Data Transformation and Migration in Polystores

Adam Dziedzic, Aaron Elmore & Michael Stonebraker

THE UNIVERSITY OF CHICAGO

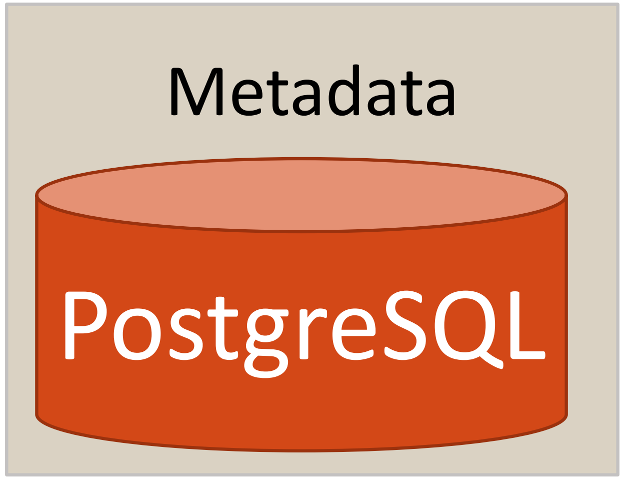
DBg Database Group
MIT Computer Science and Artificial Intelligence Lab

September 15th, 2016

Agenda

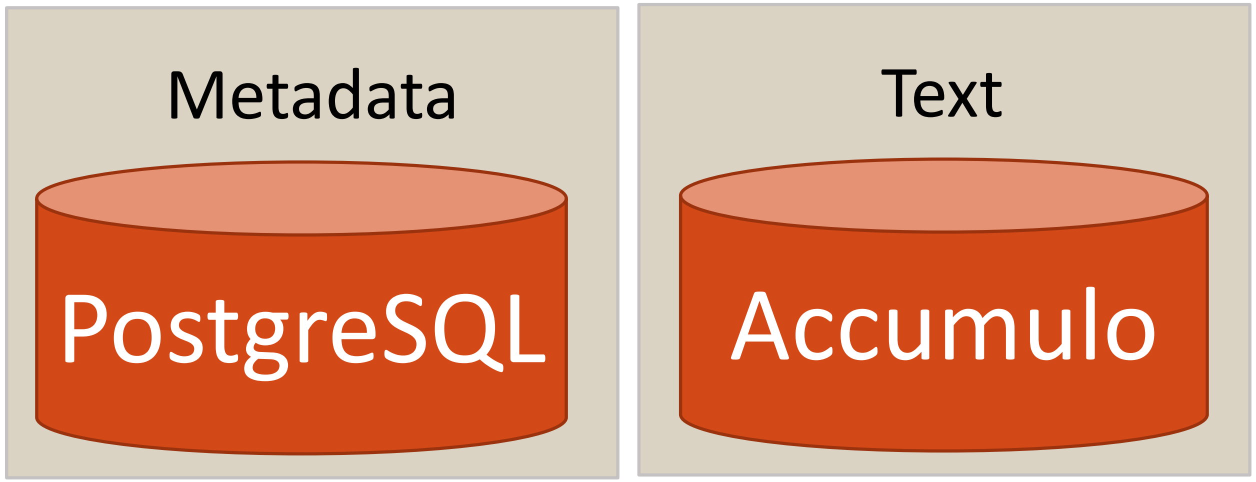
- **Data Migration for Polystores**:
  - What & Why?
  - How?
- **Acceleration of physical data migration** via:
  - Data formats and transformations
  - Resource-awareness
  - Parallelism and compression
  - Adaptivity
- Conclusion: **Fast Data Migrator**

Polystore: ”One size does not fit all”



1	Adam	…
12	Aaron	…
34	Mike	…

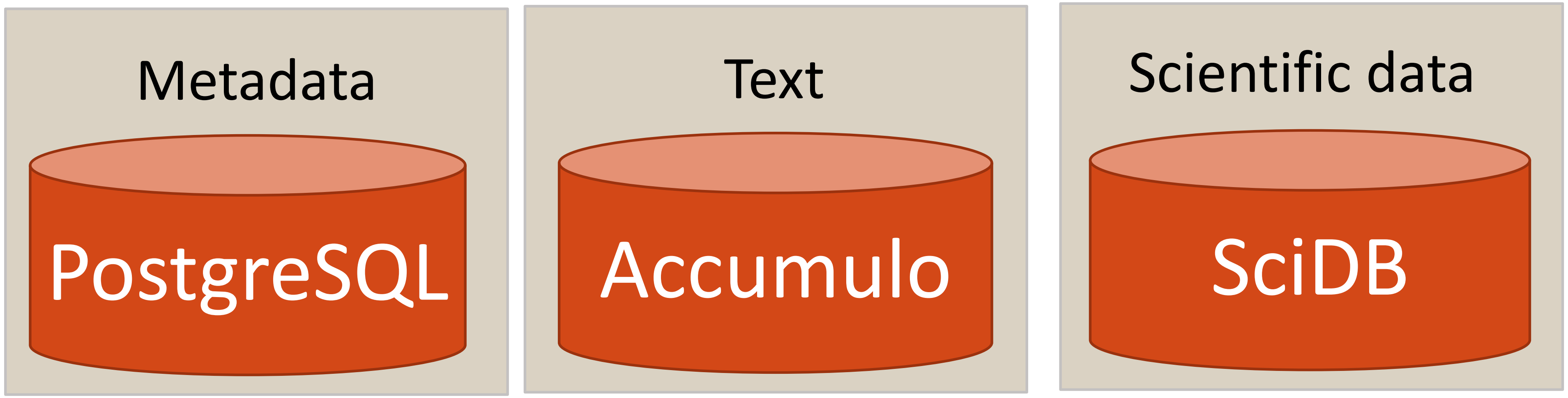
Polystore: ”One size does not fit all”



1	Adam	…
12	Aaron	…
34	Mike	…

12	Aaron Elmore
1	Adam Dziedzic
34	Mike Stonebraker
…	

Polystore: ”One size does not fit all”



Metadata

PostgreSQL

Text

Accumulo

Scientific data

SciDB

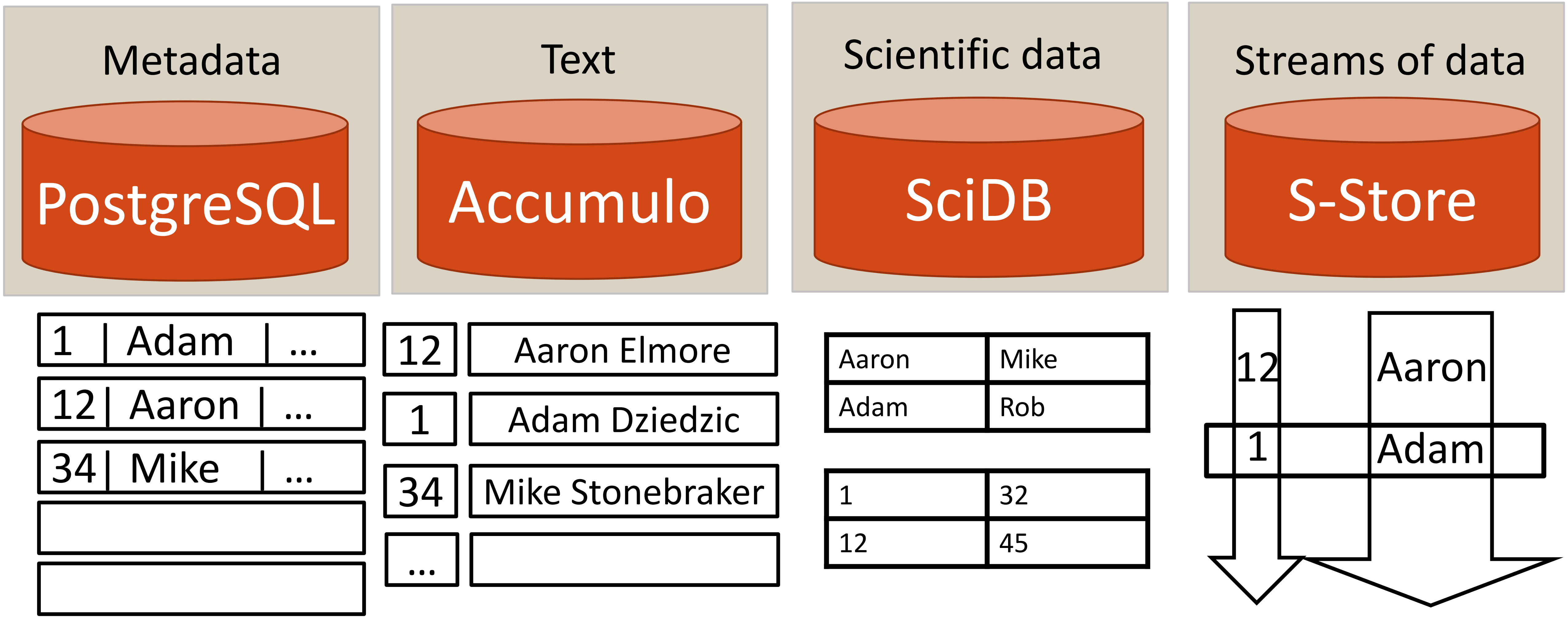
1	Adam	…
12	Aaron	…
34	Mike	…

12	Aaron Elmore
1	Adam Dziedzic
34	Mike Stonebraker
…	

Aaron	Mike
Adam	Rob

1	32
12	45

Polystore: ”One size does not fit all”



Metadata

PostgreSQL

1 | Adam | …

12 | Aaron | …

34 | Mike | …

Text

Accumulo

12	Aaron Elmore
1	Adam Dziedzic
34	Mike Stonebraker
…	

Scientific data

SciDB

Aaron	Mike
Adam	Rob

1	32
12	45

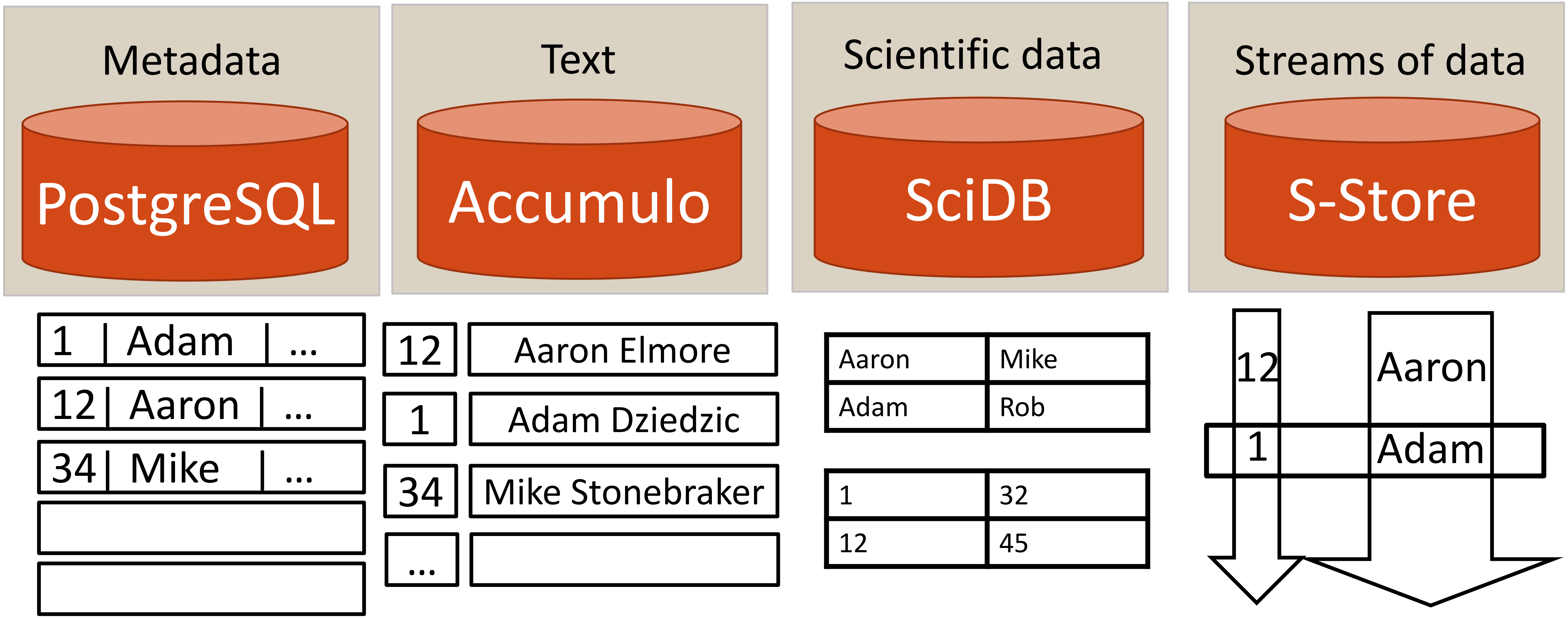
Streams of data

S-Store

12 Aaron

1 Adam

Polystore: ”One size does not fit all”



Metadata

PostgreSQL

1	Adam	…
12	Aaron	…
34	Mike	…

Text

Accumulo

12	Aaron Elmore
1	Adam Dziedzic
34	Mike Stonebraker
…	

Scientific data

SciDB

Aaron	Mike
Adam	Rob

1	32
12	45

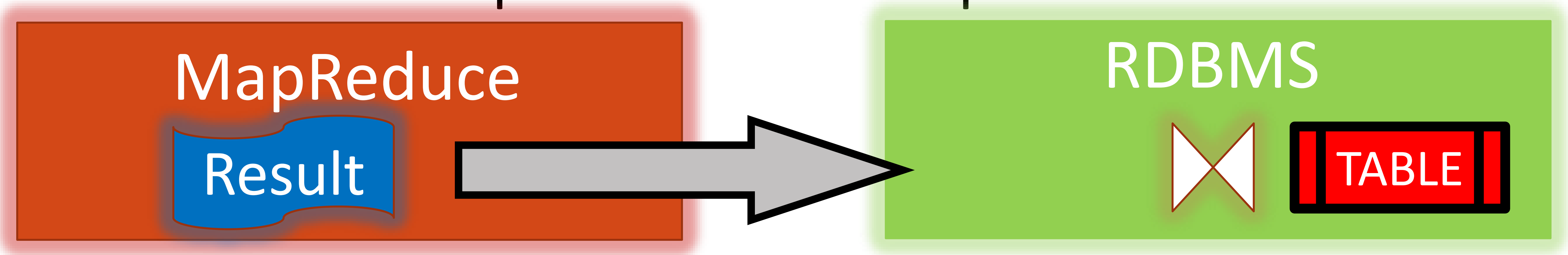
Streams of data

S-Store

Polystore couples diverse data models

Data Migration in Polystores: TWO WAYS

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



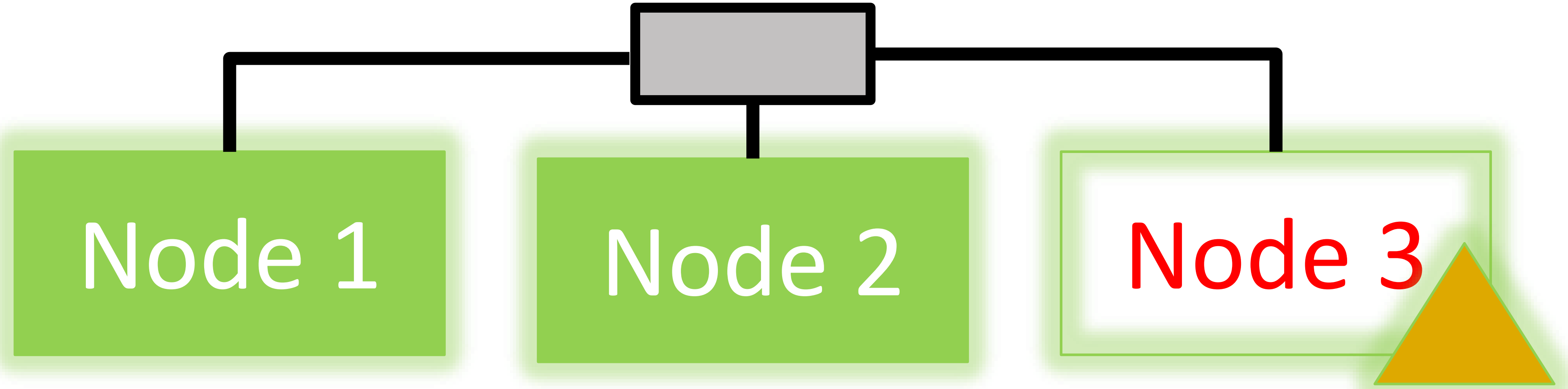
MapReduce

Result

RDBMS

TABLE

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



Node 1

Node 2

Node 3

Data Migration: current approach vs. our methods

METHOD	TIME (sec)
From PostgreSQL to SciDB *(MIMIC II data, 10 GB)*	
CSV (common approach)	772
From S-Store to SciDB *(TPC-C data, 10 GB)*	
CSV (common approach)	823

