Performance Evaluation in Database Research: Principles and Experiences

Ioana Manolescu[1] Stefan Manegold[2]

[1]INRIA Saclay–Île-de-France, France, Ioana.Manolescu@inria.fr

[2]CWI Amsterdam, The Netherlands, Stefan.Manegold@cwi.nl

ICDE 2008

Acknowledgments

We are grateful to:

- EXPDB 2006 PC and participants
- VLDB 2007 Performance Evaluation participants and audience
- Dennis Shasha, SIGMOD 2008 PC chair
- SIGMOD 2008 authors of 298 (out of 436) papers who provided code for repeatability testing
- SIGMOD 2008 Repeatability Assessment committee
- Many members of the community for wide-ranging opinions and suggestions
- The ICDE 2008 organizers, in particular *Malu Castellanos*, Mike Carey & Qiong Luo, for providing the opportunity to present this seminar at ICDE 2008.

Performance evaluation

Disclaimer

- There is no single way how to do it right.
- There are many ways how to do it wrong.
- This is not a “mandatory” script.
- This is more a collection of anecdotes or fairy tales — not always to be taken literally, only, but all provide some general rules or guidelines *what (not) to do*.

1. Planning & conducting experiments
2. Design
3. Presentation
4. Repeatability
5. In their words
6. Summary

1. Planning & conducting experiments
   - From micro-benchmarks to real-life applications
   - Choosing the hardware
   - Choosing the software
   - What and how to measure
   - How to run
   - Comparison with others
   - CSI
2. Design
3. Presentation
4. Repeatability
5. In their words
6. Summary

Planning & conducting experiments

What do you plan to do / analyze / test / prove / show?

Planning & conducting experiments

What do you plan to do / analyze / test / prove / show?

- 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?
- CSI: How to find out what is going on?

Data sets & workloads

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

Data sets & workloads

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

- No general simple rules, which to use when
- But some guidelines for the choice...

Micro-benchmarks

Definition

- Specialized, stand-alone piece of software
- Isolating one particular piece of a larger system
- E.g., single DB operator (select, join, aggregation, etc.)

Micro-benchmarks

Pros

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

Micro-benchmarks

Cons

- Neglect larger picture
- Neglect contribution of local costs to global/total costs
- Neglect impact of micro-benchmark on real-life applications
- Neglect embedding in context/system at large
- Generalization of result difficult
- Application of insights in full systems / real-life applications not obvious
- Metrics not standardized
- Comparison?

Standard benchmarks

Examples

- 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 benchmarks

Pros

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

Standard benchmarks

Cons

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

Real-life applications

Pros

- There are so many of them
- Existing problems and challenges

Real-life applications

Cons

- There are so many of them
- Proprietary datasets and workloads

Two types of experiments

Analysis: "CSI"

- Investigate (all?) details
- Analyze and understand behavior and characteristics
- Find out where the time goes and why!

Publication

- "Sell your story"
- Describe picture at large
- Highlight (some) important / interesting details
- Compare to others

Choosing the hardware

Choice mainly depends on your problem, knowledge, background, 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

Which DBMS to use?

Commercial

- Require license
- "Free" versions with limited functionality and/or optimization 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

Choosing the software

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

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

Metrics: What to measure?

- Basic
  - Throughput: queries per time
  - Evaluation time
    - wall-clock ("real")
    - CPU ("user")
    - I/O ("system")
    - Server-side vs. client-side
  - Memory and/or storage usage / requirements
- Comparison
  - Scale-up
  - Speed-up
- Analysis
  - System events & interrupts
  - Hardware events

Metrics: What to measure?

- 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

	server		client			
	user	real	real	real		… time (milliseconds)
Q						
1	2830	3533	3534	3575		
16	550	618	707	1468		

Metrics: What to measure?

- 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

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

Metrics: What to measure?

- 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

	server		client			
	user	real	real	real	result	… time (milliseconds)
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: What to measure?

- 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

	server		client			
	user	real	real	real	result	... time (milliseconds)
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	

Be aware *what* you measure!

Metrics: How to measure?

Which tools, functions and/or system calls to use for measuring time?

- Unix: `/usr/bin/time`, shell built-in `time`
  - Command line tool ⇒ works with any executable
  - Reports "real", "user" & "sys" time (*milliseconds*)
  - Measures entire process incl. start-up
  - Note: output format varies!
- Unix: `gettimeofday()`
  - System function ⇒ requires source code
  - Reports timestamp (*microseconds*)
- Windows: `timeGetTime()`
  - System function ⇒ requires source code
  - Reports timestamp (*milliseconds*)
  - Resolution implementation dependent; default can be as low as 10 milliseconds

Metrics: How to measure?

Use timings provided by the tested software (DBMS)

- IBM DB2
  - db2batch
- Microsoft SQLserver
  - GUI and system variables
- PostgreSQL

postgresql.conf

```
log_statement_stats = on
log_min_duration_statement = 0
log_duration = on
```

- MonetDB/XQuery & MonetDB/SQL
  - `mclient -lxquery -t`
  - `mclient -lsql -t`
  - `(PROFILE|TRACE) select ...`

Metrics: How to measure?

```
mclient -lxquery -t -s'1+2'
```

```
3


Trans 11.626 msec
Shred  0.000 msec
Query  6.462 msec
Print  1.934 msec
```

```
mclient -lsql -t PROFILE_select_1.sql
```

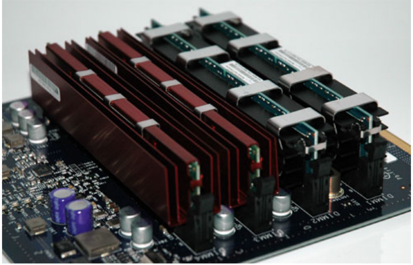
```
% .  # table_name
% single_value # name
% tinyint # type
% 1 # length
[ 1 ]
#times real 62, user 0, system 0, 100
Timer 0.273 msec
```

How to run experiments

"We run all experiments in warm memory."

How to run experiments

"We run all experiments in warm memory."



“hot” 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 filesystem 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 filesystem caches, main memory, and CPU caches.

Hot 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.

- Be aware and document what you do / choose

“hot” vs. “cold”

- 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

	cold		hot		
Q					time (milliseconds)
1	2930		2830		

“hot” vs. “cold”

- 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

	cold		hot		
Q	user		user		… time (milliseconds)
1	2930		2830		

“hot” vs. “cold” & 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

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

“hot” vs. “cold” & 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

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

Be aware *what* you measure!

Of apples and oranges

Once upon a time at CWI ...

- Two colleagues A & B each implemented one version of an algorithm, 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.

Of apples and oranges

Once upon a time at CWI ...

- Two colleagues A & B each implemented one version of an algorithm, 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 ...

Of apples and oranges

Once upon a time at CWI ...

- Two colleagues A & B each implemented one version of an algorithm, 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 optimization enabled, while B had not ...

Of apples and oranges

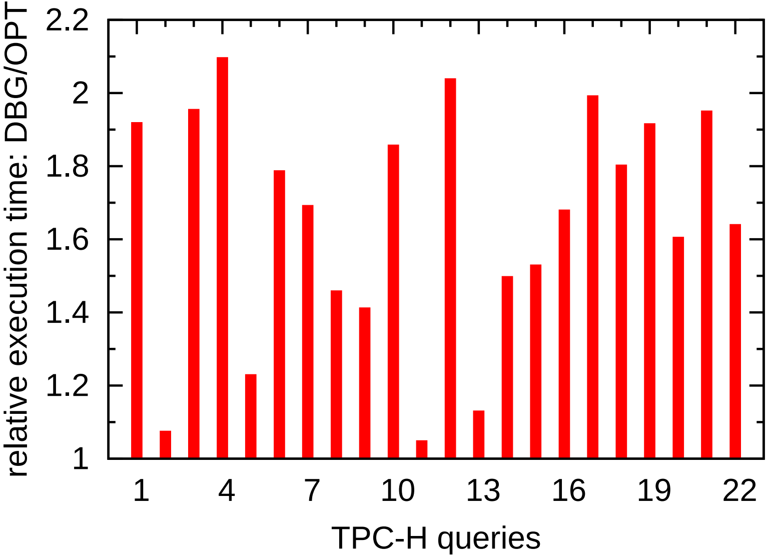
DBG

```
configure --enable-debug --disable-optimize --enable-assert

CFLAGS = "-g [-O0]"
```

OPT

```
configure --disable-debug --enable-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
"
```

Of apples and oranges



Of apples and oranges

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

Of apples and oranges

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

Our problem-specific, hand-tuned, prototype X outperforms an out-of-the-box installation of a full-fledged off-the-shelf system Y,

Of apples and oranges

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

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.

Of apples and oranges

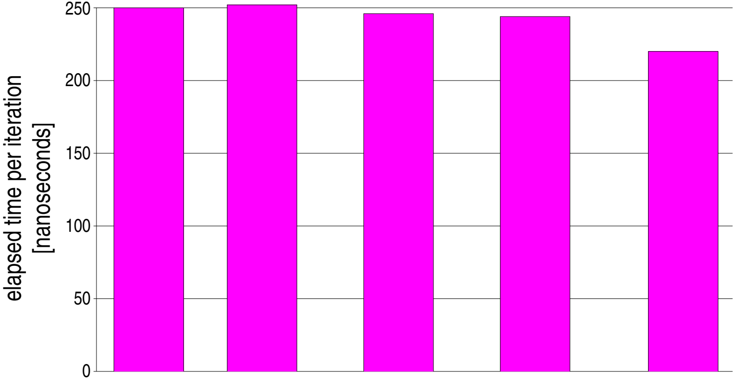
- Compiler optimization ⇒ up to factor 2 performance difference
- DBMS configuration and tuning ⇒ factor x performance difference ($2 \leq x \leq 10$?)
  - "Self-*" still research
  - Default settings often too "conservative"
  - Do you know all systems you use/compare equally well?

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.

- "Absolutely fair" comparisons virtually impossible
- But:
  Be at least aware of the the crucial factors and their impact, and document accurately and completely what you do.

Do you know what happens?

Simple In-Memory Scan: `SELECT MAX(column) FROM table`



year	1992	1996	1997	1998	2000
system	Sun LX	Sun Ultra	SunUltra	DEC Alpha	Origin2000
CPU type	Sparc	UltraSparc	UltraSparcII	Alpha	R12000
CPU speed	50 MHz	200 MHz	296 MHz	500 MHz	300 MHz

Do you know what happens?

Simple In-Memory Scan: `SELECT MAX(column) FROM table`

- No disk-I/O involved
- Up to 10x improvement in CPU clock-speed

⇒ Yet hardly any performance improvement!??

Do you know what happens?

Simple In-Memory Scan: `SELECT MAX(column) FROM table`

- No disk-I/O involved
- Up to 10x improvement in CPU clock-speed

⇒ Yet hardly any performance improvement!??

- Research: Always question what you see!

Do you know what happens?

Simple In-Memory Scan: `SELECT MAX(column) FROM table`

- No disk-I/O involved
- Up to 10x improvement in CPU clock-speed

⇒ Yet hardly any performance improvement!??

- Research: Always question what you see!
- Standard profiling (e.g., '`gcc -gp`' + '`gprof`') does not reveal more (in this case)

Do you know what happens?

Simple In-Memory Scan: `SELECT MAX(column) FROM table`

- No disk-I/O involved
- Up to 10x improvement in CPU clock-speed

⇒ Yet hardly any performance improvement!??

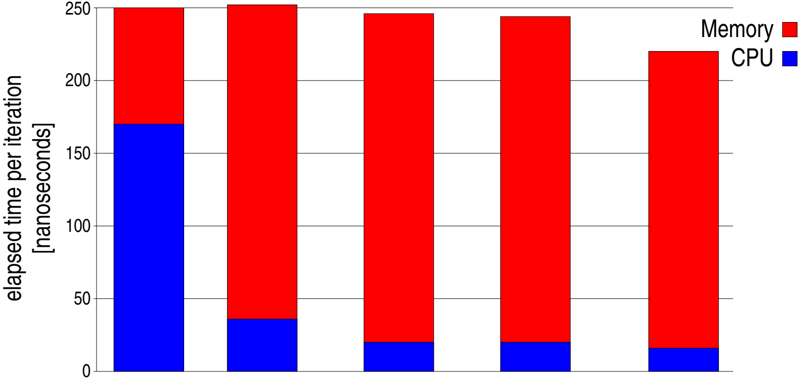
- Research: Always question what you see!

