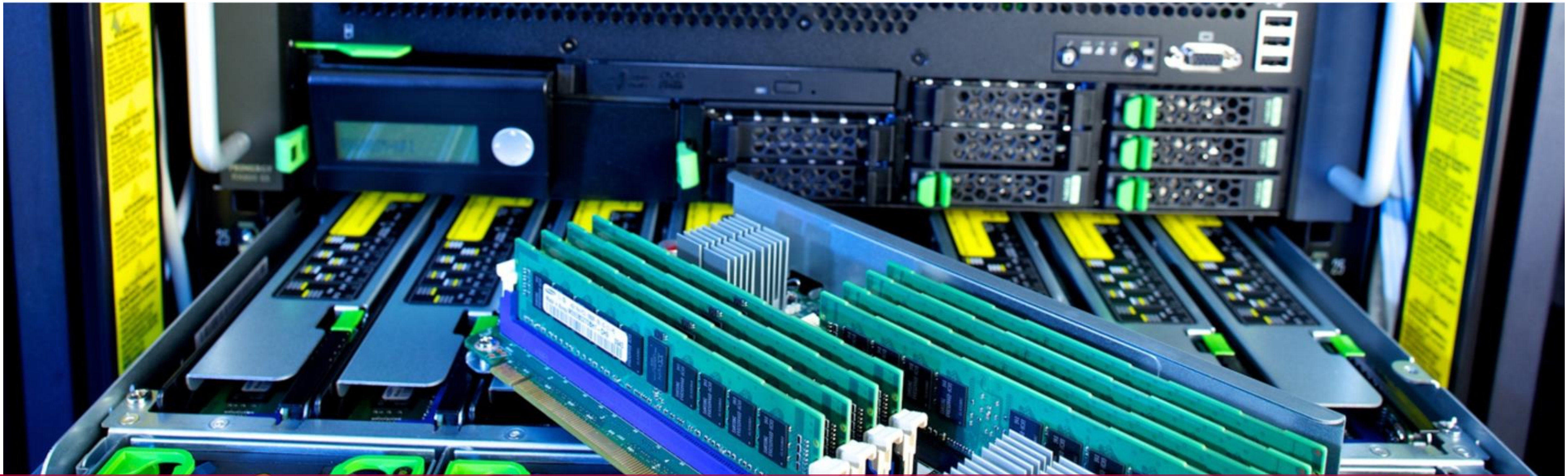
DES DATA ENGINEERING SYSTEMS GROUP

HPI Hasso Plattner Institut
Digital Engineering · Universität Potsdam



Benchmarking & Measurement

Guest Lecture ITU Copenhagen – March 18, 2025

Prof. Tilmann Rabl
Data Engineering Systems
Hasso Plattner Institute

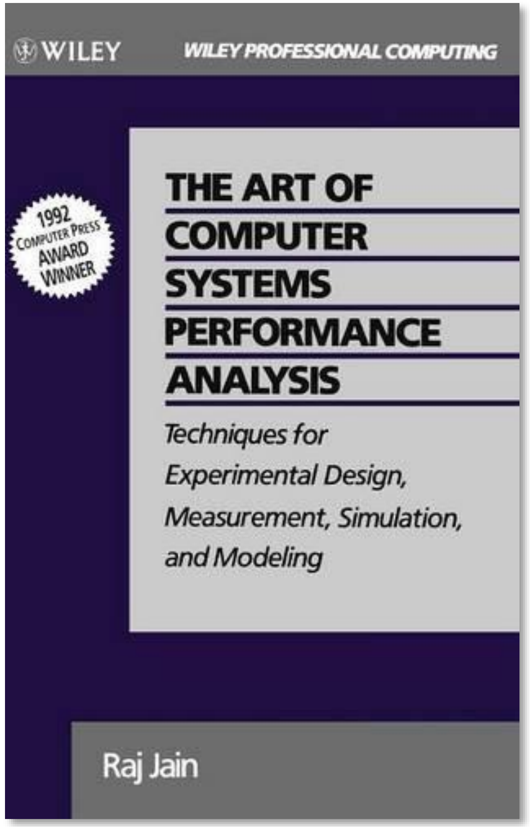
Hasso Plattner Institute @ Uni Potsdam

Research areas :

- → **Systems**
- → **Digital Health**
- → **Data and AI**
- → **Cybersecurity**
- → **Foundations**

This Lecture

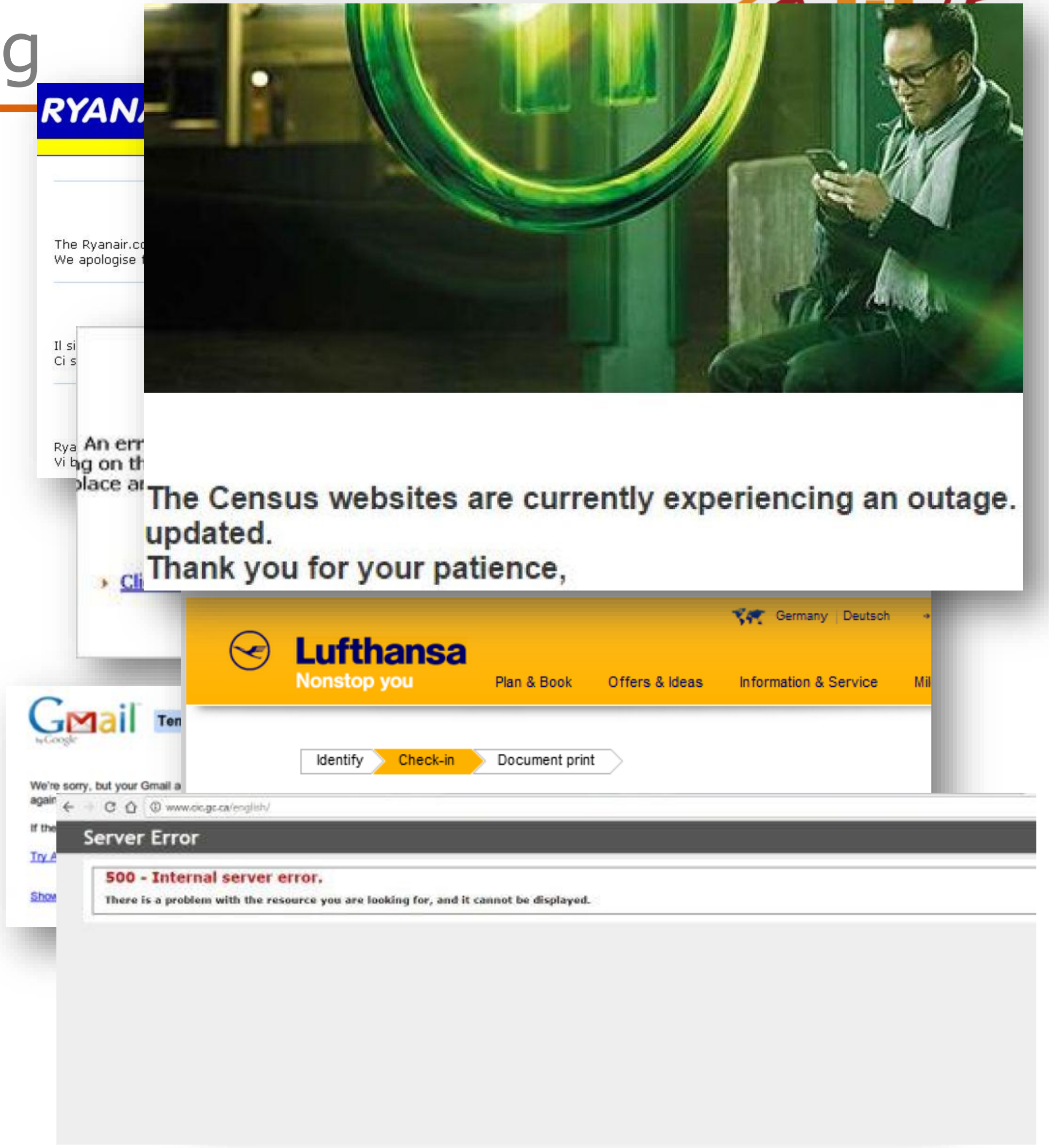
1. Introduction to performance analysis
2. Back of the envelope calculations
3. Measurement
4. Benchmarks
5. BigBench
6. Fair benchmarking



