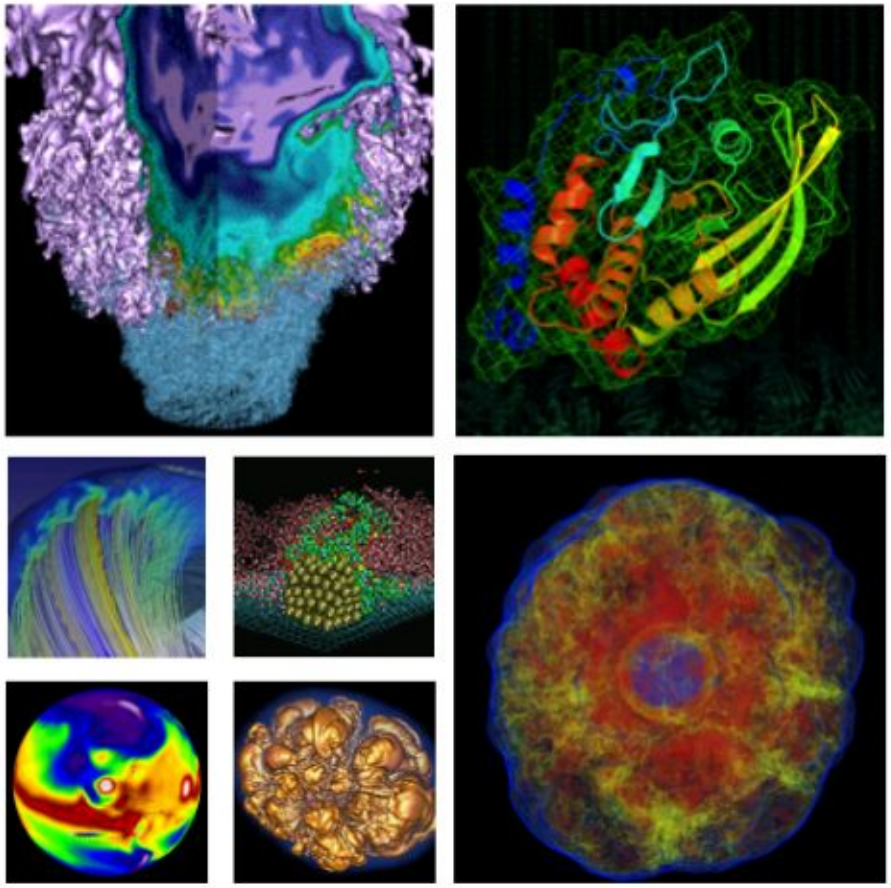
Using Spark on Cori



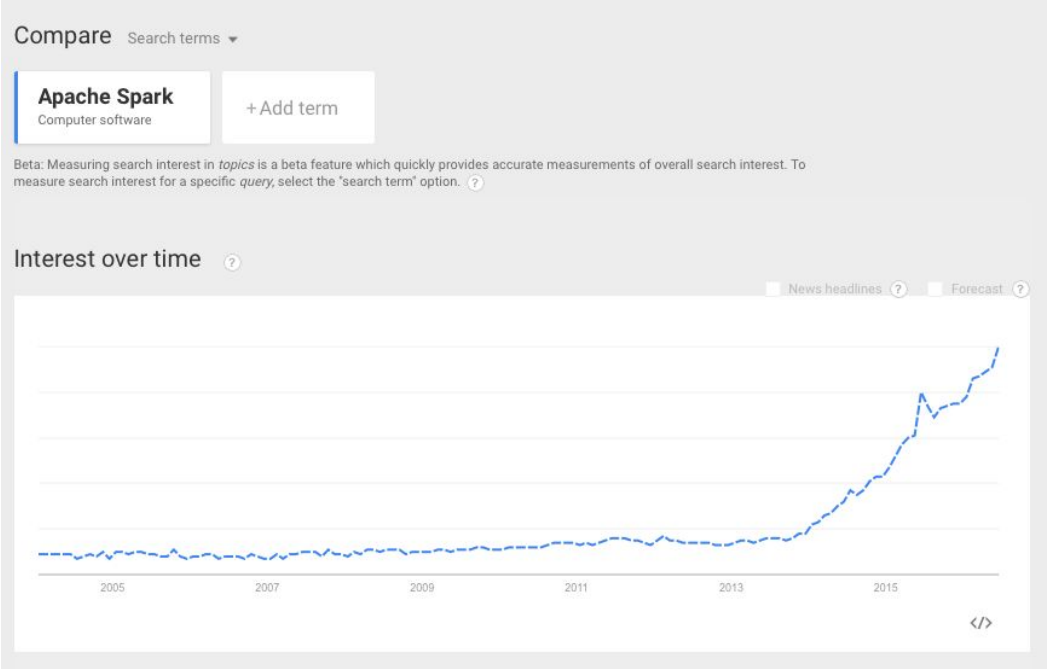
Lisa Gerhardt, Evan Racah
DAS Data Day

What is Spark?