- 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!

Simple In-Memory Scan: `SELECT MAX(column) FROM table`



year	1992	1996	1997	1998	2000
system	Sun LX	Sun Ultra	SunUltra	DEC Alpha	Origin2000
CPU type	Sparc	UltraSparc	UltraSparcII	Alpha	R12000
CPU speed	50 MHz	200 MHz	296 MHz	500 MHz	300 MHz

Find out what happens!

Use info provided by the tested software (DBMS)

- IBM DB2
  - `db2expln`
- Microsoft SQLserver
  - GUI and system variables
- MySQL, PostgreSQL
  - `EXPLAIN select ...`
- MonetDB/SQL
  - `(EXPLAIN|TRACE) select ...`

Find out what happens!

Use profiling and monitoring tools

- ‘`gcc -gp`‘ + ‘`gprof`‘
    - Reports call tree, time per function and time per line
    - Requires re-compilation and static linking
- ‘`valgrind --tool=callgrind`‘ + ‘`kcachegrind`‘
    - Reports call tree, times, instructions executed and cache misses
    - Thread-aware
    - Does not require (re-)compilation
    - Simulation-based ⇒ slows down execution up to a factor 100
- Hardware performance counters
    - to analyze cache-hits, -misses & memory accesses
    - VTune, oprofile, perfctr, perfmon2, PAPI, PCL, etc.
- System monitors
    - `ps`, `top`, `iostat`, ...

Find out what happens!

TPC-H Q1 ($sf = 1$) (AMD AthlonMP @ 1533 GHz, 1 GB RAM)

cum.	**excl.**	**calls**	**ins.**	IPC	function
11.9	11.9	846M	6	0.64	ut_fold_ulint_pair
20.4	8.5	0.15M	27K	0.71	ut_fold_binary
26.2	5.8	77M	37	0.85	memcpy
29.3	**3.1**	**23M**	**64**	**0.88**	**Item_sum_sum::update_field**
32.3	3.0	6M	247	0.83	row_search_for_mysql
35.2	**2.9**	**17M**	**79**	**0.70**	**Item_sum_avg::update_field**
37.8	2.6	108M	11	0.60	rec_get_bit_field_1
40.3	2.5	6M	213	0.61	row_sel_store_mysql_rec
42.7	2.4	48M	25	0.52	rec_get_nth_field
45.1	2.4	60	19M	0.69	ha_print_info
47.5	2.4	5.9M	195	1.08	end_update
49.6	2.1	11M	89	0.98	field_conv
51.6	2.0	5.9M	16	0.77	Field_float::val_real
53.4	1.8	5.9M	14	1.07	Item_field::val
54.9	1.5	42M	17	0.51	row_sel_field_store_in_mysql..
56.3	1.4	36M	18	0.76	buf_frame_align
57.6	**1.3**	**17M**	**38**	**0.80**	**Item_func_mul::val**
59.0	1.4	25M	25	0.62	pthread_mutex_unlock
60.2	1.2	206M	2	0.75	hash_get_nth_cell
61.4	1.2	25M	21	0.65	mutex_test_and_set
62.4	1.0	102M	4	0.62	rec_get_1byte_offs_flag
63.4	1.0	53M	9	0.58	rec_1_get_field_start_offs
64.3	0.9	42M	11	0.65	rec_get_nth_field_extern_bit
65.3	**1.0**	**11M**	**38**	**0.80**	**Item_func_minus::val**
65.8	**0.5**	**5.9M**	**38**	**0.80**	**Item_func_plus::val**

MySQL gprof trace

SF=1		SF=0.001		tot	res	(BW = MB/s)
ms	BW	us	BW	MB	size	**MIL statement**
127	352	150	305	45	5.9M	s0 := select(l_shipdate).mark
134	505	113	608	68	5.9M	s1 := join(s0,l_returnflag)
134	506	113	608	68	5.9M	s2 := join(s0,l_linestatus)
235	483	129	887	114	5.9M	s3 := join(s0,l_extprice)
233	488	130	881	114	5.9M	s4 := join(s0,l_discount)
232	489	127	901	114	5.9M	s5 := join(s0,l_tax)
134	507	104	660	68	5.9M	s6 := join(s0,l_quantity)
290	155	324	141	45	5.9M	s7 := group(s1)
329	136	368	124	45	5.9M	s8 := group(s7,s2)
0	0	0	0	0	4	s9 := unique(s8.mirror)
206	440	60	1527	91	5.9M	r0 := [+](1.0,s5)
210	432	51	1796	91	5.9M	r1 := [-](1.0,s4)
274	498	83	1655	137	5.9M	r2 := [*](s3,r1)
274	499	84	1653	137	5.9M	r3 := [*](s12,r0)
165	271	121	378	45	4	r4 := {sum}(r3,s8,s9)
165	271	125	366	45	4	r5 := {sum}(r2,s8,s9)
163	275	128	357	45	4	r6 := {sum}(s3,s8,s9)
163	275	128	357	45	4	r7 := {sum}(s4,s8,s9)
144	151	107	214	22	4	r8 := {sum}(s6,s8,s9)
112	196	145	157	22	4	r9 := {count}(s7,s8,s9)
3724		2327		**TOTAL**		

MonetDB/MIL trace

1. Planning & conducting experiments
2. Design
   - Preliminaries
   - Full factorial designs
   - Fractional factorial designs
   - Conclusion
3. Presentation
4. Repeatability
5. In their words
6. Summary

Experiment design

The purpose

Design measurement and simulation experiments to provide *the most information with the least effort*

Scenario:

- 5 parameters, each has between 10 and 40 values
- What to do?
  1. Ignore 4 parameters (!)
  2. Perform 10^5 experiments
  3. Anything better?...

Content from Raj Jain, *The Art of Computer Systems Performance Analysis*, Wiley, 1991

Experiment design terminology

Response measure result

Factor any variable that affects the response variable: parameter to be set, or environment (outer) variable

Levels of a factor: possible values

Effect change in the response variable due to factor level change

Replication how many times the experiment was performed

Interaction two factors interact if the effect of one depends on the level of another

Design choice of experiments, factor level combinations and replication for each experiment

Factor interaction

Assume two factors, A and B, with levels $\{A_1, A_2\}$ resp. $\{B_1, B_2\}$.

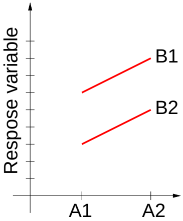
(a)	A_1	A_2
B_1	3	5
B_2	6	8

Same effect of A change regardless of B

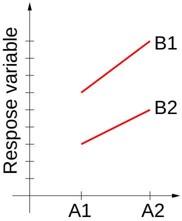
(b)	A_1	A_2
B_1	3	5
B_2	6	9

Different effect of A change depending on B

No interaction



Interaction



Common mistakes

1. Variation due to experimental error is ignored: the variation due to a factor must be compared to that due of errors!
2. Important parameters are not controlled
3. Effects of different factors are not isolated (varying many factors simultaneously)
4. Simple one-at-a-time experiment design: equally meaningful results can be obtained with less (to be seen)
5. Interactions are ignored
6. Too many experiments are conducted (enormous design). Recommended: two-stage approach
    - First experiments help identify meaningful factors and levels
    - Then conduct detailed experiments

Classical designs: Simple design

Assume k factors, such that the i-th factor has n_i levels.
Fix a common configuration and vary one factor at a time.
This requires $n = 1 + \sum_{i=1}^{k}(n_i - 1)$ experiments.

Classical designs: Simple design

Assume k factors, such that the i-th factor has n_i levels.
Fix a common configuration and vary one factor at a time.
This requires $n = 1 + \sum_{i=1}^{k}(n_i - 1)$ experiments.

f_1	f_2	f_3	$\dots$	f_k	r
c_1^1	c_2^1	c_3^1	$\dots$	c_k^1	r_0
c_1^2	c_2^1	c_3^1	$\dots$	c_k^1	r_1
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
$c_1^{n_1}$	c_2^1	c_3^1	$\dots$	c_k^1	r_{n_1}
c_1^1	c_2^2	c_3^1	$\dots$	c_k^1	r_{n_1+1}
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
c_1^1	$c_2^{n_2}$	c_3^1	$\dots$	c_k^1	$r_{n_1+n_2}$
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$

Classical designs: Simple design

Assume k factors, such that the i-th factor has n_i levels.
Fix a common configuration and vary one factor at a time.
This requires $n = 1 + \sum_{i=1}^{k}(n_i - 1)$ experiments.

f_1	f_2	f_3	$\dots$	f_k	r
c_1^1	c_2^1	c_3^1	$\dots$	c_k^1	r_0
c_1^2	c_2^1	c_3^1	$\dots$	c_k^1	r_1
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
$c_1^{n_1}$	c_2^1	c_3^1	$\dots$	c_k^1	r_{n_1}
c_1^1	c_2^2	c_3^1	$\dots$	c_k^1	r_{n_1+1}
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
c_1^1	$c_2^{n_2}$	c_3^1	$\dots$	c_k^1	$r_{n_1+n_2}$
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$

Impossible to identify interactions (when one parameter varies, the others are constant)

Classical designs: Full factorial design

Test all possible level combinations. This requires $n = 1 + \prod_{i=1}^{k}(n_i)$ experiments.

Classical designs: Full factorial design

Test all possible level combinations. This requires $n = 1 + \prod_{i=1}^{k}(n_i)$ experiments.

f_1	f_2	f_3	$\dots$	f_k	r
c_1^1	c_2^1	c_3^1	$\dots$	c_k^1	r_0
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
$c_1^{n_1}$	c_2^1	c_3^1	$\dots$	c_k^1	r_{n_1}
c_1^1	c_2^2	c_3^1	$\dots$	c_k^1	r_{n_1+1}
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
c_1^1	$c_2^{n_2}$	c_3^1	$\dots$	c_k^1	$r_{n_1+n_2}$
c_2^1	c_2^2	c_3^1	$\dots$	c_k^1	$r_{n_1+n_2+1}$
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
c_2^1	$c_2^{n_2}$	c_3^1	$\dots$	c_k^1	$r_{n_1+2\times n_2}$
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$

Classical designs: Full factorial design

Test all possible level combinations. This requires $n = 1 + \prod_{i=1}^{k}(n_i)$ experiments.

f_1	f_2	f_3	$\dots$	f_k	r
c_1^1	c_2^1	c_3^1	$\dots$	c_k^1	r_0
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
$c_1^{n_1}$	c_2^1	c_3^1	$\dots$	c_k^1	r_{n_1}
c_1^1	c_2^2	c_3^1	$\dots$	c_k^1	r_{n_1+1}
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
c_1^1	$c_2^{n_2}$	c_3^1	$\dots$	c_k^1	$r_{n_1+n_2}$
c_2^1	c_2^2	c_3^1	$\dots$	c_k^1	$r_{n_1+n_2+1}$
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
c_2^1	$c_2^{n_2}$	c_3^1	$\dots$	c_k^1	$r_{n_1+2\times n_2}$
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$

Too many tests (especially if some factors are independent)

Classical designs: 2^k

Assume k factors, such that the i-th factor has 2 levels. This requires $n = 2^k$ experiments.

f_1	f_2	f_3	$\dots$	f_k	r
c_1^1	c_2^1	c_3^1	$\dots$	c_k^1	r_0
c_1^1	c_2^1	c_3^2	$\dots$	c_k^1	r_1
c_1^1	c_2^2	c_3^1	$\dots$	c_k^1	r_2
c_1^1	c_2^2	c_3^2	$\dots$	c_k^1	r_3
c_1^2	c_2^1	c_3^1	$\dots$	c_k^1	r_4
c_1^2	c_2^1	c_3^2	$\dots$	c_k^1	r_5
c_1^2	c_2^2	c_3^1	$\dots$	c_k^1	r_6
c_1^2	c_2^2	c_3^2	$\dots$	c_k^1	r_7
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
c_2^2	c_2^2	c_3^2	$\dots$	c_k^2	r_{2^k}

Very useful for a first-cut analysis!

