Embarrassingly Parallel

By: Noureddin Sadawi

20/02/2019

Brunel University London

https://goo.gl/TJyhmL

Contents

- Concurrency vs Parallelism
- Ideal Parallelism
- Example Parallelisation Approaches
- Memory Coalescing (Matrix Multiplication)
- Several Example Parallel Problems/Techniques
- Parallelism at Multiple Levels
- Measuring Scaling Performance

time permitting:

- The KubeNow Project
- The H2O Platform

Concurrency vs Parallelism

- **Concurrency/Multithreading:** is when two or more tasks can start, run, and complete in overlapping time periods. It doesn't necessarily mean they'll ever be running at the same instant. Eg. multitasking on a single-core machine.
- **Parallelism:** is when tasks literally run at the same time, eg. on a multicore processor.

https://stackoverflow.com/a/1050257

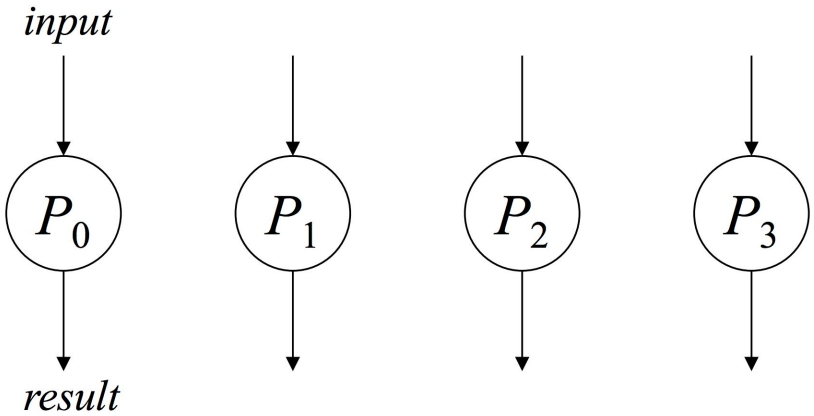
Quoting *Oracle's Multithreaded Programming Guide*:

- **Concurrency:** A condition that exists when at least two threads are making progress. A more generalized form of parallelism that can include time-slicing as a form of virtual parallelism.
- **Parallelism:** A condition that arises when at least two threads are executing simultaneously.

Java Concurrency and Multithreading Tutorial:
http://tutorials.jenkov.com/java-concurrency/index.html

Ideal Parallelism

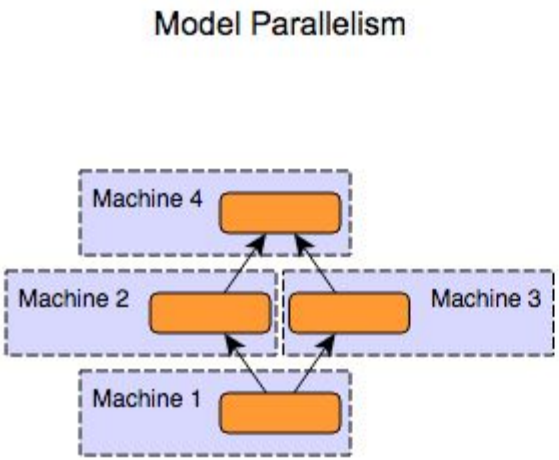
- An ideal parallel computation can be immediately divided into completely independent parts
  - “Embarrassingly parallel”
  - “Naturally parallel”
- No special techniques or algorithms required

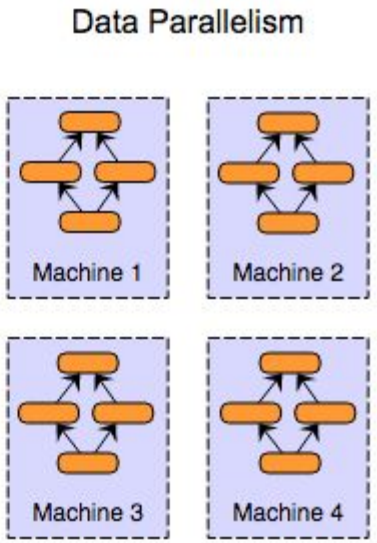


https://www.cs.fsu.edu/~engelen/courses/HPC/Algorithms1.pdf

Parallelisation Approaches

- **Model Parallelism:** different machines in the distributed system are responsible for the computations in different parts of a single network - for example, each layer in the neural network may be assigned to a different machine
- **Data Parallelism:** different machines have a complete copy of the model; each machine simply gets a different portion of the data, and results from each are somehow combined.

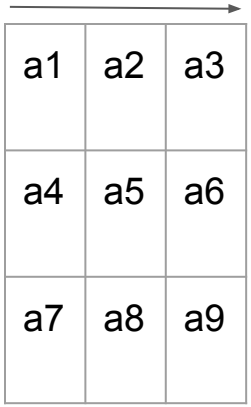


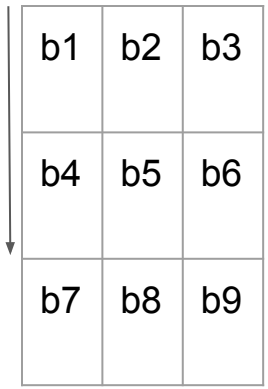


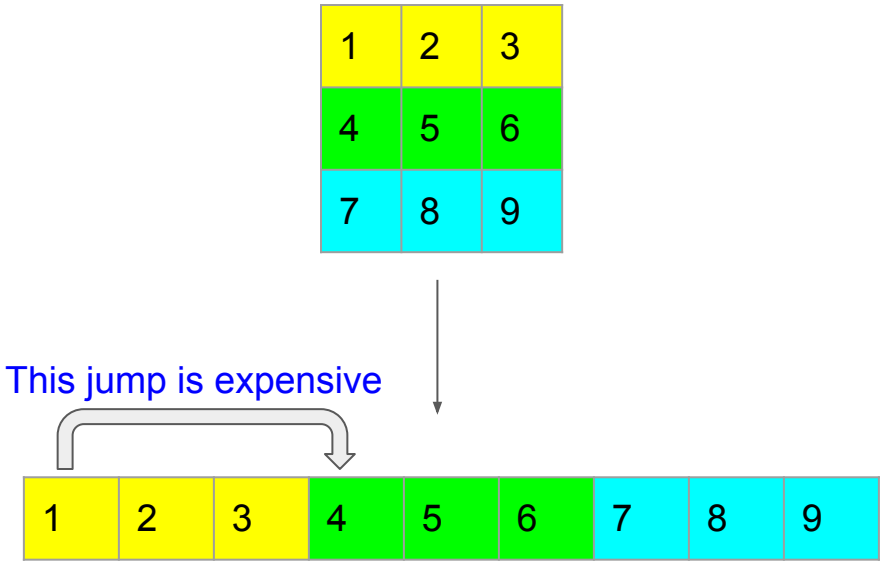
http://engineering.skymind.io/distributed-deep-learning-part-1-an-introduction-to-distributed-training-of-neural-networks

