INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

SCALABILITY ANALYSIS OF LARGE CODES USING SYNTHETIC PERTURBATIONS AND FACTORIAL DESIGNS

A Dissertation

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

degree of

Doctor of Philosophy

By

Mohammed A. M. Al-Abdulkareem

Norman, OK

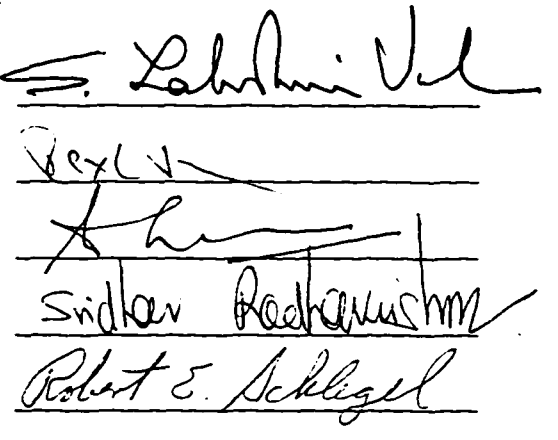
1997

SCALABILITY ANALYSIS OF LARGE CODES USING SYNTHETIC PERTURBATIONS AND FACTORIAL DESIGNS

A Dissertation APPROVED FOR THE

SCHOOL OF COMPUTER SCIENCE

BY



To my respected parents for their support and prayers.

To my wife Muna for her continuous encouragement, sacrifice, and love.

And to my little lovely daughters Sundos, Haneen, and Shaza.

ACKNOWLEDGMENTS

I wish to record my thanks and gratitude to my teacher and advisor Dr. S. Lakshmivarahan, George Lynn Cross Research Professor, for his scientific guidance and continuous encouragement. I will never forget his kindness and help during the years I spent at the University of Oklahoma.

I thank the members of my advisory committee Professors Sudarshan K. Dhall, Rex Page, Sridhar Radhakrishnan, and Robert Schlegel for their support, and valuable suggestions and discussions. My thanks are also due to Dr. Gordon Lyon, NIST for suggesting the approach described in this work and for his continued guidance.

ARPS model code used in this analysis is under development at the Center for the Analysis and Prediction of Storms (CAPS), an NSF Science and Technology Center at the University of Oklahoma.

Thanks to the Environmental Computing and Applications Systems at the University of Oklahoma (ECAS) for providing the access to CRAY J90. I also thank the Department of Physics and Astronomy at the University of Oklahoma for providing the access to the IBM SP2 used in this study. Mr. Jay Liang the ECAS manager and Mr. Andy Feldt the systems and network manager for the Department of Physics and Astronomy, were very cooperative and helpful during my work on the CRAY J90 and IBM SP2. The author was supported by a grant from King Saud University, Riyadh, Saudi Arabia.

CONTENTS

LIST OF TABLES

LIST OF ILLUSTRATIONS

ABSTRACT

The issue of code's scalability is becoming more crucial with the existence of advanced scalable architectures. While speedup relates the reduction in time when going from serial to parallel computation, scalability focuses on the overall performance resulting from the increase in problem size and the number of processors.

Scalability will be limited by serial bottlenecks in the code. Locating these bottlenecks in parallel environment is not trivial. We used factorial designs to estimate empirically an approximation of a multivariate Taylor's expansion for the code's execution response function. The first order terms in the Taylor's expansion function correspond to the suspected bottlenecks and scale factors. The coefficients of these terms are estimates of the code's sensitivity to changes in these suspected locations and scale factors. The higher order terms are utilized as informal relative indicators of the code's scalability. This approach was applied to a large meteorology code running on the CRAY J90 and the IBM SP2 scalable distributed memory machine.

A class of interaction plots using speedup is introduced in this dissertation that will enable the investigator in comparing the scalability of two parallel systems.

GLOSSARY

θ	actual or real response
μ	the estimated global mean
ε	uncontrolled noise
β_i	main effect of factor F_i
$\beta_{i,j}$	interaction effect of factors F_i *and* F_j
β_{ij}	interaction effect of factors F_i *and* F_j
E_p	efficiency
F	a factor in factorial experiment
F_i	a factor i in factorial experiment
f_i	level of factor F_i
l	number of levels of factor F
l_i	number of levels of factor F_i
n	number of treatments
N	problem size
P	number of processors
p	time for parallel algorithm
p'	time for parallel portion
R	measured response
r	number of replications
Ri	the i replicate of the measured response R
R_i	the measured response at treatment I
s	time for serial algorithm
$S(N)$	asymptotic speedup
s'	time for serial portion
s^2	estimated sample variance

SE estimated standard error

S_P speedup

t a treatment or a combination of levels

$T(N)$ time required by serial algorithm to solve a problem of size N

t_i treatment i in a factorial experiment

$T_P(N)$ time required by parallel algorithm to solve a problem of size N using P processors

$T_{Par}(N)$ asymptotic minimum serial time

$T_{Seq}(N)$ asymptotic serial time

CHAPTER 1

INTRODUCTION

One of the main goals of parallel processing is to solve large problems in time much shorter than that for the serial processing. To achieve this goal, the effect of the serial part of the parallel program needs to be minimized. The notion of speedup, which is defined as the ratio of the best known serial time to the given parallel time, was used to assess parallel code. Amdahl's law 1967, see (Lakshmivarahan and Dhall 1990), was the first step on the way to analyze parallel code. The notion of *scalability* is used to measure how good a parallel system is when the system size is increased. The system size is increased usually by using more processors to solve a problem. An ideal parallel algorithm will have a speedup proportional to the number of processors. However, this is not the case in real life applications. A parallel program will reach a limit, called *parallel balance point*, after which adding more processors will increase the time taken to solve a fixed size problem (Wilson 1993).

In scalability studies, three types of scalability are often defined. One is called the *machine scalability*, the second is the *algorithm scalability*, and the third is the *scalability of a parallel system* consisting of algorithm-machine combination. Machine scalability was defined based on the asymptotic speedup for a given algorithm and problem size on the architecture under study. The asymptotic speedup is the best achievable speedup using unlimited number of processors (Nussbaum and Agarwal

1991). However, scalability of the machine may include many factors other than speedup. Cost, addressing, communication, and physical attributes should be considered to meet the machine's scalability requirements (Gustavson 1994).

When studying the algorithm scalability the overhead related to the target machine will be described by a general function. This function will either depend on a general model or some specific assumptions about the hardware. In the literature many parallel hardware models are available such as the PRAM (Parallel Random Access Machine), BSP (Bulk-Synchronous Parallel), and LogP (Latency, Overhead, Gap, and Processor count).

There are different approaches to quantify the parallel system scalability. Grama et al. 1993, defined scalability as the ability of parallel system to increase speedup as the number of processors increases (Grama, Gupta, and Kumar 1993). Following an analytical approach, they introduced the *isoefficiency function* that relates problem size to the number of processors required to keep the efficiency fixed. The isoefficiency function provides the rate at which the problem size must increase in order to keep the efficiency fixed as the number of processors increased. As a measure of scalability, a system with small isoefficiency function is more scalable.

The *isospeed* metric was proposed by (Sun and Rover 1994). Based on this metric the scalability of parallel algorithm-machine combination was defined. An algorithm-machine combination is scalable if the achieved average speed remains

constant when the number of processors is increased assuming the problem size can be increased.

In another approach, *Network latency* was used to measure and evaluate the scalability of parallel programs and architecture (Zhang, Yan, and Mia 1994). The evaluation of the scalability can be used to predict the performance of large problems on large machines. The scalability of a parallel system at a fixed efficiency level for two machine sizes is the ratio of the smaller machine's latency to the latency of the larger machine. The efficiency is kept at fixed level by scaling the problem size.

The analytical approach proposed by (Tambouris and Santen 1995), consists of six steps methodology to study the scalability of a parallel system. These steps include analysis of the parallel system, construction of asymptotic performance models, and use of these models for first order approximation of scaling behavior.

The approach followed by Sivasubramaniam et al. 1994, used an execution-driven simulator to study the scalability of five applications. They defined the notion of *overhead functions* associated with algorithmic and architectural characteristics to study the scalability.

A promising technique was suggested by Lyon et al. 1994, for tuning parallel code by identifying the program bottlenecks (Lyon et. al. 1994,1995). This technique treats the code as a black box with number of input parameters and a measured output. An approximation of the multivariate Taylor's expansion for the code's execution

response function was estimated by using statistically designed experiments. While this technique is in use for industrial processes, it was not used for computer programs because of the lack of natural parameters.

The new approach is to incorporate artificial parameters into the program text (Lyon et al. 1994). These parameters can be delay routines inserted in the code to simulate changes in code's performance. The number of processors used to execute the code is another parameter to be considered. We used the above approach to analyze the ARPS (Advanced Regional Prediction System) code.

ARPS is a non-hydrostatic atmospheric prediction model appropriate for use on scales ranging from few meters to hundreds of kilometers. The governing equations of the atmospheric model components of the ARPS include momentum, heat, mass, water substances, and the equation of state. ARPS solves prognostic equations for *x, y, and z* components of the Cartesian velocity, the perturbation potential temperature, perturbation pressure, and six categories of water substance. More details about ARPS can be found in (Droegemeier et al. 1992), (Xue et al. 1995) and (Sathye et al. 1995). The ARPS source code consists of 300 subroutines and functions, developed over the past six years by about 30 scientific and support personnel. The analytical approach will be expensive and cumbersome to apply for such huge code. For this application the reduction of overall wall-clock run time is the primary goal. For practical operation the system should run faster than the speed of the weather.

In this study experimental design techniques were employed to measure sensitivity of ARPS code to changes in performance. The analysis of the effects of these changes will point out the primary locations of code to optimize. The experiments in this work were conducted on a CRAY J90 and on IBM SP2. The results of these experiments are covered in Chapters 4 and 5. The experiments were classified into two major parts. The first part scales problem size and the second part scales the machine size in terms of the number of processors. In Chapter 2, a review of the literature related to this work is provided. A preliminary introduction to experimental design techniques used in this study and how it was applied to computer systems are presented in Chapter 3. Concluding remarks are contained in Chapter 6.

CHAPTER 2

LITERATURE REVIEW

In this chapter a review of literature related to the performance and scalability of parallel systems is presented. This chapter will give an overall view of definitions, metrics, and approaches used to assess the parallel systems.

2.1 Speedup as a Performance Measure

The most widely used measure to characterize a parallel algorithm is the speedup ratio that relates the required time to solve a given problem using the best known serial algorithm to the time required to solve the same problem by parallel algorithm using P processors (Lakshmivarahan and Dhall 1990). In the literature there exist many variations of speedup definitions.

2.1.1 Fixed Size Speedup

Let $T(N)$ denotes the time required by the best known serial time to solve a problem of size N, and the time required to solve the same problem size on a parallel machine with P processors by a parallel algorithm is $T_P(N)$, then

(2.1) $$S_P = T(N)/T_P(N)$$

It is clear that speedup is a function of N, the problem size, and P, the number of processors. Under the normal conditions, which assumes a *paracomputer* consisting of a set of identical processors each with its own memory, the speed up is bounded by the

number of processors, that is $S_P \leq P$. However, in the literature examples can be found for $S_P > P$, see (Sun and Zhu 1995). This is known as *superlinear speedup*, that may be encountered when the serial algorithm requires time longer than $T_P(N)/P$. The cause of this problem could be an enhancement in the parallel system that is not available when using single processor, or using a single processor exhibits a very low performance resulting from data access from secondary stoarge. As an example the KSR-1 shared virtual memory show a superlinear speedups because of the longer access time to the remote memory as reported by (Ramachandran, Shah, and Ravikumar 1993).

2.1.2 Other Fixed Size Speedup Definitions

In the definition of speedup we used the time required the best known serial algorithm to measure the serial time. However, the best known algorithm at the current time could be replaced by a better serial algorithm in the future. On the other hand, many researchers will use the serial version of the parallel algorithm instead of spending time developing the best serial algorithm. When the same algorithm is used to measure the serial and parallel time the speedup is called *relative speedup*, while the original definition that uses the best known serial algorithm is called *real speedup* (Sahni and Thanvantri 1996). Following these two definitions the *absolute speedup* is defined as the ratio of the required time to solve a problem of size N by the best known serial algorithm using the fastest processor, to the time required to solve the same problem

size by a parallel algorithm using P processors. The problem now is not only to find the best known serial algorithm, but to use the fastest processor as well.

2.1.3 Scaled Size Speedup

The fixed size speedup may be suitable for some applications. While in other cases it is desirable to increase the problem size with the increase of the number of processors. The fixed size speedup was governed by Amdahl's law 1967.

Let s denotes the amount of time spent by the serial portion of the algorithm, and p denotes the amount of time spent by the parallel portion of the algorithm. Then the speedup by Amdahl's law

(2.2) $$Speedup = (s+p)/(s+p/P)$$

$$= 1/(s+p/P).$$

Amdahl's law bounds the speedup by $1/s$, so the smaller this fraction the higher the speedup regardless of the number of processors used. In 1988, the definition of the *scaled speedup* was introduced by (Gustafson 1988) and (Gustafson, Montry, and Benner 1988). Let s' denotes the time spent on the serial portion, and p' denotes the time spent on the parallel portion of the parallel algorithm using P processors, then using one processor the time spent by the algorithm is $(s'+p'P)$. The scaled speedup then is given by

(2.3) $$Scaled\_speedup = \frac{(s'+p'P)}{(s'+p')}$$

When the problem size increased the serial portion s' is relatively decreased. It should be noticed that in most cases the scaled problem size can not run on single serial processor because of memory limitations. Hence, instead of using the measured serial time, it is estimated by measuring interprocessor communication time and idle time on each processor. The results presented in (Gustafson, Montry, and Benner 1988) were derived using estimations of the serial time.

2.1.4 Time Constrained Speedup

The above definition of scaled speedup allows the problem size to scale to fill the available memory. There was no limitations on the execution time. In some applications there is an upper bound on execution time. Using the time constrained it may be desirable to estimate the limit of the problem size to be solved using P processors. Examples of deriving time constrained models for some algorithms are in (Worley 1990). However, the analysis was not very realistic since many costs were ignored in deriving such models.

2.1.5 Average Speedup

For randomized algorithms the case may be slightly different. The serial and parallel running times are described by two random variables and the speedup would be the ratio of the expected values of these two variables. Thus if $T(N)$ and $T_P(N)$ are the random variables of the running time on one and P processors respectively, then the *speedup of the average running times* is given as

(2.4) $$\textit{Speed up of the average running times} = \frac{E(T(N))}{E(T_P(N))}$$

The above ratio is of use only if the we have multiple runs and the total running time of all runs is of interest. If we have only one run then the speedup will depends greatly on the distribution of the random variables (Ertel 1994).

2.2 Efficiency and Other Measures of Performance

While speedup measures how much faster the algorithm is capable to run on more than one processor, the *processor efficiency* measures how the processor are utilized. The efficiency is defined as the ratio of the speedup to the number of processors so we may write

(2.5) $$E_P = S_P/P$$

The efficiency could be used as a measure of the wasted processor cycles. In parallel computers not all processors are busy all the time. There will be usually a tradeoff between speedup and efficiency. Finding bounds on such tradeoffs was presented in (Eager, Zahorjan, and Lazowska 1989), in their work the *average parallelism* was used to characterize these tradeoffs. The average parallelism could be simply defined as the average number of processors that are kept usefully busy during the execution time of a given algorithm using unlimited number of processors.

Another factor that can be used to characterize performance is the *redundancy*. As it has been observed, the development of parallel algorithms introduce extra scalar computations to achieve higher speedup. When the communication cost is higher than

computations extra code will replace communication code by local computations. The redundancy factor R_P, is defined as the ratio of the total scalar operations performed by a P-processor parallel algorithm to the total scalar operations of serial algorithm (Lakshmivarahan and Dhall 1990).

2.3 The Scalability

The word scalable, as explained in the dictionary, is an adjective of something that can be scaled. However, in parallel processing it is used to describe a machine, an algorithm, or an algorithm-machine combination. In the following sections various definitions of scalability are presented along with a review of some approaches used to study scalability. The approach we followed in studying scalability is introduced in this chapter and explained in more detail in Chapter 3.

2.3.1 Machine Scalability

The designers, as well as the users, of parallel machines looked at the scalability of the machine and how it will affect the performance (Ramachandran et al. 1993), (Liotopoulos 1994), (Marenzoni 1995), and (Koufaty et al. 1996). While for sequential machines the effect of adding more memory or replacing an existing processor with a faster one is evident. For parallel machines the effect of adding more memory to accommodate larger problem sizes or increasing the number of processors may not be obvious. Many aspects need to be considered in parallel environments before scaling the parallel machine to keep the performance from decreasing. In many cases the scalability need to be addressed for general purpose machines rather than for

machines designed with specific application in mind. For example in distributed systems, partitioning the jobs and balancing the load is an important factor that affect the scalability (Kremien 1995).

M. Hill 1990, asked the question "What is Scalability?" and addressed many difficulties in finding a crisp definition of machine scalability. One of these difficulties is the absence of a reference model architecture to compare the architecture under study with (Hill 1990).

Besides the system's performance, many other factors need to be considered when examining scalability of parallel machines (Gustavson 1994). The cost of scaling the machine in the future need to be considered. The physical limitations of the technology used is another important factor when designing parallel computers. Addressing is another limitation problem when scaling a machines. The use of hierarchical variable length addressing may solve this problem, however, for a tightly coupled multiprocessors computers this may not be an efficient solution.

Nussbaum and Agarwal 1991, based their definition of parallel machine scalability on *asymptotic speedup*. For a given algorithm, architecture, and a problem size the asymptotic speedup, denoted $S(N)$, is the best speedup that can be attained by varying only the number of processors. Let N be the problem size, $T_{Seq}(N)$ is the asymptotic serial running time, and $T_{Par}(N)$ is the asymptotic minimum parallel running time, then

(2.6) $$S(N) = T_{Seq}(N)/T_{Par}(N).$$

The parallel time used above $T_{Par}(N)$, is calculated for a given parallel algorithm using problem size N, without limitation on the number of processors. In other words it is the minimum achieved running time using as many processors as needed.

The scalability of parallel machine $\Psi(N)$ is then defined as the ratio of the asymptotic speedups on the real machine (under investigation) and the ideal realization of an EREW PRAM. Let $S_R(N)$ and $S_I(N)$ be the asymptotic speedups for the given real architecture and for the ideal machine respectively, then

(2.7) $$S_R(N) = T_{SerR}(N)/T_{ParR}(N)$$

(2.8) $$S_I(N) = T_{SerI}(N)/T_{ParI}(N)$$

(2.9) $$\Psi(N) = S_R(N)/S_I(N) = T_{ParI}(N)/T_{ParR}(N).$$

2.3.2 Algorithm Scalability

When studying parallel algorithm scalability it is hard to isolate the analysis of the algorithm from the target real or virtual machine. Analysis of sequential algorithms is based on the von Neumann model. However, the parallel environment is lacking a realistic widely accepted unified model. Thus before analyzing the scalability of a given algorithm certain assumptions about the target hardware should be stated. The work in (Muller-Wichards and Ronsch 1995) is an example of such analysis. The timing model they used was

(2.10) $$\tau(P,N) = \alpha(N) + \frac{\beta(N)}{P} + \sigma(P,N)$$

where $\alpha(N)$ and $\beta(N)$ are proportional to the sequential and parallel amount of computation respectively. And σ denotes an overhead function of P and N.

