Online Expansion of Large-scale Data Warehouses

Jeffrey Cohen*
Brian Hagenbuch
Joy Kent
Christopher Pedrotti
Gavin Sherry*
Florian Waas
John Eshleman

* Presenting

The challenge

Customers want to capture and retain more data, longer.

They do so at a rate that outpaces *Moore's Law* and storage density improvements.

The (business) opportunity: grow at the pace of our customers.

The solution

... but that's *simple* - just add more cores and more drives.

The actual challenge: getting the data to those new drives.

This is *expansion.*

Detailed requirements

1. Minimal down time (*online* expansion)

2. No suspension of fault tolerance

3. Must meet operational expectations of DBAs
    - Configurability and transparency of process

Detailed requirements (cont.)

4. Must support data warehouse specific design patterns

5. An expanded system must behave like a freshly loaded system of the same scale

6. Minimal impact upon queries during expansion

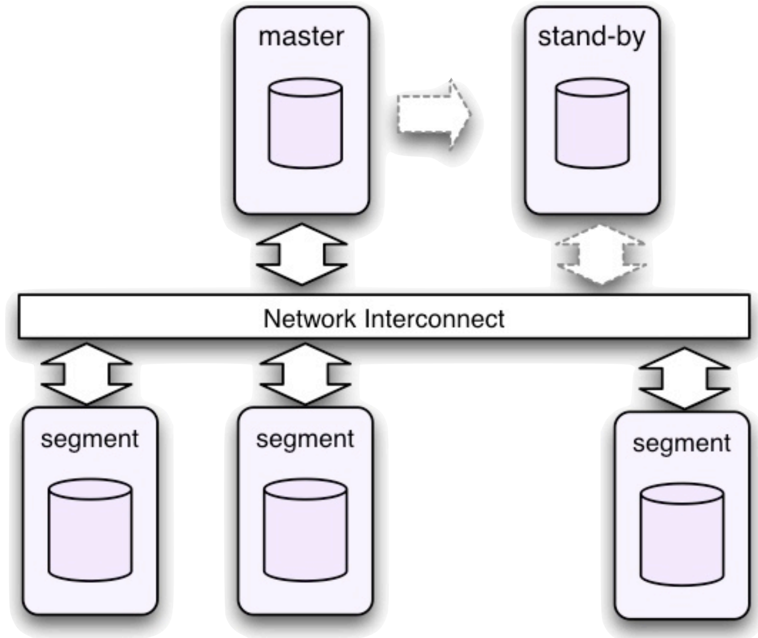
Greenplum System Architecture

Classic relational DBMS (PostgreSQL)

Single instance *illusion*

Shared nothing architecture

Master and worker (segment) nodes



Greenplum System Architecture

Data distributed randomly or by hash on user specified columns

Data distribution known globally

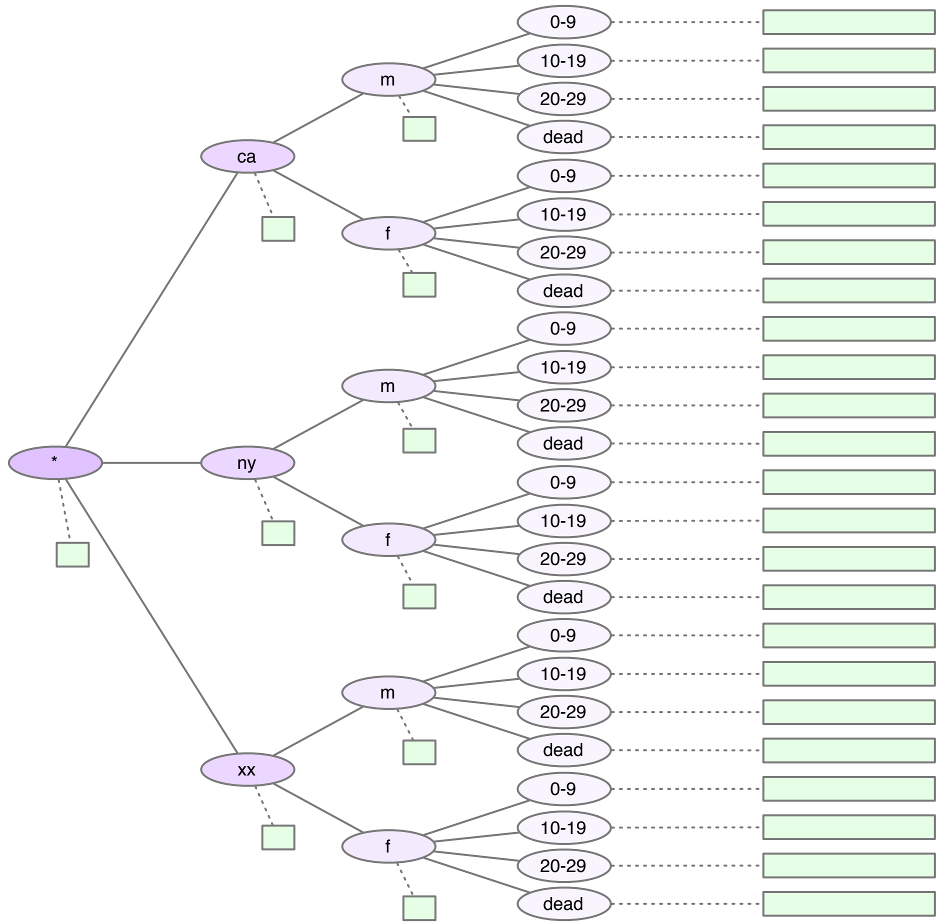
Fundamental parallelism: a worker per core, disk array and NIC

Distributed snapshot isolation

Greenplum System Architecture

Large tables partitioned into hierarchy of smaller *tables*

Partitions maybe distributed like their parents or randomly



Wednesday, 31 August 2011

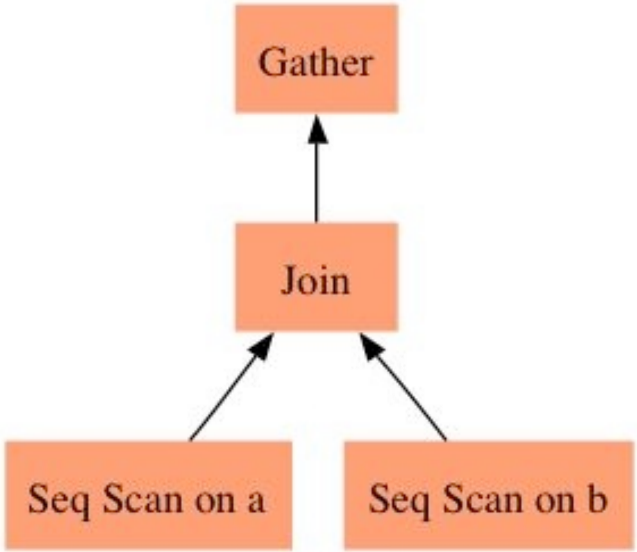
Parallel query execution fundamentals

Co-located tables can be joined locally

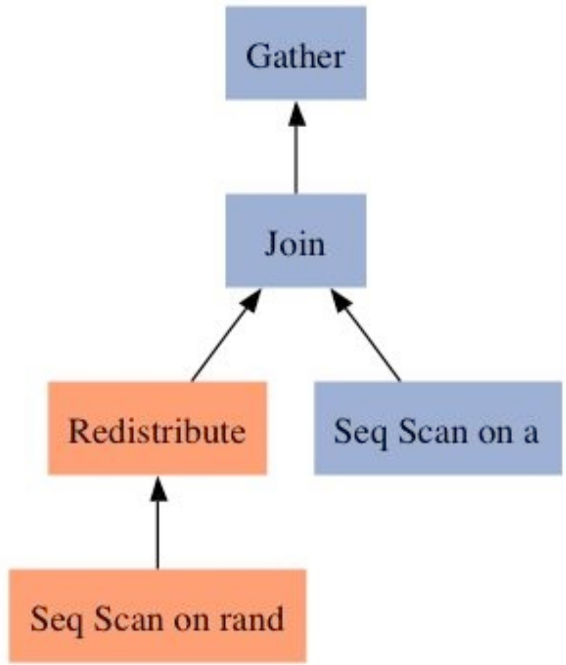
When this isn't the case, we arrange for execution time redistribution of data to satisfy the join

