
WISE 2004
**What does it mean to
“Measure Performance”?**

Alistair Moffat
The University of Melbourne

Justin Zobel
RMIT University

1

What do we do?

As computer scientists, what do we do?

Dijkstra joked that things would have been different if astronomy had been called “telescope science”. He was right – we suffer from our name.

That is why we always get asked about Windows problems at parties.

To me, the most important thing that we do as a community is *algorithmics*.

We study, design, invent, implement, test, and apply, algorithms.

3

Roadmap

Today’s talk:

- What do we do?
- Why do we do it?
- How do we do it?
- How should we do it?
- Case study: Distributed text search
- Outcomes?

2

What do we do?

Various groups of people contribute in different ways.

For example, software engineers put algorithms to productive use, to increase our living standards.

As research academics, we regard our job as being to develop new algorithms, to do the *science* associated with our discipline.

We are “*algorithmicians*”.

An even better word might be “*algorithmagicians*”.

4

Why do we do it?

Anyone?

5

How do we do it?

We need to remember that what we are doing is *science*, and that there is an established process to be applied:

1. Design a theory that accounts for all known observations;
2. Using that theory, predict (as a *hypothesis*) further behaviors that would not have been forecast by previous theories;
3. Design experiments to distinguish between the alternative behaviors predicted by the new and old theories;
4. Carry out the experiments in an impartial manner;

7

Why do we do it?

Primarily, because it is fun.

- the field is young and the pace of change is dramatic;
- outcomes can (still) be significant;
- we can create elegant intellectual structures;
- we keep our hands clean, with no formaldehyde or grime; and
- we get to play with neat technology.

6

How do we do it?

5. Evaluate the outcomes of the experiments to determine whether they support or contradict the hypothesis;
6. Communicate our findings to the community, in sufficient detail that others can replicate the experiments and confirm our claims.

In experimental science very little gets “proven”. Rather, a theory is accepted for a while as explaining all known facts – but only until other facts are encountered that cannot be so explained.

We also have an obligation to choose the simplest theory that is consistent with the available observations.

8

How do we do it?

In algorithmics, a theory is a claim about the behavior of an algorithm, as applied to some particular problem, or class of problem, or problem domain.

These theories often include two parts: a statement about *effectiveness*, and a statement about *efficiency*.

Sometimes the statements are absolute: “the list is sorted”, or “the running time is $O(n^2 \log n \log \log n)$ ”.

Sometimes the statements in the theorem are relative: “compression is improved”; or “the new method is more scalable than previous methods”.

9

How do we do it?

Unfortunately, we sometimes fail somehow.

We develop a new or improved technique, and write a paper describing it.

Then, if we think we won't get the paper accepted at our chosen journal or conference otherwise, we test it by:

- getting a summer student to implement our proposal;
- picking up an alternative solution from someone's web site;
- finding some data; and
- doing lots of runs and drawing lots of graphs.

11

How do we do it?

Algorithmic claims are typically defended by some combination of

- mathematical argument, or proof;
- simulation;
- rhetoric; and
- *demonstration*.

If we rely on a demonstration, it needs to meet the standards associated with experimental science.

10

How do we do it?

But, a year or two later, our paper is gone; and our theorem may as well have never existed.

Our work doesn't even get the scientific “status” of having been overturned.

Instead, it is simply ignored, and earns few (or zero!) citations.

Even though our paper reported an “improvement” in effectiveness or efficiency or both.

Why does this happen?

12

How do we do it?

One answer is that the landscape is very cluttered – there are too many theories. All but a few are fated to be forgotten (and then rediscovered).

Another possible answer – and the topic of this talk – is that our arguments for usefulness sometimes lack credibility.

Perhaps our experiments were flawed in some way, or our data was inadequate, or the baselines inappropriate, or we were so intent on the “good” that we failed to account for the “bad”.

13

Baselines

Experimental work is rarely absolute. So we should be clear as to a reference point, or *baseline*.

For our work to be taken seriously, the baseline should be a good one.