2.3.3 Scalability of Parallel Systems

Probably the most common practice is to study the scalability of a parallel system which is a combination of an algorithm and a given machine examples of such studies can be found in (Hanebutte, Joslin, and Zubair 1994), (Johan et al. 1994), (Gupta, kumar, and Sameh 1995) and (Barragy, Carey, and De Geun 1995). Many approaches were presented in the literature to describe and evaluate scalability of parallel systems. When studying the scalability of parallel systems three important factors should be considered. The first is the parallel machine and its architecture. The second is the algorithm, and the third is the problem size. We will use the following informal definition of scalability for a parallel system.

Definition 2.1: A parallel system consisting of an algorithm A with problem size N running on a parallel machine M with P processors is said to be scalable if the performance will not decrease by increasing the number of processors to $P' > P$.

In the above definition the performance will be measured by the running time. It should be noticed that all factors are kept the same except the number of processors. In some cases the problem size may be increased to keep the performance from decreasing. In that case the parallel system is not scalable under the above definition.

For some scientific applications the researcher may desire to increase some parameters of the algorithm to achieve higher accuracy. In this case it could be

considered as an increase in the problem size. A methodology followed by Singh, Hennessy, and Gupta 1993 for scaling scientific simulation programs on parallel computers, was to scale some parameters that affect the sources of simulation errors so their error contribution is equal when larger number of processors are used. This requires an understanding of the relationship between the program parameters in terms of their error contribution.

2.3.3.1 Isoefficiency

An analytical approach to measure the scalability of parallel system was introduced by (Grama, Gupta, and Kumar 1993). They defined scalability as the system's ability to increase speedup as the number of processors increased. The *isoefficiency function* was defined and used as a measure of scalability which relates the problem size to the number of processors to keep the efficiency at a fixed level. The problem size was defined in terms of the total number of basic operations instead of input size. Let W be the total number of basic operations and t_c is the time required for each basic operation, then the serial time

(2.16) $$T_1 = Wt_c$$

Let T_o denotes the overhead time and T_P the parallel time using P processors then

(2.17) $$T_P = (T_1 + T_o)/P$$

The speedup, as in equation (2.3), is then given by

$$S_P = T_1/T_P$$

(2.18) $$= PT_1/(T_1 + T_o)$$

the efficiency is then computed from (2.5) by using (2.18) for speedup which yields

$$E_P = \frac{T_1}{T_1 + T_o}$$

(2.19) $$= \frac{1}{1 + T_o/T_1}$$

by replacing T_1 in (2.19) by its equivalent from (2.16) we get

(2.20) $$E_P = \frac{1}{1 + T_o/Wt_c}$$

(2.21) $$\frac{T_o}{W} = t_c\left(\frac{1 - E_P}{E_P}\right)$$

(2.22) $$W = \frac{1}{t_c}\left(\frac{E_P}{1 - E_P}\right)T_o$$

or it can be written as

(2.23) $$W=KT_o$$

where $K=E_P/t_c(1-E_P)$. The formula (2.23) relates the amount of work W to the efficiency E_P, so with some algebraic manipulation it can be used to relate W to the number of processors P. As an example, the sum of n numbers will require n operations on a sequential machine (indeed $n-1$ operations are needed, but for large n the difference is small) and the sequential time will be $T_1=nt_c$. If we add n numbers on a parallel machine using P processors, then we may assign n/P numbers for each processor (assuming n is divisible by P). The parallel time will be the time required to add the numbers on each processor, that is n/P, plus the overhead time to add the partial sums $\log P$ ignoring the communication time. So parallel time $T_P=n/P+\log P$

and the total overhead time T_o=log$P$. By substituting in (2.23) we get W=Klog$P$. The isoefficiency function for this system is Θ(log$P$), that means when increasing the number of processors form P to P', the problem size n need to be increased in the rate $\frac{\log P'}{\log P}$ to keep the efficiency fixed. When using two algorithms the scalability is compared by comparing their isoefficiency functions. This approach may not be adequate for some applications since finding an isoefficiency functions is not as simple as in the above example. Moreover, this function will tell how the problem size should be increased, when used to compare two algorithms will tell which one is scalable in terms of efficiency, but ignoring which one will require longer time. In practice changes in the number of processors and the problem size may introduce some variations in the measured run time see (Gupta, Kumar, and Sameh 1995), (Jamieson, Khokhar, and Patel 1995) and (Sahni and Thanvantri 1996).

2.3.3.2 Isospeed

Another analytical approach to study the scalability of a parallel system used the *isospeed* metric (Sun and Rover 1994). Their definition of scalability was based on the *average unit speed* or the *average speed* definition.

Definition 2.2: The average unit speed is the achieved speed of a given computing system divided by P, the number of processors.

Definition 2.3: An algorithm-machine combination is scalable if the achieved average speed of the algorithm on the given machine can remain constant with increasing numbers of processors, provided the problem size can be increased.

In the above definition of scalability the problem size is used equivalently to W the amount of work assuming a relation could be derived between the two. As it was the case with isoefficiency the problem size need to be increased to maintain the same average speed. Let W be the amount of work of an algorithm when P processors are used, and W' is the amount of work of the same algorithm when $P' > P$ processors are used to keep the same average speed then the scalability is

$$\psi(P,P') = \frac{P'W}{PW'} \tag{2.24}$$

where W' is determined by the isospeed constraint. The ideal case when $W' = P'W / P$, the scalability $\psi(P,P') = 1$. But in practice the $W' > P'W / P$ which results in scalability $\psi(P,P') < 1$. One way to find required W' to keep the average speed constant is to have a control program that run the algorithm under investigation and increase the problem size until the desired average speed is reached. If a relation can be found between the initial average speed and the amount of work then it can be utilized to predict scalability under the assumption that it will hold true for larger system sizes.

2.3.3.3 Other Approaches to Analyze Scalability

A memory-constrained scalability metric was proposed by (Fienup and Kothari 1994). Similar to the isoefficiency function, an asymptotic function will indicate the scalability of a parallel system. The CMP (Constant Memory per Processor) scalability is defined as the function that describes the asymptotic growth of *CMP_speedup(P)* as P goes to

infinity. If the function goes to infinity as P goes to infinity then the algorithm is said to be scalable. The *CMP_Speedup(P)* is defined as a function of P while the size of the local storage used per processor is constant.

An experimental approach was followed by (Zhang, Yan, and Ma 1994). In that approach they used a measurement of the network latency to evaluate the scalability. They defined the *average latency L(W,P)* as the average amount of overhead time needed for each processor to complete the assigned work. This latency is a function of the problem size W and the machine size P. On the contrast of isoefficiency and isospeed, since different implementations of an algorithm may have different impacts on scalability, see also (Jamieson, Khokhar, and Patel 1995), the latency metric will consider different implementations of the algorithm. In this case it will define the scalability of a parallel systems consisting of a parallel algorithm implementation and a parallel machine. The scalability metric based on latency is defined below

$$scale(e,(P,P')) = \frac{L_e(W,P)}{L_e(W',P')} \tag{2.25}$$

where L_e is the average latency when the efficiency is kept at a fixed level. In general since more overhead is expected in scaled system the value $scale(e,(P,P')) \leq 1$. This method may be applied in environments were the latency measurement is available. This approach was followed to a physics simulation program on a KSR-1 machine (Zhang, Yan, and Ma 1994). The average latency was measured with the help of special hardware monitor.

A fairly different approach used a simulation of a parallel shared memory machine (Sivasubramaniam et al. 1994). The execution-driven simulator SPASM was used in studying scalability by quantifying the *overhead functions*. The overhead was classified into *algorithmic overhead* due to the nature of the algorithm and *interaction overhead* caused by the interaction between the architecture and the algorithm. The algorithmic overhead was quantified by computing the time taken to execute the parallel program on an ideal machine, as PRAM, and measuring the deviation from the linear speedup curve. Furthermore, interaction overhead was separated into *latency overhead* and *contention overhead*. Latency overhead is due to waiting for a message to be available assuming the message did not contend on any link. The contention overhead is the time taken by a processor waiting for a link to be available. The use of a simulator provides more flexibility in choosing the hardware system configuration, but the results depend greatly on the simulation parameters.

2.4 Monitoring Tools for Performance Measurement

Monitoring tools in the parallel environment face many challenges. A monitoring tool usually is provided by hardware manufacturer to aid in the development and tuning of parallel code. Portable tools are difficult to implement since they can not be isolated from the specifications of the hardware. A portable toolkit named AIMS (Automated Instrumentation and Monitoring System) that uses both simulation and measurement to predict performance is an example of such a tool. Portability of AIMS is kept by using a modular design of the tool components. The basic idea behind this tools is to

collect a stream of events into a trace file with consistent format across architucters. Since the measurement at run time may perturb the execution, see (Malony, Reed, and Wijshoff 1992), such intrusions are removed by an intrusion-compensation module depending on the underlying architecture (Yan, Sarukkai, and Mehra 1995).

For distributed systems a tool named ZM4/SIMPLE, used a combination of software and hardware monitoring (Hofmann et al. 1994). The hardware monitor ZM4 (abbreviation for German "Zahlmonitor 4") is structured as master/slave system with a CEC (Control and Evaluation Computer) that is a master, and number of MAs (Monitor Agents) as slaves. The MA is a PC with special hardware components to detect and record events on the network. The software part SIMPLE is implemented to analyze event traces collected by MAs.

2.5 Analysis of Scalability by Estimating Code's Sensitivity

The approach we followed in this work is inspired by the work of G. Lyon et al 1995, and based on experimental design techniques to study the sensitivity of the code to changes in its performance with respect to changes in the number of processors or the problem size. This approach unlike other approaches neither needs special hardware measurement nor it needs a deep knowledge of the code. Yet, it provides a model that indicates how the system performance will change with the scaling of system parameters. Moreover, it will point out the segments of code that have high effect on the performance. The flexibility of this approach makes it suitable for wide range of applications. This technique was applied to some sample codes as in (Snelick et al.

1993), (Lyon, Snelick, and Kacker 1994) and (Lyon, Kacker, Linz 1995). The following chapter will give a background on experimental design techniques we used in this work and how it was used to estimate scalability. Our initial experiment on a cluster of work-stations is also presented.

CHAPTER 3

EXPERIMENTAL DESIGN

Experiments are essential for scientific and engineering development. Scientists used experiments to study an observed phenomenon and separate the effects from other circumstances. Engineers used experiments to assist in the development of manufacturing process. Products need to be tested and modified before reaching the market. A well designed experiment should derive the required information at the least expenditure of resources. This chapter introduces factorial designs and how it can be used to obtain the required information with minimum cost. The *SPT* (Synthetic Perturbation Tuning) approach is introduced with a small illustrative example on how to utilize factorial designs to analyze computer codes. The initial experiment of the ARPS code on a cluster of work stations is presented with the analysis of the results.

3.1 Designing an Efficient Experiment

Experimental investigation could be a long and expensive process. Experimental design aims to obtain the required information with high accuracy and minimum cost. An investigator with no planed experiments could end up with inaccurate information and waste of resources. Three key elements are essential for any successful experiment: knowledge of the process, measured response variables, and clear goals and objectives.

A sufficient knowledge of the process is necessary, and this knowledge could be built by a sequence of preliminary experiments before going to the full experiment rather than studying the process deeply. These sequence of experiments aim to point out the significant factors that have high influence on the process. Before starting any experiment the response variable must be specified. The measurement of the response need to be quantitative, qualitative response variables are difficult to compute, and need to be transformed into quantitative formats before any computations can take place. For parallel code the total run time (wall-clock time) is usually the response to be measured (Crowl 1994) and (Lyon, Kacker, and Linz 1995). The goal of an experiment applied to parallel system could be studying the effect of changes in some of the systems parameters. If the parameters include the number of processors and the problem size then it may be used to study the scalability of the parallel system. The objective, in this case, would be to find the bottlenecks in the code that are not scalable and enhance them. Any experimenter should keep the above three elements in mind before starting experimentation. In the following section we will introduce the basic aspects of factorial desings.

3.2 Factorial Designs

Unlike one factor at a time experiments, factorial designs are powerful technique to study the effect of more than one factor at a time. As we will see, the number of treatments grow exponentially with the number of factors under investigation, hence fractional factorial designs may be used to reduce the number of treatments. We will

state some definitions before introducing factorial experiments and how they can be used to study parallel systems.

3.2.1 Background and Definitions

Any designed experiment will consist of several elements. The following definitions are necessary to introduce the factorial experiments. A *factorial experiment* is an experiment in which we wish to study simultaneously the effects of several factors. Each factorial experiment will have one or more factors to which meaningful changes can be made to observe their effect on the response variable. We will consider discrete factors were each can have l discrete levels. We may denote these levels as 0, 1, 2, ...,l-1. Also a factorial experiment will have a measured response.

Let R denote the measured response, to study the effect of factor F on the response R we may compare the values of R at different levels of F. In most cases the response R is subject to other *nuisance factors* not included in the study. If a nuisance factor can be easily identified so the experiments can be classified accordingly, then effect of this nuisance factor can be eliminated by using block designs. When the nuisance factors are small and the experiments can not be classified under these factors then an unknown noise will be noticed in the response. This noise will appear as variance of R. Let the actual response without noise denoted as θ. Then we may express the relation between the measured response R and the actual response as follows,

(3.1) $$R = \theta + \varepsilon .$$

Where ε is the noise resulting from non-controllable factors .

Definition 3.1: For an experiment with m factors, each factor F_i (i=1,2, ...,m.) has l_i levels. Let f_i denote the current selected level for factor F_i then we define a *treatment* $t = (f_1, f_2, \ldots, f_m)$ as a combination of level settings of factors $F_1, F_2, \ldots, F_m$.

It is clear that the total number of treatments will be $l_1 \times l_2 \times \ldots \times l_m$. Now we may rewrite equation (3.1) as follows

(3.2) $$R(t) = \theta(t) + \varepsilon(t).$$

We assume the noise is a random variable with mean zero and variance σ^2, that is, $E[\varepsilon(t)] = 0$ and $V[\varepsilon(t)] = \sigma^2$, where σ^2 is unknown but could be estimated. Now we define one way to compare two levels of a factor F. We will denote the actual theoretical yield at treatment t_i as $\theta(t_i)$, or for short θ_i.

Definition 3.2: For a factor F at l levels then θ_i is the actual yield at treatment $t_i = (f_i)$. We compare two treatments t_i and t_j by estimating the value of $\{\theta(t_i) - \theta(t_j)\}$ or $(\theta_i - \theta_j)$ that is called the *contrast* between the two treatments of F.

Also we may define the contrast between all the levels of a given factor as follows,

Definition 3.3: Let $\theta^t = (\theta_0, \theta_1, \ldots, \theta_{l-1})$ be the actual responses for a factor F with l treatments, with each level corresponds to one treatment and $a^t = (a_0, a_1, \ldots, a_{l-1})$ the treatment vector of F such that $\sum_{i=0}^{l-1} a_i = 0$.

Then we say that $a^t\theta = (a_0\theta_0 + a_1\theta_1 + \cdots + a_{l-1}\theta_{l-1})$ is a *contrast* between the θ_i at all the treatments of the factor F.

Before describing a factorial experiment using the above definitions we need to introduce Hadamard matrices. This type of matrices is related to factorial designs at two levels as we will see shortly.

Definition 3.4: Let $a^t = (a_0, a_1, \ldots, a_{l-1})$ and $b^t = (b_0, b_1, \ldots, b_{l-1})$ be two vectors, then a and b are said to be *orthogonal* if the inner product of the two vectors is equal zero, that is $a^t b = (a_0 b_0 + a_1 b_1 + \cdots + a_{l-1} b_{l-1}) = 0$.

Definition 3.5: A squared matrix H of order n whose entries are +1 or -1 is called a *Hadamard matrix* of order n if each two rows are orthogonal. That is $HH^t = nI$, where I is the identity matrix of order n.

We write a Hadamard matrix of order $n=2^m$ as H_m. We will use "+" and "-" to denote the matrix element as abbreviation of "+1" and "-1". The following are some examples of Hadamard matrices:

$$H_1 = \begin{bmatrix} + & + \\ + & - \end{bmatrix} \qquad H_2 = \begin{bmatrix} + & + & + & + \\ + & - & + & - \\ + & + & - & - \\ + & - & - & + \end{bmatrix}$$

$$H_3 = \left[\begin{array}{cccc:cccc} + & + & + & + & + & + & + & + \\ + & - & + & - & + & - & + & - \\ + & + & - & - & + & + & - & - \\ + & - & - & + & + & - & - & + \\ \hdashline + & + & + & + & - & - & - & - \\ + & - & + & - & - & + & - & + \\ + & + & - & - & - & - & + & + \\ + & - & - & + & - & + & + & - \end{array}\right]$$

Definition 3.6: Let A be a matrix of size $m \times n$, and B a matrix of size $p \times q$ the *direct product* of the two matrices A and B written as $(A \otimes B)$ is defined by:

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \cdots & a_{1n}B \\ \cdot & \cdot & \cdots & \cdot \\ a_{m1}B & a_{m2}B & \cdots & a_{mn}B \end{bmatrix}. \tag{3.3}$$

We can obtain a Hadamard matrix H_m by taking the direct product of H_1 m times. For example, $H_2 = H_1 \otimes H_1$, also $H_3 = H_1 \otimes H_1 \otimes H_1$. We may find other methods to build Hadamard matrices in (Hedayat and Wallis 1978) and (Drouin 1993).

Now using the above definitions we will describe a small factorial design and how useful information can be obtained by simple calculations. Consider the case where each factor is at two levels. The two levels are denoted as "-1" and "+1" or for short we may write "-" and "+". Let F_1 and F_2 be two factors with l_1 and l_2 levels, where $l_1 = l_2 = 2$. Let $a_0^t = (+,+,+,+)$, $a_1^t = (+,-,+,-)$, $a_2^t = (+,+,-,-)$, and $a_{12}^t = (a_1 \bullet a_2)^t = (+,-,-,+)$, where $a_1 \bullet a_2$ is the element by element product of the

two vectors (also called the direct product). Let $\theta^t = (\theta_1, \theta_2, \theta_3, \theta_4)$, where θ_i is the actual response at treatment t_i. Then the following contrasts are defined,

$$\beta_0 = a_0^t \theta,$$

$$\beta_1 = a_1^t \theta,$$

$$\beta_2 = a_2^t \theta,$$

and $$\beta_{12} = a_{12}^t \theta.$$

Ignoring a constant multiplier, we will use β_0 to find the global mean which is denoted also as μ. We will use the contrasts β_1 and β_2 to estimate the main effects of factors F_1 and F_2 respectively. The last contrast β_{12} will be used to estimate the interaction effect of the two factors F_1 and F_2. We may write this design in a matrix format using Hadamard matrix of order 4 as follows:

(3.4)
$$\begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_{12} \end{bmatrix} = \begin{bmatrix} + & + & + & + \\ + & - & + & - \\ + & + & - & - \\ + & - & - & + \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \end{bmatrix}.$$

3.2.2 Factorial Designs at Two Levels

To perform a general factorial design an experimenter selects fixed number of levels for each factor, and then run the experiments with all possible combinations. As we have seen before, for an experiment with m factors, where factor F_i has l_i levels the total number of combinations is $l_1 \times l_2 \times \cdots \times l_m$. When each factor has exactly two levels it is called *two level factorial design*. A two level factorial design with m factors

has 2^m treatments. The two levels of each factor will be denoted with "-" for the low setting and "+" for the high setting. The contrasts we have defined before are used to estimate the main effects and interaction effect as we will see later.

