Scalability of Heterogeneous Computing

Xian-He Sun, Yong Chen, Ming Wu
Department of Computer Science
Illinois Institute of Technology
{sun, chenyon1, wuming}@iit.edu

Abstract

Scalability is a key factor of the design of distributed systems and parallel algorithms and machines. However, conventional scalabilities are designed for homogeneous parallel processing. There is no suitable and commonly accepted definition of scalability metric for heterogeneous systems. Isospeed scalability is a well-defined metric for homogeneous computing. This study extends the isospeed scalability metric to general heterogeneous computing systems. The proposed isospeed-efficiency metric is suitable for both homogeneous and heterogeneous computing. Through theoretical analysis, we derive methodologies of scalability measurement and prediction for heterogeneous systems. Experimental results verify the analytical results and confirm that the proposed isospeed-efficiency scalability works well in both homogeneous and heterogeneous environments.

1. Introduction

Scalability is an essential factor for performance evaluation and optimization of parallel and distributed systems. It has been used widely for describing how the system size and problem size will influence the performance of parallel computers and algorithms. It measures the ability of parallel architectures to support parallel processing at different machine ensemble sizes, and measures the inherent parallelism of parallel algorithms.

Although scalability is important for parallel and distributed systems, most of current research is focused on homogeneous environments. As computing environments evolving, understanding scalability of heterogeneous environments becomes timely important and necessary. In this paper, we propose an isospeed-efficiency method for general heterogeneous computing, based on the isospeed metric proposed in [10]. Analytical and experimental studies are conducted to confirm the correctness and effect of the newly proposed scalability metric. Results show that the isospeed-efficiency metric is practical and effective.

The following of this paper is organized as follows: Section 2 reviews related work. Section 3 presents the proposed isospeed-efficiency scalability metric. The analytical results of the proposed scalability metric are provided in Section 4. Section 5 presents experimental results that match the analytical results well. Finally, we summarize our current work and discuss future work.

2. Related work

Several metrics are proposed to measure the scalability of algorithms and parallel machines[3][5][7][8][9][10], but most of these metrics are designed for homogeneous environments. They cannot readily be applied to heterogeneous environments. In [10], Sun and Rover proposed the isospeed scalability metric to describe the scalability of an algorithm-machine combination in homogenous environments. An algorithm-machine combination is defined to be scalable if the achieved average unit speed of the algorithm on the given machine can remain constant with increasing number of processors, provided the problem size can be increased with the system size. By this definition, a scalability function can be defined as

$$\psi(p, p') = \frac{p'W}{pW'}$$

where p and p' are the initial and scaled number of processors of the system respectively, and W and W' are the initial and scaled work (problem size) respectively. The isospeed scalability works well in homogeneous environment and is well cited in scholarly publications, including several widely used textbooks [2][4][7].

There is another well-known scalability metric, isoefficiency scalability[3]. The isoefficiency scalability is defined as the ability of parallel machine to keep the

parallel efficiency constant when the system and problem size increases, where the parallel efficiency is defined as speedup over the number of processors. Speedup, in turn, is defined as the ratio of sequential execution time and parallel execution time. In theoretical analysis, the requirement of sequential execution time does not appear to be a problem. In practice, to measure the execution time of large applications on a single node is problematic, if not impossible. Scalability is to measure the ability of parallel systems at different system and problem sizes. It does not need to refer single-node sequential execution time of large scale computing. Isoefficiency scalability may have some difficulty to extend to general heterogeneous environments directly.

Isospeed scalability uses average unit speed as efficiency and does not refer to sequential execution time. It is practical. However, similar to isoefficiency scalability, it is based on the assumption that the underlying parallel machine is homogeneous. This assumption becomes too restricted for modern computing systems. It is necessary to extend the isospeed scalability metric to general parallel computing environments. In this study, we combine the merits of both isospeed and isoefficiency scalability to propose the isospeed-efficiency scalability for heterogeneous computing.

