THE UNIVERSITY OF CHICAGO | Research Computing Center

Overview of HPC Resources & Use of the Research Computing Center Cluster

Jonathan Skone

Organization of RCC Material

Provide information on HPC resources and services available to users

- Overview of RCC
- HPC Terminology and landscape
- UofC's centralized HPC cluster -- Midway2

Demonstration on accessing and Navigating Midway

- Connecting--client setup
- Navigating the shell
- Transferring data to/from the cluster
- Using software modules

Using the Scheduler and other Computing Tips

- Submitting Jobs to resource scheduler
- Monitoring/inspecting jobs
- Efficient Job Use (Job arrays, inspection of past use & note on priority)

RCC: the What and Where

The What:

The RCC is a unit under the Office of the Executive Vice President for Research, Innovation and National Labs

Data Center
@ 6045 Kenwood

The Where:

Data Visualization Lab

Zar Room

Walk-in

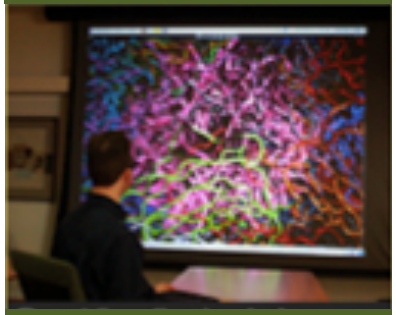
Regenstein room 216

A Go-To Place

Central office Located at:

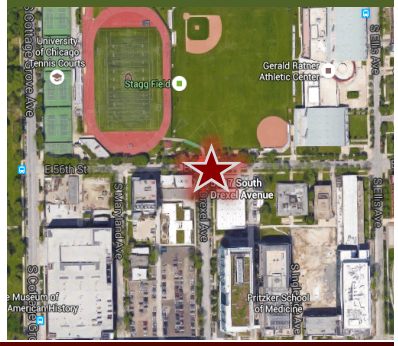
5607 S Drexel Avenue

Crear Library
Zar Room

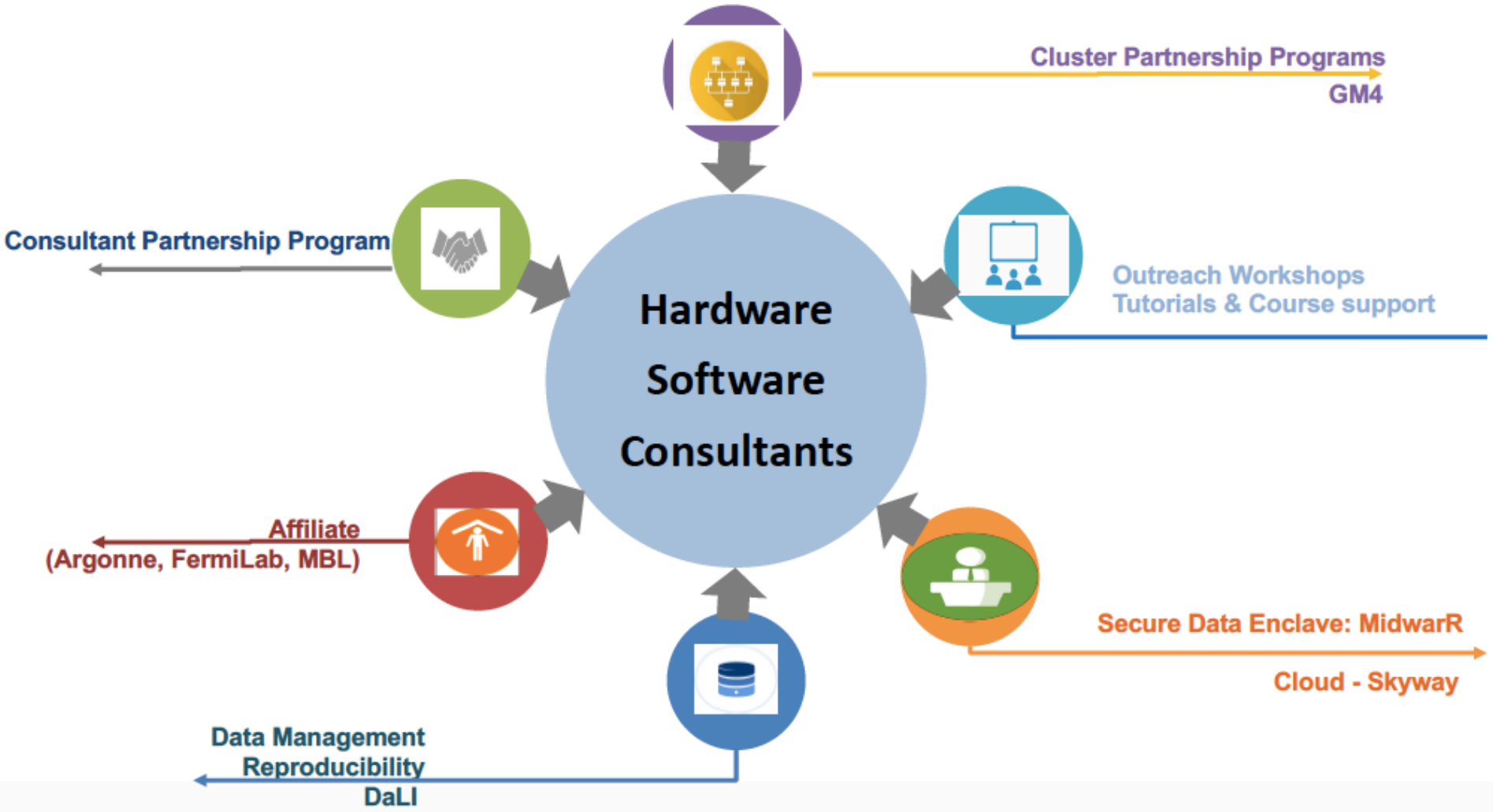


Regenstein 216

TACC



RCC core resources, programs, and services



Computational Scientists Group

Dora Tszasz
Image Analysis & Visualization

Brooke Luetgert
Data Analysis & Visualization

Jonathan Skone
Computational Chemistry & Materials Science

Ping-Chang Lin
Machine Learning & Data Analysis

Jeffrey Tharsen
Digital Humanities & Philology

Peter Carbonetto
Bioninformatics

Yuxing Peng
Computational Chemistry

Parmanand Sinha
Spatial Science & GIS

Application Web Development

Application web development is another service RCC has initiated supporting for the University of Chicago Community. If interested in this service contact help@rcc.uchicago

Examples of data portal projects:

SAGA
saga.rcc.uchicago.edu

DMREF Datahub
datahub.rcc.uchicago.edu/dmref

Diversity mapping
diversity.rcc.uchicago.edu

RDCEP US Energy Visualization
us-sankey.rcc.uchicago.edu/

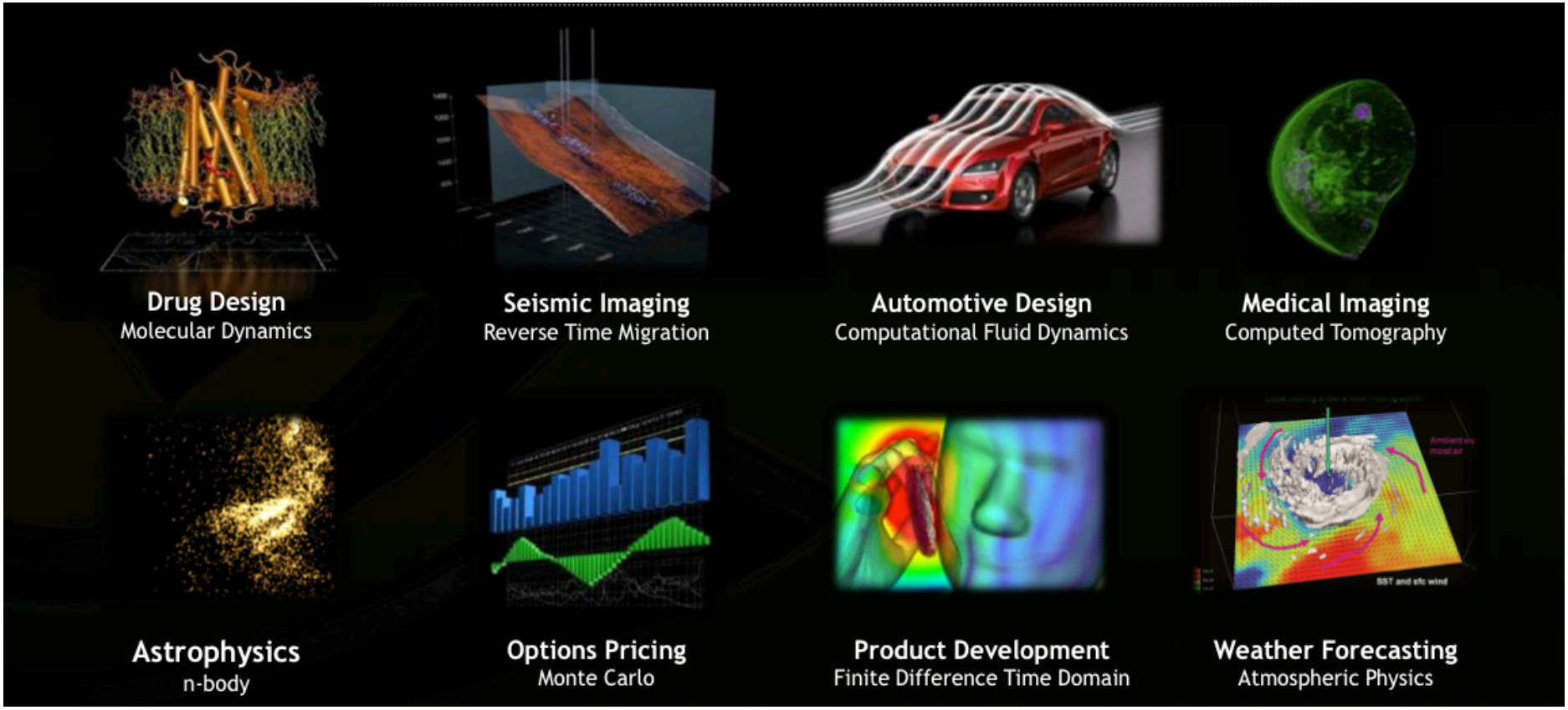
Milson Munakami
Scientific application developer

Mengxing Peng
Systems Engineer

Kalyan Reddivari
Lead Application Developer

Prathprathyusha Merla
Application Software Dev.

Computing and Big Problems



- Computing-based science discovery deals with Big problems that can be compute intense, memory intense, I/O intensive or all these.

- High Performance Computing (HPC can address these problems if the software can scale up or across resources.

