Supporting Highly Multitenant Spark Notebook Workloads

Brad Kaiser, IBM/TWC

Craig Ingram, IBM/TWC

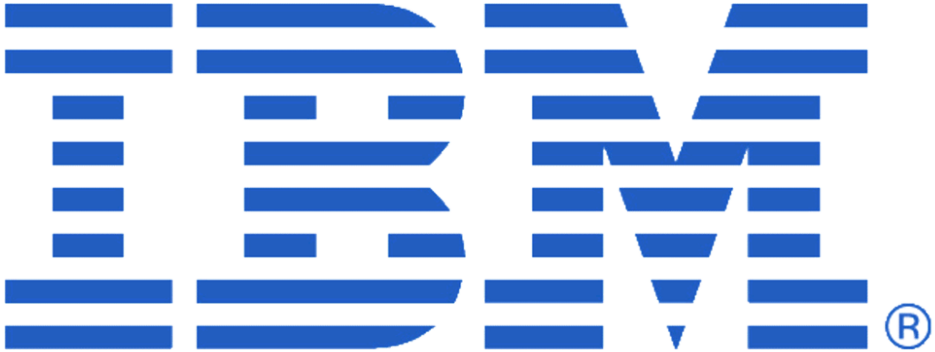
#EUdev8

Hosting Multitenant Spark Notebooks Is Hard But It Doesn't Have To Be

- Our Journey
- Best Practices
- New Work

Our Journey

Who we are



IBM's Commitment to Open Source

- Contribute intellectual and technical capital to the Apache Spark community.
- Make the core technology enterprise- and cloud-ready.
- Build data science skills to drive intelligence into business applications — https://cognitiveclass.ai/
- http://spark.tc

Key Open source steering committee memberships

OSS Advisory Board

Mission

- Provide:
  - a secure, performant, stable cluster.
  - interactive analytics, visualizations, and reports.
  - collaboration and sharing with other data scientists, engineers, and consumers.
  - job scheduling capabilities.
  - a quick and easy way to get started with Spark.

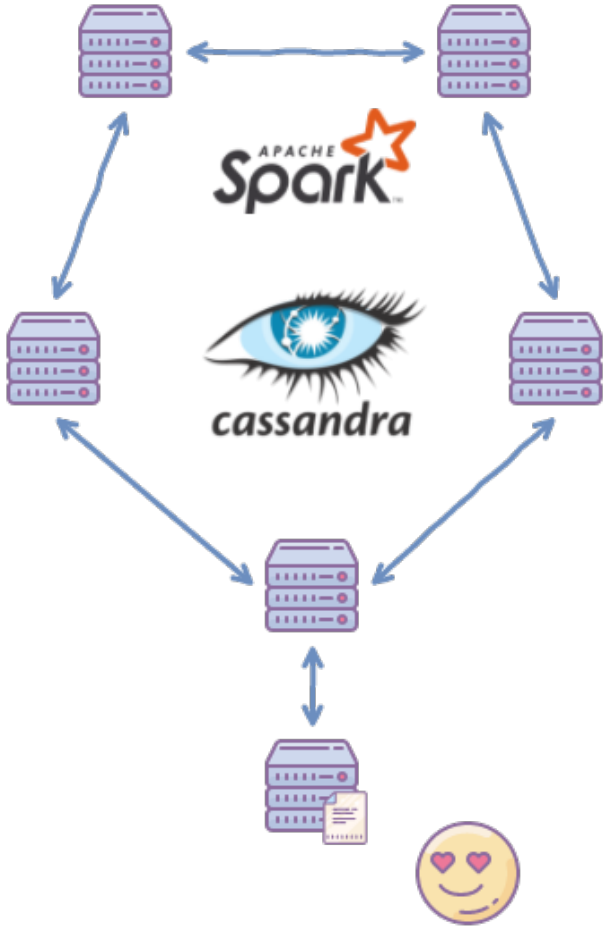
Goals

- Support hundreds of analysts/data scientists using Spark
  - Quick kernel creation (<10s from notebook creation to available sc).
  - Utilize cluster resources efficiently.
  - Elastically scale based on load.