There are some recent attempts to generalized scalability metrics. Jogalekar and Woodside proposed a strategy-based scalability metric for general distributed systems[5]. The scalability is based on productivity which is defined as the value delivered by the system divided by its cost (money charge) per unit time. A system is scalable if productivity keeps pace with cost. Their scalability metric measures the worthiness of renting a service. However, commercial charge varies from customer to customer, based on business considerations, and does not necessarily reflect the inherent scalability of the underlying computing system. Pastor and Bosque proposed a heterogeneous efficiency function to define the heterogeneous scalability[7]. Their work tries to extend the homogeneous isoefficiency scalability model to heterogeneous computing and, therefore, inherits the limitation of parallel speedup, requiring the measurement of solving large-scale problem on single node as we analyzed for homogeneous isoefficiency scalability.

3. Isospeed-efficiency scalability

3.1 The marked speed

In order to fully describe the characteristic of a given algorithm-system combination in a heterogeneous environment, we need to describe all the computing features of the system. These features include the computing power, memory capacity and memory speed, network bandwidth, I/O speed and other features of the system. In engineering practice, however, we cannot get into all the details; otherwise, the scalability model will be too complex to use. We need to balance the simplicity and effectiveness. For this reason, we introduce a new concept, 'marked speed', to describe the combined computing power of a general distributed system. First, we give a definition of 'marked speed of a computing node'.

Definition 1 The marked speed of a computing node is a (benchmarked) sustained speed of that node.

Marked speed can be calculated based on hardware peak performance, which in general is much higher than an actual delivered performance. In practice, we can use benchmarks to measure the "power" or marked speed of each node. Standard benchmarks can be selected for different applications, for instance Linpack or NPB for scientific computing or appropriate benchmark from the Perfect benchmarks suite. Once the marked speed of a computing node is measured, it should be used as a constant parameter. Let C_i denote the marked speed of node i. In a heterogeneous environment, C_i may be different from each other due to the heterogeneity of the nodes. In homogeneous environment, all C_i are the same.

Definition 2 The marked speed of a computing system is the sum of the marked speed of each node that composes the computing system.

Let C stand for the marked speed of a computing system. According to definition, we have $C = \sum_{i=1}^{p} C_i$ in a general parallel computing environment with p nodes.

3.2 Definition of isospeed-efficiency scalability

Let S denote the actual achieved speed, which is defined as work divided by execution time, of a computing system. Let W denote work and T denote execution time. So we have $S = W / T$.

Marked speed describes the computational capability of a computing system, which is a constant for a study. The achieved speed of an application may not be the same as the benchmarked marked speed given by Definition 2, especially for distributed/parallel computing where the marked speed does not consider the communication cost. Achieved speed describes the actual computational performance

when the system tries to solve users' applications. It varies with the system and problem size.

Definition 3 The speed-efficiency of a computing system is defined as the achieved speed divided by the marked speed of the computing system.

The speed-efficiency reflects the performance gain of an algorithm-system combination. Let E_s denote the speed-efficiency. So we have $E_s = \frac{S}{C} = \frac{W}{TC}$.

Based on previous definitions and discussion, we propose the following isospeed-efficiency scalability for any algorithm-system combination on a general distributed computing system.

Definition 4 Isospeed-efficiency Scalability An algorithm-system combination is scalable if the achieved speed-efficiency of the combination can remain constant with increasing system ensemble size, provided the problem size can be increased with the system size.

In this isospeed-efficiency scalability definition, the computing system can be either a homogeneous or heterogeneous system. The method for increasing system size includes increasing nodes, increasing the number of processors in one or more nodes, or upgrading to more powerful nodes. The approach to increase problem size depends on the algorithm.

3.3 Isospeed-efficiency scalability function

For a scalable algorithm or application, its communication requirement should increase slower than its computation requirement. So we can increase the problem size to keep the speed-efficiency constant when the system size is increased. The increment of problem size depends on the underlying algorithm and computing system. This variation provides a quantitative measurement of scalability.

