UC **SANTA BARBARA**

CNSI MRL

CSC Center for Scientific Computing

ENTERPRISE TECHNOLOGY SERVICES

HPC Workshop 2

Feb. 28, 2024

02:00 – 03:00 pm (followed by dessert)

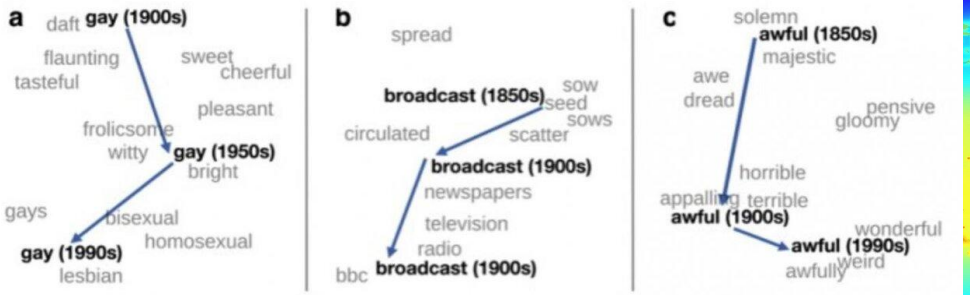
Location: Elings Hall 1601

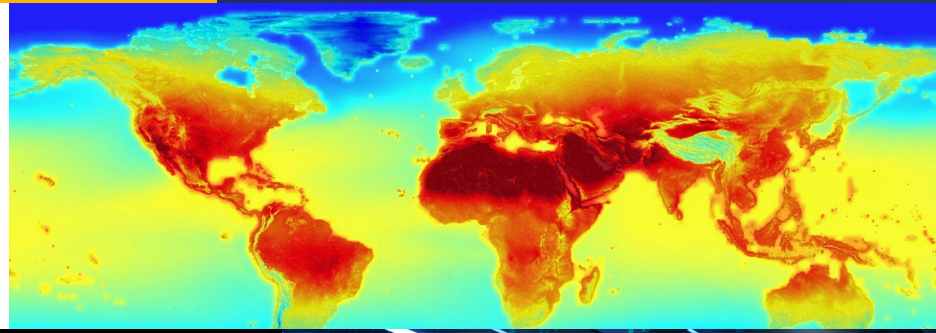
Register @ https://csc.cnsi.ucsb.edu

Quickly start using HPC resource at UCSB

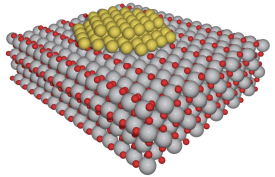
- SLURM Array and Job Dependency
- Running Jupyter Notebook/Lab and VS Code on the Cluster
- NSF ACCESS allocation
- National & Commercial Cloud Computing Resources

Computational Linguistics





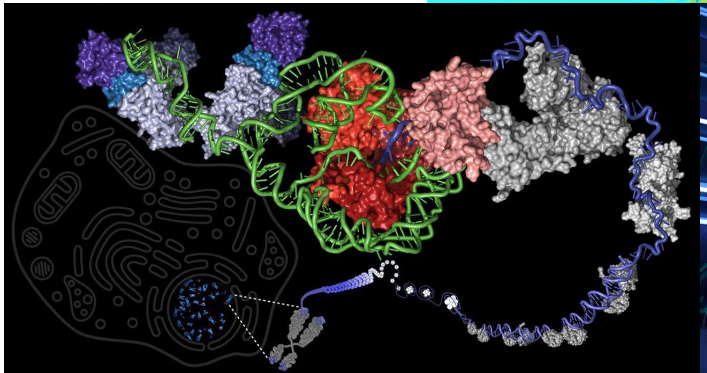
KS-DFT



The total energy in Kohn-Sham Density Functional Theory (KS-DFT) is expressed as

$$E_{total} = T_s + \int d\boldsymbol{r} V_{ext}(\boldsymbol{r})\rho(\boldsymbol{r}) + E_{xc}[\rho] + \frac{1}{2}\int\int d\boldsymbol{r}d\boldsymbol{r}' \frac{\rho(\boldsymbol{r})\rho(\boldsymbol{r}')}{|\boldsymbol{r}-\boldsymbol{r}'|}$$

Kohn-Sham kinetic energy | External energy | XC energy | Hartree energy



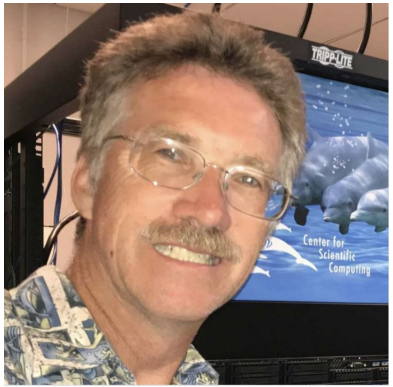
UC **SANTA BARBARA**

Introduction to High-Performance Computing (HPC)

Paul Weakliem, Fuzzy Rogers, and Jay Chi

February 28, 2024

Our Team



Paul Weakliem, PhD
Co-Director
Center for Scientific Computing &
California Nanosystems Institute
Eling 3231
weakliem@cnsi.ucsb.edu

Fuzzy Rogers
Research Computing Administrator
Center for Scientific Computing &
Materials Research Laboratory
MRL 2066B
fuz@mrl.ucsb.edu

Yu-Chieh “Jay” Chi, PhD
Research Computing Consultant
Center for Scientific Computing &
Enterprise Technology Services
Elings 3229
jaychi@ucsb.edu

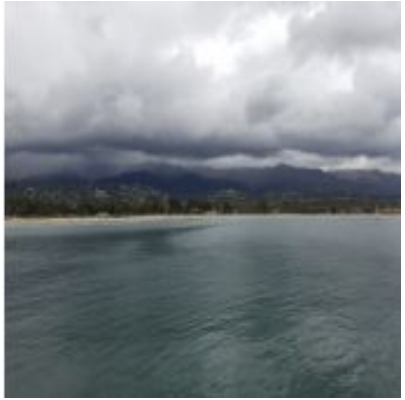
CSC Center for Scientific Computing

CNSI

MRL

ENTERPRISE TECHNOLOGY SERVICES

Our Research IT Partners



Letters & Science
INFORMATION TECHNOLOGY

Mike Edwards
Director of Engineering Computing Infrastructure
3152A Harold Frank Hall
mcs@engineering.ucsb.edu

Michael Colee
Director of Earth Research Institute Computing (ERI)
6703 Ellison Hall
mtc@eri.ucsb.edu

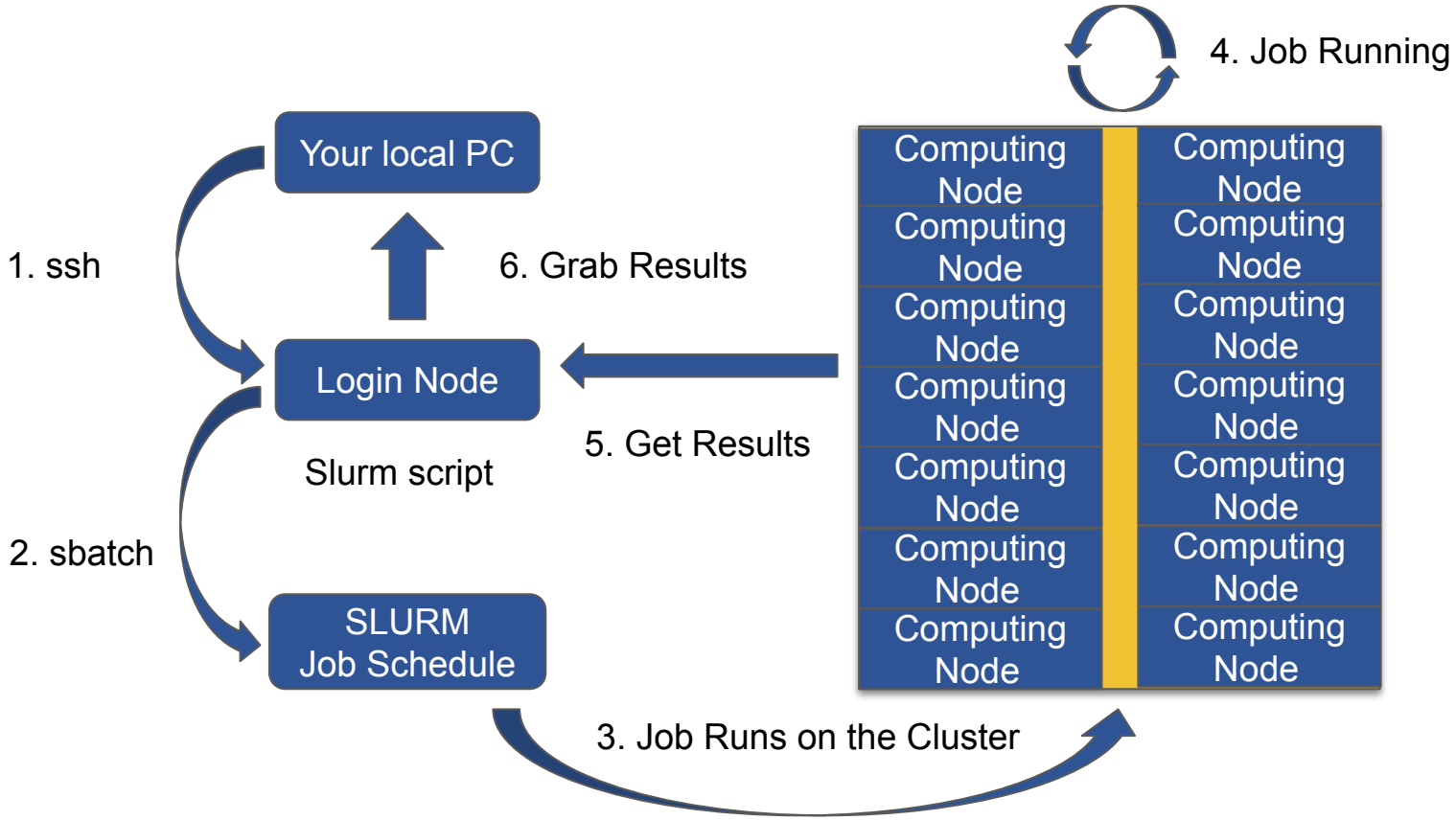
Ted Cabeen
Director of Life Science Computing Group (LSCG)
2306 Life Science
ted.cabeeen@lscg.ucsb.edu

