Principles of Experimental Evaluation

Ioana Manolescu

Principles of Experimental Evaluation

Ioana Manolescu

INRIA and Université Paris-Sud, France, Ioana.Manolescu@inria.fr

Partially based on
I. Manolescu and S. Manegold
"Performance Evaluation in Database Research: Principles and Experience"
IEEE ICDE 2008 and EDBT 2009

Summer School MDD 2014

Experimental evaluation and repeatability

Experiments are a "must-have" component of a data management papers (more at the end of this talk).

- What characterizes good experiments?
- Is there a right way of conducting them?
- Can this be taught?

Experimental evaluation and repeatability

Experiments are a "must-have" component of a data management papers (more at the end of this talk).

- What characterizes good experiments?
- Is there a right way of conducting them?
- Can this be taught?

ExpDB workshop 2006, panel at VLDB 2007

Experimental evaluation and repeatability

Experiments are a "must-have" component of a data management papers (more at the end of this talk).

- What characterizes good experiments?
- Is there a right way of conducting them?
- Can this be taught?

ExpDB workshop 2006, panel at VLDB 2007

ACM SIGMOD Repeatability (and Workability) Effort

Started in 2008 (Chair: IM; PC Chair: D. Shasha)

- Voluntary; 298 (out of 436) papers

Experimental evaluation and repeatability

Experiments are a "must-have" component of a data management papers (more at the end of this talk).

- What characterizes good experiments?
- Is there a right way of conducting them?
- Can this be taught?

ExpDB workshop 2006, panel at VLDB 2007

ACM SIGMOD Repeatability (and Workability) Effort

Started in 2008 (Chair: IM; PC Chair: D. Shasha)

- Voluntary; 298 (out of 436) papers

2009-2010: Repeatability and Workability co-chair with S. Manegold (CWI); A. Arion, P. Senellart, K. Karanasos, A. Katsifodimos, D. Laurent, V. Sens in the commitee

$\Rightarrow$ Tutorial with S. Manegold: ICDE 2008, EDBT 2009

Experimental evaluation and repeatability

Experiments are a "must-have" component of a data management papers (more at the end of this talk).

- What characterizes good experiments?
- Is there a right way of conducting them?
- Can this be taught?

ExpDB workshop 2006, panel at VLDB 2007

ACM SIGMOD Repeatability (and Workability) Effort

Started in 2008 (Chair: IM; PC Chair: D. Shasha)

- Voluntary; 298 (out of 436) papers

2009-2010: Repeatability and Workability co-chair with S. Manegold (CWI); A. Arion, P. Senellart, K. Karanasos, A. Katsifodimos, D. Laurent, V. Sens in the commitee

$\Rightarrow$ Tutorial with S. Manegold: ICDE 2008, EDBT 2009

R&W continues (P. Bonnet, J. Freire)

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 tutorial: a set of general rules or guidelines on *what (not) to do*.

1. Planning
2. Repeatability
3. Summary

1. Planning
   - From micro-benchmarks to real-life applications
   - Choosing the hardware
   - What and how to measure
   - How to run
   - Comparison with others
   - CSI
2. Repeatability
3. Summary

1. Planning
   - From micro-benchmarks to real-life applications
   - Choosing the hardware
   - What and how to measure
   - How to run
   - Comparison with others
   - CSI
2. Repeatability
3. Summary

Planning & conducting experiments

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

- Which data set 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?
- *Crime Scene Investigation* (CSI): How to find out what is going on?

Planning & conducting experiments

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

- Which data set 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?
- *Crime Scene Investigation* (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 piece of a larger system
- E.g., single DB operator (select, join, aggregation, etc.)

