UNIVERSITY OF CAMBRIDGE

Institute of Astronomy

# Intro to High Performance Computing

Gurjeet Jagwani

# Code and Slides: github.com/gurjeetjagwani/ioa hpc

# Outline

- What is HPC? Why should I care about using HPC?
- Scaling and effective use of HPC systems.
- Cambridge Facilities
- Accessing CSD3
- Resources and Modules
- Job Scheduler, Submitting and Monitoring Jobs
- Copying Data
- Questions

# What is High Performance Computing?

- Supercomputers or Clusters
- High Speed with many cores (CPUs or GPUs)
- High Memory (RAM)
- Share Memory Jobs (single node)
- Massively Parallel Jobs (many nodes)
- Controlled using SLURM (PBS, LFS, Kubernetes)
- https://top500.org/


Fastest Supercomputer in the world:
https://www.olcf.ornl.gov/frontier/

# Why care about using HPC?

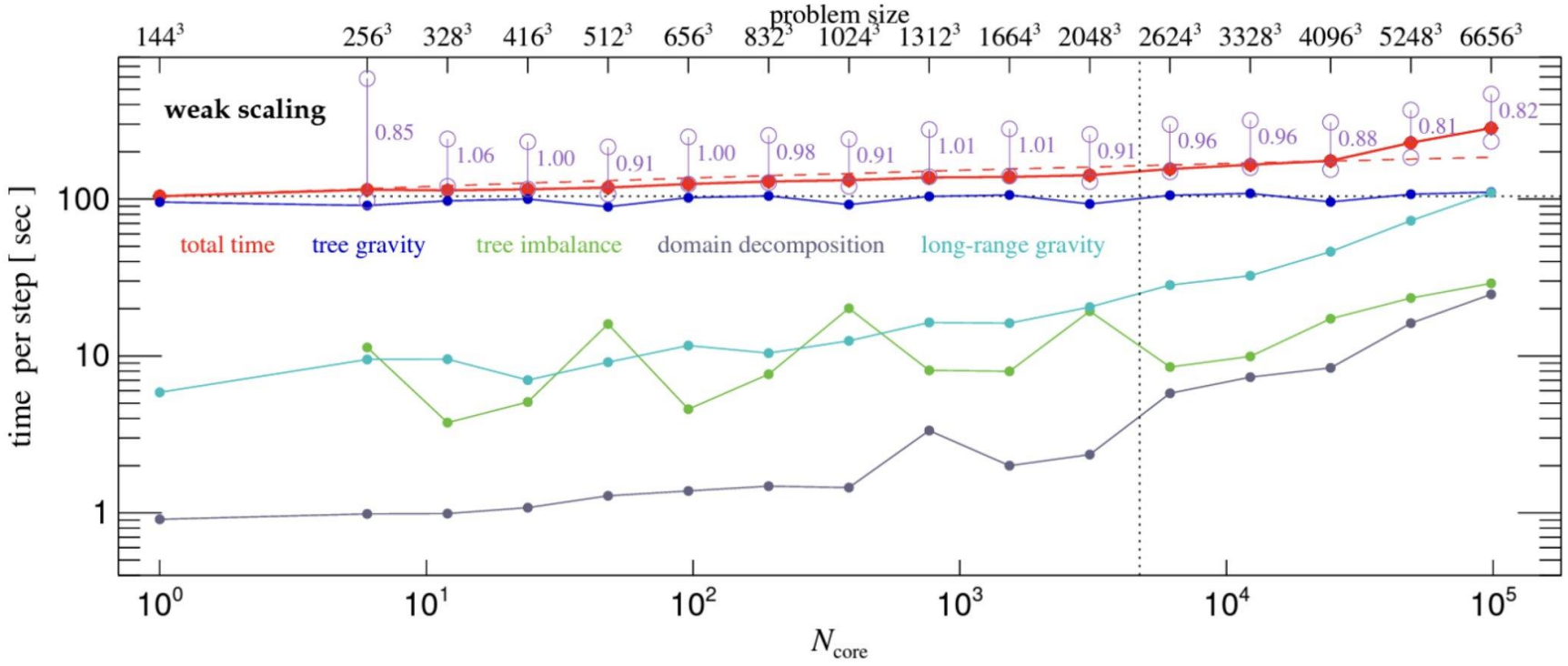
- Astronomical datasets are only getting larger!
- Astronomical simulations are only getting more complex!
- Solve complex and data intensive problems quickly
- Machine Learning workflows (Training and Inference)