Highly skewed data distribution supported

Query Execution



Co-located join plan



Join plan requiring redistribution

Key insights

Partitions as a unit of expansion

We can always query randomly distributed data

Atomic rebuild using isolation mechanism

Expansion workflow

1. Expansion configuration planning
2. New nodes installed, configured and validated
3. Catalog cloned and installed on new nodes (offline)
4. Cache user table distribution settings (offline)
5. Set all tables to random distribution (offline)

Expansion workflow (cont)

6. Initialize expansion schedule

7. Iteratively expand tables/partitions according to schedule

8. Expansion may be monitored, reconfigured, paused, resumed as required

9. Done

Backend expansion primitive

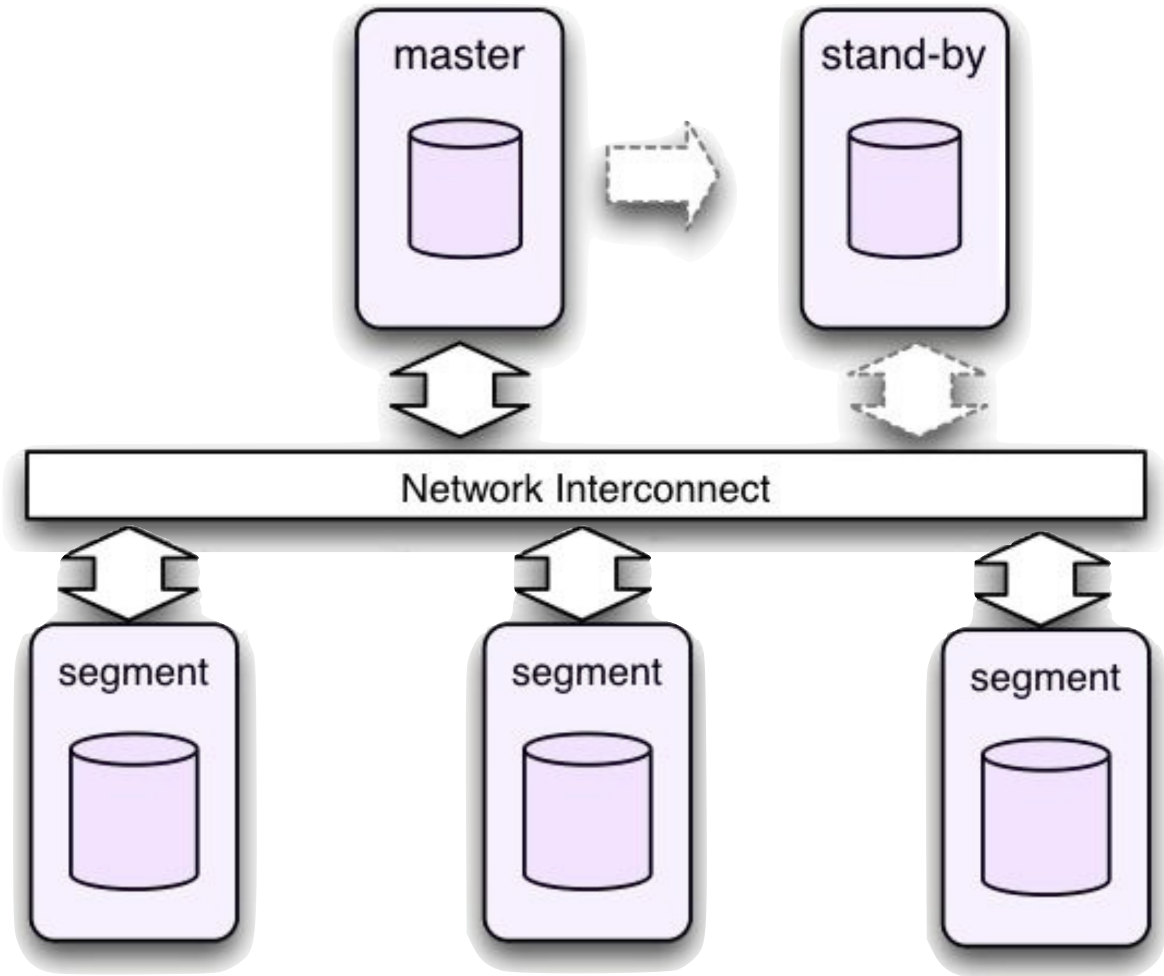
ALTER TABLE SET DISTRIBUTED BY ()