Andreas Boschke
Director of Infor. Infrastructure at Letter & Science IT (LSIT)
2306 Life Science
andreas@lsit.ucsb.edu

Agenda

- HPC Workflow
- SLURM Array
- SLURM Job Dependency
- Running Jupyter Notebook/Lab and VS Code on the cluster
  - Running Jupyter Notebook/lab on the POD/Braid2
  - Running VS Code on the POD/Braid2
  - Google Colab
  - Running Jupyter Lab on the SDSC Expanse
- Introduction to National HPC/Supercomputer resources
  - ACCESS allocation
  - Cloud Computing: JetStream2 from Indiana University

General HPC Workflow



Job Arrays

- According to the Slurm Workload Manager, “Job arrays offer a mechanism for submitting and managing collections of similar jobs quickly and easily, … . All jobs must have the same initial options (e.g., size, time limit, etc.)”
- In general, job arrays are useful for applying the same processing routine to a collection of multiple input data files. Job arrays offer a very simple way to submit a large number of independent processing jobs.

```
#!/bin/bash
#SBATCH -J 'slurmArray'                    ### Job Name
#SBATCH --nodes=1                          ### No. of Nodes
#SBATCH --ntasks=1                         ### No. of Tasks
#SBATCH -p short                           ### Submit the job to Partition (Optional)
#SBATCH -o outLog_%A_%a                    ### Output Log File (Optional)
#SBATCH -e errLog_%A_%a                    ### Error Log File (Optional but suggest to have it)
#SBATCH -t 00:10:00                        ### Job Execution Time
#SBATCH –array=0-3
#SBATCH --mail-user=usernam@ucsb.edu       ### Mail to you (Optional)
#SBATCH --mail-type ALL                    ### Mail send you when the job starts and end (Optional)
```

Slurm Job Array Submission script

- The %A_%a construct in the output and error file names is used to generate unique output and error files based on the master job ID (%A) amd the array-tasks ID (%a).
- Job Array indices can be specified array index values, a range of index values, and an optional step size.

```
# Submit a job array with index values between 0 and 15
#SBATCH –array=0-15

# Submit a job array with index values of 1, 3, 9, and 15
#SBATCH –array=1, 3, 9, 15

# Submit a job array with index values between 1 and 16 with a step size of 2
#SBATCH –array=1-16:2
```

Slurm Job Array Submission script

```
#!/bin/bash
#SBATCH -J 'slurmArray'
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH -p short
#SBATCH -o outLog_%A_%a
#SBATCH -e errLog_%A_%a
#SBATCH -t 00:10:00
#SBATCH –array=0-3

echo "SLURM_JOB_ID: " $SLURM_JOBID
echo "SLURM_ARRAY_JOB_ID: " $SLURM_ARRAY_JOB_ID
echo "SLURM_ARRAY_TASK_ID: " $SLURM_ARRAY_TASK_ID
echo "SLURM_ARRAY_TASK_COUNT: " $SLURM_ARRAY_TASK_COUNT
echo "SLURM_ARRAY_TASK_MAX: " $SLURM_ARRAY_TASK_MAX
echo "SLURM_ARRAY_TASK_MIN: " $SLURM_ARRAY_TASK_MIN
```

```
SLURM_JOB_ID:  3405632
SLURM_ARRAY_JOB_ID:  3405629
SLURM_ARRAY_TASK_ID:  0
SLURM_ARRAY_TASK_COUNT:  4
SLURM_ARRAY_TASK_MAX:  3
SLURM_ARRAY_TASK_MIN:  0

SLURM_JOB_ID:  3405633
SLURM_ARRAY_JOB_ID:  3405629
SLURM_ARRAY_TASK_ID:  1
SLURM_ARRAY_TASK_COUNT:  4
SLURM_ARRAY_TASK_MAX:  3
SLURM_ARRAY_TASK_MIN:  0

SLURM_JOB_ID:  3405629
SLURM_ARRAY_JOB_ID:  3405629
SLURM_ARRAY_TASK_ID:  3
SLURM_ARRAY_TASK_COUNT:  4
SLURM_ARRAY_TASK_MAX:  3
SLURM_ARRAY_TASK_MIN:  0
```

Dependency Jobs

- You can schedule jobs depending on the termination status of previously scheduled jobs. This way, you can concatenate your jobs into a pipeline or expand to more complicated dependencies.
- For example, job1.s is a submission script you plan to submit a batch job:

```
#!/bin/bash
#SBATCH -J 'JobDep1'                    ### Job Name
#SBATCH --nodes=1                       ### No. of Nodes
#SBATCH --ntasks=1                      ### No. of Tasks
#SBATCH -p short                        ### Submit the job to Partition (Optional)
#SBATCH -o outLog_%x_%j                 ### Output Log File (Optional)
#SBATCH -e errLog_%x_%j                 ### Error Log File (Optional but suggest to have it)
#SBATCH -t 00:10:00                     ### Job Execution Time
#SBATCH --mail-user=usernam@ucsb.edu    ### Mail to you (Optional)
#SBATCH --mail-type ALL                 ### Mail send you when the job starts and end (Optional)

# Run Bash Command
echo "***** My first Program *****"
echo "***** Prepare the Data *****"
echo "*****Done for Parparation *****"
echo "Time: " $(date +"%T")
```

Dependency Jobs

- Submit the job script to the Slurm job scheduler from the POD login node:

  $ sbatch job.s
  Submitted batch job 1234567

- You can submit another job that is put on the waiting list of the queue.

  $ sbatch -dependency=aftercorr:1234567 job2.s

- This command indicates that job2.s will be put in the queue after the job ID 1234567 is terminated for any reason. The *dependency option flag* can be after, afterany, aftercorr, afterok, and afternotok.
- The following command would submit 2 jobs with respect to their dependencies.

```
# First Job
jobID_1=$(sbatch job1.s | cut -f 4 -d' ')

# Second Job - this job depends on the first job
sbatch --dependency=aftercorr:$jobID_1 job2.s
```

Slurm Job Dependency Submission script

after	This job is execution after the specified jobs have begun execution
afterany	This job can begin execution after the specified jobs have been terminated
aftercorr	A task of this job array can begin execution after the corresponding task ID in the specified job has completed successfully
afternotok	This job can begin execution after the specified jobs have terminated in some failed state
afterok	This job can begin execution after the specified jobs have been successfully executed
singleton	This job can begin execution after any previously launched jobs sharing the same job name and the user has terminated

