and 16-processor runs, respectively, which is a characteristic of a parallel program with very small overhead. This is to be expected since there is communication only at the start and completion of phase 1.

The second phase, however, gives corresponding speedups of 1.2 and 2.38. Parallelization is less effective for phase 2 since the execution time of phase 2 is much less than that of phase 1 for the *small* data set, making the communication overhead more significant in comparison to the computation time. Seismic is parallelized for large granularity message-passing systems which is also evident in the fourth phase.

Phase 3 has reached the point (at only four processors) where running it in parallel across more processors is not efficient, that is, it does not reduce the execution time significantly. This phase is not as significant in terms of execution time as the other three phases and is therefore not critical in terms of finding opportunities for overall performance improvement.

Most of the execution time of Seismic is spent in phase 4. It is more efficiently parallelized

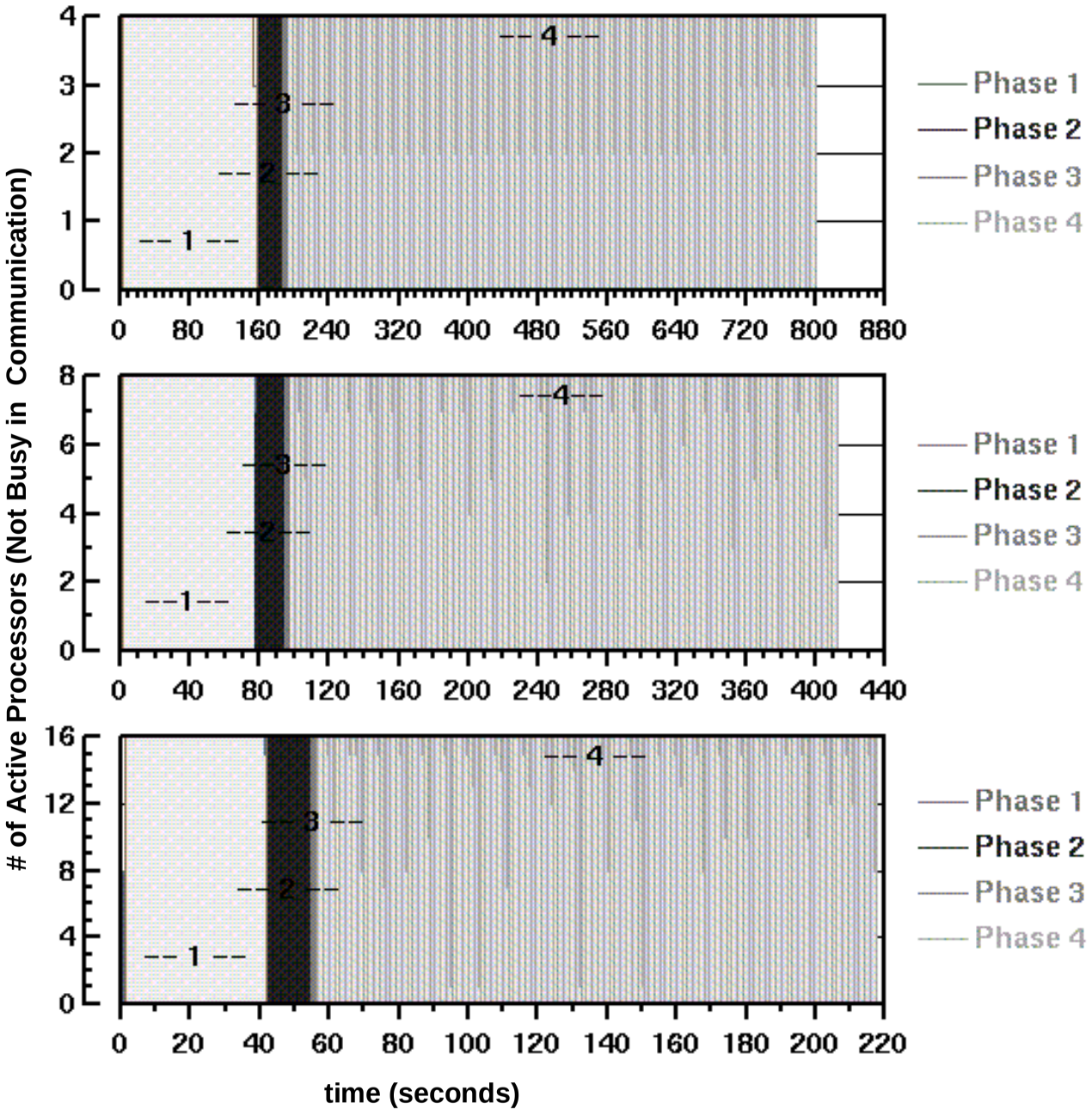

Figure 1: Communication Profiles for the *Small* Data Set Executed on 4, 8, & 16 Processors, Showing How Communication Scales with P.

