Tutorial: Big Data System Benchmarking

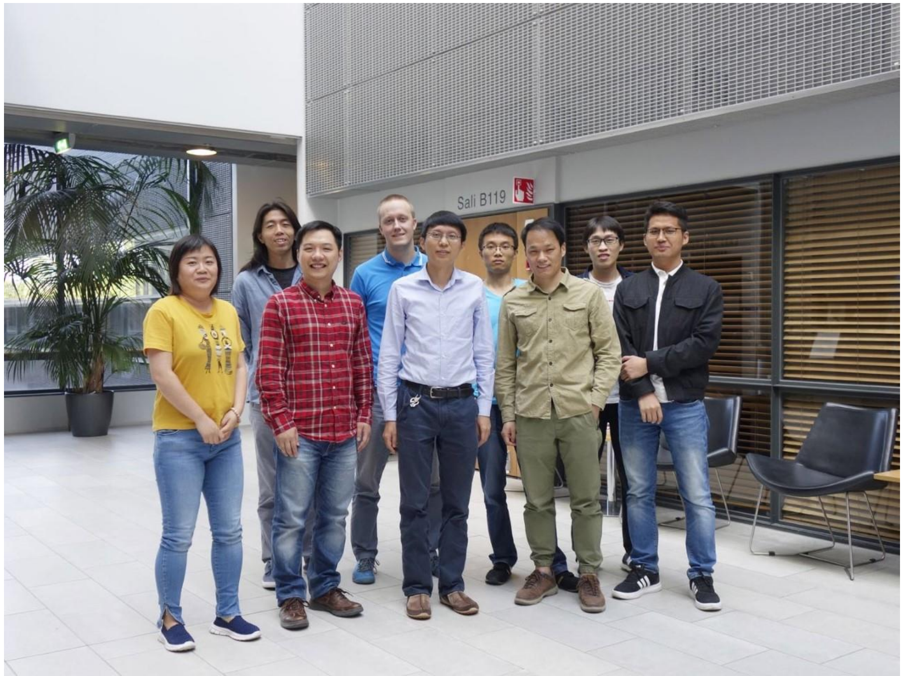
-- State of the Art, Current Practices, and Open Challenges

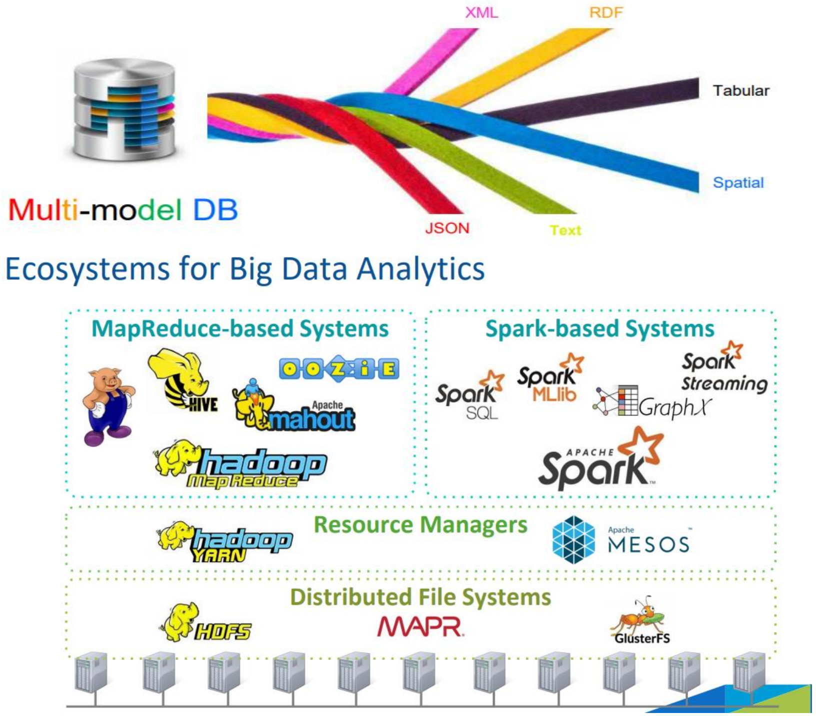
Chao Zhang and Jiaheng Lu
Department of Computer Science
University of Helsinki

UNIVERSITY OF HELSINKI
FACULTY OF SCIENCE

About us

- Database Group at University of Helsinki, Finland
- Website : http://helsinki.fi/udbms





Outline

- Introduction to Big Data System Benchmarking(15')
- Benchmarking SQL Big Data Analytical Systems(25')
- Benchmarking Map-Reduce/NoSQL Systems(15')
- Benchmarking Graph-based Big Data Systems(20')
- Benchmarking Multi-Model Big Data Systems(35')
- Open Challenges and Future Directions(10')

We are in the era of big data

- Lots of data is being collected
  - Web data, e-commerce
  - Bank/Credit Card transactions
  - Social Network
  - Scientific data

Four V's of big data

Big data systems are ubiquitous

IT log analysis | Banking business | Social network | Social Commerce

Data warehouse | E-commerce | Fraud detection | Healthcare

Big Data Systems

Apache Spark | Teradata | JanusGraph | ArangoDB

Hadoop | TiDB | neo4j | AGENS Graph

Hive | mongoDB | TigerGraph | OrientDB

Magic mirror in my hand, which is the best in the land?

IT log analysis | Banking business | Social network | Social Commerce

Data warehouse | E-commerce | Fraud detection | Healthcare

Big Data Systems

Apache Spark, hadoop, HIVE

TERADATA, TiDB, mongoDB

JanusGraph, neo4j, TigerGraph

ArangoDB, AGENS Graph, OrientDB

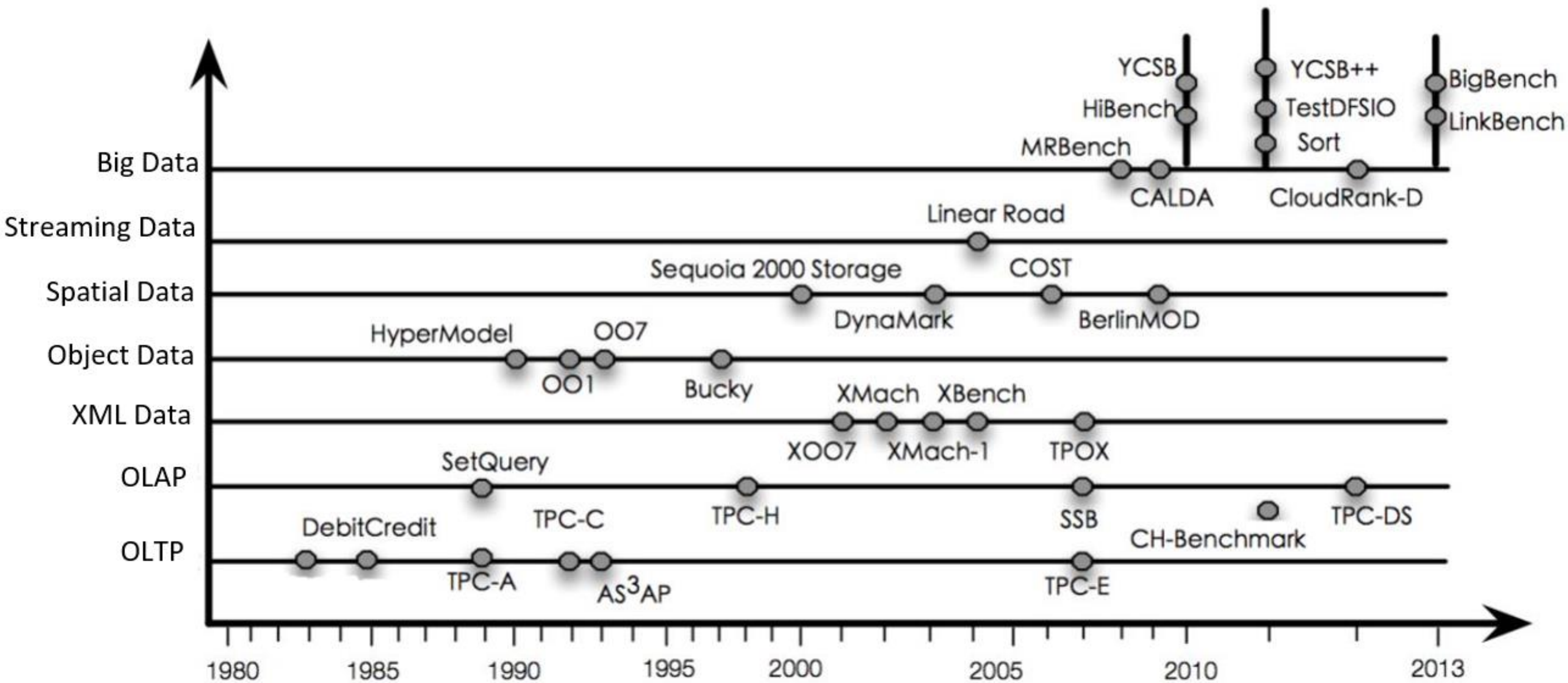
Big data benchmarks make it easy

- A suite of programs that make competing products comparable, help practitioners choose the right big data systems

- Identify the performance bottlenecks to make big data systems better

- One size doesn't fit all, i.e., we need specific benchmarks for various cases

Timeline of database benchmarks



Key elements of benchmarks

- Domain with data schema
- Synthetic data generators
- Specified workloads, e.g., queries
- Performance metrics, e.g., latency
- Execution rules, e.g., power/throughput test

Taxonomy of big data benchmarks

System domain	System examples	Benchmarks
Map-Reduce based	Hadoop, Spark, Flink	AMP Benchmark and HiBench
NoSQL based	MongoDB, Cassandra, Redis	YCSB
SQL based	Hive, Teradata, Presto, Spark SQL	TPC-H, TPC-DS, BigBench
Graph based	Neo4j, JanusGraph, Giraph	LDBC Graphalytics, SNB
Multi-model based	ArangoDB, OrientDB, AgensGraph	TPC-DI, PolyBench, UniBench
Others	Streaming, Spatial, RDF, or Micro-benchmarks	

Main topics of this tutorial

System domain	System examples	Benchmarks
Map-Reduce based	**Hadoop, Spark, Flink**	**AMP Benchmark and HiBench**
NoSQL based	**MongoDB, Cassandra, Redis**	**YCSB**
SQL based	**Hive, Teradata, Presto, Spark SQL**	**TPC-H, TPC-DS, BigBench**
Graph based	**Neo4j, JanusGraph, Giraph**	**LDBC Graphalytics, SNB**
Multi-model based	**ArangoDB, OrientDB, AgensGraph**	**TPC-DI, PolyBench, UniBench**
Others	Streaming, Spatial, RDF, or Micro-benchmarks	

At the end of this talk

You are expected to acquire the following knowledge:

- **The key techniques of various big data benchmarks**
- **The relationship of big data benchmarks and their applications**
- **Current practices and Future directions**

Outline

- Introduction to Big Data System Benchmarking(15')
- Benchmarking SQL Big Data Analytical Systems(25')
- Benchmarking Map-Reduce/NoSQL Systems(15')
- Benchmarking Graph-based Big Data Systems(20')
- Benchmarking Multi-Model Big Data Systems(35')
- Open Challenges and Future Directions(10')

Benchmarking SQL Analytical Systems

- TPC to the rescue (http://www.tpc.org/)
- Complex business analysis applications with structured data
- We look at three representatives: TPC-H, TPC-DS, TPCx-BB

TPC-H -- an overview

- Based on a business analysis application with 8 tables, e.g., customers and orders
- Data generation with scale factor, e.g., 1
- 22 business queries with choke-point design

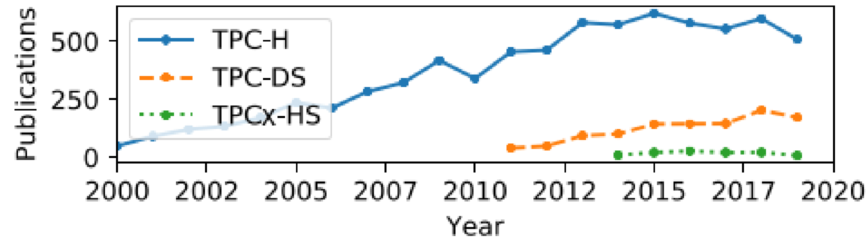


Figure 1: Number of publications indexed on Google Scholar referencing "TPC-H", "TPC-DS", or "TPCx-HS", starting with the year of the benchmark's publication. In the nine years after being published (1999-2007), TPC-H was referenced in 1 633 publications, while TPC-DS was only referenced 1 094 times in the respective nine-year frame.

Figure from Markus et al. Quantifying TPC-H Choke Points and Their Optimizations, PVLDB 2020.

TPC-H 22 queries

- 28 choke points
- 6 categories
  - *aggregation*
  - *join*
  - *data access locality*
  - *expression calculation*
  - *correlated subqueries*
  - *parallelism&concurrency*

Figure from Peter Boncz et al. TPC-H Analyzed: Hidden Messages and Lessons Learned from an Influential Benchmark. TPCTC 2013.

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14	Q15	Q16	Q17	Q18	Q19	Q20	Q21	Q22
CP1	strong	medium	strong	strong	medium	medium	medium	medium	medium	strong	medium	medium	strong	medium	medium	medium	medium	strong	medium	strong	strong	medium
CP2	light	medium	medium	medium	medium	light	medium	medium	strong	medium	medium	medium	medium	medium	medium	medium	medium	strong	medium	medium	medium	medium
CP3	light	light	strong	strong	strong	medium	strong	strong	light	strong	light	strong	light	medium	medium	light	light	light	light	medium	light	light
CP4	strong	medium	medium	light	medium	medium	medium	light	medium	light	light	medium	strong	medium	light	strong	light	light	strong	medium	light	strong
CP5	light	strong	light	medium	light	light	light	light	light	light	strong	light	light	light	strong	medium	strong	light	light	strong	strong	medium
CP6	light	strong	medium	medium	strong	light	strong	strong	strong	medium	medium	medium	medium	medium	strong	strong	strong	medium	strong	strong	strong	medium

CP1 Aggregation Performance. Performance of aggregate calculations.

CP1.1 QEXE: Ordered Aggregation.
CP1.2 QOPT: Interesting Orders.
CP1.3 QOPT: Small Group-by Keys (array lookup).
CP1.4 QEXE: Dependent Group-By Keys (removal of).

CP2 Join Performance. Voluminous joins, with or without selections.

CP2.1 QEXE: Large Joins (out-of-core).
CP2.2 QEXE: Sparse Foreign Key Joins (bloom filters).
CP2.3 QOPT: Rich Join Order Optimization.
CP2.4 QOPT: Late Projection (column stores).

CP3 Data Access Locality. Non-full-scan access to (correlated) table data.

CP3.1 STORAGE: Columnar Locality (favors column storage).
CP3.2 STORAGE: Physical Locality by Key (clustered index, partitioning).
CP3.3 QOPT: Detecting Correlation (ZoneMap,MinMax,multi-attribute histograms).