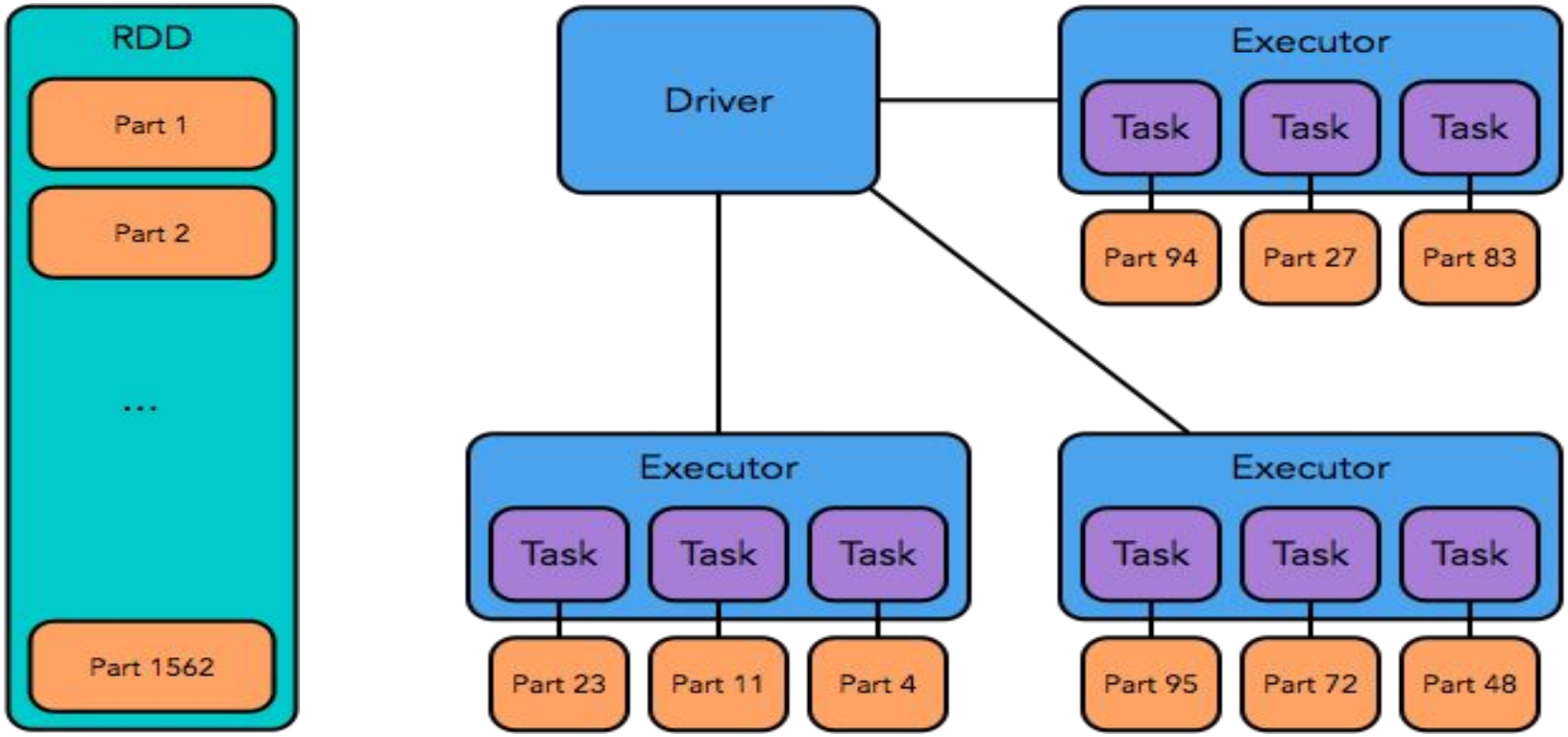
- Apache Spark is a fast and general engine for large-scale data processing
- **Data-parallel model**
  - Similar to Hadoop, except it harnesses in-memory data for fast data processing
- Robust body of libraries: machine learning, sql, graphx, streaming
- Created at the AMPLab in UCB, currently maintained by DataBricks, more than 400 contributors to git repository
- **Requires no parallel programming expertise**: Can take Python, Java, or Scala code and automatically parallelize it to an arbitrary number of nodes

Spark: So hot right now

- Spark Summit in June had more than 2500 attendees
- Used by Airbnb, amazon, eBay, Groupon, IBM, MyFitnessPal, NetFlix, OpenTable, Salesforce, Uber, etc.
- And by NASA (SETI), UCB (ADAM)
- Used at NERSC for metagenome analysis, astronomy, and business intelligence



