Séminaires Big Data
DECOL

- **ETL et Reporting pour le Big-Data: Acquisition, Intégration, et Restitution de Données.**
  - Roukia Eladioui, CGI France, 25 novembre 14h-15h

- **Défis et Technologies du Big-Data dans le marché Informatique**
  - Christophe Menichetti, IBM Montpellier, 9 décembre, 14h-17h

Organisation

(12/10) 2CM+ TP : Panoramique (début mini-projet)

~~(19/10)~~ pas de cours

(21/10) CM+ 2TP : Zoom sur les Graphes

(26/10) 3TP

(7/11) rendu partie 1 mini-projet : moteur

(9/11) CM+ 2TP : Évaluation Performances

(14/11) rendu partie 2 mini-projet : analyse perf.

(16/11) Soutenances

Performance Evaluation in Databases

Federico Ulliana

Slides collected from Ioana Manolescu, Stefan Manegold, John Mellor-Crummey and Fabian Suchanek

Database Performance Evaluation

- There is no single way how to do it right.
- There are many ways how to do it wrong.
- This is more a collection of anecdotes
  - providing some guidelines for what to/not-to do.

Make experiments

Practical work requires experiments

Experiments should show
- that a system works
- that a system works better than another one

Saying that a solution is more sophisticated and hence better is NOT a valid argument!

Warning : Making experiments is not science, is an art.

Questions we will try to answer today

Planning and Conducting **Database** Experiments

- Which data / data sets should be used?
- Which workload / queries should be run?
- Which hardware & software should be used?

Metrics:

- What to measure ? How to measure?
- How to compare ?
- How to find out what is going on ?

LET’S START FROM BENCHMARKS

What is a **Database** Benchmark ?

«A framework to assess the abilities of a database »

It typically contains

- data
- queries / updates
- other specific operations

Data sets and workloads

- Micro-benchmarks
- Standard benchmarks
- Real-life applications

To recognize them, ask yourself : who designs it ?

Micro-benchmarks

- Specific, stand-alone piece of code (ususally written by the developers) isolating a particular piece of a larger system

- E.g., a single database operator
  - dictionnary creation
  - merge-join
  - aggregation
  - update

Micro-benchmarks : Pros

- Focused on a single problem
- Controllable workload and data characteristics
  - Data (synthetic & real) and size (scalability)
  - Value ranges and distribution correlation
  - Type of Queries and Workload size (scalability)
- Allow broad parameter range(s)
  - Useful for detailed, in-depth analysis
- Low setup threshold; easy to run

Micro-benchmarks : Cons

- Generalization of result sometimes difficult
- Neglect larger picture
  - e.g. contribution of local costs to global/total costs
- Neglect embedding in context/system at large
  - Application of insights in full systems / real-life applications no obvious

Standard Benchmarks

- Collaboratively-written benchmarks (by consortiums) aiming at testing a whole system

- RDBMS, OODBMS, ORDMBS: TPC-{A,B,C,H,R,DS}, OO7, ...
- XML, XPath, XQuery, XUF, SQL/XML: MBench, XBench, XMach-1, XMark, X007, TPoX, ...
- General Computing: SPEC (Standard Performance Evaluation Corporation), ...

Standard Benchmarks :

TPC (Transaction Processing Performance Council)

TPC-C - Ten Most Recently Published Results

As of 4-Nov-2015 at 1:40 PM [GMT]

Note 1: The TPC believes it is not valid to compare prices or price/performance of results in different currencies.

Date Submitted	Company	System	Performance (tpmC)	Price/tpmC	Watts/KtpmC	System Availability	Database
11/25/14	SAP	Dell PowerEdge T620	112,890	.19 USD	NR	11/25/14	SQL Anywhere 16
03/26/13	ORACLE	SPARC T5-8 Server	8,552,523	.55 USD	NR	09/25/13	Oracle 11g Release 2 Enterprise Edition with Oracle Partitioning
02/22/13	IBM	IBM System x3650 M4	1,320,082	.51 USD	NR	02/25/13	IBM DB2 ESE 9.7

'NR' in the Watts/Ktpmc column indicates that no energy data was reported for that benchmark.

Standard Benchmarks : Pros

- Mimic real-life scenarios
- Publicly available
- Well defined and scalable data sets and workloads (if well designed ...)
- Metrics well defined
- Easily comparable (?)

Standard Benchmarks : Cons

- Often “outdated” (standardization takes long)
- Often compromises
- Often very large and complicated to run
- Systems often optimized for the benchmark(s) !!!
- Limited dataset / workload variation

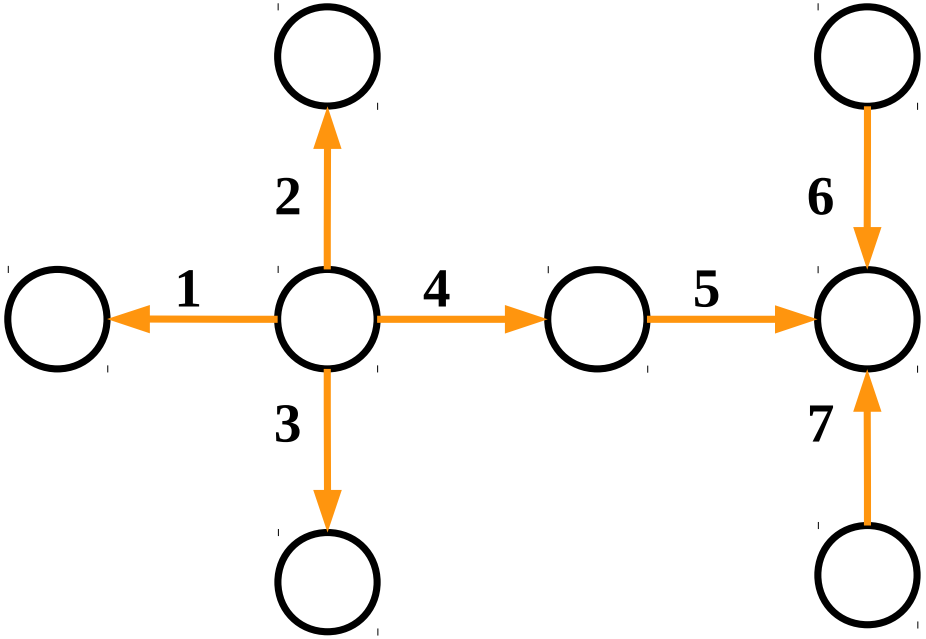
RDF BENCHMARK ANALYSIS

Benchmark Analysis

- Often benchmarks are judged by the data they can produce
- But workloads are important as well !!
- We will survey some structural properties of queries and see whether they are testable by popular RDF benchmarks

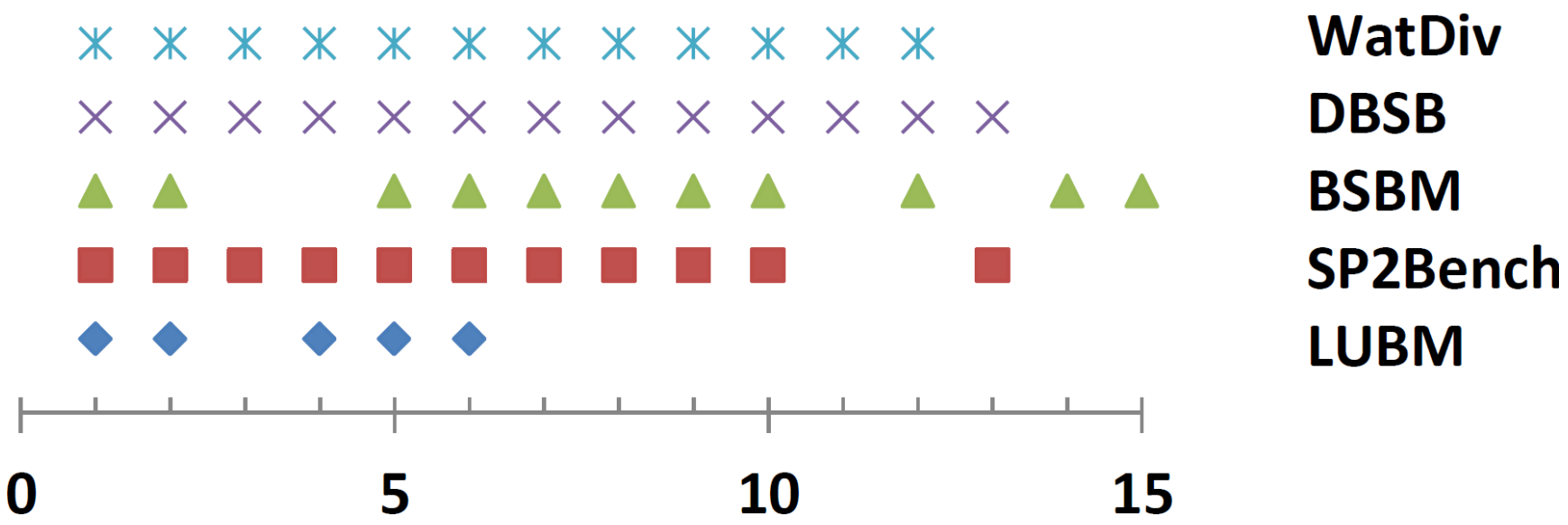
Structural Features

[Triple Pattern Count]



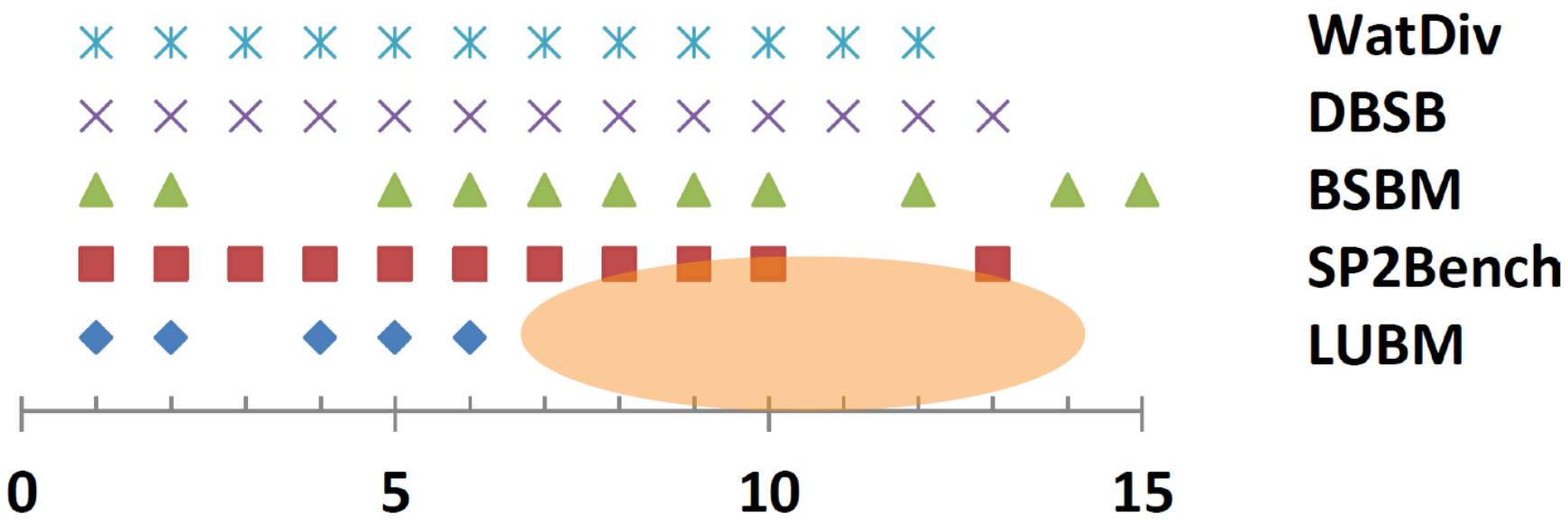
How Diverse are SPARQL Benchmarks?

[Triple Pattern Count]



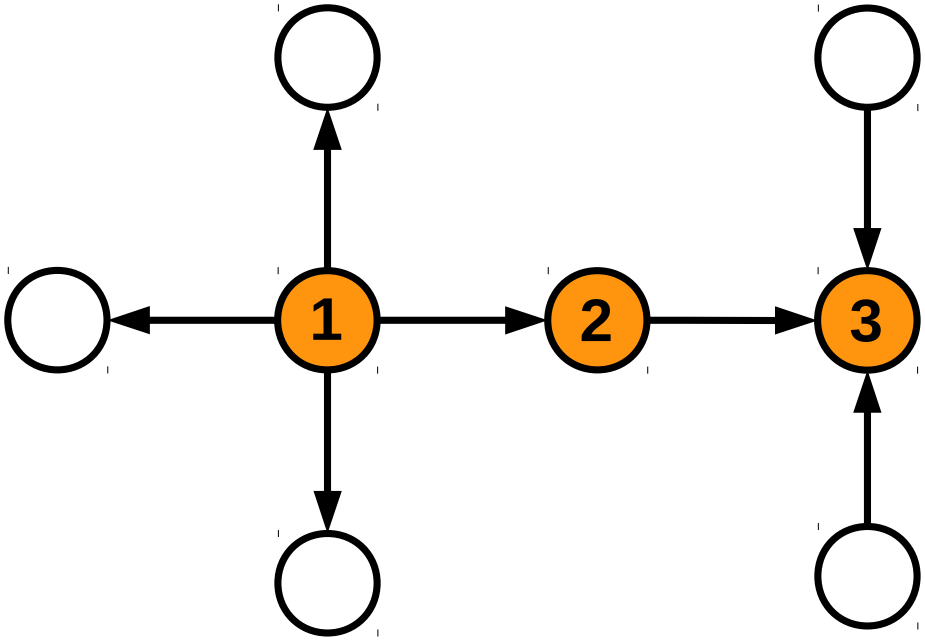
How Diverse are SPARQL Benchmarks?

