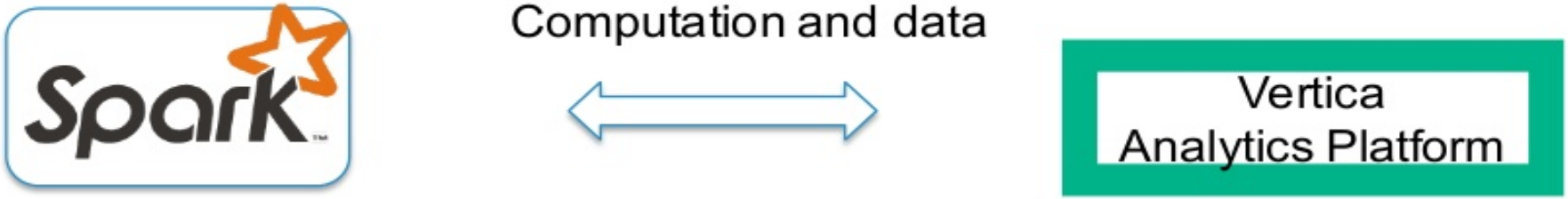
Vertica and Spark: Connecting Computation and Data

Rui Liu and Edward Ma

Hewlett Packard Enterprise Vertica Advanced R&D Labs

SPARK SUMMIT 2016
DATA SCIENCE AND ENGINEERING AT SCALE
JUNE 6-8, 2016 SAN FRANCISCO

Overview

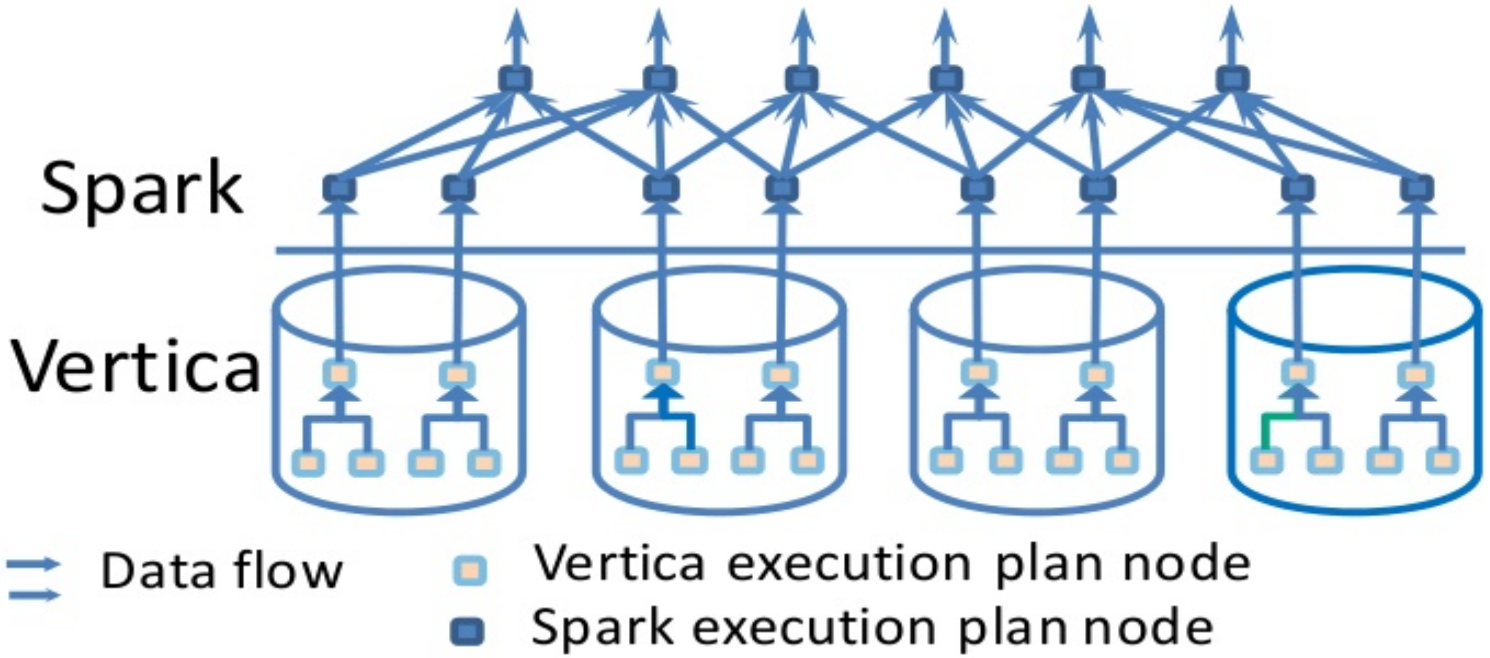


- The Vertica-Spark connector connects both data and computation between Vertica and Spark
  - VerticaRDD and Vertica Data Source APIs
  - Data-locality optimization
  - Computation pushdown
  - Save data from Spark to Vertica

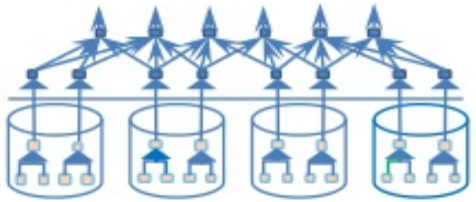
HPE Vertica

- An advanced SQL analytics platform
- Shared-nothing MPP architecture
- Column-oriented storage organization
- Standard SQL interface with many analytics capabilities built-in
- Extensible via User-Defined Functions

Connected pipeline



Connected pipeline

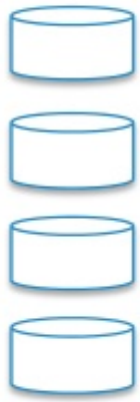


- Connects the computation pipelines of Spark and Vertica in an optimized way
- Utilizes parallel channels (parallel queries) for data movement
- Leverages data-locality optimization on Vertica
- Ensures computation push-down into Vertica as appropriate (filters and projections)

Vertica data segmentation on hash ring

CREATE TABLE t …
SEGMENTED BY HASH(id) …

Id	Name	Age	email	…

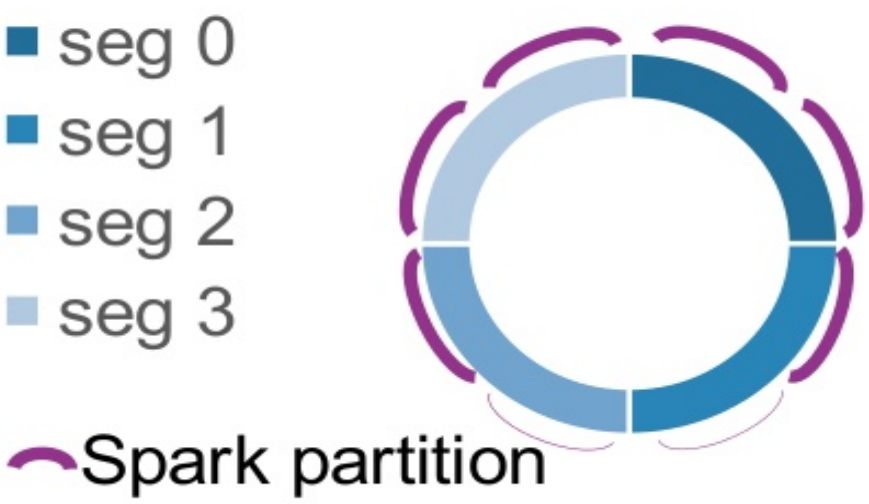


Spark locality-aware partitions

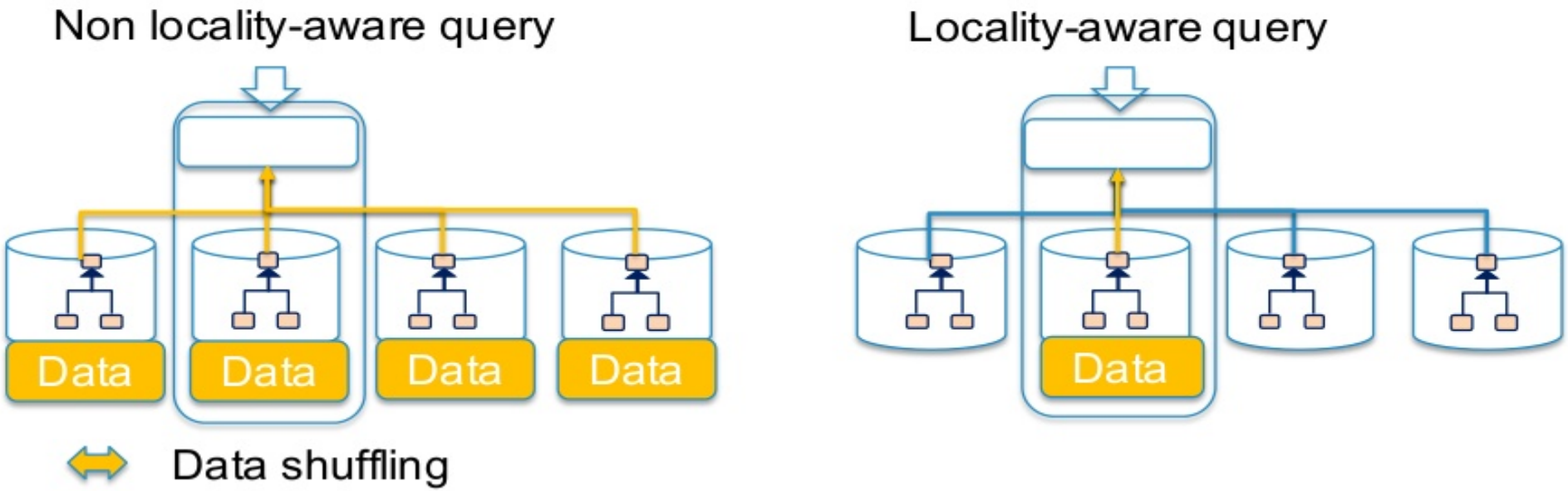
Locality-aware partition:

- Uses Vertica data-distribution information
- Generates locality-aware queries

```
SELECT id, name  FROM T
where
   0 < 0xffffffff & hash(id)
AND
   0xffffffff & hash(id) < 278985;
```