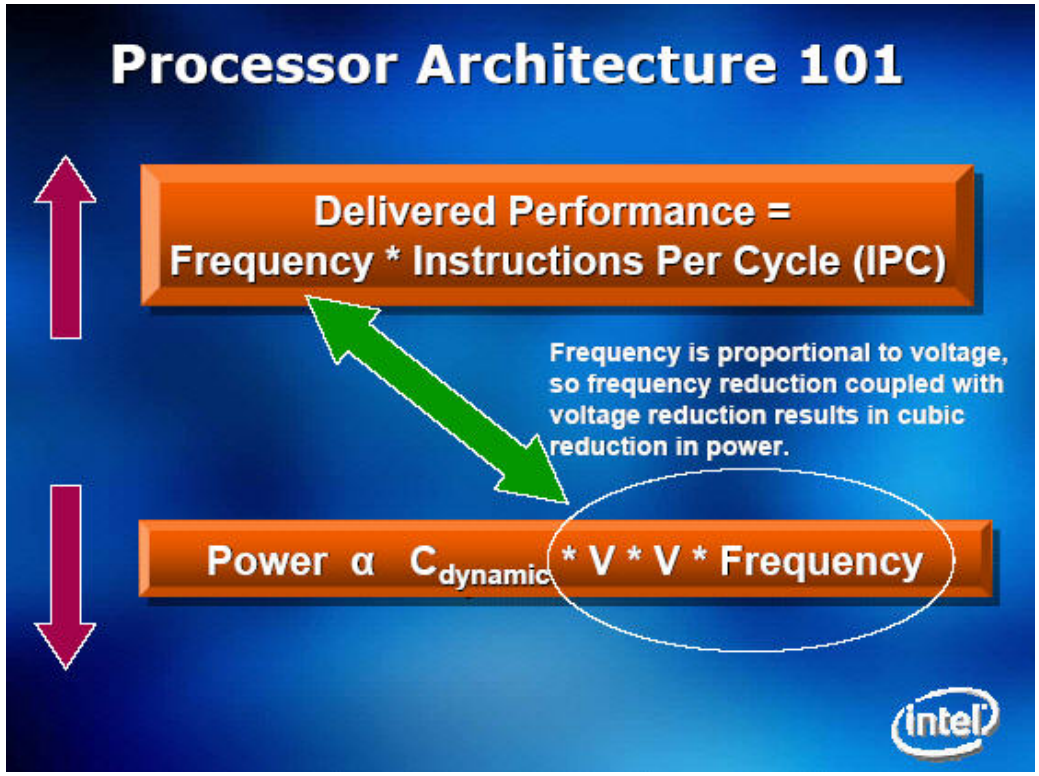
Types of Big problems that suit HPC

- **Compute Intensive**
  - A single problem requires a large amount of computation
- **Memory Intensive**
  - A single problem requires a large amount of memory
- **Data Intensive**
  - A single problem operates on a large amount of data
- **High Throughput**
  - Many copies of the same problem to be executed in parallel

Big Problems require Big Computers

Faster CPUs

- Processor clock speeds have flattened out
- The fastest commercially available chip is about 5.0 GHz
- Clock speed is limited by power consumption, heat dissipation, current leakage



Big Problems require Big Computers

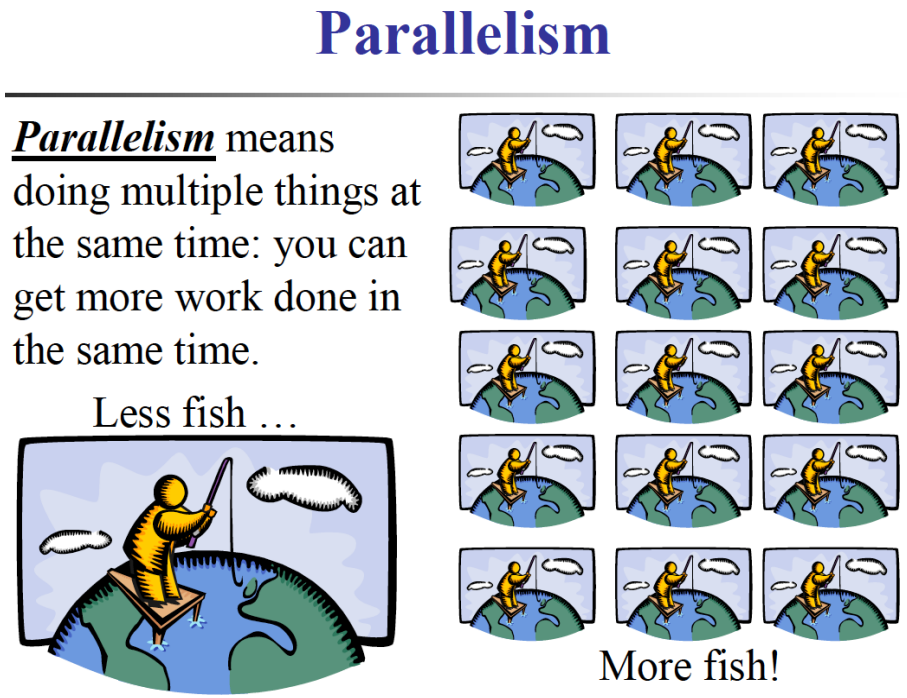
- Many CPUs
- Lots of memory
  - Limited by how much memory a CPU can support
- Large amount of storage space

Parallelism, the path toward future performance gains

  - Trend is toward multi-core and many core

What is HPC?

High Performance Computing is the “practice of aggregating computing power in a way that delivers much higher performance than one could get out of a typical desktop/laptop computer”[1]



[1]http://insidehpc.com/hpc-basic-training/what-is-hpc/
Picture from “supercomputing in plain English”, http://www.oscer.ou.edu/education.php

Parallel Computer Architectures

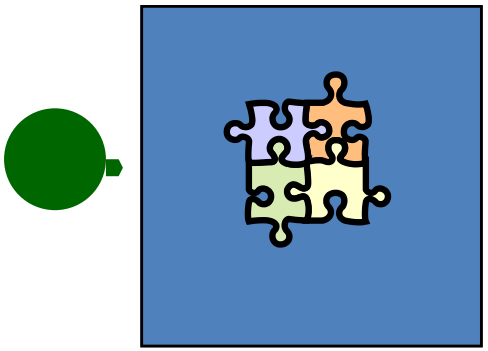
The Jigsaw Puzzle

- Say you have a jigsaw puzzle with 1000 pieces
- How can you put it together as fast as possible?

Adapted from “supercomputing in plain English”, http://www.oscer.ou.edu/education.php

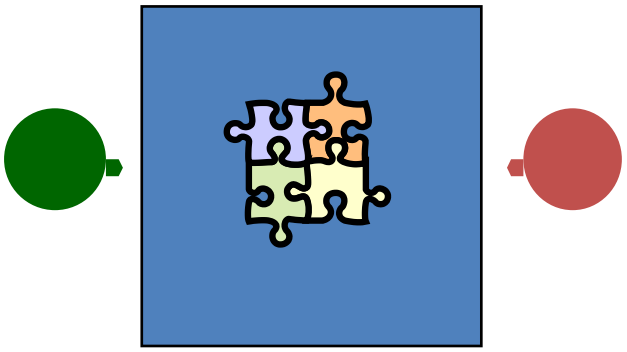
The Jigsaw Puzzle

- Serial Computing
  - You sit down at the table by yourself and put together all 1000 pieces, one after the next
  - It takes you 1 hour to assemble the puzzle
  - Well done!
  - But can we do better?



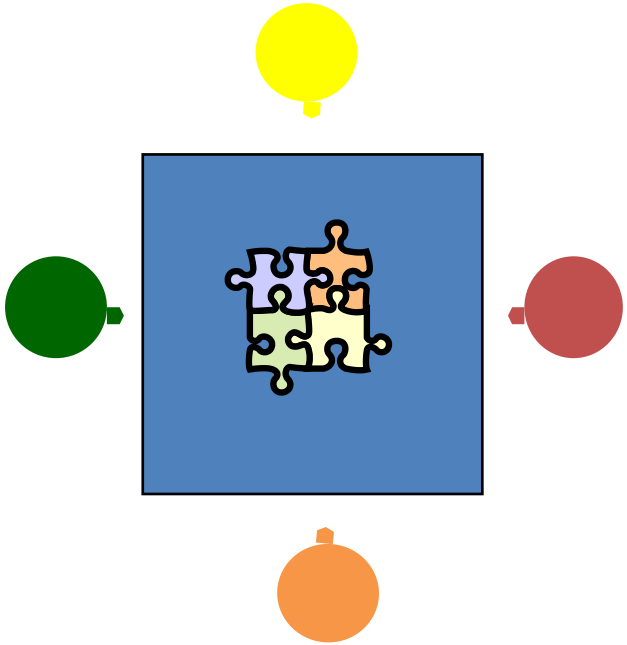
The Jigsaw Puzzle

- Shared Memory Parallelism
  - If your friend sits across from you, then he can work on his half of the puzzle and you can work on yours
  - Once in a while, you'll both reach into the pile for the same piece (you will **contend** for the same resource) which causes a little slowdown
  - From time to time, you'll have to work together (**communicate**) at the interface of your halves
  - If all goes well, you'll get a 2x speedup
  - But we have communication and contention overhead so it will probably take more like 35 minutes instead of 1 hour



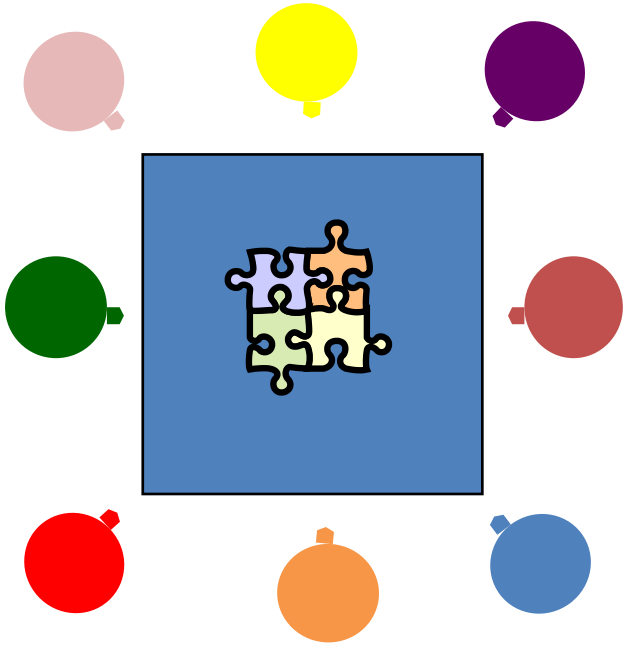
The Jigsaw Puzzle

- More Shared Memory Parallelism?
  - Let's say you add two more friends
  - Each person works on their quadrant of the puzzle, we should get a 4x speedup, right?
  - But, there will be a lot more contention for pieces and a lot more communication
  - Instead of putting together puzzle in ¼ hours or 15 minutes, you'll probably be closer to 20 minutes