Memory Coalescing 1/2

- 2D matrices are represented as vectors in memory
- Take two **square** Matrices **A** and **B**, we want to: **A.B**
- When Multiplying Matrices, we make expensive moves in **B**

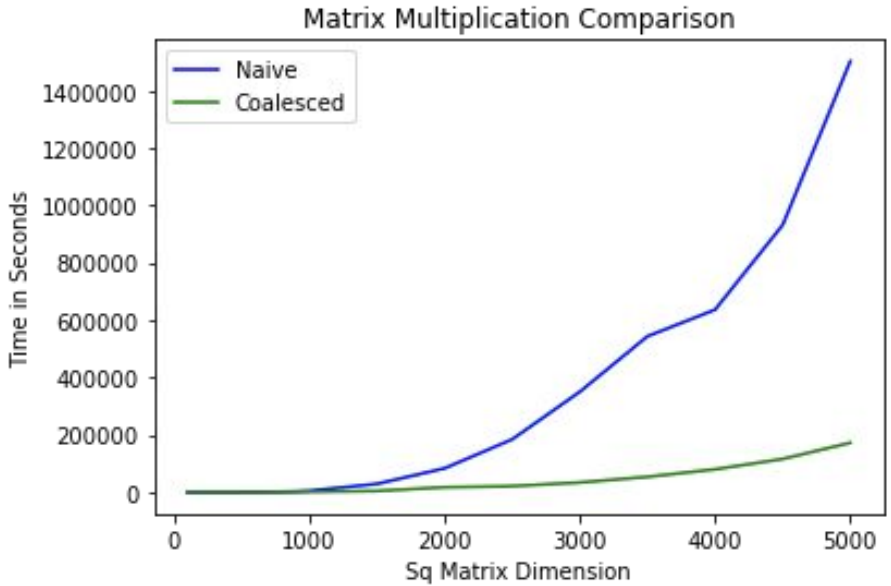






Memory Coalescing 2/2

- Square Matrix Multiplication in Java
- Transpose B
- Move row by row instead of column by column

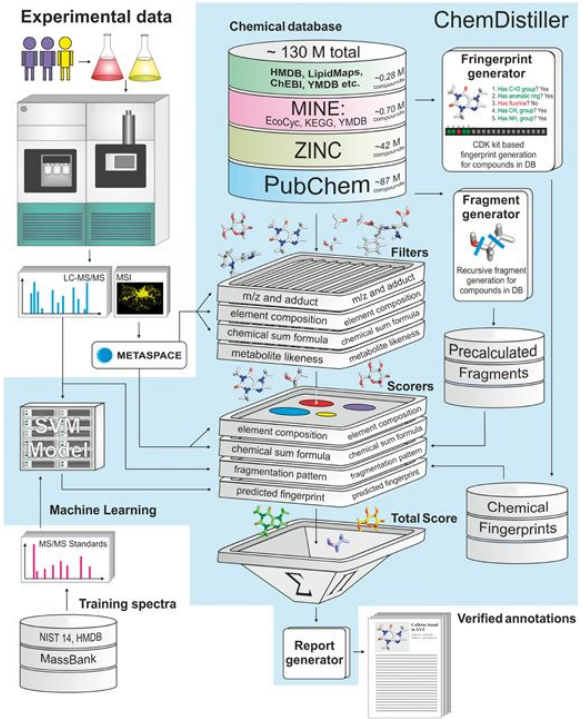


HPC Usage Example 1:

- Task - Permutation test
  - Data: one dataset -> created **10k** datasets by randomising the variables (not the outcome)
  - Run linear regression with 10 fold cross validation on each dataset to obtain a score (e.g. RMSE)
- Performance:
  - When done sequentially, using 1 core, 10k datasets took **> 83 hours**
  - When parallelising the task, using a HPC cluster, 10k datasets took **< 83 minutes**

HPC Usage Example (The ChemDistiller Project):

- Task - Compute Fingerprints for ~130 Million Chemical Compounds
- Data: 13k files, each containing 10k compounds
  - Input compound representation: SMILES
- Performance:
  - Using 1 core, in average, 1 file takes ~ 8 hours, up to 24 hours for files with larger molecules
  - When done sequentially, using 1 core, the 130M compounds would finish in **> 11 YEARS (1-2 years using 8 cores)**
  - **When parallelising the task, using a HPC cluster, 130M compounds took ~ 22 days**