Consider a design with three factors, F_1, F_2, and F_3 at two levels each. The total number of treatments required for a full factorial design will be $2^3 = 8$ treatments. The design shown in Table 3-1 contains the columns corresponding to the treatment vectors of the main factors. The signs in each column are said to be in the *standard order*. At each row representing a treatment t_i a response θ_i corresponds to that treatment. A design is said to be in standard order if the first column consists of successive minus signs and plus signs, the second column consists of successive pairs of minus signs and plus signs, the third consists of four minus signs followed by four plus signs, and so on. The following is a general definition of the standard order

Definition 3.7: A two level full factorial design with m factors is said to be in *standard order* if column i consists of 2^{i-1} minus signs followed by 2^{i-1} plus signs until the column is full.

Treatment	F_1	F_2	F_3	θ
t_1	-	-	-	θ_1
t_2	+	-	-	θ_2
t_3	-	+	-	θ_3
t_4	+	+	-	θ_4
t_5	-	-	+	θ_5
t_6	+	-	+	θ_6
t_7	-	+	+	θ_7
t_8	+	+	+	θ_8

Table 3-1: Full factorial design with three factors in standard order.

Consider the two treatments t_1 and t_2 in Table 3-1,

t_1 - - - θ_1

t_2 + - - θ_2

it is a one factor at a time experiment where we change the level of F_1 with contrast $\theta_2 - \theta_1$. Also the treatments t_3 and t_4, t_5 and t_6, and t_7 and t_8 will have the contrasts $\theta_4 - \theta_3$, $\theta_6 - \theta_5$, and $\theta_8 - \theta_7$. These are four estimates of the effect of F_1. The average of all the estimates will be

$$\beta_1 = \frac{(\theta_2 - \theta_1) + (\theta_4 - \theta_3) + (\theta_6 - \theta_5) + (\theta_8 - \theta_7)}{4} \tag{3.5}$$

or we may write

$$\beta_1 = \frac{(\theta_2 + \theta_4 + \theta_6 + \theta_8)}{4} - \frac{(\theta_1 + \theta_3 + \theta_5 + \theta_7)}{4} \tag{3.6}$$

which is the average of the four responses when F_1 was at the high level "+" minus the average of the four runs when the F_1 was at the low level "-". The numerator in (3.5) is the contrast between the θ_i, while the denominator is half the number of runs. We may now define the main effect, interaction effect, and the mean as follows,

Definition 3.8: For a factorial design the *main effect* of a factor F is the contrast between the responses of all treatments divided by half the number of runs. We write

$$\beta = \frac{a^t \theta}{n/2}. \tag{3.7}$$

Where n is the number of treatments, and $a^t\theta$ is the contrast obtained from the inner product of a^t, the transpose of the treatment vector of factor F, and θ the response vector.

Definition 3.9: For a factorial design with m factors and n treatments, let a and b be the treatment vectors for factors F_i and F_j respectively. Then the *interaction vector* of the two factors is a new vector c obtained from the direct product of $a \bullet b$. That is $c = (a_1b_1, a_2b_2, \ldots, a_nb_n)$.

The above definition could be generalized for more than two factors. For example, the interaction vector for the three factors F_i, F_j, and F_k with treatment vectors a, b, and c will be the direct product $a \bullet b \bullet c$. Now we define the interaction effect as follows

Definition 3.10: For a factorial design the *interaction effect* of m factors F_1, F_2, ..., F_m is the contrast of the interaction of theses factors divided by half the number of treatments. This interaction is said to be of order m. The contrast of the interaction is obtained from the dot product of the

interaction vector a by the response vector θ. We write the interaction effect as

$$\beta_{12\ldots m} = \frac{a^t \theta}{n/2}. \tag{3.8}$$

The mean, however, is defined as

Definition 3.11: For a factorial experiment with m factors and n treatments the *global mean* (or the mean) is defined as the sum of all responses divided by the number of treatments. That is

$$\mu = \frac{\sum_{i=1}^{n} \theta_i}{n}. \tag{3.9}$$

The main and interaction effects will be used as indicators to the important factors. One way to judge the importance of a factor is to pickup the factors with the highest effects. After picking these significant factors the magnitude and sign of the effects are considered in the analysis. However, this method may lead to wrong conclusions. When the uncontrolled noise is high the values of the effects could be due to the noise effect and not a real effect of the factors. A better way to judge theses effects is to compare them against the *standard error interval*. Only the effects and interactions found to be out of this interval will be considered significant.

3.2.3 Estimating the Standard Error

The experimenter could use the high order interaction effects as estimates of the standard error under the assumption that the high order interactions are negligible. This may happen when the experimenter had a high confidence that the high order

interactions does not exist in the process under study, and the values of these effects are only due to the noise effects. Nonetheless, this assumption does not always hold, and when the replication of the experiments is possible we could have a better estimate of the standard error. If the experimenter could replicate each treatment r times, then the total number of runs would be $r \times n$. The variation between the replicates at each treatment is used to estimate the standard deviation for that treatment. Then we estimate the standard error for the experiment based on the standard deviations of the treatments.

If we consider each replicated treatment as a set of experiments with the same conditions then let n be the number of treatments, and r_i is the number of replicates for treatment t_i, where $i = 1, 2, \ldots, n$. Let $v_i = r_i - 1$ denote the degrees of freedom of the ith set. The variance s_i^2 is an estimate of σ^2 with v_i degrees of freedom. The pooled estimate of all the runs variance is

$$s^2 = \frac{v_1 s_1^2 + v_2 s_2^2 + \cdots + v_n s_n^2}{v_1 + v_2 + \cdots + v_n} \qquad (3.10)$$

When $v = v_1 = v_2 = \cdots = v_n$, then

$$s^2 = \frac{v \sum_{i=1}^{n} s_i^2}{nv}$$

$$= \frac{\sum_{i=1}^{n} s_i^2}{n} \qquad (3.11)$$

In equation (3.6) the main effect was represented as the difference between the average of low runs and the high runs. We will replace the actual response θ_i by the measured response R_i, then we rewrite (3.6) as follows

(3.12) $$\beta_1 = \frac{(R_2 + R_4 + R_6 + R_8)}{4} - \frac{(R_1 + R_3 + R_5 + R_7)}{4}.$$

If we denote the average of responses at low level as $\bar{R}_-$ and the average of responses at high level as $\bar{R}_+$, then we rewrite (3.12) as follows

(3.13) $$\beta_1 = \bar{R}_+ - \bar{R}_-.$$

In general each main effect or interaction is a statistic of the form

(3.14) $$\beta = \bar{R}_+ - \bar{R}_-$$

where each average contain $\frac{r \times n}{2}$ responses, assuming independent error the variance is given by

$$V(\beta) = V(\bar{R}_+ - \bar{R}_-)$$

$$= \left(\frac{2}{r \times n} + \frac{2}{r \times n}\right)\sigma^2$$

(3.15) $$= \frac{4}{r \times n}\sigma^2$$

where σ^2 is substituted by the estimate s^2. So we write

(3.16) $$V(\beta) = \frac{4s^2}{r \times n}$$

and the variance for the mean

(3.17) $$V(\mu) = \frac{s^2}{r \times n}.$$

The standard error $SE = \sqrt{V(\beta)}$ and the mean standard error $\sqrt{V(\mu)}$. As we mentioned earlier, when comparing the main effects and interactions effects with the standard error interval, denoted SE, we use $2SE$ corresponding to 95.45% under the normal distribution curve or $3SE$ corresponding to 99.73%. According to the central limit theorem the error noise is very close to the normal distribution when the runs are randomized and the number of runs is reasonably large.

3.2.4 Fractional Factorial Designs

Since the number of runs of a two level factorial design increases geometrically as m the number of factors increases, it is necessary to find a reduced set of treatments. Half-fractional factorial design are used to obtain the required information about the process with m factors in 2^{m-1} treatments. The cost, however, is to lose some of the high order interactions assuming they are negligible.

An example of half-fractional factorial design with m=3 factors is constructed using H_2 Hadamard matrix. Let F_1, F_2, and F_3 be three factors, then we rewrite (3.4) as follows

(3.18) $$\begin{bmatrix} \mu \\ \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix} = \begin{bmatrix} + & + & + & + \\ + & - & + & - \\ + & + & - & - \\ + & - & - & + \end{bmatrix} \begin{bmatrix} R_1 \\ R_2 \\ R_3 \\ R_4 \end{bmatrix}$$

where the treatment vector for F_3 is equivalent to the interaction vector of the two factors F_1 and F_2 . The number of treatments is 2^{3-1} which is half of the number of treatments in 2^3 full factorial design. The design is constructed by assuming the interaction effect β_{12} negligible so β_3 could be aliased with it.

Treatment	F_1	F_2	F_3	R
t_1	+	+	+	R_1
t_2	-	+	-	R_2
t_3	+	-	-	R_3
t_4	-	-	+	R_4

Table 3-2: An example of 2^{3-1} fractional factorial design.

The orthogonality of Hadamard matrices and the similarity between the entries of Hadamard matrix and two level factorial designs is the motivation to use Hadamard matrices to construct full and fractional factorial designs. The fractional factorials are usually described by the number of factors and the resolution of the design. For an integer k, a fractional factorial design is said to be of resolution $2k+1$ if it satisfies the condition that all effects of order k or less are estimable whenever all effects of order higher than k are assumed to be zero. A fractional factorial design is said to be of resolution $2k$ if all effects of order k-1 or less are estimable whenever all effects of order $k+1$ or higher are assumed to be zero. It have been proved that the existence of a Hadamard matrices of size $4k$ implies the existence of orthogonal fractional design of

resolution III for $4k$-1 factors each at two levels (Hedayat and Wallis 1978). The design in Table 3-2 is of resolution III, denoted 2^{3-1}_{III}, the subscript denotes the resolution of the design. The designs of resolution III, IV, and V are of our interest since they are very common. For designs of resolution III no main effect is aliased with any other main effect, but they are aliased with two factor interactions and some two factor interactions may be aliased with each other. In the design of resolution IV no main effects are aliased with each other or with two factor interactions, but two factor interactions are aliased with each other. Designs of resolution V will have no main effect or two factor interaction aliased with any other main effect or interaction, but two factor interactions are aliased with three factor interactions.

3.2.5 Building a Linear Model to Describe the Process

The computed effects can be utilized to build a linear model for predicting the performance with change of the factors. Let $F_1, F_2, \ldots, F_m$ be the factors involved in the study with estimated effects $\beta_1, \beta_2, \ldots, \beta_m$; and interaction effects $\beta_{12}, \beta_{13}, \ldots, \beta_{12\ldots m}$. Let $\theta(t)$ be the actual response at treatment $t = (f_1, f_2, \ldots, f_m)$, μ be the estimated global mean, and SE the estimated standard error, then

$$\theta(t) = \mu + \beta_1 F_1 + \beta_2 F_2 + \cdots + \beta_m F_m + \beta_{12} F_{12} + \cdots + \beta_{12\ldots m} F_{12\ldots m} + SE . \tag{3.19}$$

This similar to Taylor's expansion when the second and higher order derivatives in the Taylor's series are assumed to be zero.

3.3 Design of Experiments to Analyze Parallel Codes

The full and fractional factorial designs at two levels were introduced in the previous sections, this section explains how these factorial designs can be used to tune parallel code. The goal of the experiment is find out how to increase the performance of the parallel code when the number of processors increased. However, increasing the performance can be done if the bottlenecks of the code are identified, and then minimized or eliminated. The question is what factors could be used when analyzing parallel codes.

3.3.1 The Use of Synthetic Perturbations

The technique used by Snelick et al. 1993, is a promising approach to identify the sources of the poor performance in the parallel code. The *SPT* (Synthetic Perturbation Tuning) technique introduces the notion of inserting artificial delays into the source code and apply experimental design techniques to capture the effects of these delays on the performance (Lyon, Snelick, and Kacker 1994) and (Lyon, Kacker, and Linz 1995). The logic is that if a delay decrease the performance then improvements of the code will result in improved performance. The main problem here is choosing these segments of the code where delays should be inserted. Screening is used to identify the segments of code that have a significant effect on the performance. The knowledge about the structure of the code will help, also, in choosing the suspected segments of code. For large code the investigator may need to do some preliminary experiments

(screening), followed by experiments that focus on the factors that appeared to have significant effect on the performance.

The synthetic perturbations and factorial designs will be used together to provide an economical way for tuning the code. We need a set of factors that affect the performance, also we need a measured response. Most of parallel codes lack adjustable parameters to be used as experiment factors, synthetic perturbation is a good economical alternative to simulate local adjustable efficiency changes. The inserted delay can be easily inserted and removed without changing the original code. Each synthetic perturbation is an extra code that causes delay and should have no effect on the computations. The inserted delay should be large enough to be distinguished from the background noise, yet the delay should not be significantly large to slow the system. The amount of delay needed could be determined after some initial small experiments.

Since scalability is related to the measured performance when the size of the machine or the problem size are scaled, the experiment should be designed with the machine size and problem size as a factors along with the other factors representing segments of the code. The factorial design will be used to isolate important factors, and to map a relation between the input factors and the output response using a linear model.

3.3.2 Example of Factorial Experiment with Parallel Code

Given a parallel program we want to test its scalability. First, we find which segments of the program are potential bottlenecks. At each segment a delay is inserted which can be switched on or off. Each artificial delay will be associated with a factor. The factor may be set at one of two levels, low level denoted "-" when no delay is applied and high level denoted "+" when delay is applied. The number of processors used to run the parallel program and the problem size are factors as well. For the factor associated with number of processors, the sitting of this factor to low "-" will correspond to P processors, and setting to high "+" corresponds to P' processors, where $P < P'$. If the scale factor is associated with the problem size the low "-" setting will represent small problem size N and high "+" setting will represent the scaled problem size N', where $N < N'$. After running the experiment the investigator selects the significant factors, and perform more experiments to find which parts of the code affects scalability.

Let F_1, F_2, and F_3 be three factors where the factors F_1 and F_2 corresponds to two different segments of the code with inserted delay, and the factor F_3 corresponds to the number of processors. Table 3-3 shows the 2^3 full factorial design, with $r = 2$, for two replicates.

F_1	F_2	F_3	$R1$	$R2$	Total	Average R
-	-	-	19	20	39	19.5
-	-	+	10	8	18	9
-	+	-	17	21	38	19
-	+	+	9	8	17	8.5
+	-	-	18	20	38	19
+	-	+	9	9	18	9
+	+	-	19	18	37	18.5
+	+	+	9	9	18	9

Table 3-3: Responses of an experiment applied to a parallel program with three factors.

The settings of factors F_1 and F_2 to "-" represents no delay is applied, and the sitting to "+" represents applied delay at that segment of code. The settings of the third factor F_3 at low "-" when the number of processors $P = 4$, and at high "+" when $P' = 8$ processors were used. Each row of the Table 3-3, corresponds to one treatment of the experiment. The first row means to run the program without delay and using 4 processors, while the second row means to run the program without delay using 8 processors and so on. The order in which the experiments were performed is not the same as it appears in the table. The experiments were run in random order to ensure the normality of noise. The response to be measured is the total elapsed (wall clock) time. The average response of each treatment is in the last column of the table. We now estimate the effects of the factors following the definitions given earlier and using the average response at each treatment instead of the actual response.

$$\beta_1 = \frac{19+9+18.5+9-19.5-9-19-8.5}{4}$$
$$= -0.125$$

$$\beta_2 = \frac{19+8.5+18.5+9-19.5-9-19-9}{4}$$
$$= -0.375$$

$$\beta_3 = \frac{9+8.5+9+9-19.5-19-19-18.5}{4}$$
$$= -10.125$$

The interaction effects were estimated in the same way

$$\beta_{12} = \frac{19.5+9+18.5+9-19-8.5-19-9}{4}$$
$$= 0.125$$

$$\beta_{13} = \frac{19.5+19+9+9-9-8.5-19-18.5}{4}$$
$$= 0.375$$

$$\beta_{23} = \frac{19.5+8.5+19+9-9-19-9-18.5}{4}$$
$$= 0.125$$

$$\beta_{123} = \frac{9+19+19+9-19.5-8.5-9-18.5}{4}$$
$$= 0.125$$

The estimated mean will be

$$\mu = \frac{19.5+9+19+8.5+19+9+18.5+9}{8}$$
$$= 13.938$$

The above effects and interactions are compared against the standard error. If the value is out of the standard error interval $2SE$, then the factor is considered significant,

otherwise it could be a result of the background noise. The standard error was estimated by estimating

$$V(\beta) = \frac{4(1.688)}{2 \times 8}$$
$$= 0.422$$

then the estimated standard error

$$SE = \sqrt{0.422}$$
$$= \pm 0.65$$

Similarly the estimates for the mean

$$V(\mu) = \frac{(1.688)}{2 \times 8}$$
$$= 0.106$$

then the estimated standard error is $\sqrt{0.106} = \pm 0.326$.

The above estimates does not show any significant interaction out of the error interval $2SE$. The high order interactions (as the interaction β_{123} in this example) could be used as an estimate of the standard error if there is no replications. It is important to check if there is an interaction before interpreting the main effects. The main effects of factors F_1 and F_2 are within the error interval. The effect of F_3, the scale factor, is large and out of the error interval. The negative sign indicates that scaling the number of processors will decrease the response that is the run time in this case. The linear model corresponding to the above process

$$\theta(t) = 13.94 - 0.13F_1 - 0.38F_2 - 10.13F_3 + 01.3F_{12} + 0.38F_{13} + 0.13F_{23} + 0.13F_{123} \pm 0.65.$$

3.4 Initial Test of ARPS Code

This section explains how initial experiments were applied to ARPS and the preliminary results of the initial study on a cluster of work stations. We start with some information about ARPS. More detailed information can be found in (Droegemeier et al. 1992), (Xue et al. 1995) and (Sathye et al. 1995).

3.4.1 The Advanced Regional Prediction System (ARPS)

The Advanced Regional Prediction System (ARPS) was developed by group of researchers at the Center for Analysis and Prediction of Storms (CAPS), University of Oklahoma. ARPS model is one of six major areas under CAPS program. The development of the ARPS model started in July, 1990 with ARPS version 1.0, but the first formal release was version 3.0 in September, 1992.

The ARPS model is a three-dimensional, non hydrostatic code designed for the prediction of small scale, short duration events like thunderstorms, snow bands, and downslope windstorms. The location of the events range from 1 to 50 kilometers, and timing of events ranging from 5 minutes to 1 hour. The developed model included governing equations for momentum, heat, mass, water substances, turbulent kinetic energy, and the equation of state. The model was developed at CAPS with three main goals: sufficient adaptability to new data assimilation strategies, ease of use, and suitability for variety of computing platforms. Moreover, the model was designed to be suitable for scalable parallel processors, which makes it a good target for the scalability study. Because the code is huge (more than 280 subroutines distributed in

30 files) and was developed by more than 30 scientific and support personnel over the last six years, the approaches mentioned earlier (see Chapter 2) are difficult to apply if not impossible. The *SPT* approach is the best economical way to test scalability of this code.

