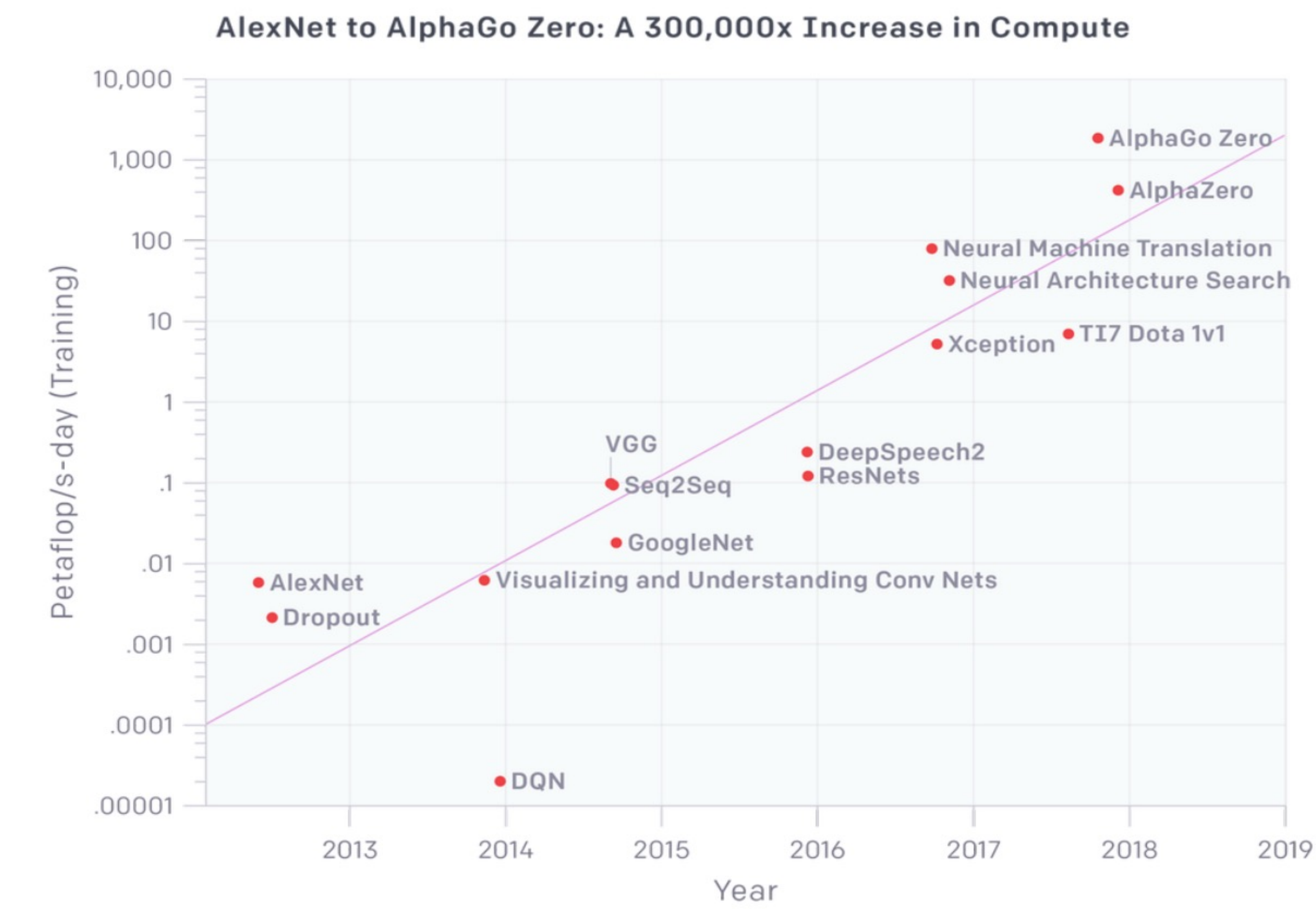
IT Infrastructure for Research:

