Formal Metrics for Large-Scale Parallel Performance

Kenneth Moreland(✉) and Ron Oldfield

Sandia National Laboratories, Albuquerque, NM 87185, USA
{kmorel,raoldfi}@sandia.gov

Abstract. Performance measurement of parallel algorithms is well studied and well understood. However, a flaw in traditional performance metrics is that they rely on comparisons to serial performance with the same input. This comparison is convenient for theoretical complexity analysis but impossible to perform in large-scale empirical studies with data sizes far too large to run on a single serial computer. Consequently, scaling studies currently rely on ad hoc methods that, although effective, have no grounded mathematical models. In this position paper we advocate using a rate-based model that has a concrete meaning relative to speedup and efficiency and that can be used to unify strong and weak scaling studies.

1 Introduction

Empirical scaling studies are an important component in the analysis of parallel algorithms and systems. A scaling study tests the performance of a parallel algorithm using different numbers of processing elements and usually over different amounts of data. A good scaling study shows how effectively additional processing elements are used by the algorithm and through trends provides evidence of behavior at future larger scales.

The practice of measuring scaling relies on the well studied models for the theory of parallel performance. However, current theoretic models rely on comparisons with algorithm behavior on a single, serial processing element. Empirically measuring serial behavior for sufficiently large parallel problems is impractical.

1.1 Performance Analysis Theory

The following is a brief overview of parallel algorithm performance.

The *speedup* of a parallel algorithm is defined as

$$S(n,p) = \frac{T^*(n)}{T(n,p)} \tag{1}$$

where $T(n,p)$ is the time it takes to run the parallel algorithm on p processing elements with an input of size n, and $T^*(n)$ the time for the best serial algorithm on the same input. The best possible serial algorithm may be different than the parallel algorithm although using the same algorithm is also common practice.

J.M. Kunkel and T. Ludwig (Eds.): ISC High Performance 2015, LNCS 9137, pp. 488–496, 2015.
DOI: 10.1007/978-3-319-20119-1_34

In theory the best possible speedup achievable is $S(n,p) = p$ [5] (although superlinear measurements can occur in practice [8]). Thus, we measure the *efficiency* as the ratio of the observed speedup to the ideal speedup.

$$E(n,p) = \frac{S(n,p)}{p} = \frac{T^*(n)}{p\,T(n,p)} \tag{2}$$

Amdahl [1] famously observes the limits of scaling any parallel algorithm based on the fraction f of inherently serial computation that exists in any algorithm. The equation derived from this observation is known as *Amdahl's law*.

$$S(n,p) \leq \frac{1}{f + (1-f)/p} \tag{3}$$

Gustafson [7] observes that the serial fraction tends to go down for larger data sizes in parallel algorithms, which justifies the use of parallel computing for large problems. The *Gustafson-Barsis law* reformulates speedup in terms of the parallel execution rather than the serial execution.

$$S(n,p) \leq p + (1-p) \cdot s(n,p) \tag{4}$$

where $s(n,p)$ is the fraction of time in the parallel execution performing sequential operations. This law shows that speedup can be increased indefinitely as long as the serial fraction drops commensurately with the processing element increase, which can often be done by increasing the problem size. Grama et al. [6] introduce an *isoefficiency* relation that determines how much a problem needs to grow to maintain a desired level of efficiency. Given a desired efficiency E_d, the following inequality must hold.

$$T(n,1) \geq \frac{E_d}{1-E_d} T_o(n,p) \tag{5}$$

where $T_o(n,p)$ is the total overhead (redundant, idle, and extra computation plus communication) for running the algorithm on p processing elements for data of size n.

Performance analysis theory is reviewed in much more detail in many parallel computing textbooks such as Quinn's [14].

1.2 Limitations of Performance Analysis

Although our definition for speedup and its derived quantities work well for theoretical complexity analysis, they all rely in some way on knowing the serial performance. With large-scale simulations today reaching orders of billions to trillions of elements [2,3,9,15], directly measuring serial performance is often impossible. The Gustafson-Barsis law needs only the serial fraction, but estimates for serial fraction such as the *Karp-Flatt* metric [12] require knowing the serial performance anyway.

Because of this issue, most studies attempt to assess scalability with a pair of trends named strong scalability and weak scalability [11]. *Strong scaling* demonstrates an algorithm's behavior by measuring its run time on a particular data set for various numbers of processing elements. Perfect strong scaling has a running time proportional to $1/p$. As we shall see in examples later, it is difficult to compare the quality of plotted curves to this perfect hyperbolic on both linear and log scales.