Let C, W and T be the initial system size (we call a system with marked speed C as a system with system size C in the rest of this study), problem size and execution time. Let C' be the increased system size, W' be the increased problem size, and T' be the new execution time for the scaled problem size. Then, we have the isospeed-efficiency condition:

$$\frac{W}{TC} = \frac{W'}{T'C'},$$

This condition is to constrain the new problem size W'.

We define the **isospeed-efficiency scalability function** as:

$$\psi(C, C') = \frac{C'W}{CW'}$$

In the ideal situation, $W' = C'W/C$ and $\psi(C, C') = 1$. Generally, $W' > C'W/C$ and $\psi(C, C') < 1$.

When we apply the isospeed-efficiency scalability to a homogeneous environment, because all C_i are equal, we have $C = pC_i$, and $C' = p'C_i$. Thus, the scalability function is

$$\psi(C, C') = \frac{C'W}{CW'} = \frac{p'W}{pW'}$$

This shows that the original homogeneous isospeed scalability metric is a special case of isospeed-efficiency scalability metric.

3.4 Theoretical studies

The following theoretical results are important for scalability study.

Theorem 1: Let an algorithm has a balanced workload on each node and the sequential portion (which cannot be parallelized) of the algorithm is α, if we can find a problem size to keep the speed-efficiency constant when the system size is increased, then the system is scalable and the scalability is

$$\psi(C, C') = \frac{t_0 + T_o}{t_0' + T_o'}$$

where t_0 and t_0' are the execution time of the sequential portion, T_o and T_o' are the communication overhead of system C and C' separately.

Proof: The parallel execution time can be divided into two parts, $T = T_c + T_o$, where T_c is the computation time, and T_o is the total overhead spent on communication, synchronization and other overhead. If computing system C is used to compute a problem with size W, and W' is the increased problem size to satisfy the isospeed-efficiency condition when the computing system is increased to C', we have

$$\frac{W}{(T_c + T_o)C} = \frac{W'}{(T_c' + T_o')C'}$$

Since the algorithm is evenly load balanced and the sequential portion of the algorithm is α, we have

$$W_i = (1-\alpha)W\frac{C_i}{C}$$

where W_i is the workload assigned on node *i*, so,

$$T_c = \frac{W_i}{C_i} + t_0 = \frac{(1-\alpha)W}{C} + t_0$$

where $t_0 = \frac{\alpha W}{C_i}$, which represents the execution time of the sequential portion of algorithm

hence,

$$\frac{W}{\left[(\frac{(1-\alpha)W}{C}+t_0)+T_o\right]C}=\frac{W'}{\left[(\frac{(1-\alpha)W'}{C'}+t_0')+T_o'\right]C'}$$

Thus, the increased problem size W' is

$$W'=\frac{C't_0'+C'T_o'}{Ct_0+CT_o}\cdot W=\frac{C'(t_0'+T_o')}{C(t_0+T_o)}\cdot W$$

Therefore, the computing system is scalable and the scalability is

$$\psi(C,C')=\frac{C'W}{CW'}=\frac{C'W}{C\cdot\frac{C'(t_0'+T_o')}{C(t_0+T_o)}\cdot W}=\frac{t_0+T_o}{t_0'+T_o'} \quad \blacksquare$$

Theorem 1 not only provides a method to calculate the scalability of an algorithm-system combination, but also shows a meaningful and significant understanding for scalability. It reflects that the scalability is decided by both the sequential portion of the work and the communication overhead. When the problem size is increased to keep the speed-efficiency constant, the sequential portion of work might be increased because the problem size is increased, and the communication overhead is increased too because the system size is increased. So in practice, the scalability is likely to be smaller than 1.

Corollary 1: If an algorithm can be parallelized perfectly and has a balanced workload on each node, and if the communication overhead is constant for any problem size and system size, then the algorithm-system combination is scalable and the scalability is perfect with a constant value 1.

