Introduction to High Performance Computing

GSND 5340Q, BMDA

W. Evan Johnson, Ph.D.
Professor, Division of Infectious Disease
Director, Center for Data Science
Rutgers University – New Jersey Medical School

2023-04-28

Section 1

Introduction to High Performance Computing

Acknowedgments

Thanks to the Rutgers Office of Advanced Research and Computing (OARC), Google, Wikipedia (images), and ChatGPT for helping me make these slides!

In addition, some information was obtained from the following online slides (click here).

What is High Performance Computing?

High Performance Computing (HPC) is the use of parallel processing for running advanced application programs efficiently, reliably, and quickly. It involves aggregating computing power in a way that delivers much higher performance than a typical desktop or workstation.

Why High Performance Computing?

- HPC allows users to tackle complex problems that require substantial computational resources.
- It enables simulations, data analysis, and modeling that would be infeasible on standard computers.
- Applications include weather forecasting, molecular modeling, financial modeling, and more.

HPC at Rutgers University

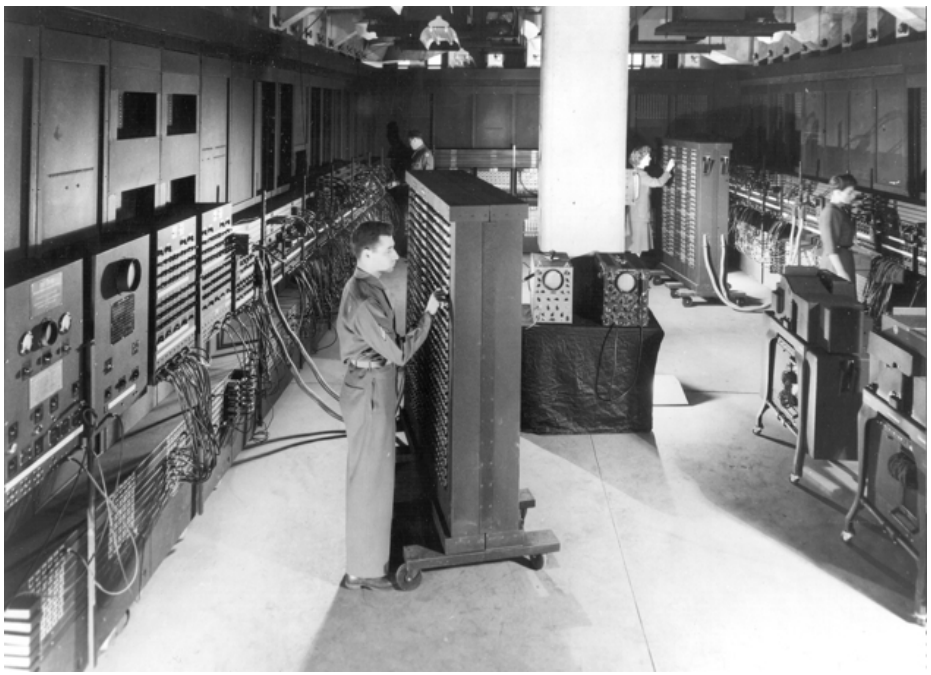
At Rugters, the primary HPC resources are governed by the Rutgers Office of Advanced Research and Computing (OARC). They provide significant online and synchronous (in person) resources for HPC, along with training in batch submission, programming languages, parallel programming, distributed systems, and HPC architectures.

Section 2

History of High-Performance Computing

Early Beginnings

1940s-1950s: The origins of HPC can be traced back to the development of early computers, such as the ENIAC (Electronic Numerical Integrator and Computer) and UNIVAC (Universal Automatic Computer).



Used for military (cracking Nazi codes) and scientific purposes, including calculations for atomic bomb development and weather forecasting.

Emergence of Supercomputers

1960s-1970s: The concept of supercomputers emerged with machines like the CDC 6600 and Cray-1.

These systems introduced vector processing and specialized architectures optimized for scientific and engineering applications.

