HAWQ Architecture

Alexey Grishchenko

HighLoad++

Who I am

Enterprise Architect @ Pivotal

- 7 years in data processing
- 5 years of experience with MPP
- 4 years with Hadoop
- Using HAWQ since the first internal Beta
- Responsible for designing most of the EMEA HAWQ and Greenplum implementations
- Spark contributor
- http://0x0fff.com

Agenda

- What is HAWQ

Agenda

- What is HAWQ
- Why you need it

Agenda

- What is HAWQ
- Why you need it
- HAWQ Components

Agenda

- What is HAWQ
- Why you need it
- HAWQ Components
- HAWQ Design

Agenda

- What is HAWQ
- Why you need it
- HAWQ Components
- HAWQ Design
- Query execution example

Agenda

- What is HAWQ
- Why you need it
- HAWQ Components
- HAWQ Design
- Query execution example
- Competitive solutions

What is  HAWQ

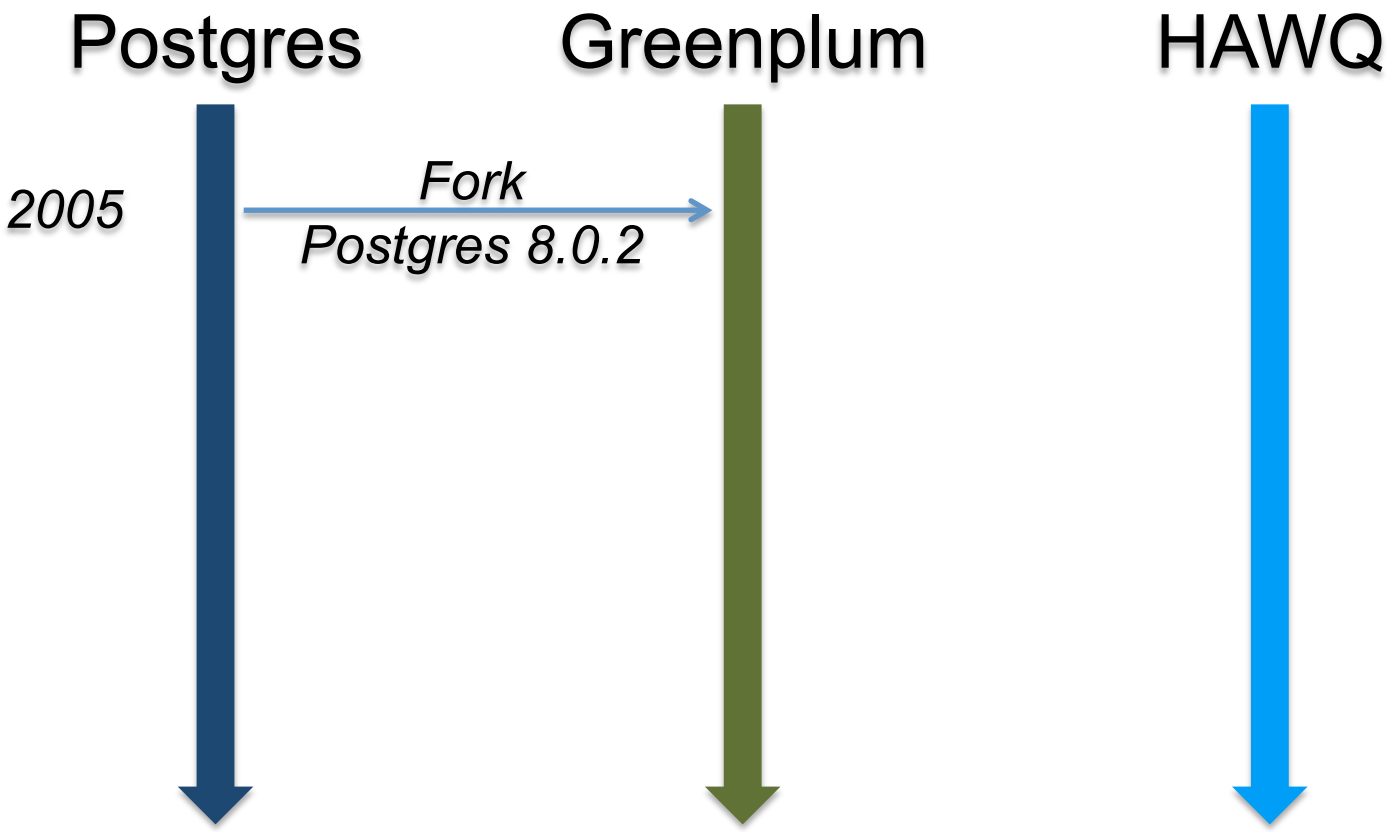
- Analytical SQL-on-Hadoop engine

What is  HAWQ

- Analytical SQL-on-Hadoop engine
- **HA**doop **W**ith **Q**ueries

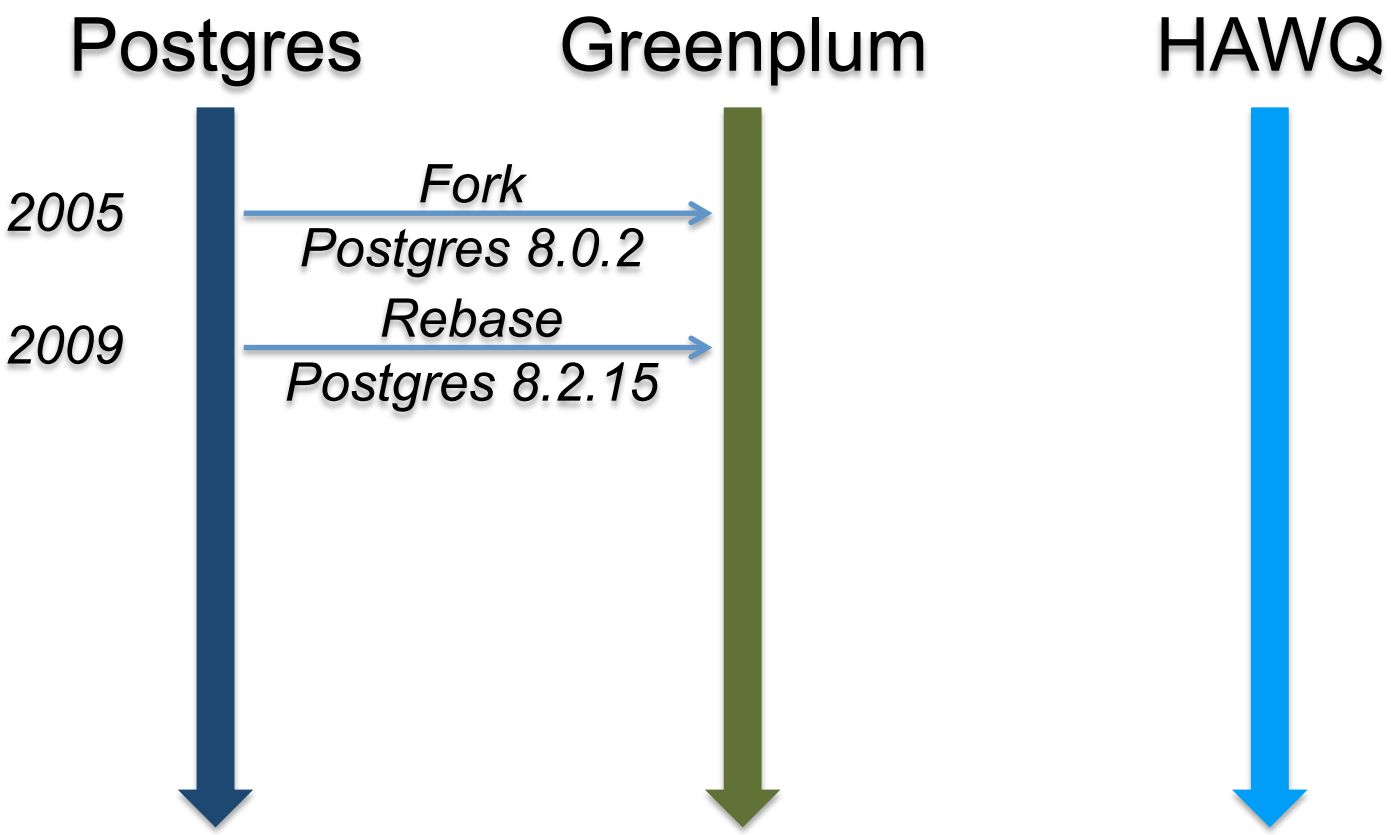
What is HAWQ

- Analytical SQL-on-Hadoop engine
- **HA**doop **W**ith **Q**ueries



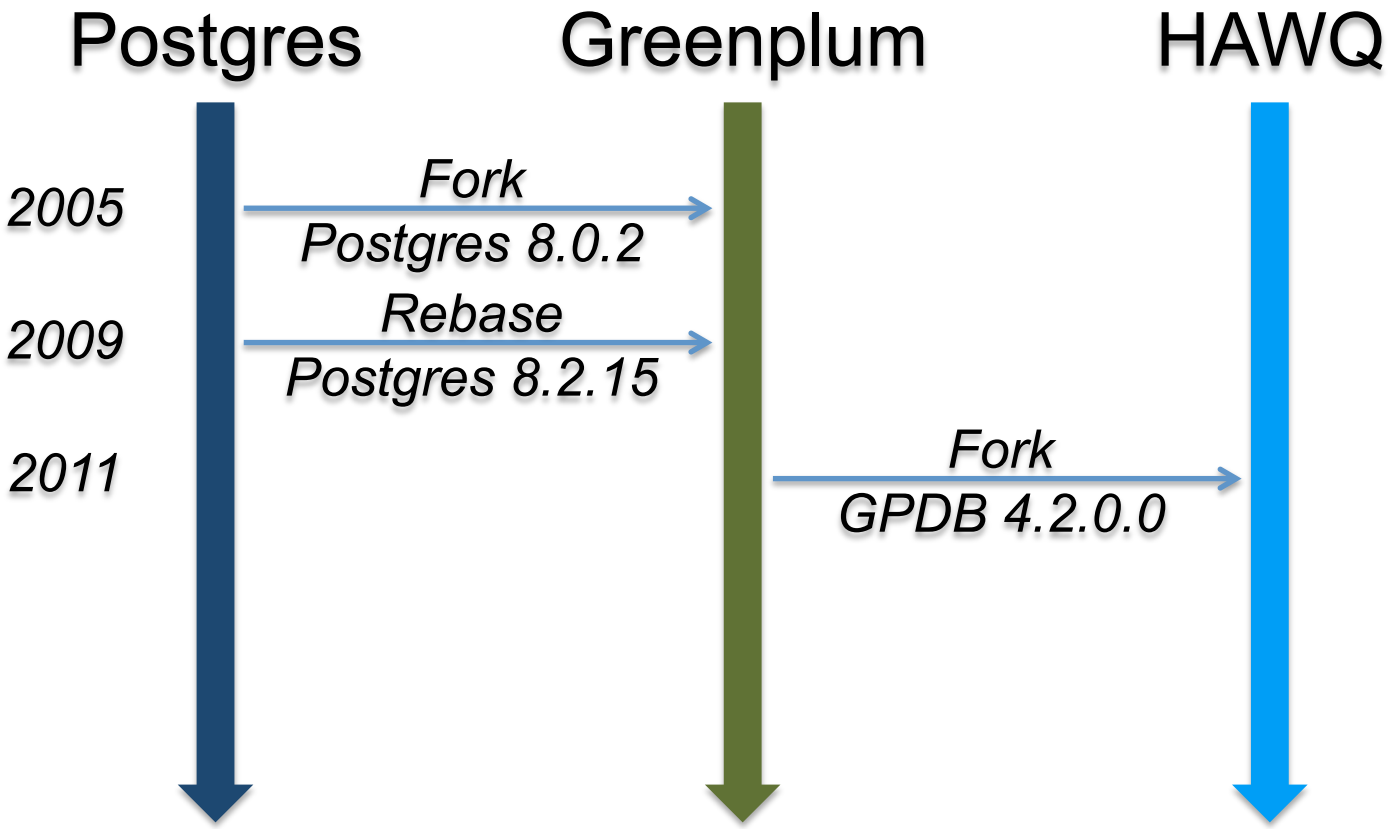
What is HAWQ

- Analytical SQL-on-Hadoop engine
- **HA**doop **W**ith **Q**ueries



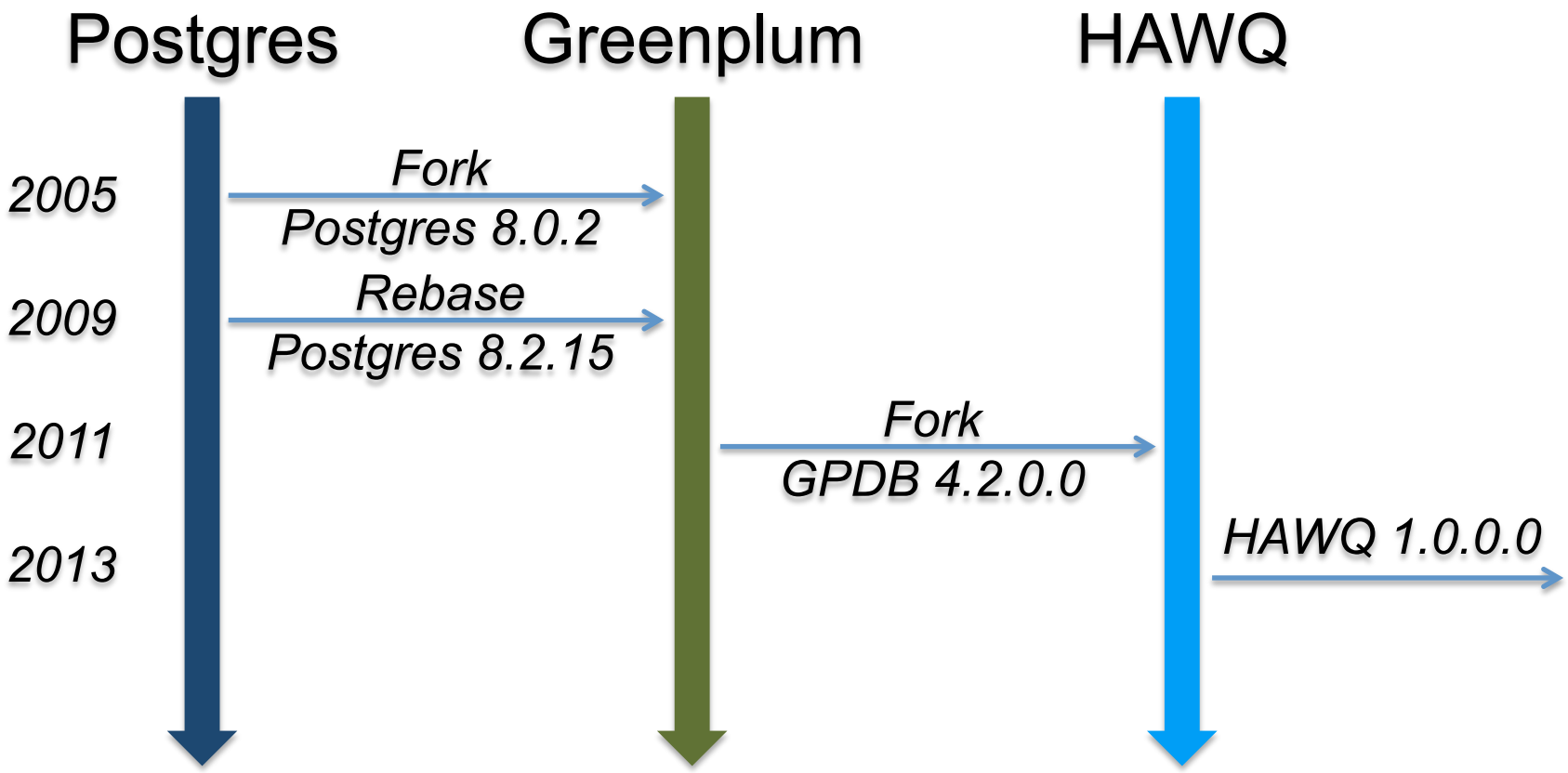
What is HAWQ

- Analytical SQL-on-Hadoop engine
- **HA**doop **W**ith **Q**ueries



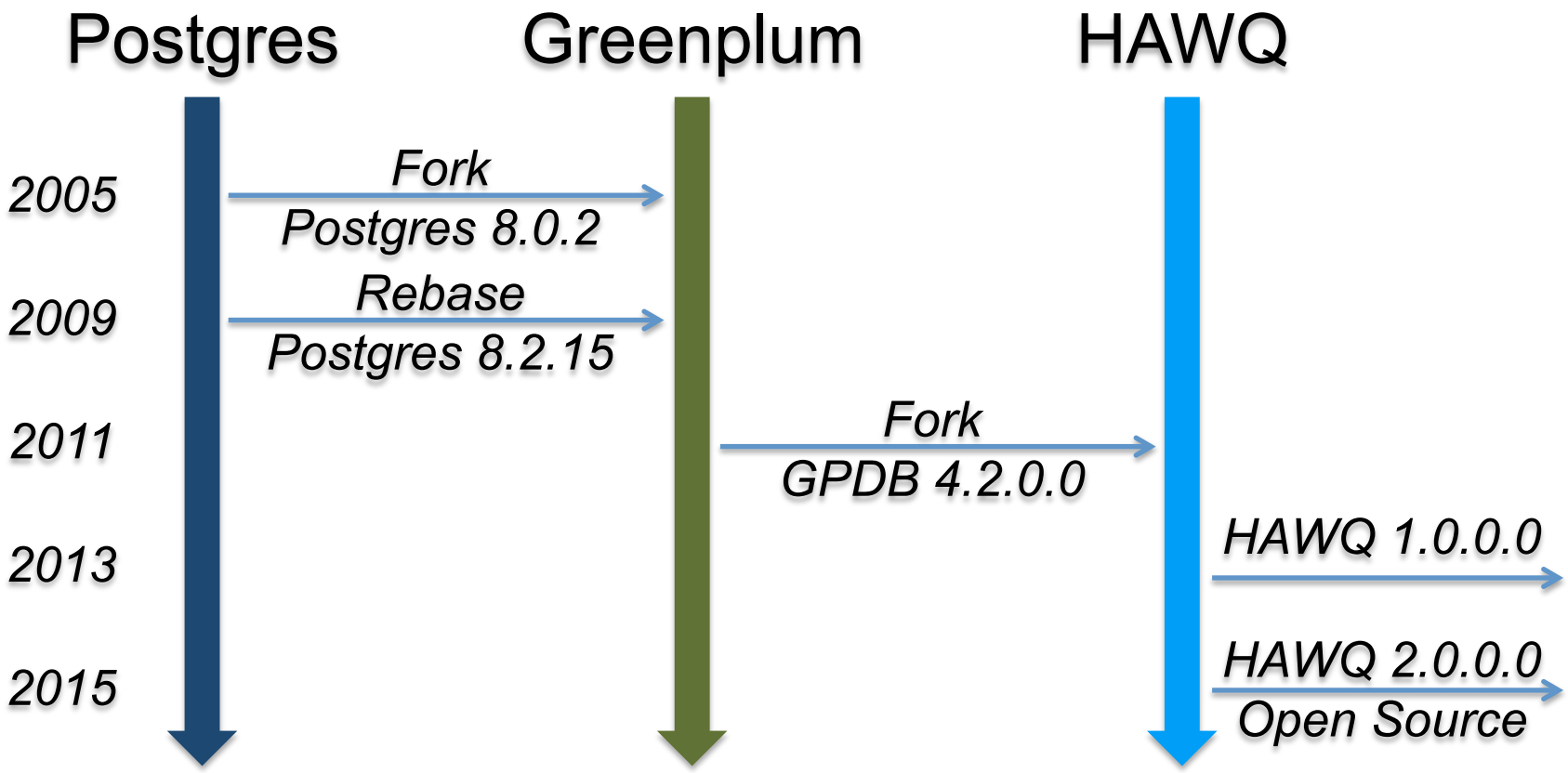
What is HAWQ

- Analytical SQL-on-Hadoop engine
- **HA**doop **W**ith **Q**ueries



What is HAWQ

- Analytical SQL-on-Hadoop engine
- **HA**doop **W**ith **Q**ueries



HAWQ is …

- 1’500’000 C and C++ lines of code

HAWQ is …

- 1’500’000 C and C++ lines of code
  - 200’000 of them in headers only

HAWQ is …

- 1’500’000 C and C++ lines of code
  - 200’000 of them in headers only
- 180’000 Python LOC

HAWQ is …

- 1’500’000 C and C++ lines of code
  - 200’000 of them in headers only
- 180’000 Python LOC
- 60’000 Java LOC

HAWQ is …

- 1’500’000 C and C++ lines of code
  - 200’000 of them in headers only
- 180’000 Python LOC
- 60’000 Java LOC
- 23’000 Makefile LOC

HAWQ is …

- 1’500’000 C and C++ lines of code
  - 200’000 of them in headers only
- 180’000 Python LOC
- 60’000 Java LOC
- 23’000 Makefile LOC
- 7’000 Shell scripts LOC

HAWQ is …

- 1’500’000 C and C++ lines of code
  - 200’000 of them in headers only
- 180’000 Python LOC
- 60’000 Java LOC
- 23’000 Makefile LOC
- 7’000 Shell scripts LOC
- More than 50 enterprise customers

HAWQ is …

- 1’500’000 C and C++ lines of code
  - 200’000 of them in headers only
- 180’000 Python LOC
- 60’000 Java LOC
- 23’000 Makefile LOC
- 7’000 Shell scripts LOC
- More than 50 enterprise customers
  - More than 10 of them in EMEA

Apache HAWQ

- Apache HAWQ (incubating) from 09’2015
  - http://hawq.incubator.apache.org
  - https://github.com/apache/incubator-hawq
- What’s in Open Source
  - Sources of HAWQ 2.0 alpha
  - HAWQ 2.0 beta is planned for 2015’Q4
  - HAWQ 2.0 GA is planned for 2016’Q1
- *Community is yet young – come and join!*

Why do we need it?

Why do we need it?

- SQL-interface for BI solutions to the Hadoop data complaint with ANSI SQL-92, -99, -2003

Why do we need it?

- SQL-interface for BI solutions to the Hadoop data complaint with ANSI SQL-92, -99, -2003
  - Example - 5000-line query with a number of window function generated by Cognos

Why do we need it?

- SQL-interface for BI solutions to the Hadoop data complaint with ANSI SQL-92, -99, -2003
  - Example - 5000-line query with a number of window function generated by Cognos
- Universal tool for ad hoc analytics on top of Hadoop data

Why do we need it?

- SQL-interface for BI solutions to the Hadoop data complaint with ANSI SQL-92, -99, -2003
  - Example - 5000-line query with a number of window function generated by Cognos
- Universal tool for ad hoc analytics on top of Hadoop data
  - Example - parse URL to extract protocol, host name, port, GET parameters

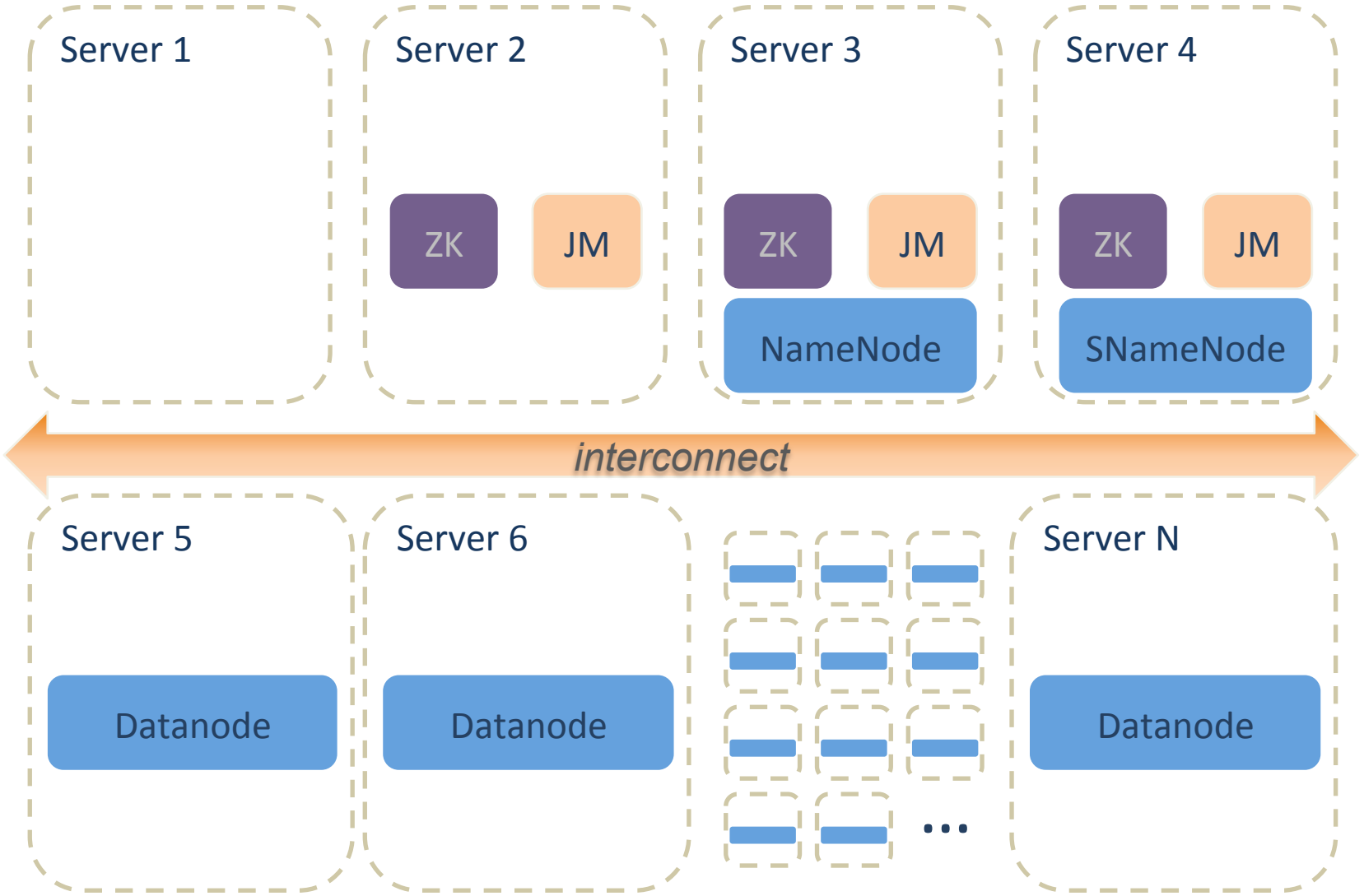
Why do we need it?