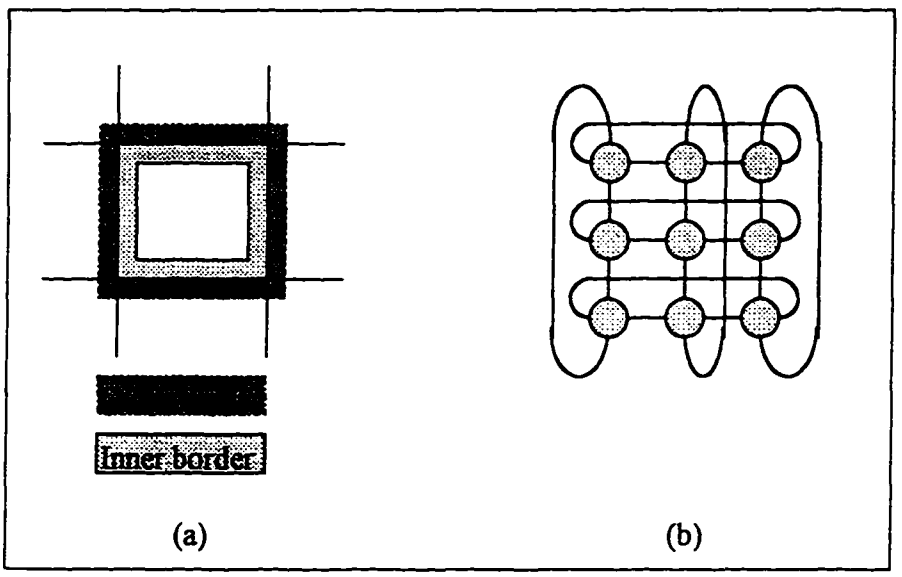


Figure 3-1: (a) The *x*-*y* decomposition with light shade representing the inner border grid point to be send to neighboring processors, and the darker shade represents the grid points to be received from the neighboring processors. (b) An example of 3×3 mesh of processors with wrap-around connections.

The parallel version of ARPS uses two dimensional domain decomposition where the grid space was partitioned along the *x* and *y* axis, as in Figure 3-1. The shaded region is the data shared with other neighboring processors. The shared region need to be exchanged after each time step (Johnson et al. 1994) and (Sathy et al. 1995).

3.4.2 The Initial Experiment

The parallel version of ARPS was run on a cluster of HP9000 model 715/64 workstations connected with 10 mb/sec Ethernet network. The processors are assumed to be arranged in a mesh with wraparound connections as in Figure 3-1(b). Each processor is assigned a subset of the grid points of the whole domain, after each time step the boundary values are exchanged with the neighboring processors. For this experiment we used the PVM (Parallel Virtual Machine) version of ARPS.

Five factors where selected for this initial study plus the scale factor. The scale factor, denoted F_0, was set to "-" for 2×2 mesh, and set to "+" for 3×3 mesh. The other factors F_1, F_2, F_3, F_4, and F_5 represent different segments of the ARPS code which are believed to be representative of other similar segments. The factors are set to low level "-" when no delay was applied to that segment of code and to high level "+" when delay was applied. The experiment was run in random order with $r = 5$ replicates and $m = 6$ factors with total of $5 \times 2^6 = 320$ runs.

Since the ARPS model has many adjustable parameters to be set before running the code, these parameters were fixed during this study to the default values, except for the model run time and the grid size which are set to values within the time and memory space limits. Before selecting the factors the calling tree of the system was obtained to aid in selecting the code segments to work as factors for this experiment (see Appendix A). The communication patterns of the subroutines exchanging the boundary conditions were considered also in selecting the factors. All

the selected factors except F_5 represent subroutines that were executed (called) during the current runs of the system. The subroutine where F_5 was located was not called under the current settings of ARPS parameters. We expect that the effect of this factor will be insignificant. For these set of experiments we set the model run time to 120 seconds.

3.4.3 The Results of the Initial Experiment

Table 3-4, shows the main effects when running on 2×2 and 3×3 meshes along with the main effects of the overall runs. The last column show the interaction between the scale factor and other factors, this column should be examined first before the main effects can be examined. It is clear that all interactions are within the error interval.

Factor	Name	Main Effect at 2×2 mesh size	Main effect at 3×3 mesh size	Overall Main Effect	Interaction with Scale
F_1	ADVU	3.35	3.91	3.63	0.28
F_2	ADVCTS	6.77	6.06	6.42	-0.35
F_3	BCSU	68.92	67.24	68.08	-0.84
F_4	BOUNDU	9.9	10.46	10.18	0.28
F_5	SLOVTKE	-3.5	0.89	-1.33	2.22

Table 3-4: Main and interaction effects with scale factor for the initial experiment on a cluster of work stations with $\mu = 113.71$ and $SE = \pm 1.34$.

The mean $\mu = 113.71$ and the standard error estimated to be $SE = \pm 1.34$. The main effect of F_0 was β_0=-12.46 is out of the error interval $2SE$. However, the

average decrease gained in run time from using five more processors is no more than 13 seconds indicating no point of using larger size mesh without modifying the code to be more scalable or enhance the underlying architecture, the change in response is within the experimental noise margin.

The same may be said about the effects of remaining factors except F_3 which have high effect β_3=68.08 which is out of the standard error interval and much larger than other effects. The high effect of F_3 (subroutine BCSU) suggest that any enhancement for this subroutine may decrease the overall response time. The next factor was F_4 with effect β_4=10.18. The factor F_5 corresponding to subroutine "SOLVTKE" is within the $2SE$ error interval as expected.

3.5 Summary

In this chapter the experimental design technique used in this research is explained. The factorial designs are used to study the effect of more than one factor at a time. The fractional factorial designs could be used when the number of factors involved in the study grows and hence the number of treatments grows exponentially.

The Hadamard matrices were used to generate factorial designs because of the similarity in the entries and their orthogonal design. We presented a small example on how to estimate the effects and to map a relation between these effects an the response via a linear response model.

Also, we introduced ARPS and our initial experiment on a cluster of work stations. The next two chapters will present the results of our experiments on a shared memory machine CRAY J90 and a distributed memory machine the IBM SP2.

CHAPTER 4

ANALYSIS OF ARPS CODE ON CRAY J90

At the end of Chapter 3 an initial experiment on a cluster of work stations was introduced. The experiments in this chapter were conducted on the shared memory machine the CRAY J90. We classified the experiments in this chapter into two major parts. In the first part we scaled the problem size, that is covered in Section 4.3, and in the second part we scaled the machine size which is covered in Section 4.4. We further analyzed the interaction in section 4.5, and examined the effect of using large delay in section 4.6. In section 4.7 we linked the speedup to scalability using interaction plots. Before going into the details of the empirical work and the analysis of the results, we will give a brief background about the environment under which we conducted the experiments, and the CRAY J90 used in this study.

4.1 The CRAY J90

The study was performed on the CRAY J90 with eight vector processors, and 2GB global shared memory divided into 64 interleaved memory banks. Under the current configuration of the system a parallel job is not guaranteed a fixed number of processors if other jobs are running in the system. To ensure that a parallel job will get the required number of processors until its completion, and to get accurate measurements of the running time we used a special exclusive queue to run our

parallel jobs that require more than one processor. This queue will allow only one job at a time, and all jobs on the other queues will be suspended while the exclusive queue is running.

To perform the experiments we used both the serial and parallel versions of ARPS. The serial version was compiled using CRAY FORTARN 77 compiling system "cf77" with the option "-Zv", while the parallel version was compiled with the "-Zp" compiler option. The first option will compile the code for maximum level of vectorization but with out auto multitasking that is known as *autotasking*. The second option will compile for both maximum level of vectorization and autotasking that will automatically enable the code to run on more than one processor. Details about the compiling system are in (Cray 1993).

4.2 Description of the Experiments

This study consisted of two major parts. We analyzed the effect of scaling the problem size while using one vector processor in the first part, and the effect of scaling the machine size by changing the number of processors in the second part. In each of the two parts, 13 factors were involved where F_0 is the scale factor as in Table 1. As it was mentioned earlier the measured response was the wall-clock running time. For all the experiments a fractional factorial design of resolution 4 was used with 32 treatments, each treatment was replicated three times. The fractional design is shown in Appendix B.

4.3 Scaling the Problem Size

In the first part the scale factor F_0 represented the scaling of the problem size (the grid size in this case) while running on one vector processor. This part involved two experiments. In the first experiment the scale factor F_0 was set at low level "-" for 67×67×35 grid size, and set at high level "+" for 256×256×35 grid size. The other factors represent selected segments of the ARPS code with synthetic perturbations. In the other experiment the problem size scale factor F_0 was set at low level "-" for 128×128×35 grid size, and set at high level "+" for 256×256×35 grid size.

The data in Table 4-1, summarizes the results of the first experiment. It can be observed from the table that the grid size scaling factor F_0 had a main effect β_0=2052.94, indicating a possible increase of about 34 minutes (2052 seconds) in run time when the grid size is scaled to 256×256×35. However, there were no significant interaction effect between the scale factor and other factors.

The factors F_8 and F_9 seem to have a high effect out of the error interval. These two factors, F_8 and F_9, represent the subroutines "BCSU" and "BCSV" which set the boundary conditions for the u-velocity and v-velocity components respectively. No other main effects seem to be real, since they are within the error margin.

Factor	Name	Main Effect	Interaction with Scale
F_0	Grid Size	2052.94	N/A
F_1	ADVCTS	9.845	-10.72
F_2	ADVU	2.98	-1.06
F_3	ADVV	13.79	9.17
F_4	ADVW	-2.2	-5.47
F_5	BCKMKH	5.03	-1.68
F_6	BCS2D	-1.83	-2.78
F_7	BCSCLR	13.18	-1.24
F_8	BCSU	175.27	-0.83
F_9	BCSV	178.83	3.65
F_{10}	BOUNDU	12.25	4.49
F_{11}	BOUNDV	-0.75	-7.71
F_{12}	JACOB	-8.06	-8.09

Table 4-1: Main and interaction effects for the CRAY experiment while scaling grid size from 67×67×35 to 256×256×35. The standard error SE=±12.5 and the mean μ=1430.71.

In the same way, the second experiment was performed where the grid size was scaled from 128×128×35 to 256×256×35. As can be seen in Table 4-2, the main effect of the grid size scaling factor was β_0=1673.28. This indicates less change of effect with small change in problem size. It is clear from the tables that no significant interaction effect between the scale factor and other factors. The effect of the factors will remain the same (almost unchanged) regardless of the problem size.

Factor	Name	Main Effect	Interaction with Scale
F_0	Grid Size	1673.28	N/A
F_1	ADVCTS	11.81	-12.69
F_2	ADVU	4.86	-2.94
F_3	ADVV	14.66	8.30
F_4	ADVW	-2.34	-5.33
F_5	BCKMKH	5.48	-2.12
F_6	BCS2D	-3.89	-0.72
F_7	BCSCLR	13.76	-1.82
F_8	BCSU	175.51	-1.07
F_9	BCSV	178.78	3.70
F_{10}	BOUNDU	12.98	3.76
F_{11}	BOUNDV	-0.84	-7.63
F_{12}	JACOB	-8.15	-7.99

Table 4-2: Main and interaction effects for the CRAY experiment while scaling grid size from 128×128×35 to 256×256×35. The standard error SE=±12.52 and the mean μ=1620.55.

Following the above results, three additional experiments with full factorial design were performed to investigate the effect of the two factors F_8 and F_9 when isolated from other factors. These experiments will estimate the main effects as well as all the interactions.

The first experiment scaled the grid size from 67×67×35 to 128×128×35. Each treatment in this experiment was replicated twice. The results of this experiment are shown in Table 4-3. $R1$ and $R2$ are the measured wall-clock run time in seconds for the two replicates and R is the average of these two. The estimated main effect of

grid size scaling factor β_0=391.042, the estimated effects of the other two factors were β_8=175.218 and β_9=176.022. The estimated standard error was SE=±1.97 and the mean was estimated to be μ=565.471. There were no significant interaction effects. The results for the other two experiments are shown in Tables 4-4 and 4-5. The summary of the three experiments is given in Table 4-6.

F_0 "Grid Size"	F_8 "BCSU"	F_9 "BCSV"	$R1$	$R2$	R
-	-	-	191.305	194.967	193.136
-	-	+	371.223	369.963	370.593
-	+	-	370.869	370.249	370.582
-	+	+	545.449	545.484	545.492
+	-	-	582.291	588.367	585.329
+	-	+	756.036	768.746	762.391
+	+	-	756.097	765.497	760.796
+	+	+	931.959	938.947	935.453

Table 4-3: The full factorial design and the measured responses on CRAY when grid size changed from 67×67×35 to 128×128×35.

We may observe that the effect of grid size is growing linearly with the scale factor. Also the experimental error is getting larger with large grid size as a result of longer running time that allows more noise to be encountered. The estimated effects for factors F_8 and F_9 were almost the same which confirms that the problem size has no interaction with these two factors. In other words no matter how the grid size changes, the main effect of F_8 and F_9 are high.

F_0 "Grid Size"	F_8 "BCSU"	F_9 "BCSV"	$R1$	$R2$	R
-	-	-	191.305	194.967	193.136
-	-	+	371.223	369.963	370.593
-	+	-	370.869	370.249	370.582
-	+	+	545.499	545.484	545.492
+	-	-	2327.574	2217.783	2272.679
+	-	+	2497.128	2398.826	2447.977
+	+	-	2480.403	2411.221	2445.812
+	+	+	2574.366	2692.634	2633.500

Table 4-4: The full factorial design and the measured responses on CRAY when grid size changed from 67×67×35 to 256×256×35.

F_0 "Grid Size"	F_8 "BCSU"	F_9 "BCSV"	$R1$	$R2$	R
-	-	-	191.305	194.967	193.136
-	-	+	371.223	369.963	370.593
-	+	-	370.869	370.249	370.582
-	+	+	545.449	545.484	545.492
+	-	-	2327.574	2217.783	2272.679
+	-	+	2497.128	2398.826	2447.977
+	+	-	2480.403	2411.221	2445.812
+	+	+	2574.366	2692.634	2633.500

Table 4-5: The full factorial design and the measured responses on CRAY when grid size changed from 128×128×35 to 256×256×35.

When we compare the mean μ=1409.971 in the second column of Table 4-6 with the mean in Table 4-1, μ=1430.71 we notice that they are very close. This emphasizes that

the two factors F_8 and F_9 are dominant. This is also true about the mean in the third column of Table 4-6 and that in Table 4-2.

Grid Size Change	from 67X67X35 to 128X128X35	from 67X67X35 to 256X256X35	from 128X128X35 to 256X256X35
Effect of F_0	391.042	2080.041	1689
Effect of F_8	175.218	177.750	176.796
Effect of F_9	176.022	178.838	178.676
Interaction effect β_{08}	-0.954	1.578	2.532
Interaction effect β_{09}	-0.162	2.655	2.817
Interaction effect β_{89}	-1.238	2.461	2.496
Interaction effect β_{089}	0.036	3.734	3.699
Standard Error (SE)	±1.97	±29.58	±29.60
Mean (μ)	565.471	1409.971	1605.492

Table 4-6: Summary of the results obtained from the three full factorial experiments when scaling the grid size.

4.4 Scaling the Number of Processors

The second part of this study required scaling of the machine size. Two sets of experiments were performed in this part. The first set was performed at 67×67×35 grid size. The number of processors in this set was scaled from one processor to two processors in one experiment, and from one to four processors in the other. The second set of experiments was performed at 128×128×35 grid size and the number of processors was scaled in the same way as the first set. We noticed that the standard

errors in the second part were smaller that allows other main and interaction effects to be out of the error interval as we will see soon.

4.4.1 Scaling the Number of Processors at 67x67x35 Grid Size

The results in Table 4-7, represent the main effects and interaction effects with scale factor when the machine size was scaled from one to two processors at grid size 67x67x35.

Factor	Name	Main Effect	Interaction with Scale
F_0	Machine Size	-97.01	N/A
F_1	ADVCTS	20.69	1.57
F_2	ADVU	4.49	-0.67
F_3	ADVV	4.65	-1.10
F_4	ADVW	3.41	0.12
F_5	BCKMKH	7.87	-0.14
F_6	BCS2D	0.80	-0.66
F_7	BCSCLR	14.39	0.95
F_8	BCSU	175.75	0.01
F_9	BCSV	175.40	0.13
F_{10}	BOUNDU	7.98	-0.95
F_{11}	BOUNDV	7.10	0.59
F_{12}	JACOB	0.01	0.21

Table 4-7: Main effects and interactions with scale for the CRAY experiment when scaling machine size from 1 to 2 processors with grid size 67x67x35. The standard error SE=±0.48 and the mean μ=354.8.

The scale factor effect was β_0=-97.01. This means on average 97 seconds decrease in wall-clock run time could be achieved by adding one more processor at this grid size. The factors F_8 and F_9 are the highest with estimated effects β_8=175.75 and β_9=175.40.

Factor	Name	Main Effect	Interaction with Scale
F_0	Machine Size	-138.87	N/A
F_1	ADVCTS	20.70	1.57
F_2	ADVU	4.45	-0.70
F_3	ADVV	4.62	-1.12
F_4	ADVW	3.48	0.20
F_5	BCKMKH	7.73	-0.28
F_6	BCS2D	0.89	-0.67
F_7	BCSCLR	14.18	0.74
F_8	BCSU	176.02	0.28
F_9	BCSV	175.81	0.54
F_{10}	BOUNDU	8.15	-0.78
F_{11}	BOUNDV	7.17	0.65
F_{12}	JACOB	0.06	0.25

Table 4-8: Main effects and interactions with scale for the CRAY experiment while scaling machine size from 1 to 4 processors with grid size 67×67×35. The standard error SE=±0.46 and the mean μ=333.87.

The effect of scaling the machine size from one processor to four processors, in Table 4-8, for the same problem size was β_0=-138.872, promising a decrease of about 139 seconds in the run time when using four processors or we could say 46 seconds per

additional processor. These results indicate that the code is scaling with number of processors. However, this scalability is not linear to the number of processors. In both experiments, besides the effects of F_8 and F_9 the effects of factors F_1, F_2, F_3, F_4, F_5, F_7, F_{10}, and F_{11} are now out of the error interval $2SE$.

4.4.2 Scaling the Number of Processors at 128×128×35 Grid Size

In the second set we used 128×128×35 grid size. Similar to the previous set of experiments the machine size was scaled from one processor to two, and from one processor to four. The results in Table 4-9, were obtained when scaling machine size from one processor to two processors. The effect of the scale factor β_0=-167.825 promises a possible run time reduction of 168 seconds. On the other hand, when scaling the number of processors from one to four the effect of scale factor was β_0=-349.51 as we can see in Table 4-10. An average of 117 seconds reduction in run time per additional processor.