Lessons Learned at TWC

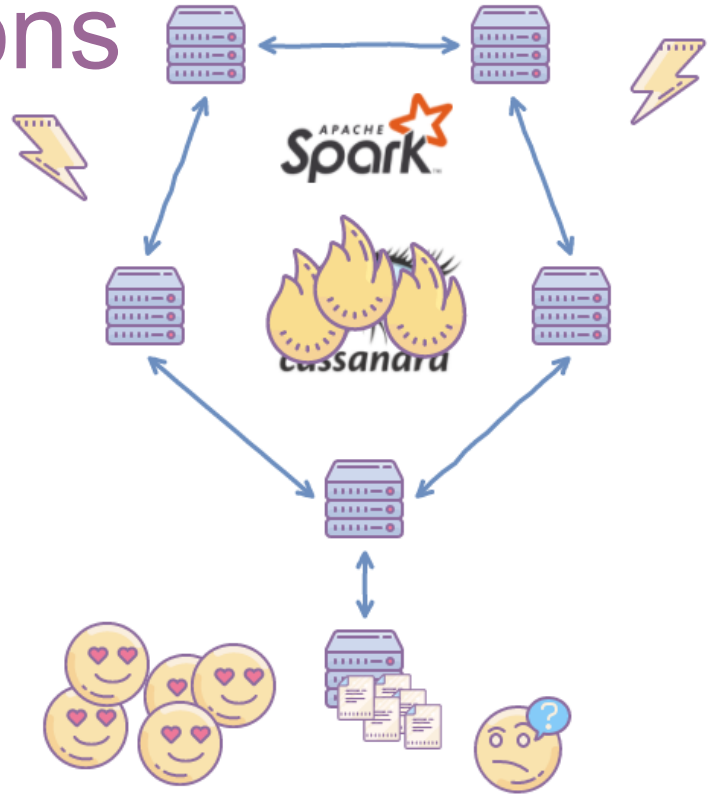
One Big Cluster

- Spark collocated with Cassandra
- Fast
- Stable



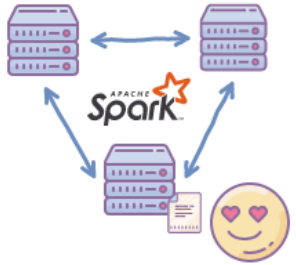
One Big Cluster - Cons

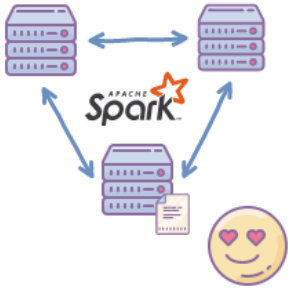
- Outside analysts start using our cluster
- Provided notebook services
- Interrupted our perfectly scheduled jobs
- Used a lot of resources causing Cassandra to crash

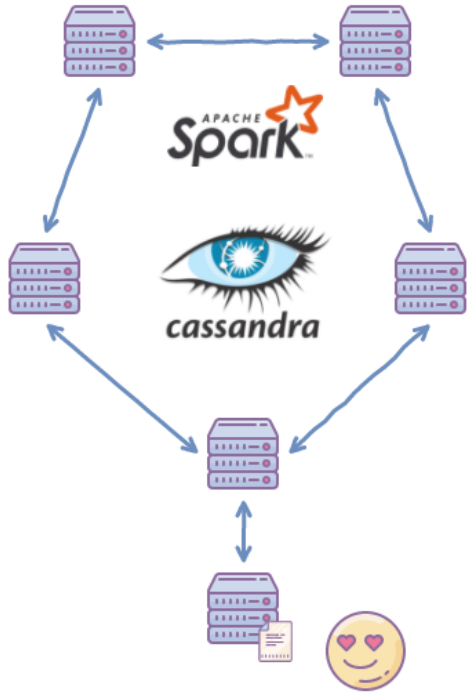


Add Smaller Clusters

- We built some smaller clusters
- Still platform agnostic
- Other teams couldn't affect our production cluster

