Scalable Computing in the Multicore Era

Xian-He Sun, Yong Chen and Surendra Byna

Illinois Institute of Technology, Chicago IL 60616, USA

Abstract. Multicore architecture has become the trend of high performance processors. While it is generally accepted that we have entered the multicore era, concerns exist on scaling multicore processors. Technology is available, but major vendors are hesitant in entering the multicore market with processors that have large number of cores, citing Amdahl's law. This is a very interesting phenomenon, where history seems to repeat itself on the scalability debate of parallel processing occurred 20 years ago. Following the scalable computing concept, especially the fixed-time and memory-bounded speedup metrics, in this study, we argue that the scalability of multicores is not limited by Amdahl's law. We study two speedup models of multicore architecture from the scalable computing point of view. These two models show that multicores have a good scalability and add a new dimension of scalable computing.

1 Introduction

High performance computing has received long missed intensive attention from industry and academia recently. This renewed interest is due to two new developments in computing: cloud computing and multicore processors. Cloud computing employs a cloud of computers, usually a cluster of supercomputers, and Grid technology to provide virtual computers on demand. Multicore processors integrate many cores into one chip to overcome the physical constraints of uniprocessor architecture and deliver high computing power with single chip.

Cloud computing and traditional high-end computing applications demand high performance power; multicore architecture has emerged as an able technology to meet that demand. By scaling up the number of cores, multicore processors provide a new dimension to scale up performance. However, recently we have noticed a very interesting phenomenon. While some small start-up companies are making large-scale multicores [16][10], established companies are reluctant to enter the multicore market with processors that have large number of cores. IBM's Cell processor [6], for instance, has only 8 cores (plus a master core). AMD's mainstream processors, Phenom and Opteron families, have only four cores. While Intel has a road map for multicores, building an 80-core processor in 2011 [15], it is too conservative and slow moving. This slow movement has its theoretical foundations. Based on Amdahl's law, many researchers believe multicore systems are not scalable [1][9]. Amdahl's law states that if a portion of a computer can be improved and another portion of the architecture cannot be improved, then the portion that cannot be improved will quickly dominate

the performance, and further improvement of the improvable portion will have little impact. With the known memory-wall problem [14], many believe that multicore processor with a small number of cores, such as 8, is a good design choice. History seems to repeat itself and reminds the days before Gustafson introduced the scalable computing concept for parallel processing in 1988 [7]. IBM and Cray were making parallel machines with 2 to 8 processors, such as IBM 7030 Stretch Data Processing System and Cray Y-MP before scalable computing was introduced. The introduction of the scalable computing concept changed the view of parallel processing and made major vendors entered the massively parallel processing arena. Today, IBM's Petaflop machine Roadrunner at Los Alamos National Laboratory has 25,200 processors. The Ranger supercomputer at Texas Advanced Computing Center has 15,744 processors. Unfortunately, the scalable computing concept [7][13][5][12] has not been well introduced into the multicore processors design at this time. Studying the scalability of multicore processors is a timely research effort. Recently Hill and Marty have studied the Amdahl's law applicability for multicore design and call for models of multicore performance [8]. In response to their call, we study the scalability of multicore processors and analyze two speedup models following the results in scalable parallel processing in this research. We first revisit the three speedup models of parallel processing, fixed-size (Amdahl's law), fixed-time and memory-bounded speedup; we then extend them to multicore scalability analysis. Detailed case studies are presented. We conclude that multicore architecture is scalable and has the potential to achieve linear speedup in scalable computing, where problem size is increased with the number of cores.

2 Speedup Models of Parallel Processing

In this section, we review the three classic speedup models, Amdahl's law [1], Gustafson's law [7], and Sun and Ni's law [13], of parallel processing in brief.

2.1 Amdahl's Law

The original idea presented by Amdahl [1] is a general observation about the performance improvement limit of any enhancement, and was later summarized as the well-known Amdahl's law. When we apply Amdahl's law to parallel processing, we have the speedup metric as:

$$S_{Amdahl} = \frac{Performance_{new}}{Performance_{old}} = \frac{SequentialExecutionTime}{ParallelExecutionTime} = \frac{T_s}{T_p} \quad (1)$$

Suppose α is the fraction of the code that is sequential, which cannot be parallelized, and p is the number of processors. Assuming that all overheads are ignored, we have: $T_p = \alpha T_s + (1-\alpha)T_s/p$. Therefore,

$$S_{Amdahl} = \frac{T_s}{T_p} = \frac{T_s}{\alpha T_s + (1-\alpha)T_s/p} = \frac{1}{\alpha + (1-\alpha)/p} \quad (2)$$