Pushes data through a graph from pre-expansion node to post-expansion node

Segment-to-segment parallelism

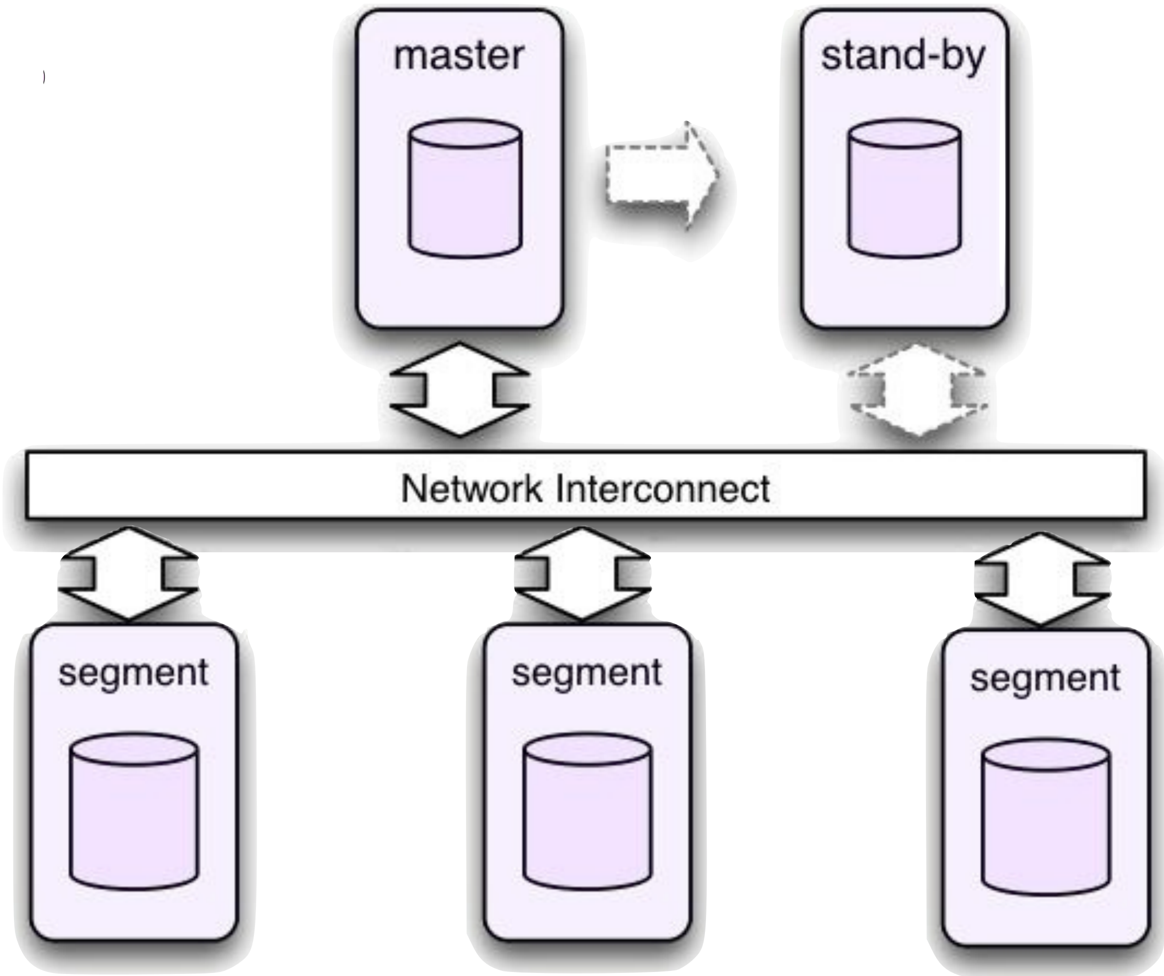
Analogous to INSERT ... SELECT

Greenplum System Architecture

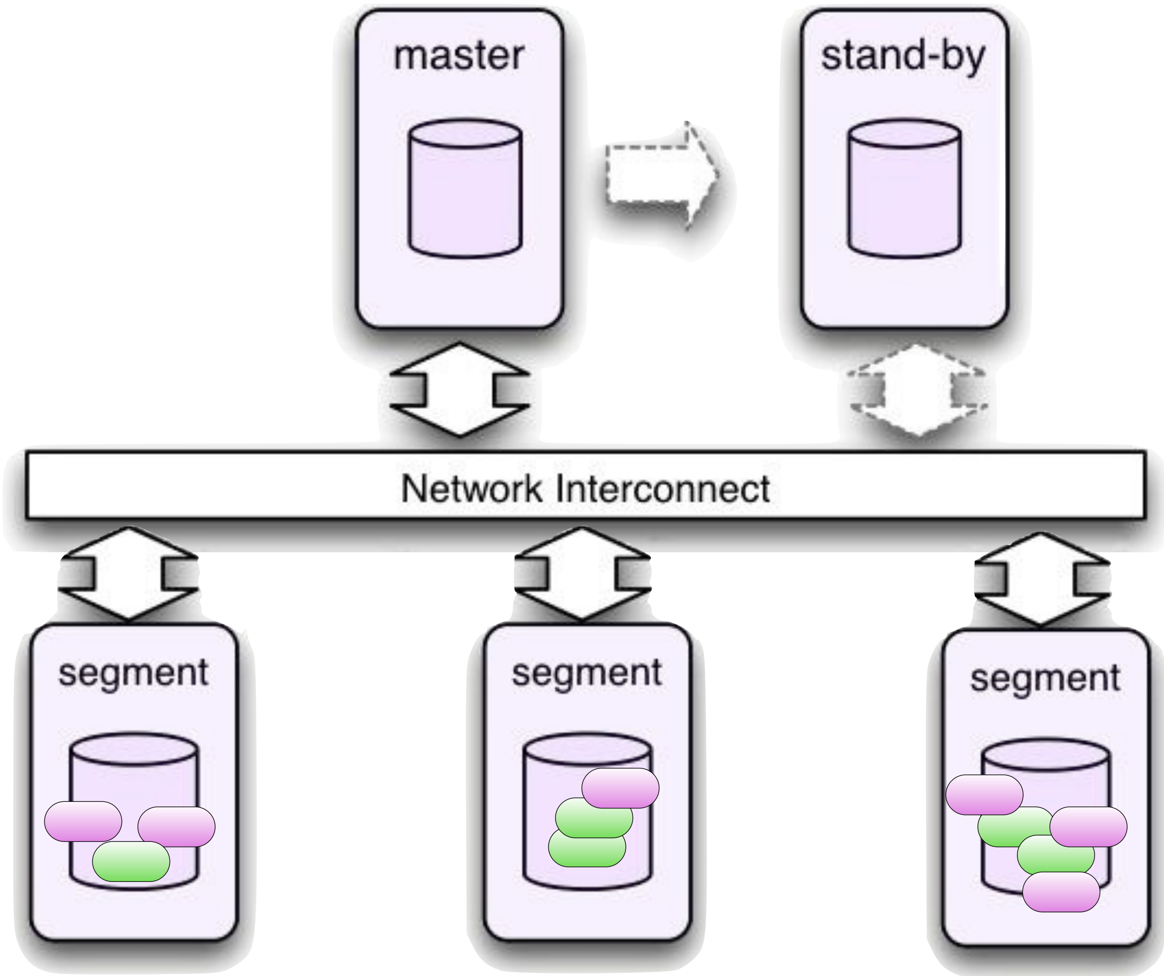