Proof: According to theorem 1, we have

$$\psi(C,C')=\frac{t_0+T_o}{t_0'+T_o'}$$

In an ideal case, the algorithm can be parallelized perfectly, which means $\alpha=0$. Thus, $t_0=t_0'=0$.

If the communication overhead is constant at any problem size and system size, we have $T_o=T_o'$.

Therefore, the scalability is 1. ■

Corollary 1 shows the scalability of an ideal case. According to previous analysis of the isospeed-efficiency scalability, the scalability of an ideal case is 1.

Corollary 2: If an algorithm can be parallelized perfectly and has a balanced workload on each node, and if we can find a problem size to keep the speed-efficiency constant when the system size is increased, then the algorithm-system combination is scalable and the scalability is

$$\psi(C,C')=\frac{T_o}{T_o'}$$

Proof: Similar to the proof in Corollary 1, if the algorithm can be parallelized perfectly, we have $\alpha=0$ and $t_0=t_0'=0$. According to theorem 1, the scalability is

$$\psi(C,C')=\frac{t_0+T_o}{t_0'+T_o'}=\frac{T_o}{T_o'} \quad \blacksquare$$

Corollary 2 demonstrates that if an algorithm can be parallelized perfectly and has a balanced workload on each node, then the scalability will only be decided by the total overhead at different problem and system size.

These theorems and corollaries show that if we are able to analyze the total overhead at system C and C', and the sequential ratio of the algorithm, we can calculate and predict the scalability of system with size C' based on the system with size C. We will show this method in experiments.

3.5 Calculation of isospeed-efficiency scalability

The isospeed-efficiency scalability can be obtained in many ways. The most straightforward one is to compute the scalability. This method is to measure the execution time at different system and problem sizes and calculate the scalability by using isospeed-efficiency scalability definition.

Another approach is to analyze and predict the scalability. This method is to analyze the computational part and communicational part of the algorithm and the communication latency of the machine, and then use derived theoretical results to calculate the scalability. This method can also be used to verify the computed scalability or predict the scalability.

The following experiments will show how we can compute or predict the isospeed-efficiency scalability.

4. Experimental results and analyses

Experimental testing has been conducted to verify the correctness of the isospeed-efficiency and the associated analytical results and to show the isospeed-efficiency is practically applicable.

4.1 Algorithm and implementation

Two classical algorithms, Gaussian Elimination algorithm and Matrix Multiplication algorithm, are selected for testing. Both of them are used widely in scientific computing.

4.1.1 Gaussian Elimination (GE). GE algorithm solves dense linear equations $Ax=b$, where A is a

known matrix of size $N \times N$, x is the required solution vector, and b is a known vector of size N. The algorithm has two stages:

(1) Gaussian elimination stage: the original system of equations is reduced to an upper triangular form $Ux = y$, where U is a matrix of size $N \times N$ in which all elements below the diagonal are zeros and the diagonal elements have the value 1. The vector y is the modified version of vector b.

(2) Back substitution stage: the new system of equations is solved to obtain the value of x.

The parallel GE algorithm used in experiments is described as following.

(1) Process 0 distributes the data of matrix A and vector b proportionally to other nodes according to their marked speeds by using row-based heterogeneous cyclic distribution[6]

(2) All processes compute concurrently:

(2.1) For ($i = 0$; $i < N - 1$; i++)

(2.1.1) The process which owns the pivot row broadcasts the pivot row to all processes

(2.1.2) For ($j = i + 1$; $j < N$; j++)

(a) Each process judges if row j belongs to itself or not

(b) If yes, then conducts Gaussian elimination on this row

(2.2) Synchronize all processes due to data dependence

(3) Process 0 collects temporary results from other processes and conducts the back substitution stage

Through analyzing the algorithm, the workload of this algorithm is,

$$W(N) = \frac{2}{3}N^3 - \frac{1}{2}N^2 - 3\frac{1}{6}N + 3$$

This polynomial is used to calculate the workload in our experiments.