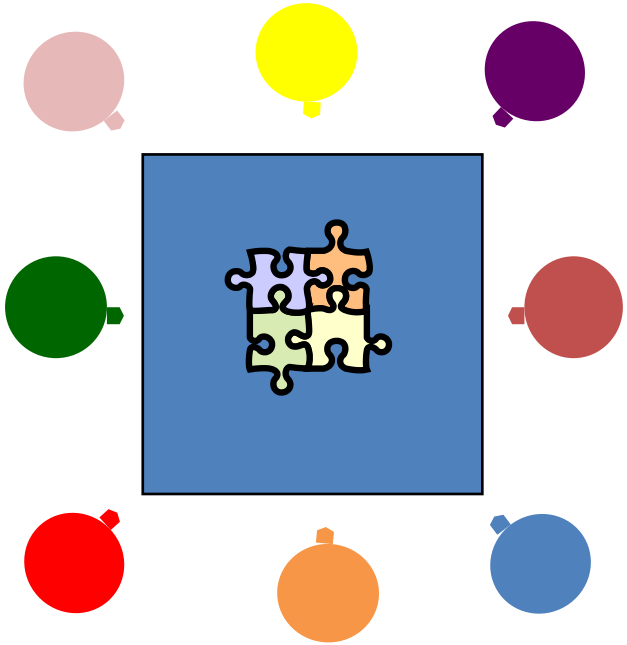
The Jigsaw Puzzle

- Diminishing returns?
  - Let's say you add 4 more friends
  - MUCH more contention for pieces and a TON more communication
  - If you actually manage to get a 8x speedup it would be very impressive



The Jigsaw Puzzle

- But...
  - Let's assume the 8 of you are extremely good at working together
  - Maybe you can get an almost 8x speedup
  - But we want to go faster yet
  - Problem! There's only enough space for 8 people at the table!

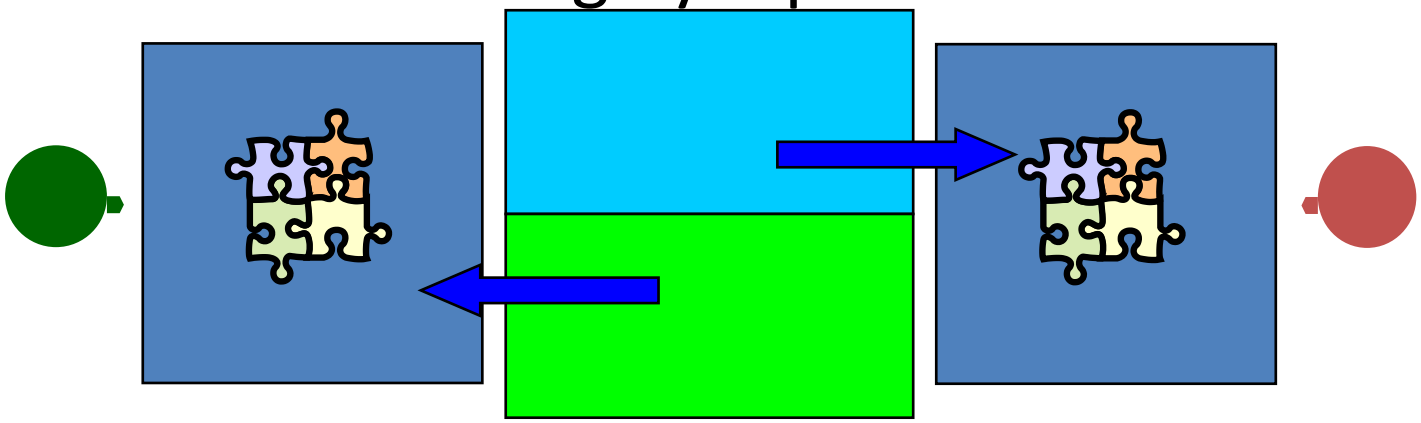


The Jigsaw Puzzle

- Distributed Parallelism
  - You sit at table 1 and your friend sits at table 2
  - PRO: Plenty of elbow room
  - PRO: You can work without contention for resources
  - CON: Communication is much more difficult. You need to carry pieces from one table to the other to assemble them
  - Other problems?

The Jigsaw Puzzle

- Load balancing and Domain Decomposition
  - Say the puzzle is half blue sky and half green grass
  - Put all the blue pieces on one table and all the green pieces on the other
  - Pieces/table are roughly equal so each half should be assembled in roughly equal amount of time



A word about load balancing

- **Q:** How long does it take to finish 100 parallel calculations that all start at the same time?

A word about load balancing

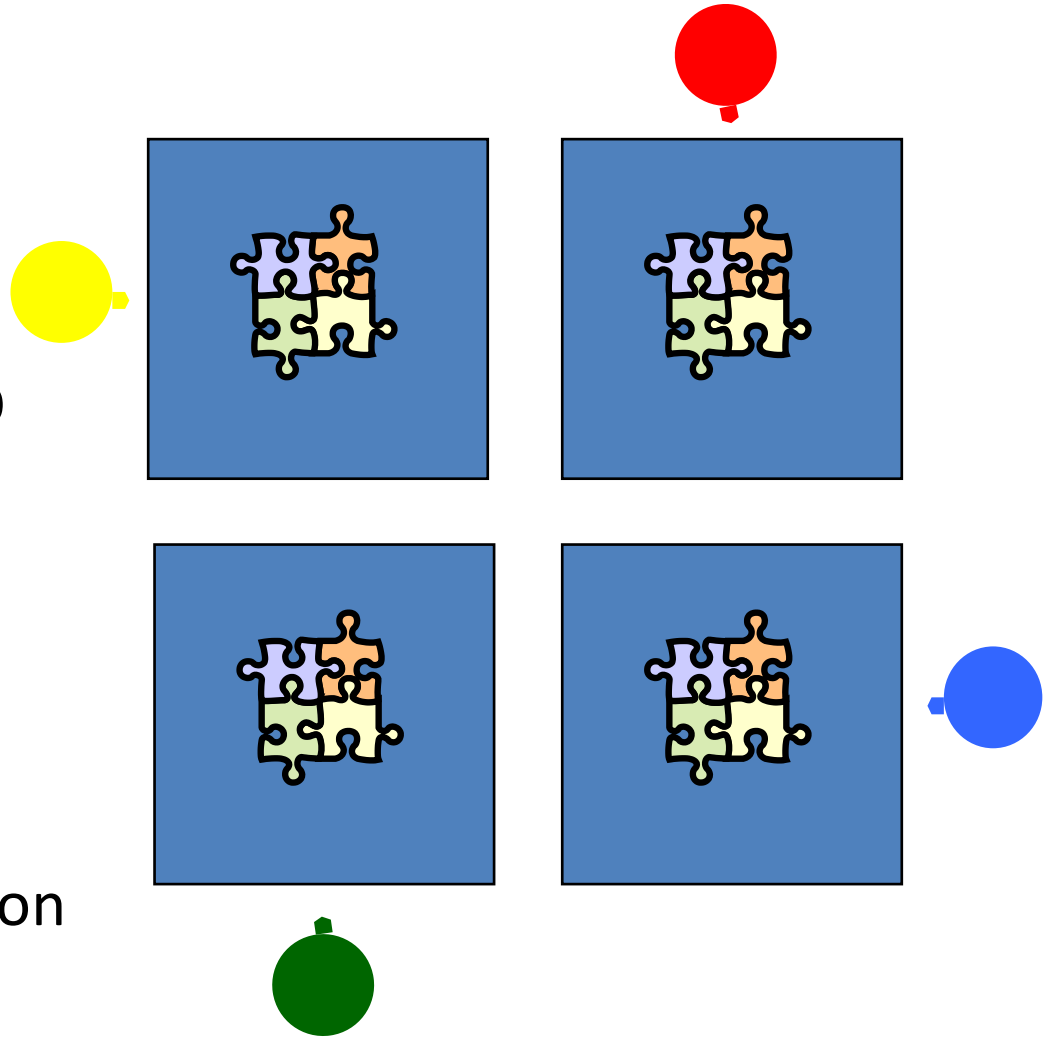
- **Q:** How long does it take to finish 100 parallel calculations that all start at the same time?

- **A:** However long the slowest of the 100 calculations takes

If one calculation is significantly slower than all the rest, parallelism will not help you

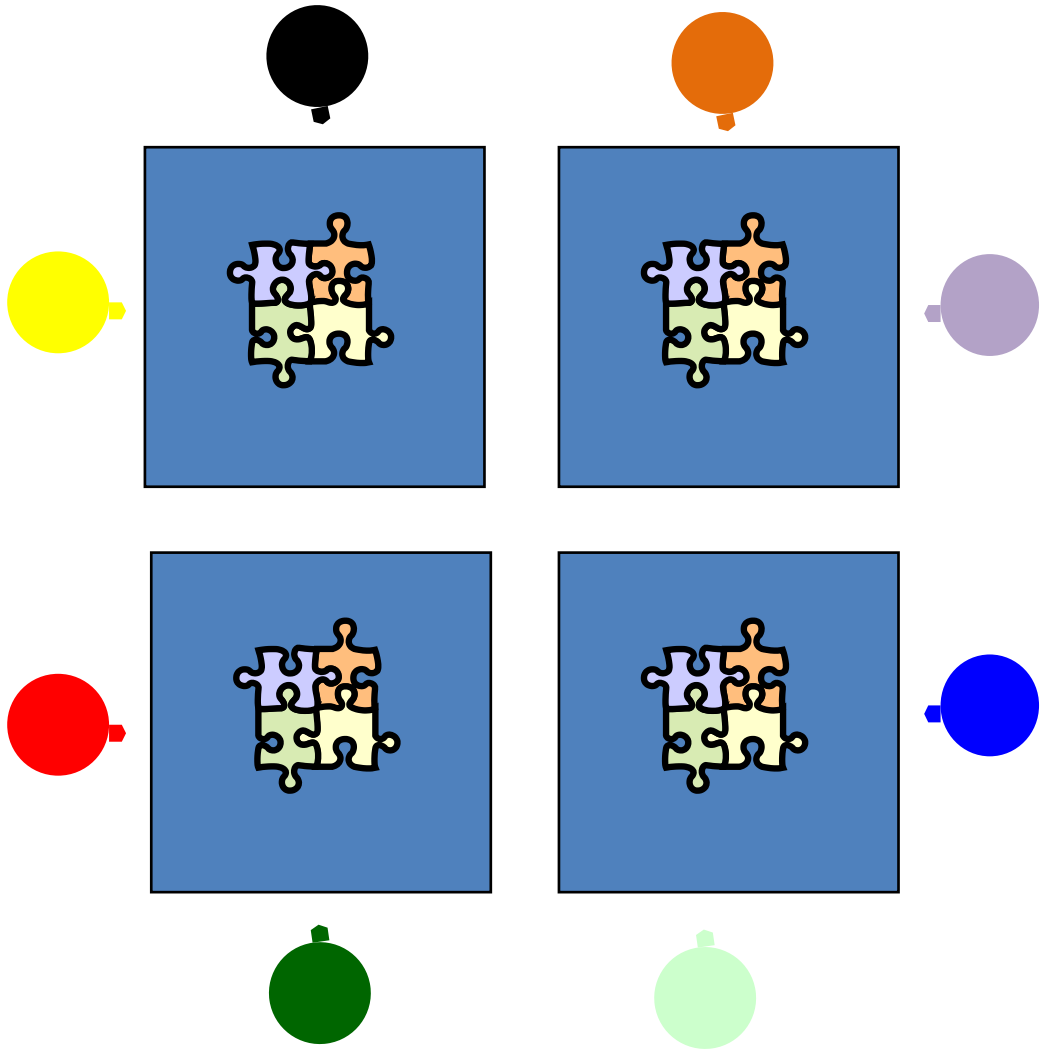
The Jigsaw Puzzle

- More Distributed Parallelism?
  - It's very easy to keep adding more tables
  - But…
    - Communication
    - Load balancing
    - Domain decomposition



The Jigsaw Puzzle

- Hybrid Parallelism
  - Combine distributed and shared models
  - What are the problems?
  - What about advantages?



