Building PetaByte Warehouses with Unmodified PostgreSQL

Emmanuel Cecchet
Member of Research Staff
May 21st, 2009

Topics

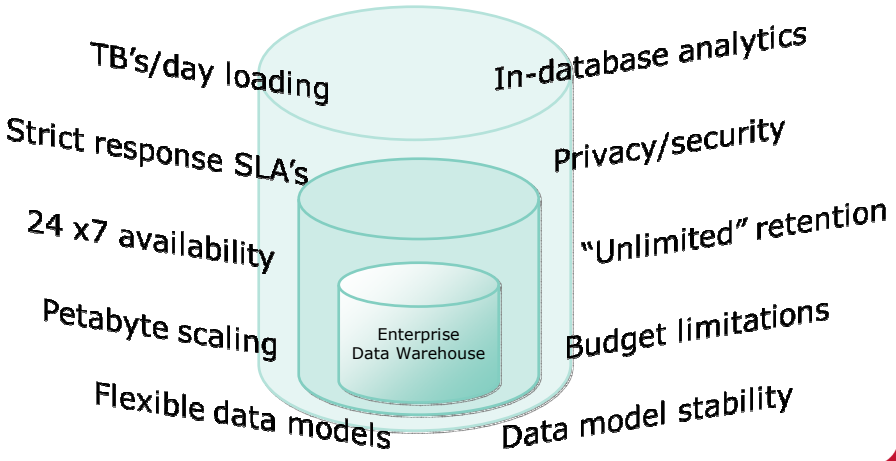
Introduction to frontline data warehouses

PetaByte warehouse design with PostgreSQL

Aster contributions to PostgreSQL

Q&A

Enterprise Data Warehouse Under Stress



ASTER

Offloading The Enterprise Data Warehouse

- Petabytes source data
- 24 x7 Availability
- TB/Day Load capacity
- In-database transforms
- Rapid data access

- Petabytes detailed data
- Low cost/TB
- Flexible compression
- On-line access
- Aging out to off-line

ASTER

Requirements for Frontline Data Warehouses

"Always Parallel"

- All tasks fully parallelized for maximum performance
- Query, load/export, and backup/restore

"Always On"

- Eliminates unplanned downtime
- Minimizes planned downtime

In-Database Analytics

- Analyze data in-place vs. extract then analyze

ASTER

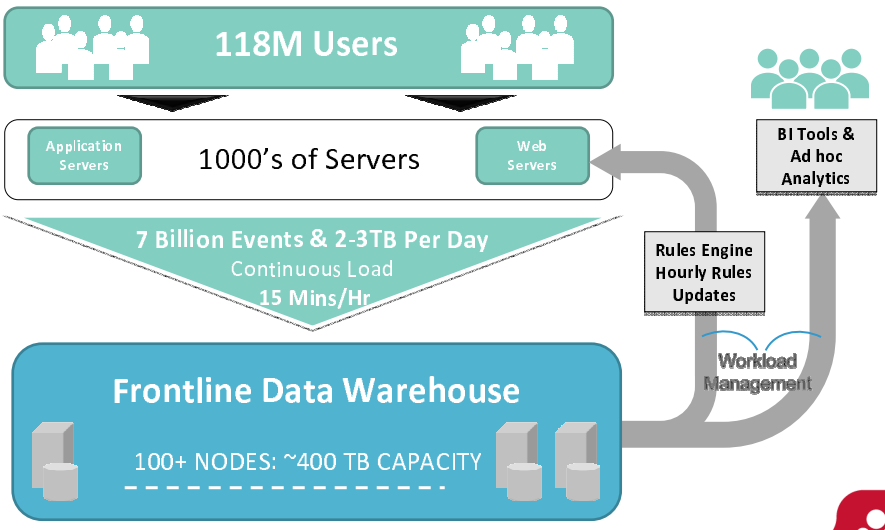
Who is Aster Data Systems?

