PinPoints: Simulation Region Selection with PinPlay and Sniper

Harish Patil, T. Mack Stallcup

Intel Corporation

With contributions from:

Wim Heirman (Intel), Trevor Carlson (Ghent University)

ISCA 2014 Tutorial T7 June 15, 2014

Representative simulation point selection

1. How to select and checkpoint representative regions for simulation
2. How to use checkpoints with Sniper for simulation and projection

Objective: Tools and techniques for representative simulation region selection

Schedule & house rules

8:45 – 9:30 Intro + Background (Harish)

9:30 – 10 Demo Part I (Mack)

10 – 10:30 Break

10:30 – 11:15 Demo Part II (Mack)

11:15 – 11:45 Advanced Topics (Harish)

11:45 – noon Wrap-up + Q&A (all)

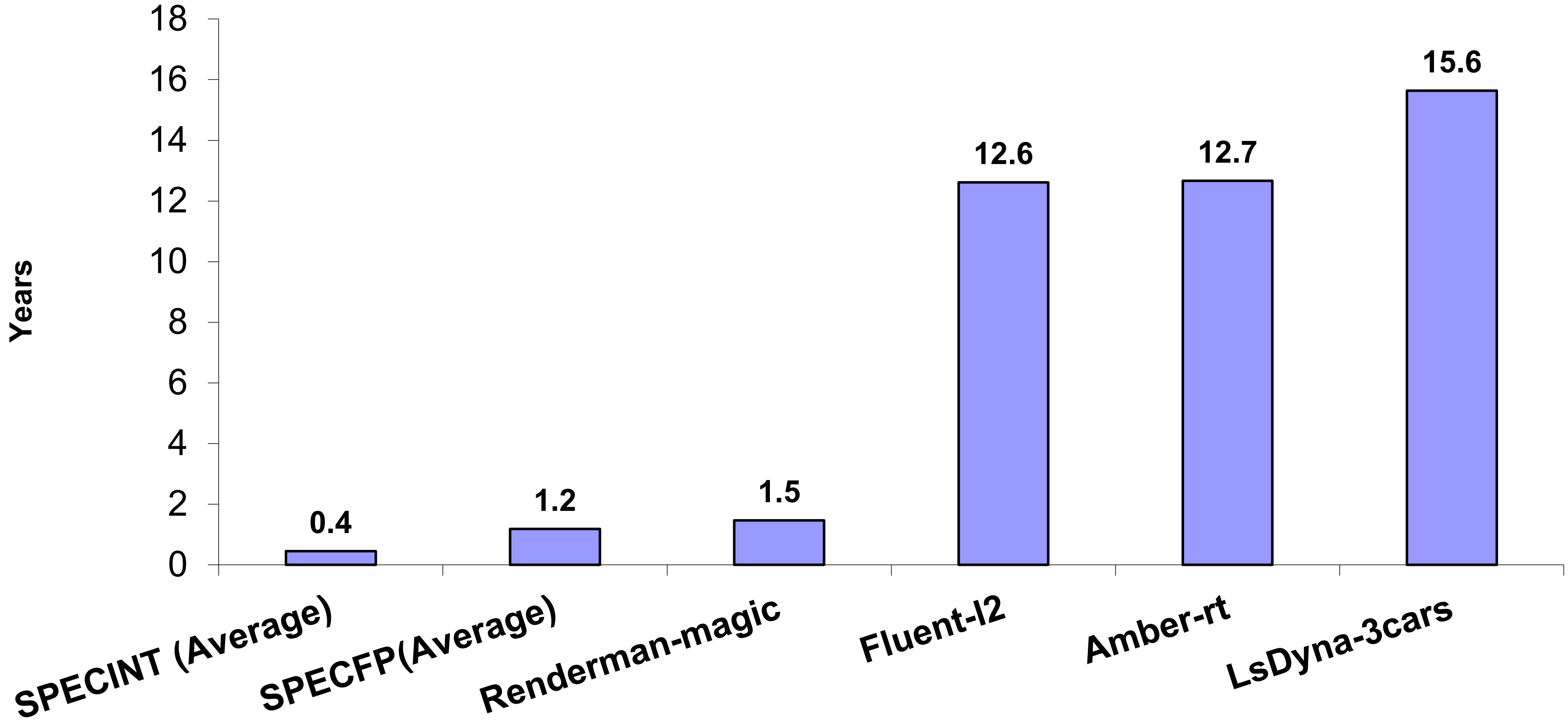
Ask questions --- or we will ☺

Harish.patil@intel.com t.mack.Stallcup@intel.com

Why simulation region selection?

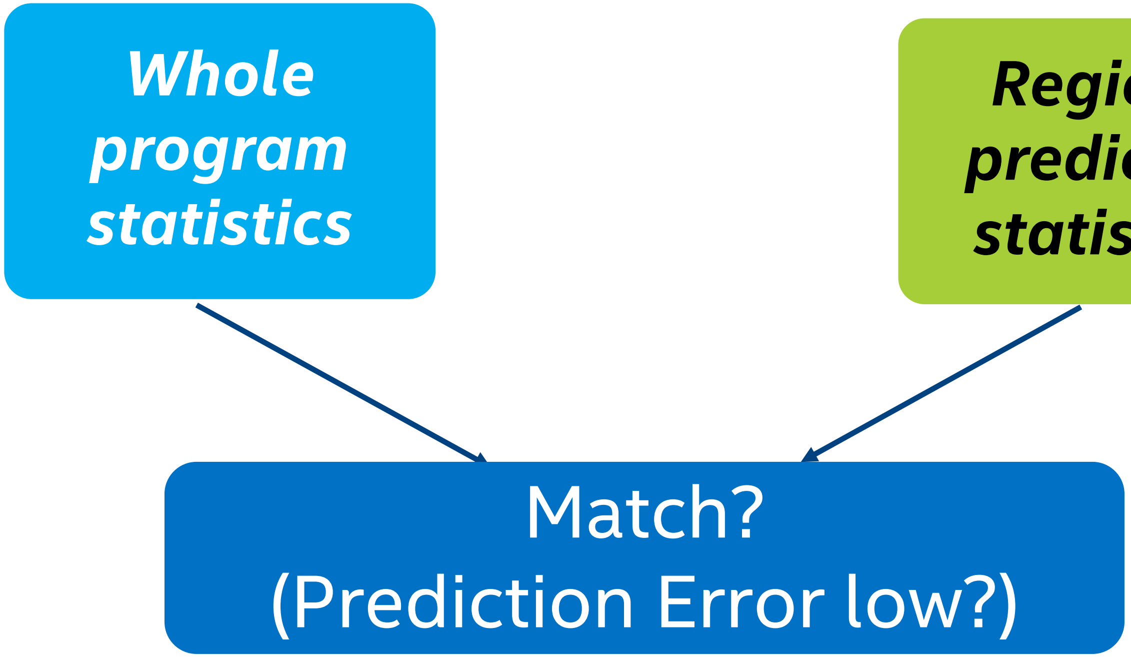
Complex Processors / Slow Simulation

Simulation Time in YEARS
@ 10,000 Instructions/Second



Whole-program simulation is very slow!

Representative or not?

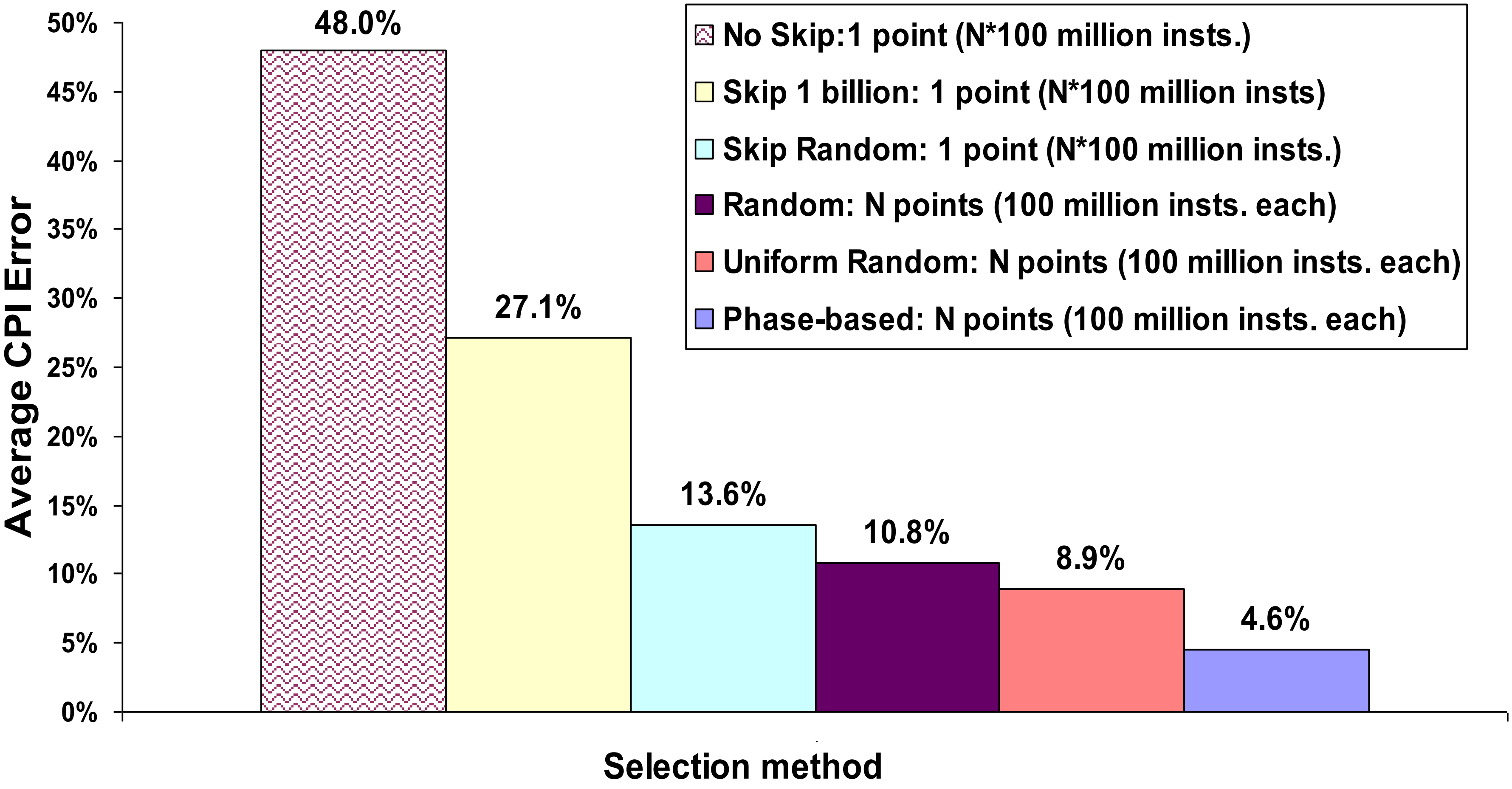


Whole program statistics

Region-predicted statistics

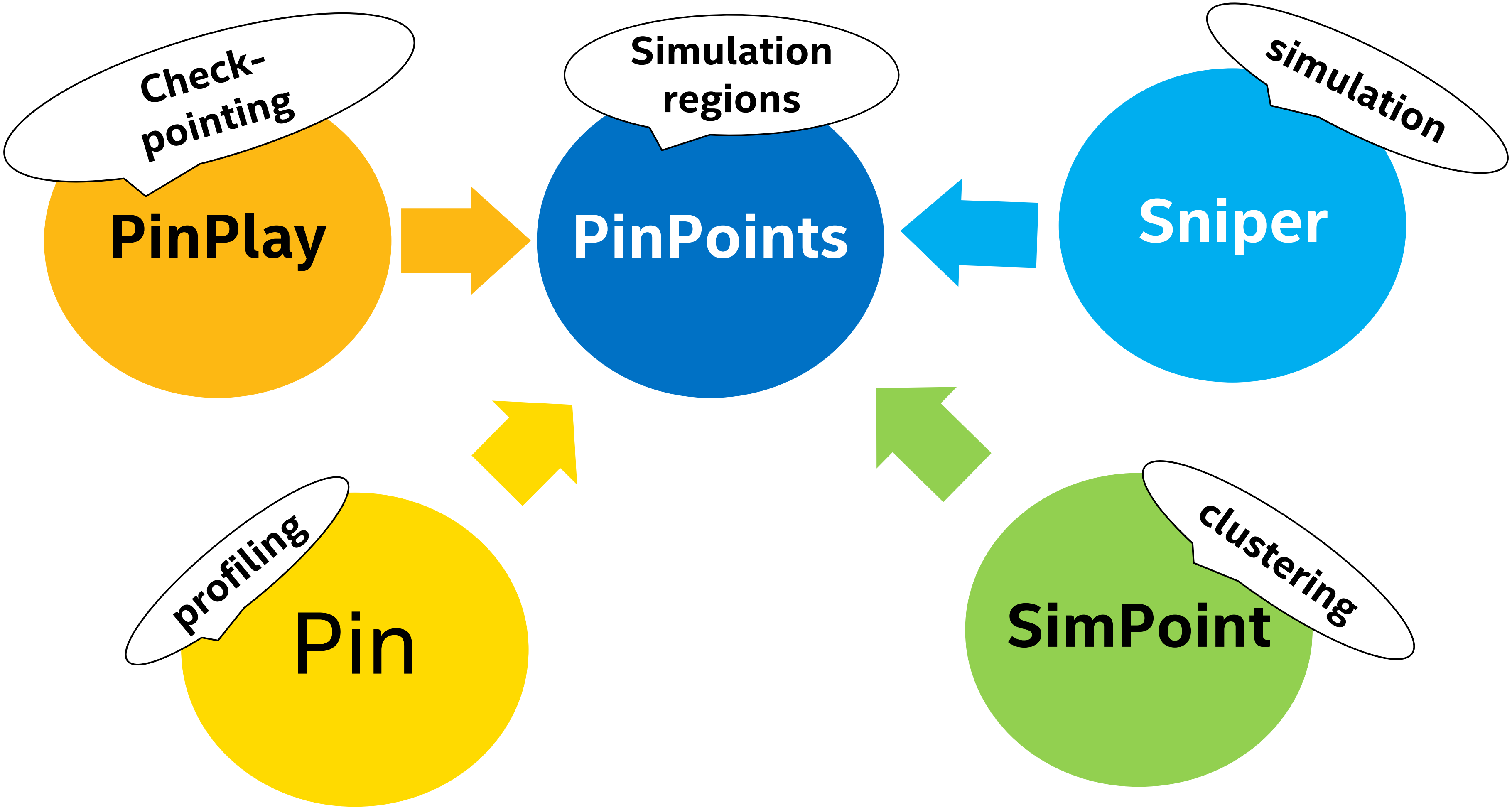
Match?
(Prediction Error low?)

Comparing region selection techniques



SPEC2000 (x86) CPI : Measured using HW counters

PinPoints : Tools and techniques



SimPoint *Timothy Sherwood, Erez Perelman, Greg Hamerly and Brad Calder (UCSD)*

sniper *Trevor Carlson, Wim Heirman, Lieven Eeckhout (Ghent University)*

What you will learn

1. How to download and install PinPlay and Sniper
2. How to use PinPlay for recording execution (pinballs)
3. How to profile and find representative regions using PinPlay and SimPoint, and create checkpoints (pinballs)
4. How to run Sniper with a pinball
5. How to find the quality of selected simulation region
6. How to tune the selection for better quality
7. How to download/use SPEC2006 PinPoints pinballs with Sniper

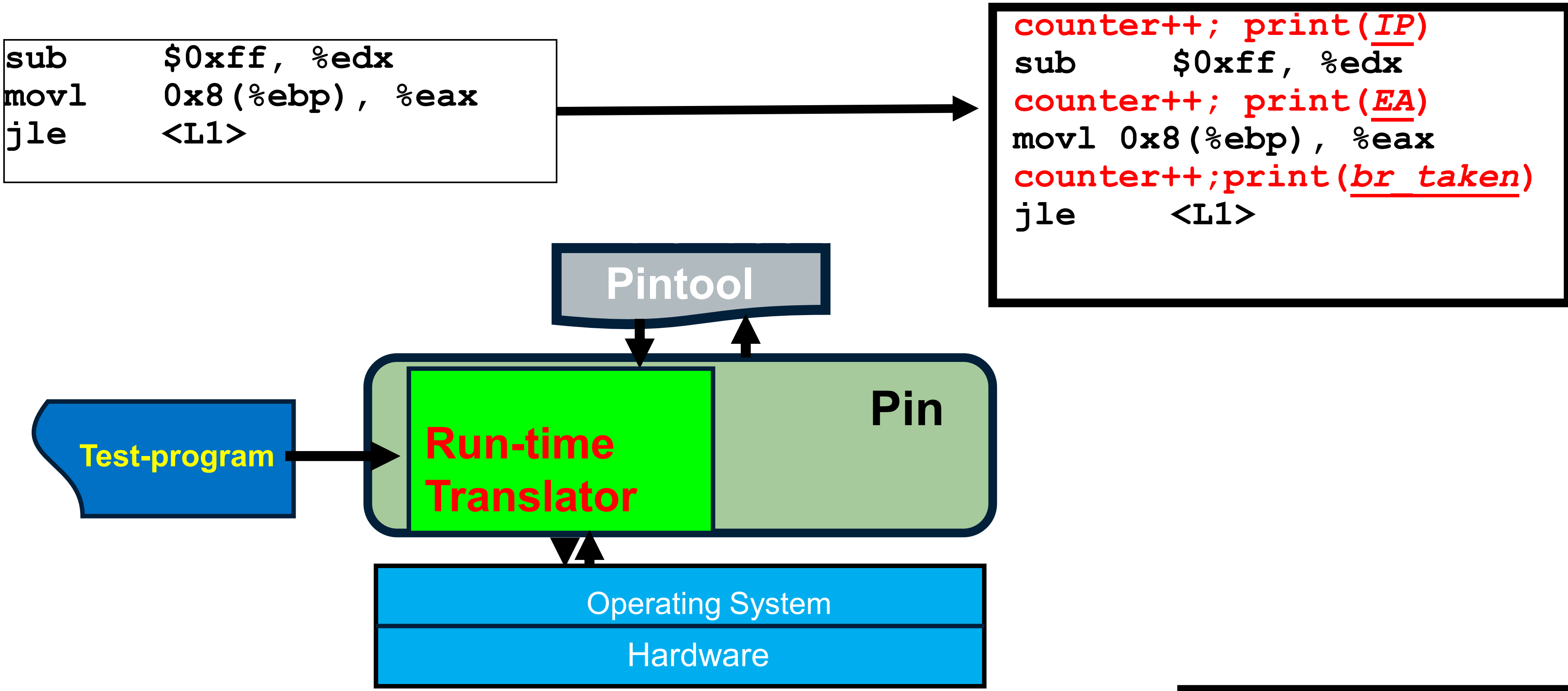
Outline

- Background
- Downloading tools & required packages
- Generating representative regions
- Prediction error
- Tune for better representative regions
- Advanced topics
- Summary

Background

Pin, SimPoint, PinPlay, Sniper

Pin: A Tool for Writing Program Analysis Tools



```
sub      $0xff, %edx
movl     0x8(%ebp), %eax
jle      <L1>
```

```
counter++; print(IP)
sub      $0xff, %edx
counter++; print(EA)
movl  0x8(%ebp), %eax
counter++;print(br_taken)
jle      <L1>
```

Pintool

Pin

Run-time Translator

Test-program

Operating System

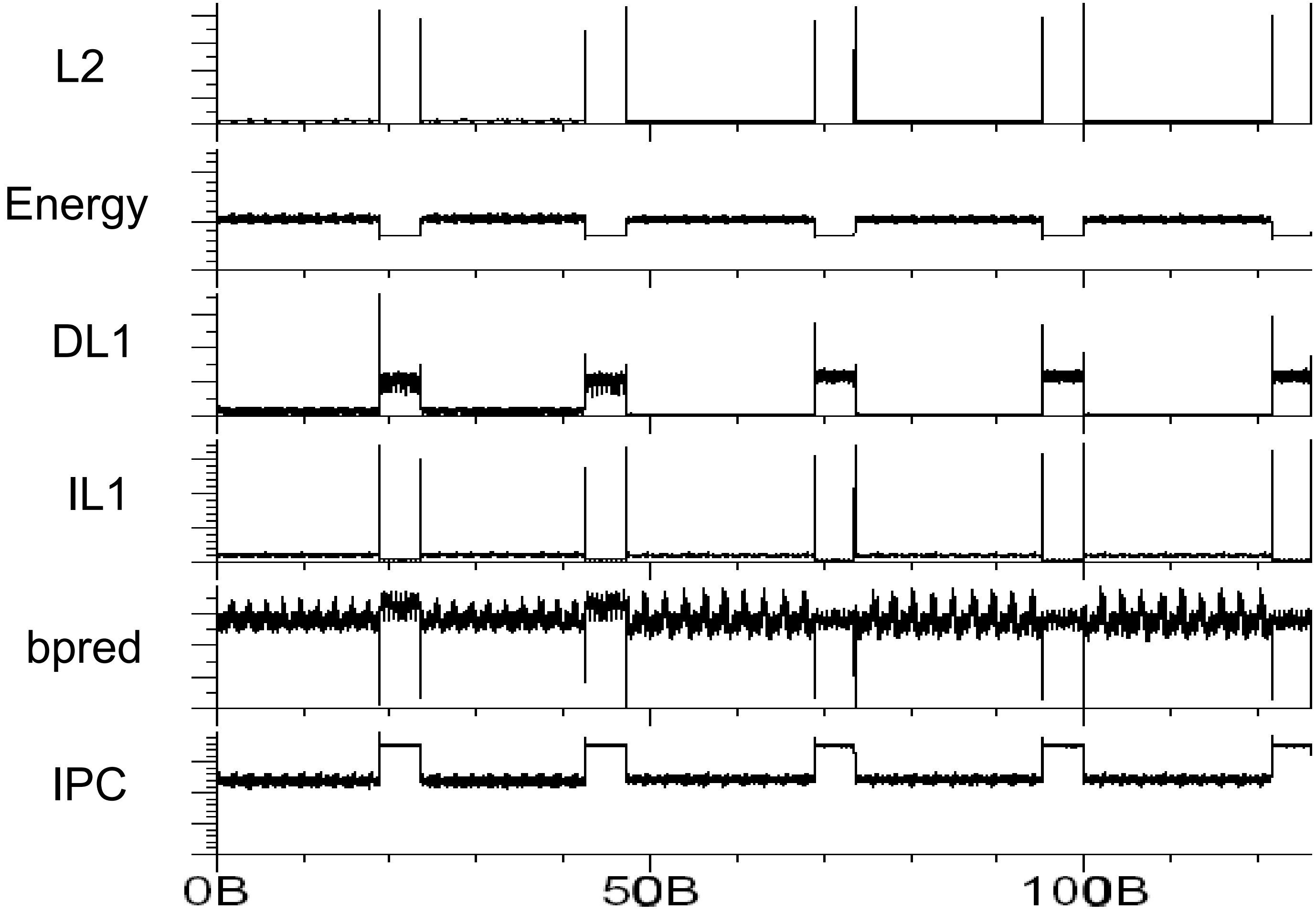
Hardware

```
$ pin -t pintool -- test-program
```

