2007

Investigation of Downey model for speedup prediction

Lin Lan
University of Windsor

Investigation of Downey Model for Speedup Prediction

by

Lin Lan

A Thesis

Submitted to the Faculty of Graduate Studies

through the School of Computer Science

in Partial Fulfillment of the Requirements for the Degree of Master of Science

at the University of Windsor

Windsor, Ontario, Canada

2007

Your file Votre référence
ISBN: 978-0-494-34972-4
Our file Notre référence
ISBN: 978-0-494-34972-4

Abstract

In parallel computing, accurate prediction of speedup is important for job schedulers with adaptive resource allocation. The predicted speedup determines the expected runtime on a certain number of nodes and the efficiency by which the resources are used. Among the existing speedup prediction models, the Downey model [5, 6] is simple but promising. However, the prediction accuracy of the Downey model needs to be investigated in realistic scenario setups. In this thesis, we use the NAS benchmarks and synthetic benchmarks [19] to generate scenarios in which the performance of the Downey model is examined. Based on these experiments, conditions are suggested for the successful application of the Downey model.

Dedication

To

My parents for their endless love and encouragement

Acknowledgements

First of all, I really appreciate the great help of my supervisor Dr. Angela C. Sodan during my thesis work. Without her constant encouragement and valuable guidance, it would not possible for me to finish this work.

I would also like to thank my thesis committee members, Dr. Karen Y. Fung, Dr. Jianguo Lu, and Dr. Dan Wu for their time, effort, and helpful comments and suggestions.

I am grateful to my parents, my sister, my brother in law, and my uncle. They give me great confidence and solid support whenever I met any kind of problems.

I want to acknowledge my colleagues, Xijie Zeng, Arun K. Kanavallil, Xiaorong Cao, Peiyu Cai, Jiaying Shi, and Wei Jin for helping on my thesis work.

Finally, I thank all of my friends for their endless help, support, and encouragement.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In recent years, computer clusters play a more and more important role in high performance computing. Basically, a computer cluster can be viewed as a supercomputer consisted of multiple nodes. A node in a supercomputer may have one or more CPUs, it also includes memory modules, maybe disks and is usually capable of communicating with other nodes and possible upper-level controllers. As a result, multiple nodes in a computer cluster can operate simultaneously and cooperate with each other. This makes it possible to process a job using multiple nodes instead of using a single node. Compared with using a single node, the much stronger processing power of multiple nodes should lead to a much shorter processing time for the same job.

In a computer cluster, a job scheduler uses a scheduling algorithm to manage the cluster resources and assign them to jobs. A good scheduling algorithm should achieve high efficiency for the computer cluster, it should also minimize the waiting time for incoming jobs. While jobs compete with each other for priority and more resources so that they can be processed as quickly as possible, the job scheduler has to balance among all the jobs so that a global optimal scheduling should be achieved. However, this good scheduling requires accurate information on the processing time

versus resource allocation choices for each job. In other words, the scheduler needs to know how fast a job can be processed using different numbers of nodes. The availability of this information enables the use of advanced job scheduling methods, such as flexible time sharing and adaptive resource allocation [17].

In the literature, the information on the processing time versus resource allocation choices for a job is described in the concept of speedup. Speedup is defined as the ratio of job processing time using a single node to the processing time using n nodes. It indicates how much faster a job can be done using n nodes compared to that using one node. Ideally, when a job is processed by n nodes, the processing time should be 1/n compared with using a single node. In reality, this relationship does not hold. Nodes participating in the processing of a common job need to communicate with each other for synchronization, data exchange, scheduling, and so on. Hardware factors such as memory hierarchy bandwidth and latency, job instruction mix structure, communication frequency, bandwidth and serialization all have influence on these additional processing overheads [13]. As a result, accurate computation of speedup is very difficult and often impossible. Instead, in the literature models are used to predict speedup. Examples of speedup prediction models include Dowdy's Model [4] and its two modified versions proposed by Chiang et al. [3] and Brecht and Guha [2, 9], the Downey Model [5], the model proposed by Smirni et al. [16] and the recent model proposed by Lafreniere et al. [10]. The model built by Lafreniere et al. uses the knowledge of the application structure which is different from others. All of these models use parameters to summarize characteristics of the processing system. Records on executed jobs in the past are used to tune the models. Well-tuned models are then used to predict speedup for future jobs.

Among the aforementioned speedup prediction models, the Downey model

is simple, promising, and its parameters have physical meanings and are easy to obtain from existing job execution data. However, the applicability of the Downey model to the speedup prediction problem under realistic system scenarios needs to be investigated in detail.

In this thesis, we use the NAS parallel benchmarks (NPB) [1] and synthetic benchmarks developed by our research group to generate scenarios in which the Downey model is applied for speedup prediction. The NAS parallel benchmarks [1] are a famous suite of benchmarks in the high performance computing community. It is developed by the NASA Advanced Supercomputing (NAS) Division. The benchmarks have been used by a number of authors to evaluate computing system performances [5, 10]. Our synthetic benchmarks are another suit of parallel benchmarks [19]. Compared with the well-established NAS parallel benchmarks, our synthetic benchmarks can measure the communication time as well as the running time of the system, while most of the NAS parallel benchmarks overlap the communication.

In our experiments, the two benchmarks are applied on the SHARCNET (Shared Hierarchical Academic Research Computing Network) [15] which links a number of high performance clusters built for universities and colleges in Canada. For each generated scenario, we applied the Downey model using the Levenberg-Marquardt (LM) algorithm [12] to estimate parameters. The model is then used for the prediction of speedup for desired sub-cluster sizes.

From the experimental results obtained through the aforementioned approach, we find that the predicted speedup values by the Downey model match with the speedup measurements generated by the benchmarks. This validates the application of the Downey model in practice. However, to achieve a better speedup prediction, the Downey model should be provided with speedup measurements with at least three

measurements, which should correspond to a small n in the linear part, a large n close to the peak speedup point, and a n in the transition section between the linear and nonlinear parts.

The thesis is organized as follows: Chapter 2 explains the importance of speedup prediction and motivates the thesis; Chapter 3 provides a brief literature summary on existing speedup prediction models and explains the Downey model in detail; in Chapter 4 we introduce the two benchmarks (the NPB and synthetic benchmarks) used in our experiments; and introduce the implementation of the Downey model in Chapter 5; the test plan and test results are provided and analyzed in Chapter 6; and finally, Chapter 7 concludes the whole thesis.

Chapter 2

Motivation

In modern parallel job scheduling for a processing system with multiple processing units, a job can be allocated with a number of available processing units. Compared with single processing unit allocation, multiple processing unit allocation can better utilize the processing capacity of the system and speed up the processing of a job. These allocated units work on the same job simultaneously, usually communicate with each other during the process. If we call each processing unit a node, then the whole processing system can be viewed as a cluster of nodes. Consequently, the allocated units for a particular job form a sub-cluster in the system.

When multiple jobs need to be processed, optimal scheduling becomes a key element to achieve high efficiency of the system. However, optimal scheduling needs the basic information on the relationship between job processing time and available sub-cluster sizes. This relationship is summarized in the concept of the so-called speedup, which is defined as the ratio of job processing time on a single node to the processing time using n node. Speedup (defined in Section 1) indicates how much faster a job can be done using n nodes compared to that using one node.

In traditional job scheduling, accurate information on speedup for various jobs has been shown to be helpful on reducing average job response time and improving

performance of scheduling algorithms. In modern job scheduling, the speedup information is important for new job scheduling approaches including flexible time sharing and adaptive resource allocation [17]. In flexible time sharing, if jobs can be well matched, global job control can be abandoned, or global synchronous gang scheduling can be relaxed [18, 8]. In adaptive resource allocation, sub-cluster size can be adaptively adjusted during the processing of a job. In both approaches, information on job processing time versus sub-cluster size is crucial for the scheduling decisions.

Under ideal conditions, when a job is processed by a sub-cluster (with n nodes), each node performs 1/n of the total processing steps. Here it is assumed that there is no additional processing step, i.e., all the performed processing steps by each node in the sub-cluster are necessary for the job itself. Obviously, the speedup equals to n. This value actually represents the theoretical upper bound of speedup.

However, parallel job processing in the real world usually includes additional processing steps, communication among nodes in the sub-cluster, synchronization among nodes, and so on. These processing steps are necessary for a job to be successfully processed by a sub-cluster, but are not performed otherwise if the job is handled by a single node. Moreover, it is observed from practical applications that the processing overhead incurred by these additional processing steps increase with the number of involved nodes (n), and as a result the ratio of speedup to number of nodes decreases as n becomes bigger. Eventually, the speedup reaches a maximum value at some specific n_{max} value. If more than n_{max} nodes are allocated to the job, the overhead becomes too big and the speedup value decreases as n increases. These phenomenon are clearly shown in the following example.

Example 2.1

In this example we use NPB (NASA Advanced Supercomputing Division Par-

allel Benchmarks) to generate speedup values on SHARCNET [15]. SHARCNET (Shared Hierarchical Academic Research Computing Network) links a number of high performance clusters built for universities and colleges in Canada. The detailed introduction to NPB and SHARCNET are provided in Section 4.1 and 6.1 respectively. The NPB benchmarks are a suit of eight benchmarks. Here we use the SP (Pentadiagonal Solver) benchmarks to obtain the speedup values on SHARCNET. In our experiment, the SP benchmarks generate speedup values for number of nodes $n = k^2$ where k is a positive integer. Figure 2.1 shows the speedup versus n curve for the range $k \in [1, 12]$. We also show the theoretical speedup upperbound in the figure for comparison purposes. From the figure, we have the following observations:

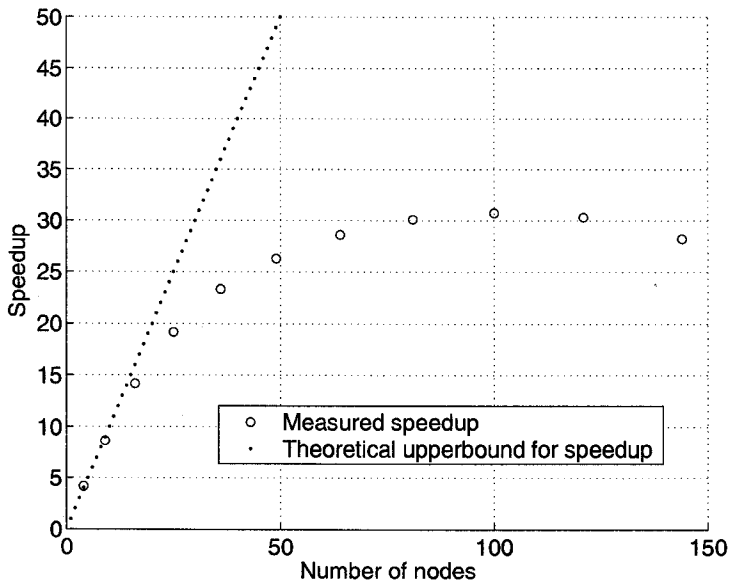

Figure 2.1: Speedup curve

1. When n is small, for example, n=4, 9, 16, the speedup value is very close to the theoretical upperbound. This is mainly because that when n is small, the corresponding overhead caused by the inter-operation between nodes is also small.