Wednesday, 31 August 2011

Greenplum System Architecture

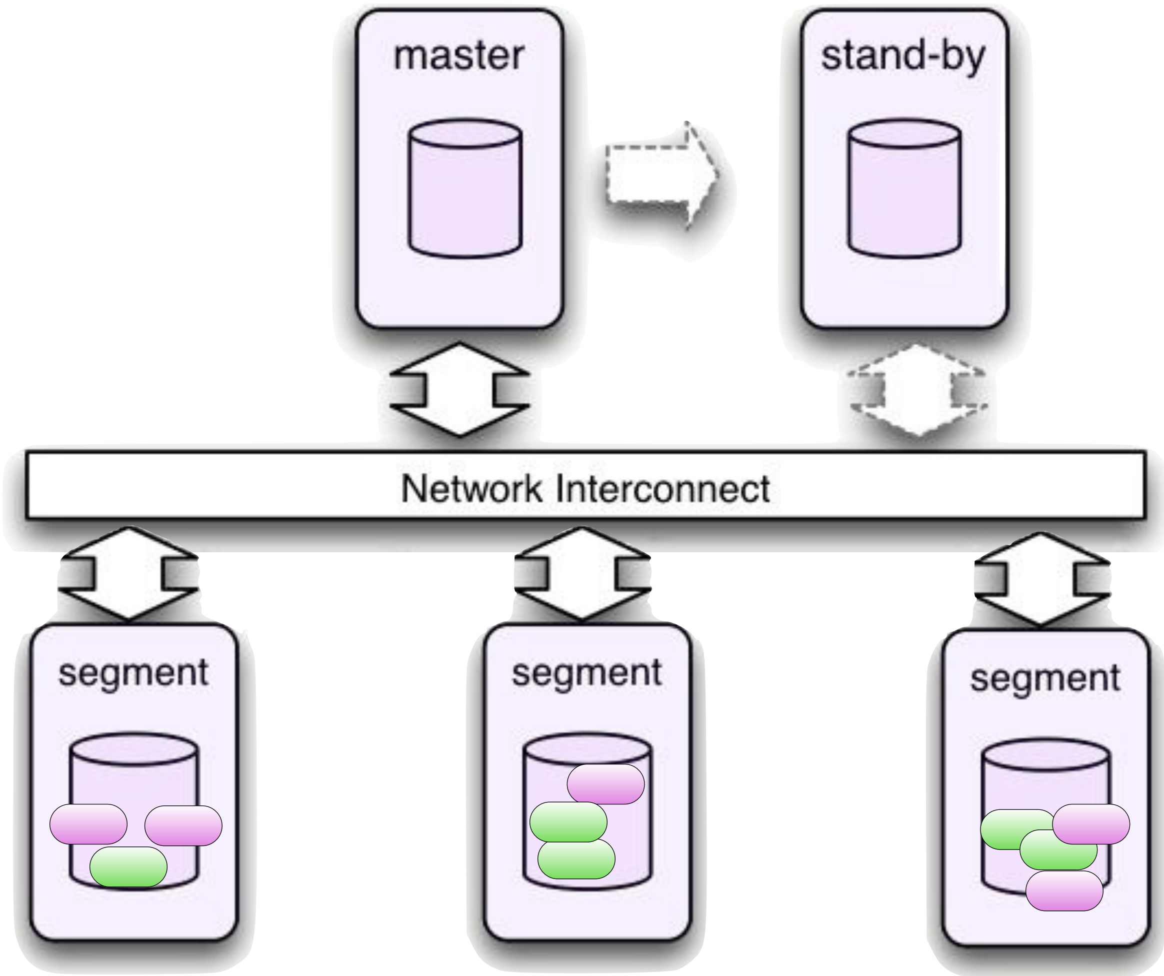


Greenplum System Architecture



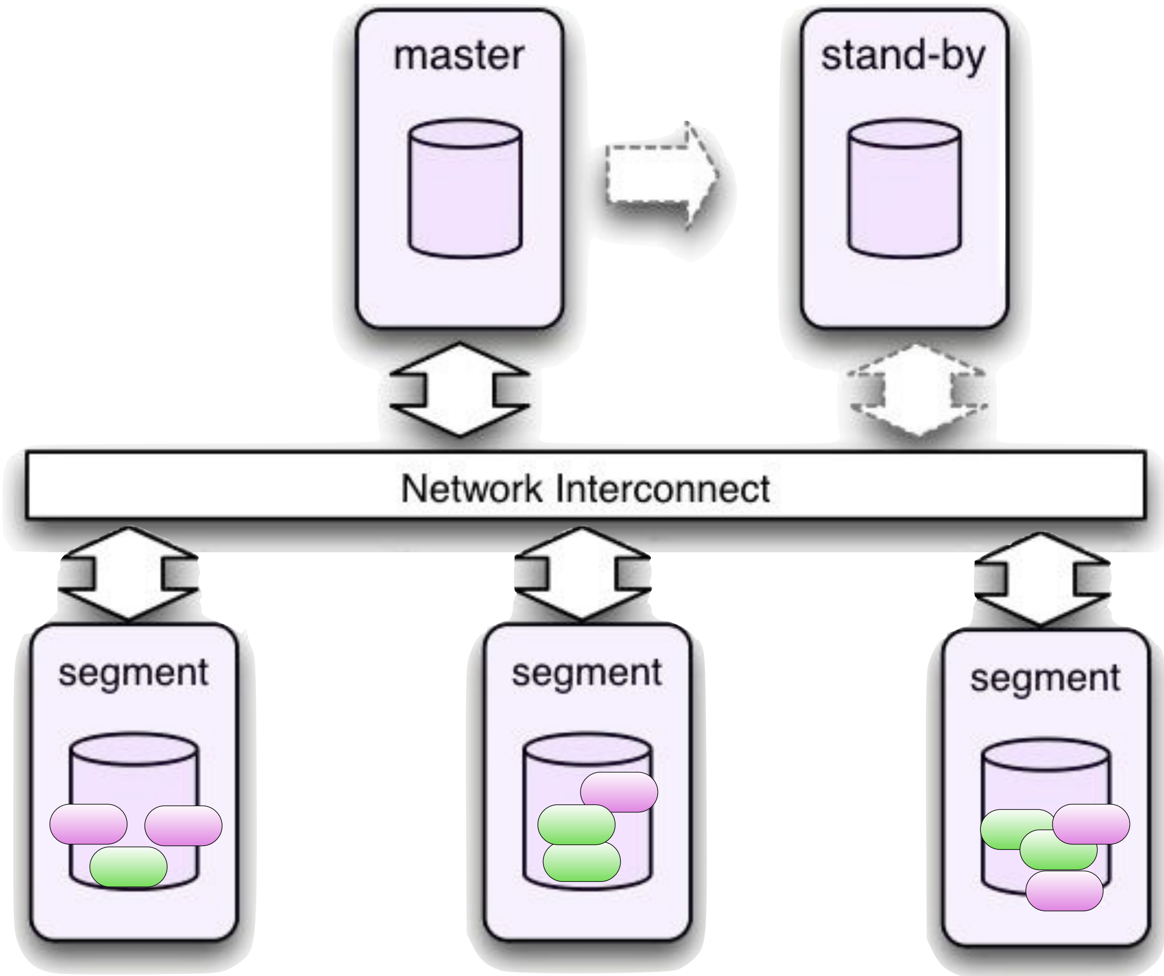
Wednesday, 31 August 2011

Greenplum System Architecture