[Triple Pattern Count]



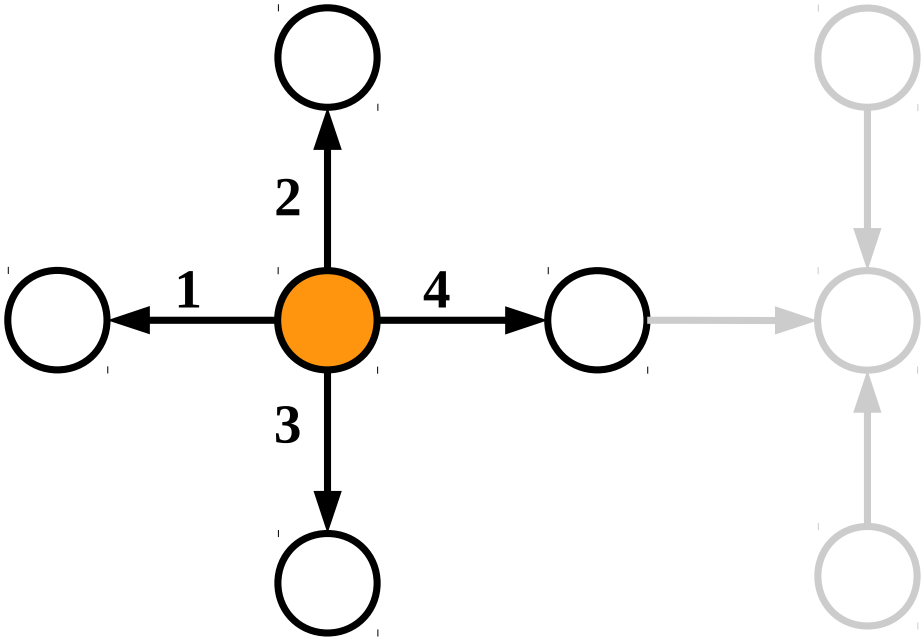
Structural Features

[Join Vertex Count]



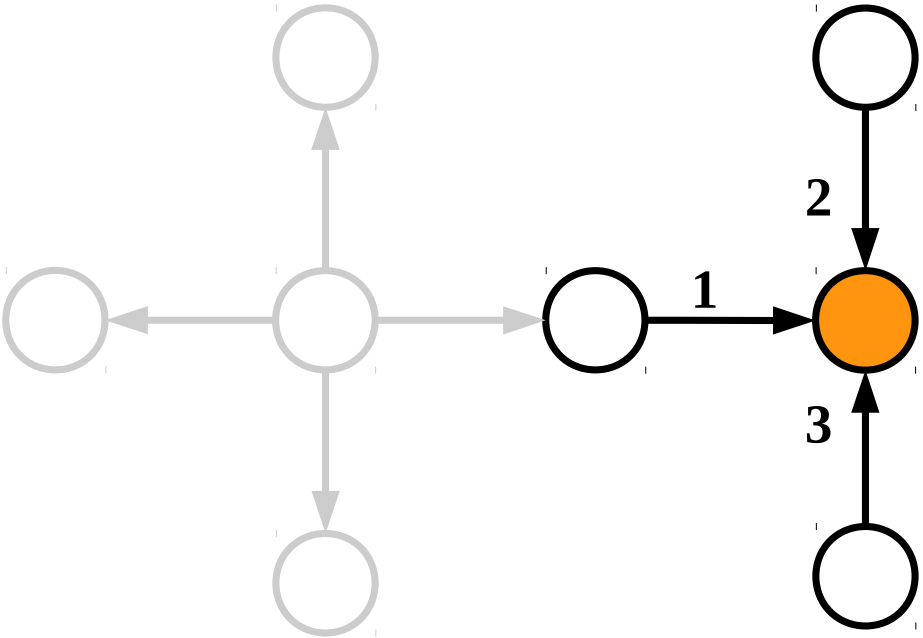
Structural Features

[Join Vertex Degree]



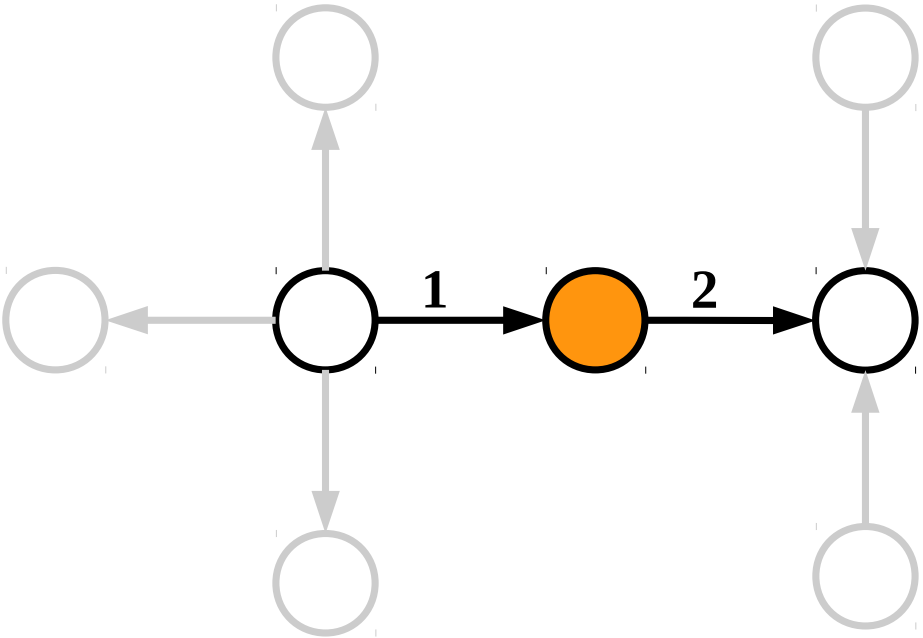
Structural Features

[Join Vertex Degree]

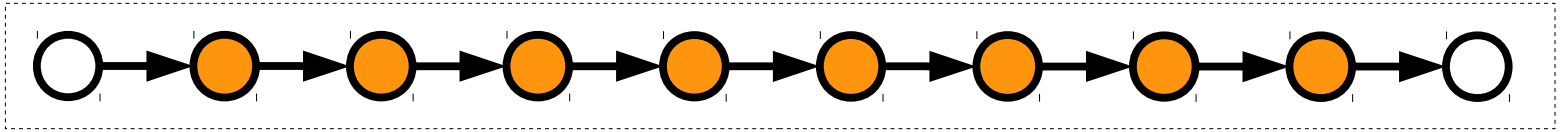


Structural Features

[Join Vertex Degree]

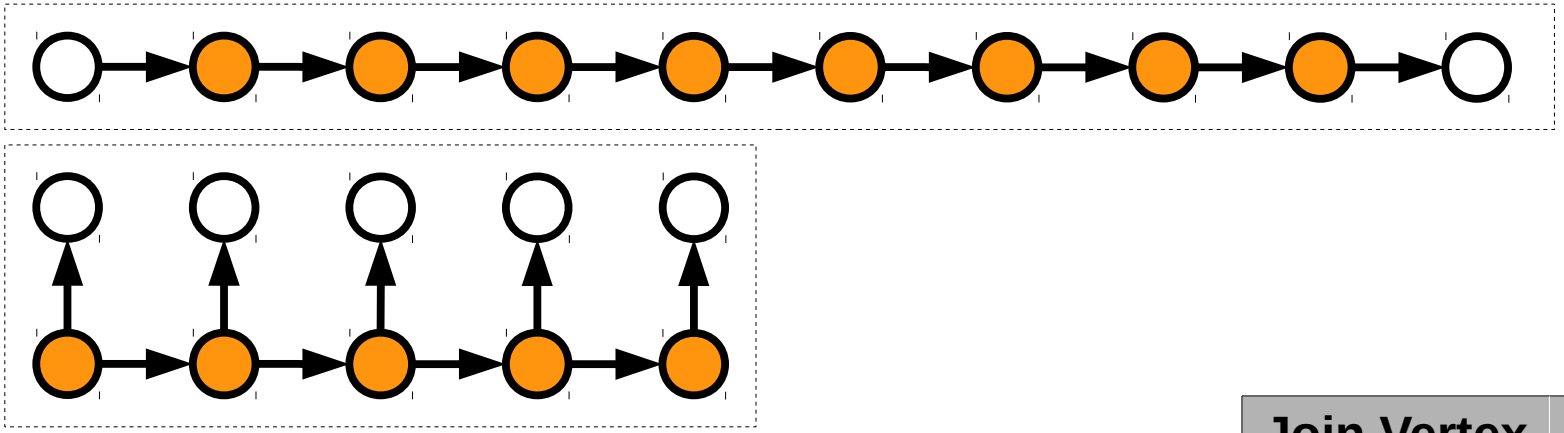


Structural Features



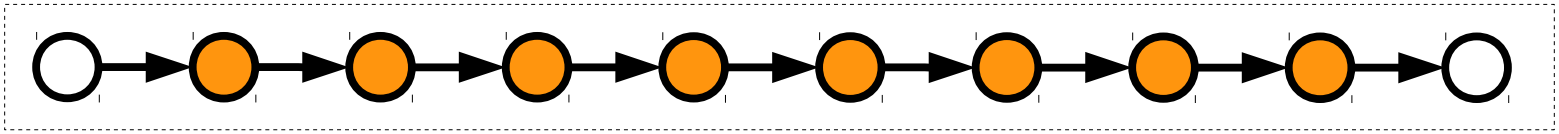
Join Vertex Count	Mean Join Vertex Degree
8	*2.0*

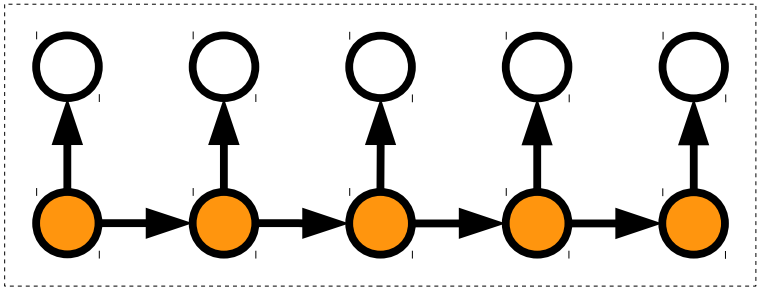
Structural Features

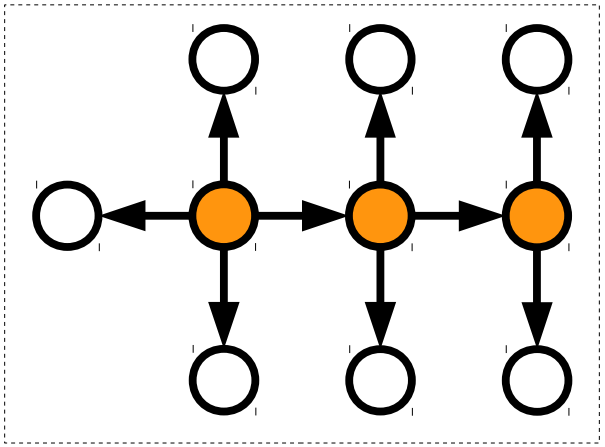


Join Vertex Count	Mean Join Vertex Degree
8	*2.0*
5	***2.6***

Structural Features

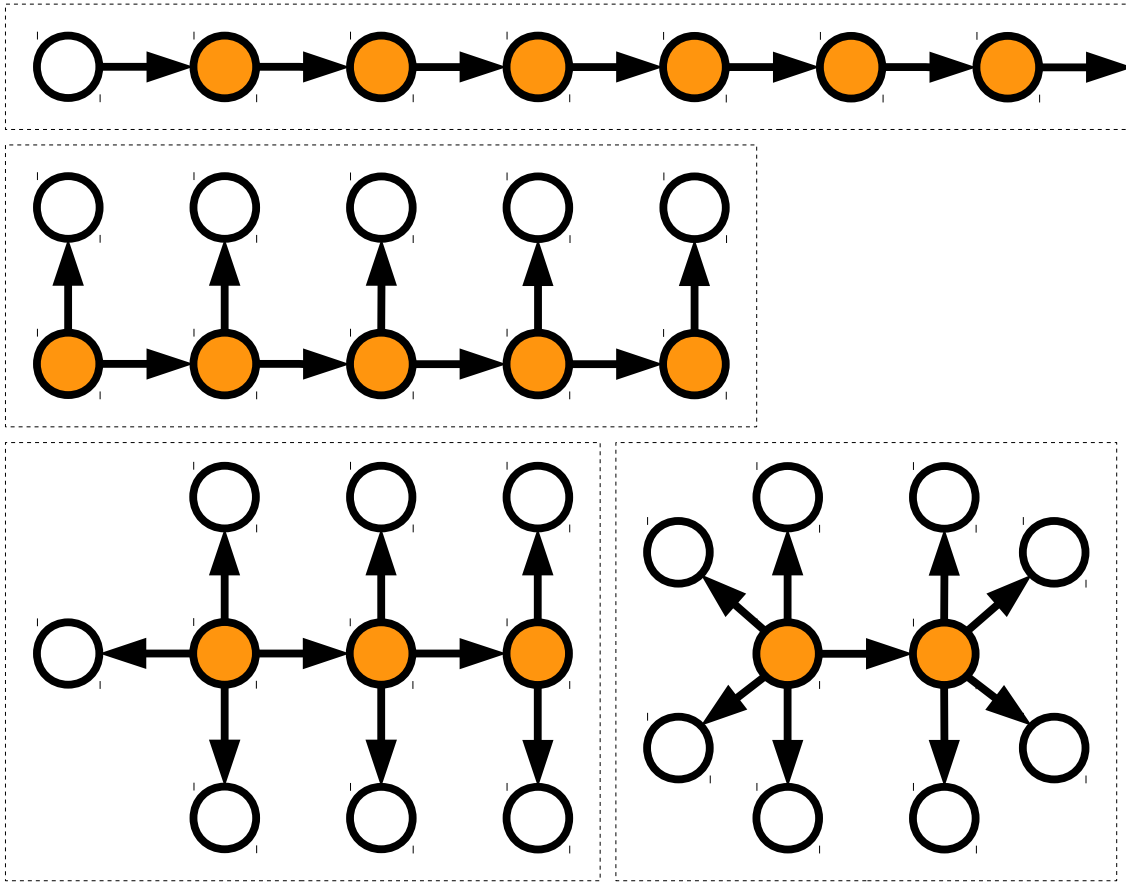






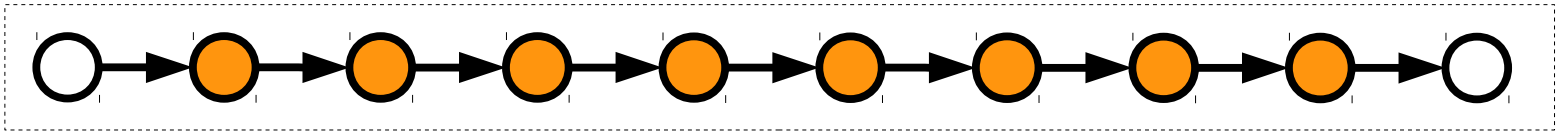
Join Vertex Count	Mean Join Vertex Degree
8	*2.0*
5	*2.6*
3	***~3.7***