2. As n becomes bigger (n=25, 36, 49,), the speedup value keeps increasing, but

the ratio of speedup to n decreases for bigger n. This phenomenon corresponds to the fact that as more nodes are involved in the job, communication overhead is needed for each node.

3. The speedup reaches its maximum value (30.75) at n=100. For n values bigger than 100, the speedup values decreases as n increases. Obviously, in this case adding more nodes for the job results in too much more processing overhead and does not have any benefit on speedup any more

In fact, the curve shown in Figure 2.1 is very typical for speedup. Specifically, a typical speedup curve is consisted of three parts, namely the linear part, the nonlinear part, and the decline part. With n increasing from 1 to a very large number, the speedup travels along the speedup curve, passing in turn through the linear part, then the nonlinear part, and finally the decline part. The linear part of the speedup curve corresponds to small n values. As discussed in the above example, the speedup curve in this part increases almost linearly with n. In comparison, in the nonlinear part of the curve, the speedup increases nonlinearly with n till reaching the maximum speedup point. Finally, the decline part of the speedup curve represents the part in which the speedup decreases with n, this happens when too many nodes are assigned to a job and the overhead is so high that increasing n has a negative effect on speedup.

In order to obtain accurate speedup values for a given number of nodes allocated for a job, we must know information on the processing overhead. However, in practice, the processing overhead is influenced by many factors. Memory hierarchy bandwidth and latency, job instruction mix structure, communication frequency, bandwidth and serialization are some examples which have influence on the processing overhead [13]. As a result, accurate computation of speedup is very difficult and often impossible. Instead, in the literature, models are used to predict speedup. These

models use parameters to describe the algorithms. Records on executed jobs in the past are used for the tuning of the models. Well-tuned models are then applied on future jobs to predict speedup for different sub-cluster sizes.

Among the existing speedup prediction models, Dowdy's model [4] is the earliest and simplest model, it has two variations proposed by Chiang et al. [3] and Brecht and Guha [2, 9]. Smirni et al. proposed another model [16] to facilitate speedup analysis. Parameters in these models do not have physical meanings and are therefore difficult to determine from observation data. Recently, a so-called ScoPred model is proposed [10]. The model requires both historical running information and the users' own knowledge of his or her parallel applications.

Compared with the aforementioned speedup prediction models, the model proposed by Downey [5, 6] is simple, accurate, and can be used to interpolate between existing speedup measurements. The Downey model uses two parameters to summarize the processing system: the so-called average parallelism of the program (denoted by A), and the variance in parallelism (denoted by V). The two parameters have physical meanings and are easy to determine from historical application execution data. In addition, the model does not require any information based on users' experience. However, although Downey illustrated the application to speedup prediction in his papers [5, 6], the prediction performance of the model in realistic application scenarios is not thoroughly investigated. For successful application of the Downey model to real world systems, the model needs to be tested under typical scenarios, its prediction performances need to be analyzed, and guidelines need to be proposed.

In this thesis, the Downey model is applied to a standard job-execution system running primarily MPI based applications. Furthermore, we use two benchmarks (the NAS parallel benchmarks and our own synthetic benchmarks) to generate simulated

testing scenarios. The Downey model is applied to these scenarios and its performance is evaluated and analyzed. Based on the experimental results, we then propose some suggestions on the application of the Downey model.

Chapter 3

Speedup prediction

3.1 Some existing speedup models

The simplest speedup prediction model is proposed by Dowdy [4]. This model is based on Amdahl's law. In this model, the job execution time on a sub-cluster with n nodes T(n) is modeled as

$$T(n) = c1 + c2/n, \tag{3.1}$$

where c1 and c2 are model coefficients. Specifically, c1 is called the sequential component, c2 is called the parallel component. The above model can be understood as follows. In parallel computing, the processing executed at each node can be separated into two parts: processing for the job itself, and additional processing for synchronization, message exchange with other nodes, scheduling and so on. Execution time for the former part decreases when more nodes are allocated for the job, in other words, the corresponding processing time is reversely related with n. This time is represented by the second term in Eq. (3.1). In comparison, the first term c1 represents the processing overhead in the parallel processing of the job. The corresponding execution time for this part is believed to be constant for different sub-cluster sizes.

It is represented by the first term in Eq. (3.1).

Based on the same rational, Chiang proposed the following speedup model [3]:

$$S(n) = (1 + \beta)n/(n + \beta). \tag{3.2}$$

Here, $\beta \geq 0$ represents a program characteristic indicating whether the program can be efficiently executed in parallel. Specifically, for a sequential program job, β equals to 0 and S(n)=1. In other words, adding more nodes to the job does not lead to any saving in the execution time. This is because the program cannot be executed on more than one node in parallel. In comparison, for a program perfectly suited for parallel execution, β approaches infinity, and S(n) approaches n. Note that this is the case of perfect parallel processing in which the theoretical upper bound for speedup is reached.

Perhaps the biggest problem with the above models is that the parameters do not have physical meanings and are difficult to determine from speedup measurements (or observations in short) obtained from practical applications. This reduces the accuracy of speedup prediction in practice.

Another example that lack physical meaning for its parameter is the model proposed in [16]. The running time for this model is

$$S(n) = (1 - r^n)/(1 - r). \tag{3.3}$$

Here, $0 \leq r \leq 1$. The model was designed to facilitate analysis. Again, the parameter r has no real meaning.

Compared with the above models, the model proposed by Downey [5] uses parameters which have concrete physical meanings. The model uses two parameters, the so-called average parallelism of the program, and the variance in parallelism. The parameters can be evaluated using historic speedup data. The Downey model has two sub-models to characterize speedup more accurately. Moreover, it matches the

theoretical speedup bounds. For detailed introduction on the Downey model, please refer to Section 3.2.

Recently, a new speedup prediction method is proposed [10]. It is called ScoPred. ScoPred uses two kinds of information: historical running information and the users' own knowledge of his or her parallel application. ScoPred uses multiple linear regression technique for the prediction of runtime on different number of nodes and for different problem sizes. The prediction result includes mean values, confidence, and prediction intervals. Once well-tuned, its scalable prediction can be very accurate. However, though very helpful for speedup prediction, the users' knowledge may be difficult to obtain.

3.2 The Downey model

The Downey model was proposed by Allen B. Downey [5, 6]. The model is based on two parameters: the so-called average parallelism of the program (denoted by A), and the variance in parallelism (denoted by V). The model aims to find the speedup curve corresponding to the A and V values. As introduced in previous sections, the speedup is defined as the ratio of job running time on a single node (denoted by T(1)) to the running time using n nodes (denoted by T(n)), it indicates how much faster a job can be done using n nodes compared to using one node. Correspondingly, the speedup curve is defined as the T(1)/T(n) vs. n curve. Figure 3.1 and 3.2 show some typical speedup curves predicted using the Downey model.

The parameters A and V describe the basic characteristics of a job. Basically, the average parallelism of the program A is a measure of the maximum speedup achievable for a job. A larger value of A corresponds to a larger speedup a job can achieve in a parallel processing system. In comparison, the variance in parallelism

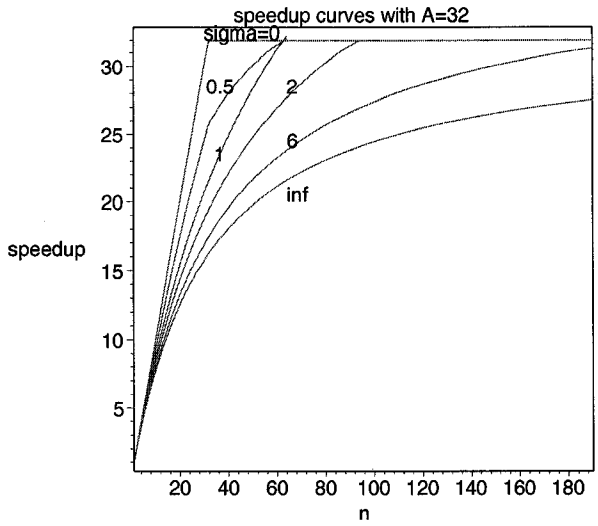

Figure 3.1: Speedup curves with $A = 32$

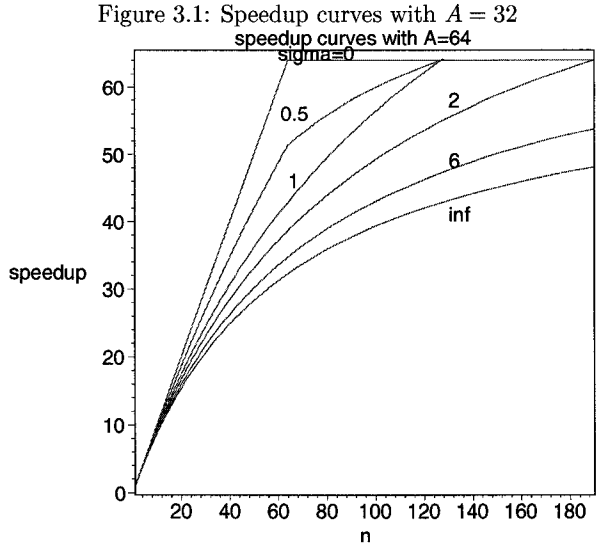

Figure 3.2: Speedup curves with $A = 64$

V indicates the closeness to linearity for the speedup curve. V=0 corresponds to a linear speedup curve, and a larger V corresponds to a greater deviation from the linear case. Given the values of A and V, the so-called coefficient of variation CV can be given as $CV = \sqrt{V}/A$. Downey also defined σ as an approximation of CV, σ is related to A and V as $V = \sigma(A-1)^2$. From the above, we get $CV^2 = V/A^2$. If substituting $V = \sigma(A-1)^2$ into this formula in the right-hand side, we obtain

$CV^2 = \sigma(A-1)^2/A^2$. If A is large enough, CV^2 can be approximated by σ. For many applications, σ is in the range between 0 and 2 [6]. Obviously, the Downey model is determined by A and one of the three parameters V, CV and σ. In the rest of the thesis, we use A and σ to characterize the Downey model.

The Downey model is divided into two sub-models, the so-called Low variance model and the High variance model. The two sub-models correspond to different ranges of σ , where σ is the approximation of the coefficient of variance in parallelism. The Low variance model has $\sigma \leq 1$ while the High variance model has $\sigma \geq 1$. In the following sections, we introduce the two sub-models in detail.