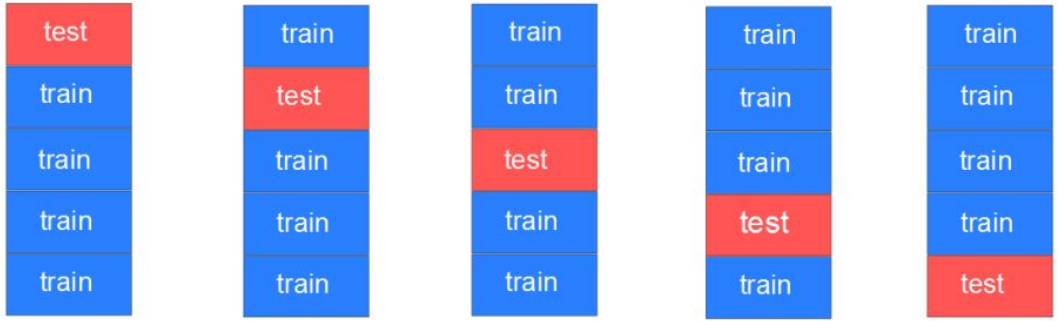


ChemDistiller: an engine for metabolite annotation in mass spectrometry

https://academic.oup.com/bioinformatics/article/34/12/2096/4852828

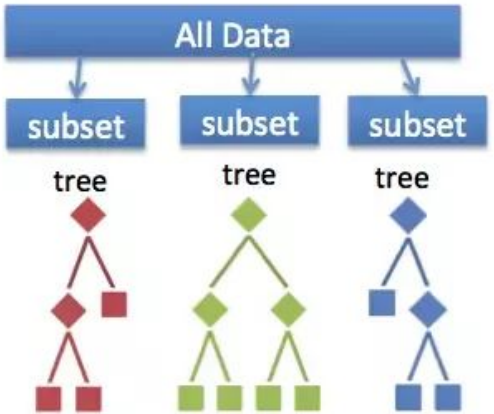
Cross Validation

- A model is trained using **k-1** of the folds as training data
- The resulting model is validated on the remaining part of the data (i.e., it is used as a test set to compute a performance measure such as accuracy).

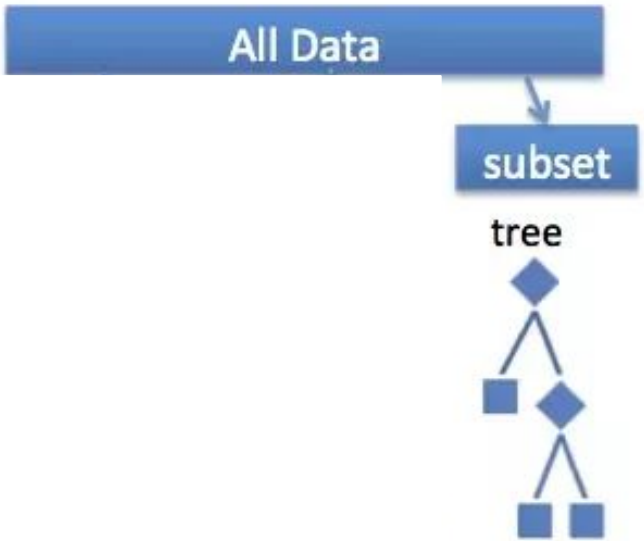


Random Forest

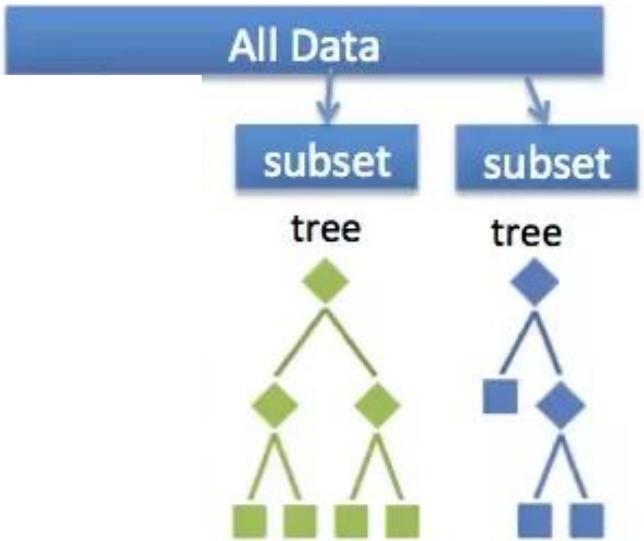
- Random forest algorithm is a supervised classification algorithm. As the name suggests, this algorithm creates the forest with a number of trees
    - In general, the **more trees in the forest** the more robust the forest looks like.
- Bootstrapping algorithm with Decision tree (CART) model.
- Say, we have **n** observations in the complete population with **m** variables.
- Random forest tries to build multiple CART models with different samples and different initial variables.
    - For instance, it will take a random sample of **i** observations and **j** randomly chosen initial variables to build a CART model (**j << m**).
    - It will repeat the process (say) **k** times and then make a final prediction on each observation.
    - Final prediction can simply be the mean (or mode) of each prediction.