Classical designs: fractional factorial designs

Smart selection of level combinations

Experiment Number	CPU	Memory Level	Workload Type	Educational Level
1	6800	512 K	Managerial	High school
2	6800	2 M	Scientific	Postgraduate
3	6800	8 M	Secretarial	College
1	Z80	512 K	Scientific	College
2	Z80	2 M	Secretarial	High school
3	Z80	8 M	Managerial	Postgraduate
1	8086	512 K	Secretarial	Postgraduate
2	8086	2 M	Managerial	College
3	8086	8 M	Scientific	High school

Classical designs: fractional factorial designs

Smart selection of level combinations

Experiment Number	CPU	Memory Level	Workload Type	Educational Level
1	6800	512 K	Managerial	High school
2	6800	2 M	Scientific	Postgraduate
3	6800	8 M	Secretarial	College
1	Z80	512 K	Scientific	College
2	Z80	2 M	Secretarial	High school
3	Z80	8 M	Managerial	Postgraduate
1	8086	512 K	Secretarial	Postgraduate
2	8086	2 M	Managerial	College
3	8086	8 M	Scientific	High school

Less experiments

Classical designs: fractional factorial designs

Smart selection of level combinations

Experiment Number	CPU	Memory Level	Workload Type	Educational Level
1	6800	512 K	Managerial	High school
2	6800	2 M	Scientific	Postgraduate
3	6800	8 M	Secretarial	College
1	Z80	512 K	Scientific	College
2	Z80	2 M	Secretarial	High school
3	Z80	8 M	Managerial	Postgraduate
1	8086	512 K	Secretarial	Postgraduate
2	8086	2 M	Managerial	College
3	8086	8 M	Scientific	High school

Less experiments

Some information loss (interactions!) Maybe they were negligible?

2^2 design

Example: impact of memory size and cache size on a workstation performance.

Performance in MIPS

Cache size	Memory size 4MB	Memory size 16 MB
1 KB	15	45
2 KB	25	75

Define the following variables:

$$x_A = \begin{array}{ll} \text{-1} & \text{if 4MB memory} \\ 1 & \text{if 16MB memory} \end{array} ; \; x_B = \begin{array}{ll} \text{-1} & \text{if 1KB cache} \\ 1 & \text{if 2 KB cache} \end{array}$$

2^2 design

Example: impact of memory size and cache size on a workstation performance.

Performance in MIPS

Cache size	Memory size 4MB	Memory size 16 MB
1 KB	15	45
2 KB	25	75

Define the following variables:

$$x_A = \begin{cases} -1 & \text{if 4MB memory} \\ 1 & \text{if 16MB memory} \end{cases} ; \quad x_B = \begin{cases} -1 & \text{if 1KB cache} \\ 1 & \text{if 2 KB cache} \end{cases}$$

Nonlinear regression model

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

2^2 factorial design

Nonlinear regression model

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

$$\begin{aligned} 15 &= q_0 - q_A - q_B + q_{AB} \\ 45 &= q_0 + q_A - q_B - q_{AB} \\ 25 &= q_0 - q_A + q_B - q_{AB} \\ 75 &= q_0 + q_A + q_B + q_{AB} \end{aligned}$$

Solving this leads to:

$$y = 40 + 20x_A + 10x_B + 5x_A x_B$$

interpreted as: the mean is 40; the effect of memory is 20 MIPS; the effect of cache is 10 MIPS; the interaction between memory and cache accounts for 5 MIPS.

Analysis of 2^2 factorial design

More generally:

$$\begin{aligned}
y_1 &= q_0 - q_A - q_B + q_{AB} \\
y_2 &= q_0 + q_A - q_B - q_{AB} \\
y_3 &= q_0 - q_A + q_B - q_{AB} \\
y_4 &= q_0 + q_A + q_B + q_{AB}
\end{aligned}$$

Resolution leads to:

$$\begin{aligned}
q_0 &= \frac{1}{4}(y1 + y2 + y3 + y4) \\
q_A &= \frac{1}{4}(-y1 + y2 - y3 + y4) \\
q_B &= \frac{1}{4}(-y1 - y2 + y3 + y4) \\
q_{AB} &= \frac{1}{4}(y1 - y2 - y3 + y4)
\end{aligned}$$

Analysis of 2^2 factorial design

Experiment	A	B	AB	y
1	-1	-1	1	y_1
2	1	-1	-1	y_2
3	-1	1	-1	y_3
4	1	1	1	y_4

$$q_0 = \frac{1}{4}(y1 + y2 + y3 + y4)$$

$$q_A = \frac{1}{4}(-y1 + y2 - y3 + y4)$$

$$q_B = \frac{1}{4}(-y1 - y2 + y3 + y4)$$

$$q_{AB} = \frac{1}{4}(y1 - y2 - y3 + y4)$$

Analysis of 2^2 factorial design

Experiment	A	B	AB	y
1	-1	-1	1	y_1
2	1	-1	-1	y_2
3	-1	1	-1	y_3
4	1	1	1	y_4

$$q_0 = \frac{1}{4}(y1 + y2 + y3 + y4)$$

$$q_A = \frac{1}{4}(-y1 + y2 - y3 + y4)$$

$$q_B = \frac{1}{4}(-y1 - y2 + y3 + y4)$$

$$q_{AB} = \frac{1}{4}(y1 - y2 - y3 + y4)$$

Analysis of 2^2 factorial design

Experiment	A	B	AB	y
1	-1	-1	1	y_1
2	1	-1	-1	y_2
3	-1	1	-1	y_3
4	1	1	1	y_4

$$q_0 = \frac{1}{4}(y1 + y2 + y3 + y4)$$

$$q_A = \frac{1}{4}(-y1 + y2 - y3 + y4)$$

$$q_B = \frac{1}{4}(-y1 - y2 + y3 + y4)$$

$$q_{AB} = \frac{1}{4}(y1 - y2 - y3 + y4)$$

Analysis of 2^2 factorial design

Experiment	A	B	AB	y
1	-1	-1	1	y_1
2	1	-1	-1	y_2
3	-1	1	-1	y_3
4	1	1	1	y_4

$$q_0 = \frac{1}{4}(y1 + y2 + y3 + y4)$$

$$q_A = \frac{1}{4}(-y1 + y2 - y3 + y4)$$

$$q_B = \frac{1}{4}(-y1 - y2 + y3 + y4)$$

$$q_{AB} = \frac{1}{4}((y1 - y2 - y3 + y4))$$

Sign table method of calculating effects

Simple algorithm to obtain q_0, q_A, q_B, q_{AB} based on the sign matrix:

Sign table method of calculating effects

Simple algorithm to obtain q_0, q_A, q_B, q_{AB} based on the sign matrix:

I	A	B	AB	y
1	-1	-1	1	y_1
1	1	-1	-1	y_2
1	-1	1	-1	y_3
1	1	1	1	y_4

$\Rightarrow$

$$\begin{aligned} q_0 &= \tfrac{1}{4}(I \cdot y) \\ q_A &= \tfrac{1}{4}(A \cdot y) \\ q_B &= \tfrac{1}{4}(B \cdot y) \\ q_{AB} &= \tfrac{1}{4}(AB \cdot y) \end{aligned}$$

Sign table method of calculating effects

Simple algorithm to obtain q_0, q_A, q_B, q_{AB} based on the sign matrix:

I	A	B	AB	y
1	-1	-1	1	y_1
1	1	-1	-1	y_2
1	-1	1	-1	y_3
1	1	1	1	y_4

$\Rightarrow$

$$\begin{aligned} q_0 &= \tfrac{1}{4}(I \cdot y) \\ q_A &= \tfrac{1}{4}(A \cdot y) \\ q_B &= \tfrac{1}{4}(B \cdot y) \\ q_{AB} &= \tfrac{1}{4}(AB \cdot y) \end{aligned}$$

The mean response is $\bar{y} = q_0$.

Allocation of variation: understanding factor impact

1. The total variation of y or sum of squares total is:

$$SST = \sum_{i=1}^{2^2} (y_i - \bar{y})^2$$

Allocation of variation: understanding factor impact

1. The total variation of y or sum of squares total is:

$$SST = \sum_{i=1}^{2^2} (y_i - \bar{y})^2$$

2. Distribute SST among the factors. For a 2^2 design:

$$SST = 2^2 q_A^2 + 2^2 q_B^2 + 2^2 q_{AB}^2$$

Allocation of variation: understanding factor impact

1. The total variation of y or sum of squares total is:

$$SST = \sum_{i=1}^{2^2}(y_i - \bar{y})^2$$

2. Distribute SST among the factors. For a 2^2 design:

$$SST = 2^2 q_A^2 + 2^2 q_B^2 + 2^2 q_{AB}^2$$

% of variation explained by A: $2^2 q_A^2 / SST \sim$ importance of A

Allocation of variation: understanding factor impact

1. The total variation of y or sum of squares total is:

$$SST = \sum_{i=1}^{2^2}(y_i - \bar{y})^2$$

2. Distribute SST among the factors. For a 2^2 design:

$$SST = 2^2 q_A^2 + 2^2 q_B^2 + 2^2 q_{AB}^2$$

% of variation explained by A: $2^2 q_A^2 / SST \sim$ importance of A

% of variation explained by B: $2^2 q_B^2 / SST \sim$ importance of B

Allocation of variation: understanding factor impact

1. The total variation of y or sum of squares total is:

$$SST = \sum_{i=1}^{2^2} (y_i - \bar{y})^2$$

2. Distribute SST among the factors. For a 2^2 design:

$$SST = 2^2 q_A^2 + 2^2 q_B^2 + 2^2 q_{AB}^2$$

% of variation explained by A: $2^2 q_A^2 / SST \sim$ importance of A

% of variation explained by B: $2^2 q_B^2 / SST \sim$ importance of B

% of variation explained by the interaction of A and B:
$2^2 q_{AB}^2 / SST \sim$ importance of the interaction of A and B

Example: allocation of variation

Memory interconnection networks: {Omega, Crossbar}

Example: allocation of variation

Memory interconnection networks: {Omega, Crossbar}
Two different address reference patterns: {Random, Matrix}

Example: allocation of variation

Memory interconnection networks: {Omega, Crossbar}
Two different address reference patterns: {Random, Matrix}
Response variables: average throughput T; 90% transit time in cycles N; average response time R.

Example: allocation of variation

Memory interconnection networks: {Omega, Crossbar}
Two different address reference patterns: {Random, Matrix}
Response variables: average throughput T; 90% transit time in cycles N; average response time R.

Symbol	Factor	Level -1	Level 1
A	Type of network	Crossbar	Omega
B	Address pattern used	Random	Matrix

Example: allocation of variation

Memory interconnection networks: {Omega, Crossbar}
Two different address reference patterns: {Random, Matrix}
Response variables: average throughput T; 90% transit time in cycles N; average response time R.

Symbol	Factor	Level -1	Level 1
A	Type of network	Crossbar	Omega
B	Address pattern used	Random	Matrix

Example: allocation of variation

Memory interconnection networks: {Omega, Crossbar}
Two different address reference patterns: {Random, Matrix}
Response variables: average throughput T; 90% transit time in cycles N; average response time R.

Symbol	Factor	Level -1	Level 1
A	Type of network	Crossbar	Omega
B	Address pattern used	Random	Matrix

A	B	T	N	R
-1	-1	0.6041	3	1.655
1	-1	0.4220	5	2.378
-1	1	0.7922	2	1.262
1	1	0.4717	4	2.190

⇒

Example: allocation of variation

Memory interconnection networks: {Omega, Crossbar}
Two different address reference patterns: {Random, Matrix}
Response variables: average throughput T; 90% transit time in cycles N; average response time R.