Normal output
+ *Analysis output*

Pin: A Dynamic Instrumentation Framework from Intel
http://www.pintool.org

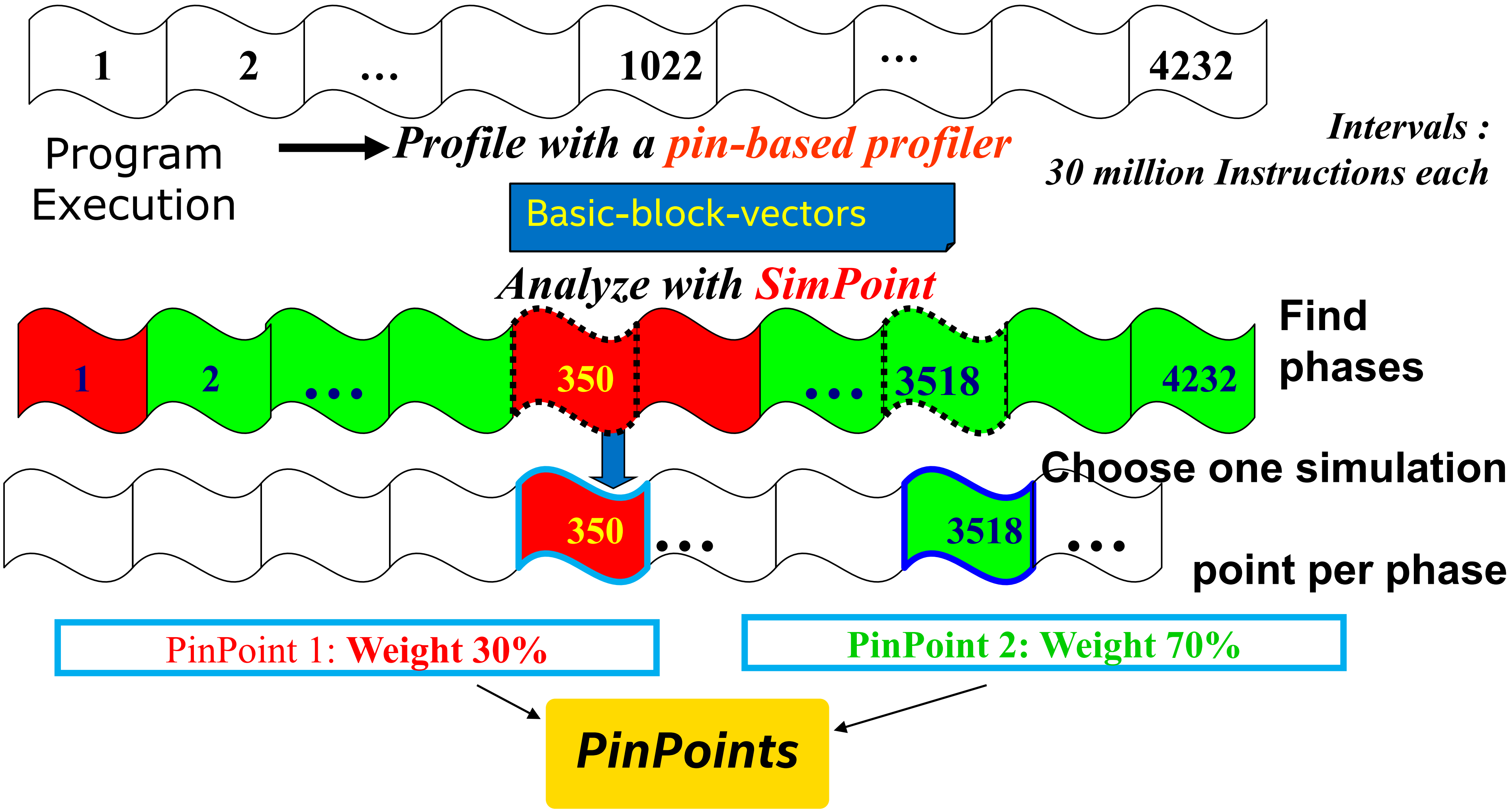
SimPoint: Program phase detection tool



UCSD

gzip (SPEC2000) : various properties vs. dynamic instruction count

PinPoints = Pin + SimPoint



1

2

…

1022

…

4232

Program Execution

Profile with a pin-based profiler

Intervals : 30 million Instructions each

Basic-block-vectors

Analyze with SimPoint

1

2

…

350

…

3518

4232

Find phases

350

…

3518

…

Choose one simulation point per phase

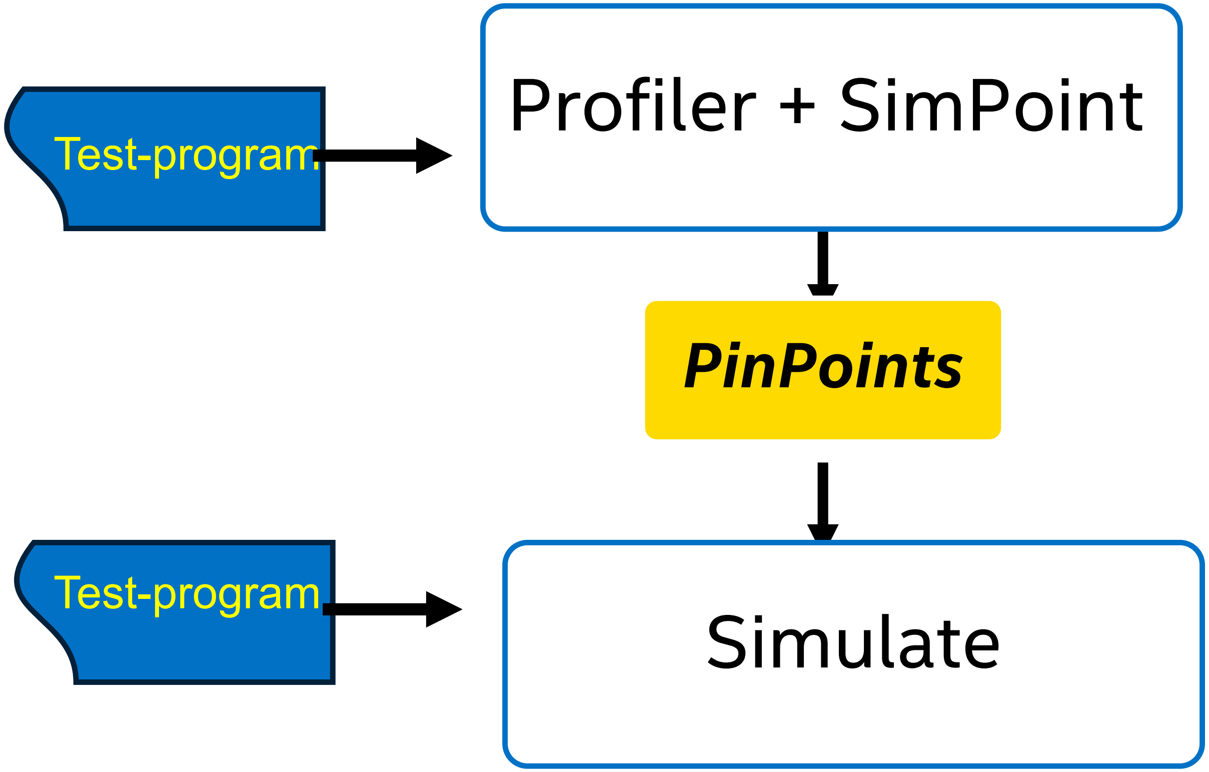
PinPoint 1: **Weight 30%**

PinPoint 2: **Weight 70%**

PinPoints

Two Phases => Two PinPoints

PinPoints : The repeatability challenge

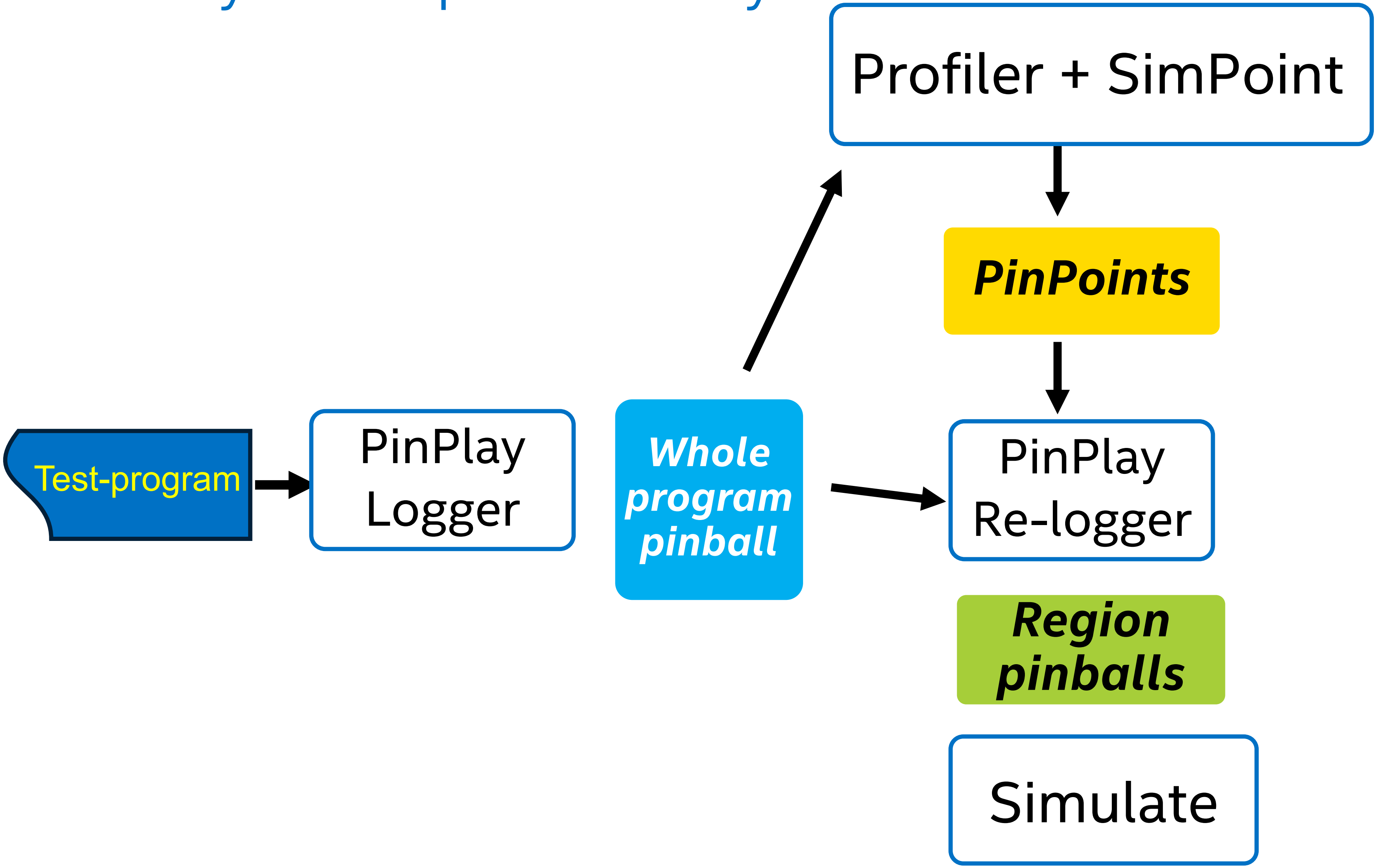


Problem: Two runs are not exactly same → PinPoints missed

Found this for 25/54 SPEC2006 runs!

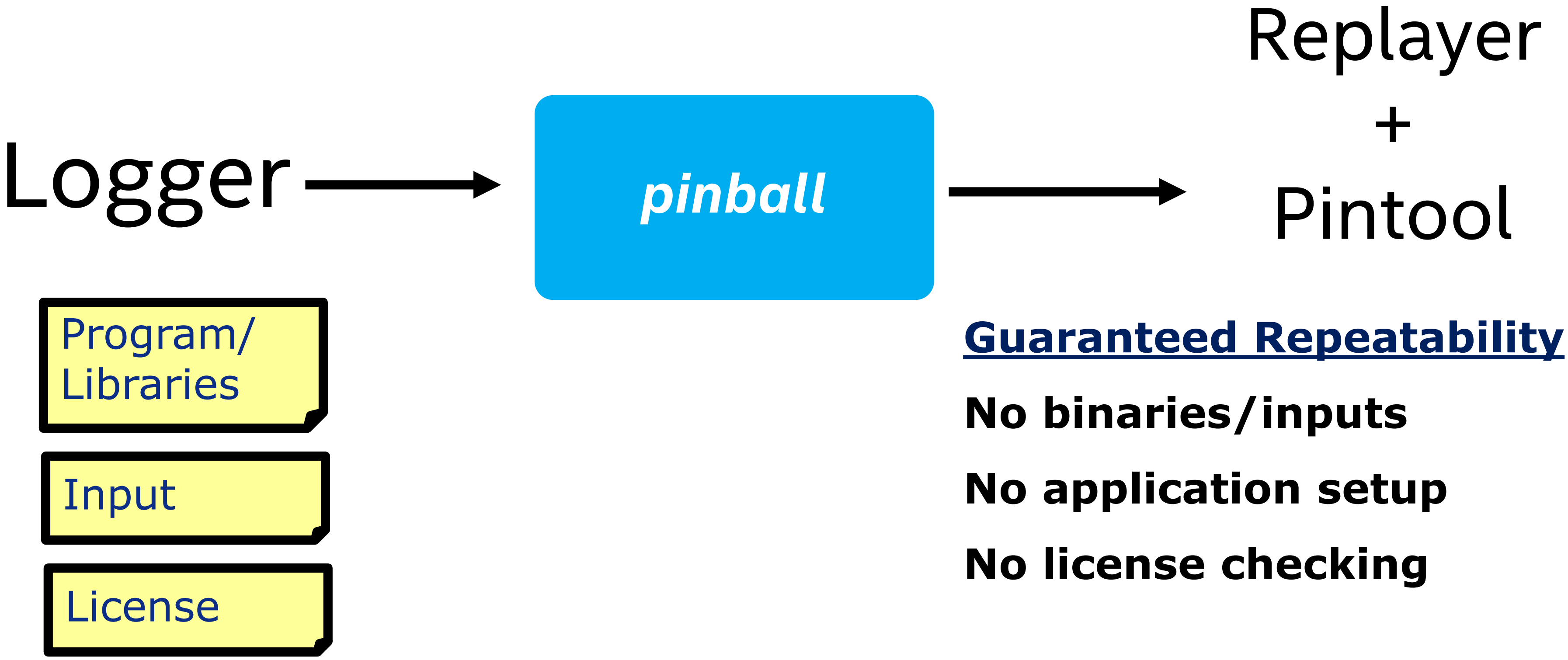
[*"PinPoints out of order" "PinPoint End seen before Start"*]

PinPlay → Repeatability



Pinballs: Portable, OS independent, provide determinism

PinPlay*: execution capture and deterministic replay framework



<u>**Guaranteed Repeatability**</u>

No binaries/inputs

No application setup

No license checking

Record once : Analyze multiple times, anywhere!

** Co-developers: Cristiano Pereira, James Cownie, Harish Patil*

"Program Record/Replay Toolkit" from Intel
http://www.pinplay.org

Sniper: A fast and accurate simulator

Hybrid simulation approach

- Analytical *interval* core model
- Micro-architecture structure simulation
  - branch predictors, caches (incl. coherency), NoC, etc.
- NEW: SniperLite : cache-only model for PinPoints validation

UNIVERSITEIT GENT

sniper

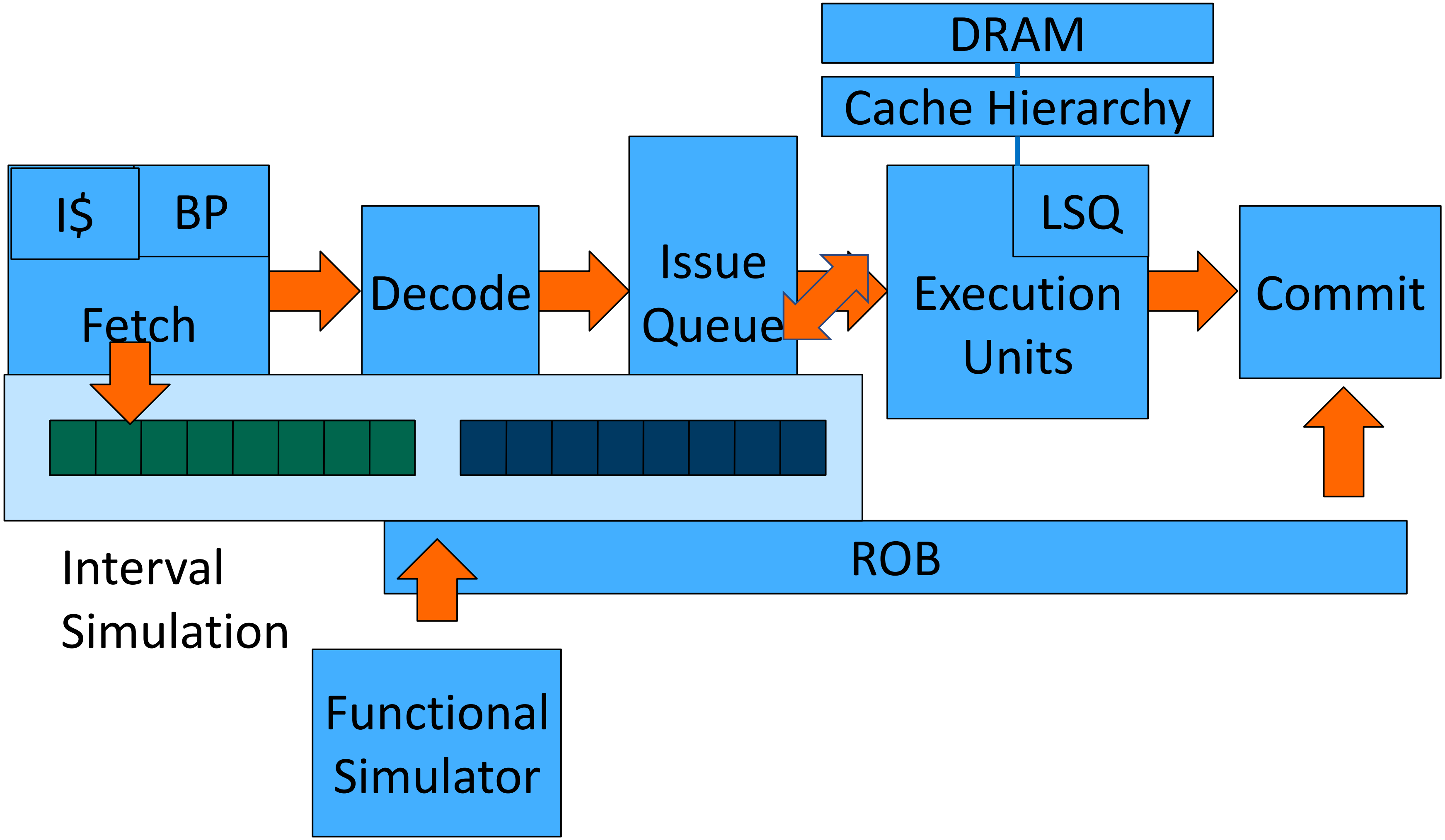
Models multi/many-cores running multi-threaded and multi-program workloads

Parallel simulator scales with the number of simulated cores

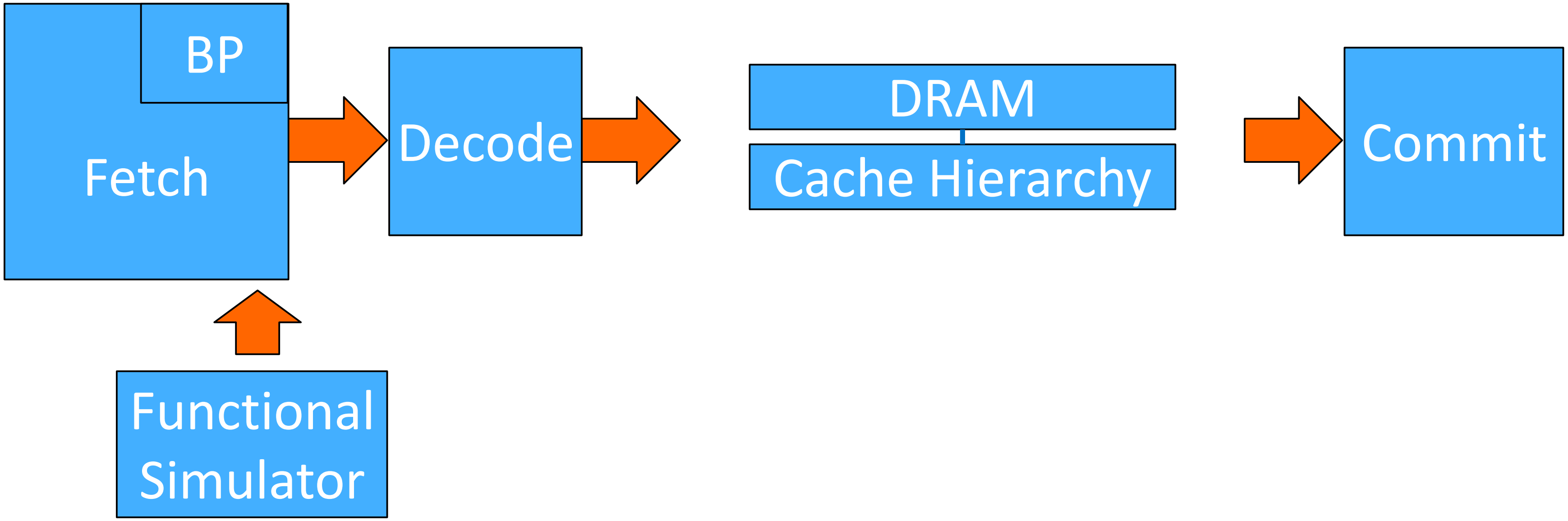
SniperLite: fast enough for simulating entire programs in reasonable time