Aster nCluster is a software-only RDBMS for large-scale frontline data warehousing

- High performance → "Always Parallel" MPP architecture
- High availability → "Always On" on-line operations
- High value analytics → In-Database MapReduce
- Low cost → Petabytes on commodity HW

ASTER

MySpace Frontline Data Warehouse



ASTER

Topics

Introduction to frontline data warehouses

PetaByte warehouse design with PostgreSQL

Aster contributions to PostgreSQL

Q&A

ASTER

Petabyte Datawarehouse Design

PostgreSQL as a building block

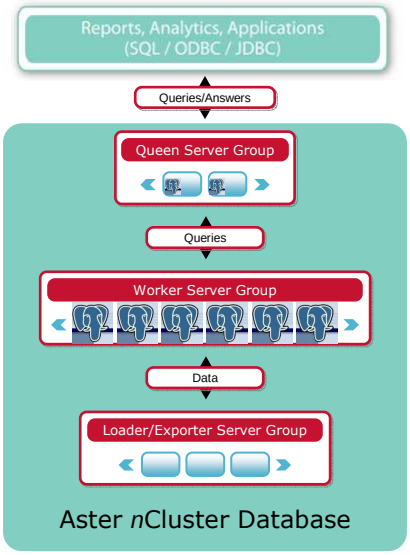
- Do not hack PostgreSQL to serve the distributed database
- Build on top of mainline PostgreSQL
- Use standard Postgres APIs

Service Oriented Architecture

- Hierarchical **query planning** and **optimization**
- Shared nothing **replication** treating Postgres and Linux as a service
- **Compression** at the OS level transparently to PostgreSQL
- In-database **Map-Reduce** out-of-process

ASTER

Aster *n*Cluster Database



Queen Node

- Manages system, configurations, schema, and error handling
- Coordinates queries

Worker Node

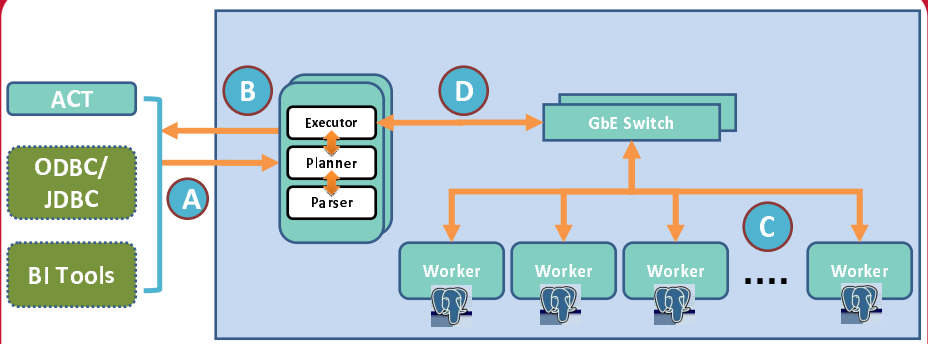
- Executes Queen's orders (query, incorporate, replication, re-partition)
- Stores partitioned data and interacts with Queen and other Workers

Loader Node

- Receive new data via ACT or Bulk Feeder
- Partition this data into appropriate segments
- Distribute the segmented data across Workers

ASTER

Query Processing: How It Works



A: Users send queries to the Queen (via ACT, ODBC/JDBC, BI tools, etc.)