The effects of scale show a good scalability of ARPS on the CRAY especially with the large grid size. The two factors F_8 and F_9 may catch our attention since they have much larger effect than other factors. The interactions with scale are not far out of the error interval, but a closer look at these interactions is desired. The following section provides further analysis of the interactions appearing in Table 4-10.

Factor	Name	Main Effect	Interaction with Scale
F_0	Machine Size	-167.83	N/A
F_1	ADVCTS	28.84	2.71
F_2	ADVU	7.85	0.72
F_3	ADVV	0.68	-6.04
F_4	ADVW	-1.25	-4.69
F_5	BCKMKH	10.48	3.05
F_6	BCS2D	1.60	3.58
F_7	BCSCLR	20.51	4.11
F_8	BCSU	180.33	4.91
F_9	BCSV	173.58	-2.90
F_{10}	BOUNDU	5.18	-3.39
F_{11}	BOUNDV	0.97	-3.52
F_{12}	JACOB	-3.81	-0.88

Table 4-9: Main effects and interactions with scale for the CRAY experiment while scaling machine size from 1 to 2 processors with grid size 128×128×35. The standard error SE=±0.44 and the mean μ=701.34.

Factor	Name	Main Effect	Interaction with Scale
F_0	Machine Size	-349.51	N/A
F_1	ADVCTS	28.62	4.11
F_2	ADVU	0.60	-7.20
F_3	ADVV	-0.25	-6.61
F_4	ADVW	-1.84	-4.83
F_5	BCKMKH	2.30	-5.29
F_6	BCS2D	-6.54	-3.36
F_7	BCSCLR	20.54	4.95
F_8	BCSU	181.42	4.83
F_9	BCSV	180.60	5.52
F_{10}	BOUNDU	13.44	4.21
F_{11}	BOUNDV	1.99	-4.80
F_{12}	JACOB	5.22	5.38

Table 4-10: Main effects and interactions with scale for the CRAY experiment while scaling machine size from 1 to 4 processors with grid size 128×128×35. The standard error SE=±0.37 and the mean μ=609.15.

4.5 Interaction Analysis

This section provides further analysis of the two factor interactions with the scale factor F_0 at 128×128×35 grid size. The number of processors was scaled from one processor to four processors (see Table 4-10). The fractional factorial design used in that experiment was of resolution 4 with 32 treatments and assumed that some of the two factor interactions are negligible so they can be confounded with each other. Each

of the two factor interaction effects estimated in the experiment with the fractional factorial design represents the sum of the effects for two factor interactions confounded with each other.

One way to find which factors are responsible for these two factor interactions is to use two-way tables and interaction plots. The two-way table for any two factors in our experiment can be obtained from the responses in the fractional factorial design. Each corner in the two-way table represents the average of eight runs where the levels of the two factors under investigation are at one of the four possible combinations: {(-,-), (-,+), (+,-), (+,+)} (See 3.2.2). Let the interaction of two factors F_0 and F_1 be denoted as F_0*F_1. The aliasing structures for the F_0*F_1, F_0*F_2, ... , F_0*F_{12} two factor interactions used in the fractional experiment are:

$$F_0*F_1 = F_4*F_{11} = F_5*F_6 = F_7*F_8 = F_9*F_{10}$$
$$F_0*F_2 = F_4*F_{10} = F_5*F_7 = F_6*F_8 = F_9*F_{11}$$
$$F_0*F_3 = F_4*F_8 = F_5*F_9 = F_6*F_{10} = F_7*F_{11}$$
$$F_0*F_4 = F_1*F_{11} = F_2*F_{10} = F_3*F_8 = F_5*F_{12}$$
$$F_0*F_5 = F_1*F_6 = F_2*F_7 = F_3*F_9 = F_4*F_{12}$$
$$F_0*F_6 = F_1*F_5 = F_2*F_8 = F_3*F_{10} = F_{11}*F_{12}$$
$$F_0*F_7 = F_1*F_8 = F_2*F_5 = F_3*F_{11} = F_{10}*F_{12}$$
$$F_0*F_8 = F_1*F_7 = F_2*F_6 = F_3*F_4 = F_9*F_{12}$$
$$F_0*F_9 = F_1*F_{10} = F_2*F_{11} = F_3*F_5 = F_8*F_{12}$$
$$F_0*F_{10} = F_1*F_9 = F_2*F_4 = F_3*F_6 = F_7*F_{12}$$
$$F_0*F_{11} = F_1*F_4 = F_2*F_9 = F_3*F_7 = F_6*F_{12}$$
$$F_0*F_{12} = F_4*F_5 = F_6*F_{11} = F_7*F_{10} = F_8*F_9$$

Each of the two factor interactions in the same line above has the same columns of signs in the 32 treatments fractional factorial design. The interaction effect for F_0*F_1, that appeared in Table 4-10, could be due to one or more of the interactions in the same line. If we could assume all interactions but one are negligible then no further analysis is needed. However, we can not make this assumption with enough confidence so it is recommended to investigate these interactions. The two-way table for F_0*F_1 is:

F_0 at (+)	418.031	450.764
F_0 at (-)	771.654	796.161
	F_1 at (-)	F_1 at (+)

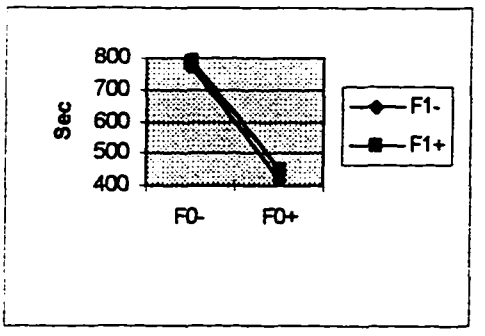


Figure 4-1: Interaction plot for the two factors F_0 and F_1.

The plot, in Figure 4-1, indicates no significant interaction between F_0 and F_1. An interaction occurs when the levels of one factor interrelate significantly with the levels of the second factor in influencing the response. In other words this can be thought of as a twist in the response surface. This is more clear in the following two-way table for the interaction F_4*F_{11}:

F_4 at (+)	605.18	611.28
F_4 at (-)	611.14	609.0
	F_{11} at (-)	F_{11} at (+)

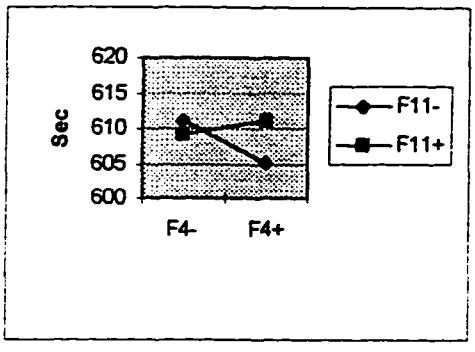


Figure 4-2: Interaction plot for the two factors F_4 and F_{11}.

The plot, in Figure 4-2, shows an interaction between F_4 and F_{11}. The effect of F_4 on the response is not the same when the level of factor F_{11} is changed. So we may accept this as a source of the interaction that appeared in Table 4-10. However, another interaction seems to exist between F_5 and F_6 as we can see in Figure 4-3 and the two-way table:

F_5 at (+)	611.52	609.09
F_5 at (-)	613.32	602.68
	F_6 at (-)	F_6 at (+)

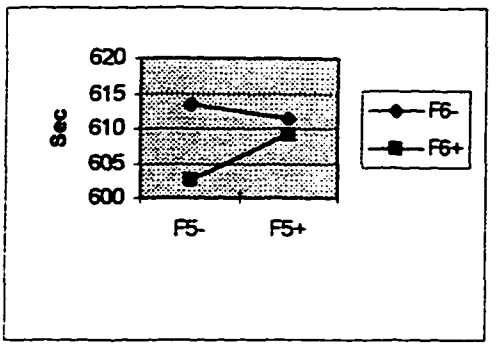


Figure 4-3: Interaction plot for the two factors F_5 and F_6.

The two factors F_7 and F_8 do not interact with each other as it is clear from the interaction plot in Figure 4-4. The two-way table was:

F_7 at (+)	526.65	712.19
F_7 at (-)	510.23	687.54
	F_8 at (-)	F_8 at (+)

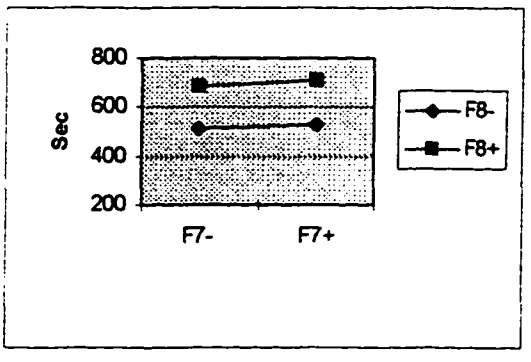


Figure 4-4: Interaction plot for the two factors F_7 and F_8.

The last interaction confounded with the above interactions is F_9*F_{10}. Figure 4-5 shows the interaction plot, and the two-way table is as follows:

F_9 at (+)	690.67	708.23
F_9 at (-)	514.19	523.52
	F_{10} at (-)	F_{10} at (+)

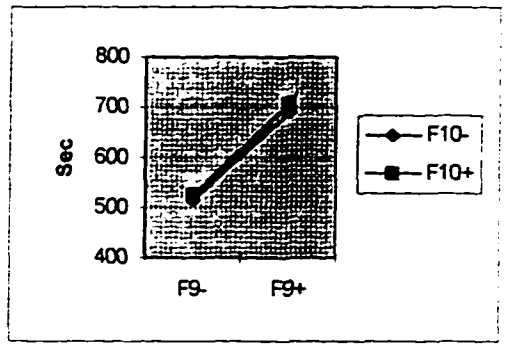


Figure 4-5: Interaction plot for the two factors F_9 and F_{10}.

From the above analysis it is clear that the assumed interaction between scale factor and F_1 in Table 4-10, was in fact due to the interactions $\beta_{4,11}$ and $\beta_{5,6}$. The other interactions, in Table 4-10, were also analyzed in the same way. The summary of the results is in Table 4-11, that was constructed from the interaction plots for all other interactions appearing in Table 4-10. The table lists the interactions that are expected to be the real source of the interaction effects.

Interaction	Effect	Source of interaction
F_0*F_1	4.11	$\beta_{4,11}$, $\beta_{5,6}$
F_0*F_2	-7.20	$\beta_{4,10}$, $\beta_{5,7}$
F_0*F_3	-6.61	$\beta_{6,10}$, $\beta_{7,11}$
F_0*F_4	-4.83	$\beta_{1,11}$, $\beta_{2,10}$, $\beta_{5,12}$
F_0*F_5	-5.29	$\beta_{1,6}$, $\beta_{2,7}$, $\beta_{4,12}$
F_0*F_6	-3.36	$\beta_{2,8}$, $\beta_{3,10}$, $\beta_{11,22}$
F_0*F_7	4.95	$\beta_{2,5}$, $\beta_{3,11}$, $\beta_{10,12}$
F_0*F_8	4.83	$\beta_{2,6}$, $\beta_{3,4}$, $\beta_{9,12}$
F_0*F_9	5.52	$\beta_{1,10}$, $\beta_{2,11}$, $\beta_{3,5}$
F_0*F_{10}	4.21	$\beta_{2,4}$, $\beta_{3,6}$, $\beta_{7,12}$
F_0*F_{11}	-4.80	$\beta_{1,4}$, $\beta_{3,7}$, $\beta_{6,12}$
F_0*F_{12}	5.38	$\beta_{4,5}$, $\beta_{6,11}$, $\beta_{7,10}$

Table 4-11: The expected sources of two factor interactions obtained form interaction plots analysis.

4.6 The Effect of Using Larger Delay

All the previous experiments used the same amount of delay. In this section we examined how the change of the delay amount affects the results. The experiment used a fixed grid size of 128×128×35 while scaling number of processors from 1 to 4, with a delay three times larger than what we used in the original experiments. The results were then compared to those in Table 4-10, which have the same conditions except for the amount of the delay.

As can be seen, in Table 4-12, the significant main effects were trebled except for the scale factor F_0 which remains almost the same. The interaction $\beta_{0,1}$ in Table 4-12, is about seven times larger than the same interaction in Table 4-10. The interaction plots for the large delay experiment still point to $\beta_{4,11}$ and $\beta_{5,6}$ as the expected real source of this interaction as it can be seen in Figures 6 and 7. It should be noticed that the interaction effects are still very small compared to the effects of factors F_8 and F_9.

Factor	Name	Main Effect	Interaction with Scale
F_0	Machine Size	-347.961	N/A
F_1	ADVCTS	72.97	28.99
F_2	ADVU	-9.37	-13.97
F_3	ADVV	-0.92	-23.78
F_4	ADVW	-3.81	-20.11
F_5	BCKMKH	4.46	-17.20
F_6	BCS2D	-16.72	-17.3
F_7	BCSCLR	61.04	18.16
F_8	BCSU	541.55	20.07
F_9	BCSV	545.83	15.92
F_{10}	BOUNDU	45.98	11.26
F_{11}	BOUNDV	2.95	-15.16
F_{12}	JACOB	18.99	16.95

Table 4-12: Main effects and interactions with scale for large delay experiment, while scaling machine size from 1 to 4 processors with grid size 128×128×35. The standard error SE=±4.87 and the mean μ=1043.12.

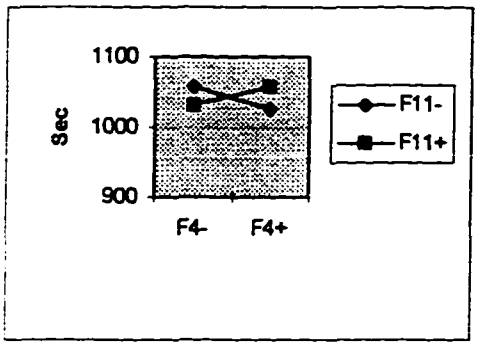


Figure 4-6: Interaction plot for the two factors F_4 and F_{11} with large delay.

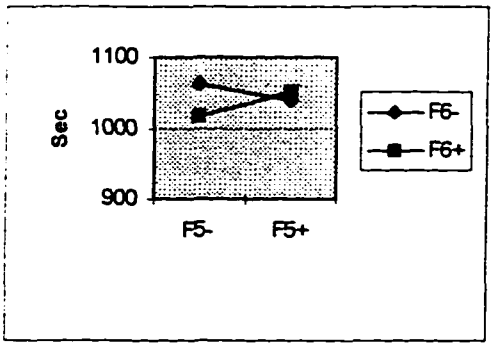


Figure 4-7: Interaction plot for the two factors F_5 and F_6 with large delay.

4.7 Speedup and Scalability

As we reviewed in Chapter 2, speedup is an important measure for parallel systems performance. The interaction plots and the analysis we saw focused on the overall wall-clock run time. It is highly recommended not use only one measure when evaluating parallel code. In this section we will look to the interaction plots from a

different angel. The y-axis in the interaction plots will represent the speedup instead of the run time. We picked the two factors F_8 and F_9 for this purpose. The average response at each row in Table 4-13 is the average of four treatments obtained from the 13 factor fractional factorial experiment at each of the three factor combinations. The speedup is then obtained from dividing the average serial run time by the average parallel run time. The speedup in this case is called relative speedup (see Chapter 2). Since the run time we used is an average of four treatments we will call this the *average relative speedup*. The speedup plot is in Figure 4-8.

F_0	F_8	F_9	Average Response
-	-	-	608.15
-	-	+	783.07
-	+	-	784.59
-	+	+	959.82
+	-	-	348.82
+	-	+	443.09
+	+	-	429.13
+	+	+	619.1601

Table 4-13: The average responses obtained from fractional factorial design.

It is clear from the plot in Figure 4-8, that the speedup is less than 2. Also an interaction seems to exist. However, the data were obtained from a 13 factor experiment with serial delays inserted in different segments of the code which might explain the low speedup. On the other hand, the interaction need to be investigated to

examine if there is a real scalability problem. For this purpose a full factorial experiment was performed with three factors F_0, F_8, and F_9 that have 8 treatments. The data in Table 4-14 is the average response of three replicates at each treatment. The speedup in this case is *relative speedup* which is plotted in Figure 4-9.

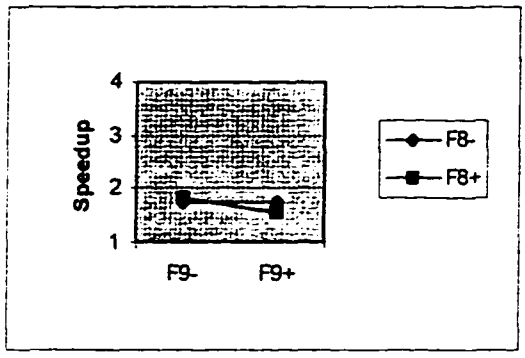


Figure 4-8: Interaction plot using speedup obtained from fractional factorial experiment.

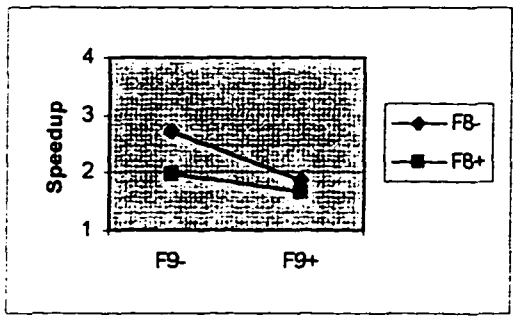


Figure 4-9: Interaction plot using relative speedup for the full factorial experiment.

F_0	F_8	F_9	Average Response
-	-	-	605.35
-	-	+	742.55
-	+	-	785.73
-	+	+	953.76
+	-	-	224.87
+	-	+	400.26
+	+	-	400.02
+	+	+	575.95

Table 4-14: The average responses of three replicates of a full factorial design.

It was clear that the speedup was higher in the full factorial since many of the serial delays inserted in the code are not present in this case. On the other hand, the interaction of the two factors seems to be a result of the averaging of four treatments which may introduce some noise from the other factors. The isolation of the effects of these factors using blocking is not possible since we used fractional factorial design. We may conclude that the absence of interaction indicates no scalability problems of the investigated segments of the code.

4.8 Summary

The results obtained from the 13 factor experiments revealed very useful information about the possible bottlenecks in the ARPS code when executed on CRAY J90. The subroutines "BCSU" and "BCSV" are of great interest since they have a high effect

on the response (overall wall-clock run time). The effect of these subroutines remained significant and almost the same regardless of the changes in grid size or number of processors. These two subroutines are a great target for future optimization of the ARPS code.

The effect of the scale factor, F_0, could be used as a relative measure of scalability as we did in sections 4.3 and 4.4. On the other hand, the absence of real interaction between scaling the number of processors and other factors indicates no scalability problems with these factors. Increasing the amount of the delay did not change the primary conclusions about the bottlenecks in the code as we experienced in Section 4.6.

We examined the interactions with scale factor, in Section 4.5, which indicates some other two factor interactions. The interactions between two factors representing segments of code in a shared memory environment could indicate some kind of contention. A closer look at these subroutines representing the interacting factors may lead to more clues about the nature of these interactions. Also a full factorial experiment for the interacting factors would be informative in this case. However, the interaction effects were not as large compared to the effects of F_8 and F_9 even with larger delay.

