From BigBench to TPCx-BB: Standardization of a Big Data Benchmark

Paul Cao, Bhaskar Gowda, Seetha Lakshmi, Chinmayi Narasimhadevara, Patrick Nguyen, John Poelman, Meikel Poess, *Tilmann Rabl*

TPCTC – New Delhi, 09/05/2016

Agenda

TPCx-BB

- from research idea
- to full big data benchmark
- to industry standard
- to wider adoption

Overview, changes, experiments, analysis, outlook.

Before BigBench

Micro-Benchmarks

- System level measurement
- Illustrative not informative
- See keynote

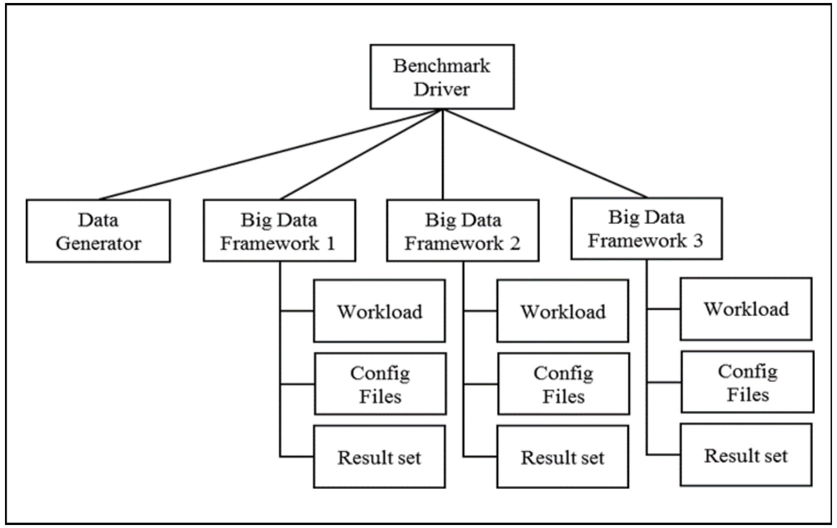
Functional Benchmarks

- Better than micro-benchmarks
- Simplified approach
- E.g., sorting

Benchmark suites

- Collection of micro and functional
- Standardization problems
- E.g., HiBench

The BigBench Proposal



End-to-end, application level benchmark

Focused on Parallel DBMS and MR engines

- Framework agnostic
- SW based reference implementation

History

- Launched at 1st WBDB, San Jose, 2012
- Published at SIGMOD 2013
- Full kit at WBDB 2014
- TPC BigBench Working Group in 2015
- TPCx-BB standardized in Jan 2016
- First published result Mar 2016

Collaboration with Industry & Academia

- First: Teradata, University of Toronto, Oracle, InfoSizing
- Now: Actian, bankmark, CLDS, Cisco, Cloudera, Hortonworks, IBM, Infosizing, Intel, Microsoft, Oracle, Pivotal, SAP, TU Berlin, UoFT, …

