Data Loading & Migration for Database Management Systems

PostgreSQL

Speaker: Adam Dziedzic

Advisor: Prof. Aaron Elmore

THE UNIVERSITY OF CHICAGO

CSV vs. DBMS

- ☒ BIG **RAW** DATA files
- ☒ Expensive raw analysis
- ☒ Fresh data = interesting data but unexplored
- ☒ Exponential growth

- ☑ ACID guarantees
- ☑ SQL interface
- ☑ Top query performance & convenient, fast analysis
- ☑ Linear complexity of loading

CSV vs. DBMS

- ☒ BIG **RAW** DATA files
- ☒ Expensive raw analysis
- ☒ Fresh data = interesting data but unexplored
- ☒ Exponential growth

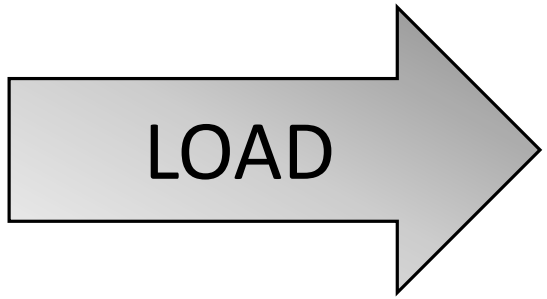
- ☑ ACID guarantees
- ☑ SQL interface
- ☑ Top query performance & convenient, fast analysis
- ☑ Linear complexity of loading

Only a small fraction of raw data is loaded to DBMSs [IDC 12]

How to accelearate data loading & migration?

1. **Loading:**
   a) CSV vs. DBMS
   b) General principles
   c) Thread/Process Level **Parallelism**
   d) Identify Bottlenecks
   e) Using **Storage Devices**
   f) Data level parallelism (SIMD)
2. **Migration:**
   a) Leverage Diverse Databases
   b) CSV vs. **Binary format**

Data Loading

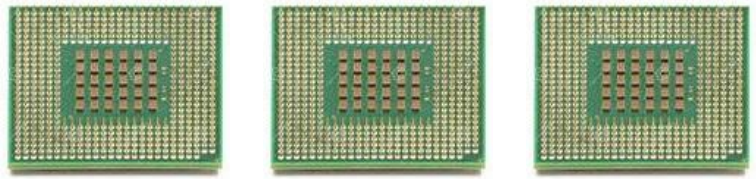


Data Loading

LOAD

Read → Process → Write

HDD

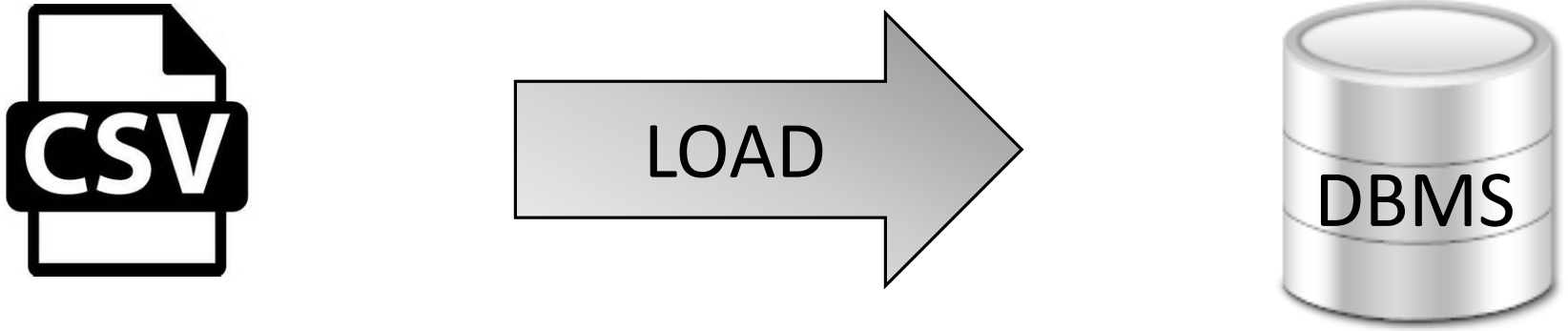


RAID-0 (24 disks)

Data Loading: general principles

1. **DON'T USE SLOW INSERT INTO** or prepared statements.
2. **USE BULK LOADING COPY INTO** in PostgreSQL, MonetDB, Vertica, parallel load via external tables in Oracle, bulk insert in SQLServer.
3. Ask you data provider for:
   a) **Dividing the raw data into many files (few GBs)** and load them in parallel - the fastest method to achieve good performance;
   b) **Data in a binary format** and a fast parser to process them;
4. **Clean your data:**
   a) PostgreSQL **ABORTS** loading when data is ill-formatted;
   b) Other databases store ill-formatted rows in *reject tables* or *error files* but all the well-formatted rows are loaded.

BULK Data Loading



How long does it take to load data into PostgreSQL?

Data Loading – current state of the art

TPC-H data 10GB, 2 sockets, 8 cores/socket, 2GHz, 20MB L3, 64GB RAM, from HDD->DAS, RHEL 6

Database	Loading time (seconds)
PostgreSQL	395.39
DBMS-B (commercial column-store)	234.94
MonetDB (open-source column-store)	149.11
DBMS-A (commercial row-store)	137.59

Data Loading – current state of the art

TPC-H data 10GB, 2 sockets, 8 cores/socket, 2GHz, 20MB L3, 64GB RAM, from HDD->DAS, RHEL 6

Database	Loading time (seconds)
PostgreSQL	395.39
DBMS-B (commercial column-store)	234.94
MonetDB (open-source column-store)	149.11
DBMS-A (commercial row-store)	137.59
PARALLEL loading to PostgreSQL (with a single reader)	**104.37**

How to improve the performance of data loading?

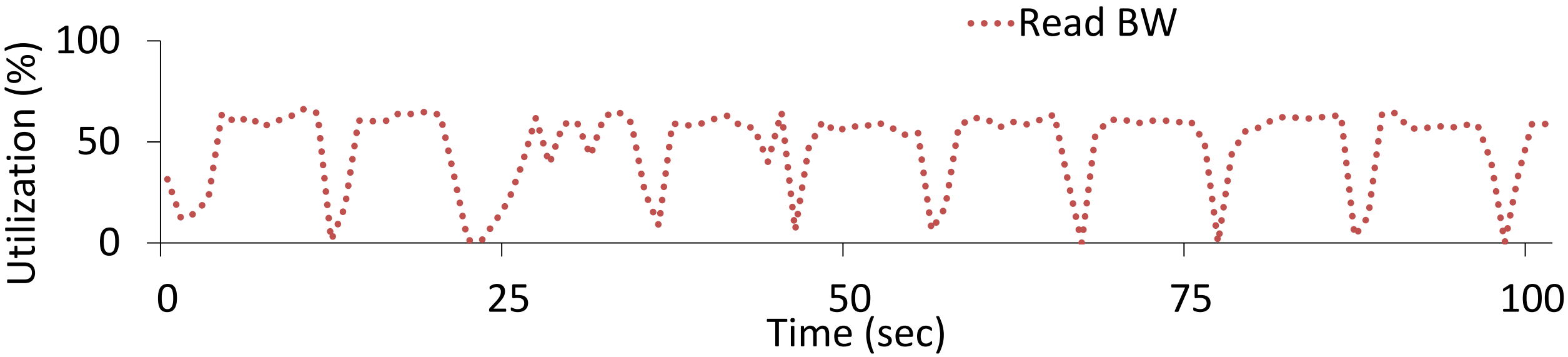
How to accelearate data loading and migration?

1. **Loading:**
   a) CSV vs. DBMS
   b) General principles
   c) Thread/Process Level **Parallelism**
   d) Identify Bottlenecks
   e) Using **Storage Devices**
   f) Data level parallelism (SIMD)
2. **Migration:**
   a) Leverage Diverse Databases
   b) CSV vs. Binary Migration

Resource utilization during **Data Loading**

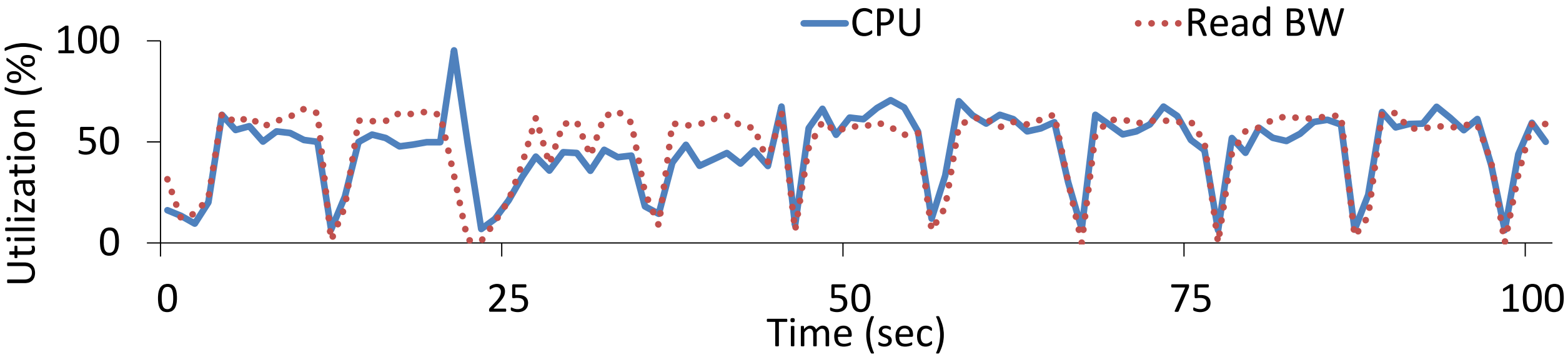
Read bandwidth underutilized

DBMS-A, TPC-H 10 GB from HDD to DAS, 16 threads, measured with sar (System Activity Reporting)



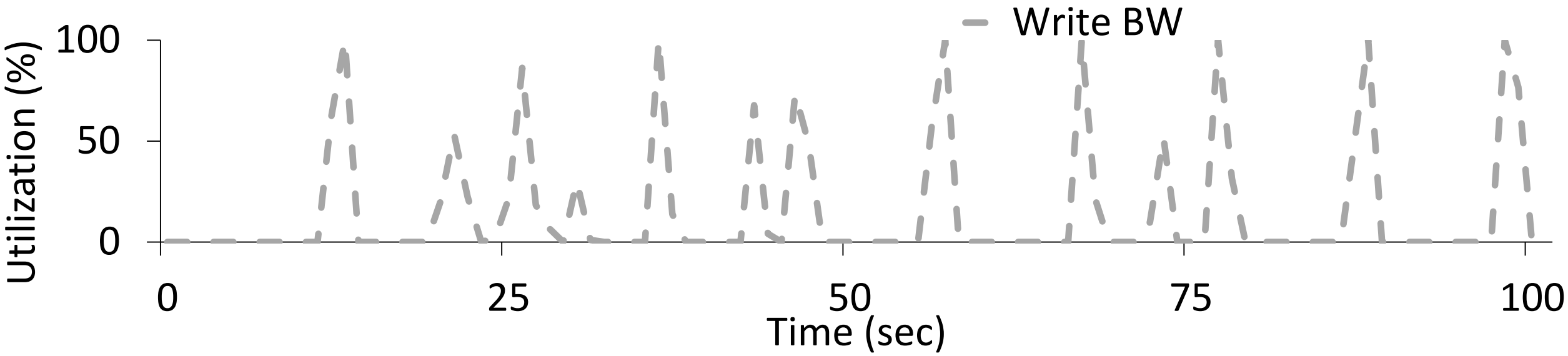
Low CPU utilization

DBMS-A, TPC-H 10 GB from HDD to DAS, 16 threads, measured with sar (System Activity Reporting)



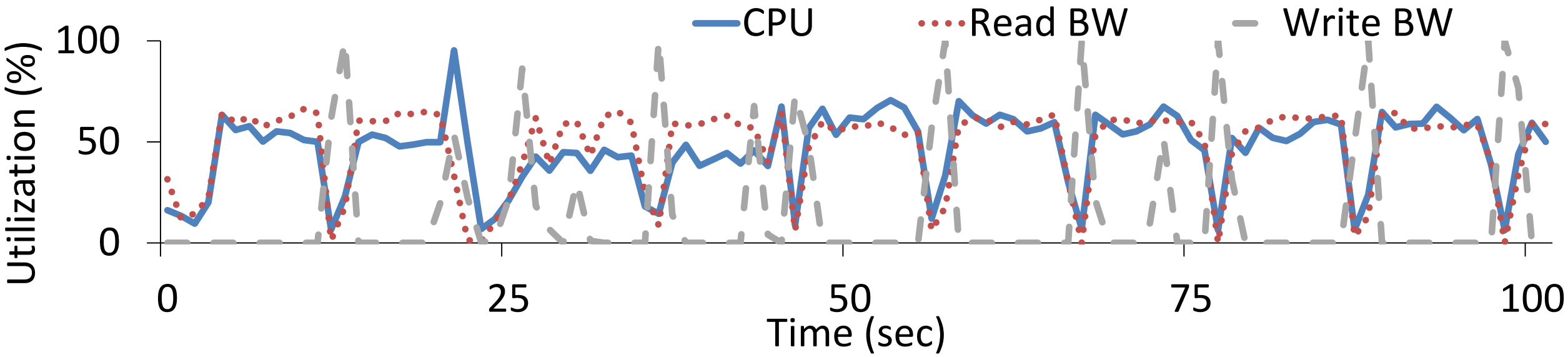
Write bandwidth underutilized

DBMS-A, TPC-H 10 GB from HDD to DAS, 16 threads, measured with sar (System Activity Reporting)



Alternating cycles in resource utilization

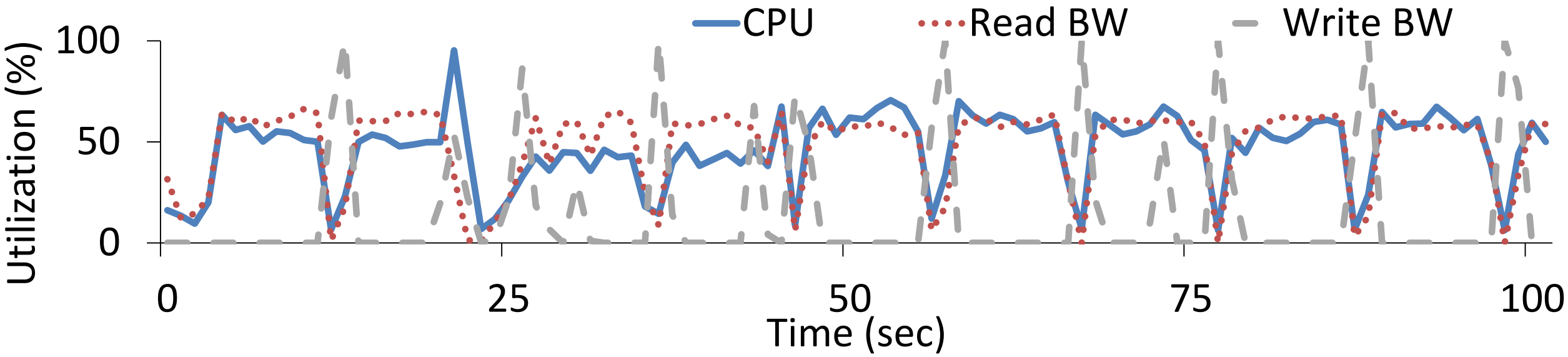
DBMS-A, TPC-H 10 GB from HDD to DAS, 16 threads, measured with sar (System Activity Reporting)



Unable to saturate resources

LOAD: Read → Process → Write

DBMS-A, TPC-H 10 GB from HDD to DAS, 16 threads, measured with sar (System Activity Reporting)

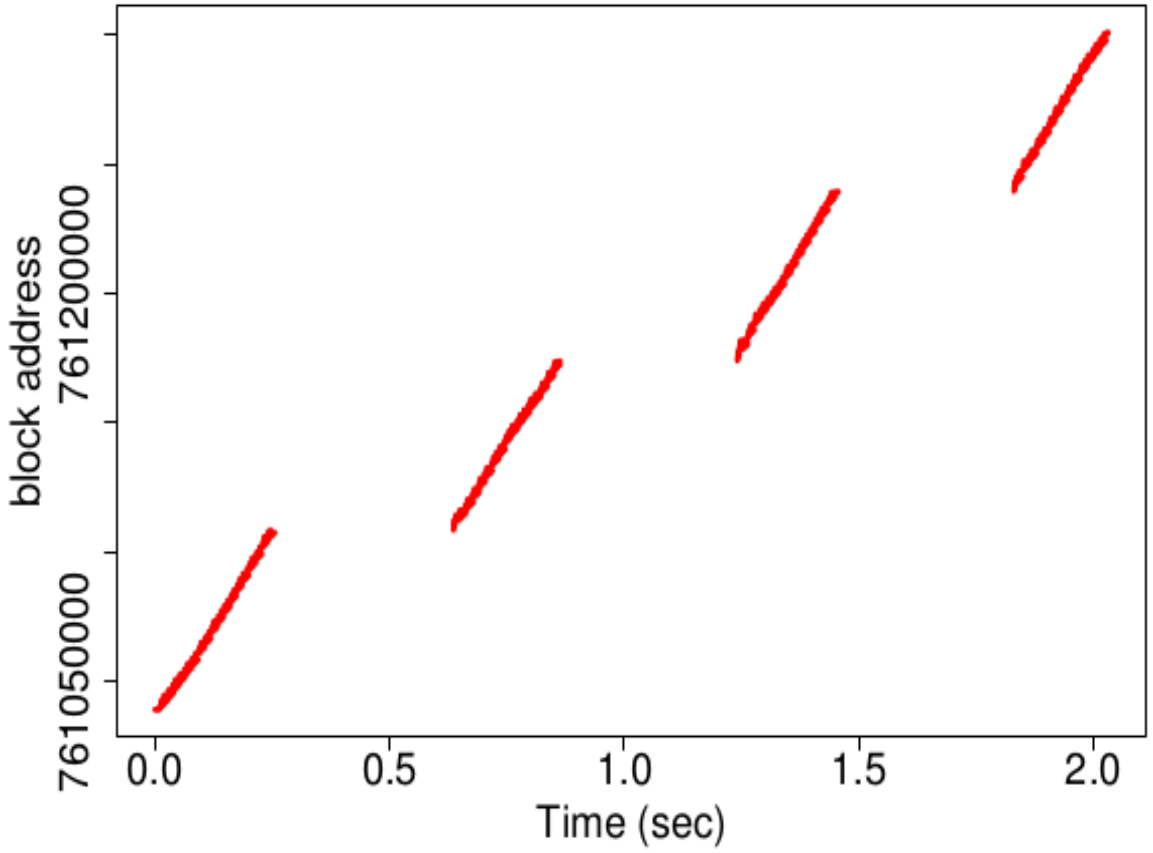


What causes the read bandwidth underutilization?

