Introduction to High Throughput Computing

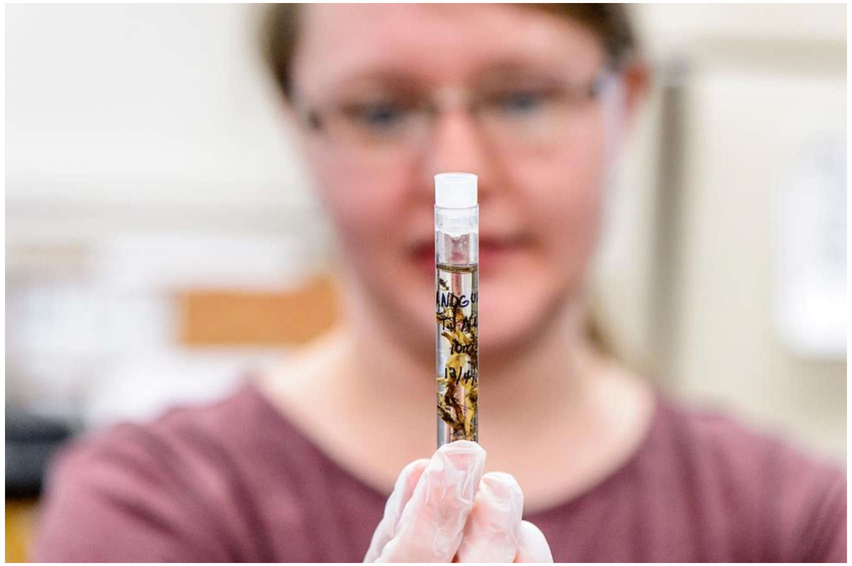
Christina Koch

OSG School 2025

June 23, 2025

Researcher Problems

Researcher #1



- New research student
- Working in plant pathology lab, studying plant genomes
- Can run first step of pipeline on one sample
- Now has 50 samples to run

Researcher #2

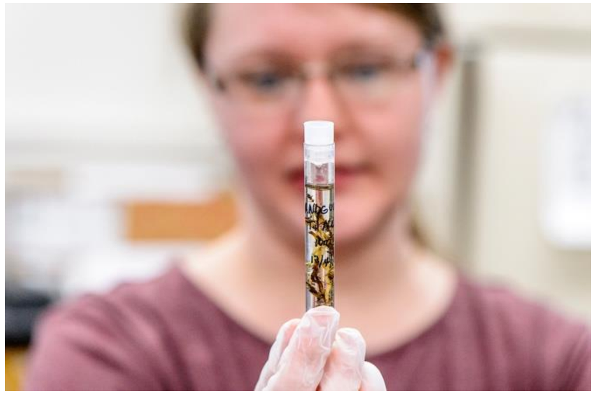
- Starting master's project
- Using self-written model which predicts accuracy of a medical trial design
- Model takes 3-4 hours to run
- Want to test many designs (each design is expressed as a combination of parameters)

Researcher #3

- Member of a large physics collaboration
- Want to predict (with probability) behavior of particle in detector
- Collaboration has particle simulation code already
- Probability estimate comes from running millions of particle simulations

What do they have in common?

Each researcher has a (non-ordered) list of **tasks** that would take too long to run sequentially on their local computer.



Running **analysis pipeline** for each sample.