Download Sniper from
http://www.snipersim.org

Sniper Interval Model



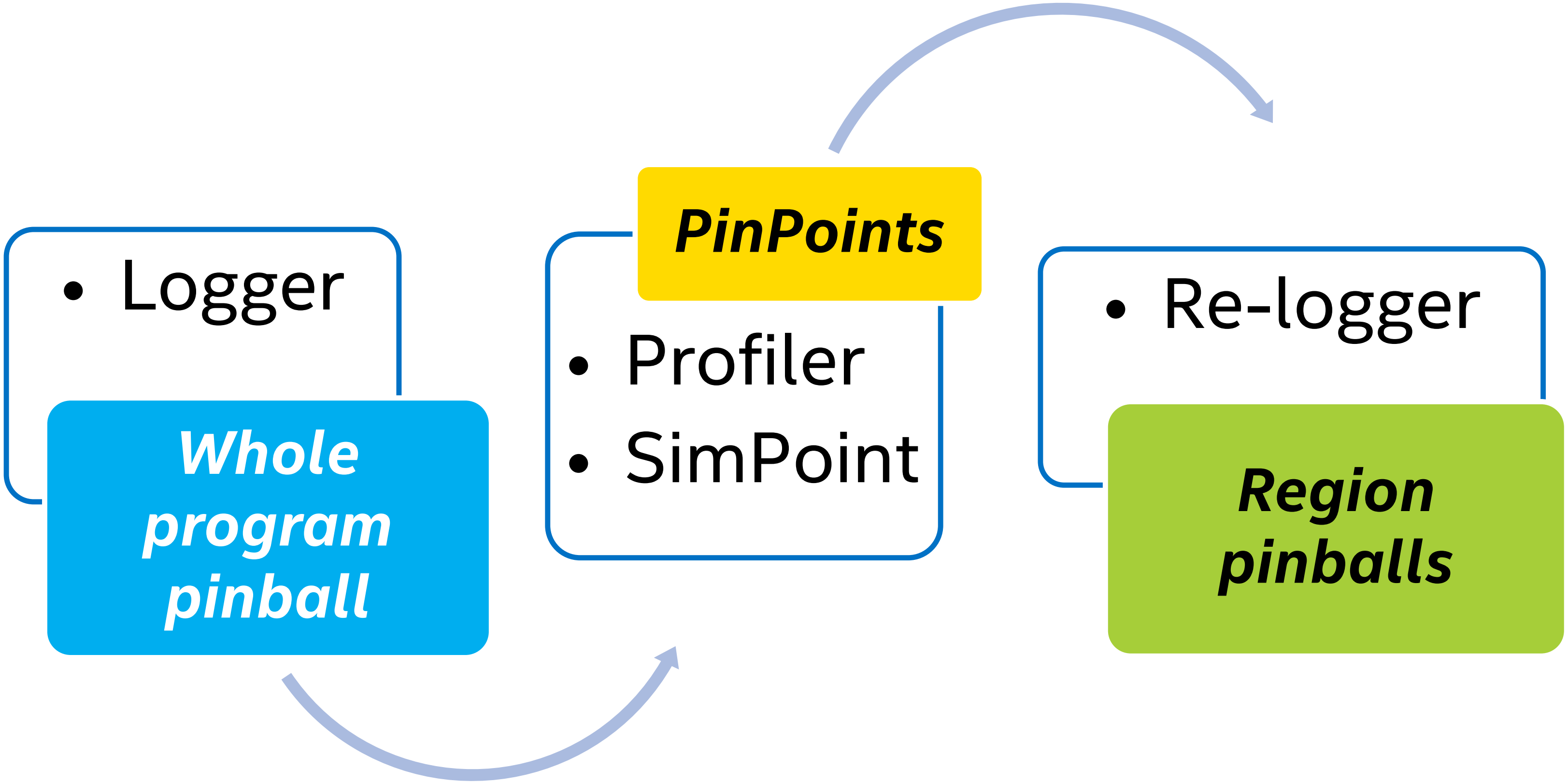
SniperLite Model



- Single-IPC
- Data cache simulation with fixed miss penalties
- Fast-enough to simulate large applications in reasonable time

Do not use for ‘real’ simulation : used only for PinPoints validation

PinPlay + PinPoints: Basic flow



- Logger

Whole program pinball

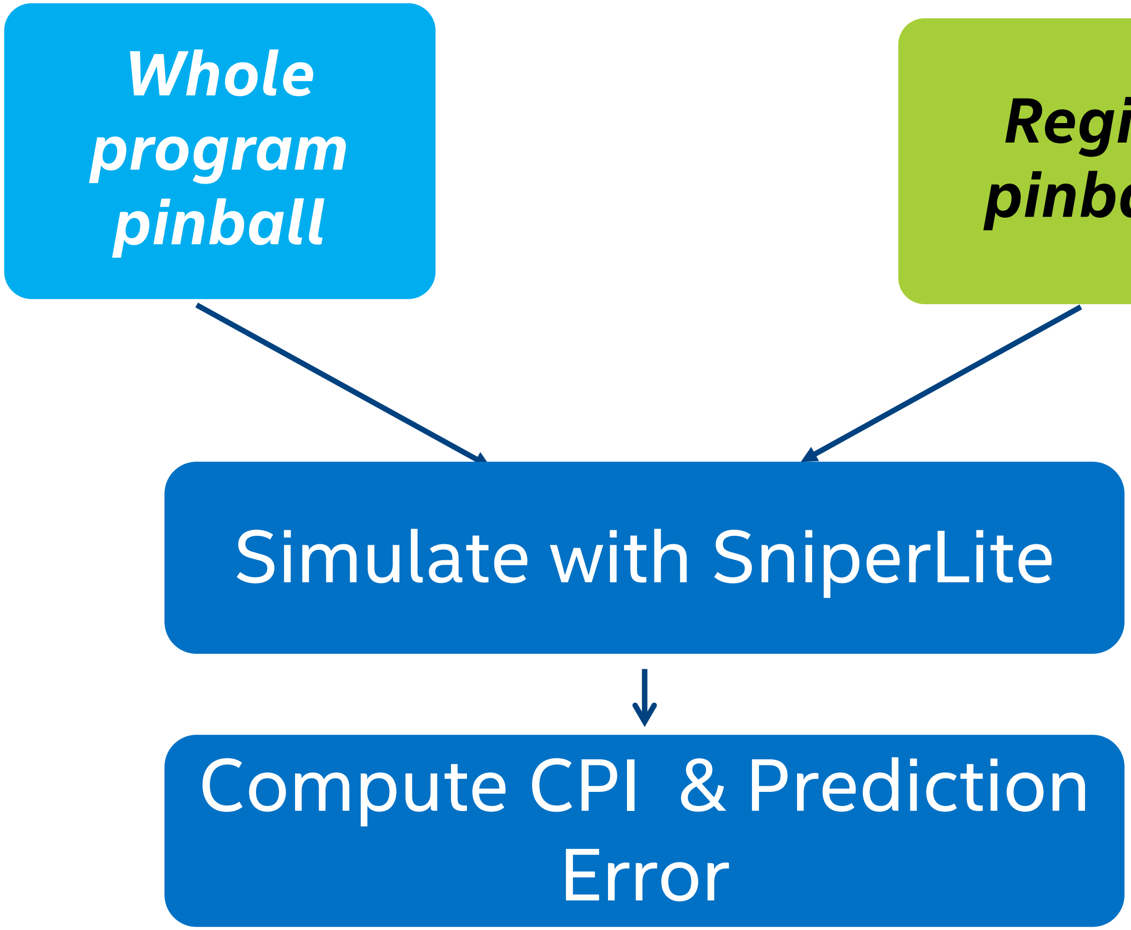
PinPoints

- Profiler
- SimPoint

- Re-logger

Region pinballs

PinPoints: Validation



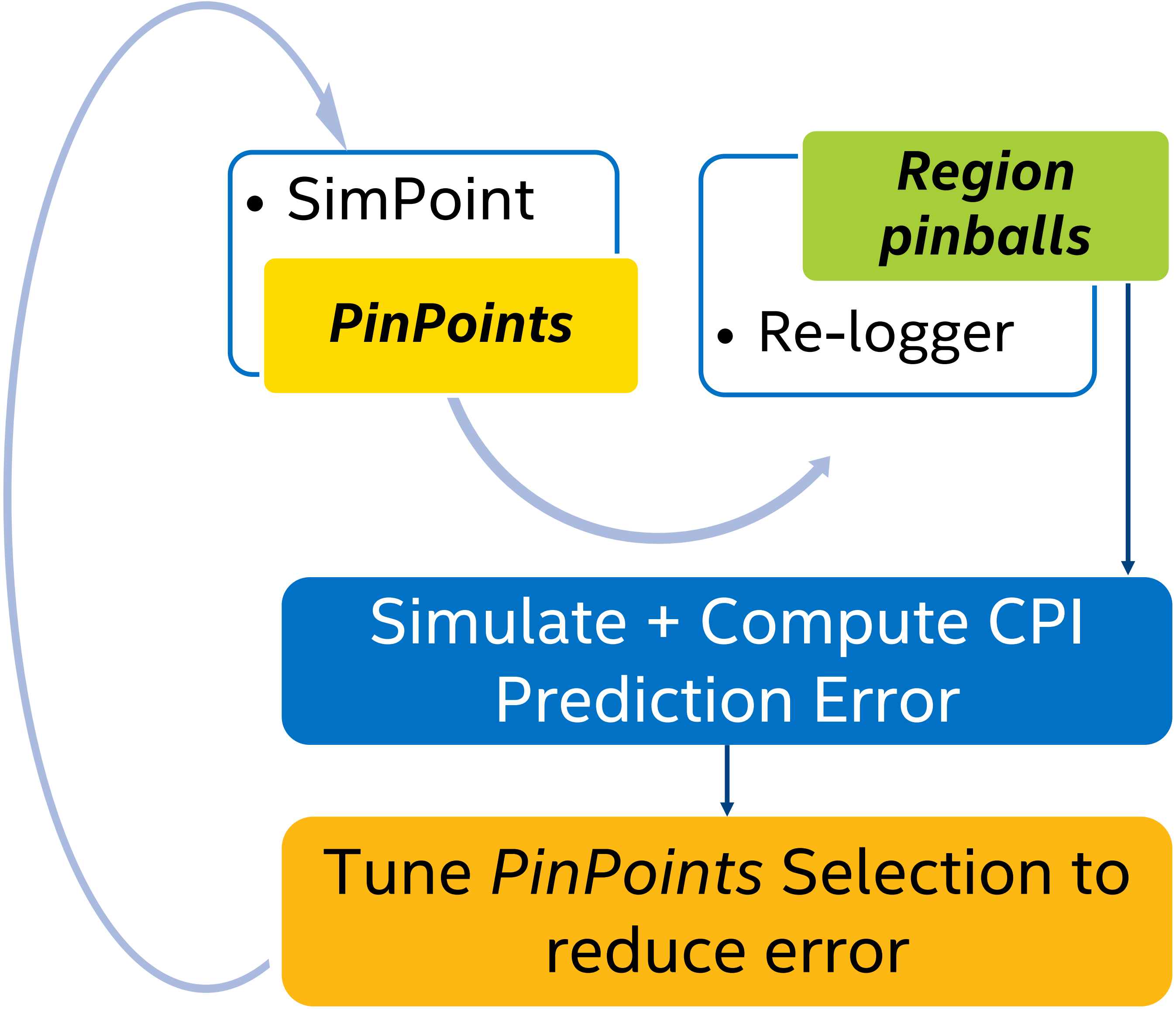
Whole program pinball

Region pinballs

Simulate with SniperLite

Compute CPI & Prediction Error

PinPoints: Tuning



- SimPoint

PinPoints

Region pinballs

- Re-logger

Simulate + Compute CPI Prediction Error

Tune *PinPoints* Selection to reduce error

Downloading Tools & Required Packages

PinPlay and Sniper setup on Ubuntu

Getting Ubuntu ready for Pin

Install required packages

```
sudo apt-get install gcc-multilib
sudo apt-get install g++
sudo apt-get install build-essential
sudo apt-get install g++-multilib
```

For Sniper

```
sudo apt-get install zlib1g-dev
sudo apt-get install libbz2-dev
sudo apt-get install libboost-dev
sudo apt-get install libsqlite3-dev
```

Download and install PinPlay kit 1.3

Change configuration

```
sudo sh
# echo 0 > /proc/sys/kernel/yama/ptrace_scope
# exit
```

Download kit 1.3 (updated June 10th 2014) from http://www.pinplay.org

```
tar -zxf <downloaded pinplay tar.gz file>
export PIN_ROOT=<path to unpacked PinPlay kit>
cd $PIN_ROOT/extras/pinplay/examples
make
```

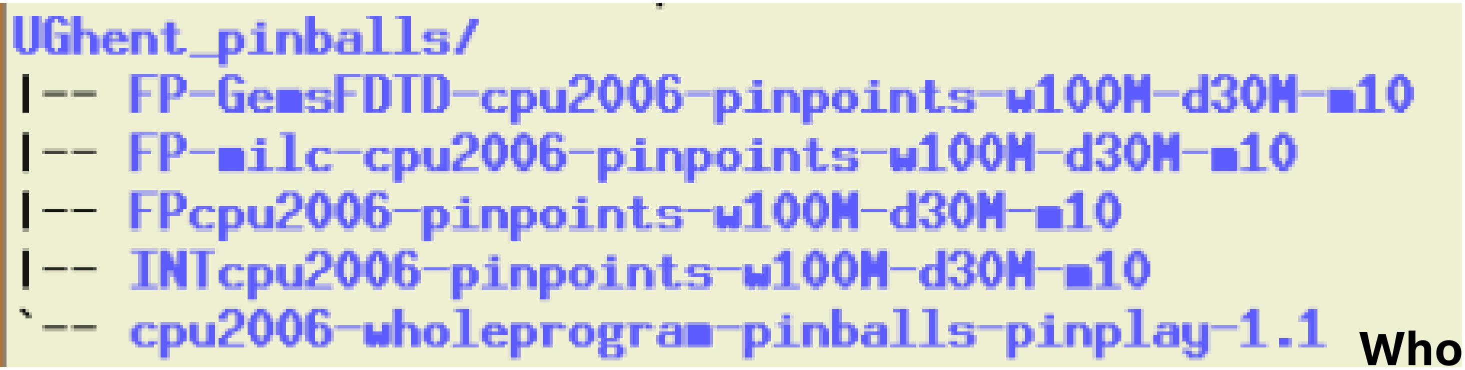
Download and install Sniper 6.0

Download from http://www.snipersim.org

(registration required; download link arrives in email)

```
tar -zxf <downloaded sniper tar.gz file>
export SNIPER_ROOT=<path to Sniper dir>
cd $SNIPER_ROOT
ln -sf $PIN_ROOT pin_kit
make
```

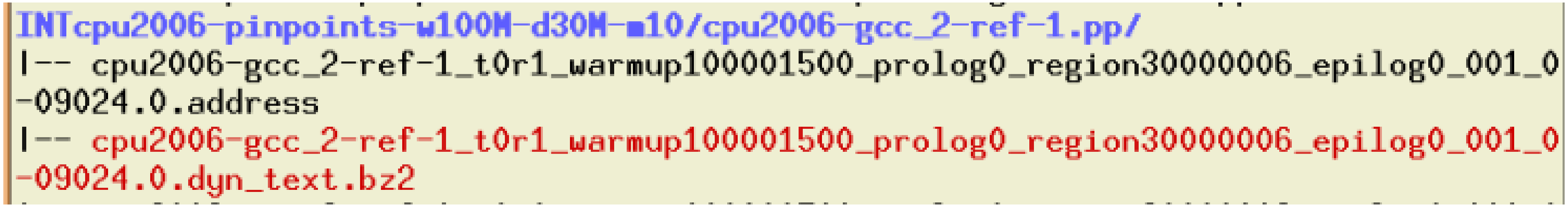
SPEC : CPU2006 Pinballs for download www.snipersim.org/Pinballs



```
UGhent_pinballs/
|-- FP-GemsFDTD-cpu2006-pinpoints-w100M-d30M-m10
|-- FP-milc-cpu2006-pinpoints-w100M-d30M-m10
|-- FPcpu2006-pinpoints-w100M-d30M-m10
|-- INTcpu2006-pinpoints-w100M-d30M-m10
`-- cpu2006-wholeprogram-pinballs-pinplay-1.1
```

Region pinballs

Whole-program pinballs



```
INTcpu2006-pinpoints-w100M-d30M-m10/cpu2006-gcc_2-ref-1.pp/
|-- cpu2006-gcc_2-ref-1_t0r1_warmup100001500_prolog0_region30000006_epilog0_001_0
-09024.0.address
|-- cpu2006-gcc_2-ref-1_t0r1_warmup100001500_prolog0_region30000006_epilog0_001_0
-09024.0.dyn_text.bz2
```

Warmup	Prolog	Simulation	Epilog
100 million	0	30 million	0

Schedule

~~8:45 – 9:30 Intro + Background (Harish)~~

9:30 – 10 Demo Part I (Mack)

10 – 10:30 Break

10:30 – 11:15 Demo Part II (Mack)

11:15 – 11:45 Advanced Topics (Harish)

11:45 – noon Wrap-up + Q&A (all)

Never precede any demo by a comment more predictive than 'Watch this!'

-- Michael Stallcup

30 years experience as NASA Engineer

Nomenclature & terms

Definitions used in presentation

- Workload - an application/input file(s) combination which specifies a given run of the application
- Example command lines use `this font`

Generating Representative Regions

Parameters

PinPoint scripts allow users to define parameters in a configuration file

- Almost all options can be put into config file
- Reduces re-typing common options when running scripts
- Can also use cmd line options to define parameters
- Options override parameters in config file
- String `“[Parameters]”` must be first line in config file
- Parameters defined as a key/value pair. For example:

```
program_name:   omnetpp
```

Configuration File

MUST define these 4 parameters which are used to describe the workload (app/input) configuration

```
program_name:
input_name:
command:
mode:
```

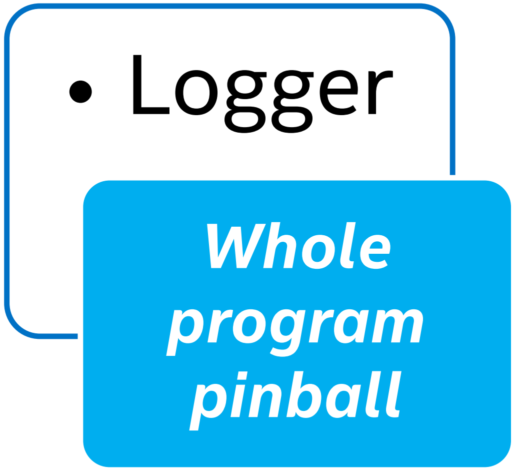
Mode needs to be one of:

- st = single thread
- mt = multi-thread
- mpi= MPI single-threaded
- mpi_mt = MPI multi-threaded

Demo configuration file

```
# Must include [Parameters] as the first non-comment line
[Parameters]
program_name:   omnetpp
input_name:     p10000-s10
command:        ./dtlb5-lin64 -p10000 -s10
maxk:           5
mode:           st
warmup_length:  1000000
slice_size:     3500000
pinplayhome:    pinplay-1.3-pin-2.13-65163-gcc.4.4.7-linux
sniper_root:    /home/tmstall/sniper-6.0
```