SKA Estimated Data Rates :
20 PBs/observation

Vera C Rubin Data Rates:
15 TBs/night

# Weak Scaling

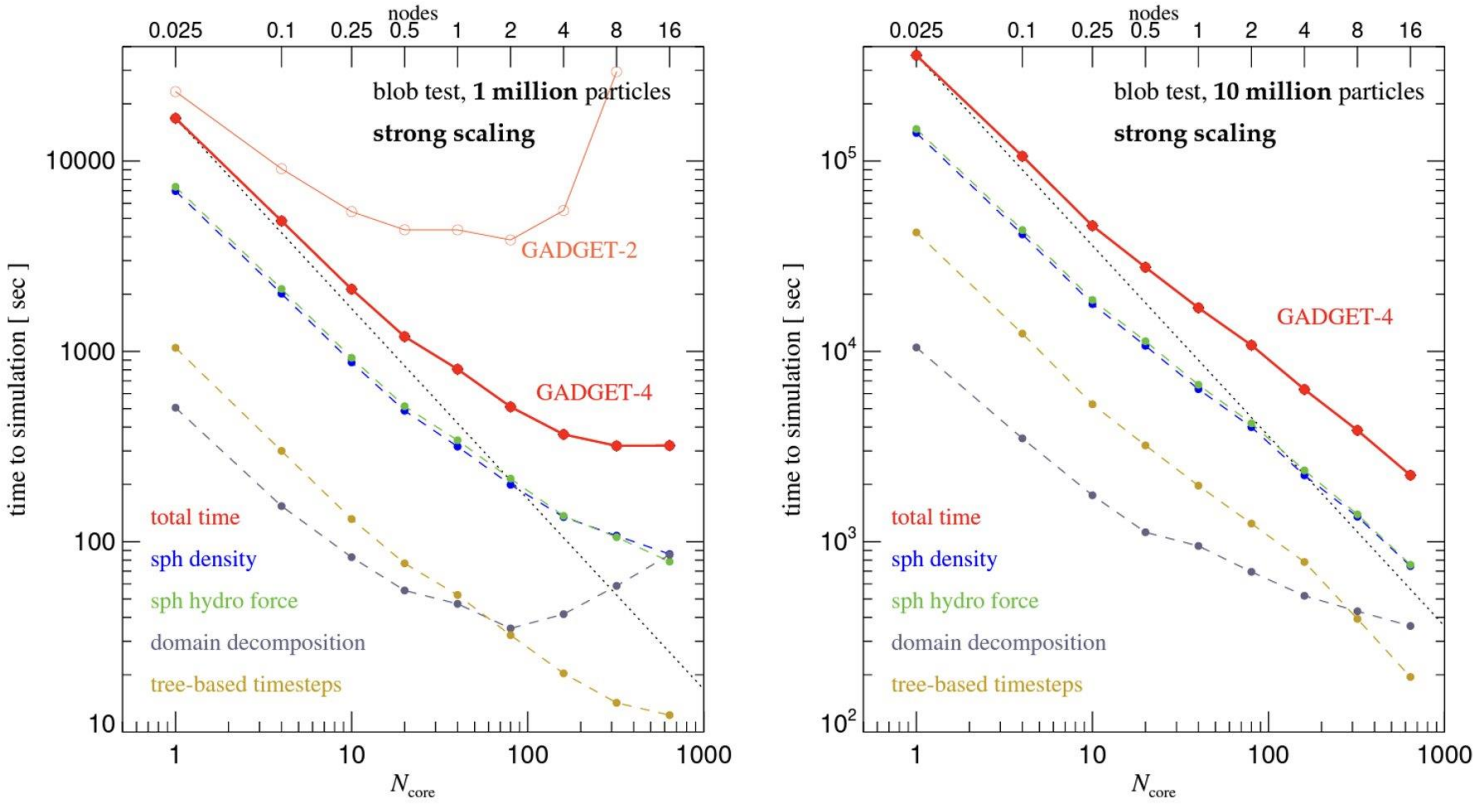
The problem size increases at the same rate as the number of processors, keeping the amount of work per processor the same.




Springel et al. 2021, Gadget 4(Cosmic Structure Simulation)

# Strong Scaling

Total problem size stays the same as the number of processors increases.