Source: Raj Jain (Washington University in St. Louis) - Computer Systems Analysis https://www.cse.wustl.edu/~jain/cse567-17/index.html

Why Measurement and Benchmarking

- Systems are increasingly complex
- Single transactions can span 1000 components / nodes
- Consumer page load time expectation decreases
  - 1999 – 8 sec
  - 2009 – 2 sec
  - 2018 – 3 sec -> 50% consumers leave page
- Low performance or outages cost $$



Why Measurement and Benchmarking

Prof. Rabl‘s Famous 7 Step Paper/Thesis Recipe

1. Literature search
2. Identify a research problem
3. Describe a novel solution
4. Perform **BotEC** to show it might work
5. Perform **experiments** that show it does work
6. Write up the paper
7. Endure revision cycles

Benchmark vs Analysis

- Analysis
  - Single system/algorithm
  - Individual optimizations
  - Micro benchmarks

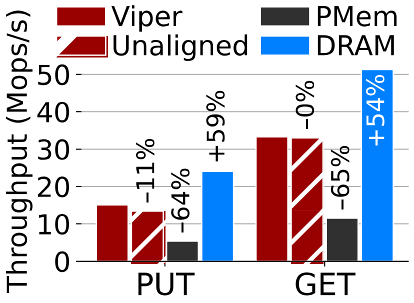


Figure 11: Viper versions.

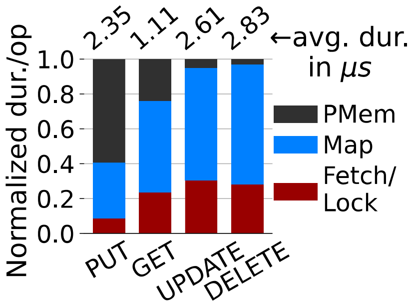


Figure 12: Op breakdown.

- Benchmark
  - Compare multiple systems
  - Standard or real workload

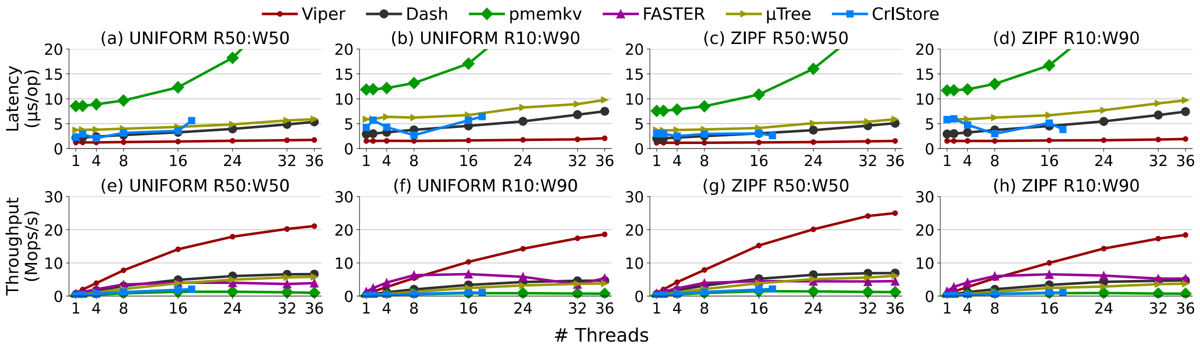


Figure 13: YCSB latency and throughput.

- ***A good paper has both!***

Understanding System Performance

- Modeling
  - Back of the envelope calculation
  - Analytical model
- Measurement
  - Experimental design
  - Benchmarks
- Simulation
  - Emulation
  - Trace-driven
  - …

- Rule of Validation:
  - Do not trust result of a single technique but validate with another
  - Often: validate measurements with model

Back of the Envelope Calculation

How to get good (enough) performance?

- Understand your application
  - Back of the envelope calculation
  - Estimate your system performance within an order of magnitude

- A.k.a., B.S.-filter – filter out stupid ideas early

- Want to know for sure? – Benchmark!

Von Scan:de:Benutzer:Superbass - eigener Scan, Gemeinfrei,
https://commons.wikimedia.org/w/index.php?curid=5205050

BotEC from Jeff Dean
Google system guru

Useful Latency Numbers

L1 cache reference	0.5 ns		
Branch mispredict	5 ns		
L3 cache reference	20 ns		
Mutex lock/unlock	25 ns		
Main memory reference	100 ns		
Compress 1K bytes with Snappy	3,000 ns	3 us	
Send 2K bytes over 10Ge	2,000 ns	2 us	
Read 1 MB sequentially from memory	100,000 ns	100 us	
Read 4KB from NVMe Flash	50,000 ns	50 us	
Round trip within same datacenter	500,000 ns	500 us	
Disk seek	10,000,000 ns	10,000 us	10 ms
Read 1 MB sequentially from disk	20,000,000 ns	20,000 us	20 ms
Send packet CA → Europe → CA	150,000,000 ns	150,000 us	150 ms

DDR4 channel bandwidth	20 GB/sec
PCIe gen3 x16 channel	12.5 GB/sec
NVMe Flash bandwidth	2GB/sec
GbE link bandwidth	10 – 100 Gbps
Disk bandwidth	6 Gbps
NVMe Flash 4KB IOPS	500K – 1M
Disk 4K IOPS	100 – 200

(Jeff Dean)

Basic Considerations

- Do I have a big data problem?
  - My data fits in memory -> probably no

- Example: Find max/avg/min element in list with N integers
  - N=16,000 fits in L1
  - N=64,000 fits in L2
  - N=13,000,000 ~ 52MB ~ 200 ms = instantaneous on 8yr old laptop*

*T.A. Limoncelli. 10 Optimizations on Linear Search. CACM 59(09), 2016

Simple BotEC Example*

- How long to generate image results page (30 thumbnails)?

