Scaling Analysis

1 Overview

Most parallel applications can be run using a range of different core counts, but the relative performance of the application will likely diminish as the number of cores is increased. For example, a calculation may see a nearly exact twofold speedup when the number of cores is doubled. If the number of cores is quadrupled, however, it may only speedup by a factor of 3.7 (as opposed to 4).

This deviation from the expected speedup is normal for many parallel applications, but it means that you must decide carefully on the core-count to run your jobs at. The appropriate core count might differ between large and small applications, and it is thus important to measure performance at a range of core counts for all representative calculation sizes you intend to run. For example, you may intend to evolve multiple fluids models run on grids of different sizes. Alternatively, you may be working on a genomics project that entails comparing differently-sized sets of genomes depending on the study you are carrying out.

Typically, but not always, larger calculations will run efficiently on larger core counts than will a smaller calculation. Because of these considerations, we ask that you use your general-allocation access to acquire representative Summit-performance numbers for each calculation size you intend to perform. For each job size, run at a range of core counts, record the time required to perform a calculation at that core count, and report that information here.

You may choose to provide this data in tabular form, graphical form, or both. We will refer to this data when evaluating your allocation request. If you need help deciding on how to make these measurements, do not hesitate to contact rc-help@colorado.edu and ask for advice.

Please frame your allocation request in terms of either the *weak-scaling* or *strong-scaling* performance of your application.

2 Strong Scaling

Strong scaling refers to an application's performance when the total problem size is kept fixed, and the number of cores is varied. The questions asked in a strong-scalings study are essentially:

If I double the core count, does the calculation time fall by half?

If I quadruple the core count, does the calculation time fall by a quarter?

etc.

When performing these measurements, it is useful to be aware of the notion of of *ideal* performance and *efficiency*. The *ideal* calculation time is just the time you *expect* at a given core count based on the measured time at your smallest core count. If you measured a time of 10 seconds when using 2 cores, 2.5 second represents the *ideal* calculation time when running with 8 cores. The formula for ideal time is given by:

$$\text{Ideal Time on N cores} = (\text{Measured Time at Lowest Core Count}) \times (\text{Lowest Core Count}) / \text{N}.$$

The ideal calculation time can be used to evaluate a calculation's efficiency, namely the ratio of the measured time to the expected time:

$$\text{Efficiency} = \text{Ideal Time} / \text{Measured Time}$$

Strong-Scaling Data for Fluid Simulations			
Small-Run (128^3) Timings			
Cores	Measured Time (seconds)	Ideal Time (seconds)	Efficiency
6	60.5	60.50	1.00
12	31.84	30.25	0.95
24	16.44	15.13	0.92
48	8.90	7.56	0.85
96	4.98	3.78	0.76
192	3.15	1.89	0.60
384	2.36	0.95	0.40
Large-Run (512^3) Timings			
24	181.20	181.20	1.00
48	95.37	90.60	0.95
96	49.24	45.30	0.92
192	25.45	22.65	0.89
384	13.64	11.33	0.83
768	8.71	5.67	0.65
1536	5.6625	2.83	0.50

Table 1: Sample strong-scaling data (fixed problem size, variable core counts) for two simulation sizes. This data is illustrated graphically by the red circles in Figure 1

Efficiency is what you should use to judge the number of cores appropriate for a particular calculation size. Note that anything approaching 100% efficiency is rarely achieved in practice. Instead, you must choose the trade-off point between the efficient use of computational resources and the time you must wait for your calculation to complete.

NOTE: As a general rule of thumb, we suggest running with core counts that achieve a minimum of 80% efficiency. If you plan to run at a lower efficiency level, please explain why. This might happen, for instance, if your application requires large amounts of memory.

For an example of how you might present your timing data, see Table 1 and Figure 1. Those figures present performance data from a study involving small and large calculations.

