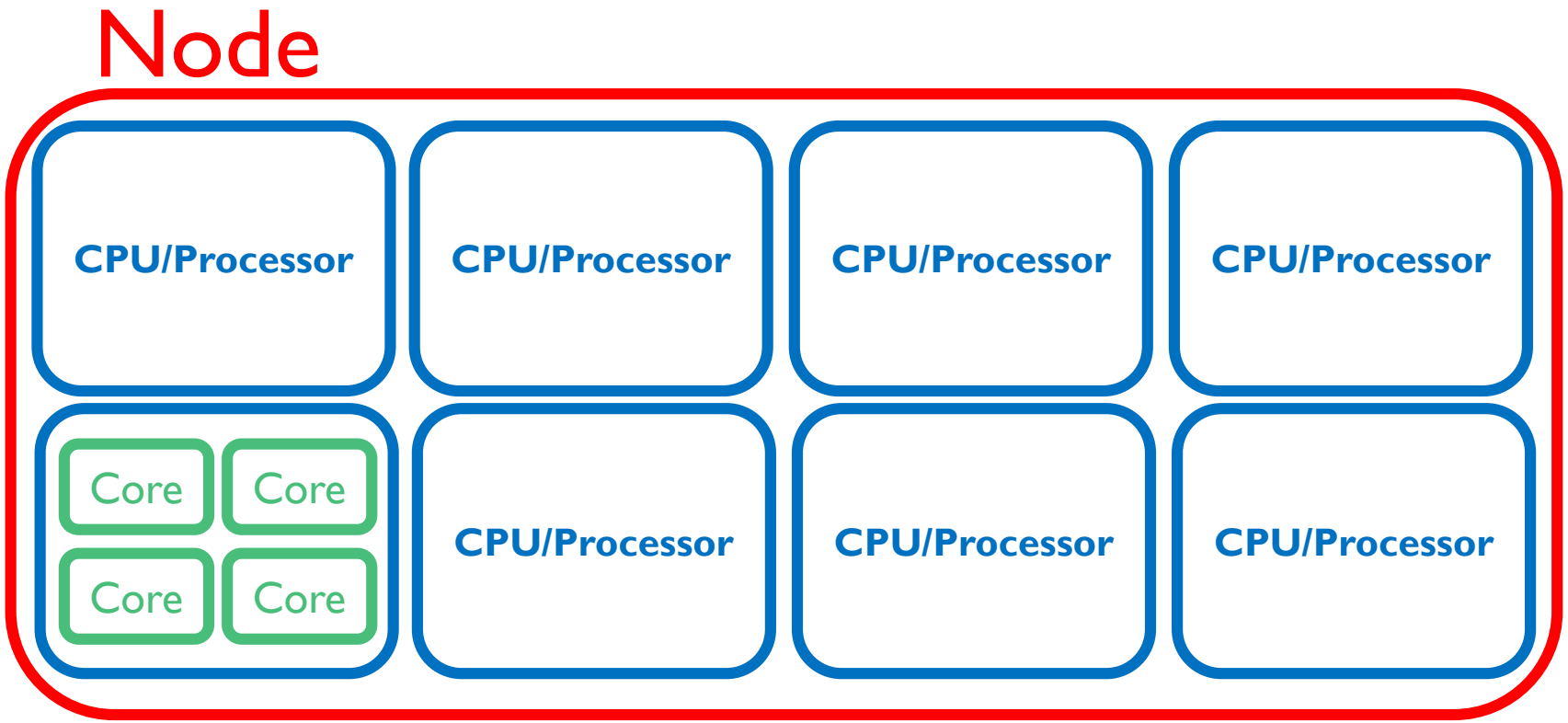
HIGH PERFORMANCE COMPUTING CLUSTER TUTORIAL FOR THE KISHIDA LAB

SANJEEV NAMJOSHI | AUGUST 20TH, 2018

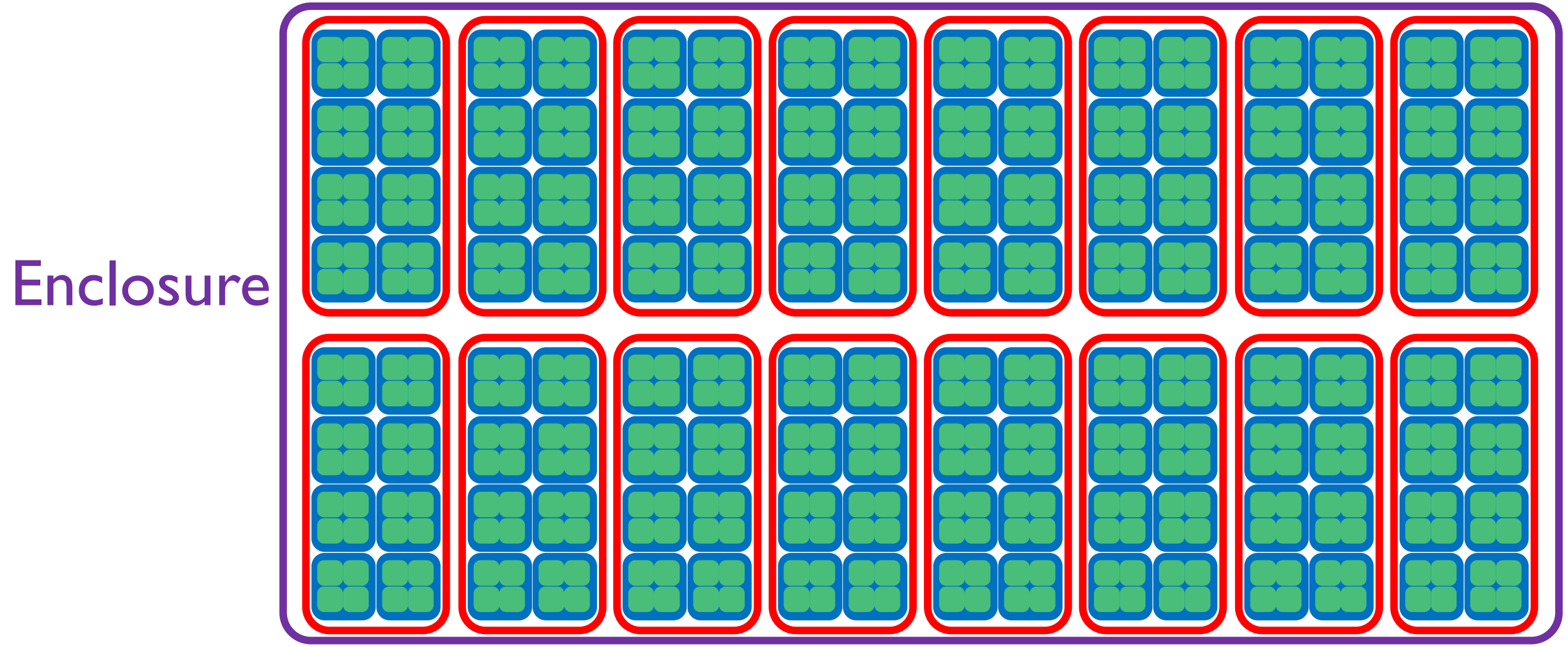
OUTLINE

1. Intro to high performance computing
2. The Wake Forest Medical HPCC
3. Using and accessing the HPCC
4. SLURM
5. Kishida Lab job submission interface
6. Demonstration
7. Help and other resources

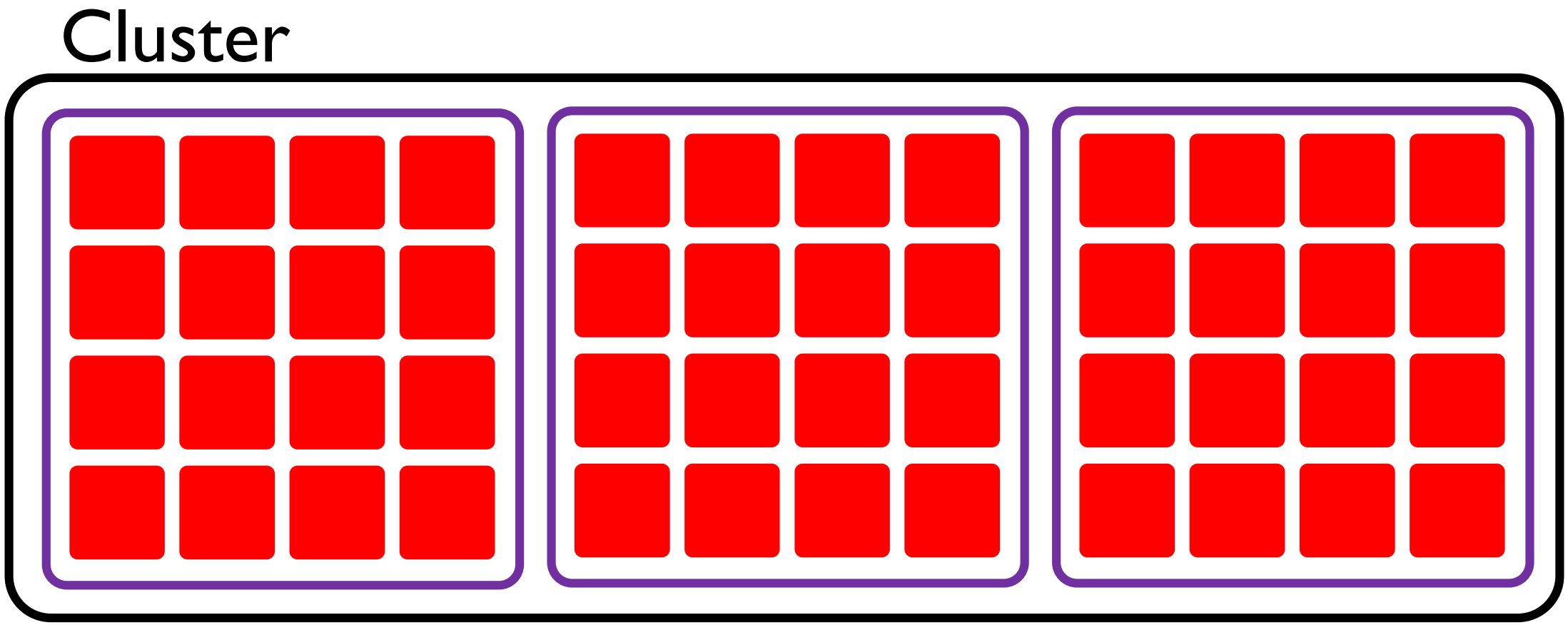
HPCC DESIGN AND LAYOUT (NODES)



HPCC DESIGN AND LAYOUT (ENCLOSURES)



HPCC DESIGN AND LAYOUT (CLUSTER)



TYPES OF NODES

- Login/head nodes
- Compute nodes
- GPU nodes

WAKE FOREST MEDICAL HPCC HARDWARE (DEMON) LAYOUT

- 50 nodes
  - 46 compute, 2 login, 2 GPU
- Distributed memory
  - 128 GB/compute node, 5 GB/GPU node
  - Total of 6.4 TB (compute) and 20 GB (GPU)
- Shared network file system
- Local disk with 300 GB (SATA) and 1.5 PB aggregate
- Node interconnectivity (Infiniband Mellanox Switches) with data transfer rate of 56 Gb/s
- OS: Linux Red hat 6.4
- Management system: The Simply Linux Utility for Resource Management (SLURM)

DEMON NODES DETAIL

- Head nodes:
  - Administrative nodes to manage updating, monitoring, and SLURM
- Login (test) nodes:
  - Accessed by user to interface with HPCC
- Compute nodes:
  - 48 compute nodes: 16 Dell M620 nodes inside 3 Dell M1000e enclosures (16 nodes/enclosure)
  - Each node has two Intel E5-2670 32-core Sandy Bridge EP processors with 128 GB DDR3 memory
- GPU nodes:
  - 2 Intel CPU E5-2670 32-core Sandy Bridge EP processors with 128 GB DDR3 memory
  - 2 NVIDIA K20m GPU 2496-CUDA-core (Tesla) accelerators with 5 GB GDDR5 memory per GPU

USING AND ACCESSING THE HPC

- MacOS/Linux users:
  - Easy. Once you have your username approved, just launch the terminal to get started
- Windows users:
  - Difficult. Use an SSH client on windows, dual-boot to Linux, or run Linux/MacOS in a virtual environment

SETTING UP A LINUX VIRTUAL MACHINE

- Run Linux as a separate operating system within Windows (or MacOS)
- Use VirtualBox as your virtual machine
- Download your Linux operating system ISO image (I suggest Ubuntu)
- Install using VirtualBox

BASIC TERMINAL COMMANDS AND TERMINOLOGY (1)

- The distinction between command line, terminal, shell, console, are not important for our purposes. They are all similar and a means to interact with the operating system.
- The shell that is usually installed on Linux based distributions (or MacOS) is called Bash (the **B**ourne **A**gain **Sh**ell). Other shells are available that provide the same functionality with extra features.