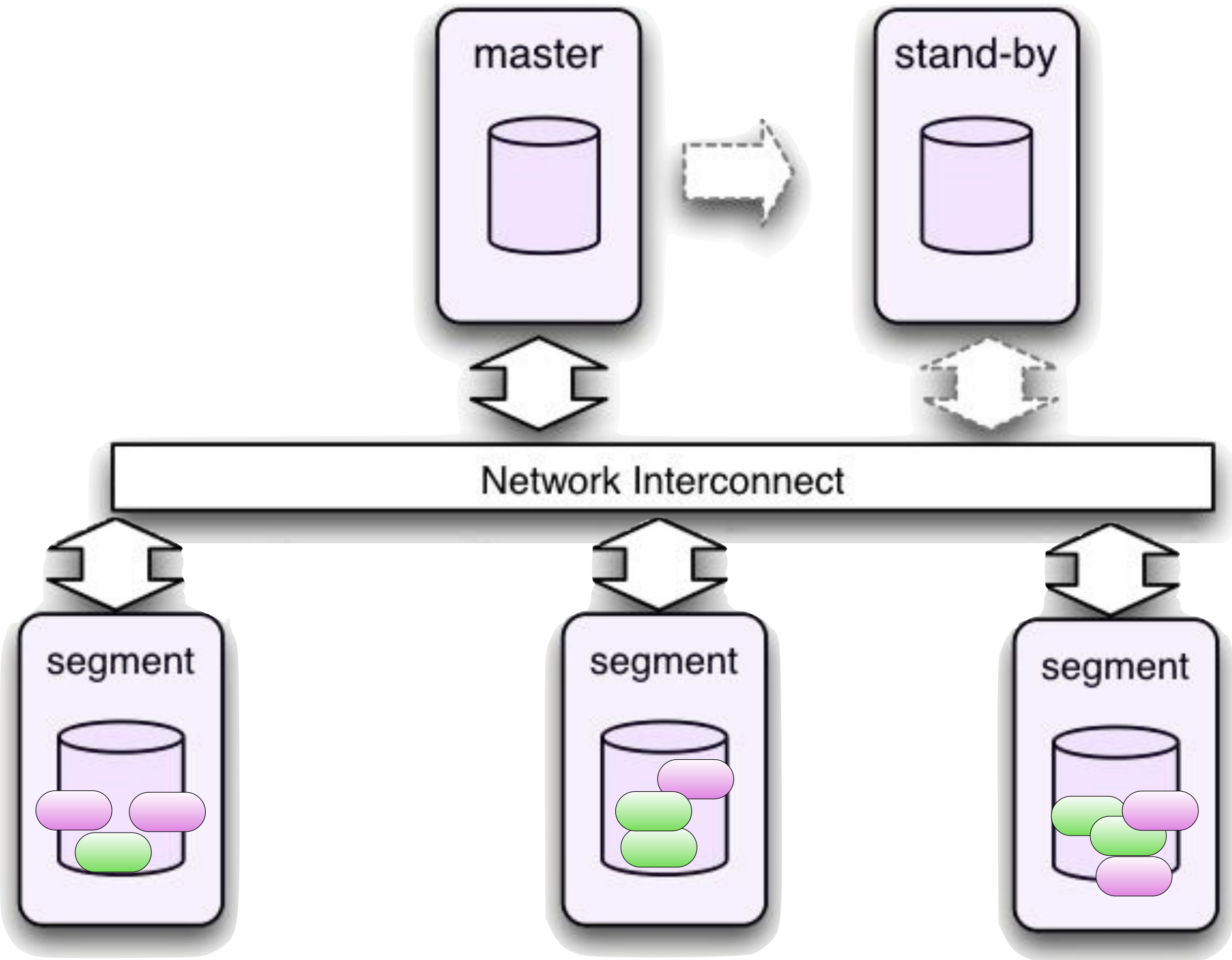
Wednesday, 31 August 2011

Greenplum System Architecture

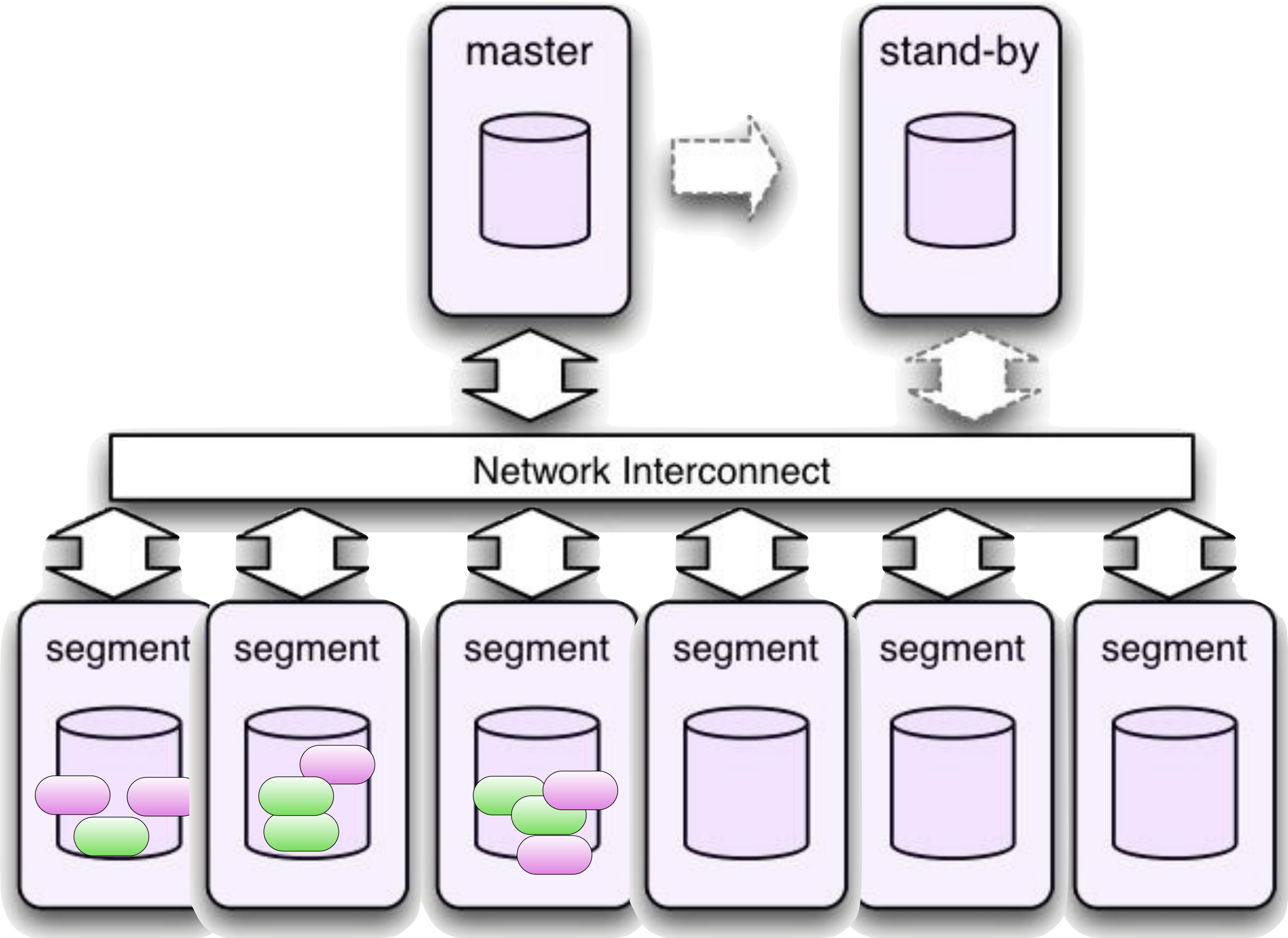


Wednesday, 31 August 2011

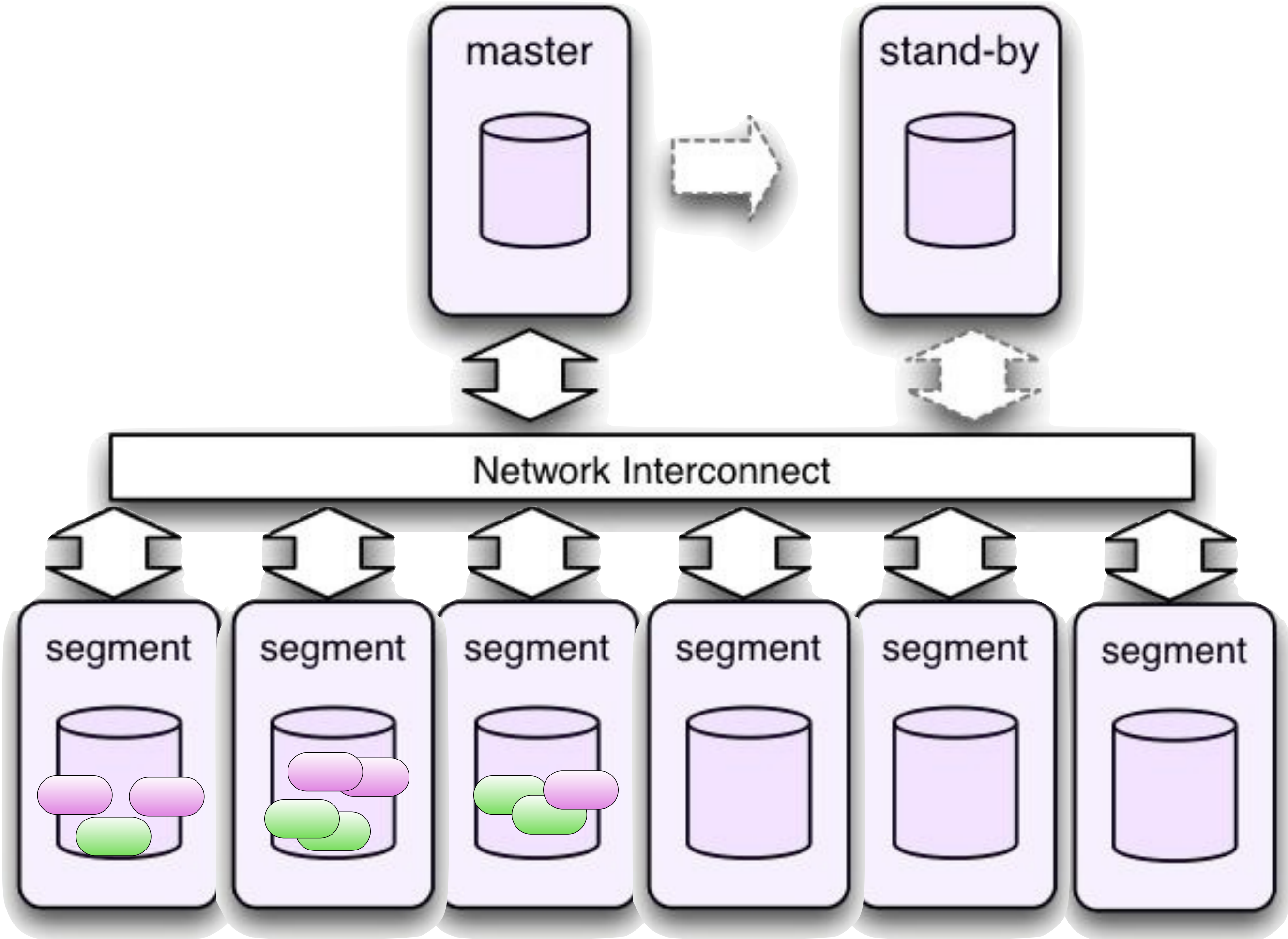
Greenplum System Architecture