3.2.1 Low variance model

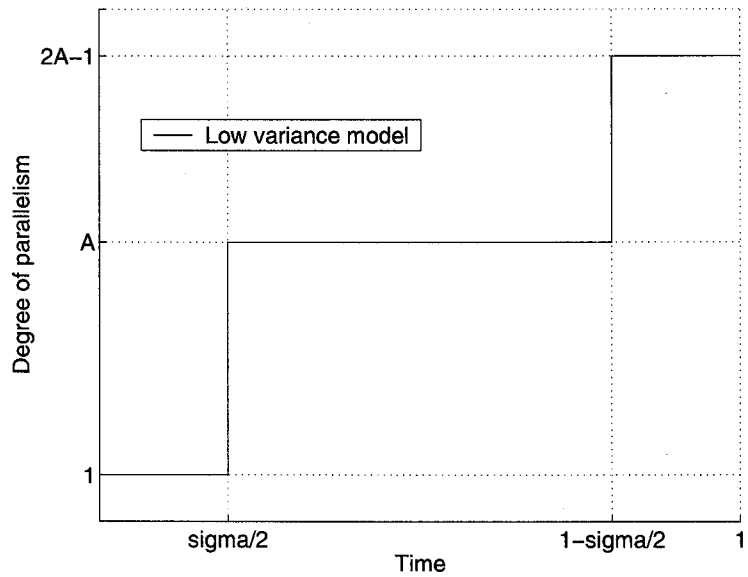

Figure 3.3: The parallelism profile for the low variance speedup model (sigma=σ)

In this sub-model, the total job time is divided into three sections: for a period of $\sigma/2$, the degree of parallelism is 1, for a second period of $\sigma/2$, the degree of parallelism is 2A-1, and for the rest of the job time $1-\sigma$, the parallelism is A. Figure 3.3 shows the above three time periods and their corresponding degree of parallelism

values. Under this hypothetical parallelism profile, the run time T(n) can be written as (3.4):

$$T(n) = \begin{cases} (A - \sigma/2)/n + \sigma/2, & 1 \leq n \leq A, \\ \sigma(A - 1/2)/n + 1 - \sigma/2, & A \leq n \leq 2A - 1, \\ 1, & n \geq 2A - 1. \end{cases} \tag{3.4}$$

Here, n is the number of participating nodes. Note that T(1)=A and $T(\infty) = 1$. Therefore T(n) is actually the normalized run time for sub-cluster size n. The speedup S(n)=T(1)/T(n) can be expressed as

$$S(n) = \begin{cases} An/(A + \sigma/2(n-1)), & 1 \leq n \leq A, \\ An/(\sigma(A - 1/2) + n(1 - \sigma/2)), & A \leq n \leq 2A - 1, \\ A, & n \geq 2A - 1. \end{cases} \tag{3.5}$$

3.2.2 High variance model

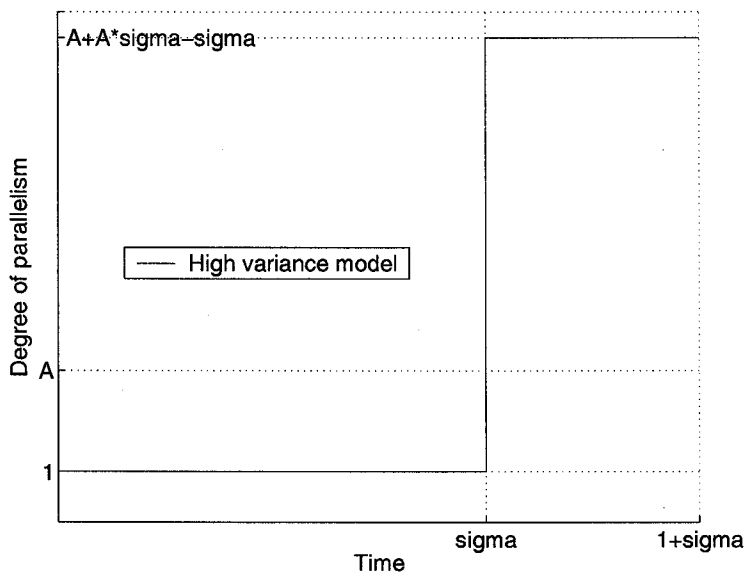

Figure 3.4: The parallelism profile for the high vaariance speedup model (sigma=σ)

The High variance model corresponds to large σ values $\sigma \geq 1$. The total job time is divided into two sections: a period of with degree of parallelism equal to 1, and a period of 1 with degree of parallelism value $A + A\sigma - \sigma$. This hypothetical parallelism profile is shown in Figure 3.4. As a result, the run time T(n) is

$$T(n) = \begin{cases} \sigma + (A + A\sigma - \sigma)/n, & 1 \leq n \leq A + A\sigma - \sigma, \\ \sigma + 1, & n \geq A + A\sigma - \sigma. \end{cases} \tag{3.6}$$

Obviously, $T(1) = A(\sigma + 1)$ and $T(\infty) = \sigma + 1$. Consequently, the speedup can be obtained as

$$S(n) = \begin{cases} nA(\sigma + 1)/(\sigma(n + A - 1) + A), & 1 \leq n \leq A + A\sigma - \sigma, \\ A, & n \geq A + A\sigma - \sigma. \end{cases} \tag{3.7}$$

It is easy to verify that when $\sigma = 1$, the S(n) values resulted from the two sub-models are identical.

A very favorable feature of the Downey model is that in the two extreme cases where $\sigma = 0$ and σ approaches infinity respectively, the speedup resulting from the model match with the corresponding theoretical bounds. Specifically, when $\sigma = 0$, the resulting speedup curve matches the theoretical upper bound for speedup. Here the curve is first bounded by the hardware limit (a 45 degree line), and after the curve reaches the value of A, it is bounded by the software limit (average parallelism A). In comparison, when σ approaches infinity, the speedup curve provided by the Downey model approaches the theoretical lower bound on speedup [7]:

$$S_{low}(n) = An/(A + n - 1). \tag{3.8}$$

In Figure 3.1 and 3.2, the two figures correspond with different A values (A=32 and A=64) respectively. In each figure, the speedup curves corresponding with different values are shown. Note that different σ values may result in the use of different

sub-models. However, when $\sigma = 1$, curves from the two sub-models are identical with each other. In addition, the figures clearly show the limiting cases of $\sigma = 0$ and $\sigma \to \infty$, when the curves match with theoretical bounds.

Chapter 4

NAS and synthetic benchmarks

In the literature, performance of clusters is evaluated through the use of so-called benchmarks. In our approach, we use two kinds of benchmarks. The first are the famous parallel NAS benchmarks [1]. They have been used by a number of authors to evaluate computing system performances [5, 10]. The second, the synthetic benchmarks, are another suit of parallel parallel benchmarks [19]. Compared with the well-established NAS parallel benchmarks, our synthetic benchmarks can measure the communication time as well as the running time of the system, while most of the NAS parallel benchmarks overlap the communication. In the following sections, we introduce the two benchmarks in detail.

4.1 NAS parallel benchmarks

The NAS parallel benchmarks, or NPB in short, are a suite of well-known benchmarks in the high performance computing community. The benchmarks are developed by the NASA Advanced Supercomputing (NAS) Division. They are designed to measure the performance of parallel supercomputers. From its first version

(NPB1), the suite of NAS parallel benchmarks have been updated several times; the most recent version is NPB3.2.1.

NPB 3.2.1 includes eight individual benchmarks: Embarrassingly Parallel (EP), Pentadiagonal Solver (SP), 3-D FFT PDE (FT), Block Tridiagonal Solver (BT), Multgrid (MG), LU Solver (LU), Conjugate Gradient (CG), and Integer Sort (IS). According to the specifications given by NAS, SP solves Navier-Stokes equations in 3-D by Gaussian elimination without pivoting and its resulting system is scalar pentadiagonal. BT solves Navier-Stokes equations using the Beam-Warming method. MG solves Poisson's equation using a V-cycle multigrid algorithm. LU solves Navier-Stokes equations in 3-D by LU decomposition and successive over-relaxation. FT solves a specified partial differential equation with FFTs. IS sorts N keys created by the sequential key generation algorithm in parallel. CG solves an unstructured sparse linear system with the conjugate gradient algorithm. EP creates pairs of Gaussian random deviates.

Each of the above benchmarks can be applied to clusters with different numbers of nodes (denoted by M). However, M should follow certain rules. Specifically, BT and SP can only be run with $M = K^2$, where $K \geq 1$ is an integer. IS, CG, MG, FT, and LU require that $M = 2^k$, with $K \geq 0$ as an integer. EP has no restriction on the value of M.

In addition, NPB 3.2.1 specifies six classes of problem sizes, namely, Class S, Class W, Class A, Class B, Class C, and Class D. The problem becomes bigger from Class S to Class C. These problem sizes (except Class D) can be applied to all the aforementioned benchmarks.

4.2 Synthetic benchmarks

Our own research group presented another suit of efficient benchmarks for performance measurement, it is called the synthetic benchmarks [19].

In MPI and other applications, a number of nodes in a cluster may be grouped together to perform a certain operation. Our synthetic benchmarks specified six different patterns based on the ways messages are distributed among the nodes in a sub-cluster. In the patterns, they include communication, computation and intialment.The six patterns in the synthetic benchmarks are:

- Master-Slave Pattern: a pre-defined master node sends and returns messages to/from all other nodes (slave nodes) in the sub-cluster.
- Stream Pattern: in this pattern, a message can only be transferred node-by-node organized as a pipe line.
- Nearest Neighbor Pattern: a node exchanges messages with its pre-defined neighboring nodes in the sub-cluster
- Random Pattern: a node sends a message to another randomly chosen node in the sub-cluster. Note that the receiving node is chosen randomly on a message-by-message base. In other words, messages from the same node may be sent to different nodes.
- Broadcast Pattern: A node sends the same messages to all other nodes.
- All-to-All Pattern: a node sends its messages to all other nodes in the sub-cluster. It also receives messages from all the other nodes.

Similar to the NAS parallel benchmarks, the synthetic benchmarks can also specify different problem sizes. The sizes are based on information such as computa-

tion assumed in the problem, number of loops assumed in computation, and size of messages.

Chapter 5

Implementation

In this chapter, we introduce some important implementation issues in our experiments. First of all, the two parameters A and σ in the Downey model are estimated using the Levenberg-Marquardt (LM) algorithm [12]. The LM algorithm is introduced in Section 5.1, its application to the estimation of parameters A and σ is given in Section 5.2. The last section of the chapter introduces implementation issues, including speedup measurement generation using the NAS parallel benchmarks and the synthetic benchmarks, parameter estimation for the Downey model, and speedup prediction by the Downey model.

5.1 Levenberg-Marquardt algorithm

In the implementation, the key step is to estimate parameters A and σ in the Downey model. Obviously, the accuracy on the estimation of these parameters directly influences the speedup prediction accuracy. Here we use the Levenberg-Marquardt (LM) algorithm for the estimation [12] already as proposed by Downey [5]. The LM algorithm is a popular method to solve nonlinear least-squares problem. It is an iterative algorithm to locate the minimum summation of squares of nonlinear

functions. The algorithm is a combination of the steepest descent method and the Gauss-Newton method. Specifically, when the interim solution in an iteration is far from the optimal solution, LM drives the solution towards the optimal point in a manner similar to the steepest descent method. However, when the interim solution approaches the optimal point, the LM algorithm converges in a similar way as the Gauss-Newton method [12].