Data Set	*Total*	*Phase*			
		1	*2*	*3*	*4*
Test	42	6.95	3.6	0.574	31.2
Small	215	41.3	12.2	0.632	161
Medium	6,084	664	134	1.45	5,285

Table 2: Execution Time (in seconds) for the *Test*, *Small*, & *Medium* Data Sets on 16 Processors.

Component of Execution Time	*Total*	*Phase*			
		1	*2*	*3*	*4*
$T_{\texttt{Comp}}$	25.1	5.530	2.92	0.283	16.4
$T_{\texttt{Comm}}$	3.37	0.470	0.98	0.283	1.63
$T_{\texttt{IO}}$	0.76	0.042	0.33	0.011	0.29

Table 3: Computation, Communication, & Disk IO Components of the Execution Time (in seconds) for the *Test* Data Set on 16 Processors.

than phase 2 with speedups over the 4-processor run of 1.92 and 3.84 for the 8 and 16-processor run, respectively. This indicates that phase 4 is a nearly perfectly parallel program, yet it is expected that the communication throughout the phase will have a more significant impact on the execution time than with phase 1, which has communication at only two points.

Communication Scaling with Data Set Size

The communication profiles of Figure 2 compare three runs of the benchmark for three different data sets: *test* (0.011 GB trace data file), *small* (0.22 GB trace data file), and *medium* (1.73 GB trace data file.) This profile exhibits how the benchmark's performance scales as the data set size is increased. Again, the phases are distinguished with different shadings. All runs were done using the Silicon Graphics Power Challenge with 16 processors.

Scaling the data size is important for forecasting performance, which is discussed below. The four phases vary in their significance with respect to execution time for the different data sets. (See Table 2 for numeric values from the profile graphs.) For the *test* data set, phase 1, 2, 3, and 4 account for 17%, 9%, 1%, and 74% of the execution time respectively. Similarly, the *small* data set consists of phases which account for 19%, 6%, 1%, and 75% of the total execution time. However, phase 4 is more significant in the *medium* data set, where the phases contribute 11%, 2%, 1%, and 87% of the execution time, respectively. The significance of a phase is determined by the input parameters to the application which are consistent with realistic seismic processing needs. Phase 4 is more significant in the *medium* data set than in the *test* and *small* data sets since it depends on different aspects of the data than phases 1, 2, and 3 do. Thus, the scaling of the input parameters is a deciding factor in how Seismic will perform for different data sets.

