Evaluating whether the training data provided for profile feedback is a realistic control flow for the real workload.

Darryl Gove, Lawrence Spracklen, John Henning
Sun Microsystems Inc.

Outline

- The trouble with feedback
- Correspondence Values
- Coverage
- Concluding remarks

The trouble with feedback

- Profile feedback uses a training run of the code
- At minimum improves decisions about:
  - > Basic block layout
  - > Inlining of routines
- But....

The trouble with feedback

- Profile feedback uses a training run of the code
- At minimum improves decisions about:
  - > Basic block layout
  - > Inlining of routines
- But....
  - > It takes twice as long to build
  - > Requires 'representative' training data

The trouble with feedback

- Profile feedback uses a training run of the code
- At minimum improves decisions about:
  - Basic block layout
  - Inlining of routines
- But....
  - ~~It takes twice as long to build~~
  - Requires 'representative' training data

Out of scope

But what does this really mean?

Method

- Using SPEC CPU2000 benchmark suite
- Checking to see how well the training workloads match the reference workloads
- Benchmarks compiled with low optimisation
- Instrumented to gather data on basic block counts and whether particular branches are taken or not
- Multiple training (or reference) datasets added together to give a single training (or reference) workload.

What can't be used

- For SPEC CPU profile is used on multiple platforms
- Each platform may do different optimisations
- So performance cannot be used as a test for representative data sets
- If Platform A gets faster with profile feedback it does not imply that Platform B also will.
- And similarly if Platform A derives no benefit.
- So metrics have to be derived from platform agnostic metrics (if possible)

Static and dynamic branches

- A **static branch** is a branch instruction that exists in the code.
- A **dynamic branch** is one that occurs at runtime.
- Hence one static branch can contribute many dynamic branches.

A representative workload is....

A representative training workload is one for which each static branch is either:

- usually taken by both the training and reference workloads,

  or

- usually untaken by both of them.

Correspondence Value

- Correspondence Value for a benchmark
- Total number of correctly predicted dynamic branches divided by the total number of dynamic branches
- Ranges from zero (no branches correctly predicted)
- To 100% (meaning all branches correctly predicted)

$$CV = \frac{\sum_{branches} (Frequency_{branch} * (TakenTrain_{branch} \equiv TakenRef_{branch}))}{\sum_{branches} Frequency_{branch}}$$