CP4 Expression Calculation. Efficiency in evaluating (complex) expressions.

CP4.1 Raw Expression Arithmetic.
CP4.1a QEXE: Arithmetic Operation Performance.
CP4.1b QEXE: Overflow Handling (in arithmetic operations).
CP4.1c QEXE: Compressed Execution.
CP4.1d QEXE: Interpreter Overhead (vectorization; CPU/GPU/FPGA JIT compil.).
CP4.2 Complex Boolean Expressions in Joins and Selections.
CP4.2a QOPT: Common Subexpression Elimination (CSE).
CP4.2b QOPT: Join-Dependent Expression Filter Pushdown.
CP4.2c QOPT: Large IN Clauses (invisible join).
CP4.2d QEXE: Evaluation Order in Conjunctions and Disjunctions.
CP4.3 String Matching Performance.
CP4.3a QOPT: Rewrite LIKE(X%) into a Range Query.
CP4.3b QEXE: Raw String Matching Performance (e.g. using SSE4.2).
CP4.3c QEXE: Regular Expression Compilation (JIT/FSA generation).

CP5 Correlated Subqueries. Efficiently handling dependent subqueries.

CP5.1 QOPT: Flattening Subqueries (into join plans).
CP5.2 QOPT: Moving Predicates into a Subquery.
CP5.3 QEXE: Overlap between Outer- and Subquery.

CP6 Parallelism and Concurrency. Making use of parallel computing resources.

CP6.1 QOPT: Query Plan Parallelization.
CP6.2 QEXE: Workload Management.
CP6.3 QEXE: Result Re-use.

Table 1. TPC-H Choke Point (CP) classification, and CP impact per query (white=light, gray=medium, black=strong).

TPC-H -- metrics

- Composite Query-Per-Hour Performance Metric

$$\text{QphH@Size} = \sqrt{Power\ @\ Size * Throughput\ @\ Size}$$

$$\text{TPC-H Power@Size} = \frac{3600 * SF}{\sqrt[24]{\prod_{i=1}^{i=22} QI(i,0) * \prod_{j=1}^{j=2} RI(j,0)}}$$

$$\text{TPC-H Throughput@Size} = (S*22*3600)/T_s *SF$$

- Price/Performance Metric

$$\text{TPC-H Price-per-QphH@Size} = \$/\text{QphH@Size}$$

- Availability Date
- Energy Metric Watts/KQphH@Size

From TPC-H to TPC-DS, Why?

- Linear scaling of tables
- Homogeneous data distribution
- Third Normal Form (3NF), rather than Star Schema
- Simple-structured ad-hoc queries, update workloads are simple

	TPC-H	TPC-DS
Schema type	3rd Normal Form	Multiple Snowflake
Number of tables	8	24
Number of columns (min)	3	3
Number of columns (max)	16	34
Number of columns (avg)	~ 7.6	18
Number of foreign keys	9	104

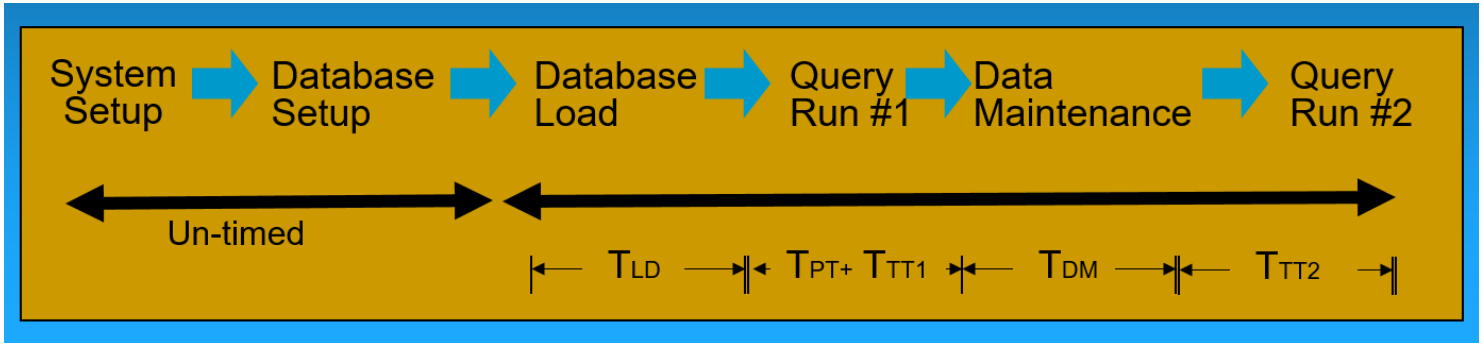
Figure from https://medium.com/hyrise/a-summary-of-tpc-ds-9fb5e7339a35

TPC-DS: A Decision Support Benchmark

- V1 during 2000-2012, introduce V2 in 2015 to support Hive/Hadoop
- Snowflake schema with 24 tables including 7 fact tables, e.g., sales, and 17 dimension tables
- More realistic data scaling with non-uniform distribution
- 99 query templates with 4 types, i.e., reporting, ad-hoc, iterative, and data mining

TPC-DS: Execution rules and Metrics

- Execution Rule:

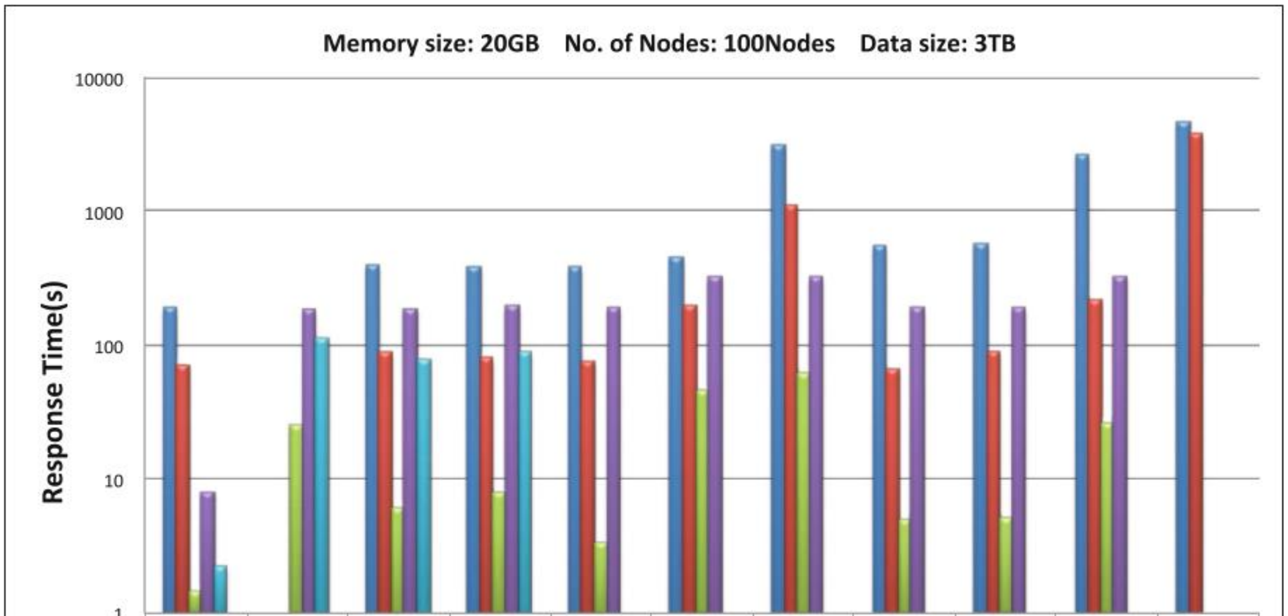


- Query-Per-Hour Performance Metric:

$$QphDS@SF = \left\lfloor \frac{SF * Q}{\sqrt[4]{T_{PT} * T_{TT} * T_{DM} * T_{LD}}} \right\rfloor$$

When SQL meets Hadoop

Evaluation by Yueguo Chen et.al at BPOE, 2014



	01.q5A	02.q5Ao	03.q5Ao1	04.q9A	05.q9A1	06.q9A2	07.q10Bo	08.q32B	09.q32Bg1	10.q6Cgo	11.q27Cgho
hive	188.90		394.15	380.45	374.85	452.81	3104.27	550.76	560.60	2630.28	4507.96
stinger	70.99		87.56	79.42	76.18	193.07	1104.75	65.29	88.64	216.74	3724.80
impala	1.47	25.16	6.13	7.77	3.36	45.09	60.79	4.97	5.09	26.17	
presto	8.00	184.00	183.00	193.00	191.00	320.00	319.00	191.00	190.00	319.00	
shark	2.27	111.58	78.08	89.30							

From TPC-DS to TPCx-BB

- An end-to-end application-level benchmark for Big Data Analytical Systems at 2016

- Based on TPC-DS, and Originate from the proposal of BigBench V1 at SIGMOD 2013

- With volume, variety and velocity.

The Volume of TPCx-BB

- Similar scale factors to TPC-DS, new data:
  - buyer clicks $c_b = |\text{web\_sales}| \times (\text{pages per item} + \frac{\text{pages per b}}{\text{items per s}}$
  - visitor clicks $c_v = (|\text{web\_sales}| \times \text{pages per item}) \times \text{visitor ratio}$
  - reviews: $|\text{reviews}| = |\text{items}| \times 5 + |\text{customers}| \times 0.2 + |\text{web\_sales}| \times 0.15$

- PDGF for parallel data generation
  - proposed by Tilmann Rabl et al, TPCTC 2010
  - scalable and extensible data generator
  - random seeding strategy

The Variety of TPCx-BB

- Data Model

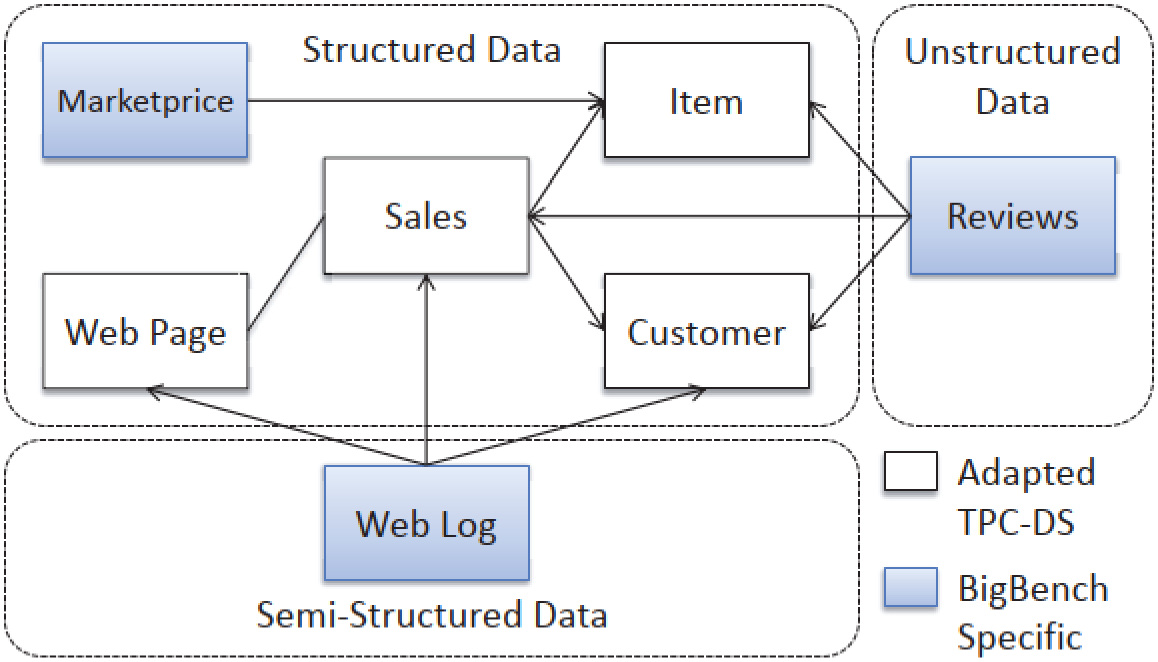


Figure 1: Big Data Benchmark Data Model

Figure from TPCx-BB documentation v1.4.0.

The Velocity of TPCx-BB

- A periodic data refresh process considering (i) the amount of data;(2) the time interval.

- Refresh velocities for each of data types $V_{structured} = 1$, $V_{unstructured} = 2$, $V_{semistructured} = 4$

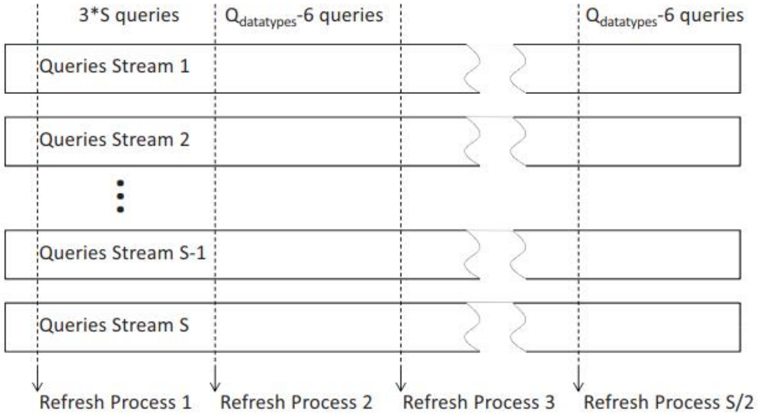


Figure 3: Scheduling of refresh processes based on executed queries per data type

Figure from BigBench paper at Sigmod 2013

TPCx-BB workloads

- 30 complex Queries, 10 of which are based on the TPC DS
- 5 business categories from Mckinsey's reports.
- 4 technique dimensions implemented by Hive Queries with MapReduce, NLP, and MLlib programs
- Metric: $$\mathrm{BBQpm@SF} = \frac{\mathrm{SF} * 60 * \mathrm{M}}{\mathrm{T_{LD}} + \sqrt[2]{\mathrm{T_{PT}} * \mathrm{T_{TT}}}}$$

Table 3: Business Categories Query Breakdown

Business category	Total	Percentage(%)
Marketing	18	60.0
Merchandising	5	16.7
Operations	4	13.3
Supply chain	2	6.7
New business models	1	3.3

Table 4: Technical Dimensions Breakdown

Query processing type	Total	Percentage(%)
Declarative	10	33.3
Procedural	7	23.3
Mix of Declarative and Procedural	13	43.3
Data sources	**Total**	**Percentage(%)**
Structured	18	60.0
Semi-structured	7	23.3
Un-structured	5	16.7
Analytic techniques	**Total**	**Percentage(%)**
Statistics analysis	6	20.0
Data mining	17	56.7
Reporting	8	26.7

Tables from BigBench paper at Sigmod 2013

An example of TPCx-BB workload

Q10: For all products, extract sentences from its product reviews that contain positive or negative sentiment and display for each item the sentiment polarity of the extracted sentences (POS OR NEG) and the sentence and word in sentence leading to this classification.

```
SELECT pr_item_sk, out_content, out_polarity,
            out_sentiment_words
FROM ExtractSentiment
(
    ON product_reviews
    TEXT_COLUMN ('pr_review_content')
    MODEL ('dictionary')
    LEVEL ('sentence')
    ACCUMLATE ('pr_item_sk')
)
WHERE out_polarity = 'NEG' or out_polarity = 'POS';
```

Most recent results of TPCx-BB

TPCx-BB Ten Most Recently Published Results

Version Results   As of 11-Nov-2020 at 8:54 AM [GMT]

Note 1: The TPC believes that comparisons of TPCx-BB results measured against different database sizes are misleading and discourages such comparisons.
Note 2: The TPC believes it is not valid to compare prices or price/performance of results in different currencies.