- Design 1: Read serially, thumbnail 256KB images on the fly
  - `30 seeks * 10 ms/seek + 30 * 256KB / 30 MB/s = 560 ms`

- Design 2: Issue reads in parallel
  - `10 ms/seek + 256KB read / 30 MB/s = 18 ms`
  - (Ignores ***variance***, so really more like 30-60 ms, probably)

- Lots of variations:
  - caching (single images? whole sets of thumbnails?)
  - pre-computing thumbnails

- Back of the envelope helps identify most promising.
- Often, you need a **simple** prototype to get useful numbers.

Sorting Example

- How long does it take to sort 1 GB of 4 Byte numbers?
  - 1GB = 2^{28} int numbers (on most 32 or 64 bit machines)
  - Quicksort et al.: O(n log n)
    - $\log(2^{28})$ passes over 2^{28} numbers ≈ 2^{33} comparisons
    - ½ mispredicts -> 2^{32} mispredicts * 5ns/mispredict = 21 secs
  - Memory bandwidth: mostly sequential streaming
    - 2^{30} Bytes * 28 passes = 28 Gbyte. Memory BW ≈ 4GB/sec -> 7 sec
  - Roughly 30 seconds to sort 1GB on one CPU.
- Let's try

Complete Sorting Program

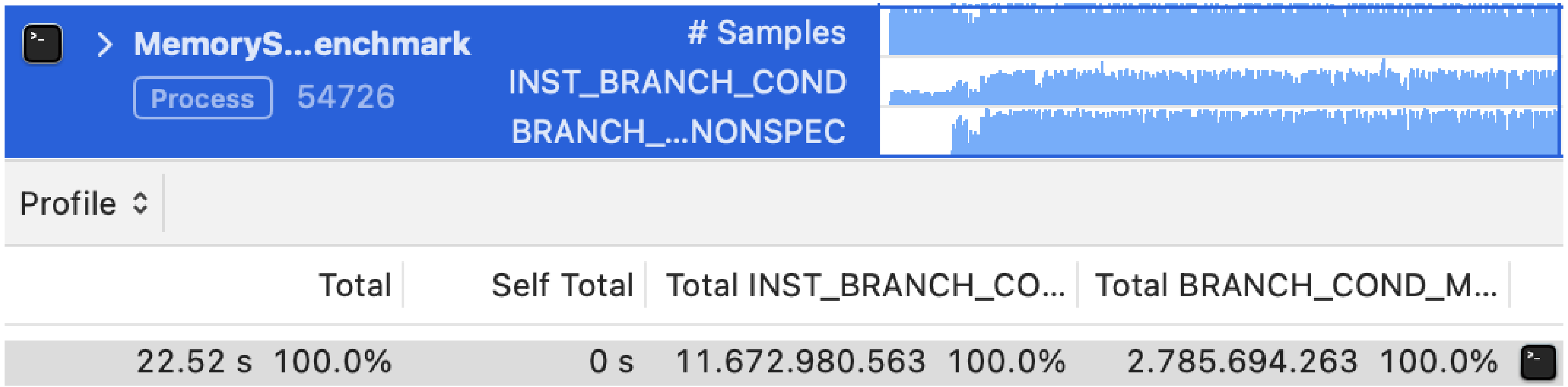
```
#include <iostream>
#include <algorithm>
#include <vector>
int main(int argc, const char * argv[]) {

   int size = 268435456;
   std::vector<int> data;
   data.reserve(size);
   uint64_t dur = 0;
   // Fill array with random numbers
   for (int i = 0; i < size; i++) {
      data.emplace_back(rand()); // % 10000;
   }
   // Measure and sort
   const auto start = std::chrono::steady_clock::now();
   std::sort(data.begin(), data.end());
   const auto end = std::chrono::steady_clock::now();
   // Output result
   dur += std::chrono::duration_cast<std::chrono::microseconds>(end - start).count();
   std::cout << "Total duration: " << dur/1000 << "ms\n";
   return 0;
}
```

Results

Total duration: 20290ms

Profiler output:

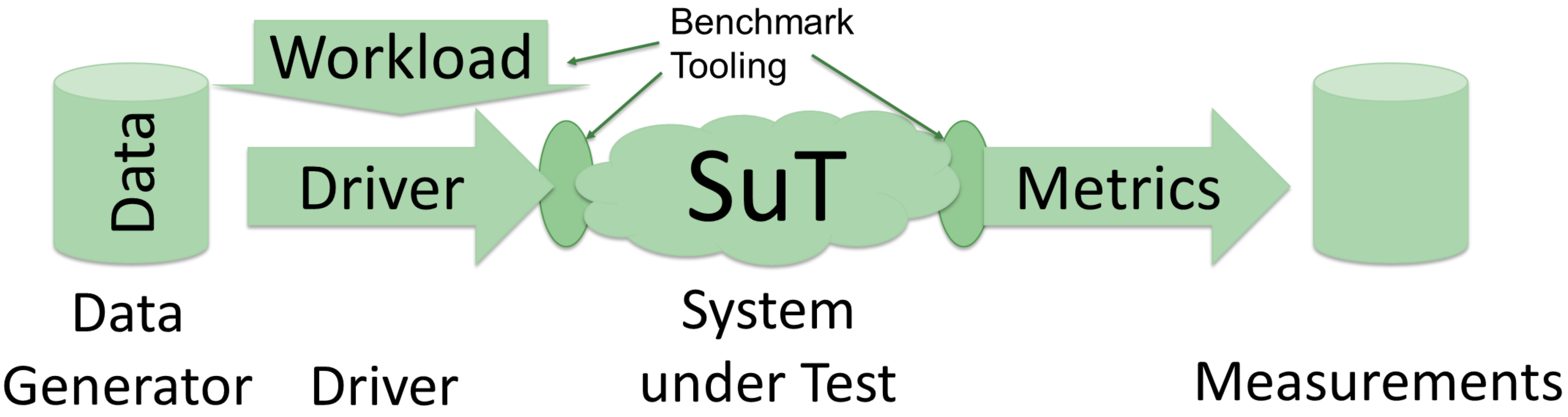
- Comparisons: 11.673.267.976 ≈ $2^{33,4}$
- Branch mispredictions: 2.785.751.957 ≈ $2^{31,4}$



- Not bad…

Measurement & Metrics

Basic Terminology



- System under Test
  - Deployment comprised of hardware, software, data
- Workload
  - Requests by users
- Metrics
  - Criteria used for evaluation

Questions to be Answered Beforehand

- Which scenario do I want to evaluate?
  - Which data / data-sets should be used?
  - Which workload / queries should be run?