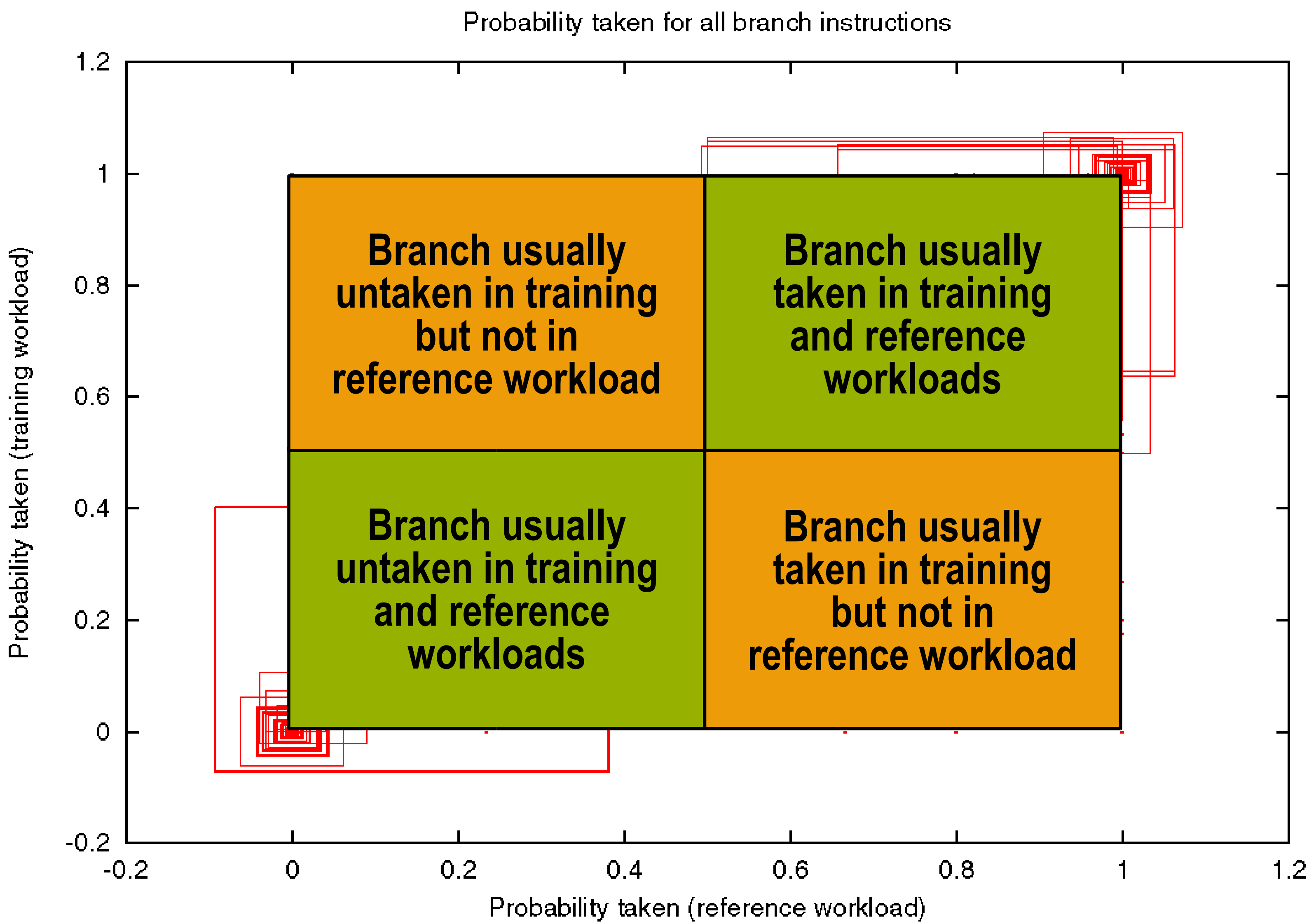
Correspondence Values for CPU2000

CPU2000_INT Benchmark	Correspondence between train and reference	CPU2000_FP Benchmark	Correspondence between train and reference
164.gzip	100%	168.wupwise	100%
175.vpr	100%	171.swim	100%
176.gcc	98%	172.mgrid	98%
181.mcf	100%	173.applu	100%
186.crafty	96%	177.mesa	96%
197.parser	99%	178.galgel	83%
252.eon	100%	179.art	100%
253.perlbmk	95%	183.equake	100%
254.gap	95%	187.facerec	100%
255.vortex	100%	188.ammp	100%
256.bzip2	96%	189.lucas	89%
300.twolf	100%	191.fma3d	100%
		200.sixtrack	100%
		301.apsi	72%