Date Submitted	Scale Factor	Company	System	BBQpm	Price/BBQpm	Watts/BBQpm	System Availability	DBMS Software (Big Data Software Framework)	Operating System	Nodes
10/02/20	SF100000	Alibaba.com	Alibaba Cloud MaxCompute	26,501.53	138.66 USD	NR	10/02/20	Alibaba Cloud MaxCompute	Alibaba Group Enterprise Linux Server 7.2 (Paladin)	70
09/25/20	SF30000	Alibaba.com	Alibaba Cloud MaxCompute	9,296.45	115.71 USD	NR	10/02/20	Alibaba Cloud MaxCompute	Alibaba Group Enterprise Linux Server 7.2 (Paladin)	20
10/11/19	SF10000	DELL	Dell 14G R640/R740xd	3,089.93	377.46 USD	NR	10/11/19	Hortonworks Data Platform 3.0	Red Hat Enterprise Linux 7.6	19
09/17/19	SF30000	Alibaba.com	Alibaba Cloud MaxCompute	6,427.86	169.76 USD	NR	09/18/19	MaxCompute v3.31	Alibaba Group Enterprise Linux Server 7.2 (Paladin)	15
09/17/19	SF100000	Alibaba.com	Alibaba Cloud MaxCompute	25,641.21	224.49 USD	NR	09/18/19	MaxCompute v3.31	Alibaba Group Enterprise Linux Server 7.2 (Paladin)	100
07/14/19	SF30000	Lenovo	ThinkSystem SR650	3,767.91	380.55 USD	NR	07/15/19	Cloudera for Apache Hadoop (CDH) 5.12.1	Red Hat Enterprise Linux 7.6	39
05/07/19	SF10000	Hewlett Packard Enterprise	Hewlett Packard Enterprise ProLiant DL Gen10 for B	1,789.75	510.19 USD	NR	05/07/19	Cloudera Enterprise 5.16.x	Red Hat Enterprise Linux 7.6	21
03/22/18	SF10000	DELL	Dell 14G R640/R740xd	1,660.75	546.82 USD	NR	03/22/18	Cloudera Distribution for Apache Hadoop (CDH) 5.13.1	Red Hat Enterprise Linux Server 7.3	19
01/21/18	SF30000	Sugon	Sugon Cluster	3,383.95	307.86 USD	NR	01/22/18	Cloudera for Apache Hadoop (CDH) 5.11.1	Red Hat Enterprise Linux Server 7.3	33

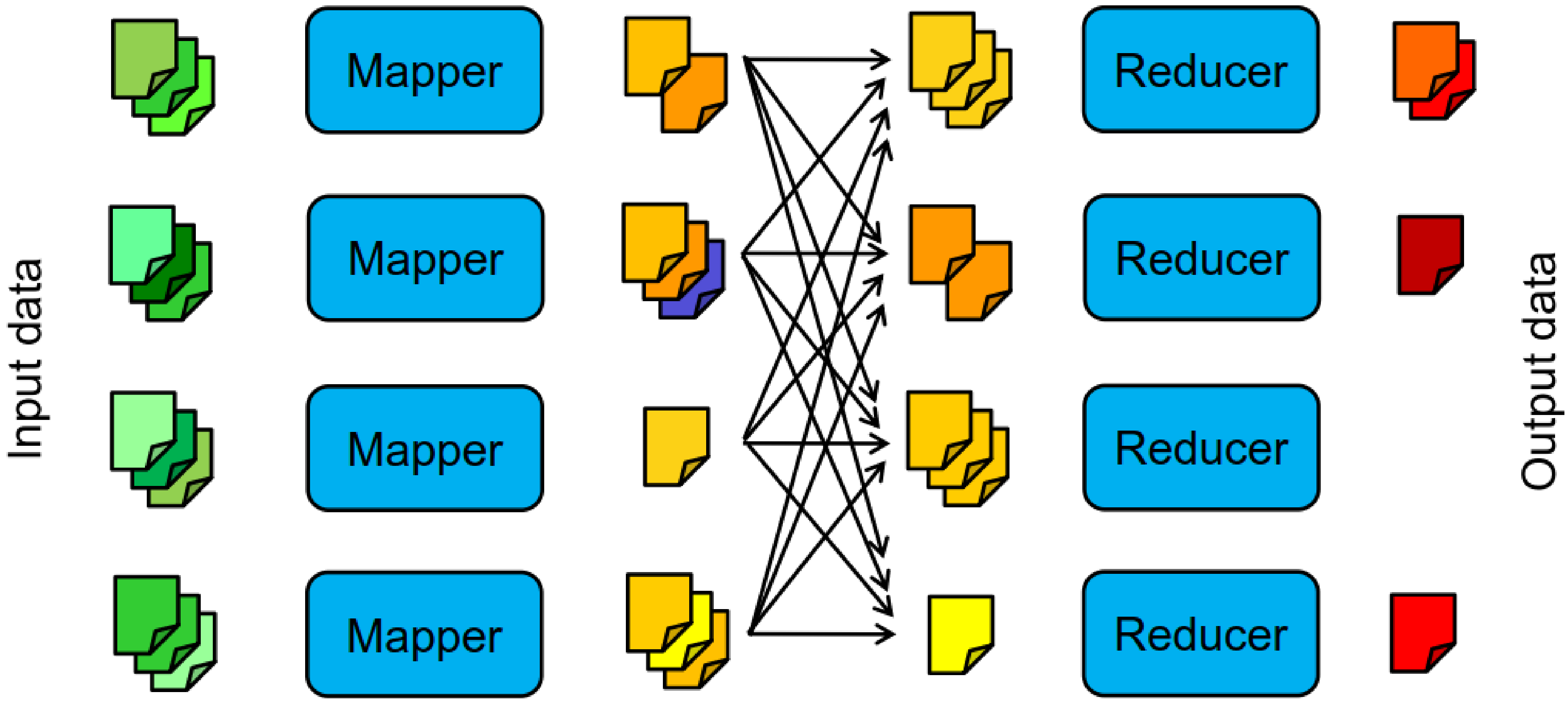
Link： http://tpc.org/tpcx-bb/results/tpcxbb_last_ten_results5.asp

Outline

- Introduction to Big Data System Benchmarking(15')
- Benchmarking SQL Big Data Analytical Systems(25')
- Benchmarking Map-Reduce/NoSQL Systems(15')
- Benchmarking Graph-based Big Data Systems(20')
- Benchmarking Multi-Model Big Data Systems(35')
- Open Challenges and Future Directions(10')

Map-Reduce Paradigm

- Proposed by Google at OSDI 2004



Benchmarking Map-Reduce BDS

- Representative Open-source Systems: Hadoop, Tez, Hive, Spark, etc.
- We will look at two benchmarks: AMP big data benchmark and HiBench
- We will discuss the main findings from their existing evaluation

AMP Big Data Benchmark

https://amplab.cs.berkeley.edu/benchmark/#

- Originate from the paper "A Comparison of Approaches to Large-Scale Data Analysis" by Pavlo et al. *SIGMOD 2009*

- Three datasets: (1) a set of unstructured HTML documents; two SQL tables, (2) *Rankings with pagerank* and (3) *UserVisits*.

- Four queries for selection, join, aggregation, and UDF tasks, respectively

When MR meets Parallel DBMSs

Evaluation by Andrew Pavlo et.al at SIGMOD, 2009

Benchmark performance on a 100-node cluster.

	Hadoop	DBMS-X	Vertica	Hadoop/DBMS-X	Hadoop/Vertica
Grep	284s	194s	108x	1.5x	2.6x
Web Log	1,146s	740s	268s	1.6x	4.3x
Join	1,158s	32s	55s	36.3x	21.0x

Figure from Michael Stonebraker et.al from ACM communication, 2010

When MR meets Parallel DBMSs

Insights form Michael Stonebraker et.al from ACM communication, 2010

1. MR processing model is slower because of (1) repetitive record parsing;(2) write the intermediate results (3) block-based scheduling
2. Parallel DBMSs need one-button installs, automatic tuning, better documentation.
3. Parallel DBMSs excel at efficient querying of large data sets; MR-style systems excel at complex analytics and ETL tasks.

1. The best solution is to combine Parallel DBMSs with MR framework e.g., HadoopDB, Hive, Aster, Greenplum, Cloudera, and Vertica

HiBench

- A big data benchmark suite with four categories

TABLE I
CONSTITUENT BENCHMARKS

Category	Workload
Micro Benchmarks	Sort WordCount TeraSort
Web Search	Nutch Indexing PageRank
Machine Learning	Bayesian Classification K-means Clustering
HDFS Benchmark	EnhancedDFSIO

Figure from HiBench paper from ICDE 2010

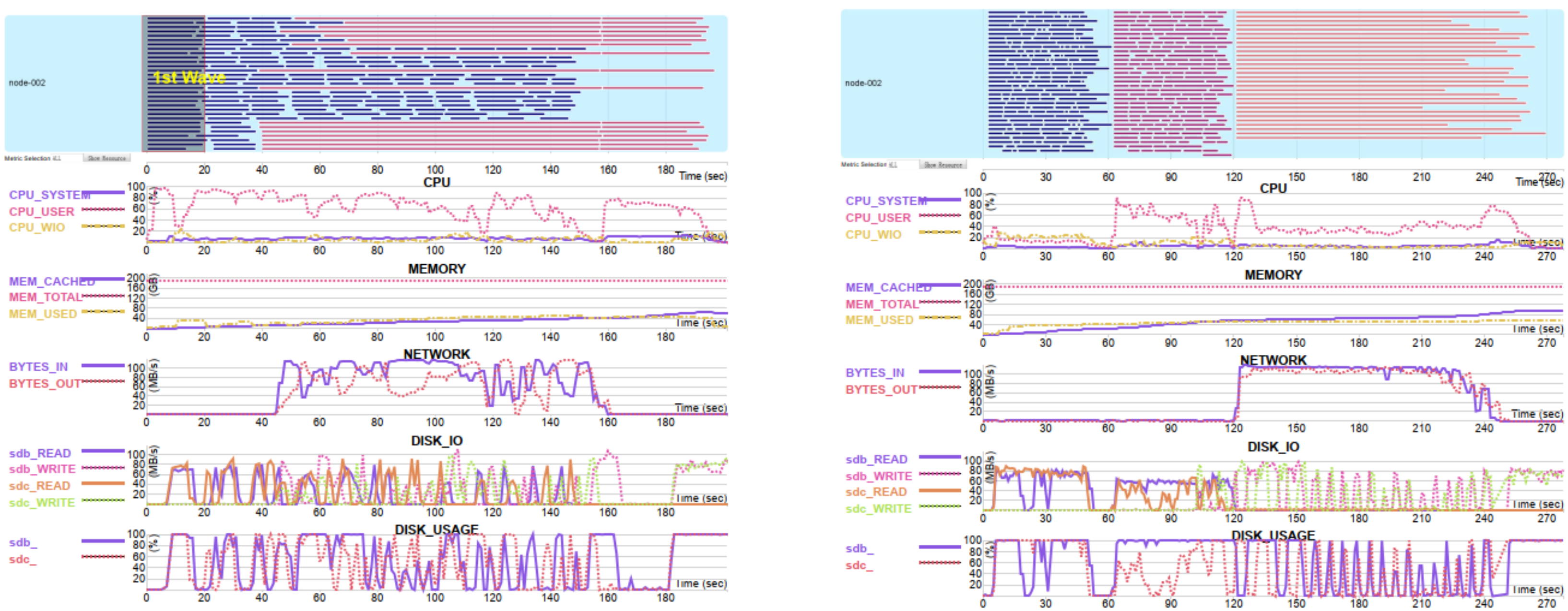
- HiBench 7.1 with a streaming workload and a parallel graph algorithm

When MapReduce meets Spark

Evaluation by Juwei Shi et.al at VLDB 2015

1. Spark is about 2.5x, 5x, and 5x faster than MapReduce, for WordCount, K-means, and PageRank, respectively.

1. MapReduce is 2x faster than Spark in Sort workload.



(a) MapReduce (b) Spark

Figure 2: The Execution Details of Sort (100 GB Input)

Other Big Data Benchmark Solutions

- Yahoo! Cloud Serving Benchmark (YCSB)
- Yahoo Streaming Benchmark
- BigDataBench
- ...

Benchmarking NoSQL BDS: YCSB

- Yahoo! Cloud Serving Benchmark
- Aim for Cloud-based OLTP
- Metrics: Throughput, scalability, elasticity
- Extensions include YCSB++, YCSB+T, etc.

