Argonne NATIONAL LABORATORY

Getting Started with Spark on Theta

Xiao-Yong Jin

Oct 3, 2019
ALCF Simulation, Data, and Learning Workshop

Spark or MPI?

- Use MPI or even lower APIs to get the absolute performance

- Use Apache Spark if human time > machine time
  - Ease of use: parallelize quickly in Java, Scala, Python, R, and SQL
  - Built-in libraries: SQL and DataFrames, MLlib for machine learning, GraphX, and Spark Streaming

PySpark examples on Theta

/projects/SDL_Workshop/training/GettingStartedWithSparkOnTheta

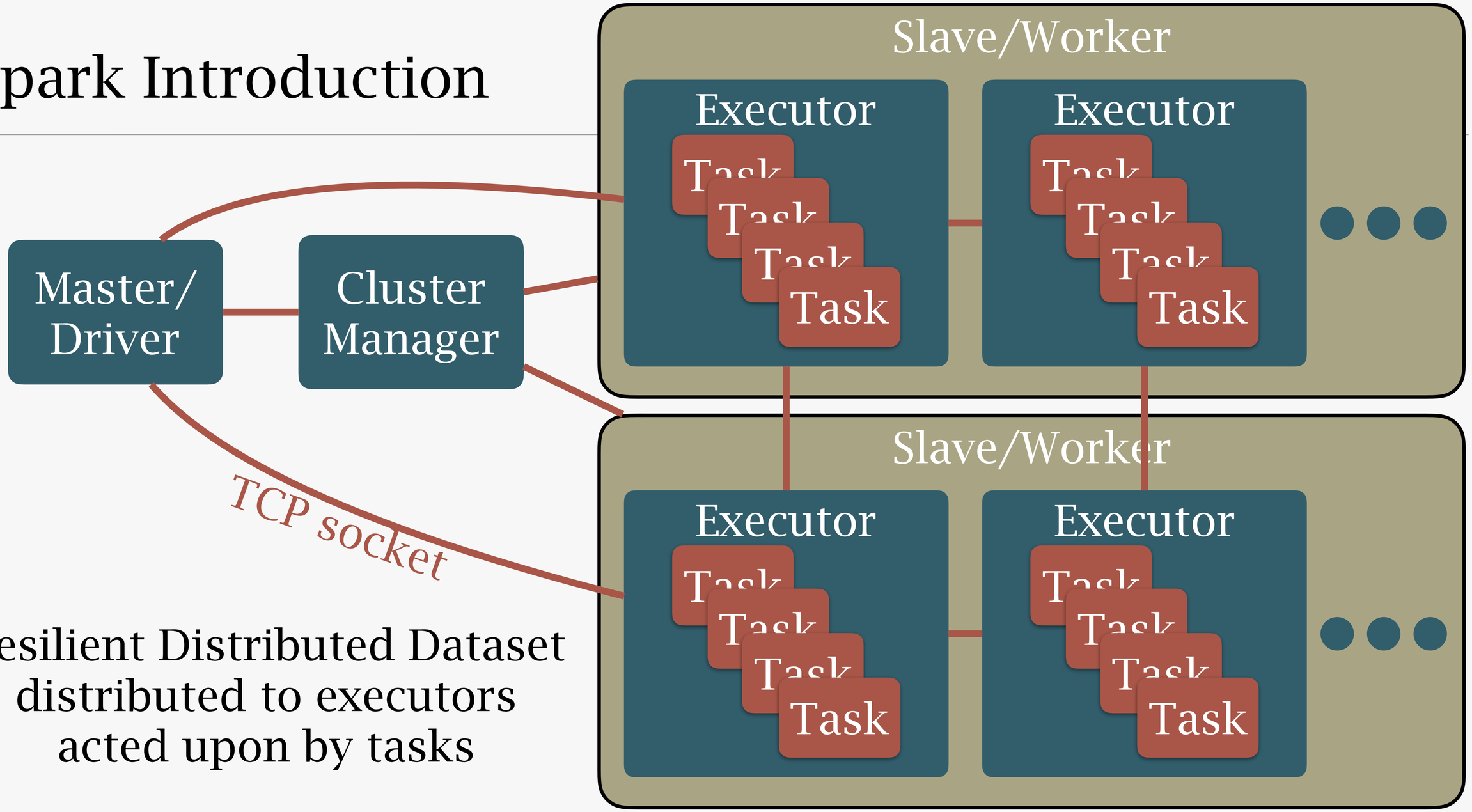
Getting Started with Spark on Theta (noninteractive)

```
/soft/datascience/Spark_Job/submit-spark.sh \
     -A SDL_Workshop -t 10 -n 2 -q training \
     run-example SparkPi

/soft/datascience/Spark_Job/submit-spark.sh \
     -A SDL_Workshop -t 10 -n 2 -q training \
     --class YOUR.SPARK.APP.CLASS \
     local:///ABSPATH/TO/YOUR/SPARK/APP.jar [EXTRA_ARGS ...]

/soft/datascience/Spark_Job/submit-spark.sh \
     -A SDL_Workshop -t 10 -n 2 -q training \
     PATH/TO/YOUR/PYSPARK/SCRIPT.py [EXTRA_ARGS ...]
```