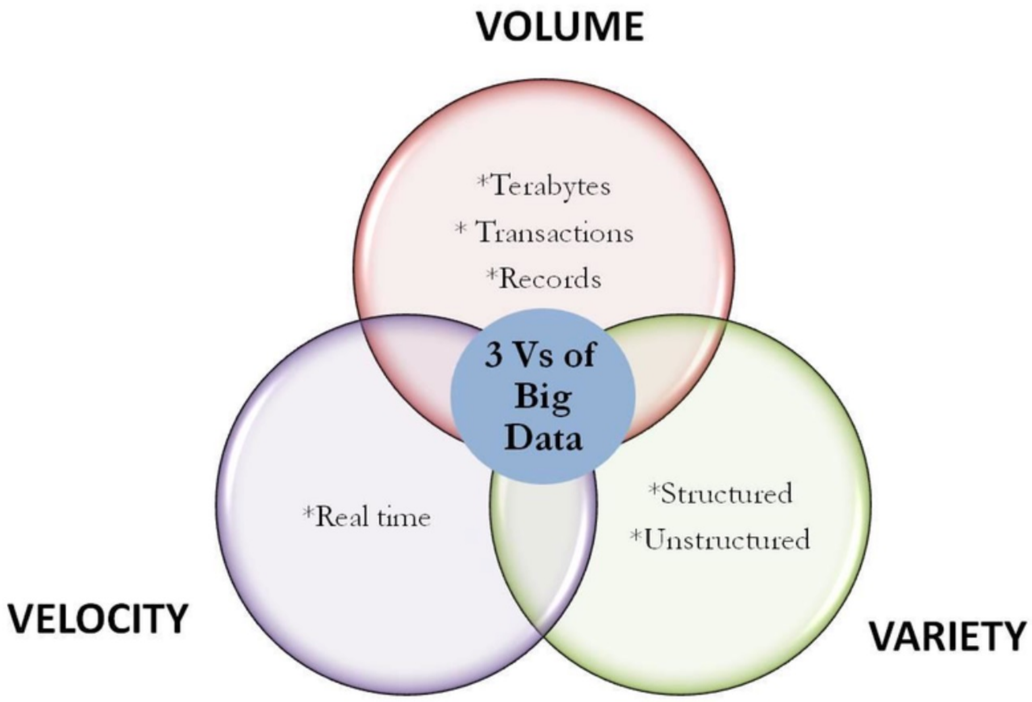
HPCs, Cloud Computing and beyond!

Mahsa Lotfi
April 19th , 2021

Special Thanks:
Mark Piercy
Riccardo Murri

Total Amount of Compute Power Used for different Machine Learning Models





We are exposed to “Big Data”

We need to be prepared!

A revolution for computing in science

Running hundreds of tasks

Managing hundreds of CPU hours for projects

It’s too much workload management
We need to have tools!

Alpha Pattern:

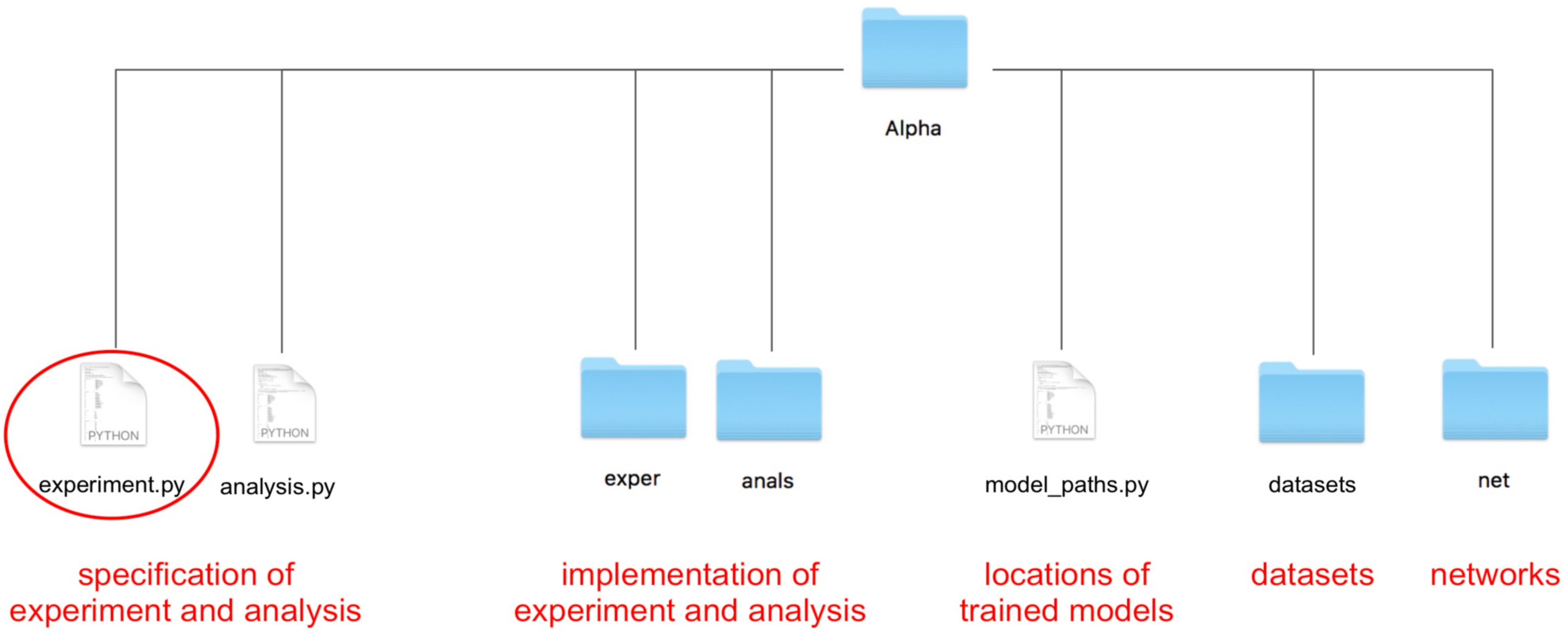


Table of Contents:

Platforms for Scientific Computing

Personal Workstations

- Interactive use
- Complete control over both the software and hardware
- Limited computing power

Large shared batch-queuing systems

- Centrally provided and administered
- Usually have “Job Schedulers”
- Standard “commodity servers” as compute nodes
- High-performance network interconnecting nodes
- Shared file system

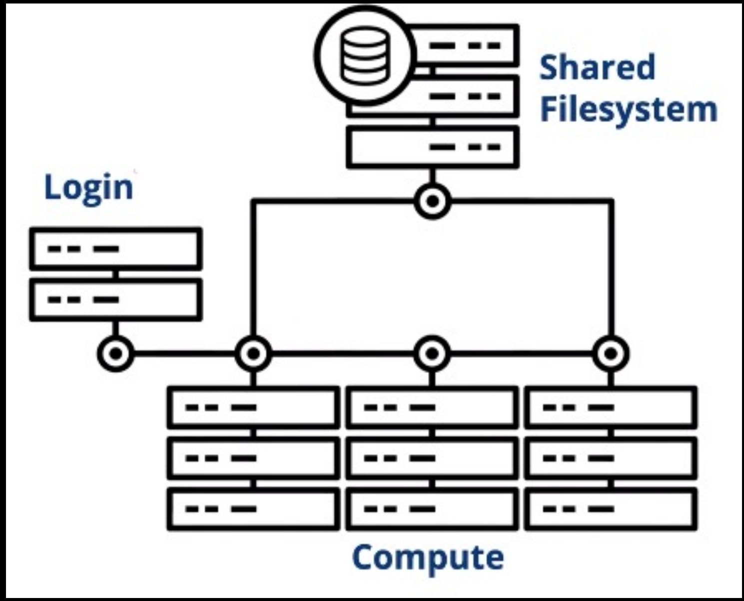
Cloud Computing

High Performance Computing (HPC)

- **HPC:** A collection of many separate servers/computers called "**nodes**"
- **Fast interconnections**
- **Distributed System:** components/nodes located on different networked computers
- **Parallel Computing:**
  - Large problems are divided into smaller ones
  - Simultaneously compute each smaller part

Cluster

- Multiple racks of computers without displays or keyboards
- Have common access to a common storage
- **Login Nodes:** Gateway to the cluster
- **Compute Nodes:** Computations are done here!
- **Shared Filesystem:** Presents data across all nodes



Login Nodes:

- Small number of head nodes (1 or 2)
- Access point for users to run jobs on the cluster
- Many users simultaneously log into the head node so no intensive jobs should run on the login node
- Good for basic tasks such as:
  - Uploading data
  - Managing files
  - Checking/ Managing jobs
  - Submitting jobs to the scheduler
  - Mostly has memory limit

Compute Nodes:

- Majority of computations running on these nodes
- Many more CPUs compared to regular computers (20-48)
- Big RAM compared to regular computers
- Some have accelerators (GPUs)
- GPU nodes: Both CPU cores and GPU available for running jobs

Why Using Clusters?