- SQL-interface for BI solutions to the Hadoop data complaint with ANSI SQL-92, -99, -2003
  - Example - 5000-line query with a number of window function generated by Cognos
- Universal tool for ad hoc analytics on top of Hadoop data
  - Example - parse URL to extract protocol, host name, port, GET parameters
- Good performance

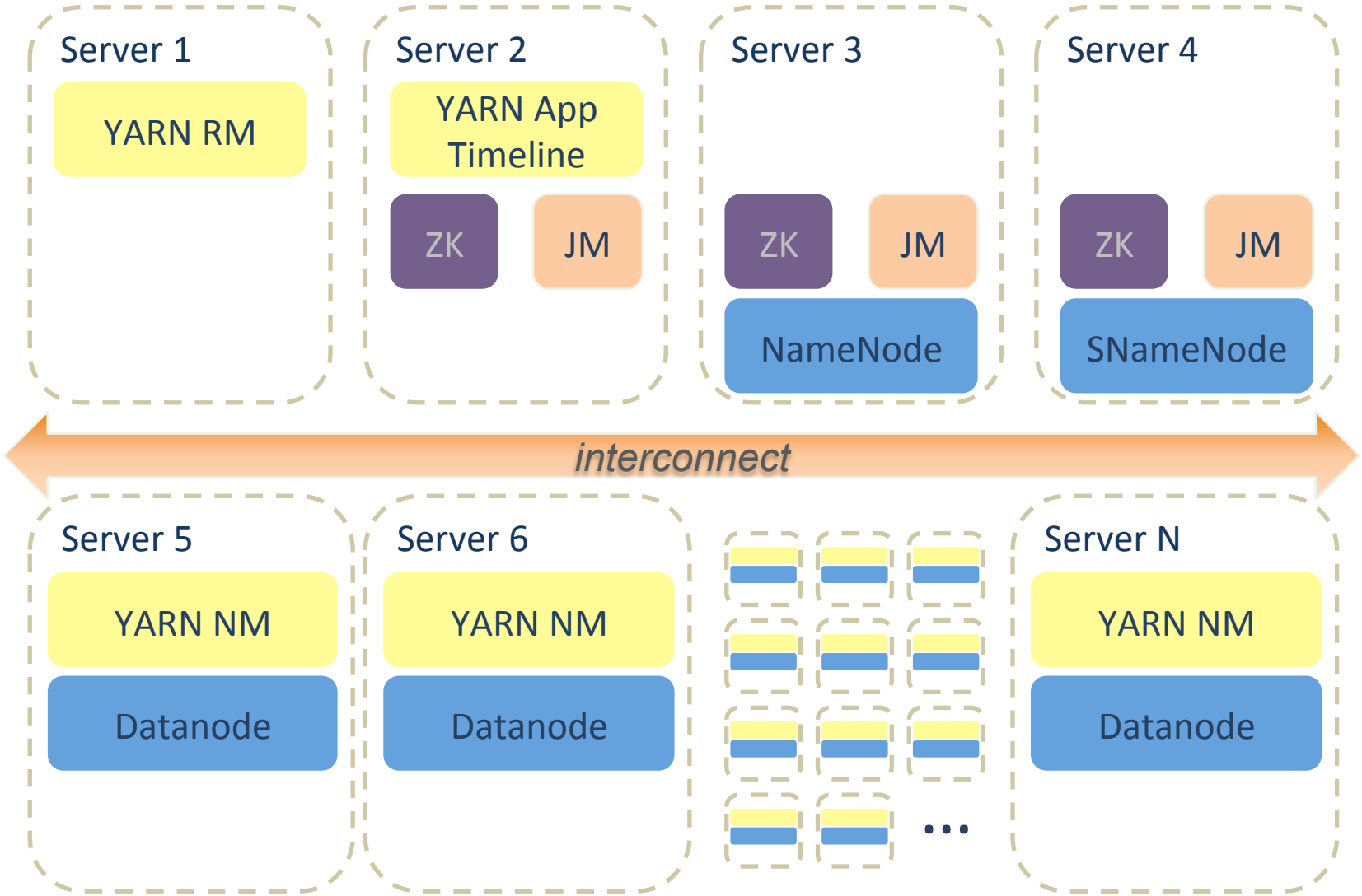
Why do we need it?

- SQL-interface for BI solutions to the Hadoop data complaint with ANSI SQL-92, -99, -2003
  - Example - 5000-line query with a number of window function generated by Cognos
- Universal tool for ad hoc analytics on top of Hadoop data
  - Example - parse URL to extract protocol, host name, port, GET parameters
- Good performance
  - How many times the data would hit the HDD during a single Hive query?

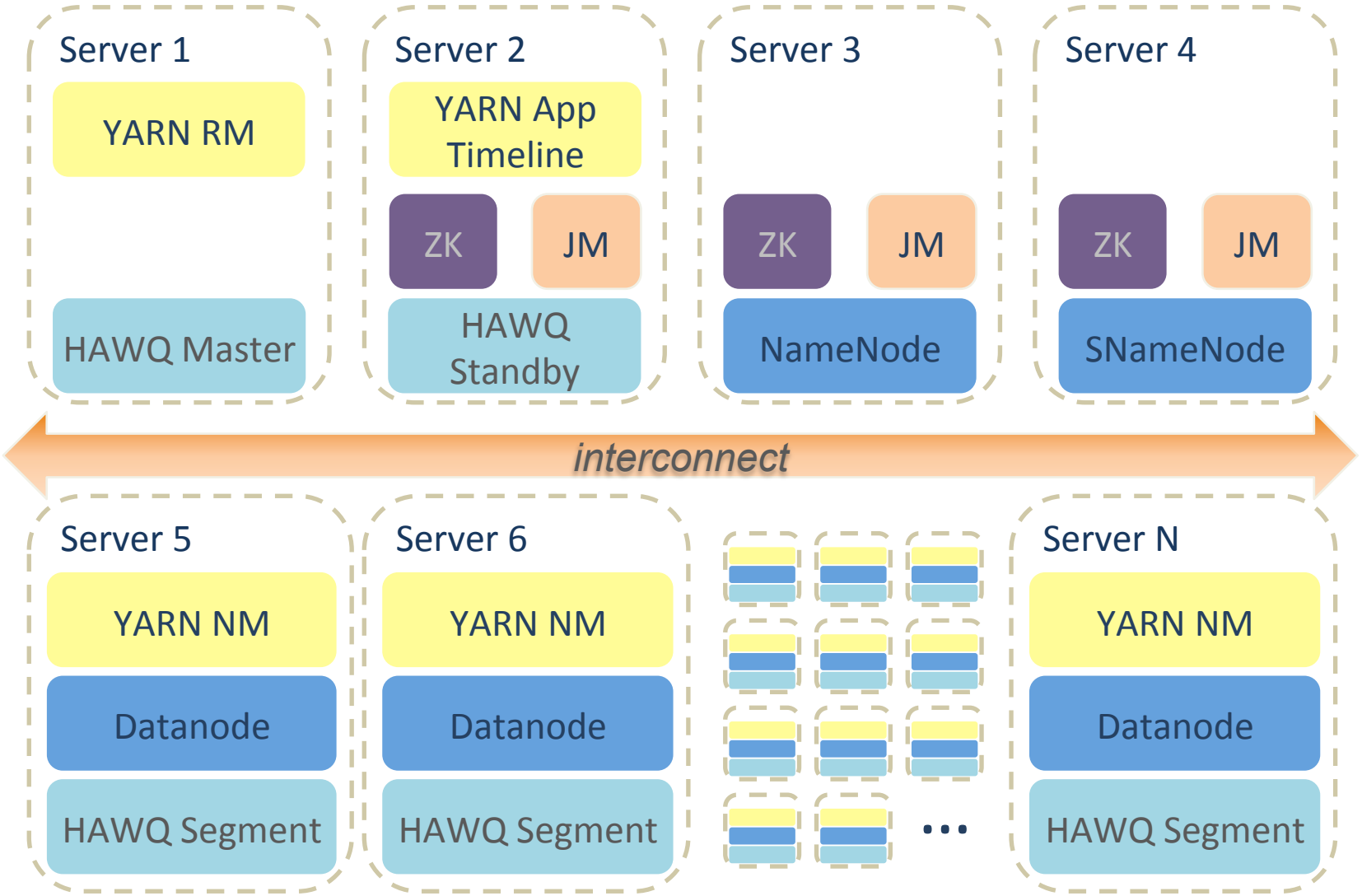
HAWQ Cluster



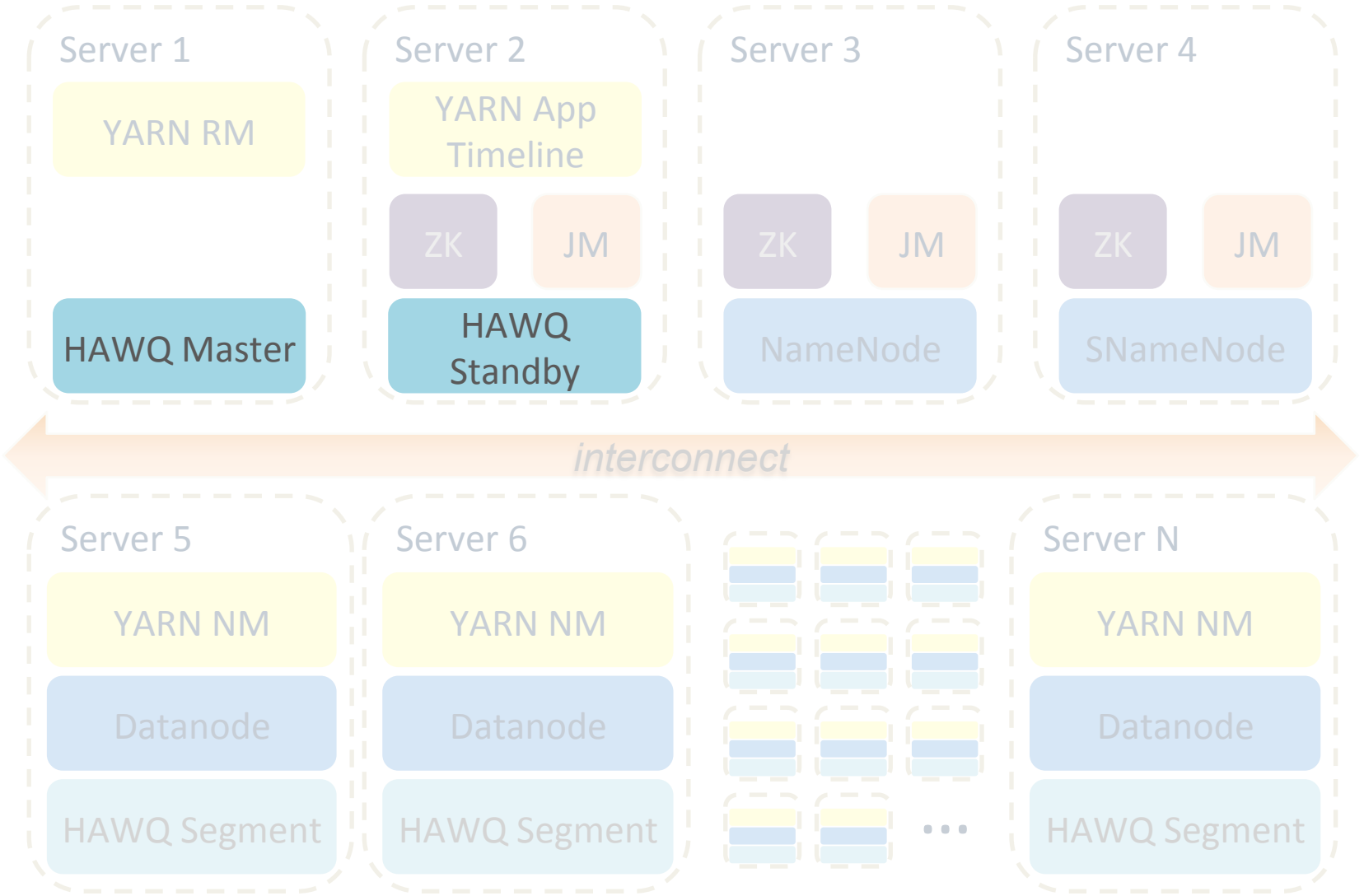
HAWQ Cluster



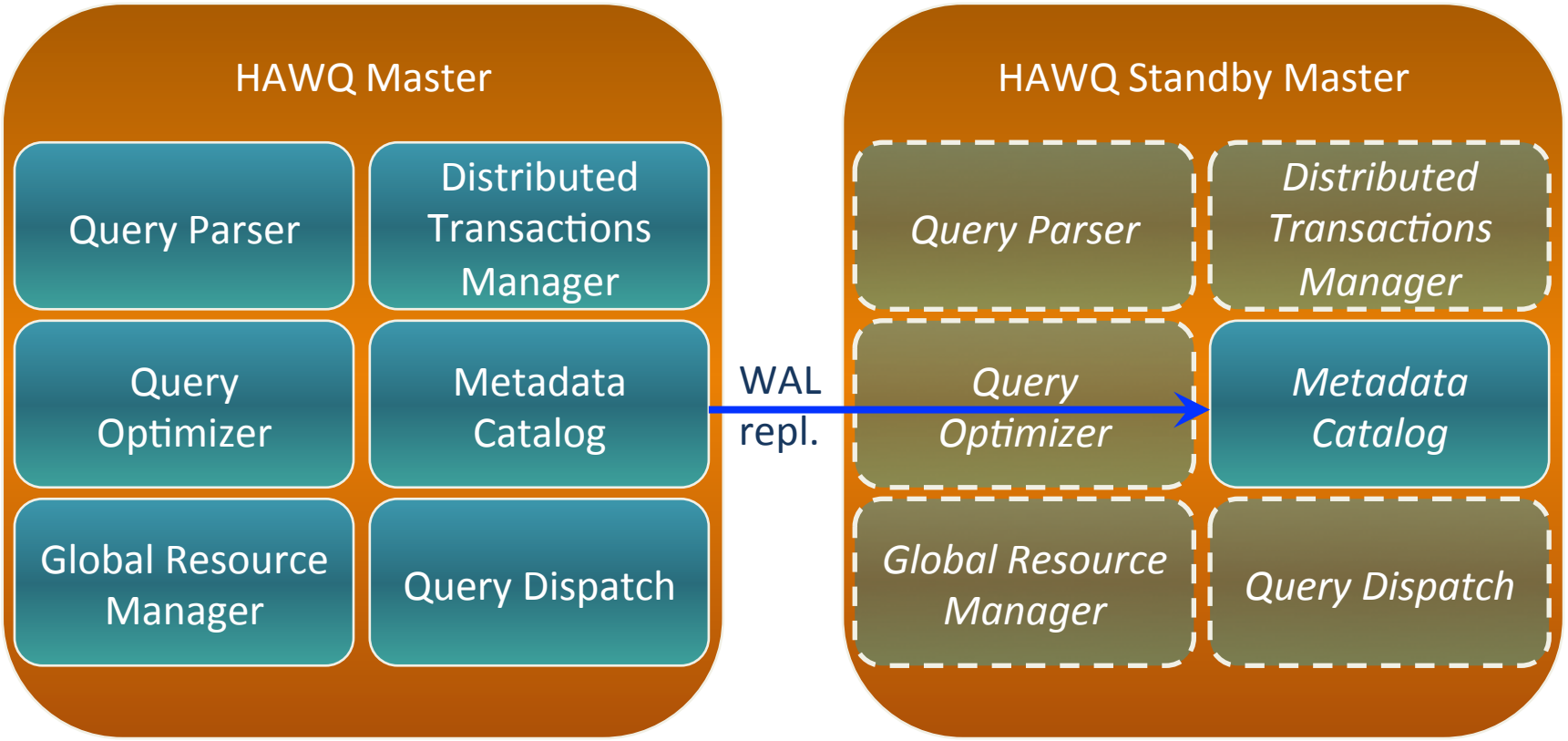
HAWQ Cluster



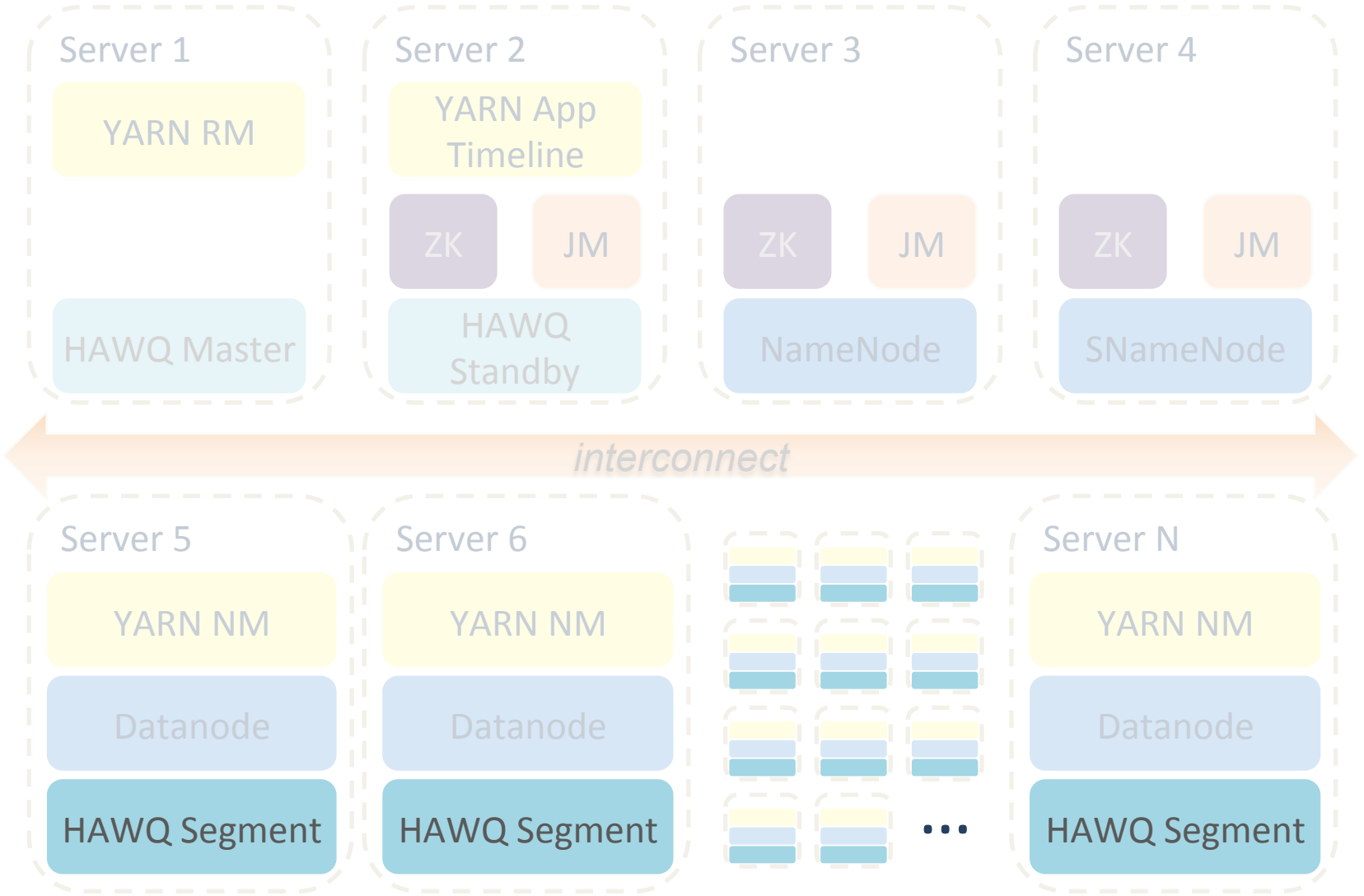
Master Servers



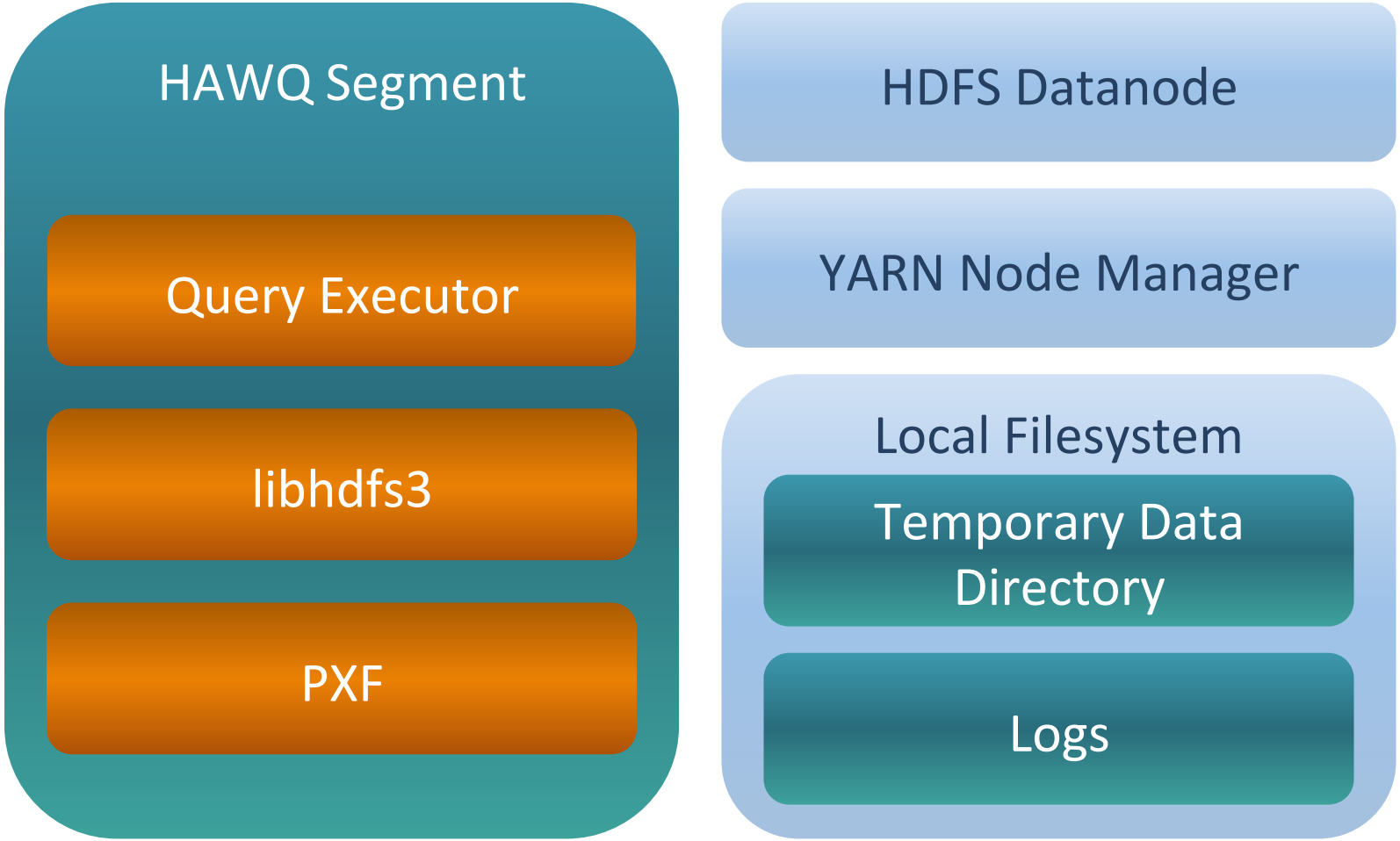
Master Servers



Segments



Segments



Metadata

- HAWQ metadata structure is similar to Postgres catalog structure

Metadata

- HAWQ metadata structure is similar to Postgres catalog structure
- Statistics
  - Number of rows and pages in the table

Metadata

- HAWQ metadata structure is similar to Postgres catalog structure
- Statistics
  - Number of rows and pages in the table
  - Most common values for each field

Metadata

- HAWQ metadata structure is similar to Postgres catalog structure
- Statistics
  - Number of rows and pages in the table
  - Most common values for each field
  - Histogram of values distribution for each field

Metadata

- HAWQ metadata structure is similar to Postgres catalog structure
- Statistics
  - Number of rows and pages in the table
  - Most common values for each field
  - Histogram of values distribution for each field
  - Number of unique values in the field

Metadata

- HAWQ metadata structure is similar to Postgres catalog structure
- Statistics
  - Number of rows and pages in the table
  - Most common values for each field
  - Histogram of values distribution for each field
  - Number of unique values in the field
  - Number of null values in the field

Metadata

- HAWQ metadata structure is similar to Postgres catalog structure
- Statistics
  - Number of rows and pages in the table
  - Most common values for each field
  - Histogram of values distribution for each field
  - Number of unique values in the field
  - Number of null values in the field
  - Average width of the field in bytes

Statistics

- No Statistics

  How many rows would produce the join of two tables?