  → *Can an existing benchmark supply those?*

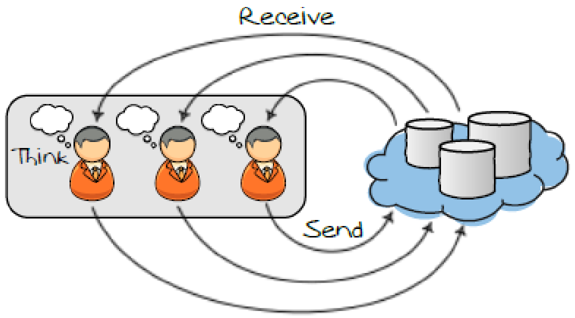
- Which hardware & software should be used?

- Metrics:
  - What to measure?
  - How to measure?
  - How to compare?

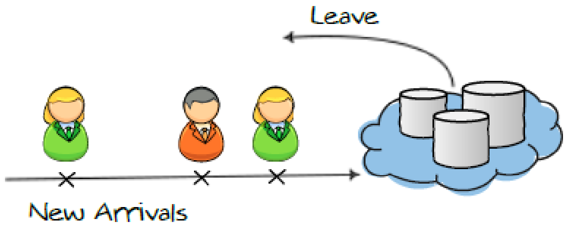
- Crime Scene Investigation: How to find out what is going on?

Things to Consider when Evaluating Fast Systems

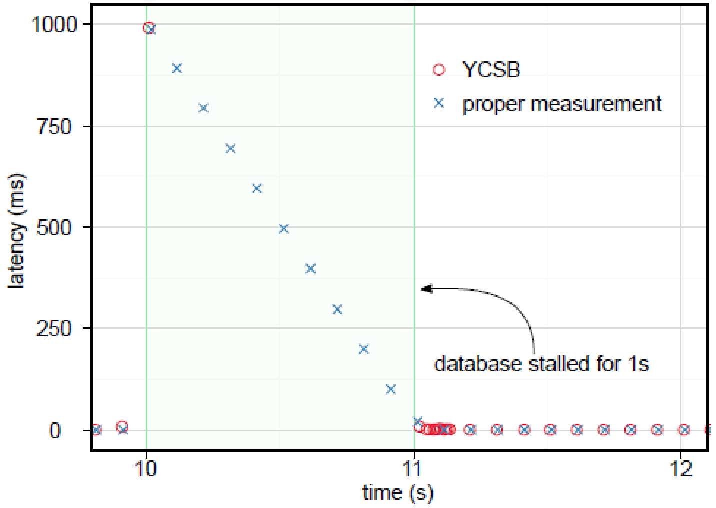
- Bottlenecks
  - Driver (probably multiple machines needed for data generation)
  - Network (GigE is saturated quickly ~ 110MB/s)
- Semantics (event time vs processing time)
- Coordinated Omission (Friedrich et al., SCDM 2017)



(a) Closed system model



(b) Open system model



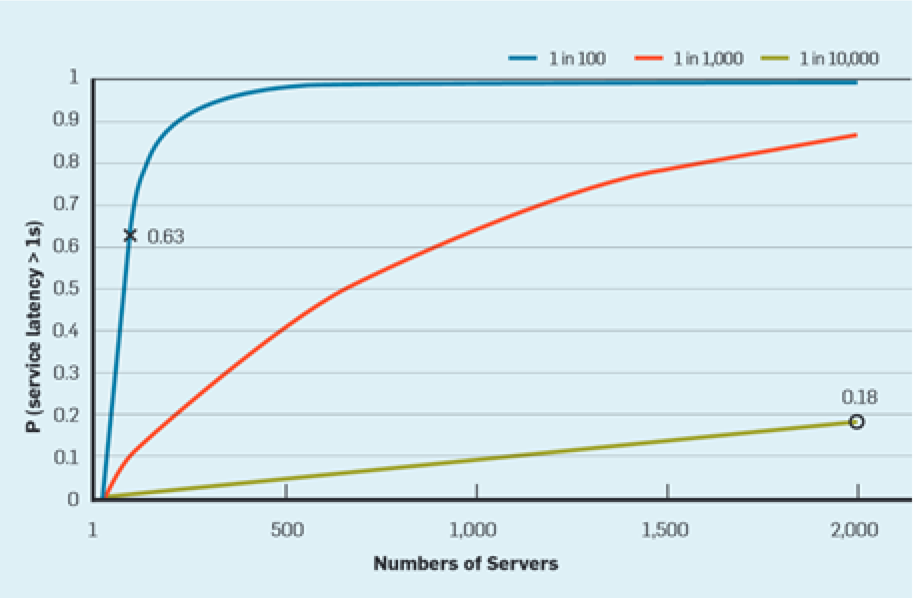
Common Metrics

- Performance
  - Throughput
  - Latency
  - Accuracy
  - Capacity
- Fault-tolerance
  - Time to failure
  - Availability
- Efficiency
  - Energy
  - Cost
  - Fairness
- Scalability

- Selection criteria
  - Low variability
  - Non-redundancy
  - Completeness

Throughput / Latency

- Throughput
  - Requests per second
  - Concurrent users
  - GB/sec processed
  - …
- Latency
  - Execution time
  - Per request latency
- The 95th or 99th percentile request latency
  - End-to-end with all tiers included
  - Larger scale → more prone to high tail latency



[Dean & Barroso,’13]

Capacity

- Nominal Capacity
  - Maximum achievable throughput under ideal workload conditions. E.g., bandwidth in bits per second. The response time at maximum throughput is often too high.
- Usable capacity
  - Maximum throughput achievable without exceeding a pre-specified response-time limit
  - Also: sustainable throughput
- Knee Capacity
  - Knee = Low response time and High throughput