PinPlay + PinPoints : Basic Flow



- Logger

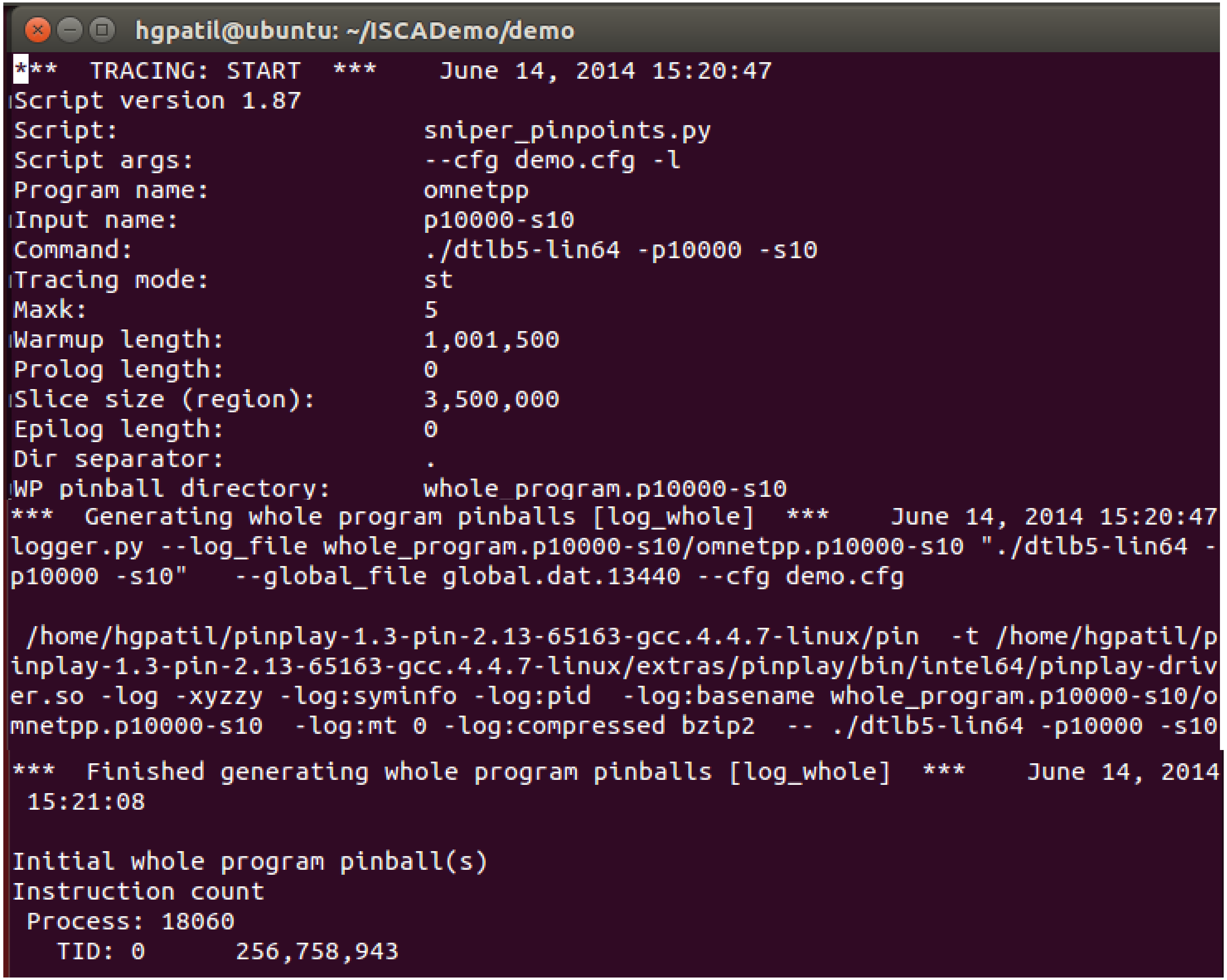
Whole program pinball

Generate whole program pinballs

Records all instructions and data as application runs workload

- Run logger to collect whole program data

```
sniper_pinpoints.py --cfg demo.cfg  -1 >& out_1.txt
```

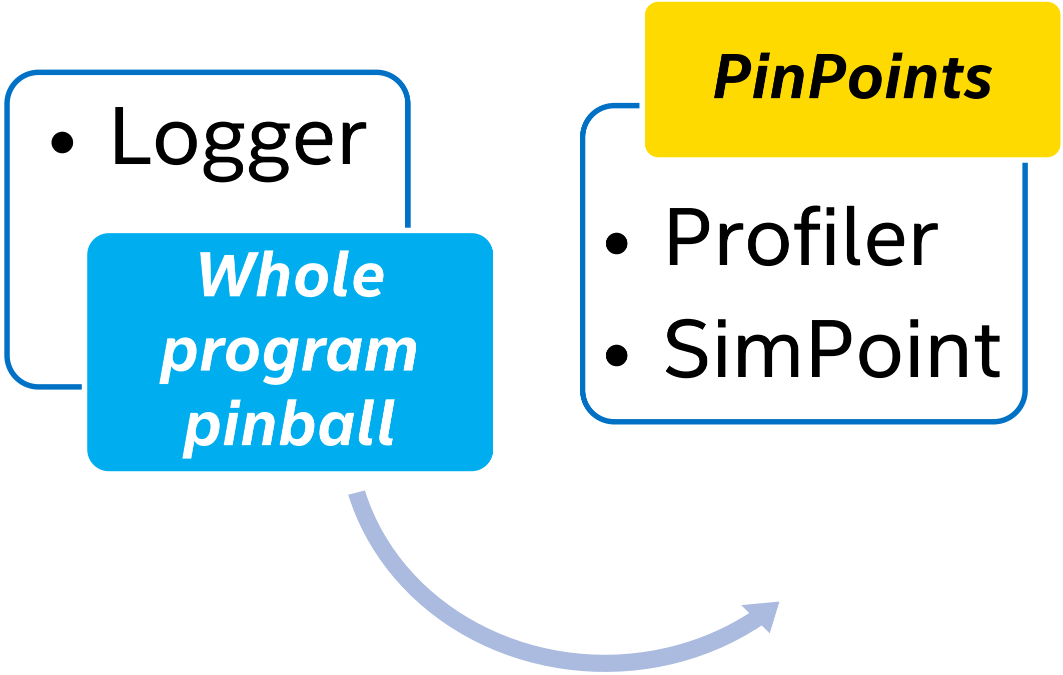


```
***  TRACING: START  ***    June 14, 2014 15:20:47
Script version 1.87
Script:                     sniper_pinpoints.py
Script args:                --cfg demo.cfg -l
Program name:               omnetpp
Input name:                 p10000-s10
Command:                    ./dtlb5-lin64 -p10000 -s10
Tracing mode:               st
Maxk:                       5
Warmup length:              1,001,500
Prolog length:              0
Slice size (region):        3,500,000
Epilog length:              0
Dir separator:              .
WP pinball directory:       whole_program.p10000-s10
***  Generating whole program pinballs [log_whole]  ***    June 14, 2014 15:20:47
logger.py --log_file whole_program.p10000-s10/omnetpp.p10000-s10 "./dtlb5-lin64 -
p10000 -s10"   --global_file global.dat.13440 --cfg demo.cfg

 /home/hgpatil/pinplay-1.3-pin-2.13-65163-gcc.4.4.7-linux/pin  -t /home/hgpatil/p
inplay-1.3-pin-2.13-65163-gcc.4.4.7-linux/extras/pinplay/bin/intel64/pinplay-driv
er.so -log -xyzzy -log:syminfo -log:pid  -log:basename whole_program.p10000-s10/o
mnetpp.p10000-s10  -log:mt 0 -log:compressed bzip2  -- ./dtlb5-lin64 -p10000 -s10
***  Finished generating whole program pinballs [log_whole]  ***    June 14, 2014
 15:21:08

Initial whole program pinball(s)
Instruction count
 Process: 18060
   TID: 0       256,758,943
```

PinPlay + PinPoints : Basic Flow



- Logger

Whole program pinball

PinPoints

- Profiler
- SimPoint

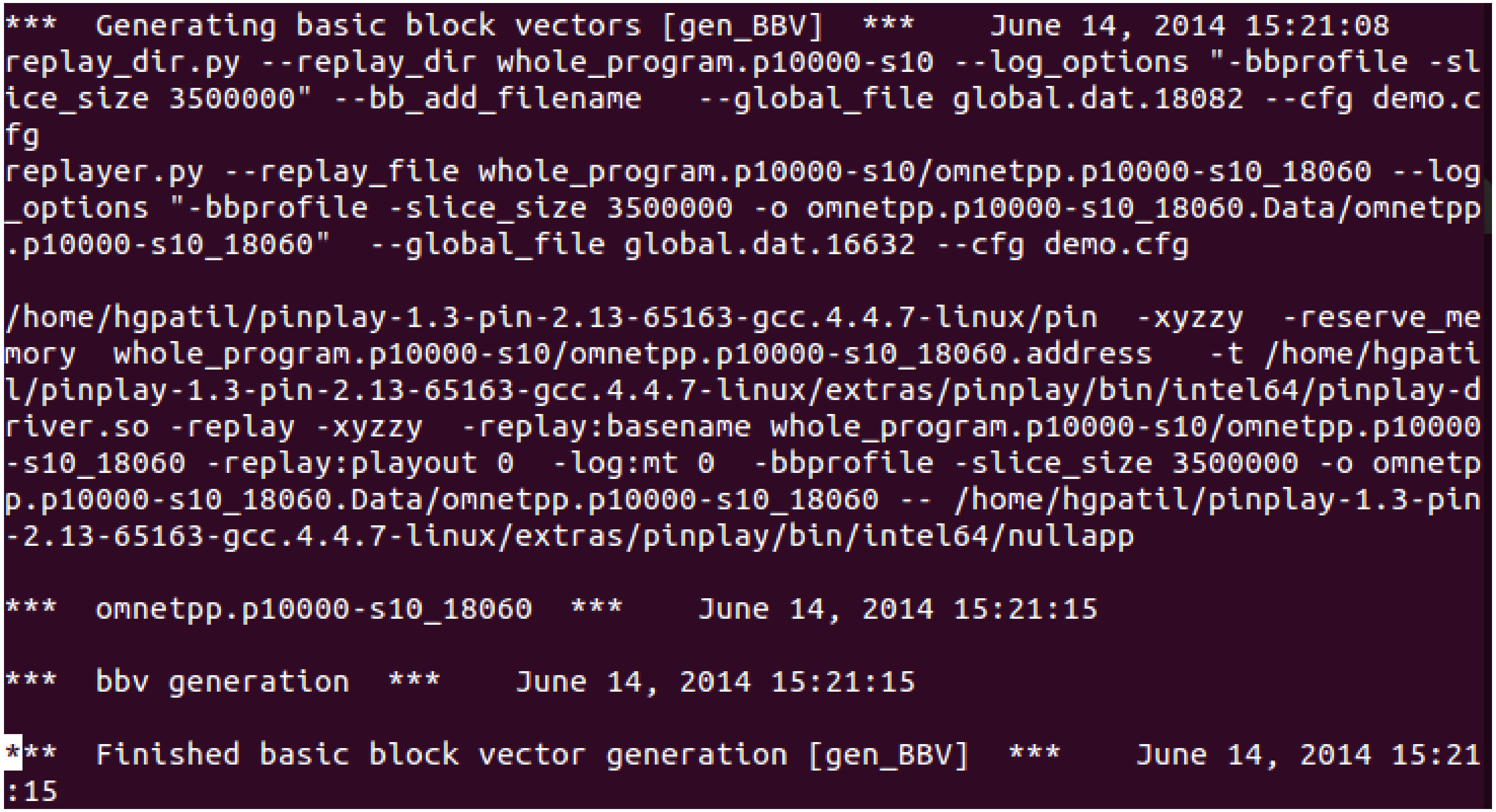
Generate PinPoints file

Profiler generates Basic Block Vectors (BBV)

```
sniper_pinpoints.py --cfg demo.cfg -b >& out_2.txt
```

SimPoint uses k-means clustering to generate representative regions

```
sniper_pinpoints.py --cfg demo.cfg -s >& out_3.txt
```



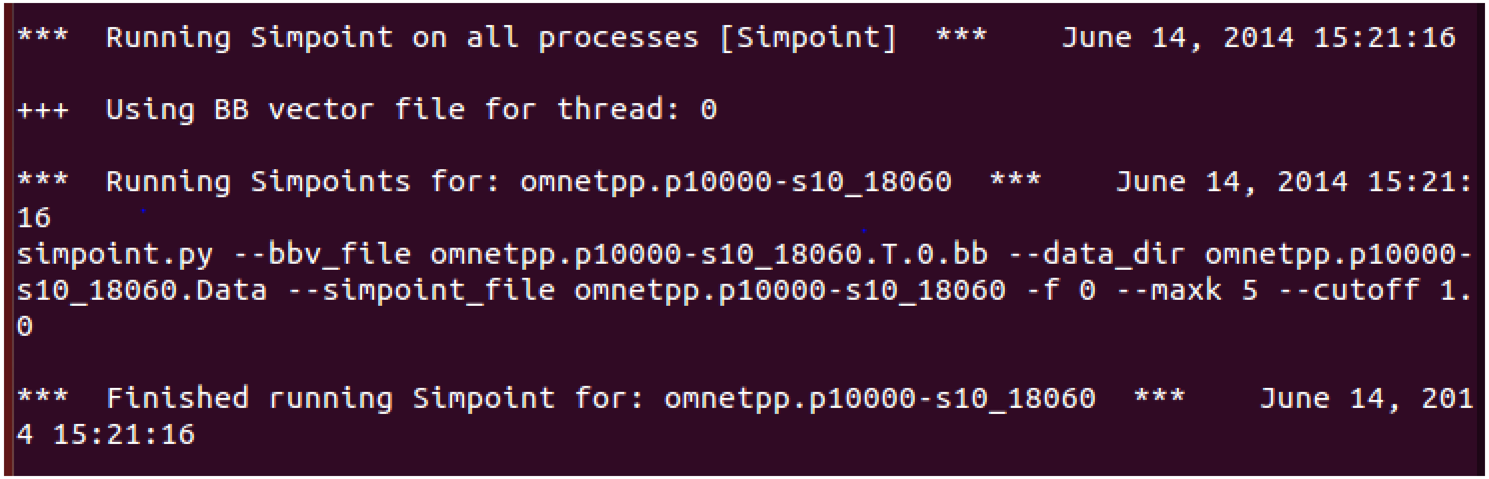
```
***  Generating basic block vectors [gen_BBV]  ***    June 14, 2014 15:21:08
replay_dir.py --replay_dir whole_program.p10000-s10 --log_options "-bbprofile -sl
ice_size 3500000" --bb_add_filename   --global_file global.dat.18082 --cfg demo.c
fg
replayer.py --replay_file whole_program.p10000-s10/omnetpp.p10000-s10_18060 --log
_options "-bbprofile -slice_size 3500000 -o omnetpp.p10000-s10_18060.Data/omnetpp
.p10000-s10_18060"  --global_file global.dat.16632 --cfg demo.cfg

/home/hgpatil/pinplay-1.3-pin-2.13-65163-gcc.4.4.7-linux/pin  -xyzzy  -reserve_me
mory  whole_program.p10000-s10/omnetpp.p10000-s10_18060.address   -t /home/hgpati
l/pinplay-1.3-pin-2.13-65163-gcc.4.4.7-linux/extras/pinplay/bin/intel64/pinplay-d
river.so -replay -xyzzy  -replay:basename whole_program.p10000-s10/omnetpp.p10000
-s10_18060 -replay:playout 0  -log:mt 0  -bbprofile -slice_size 3500000 -o omnetp
p.p10000-s10_18060.Data/omnetpp.p10000-s10_18060 -- /home/hgpatil/pinplay-1.3-pin
-2.13-65163-gcc.4.4.7-linux/extras/pinplay/bin/intel64/nullapp

***  omnetpp.p10000-s10_18060  ***    June 14, 2014 15:21:15

***  bbv generation  ***    June 14, 2014 15:21:15

***  Finished basic block vector generation [gen_BBV]  ***    June 14, 2014 15:21
:15
```



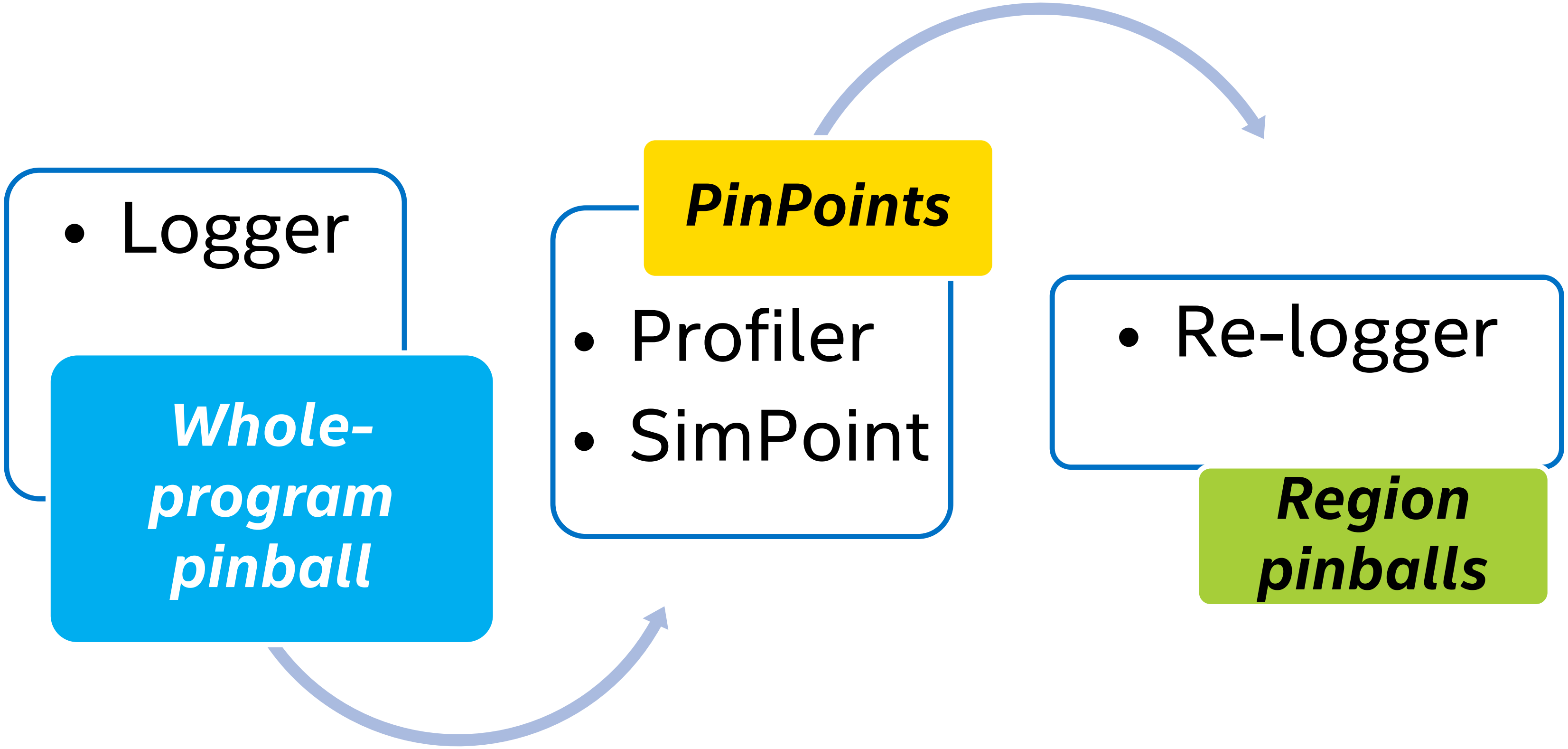
```
***  Running Simpoint on all processes [Simpoint]  ***    June 14, 2014 15:21:16

+++  Using BB vector file for thread: 0

***  Running Simpoints for: omnetpp.p10000-s10_18060  ***    June 14, 2014 15:21:
16
simpoint.py --bbv_file omnetpp.p10000-s10_18060.T.0.bb --data_dir omnetpp.p10000-
s10_18060.Data --simpoint_file omnetpp.p10000-s10_18060 -f 0 --maxk 5 --cutoff 1.
0

***  Finished running Simpoint for: omnetpp.p10000-s10_18060  ***    June 14, 201
4 15:21:16
```

PinPlay + PinPoints : Basic Flow



- Logger

Whole-program pinball

PinPoints

- Profiler
- SimPoint

- Re-logger

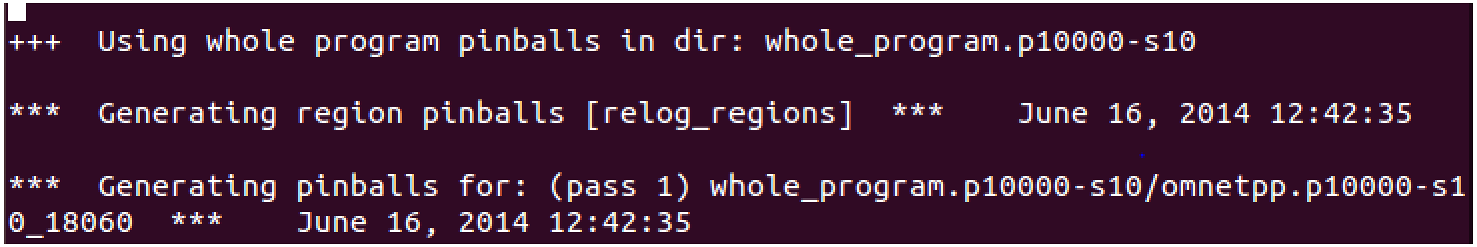
Region pinballs

Generate region pinballs

Records instructions and data the app uses to execute each region (region checkpoint)

- Pinball allows replay of just the region
- Includes warmup instructions
- Replays whole program pinball and log just instructions/data for each representative region
- Also called ‘relogging’

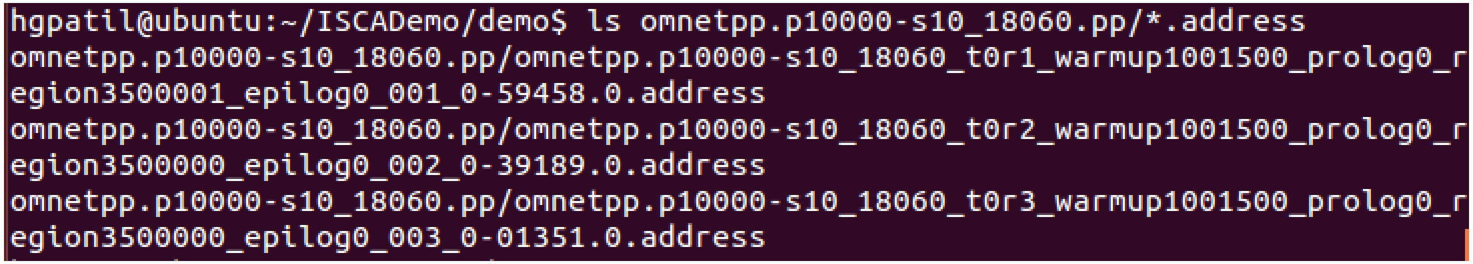
```
sniper_pinpoints.py --cfg demo.cfg  -p >& out_4.txt
```



```
+++  Using whole program pinballs in dir: whole_program.p10000-s10

***  Generating region pinballs [relog_regions]  ***    June 16, 2014 12:42:35

***  Generating pinballs for: (pass 1) whole_program.p10000-s10/omnetpp.p10000-s1
0_18060  ***    June 16, 2014 12:42:35
```