Some definitions

- **Core**: smallest computation unit that can run a program (used to be called a processor, still is, also called a CPU — Central Processing Unit)

- **Socket**: a computational unit, packaged as one and usually made of a single chip often called processor. Modern sockets carry many cores (2, 4 on most laptops, 8 to 16 on most servers)

- **Node**: a stand-alone computer system that contains one or more sockets, memory, storage, etc. connected to other nodes via a fast network interconnect

Question

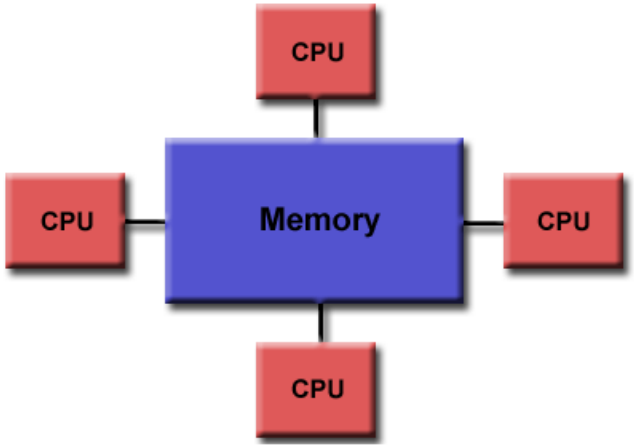
- Find out how many cores, sockets, how much main memory (aka RAM), and hard drive your laptop has.

A super computer is...

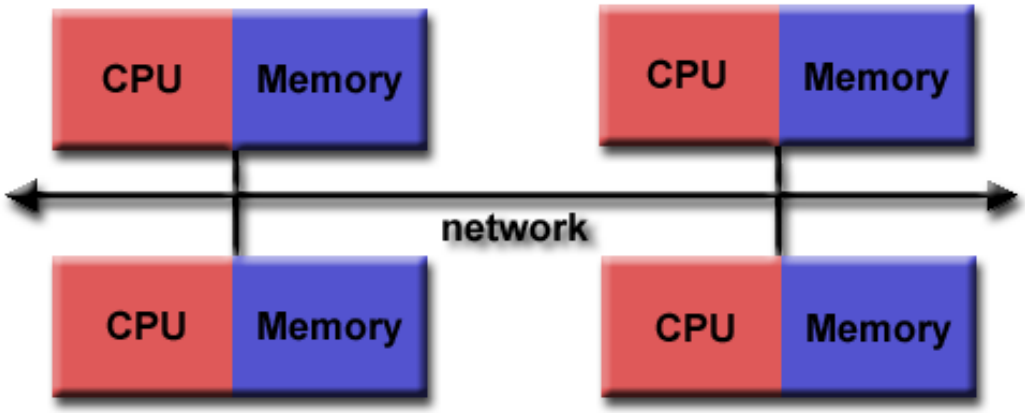
- A collection of small computers, called **nodes**, hooked together by an **interconnection network** (or interconnect for short)
- Software that allows nodes to communicate with one another
- All of these nodes work together as if they are one big computer … a supercomputer!

Parallel Computer Architectures

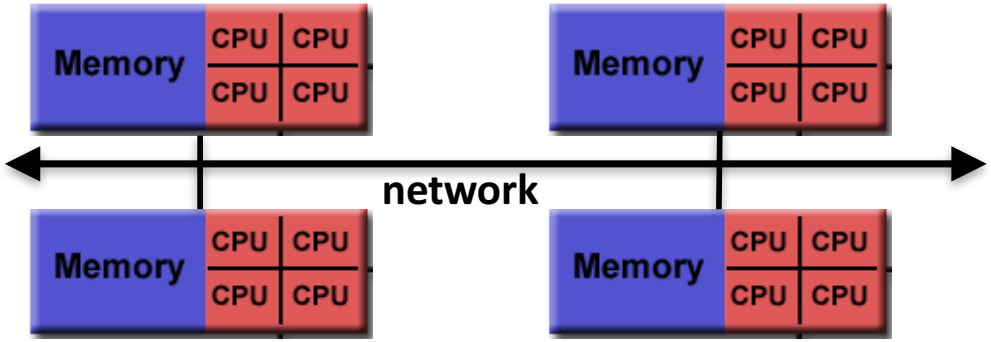
Shared Memory



Distributed Memory



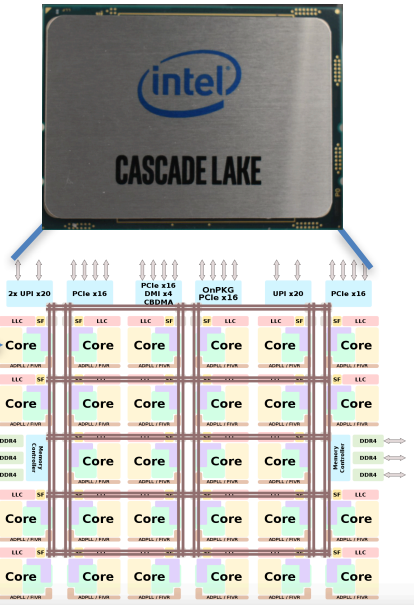
Hybrid



What Does HPC look like?

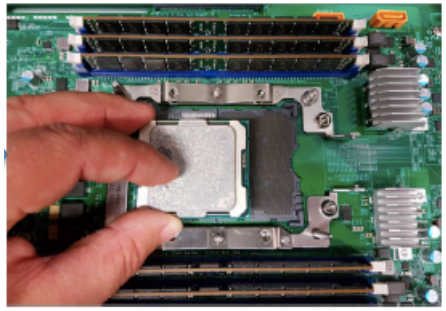
From CPU cores to Cluster

CPU

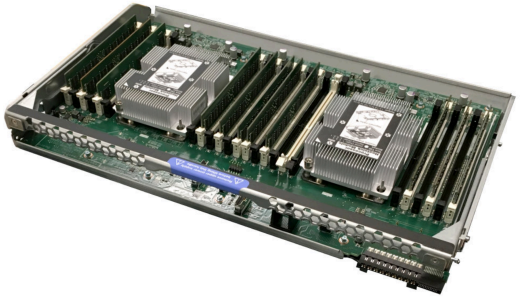


core

Socket



Motherboard



Node

Cabinet

(A collection of interconnected nodes)

Cluster

(A collection of interconnected cabinets)

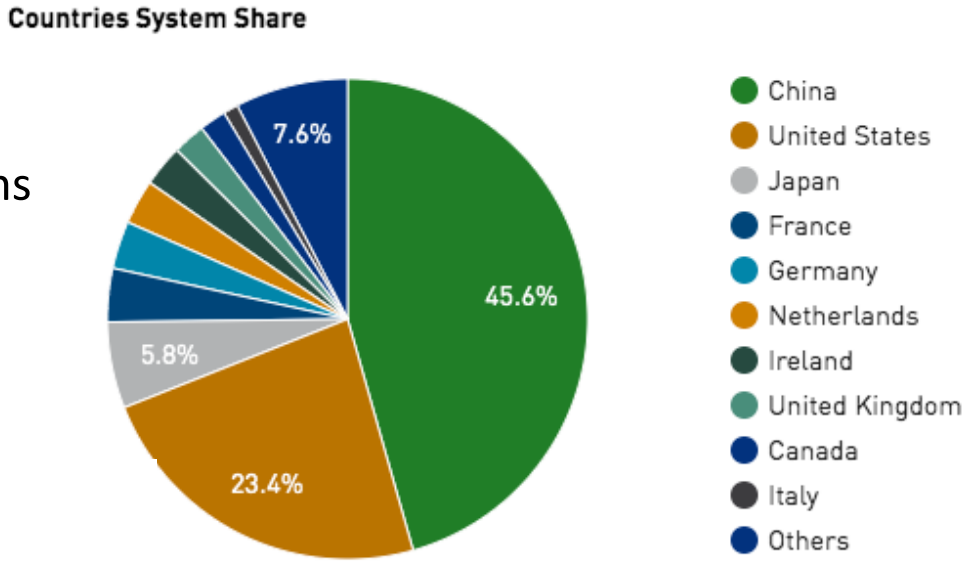
Fastest Supercomputers

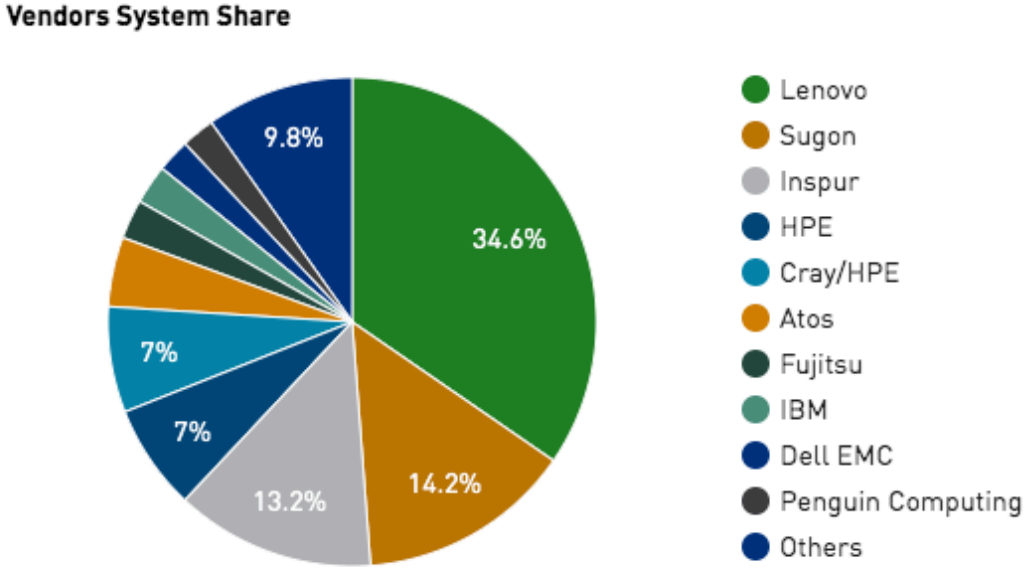
As of June 2020

Rank	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	**Supercomputer Fugaku** - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu RIKEN Center for Computational Science Japan	7,299,072	415,530.0	513,854.7	28,335
2	**Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM DOE/SC/Oak Ridge National Laboratory United States	2,414,592	148,600.0	200,794.9	10,096
3	**Sierra** - IBM Power System AC922, IBM POWER9 22C 3.1GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM / NVIDIA / Mellanox DOE/NNSA/LLNL United States	1,572,480	94,640.0	125,712.0	7,438

Fastest Supercomputers

China has the most supercomputing systems in the top 500 followed by the US.