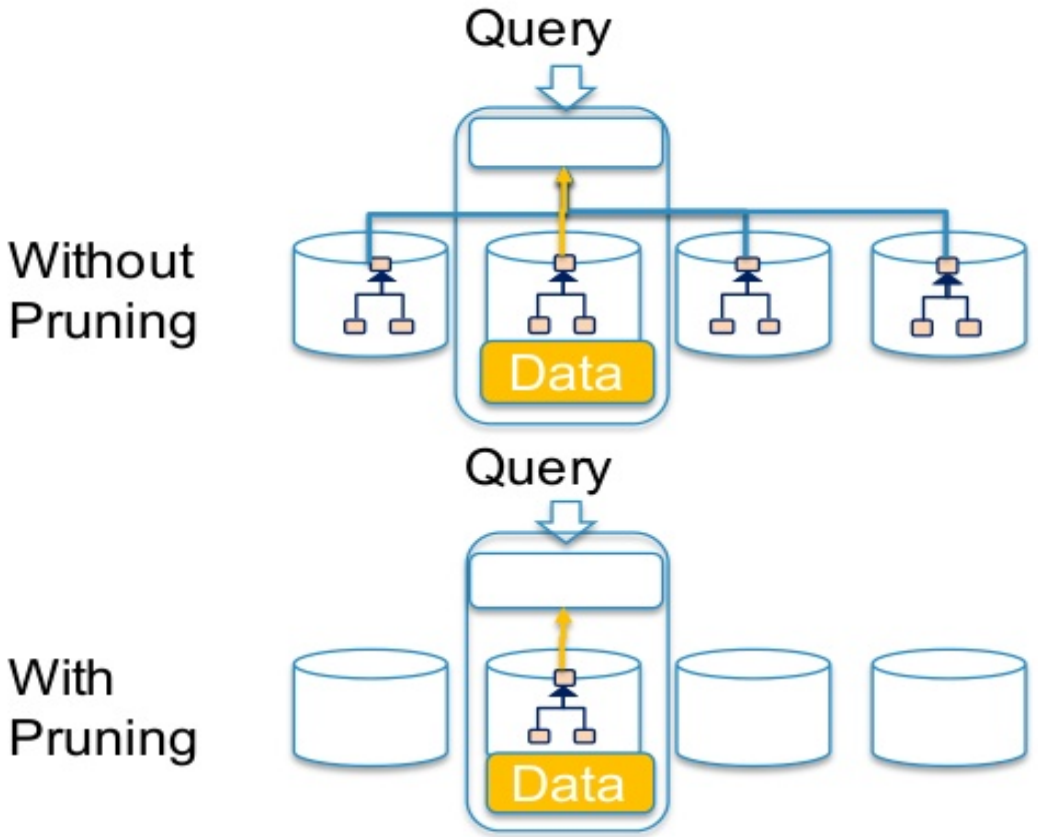


Locality-aware query



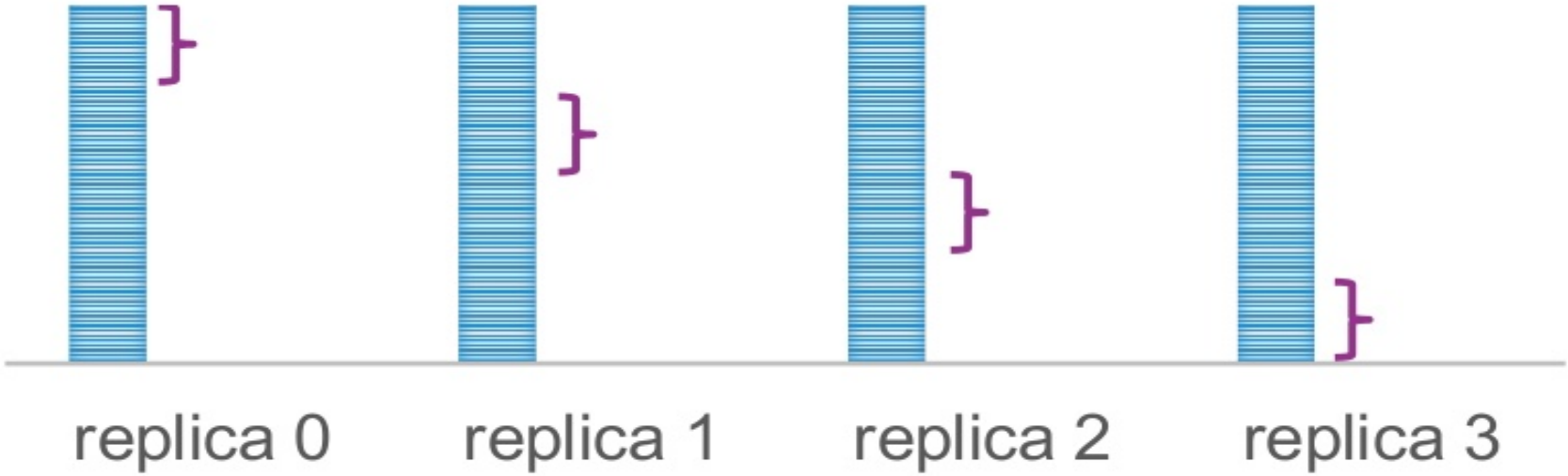
Vertica query pruning

- For a locality-aware query, the execution can be pruned
- Query only executes on nodes that contain the data

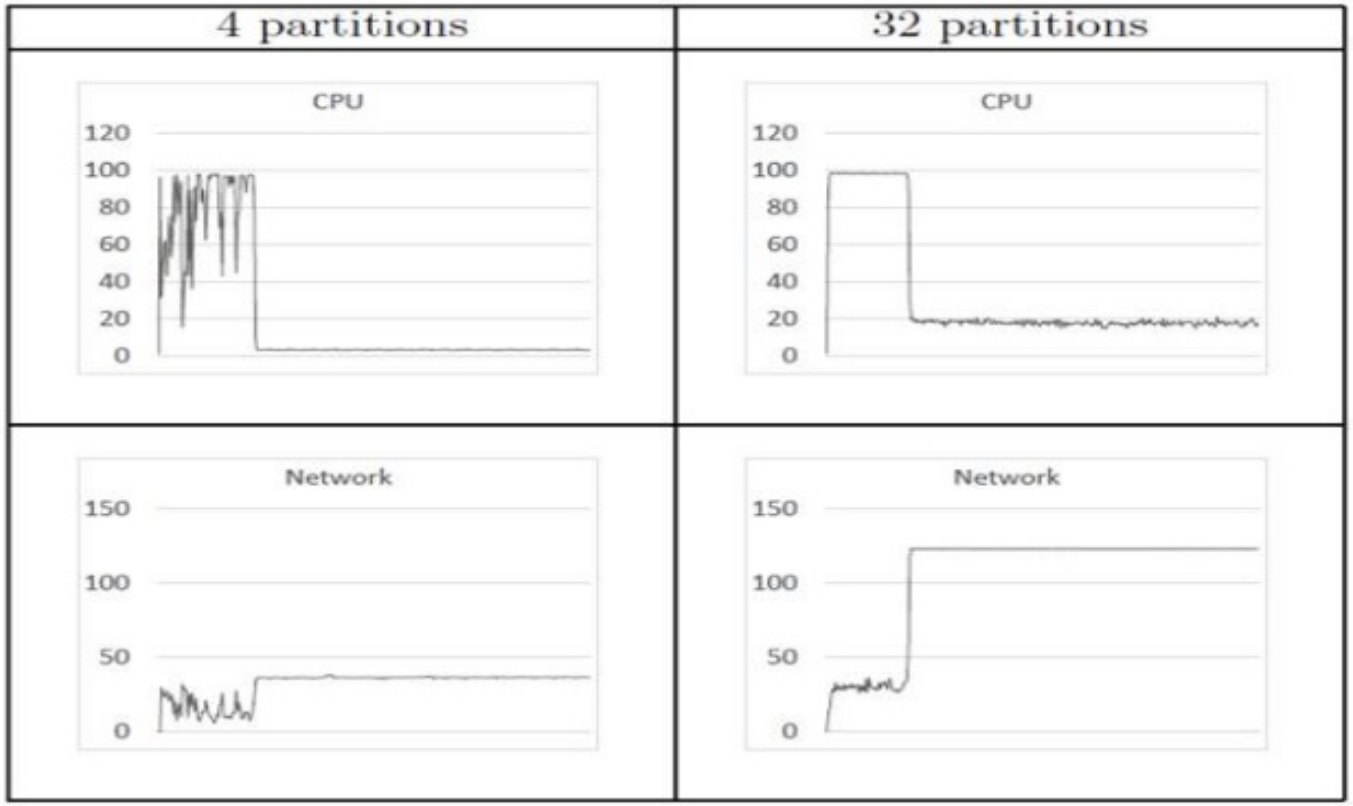


Arbitrary number of partitions

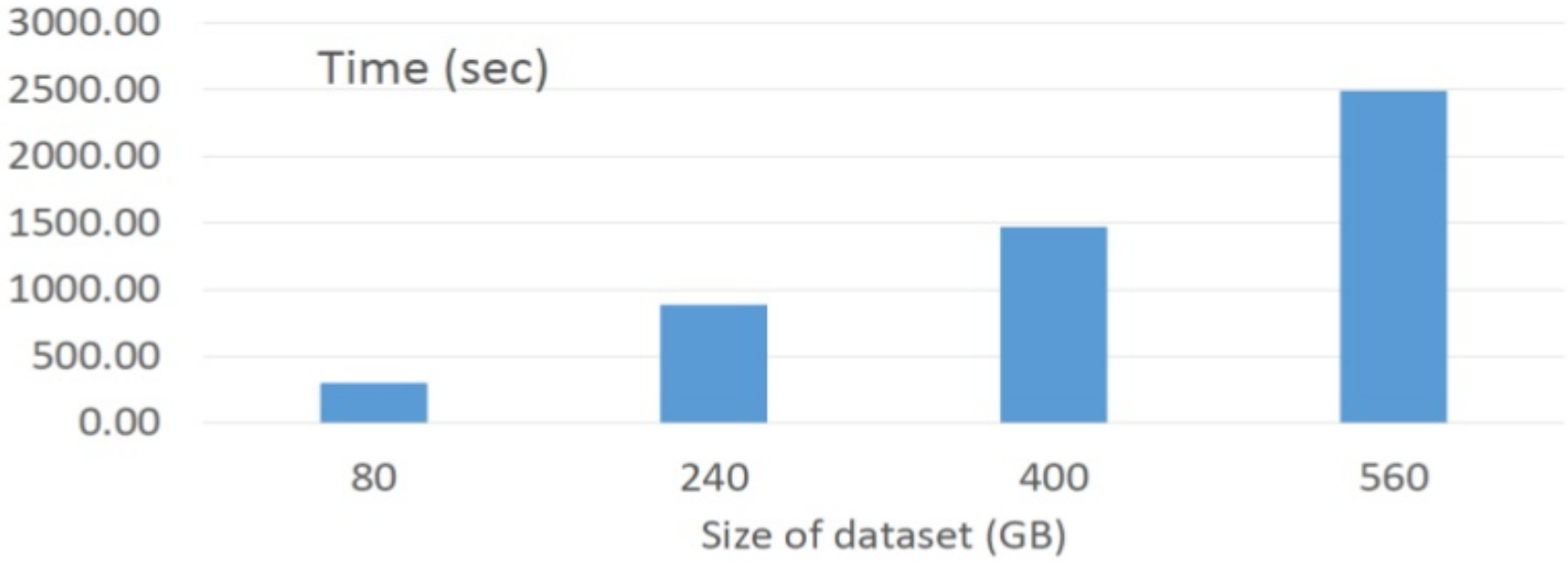
Spark partitions over unsegmented table replicas