BASIC TERMINAL COMMANDS AND TERMINOLOGY (2)

- When you launch the terminal you will usually be in your home directory by default. This directory is represented by the symbol: ~
- You will type commands into the command line to move between different directories and perform actions on the files there
- Not all users can access every directory – some permissions are revoked. The root user can access everything.
- When you type a command, you can often add extra arguments afterwards to modify the behavior of the command.
  - Example: `ls` lists files but not hidden files. `ls -a` lists all files including hidden files.

BASIC TERMINAL COMMANDS AND TERMINOLOGY (3)

- Commonly used commands:
  - `cd`: used to navigate to new directories
  - `ls`: list all files in the current directory
  - `mkdir`: create a new directory
  - `cp`: copy a directory
  - `mv`: move a directory
  - `rm`: remove a file
  - `rm -rf`: remove a directory and all subdirectories. Be careful with this command.
  - `nano`: simple text editor for the command line
  - `man`: see the documental (manual) for a command
  - `touch`: create a file in a directory. Touch has many other functions as well.

ACCESSING THE HPCC AND FIRST TIME SETUP

- Login to the login node with your wakehealth.edu AD account:
  - demonln1.hpcc.wfubmc.edu
  - demonln2.hpcc.wfubmc.edu
- Create and source your `.bash_profile`
- Login node etiquette

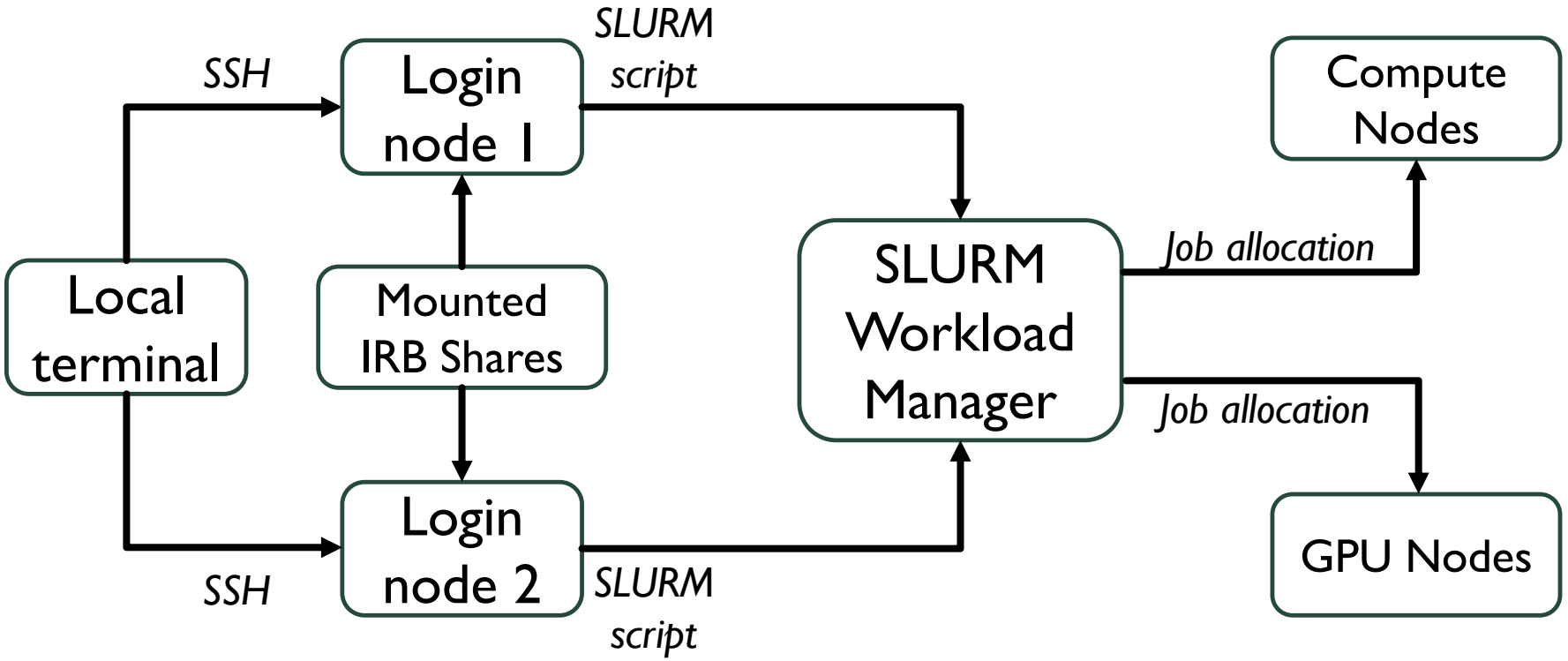
R/MATLAB PACKAGE MANAGEMENT

- R package management
  - Load the R module and any other needed modules
  - Run R
  - Check to ensure your package is not already installed
  - Install your package
- MATLAB package management
  - Requires licenses for Mathworks toolboxes
  - Other licensed toolboxes need system administrator approval to add
  - 3rd party scripts can be placed in the same directory where you run your analysis