4.1.2 Matrix Multiplication (MM). MM algorithm calculates the product of two matrices, $C = A \times B$. For simplicity, we restrict matrix A and B to be square $N \times N$ matrices. [1] conducted a thorough research for matrix multiplication optimization on heterogeneous platform. It stated that the matrix multiplication optimization problem on heterogeneous platform is the problem to balance the workload with different speed resources and minimize the communication overhead. Unfortunately, this problem has been proved to be a NP-Complete problem. A polynomial heuristic algorithm called Optimal Column-based Tiling was thus proposed and proved to be a good solution for heterogeneous platforms in [1].

Our experiments are designed to verify the proposed scalability metric. There is no intent to introduce new algorithms on heterogeneous platforms. We have implemented a simple row-based heuristic algorithm. Our algorithm adopts the HoHe strategy proposed in [6]. The HoHe strategy distributes homogeneous processes over different speed processors with each process running on a separate processor, while distribution of matrices over the processes is heterogeneous block cyclic pattern. In our algorithm, first, process 0 distributes matrix A by using a row-based heterogeneous block distribution, which means A is distributed proportionally into other nodes according to these nodes' marked speeds. Then process 0 distributes matrix B to other nodes. After data distribution, each node computes part of the matrix multiplication on its own data. Finally, process 0 collects all results from other processes. Since there is no communication during computation, communication only occurs in data distribution and collection. This algorithm is not a perfect algorithm, but this algorithm does balance the workload between different speed resources, since each node will work on $N \times C_i / C$ rows of data and the workload of each node is $2 \times N^3 \times C_i / C$. The total workload of our algorithm is $W(N) = 2 \times N^3$. The implementation of this algorithm follows HoHe strategy, which generates the same number of processes as the number of processors and distributes each process on a separate processor.

4.2 Experimental platform

The experimental platform is the Sunwulf cluster in the Scalable Computing Software (SCS) laboratory at Illinois Institute of Technology. Sunwulf is composed of one SunFire server node (sunwulf node), 64 SunBlade compute nodes (hpc-1 to hpc-64) and 20 SunFire V210 compute nodes (hpc-65 to hpc-84). The server node has four CPUs and 4GB memory. Each CPU is 480 MHz. The SunBlade compute node has one 500-MHz CPU and 128M memory. The SunFire V210 compute node has two 1GHz CPUs and 2GB memory. The network connecting all these nodes is 100M Ethernet. The software platform includes SunOS 5.8 and MPICH 1.2.5 release version.

4.3 Measuring the marked speed

In our experiments, we use NASA Parallel Benchmark to measure the marked speed. We run each benchmark, including LU, FT, BT, and etc., on each node and take the average speed on each node as its marked speed. Table 1 gives the measured marked speed of the server node with one CPU, the SunBlade

compute node and the SunFire V210 compute node. Once the marked speed of each node is measured, the marked speed of a whole computing system can be calculated according to Definition 2. For example, if we choose the following nodes to participate computation: Server node with 1 CPU, one SunBlade compute node and two SunFire compute nodes with 1 CPU, the marked speed of this computing system is:

$$20.88 + 20.29 + 2 \times 36.45 = 114.07 \text{ (Mflops)}$$

Table 1 Marked speed of Sunwulf nodes (Mflops)

Node	Server Node (1 CPU)	SunBlade	SunFire V210 (1 CPU)
Marked Speed	20.88	20.29	36.45

4.4 Experimental results and analyses

4.4.1 GE experimental results. Experiments have been conducted to analyze the proposed scalability metric for GE algorithm with different system configurations. We start with two nodes of Sunwulf, one SunBlade node and one server node. In this case, server node uses two CPUs. From Table 1, we can calculate the marked speed of this environment.

$$C_2 = 20.88 \times 2 + 20.29 = 62.05 \text{ (Mflops)}$$

Table 2 shows the workload, execution time, achieved speed and speed-efficiency of GE at different matrix sizes on two nodes. The speed-efficiency is calculated according to Definition 4.