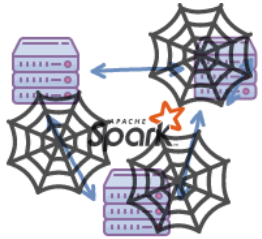


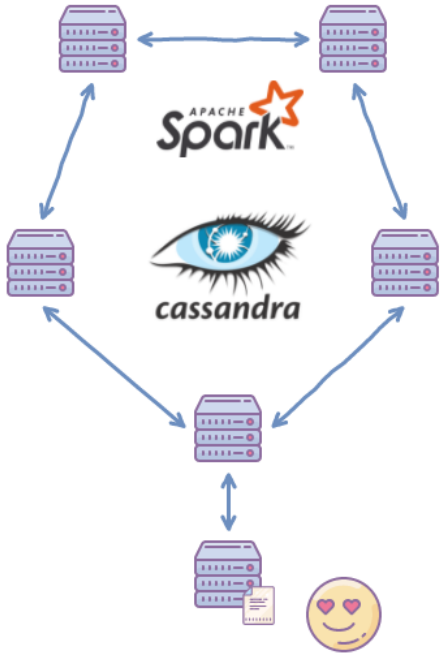




Add Smaller Clusters - Cons

- Hassle to set up
- Required a lot of maintenance
- Sat idle





EMR

- Analysts make ad hoc clusters for their needs
- No maintenance from us
- Learning curve for analysts
- They tend to leave them running

Lessons Learned - IBM

Data Science Experience

- Collaborative environment on the front end
  - Collaboration Tools
  - Shared Data Sets
  - Flows
  - GitHub Integration
- Multiple compute environments on the back end
  - DSX on the cloud: compute runs on IBM cloud
  - DSX Local: compute runs on private cloud or Z

Lessons Learned - IBM

- You need kernel remoting
  - Allows advanced collaborative tools in the application tier
  - Allows resource consolidation in the analytics tier
- Resource consolidation puts stress on the analytics tier
  - Starvation
  - Management of cached data
  - Performance bottlenecks (example: Spark web UI)

Best Practices

Best Practices

- Use kernel remoting
- Use fewer, bigger clusters
- Know your workloads
- Isolate users
- Schedule resources efficiently