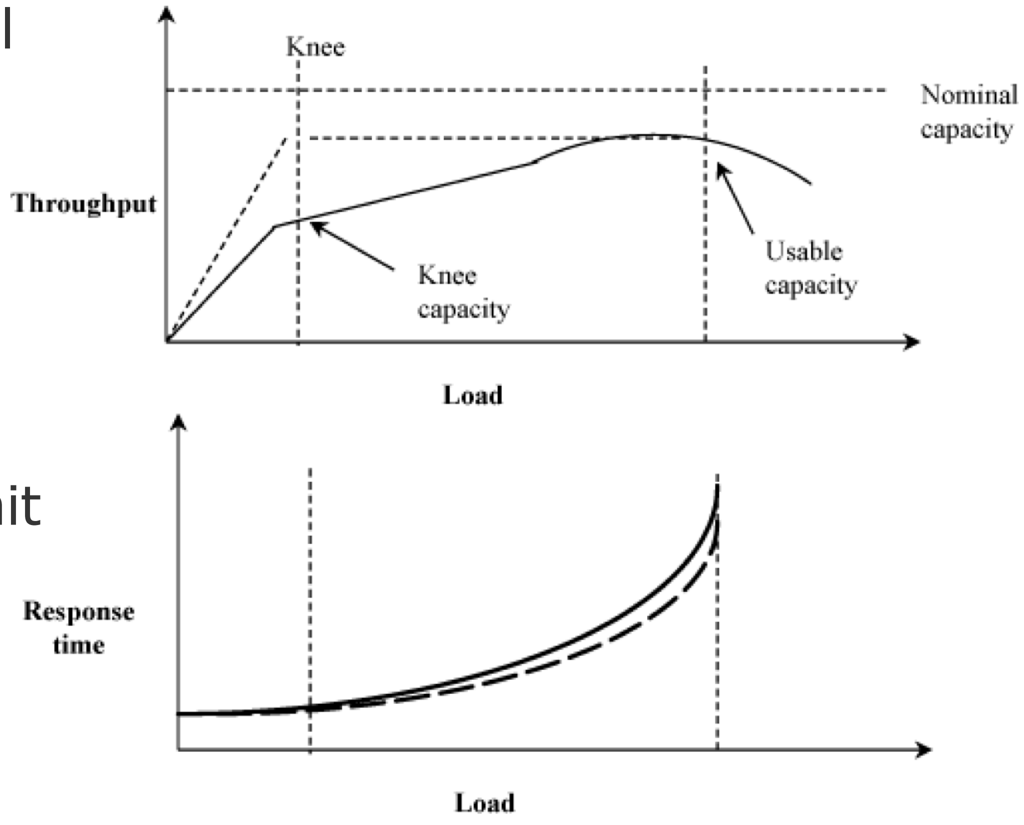


Image source: Raj Jain

Benchmarks

Desire for a Benchmark

- With creating a system comes the need to evaluate it.

- Because we are system programmers, we always think of *performance*, when we think "benchmarking", rather than for example *usability*
  - Other metrics are important as well, like energy efficiency or price-per-performance

- Here: Performance evaluation

Types of Benchmarks

- Micro-benchmarks. To evaluate specific lower-level, system operations
  - E.g., min/max/avg on single laptop…
- Functional / component benchmarks. Specific high-level function.
  - E.g., Sorting: Terasort
  - E.g., Basic SQL: Individual SQL operations, e.g. Select, Project, Join, Order-By, …
- Genre-specific benchmarks. Benchmarks related to type of data
  - E.g., Graph500. Breadth-first graph traversals
- Application-level benchmarks.
  - Measure system performance (hardware and software) for a given application scenario—with given data and workload
- Real applications

Standardized vs. widely used vs. other benchmarks

Micro-Benchmark Example

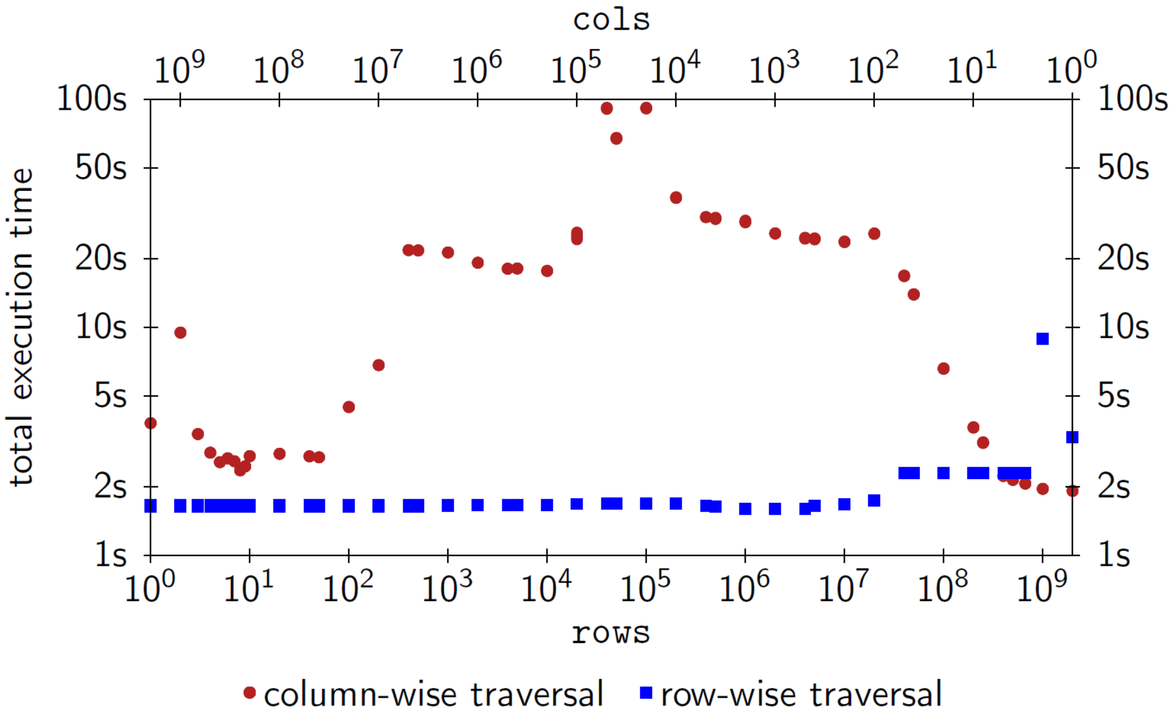
- Add up all numbers of some columns in a 2D array

Variant 1 (row-wise order)

```
for (r = 0; r < rows; r++)
    for (c = 0; c < cols; c++)
        sum += src[r * cols + c];
```

Variant 2 (column-wise order)

```
for (c = 0; c < cols; c++)
    for (r = 0; r < rows; r++)
        sum += src[r * cols + c];
```



Micro-Benchmark Pros/Cons

PROs

- Focused on problem at hand
- Controllable workload and data characteristics
  - Data sets (synthetic & real)
  - Data size / volume (scalability), value ranges and distribution, correlation
- Allow broad parameter range(s)
- Useful for detailed, in-depth analysis
- Low setup threshold; easy to run

CONs

- Neglect larger picture
- Neglect contribution of local costs to global/total costs
- Neglect embedding in context/system at large

→ Generalization of result difficult

Application-level Benchmarks

Example

- Transaction Processing Performance Council (TPC)
  - Benchmark standardization organization
  - Non profit, *vendor neutral*
  - Members (2021): Actian, Alibaba, AMD, Cisco, Dell, Fujitsu, HPE, Hitachi, Huawei, IBM, Inspur, Intel, Lenovo, Microsoft, Nutanix, Oracle, Red Hat, Transwarp, TTA, VMWare