Performance characteristics

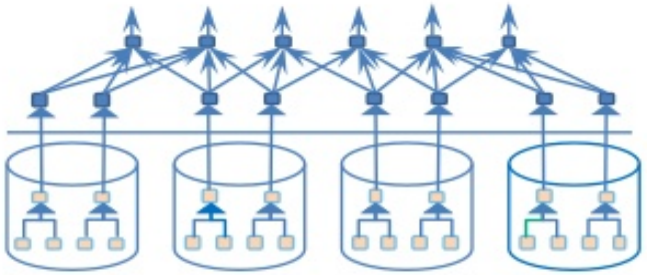


Scalability of throughput



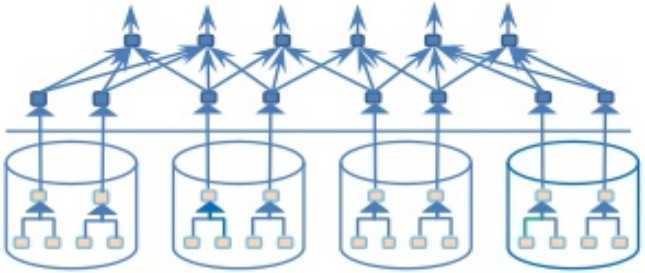
Computation push-down

- Implemented:
  - Filters
  - Projections
  - Count(*)
- Works in progress:
  - Joins
  - Aggregations
- Future work:
  - User-Defined Functions



Computation push-down schemes

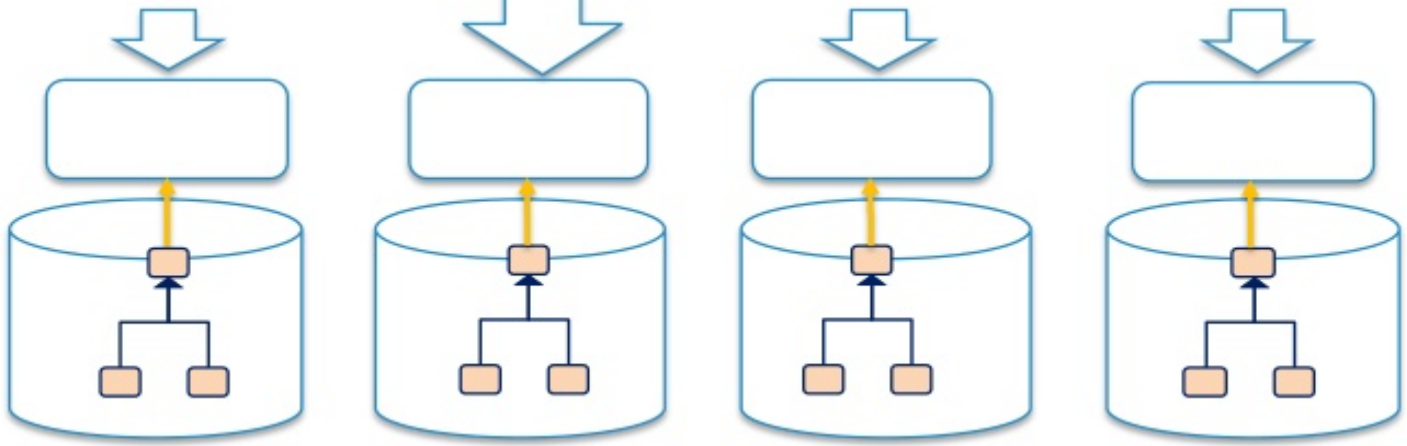
- Per-tuple computation (filtering, projection)
  - Always pushed down
- Joins & aggregations
  - Locality-aware: when join and aggregation keys are co-segmented in Vertica
  - Single query: run single query with join/aggregation inside Vertica



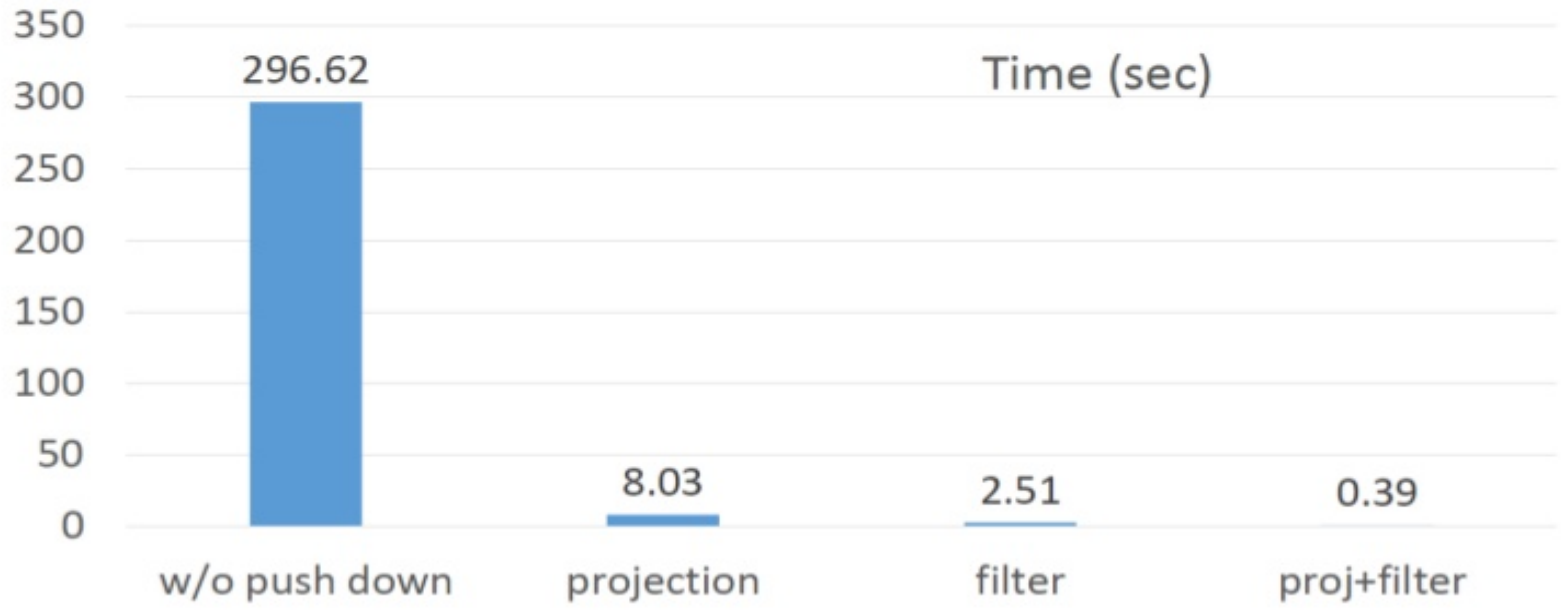
Original Query: select c,d,f from A JOIN B on A.foo = B.foo

Locality-aware push-down query:

```
select c,d,f from
A JOIN B on A.foo = B.foo
where
0 < 0xffffffff & hash(foo) AND
0xffffffff & hash(foo) < 278985 …
```



Performance Impact of Pushdowns



Vertica Analytics

- Geospatial analytics
- Sentiment analysis
- Sessionization of event streams
- Time series pattern matching
- ... (Many more)

The Other Direction

Spark as an ETL Engine for Vertica

The Other Direction

Spark as an ETL Engine for Vertica

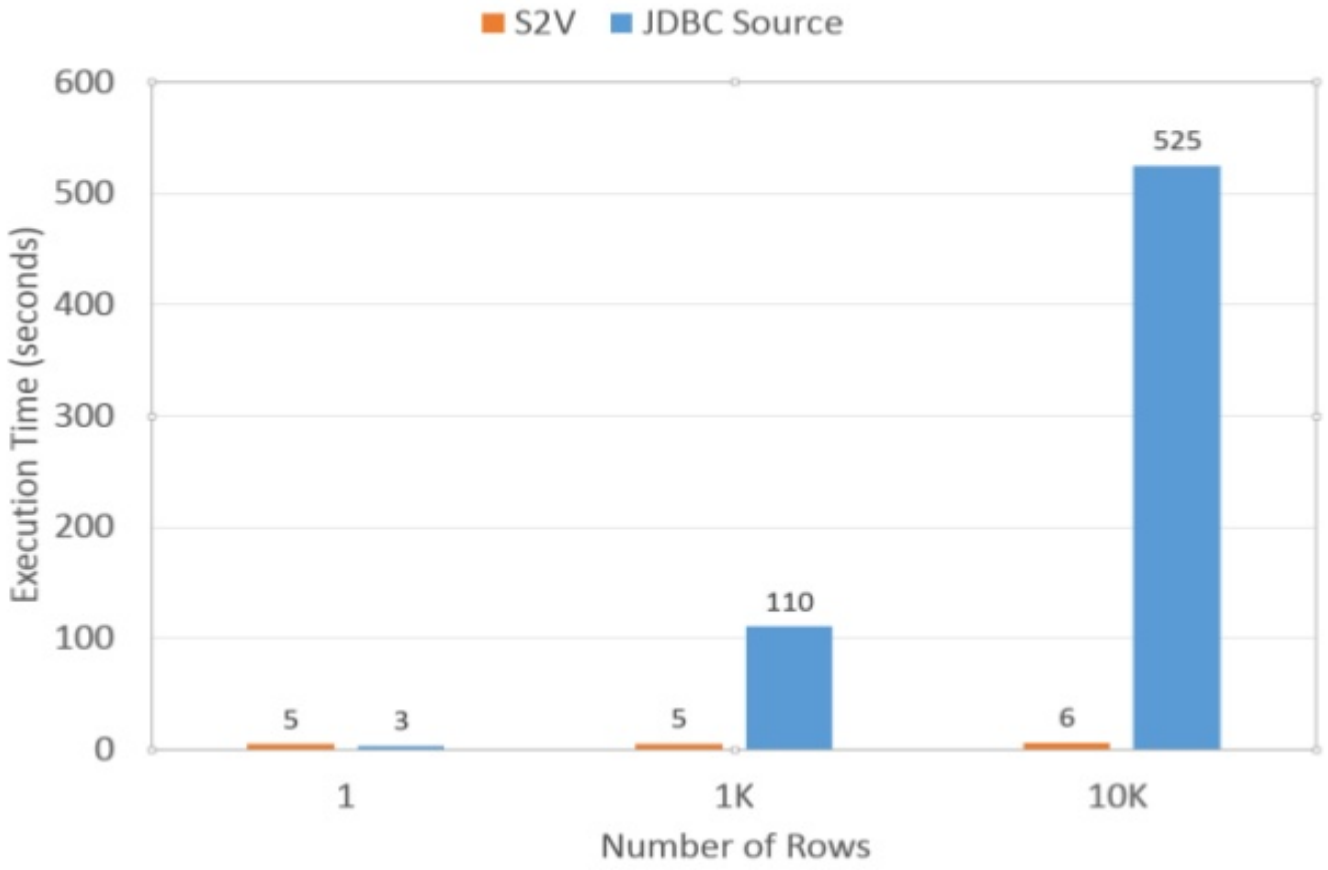
Objective: Want reliable and fast bulk loading