Symbol	Factor	Level -1	Level 1
A	Type of network	Crossbar	Omega
B	Address pattern used	Random	Matrix

A	B	T	N	R
-1	-1	0.6041	3	1.655
1	-1	0.4220	5	2.378
-1	1	0.7922	2	1.262
1	1	0.4717	4	2.190

$\Rightarrow$

Variation explained (%)			
	T	N	R
q_A	17.2	20	10.9
q_B	77.0	80	87.8
q_{AB}	5.8	0	1.3

Example: allocation of variation

Memory interconnection networks: {Omega, Crossbar}
Two different address reference patterns: {Random, Matrix}
Response variables: average throughput T; 90% transit time in cycles N; average response time R.

Symbol	Factor	Level -1	Level 1
A	Type of network	Crossbar	Omega
B	Address pattern used	Random	Matrix

A	B	T	N	R
-1	-1	0.6041	3	1.655
1	-1	0.4220	5	2.378
-1	1	0.7922	2	1.262
1	1	0.4717	4	2.190

⇒

Variation explained (%)			
	T	N	R
q_A	17.2	20	10.9
q_B	77.0	80	87.8
q_{AB}	5.8	0	1.3

Conclusion: the address pattern influences most. The chosen patterns are very different.

From 2^2 to 2^k

k factors, each with 2 levels There will be:

- k main effects
- C_k^2 two-factors interactions
- C_k^3 three-factors interactions
- $\ldots$

$$\begin{aligned} y = & q_0 + q_{A_1}x_{A_1} + q_{A_2}x_{A_2} + \ldots + q_{A_k}x_{A_k} + \\ & q_{A_1A_2}x_{A_1}x_{A_2} + q_{A_1A_3}x_{A_1}x_{A_3} + \ldots + q_{A_{k-1}A_k}x_{A_{k-1}}x_{A_k} + \\ & \ldots + \\ & q_{A_1A_2\ldots A_k}x_{A_1}x_{A_2}\ldots x_{A_k} \end{aligned}$$

2^k experiments allow to compute $q_0, q_{A_1}, \ldots, q_{A_1A_2\ldots A_k}$. Then the analysis proceeds as for 2^2.

Preparing a fractional factorial design

There are k parameters, each with 2 levels. Instead of 2^k, we aim to judiciously choose 2^{k-p} level combinations to test.

Preparing a fractional factorial design

There are k parameters, each with 2 levels. Instead of 2^k, we aim to judiciously choose 2^{k-p} level combinations to test.
The net effect is *simplifying the dependency model*:

Preparing a fractional factorial design

There are k parameters, each with 2 levels. Instead of 2^k, we aim to judiciously choose 2^{k-p} level combinations to test.

The net effect is *simplifying the dependency model*:

$$\begin{aligned} y = & q_0 + q_{A_1} x_{A_1} + q_{A_2} x_{A_2} + \ldots + q_{A_k} x_{A_k} + \\ & q_{A_1 A_2} x_{A_1} x_{A_2} + q_{A_1 A_3} x_{A_1} x_{A_3} + \ldots + q_{A_{k-1} A_k} x_{A_{k-1}} x_{A_k} + \\ & \ldots + q_{A_1 A_2 \ldots A_{k-1}} x_{A_1} x_{A_2} \ldots x_{A_{k-1}} + \\ & q_{A_1 A_2 \ldots A_k} x_{A_1} x_{A_2} \ldots x_{A_k} \end{aligned}$$

becomes:

$$\begin{aligned} y = & q_0 + \qquad q_{A_2} x_{A_2} + \ldots + q_{A_k} x_{A_k} + \\ & q_{A_1 A_2} x_{A_1} x_{A_2} + \qquad \ldots + q_{A_{k-1} A_k} x_{A_{k-1}} x_{A_k} + \\ & \ldots + q_{A_1 A_2 \ldots A_{k-1}} x_{A_1} x_{A_2} \ldots x_{A_{k-1}} \end{aligned}$$

Preparing a fractional factorial design

There are k parameters, each with 2 levels. Instead of 2^k, we aim to judiciously choose 2^{k-p} level combinations to test.

The net effect is *simplifying the dependency model*:

$$\begin{aligned} y = & q_0 + q_{A_1}x_{A_1} + q_{A_2}x_{A_2} + \ldots + q_{A_k}x_{A_k} + \\ & q_{A_1A_2}x_{A_1}x_{A_2} + q_{A_1A_3}x_{A_1}x_{A_3} + \ldots + q_{A_{k-1}A_k}x_{A_{k-1}}x_{A_k} + \\ & \ldots + q_{A_1A_2\ldots A_{k-1}}x_{A_1}x_{A_2}\ldots x_{A_{k-1}} + \\ & q_{A_1A_2\ldots A_k}x_{A_1}x_{A_2}\ldots x_{A_k} \end{aligned}$$

becomes:

$$\begin{aligned} y = & q_0 + \qquad q_{A_2}x_{A_2} + \ldots + q_{A_k}x_{A_k} + \\ & q_{A_1A_2}x_{A_1}x_{A_2} + \qquad \ldots + q_{A_{k-1}A_k}x_{A_{k-1}}x_{A_k} + \\ & \ldots + q_{A_1A_2\ldots A_{k-1}}x_{A_1}x_{A_2}\ldots x_{A_{k-1}} \end{aligned}$$

2^{k-p} measures / equations / coefficients instead of 2^k

Preparing a fractional factorial design

There are k parameters, each with 2 levels. Instead of 2^k, we aim to judiciously choose 2^{k-p} level combinations to test.
The net effect is *simplifying the dependency model*:

$$\begin{aligned} y = & q_0 + q_{A_1}x_{A_1} + q_{A_2}x_{A_2} + \ldots + q_{A_k}x_{A_k} + \\ & q_{A_1A_2}x_{A_1}x_{A_2} + q_{A_1A_3}x_{A_1}x_{A_3} + \ldots + q_{A_{k-1}A_k}x_{A_{k-1}}x_{A_k} + \\ & \ldots + q_{A_1A_2\ldots A_{k-1}}x_{A_1}x_{A_2}\ldots x_{A_{k-1}} + \\ & q_{A_1A_2\ldots A_k}x_{A_1}x_{A_2}\ldots x_{A_k} \end{aligned}$$

becomes:

$$\begin{aligned} y = & q_0 + \qquad\qquad q_{A_2}x_{A_2} + \ldots + q_{A_k}x_{A_k} + \\ & q_{A_1A_2}x_{A_1}x_{A_2} + \qquad\qquad\qquad \ldots + q_{A_{k-1}A_k}x_{A_{k-1}}x_{A_k} + \\ & \ldots + q_{A_1A_2\ldots A_{k-1}}x_{A_1}x_{A_2}\ldots x_{A_{k-1}} \end{aligned}$$

2^{k-p} measures / equations / coefficients instead of 2^k

Ideally, the coefficients replaced with 0 are small

Preparing a fractional factorial design

We need a sign table of dimension 2^{k-p}

- Each column consists of -1 and $+1$ and has the sum zero.
- Columns should be orthogonal.

Preparing a fractional factorial design

We need a sign table of dimension 2^{k-p}

- Each column consists of -1 and +1 and has the sum zero.
- Columns should be orthogonal.

Method:

1. Pick $k - p$ factors, build a full factorial design of size $k - p$.
2. Chose p among the rightmost $2^{k-p} - k + p - 1$ columns and label them with the p factors not chosen in step 1.

Preparing a fractional factorial design of 2^{7-4}

We start with $k = 7$ factors: A, B, C, D, E, F, G.
We pick the first $k - p = 7 - 4 = 3$ factors: A, B, C and build a full factorial design for these:

Exp.	A	B	C	AB	AC	BC	ABC
1	-1	-1	-1	1	1	1	-1
2	1	-1	-1	-1	-1	1	1
3	-1	1	-1	-1	1	-1	1
4	1	1	-1	1	-1	-1	-1
5	-1	-1	1	1	-1	-1	1
6	1	-1	1	-1	1	-1	-1
7	-1	1	1	-1	-1	1	-1
8	1	1	1	1	1	1	1

Preparing a fractional factorial design of 2^{7-4}

We change the names of the rightmost 4 columns into D, E, F, G:

Exp.	A	B	C	D	E	F	G
1	-1	-1	-1	1	1	1	-1
2	1	-1	-1	-1	-1	1	1
3	-1	1	-1	-1	1	-1	1
4	1	1	-1	1	-1	-1	-1
5	-1	-1	1	1	-1	-1	1
6	1	-1	1	-1	1	-1	-1
7	-1	1	1	-1	-1	1	-1
8	1	1	1	1	1	1	1

- 7 zero-sum columns: so that both levels get equally tested
- 3 orthogonal factor columns (A, B and C): any two of these factors agree (product=1) as often as they disagree (product=-1)
- all coefficients of interactions have been erased.

Preparing a fractional factorial design of 2^{4-1}

Exp.	A	B	C	AB	AC	BC	D
1	-1	-1	-1	1	1	1	-1
2	1	-1	-1	-1	-1	1	1
3	-1	1	-1	-1	1	-1	1
4	1	1	-1	1	-1	-1	-1
5	-1	-1	1	1	-1	-1	1
6	1	-1	1	-1	1	-1	-1
7	-1	1	1	-1	-1	1	-1
8	1	1	1	1	1	1	1

Assume 8 experiments lead to results $y_1, y_2, \ldots, y_8$.
The *confounded effects of D and the ABC interaction* is:

$$y \cdot D = y \cdot A \cdot B \cdot C = -y_1 + y + 2 + y_3 - y_4 + y_5 - y_6 - y_7 + y_8.$$

This particular fractional design is denoted D=ABC.

Fractional design D=ABC for 2^{4-1}

If $D = ABC$, then also:

- $A \cdot D = A \cdot ABC = I \cdot BC = BC$; $AD = BC$
- $B \cdot D = B \cdot ABC = I \cdot AC = AC$; $BD = AC$
- $C \cdot D = C \cdot ABC = I \cdot AB = AB$; $AB = CD$
- $BC \cdot D = BC \cdot ABC = I \cdot A$; $A = BCD$
- Also: $B = ACD$, $C = ABD$, $I = ABCD$

This design confounds:

- the mean with the 4th order interaction;
- the main effects with 3rd order interactions.

Fractional design D=ABC for 2^{4-1}

If $D = ABC$, then also:

- $A \cdot D = A \cdot ABC = I \cdot BC = BC$; $AD = BC$
- $B \cdot D = B \cdot ABC = I \cdot AC = AC$; $BD = AC$
- $C \cdot D = C \cdot ABC = I \cdot AB = AB$; $AB = CD$
- $BC \cdot D = BC \cdot ABC = I \cdot A$; $A = BCD$
- Also: $B = ACD$, $C = ABD$, $I = ABCD$

This design confounds:

- the mean with the 4th order interaction;
- the main effects with 3rd order interactions.

The hope is that 3rd and 4th order interactions are small.

