Course "Empirical Evaluation in Informatics"

Benchmarking

Prof. Dr. Lutz Prechelt
Freie Universität Berlin, Institut für Informatik
http://www.inf.fu-berlin.de/inst/ag-se/

- Example 1: SPEC CPU2000
- Benchmark = measure + task sample + comparison
- Problems: cost, task composition, overfitting
- Quality attributes: accessibility, affordability, clarity, portability, scalability, relevance.
- Example 2: TREC
"Empirische Bewertung in der Informatik"

**Vergleichstests (Benchmarks)**

Prof. Dr. Lutz Prechelt  
Freie Universität Berlin, Institut für Informatik  
http://www.inf.fu-berlin.de/inst/ag-se/

- Beispiel 1: SPEC CPU2000  
- Benchmark = Maß + Aufgabe + Vergleich  
- Probleme: Kosten, Aufgabenauswahl, Überanpassung  
- Qualitätsmerkmale: Zugänglichkeit, Aufwand, Klarheit, Portierbarkeit, Skalierbarkeit, Relevanz  
- Beispiel 2: TREC
"Benchmark"

Merriam-Webster online dictionary, m-w.com:

• a mark on a permanent object indicating elevation and serving as a reference in topographic surveys and tidal observations

• a point of reference from which measurements may be made

• a standardized problem or test that serves as a basis for evaluation or comparison (as of computer system performance)
Example 1: SPEC CPU2000

- **SPEC = Standard Performance Evaluation Corporation**
  - A not-for-profit consortium of HW and SW vendors etc.
  - Develops standardized measurement procedures (benchmarks) for various aspects of computer system performance
    - CPU (including cache and memory)
    - Enterprise services (Web Services)
    - Graphics
    - High-performance computing: message-passing, shared-memory parallel computing
    - Java (client, server)
    - Mail server
    - Network file system
    - Web server
  - We consider the CPU benchmark
Sources

- http://www.spec.org


- (The current version is SPEC CPU2006)
- (Previous versions were defined in 1992 and 1995)
CPU2000 approach

- Select a number of real-world programs
  - must be portable to all Unix and Windows systems of interest
  - in different languages: Fortran, C, C++
  - balance different aspects such as pipelining, cache, memory performance etc.
  - some emphasize floating point computations (SPECfp2000)
  - others have only integer operations (SPECint2000)
    - now renamed CFP2006 and CINT2006

- Specify concrete program runs for each program

- Package programs and runs so as to make them easily applicable on any new system
  - application requires recompilation:
    - SPEC also tests compiler performance!
CPU2000 performance measures

There are 2 x 2 different measurement modes:

• 2 different compiler settings:
  • using basic compiler optimization settings
    • \( \rightarrow \) SPECint_base2000, SPECfp_base2000
  • using aggressive settings
    • \( \rightarrow \) SPECint2000, SPECfp2000
    • requires experimentation and experience with the compiler

• 2 different measurements:
  • measuring speed (1 task)
  • measuring throughput (multiple tasks)
    • \( \rightarrow \) SPECint_rate2000, SPECint_rate_base2000 etc.
    • throughput is relevant for multi-user systems or long-running processes

Benchmarks need to decide on many details!
CPU2000 performance measures (2)

- Performance is expressed relative to a reference machine
  - Sun Ultra 5, 300 MHz
  - defined to have performance 100

- Overall performance is determined as the geometric mean over the n benchmark programs
  - geometric mean: n-th root of the product
  - e.g. mean of 100 and 200 is 141
  - best results require steady performance across all programs
## CPU2000
integer benchmark composition