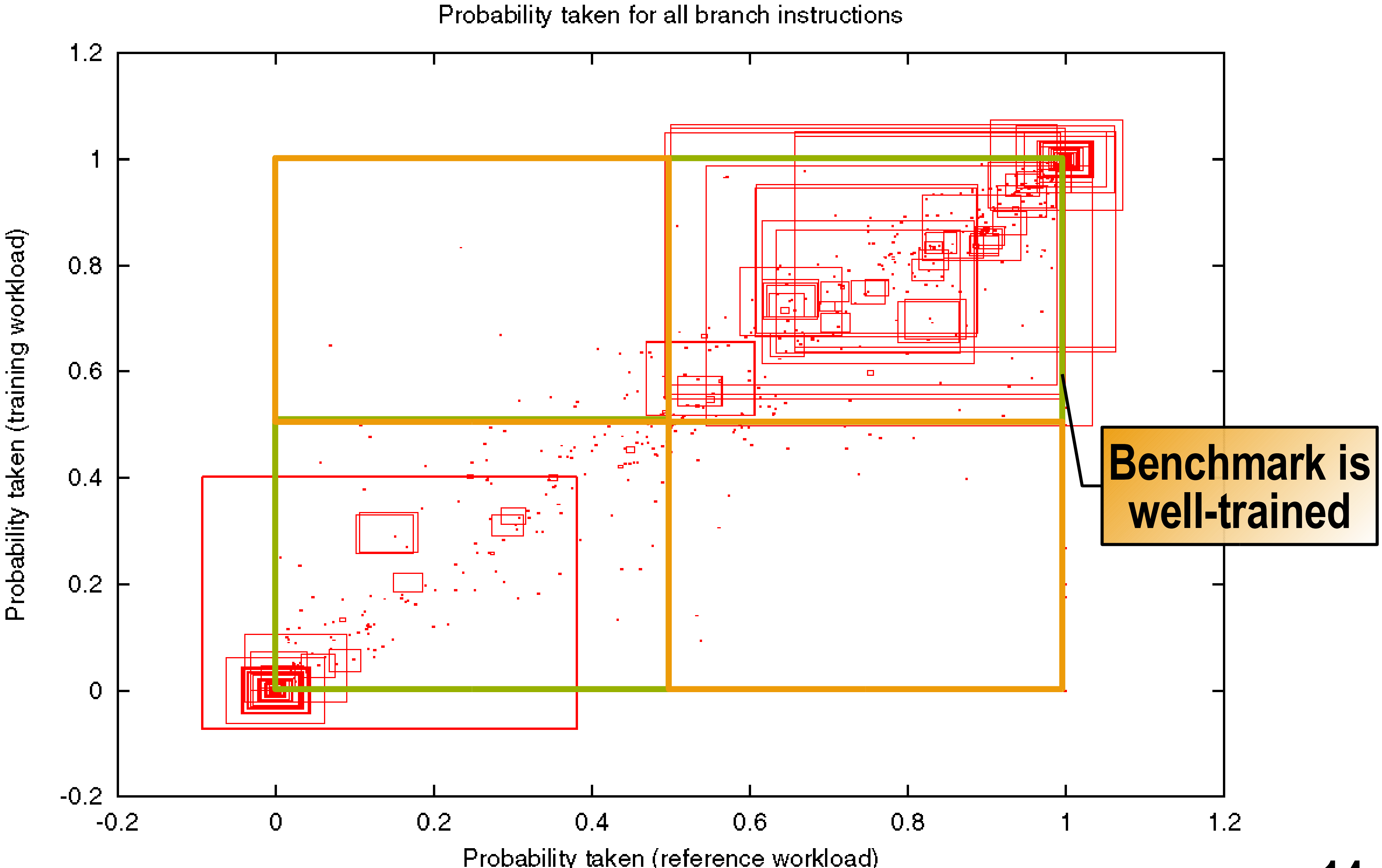
Visualising Correspondence Values

- Results are easier to understand as graphs
- x-axis is probability taken in reference workload
- y-axis is probability taken in training workload
- Size of mark is proportional to frequency encountered (ie taken or untaken) in reference workload.