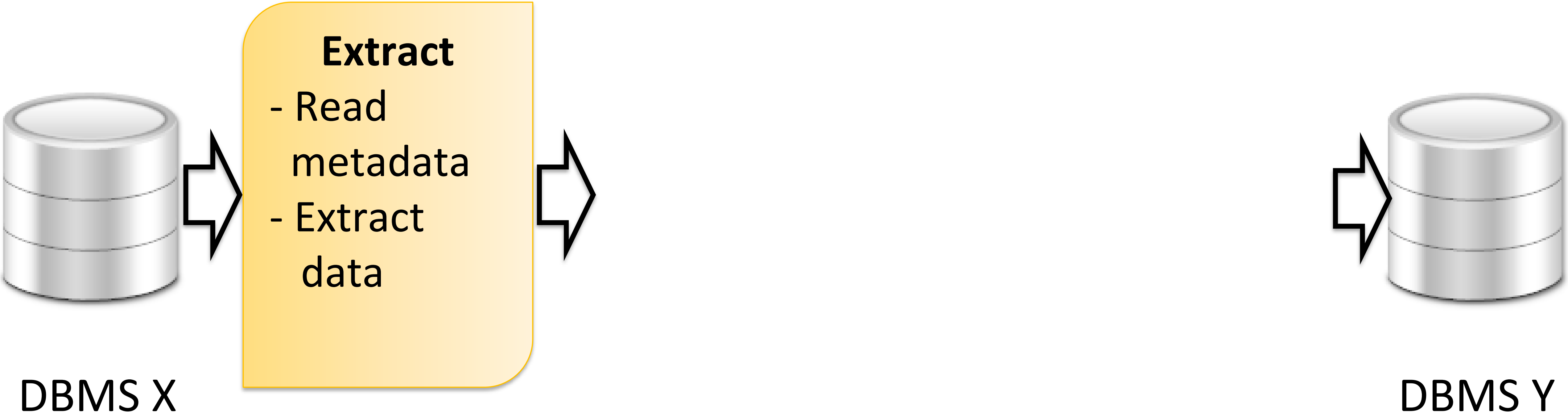
Data Migration: current approach vs. our methods

METHOD	TIME (sec)
From PostgreSQL to SciDB *(MIMIC II data, 10 GB)*	
CSV (common approach)	772
Direct parallel binary migration with compression	**75**
From S-Store to SciDB *(TPC-C data, 10 GB)*	
CSV (common approach)	823
Parallel (16 X) direct binary migration	**100**

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



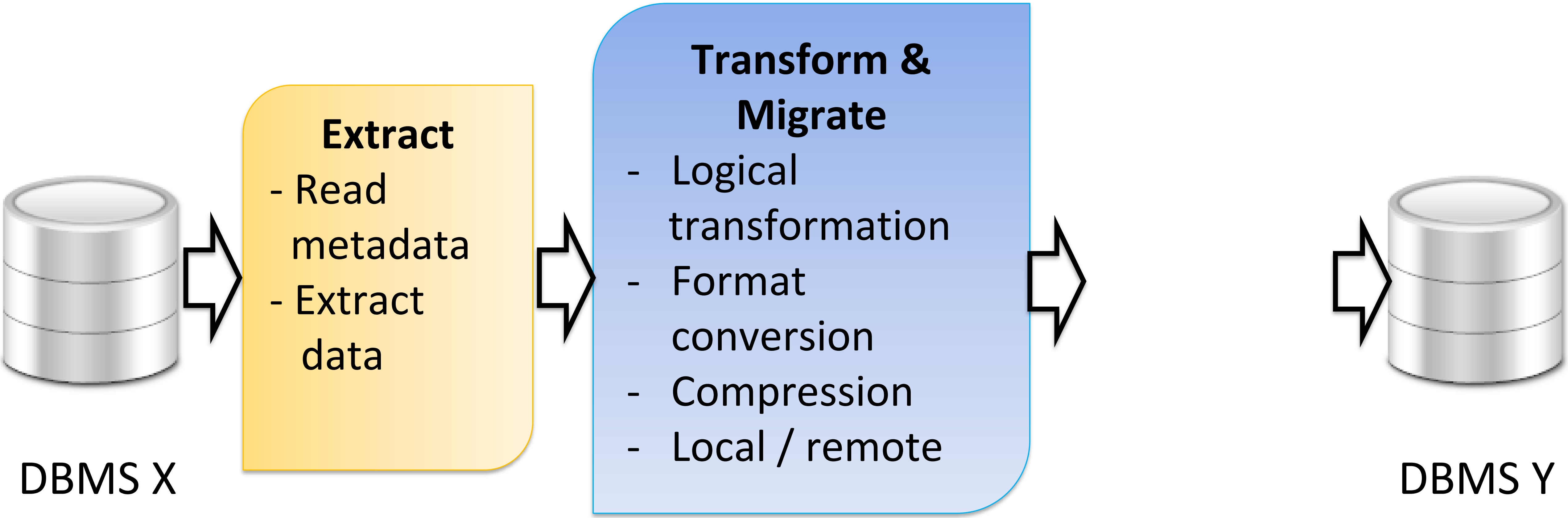
Extract

- Read metadata
- Extract data

DBMS X

DBMS Y

Data Migrator Pipeline



DBMS X

Extract

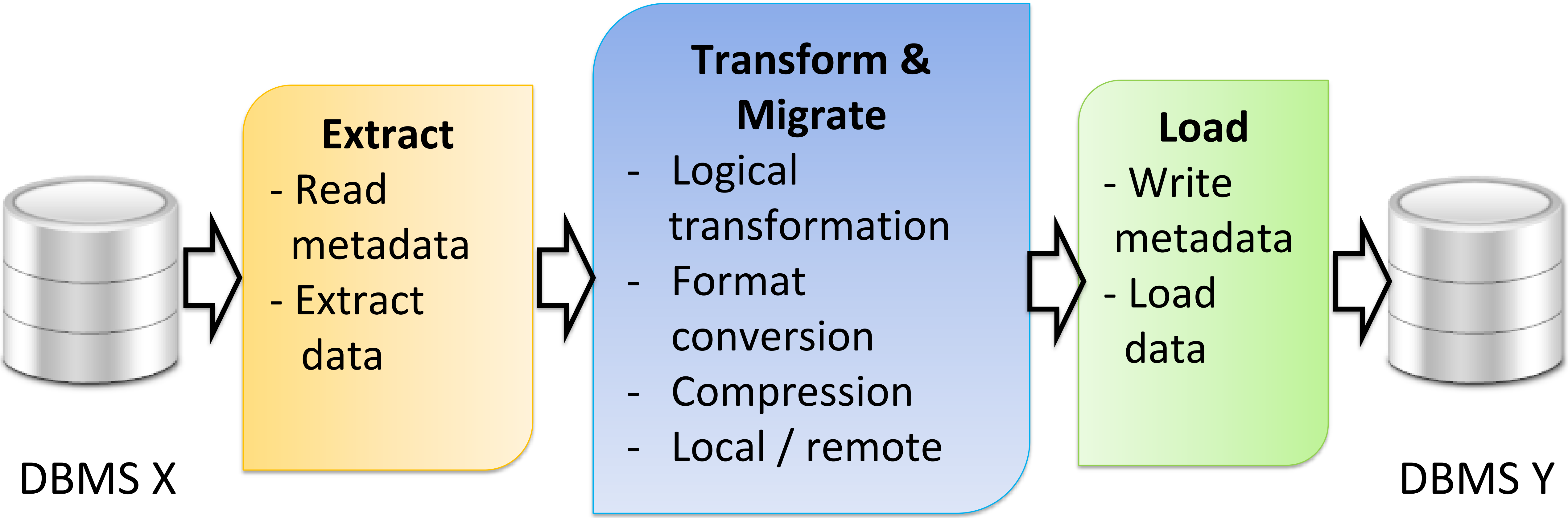
- Read metadata
- Extract data

Transform & Migrate

- Logical transformation
- Format conversion
- Compression
- Local / remote

DBMS Y

Data Migrator Pipeline



DBMS X

Extract

- Read metadata
- Extract data

Transform & Migrate

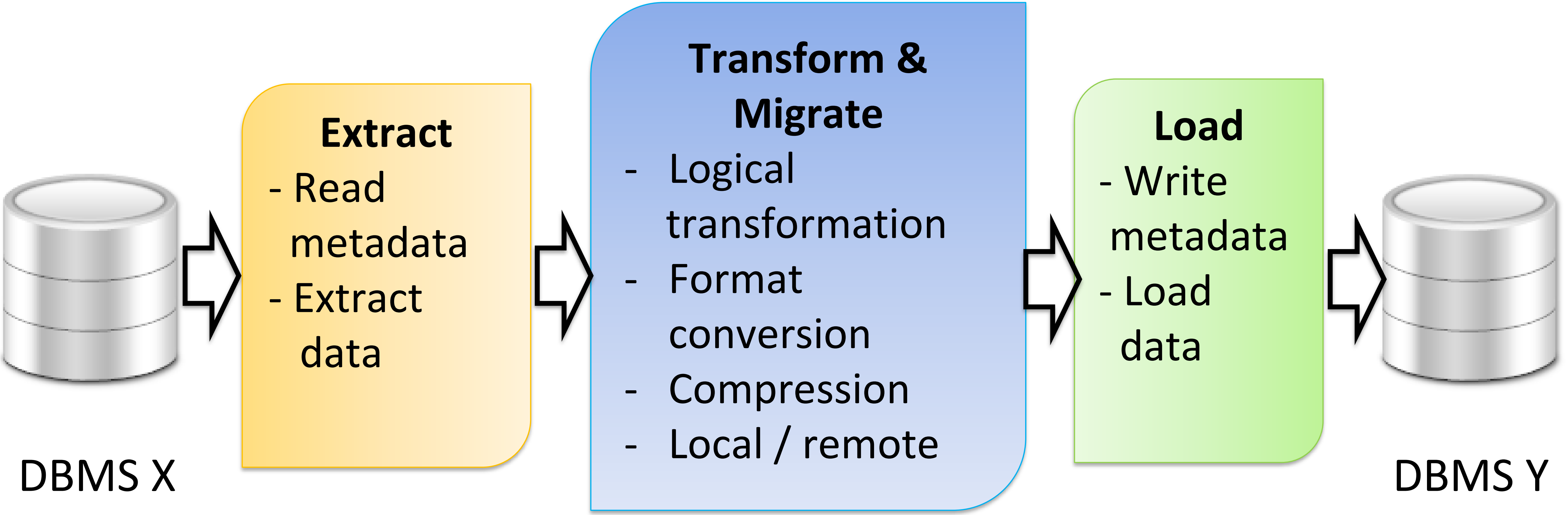
- Logical transformation
- Format conversion
- Compression
- Local / remote