Two Possible Approaches

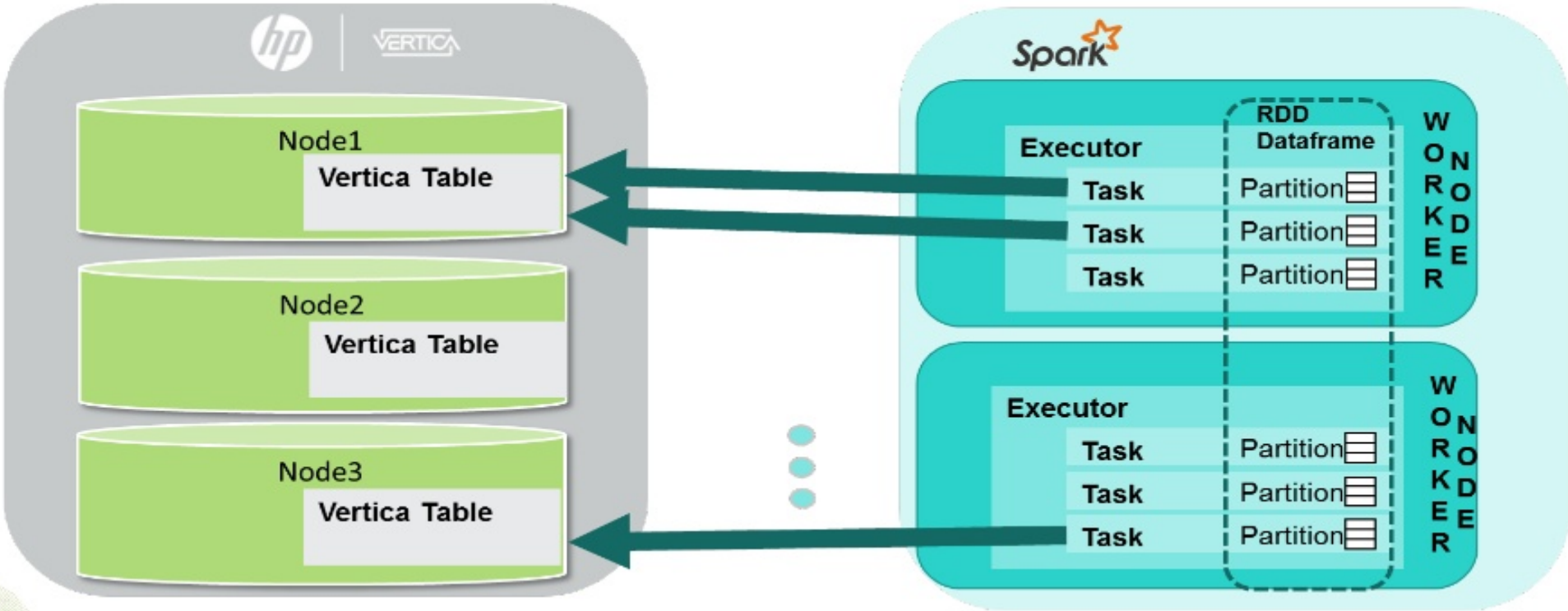
Direct

Indirect (Two-Stage)

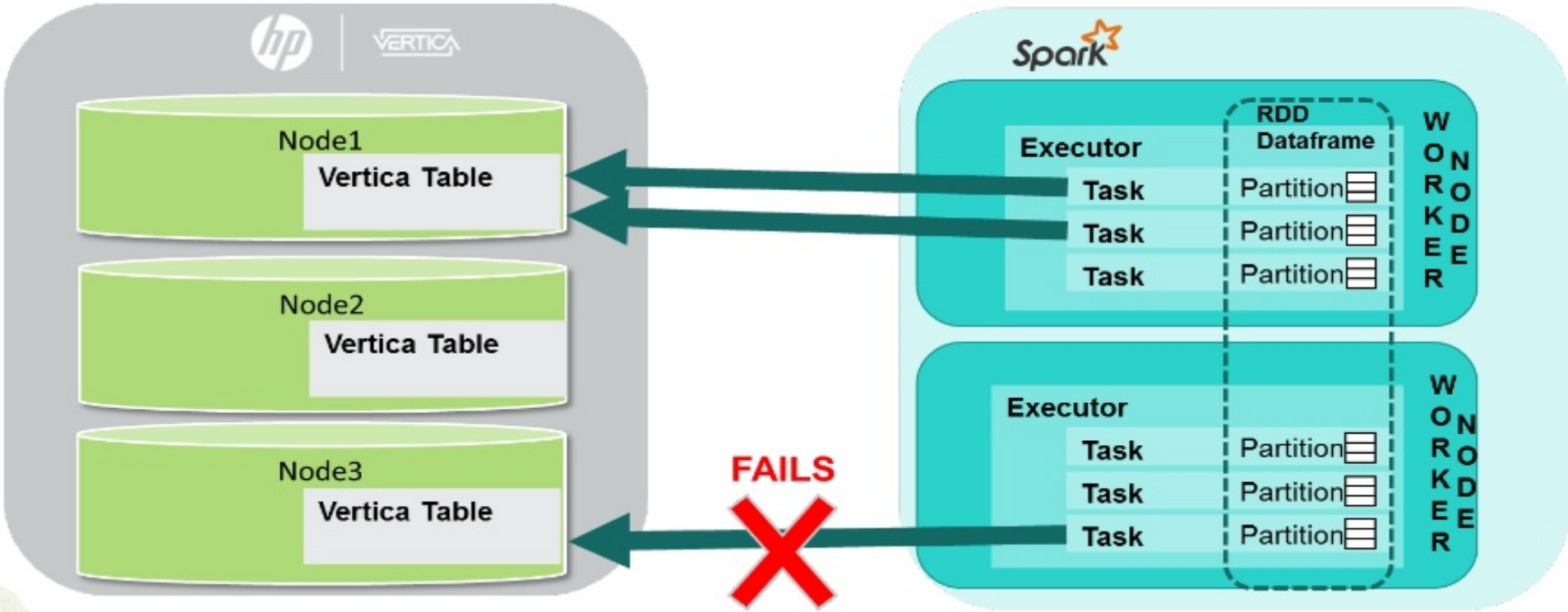
Comparison to JDBC Source



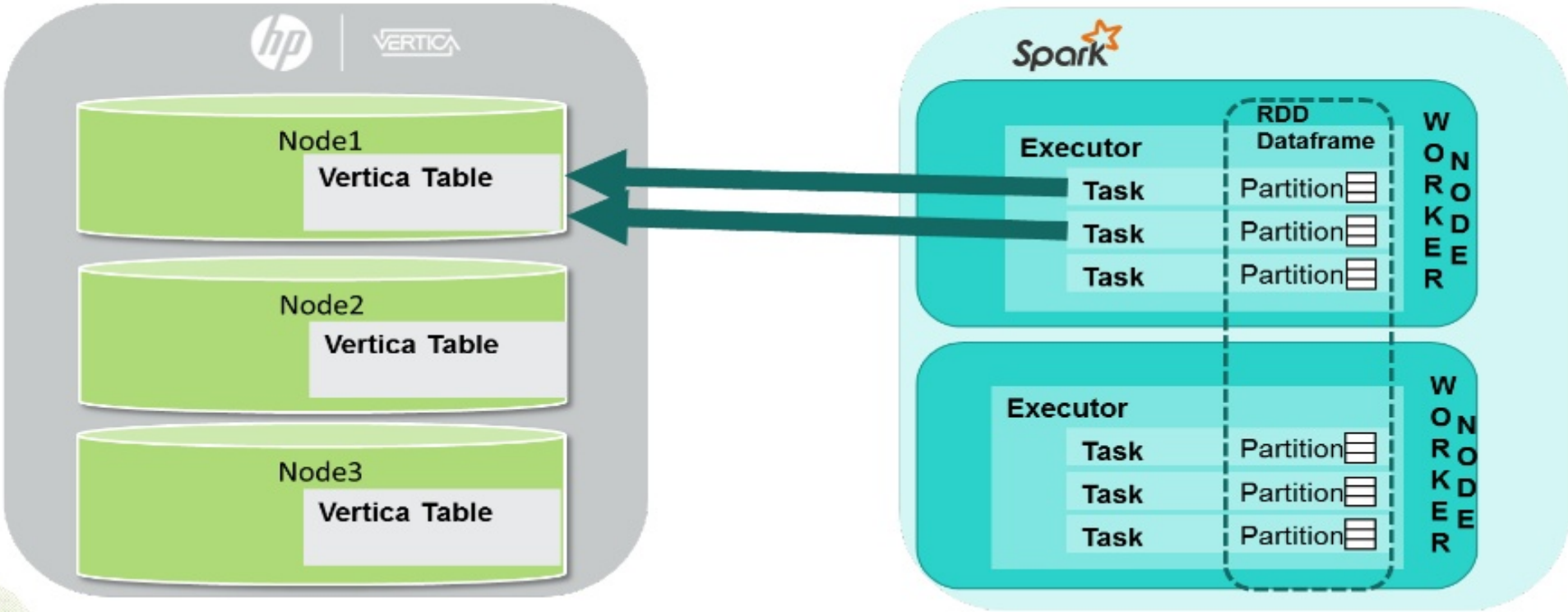
What could happen?