Derived from TPC-DS

Multiple snowflake schemas with shared dimensions

24 tables with an average of 18 columns

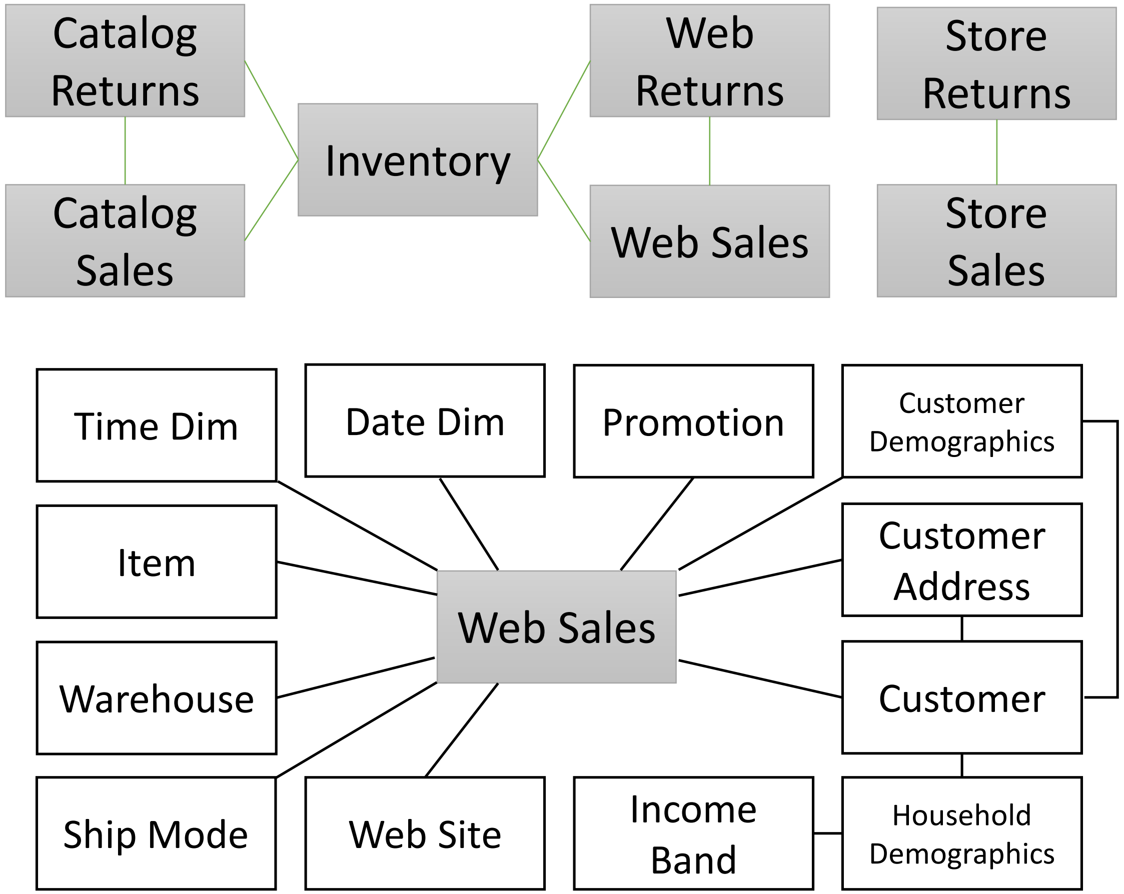
99 distinct SQL '99 queries with random substitutions