- ✓ Dealing with large datasets
- ✓ Need more disc space
- ✓ Need more memory
- ✓ Parallel jobs for faster results
- ✓ Using accelerators (GPUs)

Limitations:

- ✓ Not recommended if the jobs run for a long time (unless check-pointed)
- ✓ Not ideal for graphical tasks

Personal Workstations vs. HPC

Typical Sherlock Node:

- 24 CPUs in two sockets Intel 2.4GHz Xeon Skylake CPU
- With symmetric multithreading: **48** jobs simultaneously running
- About 8,192 CPUs for “owners” partition
- RAM size: 192GB
- GPU nodes (NVIDIA Kepler K80, K40, Volta V100)
- Big memory nodes (512GB, 1.5 and 3TB RAM)

Mac Book Pro Laptop:

- One CPU with multiple-cores
- 16 -64 GB RAM
- 512 GB Solid State Disk
- 6 cores with symmetric multithreading: **12** jobs simultaneously running

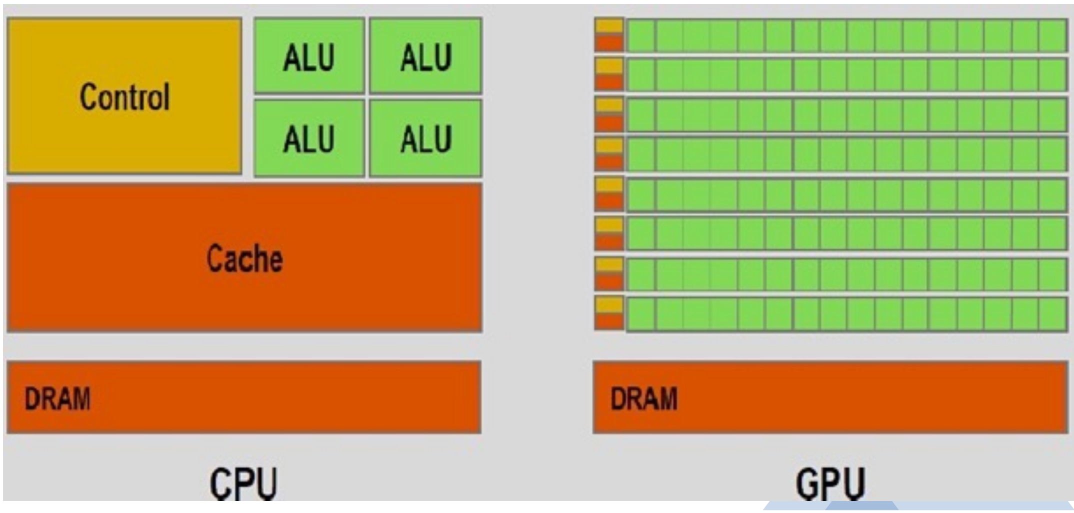
CPU VERSUS CORE

CPU	CORE
An electronic circuit inside the computer that handles all instructions it receives from hardware and software running on the computer	Processing unit that receives instructions to carry on actions based on the instructions
A component inside the computer	Located inside the CPU
A computer can have multiple CPUs or processors	A CPU can have single or multiple cores

GPU: Graphic Processing Unit

- Highly parallel structure thus more efficient than general-purpose CPUs
- Good for parallel processing of large blocks of data
- Examples of GPU use cases:
  - Rendering graphics to a screen
  - Running Monte Carlo simulations
  - Multiplying large matrices for a machine learning algorithm
- More arithmetic logical units (ALUs) to calculate with

NOTE: Codes need to be modified to be GPU-compatible



Pros and Cons of HPC:

Pros:

- Servers are always on
- Accessible from anywhere by anyone in your PI group
- Much more compute power, hundreds of CPUs
- Large memory servers up to 3TB of RAM
- Job Scheduler

Cons:

- Learn how to use a job scheduler and the Linux shell
- Need permission for some software installations
- Wait in the queue for the “Job Scheduler”

What is SLURM?

Simple Linux Utility Resource Management

SLURM

- Open source, fault-tolerant and highly scalable cluster/workload management system
- Job scheduling system for large and small Linux clusters
- You need to tell the scheduler:
    - What resources you need such as # of CPUs, RAM, time, partition
    - Load the modules and run your code
    - Need to request as few resources as you need so your jobs pend for as small a time as possible
- Why a management system?
    - Managing and balancing the compute resources availability
    - Balancing the workloads

Fairshare

- Goal of having a job queue:
    - Maximize utilization of the compute power
    - Ease the workload for users who do a lot of computation
    - Be fair to all users
- Used by SLURM to prioritize the tasks/jobs
- The more resources you use (CPU/RAM/Time/Nodes) in a 2 week sliding window, the lower your Fairshare score is and the more likely your jobs will wait in the queue

Fairshare (Cont.)

