Testing a Distributed Massively Parallel Database System

Florian Waas

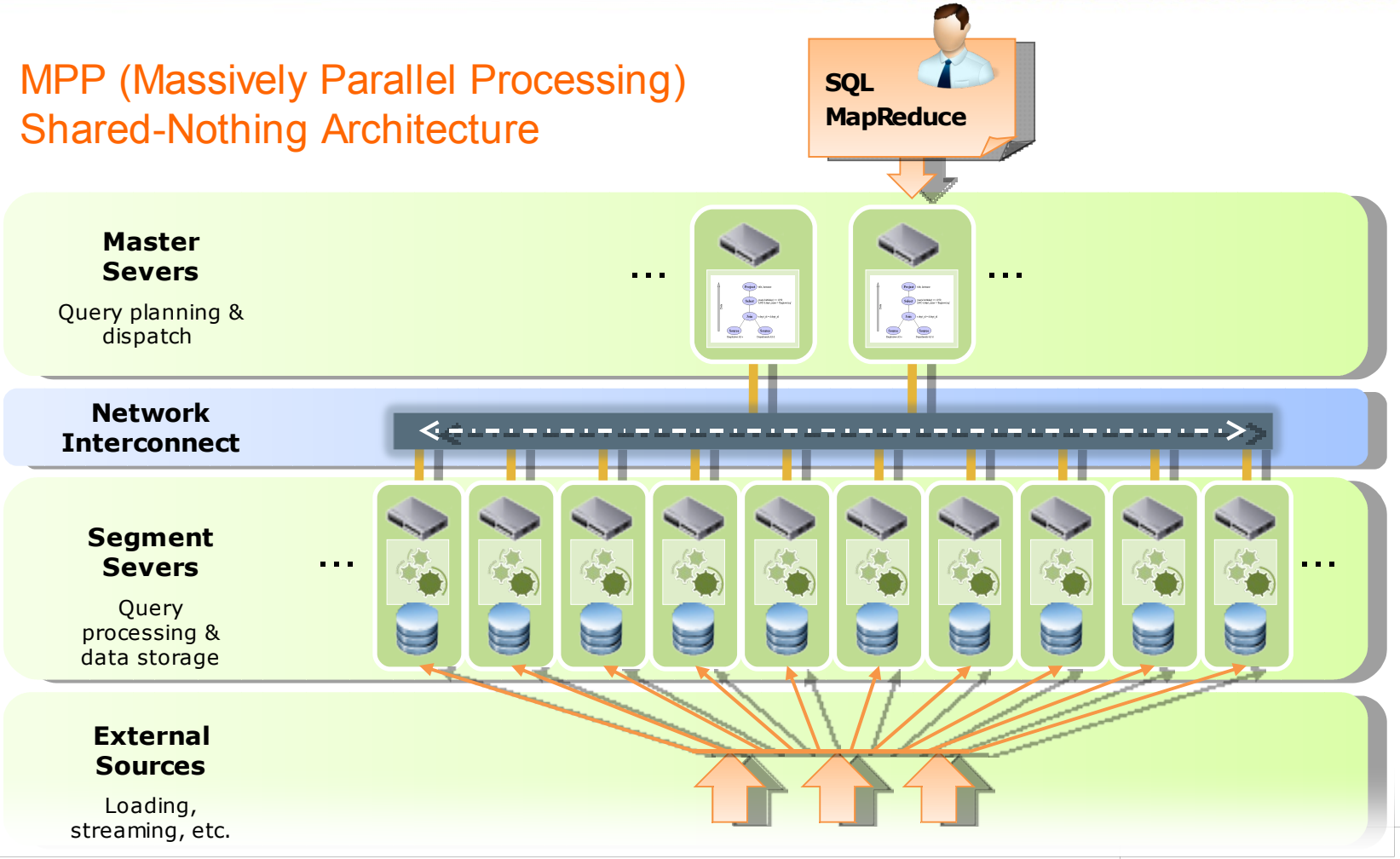
June 7, 2010

Greenplum Database

- Full-featured database system
  - Emphasis on support for OLAP
- Software-only solution
  - Not an appliance
  - No proprietary hardware
- Shared-nothing MPP architecture
  - High-end data analytics market

Greenplum Database Architecture

MPP (Massively Parallel Processing)
Shared-Nothing Architecture



Customer Profiles

- 100+ global enterprise customers
- 10's of TB to multi-PB
- 10's of tables to 15,000+
- 10's of concurrent users to 150+
- Integration w/ existing data analysis ecosystem, e.g., BI tools, ETL/ELT infrastructure
- Compute intensive, storage intensive
- Mixed workloads
- Urban myth: pure DW workloads

Specific Challenges

- Parallel processing
  - Distributed transactions
  - Distributed DDL
  - Parallel query processing
  - Multi-core aware query optimization
- Data management
  - In-cluster replication
  - DAS = unmanaged storage
  - TB's per box
    - ~10PB today, ~100PB "tomorrow"
- These are general trends for all DBMS's in the future

Test Challenges: General

- Functional tests, e.g., correctness of queries
  - Same/similar to conventional DBMS
- Stress tests
  - Similar; more components
- Performance
  - Same basic principles; much more data to capture
- In-cluster replication
  - Special hooks into product
- Scale testing
  - True at-scale testing prohibitively expensive
- Fault-tolerance
  - Elaborate external harnesses

Test challenges: Fault-tolerance

- Significantly larger test matrix
- Need to capture detailed system state
  - Observing distributed systems cannot rely on time stamps
- Fault scenarios
  - Network bisection
  - Node failures
  - Gradually degrading hardware
- Test strategies
  - Random fault injection
  - Network/drive failure simulations
  - Explicit K-safety scenarios

Greenplum

Up-ing the challenge

- Don't other people have the same problem?

- Database Systems are true enterprise software
  - Not single instances like Google, Yahoo!, Amazon.com, Facebook, etc.
  - No test by controlled-flooding
  - No developers to maintain it
- Higher standards of quality
- Transactional data management
- Test methods highly specialized
  - System specific
  - Programming language specific

The Road Ahead

- Distributed programming will become pervasive
- Above test challenges will be ubiquitous
- Cannot afford to implement these as black-box tests outside of system
  - Re-implements highly complex logic
  - Too costly to maintain
- Need textbook of distributed test methodologies
  - Checklists for fault-scenarios
  - Integration with existing methods
  - Both theory and tools
- Lots of exciting work ahead!