Per Amdahl's law, there are inevitable limits to strong scaling. In contrast *weak scaling* varies the problem size proportionally with the number of processing elements. It may not always be possible to keep the problem size proportional to the job size, which makes weak scaling more difficult. The metrics advocated in this paper simplify studying scalability by removing the dependence between problem size and number of processing elements. We encourage using them to sample the 2D parameter space of problem size and number of processing elements as widely as possible for a broader view of the scalability.

Some studies use an ad hoc version of speedup or efficiency that replaces the immeasurable $T^*(n)$ with some arbitrarily chosen measurement, usually the time run on the smallest number of processing elements. The problem with this approach is that the absolute meaning of "speedup" and "efficiency" changes between experiments in a study. Furthermore, the metric cannot be used in weak scaling because the problem size is not held constant.

Finally, some studies use rate in terms of the size of input computed per unit time rather than absolute run time to assess scalability [11]. Rate is formally defined as

$$R(n, p) = \frac{n}{T(n, p)} \tag{6}$$

Some analysts have discovered that rate, being essentially a reciprocal of time, provides a much better visual analysis of scaling, and it is an essential mechanism advocated in this position paper.

This paper establishes a more pragmatic definition and efficiency that can be easily measured empirically. Furthermore, we demonstrate how rate can be used as a proxy for speedup and can unify strong and weak scaling to provide a more complete analysis. These metrics are demonstrated using real performance data.

2 Deriving Efficiency from Cost Analysis

In this section we will use *cost*, a metric that is simple to measure, to define efficiency in lieu of the immeasurable speedup. Cost is intuitively the number of processing elements used multiplied by the amount of time they are used.

$$C(n, p) = p\, T(n, p) \tag{7}$$

Cost is sometimes used in theoretical algorithm analysis [10] and is often used for HPC allocations, which are typically measured in core-hours.

Clearly the most efficient algorithm will be the one that costs the least to run. Although we expect the cost to go up with the problem size, a perfectly scaled algorithm on a fixed input size will cost the same regardless of how many processors are used. That is, adding processors reduces the time proportionally. Given a strong scaling study on a problem of a particular size, we can identify the best (minimal) cost, $C^*(n)$, that uses p^* processing elements. With this best cost we can redefine efficiency as the ratio of this best cost to the actual cost.

$$E(n,p) = \frac{C^*(n)}{C(n,p)} \tag{8}$$

If we make the typical assumptions that the minimal cost is when the serial algorithm is run ($C^*(n) = T^*(n)$), then we observe that Eq. 8 simplifies to Eq. 2, making this definition of efficiency equivalent but broader than the traditional definition. And unlike the traditional definition of efficiency, determining efficiency from cost is straightforward at large scales.

3 Strong and Weak Scaling with Cost per Unit

Our previous definition of efficiency (Eq. 8) works well for strong scaling where the data size is constant but cannot be compared across different data sizes for weak scaling. To describe efficiency under weak scaling we define the new metric *cost per unit*, C_u. Cost per unit is the amortized computational cost for one unit of data.

$$C_u(n,p) = \frac{C(n,p)}{n} = \frac{p\,T(n,p)}{n} = \frac{p}{R(n,p)} \tag{9}$$

The important feature of cost per unit is that under perfect scaling the cost per unit is constant under any number of processing elements *or* data sizes. Thus, given multiple strong scaling studies over data of different sizes, we can find the best cost per unit, C_u^*, that uses p^* processing elements operating on data of size n^*. The best cost per unit is generally the minimum. We are not considering, however, outlier experiments where data sizes are not representative of practical runs. For example, it is well known that a sufficiently small data size could fit in the cache of a sufficiently large parallel job [8]. An experiment of this nature would report a much lower cost per unit but would be of little relevance to the scale of problems run in practice and therefore should be disqualified from ideal.

With this best cost per unit we can adjust the efficiency to be comparable across all possible configurations.

$$E(n,p) = \frac{C_u^*}{C_u(n,p)} \tag{10}$$

This this definition of efficiency allows us to combine strong and weak scaling studies into one unified analysis.

4 Rate as a Proxy for Speedup

Both efficiency and speedup are good metrics for parallel performance analysis. However, many analysts prefer using speedup, particularly for visual (chart) analysis. This is because good scaling shows an upward sloping speedup as jobs get larger whereas even a good scaling algorithm will show a gradual drop-off from a perfect efficiency of 1.

