2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum

Speedup for Multi-Level Parallel Computing

Shanjiang Tang[1], Bu-Sung Lee[1,2], Bingsheng He[1]
[1]School of Computer Engineering
Nanyang Technological University
{*stang5, ebslee, bshe*}*@ntu.edu.sg*
[2]Service Platform Lab
HP Labs Singapore
francis.lee@hp.com

***Abstract*—This paper studies the speedup for multi-level parallel computing. Two models of parallel speedup are considered, namely, fixed-size speedup and fixed-time speedup. Based on these two models, we start with the speedup formulation that takes into account uneven allocation and communication latency, and gives an accurate estimation. Next, we propose a high-level abstract case with providing a global view of possible performance enhancement, namely *E-Amdahl's Law* for fixed-size speedup and *E-Gustafson's Law* for fixed-time speedup. These two laws demonstrate seemingly opposing views about the speedup of multi-level parallel computing. Our study illustrates that they are not contradictory but unified and complementary. The results lead to a better understanding in the performance and scalability of multi-level parallel computing. The experimental results show that *E-Amdahl's Law* can be applied as a prediction model as well as a guide for the performance optimization in multi-level parallel computing.**

***Keywords*-E-Amdahl's Law, E-Gustafson's Law, Multi-Level Parallel Computing**

I. Introduction

High performance computing systems continue to grow rapidly over the past few years. The rapid development of multi/many-core technology in recent years has made it a denser parallel architecture (i.e. more processing elements per physical node) [1]. For instance, more than 80% of the systems of the Top500 supercomputers [2] belong to the multi/many-core processor family. That is, most systems have featured a hierarchical hardware design: computing nodes with multi/many-core CPUs are connected via network infrastructure. They have evolved to support parallelism of multiple levels (e.g., at levels of physical machines, CPUs and cores), in contrast with a single level of physical machines.

To fully exploit the potential of multi-level parallelism of hardware architecture, it is natural to employ the multi-level parallel computing paradigm which takes processes for coarse-grained parallelism across the nodes and threads for fine-grained parallelism within the node at the same time [3]. For example, it is quite suitable for a cluster of SMP nodes that MPI is needed for parallelism across nodes and OpenMP can be used to exploit the parallelism within a node [4].

Speedup has been a classical metric to measuring the performance and scalability of parallel processing [5]. It is generally defined as sequential execution time over parallel execution time. There are two well-known speedup models, *Amdahl's Law* [6] and *Gustafson's Law* [7]. *Amdahl's Law* models the speedup of parallelized implementations under the assumption that the problem size remains the same when parallelized. In contrast, Gustafson's Law models the speedup involving arbitrarily large data sets can be efficiently parallelized.

Both speedups captured by *Amdahl's Law* and *Gustafson's Law* are indeed based on the single-level parallelism, whereas multi-level parallelism is not considered. In comparison to single-level parallelism, multi-level parallelism is more complicated, involving a nested parallelism with granularity from coarse to fine. Neither *Amdahl's Law* nor *Gustafson's Law* is able to differentiate the varying granularity of multi-level parallel computing, from coarse-grained parallelism to fine-grained parallelism. To evaluate the speedup for the multi-level parallel computing, we need to combine speedups at different levels of parallelism. That motivates us to study speedup for multi-level parallel computing.

In this paper, we propose a general multi-level parallelism model. Based on this model, the fixed-size speedup and fixed-time speedup are studied. With both uneven workload allocation and communication latency considered, the formulations of generalized fixed-sized speedup and fixed-time speedup are derived, giving accurate performance estimations for different applications. In order to facilitate model the speedup of multi-level parallelism, we propose a fixed-size speedup named *E-Amdahl's Law* and a fixed-time speedup named *E-Gustafson's Law*, with assumptions that the communication overhead is zero and the workload only consists of sequential and perfectly parallel portion. Those models can be viewed as extensions of *Amdahl's Law* and *Gustafson's Law* respectively to multi-level parallel computing.

We further study the implications of *E-Amdahl's Law* and *E-Gustafson's Law* on performance optimization of multi-level parallel computing. *E-Amdahl's Law* indicates that if the degree of parallelism at the first level is not large, increasing the degree of parallelism at the second level will not significantly increase the overall performance. It is important but easily ignored by users during the programming and optimization of the multi-level computing. For example, in multi-GPU programming [8], programmers often focus most of their attentions on optimizing intra-GPU parallelism, since they generally regard the intra-GPU parallelism as the key part for performance optimization. Thus, the optimization work of parallelism across different GPUs might be neglected. On the other hand, *E-Gustafson's Law* suggests that the speedup of multi-level parallel computing is unbounded. While the two laws look conflictive, they are unified and complementary with each other, and reflect different viewpoints and different types of applications.

978-0-7695-4676-6/12 $26.00 © 2012 IEEE
DOI 10.1109/IPDPSW.2012.72

Since *E-Gustafson's Law* and *E-Amdahl's Law* are unified, we take *E-Amdahl's Law* as a case study to evaluate the performance of NAS parallel benchmark with multi-zone version [14]. The results show that the estimated speedup based on *E-Amdahl's Law* is much more accurate than that with *Amdahl's Law* for multi-level parallel computing. Moreover, *E-Amdahl's Law* always gives out the upper bound for the speedup. Thus, *E-Amdahl's Law* can be used as a prediction model as well as a guide to users on performance optimization of multi-level parallel computing.

This paper is organized as follows. Section II reviews the related work. Section III presents the multi-level parallelism model, followed by the description of the generalized multi-level parallel speedups. The high-level abstract multi-level parallel speedups are presented in Section V. Section VI presents the performance evaluation. Section VII concludes the paper and gives out the future work.

II. Related Work

Speedup is a key performance metric for parallel computing. Two basic definitions of speedup are available, namely, *absolute speedup* and *relative speedup* [9]. The *absolute speedup* refers to how much faster a problem can be solved with N processors by comparing the best sequential algorithm with the parallel algorithm under consideration. The *relative speedup* is defined as the ratio of elapsed time of the parallel algorithm on one processor to elapsed time of the parallel algorithm on N processors. It focuses on the inherent parallelism of the parallel algorithm under consideration. Moreover, there are other kinds of speedup definitions such as *real speedup* and *asymptotic speedup*. All types of speedup mentioned, together with their advantages and disadvantages, are surveyed in [16]. However, all those definitions of speedup are based on the the single-level parallelism.

Assuming that the problem size is fixed, *Amdahl's Law* [6] was proposed. It describes how much performance enhancement can be achieved when increasing the number of processing elements. It pessimistically indicates that the maximum speedup is bounded by the sequential fraction of the workload and massively parallel processing may not obtain high performance. Later, *Gustafson's Law* [7] was proposed. It is concerned with the fixed-time scenario and interested in the scale of a problem that can be handled within a given time. It optimistically suggests that the speedup is unbounded and increases linearly with the number of processing elements available. The above laws are simple and have great influences on the performance optimizations of parallel computing. Sun and Ni's work [5], [11] has also described other speedup models for memory-bounded scenarios. However, all the previous works of speedup model are implicitly based on the single-level parallelism, which are unsuitable for the multi-level parallelism. In contrast, our work in this paper extends the fixed-size and fixed-time speedup models for the multi-level parallelism, giving both the generalized formulations as well as high-level abstract formulations.