Micro-benchmarks: Pros

1. Focused on one problem
2. Controllable workload and data characteristics
   - Data sets (synthetic & real)
   - Data size / volume (scalability)
   - Value ranges and distribution
   - Correlation
   - Queries
   - Workload size (scalability)
3. Allow broad parameter range(s)
4. Low setup; easy to run

Micro-benchmarks: Pros

1. Focused on one problem
2. Controllable workload and data characteristics
   - Data sets (synthetic & real)
   - Data size / volume (scalability)
   - Value ranges and distribution
   - Correlation
   - Queries
   - Workload size (scalability)
3. Allow broad parameter range(s)
4. Low setup; easy to run

Micro-benchmarks are useful for

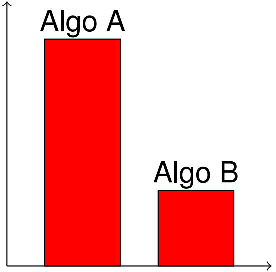
Detailed, in-depth analysis

Micro-benchmarks: Cons

- Neglect larger picture: embedding in context/system at large

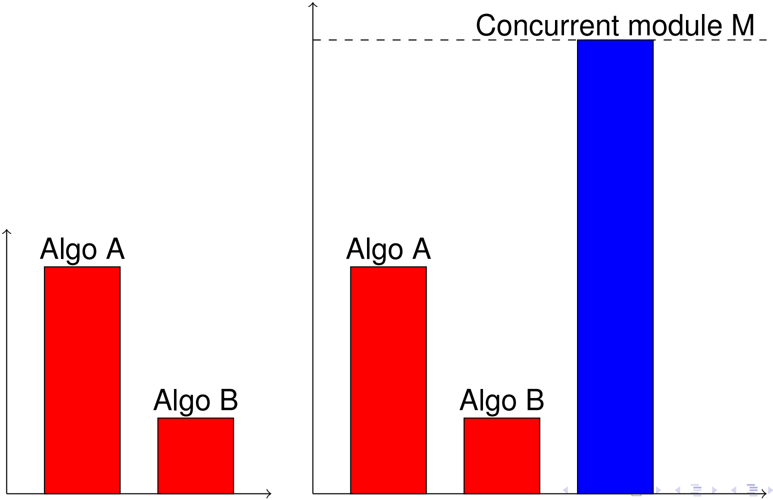
Micro-benchmarks: Cons

- Neglect larger picture: embedding in context/system at large



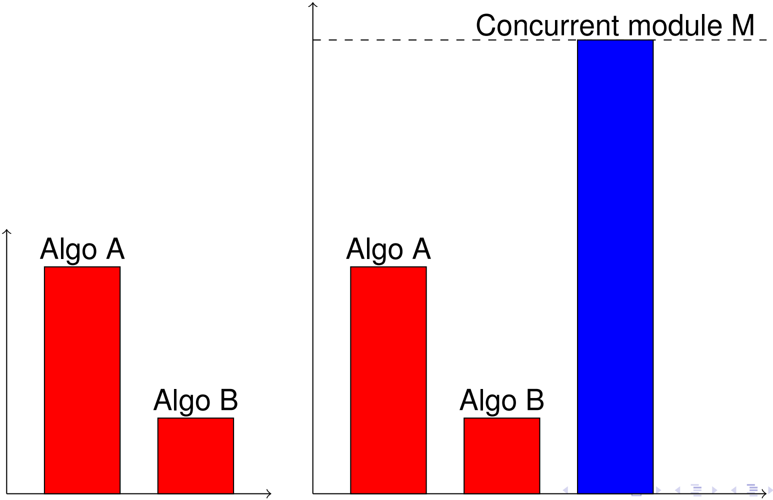
Micro-benchmarks: Cons

- Neglect larger picture: embedding in context/system at large



Micro-benchmarks: Cons

- Neglect larger picture: embedding in context/system at large



Micro-benchmarks: Cons

- Neglect larger picture: embedding in context/system at large
- Neglect contribution of local costs to global/total costs
- Generalization of result to real-life applications not obvious
- Metrics not standardized
- Comparison?

Standard benchmarks

Examples

- RDBMS, OODBMS, ORDMBS:
  TPC-{A,B,C,H,R,DS, W}, OO7, ...
- XML, XPath, XQuery, XUF, SQL/XML:
  MBench, XBench, XMach-1, XMark, X007, TPoX, ...
- RDF:
  DBLP, DBPedia, Lehigh University Benchmark (LUBM), ...
- ...