This formula is called Amdahl's law for parallel processing. When p, the number of processors, increases to infinity, the speedup becomes $\lim_{p\to\infty} S_{Amdahl} = \lim_{p\to\infty} \frac{1}{\alpha+(1-\alpha)/p} = 1/\alpha$. This equation shows that the speedup is limited by the sequential fraction, a nature of the problem under study, even when the number of processors is scaled to infinity. Amdahl's law advocates that a large-scale parallel processing is less interesting because the speedup has an upper bound of $1/\alpha$.

2.2 Gustafson's Law

A tacit assumption in Amdahl's law is that the problem size, or the workload, is fixed. The speedup emphasizes time reduction of a given problem. Amdahl's law is thus also called *fixed-size speedup* model. In 1988, Gustafson introduced *fixed-time speedup* model [7] to motivate large-scale parallel processing. The fixed-time speedup model suggests powerful machines can be designed for solving large problems and the problem size should be scaled to match the increased computing capability in a parallel processing system. Thus, the fixed-time speedup is defined as:

$$S_{Gustafson} = \frac{SequentialTimeofSolvingScaledWorkload}{ParallelTimeofSolvingScaledWorkload} \tag{3}$$

Suppose the original workload and the scaled workload finished in the same amount of time are W and W', respectively. We have $W' = \alpha W + (1-\alpha)pW$. Therefore,

$$S_{Gustafson} = \frac{SequentialTimeofSolvingW'}{SequentialTimeofSolvingW} = \frac{W'}{W} = \alpha + (1-\alpha)p \tag{4}$$

This equation is known as Gustafson's law. It states that the fixed-time speedup is a linear function of p if the workload is scaled up to maintain a fixed execution time. Gustafson's law suggests that it is beneficial to build a large-scale parallel system as the speedup can grow linearly with the system size.

2.3 Sun and Ni's Law

Many parallel applications cannot scale up to meet the time bound constraint due to some physical constraint. In practice, the physical constraint is often the memory limitation. In distributed-memory machines, the number of processors and memory are increased in pair. Out-of-core computing will reduce the performance significantly and is largely prohibited. With this in mind, Sun and Ni proposed *memory-bounded speedup* model [13]. Let W^* be the scaled workload under memory space constraint. The memory-bounded speedup is defined as:

$$S_{SunNi} = \frac{SequentialTimeofSolvingScaledWorkload, W^*}{ParallelTimeofSolvingScaledWorkload, W^*} \tag{5}$$

Assume that the parallel portion of the workload can be scaled up $G(p)$ times. That is, the scaled workload is $W^* = \alpha W + (1-\alpha)G(p)W$. The factor $G(p)$ reflects the increase in the workload as the memory capacity increases p times. Therefore,

$$S_{SunNi} = \frac{\alpha W + (1-\alpha)G(p)W}{\alpha W + (1-\alpha)G(p)W/p} = \frac{\alpha + (1-\alpha)G(p)}{\alpha + (1-\alpha)G(p)/p} \tag{6}$$

Sun and Ni's law is a generalization of Amdahl's law and Gustafson's law, where Amdahl's law is a special case with $G(p) = 1$, and Gustafson's law is a special case with $G(p) = p$. In general, the computational workload increases faster than the memory requirement, thus $G(p) > p$ and the memory-bounded speedup model gives a higher speedup than the fixed-size and fixed-time speedup.

3 Multicore Architecture Assumptions and Definitions

We follow the models of parallel processing to study the scalability of multicore architectures. We take data access as the bottleneck that we cannot improve, and study the scalability of multicores in terms of cores in a processor.

3.1 Multicore Architecture

To simplify the discussion, this study assumes symmetric multicore processor architectures. We assume that the multicore processor under study has n cores, and each core has a dedicated primary cache, L1 cache, and all cores share remaining levels of the memory hierarchy. This assumption matches with most of existing multicore/manycore processors that are either commercially available or in production. For simplicity, we also assume that there is no context switch while running a parallel application on a multicore processor.

3.2 Definitions

We introduce the following definitions in order to describe the scalability analysis model of a multicore architecture.

