Data Transformation and Migration in Polystores

Adam Dziedzic, Aaron Elmore & Michael Stonebraker

THE UNIVERSITY OF
CHICAGO

Database Group
MIT Computer Science and Artificial Intelligence Lab

September 15th, 2016
Agenda

- **Data Migration for Polystores:**
  - What & Why?
  - How?
- **Acceleration of physical data migration via:**
  - Data formats and transformations
  - Resource-awareness
  - Parallelism and compression
  - Adaptivity
- **Conclusion:** **Fast Data Migrator**
Polystore: ”One size does not fit all”

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Adam</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Aaron</td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>Mike</td>
<td></td>
</tr>
</tbody>
</table>

Metadata

PostgreSQL
Polystore: "One size does not fit all"

<table>
<thead>
<tr>
<th></th>
<th>Adam</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>34</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Aaron Elmore</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Adam Dziedzic</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Mike Stonebraker</th>
</tr>
</thead>
<tbody>
<tr>
<td>34</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Polystore: "One size does not fit all"
Polystore: "One size does not fit all"

- **Metadata**: PostgreSQL
- **Text**: Accumulo
- **Scientific data**: SciDB
- **Streams of data**: S-Store

<table>
<thead>
<tr>
<th>1</th>
<th>Adam</th>
<th>…</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>Aaron Elmore</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Adam Dziedzic</td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>Mike Stonebraker</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>12</th>
<th>Mike</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>45</td>
</tr>
</tbody>
</table>
Polystore: "One size does not fit all"

Metadata
PostgreSQL

Text
Accumulo

Scientific data
SciDB

Streams of data
S-Store

Polystore couples diverse data models
Data Migration in Polystores: **TWO WAYS**

- **Short-term** for partial results of queries

  - MapReduce
  - Result

  →

  - RDBMS
  - TABLE

- **Long-term** for evolving workload and load-balancing

  - Node 1
  - Node 2
  - Node 3
## Data Migration: current approach vs. our methods

<table>
<thead>
<tr>
<th>METHOD</th>
<th>TIME (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>From PostgreSQL to SciDB (MIMIC II data, 10 GB)</strong></td>
<td></td>
</tr>
<tr>
<td>CSV (common approach)</td>
<td>772</td>
</tr>
<tr>
<td><strong>From S-Store to SciDB (TPC-C data, 10 GB)</strong></td>
<td></td>
</tr>
<tr>
<td>CSV (common approach)</td>
<td>823</td>
</tr>
</tbody>
</table>
Data Migration: current approach vs. our methods

<table>
<thead>
<tr>
<th>METHOD</th>
<th>TIME (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>From PostgreSQL to SciDB</strong> <em>(MIMIC II data, 10 GB)</em></td>
<td></td>
</tr>
<tr>
<td>CSV (common approach)</td>
<td>772</td>
</tr>
<tr>
<td>Direct parallel binary migration with compression</td>
<td>75</td>
</tr>
<tr>
<td><strong>From S-Store to SciDB</strong> <em>(TPC-C data, 10 GB)</em></td>
<td></td>
</tr>
<tr>
<td>CSV (common approach)</td>
<td>823</td>
</tr>
<tr>
<td>Parallel (16 X) direct binary migration</td>
<td>100</td>
</tr>
</tbody>
</table>
Agenda

- Data Migration for Polystores:
  - What & Why?
  - How?
- Acceleration of physical data migration via:
  - Data formats and transformations
  - Resource-awareness
  - Parallelism and compression
  - Adaptivity
- Conclusion: **Fast Data Migrator**
Data Migrator Pipeline

DBMS X

Extract
- Read metadata
- Extract data

DBMS Y
Data Migrator Pipeline

Extract
- Read metadata
- Extract data

Transform & Migrate
- Logical transformation
- Format conversion
- Compression
- Local / remote

DBMS X → Extract → Transform & Migrate → DBMS Y
Data Migrator Pipeline

**Extract**
- Read metadata
- Extract data

**Transform & Migrate**
- Logical transformation
- Format conversion
- Compression
- Local / remote

**Load**
- Write metadata
- Load data

DBMS X → Extract → Transform & Migrate → Load → DBMS Y
Data Migrator Pipeline

**Extract**
- Read metadata
- Extract data

**Transform & Migrate**
- Logical transformation
- Format conversion
- Compression
- Local / remote

**Load**
- Write metadata
- Load data

**No disk materialization**
Agenda

- Data Migration Framework for Polystores:
  - Why?
  - How?