Slurm Job Dependency Submission script

Slurm job script file: job1.s

```
#!/bin/bash
#SBATCH -J 'JobDep1'                    ### Job Name
#SBATCH --nodes=1                       ### No. of Nodes
#SBATCH --ntasks=1                      ### No. of Tasks
#SBATCH -p short                        ### Submit the job to Partition (Optional)
#SBATCH -o outLog_%x_%j                 ### Output Log File (Optional)
#SBATCH -e errLog_%x_%j                 ### Error Log File (Optional but suggest to have it)
#SBATCH -t 00:10:00                     ### Job Execution Time
#SBATCH --mail-user=usernam@ucsb.edu    ### Mail to you (Optional)
#SBATCH --mail-type ALL                 ### Mail send you when the job starts and end (Optional)

# Run Bash Command
echo "***** My first Program *****"
echo "***** Prepare the Data *****"
echo "***** Done for Parparation *****"
echo "Time: " $(date +"%T")
```

Slurm Job Dependency Submission script

Slurm job script file: job2.s

```
#!/bin/bash
#SBATCH -J 'JobDep2'                    ### Job Name
#SBATCH --nodes=1                       ### No. of Nodes
#SBATCH --ntasks=1                      ### No. of Tasks
#SBATCH -p short                        ### Submit the job to Partition (Optional)
#SBATCH -o outLog_%x_%j                 ### Output Log File (Optional)
#SBATCH -e errLog_%x_%j                 ### Error Log File (Optional but suggest to have it)
#SBATCH -t 00:10:00                     ### Job Execution Time
#SBATCH --mail-user=usernam@ucsb.edu    ### Mail to you (Optional)
#SBATCH --mail-type ALL                 ### Mail send you when the job starts and end (Optional)

# Run Bash Command
echo "***** Start the Program *****"
echo "***** Code1 is Running *****"
echo "***** Code2 is Running *****"
echo "***** End the Program *****"
echo "Time: " $(date +"%T")
```

Slurm Job Dependency Submission script

Slurm job script file: job3.s

```
#!/bin/bash
#SBATCH -J 'JobDep3'                    ### Job Name
#SBATCH --nodes=1                       ### No. of Nodes
#SBATCH --ntasks=1                      ### No. of Tasks
#SBATCH -p short                        ### Submit the job to Partition (Optional)
#SBATCH -o outLog_%x_%j                 ### Output Log File (Optional)
#SBATCH -e errLog_%x_%j                 ### Error Log File (Optional but suggest to have it)
#SBATCH -t 00:10:00                     ### Job Execution Time
#SBATCH --mail-user=usernam@ucsb.edu    ### Mail to you (Optional)
#SBATCH --mail-type ALL                 ### Mail send you when the job starts and end (Optional)

# Run Bash Command
echo "***** The Last Step *****"
echo "***** Analyze the Data *****"
echo "***** Done for analyzing data *****"
echo "Time: " $(date +"%T")
```

Slurm Job Dependency Submission script

batch script file: depJOB.s

```
#!/bin/bash

# First Job
jobID_1=$(sbatch job1.s | cut -f 4 -d' ')

# Second Job - this job depends on the first job
jobID_2=$(sbatch --dependency=aftercorr:$jobID_1 job2.s | cut -f 4 -d' ')

# Third Job - this job also depends on the second job
sbatch --dependency=aftercorr:$jobID_2 job3.s
```

- Execute the batch job script from the POD login node:

  $ sh depJOB.s
  Submitted batch job 1234567

Running Interactive Job

- Interactive computing refers to software which accepts input from the user as it runs. **Interactive computing** involves *real-time* user inputs to perform tasks on a set of compute node(s) including:
  - Coe development, real-time data exploration, etc.
  - Used when applications have large data sets or are too large to download to local device, or too large to compute on the local device
  - Actions performed on remote compute nodes as a result of user input or program out.
- To request an interactive computing node with 4 cores for 4 hours:

```
$ srun -N 1 -n 4 -p batch --time=4:00:00 --pty bash -i
```

- To request an interactive computing GPU node for 4 hours:

```
$ srun -N 1 -n 1 -p gpu  –gres=gpu:1 --time=4:00:00 --pty bash -i
```

Why Run Jupyter Notebook/Lab, VS Code on the cluster?

- Computational resource requirement (GPU, multiple Cores, and etc.)
- Large memory requirement for your data
- Convenience to analyze your large scale data on the cluster
- Scaling up to long runtimes

Set Up Your Jupyter Notebook on the POD/Braid2

- Get to a compute node from the login node

$ srun -N 1 -n 1 -p gpu –gres=gpu:1 --time=04:00:00 --pty bash -i

```
[jay@pod-login1 ~]$
```

```
[jay@node122 ~]$
```

- Make sure your conda environment is activated

```
(base) [jay@node122 ~]$
```

- Activate the specific conda environment

$ conda activate pytorch112_gpu116

```
alphafold                /home/jay/Softwares/anaconda3/envs/alphafold
biosynthesis             /home/jay/Softwares/anaconda3/envs/biosynthesis
biosynthesis_test        /home/jay/Softwares/anaconda3/envs/biosynthesis_test
gnina_env                /home/jay/Softwares/anaconda3/envs/gnina_env
methylC_analyzer_env     /home/jay/Softwares/anaconda3/envs/methylC_analyzer_env
multiProcess             /home/jay/Softwares/anaconda3/envs/multiProcess
p4dev                    /home/jay/Softwares/anaconda3/envs/p4dev
pytorch112_gpu116        /home/jay/Softwares/anaconda3/envs/pytorch112_gpu116
pytorch201_gpu117        /home/jay/Softwares/anaconda3/envs/pytorch201_gpu117
pytorch_cpu              /home/jay/Softwares/anaconda3/envs/pytorch_cpu
```

```
(base) [jay@node122 ~]$ conda activate pytorch112_gpu116
(pytorch112_gpu116) [jay@node122 ~]$
```

Set Up Your Jupyter Notebook on the POD/Braid2

- Make sure the jupyter has been installed in the conda environment

$ conda list jupyter

```
(pytorch112_gpu116) [jay@node122 ~]$ conda list jupyter
# packages in environment at /home/jay/Softwares/anaconda3/envs/pytorch112_gpu116:
#
# Name                    Version                   Build  Channel
jupyter-client            8.3.1                    pypi_0    pypi
jupyter-core              5.3.2                    pypi_0    pypi
jupyter-events            0.7.0                    pypi_0    pypi
jupyter-lsp               2.2.0                    pypi_0    pypi
jupyter-server            2.7.3                    pypi_0    pypi
jupyter-server-terminals  0.4.4                    pypi_0    pypi
jupyterlab                4.0.6                    pypi_0    pypi
jupyterlab-pygments       0.2.2                    pypi_0    pypi
jupyterlab-server         2.25.0                   pypi_0    pypi
```

- Get the ip from the host

$ hostname -i

```
(pytorch112_gpu116) [jay@node122 ~]$ hostname -i
10.1.50.122
```

- Launch the Jupyter notebook from the server

$ jupyter-notebook --no-browser --port=8888 --ip=10.1.50.122

```
    To access the server, open this file in a browser:
        file:///home/jay/.local/share/jupyter/runtime/jpserver-88004-open.html
    Or copy and paste one of these URLs:
        http://10.1.50.122:8888/tree?token=bff1d6512a520a13deb981d6c627d791198e25210536a840
        http://127.0.0.1:8888/tree?token=bff1d6512a520a13deb981d6c627d791198e25210536a840
[I 2024-02-26 15:29:41.318 ServerApp] Skipped non-installed server(s): bash-language-server, dockerfile-language
-server-nodejs, javascript-typescript-langserver, jedi-language-server, julia-language-server, pyright, python-l
anguage-server, python-lsp-server, r-languageserver, sql-language-server, texlab, typescript-language-server, un
ified-language-server, vscode-css-languageserver-bin, vscode-html-languageserver-bin, vscode-json-languageserver
-bin, yaml-language-server
```

Set Up Your Jupyter Notebook on the POD/Braid2

- Open a new terminal in order to access the Jupyter notebook from your remote machine over ssh

  $ ssh -N -L 8888:10.1.50.122:8888 your_user_name@pod.cnsi.ucsb.edu

```
(base) EEUC-YT61Y2PL:~ jaychi$ ssh -N -L 8888:10.1.50.122:8888 jay@pod-login1.cnsi.ucsb.edu
```

- Open a browser window, copy the http://127.0.0.1:8888/tree?token=44b5cf8a93ea5edf5e70f202e81480e07aef19690bcd2c22 and pate it to the browser.

- After you finish your job, don't forget to release your resource.