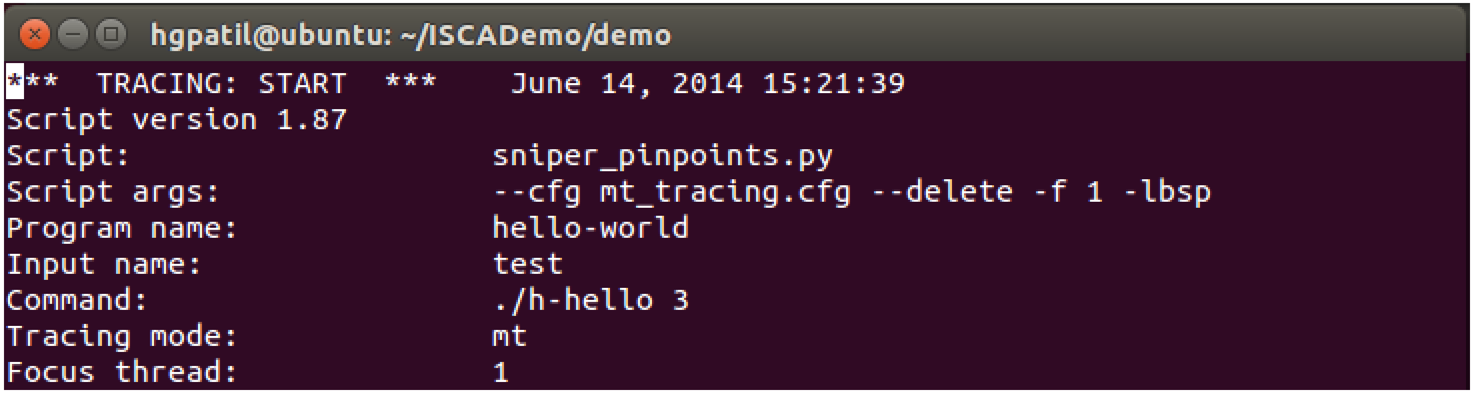
```
hgpatil@ubuntu:~/ISCADemo/demo$ ls omnetpp.p10000-s10_18060.pp/*.address
omnetpp.p10000-s10_18060.pp/omnetpp.p10000-s10_18060_t0r1_warmup1001500_prolog0_r
egion3500001_epilog0_001_0-59458.0.address
omnetpp.p10000-s10_18060.pp/omnetpp.p10000-s10_18060_t0r2_warmup1001500_prolog0_r
egion3500000_epilog0_002_0-39189.0.address
omnetpp.p10000-s10_18060.pp/omnetpp.p10000-s10_18060_t0r3_warmup1001500_prolog0_r
egion3500000_epilog0_003_0-01351.0.address
```

Multi-threaded apps

PinPoint scripts can be used on multi-threaded apps

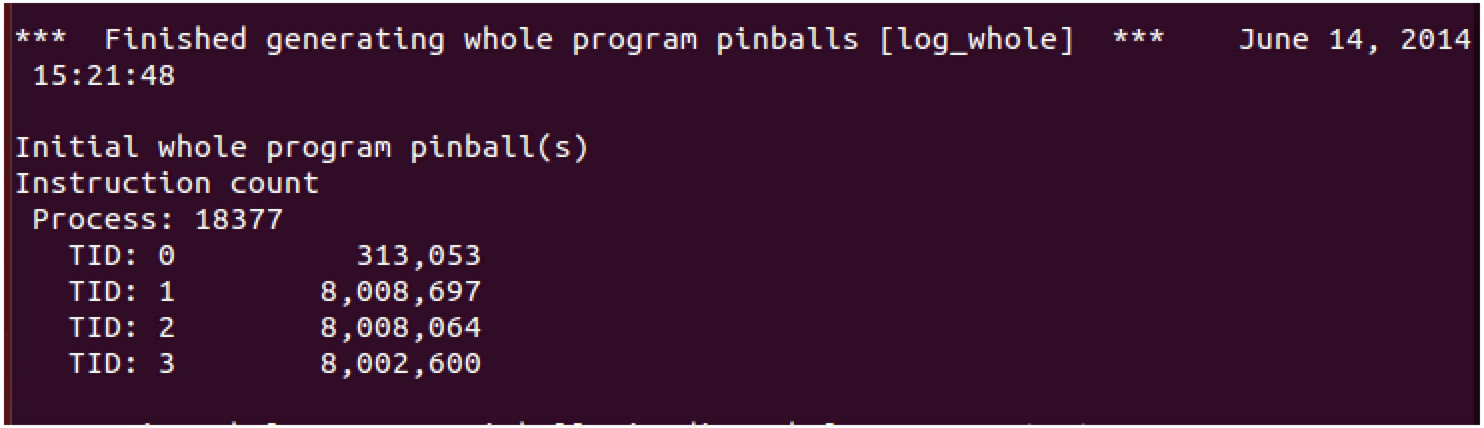
- Current limitation is must select focus thread
- Chose focus thread using '-f 1'
- Default focus thread is 0
- Must set parameter 'mode' to: mt

```
sniper_pinpoints.py --cfg mt_tracing.cfg  -f 1 -lbsp >&
out_4_1.txt
```



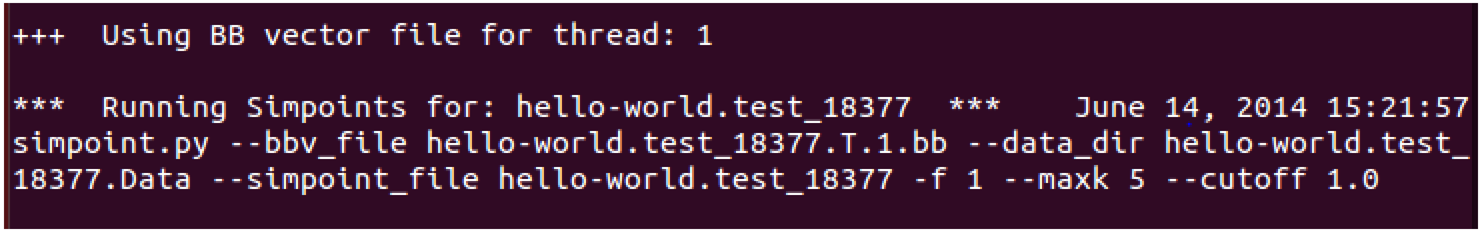
hgpatil@ubuntu: ~/ISCADemo/demo

```
*** TRACING: START  ***    June 14, 2014 15:21:39
Script version 1.87
Script:                    sniper_pinpoints.py
Script args:               --cfg mt_tracing.cfg --delete -f 1 -lbsp
Program name:              hello-world
Input name:                test
Command:                   ./h-hello 3
Tracing mode:              mt
Focus thread:              1
```



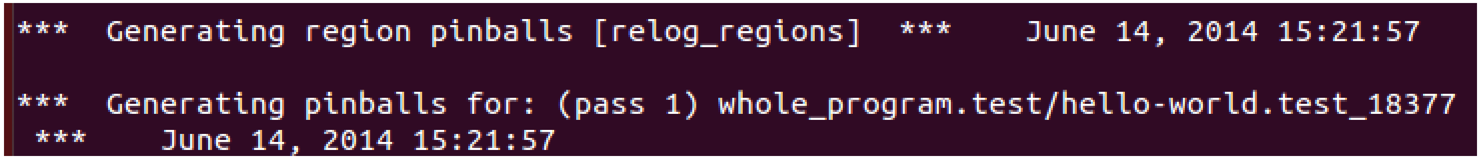
```
***  Finished generating whole program pinballs [log_whole]  ***    June 14, 2014
 15:21:48

Initial whole program pinball(s)
Instruction count
 Process: 18377
   TID: 0          313,053
   TID: 1        8,008,697
   TID: 2        8,008,064
   TID: 3        8,002,600
```



```
+++  Using BB vector file for thread: 1

***  Running Simpoints for: hello-world.test_18377  ***    June 14, 2014 15:21:57
simpoint.py --bbv_file hello-world.test_18377.T.1.bb --data_dir hello-world.test_
18377.Data --simpoint_file hello-world.test_18377 -f 1 --maxk 5 --cutoff 1.0
```



```
***  Generating region pinballs [relog_regions]  ***    June 14, 2014 15:21:57

***  Generating pinballs for: (pass 1) whole_program.test/hello-world.test_18377
 ***    June 14, 2014 15:21:57
```

Prediction Error

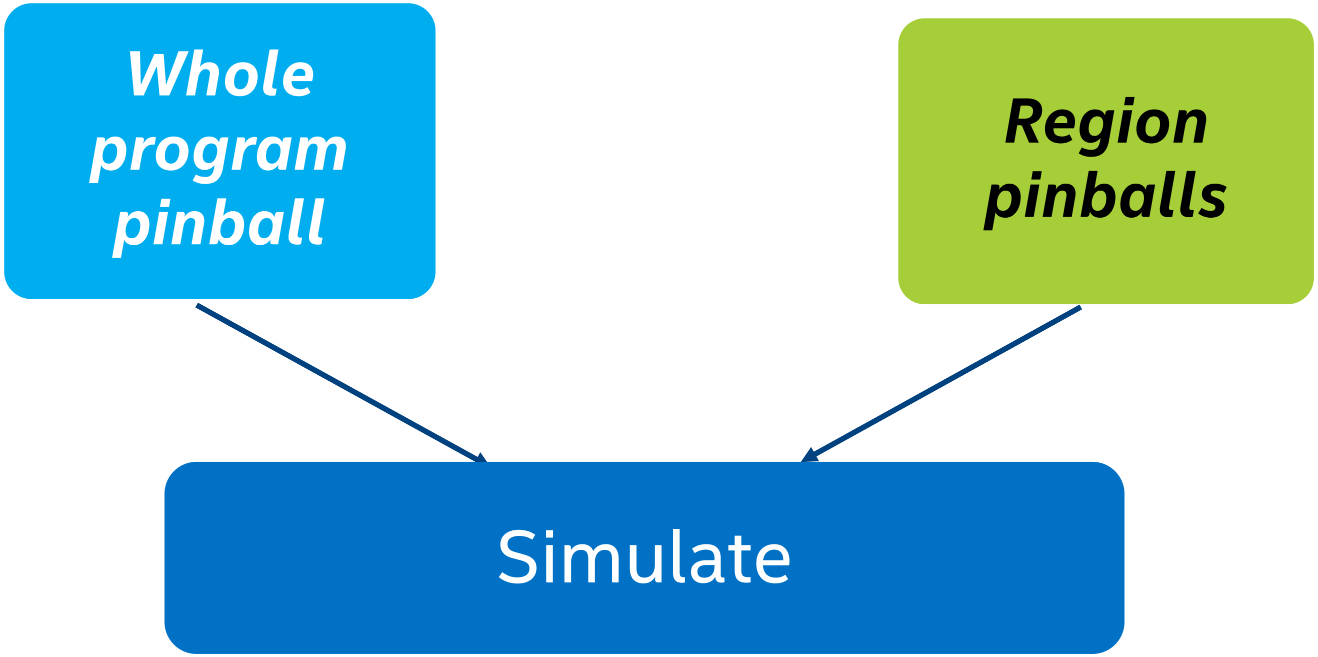
What is prediction error?

Prediction error is a measure of how representative the regions are of the entire workload behavior

- Use Sniper simulator in a ‘Lite’ configuration to collect data required to calculate prediction error
- SniperLite is much faster than Sniper, but there are some restrictions

A metric for judging the quality of region selection

PinPoints: Validation



Whole program pinball

Region pinballs

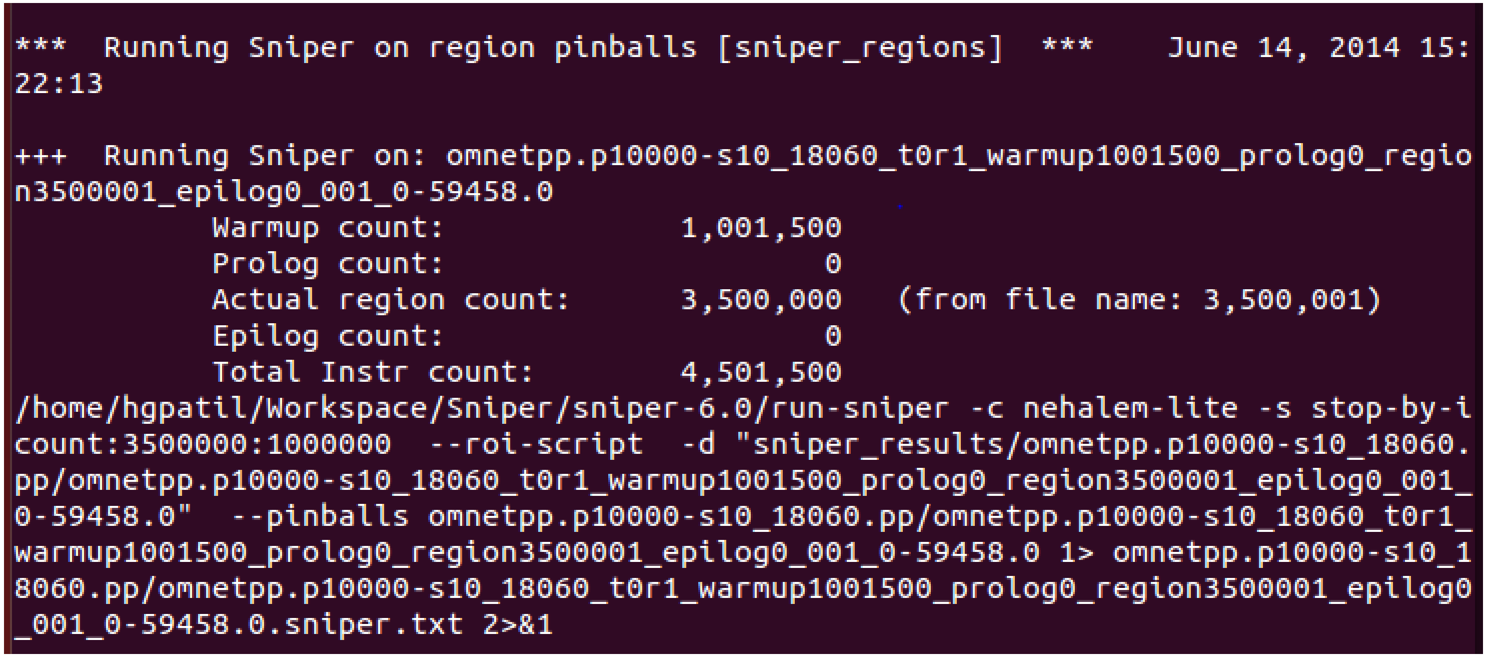
Simulate

Run SniperLite on WP/region pinballs

Sniper 6.0 has configuration file which enables SniperLite

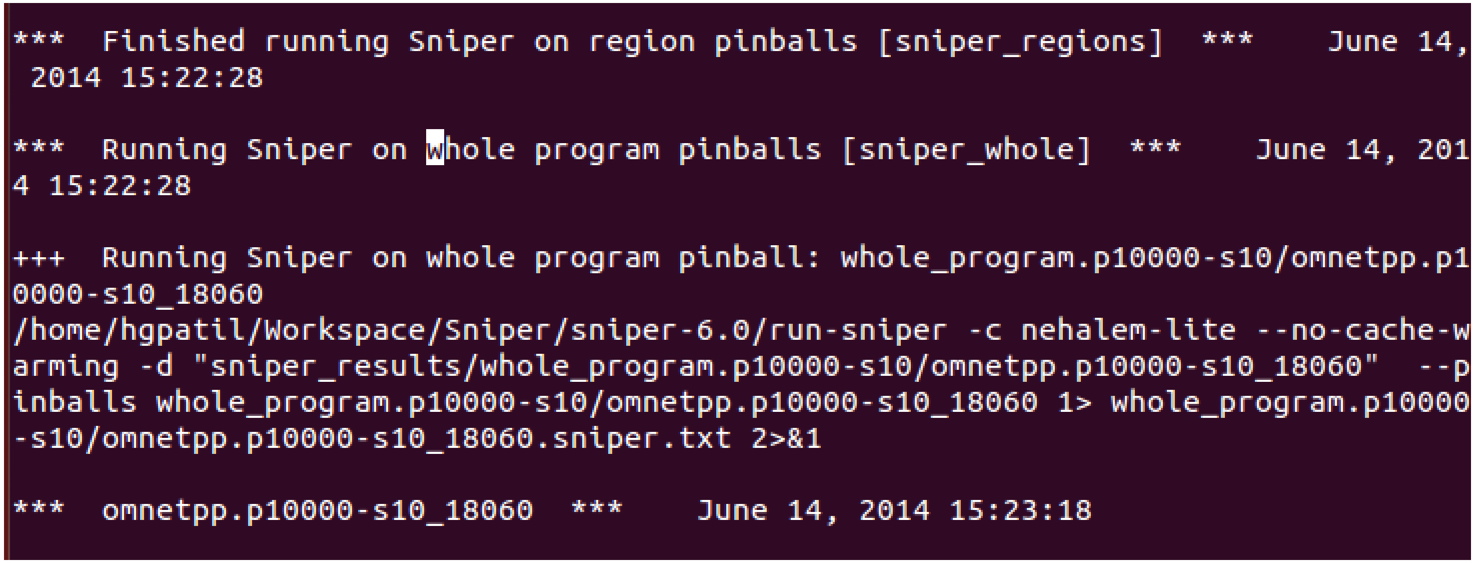
- Use option: `-c nehalem-lite`
- Only a limited set of SniperLite metrics are ‘reliable’

```
sniper_pinpoints.py --cfg demo.cfg  -TW >& out_5.txt
```



```
***  Running Sniper on region pinballs [sniper_regions]  ***    June 14, 2014 15:
22:13

+++  Running Sniper on: omnetpp.p10000-s10_18060_t0r1_warmup1001500_prolog0_regio
n3500001_epilog0_001_0-59458.0
           Warmup count:              1,001,500
           Prolog count:                      0
           Actual region count:       3,500,000   (from file name: 3,500,001)
           Epilog count:                      0
           Total Instr count:         4,501,500
/home/hgpatil/Workspace/Sniper/sniper-6.0/run-sniper -c nehalem-lite -s stop-by-i
count:3500000:1000000  --roi-script  -d "sniper_results/omnetpp.p10000-s10_18060.
pp/omnetpp.p10000-s10_18060_t0r1_warmup1001500_prolog0_region3500001_epilog0_001_
0-59458.0"  --pinballs omnetpp.p10000-s10_18060.pp/omnetpp.p10000-s10_18060_t0r1_
warmup1001500_prolog0_region3500001_epilog0_001_0-59458.0 1> omnetpp.p10000-s10_1
8060.pp/omnetpp.p10000-s10_18060_t0r1_warmup1001500_prolog0_region3500001_epilog0
_001_0-59458.0.sniper.txt 2>&1
```



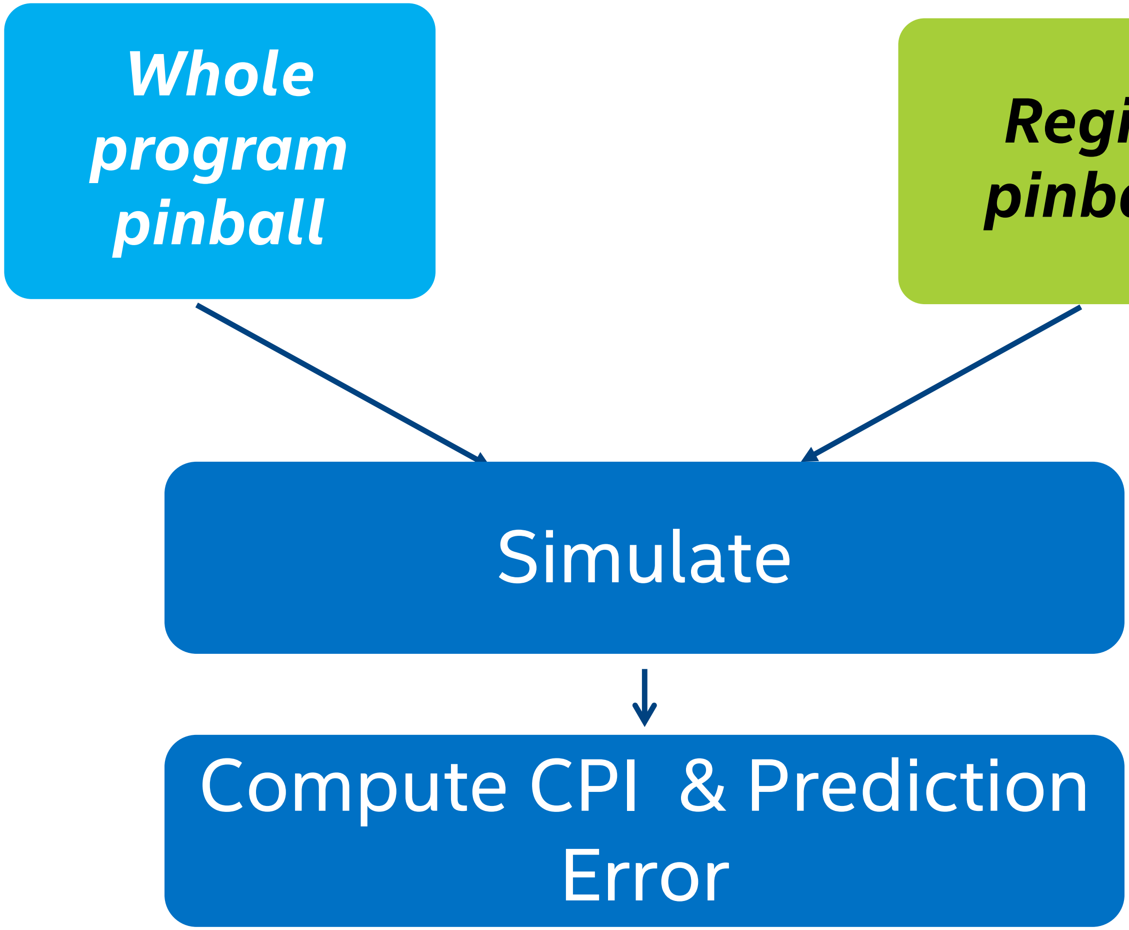
```
***  Finished running Sniper on region pinballs [sniper_regions]  ***    June 14,
 2014 15:22:28

***  Running Sniper on whole program pinballs [sniper_whole]  ***    June 14, 201
4 15:22:28

+++  Running Sniper on whole program pinball: whole_program.p10000-s10/omnetpp.p1
0000-s10_18060
/home/hgpatil/Workspace/Sniper/sniper-6.0/run-sniper -c nehalem-lite --no-cache-w
arming -d "sniper_results/whole_program.p10000-s10/omnetpp.p10000-s10_18060"  --p
inballs whole_program.p10000-s10/omnetpp.p10000-s10_18060 1> whole_program.p10000
-s10/omnetpp.p10000-s10_18060.sniper.txt 2>&1

***  omnetpp.p10000-s10_18060  ***    June 14, 2014 15:23:18
```

