Big Data on HPC

Charles Peterson

Overview

In this workshop, we will using Big Data techniques with High-Performance Computing Resources

We will go over:

- General concepts of Big Data
- Simple examples on Hoffman2

Any suggestions for upcoming workshops, email me at cpeterson@oarc.ucla.edu

Files for this Presentation

This presentation can be found on our UCLA OARC's github repo

https://github.com/ucla-oarc-hpc/WS_BigDataOnHPC

View slides:

- PDF format:
  - BigDataHPC.pdf
- html format:
  - https://ucla-oarc-hpc.github.io/WS_BigDataOnHPC

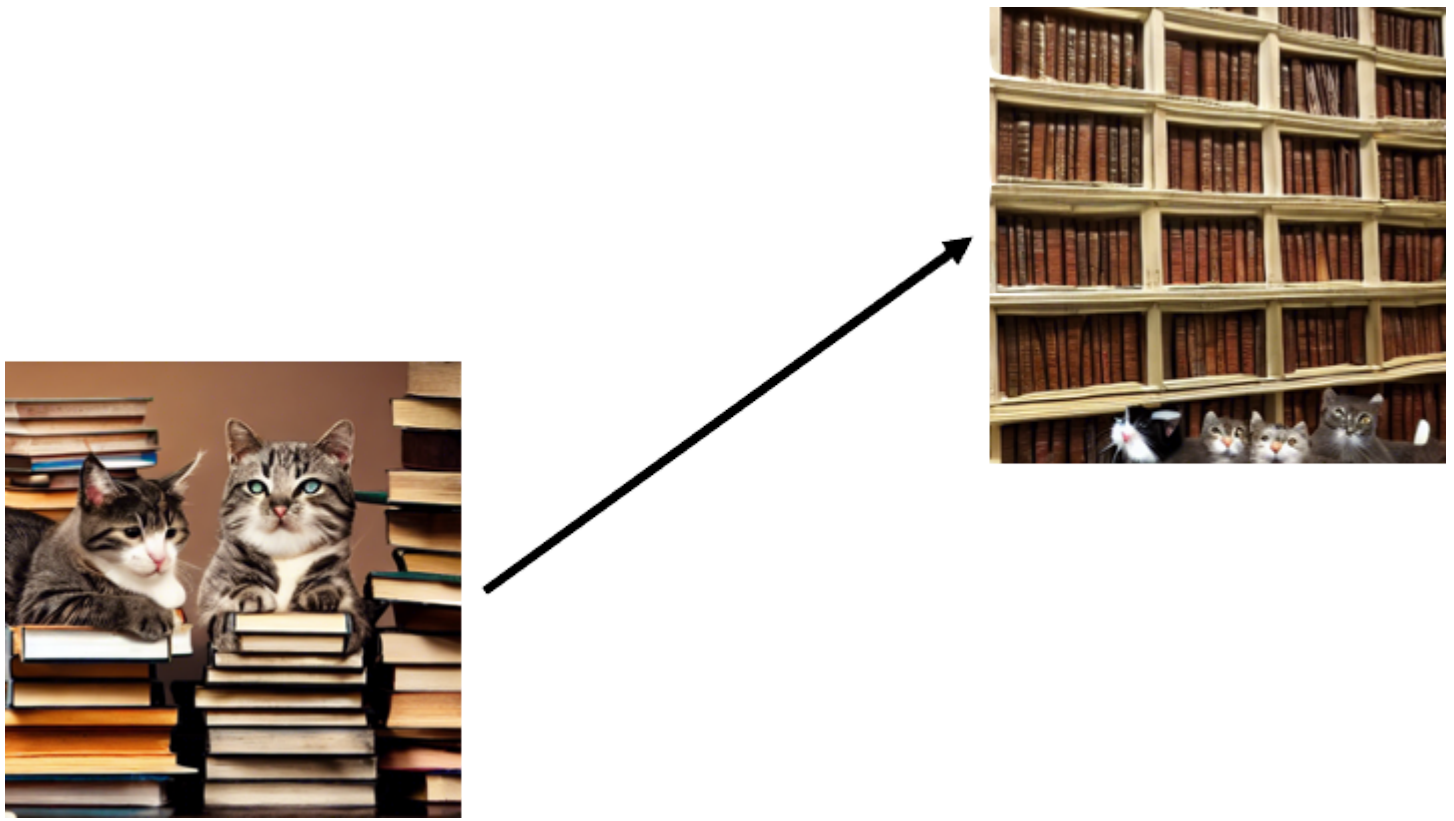
Note

This presentation was built with Quarto and RStudio.