Workload	Operations	Record selection	Application example
A—Update heavy	Read: 50% Update: 50%	Zipfian	Session store recording recent actions in a user session
B—Read heavy	Read: 95% Update: 5%	Zipfian	Photo tagging; add a tag is an update, but most operations are to read tags
C—Read only	Read: 100%	Zipfian	User profile cache, where profiles are constructed elsewhere (e.g., Hadoop)
D—Read latest	Read: 95% Insert: 5%	Latest	User status updates; people want to read the latest statuses
E—Short ranges	Scan: 95% Insert: 5%	Zipfian/Uniform*	Threaded conversations, where each scan is for the posts in a given thread (assumed to be clustered by thread id)

Figure from YCSB paper from Socc 2010

Yahoo Streaming Benchmark

https://github.com/yahoo/streaming-benchmarks

- Simulate a simple advertisement application

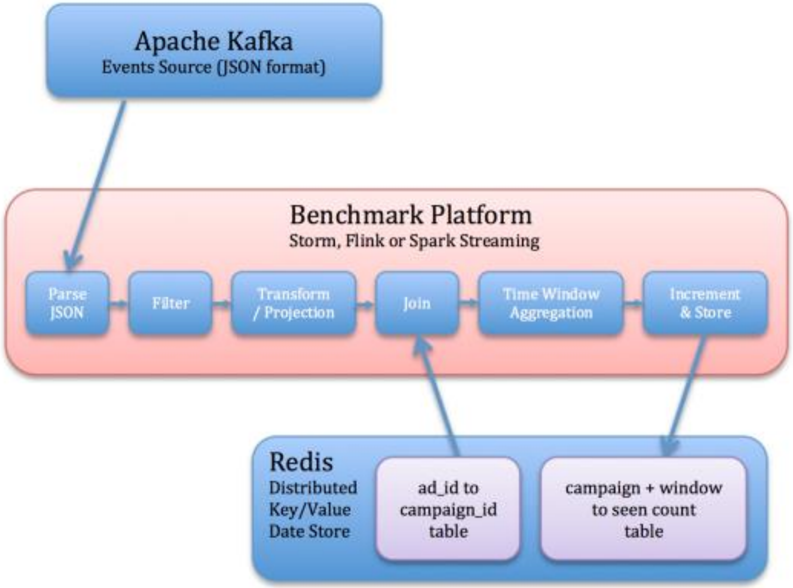


Figure from
https://developer.yahoo.com/blogs/135370591481/?guccounter=1&guce_referrer=aHR0cHM6Ly93d3cuZ29vZ2xlLmNvbS8&guce_referrer_sig=AQAAADn8heo1tz6UYHLnFbHQoa_bxkN_ouhjHJLNSj1XPv2_zwJTsFg6qvPJkD-nz75FhWkZ7JYO3WUvltEMa_rLVPHCyBFb3AzniLFLHJmNoegeeG6aWhiMYuwINEizGtr61AjtTgfNgvVfmfzMn-a9Rsp7-W_HBX-Lx3gyFAZ36Uqp

Big Data Benchmark: BigDataBench

- by Jianfeng Zhan et al, Chinese Academy Sciences
- International Open Benchmark Council (BenchCouncil)
- AIBench with 17 AI-based tasks https://www.benchcouncil.org/AIBench/index.html

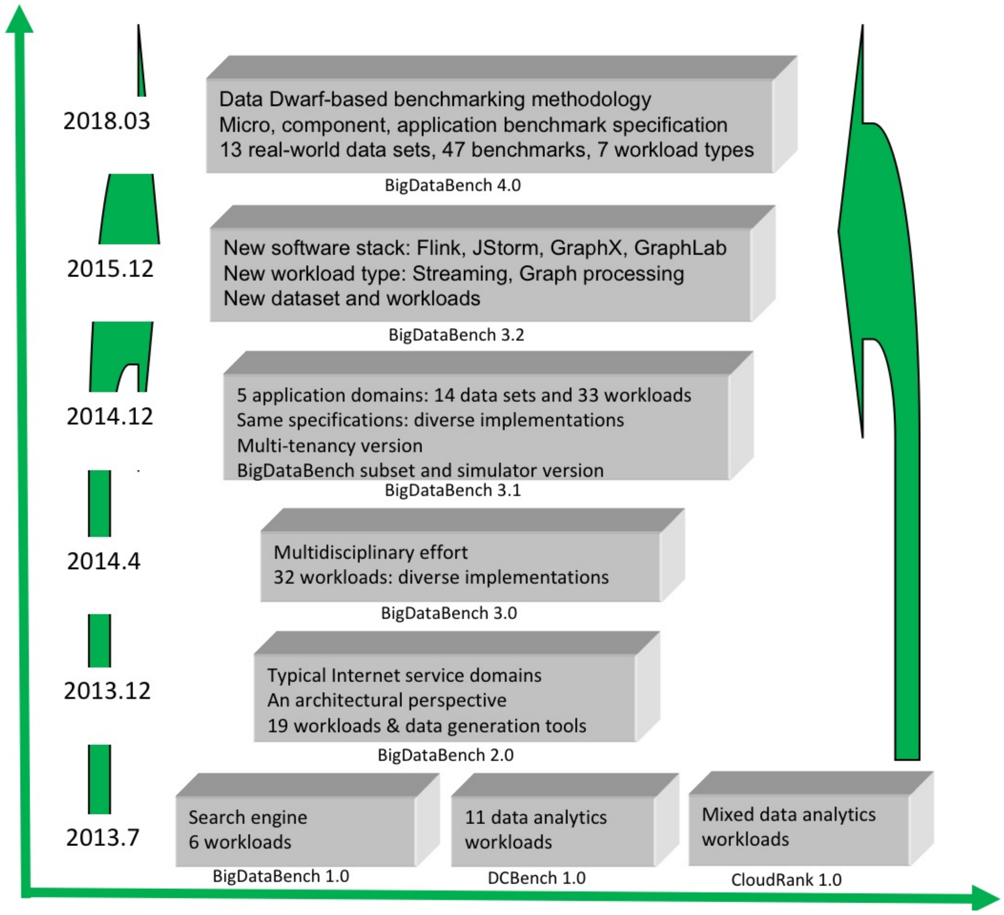


Figure from https://www.benchcouncil.org/BigDataBench/index.html

Outline

- Introduction to Big Data System Benchmarking(15')
- Benchmarking SQL Big Data Analytical Systems(25')
- Benchmarking Map-Reduce/NoSQL Systems(15')
- Benchmarking Graph-based Big Data Systems(20')
- Benchmarking Multi-Model Big Data Systems(35')
- Open Challenges and Future Directions(10')

Graph databases vs. Graph processing systems

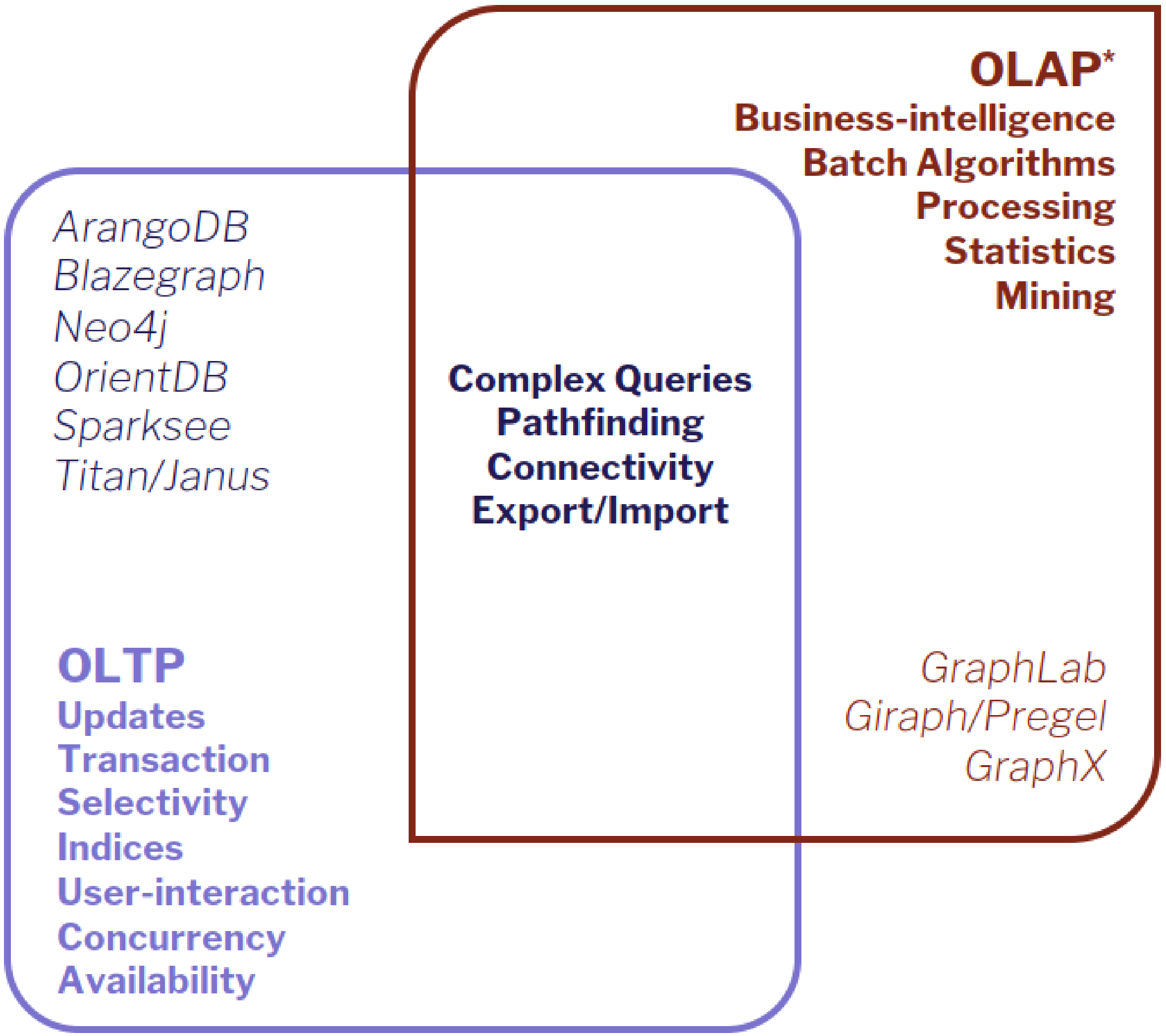


Figure from Graph Databases Evaluation – Matteo Lissandrini

LDBC The graph & RDF benchmark reference

- Linked Data Benchmark Council (LDBC)
- LDBC Social Network Benchmark
- LDBC Graphalytics Benchmark
- LDBC Semantic Publishing Benchmark
- Link: http://ldbcouncil.org/benchmarks

LDBC Social Network Benchmark

- A data model of social network with 14 entities, e.g., persons, and 20 relations, e.g., knows
- A synthetic data generator with scale factors
- Interactive workloads with 14 complex queries, 7 short read operations and 6 update operations
- Business workloads with 25 complex queries
- Choke point designs with 8 categories

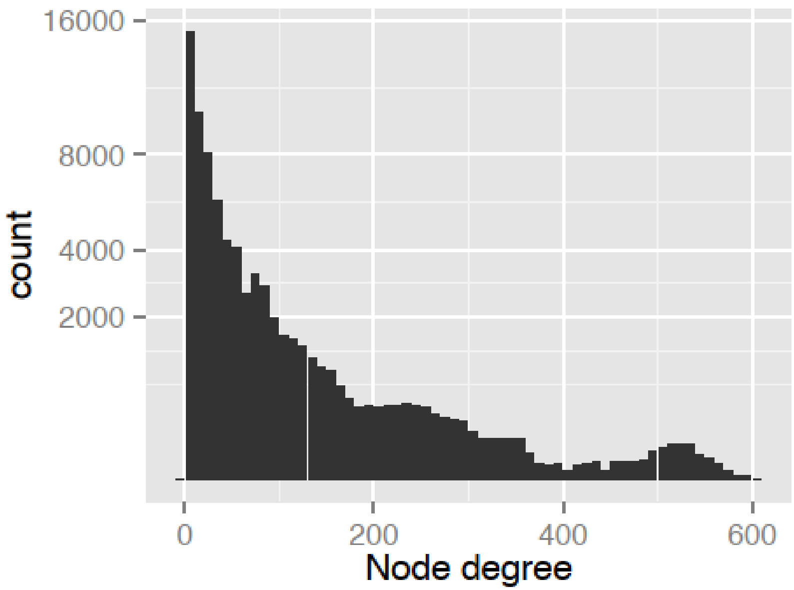
LDBC Data Generation

- Value correlation with power-law distribution
- Implementation based on Hadoop

Name	Number
Karl	215
Hans	190
Wolfgang	174
Fritz	159
Rudolf	159
Walter	150
Franz	115
Paul	109
Otto	99
Wilhelm	74

Name	Number
Yang	961
Chen	929
Wei	887
Lei	789
Jun	779
Jie	778
Li	562
Hao	533
Lin	456
Peng	448

Table 2: Top-10 person.firstNames (SF=10) for persons with person.location=Germany (left) or China (right).



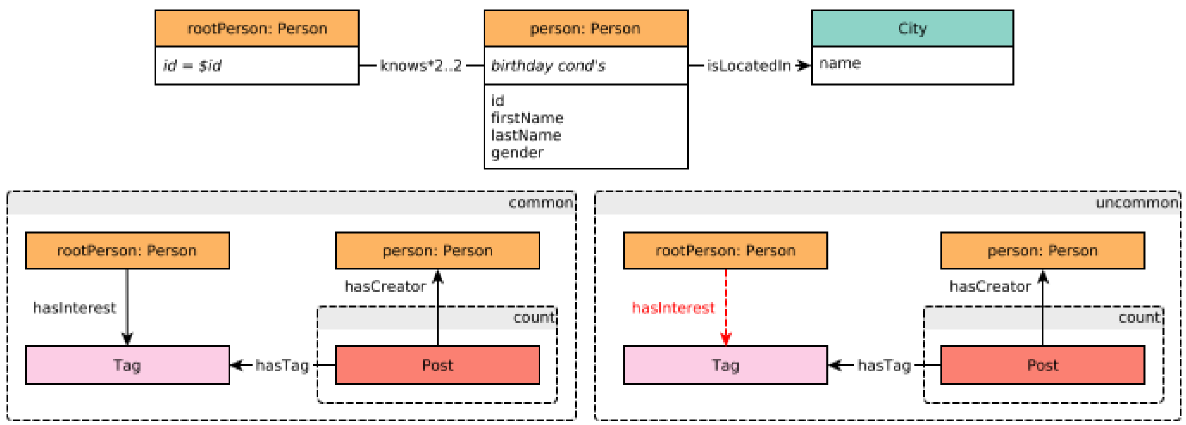
Figures from LDBC paper at SIGMOD 2015

LDBC Choke Point Designs

Inspired by the TPC-H choke point designs

1. Aggregation Performance
2. Join Performance
3. Data Access Locality
4. Expression Calculation
5. Correlated Sub-queries
6. Parallelism and Concurrency

LDBC Choke Point Designs (con’t)

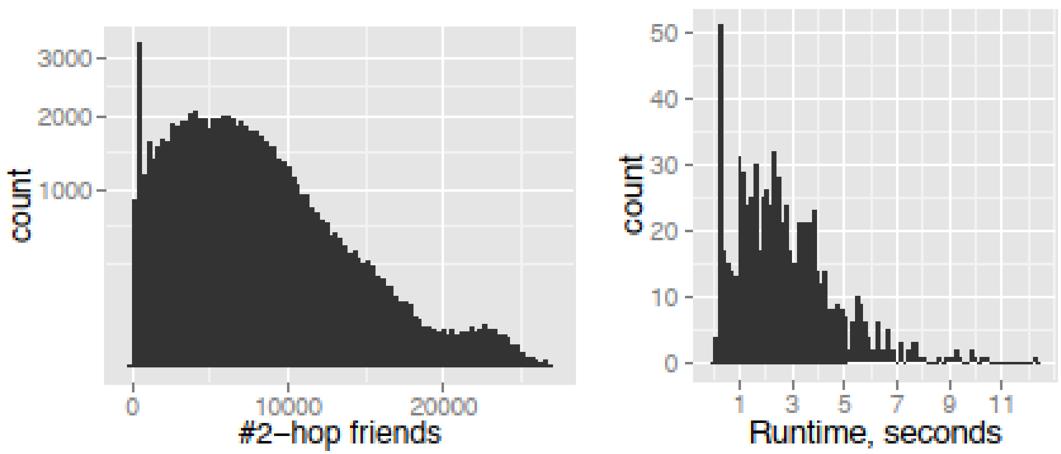
<table>
<tr><td>query</td><td>Interactive / complex / 10</td></tr>
<tr><td>title</td><td>Friend recommendation</td></tr>
<tr><td>pattern</td><td>
</td></tr>
<tr><td>desc.</td><td>Given a start Person with id <code>personId</code>, find that Person’s friends of friends (<code>person</code>) – excluding the start Person and his/her immediate friends –, who were born on or after the 21st of a given <code>month</code> (in any year) and before the 22nd of the following month. Calculate the similarity between each <code>person</code> and the start Person, where <code>commonInterestScore</code> is defined as follows:

<code>common</code> = number of Posts created by <code>person</code>, such that the Post has a Tag that the start Person, is interested in
<code>uncommon</code> = number of Posts created by <code>person</code>, such that the Post has no Tag that the start Person, is interested in
<code>commonInterestScore</code> = <code>common</code> - <code>uncommon</code>
</td></tr>
</table>

Figure from LDBC SNB Specification v0.3.2

LDBC Parameter Curation

Q2: Given a start Person, find the top 20 Forums the friends and friends of friends of that Person joined after a given Date.



