UNIVERSITEIT GENT

intel

Pinballs:

Portable and Shareable User-level Checkpoints for Reproducible Analysis and Simulation

Harish Patil (*Intel Corporation*)
Trevor E. Carlson (*Ghent University*)

February 15th, 2014 REPRODUCE2014/HPCA Orlando, Florida

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.

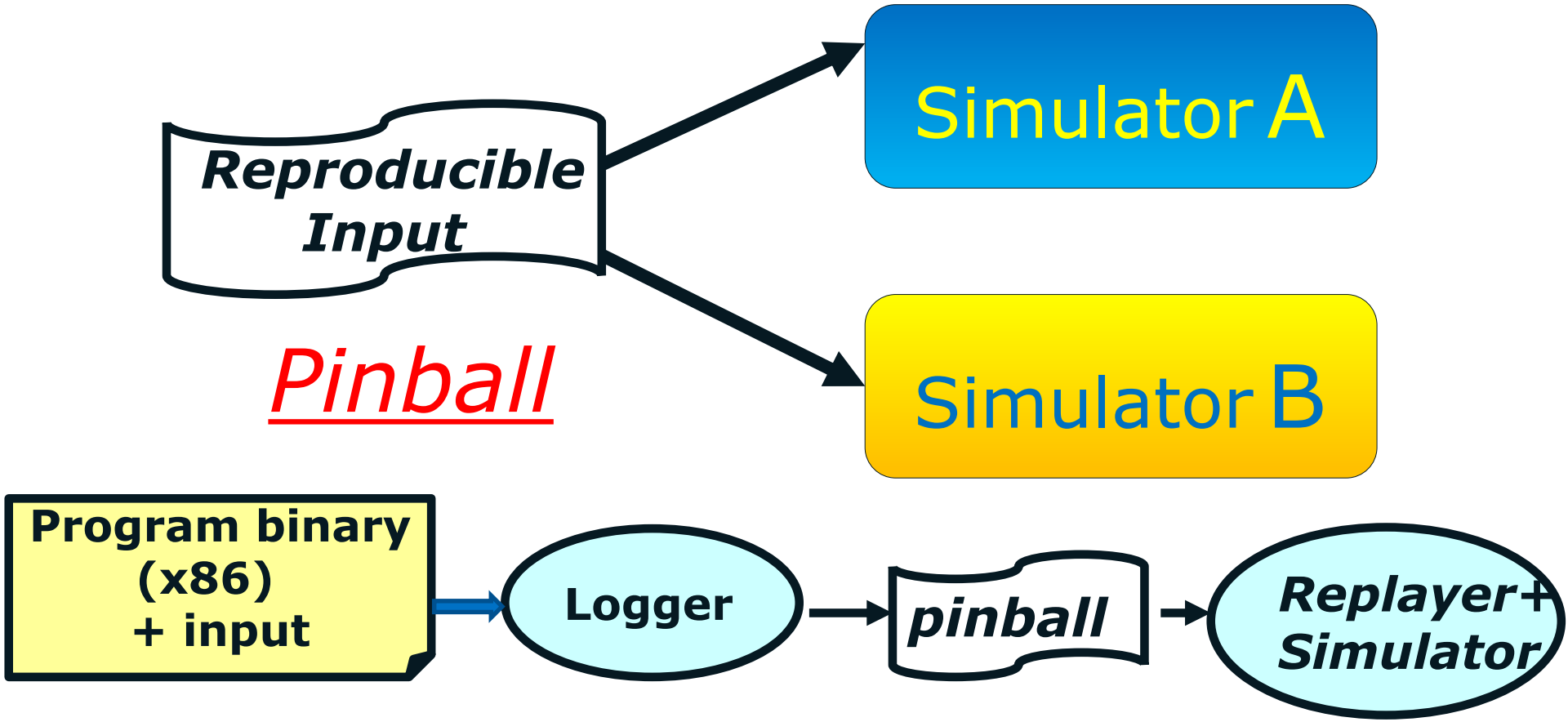
Optimization Notice

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

Objective : Reproducible Input for Simulators

Reproducible Input ➔ Apples to apples comparison



Check-pointing is not enough…Need to control non-determinism (Pinball does)

Single-threaded programs:

1. Changes to : Initial **stack/heap** addresss ; **shared library load location** (randomized address space)
2. Processor-specific instruction behavior changes: **CPUID, RDTSC**
3. **Signals** : Arbitrary memory/register/control-flow changes
4. **System call** behavior changes (depending on work-directory, environment variables: *gettimeofday() uname()*)

Multi-threaded programs: Shared-memory locations access order can change from run-to-run