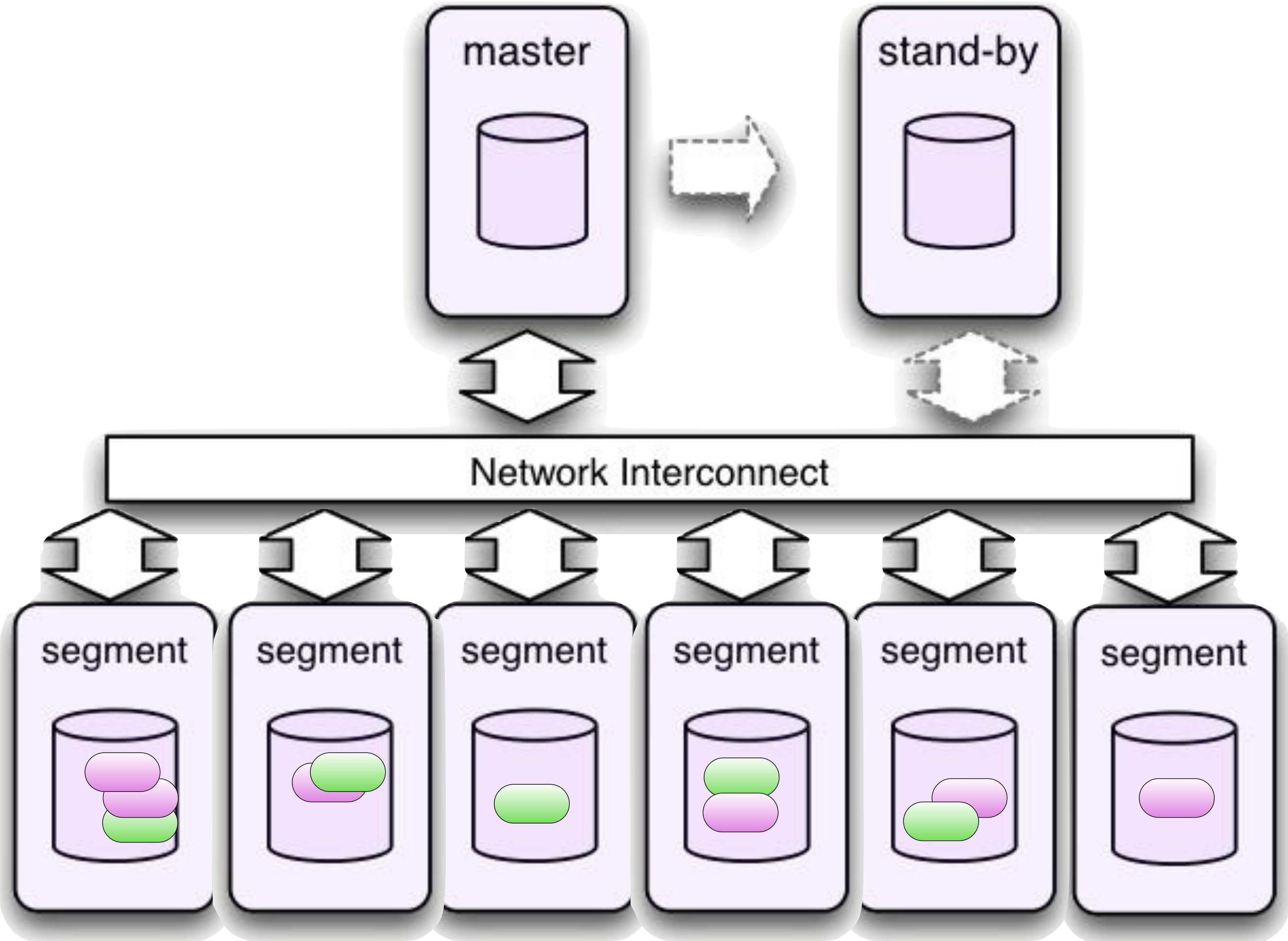
Greenplum System Architecture



Greenplum System Architecture



Greenplum System Architecture



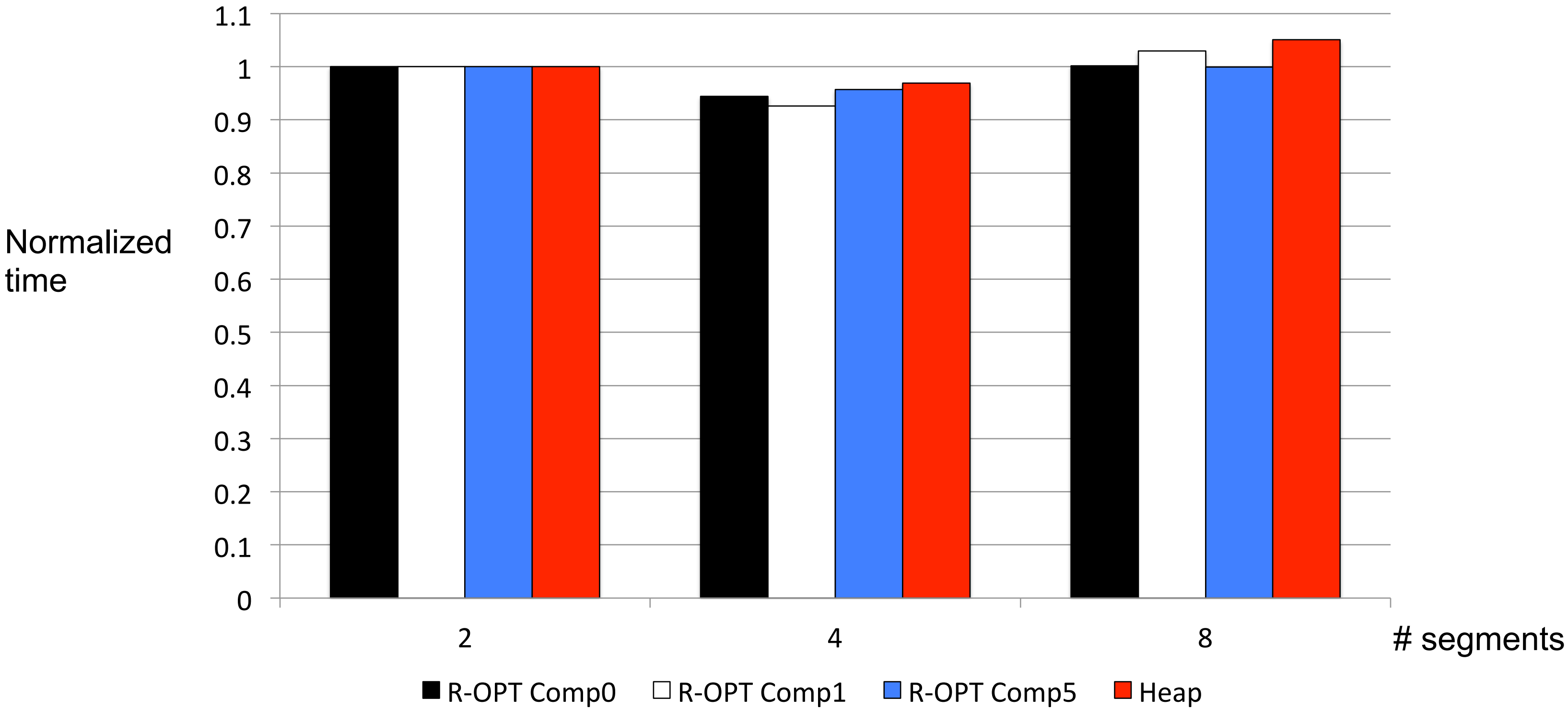
Verification and testing

Verify that redistribution scales with cluster size