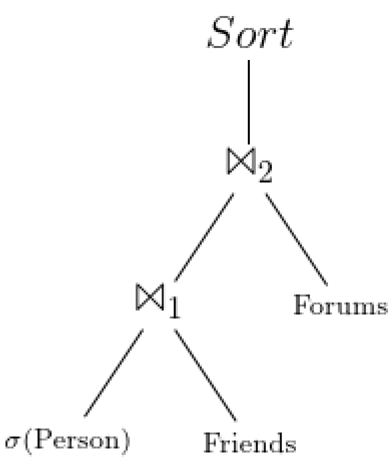
(a) Distribution of size of 2-hop friend environment (SNB SF10)
(b) Query 5 runtime distr.

Figure 5: Correlations cause high runtime variance (Q5)

Figure from LDBC paper at SIGMOD 2015

LDBC Parameter Curation (con’t)

- Problem: select a subset S of size k in the PC table such that the variance across all columns is minimized.
- Solution: A greedy-based method



(a) Intended Plan

PersonID	$\lvert\bowtie_1\rvert$	$\lvert\bowtie_2\rvert$
...	...	...
1542	60	99
1673	60	102
7511	60	103
958	60	120
1367	61	101
...	...	...

(b) Parameter-Count table

Figure 6: Parameter Curation for Query 2

Figure from LDBC paper at SIGMOD 2015

LDBC Graphalytics

- 6 real datasets and 2 synthetic generators
- 6 implementations, e.g., Giraph, GraphX
- 6 graph algorithms
  - Breadth-first search (BFS)
  - PageRank (PR)
  - Weakly connected components (WCC)
  - Community detection using label propagation
  - Local clustering coefficient (LCC)
  - Single-source shortest paths (SSSP)

LDBC Graphalytics (con't)

Metrics: processing time, makespan, scalability

Table 8: T_{proc} and makespan for BFS on D300(L).

Time	Giraph	GraphX	P'Graph	G'Mat(S)	OpenG	PGX(S)
Makespan	277.9 s	278.4 s	216.5 s	23.3 s	5.7 s	14.3 s
T_{proc}	23.4 s	97.9 s	2.1 s	0.3 s	1.9 s	0.05 s
Ratio	8.4%	35.2%	1.0%	1.3%	33.3%	0.3%

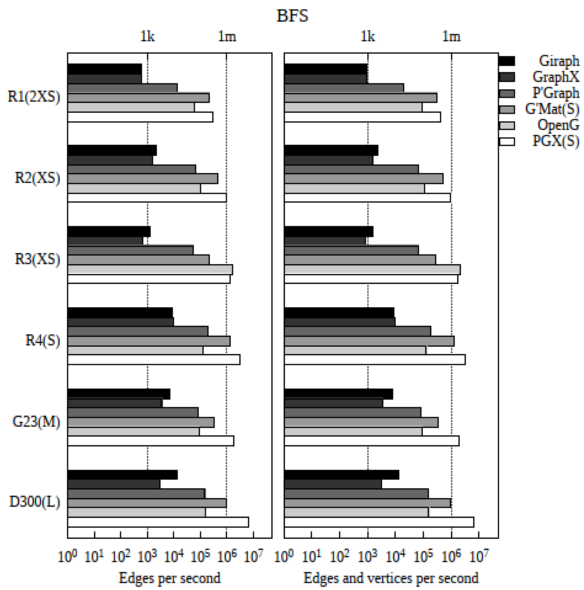


Figure 5: Dataset variety: EPS and EVPS for BFS.

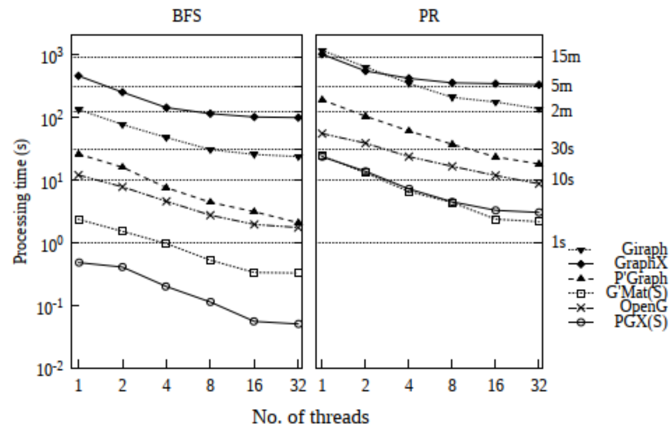


Figure 7: Vertical scalability: T_{proc} vs. #threads.

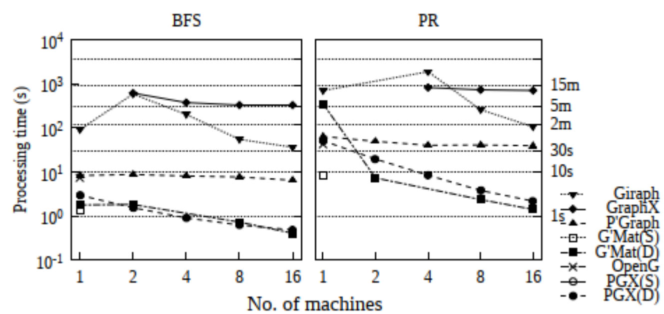


Figure 8: Strong scalability: T_{proc} vs. #machines.

Figures from LDBC paper at PVLDB 2016

Other Big Graph Benchmarks

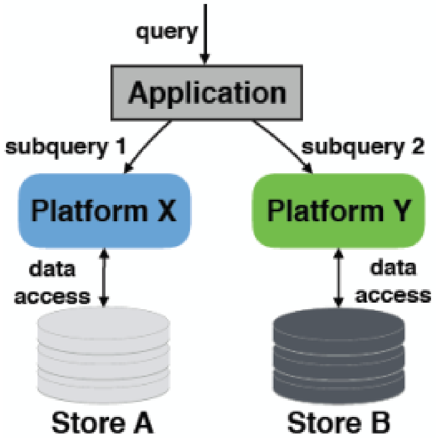
- LinkedBench and BG for social network
- Graph 500 for graph analytics
- LUBM, BSBM, and SP2Bench for RDF
- GMARK for graph query generation
- ...

Outline

- Introduction to Big Data System Benchmarking(15')
- Benchmarking SQL Big Data Analytical Systems(25')
- Benchmarking Map-Reduce/NoSQL Systems(15')
- Benchmarking Graph-based Big Data Systems(20')
- Benchmarking Multi-Model Big Data Systems(35')
- Open Challenges and Future Directions(10')

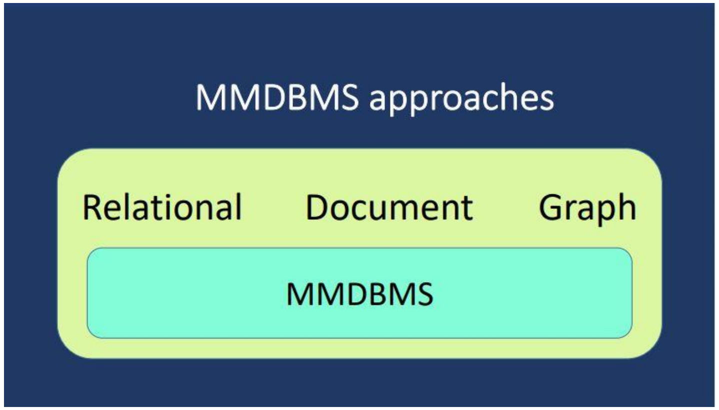
MMDBMS : one size fits a bunch

One size doesn't fit all

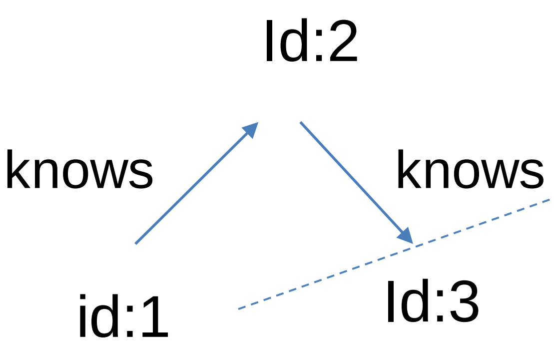


(d) Polystore

One size fits a bunch



Consider a social commerce scenario



{"Order_no": "20201212",
“customer_no":1,
“orderlines":[
{"item_no":"cn2523",
"type":“cellphone",
“name”: “ Huawei P40"},
{"item_no":“cn2526“,
"type":“headphone",
"name": “Freebuds 3"}]
}

id	name	credit_limits
1	James	2,000
2	David	3,000
3	Mary	5,000

An example of multi-model query (ArangoDB)

Product recommendation: recommend the bought cellphones by James to their 2-hop friends whose credit limit is greater than 3000.

AQL:

For c in customers For o in orders For f in outbound 2..2 c.id GRAPH Knows

Filter c.name=="James" and f.credit_limits>3000 and o.customer_no==c.id and o.orderline[*].type=="cellphone“ and f.id==c.id

Return {friend:f, orders:o}

Benchmarking Multi-Model BDS

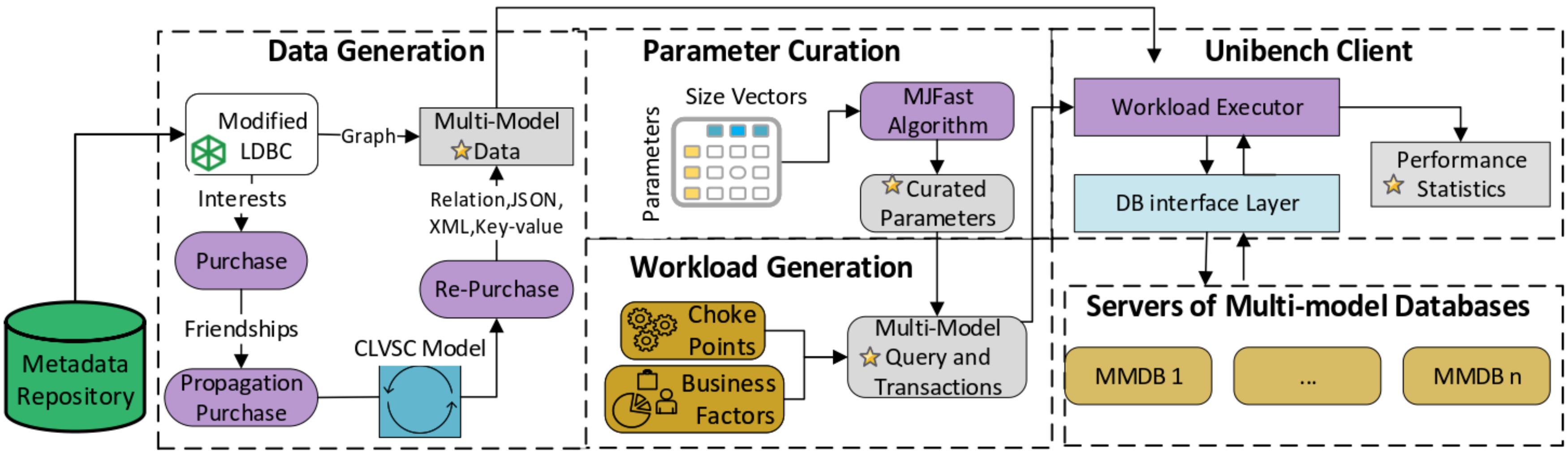
Table 1: Comparison of Multi-Model DBMSs

System	Query Language	Primary Model	Secondary Model	Storage Strategy
AgensGraph	OpenCypher, SQL	Relational	Graph, JSON	[illegible]
ArangoDB	AQL	JSON	[illegible]	[illegible]
OrientDB	SQL-like	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible] Key-value	Multiple Engines
[illegible]	[illegible]	-	**ALL** but XML	Multiple Engines
[illegible]	SQL-extension	Relational	**ALL**	**Both**

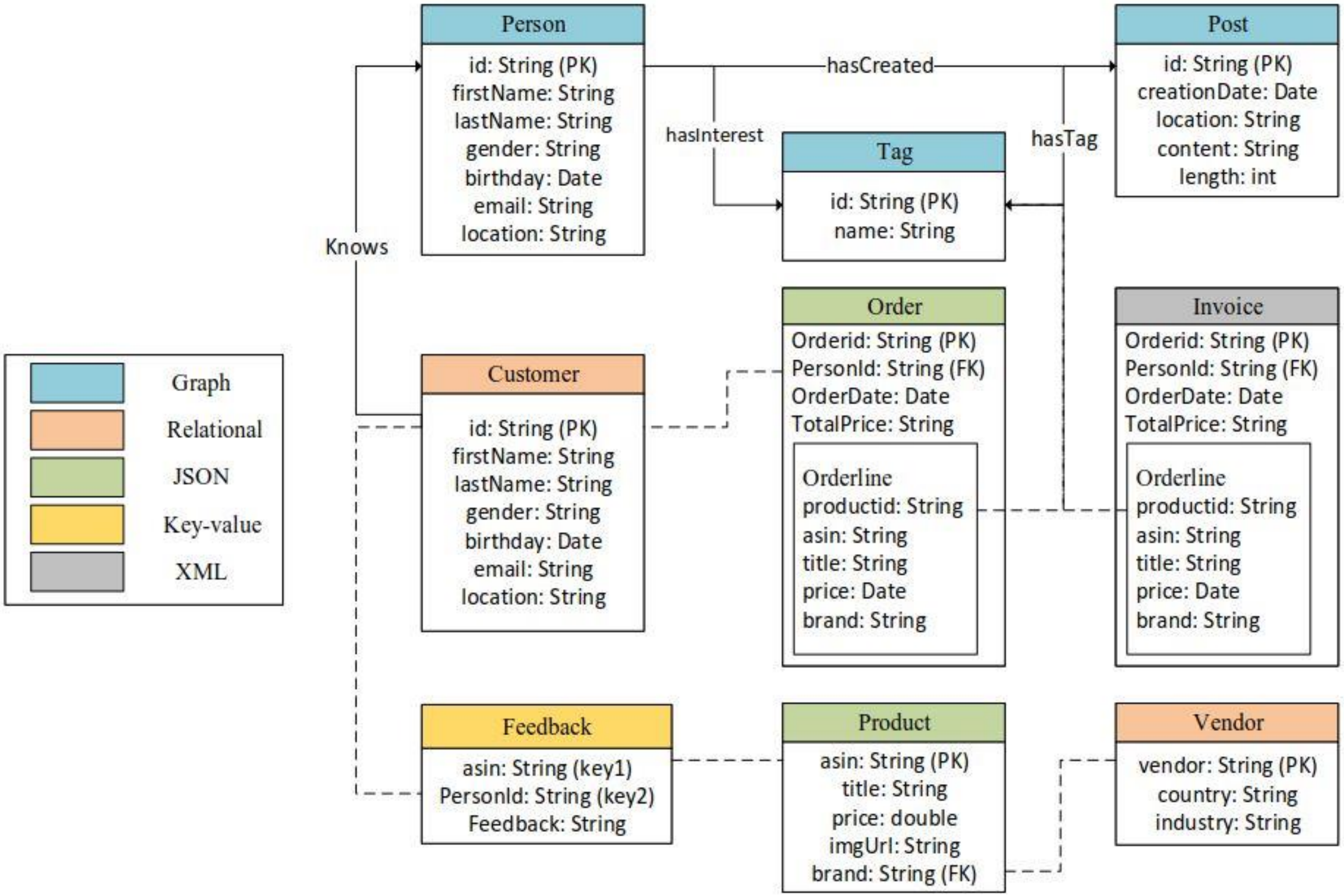
There is a need for an end-to-end benchmark for multi-model databases.