Parallel Processing and Clusters

Seymore Cray was an American electrical engineer and supercomputer architect who designed a series of computers that were the fastest in the world for decades, and founded **Cray Research** which built many of these machines. He is often called "the father of supercomputing," and has been credited with creating the supercomputer industry.

However, Cray resisted the massively parallel solution to high-speed computing, offering a variety of reasons that it would never work as well as one very fast processor. He famously quipped "If you were plowing a field, which would you rather use: two strong oxen or 1024 chickens?"

Parallel Processing and Clusters

1980s-1990s: During the speed in supercomputers was primarily achieved through two mechanisms:

- **Vector processors:** these were designed using a pipeline architecture to rapidly perform a single floating point operation on a large amount of data. Achieving high performance depended on data arriving in the processing unit in an uninterrupted stream.
- **Shared memory multiprocessing:** a small number (up to 8) processors with access to the same memory space. Interprocess communication took place via the shared memory.

Parallel Processing and Clusters

1980s-1990s:Parallel processing became a key focus in HPC, leading to the development of massively parallel supercomputers like the Connection Machine and Thinking Machines CM-5.

- Clusters of commodity hardware also gained popularity due to their cost-effectiveness and scalability.
- Memory contention was a major impediment to increasing speed; the vector processors required high-speed access to memory but multiple processors working simultaneously created contention for memory that reduced access speed.
- Vector processing worked well with 4 or 8 processors, but memory contention would prevent a 64 or 128 processor machine from working efficiently.