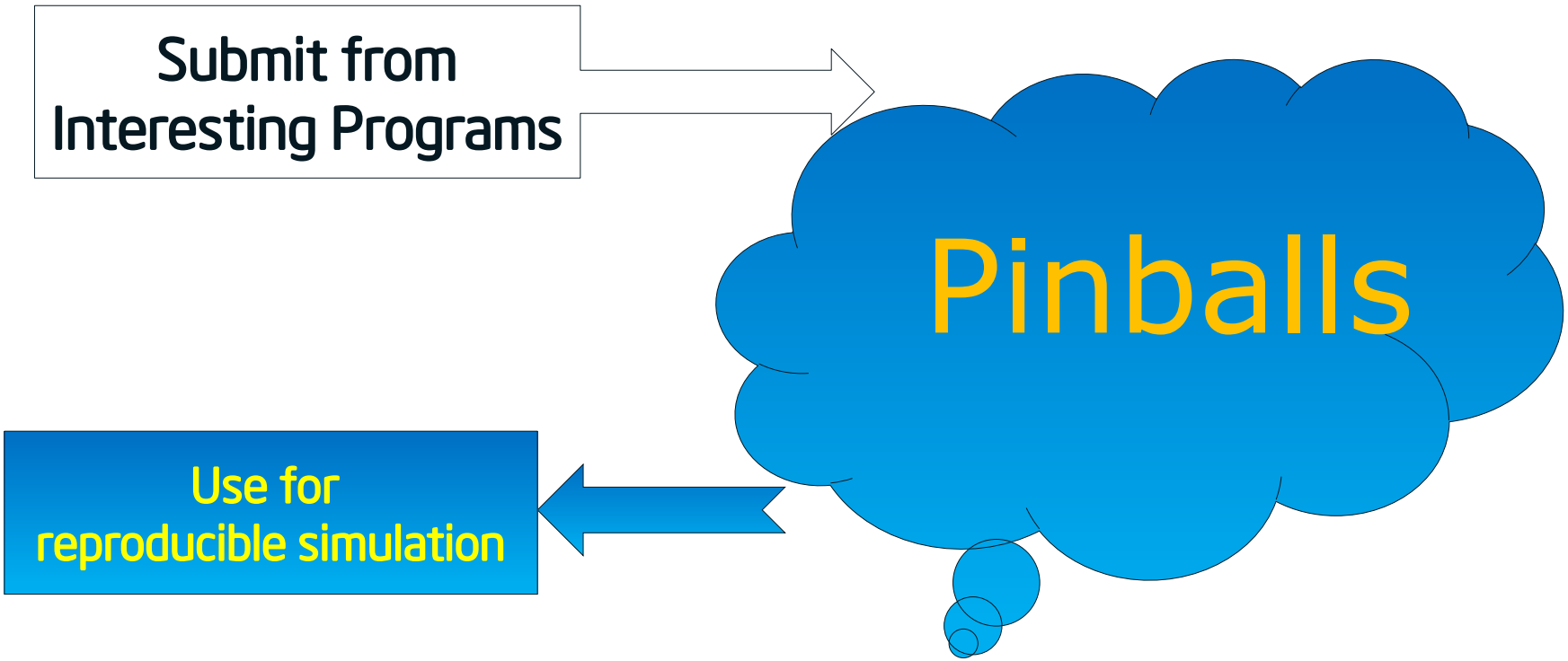
- **Accesses to locks/ mutexes**
  Different threads may "win" the race on different runs
- **OS scheduling**, load, machine configuration

Pinballs : Properties

- **Enable deterministic simulation : ST and MT**
  - See Trevor's talk on MT simulation
- **Independent of operating system (version/type)**
  - System call side-effects injected
- **Small in size**
  - Whole-program or regions

SPEC2006 (Average for 55 reference runs)	Instruction count	Pinball size	Logging slowdown (X native run-time)	Replay slowdown (X native run-time)
Whole-program	924 billion	39 MB	180X	20X
Region	130 million	12 MB		