Comparison of two 2^{4-1} designs

A	B	C	AB	AC	BC	ABC
-1	-1	-1	1	1	1	-1
1	-1	-1	-1	-1	1	1
-1	1	-1	-1	1	-1	1
1	1	-1	1	-1	-1	-1
-1	-1	1	1	-1	-1	1
1	-1	1	-1	1	-1	-1
-1	1	1	-1	-1	1	-1
1	1	1	1	1	1	1

Fractional design $D = ABC$ $\swarrow$ $\searrow$ Fractional design $D = AB$

A	B	C	AB	AC	BC	D
-1	-1	-1	1	1	1	-1
1	-1	-1	-1	-1	1	1
-1	1	-1	-1	1	-1	1
1	1	-1	1	-1	-1	-1
-1	-1	1	1	-1	-1	1
1	-1	1	-1	1	-1	-1
-1	1	1	-1	-1	1	-1
1	1	1	1	1	1	1

A	B	C	D	AC	BC	ABC
-1	-1	-1	1	1	1	-1
1	-1	-1	-1	-1	1	1
-1	1	-1	-1	1	-1	1
1	1	-1	1	-1	-1	-1
-1	-1	1	1	-1	-1	1
1	-1	1	-1	1	-1	-1
-1	1	1	-1	-1	1	-1
1	1	1	1	1	1	1

Comparison of two 2^{4-1} designs

Confoundings of $D = ABC$:

- $AD = BC$, $BD = AC$, $AB = CD$
- $A = BCD$, $B = ACD$, $C = ABD$
- $I = ABCD$

Confoundings of $D = AB$:

- $A = BD$, $B = AD$, $D = AB$
- $I = ABD$
- $AC = BCD$, $BC = ACD$, $CD = ABC$
- $C = ABCD$

Comparison of two 2^{4-1} designs

Confoundings of $D = ABC$:

- $AD = BC$, $BD = AC$, $AB = CD$
- $A = BCD$, $B = ACD$, $C = ABD$
- $I = ABCD$

Confoundings of $D = AB$:

- $A = BD$, $B = AD$, $D = AB$
- $I = ABD$
- $AC = BCD$, $BC = ACD$, $CD = ABC$
- $C = ABCD$

$D = ABC$ is preferred

It is assumed higher order interactions are less important than lower order interactions (“sparsity of effects” principle). Therefore, designs that confound higher order interactions are preferred.

Conclusion on experiment design

Design

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

Conclusion on experiment design

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

Conclusion on experiment design

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

Conclusion on experiment design

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:
  1. Run a 2^k (or a 2^{k-p}) design
  2. Evaluate factor importance
  3. Pick important factors and possibly refine levels

1. Planning & conducting experiments
2. Design
3. Presentation
    - Guidelines
    - Mistakes
4. Repeatability
5. In their words
6. Summary

Graphical presentation of results

We all know

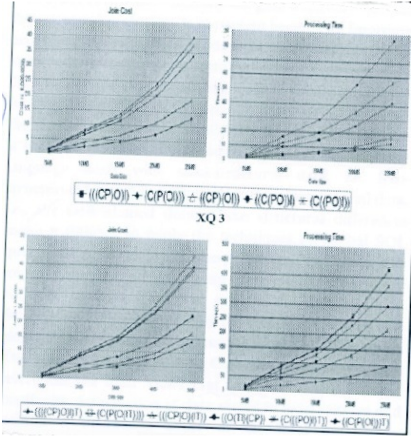
A picture is worth a thousand words

Graphical presentation of results

We all know

A picture is worth a thousand words

Er, maybe not all pictures…

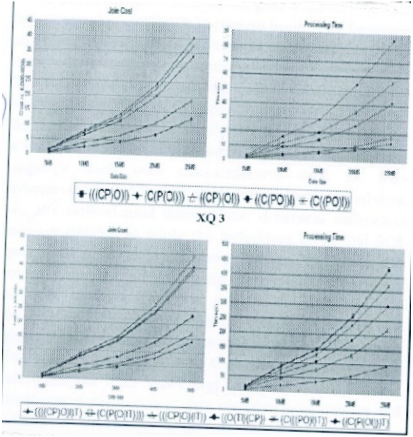


Graphical presentation of results

We all know

A picture is worth a thousand words

Er, maybe not all pictures...



(Borrowed from T.Grust's slides at VLDB 2007 panel)

Guidelines for preparing good graphic charts

Require minimum effort from the reader

Guidelines for preparing good graphic charts

Require minimum effort from the reader

- Not the minimum effort from you

Guidelines for preparing good graphic charts

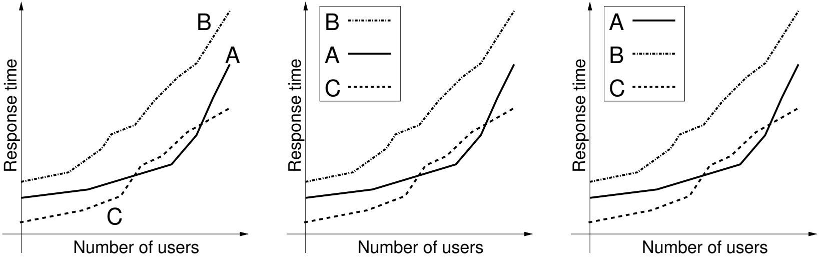
Require minimum effort from the reader

- Not the minimum effort from you
- Try to be honest: how would you like to see it?

Guidelines for preparing good graphic charts

Require minimum effort from the reader

- Not the minimum effort from you
- Try to be honest: how would you like to see it?



Guidelines for preparing good graphic charts

Maximize information: try to make the graph self-sufficient

- 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" to "I/Os"
- Include units in the labels: prefer "CPU time (ms)" to "CPU time"

Guidelines for preparing good graphic charts

Maximize information: try to make the graph self-sufficient

- 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" to "I/Os"
- Include units in the labels: prefer "CPU time (ms)" to "CPU time"

Use commonly accepted practice: present what people expect

- *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 preparing good graphic charts

Minimize ink: present as much information as possible with as little ink as possible

Guidelines for preparing good graphic charts

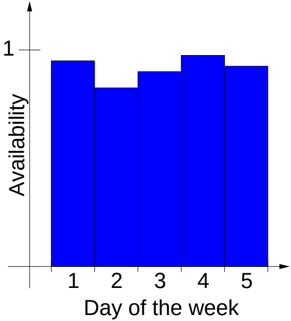
Minimize ink: present as much information as possible with as little ink as possible

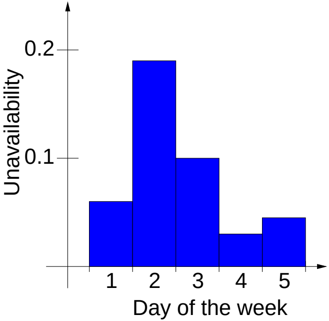
Prefer the chart that gives the most information out of the same data

Guidelines for preparing good graphic charts

Minimize ink: present as much information as possible with as little ink as possible

Prefer the chart that gives the most information out of the same data





Reading material

Edward Tufte: “The Visual Display of Quantitative Information”

`http://www.edwardtufte.com/tufte/books_vdqi`

Common presentation mistakes

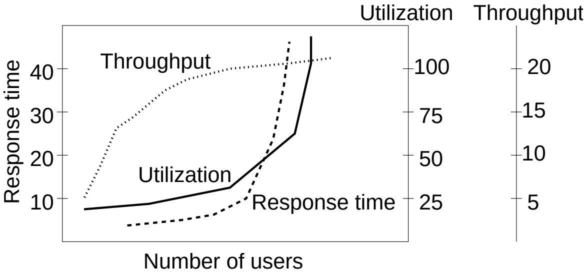
Presenting too many alternatives on a single chart

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

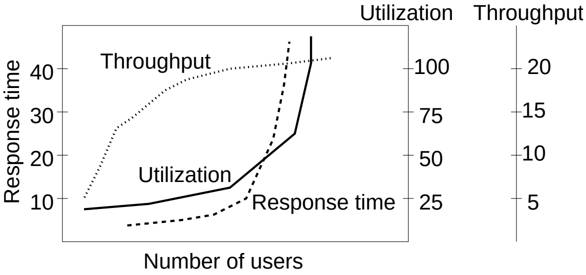
Common presentation mistakes

Presenting many result variables on a single chart
Commonly done to fit into available page count :-(



Common presentation mistakes

Presenting many result variables on a single chart
Commonly done to fit into available page count :-(



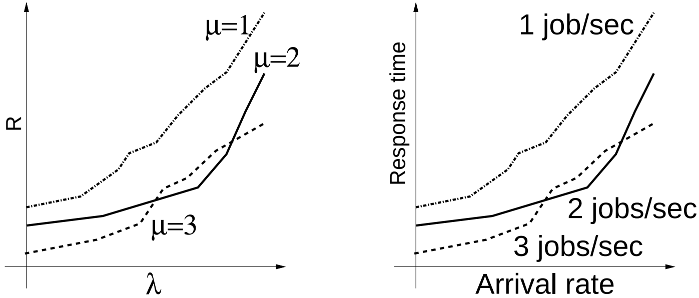
Huh?

Common presentation mistakes

Using symbols in place of text

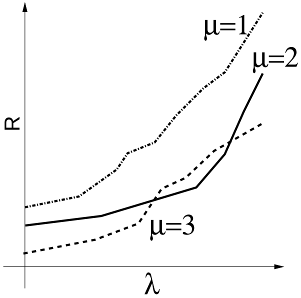
Common presentation mistakes

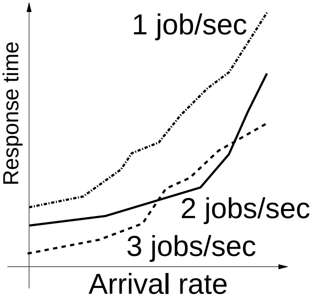
Using symbols in place of text



Common presentation mistakes

Using symbols in place of text

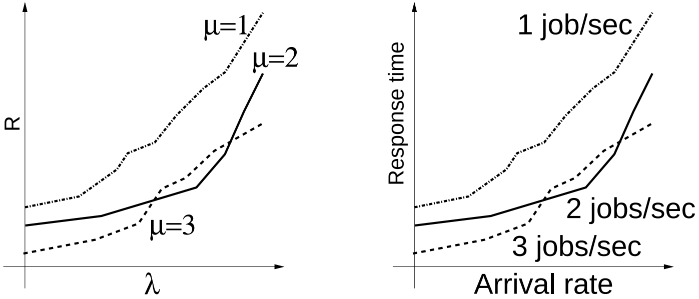




Human brain is a poor join processor

Common presentation mistakes

Using symbols in place of text



Human brain is a poor join processor

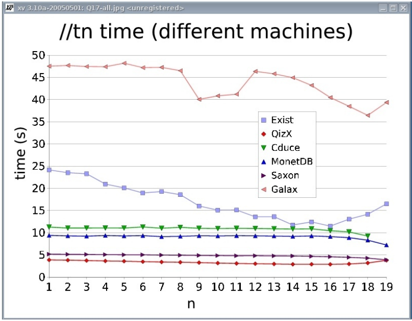
Humans get frustrated by computing joins

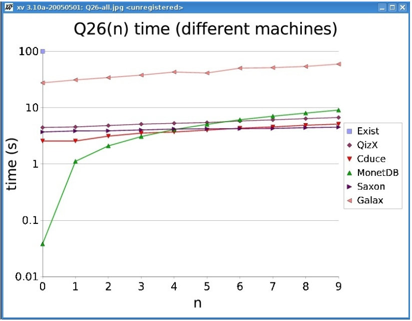
Common presentation mistakes

Change the graphical layout of a given curve from one figure to another

Common presentation mistakes

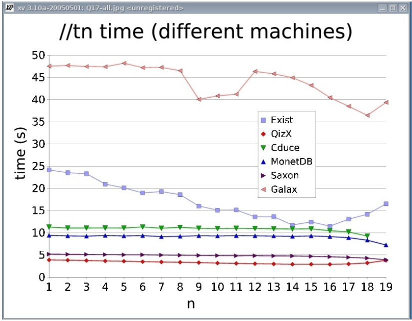
Change the graphical layout of a given curve from one figure to another

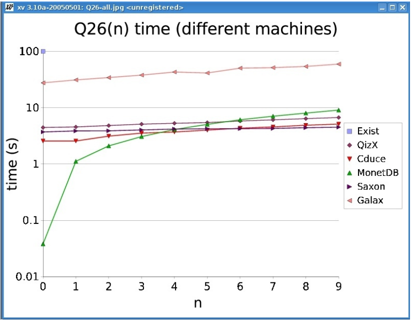




Common presentation mistakes

Change the graphical layout of a given curve from one figure to another





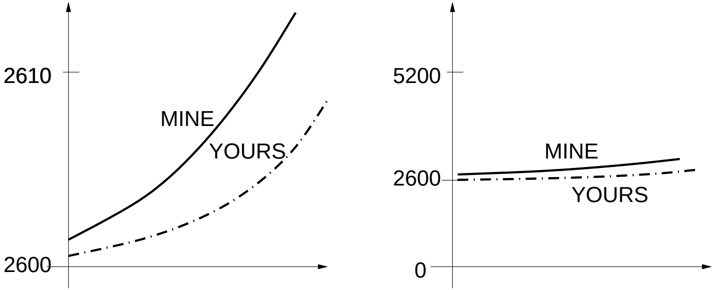
What do you mean "my graphs are not legible"?