Definition 1. *The work (or workload, or problem size) is defined as the number of instructions that are to be executed.*

Let I denote the number of instructions, or the work. The work is composed of computation instructions and data access instructions. Let I_p denote the number of computation instructions, and I_c denote the number of data communication instructions. Therefore, $I = I_p + I_c$.

Definition 2. *The execution time is defined as the number of CPU cycles spent for executing the instructions, either for computation or for data access.*

Let T denote the execution time. The execution time is composed of the computation time spent on processing units and the communication to wait the data to be ready. Let T_p denote the computation time, and T_c denote the data access time. Therefore,

$$T = T_p + T_c \tag{7}$$

Note that, in the context of parallel processing, this formula is under two assumptions: the load is balanced and every processing unit performs computation and communication at the same time. Following Amdahl's law of parallel processing, we assume that computing is perfectly parallelized and the load is balanced. Since we have assumed that the studied multicore processor is a symmetric architecture, the computation instructions are issued with a same speed on each core. Equation 7 stands under our study. If we assume the execution speed of each core is τ instructions per second, we have $T_p = I_p/\tau$. We can further analyze the data access time T_c as well in a typical multicore architecture. A major fraction of data access time is spent on the data transfer between the lowest-level cache memory, L2 cache in this study, and the main memory. The data access time can thus be roughly modeled as:

$$T_c = (I_c \times (1 - H_{L1}) \times (1 - H_{L2}) \times [F \times L_{word} + (1 - F) \times L_{cache}])/B \tag{8}$$

where H_{L1} is the L1 cache hit ratio, H_{L2} is the L2 cache hit ratio, L_{word} and L_{cache} are word size and L2 cache line size respectively, F is a locality factor to represent the spatial locality characteristic of the work, ranging from 0 in the case of totally random accesses to 1 in the case of totally sequential accesses, and B is the memory bandwidth. When the cache hit ratio, locality factor, word size and cache line size are fixed for a study, we rewrite T_c as: (c_1 is a constant)

$$T_c = c_1 I_c \tag{9}$$

In computer architecture, speedup is a measure of improvement. The following definition explains the speedup concept we employs in this study.

Definition 3. *The speedup, in the context of computer architecture, is defined as the ratio of the execution time in the original architecture and the execution time in the enhanced architecture.*

4 Scalability of Multicore Architecture

Now, we are ready to extend the three speedup models of parallel processing into multicore architectures and present theoretical analysis.

4.1 Amdahl's Law on Multicore Architecture

Amdahl's law [1] is a basic law of architecture design. It shows that the inherited limitation of any architecture improvement is determined by some performance

factor that cannot be improved with architecture improvement. For parallel computing, this factor is the portion of the application which must be solved sequentially. In the context of multicore architecture, the data access delay is such a limiting factor that cannot be enhanced no matter how many cores are utilized in computation. With a similar assumption used in the Amdahl's law in parallel processing, we assume that computing can be perfectly parallelized and data access is independent of problem size and the number of cores. Then with an n-core processor architecture, the new execution time is: $T' = T'_p + T'_c = T_p/n + T_c$. Thus, the Amdahl's speedup is:

$$S_{FS} = \frac{T}{T'} = \frac{T_p + T_c}{T_p/n + T_c} \tag{10}$$

Amdahl's law states that the performance gain of a multicore architecture is quickly limited by the data access delay, with an upper bound of $(T_p+T_c)/T_c$. The gap between data access and computing speed has been growing larger and larger during the last three decades and is known as the *memory-wall problem* [14]. By Amdahl's law, the multicore architecture is not scalable. It is a pessimistic view, and has accepted by many in both academia and industry [8][9].

As in parallel processing, Amdahl's law of multicore architecture is under a tacit assumption: the application workload is fixed. Multicore is also a way of parallel processing. The scalable computing concept of parallel processing should be extended to multicore design. Considering data access as the non-improvable performance factor, we study two scaled speedup models in the following section.

4.2 Multicore Architecture for Scalable Computing

Allowing problem size increases with computing power, we now study the scalability of multicore architectures from scalable computing point of view. We first present a *fixed-time* scaled speedup model.