Running a **simulation** for each parameter combination

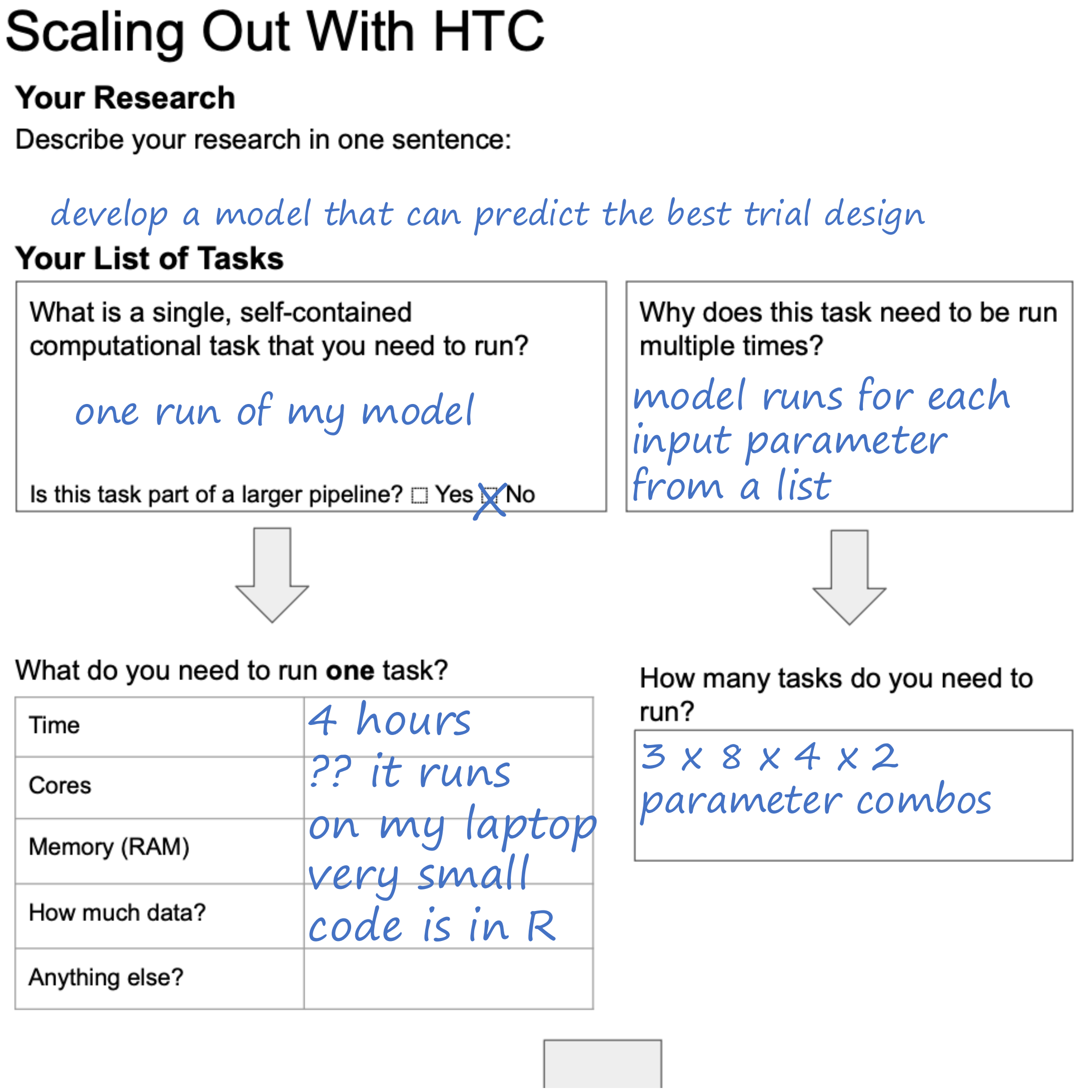
Running millions of **simulations**.

Your Turn

In the worksheet, write down the following

- A one-sentence summary of your research
- A typical computational **task**
  - This should be the smallest *self-contained* piece of your workflow
- What is your list of tasks?
  - “I need to run <computational task> for each <list of inputs>”
- Estimate how computationally intense *one* task is
- How many tasks do you have?

Example



Scaling Out With HTC

Your Research

Describe your research in one sentence:

develop a model that can predict the best trial design

Your List of Tasks

What is a single, self-contained computational task that you need to run?

one run of my model

Is this task part of a larger pipeline? □ Yes ☒ No

Why does this task need to be run multiple times?

model runs for each input parameter from a list

What do you need to run **one** task?

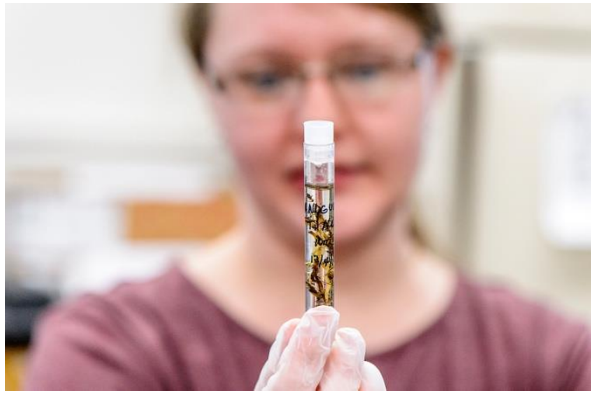
Time	4 hours
Cores	?? it runs
Memory (RAM)	on my laptop
How much data?	very small
Anything else?	code is in R

How many tasks do you need to run?