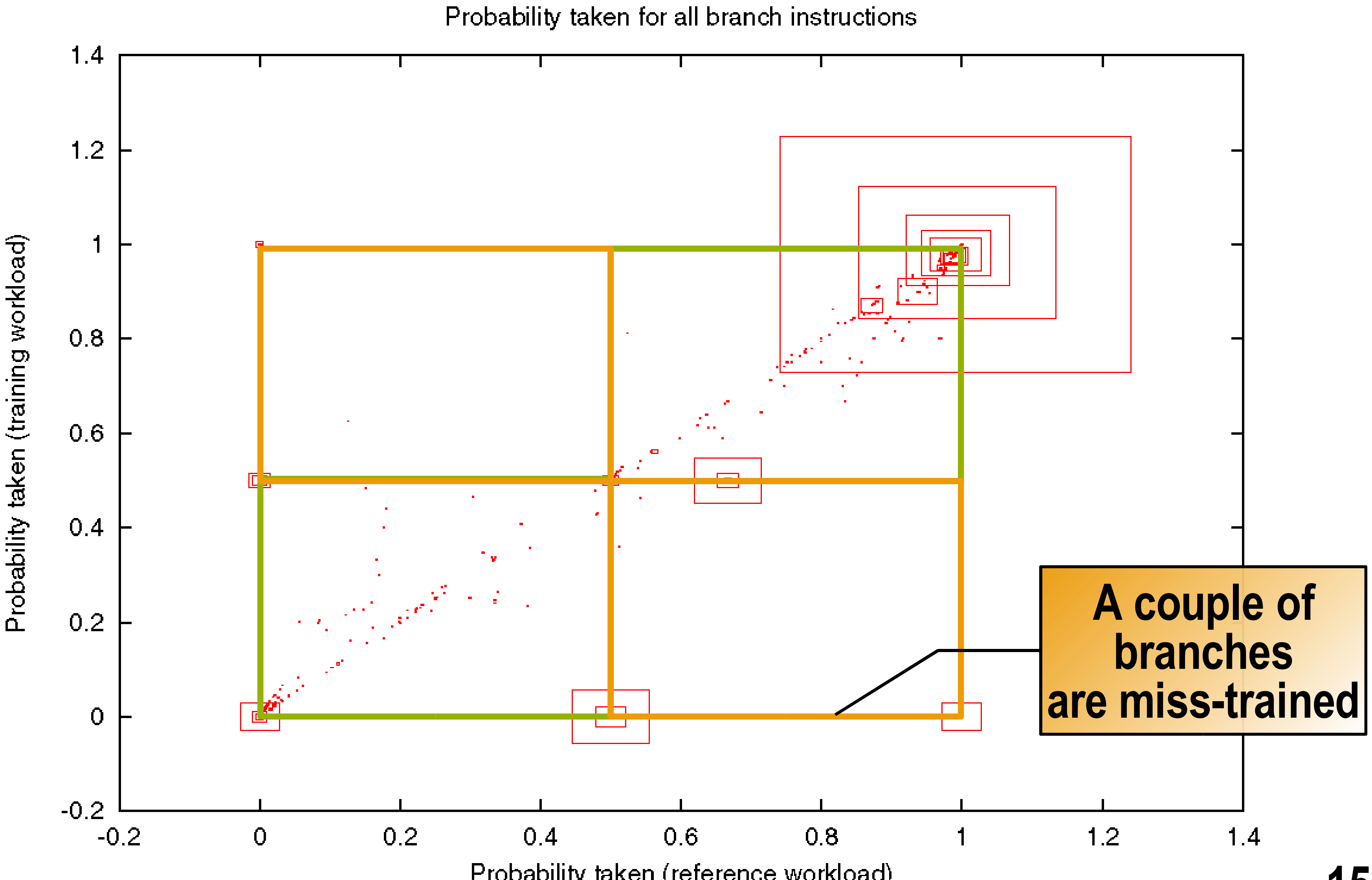
Visualised Correspondence Value