PinPoints: Validation


Whole program pinball

Region pinballs

Simulate

Compute CPI & Prediction Error

Calculate prediction error

- Apply weights to CPI for regions to get predicted CPI

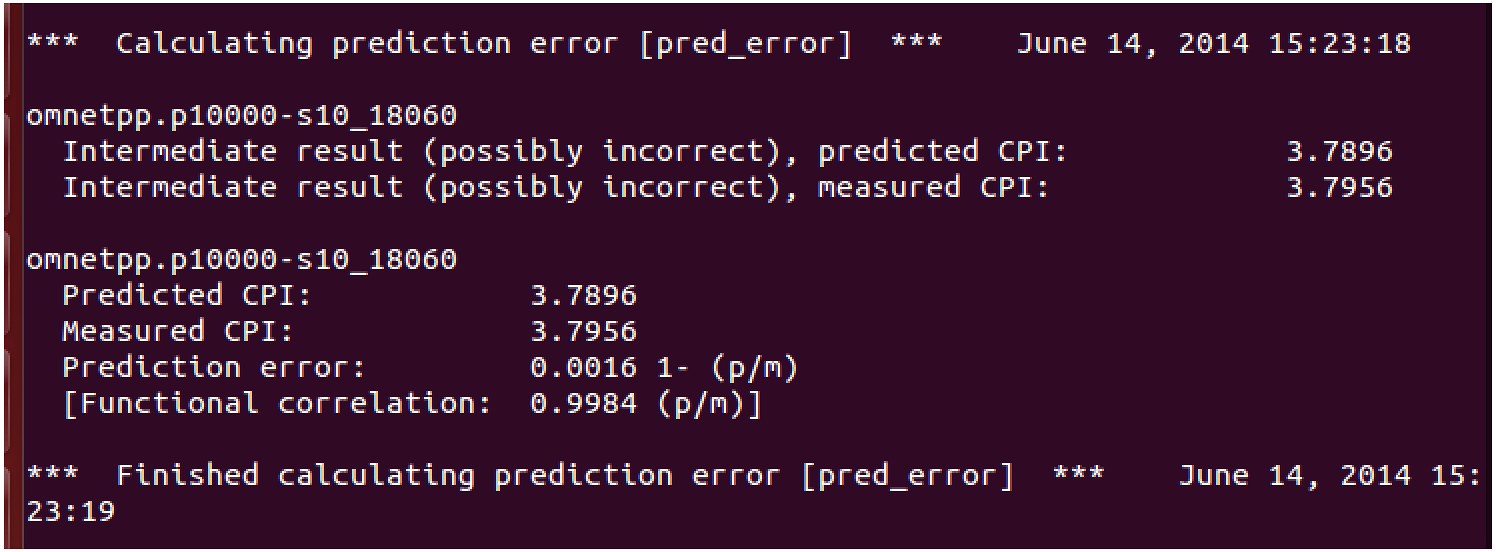
$$\mathbf{(wt\_r_1 * CPI\_r_1) + (wt\_r_2 * CPI\_r_2) + \ldots (wt\_r_n * CPI\_r_n)}$$

- Measured CPI from SniperLite run on WP pinballs
- Prediction error is deviation from measured whole program CPI

PE = 1 - (predicted CPI/measured CPI)

- Normally accepted values +/- 5%

```
sniper_pinpoints.py --cfg demo.cfg  -c >& out_6.txt
```



```
***  Calculating prediction error [pred_error]  ***    June 14, 2014 15:23:18

omnetpp.p10000-s10_18060
  Intermediate result (possibly incorrect), predicted CPI:           3.7896
  Intermediate result (possibly incorrect), measured CPI:            3.7956

omnetpp.p10000-s10_18060
  Predicted CPI:             3.7896
  Measured CPI:              3.7956
  Prediction error:          0.0016 1- (p/m)
  [Functional correlation:  0.9984 (p/m)]

***  Finished calculating prediction error [pred_error]  ***    June 14, 2014 15:
23:19
```

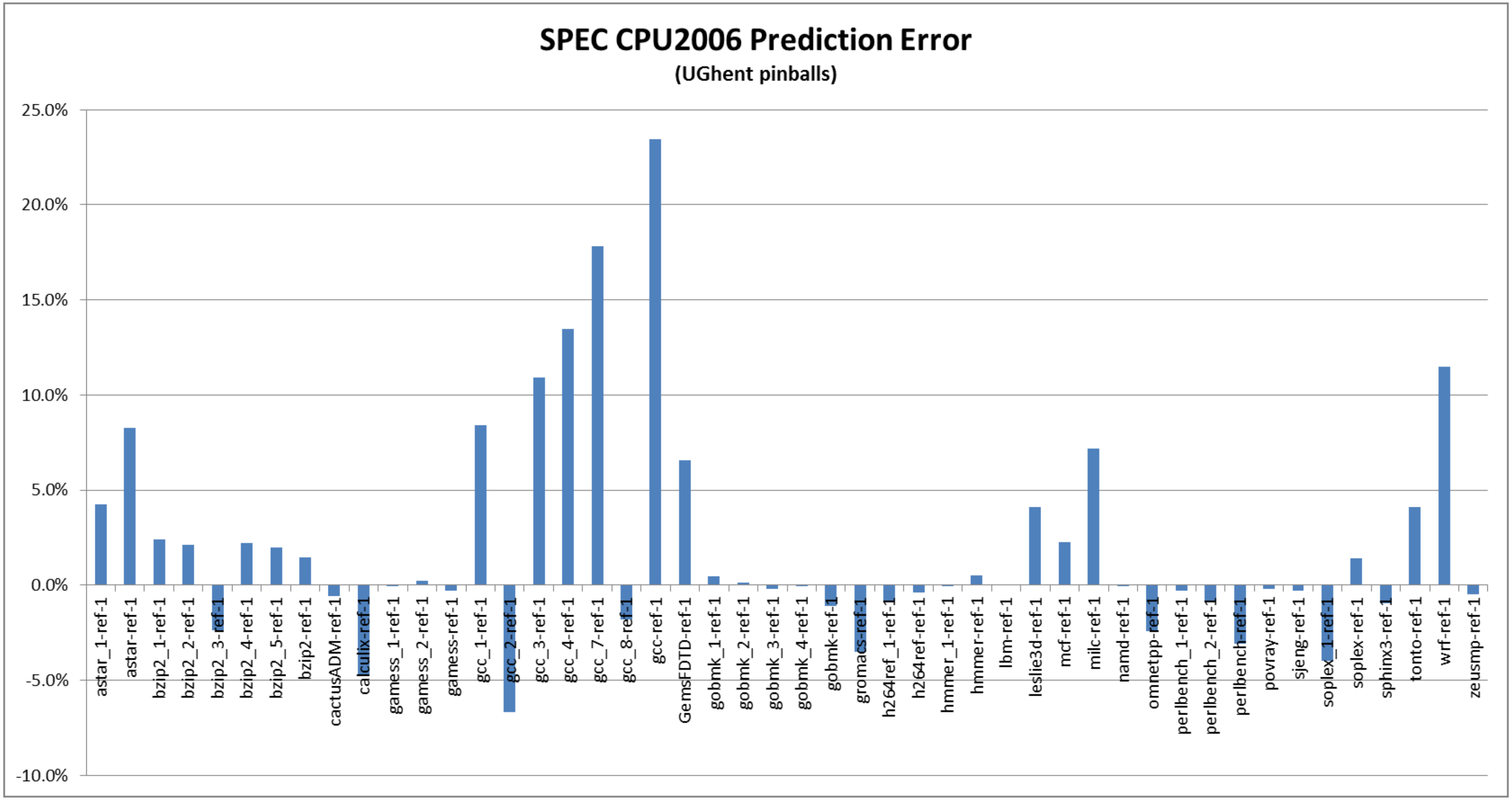
Tune for Better Representative Regions

Why tune?

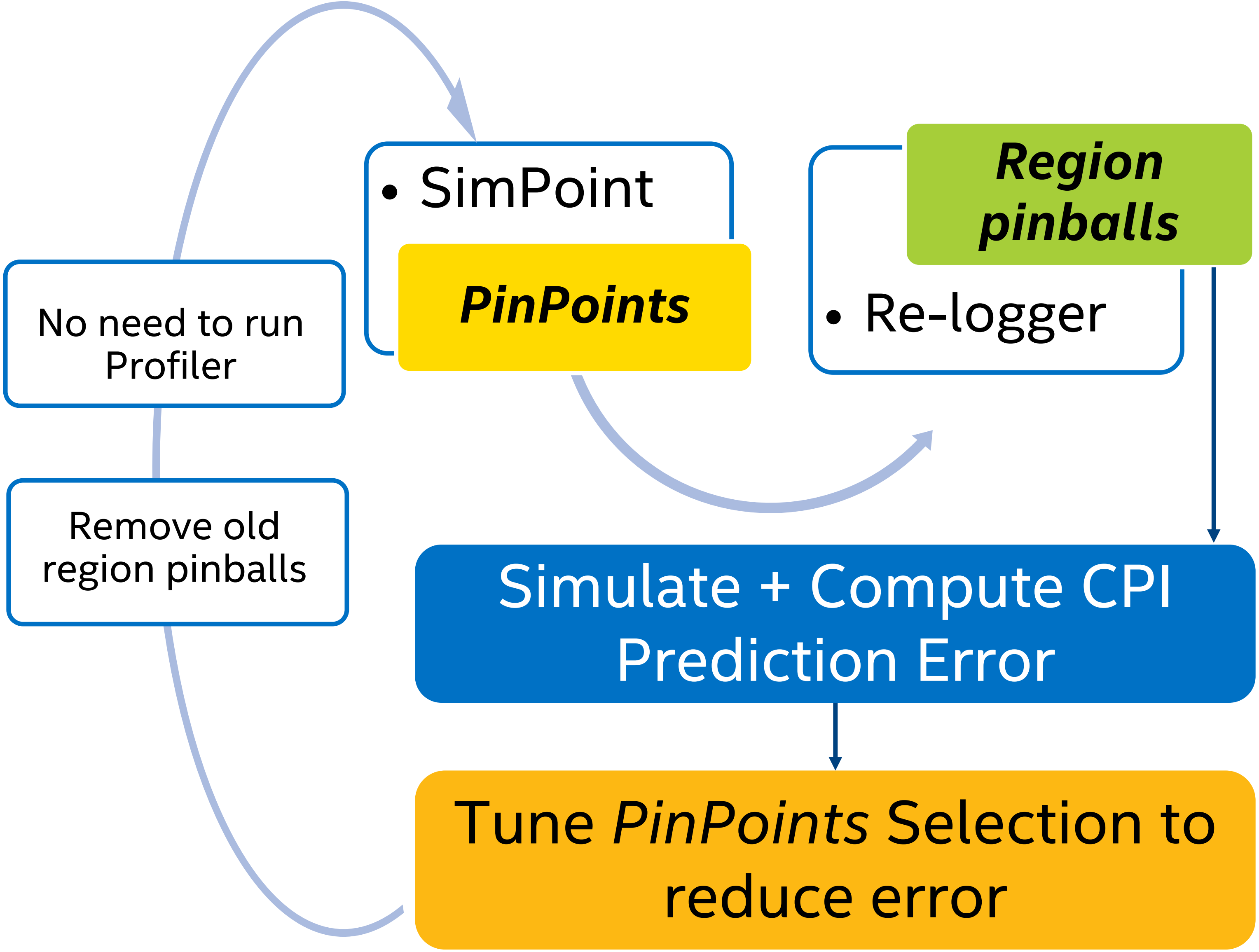
High prediction error indicates regions don’t accurately reflect whole program behavior

- Iterative process to chose alternative set(s) of representative regions which are more predictive of whole program behavior

Example of prediction error



PinPoints: Tuning



- SimPoint

PinPoints

Region pinballs

- Re-logger

No need to run Profiler

Remove old region pinballs

Simulate + Compute CPI Prediction Error

Tune *PinPoints* Selection to reduce error

How to tune

Add SimPoint options to generate alternative set(s) of representative regions

Same process used for each iteration:

- Remove old region pinballs
- Use same BBV file (no profiler)
- Rerun SimPoint, relog, SniperLite on just new regions (not WP pinballs) and get prediction error for new regions (-spTc)

- Repeat until acceptable prediction error achieved

Tuning is an iterative process

Cleanup old data before tuning

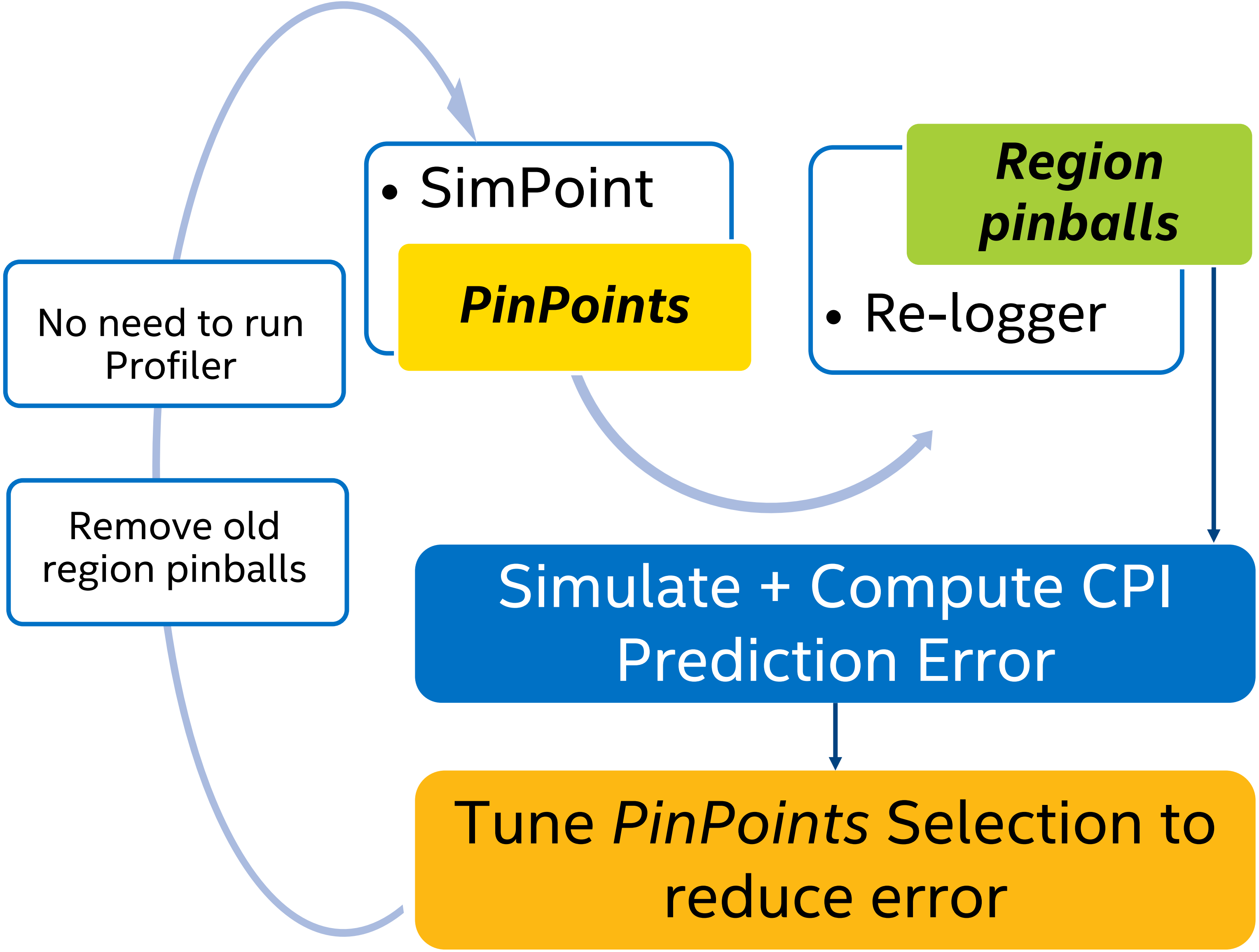
If have poor prediction error, must remove old region pinballs

- Tuning will create new pinballs with different names.
- Pinballs located in *.pp directory
- Need to move/remove this directory for each tuning iteration

```
$ ls | grep "\.pp"
omnetpp.p10000-s10_29800.pp/
$ rm -rf omnetpp.p10000-s10_29800.pp
```

- Do NOT remove whole_progam* or *.Data directories

PinPoints: Tuning



- SimPoint

PinPoints

Region pinballs

- Re-logger

No need to run Profiler

Remove old region pinballs

Simulate + Compute CPI Prediction Error

Tune *PinPoints* Selection to reduce error

SimPoint pseudo code

```
best_cluster= none
execute binary search from K=1  to MAXK
  best_K_cluster = none
  for M=1 to numInitSeed
      use random number to get new set of K initial clusters
      for N=1 to iters
        use k-means to generate cluster_M_N
        If cluster_M_N > best_K_cluster
            best_K_cluster = cluster_M_N
   if best_K_cluster > best_cluster
      best_cluster = best_K_cluster
```

">" comparison using *BIC* score
Three nested loops, use best clustering found

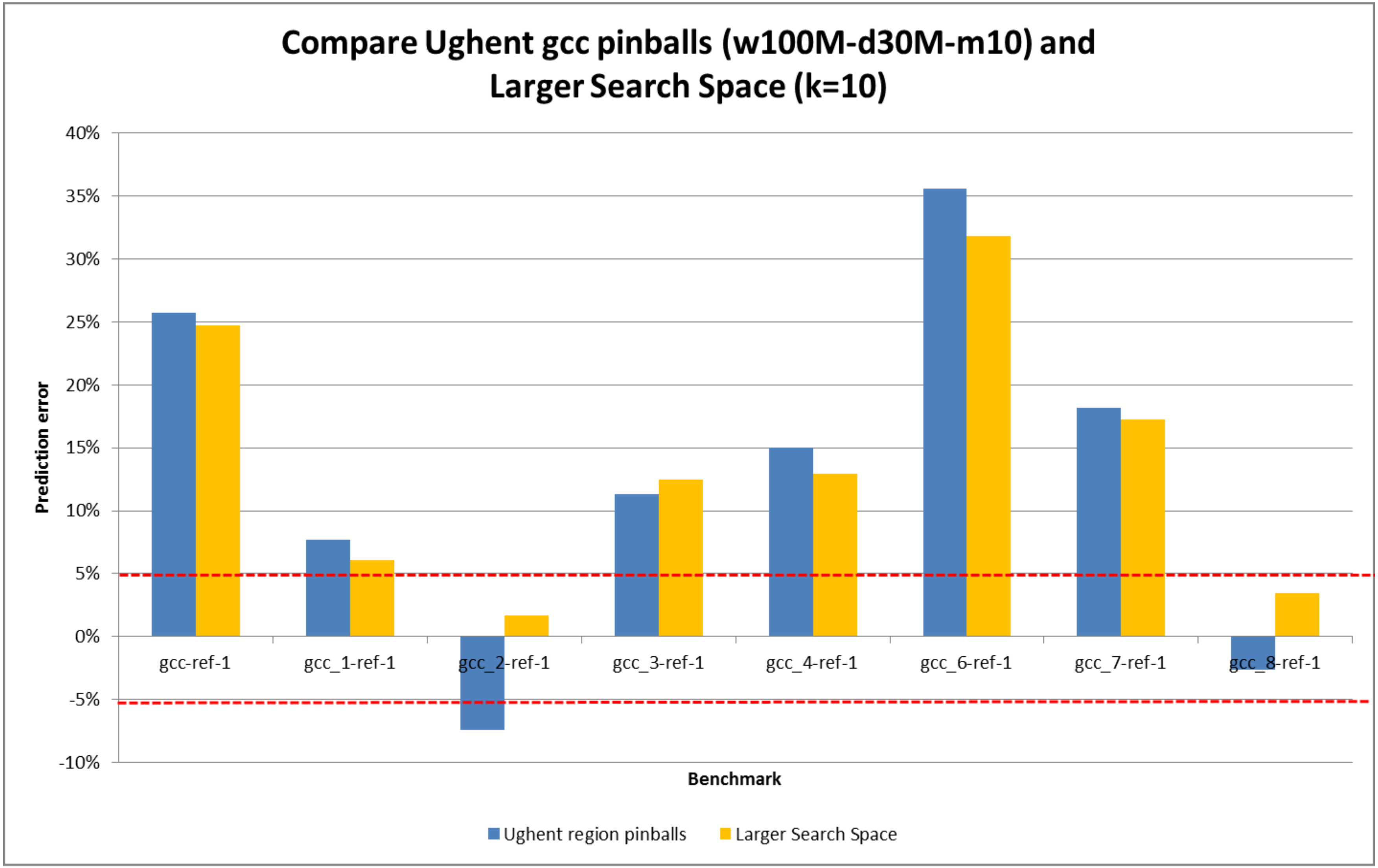
Tuning : method 1

Add options to SimPoint invocation to increase ‘search space’

- SimPoint limits number of random initializations for each cluster of size k
  - `--numInitSeeds X` Default value is 5
- SimPoint limits number of k-means iterations per clustering
  - `--iters X` Default value is 100

```
sniper_pinpoints.py --cfg demo.cfg  -spTc \
--simpoint_options ‘--numInitSeeds 150 --iters 250’ \
>& out_7.txt
```