Definition 4. *The fixed-time speedup of a multicore architecture machine is defined as the ratio of execution time of solving the scaled workload on a single core to execution time of solving the scaled workload on multiple cores, where the scaled workload is the amount of work that is finished in the enhanced mode within the same amount of time as in the original mode.*

Following a similar assumption of Gustafson's law of parallel processing, we assume that the communication work/cost is fixed. The assumption is valid since the goal of this study is to show the potential scalability of multicore architectures. According to the definition and the assumption, the scaled workload is $nT_p + T_c$. Therefore, the fixed-time speedup of the multicore architecture from the scaled computing viewpoint is:

$$\begin{aligned} S_{FT} &= \frac{Time of Solving Scaled Workload in Original Mode}{Time of Solving Scaled Workload in Enhanced Mode} \\ &= \frac{Time of Solving Scaled Workload in Original Mode}{Time of Solving Original Workload in Original Mode} = \frac{nT_p+T_c}{T_p+T_c} \end{aligned} \tag{11}$$

This result reveals that, from the scalable computing viewpoint, the multicore architecture is linearly scalable and suitable for large-scale manufacturing as long as the data communication time is fixed. When the number of cores, n, goes to infinity, the speedup can grow linearly with n. This finding confirms that multicore architecture with large number of cores is meaningful and has real scalability potential. We do not need to reduce the data access delay, but the assumption here is the data access delay is fixed and does not increase with the number of cores and the problem size. While formula (11) shows the potential of large-scale multicores, data access remains as a technical hurdle that needs to be overcome.

The fixed-time scaled speedup model exposes an optimistic view of the scalability of multicore architectures. Nevertheless, the problem size scaling up is often not constrained by the execution time, but the memory capacity. Please notice that the term memory here is a general term. It is not only for main memory, but for any layer in a memory hierarchy. When the problem size increases and the amount of data is larger than the storage of a layer in the memory hierarchy (for instance, accessing data via virtual memory), data access delay will have a significant increase and performance will have a significant drop. We consider such a performance drop is intolerable. In other words, data access bandwidth and data access latency prohibit accessing the next layer of the memory hierarchy and the problem size increment is limited by the memory available in the current memory hierarchy under consideration. In general, memory capacity is increased with the computing power. In distributed parallel processing, processor and its local memory are paired together. In multicore architecture, core and L1 cache are paired together (although we could scale up L2 cache along with the increase of the number of cores). The constraint of scalable computing is often to be the memory availability. Thus, following the speedup models of parallel processing, we study a *memory-bounded speedup* model for multicore architecture. Similar to the fixed-time speedup model, the memory-bounded speedup model also provides a scaled computing view of the multicore architecture. The difference is that the execution time is the limiting factor in the fixed-time speedup, while the data access capability is the limiting factor in the memory-bounded speedup. In addition to show the potential of large-scale multicore systems, the memory-bounded model also reveals the tradeoff of computing speed and memory capacity in multicore design.

Definition 5. *The memory-bounded speedup of a multicore architecture is defined as the ratio of execution time of solving the scaled workload on a single core to execution time of solving the scaled workload on multiple cores, where the scaled workload is the amount of work that is finished in the enhanced mode with a constraint on the data access latency.*

Suppose the work is scaled with a function $g(n)$ under memory bound constraint. According to the memory-bounded speedup model, the amount of work that can be scaled exactly with the data access constraint is dependent on the available memory and the memory requirement of the application under study.

For instance, a matrix multiplication application has $3N^2$ memory requirement and $2N^3$ computing power requirement, where N is the matrix rank, assuming two source matrices are square $N \times N$ matrices. Therefore, the computing requirement in terms of the memory requirement, function $g(n)$, is about $0.38n^{3/2}$. Hence, the scaled workload under memory bound constraint is $g(n)T_p + T_c$. The memory-bounded speedup is

$$S_{MB} = \frac{TimeofSolvingScaledWorkloadinOriginalMode}{TimeofSolvingScaledWorkloadinEnhancedMode} = \frac{g(n)T_p + T_c}{g(n)T_p/n + T_c} \tag{12}$$

When $g(n) = 1$, we have $S_{MB} = \frac{g(n)T_p+T_c}{g(n)T_p/n+T_c} = \frac{T_p+T_c}{T_p/n+T_c}$. When $g(n) = n$, we have $S_{MB} = \frac{g(n)T_p+T_c}{g(n)T_p/n+T_c} = \frac{nT_p+T_c}{T_p+T_c}$. This shows that the fixed-size speedup and the fixed-time scaled speedup are special cases of the memory-bounded scaled speedup when the scaling function under the data access constraint is 1 and n respectively, similar as the property of the memory-bounded speedup model of parallel processing. In general, $g(n) > n$, such as $g(n) = 0.38n^{3/2}$ (for sufficiently large n). Thus, $S_{MB} > S_{FT}$.