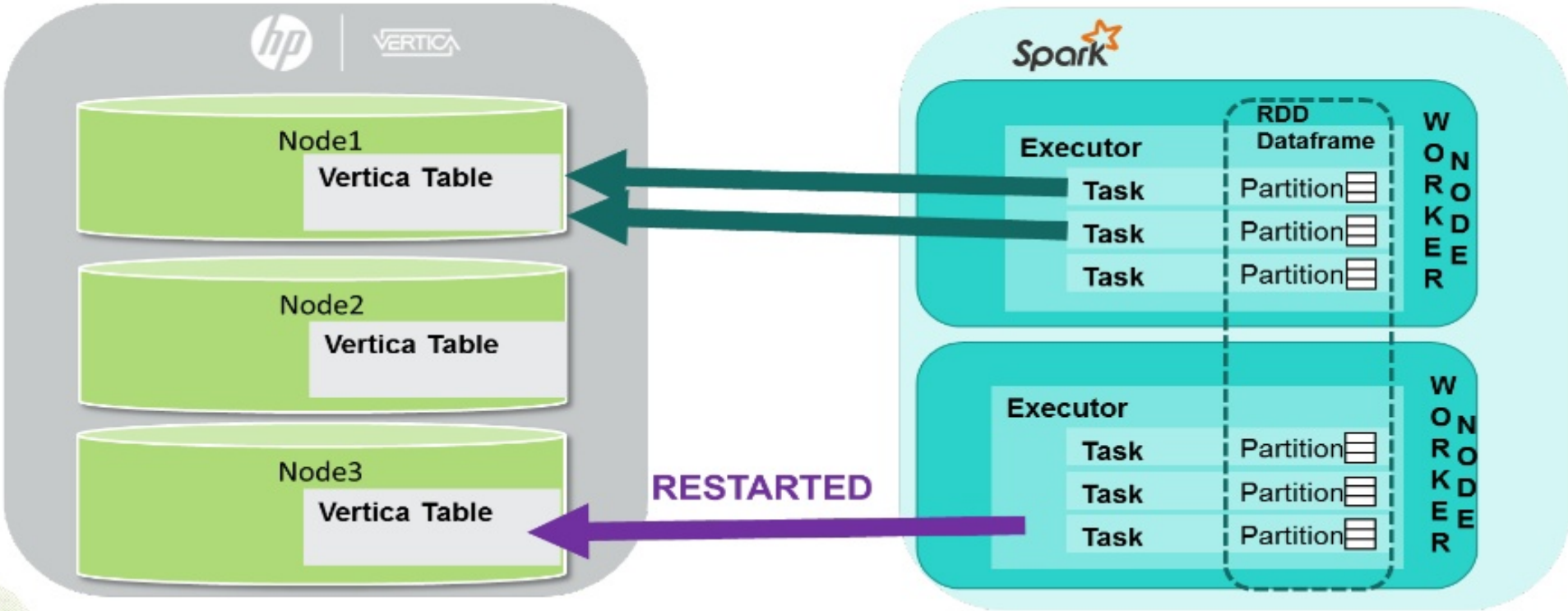
What could happen?



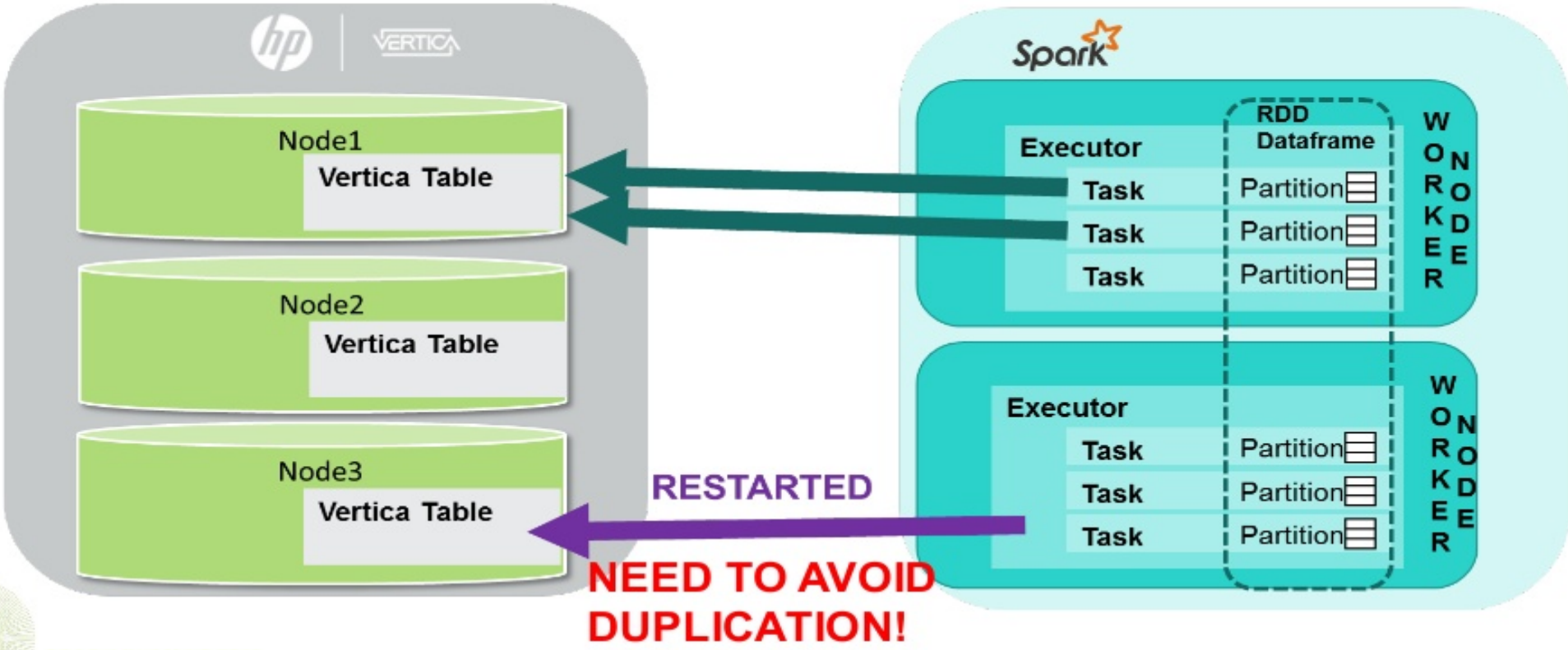
What could happen?



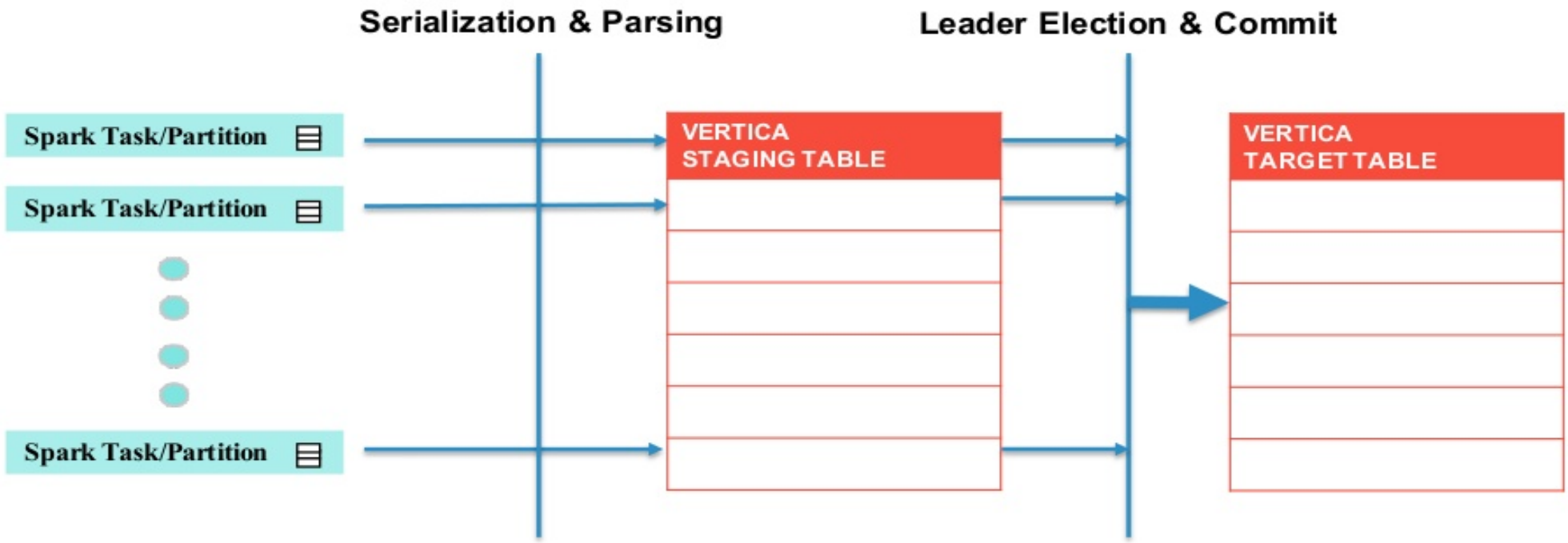
What could happen?



What could happen?

