CENTER FOR HIGH THROUGHPUT COMPUTING

Advance Your Research: Scale Out Your Computing

OSG User School 2023
August 7, 2023

Why Scale Out Your Computing?

Richard Ayoade in the IT Crowd, Series 1, Episode 2
https://giphy.com/gifs/richard-ayoade-it-crowd-maurice-moss-dbtDDSvWErdf2

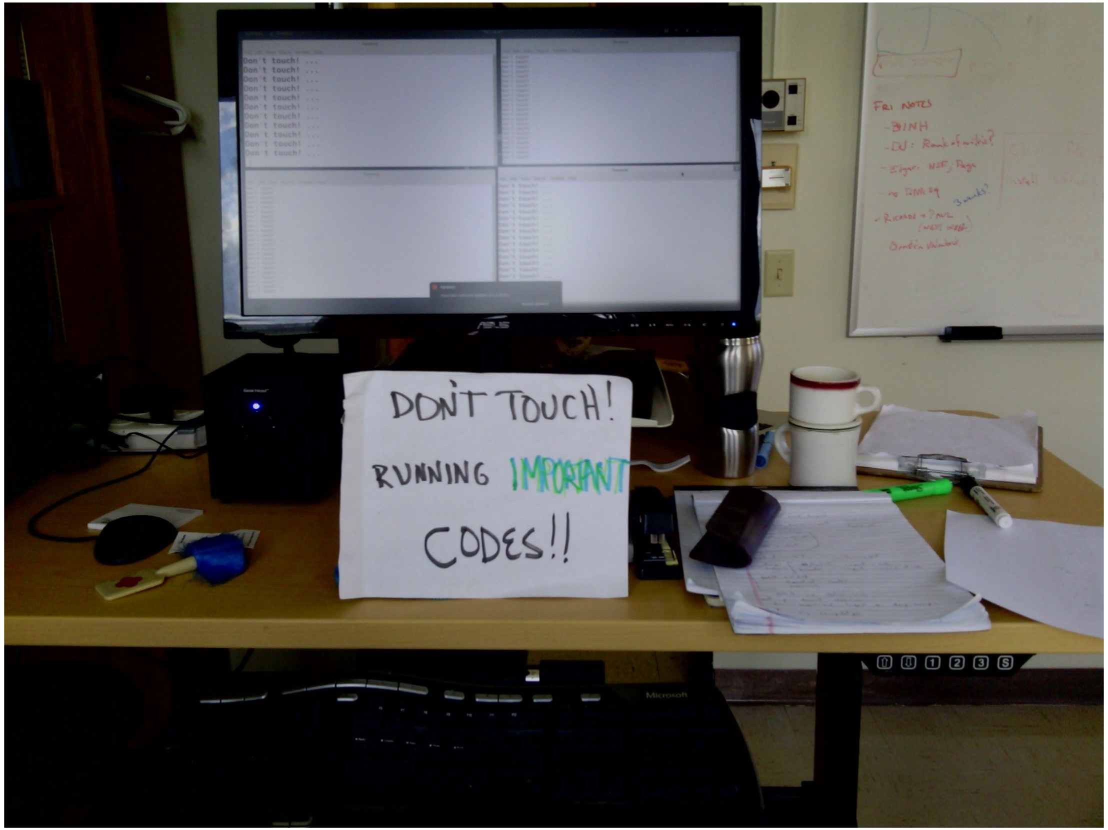


Photo credit Steve Goldstein

Dr. Timothée Poisot
@tpoi

Just use this simple heuristic: if it's slow and the computer goes WOOOOOOOOSH, the bottleneck is the CPU; if it's slow and the mouse stop moving, the bottleneck is the memory.