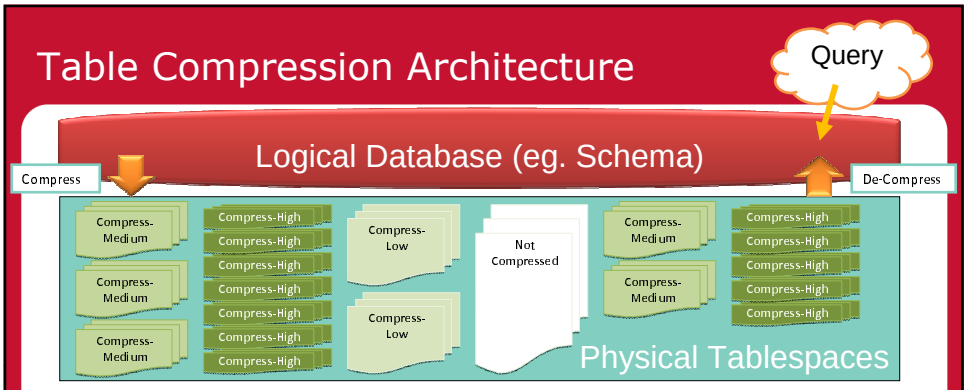
B: The Queen parses the query, creates an optimal distributed plan, sends subqueries to the Workers, and supervises the processing

C: Workers execute their subqueries (locally or via distributed communication)

D: The Queen aggregates Worker results and returns the final result to the user

ASTER

Table Compression Architecture



How It Works

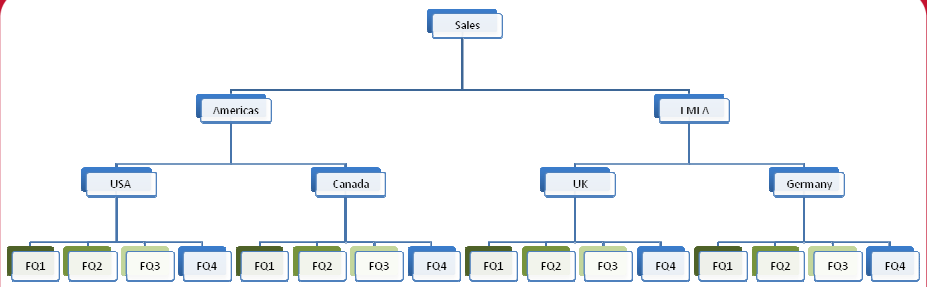
- Data is compressed & decompressed below logical database
- Stored in different tablespaces depending on compression level

Architecture Benefits:

- High compression ratios (bigger block sizes yield more cost savings)
- Database transparent (ease of future Postgres upgrades)
- Concurrency (high-performance multi-table compression)
- Performance: "Few-row" queries don't need full table decompression

ASTER

Table Compression Enables Powerful Archival



- Compressed - High
- Compressed - Medium
- Compressed - Low
- Not Compressed

Older data accessed less frequently

Compress to save space and cost

Oldest data is compressed the most, recent is compressed the least

Compressed tables are fully available for queries (true online archival)

ASTER

What is "MapReduce" and Why Should I Care?

What is MapReduce?

- Popularized by Google
  http://labs.google.com/papers/mapreduce.html
- Processes data in parallel across distributed cluster

Why is MapReduce significant?

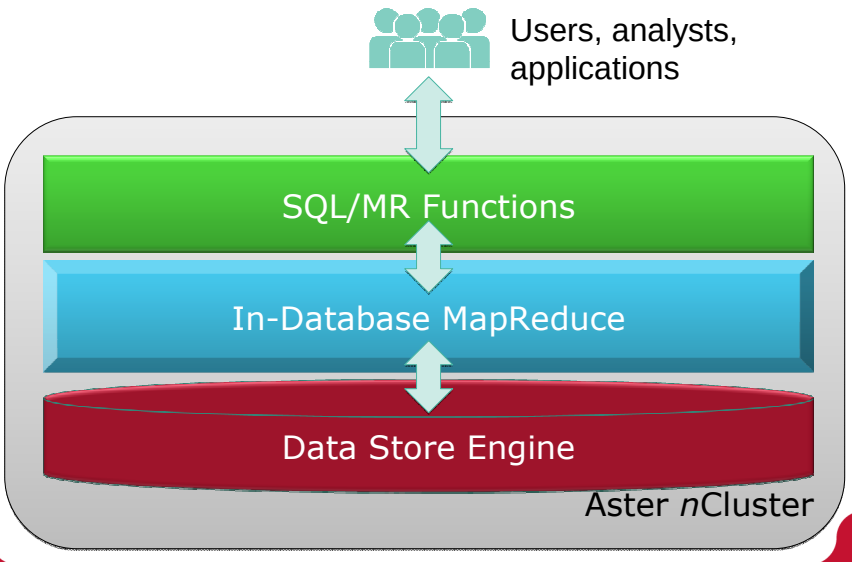
- Empowers ordinary developers
- Write application logic, not debug cluster communication code

