Comparing Benchmarks Using Key Microarchitecture-Independent Characteristics

IISWC 2006
October 26th 2006
San Jose, California, US

Kenneth Hoste and Lieven Eeckhout
ELIS, Ghent University, Belgium
Comparing benchmarks is easy… or is it?

Hardware performance counters are a popular tool to compare emerging workloads with established benchmark suites.

examples:

- BioInfoMark (vs. SPECint CPU2000)
  Workload Characterization of Bioinformatics Applications
  (Li et.al, MASCOTS 2005)

- BioMetricsWorkload (vs. SPECint CPU2000)
  Workload Characterization of Biometrics Applications on Pentium4 Microarchitecture
  (Cho et.al., IISWC 2005)

- ...

How reliable are these metrics?
How can we catch true inherent program behavior?
Beware of the pitfall!

Hardware performance counters

- instruction per cycle (IPC)
- branch misprediction rate
- L1 D-cache and I-cache miss rate
- L2 cache miss rate
- D-TLB miss rate

- measure native execution of benchmarks ⇒ fast
- no need to instrument code or implement analysis
- expose performance bottlenecks

BUT:
- true inherent program behavior may be hidden, which can be misleading
### How to avoid the pitfall

**microarchitecture-independent characteristics**

<table>
<thead>
<tr>
<th>Instruction mix</th>
<th>Instruction-level parallelism (ILP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Register traffic</td>
<td>(data and instr.) working set size</td>
</tr>
<tr>
<td>Data stream strides</td>
<td>Branch predictability (PPM)</td>
</tr>
</tbody>
</table>

- are able to catch true inherent program behavior
- independent of the microarchitecture
  - (cache configuration, issue width, # functional units, ...)

**BUT:**
- more time needed to measure them (time-consuming profiling)

---

Comparing Benchmarks Using Microarchitecture-Independent Characteristics - Kenneth Hoste - 2006-10-26
Faculty of Engineering - Department of Electronics and Information Systems (ELIS) - Ghent University
Quantifying the pitfall

20% of maximum distance

41.1% of all benchmark pairs
Comparing benchmarks: a case study

comparing bzip2 (SPEC CPU2000) with blast (BioInfoMark)

—Comparing benchmarks using microarchitecture-independent characteristics—

minor difference in branch mispred. rate

⇒ these benchmarks are quite similar

microarchitecture-independent characteristics

various differences noticeable (working set sizes!)

⇒ these benchmarks are quite different

hardware performance counters
Efficiently comparing benchmarks

measuring microarchitecture-independent characteristics takes more time

on Alpha:

110 machine-days (instrumentation using ATOM)

vs

4 machine-days (dcpi on Alpha 21164/21264A)

Problem

How can we limit the time needed to characterize benchmarks?

Solution

limit the number of characteristics without losing too much information

How?

exploit correlation between characteristics (2 techniques)
Eliminating correlation between characteristics

identify the pair of characteristics with the highest correlation, and drop one characteristic

for example:

- data work.set (block level) & data work. set (page level) => 97.96% correlation
- instr. work.set (block level) & instr. work. set (page level) => 97.70% correlation
- ILP (win.size=256) & ILP (win.size=128) => 97.40% correlation
- ILP (win.size=64) & ILP (win.size=32) => 96.75% correlation
- global store stride (prob. < 4096) & global store stride (prob. < 512) => 96.73% correlation
- ILP (win.size=128) & ILP (win.size=32) => 96.60% correlation
Finding the optimal set of characteristics

learn how to retain maximum correlation with the full set of characteristics with as few characteristics as possible

using a genetic algorithm:

1) start with a random population of subsets of characteristics
2) score each subset with a fitness score

\[ f = \rho (1 - \frac{n}{N}) \]

\( f \): correlation with full set
\( n \): number of characteristics in subset
\( N \): total number of characteristics

3) fittest subsets produce offsprings (using crossover and mutation)
4) repeat step 2 and 3 for subsequent generations
*) search stops when solutions converge, or when a maximum number of generations is reached
What about PCA?

Principal Components Analysis (PCA) is often used to obtain uncorrelated characteristics from a given set.

Each Principal Component = linear combination of characteristics

\[ PC_i = \sum_j w_{ij} c_j \]

Hence, we still need to measure all characteristics in order to obtain uncorrelated principal components.

Pcs are hard to interpret in terms of original program characteristics.
Which subset of characteristics is optimal?

110 machine-days
37 machine-days

47 characteristics
8 characteristics

1) percentage loads
2) average number of input operands
3) prob. register dependency distance ≤ 8
4) prob. local load stride ≤ 64
5) prob. global load stride ≤ 512
6) prob. local store stride ≤ 4096
7) D-stream working set size (4KB page level)
8) ILP (256-entry window)
Comparing existing benchmark suites

using this optimal subset of characteristics, we compare:

✦ 6 benchmark suites
  BioInfoMark, BioMetricsWorkload, CommBench, MediaBench, MiBench, SPEC CPU2000
✦ 122 benchmarks

clustering of the benchmarks based on the subset of characteristics is done using k-means clustering
Different inputs yield different behavior... or not

Different inputs for phylip (BioInfoMark) & mpeg2 (MediaBench) yield quite different behavior.

Different inputs for gzip (SPEC CPU2000) yield quite similar behavior.

- prob. register dependence ≤ 8
- prob. local load stride ≤ 64
- prob. global load stride ≤ 512
- prob. local store stride ≤ 4096
- avg. number of inputs operands
- percentage loads
- ILP (256-entry window)
- working set size (D-stream at the 4KB page level)
Some benchmarks are quite unique

- gcc (5)
- mcf
- blast
- csuBayesian (2)
- csuSubspaceProject (2)
- csuSubspaceTrain (2)
- art (2)

- prob. register dependence ≤ 8
- prob. local load stride ≤ 64
- prob. global load stride ≤ 512
- prob. local store stride ≤ 4096
- avg. number of inputs operands
- percentage loads
- ILP (256-entry window)
- working set size (D-stream at the 4KB page level)
Others are very similar to each other

- gzip (5)
- jpeg.cjpeg
- FFT.fft
- FFT.fftinv
- ghostscript (MediaBench)
- ispell
- bzip2 (3)
- dijkstra

- prob. register dependence ≤ 8
- prob. local load stride ≤ 64
- prob. global load stride ≤ 512
- prob. local store stride ≤ 4096
- avg. number of inputs operands
- percentage loads
- ILP (256-entry window)
- working set size (D-stream at the 4KB page level)
Don’t be fooled: bzip2 vs blast
Interesting observations on benchmark suites

- 9 SPECfp benchmarks are isolated in a single cluster

- various recently introduced benchmarks exhibit dissimilar behavior compared to SPEC CPU2000
  - blast, fasta, hmmer, phylip.promlk (*BiolInfoMark*)
  - csu (*BioMetricsWorkload*)

  ⇒ important to take into account!

- most MediaBench and MiBench benchmarks are similar to SPEC CPU2000 benchmarks
Conclusions

✦ using microarchitecture-dependent metrics might be misleading
✦ microarchitecture-independent metrics are a solution, but take longer to measure
✦ using a genetic algorithm, we limited the number of characteristics to measure from 47 to 8
✦ comparison of 122 workloads from 6 benchmark suites yields various interesting results
Questions?