In the following, we briefly introduce the LM algorithm. For details of the algorithm, please refer to [12].

Suppose we have a measured vector $\mathbf{x} \in \Re^M$, we would like to approximate $\mathbf{x}$ by an estimate vector $\hat{\mathbf{x}} \in \Re^M$ given by $\hat{\mathbf{x}} = f(\mathbf{p})$. Here $\mathbf{p} \in \Re^K$ is the parameter vector, and $f(\cdot)$ is the estimation function. Let $\mathbf{e} = \mathbf{x} - \hat{\mathbf{x}}$ represents the error between $\mathbf{x}$ and its estimate $\hat{\mathbf{x}}$, the LM algorithm aims to find the optimal parameter vector $\mathbf{p}^+$ which minimizes the square estimation error $\mathbf{e}^T\mathbf{e}$.

To achieve the above objective, the LM algorithm starts from a initial parameter vector $\mathbf{p}_0$, and iteratively update the parameter vector to converge to the optimal $\mathbf{p}^+$. Suppose that at the i'th iteration, the interim parameter vector is denoted by $\mathbf{p}_i$, the interim estimate is $\hat{\mathbf{x}}_i = f(\mathbf{p}_i)$, the estimation error is $\mathbf{e}_i = \mathbf{x} - \mathbf{x}_i$, and the Jacobian matrix of $f(\mathbf{p})$ is $\mathbf{J}_i = \frac{\partial f(\mathbf{p})}{\partial \mathbf{p}}|_{\mathbf{p}_i}$. Also suppose that we update the parameter vector to $\mathbf{p}_{i+1} = \mathbf{p}_i + \Delta_{\mathbf{p}_i}$. Then for a small enough value of $\|\Delta_{\mathbf{p}_i}\|$ ($\|\cdot\|$ denotes the 2-norm), we can approximate $f(\mathbf{p}_{i+1})$ by the first two terms of its Taylor series expansion, i.e.,

$$f(\mathbf{p}_{i+1}) \approx f(\mathbf{p}_i) + \mathbf{J}_i\Delta_{\mathbf{p}_i}. \tag{5.1}$$

Consequently, the squared estimation error for the i+1'th iteration can be approximated as

$$\mathbf{e}_{i+1}^T\mathbf{e}_{i+1} = \|\mathbf{x} - f(\mathbf{p}_i + \Delta_{\mathbf{p}_i})\|^2 \approx \|\mathbf{x} - f(\mathbf{p}_i) - \mathbf{J}_i\Delta_{\mathbf{p}_i}\|^2 = \|\mathbf{e}_i - \mathbf{J}_i\Delta_{\mathbf{p}_i}\|^2. \tag{5.2}$$

Minimization of $\|\mathbf{e}_i - \mathbf{J}_i\Delta_{\mathbf{p}_i}\|^2$ is the well-known least-squares problem, and the optimal $\Delta_{\mathbf{p}_i}$ satisfies the condition that $\mathbf{e}_i - \mathbf{J}\Delta_{\mathbf{p}_i}$ is orthogonal to $\mathbf{J}_i$. In other words, the optimal solution of $\Delta_{\mathbf{p}_i}$ can be obtained through the following normal equations:

$$\mathbf{J}_i^T\mathbf{J}_i\Delta_{\mathbf{p}_i} = \mathbf{J}_i^T\mathbf{e}_i. \tag{5.3}$$

Note that the $\Delta_{\mathbf{p}_i}$ solved from Eq. 5.3 only minimizes an approximation of the squared estimation error. Taking this into consideration, the LM algorithm solves the following slight variation of Eq. 5.3 instead:

$$\mathbf{N}_i\Delta_{\mathbf{p}_i} = \mathbf{J}_i^T\mathbf{e}_i, \tag{5.4}$$

which is called the augmented normal equations. Here, $\mathbf{N}_i \in \Re^{K\times K}$, and its elements are identical to the corresponding elements of $\mathbf{J}_i^T\mathbf{J}_i$, except the diagonal elements which are given by

$$[\mathbf{N}_i]_{jj} = \mu + [\mathbf{J}_i^T\mathbf{J}_i]_{jj}. \tag{5.5}$$

Here, $\mu > 0$ is called the damping term. When μ is large, $\mathbf{N}_i$ is close to diagonal, and the obtained $\Delta_{\mathbf{p}_i}$ is near the steepest descent direction. Note that a large μ also reduces the magnitude of $\Delta_{\mathbf{p}_i}$. In comparison, when μ is small, the solution of the augmented normal equations is close to that of Eq. 5.3.

In each iteration of the LM algorithm, the damping term is adjusted adaptively: a one dimensional optimization process is performed to determine the optimal value of μ which leads to the largest reduction of $\mathbf{e}^T\mathbf{e}$. The use of the damping term enables the LM algorithm to have the similar convergence behavior of both the steepest decent algorithm when the interim solution is far from $\mathbf{p}^+$, and the Gauss-Newton method when the interim solution is near $\mathbf{p}^+$.

5.2 Estimation of the Downey model parameters

The parameters A and σ in the Downey model are estimated using the LM algorithm introduced in the previous section. The implementation is taken from the source [11]. Here, we first obtain a number of speedup measurements using the two benchmarks introduced in the previous chapter, and then estimate A and σ through the minimization of the squared error between the predicted speedup by the Downey model and the measured values.

Suppose that from some benchmark we obtain M speedup measurements s_1, ..., s_M corresponding to sub-cluster sizes n_1, ..., n_M. In other words, the measurement vector $\mathbf{x} = [s_1 \quad s_2 \quad \cdots \quad s_M]^T$. Similarly, the estimated measurement vector $\hat{\mathbf{x}} = [s_1(n_1, A, \sigma) \quad s_2(n_2, A, \sigma) \quad \cdots \quad s_M(n_M, A, \sigma)]^T$, where $s_m(n_m, A, \sigma)$ is the estimated speedup by the Downey model. With n_m given, $s_m(n_m, A, \sigma)$ is a function of A and σ. The parameter vector is $\mathbf{p} = [A \quad \sigma]^T$. The Jacobian matrix $\mathbf{J} \in \Re^{M \times 2}$ is given by

$$\mathbf{J} = \begin{bmatrix} \frac{\partial s_1(n_1, A, \sigma)}{\partial A} & \frac{\partial s_1(n_1, A, \sigma)}{\partial \sigma} \\ \frac{\partial s_2(n_2, A, \sigma)}{\partial A} & \frac{\partial s_2(n_2, A, \sigma)}{\partial \sigma} \\ \cdots & \cdots \\ \frac{\partial s_M(n_M, A, \sigma)}{\partial A} & \frac{\partial s_M(n_M, A, \sigma)}{\partial \sigma} \end{bmatrix}. \tag{5.6}$$

By applying the above parameters to the LM algorithm, we can obtain the optimal A and σ which minimize the squared estimation error $\mathbf{e}^T\mathbf{e}$ where $\mathbf{e} = \mathbf{x} - \hat{\mathbf{x}}$.

5.3 Implementation issues

There are three important parts in our implementation: speedup measurement generation, parameter estimation for the Downey model, and speedup prediction using the optimally tuned Downey model.

In the speedup measurement generation part, we use the NAS parallel benchmarks (NPB) and the synthetic benchmarks. The NAS parallel benchmarks have a version number. In our implementation, we use the latest version NPB 3.2.1 obtained from the NAS website [14]. The benchmarks need a simple installation and compilation procedure for the MPI applications in our case. To generate the desired speedup measurements, we need to specify which benchmark to use (recall that the NPB are a suite of different benchmarks), the number of nodes in the sub-cluster, and the class name. The class name specifies one of the six problem sizes used in the specified benchmark. Note that the generated measurements are job run time values on specific sub-cluster sizes, speedup measurements are obtained by dividing $T(1)$ (the run time on single node) by the run time data.

Speedup measurement generation of the synthetic benchmarks are similar to the NAS parallel benchmarks. Here, we specify parameters such as problem size and inter-node communication pattern. A very important feature of the synthetic benchmarks is that it can obtain runtime measurements on computation and communication separately for the same job. This is achieved by first measuring the communication run time, and then obtaining the computation run time by subtracting the communication run time from the total run time. The communication run time can be obtained by blocking the computation part in the MPI code of the benchmark. Similar to NPB, the synthetic benchmarks only generate run time measurements, we need to perform a simple transformation to obtain speedup measurements.

The parameter estimation part uses the LM algorithm to estimate parameters A and σ for the Downey model. The input to the algorithm includes speedup measurements and initial values of the two parameters. Here we initiate the parameters as $A_0 = \max(s_1, \ldots, s_M)$ and $\sigma_0 = 0$. Among all the speedup measure-

ments, $\max(s_1, \ldots, s_M)$ is the closest to the theoretical speedup upperbound, i.e., $\max(s_1, \ldots, s_M)$ is the closest to the optimal value of A. This rationalizes the use of $\max(s_1, \ldots, s_M)$ as the initial value of A. We have mentioned that in Section 3.2 that σ is in the range between 0 and 2 for many applications, so we chose 0 as the initial value of σ.

The LM algorithm uses an iteration processes to update interim parameter values towards the optimal solution. In each iteration of the estimation process, we need to determine the damping term μ. As introduced in Section 5.1, this can be achieved through a one-dimensional optimization procedure. However, because there is no analytic solution to the optimal μ, we would apply another iteration process to search for the optimal damping term. Obviously, searching for the accurate value of optimal μ involves too much computation and is not necessary. Instead, a sub-optimal value of μ which leads to steady update towards the optimal parameter vector provides a good trade-off between computation and LM algorithm convergence performance. In our implementation, we apply this sub-optimal approach by increasing or decreasing the interim μ by a pre-defined factor a (for example, $a = 10$) according to the change in the squared estimation error.

There are a number of practical criteria for the termination of the iteration in the LM algorithm. The criterion used in our implementation is to terminate the iteration when the relative change in the squared estimation error drops below a pre-defined threshold, for example, 0.01.

LM is applied to two different variations of the Downey model, the low variance model and the high variance model. The model can deal with phase function.

The above implementation issues are clearly illustrated in the following example.

Example 5.1

In this example, we use the NAS parallel benchmarks to generate speedup measurements. The specific benchmark is the LU solver (introduced in Section 4.1) for class W, the number of nodes are 2, 4, 8, 16, 32, 64 (recall that the LU solver require that $n = 2k$, with $k > 0$ as an integer). The generated speedup measurements are listed as follows,

No. of nodes	2	4	8	16	32	64
Measured speedup	2.00	3.92	7.25	13.29	20.23	24.95
Predicted speedup	1.97	3.83	7.24	13.07	20.77	24.70

Table 5.1: The generated speedup and predicted speedup in Example 5.1

With these measured speedup values, the LM algorithm estimates the optimal parameters as $\hat{A} = 24.70$ and $\hat{\sigma} = 0.74$. Accordingly, we can obtain the speedup prediction for number of nodes range from $n = 2$ to $n = 64$. The predicted speedup corresponding to $n = 2, 4, 8, 16, 32, 64$ are listed in the above table. The fitting of the prediction to the measurement is also shown in Figure 5.1. Not surprisingly, the predicted speedup matches well with the measured speedup values.