Getting Started with Spark on Theta (Jupyter)

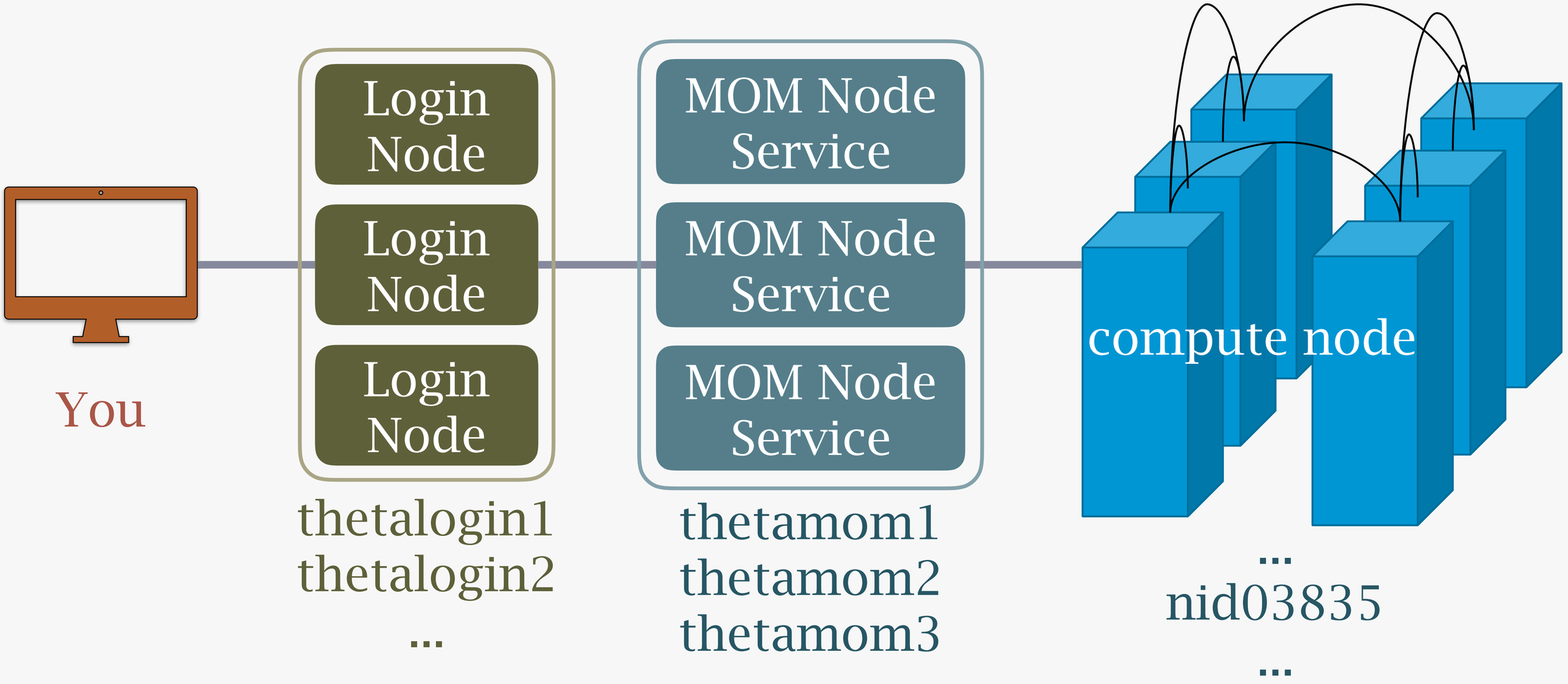
```
thetalogin$ /soft/datascience/Spark_Job/submit-spark.sh \
            -A SDL_Workshop -t 60 -n 2 -q training -I
...
SPARKJOB_JOBID=325700
...
# Spark is now running (SPARKJOB_JOBID=325700) on:
# nid03835      nid03836
declare -x SPARK_MASTER_URI="spark://nid03835:7077"
# Spawning bash on host: nid03835
...
nid03835$ export PYSPARK_DRIVER_PYTHON=jupyter
nid03835$ export PYSPARK_DRIVER_PYTHON_OPTS="notebook --no-browser --
ip=nid03835 --port=8008"
nid03835$ /soft/datascience/apache_spark/bin/pyspark \
          --master $SPARK_MASTER_URI

local$ ssh -L 8008:localhost:8008 theta ssh -L 8008:nid03835:8008 thetamom1
```