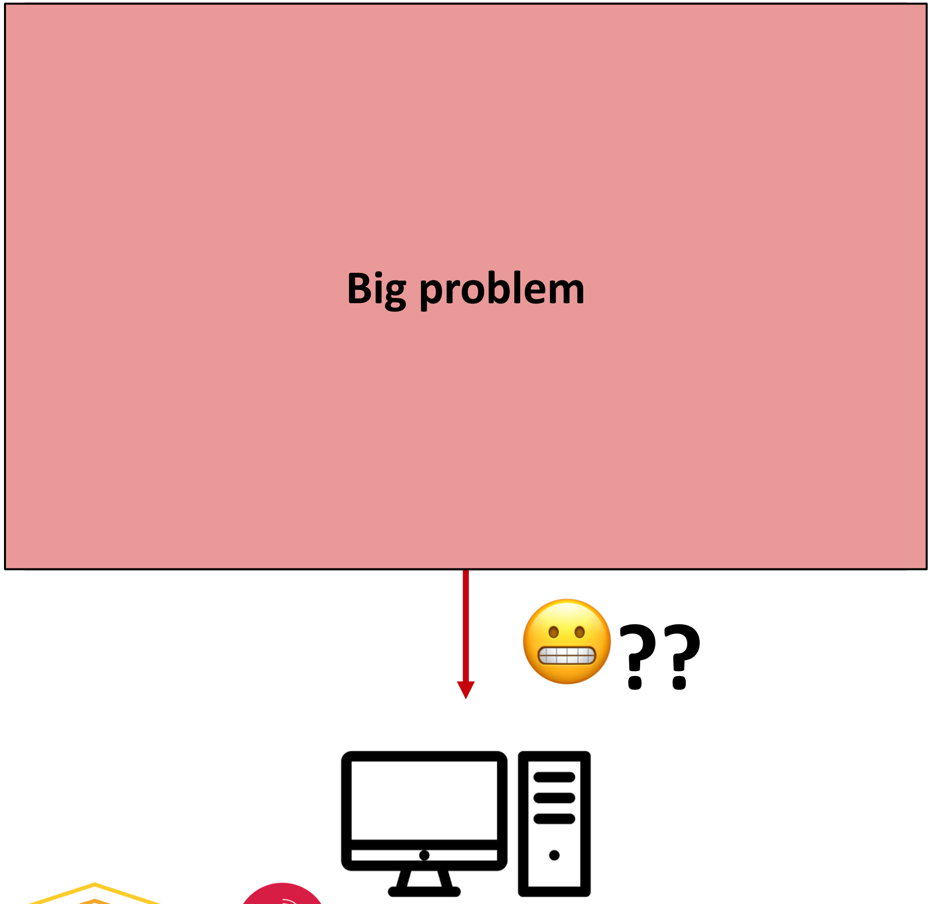
https://twitter.com/tpoi/status/1169692855201402885

Big Problems

How do you solve a big computational problem?

For example:

- Lots of data
- Thousands of parameters
- Many grid points

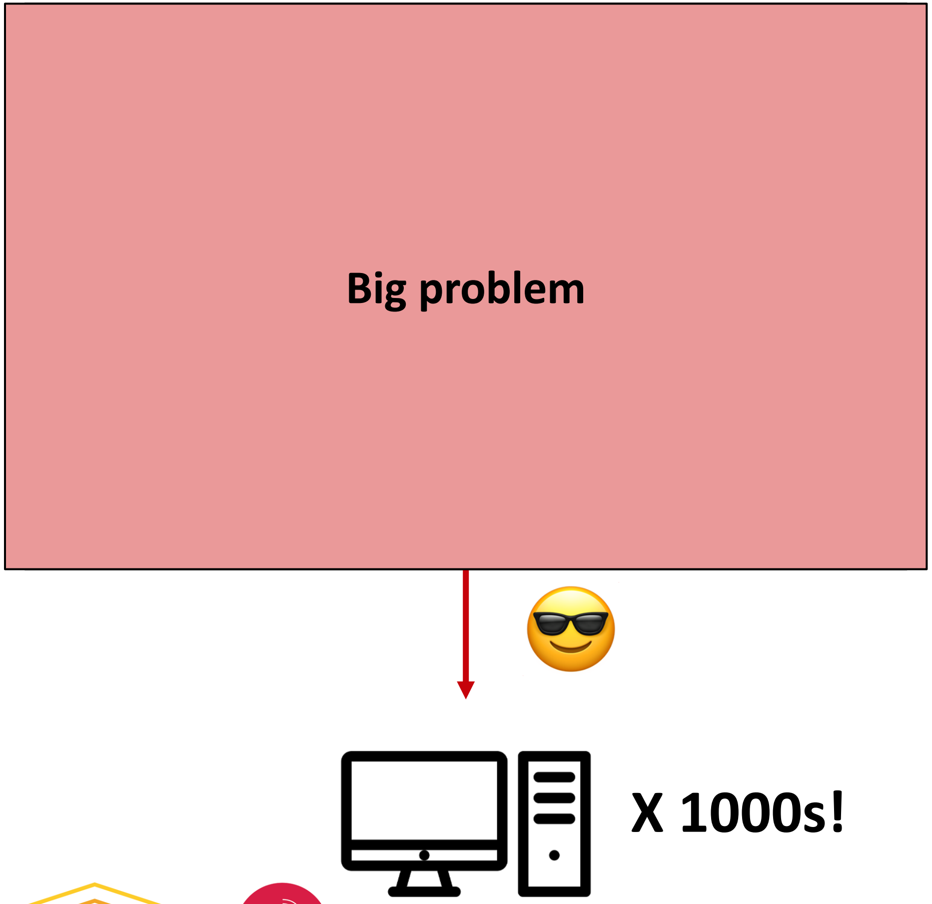


Our Goal: Scaling Out

We want to be able to tackle big problems.

We want to go from using a small number of resources to a LOT of resources.

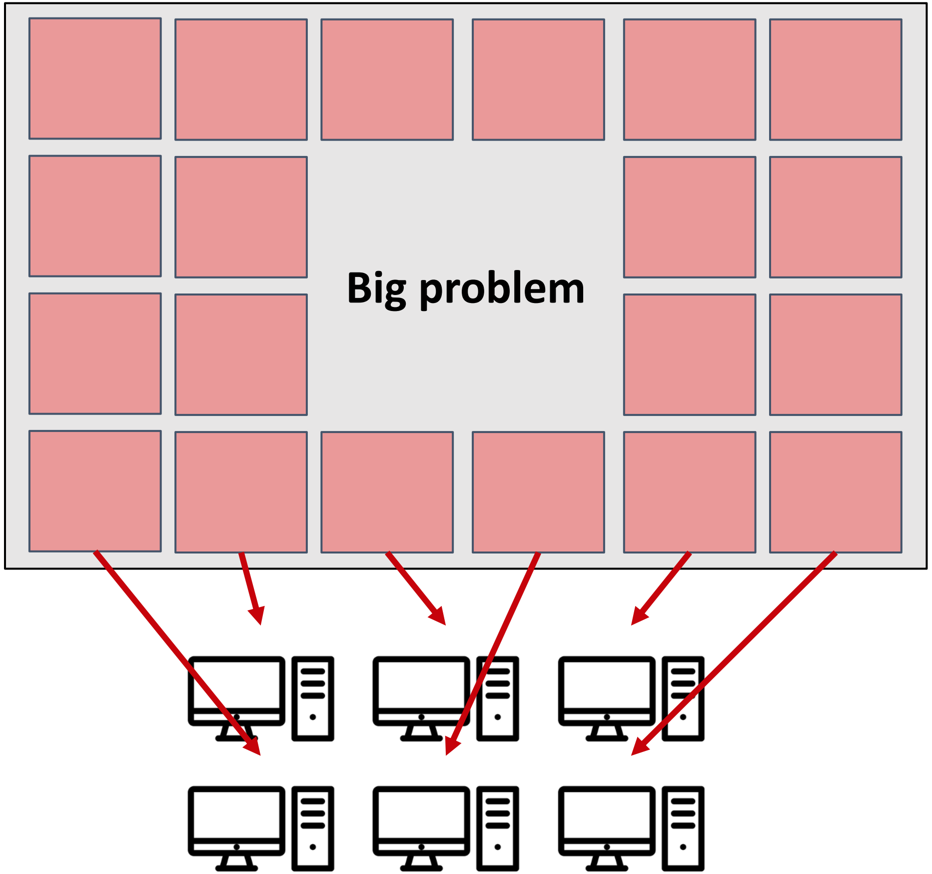
There is one strategy to do this with computers: divide and conquer



Big Solutions

To solve a big problem, break it into many sub-problems and use more/specialized computers.

There are different ways to do this.