  $ scancel your_job_id

Set Up Your Jupyter Notebook on the Braid

- Get to a compute node from the login node

$ qsub -I -l nodes=1:ppn=2 -l walltime=02:00:00

```
(base) -bash-4.1$ qsub -I -l nodes=1:ppn=4 -l walltime=02:00:00
qsub: waiting for job 5343731.braid.cnsi.ucsb.edu to start
qsub: job 5343731.braid.cnsi.ucsb.edu ready

-bash-4.1$
```

- Make sure your conda environment is activated

```
(base) -bash-4.1$
```

- Get the ip from the host

$ hostname -i

```
(base) -bash-4.1$ hostname -i
10.0.90.50
```

Set Up Your Jupyter Notebook on the Braid

- Make sure the jupyter has been installed in the conda environment

  $ conda list jupyter

```
jupyter                   1.0.0             pyhd8ed1ab_10    conda-forge
jupyter-lsp               2.2.2              pyhd8ed1ab_0    conda-forge
jupyter_client            8.6.0              pyhd8ed1ab_0    conda-forge
jupyter_console           6.6.3              pyhd8ed1ab_0    conda-forge
jupyter_core              5.7.1           py310hff52083_0    conda-forge
jupyter_events            0.9.0              pyhd8ed1ab_0    conda-forge
jupyter_server            2.12.5             pyhd8ed1ab_0    conda-forge
jupyter_server_terminals  0.5.1              pyhd8ed1ab_0    conda-forge
```

- Launch the Jupyter notebook from the server

  $ jupyter notebook --no-browser --port=8888 --ip=10.0.90.50

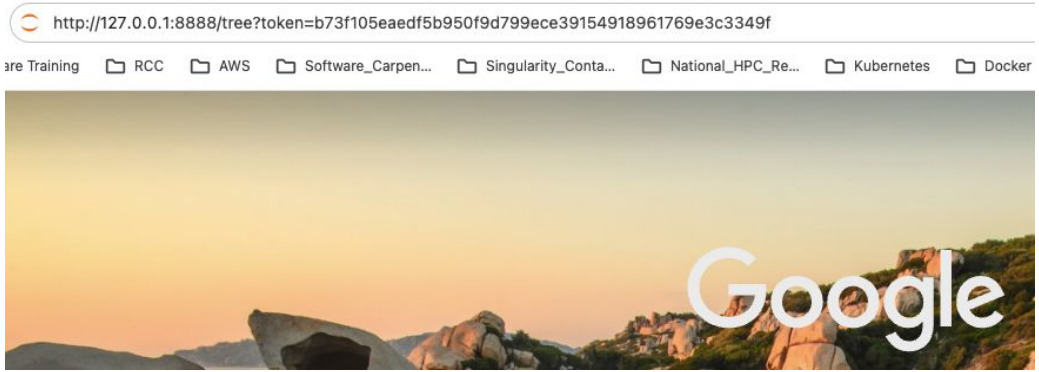
```
To access the server, open this file in a browser:
    file:///home2/jay/.local/share/jupyter/runtime/jpserver-20452-open.html
Or copy and paste one of these URLs:
    http://10.0.90.50:8888/tree?token=b73f105eaedf5b950f9d799ece39154918961769e3c3349f
    http://127.0.0.1:8888/tree?token=b73f105eaedf5b950f9d799ece39154918961769e3c3349f
```

Set Up Your Jupyter Notebook on the Braid

- Open a new terminal in order to access the Jupyter notebook from your local machine over ssh
  $ ssh -oHostKeyAlgorithms=+ssh-rsa -N -L 8888:10.0.90.50:8888 your_username@braid.cnsi.ucsb.edu

```
(base) EEUC-YT61Y2PL:~ jaychi$ ssh -oHostKeyAlgorithms=+ssh-rsa -N -L 8888:10.0.
90.50:8888 jay@braid.cnsi.ucsb.edu
jay@braid.cnsi.ucsb.edu's password:
```

- Open a browser window, copy the http://127.0.0.1:8888/tree?token=b73f105eaedf5b950f9d799ece39154918961769e3c3349f and pasta it to the browser.



Connect Visual Studio Code to POD

- According to the Wikipedia, "*Visual Studio Code (VS Code) is a source code editor that support a variety of programming languages, including C, C#, C++, Fortran, Go, Java, JavaScript, Node.js, Python, Rust, and Julia*."

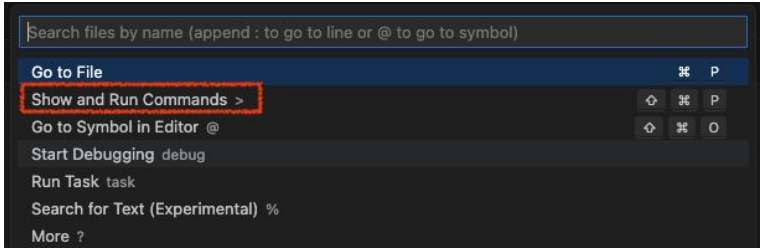
- GitHub Copilot is a code completion tool developed by GitHub and OpenAI that assists users of Visual Studio Code integrated development environments (IDEs) by autocompleting code.
- Get to a compute node from the login node

$ srun -N 1 -n 1 -p gpu –gres=gpu:1 --time=04:00:00 --pty bash -i

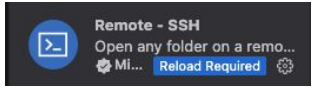
```
[jay@pod-login1 ~]$ srun -N 1 -n 1 -p gpu --gres=gpu:1 --time=04:00:00 --pty bash -i
```

Connect Visual Studio Code to POD

- Connect VS Code locally to the Computing Node in HPC
  - Open VS Code command palette



  - Install Remote - SSH

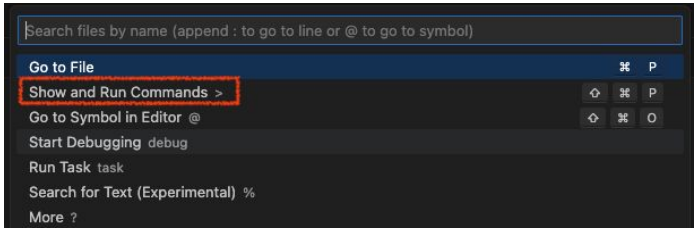


- Configuring SSH
  - Open SSH config file
  - Add the following config detail

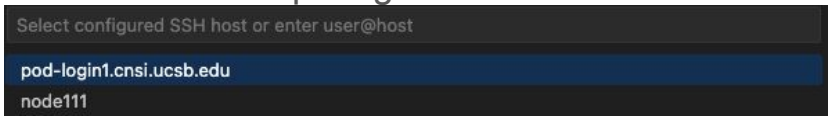

```
Host pod-login1.cnsi.ucsb.edu
  HostName pod-login1.cnsi.ucsb.edu
  User jay

Host node111
  ProxyJump pod-login1.cnsi.ucsb.edu
  User jay
```

Connect Visual Studio Code to POD

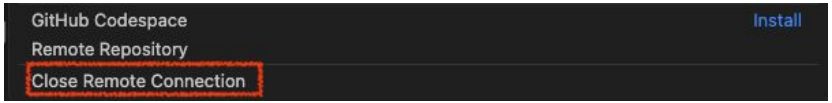
- Open VS Code command palette
  - Remote-SSH: Connect to Host



- Choose the computing node

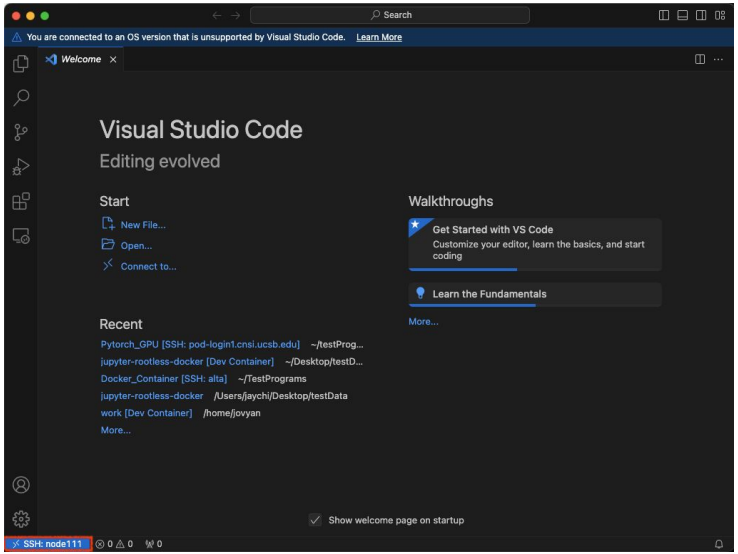


- Disconnect to the HPC



- After you finish your job, don’t forget to release your resource.

