Performance
CS 540- High Performance Computing

Spring 2017

Performance Analysis

- Measure the gain in application performance, leveraging multiple processors/threads
- Performance gain is not perfect and not always guaranteed
- Limitations of parallel computing
- Very simple techniques for calculating performance gain (i.e. speedup, Amdahl's law)

Speedup

- The performance gain achieved by running your algorithm/application in parallel (i.e. multiple cores)
- The length of time it takes a program to run on a single processor divided by the time it takes to run on multiple processors.
- speedup ranges between 0 – p, p is the number of processors

Speedup

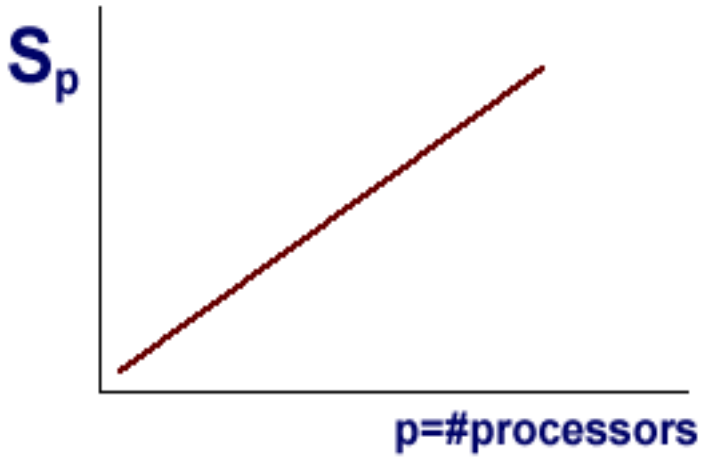
- Very simple method for calculating speedup

$$S_p = \frac{\text{time for a single processor run}}{\text{time using p processors}}$$

Source: National Center for Supercomputing Applications

Linear Speedup

- If one processor executes a specific task in time *t* and the number processors, *p*, can do the task in time t / p
- For example: if 4 processors improve runtime by a factor of 4.



Source: National Center for Supercomputing Applications

Scalability

- The ability to achieve performance proportional to the number of processors used is called, *scalability*.
- Ideally, performance should improve as your parallel code leverages multiple processor
- Computing speedup is a good indicator of how efficient a program scales as the number of processor cores increases.

Efficiency

- The average speedup per processor core .
- Efficiency is a fraction that ranges between 0 and 1

$$E_p = \frac{S_p}{p}$$

Source: National Center for Supercomputing Applications

Amdahl's Law

- An alternative formula for calculating speedup
- Named after Gene Amdahl, well respected computer scientists
- The law states speedup will be limited by its percentage (fraction) of sequential code, regardless of the number of processor cores used in a parallel application.

Amdahl's Law cont.

- Defines speedup with p processors as follows:

$$S_p = \frac{1}{f + (1 - f) / p}$$

Source: National Center for Supercomputing Applications

Amdahl's Law cont.

- The term *f* stands for the fraction of instructions executed sequentially with one processor
- The term *(1 - f)* stands for the fraction of instructions executed in perfect parallelism with ***p*** processors.
- The sequential fraction of code, ***f***, ranges between 0 and 1.
- If *f* is 0, there is no sequential code.
- If *f* is 1, there is no parallel code, then speedup is 1.
- Amdahl's speedup ranges between 1 and p, where **p** is the number of processors used in a parallel processing application.

Poor Performance

- When the calculated speedup is less than one, it’s a good indicator of poor performance.
- the parallel code runs slower than the sequential code.
- When there isn't enough processing, or work, to be completed by each processor core.
- The computing overhead of creating and managing the concurrent threads offsets the performance gain of parallel computing.

Speedup Limitations

- **Too much I/O,** when the code is I/O bound. too much I/O compared to the amount of computation.
- **Wrong algorithm,** when the numerical algorithm is not appropriate for a parallel computer.
- **Too much memory contention,** when there is too much memory contention.
- **Wrong problem size,** when the date set is too small to take advantage of a parallel computer architecture.

Speedup Limitations

- **Too much sequential code,** when there's too much sequential code. This is proved by Amdahl's Law.
- **Too much parallel overhead**, when there is too much parallel overhead compared to the amount of computation. Examples are the additional CPU cycles accumulated in creating parallel regions, creating threads, synchronizing threads, blocking threads.
- **Load imbalance,** when the processors have different workloads. The processors that finish early will be idle while they wait for other processors cores to complete or catch up.

References

1. Blaise Barney, Lawrence Livermore National Laboratory, https://computing.llnl.gov/tutorials/parallel_comp/#Overview
2. https://www.citutor.org/index.php, Parallel Computing Explained