- Quarto file: `BigDataHPC.qmd`

What is Big Data?

The term **Big Data** refers to data sets, data processing, data modeling, and other data science tasks that become too large and complex.



Traditional techniques are not enough to solve these projects.

Is Big Data for Me?

If you use data, then YES!

Big Data can solve problems for all types of research areas.

Big Data is used to scale up research.

Big Data

Problems that arise from increasing data projects

- Not enough RAM memory to fit data sets
- Data processing takes too long
  - Typical tasks (Pandas, SQL, etc.) cannot keep up with increasing data
- Machine Learning models become too complex
  - Training models may require computational extensive techniques for better accuracy

High-Performance Computing resources can help solve Big Data challenges by providing more computing power than typical workstations.

The 3 V's of Big Data

Big Data can be described by

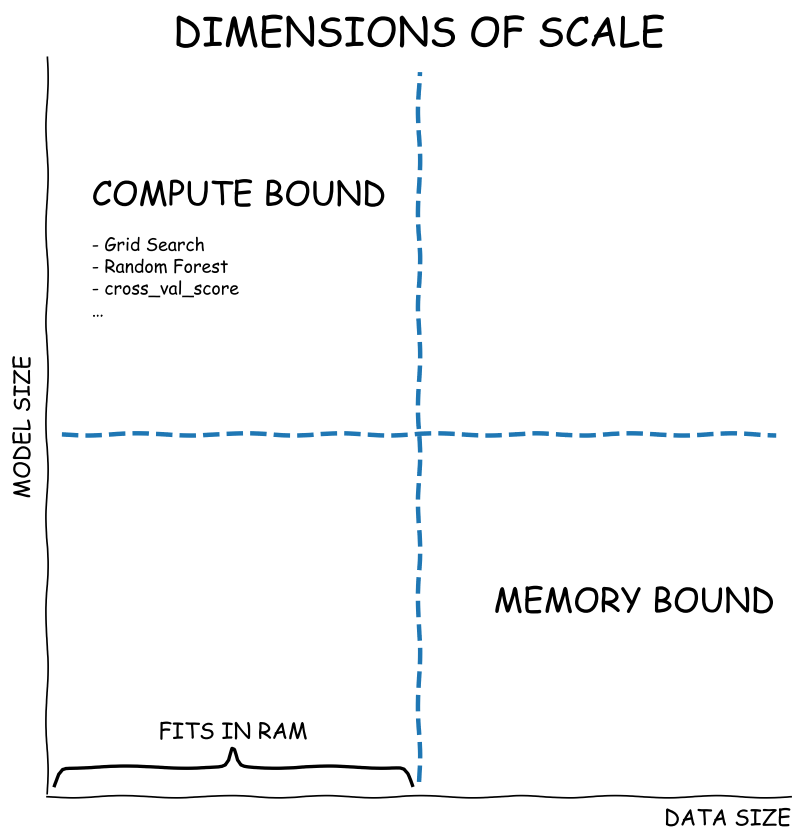
- Volume
  - The Size of data
- Velocity
  - Speed of the Data
  - Think about how fast (or slow) data is being transferred and processed.
  - Data being transferred over different servers.
  - Performing tasks in memory or on disk.
- Variety
  - Understanding the different types, sources, and nature of the data. Structured vs unstructured data.
  - Any preprocessing or data cleanup before any modeling can start.

Other V's

- Value
  - What is the 'worth' of information that can be extracted from the certain data.
- Veracity
  - The reliability and overall quality of the data. Missing Values (data imputation).
- Variability
  - ability of data to be used in changing formats, sources, and current data science methods. Raw and unstructured data can be difficult to modulate.