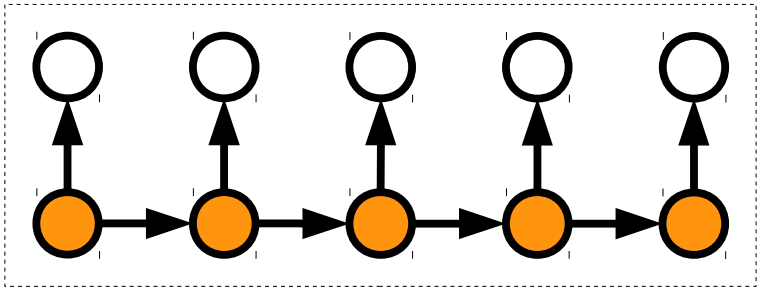
Structural Features

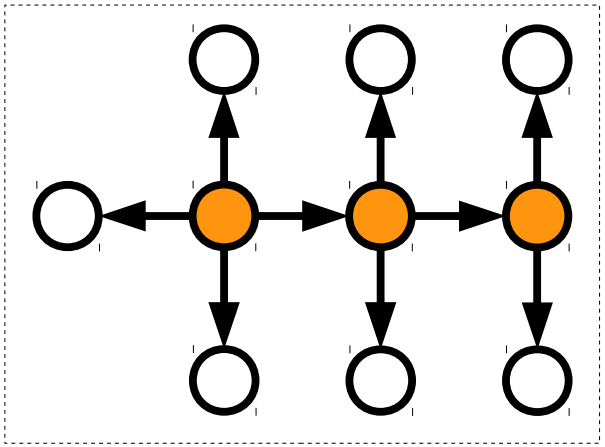


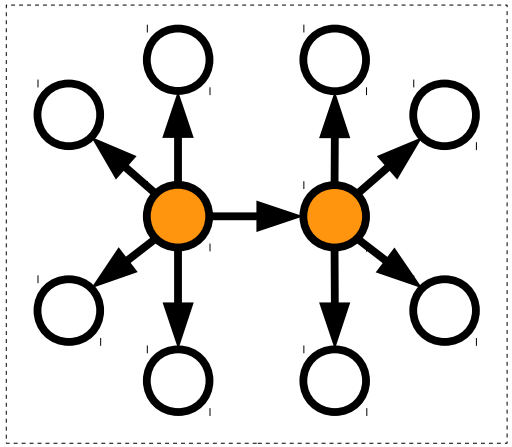
Join Vertex Count	Mean Join Vertex Degree
8	*2.0*
5	*2.6*
3	*~3.7*
2	***5.0***

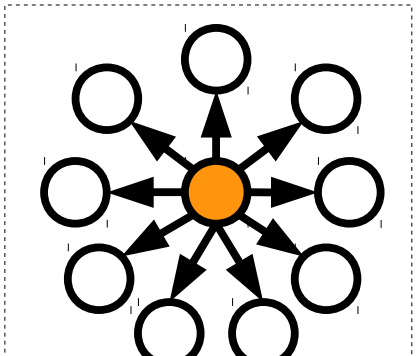
Structural Features







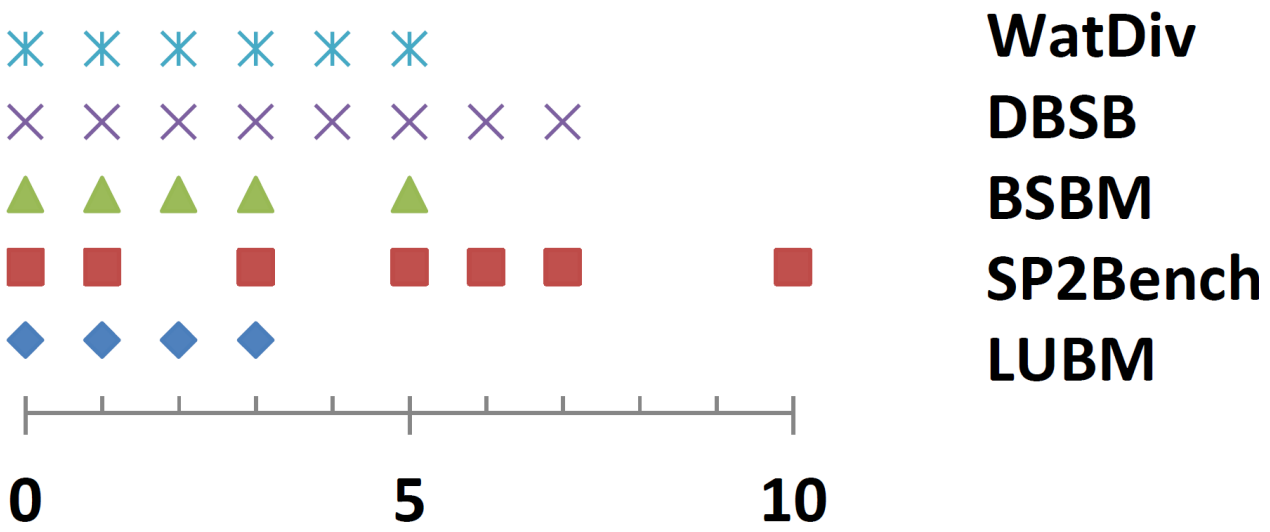




Join Vertex Count	Mean Join Vertex Degree
8	*2.0*
5	*2.6*
3	*~3.7*
2	*5.0*
1	***9.0***

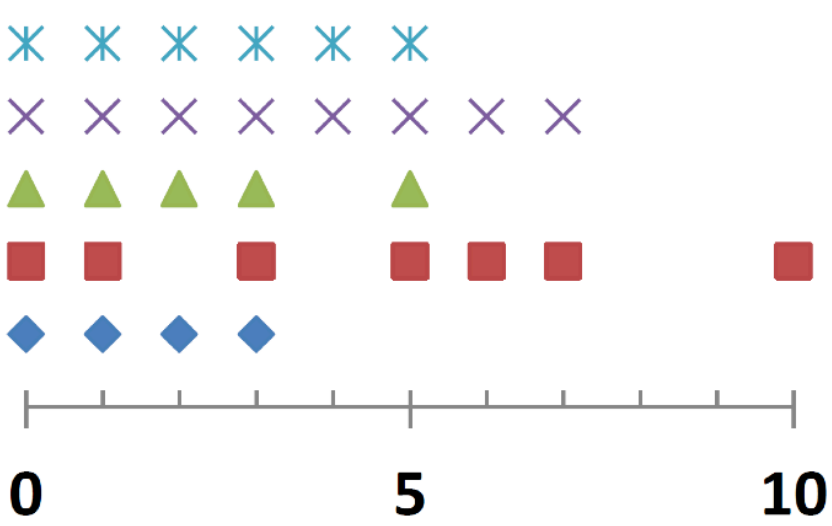
How Diverse are SPARQL Benchmarks?

[Join Vertex Count]



How Diverse are SPARQL Benchmarks?

[Join Vertex Count]



WatDiv

DBSB

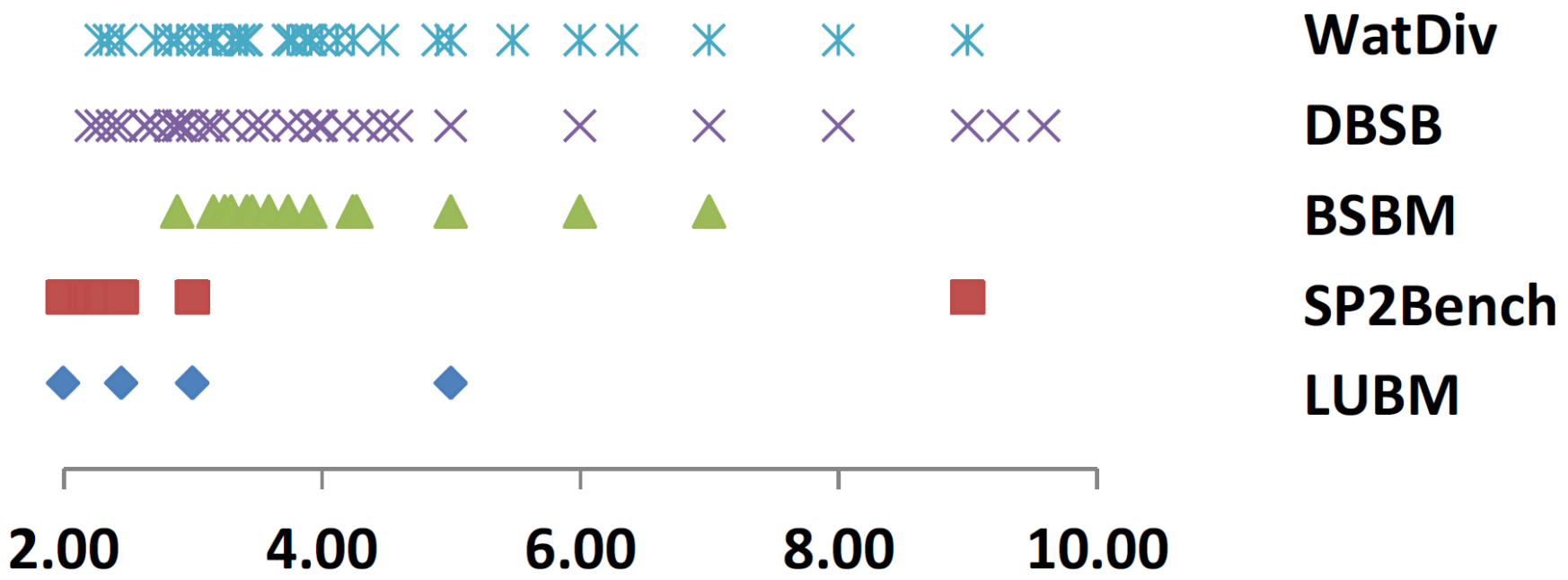
BSBM

SP2Bench

LUBM

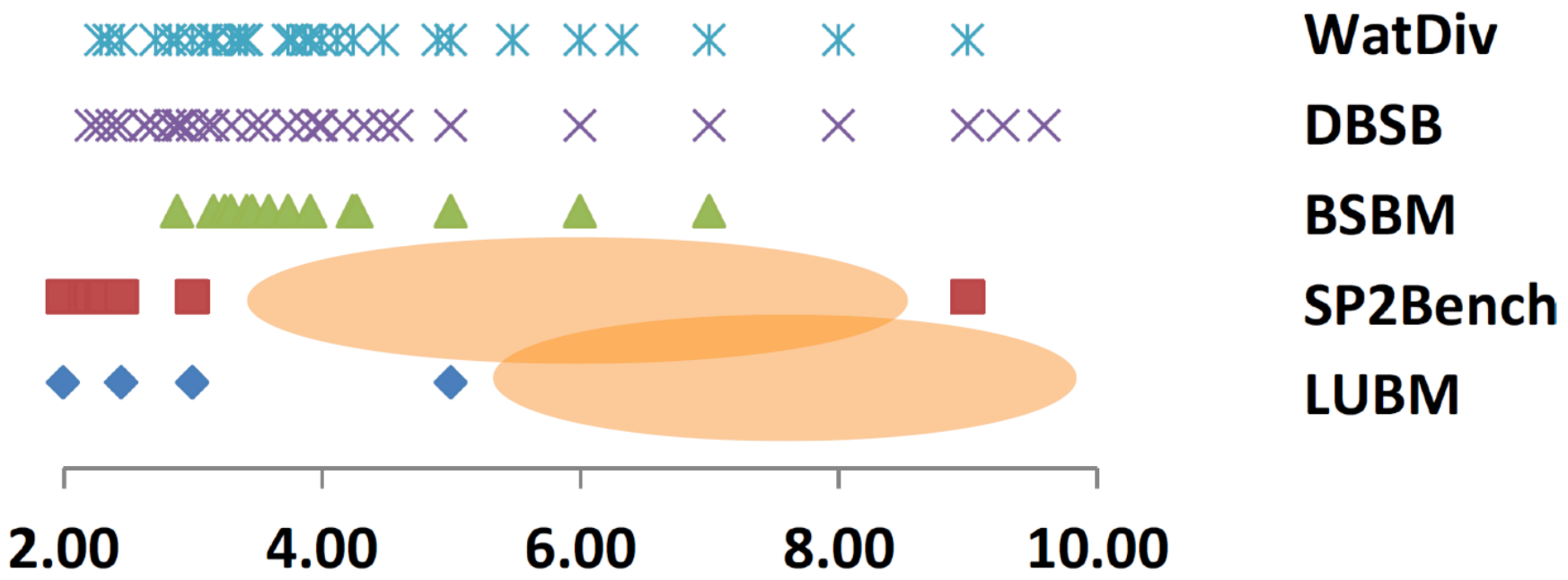
How Diverse are SPARQL Benchmarks?

[Join Vertex Degree – mean]



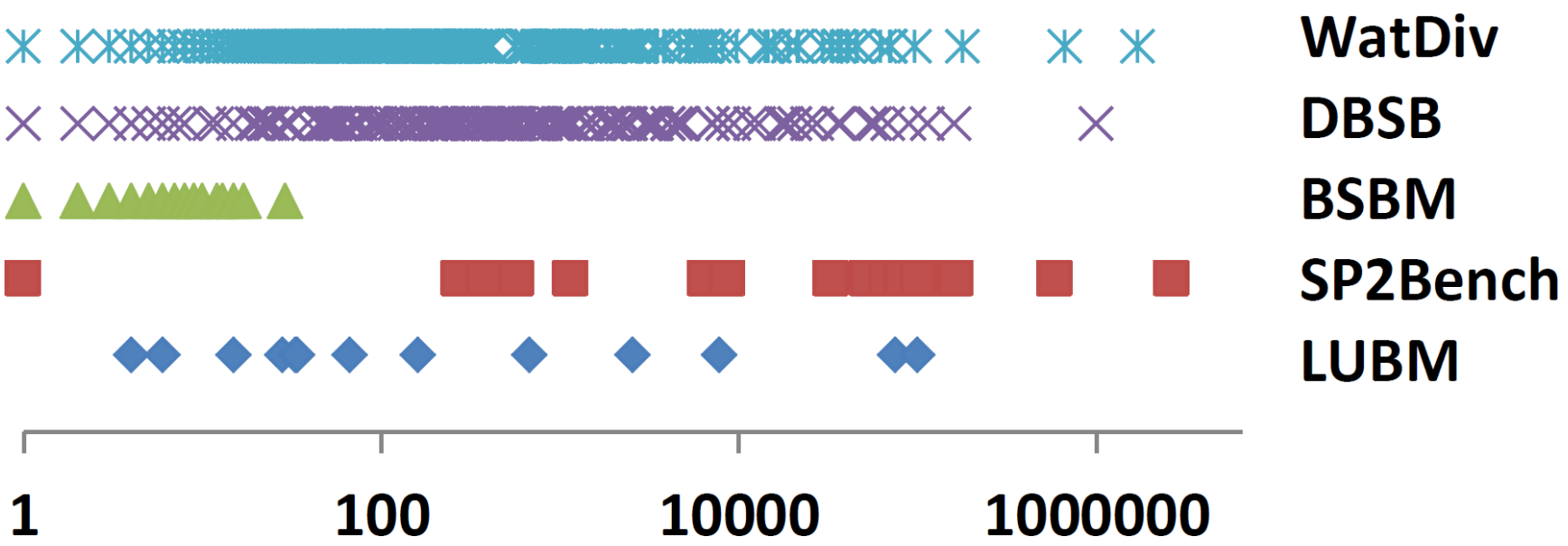
How Diverse are SPARQL Benchmarks?