It should represent the best competition (the current theory), and have been implemented with as much attention to detail as the new work.

15

How should we do it?

A number of aspects of any experiment need to be carefully considered:

- Setting an appropriate baseline;
- Using plausible data;
- Measuring the right thing;
- Reporting sensibly.

14

Baselines

If the argument is primarily one of *effectiveness*, then numeric results from previous papers can be drawn on with reasonable confidence, provided the same data is used in the same way.

If the main claim is one of *efficiency*, hardware/software/system differences mean that the baseline must be executed afresh.

16

Baselines

At the end of your experiments, consider making your software publicly available, or available on request. This is a community contribution that helps others do their science.

“If I have seen further than others, it is because I have stood on the shoulders of giants”, and “We build too many walls and not enough bridges” (Isaac Newton).

It is ok to ask/expect that use of your software be acknowledged in the usual manner – via a citation or explicit statement of thanks. You can also request that software be used for “research purposes only”.

17

Data

Generation of reference data sets is a valid research activity, and “gold standards” are enormously valuable.

In the area of compression, there are several standard corpora used for comparative effectiveness results. They are low-cost standards, but very important ones. The first of these was publicised in 1990.

In the area of text retrieval, millions of dollars have been spent over more than a decade to create the various TREC resources.

These public data sets have corresponded to periods of considerable improvement in the technology for handling the problems they quantify.

19

Data

Data can be even more problematic than software.

- It should be as real as possible, as an experiment is an argument for practical use. Use of artificial data introduces a risk that the results will not generalise.
- It should be of a realistic volume to the task being considered (be aware of cache and main memory sizes).
- It should ideally be something that is (or can be) shared with other researchers.

(Does anyone actually ever need to sort random integers?)

18

Measurement

Measurement of efficiency is rarely straightforward. There will be complex interactions between hardware and software that you may never track down.

So be a bit circumspect in your claims, especially when they revolve around differences of 10% or less.

“Send excited email to your supervisor telling them that you have been tuning some code that your supervisor wrote, and you now have it running about 30% faster than it was a year earlier when you first benchmarked it. Admit a week later that the machine you used for your most recent testing is 50% faster than the one used a year earlier.”

(http://www.cs.mu.oz.au/~alistair/research_success.html).

20

Measurement

Different factors are usually in tension, and can be traded against each other. Try to evaluate the spectrum, rather than isolated points at the extremes.

(“Install insulation: be warmer in winter and cooler in summer *while you save on your energy bills*”.)

Try to vary just one dimension at a time. In each experiment, be sure that you understand what the “variable” is, and what it is you are measuring as a function of that variable.

Be aware of biases that might be introduced by the data or hardware.

21

Reporting

If appropriate, perform a suitable significance tests on the numbers you are generating. You might be surprised by the outcome.

Be open about the warts. If your superb performance can only be achieved when all the data is in main memory rather than disk, then say so. Try to give some results for the adverse situation too.

Can you quantify the region of superiority? Can you describe the extent of the complementary region in which it is inferior?

One interesting type of graph shows the tradeoffs between efficiency and effectiveness offered by alternative techniques.

23

Measurement

When testing for effectiveness, be aware of the risk of over fitting.

If your method has “tuning factors” that need to be set, split the data into a training part and a test part.

Tune on the training set. Then evaluate on the test set. Distinguish between the two activities, and don’t let them inform each other.

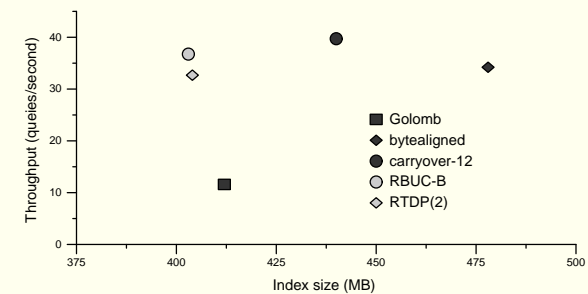
Then, if you can get better results by perturbing the knobs while running on the test data, be candid – admit in your paper that you have encountered instability.