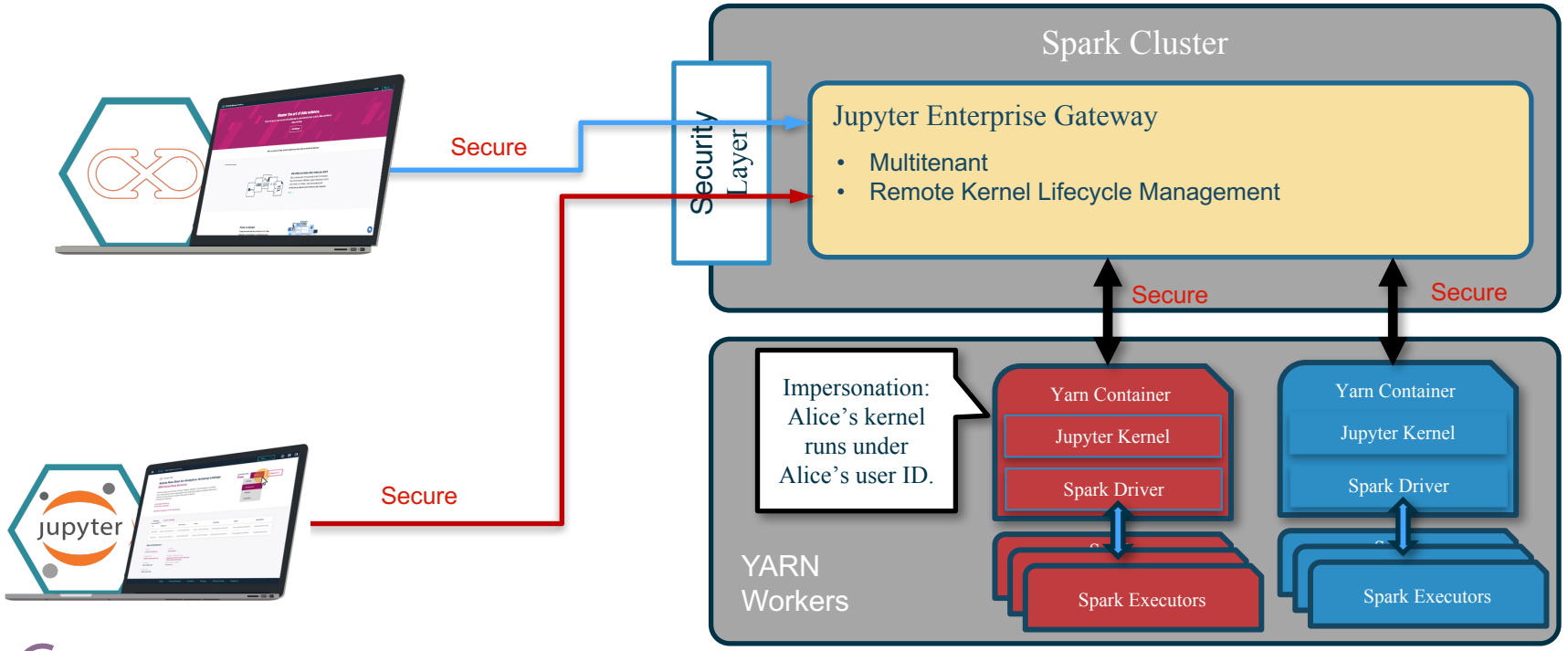
Use Kernel Remoting

- Running all of your notebook kernels on the same server is a bottle neck
- Run your kernels distributed on the cluster
- You can run a lot more notebooks

Jupyter Enterprise Gateway

- New Open Source project from IBM
- Goals:
  - Allow hundreds of notebook users to share a single Spark cluster…
  - …with enterprise-level security and performance.
- Used in IBM Analytics Engine (GA)
- developer.ibm.com/code/openprojects/jupyter-enterprise-gateway-2

Jupyter Enterprise Gateway: How it Works



Use Fewer, Bigger Clusters

- Better resource utilization through statistical multiplexing
- Improved security and auditing due to centralization
  - Hive, Ranger, and Atlas are common in the ecosystem.
  - Many new, platform specific solutions to address this problem.
- Easier collaboration between users
  - Shared notebooks with interactive visualizations and markdown support.
  - GitHub integration for versioning and external sharing.
  - Catalog based data discovery and sharing.
  - Governance and auditing support.

Know your workloads

- What is the main resource they use?
- Overprovision the hardware
- When to scale up and down?
  - CPU load
  - YARN/RM Queue Stats (depth, waiting jobs, available CPU/mem, preemptions)
    - If containers are getting preempted, it's due to queues filling up.