Choice of benchmark a question in itself

Standard benchmarks: Pros

1. Mimic real-life scenarios
2. Publicly available
3. Well defined (in theory ...)
4. Scalable data sets and workloads (if well designed ...)
5. Metrics well defined (if well designed ...)
6. Easily comparable results (?)

Standard benchmarks: Cons

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

Standard benchmarks: Cons

1. Often "outdated" (standardization takes (too?) long)
2. Often compromises
3. Often very large and complicated to run
4. Limited dataset variation
5. Limited workload variation
6. Systems are often optimized for the benchmark(s), only!

The fate of a successful benchmark

It is to be replaced / abandoned after a while.

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 what happens and why!

Publication

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

Exercise: interpreting experimental results

Raj Jain, The Art of Computer Systems Performance Analysis (Wiley, 1992)

Running experiments has produced the following results:

Workload	1	2
System A	10	30
System B	30	10

Compare the performance of the two systems and show that:

1. System A is better.
2. System B is better.

Choosing the hardware

The following choices are rarely "consciously" made

Yet there should be some questioning.
Others may question the choices even if you don't!
Better be the first.

Depending on: problem, knowledge, background, taste, available resources etc.

Distributed hardware:

- Rent out from cloud providers (Amazon Web Services, Google Cloud, Microsoft Azure etc.)
- Lab cluster
- Distributed network (of clusters), e.g. Grid5000 in France

Choosing the software

Which DBMS to use?

Commercial

- Require license / "free" versions (limitations?)
- Limitations on publishing results
- No access to code
- *Established* systems benefit from 100s MxY of development, testing. Distinguish from the *latest start-up bunch of code*.

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; centralized vs. distributed (parallel)
  - Memory and/or storage usage / requirements
- Comparison
  - Scale-up
  - Speed-up
- Analysis
  - System events & interrupts
  - Hardware events

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

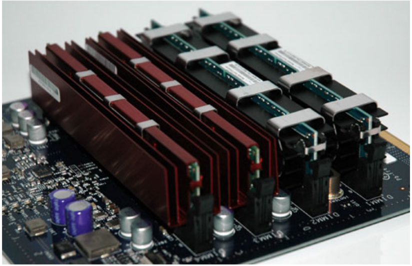
```
log_statement_stats = on
log_min_duration_statement = 0
log_duration = on
```

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]"
```

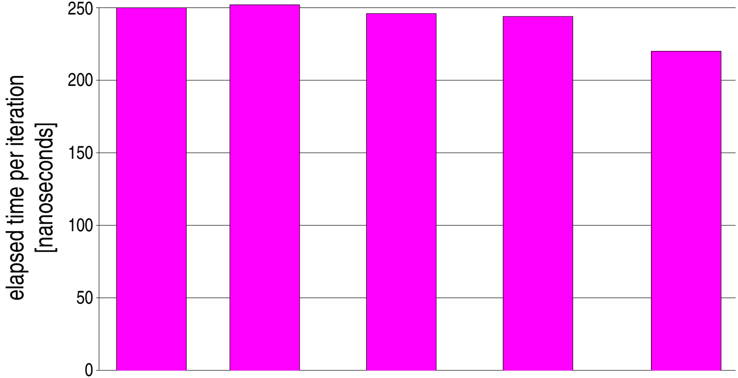
OPT