This is just basic good science. Think about the care that goes into a drug trial, for example.

22

Reporting

Tradeoffs in effectiveness, measured as index size for an 18 GB document collection, and querying throughput, measured as queries per second on a dual 2.8 Ghz Intel Xeon with 2 GB of RAM, twelve 146 GB SCSI disks in a RAID-5 configuration, and running Debian GNU/Linux, over 10,000 queries averaging 3 terms each.



24

Reporting

Try to give enough information in your report that someone else could execute the same experiment, to validate your results.

Give a precise description of your data and hardware.

Cite any software implementations that you have used in your testing – the documentation will tell you who the authors were.

Finally, make it clear as to whether you are actually recommending the new technique, or merely describing its existence.

25

Case Study – Distributed text search

The size and exact shape of the feasible region depends on many things:

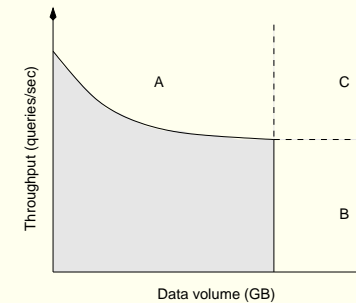
- the processor speed, and other characteristics;
- the amount of main memory;
- the amount of disk, and access rates;
- the quality of the compiler;
- the skill of the programmer; and
- the fundamental nature of the algorithm.

As an example of the latter point, compression can extend the feasible region to the right; and possibly upward as well.

27

Case Study – Distributed text search

Take a “unit” computer, with a fixed volume of disk, memory, and a given processor speed. It can deal with a given amount of data, and a given query load. There is a feasible region, and three zones beyond it.



26

Case Study – Distributed text search

To cross into region “A”, replication of the service is required. With identical data mirrored on a second machine, a greater query load can be handled. This kind of growth is highly scalable.

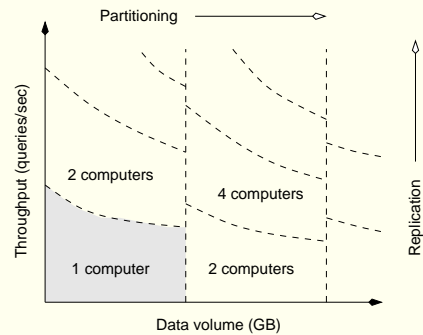
To enter region “B”, the data must be partitioned. This is the interesting case. Scalability cannot be guaranteed, and there are two non-trivial questions to be dealt with:

- how to partition the collection, while minimising redundant storage; and
- how to synthesise an overall result from partial ones, while minimising redundant computation.

28

Case Study – Distributed text search

For arbitrary data volumes and query loads:



29

Data partitioning for text search

Performance indicator	Monolithic system	Index partitioned	Document partitioned
Number of servers	1	q	k
<i>Per processor</i>			
Disk seeks and transfers	q	1	q
Index volume transferred	I	I/q	I/k
Number of documents scored	r	0	r
<i>Plus</i>			
Receptionist load	n/a	$I + r$	kr
Network volume	n/a	I	kr
<i>Total cost</i>	$I + q + r$	$I + q + r$	$I + kq + kr$

31

Data partitioning for text search

There are two standard ways of partitioning in text retrieval systems.

Document-based partitioning: each server holds all of the index information for a subset of the documents. If there are k servers and r answers required, then the final r are selected from a pool of $r \cdot k$ answers returned from the subcollections.

Index-based partitioning: each server holds all of the index information for a subset of the vocabulary. Queries are answered by merging the lists for the query terms. Fewer servers need to cooperate on each query.

Suppose that a query has q terms, that the volume of index data to be processed is I , that r answers are required, and that k processors are in use.

30

Data partitioning for text search

From this discussion, it looks like index partitioning is more scalable than document partitioning. Less redundant computation is performed, and the total cost appears to be independent of k .

In fact, I and k have to be related – it makes no sense to increase k except in response to large index sizes.