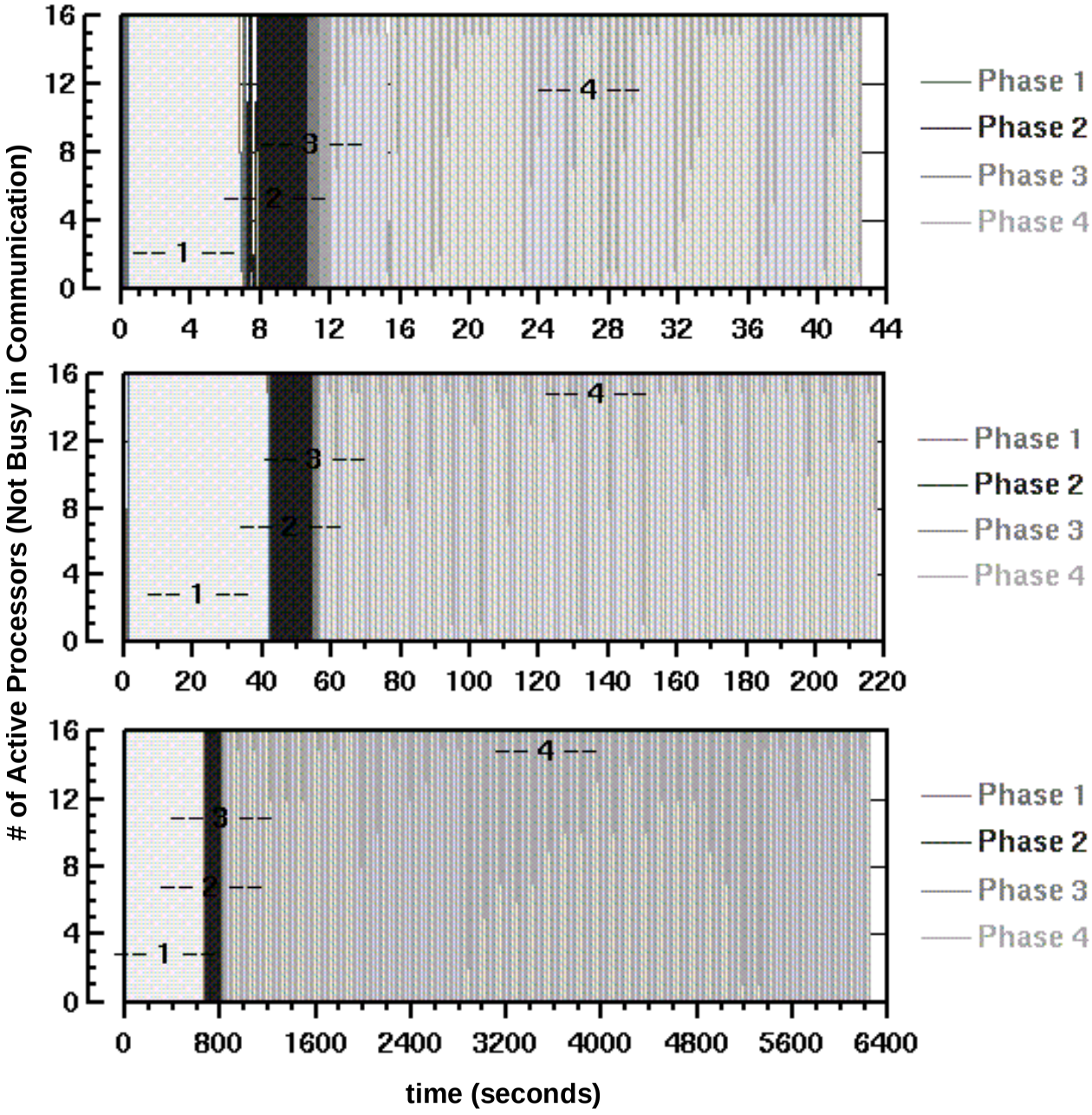

Figure 2: Communication Profiles for the *Test*, *Small*, & *Medium* Data Sets Executed on 16 Processors, Showing How Communication Varies with the Data Set Size.