Orders TPC-H 10 GB ***HDD -> DAS****, 16 threads, iosnoop*

MonetDB

Sequential reader

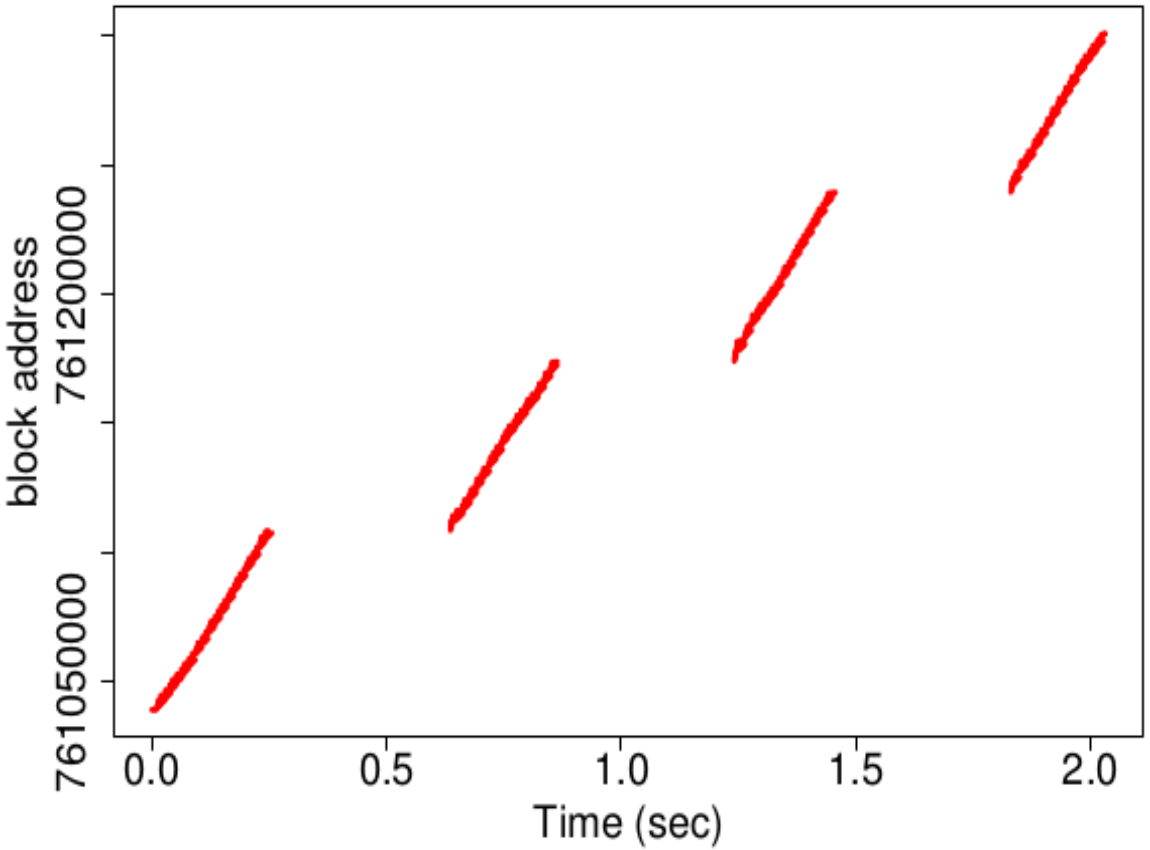


Read patterns

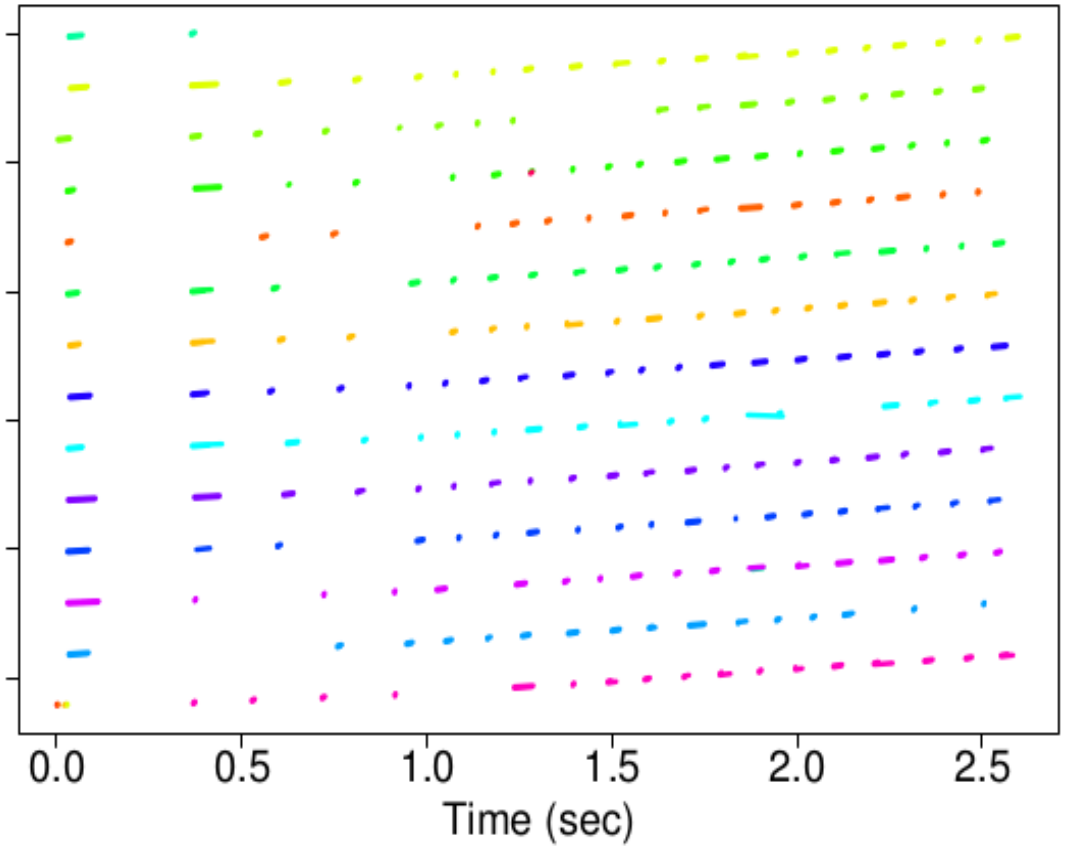
Read patterns

Reading orders file from TPC-H 10 GB;
***from HDD to DAS**, 16 threads*

MonetDB
Sequential reader



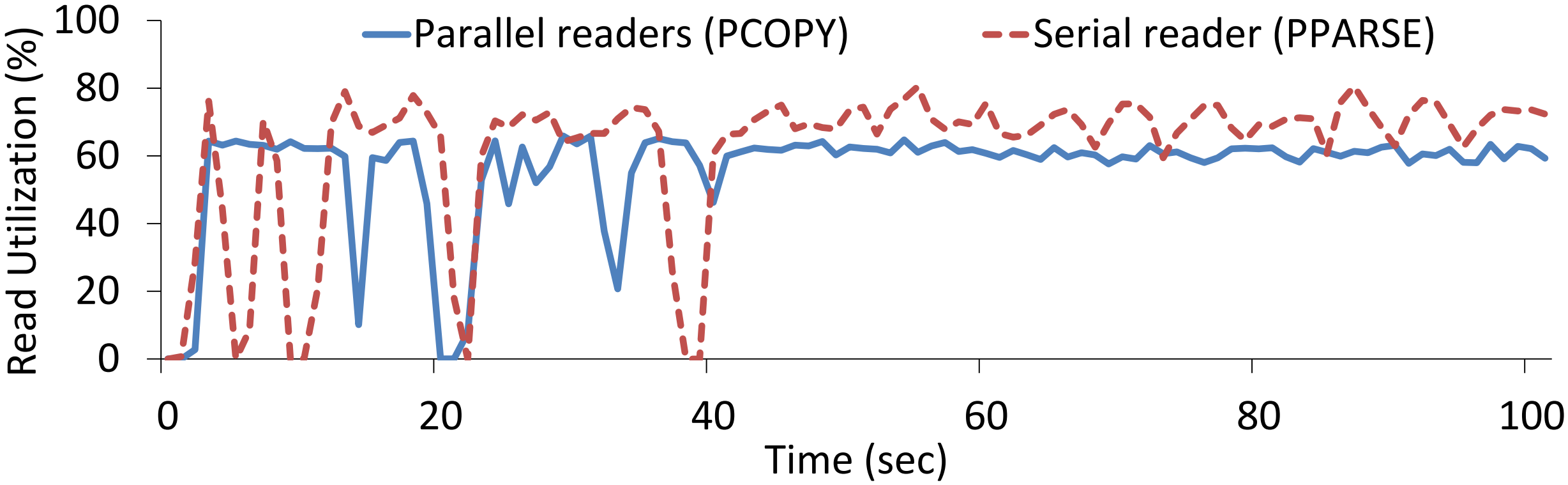
DBMS-A
Parallel readers



Random I/O causes underutilization

Storage aware PostgreSQL: serial vs. parallel reader

TPC-H 10 GB from HDD to DAS; loading times: PCOPY 111 sec vs. PPARSE 104 sec



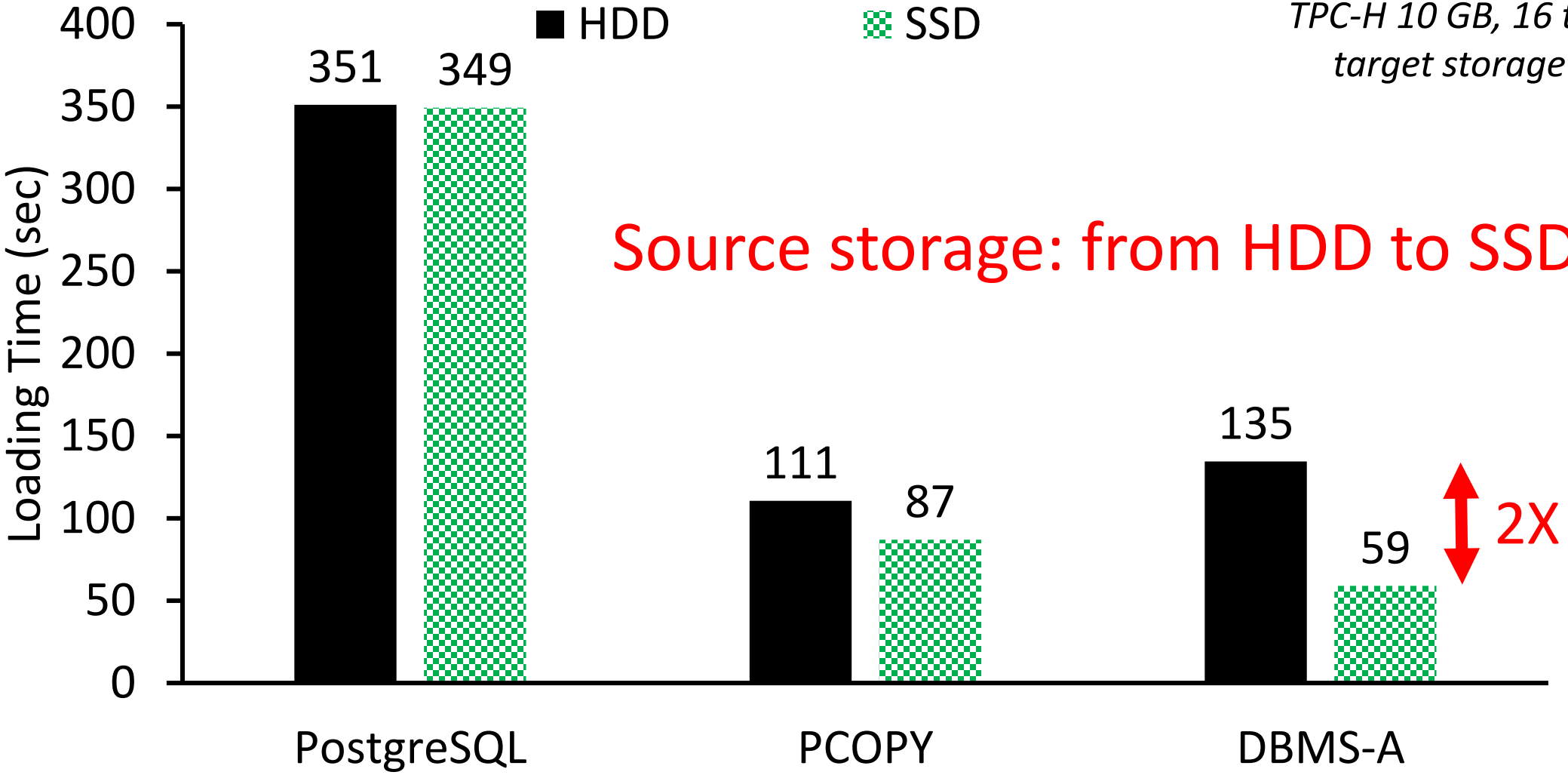
Serial reader improves read utilization

readers depends on input device

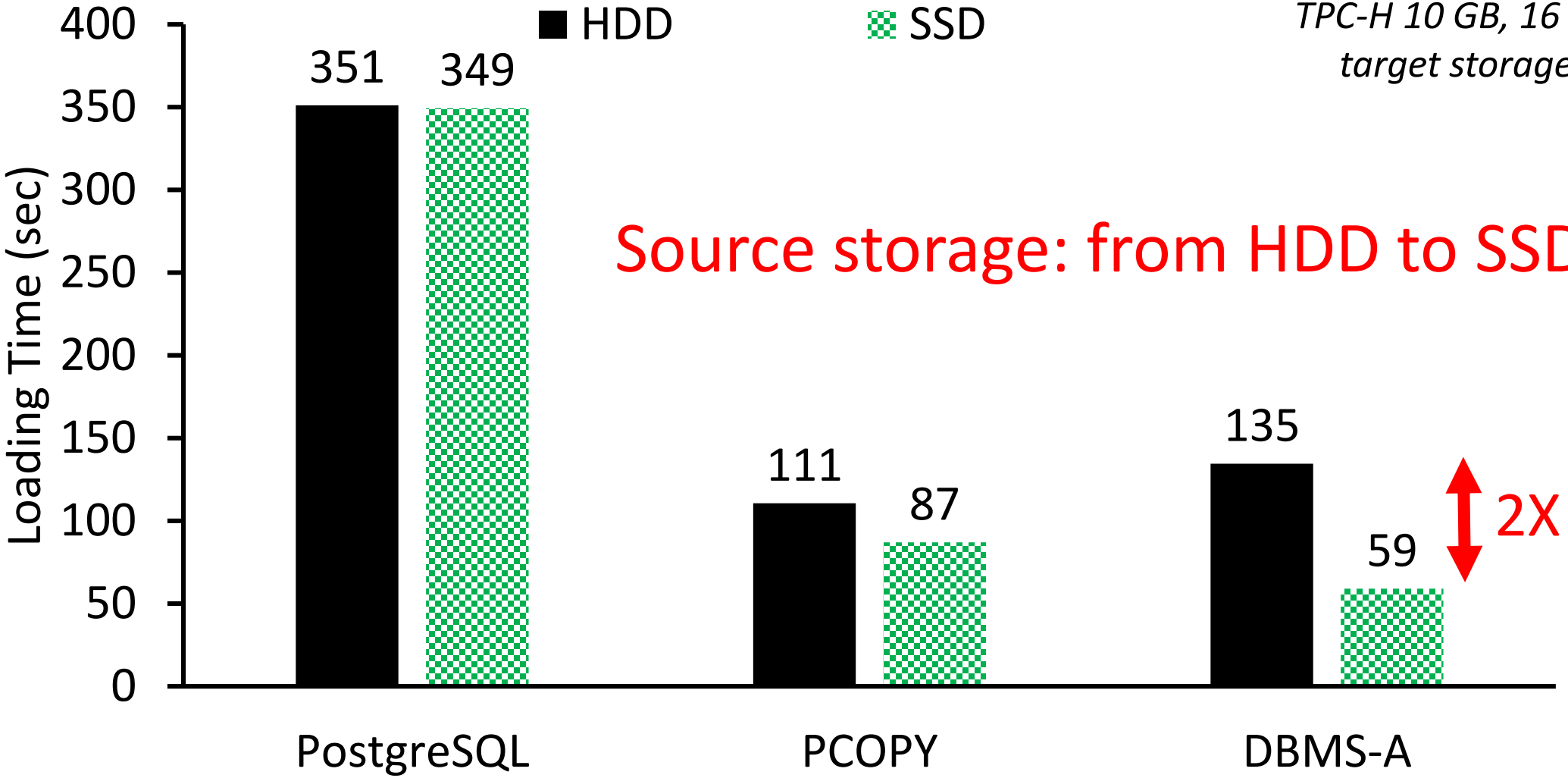
How to accelearate data loading and migration?

1. **Loading:**
   a) CSV vs. DBMS
   b) General principles
   c) Thread/Process Level **Parallelism**
   d) Identify Bottlenecks
   e) Using **Storage Devices**
   f) Data level parallelism (SIMD)
2. **Migration:**
   a) Leverage Diverse Databases
   b) CSV vs. Binary format

Storage device for parallel READERS



Storage device for parallel READERS

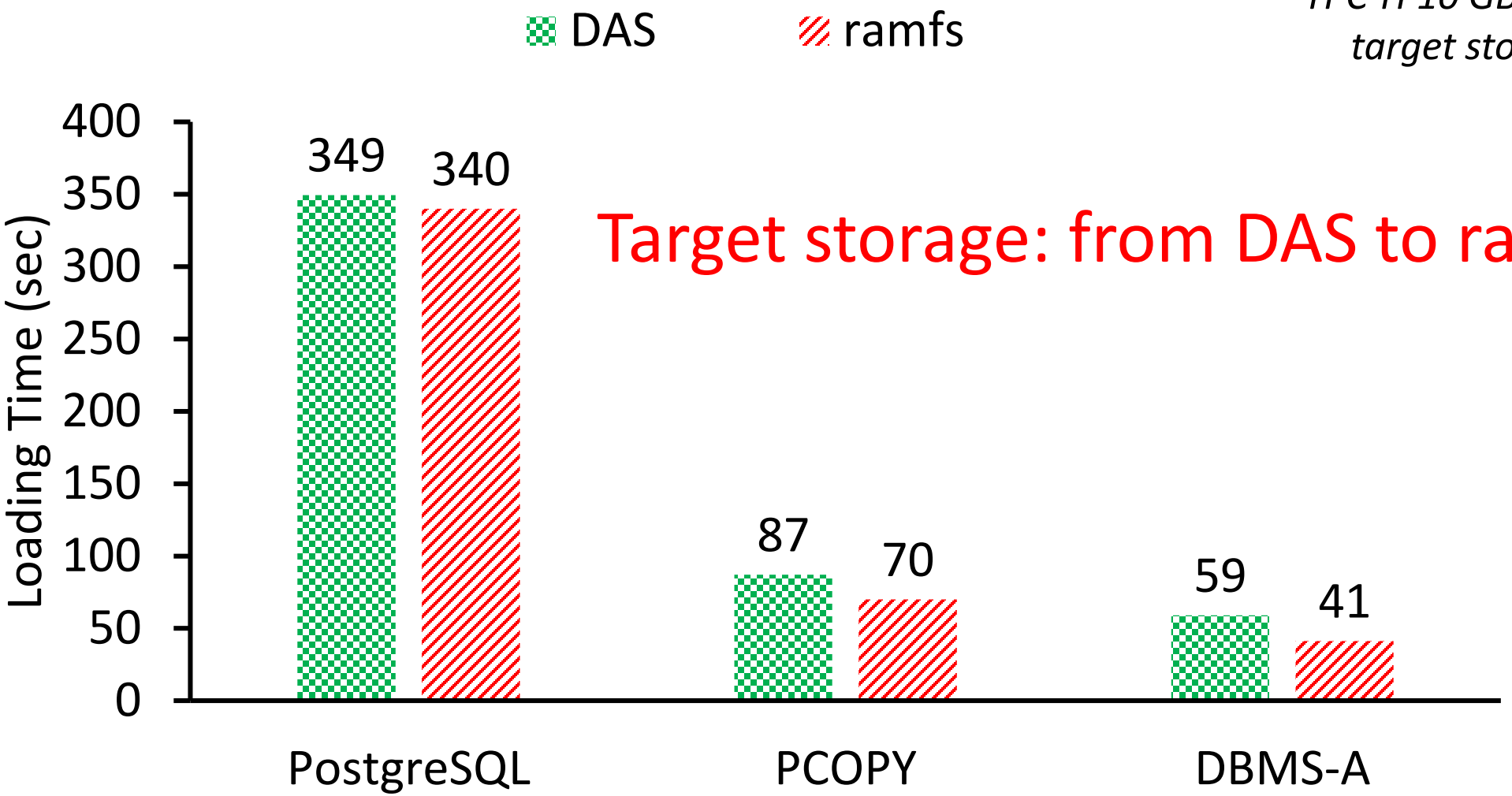


Source storage: from HDD to SSD

Single reader on HDD & Parallel readers on SSD

Storage device for WRITES

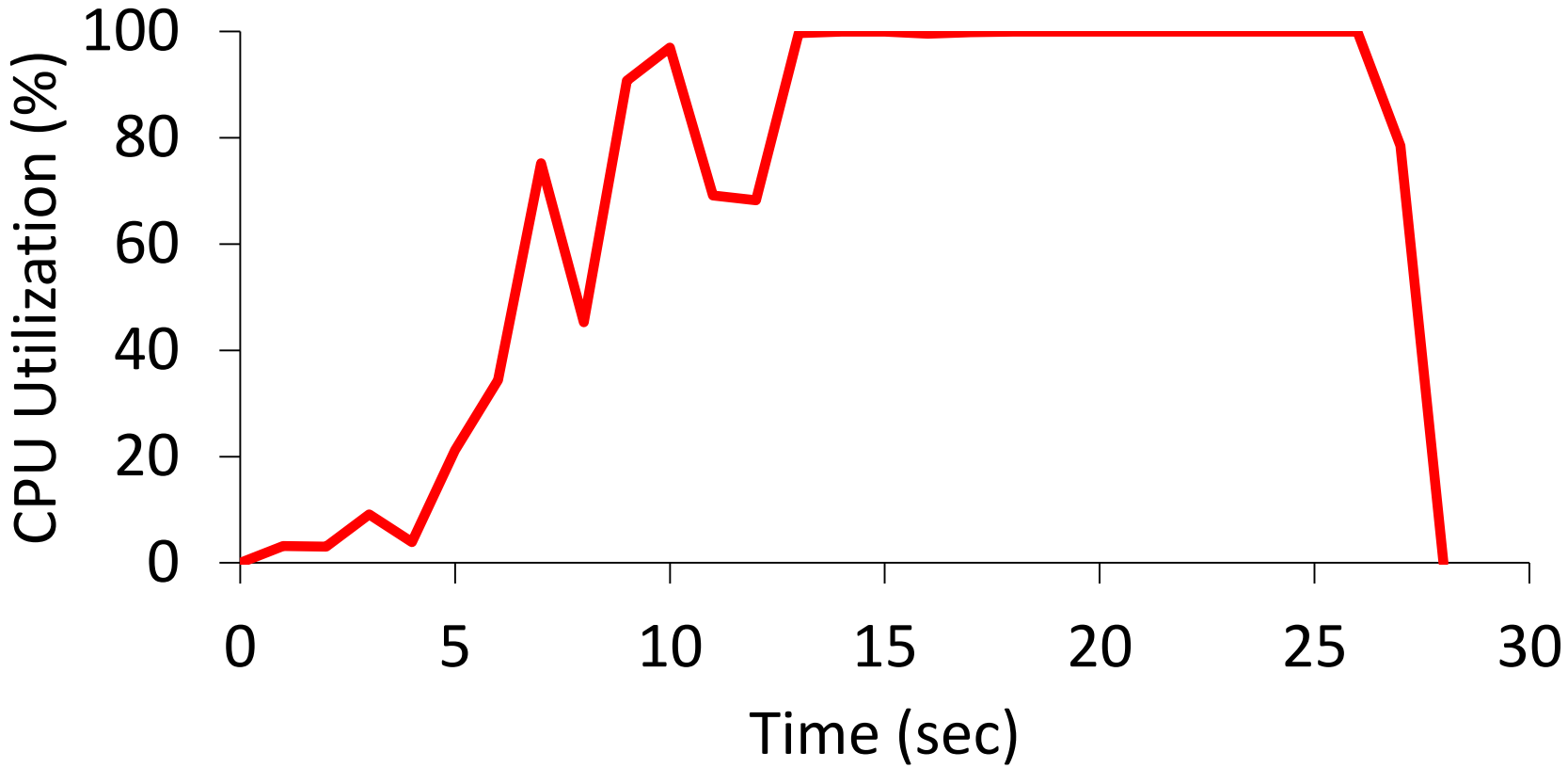
TPC-H 10 GB, 16 threads, target storage: DAS



Target storage: from DAS to ramfs

Best case storage scenario

DBMS-A, TCP-H 10 GB, from ramfs to ramfs, 16 threads

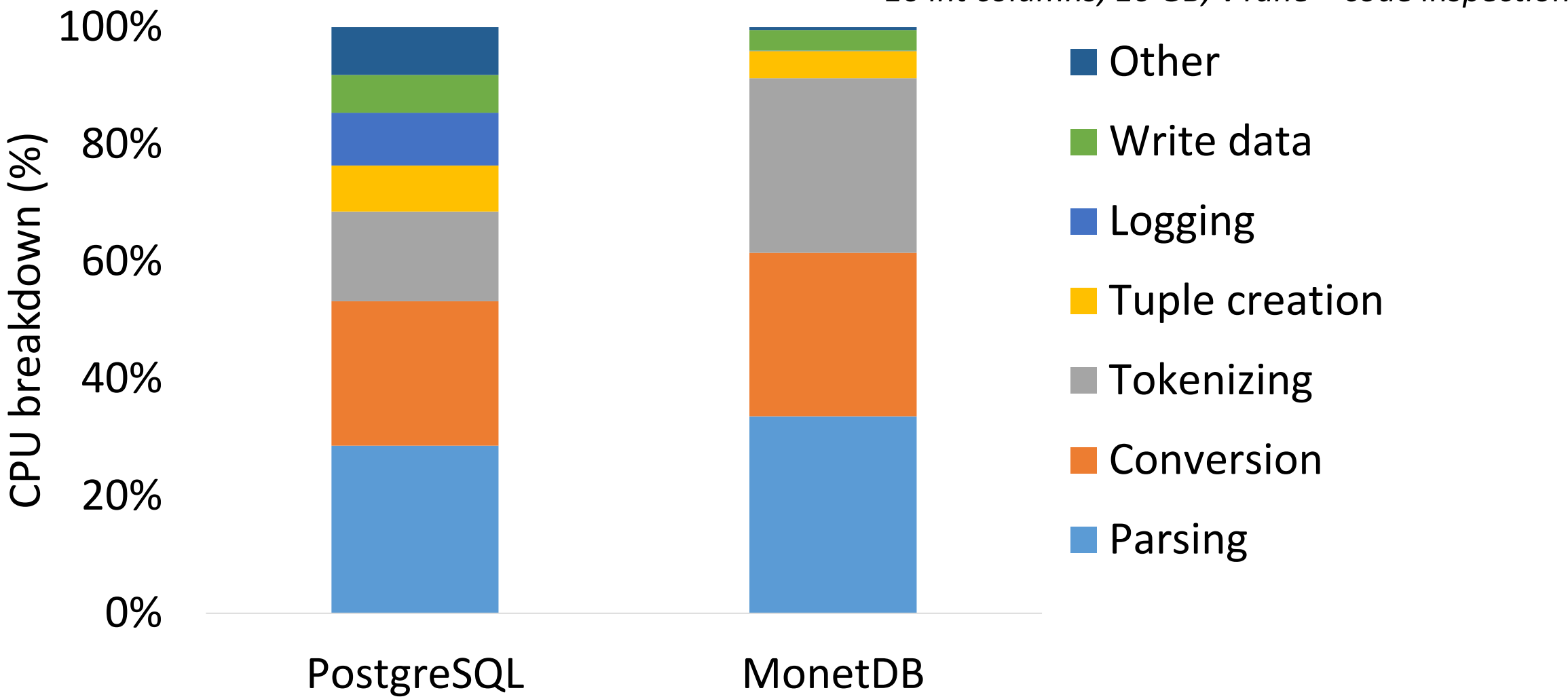


Device Bandwidth: 12.8 GB/sec

Read Rate: 250 MB/sec

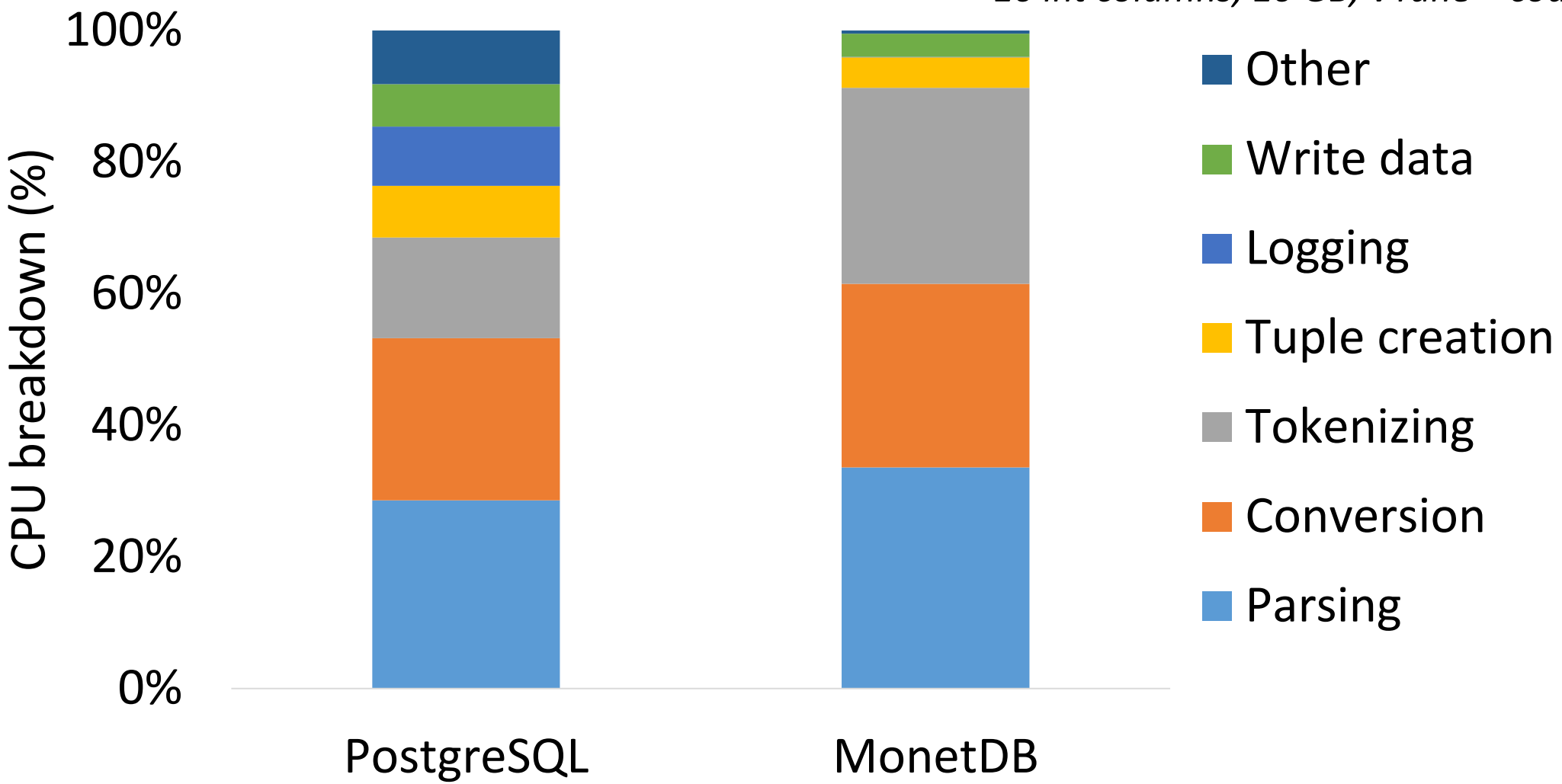
Where does the CPU time go?

