GPU Support in Spark and GPU/ CPU Mixed Resource Scheduling at Production Scale

Yonggang Hu, IBM, DE
Junfeng Liu, IBM, Architect

SPARK SUMMIT 2016
DATA SCIENCE AND ENGINEERING AT SCALE

About us

- Yonggang Hu

  Distinguished Engineer, IBM

  Chief Architect at Platform Computing, IBM.

  Vice President and Application Architect at JPMorgan Chase

  Working on distributed computing, grid, cloud and big data for the past 20 years.

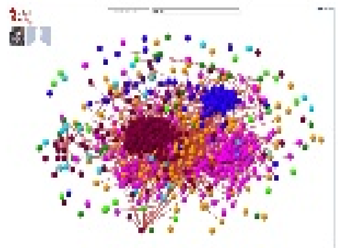
- Junfeng Liu

  IBM Platform Computing Architect, focusing on Big data platform design and implementation. Successfully delivering solutions to several key customers.

Agenda

- GPU and Spark integration motivation
- The challenges in production deployments
- Solutions in IBM Conductor with Spark
- Demo

Spark & GPU



Spark MLlib

Graph Analytics
Security, Fraud Detection
Social Network Analytics
GraphX

Machine Learning
Predicative analytics,
Logistic regression, ALS
Kmeans, etc.

Financial Risk Analytics
Market simulation
Credit risk. home-grown,
apps from Murex, Misys