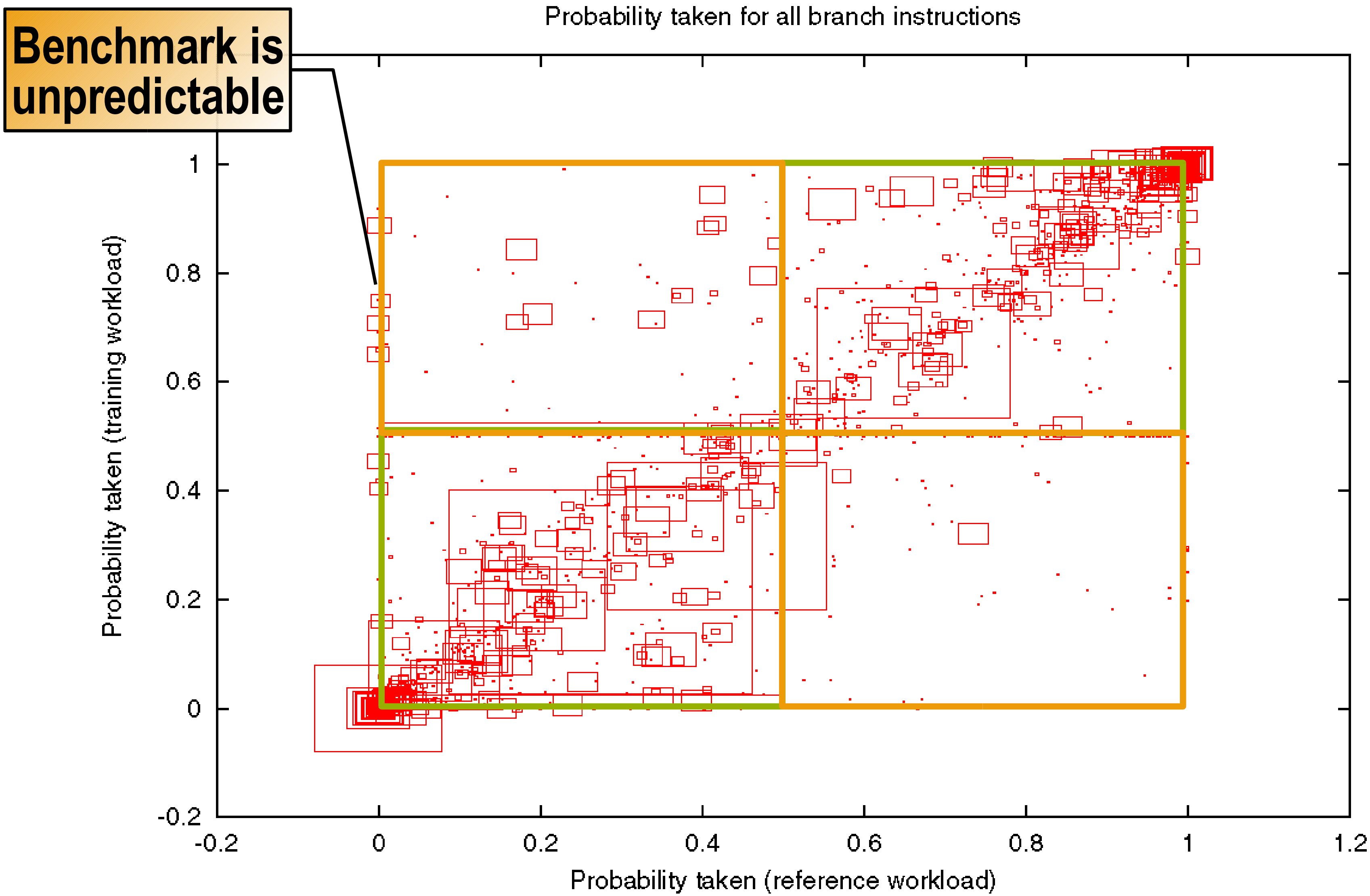
300.twolf (CV=100%)