3 x 8 x 4 x 2 parameter combos

What do they have in common?

Each researcher has a (non-ordered) list of **tasks** that would take too long to run sequentially on their local computer.



Running **analysis pipeline** for each sample.

Running a **simulation** for each parameter combination

Running millions of **simulations.**

Why do we care?

Each researcher has a (non-ordered) list of tasks that would take **too long to run sequentially on their local computer**.

Scaling Up

Each researcher has a (non-ordered) list of tasks that would take **too long to run sequentially on their local computer**.

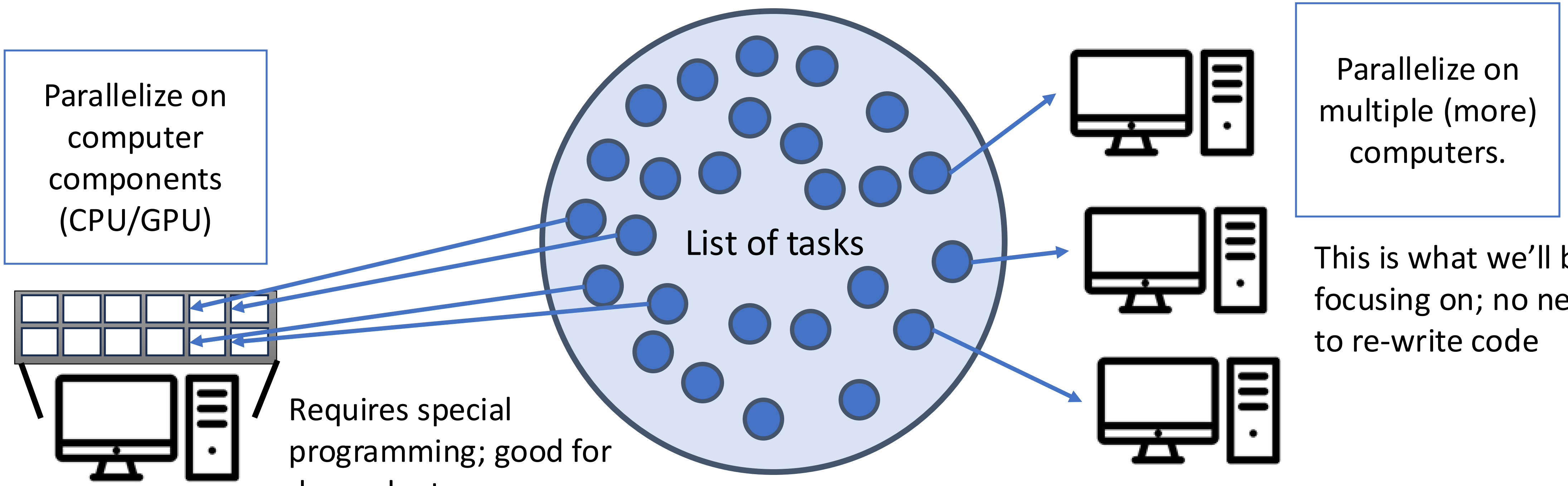
Don’t let computing be a barrier to your research!! We want to be able to tackle big problems.

The strategy to speed things up with computers is ALWAYS to run in parallel.

Then you can go from using a small number of resources (one computer) to a LOT of resources (for example, many computers).