UniBench to the rescue

Three key components: Data generation, Workload generation, and Parameter Curation



Data Schema of UniBench



Three-phase data generation

(1) Purchase : interest-oriented transaction

$$p(x) \propto \frac{\alpha - 1}{x_{min}} \left(\frac{x}{x_{min}} \right)^{-\alpha}$$

(2) Pro-purchase: friend-influenced transaction

$$S_{ui} = \sum_k k \times Pr(R_{ui} = k | A = a_u) + E(R_{vi} : \forall v \in N(u))$$

(3) Re-purchase: probabilistic transaction

$$S_{ib}(CLVSC) = E(X^* \mid n^*, x', n, m, \alpha, \beta, \gamma, \delta) \times (E(M \mid p, q, \upsilon, m_x, x) + E(S \mid \bar{s}, \theta, \tau))$$

Realistic correlated distributions

Name	Country	DBpedia Ranking
Li	China	1
Chen	China	2
Zhang	China	3
Andy	USA	4
Olivia	USA	5

LDBC

Country	Brand	Wikidata Ranking
USA	Nike	1
USA	Adidas	2
China	Peak	3
China	Anta	4
China	361 degree	5

DBpedia

Brand	Products	Sales Ranking
Nike	B0000001	1
Adidas	B0000010	2
Peak	B0000100	3
Anta	B0001000	4
361 degree	B0010000	5

Amazon review

DATAGEN: scaling with scale factor

Table 2. Characteristics of datasets.

SF	Generation time(min)	Number ($\times 10^4$) & Size in Megabytes				
		Relational entries	Key-value pairs	JSON objects	XML objects	Nodes and Edges of Graph
1	10	1.2 & 1.1	25.2& 233.7	25.2& 219.2	25.2& 326.5	(123.1, 338.9) & 236.6
10	40	7.4 & 6.5	234.2 & 2313.1	234.2& 2189.8	234.2& 3568.6	(969.3, 3208.3) & 2095.8
30	60 (3 nodes)	18.3 & 15.8	636.8& 6367.8	636.8& 6184.9	636.8& 11771.3	(2674.3, 10951.5) & 6191.5

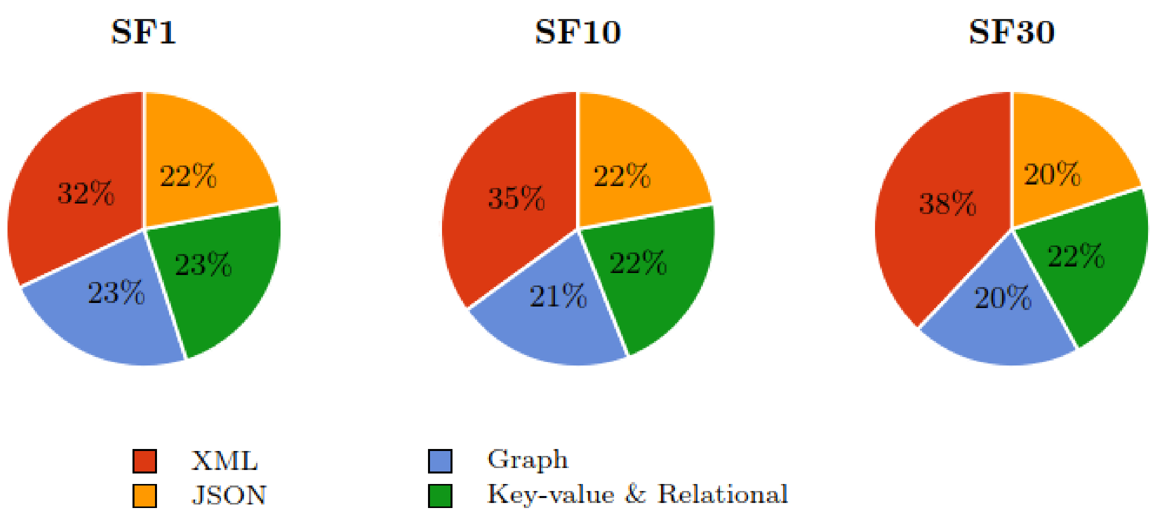


Figure 7. Multi-model distribution of the generated dataset.

Choke-point designs

- Choosing the right join type and order
- Performing complex aggregation
- Ensuring the consistency and efficiency

Choke-point design: join ordering

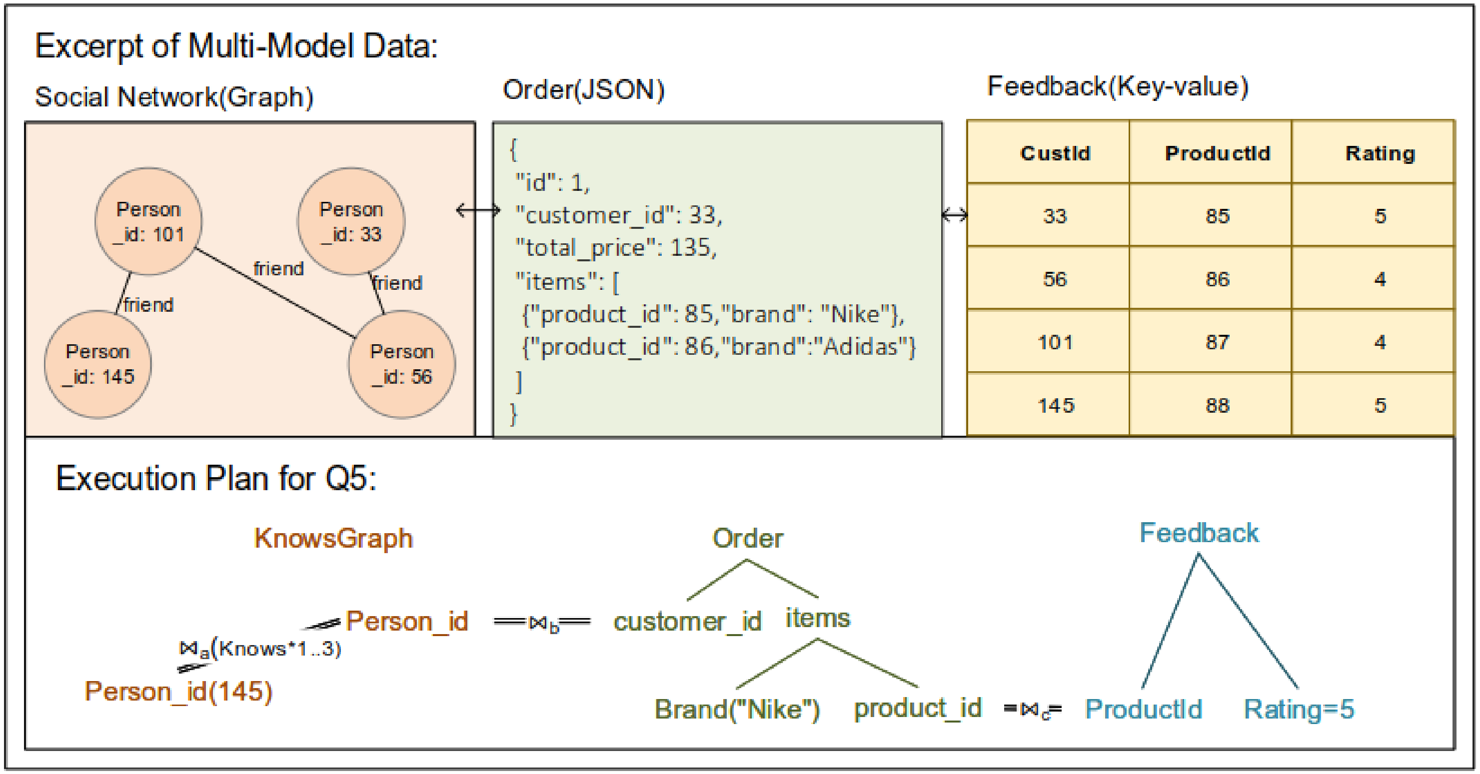
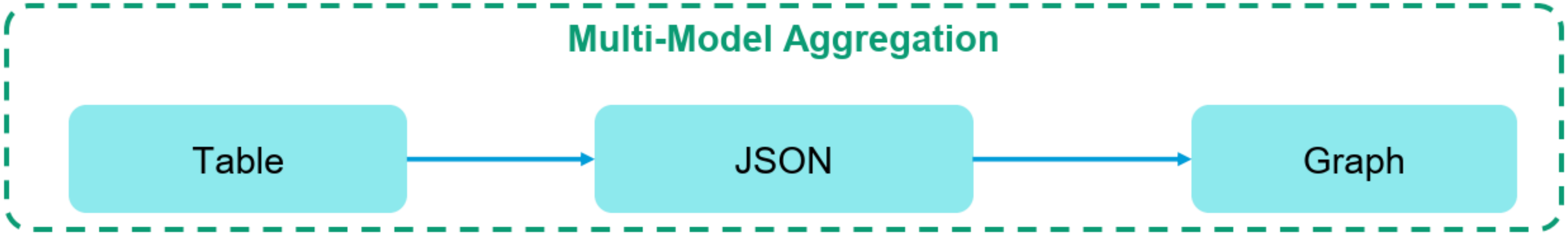


Fig. 3: Example of multi-model join

Choke-point design: aggregation

For all the products of a given brand during a given year, compute its total sales amount, and measure its popularity in the social media.



Choke-point design: transaction

New Order:

(i) create and insert the order

(ii) update the quantity of involved products,

(iii) insert the invoice.

Payment.

(i) retrieve the unpaid order,

(ii) update the balance of the seller and buyer,

(iii) update the order status to paid,

(iv) update the related invoice.

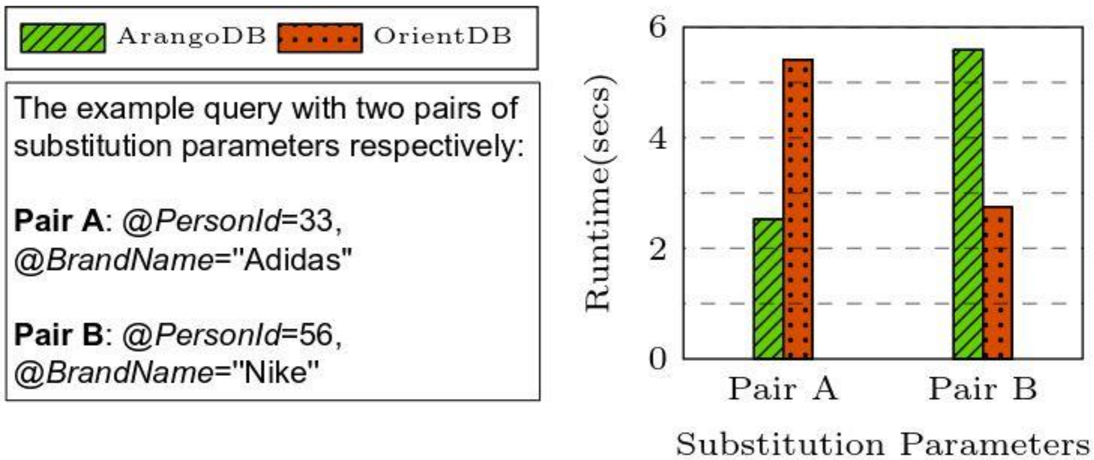
An overview of the Workloads

UniBench Workload:
4 business categories,
10 queries,
2 transactions

Table 2: Characteristics of Workload

Label	Business category	Technique dimension	Description
Q1	Individual	Perform point query on a customer's all multi-model data.	For a given *customer*, find her **profile**, **orders**, **feedback**, and **posts**.
Q2	Conversation	Join data from Relation, Graph, and JSON.	For a given *product*, find the **persons** who had bought it and posted on it.
Q3	Conversation	Join data from Relation, Graph, and Key-value, filter structured and unstructured data.	For a given *product*, find **persons** who have commented and posted on it, and detect negative **sentiments** from them.
Q4	Community	Aggregate and sort the JSON order, Perform the 3-hop graph traversal in the subgraph, return the intersection of two sets.	Find the top-2 **persons** who spend the highest amount of money in orders. Then for each person, traverse her knows-graph with 3-hop to find the friends, and finally return the common **friends** of these two persons.
Q5	Community	Join data from Relation, Graph, and Key-value with two predicates, recursive path query for Graph, embedded array operation for JSON, and composited-key lookup for Key-value.	Given a start *customer* and a product *category*, find **persons** who are this customer's friends within 3-hop friendships in knows-graph, and they have bought products in the given category. Finally, return **feedback** with the 5-rating review of those bought products.
Q6	Community	Perform the shortest path calculations between two nodes, find the correlated JSON orders of nodes in the path, aggregation on returned JSON orders.	Given *customer 1* and *customer 2*, find persons in the **shortest path** between them in the subgraph, and return the TOP 3 **best sellers** from all these persons' purchases.
Q7	Commerce	Join data from Relation, JSON and Key-value, compare the aggregation results between two periods, identify the reviews with negative sentiment.	For the *products* of a given *vendor* with declining sales compare to the former quarter, analyze the **reviews** for these items to see if there are any negative sentiments.
Q8	Commerce	Perform the embedded array filtering and aggregation on JSON order, aggregate the correlated graph data for each records.	For all the *products* of a given *category* during a given year, compute its **total sales amount**, and measure its **popularity** in the social media.
Q9	Commerce	Perform the embedded array filtering, aggregation, and sorting on JSON order, then find the correlated graph data.	Find top-3 **companies** who have the largest amount of sales at one *country*, for each company, compare the number of the male and female customers, and return the most recent **posts** of them.
Q10	Commerce	Perform the aggregation and sort on graph data, then find the correlated Key-value and JSON data.	Find the top-10 most active **persons** by aggregating the *posts* during the last year, then calculate their **RFM (Recency, Frequency, Monetary) value** in the same period, and return their recent **reviews** and tags of **interest**
T1	New Order Transaction	Check the ACID properties and evaluate the efficiency on read-heavy multi-model transaction that involves JSON and XML.	(i) create and insert the **order**, (ii) update the quantity of involved **products**, (iii) insert the **invoice**.
T2	Payment Transaction	Check the ACID properties and evaluate the efficiency on write-heavy multi-model transaction that involves Relation, JSON and XML.	(i) retrieve the unpaid **order**, (ii) update the balance of the **seller** and **buyer**, (iii) update the **order** status to paid, (iv) update the related **invoice**.