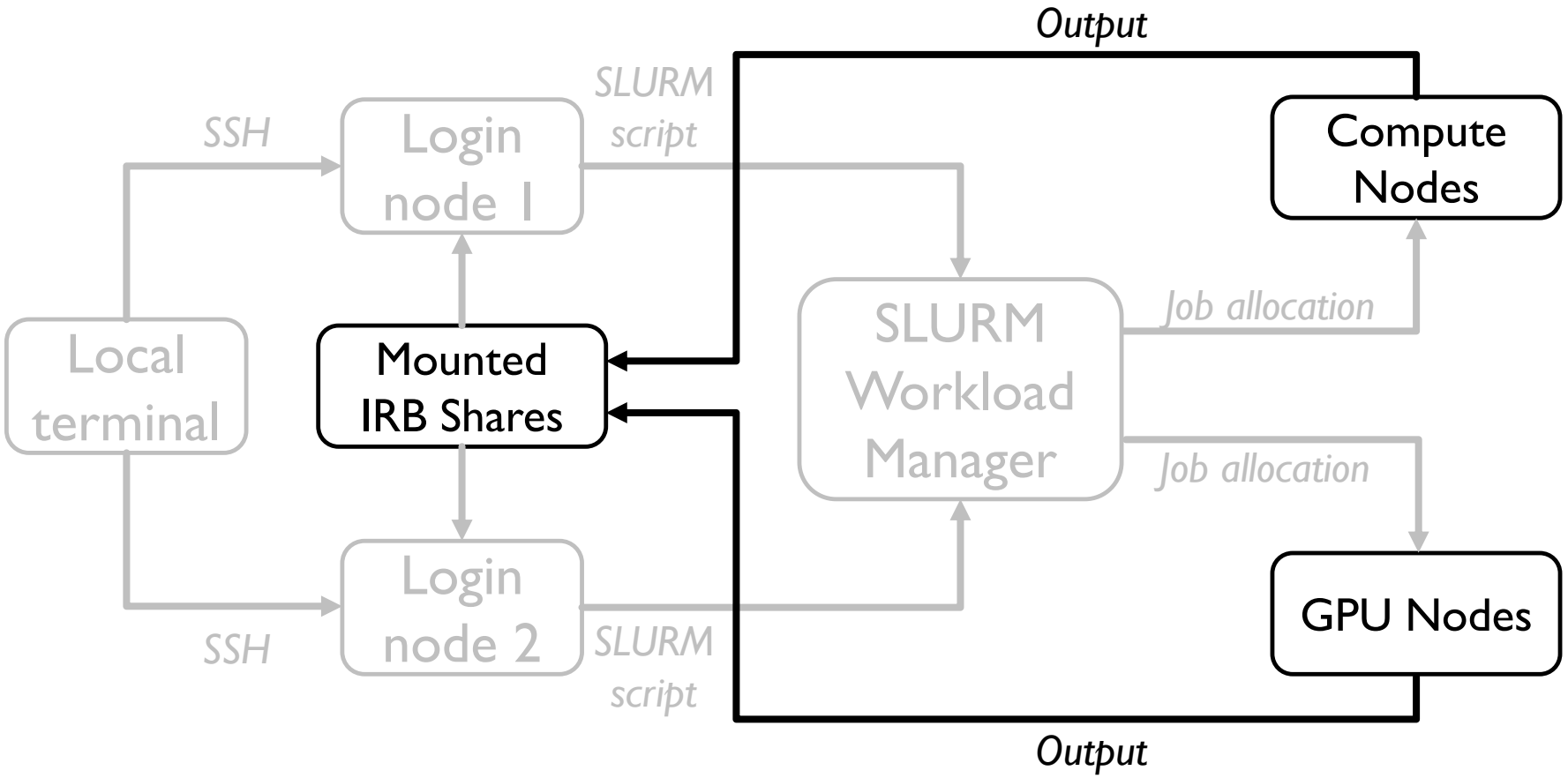
TERMINAL OVERVIEW, SETUP, AND PACKAGE MANAGEMENT

- Demonstration in Linux (VirtualBox with Windows host)
  - Overview of terminology
  - Basic overview of terminal commands
  - Accessing the HPCC through a secure shell (SSH)
  - Bash profile setup
  - R package management
  - Checking for installed packages

OVERALL WORKFLOW



OVERALL WORKFLOW



SLURM OVERVIEW

- Interface to send information to compute nodes
- Manages requests from all incoming users accessing the HPCC
- Allocates resources

SLURM TERMINOLOGY

- Jobs: Submitted by the user indicating what SLURM is supposed to do
- Job steps: One or more steps involved in executing the job. Some jobs may only have one step.
- Tasks: Instances of a program to be run, specified by the user
  - Each task may uses or more CPUs
  - Tasks and cores are sometimes used synonymously in documentation/help online

COMMON SLURM COMMANDS

- `scancel [job_id]`: Cancels the specified job. There are many other options such as cancelling all jobs for a specific user.
- `sinfo`: Get information about current nodes and partitions available and general system information
  - Useful to see if nodes are down or other issues
- `squeue`: Shows all jobs currently running in the queue. There are three separate queues:
  - `defq`: Computational nodes (46x)
  - `testq`: System login nodes (2x)
  - `gpu`: GP computing nodes (2x)
- `srun`: Initiate and run a parallel job on the cluster
- `sbatch`: Run a SLURM batch script (more on this later)

BUILDING A SLURM SCRIPT

- Contains all relevant information for your run:
  - Specifications of the job run with `sbatch` (more on this later)
  - Output directory
  - Absolute paths to directories with analysis files
  - Absolute paths to directories with R or Matlab script files
- Created automatically by the Kishida Lab job submission interface

IMPORTANT SBATCH OPTIONS

- `-a, --array`: Used to submit a job array.
- `-c, --cpus-per-task`: Specifies the number of CPUs to be used per each task that was requested
- `-J, job-name`: Name of the job
- `--mem-per-cpu`: The minimum amount of memory requested for each allocated CPU
- `-N, --nodes`: The minimum nodes to be allocated for the job.
- `-n, --ntasks`: Specifies the number of tasks (processes) to run and to allocate enough resources for these tasks.
- `--ntasks-per-core`: Specifies the maximum number of `--ntasks` that would be run on each core.
- `--ntasks-per-node, --tasks-per-node`: Requests that `--ntasks` be run on each allocated node.
- `-t, –time`: A limit on the total run time of the job. This is specified in the format 00:00:00 indicating hours, minutes, and seconds.

