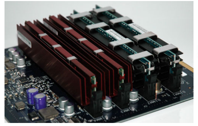
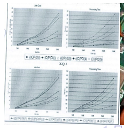
Performance Evaluation in Database Research: Principles and Experiences

Stefan Manegold

Stefan.Manegold@cwi.nl
CWI (Centrum Wiskunde & Informatica)
Amsterdam, The Netherlands
http://www.cwi.nl/~manegold/






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*.

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

- 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, ...
- Stream Processing: Linear Road, ...
- 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 time ("real")
    - user CPU time ("user")
    - system CPU time ("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 3*rd* (& 4*th*) of four consecutive runs

	server		client			run
	3*rd*		3*rd*	4*th*		
Q	user	real	real	real		... time (milliseconds)
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 3*rd* (& 4*th*) of four consecutive runs

	server		client			run
	3*rd*		3*rd*	4*th*		
Q	user	real	real	real		... time (milliseconds)
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 3*rd* (& 4*th*) of four consecutive runs

	server		client			run
	3*rd*		3*rd*	4*th*		
	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 3*rd* (& 4*th*) of four consecutive runs

	server		client			run
	3*rd*		3*rd*	4*th*		
	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?

Tools, functions and/or system calls to measure 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!
- `gettimeofday()`
  - System function ⇒ requires source code
  - Reports timestamp (*microseconds*)

Metrics: How to measure?

Tools, functions and/or system calls to measure time: Windows

- `TimeGetTime()`, `GetTickCount()`
  - System function ⇒ requires source code
  - Reports timestamp (*milliseconds*)
  - Resolution can be as coarse as 10 milliseconds
- `QueryPerformanceCounter()` / `QueryPerformanceFrequency()`
  - System function ⇒ requires source code
  - Reports timestamp (*ticks per seconds*)
  - Resolution can be as fine as 1 microsecond
- cf., `http://support.microsoft.com/kb/172338`

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
  - `mclient --interactive --timer=(clock,performance)`
  - `TRACE select ...`

Metrics: How to measure?

echo 'TRACE select 1;' | mclient --interactive

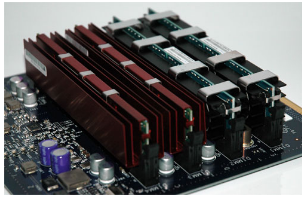
```
+--------------+
| single_value |
+==============+
|            1 |
+--------------+
1 tuple (5.977ms)
+-------+---------------------------------------------------
| ticks | stmt
+=======+===================================================
|    16 | sql.exportValue(1,".","single_value":str,"tinyint
|     9 | end s0_1;
|    50 | function user.s0_1(A0=1:bte);
|   318 | X_5:void  := user.s0_1(1:bte);
+-------+---------------------------------------------------
4 tuples (6.164ms)
```

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: MonetDB

DeBuG

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

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

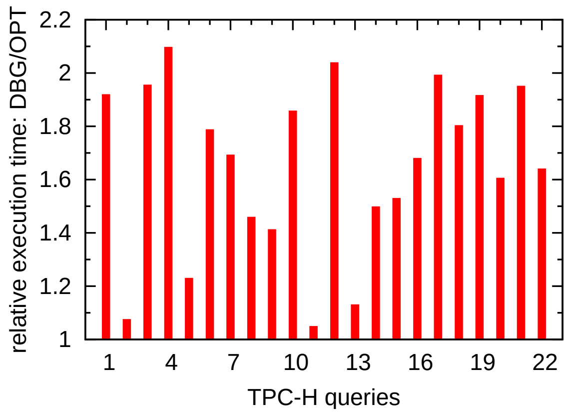
OPTimized

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

CFLAGS = "-O3 -fomit-frame-pointer -pipe ..."
```

in case of doubt, check:

```
mserver5 --version

[...]
Compilation:  gcc -O3 -fomit-frame-pointer -pipe ...
[...]
```


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 X, we focus on pure query execution time, omitting the times for query parsing, translation, optimization and result printing;

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 X, we focus on pure query execution time, omitting the times for query parsing, translation, optimization and result printing; we did not manage to do the same for 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 X, we focus on pure query execution time, omitting the times for query parsing, translation, optimization and result printing; we did not manage to do the same for Y.

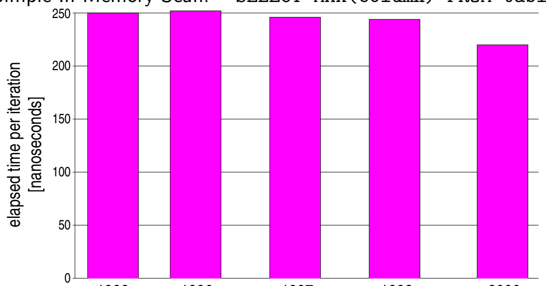
- "Absolutely fair" comparisons virtually impossible
- But:
  Be at least aware of the the crucial factors and their impact, and document accurately and completely what you do.

Do you know what happens?

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

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`

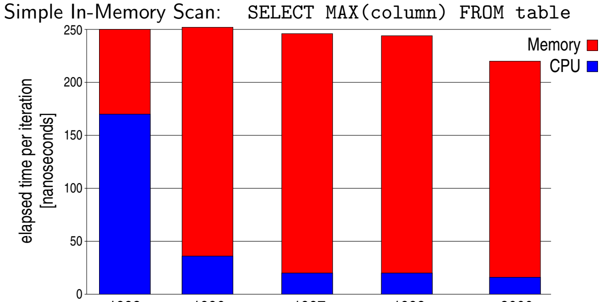
- No disk-I/O involved
- Up to 10x improvement in CPU clock-speed

⇒ 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
  - `(PLAN|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`, ...

Graphical presentation of results

We all know

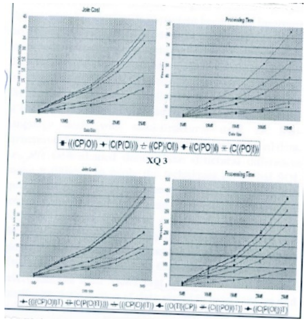
A picture is worth a thousand words

Graphical presentation of results

We all know

A picture is worth a thousand words

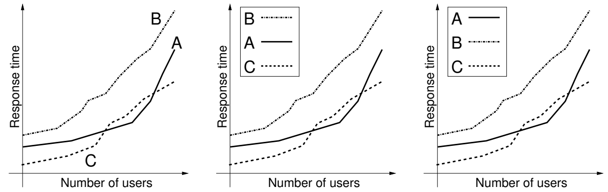
Er, maybe not all pictures...



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"

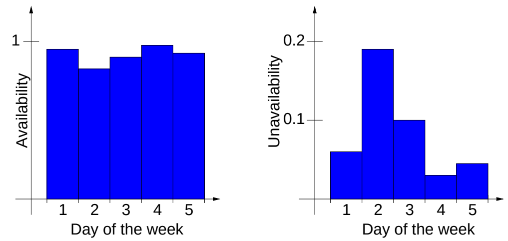
Use commonly accepted practice: present what people expect

- *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

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

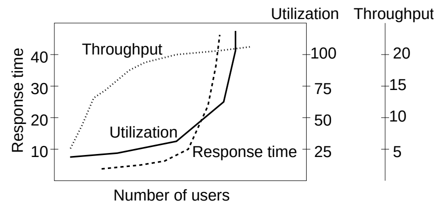
Rules of thumb, to override with good reason:

- A line chart should be limited to 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

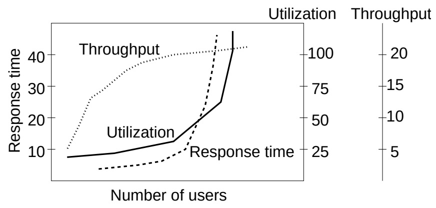
Presenting many result variables on a single chart

Commonly done to fit into available page count :-(



Common presentation mistakes

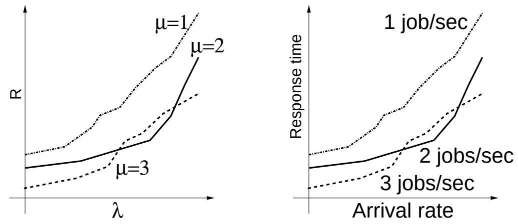
Presenting many result variables on a single chart
Commonly done to fit into available page count :-(



Huh?

Common presentation mistakes

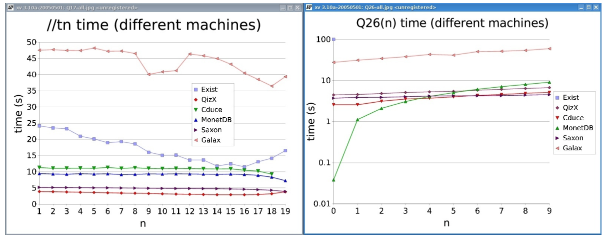
Using symbols in place of text



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

Common presentation mistakes

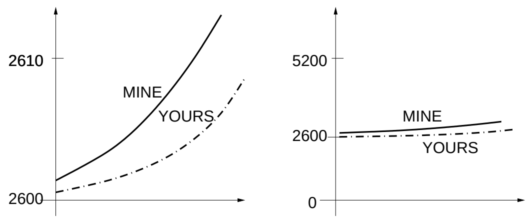