III. Model and Motivation

In this section, we present the concept of multi-level parallel computing and the motivation for new speedup models.

A. Multi-level Parallel Computing

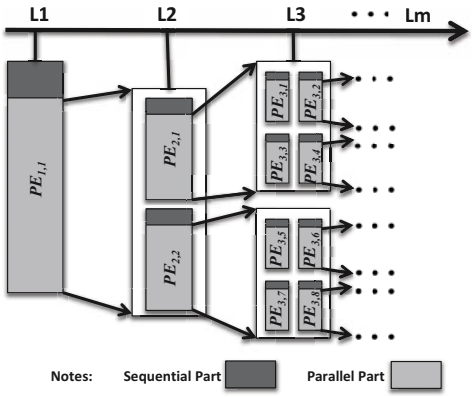


Fig. 1: Multi-level parallelism model.

Figure 1 illustrates a general parallelism model for the multi-level parallel computing. It is a nested parallelism from coarser granularity to finer granularity. The number of parallelism levels is $m(m \geq 1)$. We call each processing part as a parallelism unit. Let $PE_{i,j}$ denote the parallelism unit with the id of j at the i^{th} level. The model assumes that a parallel program can be separated into two parts: *sequential part* and *parallel part*. For the parallel part, we parallelize it in multiple levels of parallelism. The parallel part of the coarse-grained parallelism can be further parallelized with fine-grained parallelism implementations. For example, we can write a two-level program with MPI/OpenMP programming paradigm for SMP clusters. The first-level parallelism (*L1*) is a process-level parallelism with MPI across the compute nodes. The second-level parallelism (*L2*) is a thread-level parallelism using OpenMP within a process. More levels of parallelism can also be considered, e.g., instruction-level parallelism from the compiler aspect.

B. A Motivating Example

Multi-level parallelization has been widely adopted in many applications. To motivate the importance of extending the traditional concept and formula of speedup for the multi-level parallelism, we take as a case study benchmark LU-MZ [14], implemented with MPI/OpenMP programming paradigm.

We compare the performance estimation using *Amdahl's Law*[1] and with our proposed *E-Amdahl's Law*, as illustrated in Figure 2. The details on *E-Amdahl's Law* is presented in Section V. The performance of the benchmark is measured on a SMP cluster consisting of 8 compute nodes, each with two quad-core chips. Let p denote the number of processes

[1]Amdahl's Law: *speedup*$= \frac{1}{1-F+\frac{F}{N}}$, where F is the parallel fraction of the workload, N is the number of processors used.

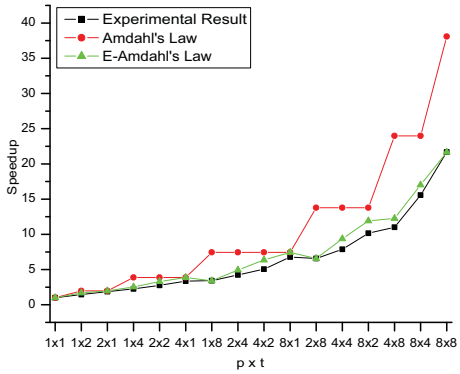


Fig. 2: Experimental and estimated speedups for NAS multi-level parallel computing benchmark LU-MZ, implemented with hybrid MPI/OpenMP programming paradigm. p denotes the number of processes, t denotes the number of threads per process. We estimate the speedup with *E-Amdahl's Law* by using Formula (17). We estimate that $\alpha = 0.9892, \beta = 0.8161$ with *Algorithm 1* (in Section VI(A)), where β is the parallel fraction of the thread-level parallelism.

and t denote the number of threads per process. With the perspective of the number of processors (i.e. CPU cores) used for the multi-level parallelism, we estimate the speedup based on *Amdahl's Law* by using the formula: $\frac{1}{1-\alpha+\frac{\alpha}{p\times t}}$, where α is the parallel portion at the process-level parallelism. Note that *Amdahl's Law* cannot well estimate the speedup for the benchmark with multi-level parallelism. Particularly, *Amdahl's Law* is unable to differentiate the coarse-grained and fine-grained parallelism. For example, there is no difference in speedup when $p \times t = 1 \times 8, 2 \times 4, 4 \times 2, 8 \times 1$ using *Amdahl's Law*. Moreover, the estimated speedup of *Amdahl's Law* becomes more inaccurate when the number of threads per process (t) increase. For instance, the estimation error at point $p \times t = 8 \times 8$ is much larger than that of $p \times t = 8 \times 4$. In contrast, *E-Amdahl's Law* is able to well distinguish the coarse-grained and fine-grained parallelism, and estimate the speedup for the multi-level parallelism. Specifically, in this example, the *average ratio of estimation error*[2] for *Amdahl's Law* is 55%. In comparison, the *average ratio of estimation error* for *E-Amdahl's Law* is 11%. This significant estimation error difference motivates us to model the speedup of multi-level parallelism.

IV. Generalized Multi-Level Parallel Speedup

The parallelism within an application can be characterized in different ways [5]. Two major degradation factors for effecient parallelism are considered in the paper, *uneven allocation* (load unbalance) and *communication latency*. Both degradations should be well considered in order to get an accurate estimation. The uneven allocation is measured by *degree of parallelism* [5].

[2]Average ratio of estimation error $= \frac{1}{n}\sum\frac{|R-E|}{R}$, where R is the experimental result, E is the estimated result, n is the number of testings.

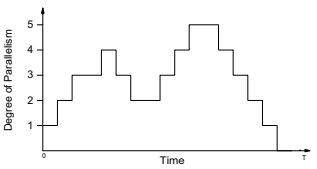


Fig. 3: Parallelism profile of a hypothetical application.

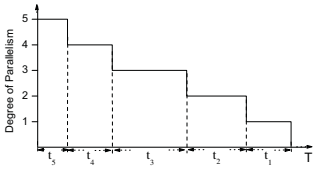


Fig. 4: Shape of the application.

Definition 1. *The degree of parallelism is an integer which indicates the number of processing elements in a parallelism level that are busy during the execution of the program, given an unbounded number of available processing elements (e.g. computing nodes, CPUs).*

Figure 3 shows a graph of the degree of parallelism in a parallelism level over the execution time for a hypothetical parallel application. We refer to this as the *parallelism profile*. It can be rearranged to form the shape (See Figure 4) of the application execution by gathering the time taken at each degree of parallelism [10].