Spark Introduction



Resilient Distributed Dataset
distributed to executors
acted upon by tasks

Theta Reminder



Spark Job (Script for working with COBALT)

- Installed under `/soft/datascience/Spark_Job`
- Designed to minimize the changes required for deploying on Theta
- Check out the readme file: `/soft/datascience/Spark_Job/readme`
- Look in the example directory: `/soft/datascience/Spark_Job/example`
- Under heavy development, guaranteed interface: `submit-spark.sh`
- Absolute stability, use explicit version number, eg: `/soft/datascience/Spark_Job_v1.1.0`

Spark Job [submit-spark.sh] usage

```
submit-spark.sh [options] [JOBFILE [arguments ...]]

JOBFILE (optional) can be:
    script.py           pyspark scripts
    bin.jar             java binaries
    run-example CLASS   run spark example CLASS
    scripts             other executable scripts (requires `-s`)

Required options:
    -A PROJECT          Allocation name
    -t WALLTIME         Max run time in minutes
    -n NODES            Job node count
    -q QUEUE            Queue name

Optional options:
    -o OUTPUTDIR        Directory for COBALT output files (default: current dir)
    -s                  Enable script mode
    -m                  Master uses a separate node
    -p <2|3>            Python version (default: 3)
    -I                  Start an interactive ssh session
    -w WAITTIME         Time to wait for prompt in minutes (default: 30)
    -h                  Print this help message
```

Environment Variables (Information)

- The scripts set a few environment variables for informational purposes, and for controlling the behavior.

- Information (taken from the command line, the job scheduler, the system):

```
SPARKJOB_HOST="theta"
SPARKJOB_INTERACTIVE="1"
SPARKJOB_JOBID="242842"
SPARKJOB_PYVERSION="3"
SPARKJOB_SCRIPTMODE="0"
SPARKJOB_SCRIPTS_DIR="/lus/theta-fs0/projects/datascience/xyjin/Spark_Job"
SPARKJOB_SEPARATE_MASTER="0"
SPARKJOB_OUTPUT_DIR="/lus/theta-fs0/projects/datascience/xyjin/Spark_Job/example"
SPARK_MASTER_URI=spark://nid03838:7077
MASTER_HOST=nid03838
```

Environment Variables (Customizable)

```
SPARK_HOME="/soft/datascience/apache_spark"
SPARK_CONF_DIR="/lus/theta-fs0/projects/datascience/xyjin/Spark_Job/example/242842/conf"
PYSPARK_PYTHON="/opt/intel/python/2017.0.035/intelpython35/bin/python"
SPARKJOB_WORKING_DIR="/lus/theta-fs0/projects/datascience/xyjin/Spark_Job/example/242842"
SPARKJOB_WORKING_ENVS="/lus/theta-fs0/projects/datascience/xyjin/Spark_Job/example/242842/envs"
SPARKJOB_DELAY_BASE=15
SPARKJOB_DELAY_MULT=0.125
```

- The above is the environment set up when running a job under `OUTPUTDIR /projects/datascience/xyjin/Spark_Job/example`
- The variable `SPARKJOB_OUTPUT_DIR` contains the directory path, which `SPARKJOB_WORKING_DIR` and `SPARKJOB_WORKING_ENVS` depend on
- `SPARKJOB_DELAY_BASE` and `SPARKJOB_DELAY_MULT` controls how much time in seconds we wait until starting the Spark slave processes.