10 int columns, 10 GB, VTune + code inspection



Where does the CPU time go?

10 int columns, 10 GB, VTune + code inspection

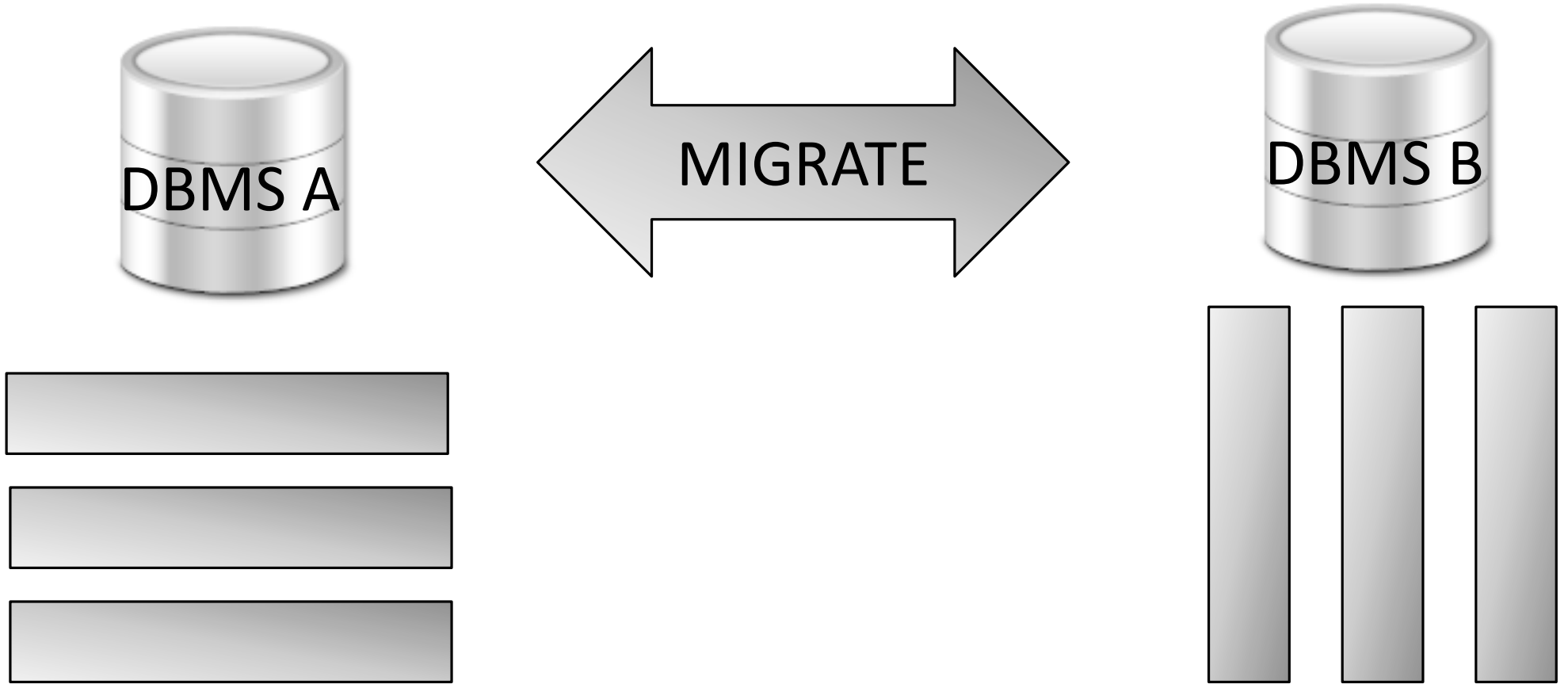


Use SIMD for faster parsing

How to accelearate data loading and migration?

1. **Loading:**
   a) CSV vs. DBMS
   b) General principles
   c) Thread/Process Level **Parallelism**
   d) Identify Bottlenecks
   e) Using **Storage Devices**
   f) Data level parallelism (SIMD)
2. **Migration:**
   a) Leverage Diverse Databases
   b) CSV vs. Binary format

Data Migration: leverge features of diverse databases



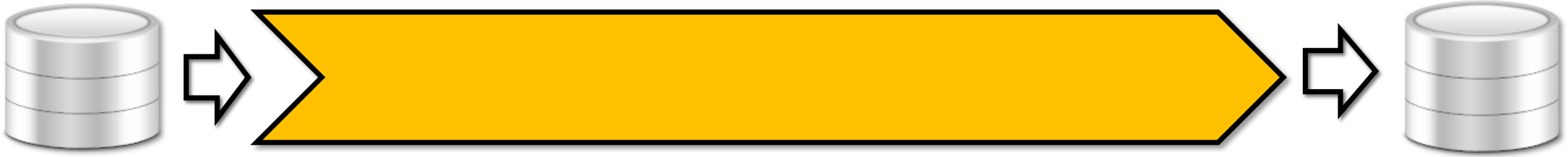
OLTP (row oriented)

Analytics (column oriented)

Current approach: CSV migration

CSV format

1,”Adam”,6.00; 2,”Aaron”,7.00

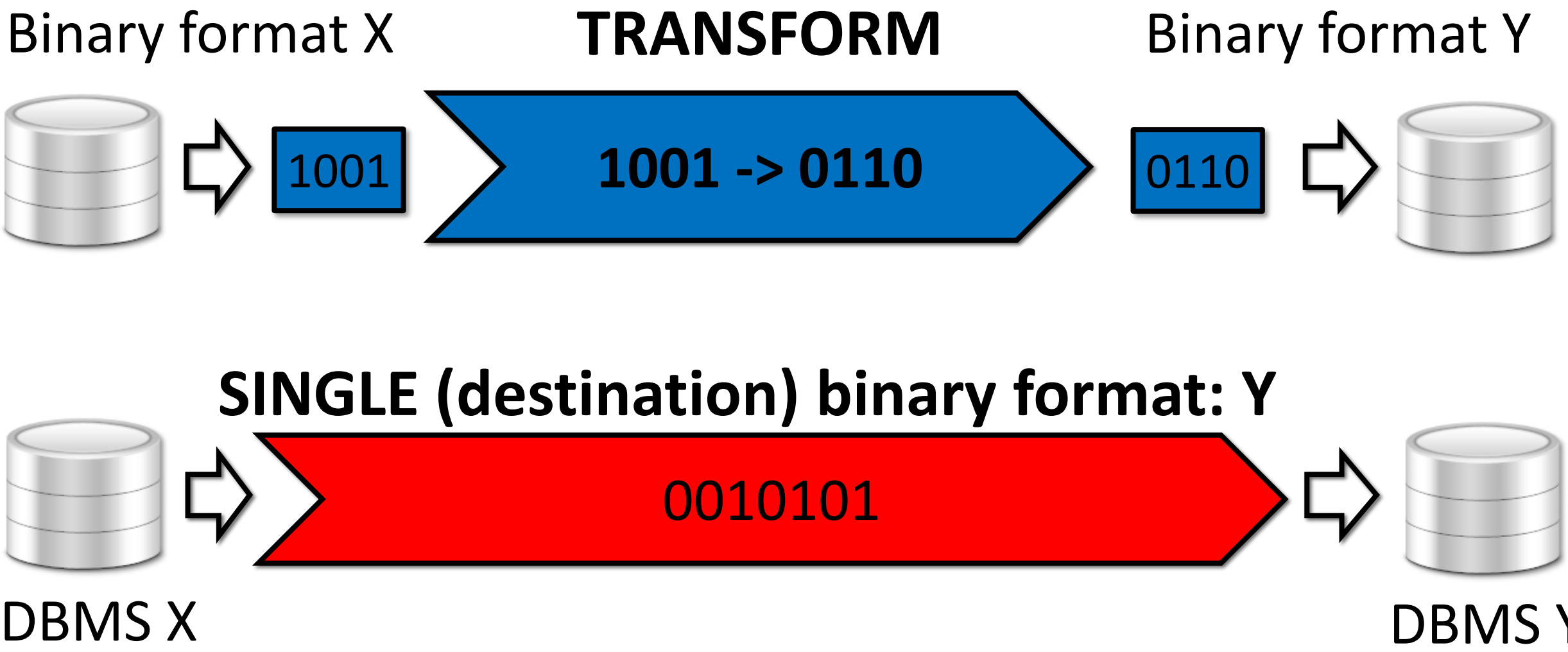


DBMS X

DBMS Y

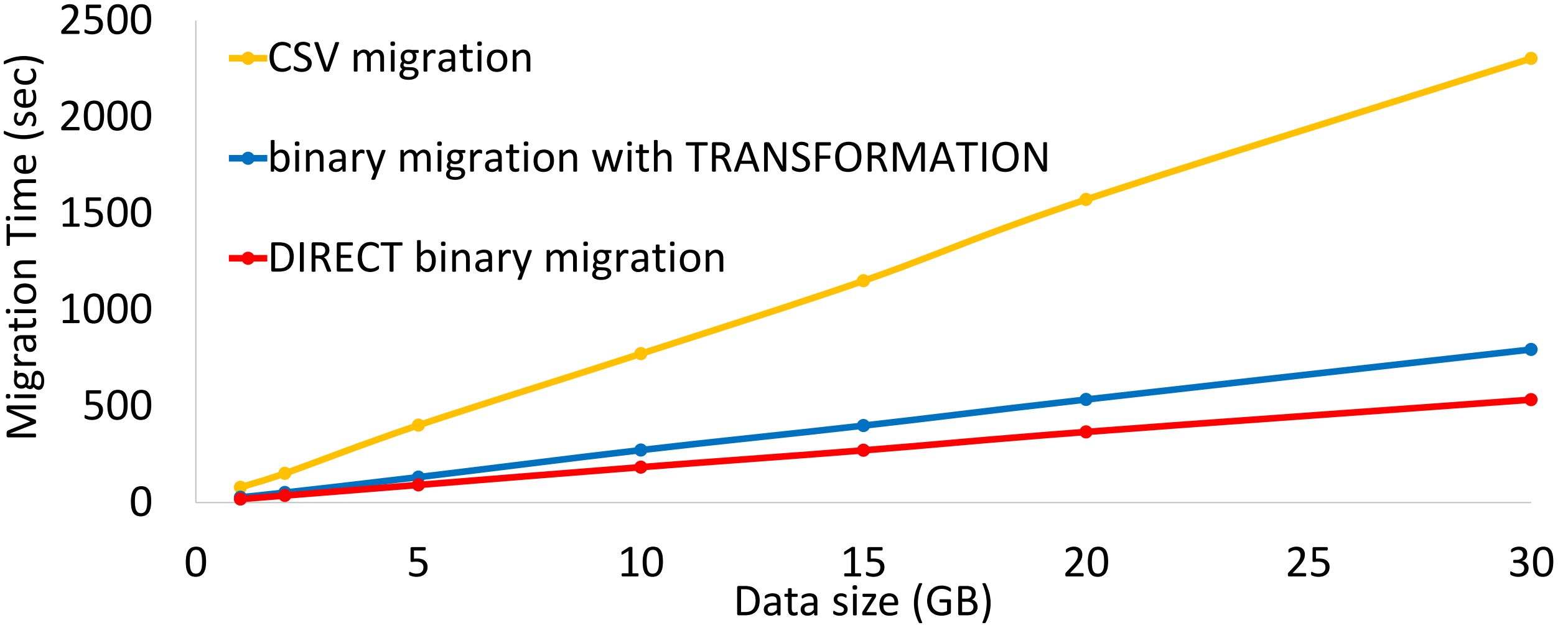
Data already loaded to the source database

Our approach: binary migration



Data Migration from PostgreSQL to SciDB

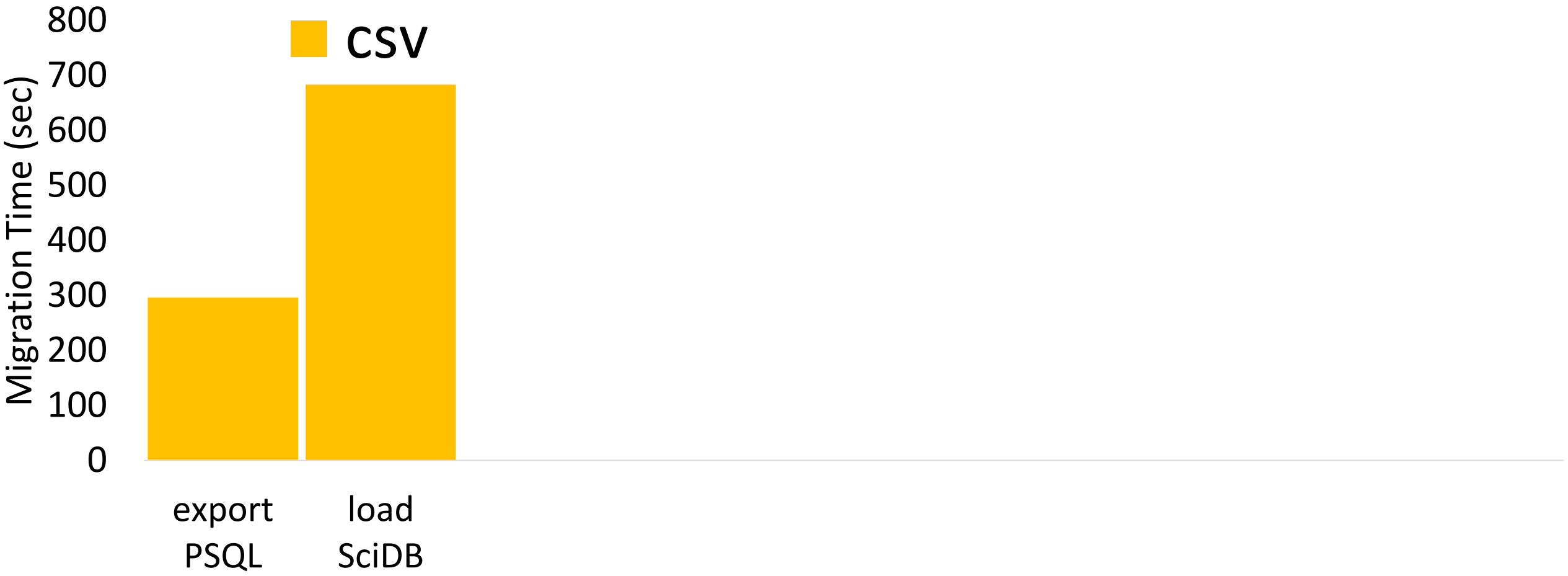
MIMIC II data - waveform(int, int, double)



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

Breakdown: migration from PostgreSQL to SciDB

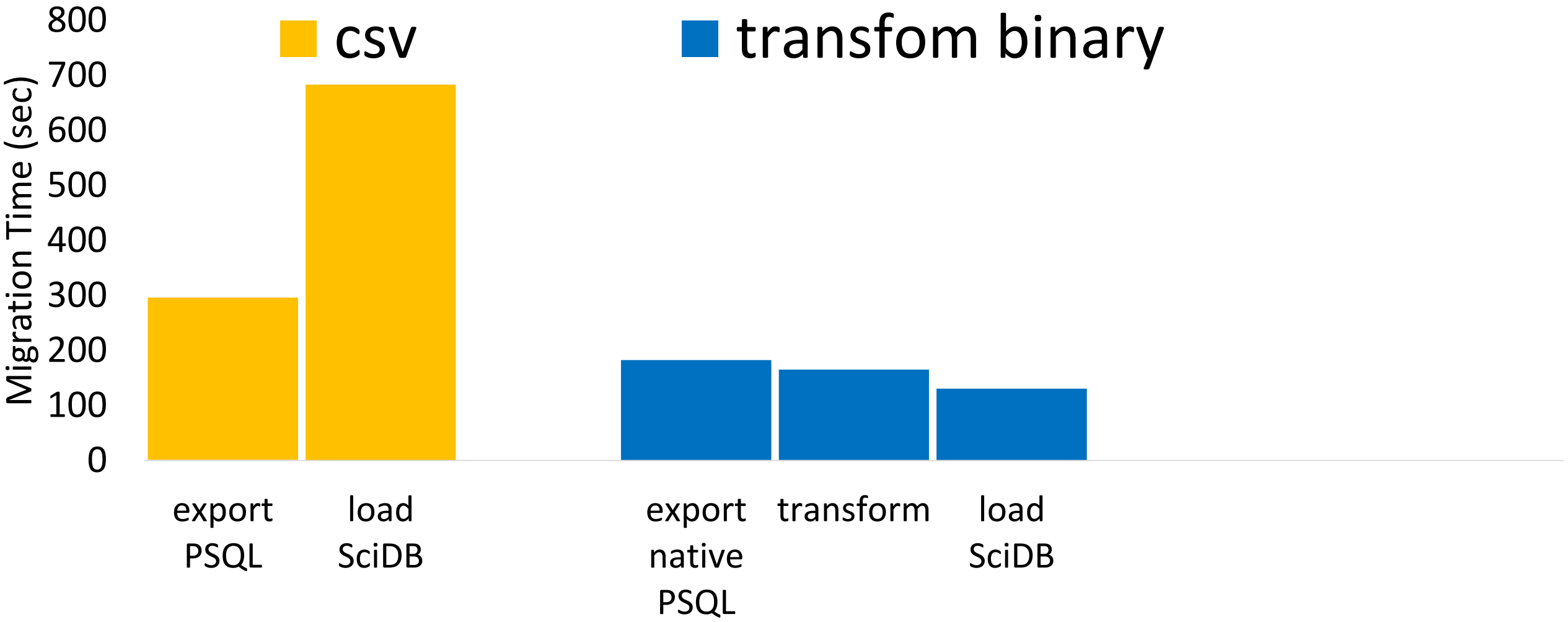
MIMIC II waveform data (int, int, double) 10 GB



Slow CSV loading

Breakdown: migration from PostgreSQL to SciDB

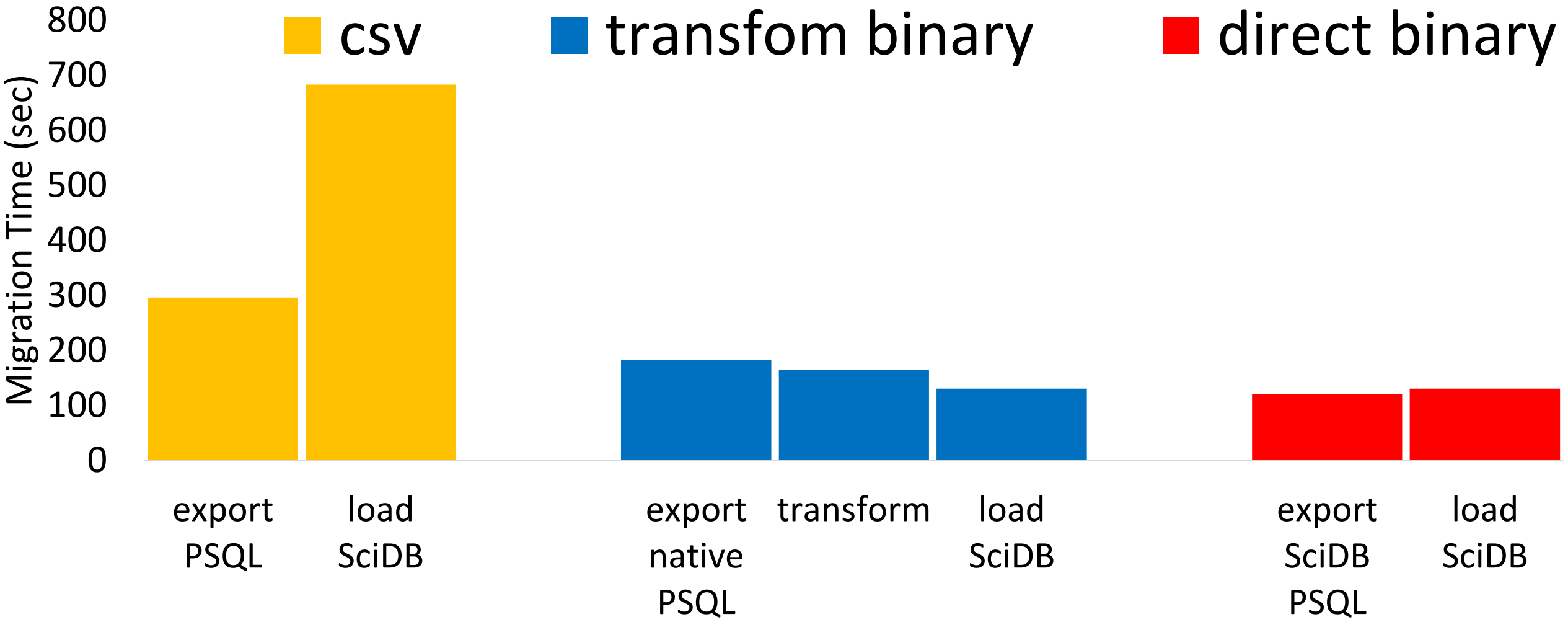
MIMIC II waveform data (int, int, double) 10 GB



Binary Export SLOWER than Binary Loading

Breakdown: migration from PostgreSQL to SciDB

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



Use Fast Direct Binary Migration

Conclusions

- **Parallelism** is the key:
  - Ask the data provider for **splitting the data** during generation and **BULK load** the data files in parallel;
  - Tune your database and set up **parallel loading**;
  - Use SIMD (data level parallelism) for faster parsing.
- **Optimize I/O to fully utilize CPU:**
  - Input I/O path with single (HDD) or parallel reader (SSD);
  - Output I/O path to reduce pauses caused by data flushes.
- **Binary data format** for data transfers:
  - Select a single, concise, SIMD-friendly & **binary format**.