The Vision: A Global Repository of Pinballs

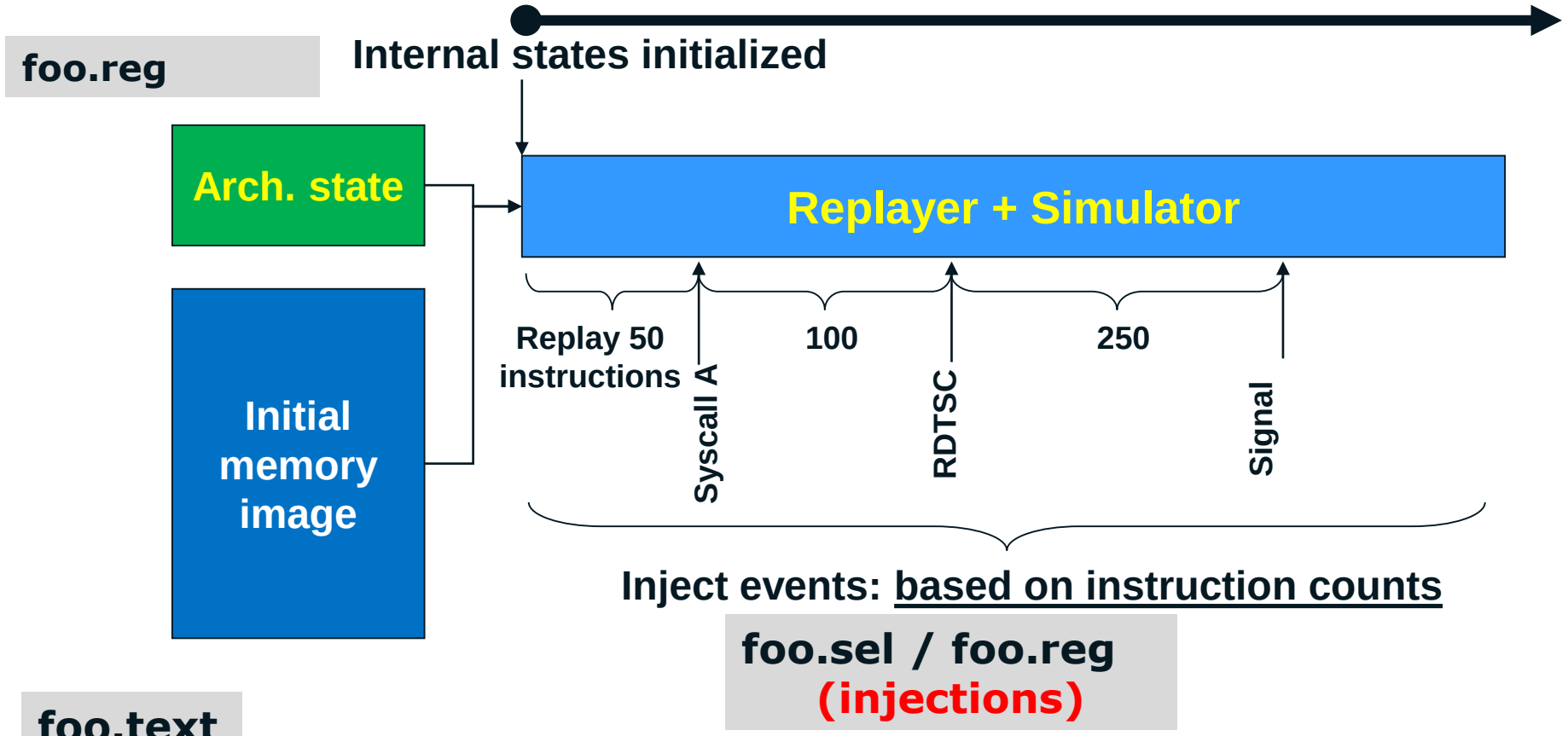


Started already (SPEC2006) :
http://snipersim.org/Pinballs

Outline

- Pinballs
  - **What are they?**
  - How are they generated?
  - How to use them for simulation?

Pinball (ST) = Initial memory/register + injections



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

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

foo.reg (per-thread)

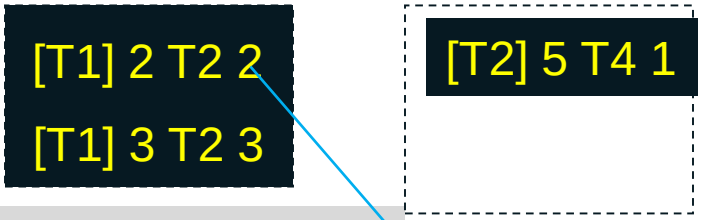
foo.text

Application Memory (common)

foo.reg (per-thread)

foo.sel (per-thread)