Large Scale Computing Approaches

A bigger computer

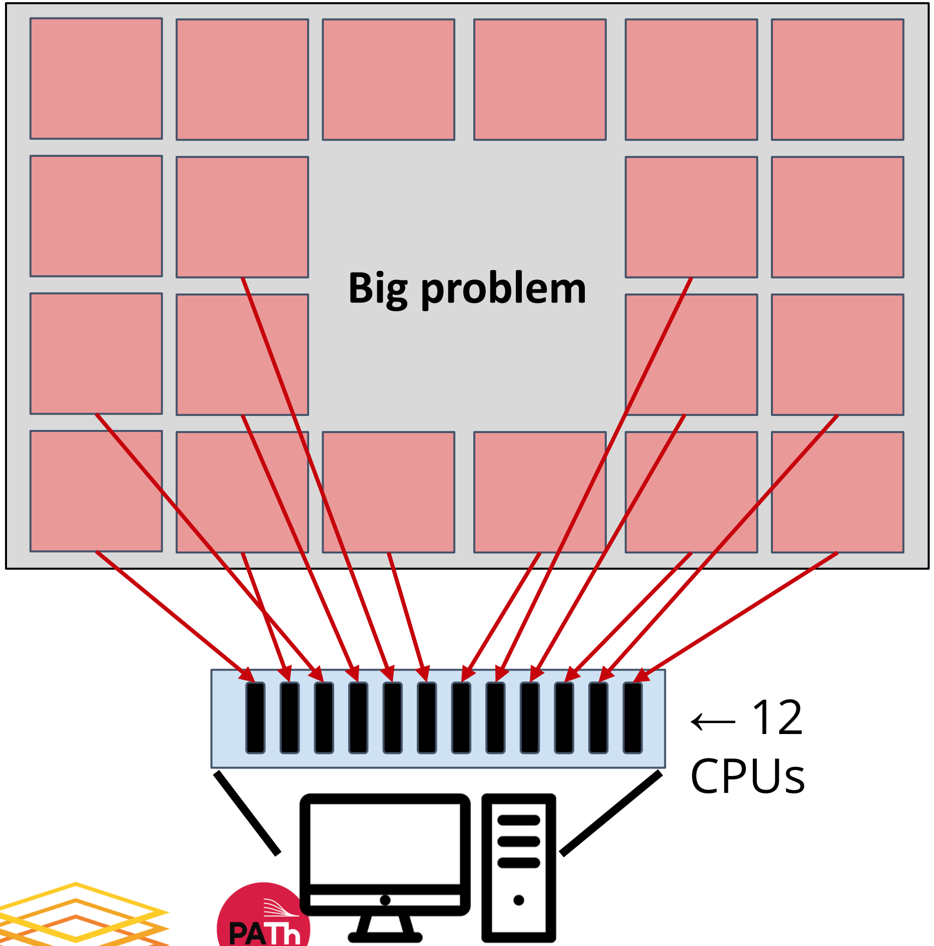
Sometimes, getting a larger computer can solve your problem - more CPUs, more memory, more disk space.

- Workstation
- Server that requires remote log in

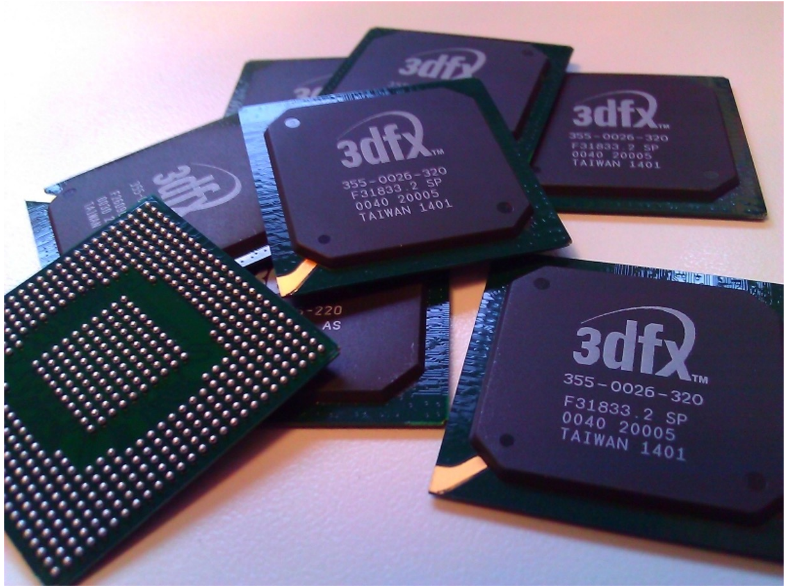
Option 1: More CPUs

Run sub-problems on individual CPUs.

Requires programming this capability into the code.



Specialized computer components (like GPUs)



GPUs are a common way to accelerate certain kinds of computing.