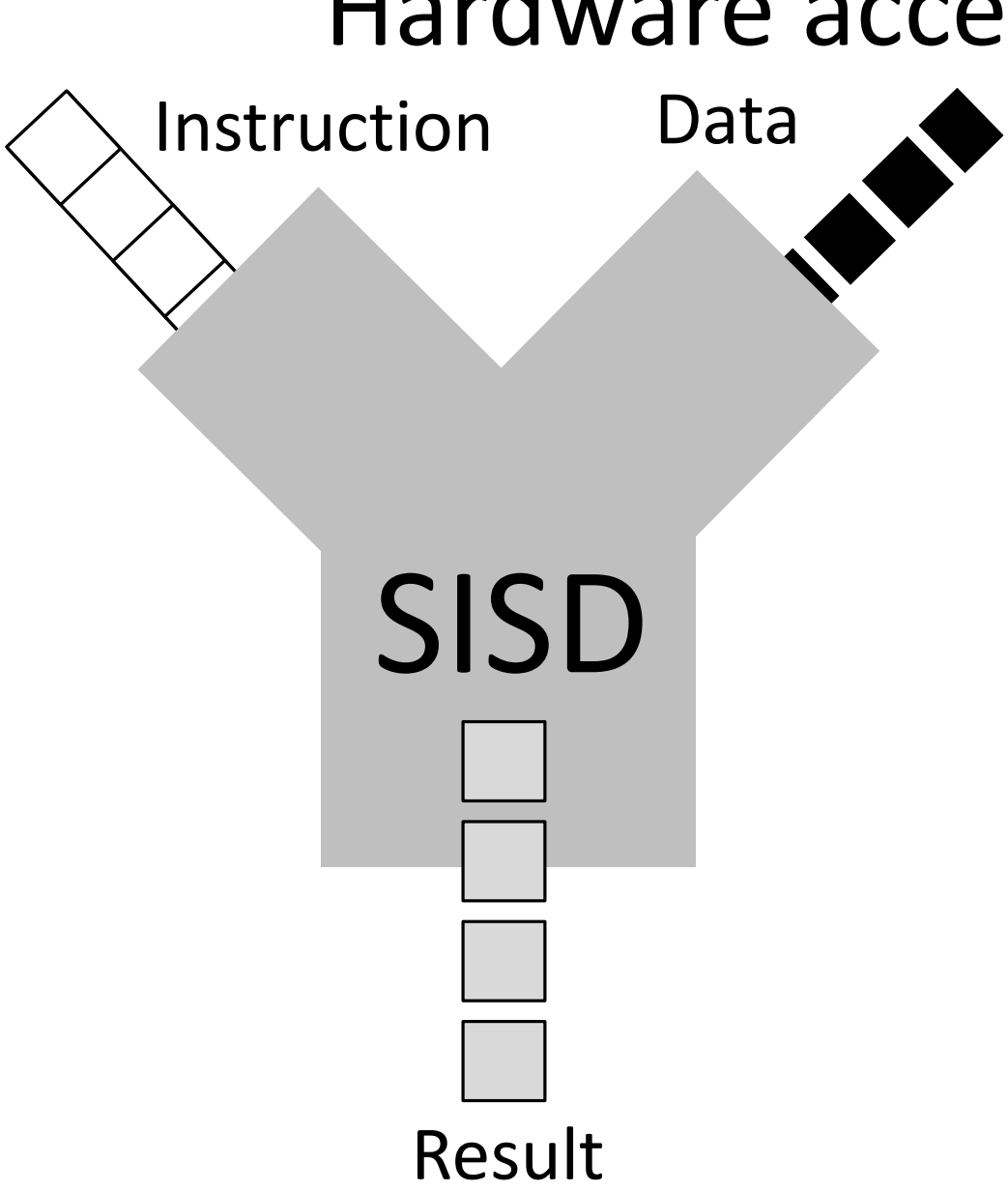
Thank you

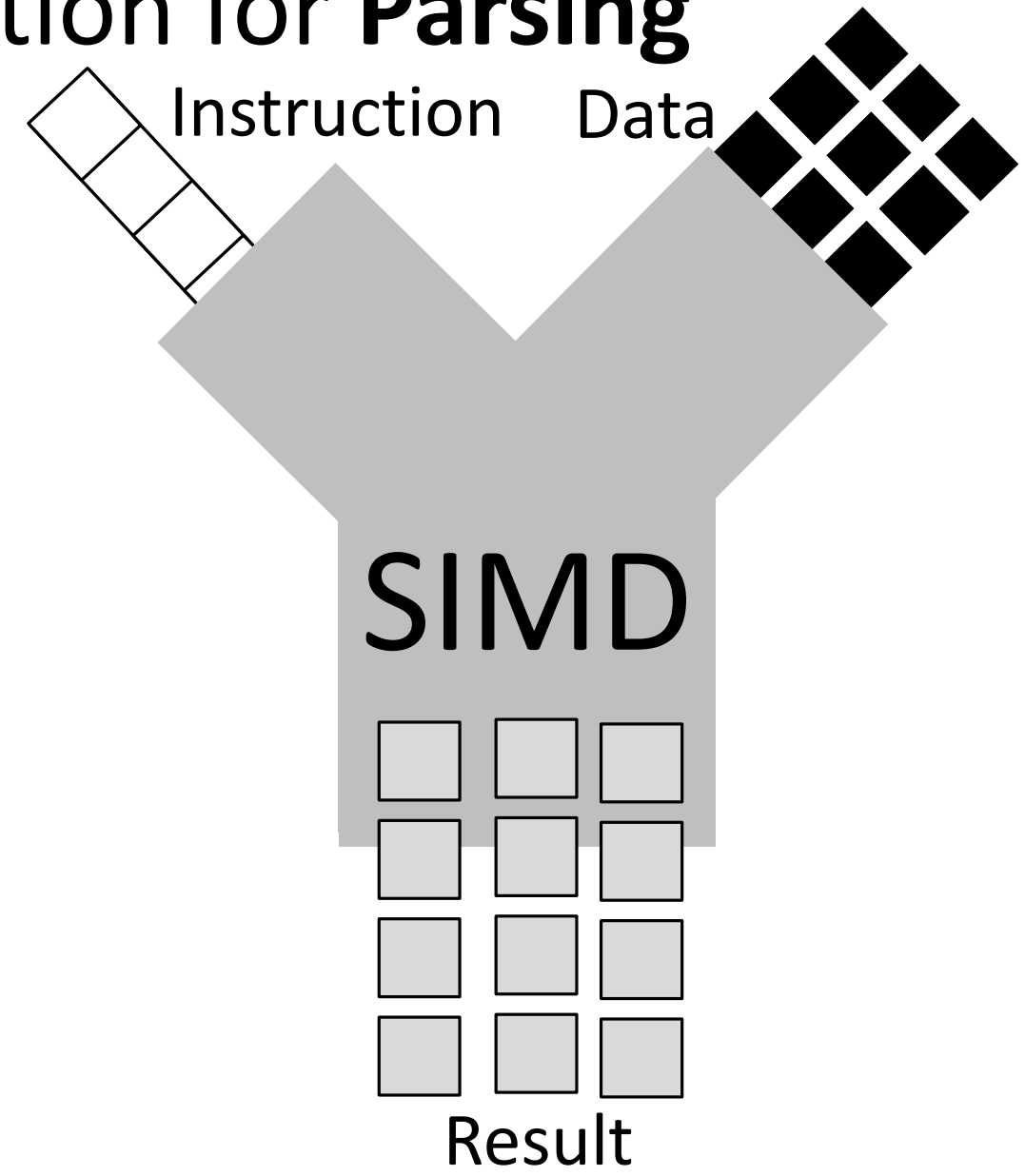
Backup slides

Faster loading, export & migration for PostgreSQL

- **Load** - external loader for PostgreSQL:
  - Read data from many input files using many psql clients;
  - External parallel loader that is aware of storage devices.
- **Export**:
  - Both, parallel export in CSV format and export in a specific (non-postgres) binary format require changes to the PostgreSQL's source code.
- **Migrate** = export + load:
  - CSV: no changes to the database required (unless parallel);
  - Binary: either external transformation or internal changes.

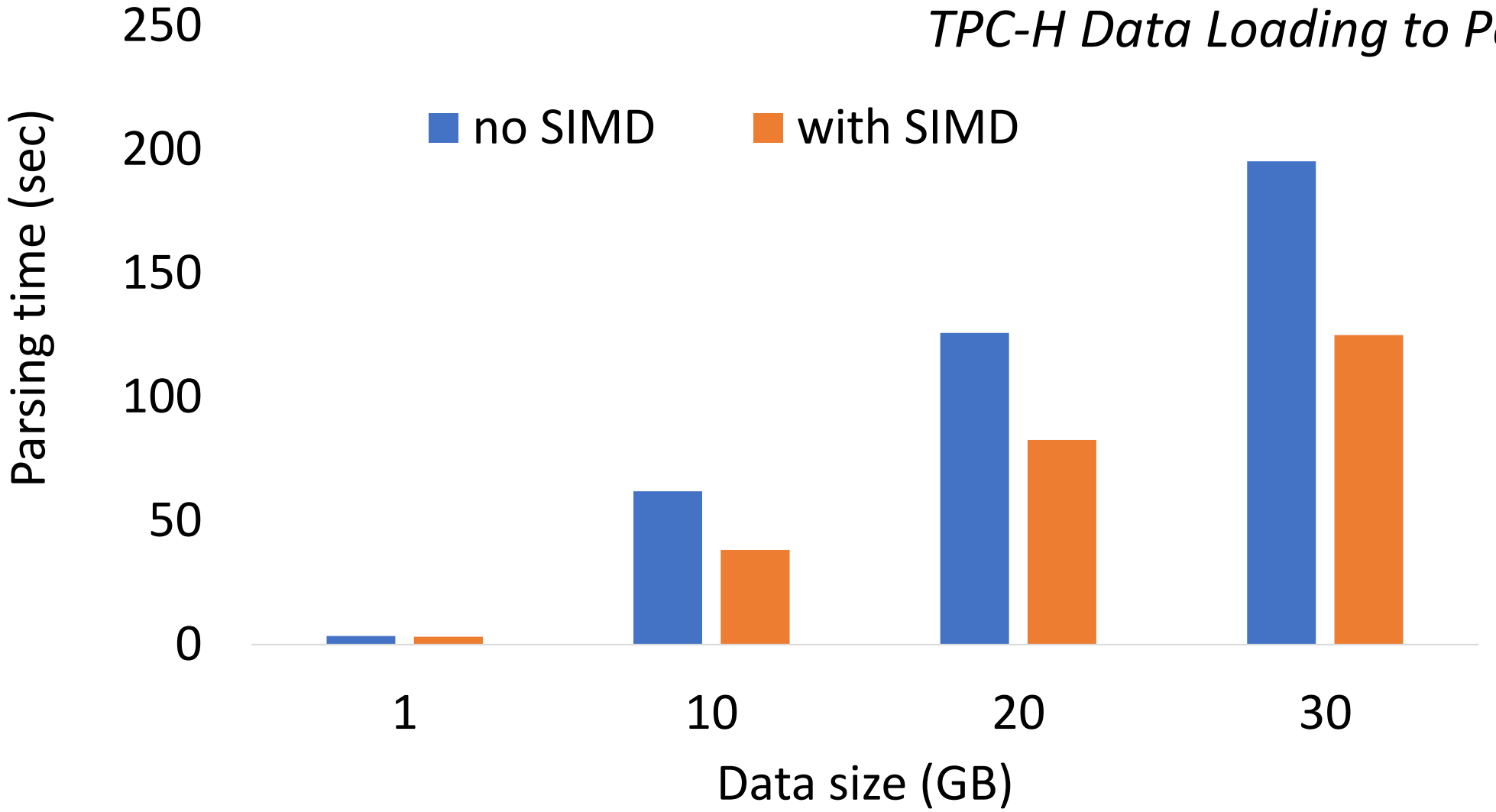
Hardware acceleration for **Parsing**





SIMD: for parsing lines in input CSV file

TPC-H Data Loading to PostgreSQL

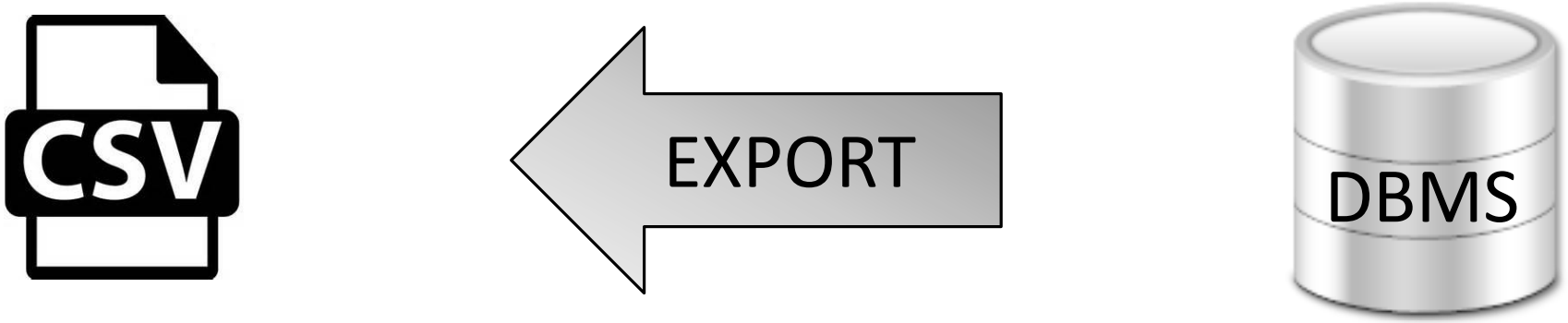


SIMD gives 1.6X speedup for parsing lines

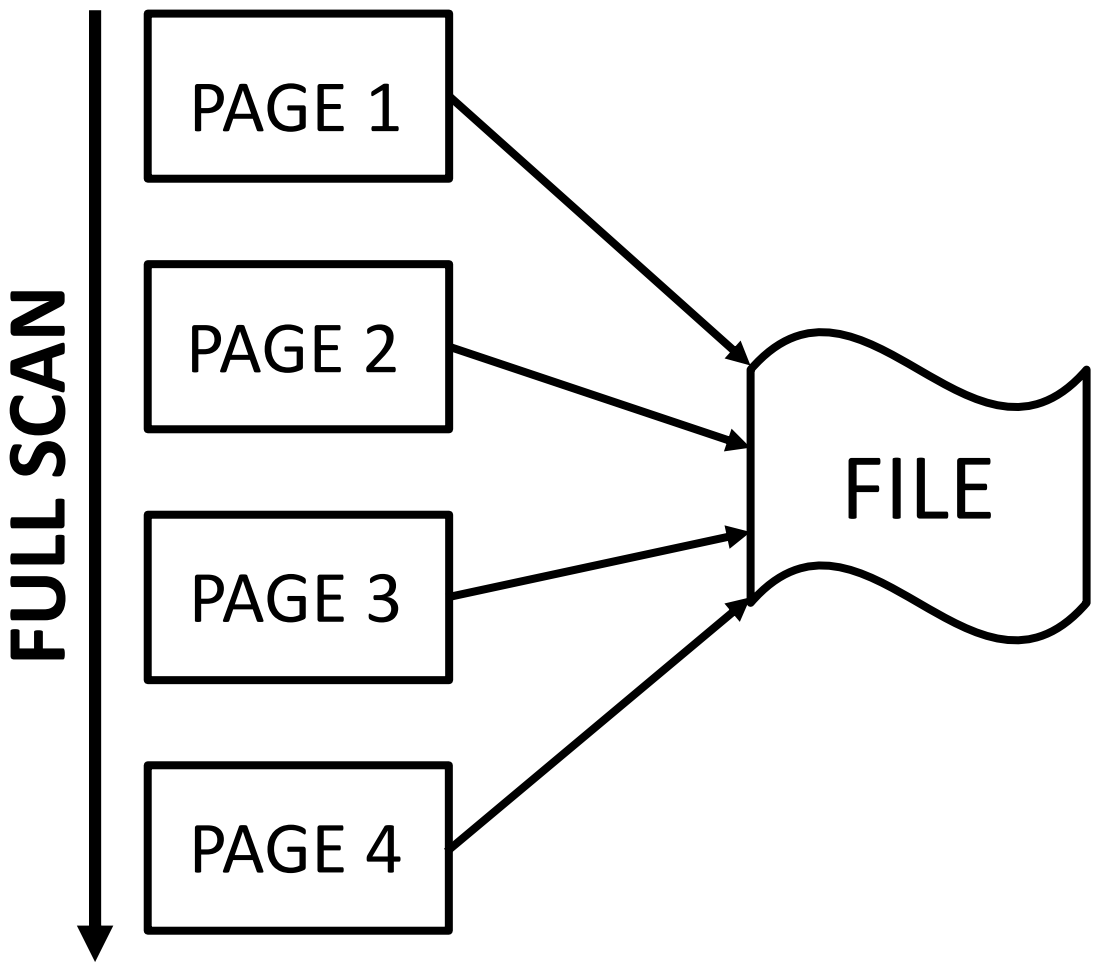
How to accelearate data loading, export and migration?

1. **Loading:**
   a) Intro: CSV vs. DBMS, and basic recommendations;
   b) Thread/process level **parallelism**;
   c) Identify the bottlenecks;
   d) Impact of storage devices;
   e) Data level parallelism (hardware accelerator: SIMD).
2. **Export:**
   a) Extract database pages (collection of rows) to separate files with many processes.
3. **Migration:**
   a) Single, concise, and binary data format for direct data migration.

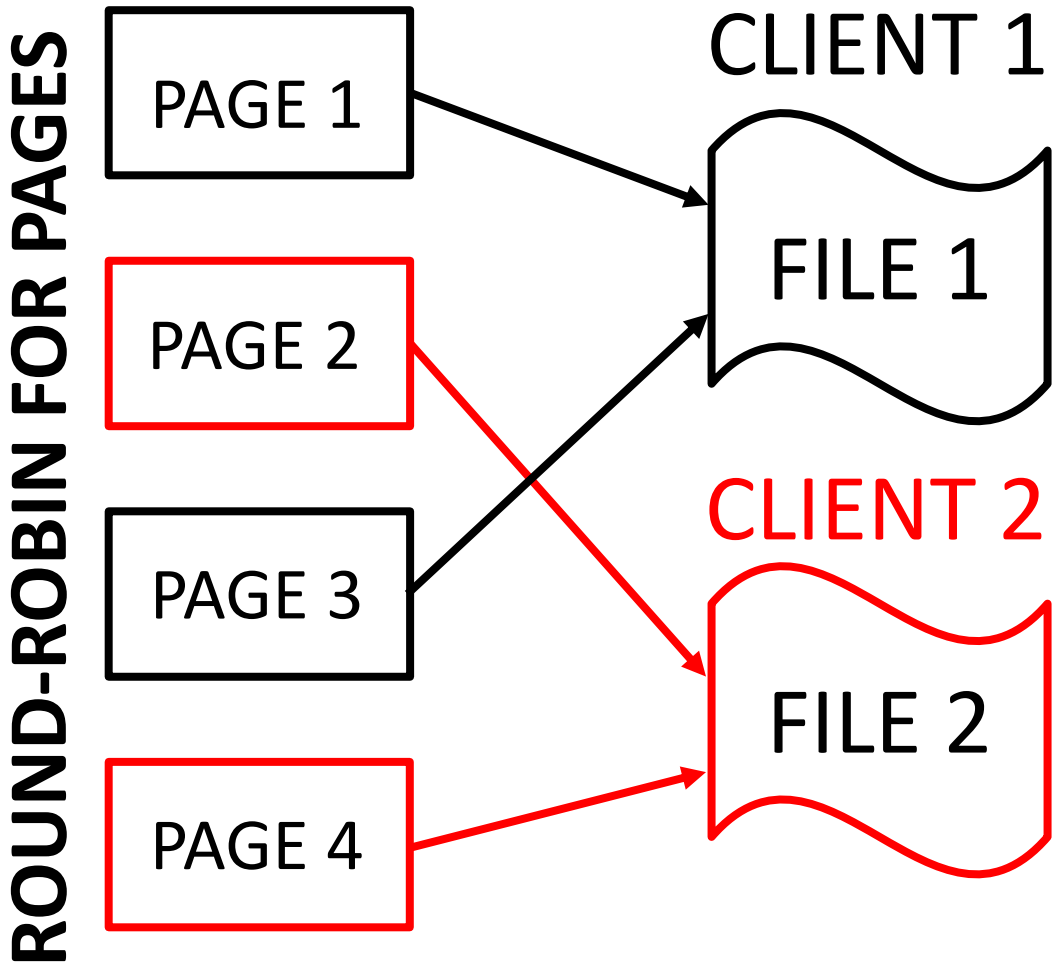
Data Export: don’t get **locked** into a **database vendor**