Parameter Curation



Observation: The same queries with different parameters differ in sizes of intermediate results.

Parameter Curation (con'd)

Query: For a person **p** and a product brand **b**, find p's friends who have bought products with brand b.

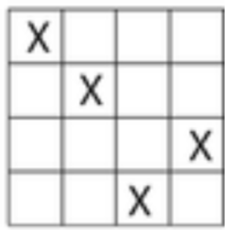
Parameter Domain | Size Vector

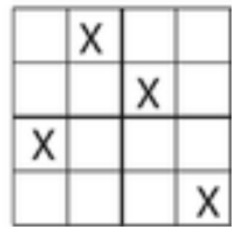
PersonId	Brand	|G|	|J|	|GJ| ...
...	...	...	...	
5137	Adidas	2	100	5
9001	Adidas	50	100	20
9001	Nike	50	200	301
2995	Nike	100	200	405
4145	Puma	100	300	1001
...	...	...	...	...

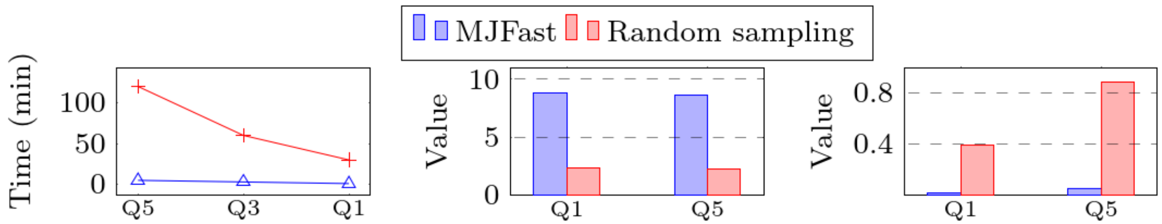
Multi-model Parameter Curation: Given a multi-model query Q with a d-dimensional parameter domain P^d that is a Cartesian product of the base domain, as well as the size k. The objective is to select a subset $S_k \subset P^d$ such that the parameter diversity of S_k is maximal.

Parameter Curation (con’d)

Our approach: Latin Hypercube Sampling (LHS)







(a) Generation time (b) Sum of pair distance (c) KL Divergence

Figure 8. Parameter Curation in efficiency, diversity, and stability.

DB layer implementations

UniBench has implemented all the designed queries in AQL, Orient SQL and SQL/Cypher

An example of Q5

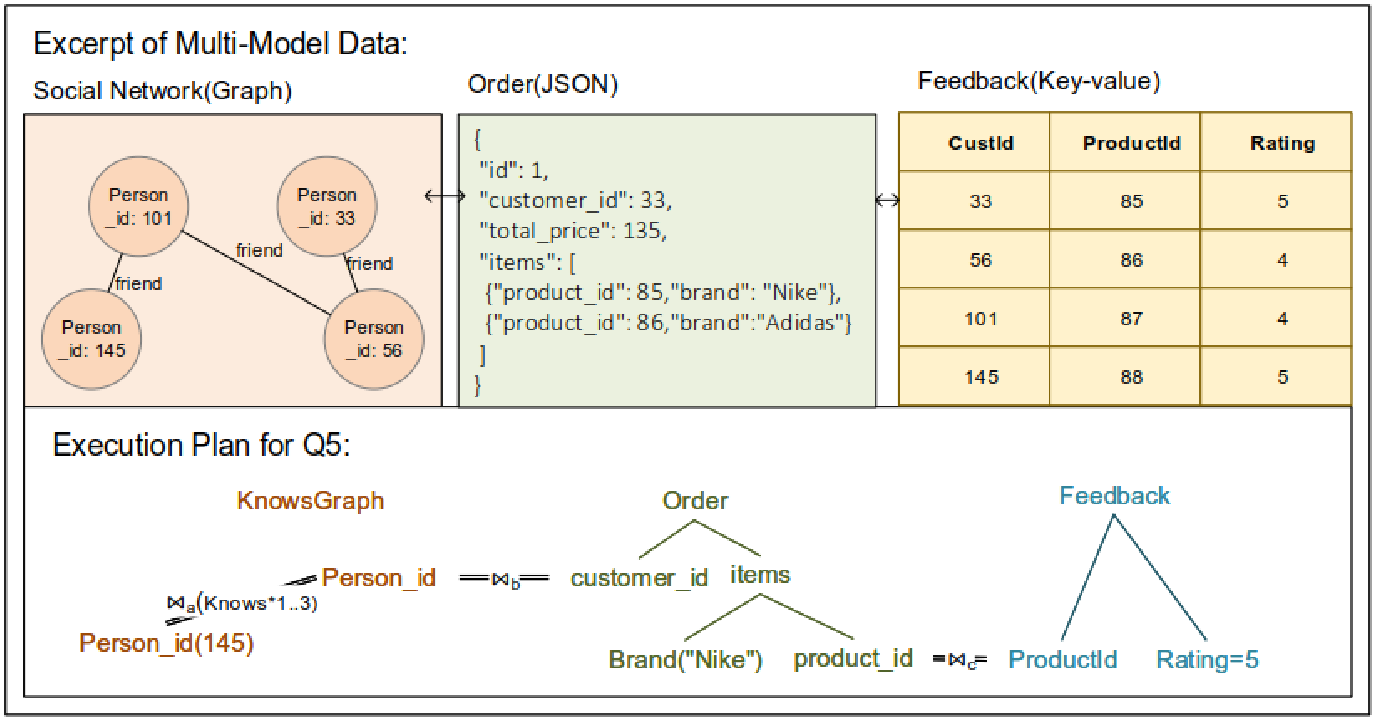


Fig. 3: Example of multi-model join

An example of Q5: ArangoDB

```
FOR friend IN 1..3 OUTBOUND PersonId/56 KnowsGraph
FOR order IN Order
FOR feedback IN Feedback
FILTER order.customer_id==friend._id AND
BrandName/"Nike" IN order.items[*].brand AND
friend._id==feedback.custID AND
feedback.Rating==5
RETURN {person:friend, feedback:feedback}
```

An example of Q5: OrientDB

```
SELECT person, person.feedback
FROM
    (TRAVERSE Expand(Out('Knows') )
     FROM person
     WHERE PersonId=56 and $depth<3)
WHERE "Nike" in Order.items.brand and feedback.Rating==5
UNWIND Order.items
```

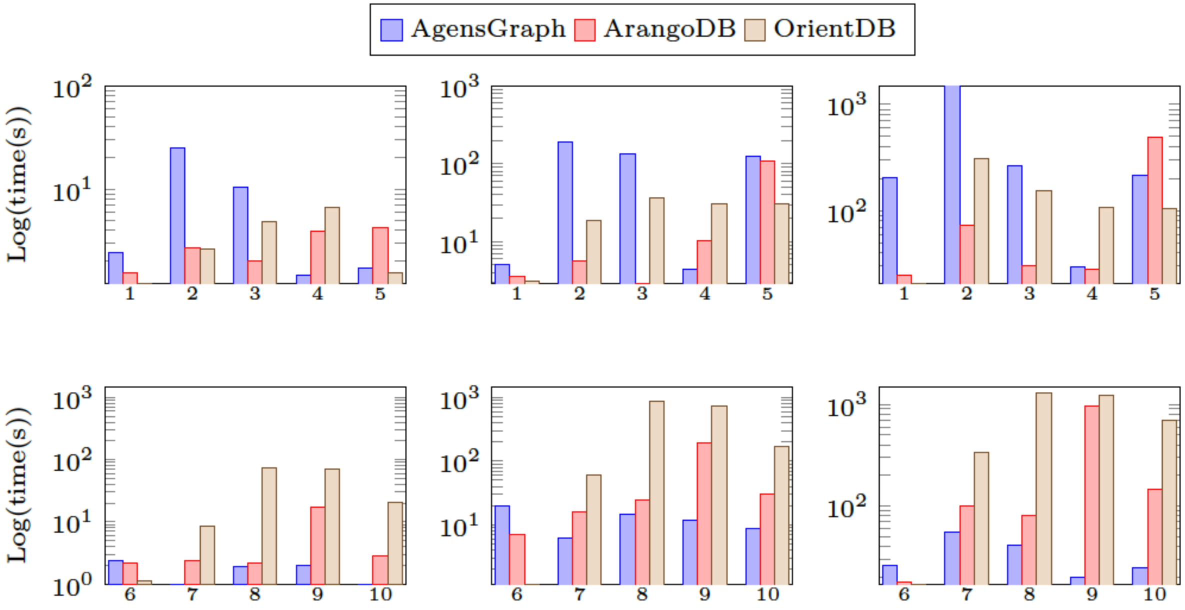
An example of Q5: AgensGraph

```
SELECT person, feedback
FROM orders,
  jsonb_array_elements(orders.data->'items') element
INNER JOIN feedback
ON feedback.asin=element->>'asin'
INNER JOIN
  (MATCH(c:customers {id:'56'})-[:KNOWS*1..3]->(person:persons)
   RETURN person)
ON person->>'id'=feedback.personid;
```

An example of Q5: Spark SQL

```
val persons = sqlContext.read.format("csv")
                   .load("HDFS://person.csv").toDF()
val orders   = sqlContext.read.format("json")
                   .load("HDFS://order.json").toDF()
val knows    = sqlContext.read.format("csv")
                   .load("HDFS://knows.csv").toDF("src", "dst")
val graph    = GraphFrame(persons, knows)
val friends  = graph.find("(a)-[e1]->(b);(b)-[e2]->(c)")
                   .filter("a.id=56")
                   .select(explode(array("a.id", "b.id"))
                   .alias("PersonId")).distinct
val orders=orders.where(array_contains(col("items.brand"),"Nike"))
val result =   orders.join(friends,Seq("PersonId"),"inner")
                   .select("PersonId","items").collect()
```

UniBench Evaluation



(a) SF1 (b) SF10 (c) SF30

- For query processing, OrientDB is excel at graph-based queries
- ArangoDB is the best at document filtering with joining query
- AgensGraph outperforms the others in performing complex aggregation queries

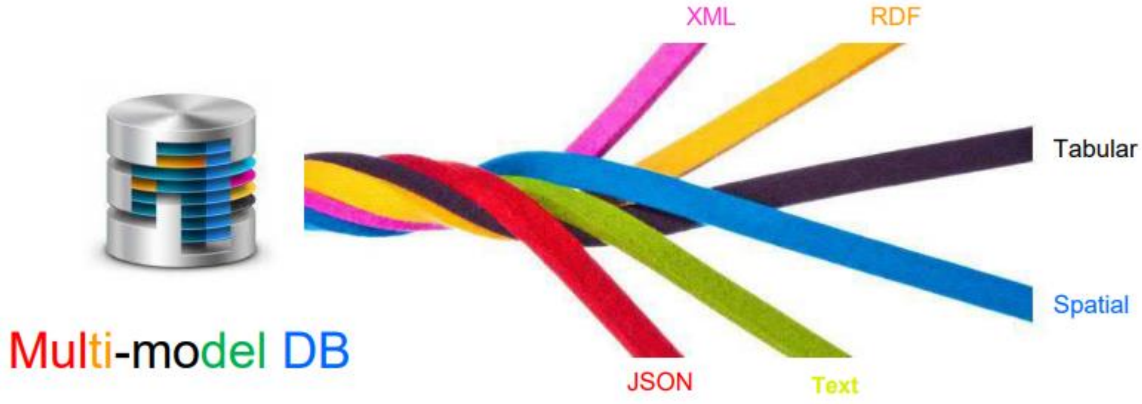
Find out more details from https://link.springer.com/article/10.1007/s10619-019-07279-6

An Demonstration of UniBench

In this part, we demonstrate how to use UniBench to benchmark a multi-model database, ArangoDB

UniBench 2.0

- More data models, e.g., RDF
- More cross-model queries
- Stay tuned on https://github.com/HY-UDBMS/UniBench