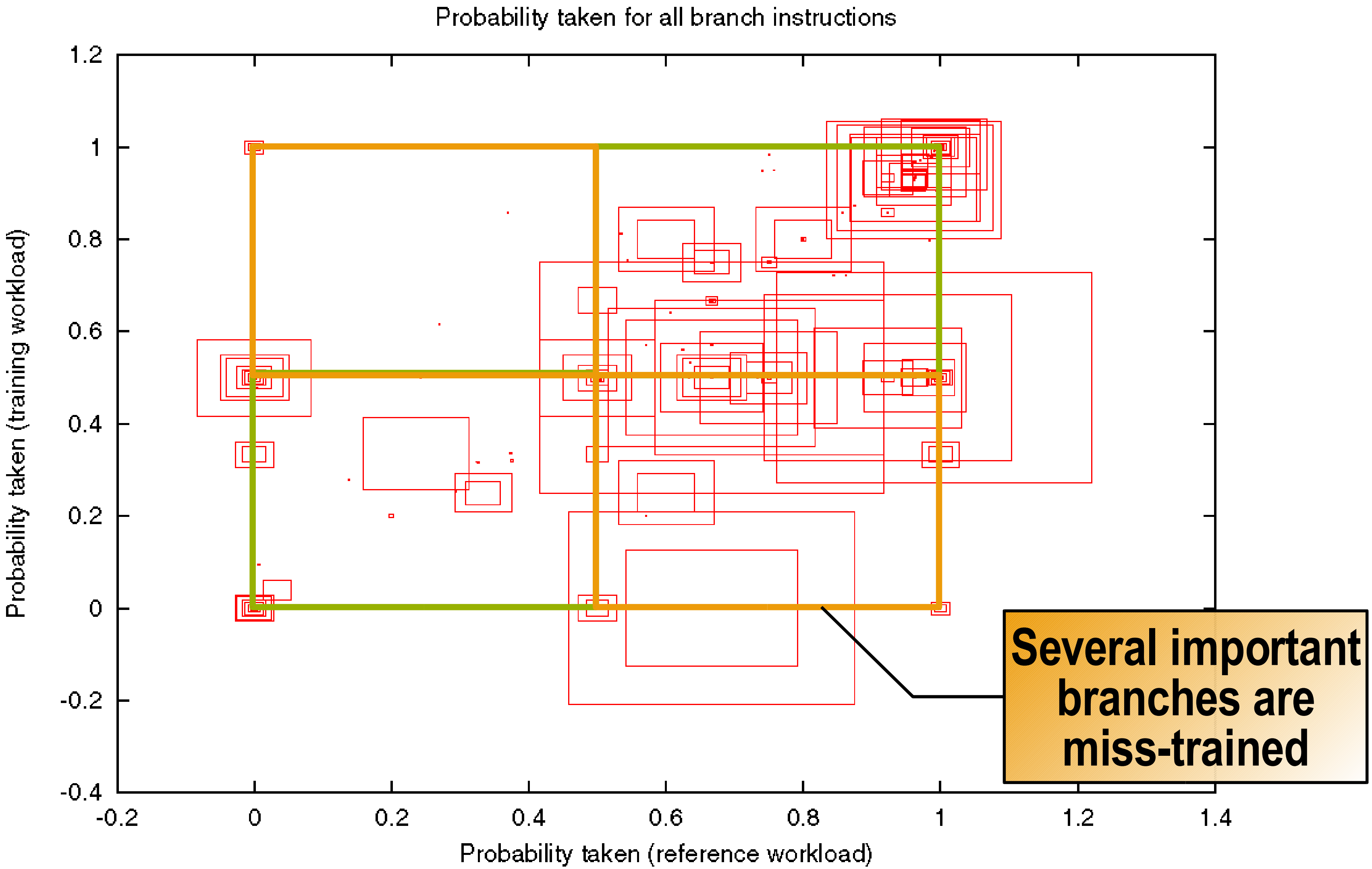
178.galgel (CV=83%)



186.crafty (CV=96%)



301.apsi (CV=72%)



Coverage data

- However, one branch instruction might have multiple targets.
- Data per branch is not easily accessible.
- Basic block counts are more easily accessible, and unique.

Coverage data

- However, one branch instruction might have multiple targets.
- Data per branch is not easily accessible.
- Basic block counts are more easily accessible, and unique.
- However,
  - Some basic block counts scale with runtime (eg inner loop)
  - Some basic block counts are constant for all runtimes (eg initialisation code)

A representative workload is...

- A representative training workload will exercise all the critical basic blocks of the reference workload.

Coverage definition

- The coverage is the
- Sum of the dynamic basic block counts for the reference workload that are also executed by the training workload
- Divided by the sum of all the dynamic basic block counts for the reference workload.

$$coverage = \frac{\sum_{blocks} FrequencyRef_{block} * (FrequencyTrain_{block} > 0)}{\sum_{blocks} FrequencyRef_{block}}$$