The supercomputer vendor Lenovo has the largest market share of HPC systems followed by Sugon and Inspur

The RCC Compute Cluster

Computing hardware

Ranked 442nd in November 2016

Vendor: Lenovo

400 nodes total

- **342** tightly coupled Broadwell nodes (10,360 cores)
  Two Intel E5-2680v4 processors per node (14 cores)
  155 nodes have EDR network card
  187 nodes have FDR network card
- **6** NVidia Tesla K80 GPU nodes (4 GPU cards/node)
- **5** large shared memory nodes (512GB each)
- **14** dual socket loosely-coupled Broadwell nodes

Midway2 Compute Racks

Cluster Partnership Program: 1000+ nodes

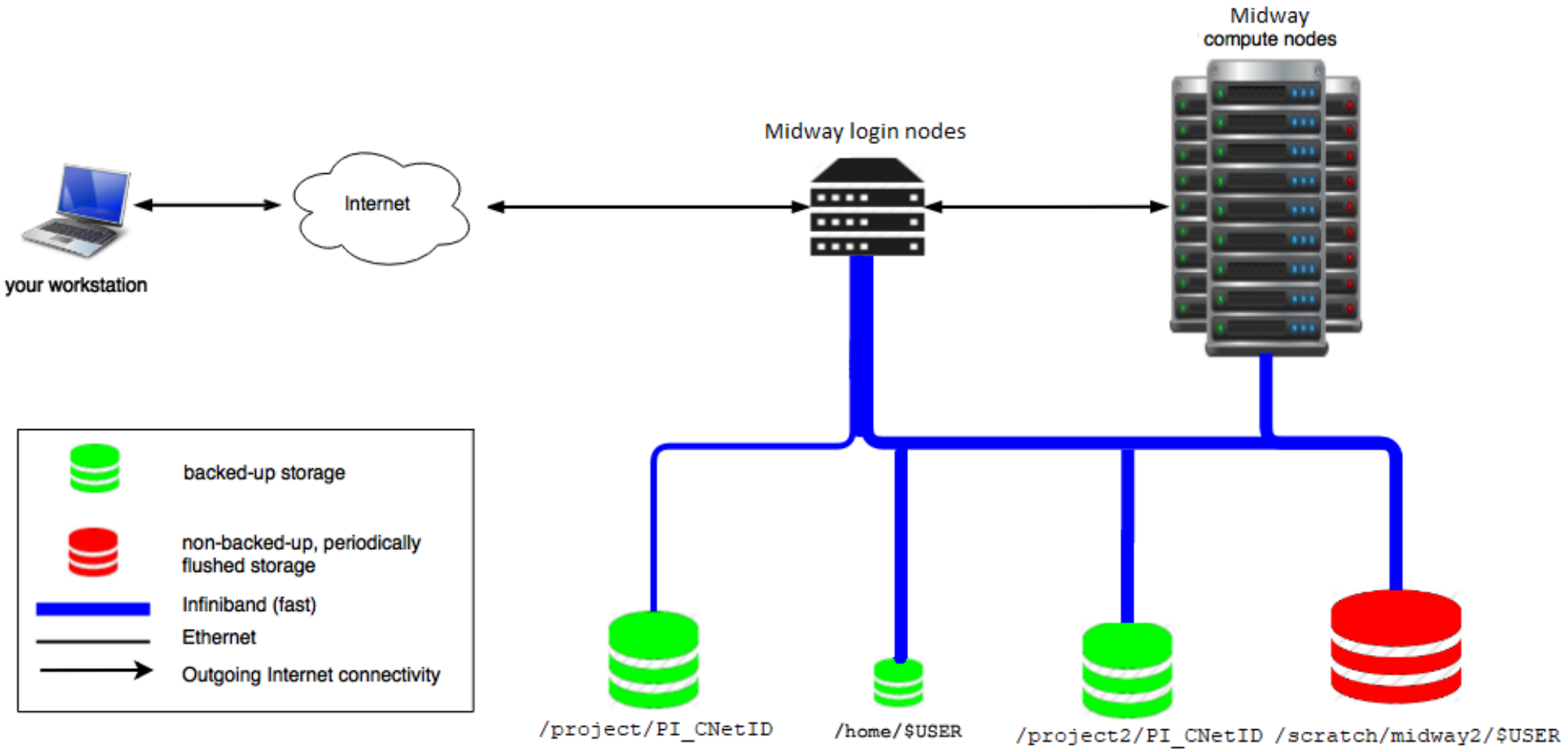
- **900+** tightly coupled infiniband nodes
- **20+** Big memory nodes
- **100+** Nvidia GPU nodes

Midway2 Shared Resources

Cores(Nodes)	CPU	Memory	Queue	Network Connection
5,236 (187)	2.4GHz Intel Broadwell	64GB	broadwl	FDR infiniband
4,340 (155)	2.4GHz Intel Broadwell	64GB	broadwl	EDR infiniband
140 (5)	2.4GHz Intel Broadwell	512GB	bigmem2	FDR infiniband
392 (14)	2.4GHz Intel Broadwell	64GB	broadwl-lc	no infiniband

Cores (Nodes)	CPU Accelerator	Memory	Queue	Network Connection
168 (6)	2.4GHz Intel Broadwell NVIDIA Tesla K80	128GB	gpu2	FDR infiniband

Schematic of Midway2 Cluster



Midway2 Storage

High Capacity storage: /project2

- **3.8 PB** of storage
- Backed up to tape system
- **7 daily and 4 weekly snapshots** located at /snapshots/project2
- 7 day grace period on over quota

High Performance storage: /scratch/midway2

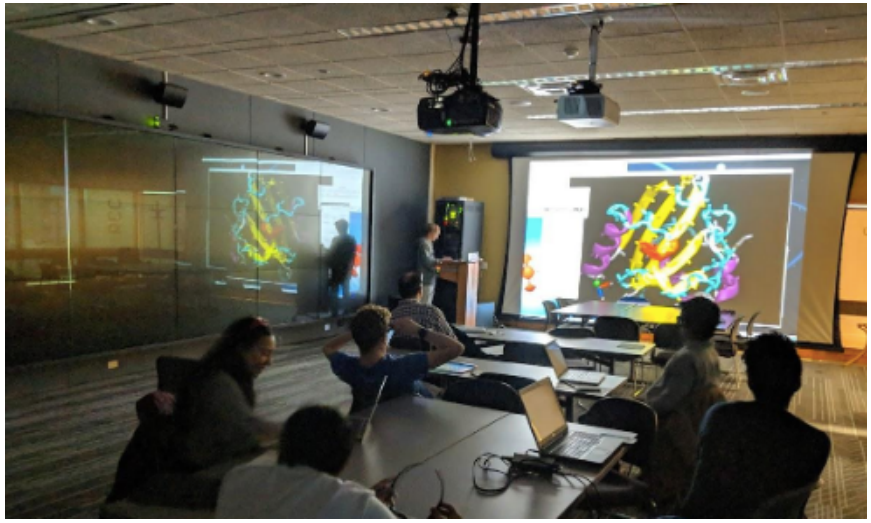
- **190 TB** usable
- Not backed up
- 100 GB user soft quota
- 30 day grace period on over quota

Home directory space: /home

- **61 TB** of capacity
- Each user has 30 GB quota
- 7 day grace period on over quota
- **7 daily and 2 weekly snapshots** located at /snapshots/home

Data Visualization Laboratory

- 3D MRI-CT Reconstruction
- HTC Vive Virtual Reality System
- High-resolution data visualization wall
- General scientific visualization
- Remote visualization and sharing SAGE2
- Visualization tools…Paraview, Amira, etc.



3DSlicer

RCC Allocation Account Types

- **Startup**: small allocations given to all PIs.
- **Research**:
  - Type I: Allocation for small research projects. Reviewed by RCC and provided upon request.
  - Type II: Larger research projects with significant resource requirements. Reviewed bi-annually by allocation committee.
- **Education:** allocation to support courses or workshops for a number of users over a limited period.
- **Special:** Timely or "heroic" research programs that cannot be satisfied by the standard research allocation.

Allocation Policy

- Creation of PI and user accounts provided throughout the year
- All Principal Investigators (PIs) receive **a startup allocation of 5,000** Service Units (SUs) upn PI account creation.
- PIs can request a **Research I allocation** at any time during the year. This **provides 50K SUs to non-CPP partners or 100K SUs for CPP partners**.
- Larger allocations are done through a Research Allocation II request. This allocation request require a 3+ page proposal that is reviewed by a committee. The Research II allocation is accepted in the spring and fall and allows a PI to request up to 1 million SUs for non-CPP partners or 2 million SUs for CPP partners.
- Allocation cycle runs from October 1st to September 30th of the following year.
- **Any unused allocation at the end of the allocation cycle is NOT rolled over to the next cycle.**

What is a Service Unit?