In Figure 1, we assume that parallelism units at the same level are identical. Thus we only need to consider one multi-level parallelism path (e.g. $\sum PE_{i,1}, (1 \le i \le m)$) for the whole computation. Let W be the whole amount of computation (work) for an application. Let $p(i)$ be the number of processing elements used to execute the parallelism units at the i^{th} level, which are spawned from the same PE. For example, in Figure 1, we have $p(1) = 1, p(2) = 2, p(3) = 4$. Let P represent the whole processing elements. Formally, the speedup for a multi-level hardware architecture with P processing elements and with the total amount of work W is defined as

$$SP_P(W) = \frac{T_1(W)}{T_P(W)}, \tag{1}$$

where $T_i(W)$ is the time needed to complete W amount of work on i processing elements. Let $W_{i,j}$ be the amount of work for each node at the i^{th} level with the degree of parallelism j, and m_i be the maximum degree of parallelism at the i^{th} level. Thus, $W = \sum_{j=1}^{m_1} W_{1,j}$. For each $PE_{i,i'}(1 \le i < m)$, the sequential portion of work $W_{i,1}$ is computed and the parallel portion of work $W_{i,j}(1 < j < m_i)$ is distributed to its low-level nodes. Thus, it has:

$$\sum_{j=2}^{m_i} \frac{W_{i,j}}{j} = \sum_{j=1}^{m_{i+1}} W_{i+1,j}, \qquad (1 \le i < m). \tag{2}$$

Let Δ be the computing capacity of each processing element, the execution time on a single processing element for the total work W will be

$$T_1(W) = \frac{W}{\Delta}. \tag{3}$$

The multi-level parallelism is a recursive master-slave execution process. When one node $PE_{i,i'}(1 \le i < m)$ completes

its sequential execution, it will wait until its parallel portion is solved by its low-level nodes. In contrast, $PE_{m,i'}$ at the bottom level will not dispatch its parallel portion, but instead it completes its execution of sequential and parallel portions by itself. By the definition of degree of parallelism, any two subtasks (e.g. $W_{i,j}$ and $W_{i,k}$, $j \neq k$) at the i^{th} level with different degrees of parallelism cannot be solved simultaneously. The total execution time on a multi-level hardware platform with an unbounded number of processing elements available will be

$$T_\infty(W) = \frac{W_{1,1} + W_{2,1} + \cdots + W_{m-1,1}}{\Delta} + \sum_{j=1}^{m_m} \frac{W_{m,j}}{j\Delta}. \quad (4)$$

Thus, the speedup will be

$$\begin{aligned} SP_\infty(W) &= \frac{T_1(W)}{T_\infty(W)} \\ &= \frac{\frac{W}{\Delta}}{\frac{W_{1,1}+W_{2,1}+\cdots+W_{m-1,1}}{\Delta} + \sum_{j=1}^{m_m} \frac{W_{m,j}}{j\Delta}} \\ &= \frac{W}{\sum_{i=1}^{m-1} W_{i,1} + \sum_{j=1}^{m_m} \frac{W_{m,j}}{j}} \\ &= \frac{\sum_{j=1}^{m_1} W_{1,j}}{\sum_{i=1}^{m-1} W_{i,1} + \sum_{j=1}^{m_m} \frac{W_{m,j}}{j}}. \end{aligned} \quad (5)$$

When the communication overhead is considered and the number of processing elements for each node in each parallelism level is finite, the speedup will be smaller than the speedup SP_∞. For each node $PE_{i,i'}$ at the i^{th} level, if the number of its parallel processing elements is $p(i)$ and $p(i) < j$, then some processing elements have to do $\frac{W_{i,j}}{j}\lceil\frac{j}{p(i)}\rceil$ work and the remaining ones will do $\frac{W_{i,j}}{j}\lfloor\frac{j}{p(i)}\rfloor$ work. We assume that at each level, the allocation of work strictly follows an order of the ids of PE from small to large and distributes the work with $\frac{W_{i,j}}{j}\lceil\frac{j}{p(i)}\rceil$ first and later the work with $\frac{W_{i,j}}{j}\lfloor\frac{j}{p(i)}\rfloor$. Then for the nodes in the path $\sum PE_{i,1}, (1 \leq i \leq m)$, it holds:

$$\sum_{j=2}^{m_i} \frac{W_{i,j}}{j}\lceil\frac{j}{p(i)}\rceil = \sum_{j=1}^{m_{i+1}} W_{i+1,j}, \quad (1 \leq i < m). \quad (6)$$

The execution time on the multi-level hardware architecture, with work W and limited number of processing elements, is

$$T_P(W) = \frac{W_{1,1} + W_{2,1} + \cdots + W_{m-1,1}}{\Delta} + \sum_{j=1}^{m_m} \frac{W_{m,j}}{j\Delta}\lceil\frac{j}{p(m)}\rceil. \quad (7)$$

Hence, the speedup is

$$\begin{aligned} SP_P(W) &= \frac{T_1(W)}{T_P(W)} \\ &= \frac{\frac{W}{\Delta}}{\frac{W_{1,1}+W_{2,1}+\cdots+W_{m-1,1}}{\Delta} + \sum_{j=1}^{m_m} \frac{W_{m,j}}{j\Delta}\lceil\frac{j}{p(m)}\rceil} \\ &= \frac{W}{\sum_{i=1}^{m-1} W_{i,1} + \sum_{j=1}^{m_m} \frac{W_{m,j}}{j}\lceil\frac{j}{p(m)}\rceil} \\ &= \frac{\sum_{j=1}^{m_1} W_{1,j}}{\sum_{i=1}^{m-1} W_{i,1} + \sum_{j=1}^{m_m} \frac{W_{m,j}}{j}\lceil\frac{j}{p(m)}\rceil}. \end{aligned} \quad (8)$$

Another important degradation factor of performance is communication latency. In contrast to unbalance workload, communication latency is communication network dependent [11], e.g, routing schemes and switching techniques, etc. Let $Q_P(W)$ be the communication overhead of the multi-level parallel computing when P processing elements of the multi-level hardware architecture are used to complete P amount of work. $Q_P(W)$ depends on lots of factors including the communication pattern, message sizes of the application, system-dependent communication latency, etc. Assuming the degree of parallelism is not affected by the communication overhead, the general speedup is

$$SP_P(W) = \frac{\sum_{j=1}^{m_1} W_{1,j}}{\sum_{i=1}^{m-1} W_{i,1} + \sum_{j=1}^{m_m} \frac{W_{m,j}}{j}\lceil\frac{j}{p(m)}\rceil + Q_P(W)}. \quad (9)$$

Note that all above derivations for speedup of the multi-level parallel computing are indeed for the fixed problem size, or the fixed workload. The speedup emphasizes the time reduction of a given problem. This speedup formulation is called *fixed-size speedup* [11]. Equation (9) is the general speedup formula for fixed-size speedup. It is suitable for many applications whose sizes cannot be scaled. In contrast, another speedup model called *fixed-time speedup* [7], where the workload of application is scaled up with the number of processing elements available. One typical application for this is weather broadcasting [12]. Given more computation power, we may not want to get the result earlier. Instead, we may want to increase the problem size by adding more relevant factors into the weather model and obtain a more accurate solution, which in turn gives a more precise forecast. Moreover, it assumes that the workload scaling occurs only at the parallel portion. Let $W^{'}$ be the total amount of scaled work. Let $W^{'}_{i,j}$ be the amount of scaled work for each node at the i^{th} parallelism level with the degree of parallelism j, and $m^{'}_i$ be the maximum degree of parallelism of the scaled workload with $p(i)$ processing elements available at the i^{th} level. Then it has: $W^{'} = \sum_{j=1}^{m^{'}_1} W^{'}_{1,j}$ and $W_{i,1} = W^{'}_{i,1}, (1 \leq i \leq m)$. For the nodes in the path $\sum PE_{i,1}, (1 \leq i \leq m)$, it holds:

$$\sum_{j=2}^{m^{'}_i} \frac{W^{'}_{i,j}}{j}\lceil\frac{j}{p(i)}\rceil = \sum_{j=1}^{m^{'}_{i+1}} W^{'}_{i+1,j}, \quad (1 \leq i < m). \quad (10)$$

$$\sum_{j=2}^{m_i} W_{i,j} = \sum_{j=1}^{m_{i+1}} W_{i+1,j}, \quad (1 \le i < m). \tag{11}$$