[Join Vertex Degree – mean]



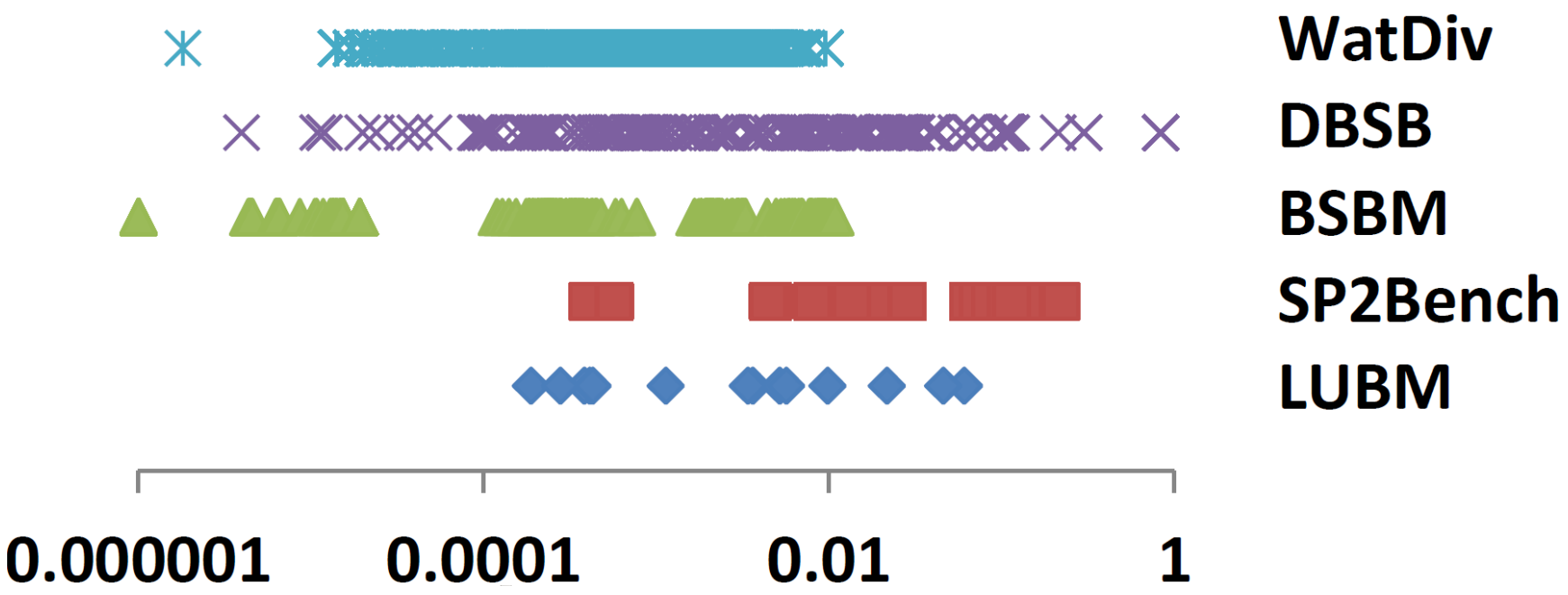
How Diverse are SPARQL Benchmarks?

[Result Cardinality]



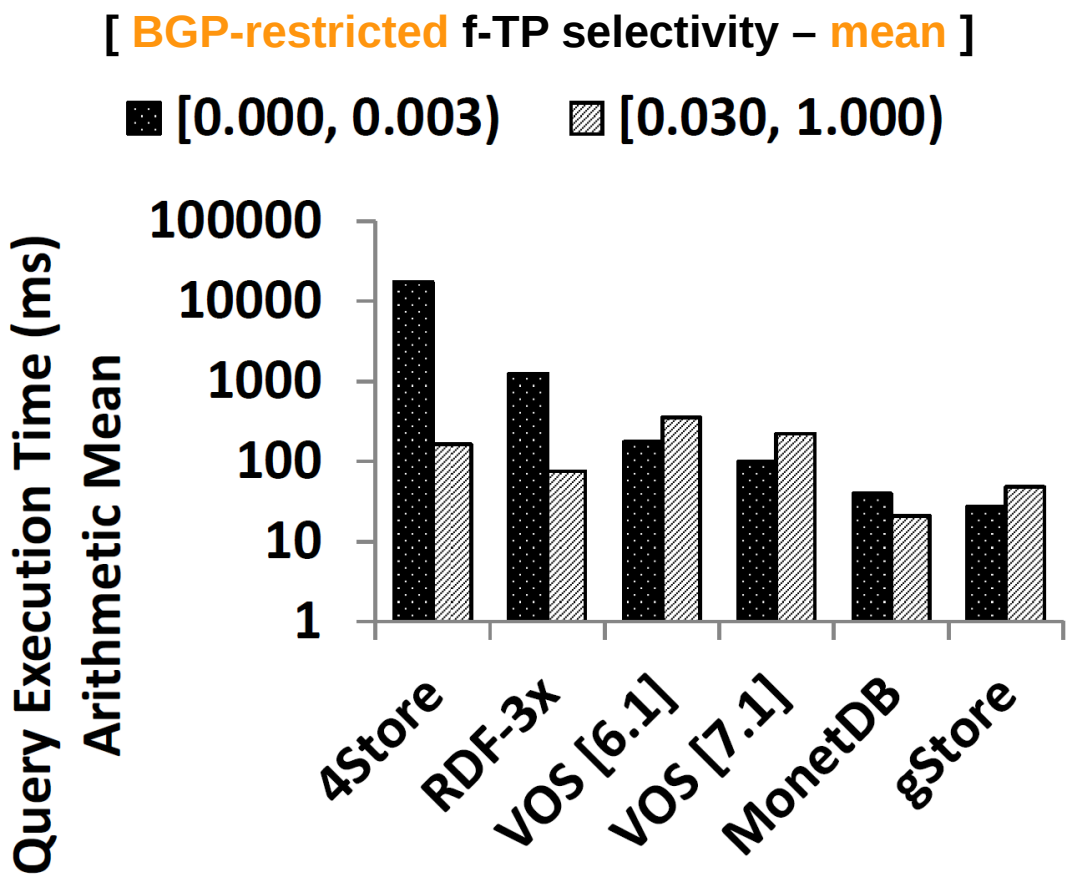
How Diverse are SPARQL Benchmarks?

[f-TP Selectivity – mean]



How Robust are Systems across WatDiv Workloads?

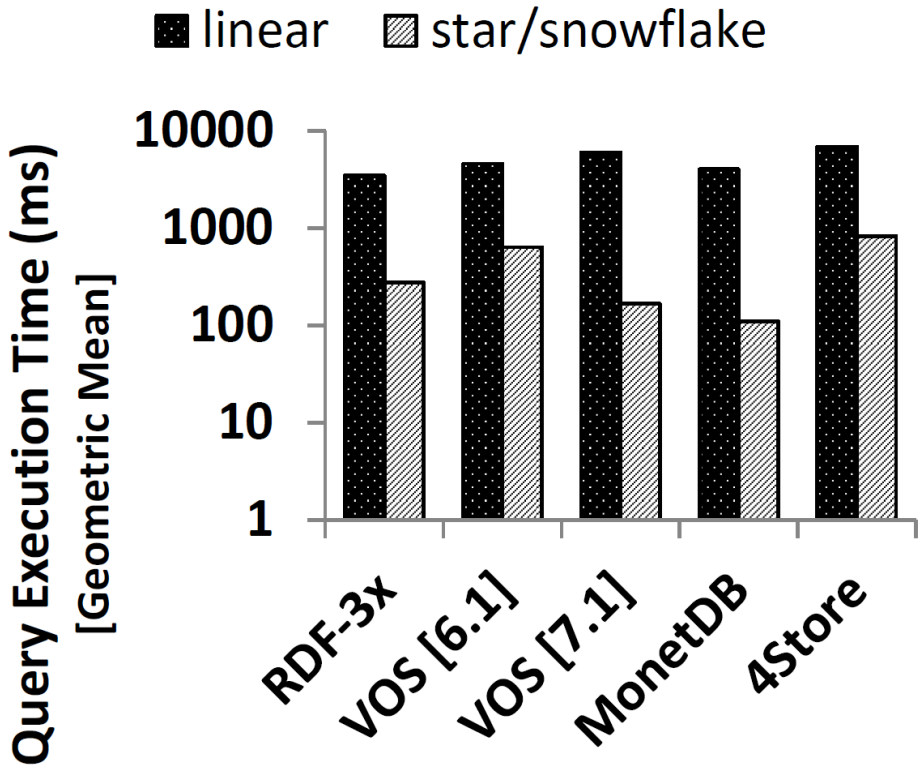
GOOD NEWS



WatDiv 100M triples, queries w/ single join vertex, result cardinality ≤ 2000

How Robust are Systems across WatDiv Workloads?

GOOD NEWS



WatDiv 10M triples

linear = { mean join vertex degree ≤ 3.0, join vertex count ≥ 3}

star/snowflake = { mean join vertex degree ≥ 5.0, join vertex count ≤ 2}

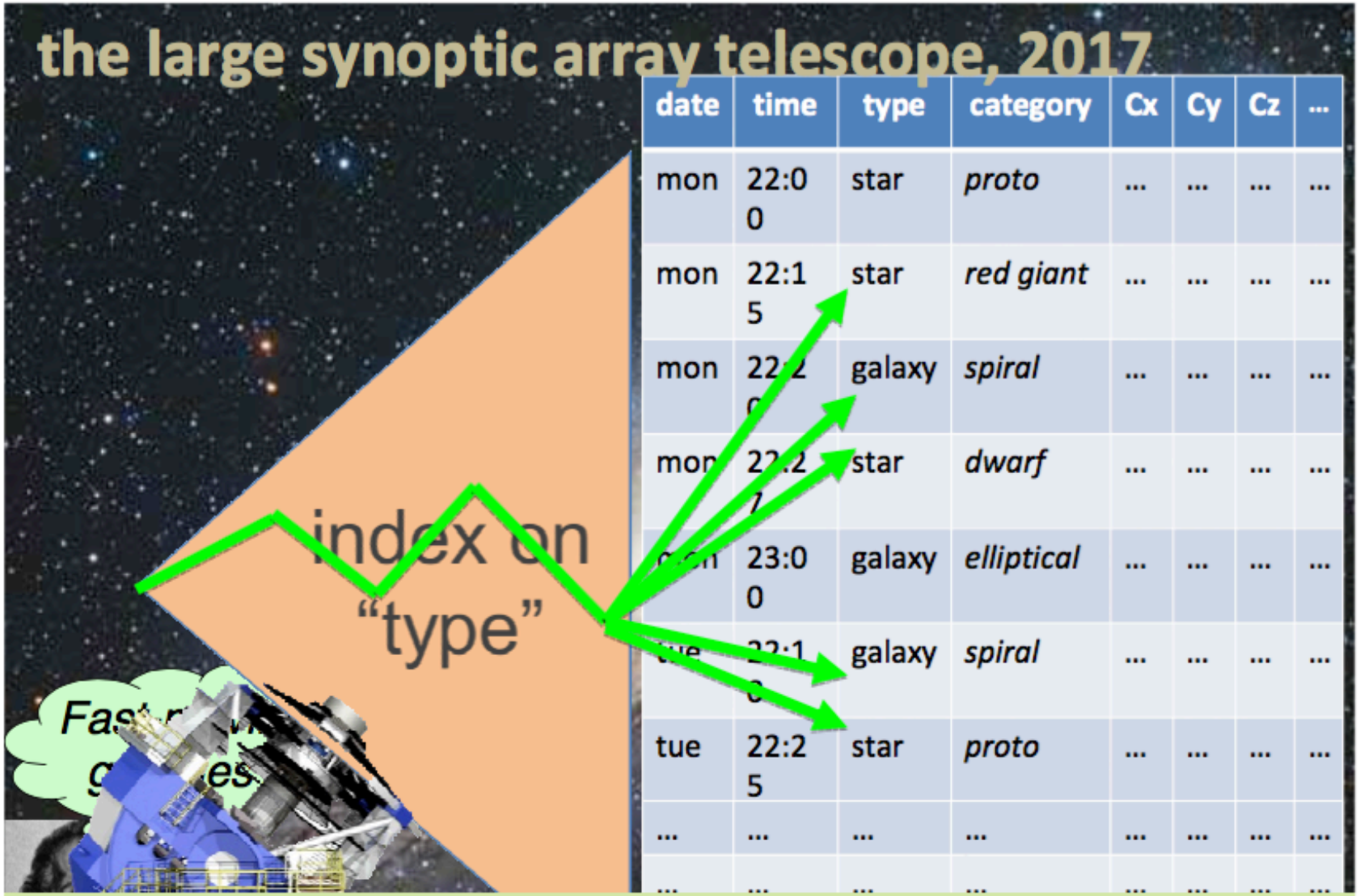
Real-life applications : Pros

- Existing problems and challenges

Real-life applications : Cons

- Proprietary datasets and workloads

- Next example : obtaining scientific data
  - broader vision on scientific data exchange



date	time	type	category	Cx	Cy	Cz	…
mon	22:00	star	*proto*	…	…	…	…
mon	22:15	star	*red giant*	…	…	…	…
mon	22:2 0	galaxy	*spiral*	…	…	…	…
mon	22:2 1	star	*dwarf*	…	…	…	…
mon	23:00	galaxy	*elliptical*	…	…	…	…
tue	22:1 0	galaxy	*spiral*	…	…	…	…
tue	22:25	star	*proto*	…	…	…	…
…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…

100 billion objects, 20 PB/night

Meeting domain scientists

(from A. Ailamaki)

1. *"Hello, we're SO HAPPY to meet you. We have SO MUCH data! PLEASE come visit !"*
2. Visit lab, pretty pictures (we have SO MUCH data)
3. *"Let's have lunch!"* (we have SO MUCH data)
4. Revisit lab, receive promise to get some data
5. Ask for data, no reply
6. Play DBA (design/normalize schema,design data, write queries, rewrite queries, talk to tech staff)
7. Ask for data, receive 2 GB
8. Repeat (6), then ask again for data, receive 4GB

Reasons why collaboration (with scientists) can be hard
(from A. Ailamaki)

- Data is their achievement
  - They do not understand what we will do with it
  - They are afraid of what we may do with it
    - They may think that we will put it on the internet
  - They are not sure how we will help them
  - Do not recognize their problems in our demos

CHOOSING HARDWARE AND SOFTWARE

Choosing the hardware

- Choice mainly depends on your problem, knowledge, background, resources, taste, etc.