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



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

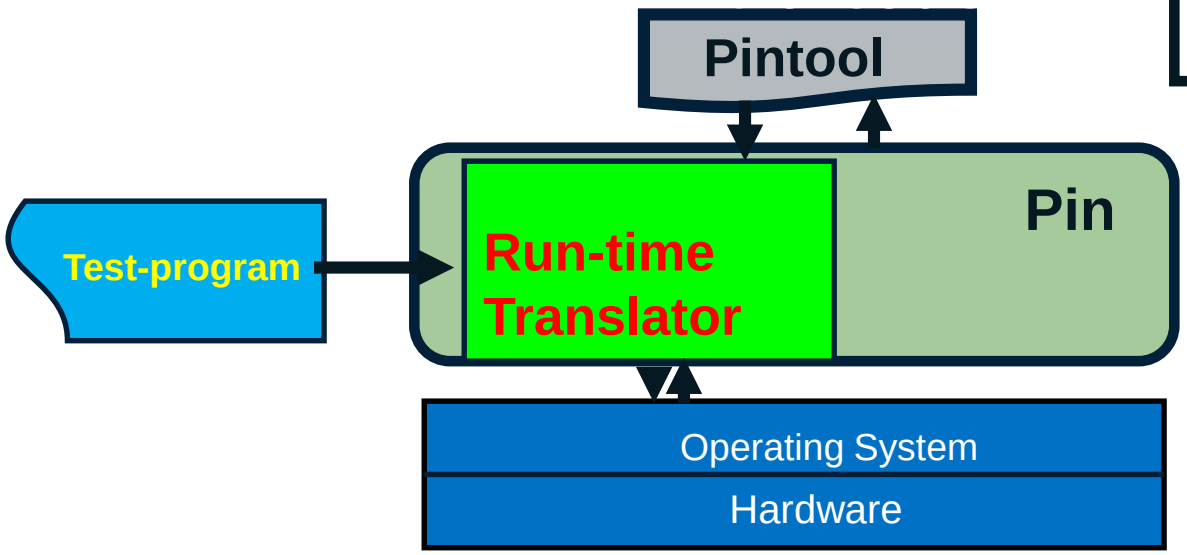
Outline

- Pinballs
  - What are they?
  - **How are they generated?**
  - How to use them for simulation?

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

PinPlay: Workload Capture and Deterministic Replay Framework

Program/ Libraries

Input

License

Guaranteed Repeatability

No binaries/inputs

No application setup

No license checking

Record Once Replay + Analyze Multiple Times Anywhere!

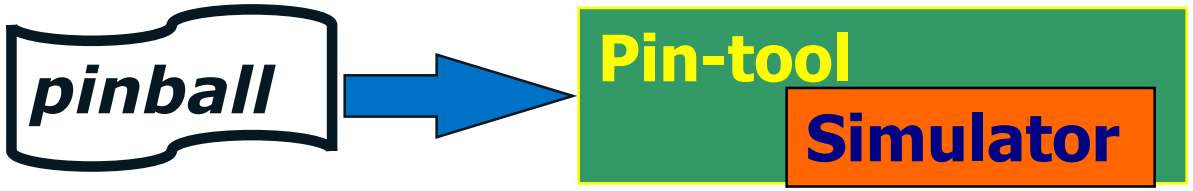
Program Record/Replay Toolkit (Linux only) from Intel
http://www.pinplay.org

Outline

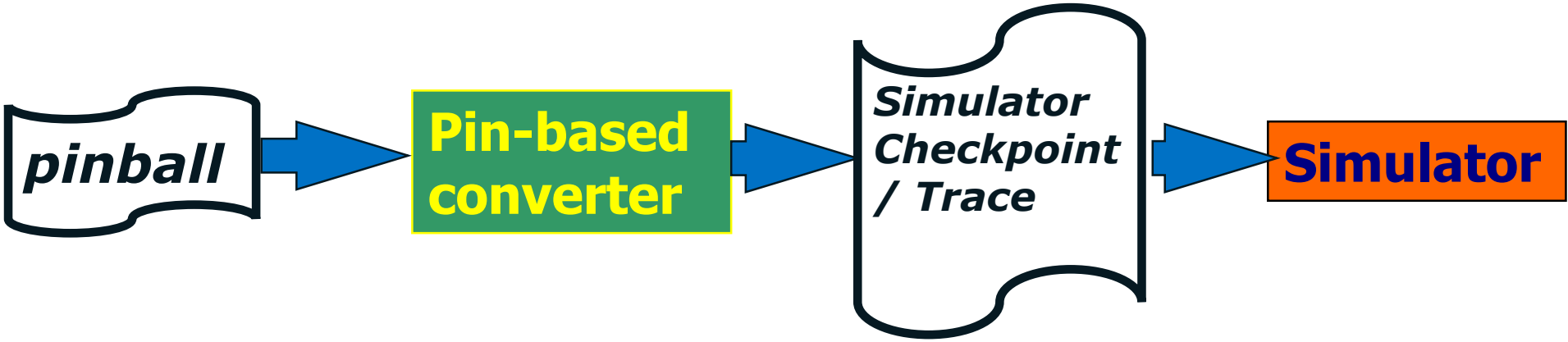
- Pinballs
  - What are they?
  - How are they generated?
  - **How to use them for simulation?**