Representative skewed database content

Sub-linear scaling of non-fact tables

Ad-hoc, reporting, iterative and extraction queries

Now in Version 2 for SQL on Hadoop

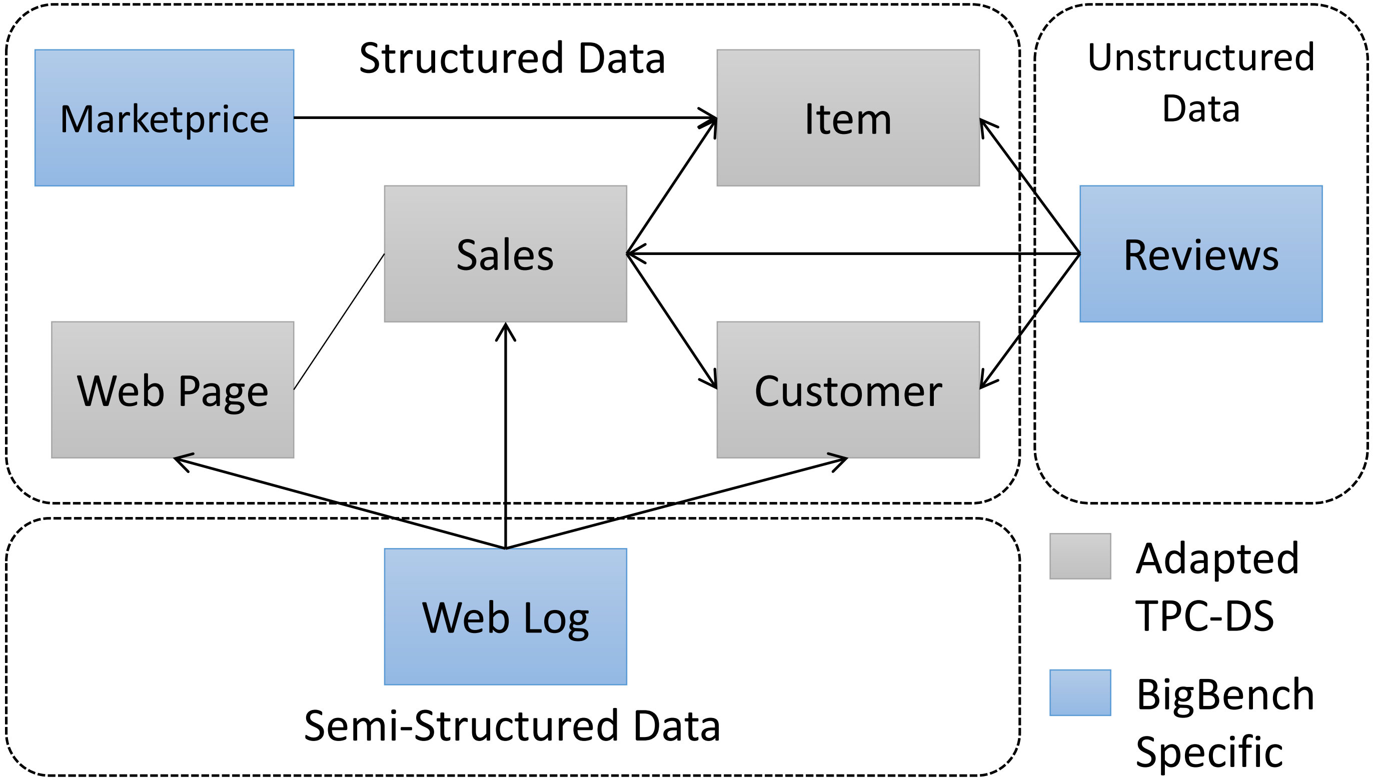


BigBench Data Model

Structured: TPC-DS + market prices

Semi-structured: website click-stream

Unstructured: customers' reviews



Scaling

Continuous scaling model

- Only SF 1, 3, 10, 30, … allowed

SF 1 ~ 1 GB

Different scaling speeds

- Adapted from TPC-DS
  - Static
  - Square root
  - Logarithmic
  - Linear (LF)

$$LF = SF + (SF - (\log_5(SF) * \sqrt{SF})) = 2SF - \log_5(SF) * \sqrt{SF}$$

Table Name	# Rows SF 1	Bytes/Row	Scaling
date	109573	141	static
time	86400	75	static
ship_mode	20	60	static
household_demographics	7200	22	static
customer_demographics	1920800	40	static
customer	100000	138	square root
customer_address	50000	107	square root
store	12	261	square root
warehouse	5	107	logarithmic
promotion	300	132	logarithmic
web_page	60	134	logarithmic
item	18000	308	square root
item_marketprice	90000	43	square root
inventory	23490000	19	square root * logarithmic
store_sales	810000	143	linear
store_returns	40500	125	linear
web_sales	810000	207	linear
web_returns	40500	154	linear
web_clickstreams	6930000	27	linear
product_reviews	98100	670	linear

Workload

Business functions (adapted from McKinsey report)

- **Marketing**
  - Cross-selling, customer micro-segmentation, sentiment analysis, enhancing multichannel consumer experiences
- **Merchandising**
  - Assortment optimization, pricing optimization
- **Operations**
  - Performance transparency, product return analysis
- **Supply chain**
  - Inventory management
- **Reporting (customers and products)**

30 queries covering all functions

Query 1

Find products that are sold together frequently in given stores. Only products in certain categories sold in specific stores are considered and "sold together frequently" means at least 50 customers bought these products together in a transaction.