Running in Parallel

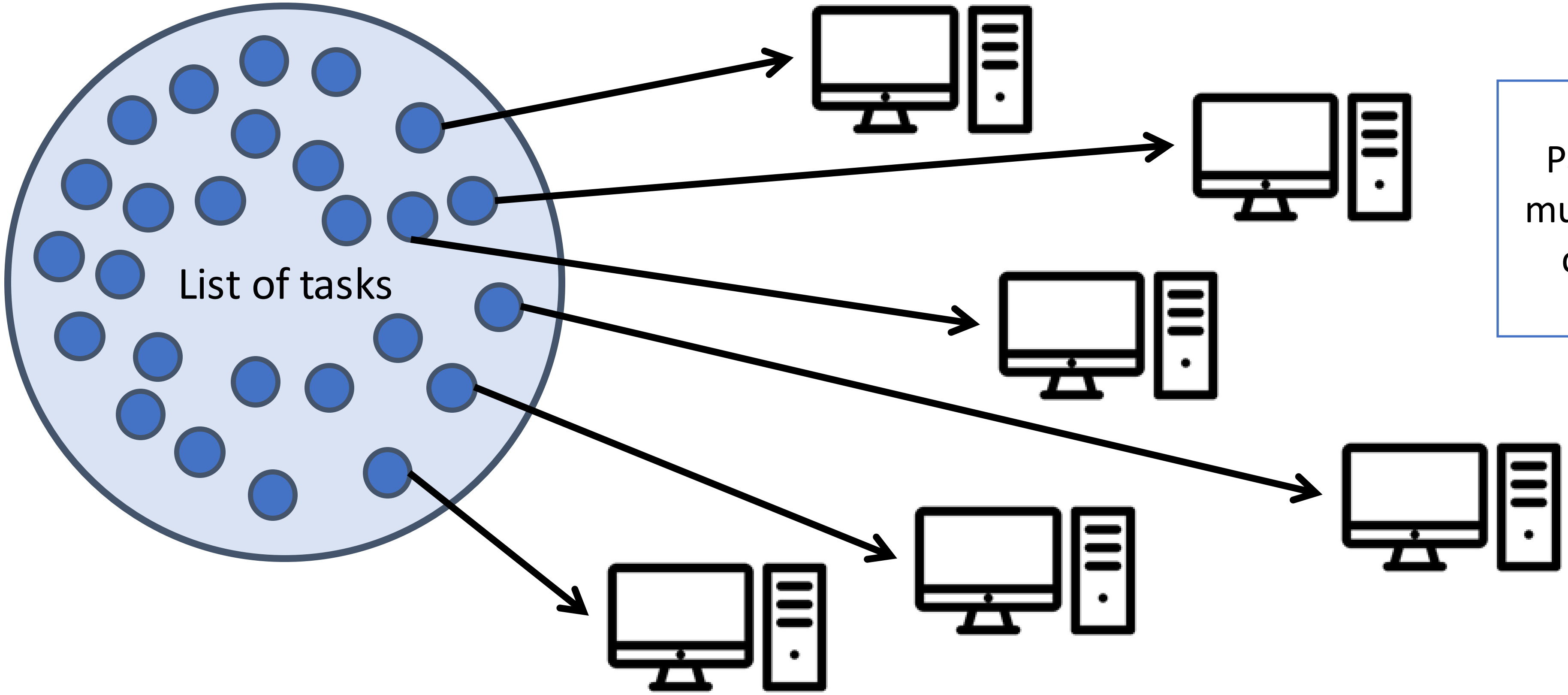
There are different ways to parallelize in computing:


Parallelize on computer components (CPU/GPU)
List of tasks
Parallelize on multiple (more) computers.

