Finding Concurrency Errors in Sequential Code

Oren Laadan, Chia-Che Tsai, Nicolas Viennot,

Chris Blinn, Peter Senyao Du, Junfeng Yang, Jason Nieh

Columbia University

orenl@cs.columbia.edu

Pre-Talk Quiz

- Instructions:
  - this is not a trick question

Pre-Talk Quiz

- Instructions:
  - this is not a trick question
- How many lines does this produce ?

```
$ ps aux | grep XYZ
```

Pre-Talk Quiz

- Instructions:
  - this is not a trick question
- How many lines does this produce ?

  $ ps aux | grep XYZ
- Answer:

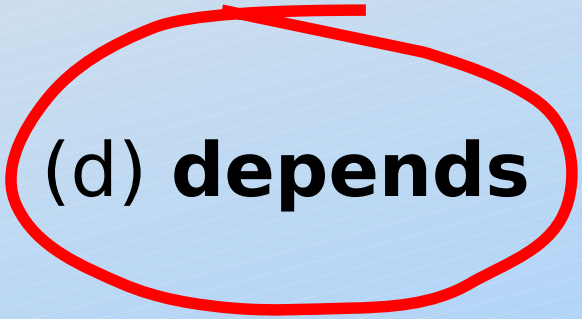
  (a) **0** (b) **1** (c) **neither** (d) **depends**

Pre-Talk Quiz

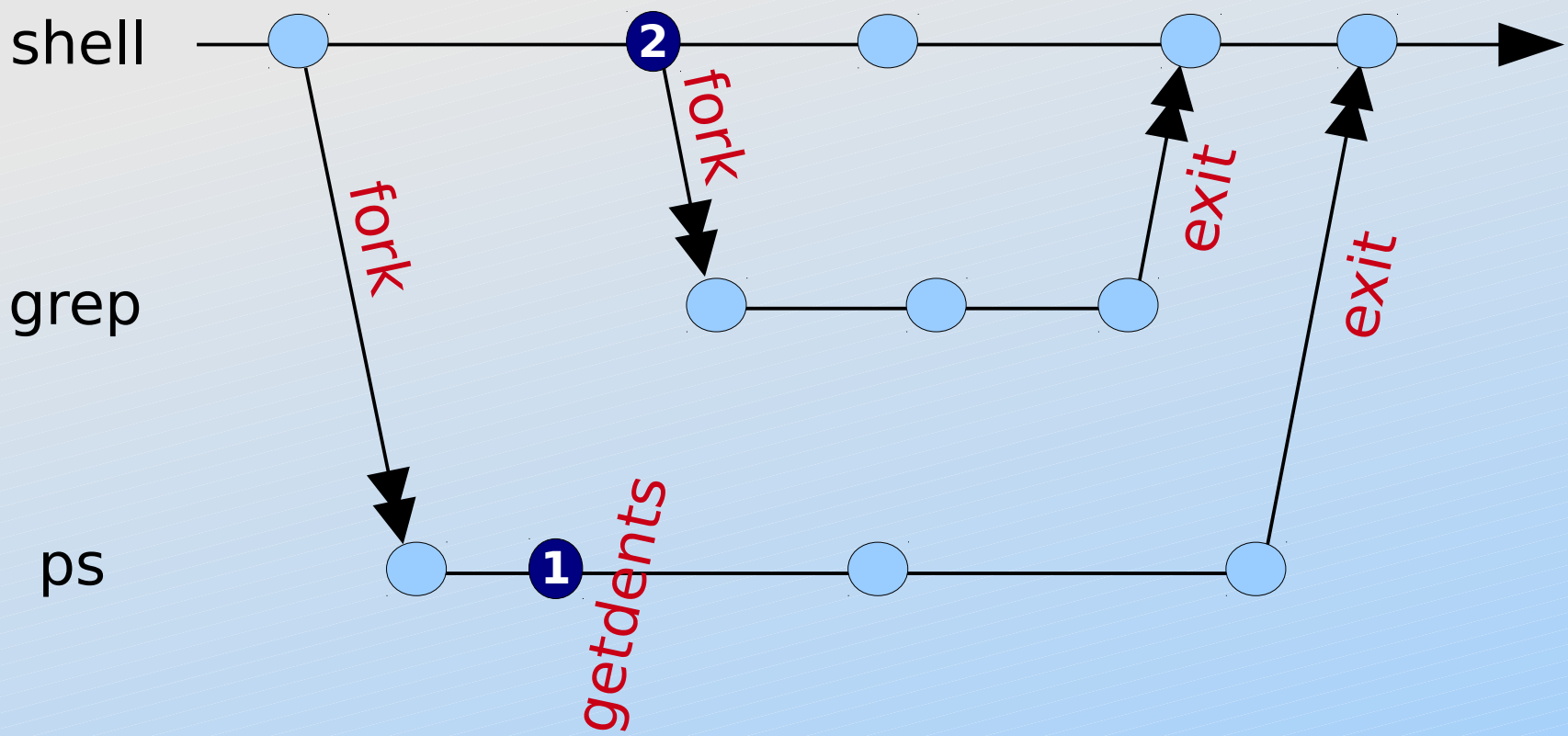
- Instructions:
  - this is not a trick question
- How many lines does this produce ?