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Language</th>
<th>KLOC</th>
<th>Resident size (Mbytes)</th>
<th>Virtual size (Mbytes)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>164.gzip</td>
<td>C</td>
<td>7.6</td>
<td>181</td>
<td>200</td>
<td>Compression</td>
</tr>
<tr>
<td>175.vpr</td>
<td>C</td>
<td>13.6</td>
<td>50</td>
<td>55.2</td>
<td>FPGA circuit placement and routing</td>
</tr>
<tr>
<td>176.gcc</td>
<td>C</td>
<td>193.0</td>
<td>155</td>
<td>158</td>
<td>C programming language compiler</td>
</tr>
<tr>
<td>181.mcf</td>
<td>C</td>
<td>1.9</td>
<td>190</td>
<td>192</td>
<td>Combinatorial optimization</td>
</tr>
<tr>
<td>186.crafty</td>
<td>C</td>
<td>20.7</td>
<td>2.1</td>
<td>4.2</td>
<td>Game playing: Chess</td>
</tr>
<tr>
<td>197.parser</td>
<td>C</td>
<td>10.3</td>
<td>37</td>
<td>62.5</td>
<td>Word processing</td>
</tr>
<tr>
<td>252.eon</td>
<td>C++</td>
<td>34.2</td>
<td>0.7</td>
<td>3.3</td>
<td>Computer visualization</td>
</tr>
<tr>
<td>253.perlbmk</td>
<td>C</td>
<td>79.2</td>
<td>146</td>
<td>159</td>
<td>Perl programming language</td>
</tr>
<tr>
<td>254.gap</td>
<td>C</td>
<td>62.5</td>
<td>193</td>
<td>196</td>
<td>Group theory, interpreter</td>
</tr>
<tr>
<td>255.vortex</td>
<td>C</td>
<td>54.3</td>
<td>72</td>
<td>81</td>
<td>Object-oriented database</td>
</tr>
<tr>
<td>256.bzip2</td>
<td>C</td>
<td>3.9</td>
<td>185</td>
<td>200</td>
<td>Compression</td>
</tr>
<tr>
<td>300.twolf</td>
<td>C</td>
<td>19.2</td>
<td>1.9</td>
<td>4.1</td>
<td>Place and route simulator</td>
</tr>
</tbody>
</table>
### SPECf2000

<table>
<thead>
<tr>
<th>Program</th>
<th>Language</th>
<th>Time</th>
<th>Memory</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>168.wupwise</td>
<td>F77</td>
<td>1.8</td>
<td>176</td>
<td>Physics: Quantum chromodynamics</td>
</tr>
<tr>
<td>171.swim</td>
<td>F77</td>
<td>0.4</td>
<td>191</td>
<td>Shallow water modeling</td>
</tr>
<tr>
<td>172.mgrid</td>
<td>F77</td>
<td>0.5</td>
<td>56</td>
<td>Multigrid solver: 3D potential field</td>
</tr>
<tr>
<td>173.applu</td>
<td>F77</td>
<td>7.9</td>
<td>181</td>
<td>Partial differential equations</td>
</tr>
<tr>
<td>177.mesa</td>
<td>C</td>
<td>81.8</td>
<td>9.5</td>
<td>3D graphics library</td>
</tr>
<tr>
<td>178.galgel</td>
<td>F90</td>
<td>14.1</td>
<td>63</td>
<td>Computational fluid dynamics</td>
</tr>
<tr>
<td>179.art</td>
<td>C</td>
<td>1.2</td>
<td>3.7</td>
<td>Image recognition/neural networks</td>
</tr>
<tr>
<td>183.equake</td>
<td>C</td>
<td>1.2</td>
<td>49</td>
<td>Seismic wave propagation simulation</td>
</tr>
<tr>
<td>187.facerec</td>
<td>F90</td>
<td>2.4</td>
<td>16</td>
<td>Image processing: Face recognition</td>
</tr>
<tr>
<td>188.ammp</td>
<td>C</td>
<td>12.9</td>
<td>26</td>
<td>Computational chemistry</td>
</tr>
<tr>
<td>189.lucas</td>
<td>F90</td>
<td>2.8</td>
<td>142</td>
<td>Number theory/primality testing</td>
</tr>
<tr>
<td>191.fma3d</td>
<td>F90</td>
<td>59.8</td>
<td>103</td>
<td>Finite-element crash simulation</td>
</tr>
<tr>
<td>200.sixtrack</td>
<td>F77</td>
<td>47.1</td>
<td>26</td>
<td>Nuclear physics accelerator design</td>
</tr>
<tr>
<td>301.apsi</td>
<td>F77</td>
<td>6.4</td>
<td>191</td>
<td>Meteorology: Pollutant distribution</td>
</tr>
</tbody>
</table>
Reasons for selecting a program (or not)

- Should candidate program X be part of the benchmark?
  - Yes if:
    - it has many users and solves an interesting problem
    - it exercises hardware resources significantly
    - it is different from other programs in the set
  - No if:
    - it is not a complete application
    - it too difficult to port
    - it performs too much I/O
    - it is too similar to other programs in the set