Pinball-based Simulation: Two Usage Models

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



2. Other, non-Pin, simulators: e.g. Intel-internal
   Looking for collaboration : QEMU-based/ other simulators



Simulation Challenge + Solution

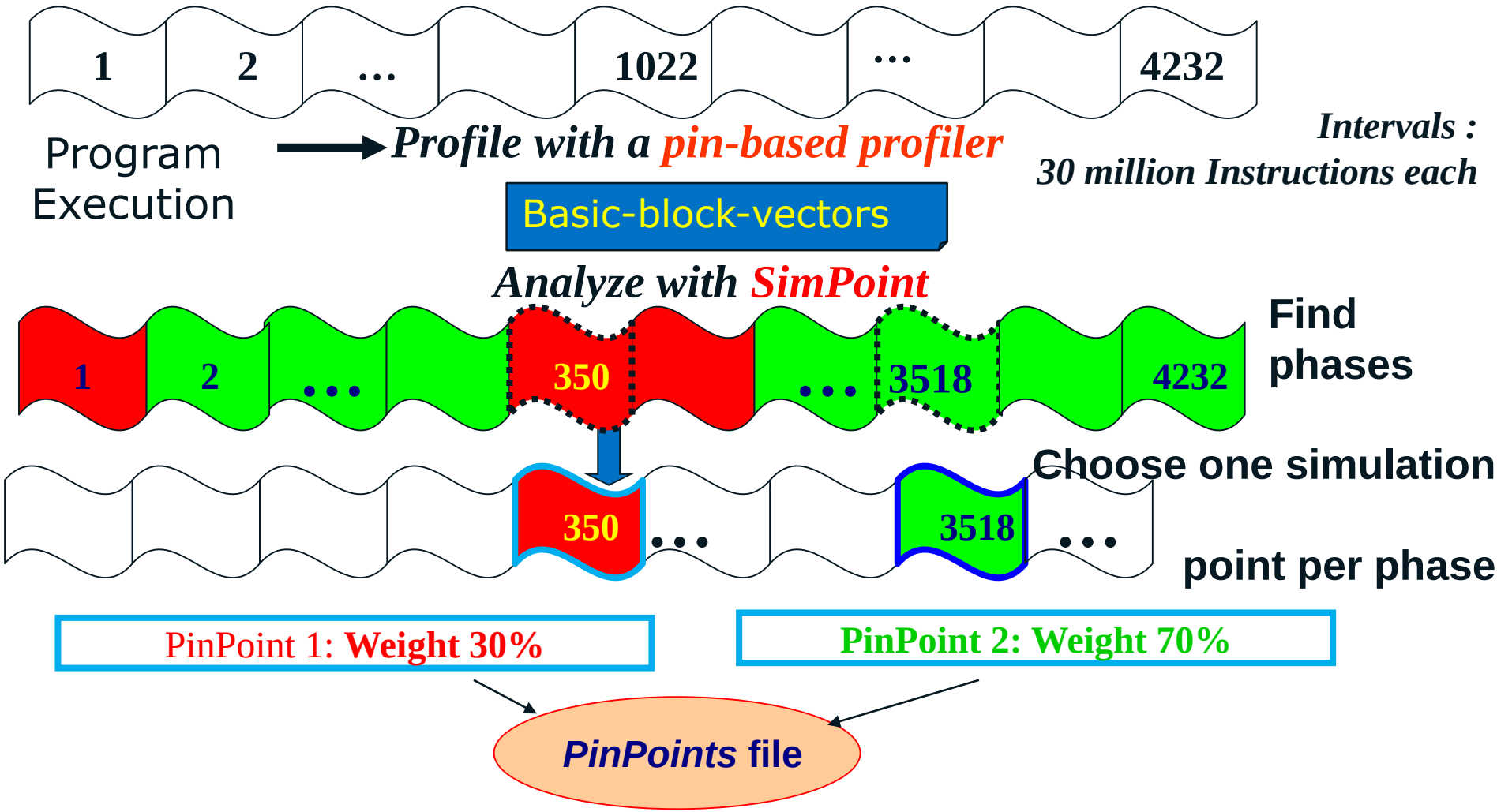
Detailed, cycle-accurate, simulation is slow

Solution: Find/simulate only representative regions (*PinPoints*)