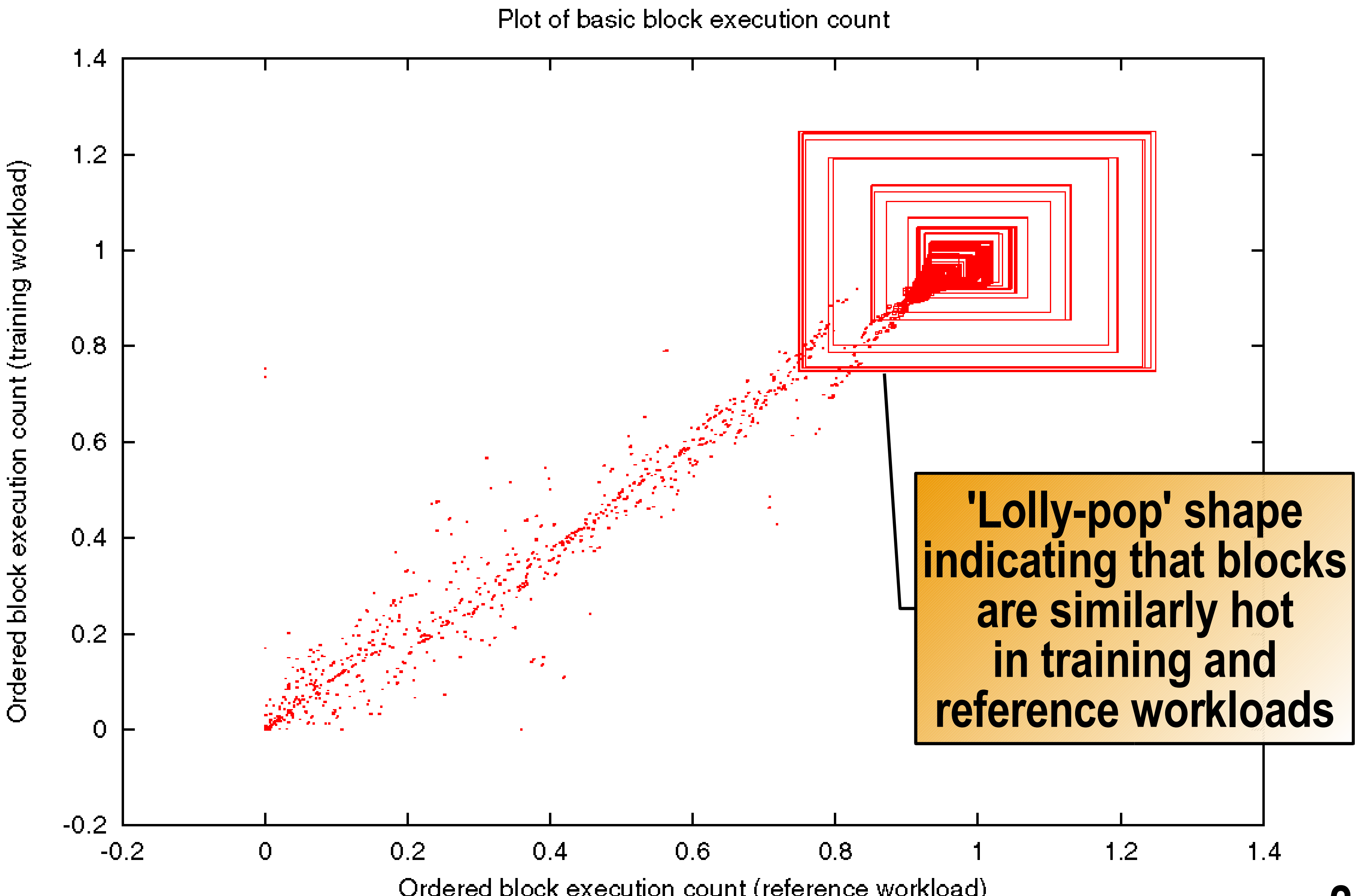
Coverage CPU2000

CPU2000_INT Benchmark	Coverage	CPU2000_FP Benchmark	Coverage
164.gzip	100%	168.wupwise	100%
175.vpr	100%	171.swim	100%
176.gcc	100%	172.mgrid	100%
181.mcf	100%	173.applu	100%
186.crafty	100%	177.mesa	98%
197.parser	100%	178.galgel	85%
252.eon	100%	179.art	100%
253.perlbmk	100%	183.equake	100%
254.gap	99%	187.facerec	100%
255.vortex	100%	188.ammp	100%
256.bzip2	100%	189.lucas	81%
300.twolf	100%	191.fma3d	100%
		200.sixtrack	100%
		301.apsi	37%