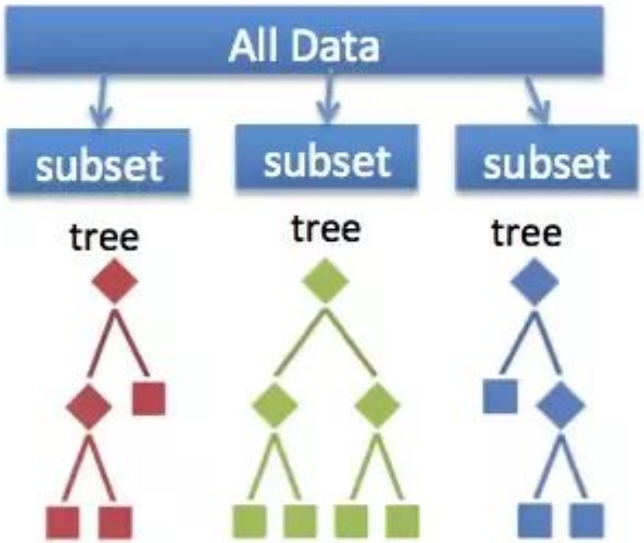
Random Forest



Random Forest



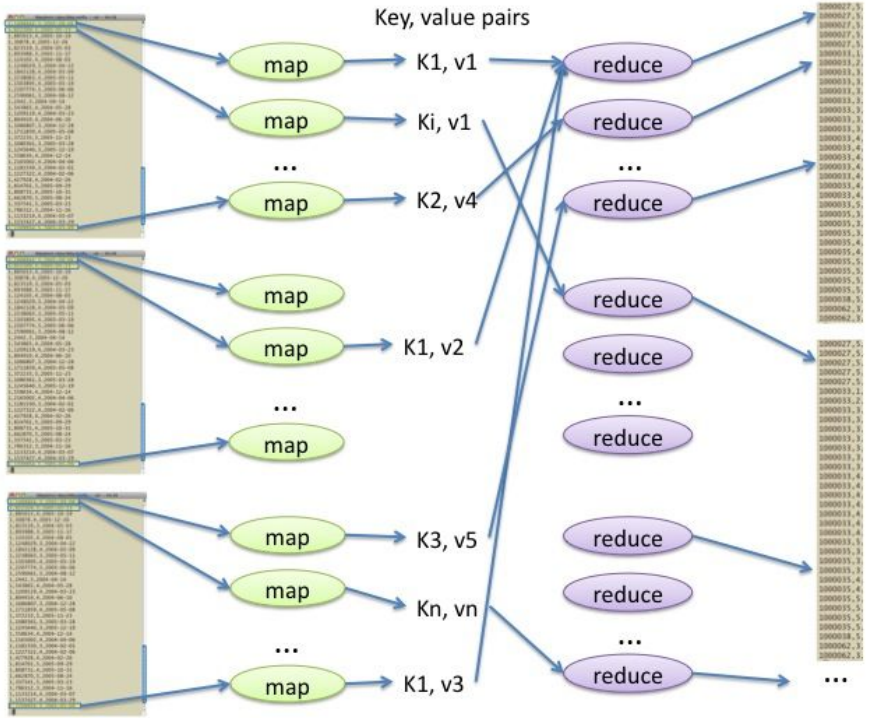
Random Forest



Analysing Very Large Files

- **Map** functions can run in parallel and pass their results to **Reduce** functions
- Results are output in sorted order by the keys created by the reduce function

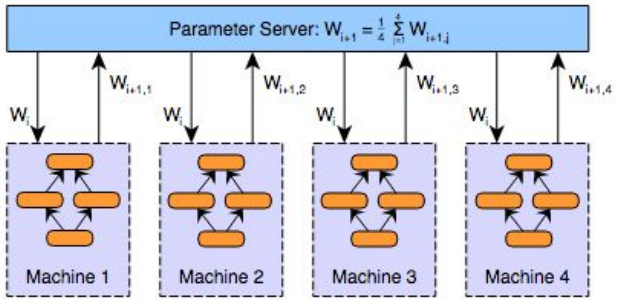
- **Sorting very large files?**



http://selkie.macalester.edu/csinparallel/modules/IntroWMR/build/html/MapReduceIntro/MapReduceIntro.html

Neural Nets Parameter Averaging

1. Initialize the network parameters randomly based on the model configuration
2. Distribute a copy of the current parameters to each worker
3. Train each worker on a subset of the data
4. Set the global parameters to the average the parameters from each worker
5. While there is more data to process, go to step 2



- Steps 2 through 4 are demonstrated in the diagram
- *W* represents the parameters (weights, biases) in the neural network
- Subscripts are used to index the version of the parameters over time, and where necessary for each worker machine

Correlation Matrix

$$r = \frac{1}{n-1} \sum_{i=1}^{n} \left(\frac{x_i - \bar{x}}{s_x} \right) \left(\frac{y_i - \bar{y}}{s_y} \right)$$

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	probeset	ITCC0600	ITCC0601	ITCC0602	ITCC0604	ITCC0607	ITCC0608	ITCC0609	ITCC0611	itcc0001	itcc0002	itcc0003	itcc0008	itcc0009	itcc0010	itcc0013
2	1007_s_at	430.8	226.1	130.6	75	54.9	195.5	124.8	221.5	82.1	57.5	67.1	174	55.8	143.6	110.2
3	1053_at	79.5	95.7	178.8	185.8	144.7	113.1	150.2	157.5	138.4	140.7	198.1	123.7	254.8	130.4	161.5
4	117_at	65.9	14.9	1.8	28.7	4.7	35.2	3	25.5	39.3	17.3	23.7	39.7	4.1	29.8	64.1
5	121_at	203.3	79.4	74.1	60.3	79.7	54.5	60.1	113.3	95.6	85.8	80.1	81.9	72.7	91.9	112.1
6	1255_g_at	5	129.8	177.8	291.1	192.4	20	255.4	5.6	54.9	56.6	63.8	61.2	300	41.1	40.9
7	1294_at	114.6	57.1	40.1	37.1	44.2	50.9	40.4	28.3	56.3	51.6	38.6	127.8	44	79.4	83.8
8	1316_at	113	92.7	115.1	106.5	63.5	103.6	121.6	97.8	26.1	33.2	27	33.5	24	17.3	52.9
9	1320_at	12.1	17.9	1.7	1	2.2	11.2	17.2	16.1	33.2	58.1	27.4	14.3	46.6	37.9	40.4
10	1405_i_at	313.8	135.4	26.1	89.8	164.4	223.2	81	15.7	64.1	29	13	406.7	28.9	149.7	233.3
11	1431_at	18.8	13.7	16.2	8.6	8.2	66	10.3	15.1	33	14.2	79.8	12.5	15.1	7.4	20.9
12	1438_at	2.4	51.9	5.2	9.4	22	6.2	8.3	38.6	69.7	193.6	31.5	107.6	59.1	100.6	120.6
13	1487_at	94	88.9	96	52.1	98.3	67.8	92	91.4	63	99.1	160.7	106.4	54.8	84.4	101.6
14	1494_f_at	30	18.8	33	49	34	33.5	44.9	35.3	26.1	22.9	19.8	18.7	19.8	17	29.1
15	1552256_a_at	33.4	55.5	57.9	53.1	52	60.2	63.7	117	182.2	79.1	185.9	187.1	137.1	143.8	251.8
16	1552257_a_at	109.2	78.6	70	83.8	62.7	97.1	73.4	245.7	126.6	113.4	151.9	117.9	112.2	175.1	159.7
17	1552258_at	31.2	8.3	5.3	20.3	7	20.7	35	22.7	22.6	12.2	11.5	24.9	27.3	18.7	16.5
18	1552261_at	42.2	5.9	5.1	14.1	13.7	6.5	18.9	15.2	19.7	22.5	9.9	29.8	12.8	13.8	24.9

Parallelism at Multiple Levels

- Sometimes the problem at hand is parallelizable at more than one level
- A typical example is when we want to run a parallel algorithm several times
- An example is to run RandomForest on thousands of datasets
- Which level do we choose?