- PinPoints cover << 1% of whole-program execution

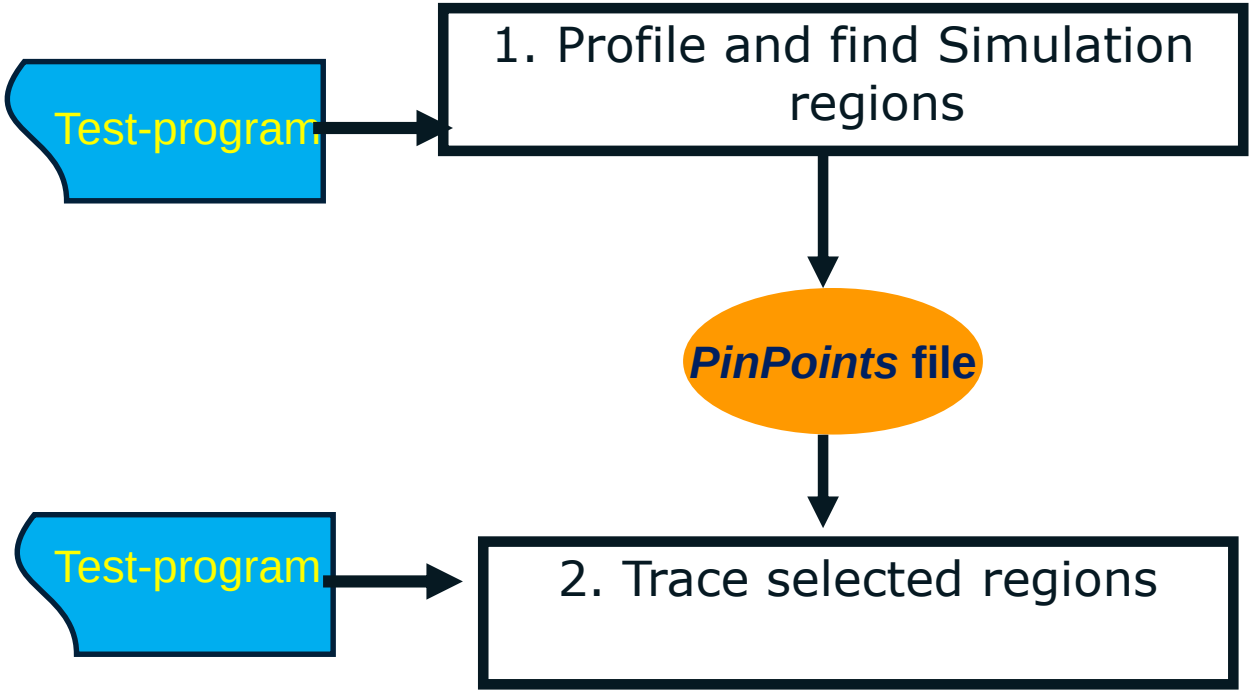
➔ **vastly reduced simulation time**

PinPoints = Pin + SimPoint (UC San Diego)