Understanding your data can help you choose what Big Data techniques to run.

Big Data Challenges



- Scaling Data Size
  - Datasets can become so large that they cannot fit into RAM
- Scaling Model/Task Size
  - Machine Learning (or other tasks) become so complex that 1 CPU core is not adequate

Big Data and HPC concepts

- Parallel computing
  - Typical python/R tasks are run on 1 CPU core
  - Performing tasks simultaneous over multiple CPUs can make code run faster
  - Many packages/libraries that can run tasks over multiple CPUs
    - Python https://wiki.python.org/moin/ParallelProcessing
    - R https://bookdown.org/rdpeng/rprogdatascience/parallel-computation.html

- Cluster computing
  - Multiple computer resources working together
  - HPC resources like Hoffman2 are great examples
    - Multiple compute nodes with access to many cores.

More concepts

- Symmetric Multiprocessing (SMP) programming
  - SMP sets up parallel tasks over a shared memory
  - Typically, these tasks can ONLY be ran on a single compute node
  - Uses a single process over multiple threads

- Cluster or Distributed programming
  - Distributes tasks over multiple processes
  - Allows tasks to be distributed over multiple compute nodes

Even more concepts

- Lazy evaluation
  - Delaying a task or expression until that resulting value is needed.
  - You can setup functions and calculations so they can be later run in parallel

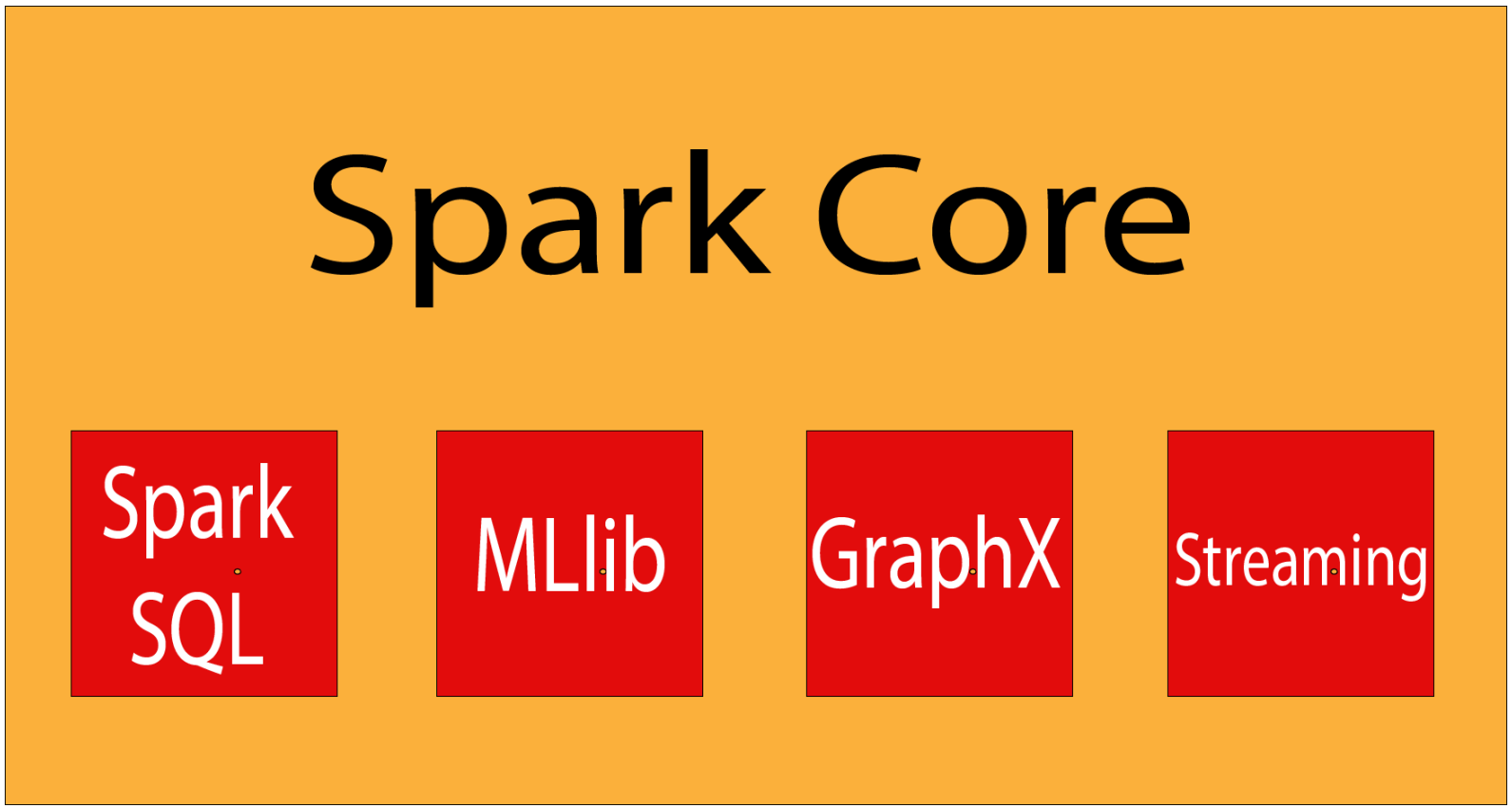
- in-memory computing
  - Performing tasks on data in the RAM (memory) of the computer
  - Data would be computed in RAM instead of disk in order to run faster