Why is In-Database MapReduce significant?

- Unites MapReduce with SQL: power invoked from SQL
- Develop SQL/MR functions with common languages

ASTER

Aster In-Database MapReduce



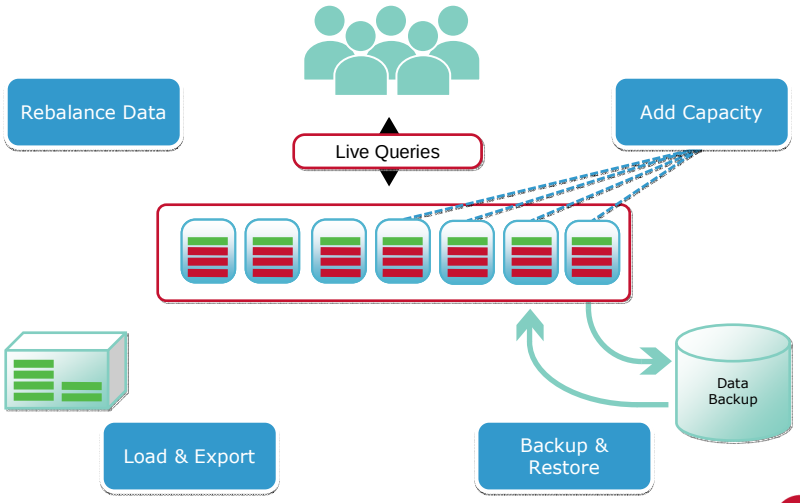
ASTER

In-Database MapReduce

- Extensible framework (MapReduce + SQL)
  - Flexible: Map-Reduce expressiveness, languages, polymorphism
  - Performance: Massively parallel, computational push-down
  - Availability: Fault isolation, resource management
- Out-of-process executables
  - Does not use PL/* for custom code execution
  - Can execute Map and Reduce functions in **any** language that has a runtime on Linux (e.g. Java, Python, C#, C/C++, Ruby, Perl, R, etc…)
  - Standard PostgreSQL APIs to send/receive data to executables
  - Fault isolation, security and resource management for arbitrary user code

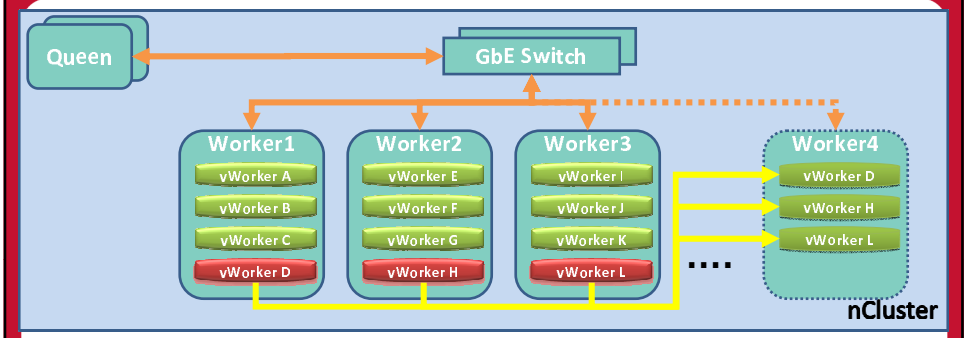
ASTER

Always On: Minimize Planned Downtime



ASTER

Precision Scaling



When more CPU/memory/capacity are needed, new nodes can be added for scale-out.

