PinPoints: A methodology for *Simulation Region Selection*, *Validation*, and *Simulator Staging* with Intel Pin and SDE

Harish Patil
Principal Engineer, Development Tools Software, Intel Corporation
With input from Alen Sabu (National University of Singapore)

November 7th, 2024
Last modified: 14th November 2024

https://github.com/intel/pinplay-tools/tree/main/PinPoints

Architecture Simulation : Key Questions

Where to simulate?

Regions of Interest (ROIs)
(<< 0.1 % of whole-program)
NOT whole-program

How to simulate?

Drive simulations for ROIs

Are the regions representative?

Region Selection Validation

PinPoints Methodology

Where to simulate?

SDE-profiler + SimPoint

Region Selection

How to simulate?

1. **Checkpoint (pinball) driven**
2. **Binary-driven**
   - **Pinballs → ELFies**

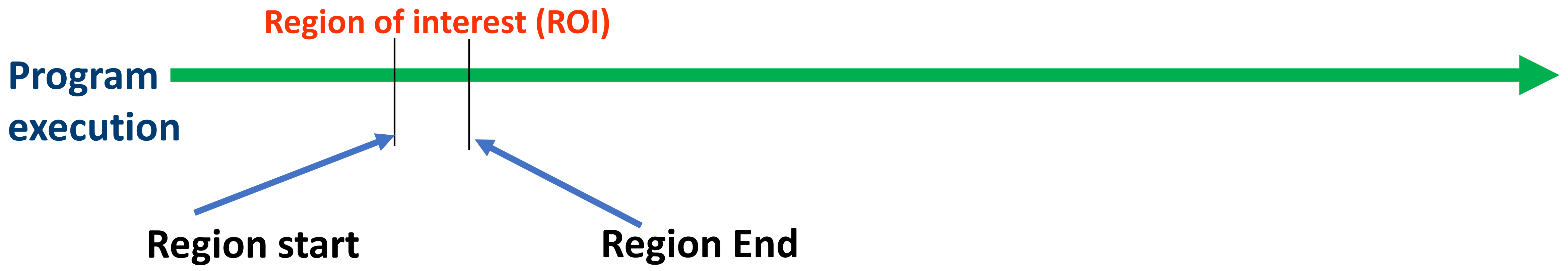
Simulator Staging

Are the regions representative?

ROIPerf:
CPU time-stamps with a Pin-probe tools

Region Validation

What is a "Region"?



Region specification: *Icount* vs *Pccount*

- ***ICount*: instruction count based**

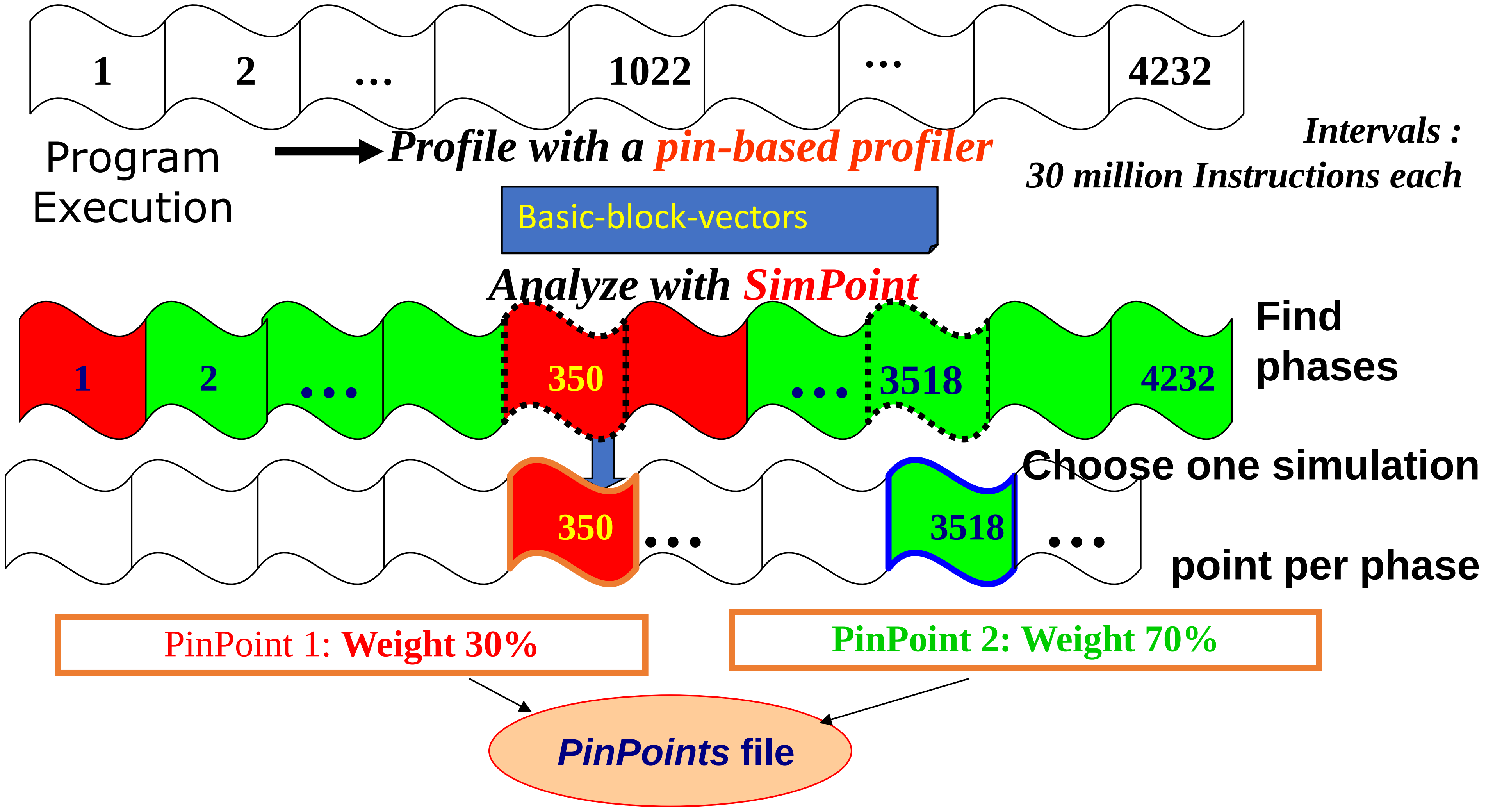
Region Start: *icount1*
Region End: *icount2 (or length after Start)*

Drawbacks:
1. Different tools count instructions differently (e.g. REP-prefixed instructions as 1 or N)
2. (Multi-threaded programs) Instruction counts can change widely from run to run

- *PCcount*: program-counter (pc) based (select marker PCs with invariant counts across runs)

Region Start: *PC1+count1*
Region End: *PC2+count2 (relative to Start)*

PinPoints = Pin + SimPoint (UC San Diego)



Two Phases => Two PinPoints

Projection Formulas

1. With weights (**ICount regions**) : ST only

   $CPI_{predicted} = \Sigma\, Weight_{region} * CPI_{region}$

$Cycles_{predicted} = CPI_{predicted}$ X Total_Instruction_Count (pathlength)

2. With multipliers (**PCCount regions**) :ST and MT (no instruction counts used directly)

   $Cycles_{predicted} = \Sigma\, Cycles_{region} * Multiplier_{region}$

   where

   $Multiplier_j = inscount_j \;/ \sum i{=}0..m \;\; inscount_i$

for m, the number of regions that are represented by the j^{th} representative region

Agenda

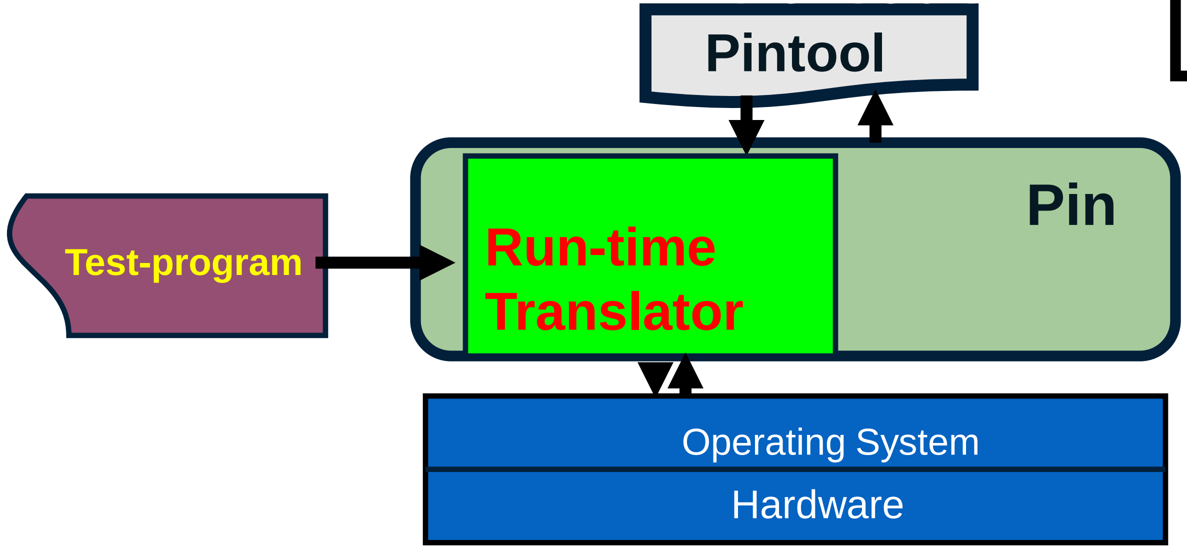
1. Tools Overview
2. Example/Demo
3. Discussion