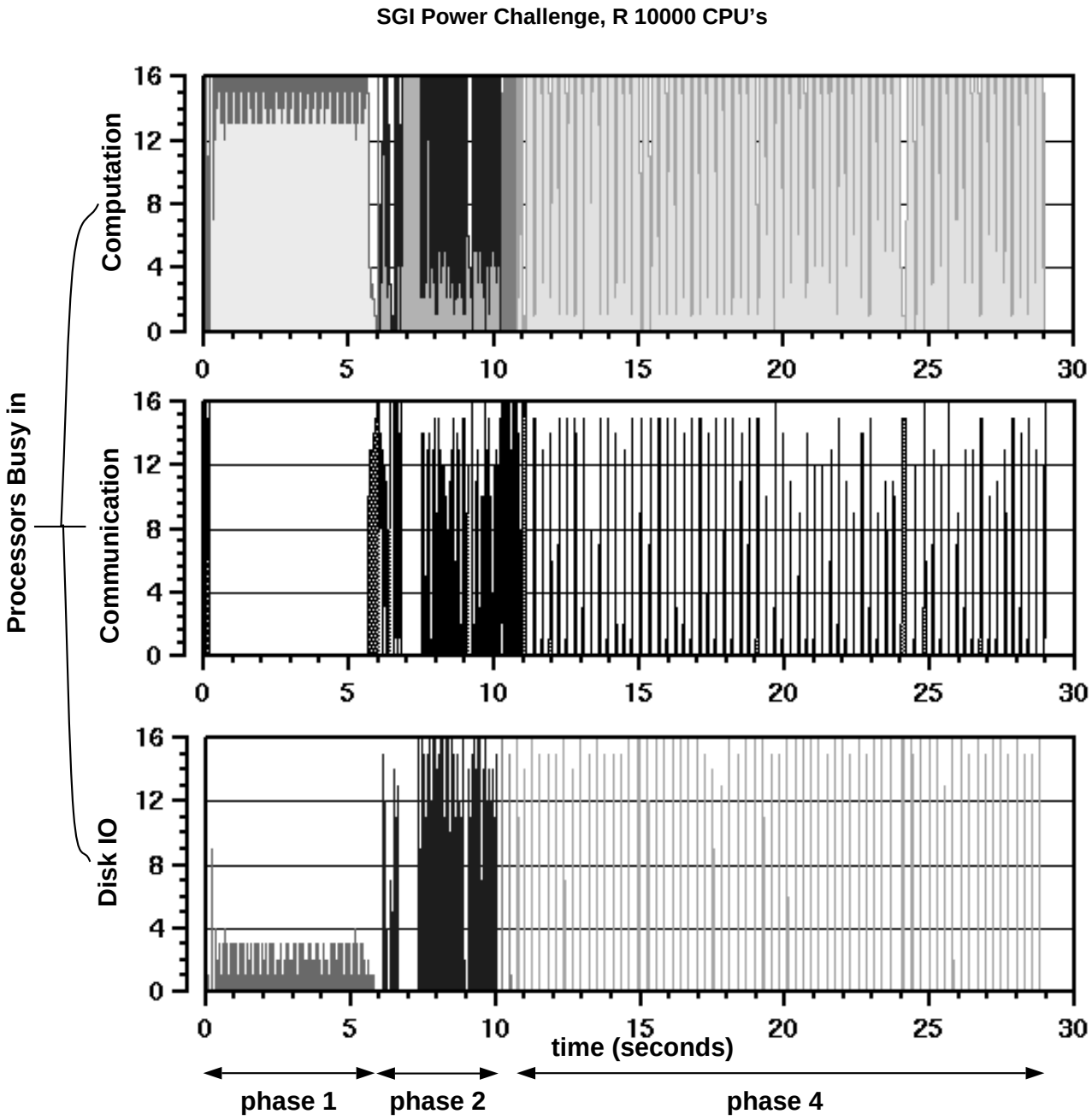

Figure 3: Computation, Communication, & IO Profiles for the *Test* Data Set Executed on 16 Processors, Separating the Components of Execution Time During the Application's Runtime.

Computation, Communication, and Disk IO Profiles

Instrumenting the disk accesses as well as the communication commands reveals how the disk IO varies across the four seismic phases. The characteristics of disk IO are unique within each phase since data is accessed across different dimensions of the trace data file (such as accessing a certain trace from every group instead of sequentially accessing all traces in a single group.) An example of results from such instrumentation is contained in Figure 3 which contains graphs for the *test* data set run on 16 processors of the Silicon Graphics Power Challenge machine.

Though disk IO for the *test* data set does not contribute much in terms of execution time for Seismic, it is interesting since the phases use significantly different types of disk IO. In the profile graphs of Figure 3, the shaded areas indicate "activity," defined as the number of processors active doing computations, communication, or disk IO at each point in the runtime of Seismic. When the number of processors doing disk IO is higher, as in phases 2 and 4 where nearly 15 processors do disk IO simultaneously throughout the phase, there is a much higher risk of contention. In contrast to the disk IO characteristics of phases 2 and 4, usually no more than three processors are active doing disk IO in phase 1. Thus, the disk IO system is a significant aspect of the execution times of phases 2 and 4, though, with the *test* data set run on a Silicon Graphics Power Challenge machine, the disk IO does not contribute significantly to the execution time of Seismic.

3 Performance Forecasting: Extrapolating to Future Computer Systems

In order to study the performance of Seismic on future architectures we have modeled the computation, communication, and disk IO amounts in terms of "Resource Usage Equations." These equations compute the program performance from computation, communication, and disk IO parts. The parts are expressed in terms of program input variables and architectural parameters. The Resource Usage Equations are the basis for our performance forecasting models. They allow us to observe the performance behavior as we vary (scale up according to seismic processing needs) the input and architectural parameters. In this section we will first introduce the Resource Usage Equations and show how we verify their coefficients with actual program runs. Then we will present several "Forecast diagrams" that discuss the behaviors of the Seismic application on very large data sets and highly-parallel architectures.

3.1 Resource Usage Equations

The total execution time of an application is divided into several parts, representing the time taken by actual computation, communication, and disk IO. We can express this as follows. (Refer to Table 4 for definitions of the variables used in the following equations.)

$$T_{\texttt{Exec}} = T_{\texttt{Comp}} + T_{\texttt{Comm}} + T_{\texttt{IO}} \tag{1}$$

These terms are discussed in the following paragraphs.