Even worse, there is a bottle neck created at the receptionist. Only q servers are active on any given query, and every query requires time at the receptionist.

There is a real risk that $k - q$ processors will be idle a lot of the time.

32

A third paradigm

The index-partitioned arrangement has the advantage of requiring less access to the inverted file – fewer seeks for the same volume of data transferred.

Earlier in 2004 (with Ricardo Baeza-Yates and William Webber) we developed the idea of a *pipelined* approach, in which a partially evaluated query is passed through the relevant servers.

The goal was to avoid the additional disks costs and redundant scoring of document-partitioning, but also to bypass the receptionist as a bottle neck.

33

An experiment

Software: we started with the public ZettaIR system developed at RMIT University, and extended it considerably. Approximately 0.5 person-years have been spent so far. Once the experiments are complete and we have confirmed (or not) our theory, the improvements will be fed back into the release version.

Data: We used several of the document collections established by TREC, including the *wt100g* collection of 100 GB. This is big, but still small by web standards, unfortunately.

Experimental hardware: We have a cluster of 20 PCs of two clock speeds, a result of grant funding, plus access to other similar clusters.

35

A third paradigm

In the pipelined method, q servers need to handle each query; each server accesses an average of I/q as an index volume with one disk transfer; a total of r documents are fully scored; and there is no need for any coordination at the receptionist.

This is an attractive combination of attributes.

And guess what? We needed to design and carry out an experiment!

34

An experiment

Measurement: We measured elapsed time to process a stream of queries. From this we calculated throughput. Effectiveness is not an issue, since we are simply duplicating a previous computation. This experiment is purely about efficiency.

Start-up costs were eliminated by taking the difference between the time for 20,000 and 10,000 queries.

It quickly became clear that threading was a critical issue. Having more than one query active at a time was very important, and we made the necessary changes.

36

An experiment

Reporting: units of

$$\frac{\text{GB} \times \text{queries}}{\text{machines} \times \text{seconds}}$$

were used to relate the observed results back to the diagram shown previously.

For any given query, the length I of the inverted lists involved grow linearly as the size of the collection. So $\text{GB} \times \text{queries}$ is a measure of work to be performed; $\text{machines} \times \text{seconds}$ is a measure of the resource spent doing that work.

Big numbers are good. Small numbers are bad.

37

Outcomes?

On the negative side: our paper describing the idea of pipelining, and the first round of experiments, was rejected.

One referee was uncomfortable with our reporting metric and the way our results were presented.

A second asked for experiments on terabytes or petabytes of data, and expressed scepticism of the scalability of the system.

So we clearly still have work ahead of us to demonstrate the usefulness of our approach. The experiments have not yet reached the stage of being plausible.

39

Outcomes?

By now, everyone is on the edge of their seat wondering about outcomes.

k	wt100g		
	Doc	Term	Pipe
1	248.4	—	—
2	236.3	148.1	229.1
4	220.0	71.7	197.3
8	193.7	35.4	189.3

Pipelining is as good as document partitioning.

On the other hand, term partitioning is flawed.

38

Outcomes?

On the positive side: there was also encouragement from the referees. We are running experiments on a 500 GB collection at the moment, and hope to generate a more compelling validation.

We are also developing a variant that offers promise of further enhancement.

And we have ahead of us the pleasure of doing better experiments, continuing our science, and having more fun.

40

A real system

What does Google use?

According to their published descriptions, Google indexes around 20 TB of data using a document-partitioned indexing, with a cluster of processors collectively housing a complete copy of the index. Each incoming query is dynamically routed to one cluster.

To achieve the query load, clusters are replicated as required. New data centers are commissioned as load rises, each housing hundreds of clusters and tens of thousands of computers.

But their expertise does not mean that we should discontinue the search for alternative approaches.

Thanks

To the Australian Research Council and the Center for Perceptive and Intelligent Machines in Complex Environments, for financial support.

To colleagues, who have been encouraging when required, and critical when appropriate.

To the WISE organisers, for their vote of confidence.

To you, for listening.