Tools Overview

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>
```



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

Normal output
+ *Analysis output*

Pin: A Dynamic Instrumentation Framework from Intel
http://www.pintool.org (or search "Intel Pintool"

JIT Mode vs Probe Mode

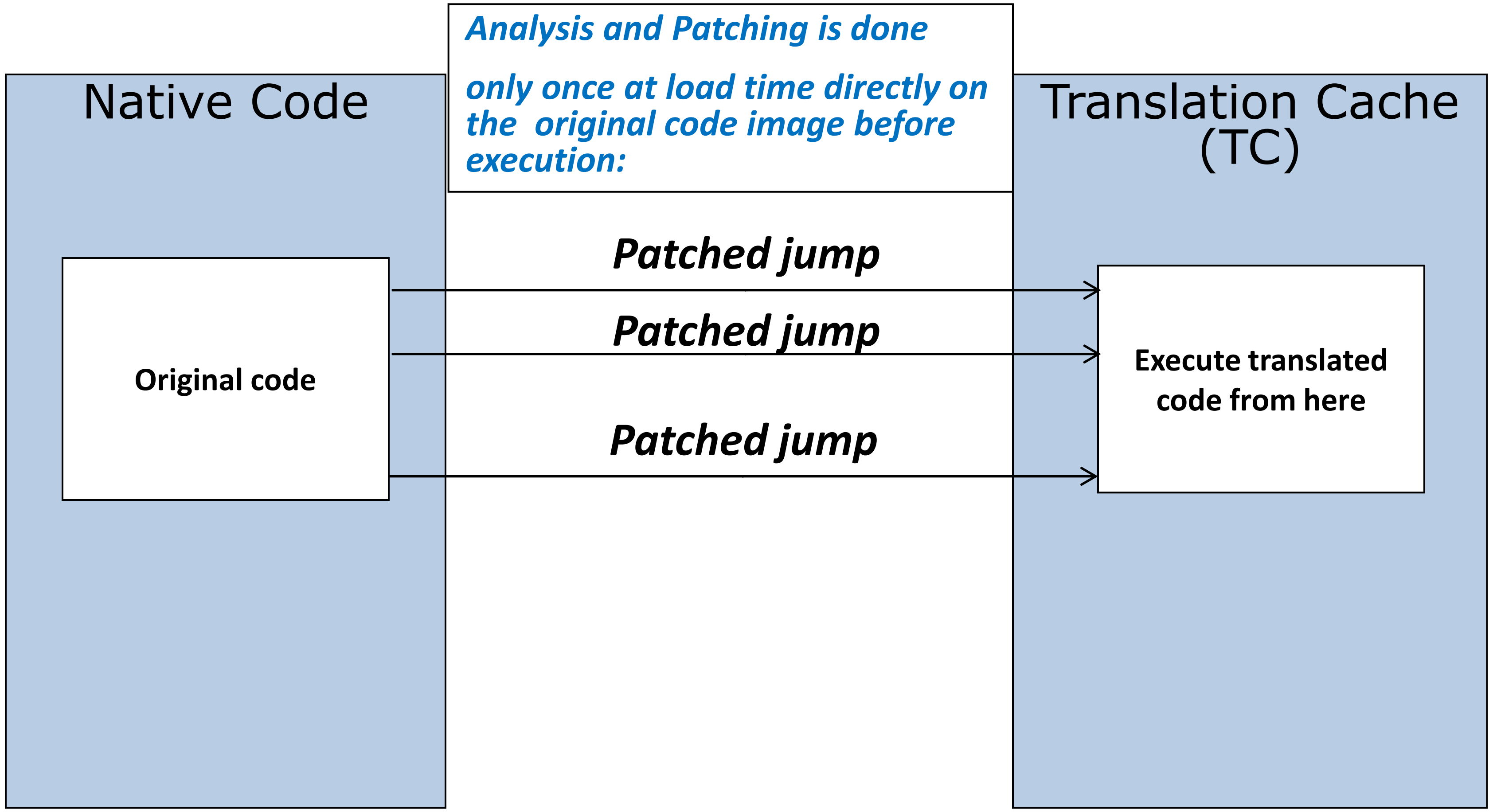
- JIT Mode
  - Pin creates a modified copy of the application on-the-fly
  - Original code never executes

➢More flexible, more common approach

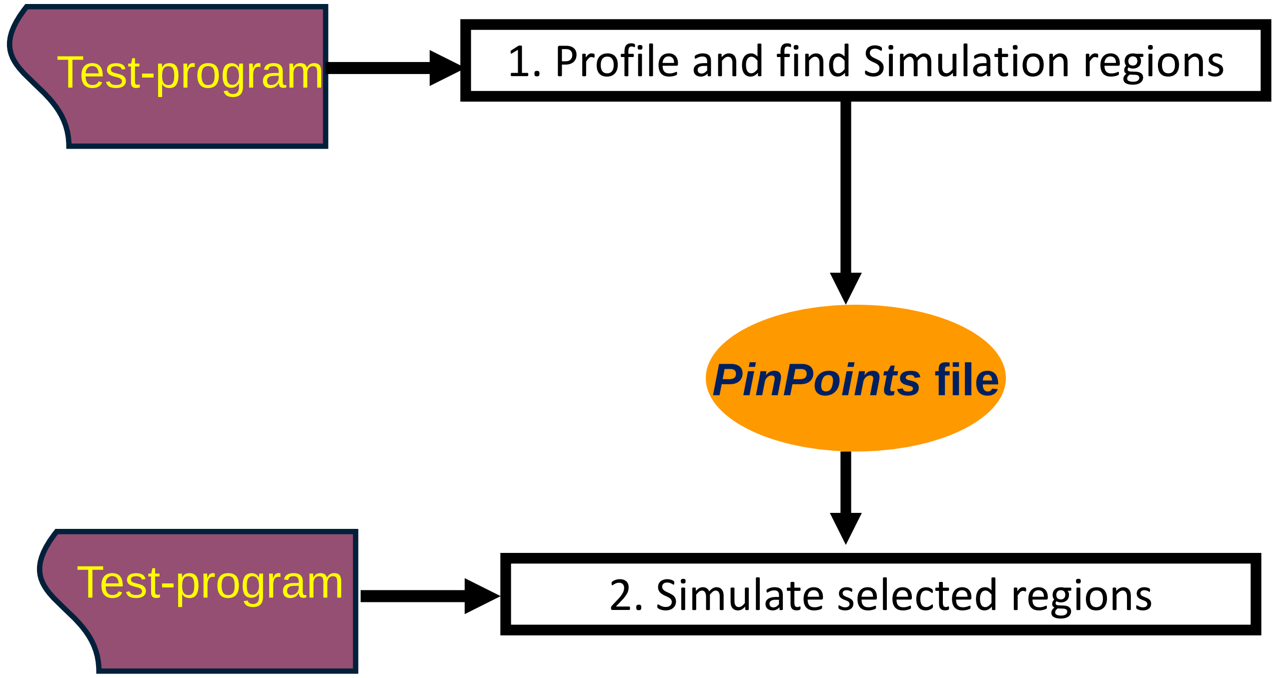
- Probe Mode
  - Pin modifies the original application instructions
  - Inserts jumps to instrumentation code (trampolines)

➢Lower overhead (less flexible) approach

Pin Probe Mode – General Flow


Analysis and Patching is done
only once at load time directly on the original code image before execution:
Native Code
Translation Cache (TC)
Original code
Patched jump
Patched jump
Patched jump
Execute translated code from here

PinPoints : The Repeatability Challenge

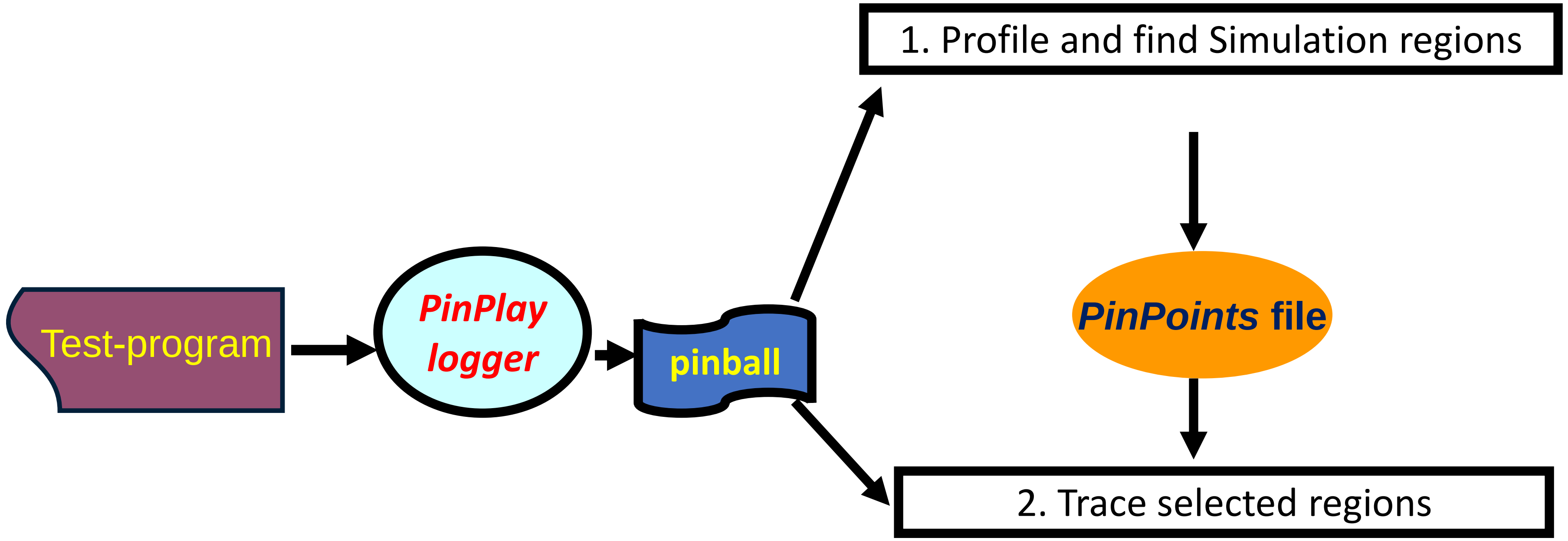

Test-program
1. Profile and find Simulation regions
PinPoints file
Test-program
2. Simulate selected regions

Problem: Two runs are not exactly same → PinPoints missed

Found this for 25/54 SPEC2006 runs!

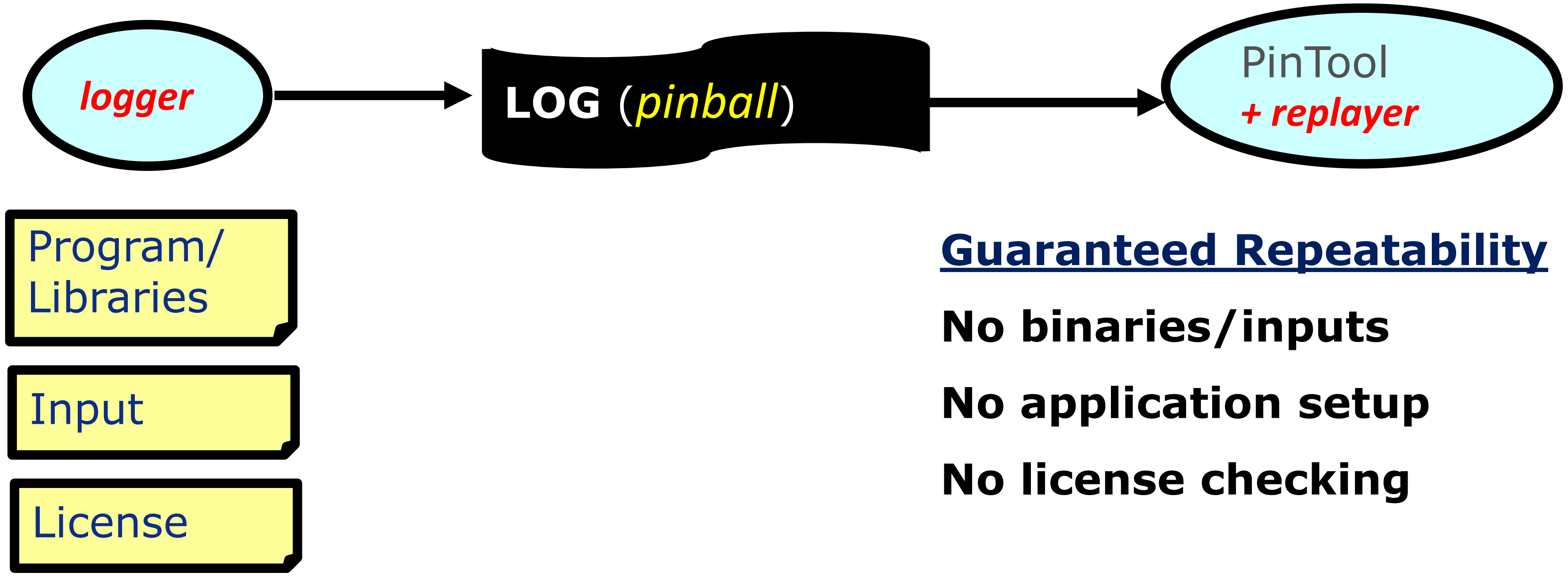
[*"PinPoints out of order" "PinPoint End seen before Start"*]

Enters PinPlay To Provide Repeatability


Test-program
PinPlay logger
pinball
1. Profile and find Simulation regions
PinPoints file
2. Trace selected regions

Two runs are same → PinPoints guaranteed to be reached

PinPlay*: Workload Capture and Deterministic Replay Framework



Guaranteed Repeatability

No binaries/inputs

No application setup

No license checking

Record Once Replay + Analyze Multiple Times Anywhere!

PinPlay: Included in SDE

SDE : Pin + Emulation + PinPlay

- SDE driver runs with a default tool (sde-mix.so)
  - Emulation, log/replay, loop generation (DCFG json files), checkers…..
- SDE Kit also allows people to write tools
  - Regular Pin tools + use of additional SDE API
  - Can use emulation, log/replay, DCFG….

SDE : Intel Software Development Emulator
(search “Intel SDE”)

Direct pinball generation with SDE

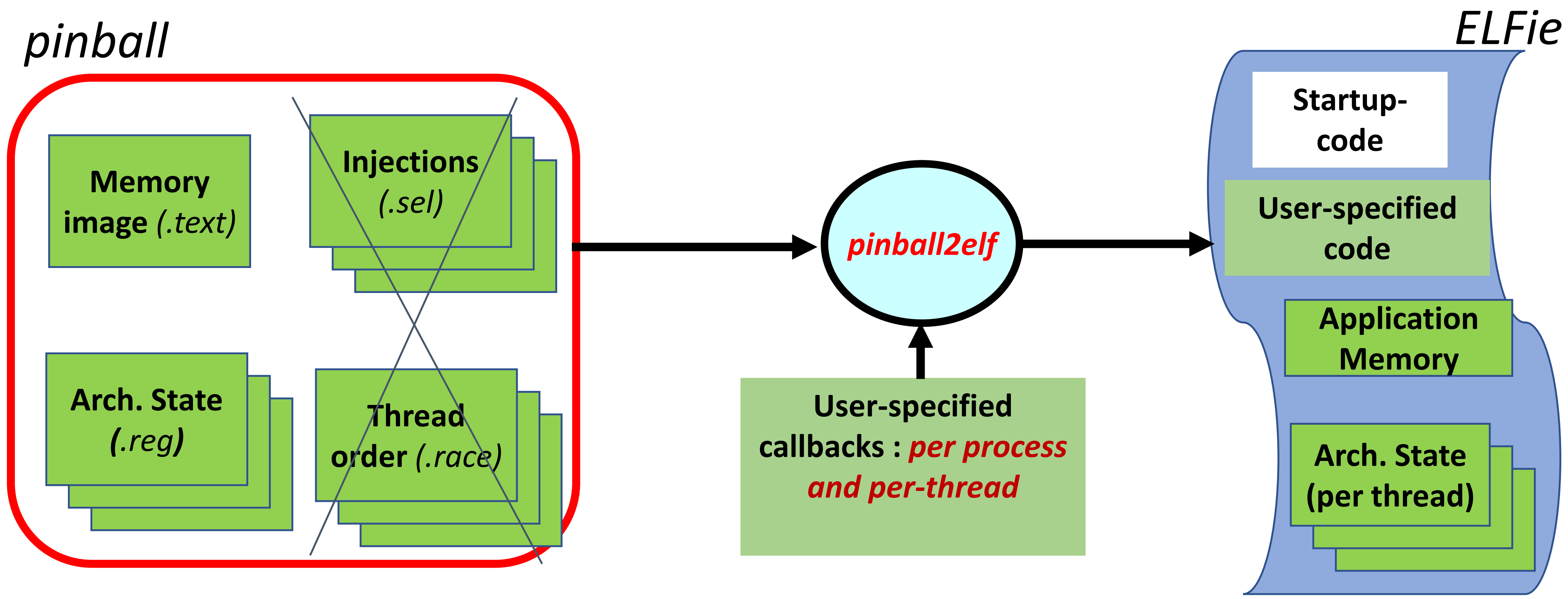
If you know your “Regions of Interest”, you may use SDE directly to generate pinballs for them

- SDE supports a “Controller” mechanism which has multiple ways of specifying ROI
- See this article or search for “Pintool regions”
- Example: (See run.sde.binary.log.sh in the PinPoints/Test directory)