Isolate Users

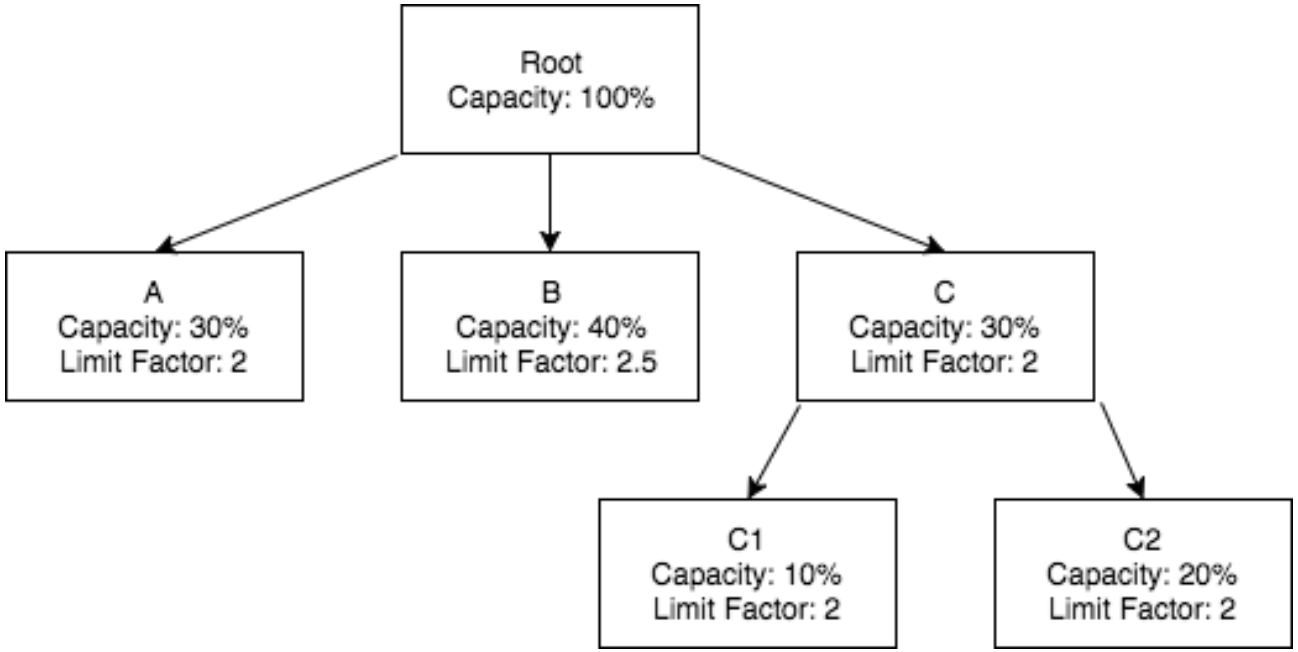
- Don't have a generic user account
- Catalog and Governance
  - Hive
  - Atlas
  - Ranger
- You don't want users embedding keys and passwords in their notebooks.

Schedule Resources Efficiently

YARN Queues

- Take advantage of YARN's hierarchical queue system to manage and organize resources.
  - Over-allocate queues for better resource utilization and sharing.
  - Take advantage of node labels for users that have priority jobs that require an SLA.
  - Intra-queue preemption and asynchronous container allocation should be available in YARN 3.0.

YARN Queues


Root
Capacity: 100%
A
Capacity: 30%
Limit Factor: 2
B
Capacity: 40%
Limit Factor: 2.5
C
Capacity: 30%
Limit Factor: 2
C1
Capacity: 10%
Limit Factor: 2
C2
Capacity: 20%
Limit Factor: 2

Dynamic Allocation

- Dynamic allocation lets you take advantage of varying activity
  - Proactively scales the number of executors based on the scheduler's backlog.
  - Removes idle executors after a timeout.
  - Be sure to set a sensible number of initial executors and minimum executor floor and let them ramp up on demand.
- Static allocation best for known workloads