GENERAL JOB PARAMETER GUIDELINES

- Do not unnecessarily request a high amount or resource allocation
- When you run `--ntasks` it is indicating the numbers of tasks spread out over the nodes. If you run --ntask=16 --nodes=1 you will run 16 tasks per node. If you run `--ntasks=16 --nodes=2` then will run 8 tasks per node (16/2).
- A good general strategy is to start on the low end and increase the allocated resources when you have a better sense of what you need.
- Test your needs over one folder and add resources as necessary.

OTHER CONSIDERATIONS

- If you are submitting many jobs at once through a single shell script, pause 0.5 to 1 second between each `sbatch` submit
- Ensure each job takes at least 5-10 minutes to run so the scheduler can spend the right amount of time to set up, run, and break down scheduled jobs.

COMMON SLURM COMMANDS

- Demonstration in Linux (VirtualBox with Windows host)
  - Investigating common SLURM commands at the login node
  - Looking at a sample SLURM batch script

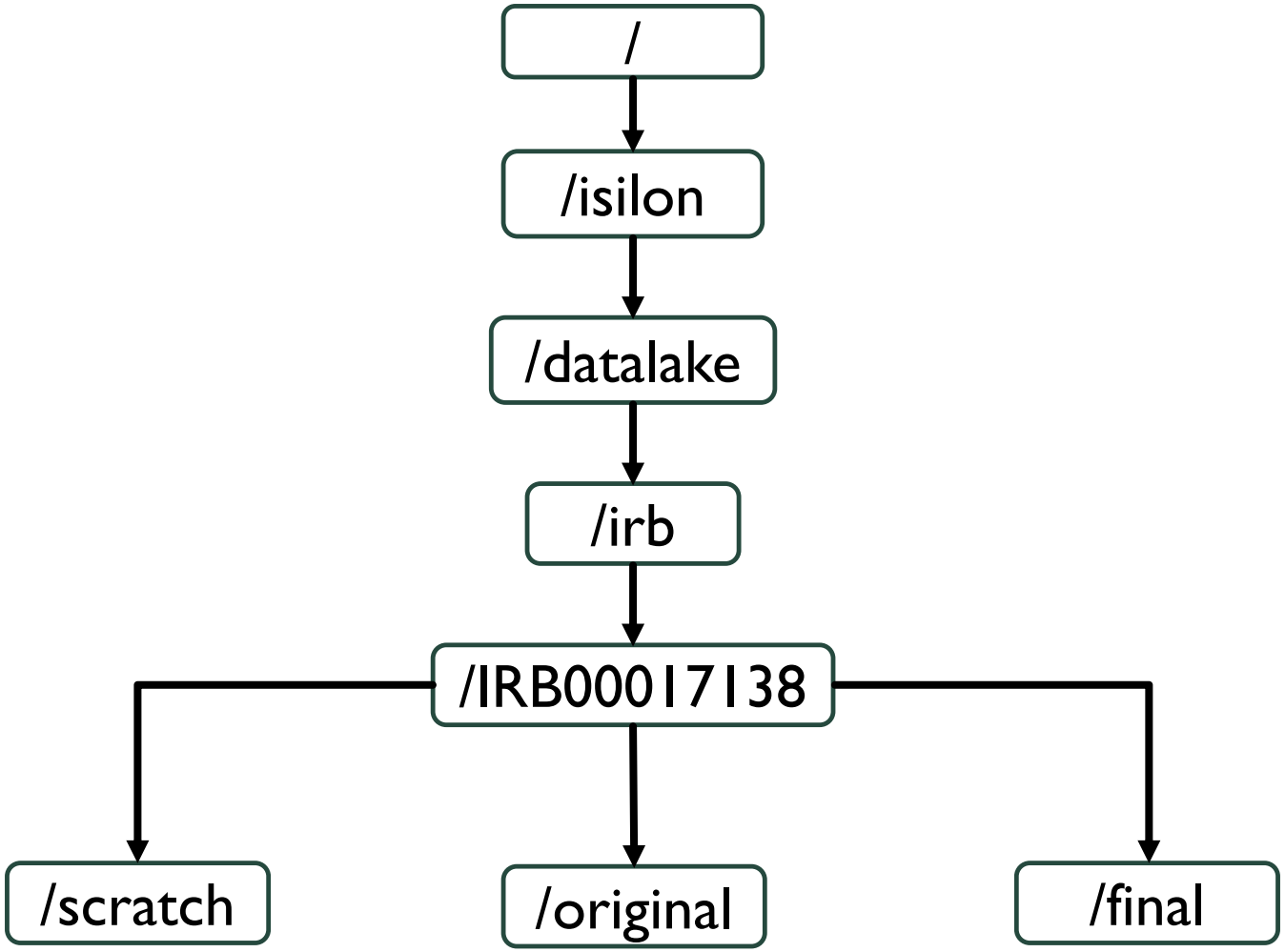
PARALLEL V. SERIAL COMPUTING

- Serial jobs (non-parallel nodes, non-parallel CPUs)
  - Default and simplest way to run a job
  - SLURM processes the job on one core at a time (serially) on the first node assigned to the job
- Multithreaded jobs on a single node (non-parallel nodes, parallel CPUs)
  - The job is distributed over multiple CPUs within a single node
  - Job arrays can do parallel multithreaded by sending different jobs to different nodes
  - Otherwise, code is split into chunks using a parallel processing framework (OpenMP)
- MPI jobs (parallel nodes, parallel CPUs)
  - Uses the message passage interface (MPI) to parallelize jobs
  - Most efficient way to run large jobs but requires heavy modification of user's scripts