```
% $SDE_ROOT/sde64 -controller_log -controller_olog region.binary -control start:address:dotproduct-st+0x12b0:count2692 -control stop:icount:300000 -log -log:basename pinball.region.binary/rpb -- dotproduct-st
```

Will create a pinball region starting at the PC=“image+offset”=“dotproduct-st+0x12b0” for a length of 300,000 instructions (as counted by SDE)

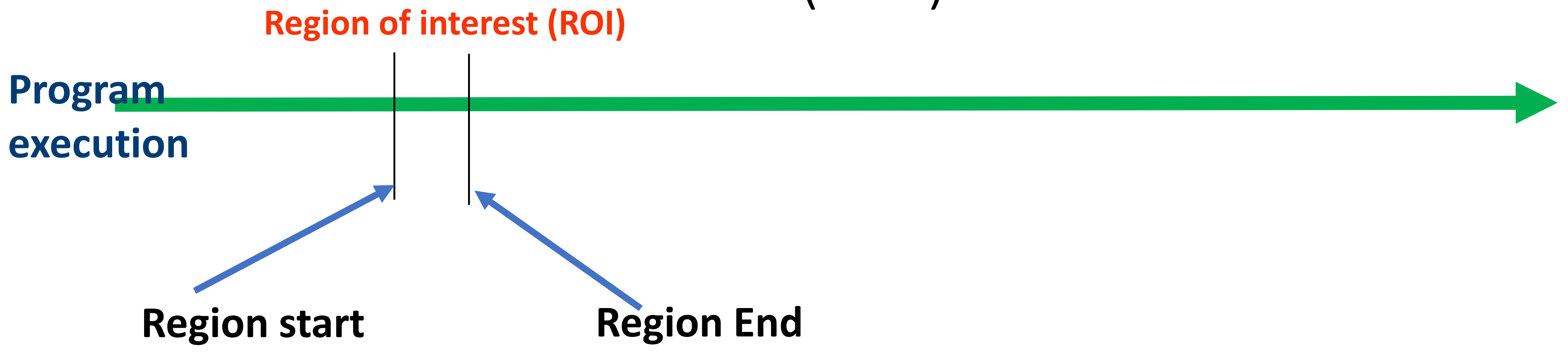
pinball2elf: Pinball converter to ELF



- No injections/thread-order enforcement
- System calls are re-executed

Open-sourced on *GitHub* : http://www.pinelfie.org
(or search "pinball2elf")

ROIperf: A Pin-probe tool for Targeted Performance monitoring of a region-of-interest (ROI)



Output: '*rdtsc*' and HW counter values at region boundaries

Hardware performance counter specification:

Comma separated pairs '***perftype:counter***'

ROIPERF_LIST=*"0:0,0:1…"*

Counter: 0:0 hw_cpu_cycles, 0:1: hw_instructions

based on /usr/include/linux/perf_event.h

perftype: 0 --> HW 1 --> SW
HW counter: 0 --> PERF_COUNT_HW_CPU_CYCLES
HW counter: 1 --> PERF_COUNT_HW_CPU_INSTRUCTION
SW counter: 0 --> PERF_COUNT_SW_CPU_CLOCK
... <see 'enum perf_hw_id' and 'enum perf_sw_id'>

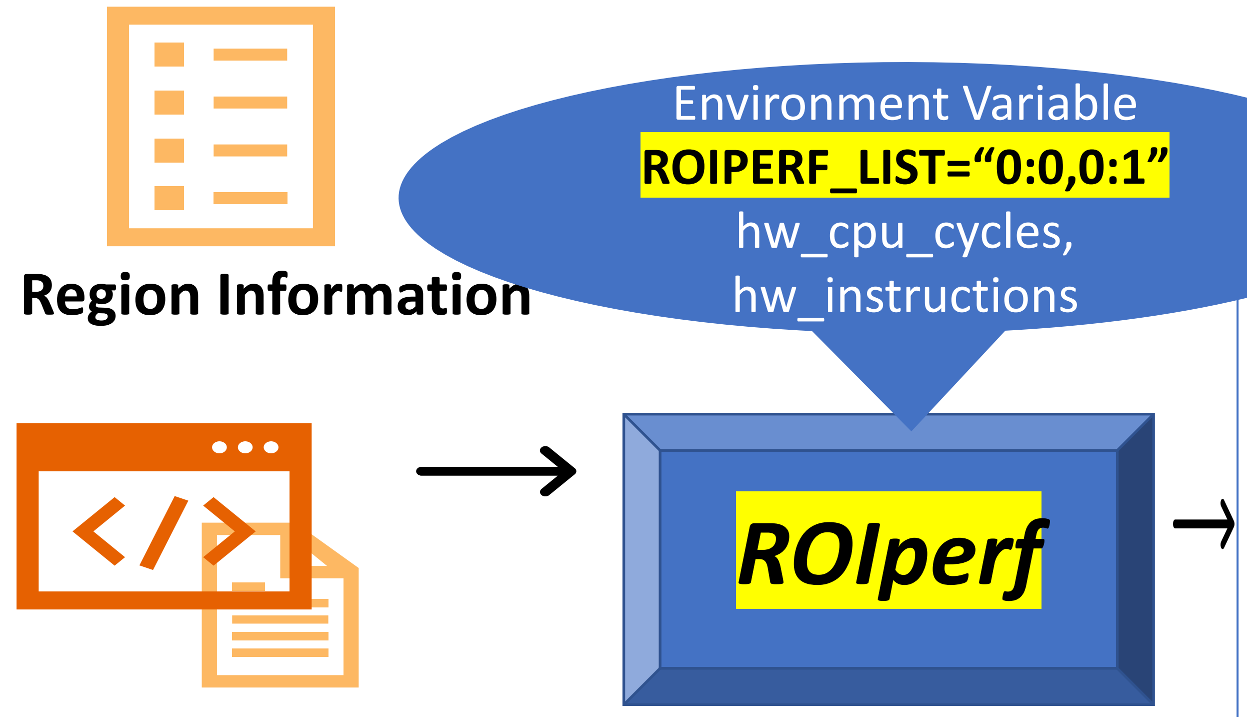
Usage example