- What ever is required by / adequate for your problem
  - A laptop might not be the most suitable, representative database server...

Choosing the Software

- Operative system, DBMS, …
- Commercial
  - Require license
  - “Free” versions with limited capabilities?
  - Limitations on publishing results
  - No access to code
  - Optimizers
  - Analysis & Tuning Tools
- Open source
  - Freely available
  - No limitations on publishing results
  - Access to source code

Proprietary Software

*"Yes, I'll be happy to let you have a time-limited free license of **X** for [research] purposes. (is 12 months adequate?)*

*I only make one condition, which is that you give me the opportunity to comment on any factual statements about **X** before publication.*

*I ask for that because I've sometimes seen researchers come to incorrect conclusions about how **X** is behaving, that could easily have been avoided if they had asked me a few questions."*

Choosing the software

- Other choices depend on your problem, knowledge, background, taste, etc.

- Operating system
- Programming language
- Compiler
- Scripting languages
- System tools
- Visualization tools

CHOOSING THE METRICS

Metrics

"Standards of measurement by which efficiency, performance, progress, or quality of a plan, process, or product can be assessed"

Database Metrics

- Performance to an end-user
  - means response time

- Performance from a systems viewpoint
  - means throughput
  - and capability to handle a given load

Database Metrics

- **Response time** : The time to get an answer to an individual query

- **Throughput** : The number of queries that can be run in any time period.
  - e.g. Queries per second

Metrics : What to measure ?

- Evaluation time
  - wall-clock (“real”) CPU (“user”) I/O (“system”)
  - Server-side vs. client-side
- Memory (and|or) storage usage & requirements
- Scale-up (size) / Speed-up (time)
- Analysis
  - System events & interrupts
  - Hardware events

Unix Time

- **User** is the amount of CPU time spent in user-mode code (outside the kernel) within the process.
- **Sys** is the amount of CPU time spent in the kernel within the process.
- Both are actual CPU time used in executing the process.
  - When a program loops through an array, it is accumulating user CPU time.
  - Conversely, when a program executes a system call such as exec or fork, it is accumulating system CPU time

Unix Time

- **Real** time, is time from start to finish of the call.
  - This is all elapsed time including time slices used by other processes and time the process spends blocked (for example if it is waiting for I/O to complete).

- The total CPU time (user time + sys time) may be *more* or *less* than real time.
  - A program may spend some time waiting.
  - On a multicore system, the user and/or sys time (as well as their sum) can actually exceed the real time.

Printing in system output

- Laptop: 1.5 GHz Pentium M (Dothan), 2 MB L2 cache, 2 GB RAM, 5400 RPM disk
- Benchmark : TPC-H (sf = 1)
- MonetDB/SQL v5.5.0/2.23.0
- measured last of three consecutive runs

Q	server user	server real	client real	client real	result size	... time (milliseconds)
1	2830	3533	3534	3575	1.3 KB	
16	550	618	707	1468		

Metrics : Beware of what you measure

- Laptop: 1.5 GHz Pentium M (Dothan), 2 MB L2 cache, 2 GB RAM, 5400 RPM disk
- Benchmark : TPC-H (sf = 1)
- MonetDB/SQL v5.5.0/2.23.0
- measured last of three consecutive runs

	server		client			... time (milliseconds)
	user	real	real	real	result	
Q	file	file	file	terminal	size	output went to ...
1	2830	3533	3534	3575	1.3 KB	
16	550	618	707	1468	1.2 MB	

Metrics : Beware of what you measure

1/11/2016

“Le programme a mis 13 minutes pour arriver à […]

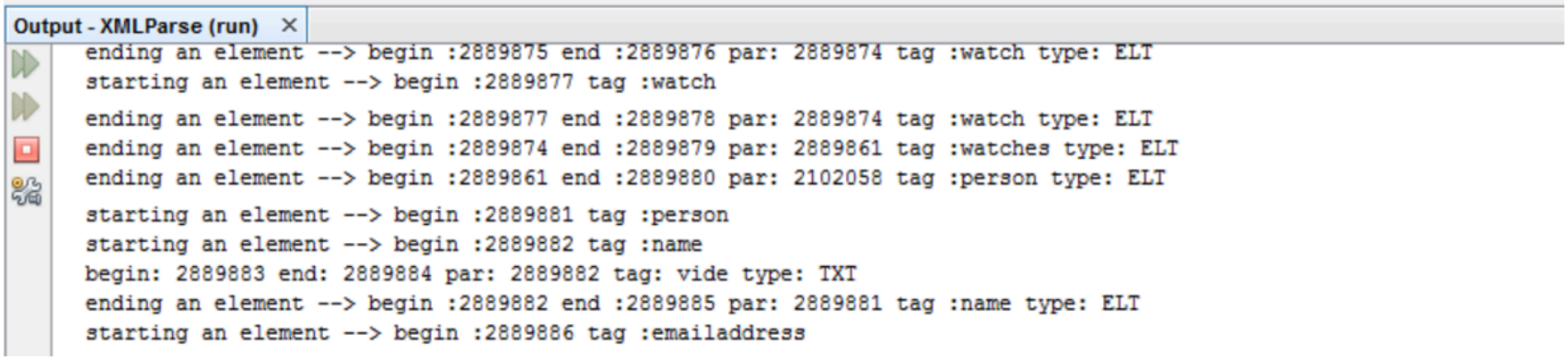
malheureusement je ne peux pas laisser le programme en marche car mon laptop a des problèmes de ventilation et ne supportera pas le processus pour une longue durée.

*Le programme mettra environ **23** heures.”*

Metrics : Beware of what you measure

03/11/2015

“Mon pc est entrain de parser le document […] depuis le soir du 01/11/2015, […] suis-je sur le bon chemin ?”



Time without printing : in the order of seconds

Statistical Estimators

Round 1

22 ms
10 ms
16 ms
16 ms
15 ms
13 ms
13 ms
14 ms
13 ms
15 ms
10 ms
16 ms
10 ms
553 ms
10 ms
5 ms
8 ms

Statistical Estimators

Round 1

22 ms
10 ms
16 ms
16 ms
15 ms
13 ms
13 ms
14 ms
13 ms
15 ms
10 ms
16 ms
10 ms
553 ms
10 ms
5 ms
8 ms

Use Robust Mean

- Take 5 (or 3, or 10) measurements
- Discard lower and higher values
- Average on the remaining values

Result Characterization

- **Arithmetic Mean** (caution, not always the best estimator)

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

AVG{0,10} = 5 = AVG{4,6}

- Add **variance** if result-set has large variability

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n} (x_i - \bar{x})^2$$

Result Characterization

- **Geometric Mean**

$$\left(\prod_{i=1}^{n} a_i\right)^{\frac{1}{n}} = \sqrt[n]{a_1 a_2 \cdots a_n}.$$

- A metric based on an **arithmetic** mean of query execution times may be dominated by the longest running queries.

- To assure that a query does not skew the results, a geometric mean is considered (small values more influent).

WARM VS CLOLD

“We run all experiments in warm memory”

What to choose ? Warm vs Cold

- Depends on what you want to show / measure / analyze

- No formal definition, but “common sense”

Cold Run

- A cold run is a run of the query right after a DBMS is started and no (benchmark-relevant) data is preloaded into the system's main memory, neither by the DBMS, nor in file system caches.

- Such a clean state can be achieved via a system reboot or by running an application that accesses sufficient (benchmark-irrelevant) data to flush file system caches, main memory, and CPU caches.

Warm Run

- A hot run is a run of a query such that as much (query-relevant) data is available as close to the CPU as possible when the measured run starts.

- This can (e.g.) be achieved by running the query (at least) once before the actual measured run starts.

Cold vs Warm

- When a **cold** measure is interesting ?
  - to understand the worst case cost of a process

- When a **warm** measure is interesting ?
  - to understand the average/real cost of a process
    - think of a query asked many times (i.e. keyword search)

Hot vs Warm & User vs. Real Time

- Laptop: 1.5 GHz Pentium M (Dothan), 2 MB L2 cache, 2 GB RAM, 5400 RPM disk
- TPC-H (sf = 1)
  MonetDB/SQL v5.5.0/2.23.0 measured last of three consecutive runs

Q	cold		hot		... time (milliseconds)
	user	real	user	real	
1	2930	13243	2830	3534	

Be aware *what* you measure!

COMPARATIVE ANALYSIS

Apples and Oranges (Comparative Analysis)

- Two colleagues A & B (at CWI) each implemented one version of an algorithm for Monet-DB, A the “old” version and B the improved “new” version
- They ran identical experiments on identical machines, each for his code.
- Though both agreed that B’s new code should be significantly better, results were consistently worse.
- They tested, profiled, analyzed, argued, wondered, fought for several days ...
- ... and eventually found out that A had compiled with C++ optimization enabled, while B had not .

Apples and Oranges

- A) `configure --enable-debug --enable-optimize --enable-assert CFLAGS = "-g [-O0]"`

- B) `configure --disable-debug --disable-optimize --disable-assert CFLAGS = " -O6 -fomit-frame-pointer -finline-functions -malign-loops=4 -malign-jumps=4 -malign-functions=4 -fexpensive-optimizations -funroll-all-loops -funroll-loops -frerun-cse-after-loop -frerun-loop-opt -DNDEBUG "`

Apples and Oranges (Comparative Analysis)

- Compiler optimization $\Rightarrow$ up to factor 2 performance difference
- DBMS configuration and tuning $\Rightarrow$ factor x performance difference ($2 \leq x \leq 10$?)
- Default settings often too “conservative”
- Do you know all systems you use/compare equally well?

Is there any correct formulation ?

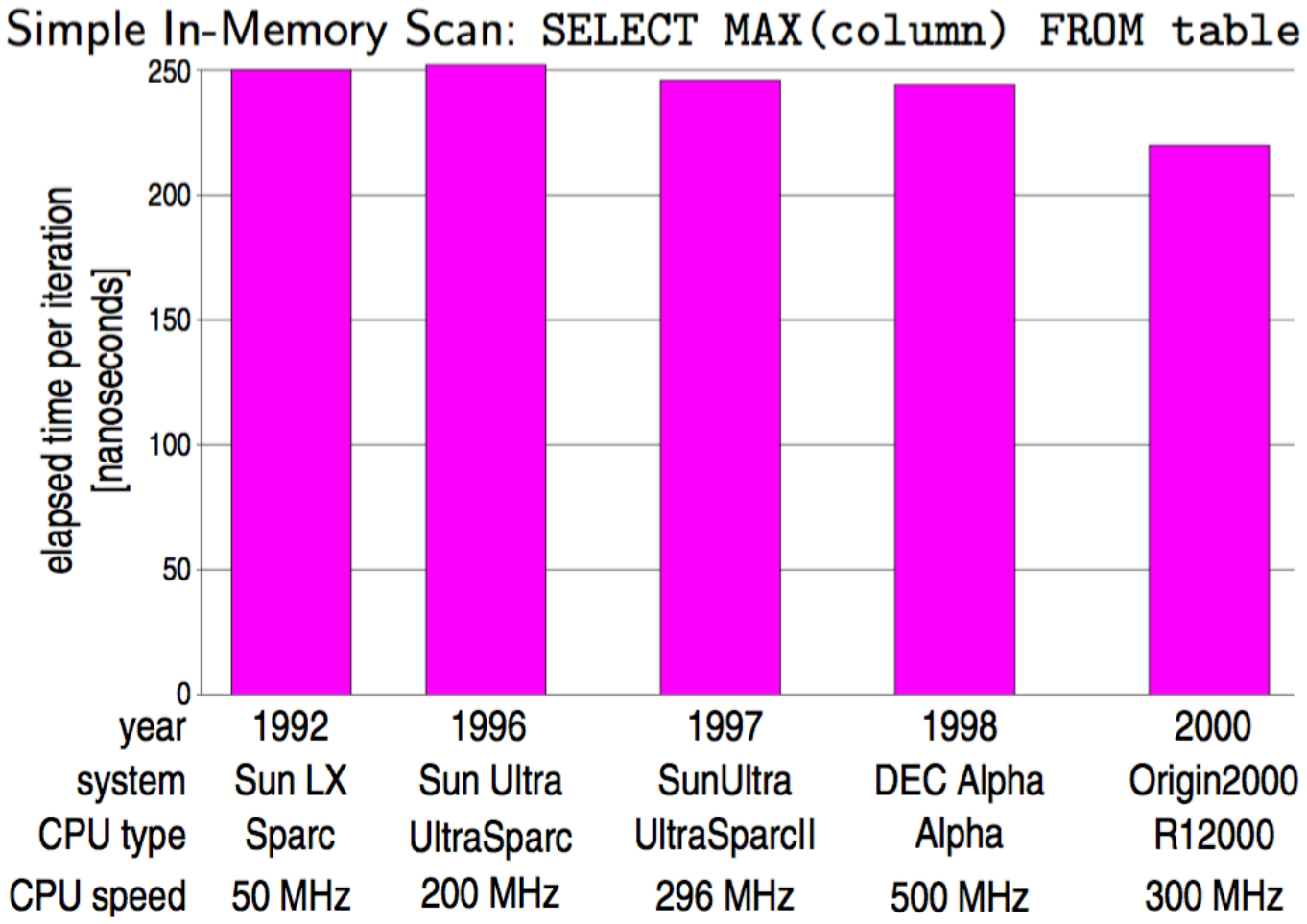
- "Absolutely fair" comparison of complex systems virtually impossible

- But, be at least aware of the the crucial factors and their impact, and document accurately and completely what you do.