- These factors are weighed against each other
Some results

- From top to bottom (in each group of 4 machines):
  - Processor clock speed: 500, 500, 533, 500 MHz
  - L1 cache size: 16, 16, 16, 128 KB
  - L3 cache size: 8, 2, 4, 4 MB
Problems of SPEC CPU2000

• Portability
  • It is quite difficult to get all benchmark programs to work on all processors and operating systems
  • SPEC uses 'benchathons': multi-day meetings where engineers cooperate to resolve open problems for the next version of the benchmark

• Which programs go into the benchmark set?
  • Won't one company's SPEC members try to get programs in that favor that company's machines?
  • No, for two reasons:
    1. SPEC is rather cooperative. These are engineers; they value technical merit
    2. The benchmark is too complex to predict what program might benefit my company's next-generation machine more than its competitors
Problems of SPEC CPU2000 (2)

Or: How to shoot yourself in the foot

• Compiler optimizations can break a program's semantics
  • SPEC has to check the results produced for correctness

• Is execution time the right basic measurement?
  • The programs do have small source code differences on various operating systems (in particular for C and C++: #ifdef )
    • library not fully standardized, big-endian vs. little-endian etc.
  • Even identical programs with identical inputs may do different numbers of iterations
    • implementation differences of floating point operations
    • SPEC allows such differences within limits
General benchmarking methodology

- Benchmarking is one of several evaluation methods
- We have now seen a concrete example
  - SPEC CPU2000
- Now let us look at the general methodology
Source

• Literature:
Benchmark parts

A benchmark consists of three main ingredients:

• **Performance measure(s)**
  - As a measure of fitness-for-purpose
  - Measurement is often automatic and usually quantitative, but could also be manual and/or qualitative

• **Task sample**
  - One or several concrete tasks, specified in detail
  - Should be relevant and representative

• **Comparison**
  - Measurement results are collected and compared
  - Provides motivation for using the benchmark
  - Promotes progress
Benchmarking methodology

1. Agree on a performance measure
2. Agree on a benchmarking approach
3. Define the benchmark content
4. Define a benchmarking procedure
5. Define a result report format
6. Package and distribute benchmark
7. Collect and catalog benchmark results
Benchmarks define paradigms

- A scientific benchmark operationalizes a research paradigm
  - Paradigm: Dominant view of a discipline
  - Reflects consensus on what is important
  - Immature fields cannot agree on benchmarks

- A commercial benchmark (such as SPEC) reflects a mainstream
Why are benchmarks helpful?

- Technical factors
  - Easy-to-understand and easy-to-use technique
  - High amount of control
  - Support replication of findings, hence credibility

- Sociological factors
  - Focus attention to what is (considered) important
  - Define implicit rules for conducting research
    - hence promote collaboration among researchers
    - help create a community with common interest
  - Promote openness
    - force the dirty details into the open
    - make hiding flaws difficult
Problems with benchmarks

- **Cost**
  - Designing, composing, implementing, and packaging a benchmark is a very work-intensive task
    - Can only be done by a significant group of experts; takes long

- **Task composition**
  - Agreeing on what exactly goes into a benchmark task is difficult:
    - different players may have different foci of interest
    - different players may want to emphasize their own strengths
    - real-world usage profiles are usually unkown

- **Overfitting**
  - If the same benchmark task is used too long, the systems will adapt to it too specifically
    - benchmark performance will increase although real performance does not
Quality attributes of good benchmarks

- **Accessibility**
  - should be publicly available and easy to obtain
- **Affordability**
  - effort required for executing benchmark must be adequate
- **Clarity**
  - specification must be unambiguous
- **Portability, Scalability**
  - must be easily applicable to different objects under study
- **Relevance**
  - task must be representative of real world
- **Solvability** (relevant for methods benchmarks)
  - objects under study must be able to "succeed"
A short benchmark example

- Image Segmentation benchmark
  - http://research.microsoft.com/vision/cambridge/segmentation/
  - Given a picture, the user marks known foreground (white), and possible foreground (gray)
  - Segmentation algorithm tries to extract exactly all foreground
  - Result is compared against "ground truth"