Parallel Export from PostgreSQL



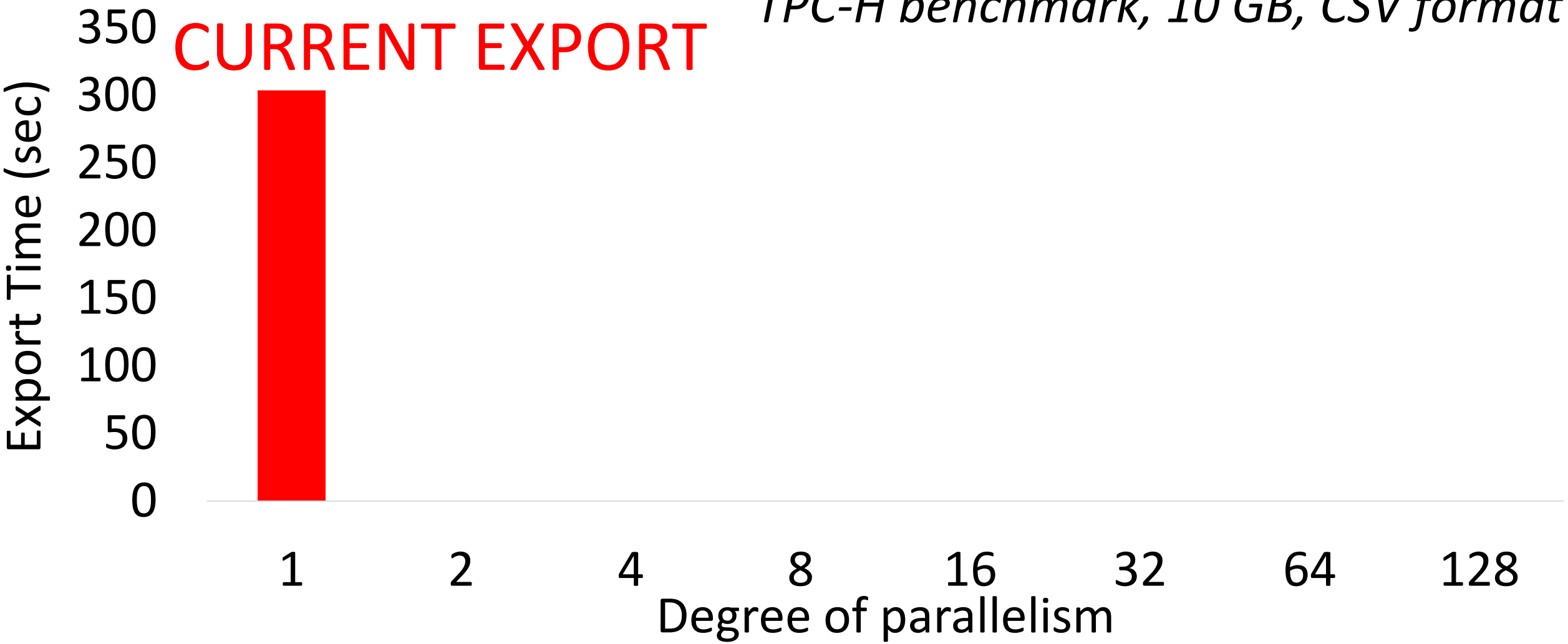
Current single-threaded export



New parallel export

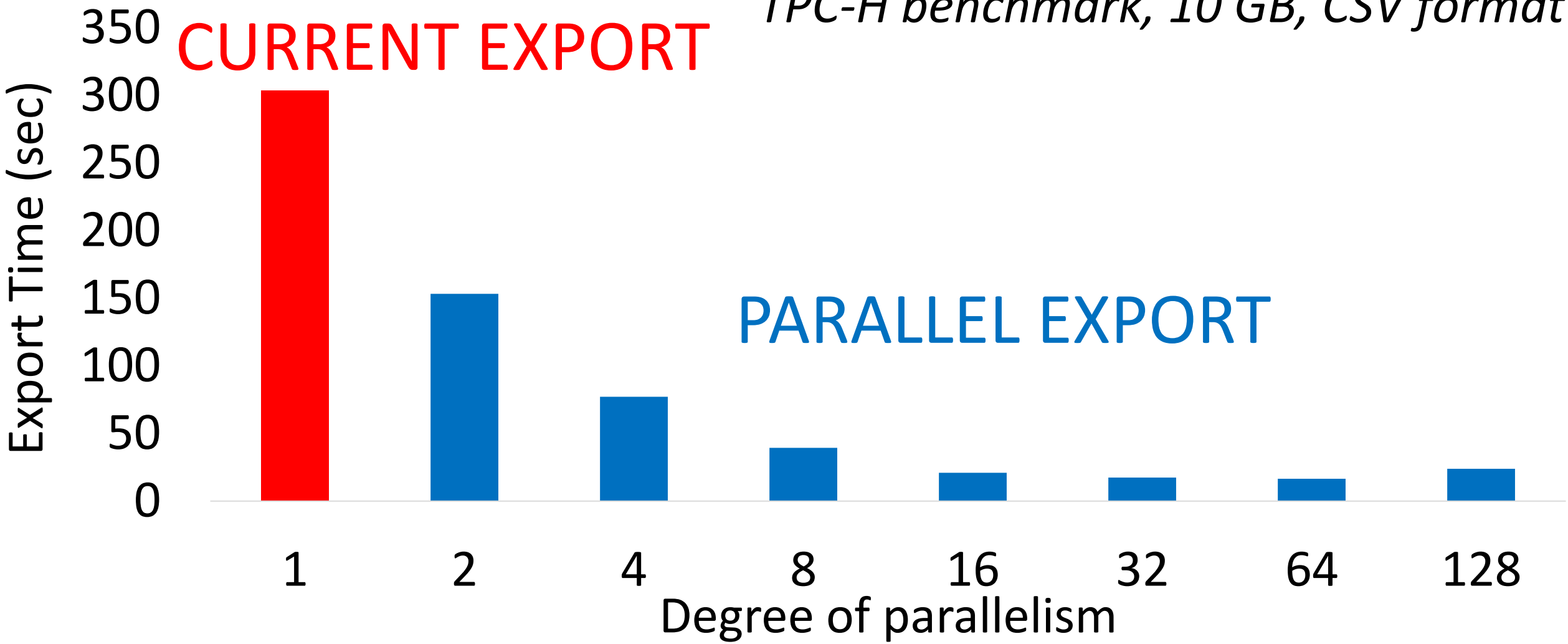
Parallel export from PostgreSQL

TPC-H benchmark, 10 GB, CSV format



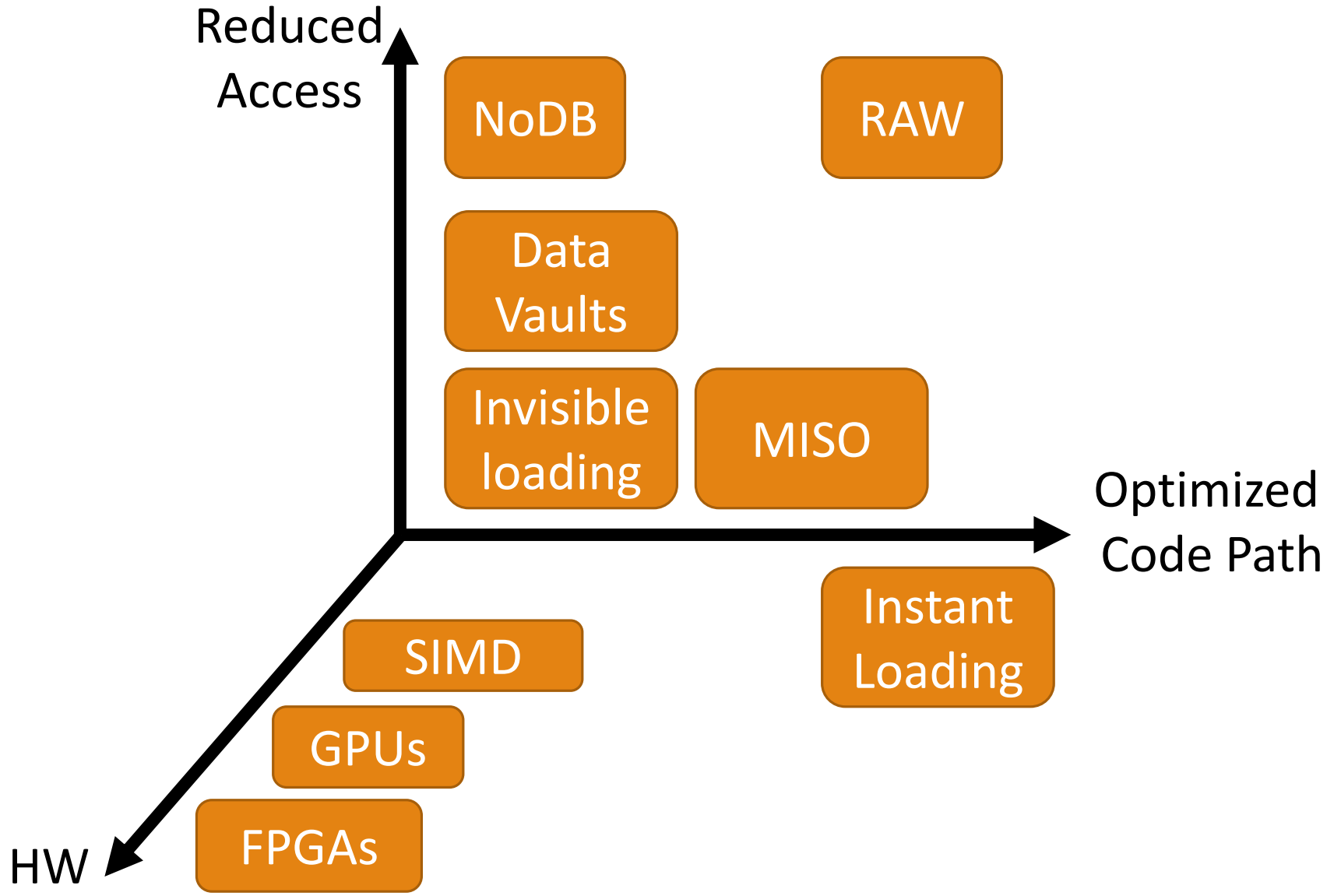
Parallel export from PostgreSQL

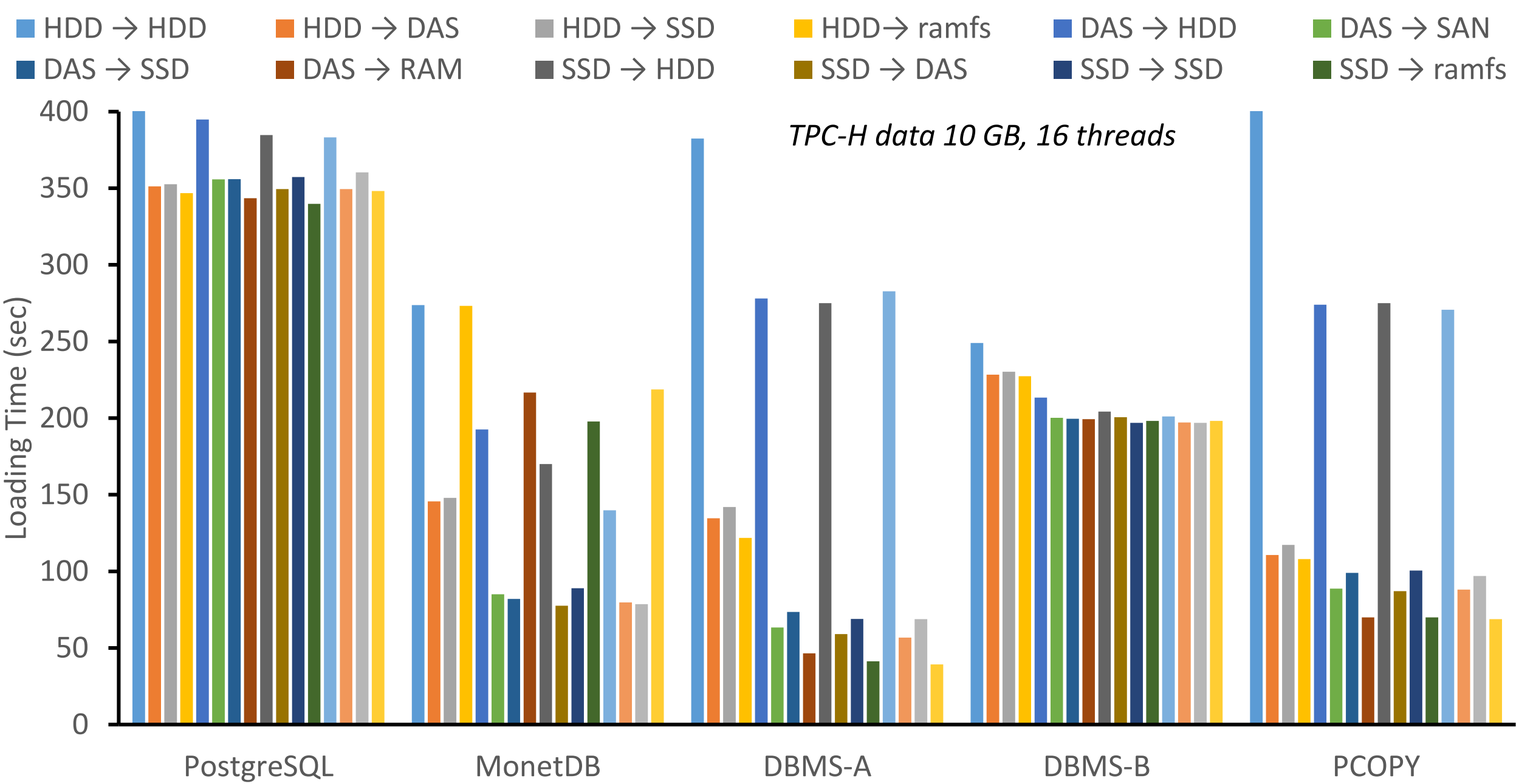
TPC-H benchmark, 10 GB, CSV format



New Parallel export 20X faster than Current export

Reducing data loading overheads: related work




HDD → HDD
HDD → DAS
HDD → SSD
HDD→ ramfs
DAS → HDD
DAS → SAN
DAS → SSD
DAS → RAM
SSD → HDD
SSD → DAS
SSD → SSD
SSD → ramfs
TPC-H data 10 GB, 16 threads
Loading Time (sec)
400
350
300
250
200
150
100
50
0
PostgreSQL
MonetDB
DBMS-A
DBMS-B
PCOPY

Experimental Setup

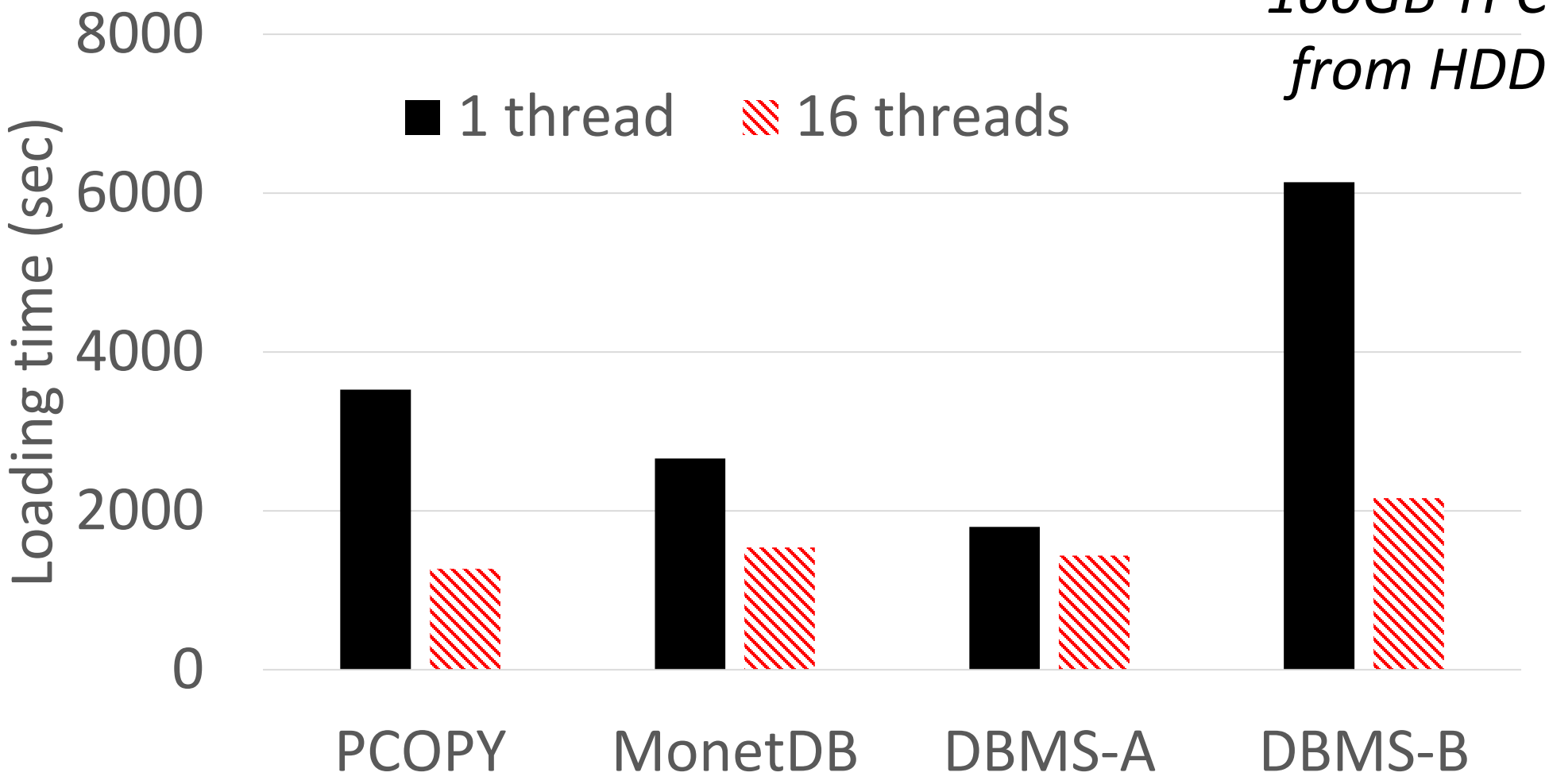
- Hardware
  - Dual socket 8 cores Intel(R) Xeon(R) CPU E5-2640, main memory: 64 GB
  - HDD: 4 x 500 GB 10k RPM SATA disks
    Max read/write throughput: 170 MB/s, 160 MB/s
  - DAS: 24 x 558 GB 10k RPM SATA disks (RAID-0)
    Max read/write throughput: 1100 MB/s, 330 MB/s
  - SSD: 3 x 200 GB, Max read/write throughput: 565 MB/s, 268 MB/s
- Software
  - Red Hat Enterprise Linux 6.6 (Santiago - 64bit) with kernel version2.6.32
  - Row-oriented DBMSs: PostgreSQL (v. 9.3.2), DBMS-A
  - Column-oriented DBMSs: MonetDB (v. 11.19.9), DBMS-B
  - PostgreSQL parallel external loaders: PCOPY and PPARSE
- Datasets
  - benchmark (TPC-H, TPC-C, SDSS) and real-world (Symantec, MIMIC-II)

Storage devices

Name	Capacity	Configuration	Read speed	Write speed	RPM
HDD	1.8 TB	4 x HDD (RAID-0)	170 MB/sec	160 MB/sec	7.5k
DAS	13 TB	24 x HDD (RAID-0)	1100 MB/sec	330 MB/sec	7.5k
SSD	550 GB	3 x SSD (RAID-0)	565 MB/sec	268 MB/sec	N/A
ramfs	64 GB	N/A	12.8 GB/sec	12.8 GB/sec	N/A

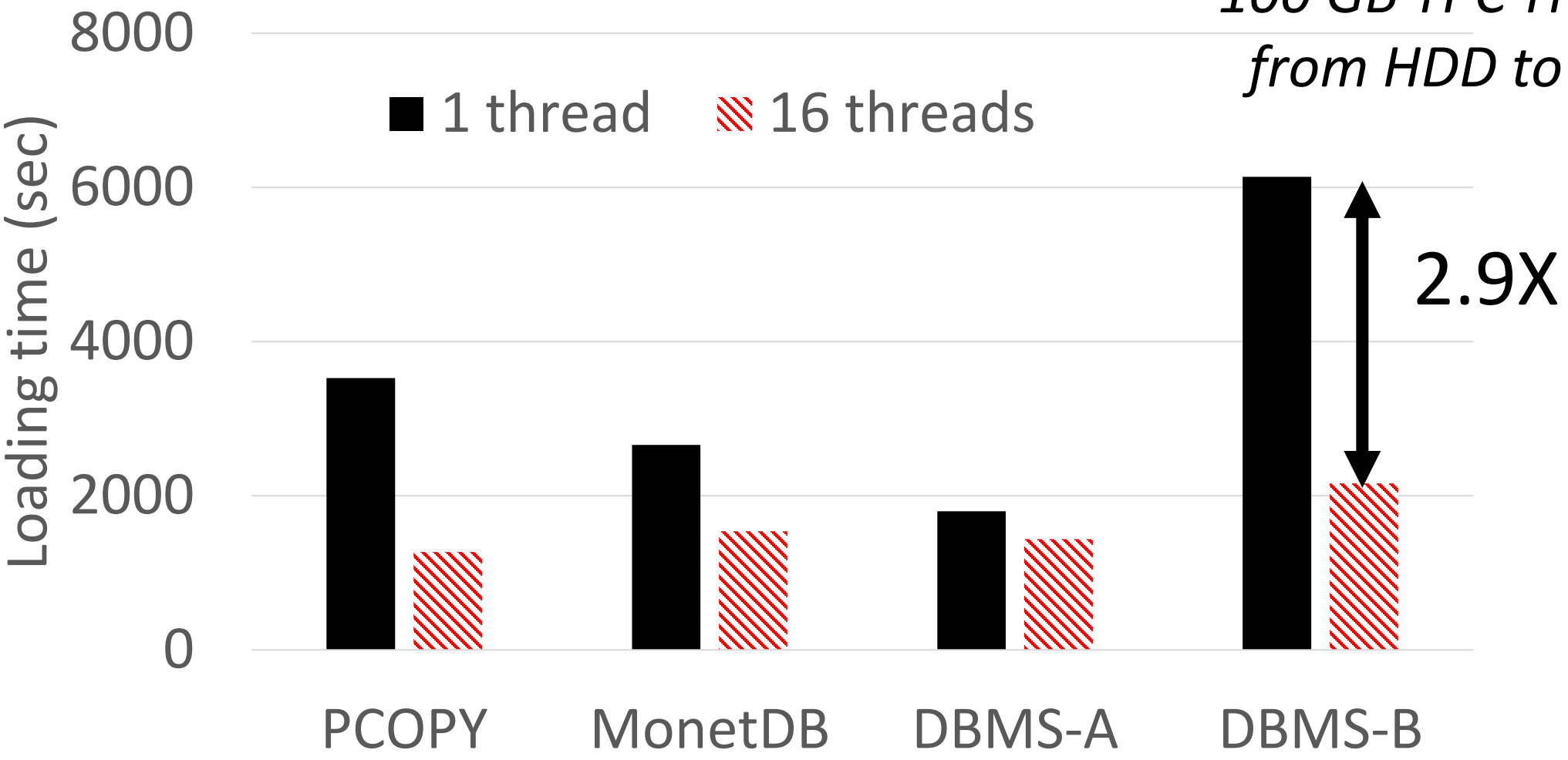
Single threaded vs. Parallel data loading