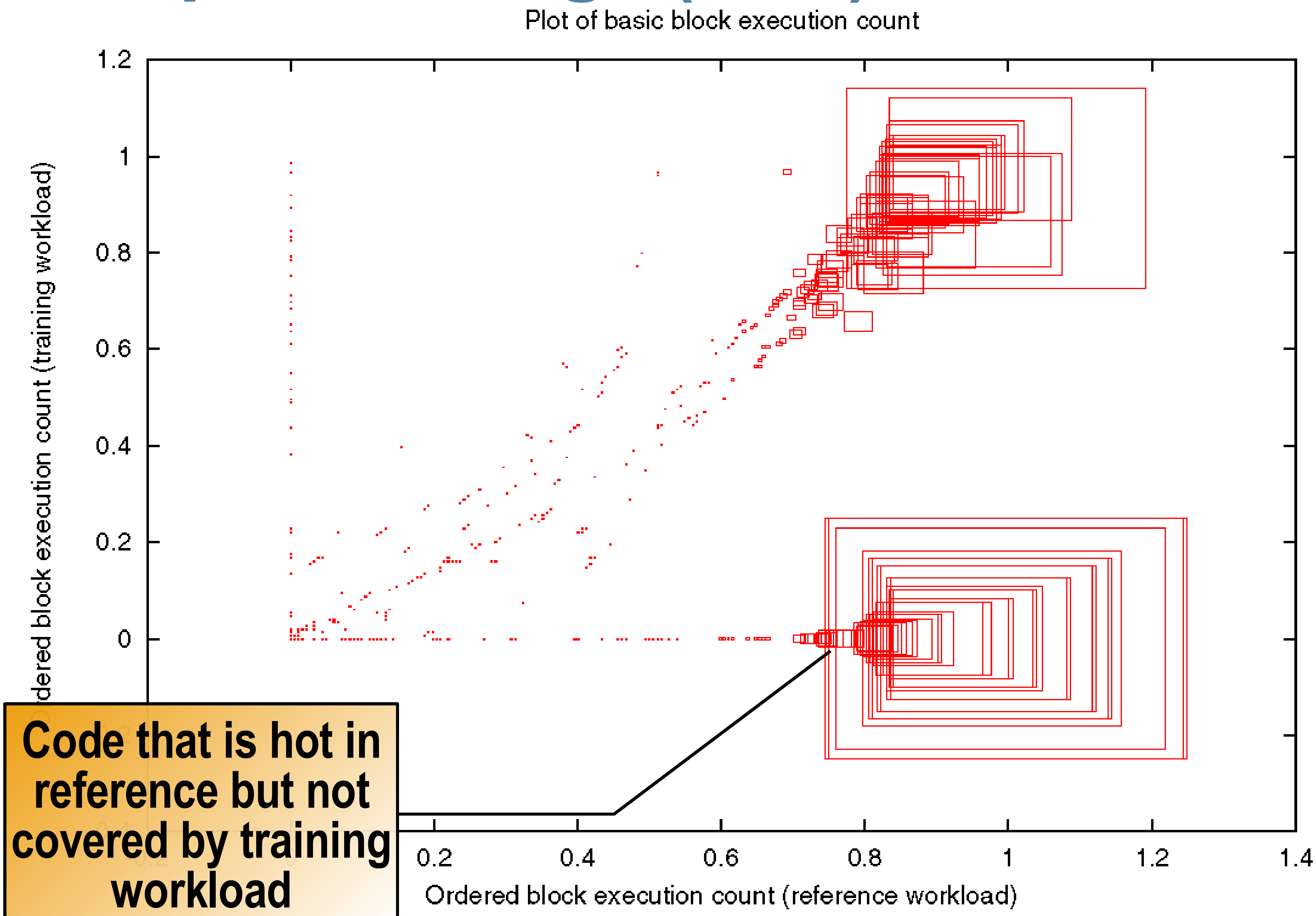
Visualising coverage

- Sort the blocks in order of increasing execution count
- x-axis is sorted basic block count for reference
- y-axis is sorted basic block count for train
- Size of mark is proportional to the execution count for the reference workload

300.twolf coverage (100%)



301.apsi coverage (37%)



Concluding remarks

- Coverage is easy to calculate, and provides a low-bar for representative training workloads.
- *If a block is not covered it cannot have been trained*
- Correspondence Value calculations are a more detailed approach.
- As can be seen from the apsi results, both approaches are complementary.
- Using these calculations it is possible to evaluate whether the current training workloads are sufficient for code path optimisations.

Evaluating whether the training data provided for profile feedback is a realistic control flow for the real workload.

John Henning
John.Henning@sun.com