As shown above, the theoretical analysis of the fixed-time and memory-bounded scaled speedup of multicore architecture reveals that multicore architectures can scale up well. The memory-bounded scaled speedup with data access as the constraint can usually achieve an even better performance than the fixed-time model does. It shows the memory constraint on performance and indicates the tradeoff between memory capacity and computing power of cores in a multicore design. Memory-bounded speedup has its role in scalable multicore/manycore processor design.

5 Case Studies

In this section, we analyze the three speedup models of the multicore architecture with different case studies, and compare the results in detail.

5.1 Amdahl's Speedup of Multicore Architecture

According to the definition and assumptions, we can further analyze Amdahl's speedup and examine its property with parameters of real practice. Amdahl's speedup, or the fixed-size speedup, is:

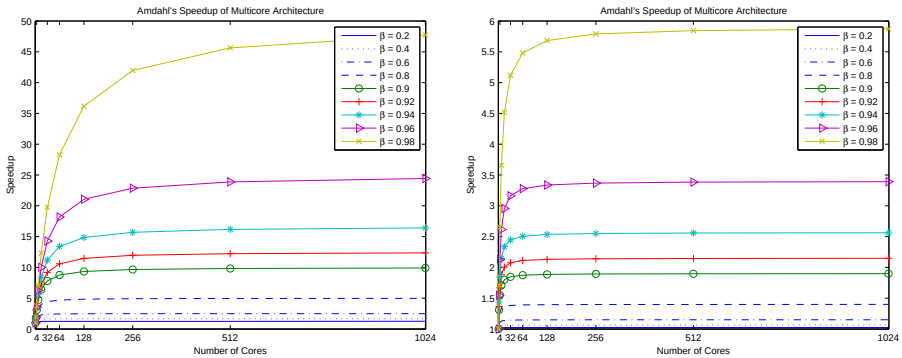
$$S_{FS} = \frac{T_p + T_c}{T_p/n + T_c} = \frac{I_p/\tau + c_1 I_c}{I_p/n\tau + c_1 I_c} \tag{13}$$

As we discussed previously, c_1 and τ are constants in a study. The value of c_1 has a magnitude of 10^{-8} to 10^{-9} according to formula 8 and 9, and the practical value of L1 and L2 cache hit ratio, word size, cache line size and memory bandwidth. The value of τ has a magnitude of 10^9 according to the contemporary processor core's clock frequency and instruction issue bandwidth. Therefore, the

product value of $c_1\tau$ is around 1 or 10 in practice. We analyze the Amdahl's speedup value under these two different scenarios.

Fig. 1a shows the Amdahl's speedup under different ratios of computation and data access instructions, where $c_1\tau = 1$. We define the ratio of the computation instructions and the overall instructions as β, that is $\beta = I_p/(I_p + I_c)$. For example, the case with $\beta = 0.2$ represents the computation instruction is 20% of the total instructions (or the application workload). We compute Amdahl's speedup following formula 13 and plot the results in Fig. 1a. The horizontal axis represents the number of cores, scaled from 1 to 1024. The vertical axis represents the speedup value. As we can see clearly from this figure, the Amdahl's speedup illustrates a very limited scalability of the multicore architecture. For instance, when $\beta = 0.8$, the achieved speedup with 32 and 64 cores is about 4.44 and 4.71 respectively. The speedup has quickly reached its limit, around 5, when the number of cores is scaled to 128. If β is increased to 0.9, the speedup has a similar pattern to the previous case, except the speedup limit is around 10 and is reached when the number of cores is scaled to 256. Even if the ratio of computation instructions is increased to 98%, the speedup is improved slightly, but still quickly reaches the limit with 512 cores and the maximum speedup is only around 50.

Fig. 1b shows Amdahl's speedup under similar scenarios as the previous case study, however with $c_1\tau = 10$. This figure illustrates that the speedup values have a similar pattern as the previous case study results, but the speedups are even worse. For instance, the speedup starts to reach its limit with 128 cores even $\beta = 0.98$, and the maximum speedup is merely about 5.9.