```
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
"
```

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

$\Rightarrow$ 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

$\Rightarrow$ 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`

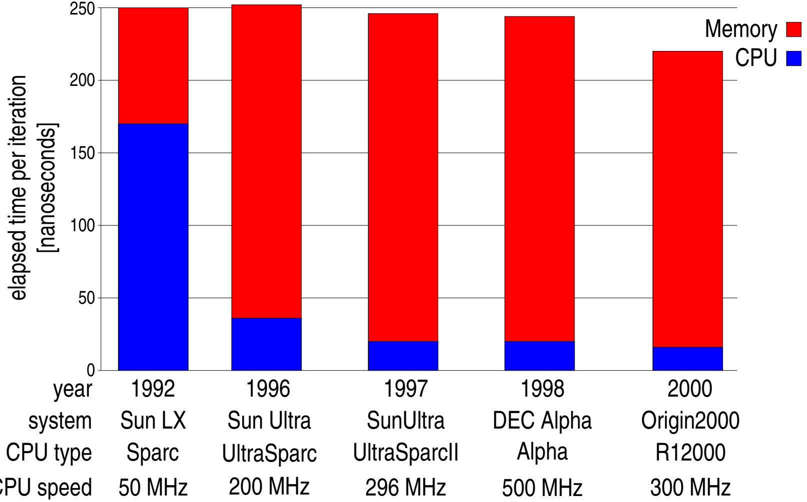
- No disk-I/O involved
- Up to 10x improvement in CPU clock-speed

⇒ 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`



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 ...`

1. Planning

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

3. Summary

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

Making experiments repeatable

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

- Your supervisor / your students

This has actually happened

1. Intern *I* does a summer internship with supervisor *S*. Internship results in code based on which measures are made which go into a publication.
2. Intern *I* leaves.
3. PhD student *P* working with *S* is asked to improve the code. *P* examines code and finds that the code does nothing (other than print a hardcoded table of numbers, assumed to be running times...)

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!)

It is actually more serious than this:

1. If the experiment is not repeatable, it is not science.

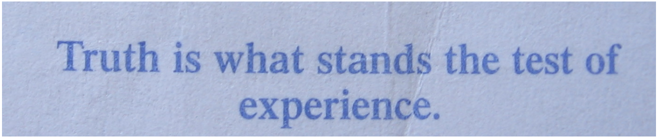
Making experiments repeatable

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!)


Truth is what stands the test of experience.

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 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 software tools (databases, compilers, plotters...)

Clearly, **scientific needs go first** (data processing on GPUs; smart card research; energy consumption study...)

The size of the audience is also impacted by the relevance / availability of your platform.

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** login info for the OS, database, other
- **environment** variables (important ones)
- **paths** and directories (see: environment variables)
- **input** where to find it
- **switches** of the system being tested (pre-process, optimize, prune, materialize, plot . . .)
- **output** where to write it

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 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`

1. When the code starts, it calls `Parameters.init()`, loading the defaults
   - May be overridden later from the code by calling `set`
2. Properties are stored in one place, visible from anywhere in the code
3. 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 DB server
- Make it easy to produce a point

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 DB server
- Make it easy to produce a point

Bottom line (again)

If it is very difficult to produce a new point, stop and 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 DB server
- Make it easy to produce a point

Bottom line (again)

If it is very difficult to produce a new point, stop and ask questions

You may omit coding like this (SIGMOD Repeatability):

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

Building a test suite

You already have:

- Easy way to get any measure point
- *Experiment design (choice of parameters)*: see long version of this tutorial

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:

- Easy way to get any measure point
- *Experiment design (choice of parameters)*: see long version of this tutorial

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 (SIGMOD Repeatability):

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 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` for plotting 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 (SIGMOD Repeatability):

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 (SIGMOD Repeatability):

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:

Generating graphs by copy-paste

File `avgs.out` with times over three runs:

a	b
1	13.666
2	15
3	12.3333
4	13

Generating graphs by copy-paste

File `avgs.out` with times over three runs:

a	b
1	13.666
2	15
3	12.3333
4	13

Copy-paste into a French-set spreadsheet:

a	b
1	13666
2	15
3	123333
4	13

Generating graphs by copy-paste

File `avgs.out` with times over three runs:

a	b
1	13.666
2	15
3	12.3333
4	13

Copy-paste into a French-set spreadsheet:

a	b
1	13666
2	15
3	123333
4	13

The graph doesn't look good :-(

Generating graphs by copy-paste

File `avgs.out` with times over three runs: ('.' decimals)

a	b
1	13.666
2	15
3	12.3333
4	13

Copy-paste into a French-set spreadsheet: (expecting ',' decimals)

a	b
1	13666
2	15
3	123333
4	13

The graph doesn't look good :-(

An example we are (more!) proud of: RDFViewS

Recommend views to materialize to speed up an RDF query workload