Although there is no way to compute the speedup at large scales, we can show that rate (Eq. 6) is a valid proxy for speedup. If we substitute rate for time in Eq. 1, we get the following.

$$S(n,p) = \frac{T^*(n)}{T(n,p)} = \frac{T^*(n)}{n} R(n,p) \tag{11}$$

We can observe that for a given problem size (i.e. n held constant) the speedup is proportional to the rate. This means that the rate curve will have the exact same shape as the speedup curve, and visually they will be identical with the appropriate scaling of the ordinate axis.

For a proper parallel performance analysis we need to compare our measured metrics with the ideal metrics. These ideal values are implicit in the definition of efficiency ($E_{\text{ideal}}(n,p) = 1$) and speedup ($S_{\text{ideal}}(n,p) = p$). The ideal rate can be derived from Eq. 10 by substituting the cost per unit with the rate (Eq. 9) and solving for rate when the efficiency is the optimal value of 1.

$$R_{\text{ideal}}(n,p) = \frac{p}{C_u^*} \tag{12}$$

Note that the curve for $R_{\text{ideal}}(n,p)$ is independent of n, which means we can use the same ideal rate for both strong and weak scaling analysis and can compare these rates with each other.

5 Examples

In the previous sections we provide mathematical derivations to show how to use rate as a proxy for speedup and to use cost per unit to find the efficiency and ideal rate across all scales. In this section we demonstrate using these metrics on real measured data. We can observe that the metrics of rate and efficiency make it easier to visually identify the behavior and trends at different scales of processing elements.

5.1 Gordon Bell Finalist

Our first data set comes from a study by Habib et al. [9], which is one of the 2013 Gordon Bell finalists. We choose this source because in addition to showing impressive scaling, the authors make many measurements across many scales and report the results completely enough to extract the information and continue analysis. In particular, we look at the performance data for scaling the full HACC code on Titan (Sect. 4.3.2 in the original paper).

Figure 1 on the left shows the performance data from the strong and weak scaling studies using a traditional time plot. The curves for the data are very similar for what we would expect for perfect scaling: a hyperbolic curve for strong scaling and a horizontal line for weak scaling. Habib et al. also provide an ad hoc metric of time over data size to unify the curve shape of the two plots, which is also replicated in Fig. 1 on the right. Again, both curves appear close to perfect.

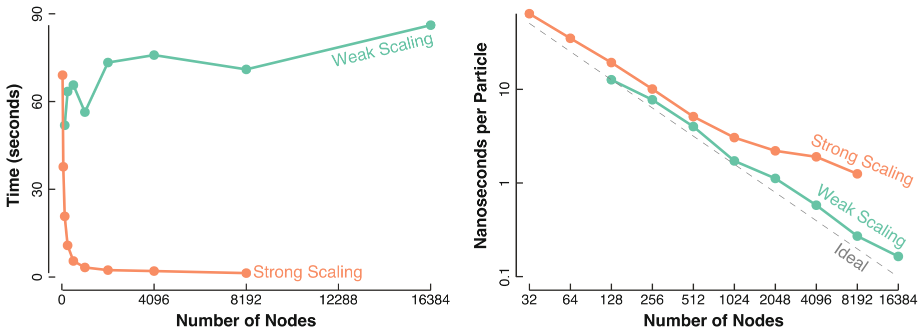

Fig. 1. Data from the Habib et al. [9] Gordon Bell finalist. The left chart shows the data using a traditional time metric. The right chart replicates the presentation of Fig. 3 in the original paper using an ad hoc metric and a log-log scale. Both charts present the data in a way to suggest near perfect scaling for both scaling studies.

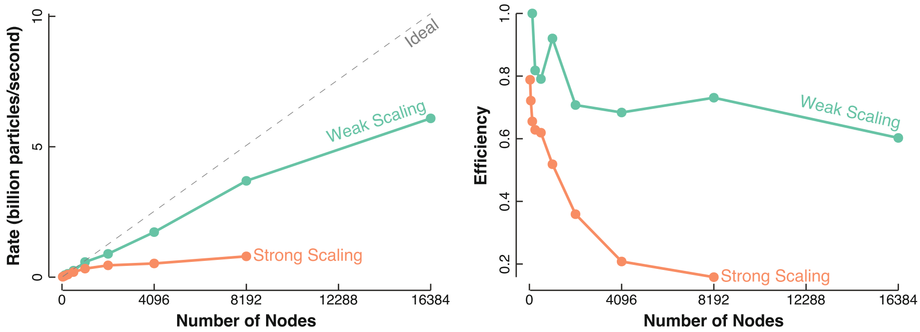

Fig. 2. Data from the Habib et al. [9] Gordon Bell finalist using the rate (left) and efficiency (right) metrics advocated in this paper. These plots give a more realistic and visually measurable representation of scaling than the charts in Fig. 1.