Pictorial games

MINE is better than YOURS!

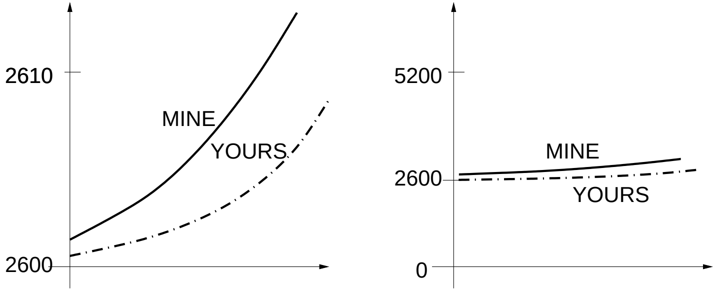
Pictorial games

MINE is better than YOURS!



Pictorial games

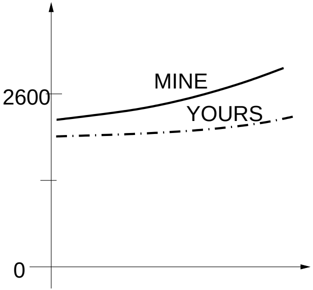
MINE is better than YOURS!



A-ha

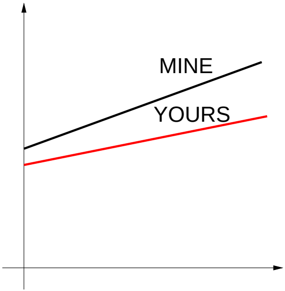
Pictorial games

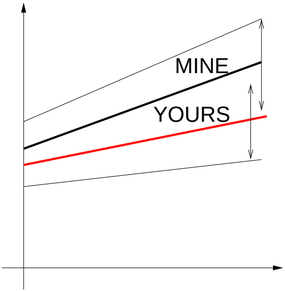
Recommended layout: let the useful height of the graph be 3/4th of its useful width



Pictorial games

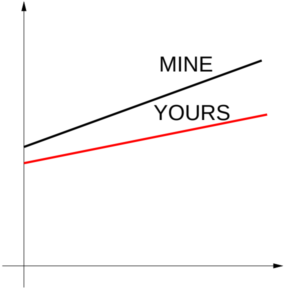
Plot random quantities without confidence intervals

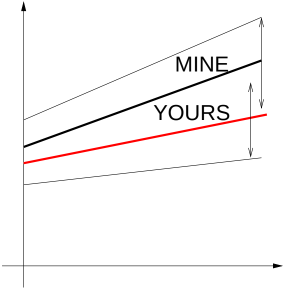




Pictorial games

Plot random quantities without confidence intervals

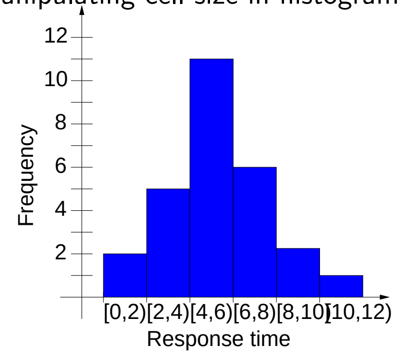


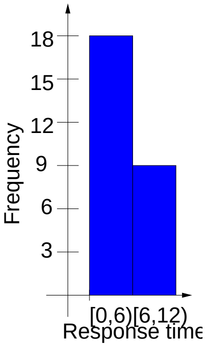


Overlapping confidence intervals sometimes mean the two quantities are statistically indifferent

Pictorial games

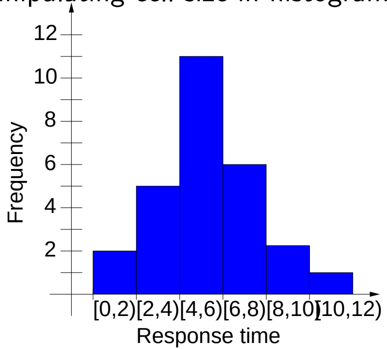
Manipulating cell size in histograms

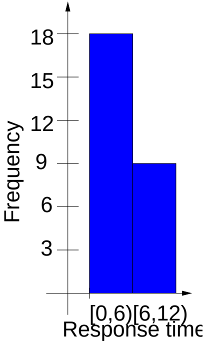




Pictorial games

Manipulating cell size in histograms

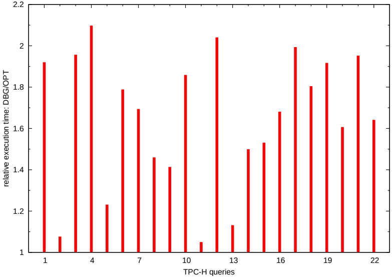


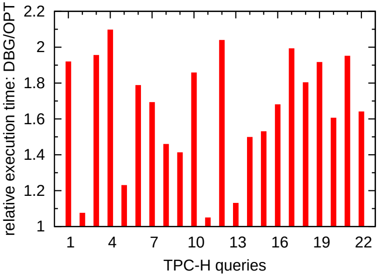


Rule of thumb: each cell should have at least five points

Not sufficient to uniquely determine what one should do.

Pictorial games: gnuplot & LaTeX

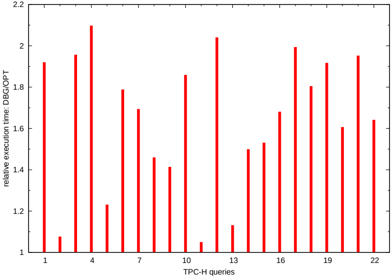




Pictorial games: gnuplot & LaTeX

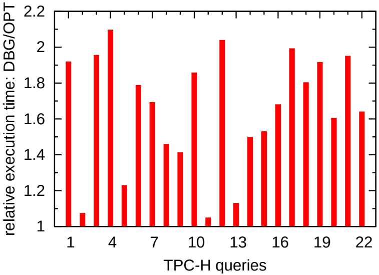
default:

`set size ratio 0 1,1`



better:

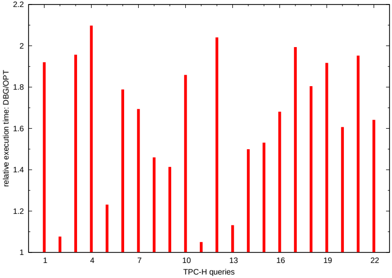
`set size ratio 0 0.5,0.5`



Pictorial games: gnuplot & LaTeX

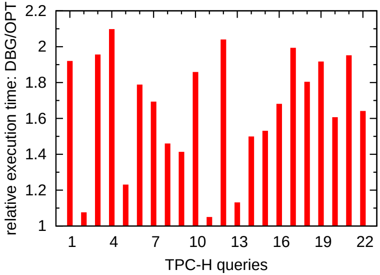
default:

`set size ratio 0 1,1`



better:

`set size ratio 0 0.5,0.5`



Rule of thumb for papers:

width of plot = *x*\textwidth

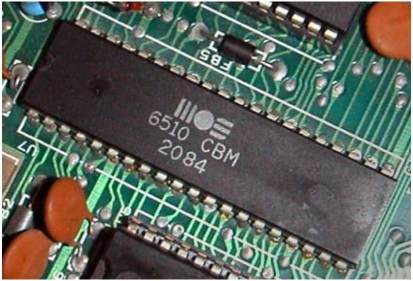
⇒ `set size ratio 0` *x*`*1.5,`*y*

Specifying hardware environments

“We use a machine with 3.4 GHz.”

Specifying hardware environments

"We use a machine with 3.4 GHz."

3400x  ?

Specifying hardware environments

“We use a machine with 3.4 GHz.”

⇒ Under-specified!

Specifying hardware environments

cat /proc/cpuinfo

```
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 13
model name      : Intel(R) Pentium(R) M processor 1.50GHz
stepping        : 6
cpu MHz         : 600.000
cache size      : 2048 KB
fdiv_bug        : no
hlt_bug         : no
f00f_bug        : no
coma_bug        : no
fpu             : yes
fpu_exception   : yes
cpuid level     : 2
wp              : yes
flags           : fpu vme de pse tsc msr mce cx8 mtrr pge mca cmov pat clflush
                  dts acpi mmx fxsr sse sse2 ss tm pbe up bts est tm2
bogomips        : 1196.56
clflush size    : 64
```

Specifying hardware environments

/sbin/lspci -v

```
00:00.0 Host bridge: Intel Corporation 82852/82855 GM/GME/PM/GMV Processor to I/O Controller (rev 02)
        Flags: bus master, fast devsel, latency 0
        Memory at <unassigned> (32-bit, prefetchable)
        Capabilities: <access denied>
        Kernel driver in use: agpgart-intel

...

01:08.0 Ethernet controller: Intel Corporation 82801DB PRO/100 VE (MOB) Ethernet Controller (rev 83)
        Subsystem: Benq Corporation Unknown device 5002
        Flags: bus master, medium devsel, latency 64, IRQ 10
        Memory at e0000000 (32-bit, non-prefetchable) [size=4K]
        I/O ports at c000 [size=64]
        Capabilities: <access denied>
        Kernel driver in use: e100
        Kernel modules: e100
```

/sbin/lspci -v | wc

```
    151 lines
    861 words
   6663 characters
```

Specifying hardware environments

/sbin/lspci -v

```
00:00.0 Host bridge: Intel Corporation 82852/82855 GM/GME/PM/GMV Processor to I/O Controller (rev 02)
        Flags: bus master, fast devsel, latency 0
        Memory at <unassigned> (32-bit, prefetchable)
        Capabilities: <access denied>
        Kernel driver in use: agpgart-intel

...

01:08.0 Ethernet controller: Intel Corporation 82801DB PRO/100 VE (MOB) Ethernet Controller (rev 83)
        Subsystem: Benq Corporation Unknown device 5002
        Flags: bus master, medium devsel, latency 64, IRQ 10
        Memory at e0000000 (32-bit, non-prefetchable) [size=4K]
        I/O ports at c000 [size=64]
        Capabilities: <access denied>
        Kernel driver in use: e100
        Kernel modules: e100
```

/sbin/lspci -v | wc

```
    151 lines
    861 words
   6663 characters
```

⇒ Over-specified!

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
- Network (interconnection): type, speed & topology
  1 GB shared Ethernet

Specifying software environments

- Product names, exact version numbers, and/or sources where obtained from

1. Planning & conducting experiments

2. Design

3. Presentation

4. Repeatability
   - Portable parameterizable experiments
   - Test suite
   - Documenting your experiment suite

5. In their words

6. Summary

Making experiments repeatable

Purpose: another human equipped with the appropriate software and hardware can repeat your experiments.

- Your supervisor / your students

Making experiments repeatable

Purpose: another human equipped with the appropriate software and hardware can repeat your experiments.

- Your supervisor / your students
- Your colleagues

Making experiments repeatable

Purpose: another human equipped with the appropriate software and hardware can repeat your experiments.

- Your supervisor / your students
- Your colleagues
- Yourself, 3 months later when you have a new idea

Making experiments repeatable

Purpose: another human equipped with the appropriate software and hardware can repeat your experiments.

- Your supervisor / your students
- Your colleagues
- Yourself, 3 months later when you have a new idea
- Yourself, 3 years later when writing the thesis or answering requests for that journal version of your conference paper

Making experiments repeatable

Purpose: another human equipped with the appropriate software and hardware can repeat your experiments.

- Your supervisor / your students
- Your colleagues
- Yourself, 3 months later when you have a new idea
- Yourself, 3 years later when writing the thesis or answering requests for that journal version of your conference paper
- Future researchers (you get cited!)

Making experiments repeatable

Purpose: another human equipped with the appropriate software and hardware can repeat your experiments.