Measuring Scaling Performance

Strong Scaling:

- Fixed data size (ex: 10000 datasets)
- Change number of parallel processes
- Check performance

Weak Scaling:

- Variable data size (ex: 10, 100, 1000, 10000 datasets)
- Number of parallel processes changes with data size
- Check performance

Metrics:

- Speedup $S(n) = \frac{T_1}{T_n}$

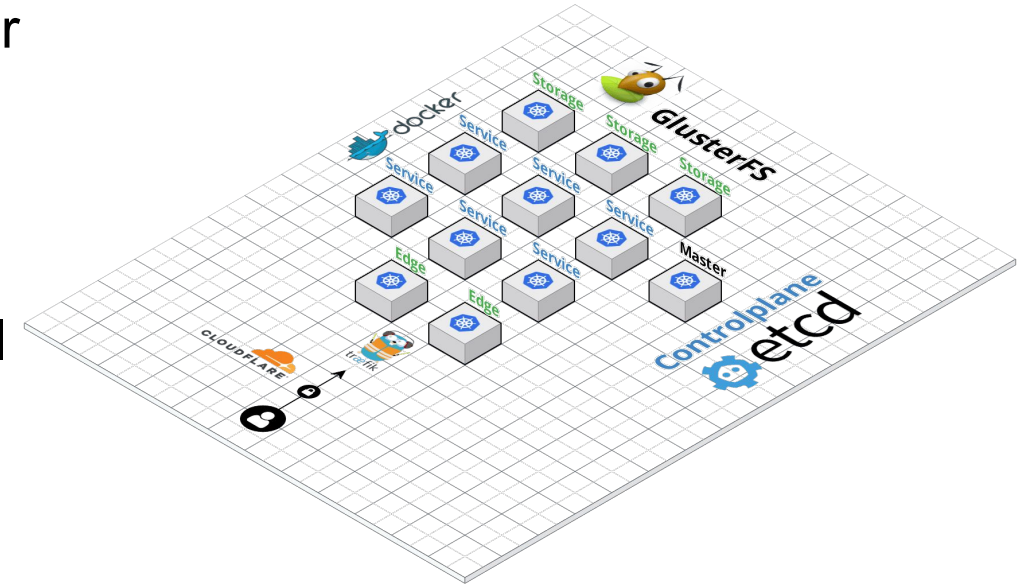
Scaling Efficiency $E(S) = \frac{T_1}{nT_n} = \frac{S(n)}{n}$

Summary

1. Embarrassingly Parallel problems are everywhere
2. It is a mindset .. a way of thinking about problem solving
3. Plenty of platforms
4. Sometimes it is a matter of mapping the problem into a format that a parallel platform can process
5. Many real life examples show it is worth the effort!

KubeNow

- A cloud agnostic platform for microservices, based on Docker and Kubernetes
- Fast Kubernetes operations
- Helps you in lifting your final application configuring DNS records and distributed storage