- Acceleration of physical data migration via:
  - Data formats and transformations
  - Parallelism
  - Adaptivity
  - Resource-awareness

- Conclusion: Fast Data Migrator
Current approach: CSV migration

CSV format
1,"Adam",6.00; 2,"Aaron",7.00
Current approach: CSV migration

CSV format
1,”Adam”,6.00; 2,”Aaron”,7.00

Data already loaded to the source database
Our approach: binary migration

Binary format X -> 1001

TRANSFORM

1001 -> 0110

Binary format Y

SINGLE Binary format: Y

0010101

DBMS X -> 0110

DBMS Y
Data Migration from PostgreSQL to SciDB

TRANSFORMATION is 3X, DIRECT is 4X faster than CSV migration

MIMIC II data - waveform(int, int, double)
Breakdown: migration from PostgreSQL to SciDB

MIMIC II waveform data (int, int, double) 10 GB

Migration Time (sec)

- CSV

Slow CSV loading
Breakdown: migration from PostgreSQL to SciDB

Migration Time (sec)

- **csv**
- **transform binary**

- **export PSQL**
- **load SciDB**
- **export native PSQL**
- **transform**
- **load SciDB**

MIMIC II waveform data (int, int, double) 10 GB

Binary Export SLOWER than Binary Loading
Breakdown: migration from PostgreSQL to SciDB

MIMIC II waveform data (int, int, double) 10 GB

Migration Time (sec)

- **export PSQL**
- **load SciDB**
- **export native PSQL**
- **transform**
- **load SciDB**
- **export SciDB PSQL**
- **load SciDB**

- **csv**
- **transform binary**
- **direct binary**

**Fast Direct Binary Migration**
Agenda

- Data Migration for Polystores:
  - What & Why?
  - How?

- Acceleration of physical data migration via:
  - Data formats and transformations
  - Resource-awareness
  - Parallelism and compression
  - Adaptivity

- Conclusion: Fast Data Migrator
Resource usage: CSV waveform data loading to SciDB

- Throughput (MB/sec)
- Usage (%)
- Time (sec)
- CPU (%)
- RAM (%)
- Disk Write (MB/sec)
- Network Received (MB/sec)

The chart shows the resource usage over time during the process of CSV waveform data loading to SciDB. The x-axis represents time in seconds, with key stages labeled as 'Load' and 'Redimension.' The y-axis indicates usage percentages for CPU and RAM, as well as throughput and disk write speeds. The network received data also peaks at certain times, showing fluctuations in resource utilization.
Resource usage: CSV waveform data loading to SciDB

- **CPU (%)**
- **RAM (%)**
- **Disk Write (MB/sec)**
- **Network Received (MB/sec)**

Compress/Decompress to utilize spare CPU cycles
Resource usage: CSV waveform data loading to SciDB

What is an optimal degree of parallelism?
Agenda

- Data Migration for Polystores:
  - What & Why?
  - How?
- Acceleration of physical data migration via:
  - Data formats and transformations
  - Resource-awareness
  - **Parallelism and compression**
  - Adaptivity
- Conclusion: **Fast Data Migrator**
Data Migration from S-Store to PostgreSQL & SciDB

- Enhanced data export from S-Store
  - Binary PostgreSQL
  - Binary SciDB
- Parallel export via partitioning
Data Migration from S-Store to PostgreSQL & SciDB

Time for CSV and binary migration converges for high degree of parallelism
Design of Parallel Export from PostgreSQL: We CARE

Current single-thread export

New parallel export
Parallel export from PostgreSQL

TPC-H benchmark, 10 GB, CSV format

CURRENT EXPORT

Export Time (sec)

0 50 100 150 200 250 300 350

1

Degree of parallelism

1 2 4 8 16 32 64 128
Parallel export from PostgreSQL

*TPC-H benchmark, 10 GB, CSV format*

New Parallel export 20X faster than Current export
Single-threaded vs. Parallel Export from PostgreSQL

Single-thread export
(1 reader from disk)

4-thread export
(4 readers from disk)
COMPRESSSION for direct binary parallel migration

From PostgreSQL to SciDB, 4 threads, waveform data (int,int,double), 10 GB

Migration time (sec)