- OLAP: TPC-H, TPC-DS
- OLTP: TPC-C, TPC-E
- Other (with kit)
  - TPCx-BB: Express, Big Data Benchmark
  - TPCx-AI: Express, Artificial Intelligence

Other Relevant Benchmarks

- Star Schema Benchmark (SSB) – O’Neil et al.
  - TPC-H in star schema with simpler, structured query flights
- CH-benCHmark – (TPC-C + TPC-H) – Funke et al.
  - HTAP benchmark – result of a Dagstuhl seminar
  - https://db.in.tum.de/research/projects/CHbenCHmark/
- Yahoo Cloud Serving Benchmark (YCSB) – Cooper et al.
  - Key-Value store benchmark
  - https://github.com/brianfrankcooper/YCSB
- Join Order Benchmark (JOB) – Leis et al.
  - 113 complex join queries on IMDB
  - https://github.com/gregrahn/join-order-benchmark

Real Applications

- Take an application, put it on top of your system and see how it runs
- Pro:
  - So many real life applications out there
  - Real challenges, no academic or simplified view
- Cons:
  - So many real life applications out there
  - Proprietary datasets and workloads (confidential information)
  - Not scalable
- Often, such applications are analyzed and a synthetic benchmark is designed after them
  - BigBench uses distributions from Census Bureau, Amazon data sets, etc.
  - Or use a data generator to synthesize your own data!
  - Synthetic data does not have privacy/security/scalability issues
- ***A good thesis/paper has micro benchmarks, application level benchmarks, and some real application/data!***

Comparing to Other Systems/Work

- Ensure
  - All systems have equal functionality
  - You are able to reproduce original numbers
- Ask the original authors
  - Most people are helpful and happy if you are interested in their work

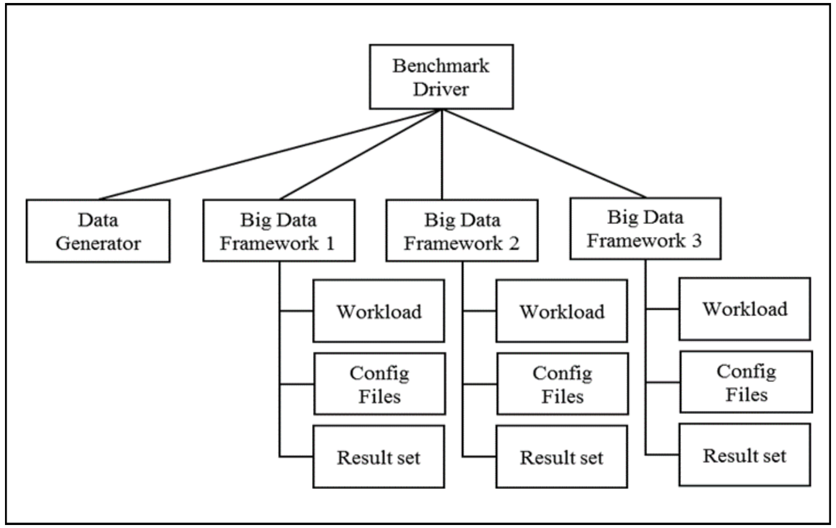
BigBench / TPCx-BB – Big Data Benchmark

The BigBench Proposal

End-to-end, application level benchmark

Focused on Parallel DBMS and MR engines

- Framework agnostic
- SW based reference implementation



History

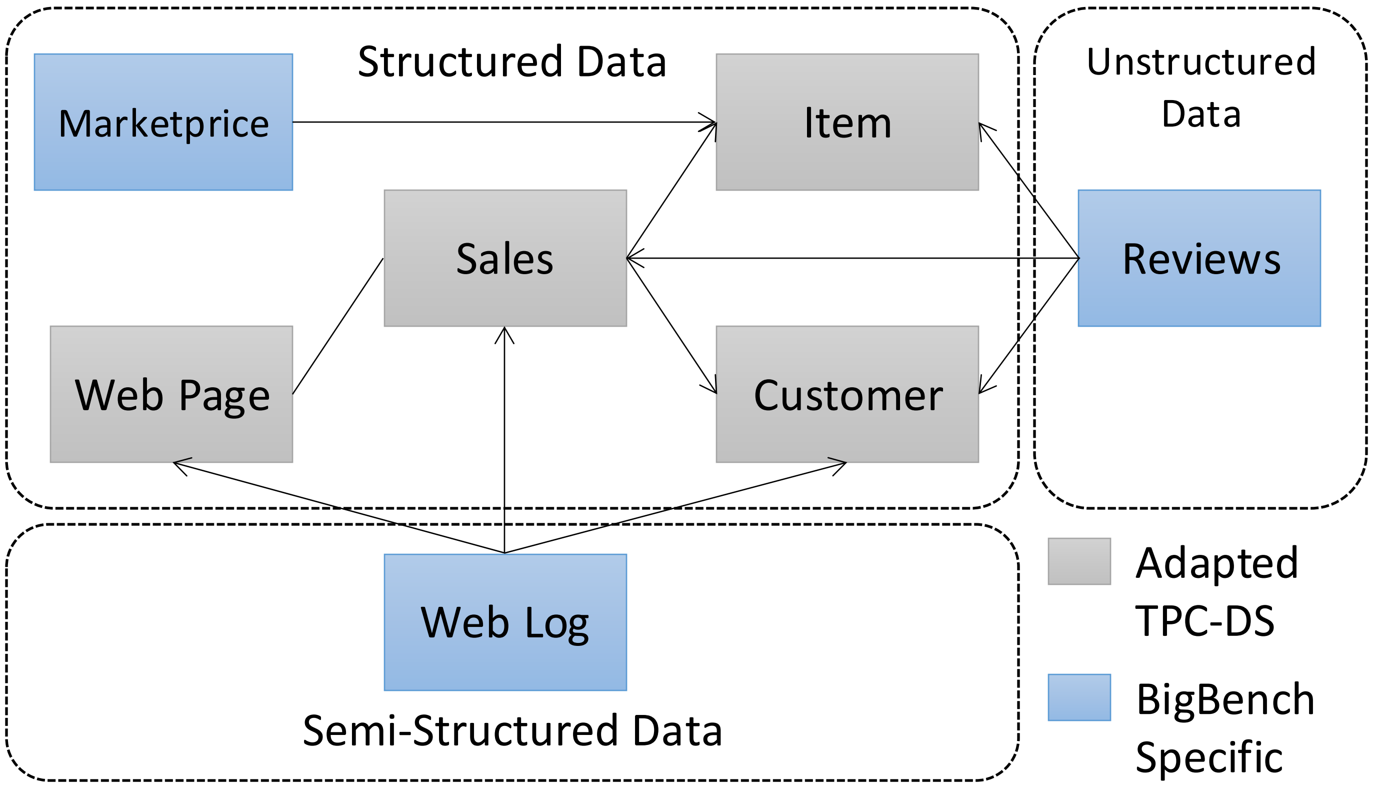
- Launched at 1st WBDB, San Jose, 2012
- Published at SIGMOD 2013
- Full kit at WBDB 2014
- TPC BigBench Working Group in 2015
- TPCx-BB standardized in Jan 2016
- First published result Mar 2016