- Each job's priority in queue is determined by multiple factors, among them the user’s Fairshare score
- Past usage computed based on a sliding window and progressively forgotten over time
- Stanford Sherlock uses *backfill:* smaller jobs can go in front of larger jobs, often regardless of the users Fairshare factor, thus increasing clusters utilization

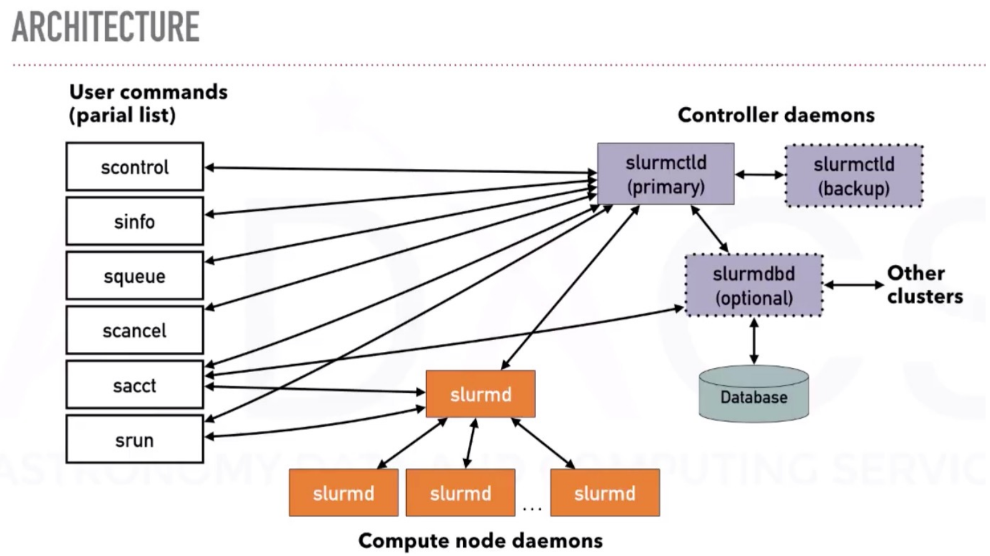
SLURM Architecture

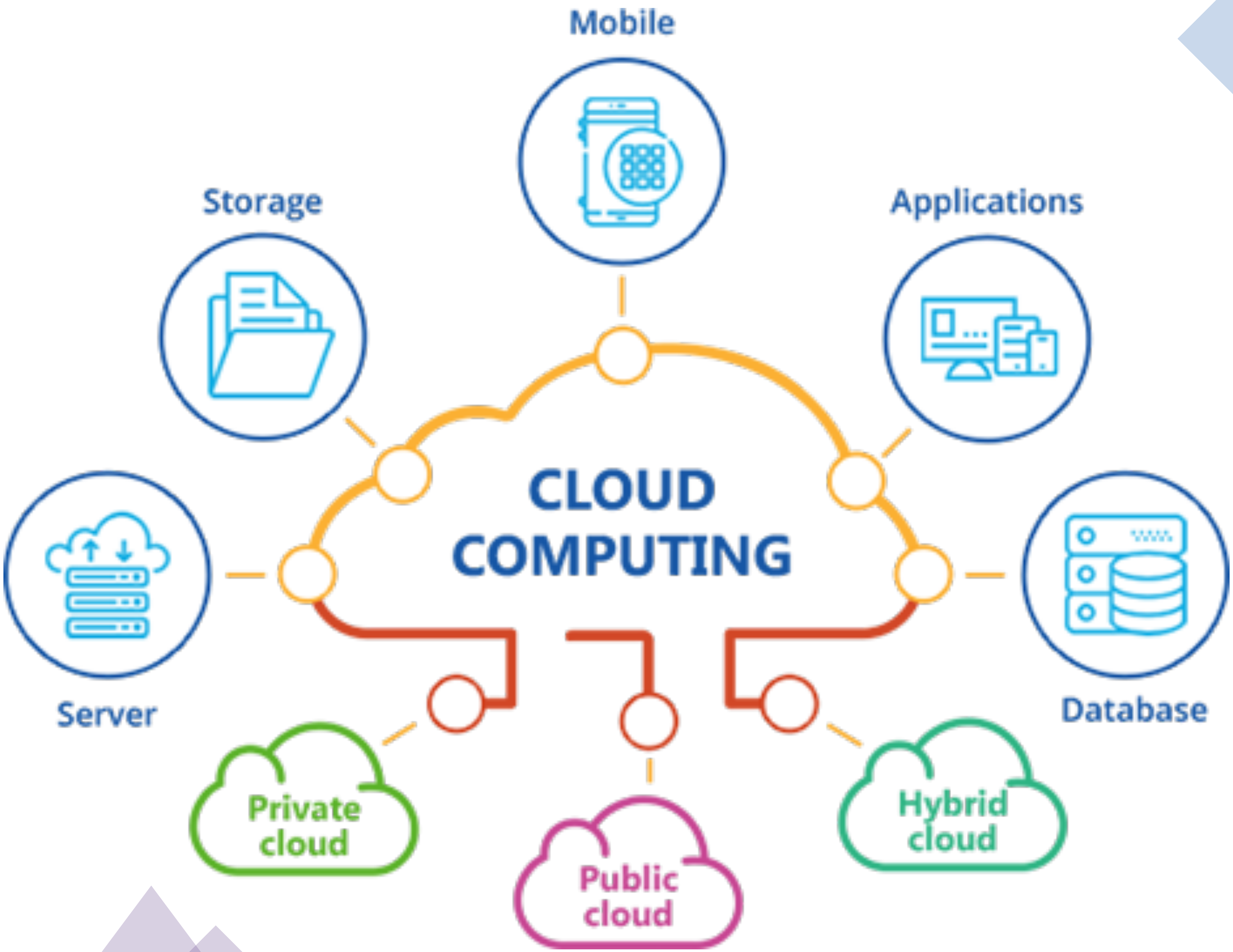
- **Slurmd**: Slurm daemon running on each compute node
  - Accepting and monitoring tasks
  - Launching the tasks
  - Killing the tasks