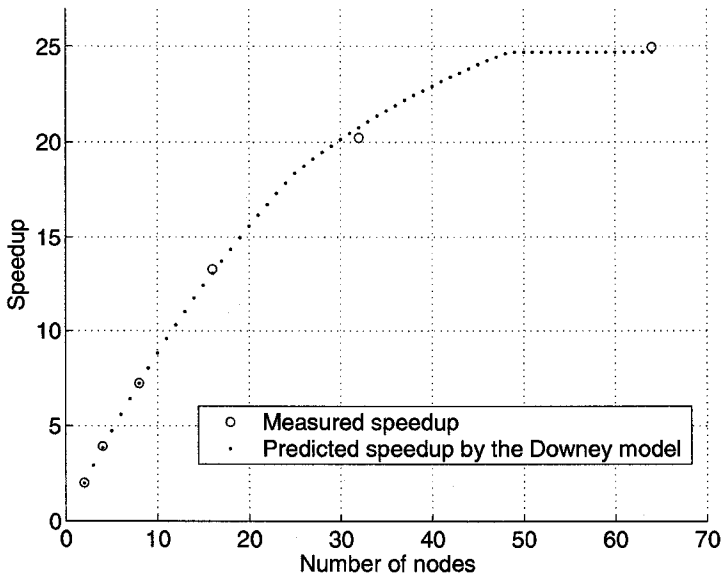

Figure 5.1: Fit of predicted speedup to measured speedup in Example 5.1

Chapter 6

The experiments

In this chapter, we introduce the experiments on the Downey model. The experiments aim to test the fitting of the Downey model to the measured speedup values on sub-clusters of different sizes. We first introduce the experiment environment in Section 6.1, then propose the experiment test cases in Section 6.2. Sections 6.3, 6.4 and 6.5 present the experiment results for the three test cases. Finally, Section 6.6 summaries the observations obtained from the experiments.

6.1 Test environment

We perform all our experiments on SHARCNET (Shared Hierarchical Academic Research Computing Network) [15]. SHARCNET links a number of high performance clusters built for universities and colleges in Canada. In total, SHARCNET includes thousands of processors. All of our experiments are run on the HP narwhal cluster (which is one of the clusters in SHARCNET). Narwhal has 267 nodes and each contains one AMD 2.20 GHz dual core CPU with 8 GB memory but we only use one core of each CPU. The nodes in narwhal are interconnected by Myrinet G2 high speed network. The operating system for narwhal is HP Linux XC 3.1. The

MPI package we used is MPICH 1.2.

6.2 Test cases

We designed several test cases to investigate the performance of the Downey model. The test cases are introduced as follows.

- Case 1: fitting of the Downey model on the same range of speedup measurements. In this test case, we first generate measured speedup values for some pre-selected sub-cluster size (n). The sub-cluster sizes cover a specific range of n of our interest. Note that the selection may not cover all the three parts of the speedup curve, namely the linear part, nonlinear part and the decline part. The Downey model is then applied to provide speedup prediction for the same range of n. Comparison of the predicted speedup with the measured speedup enables us to test the fitting of the Downey model in the same range of available measurements.

- Case 2: speedup prediction with a small number of speedup measurements. Intuitively, decreasing the amount of available measured speedup data leads to under-tuning of the speedup prediction model, and in turn leads to a loss of accuracy in speedup prediction. Moreover, when the number of available measurement points is small, the distribution of n has significant influence on the accuracy of the tuned model. For example, if all the available measurement points are located in the linear part of the speedup curve, then the measurement data provide little information on the speedup curve in the nonlinear part and the decline part. As a result, the tuned model may not fit well in these two parts. In this test case, we limit the number of measurement points to a small number (typically three or four), and test the Downey model under several typ-

ical distributions of n.

- Case 3: separate speedup prediction for communication and computation.

  In the previous test cases, computation and communication are not separated for a job. In this case, we separate these two elements and test the Downey model for speedup prediction on communication and computation separately. The test case is based on the synthetic benchmarks which can generate speedup measurements on computation and communication separately. Note that speedup on communication is directly related to the pattern in which messages are distributed in the sub-cluster.

Obviously, the above test cases cover typical practical scenarios to which the Downey model may be applied. From the experimental results in these test cases, we can draw some conclusions on the applicability of the Downey model to practical applications.

6.3 Experimental results in Test Case 1

As introduced in Section 6.2, Test Case 1 is proposed to test the fitting of the Downey model in the same sub-cluster size range of the available measured speedup. For a better understanding of the behavior of the Downey model in this case, we investigate the following four scenarios.

First, we assume that the measured speedup data only covers the linear part of the speedup curve. We expect the Downey model to fit well with the measurement in the same sub-cluster size range. This is confirmed in the experimental results shown in Figure 6.1. In this experiment we use the NAS-BT benchmarks with class B to generate speedup measurements for the range n=4 to n=144. This range roughly

corresponds to the linear part of the speedup curve. Note that at n=121 and n=144, the error between the measurement and prediction is larger than those at other n values. We believe that this is caused by the fact that these sub-cluster sizes are located at the transition part between the linear and nonlinear parts.

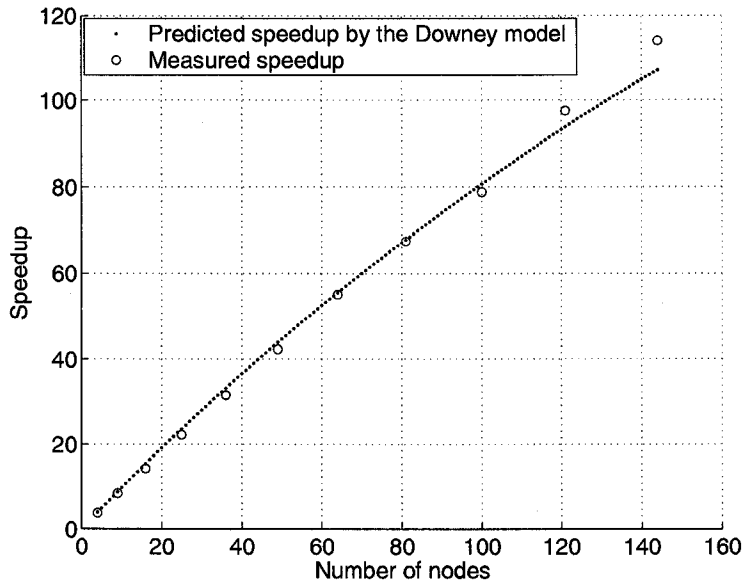

Figure 6.1: Test Case 1: measurement data cover linear part of speedup curve

Secondly, we assume that the measured speedup covers the full range of the linear part and the nonlinear part of the speedup curve. The fitting of the Downey model is illustrated in Figure 6.2. Here, we use the NAS-BT benchmarks with class W to generate speedup measurements for sub-cluster size n=1, 4, 9, 16, 25, 36, 49, 64. Note that in this case the sub-cluster size is required to satisfy condition $n = k^2$ with $k > 0$ to be an integer. For this particular case the maximum speedup value is located at n=64. As can be seen from the figure, the predicted speedup curve fits well with the measurement data.

Next we test the Downey model using measurements in the range of the full linear part and a portion of the nonlinear part of the speedup curve. Intuitively, since

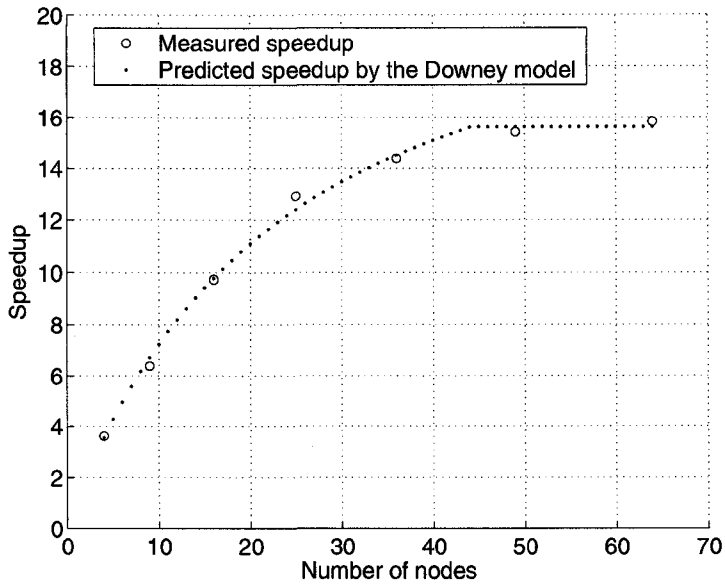

Figure 6.2: Test Case 1: measurement data cover linear and nonlinear parts of speedup curve

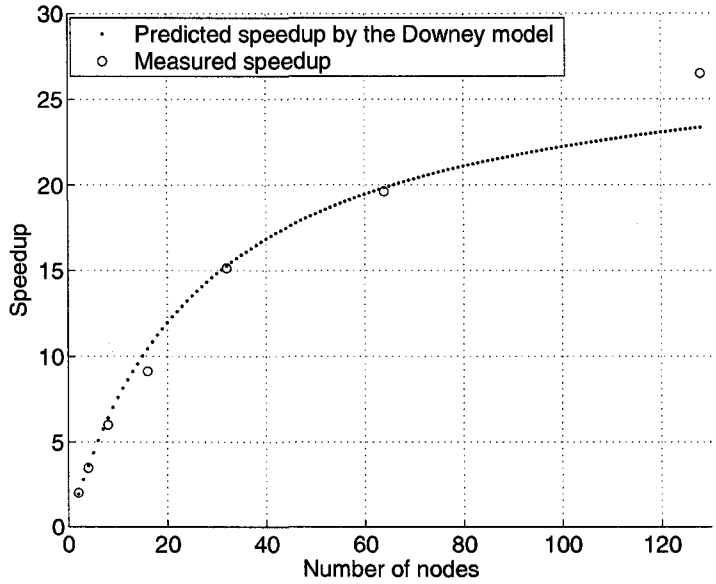

Figure 6.3: Test Case 1: measurement data cover linear part and partial nonlinear part of speedup curve

only a portion of the nonlinear part is covered in the measurement data, we expect the predicted curve fits better with the measurements in the linear part than in the nonlinear part. A typical experiment result is shown in figure 6.3. The measurement

data are generated by NAS-CG benchmarks with Class A. The range of n is from 2 to 128, with $n = 2^k$ where k is a positive integer. The results shown in the figure match with our intuition by showing larger estimation error in the nonlinear part.