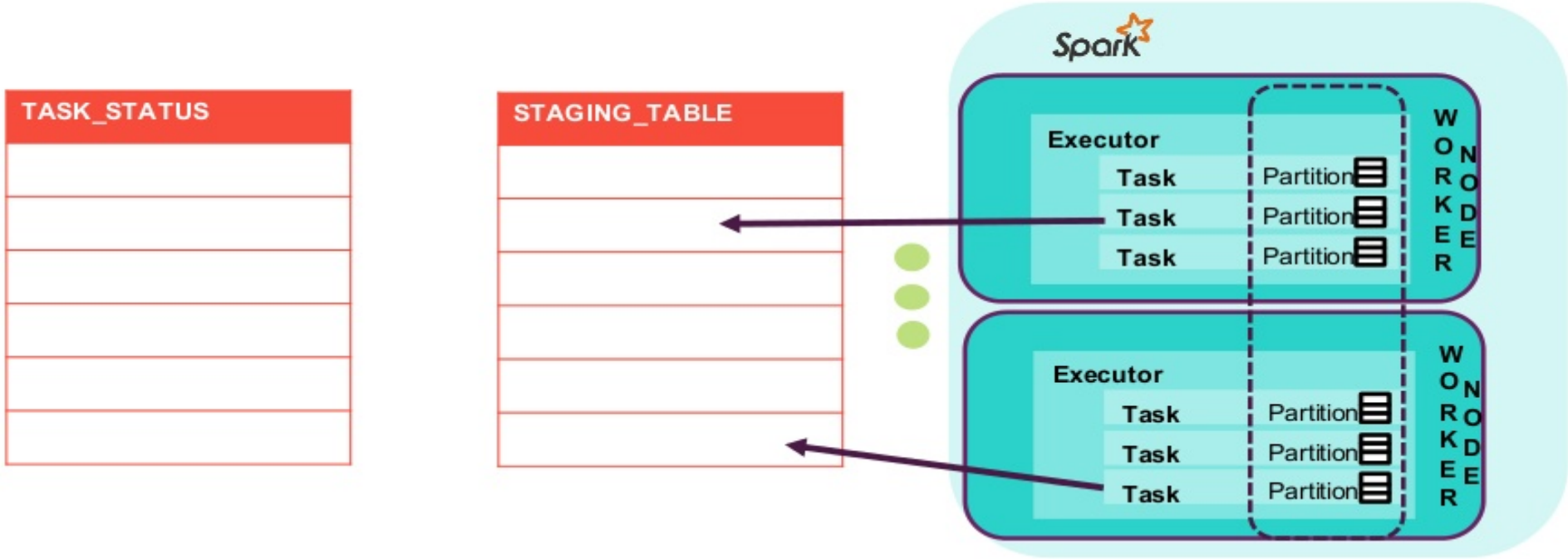


Big-Picture (Direct)