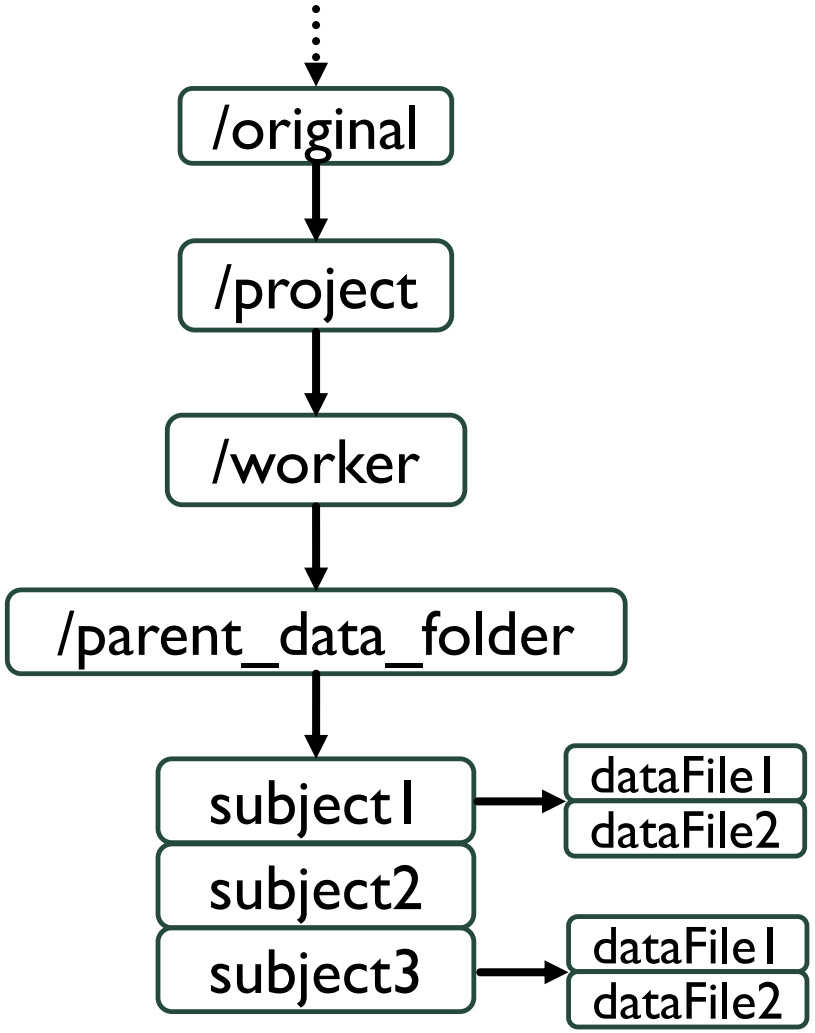
KISHIDA LAB JOB SUBMISSION INTERFACE (JSI)

- Manages all of the directory creation, file moving, and SLURM script generation
- Builds SLURM script based on user's input preferences and files
- Submits SLURM script using `sbatch`
- Error checking to ensure everything is in the right location
- Email alerts
- Archival function
- Flexible so users can run in default mode (JSI generates SLURM script) or users can submit their own modified scripts.