The last scenario corresponds to the case in which the measurement data covers all three parts of the speedup curve, i.e., the linear, nonlinear and the decline parts. The results are shown in Figure 6.4. Here, the measurement data are generated by NAS-SP benchmarks with class A. The range of sub-cluster size is from 4 to 144 with $n = k^2$, where k is a positive integer. As shown in the figure, the predicted curve fits well with measurement data in the linear and nonlinear part. However, the estimation error is large in the decline part. This certainly is caused by the fact that the Downey model only models the linear and nonlinear parts of the speedup curve. The measurements in the decline part, if used for the tuning of the model, have only destructive effort on the accuracy of the tuned model.

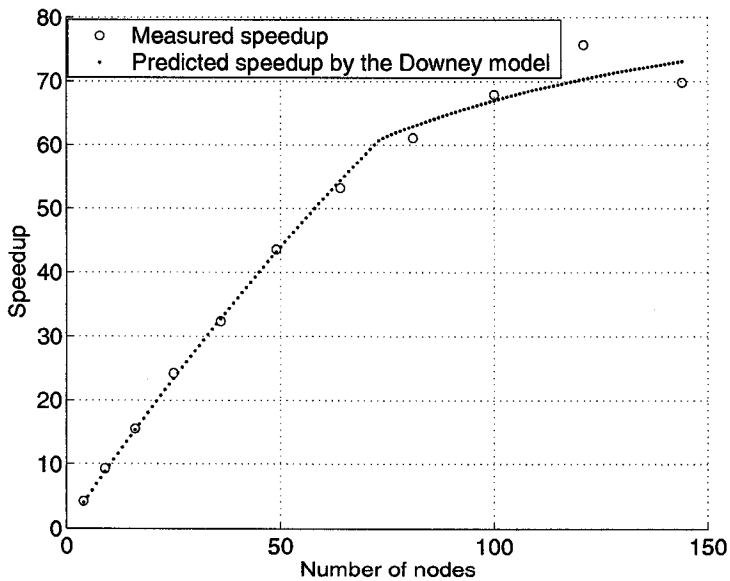

Figure 6.4: Test Case 1: measurement data cover linear, nonlinear and decline parts of speedup curve

6.4 Experimental results in Test Case 2

In the Test Case 2 proposed in Section 6.2, the number of measurement points are small (typically three or four). As pointed out in Section 6.2, the distribution of the n values has a crucial influence on the prediction accuracy. This test case is important for the application of the Downey model. On one hand, in practice, environmental conditions often prevent the availability of speedup measurement data on more than a few sub-cluster sizes. As a result, the speedup prediction has to be based on a few measurement points. On the other hand, a good prediction model should have the capability to capture key characteristics in the speedup behavior in the system through the use of as few measurement points as possible.

In this section, we test the Downey model for this test case, and show the prediction behavior of the model with the following typical distributions of n:

- Scenario A: all the measurement points are located in the linear part of the speedup curve;
- Scenario B: there is one measurement point in the nonlinear part of the speedup curve, all the other points are in the linear part;
- Scenario C: the generated measurement points cover both linear and nonlinear parts of the speedup curve, but only half of the measurement points (one of every two neighboring measurement points) are used for the tuning of the model, in other words the measurements used for model tuning contain information on the speedup curve in the whole range of n;
- Scenario D: for the same measurement set in Scenario C, choose three measurements corresponding to the smallest n, the largest n, and the medium n.

The experimental results for these scenarios are shown as follows.

For comparison purposes, we use the same set of speedup measurements for all four scenarios. The measurements are generated by the NAS-CG benchmarks with class A, the sub-class sizes are n=2, 4, 8, 16, 32, 64, 128.

Figure 6.5 shows the tuned Downey model using measurements at n=2, 4, 8. These three points are all in the linear part of the speedup curve. In other words, this experiment is of Scenario A. As can be seen from the figure, the tuned Downey model only matches the measurements in the linear part of the curve. Obviously, this is because with the three measurement points, there is no information on the speedup curve outside the linear part.

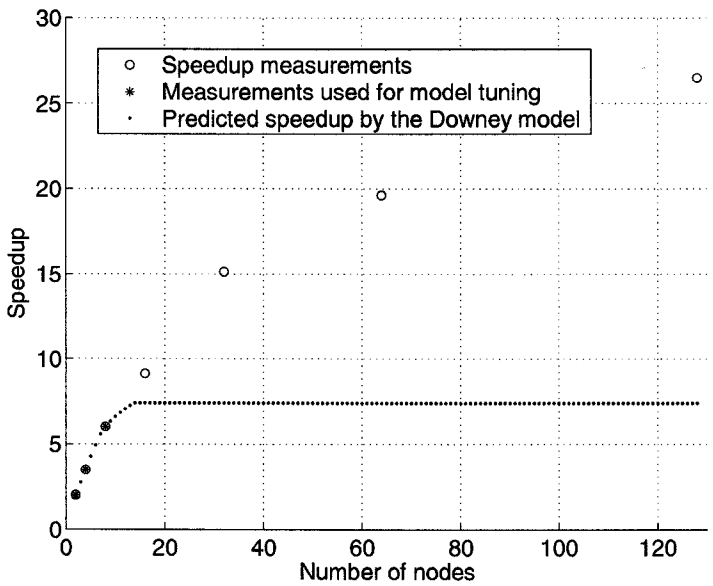

Figure 6.5: Test Case 2, Scenario A: Downey model tunning using measurements at n=2,4,8.

To test for Scenario B, we add one more measurement to the three points used in Scenario A. Specifically, we use measurements at n=2, 4, 8, 16 to tune the Downey model. The result is shown in Figure 6.6. The tuned model only matches with the four used measurements, and does not fit with the other measurements. Obviously, adding one measurement point in the nonlinear part is not enough for the tuning of

the model.

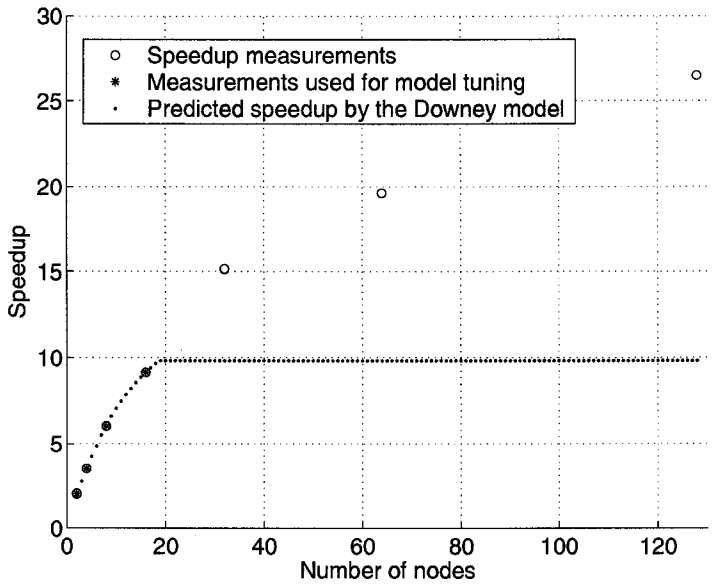

Figure 6.6: Test Case 2, Scenario B: Downey model tunning using measurements at n=2,4,8,16.

Scenario C requires half of the available measurement points. Here we choose the measurements at n=2, 8, 32, 128. The tuned Downey model is shown in Figure 6.7. From the figure we see that the model fits well with the measurements in the whole range of n from 2 to 128.

As for the last scenario, we use the three measurements at n=2, 16 and 128. Figure 6.8 shows the resulting model. The model is almost identical with the one in Figure 6.7 (which is also show in Figure 6.8). We believe that the distribution of the three measurements ensures that the measurements contain the key characteristic information for the speedup curve. This in turn ensures the good fitting of the resulted prediction model.

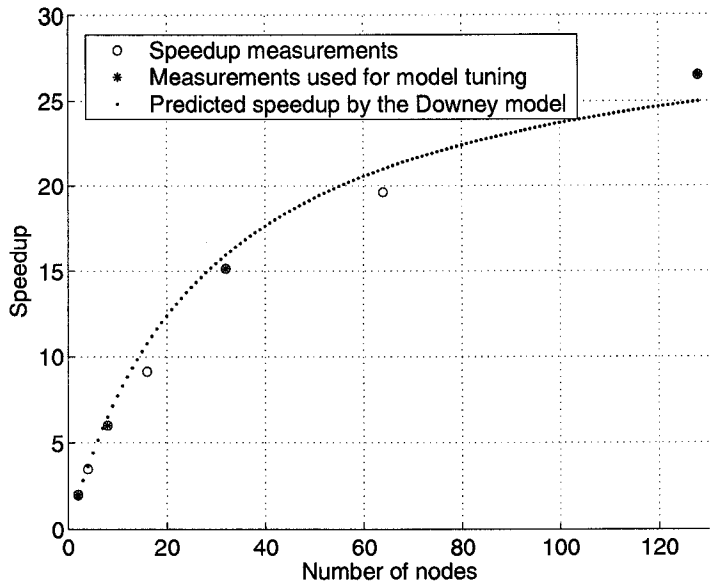

Figure 6.7: Test Case 2, Scenario C: Downey model tunning using measurements at n=2,8,32,128.

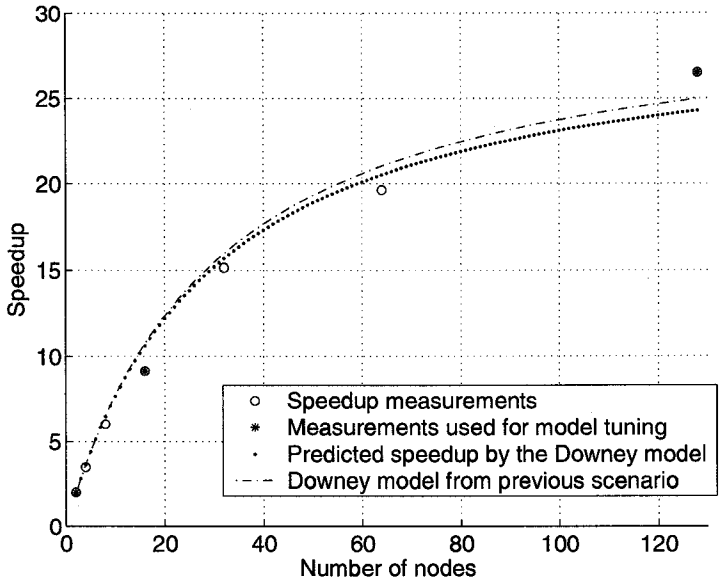

Figure 6.8: Test Case 2, Scenario D: Downey model tunning using measurements at n=2,16,128.

6.5 Experimental results in Test Case 3

In this test case, we use the synthetic benchmarks to separate the run time used for computation and communication in a job. The objective is to test the Downey model on the prediction for computation and communication.