Statistics

- No Statistics

  How many rows would produce the join of two tables?

  - ✓ *From 0 to infinity*

Statistics

- No Statistics

  How many rows would produce the join of two tables?

  - ✓ *From 0 to infinity*

- Row Count

  How many rows would produce the join of two 1000-row tables?

Statistics

- No Statistics

  How many rows would produce the join of two tables?

  - ✓ *From 0 to infinity*

- Row Count

  How many rows would produce the join of two 1000-row tables?

  - ✓ *From 0 to 1’000’000*

Statistics

- No Statistics

  How many rows would produce the join of two tables?

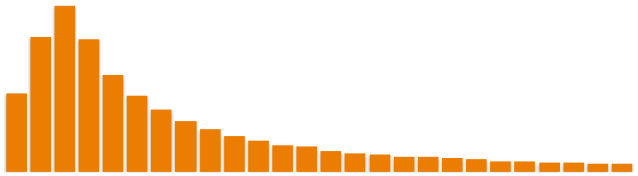
  - ✓ *From 0 to infinity*

- Row Count

  How many rows would produce the join of two 1000-row tables?

  - ✓ *From 0 to 1'000'000*

- Histograms and MCV



  How many rows would produce the join of two 1000-row tables, with known field cardinality, values distribution diagram, number of nulls, most common values?

Statistics

- No Statistics

  How many rows would produce the join of two tables?

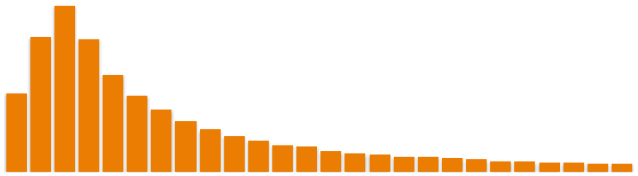
  - ✓ *From 0 to infinity*

- Row Count

  How many rows would produce the join of two 1000-row tables?

  - ✓ *From 0 to 1’000’000*

- Histograms and MCV



  How many rows would produce the join of two 1000-row tables, with known field cardinality, values distribution diagram, number of nulls, most common values?

  - ✓ *~ From 500 to 1’500*

Metadata

- Table structure information

ID	Name	Num	Price
1	Яблоко	10	50
2	Груша	20	80
3	Банан	40	40
4	Апельсин	25	50
5	Киви	5	120
6	Арбуз	20	30
7	Дыня	40	100
8	Ананас	35	90

Metadata

- Table structure information
  - Distribution fields

ID	Name	Num	Price
1	Яблоко	10	50
2	Груша	20	80
3	Банан	40	40
4	Апельсин	25	50
5	Киви	5	120
6	Арбуз	20	30
7	Дыня	40	100
8	Ананас	35	90

Metadata

- Table structure information
  - Distribution fields
  - Number of hash buckets

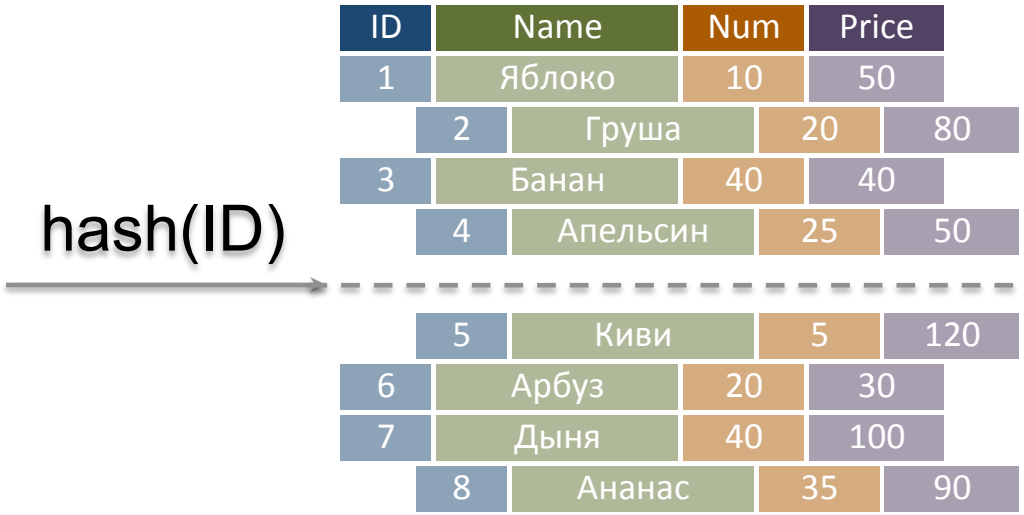
ID	Name	Num	Price
1	Яблоко	10	50
2	Груша	20	80
3	Банан	40	40
4	Апельсин	25	50
5	Киви	5	120
6	Арбуз	20	30
7	Дыня	40	100
8	Ананас	35	90

ID	Name	Num	Price
1	Яблоко	10	50
2	Груша	20	80
3	Банан	40	40
4	Апельсин	25	50
5	Киви	5	120
6	Арбуз	20	30
7	Дыня	40	100
8	Ананас	35	90

Metadata

- Table structure information
  - Distribution fields
  - Number of hash buckets
  - Partitioning (hash, list, range)

ID	Name	Num	Price
1	Яблоко	10	50
2	Груша	20	80
3	Банан	40	40
4	Апельсин	25	50
5	Киви	5	120
6	Арбуз	20	30
7	Дыня	40	100
8	Ананас	35	90



Metadata

- Table structure information
  - Distribution fields
  - Number of hash buckets
  - Partitioning (hash, list, range)
- General metadata
  - Users and groups

Metadata

- Table structure information
  - Distribution fields
  - Number of hash buckets
  - Partitioning (hash, list, range)
- General metadata
  - Users and groups
  - Access privileges

Metadata

- Table structure information
  - Distribution fields
  - Number of hash buckets
  - Partitioning (hash, list, range)
- General metadata
  - Users and groups
  - Access privileges
- Stored procedures
  - PL/pgSQL, PL/Java, PL/Python, PL/Perl, PL/R

Query Optimizer

- HAWQ uses cost-based query optimizers

Query Optimizer

- HAWQ uses cost-based query optimizers
- You have two options
  - Planner – evolved from the Postgres query optimizer
  - ORCA (Pivotal Query Optimizer) – developed specifically for HAWQ

Query Optimizer

- HAWQ uses cost-based query optimizers
- You have two options
  - Planner – evolved from the Postgres query optimizer
  - ORCA (Pivotal Query Optimizer) – developed specifically for HAWQ
- Optimizer hints work just like in Postgres
  - Enable/disable specific operation
  - Change the cost estimations for basic actions

Storage Formats

Which storage format is the most optimal?

Storage Formats

Which storage format is the most optimal?

- ✓ It depends on what you mean by “optimal”

Storage Formats

Which storage format is the most optimal?

- ✓ It depends on what you mean by “optimal”
  - Minimal CPU usage for reading and writing the data

Storage Formats

Which storage format is the most optimal?

- ✓ It depends on what you mean by “optimal”
  - Minimal CPU usage for reading and writing the data
  - Minimal disk space usage

Storage Formats

Which storage format is the most optimal?

- ✓ It depends on what you mean by “optimal”
  - Minimal CPU usage for reading and writing the data
  - Minimal disk space usage
  - Minimal time to retrieve record by key

Storage Formats

Which storage format is the most optimal?

- ✓ It depends on what you mean by “optimal”
  - Minimal CPU usage for reading and writing the data
  - Minimal disk space usage
  - Minimal time to retrieve record by key
  - Minimal time to retrieve subset of columns
  - etc.

Storage Formats

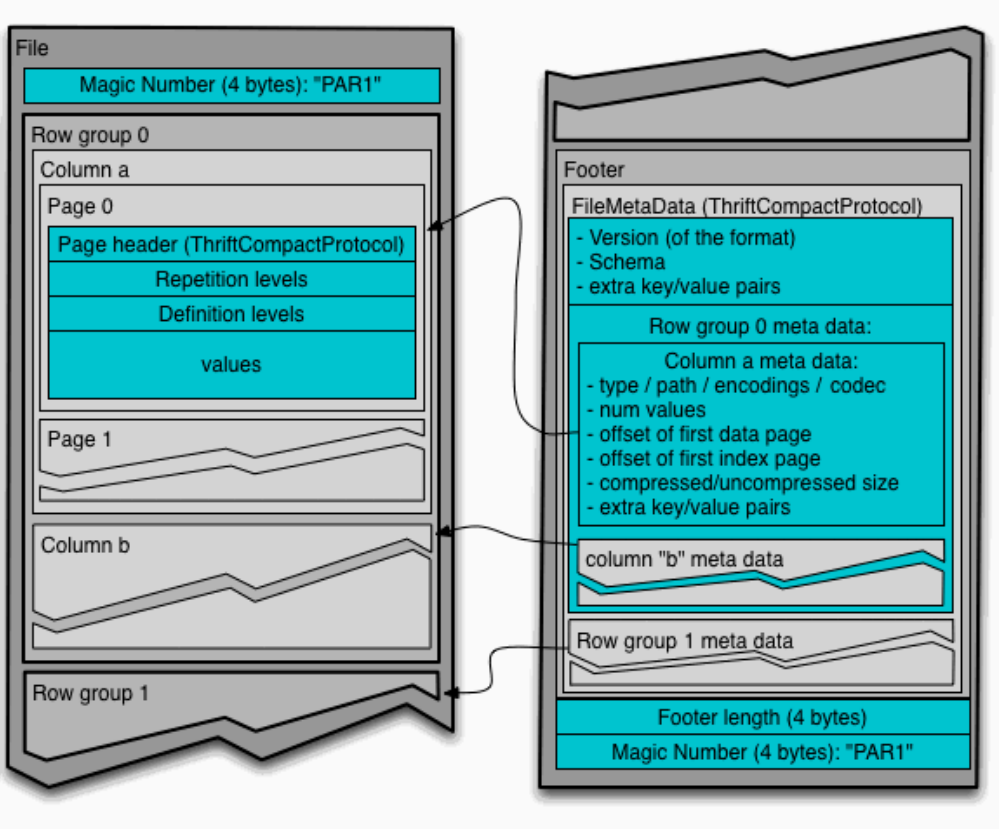
- Row-based storage format
  - Similar to Postgres heap storage
    - No toast
    - No ctid, xmin, xmax, cmin, cmax

Storage Formats

- Row-based storage format
  - Similar to Postgres heap storage
    - No toast
    - No ctid, xmin, xmax, cmin, cmax
  - Compression
    - No compression
    - Quicklz
    - Zlib levels 1 - 9

Storage Formats

- Apache Parquet
  - Mixed row-columnar table store, the data is split into “row groups” stored in columnar format



Storage Formats

- Apache Parquet
  - Mixed row-columnar table store, the data is split into “row groups” stored in columnar format
  - Compression
    - No compression
    - Snappy
    - Gzip levels 1 – 9

Storage Formats

- Apache Parquet
  - Mixed row-columnar table store, the data is split into “row groups” stored in columnar format
  - Compression
    - No compression
    - Snappy
    - Gzip levels 1 – 9
  - The size of “row group” and page size can be set for each table separately

Resource Management

- Two main options
  - Static resource split – HAWQ and YARN does not know about each other

Resource Management

- Two main options
  - Static resource split – HAWQ and YARN does not know about each other
  - YARN – HAWQ asks YARN Resource Manager for query execution resources

Resource Management

- Two main options
  - Static resource split – HAWQ and YARN does not know about each other
  - YARN – HAWQ asks YARN Resource Manager for query execution resources
- Flexible cluster utilization
  - Query might run on a subset of nodes if it is small

Resource Management

- Two main options
  - Static resource split – HAWQ and YARN does not know about each other
  - YARN – HAWQ asks YARN Resource Manager for query execution resources
- Flexible cluster utilization
  - Query might run on a subset of nodes if it is small
  - Query might have many executors on each cluster node to make it run faster

Resource Management

- Two main options
  - Static resource split – HAWQ and YARN does not know about each other
  - YARN – HAWQ asks YARN Resource Manager for query execution resources
- Flexible cluster utilization
  - Query might run on a subset of nodes if it is small
  - Query might have many executors on each cluster node to make it run faster
  - You can control the parallelism of each query

Resource Management

- Resource Queue can be set with
  - Maximum number of parallel queries

Resource Management

- Resource Queue can be set with
  - Maximum number of parallel queries
  - CPU usage priority

Resource Management

- Resource Queue can be set with
  - Maximum number of parallel queries
  - CPU usage priority
  - Memory usage limits

Resource Management

- Resource Queue can be set with
  - Maximum number of parallel queries
  - CPU usage priority
  - Memory usage limits
  - CPU cores usage limit

Resource Management

- Resource Queue can be set with
  - Maximum number of parallel queries
  - CPU usage priority
  - Memory usage limits
  - CPU cores usage limit
  - MIN/MAX number of executors across the system

Resource Management

- Resource Queue can be set with
  - Maximum number of parallel queries
  - CPU usage priority
  - Memory usage limits
  - CPU cores usage limit
  - MIN/MAX number of executors across the system
  - MIN/MAX number of executors on each node

Resource Management

- Resource Queue can be set with
  - Maximum number of parallel queries
  - CPU usage priority
  - Memory usage limits
  - CPU cores usage limit
  - MIN/MAX number of executors across the system
  - MIN/MAX number of executors on each node
- Can be set up for user or group

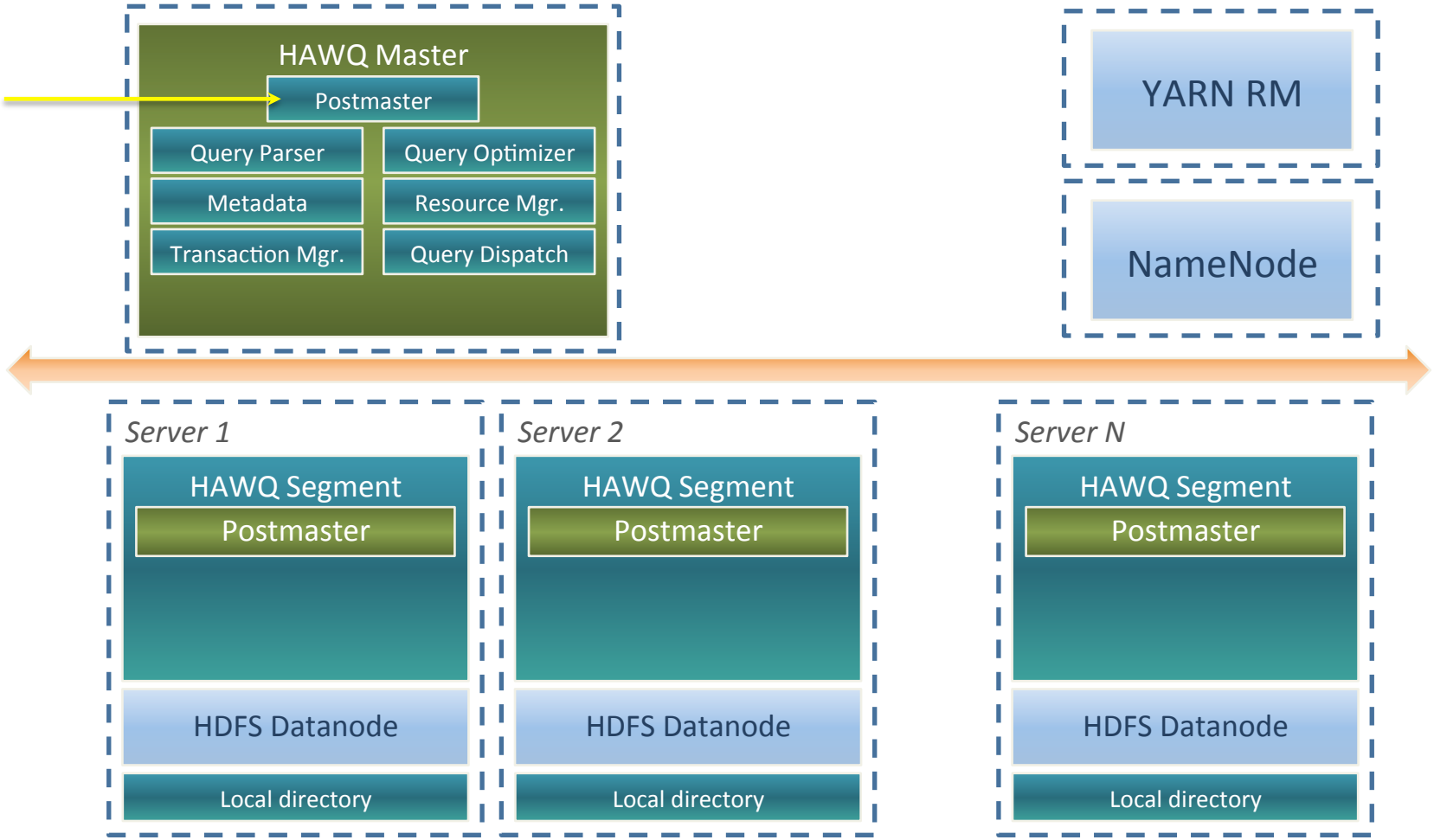
External Data

- PXF
  - Framework for external data access
  - Easy to extend, many public plugins available
  - Official plugins: CSV, SequenceFile, Avro, Hive, HBase
  - Open Source plugins: JSON, Accumulo, Cassandra, JDBC, Redis, Pipe

External Data

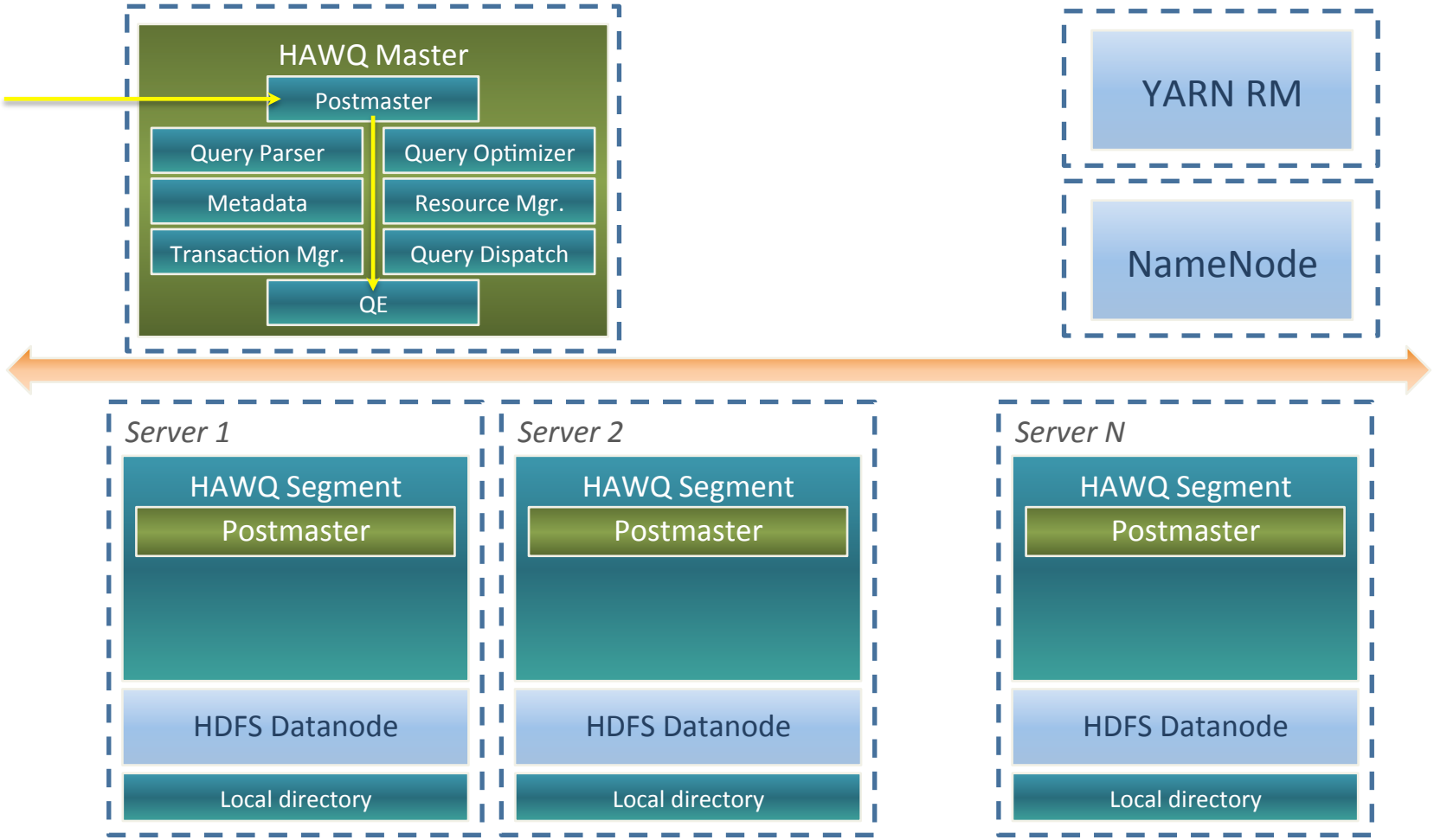
- PXF
  - Framework for external data access
  - Easy to extend, many public plugins available
  - Official plugins: CSV, SequenceFile, Avro, Hive, HBase
  - Open Source plugins: JSON, Accumulo, Cassandra, JDBC, Redis, Pipe
- HCatalog
  - HAWQ can query tables from HCatalog the same way as HAWQ native tables