Table 2 Experimental results on two nodes

Rank N	Workload W	Execution Time T	Achieved Speed	Speed-efficiency
100	661353	260.770	2.536	0.041
200	5312703	473.786	11.213	0.181
300	17954053	925.242	19.405	0.313
400	42585403	1587.725	26.822	0.432
500	83206753	2657.918	31.305	0.505

Based on the experimental results, we show the relationship between speed-efficiency and matrix size in Fig. 1. Since the function between speed-efficiency and matrix size is polynomial, we use a polynomial trend line to approach the sample results. From the polynomial trend line, we can read the approximate value of speed-efficiency at any matrix size or read the approximate required matrix size to obtain a specified speed-efficiency. For example, if we want to obtain a speed-efficiency of 0.3, the required matrix size should be around 310. We measured the speed-efficiency when matrix size is 310 and the result is 0.312, which is shown with a light gray dot in Fig. 1. This verifies the method that we can read the required matrix size for a specified speed-efficiency from trend line works.

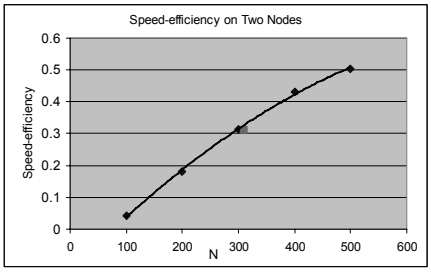

Fig. 1 Speed-efficiency on two nodes

Now, we increase the system size to four nodes and the configuration of four nodes has also changed. The new computing system is composed of hpc-40, hpc-41, hpc-42 and server node with two CPUs. Similar to the analysis in the case of two nodes, we calculate the workload and marked speed, then calculate the speed-efficiency. In this case, the marked speed is C_4 = 102.63Mflops. The required matrix size to obtain a 0.3 speed-efficiency is around 480.

Based on these results, we use the first method in Section 3.5 to calculate the scalability when system size increases from two nodes to four nodes. We select the speed-efficiency at 0.3. The required matrix size will be around 310 in the case of two nodes and 480 in the case of four nodes. According to the scalability function definition, we have

$$\psi(C_2, C_4) = \frac{C_4 \cdot W(N)}{C_2 \cdot W(N')} = 0.445$$

Similar to the previous analysis, we obtain the results of the case of 8 nodes, 16 nodes and 32 nodes. In each case, one node is server node and the rest nodes are SunBlade compute nodes. We select speed-efficiency at 0.3 and read the required matrix size from all those figures. The result is shown in **Table 3**.

Table 3 Required rank to obtain 0.3 speed-efficiency

System Configuration	Rank N	Workload W	Marked Speed(Mflops)
2 Nodes, C_2	310	1819093	62.05
4 Nodes, C_4	480	11682713	102.63
8 Nodes, C_8	1000	6.66E+08	183.79
16 Nodes, C_{16}	1700	3.27E+09	346.11
32 Nodes, C_{32}	3200	2.18E+10	670.75

The measured scalability of GE algorithm on Sunwulf is shown in following table.

Table 4 Measured scalability of GE on Sunwulf

$\psi(C_2, C_4)$	$\psi(C_4, C_8)$	$\psi(C_8, C_{16})$	$\psi(C_{16}, C_{32})$
0.445	0.198	0.383	0.290

4.4.2 MM experimental results. Another experiment we have conducted to analyze the proposed scalability metric is MM algorithm. We have tested the MM algorithm on 2, 4, 8, 16 and 32 nodes of Sunwulf cluster. In each case, half nodes are SunBlade compute nodes and the other half nodes are SunFire V210 nodes except one node is server node. For example, in the case of 8 nodes, the computing system is composed of one server node, three SunBlade compute nodes and four SunFire V210 compute nodes. The marked speed of the computing system in each case is different with previous experiment due to different system configuration. For instance, in the case of 8 nodes, the marked speed is:

$C_8' = 20.88 + 3 \times 20.29 + 4 \times 36.45 = 227.55$ (Mflops)

The experiment is similar to the previous experiment. The details are omitted here. The speed-efficiency of MM algorithm at different system configurations is given in Fig. 2.