Spark Architecture



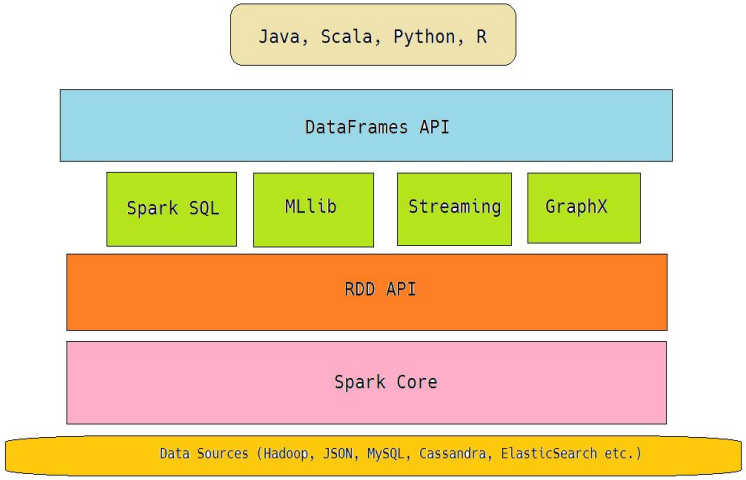
- One driver process, many executors
- Driver creates DAG and coordinates data and communication (via akka TCP)
- Datasets generally cached in memory, but can spill to local disk

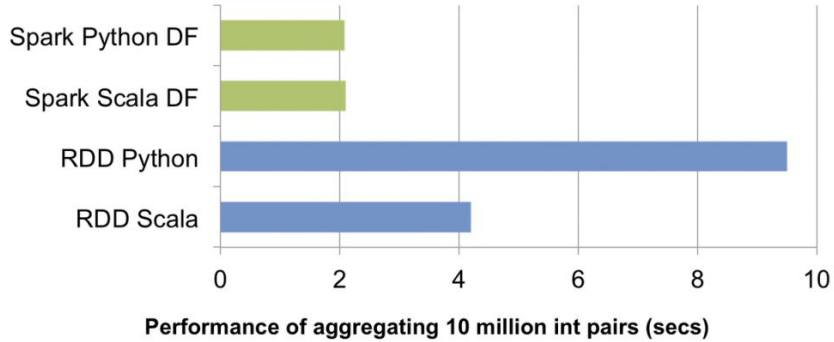
Spark API

- Can use Java, Python, or Scala
- Latest version also has an R interface and a Jupyter notebook
- Most newer functionality comes to Scala interface first
- Also some indications that Scala performs better
- Python is easier to use, so we recommend using that to start out
  - Can always reoptimize in scala later if necessary
- Default version at NERSC is Spark 2.0.0
- Recommend running on Cori
  - Larger memory nodes and faster connections with scratch file system

Spark Data Formats

- **RDD**: Resilient Distributed Dataset. Immutable collection of **unstructured** data.
- **DataFrames**: same as RDD but for **structured** data. Familiar pandas-like interface
- Which to use?
  - Future: DataFrames
  - Now: DataFrames and fall back to RDDs for more fine tuned control