Precision Scaling **uses standard PostgreSQL APIs** to migrate vWorker partitions to new nodes either for load balancing (more compute power) or capacity expansion

Example: Assume Workers 1/2/3 are 100% CPU-bottlenecked. Incorporation adds a new Worker4 node and migrates over vWorker partitions D/H/L. As a result of load-balancing, CPU-utilization drops to 75% per node, eliminating hotspots.

ASTER

Replication

Replication provides distributed fault tolerance and system availability.

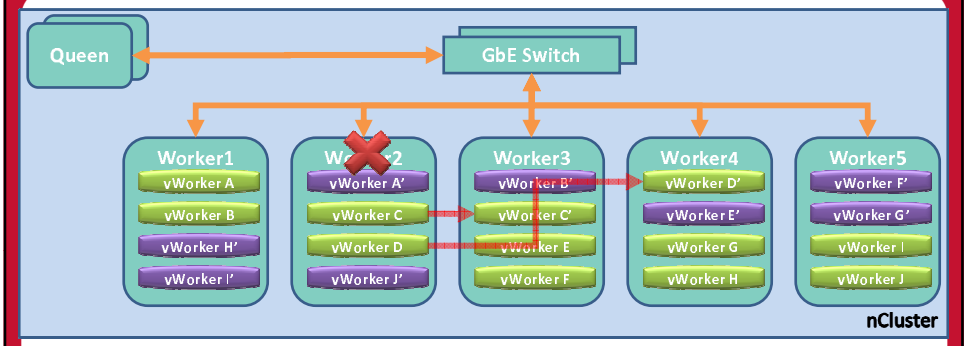
- Each Partition (vWorkers) is replicated at least once on another node
- In the event of node failure, replication protects against data loss and system outages

Treating PostgreSQL and Linux "as a service"

- Transparent to Postgres
- Mix of WAL and Data shipping
- Linux OS facilities to capture data changes

ASTER

Fault Tolerance & Automatic Online Failover



Replication Failover

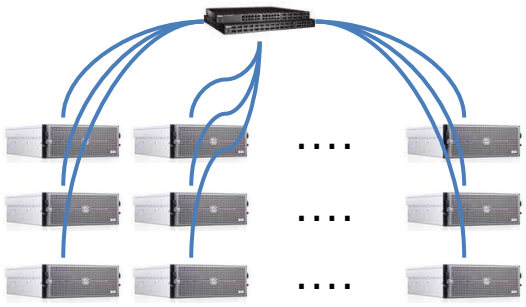
- Automatic, non-disruptive, graceful performance impact

Replication Restoration

- Delta-based (fast) and online (non-disruptive)

ASTER

Using Commodity Hardware



Commodity Network
1.Gigabit Ethernet (GbE)
2.Cisco, Juniper, etc.

Queen Nodes (Coordinate)

Worker Nodes (Query)

Loader Nodes (Transfer)

Commodity Servers
1.Multi-Core CPUs x Multiple CPUs per server
2.Distributed Memory (eg. 16GB RAM)
3.SAS/SATA Disk Drives (varying RPM, capacity)
4.Dell, HP, IBM, Sun, etc.