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, …

BigBench Data Model

- Structured: TPC-DS + market prices
- Semi-structured: website click-stream
- Unstructured: customers' reviews



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 business 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.

```
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;
```

Benchmark Process – TPCx-BB

No update

Measured processes

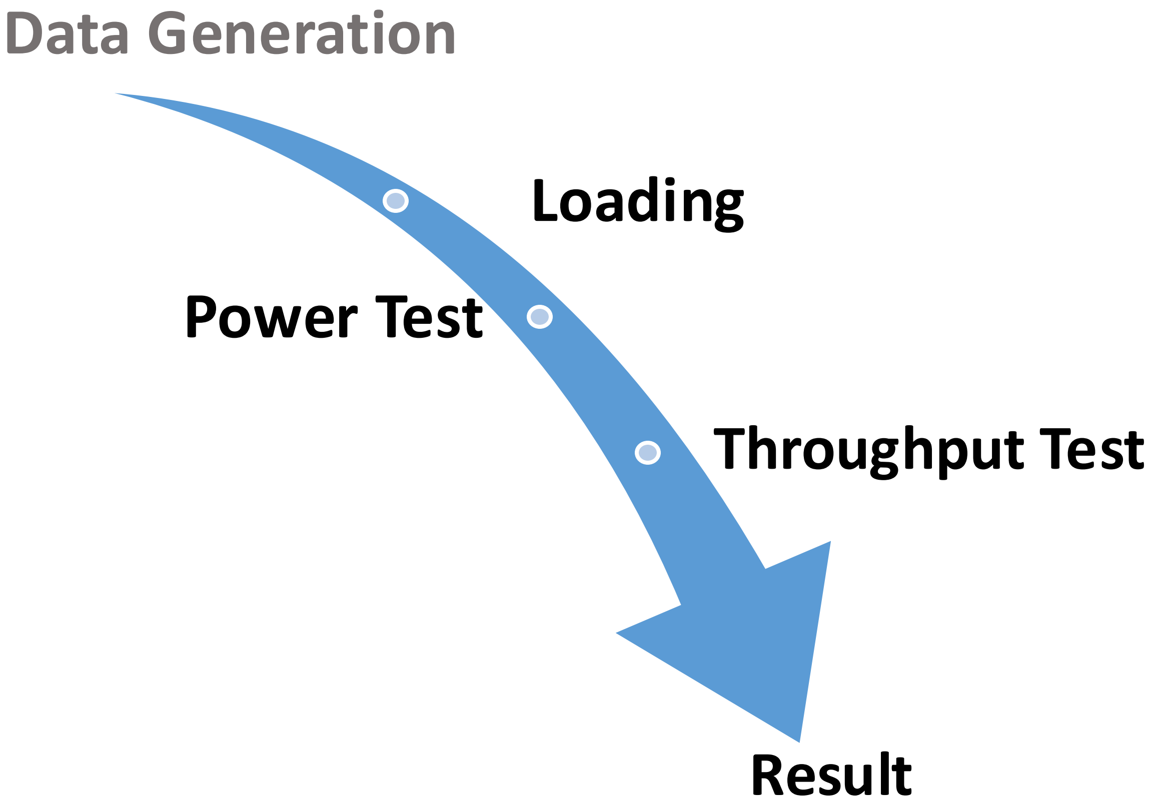
- Loading
- Power Test (single user run)
- Throughput Test (multi user run)

Result

- Mixed metric

Two runs

- Lower number reported

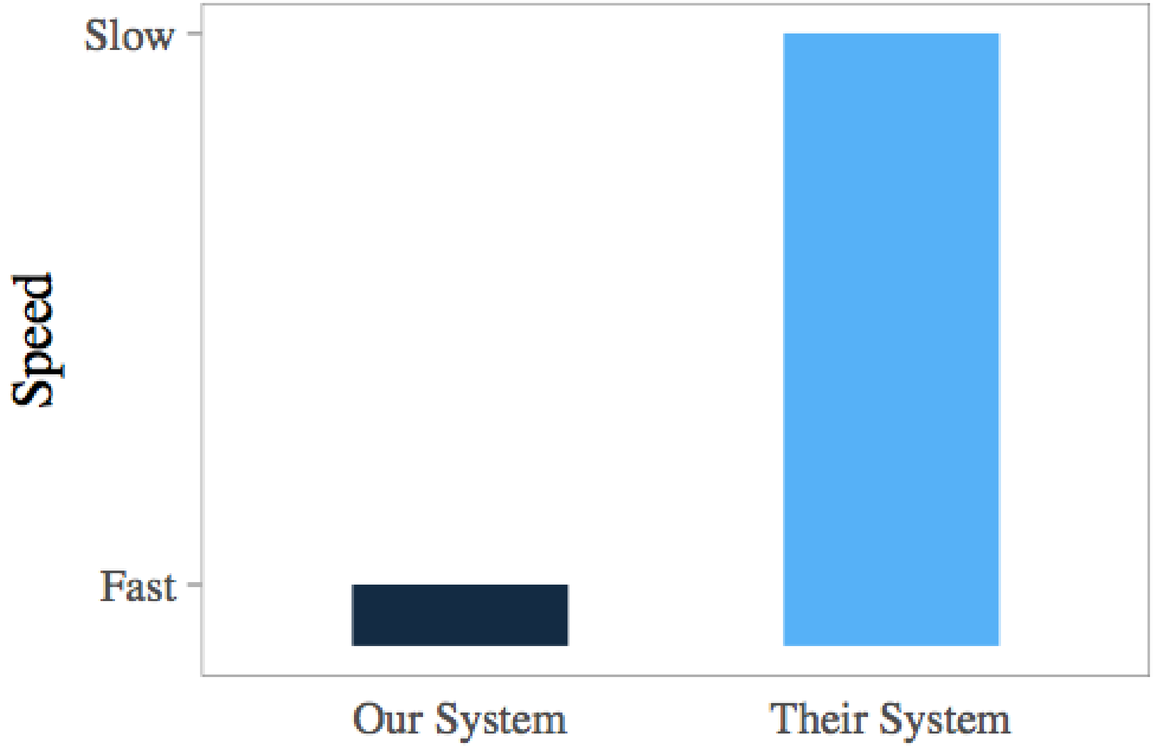


Fair Benchmarking

Comparing to Other Systems/Work

- Ensure
  - All systems have equal functionality
  - You are able to reproduce original numbers
- Ask the original authors
  - Most people are helpful and happy if you are interested in their work