Big Data

Various frameworks, APIs, and libraries for Big Data projects

Spark

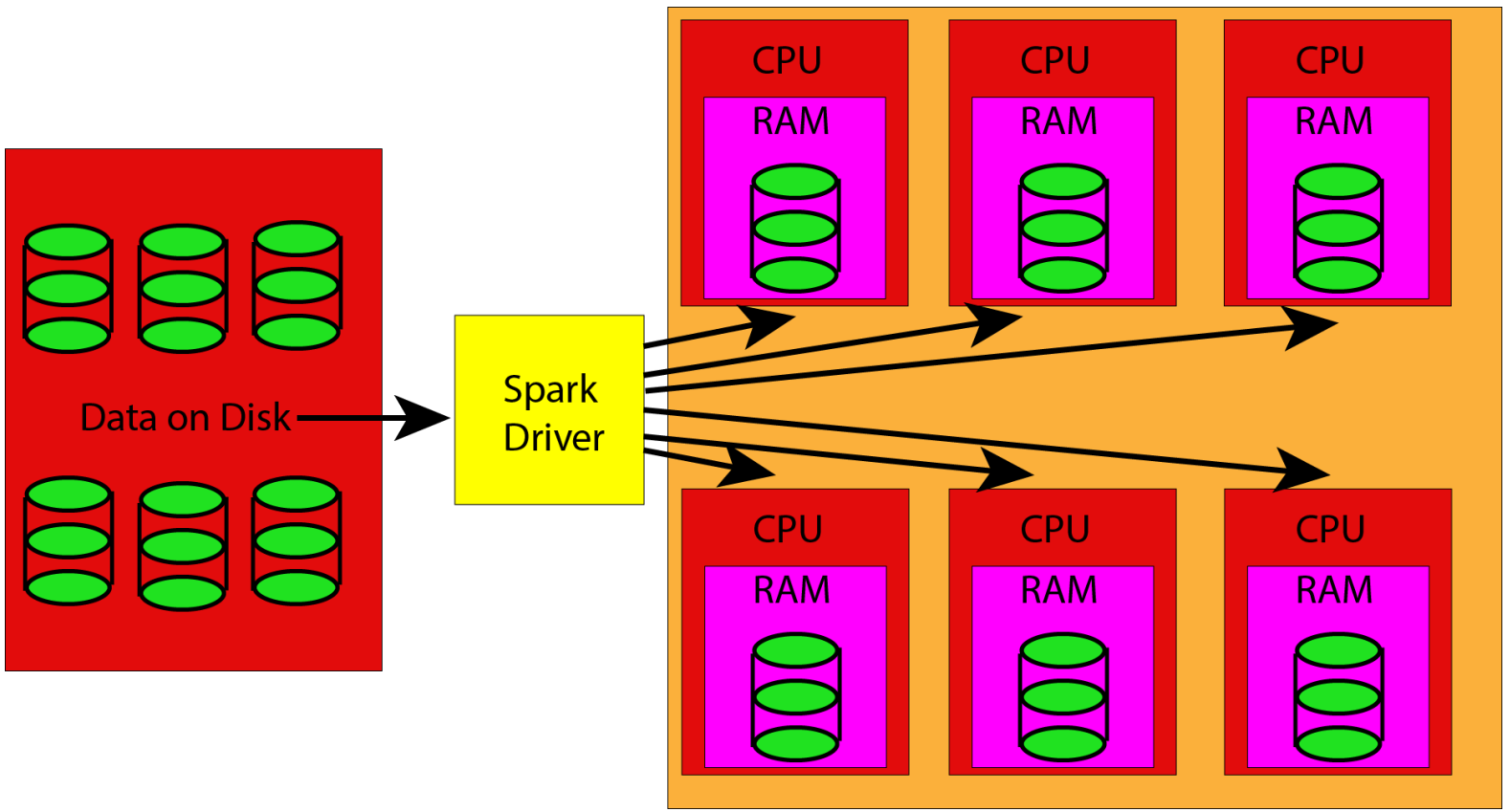
Spark Components



- Spark has APIs that can work with
  - Python (PySpark)
  - R (SparkR)

Spark Data

RDD - Resilient Distributed Dataset



Large Datasets can be spread over multiple compute nodes

Data Persistence

Spark supports different levels of persistence

- MEMORY_ONLY
  - All RDD data is in-memory
    - Need enough RAM to fit data
  - Highest RAM usage
  - Fastest CPU time
- MEMORY_AND_DISK
  - RDD data is stored in memory and disk
  - RDD data is moved in memory when needed
- DISK_ONLY
  - RDD data is stored only on Disk
  - Low RAM usage
  - High CPU time

Spark DataFrames

Along with RDDs, Spark also has a API DataFrame (Pandas-like)

This is a SQL-like library that can collect data with named columns

Great for structured/semi-structured data

Spark Session

The `SparkSession` is the entry point for Spark

- Underlying Spark functionality
- Creates the main Spark driver
- Used to create DataFrames and SQL-like tasks

```
spark = SparkSession.builder \
          .appName("MyPySpark") \
          .config("spark.driver.memory", "15g") \
          .getOrCreate()
```

The `SparkContext` is the entry point for RDD

- Created with the `spark` object from `SparkSession.builder`

```
sc = spark.sparkContext
```

Installing PySpark

Easiest way to install PySpark is by anaconda3.

This is great when running PySpark on a single compute node.

```
module load anaconda3/2022.05
conda create -n mypyspark openjdk pyspark python=3.9 \