Changing the graphical layout of a given curve from one figure to another



What do you mean “my graphs are not legible”?

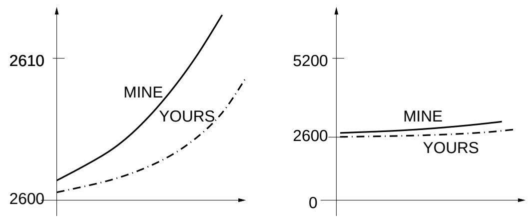
Pictorial games

MINE is better than YOURS!



Pictorial games

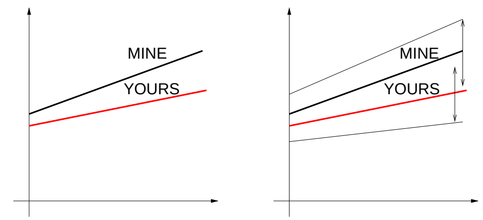
MINE is better than YOURS!



A-ha

Pictorial games

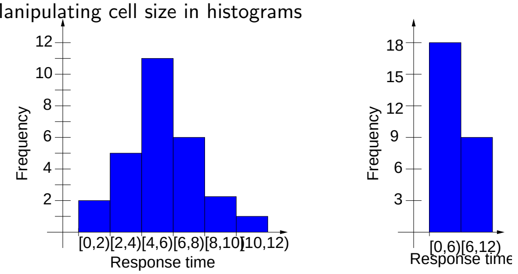
Plot random quantities without confidence intervals



Overlapping confidence intervals sometimes mean the two quantities are statistically indifferent

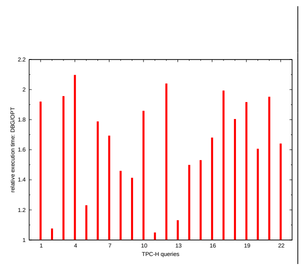
Pictorial games

Manipulating cell size in histograms

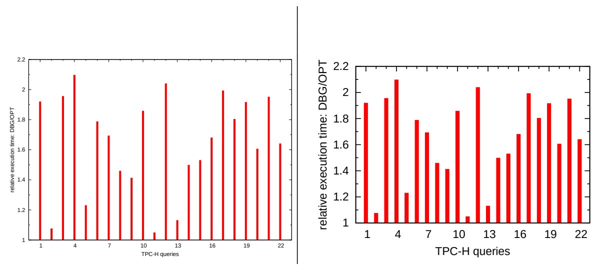


Rule of thumb: each cell should have at least five points
Not sufficient to uniquely determine what one should do.

Pictorial games: gnuplot & LaTeX

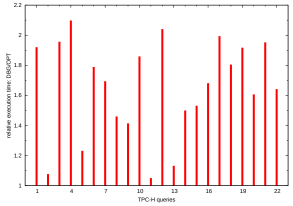


Pictorial games: gnuplot & LaTeX

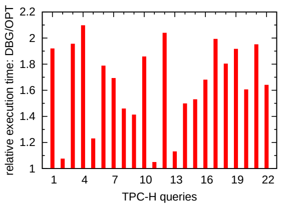


Pictorial games: gnuplot & LaTeX

default:
`set size ratio 0 1,1`

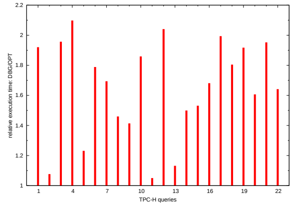


better:
`set size ratio 0 0.5,0.5`

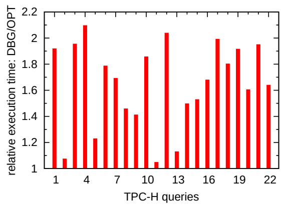


Pictorial games: gnuplot & LaTeX

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."

3400x 

?

⇒ 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

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 ⇐= throttled down by speed stepping!
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

Making experiments repeatable

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

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

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:

- 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:

- `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

Pro: human-readable even without running code
Con: the values are read when the process is created

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` to plot this graph:

```
set data style linespoints
set terminal postscript eps 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` to plot this graph:

```
set data style linespoints
set terminal postscript eps 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: ('.' decimals)

a	b
1	13.666
2	15
3	12.3333
4	13

Copy-paste into OpenOffice 2.3.0-6.11-fc8: (expecting ',' decimals)

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 experiments are 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 experiments are 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

Summary & conclusions

- Good and repeatable performance evaluation and experimental assessment require no fancy magic but rather solid craftsmanship
- 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.*