- **Slurmctld**: Slurm central daemon running on management node

- Information queried by several commands:

  - Sacct
  - Salloc
  - Sattach
  - Sview
  - Sinfo
  - Scontrol
  - etc.




Mobile
Storage
Applications
CLOUD
COMPUTING
Server
Database
Private
cloud
Public
cloud
Hybrid
cloud

Embarrassingly Parallel Problems (EPP)

- Problems in which little/no effort is needed to separate the problem into a number of parallel tasks (Naturally Parallel Algorithms)

- The opposite of EPP: **Inherently Serial Problems** (cannot be parallelized at all)

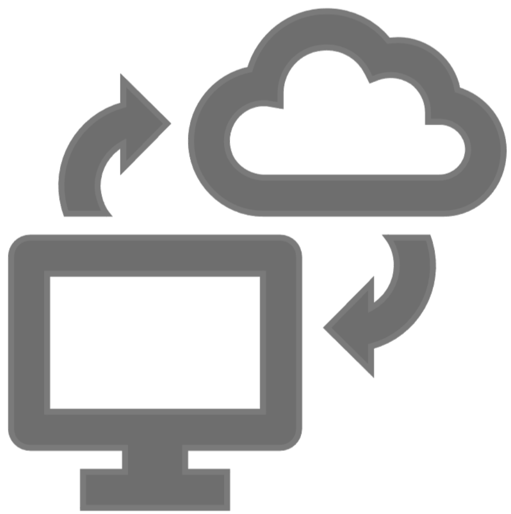
- **Examples:**
    - DFT, each harmonic calculated independently
    - BLAST
    - Large Scale Face Recognition, etc.

- **Features:**

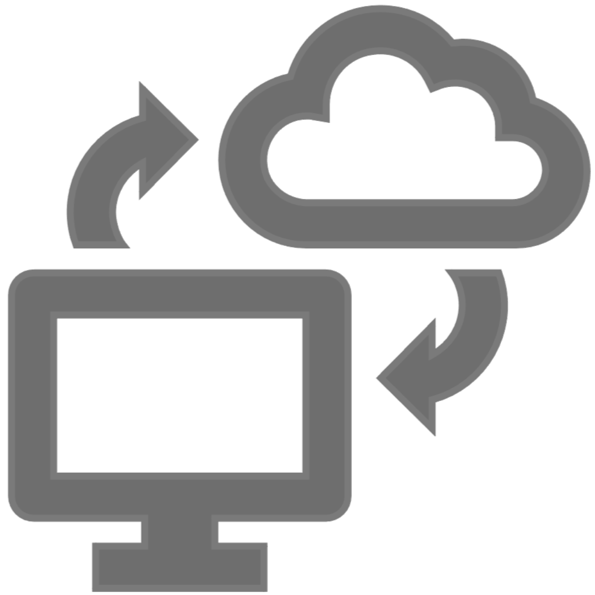
    - Almost no communication between the processes
    - Sub-solutions stored in disjoint memory locations
    - Sub-solution computations completely independent

