Spark on YARN: The Road Ahead

Marcelo Vanzin

What to expect...

- What is YARN?
- Why YARN for Spark?
- Current state and future direction for:
  - Resource Management
  - Security

What is YARN?

Resource management and negotiation

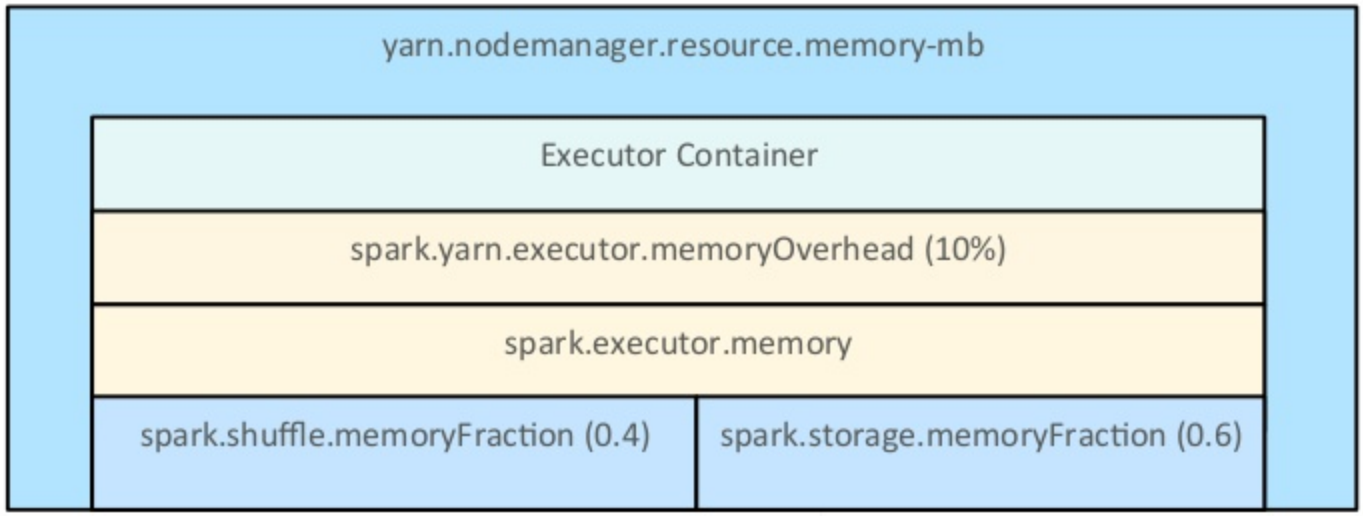
- Manages vcores and memory
- Multiple users
- Multiple apps
- Advanced features

Why YARN for Spark?

- Advanced resource management features
- Dynamic allocation for Spark executors
- Security

Resource Management

Lots of knobs to turn, lots of trial and error.

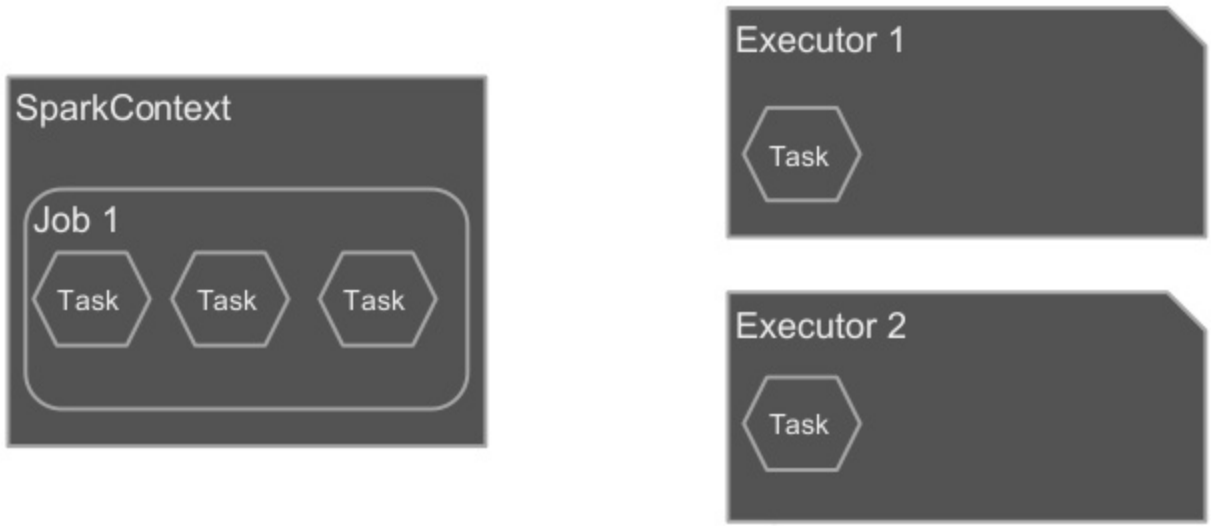


RM (Cont'd)

