Project Jupyter: Computational Narratives as the Engine of Collaborative Data Science

07 JULY 2015

Vocabulary

• Parallel computing

• Concurrency
Parallelism for a single query

Think of motivation in one of two equivalent ways
  a) Lower latency to run the query on a whole dataset
  b) Higher throughput in terms of records or documents processed per second
Metric for performance comparison: Time

My program runs in 100 seconds

If I “parallelize it” on 10 processors I saw that it runs in 12 seconds

What is the speedup?

\[ \frac{T_{\text{serial}}}{T_{\text{parallel}}} = \frac{100}{12} = 8.33X \]
Predicting parallel running time \( T_{\text{par}} \) from serial running time

My program runs in \( T_{\text{ser}} = 100 \) seconds

If I “parallelize it” on 10 processors, how fast will it run (i.e., what is \( T_{\text{par}} \)?)

\[
T_{\text{improved}} = T_{\text{original}} \ast \left( (1 - r) \ast 1 + r \ast \frac{1}{s} \right)
\]

\( r = \) fraction of program that is able to be improved
\( s = \) speedup when applying the improvement

In this form, it is called **Amdahl’s law**: says your speedup is limited by how much of the program is improved (e.g., parallelize)
Amdahl’s law applied to parallelization

What is the speedup of P processors over 1 processor in this query? (derived from last meeting)

A

B

h is a hash
Analyzing parallel algorithms

• Sometimes the speedup of the parallel portion is not just a linear function of the serial version

• to calculate $T_{parallel}$ when there is communication between processors, we don’t necessarily know $r$ and there might even be other terms for parts of program sped up by different amounts

$T_{improved} = T_{original} \ast ((1 - r_1 - r_2) \ast 1 + r_1 \ast 1/s_1 + r_2 \ast 1/s_2)$

• How do we more precisely analyze a parallel algorithm to find running time?

• Abstract machine models
One of the foundational parallel machine models: Parallel Random Access Machine (PRAM)

- All processors are attached to a shared memory
- Memory access takes 1 step
- More realistic variants of PRAM incur greater cost for “conflicting” memory accesses
- Used very often for understanding the speedup limits of parallel algorithms; not very realistic
Quick background: A sequential abstract machine model you already know

- RAM: random access memory
- just like any other computational step, accessing memory is cost of 1
Candidate Type Architecture (CTA)

- Observation: accessing different parts of memory has different latencies (more realistic than PRAM)
- Local memory: 1 step, remote memory $\lambda >> 1$
- $\lambda$ depends on the machine, so its value may change what the best algorithm is

$\text{RAM} = \text{sequential processor} + \text{memory}$
One of the foundational parallel machine models: Bulk synchronous parallel (BSP)

this abstract machine does not support as many algorithms as CTA, but it is simpler

(see blackboard notes)

https://en.wikipedia.org/wiki/Bulk_synchronous_parallel
Summary

• speedup is $\frac{T_{\text{serial}}}{T_{\text{parallel}}}$
• speedup is limited by the fraction of the program that is parallelizable (in general, improveable)
• for more precise analysis of parallel programs, which involve nontrivial communication, we use an abstract machine model to analyze a parallel program
• BSP is a simple yet practical model
  • it is a common algorithmic framework used for parallel database operations
Administriivia

• HW8 out, due 12/9
  • may work closely with another student but this time will submit your own

• Debug Your Brain, special edition
  • 5-7pm in 134 TH
  • lots of lab space to work