Cloud Computing: IaaS

- **Infrastructure as a Service (IaaS)**
  - Virtualization of computing resources over the internet
  - Users log into the platform, create virtual machines, install OS etc.
  - **Examples:** AWS, GCE, Azure



Cloud Computing: PaaS



- **Platform as a Service (PaaS)**
  - Various hardware and software tools are available for application development to users over the internet
  - IT services available for users, accessible anywhere via a web browser
  - **Examples:** AWS Elastic Beanstalk, Google App Engine, Google Big Table, etc.

Cloud Computing: SaaS

- **Software as a Service (SaaS)**
  - Utilized for businesses in the cloud market
  - Utilizes the internet to deliver applications managed by a third-party vendor to its users
  - Directly through web browser, do not require any downloads or installations on the client side
  - **Examples:** Google Workspace, Concur, Cisco WebEx, etc.
- **On-premises** software deployment: Software installed directly on the user’s local machine, users have physical control over the hardware and the software

HPC vs. Cloud

- Good for EPP
- Good for large scale computing
- Mostly outperforms Cloud (Faster)
- Requires expensive hardware
- Fast interconnections

- Waiting for resources
- Learning a scheduler
- Not root, can't install some SWs

- Good for EPP
- Good for large scale computing
- Slow connections between nodes
- Cheaper than HPC
- Run on low cost commodity hardware
- No expensive HW/ SW upgrades
- No need to learn a job scheduler
- No waiting for resources not competing with hundreds of users for CPUs, RAM
- As a root ,install anything you want

ClusterJob

ClusterJob

- An automation system for high-throughput reproducible computations
- Easier parallelization of tasks
- CJ builds 'reproducible' computational packages that are easy to share with others
- Mainly written in Perl
- simple, easy-to-learn commands
- Currently supports MATLAB, Python and R
- Check pointing not necessary: Rerun the sub-problem

ClusterJob Useful Commands:

1. Write your Python/MATLAB/R code in a simple nested “for loop” format

2. Submitting jobs:
   - One job: cj run file.py sherlock –dep Files –alloc “-p owners” –m “test”
   - **Multiple jobs: cj parrun file.py sherlock –dep Files –alloc “-p owners” –m “test”**

3. Check the status of the jobs: cj state PID

4. Retrieve information: cj log PID

5. Gather all the results: cj reduce PID

6. Get the results in local machine: cj get PID

ElastiCluster

ElastiCluster

- Open-source software started at UZH
- Automated provisioning of virtual private clusters in the cloud
- Command line tool to **create, set up** and **scale** clusters with customized attributes and policies hosted on cloud
- Bespoke cluster up and running with a single command
- Additional commands can scale the cluster up and down

ElastiCluster Features:

- Supports several distros as base OS:
  - Debian
  - Ubuntu
  - CentOS
- Run on multiple clouds:
  - AWS
  - Google Cloud Engine
  - OpenStack
- Issue: **setup time** grows **linearly** with the number of cluster nodes

ElastiCluster **Config File**

```
# Create a cloud provider
[cloud/google]
provider=google
noauth_local_webserver=True
gce_client_id=*****
gce_client_secret=*****
gce_project_id=*****
zone=us-west1-b

[login/google]
image_user=ubuntu
image_user_sudo=root
image_sudo=True
user_key_name=elasticluster
user_key_private=~/.ssh/id_rsa
user_key_public=~/.ssh/id_rsa.pub

[cluster/gce]
cloud=google
login=google
setup=ansible-slurm
security_group=default
frontend_nodes=1
compute_nodes=1
ssh_to=frontend
# Ask for 500G of disk
boot_disk_type=pd-standard
boot_disk_size=500

[cluster/gce/frontend]
flavor=n1-standard-32
image_id=ubuntu-1604-xenial-v20171107b

[cluster/gce/compute]
flavor=n1-standard-32
#flavor=n1-highmem-8
image_id=ubuntu-1604-xenial-v20171107b
accelerator_count=1
accelerator_type=nvidia-tesla-k80
```