Verify that redistribution scales with storage method

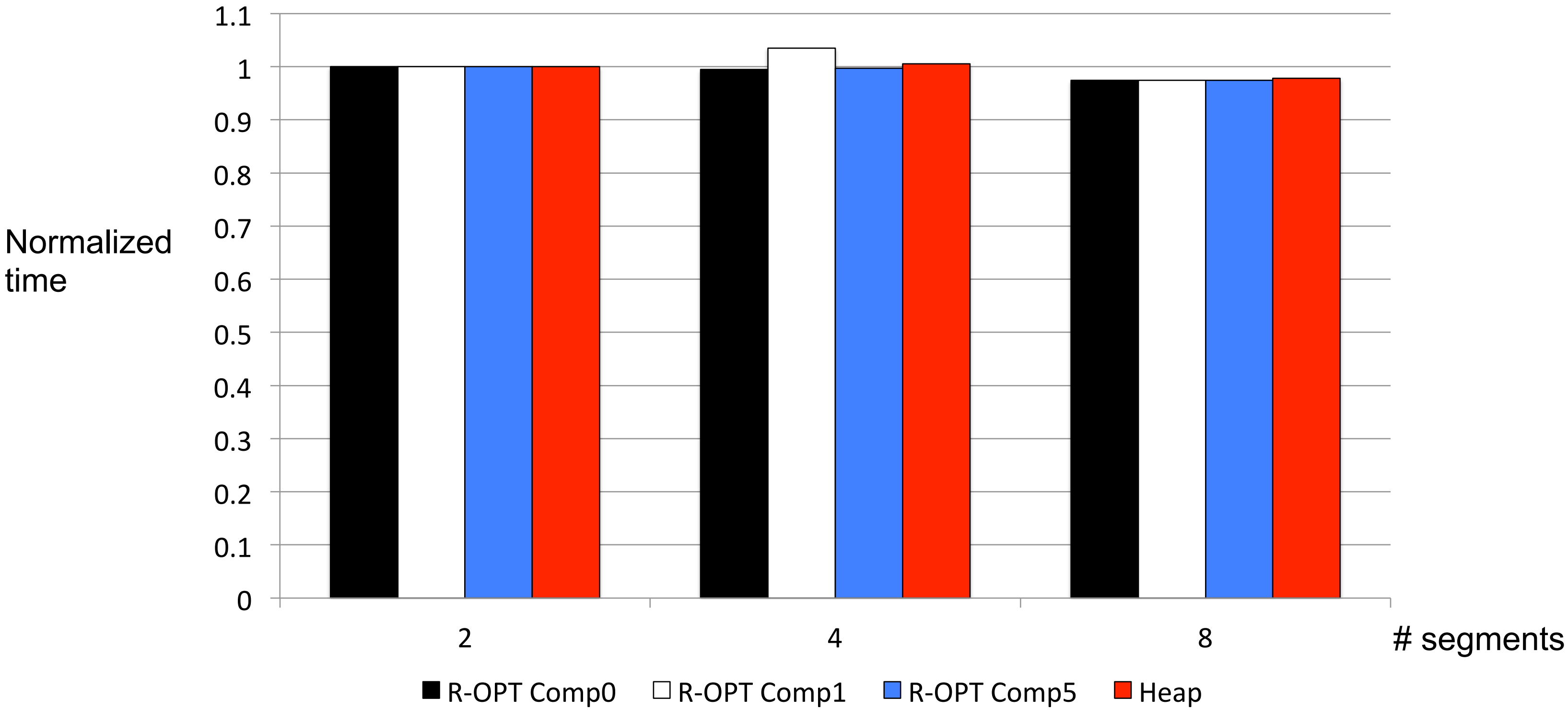
Verify that scaling is data independent

Scalability – TPCH data



N:N redistribution on clusters of different sizes
Data per segment remains constant