Customizable Variables in `env_local.sh`

- See `/soft/datascience/Spark_Job/example/env_local.sh`
- You can use SPARKJOB_HOST to detect the running system.

```
if [[ $SPARKJOB_HOST == theta ]];then
   module rm intelpython36
   module load miniconda-3
   export PYSPARK_PYTHON="$(which python)"
fi
```

- On Cooley, interactive Spark jobs setup IPython notebook by defaults. You can change it here, along with setting up your other python environment.

```
unset PYSPARK_DRIVER_PYTHON
unset PYSPARK_DRIVER_PYTHON_OPTS
```

Customizable Variables in `env_local.sh`

- Create `spark-defaults.conf` file affecting Spark jobs submitted under the current directory where this file resides, c.f. `$SPARK_CONF_DIR`

- The parameters require tuning depending on the machine and workload.

```
[[ -s $SPARK_CONF_DIR/spark-defaults.conf ]] ||
   cat > "$SPARK_CONF_DIR/spark-defaults.conf" <<'EOF'
spark.task.cpus                    4
spark.driver.memory               32g
spark.executor.memory            128g
spark.driver.extraJavaOptions   -XX:+UseParallelGC -XX:ParallelGCThreads=8
spark.executor.extraJavaOptions -XX:+UseParallelGC -XX:ParallelGCThreads=8
EOF
```

Spark on Theta

- Don't run Spark on the MOM node!
- Should the master share one node with the slaves?
- How many workers per node?
- How many executors per worker?
- How many tasks per executor?
- Is thread affinity useful?
- It all depends on your workload.

Tuning parameters (spark-defaults.conf)

Tune these numbers for your workload

```
spark.task.cpus                                          4
spark.rpc.netty.dispatcher.numThreads                    8

spark.scheduler.maxRegisteredResourcesWaitingTime     4000s
spark.scheduler.minRegisteredResourcesRatio              1
spark.scheduler.listenerbus.eventqueue.capacity     100000

spark.worker.timeout                                 24000
spark.executor.heartbeatInterval                      4000s
spark.files.fetchTimeout                             12000s
spark.network.timeout                                24000s
spark.locality.wait                                   6000s

spark.driver.memory                                     16g
spark.executor.memory                                  128g

spark.driver.extraJavaOptions   -XX:+UseParallelGC -XX:ParallelGCThreads=8
spark.executor.extraJavaOptions -XX:+UseParallelGC -XX:ParallelGCThreads=8
```

Tuning parameters (spark-defaults.conf)

Tune these numbers for your workload

```
spark.task.cpus                           4
spark.rpc.netty.dispatcher.numThreads     8
```

- JVM sees 256 cores on each Theta node
- By default, JVM launches 256 tasks simultaneously if memory allows
- `spark.task.cpus` makes JVM count each task as using 4 cores
- `spark.rpc.netty.dispatcher.numThreads` limits the netty thread pool
- Applies for PySpark applications, too
- More thread tuning: [SPARK-26632][Core] Separate Thread Configurations

Tuning parameters (spark-defaults.conf)

Tune these numbers for your workload

```
spark.scheduler.maxRegisteredResourcesWaitingTime   4000s
spark.scheduler.minRegisteredResourcesRatio            1
```

- Wait for resources on-line to avoid performance impact in the beginning
- Depends on your resource usage

```
spark.scheduler.listenerbus.eventqueue.capacity   100000
```

- If you see related warnings
- It happens if you use large amount of nodes

Tuning parameters (spark-defaults.conf)

Tune these numbers for your workload

```
spark.worker.timeout                    24000
spark.executor.heartbeatInterval         4000s
spark.files.fetchTimeout                12000s
spark.network.timeout                   24000s
```

- Extra overhead compared to your MPI programs
- Impacts the FAULT TOLERANCE capability

```
spark.locality.wait                      6000s
```

- Transferring data over the network incurs a larger overhead than waiting

Tuning parameters (`spark-defaults.conf`)

Tune these numbers for your workload

```
spark.driver.memory              16g
spark.executor.memory           128g
```

- You absolutely must set these to some large number
- The default `1g` is too small unless you run multiple workers/executors

Tuning parameters (spark-defaults.conf)

Tune these numbers for your workload

```
spark.driver.extraJavaOptions   -XX:+UseParallelGC -XX:ParallelGCThreads=8
spark.executor.extraJavaOptions -XX:+UseParallelGC -XX:ParallelGCThreads=8
```

- Depending on you application
- Tuning GC is another work of art
- Make sure GC time does not dominate