Two Phases => Two PinPoints

PinPoints : The Repeatability Challenge

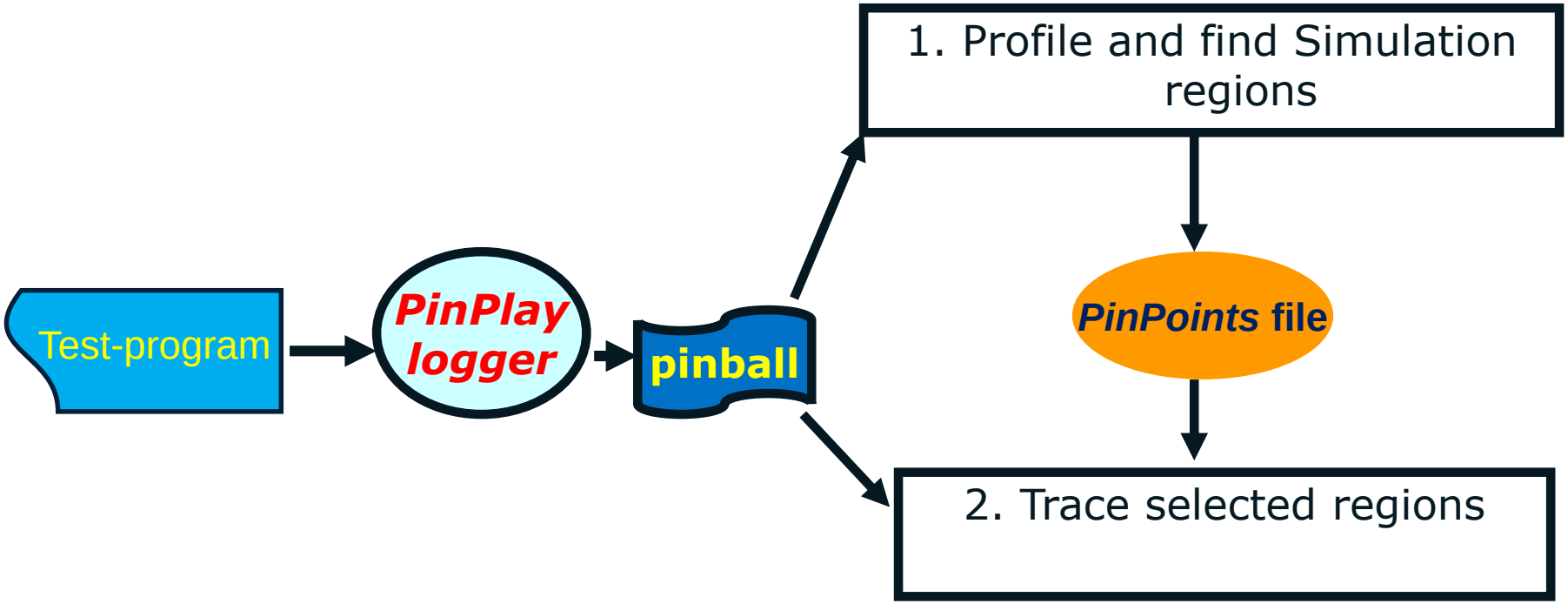


Problem: Two runs are not exactly same → PinPoints missed

Found this for 25/54 SPEC2006 runs!

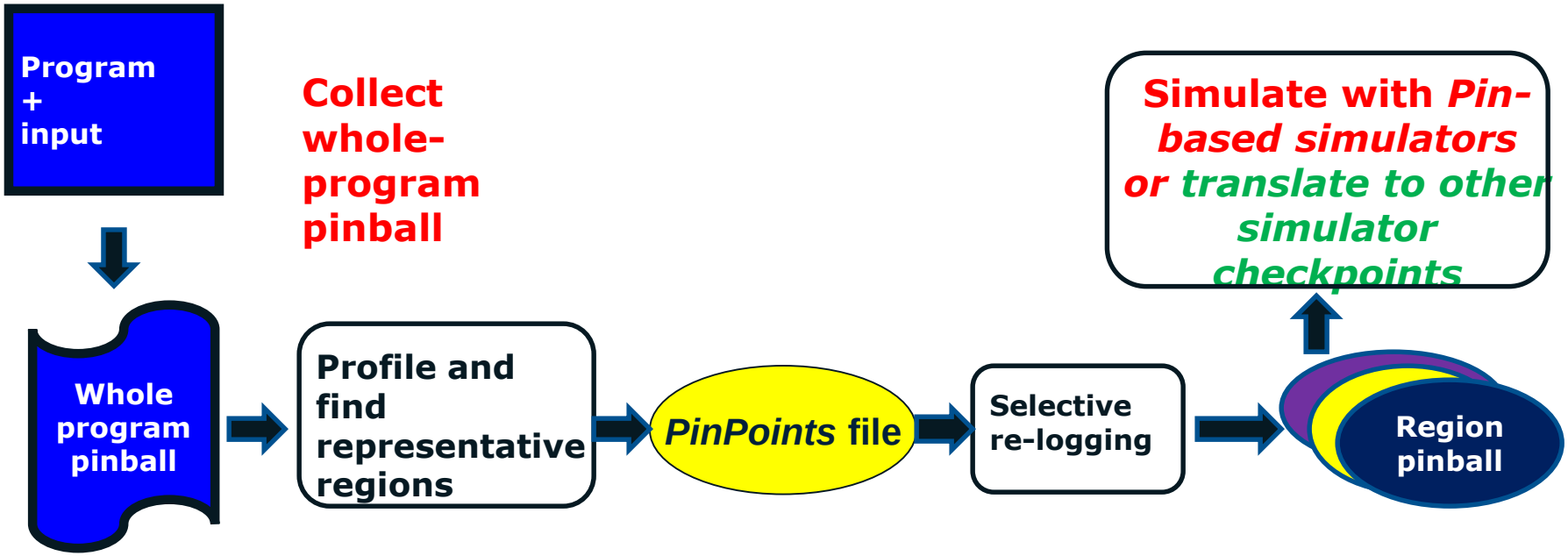
[*"PinPoints out of order" "PinPoint End seen before Start"*]