HiveQL Query 1

```
SELECT pid1, pid2, COUNT (*) AS cnt
FROM (
        FROM (
                SELECT s.ss_ticket_number AS oid , s.ss_item_sk AS pid
                FROM store_sales s
                INNER JOIN item i ON s.ss_item_sk = i.i_item_sk
                WHERE i.i_category_id in (1 ,2 ,3) and s.ss_store_sk in (10 , 20, 33, 40, 50)
                CLUSTER BY oid
        ) q01_map_output
        REDUCE q01_map_output.oid, q01_map_output.pid
        USING 'java -cp bigbenchqueriesmr.jar:hive-contrib.jar de.bankmark.bigbench.queries.q01.Red'
        AS (pid1 BIGINT, pid2 BIGINT)
) q01_temp_basket
GROUP BY pid1, pid2
HAVING COUNT (pid1) >= 50
ORDER BY pid1, cnt, pid2;
```

Towards an Industry standard

- Collaboration with TPC
- Enterprise vs Express benchmark

Enterprise	Express
Specification	Kit
Specific implementation	Kit evaluation
Best optimization	System tuning (not kit)
Complete audit	Self audit / peer review
Price requirement	No pricing
Full ACID testing	ACID self-assessment (no durability)
Large variety of configuration	Focused on key components
Substantial implementation cost	Limited cost, fast implementation

- Consensus based development in subcommittee

- Specification sections
  - Preamble
    - High level overview
  - Database design
    - Overview of the schema and data
  - Workload scaling
    - How to scale the data and workload
  - Metric and execution rules
    - Reported metrics and rules on how to run the benchmark
  - Pricing
    - Reported price information
  - Full disclosure report
    - Wording and format of the benchmark result report
  - Audit requirements
    - Minimum audit requirements for an official result, self auditing scripts and tools