Dynamic allocation

- No need to specify number of executors!
- Application grows and shrinks based on outstanding task count
- Still need to specify everything else…

Dynamic Allocation

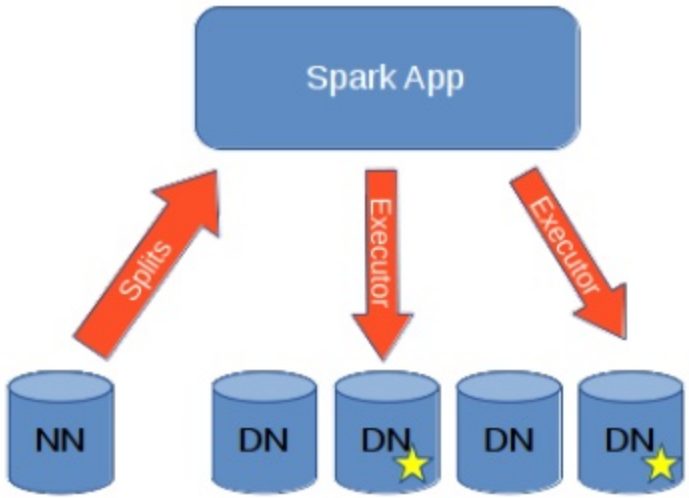


Dynamic Allocation (Cont'd)

How to make dynamic allocation better?

- Reduce allocation latency
- Data locality hints
- Handle cached RDDs

Data Locality



- Allocate executors close to data
- More predictable performance
- SPARK-4352

Cached RDDs

- Short term: avoid discarding cached data
  - Keep executors around.
- Long term:
  - Cache rebalance
  - Container resizing
  - Off-heap caching

Memory Sizing

How to make application sizing easier?

- Soft container limits (YARN-3119)
- Overhead memory
- Simplified parameters

Overhead Memory

Everything that is not the Java heap. Very opaque, hard to size correctly.

- Better metrics to understand usage
- Better heuristics for automatic sizing

Future: Simplified Allocation

Single parameter: “task size”. Spark figures out the rest (vcores, heap, overhead).

Container sizes can change to match available resources, data locality, etc.

RM Summary

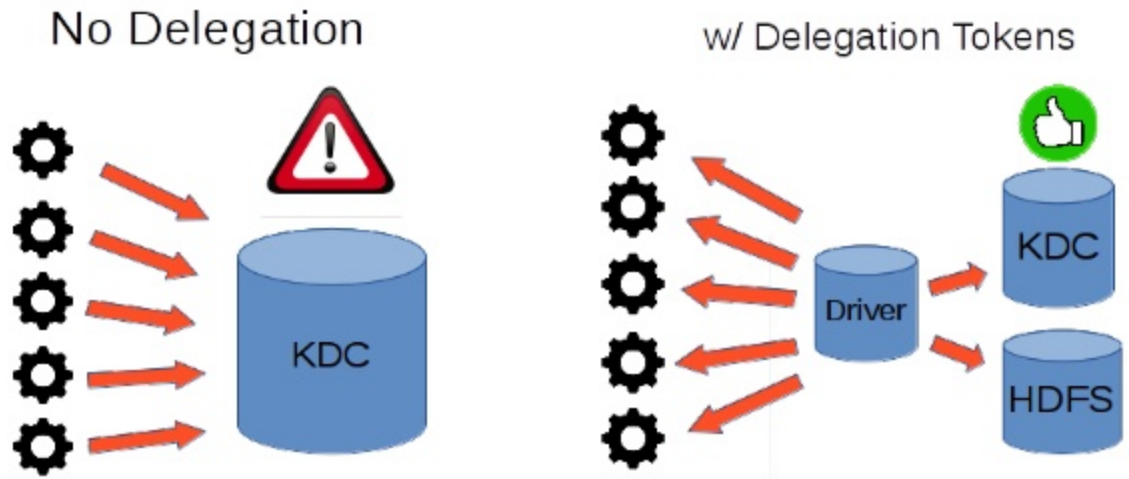
"Dynamic allocation is not for everybody. It can only meet the needs of 90% of applications."

Security: Kerberos

- User authentication via Kerberos
- Supported in all services (YARN, HDFS, HBase, Hive, etc)
- Uses per-service delegation tokens

Security: Delegation Tokens

Delegation tokens allow for authentication without running into KDC limitations.



Tokens (Cont'd)

Tokens have a TTL. This is a problem for long-lived applications

- Spark Streaming
- Thrift Server

Solution: re-generate tokens periodically.

Tokens (Cont'd)

Generating tokens requires a kerberos ticket

- Spark driver handles KDC login
- Need access to user credentials (keytab)

Secure if coupled with encryption.

Security: Encryption

- HDFS supports wire encryption
- YARN supports wire encryption
- Spark not so much

Encryption (Cont'd)

Where does Spark need encryption?

- Control plane
- File distribution
- Block Manager
- User UI / REST API
- Data-at-rest (shuffle files)