In the lowest level, $i = m$, in order to keep the same turnaround time as the sequential version, its scaled workload must hold

$$\sum_{j=1}^{m_m} W_{m,j} = \sum_{j=1}^{m_m^{'}} \frac{W_{m,j}^{'}}{j} \lceil \frac{j}{p(m)} \rceil, \quad (i = m). \tag{12}$$

Thus, based on Equations (10), (11), (12), the generalized speedup formula for fixed-time speedup is

$$\begin{aligned} SP_P(W^{'}) &= \frac{T_1(W^{'})}{T_p(W^{'})} \\ &= \frac{\frac{W^{'}}{\Delta}}{\frac{W_{1,1}^{'}+W_{2,1}^{'}+\cdots+W_{m-1,1}^{'}}{\Delta} + \sum_{j=1}^{m_m^{'}} \frac{W_{m,j}^{'}}{j\Delta} \lceil \frac{j}{p(m)} \rceil + Q_P(W^{'})} \\ &= \frac{W^{'}}{\sum_{i=1}^{m-1} W_{i,1}^{'} + \sum_{j=1}^{m_m^{'}} \frac{W_{m,j}^{'}}{j} \lceil \frac{j}{p(m)} \rceil + Q_P(W^{'})} \\ &= \frac{W^{'}}{\sum_{i=1}^{m-1} W_{i,1} + \sum_{j=1}^{m_m} W_{m,j} + Q_P(W^{'})} \\ &= \frac{W^{'}}{W + Q_P(W^{'})}. \end{aligned} \tag{13}$$

V. High-level abstract Multi-Level Parallel Speedup

The generalized speedup formulations have been proposed for the fixed-size speedup model and the fixed-time speedup model. The formulations consider the performance degradation due to uneven allocation and communication latency. The generalized speedups are application dependent and much closer to the real speedup. However, it is very complicated and difficult to understand and to have a global view of possible performance gains since they contain lots of detailed information for each application. In this section, we make some assumptions to have a high-level speedup model, focusing on the essential characteristic of the multi-level parallel computing. We assume that the communication overhead cost is zero (i.e. $Q_P = 0$), and that the workload at each parallelism level only consists of two portions, a sequential portion and a perfectly parallel portion. Thus, $W_{i,j} = W_{i,j}^{'} = 0$, for $1 \le i \le m$, $j \ne 1$ and $j \ne p(i)$. We call the high-level abstract fixed-size speedup as *E-Amdahl's Law*, as an extension of *Amdahl's Law* and the high-level abstract fixed-time speedup as *E-Gustafson's Law*, as an extension of *Gustafson's Law* for the multi-level parallel computing. In the following, let $f(i)$ be the portion of the workload at the i^{th} level that can be parallelized, and $sp(i)$ denote the multi-level speedup at the i^{th} level.

A. E-Amdahl's Law

The multi-level speedup can be viewed as the relative computing capacity of a multi-level hardware platform with respect to an uniprocessor. To figure out the high-level abstract fixed-size speedup for the multi-level parallel computing model, a bottom-up method is adopted. That is, we first compute the lowest-level result and then deduce the nearest upper-level one based on current level result. Such process continues until the final result(first-level speedup) is obtained. Specifically, it is as follows:

1). When $i = m$, it is the bottom level. Since there is no more level below it, its speedup can be obtained directly in terms of *Amdahl's Law*, that is:

$$sp(i) = \frac{1}{1 - f(m) + \frac{f(m)}{p(m)}}, \quad (i = m). \tag{14}$$

2).When $1 \le i < m$, the speedup at the i^{th} level directly depends on the $(i+1)^{th}$ level. It is:

$$sp(i) = \frac{1}{1 - f(i) + \frac{f(i)}{p(i)sp(i+1)}}, \quad (1 \le i < m). \tag{15}$$

In summary, we obtain *E-Amdahl's Law* with the following formula:

$$sp(i) = \begin{cases} \frac{1}{1-f(m)+\frac{f(m)}{p(m)}} & (i = m) \\ \frac{1}{1-f(i)+\frac{f(i)}{p(i)sp(i+1)}} & (1 \le i < m). \end{cases} \tag{16}$$

Generally, we often consider the multi-level parallel computing with two levels($m = 2$). Let's consider clusters of SMP nodes. It often takes hybrid MPI/OpenMP programming model [13], which uses MPI for parallelism across the nodes and OpenMP for parallelism within a node. Suppose it is one process per node and the number of processes(nodes) is p. For each process(node), we assume the number of threads is t. In the process level, the fraction of computation that can concurrently execute is $\alpha (0 \le \alpha \le 1)$. The parallelizable portion of computation in the thread level is $\beta (0 \le \beta \le 1)$. In terms of Formula (16), we have the multi-level speedup as follows:

$$\widehat{sp}(\alpha, \beta, p, t) = sp(i = 1) = \frac{1}{1 - \alpha + \frac{\alpha(1-\beta+\frac{\beta}{t})}{p}}. \tag{17}$$

Some properties can be attained for Formula (17):

a). $\widehat{sp}(\alpha, \beta, 1, 1) = 1$: The sequential condition holds.

b). $\widehat{sp}(\alpha, \beta, p, 1) = \frac{1}{1-\alpha+\frac{\alpha}{p}}$: It turns to be a traditionally single-level *Amdahl's Law* with parallelizable percent of α when $t = 1$.

c). $\widehat{sp}(\alpha, \beta, 1, t) = \frac{1}{1-\alpha\beta+\frac{\alpha\beta}{t}}$: It becomes a single-level *Amdahl's Law* with parallelizable portion of $\alpha\beta$ when $p = 1$.

Figure 5 presents fixed-size speedup curves under *E-Amdahl's Law* of Equation (17) when $m = 2$. The x-axis denotes the number of processes (p). The y-axis gives the speedup for the multi-level parallel computing. Curves within

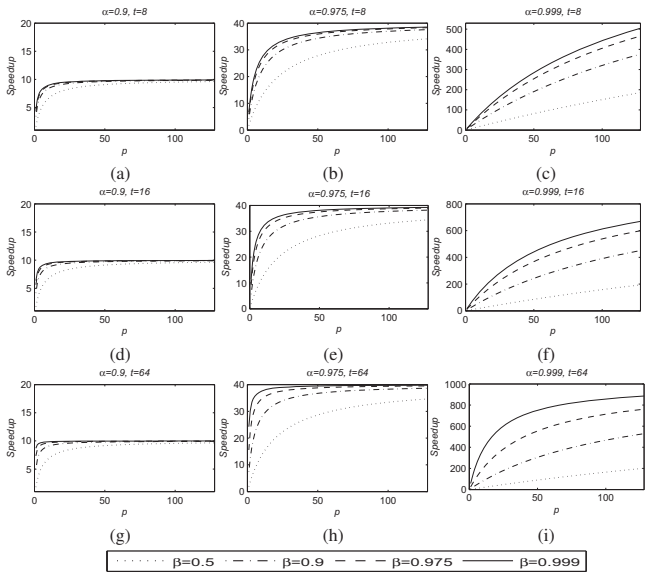


Fig. 5: Speedup under *E-Amdahl's Law*. The graphs from left to right in each row give the variation of speedup when increasing the value of α. The graphs from top to down in each column present the variation of speedup when we increase the number of threads (t). The curves within each graph show speedups under different values of β.

each graph show speedups under different values of β. The graphs from left to right in each row give the variation of speedup when increasing the value of α. The graphs from top to down in each column present the variation of speedup when we increase the number of threads (t).