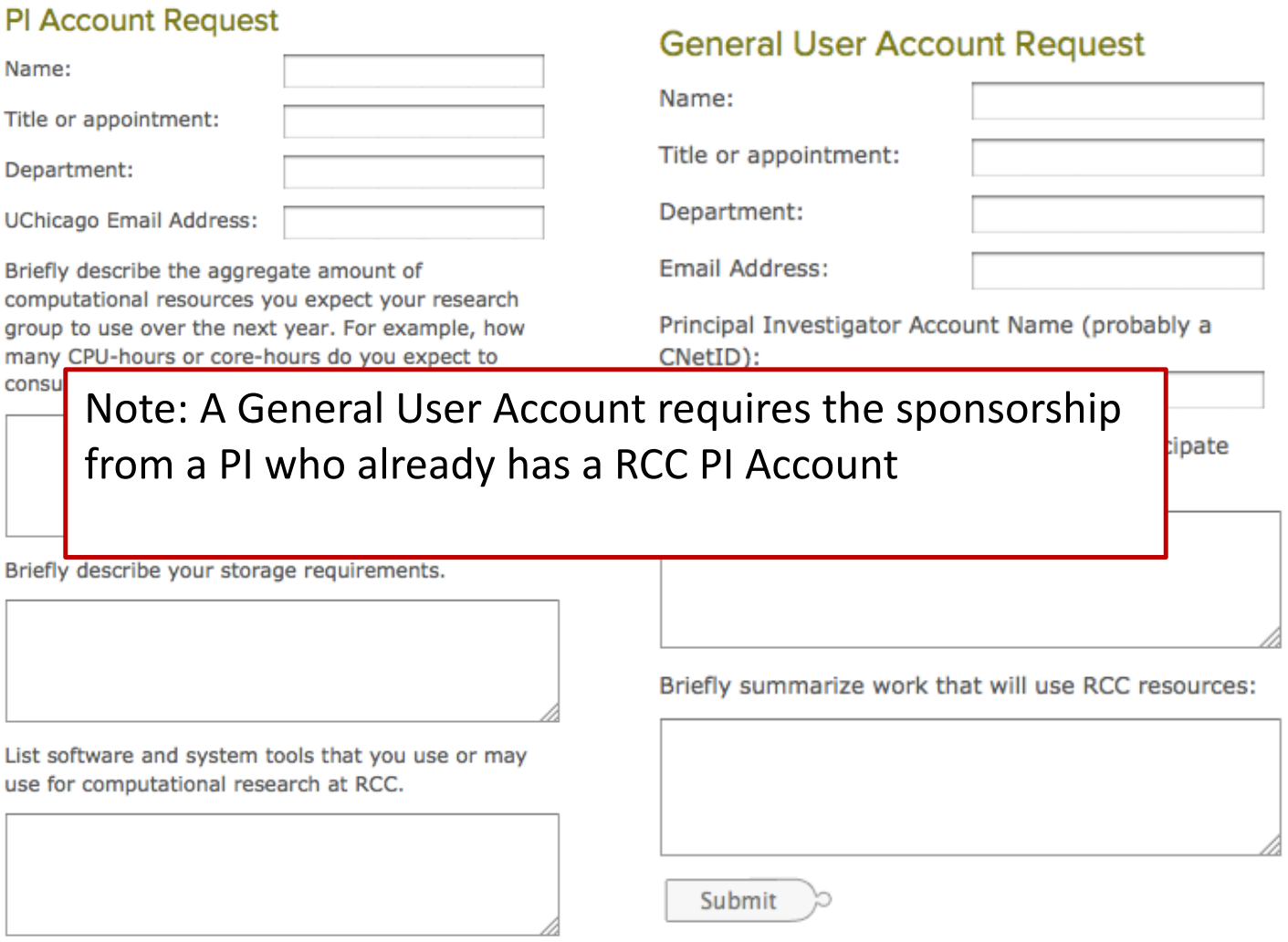
- Service Units (SUs) are a measure of the amount of computing resources consumed on a compute cluster
- SUs are virtual money to do computation on a cluster
- In standard settings, 1 SU equals to the usage of 1 processing unit (e.g., core) for 1 hour
- SUs are allocated through different allocation mechanisms

https://rcc.uchicago.edu/accounts-allocations/calculations-service-units

Getting an Account at RCC

- Complete a form at:

https://rcc.uchicago.edu/accounts-allocations/request-account



PI Account Request

Name:

Title or appointment:

Department:

UChicago Email Address:

Briefly describe the aggregate amount of computational resources you expect your research group to use over the next year. For example, how many CPU-hours or core-hours do you expect to consu

Briefly describe your storage requirements.

List software and system tools that you use or may use for computational research at RCC.

General User Account Request

Name:

Title or appointment:

Department:

Email Address:

Principal Investigator Account Name (probably a CNetID):

ipate

Briefly summarize work that will use RCC resources:

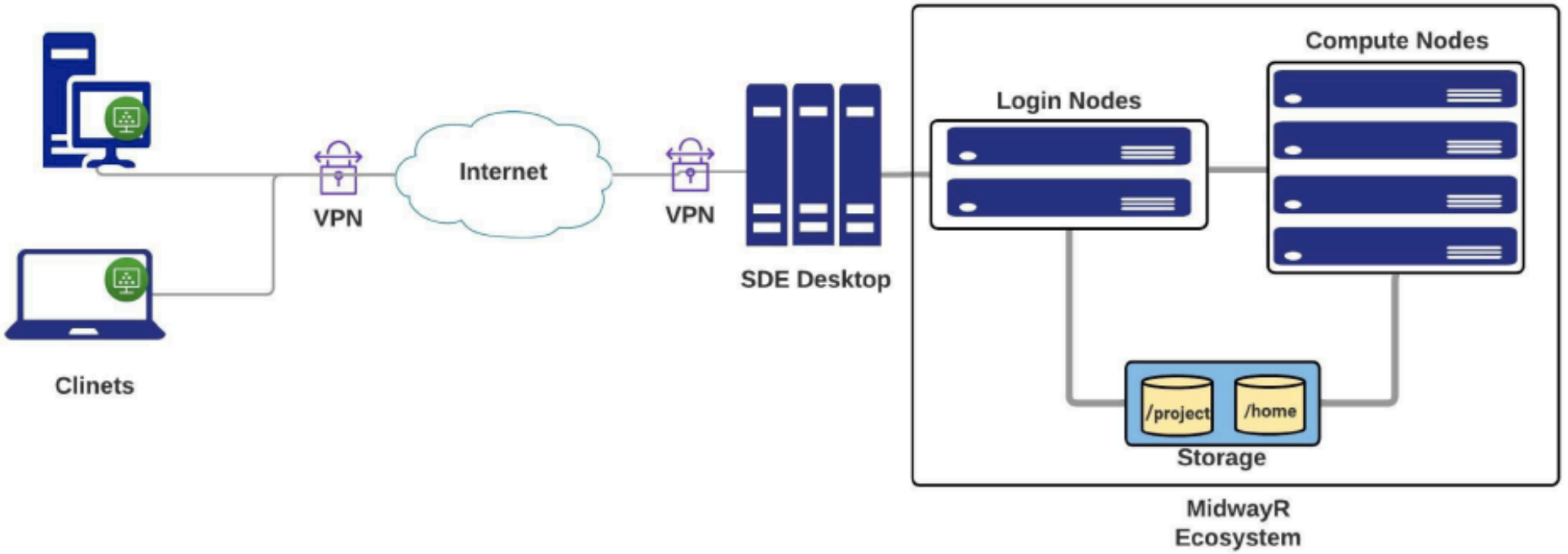
Submit

Note: A General User Account requires the sponsorship from a PI who already has a RCC PI Account

Midway Secure Data Environment (MidwayR)

- Specifically designed to host regulated data
- Two login nodes (each node with 32 cores and 96GB of memory)
- 4 compute nodes (each node with 40 cores and 96GB of memory)
- 441TB of high performance storage
- All nodes are connected using 100Gbps EDR Infiniband
- CentOS 7 operating system
- One large memory node (40 cores and 1TB of memory)

MidwayR Schematic



How to Get a MidwayR Account

- To use the MidwayR ecosystem, you need an RCC MidwayR account: https://sde-midwayr.rcc.uchicago.edu/getting-started/
- Types of an account:
  - Principal Investigator (PI) account
  - General user account
- Eligibility
  - Any UChicago faculty working with sensitive data
  - Any UChicago researcher working with a UChicago faculty
- Prerequisites to get an account
  - Having a DUA executed by URA or generating sensitive data under an IRB
  - Having an initial discussion with the SDE team
  - Approved by the Chief Information Security Officer (CISO)

Accessing Midway: Mac/Linux Users

Midway logins: **midway2.rcc.uchicago.edu** Midway 2

- **Mac/Linux users:**
  Open the terminal app: Applications – utilities -> terminal
  Drag it to your dock so that it is easy to locate in future.

```
w/o X11 [johnnyb@volpe]$ ssh $USER@midway2.rcc.uchicago.edu

X11 enabled: [jonnyb@volpe]$ ssh -X $USER@midway2.rcc.uchicago.edu
```

- **Mac users – enabling X11 forwarding**
  Mac users will not have XQuartz X server installed by default. You will need to download and install X11 server and client from the XQuartz project page.
  NOTE: Users are encouraged to install the older 2.7.8 version of XQuartz, if they intend to run OpenGL applications (e.g. VMD, gaussview, etc).

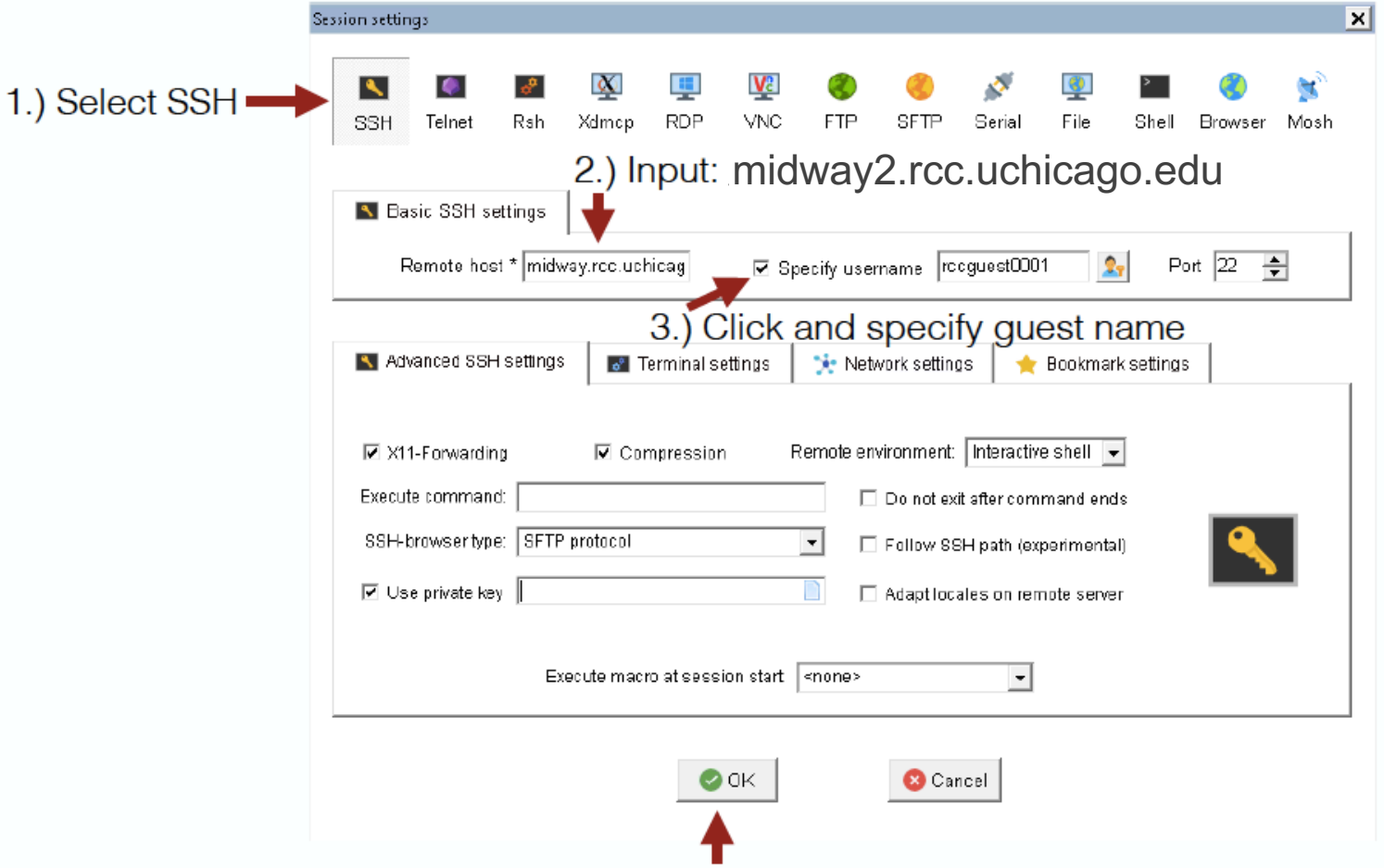
Accessing Midway: Windows Users

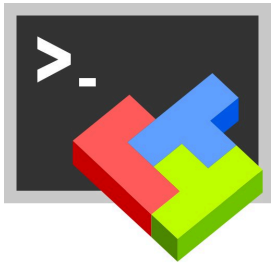
Midway logins: **midway2.rcc.uchicago.edu** Midway 2