Joint work with F. Goasdoué, K. Karanasos and J. Leblay (2011-2012)

Parameters

- Queries: number, shape, number of variables/constants, degree of commonality between the queries
- Database (dictates the statistics)
- View recommendation: strategy

An example we are (more!) proud of: RDFViewS

Recommend views to materialize to speed up an RDF query workload
Joint work with F. Goasdoué, K. Karanasos and J. Leblay (2011-2012)

Parameters

- Queries: number, shape, number of variables/constants, degree of commonality between the queries
- Database (dictates the statistics)
- View recommendation: strategy

The next few slides courtesy of Julien Leblay

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

Repeatability, 6 years after

- Effort continued next to SIGMOD until 2012, with diminishing participation and visibility.
- Failure: `pubzone.org`.
- VLDB tried a dedicated track (G. Alonso).
- Attempts to use (scientific) workflow tools to model / capture the experiments. Informative for the scientific wokflow experts....
- Repeatability has become a conversation topic
- CMT has started to be configured with repeatability questions!

Repeatability, 6 years after

- Effort continued next to SIGMOD until 2012, with diminishing participation and visibility.
- Failure: `pubzone.org`.
- VLDB tried a dedicated track (G. Alonso).
- Attempts to use (scientific) workflow tools to model / capture the experiments. Informative for the scientific wokflow experts....
- Repeatability has become a conversation topic
- CMT has started to be configured with repeatability questions!
  - Is this a good thing? :)

Repeatability, 6 years after

- Effort continued next to SIGMOD until 2012, with diminishing participation and visibility.
- Failure: `pubzone.org`.
- VLDB tried a dedicated track (G. Alonso).
- Attempts to use (scientific) workflow tools to model / capture the experiments. Informative for the scientific wokflow experts....
- Repeatability has become a conversation topic
- CMT has started to be configured with repeatability questions!
  - Is this a good thing? :)
  - "*Experiments as Research Validation – Have We Gone too Far?*" Jeffrey D. Ullman, July 9, 2013 (see next)

"Experiments as Research Validation – Have We Gone too Far?"

I recently submitted a paper to VLDB, and when I got the reviews back, I noticed that the review form now has a question referees are required to answer, about *whether the experiments were well carried out, with choices like "believable" and "not believable."*

The reviewers had a bit of trouble with that question, because my paper had no experiments; it was a paper about computational complexity of MapReduce algorithms. Two of the reviewers said the nonexistent experiments were not believable, which is wrong – you have to see something to disbelieve it.

"Experiments as Research Validation – Have We Gone too Far?"

It appears the database community has now reached the point where *experiments are no longer an option* [...] It is time to restore the balance, where *experiments are used when appropriate, and ideas that require analysis rather than experiments are handled appropriately*, rather than "justified" by inappropriate and meaningless experiments.

Good ideas should stand on their own. Look at the two database ideas that have won the Turing award: the relational model and 2-phase locking [...] Neither paper was about experiments. *Should we have rejected a paper that said "let's organize data into relations" because there was no experiment to prove that its queries were executable more efficiently? Would we want to reject the 2PL paper because it did not measure experimentally the space required by locking tables?*

Summary & conclusions

- Good and repeatable performance evaluation and experimental assessment require no fancy magic but rather solid craftmanship
- Proper planning keeps from “getting lost” and ensure repeatability
- **Repeatable experiments simplify your own work** (and help others understand it better)
- There is no single way how to do it right.
- There are many ways how to do it wrong.
- Simple rules and guidelines on *what (not) to do*.
  1. Find out (check!) **what happens**.
  2. Make it **easy to obtain a point**.
  3. Automate the rest (plots etc.)

More on performing and presenting experiments

The Raj Jain book

A classic, including:

- Experimental design: chosing parameter values
- Statistics (data analysis for meaningful conclusions)
- Presenting experimental results

Long version of this tutorial

- Introduction to experimental design; experiment presentation
- War stories from the SIGMOD 2008 repeatability evaluation

Thank you!

Questions?