```
start:main:1
istart:76400001418
istop:76600001434
```

Counting/monitoring start

Simulation start

Simulation End



Application and Inputs

*pin –t **ROIperf.so** –probecontrol:in probe.in –perfout perf.txt -- app-binary <arguments>*

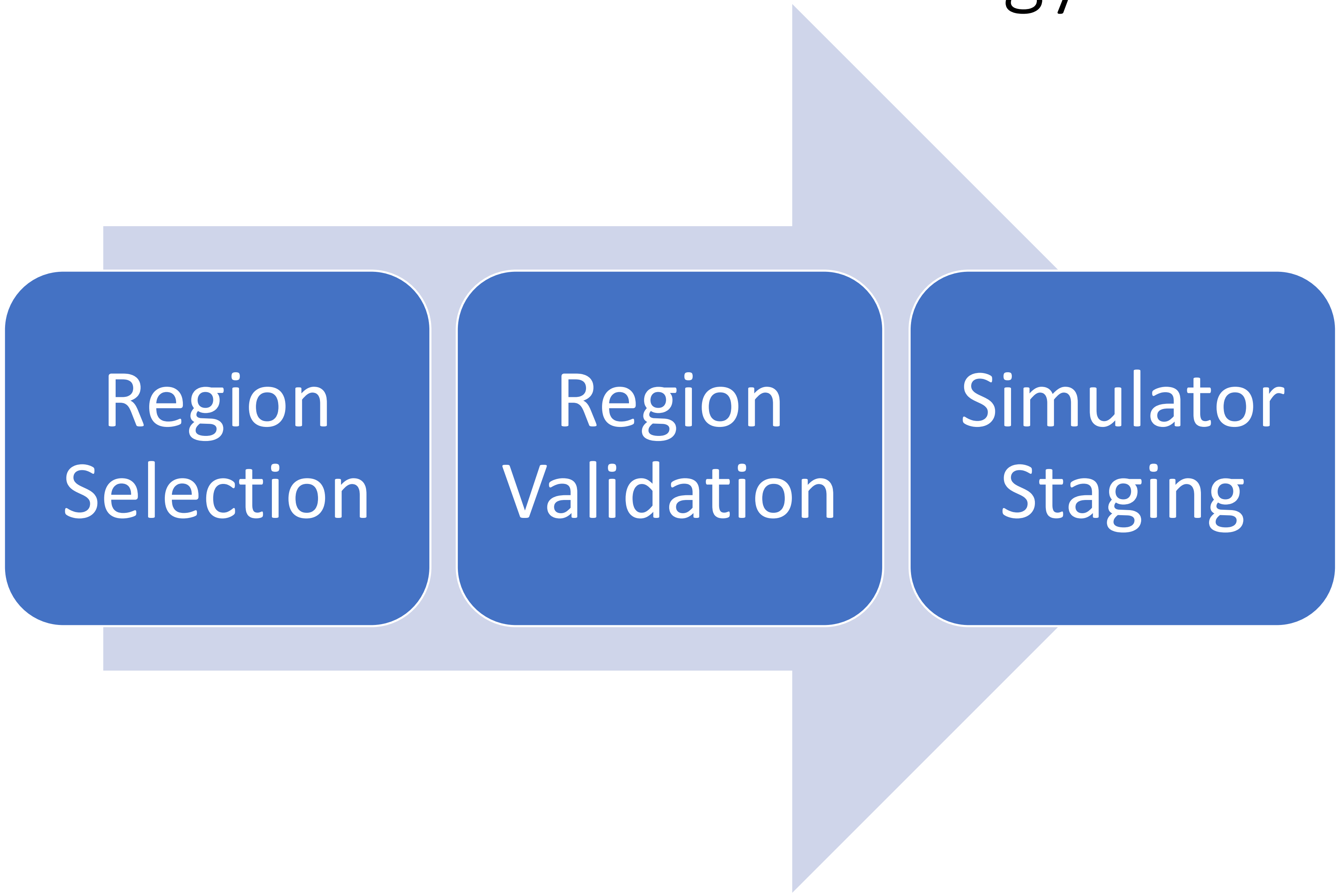
Counting/monitoring start

Simulation start

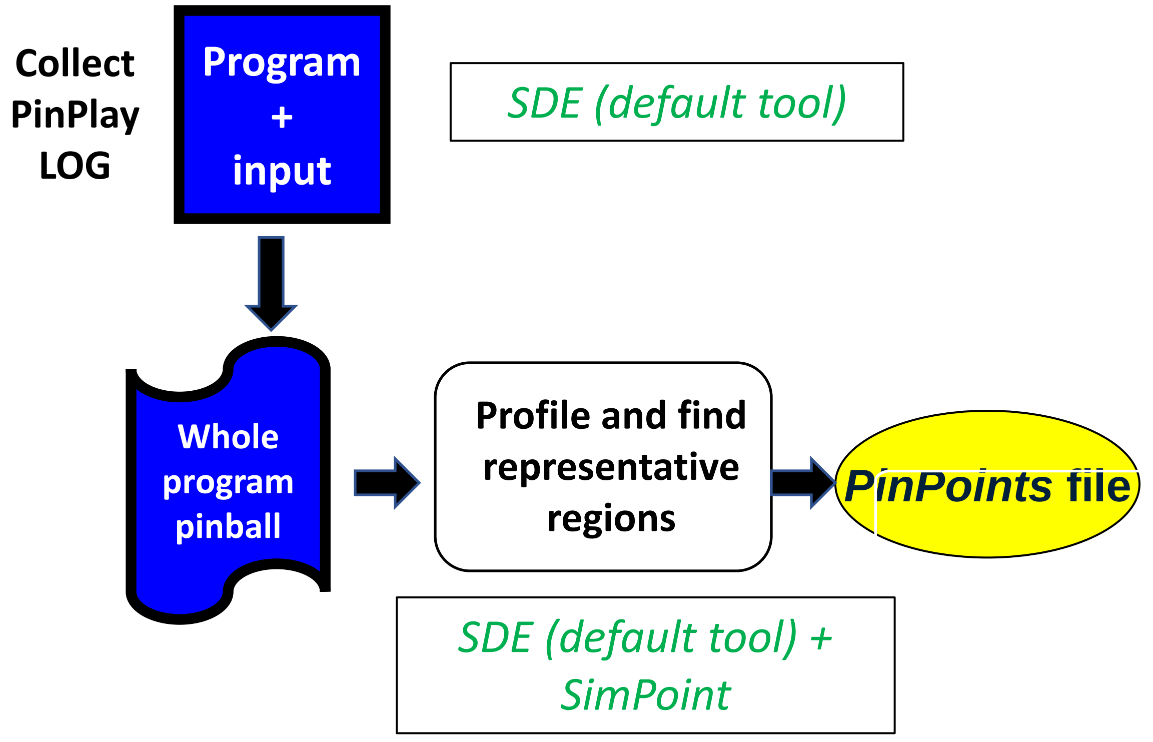
Simulation End