Load

- Write metadata
- Load data

DBMS Y

Data Migrator Pipeline



DBMS X

Extract

- Read metadata
- Extract data

Transform & Migrate

- Logical transformation
- Format conversion
- Compression
- Local / remote

Load

- Write metadata
- Load data

DBMS Y

No disk materialization

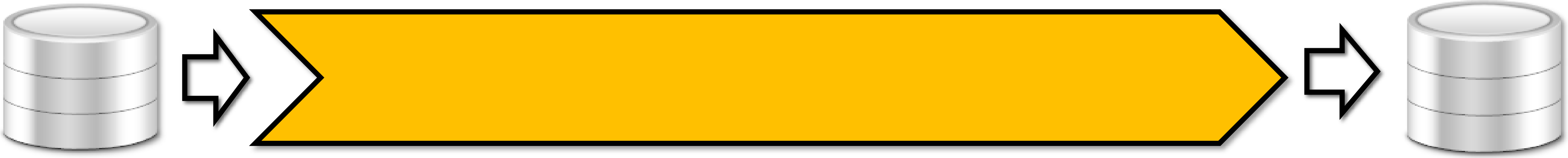
Agenda

- ❑ 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

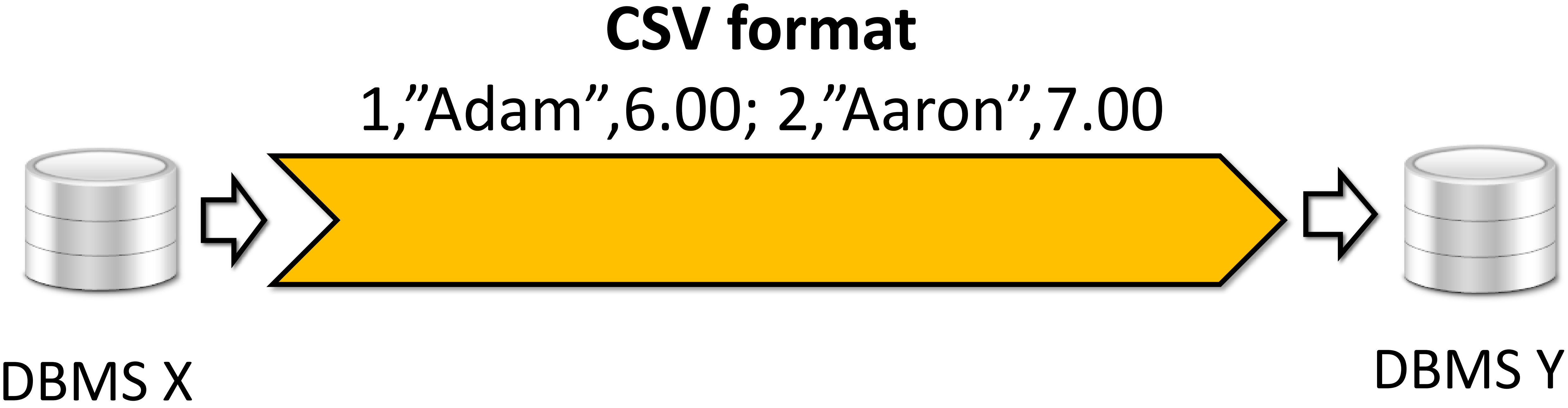
1,”Adam”,6.00; 2,”Aaron”,7.00



DBMS X

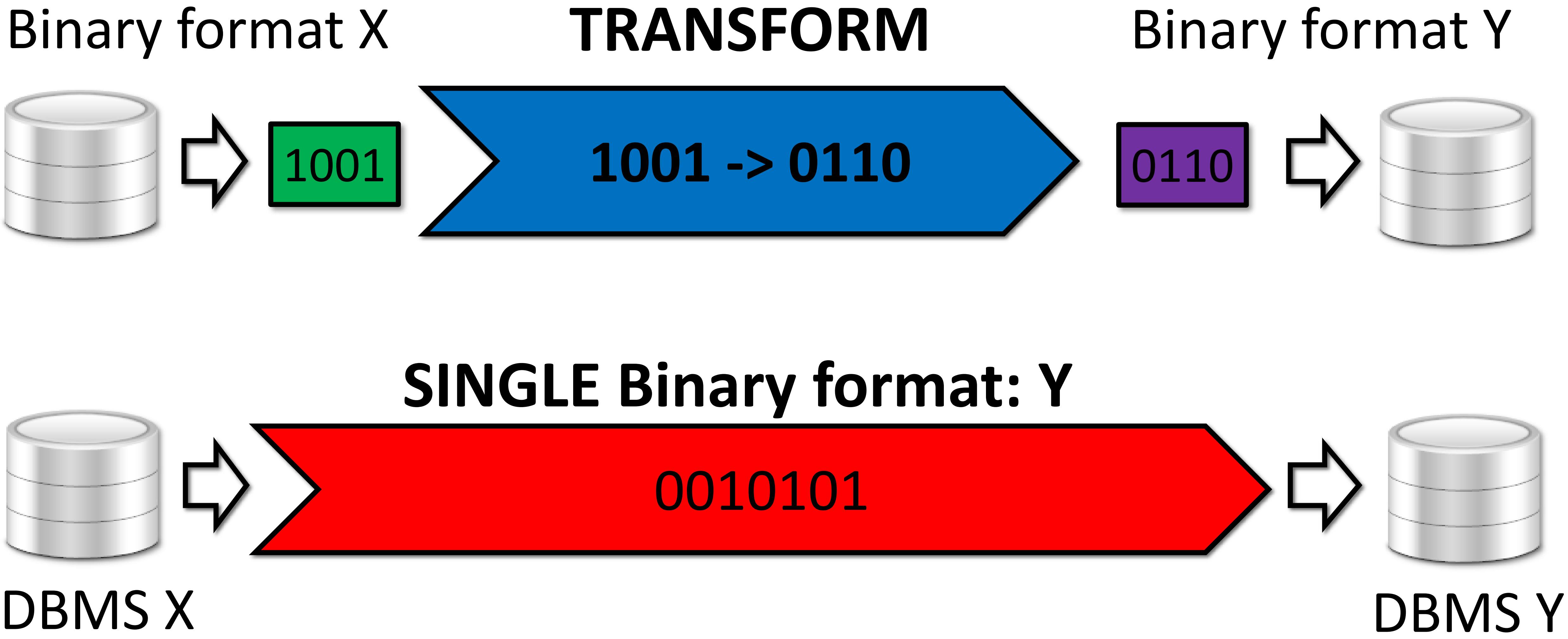
DBMS Y

Current approach: CSV migration



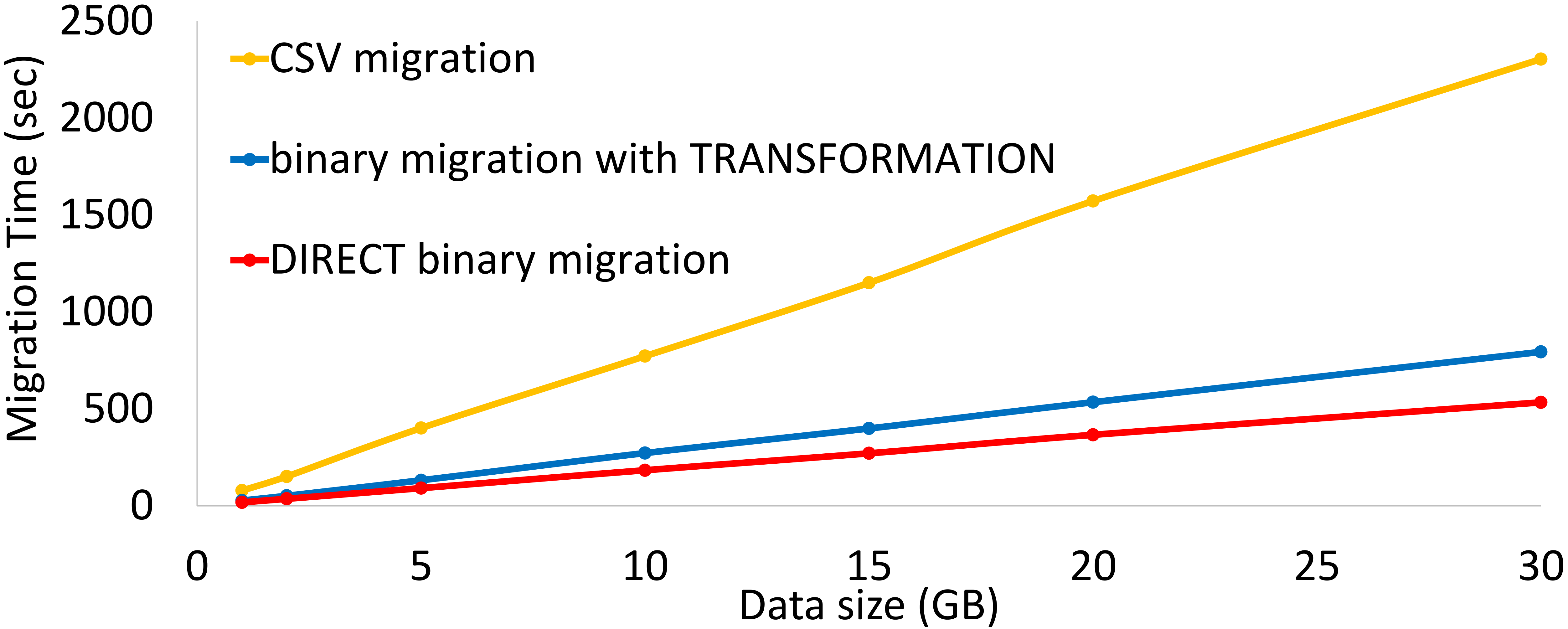
Data already loaded to the source database

Our approach: binary migration



Data Migration from PostgreSQL to SciDB

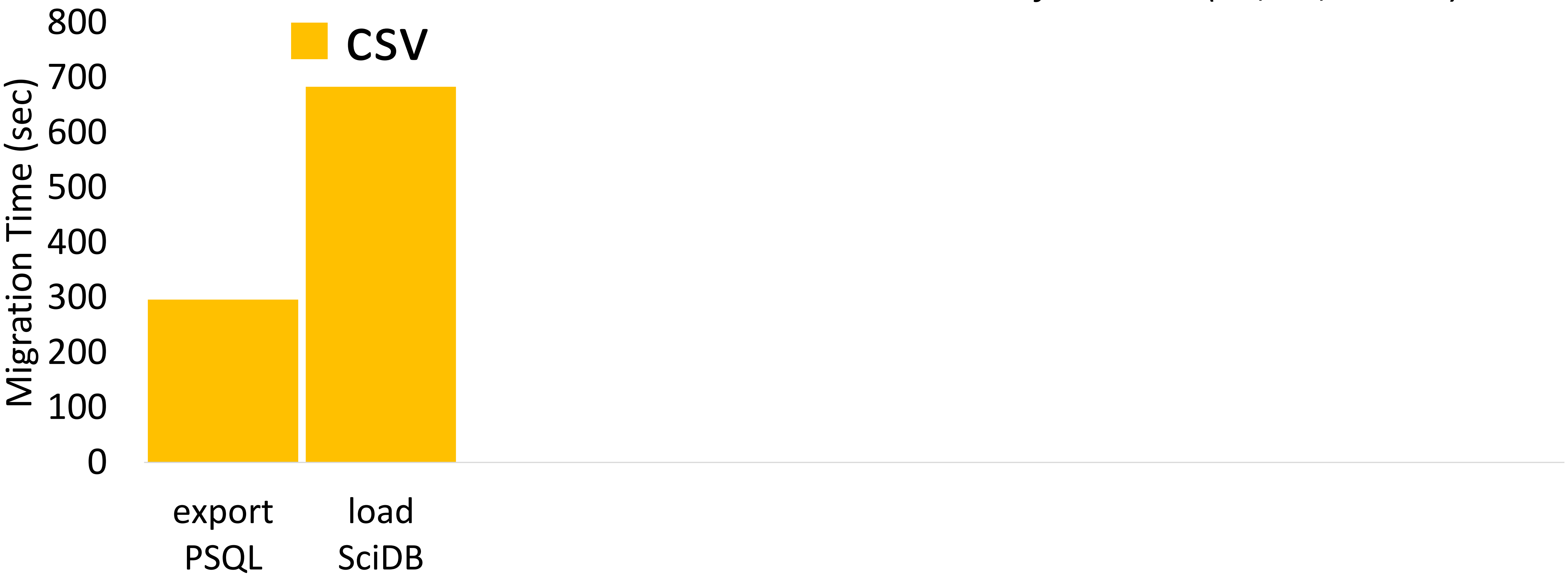
MIMIC II data - waveform(int, int, double)



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

Breakdown: migration from PostgreSQL to SciDB

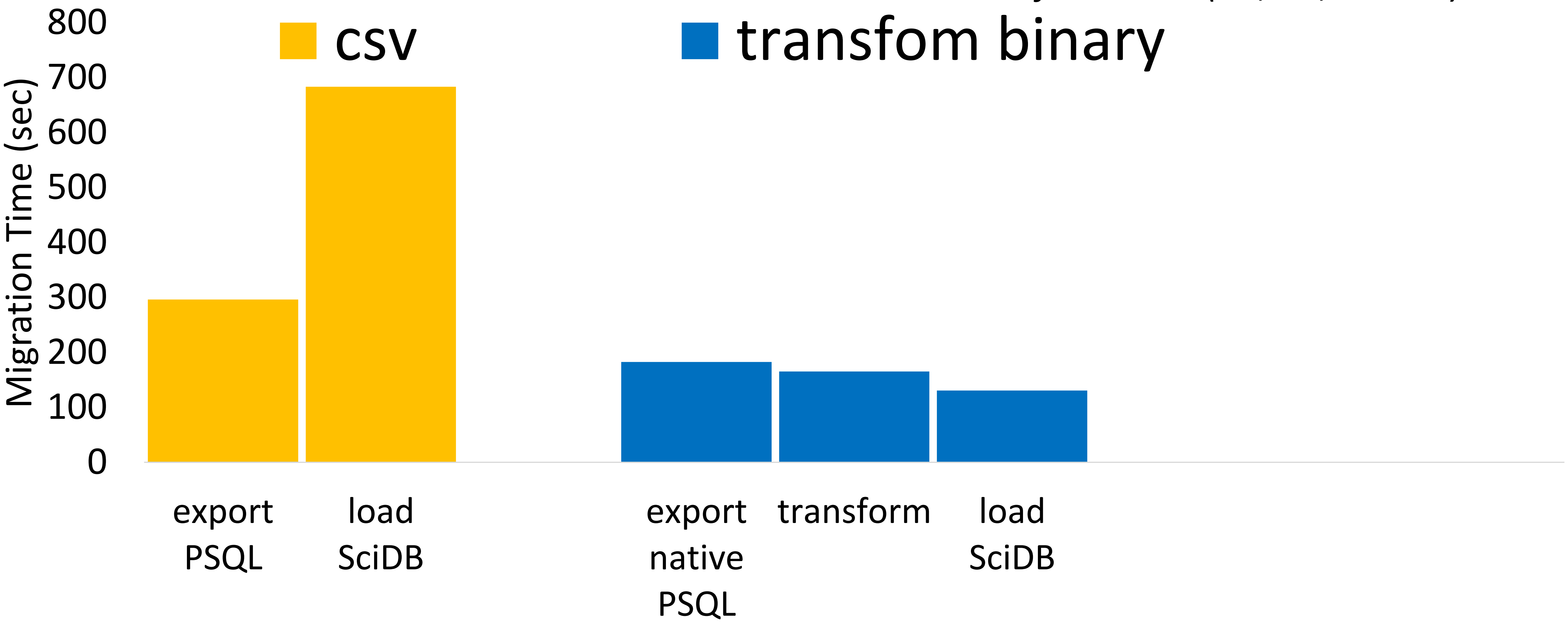
MIMIC II waveform data (int, int, double) 10 GB



Slow CSV loading

Breakdown: migration from PostgreSQL to SciDB

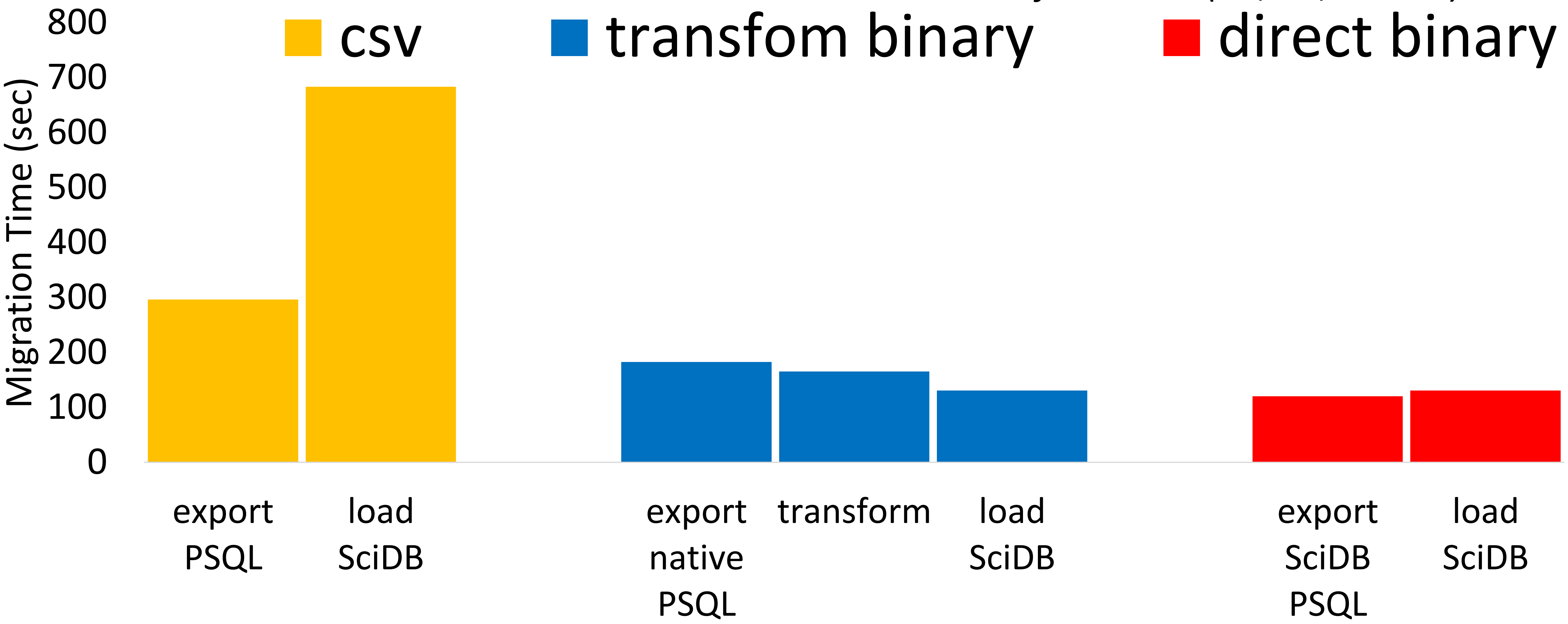
MIMIC II waveform data (int, int, double) 10 GB



Binary Export SLOWER than Binary Loading

Breakdown: migration from PostgreSQL to SciDB

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

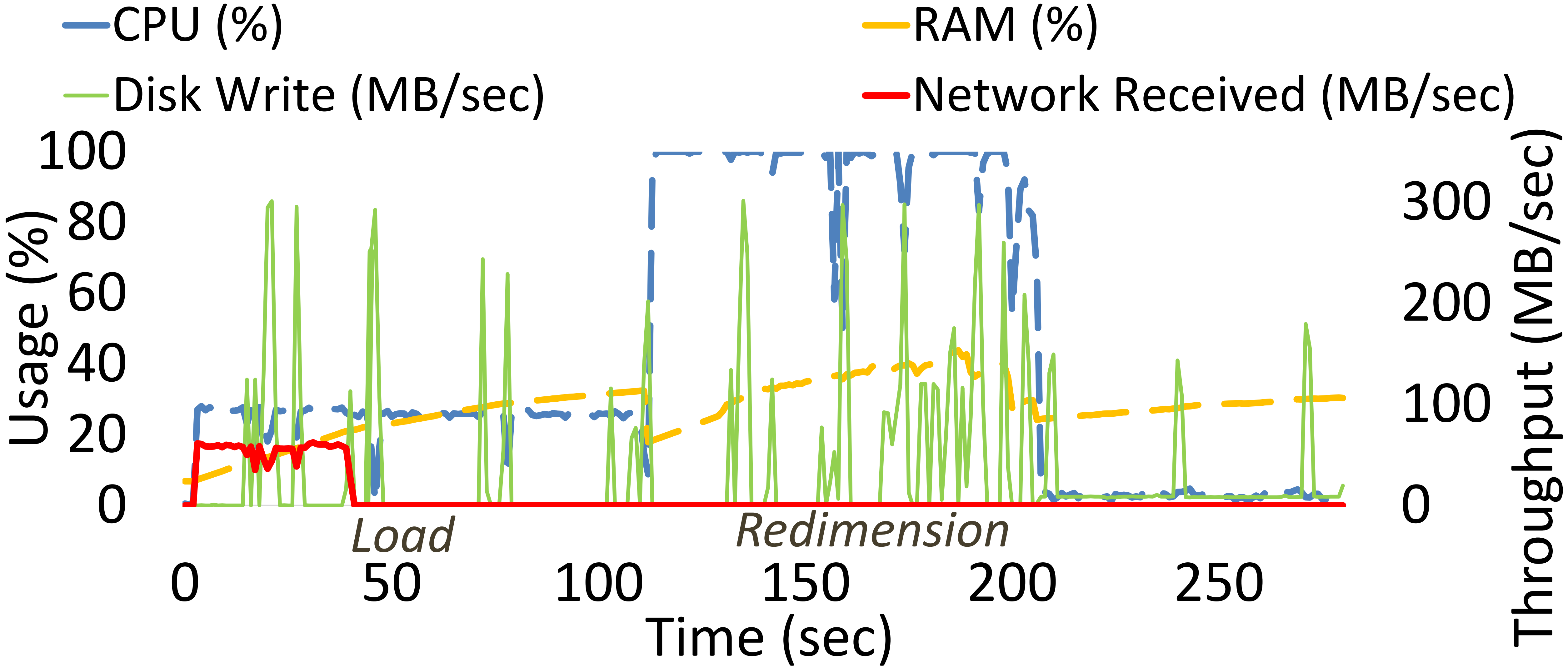


Fast Direct Binary Migration

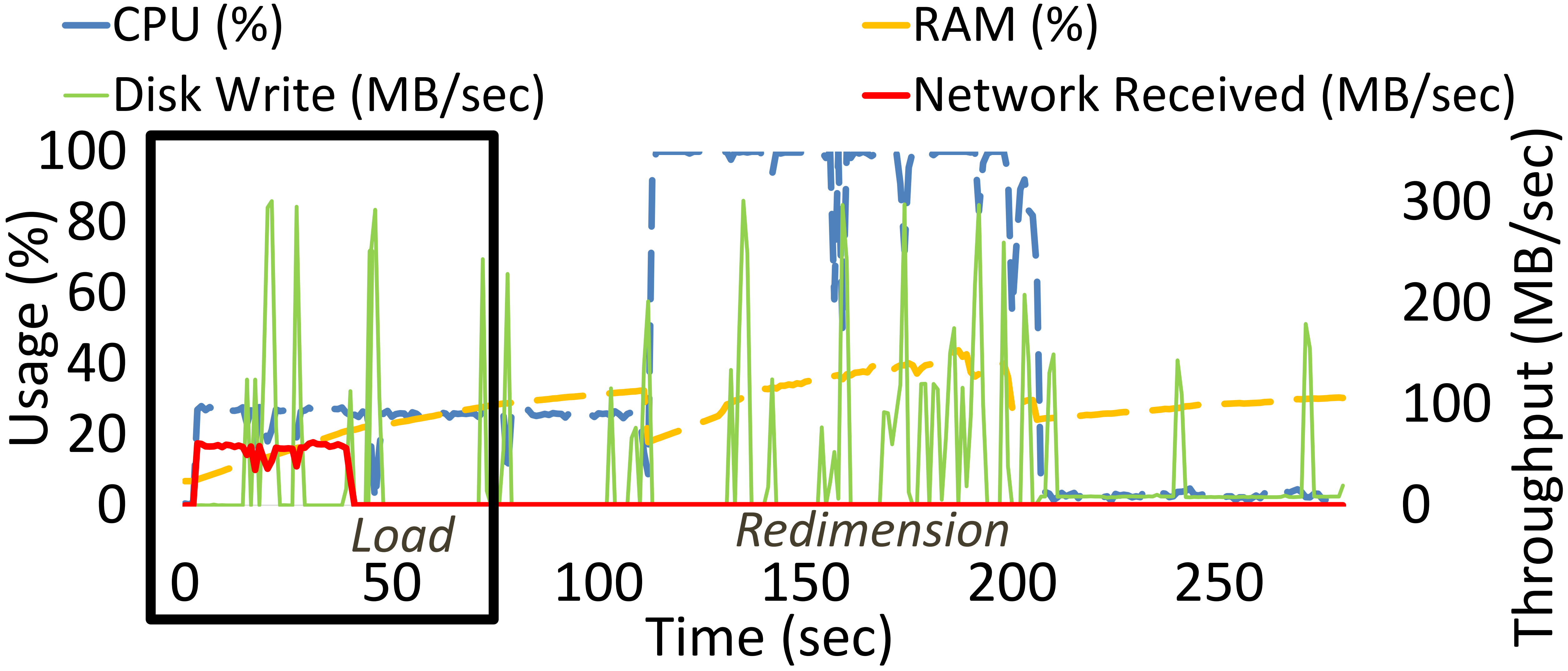
Agenda

- ❑ 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 wavefrom data loading to SciDB