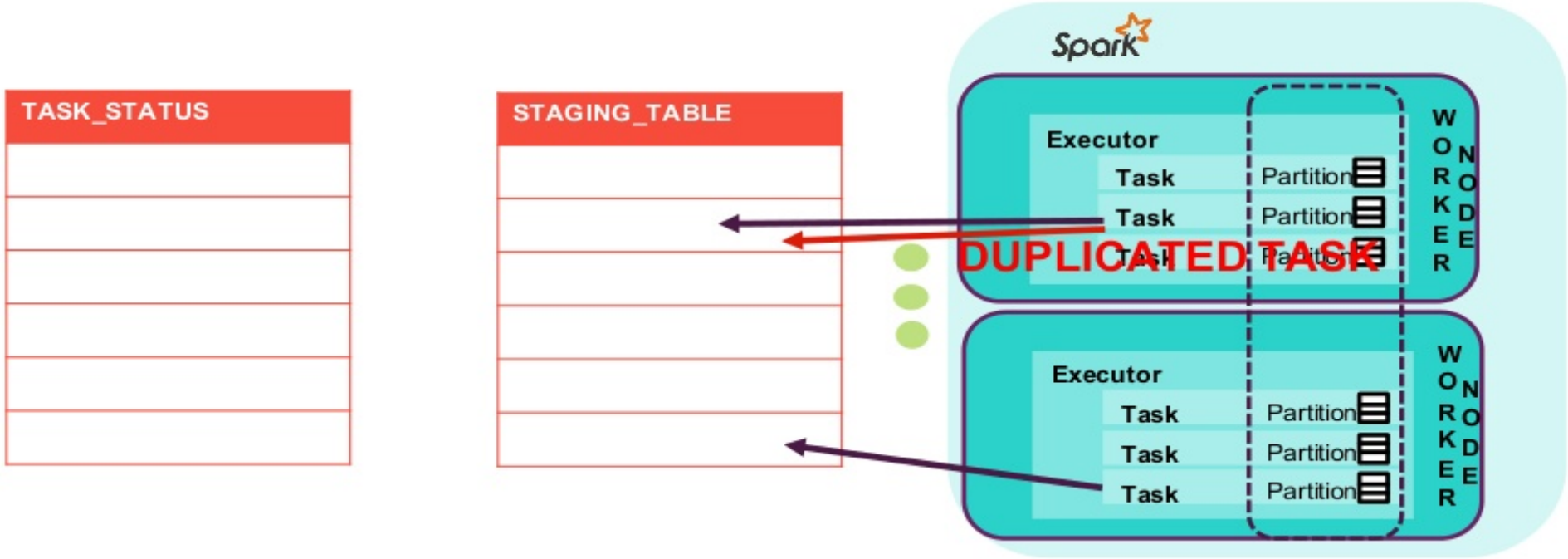
Exactly-once Semantics

Never results in partial or duplicate table loads



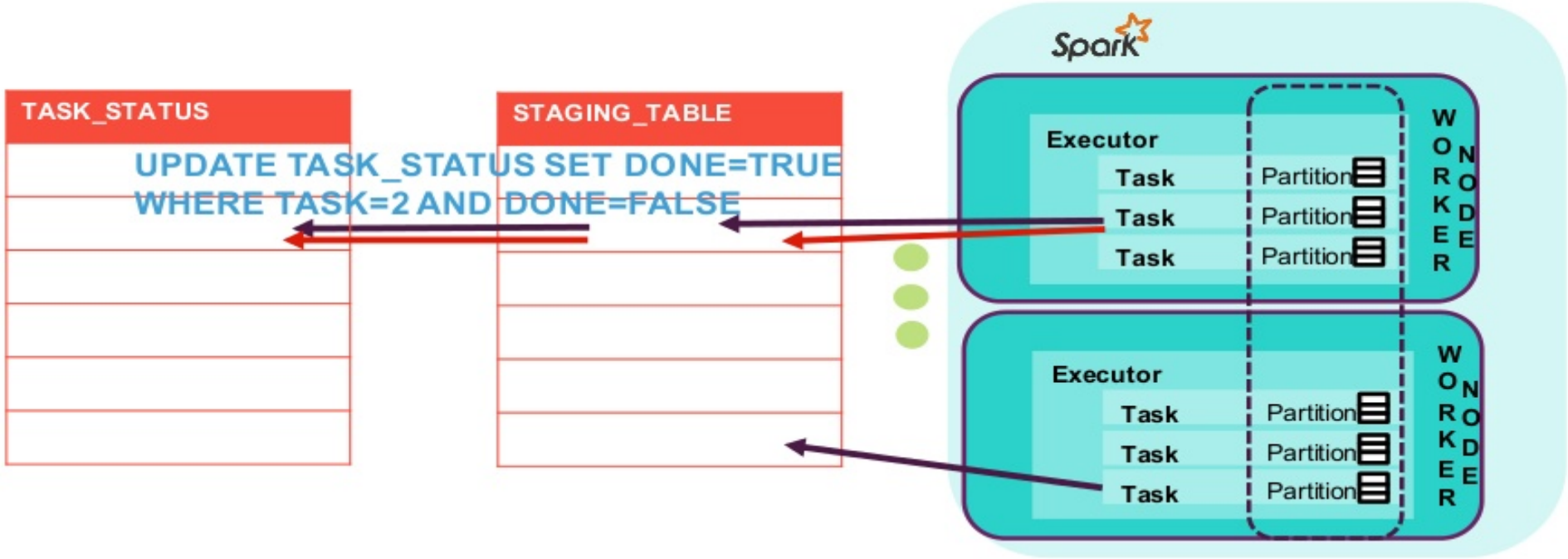
Exactly-once Semantics

Never results in partial or duplicate table loads



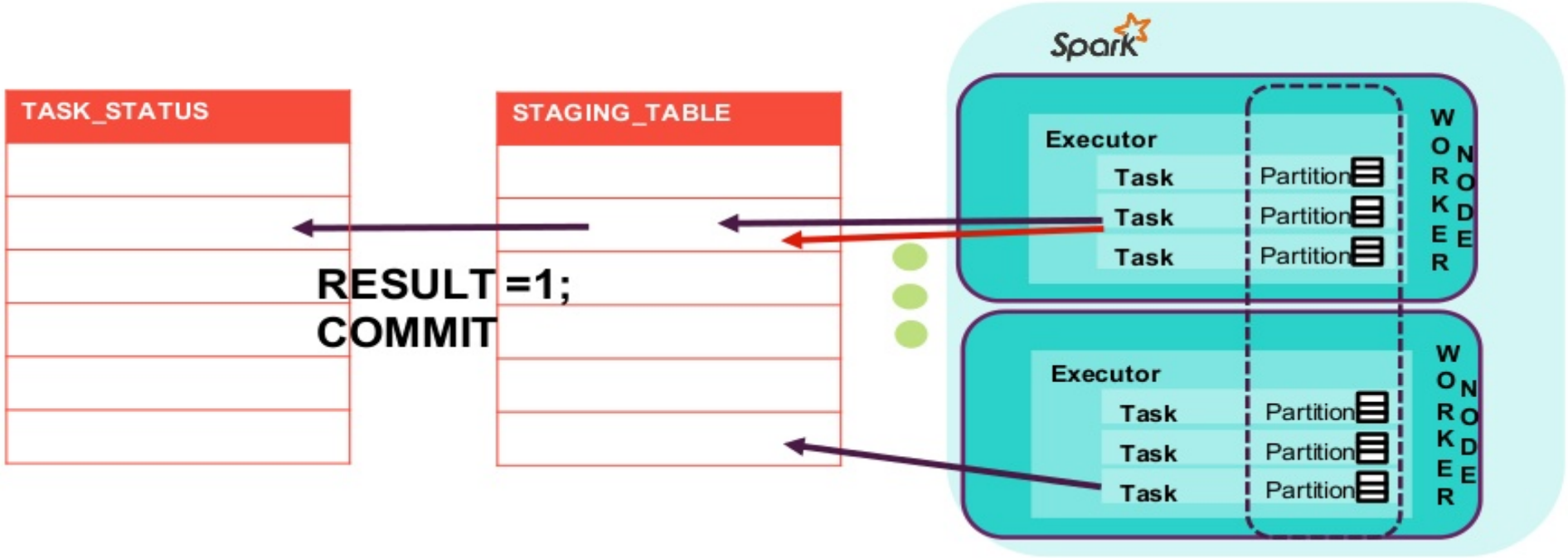
Exactly-once Semantics

Never results in partial or duplicate table loads



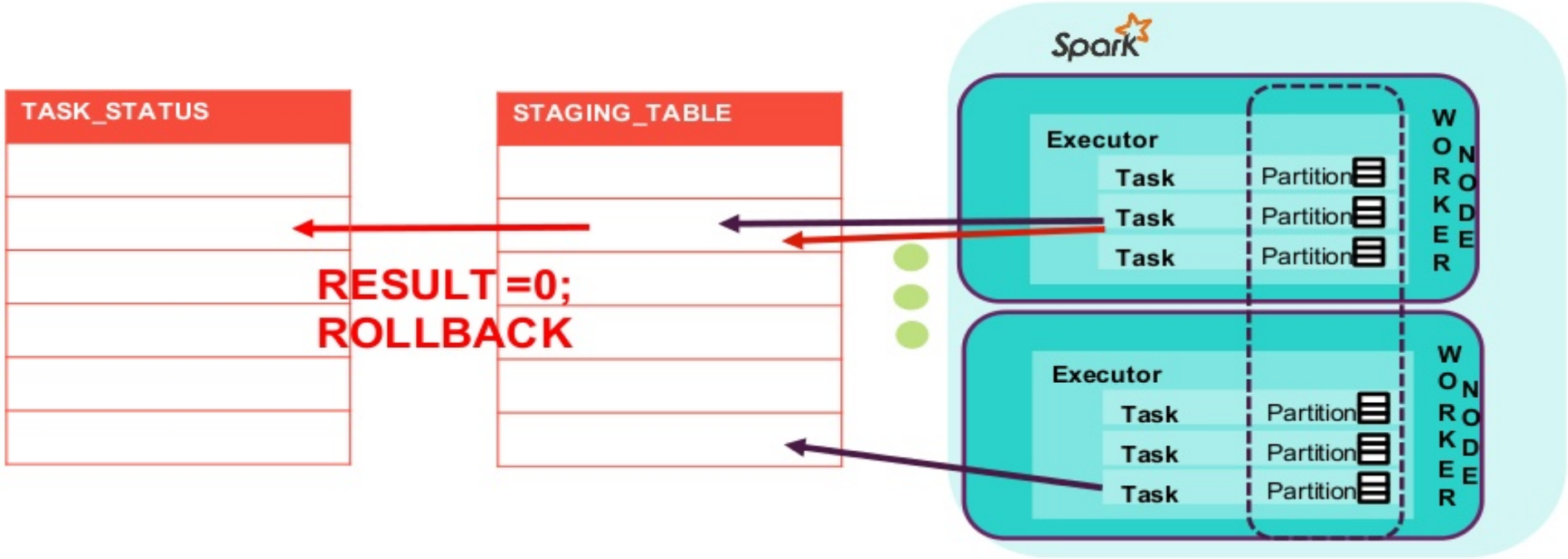
Exactly-once Semantics

Never results in partial or duplicate table loads



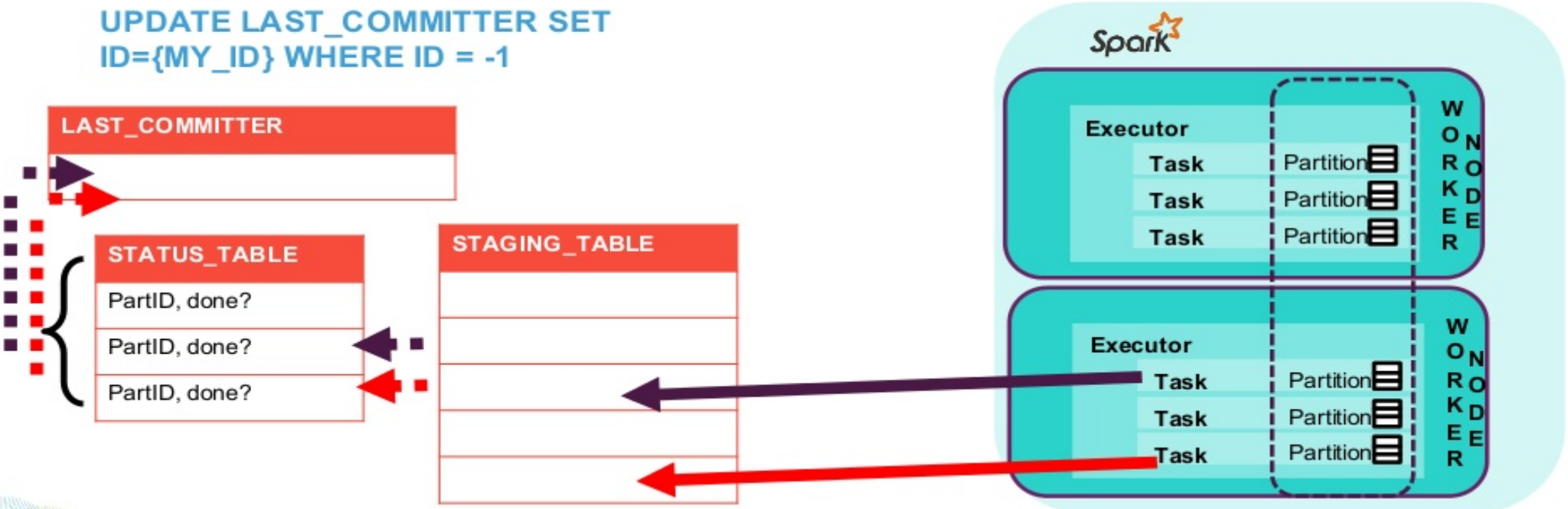
Exactly-once Semantics

Never results in partial or duplicate table loads



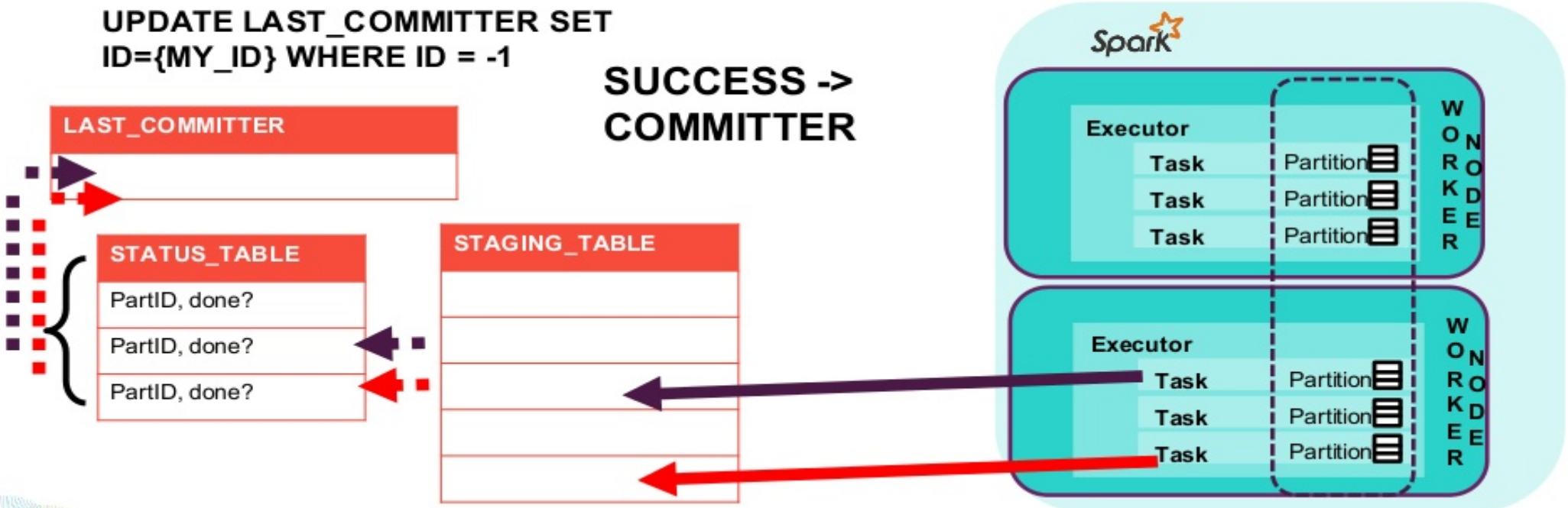
Leader Election

Find Out Which Task Won The Race



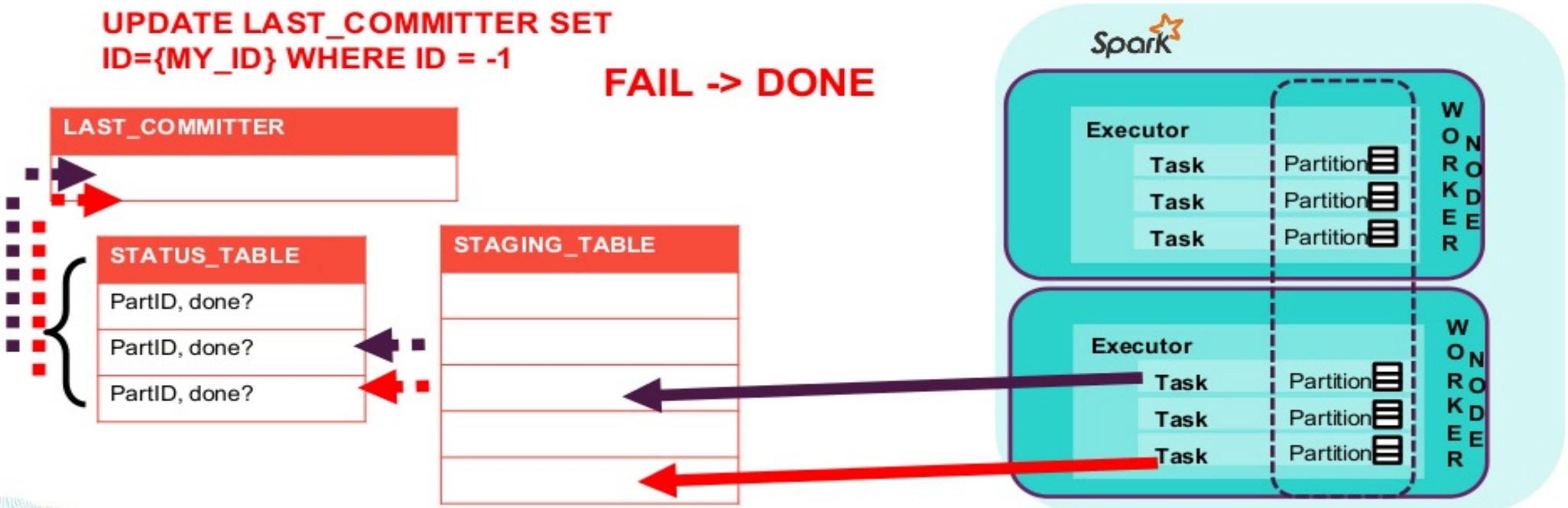
Leader Election

Find Out Which Task Won The Race

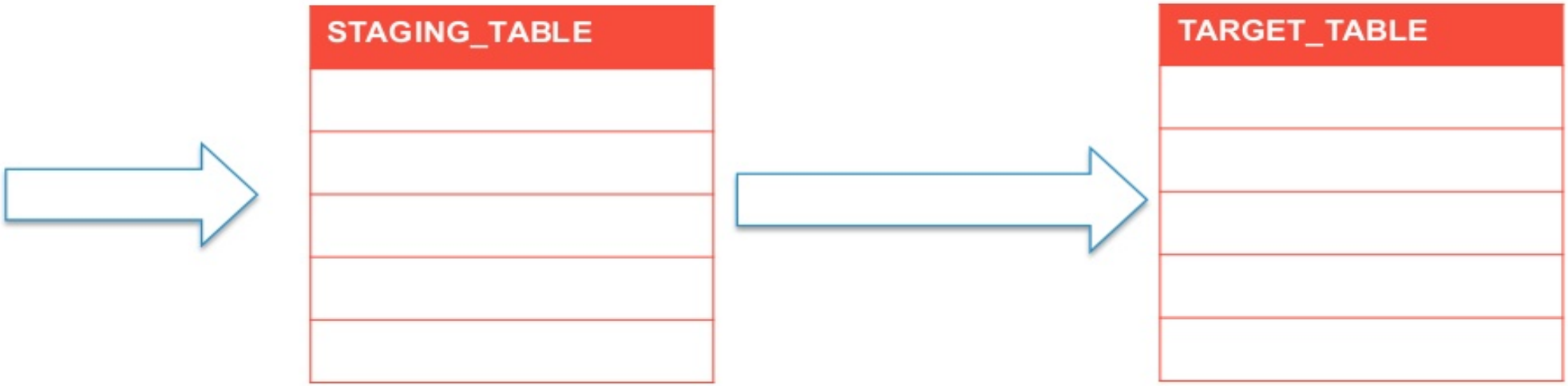


Leader Election

Find Out Which Task Won The Race



Move to Target Table



Direct: Pros & Cons

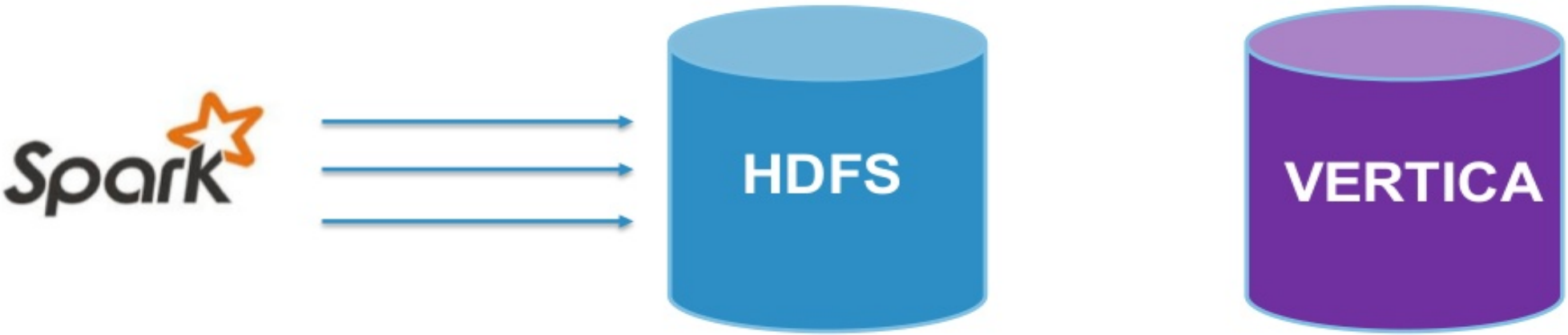
+ Requires no external system beyond Vertica and Spark

+ Less I/O: Does not involve an additional write stage

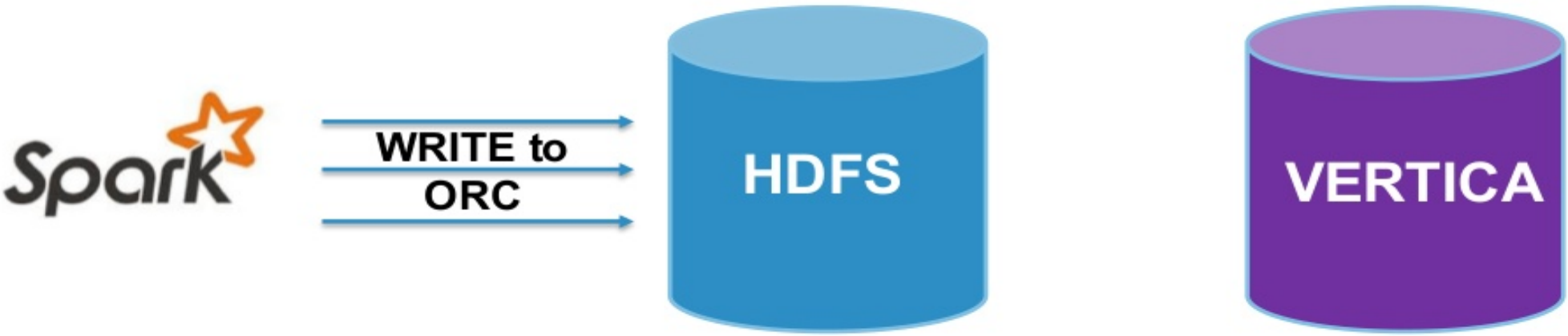
- Exactly-once semantics are non-trivial to ensure

Big-Picture (Two-Stage)

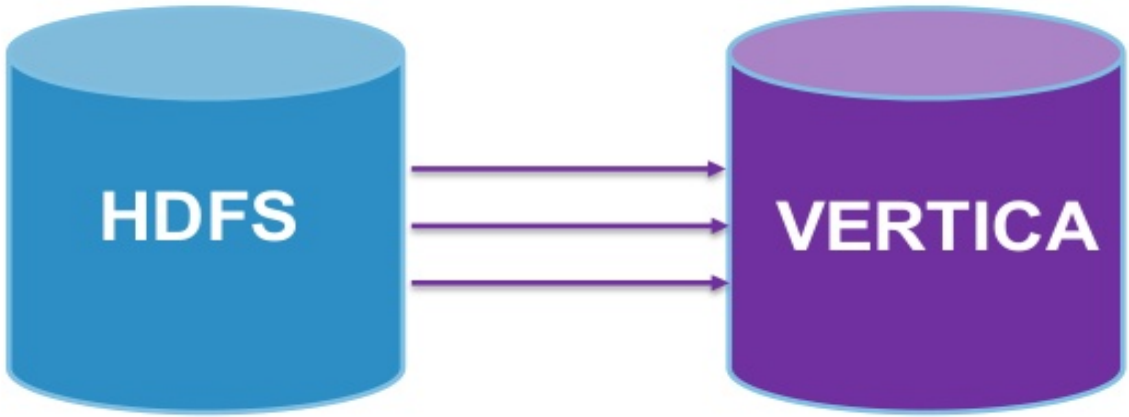
Big-Picture (Two-Stage)



Big-Picture (Two-Stage)

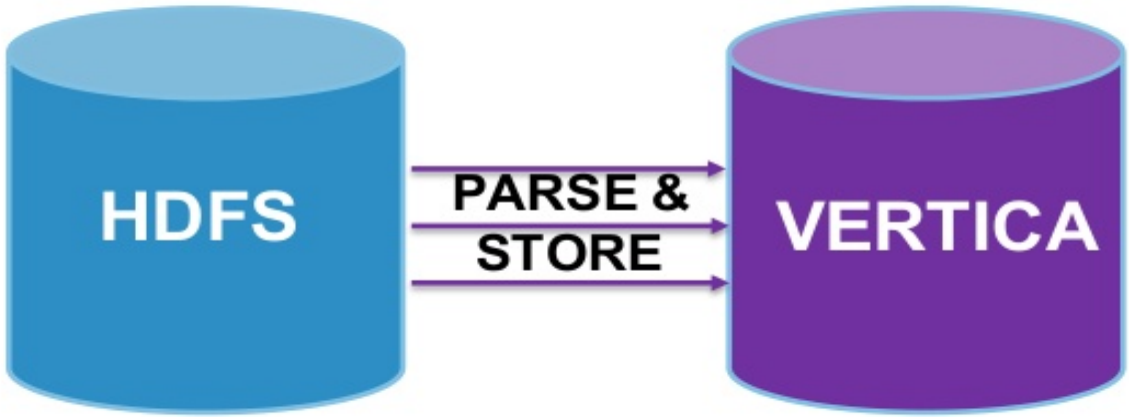


Big-Picture (Two-Stage)



COPY target_table FROM 'hdfs://…' ON ANY NODE ORC;

Big-Picture (Two-Stage)