```
ROI start: TSC 1565618513658386
hw_cpu_cycles:2639 hw_instructions:572
--------------------------------------------
Warmup end: TSC 1565669952726412 → 'rdtsc'
  Warmup-end-icount 76400001418
hw_cpu_cycles:49830146369 hw_instructions:76399998590
--------------------------------------------
Simulation end: TSC 1565670056267816
  Sim-end-icount 76600001421
hw_cpu_cycles:49933072325 hw_instructions:76600001540
--------------------------------------------
Thread end: TSC 1565670056630234
```

PinPoints Methodology

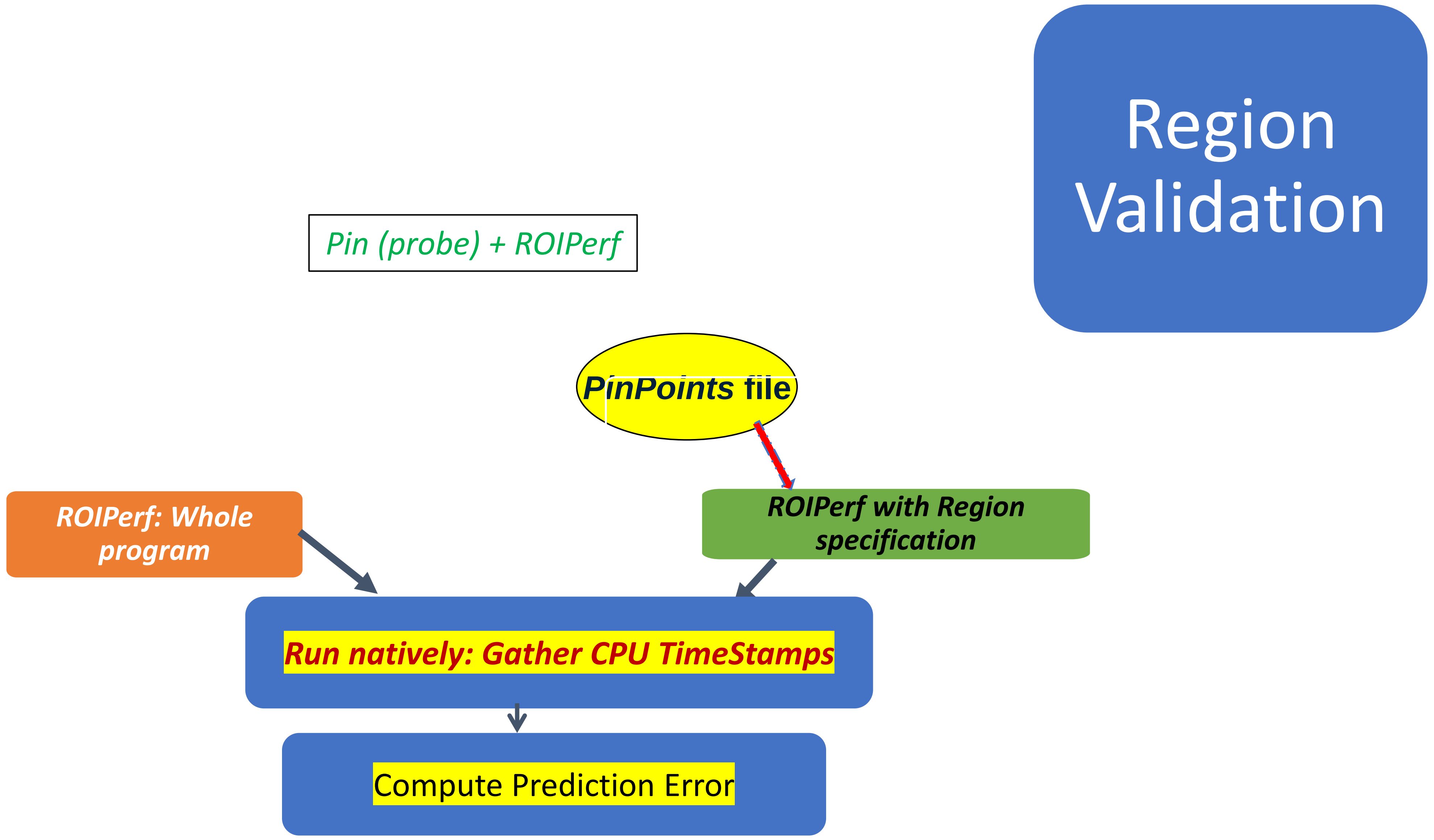


PinPoints Methodology: Simulation Region Selection

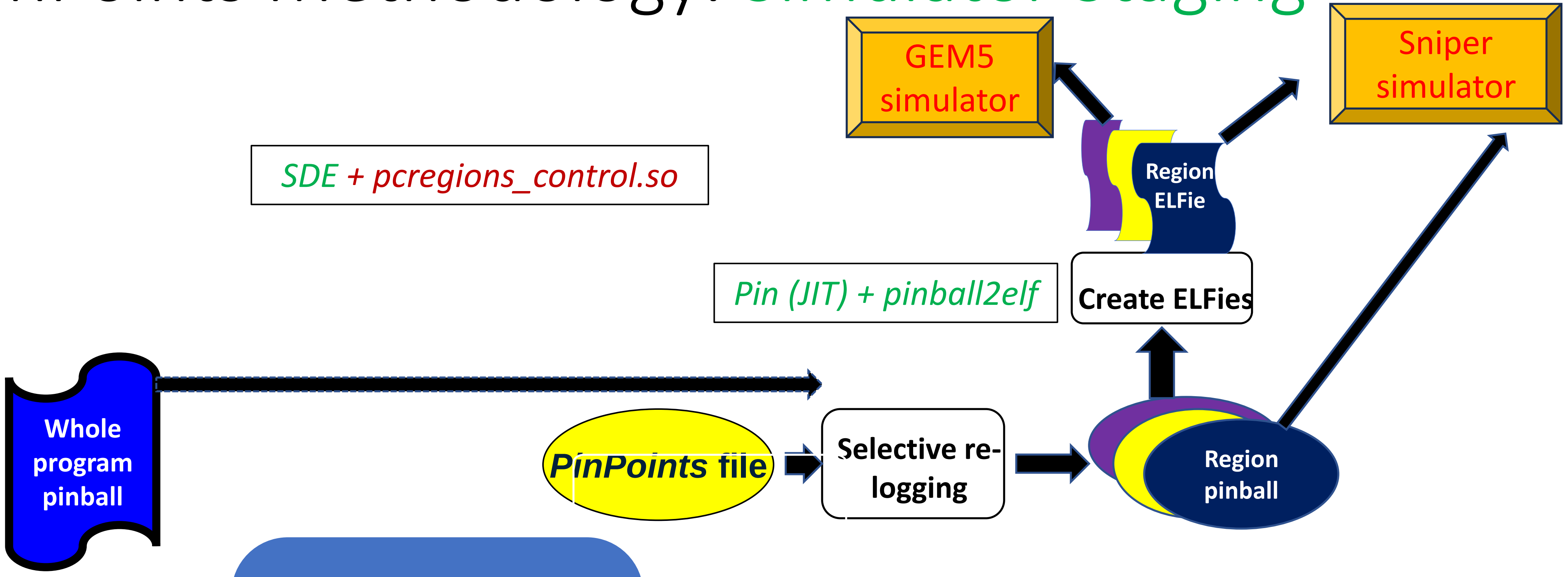


Region Selection

PinPoints Methodology: Validation

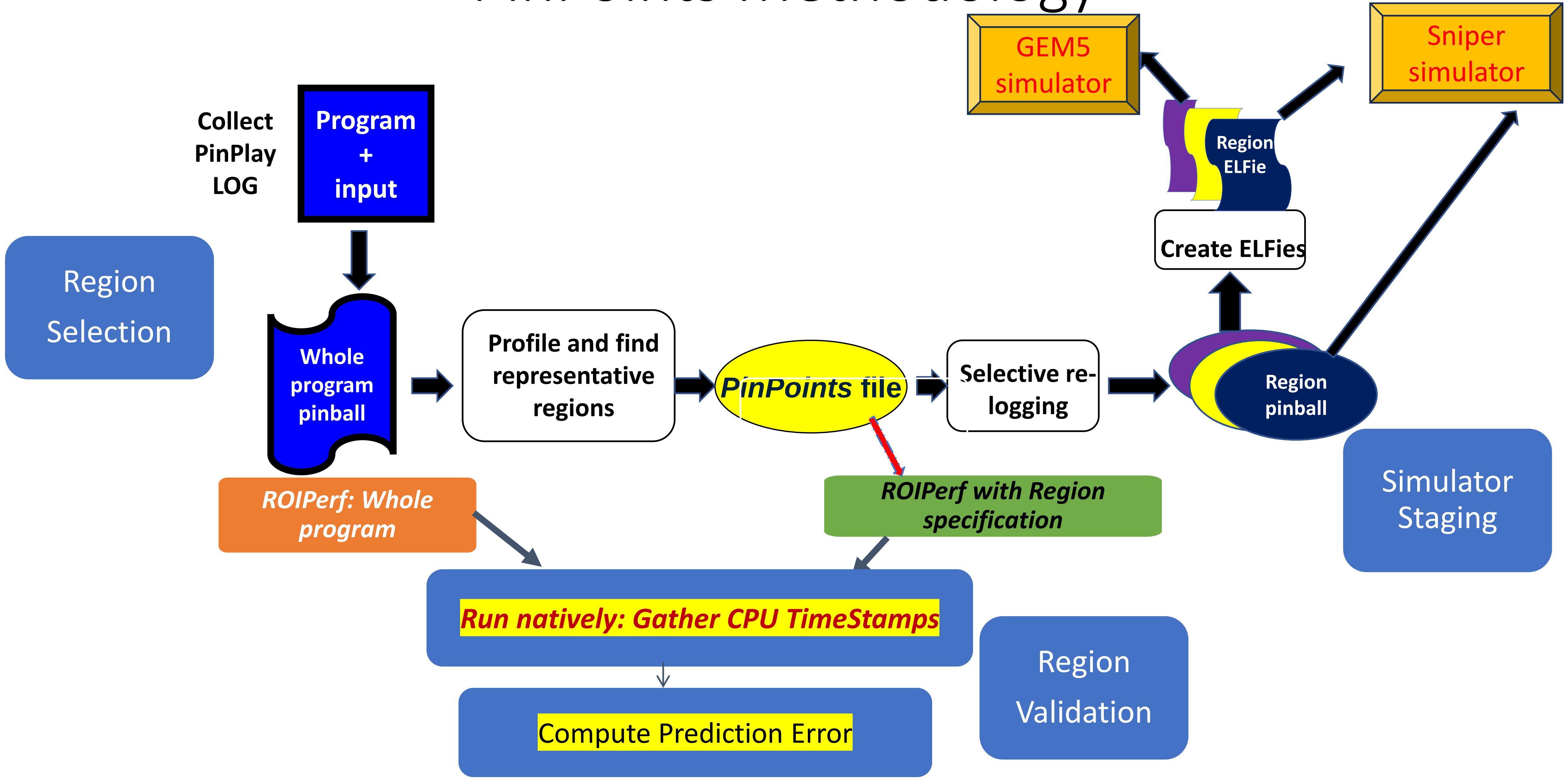


PinPoints Methodology: Simulator Staging



Simulator Staging

PinPoints Methodology



Demo:Part 1: Region Selection + region pinball generation

Pre-requisites

- Pin kit 3.31 : http://pintool.intel.com (or this link)
  - export PIN_ROOT=<path to the local copy of Pin kit 3.31>
- Intel SDE 9.44 : Use this link
  - export SDE_BUILD_KIT=<path to the local copy of SDE Kit>
- Pinplay-tools repo: https://github.com/intel/pinplay-tools
  - export PINPLAY_TOOLS=<path to the local clone of the pinplay-tools repo>
- Pinball2elf repo: https://github.com/intel/pinball2elf
  - export PINBALL2ELF=<path to the local clone of the pinball2elf repo>

< Assumption: using "bash" >

< put "." in PATH >

Build all required tools

- Make sure the pre-requisites are satisfied and the following environment variables are set
  - SDE_BUILD_KIT
  - PIN_ROOT
  - PINBALL2ELF