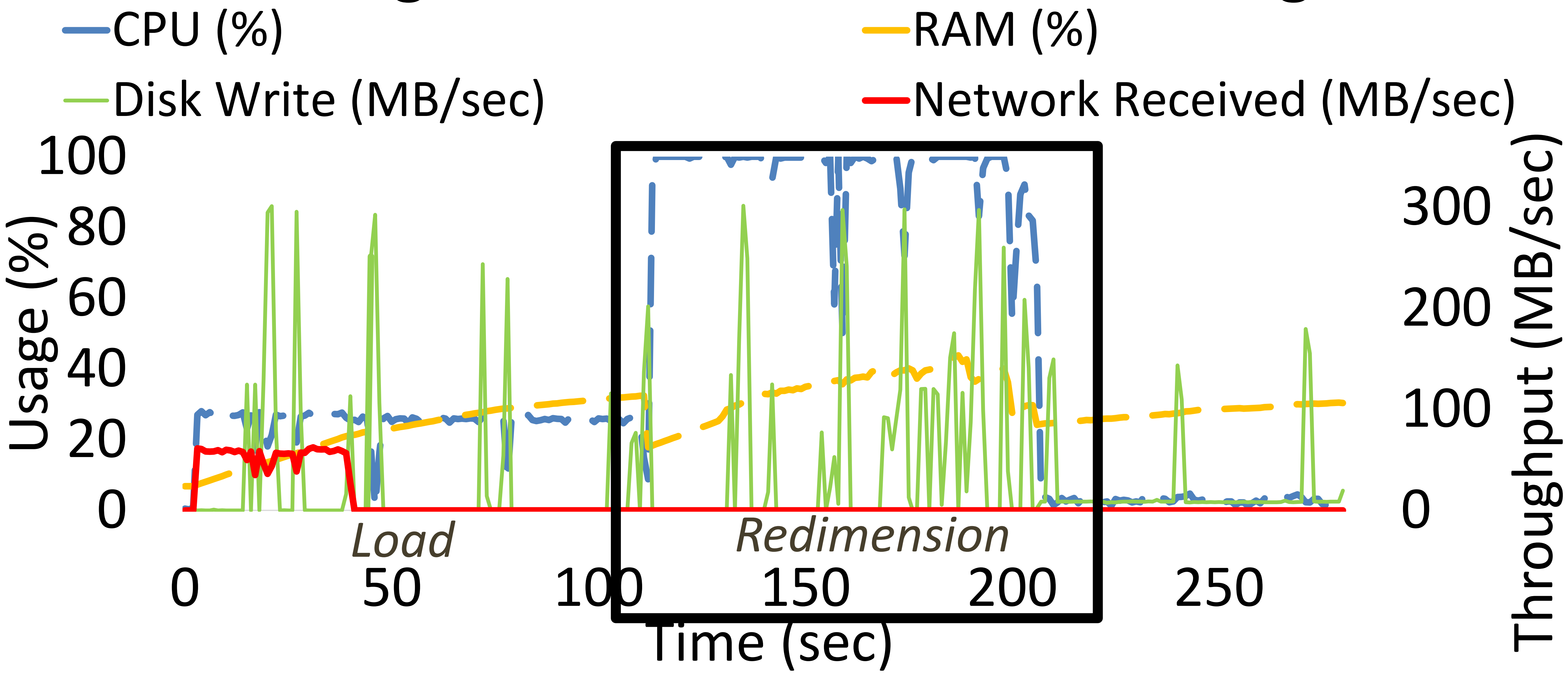


Resource usage: CSV wavefrom data loading to SciDB



Compress/**Decompress** to utilize spare CPU cycles

Resource usage: CSV wavefrom data loading to SciDB



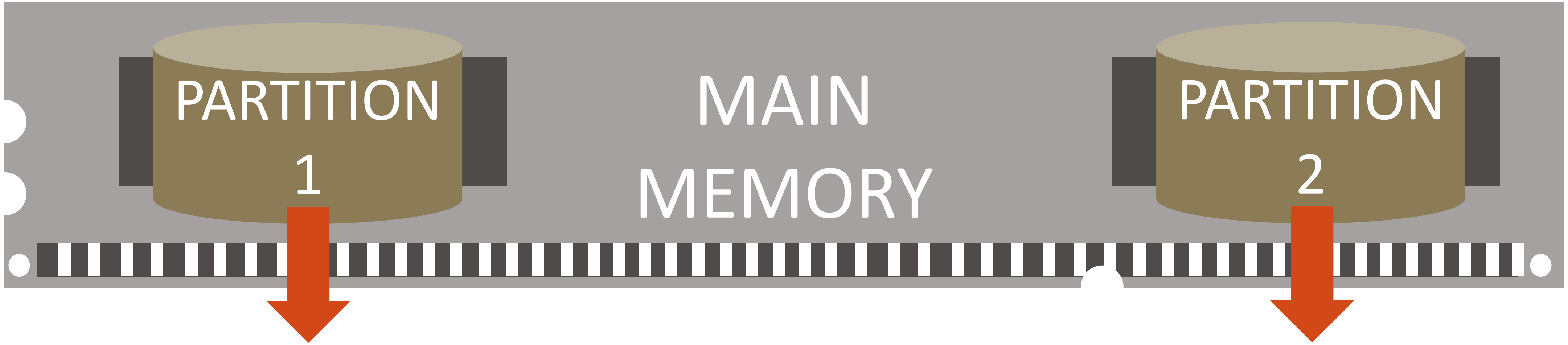
What is an optimal degree of parallelism?

Agenda

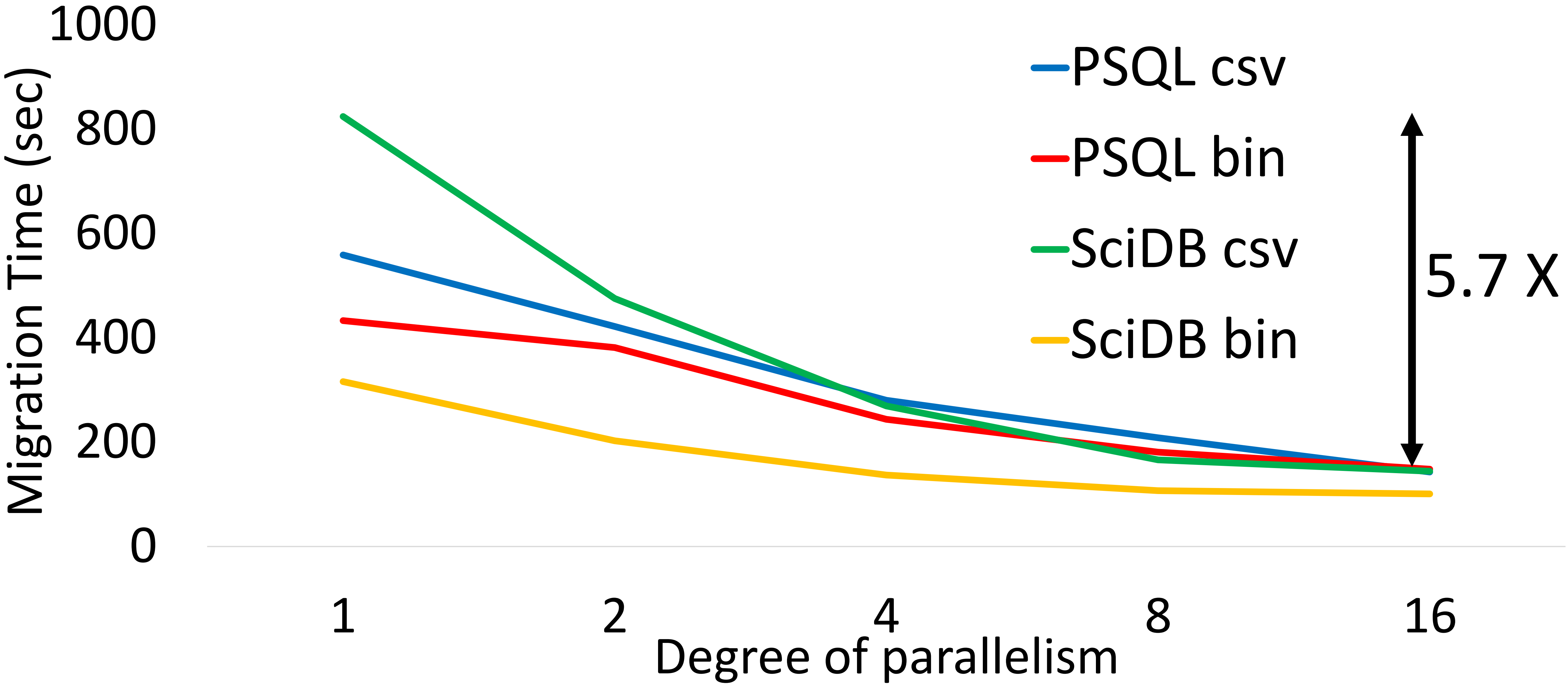
- ❑ Data Migration for Polystores:
  - ❑ What & Why?
  - ❑ How?
- ❑ Acceleration of physical data migration via:
  - ❑ Data formats and transformations
  - ❑ Resource-awareness
  - ❑ **Parallelism and compression**
  - ❑ Adaptivity
- ❑ Conclusion: **Fast Data Migrator**

Data Migration from S-Store to PostgreSQL & SciDB

- Enhanced data export from S-Store
  - Binary PostgreSQL
  - Binary SciDB
- **Parallel export** via partitioning