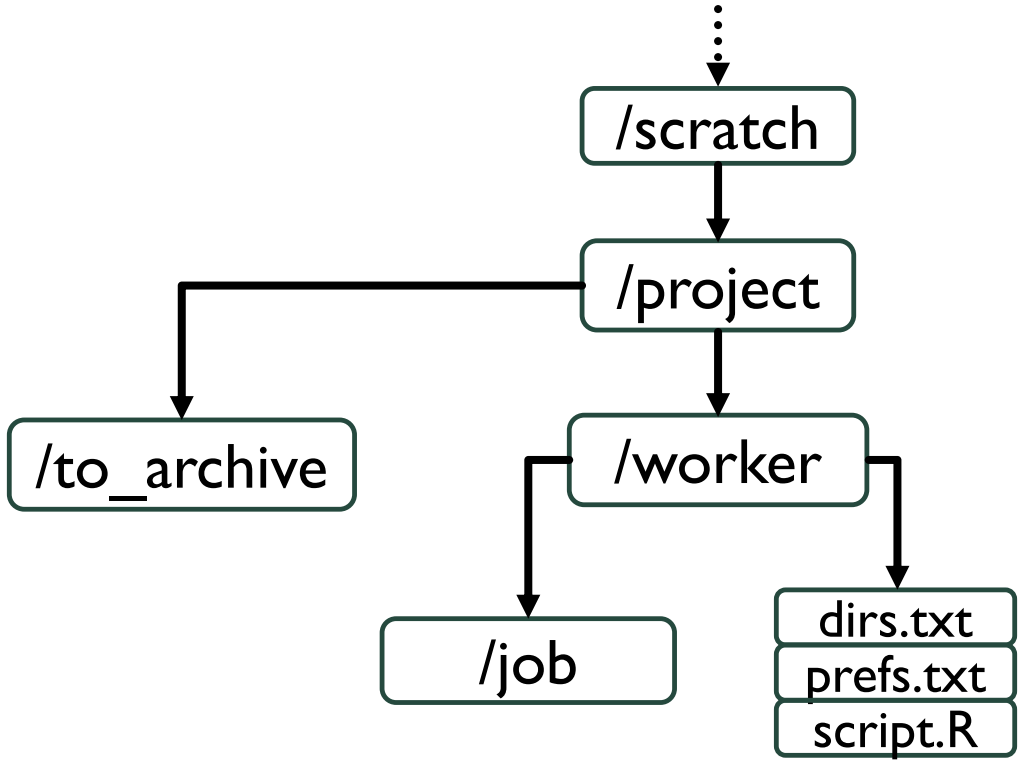
IRB DIRECTORY STRUCTURE FOR JSI



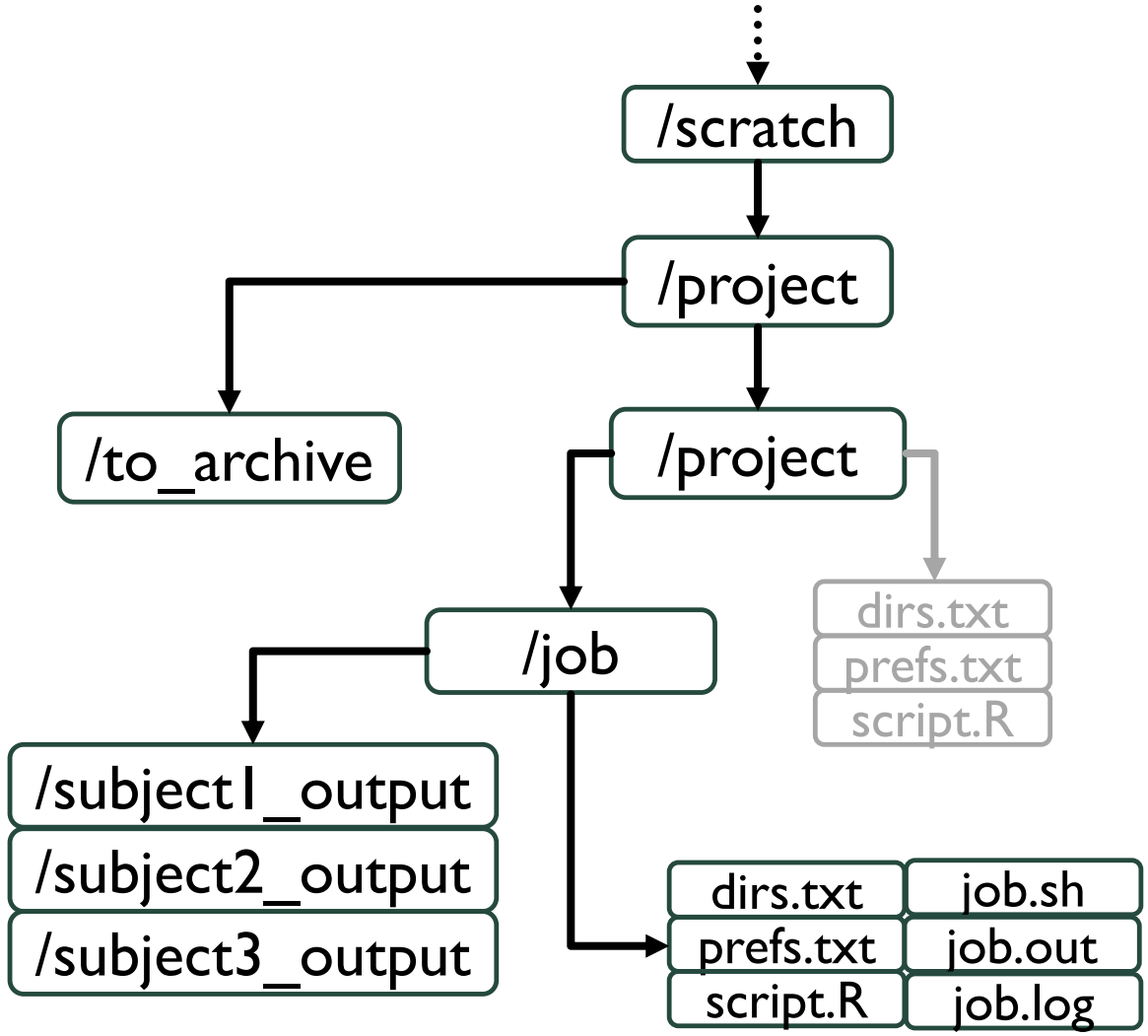
/ORIGINAL


/original
/project
/worker
/parent_data_folder
subject1
subject2
subject3
dataFile1
dataFile2
dataFile1
dataFile2

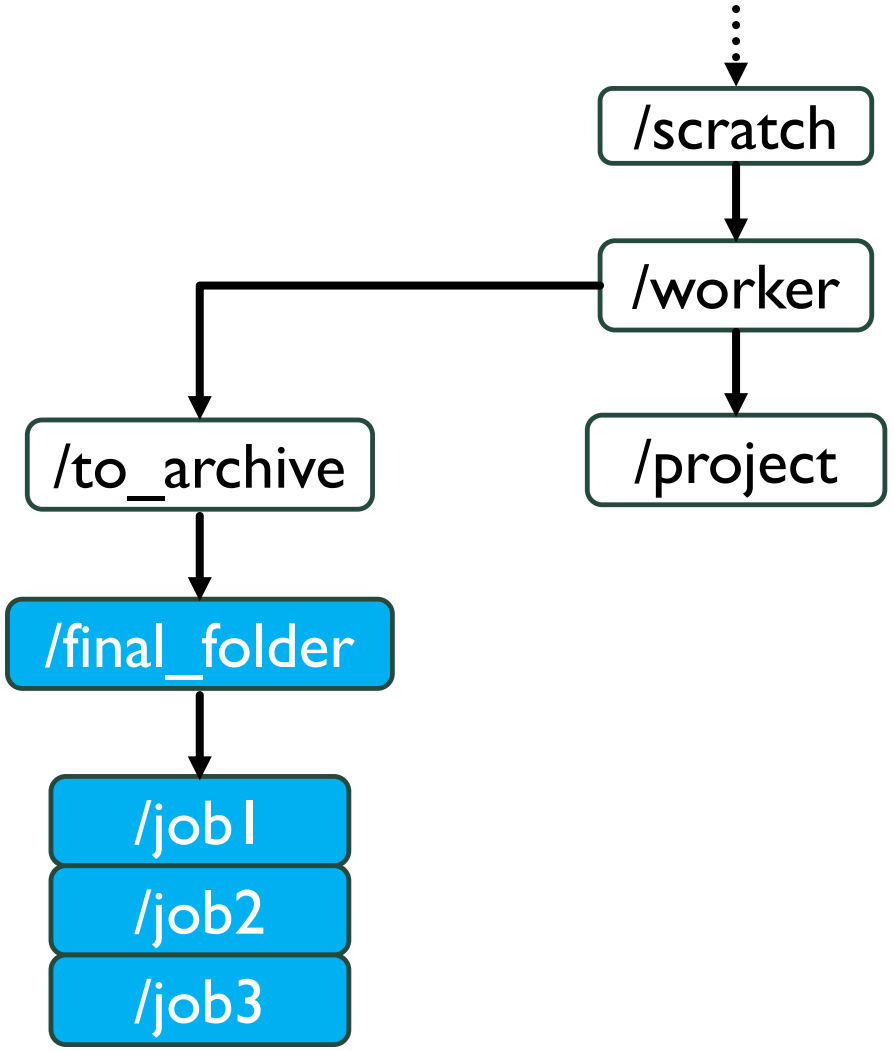
/SCRATCH (PRE-RUN)