Video/Speech Analytics
Object Recognition
Dialog Caffe theano torch

Spark

Spark apps are CPU intensive

Need to handle more data and bigger models

GPU-enable Spark apps

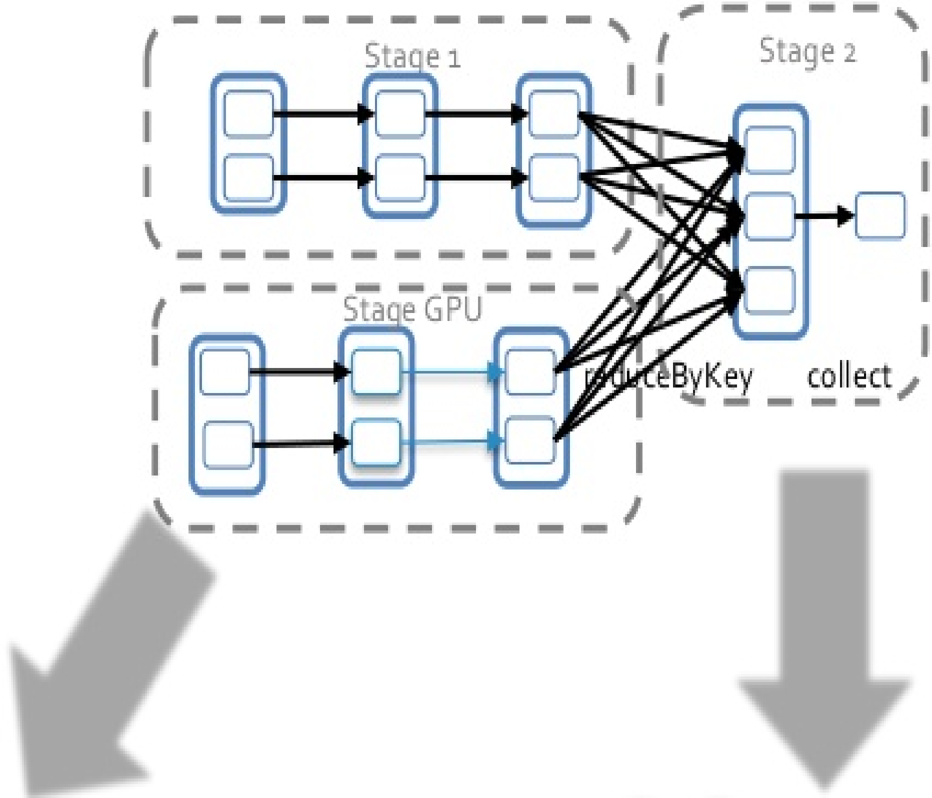
Spark-enable existing GPU apps

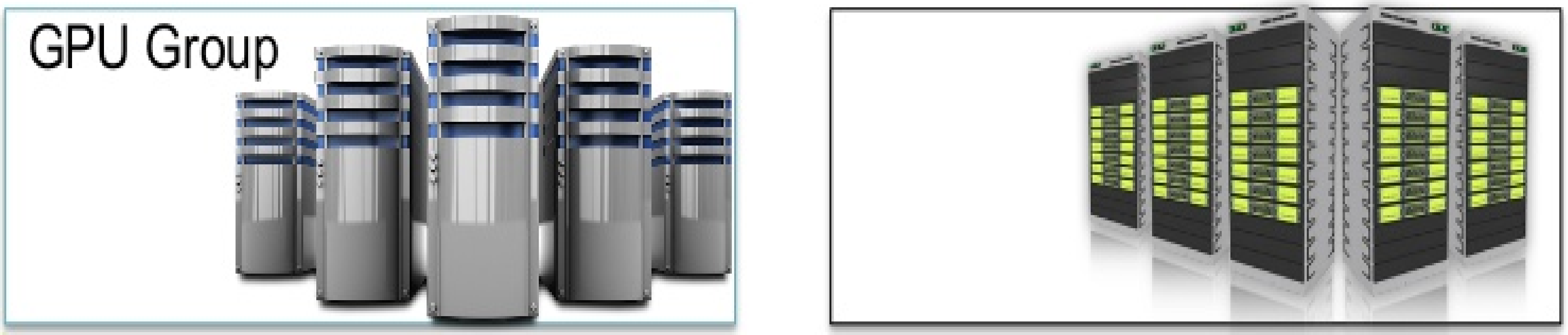
Various ways to enable Spark & GPU

- Use GPUs for accelerating Spark Libraries and operations without changing interfaces and underlying programming model.
- Automatically generate CUDA code from the source Spark Java code
- Integrate Spark with GPU-enabled application & system (e.g., Spark integrated with Caffe, TensorFlow and customer applications)

Production Challenges

- However
  - Identification of GPU execution vs. CPU execution in DAG
  - Data preparation for GPU execution
  - Low resource utilization for CPU or GPU or both
    - Cannot assume all compute hosts are identical and have GPU resource available
    - **GPU is a lot more expensive !!!**
  - Overload and contention when running mixed GPU & CPU workload
  - Long tail & GPU & CPU tasks failover
  - Task ratio control on different resources