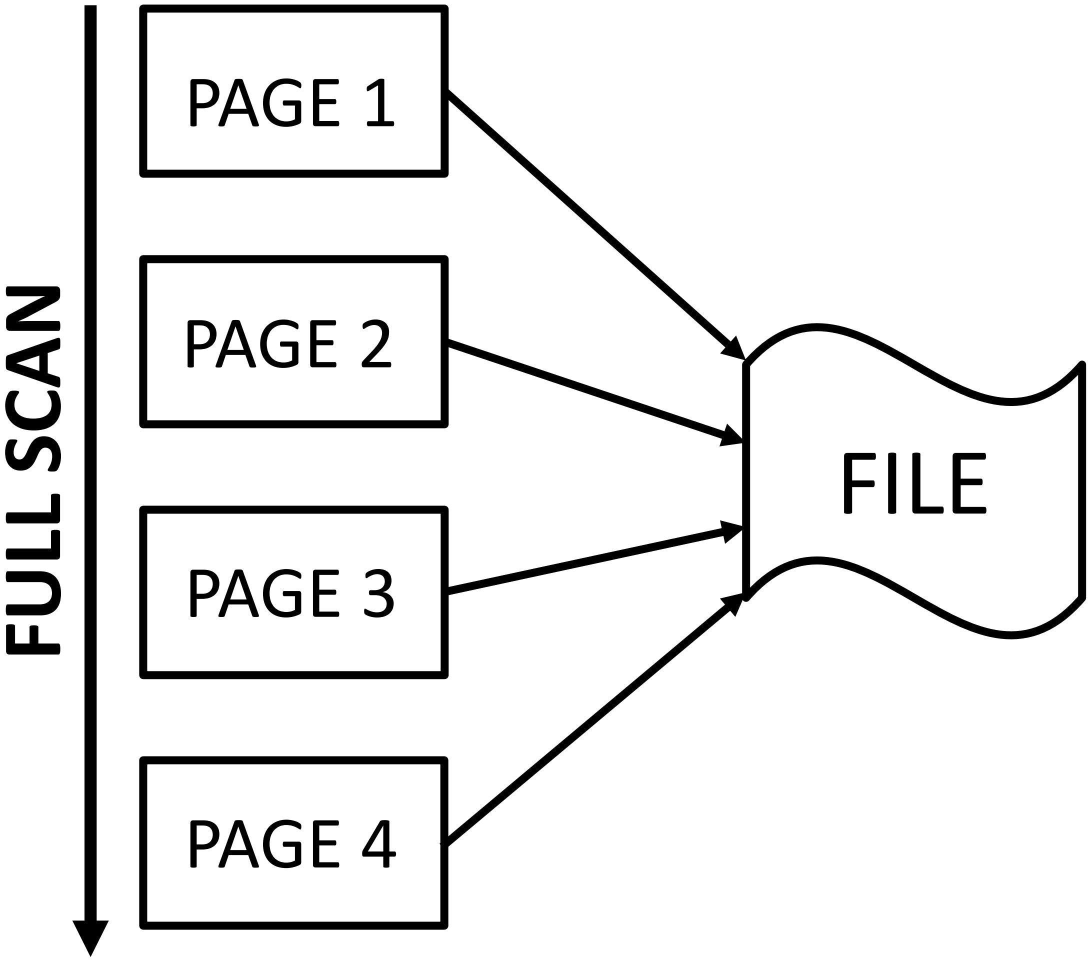


Data Migration from S-Store to PostgreSQL & SciDB

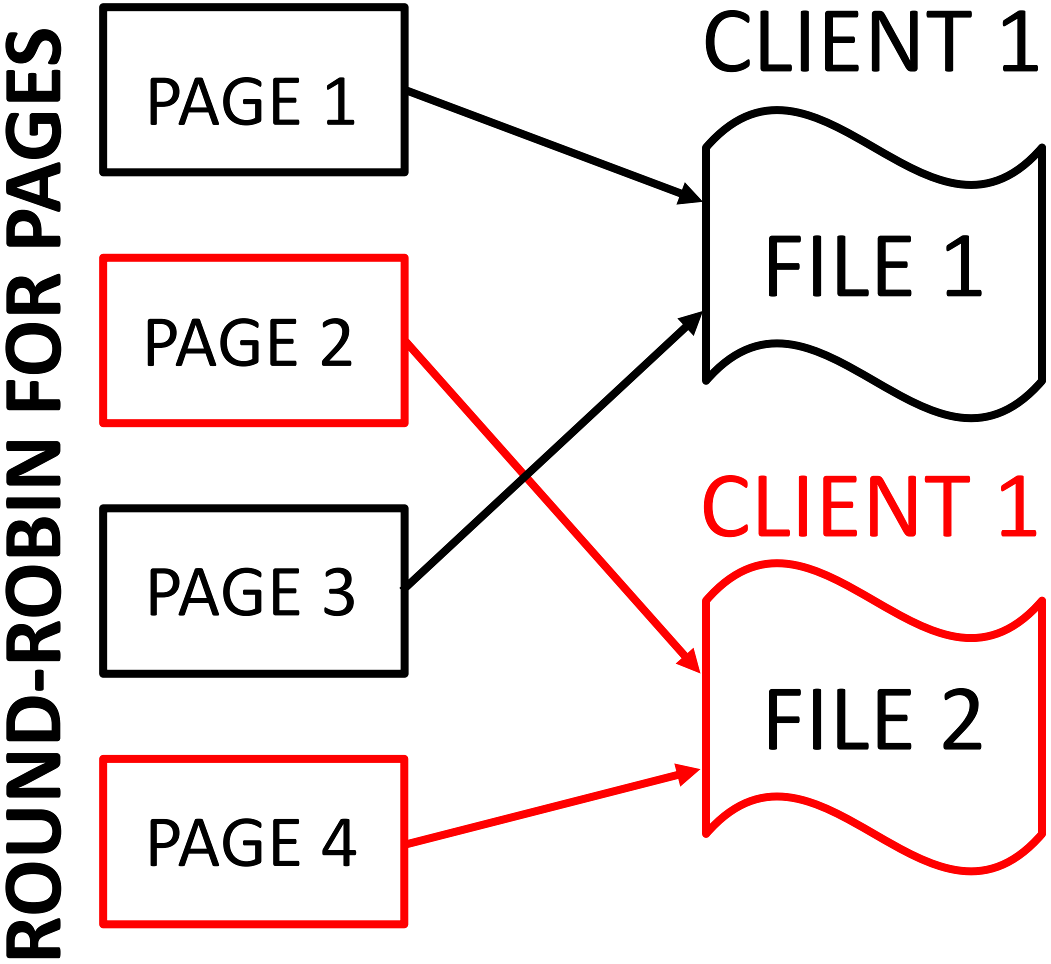


Time for CSV and binary migration converges for high degree of parallelism

Design of Parallel Export from PostgreSQL: **We CARE**



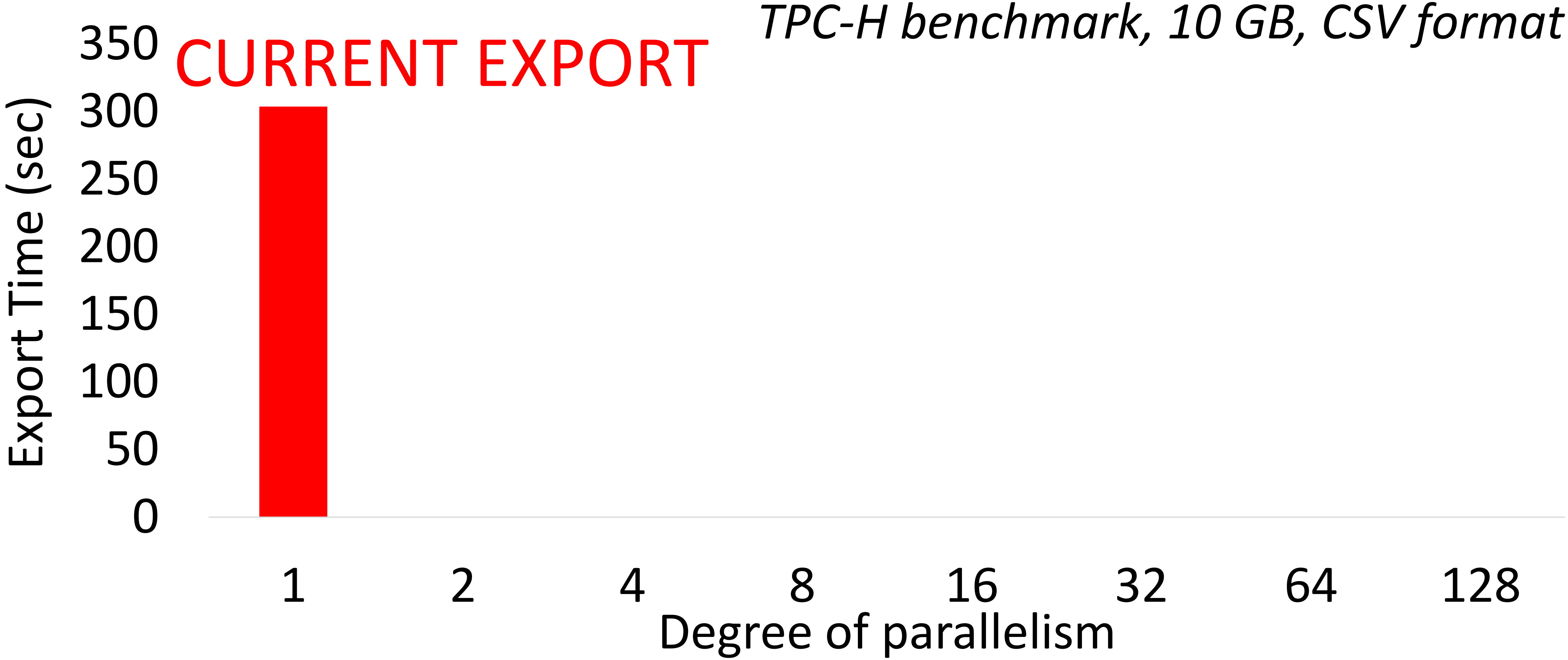
Current single-thread export



New parallel export

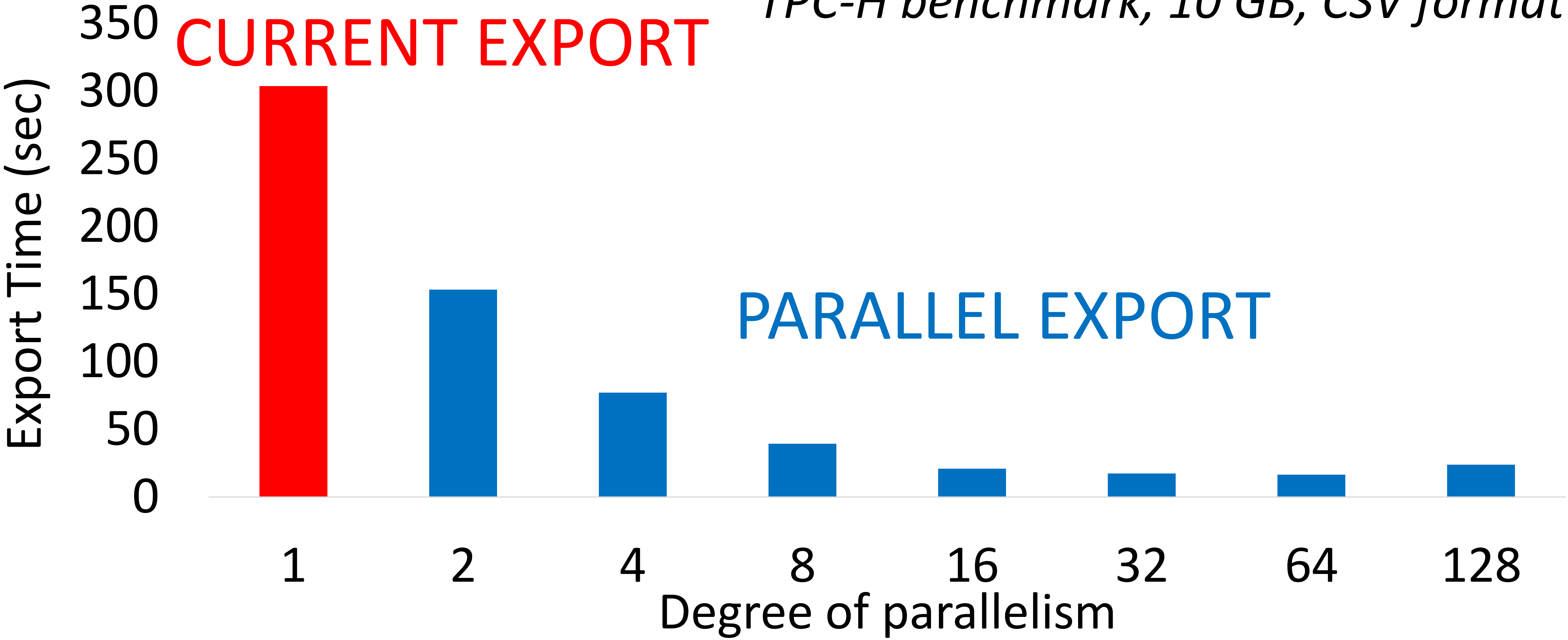
Parallel export from PostgreSQL

TPC-H benchmark, 10 GB, CSV format



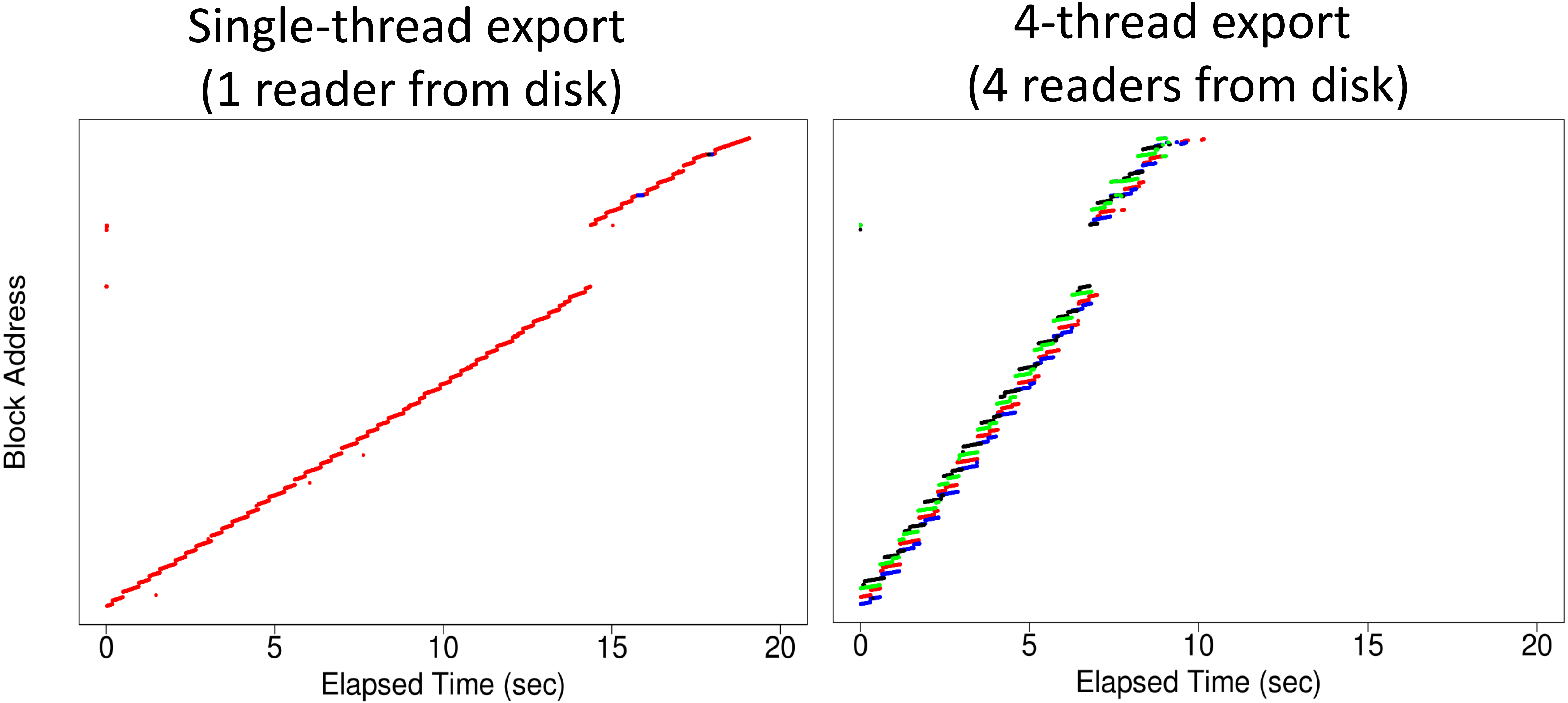
Parallel export from PostgreSQL

TPC-H benchmark, 10 GB, CSV format



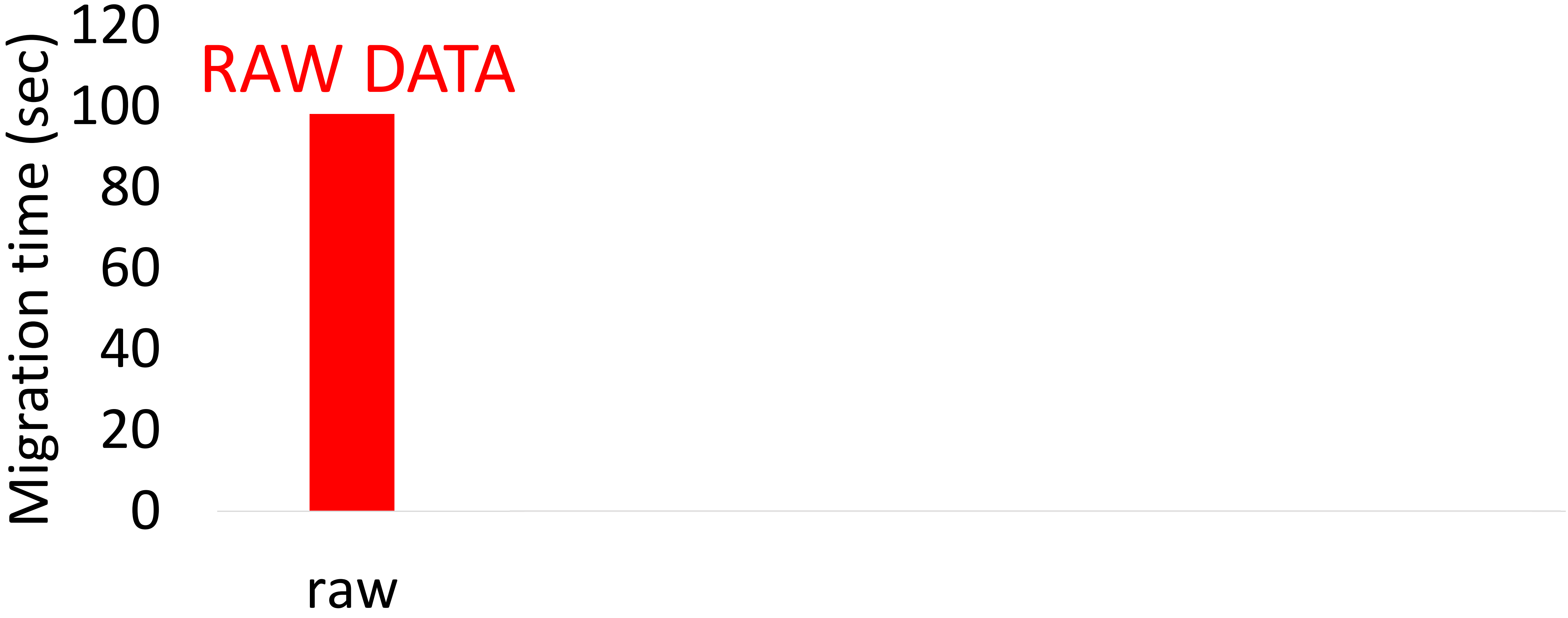
New Parallel export 20X faster than Current export

Single-threaded vs. Parallel Export from PostgreSQL



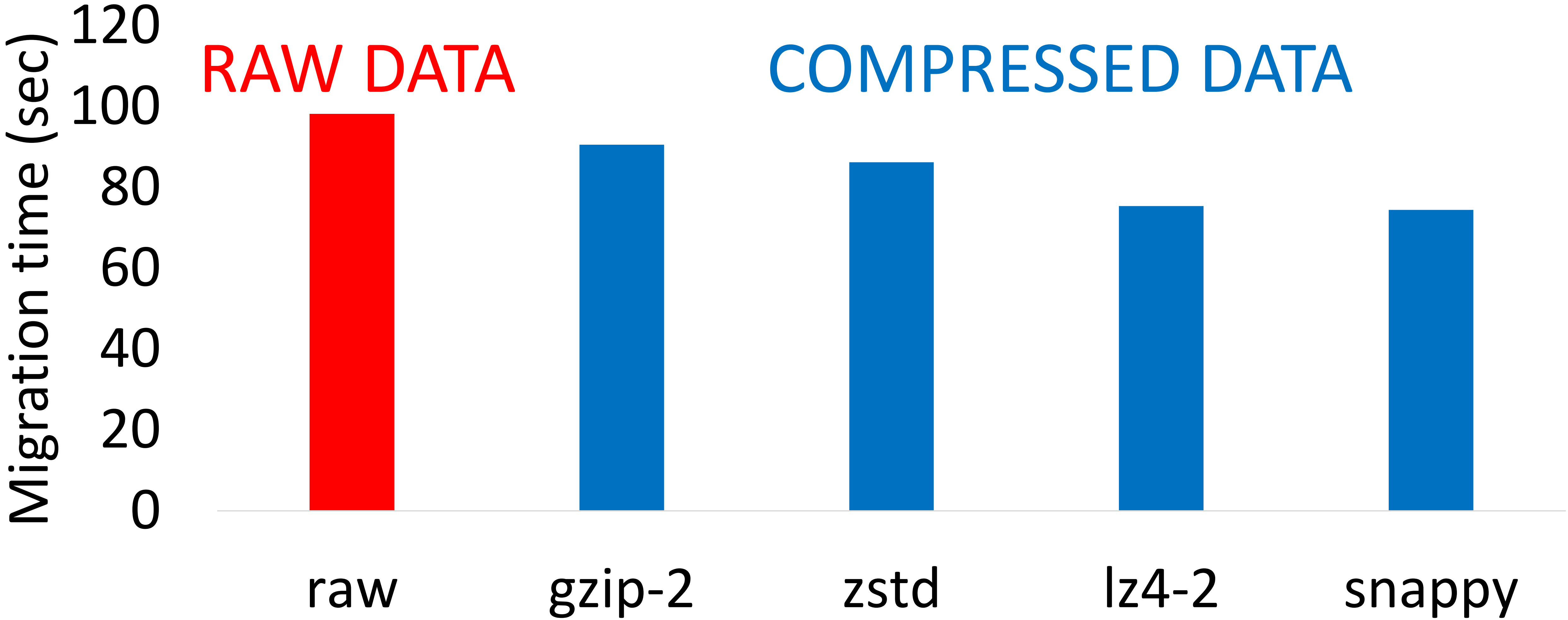
COMPRESSION for direct binary parallel migration