Key Benefits
1.Scalable Performance
2.Highly Manageable
3.Highly Available
4.Lowest TCO

ASTER

Scaling On-Demand to a PetaByte

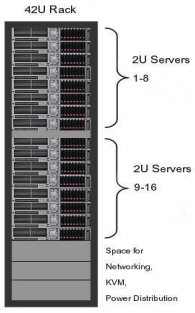
Commodity Hardware

2 TB Building Block
Dell, HP, IBM, Intel x86
16 GB Memory
2.4TB of Storage
- 8 Disks

$5k to $10k Node

***One n*Cluster Node**

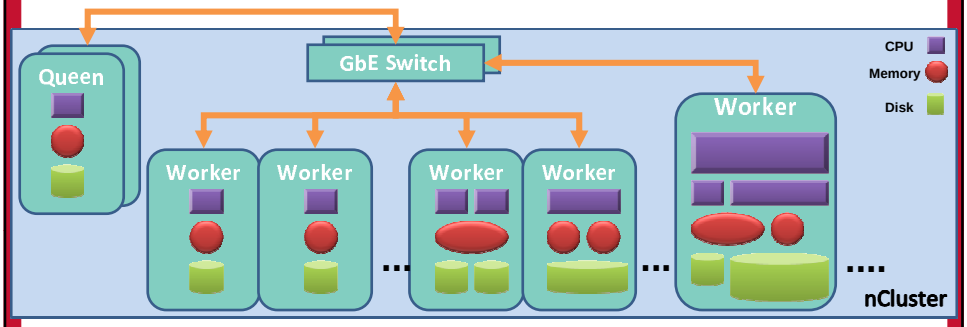
More Blocks = More Power



Massive Power Per Rack
- 160 Cores
- 640 GB RAM
- 48 TB SAS

ASTER

Heterogeneous Hardware Support



Heterogeneous HW support enables customers to add cheaper/faster nodes to existing systems for investment protection

Mix-n-match different servers as you grow (faster CPUs, more memory, bigger disk drives, different vendors, etc)

ASTER

Topics

Introduction to frontline data warehouses

PetaByte warehouse design with PostgreSQL

Aster contributions to PostgreSQL

Q&A

ASTER

Error Logging in COPY

- Separate good from malformed tuples
- Per tuple error information
- Errors to capture
  - Type mismatch (e.g. text vs. int)
  - Check constraints (e.g. int < 0)
  - Malformed chars (e.g. invalid UTF-8 seq.)
  - Missing / extra column
- Low-performance overhead
- Activated on-demand using environment variables

Error Logging in COPY

```
COPY tablename [ ( column [, ...] ) ]
    FROM STDIN
    [ [ WITH ]
          [ DELIMITER [ AS ] 'delimiter' ]
          [ NULL [ AS ] 'null string' ]
          [ CSV [ QUOTE [ AS ] 'quote' ]
                [ ESCAPE [ AS ] 'escape' ] ] ]
    [ LOG ERRORS
      [ [ INTO 'errortablename' ] [ WITH LABEL [ AS ] 'label' ] | NOWHERE ]
      [ ERRORLIMIT [ AS ] 'limit' ]
    ]
```

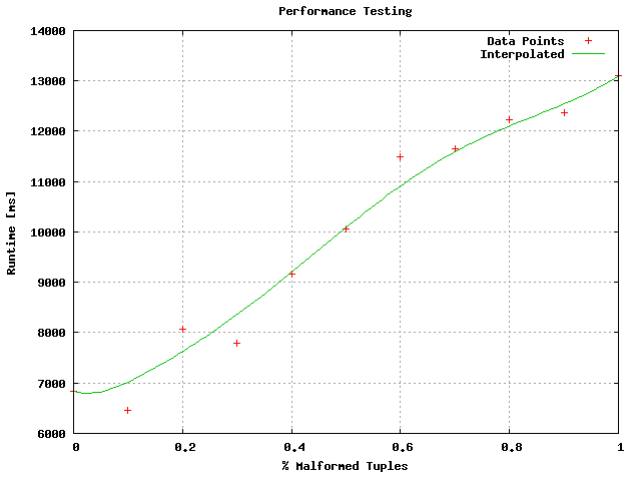
- Detailed error context is logged along with tuple content