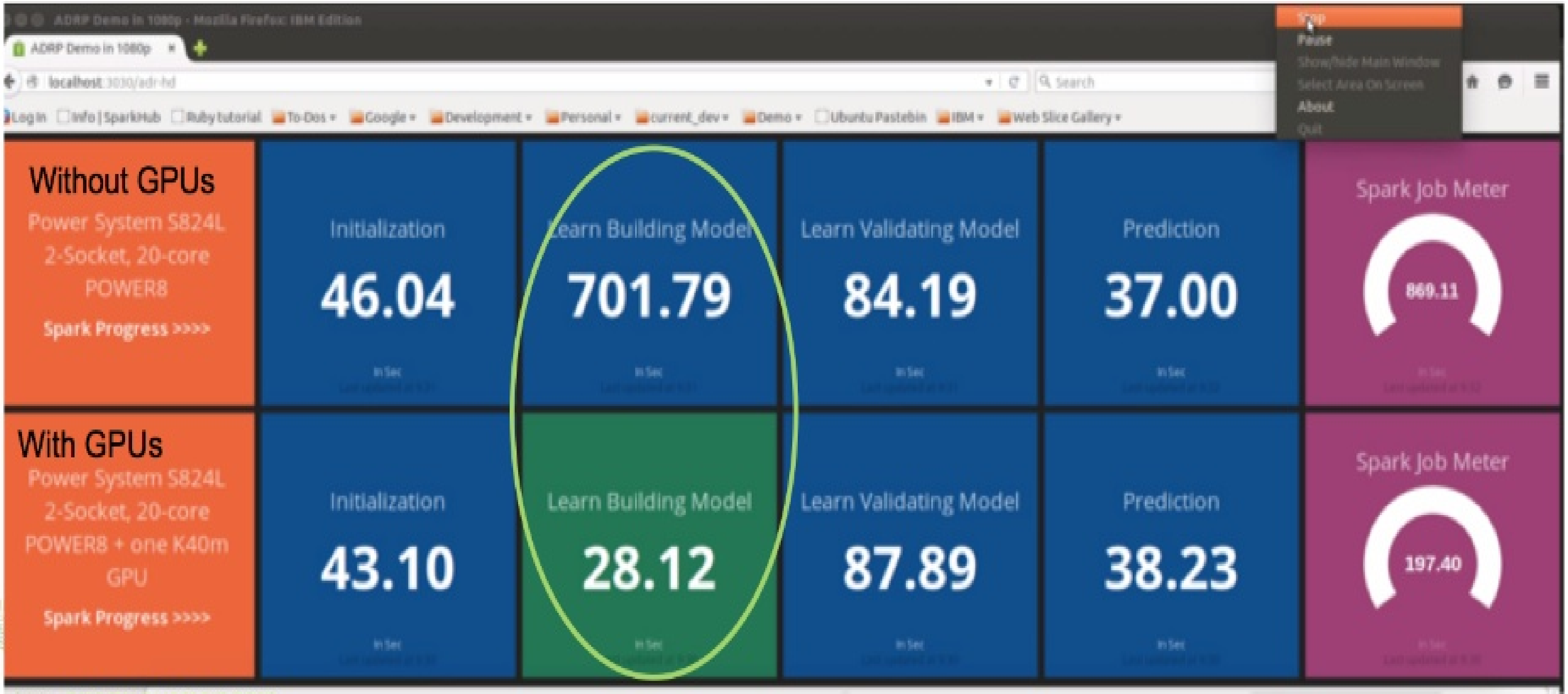




A typical example –

Personalized Medicine – Adverse Drug Reaction Workload

- 30X faster at learning speed and 4.3 X speed up at end-2-end
- Need to fully utilize both GPU and CPU resources to get economic benefits