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



COMPRESSION for direct binary parallel migration

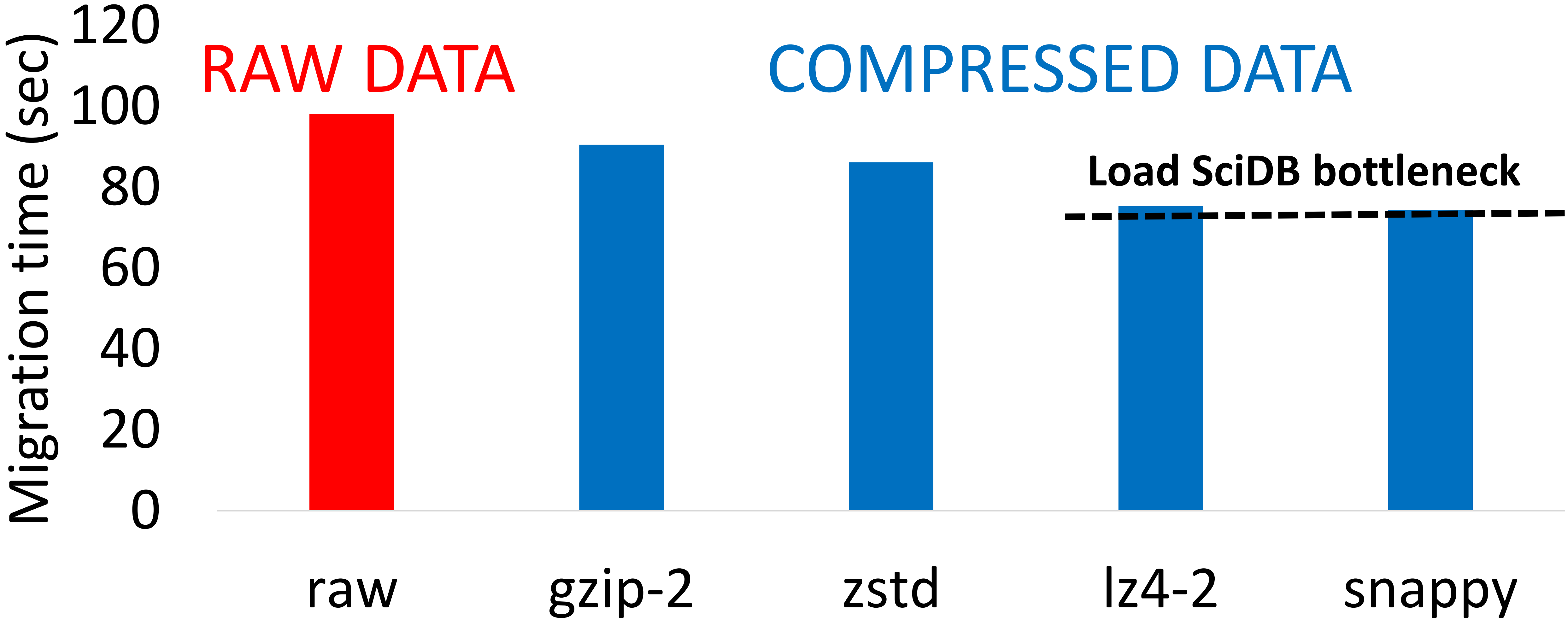
From PostgreSQL to SciDB, 4 threads, waveform data (int,int,double), 10 GB



Lightweight compression for data transfer via network

COMPRESSION for direct binary parallel migration

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

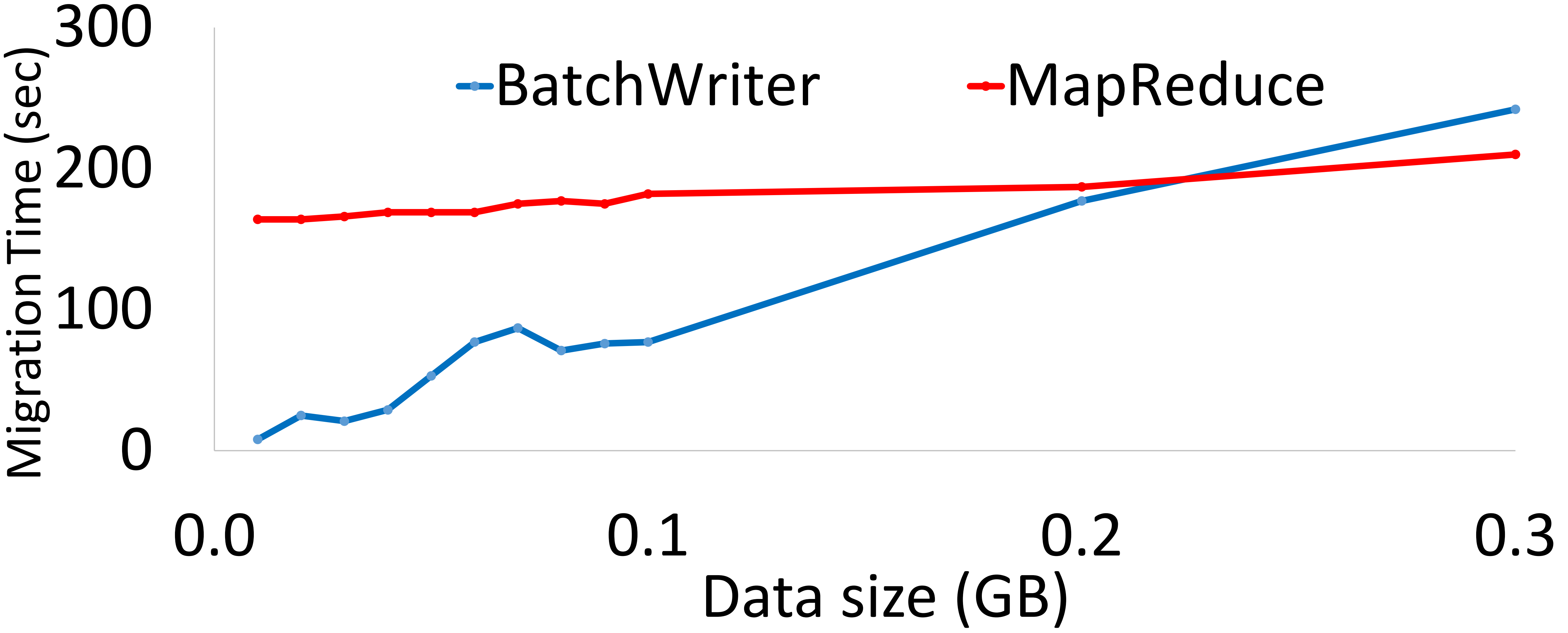


Lightweight compression for data transfer via network

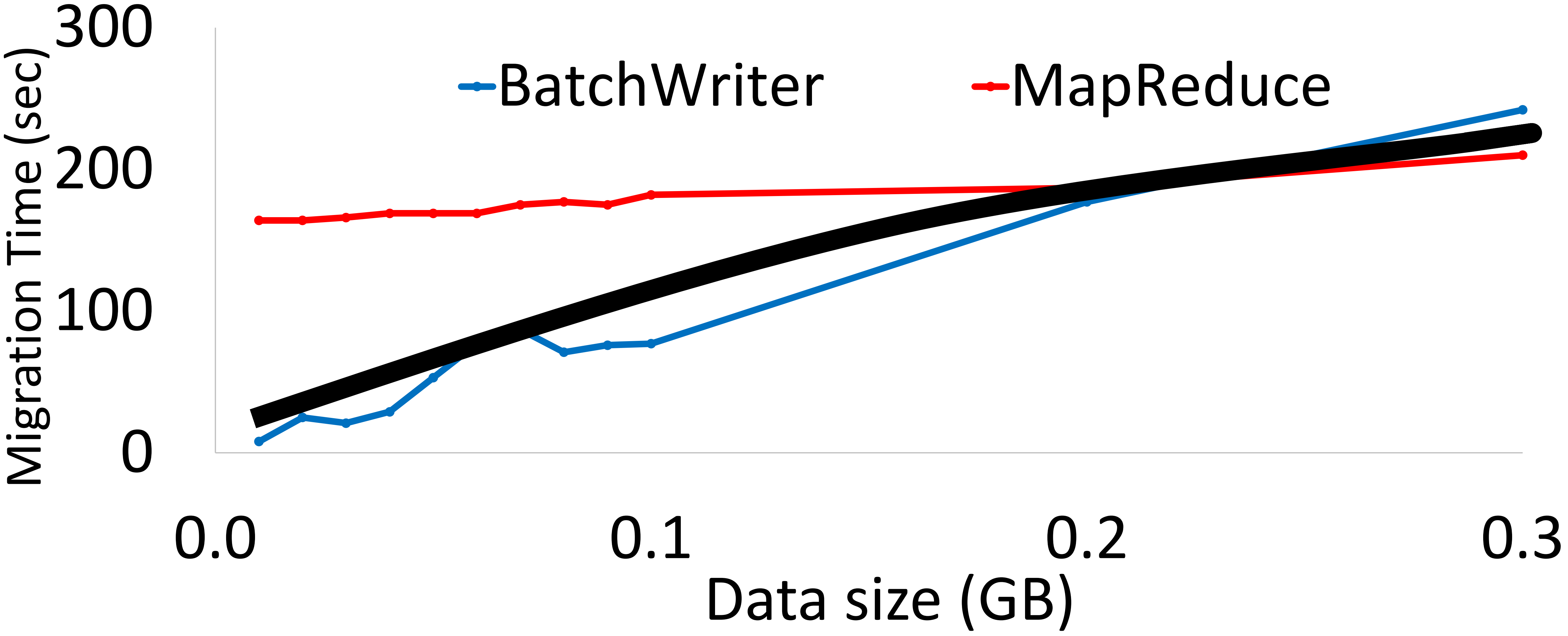
Agenda

- ❑ 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

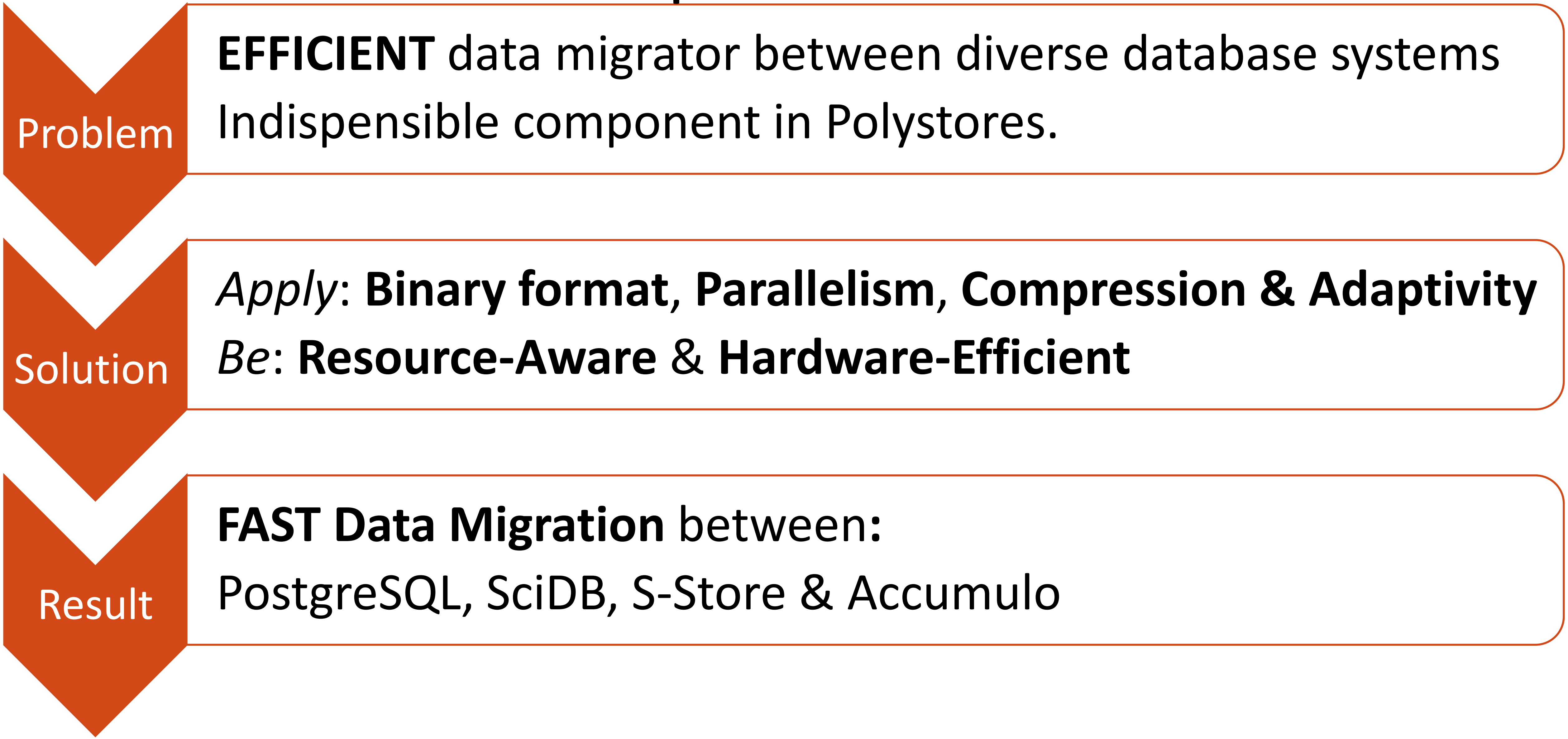


Data Migration from PostgreSQL to Accumulo



Adaptive data loading method

3 Step Conclusion



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

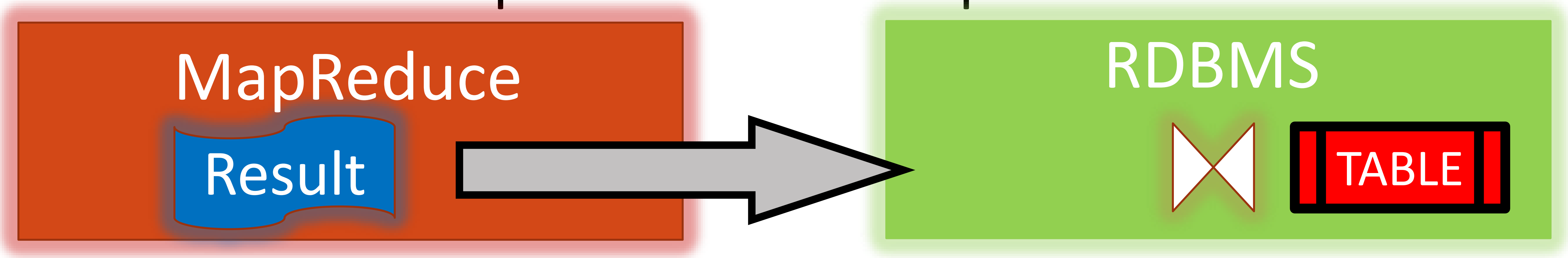
Complex analytics and many more database management systems require data migration!

Why binary despite parallel CSV migration?