*The joke : "Our problem-specific, hand-tuned, prototype **X** outperforms an out-of-the-box installation of a full-fledged off-the-shelf system **Y** , in particular when omitting query parsing, translation, optimization and result printing in **X** , while including them in **Y**"*

ANALYZING THE DATA

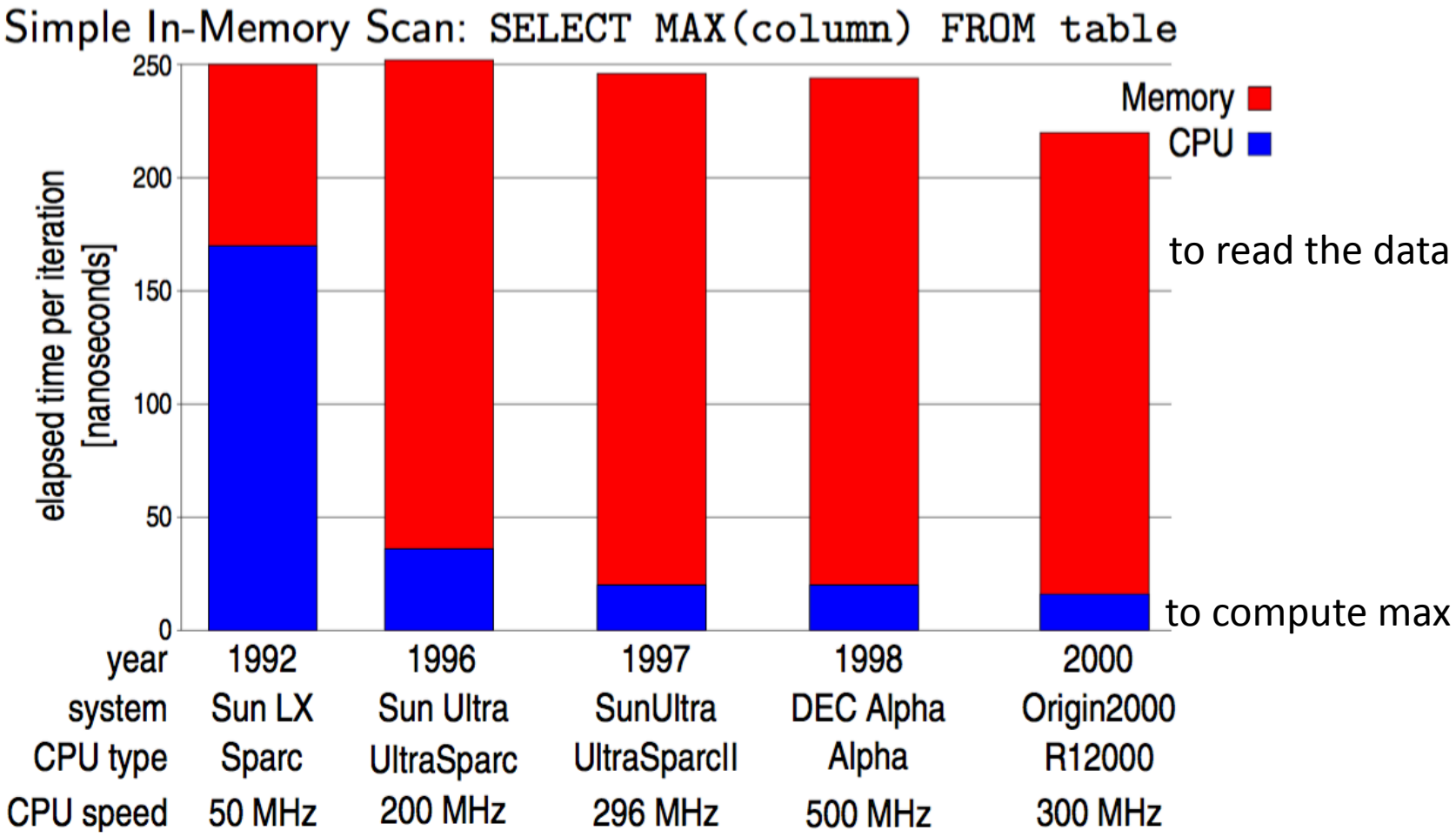
Do you know what happens?



Do you know what happens?

- Simple **In-Memory** Scan (No disk-I/O)
  - SELECT MAX(column) FROM table
- Up to 10x improvement in CPU clock-speed
  ⇒ *Yet hardly any performance improvement !??*
- Standard profiling (e.g., ‘gcc -gp‘ + ‘gprof‘) does not reveal more (in this case)
- Need to dissect CPU & memory access costs
  - Use hardware performance counters to analyze cache-hits, -misses & memory accesses
    - VTune, oprofile, perfctr, perfmon2, PAPI, PCL, etc.

Find out what happens !



Find out what happens !

[Understanding, Modeling, and Improving Main-Memory Database Performance]

- *“[…] Consequently, while our experiment was still largely CPU-bound on the Sun from 1992, it is dominated by* ***memory access costs*** *on the modern machines.*

  This can be attributed to the complex memory subsystem that comes with SMP architectures, resulting in a high memory latency […]”

EXPERIMENT DESIGN

Experiment design

- Main issue with experiments : testing all possible level combinations is unfeasible
- Ignoring important parameter leads to brittle results
- Goal of design : maximize information
  minimize experiments

Experiment design : Terminology

- **Factors**: variables with alternatives that affect a measurement
  - e.g. CPU type, memory size, # disk drives, workload used, dataset used, DB used
- **Levels**: values a factor can assume
  - e.g. memory size levels: 512MB, 1GB, 2GB, 4GB

Goal : Understand Factor Interaction

- Non-Interaction: level of A does not changes effect of level change of B

	Memory 2GB	Memory 8GB
DiskType HDD	10 (s)	8 (s)
DiskType SSD	5 (s)	4 (s)

Goal : Understand Factor Interaction

- Non-Interaction: level of A does not changes effect of level change of B

	Memory 2GB	Memory 8GB
DiskType HDD	10 (s)	8 (s)
DiskType SSD	5 (s)	4 (s)

- Interaction:when level of A changes effect of level change of B

	Memory 2GB	Memory 8GB
DiskType HDD	10 (s)	8 (s)
DiskType SSD	5 (s)	1 (s)

Types of Experimental Designs

- Simple designs
- Full factorial design
- Fractional factorial design

Simple Design (Not Recommended)

- What is a simple design?
  - start with a typical configuration
  - vary one factor at a time; see how performance changes

Simple Design (Not Recommended)

- Example: comparing workstation configurations
  - run typical configuration using a benchmark
  - run experiments to pick the best CPU by varying CPU only
  - using the best CPU, run a set of experiments to find the minimum memory size yielding good performance
  - using the best (CPU, memory) configuration, examine the impact of disk RPM on the benchmark's performance

- Drawbacks
  - if factors interact, may yield wrong conclusions
  - fails to make best use of # experiments

Full Factorial Design (the Dream..)

- Explores every possible combination at all levels of factors
  - exponential number of experiments (thus, a dream)

- Example
  (5 CPU types)(4 memory sizes)(2 disk RPMs)
  (4 workloads) = 160 experiments

- Advantages
  - every possible configuration of workload is examined
- Disadvantages
  - cost: too many experiments

Ways to reduce cost

- Reduce number of levels per factor
  - 2 is very popular

- Reduce # factors:
  - initially only examine a few levels of each factor
  - prune unimportant factors, then try more factors per level
  - use fractional factorial designs

Fractional Designs

Carefully chosen subset (fraction) of the experimental runs of a full factorial design

- full factorial design
  - (2 CPU types)(2 memory sizes)(2 disk RPMs)(2 workloads) = **16 exp.**

- Remove some tests
  - here a $2^{(4-1)}$ design
  - only 8 experiments
    - left ones maybe negligible ?
  - do a smart selection

CPU	Memory	Disk	Workload
CPU	Memory	Disk	Workload
CPU	Memory	Disk	Workload
CPU	Memory	Disk	Workload
CPU	Memory	Disk	Workload
CPU	Memory	Disk	Workload
CPU	Memory	Disk	Workload
CPU	Memory	Disk	Workload

2^k Factorial Designs

- What are they?
  - design to determine effect of k factors, each with 2 levels
- Why consider them?
  - easy to analyze
  - helps order factors based on impact
  - useful to identify factors that have significant impact
    - $\Rightarrow$ study with full factorial design
  - factors have little impact
    - $\Rightarrow$ not of interest for quantitative study

2^k Factorial Designs

- How to select 2 levels
  —factor effect is expected to be unidirectional

  $\Rightarrow$ select min (low), max (high)

- Performance increases or decreases w/ factor
  – e.g. performance improves with more memory

2^2 Factorial Designs

- Special case of 2^k factorial designs, k = 2
  - two factors at two levels

- Utility
  —2^2 designs simple to analyze with regression

Result Interpretation

- Example : impact of memory size and data size on a performance.

	Memory 1GB	Memory 2GB
Data Size 0.1GB	1000 (ms)	300 (ms)
Data Size 1GB	3500 (ms)	3000 (ms)

- Non-linear regression model to interpret result
  - Define variables accounting for low/high levels
  - x_A = -1 if 1GB memory +1 if 2GB memory
  - x_B = -1 if 0.1GB data +1 if 1GB data

Non-linear regression model

A=low B=low

A=high B=high

$$y = q_0 + q_A x_A + q_B x_B + q_{AB} x_A x_B$$

A=high B=low

A=low B=high

Non-linear regression model

A=low B=low

A=high B=high

$$y = q_0 + q_A x_A + q_B x_B + q_{AB} x_A x_B$$

A=high B=low

A=low B=high

- (low,low) mem.=1G, data=0.1G, time=1000 (ms)

$$1000 = q_0 + q_A x_A + q_b x_B + q_{AB} x_A x_B$$

$$1000 = q_0 + q_A * -1 + q_b * -1 + q_{AB} -1 * -1$$

$$1000 = q_0 - q_A - q_b + q_{AB}$$

Non-linear regression model

$$y = q_0 + q_A x_A + q_B x_B + q_{AB} x_A x_B$$

- (high,low) mem.=2G, data=0.1G, time=300(ms)

$$300 = q_0 + q_A x_A + q_b x_B + q_{AB} x_A x_B$$

$$300 = q_0 + q_A * +1 + q_b * -1 + q_{AB}\ 1 * -1$$

$$300 = q_0 + q_A - q_b - q_{AB}$$

Non-linear regression model

$$y = q_0 + q_A x_A + q_B x_B + q_{AB} x_A x_B$$

- (low,high) mem.=1G, data=1G, time=3500(ms)

$$3500 = q_0 + q_A x_A + q_b x_B + q_{AB} x_A x_B$$

$$3500 = q_0 + q_A * -1 + q_b * +1 + q_{AB}\ 1 * -1$$

$$3500 = q_0 - q_A + q_b - q_{AB}$$

Non-linear regression model

$$y = q_0 + q_A x_A + q_B x_B + q_{AB} x_A x_B$$

- (high,high) mem.=2G, data=1G, time=3000(ms)

$$3000 = q_0 + q_A x_A + q_b x_B + q_{AB} x_A x_B$$

$$3000 = q_0 + q_A * +1 + q_b * +1 + q_{AB}\ 1*1$$

$$3000 = q_0 + q_A + q_b + q_{AB}$$

Non-linear regression model

$$1000 = q_0 - q_A - q_B + q_{AB}$$
$$300 = q_0 + q_A - q_B - q_{AB}$$
$$3500 = q_0 - q_A + q_B - q_{AB}$$
$$3000 = q_0 + q_A + q_B + q_{AB}$$

- 4 equations in 4 variables : unique solution

$$y = \mathbf{1950} - \mathbf{300}x_A + \mathbf{1300}x_B + \mathbf{50}x_Ax_B$$

Non-linear regression model

$$y = \mathbf{1950} - \mathbf{300}x_A + \mathbf{1300}x_B + \mathbf{50}x_Ax_B$$

Interpreted as:

- the mean response time is **1950** ms
- effect of memory improves performances of **300**ms
  - regressive effect because of negative sign
- more data can lower performances of **1300**ms
- the interaction between memory and data accounts for **50**ms
  - the greater the factor (wrt mean) the greater the interaction

Shortcut to compute coefficients

- More generally

$$y_1 = q_0 - q_A - q_B + q_{AB} \qquad y_2 = q_0 + q_A - q_B - q_{AB}$$
$$y_3 = q_0 - q_A + q_B - q_{AB} \qquad y_4 = q_0 + q_A + q_B + q_{AB}$$

- Resolution always leads to

$$\mathbf{q_0} = \tfrac{1}{4}(y_1 + y_2 + y_3 + y_4)$$
$$\mathbf{q_A} = \tfrac{1}{4}(-y_1 + y_2 - y_3 + y_4)$$
$$\mathbf{q_B} = \tfrac{1}{4}(-y_1 - y_2 + y_3 + y_4)$$
$$\mathbf{q_{AB}} = \tfrac{1}{4}(+y_1 - y_2 - y_3 + y_4)$$

(Estimated) Variation due to memory

$$\frac{4(q_A)^2}{4(q_A)^2 + 4(q_B)^2 + 4(q_{AB})^2}$$

~ 5%

(Estimated) Variation due to data

$$\frac{4(q_B)^2}{4(q_A)^2 + 4(q_B)^2 + 4(q_{AB})^2}$$

$$\sim 95\%$$

(Estimated) Variation due to interaction

$$\frac{4(q_{AB})^2}{4(q_A)^2 + 4(q_B)^2 + 4(q_{AB})^2}$$

~ 0.1%

Mean of replicates

Treatment Combination	Replicate		
	I	II	III
A low, B low	28	25	27
A high, B low	36	32	32
A low, B high	18	19	23
A high, B high	31	30	29