100GB TPC-H data, from HDD to DAS



Single threaded vs. Parallel data loading

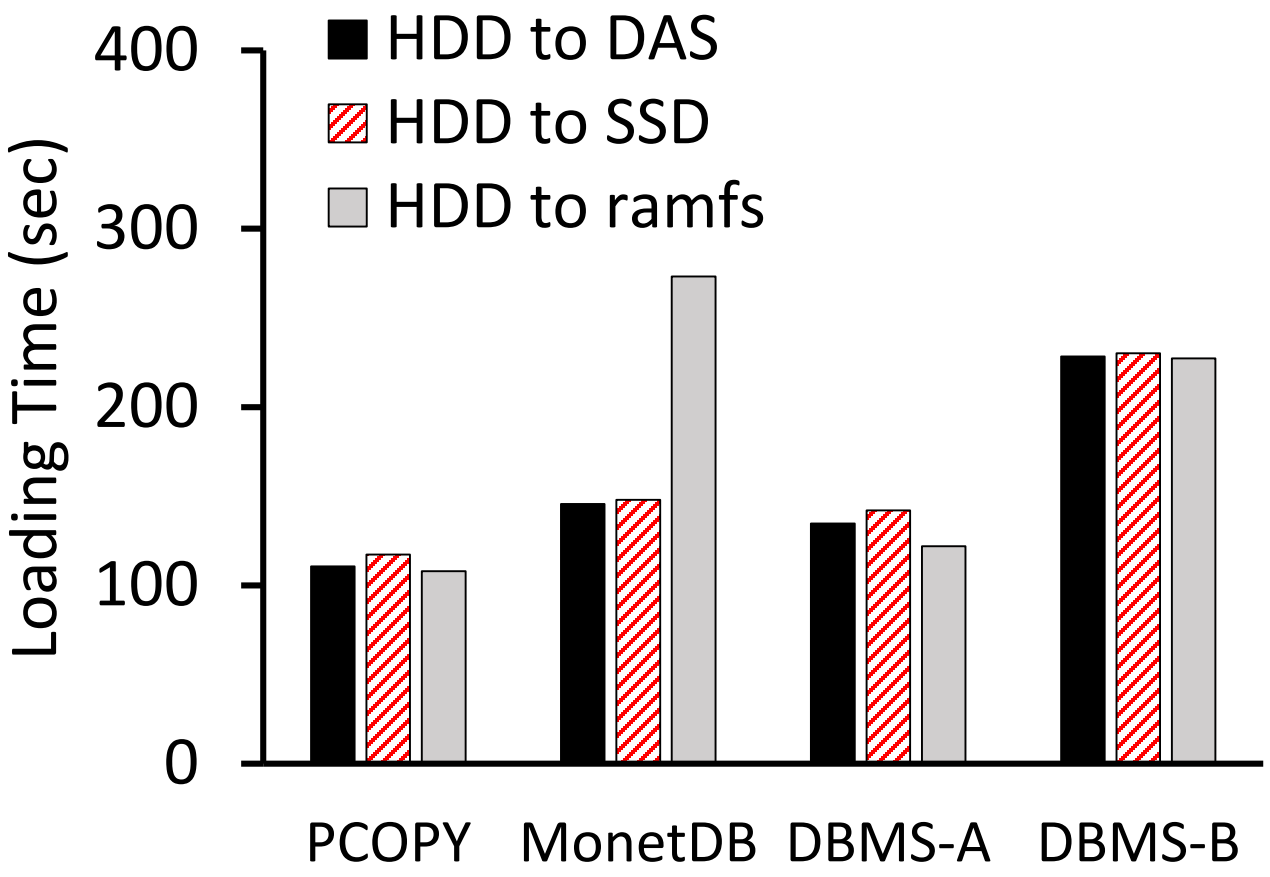
100 GB TPC-H data, from HDD to DAS

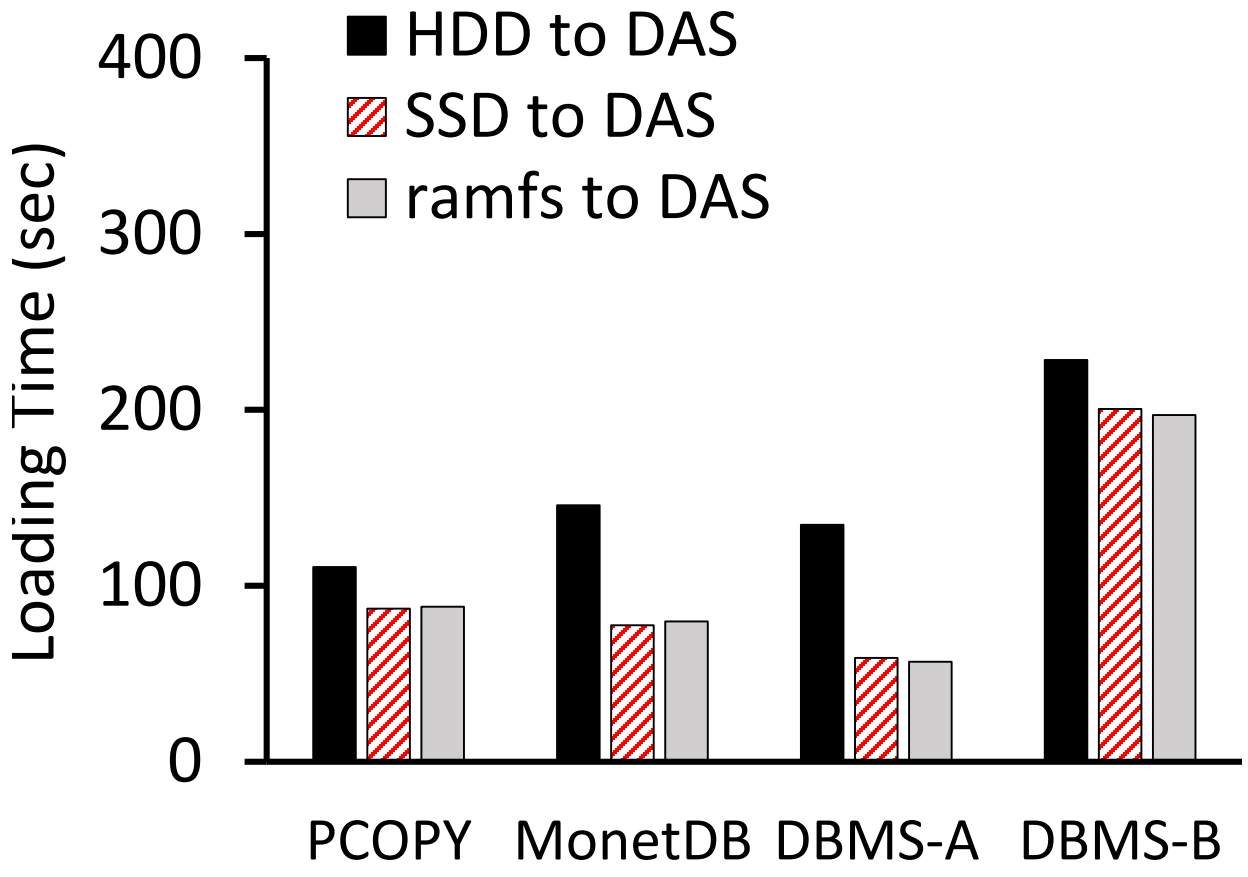


Sublinear speedup for 16 threads

Impact of storage

TPC-H SF 10 GB, 16 threads



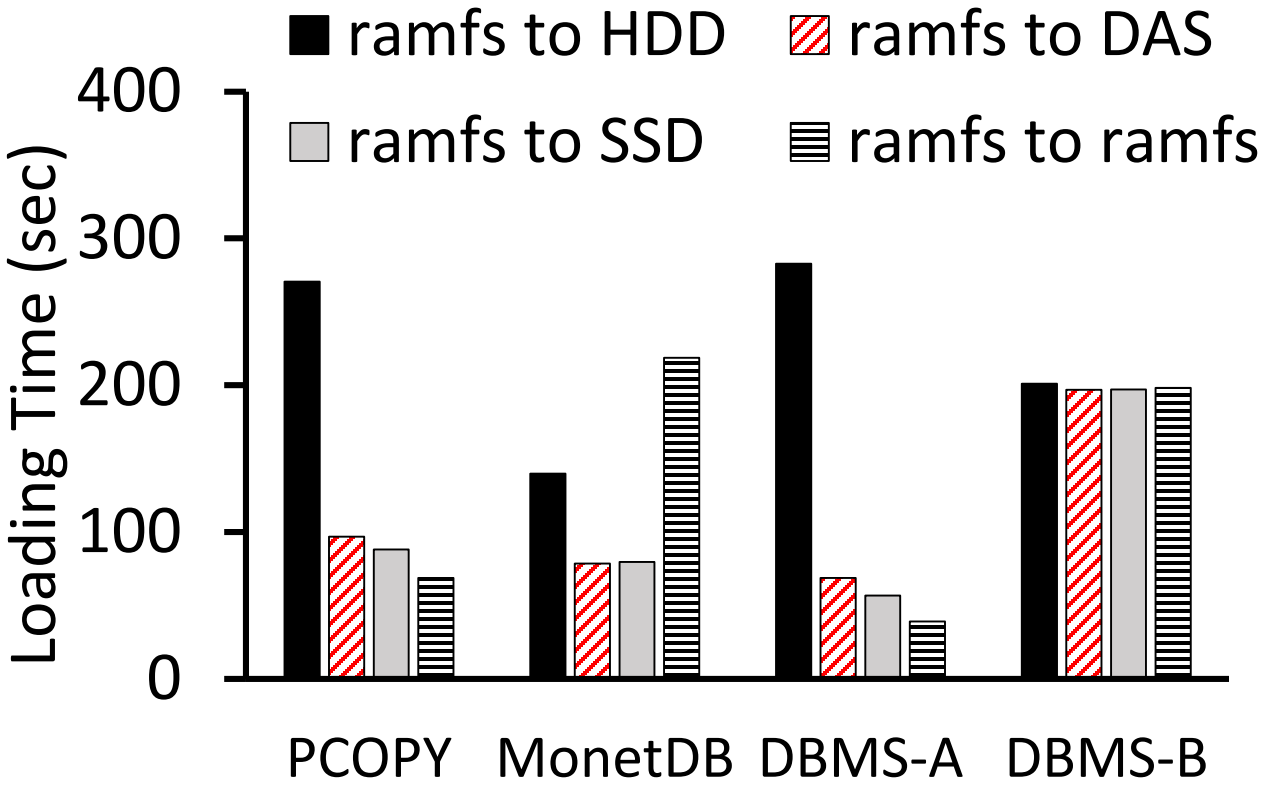


Slow source storage bottlenecks all systems

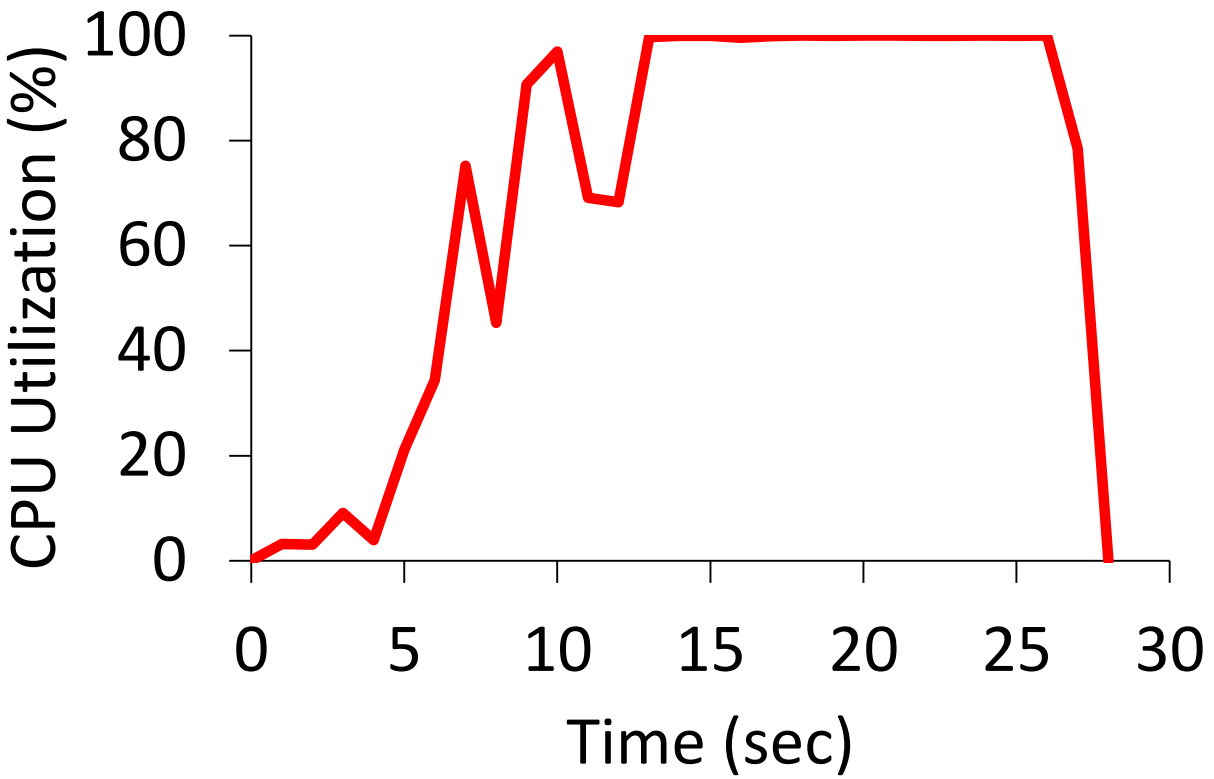
Write bottleneck when source storage is fast

Best-case storage scenario

TPC-H 10 GB, 16 threads



DBMS-A, from ramfs to ramfs, 16 threads

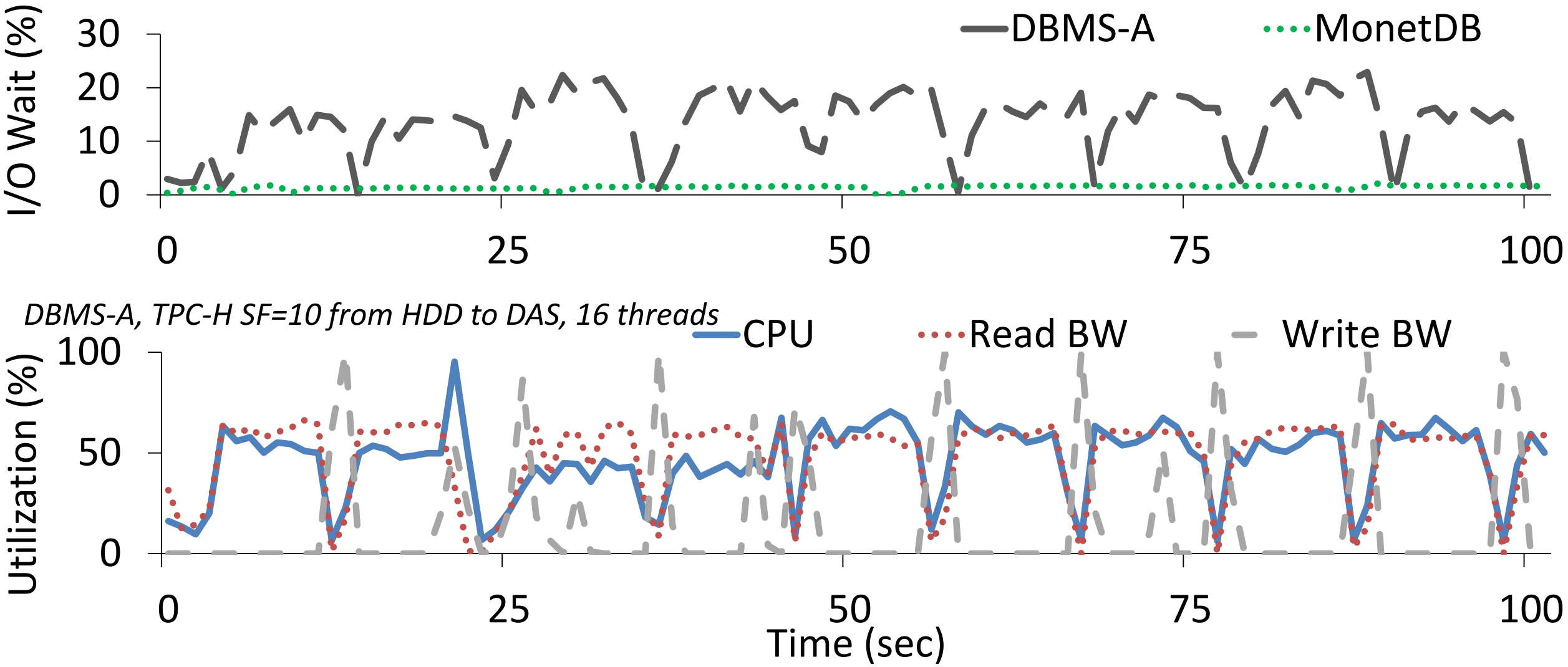


Device Bandwidth: 12.8 GB/sec

Read Rate: 250 MB/sec

100% CPU utilization, yet B/W still underutilized

Resource utilization



Unable to saturate resources

Resource utilization: I/O & CPU

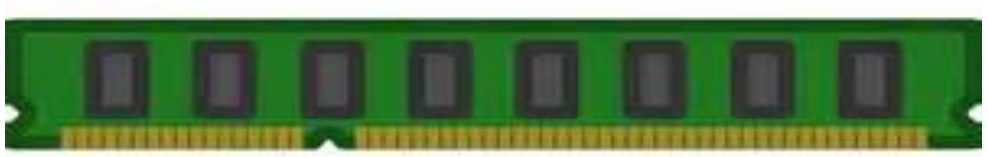
- Understand the lack of scalability under parallel loading
- Analyze the utilization of resources:
  - CPU
  - I/O
- **What is the alternative view of the previous figures?**

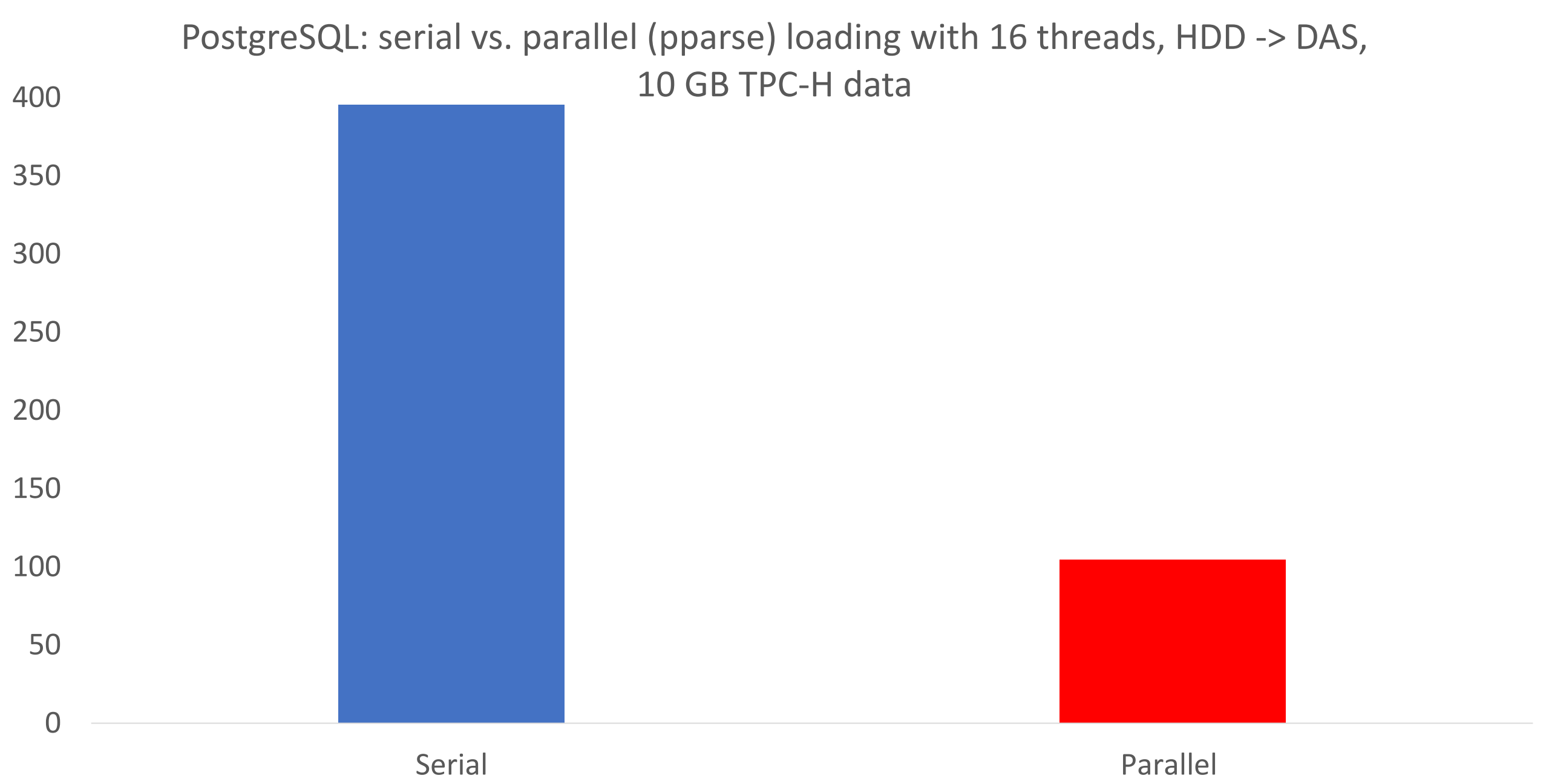
Impact of storage

- A typical DBMS setup underutilizes I/O bandwidth and CPU
- Problem: random I/O for **parallel loading from HDD**
- **How different storage sub-systems affect data loading speed?**

DBMS

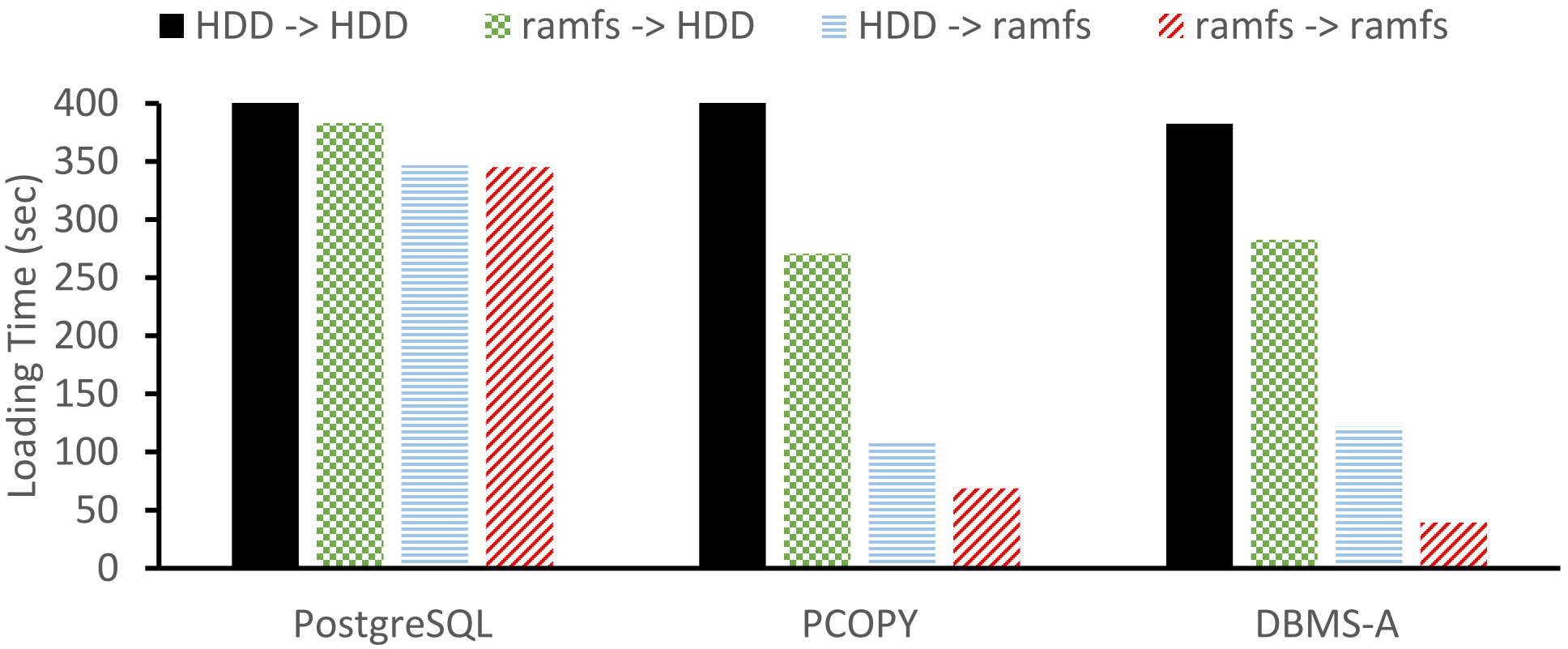
STORAGE




PostgreSQL: serial vs. parallel (pparse) loading with 16 threads, HDD -> DAS, 10 GB TPC-H data
400
350
300
250
200
150
100
50
0
Serial
Parallel