RAW DATA

raw
COMPRESSION for direct binary parallel migration

From PostgreSQL to SciDB, 4 threads, waveform data (int,int,double), 10 GB

Migration time (sec)

RAW DATA

COMPRESSED DATA

Lightweight compression for data transfer via network
COMPRESSSION for direct binary parallel migration

From PostgreSQL to SciDB, 4 threads, waveform data (int,int,double), 10 GB

Migration time (sec)

RAW DATA

COMPRESSED DATA

Load SciDB bottleneck

Lightweight compression for data transfer via network
Agenda

- Data Migration for Polystores:
  - What & Why?
  - How?
- Acceleration of physical data migration via:
  - Data formats and transformations
  - Resource-awareness
  - Parallelism and compression
  - **Adaptivity**
- Conclusion: **Fast Data Migrator**
Data Migration from PostgreSQL to Accumulo

Migration Time (sec)

BatchWriter | MapReduce

Data size (GB)

0.0 0.1 0.2 0.3

0 100 200 300
Data Migration from PostgreSQL to Accumulo

Adaptive data loading method

Migration Time (sec)

Data size (GB)

BatchWriter
MapReduce
Problem

**EFFICIENT** data migrator between diverse database systems Indispensible component in Polystores.

Solution

*Apply:* Binary format, Parallelism, Compression & Adaptivity

*Be:* Resource-Aware & Hardware-Efficient

Result

**FAST Data Migration** between:
PostgreSQL, SciDB, S-Store & Accumulo
Thank you
Backup slides
Polystores require EFFICIENT data migrator

”multistore fail to achieve the full potential b/c high cost of data movement and loading”

MISO paper, SIGMOD 2014

”Optimizing Database Load and Extract for Big Data Era – this bottleneck led to ETL.”

DASFAA 2014

Complex analytics and many more database management systems require data migration!
Why binary despite parallel CSV migration?

- Binary migration for high degree of parallelism (e.g. 16) is still about 44% faster than CSV migration (from S-Store to SciDB)
- Cannot allocate all the cores to the migration process
- CSV migration incurs greater energy consumption
- It is not always feasible to divide the CSV data (evenly) into chunks / partitions (e.g. due to skew in the data)
- There can be fewer partitions (in S-Store) than physical cores & many servers operate with 4 to 8 cores
Data Migration in Polystores: **TWO WAYS**

- **Short-term** for partial results of queries
  - MapReduce → Result

- **Long-term** for evolving workload and load-balancing
  - Node 1 → Node 2 → Node 3 → RDBMS → TABLE
# Data migration from PostgreSQL to SciDB

*TPC-H benchmark, 10 GB*

<table>
<thead>
<tr>
<th>METHOD</th>
<th>TIME (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDBC</td>
<td>1000</td>
</tr>
<tr>
<td>CSV</td>
<td>800</td>
</tr>
<tr>
<td>Binary format with transformation</td>
<td>270</td>
</tr>
<tr>
<td>Direct binary format</td>
<td>180</td>
</tr>
<tr>
<td><em>Parallel direct binary format</em></td>
<td>90</td>
</tr>
<tr>
<td><em>Parallel direct database native storage</em></td>
<td>62</td>
</tr>
<tr>
<td><em>GPU parallel direct database native storage</em></td>
<td>40</td>
</tr>
</tbody>
</table>
Future directions for Data Migration Framework

- Monitor usage of resources (rate limit) & select migration approach
- Apply compression, select # cores for parallel loading, utilize hardware

- Extract
  - Read metadata
  - Extract data

- Transform
  - Logical and binary conversion

- Migrate
  - Adapt to rate and data size

- Load
  - Write metadata
  - Load data

Parallelism  | Compression  | Rate Limit  | SIMD, GPU, RDMA
Compression in PostgreSQL backup utilities

TPC-H benchmark, 10 GB, CSV format

Dump & Restore Time (sec)

Compression Ratio

RAW DATA
Compression in PostgreSQL backup utilities

*TPC-H benchmark, 10 GB, CSV format*

**Dump & Restore Time (sec)**

<table>
<thead>
<tr>
<th>Compression Ratio</th>
<th>Dump &amp; Restore Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>RAW DATA</td>
</tr>
<tr>
<td>9</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
</tr>
</tbody>
</table>

