Lecture 14: Discussing Speedup

William Gropp
What is Speedup?

- In the simplest form,
  - \( \text{Speedup}(\text{code}, \text{sys}, p) = \frac{T_B}{T_p} \)

- Speedup measures the ratio of performance between two objects
  - Versions of same code, with different number of processors
  - Serial and vector versions
  - C and Fortran
  - Two algorithms computing the “same” result
Speedup Can Be Useful

• The key is choosing the correct *baseline* for comparison
  ♦ For our serial vs. vectorization examples, using compiler-provided vectorization, the baseline is simple – the same code, with vectorization turned off

• For parallel applications, this is much harder:
  ♦ Choice of algorithm, decomposition, performance of baseline case
Parallel Speedup

• For parallel applications, Speedup is typically defined as
  ♦ $\text{Speedup}(\text{code,sys}, p) = \frac{T_1}{T_p}$
  ♦ Where $T_1$ is the time on one processor and $T_p$ is the time using $p$ processors
• Can $\text{Speedup}(\text{code,sys}, p) > p$?
  ♦ That means using $p$ processors is more than $p$ times faster than using one processor
Speedup and Memory

• Yes, speedup on \( p \) processors can be greater than \( p \).
  ♦ Consider the case of a memory-bound computation with \( M \) words of memory
  ♦ If \( M/p \) fits into cache while \( M \) does not, the time to access memory will be different in the two cases:
    • \( T_1 \) uses the STREAM main memory bandwidth
    • \( T_p \) uses the appropriate cache bandwidth
Are there Upper Bounds on Speedup?

Let's look at a simple code. Assume that almost all of it is perfectly parallelizable (fraction $f$). The remainder, fraction $(1-f)$ can’t be parallelized at all.

That is, there is work that takes time $W$ on 1 process; a fraction $f$ of that work will take time $Wf/p$ on $p$ processors.

What is the maximum possible speedup as a function of $f$?
Question

• Stop here and try to compute the maximum speedup by computing $T_1$ and $T_p$ in terms of $p$ and $f$. 
Amdahl’s Law

- \( T_1 = (1-f)W + fW = W \)
- \( T_p = (1-f)W + \frac{fW}{p} \)
- Speedup = \( \frac{T_1}{T_p} = \frac{W}{((1-f)W+fW/p)} \)
- As \( p \) goes to infinity, \( fW/p \) goes to zero, and the maximum speedup is
- \( 1/(1-f) \)
- So if \( f = 0.99 \) (all but 1% parallelizable), the maximum speedup is \( 1/(1-.99)=1/(.01)=100 \)
Notes on Amdahl’s Law

• Its pretty depressing – if any non-parallel code slips into the application, the parallel performance is limited

• In many simulations, however, the fraction of non-parallelizable work is $10^{-6}$ or less
  ♦ Due to large arrays or objects that are perfectly parallelizable
$N_{1/2} –$ Another Measure

- When measuring performance as a function of a parameter (such as number of processors) one question is:
  - At what value of $p$ is half of the possible performance achieved?
  - For example, for parallel performance, how many processors are required to achieve half of the possible performance?

- Answer depends on the specific situation
Example $N_{1/2}$

- Consider the Amdahl’s law example
  - Maximum possible speedup at an infinite number of processes is $1/(1-f)$
- Question: At how many processes is half of the possible speedup achieved?
Answer for $N^{1/2}$

- $\frac{1}{2}$ of maximum speedup is $1/(2(1-f))$
- Speedup($p$) = $1/((1-f)+f/p)$
- To find $p$, set these equal (use their inverses)
  - $2(1-f) = (1-f) + f/p$
  - $1-f = f/p$
  - $P = f/(1-f)$
- E.g., for $f = .99$, $p = 9900$ (for a speedup of only 50!)
Overhead and Performance

• $N_{1/2}$ a convenient way to look at performance whenever
  ♦ $T = \text{overhead} + cn$

• In the Amdahl’s law case, the overhead is the serial (non-parallelizable) fraction, and the number of processors is $n$

• In vectorization, $n$ is the length of the vector and the overhead is any cost of starting up a vector calculation
  ♦ Including checks on pointer aliasing, pipeline startup, alignment checks