                          pyspark=3.3.0 py4j jupyterlab findspark \
                          h5py pytables pandas \
                          -c conda-forge -y
conda activate mypyspark
pip install ipykernel
ipython kernel install --user --name=mypyspark
```

This will create a conda env named, mypyspark, with access to Jupyter

This conda env will have both Spark and PySpark installed

Note

Information on using Anaconda can be found from a previous workshop

https://github.com/ucla-oarc-hpc/H2HH_anaconda

PySpark: Basic operations

Let's go over basic PySpark functions

- `spark-ex1` from github repo

Make sure you download this Workshop on github in your Hoffman2 scratch directory

```
cd $SCRATCH
git clone https://github.com/ucla-oarc-hpc/WS_BigDataOnHPC
cd WS_BigDataOnHPC
cd spark-ex1
```

- Jupyter Notebook: `Spark_basics.ipynb`

In this example, we will use data from Project Gutenberg

We will download "The Hound of the Baskervilles", by Arthur Conan Doyle

```
wget https://www.gutenberg.org/files/3070/3070.txt
```

PySpark: Basic operations: Starting the notebook

We will use the `h2jupynb` script to start Jupyter on Hoffman2

You will run this on your LOCAL computer.

```
wget https://raw.githubusercontent.com/rdauria/jupyter-notebook/main/h2jupynb
chmod +x h2jupynb

#Replace 'joebruin' with you user name for Hoffman2
#You may need to enter your Hoffman2 password twice

python3 ./h2jupynb -u joebruin -t 5 -m 10 -e 2 -s 1 -a intel-gold\\* \
                   -x yes -d /SCRATCH/PATH/WS_BigDataOnHPC
```