Conclusion on experiment design

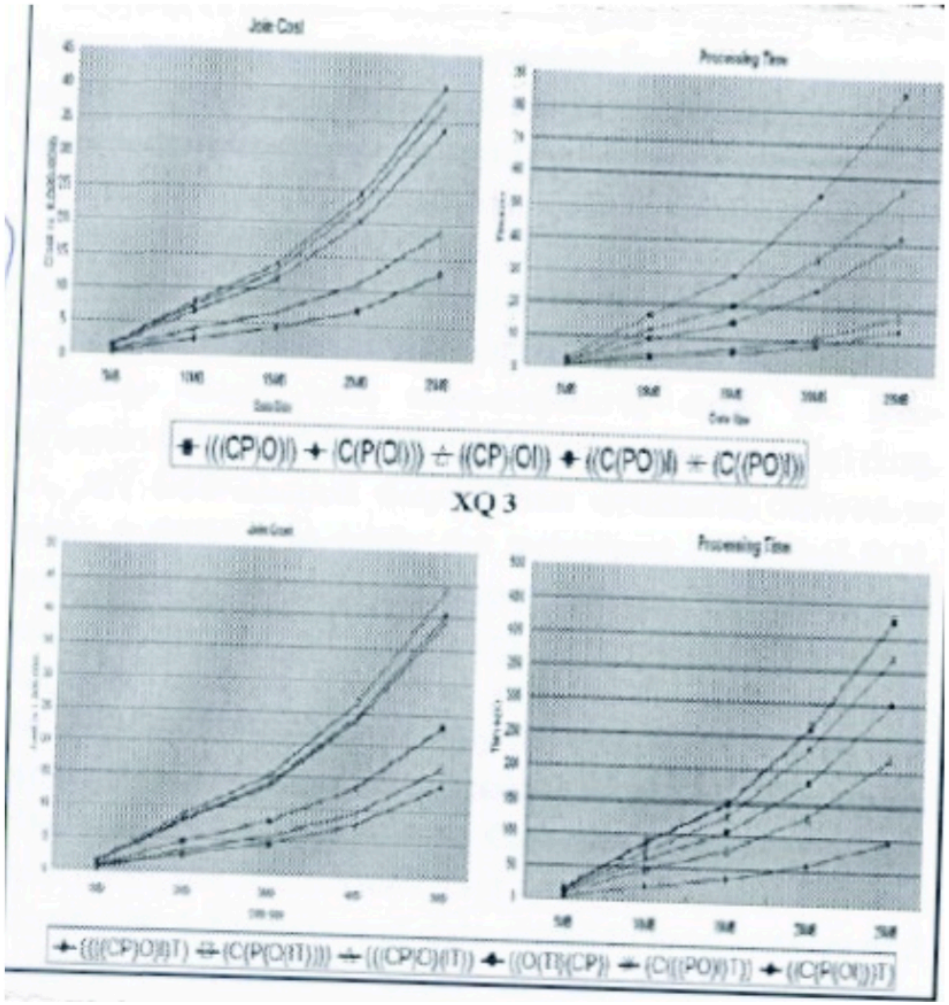
- Picking the factors, their levels, and the replication degree (number of repetitions)

- Design has a huge impact:
  - You don't know what you haven't tested
  - Ignoring important parameter leads to brittle results
  - Testing all possible level combinations is unfeasible

- There exist standard, well-founded procedures for getting the same information with less effort:
  - Run a $\mathbf{2^k}$ (or a $\mathbf{2^{k-p}}$) design (only one level for p factors)
  - Evaluate factor importance
  - Pick important factors and possibly refine levels

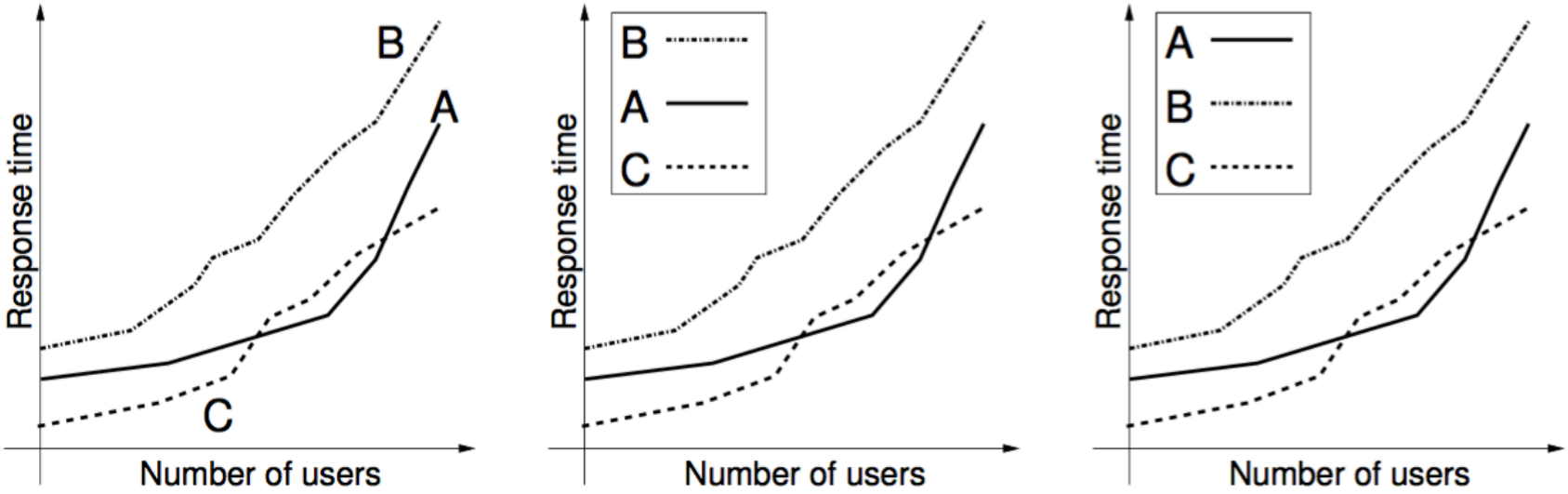
REPORTING ON EXPERIMENTS

A picture worth a thousand words (maybe..)



Guidelines for Graphical Charts

- Require minimum effort from the reader
  - Not the minimum effort from you !!



Guidelines for Graphical Charts

- Maximize information: try to make the graph self-sufficient (no missing descriptions/legend)

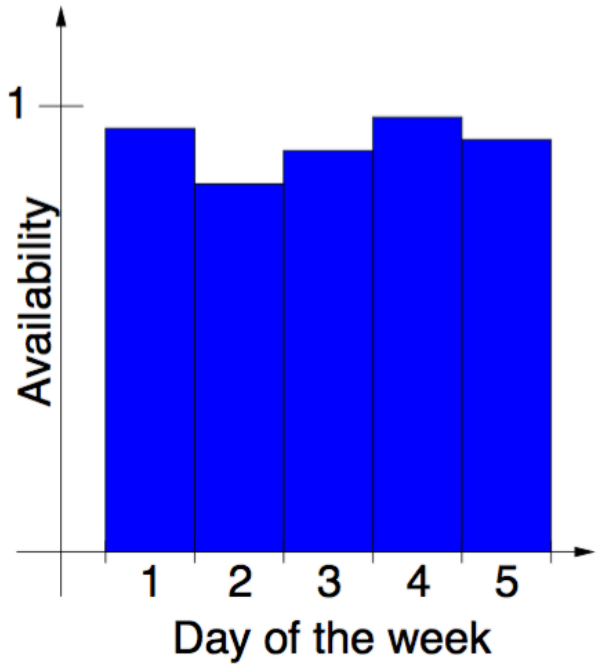
- Use keywords in place of symbols to avoid a join in the reader's brain

- Use informative axis labels:
  prefer "*Average I/Os per query*"
  to "*Average I/Os*" or "*I/Os*"

Guidelines for Graphical Charts

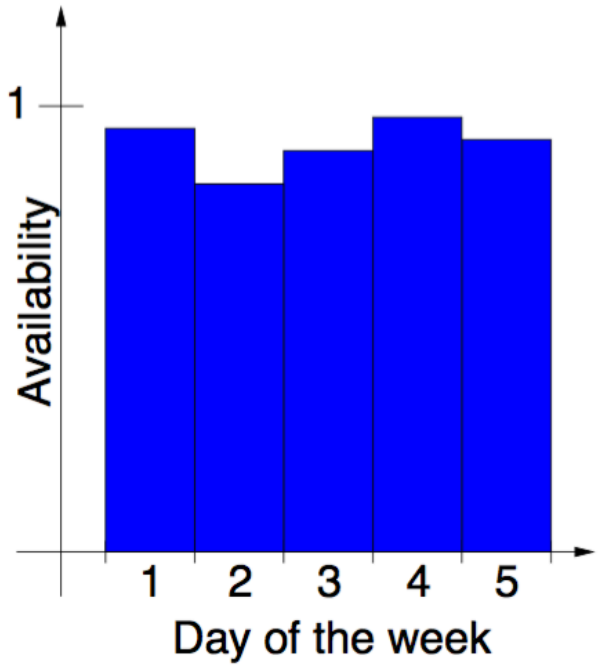
- Include units in the labels: prefer “CPU time (ms)” to “CPU time”
- Usually axes begin at 0, the factor is plotted on x, the result on y
- Usually scales are linear, increase from left to right, divisions are equal
  - Use exceptions as necessary

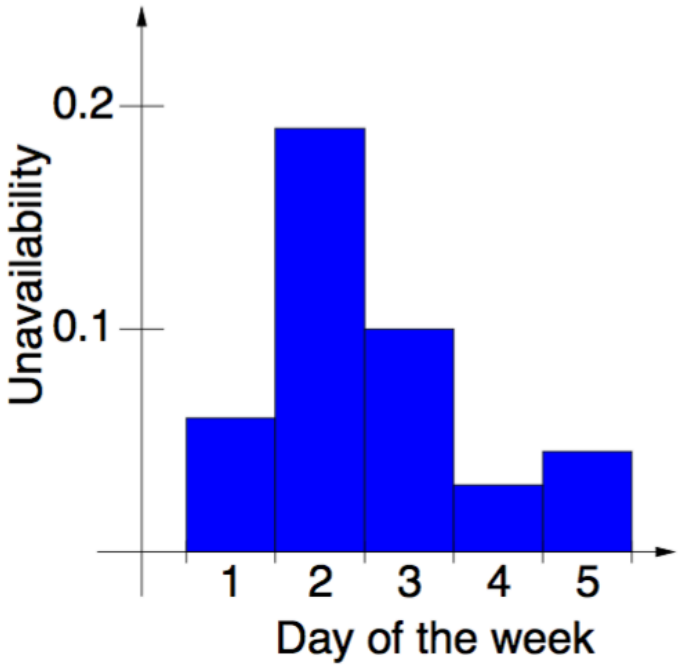
Guidelines for graphical charts



Guidelines for graphical charts

- Prefer the chart that minimize ink while maximizing information





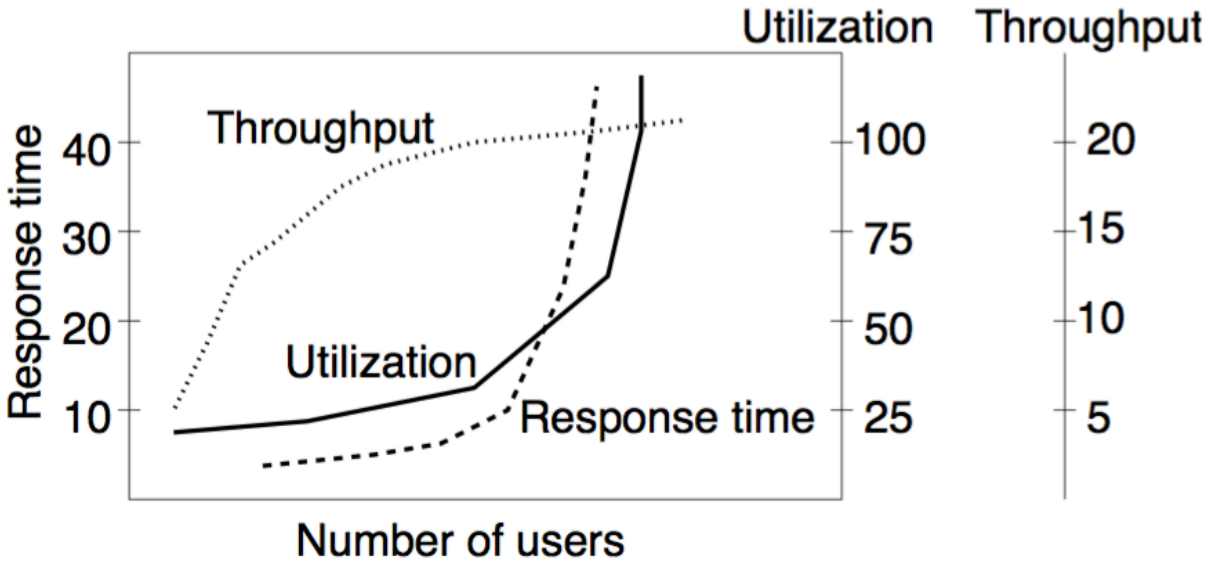
Common presentation mistakes

- Avoid presenting too many alternatives on a single chart

- Rules of thumb, to override with good reason:
  - A line chart should be limited at 6 curves
  - A column chart or bar should be limited to 10 bars
  - A pie chart should be limited to 8 components
  - Each cell in a histogram should have at least five data points

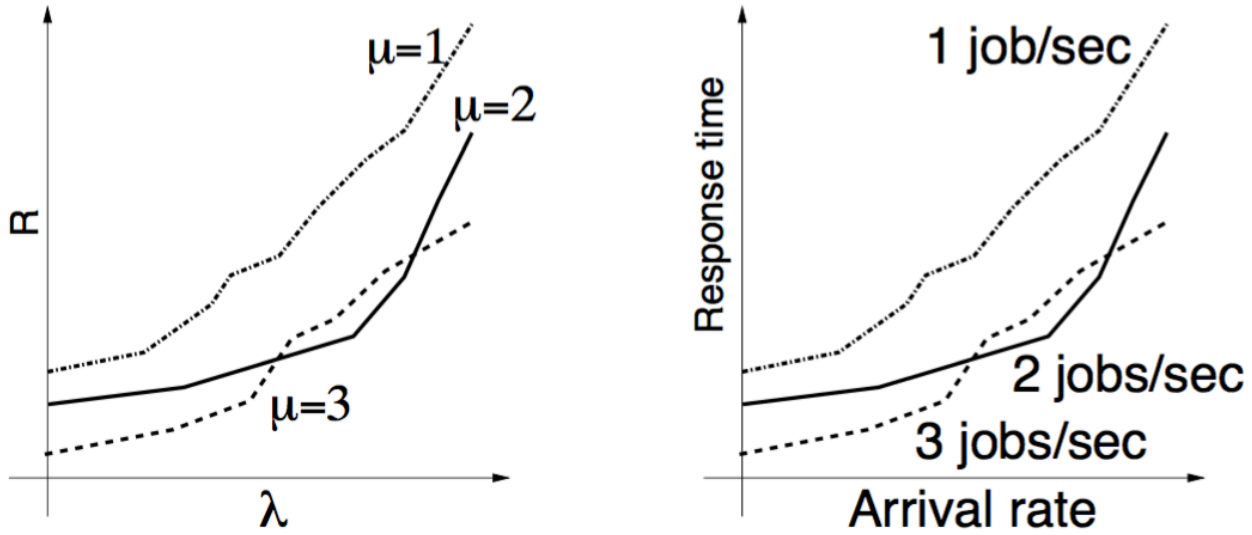
Common presentation mistakes

- Presenting many result variables on a single chart
  - (commonly done to fit into available space)