Figure 2 shows the same data using the rate and efficiency metrics advocated in this paper. The weak scaling is shown to diverge from ideal by a measurable fraction, which is to be expected when scaling over 3 orders of magnitude. The strong scaling study is shown to diverge very far from ideal, which is not at all apparent in the original charts.

5.2 Imperfect Scaling

Our second data set comes from a study by Oldfield et al. [13]. In this study a visualization algorithm has a high communication overhead and therefore has a known limit on its scalability. The study shows how transferring data between parallel jobs of different sizes can sometimes be faster than combining both in one large job.

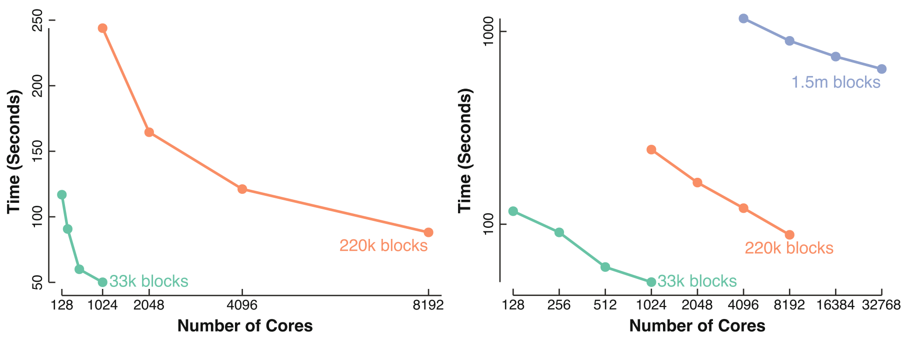

Fig. 3. Data from the Oldfield et al. [13] simulation-visualization integration study where the visualization (shown here) has a high communication overhead. These plots use the traditional method of showing the trend of run time using linear and log-log scaling.

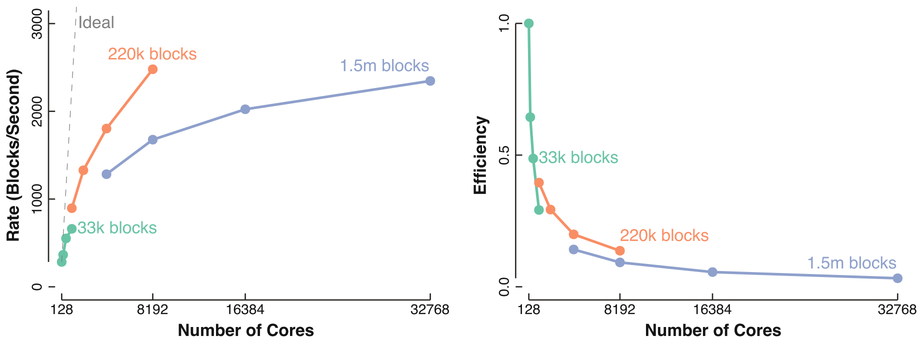

Fig. 4. Data from Oldfield et al. [13] using rate (left) and efficiency (right) to reveal that the communication overhead has a severe impact on the overall scalability.

A traditional plot of the time of the visualization component shown in Fig. 3 gives curves that suggest good scaling performance. However, when we show the same data using the rate and efficiency metrics, shown in Fig. 4, we can clearly see the effect the communication overhead has on the scalability of the algorithm. Without such a presentation, erroneous interpretation of the performance is sure to occur.

6 Discussion

In this paper we discuss limitations of traditional parallel performance analysis and problems with the current metrics often used. We provide derivations of rate as a proxy for speedup, ideal rate, and efficiency and demonstrate with real data how these provide accurate visual representations of scalability. As scientists, we should demand this high level of transparency and honesty in performance analysis.

With these observations, we provide the following recommendations for the visual display of parallel performance.