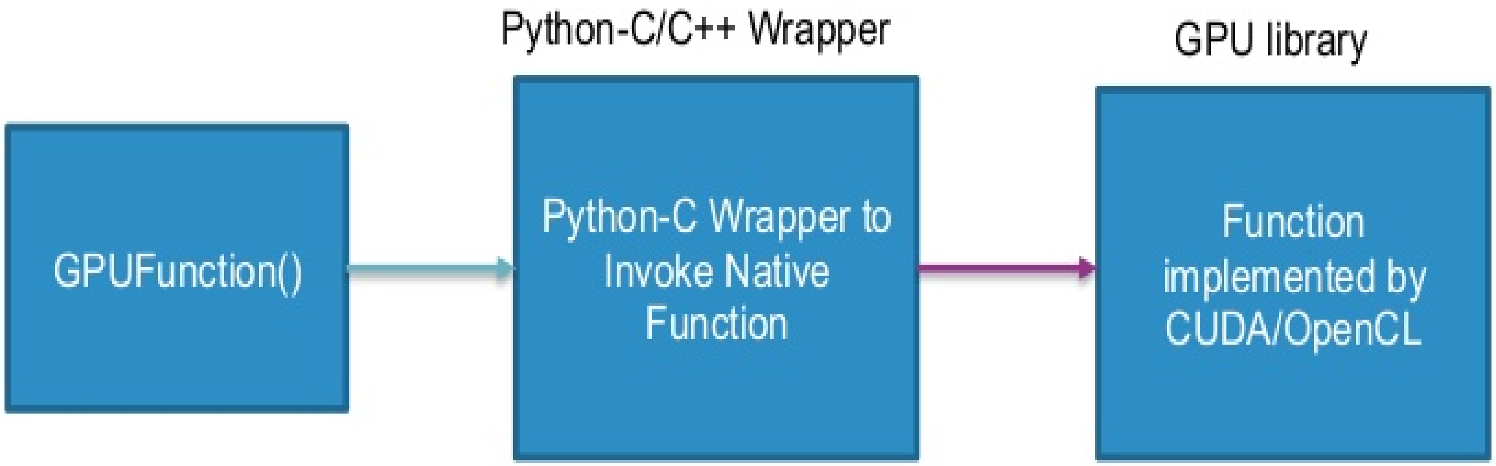


Scheduling Granularity

- Scheduling at application level
  - Mesos and Yarn tag the GPU machine with label
  - Schedule the application on GPU hosts based resource requirement of application
  - Corse grained scheduling leads to low utilization of CPU/GPU.
- Scheduling at DAG level
  - Need fine grained sharing for GPU resources rather than reserving entire GPU machines
  - Identify GPU operation
  - Optimize the DAG tree by decupling GPU operations from CPU operations and by inserting new GPU stages
  - Reduce GPU wait time, enable sharing GPU among different jobs and therefore improve the overall GPU utilization

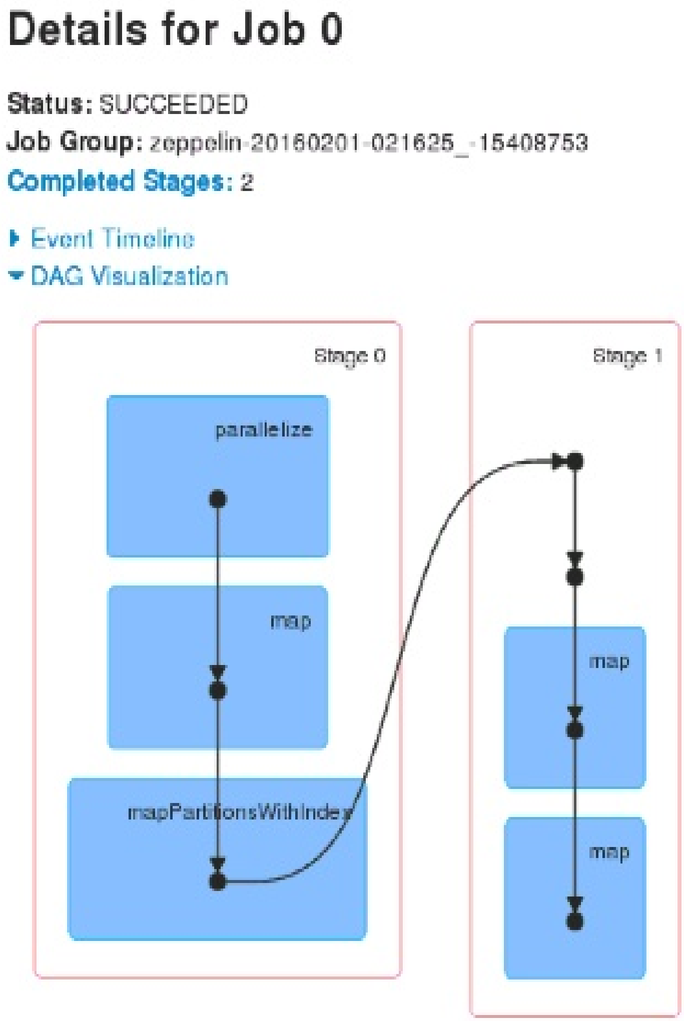
GPU tasks recognition

- GPU and CPU tasks mixed together
- Separate the workload is necessary for scheduling control



GPU tasks recognition

- Mark the GPU workload by DAG operation
  - Go through the DAG tree to identify the stages with GPU requirement
  - Optimize the distribution by inserting GPU stage