  $ scancel your_job_id

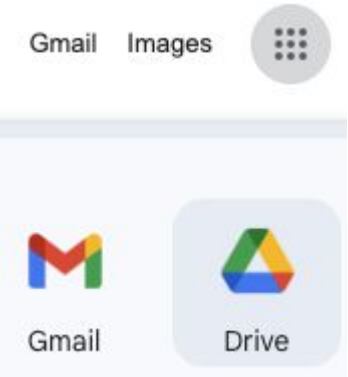


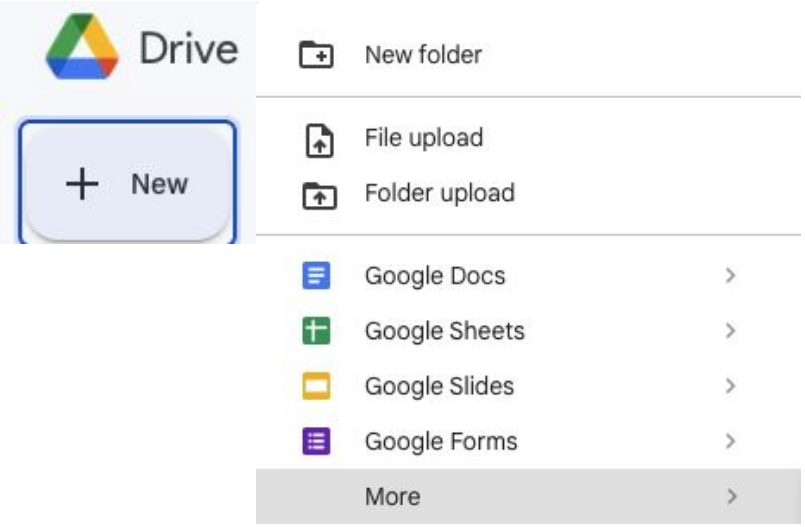
Google Colab

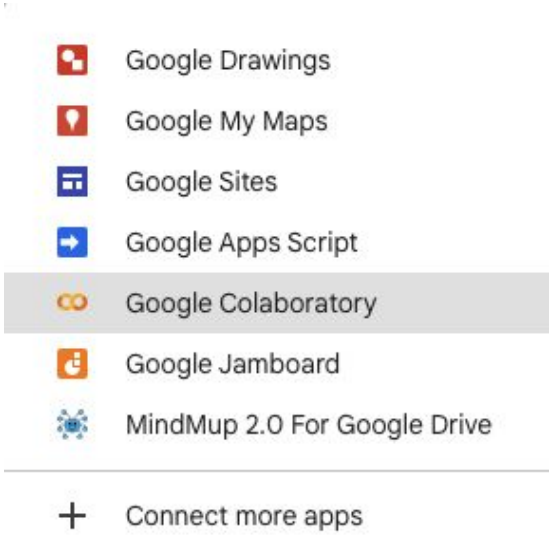
- What is Google Colab?
  - Google Colab(Colaboratory) allows you to write and execute Python and R in your browser with
    - No need to install packages
    - GPU access
    - Sharing with your partners

- Tutorial link
  - https://reurl.cc/Epg3M0

Getting Started to use Colab







Getting Started to use Colab

```python
import numpy as np

arrA = np.array([[1., 2., 3.], [2., 3., 4.]])
print('Arrary A:\n', arrA)
print('\nThe shape of Array is: ', arrA.shape)
print('\nIf GPU is available: ', torch.cuda.is_available())
```

```
Arrary A:
 [[1. 2. 3.]
 [2. 3. 4.]]

The shape of Array is:  (2, 3)

If GPU is available:  False
```

Executing the Code Block: Shift + return

```
# This is my first Colab
## 1.1 Introduction the Colab
### 1.1.1 Introduction to Python

### 1.1.2 Introduction to Torch
```

Add code cell
⌘/Ctrl+M B

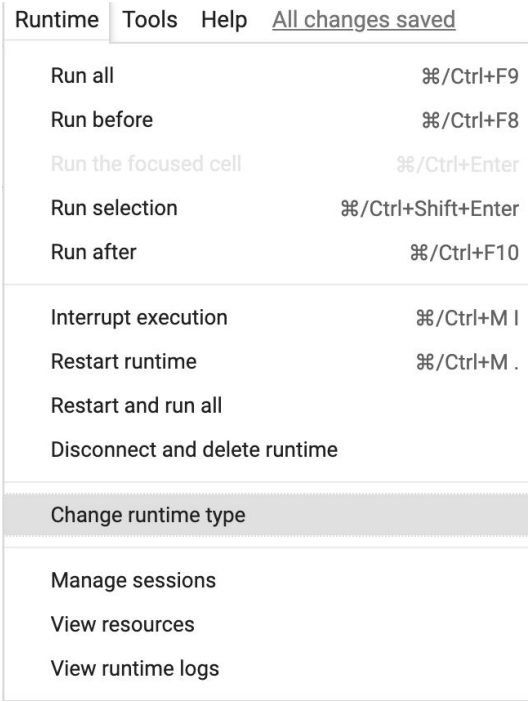
This is my first Colab

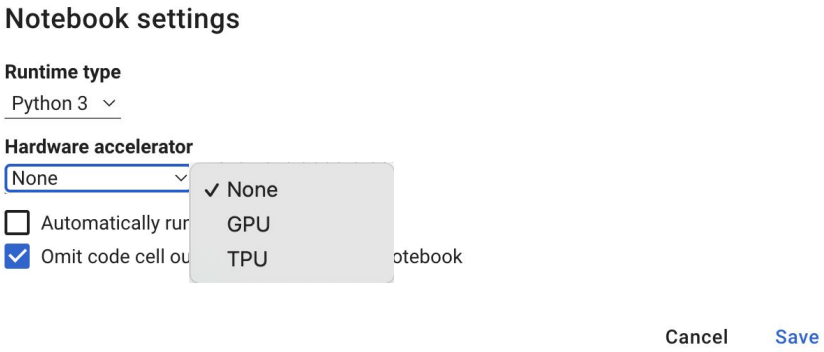
1.1 Introduction the Colab

1.1.1 Introduction to Python

1.1.2 Introduction to Torch

Changing Runtime Type





Run Jupyter Notebook/Lab on the SDSC Expanse

- Connect to the Expanse Portal: https://portal.expanse.sdsc.edu/

Expanse User Guide

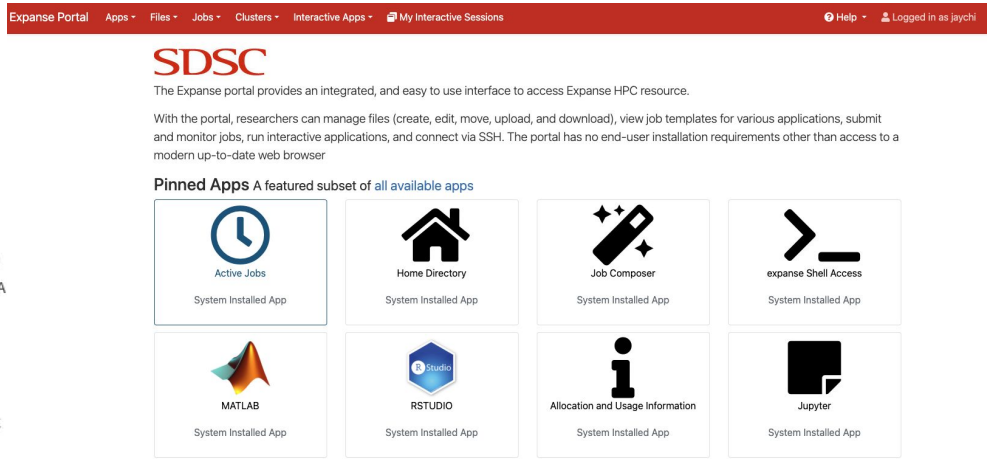
Technical Summary

Expanse is a dedicated Advanced Cyberinfrastructure Coordination Ecosystem: Services and Support Advanced Cyberinfrastructure Coordination Ecosystem: Services and Support (ACCESS) cluster designed by Dell and SDSC delivering 5.16 peak petaflops, and will offer Composable Systems and Cloud Bursting.

Expanse's standard compute nodes are each powered by two 64-core AMD EPYC 7742 processors and contain 256 GB of DDR4 memory, while each GPU node contains four NVIDIA V100s (32 GB SMX2) connected via NVLINK and dual 20-core Intel Xeon 6248 CPUs. *Expanse* also has four 2 TB large memory nodes.

Expanse is organized into 13 SDSC Scalable Compute Units (SSCUs), comprising 728 standard nodes, 54 GPU nodes and 4 large-memory nodes. Every *Expanse* node has access to a 12 PB Lustre parallel file system (provided by Aeon Computing) and a 7 PB Ceph Object Store system. *Expanse* uses the Bright Computing HPC Cluster management system and the SLURM workload manager for job scheduling.