**Speed-up migration and decrease data size 10X**
PostgreSQL backup utilities: compression ratio
2 types of CSV loading to SciDB

MIMIC II waveform data (int, int, double) 10 GB

- Blue: split (1 thread)
- Yellow: from CSV to SciDB format
- Red: load to flat array

The split phase is very slow!
Experimental setup for MIMIC-II data

- **Software:**
  - PostgreSQL 9.4.5 (built with -O2 optimization)
  - SciDB 14.12 (installed on a single node, 4 instances)

- **Hardware:**
  - Single node (Accumulo deployed on a cluster of 5 nodes)
  - Quad Core CPU with frequency of 3.1 GHz
  - 16 GB of main memory
  - 250 GB SSD (reads: 517 MB/sec, writes: 267 MB/sec)

- **Data:** waveform data (int, int, double), 10 GB
  - Dimensions: [int, int], attribute: [double]
Experimental setup for S-Store

- **Software:**
  - PostgreSQL 9.4.5 (built with -O2 optimization)
  - SciDB 14.12 (installed on a single node, 4 instances)
  - S-Store (latest version from github)

- **Hardware:**
  - Single node
  - Xeon Server E7-4800 32 cores with frequency of 2.4 GHz
  - 256 GB of main memory
  - RAID-0 20 disks (reads: 1 GB/sec, writes: 420 MB/sec)

- **Data:** TPC-C, YCSB
BigDAWG: Data migration from PostgreSQL to SciDB

flat (to a flat array), full - with redimension, MIMIC II data - 10 GB waveform (int, int, double)

Flat bin migration 3X faster than csv, redimension nullifies the difference
BigDAWG: Data migration from SciDB to PostgreSQL

flat (from flat array) full (from multi-dim. array) MIMIC II data - 10 GB waveform (int, int, double)

Flat bin migration 3X faster than csv, no binary migration from full array
Future work

- Use MPI (Message Passing Interface) to fully leverage different network fabrics
- Integrate with Spark by implementing the Data Source API
- Extend the supported binary formats: Parquet, Vertica, ...
- Introduce intermediate transformations during migration and semi-automatic migration
- Add adaptive encoding / compression / encryption
- Bottom line: migration between internal binary formats (in which data is stored natively in databases)
- Use recent hardware (SIMD, RDMA, UAP) & JIT compilation
Distributed Data Migrator

- Initial version works for:
  - PostgreSQL <-> PostgreSQL
  - PostgreSQL <-> SciDB
  - SciDB <-> PostgreSQL

- Implementation:
  - Requires BigDAWG on each node of the system
  - Send messages using ZeroMQ
  - One master which handles all the requests
  - Master distributes a migration task and waits for the result (RPC pattern)
SciDB opaque format for multi-dimensional array

<table>
<thead>
<tr>
<th>Operation</th>
<th>Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>load flat bin</td>
<td>131</td>
</tr>
<tr>
<td>export flat bin</td>
<td>134</td>
</tr>
<tr>
<td>opaque load flat</td>
<td>48</td>
</tr>
<tr>
<td>opaque save flat</td>
<td>86</td>
</tr>
<tr>
<td>load full bin</td>
<td>1,200</td>
</tr>
<tr>
<td>opaque load full</td>
<td>304</td>
</tr>
<tr>
<td>opaque save full</td>
<td>52</td>
</tr>
</tbody>
</table>
Single-threaded vs. Parallel Export from PostgreSQL

Better utilization of read bandwidth => better utilization of CPU
## Polystore system vs. Federated database

<table>
<thead>
<tr>
<th>Item</th>
<th>Polystore system</th>
<th>Federated database</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data models</td>
<td>Very diverse</td>
<td>Mainly relational</td>
</tr>
<tr>
<td>Control</td>
<td>One admin</td>
<td>Many admins</td>
</tr>
<tr>
<td>Placement</td>
<td>Collocated (one rack/datacenter)</td>
<td>Geographically decentralized</td>
</tr>
<tr>
<td>Components</td>
<td>Tightly coupled</td>
<td>Loosely connected</td>
</tr>
<tr>
<td>Concept</td>
<td>Data virtualization</td>
<td>Data federation</td>
</tr>
</tbody>
</table>
Data Migration in Polystores: **TWO WAYS**

- **Short-term** for partial results of queries
- **Long-term** for evolving workload and load-balancing