Variables	Descriptions
P:	number of processors
$T_{\texttt{Exec}}$:	total execution time including time spent in communication, disk IO, and etc.
$T_{\texttt{Comp}}$:	time spent in computation per processor for an entire seismic phase
$T_{\texttt{Comm}}$:	time spent communicating per processor for an entire seismic phase
$T_{\texttt{IO}}$:	time spent reading and writing to disk per processor for an entire seismic phase
$Its_{\texttt{Main-Loop}}$:	number of iterations of a phase's main-loop
$T_{\texttt{It-Comp}}$:	Time to do all computations required by the seismic processing functions during one iteration of a phase's main-loop
$T_{\texttt{loop}i}$:	percentage of a phase's total amount of computation spent in $\texttt{loop}_\texttt{i}$ (these are nested loops within a phase's main-loop; the values for the amount of computation come from the number of operations per iteration or from uniprocessor timings)
$param_\texttt{i}$:	value of an input parameter to the seismic application, the size of the data and the number of iterations of a phase's main-loop are also determined by these parameters
$\tau_\texttt{i}$:	constant coefficients indicating the proportion with which the loop range (of $\texttt{loop}_\texttt{i}$) depends on parameter i,
$X_{\texttt{Msgs}}$:	number of messages sent per processor during one iteration of a phase's main-loop
$Sz_{\texttt{Msgs}}$:	size of each message in bytes
$C_{\texttt{Comm-Start}}$:	communication startup latency per message in seconds per message
$C_{\texttt{Comm-Trans}}$:	communication transmission latency in seconds per byte
$X_{\texttt{Rds}}$:	number of reads done per processor during one iteration of a phase's main-loop
$Sz_{\texttt{Rds}}$:	size of each read access in bytes
$C_{\texttt{Rd-Start}}$:	startup latency per each disk read access
$Sz_{\texttt{Wrs}}$:	size of each write access in bytes
$C_{\texttt{Wr-Start}}$:	startup latency per each disk write access
$C_{\texttt{Mem-Trans}}$:	disk access transmission latency per byte (for both reads and writes)

Table 4: Glossary of Variables Used in Performance Modeling Equations.

Phase	*Subroutine*	*Loop ID*	*Loop Range*	*Amount of Computation per Iteration*
1	dgenb	1	*traces/group*	97,011 operations
		1.1	*samples/trace* $\times$ *traces/group*	5 operations
	nmocb	1	*groups/line*	13,676 operations
		1.1	*samples/trace* $\times$ *groups/line*	8 operations
	$\vdots$			
2	dmocb	1	*groups/line*	1,280 operations
		1.1	*groups/line* $\times$ *samples/trace*	5 operations
		2	*groups/line*	23,068 operations
		3	*samples/trace*	256 operations
		3.1	*groups/line* $\times$ *samples/trace*	8 operations
	dmocapr	1	*samples/trace*	15 operations
	dmocint	1	*groups/line* $\times$ *samples/trace*	90 operations
	stak-add	2	*groups/line*	2,466 operations
		2.1	*groups/line* $\times$ *samples/trace*	10 operations
	stak-save	1	*groups/line*	1,536 operations
		1.1	*groups/line* $\times$ *samples/trace*	6 operations
	pfrdc	1	*groups/line*	48 operations
$\vdots$				

Table 5: Loop-Level Components Used in the Model Computation Time.

Modeling Computation

Ideally the computation time, $T_{\texttt{Comp}}$, can be modeled by a formula whose variables are the application's input parameters. To develop such a formula, the components that make up the computation load are defined. The number of computations required by the phases of Seismic is relatively the same for each iteration of the phase's main-loop. Consequently, the computation time can be modeled as the number of iterations for the main-loop multiplied by the time spent in computation within one of the main-loop's iterations. An iteration of the main-loop consists of a sequence of subroutine calls, where each subroutine includes a set of loops.

Each loop iterates over a range of data which is determined from some of the input parameters of Seismic. Values for the average number of operations performed within a single iteration of each loop are obtained through a combination of hand analysis and a simple operation count tool.