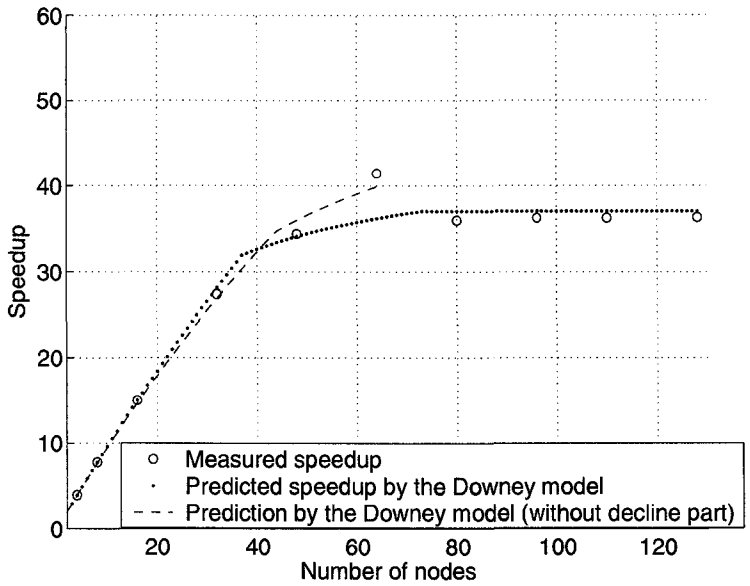

Figure 6.9: Test Case 3, broadcast pattern: speedup measurements and predictions

First we use the synthetic benchmarks with the broadcast communication pattern, the problem size is chosen in such a way that the run time for communication is comparable with that for the computation. If the run time for communication is too small, the speedup curve would lack information on the nonlinear part and the decline part. On the contrary, if the run time for communication is too long, the speedup curve would have too much emphasis on the decline part. The run time measurements correspond to sub-cluster sizes n=1, 2, 4, 8, 16, 32, 48, 64, 80, 96, 112, 128. All the measurements are used in the tuning of the Downey model. Figure 6.9 shows the speedup measurements and the prediction curve by the Downey model. The measured speedup clearly shows all the three parts, i.e., the linear, nonlinear and decline parts. The predicted speedup matches with the measurements in all three parts, but has significant error at the speedup peak point ($n = 64$). This phenomenon is similar to that in the last scenario of Test Case 1 (Figure 6.4). Recall that the Downey model does not model the speedup reduction in the decline part.

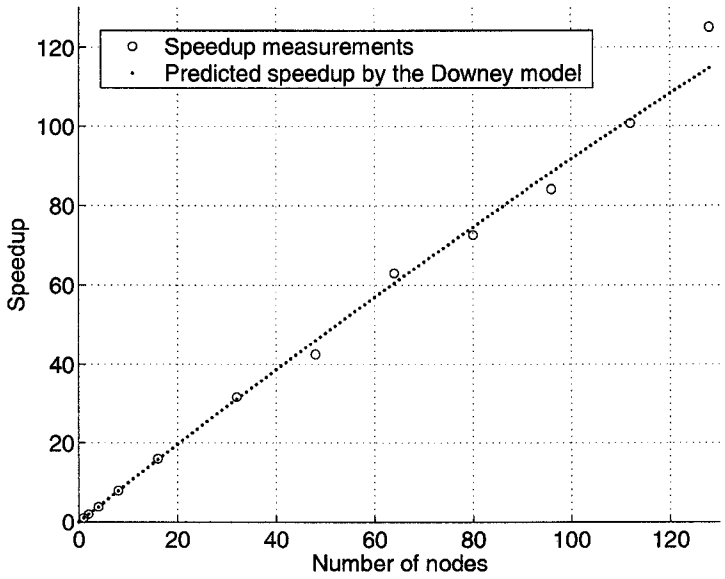

Figure 6.10: Test Case 3, broadcast pattern: speedup measurements and predictions for computation

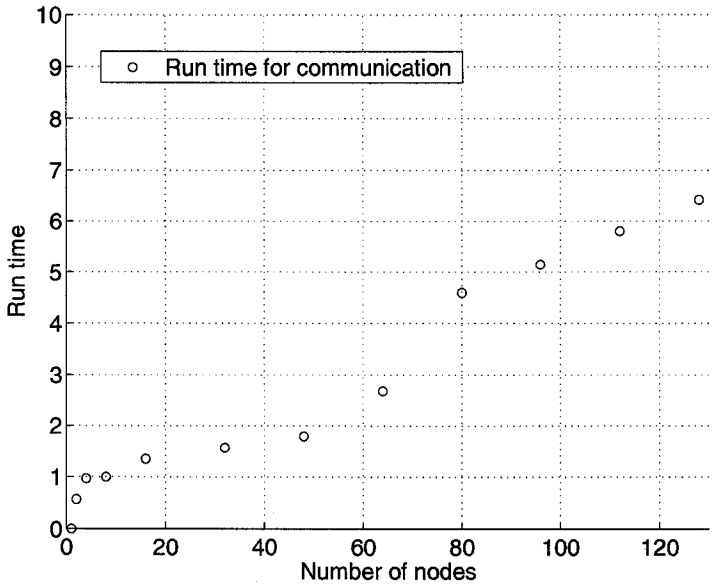

Figure 6.11: Test Case 3, broadcast pattern: run time measurements for communication

Consequently, using measurements in the decline part has destructive effect on the accuracy of the model.

As for the computation involved in the job, we define the speedup for computation as the $S_c(n) = \frac{T_c(1)}{T_c(n)}$, where $T_c(n)$ is the run time for computation. Figure

6.10 shows the speedup measurements and prediction for computation. Obviously, the job simulated in the synthetic benchmarks is well-suited for parallel processing. For each simulated sub-cluster size n, the computation performed on each node is approximately $\frac{1}{n}$ of the total computation. In other words, the speedup curve for computation of the synthetic benchmark increases linearly with n. Maybe in the future, we could try to use another benchmark whose speedup curve for computation increases nonlinearly with n.

We show the run time for communication in Figure 6.11. The communication time increases with n while the time decreases with n in the Downey model, so we can not use the Downey model to predict communication time. As discussed in Chapter 2, the communication in a job is influenced by a number of factors which are difficult to model.

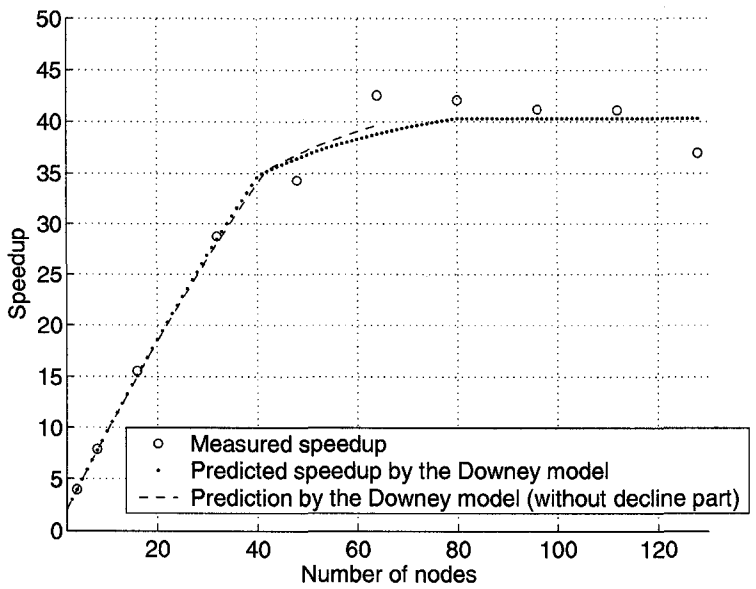

Figure 6.12: Test Case 3, all-to-all pattern: speedup measurements and predictions

The above results are confirmed in the second experiment. Here we use the all-to-all communication pattern in the synthetic benchmarks. The measurements are

for the same set of sub-cluster sizes in the above experiment.

Figure 6.12 shows the speedup measurements and the Downey model predictions. The measurement curve covers the linear, nonlinear and decline parts. Similar to the prediction curve in Figure 6.9, the measurements in the decline part negatively

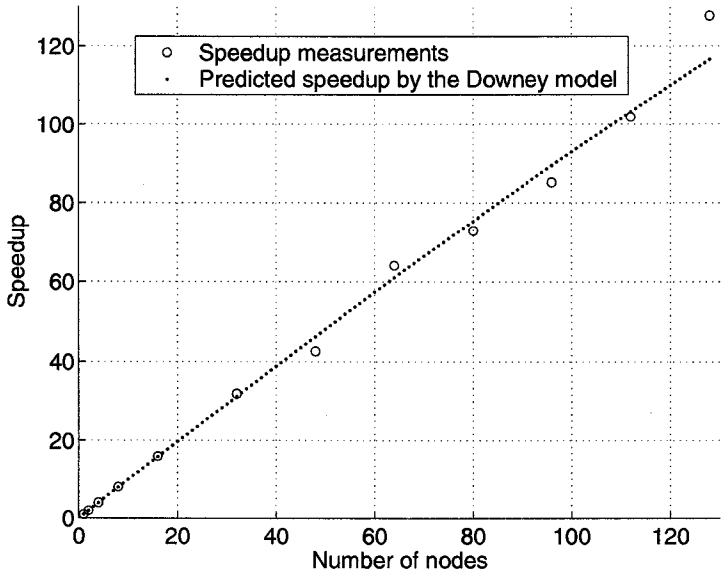

Figure 6.13: Test Case 3, all-to-all pattern: speedup measurements and predictions for computation

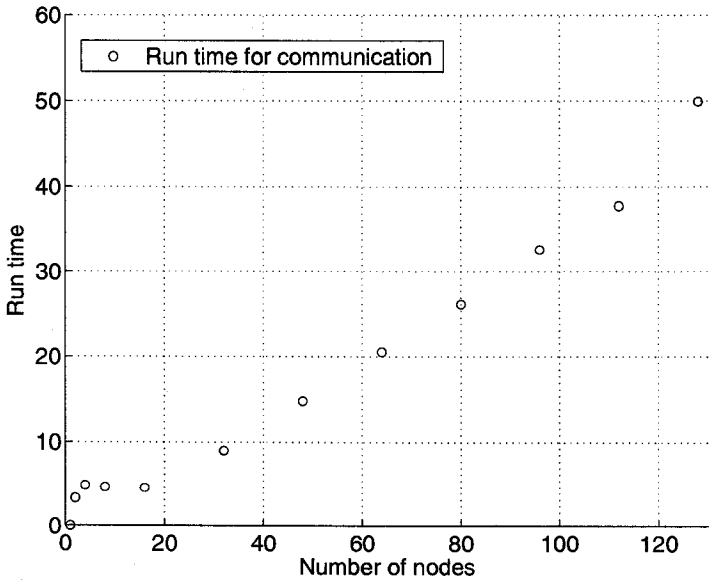

Figure 6.14: Test Case 3, all-to-all pattern: run time measurements for communication

influences the prediction accuracy.

The speedup measurements and prediction for computation is shown in Figure 6.13. Again, the job is well-suited for parallel processing, and the computation speedup shows a linear relationship with respect to the sub-cluster size.

The run time for communication is shown in Figure 6.14. The curve, especially the part for small sub-cluster sizes, show changes which are difficult to explain and model.