Access the Web Interface

- Find the driver node ID, `nid0NNNN`
- Use SSH LocalForward

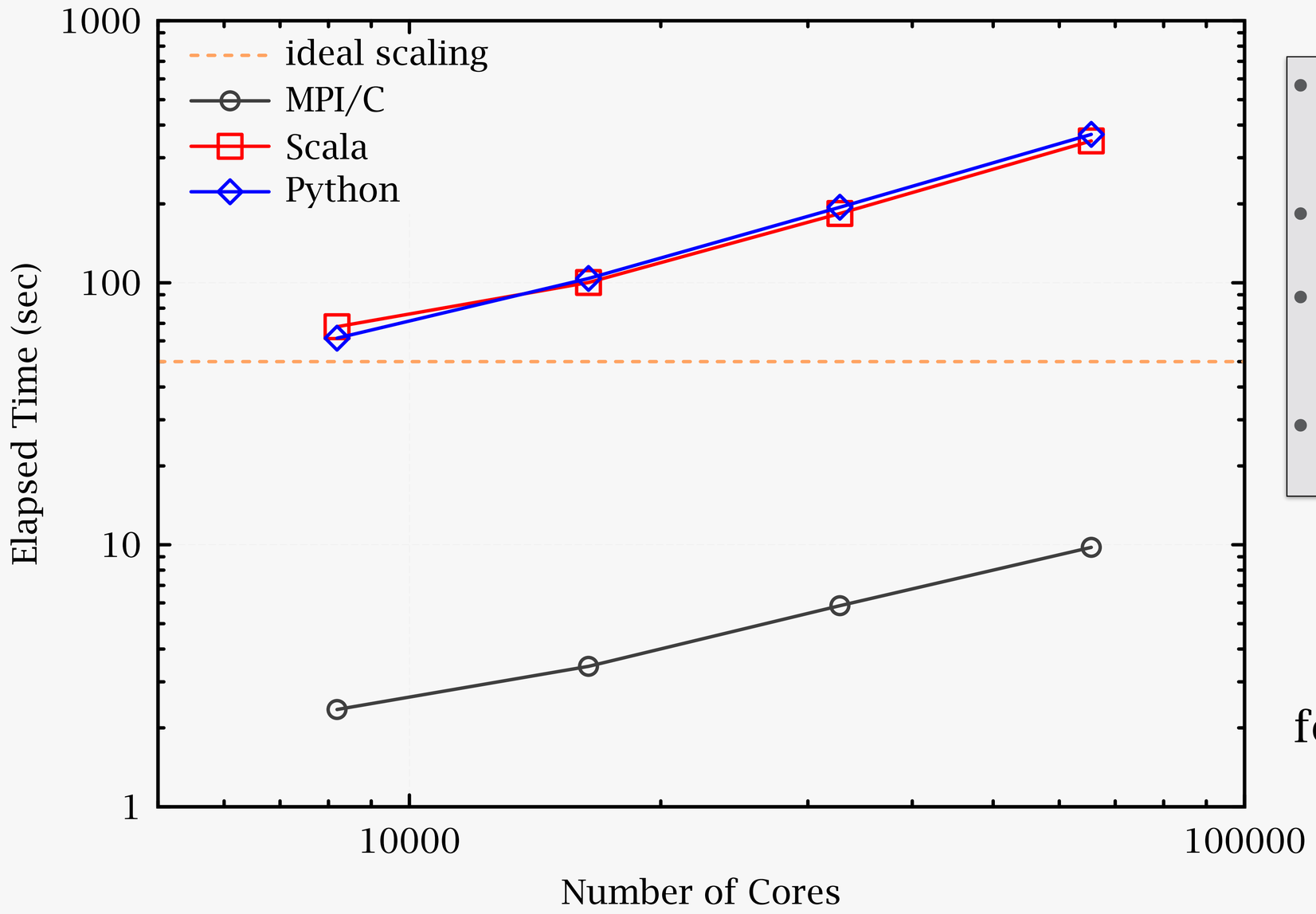
```
ssh -L 8080:localhost:8080 -L 4040:localhost:4040 -t theta \
  ssh -L 8080:nid0NNNN:8080 -L 4040:nid0NNNN:4040 thetamom1
```

- Go to http://localhost:8080 on your local machine

Other things to consider

- Number of partitions for your RDD
- Point `spark.local.dir` to the local SSD
- Do not use "Dynamic Allocation" unless you have a strong reason
- Beyond the scope of this presentation: shuffle, other cluster managers, etc.
  - Please contact us
  - We are interested in Spark usage in scientific applications

Overhead Dominated Weak Scaling (Preliminary)



- https://arxiv.org/abs/1904.11812
- https://github.com/SparkHPC
- Memory bandwidth limited operation
- No shuffle, no disk, minimal network

$$\frac{1}{N_{\text{block}} S_{\text{block}}} \sum_{n=1}^{N_{\text{block}}} \sum_{s=1}^{S_{\text{block}}} \left(V_{n,s,i} + C_i \right),$$

for $i \in \{0,1,2\}$ and V_n is an RDD


DON'T PANIC
xjin@anl.gov