```
% cd pinplay-tools/PinPoints

% sde-pin-build-PinPoints.sh
```

Build SDE profiler: ‘pcregions_control.so’ *(already built by ‘sde-pin-build-PinPoints.sh’)*

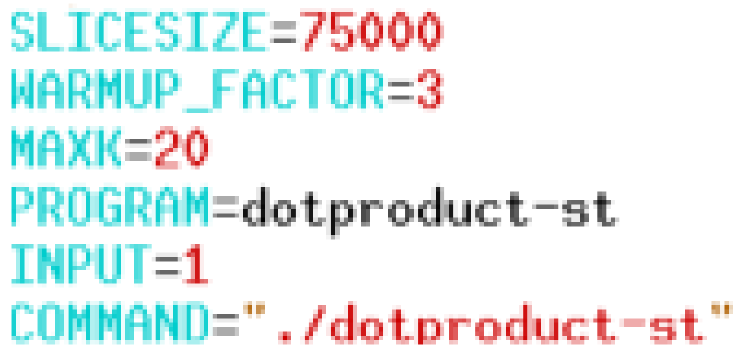
```
% cd $SDE_BUILD_KIT/pinkit/sde-example/example
% make TARGET=intel64 clean;  make TARGET=intel64
% cp obj-intel64/pcregions_control.so $SDE_BUILD_KIT/intel64
```

Test case: $PINPLAY_TOOLS/PinPoints/Test

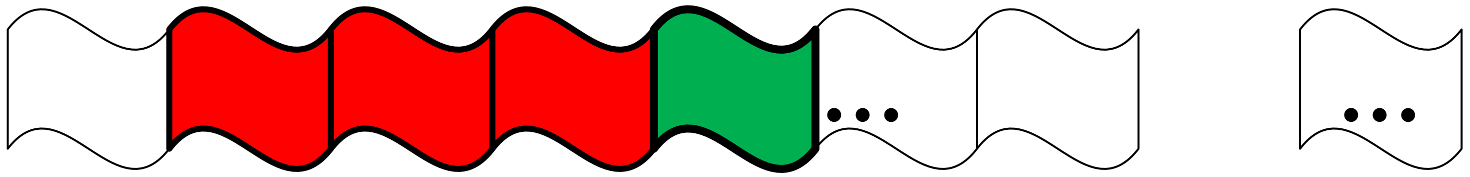
- cd $PINPLAY_TOOLS/PinPoints/Test
- < Edit *sde-run.pinpoints.single-threaded.sh* >

% make clean; make

% sde-run.pinpoints.single-threaded.sh

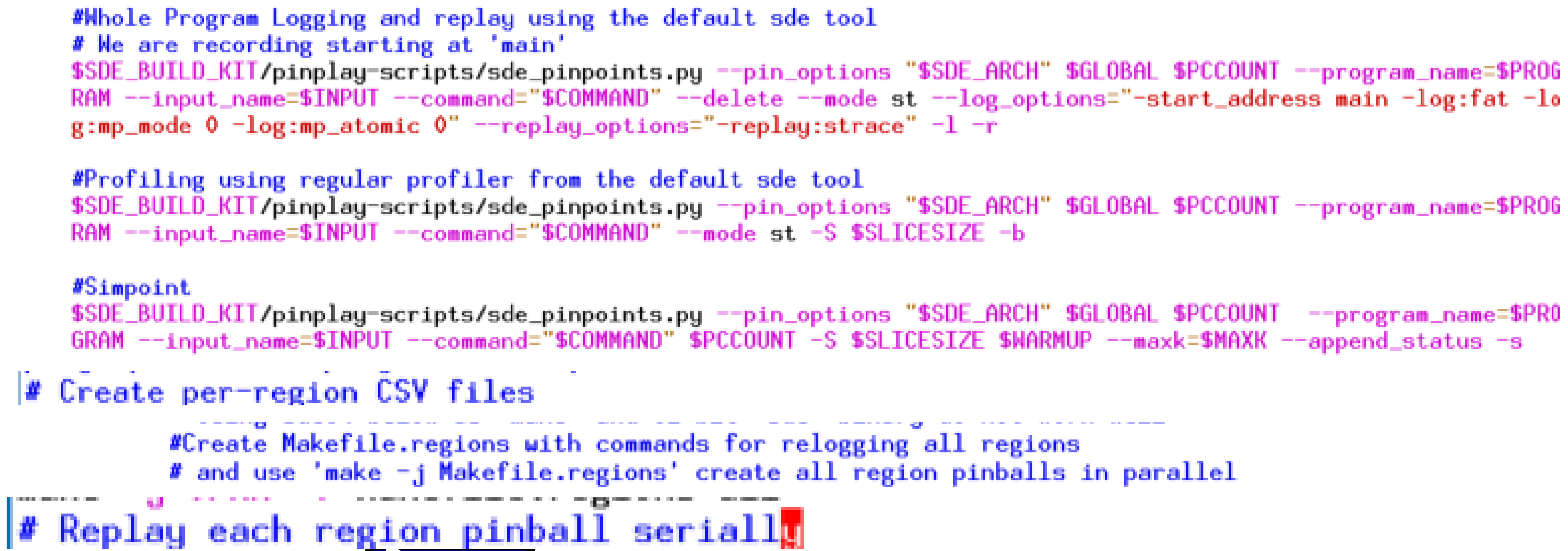


```
SLICESIZE=75000
WARMUP_FACTOR=3
MAXK=20
PROGRAM=dotproduct-st
INPUT=1
COMMAND="./dotproduct-st"
```



WARMUP_FACTOR=3
➔
Include 3 slices before simulation region for warmup

sde-run.pinpoints.single-thread.sh

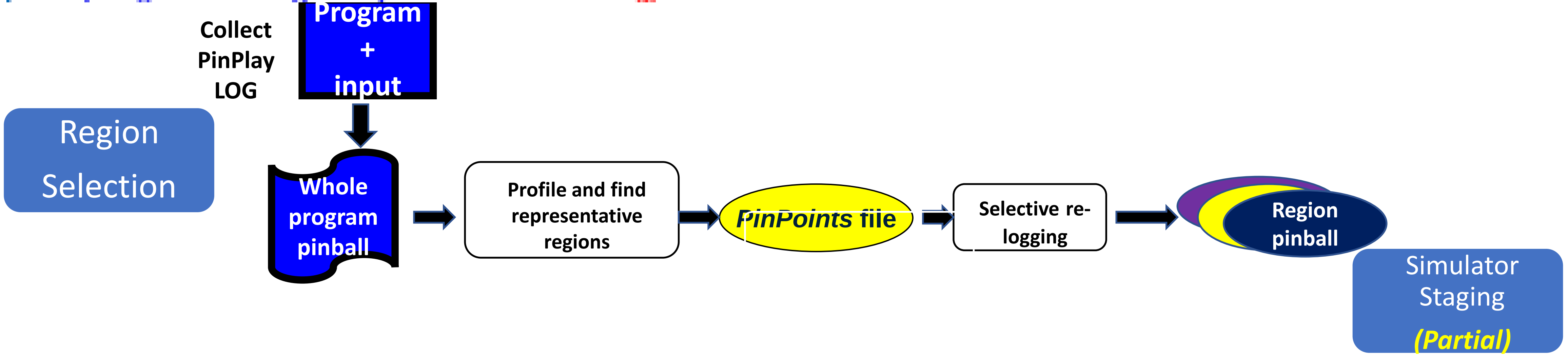


```
#Whole Program Logging and replay using the default sde tool
# We are recording starting at 'main'
$SDE_BUILD_KIT/pinplay-scripts/sde_pinpoints.py --pin_options "$SDE_ARCH" $GLOBAL $PCCOUNT --program_name=$PROG
RAM --input_name=$INPUT --command="$COMMAND" --delete --mode st --log_options="-start_address main -log:fat -lo
g:mp_mode 0 -log:mp_atomic 0" --replay_options="-replay:strace" -l -r

#Profiling using regular profiler from the default sde tool
$SDE_BUILD_KIT/pinplay-scripts/sde_pinpoints.py --pin_options "$SDE_ARCH" $GLOBAL $PCCOUNT --program_name=$PROG
RAM --input_name=$INPUT --command="$COMMAND" --mode st -S $SLICESIZE -b

#Simpoint
$SDE_BUILD_KIT/pinplay-scripts/sde_pinpoints.py --pin_options "$SDE_ARCH" $GLOBAL $PCCOUNT  --program_name=$PRO
GRAM --input_name=$INPUT --command="$COMMAND" $PCCOUNT -S $SLICESIZE $WARMUP --maxk=$MAXK --append_status -s
```