Enters PinPlay To Provide Repeatability



Two runs are same → PinPoints guaranteed to be reached

PinPlay + PinPoints: Basic Flow



- PinPoints are representative (validation/tuning possible)
- PinPoints cover << 1% of whole-program execution

➔ **vastly reduced simulation time**

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

Pinballs

Download Pinballs for full runs and PinPoints

SPEC CPU 2006

- Pinballs in Pinplay 1.1 format (Sniper 5.2+)
  - Whole program pinballs
    - Download 2.1 GiB (md5)
  - PinPoints
    - Download 2.1 GiB, no warmup, 1B instruction detailed region, maxK=1 (md5)
    - Download (p1/md5, p2/md5, p3/md5, p4/md5) 6.9 GiB, 100M instruction warmup, 30M instruction detailed region,

Summary : Pinballs

User-level x86 checkpoints:

1. Enable deterministic simulation
2. Portable : No OS dependence
3. Shareable : Relatively small in size

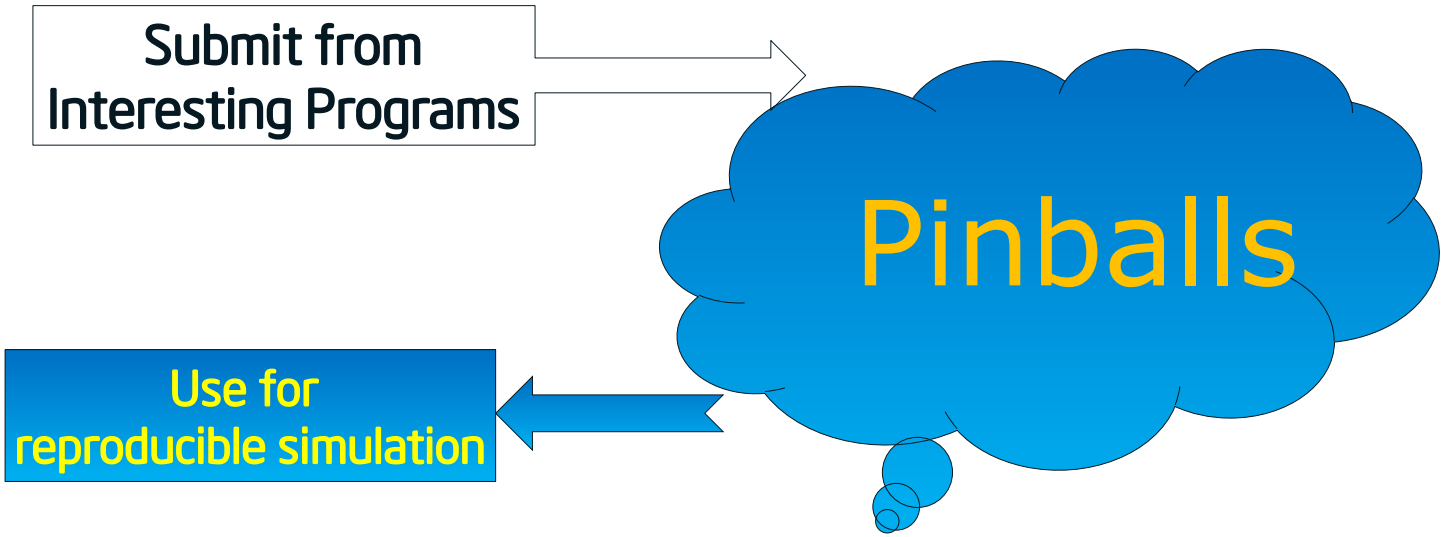
Generate with PinPlay : http://www.pinplay.org

Use for simulation:

1. Replay-enabled Pin-based simulator OR
   Translate to other simulator inputs
2. Whole-program or representative regions (*PinPoints*)

Call for action

1. Make a global repository a reality



2. Pinball converter for QEMU-based/other simulators

Upcoming Events

1. **Use of Pinball for debugging**:

DrDebug: CGO 2014 talk (Monday 2/17/2014) @4 PM

15:15PM-16:30PM Session 3: Tools for Program Introspection

DrDebug: Deterministic Replay based Cyclic Debugging with Dynamic Slicing

Yan Wang (UC Riverside), Harish Patil, Cristiano Pereira, Gregory Lueck (Intel Corporation), Rajiv Gupta, Iulian Neamtiu (UC Riverside)

2. **Creation of Pinballs for representative regions**:

PinPoints: ISCA 2014 tutorial (Sunday 15th June, 2014)

- T7: PinPoints: Simulation Region Selection with PinPlay and Sniper (morning)

(https://sites.google.com/site/pinpointstutorialisca14/)