- **Windows users:**
  Download either mobaxterm or cygwinX if you don't have it already.
  Strongly encourage windows users to use mobaXterm even if you have previously used putty.
  Windows10 users can use the Windows subsystem for Linux – activate dev mode.

- **MobaXterm**
  Comes bundled with X Server so it is X11 forwarding capable.
  Also has an sftp tab built in to the client permitting easy drag and drop file transfer from remote to local machine.
  To get MobaXterm click the Download tab at top of their site and choose the "Free Home Edition" of MobaXterm.
  Download the MobaXterm Home (Installer edition) zip file and extract contents.

Windows Users: MobaXterm client





Accessing Midway via GUI

We use the Thinlinc Client.
You can either access the webversion of this or download the client

Access from browser the following:
https://midway2.rcc.uchicago.edu

Note: rcc-guest users can NOT use ThinLinc

This is the safest bet for displaying applications that require X-forwarding if you are having trouble working with your ssh X11 forwarding enabled session.

From terminal X forwarding can be achieved with –X or –Y For example:

```
[johnnyb@volpe]$ ssh -Y $USER@midway2.rcc.uchicago.edu
```

For Linux distributions X Server is typically installed by default.
For Windows users using MobaXterm, the default ssh connection is with X11 forwarding enabled.

Transferring Files

- Secure copy protocol: **scp** on Linux and OSX

```
Single files: $ scp <some file> <CNetID>@midway2.rcc.uchicago.edu:
Directories:  $ scp -r <some dir> <CNetID>@midway2.rcc.uchicago.edu:
```

Or, to connect to a directory on Midway (/project2, for example):

```
$ scp -r <some dir> <CNetID>@midway2.rcc.uchicago.edu:/project2
```

- Windows users can use **WinSCP** or the ftp window in mobaXterm
- Users interested in connecting to cloud object storage can use rclone

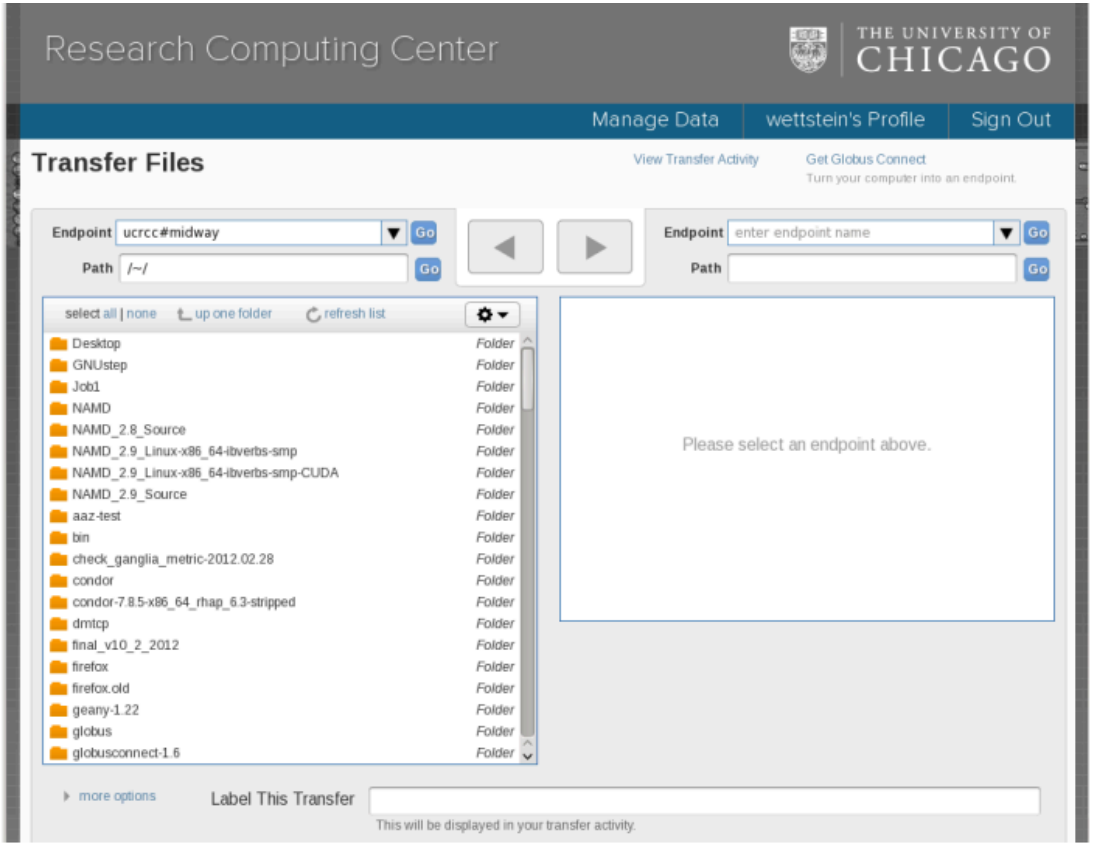
Transferring Files

- **Globus Online**

```
URL:       https://globus.rcc.uchicago.edu
End Point: ucrcc#midway
```

Advantages:

- Uses gridftp for better for many file or large file transfers
- Transfer happens in background so user can end session without disruption of transfer
- Transfer can be interrupted if your local computer is disconnected from internet, will resume upon reconnection.
- Is fault tolerant



HTTP file serving from Midway

- RCC provides web access to some user data from midway

Local path	Corresponding URL
`/home/<cnetid>/public_html/research.dat`	`http://users.rcc.uchicago.edu/~<cnetid>/research.dat`

Be sure your home directories and `public_html` have the execute bit set, and that public_html has read permissions if you would like to allow indexing (directory listings and automatic selection of index.html). For example, these are the commands for setting up Web access to your home directory, where `$HOME` is the environment variable specifying the location of your home directory:

```
chmod o+x $HOME
mkdir -p $HOME/public_html
chmod o+x $HOME/public_html
chmod o+r $HOME/public_html
```

The last line is optional and only needed if you would like to allow directory listing.

Files in public_html must also be readable by the web user (other), but should *not* be made executable, e.g.,

```
chmod o+r $HOME/public_html/research.dat
```

Navigating Midway2 Environment

- The Midway2 nodes are all running linux and use the bash shell by default

Please see the Documentation on Navigating Midway for this part.

Software

- **System Tools and Math Libraries**
  - Compilers: Intel, Portland, GCC
  - Debuggers: Alinea DDT, Intel Roofline
  - Python, Perl, Ruby interpreters
  - MPI, MKL, FFTW, GSL, CUDA, etc.
- **Commercial software**
  - MATLAB, STATA, AMIRA, Gaussian, etc.
- **Domain Specific Applications**
  - chemistry
  - biology
  - Physics
  - etc.

- Use **module system** to manage user's software environment.

Up to date list of software modules installed on midway:

https://rcc.uchicago.edu/docs/software/modulelist.html

Module System

HPC centers typically use a software module system to manage the software packages that loaded into your environment.

This is useful in that you don't have to install the software your self and can selectively choose which software packages are accessible to you so that you can possibly avoid software conflicts.

Useful Module Commands

```
[johnnyb@midway1]$ module help       # information about using module

[johnnyb@midway1]$ module list       # list your currently loaded modules

[johnnyb@midway1]$ module avail      # list all avail software packages

[johnnyb@midway1]$ module load  <package>  # load <package> into your env

[johnnyb@midway1]$ module unload <package> # unload <package> from env
```

Using the Midway Module System

- To see the available versions for a particular software, use the *avail* command.

```
[johnnyb@midway2]$ module avail matlab
```

- To load a software module use the load command. The default version will load if no version is specified.

```
[johnnyb@midway2]$ module load matlab
```

- To see the list of software modules currently loaded in your environment use the *list* command

```
[johnnyb@midway2]$ module list
```

- To get more details about any software module, use the *show* command.

```
[johnnyb@midway2]$ module show matlab
```

Python Modules on Midway

- On **September 2nd 2020 we will be removed many old python modules** and changed the organization and naming convention of python modules on Midway2.

- For information on these changes please see the news link on the RCC website. Link to RCC Restructure Details

- Will primarily maintain Anaconda distribution of python, but the module will be renamed as python. To search availably python modules use module keyword ”python”

```
[johnnyb@midway2]$ module avail python
```

- A python standard python.org distribution of python and intel distribution of python will also be available.

Python Modules on Midway

There are several python modules on midway. What's the difference and which to choose?

- **Recommendation is to use default python module**

```
[johnnyb@midway1]$ module avail python
python/3.7.0                          python/anaconda-2019.03
python/intel-2020.up1python/3.8.5     python/anaconda-2020.02(default)

[johnnyb@midway1]$ module load python/anaconda-2019.03
```

- For all python/anaconda modules the base conda environment contains the python scientific package stack (i.e. matplotlib, jupyter, numpy, scipy, scikit-learn, and pandas)

Installing Python Packages

- People will commonly request that they have xyz python package installed.
- Users can do this themselves. Its recommended that users do so by creating and managing their own conda environment

```
[johnnyb@midway2]$ module load python

[johnnyb@midway2]$ conda create –name=<my-env> python==<python-vers>

[johnnyb@midway2]$ conda activate <my-env>
```

- where <my-env> is replaced by the name you wish to give the environment and <python-vers> is the version of python (e.g. 3.7.6)

- Then from within the environment you can use conda to search and install the packages you require.

```
(my-env)[johnnyb@midway2]$ conda search pymongo

(my-env)[johnnyb@midway2]$ conda install pymongo==3.8
```

Installing Python Packages

- Only resort to using pip or a setup wheel to install a python package within a conda environment it the package is NOT available on any conda channel. Note that conda and pip don't share information about packages installed and this can potentially lead to conflicts.

```
(my-env)[johnnyb@midway2]$ pip search torch-geometric

(my-env)[johnnyb@midway2]$ pip install pip install torch-geometric --user
```

- Notice the --user flag is used to indicate that pip is to install the package in the user's local path. All packages installed with the --user flag are stored in ~/.local unless the user has set the PYTHONUSERBASE environment variable.

