

A scenic view of the University of Colorado Boulder campus. In the foreground, a large, historic red brick building with a central tower and arched windows is partially obscured by trees with vibrant autumn foliage in shades of yellow, orange, and green. In the background, a massive, rugged mountain with steep, rocky slopes and patches of evergreen forest rises against a blue sky with light, wispy clouds. An American flag flies from a tall pole on the roof of the central tower.

High Throughput Computing

Be Boulder.



University of Colorado **Boulder**

High Throughput Computing

- Thus far: High Performance Computing (HPC)
- Typical HPC: employ multiple processors to
 - Solve a problem faster
 - Solve larger problems
- Today: High Throughput Computing (HTC)
- Typical HTC:
 - Multiple small jobs spread across many processors

High Throughput Computing

- HTC useful when have many small jobs that require little computational power or memory
- Jobs are typically serial, and not parallel
- HTC advantage:
 - Small serial jobs can fill in the “gaps” left by large parallel jobs
 - E.g., Open Science Grid
 - Effectively parallel: batch of jobs completes faster when spread across multiple cores
- Example: Image analysis

Advantages and Disadvantages

- Advantages
 - Simplicity
 - Much easier to match one task to one CPU rather than many at once
 - Doesn't require knowledge of parallelization for programmer
- Disadvantage
 - Your HPC center might not be set up ideally for HTC
 - Might not allow for node sharing
 - No batch submission system to manage multiple small jobs
 - Possibly requires heavy scripting to manage workflow

HTC Mechanics

- No real tricks
- Break down your problem and then submit a lot of smaller jobs
 - For example, if you are analyzing 1 million images, rather than submitting one job to analyze all 1 million images, submit one thousand jobs that analyze 1000 images each
- The resource manager (i.e., Slurm) should take care of the rest
- Is HTC appropriate? Problem dependent!
 - Is the serial execution time reasonable?
 - Can the problem fit into one core's worth of memory?