```
# Create per-region CSV files
```

```
        #Create Makefile.regions with commands for relogging all regions
        # and use 'make -j Makefile.regions' create all region pinballs in parallel
```

```
# Replay each region pinball serially
```



BYOP: Bring Your Own (whole-program) Pinball

- Generate pinball any other way, say at another site

- Say the pinball is input.pinball/log_0

```
% $SDE_BUILD_KIT/pinplay-scripts/sde_pinpoints.py ..
--whole_pgm_dir input.pinball --pin_options
"$SDE_ARCH" $GLOBAL $PCCOUNT --program_name=$PROGRAM
--input_name=$INPUT --command="$COMMAND" --mode st -S
$SLICESIZE -b
```

Demo:Part 2: Region Selection Validation

Build sde-global-event-icounter tool

(used for PCCount region processing)

(already built by 'sde-pin-build-PinPoints.sh')

```
% cd $PINPLAY_TOOLS/GlobalLoopPoint/EventCounter/
% make clean TARGET=intel64; make build TARGET=intel64
```

Build ROIPerf tool

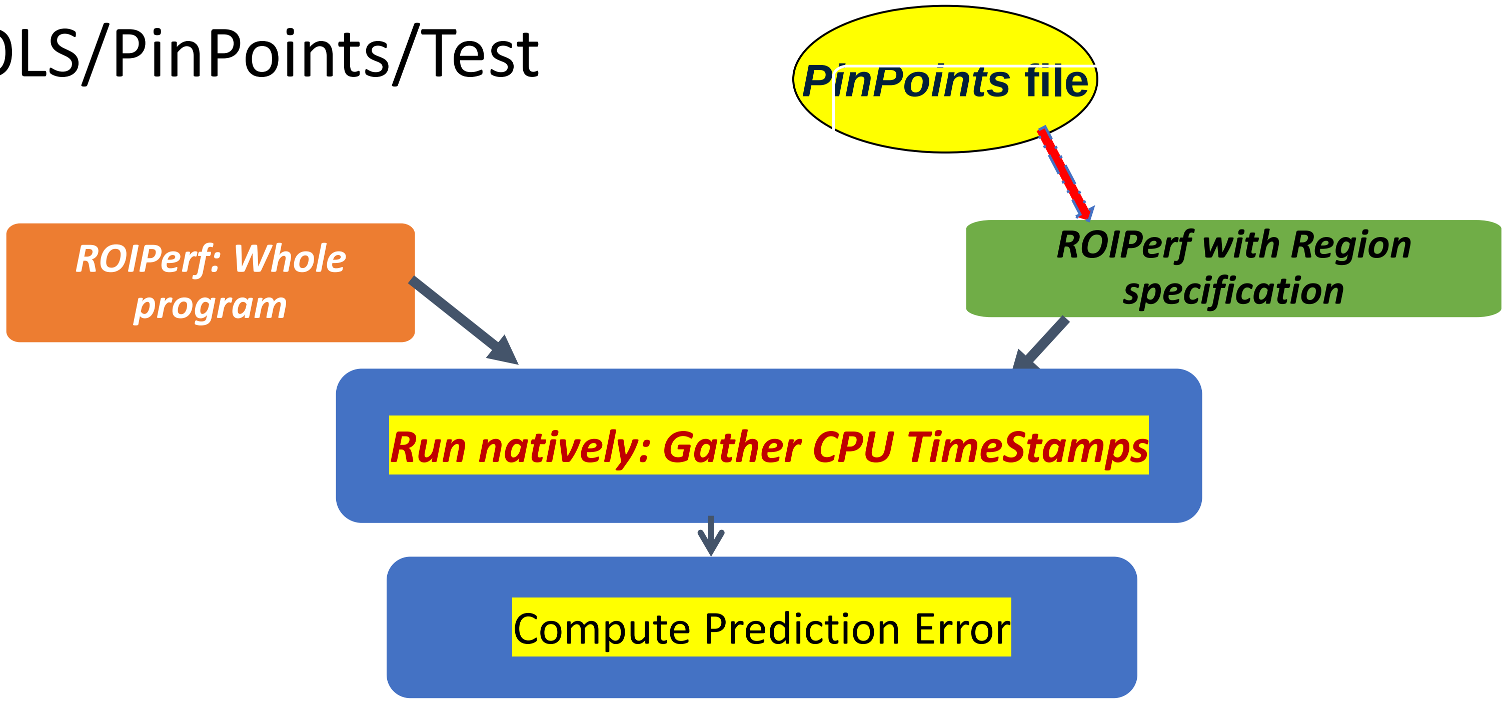
(used for time-stamp[rdtsc] generation)

(already built by ‘sde-pin-build-PinPoints.sh’)

```
% cd $PINBALL2ELF/pintools/ROIProbe
% make clean; make
% cp obj-intel64/pcregions_control.so $SDE_BUILD_KIT/intel64
```

Run ROIPerf-based validation

```
% cd $PINPLAY_TOOLS/PinPoints/Test
% run.ROIPerf.sh
```


PinPoints file
ROIPerf: Whole program
ROIPerf with Region specification
Run natively: Gather CPU TimeStamps
Compute Prediction Error

```
wp_rdtsc,     region_rdtsc,        err%
702607571.25,     504807628.48800004,    28.15%
```

- Prediction error can be tuned by changing SLICESIZE and MAXK
- Error high for short-running programs due to measurement overhead

Validation of PinPoints
SPEC2017 Rate/Train input: Simulation vs ROIperf

		% CPI Prediction Error	
	Intel-Sniper: Skylake Server	ROIperf: BroadWell Server	ROIperf: Skylake Server
perlbench_r.train.1	0.3%	0.5%	0.7%
perlbench_r.train.2	1.3%	0.0%	2.7%
perlbench_r.train.3	0.8%	0.0%	-0.1%
perlbench_r.train.4	0.2%	3.6%	3.0%
perlbench_r.train.5	8.9%	-6.6%	-0.7%
gcc_r.train.1	10.4%	-3.7%	-6.1%
gcc_r.train.2	1.0%	3.7%	0.4%
gcc_r.train.3	0.8%	3.9%	4.3%
mcf_r.train.1	8.8%	2.8%	0.2%
omnetpp_r.train.1	1.2%	2.8%	1.4%
x264_r.train.1	3.3%	0.0%	0.0%
leela_r.train.1	2.1%	-0.2%	-0.4%
exchange2_r.train.1	5.4%	0.0%	0.9%
xz_r.train.1	4.0%	-8.2%	-7.3%
xz_r.train.2	3.5%	-3.7%	-3.3%

Sniper Simulation time: up to 5 weeks

ROIperf validation time (3 trials each): few hours

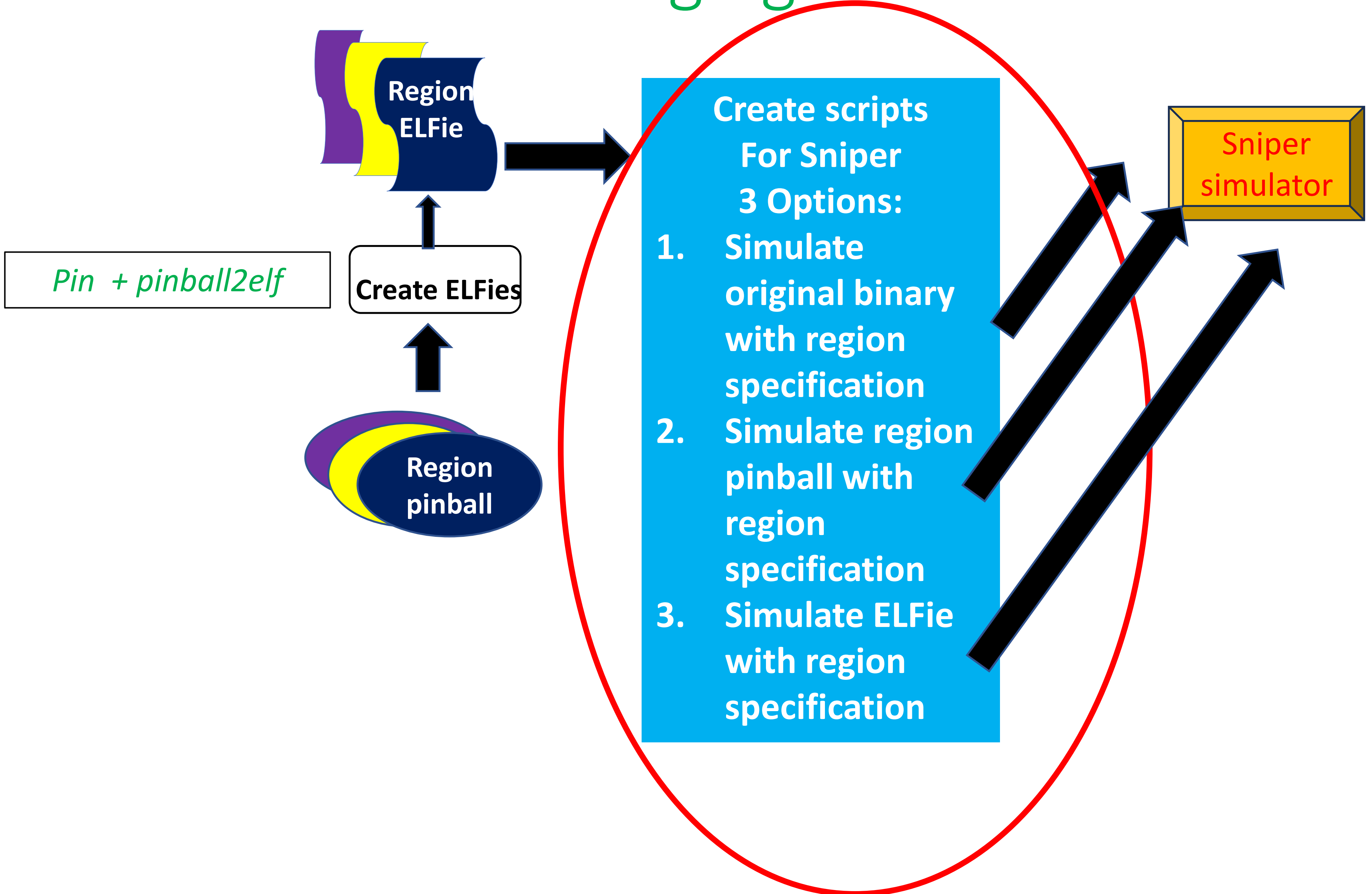
Demo:Part 3: Creating ELFies and script templates for Sniper and GEM-5

ELFie generation