Expanse Portal Login



Run Jupyter Notebook/Lab on the SDSC Expanse

Open OnDemand / Jupyter Session

Jupyter Session

Account:

Partition (Please choose the gpu, gpu-shared, or gpu-preempt as the partition if using gpus):

shared

Time limit (min):

30

Number of cores:

1

Memory required per node (GB):

2

GPUs (optional):

0

Singularity Image File Location: (Use your own or to include from existing container library /cm/shared/apps/containers/singularity/pytorch/pytorch-latest.sif)

Environment modules to be loaded (E.g., to use latest version of system Anaconda3 includ

Conda Environment (Enter your own conda environment if any):

Satellite Reverse Proxy Service

SDSC Expanse

Job State: Mapped

In Queue — Running — Mapped — Proxied

In Queue
Job has not yet started.

Running
Job has started, but has not redeemed Satellite Token.

Mapped
Job has redeemed Satellite Token, but no proxy entry exists yet.

Proxied
Proxy entry created, ready to go!

Dead
Job died or exited, no further progress will occur.

ACCESS

Free National Supercomputer Resources

ACCESS | Advancing Innovation

National Supercomputer Resources: ACCESS

- Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS)

- ACCESS is an advanced computing and data resource supported by the National Science Foundation (NSF).

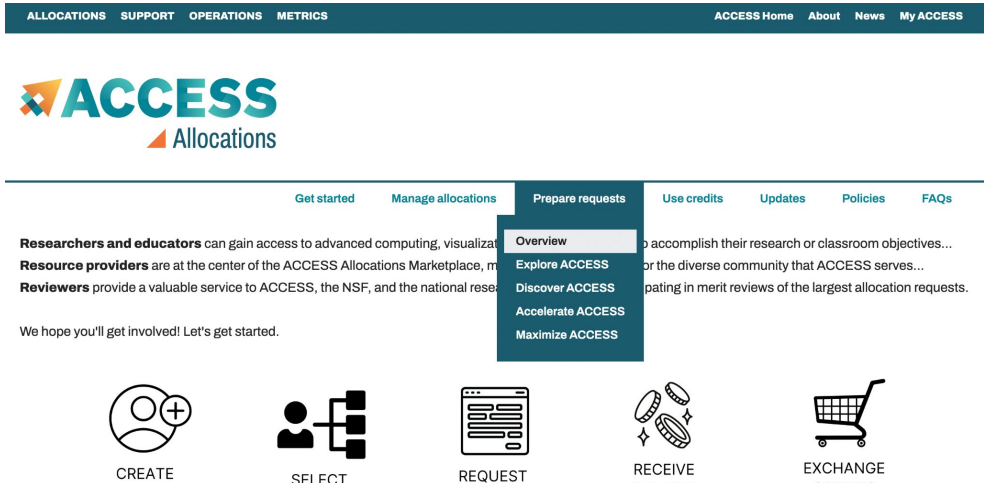
- ACCESS Services include Allocations, Support, Operation and Metrics, along with a Coordination Office

- Access website: https://access-ci.org/

National Supercomputer Resources: ACCESS

Four Allocation Opportunities to suit a variety of needs (credit thresholds):

- ***Explore (400,000)***
  - Best-suited for endeavors with light resource requirements
    - Grad students can be PIs
- ***Discover (1,500,000)***
  - Minimal effort to start production research activities
    - Potential best-fit for Campus Champion Allocations
- ***Accelerate (3,000,000)***
  - More substantial resource requirements
    - Multi-grand research, Gateways, etc.
- ***Maximize (No upper limit)***
  - For large-scale research project with extreme resource needs
    - Will largely resemble XRAC process



ACCESS Credits and Thresholds

Researchers have opportunities to request ACCESS allocations at four levels, which are described at the links in the table.

Allocation	Credit Threshold
Explore ACCESS	400,000
Discover ACCESS	1,500,000
Accelerate ACCESS	3,000,000
Maximize ACCESS	Not awarded in credits.

Allocation Eligibility

- Available to any research or educator as US academic, non-profit research, or educational institution.
- Can be in any official position including adjunct or instructional
- Postdoctoral researchers can be a PI of any project type
- Graduate students can lead an “Explore” ACCESS allocation under their advisor’s guidance
- NSF Graduate Fellows and Honorable mentions can apply for “Discover” allocations
- Ref: https://allocations.access-ci.org/access-allocations-policies#eligibility

Comparison Table

Comparison Table

Opportunity	Explore	Discover	Accelerate	Maximize
Purpose	Resource evaluation, grad student projects, small classes and training events, benchmarking, code development and porting, similar small-scale uses.	Grants with modest resource needs, Campus Champions, large classes and training events, NSF graduate fellowships, benchmarking and code testing at scale, gateway development.	Mid-scale resource needs, consolidating multi-grant programs, collaborative projects, preparation for Maximize ACCESS requests, gateways with growing communities.	Large-scale research projects.
Allocation credit threshold	Small	Medium	Large	No upper limit
Allocation duration	Supporting grant duration or 12 months	Supporting grant duration or 12 months	Supporting grant duration or 12 months	12 months
Requests accepted	Continuously	Continuously	Continuously	Every 6 months
	Multiple requests allowed	Multiple requests allowed	Multiple requests allowed	1 allowed (some exceptions)
Requirements and review process	Overview	1-page proposal	3-page proposal (max. length)	10-page proposal (max. length)
	Confirmation of eligibility and suitability of requested resources	Confirmation of eligibility and suitability of requested resources	Panel merit review	Panel merit review

Ref: https://allocations.access-ci.org/prepare-requests-overview

National Supercomputer Resources: ACCESS

ACCESS Allocations

Get started | Manage allocations | Prepare requests | Use credits | Upd

- Overview
- Submit a request
- Manage my projects
- Manage users
- Allocations Usage

Researchers and educators can gain access to advanced , and data resources to accomplish their resea
Resource providers are at the center of the ACCESS Alloc ng research possible for the diverse communit
Reviewers provide a valuable service to ACCESS, the NSF h community by participating in merit reviews o

We hope you'll get involved! Let's get started.

Maximize ACCESS – March 2023

Submissions open: 2022-12-15 – 2023-01-15

For projects with resource needs beyond those provided by an Accelerate ACCESS project, a Maximize ACCESS request is required. ACCESS d an upper limit on the size of allocations that can be requested or awarded at this level, but resource providers may have limits on allocation amoun resources.

SUBMIT A MAXIMIZE ACCESS – MARCH 2023 REQUEST

ACCESS Allocations

Get started | Manage allocations | Prepare requests | Use credits | Updates | Policies | FAQs

Available Opportunities

Here are the open opportunities for which you may request an allocation. Find the opportunity that aligns with your best estimate of your resource needs. Don't worry about starting too small. As you clarify your needs, you can upgrade to a larger-scale opportunity when you're ready.

Explore ACCESS

Explore ACCESS allocations are intended for purposes that require small resource amounts. Researchers can try out resources or run benchmarks, instructors can provide access for small-scale classroom activities, research software engineers can develop or port codes, and so on. Graduate students can conduct thesis or dissertation work.

SUBMIT AN EXPLORE ACCESS REQUEST

Discover ACCESS

Discover ACCESS projects are intended to fill the needs of many modest-scale research activities or other resource needs. The goal of this opportunity is to allow many researchers to request allocations with a minimum amount of effort so they can complete their work. To submit a request, you will need to submit a one-page description of the project to address the review criteria. You can also ask for an advisory review from the community to guide you to appropriate resources.

SUBMIT A DISCOVER ACCESS REQUEST

Accelerate ACCESS

Accelerate ACCESS projects support activities that require more substantial resource amounts to pursue their research objectives. Researchers are expected to have reasonably well defined plans for their resource use and to submit a 3-page project description for merit review. Reviewers will look more closely at how your resource usage plan addresses the review criteria.

SUBMIT AN ACCELERATE ACCESS REQUEST

Resource Providers (PRs)

- ACCESS consists of a set of Resource Providers (PRs) that offer a wide range of computational resources including systems such as high-performance computing (HPC) clusters, virtualization (cloud-style) clusters, high throughput computing (HTC) clusters, massive storage clusters, large memory clusters, and composable clusters.