- If you pip --user install a package it will be available for that version of python. For example if you are using python 3.7.6 the package will only be available to this version of python.

- Note you can only conda install and pip install packages from the login nodes since the compute nodes do NOT have access to external network.

Running Jupyter Notebooks on Midway

To run a notebook from the **login nodes**:

```
[johnnyb@midway2]$ module load python

[johnnyb@midway2]$ export IP_ADDR=`ifconfig eno1 | grep 'inet ' |  awk '{print $2}'`

[johnnyb@midway2]$ export PORT_NUM=$(shuf -i8000-9000 -n1)

[johnnyb@midway2]$ jupyter notebook --no-browser --ip=${IP_ADDR} --port $PORT_NUM
```

After executing the jupyter command above, a URL will be printed to screen, (e.g. http://10.50.220.71:8117/?token=4d36b0) that you then should copy and paste to your local web browser to connect to the notebook server.

Note: Do not run anything computationally intensive from the login nodes.

Running Jupyter Notebooks on Compute nodes

To run a notebook from a **compute node**

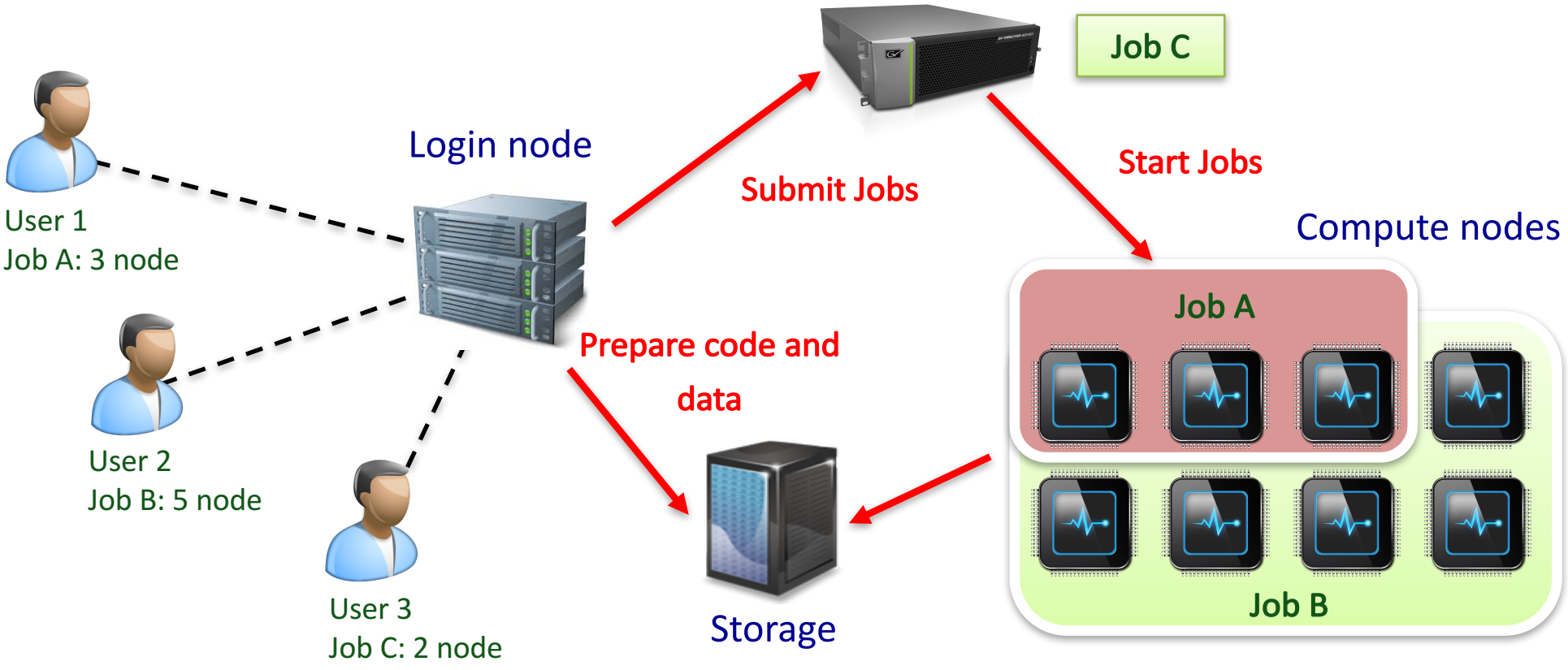
See the git repo: https://git.rcc.uchicago.edu/jhskone/jupyter-lab/tree/master

The *launch-jlab.sh* script in this repo allows one to run a jupyter lab notebook from a compute node with the resources specified in the header of the script.

Note: the compute nodes are only accessible from the campus network. If you are off campus you should first connect to the campus VPN before trying to launch a Jupyter notebook on a compute node to connect to from your local web browser.

Let's take a moment to try this script. We will need to modify it.

Job Scheduling on Midway2



Running Jobs on Midway

Interactive jobs

- Interactively access a node to run directly from the command line:

```
[johnnyb@midway]$ sinteractive --partition=broadwl
```

  - Default partition is broadwl
  - Default wall time is 2 hours
  - Default number of tasks (cores) is 1

- To request a different partition, walltime, and number of tasks you can specify this after the sinteractive command.

```
[johnnyb@midway]$ sinteractive -partition=broadwl-lc -ntasks=14 -time=12:00:00
```

Running Jobs on Midway

SBATCH jobs

- To submit jobs to the SLURM batch job scheduler you need to create a slurm job submission script like the job.sbatch script at right, where you specify the resources your job requires with the #SBATCH directives followed by the set of commands and or application you wish to run.

```
#!/bin/bash

#SBATCH --job-name=test
#SBATCH --output=test.out
#SBATCH --error=test.err
#SBATCH --nodes=1                  # Defulat is 1
#SBATCH –partition=broadwl-lc      # Default is broadwl
#SBATCH --tasks-per-node=14        # Default is 1 core
#SBATCH --time=1:00:00             # Default is 36 hours

# LOAD REQUIRED MODULES
module load Anaconda3/2019.03

# RUN YOUR JOB
python my_app.py
```

- To then submit the job to the scheduler you would issue:

```
[johnnyb@midway2]$ sbatch job.sbatch
```

SLURM Commands

Command	Description
`sbatch script.sbatch`	Submits `script.sbatch` job script
`squeue –u $USER`	Reports the status of your jobs
`sacct –u $USER`	Displays accounting data for your job(s)
`scancel jobid`	Cancels a running job or removes it from the queue
`scontrol show job jobid`	Displays details of a running job

More info on using SLURM: https://slurm.schedmd.com/

Inspecting Your Submitted Job

- The user can directly login to the node that their job is running on and observe the progress of the job.

```
squeue –u <userid>
```

Will return job ids of jobs you have running or pending.
Get a running jobs list of nodes it is using:

```
squeue -O nodelist -j <job_id>
```

Login directly to that node and run commands `top` and `free –g`

If the job is not running check its priority:

```
squeue -O prioritylong -j <job_id>
```

```
squeue -p broadwl –O jobid,numnodes,reason,timelimit,prioritylong
```

Other Useful User Tools: rcchelp

The **rcchelp** command is meant to be a Swiss army knife for information. Simply type *rcchelp* to get a full list of options.

Get info about your account and groups you belong to:

```
rcchelp user <userid>
```

Get information about particular partition:

```
rcchelp sinfo -p broadwl
```

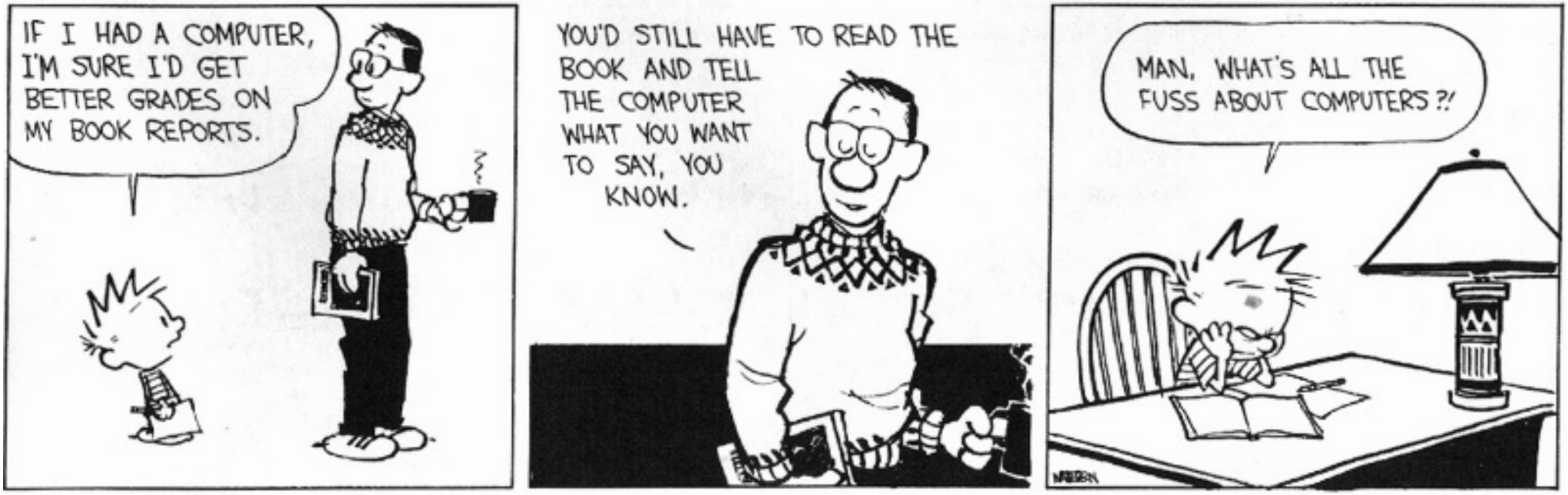
Get information about your resource consumption:

```
rcchelp usage
```

Get information about storage use you have access to:

```
rcchelp quota
```

RCC Help



- Email: help@rcc.uchicago.edu
- Web: rcc.uchicago.edu
- Phone: 773-795-2667
- Walk-in: Regenstein Library, suite 216

END OF SLIDE SHOW