Note

The `-d` option in the `python3 ./h2jupynb` will need to have the `$SCRATCH/WS_BigDataOnHPC` full PATH directory

This will start a Jupyter session on Hoffman2 with ONE entire intel-gold compute node (36 cores)

More information on the `h2jupynb` can be found on the Hoffman2 website

PySpark: Machine Learning

This example will use Spark's Machine Learning library (MLlib)

We will use data from the Million song subset

This subset has ~500,000 songs with:

- Year of the song
- 90 features relating to the timbre average and covariance of the song

Download the dataset

```
cd $SCRATCH/WS_BigDataOnHPC
cd spark-ex2
wget https://archive.ics.uci.edu/ml/machine-learning-databases/00203/YearPredictionMSD.tx
unzip YearPredictionMSD.txt.zip
```

We will use multiple nodes to run Spark

PySpark: Multi-node setup

In the previous `Basic operations` example, we used pyspark with 1 (36-core) compute node.

- We will run PySpark over multiple nodes.
- This will:
  - increase the number of cores
  - increase the available RAM to fit a large dataset

To do this, we will NOT use the Spark installation from our conda install, but use spark from a build that we will download from the spark website.

```
mkdir -pv $SCRATCH/WS_BigDataOnHPC/apps/spark
cd $SCRATCH/WS_BigDataOnHPC/apps/spark

wget https://dlcdn.apache.org/spark/spark-3.3.0/spark-3.3.0-bin-hadoop3.tgz
tar -vxf spark-3.3.0-bin-hadoop3.tgz
```

Note

Though we will not use the Spark from conda, we will still use the PySpark package that was install with conda. The Spark and PySpark packages will need to be the same version (3.3.0 in this example)

PySpark: Multi-node setup - Starting the notebook

Since we are using our Spark build that we just downloaded, we will start spark and submit it as a job, then connect to jupyter.

- Example: `spark-ex2`
  - Job script `pyspark-multi-jupyter.job`
  - Starts Spark and a Jupyter session for us to connect.
  - We start this with the line `$SPARK_HOME/bin/pyspark`

In this example, we will use 3 compute nodes in total.

- One compute node will be the master that will run the Spark driver.
- The other two will be workers that will run the tasks

> **Tip**

For large data jobs, I like to have the Spark driver to be separate from the workers.

Large data jobs may require the Spark driver to have a heavy CPU load and memory.

PySpark: Multi-node setup - Starting the notebook

- Submitting the Job

```
qsub pyspark-multi-jupyter.job
```

- Once the job starts, we will connect to this Jupyter session
  - The `spark-test.JOBID` file will display the MASTER node name
  - This master node will have the Spark driver and Jupyter process

Run this `ssh -L` command on your LOCAL computer

```
# Replace NODENAME with the name of the MASTER node
# Replace joebruin with you Hoffman2 user name
ssh -L 8888:NODENAME:8888 joebruin@hoffman2.idre.ucla.edu
```

- This will create a SSH tunnel to the master compute node so we can open Jupyter at `http://localhost:8888`
- Then we can open the notebook named `MSD.ipynb`

Spark dashboard

Spark has a visual dashboard that can view the tasks in real-time

- By default, Spark will run this dashboard on port 4040
- Create a ssh tunnel to the compute node to view the dashboard on your local machine

```
1 ssh -L 4040:NODENAME:4040 joebruin@hoffman2.idre.ucla.edu
```

You will need to replace NODENAME with the master compute node that has your Spark job

Spark batch job

You can run Spark as a batch job to run non-interactively.

- We will use the command `spark-submit` to start the pyspark calucation located at:
  - `$SPARK_HOME/bin/spark-submit`

```
1 qsub pyspark-multi-batch.job
```

Spark Bonus Example

I have another Machine Learning example for Spark that I may not have time to go over in this workshop.