Impact of storage devices

TPC-H data 10 GB, 16 threads



Slow source storage bottlenecks all systems

Write bottleneck when source storage is fast

High cost of Data Migration

2 nodes, each 4 cores 3.10 GHz, 4 MB L3, 16 GB RAM, SSD 250 GB, Ubuntu 14.04

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

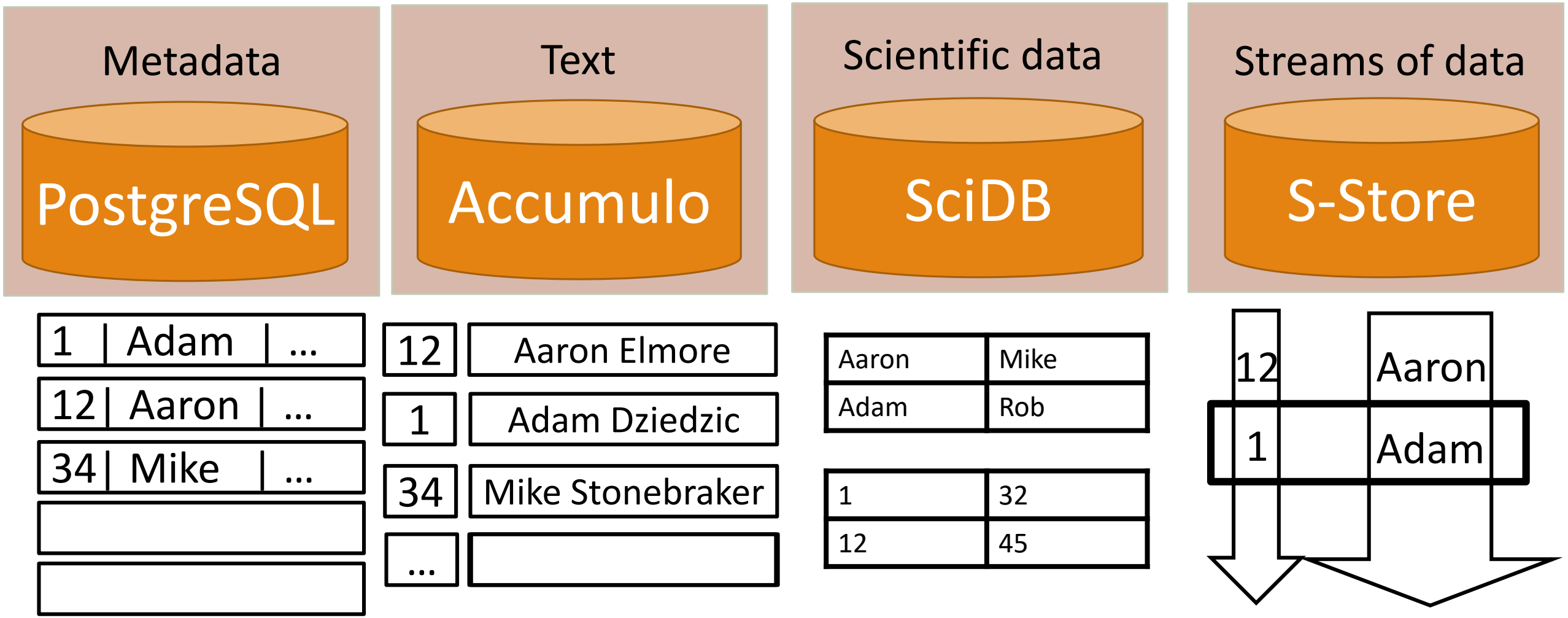
High cost of Data Migration

2 nodes, each 4 cores 3.10 GHz, 4 MB L3, 16 GB RAM, SSD 250 GB, Ubuntu 14.04

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**

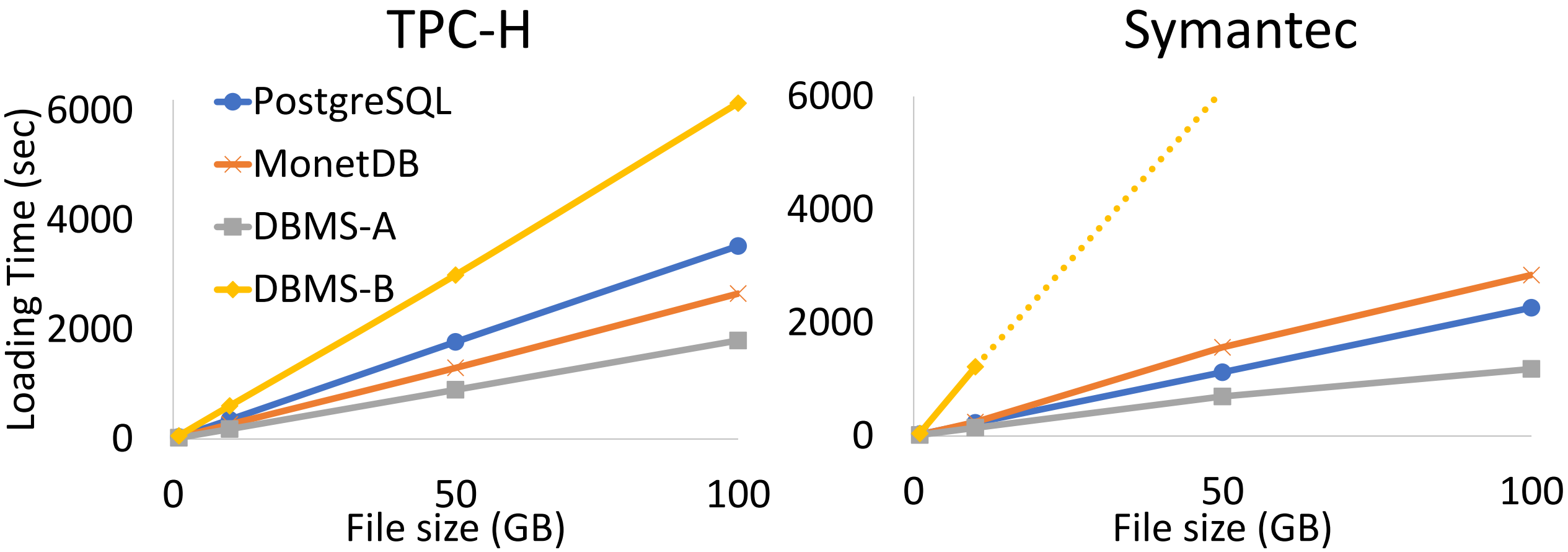
Data migration can be improved

Polystore: ”One size does not fit all”



Polystore couples diverse data models

Single-threaded data loading

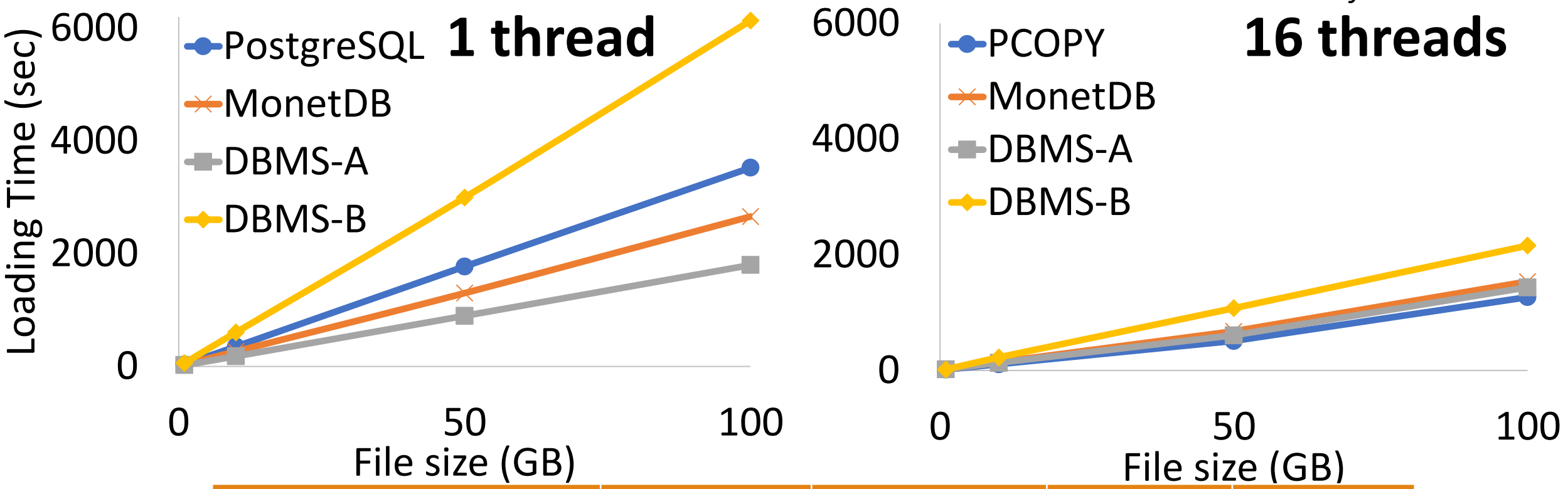


Dataset characteristics matter

Effect of compression

Single vs. parallel data loading

TPC-H data, from HDD to DAS

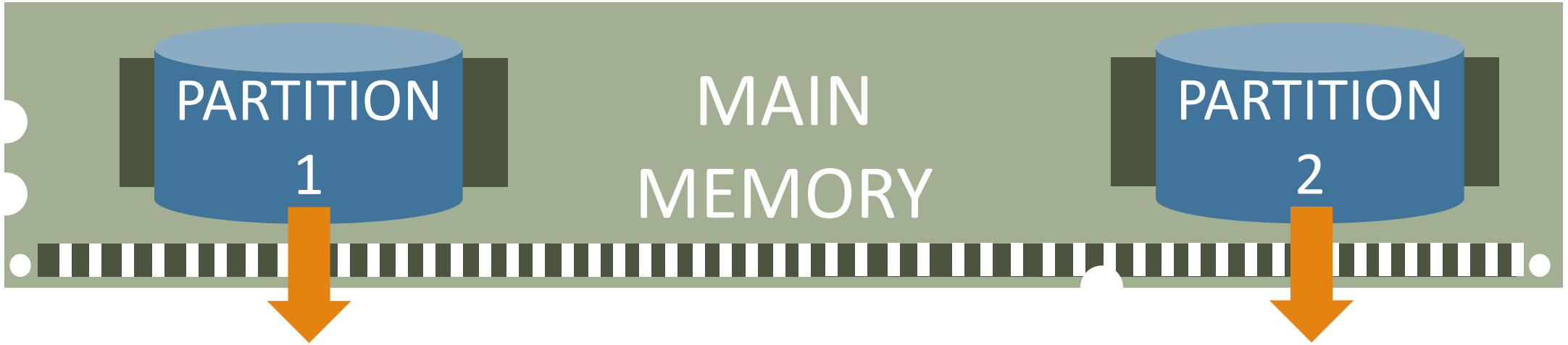


Speedup 16 threads	PCOPY	MonetDB	DBMS-A	DBMS-B
TPC-H 100 GB	2. 8	1.7	1.3	2.8
Symantec 100 GB	1.9	2. 2	**0.88**	-

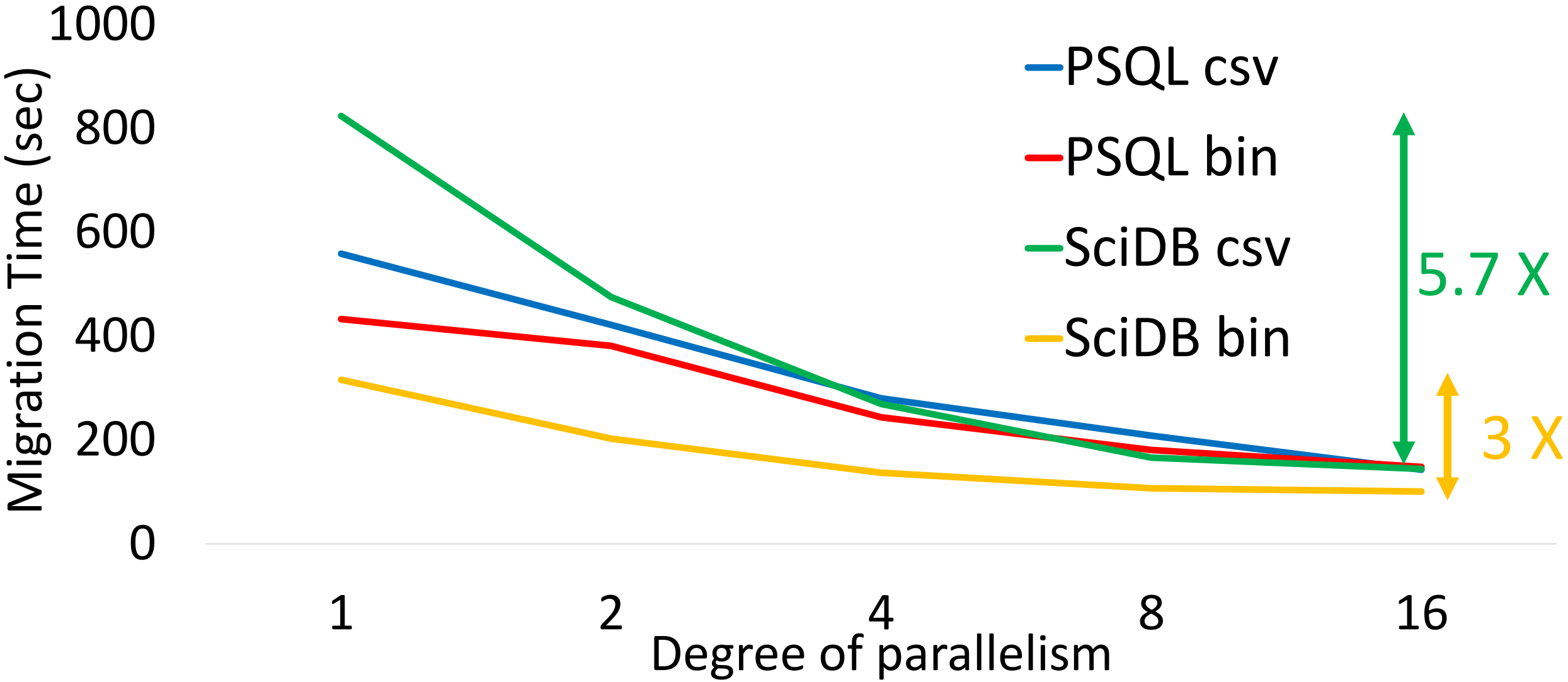
Sublinear speedup for 16 threads

Data Migration from S-Store to PostgreSQL & SciDB

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



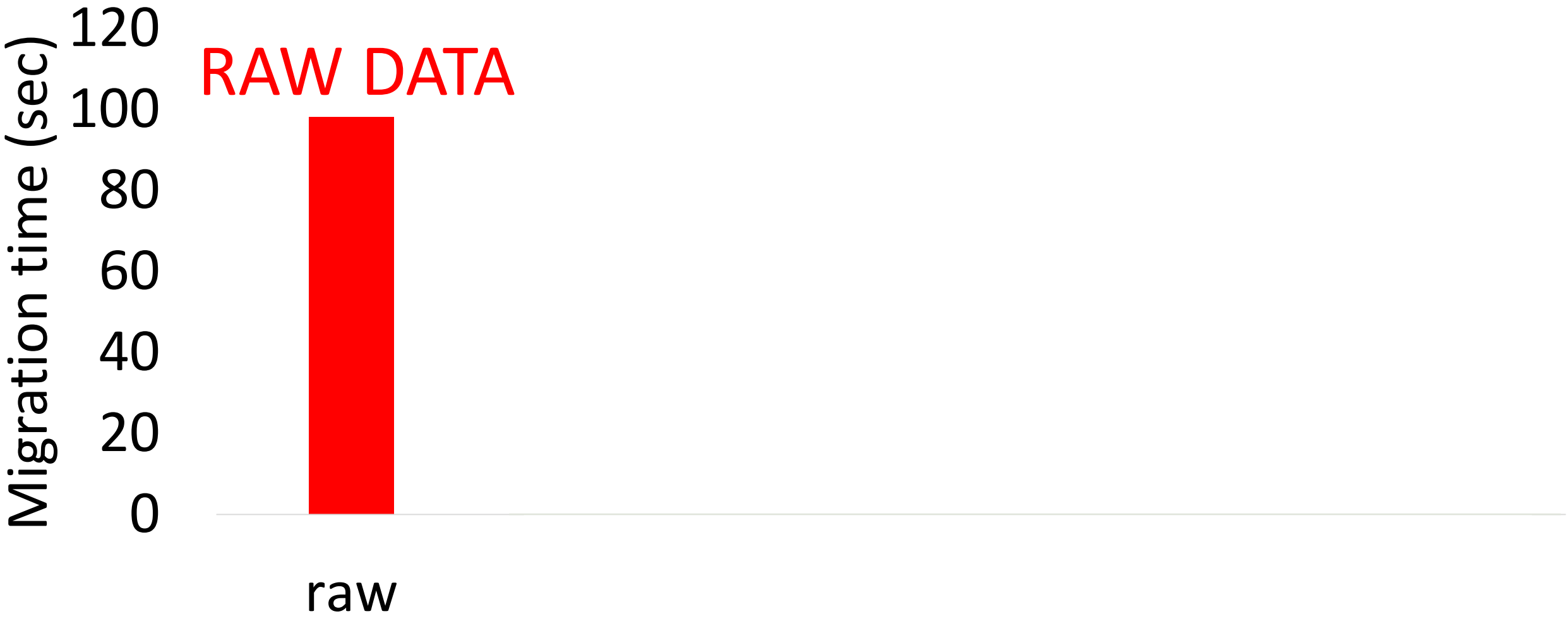
Data Migration from S-Store to PostgreSQL & SciDB



Faster parallel migration: 3 X for binary format & 5.7 X for CSV

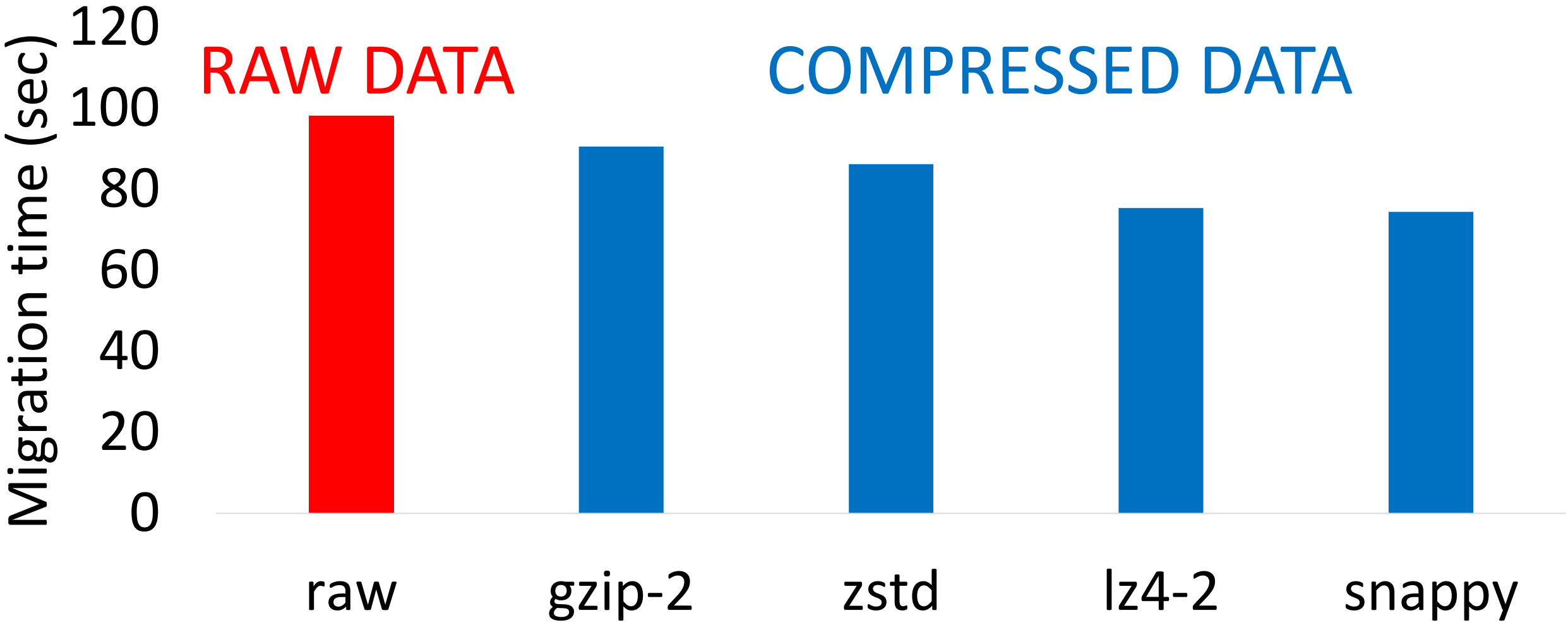
COMPRESSION for direct binary parallel migration

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



COMPRESSION for direct binary parallel migration

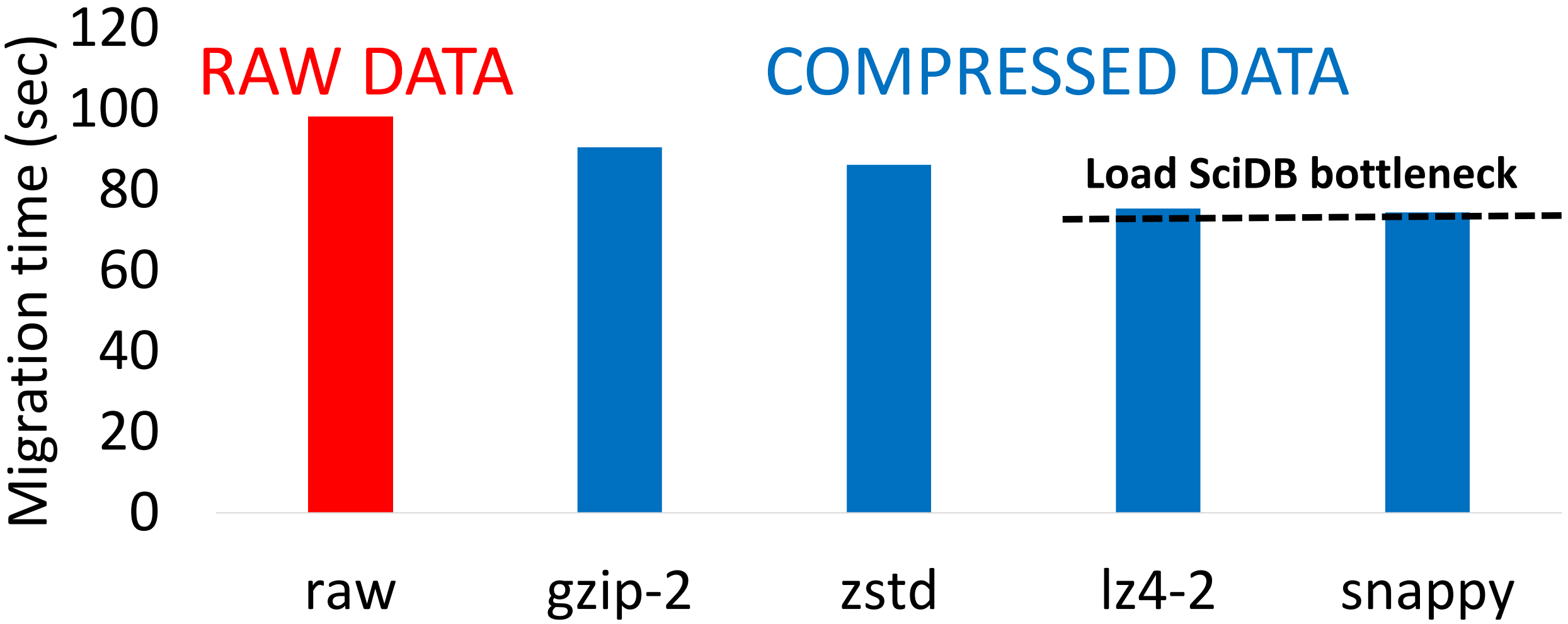
From PostgreSQL to SciDB, 4 threads, waveform data (int,int,double), 10 GB