tupletimestamp	targettable	errmessage	sqlerrcode
2009-05-18 15:37:32.423139-07	"public"."test"	invalid input syntax on partition key for type bigint: malformed 10	22000
2009-05-18 15:37:32.423208-07	"public"."test"	invalid input syntax on partition key for type bigint: malformed 85	22000
2009-05-18 15:37:32.423158-07	"public"."test"	invalid input syntax on partition key for type bigint: malformed 20	22000
2009-05-18 15:37:32.423175-07	"public"."test"	invalid input syntax on partition key for type bigint: malformed 22	22000
2009-05-18 15:37:32.409729-07	public.test	new row for relation "test" violates check constraint "test_a_check"	23514
2009-05-18 15:37:32.40988-07	public.test	new row for relation "test" violates check constraint "test_a_check"	23514
2009-05-18 15:37:32.409895-07	public.test	new row for relation "test" violates check constraint "test_a_check"	23514
2009-05-18 15:37:32.423241-07	"public"."test"	invalid input syntax on partition key for type bigint: 101 this is a test	22000
2009-05-18 15:37:32.423191-07	"public"."test"	invalid input syntax on partition key for type bigint: malformed 61	22000
2009-05-18 15:37:32.423225-07	"public"."test"	invalid input syntax on partition key for type bigint: malformed 96	22000
2009-05-18 15:37:32.423104-07	"public"."test"	invalid input syntax on partition key for type bigint: 2.1.1 1 byte (U-00000000): "0	22000

(11 rows)

Error Logging Performance

- 1 million tuples COPY performance



ASTER

Auto-partitioning in COPY

➲ COPY into a parent table route tuples directly in the child table with matching constraints

```
CREATE TABLE y2008 (
  id     int not null,
  date   date not null,
  value  int
);

CREATE TABLE jan2008 (
    CHECK ( date >= DATE '2008-01-01' AND date < DATE '2008-02-01' )
) INHERITS (y2008);

CREATE TABLE feb2008 (
    CHECK ( date >= DATE '2008-02-01' AND date < DATE '2008-03-01' )
) INHERITS (y2008);

CREATE TABLE mar2008 (
    CHECK ( date >= DATE '2008-03-01' AND date < DATE '2008-04-01' )
) INHERITS (y2008);
```

➲ COPY y2008 FROM 'data.txt'

```
SELECT * FROM y2008;
 id |    date    | value
----+------------+-------
 11 | 01-10-2008 |    11
 21 | 01-10-2008 |    11
 31 | 01-10-2008 |    11
 41 | 01-10-2008 |    11
 12 | 02-15-2008 |    12
 22 | 02-15-2008 |    12
 32 | 02-15-2008 |    12
 42 | 02-15-2008 |    12
 13 | 03-15-2008 |    13
 23 | 03-15-2008 |    13
 33 | 03-15-2008 |    13
 43 | 03-15-2008 |    13
(12 rows)

SELECT * FROM jan2008;
 id |    date    | value
----+------------+-------
 11 | 01-10-2008 |    11
 21 | 01-10-2008 |    11
 31 | 01-10-2008 |    11
 41 | 01-10-2008 |    11
(4 rows)

SELECT * FROM feb2008;
 id |    date    | value
----+------------+-------
 12 | 02-15-2008 |    12
 22 | 02-15-2008 |    12
 32 | 02-15-2008 |    12
 42 | 02-15-2008 |    12
(4 rows)

SELECT * FROM mar2008;
 id |    date    | value
----+------------+-------
 13 | 03-15-2008 |    13
 23 | 03-15-2008 |    13
 33 | 03-15-2008 |    13
 43 | 03-15-2008 |    13
(4 rows)
```


ASTER

Auto-partitioning in loading

- Activated on-demand
  - set tuple_routing_in_copy = 0/1;
- Configurable LRU cache size
  - set tuple_routing_cache_size = 3;
- COPY performance into parent table similar to direct COPY in child table if data is sorted
- Will leverage partitioning information in the future (WIP for 8.5)

Other contributions

- Temporary tables and 2PC transactions
- [Auto-]Partitioning infrastructure
- Regression test suite
- LFI (http://lfi.sourceforge.net/)
  - Fault injection at the library level or below
  - out-of-memory conditions, network connection errors, interrupted system calls, data corruption, hardware failures, etc…
  - Lightning talk on Friday!