Scalability – customer data



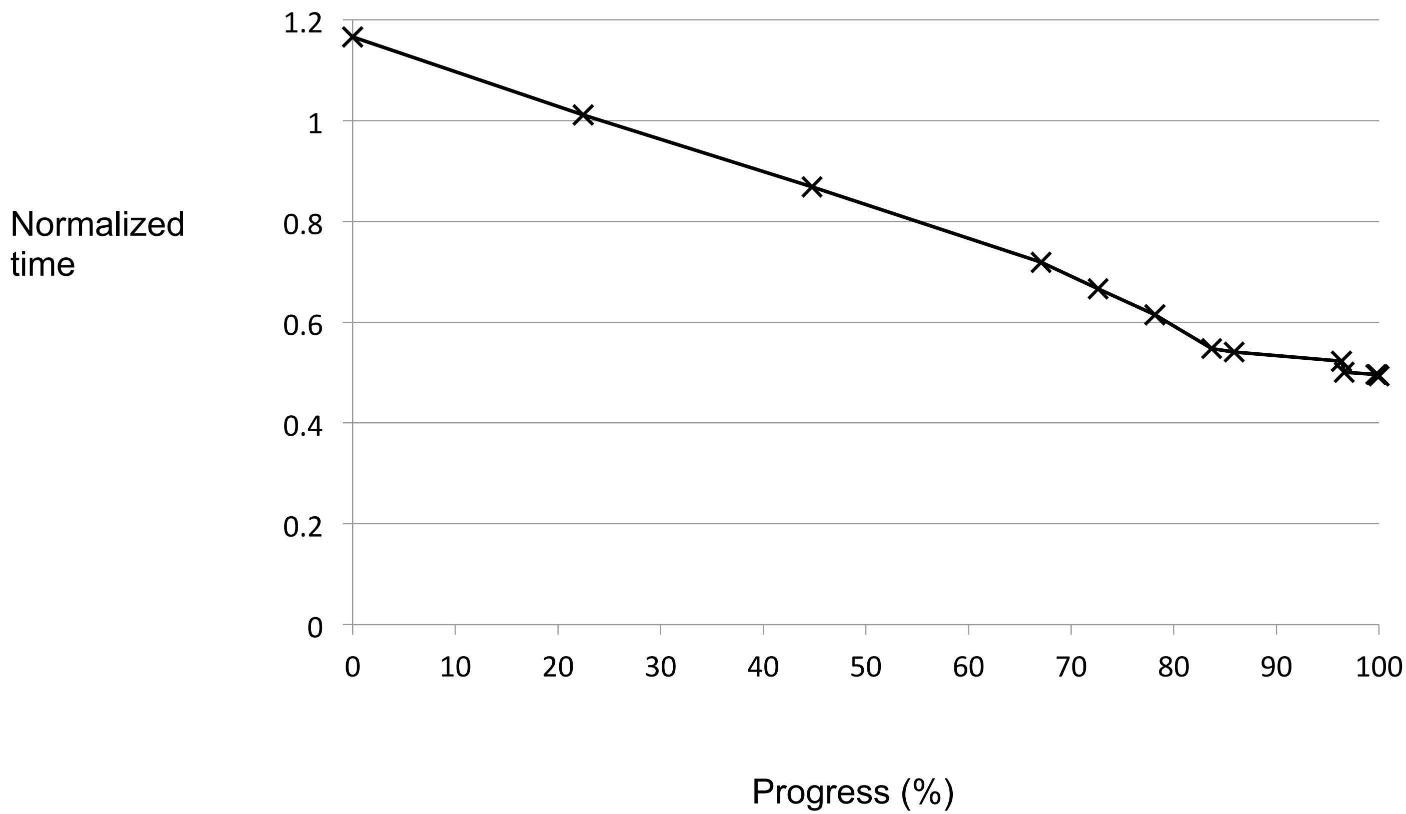
N:N redistribution on clusters of different sizes
Data per segment remains constant

Verification and testing

Verify minimal impact on workloads during expansion

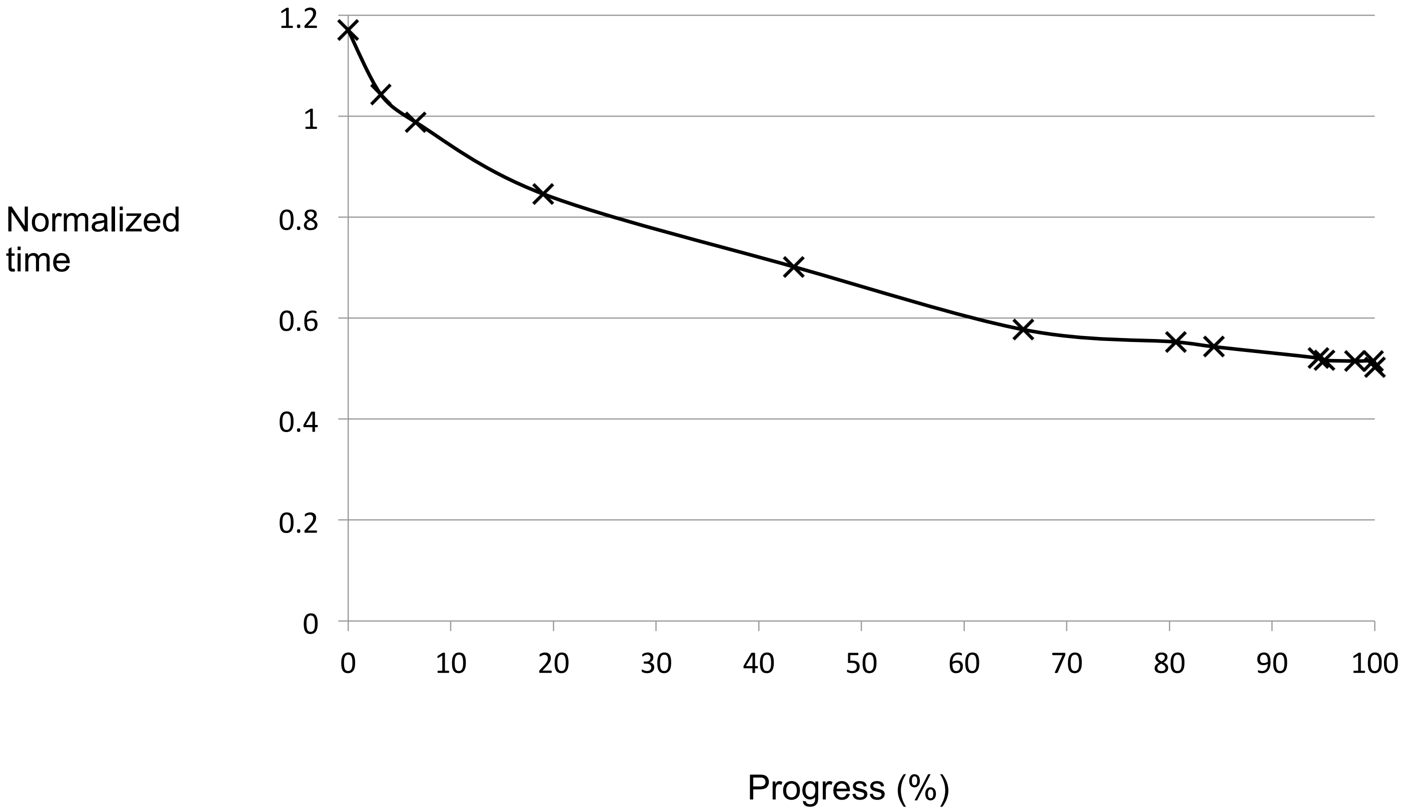
Verify benefit of optimal expansion schedule

Scalability and Testing – customer data



Effect of N:2N expansion on TPC-H runtime
Naïve expansion strategy

Scalability and Testing – customer data



Effect of N:2N expansion on TPC-H runtime
Best expansion strategy

Questions