Like before, can be used from a:

- Workstation
- Server

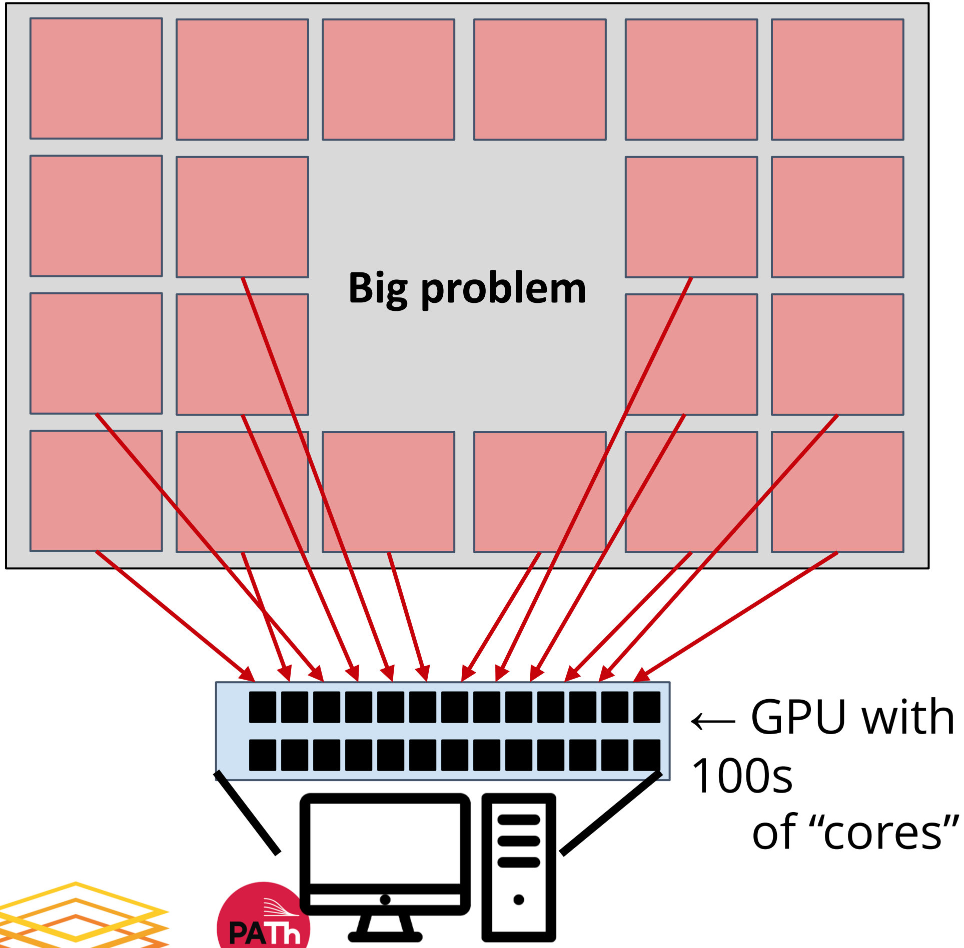
Option 2: Use a GPU

Run sub-problems on "cores" inside a GPU card.

Like before, requires programming this capability into the code.

Examples:

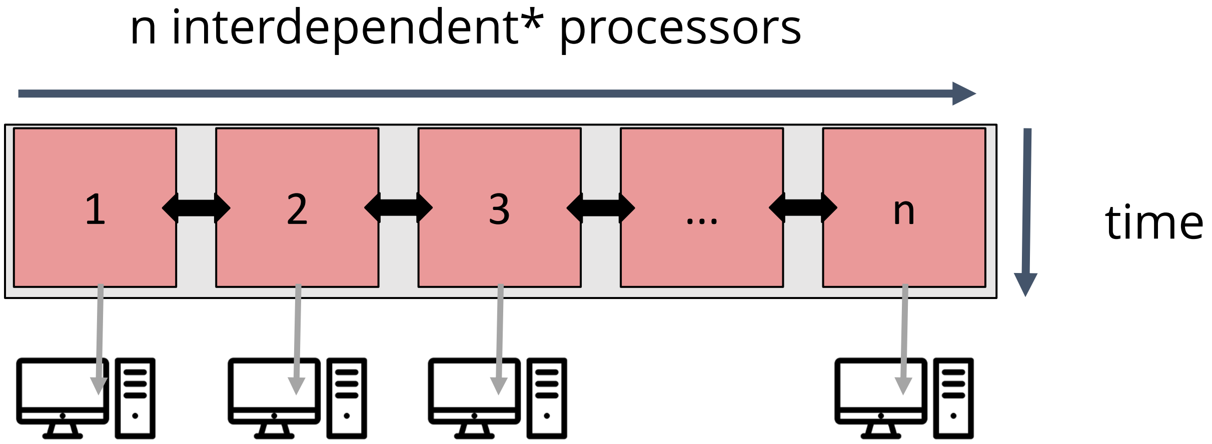
- Neural networks and other deep learning algorithms
- Certain molecular dynamic codes



Use more computers at once.