Example 2: TREC

- **Text Retrieval Conference**
  - annually since 1992
  - Topic: *Information Retrieval* of text documents
    - Given large set of documents and query, find all documents relevant to the query and no others (like a web search engine)
    - Documents are ranked by perceived relevance
    - Performance measures:
      - **Precision**: Fraction of retrieved documents that are relevant
      - **Recall**: Fraction of relevant documents that are retrieved
  - Core activity is comparing results (and approaches for getting them) on pre-defined tasks used by the participants

- TREC now has many different *tasks*
  - each of them is a separate benchmark
  - we will look at only one of them: "Ad-hoc retrieval"
Sources


- Ellen M. Voorhees, Donna Harman: "Overview of the Eighth Text REtrieval Conference (TREC-8)", 1999
TREC "Ad hoc retrieval" task

- started at TREC-1 (1992), used through TREC-8 (1999)
  - then discontinued because performance had leveled off
  - 1992 TREC-1 had 2 tasks, 2005 TREC-14 had 15 tasks
- Corpus contained 740 000 news articles in 1992
  - had grown to 1.5 Mio (2.2 GB) by 1998

Benchmark composition:
- 50 different query classes (called 'topics') are used
  - and changed each year
- Performance measures are Precision and Recall
- Comparison is done at the conference
An example 'topic definition'

- From TREC-8 (1999)

```
<num> Number: 409
<title> legal, Pan Am, 103
<desc> Description:
What legal actions have resulted from the destruction of Pan Am Flight 103 over Lockerbie, Scotland, on December 21, 1988?
<narr> Narrative:
Documents describing any charges, claims, or fines presented to or imposed by any court or tribunal are relevant, but documents that discuss charges made in diplomatic jousting are not relevant.
```

- earlier topic definitions were more detailed
TREC procedure

- Dozens of research groups from universities and companies participate:
  - run all 50 queries through their system
    - conversion from topic definition to query can be automatic or manual → two separate performance comparisons
  - submit raw retrieval results
  - conference organizers evaluate results and compile performance statistics
  - at the conference, performance of each group is known
  - presentations explain the techniques used
Results (TREC-8, automatic query formulation)
Results (TREC-8, manual query formulation)
Year-to-year improvement levels off

- Results for only 1 system (SMART), but would be similar for most others
Problem: How to judge query results

• How can anyone possibly know which of 1.5 Mio documents are relevant for any one query?
  • necessary for computing recall

• TREC procedure:
  • For each query, take the results of a subset of all participants
  • Take the top 100 highest ranked outputs from each
    • e.g. TREC-8: 7100 outputs from 71 systems
  • Merge them into the candidate set
    • e.g. TREC-8: 1736 unique documents
  • Have human assessors judge relevance of each document
  • Overall, consider only those documents relevant that were (a) in this set and (b) were judged relevant by the assessor
    • e.g. TREC-8: 94 relevant documents

• (What are the problems with this procedure?)
Relevance judgement problems

- Human assessors make errors
  - This is bad for all participants who (at those points) do not

- There are often many more relevant documents in the corpus beyond the candidate set
  - The procedure will consider them all irrelevant

- This is bad for participants who did not contribute to the candidate set and
  - find documents of a different nature than the contributors or
  - rank relevance different than the contributors

- How could TREC evaluate how serious this problem is?
Precision decrease for system A when hits unique to system A are left out
Summary

- Benchmarks consist of a performance measure, a task, and direct comparison of different results
  - Selecting tasks (and sometimes measures) is not straightforward!
- They apply to classical performance fields such as hardware, to capabilities of intelligent software (e.g. TREC), or even to methods to be applied by human beings
  - Measurement in a benchmark may even have subjective components
  - Even benchmarks can have credibility problems
- Putting together a benchmark is difficult, costly, and usually produces disputes over the task composition
- A good benchmark is a powerful and cost-effective evaluation tool
Further literature

- IEEE Computer 38(2), February 2003
  - special issue on workloads for computer systems
    (simulation, benchmarking, architecture design etc.)

- Web search for other benchmarks, such as
  - TPC, ECperf, SPECweb

- Related approach: RoboCup
  - Robot performance cannot be quantified, so use direct games and tournaments instead
  - Likewise, there are championships for other game-playing programs (e.g. Chess, Go)
Thank you!