In Section 4.7, we used speedup to plot the interaction of two factors as an indicator for scalability. The average relative speedup is not suitable in this case, as we experienced, the relative speedup is more appropriate for interaction plots since it

eliminates the effects of other factors in the fractional factorial experiment with large number of factors.

The next chapter will cover the experiments on a distributed memory platform. The ARPS code is designed to be portable, and some of the results for a shared memory machine may not hold on a distributed memory machine.

CHAPTER 5

ANALYSIS OF ARPS CODE ON IBM SP2

In Chapter 4 the ARPS code was analyzed on the shared memory machine CRAY J90. We will present in this chapter the experiments results and analysis of the ARPS code on IBM SP2 which is a distributed memory machine with high performance switches connecting the processing elements. The effects of varying the problem size as well as varying the number of processors are discussed in Section 5.2. In section 5.3 we examine scalability by employing the speedup plots. Moreover, we examined the effect of changing the communication medium in Section 5.4. The following section describes the environment we used to run the experiments and the organization of the IBM SP2.

5.1 The IBM SP2

The IBM SP2 machine we used for this analysis had 8 thin nodes (66.7 MHz) each with 512MB RAM of local memory. Under the current configuration only fife of these eight nodes were available for parallel jobs, the other three are used by serial jobs and interactive jobs. When a parallel job acquires a number of processors these processors remain reserved for it until termination of the job. The program may use either the high performance switches (*hps*) or the slow Ethernet links as a communication hardware.

For the experiments in this report we used the *hps* as communication medium unless otherwise stated.

We used the parallel version of ARPS which uses MPI (Message Passing Interface) for exchanging boundary data between processors. The details about the parallel version of ARPS could be found in (Sathye et al. 1995) and (Xue et al. 1995). The parallel code was compiled using the "mpxlf" script which links the communication libraries for message passing to the code. While the serial version was compiled using "xlf" which does not include the message passing libraries. Since no communication library was specified for the parallel version at compile time, the proper library will be linked dynamically at run time depending on the system varaiables.

5.2 Description of the Experiments

Similar to the experiments we performed on CRAY J90, we used the same 13 factors fractional factorial design with 32 treatments. While we were able to reach 256×256×35 grid size on CRAY J90, we were limited by the memory space of each individual processor on the distributed memory machine. As we did in Chapter 4, we will examine the effect of scaling the problem size when running on one node by scaling the grid size from 67×67×35 to 131×131×35. Then at 67×67×35 we will scale the number of processors from one to two in one experiment, and from one to four in the other experiment. At 131×131×35 grid size the number of processors will be scaled in the same manner. We were unable to reach 259×259×35 on a single node

because of the memory limitations. The grid sizes we used on the distributed memory machine were slightly different from the shared memory machine. This change is needed just to make the grid size divisible by the number of processors along each axis when calculating the sub-grid sizes. In the following sections each of the experiments will be described in more details.

5.2.1 Scaling the Problem Size

The problem size scaling experiment was performed on one of the IBM SP2 nodes when the problem size was scaled from 67×67×35 to 131×131×35 grid size. The estimated effect of scaling the problem size was β_0=1464.86 indicating about 24 minutes possible increase in average run time. The standard error was SE=±2.33 and the mean μ=1493.47 seconds. There were no significant interactions between the scale factor and the other factors outside the $2SE$ interval as it appears in Table 5-1.

The other significant factors outside the $2SE$ interval were: F_1, F_2, F_3, F_4, F_5, F_7, F_8, F_9, F_{10}, and F_{11}. The factors F_8 and F_9 are still holding the highest effects. The estimated effect of factor F_8 was β_8=217.04, and the estimated effect of F_9 was β_9=213.00. The next two largest effects were those of the factors F_1 and F_7 with estimated effects β_1=27.56 and β_7=16.73 respectively.

Factor	Name	Main Effect	Interaction with Scale
F_0	Grid Size	1464.86	N/A
F_1	ADVCTS	27.56	1.28
F_2	ADVU	5.23	3.13
F_3	ADVV	8.11	2.66
F_4	ADVW	4.98	0.05
F_5	BCKMKH	9.74	-1.47
F_6	BCS2D	-0.63	-0.85
F_7	BCSCLR	16.73	-0.44
F_8	BCSU	217.04	3.23
F_9	BCSV	213.00	-1.18
F_{10}	BOUNDU	7.16	-0.79
F_{11}	BOUNDV	8.24	0.47
F_{12}	JACOB	-0.74	-1.35

Table 5-1: Main effects and interaction effects with scale when scaling the problem size from 67×67×35 to 131×131×35. The mean μ=1493.47 the estimated standard error SE=±2.33.

5.2.2 Scaling Number of Processors

The next set of experiments scaled the system size in terms of number of processors. The experiments were performed at two levels of grid sizes. For each level the number of processors was scaled from one processor to two processors and from one processor to four processors.

The effects and interactions in Table 5-2 were estimated when the system size was scaled from one to two processors at grid size 67x67x35. The effect of system scale factor β_0=-235.56 promises an average decrease of 235 seconds when scaling to two processors. The estimated standard error was SE=±0.88 and the mean was μ=643.26.

Factor	Name	Main Effect	Interaction with Scale
F_0	Machine Size	-235.56	N/A
F_1	ADVCTS	27.05	0.77
F_2	ADVU	03.01	0.91
F_3	ADVV	5.55	0.1
F_4	ADVW	4.79	-0.14
F_5	BCKMKH	9.77	-1.44
F_6	BCS2D	-0.13	-0.36
F_7	BCSCLR	18.39	1.22
F_8	BCSU	213.38	-0.43
F_9	BCSV	213.56	-0.62
F_{10}	BOUNDU	8.78	0.82
F_{11}	BOUNDV	8.3	0.53
F_{12}	JACOB	0.25	-0.37

Table 5-2: Main effects and interaction effects with scale when scaling the system size from one to two processors at grid size 67x67x35. The mean μ=643.26 and the estimated standard error SE=±0.88.

When the system size was scaled from one processor to four processors, as in Table 5-3, the effect of the scale factor was β_0=-350.02 with mean μ=586.03 and estimated standard error SE=±0.99. This could mean a decrease of 116 seconds per additional processor. In both the experiments the effects of F_8 and F_9 were the highest.

Factor	Name	Main Effect	Interaction with Scale
F_0	Machine Size	-350.02	N/A
F_1	ADVCTS	28.02	1.74
F_2	ADVU	3.17	1.07
F_3	ADVV	5.54	0.09
F_4	ADVW	4.31	-0.62
F_5	BCKMKH	10.94	-0.26
F_6	BCS2D	0.57	0.34
F_7	BCSCLR	17.42	0.25
F_8	BCSU	214.12	0.31
F_9	BCSV	214.88	0.69
F_{10}	BOUNDU	8.57	0.61
F_{11}	BOUNDV	7.72	-0.05
F_{12}	JACOB	0.31	-0.3

Table 5-3: Main effects and interaction effects with scale when scaling the system size from one to four processors at grid size 67×67×35. The mean μ=586.03 and the estimated standard error SE=±0.99.

Next we examined the effect of changing number of processors at 131×131×35 grid size. The results in Table 5-4 are obtained when the system size was scaled from one

to two processors. The effect of scale factor F_0 is β_0=-948.05 which could mean about 15 minutes reduction in run time when using two processors. The estimated standard error SE=±1.44 and the mean was μ=1747.59.

Factor	Name	Main Effect	Interaction with Scale
F_0	Machine Size	-948.05	N/A
F_1	ADVCTS	27.22	0.21
F_2	ADVU	4.63	0.0
F_3	ADVV	7.2	-2.44
F_4	ADVW	2.03	2.16
F_5	BCKMKH	6.13	2.88
F_6	BCS2D	-1.45	0.84
F_7	BCSCLR	18.41	1.14
F_8	BCSU	212.80	0.84
F_9	BCSV	213.27	-0.13
F_{10}	BOUNDU	10.19	-0.02
F_{11}	BOUNDV	8.46	-0.56
F_{12}	JACOB	1.28	-1.44

Table 5-4: Main effects and interaction effects with scale when scaling the system size from one to two processors at grid size 131×131×35. The mean μ=1747.59 and the estimated standard error SE=±1.44.

The results of scaling the system size from one processor to four processors for the same grid size are in Table 5-5. The estimated effect of the scale factor F_0 was β_0=-1431.81 that means scaling from one to four processors could result in decreasing

the run time about 23 minutes on average. The mean was μ=1505.71 and the estimated standard error was SE=±1.99. The two factors F_8 and F_9 are still holding the highest effects.

Factor	Name	Main Effect	Interaction with Scale
F_0	Machine Size	-1431.81	N/A
F_1	ADVCTS	29.55	2.54
F_2	ADVU	3.44	-1.2
F_3	ADVV	6.68	-2.95
F_4	ADVW	-0.16	-0.03
F_5	BCKMKH	6.74	3.49
F_6	BCS2D	0.28	2.57
F_7	BCSCLR	18.08	0.81
F_8	BCSU	213.76	1.8
F_9	BCSV	214.41	1.01
F_{10}	BOUNDU	9.75	-0.46
F_{11}	BOUNDV	7.73	-0.17
F_{12}	JACOB	0.49	-2.23

Table 5-5: Main effects and interaction effects with scale when scaling the system size from one to four processors at grid size 131×131×35. The mean μ=1505.71 and the estimated standard error SE=±1.99.

5.3 Speedup and Scalability

In this section the speedup of the code is analyzed with respect to changing two factors. We selected the two factors with highest effect namely F_8 and F_9. The speedup was measured at grid size 131×131×35 when changing the system size from one processor to four processors. The data in Table 4-6 represent the average of four runs obtained from the 13 factor fractional factorial experiment at each of the three factor combinations. The interaction plot in Figure 5-1 shows that the speedup is about the same when F_9 is at low level regardless of the level of F_8.

F_0	F_8	F_9	Average Response
-	-	-	2007.57
-	-	+	2223.69
-	+	-	2222.25
-	+	+	2432.93
+	-	-	693.49
+	-	+	790.603
+	+	-	790.74
+	+	+	1004.43

Table 5-6: The average responses obtained from fractional factorial design.

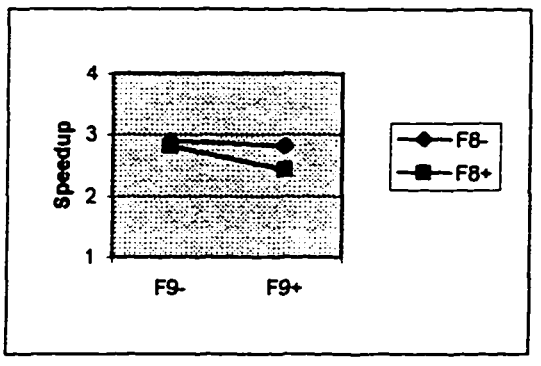


Figure 5-1: Interaction plot of the speedup from the 13 factors fractional factorial experiment.

To eliminate the effect of other factors a separate experiment was conducted with only 3 factors. The full factorial design appearing in Table 5-7 has 8 treatments, and the average response column is the average of three replicates. We noticed that the speedup rates in the full factorial experiment were higher due to the absence of the extra serial delays caused by the remaining 10 factors in the 13 factor fractional factorial experiment. However, the speedup rates appearing in Figure 5-2 have similar behavior to those plotted in Figure 4-9.

The estimated standard error for the experiment in Table 5-6 was $SE=\pm 1.99$, and the mean was $\mu=1505.71$, While the estimated standard error for the full factorial experiment in Table 5-7 was $SE=\pm 7.22$ and the mean was $\mu=1461$.

F_0	F_8	F_9	Average Response
-	-	-	1966.45
-	-	+	2173.58
-	+	-	2174.57
-	+	+	2396.68
+	-	-	530
+	-	+	744.87
+	+	-	742.81
+	+	+	960.21

Table 5-7: The average responses for the full factorial experiment.

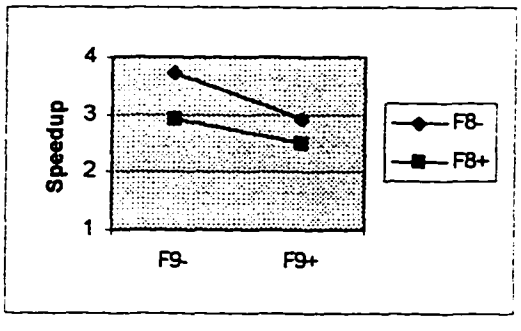


Figure 5-2: The interaction plot of the speedup for the 3 factors full factorial experiment.

5.4 The Effect of Communication Network

The IBM SP2 allows the user to select the communication medium and the network protocol for message passing. With the use of high performance switch adapters the message passing subsystem interfaces with the user space protocol. The user has also

the option to use the IP interface of the message passing subsystem. The user space protocol does not allow more than one process per node while the IP does. However, the current configuration of the machine does not allow more than one job on the parallel nodes. The other communication medium is the Ethernet adapters which allows only IP interface of the message passing subsystem.

In the previous experiments we executed the parallel ARPS using the high performance switches and the US interface. We picked the experiment at 131×131×35 grid size when the number of processors was scaled from one to four processors. Then we repeated the same set of runs using high performance switches *hps*, but using the IP instead of the US interface. Finally we performed an experiment using Ethernet adapters which allows the use of the IP interface only. The results in Table 5-5, where obtained when running the code using *hps* and US interface. In Table 8, the results when using the IP interface with *hps*. The effect of scaling number of processors from one to four was β_0=-1416.71 with estimated mean μ=1513.26 and standard error SE=±2.5.

Factor	Name	Main Effect	Interaction with Scale
F_0	System Size	-1416.71	N/A
F_1	ADVCTS	29.17	2.16
F_2	ADVU	5.26	0.63
F_3	ADVV	7.34	-2.29
F_4	ADVW	0.96	1.09
F_5	BCKMKH	7.65	4.4
F_6	BCS2D	-1.48	0.81
F_7	BCSCLR	19.54	2.26
F_8	BCSU	216.70	4.75
F_9	BCSV	213.1	-0.3
F_{10}	BOUNDU	8.22	-1.99
F_{11}	BOUNDV	9.12	1.22
F_{12}	JACOB	1.47	-1.25

Table 5-8: Main effects and interaction effects with scale when scaling the system size from one to four processors at grid size 131×131×35 using *hps* and IP. The mean μ=1513.26 and the estimated standard error SE=±2.5.

For the third experiment we used the Ethernet adapters with the IP interface. The results in Table 5-9 show the estimated effect of the scale factor β_0=-1289.45 with mean μ=1576.88 and SE=±1.5.

Factor	Name	Main Effect	Interaction with Scale
F_0	System Size	-1289.45	N/A
F_1	ADVCTS	26.80	-0.21
F_2	ADVU	4.7	0.08
F_3	ADVV	7.86	-1.77
F_4	ADVW	3.32	3.45
F_5	BCKMKH	6.72	3.47
F_6	BCS2D	-1.49	0.8
F_7	BCSCLR	17.76	0.49
F_8	BCSU	218.17	6.21
F_9	BCSV	219.02	5.62
F_{10}	BOUNDU	9.71	-0.5
F_{11}	BOUNDV	7.76	-0.14
F_{12}	JACOB	2.34	-0.39

Table 5-9: Main effects and interaction effects with scale when scaling the system size from one to four processors at grid size 131×131×35 using Ethernet and IP. The mean μ=1576.88 and the estimated standard error SE=±1.5.

Communication Medium	Mean (μ)	Effect of Scale (β_0)	Standard Error (SE)
hps with US	1505.71	-1431.81	±1.99
hps with IP	1513.26	-1416.71	±2.5
Ethernet with IP	1576.88	-1289.45	±1.5

Table 5-10: Summary of the effects of using different communication mediums.

We may notice that the effect of scaling the number of processor is reduced slightly in magnitude when using IP interface instead of US interface, that was about 15 seconds. While the effect of scale with the use of Ethernet limits the scalability of the code by about 150 seconds which is ten times the difference when using the *hps* and IP.

5.5 Summary

In this chapter the ARPS code was analyzed when running in a distributed memory environment. We used the MPI parallel version of ARPS on IBM SP2. The results are in the same line with the shared memory version on CRAY J90. The two factors "BCSU" and "BCSV" are still holding the highest effect. Their was no interaction between these two factors and the scale factor. The two factor interactions that were noticed in some of the results obtained from running ARPS on CRAY J90 are within the error margin in the IBM SP2 case.

We noticed that the wall-clock running time of CRAY J90 almost twice that of IBM SP2 when comparing the estimated mean. For grid size 67×67×35 and when scaling from one to two processors the mean was μ=354.8 on CRAY J90 while it was μ=643.26 on IBM SP2. For the same grid size when scaling from one to four processors the estimated mean on CRAY was μ=333.87 while it was μ=586.03 on IBM SP2. Similar results noticed on larger grid size. Nevertheless, the effect of scale factor was higher on the IBM SP2 than on CRAY J90. For example, the effect of scale factor F_0 at grid size 67×67×35 when scaling the number of processors from one to

four processors was β_0=-350.02on IBM SP2, while it was β_0=-138.87 on CRAY J90, that is more than twice the effect.

The speedup rates were plotted when changing two factors to study the effect of synthetic perturbations on speedup and use this as an indication of scalability. This way could be used to predict how the code will scale when the code is optimized at specific segments of code.

While the experiments in this chapter were conducted when using the high performance switches with user space communication library (US), we also examined the effect of the change in communication medium on the scalability of ARPS.

CHAPTER 6

CONCLUSIONS

The research work for this dissertation aimed to fill a gap in the process of evaluating and tuning parallel systems. Unlike sequential code, parallel code is highly dependent on the underlying architecture and the decomposition of the problem. It is a challenge to find an approach that will assess the parallel system without extra hardware and software components that will increase the cost of evaluation process and limit its portability. On the other hand, the complex nature of parallel systems requires the consideration of more than one factor at a time. Factorial designs were a good tool for studying simultaneously the effects of more than one factor. While the use of experimental design techniques existed in various scientific and engineering fields, factorial designs were only recently introduced to the evaluation process of parallel systems. The approach we followed was suggested by Gordon Lyon and his colleagues at NIST. The approach uses synthetic perturbations to simulate changes in the code and capture the response sensitivity to these changes by employing factorial designs. We applied this approach to a large code on different platforms, and estimated the scalability of the code based on the measured sensitivity. We further analyzed the interaction between different factors that will appear as a twist in the response surface. Moreover, we introduced a class of interaction plots that uses speedup ratio. Then we used these plots for comparing the scalability of two systems.

The use of speedup provided a normalized value that can be compared among different architectures rather than the effects that may have no direct meaning on different architectures.

In Chapter 2, we reviewed the literature related to the evaluation process of parallel systems. The speedup with its different versions was reviewed as well as efficiency and other measures of performance. In that chapter we looked to the scalability from three points of view: machine scalability, algorithm scalability, and scalability of machine-algorithm combination. The later was of our interest because of the nature of parallel systems. We also introduced our definition of a parallel system. Two major approaches: isoefficiency and isospeed, were explored besides some other approaches.