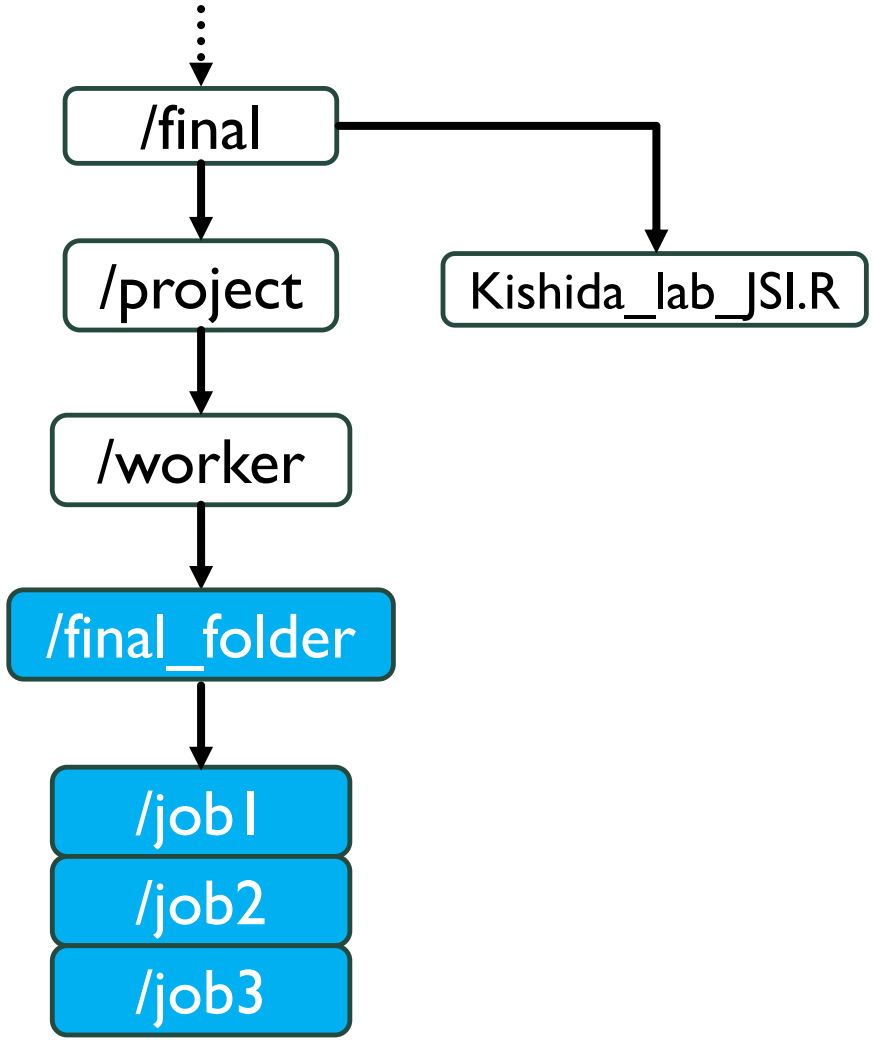
/SCRATCH (POST-RUN)



/SCRATCH (ARCHIVE)



/FINAL


/final
/project
Kishida_lab_JSI.R
/worker
/final_folder
/job1
/job2
/job3

SETUP FOR RUNNING THE JSI (INITIAL SETUP)

1. Modules must be loaded before running the JSI
2. Data folders must be in `/original/project/worker`
3. Must run R and source the JSI in the IRB root directory
4. Script file, prefs file, and dir file must be in `/scratch/project/worker`
5. dir file must indicate all directories over which your script will be run

SETUP FOR RUNNING THE JSI (PREFS AND DIRS FILES)

- dirs.txt setup
  - Directories must go on separate lines and paths must be formed relative to the `/original` directory
  - For example:
    - `/original/VoltammetryModeling/sanjeev/LOOCV_analysis_1/electrode_1`
    - `/original/VoltammetryModeling/sanjeev/LOOCV_analysis_1/electrode_2`
    - `/original/VoltammetryModeling/sanjeev/LOOCV_analysis_1/electrode_3`
- prefs.txt setup
  - You must at least fill out the job name and time
  - You cannot run this file if all parameters are empty
  - The "=" sign must be present in order for the script to load properly

SETUP FOR RUNNING THE JSI (R/MATLAB SCRIPT)

- JSI will treat the parent data directory as a working directory
- If a script needs to load a particular file in a subfolder, you should indicate this in your script
- JSI only supports R or Matlab scripts ending in .R or .m respectively

FUNCTION PARAMETERS

- Functions
  - `setup(workerName, projectName)`
  - `runAnalysis(workerName, projectName, jobname, scriptName, scriptType, email, mpthreads)`
  - `moveOutputFilesToJobFolders(workerName, projectName, jobName)`
  - `archive(workerName, projectName)`
- Parameters
  - `jobname` must be alphanumeric
  - `scriptName` must end in `.R` or `.m`
  - `scriptType` must be either "R" or "MATLAB"
  - Set `email` to `NULL` if you are not using it
  - `mpthreads` must be an integer

JSI WORKFLOW

1. Load your data into `/original/project/worker` in the IRB share folder and load scripts
2. Access the login node and navigate to the IRB share folder
3. Load the R module, any other modules needed, start R, and source `kishida_lab_JSI.R`
4. Run the `setup()` function. Fill out the resulting `dirs.txt` file and `prefs.txt` file
5. Run the `runAnalysis()` function
6. Wait for your email alert when the job is finished
7. Run `moveOutputFilesToJobFolders()` to get your output from your analysis into the respective job folders in `/scratch/worker/project/job`
8. Repeat steps 4-6 for staging if necessary
9. Run `archive()` function to permanently store your results in `/final`
10. Shut down the remote connection
11. Remove or copy your results from the job output folder

WARNING: When your job is running, do NOT modify anything in the job folder! Do not modify anything in the job folder until after you run `moveOutputFilesToJobFolders()`.

RUNNING ANALYSIS WITH THE JSI

- Demonstration in Linux (VirtualBox with Windows host)
  - Initial setup
  - Script submission
  - Looking at output
  - Archival

HELP AND OTHER RESOURCES

- Contact
  - Me: snamjosh@wakehealth.edu
  - Github page: https://github.com/snamjoshi/kishidaJSI
  - System administrator (Mike Greer): msgreer@wakehealth.edu
- Extra resources (available on Github page)
  - Written version of this presentation with more detail
    - Includes links to tutorials and training
  - Documentation for JSI