Result 1: For fixed-size applications, the results of *E-Amadahl's Law* indicate that exploring parallelism at each level of the multi-level parallel computing is crucial for performance improvement. However, if the degree of parallelism (α) at the first (process) level is not large, increasing the degree of parallelism at the second (thread) level cannot have a significant contribution to the whole performance improvement. For example, in Figure 5(a), the curves are very close to each other when increasing β from 0.5 to 0.999 in the case of $\alpha = 0.9$. In contrast, it will have a significant performance improvement when increasing the parallelism β at the second level if the parallelism (α) at the first level is large enough. Considering $\alpha = 0.999, p = 100, t = 8$ in Figure 5(c) for instance, the performance is significantly improved when the parallelism β at the second level increases from 0.5 to 0.999.

Result 2: The maximum fixed-size speedup is bounded by the degree of parallelism at the first level. For example, if $\alpha = 0.9$, its maximum speedup is 10. As shown in Figure 5(a),5(d),5(g), no matter how to increase the number of processes (p), the number of threads (t), and the degree of parallelism β at the second level, the speedup infinitely approximates to 10.

B. E-Gustafson's Law

The fixed-time speedup can be regarded as the ratio of scaled workload in parallel hardware to the workload of uniprocessor. A bottom-up method is adopted to get the high-level abstract fixed-time speedup for the multi-level parallel computing model,. We first figure out the fixed-time speedup at the bottom level and then deduce the nearest upper-level one based on current level result. The process continues until the final result (first-level speedup) is obtained. That is,

1). When $i = m$, it is the bottom level. Its speedup can be obtained directly in terms of *Gustafson's Law*[3]. that is:

$$sp(i) = 1 - f(m) + f(m)p(m), \quad (i = m). \tag{18}$$

2).When $1 \le i < m$, the speedup(workload) at the i^{th} level directly depends on the $(i+1)^{th}$ level. Note that Equation (18) can be viewed as the normalized scaled workload when the workload of uniprocessor is 1. Then it holds that

$$sp(i) = 1 - f(i) + f(i)p(i)sp(i+1), \quad (1 \le i < m). \tag{19}$$

Therefore, we obtain *E-Gustafson's Law* with the following formula:

$$sp(i) = \begin{cases} 1 - f(m) + f(m)p(m) & (i = m) \\ 1 - f(i) + f(i)p(i)sp(i+1) & (1 \le i < m). \end{cases} \tag{20}$$

In general, it is common to have two levels of parallelism for the multi-level parallel computing$(m = 2)$. Given clusters of SMP nodes, for example, it often takes hybrid MPI/OpenMP programming model [13]. We assume the number of processes is p and the number of threads per process is t. The portion of computation at the first(process) level that can be parallelized is $\alpha(0 \le \alpha \le 1)$ and that at the second(thread) level is $\beta(0 \le \beta \le 1)$. According to Formula (20), the multi-level fixed-time speedup is:

$$\widehat{sp}(\alpha, \beta, p, t) = sp(i = 1) = 1 - \alpha + (1 - \beta + \beta t)\alpha p. \tag{21}$$

There are some properties for Formula (21) as follows:

a). $\widehat{sp}(\alpha, \beta, 1, 1) = 1$: The sequential condition holds.

b). $\widehat{sp}(\alpha, \beta, p, 1) = 1 - \alpha + \alpha p$: It turns out to be a traditionally single-level *Gustafson's Law* with parallelizable percent of α when $t = 1$.

c). $\widehat{sp}(\alpha, \beta, 1, t) = 1 - \alpha\beta + \alpha\beta t$: It becomes a single-level *Gustafson's Law* with parallelizable portion of $\alpha\beta$ when $p = 1$.

Equation (21) indicates that there is a positive linear relationship between the speedup and performance factor $\theta \in \{\alpha, \beta, p, t\}$. Figure 6 gives fixed-time speedup curves under *E-Gustafson's Law* of Equation (21). From the fixed-time point of view, *E-Gustafson's Law* implies that,

[3]Gustafson's Law: *speedup* $= 1 - F + F \times N$, where F is the parallel fraction of the workload, N is the number of processors used.

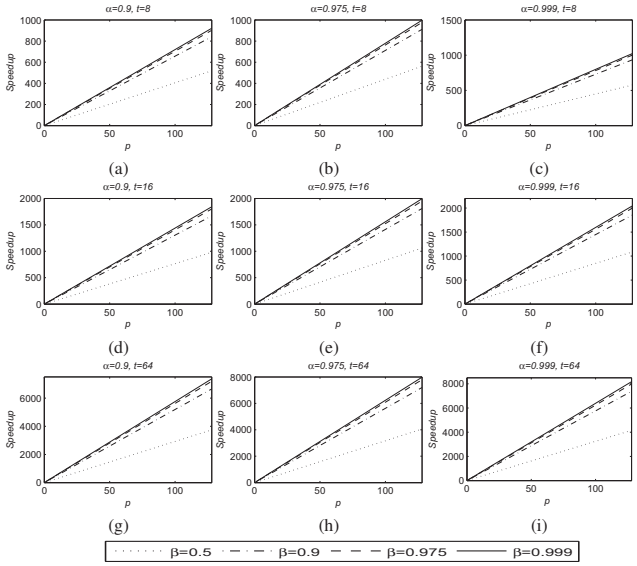


Fig. 6: Speedup under *E-Gustafson's Law*. The graphs from left to right in each row give the variation of speedup when increasing the value of α. The graphs from top to down in each column present the variation of speedup when we increase the number of threads (t). The curves within each graph show speedups under different values of β.

Result 3: For applications with scaled workload, the maximum speedup of the multi-level parallel computing is unbounded. For instance, in Figure 6(a), the speedup is scaled up linearly with p.

Note that *E-Amdahl's Law*(Result 2) and *E-Gustafson's Law*(Result 3) give totally opposite viewpoints towards the maximum speedup of the multi-level parallel computing. However, *E-Amdahl's Law* and *E-Gustafson's Law* are not conflictive but unified. They just consider the speedup from different perspectives. Indeed, *E-Gustafson's Law* is implicitly based on the assumption that the problem size is scaled and the parallel portion is unfixed, whereas *E-Amdahl's Law* supposes that the problem size is fixed and the parallel portion is unchanged. The equivalent proof is given in *APPENDIX A*.