In this example, we will train a Machine Learning model using data from LIBSVM

- Inside of `spark-bonus`
  - Jupyter notebook: `ML-bonus.ipynb`
- Download the dataset

```
cd $SCRATCH/WS_BigDataOnHPC
cd spark-ex2
wget https://raw.githubusercontent.com/apache/spark/master/data/mllib/sampl
```

Dask

Intro to Dask

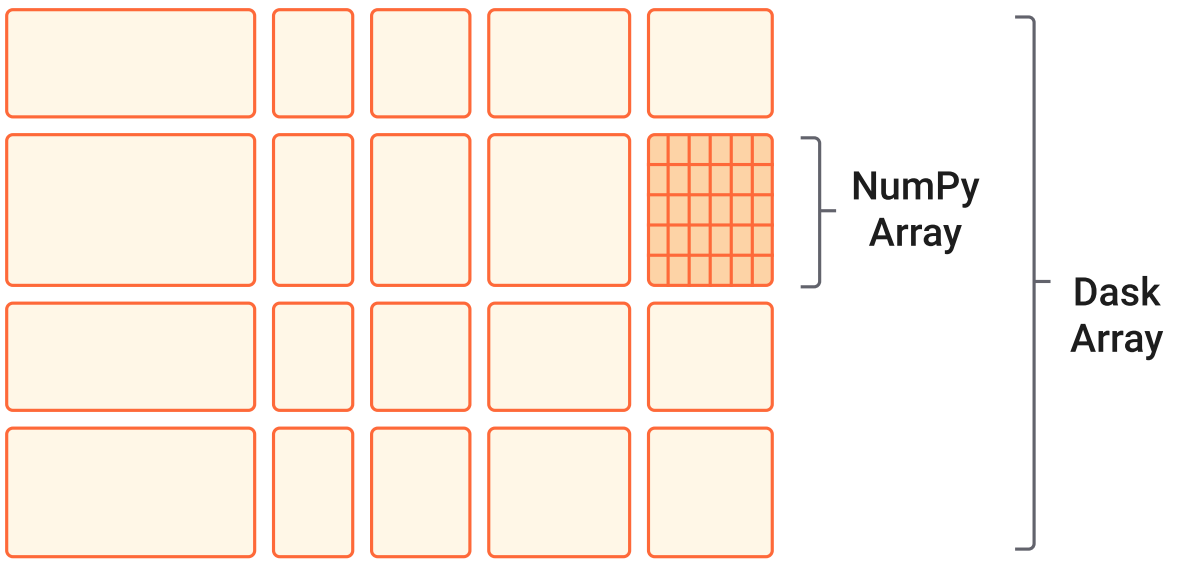
- Dask is a parallel computing library for Python
- Dask uses multiple cores to run tasks
- Can use GPUs to speed up tasks
- Very minimal changes to Python code is required for Dask
  - Dask Arrays and Dataframes are similar to Numpy and Pandas

Dask Array

Dask has an Arrays API created from NumPy-like chunks

Typically Numpy code

```
1  import numpy as np
2
3  numpy_array = np.ones((100))
4
5  print(np.average(numpy_array))
```



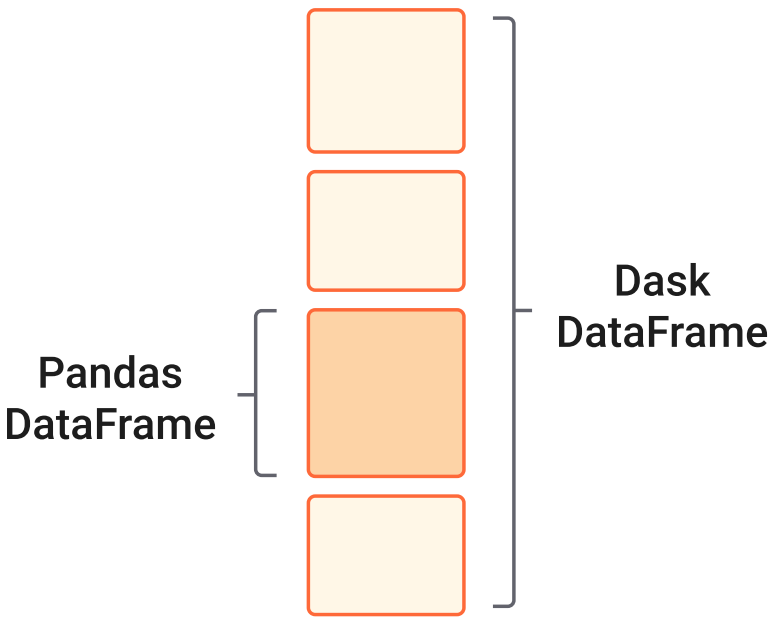
Dask code

```
1  import dask.array as da
2
3  dask_array = np.ones((100), chunks=(10))
4
5  print(da.average(dask_array).compute())
```