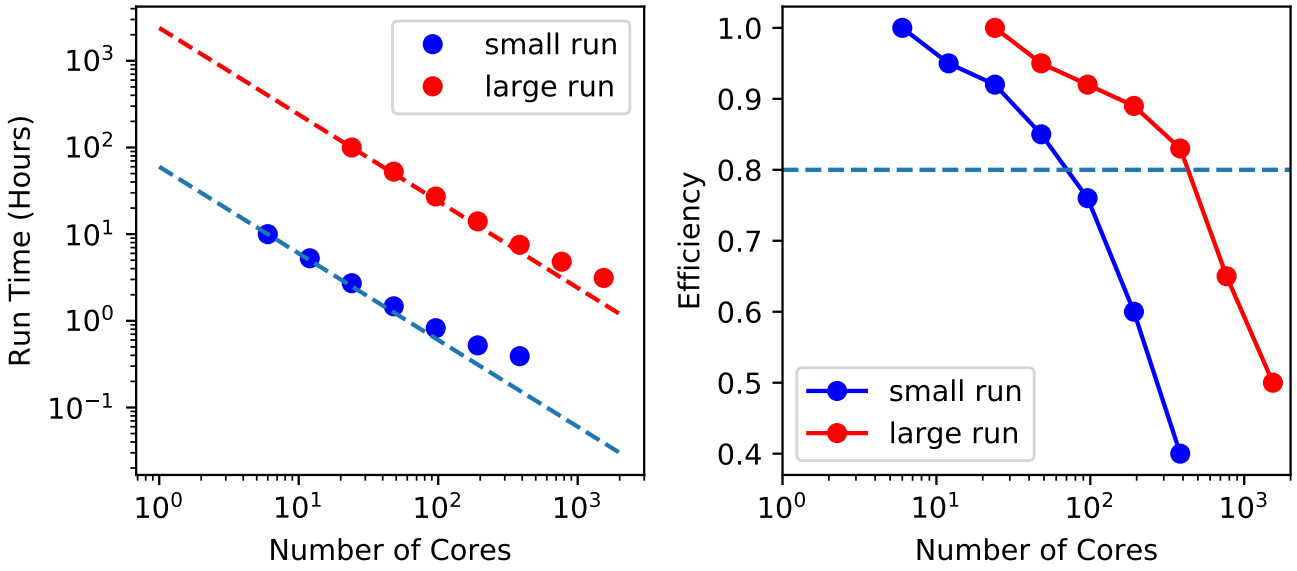

Figure 1: Sample performance data for small (blue) and large (red) runs (see Table 1). (*a*) Measured run time (circles) vs. number of cores. Ideal scaling for each case is indicated by the dashed lines. (*b*) Parallel efficiency as measured for our small and large runs. A blue, dashed reference line has been plotted to denote 80%.

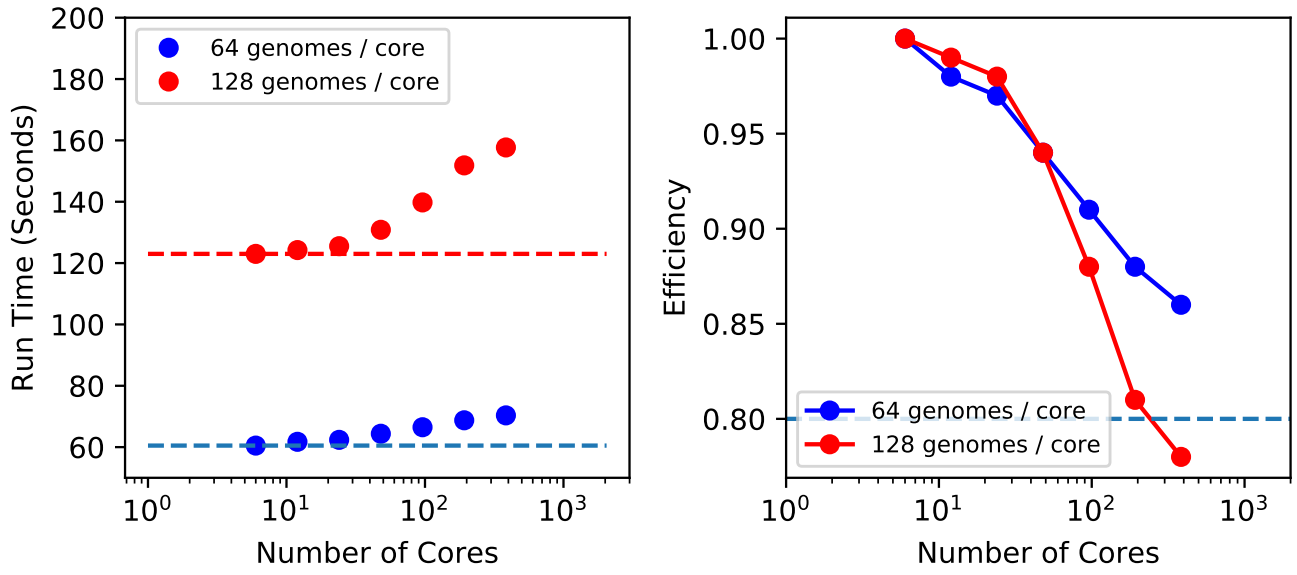

Figure 2: Performance data for our example weak scaling study (see Tables 2). (*a*) Measured run time (circles) vs. number of cores. Ideal scaling for each case is indicated by the dashed lines. (*b*) Parallel efficiency as measured for our small and large runs. A blue, dashed reference line has been plotted to denote 80%.

3 Weak Scaling

When performing a weak-scaling study, we are asking a complementary question to that asking in a strong-scaling study. Instead of keeping the problem size fixed, we increase the problem size relative to the number of cores. For example, we might initially run a genomics study comparing 128 genomes using 24 cores. Later, we might want to compare 256 genomes using 48 cores. Will the time remain constant, since the work per core has remained the same, or will the time increase due, for example, to increased communications overhead associated with the larger problem size or number of cores? A weak scaling study is one method of documenting this behavior for your application, allowing you to make an educated guess at the amount of computing time you need. We present sample data and plots for a weak-scaling study in Table 2 and Figure 2. Note that the notion of ideal time is different in this scenario; it is a constant number at all core counts.

Weak-Scaling Data for Genomics Study			
64 Genomes per Core			
Cores	Measured Time (seconds)	Ideal Time (seconds)	Efficiency
6.0	60.5	60.5	1.0
12.0	61.7346938776	60.5	0.98
24.0	62.3711340206	60.5	0.97
48.0	64.3617021277	60.5	0.94
96.0	66.4835164835	60.5	0.91
192.0	68.75	60.5	0.88
384.0	70.3488372093	60.5	0.86
128 Genomes per Core			
6.0	123.0	123.0	1.0
12.0	124.242424242	123.0	0.99
24.0	125.510204082	123.0	0.98
48.0	130.85106383	123.0	0.94
96.0	139.772727273	123.0	0.88
192.0	151.851851852	123.0	0.81
384.0	157.692307692	123.0	0.78

Table 2: Sample weak-scaling data (variable core count, fixed per-core problem size) for a genomics analysis. This data is illustrated graphically by the red circles in Figure 2