The Root Cause of Problems



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

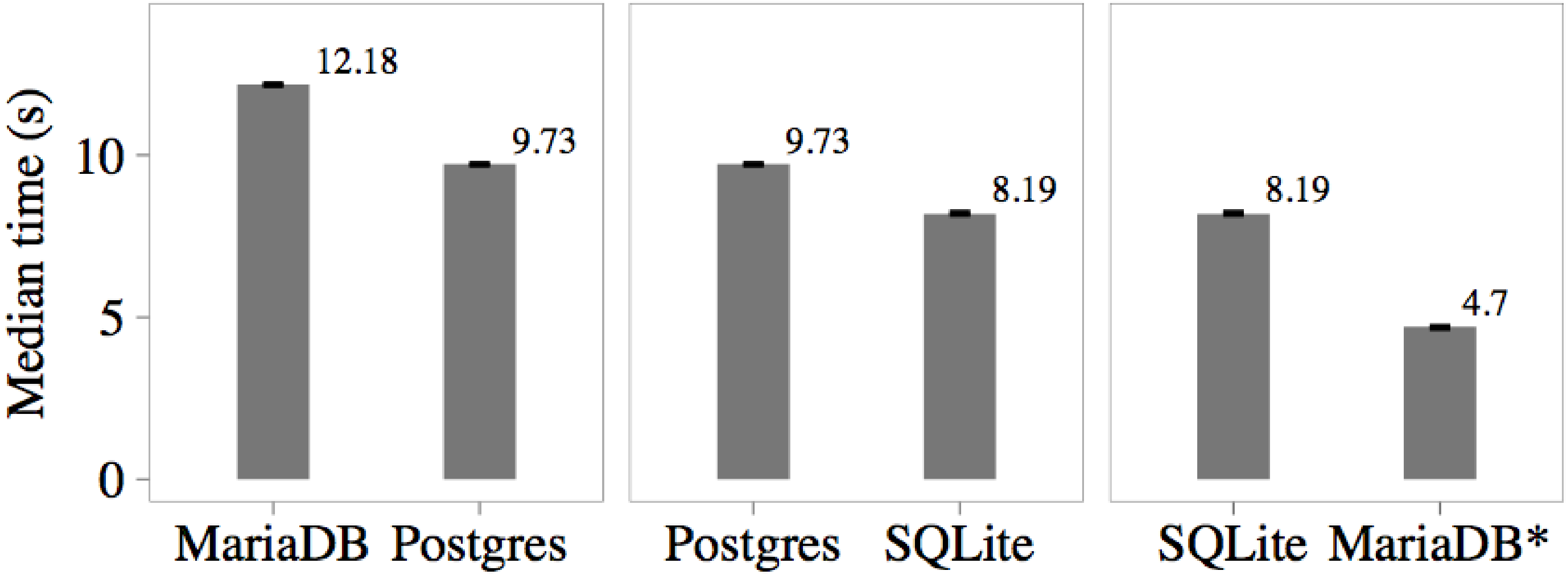
Source: Fair Benchmarking Considered Difficult: Common Pitfalls In Database Performance Testing. Raasveldt et al. DBTest ‘18

Benchmarking Games

- Differing configurations
- Hard-wired optimizations
- Specification biased to system
- Synchronized workload queue
- Arbitrary workload
- Very small benchmark
- Benchmark manually translated for performance

Source: Fair Benchmarking Considered Difficult: Common Pitfalls In Database Performance Testing. Raasveldt et al. DBTest '18

Same query & data (TPCH SF1 Q1)



- What's the crime?

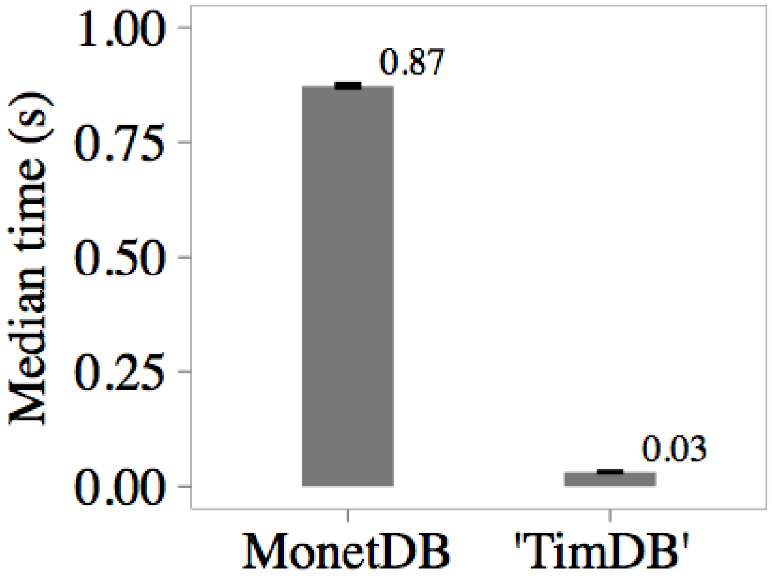
Source: Fair Benchmarking Considered Difficult: Common Pitfalls In Database Performance Testing. Raasveldt et al. DBTest '18

Same query & data (TPCH SF1 Q1)

- …same configuration parameters
- …same compilation flags
- …same version number of the database
- …different schema!
  - DOUBLE instead of DECIMAL
- Still gives correct results according to TPC-H specification

Source: Fair Benchmarking Considered Difficult: Common Pitfalls In Database Performance Testing. Raasveldt et al. DBTest '18

Same query & data (TPCH SF1 Q1)



TimDB is hand-rolled standalone C program for Q1
TimDB is not a database. Common misrepresentation.

Source: Fair Benchmarking Considered Difficult: Common Pitfalls In Database Performance Testing. Raasveldt et al. DBTest '18

Incorrect Results

- Bugs sometimes make code very fast
  - But incorrect, may be invisible in benchmark
- Always check results
- Run with different benchmark and dataset, too
- E.g. run with PostgreSQL and compare results

```
void tpchq1() {
   return;
}
```

Even TimDB can't beat!

Source: Fair Benchmarking Considered Difficult: Common Pitfalls In Database Performance Testing. Raasveldt et al. DBTest '18

Summary

- Introduction to performance analysis
- Back of the envelope calculations
- Measurement
- Benchmarks
- BigBench
- Fair benchmarking

- Questions?
  - Per email: tilmann.rabl@hpi.de

Thank you for your attention!

- Questions?
  - In Moodle
  - Per email: martin.boissier@hpi.de
  - In Q&A sessions