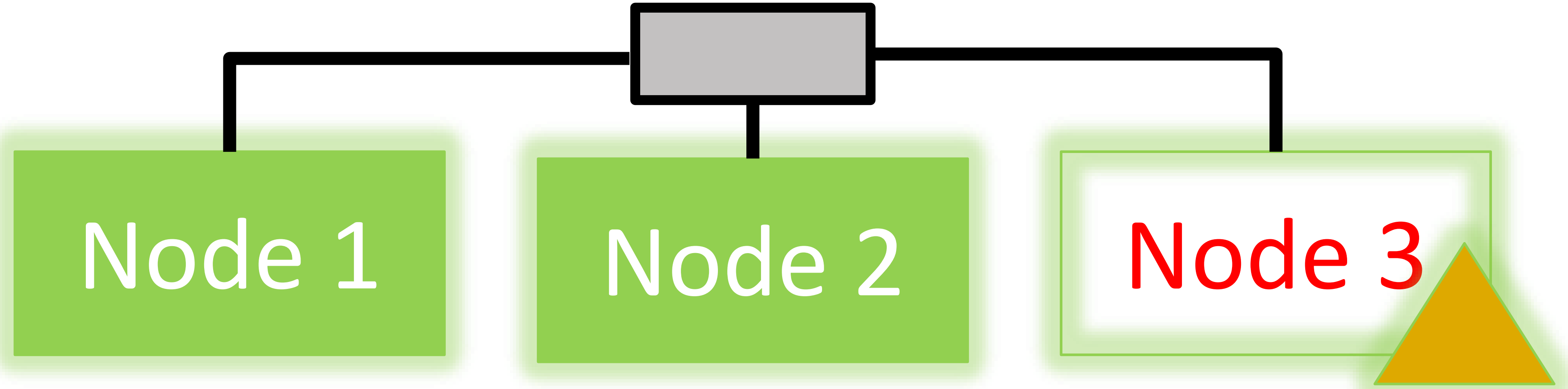
- ❑ **Binary migration for high degree of parallelism (e.g. 16) is still about 44% faster than CSV migration** (from S-Store to SciDB)
- ❑ Cannot allocate all the cores to the migration process
- ❑ CSV migration incurs greater energy consumption
- ❑ It is not always feasible to divide the CSV data (evenly) into chunks / partitions (e.g. dut 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



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

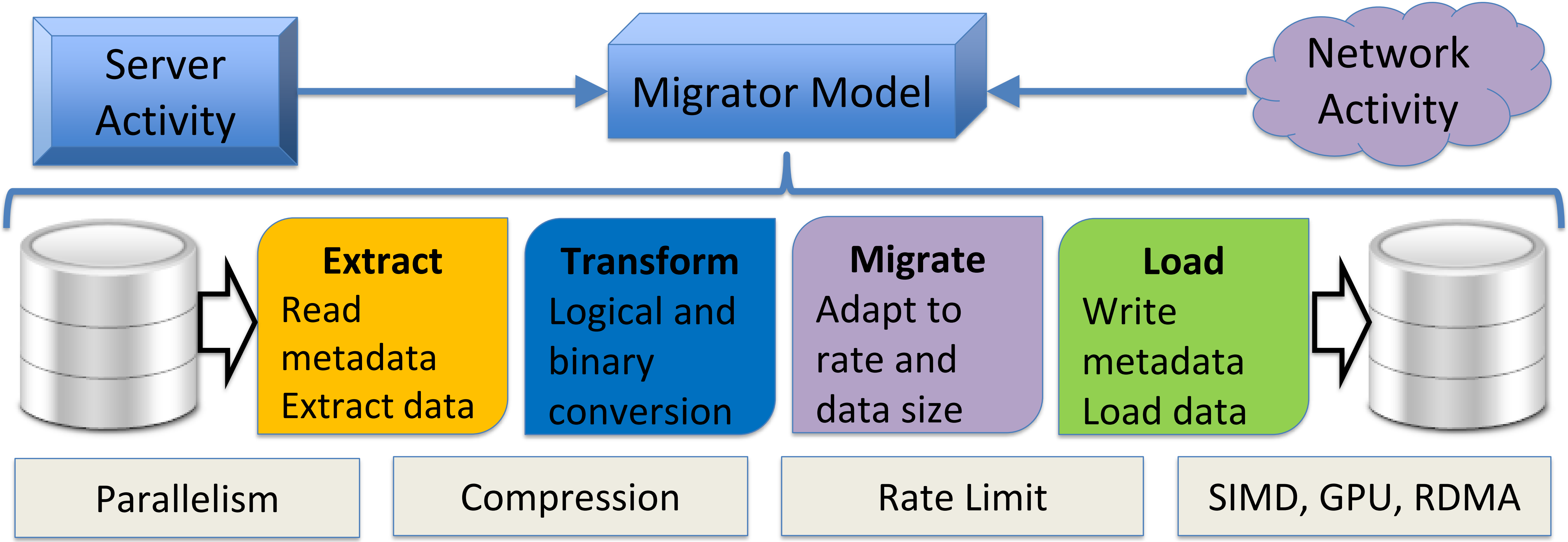


Data migration from PostgreSQL to SciDB

TPC-H benchmark, 10 GB

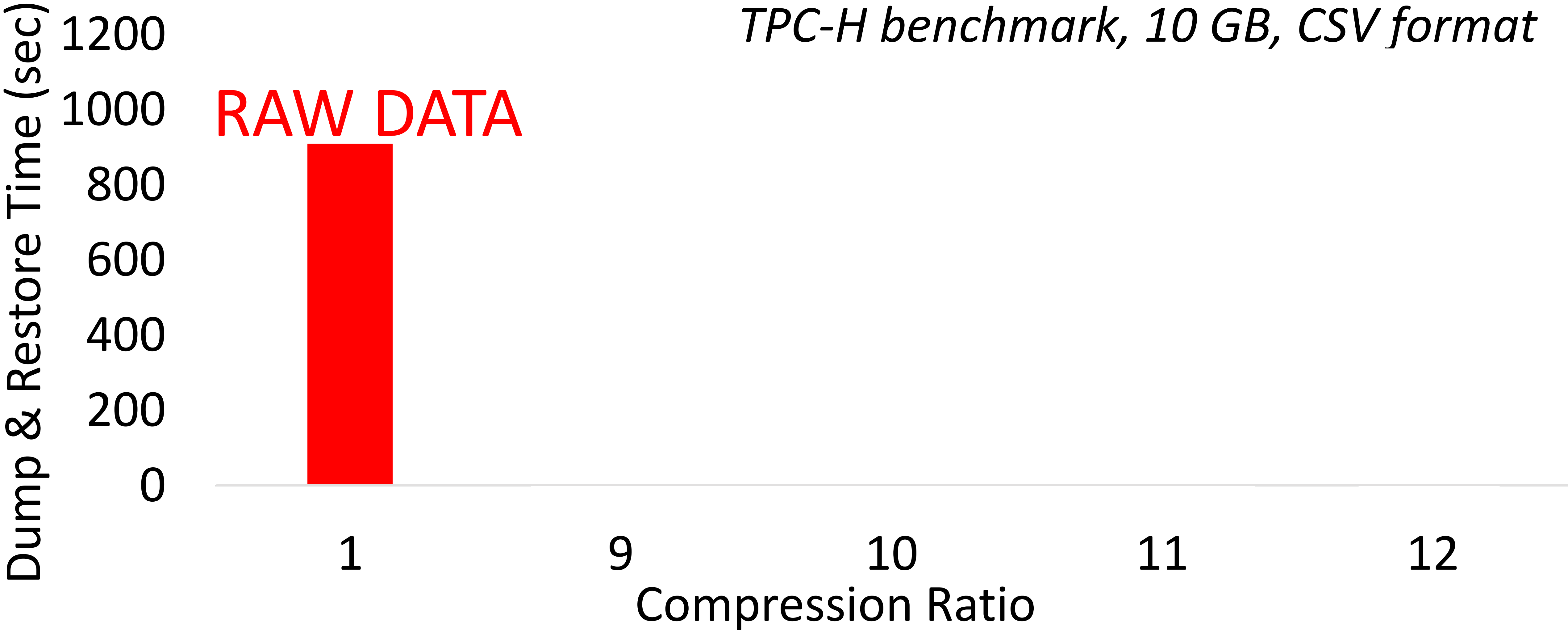
METHOD	TIME (sec)
JDBC	1000
CSV	800
Binary format with transformation	270
Direct binary format	180
Parallel direct binary format	*90*
Parallel direct database native storage	*62*
GPU parallel direct database native storage	*40*

Future directions for Data Migration Framework

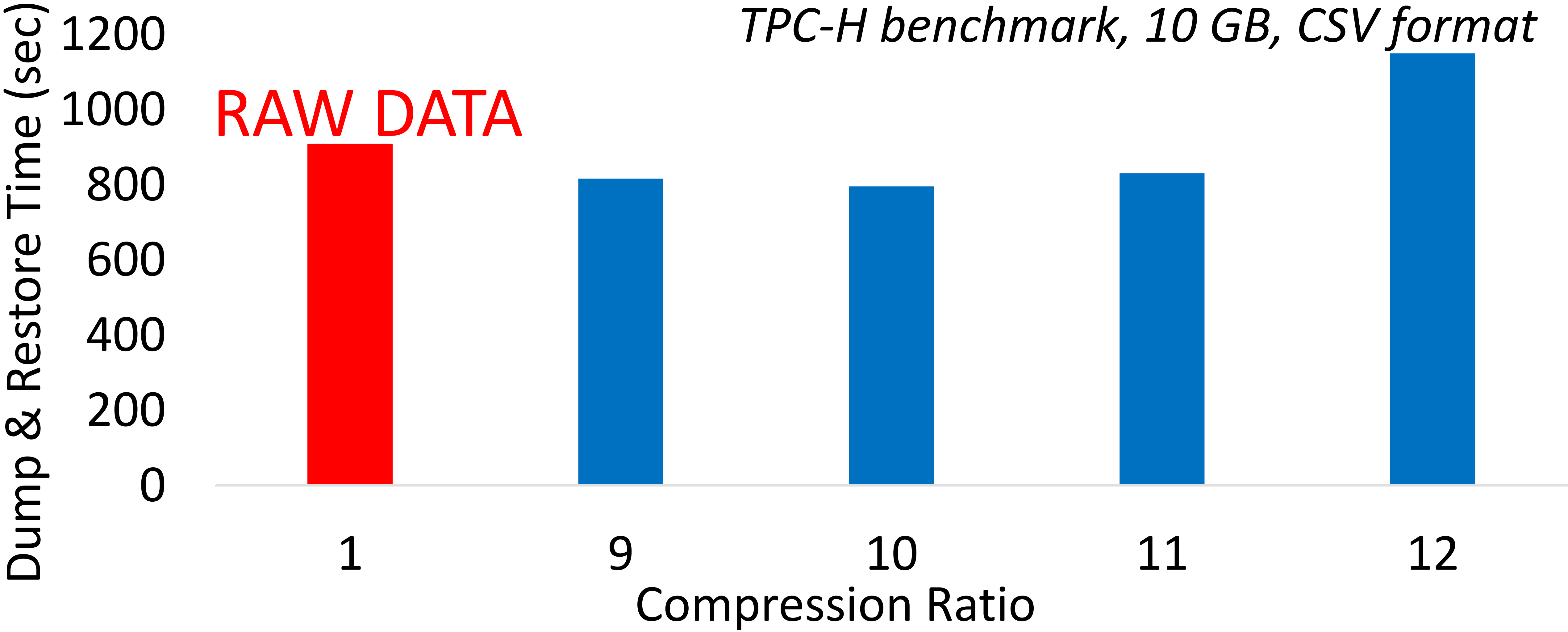


- ❑ monitor usage of resources (rate limite) & select migration approach
- ❑ apply compression, select # cores for parallel loading,utilize hardware

Compression in PostgreSQL backup utilities

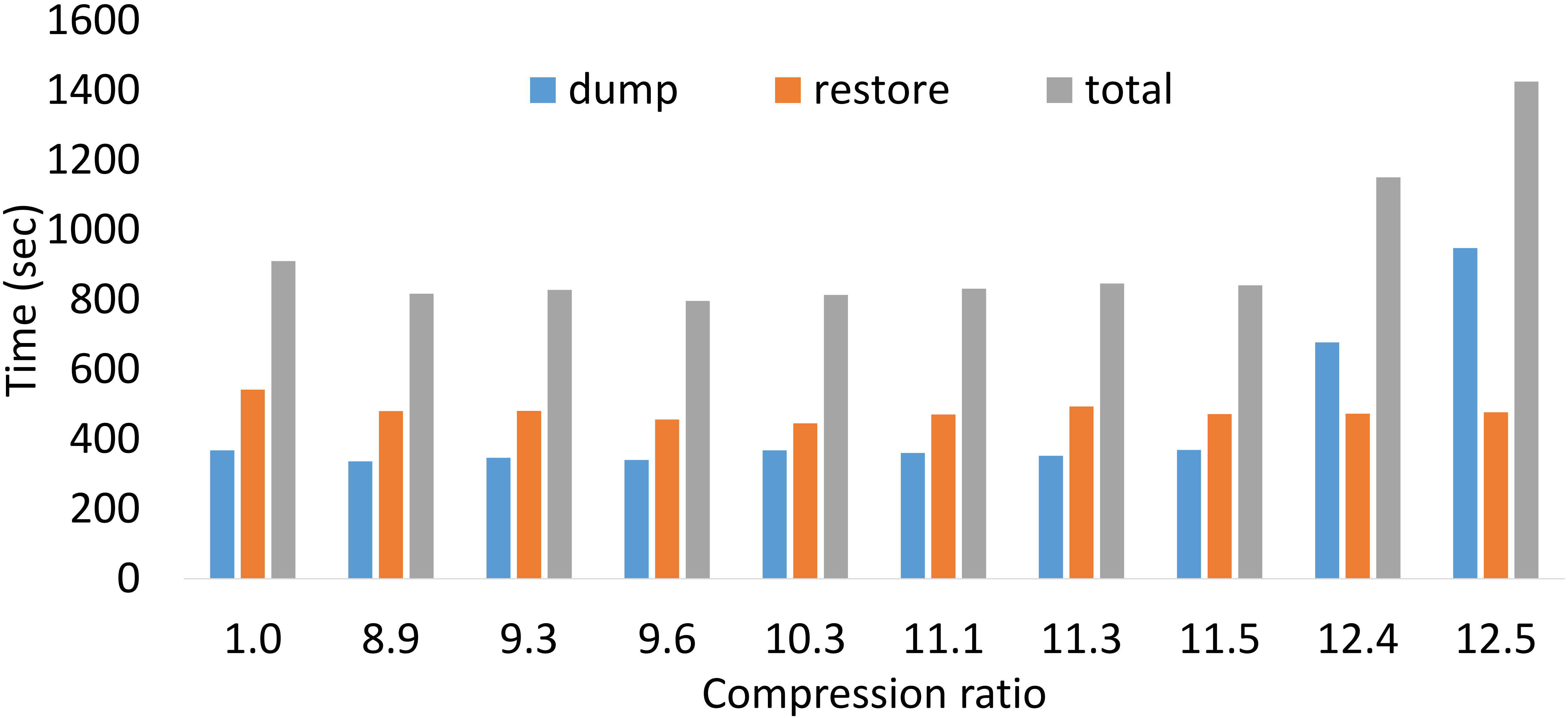


Compression in PostgreSQL backup utilities

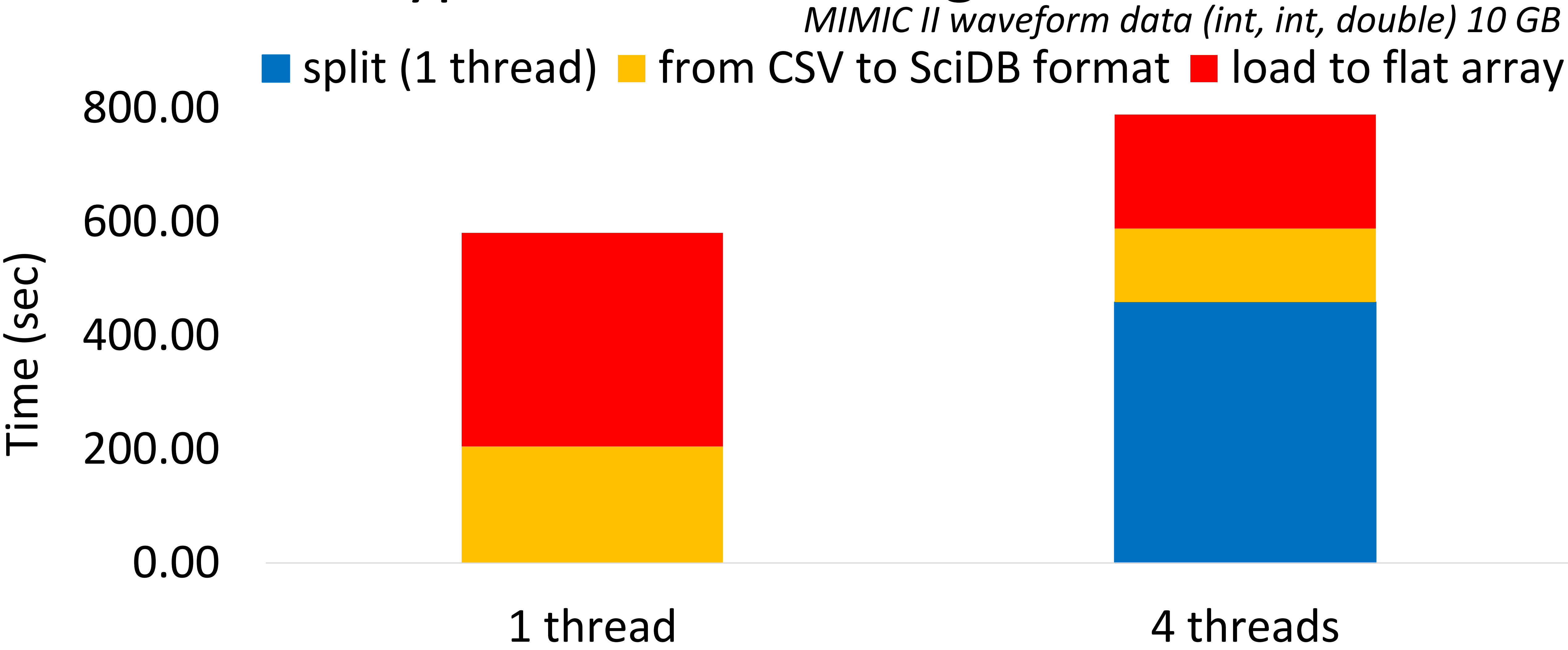


Speed-up migration and decrease data size 10X

PostgreSQL backup utilities: compression ratio



2 types of CSV loading to SciDB



The split phase is very slow!