Query Example



Plan | Resource | Prepare | Execute | Result | Cleanup

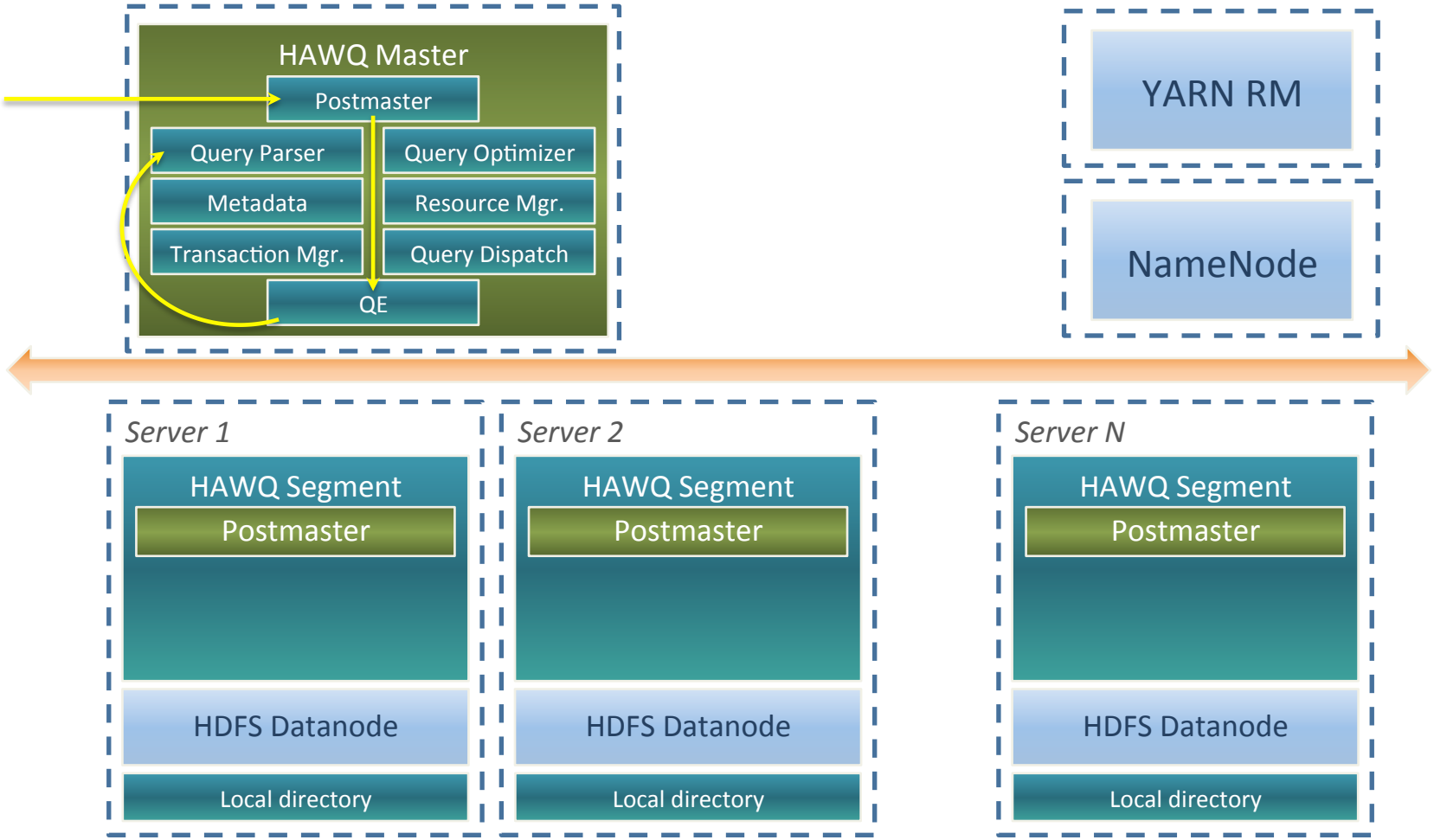
Query Example



Plan | Resource | Prepare | Execute | Result | Cleanup

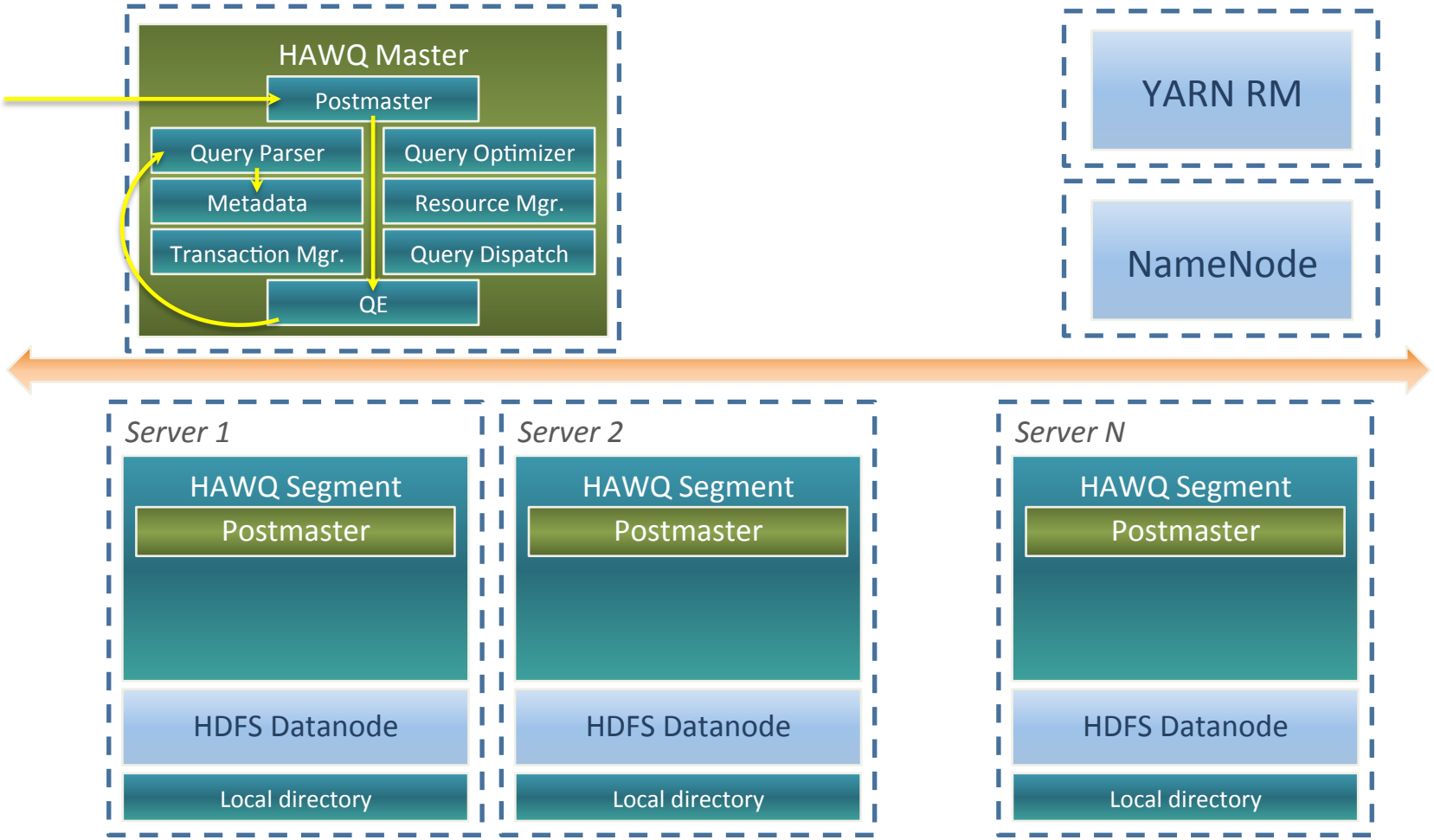
HighLoad++

Query Example



Plan | Resource | Prepare | Execute | Result | Cleanup

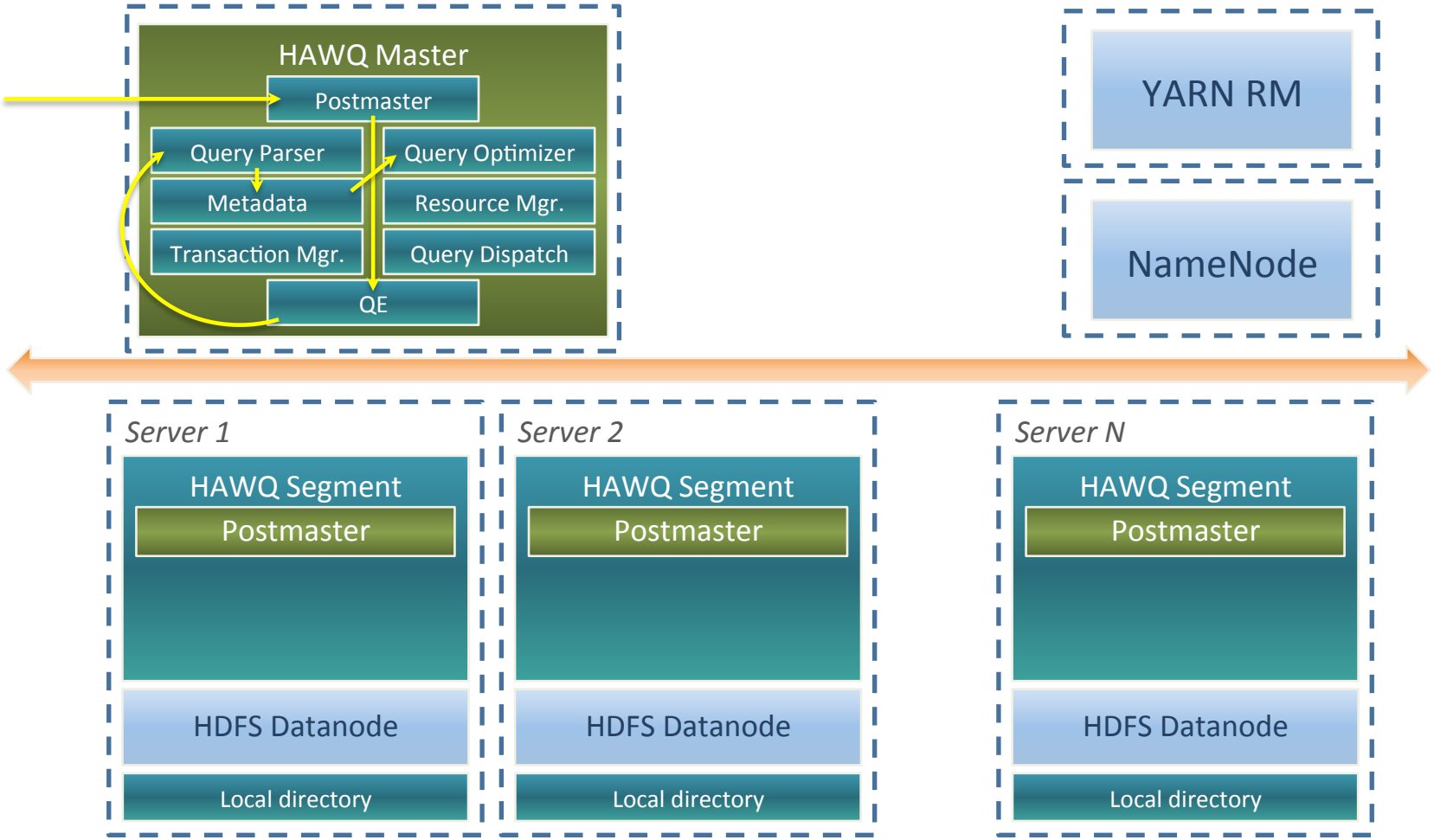
Query Example



Plan | Resource | Prepare | Execute | Result | Cleanup

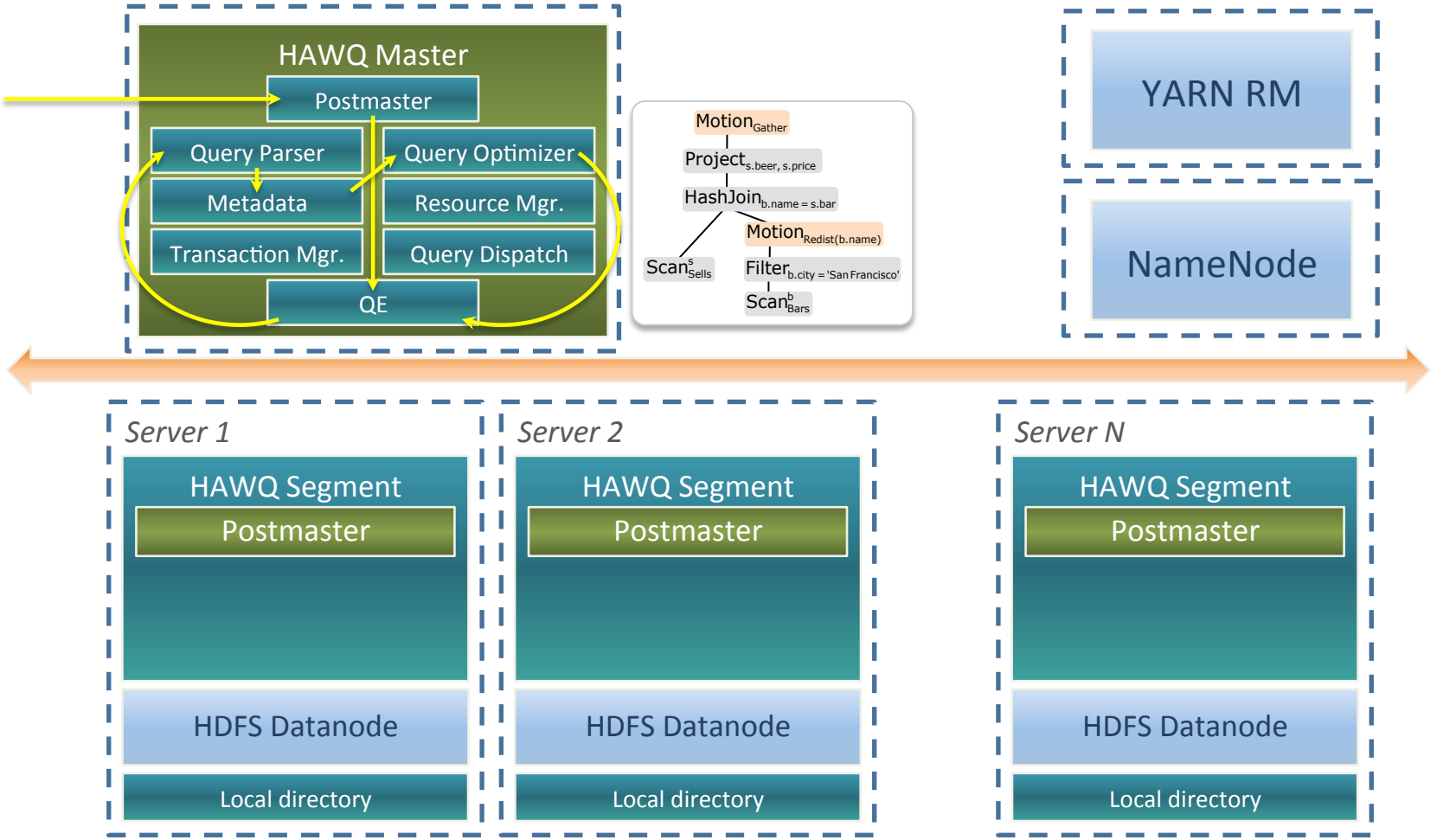
HighLoad++

Query Example



Plan | Resource | Prepare | Execute | Result | Cleanup

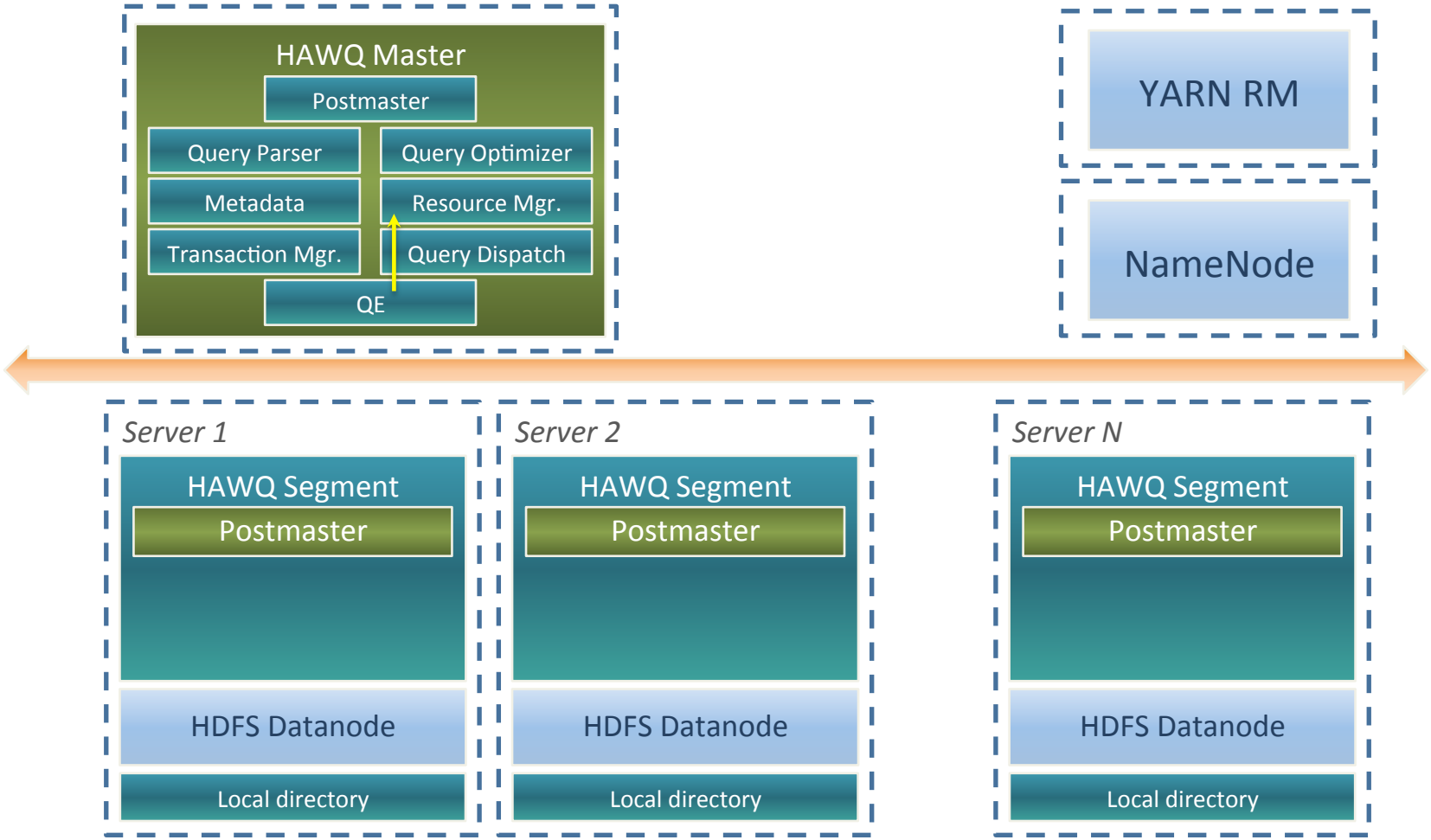
Query Example



Plan | Resource | Prepare | Execute | Result | Cleanup

HighLoad++

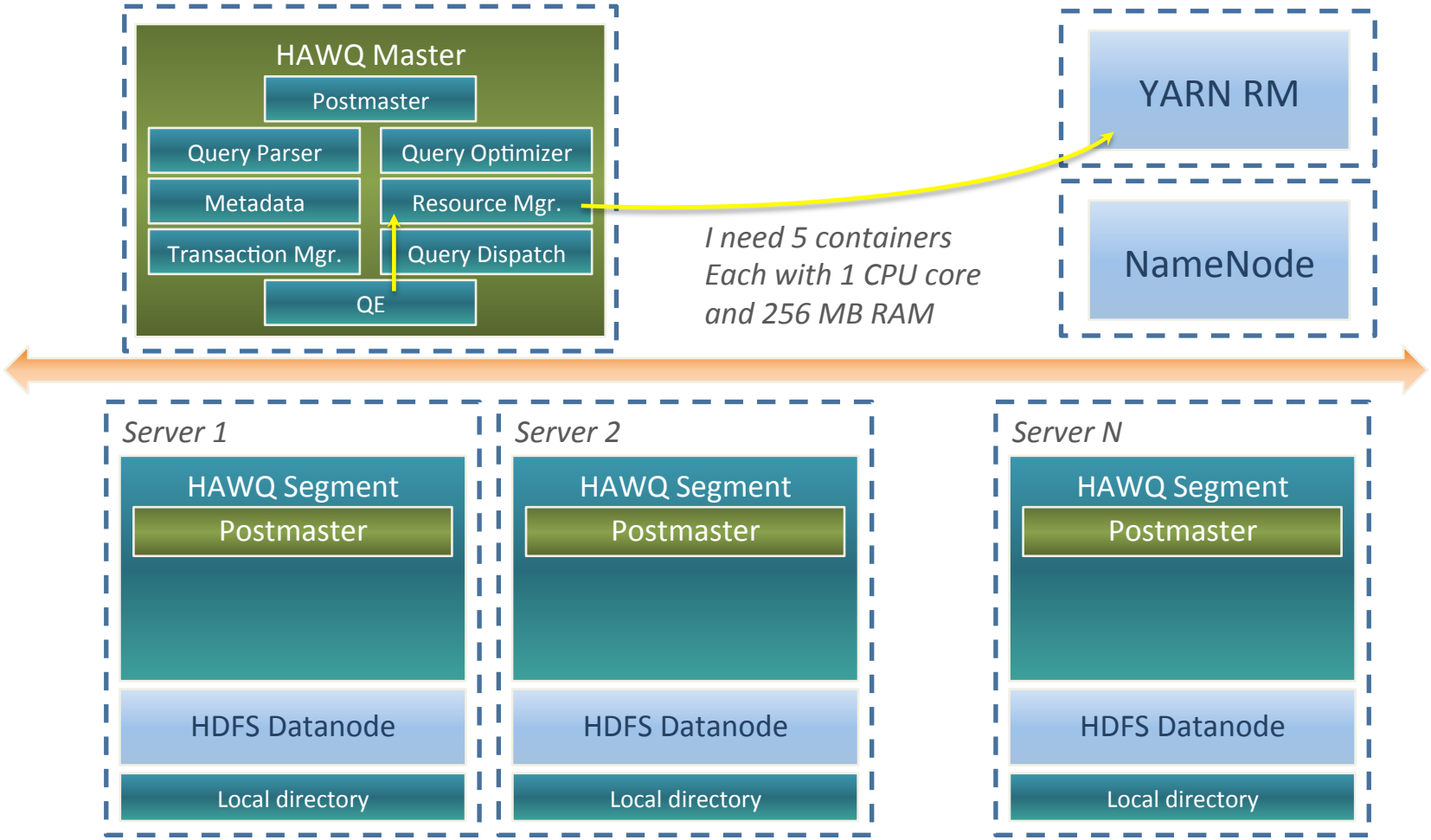
Query Example



Plan | Resource | Prepare | Execute | Result | Cleanup

HighLoad++

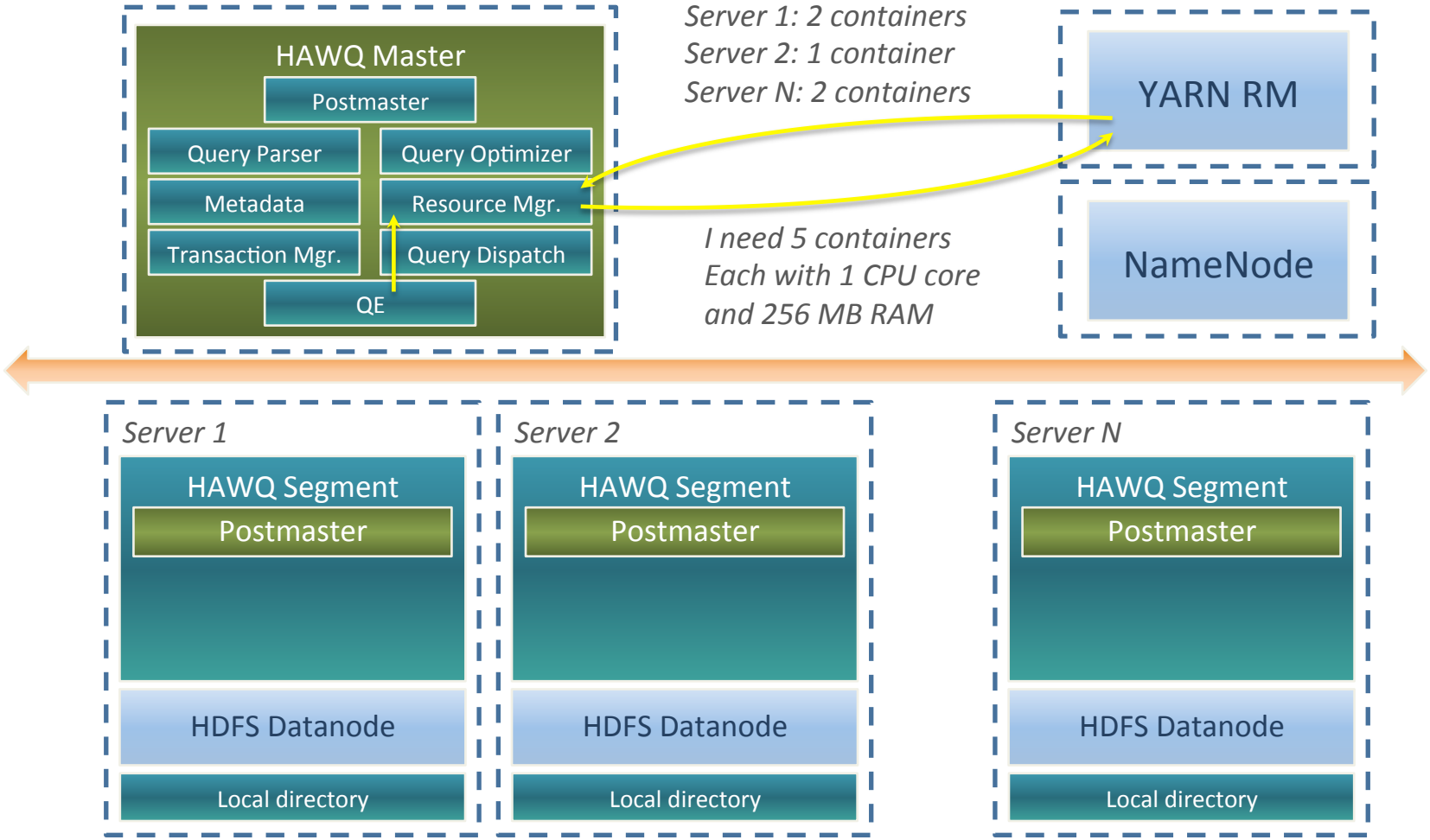
Query Example



Plan | Resource | Prepare | Execute | Result | Cleanup