TPC Transaction Processing Performance Council

Benchmark Process – BigBench

Adapted to batch systems

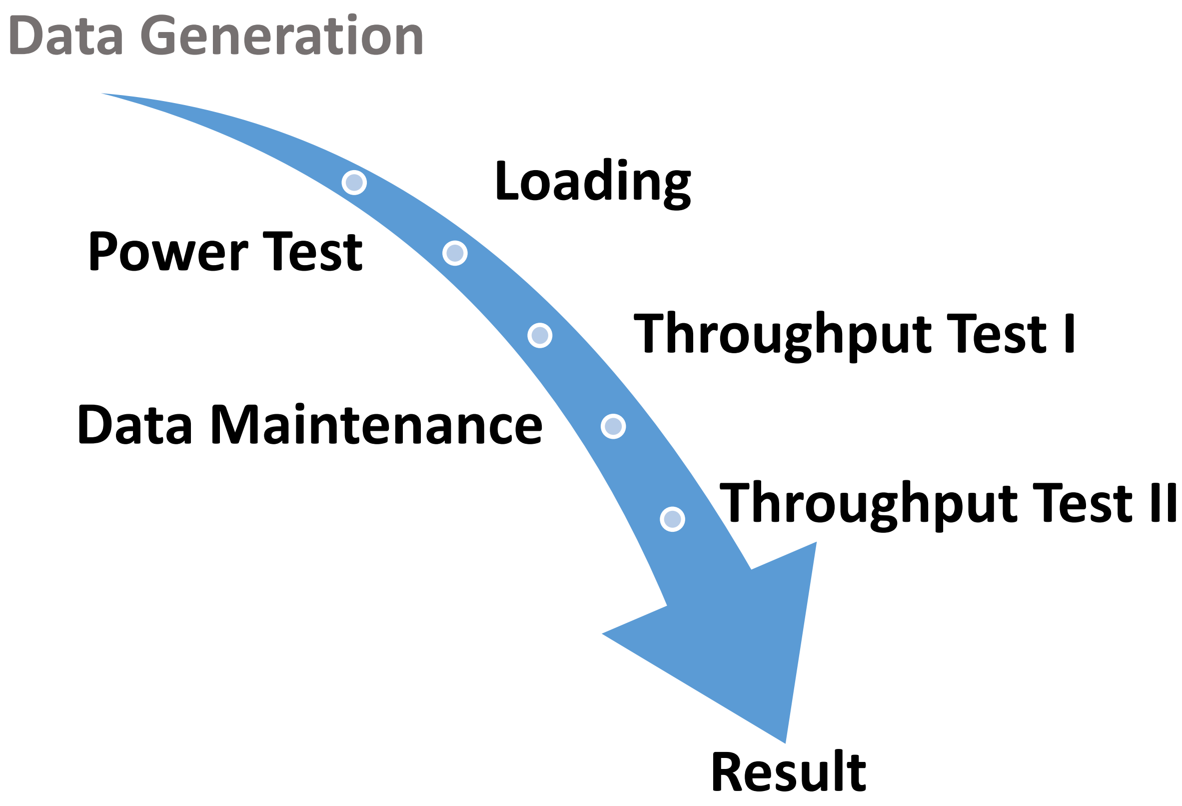
No trickle update

Measured processes

- Loading
- Power Test (single user run)
- Throughput Test I (multi user run)
- Data Maintenance
- Throughput Test II (multi user run)

Result

- Additive metric



Benchmark Process – TPCx-BB

No update

Measured processes

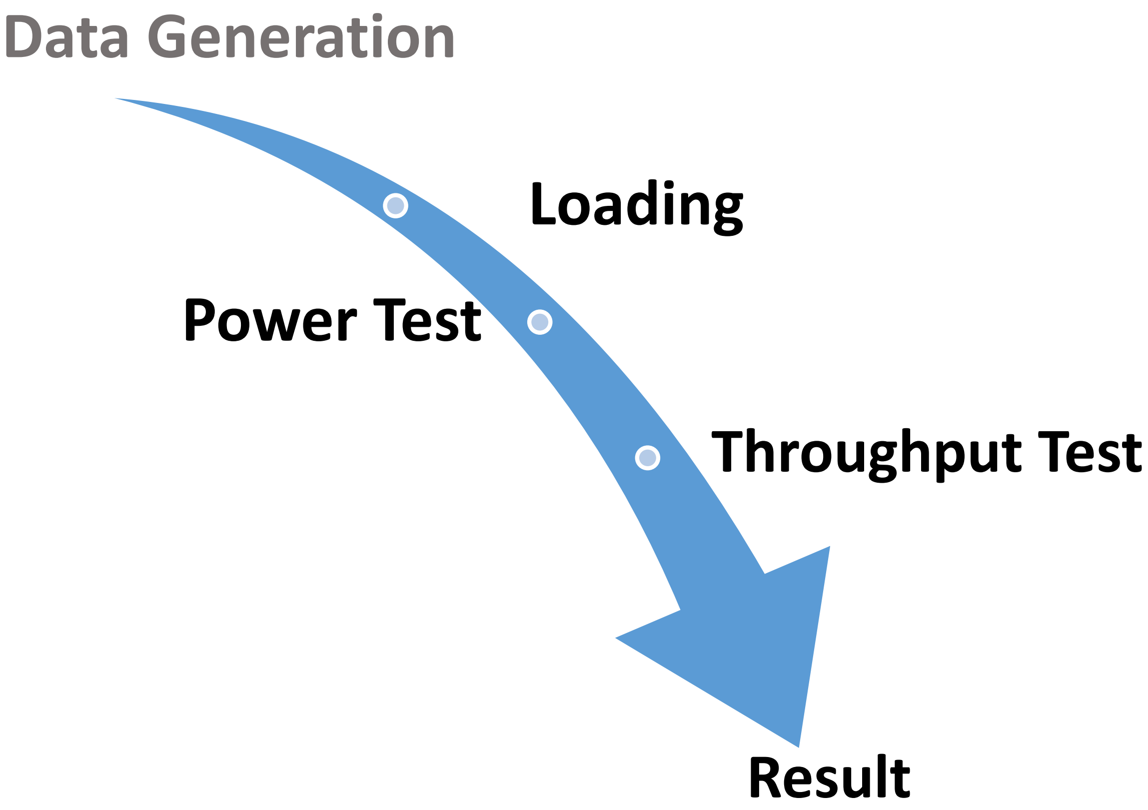
- Loading
- Power Test (single user run)
- Throughput Test (multi user run)