https://github.com/kubenow/KubeNow

Deploy PhenoMeNal with KubeNow

PhenoMeNal
Large-Scale Computing for Medical Metabolomics

KubeNow

https://goo.gl/jZx5sn

kn

kn-destroy

kn-apply

kn-az,gce

kn-ansible

kn-ssh

kn-helm

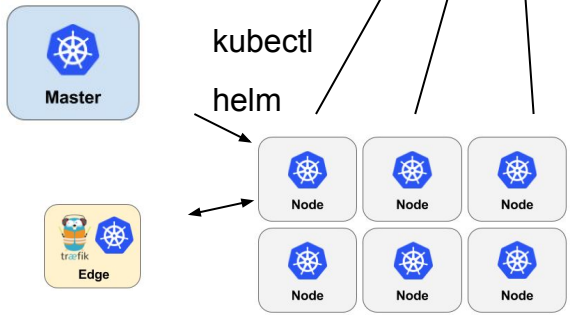
HashiCorp Terraform

ANSIBLE

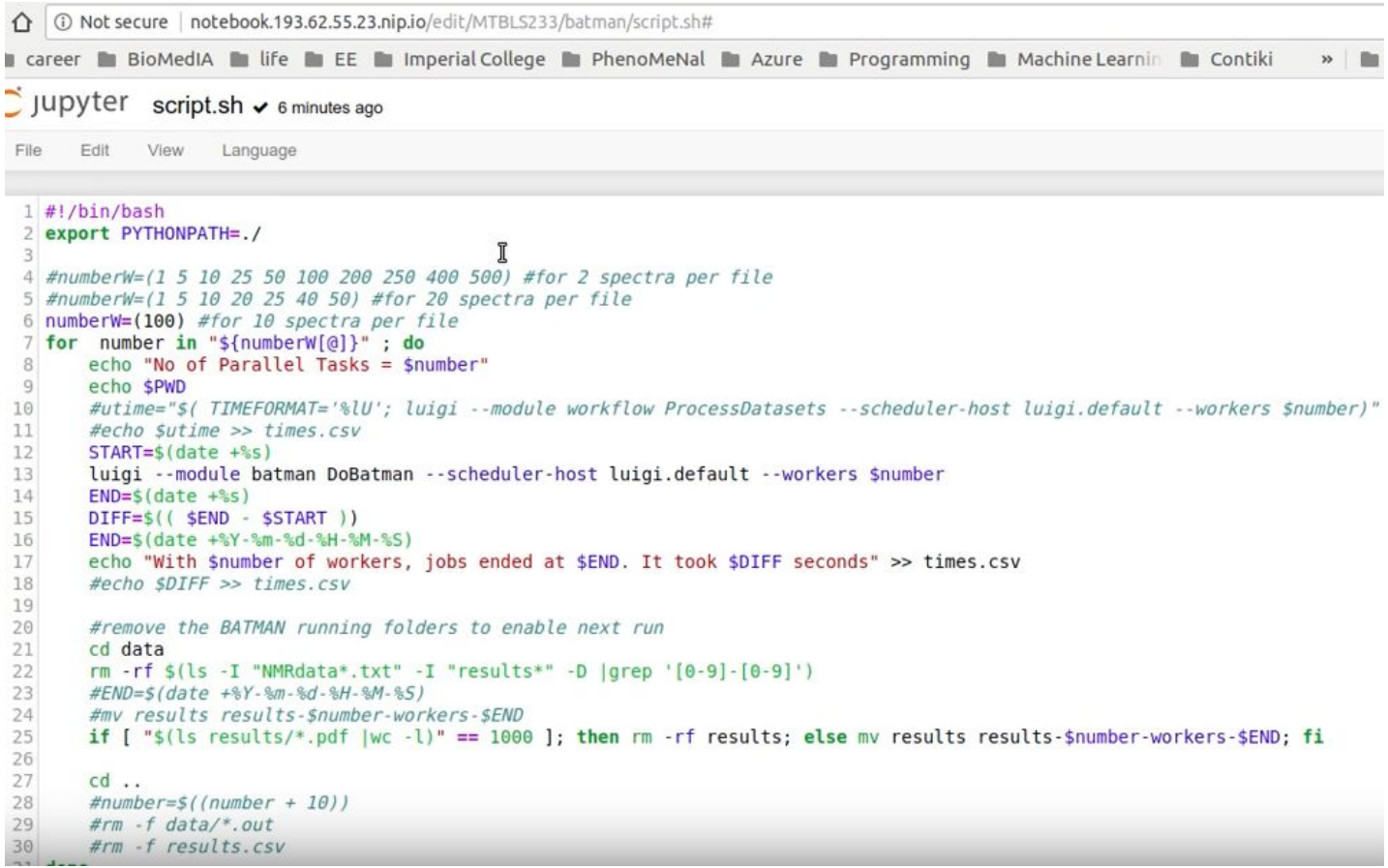
DockerHub

kubectl

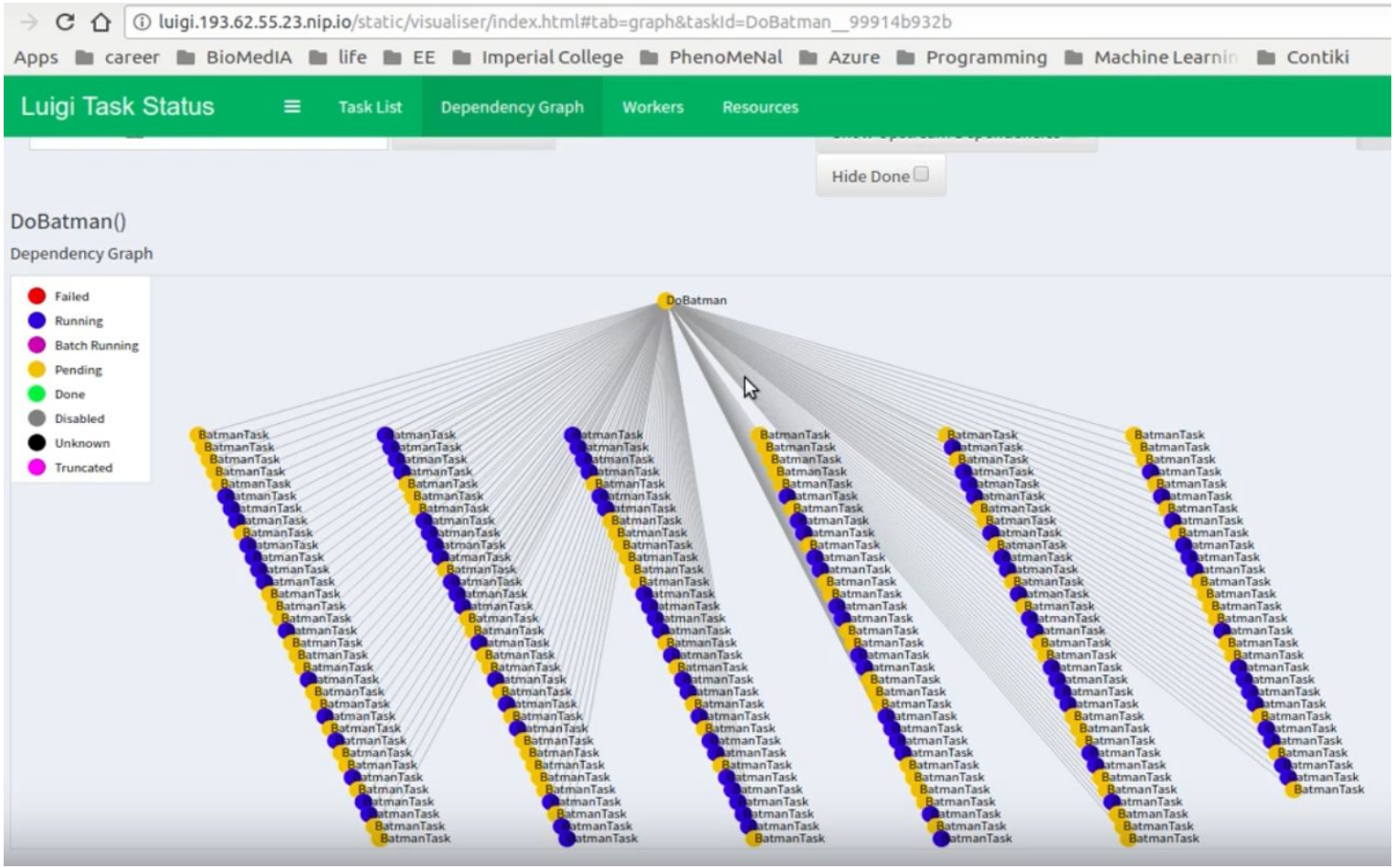
helm



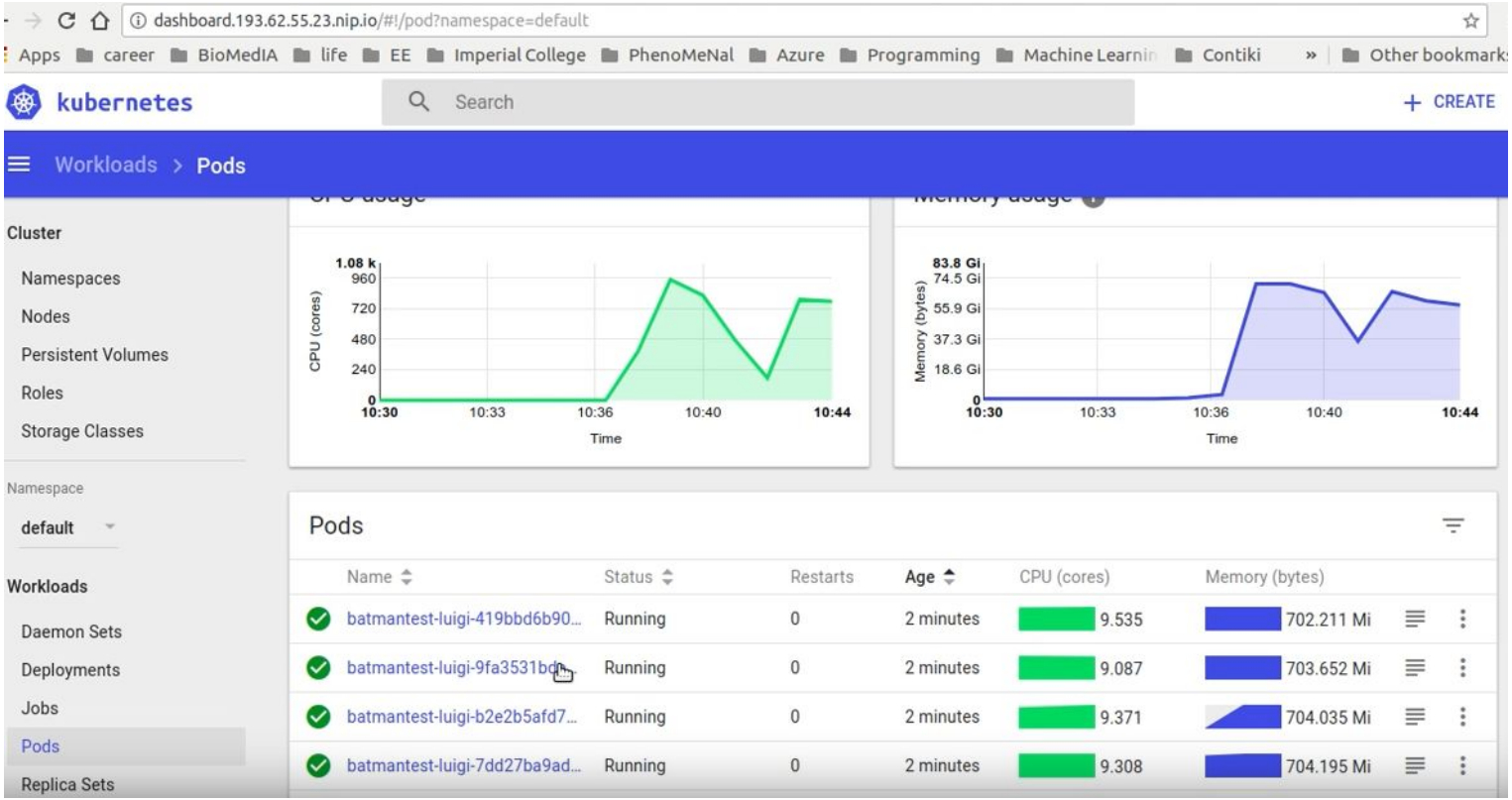
Jupyter


Not secure | notebook.193.62.55.23.nip.io/edit/MTBLS233/batman/script.sh#
career BioMedIA life EE Imperial College PhenoMeNal Azure Programming Machine Learnin Contiki
jupyter script.sh 6 minutes ago
File Edit View Language
#!/bin/bash
export PYTHONPATH=./
#numberW=(1 5 10 25 50 100 200 250 400 500) #for 2 spectra per file
#numberW=(1 5 10 20 25 40 50) #for 20 spectra per file
numberW=(100) #for 10 spectra per file
for number in "${numberW[@]}" ; do
echo "No of Parallel Tasks = $number"
echo $PWD
#utime="$(TIMEFORMAT='%lU'; luigi --module workflow ProcessDatasets --scheduler-host luigi.default --workers $number)"