HighLoad++

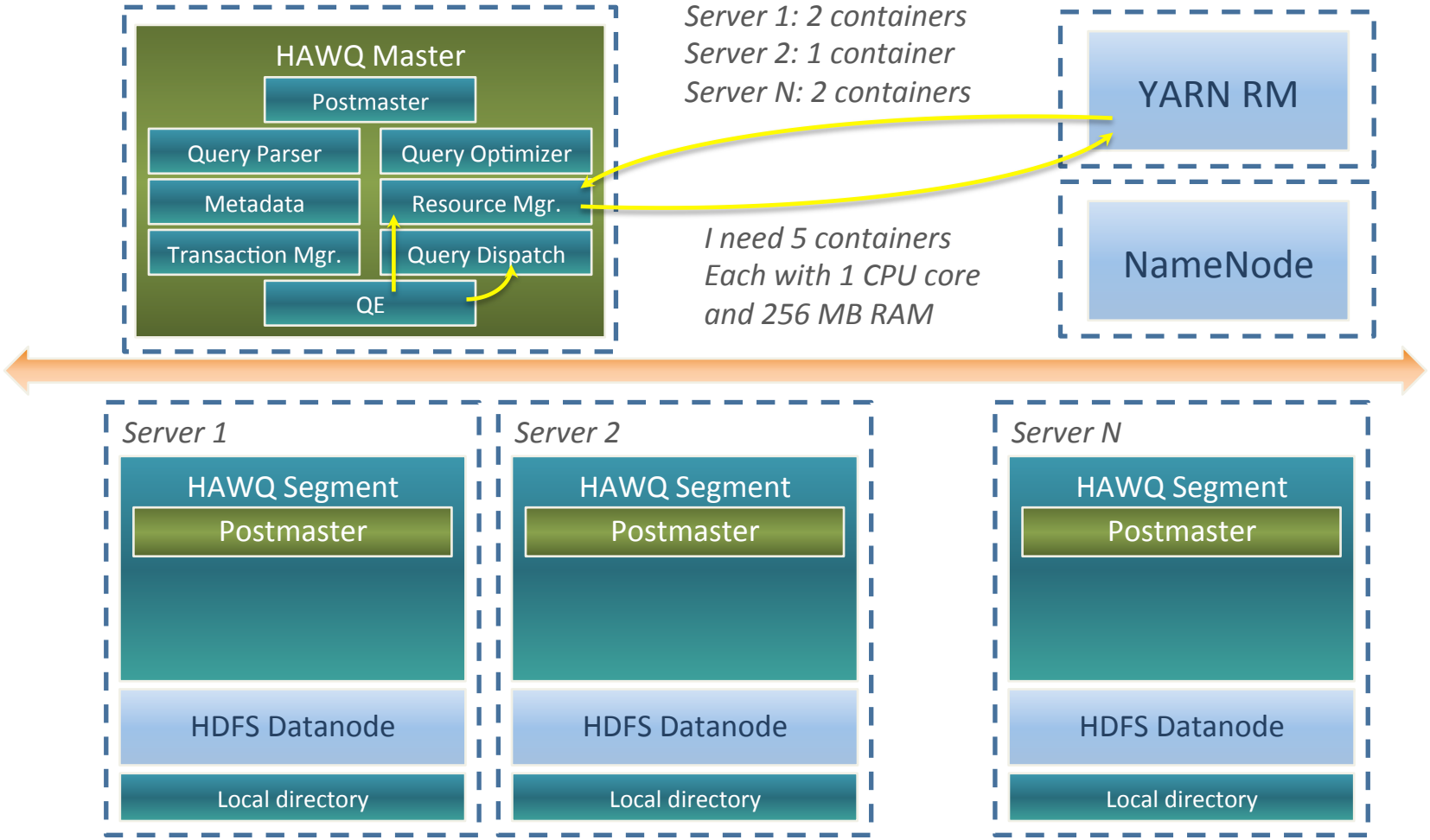
Query Example



Plan | Resource | Prepare | Execute | Result | Cleanup

HighLoad++

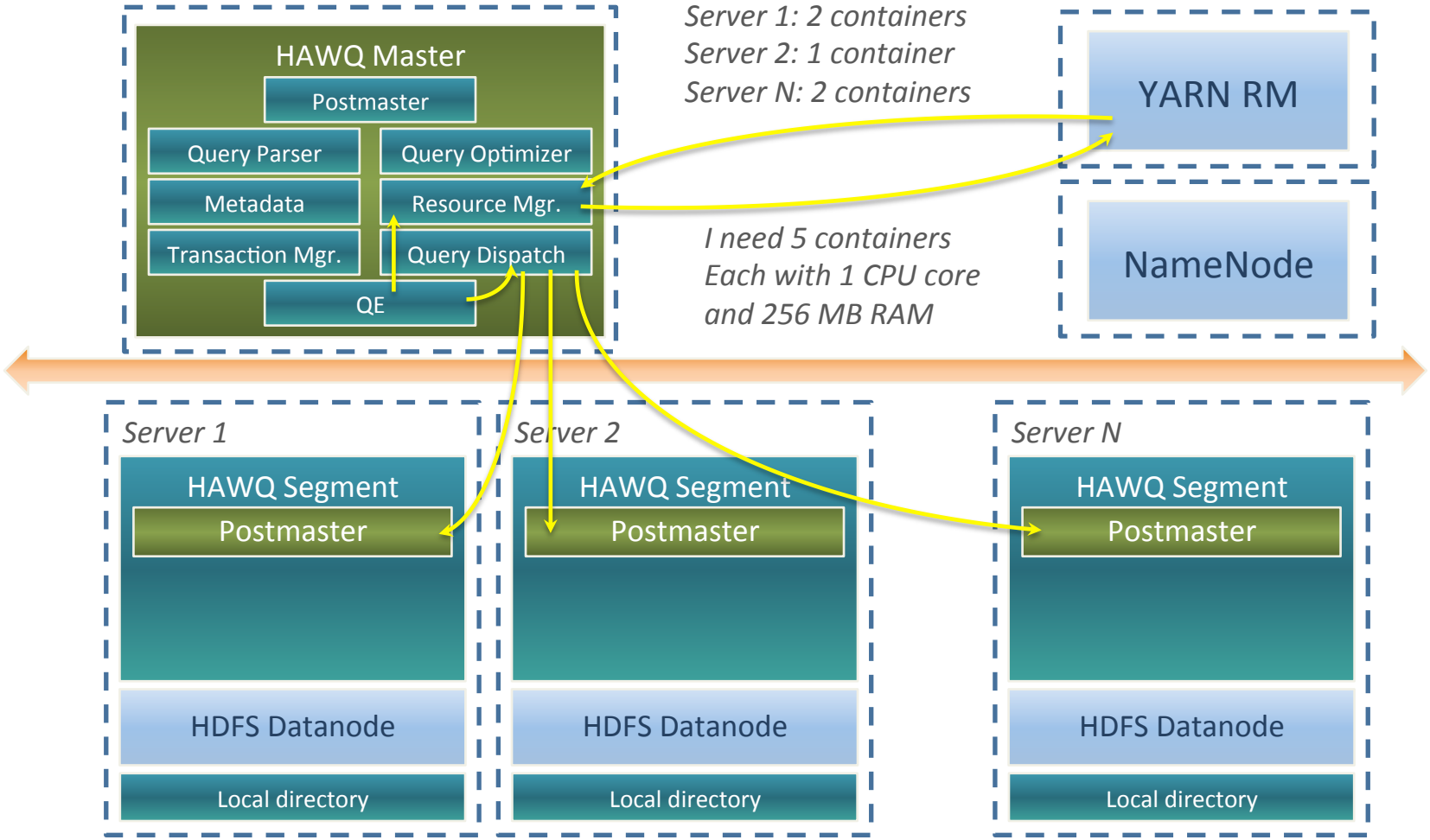
Query Example



Plan | Resource | Prepare | Execute | Result | Cleanup

HighLoad++

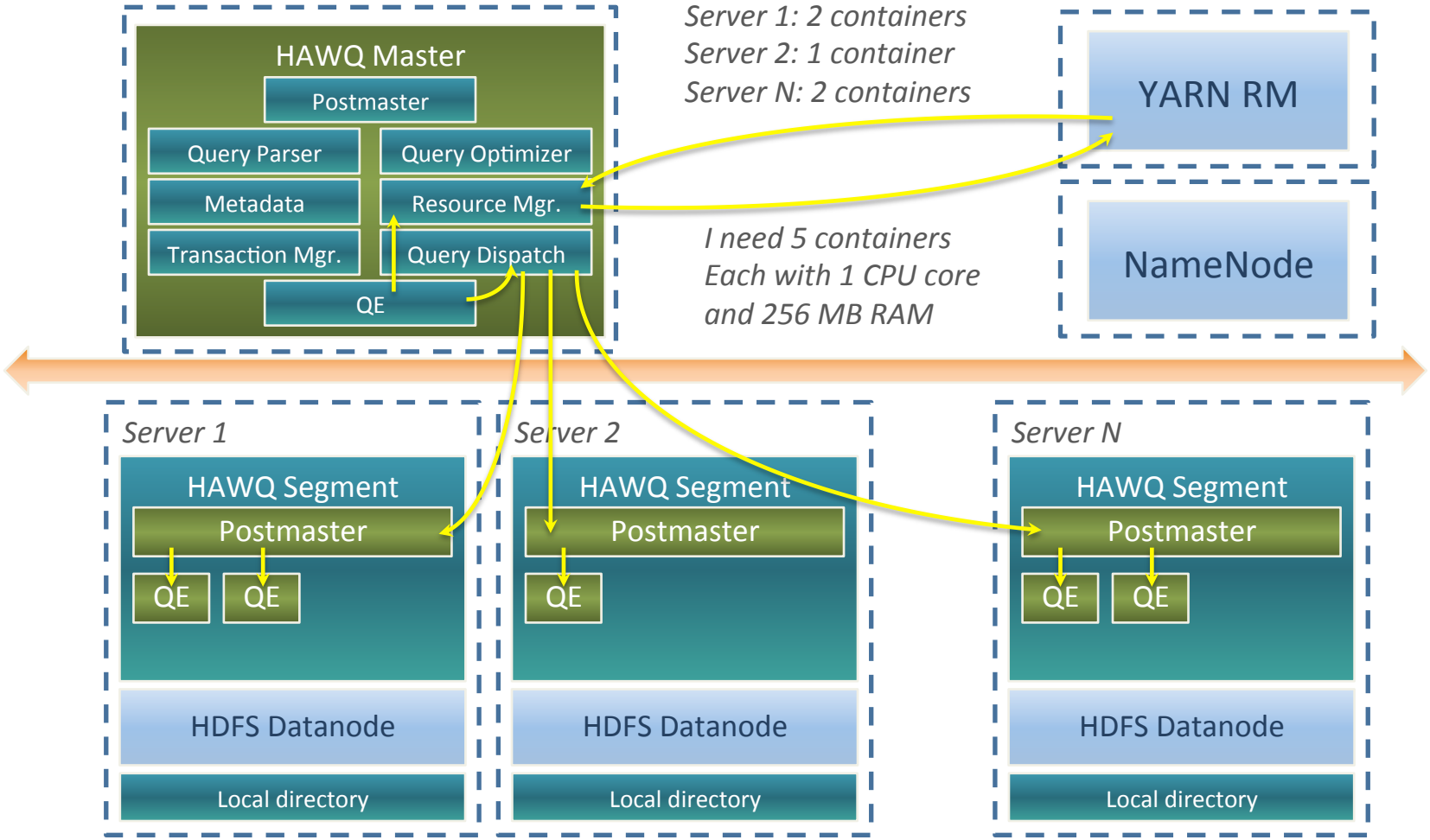
Query Example



Plan | Resource | Prepare | Execute | Result | Cleanup

HighLoad++

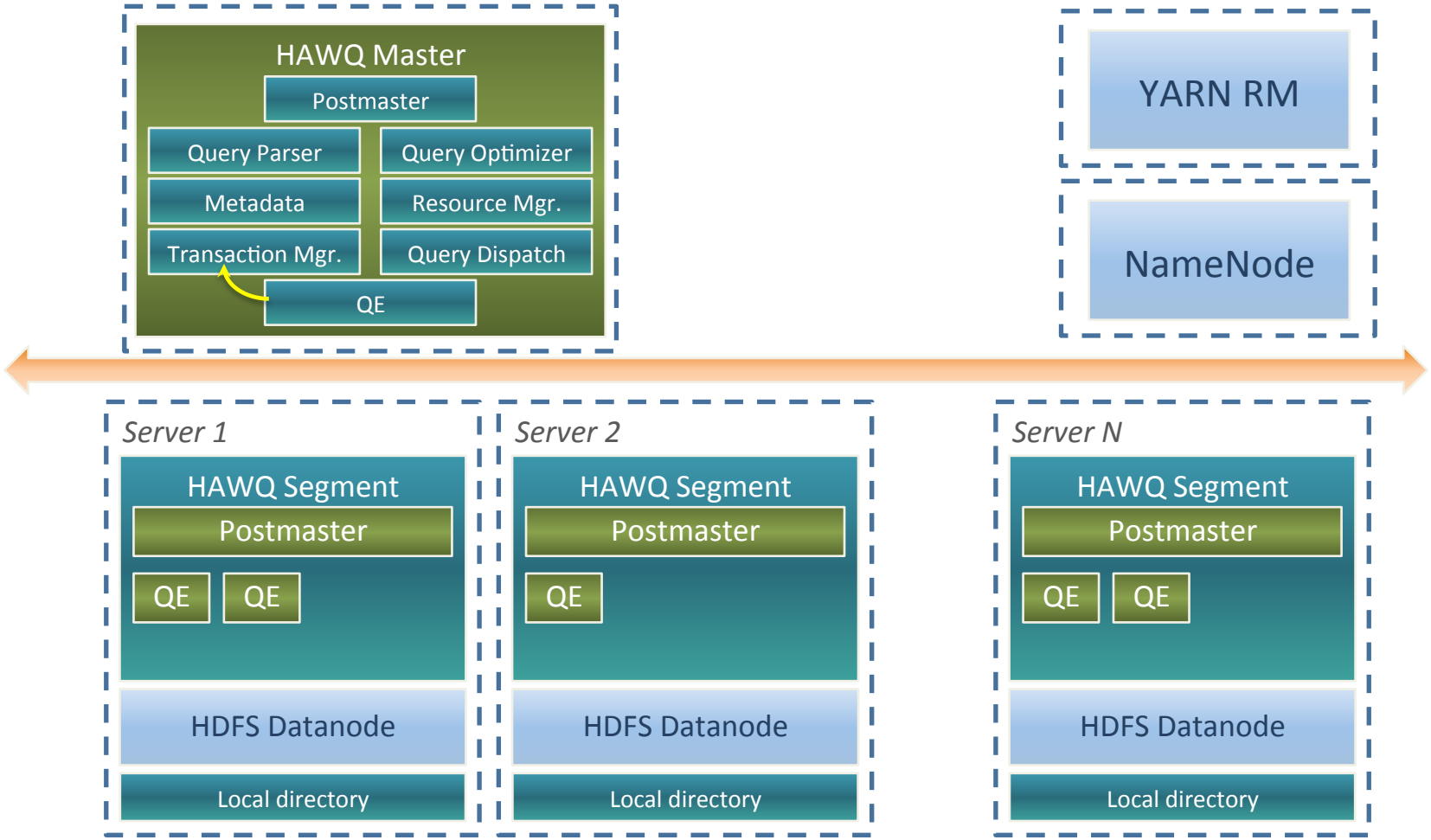
Query Example



Plan | Resource | Prepare | Execute | Result | Cleanup

HighLoad++

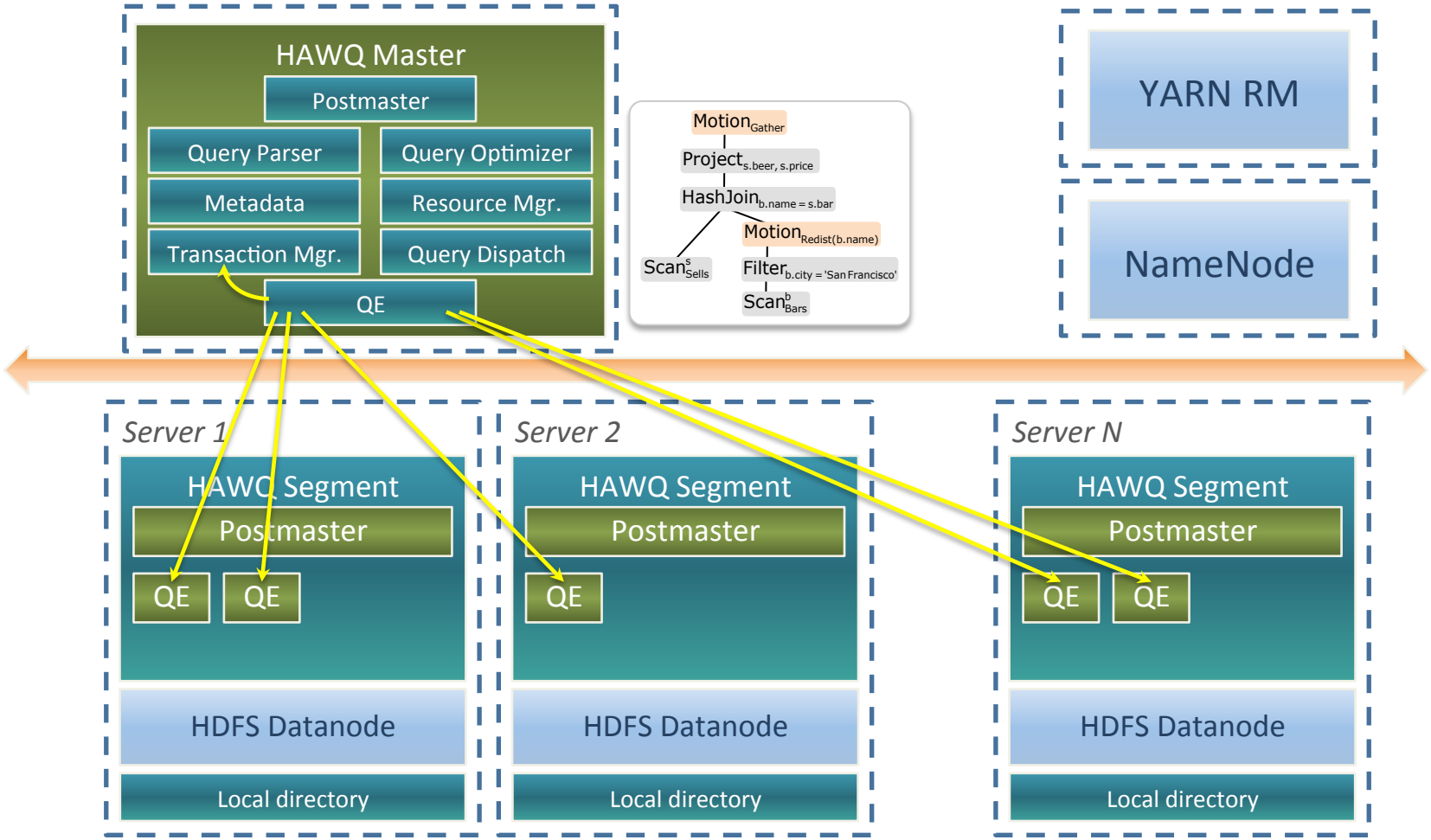
Query Example



Plan | Resource | Prepare | Execute | Result | Cleanup

HighLoad++

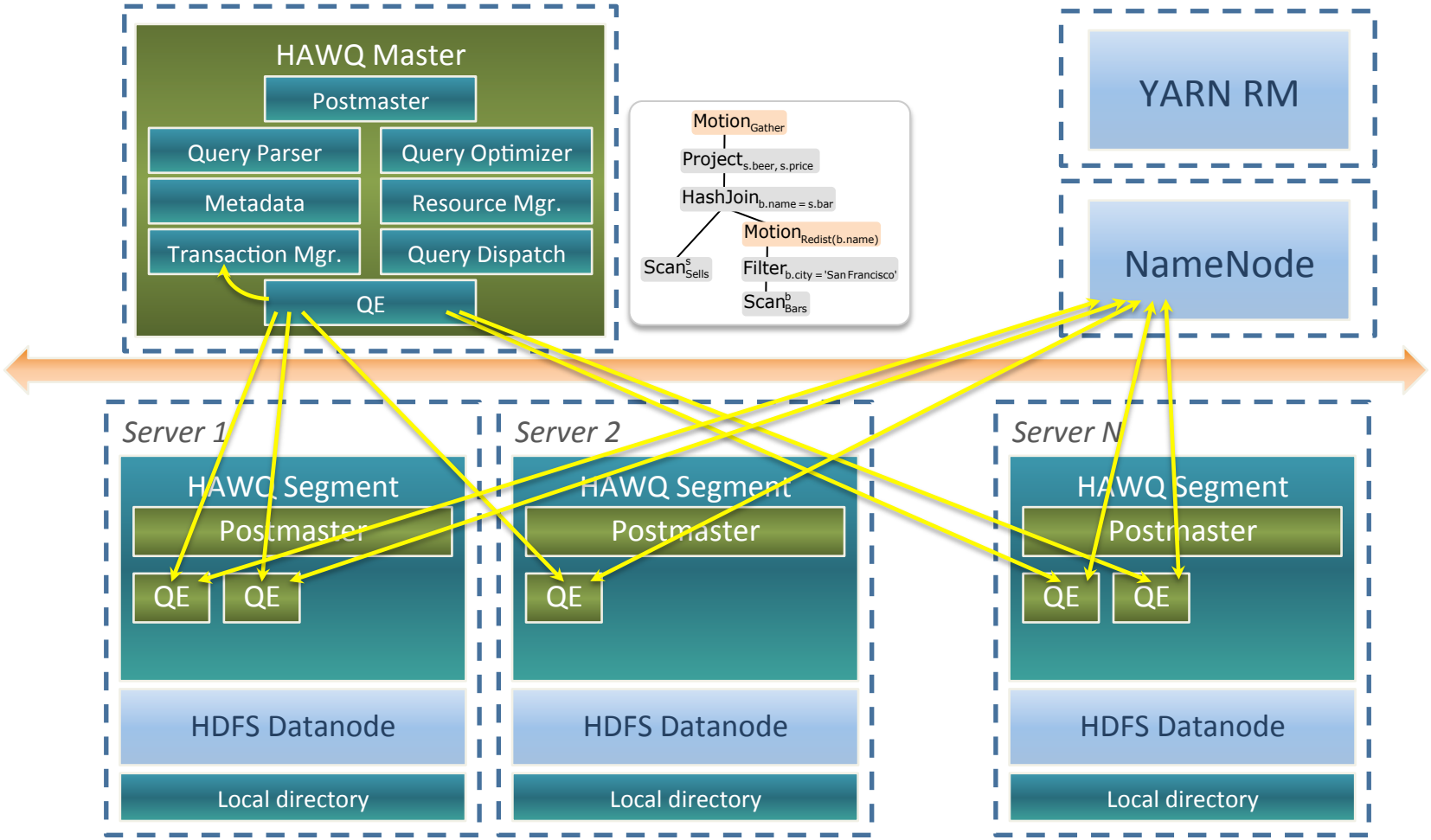
Query Example



Plan | Resource | Prepare | Execute | Result | Cleanup

HighLoad++

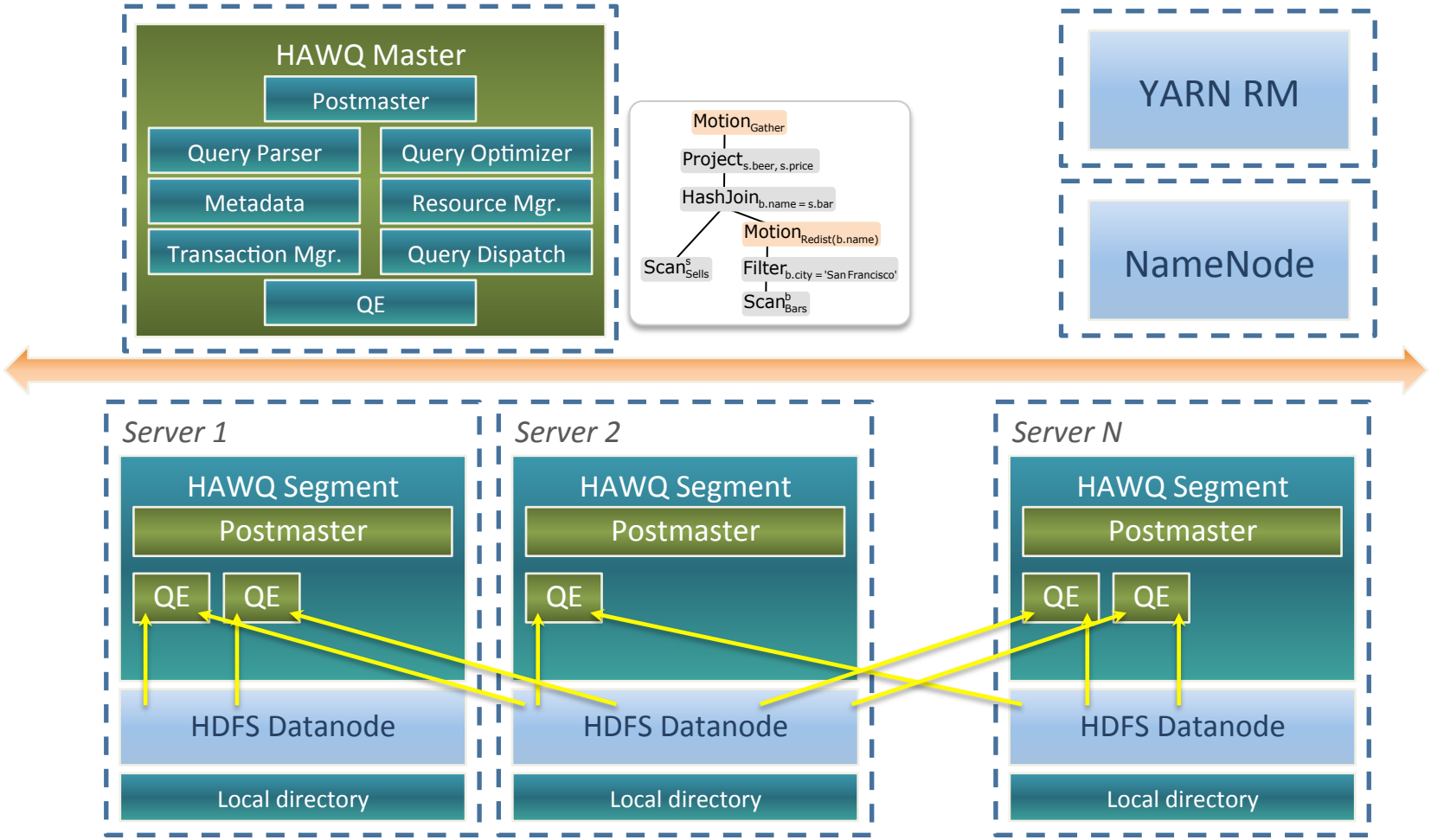
Query Example



Plan | Resource | Prepare | Execute | Result | Cleanup

HighLoad++