Result

- Mixed metric

Two runs

- Lower number reported



Workload – Technical Aspects – BigBench

Generic Characteristics

Data Sources	#Queries	Percentage
Structured	18	60%
Semi-structured	7	23%
Un-structured	5	17%

Hive Implementation Characteristics

Query Types	#Queries	Percentage
Pure HiveQL	14	46%
Mahout	5	17%
OpenNLP	5	17%
Custom MR	6	20%

Query	Input Datatype	Processing Model	Query	Input Datatype	Processing Model
#1	Structured	Java MR	#16	Structured	Java MR (OpenNLP)
#2	Semi-Structured	Java MR	#17	Structured	HiveQL
#3	Semi-Structured	Python Streaming MR	#18	Unstructured	Java MR (OpenNLP)
#4	Semi-Structured	Python Streaming MR	#19	Structured	Java MR (OpenNLP)
#5	Semi-Structured	HiveQL	#20	Structured	Java MR (Mahout)
#6	Structured	HiveQL	#21	Structured	HiveQL
#7	Structured	HiveQL	#22	Structured	HiveQL
#8	Semi-Structured	HiveQL	#23	Structured	HiveQL
#9	Structured	HiveQL	#24	Structured	HiveQL
#10	Unstructured	Java MR (OpenNLP)	#25	Structured	Java MR (Mahout)
#11	Unstructured	HiveQL	#26	Structured	Java MR (Mahout)
#12	Semi-Structured	HiveQL	#27	Unstructured	Java MR (OpenNLP)
#13	Structured	HiveQL	#28	Unstructured	Java MR (Mahout)
#14	Structured	HiveQL	#29	Structured	Python Streaming MR
#15	Structured	Java MR (Mahout)	#30	Semi-Structured	Python Streaming MR

TPCx-BB Workload

Updated software stack

- MapReduce -> Spark
- Mahout -> MLlib
- Soon: HiveQL -> SparkSQL
- All queries *deterministic*

Alternative

- SQL + UDF
- Flink + SystemML
- ML queries: equal or better result
- …

Aditive Metric – BigBench

Throughput metric

- BigBench queries per hour

Number of queries run

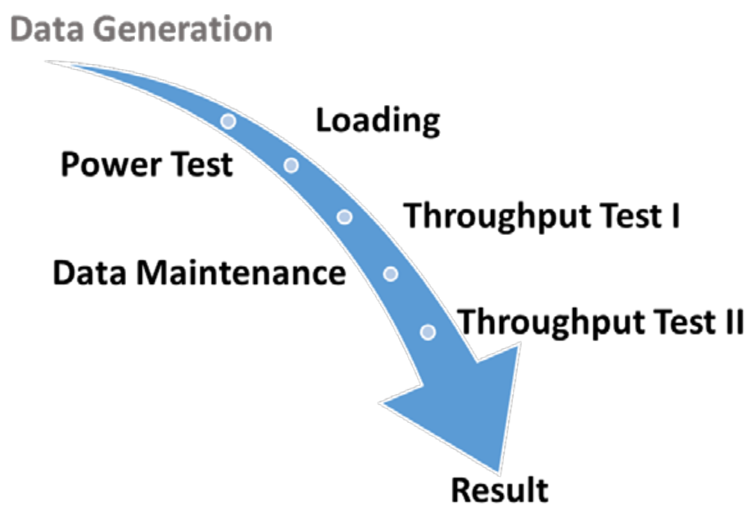
- 30*(2*S+1)