Increasing search space



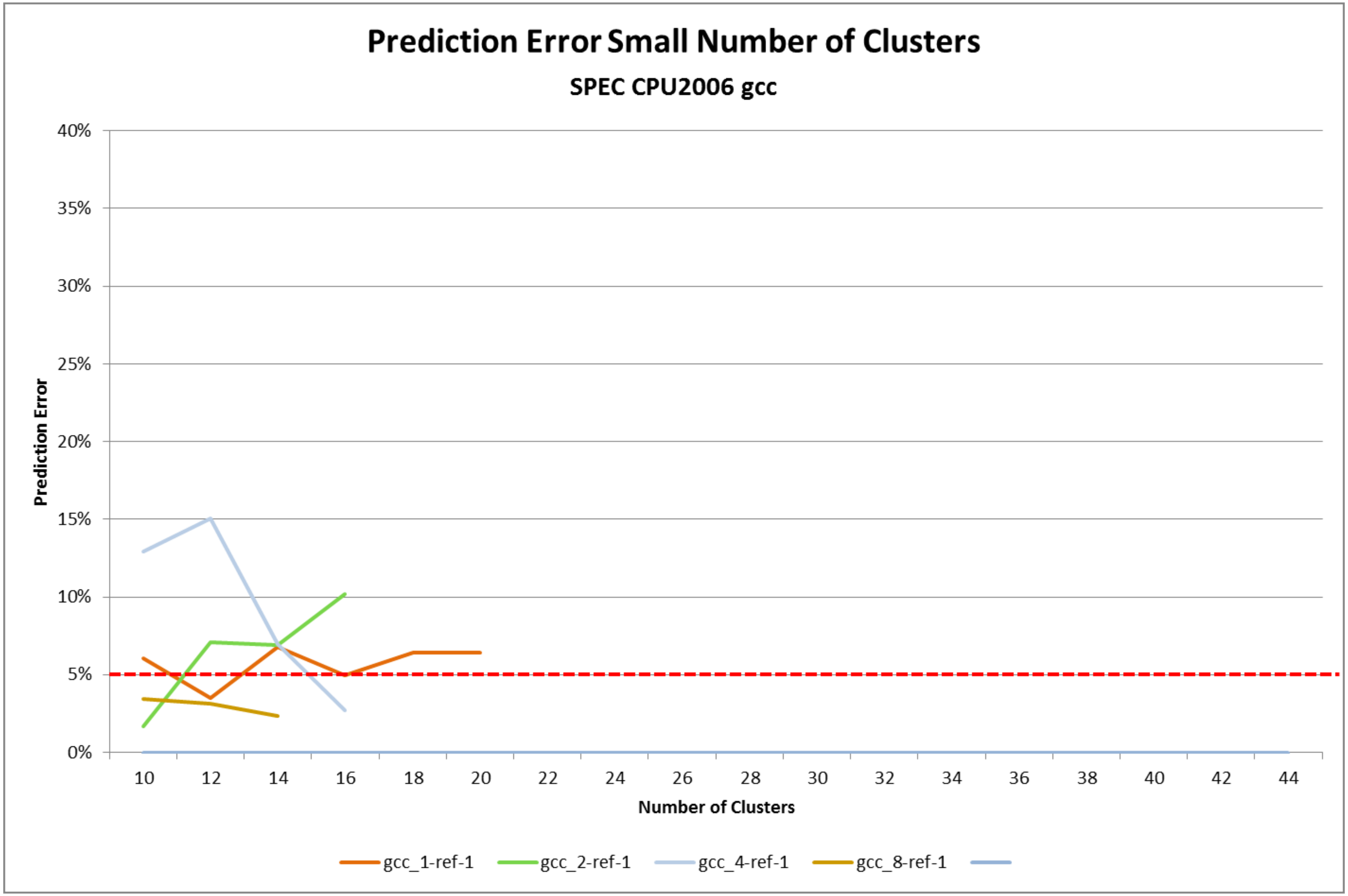
Tuning: method 2

Add option to SimPoint invocation to increase number of clusters

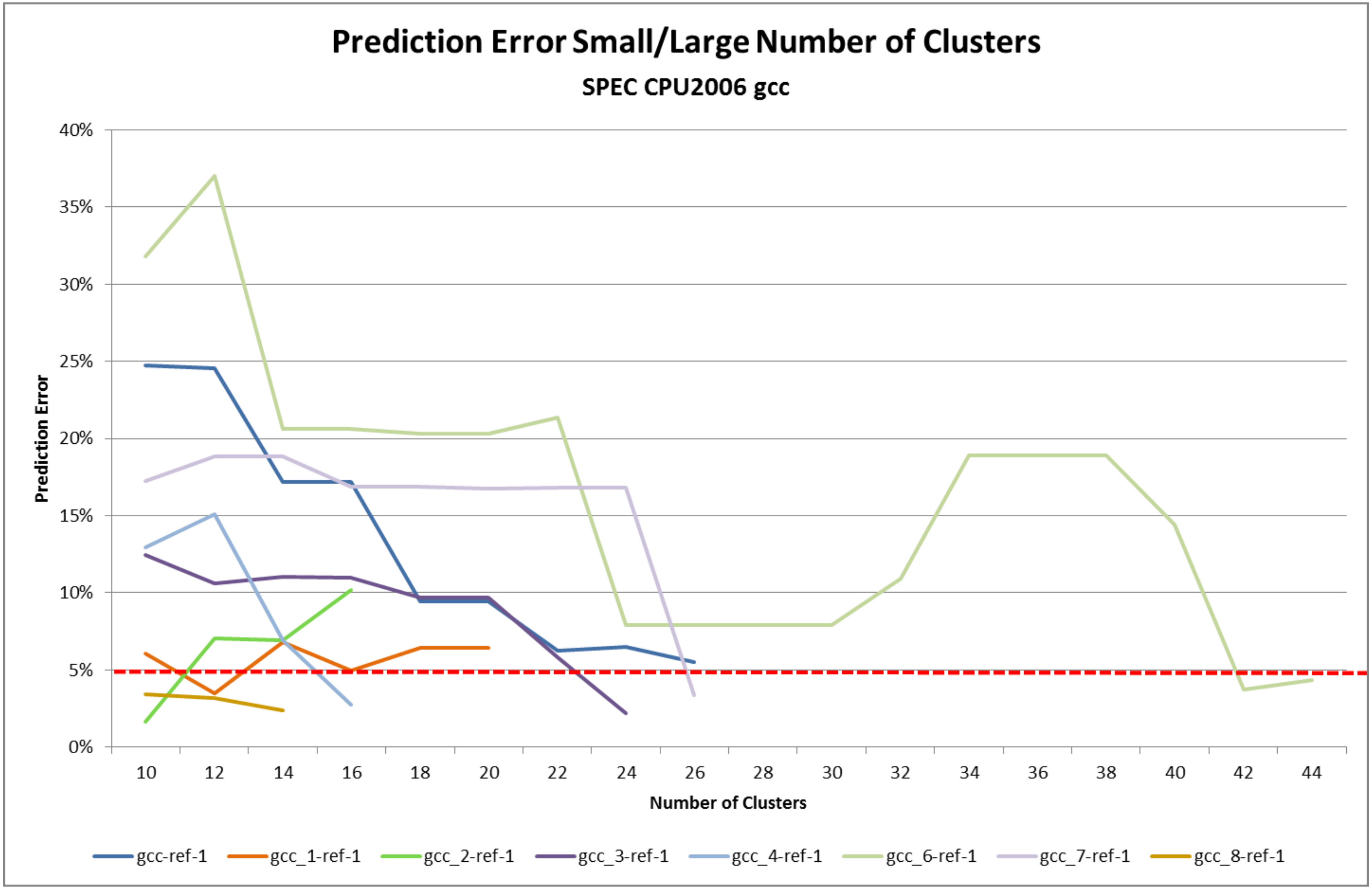
- Increase max number of clusters in which to search for best clustering
  - `-k 5:MAX`
  - Also changes from binary search to linear search.

```
sniper_pinpoints.py --cfg demo.cfg  -spTc \
--simpoint_options '-k 5:20 --numInitSeeds 150 --iters 
250' \
>& out_8.txt
```

Tuning with a small cluster count



Tuning with a large cluster count



How to Run Sniper on a Pinball

Run Sniper/SniperLite on a pinball

- Add option which gives location of pinball files

```
--pinballs pinball_path
```

- Sniper/SniperLite always creates same set of file names
- Add option to define specific output directory

```
-d output_dir
```

Sniper/SniperLite can run either a program or a pinball

Command to run with a pinball

```
$SNIPER_ROOT/run-sniper -c nehalem-lite --roi-script \
-d cpu2006-gcc-ref-1_t0r10_sniper_out/  \
--pinballs cpu2006-gcc-ref-1.pp/cpu2006-gcc-ref- \
1_t0r10_warmup100001500_prolog0_region30000003_epilog0_010_0-06994.0
```

```
[SNIPER] Start
[SNIPER] -----------------------------------------------------------------------------------------
[SNIPER] Sniper using SIFT/trace-driven frontend
[SNIPER] Running in script-driven instrumenation mode (--roi-script)
[SNIPER] Using CACHE_ONLY mode for warmup
[SNIPER] Using CACHE_ONLY mode for detailed
[SNIPER] -----------------------------------------------------------------------------------------
[TRACE:0] -- DONE --
[SNIPER] End
[SNIPER] Elapsed time: 29.82 seconds
```

Schedule

~~8:45 – 9:30 Intro + Background (Harish)~~

~~9:30 – 10 Demo Part I (Mack)~~

~~10 – 10:30 Break~~

~~10:30 – 11:15 Demo Part II (Mack)~~

11:15 – 11:45 Advanced Topics (Harish)

11:45 – noon Wrap-up + Q&A (all)

Advanced Topics

Pinball details, Multi-threading, Multi-processing

Outline

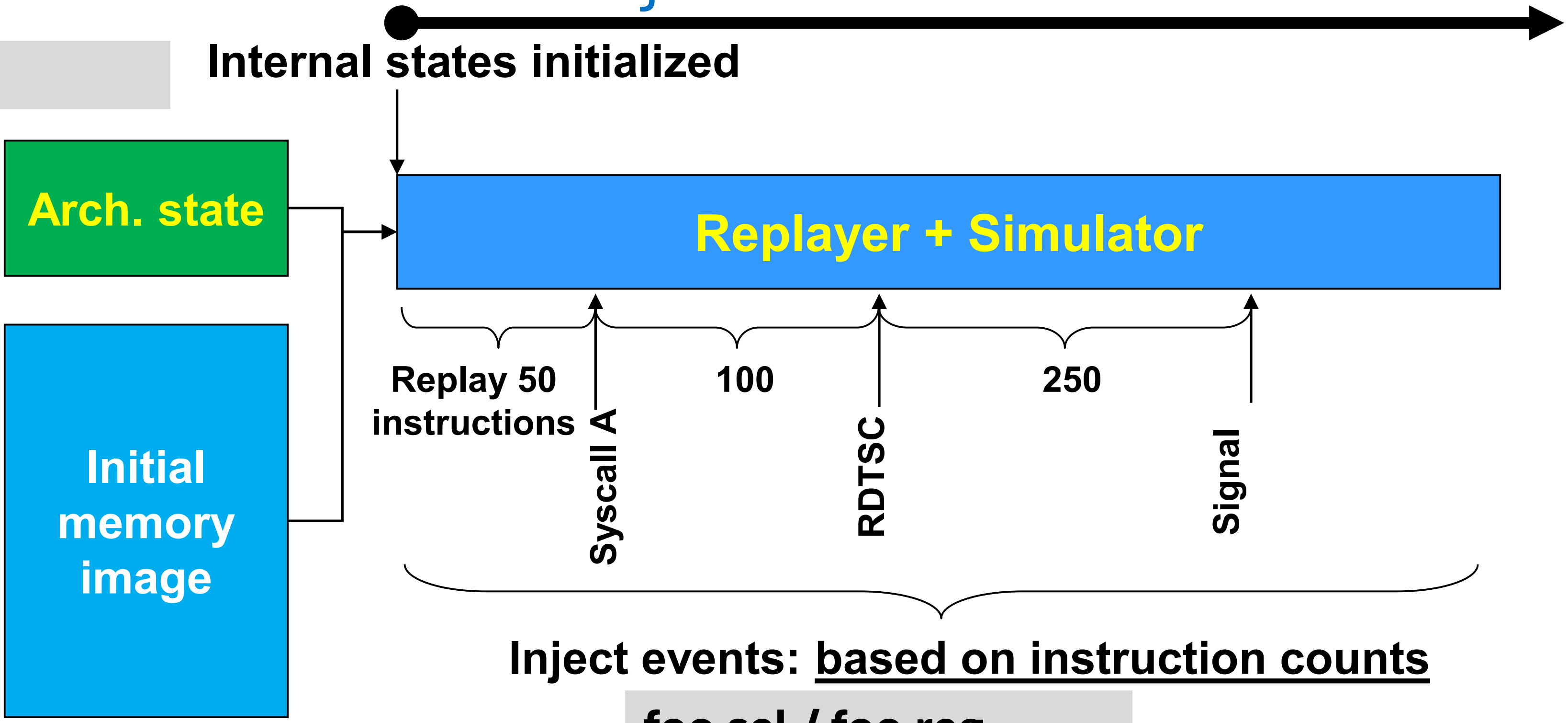
Pinballs :

- What they are
- How to use them for simulation

Multi-threaded simulation: Alternatives

Pinball (ST) = Initial memory/register + injections

foo.reg



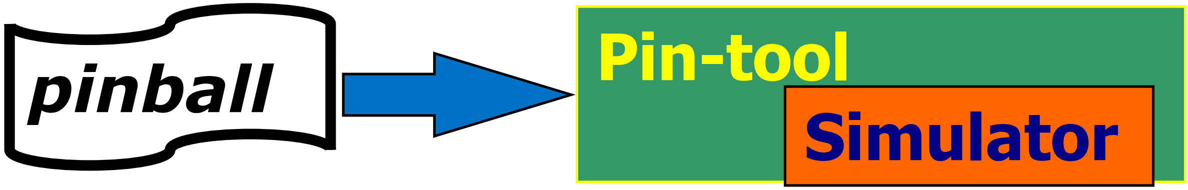
Inject events: based on instruction counts

foo.sel / foo.reg
(injections)

foo.text

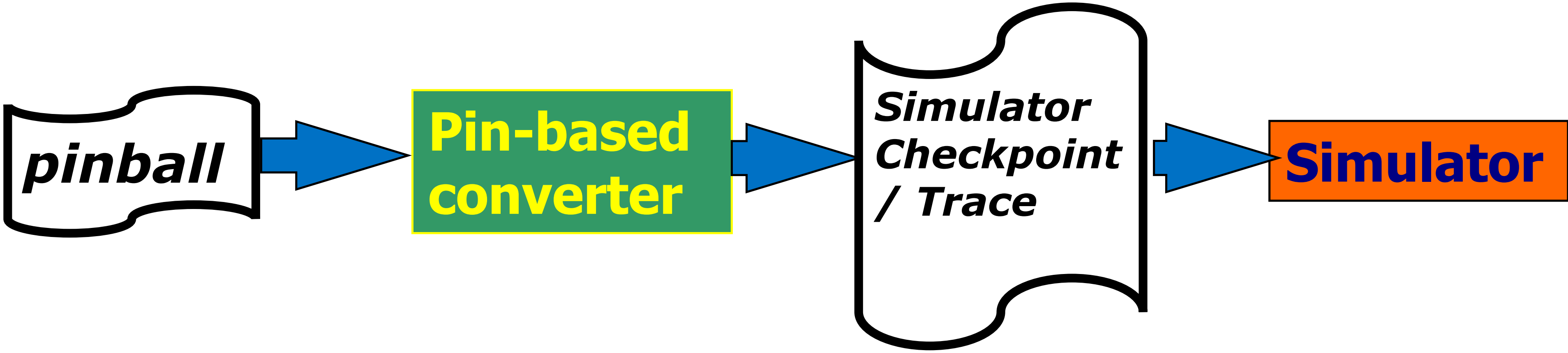
- **System calls** : skipped by injecting next rip/ memory changed
- **CPUID, RDTSC** : affected registers injected
- **Signals/Callbacks** : New register state injected

Pinball-based Simulation: Two Usage Models

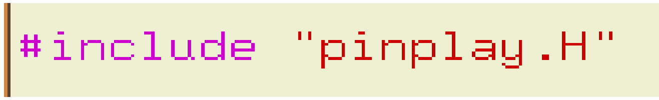


1. Pin-based simulators: e.g. Sniper from Ghent Univ.

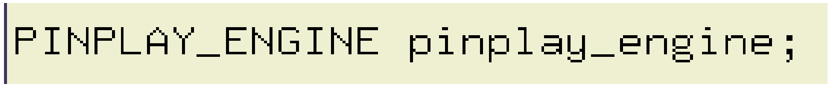
2. Looking for collaboration : QEMU-based/ other simulators



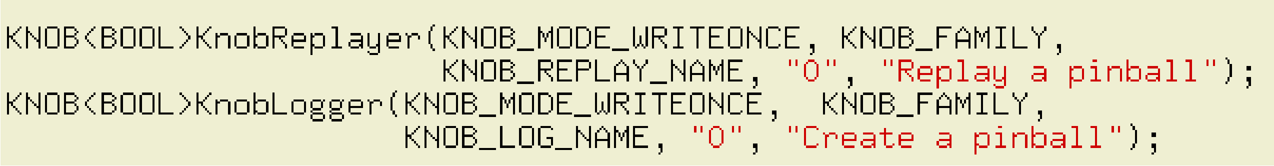
Enabling a Pintool for PinPlay



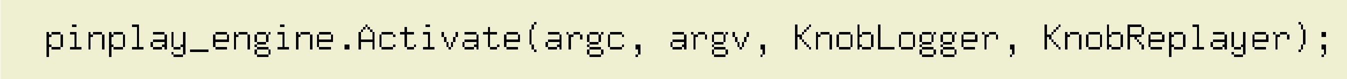
```
#include "pinplay.H"
```



```
PINPLAY_ENGINE pinplay_engine;
```



```
KNOB<BOOL>KnobReplayer(KNOB_MODE_WRITEONCE, KNOB_FAMILY,
                       KNOB_REPLAY_NAME, "0", "Replay a pinball");
KNOB<BOOL>KnobLogger(KNOB_MODE_WRITEONCE,  KNOB_FAMILY,
                     KNOB_LOG_NAME, "0", "Create a pinball");
```



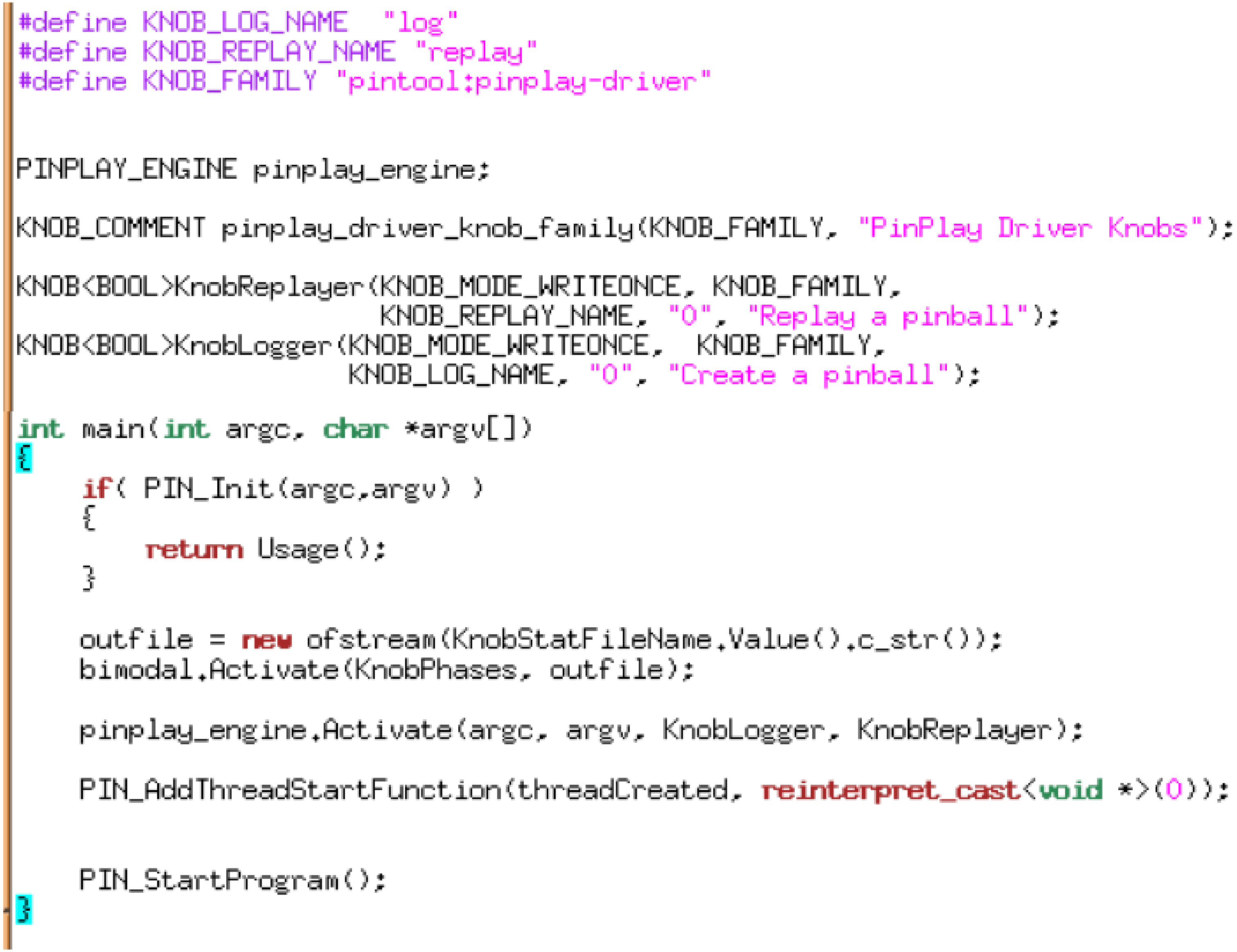
```
  pinplay_engine.Activate(argc, argv, KnobLogger, KnobReplayer);
```

<u>Link</u> `in` ***`libpinplay.a, libzlib.a, libbz2.a`***

<u>Restrictions:</u>

1. **PinTool shouldn’t change application control flow**
2. **Image API not available during <u>replay</u>**