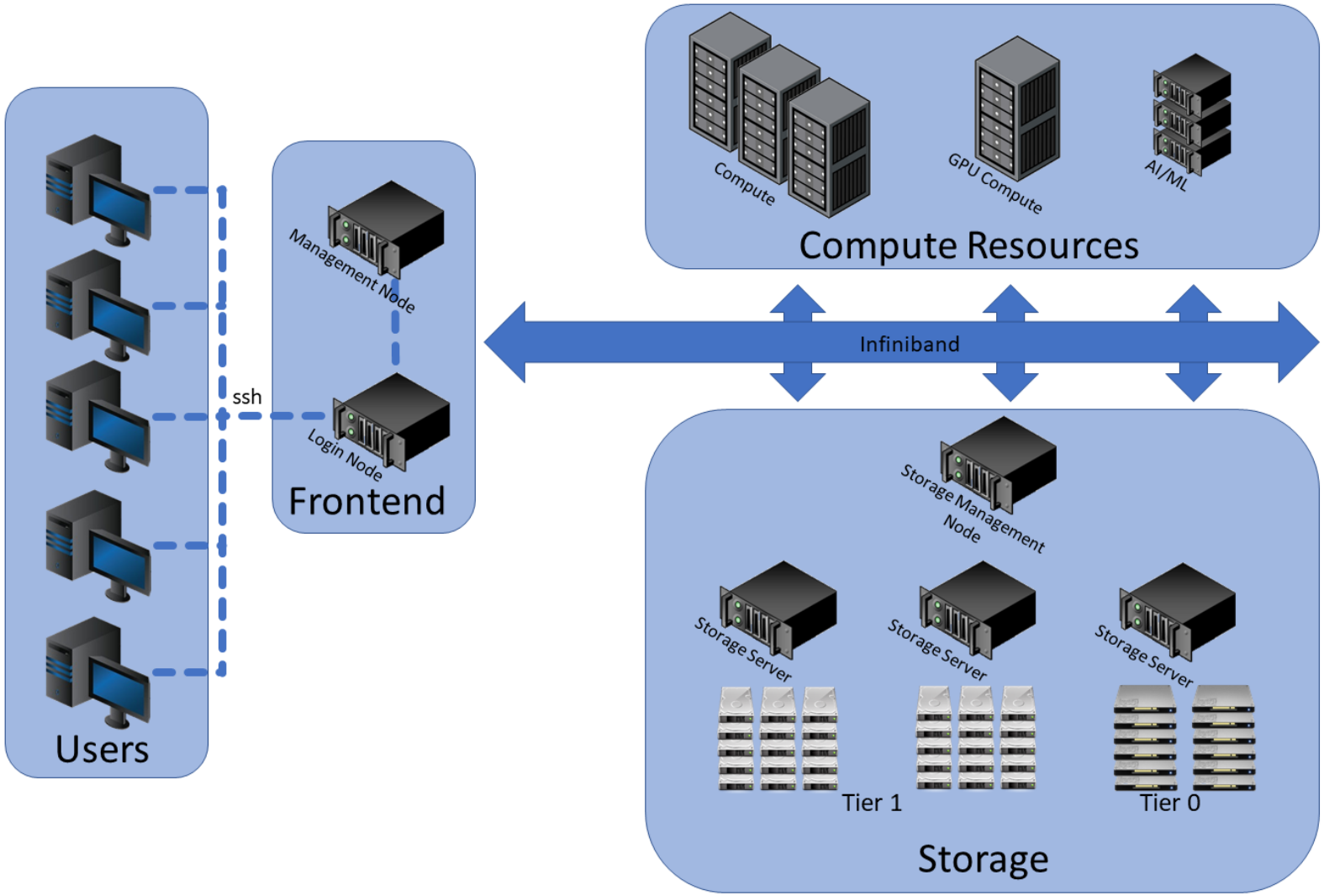
Evolution of HPC Architectures

- **2000s-Present**: HPC architectures continue to evolve with the rise of multi-core processors, accelerators (e.g., GPUs), and specialized interconnects. Most unviversities develop their own HPC resources.

Key Components of High Performance Computing

1. **Hardware**: Specialized computing infrastructure, including clusters, supercomputers, and accelerators like GPUs.
2. **Software**: Tools and libraries for parallel programming, job scheduling, and performance optimization.
3. **Networking**: High-speed interconnections for data exchange between computing nodes.
4. **Parallel Programming Models**: Frameworks and languages for writing parallel applications, such as MPI, OpenMP, and CUDA.

Typical HPC Architecture



Getting Started with High Performance Computing

1. **Learn Parallel Programming**: Understand parallel programming models and techniques.
2. **Access to HPC Resources**: Utilize computing clusters or cloud-based HPC services.
3. **Optimize Code**: Write efficient code and utilize performance profiling tools.
4. **Collaborate and Learn**: Engage with HPC communities, attend workshops, and collaborate with experienced users.

Future Trends

- **Usability and Access**: Interactive and online resources such as OnDemand have made HPC more accessible. Also, cloud computing such as the Google and Amazon Clouds have made HPC accessible to groups and companies (large and small).
- **Exascale Computing**: Efforts are underway to achieve exascale computing, capable of performing 10^18 floating point operations per second (FLOPS), with projects like the US Department of Energy's Exascale Computing Project.
- **Heterogeneous Architectures**: Continued focus on heterogeneous architectures combining CPUs, GPUs, and accelerators to optimize performance for specific workloads.
- **AI and Machine Learning**: Integration of artificial intelligence (AI) and machine learning techniques into HPC workflows for data analysis, optimization, and predictive modeling.

Section 3

Introduction to the Amarel HPC Cluster

HPC at Rutgers

About Amarel

In July 2017, the Office of Advanced Research Computing (OARC) unveiled Amarel, a "condominium" style computing environment developed to serve the university's wide-ranging research needs. The Amarel cluster provides a shared platform that optimizes resources for the benefit of all users. Named in honor of **Dr. Saul Amarel**, one of the founders of the Rutgers Computer Science Department and contributor to advanced computing and artificial intelligence research and methodologies, Amarel is designed to suit many different research applications.

Through Aberdeen (late 2023):

- 560 compute nodes
  - 27,872 Intel Xeon cores
  - 224 GPUs
  - Open OnDemand servers
  - InfiniBand FDR & EDR fabric

More information:

- Community-contributed software repository
- Spans three Rutgers data centers (Piscataway, Newark, and Camden) producing a unified compute, data, and storage system
- Rolling node/phase retirement with new node replacement, beginning in spring 2021

Click here for a short video about the Amarel cluster!