- Dask Arrays can process chuck over multiple cores.
- Great for larger than memory arrays. Chunks are compute in memory
- Dask Arrays have similar functions and methods as Numpy objects

Dask DataFrames

Dask DataFrames are Pandas-like objects



- Dask DataFrames are composed of Pandas-like “chunks”
- Can lazily read data files
  - CSV, hdf5, etc.

Dask installation

```
conda create -n mydask python pandas jupyterlab  joblib \
                        dask dask-ml nodejs graphviz python-graphviz
                        -c conda-forge -y
conda activate mydask
pip install ipykernel
ipython kernel install --user --name=mydask
```

This will create a conda env, mydask, that will have

- Dask
- Dask-ml (Machine Learning)
- Jupyter
- scikit-learn/joblib

Dask: Basic operations

We will use the `h2jupynb` script to start Jupyter on Hoffman2

You will run this on your LOCAL computer.

```
python3 ./h2jupynb -u joebruin -t 5 -m 10 -e 2 -s 1 -a intel-gold\\* -x yes \
                   -d /SCRATCH/PATH/WS_BigDataOnHPC
```

Replace `joebruin` with your Hoffman2 user account.

Replace `/SCRATCH/PATH/WS_BigDataOnHPC` with the full PATH name of the workshop on Hoffman2

- Let's go to `dask-ex1`
  - Jupyter notebook `dask_basic.ipynb`

Dask: Machine Learning

Dask has a `Dask-ML` library with scalable Machine Learning methods

There is also integration with:

- Scikit-Learn and Joblib
- XGBoost
- PyTorch
- Tensorflow and Keras

Dask ML Example

This example will use Scikit-Learn with Dask

We will use data from the Million song subset

- Download the dataset

```
cd $SCRATCH/WS_BigDataOnHPC
cd dask-ex2
wget https://archive.ics.uci.edu/ml/machine-learning-databases/00203/YearPredictionMSD.txt.zip
unzip YearPredictionMSD.txt.zip
```

- Start Jupyter notebook
  - Replace `joebruin` with your Hoffman2 user account
  - Replace `/SCRATCH/PATH/WS_BigDataOnHPC` with the full PATH name of the workshop on Hoffman2

```
python3 ./h2jupynb -u joebruin -t 5 -m 10 -e 2 -s 1 -a intel-gold\\* -x yes -d /SCRATCH/PATH/WS_BigDa
```

- Jupyter Notebook: `MSD-dask.ipynb`

Dask dashboard

Dask has a visual dashboard that can view the tasks in real-time

- By default, Dask will run this dashboard on port 8787

- Create a ssh tunnel to the compute node to view the dashboard on your local machine

```
ssh -L 8787:NODENAME:8787 joebruin@hoffman2.idre.ucla.edu
```

You will need to replace NODENAME with the compute node that has your Dask job

Wrap-up

Final thoughts

- This is a taste of what Big Data can do.
  - Too much to talk about in a hour

- Big Data is for EVERYONE!!!
  - Big Data can scale up your research
- Think about how your data is being computed
  - How is your data stored
  - CPU cores running your data in parallel
  - Memory considerations

Spark and Dask are just two popular frameworks for Big Data

Thank you!

Questions? Comments?

- cpeterson@oarc.ucla.edu
- Look at for more Hoffman2 workshops at https://idre.ucla.edu/calendar