The number of operations for each loop provides the relative importance of the loop with respect to computation time. The computation time of the entire phase is approximately proportional to the number of operations within a loop iteration multiplied by the number of iterations of the loop multiplied by the number of times each loop is executed. Table 5 contains some examples of the variables used in determining the computation time of the different phases.

$$T_{\texttt{Comp}} = Its_{\texttt{Main-Loop}} \times T_{\texttt{It-Comp}} \tag{2}$$

$$
\begin{aligned}
T_{\texttt{It-Comp}} &= T_{\texttt{loop1}} \times (\tau_{11} \times param_1 + \tau_{12} \times param_2 + \cdots) \\
&+ T_{\texttt{loop2}} \times (\tau_{21} \times param_1 + \tau_{22} \times param_2 + \cdots) \\
&+ \cdots
\end{aligned}
$$

Modeling Communication

A simple model consisting of startup and transmission latencies is used to define the time each processor spends in communication. Enhancements to the communication time equation can be made to investigate the effects of different types of network systems. To allow for networks such as the hypercube or binary tree, the average latency per message for any two communicating nodes is proportional the the log of the number of processors. Networks like 2D-meshes have average latencies per message which are proportional to the square-root of the number of processors. Network contention is not included in the current model, yet more refinement and enhancements are easy to make for obtaining more accurate approximations.

The method used in the seismic application to communicate data is an all-to-all broadcast where each processor sends to and receives from every other processor a uniformly sized message. The number of messages sent per processor is one less than the number of processors times the number of all-to-all broadcasts executed within the phase, which depends on the number of iterations in the main-loop of the phase. Therefore, the number of messages sent per processor is dependent on the number of processors and the input parameters of the application which determine the number of iterations within the phase. (The sizes of both sends and receives are consistent throughout each phase.)

$$
\begin{aligned}
T_{\texttt{Comm}} &= Its_{\texttt{Main-Loop}} \times (P-1) \times X_{\texttt{Msgs}} \\
&\times (C_{\texttt{Comm-Startup}} + C_{\texttt{Comm-Trans}} \times Sz_{\texttt{Msgs}})
\end{aligned} \tag{3}
$$

Modeling Disk IO

The model of disk IO is similar to the communication model for a bus. We have included the possibility where reads and writes differ in their access times per message, though the transmission latency per byte is considered the same for both reads and writes. The seismic code reads processed seismic traces, which were written by a previous phase, and writes to a new file. As mentioned earlier, the third and fourth phases read from the same file, which was written in the second phase. So, the third and fourth phases can be executed simultaneously. However, the reordering the phases in any other way or chaining the processing across subsequent phases is not accounted for in the design of the code. The sizes of the reads and writes are consistent throughout each phase, excluding a constant number of reads at the beginning of each phase. The size and number of the disk accesses depends on the data size, the number of processors, and whether an entire group of traces are accessed or a single trace in each group is accessed (a "transposed" read/write) each time.

$$
\begin{aligned}
T_{\texttt{IO}} &= Its_{\texttt{Main-Loop}}(X_{\texttt{reads}} \times (C_{\texttt{Read-Startup}} + C_{\texttt{Mem-Trans}} \times Sz_{\texttt{Reads}}) \\
&+ X_{\texttt{writes}} \times (C_{\texttt{Write-Startup}} + C_{\texttt{Mem-Trans}} \times Sz_{\texttt{Writes}}))
\end{aligned} \tag{4}
$$

Constants	*units*	*Value in Phase*		
		1	*2*	*4*
$C_{\texttt{Comm-Start}}$	s	0	3.18e-4	1.04e-4
$C_{\texttt{Comm-Trans}}$	s/B	0	6.62e-7	2.17e-6
$C_{\texttt{Rd-Start}}$	s	4.32e-5	8.58e-5	1.48e-4
$C_{\texttt{Wr-Start}}$	s	1.53e-3	3.04e-3	5.25e-3
$C_{\texttt{Mem-Trans}}$	s/B	2.22e-8	4.40e-8	7.60e-8

Table 6: Values for Constants in Model Equations for Communication and Disk IO.

3.2 Determining the Coefficients

The coefficients included in the Resource Usage Equations are found by fitting the values given by the model formulas with the computation, communication, and disk IO times from experimental runs. The modeled performance was compared with actual measured performance. Table 6 contains values for the communication and disk IO latencies which result from matching the formula values to experimental results. The model formulas are approximations and the constants in the formulas hide some of the actual complexity of the architecture, operating system, message-passing interface, and etc., such as the aspects of the cache which affect the disk IO and the portion of the communication latencies due to overhead of the PVM manager. The constants

are averages of the measured throughputs (messages per second, bytes per second, reads per second, writes per second, and bytes read or written per second) from actual runs of Seismic. Values for the constants are different for the different phases because the amount of computation, the types of communication and disk IO differ from phase to phase. Having higher penalties for the number of messages in phase 2 than in phase 4 is compatible with the concept of Seismic's performance on real architectures.