```
% run.pinball2elf.sh
% run.elfies.sh
```

<GEM5/Sniper template script generation to be added soon>

PinPoints Methodology: Sniper Simulator Staging



Create scripts
For Sniper
3 Options:

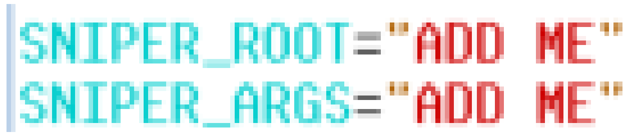
1. Simulate original binary with region specification
2. Simulate region pinball with region specification
3. Simulate ELFie with region specification

Staging for Sniper

< Edit create.sniper_region_simulation_scripts.sh>

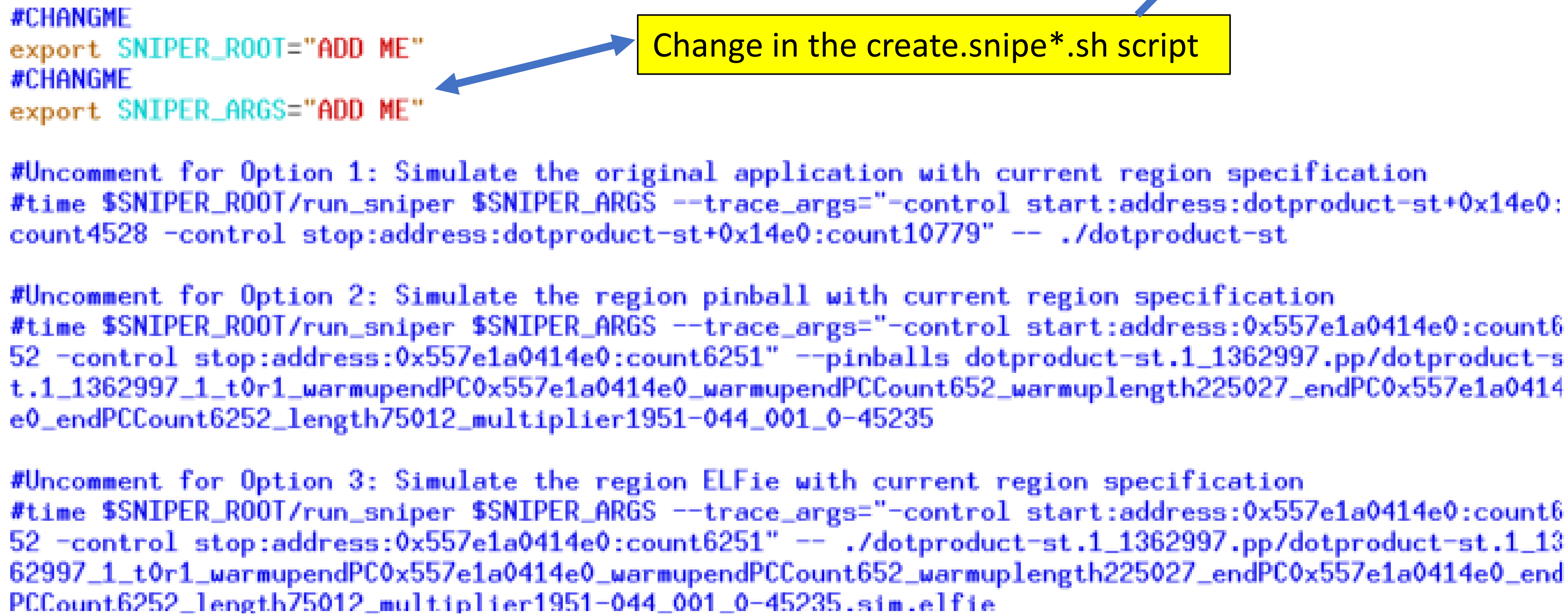
- %create.sniper_region_simulation_scripts.sh

# creates one bash script per region



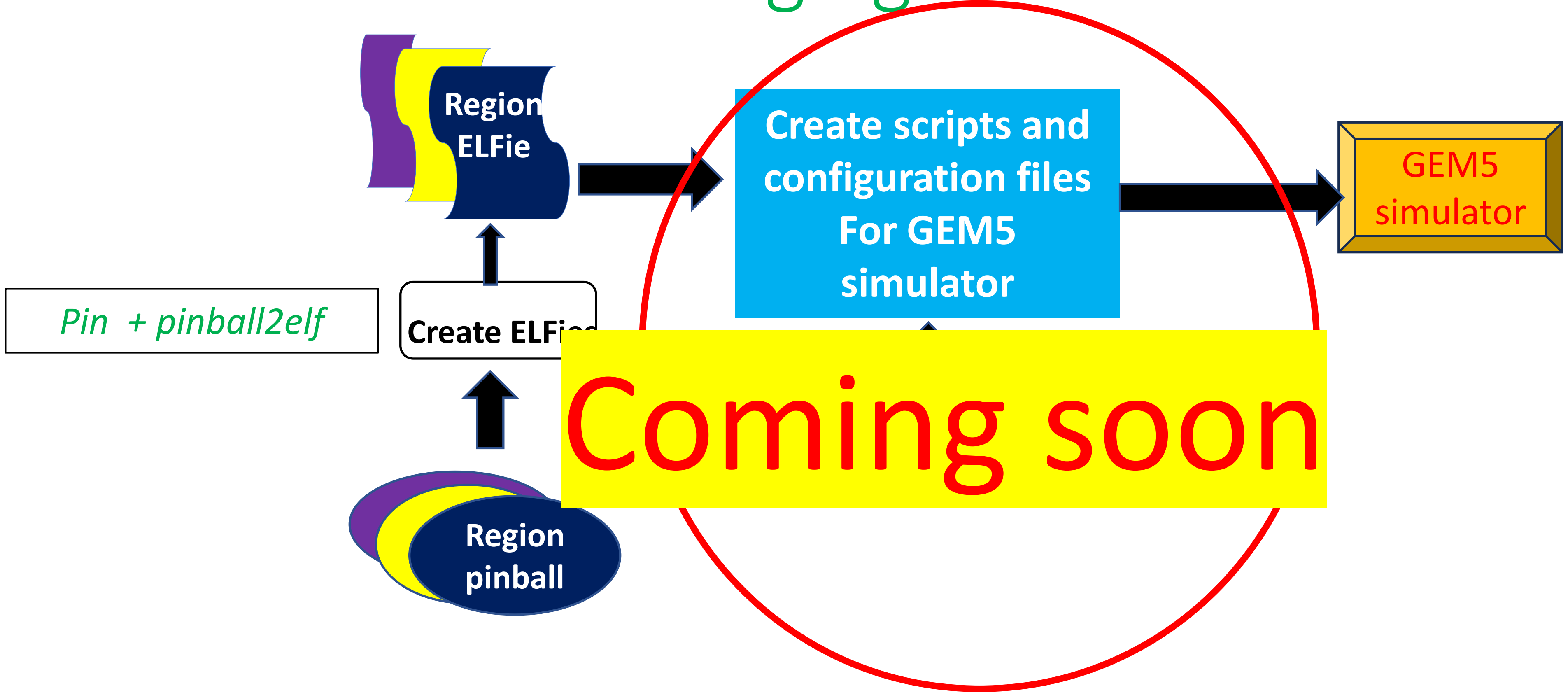
```
SNIPER_ROOT="ADD ME"
SNIPER_ARGS="ADD ME"
```

Change in the create.snipe*.sh script



```
#CHANGME
export SNIPER_ROOT="ADD ME"
#CHANGME
export SNIPER_ARGS="ADD ME"

#Uncomment for Option 1: Simulate the original application with current region specification
#time $SNIPER_ROOT/run_sniper $SNIPER_ARGS --trace_args="-control start:address:dotproduct-st+0x14e0:
count4528 -control stop:address:dotproduct-st+0x14e0:count10779" -- ./dotproduct-st

#Uncomment for Option 2: Simulate the region pinball with current region specification
#time $SNIPER_ROOT/run_sniper $SNIPER_ARGS --trace_args="-control start:address:0x557e1a0414e0:count6
52 -control stop:address:0x557e1a0414e0:count6251" --pinballs dotproduct-st.1_1362997.pp/dotproduct-s
t.1_1362997_1_t0r1_warmupendPC0x557e1a0414e0_warmupendPCCount652_warmuplength225027_endPC0x557e1a0414
e0_endPCCount6252_length75012_multiplier1951-044_001_0-45235

#Uncomment for Option 3: Simulate the region ELFie with current region specification
#time $SNIPER_ROOT/run_sniper $SNIPER_ARGS --trace_args="-control start:address:0x557e1a0414e0:count6
52 -control stop:address:0x557e1a0414e0:count6251" -- ./dotproduct-st.1_1362997.pp/dotproduct-st.1_13
62997_1_t0r1_warmupendPC0x557e1a0414e0_warmupendPCCount652_warmuplength225027_endPC0x557e1a0414e0_end
PCCount6252_length75012_multiplier1951-044_001_0-45235.sim.elfie
```

PinPoints Methodology: GEM5 Simulator Staging



Resources

- This tutorial
- Past PinPoints tutorials (commands/kits outdated)
    - ISCA2014-PinPoints-Tutorial
    - HPCA2013-PinPoints-Tutorial
- Past PinPlay tutorials (commands/kits outdated)
    - PLDI2015-PinPlay-Tutorial
    - PLDI2016-PinPlay-Tutorial
- LoopPoint Methodology (multi-threaded programs): See https://looppoint.github.io/
- Public Release and Validation of SPEC CPU2017 PinPoints (Northwestern University)
- SPEC2017 Pinballs from UT Austin
- SPEC highlights PinPoints research