New Work

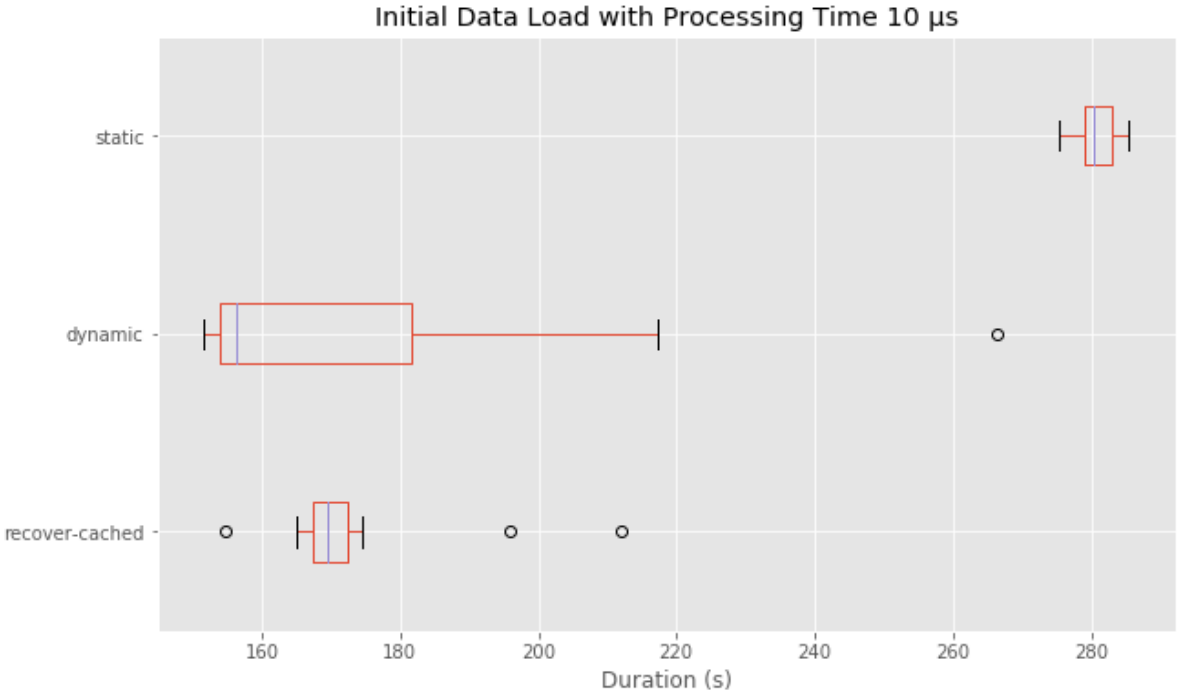
Improvements to Spark

- Alleviate the tradeoffs inherent in current best practices.
  - **Recover cached data** when shutting down idle executors
  - Proactively shut down executors to **prevent starvation**

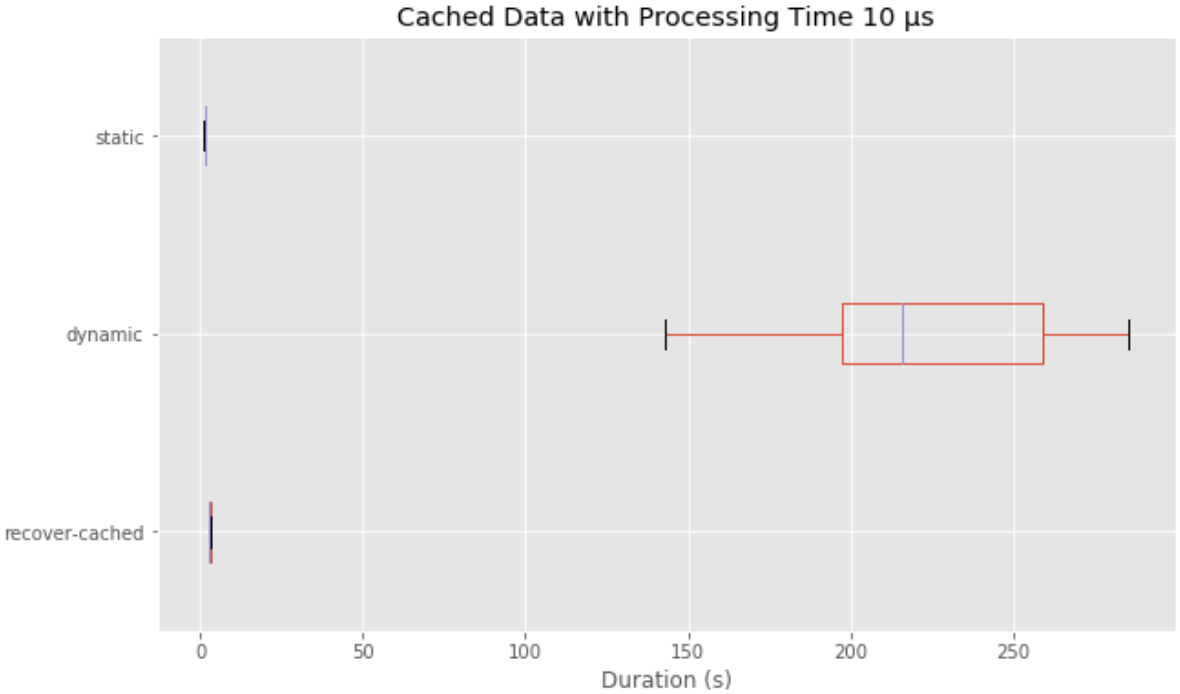
Recovering Cached Data

- Replicates cached data to remaining executors before shutting them down.
- Ameliorates cache issues with dynamic allocation
- Useful in shared spark notebook environments