Our approach was introduced in Chapter 3. In that chapter a brief background of the experimental design with a focus on factorial designs. We provided the necessary definitions to introduce factorial designs. The fractional factorial designs were introduced since it will reduce the number of treatments needed. Also we show how the computed effects can be used to build a linear model describing a relation between the investigated factors and the response. The link between factorial designs and parallel systems was introduced in Section 3.3, followed by an illustrative example. Then we briefly gave a description of ARPS followed by the initial experiment on a cluster of workstations.

We performed more detailed experiments on the CRAY J90 to explore the power of the statistical approach in estimating the scalability of the parallel system,

these results are described in Chapter 4. We scaled the problem size as well as the number of processors. The effects of scale were used to assess the scalability of the parallel system. The existence of interactions with the scale factor that may affect our conclusion about main effects, was investigated by analyzing these interactions using interaction plots. The speedup interaction plot was used in this chapter to estimate scalability of some code segments as well as the machine. We further examined the effect of the amount of delay by comparing the results of two experiments designed for this purpose.

The next Chapter 5 described the experiments performed on the IBM SP2. Similar experiments were performed within the limitations of memory space on each node. The problem size and the number of processors were both scaled. The results confirmed the primary conclusions about the code.

The comparison between the results obtained on the two machines: CRAY J90 and IBM SP2, show that the former has better performance as it is clear from the estimated means, while the later show better scalability as we observed from the speedup interaction plots. This could be due to the difference in the single processor performance between the two machines, were CRAY J90 uses vector processor and IBM SP2 uses RISC processor. Also the parallel version was decomposed by the programmer, along the x-axis and y-axis of the grid space, on the IBM SP2 that provided a scalable structure, while it was done by the compiler on the CRAY J90. Moreover, there is a possibility of memory contention in the shared memory case as it appeared in some of the interactions in Section 4.5. However, the magnitudes of these

interactions were small and the existence of such contention can be verified by using hardware monitoring tools.

The use of designed experiments with synthetic perturbations in assessing parallel systems is portable, economic, and simple. However, the results will depend on how the investigator will pickup the factors. While the problem size and the number of processors are the two important factors to study the scalability of parallel system, the selection of other factors representing segments of codes is crucial and requires a certain level of knowledge about the code as it was mentioned in Chapter 3. Screening will help in the selection of these significant factors. A study of the code and its structure without going deep in the details of each component was of great help in our case. Before selecting the factors to be included in our experiments a calling tree was constructed. Also we studied the general function and communication patterns for each subroutine involved in our experiments.

The investigation process is iterative and the results of one experiment may lead to other experiments. Therefore the screening experiments will include a large number of factors requiring a large number of treatments. The number of treatments can be reduced by using fractional designs in the screening phase, then for the significant factors a full design is used to obtain more details.

For the experiments in this research we used two level factorial and fractional factorial designs. Each factor has exactly two levels this limits the study of the scalability to fit linear models only. As we noticed in the different experiments the effect of changing the number of processors was not linear. To build a quadratic model

at least three levels are needed. However, when using more than two levels the levels should be equally spaced from one another. The future work will consider using three or more levels for the scale factors allowing building non-linear models.

The scale of problem size and number of processors were considered separately in this research. Our future trend is to consider two or more scale factors in one experiment. This may include the number of processors, the problem size, the algorithm instant, and the communication medium. The number of factors and the levels at each factor, however, will vary depending on the goals and objectives of the investigation.

REFERENCES

Alabdulkareem, M.; S. Lakshmivarahan; and S. K. Dhall. 1997. "A Scalability Analysis of Large Code of Interest in Meteorology Using Experimental Design Techniques." Proceedings of the High Performance Computing '97 (Atlanta, Georgia, April 6-10).

Barragy, E.; G. Carey; and V. De Geun. 1995. "Parallel Performance and Scalability for Block Preconditioned Finite Element (*p*) Solution of Viscous Flow." *International Journal for Numerical Methods in Engineering* 38, no. 9 (May): 1535~1554.

Box, G. and J. Hunter. 1961. "The 2^{k-p} Fractional Factorial Designs." *Technometrics* 3, no. 3 (Aug.): 311~352.

Box, G.; W. Hunter; and J. S. Hunter. 1978. Statistics for Experimenters, John Wiley & Sons Inc., New York.

Brachini, A.; A. Marconi; M. R. Nazzarelli; and S. Sabina. 1995. "An Integrated Approach to Performance and Testing Analysis for Parallel Systems." *Proceedings of the High Performance Computing and Networking International Conference and Exhibition* (Milan, Italy, May 3-5).

Carey, G. 1994. "A Prototype Scalable, Object Oriented Finite Element Solver on Multicomputers." *Journal of Parallel and Distributed Computing* 20, no. 3 (Mar.): 357~379.

Cray Research Inc. 1993. CF77 Commands and Directives, SR-3771 6.0, Cray Research Inc., Mendota Heights, MN.

Crowl, L. A. 1994. "How to Measure, Present, and Compare Parallel Performance." *IEEE Parallel & Distributed Technology* 2, no. 1 (Spring): 9~25.

Dhekne, P. S.; K. Rajesh; S. M. Mahajan. 1995. "Performance of ANUPAM system for Large Scale Parallel Computations." *Proceedings of the International Conference on High Performance Computing* (New Delhi, India, Dec. 27-30).

Droegemeier, K.; M. Xue; K. Johnson; K. Mills; M. O'keefe. 1992. "Experiences with the Scalable-Parallel ARPS Cloud/Mesoscale Prediction Model on Massively Parallel and Workstation Cluster Archtectures." *Proceedings of the Fifth ECMWF Workshop on the Use of Parallel Processors in Meteorology* (Reading, UK, Nov. 23-27).

Drouin, N. 1993. "Building Hadamard Matrices in Steps of 4 to order 200." Technical Report NISTIR-5121. U. S. Department of Commerce, Technology Administration, National Institute of Standards and Technology, Gaithersburg, MD 20899.

Eager, D.; J. Zahorjan; and E. Lazowska. 1989. "Speedup Versus Efficiency in Parallel Systems." *IEEE Transactions on Computers* 38, no. 3 (Mar.): 408~423.

Ertel, W. 1994. "On the Definition of Speedup." *In the Proceeding of the 6th International PARLE Conference* (Athens, Greece, July 4-8).

Fienup, A. and S. Kothari. 1994. "A Memory Constrained Scalability Metric." *Proceedings of the 23rd International Conference on Parallel Processing* (Aug. 15-19). CRC Press, Boca Raton, FL.

Grama, A. Y.; A. Gupta; and V. Kumar. 1993. "Isoefficiency: Measuring the scalability of parallel algorithms and architectures." *IEEE Parallel & Distributed Technology* 1, no. 3 (Aug.): 12~21.

Gunter, B. 1993. "How statistical design concepts can improve experimentation in the physical sciences." *Computers in Physics* 7, no. 3 (May): 262~272.

Gupta, A. and V. Kumar. 1993. " Scalability of Parallel Algorithms for Matrix Multiplication." *Proceedings of the 1993 International Conference on Parallel Processing Vol. III* (Syracuse, NY Aug. 16-20).

Gupta, A. and V. Kumar. 1993. "The Scalability of FTT on Parallel Computers." *IEEE Transactions on Parallel and Distributed Systems* 4, no. 8 (Aug.): 922~932.

Gupta, A.; V. Kumar; and A. Sameh. 1995. "Performance and Scalability of Preconditioned Conjugate Gradient Methods on Parallel Computers." *IEEE Transactions on Parallel and Distributed Systems* 6, no. 5 (May): 455–469.

Gustafson, J. 1988. "Reevaluating Amdahl's Law." *Communications of the ACM* 31, no. 5 (May): 532–533.

Gustafson, J.; G. Montry; and R. Benner. 1988. "Development of Parallel Methods for A 1024-Processor Hypercube." *SIAM Journal on Scientific and Statistical Computing* 9, no. 4 (July): 609–638.

Gustavson, D. 1994. "The Many Dimensions of Scalability." *Digest of Papers from the COMPCON 94* (San Francisco, California, Feb. 28 - Mar. 4). IEEE Computer Society Press.

Hanebutte, U.; R. Joslin; and M. Zubair. 1994. "Scalability Study of Parallel Spatial Direct Numerical Simulation Code on IBM SP1 Parallel Super Computer." Technical Report No. 94-80. ICASE, NASA Langley Research Center, Hampton, VA 23681-0001.

Hedayat, A. and W. Wallis. 1978. "Hadamard Matrices and Their Applications." *The Annals of Statistics* 6, no. 6 (Sep.): 1184–1238.

Hill, M. 1990. "What is Scalability?" *Computer Architecture News* 18, no. 4 (Dec.): 18–21.

Hofmann, R.; R. Klar; B. Mohr; A. Quick; and M. Siegle. 1994. "Distributed Performance Monitoring: Methods, Tools, and Applications." *IEEE Transactions on Parallel and Distributed Systems* 5, no. 6 (June): 585~598.

Jamieson, L.; A. Khokhar; and J. Patel. 1995. "Algorithm Scalability: A Poly-Algorithmic Approach." *Challenges for Parallel Processing: Proceedings of the International Conference on Parallel Processing*, edited by D. P. Agrawal. CRC Press, Boca Raton, FL.

Johan, Z.; K. Mathur; S. L. Johnsson; T. Hughes. 1994. "Scalability of Finite Element Applications on Distributed Memory Parallel Computers." *Computer Methods in Applied Mechanics and Engineering* 119, no. 1,2 (Nov.): 61~72.

Johnson, K.; J. Bauer; G. Riccardi; K. Droegemeier; and M. Xue. 1994. "Distributed Processing of A Regional Prediction Model" *Monthly Weather Review* 122, no. 11 (Nov.): 2558~2572.

Justo, G. R. 1995. "A Graphical Approach to Performance Oriented Development of Parallel Programs." *Proceedings of the International Conference on High Performance Computing* (New Delhi, India, Dec. 27-30).

Koufaty, D.; X. Chen; D. Poulsen; and J. Torrellas. 1996. "Data Forwarding in Scalable Shared Memory Multiprocessors." *IEEE Transactions on Parallel and Distributed Systems* 7, no. 12 (Dec.): 1250~1264.

Kremien, O. 1995. "Scalability in Distributed Systems, Parallel Systems and Supercomputers." *Proceedings of the International Conference and Exhibition on High Performance Computing and Networking* (Milan, Italy, May 3-5).

Lakshmivarahan, S. and S. K. Dhall. 1990. Analysis and Design of Parallel Algorithms, McGraw-Hill Inc., New York.

Liotopoulos F. K. 1994. "Sparse Generalized Hyper Grids for Performance Scalability." *Proceedings of the 6th International PARLE Conference* (Athens, Greece, July 4-8, 1994).

Lyon, G.; R. Kacker; and A. Linz. 1995. "A scalability test for parallel code." *Software Practice and Experience* 25, no. 12 (Dec.): 1299~1314.

Lyon, G.; R. Snelick; and R. Kacker. 1994. "Synthetic perturbation tuning of MIMD programs." *The Journal of Supercomputing* 8, no. 1 (Mar.): 5~27.

Malony, A. D.; D. A. Reed; and H. A. G. Wijshoff. 1992. "Performance Measurement Intrusion and Perturbation Analysis." *IEEE Transactions on Parallel and Distributed Systems* 3, no. 4 (July): 433~450.

Marenzoni, P. 1995. "Performance Analysis of Cray T3D and Connection Machine CM-5: a Comparison." *Proceedings of the High Performance Computing and Networking International Conference and Exhibition* (Milan, Italy, May 3-5).

Muller-Wichards, D. and W. Ronsch. 1995. "Scalability of Algorithms: An Analytical Approach." *Parallel Computing* 21, no. 6 (June): 937~952.

Nussbaum, D. and A. Agarwal. 1991. "Scalability of Parallel Machines." *Communications of the ACM* 34, no. 3 (Mar.): 56~61.

Panwar, R. and G. Agha. 1994. "A Methodology for Programming Scalable Architectures." *Journal of Parallel and Distributed Computing* 22, no. 3 (Sep.): 479~487.

Ramachandran, U., G. Shah, and S. Ravikumar. 1993. "Scalability Study of the KSR-1." *Proceedings of the 1993 International Conference on Parallel Processing Vol. I* (Syracuse, NY Aug. 16-20).

Ramachandran, U.; G. Shah; S. Ravikumar; and J. Muthukumarasamy. 1993. "Scalability Study of the KSR-1." *Proceedings of the 1993 International Conference on Parallel Processing Vol. I* (Syracuse, NY Aug. 16-20).

Sahni, S. and V. Thanvantri. 1996. "Performance Metrics: Keeping the Focus on Runtime." *IEEE Parallel & Distributed Technology* 4, no. 1 (Spring): 43~56.

Sarukkai S.; P. Mehra; and R. Block. 1995. "Automated Scalability Analysis of Message Passing Parallel Programs." *IEEE Parallel and Distributed Technology* 3, no. 4 (Winter): 21~32.

Sathye, A.; G. Bassett; K. Droegemeier; and M. Xue. 1995. "Towards operational severe weather prediction using massively parallel processors." *Proceedings of the International Conference on High Performance Computing* (New Delhi, India, Dec. 27-30).

Shank, C.; G. Craig; and D. Lea. 1996. "A Path to Scalability and Efficient Performance." In Languages, Compilers and Run-Time Systems for Scalable Computers, ed. Boleslaw Szymanski and Balaram Sinharoy, KLUWER ACADEMIC PUBLISHERS, Boston.

Singh, J. P.; J. L. Hennessy; and A. Gupta. 1993. "Scaling Parallel Programs for Multiprocessors: Methodology and Examples." *IEEE COMPUTER* 26, no. 7 (July): 42~50.

Sivasubramaniam, A.; A. Singla; U. Ramachandran; and H. Venkateswaran. 1994. "An approach to scalability study of shared memory parallel systems." *Proceedings of the 1994 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems* (Nashville, Tennessee, May 16-20). ACM Press.

Snelick, R.; J. Jaja; R. Kacker; and G. Lyon. 1993. "Using Synthetic-Perturbation Techniques for Tuning Shared Memory Programs." *Proceedings of the 1993 International Conference on Parallel Processing Vol. II* (Syracuse, NY Aug. 16-20).

Srivastava, J. N. 1990. "Modern Factorial Design Theory for Experimenters and Statisticians." In Statistical Design and Analysis of Industrial Experiments, ed. Subir Ghosh, 311-406, MARCEL DEKKER, INC., New York.

Sun, X. and D. Rover. 1994. "Scalability of Parallel Algorithm-Machine Combinations." *IEEE Transactions on Parallel and Distributed Systems* 5, no. 6 (June): 599~613.

Sun, X. and J. Zhu. 1995. "Performance Considerations of Shared Virtual Memory Machines." *IEEE Transactions on Parallel and Distributed Systems* 6, no. 11 (Nov.): 1185~1194.

Sun, X. and L. Ni. 1993. "Scaleable Problems and Memory-Bounded Speedup." *Journal of Parallel and Distributed Computing* 19, no. 1 (Sep.): 27~37.

Tambouris, E. and P. Santen. 1995. "A Methodology for Performance and Scalability Analysis." *Proceedings of the 22nd Seminar on Current Trends in Theory and Practice of Informatics* (Milovy, Czech Republic, Nov. 23 - Dec. 1).

Wilson, G. 1993. "A Glossary of Parallel Computing Terminology." *IEEE Parallel & Distributed Technology* 1, no. 1 (Feb.): 52~67.

Worley, Patrick 1990. "The Effect of Time Constraints on Scaled Speedup." *SIAM Journal on Scientific and Statistical Computing* 11, no. 5 (Sep.): 838~858.

Xue, M.; K. Droegemeier; V. Wong; A. Shapiro; and K. Brewster. 1995. ARPS Version 4.0 User's Guide. Center for Analysis and Prediction of Storms, The University of Oklahoma, Norman, Oklahoma.

Yan, J.; S. Sarukkai; and P. Mehra. 1995. "Performance Measurement, Visualization and Modeling of Parallel and Distributed Programs using the AIMS Toolkit." *Software-Practice and Experience* 25, no. 4 (Apr.): 429–461.

Zhang X. and Zhichen Xu. 1995. "A Semi-Empirical Approach to Scalability Study." *Proceedings of the 1995 ACM SIGMETRICS Joint International Conference on Measurement and Modeling of Computer Systems* (May 1995).

Zhang X.; Y. Yan; and Q. Mia. 1994. "Measuring and Analyzing Parallel Computing Scalability." Technical Report TR-94-03-02. High Performance Computing and Software Laboratory, The University of Texas at San Antonio, San Antonio, Texas.

APPENDIX A

OVERVEIW OF ARPS

The Advanced Regional Prediction System (ARPS) was developed by group of researchers at the Center for Analysis and Prediction of Storms (CAPS), University of Oklahoma. ARPS model is one of six major areas under CAPS program. The development of the ARPS model started in July, 1990 with ARPS version 1.0, but the first formal release was version 3.0 in September, 1992.

A.1 Description of ARPS

The ARPS model is a three-dimensional, non hydrostatic code designed for the prediction of small scale, short duration events like thunderstorms, snow bands, and downslope windstorms. The numerical model developed by CAPS has been designed to predict the above events with the following general prediction goals:

- Mesoscale Phenomena: Prediction duration ranging from 0 to 12 hours. Location of events to within 50 km and timing of events to within 1 hour.
- Stormscale Phenomena: Prediction duration ranging from 0 to 6 hours. Location of events to within 10 km and timing of events to within 15 min.
- Microscale Phenomena: Prediction duration ranging from 0 to 1 hours. Location of events to within 1 km and timing of events to within 5 min.

The developed model included governing equations for momentum, heat, mass, water substances, turbulent kinetic energy, and the equation of state. The model was developed at CAPS with three main goals: sufficient adaptability to new data assimilation strategies, ease of use, and suitability for variety of computing platforms. Moreover, the model was designed to be suitable for scalable parallel processors, which makes it a good target for the scalability study. Because the code is huge (more than 280 subroutines distributed in 30 files) and was developed by more than 30 scientific and support personnel over the last six years, the approaches mentioned in Chapter 2 are difficult to apply if not impossible. The approach described in this dissertation is the best economical way to test scalability of this code. The ARPS code and documentation are available via the anonymous FTP site "ftpcaps.ou.edu/pub/ARPS" or the CAPS home page "wwwcaps.ou.edu".

A.2 The ARPS Code Structure

The system consists of three stages, the main ARPS model stage plus preprocessing and postprocessing stages. The main ARPS model stage has fife major functions: INITIAL, EXTBDTINI, INIOUT, CORDINTG, OUTPUT, and CHKSTAB. The model has variety of initialization options. The simplest initialization does not require any external data except a single sounding. We used the "may20.snd" sounding file for our experiments. However, for applications that require real terrain and surface data, preprocessors are needed to extract these data from data bases. The output of the ARPS model is in the form of history data written into one or more files. The history

data can be later accessed by postprocessor for graphical output, data conversion or formatted printing. In this study we concentrated on the main ARPS model.