- Your supervisor / your students
- Your colleagues
- Yourself, 3 months later when you have a new idea
- Yourself, 3 years later when writing the thesis or answering requests for that journal version of your conference paper
- Future researchers (you get cited!)

Making experiments repeatable means:

1. Making experiments portable and parameterizable
2. Building a test suite and scripts
3. Writing instructions

Making experiments portable

Try to use not-so-exotic hardware

Try to use free or commonly available tools (databases, compilers, plotters...)

Making experiments portable

Try to use not-so-exotic hardware

Try to use free or commonly available tools (databases, compilers, plotters...)

Clearly, scientific needs go first (joins on graphic cards; smart card research; energy consumption study...)

Making experiments portable

Try to use not-so-exotic hardware

Try to use free or commonly available tools (databases, compilers, plotters...)

Clearly, scientific needs go first (joins on graphic cards; smart card research; energy consumption study...)

You may omit using

Matlab as the driving platform for the experiments

Making experiments portable

- Try to use not-so-exotic hardware
- Try to use free or commonly available tools (databases, compilers, plotters...)
- Clearly, scientific needs go first (joins on graphic cards; smart card research; energy consumption study...)

You may omit using

- Matlab as the driving platform for the experiments
- 20-years old software that only works on an old SUN and is now unavailable

Making experiments portable

Try to use not-so-exotic hardware

Try to use free or commonly available tools (databases, compilers, plotters...)

Clearly, scientific needs go first (joins on graphic cards; smart card research; energy consumption study...)

You may omit using

Matlab as the driving platform for the experiments

20-years old software that only works on an old SUN and is now unavailable

- If you really love your code, you may even maintain it

Making experiments portable

Try

Try

plo

Cle

rese

You

Ma

20-

una

Code maintenance

v

Making experiments portable

- Try to use not-so-exotic hardware
- Try to use free or commonly available tools (databases, compilers, plotters...)
- Clearly, scientific needs go first (joins on graphic cards; smart card research; energy consumption study...)

You may omit using

- Matlab as the driving platform for the experiments
- 20-years old software that only works on an old SUN and is now unavailable (if you really love your code, you may even maintain it)
- 4-years old library that is no longer distributed and you do no longer have (idem)

Making experiments portable

Try to use not-so-exotic hardware

Try to use free or commonly available tools (databases, compilers, plotters...)

Clearly, scientific needs go first (joins on graphic cards; smart card research; energy consumption study...)

You may omit using

Matlab as the driving platform for the experiments

20-years old software that only works on an old SUN and is now unavailable (if you really love your code, you may even maintain it)

4-years old library that is no longer distributed and you do no longer have (idem)

`/usr/bin/time` to time execution, parse the output with `perl`, divide by zero

Which abstract do you prefer?

Abstract (Take 1)

We provide a new algorithm that consistently outperforms the state of the art.

Which abstract do you prefer?

Abstract (Take 1)

We provide a new algorithm that consistently outperforms the state of the art.

Abstract (Take 2)

We provide a new algorithm that on a Debian Linux machine with 4 GHz CPU, 60 GB disk, DMA, 2 GB main memory and our own brand of system libraries consistently outperforms the state of the art.

Which abstract do you prefer?

Abstract (Take 1)

We provide a new algorithm that consistently outperforms the state of the art.

Abstract (Take 2)

We provide a new algorithm that on a Debian Linux machine with 4 GHz CPU, 60 GB disk, DMA, 2 GB main memory and our own brand of system libraries consistently outperforms the state of the art.

There are obvious, undisputed exceptions

Making experiments parameterizable

This is huge

Making experiments parameterizable

This is huge

Parameters your code may depend on:

Making experiments parameterizable

This is huge

Parameters your code may depend on:

- credentials (OS, database, other)

Making experiments parameterizable

This is huge

Parameters your code may depend on:

- credentials (OS, database, other)
- values of important environment variables (usually one or two)

Making experiments parameterizable

This is huge

Parameters your code may depend on:

- credentials (OS, database, other)
- values of important environment variables (usually one or two)
- various paths and directories (see: environment variables)

Making experiments parameterizable

This is huge

Parameters your code may depend on:

- credentials (OS, database, other)
- values of important environment variables (usually one or two)
- various paths and directories (see: environment variables)
- where the input comes from

Making experiments parameterizable

This is huge

Parameters your code may depend on:

- credentials (OS, database, other)
- values of important environment variables (usually one or two)
- various paths and directories (see: environment variables)
- where the input comes from
- switches (pre-process, optimize, prune, materialize, plot ...)

Making experiments parameterizable

This is huge

Parameters your code may depend on:

- credentials (OS, database, other)
- values of important environment variables (usually one or two)
- various paths and directories (see: environment variables)
- where the input comes from
- switches (pre-process, optimize, prune, materialize, plot . . .)
- where the output goes

Making experiments parameterizable

Purpose: have a very simple mean to obtain a test for the values

$$f_1 = v_1, f_2 = v_2, \ldots, f_k = v_k$$

Making experiments parameterizable

Purpose: have a very simple mean to obtain a test for the values

$$f_1 = v_1, f_2 = v_2, \ldots, f_k = v_k$$

Many tricks. Very simple ones:

Making experiments parameterizable

Purpose: have a very simple mean to obtain a test for the values

$$f_1 = v_1, f_2 = v_2, \ldots, f_k = v_k$$

Many tricks. Very simple ones:

- `argc` / `argv`: specific to each class' `main`

Making experiments parameterizable

Purpose: have a very simple mean to obtain a test for the values

$$f_1 = v_1, f_2 = v_2, \ldots, f_k = v_k$$

Many tricks. Very simple ones:

- `argc` / `argv`: specific to each class' `main`
- Configuration files

Making experiments parameterizable

Purpose: have a very simple mean to obtain a test for the values

$$f_1 = v_1, f_2 = v_2, \ldots, f_k = v_k$$

Many tricks. Very simple ones:

- `argc` / `argv`: specific to each class' `main`
- Configuration files
- Java `Properties` pattern

Making experiments parameterizable

Purpose: have a very simple mean to obtain a test for the values

$$f_1 = v_1, f_2 = v_2, \ldots, f_k = v_k$$

Many tricks. Very simple ones:

- `argc` / `argv`: specific to each class' `main`
- Configuration files
- Java `Properties` pattern
- + command-line arguments

Making experiments parameterizable

Configuration files

Omnipresent in large-scale software

- Crucial if you hope for serious installations: see gnu software install procedure
- Decide on a specific relative directory, fix the syntax
- Report meaningful error if the configuration file is not found

Making experiments parameterizable

Configuration files

Omnipresent in large-scale software

- Crucial if you hope for serious installations: see gnu software install procedure
- Decide on a specific relative directory, fix the syntax
- Report meaningful error if the configuration file is not found

Pro: human-readable even without running code

Making experiments parameterizable

Configuration files

Omnipresent in large-scale software

- Crucial if you hope for serious installations: see `gnu` software install procedure
- Decide on a specific relative directory, fix the syntax
- Report meaningful error if the configuration file is not found

Pro: human-readable even without running code

Con: the values are read when the process is created

Making experiments parameterizable

Java `util.Properties`

Flexible management of parameters for Java projects

Defaults + overriding

How does it go:

- `Properties` extends `Hashtable`
- `Properties` is a map of (key, value) string pairs

  {"dataDir", "./data"} {"doStore", "true"}
- Methods:
  - `getProperty(String s)`
  - `setProperty(String s1, String s2)`
  - `load(InputStream is)`
  - `store(OutputStream os, String comments)`
  - `loadFromXML(...), storeToXML(...)`

Using java.util.Properties

One possible usage

```
class Parameters{
  Properties prop;
  String[][] defaults = {{‘‘dataDir’’, ‘‘./data’’},
                         {‘‘doStore’’, ‘‘true’’} };
  void init(){
    prop = new Properties();
    for (int i = 0; i < defaults.length; i ++)
      prop.put(defaults[i][0], defaults[i][1]);
  }
  void set(String s, String v){ prop.put(s, v); }
  String get(String s){
    // error if prop is null!
    return prop.get(s);}
}
```

Using java.util.Properties

When the code starts, it calls Parameters.init(), loading the defaults

The defaults may be overridden later from the code by calling set

The properties are accessible to all the code

The properties are stored in one place

Simple serialization/deserialization mechanisms may be used instead of constant defaults

Command-line arguments and java.util.Properties

Better init method

```
class Parameters{
  Properties prop;
  ...
  void init(){
    prop = new Properties();
    for (int i = 0; i < defaults.length; i ++)
      prop.put(defaults[i][0], defaults[i][1]);
    Properties sysProps = System.getProperties();
    // copy sysProps into (over) prop!  }
  }
```

Call with: java -DdataDir=./test -DdoStore=false pack.AnyClass

Making your code parameterizable

The bottom line: you will want to run it in different settings

- With your or the competitor's algorithm or special optimization
- On your desktop or your laptop
- With a local or remote MySQL server
- Make it easy to produce a point
- If it is very difficult to produce a new point, ask questions

Making your code parameterizable

The bottom line: you will want to run it in different settings

- With your or the competitor's algorithm or special optimization
- On your desktop or your laptop
- With a local or remote MySQL server
- Make it easy to produce a point
- If it is very difficult to produce a new point, ask questions

You may omit coding like this:

The input data set files should be specified in source file:util.GlobalProperty.java.

Building a test suite

You already have:

- Designs
- Easy way to get any measure point

You need:

- Suited directory structure (e.g.: `source`, `bin`, `data`, `res`, `graphs`)
- Control loops to generate the points needed for each graph, under `res/`, and possibly to produce graphs under `graphs`
  - Even Java can be used for the control loops, but. . .
  - It does pay off to know how to write a loop in shell/perl etc.

Building a test suite

You already have:

- Designs
- Easy way to get any measure point

You need:

- Suited directory structure (e.g.: `source`, `bin`, `data`, `res`, `graphs`)
- Control loops to generate the points needed for each graph, under `res/`, and possibly to produce graphs under `graphs`
  - Even Java can be used for the control loops, but. . .
  - It does pay off to know how to write a loop in shell/perl etc.

You may omit coding like this:

Change the value of the 'delta' variable in distribution.DistFreeNode.java into 1,5,15,20 and so on.

Automatically generated graphs

You have:

- files containing numbers characterizing the parameter values and the results
- basic shell skills

Automatically generated graphs

You have:

- files containing numbers characterizing the parameter values and the results
- basic shell skills

You need: graphs

Most frequently used solutions:

- Based on Gnuplot
- Based on Excel or OpenOffice clone

Other solutions: R; Matlab (remember portability)

Automatically generating graphs with Gnuplot

1. Data file `results-m1-n5.csv`:

1	1234
2	2467
3	4623

Automatically generating graphs with Gnuplot

1. Data file `results-m1-n5.csv`:

1	1234
2	2467
3	4623

2. Gnuplot command file `plot-m1-n5.gnu` for plotting this graph:

Automatically generating graphs with Gnuplot

1. Data file `results-m1-n5.csv`:

1	1234
2	2467
3	4623

2. Gnuplot command file `plot-m1-n5.gnu` for plotting this graph:

```
set data style linespoints
set terminal postscript color
set output “results-m1-n5.eps”
set title “Execution time for various scale factors”
set xlabel “Scale factor”
set ylabel “Execution time (ms)”
plot “results-m1-n5.csv”
```

Automatically generating graphs with Gnuplot

1. Data file `results-m1-n5.csv`:

1	1234
2	2467
3	4623

2. Gnuplot command file `plot-m1-n5.gnu` for plotting this graph:

```
set data style linespoints
set terminal postscript color
set output “results-m1-n5.eps”
set title “Execution time for various scale factors”
set xlabel “Scale factor”
set ylabel “Execution time (ms)”
plot “results-m1-n5.csv”
```

3. Call `gnuplot plot-m1-n5.gnu`

Automatically producing graphs with Excel

1. Create an Excel file `results-m1-n5.xls` with the column labels:

A	B	C
1	Scale factor	Execution time
2	. . .	. . .
3	. . .	. . .