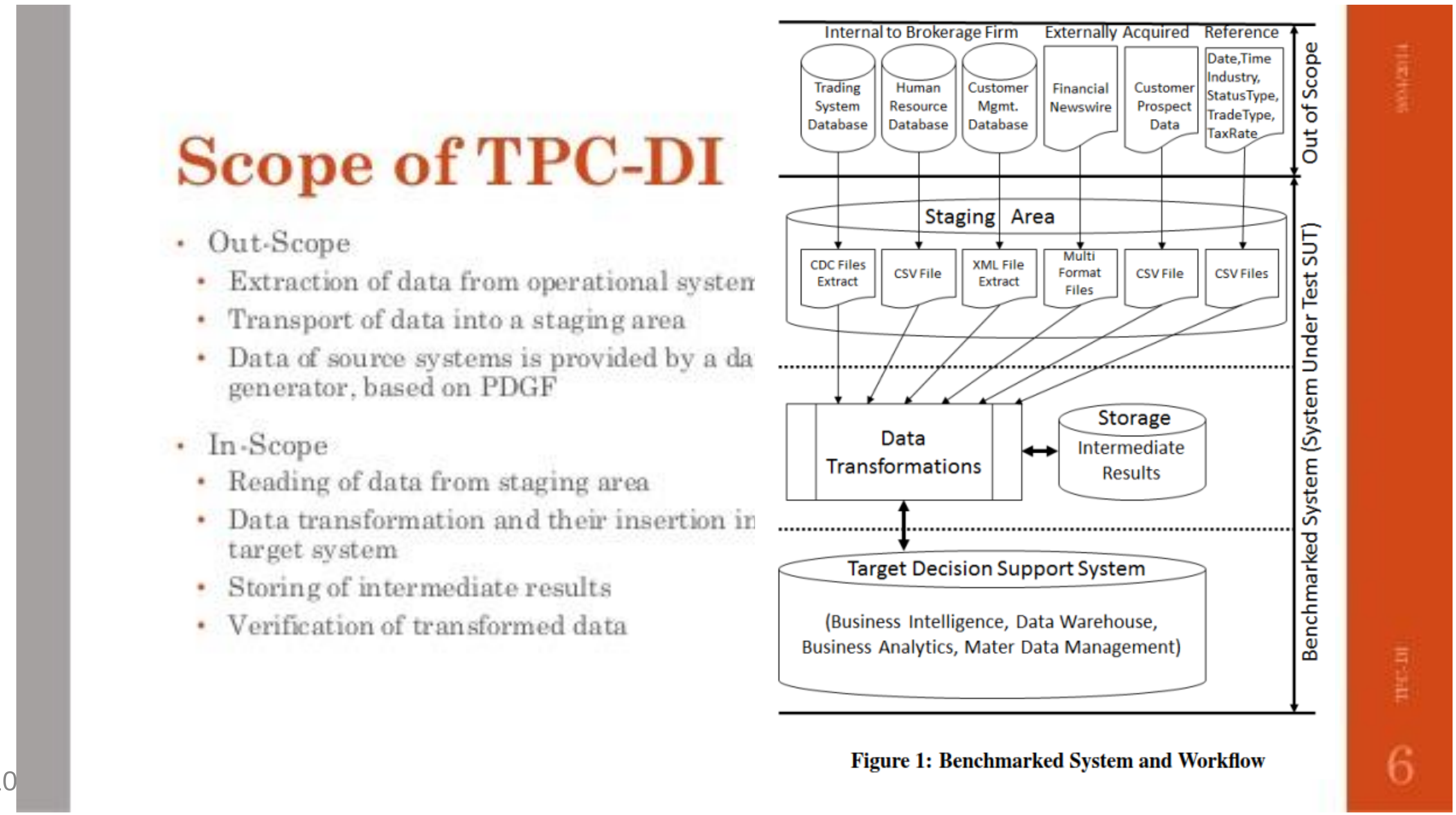


Other related benchmarks

- TPC-DI
- PolyBench
- …

TPC-DI: data integration benchmark

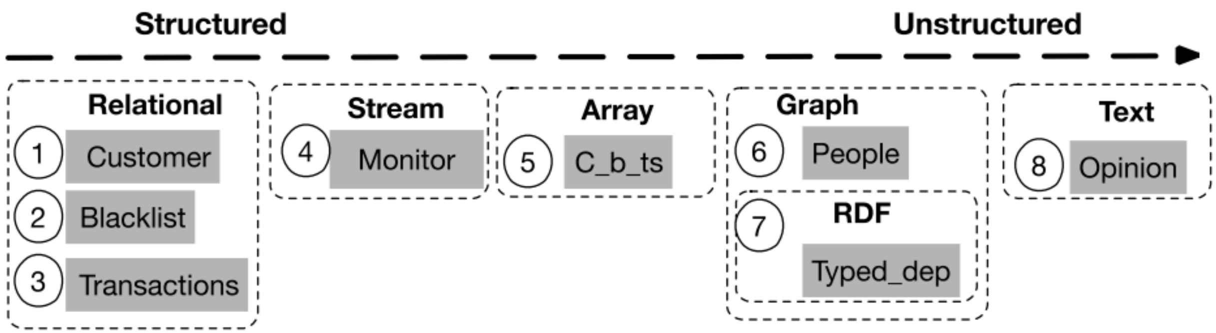
Retail Brokerage Firm



PolyBench: PolyStore benchmark

Banking business

Data model

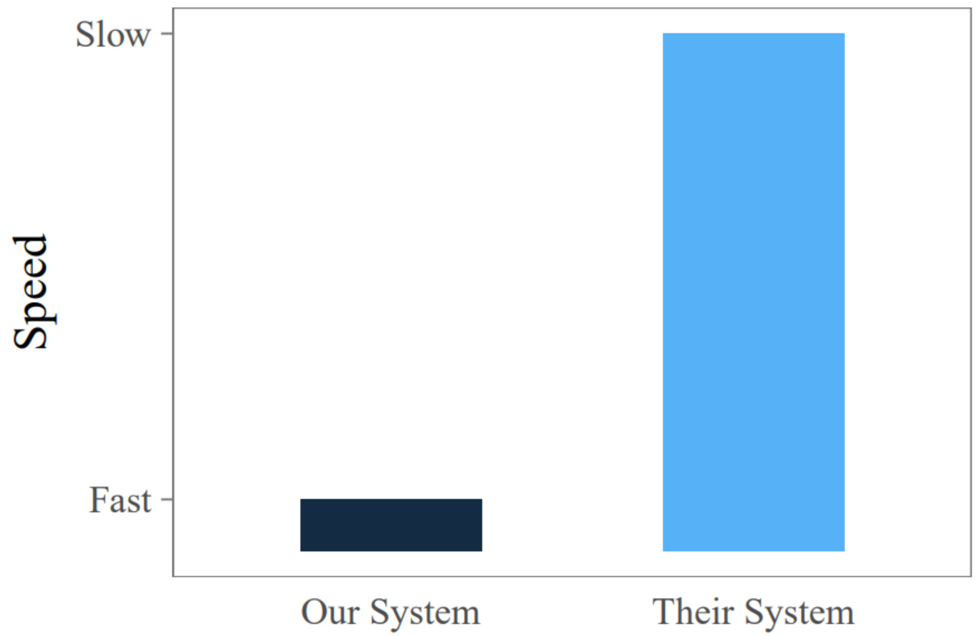


- Simulation of banking bussiness model
- Unstructured, semi-structured, and structured data

Outline

- Introduction to Big Data System Benchmarking(15')
- Benchmarking SQL Big Data Analytical Systems(20')
- Benchmarking Map-Reduce/NoSQL Systems(20')
- Benchmarking Graph-based Big Data Systems(20')
- Benchmarking Multi-Model Big Data Systems(35')
- Open Challenges and Future Directions(10')

Open Challenges



Paper without this plot will not get accepted
Product without this plot will not get traction/sold

Content from https://dbtest.dima.tu-berlin.de/media/DBTEST.io_Presentations/dbtest_raasveldt_18-06.pdf

Open Challenges

Big data benchmarking pitfalls

- Non-reproducibility
- Failure to Optimize
- Apples vs. Oranges
- Incorrect Results
- Cold vs. Hot Runs
- Data Preprocessing/ Job setup
- Overly tuning

Content from https://dbtest.dima.tu-berlin.de/media/DBTEST.io_Presentations/dbtest_raasveldt_18-06.pdf

Future direction No.1

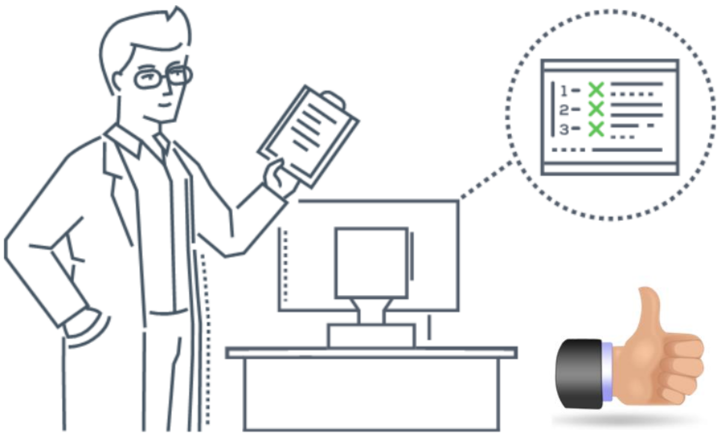
Verifiable/Probablistic big data benchmarking

- Open-Source & Reproducible
- A > B with confidence interval
- Fine-grained Benchmarking
  - Example: SQLScalpel. https://dbtest.dima.tu-berlin.de/media/DBTEST.io_Presentations/dbtest_kersten_18-06.pdf

Future direction No.2

Personalized big data benchmarking

- User-driven requirements and metrics
- Component-based
- Interactive benchmarkling
- Automated reports with insights



Future direction No.3

Benchmarking results reuse:

- Many useful evaluations
- Collect valid insights
- Build the knowledge for future use
- Trace the system evolution, e.g., TPC-DS -> TPC DS V2, Spark 2.0.0 -> Spark 3.0.0

Thank you! Any questions?

References(1/4)

- Jiaheng Lu. "Towards Benchmarking Multi-model databases". In CIDR 2017.
- Chao Zhang, Jiaheng Lu. "Holistic Evaluation in Multi-Model Databases Benchmarking". In Distributed and Parallel Databases, 2019.
- Chao Zhang, Jiaheng Lu, Pengfei Xu, and Yuxing Chen. "UniBench: A benchmark for multi-model database management systems." In *Technology Conference on Performance Evaluation and Benchmarking*, pp. 7-23. Springer, Cham, 2018.
- Chao Zhang, Jiaheng Lu. "Parameter Curation and Data Generation for Benchmarking Multi-model Queries". In VLDB 2018@PhD.
- Markus Dreseler, Martin Boissier, Tilmann Rabl, and Matthias Uflacker. "Quantifying TPC-H choke points and their optimizations." *Proceedings of the VLDB Endowment* 13, no. 8 (2020): 1206-1220.
- Peter Boncz, Thomas Neumann, and Orri Erling. "TPC-H analyzed: Hidden messages and lessons learned from an influential benchmark." In *Technology Conference on Performance Evaluation and Benchmarking*, pp. 61-76. Springer, Cham, 2013.
- Meikel Poess, Tilmann Rabl, and Hans-Arno Jacobsen. "Analysis of TPC-DS: the first standard benchmark for SQL-based big data systems." In *Proceedings of the 2017 Symposium on Cloud Computing*, pp. 573-585. 2017.
- Chen, Yueguo, Xiongpai Qin, Haoqiong Bian, Jun Chen, Zhaoan Dong, Xiaoyong Du, Yanjie Gao, Dehai Liu, Jiaheng Lu, and Huijie Zhang. "A study of SQL-on-Hadoop systems." In *Workshop on big data benchmarks, performance optimization, and emerging hardware*, pp. 154-166. Springer, Cham, 2014.

References(2/4)

- Ahmad Ghazal, Tilmann Rabl, Minqing Hu, Francois Raab, Meikel Poess, Alain Crolotte, and Hans-Arno Jacobsen. "BigBench: towards an industry standard benchmark for big data analytics." In *Proceedings of the 2013 ACM SIGMOD international conference on Management of data*, pp. 1197-1208. 2013.
- Andrew Pavlo, Erik Paulson, Alexander Rasin, Daniel J. Abadi, David J. DeWitt, Samuel Madden, and Michael Stonebraker. "A comparison of approaches to large-scale data analysis." In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, pp. 165-178. 2009.
- Michael Stonebraker, Daniel Abadi, David J. DeWitt, Sam Madden, Erik Paulson, Andrew Pavlo, and Alexander Rasin. "MapReduce and parallel DBMSs: friends or foes?." *Communications of the ACM* 53, no. 1 (2010): 64-71.
- Shengsheng Huang, Jie Huang, Jinquan Dai, Tao Xie, and Bo Huang. "The HiBench benchmark suite: Characterization of the MapReduce-based data analysis." In *2010 IEEE 26th International Conference on Data Engineering Workshops (ICDEW 2010)*, pp. 41-51. IEEE, 2010.
- Shi, Juwei, Yunjie Qiu, Umar Farooq Minhas, Limei Jiao, Chen Wang, Berthold Reinwald, and Fatma Özcan. "Clash of the titans: Mapreduce vs. spark for large scale data analytics." *Proceedings of the VLDB Endowment* 8, no. 13 (2015): 2110-2121.
- Brian F. Cooper, Adam Silberstein, Erwin Tam, Raghu Ramakrishnan, and Russell Sears. "Benchmarking cloud serving systems with YCSB." In *Proceedings of the 1st ACM symposium on Cloud computing*, pp. 143-154. 2010.

References(3/4)

- Sanket Chintapalli, Derek Dagit, Bobby Evans, Reza Farivar, Thomas Graves, Mark Holderbaugh, Zhuo Liu et al. "Benchmarking streaming computation engines: Storm, flink and spark streaming." In *2016 IEEE international parallel and distributed processing symposium workshops (IPDPSW)*, pp. 1789-1792. IEEE, 2016.
- Lei Wang, Jianfeng Zhan, Chunjie Luo, Yuqing Zhu, Qiang Yang, Yongqiang He, Wanling Gao et al. "Bigdatabench: A big data benchmark suite from internet services." In *2014 IEEE 20th international symposium on high performance computer architecture (HPCA)*, pp. 488-499. IEEE, 2014.
- Orri Erling, Alex Averbuch, Josep Larriba-Pey, Hassan Chafi, Andrey Gubichev, Arnau Prat, Minh-Duc Pham, and Peter Boncz. "The LDBC social network benchmark: Interactive workload." In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pp. 619-630. 2015.
- Alexandru Iosup, Tim Hegeman, Wing Lung Ngai, Stijn Heldens, Arnau Prat-Pérez, Thomas Manhardto, Hassan Chafio et al. "LDBC Graphalytics: A benchmark for large-scale graph analysis on parallel and distributed platforms." *Proceedings of the VLDB Endowment* 9, no. 13 (2016): 1317-1328.
- Meikel Poess, Tilmann Rabl, Hans-Arno Jacobsen, and Brian Caufield. "TPC-DI: the first industry benchmark for data integration." *Proceedings of the VLDB Endowment* 7, no. 13 (2014): 1367-1378.

References(4/4)

- Jeyhun Karimov, Tilmann Rabl, and Volker Markl. "PolyBench: The first benchmark for polystores." In *Technology Conference on Performance Evaluation and Benchmarking*, pp. 24-41. Springer, Cham, 2018.
- Todor Ivanov, Tilmann Rabl, Meikel Poess, Anna Queralt, John Poelman, Nicolas Poggi, and Jeffrey Buell. "Big data benchmark compendium." In *Technology Conference on Performance Evaluation and Benchmarking*, pp. 135-155. Springer, Cham, 2015.
- Rui Han, Lizy Kurian John, and Jianfeng Zhan. "Benchmarking big data systems: A review." *IEEE Transactions on Services Computing* 11, no. 3 (2017): 580-597.
- Fuad Bajaber, Sherif Sakr, Omar Batarfi, Abdulrahman Altalhi, and Ahmed Barnawi."Benchmarking big data systems: A survey." *Computer Communications* 149 (2020):241-251
- Todor Ivanov, Timo Eichhorn, Arne Jørgen Berre, Tomás Pariente Lobo, Ivan Martinez Rodriguez, Ricardo Ruiz Saiz, Barbara Pernici, and Chiara Francalanci. "Building the DataBench Workflow and Architecture." In *International Symposium on Benchmarking, Measuring and Optimization*, pp. 165-171. Springer, Cham, 2019.
- Wanling Gao, Fei Tang, Lei Wang, Jianfeng Zhan, Chunxin Lan, Chunjie Luo, Yunyou Huang et al. "AIBench: an industry standard internet service AI benchmark suite." *arXiv preprint arXiv:1908.08998* (2019).
- Mark Raasveldt, Pedro Holanda, Tim Gubner, and Hannes Mühleisen. "Fair benchmarking considered difficult: Common pitfalls in database performance testing." In *Proceedings of the Workshop on Testing Database Systems*, pp. 1-6. 2018.