Policies

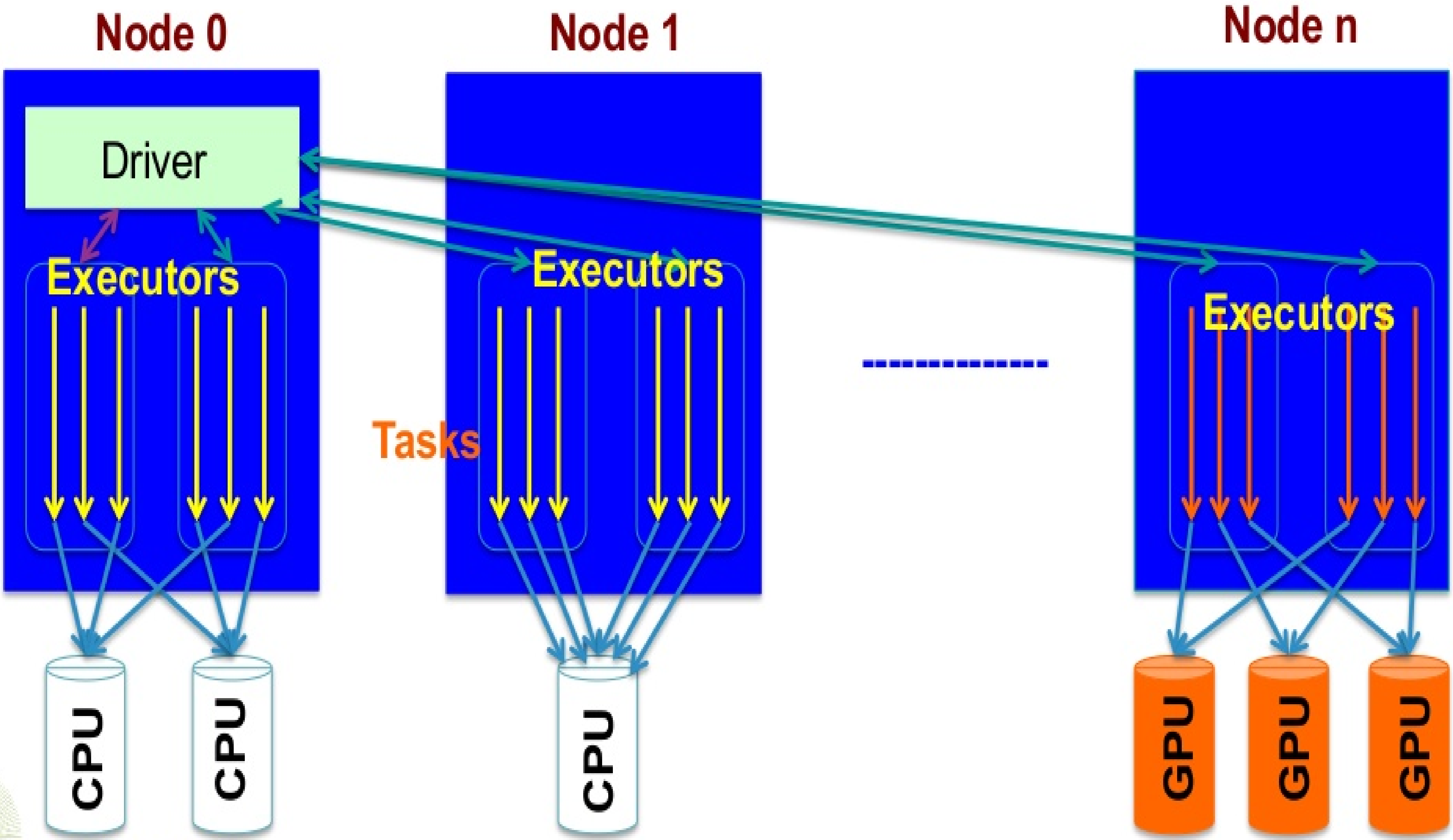
- RM needs capability to identify the GPU hosts and manage along with CPU resources
- Prioritization policy - share GPU resource among applications
- Allocation policy – control GPU and CPU allocation independently – multi-dimensional scheduling
- Fine grained policy to schedule tasks according to GPU optimized DAG plan

Adaptive Scheduling

- CPU & GPU tasks are convertible in many applications
- Scheduling needs adaptive capability
  - If GPU is available, use a portion of GPU
  - Otherwise run rest of tasks on CPU

```
dstInBlocks.join(merged).mapV
alues {
   ....
   if (useGPU) {
      loadGPULib()
      callGPU ()
   }
   else {
     //CPU version
   }
}
```

Adaptive Scheduling



Efficiency Considerations

- Do we need to wait GPU resource if there is CPU available?
- Do we need rerun the CPU tasks on GPU if tasks on CPU are long-tail?
- Do we need to have failover cross resource type?