Topics

Introduction to Aster and data warehousing

PetaByte warehouse design with PostgreSQL

Aster contributions to PostgreSQL

Q&A

PetaByte Warehouses with PostgreSQL

- **Unmodified** PostgreSQL
- "Always Parallel" MPP architecture
- "Always On" on-line operations
- In-Database MapReduce
- PetaByte on commodity Hardware

Aster Data Systems

Learn more

- www.asterdata.com
- Free TDWI report on advanced analytics:
  - asterdata.com/mapreduce
- Free Gartner webcast on mission-critical DW:
  - asterdata.com/gartner

Contact us

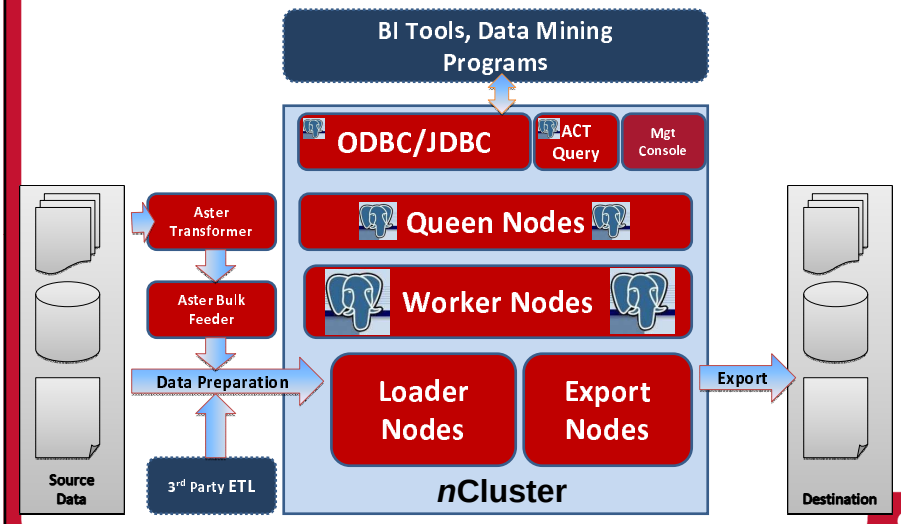
- hello@asterdata.com

ASTER

Bonus slides

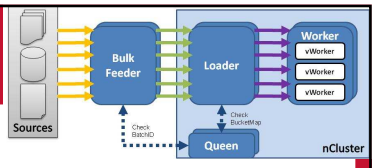
ASTER

*n*Cluster Components



ASTER

Aster Loader



- **Load Balancing**
  - **Challenge:** Large data sets create a data transfer performance bottleneck.
  - **Solution:** Multiple Loaders are mapped to vWorkers on Worker nodes. After partitioning, Loader will parallel load unique data into vWorkers.
  - **Benefit:** Linearly scalable load performance (from Loaders to Workers)

Scalable Partitioning

- **Challenge:** Partitioning must be scalable and not impede the loading process
- **Solution:** Each Loader contains a "Partitioner", which uses an algorithm to assign data into "buckets". Each bucket is uniquely mapped to within the cluster.
- **Benefit:** Fast, intelligent partitioning during massive-scale data loads

- **Fault Handling**
  - **Challenge:** Failed nodes may lose data or significantly drop load performance.
  - **Solution:** If a node fails, other streams continue to load (1/N performance hit). Admins can easily recover lost data by re-loading Bulk Feeder data.
  - **Benefit:** Data loss protection & performance consistency during node failure.

ASTER