- ACES (Texas A&M)
- Anvil (Purdue)
- Bridges-2 (PSC)
- DARWIN (Delaware)
- Delta (NCSA)
- Expanse (SDSC)
- FASTER (Texas A&M)
- Jetstream2 (IU)
- OOKAMI (Stonybrook)
- KyRIC (Kentucky)
- Rockfish (JHU)
- Stampede-2 (TACC)
- RANCH (TACC)
- Open Science Grid (OSG)
- Open Storage Network (OSN)

National Supercomputer Resources: ACCESS

ACCESS Allocations

Purdue Anvil CPU

Purdue Anvil GPU

SDSC Expanse CPU

SDSC Expanse GPU

Resource Type:	Compute
Resource Description:	Expanse GPU will be a Dell integrated cluster, NVIDIA V100 GPUs with NVLINK, interconnected with Mellanox HDR InfiniBand in a hybrid fat-tree topology. There are a total of 52 nodes with four V100 SMX2 GPUs per node (with NVLINK connectivity). There are two 20-core Xeon 6248 CPUs per node. Full bisection bandwidth will be available at rack level (52 CPU nodes, 4 GPU nodes) with HDR100 connectivity to each node. HDR200 switches are used at the rack level and there will be 3:1 oversubscription cross-rack. In addition, Expanse also has four 2 TB large memory nodes. The system will also feature 12PB of Lustre based performance storage (140GB/s aggregate), and 7PB of Ceph based object storage.
Recommended Use:	GPUs are a specialized resource that performs well for certain classes of algorithms and applications. Recommend to be used for accelerating simulation codes optimized to take advantage of GPUs (using CUDA, OpenACC). There is a large and growing base of community codes that have been optimized for GPUs including those in molecular dynamics, and machine learning. GPU-enabled applications on Expanse will include: AMBER, Gromacs, BEAST, OpenMM, NAMD, TensorFlow, and PyTorch.
Organization:	San Diego Supercomputer Center
Units:	GPU Hours
Description:	

SDSC Expanse Projects Storage

allocations | Prepare requests | Use credits | Updates | Policies | FAQs

- Overview
- Available resources
- Exchange calculator

g, visualization, and data res … r research or classroom objectives…
ketplace, making research p … mmunity that ACCESS serves…
tional research community … views of the largest allocation requests.

REQUEST ALLOCATION

Exchange Calculator

Number of units on this resource:

10,000	ACCESS Credits

Equals this many units on this resource:

186	SDSC Expanse GPU

RESET

National Supercomputer Resources: ACCESS

ACCESS Home | About | News | My ACCESS

- My Allocations
- My Engagements
- Edit Profile
- Logout

List of ACCESS Allocations Requests

Please click the View Actions link to see actions on each of your requests. You can use the Choose New Action arrow menu to add new actions to the request.

Discover ACCESS_TRA220034_Chi

Manage Users

Active from 2022-11-16 to 2023-11-15

Choose New Action

Type: New | Status: Approved | Submitted: 2022-10-12

- Supplement
- Transfer

Available Resources

For a transfer, please indicate the resource you are **transferring from** with **negative number** (e.g., -1,000), and the resource you are **transferring to** with a **positive number** (e.g., 1,000).

To request a resource, select it and enter an amount. Comments are optional.

Exchange Calculator

ACCESS Credits

-46,080.00 ACCESS Credits

Comments

Transfer to
64 Cores f

SDSC Expanse CPU

Expanse will be a Dell integrated compute cluster, with AMD Rome processors, interconnected with Mellanox HDR InfiniBand in a hybrid fat-tree topology. There are 728 compute nodes, each with two 64-core AMD EPYC 7742 (Rome) processors for a total of 93,184 cores. They will feature 1TB of NVMe storage and 256GB of DRAM per node. Full bisection bandwidth will be available at rack level (56 nodes) with HDR100 connectivity to each node. HDR200 switches are used at the rack level and there will be 3:1 oversubscription cross-rack. In addition, Expanse also has four 2 TB large memory nodes. The system will also feature 12PB of Lustre based performance storage (140GB/s aggregate), and 7PB of Ceph based object storage.

- SDSC Expanse Projects Storage is required if requesting this resource.

46,080.00 Core-hours

Comments

request 64 cores for running 30 days

SDSC Expanse Projects Storage

Allocated storage for projects using Expanse Compute and Expanse GPU resources.

- SDSC Expanse CPU is required if requesting this resource.
- SDSC Expanse GPU is required if requesting this resource.

10.00 GB

Comments

SDSC HPC for UC

- Request HPC@UC at https://www.sdsc.edu/support/hpc_uc_apply-exp.html
- Up to 500K core-hours of computing, associated data storage, and access to SDSC expertise to assist their research team.
- Awards are active for one year. NO supplements, renewals or Extensions
- Applicants must not have an active ACCESS award
- Developed to support onboarding to ACCESS and large, formal allocation requests
- SDSC staff will assist in developing these allocation applications
- Applications are reviewed on an ongoing basis. Applicants will be notified within 10 business days of the review decision.
- Ref: https://www.sdsc.edu/support/hpc_uc_apply-exp.html

Cloud Computing: Indiana JstStream2

Indiana Jetstream2

Resource Type:	Compute
Resource Description:	Jetstream2 is a user-friendly cloud environment designed to give researchers and students access to computing and data analysis resources on demand as well as for gateway and other infrastructure projects. Jetstream2 is a hybrid-cloud platform that provides flexible, on-demand, programmable cyberinfrastructure tools ranging from interactive virtual machine services to a variety of infrastructure and orchestration services for research and education. The primary resource is a standard CPU resource consisting of AMD Milan 7713 CPUs with 128 cores per node and 512gb RAM per node connected by 100gbps ethernet to the spine.
Recommended Use:	For the researcher needing virtual machine services on demand as well as for software creators and researchers needing to create their own customized virtual machine environments. Additional use cases are for research-supporting infrastructure services that need to be "always on" as well as science gateway services and for education support, providing virtual machines for students.
Organization:	Indiana University
Units:	SUs
Description:	1 SU = 1 Jetstream2 vCPU-hour. VM sizes and cost per hour are available https://docs.jetstream-cloud.org/general/vmsizes/

Indiana Jetstream2 GPU

Indiana Jetstream2 Large Memory

Indiana Jetstream2 Storage

RP: Indiana JstStream2

Jetstream2 is a user-friendly cloud computing environment for researchers and educators running on OpenStack and featuring Exosphere as the primary user interface. It is built on the successes of Jetstream1 and continues the main features of that system while extending to a broader range of hardware and services, including GPUs, large memory nodes, virtual clustering, programmable cyberinfrastructure with OpenStack Heat and Terraform, and many other features. It is designed to provide both infrastructure for gateways and other "always on" services as well as giving researchers access to interactive computing and data analysis resources on demand.

For a more in-depth description please see the System Overview.

Jetstream2 Status

Overall JS2 system status	Operational ●

Please visit https://jetstream.status.io/ for detailed system status information and planned maintenance announcements. Also see, Jetstream2 system status and information for additional information on our outages and maintenance mailing list and community chat.

Accessing Jetstream2

Access to Jetstream2 is available solely through Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) allocations. You must be on a valid allocation or the PI of a valid allocation to have access to Jetstream2.

Ref: https://docs.jetstream-cloud.org/

RP: Indiana JstStream2

Choose an Instance Source

By Type | By Image

Ubuntu

- Wide compatibility with community software packages
- Good choice for new users

22.04 (latest)

20.04

Red Hat-like

- Based on Red Hat Enterprise Linux (RHEL)
- Compatible with RPM-based software

Rocky Linux 9

Rocky Linux 8

AlmaLinux 9

AlmaLinux 8

CentOS 7

Instances

Instances used	Cores used	RAM used
3 of 25 total	32 of 63 total	122.9 of 224.5 GB

Filters: Created by me (jaychi@access-ci.org) × + Clear filters

2 instances filtered from 3 total

Jay_test_CPU_m3l ●

m3.large · created 2 minutes ago by jaychi@access-ci.org

Connect to

149.165.170.244

Jay_test_GPU_g3m ●

g3.medium · created 24 minutes ago by jaychi@access-ci.org

Connect to

149.165.154.2

RP: Indiana JstStream2

Instance Jay_test_CPU_m3l

Ready | Actions

Info 42a18f7f-dc45-4022-bea0-da1629adab40

created 4 minutes ago | by user jaychi@access-ci.org | from image Featured-Ubuntu22 | flavor m3.large | Burn rate 16.00 SUs/hour

Resource Usage

CPU — of 16 total cores

RAM — of 60 total GB

Root Dis

100% 75% 50% 25% 0%

1:16 1:26 1:36 1:46

Volumes

Volumes used 1 of 10 total | Storage used 1,000 GB of 5 TB

Filters: Created by me × + Clear filters — volume: 1

Jay_attach_HD_1 — Attached to Jay_test_CPU_m3l

1,000 GB · created 20 hours ago by me — Detach

Interactions

- Web Shell
- Web Desktop
- Native SSH : exouser@149.165.170.244
- Console

Credentials

Hostname	jay-test-cpu-m3l.tra220034.projects.jetstream-cloud.org.
Public IP Address	149.165.170.244 Unassign
IP Details	
Username	exouser
Passphrase	Show
SSH Public Key Name	jaychi_key