(a) $c_1\tau = 1$ (b) $c_1\tau = 10$

Fig. 1: Amdahl's Speedup of Multicore Architecture

Both of these case studies demonstrate that Amdahl's speedup presents a pessimistic view of the multicore architecture scalability. However, as we argue in this study, the real applications tend to utilize the enhanced multicore architecture computation capability to scale the problem size to achieve a much better and more accurate result. Amdahl's speedup, a fixed-size speedup in es-

sential, is not suitable to analyze the scalability of multicore architecture in real practice. The suggested scaled fixed-time and memory-bounded speedup model present a more realistic view of the multicore architecture scalability. The following subsections present the case study results of these two models.

5.2 Fixed-time Scaled Speedup Model of Multicore Architecture

According to the assumption and the fixed time constraint, we have

$$T'_c = T_c; T'_p = \frac{I'_p}{n\tau} = T_p = \frac{I_p}{\tau} \tag{14}$$

Hence, the scaled workload, I'_p, that is finished in the enhanced mode is: $I'_p = nI_p$. Therefore, the fixed-time speedup of the multicore architecture can also be expressed as:

$$S_{FT} = \frac{I'_p + I_c}{I_p + I_c} = \frac{nI_p + I_c}{I_p + I_c} = 1 + (n-1)\beta \tag{15}$$

Fig. 2 demonstrates the scalability of multicore architecture with the fixed-time speedup model. We compute the speedup following formula 15 under different scenarios where β ranges from 0.2 to 0.98, and plot the results in Fig. 2. The fixed-time speedup model provides a more practical characterization of the capability of a multicore architecture. As shown in Fig. 2, this speedup model presents a much more optimistic view of the multicore architecture. For instance, when $\beta = 0.2$, even though the speedup improves slowly, it still achieves a value of around 205 with 1024 cores. When the ratio of computation instructions increases, the speedup improves gradually. When β reaches 0.8 and 0.9, the speedup achieved is around 819 and 922, respectively, with 1024 cores. When β further increases, the speedup increment is slowed down and the speedup with 1024 cores is around 1003 when $\beta = 0.98$.

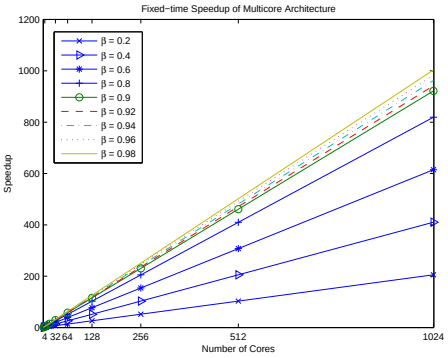
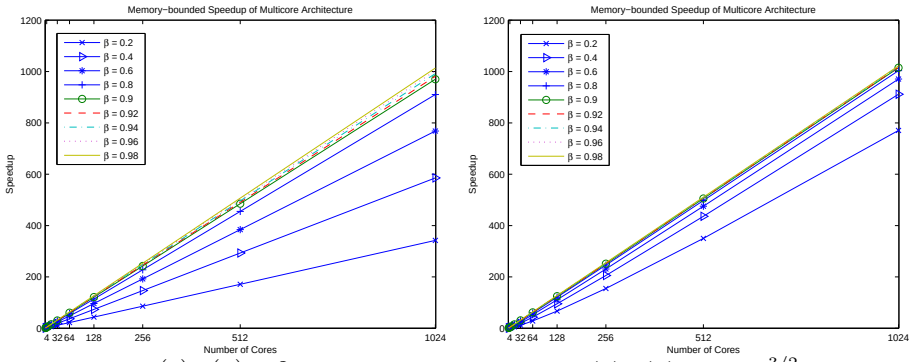


Fig. 2: Fixed-time Scaled Speedup of Multicore Architecture

5.3 Memory-bounded Speedup Model of Multicore Architecture

Fig. 3a and 3b demonstrate the speedups with the memory-bounded scaled speedup model for multicore architectures. These two figures report that the speedup value with the scaling function $g(n)$ as $2n$ and $0.38n^{3/2}$, respectively.

Similar to the fixed-time model, the memory-bounded speedup model also reveals that the multicore architecture can scale well as long as the workload size of the application can be allowed to grow with the number of cores to achieve more accurate and better results. In addition, the results of the memory-bounded speedup model show that an even better performance can be achieved when data access constraint is considered to scale workload instead of the execution time constraint. As shown in Fig. 3a and 3b, the scalability of multicore architecture can increase steadily compared with the fixed-time model. The memory-bounded speedup model can reflect the reality well and exhibit a promising view of the large-scale multicore architecture. It suggests that multicore architectures can be scaled up well.