VI. Experimental Evaluation

In the previous section, we have demonstrated the equivalence of *E-Amdahl's Law* and *E-Gustafson's Law*. Therefore, we take *E-Amdahl's Law* as a case study to evaluate the speedup model. In particular, we choose the NAS Parallel Benchmark (NPB) Multi-Zone (MZ) [14], [15] codes BT-MZ, SP-MZ and LU-MZ, with hybrid MPI+OpenMP programming model. They are executed on a Linux cluster consisting of eight compute nodes, each with two 3.0 GHz Intel Xeon quad-core chips and 16 GB of memory. We do experiments by setting one MPI process per compute node and increasing the number of OpenMP threads for each process from 1 up to 8.

In the following, we first present the argument estimation for *E-Amdahl's Law*, and then present the experimental results.

A. Argument Estimation

To evaluate the speedup based on *E-Amdahl's Law* of Formula (17), it needs to estimate the exact values of arguments α, β for a given application first. It is a challenging issue and no optimal solution is readily available, since it depends on lots of factors such as program implementation, input data and arguments, hardware platform, etc. In this paper, we give a possible argument estimation method in our experiment.

This method tries to estimate the values of α, β through the multi-level program execution. It figures out the estimated value of α, β by solving Equation (17) with sampling experimental results. This method can well contain various overhead costs such as process creation cost, communication and synchronization cost, etc. Its estimated result is much close to the actual value. The detailed estimation process is given by *Algorithm 1*.

Algorithm 1 *Argument Estimation for* α, β.

1: Execute the program k times in the multi-level parallel execution mode with the chosen parameters $(p_1, t_1), (p_2, t_2), ..., (p_k, t_k)$ respectively. Then it gets $(p_1, t_1, sp_1), (p_2, t_2, sp_2), ..., (p_k, t_k, sp_k)$.
2: Choose all possible combinations of two arrays (p_i, t_i, sp_i) and (p_j, t_j, sp_j) to figure out the value (α_s, β_s) based on Equation (17), where s denotes the s^{th} combination and $1 \leq i, j \leq k$ & $i \neq j$.
3: Check all possible pairs of value (α_s, β_s) to guarantee that the pair of estimated values is valid(i.e. $0 \leq \alpha_s \leq 1, 0 \leq \beta_s \leq 1$). Otherwise, discards it.
4: Collect all valid pairs of $(\alpha_i, \beta_i), (i = 1, 2, ..., k^{'})$, where $k^{'}$ represents the number of valid pairs. Remove the noise pairs by clustering with the guard condition: $|\alpha_i - \alpha_j| < \varepsilon$ & $|\beta_i - \beta_j| < \varepsilon$.
5: The exact value of α, β can thereby be estimated with the formula:

$$\begin{cases} \widehat{\alpha} = \frac{1}{\widehat{k}} \sum_{i=1}^{\widehat{k}} \alpha_i \\ \widehat{\beta} = \frac{1}{\widehat{k}} \sum_{i=1}^{\widehat{k}} \beta_i, \end{cases}$$

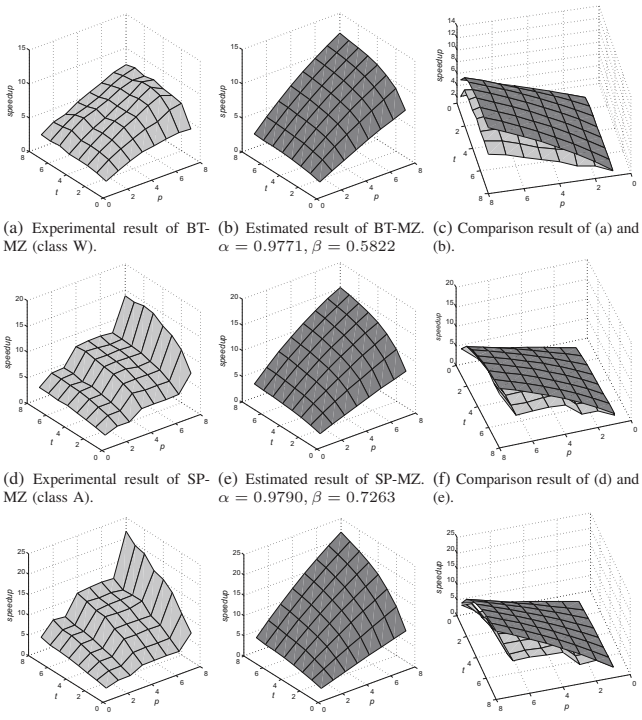
where $\widehat{k}$ is the number of clustered pairs.

Notable, the correct choice of p_i and t_i is very crucial for estimation. Particularly, we should avoid those pairs which may cause workload unbalance during the parallel execution. For example, for the benchmark SP-MZ, the suitable value of p_i and t_i should be $1, 2, 4, 8, 16, ...$, since the workload is well balanced in these cases. In contrast, it is unbalanced when p_i or t_i equals to $3, 7$.

B. Results Evaluation

The experimental and estimated speedup results are shown in Figure 7. To estimate the values of α, β with *Algorithm 1*, the values of p_i and t_i are set to be $1, 2, 4$ respectively, and $\varepsilon =$

0.01. The graphs in the first column present the experimental speedups. The estimated speedup results with *E-Amdahl's Law* are given in the second column. The comparison graphs of the corresponding experimental and estimated results are shown in the third column respectively.



(a) Experimental result of BT-MZ (class W). (b) Estimated result of BT-MZ. $\alpha = 0.9771, \beta = 0.5822$ (c) Comparison result of (a) and (b).

(d) Experimental result of SP-MZ (class A). (e) Estimated result of SP-MZ. $\alpha = 0.9790, \beta = 0.7263$ (f) Comparison result of (d) and (e).

(g) Experimental result of LU-MZ (class A). (h) Estimated result of LU-MZ. $\alpha = 0.9892, \beta = 0.8161$ (i) Comparison result of (g) and (h)

Fig. 7: Experimental and estimated speedup results for NAS Parallel Benchmarks of Multi-Zone Versions. p is the number of MPI processes spawned. t is the number of OpenMP threads for each process. The estimated speedup is based on the proposed *E-Amdahl's Law*.

Figures 7(a) $\sim$ 7(c) present speedup results for the application benchmark BT-MZ with class W[4]. It is a block tri-diagonal simulated CFD algorithm. The number of zones for class W is 4×4. However, the size of zones varies significantly, with a ratio of about 20 between the largest and smallest zone. This poses a problem for workload balancing. Its estimated coarse-grained parallelism portion is $\alpha = 0.9771$ and fine-grained parallelism portion is $\beta = 0.5822$. With *E-Amdahl's Law*, the upper bound speedup result can be estimated. The comparison result in Figure 7(c) indicates that the workload unbalance problem is becoming increasingly serious as the number of processes increases. In this scenario, *E-Amdahl's Law* can be used to evaluate the quality of the algorithm. Meanwhile, it can be a guidance to users on how much performance improvement space is available for performance optimization.

