What the Means Mean: Benchmarking and Experimental Computer Architecture

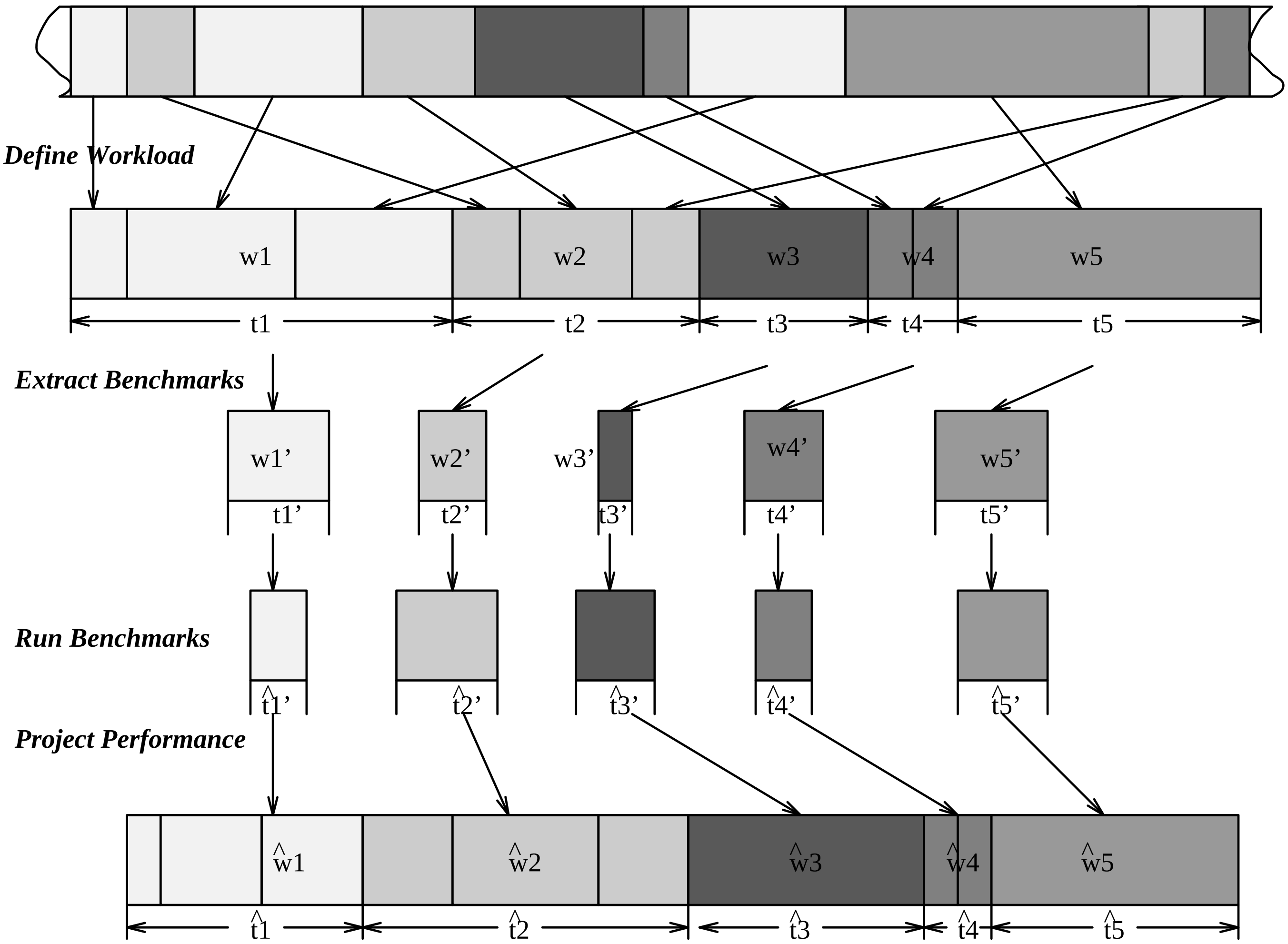
J. E. Smith

Oct. 13, 1994

Outline

- Classical Benchmarking
- What the Means Mean
- Experimental Computer Architecture
- Issues and Pitfalls

The "Classical" Benchmarking Process



The "Classical" Benchmarking Process (contd)

Steps:

1) Extract Benchmarks from applications

2) Choose performance metric

3) Execute benchmarks on candidate machines

4) Project performance in new machine environment
and summarize results

Extracting Benchmarks

$W = \sum(w_i)$: total work done

$W' = \sum(w'_i)$: total work in benchmarks

$f_i = w_i / W$: fraction of work from each job type

$f'_i = w'_i / W'$ fraction of work from each benchmark

$m_i = w_i / w'_i = f_i W / f'_i W'$

Perfect Scaling assumption: $t'_i / t_i = w'_i / w_i$

Projecting Performance

- Constant work model: $\hat{w}_i = w_i$ for all i

  What is projected time? $\hat{T}$

  $\hat{T} = \sum(\hat{t}_i) = \sum(\hat{t'}_i m_i)$ (assuming perfect scaling)

- Constant time model: $\hat{t}_i = t_i$ for all i

  What is projected work? $\hat{W}$

  $\hat{W} = \sum(\hat{w}_i) = \sum(w_i t'_i / \hat{t'}_i)$ (assuming perfect scaling)

Performance Measured as a Rate

rate = work/time $r_i = w_i / t_i$.

- For constant work model;

  $\hat{R} = \sum(\hat{w}_i)/\sum(\hat{t}_i) = \sum(Wf_i)/\sum(Wf_{i/\hat{r}'}i) = 1/\sum(f_i/\hat{r}'_i)$.

  (i.e. weighted harmonic mean rate)

- For constant time model;

  $\hat{R} = \sum(\hat{w}_i\hat{/}t_i) = \sum(\hat{r}'_i t_i)/\sum(t_i)$

  Because the t_i are fixed, this is essentially a weighted arithmetic mean.

  What about geometric mean (used by SPEC)?
  It has no basis in reality ...i.e. it follows no realistic projection rule. It only has a nice mathematical property.

Experimental Benchmarking

Steps:

1) Come up with clever implementation or architecture idea.

2) Write/modify simulator.

3) Compile (if new architecture, first devise compiler)

4) Select benchmarks

5) Simulate with benchmark set.

6) Choose performance metric

7) Summarize results showing your clever idea is wonderful.

Step 1: Clever idea

I can’t help with this...gets harder every day

Step 2: Write/modify simulator

- Model at a level of detail sufficient to give accurate results,
- But not too accurate so that the simulator
  is overly complicated.

Step 3: Compile

- A major problem with many experimental studies.

- If an existing ISA is used, an existing compiler may work, otherwise factoring out compiler dependences can be very difficult.

Step 4: Select Benchmarks

QUESTION 1: What is the application space?

This step has many pitfalls

Pitfall: covering too many application spaces Does it make sense to simultaneously cover more than one?

	application set 1				application set 2				
	#1	#2	#3	#4	#5	#6	#7	#8	hmean
Comp 1:	4	5	3	7	5	3	4	6	4.3
Comp 2:	1	2	1	2	8	7	8	9	2.3
Comp 3:	8	9	7	8	2	2	1	1	2.3

(performance given as a rate)

Even though computer 1 has the best performance overall...it is best for neither application area.

Step 4: Pitfalls (contd.)

- Choosing benchmarks from the wrong application space
  example: In a realtime paper, choosing troff as a benchmark
  (most recent ASPLOS)

- Choosing benchmarks from no application space
  synthetic workloads
  example: dhrystone

  model-generated workloads
  these may be useful in some instances
  but have a bad name due to abuse.

- Using toy benchmarks
  example: used to prove the value of RISC in early 80s

Step 4: Pitfalls (contd.)

- Missmatch of benchmark properties with scale of features being studied.
  example:
  using SPEC benchmarks for cache studies
  (memory requirements too small)

- Carelessly scaling benchmarks
  (e.g. to get reasonable runtime)
  example: truncating benchmarks; especially taking only
  first part of a trace. (commonly done)
  example: reducing program data size...
  can cause setup code to have too much importance.

- Including only "easy" cases
  may not demonstrate true value of clever idea

- Including no "easy" cases
  may exaggerate importance of clever idea

Step 5: Simulate with Benchmark Set

Check results for reasonableness ...and re-check
then re-check after every simulator modification.

Step 6: Choose Performance Metric

- Use the correct measure.
  example:
  for cache studies, miss rates are often of little value
  summarize overall performance and/or memory traffic.

- Use the correct mean.
  for example, don’t use geometric mean miss rates).

Step 7: Summarize Results

Explain the results, don’t just let the numbers speak for themselves.

Conclusion:

Keep the benchmark model in mind, and

Always ask yourself: "exactly what is it I’m trying to show?"