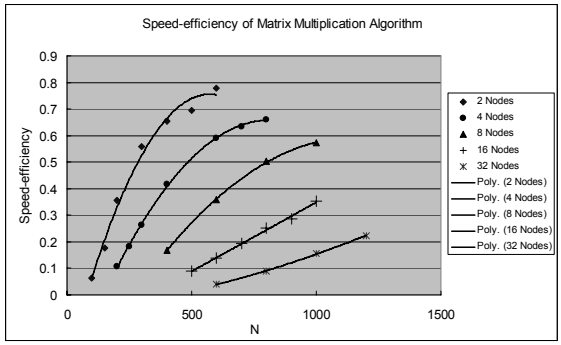

Fig. 2 Speed-efficiency of MM on Sunwulf

Similarly to the analysis in the GE experiment, we use the first method in Section 3.5 to calculate the scalability when system size changes. If the given speed-efficiency is 0.2, we read the required matrix size at different system configurations from Fig. 2. Then, we calculate the isospeed-efficiency scalability for the MM algorithm on Sunwulf as following.

Table 5 Scalability of MM on Sunwulf

$\psi(C_2', C_4')$	$\psi(C_4', C_8')$	$\psi(C_8', C_{16}')$	$\psi(C_{16}', C_{32}')$
0.539	0.416	0.443	0.470

4.4.3 Comparison of two experimental results. The above two experiments are actually two different algorithm-system combinations. Although both of them are conducted on a series of different system configurations of the Sunwulf cluster, the machine parameters, such as machine latency, are the same. By using the proposed isospeed-efficiency scalability metric, we are able to quantify the scalability of these two different combinations.

Compared with the scalability of GE-Sunwulf combination given in Table 6, we find the scalability of MM-Sunwulf combination is higher. This indicates that the MM-Sunwulf combination is more scalable than the GE-Sunwulf combination. The GE algorithm has a sequential portion and has more communications than that of the MM algorithm. Its scalability should be smaller than the scalability of the latter. Our experiment verifies this fact and presents actual quantified scalability advantage of the MM algorithm on Sunwulf cluster.

4.5 Scalability prediction

As we proved in Section 3, if we are able to analyze the sequential ratio and measure the communication latency of the machine, we can predict the scalability. We use the GE-Sunwulf combination to illustrate how scalability can be predicted.

According to the parallel GE algorithm in Section 4.1.1, the sequential ratio of this algorithm is $\alpha = O(1/N)$. When N is large enough, we treat $\alpha \approx 0$. The total communication overhead is

$T_o = T_{broadcast} + 2 \times (p-1) \times (T_{send} + T_{recv}) + N \times (2 \times T_{broadcast} + T_{barrier})$

where p is the number of processes. We have measured the parameters in the above equation on Sunwulf,

$$T_{boradcast} \approx 0.12 + p \times 0.23(ms)$$
$$T_{send} = T_{recv} \approx 0.08 + (0.00003 \times N)(ms)$$
$$T_{barrier} \approx p \times 0.39(ms)$$

The computation time can be written as

$$T_c = \frac{W(N) \times t_c}{p}$$

where t_c is the time of one unit computation, we have measured $t_c \approx 3.1 \times 10^{-5}(ms)$. Based on these analyses and parameters, according to Corollary 2, we predict the scalability is $\psi(C, C') = T_o / T_o'$, where N is constrained by the isospeed-efficiency condition.