Springel et al. 2021, Gadget 4(Cosmic Structure Simulation)

# Effective use of HPC systems

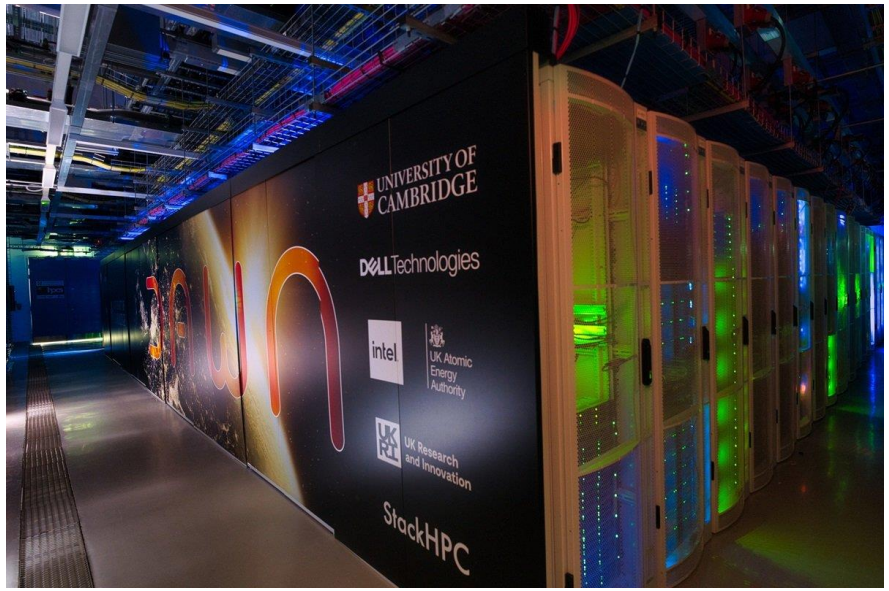
- Check your problem:
  - What is the compute and data requirements of your problem?
  - Are you using the most effective algorithms and data structure to solve your problem?
    - Algorithms and Data Structure: https://www.cs.csubak.edu/~hjafri/error.txt/Algorithms.pdf
    - RSEs can help!
  - Does your problem lend itself to parallelism?:
    - Yes (Monte Carlo Simulations, Auto-Differentiation (Machine Learning), Particle Simulations)
    - No (Markov Chains, Heirarchical Bayesian Models)

# Cambridge Facilities

CSD3 – Cambridge Service for Data Driven Discovery (https://www.hpc.cam.ac.uk/high-performance-computing)

- Wilkes3(GPU):
  - 80x Dell PowerEdge XE8545 servers, each with
  - 2x 3rd Generation AMD EPYC 64-Core CPUs
  - 1 TB memory
  - 4x NVIDIA A100-SXM-80GB GPUs
  - 2x HDR 200 Gbit InfiniBand network

- Cumulus(CPU):
  - Skylake – 18000 CPUs (~6 GB memory per CPU)
  - Cascade Lake – 37000 CPUs (~3.5 GB memory per CPU)
  - Ice Lake – 42000 CPUs (~3.5 GB memory per CPU)
  - Sapphire Rapids –12544 CPUs (~4.5 GB memory per CPU)




https://www.hpc.cam.ac.uk/dawn

# Accesing the system

- SSH: ssh <username> @login-cpu.hpc.cam.ac.uk
- Username : CRSid
- Web: https://login-web.hpc.cam.ac.uk

# Login node

- Which node am I on? - `hostname`
- Who else is using it? - `w`
- What is running? - `htop`
- Tell me about the processors – `lscpu`

# Resources available

- What is in the home directory? - **`ls`**
- How much disk space do I have/have I used? - **`quota`**
- Home directory: 40GB, backed up hourly
- Work directory: 1TB, fast, no back up, used temporarily for job output
- Where is my work directory? - **`cd rds/; ls -la`**
- What compute resources do I have? - **`mybalance`**

# Module System

- Modules – control the dev environment, allow you to load and unload specific compilers and libraries/packages
- What modules are available? - **`module avail`**
- What modules are loaded? - **`module list`**
- How do I load a module? - **`module load <module name>`**
- How do I swap a module? - **`module swap <loaded module> <replacement module>`**
- How do I unload a module? - **`module remove <module name>`**
- How do I unload all modules? - **`module purge`**
- Tell me about a module - **`module help <module name> or module whatis <module name>`**

# Job types and submitting jobs

- Batch job - job is run on some resources with no further input from user via a submission script or command line instruction
- Array job - runs multiple similar batch jobs
- Interactive - opens a command line job on resources requested (see https://docs.hpc.cam.ac.uk/hpc/user-guide/interactive.html)
- Sample submission scripts are located in your home directory as: slurm_submit.<machine>-<node-type>

# Task one - a simple batch job

- Go to the task1 directory: **`cd example_resources/task1/`**
- Look at example.cpp: **`emacs example.cpp`**
- Compile the sample code: **`g++ -std=c++11 example.cpp -o example`**
- Open the submission script: **`emacs submit_batch`**
- Change the account name (-A) option to your own (find by calling **`mybalance`** on command line)
- Submit the job: **`sbatch submit_batch`**

# How to check job progress and outcome

- Check current status: **`squeue -u <User ID>`**
- Check completed job status: **`scontrol show job=<Job ID>`**
- Cancel a job: **`scancel <Job ID>`**
- Check jobs I've run previously: **`gbalance -u <User ID>`**
- NB: can restrict to a particular project (**`-p`**) and start/end time (**`-s/-e`**)

# Task one – continued

- Once the job has completed there should (hopefully) be some new files in the directory, let's take a look: **`ls`**
- **machine.file.<Job ID>** - lists the node hostnames the job ran on
- **slurm-<Job ID>.out** - standard output is stored in this file
- **example.txt** - was produced by our example code - the text in here should match the machine file
- Now rerun this but update the submission script to use 2 nodes, how do the output files differ?