Defer Scheduling

- Traditional defer Scheduling
  – Wait for data locality
  – Cache, Host, Rack
- Resource based defer scheduling
  – Necessary if the GPU can greatly speed up task execution
  – Wait time is acceptable

Future works

- Global optimization
  - Consider the cost of additional shuffle stage
  - Consider data locality of CPU and GPU stage
  - Add time dimension to MDS
  - Optimize global DAG tree execution
  - Use historical data to optimize future execution, e.g, future iteration

Building Spark Centric Shared Service with IBM Conductor

1 Improve Time to Results

Run Spark natively on a shared infrastructure *without* the dependency of Hadoop. Reduce application wait time, improving time to results.

3 Reduce Administration Costs

Proven architecture at extreme scale, with enterprise class workload management, multi-version support for Spark, monitoring, reporting, and security capabilities.

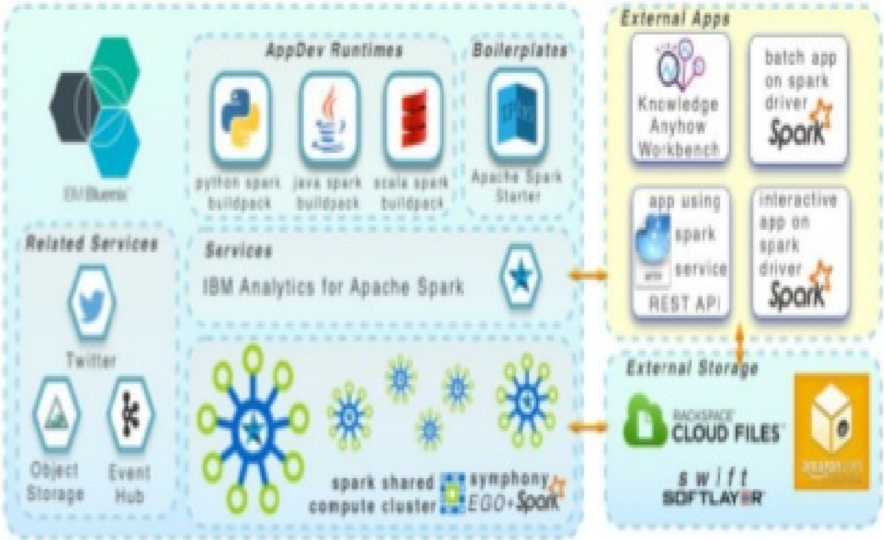
2 Increase Resource Utilization

Fine grain, dynamic allocation of resources maximizes efficiency of Spark instances sharing a common resource pool. Multi-tenant, multi-framework support. Eliminates cluster sprawl.

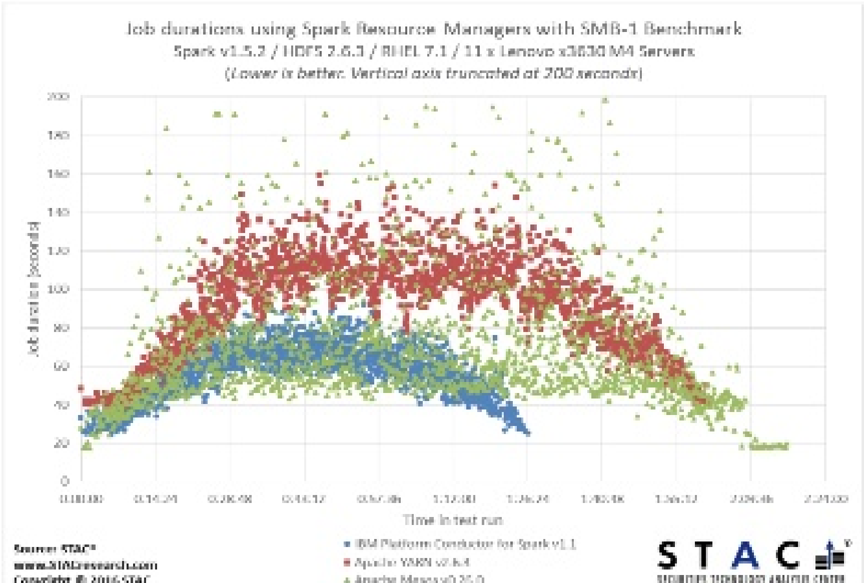
4 End-to-End Enterprise Class Solution

- IBM STC Spark Distribution
- IBM Platform Resource Orchestrator / Session Scheduler, application service manager.
- IBM Spectrum Scale FPO

IBM Conductor with Spark



IBM Bluemix Spark Cloud Service in production – thousands of users and tenants.