Benchmarks


Initial Data Load with Processing Time 10 µs
static
dynamic
recover-cached
160
180
200
220
240
260
280
Duration (s)

Benchmarks


Cached Data with Processing Time 10 µs
static
dynamic
recover-cached
0
50
100
150
200
250
Duration (s)

Check out my PR

- SPARK-21097
- github.com/apache/spark/pull/19041

Preventing Starvation

- Eliminate issues where users are unable to run anything due to other users taking up all of the cluster's resources.
- Especially useful in shared spark notebook environments where idle resources can be reclaimed easily.
- Preemption can solve this.

Enter Preemption

- Requests containers associated with over-allocated queues to shut down.
- Handle YARN's PreemptionMessage in a way that best suits the workload.
- Pick the right executors to terminate.

Keep up with the JIRA

- SPARK-21122
- PR coming soon

Call to action

- Look at our JIRAs
- Try out our PRs

Shout-outs!!!

- Our notebook workload simulator, benchmark, and tracing tools.
  - spark-bench - github.com/SparkTC/spark-bench
    - Check out Emily Curtin's talk tomorrow about spark-bench.
  - spark-tracing - github.com/SparkTC/spark-tracing
    - Matthew Schauer's baby awaiting open-source approval.
- Special thanks to Vijay Bommireddipalli and Fred Reiss for their guidance and support!