#echo $utime >> times.csv
START=$(date +%s)
luigi --module batman DoBatman --scheduler-host luigi.default --workers $number
END=$(date +%s)
DIFF=$(($END - $START))
END=$(date +%Y-%m-%d-%H-%M-%S)
echo "With $number of workers, jobs ended at $END. It took $DIFF seconds" >> times.csv
#echo $DIFF >> times.csv
#remove the BATMAN running folders to enable next run
cd data
rm -rf $(ls -I "NMRdata*.txt" -I "results*" -D |grep '[0-9]-[0-9]')
#END=$(date +%Y-%m-%d-%H-%M-%S)
#mv results results-$number-workers-$END
if ["$(ls results/*.pdf |wc -l)" == 1000]; then rm -rf results; else mv results results-$number-workers-$END; fi
cd ..
#number=$((number + 10))
#rm -f data/*.out
#rm -f results.csv

Luigi


luigi.193.62.55.23.nip.io/static/visualiser/index.html#tab=graph&taskId=DoBatman__99914b932b
Apps
career
BioMedIA
life
EE
Imperial College
PhenoMeNal
Azure
Programming
Machine Learnin
Contiki
Luigi Task Status
Task List
Dependency Graph
Workers
Resources
Hide Done
DoBatman()
Dependency Graph
Failed
Running
Batch Running
Pending
Done
Disabled
Unknown
Truncated
DoBatman
BatmanTask