Example: pinplay-branch-predictor.cpp



```
#define KNOB_LOG_NAME   "log"
#define KNOB_REPLAY_NAME "replay"
#define KNOB_FAMILY "pintool:pinplay-driver"


PINPLAY_ENGINE pinplay_engine;

KNOB_COMMENT pinplay_driver_knob_family(KNOB_FAMILY, "PinPlay Driver Knobs");

KNOB<BOOL>KnobReplayer(KNOB_MODE_WRITEONCE, KNOB_FAMILY,
                       KNOB_REPLAY_NAME, "0", "Replay a pinball");
KNOB<BOOL>KnobLogger(KNOB_MODE_WRITEONCE,  KNOB_FAMILY,
                     KNOB_LOG_NAME, "0", "Create a pinball");

int main(int argc, char *argv[])
{
    if( PIN_Init(argc,argv) )
    {
        return Usage();
    }

    outfile = new ofstream(KnobStatFileName.Value().c_str());
    bimodal.Activate(KnobPhases, outfile);

    pinplay_engine.Activate(argc, argv, KnobLogger, KnobReplayer);

    PIN_AddThreadStartFunction(threadCreated, reinterpret_cast<void *>(0));


    PIN_StartProgram();
}
```

PinPlay-enabled PinTools : 3 Modes

1. **Regular Analysis mode**

```
$ pin -t pintool -- test-program
```

Normal output
+ *Analysis output*

2. **Logging Mode**

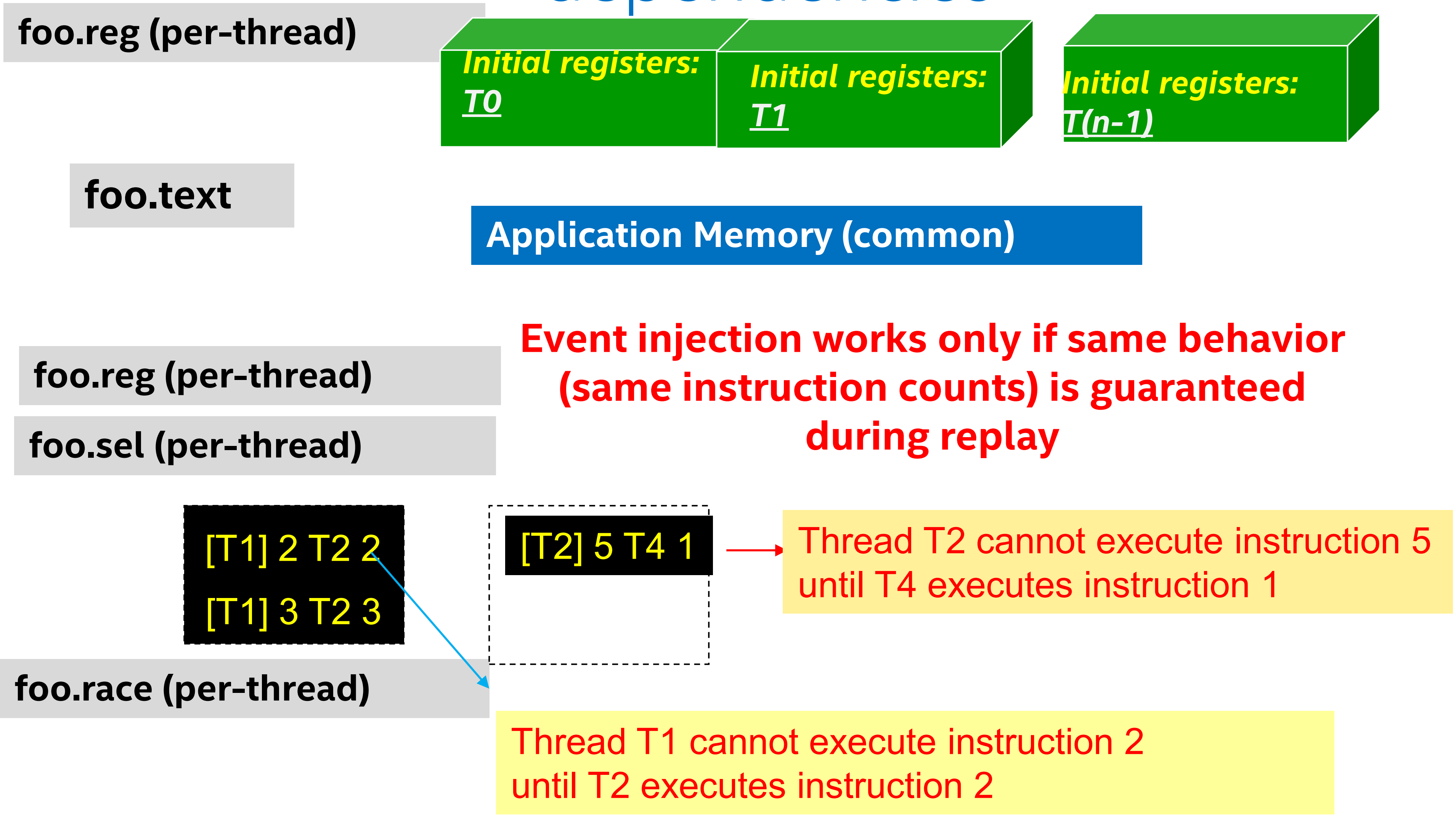
```
$ pin -t pintool -log -log:basename pinball/foo -- test-program
```

3. **Replay Mode**

pinball

```
$ pin -t pintool -replay -replay:basename pinball/foo -- nullapp
```

Pinball (MT) : Pinball (ST) + Thread-dependencies



foo.reg (per-thread)

Initial registers: T0

Initial registers: T1

Initial registers: T(n-1)

foo.text

Application Memory (common)

Event injection works only if same behavior (same instruction counts) is guaranteed during replay

foo.reg (per-thread)

foo.sel (per-thread)

[T1] 2 T2 2

[T1] 3 T2 3

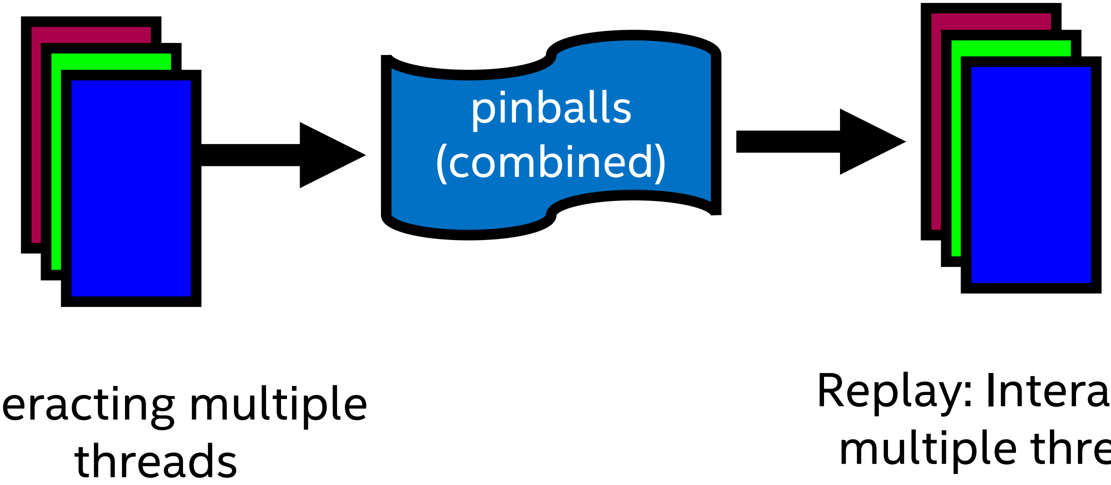
[T2] 5 T4 1

Thread T2 cannot execute instruction 5 until T4 executes instruction 1

foo.race (per-thread)

Thread T1 cannot execute instruction 2 until T2 executes instruction 2

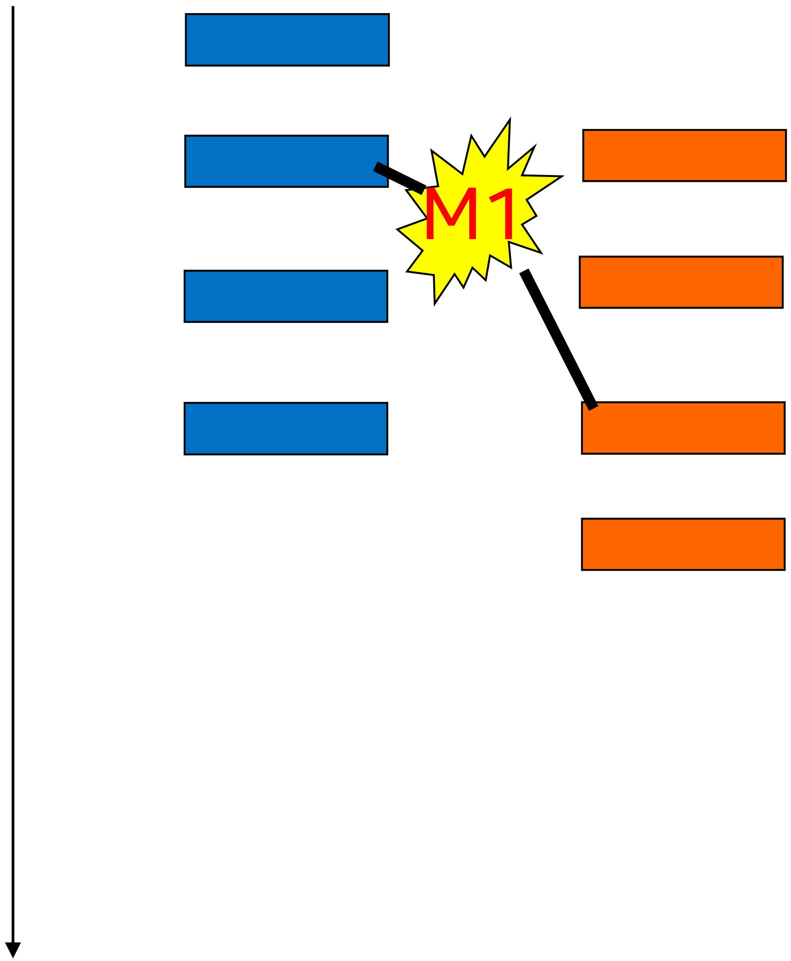
Model 1: Parallel Capture : Parallel Replay
For Multi-threaded Programs


pinballs
(combined)
Interacting multiple threads
Replay: Interacting multiple threads

Useful for parallel analysis/simulation

[Can focus on one thread with –log:focus_thread]

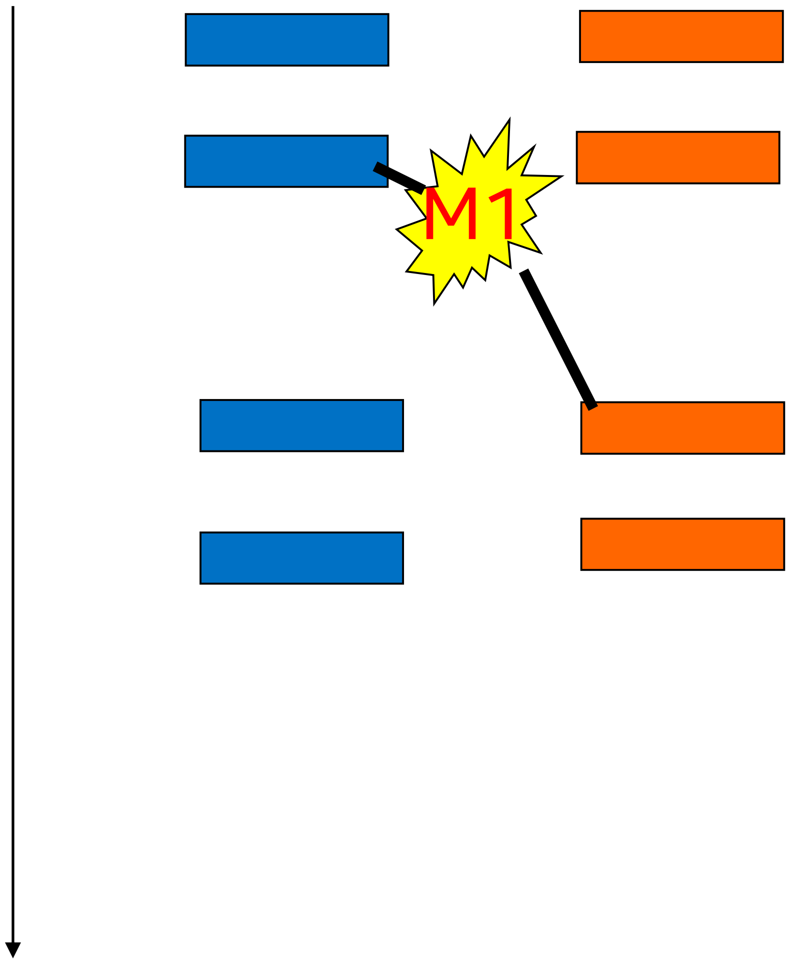
PinPlay's Determinism == Same Access Order for Conflicting Shared Memory Accesses



- Instructions from each thread replayed in program order

- RAW, WAW, WAW order for multiple threads is preserved

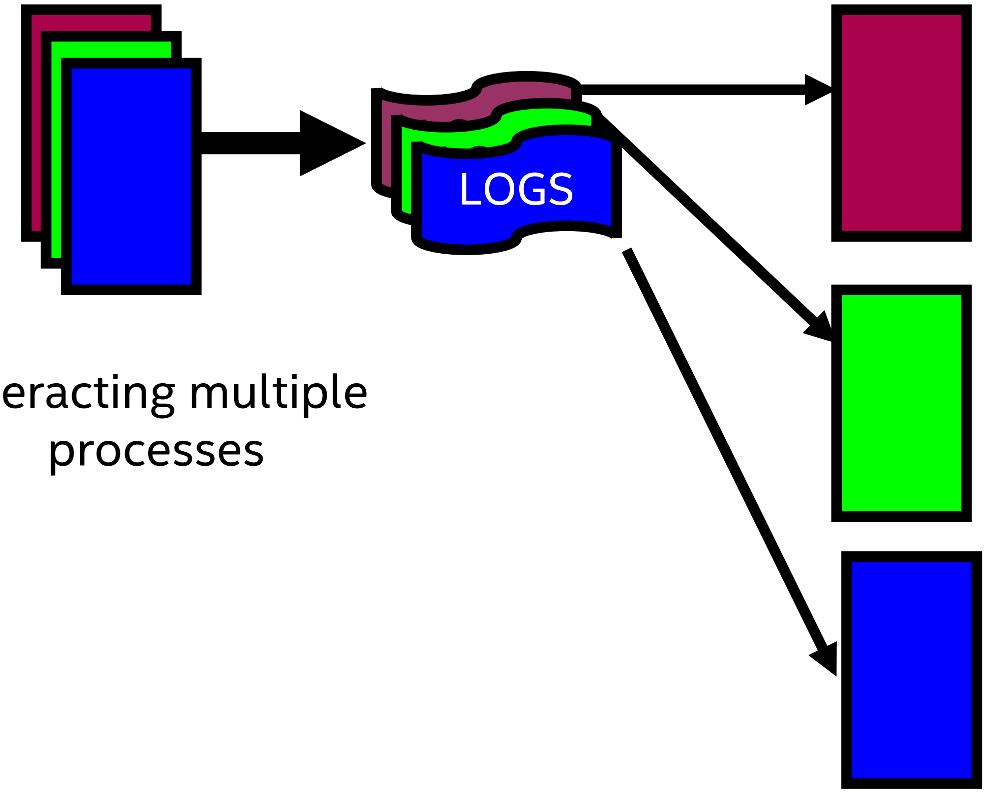
Relative Speed of Threads will Change During Replay



- Instructions from each thread replayed in program order

- RAW, WAR, WAW order for multiple threads is preserved

Model 2: Parallel Capture : Isolated Replay For Multi-process Programs



Interacting multiple processes

Per-process Replay

- multi-process ➔ multi-programmed

Challenges in multi-threaded region selection

Simulation

1. Deterministic simulation (with pinballs): too restrictive
2. Unconstrained simulation (with pinballs):
   - System calls not allowed in pinballs or need to be emulated
   - No instruction-count based memory injection possible

Projection (instruction count change)

SimPoint projection is instruction-count based
change in control flow → Change in instruction count
→ **Projection formula invalid**

PinPoints for multi-threaded programs

1. Per thread pinball: ***-log:focus_thread tid***
   Whole-program logging, PinPoints generation same as single-threaded program

2. Truly multi-threaded ("co-operative") pinball: ***Work in progress***
   - Simulation: system-call emulation, no injections
   - Projection: "*BarrierPoint*" work from Ghent university

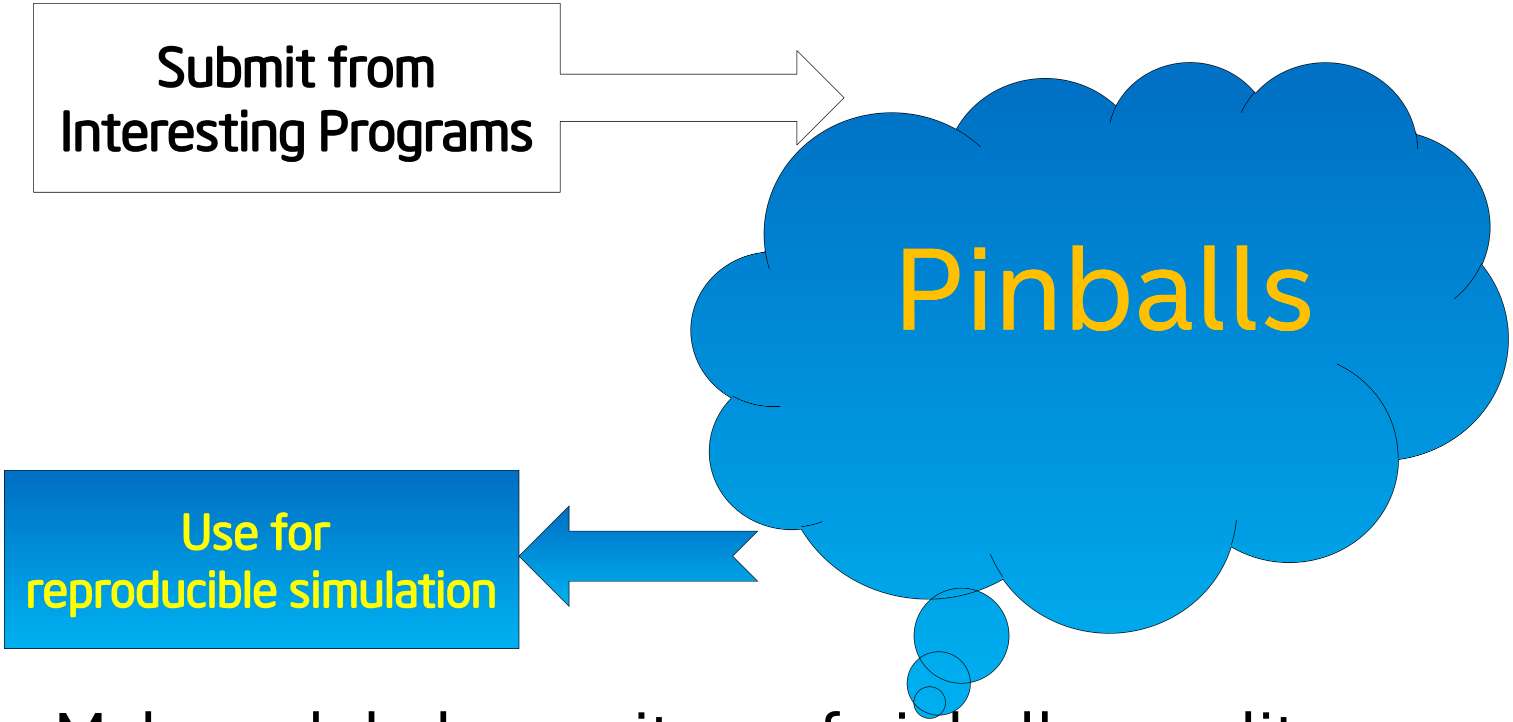
Conclusions

Demonstrated representative simulation region selection

1. How to use PinPlay for recording execution (pinballs)
2. How to profile and find representative regions using PinPlay and SimPoint, and create checkpoints (pinballs)
3. How to find the quality of selected simulation region
4. How to tune the selection for better quality

PinPlay and Sniper enable creation of high quality simulation region

Call for action


Submit from Interesting Programs
Pinballs
Use for reproducible simulation

1. Make a global repository of pinballs a reality

2. Pinball converter for QEMU-based/other simulators

References

Pin: Pin: Building Customized Program Analysis Tools with Dynamic Instrumentation; Chi-Keung Luk, Robert Cohn, Robert Muth, Harish Patil, Artur Klauser, Geoff Lowney, Steven Wallace, Vijay Janapa Reddi, and Kim Hazelwood. *Proceedings of the 2005 ACM SIGPLAN conference on Programming language design and implementation*.

PinPoints: Pinpointing Representative Portions of Large Intel® Itanium® Programs with Dynamic Instrumentation; Patil, H., Cohn, R., Charney, M., Kapoor, R., Sun, A., and Karunanidhi, A. In Proceedings of the *37th Annual IEEE/ACM international Symposium on Microarchitecture* (Portland, Oregon, December 04 - 08, 2004). **Nominated for Micro 2004 Best Paper Award.**

PinPlay: PinPlay: A Framework for Deterministic Replay and Reproducible Analysis of Parallel Programs; Harish Patil, Cristiano Pereira, Mack Stallcup, Gregory Lueck, James Cownie. CGO 2010. **CGO 2010 Best Paper Award Winner!**

SimPoint : Automatically Characterizing Large Scale Program Behavior ; Timothy Sherwood, Erez Perelman, Greg Hamerly and Brad Calder. In proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems, October 2002.

Sniper: Sniper: Exploring the Level of Abstraction for Scalable and Accurate Parallel Multi-Core Simulation; Trevor E. Carlson; Wim Heirman; Lieven Eeckhout. In proceedings of International Conference for High Performance Computing, Networking, Storage and Analysis (SC), 2011.

Legal Disclaimer & Optimization Notice

INFORMATION IN THIS DOCUMENT IS PROVIDED “AS IS”. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.

Optimization Notice

Intel’s compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804

intel®