An Insight of Speedup

속도향상에 대한 고찰

Ando Ki (기안도)   Computer System Department (하드웨어구조연구팀 선임연구원)

Speedup is often used to show scalability, but its classical definition fails to explain some real measurements such as superlinear speedup. This leads to scaled speedup which scales other system parameters as number of processors changes. In this paper, scaled speedup and architectural speedup are introduced and superlinear speedup is explained with its cause.

I. Introduction

Scalability is most basic one to evaluate multiprocessors and a measure of ability to achieve performance proportional to the number of processors. Unfortunately, it is difficult to measure the scalability directly, so that instead some other metrics are used to evaluate the scalability of multiprocessors such as *speedup*. Because speedup can be easily determined by measuring execution time of programs, it is widely accepted. However, the execution time of a certain program on a system may vary because of the program's characteristics and system environment including everything which affects the performance. Thus, a definition of speedup called architectural speedup is introduced which can separate architectural issues from measurement.

Theoretical or analytical models usually use asymptotic analysis which assumes large number of processors, large size of problem, and idealized machine models. This implies that asymptotic analysis is likely to ignore lower-order terms that can be significant for a given problem size and processor number. Furthermore, idealized machine model can be very different from the physical machine in which one is interested. As a result, speedup analysis for real machine normally uses real measurement.

In this paper, speedup is discussed and classified. Then, architectural speedup is introduced after the scaled speedup is explained. In addition, linear and superlinear speedup is discussed.

II. Speedup

The most frequently used performance metric of parallel processing on multiprocessors is *speedup* which is given by

$$S(n) = \frac{T_1}{T_n} \quad (1)$$

where n is the number of processors, T_1 is sequential execution time, and T_n is parallel execution time. Clearly, the larger the speedup, the better the parallel algorithm or architecture. Sequential execution can be based on the best serial algorithm or the single processor execution of the parallel algorithm. When the sequential execution time is chosen as the single processor execution time of the parallel algorithm, speedup is referred as *relative speedup*. On the other hand, speedup is called *absolute speedup* when the best sequential algorithm is used.

The absolute speedup can be used to evaluate parallel algorithms. The relative speedup is used to compare the algorithm itself with a different number of processors and it gives information on the variations and degradations of parallelism. In this work, relative speedup is discussed because we are not interested in comparing sequential and parallel algorithms but rather the effect of architecture on parallel algorithms.

Relative speedup can be divided into *fixed-size speedup* and *scaled speedup*. The most commonly used speedup metric is fixed-size speedup in which the problem size remains constant and is independent of the number of processors. In this context, the limit of parallel execution given by *Amdahl's law* [1] implies that the serial fraction, α (actually, it is the ratio of the serial section to parallel section), limits the parallel algorithm's speedup to an asymptotic speedup of $1/\alpha$. If an execution of a program can be divided into s and p, where s is the amount of time spent on serial part of the program and p is the amount of time spent on parallel part of the program equation (1) can be the following

$$S(n) = \frac{s+p}{s+\frac{p}{n}} \quad (2)$$

In an ideal case, p/n part of equation (2) would diminish as the number of processor, n, increases. Therefore, equation (2) can be expressed as follows.

$$\lim_{n\to\infty} S(n) = 1 + \frac{p}{s} \quad (3)$$

This equation implies that the ratio of the parallel section to the serial section determines the maximum speedup.

Equation (1) can be expressed using α $(= \frac{s}{p})$ as shown below

$$S(n) = \frac{T_1}{\alpha T_1 + \frac{1-\alpha}{n} T_1} = \frac{1}{\alpha + \frac{1-\alpha}{n}} \quad (4)$$

which will result in the following as n increases.

$$\lim_{n\to\infty} S(n) = \frac{1}{\alpha} \quad (5)$$

Above equation clearly states Amdahl's law that the improvement in performance of a parallel execution over a corresponding sequential execution is limited by the fraction of the algorithm that cannot be parallelized.

III. Scaled speedup

Amdahl's law describes by how much execution time can be reduced with parallel processing while the problem size remains constant. However, in practice, as the number of processors increases, the amount of work to be performed increases in order to obtain a more accurate or better result, such as merely specifying a finer mesh or higher resolution for the solution of some physical problem. It is a fact that, in most engineering and scientific computing problems, the serial fraction depends on the problem size. This means that the serial fraction α depends on the problem size x and $\alpha(x)$ would diminish as the size of problem increases.

$$\lim_{n,x\to\infty} S(n,x) = n \tag{6}$$

This implies that nearly linear speedup can be achieved when the effective algorithm with sufficiently large problems is considered.