(a) $g(n) = 2n$ (b) $g(n) = 0.38n^{3/2}$

Fig. 3: Memory-bounded Scaled Speedup of Multicore Architecture

6 Conclusion

We present two scaled speedup models, fixed-time and memory-bounded model, from a scaled computing view to characterize the scalability of multicore architectures in this study. Both theoretical and case study results illustrate that the multicore architecture is scalable with the number of cores.

In this study, we do not assume data access delay being changed with the increasing of computing power, but assume it is fixed. While our study shows large scalable multicore systems have a real potential, data access remains as a critical issue that needs to be addressed to achieve a desired scalability of the multicore architecture. Many approaches are proposed in recent years to address the memory-wall problem, including the data prefetching approach proposed by the authors [2]. Ours Data Access History Cache (DAHC) based data prefetching [3] provides a practical hardware solution at processor-memory level. In the meantime, the data server based push prefetching [11][4] provides a software solution at memory-disk level. A hybrid adaptive data prefetching mechanism

reduces the latency via two-stage, processor-memory and memory-disk stage, and combines both merits of prediction and pre-execution approaches [11][3][4].

The scalability of multicore architectures is an area that is largely unexplored. The scaled computing view of the multicore architecture presented in this paper provides an insight to the design and evaluation of a large-scale multicore architecture. We expect this study will inspire further discussions in this direction. We also hope that our study can shatter the pessimistic view of the limited scalability of multicore architectures in the industry and academia, and to stimulate a breakthrough in designing large-scale multicore processors.

7 Acknowledgements

Professor Guo-Liang Chen introduced scalability study in his textbook *"Parallel Computing: Architecture, Algorithm and Programming"* about ten years ago, probably the first one introducing the concept in a Chinese textbook. We visit the scalability issues again under multicore processors in this study in honor of his contribution to scalability and parallel processing research in China.

References

1. Amdahl, G. M.: Validity of the Single-Processor Approach to Achieving Large Scale Computing Capabilities. In AFIPS Conference Proceedings. (1967)
2. Byna, S., Chen, Y., Sun, X.-H.: A Taxonomy of Data Prefetching Mechanisms. In Proc. of Intl. Symp. on Parallel Architectures, Algorithms, and Networks. (2008)
3. Chen, Y., Byna, S., Sun, X.-H.: Data Access History Cache and Associated Data Prefetching Mechanisms. In Proc. of ACM/IEEE Supercomputing 2007. (2007)
4. Chen, Y., Byna, S., Sun, X.-H., Thakur, R., Gropp, W.: Exploring Parallel I/O Concurrency with Speculative Prefetching. In the 37th International Conference on Parallel Processing. (2008)
5. Chen, Y., Sun, X.-H., Wu, M.: Algorithm-System Scalability of Heterogeneous Computing. Journal of Parallel and Distributed Computing. (To appear).
6. Gschwind, M.: Chip Multiprocessing and the Cell Broadband Engine. Computing Frontiers. (2006)
7. Gustafson, J. L.: Reevaluating Amdahl's Law. Communications of the ACM. (1988)
8. Hill, M., et. al.: Amdahl's Law in the Multicore Era. IEEE Computer. (2008)
9. Madden, P.H.: Forty Years of Amdahl's Law or The Sky Is Falling. http://vlsicad.cs.binghamton.edu/Sky.pdf
10. Makino, J., Hiraki, K., Inaba, M.: GRAPE-DR: 2-Pflops Massively-Parallel Computer with 512-Core. In Proc. of ACM/IEEE Supercomputing 2007. (2007)
11. Sun, X.-H., Byna, S., Chen, Y.: Server-based Data Push Architecture for Multiprocessor Environments. Journal of Computer Science and Technology. (2007)
12. Sun, X.-H., Chen Y., Wu, M.: Scalability of Heterogeneous Computing. In Proceedings of 34th International Conference on Parallel Processing. (2005)
13. Sun, X.-H., Ni, L.: Scalable Problems and Memory-Bounded Speedup. Journal of Parallel and Distributed Computing. (1993)
14. Wulf, W.A., McKee, S.A.: Hitting the Memory Wall: Implications of the Obvious. Computer Architecture News. 23(1):20-24. (1995)
15. Intel. http://news.cnet.com/2100-1006_3-6119618.html
16. Kilocore. http://www.rapportincorporated.com/kilocore/kilocore_overview.html