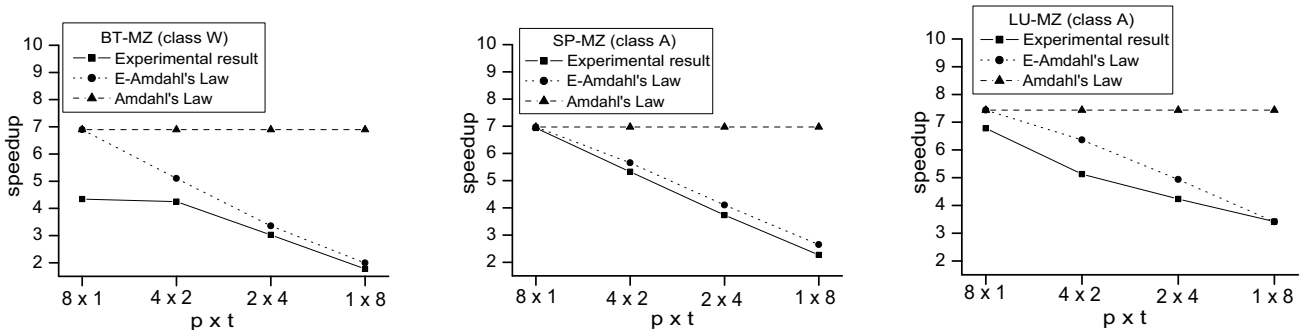
SP-MZ is a scalar penta-diagonal simulated CFD algorithm. LU-MZ is a lower-upper symmetric gauss-seidel simulated CFD algorithm. In contrast to BT-MZ, the partitioned zones are identical in size for the two algorithms. The workload should be more balanced than BT-MZ. The number of zones for class A is 4×4. The corresponding speedup results for each algorithm are shown in Figure 7(d) $\sim$ 7(f), 7(g) $\sim$ 7(i) respectively. The estimated parallelism portions are $\alpha = 0.9790, \beta = 0.7263$ for SP-MZ, and $\alpha = 0.9892, \beta = 0.8161$ for LU-MZ. The workloads are well balanced when the number of zones (e.g. 16) is divisible by the number of processes. The comparison results in Figure 7(f) and 7(i) show that the experimental results are very close to estimated results when $p = 1, 2, 4, 8$, indicating that the performance is fine and *E-Amdahl's Law* can be applied as a prediction model in these cases. However, the workload is not balanced (i.e. the number of zones cannot be evenly distributed among processes) when the number of processes p equals to $3, 5, 6, 7$. It leads to a negatively big impact on the overall parallel performance (as illustrated in Figure 7(d),7(g)). With *E-Amdahl's Law*, users can have an intuitive understanding of how much performance influence it has through comparisons of experimental and estimated results as shown in Figure 7(f) and 7(i).

C. Comparison on Estimated Speedup

Figure 8 shows the performance results of NPB-MZ benchmarks under different process-thread combinations with a fixed number of 8 CPUs. The experimental speedups as well as the estimated speedups based on *Amdahl's Law* and our proposed *E-Amdahl's Law* are illustrated. We estimate speedup based on *Amdahl's Law* with the formula: $\frac{1}{1-\alpha+\frac{\alpha}{p \times t}}$. The estimated speedup based on *E-Amdahl's Law* is figured out by using Formula (17). It can be noted that for *Amdahl's Law*, there is no any difference in estimated speedups for diverse combinations of $p \times t$ under a fixed number of processors. The reason is that, *Amdahl's Law* is based on the single-level parallelism. There is no concept of combined coarse-grained and fine-grained parallelism in the single-level parallelism. For *Amdahl's Law*, it treats all sizes of granularity of parallelism to be the same. However, the multi-level parallelism is a nested parallelism with the granularity of parallelism from coarse to fine. For the multi-level parallelism, *Amdahl's Law* is unable to differentiate its coarse-grained and fine-grained parallelism. The speedup results based on *Amdahl's Law* become more inaccurate when there are more number of processors used for fine-grained parallelism. For example, for benchmark SP-MZ, shown in Figure 8(b), when $p \times t = 8 \times 1, 4 \times 2, 2 \times 4, 1 \times 8$, the *ratio of estimation errors*[5] are $0.6\%, 31.0\%, 86.7\%, 207.5\%$ respectively. In contrast, *E-Amdahl's Law* can well identify the coarse-grained and fine-grained parallelism and accurately estimate the speedup for the multi-level parallelism. Note that the

[4]Class W refers to a type of problem size in NAS benchmark. It classifies the types of problem size into S, W, A, B, C, etc

[5]*Ratio of estimation error* $= \frac{|R-E|}{R}$, where R is the experimental result, E is the estimated result.



(a) $\alpha = 0.9771, \beta = 0.5822$ (b) $\alpha = 0.9790, \beta = 0.7263$ (c) $\alpha = 0.9892, \beta = 0.8161$

Fig. 8: Experimental and estimated speedups of NPB-MZ for different combinations of $p \times t$ under a given total number of 8 processors. The speedup based on *Amdahl's Law* is estimated with the formula $\frac{1}{1-\alpha+\frac{\alpha}{p\times t}}$. The speedup based on *E-Amdahl's Law* is estimated by using Formula (17).

speedup curves of *E-Amdahl's Law* in Figure 8 are very close to the experimental results. For the same example of benchmark SP-MZ, under speedup estimation of *E-Amdahl's Law*, the *ratio of estimation errors* are $0.6\%, 6.2\%, 9.8\%, 16.7\%$ respectively for $p \times t = 8 \times 1, 4 \times 2, 2 \times 4, 1 \times 8$. Moreover, for these sampling cases, the *average ratio of estimation errors* with *E-Amdahl's Law* for benchmarks BT-MZ, SP-MZ and LU-MZ are $25.5\%, 8.3\%, 13.1\%$ respectively. In contrast, the *average ratio of estimation errors* by using *Amdahl's Law* for benchmarks BT-MZ, SP-MZ and LU-MZ are $134.5\%, 81.5\%, 62.5\%$ respectively. Particularly, we can note that the estimated speedup curves of BT-MZ are a bit far from the experimental results in comparison to other two benchmarks based on *E-Amdahl's Law*. This is because the data size of zones for BT-MZ varies significantly, causing the workload unbalanced among processes and in turn reducing the overall performance. In contrast, for benchmarks SP-MZ and LU-MZ, the data size of zones are the same, making workload well balanced among processes and thereby with a good performance in parallelism.

VII. Conclusion and Future Work

This paper studies speedup for multi-level parallel computing. Both fixed-sized speedup model and fixed-time speedup model are considered. Moreover, we have derived *E-Amdahl's Law* for the high-level abstract fixed-size speedup, and *E-Gustafson's Law* for the high-level abstract fixed-time speedup for multi-level parallel computing. The experiment results illustrate that *E-Amdahl's Law* is much more accurate in performance estimation than *Amdahl's Law* for multi-level parallelism in NPB benchmarks. Developers can use *E-Amdahl's Law* in performance prediction and performance optimization for multi-level parallelism.

In this paper, our multi-level speedup models is targeted at the homogeneous multi-level parallelism, which requires that all the processing elements of hardware architectures are identical. It is our future work to extend the speedup model to the heterogeneous multi-level parallelism by taking into account the different computing capacities of heterogeneous processing elements. For example, GPGPU has attracted significant amount of research in HPC systems [18], [19], [20]. The multi-GPUs parallelism with the heterogeneous paradigm [21] belongs to a heterogeneous multi-level parallelism. Consider a GPU cluster [17] of computing nodes each equipped with multiple GPUs. User need to consider different computing capacities of CPU cores and GPUs in such heterogeneous processing environments.