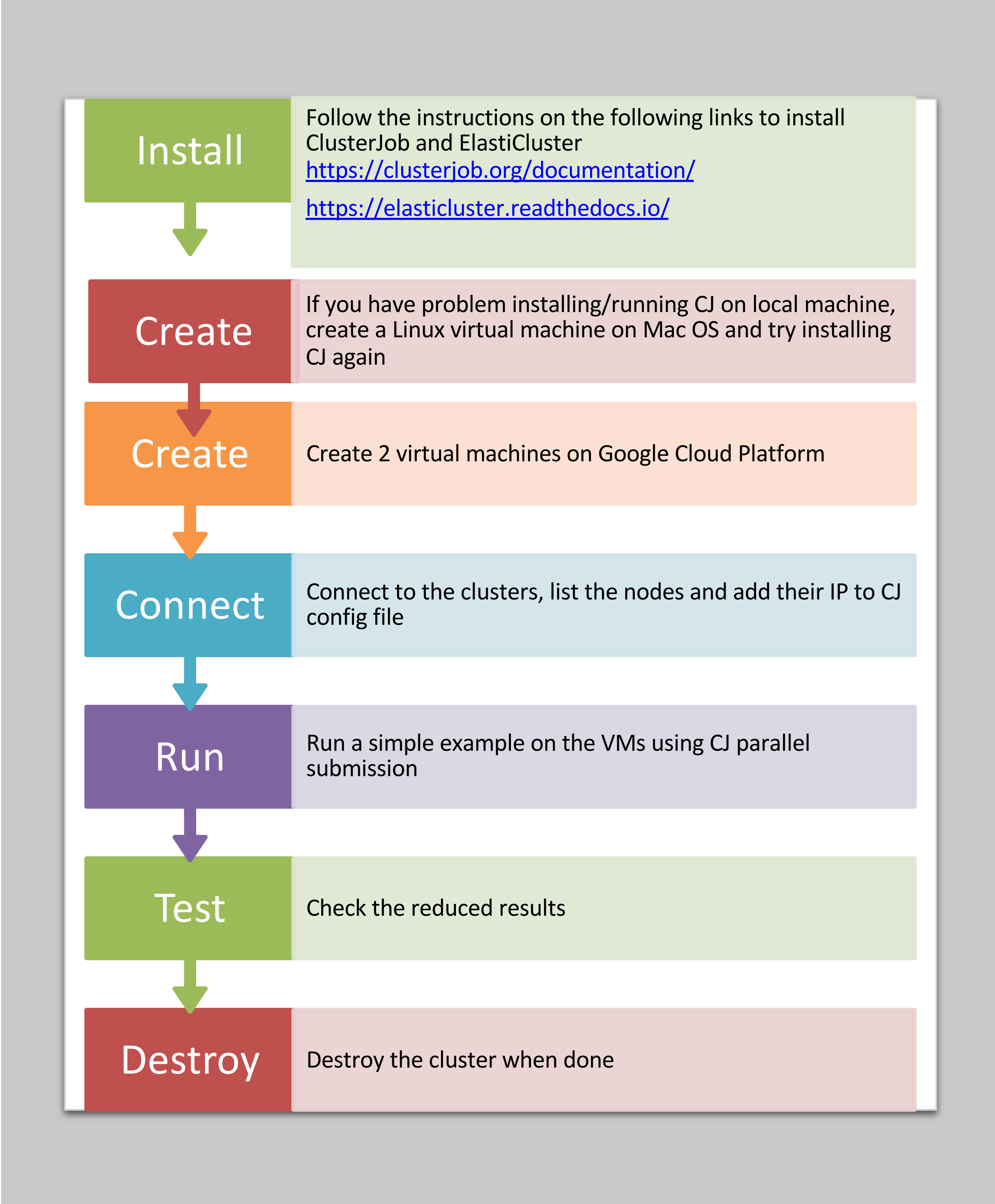
ElastiCluster-ClusterJob Model

The Cloud + ElastiCluster +

aws

python

ElastiCluster + ClusterJob Demo



Issues regarding ElastiCluster?

- ElastiCluster source code:
- http://github.com/gc3-uzh-ch/elasticluster ElastiCluster

- Documentation:
- https://elasticluster.readthedocs.org

- Mailing-list:
- elasticluster@googlegroups.com

- Chat / IRC channel:
- http://gitter.im/elasticluster/chat

Acknowledgements

Riccardo Murri
University of Zurich, Switzerland

Mark Piercy
Stanford Research Computing Center

Thanks!

Any questions?