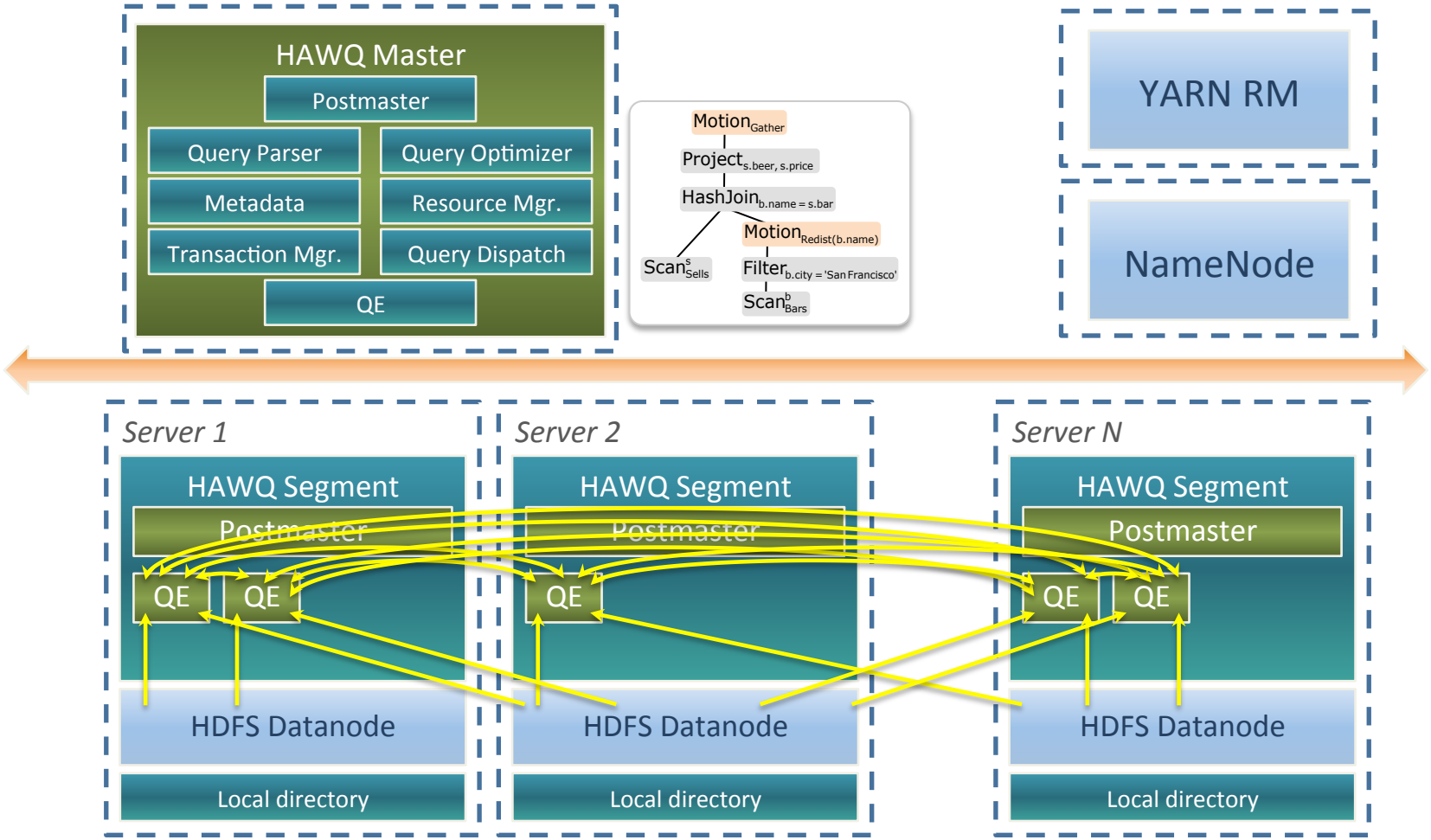
Query Example



Plan | Resource | Prepare | Execute | Result | Cleanup

HighLoad++

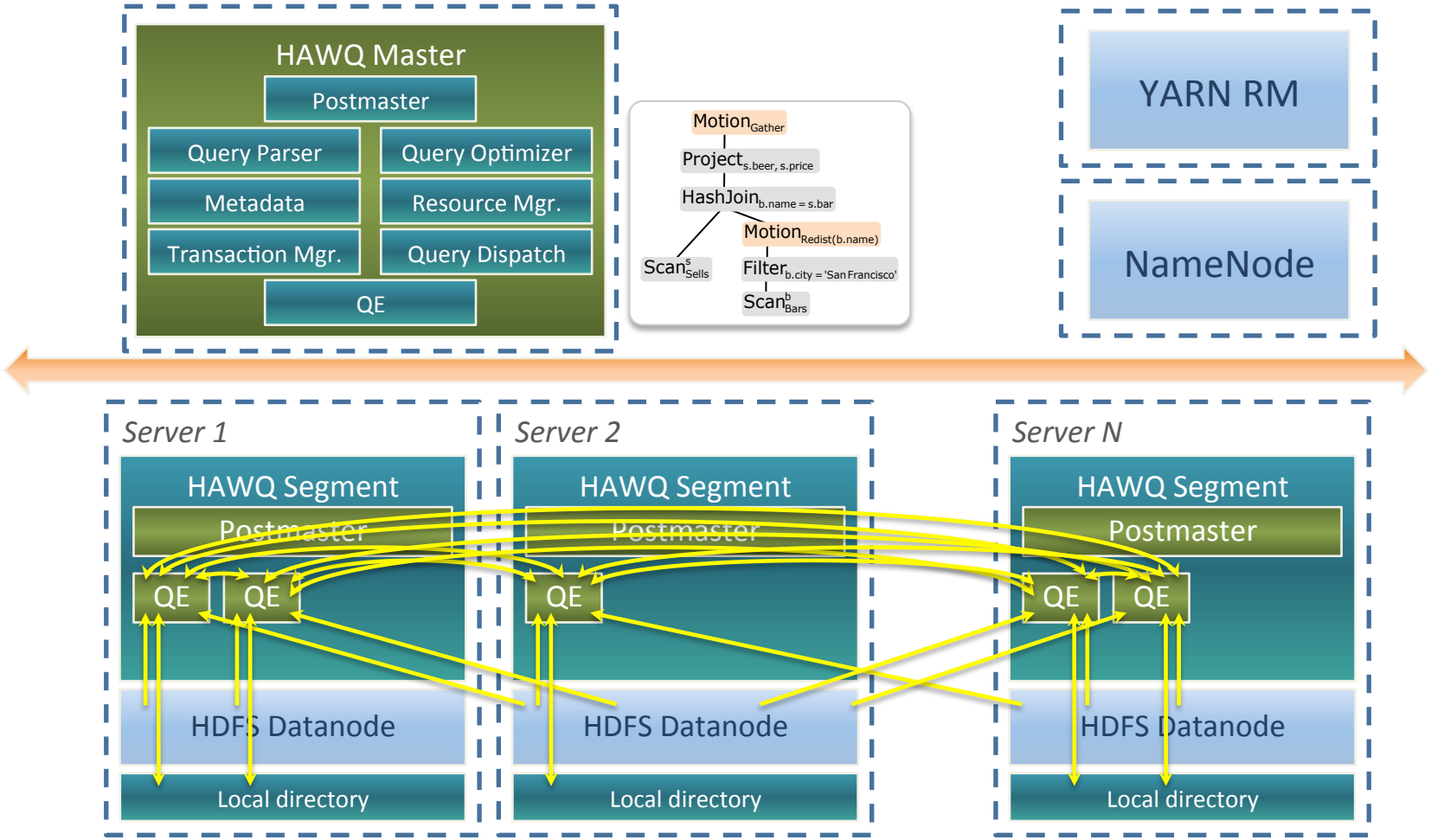
Query Example



Plan | Resource | Prepare | Execute | Result | Cleanup

HighLoad++

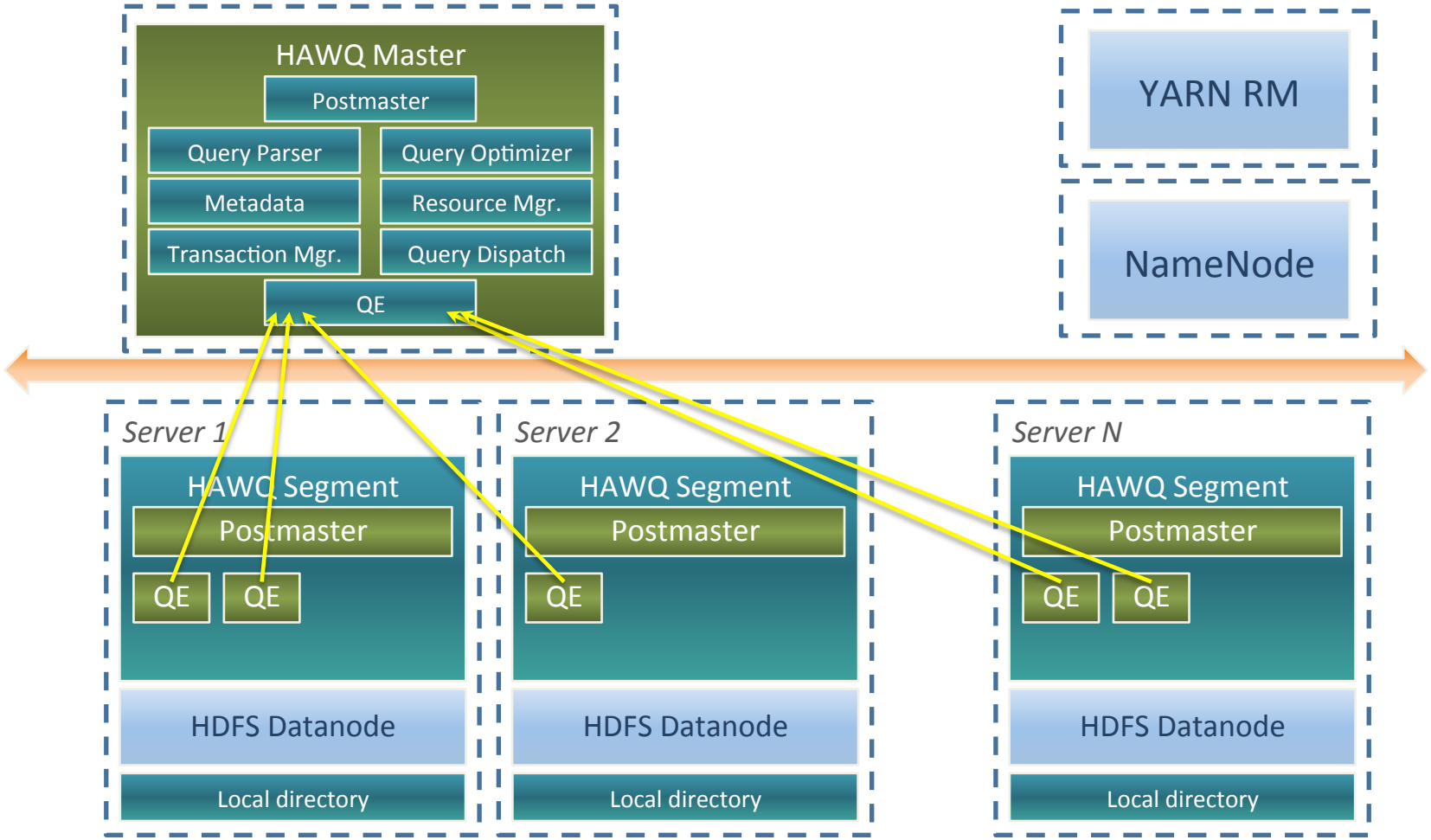
Query Example



Plan | Resource | Prepare | Execute | Result | Cleanup

HighLoad++

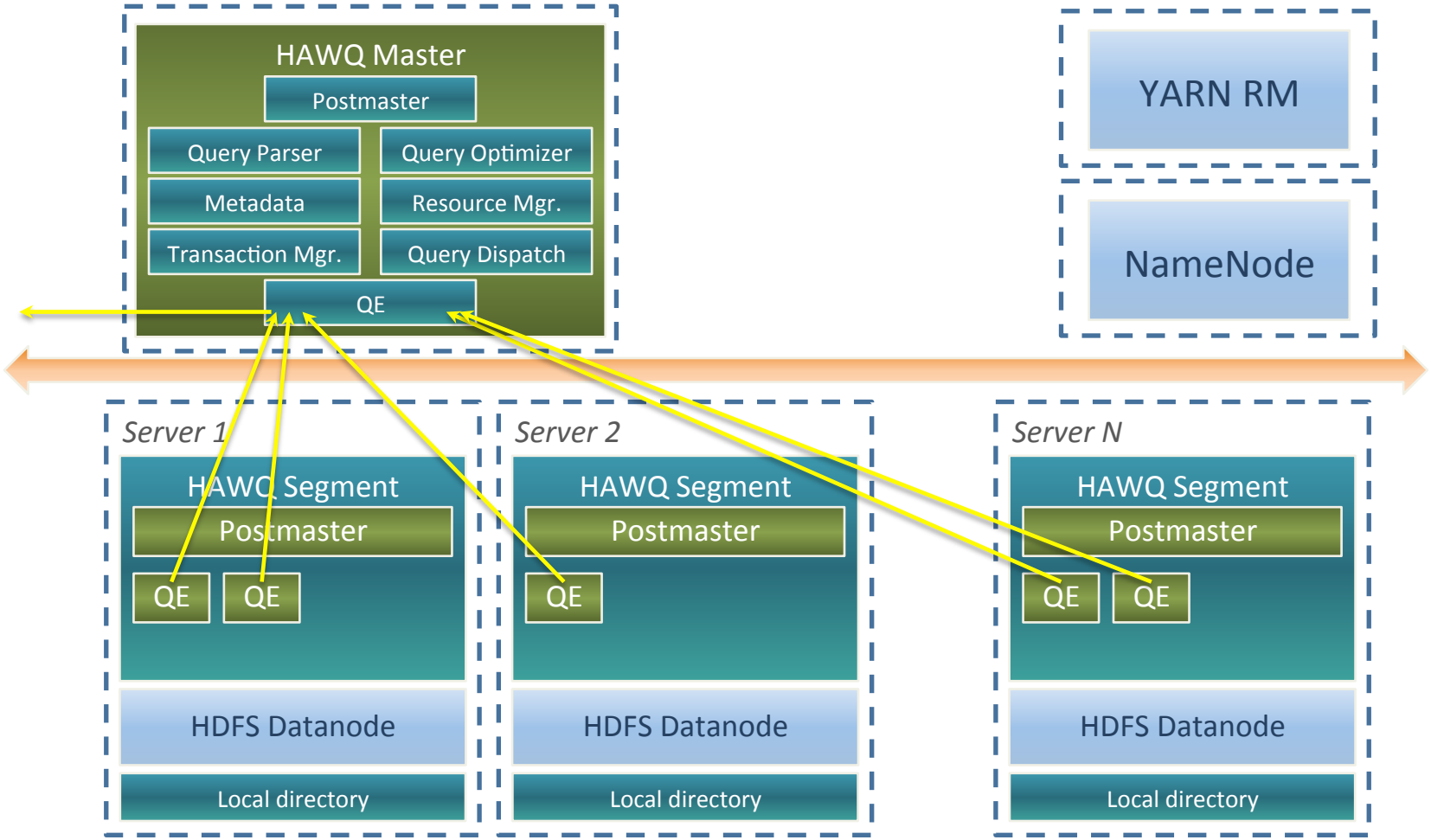
Query Example



Plan | Resource | Prepare | Execute | Result | Cleanup

HighLoad++

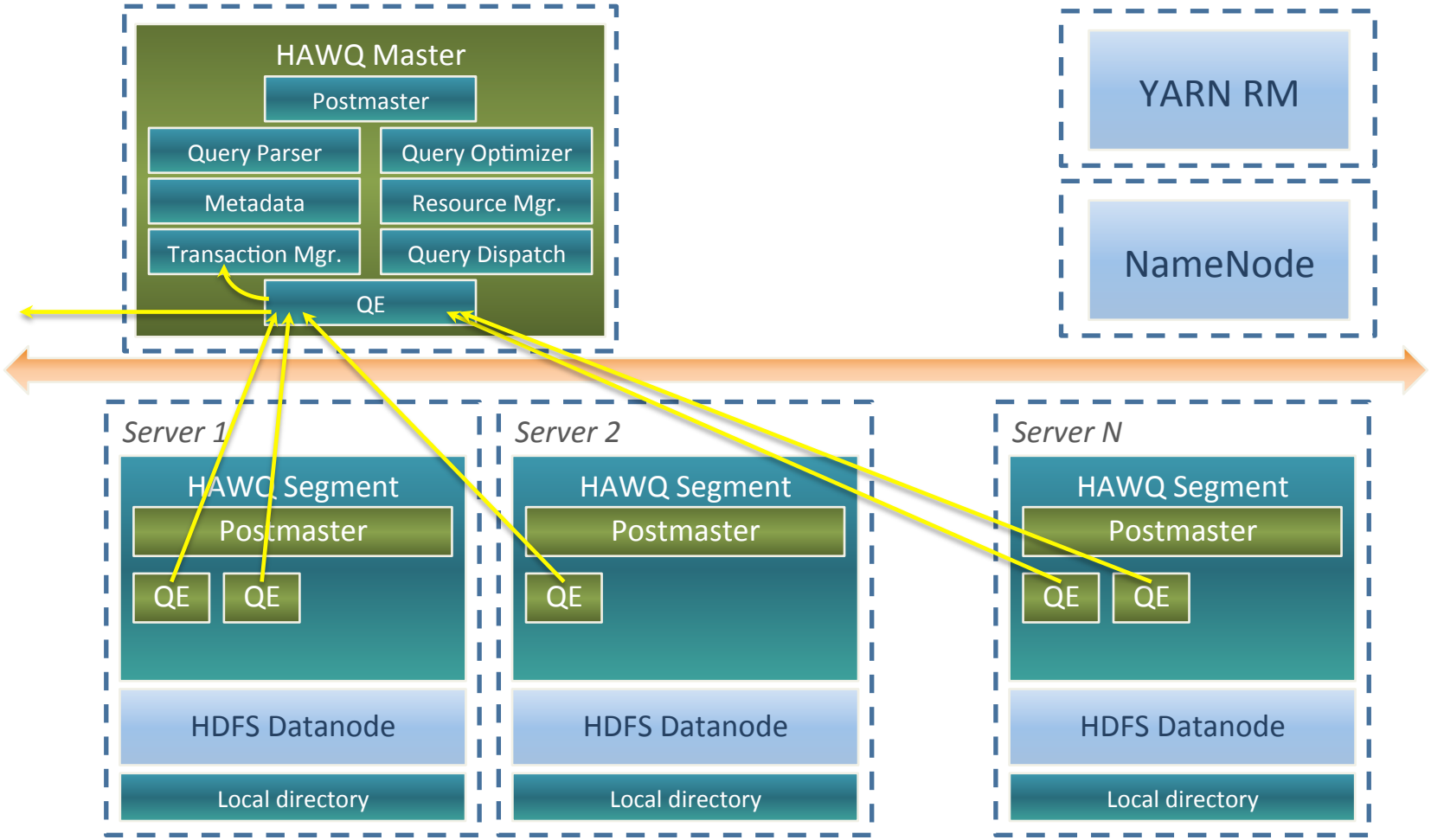
Query Example



Plan | Resource | Prepare | Execute | Result | Cleanup

HighLoad++

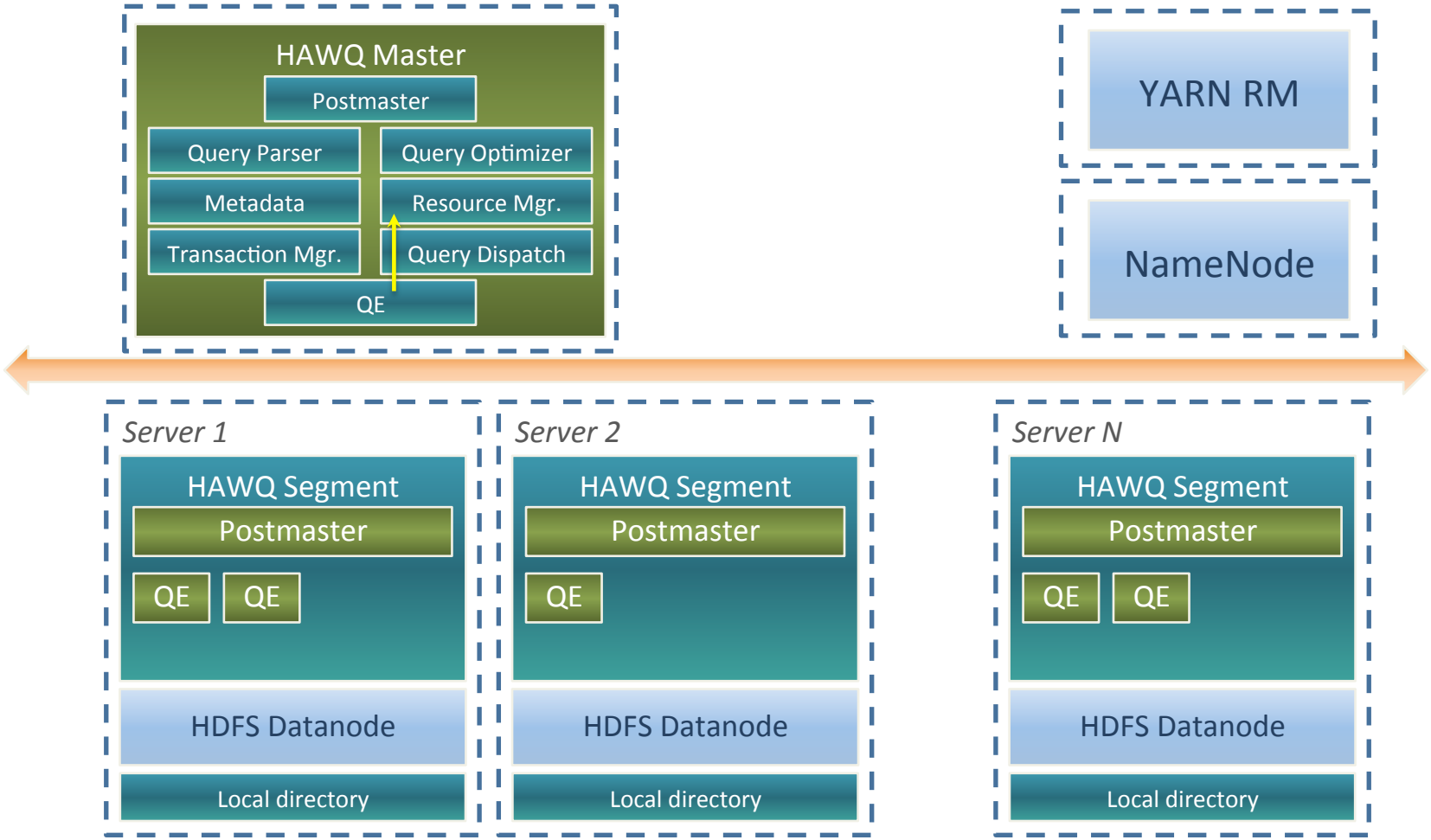
Query Example



Plan | Resource | Prepare | Execute | Result | Cleanup

HighLoad++

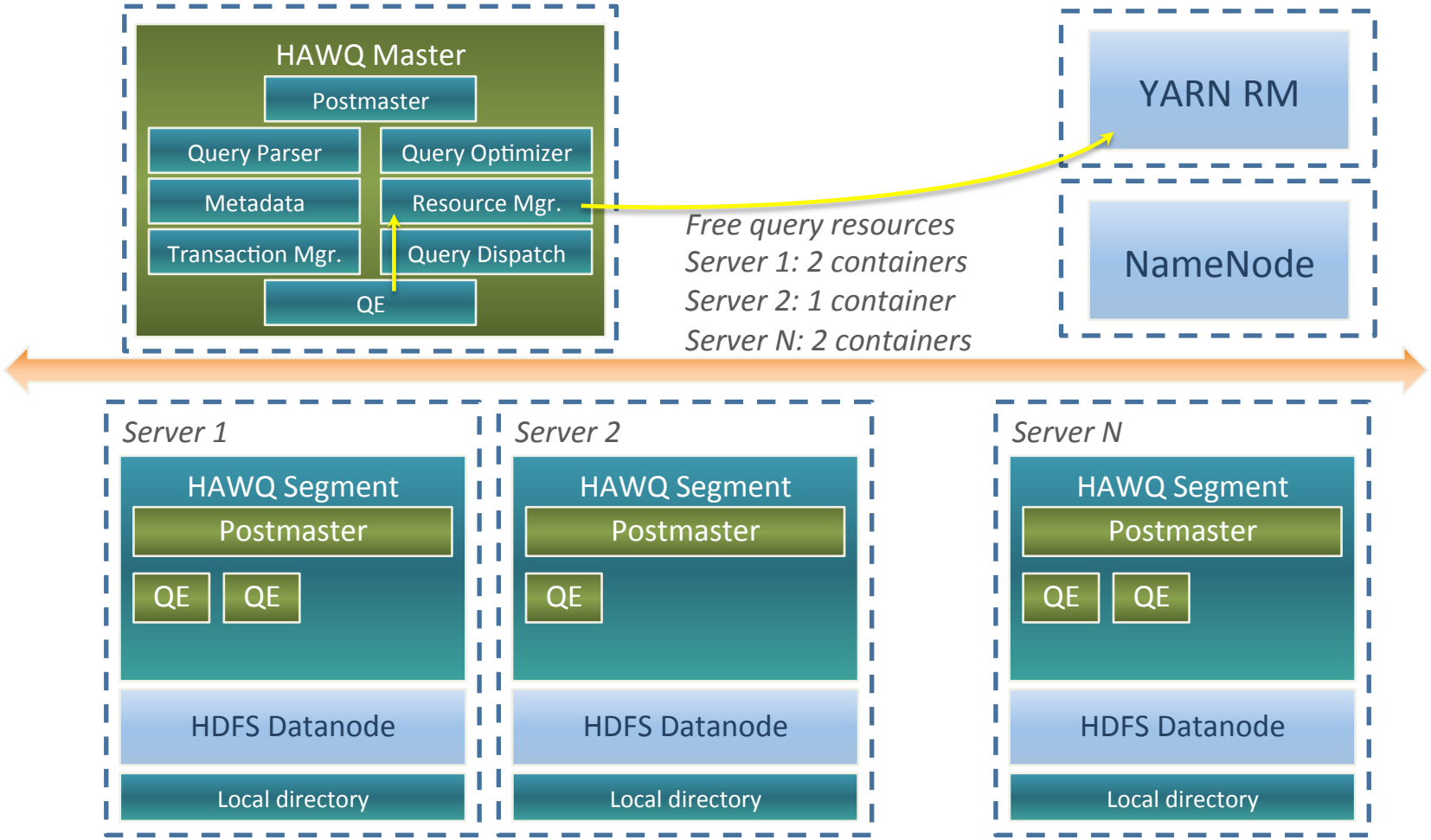
Query Example



Plan → Resource → Prepare → Execute → Result → Cleanup

HighLoad++

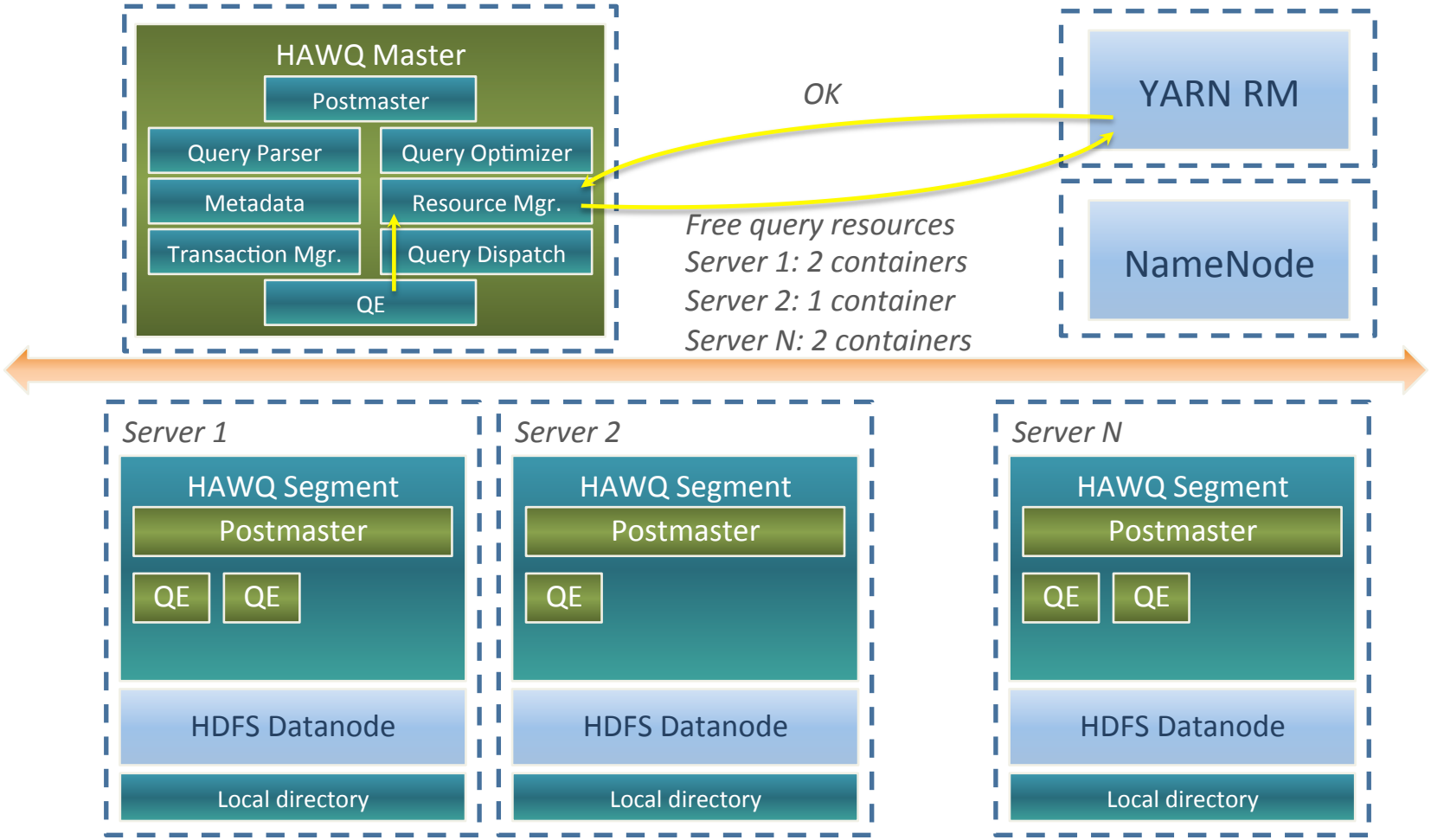