Kubernetes Dashboard

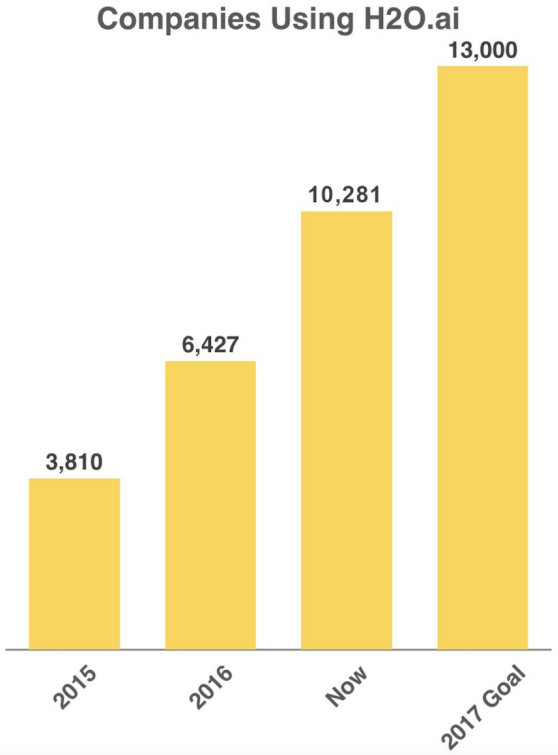


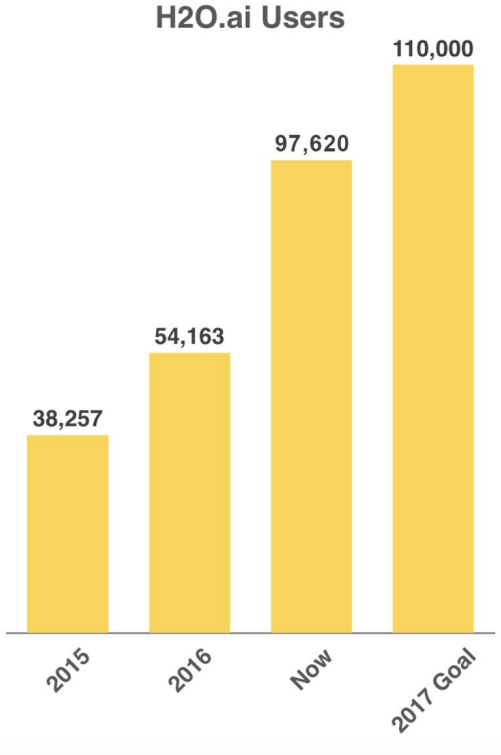
H2O.ai

- Founded in 2012, Mountain View, CA Stanford Math & Systems Engineers
- It is produced by the company *H2O.ai* (formerly *0xdata*)
- Open Source Software
- Ease of Use via Web Interface or API
- Cutting Edge Machine Learning Algorithms
- R, Python, Scala, Spark & Hadoop Interfaces Distributed Algorithms Scale to Big Data
- Simple deployment without intermediary transformations
- In-Memory Parallel Processing

- https://github.com/h2oai
- http://docs.h2o.ai
- https://www.stat.berkeley.edu/~ledell/docs/h2o_hpccon_oct2015.pdf

H2O Community





H2O: Current Algorithm Overview

Statistical Analysis

- Linear Models (GLM)
- Cox Proportional Hazards
- Naïve Bayes

Ensembles

- Random Forest
- Distributed Trees
- Gradient Boosting Machine
- R Package - Super Learner Ensembles

Deep Neural Networks

- Multi-layer Feed-Forward Neural Network
- Auto-encoder
- Anomaly Detection
- Deep Features

Clustering

- K-Means

Dimension Reduction

- Principal Component Analysis
- Generalized Low Rank Models

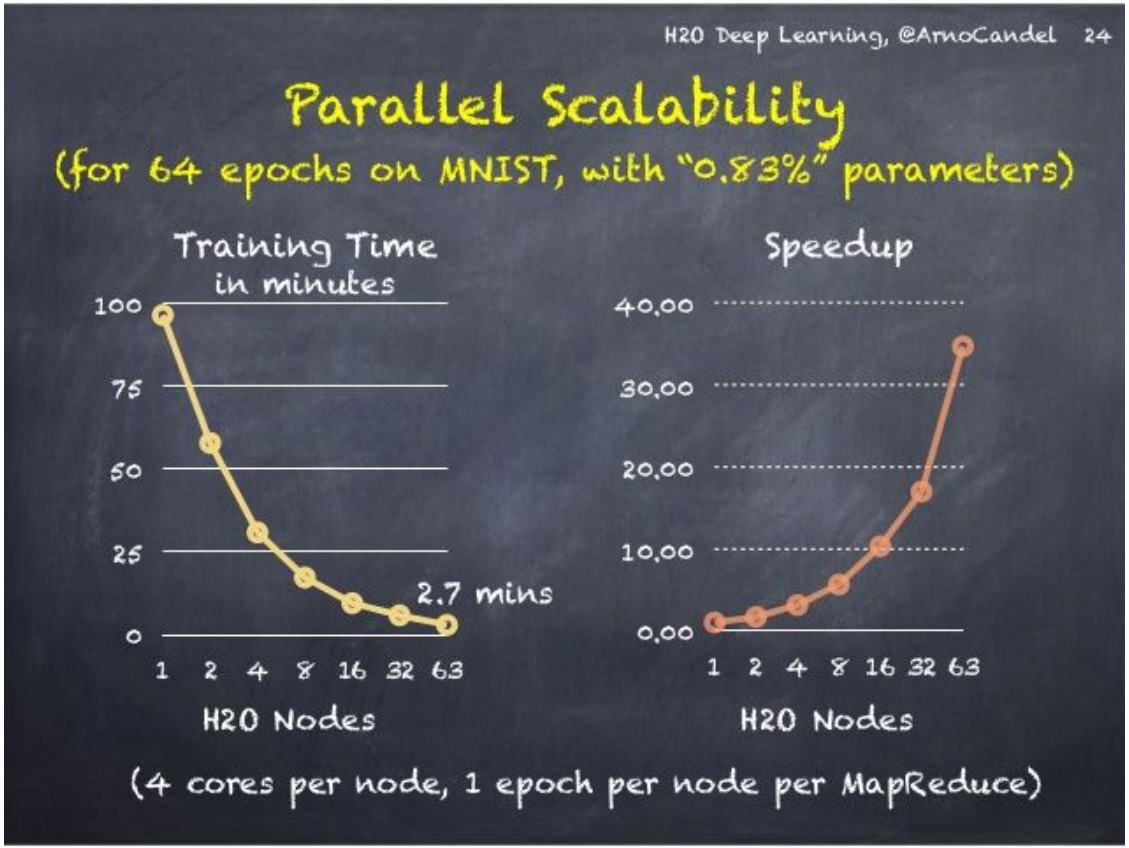
Solvers & Optimization

- Generalized ADMM Solver
- L-BFGS (Quasi Newton Method)
- Ordinary Least-Square Solver
- Stochastic Gradient Descent

Data Munging

- Integrated R-Environment
- Slice, Log Transform

H2O Scalability



Thank you