Experimental setup for MIMIC-II data

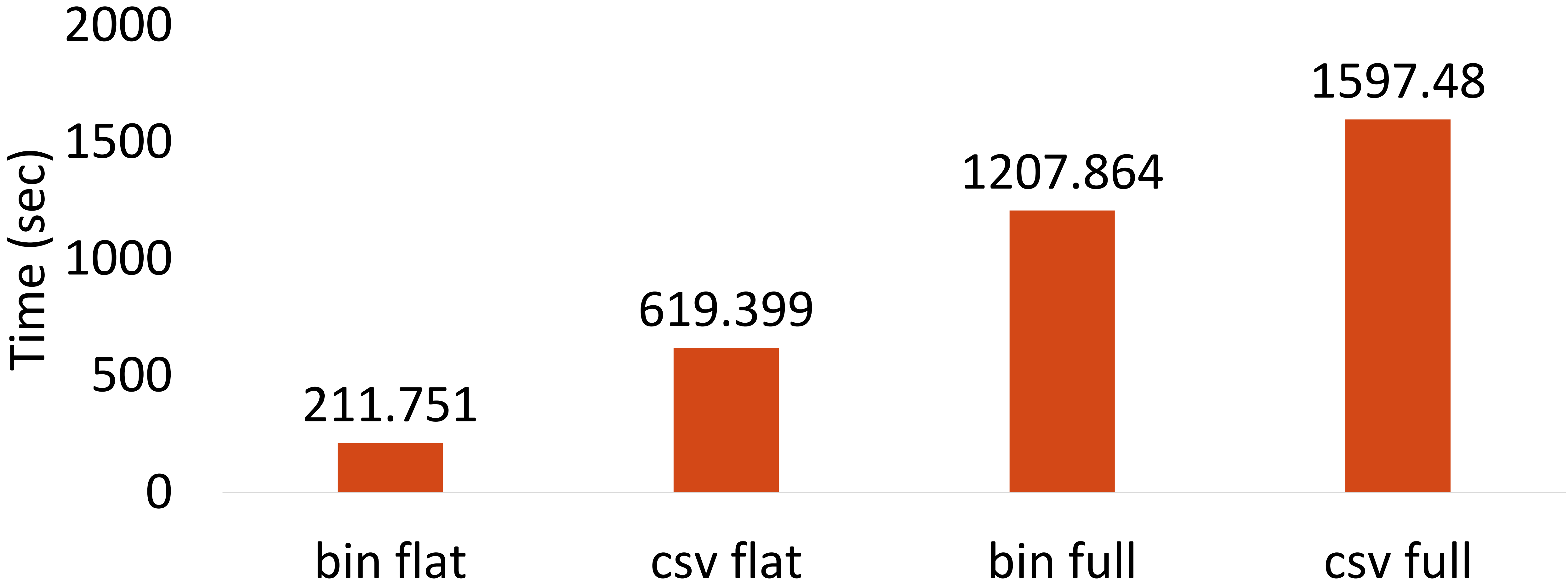
- Software:
    - PostgreSQL 9.4.5 (built with -02 optimization)
    - SciDB 14.12 (installed on a single node, 4 instances)
- Hardware:
    - Single node (Accumulo deployed on a cluster of 5 nodes)
    - Quad Core CPU with frequency of 3.1 GHz
    - 16 GB of main memory
    - 250 GB SSD (reads: 517 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 -02 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

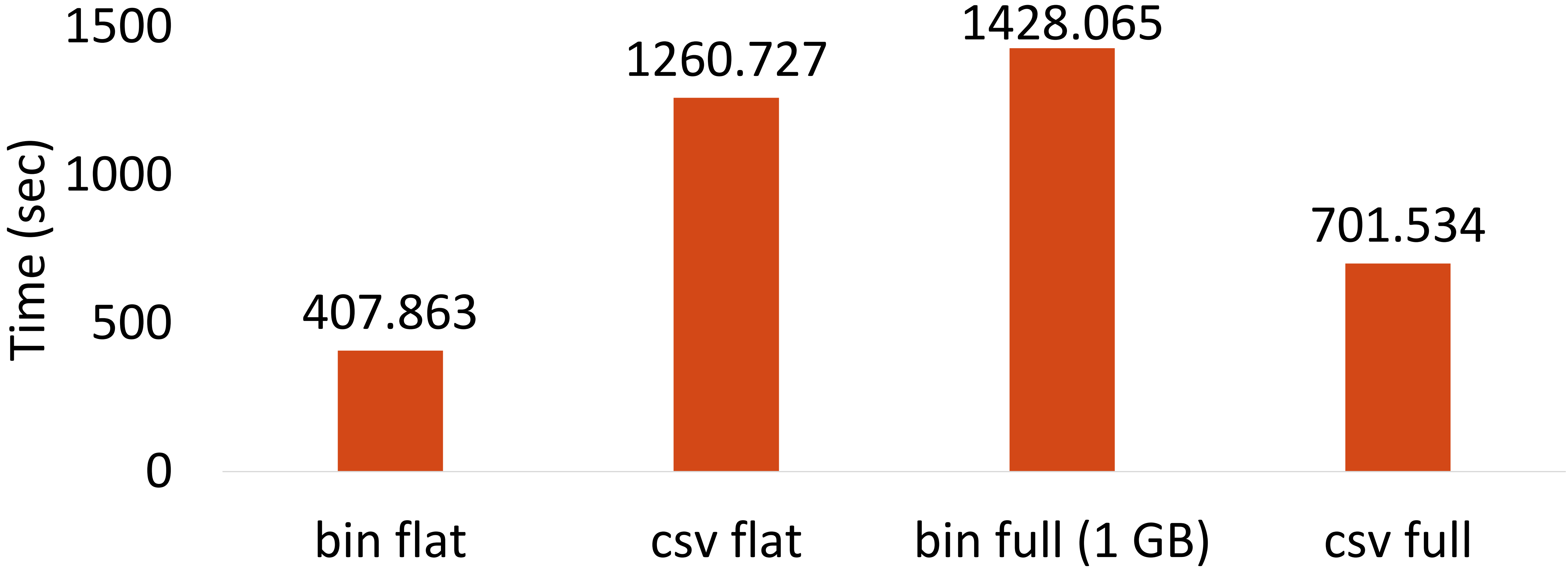
flat *(to a flat array)*, full - *with redimension , MIMIC II data - 10 GB waveform (int, int, double)*



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 binay migration from full array

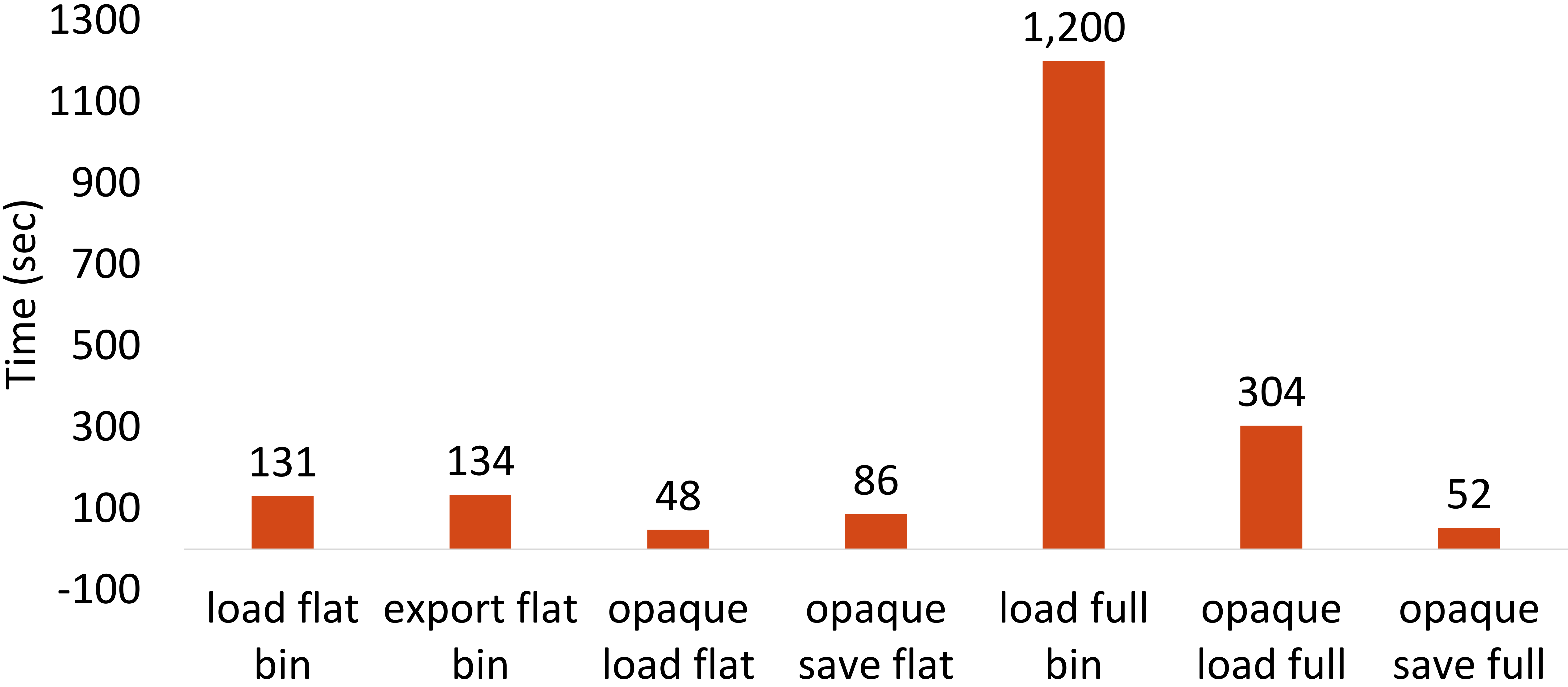
Future work

- 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

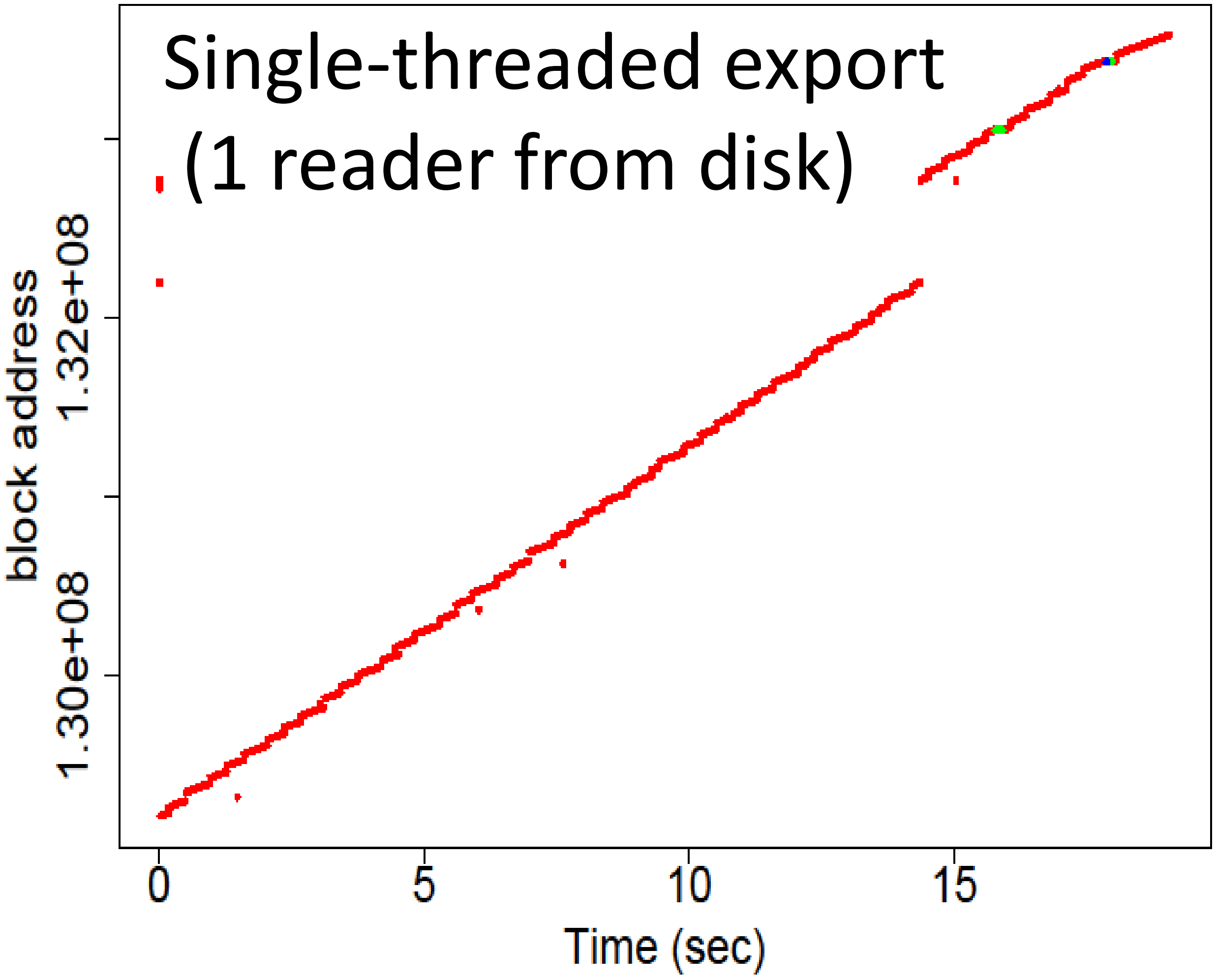
Distributed Data Migrator

- 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)

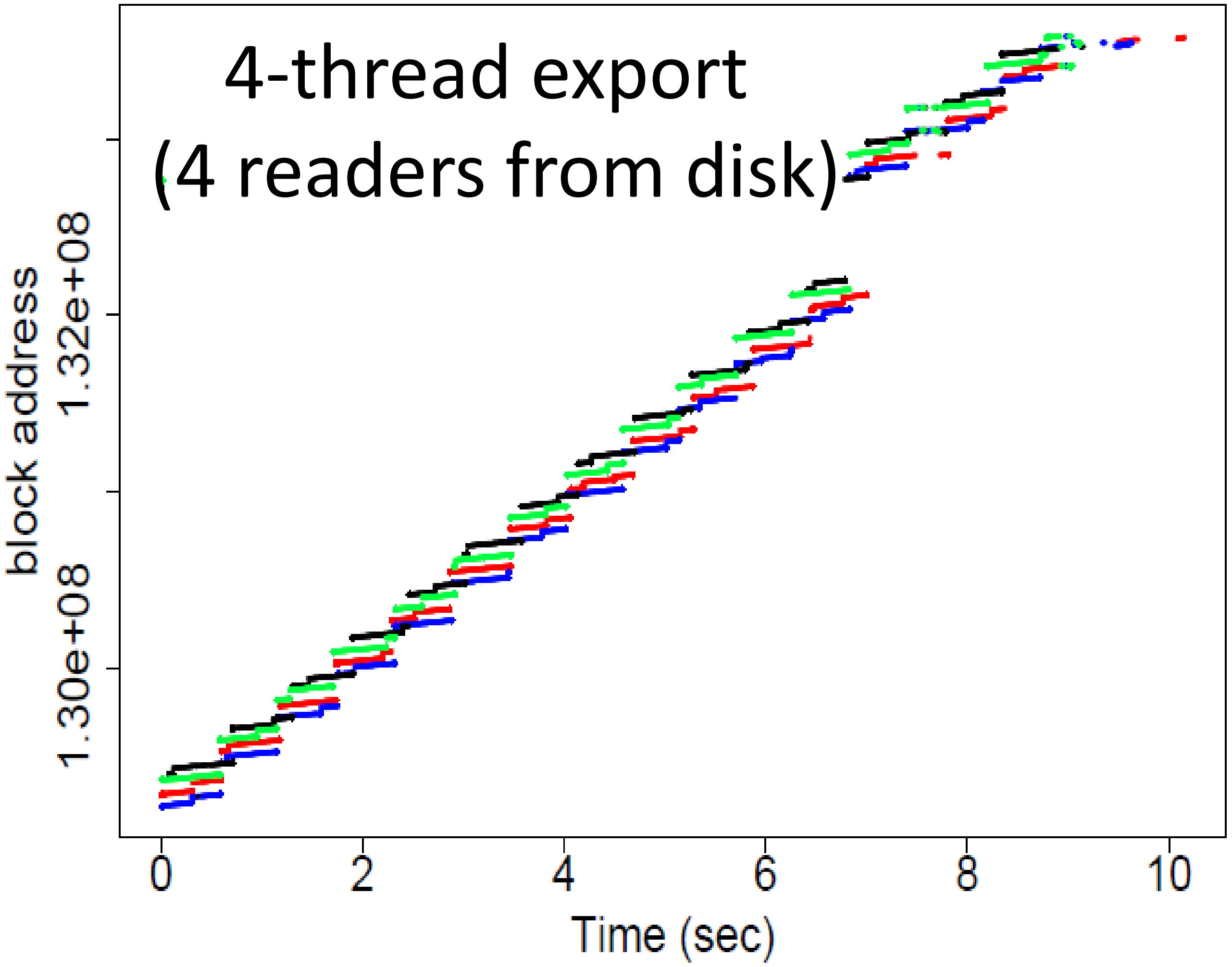
SciDB opaque format for multi-dimensional array



Single-threaded vs. Parallel Export from PostgreSQL



Single-threaded export
(1 reader from disk)



4-thread export
(4 readers from disk)

Better utilization of read bandwidth => better utilization of CPU

Polystore system vs. Federated database

Item	Polystore system	Federated database
Data models	Very diverse	Mainly relational
Control	One admin	Many admins
Placement	Collocated (one rack/datacenter)	Geographically decentralized
Components	Tightly coupled	Loosely connected
Concept	Data virtualization	Data federation

Data Migration in Polystores: **TWO WAYS**

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