The decline part of the speedup curve happens when too many nodes are assigned to a job and the communication overhead is so high that increasing n has a negative effect on speedup. Maybe in the future, we can find a model for the communication, thus we can model the decline part of the speedup curve.

6.6 Some observations from experimental results

In the previous sections, we tested the Downey model for three test cases. In each case, we generate run time measurements using the NAS parallel benchmarks or the synthetic benchmarks for some typical scenarios, and examine the fitting of the optimally tuned Downey model. From the experimental results, we obtain the following observations.

1. Since the Downey model does not model the speedup reduction in the decline part of the speedup curve, using measurements in the decline part reduces the prediction accuracy of the tuned model. The biggest prediction error corresponds to the transition section between the nonlinear part and the decline section. This is the neighborhood of n corresponding to the peak value of speedup. This phenomenon can be clearly seen in the last scenario of Test Case 1 and the two experiments in Test Case 3.

2. Model tuning using measurements in the linear part only results in good prediction in the linear part, this can be seen in the first scenario in Test Case 1. However, the tuned model does not result in good prediction for the nonlinear and decline parts. This can be seen from Scenario A in Test Case 2. Obviously, the remedy to this problem is to add measurements in the nonlinear part of curve, as shown in the second scenario of Test Case 1. However, Scenario B in Test Case 2 and the third scenario in Test Case 1 suggest that a large enough amount of measurements in the nonlinear part is necessary for good prediction.

3. If a sufficient amount of measurements covers the full range of the linear and nonlinear parts of the speedup curve, the tuned model provides good prediction. However, Scenario C and D in Test Case 2 suggest that the model can be accurately tuned using a small number of measurement points. Needless to say, in this case the distribution of n corresponding to the measurements is crucial for the accuracy of the tuned model. From Scenario D of Test Case 2, it is suggested that for the optimal tuning of the Downey model, as few as three measurements is sufficient. However, the three measurements should correspond to a small n in the linear part, a large n close to the peak speedup point, and a n in the transition section between the linear and nonlinear parts.

4. As for the separated computation and communication, we find that jobs generated by the synthetic benchmarks are well-suited for parallel processing. The communication run time is difficult to model. This is due to the fact that the communication time is influenced by multiple complicated system mechanisms.

From the above observations, we find that the Downey model can be accurately tuned using as few as three carefully located measurement points. In comparison, the speedup model in the ScoPred job scheduler can also be accurately tuned with a few

measurement points. In experiments, cases exist in which as few as two measurement points are used to achieve good tuned model in ScoPred. In addition, the distribution of the sub-cluster sizes for the measurements are not as crucial for the model accuracy as for the Downey model case.

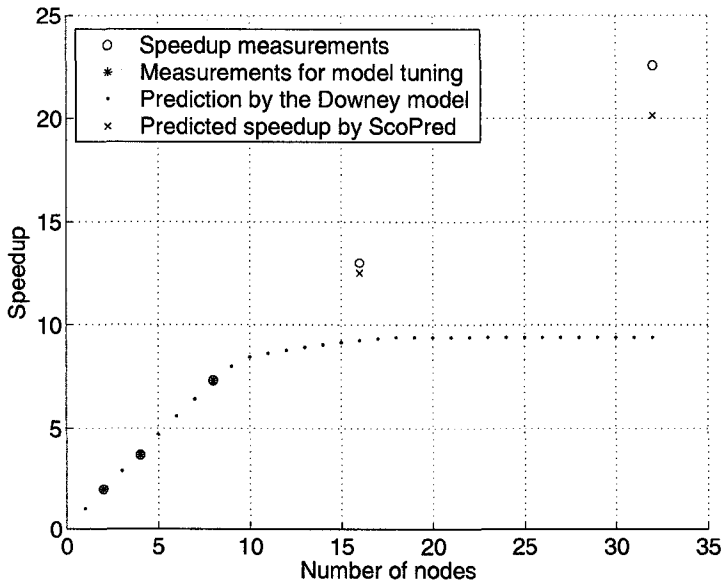

Figure 6.15: Comparison of the Downey model and the ScoPred

For the comparison of the two models, we choose one of the experiments introduced in [10]. It is easy to see that in all the experiments used in [10], the measured speedup points used for the tuning of ScoPred predictor are in the linear part. The experiment we choose uses three points with n=2, 4, 8. The tuned ScoPred predictor then predicts speedup for n=16 and 32. Figure 6.15 shows the measurements and predictions of ScoPred. Using the same set of measurements at n=2, 4, and 8, we obtained the Downey model. The resulting predicted speedup curve is also shown in the figure. Obviously, in this case ScoPred out-performs the Downey model for scalable predication. However, this experiment is of Scenario B in Test Case 2 introduced in Section 6.4. The reason for the poor prediction performance of the Downey

model, as explained in Section 6.4, is that the three measurements do not contain speedup information in the nonlinear part. For ScoPred, the lacked information is well-compensated by the experience input on the system. For the Downey model, adding another carefully chosen measurement is enough to obtain accurate speedup model.

From the above example, it is clear that ScoPred put more effort to include system information into the model in exchange for a lower requirement on the measurement set used for model tuning. In comparison, the Downey model emphasizes the simplicity of the model and obtain the key characteristics only from actual measurements. This fundamental difference between the Downey model and the ScoPred suggests the application of the two models for different scenarios.

Chapter 7

Conclusions and future research

In this thesis, we addressed the speedup prediction problem and investigated the performance of the Downey model for practical speedup prediction applications. We use the NAS parallel benchmarks and our synthetic benchmarks to generate run time measurements on the SHARCNET system, and apply the LM algorithm to tune the Downey model. Our experiments cover some typical scenarios in several proposed test cases, for each scenario prediction performance of the tuned Downey model is obtained and analyzed. From the experimental results, we find that the Downey model can capture the key characteristics of the speedup versus sub-cluster size curve. Satisfactory tuning of the model can be achieved by using very few speedup measurement points. However, the measurement points should provide information on both the linear and nonlinear parts of the speedup curve. Moreover, since the Downey model does not model the speedup reduction in the decline part of the speedup curve, we should avoid using measurements in this part for the tuning of the model. In the extreme case, the model can be accurately tuned using just three measurement points at carefully chosen sub-cluster sizes.

The experiment results suggest several related research problems. First of all,

our experiments show that the distribution of the measurements used for the tuning of the model has crucial influence on the accuracy of the model. This leads to the need to select the optimal set of measurements for the tuning of the Downey model. In the simplest sense, we should remove measurements in the decline part of the speedup curve. Another possible research problem concerns the LM algorithm. The LM algorithm we use in our experiments minimizes the summation of all the squared error on each measurement point. It treats each measurement data equally. However, in some cases some measurements may be more important than the others. Therefore emphasis measurements with different weighting factors may improve the accuracy of the tuned model. Other possible research topics include the use of system information in the speedup modeling (an idea similar to the ScoPred), and better tuning methods other than the LM algorithm.

Bibliography

[1] D. Bailey, E. Barszcz, J. Barton, D. Browning, R. Carter, L. Dagum, R.Fatoohi, S. Fineberg, P. Frederickson, T. LasinsKi, R. Schreiber, H. Simon, V. Venkatakrishnan, and S. Weeratunga. The nas parallel benchmarks. Rnr technical report rnr-94-007, NASA advanced supercomputing disvision, March 1994.

[2] T. B. Brecht and K. Guha. Using parallel program characteristics in dynamic processor allocation policies. In *Performance'96*, October 1996.

[3] S. Chiang, R. K. Mansharamani, and M. K. Vernon. Use of application characteristics and limited preemption for run-to-completion parallel processor scheduling policies. In *the ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems*, 1994.

[4] L. W. Dowdy. On the partitioning of multiprocessor system. Technical report, Vanderbilt University, March 1988.

[5] A. B. Downey. A model for speedup of parallel programs. Technical Report UCB/CSD-97-933, EECS Department, University of California, Berkeley, 1997.

[6] A. B. Downey. A parallel workload model and its implications for processor allocation. *Cluster Computing*, 1(1):133–145, 1998.

[7] D. L. Eager, J. Zahorjan, and E. L.Lazowska. Speedup versus efficiency in parallel system. *IEEE Transactions on Computers*, 38(3):408–423, March 1989.

[8] E. Frachtenberg, D. Feitelson, F. Petrini, and J. Fernandez. Flexible coscheduling: utilization of heterogeneous resources. In *Parallel and Distributed processing Symposium (IPDPS'03)*, Nice, France, April 2003.

[9] K. Guha. Using parallel program characteristics in dynamic multiprocessor allocation policies. Technical report, York University, May 1995.

[10] B. J. Lafreniere and A. C. Sodan. Scopred-scalable user-directed performance prediction using complexity modeling and historical data. In *the 11th International Workshop on Job Scheduling Strategies for Parallel Processing - JSSPP*, volume 3834 of *Lecture Notes in Computer Science*, pages 62–90, 2005.

[11] M. Lourakis. *levmar: Levenberg-Marquardt nonlinear least squares algorithms in C/C++*. Institute of Computer Science, Foundation for Research and Technology - Hellas, Heraklion, Crete, Greece. http://www.ics.forth.gr/ lourakis/levmar/.

[12] M. I. A. Lourakis. *A Brief Description of the Levenberg-Marquardt Algorithm Implemented by Levmar*. Institute of Computer Science,Foundation for Research and Technology-Hellas (FORTH), Vassilika Vouton P.O. Box 1385, GR 711 10 Heraklion, Crete, GREECE, February 11 2005.

[13] G. Marrin and J. Mellor-Crummey. Cross-architecture predictions for scientific application using parameterized models. *Proc. Joint. Internat. Conf. on Measurement and Modeling of Computer Systems(SIGMETRICS),New York, NY,USA*, June 2004.

[14] NPB. http://www.nas.nasa.gov/Resources/Software/npb.html.

[15] SHARCNET. http://www.sharcnet.ca/.

[16] E. Smirni, E. Rosti, L. W. Dowdy., and G. Serazzi. Evaluation of multiprocessor allocation policies. Technical report, Vanderbilt University, 1993.

[17] A. C. Sodan. Loosely coordinated coscheduling in the context of other dynamic approaches for job scheduling-a survey. *Concurrency and Computation: Practice and Experience*, 2005.

[18] A. C. Sodan, C. Doshi, L. Barsanti, and D. Taylor. Gang scheduling and adaptive resource allocation to mitigate advance reservation impact. *ccgrid*, 1:649–653, 2006.

[19] X. Zeng, J. Shi, X. Cao, and A. C. Sodan. Grid scheduling with atop-grid under time sharing. *CoreGrid Workshop on Grid Middleware (in conjunction with ICS), Dresden*, June 2007. to appear in Springer, 16 pages.

Vita Auctoris

Name: Lin Lan

Place of Birth: Sichuan, China

Education:

2004-2007 M.Sc. Computer Science
University of Windsor
Windsor, Ontario, Canada

1994-1999 B.Sc. Computer Science
University of Electronic Science and
Technology of China
Chengdu, Sichuan, China