VIII. Acknowledgment

This work was supported by the project "User and Domain Driven Data Analytics", granted by A*STAR Thematic Strategic Research Programme.

References

[1] P. Balaji, D. Buntinas, et al, *Fine-Grained Multithreading Support for Hybrid Threaded MPI Programming*, International Journal of High Performance Computing Applications, vol 24, pp. 49-57, 2010.

[2] Top500 supercomputer sites. http://www.top500.org/list/2010/11/, 2010.

[3] R. Rabenseifner, G. Hager, G. Jost, *Hybrid MPI/OpenMP parallel programming on clusters of multi-core SMP nodes*, 17th Euromicro International Conference on Parallel, Distributed and Network-based Processing, pp 427-436, 2009.

[4] G. Jost, H.Q. Jin, D.A. Mey, F.F. Hatay, *Comparing the OpenMP, MPI, and Hybrid Programming Paradigms on an SMP Cluster*, NAS Technical Report NAS-03-019, 2003.

[5] X.H. Sun, L.M. Ni, *Another view on parallel speedup*, Proceedings of Supercomputing '90 (Cat. No.90CH2916-5), p 324-33, 1990.

[6] G. M. Amdahl, *Validity of the Single-Processor Approach to Achieving Large Scale Computing Capabilities*, In AFIPS Conference Proceedings, pp. 483-485, 1967.

[7] J.L. Gustafson, *Reevaluating Amdahl's law*, Communications of the ACM, v 31, n 5, p 532-533, May 1988.

[8] *NVIDIA CUDA C Programming_Guide Version 4.0*, http://developer.download.nvidia.com/compute/cuda/4_0/toolkit/docs/CUDA_C_Programming_Guide.pdf. May, 2011.

[9] X.H. Sun, J.L. Gustafson,*Toward a better parallel performance metric*, Parallel Computing, v 17, n 10-11, pp 1093-109, Dec. 1991.

[10] K.C. Sevcik, *Characterizations of parallelism in applications and their use in scheduling*, International Conference on Measurement and Modeling of Computer Systems, PP. 171-180, May 1989.
[11] X.H. Sun, L.M. Ni, *Scalable problems and memory-bounded speedup*, Journal of Parallel and Distributed Computing, v 19, n 1, p 27-37, Sept. 1993.
[12] L. Wolters, G. Cats, N. Gustafsson, *Data-parallel numerical weather forecasting*, Scientific Programming, v 19, n 3, pp. 159-171, Nov. 1995.
[13] B.M. Chapman, *Parallel Application Development with the Hybrid MPI+OpenMP Programming Model*, Proceedings of the 9th European PVM/MPI Users' Group Meeting on Recent Advances in Parallel Virtual Machine and Message Passing Interface, pp. 13, 2002.
[14] R.F. Van Der Wijngaart, H. Jin, *NAS Parallel Benchmarks, Multi-Zone Versions*, NAS Technical Report NAS-03-010, NASA Ames Research Center, Moffett Field, CA, 2003.
[15] H. Q. Jin, R. F. Wijngaart, *Performance characteristics of the multi-zone NAS parallel benchmarks*, 18th International parallel and distributed processing symposium, pp. 26-30, April, 2004.
[16] S. Sahni, V. Thanvantri, *Performance metrics: keeping the focus on runtime*, IEEE Parallel and Distributed Technology: Systems and Applications, v 4, no. 1, pp. 43-56, 1996.
[17] V.V. Kindratenko, J.J. Enos, et al, *GPU clusters for high-performance computing*, 2009 IEEE International Conference on Cluster Computing and Workshops (CLUSTER), pp. 1-8, Sept. 2009.
[18] W. Fang, B. He, Q. Luo, G. Naga, *Mars: Accelerating MapReduce with Graphics Processors*, Parallel and Distributed Systems, IEEE Transactions on, pp. 608-620, 2011.
[19] He, B., Fang, W., Luo, Q., Govindaraju, N.K., and Wang, T., *Mars: a MapReduce framework on graphics processors*, In Proceedings of PACT. 2008, pp. 260-269, 2008.
[20] Bingsheng He, Ke Yang, Rui Fang et al. *Relational joins on graphics processors*, In Proceedings of the 2008 ACM SIGMOD international conference on Management of data (2008), pp. 511-524.
[21] Y. S. Tan, B.-S. Lee, R. H.Campbell, B. He. *A Map-Reduce Based Framework for Heterogeneous Processing Element Cluster Environments*. CCGrid 2012, pp.1-8.

APPENDIX A
EQUIVALENCE PROOF OF E-AMDAHL'S LAW AND E-GUSTAFSON'S LAW

The following is the proof of the equivalence between Equation (16) and (20) by the induction method with the reverse order.

Proof: **(i). Base Case:** When $i = m$, in terms of Equation (20), the scaled portion of parallelism is

$$f^{'}(m) = \frac{f(m)p(m)}{1 - f(m) + f(m)p(m)}. \tag{22}$$

According to *E-Amdahl's Law* of Equation (16), it holds:

$$\begin{aligned} sp(i = m) &= \frac{1}{1 - f^{'}(m) + \frac{f^{'}(m)}{p(m)}} \\ &= \frac{1}{1 - \frac{f(m)p(m)}{1-f(m)+f(m)p(m)} + \frac{\frac{f(m)p(m)}{1-f(m)+f(m)p(m)}}{p(m)}} \\ &= 1 - f(m) + f(m)p(m). \end{aligned} \tag{23}$$

Note that Equation (23) is identical to Equation (20). Thereby, Equation (16) and (20) are equivalent when $i = m$.

(ii). Induction Step: Suppose it is true that Equation (16) and (20) are equivalent for $i = k, (1 < k \leq m)$. We need to prove the proposition is true when $i = k - 1$:

In terms of Equation (20), the scaled portion of parallelism at $i = k - 1$ level is

$$f^{'}(i = k - 1) = \frac{f(k-1)p(k-1)sp(k)}{1 - f(k-1) + f(k-1)p(k-1)sp(k)}. \tag{24}$$

Based on *E-Amdahl's Law* of Equation (16), there is,

$$\begin{aligned} sp(i = k - 1) &= \frac{1}{1 - f^{'}(k-1) + \frac{f^{'}(k-1)}{p(k-1)sp(k)}} \\ &= \frac{1}{1 - \frac{f(k-1)p(k-1)sp(k)}{1-f(k-1)+f(k-1)p(k-1)sp(k)} + \frac{\frac{f(k-1)p(k-1)sp(k)}{1-f(k-1)+f(k-1)p(k-1)sp(k)}}{p(k-1)sp(k)}} \\ &= 1 - f(k-1) + f(k-1)p(k-1)sp(k). \end{aligned} \tag{25}$$

Note that when $i = k - 1$, Equation (20) is equivalent to Equation (25). Hence, the proposition is true when $i = k - 1$.

(iii). Conclusion Step: according to **(i)** and **(ii)**, it is true that Equation (16) and (20) are equivalent for $1 \leq i \leq m$.

■