Measured timesMeasured times

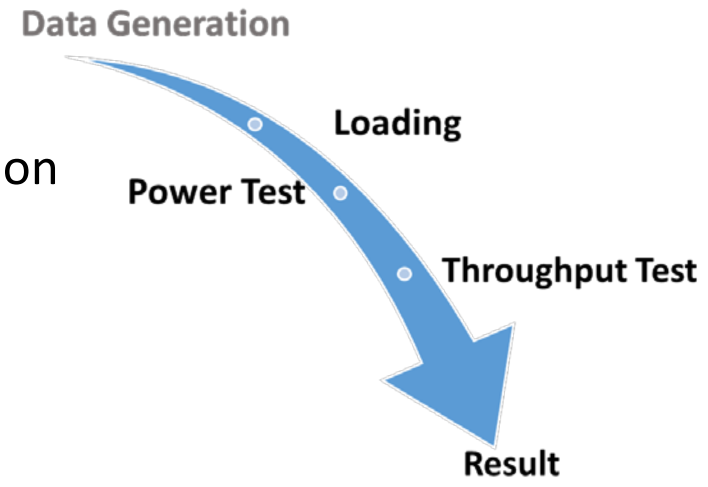
- T_L = elapse time of load test
- T_P = elapse time of power test
- T_{TT1} = elapse time of first throughput test
- T_{DM} = elapse time of data maintenance
- T_{TT1} = elapse time of first throughput test

Metric

- $\mathrm{BBQpH} = \dfrac{30 * 3 * S * 3600}{\mathrm{S} * \mathrm{T_L} + \mathrm{S} * \mathrm{T_P} + \mathrm{T_{TT1}} + \mathrm{S} * \mathrm{T_{DM}} + \mathrm{T_{TT2}}}$



Mixed Metric – TPCx-BB



Throughput metric

- BigBench queries per minute @ SF
- Mix of arithmetic and geometric mean
- Better for skewed workloads and individual query optimization

Number of queries run

- 30*(S+1)

Measured times

- T_{LD} = load time * 0.1
- T_{PT} = geometric mean of query elapse times
- T_{TT} = throughput test time divided by number of streams

Metric

- $\mathrm{BBQpm@SF} = \frac{\mathrm{SF} * 60 * \mathrm{M}}{\mathrm{T_{LD}} + \sqrt[2]{\mathrm{T_{PT}} * \mathrm{T_{TT}}}}$

Plus pricing and energy metric

Overview Experiments

Test	Nodes in Cluster	Framework	Scale Factor
1	9	Hive on MapReduce	3000
2	8	Hive on Spark	1000
3	8	Hive on Tez	3000
4	8	SparkSQL	3000
5	1	Metanautix	1
6	8	Apache Flink	300
7	60	Hive on MapReduce	100000

Overview Experiments cont‘d

Test	#Nodes	Framework	SF	Size	Load	Power	TP
1	9	Hive on MapReduce	3000	3TB	2803s	34076s	54705s
2	8	Hive on Spark	1000	1TB	9389s	13775s	13864s
3	8	Hive on Tez	3000	3TB	3719s		
4	8	SparkSQL	3000	3TB	7896s	24228s	40352s
5	1	Metanautix	1	1GB			
6	8	Apache Flink	300	300GB			
7	60	Hive on MapReduce	100000	100TB	19941s	401738s	

Detailed Experiments – HPE DL360 G8

Hive on MapReduce

- TPCx-BB on Scale Factor 3000 ~ 3 TB

Node	Role	Hardware	Software
1	Master Server	24C,192GB RAM, 8.5TB storage, 10Gbe	RHEL 6.7, CDH 5.6
2-8	Worker Node	24C,256GB RAM, 8.5TB storage, 10Gbe	RHEL 6.7, CDH 5.6