3.3 Performance Extrapolations

The formulas described above along with the calculated coefficients provide the tools to make approximations of the Seismic Benchmark's performance when executed with very large data sets and numbers of processors. The approximations accurately predicted the performance measurements of the experimental runs for the *test*, *small*, and *medium* data-sets. for data-sets of industrial magnitude, such as the *large* data-set (requiring nearly 20 GB of disk space for the file of traces) and the *ultra* data-set (requiring nearly 4 TB of disk space for the file of traces), we must use the formulas. The graphs of Figure 4 represent how the computation, communication, and disk IO times in phase 2 and phase 4 scale as the number of processors is increased for each of the six data sets included with the Seismic Benchmark: the *test*, *small*, *medium*, *large*, *xlarge*, and *ultra* data sets. The data set size ranges from 17 MB to 3.9 TB and the numbers of processors ranged from 1 to 2,048. The graphs give time in seconds and the shaded regions are the order of magnitude of the time (shaded region of value 2 means 1×10^2 seconds.) The darker shades are where combinations of data size and number of processors result in faster times than in the lighter shades. For example, communication time increases as the number of processors is increased for a given data size.

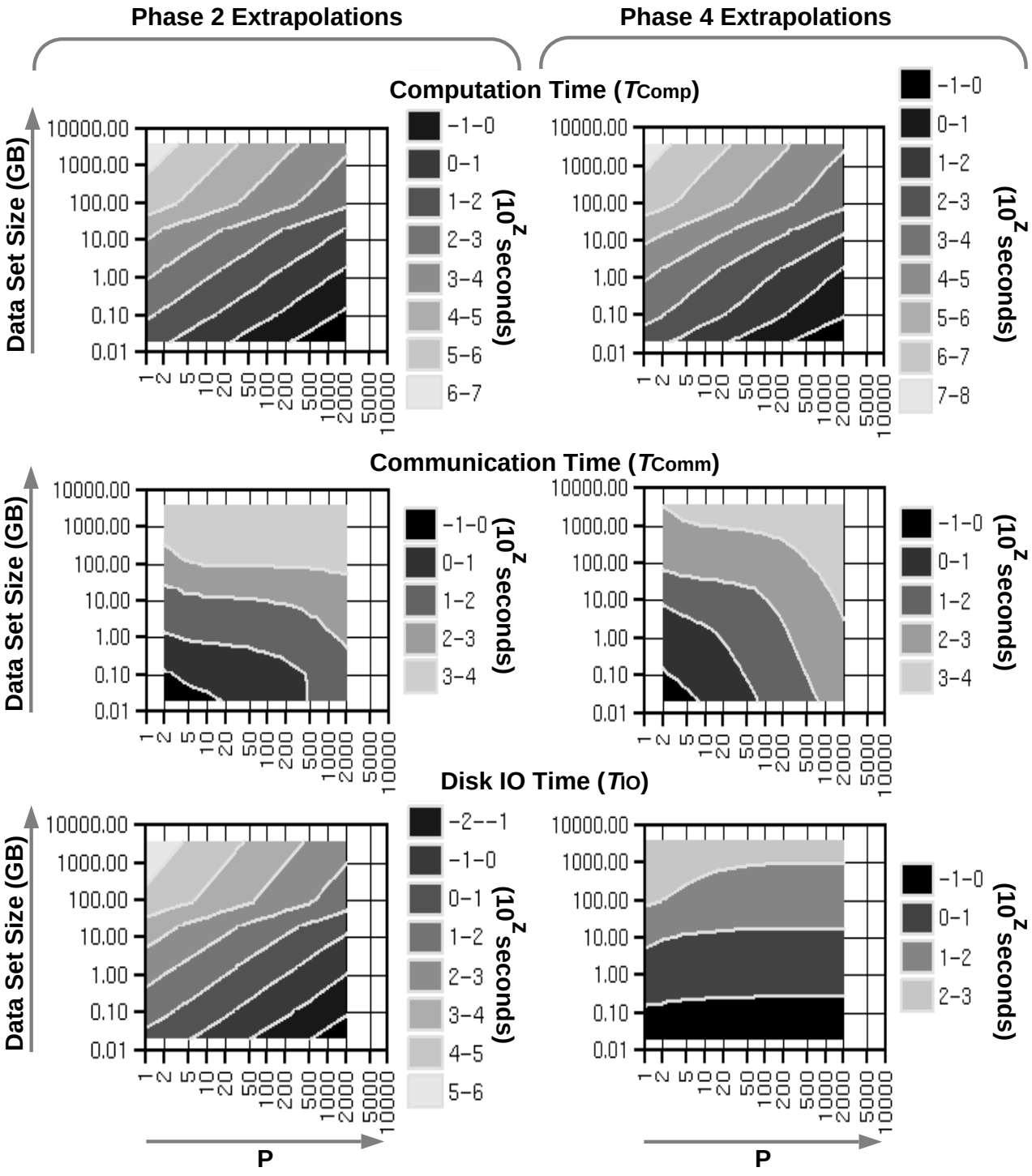

Figure 4: Extrapolations of Seismic Phases 2 & 4 for Larger Data Sets on Large Numbers of Processors.