COPY target_table FROM 'hdfs://…' ON ANY NODE ORC;

Two-Stage: Pros & Cons

+ Utilizes Vertica's built-in parallel COPY command for bulk loads, and may allow for better scalability

+ Uses the optimized ORC reader for parsing (developed in collaboration with Hortonworks).

+ Simpler code logic, since only one query is involved

- More I/O: Requires an additional writing stage.

- Requires an additional dependency on intermediate storage system.

Conclusions

- **Vertica-to-Spark**
  - Utilizes hash-range queries to minimize intra-Vertica data movement
  - Node-pruning eliminates unneeded distributed query processing in Vertica
  - Query pushdown to reduce data transfer and to take advantage of Vertica native analytics.

- **Spark-to-Vertica**
  - Direct connector approach efficiently and reliably coordinates transfer with parallel Spark tasks
  - A two-stage approach with an intermediary staging storage (HDFS) utilizes a single Vertica query to load and may be more efficient on resource consumption on Vertica side

Learn More About – and Try! - HPE Vertica

- Community Edition
  - Free Download 1TB, 3 nodes
  - my.vertica.com/

- Spark Connector – Beta Program
  - https://saas.hpe.com/marketplace/big-data/hpe-vertica-connector-apache-spark

THANK YOU.

Rui Liu (r.liu@hpe.com)
Edward Ma (ema@hpe.com)

SPARK SUMMIT 2016
DATA SCIENCE AND ENGINEERING AT SCALE
JUNE 6-8, 2016 SAN FRANCISCO