Query Example



Plan → Resource → Prepare → Execute → Result → Cleanup

HighLoad++

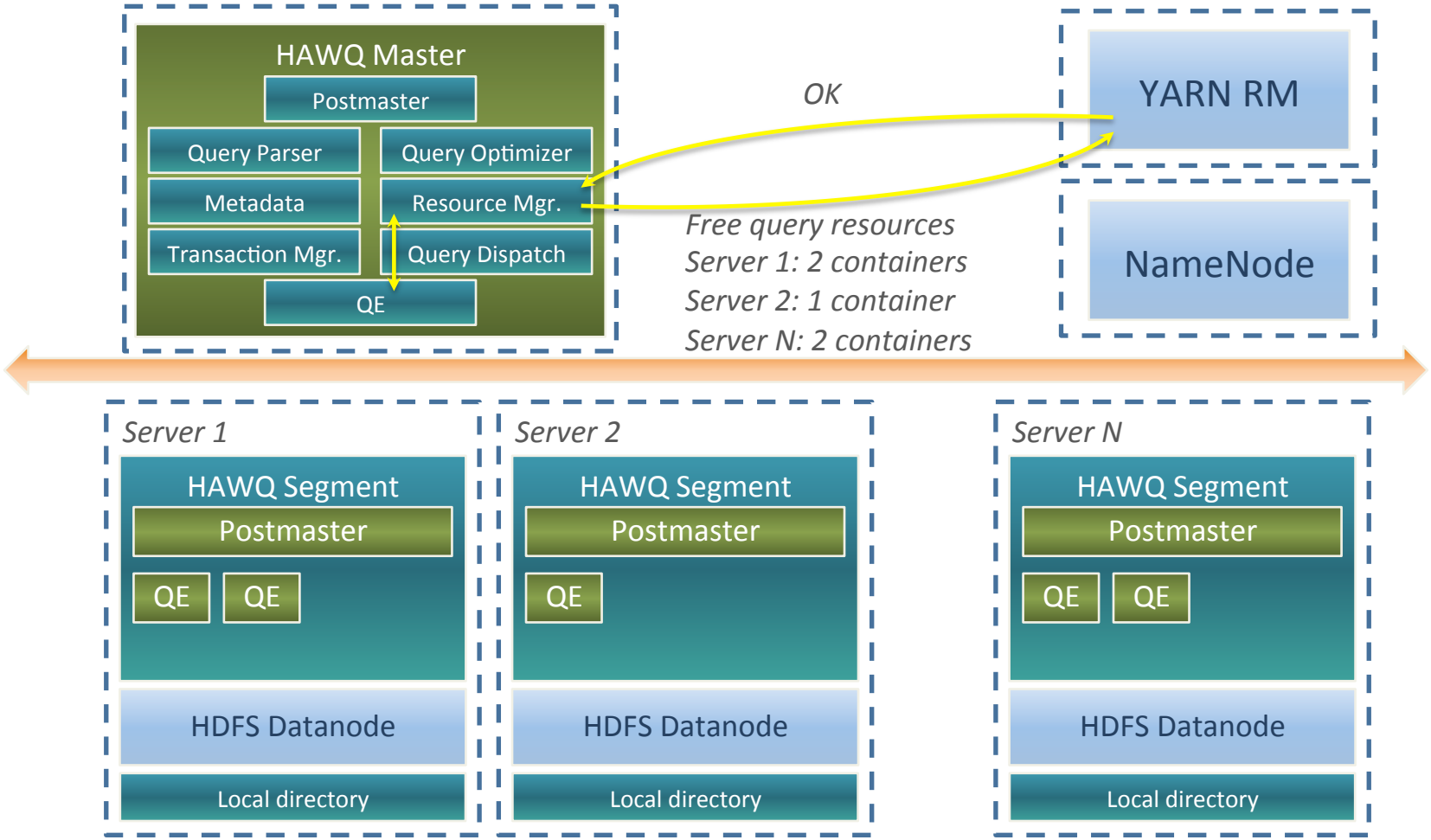
Query Example



Plan | Resource | Prepare | Execute | Result | Cleanup

HighLoad++

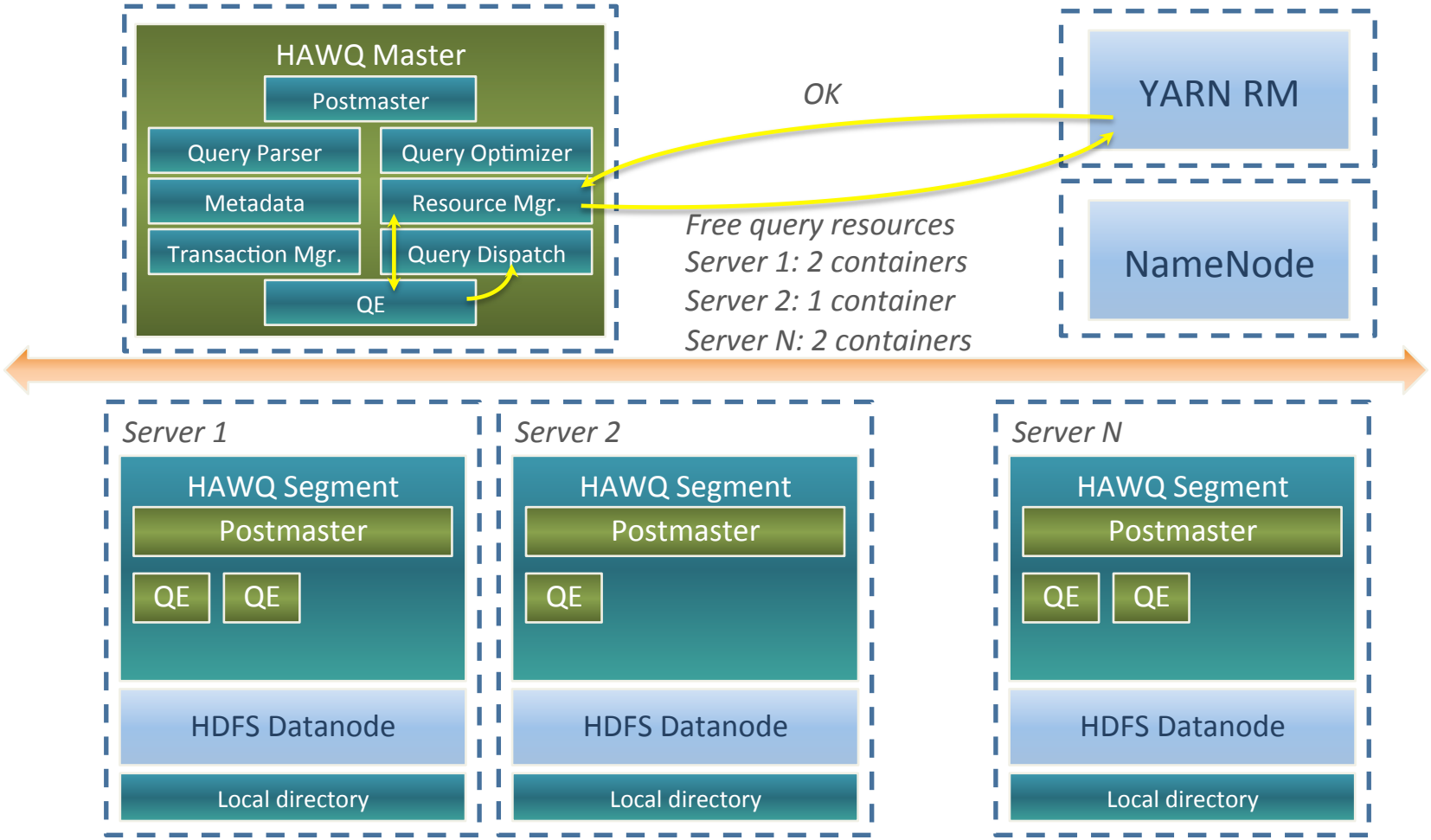
Query Example



Plan | Resource | Prepare | Execute | Result | Cleanup

HighLoad++

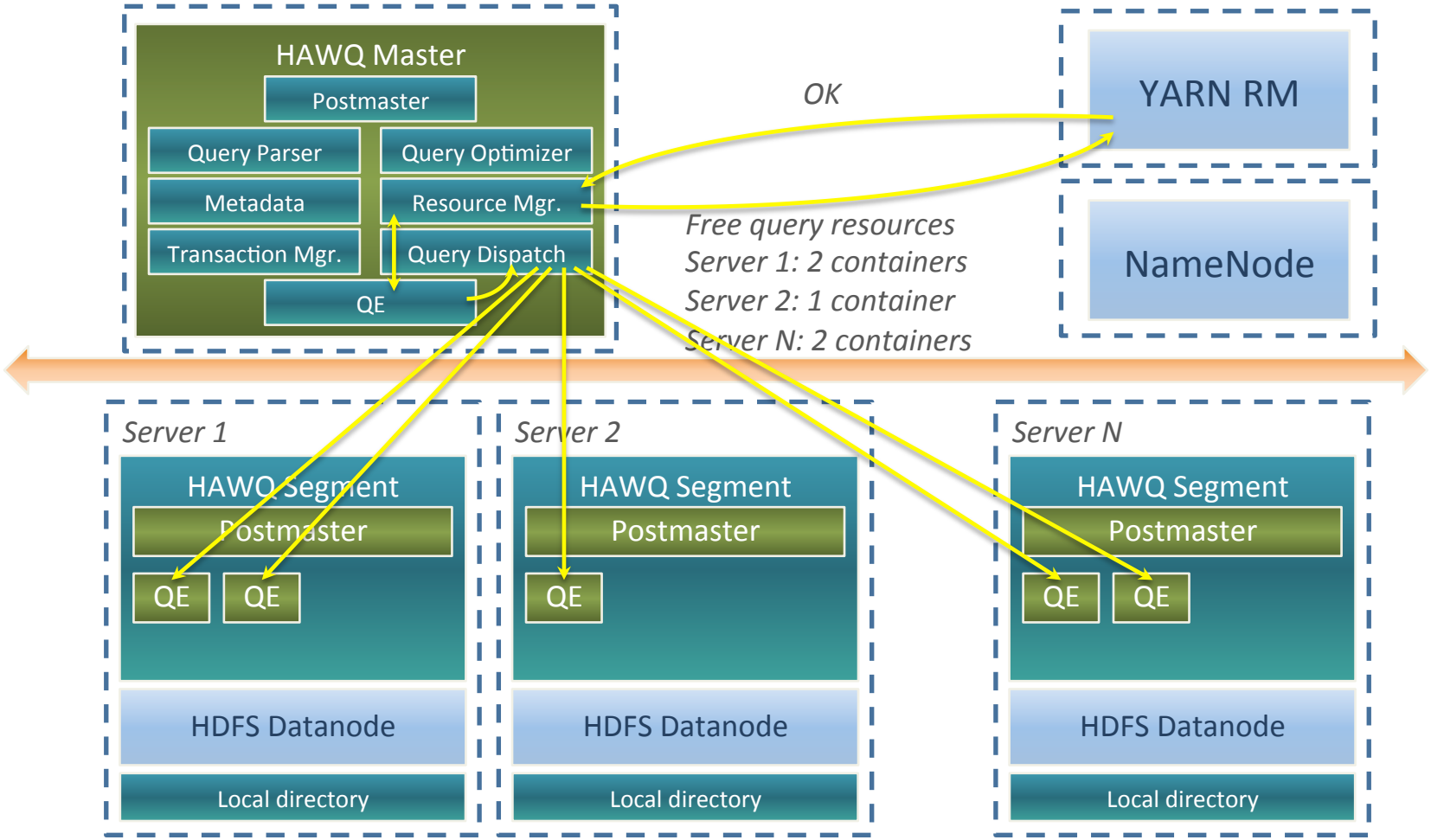
Query Example



Plan | Resource | Prepare | Execute | Result | Cleanup

HighLoad++

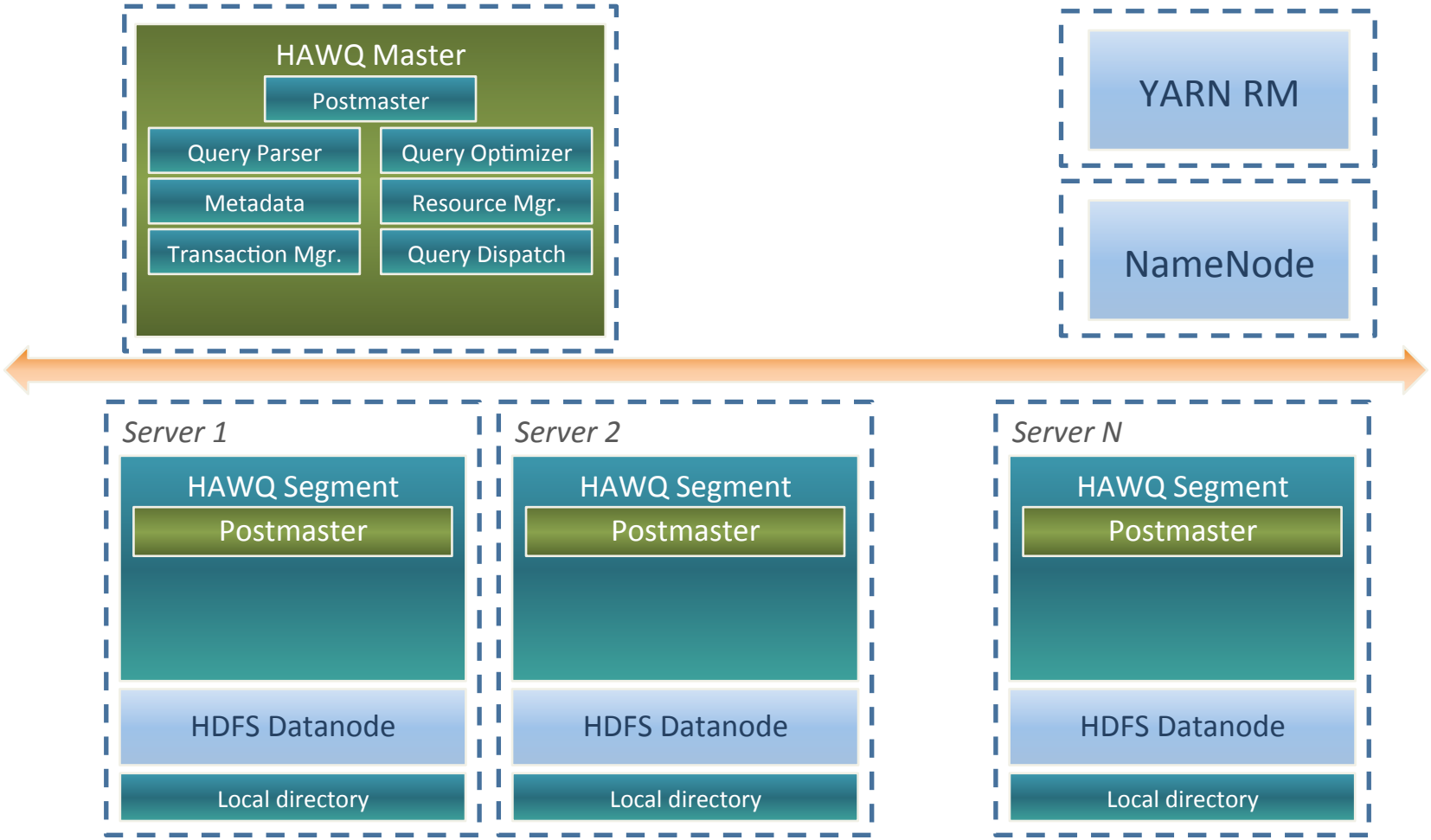
Query Example



Plan → Resource → Prepare → Execute → Result → Cleanup

HighLoad++

Query Example



Plan | Resource | Prepare | Execute | Result | Cleanup

Query Performance

- Data does not hit the disk unless this cannot be avoided

Query Performance

- Data does not hit the disk unless this cannot be avoided
- Data is not buffered on the segments unless this cannot be avoided