Key Features of Amarel

1. **Compute Nodes**: Amarel consists of multiple compute nodes with varying specifications, including CPUs and GPUs.
2. **Parallel File System**: A high-speed parallel file system for storing and accessing large datasets.
3. **Job Scheduler**: A job scheduling system for managing computing resources efficiently.
4. **Software Stack**: A comprehensive collection of software packages and libraries for various scientific computing tasks.
5. **Networking**: High-speed interconnects for fast communication between compute nodes.

Getting Started with Amarel

If you are off campus, you will need access to VPN

Click here for istructions for Windows Users

Click here for istructions for Mac Users

Getting Started with Amarel

1. **Request Access**: Contact OARC to request access to Amarel
2. **Training Workshops**: Attend training workshops offered by OARC to learn how to use Amarel effectively.
3. **Documentation and Support**: Explore documentation and seek support from OARC staff and the user community.
4. **Start Small**: Begin with small-scale experiments and gradually scale up as needed.

Section 4

Introduction to OnDemand for HPC Clusters

What is OnDemand?

- OnDemand is a web-based platform that provides a user-friendly interface for accessing and managing high-performance computing (HPC) clusters.
- It simplifies the process of submitting and monitoring jobs, accessing software, and managing data on HPC systems.
- Go to: http://ondemand.hpc.rutgers.edu for access (VPN needed for off campus users).

Key Features of OnDemand

1. **Web Interface**: Access HPC resources through a web browser from anywhere with an internet connection.
2. **Job Submission**: Submit and manage computational jobs without needing to use command-line interfaces.
3. **File Management**: Upload, download, and manage files and data on the HPC cluster directly from the browser.
4. **Interactive Sessions**: Launch interactive computing sessions for data analysis and exploration.
5. **Software Environment**: Access a variety of software packages and development tools installed on the HPC cluster.

Benefits of OnDemand

- **User-Friendly**: OnDemand provides a simplified interface, making HPC resources more accessible to a wider range of users.
- **Remote Access**: Users can access HPC resources remotely without needing to install any special software.
- **Increased Productivity**: Streamlined workflows and intuitive interfaces help users focus on their research instead of dealing with technical complexities.

Section 5

Introduction to Cluster Batch Job Scheduling with SLURM

What is SLURM?

- SLURM (Simple Linux Utility for Resource Management) is an open-source job scheduler and resource manager used on many high-performance computing (HPC) clusters.
- It allows users to submit, manage, and monitor batch jobs on computing clusters efficiently.

Key Concepts

1. **Partition**: A partition is a logical grouping of compute nodes with similar hardware characteristics or usage policies.
2. **Job Script**: A job script is a text file containing instructions for the scheduler, including job parameters, resource requests, and executable commands.
3. **Resource Allocation**: Users specify the required resources for their jobs, such as CPUs, memory, and runtime, in the job script.
4. **Job Status**: Users can monitor the status of their submitted jobs, including waiting, running, completed, or failed, using SLURM commands or monitoring tools.

Submitting a Job with SLURM

1. **Create a Job Script**: Write a job script using a text editor, specifying job parameters and commands.
2. **Submit the Job**: Use the `sbatch` command to submit the job script to SLURM.
3. **Monitor Job Status**: Check the status of the submitted job using commands like `squeue` or monitoring tools provided by SLURM.
4. **Retrieve Results**: Once the job completes, retrieve the results and data generated by the job from the designated output directory.

Example Job Script

```
#!/bin/bash
#SBATCH --job-name=my_job
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=4
#SBATCH --mem=8G
#SBATCH --time=1:00:00

# Execute your command here
./my_executable input_file.txt > output_file.txt

## Best Practices

- Resource Requests: Specify accurate resource requirements to
- Job Prioritization: Understand the partition structure and p
- Job Dependencies: Use job dependencies to sequence jobs and
- Error Handling: Include error handling and logging mechanism
```