The above concept lead to scaled speedup [3, 4]. Gustafson examined how the problem size can be scaled up while the execution time is fixed. It is *fixed-time speedup* which scales the problem size to meet the fixed execution time. Furthermore, Sun and others [6, 7] suggested a *memory-bounded speedup* which scales the problem size based on the available memory. Memory-bounded speedup is based on the concept that the size of scalable problem is often determined by the memory available. The shortage of memory is paid for in problem solution time due to the input-output activities or message-passing delays. In the case of large problem size, the speedup is limited more by the memory size than by the number of processors.

One is sometimes interested in the effects of architectural issues on parallel processing rather than the algorithm itself. Thus one wants to ignore relationship between the serial work and parallel work in a given algorithm as Amdahl's law implies.

Let us consider parallel algorithm consists of serial parts $S_0,\ S_1,\ \cdots\ S_a$ and parallel parts $P_0,\ P_1,\ \cdots\ P_b$. Practically speaking, the parallel works can be defined between thread creation and join points. Let T_{S_i} be the amount of time spent on serial work S_i and T_{P_j} be the amount of time spent on parallel work P_j. Let $T_p(n)$ be the parallel execution time using n processors. Let *architectural speedup* $AS(n)$ be defined as the ratio of the single processor execution time, $T_P(1)$, of the parallel works, to n processors execution time, $T_P(n)$, of the parallel works, so that the $AS(n)$ is

$$AS(n) = \frac{T_p(1)}{T_p(n)} \tag{7}$$

The strength of this definition is that it uses execution time of the parallelized part and thus incorporates any communication or synchronization overhead but excludes the perturbation of the serial part. The architectural speedup can give information on the variations and degradations of architectural issues. To make it simpler, it is supposed that all benchmark programs in this work are clearly coded in sequential and parallel parts.

IV. Linear and superlinear speedup

Speedup is said to be *linear*, if an n-processor yields a speedup of n. Linear speedup is not achievable, in general, because of various overheads associated with parallel computation, such as contention for shared resources, and the time required to communicate between processors and between processes or threads [2].

Under the particular situation, we can observe *superlinear speedup* which means speedup better than linear. There are some possible causes of superlinear speedup discussed in [5, 8]: cache size increased in parallel processing, overhead reduced in parallel processing, latency hidden in parallel processing, randomized algorithms, mathematical inefficiency of the serial algorithm, and higher memory access latency in the sequential processing.

Let assume a multiprocessor in which each processor has its private cache. Let suppose that the data size of a given problem is bigger than a single cache size but smaller than the size of two or more cache sizes. When a single processor executes this problem, there will be cache misses and it can take longer time than expected. However, more than one processor will execute this problem with less cache misses. Consequently, the ratio single processor's execution time over multiple processors' will be better than linear. Therefore, the problem size and memory architecture have strong relationship when we consider the fixed-size speedup.

V. Summary

Speedup is widely accepted as a performance metric since it can clearly show superiority between computers. Measuring or calculating speedup is not easy task since there are several variations although the concept of speedup is simple.

An insight, called Amdahl's law, showing achievable speedup using parallel computer was supported by a definition of speedup. However the fact that measurements from real computers did not always follow the classical definition and showed unexpected results, required correction of the definition. As the result, several new definitions of speedup were introduced and these explained unexpected results of real measurements.

In here, scaled speedup is explained in which other system parameters such as problem size or memory size are scaled as number of processors changes. In addition, architectural speedup is introduced, which can exclude effect of serial part of execution.

References

[1] G.M. Amdahl, "Validity of the Single-Processor Approach to Achieving Large Scale Computing Capabilities," In *Proceedings of the AFIPS Conference*, Vol. 30, 1967, pp. 483–485.

[2] Derek L. Eager, John Zahorjan and Edward D. Lazowska, "Speedup Versus Efficiency in Parallel Systems," *IEEE Transactions on Computers*, Vol. 3, 1989, pp. 408–423.

[3] John L. Gustafson, "Reevaluating Amdahl's Law," *Communications of the ACM*, Vol. 31, No. 5, 1988, pp. 532–533.

[4] John L. Gustafson, Gray R. Montry, and Robert E. Benner, "Development of Parallel Methods for a 1024-Processor Hypercube," *SIAM Journal of Scit. Stat. Comput.*, Vol. 9, No. 4, 1988, pp. 609–638.

[5] David P. Helmbold and Charles E. McDowell, "Modeling Speedup(n) Greater Than N," In *Proceedings of the International Conference on Parallel Processing*, 1989, pp. III–219–225.

[6] Xian-He Sun and John L. Gustafson, "Toward a Better Parallel Performance Metric," *Parallel Computing*, Vol. 17, 1991, pp. 1093–1109.

[7] Xian-He Sun and Lionel M. Ni, "Scalable Problems and Memory-Bounded Speedup," *Journal of Parallel and Distributed Computing*, Vol. 19, 1993, pp. 27–37.

[8] Xian-He Sun and Jianping Zhu, "Shared Virtual Memory and Generalized Speedup," In *Proceedings of the 8th International Parallel Processing Symposium*, 1994, pp. 637–643.