Common presentation mistakes

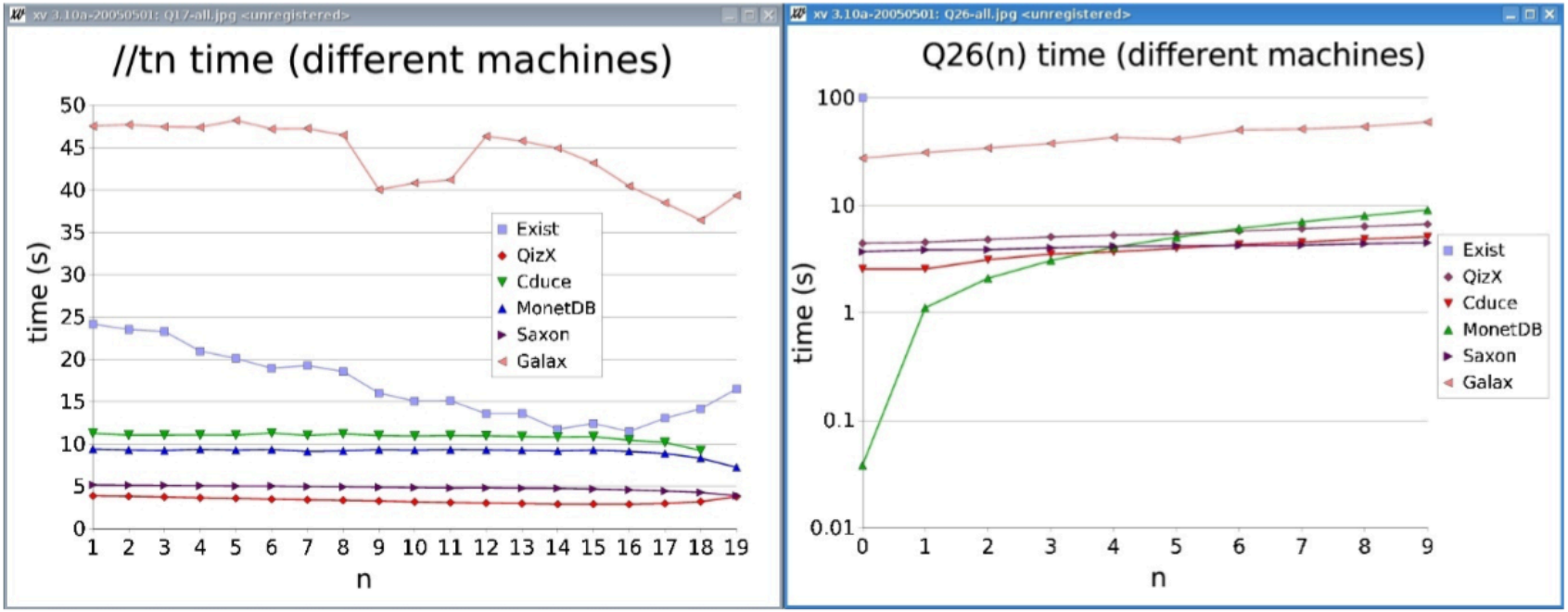
- Using symbols in place of text



- Human brain is a poor join processor
- Humans get frustrated by computing joins

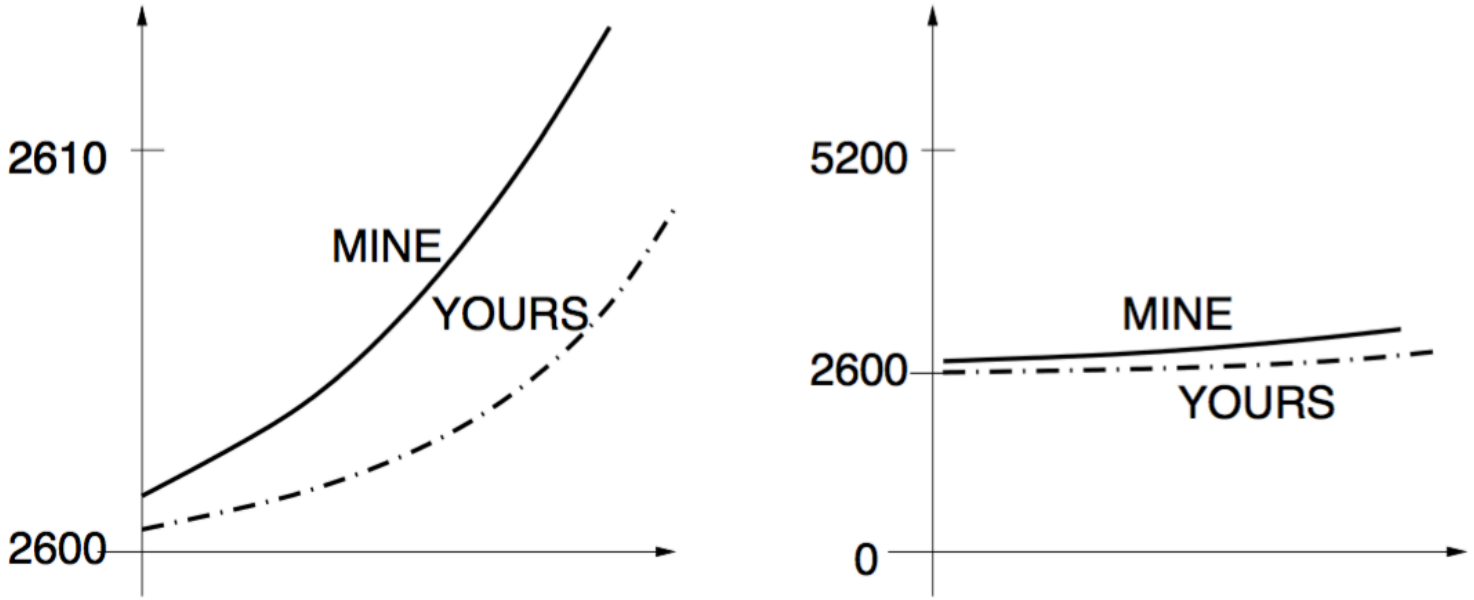
Common presentation mistakes

- Avoid using different scales for similar experiments



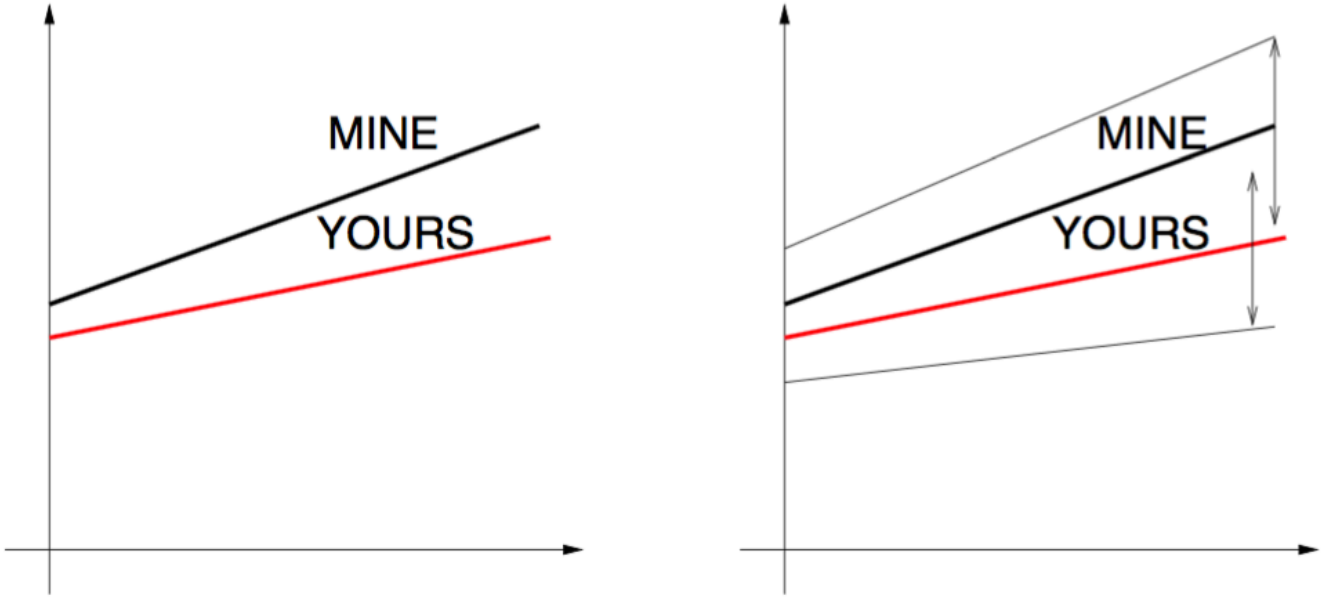
Pictorial games

- MINE is better than YOURS! (Really?)



Pictorial Games

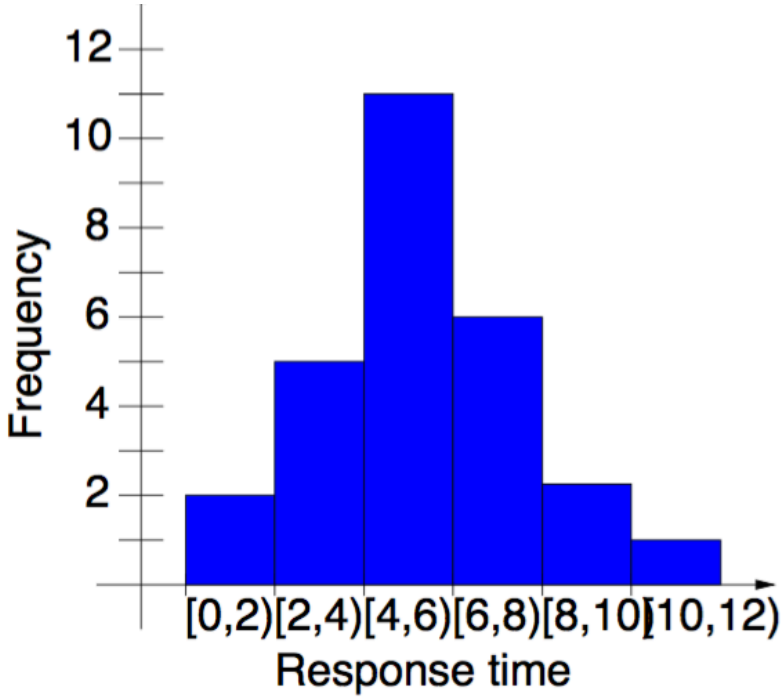
- Be careful plotting quantities w/out confidence intervals

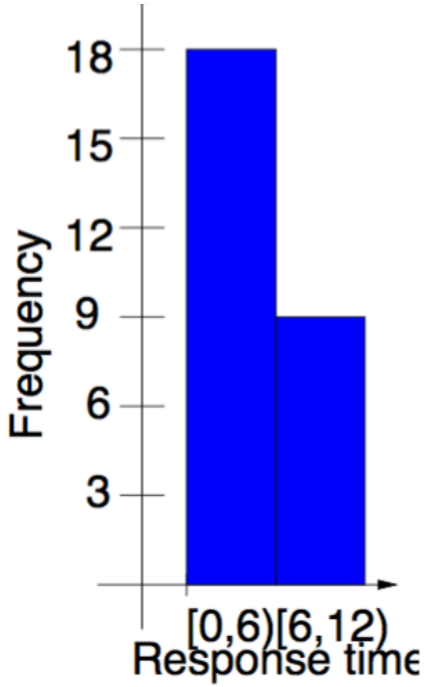


- Overlapping confidence intervals sometimes mean the two quantities are statistically indifferent

Pictorial Games

- Manipulating cell size in histograms





Specifying hardware environments

- We use a machine with 3.4 GHz.”

Underspecified

Specifying hardware environments

cat /proc/cpuinfo

```
processor vendor_id
cpu family model
model name stepping
cpu MHz
cache size fdiv_bug hlt_bug f00f_bug coma_bug
fpu fpu_exception cpuid level wp
flags
bogomips clflush size
: 0
: GenuineIntel
: 6
: 13
: Intel(R) Pentium(R) M processor 1.50GHz : 6
: 600.000
: 2048 KB
: no
: no
: no
: no
: yes
: yes
: 2
```

Overspecified

Specifying hardware environments

- CPU: Vendor, model, generation, clockspeed, cache size(s):
  - 1.5 GHz Pentium M (Dothan), 32 KB L1 cache, 2 MB L2 cache
- Main memory: size
  - 2 GB RAM
- Disk (system): size & speed
  - 120 GB Laptop ATA disk @ 5400 RPM
    1 TB striped RAID-0 system (5x 200 GB S-ATA disk @ 7200 RPM

MAKING EXPERIMENTS REPEATABLE

Making experiments repeatable

- Purpose: another human equipped with the appropriate software and hardware can repeat your experiments.
  - Your supervisor / people you supervise
  - Your colleagues
  - Yourself, 3 months later when you have a new idea
  - Yourself, 3 years later when writing a report or answering requests from people

Making experiments repeatable

- Making experiments portable and parameterizable
- Building a test suite and scripts
- Writing instructions

CONCLUSION

Make experiments

Practical work requires experiments

Experiments should show
- that a system works
- that a system works better than another one

Saying that a solution is more sophisticated and hence better is NOT a valid argument!

In the experiments…

- use standard benchmark datasets if possible
- compare with best solutions in “state of the art”
- run on different data sets (at least 3)
- run different competitors (at least 1, better 3)

In the experiments...

- run with different parameter settings
- explain all datasets, metrics, and settings
- discuss reasons for good and bad performance

An irreproachable experiment is when we use exactly the same dataset, setting and metrics among competitors

Summary

- Good and repeatable performance evaluation and experimental assessment require no fancy magic but rather solid craftmanship
- Proper planning helps to keep you from “getting lost” and ensure repeatability
- Repeatable experiments simplify your own work (and help others to understand it better)

Conclusions

Database performance evaluation

- There is no single way how to do it right.
- There are many ways how to do it wrong.
- Now, it’s your turn ! *n*-★