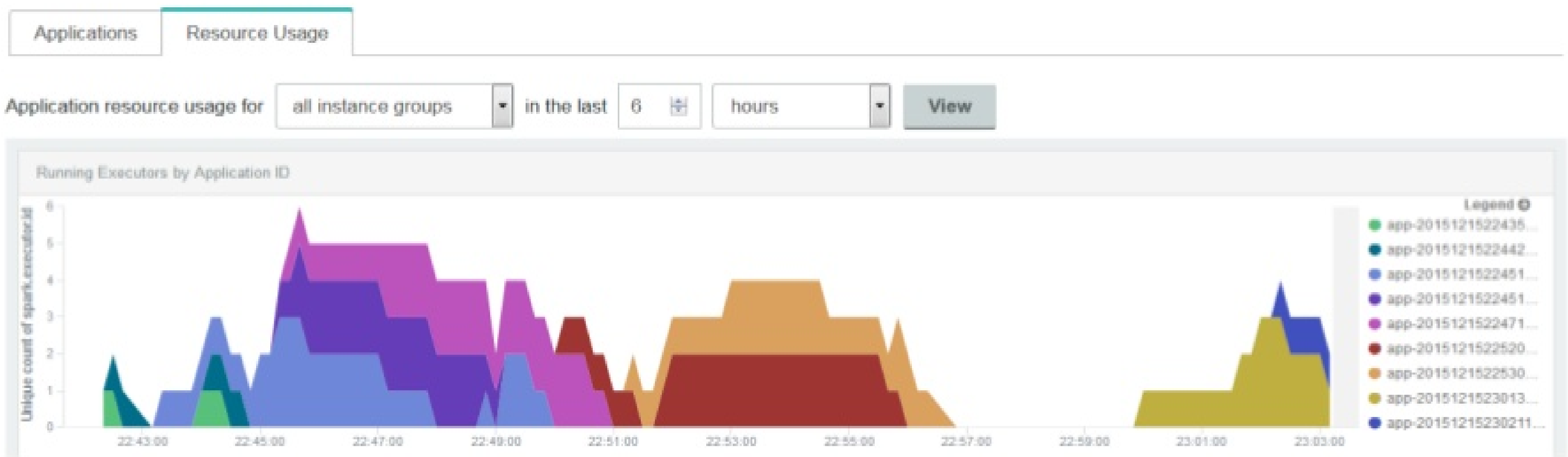
Third party audited benchmark indicated significant performance/ throughput/SLA advantages

https://stacresearch.com/news/2016/03/29/IBM160229

IBM Conductor with Spark

Monitor and Reporting with Elastic (ELK)

- Integrated Elastic Search, Logstash, Kibana for customizable monitoring
- Built-in monitoring Metrics
  - Cross Spark Instance Groups
  - Cross Spark Applications within Spark Instance Group
  - Within Spark Application
- Built-in monitoring inside Zeppelin Notebook



Demo

THANK YOU.

Contact information or call to action goes here.

Acceleration Opportunities for GPUs & Spark

Analytics Model	Computational Patterns suitable for GPU Acceleration
Regression Analysis	Cholesky Factorization, Matrix Inversion, Transpose
Clustering	Cost-based iterative convergence
Nearest-neighbor Search	Distance calculations, Singular Value Decomposition, Hashing
Neural Networks	Matrix Multiplications, Convolutions, FFTs, Pair-wise dot-products
Support Vector Machines	Linear Solvers, Dot-product
Association Rule Mining	Set Operations: Intersection, union
Recommender Systems	Matrix Factorizations, Dot-product
Time-series Processing	FFT, Distance and Smoothing functions
Text Analytics	Matrix multiplication, factorization, Set operations, String computations, Distance functions
Monte Carlo Methods	Random number generators, Probability distribution generators
Mathematical Programming	Linear solvers, Dynamic Programming
OLAP/BI	Aggregation, Sorting, Hash-based grouping, User-defined functions
Graph Analytics	Matrix multiplications, Path traversals