- Do not rely on running time for performance analysis. Instead use rate, efficiency, or both.
- Avoid using log-log scaling on plot axes, which hides major inefficiencies. If necessary, repeat linear plots at different scales.
- Rather than performing separate weak and strong scaling studies, incorporate them in one. Perform several strong scaling studies at different scales of data size. Then find an overall minimal practical cost per unit and plot all the measurements together as demonstrated in the figures in this paper.

As architectures continue to advance, many are beginning to advocate measuring electrical power instead of or in addition to speed [4]. As future work we advocate using a cost model for this as well. It would be interesting to derive efficiency in terms of optimal watt-hours per data unit (although rate may lose meaning).

The use of rate or efficiency to measure parallel performance is not itself a new technique. After all, the ubiquitous "FLOPS" measurement is itself a rate, and measurements given in data unit per time unit can be found throughout the literature. However, other metrics of varying effectiveness are also found in published material with little or no justification for the choice. The decision for scaling metrics should not be arbitrary; it can have enormous impact on the viability of the analysis. Our intention is to show that the metric visualized does matter, to provide a best practices for measuring scalability, and to explain why these metrics work better than others.

Acknowledgment. This material is based in part upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Scientific Discovery through Advanced Computing (SciDAC) program under Award Number 12-015215.

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000. SAND 2015-2890 C

References

1. Amdahl, G.M.: Validity of the single processor approach to achieving large scale computing capabilities. In: Proceedings of the AFIPS 1967, pp. 483–485, April 1967. doi:10.1145/1465482.1465560
2. Bernaschi, M., Bisson, M., Fatica, M., Melchionna, S.: 20 Petaflops simulation of proteins suspensions in crowding conditions. In: Proceedings of the SC 2013, November 2013. doi:10.1145/2503210.2504563
3. Bussmann, M., et al.: Radiative signatures of the relativistic Kelvin-Helmholtz instability. In: Proceedings of the SC 2013, November 2013. doi:10.1145/2503210.2504564
4. Cameron, K.W., Ge, R.: Generalizing Amdahl's law for power and energy. IEEE Comput. **45**(3), 75–77 (2012). doi:10.1109/MC.2012.92
5. Faber, V., Lubeck, O.M., White Jr., A.B.: Superlinear speedup of an efficient sequential algorithm is not possible. Parallel Comput. **3**(3), 259–260 (1986). doi:10.1016/0167-8191(86)90024-4
6. Grama, A.Y., Gupta, A., Kuma, V.: Isoefficiency: measuring the scalability of parallel algorithms and architectures. IEEE Parallel Distrib. Technol.: Syst. Appl. **1**(3), 12–21 (1993). doi:10.1109/88.242438
7. Gustafson, J.L.: Reevaluating Amdahl's law. Commun. ACM **31**(5), 532–533 (1988). doi:10.1145/42411.42415
8. Gustafson, J.L.: Fixed time, tiered memory, and superlinear speedup. In: Proceedings of the Fifth Distributed Memory Computing Conference, pp. 1255–1260 April 1990. doi:10.1109/DMCC.1990.556383
9. Habib, S., et al.: HACC: Extreme scaling and performance across diverse architectures. In: Proceedings of the SC 2013, November 2013. doi:10.1145/2503210.2504566
10. JáJá, J.: An Introduction to Parallel Algorithms. Addison Wesley, Boston (1992). ISBN 0-201-54856-9
11. Kaminsky, A.: Big CPU, Big Data: Solving the World's Toughest Computational Problems with Parallel Computing. Unpublished manuscript (2015), retrieved from http://www.cs.rit.edu/ark/bcbd
12. Karp, A.H., Flatt, H.P.: Measuring parallel processor performance. Commun. ACM **33**(5), 539–543 (1990). doi:10.1145/78607.78614
13. Oldfield, R.A., Moreland, K., Fabian, N., Rogers, D.: Evaluation of methods to integrate analysis into a large-scale shock physics code. In: Proceedings of the ICS 2014, pp. 83–92. June 2014. doi:10.1145/2597652.2597668
14. Quinn, M.J.: Parallel Programming in C with MPI and OpenMP. McGraw-Hill, New York (2004). ISBN 978-0-07-282256-4
15. Rossinelli, D., et al.: 11 PFLOP/s simulations of cloud cavitation collapse. In: Proceedings of the SC 2013, November 2013. doi:10.1145/2503210.2504565