This is the idea behind the large-scale computing systems like the Open Science Pool.

Option 3: High Performance Computing (HPC)



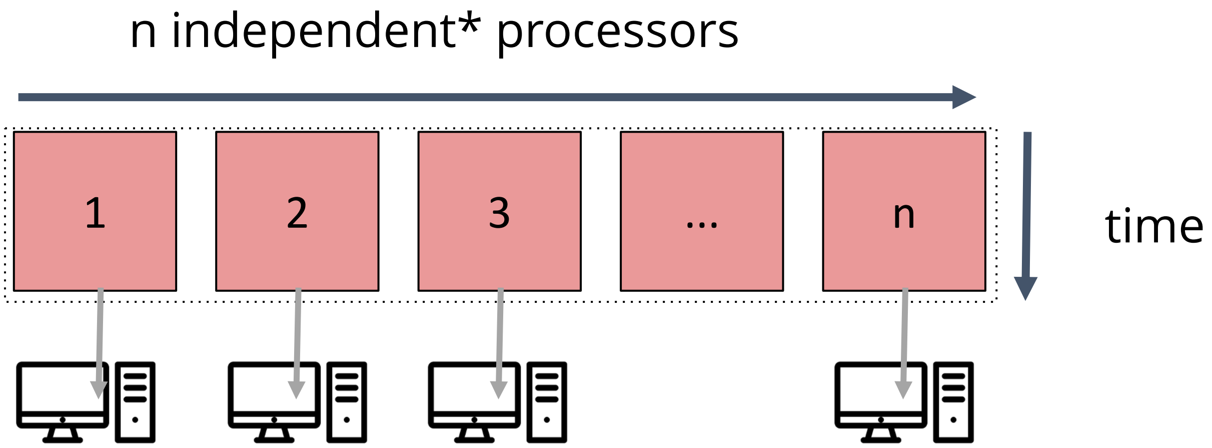
Solve sub-problems by distributing across multiple CPUs on multiple computers.

Also requires special coding (MPI) to coordinate calculations across multiple computers.

HPC Examples:

- Complex physical simulations with many inter-connected components.
- Optimization problems that use many local calculations to drive the global optimization.

Option 4: High Throughput Computing (HTC)



Sub-problems are independent and self-contained, can be run on many computers.

Most scalable.

No special programming needed.

HTC Examples:

- Test many parameter combinations
- Analyze multiple images or datasets
- Do a replicate/randomized analysis
- Align genome/RNA sequence data

One of our favorite HTC examples: baking the world's largest/longest cake

In computational terms: solving a big problem (the world's longest cake) by executing many small, self-contained tasks (individual cakes) and joining them.

Photos: Arun Sankar via https://www.theguardian.com/world/2020/jan/16/indian-bakers-rise-to-task-of-making-worlds-longest-cake

Use Case 1: Random Simulations

- **Typical task**: running a simulation
- **One computation needs**:
  - 1 core, 512Mb of memory
  - almost no data
  - 10 seconds
- **How many times?** Millions
- **Are there multiple steps involved?** No.
- **Bottleneck**: total number of simulations

Mei Monte Carlo

Needs to run many random simulations to estimate a statistical value.

Use Case 2: Analyzing Multiple Files

- **Typical task**: analyzing a data sample
- **One computation needs**:
  - 1-16 cores, 8 – 32GB of memory
  - Input of 5GB, output of 10GB
  - 8-10 hours for whole pipeline
- **How many times?** Dozens (number of samples)
- **Are there multiple steps involved?** Yes.
- **Bottleneck**: Size of data, number of samples, certain steps

Ben Bioinformatics

Needs to process 100s of genomic data files.

Discussion

See handout.

OSG School = HTC School

- This week is all about learning more about the practice of high throughput computing:
  - Job submission + troubleshooting (Mon/Tues)
  - Software portability (Tues)
  - Data handling (Wed)
- On two different high throughput computing resources
  - The PATh Facility (Mon)
  - the Open Science Pool (Tues onward)

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. 2030508. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.