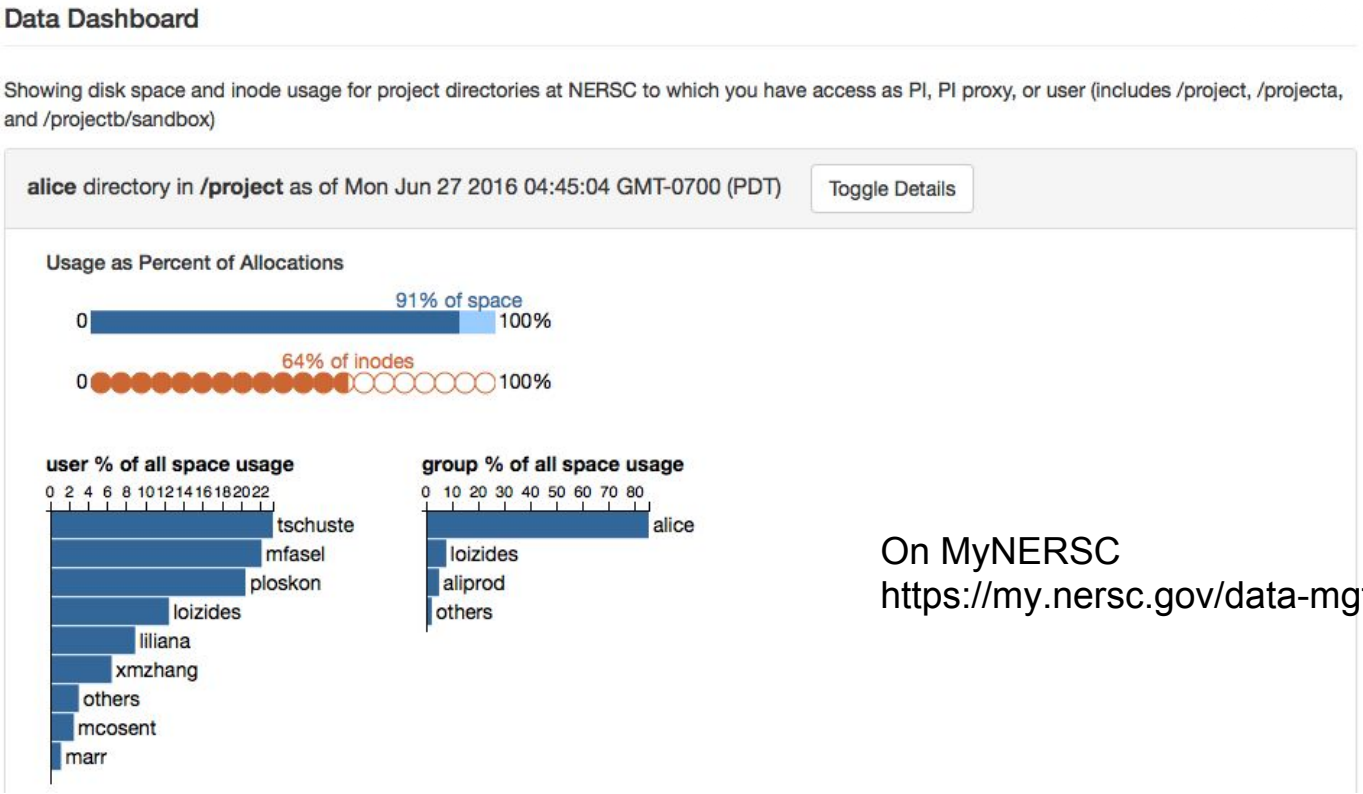
Spark Goodies

- **Streaming**: high-throughput, fault-tolerant stream processing of live data. Twitter, kafka, etc.
- **SQL**: Structured data processing, lives on top of RDD API
- **ML**: classification, regression, clustering, collaborative filtering, feature extraction, transformation, dimensionality reduction, and selection. Best only for really LARGE datasets. Demo tomorrow
- **GraphX**: Graph parallel computation. Extends RDD to a directed multigraph with properties attached to each vertex and edge

Spark Strategy

- **Best for DATA PARALLEL!!!**
- **Make sure you have enough memory**
  - Datasets are held in memory for speed
  - When memory is filled up, must spill data to disk, which slows things considerably
    - NERSC default spills to RAM file system and to Lustre scratch
  - If you get “java.lang.OutOfMemoryError” your executors have run out of memory. Fix by making the load on each node smaller. Either
    - Increase the number of nodes **OR**
    - Increase executor memory
- **Keep the cores fed**
  - Optimal number of partitions is 2 - 3 times the number of cores, automatic partitioning sizes can be a little wonky

Real World Use Case: Data Dashboard



On MyNERSC
https://my.nersc.gov/data-mgt.php

200 GB of text

Conclusion

- Spark is an excellent tool for data parallel tasks that are too big to fit on a single node
  - Easy to use python DataFrames interface
  - Does parallel heavy lifting for you
- Spark 2.0 installed at NERSC, performs well on data-friendly Cori system
- Come to the hack session tomorrow for more Spark demos!

National Energy Research Scientific Computing Center

Spark Batch Job

```
#!/bin/bash

#SBATCH -p debug
#SBATCH -N 2
#SBATCH -t 00:30:00
#SBATCH -e mysparkjob_%j.err
#SBATCH -o mysparkjob_%j.out
#SBATCH --ccm

module load spark

start-all.sh
spark-submit --master $SPARKURL --driver-memory 15G --executor-memory
20G ./testspark.py
stop-all.sh
```

Submit with sbatch <submitscriptname.slurm>

Spark Terminology

- **Driver process**: runs on head node and coordinates workers (you can adjust memory allocation)
- **Executor**: single process that runs on worker nodes
- **Task**: a unit of work that's sent to one executor
- **Application**: entire program that is run
- **Job**: spawned by a spark action (like *collect*), consists of one or more stages on multiple nodes
- **Stage**: a set of tasks that do the same operation on a different slice of the data
  - See frequent references to "job" and "stage" in the spark logs
- **RDD** (Resilient Distributed Datasets): read only collection of records, held in memory or spilled out to disk