Automatically producing graphs with Excel

1. Create an Excel file `results-m1-n5.xls` with the column labels:

A	B	C
1	Scale factor	Execution time
2	...	...
3	...	...

2. Insert in the area B2-C3 a link to the file `results-m1-n5.csv`

Automatically producing graphs with Excel

1. Create an Excel file `results-m1-n5.xls` with the column labels:

A	B	C
1	Scale factor	Execution time
2	...	...
3	...	...

2. Insert in the area B2-C3 a link to the file `results-m1-n5.csv`
3. Create in the .xls file a graph out of the cells A1:B3, chose the layout, colors etc.

Automatically producing graphs with Excel

1. Create an Excel file `results-m1-n5.xls` with the column labels:

A	B	C
1	Scale factor	Execution time
2	…	…
3	…	…

2. Insert in the area B2-C3 a link to the file `results-m1-n5.csv`
3. Create in the .xls file a graph out of the cells A1:B3, chose the layout, colors etc.
4. When the .csv file will be created, the graph is automatically filled in.

Graph generation

You may omit working like this:

In avgs.out, the first 15 lines correspond to xyzT, the next 15 lines correspond to xYZT, the next 15 lines correspond to Xyzt, the next 15 lines correspond to xyZT, the next 15 lines correspond to XyzT, the next 15 lines correspond to XYZT, and the next 15 lines correspond to XyZT. In each of these sets of 15, the numbers correspond to queries 1.1,1.2,1.3,1.4,2.1,2.2,2.3,2.4,3.1,3.2,3.3,3.4,4.1,4.2,and 4.3.

Graph generation

You may omit working like this:

In avgs.out, the first 15 lines correspond to xyzT, the next 15 lines correspond to xYZT, the next 15 lines correspond to Xyzt, the next 15 lines correspond to xyZT, the next 15 lines correspond to XyzT, the next 15 lines correspond to XYZT, and the next 15 lines correspond to XyZT. In each of these sets of 15, the numbers correspond to queries 1.1,1.2,1.3,1.4,2.1,2.2,2.3,2.4,3.1,3.2,3.3,3.4,4.1,4.2,and 4.3.

... either because you want to do clean work, or because you don't want this to happen:

Why you should take care to generate your own graphs

File `avgs.out` contains average times over three runs:

a	b
1	13.666
2	15
3	12.3333
4	13

Why you should take care to generate your own graphs

File `avgs.out` contains average times over three runs:

a	b
1	13.666
2	15
3	12.3333
4	13

Copy-paste into OpenOffice 2.3.0-6.11-fc8:

a	b
1	13666
2	15
3	123333
4	13

Why you should take care to generate your own graphs

File avgs.out contains average times over three runs:

a	b
1	13.666
2	15
3	12.3333
4	13

Copy-paste into OpenOffice 2.3.0-6.11-fc8:

a	b
1	13666
2	15
3	123333
4	13

The graph doesn't look good :-(

Why you should take care to generate your own graphs

File `avgs.out` contains average times over three runs:

a	b
1	13.666
2	15
3	12.3333
4	13

Copy-paste into OpenOffice 2.3.0-6.11-fc8:

a	b
1	13666
2	15
3	123333
4	13

The graph doesn't look good :-(

Hard to figure out when you have to produce by hand 20 such graphs and most of them look OK

Documenting your experiment suite

Very easy if they already portable, parameterizable, and if graphs are automatically generated

Specify:

1. What the installation requires; how to install
2. For each experiment
    1. Extra installation if any
    2. Script to run
    3. Where to look for the graph

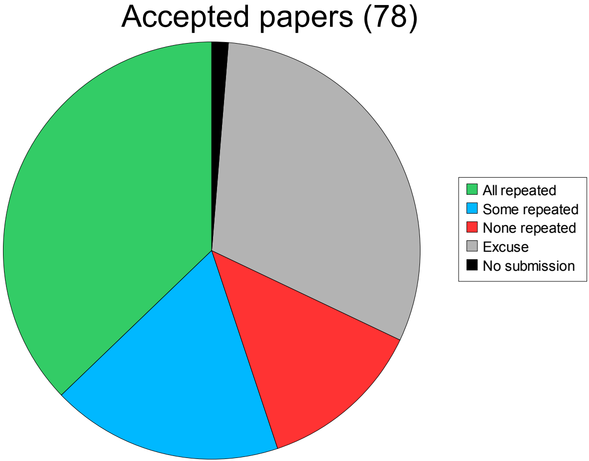
Documenting your experiment suite

Very easy if they already portable, parameterizable, and if graphs are automatically generated

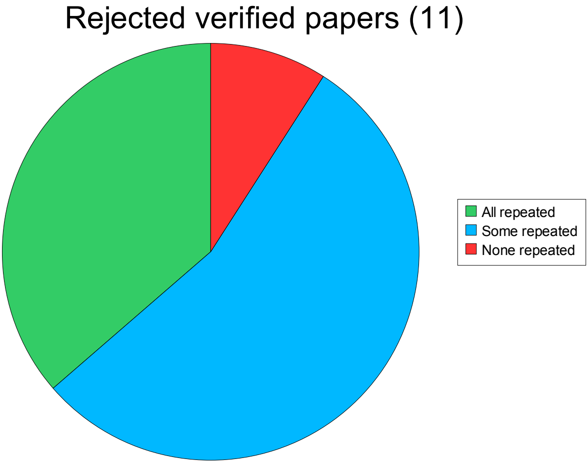
Specify:

1. What the installation requires; how to install
2. For each experiment
   1. Extra installation if any
   2. Script to run
   3. Where to look for the graph
   4. How long it takes

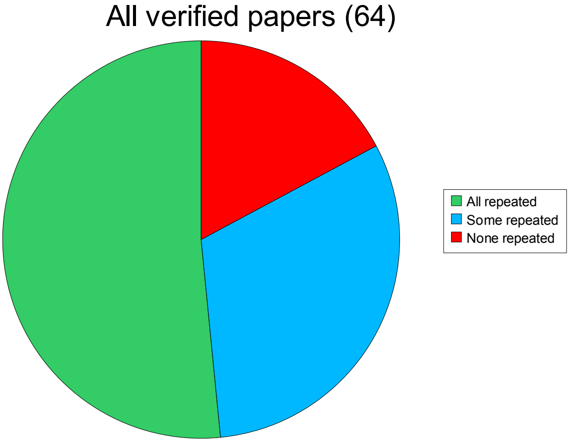
Some numbers on how SIGMOD 2008 repeatability went



Some numbers on how SIGMOD 2008 repeatability went



Some numbers on how SIGMOD 2008 repeatability went



1. Planning & conducting experiments

2. Design

3. Presentation

4. Repeatability

5. In their words
   - Excuses
   - Encouragement and thanks

6. Summary

Disclaimer

- We do *not* “blame” either the authors or the committee for anything
- We (tried to) anonymize and generalize the “war stories”
- Some of the war stories are unique, some occur more than once

Reasons not to provide the code for testing

Authors say

The work presented in this work heavily depends on the work of the primary author's Ph.D. dissertation. The primary author has graduated and due to his job commitments is unable to spend enough time to get the code base together into an executable package. The project is coupled very tightly to other on-going research work and therefore require substantiate amount of time which the primary author does not have.

Reasons not to provide the code for testing

Authors say

(1) We use other people's code and (2) we lost some old code. Due to the short notice, we could not write our own code/reproduce our lost code for these parts. If we have a 4 or 5 months ahead of the notice, we can give the code.

Reasons not to provide the code for testing

Authors say

(1) We use other people's code and (2) we lost some old code. Due to the short notice, we could not write our own code/reproduce our lost code for these parts. If we have a 4 or 5 months ahead of the notice, we can give the code.

Authors say

This system has been in development for more than three years, and it is virtually impossible to package this system in a way that it can be run from the command line.

Reasons not to provide the code for testing

Authors say

We had to manually evaluate 300 queries that were chosen randomly, and determine if a result is relevant or not, based on our judgment. This was a tedious process that we assume your committee members do not want to repeat; in addition, different people have different judgment and achieving the same results is not feasible.

Reasons not to provide the code for testing

Authors say

We had to manually evaluate 300 queries that were chosen randomly, and determine if a result is relevant or not, based on our judgment. This was a tedious process that we assume your committee members do not want to repeat; in addition, different people have different judgment and achieving the same results is not feasible.

Authors say

The subsets were chosen randomly from a large dataset, and unfortunately no trace about the identity of the used documents has been kept. The experiments were performed months ago, and it wasn't expected to send results to SIGMOD, that's why we didn't pay attention about keeping a trace.

Reasons not to provide the code for testing

Authors say

1) We can not create the batch files that reproduce the experiments in the requested format, and 2) the output of the simulator needs considerable work in order to be transformed according to the instructions, because it is based on prior work, and it is implemented before the SIGMOD instructions for the experimental evaluation. Our simulator does not take the input parameters from command line.

Encouragement from the authors

Authors say

This wasn't too hard, and I think it was definitely worth it. We even found a mistake (thankfully a minor one, not affecting our conclusions) in our submission, so I think it was very helpful. Thanks a lot for taking the time to do the repeatability eval!

Encouragement from the authors

Authors say

This wasn't too hard, and I think it was definitely worth it. We even found a mistake (thankfully a minor one, not affecting our conclusions) in our submission, so I think it was very helpful. Thanks a lot for taking the time to do the repeatability eval!

Authors say

It was helpful – we discovered an error in one of our graphs, for example, after the submission.

Encouragement from the authors

Authors say

I think the repeatability is very helpful, as we felt a great sense of achievement if other people can repeat our works and use our methods.

Encouragement from the authors

Authors say

I think the repeatability is very helpful, as we felt a great sense of achievement if other people can repeat our works and use our methods.

Authors say

I think in general it helps students to develop more solid software and algorithms although it involves work on both sides: our side to prepare more repeatable testing environment and solid test cases, and the review side to to more testing and understand the method described in the paper.

Encouragements

Senior ACM SIGMOD officer says

I personally feel that this is a VERY important direction for SIGMOD to take leadership in. It is part of a natural maturing of the field. Up until now, we've been very lax in our experimentation, but this initiative gets everyone in the field thinking about it.

Had this initiative been done for a minor workshop or conference, it would not have had much impact, but since it is done with one of the truly top conferences, I feel that everyone noticed, even those that eventually didn't submit to that conference.

It is important to continue with this requirement. It will take literally years for the field to become comfortable with it and absorb it into its consciousness. At that point, every author doing an experiment will think instinctively about repeatability, which will raise our discipline to a new level of maturity.

It can be done

Repeatability reviewer says: one command...

- Built application from sources
- Ran all experiments successfully
- Produced all tables and graphs
- Re-built paper from sources, including the re-built tables and graphs

Longest(?) war story

- Experiments: Java programs that connect to standard RDBMS

+ Instructions warned that data preparation for the full experiment might take more than 40 days(!) on a heavy 8x 3 GHz CPU server with 2 GB RAM & ≥300 GB (RAID?) disk (system)

- Authors & committee agreed to down-scale to 1/4 ⇒ 10 days for data preparation

– No info how RDBMS was (to be) configured/tuned

Longest(?) war story

- Evaluation machine: 4x dual-core Opteron @ 2 GHz, 16 GB RAM, 1 TB RAID-0 (4 disks)
- Default turned out to be (1) single threaded and (2) I/O bound, using only 200 MB of memory
- Committee tuned RDBMS to use all available memory and parallelized preparation task by distributing the workload over 4 clients

⇒ Reduced preparation task from 10 days to 2 days

Longest(?) war story

- Two more preparation steps took another 4 days (couldn't be parallelized)
- 4 experiments ran fine and finished within 4 hours in total
- Last experiment failed as the provided version could not handle the reduced data set
- Authors provided adapted version; however, simple re-run was no guaranteed to work as this experiment modified the database, and hence was not idempotent
- Re-start from scratch required another 6 days to re-crated the starting point for the last experiment.

Summary & conclusions

- 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)
- There is no single way how to do it right.
- There are many ways how to do it wrong.
- We provided some simple rules and guidelines *what (not) to do*.