Based on the case of two nodes, our prediction result for the required N to keep constant speed-efficiency at 0.3 is as following,

Table 6 Predicted required rank

Nodes	4	8	16	32
N (predicition)	492	928	1683	3226

Thus, the predicted scalability is

Table 7 Predicted scalability of GE on Sunwulf

$\psi(C_2, C_4)$	$\psi(C_4, C_8)$	$\psi(C_8, C_{16})$	$\psi(C_{16}, C_{32})$
0.413	0.267	0.316	0.275

We can see that the predicted scalability is close to our measured scalability, which also verifies the isospeed-efficiency scalability metric works well.

5. Conclusion and future work

In this study, we propose an isospeed-efficiency scalability metric for general distributed environment. The proposed metric contains the homogenous isospeed scalability[10] as a special case. We first introduce a new concept of marked speed to describe the computational capability of computing systems. Based on the marked speed concept, we present the isospeed-efficiency scalability for heterogeneous computing. We then analyze the new scalability metric in theory and derive useful formulas for calculating and predicting this scalability. We have conducted experiments to verify these formulae in a heterogeneous computing environment. These results match the analytical results well and show that the proposed isospeed-efficiency scalability metric is appropriate for a general scalable computing environment, homogeneous or heterogeneous, tightly coupled or widely distributed.

Marked speed is a simple but effective metric to reflect the computational capability of general distributed systems. In future, we plan to extend the single parameter marked speed to multi-parameter marked performance that has several parameters to describe the full capability of a computing system, to provide users more options for their needs.

We have demonstrated that the scalability can be predicted based on derived formulas. We plan to explore the possibility of extending the prediction of scalability into system support so that the scalability can be predicted automatically or semi-automatically.

6. Acknowledgements

This research was supported in part by national science foundation under NSF grant SCI-0504291, CNS-0406328, ACI-0305355, EIA-0224377, and ANI-0123930.

7. References

[1] O. Beaumont, V. Boudet, F. Rastello and Y. Robert, "Matrix Multiplication on Heterogeneous Platforms", *IEEE Trans. Parallel Distributed Systems*, Vol. 12, No. 10, pp.1033-1051, 2001.

[2] D. Culler, J. Singh and A. Gupta, *Parallel Computer Architecture: A Hardware/Software Approach*, Morgan Kaufmann Publishers, 1999.

[3] V. Kumar, A. Grama, A. Gupta and G. Karypis, *Introduction to Parallel Computing: Design and Analysis of Parallel Algorithms*, 1994.

[4] K. Hwang and Z. Xu, *Scalable Parallel Computing*, McGraw – Hill, 1998.

[5] P.P. Jogalekar and C.M. Woodside, "Evaluating the Scalability of Distributed Systems", *IEEE Trans. on Parallel and Distributed Systems*, Vol. 11, No. 6, pp.589-603, 2000.

[6] A. Kalinov and A. Lastovetsky, "Heterogeneous Distribution of Computations While Solving Linear Algebra Problems on Networks of Heterogeneous Computers", *Journal of Parallel and Distributed Computing*, Vol. 61, No. 4, pp.520-535, 2001.

[7] L. Pastor and J.L. Bosque, "An Efficiency and Scalability Model for Heterogeneous Clusters". *IEEE International Conference on Cluster Computing*, pp.427-434, 2001.

[8] X.H. Sun, "Scalability versus Execution Time in Scalable Systems", *Journal of Parallel and Distributed Computing*, Vol. 62, No. 2, pp.173 – 192, 2002.

[9] X.H. Sun and L.M. Ni, "Scalable Problems and Memory-Bounded Speedup", *Journal of Parallel and Distributed Computing*, Vol. 19, pp.27-37, 1993.

[10] X.H. Sun and D. Rover, "Scalability of Parallel Algorithm – Machine Combinations", *IEEE Trans. Parallel Distributed Systems*, Vol. 5, pp.599 – 613, 1994.