Volumes

Name	Device	Mount point
Jay_attach_HD_1	/dev/sdb	/media/volume/sdb

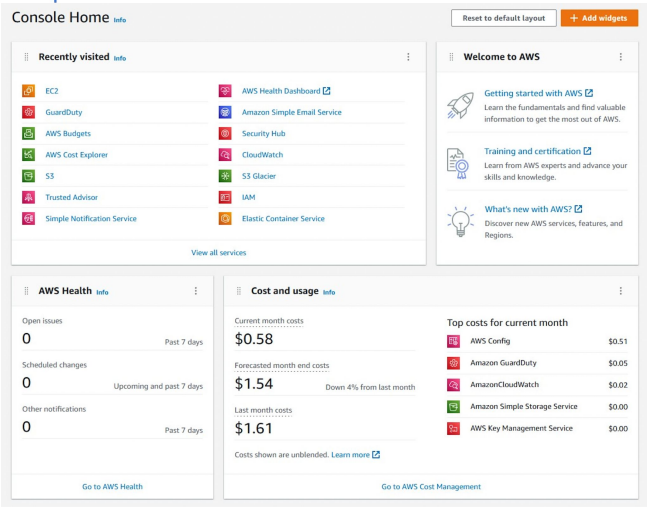
Attach volume

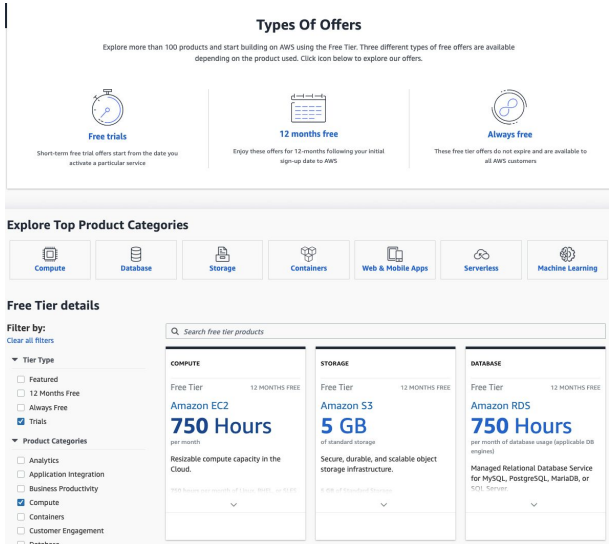
Cloud Computing: Amazon Web Services (AWS)

- If you choose to use AWS, it is recommended to take advantage of the Campus Cloud Landing Zone (LZ) for AWS. A UCSB purchases order is required to request an Campus Cloud account (https://ucsb.github.io/campus-cloud-docs/getting-started/#procurement).

Campus Single Sign On for AWS: https://aws.cloud.ucsb.edu

- Supported Campus Cloud Regions:
  - **US-West-2 (Oregon)**
  - **US-East-1 (N.Virginia)**
- We recommend starting in *US-West-2*





Important: You may need the help of a PI or Department Purchasing person to create a Purchase Order which is necessary to request an account in the Campus Cloud.

Amazon Elastic Compute Cloud (Amazon EC2)

- Use Case:
  - Run cloud-native and enterprise applications
  - Scale for HPC applications
  - Train and deploy ML applications
- EC2 Instance Types
  - General Purpose
  - Compute Optimized
  - Memory Optimization
  - Accelerated Computing
  - Storage Optimized
- More Information
  - Amazon EC2: https://aws.amazon.com/ec2/
  - Amazon EC2 Pricing Estimation: https://aws.amazon.com/ec2/pricing/on-demand/

https://instances.vantage.sh/

On-Demand Plans for Amazon EC2

Select a location type and region

Location Type: AWS Region

Region: US West (Oregon)

Select an operating system, instance type, and vCPU to view rates

Operating system: Linux

Instance type: Compute Optimized

vCPU: 36

Viewing 4 of 525 available instances

Instance name	On-Demand hourly rate	vCPU	Memory	Storage	Network performance
c5.9xlarge	$1.53	36	72 GiB	EBS Only	10 Gigabit
c5d.9xlarge	$1.728	36	72 GiB	1 x 900 NVMe SSD	10 Gigabit
c5n.9xlarge	$1.944	36	96 GiB	EBS Only	50 Gigabit
c4.8xlarge	$1.591	36	60 GiB	EBS Only	10 Gigabit

Ronin Platform RONIN

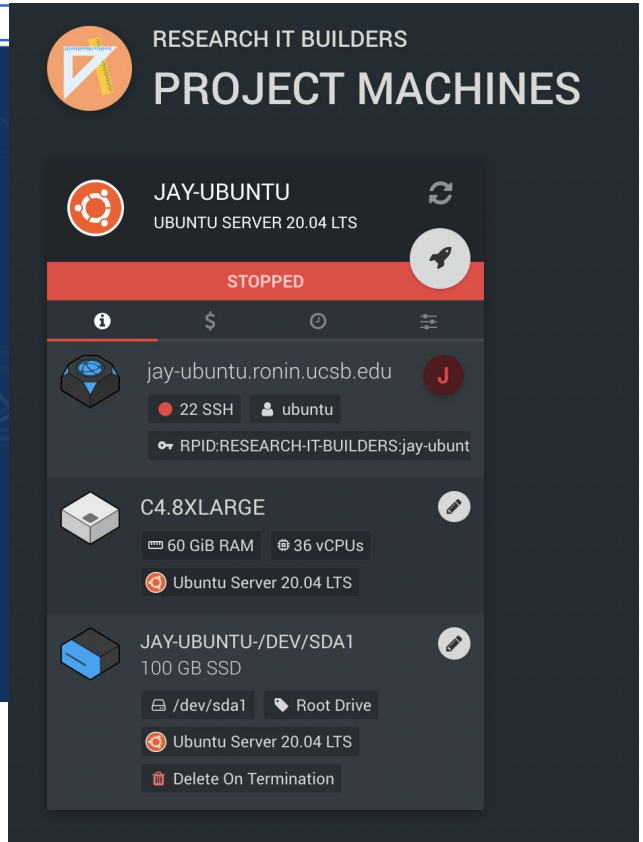
ronin.ucsb.edu/login.php

View site information

- If you like to use AWS to be your cloud computing platform, RONIN removes the enormous complexity of AWS offerings and provides an easy-to-use self-service platform.
- UCSB provides RONIN information support if you like to use AWS to do your computing research via the RONIN platform.

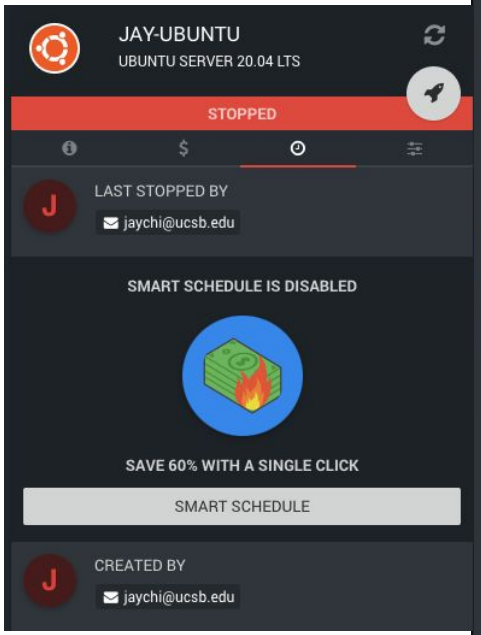
UC SANTA BARBARA

LET'S GO!



Contact with Bill Doering: billd@ucsb.edu

Ronin Platform: Control Your AWS Cost



Ack!

- Acknowledgements - https://csc.cnsi.ucsb.edu/publications

Please acknowledge the CSC in publications and presentations if you are using our facility's computational resources (including staff involvement) in your research.

"We acknowledge support from the Center for Scientific Computing from the CNSI, MRL: an NSF MRSEC (DMR-2308708) and NSF CNS- 1725797."

For users of GPU nodes, please add the grant number NSF OAC-1925717

Questions and Thought

- What else content should we cover?
- Other ideas for a workshop?
  - Running Parallel Python / Matlab / R on the Cluster, Mathematica, Singularity/Docker Container, etc.

- More Information:

  https://csc.cnsi.ucsb.edu/