Contact Info

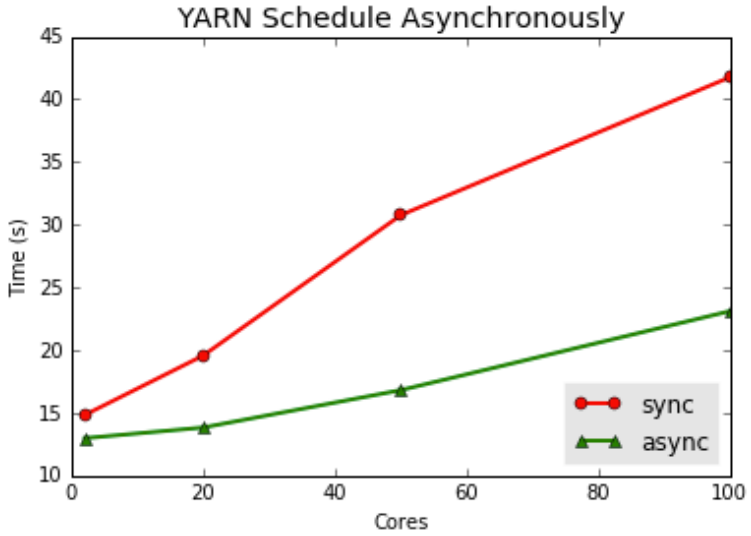
Brad Kaiser
kaiserb@us.ibm.com

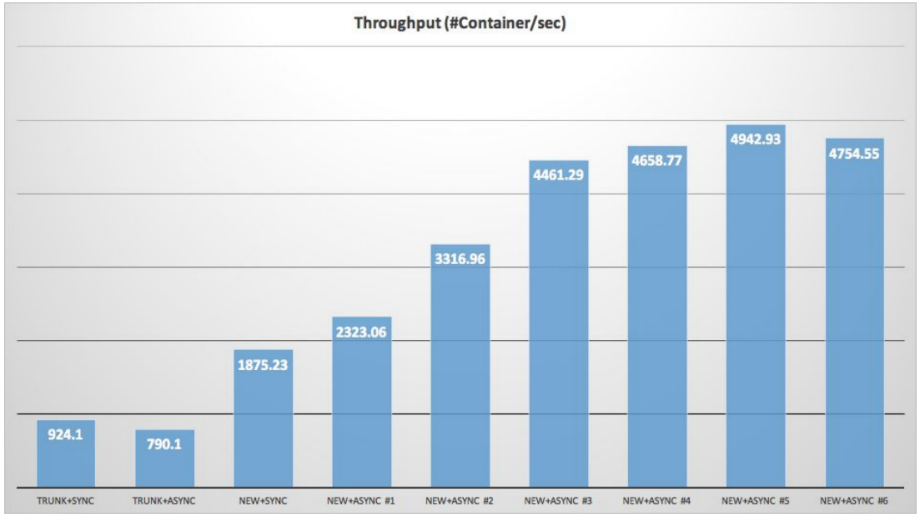
Craig Ingram
cingram@us.ibm.com

Extra Material

YARN Asynchronous Scheduling

- Enable asynchronous scheduling of containers in YARN.
  - `yarn.scheduler.capacity.schedule-asynchronously.enable`
  - YARN-7327 and YARN-5139





References

- Some Icons provided by Icons8

Quick Settings

- Use `spark.yarn.jars` or `spark.yarn.archive`.
- Runnin[illegible] mprove perform[illegible]
- Support for multiple/new versions of Spark.

Move to backup

Disable unused credential providers

- `spark.yarn.security.credentials.hive.enabled`
- `spark.yarn.security.credentials.hbase.enabled`

cor		
defa		
nohive	8.7	0.05
nohbase	7.87	0.05

Move to backup

Problem Domain

- Security
  - UI and service protection
  - Data governance and auditing
- Stability
- Performance

What’s next…

- spark-on-k8s
- Scheduler improvements
- Executor startup time reduction

- Hundreds of notebook users leads to a highly **multitenant** and **interactive** workload.

Many users at once

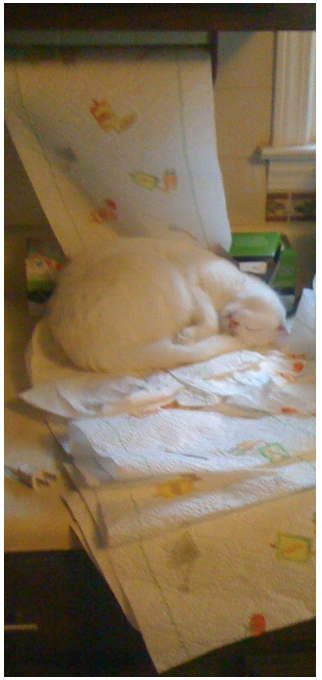
Bursty offered load

Latency is important

- Challenge: Give each user the illusion of having a large cluster all to herself

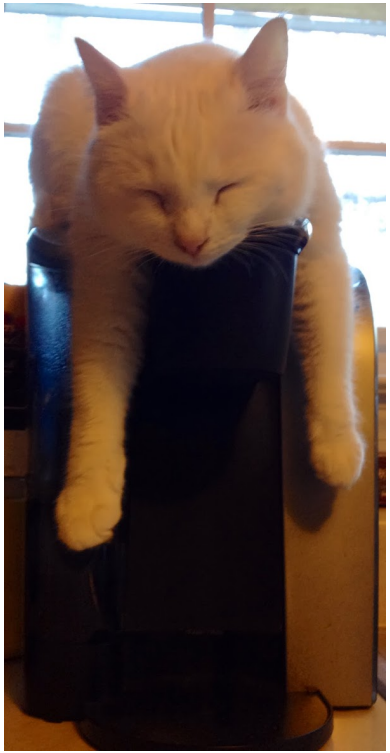
What platforms are out there… Hosted Solutions

- Data Science Experience (DSX)/Watson Data Platform (WDP)
- databricks
- Google Cloud Platform
- Microsoft Azure HDInsight
- and others!!!



What platforms are out there… Self-Hosted Solutions

- HDP – Hadoop Data Platform
- CDH – Cloudera Distribution Including Hadoop
- MapR
- Mesosphere



Reasons to Self Host at TWC

- Cloud Agnostic
- Flexibility
- Sensitive Data
- Fewer Options in 2014
- Cassandra Colocation
- Cost…maybe?

Potential Downfalls

- In a self-hosted environment, everything is up to you.
  - Security
  - Stability and Performance
  - Scalability
    - Compute
    - Storage
  - Monitoring
  - Alerting
  - Logging