Requires special programming; good for dependent or correlated sub-tasks

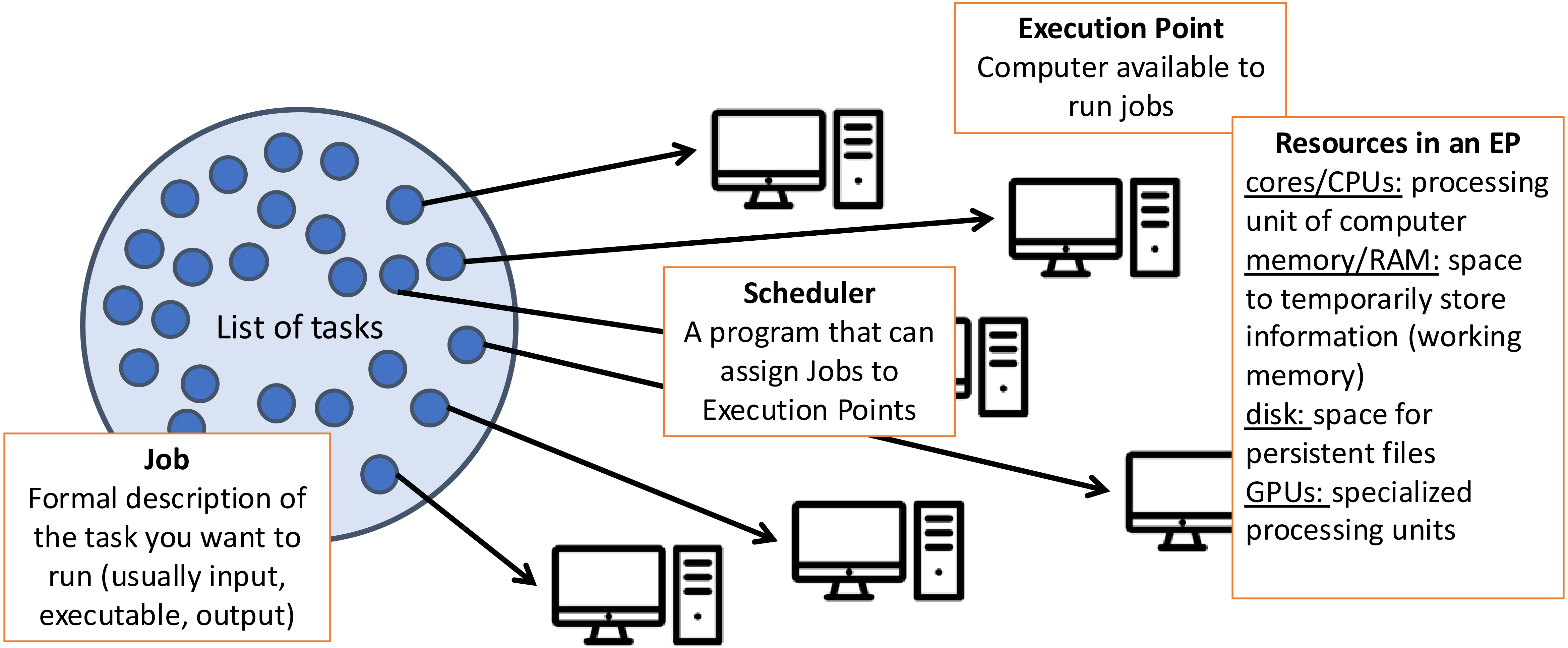
This is what we'll be focusing on; no need to re-write code

High Throughput Computing (HTC)



Parallelize on multiple (more) computers.

Some Terms



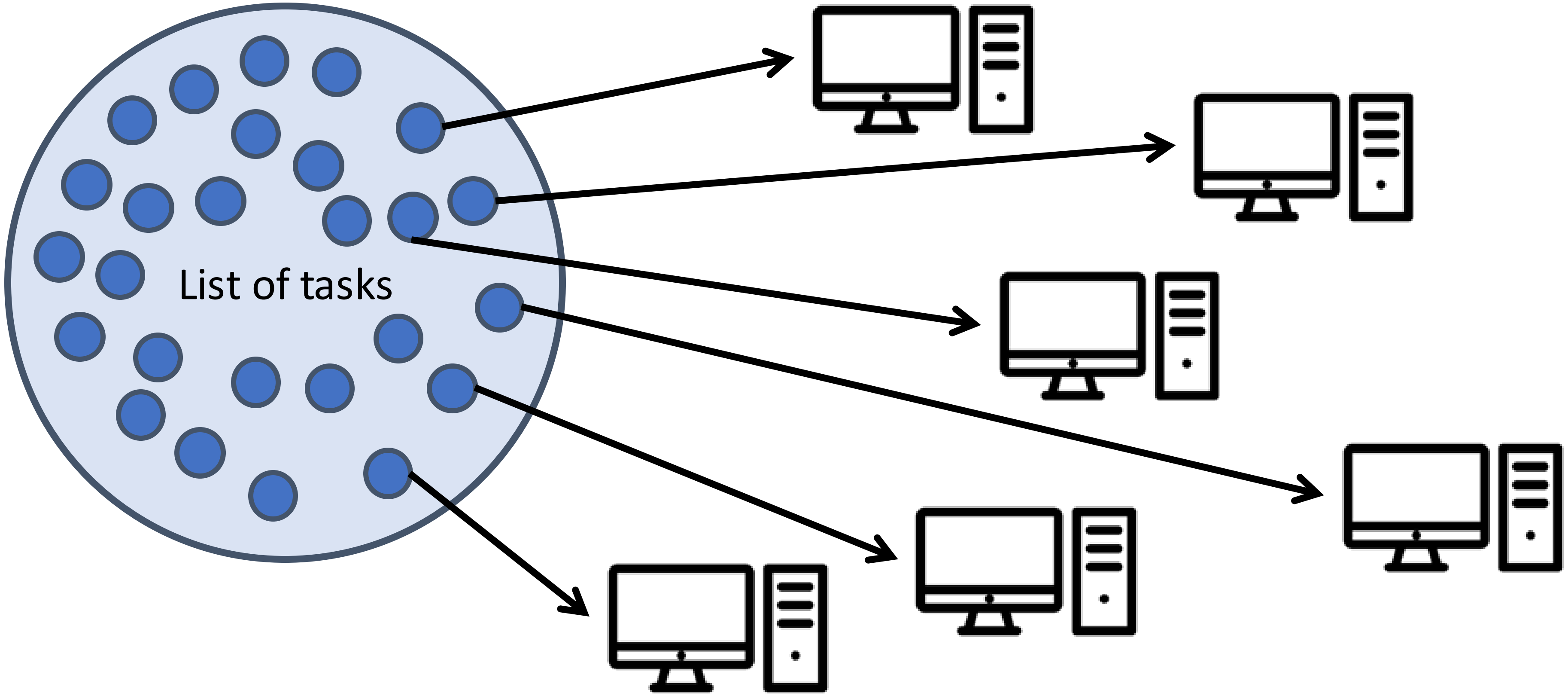
Execution Point
Computer available to run jobs