Query Performance

- Data does not hit the disk unless this cannot be avoided
- Data is not buffered on the segments unless this cannot be avoided
- Data is transferred between the nodes by UDP

Query Performance

- Data does not hit the disk unless this cannot be avoided
- Data is not buffered on the segments unless this cannot be avoided
- Data is transferred between the nodes by UDP
- HAWQ has a good cost-based query optimizer

Query Performance

- Data does not hit the disk unless this cannot be avoided
- Data is not buffered on the segments unless this cannot be avoided
- Data is transferred between the nodes by UDP
- HAWQ has a good cost-based query optimizer
- C/C++ implementation is more efficient than Java implementation of competitive solutions

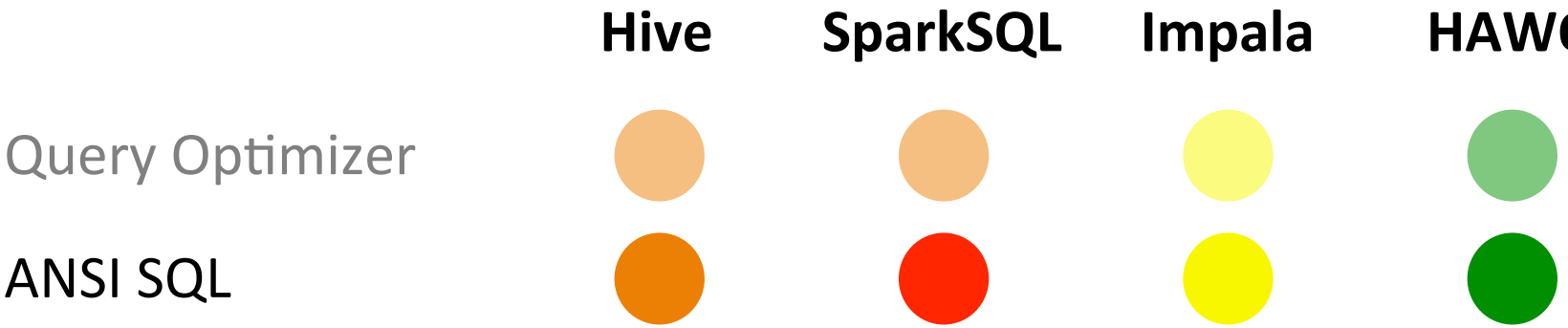
Query Performance

- Data does not hit the disk unless this cannot be avoided
- Data is not buffered on the segments unless this cannot be avoided
- Data is transferred between the nodes by UDP
- HAWQ has a good cost-based query optimizer
- C/C++ implementation is more efficient than Java implementation of competitive solutions
- Query parallelism can be easily tuned

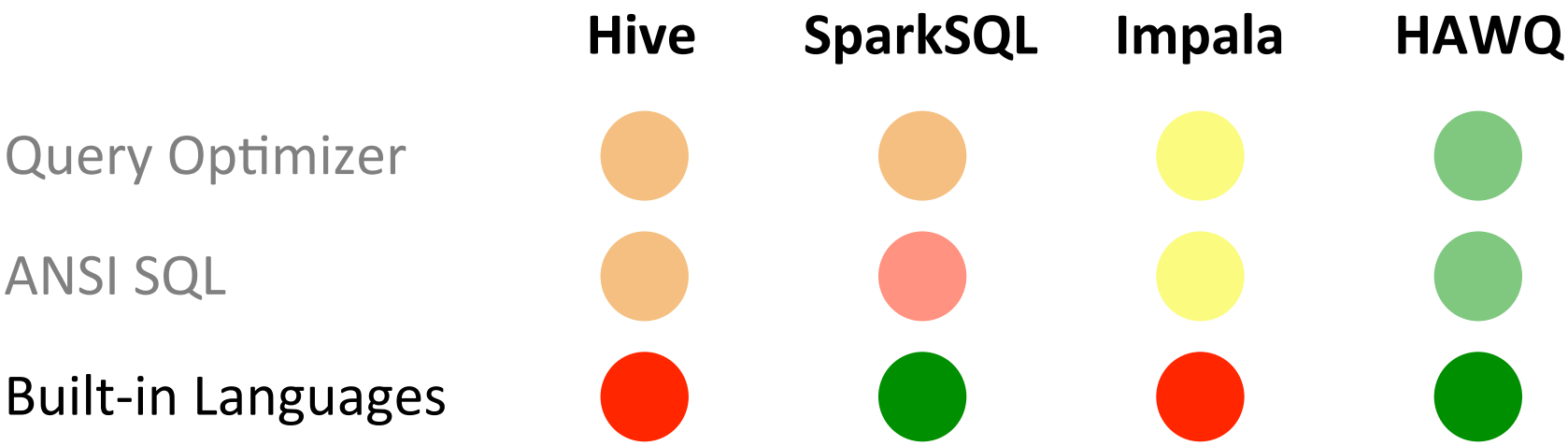
Competitive Solutions

	Hive	SparkSQL	Impala	HAWQ
Query Optimizer				

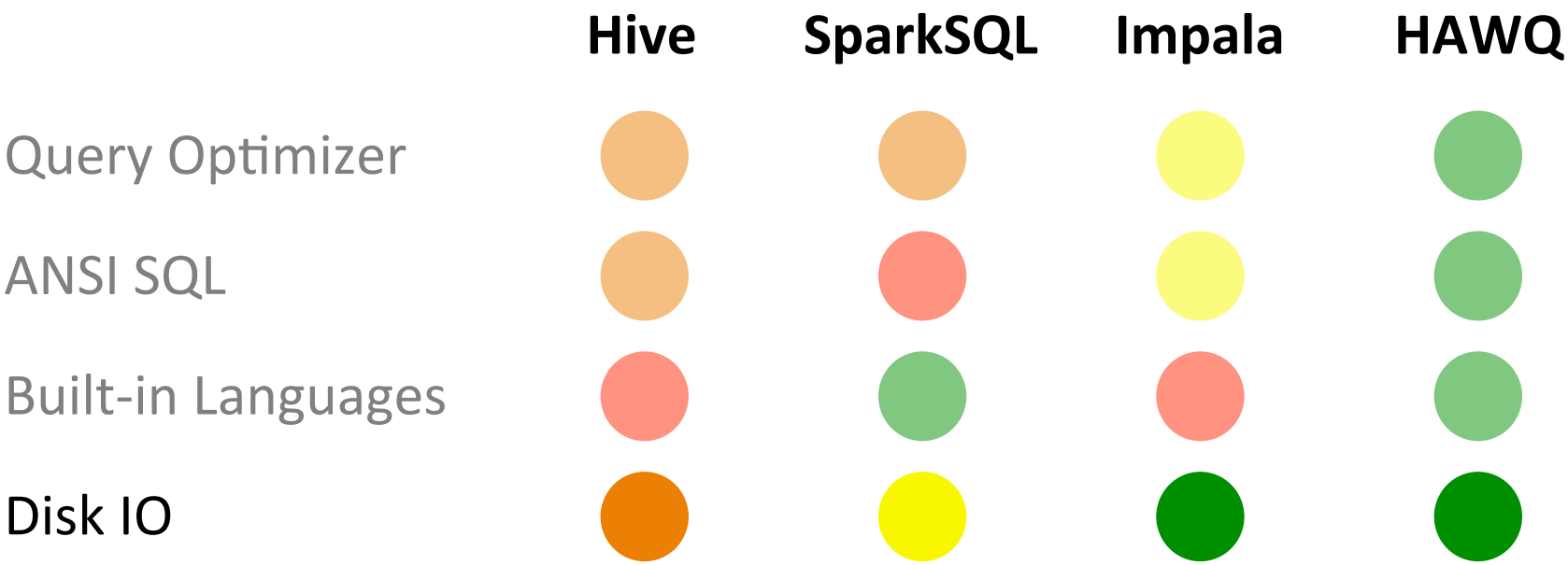
Competitive Solutions



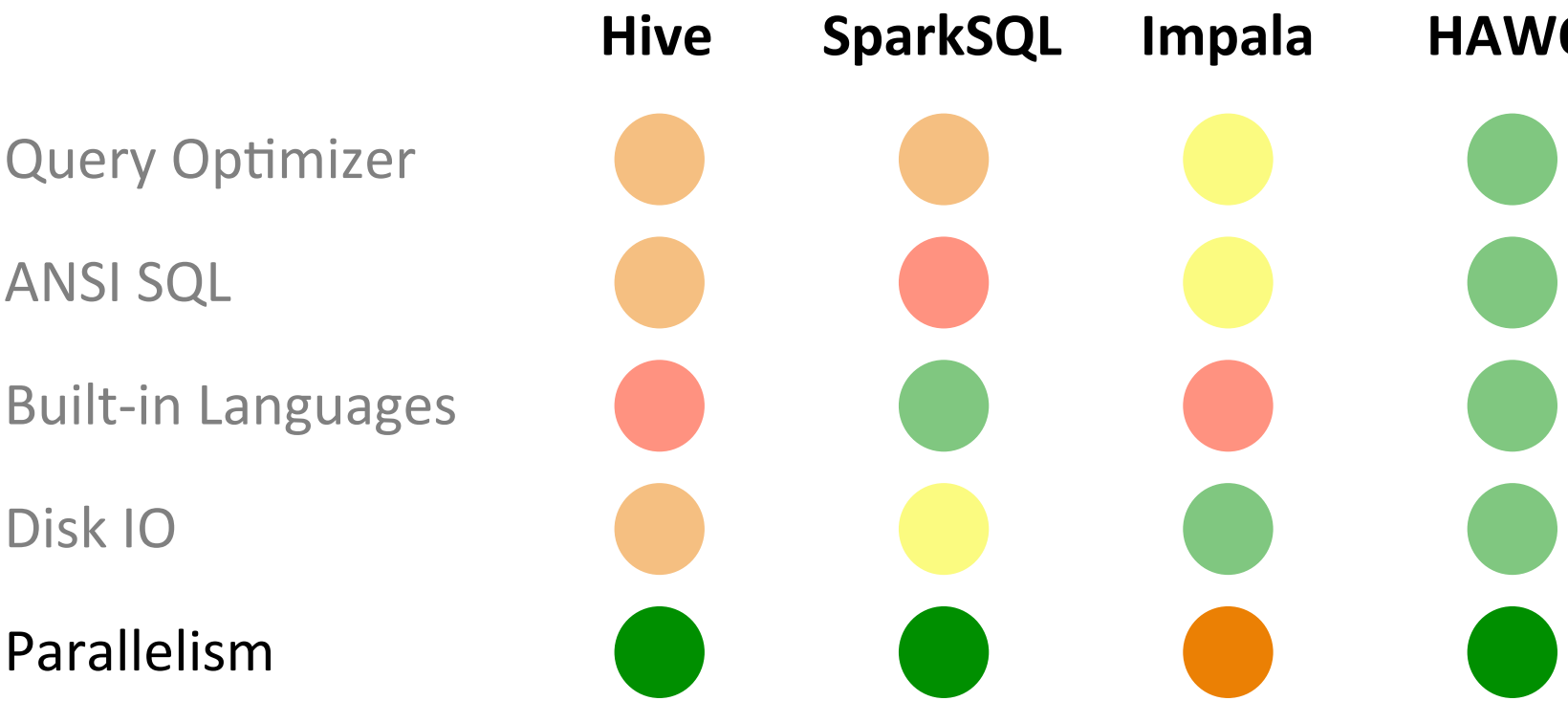
Competitive Solutions



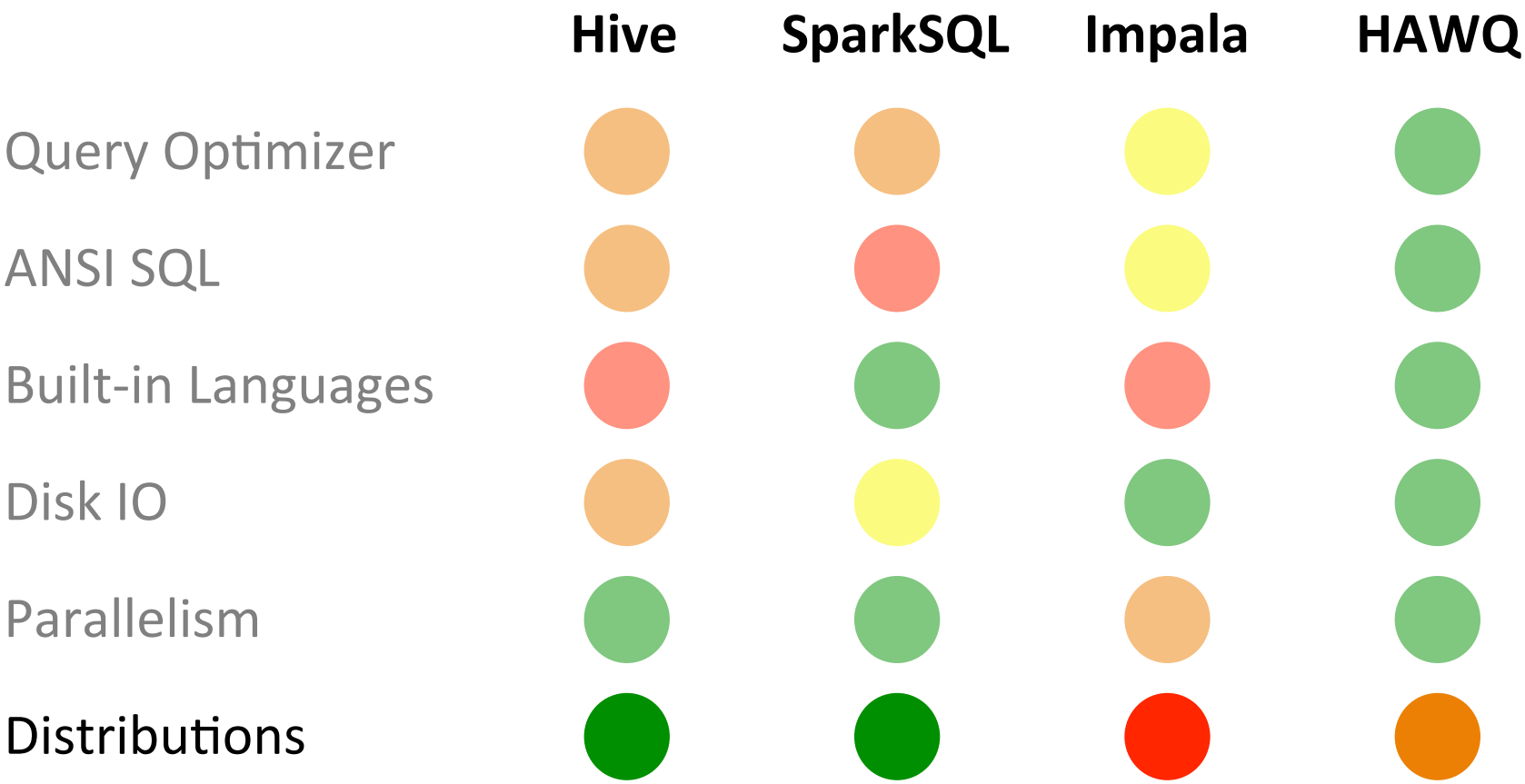
Competitive Solutions



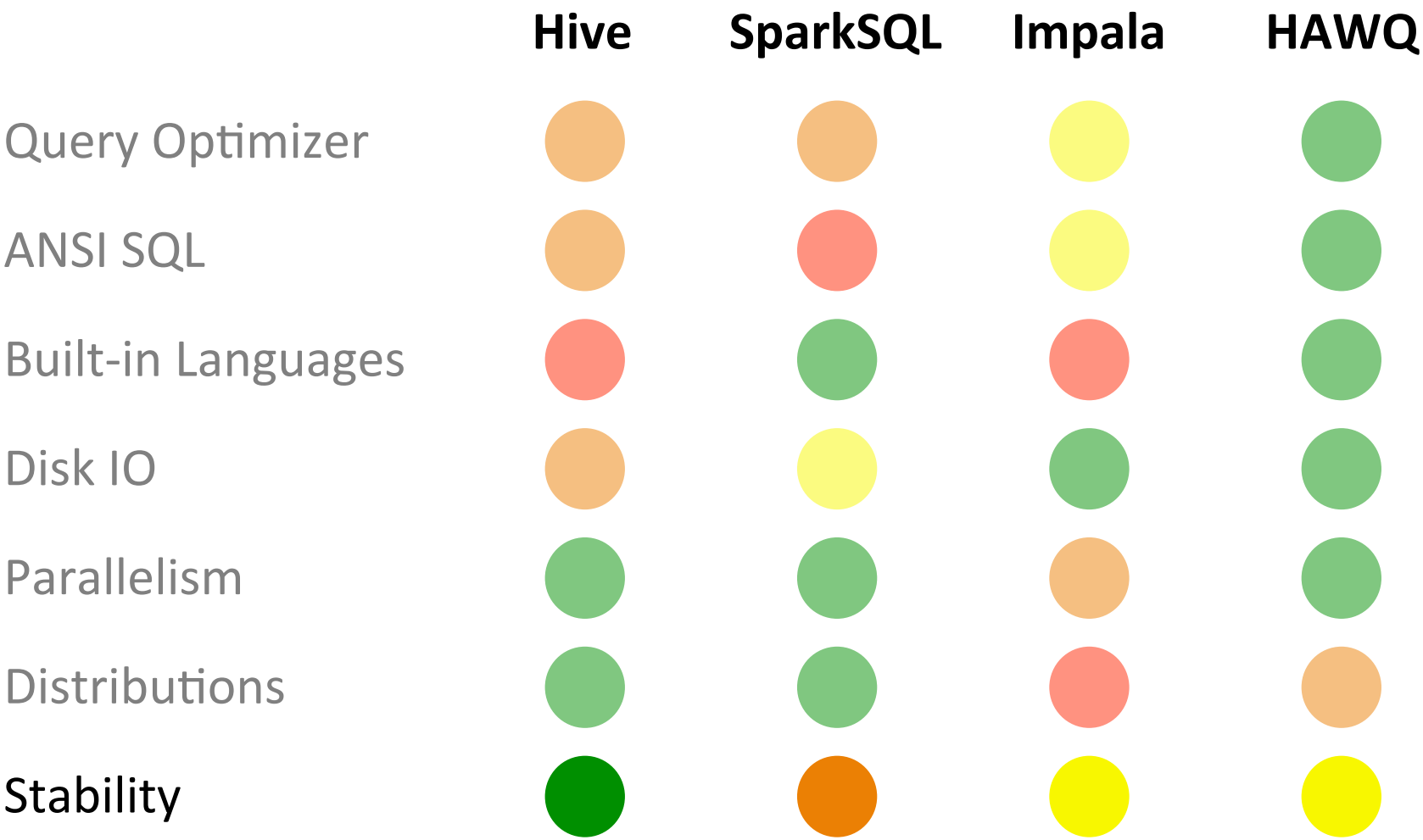
Competitive Solutions



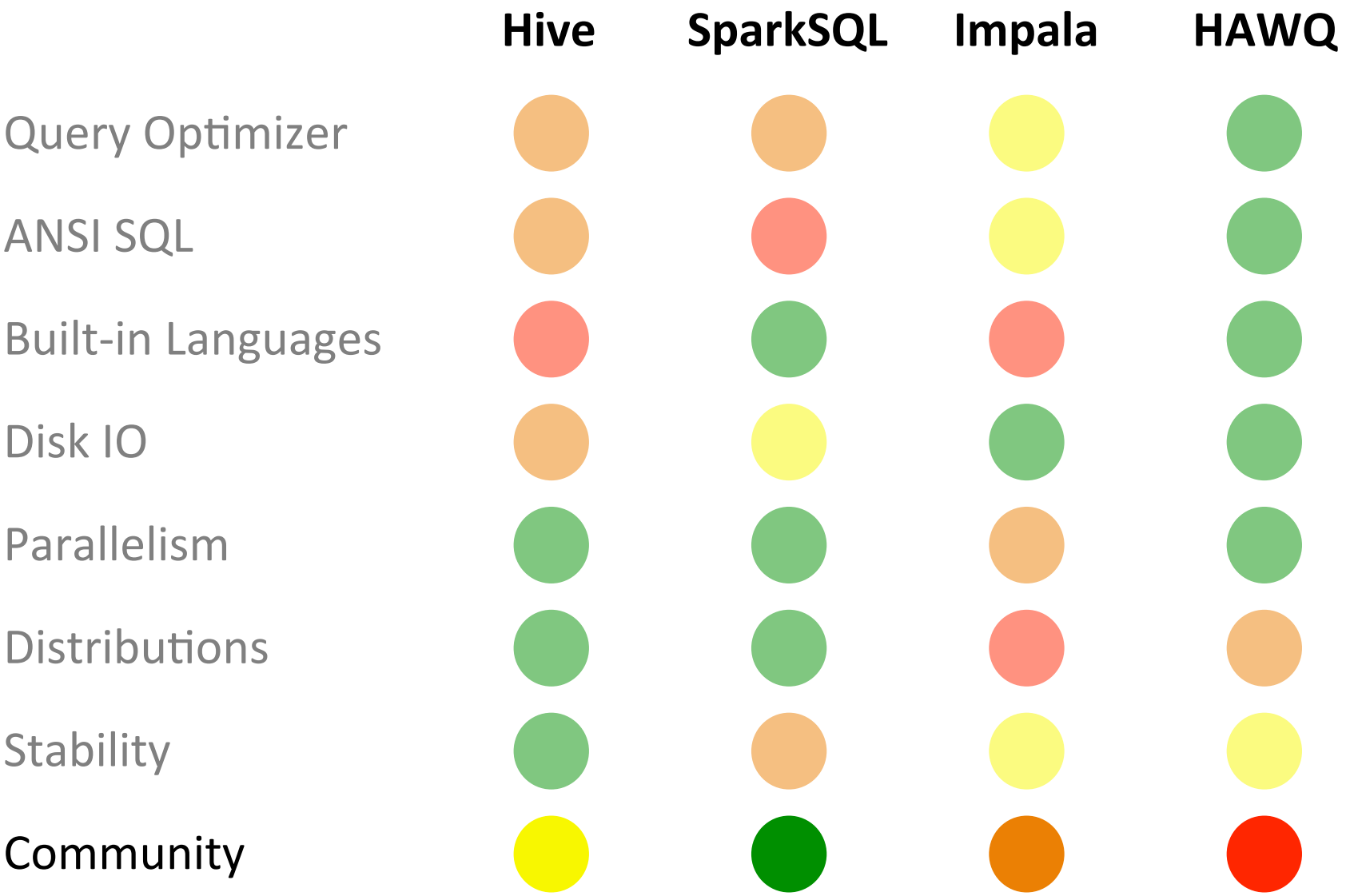
Competitive Solutions



Competitive Solutions



Competitive Solutions



Roadmap

- AWS and S3 integration

Roadmap

- AWS and S3 integration
- Mesos integration

Roadmap

- AWS and S3 integration
- Mesos integration
- Better Ambari integration

Roadmap

- AWS and S3 integration
- Mesos integration
- Better Ambari integration
- Cloudera, MapR and IBM Hadoop distributions native support

Roadmap

- AWS and S3 integration
- Mesos integration
- Better Ambari integration
- Cloudera, MapR and IBM Hadoop distributions native support
- *Make the SQL-on-Hadoop engine ever!*

Summary

- Modern SQL-on-Hadoop engine
- For structured data processing and analysis
- Combines the best techniques of competitive solutions
- Just released to the open source
- Community is very young

Join our community and contribute!

Questions

Apache HAWQ

http://hawq.incubator.apache.org

dev@hawq.incubator.apache.org

user@hawq.incubator.apache.org

Reach me on http://0x0fff.com