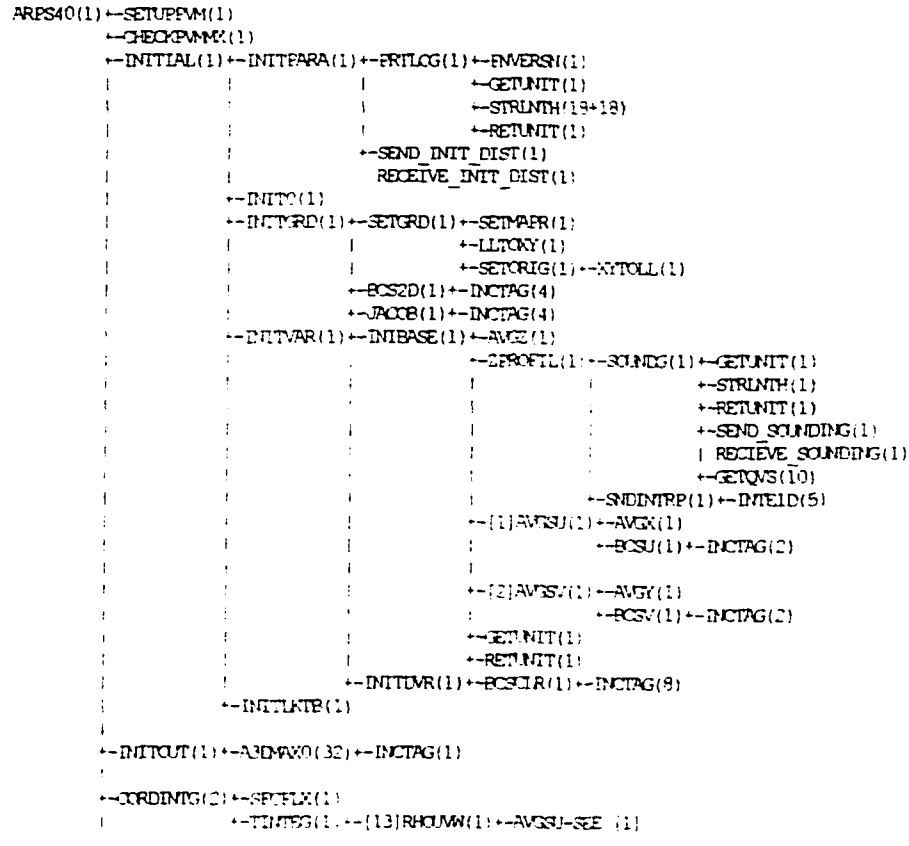
A.3 The Calling Tree

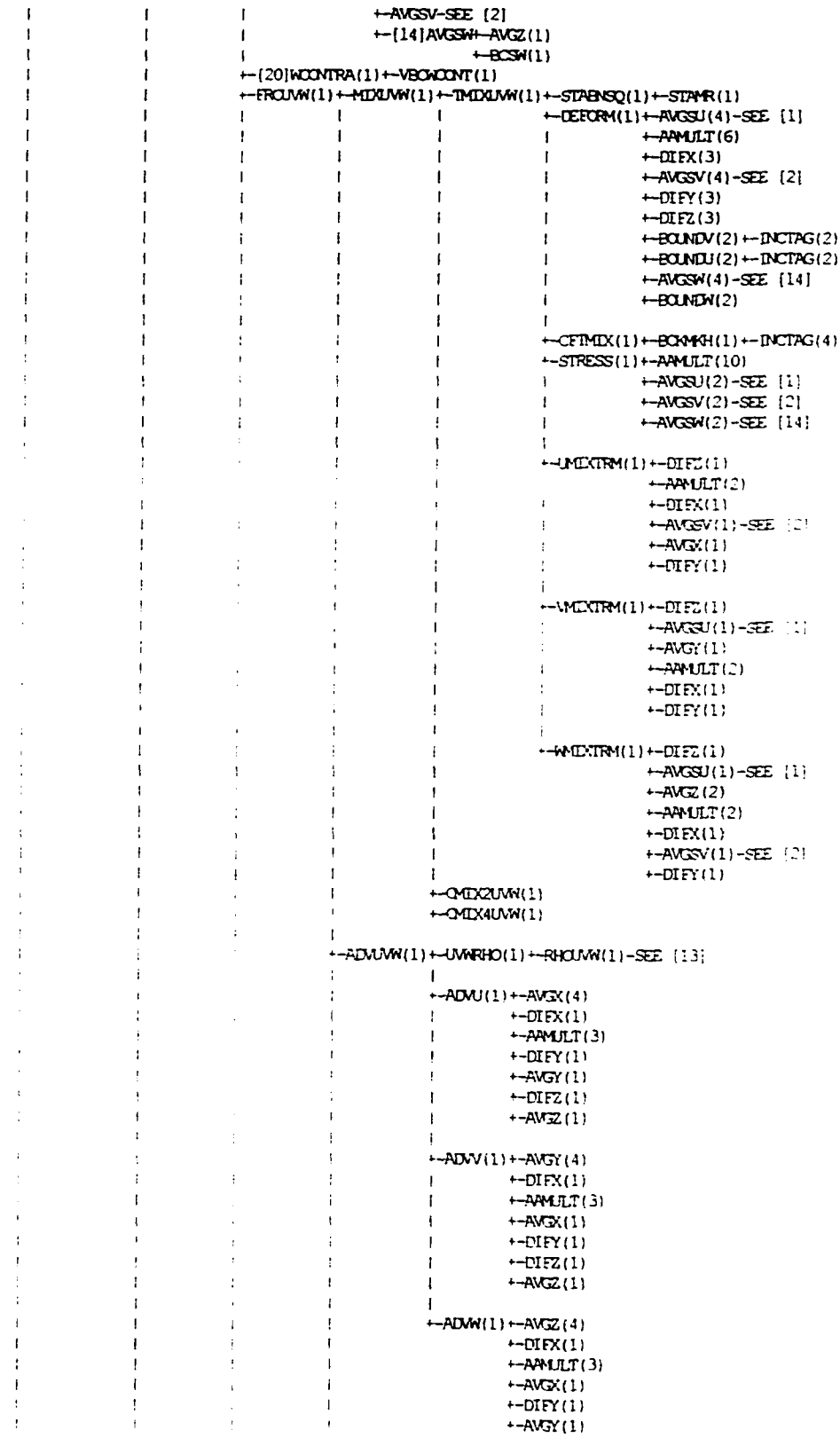
The following calling tree was constructed for the PVM version of parallel ARPS. The numbers show how many times each subroutine was called during two time steps of model run. We used:

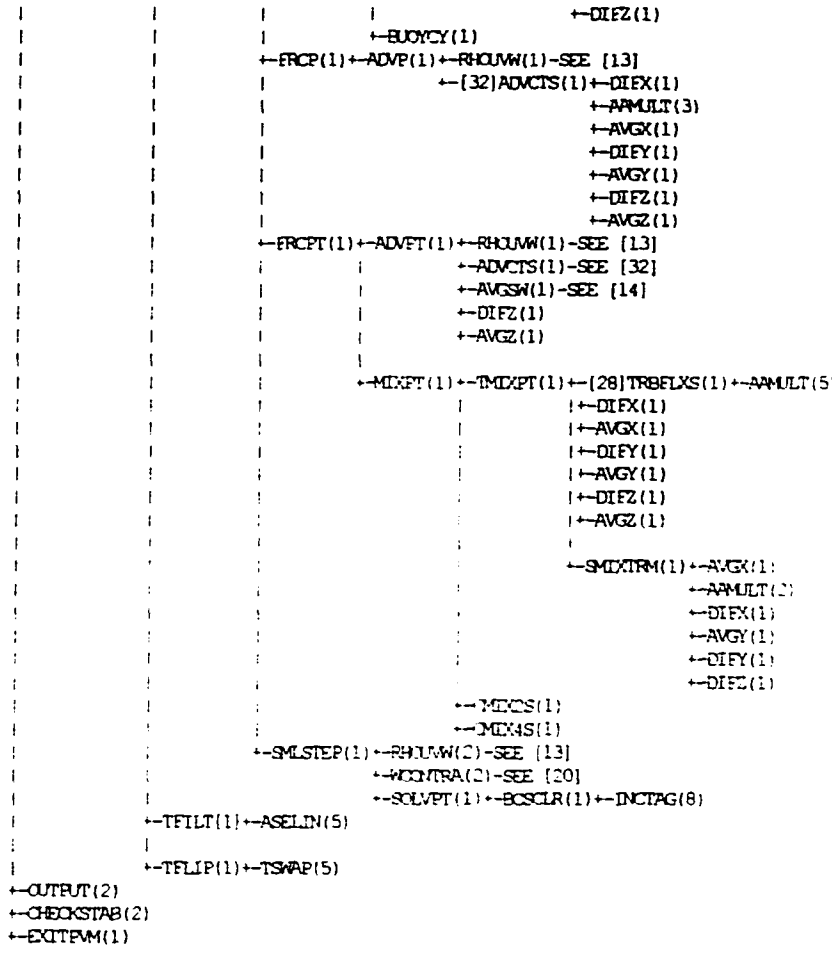
(*n*) the subroutine was called *n* times.

{} the subroutine was not called in the first step (see the file "tinteg3d.f").

[] the sub-tree number is used if the same sub-tree was called in another place.




+-AVGSV-SEE [2]
+-[14]AVGSW+-AVGZ(1)
+-BCSW(1)
+-[20]WCONTRA(1)+-VBOWCONT(1)
+-FROUVW(1)+-MIXUVW(1)+-TMIXUVW(1)+-STABNSQ(1)+-STAMR(1)
+-DEFORM(1)+-AVGSU(4)-SEE [1]
+-AAMULT(6)
+-DIFX(3)
+-AVGSV(4)-SEE [2]
+-DIFY(3)
+-DIFZ(3)
+-BOUNDV(2)+-INCTAG(2)
+-BOUNDU(2)+-INCTAG(2)
+-AVGSW(4)-SEE [14]
+-BOUNDW(2)
+-CFTMIX(1)+-BCKMKH(1)+-INCTAG(4)
+-STRESS(1)+-AAMULT(10)
+-AVGSU(2)-SEE [1]
+-AVGSV(2)-SEE [2]
+-AVGSW(2)-SEE [14]
+-UMIXTRM(1)+-DIFZ(1)
+-AAMULT(2)
+-DIFX(1)
+-AVGSV(1)-SEE [2]
+-AVGX(1)
+-DIFY(1)
+-VMIXTRM(1)+-DIFZ(1)
+-AVGSU(1)-SEE [1]
+-AVGY(1)
+-AAMULT(2)
+-DIFX(1)
+-DIFY(1)
+-WMIXTRM(1)+-DIFZ(1)
+-AVGSU(1)-SEE [1]
+-AVGZ(2)
+-AAMULT(2)
+-DIFX(1)
+-AVGSV(1)-SEE [2]
+-DIFY(1)
+-QMIX2UVW(1)
+-QMIX4UVW(1)
+-ADVUVW(1)+-UVWRHO(1)+-RHOUVW(1)-SEE [13]
+-ADVU(1)+-AVGX(4)
+-DIFX(1)
+-AAMULT(3)
+-DIFY(1)
+-AVGY(1)
+-DIFZ(1)
+-AVGZ(1)
+-ADVV(1)+-AVGY(4)
+-DIFX(1)
+-AAMULT(3)
+-AVGX(1)
+-DIFY(1)
+-DIFZ(1)
+-AVGZ(1)
+-ADVW(1)+-AVGZ(4)
+-DIFX(1)
+-AAMULT(3)
+-AVGX(1)
+-DIFY(1)
+-AVGY(1)


+-DIFZ(1)
+-BUOYCY(1)
+-FRCP(1)+-ADVP(1)+-RHOUVW(1)-SEE [13]
+-[32]ADVCTS(1)+-DIFX(1)
+-AAMULT(3)
+-AVGX(1)
+-DIFY(1)
+-AVGY(1)
+-DIFZ(1)
+-AVGZ(1)
+-FRCPT(1)+-ADVPT(1)+-RHOUVW(1)-SEE [13]
+-ADVCTS(1)-SEE [32]
+-AVGSW(1)-SEE [14]
+-DIFZ(1)
+-AVGZ(1)
+-MIXPT(1)+-TMIXPT(1)+-[28]TRBFLXS(1)+-AAMULT(5)
+-DIFX(1)
+-AVGX(1)
+-DIFY(1)
+-AVGY(1)
+-DIFZ(1)
+-AVGZ(1)
+-SMIXTRM(1)+-AVGX(1)
+-AAMULT(2)
+-DIFX(1)
+-AVGY(1)
+-DIFY(1)
+-DIFZ(1)
+-MIXCS(1)
+-MIX4S(1)
+-SMLSTEP(1)+-RHOUVW(2)-SEE [13]
+-WCONTRA(2)-SEE [20]
+-SOLVPT(1)+-BCSCLR(1)+-INCTAG(8)
+-TFILT(1)+-ASELIN(5)
+-TFLIP(1)+-TSWAP(5)
+-OUTPUT(2)
+-CHECKSTAB(2)
+-EXITPVM(1)

A.4 The Configuration Parameters

ARPS provides the user with flexible control over many of the configuration parameters. Some of the parameters are set at run time while other parameters are set before compilation. The program will read the run time parameters from an input file in NAMELIST format. We used the same default input parameters in the file "arps40.input" provided with the code except for the grid size and the model time which are set depending on the experiment. The change of grid size requires recompiling the code, while the change of model time does not. We added our parameters that represent the factors to the input file in the NAMELIST format. The number of processors was decided at run time for the CRAY J90 by sitting the environment variable NCPUS. On the other hand, the number of processors was predetermined before compilation on the IBM SP2 because it uses the message passing version.

APPENDIX B

THIRTEEN FACTOR FRACTIONAL FACTORIAL DESIGN

The following table is the fractional factorial design we used in Chapters 4 and 5. Since the runs were performed in random, the treatment number is used for identification only and does not represent the order in which the experiments were actually performed.

Treatment Number	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}
1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
2	-1	-1	-1	-1	1	1	1	1	1	1	1	1	-1
3	-1	-1	-1	1	-1	1	1	1	1	-1	-1	-1	1
4	-1	-1	-1	1	1	-1	-1	-1	-1	1	1	1	1
5	-1	-1	1	-1	-1	1	1	-1	-1	1	1	-1	1
6	-1	-1	1	-1	1	-1	-1	1	1	-1	-1	1	1
7	-1	-1	1	1	-1	-1	-1	1	1	1	1	-1	-1
8	-1	-1	1	1	1	1	1	-1	-1	-1	-1	1	-1
9	-1	1	-1	-1	-1	1	-1	1	-1	1	-1	1	1
10	-1	1	-1	-1	1	-1	1	-1	1	-1	1	-1	1
11	-1	1	-1	1	-1	-1	1	-1	1	1	-1	1	-1
12	-1	1	-1	1	1	1	-1	1	-1	-1	1	-1	-1
13	-1	1	1	-1	-1	-1	1	1	-1	-1	1	1	-1
14	-1	1	1	-1	1	1	-1	-1	1	1	-1	-1	-1
15	-1	1	1	1	-1	1	-1	-1	1	-1	1	1	1
16	-1	1	1	1	1	-1	1	1	-1	1	-1	-1	1
17	1	-1	-1	-1	-1	1	-1	-1	1	-1	1	1	-1
18	1	-1	-1	-1	1	-1	1	1	-1	1	-1	-1	-1
19	1	-1	-1	1	-1	-1	1	1	-1	-1	1	1	1
20	1	-1	-1	1	1	1	-1	-1	1	1	-1	-1	1
21	1	-1	1	-1	-1	-1	1	-1	1	1	-1	1	1
22	1	-1	1	-1	1	1	-1	1	-1	-1	1	-1	1
23	1	-1	1	1	-1	1	-1	1	-1	1	-1	1	-1
24	1	-1	1	1	1	-1	1	-1	1	-1	1	-1	-1
25	1	1	-1	-1	-1	-1	-1	1	1	1	1	-1	1
26	1	1	-1	-1	1	1	1	-1	-1	-1	-1	1	1
27	1	1	-1	1	-1	1	1	-1	-1	1	1	-1	-1
28	1	1	-1	1	1	-1	-1	1	1	-1	-1	1	-1
29	1	1	1	-1	-1	1	1	1	1	-1	-1	-1	-1
30	1	1	1	-1	1	-1	-1	-1	-1	1	1	1	-1
31	1	1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	1
32	1	1	1	1	1	1	1	1	1	1	1	1	1

The above design was obtained using the "FACTEX" procedure available with the SAS package. This procedure will generate orthogonally confounded designs including factorial and fractional factorial designs. After constructing a design, it can be printed, saved to file, or used as SAS data set. The factor confounding rules for the above design are as follows:

$$F_5 = F_0*F_1*F_2*F_3*F_4$$

$$F_6 = F_2*F_3*F_4$$

$$F_7 = F_1*F_3*F_4$$

$$F_8 = F_0*F_3*F_4$$

$$F_9 = F_1*F_2*F_4$$

$$F_{10} = F_0*F_2*F_4$$

$$F_{11} = F_0*F_1*F_4$$

$$F_{12} = F_1*F_2*F_3$$

with the following aliasing structure:

$$F_0$$

$$F_1$$

$$F_2$$

$$F_3$$

$$F_4$$

$$F_5$$

$$F_6$$

$$F_7$$

$$F_8$$

$$F_9$$

$$F_{10}$$

$$F_{11}$$

$$F_{12}$$

$$F_0*F_1 = F_4*F_{11} = F_5*F_6 = F_7*F_8 = F_9*F_{10}$$

$$F_0*F_2 = F_4*F_{10} = F_5*F_7 = F_6*F_8 = F_9*F_{11}$$

$$F_0*F_3 = F_4*F_8 = F_5*F_9 = F_6*F_{10} = F_7*F_{11}$$

$$F_0*F_4 = F_1*F_{11} = F_2*F_{10} = F_3*F_8 = F_5*F_{12}$$

$$F_0*F_5 = F_1*F_6 = F_2*F_7 = F_3*F_9 = F_4*F_{12}$$

$$F_0*F_6 = F_1*F_5 = F_2*F_8 = F_3*F_{10} = F_{11}*F_{12}$$

$$F_0*F_7 = F_1*F_8 = F_2*F_5 = F_3*F_{11} = F_{10}*F_{12}$$

$$F_0*F_8 = F_1*F_7 = F_2*F_6 = F_3*F_4 = F_9*F_{12}$$

$$F_0*F_9 = F_1*F_{10} = F_2*F_{11} = F_3*F_5 = F_8*F_{12}$$

$$F_0*F_{10} = F_1*F_9 = F_2*F_4 = F_3*F_6 = F_7*F_{12}$$

$$F_0*F_{11} = F_1*F_4 = F_2*F_9 = F_3*F_7 = F_6*F_{12}$$

$$F_0*F_{12} = F_4*F_5 = F_6*F_{11} = F_7*F_{10} = F_8*F_9$$

$$F_1*F_2 = F_3*F_{12} = F_4*F_9 = F_5*F_8 = F_6*F_7 = F_{10}*F_{11}$$

$$F_1*F_3 = F_2*F_{12} = F_4*F_7 = F_5*F_{10} = F_6*F_9 = F_8*F_{11}$$

$$F_1*F_{12} = F_2*F_3 = F_4*F_6 = F_5*F_{11} = F_7*F_9 = F_8*F_{10}$$