- Run times

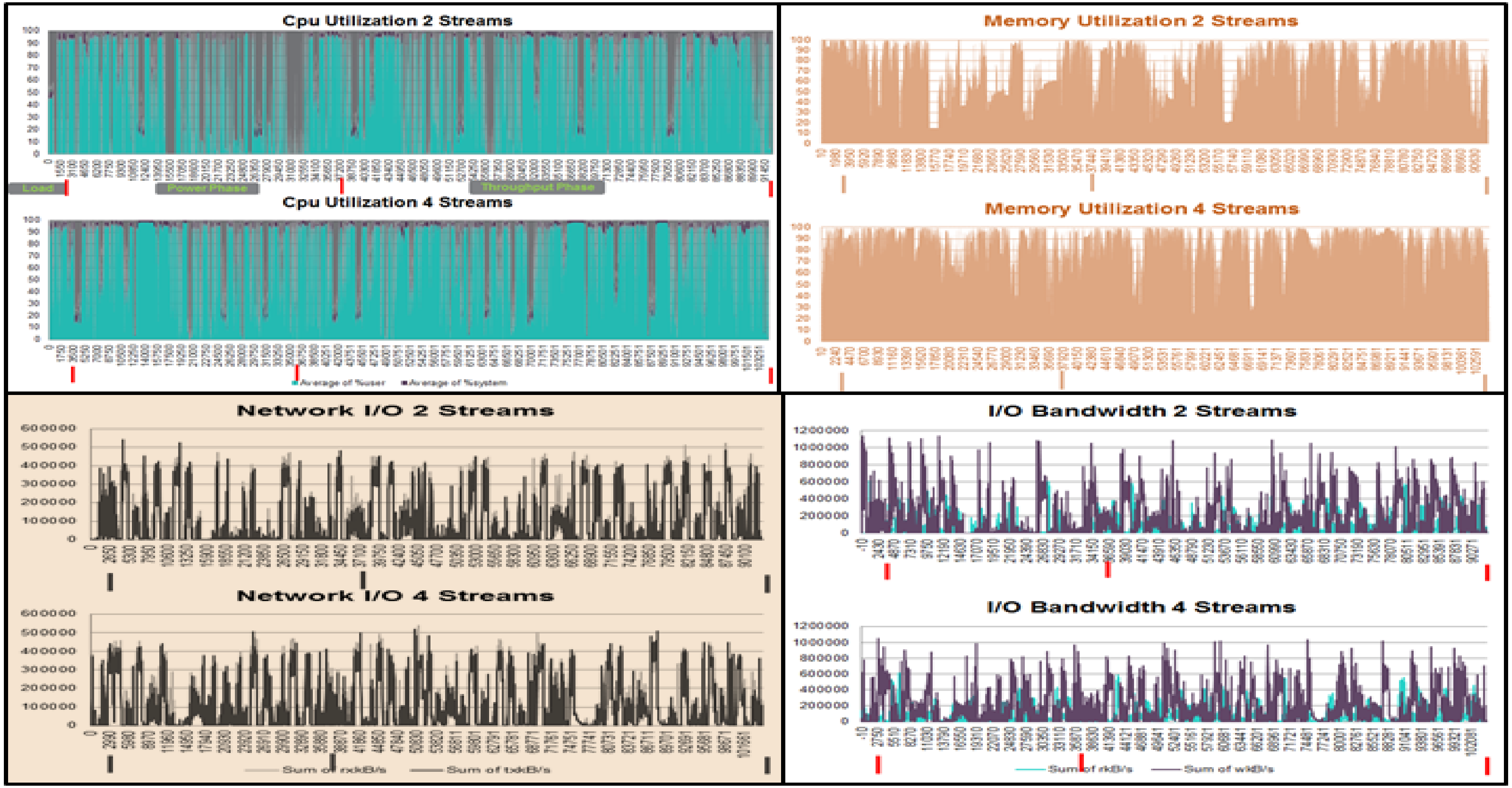
Phase	2 Streams	4 Streams
Load	2803	2796
Power	34076	34179
Throughput	54705	104565

- Result

  BBQpm@SF=162 (2 streams)

  BBQpm@SF=165 (4 streams)

HPE Experiments – Utilization



Discussion

- TPCx-BB can be run on various platforms
  - Full implementation available: Hive on MR/Tez/Spark, SparkSQL
  - Partial implementations: Metanautix, Flink, …
- HPE experiments CPU bound
  - 2 streams 70%
  - 4 streams 90%
- No significant throughput improvement with more streams
- Large scale factors are challenging due to significant skew in data

Outlook

- TPCx-BB: first industry standard end-to-end big data benchmark
  - Batch analytics – challenging for current systems
  - Widely applicable

Emerging use cases beyond TPCx-BB
  - Machine learning / deep learning
  - Graph processing
  - Stream processing

BigBench / TPCx-BB available at:
  - https://github.com/intel-hadoop/Big-Data-Benchmark-for-Big-Bench

Thank You

Questions?

Contact: Tilmann Rabl – rabl@tu-berlin.de