The computation diagrams exhibit ideal behavior. That is, time decreases with an increase in the number of processors. This is expected, as the model describes the actual computation only, where the overhead components of communication and IO are separated out.

The communication graphs show that communication time increases with increasing processor numbers for both phases, even when the data set is held constant. This is because the application uses all-to-all broadcasts, which increase the communication volume as processors scale up. The communication overhead starts out lower for phase 4 than for phase 2 (which agrees with the better speedups that we have measured for phase 4), but then increases more quickly for increasing processor numbers in the larger data sets. This is due to the way in which computation is partitioned in the two phases. With increasing processor numbers, phase 4 reduces the message sizes but leaves the number of broadcasts per processor unchanged. In contrast, phase 2 reduces the number of broadcasts as well as the message sizes. As a consequence, for very large numbers of processors the large number of messages of phase 4 become small and hence latency-dominated, which increases the overall communication overhead much more significantly than in phase 2.

The same data partitioning characteristics of the two phases affect the disk IO overhead. Phase 2 shows "good" behavior of the IO time. It is similar to the computation part, which speeds up as the number of processors increase. However, as the number of processors is increased in phase 4, the static number of increasingly smaller IO requests limit any speedup with respect to disk IO time. After a certain point, the disk IO time remains constant for a given data size. This point is higher for larger data sets.

4 Conclusions

We have developed "performance forecast models" that can show the behavior of an application on very large numbers of processors and data sets. The models are based on "resource usage equations". They describe computation, communication, and disk IO in terms of numbers of processors and input data size. Based on these models we have shown and discussed the behavior of an industrial seismic processing application as processors and data scale to high numbers. The same tools can exhibit the performance behavior of different network topologies and parameters.

Our forecast models represent a methodology of evaluating future applications on future computer systems. They complement the more common simulation methods of analyzing the detailed behavior of a proposed architecture. For example, in our application analysis we have found high overheads in the communication and disk IO behavior for large numbers of processors. Neither simulation nor running the application on available machines could have feasibly investigated these situations. However, our forecast models are more abstract than simulation methods. The accuracy of simulators is traded for the ability to look ahead at future systems that may be built in generations to come. The refinement of our models (e.g., including cache resource equations) and the combination with actual simulators is an ongoing project. This will allow us to combine the advantages of detailed system analysis and performance predictability of large systems. Specifically, we will use these methods in the design of a "petaflop" architecture [MEFT96].

References

[AD96] Gheith A. Abandah and Edward S. Davidson. Modeling the communication performance of the IBM SP2. In *Proceedings of the 10th International Parallel Processing Symposium (IPPS'96)*, April 1996.

[EH96] Rudolf Eigenmann and Siamak Hassanzadeh. Benchmarking with real industrial applications: The spec high-performance group. *IEEE Computational Science & Engineering*, 3(1):18–23, Spring 1996.

[EP94] G. R. Nudd E. Papaefstathiou, D. J. Kerbyson. A layered approach to parallel software performance prediction: A case study. Technical Report CS-RR-262, Department of Computer Science, University of Warwick, Coventry, UK, 94.

[GS95] J. Gustafson and Quinn Snell. HINT: A new way to measure computer performance. In *Proceedings of the Twenty-eight Annual Hawaii International Conference on System Sciences*, volume II, pages 392–401, 95.

[MEFT96] Zina Ben Miled, Rudolf Eigenmann, José A. B. Fortes, and Valerie Taylor. Hierarchical processors-and-memory architecture for high performance computing. In *Proc. of Frontiers'96 Conference*, Oct 96.

[MH93] C. C. Mosher and S. Hassanzadeh. ARCO seismic processing performance evaluation suite, user's guide. Technical report, ARCO, Plano, TX., 1993.

[SSRV95] A. Sivasubramanian, A. Sigla, U. Ramachandran, and H. Venkateswaran. On characterizing bandwidth requirements of parallel applications. In *Proc. of the ACM SIGMETRICS Conf. on Measurement and Modeling of Computer Systems*, may 1995.