Resources in an EP
cores/CPUs: processing unit of computer
memory/RAM: space to temporarily store information (working memory)
disk: space for persistent files
GPUs: specialized processing units

Scheduler
A program that can assign Jobs to Execution Points

Job
Formal description of the task you want to run (usually input, executable, output)

High Throughput Computing (HTC)



What you need to do HTC

- A “home” to organize and start the computation
- Access to more computing capacity (an HTC system)
- Tools to manage and run our list of tasks (the scheduler)
- Components needed to run our tasks: software, scripts, data

This is exactly what we’re going to cover this week.

A “Home” for HTC Workflows

Access Point account

❑ ap40.uw.osg-htc.org

❑ ap2003.chtc.wisc.edu (later in the week)

- OSG Online Guides
  - Main Page > Get Started on the OSPool > Account Setup
  - Account and guide portal: https://portal.osg-htc.org/

Access to HTC Systems

Open Science Pool – Mon – Fri

CHTC (local campus pool) - Thursday

- OSG School materials
  - OSPool Introduction (Tuesday)
- OSG Online Guides
  - Main Page > Get Started on the OSPool > Welcome
  - Account and guide portal: https://portal.osg-htc.org/

Managing and Running Jobs

HTCondor (for most cases) and **DAGMan** (for workflows)

- OSG School materials
  - HTCondor Introduction (Mon)
  - Troubleshooting (Tues)
  - Workflows with DAGMan (Thurs)
- OSG Online Guides
  - Main Page > Submit HTC Workloads > HTC Workload Planning, Testing and Scaling Up
  - Main Page > Submit HTC Workloads > Monitor, Review and Troubleshoot Jobs

Job Components: Software, Data, Scripts

- OSG School materials:
  - Software (Tuesday)
  - Data (Wednesday)
- OSG Online Guides
  - Main Page > Submit HTC Workloads > Managing Data for Jobs
  - Main Page > Submit HTC Workloads > Using Software

Getting Started

- We’re here to help you do the following:
  - Think about your work as a list of jobs
  - Get it running on the OSPool
- Lots of resources available:
  - OSG School materials: slides, exercises
  - OSG guides and training materials
  - Other technical lessons (unix, git, naming things, docker…let’s crowd-source other materials as needed)

Connecting with Each Other

- Connect with people who are doing similar work to you!
  - Today: lunch with people in similar domains
  - Tomorrow: lunch with people using similar tools
- Do you want to share your contact info? Or a cool resource you know about for computing? Share here:
  - **https://go.wisc.edu/hs4t52**
- Staff are also a resource! Talk to us and sign up for consultations: **https://go.wisc.edu/8hf4ly**

Acknowledgements

This work was supported by the National Science Foundation under Cooperative Agreement OAC-2030508 – Partnership to Advance Throughput Computing (PATh)