# Task two - an array job

- Go to the task2 directory: **`cd ../task2/`**
- Compile the sample code: **`g++ -std=c++11 example.cpp -o example`**
- Open the submission script: **`emacs submit_array`**
- Change the account name (**`-A`**) option to your own (find by calling **`mybalance`** on command line)
- Submit the job: **`sbatch submit_array`**

# Task Two- Continued

- Once the job has completed there should (hopefully) be some new files (and directories), let's take a look: **`ls`**
- As before there are slurm files, but now there is one for each job.
- The other files are now placed in directories named **Job_<array ID>**
- The **example.txt** files should show different options

# Task three - let's try a bigger code base

- Next we will try running Arepo
- Lets try this on ***Icelake*** nodes instead
- Log out of current CSD3 session and re-login using:
- **`ssh <username>@login-q-<1-4>.hpc.cam.ac.uk`**
- Move to the task 3 directory:
- **`cd ~/rds/hpc-work/example_resources/task3`**
- Get a copy of Arepo: **`git clone`** `https://gitlab.mpcdf.mpg.de/vrs/arepo.git`

# Task three – let's try a bigger code base

- Open the submission script: **`emacs submit_arepo`**
- Change the account name (-A) option to your own (find by calling **`mybalance`** on command line)
- Copy compilation, makefiles and submission script to the arepo directory:
- **`cp Makefile.systype Makefile submit_arepo compile.sh arepo/.`**
- **Makefile**: arepo makefile with system dependent setting
- **Makefile.systype**: contains the current system name
- **compile.sh**: a script that loads relevant modules and runs make
- **submit_arepo**: batch job submission script

# Task three - let's try a bigger code base

- We need to set up initial conditions for the run, we will be doing a 1D shock tube test. We will need to set up a python environment to do this:

- Change to home directory: **`cd ~/`**
- Load python module: **`module load python/3.6`**
- Setup a virtual environment:
- **`virtualenv --system-site-packages MY_PYTHON`**
- Activate it: **`source MY_PYTHON/bin/activate`**
- And install h5py: **`pip install h5py`**

# Task three – let's try a bigger code base

- Now move to the shock tube example folder in the arepo directory:
- **`cd rds/hpc-work/example_resources/task3/arepo/examples/shocktube_1d/`**
- Create the ICs: **`python create.py ../../`**
- Copy config and parameter files to main arepo directory:
- **`cp Config.sh param.txt ../../.`**
- Move back to main arepo directory: **`cd ../../`**
- Compile the code: **`sh compile.sh`**
- Submit the job: **`sbatch submit_arepo`**
- Once it has run, what output is produced?

# Copying Data

- There are a few ways to copy files to/from the HPC system:
- **Secure Copy Protocol (scp)**: securely copy files from one location to another
- **Rsync**: faster than scp and uses delta transfer algorithm - only copies files that are new or have changed

# SCP

- Let's try some examples
- Open a new terminal and log into an IoA server, e.g. calx042
- Make and enter a new folder in your home directory:
- **`mkdir HPCIntroFiles && cd "$_"`**
- Copy snap_000.hdf5 from the arepo run: **`scp <username>@login-cpu.hpc.cam.ac.uk:/home/<username>/rds/hpc-work/example_resources/task3/arepo/output/snap_000.hdf5 .`**
- Or to keep date stamp information: **`scp -p <username>@login-cpu.hpc.cam.ac.uk:/home/<username>/rds/hpc-work/example_resources/task3/arepo/output/snap_000.hdf5 .`**

# rsync

- Empty the local current folder: **rm snap_***
- Copy all snapshots over using rsync:
- **rsync -ravzPS -e ssh <username>@login-cpu.hpc.cam.ac.uk:/home/<username>/rds/hpc-work/example_resources/task3/arepo/output/snap_*** .
- Now copy the whole output folder content:
- **rsync -ravzPS -e ssh <username>@login-cpu.hpc.cam.ac.uk:/home/<username>/rds/hpc-work/example_resources/task3/arepo/output/*** .
- Notice the snap files are not copied again as they already exist locally

# More info

- https://docs.hpc.cam.ac.uk/hpc/