Lightweight compression for data transfer via network

COMPRESSION for direct binary parallel migration

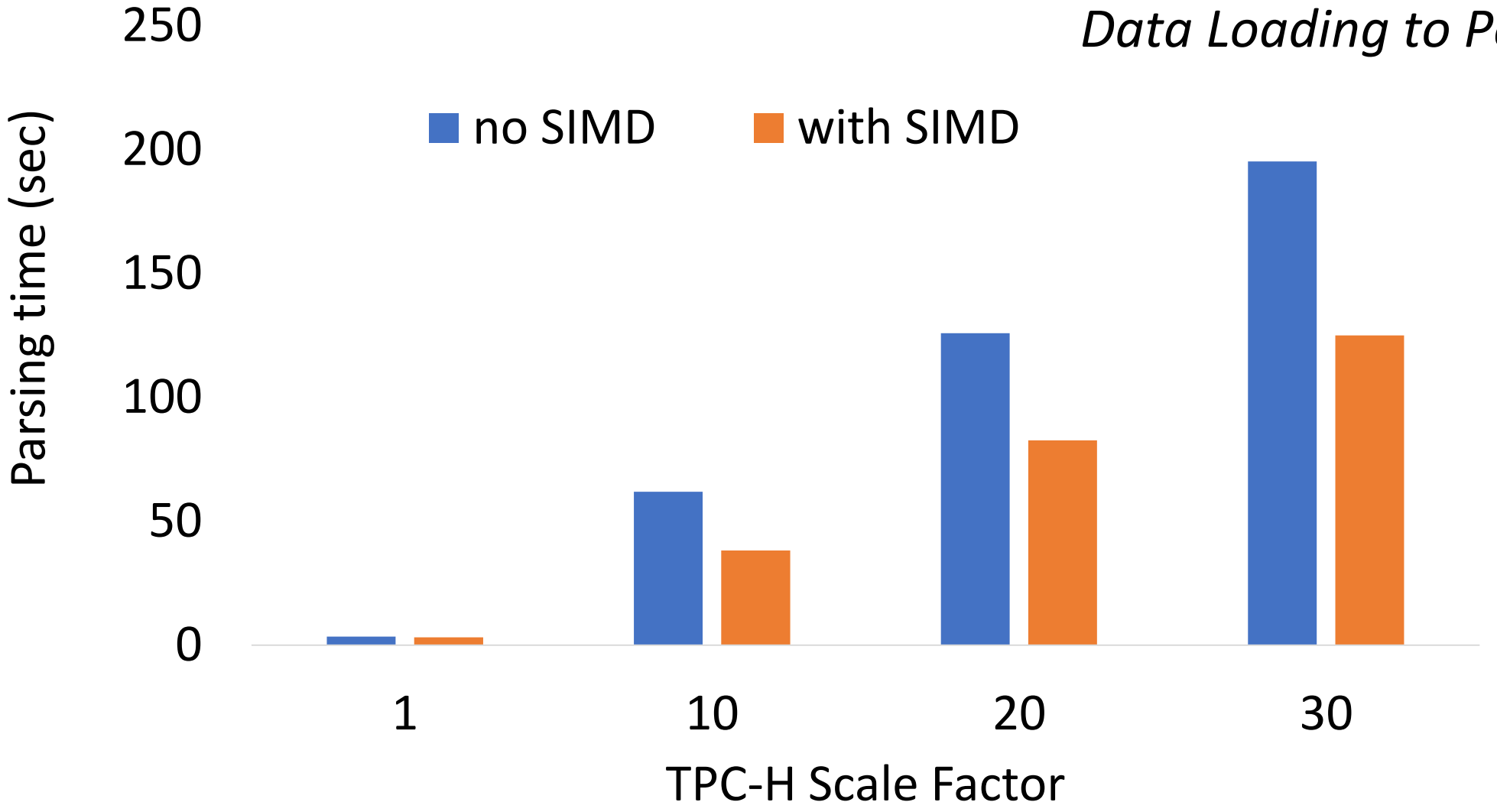
From PostgreSQL to SciDB, 4 threads, waveform data (int,int,double), 10 GB



Lightweight compression for data transfer via network

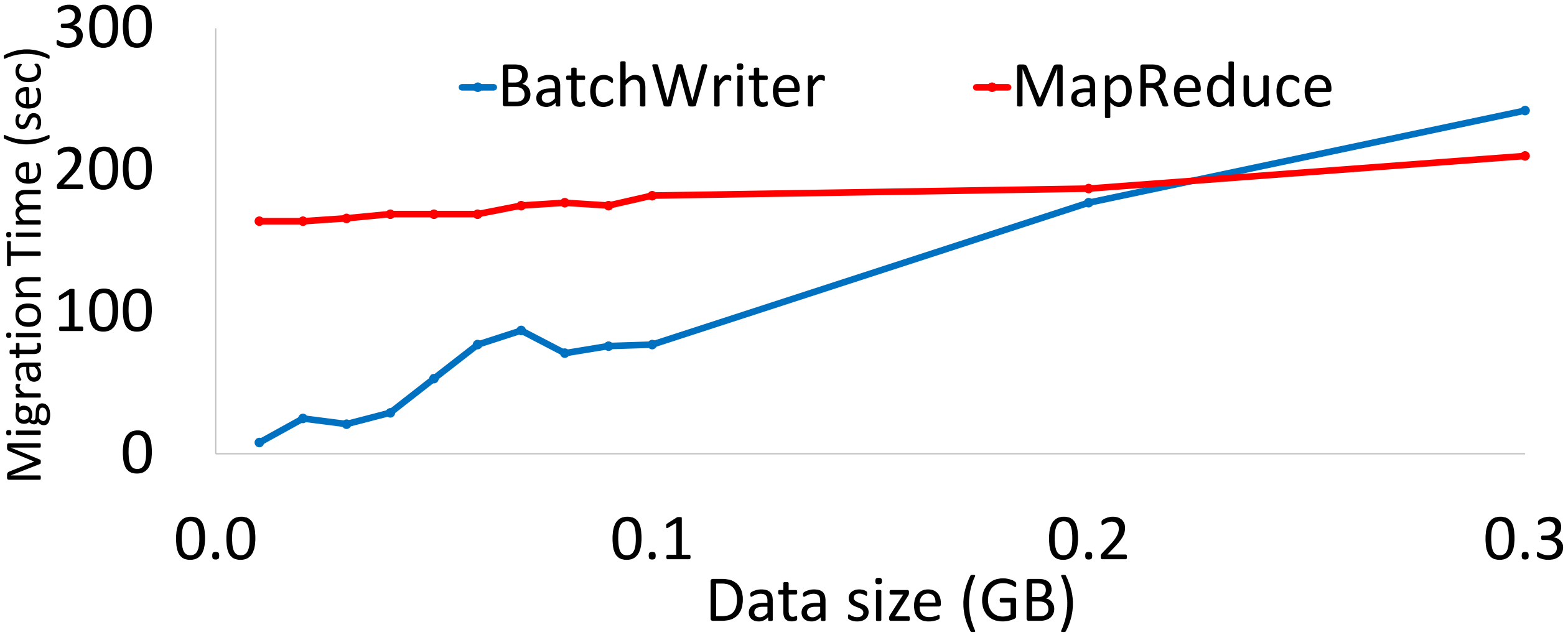
SIMD: for parsing the lines in input CSV file

Data Loading to PostgreSQL

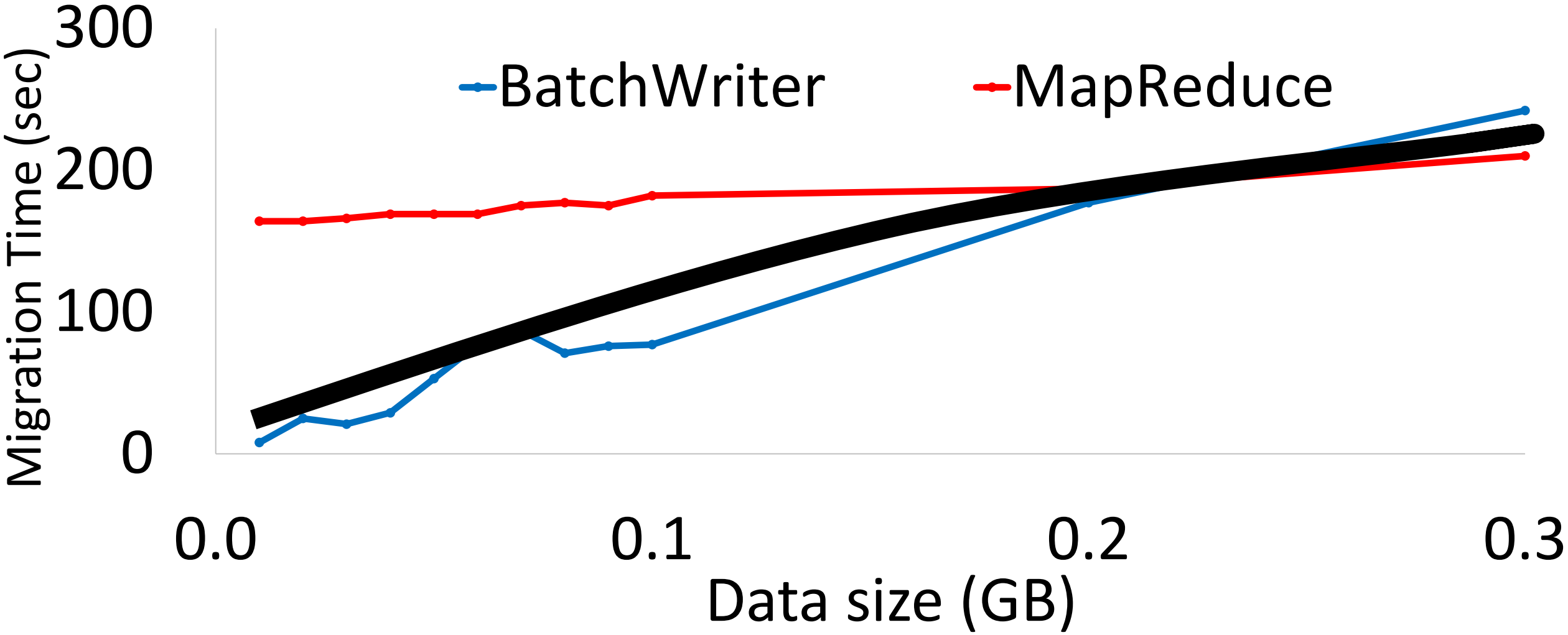


SIMD gives 1.6X speedup for parsing lines

Data Migration from PostgreSQL to Accumulo



Data Migration from PostgreSQL to Accumulo

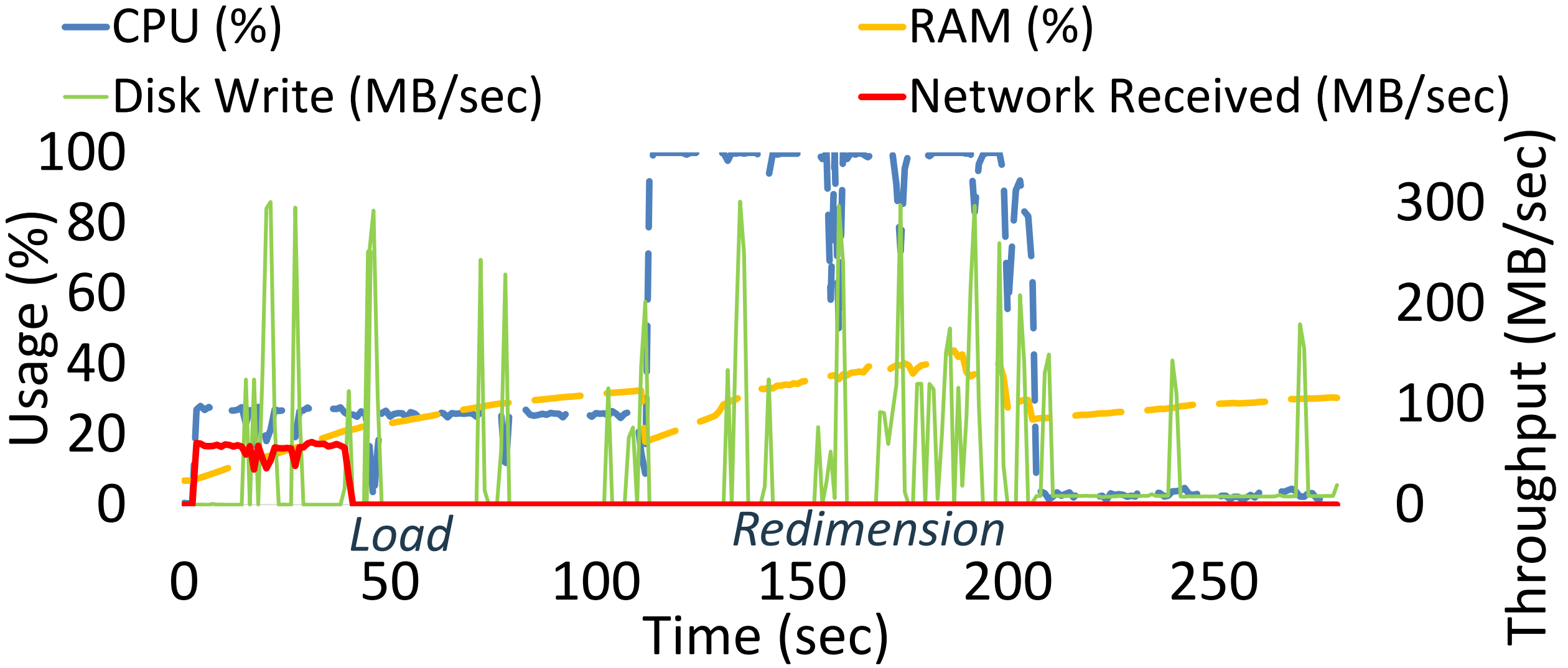


Adaptive data loading method

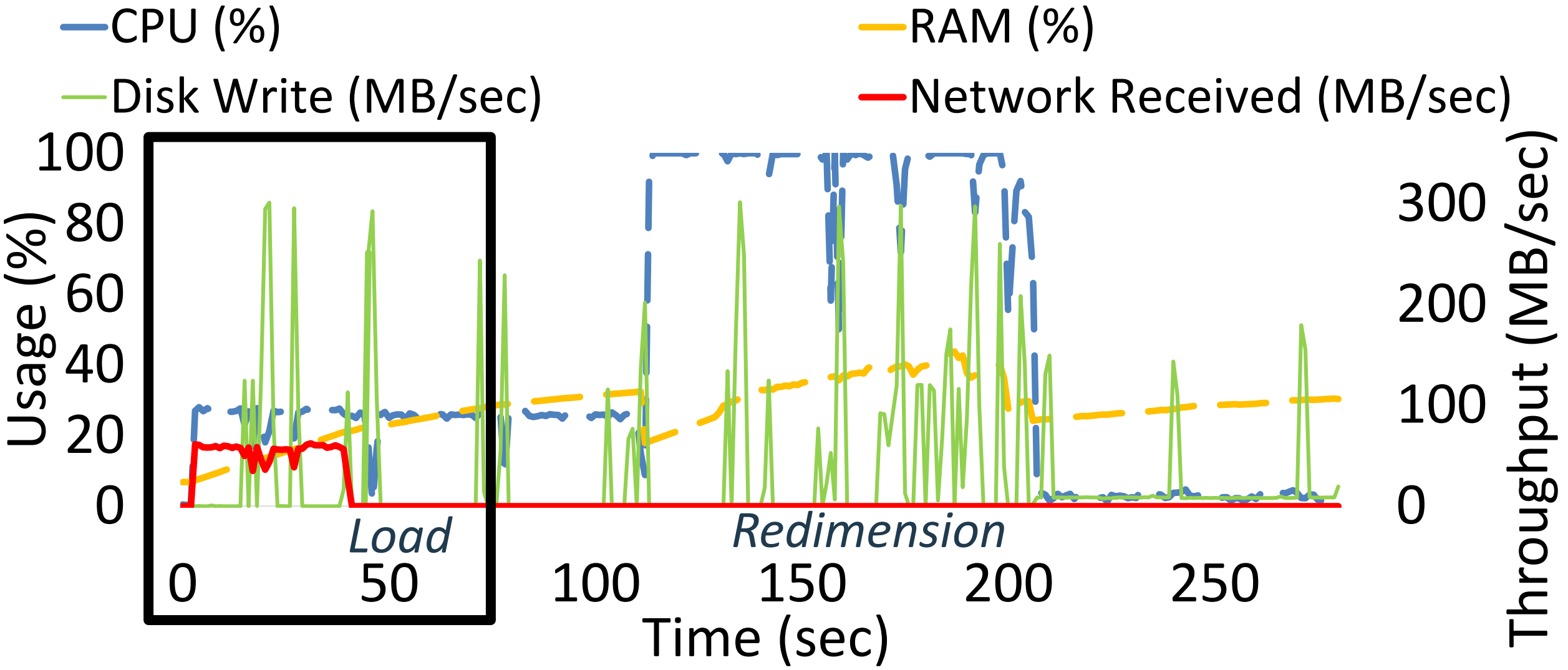
Future work

- Core migrator with a plug-in load/export interface for new DBMSs
- Common binary & SIMD-friendly format for DBMS data exchange
- Storage aware loading for the reading part & buffered writes handled in a separate thread (no bursts of writes)
- Equivalences between physical designs, e.g. mapping indexes in PostgreSQL to dimensions and chunk sizes in SciDB
- Improving the diverse analytics by integrating systems that are more homogenous for faster data sharing/migration with optimized code-paths for specialized analysis, e.g. SQL Server
- Integration with external Machine Learning tools (e.g. TensorFlow) and better data provisioning for the ML systems (e.g. via SQLite)

Resource usage: CSV wavefrom data loading to SciDB

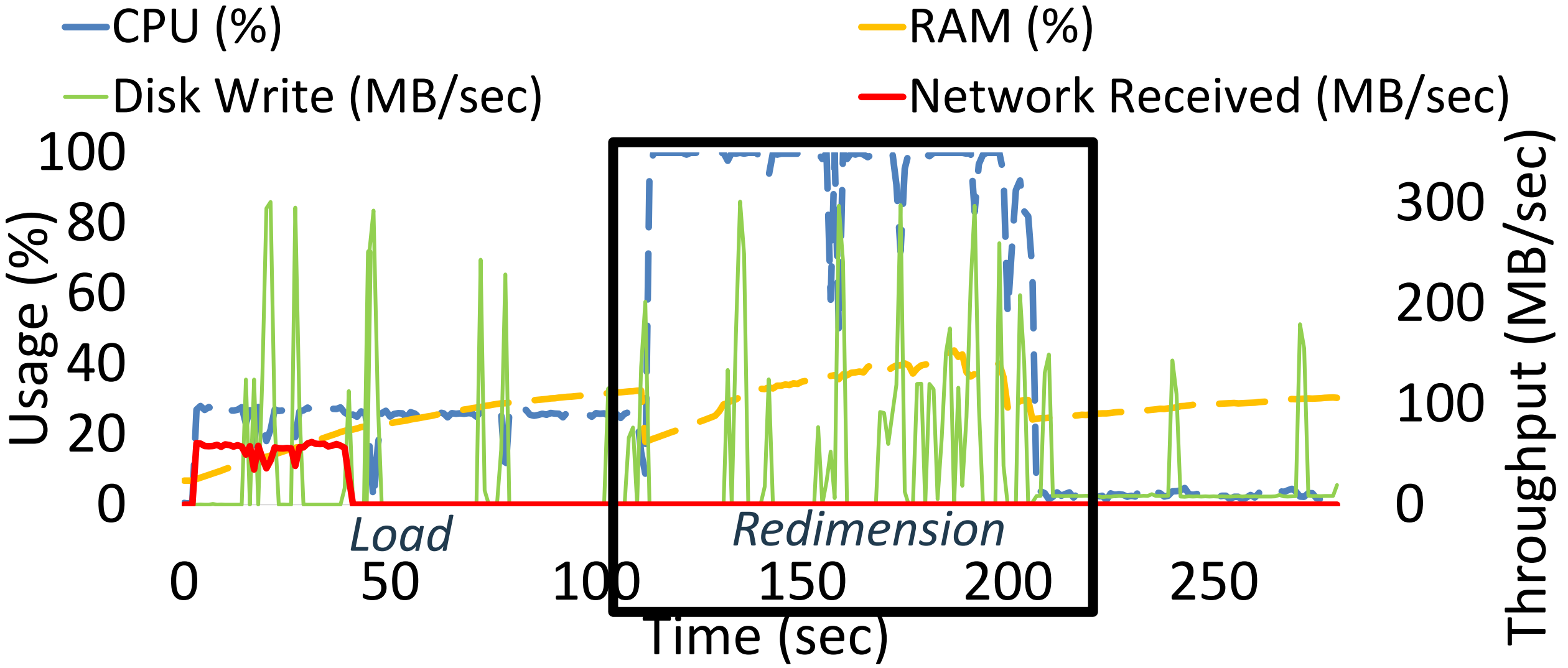


Resource usage: CSV wavefrom data loading to SciDB



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

Resource usage: CSV wavefrom data loading to SciDB

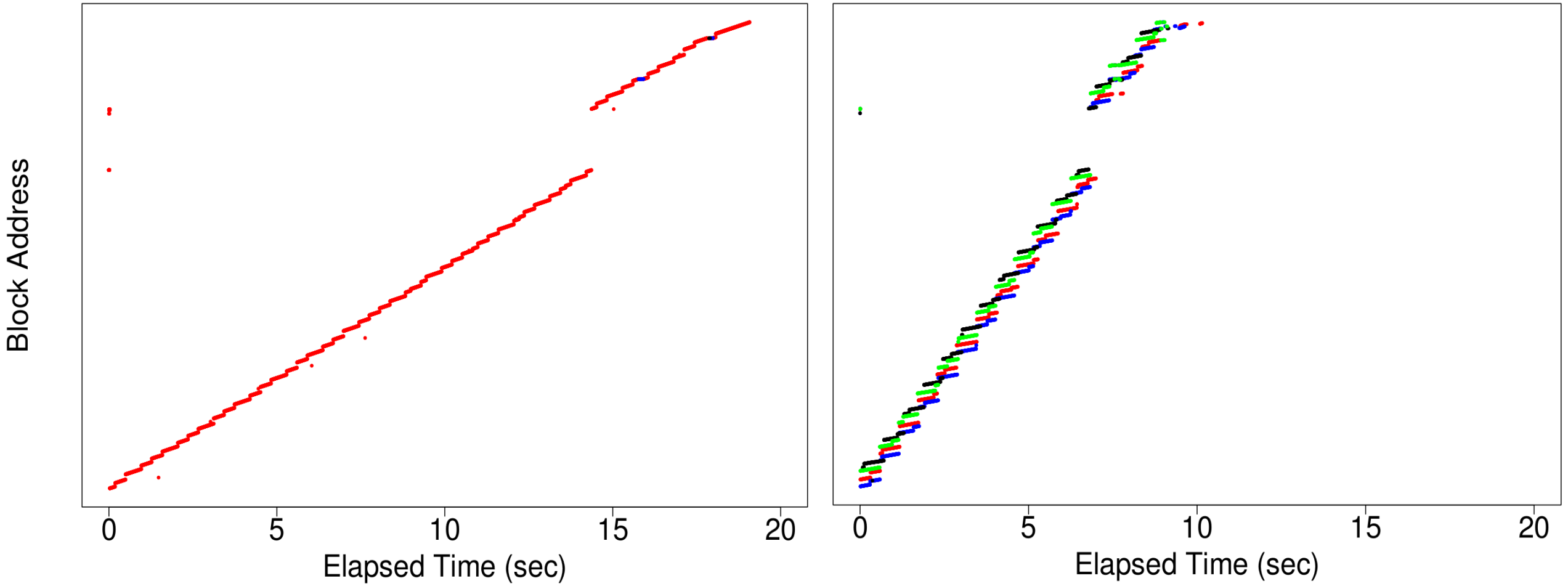


What is an optimal degree of parallelism?

Single-threaded vs. Parallel Export from PostgreSQL

Single-thread export
(1 reader from disk)

4-thread export
(4 readers from disk)



Parallel loading in the presence of PK constraints