  $ ps aux | grep XYZ
- Answer:

  (a) **0** (b) **1** (c) **neither** (d) **depends**



ps aux | grep XYZ

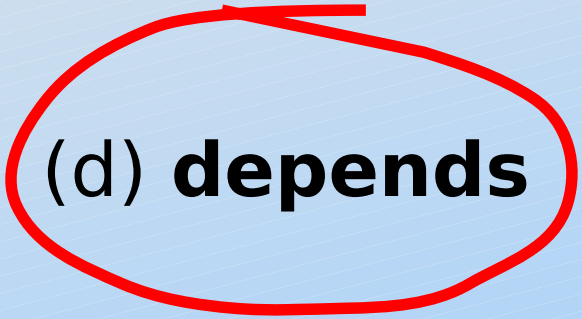


Pre-Talk Quiz

- Instructions:
  - this is not a trick question
- How many lines does this produce ?

  $ ps aux | grep XYZ
- Answer:

  (a) **0** (b) **1** (c) **neither** (d) **depends**



Does it matter ?!

Problem Statement

- **Process races** occur when multiple processes access shared OS resources without proper synchronization

Problem Statement

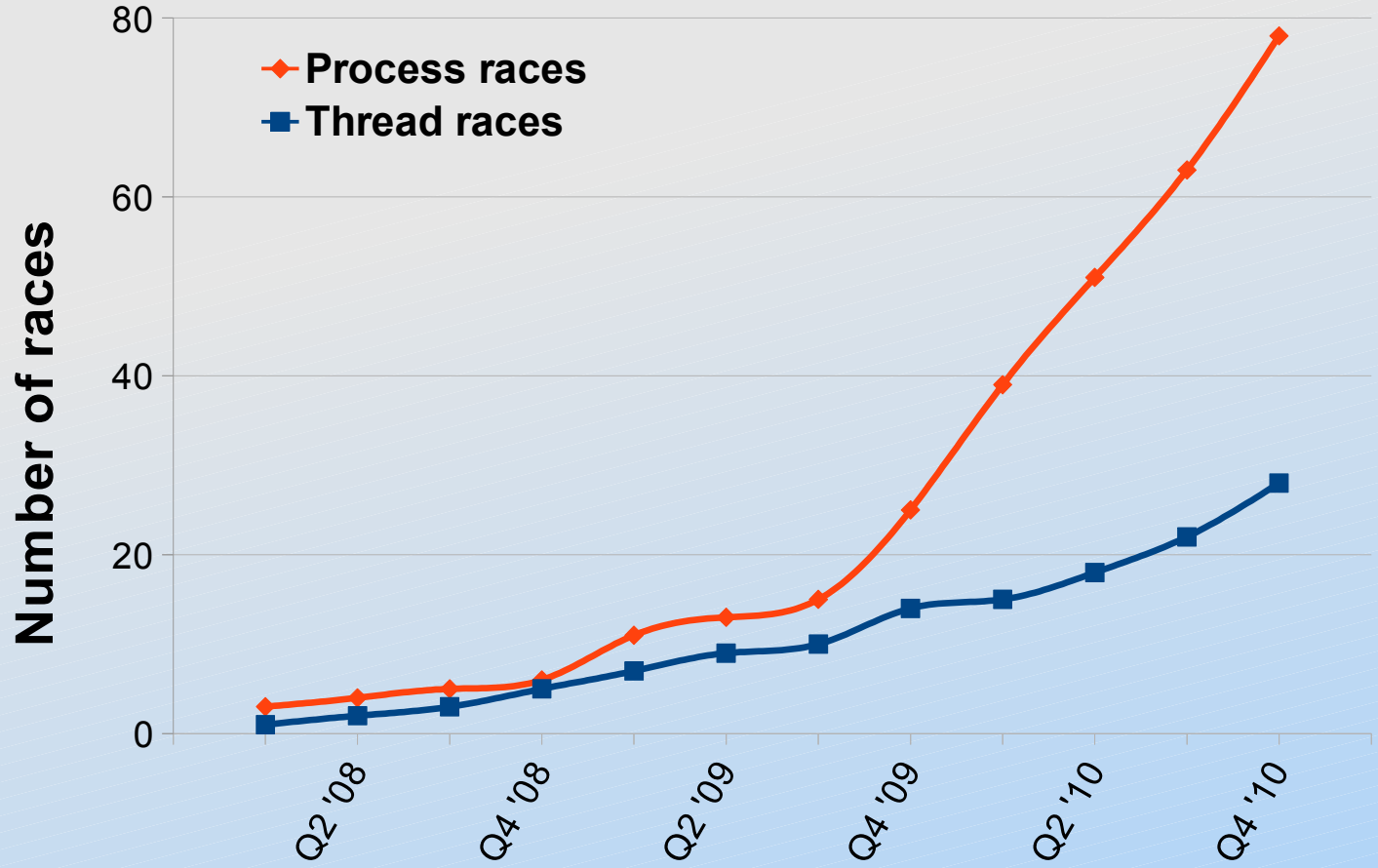
- **Process races** occur when multiple processes access shared OS resources without proper synchronization

- Examples:
  - ps aux | grep XYZ
  - startup/shutdown scripts (`upstart`)
  - parallel make (`make -j 4`)

Process Races

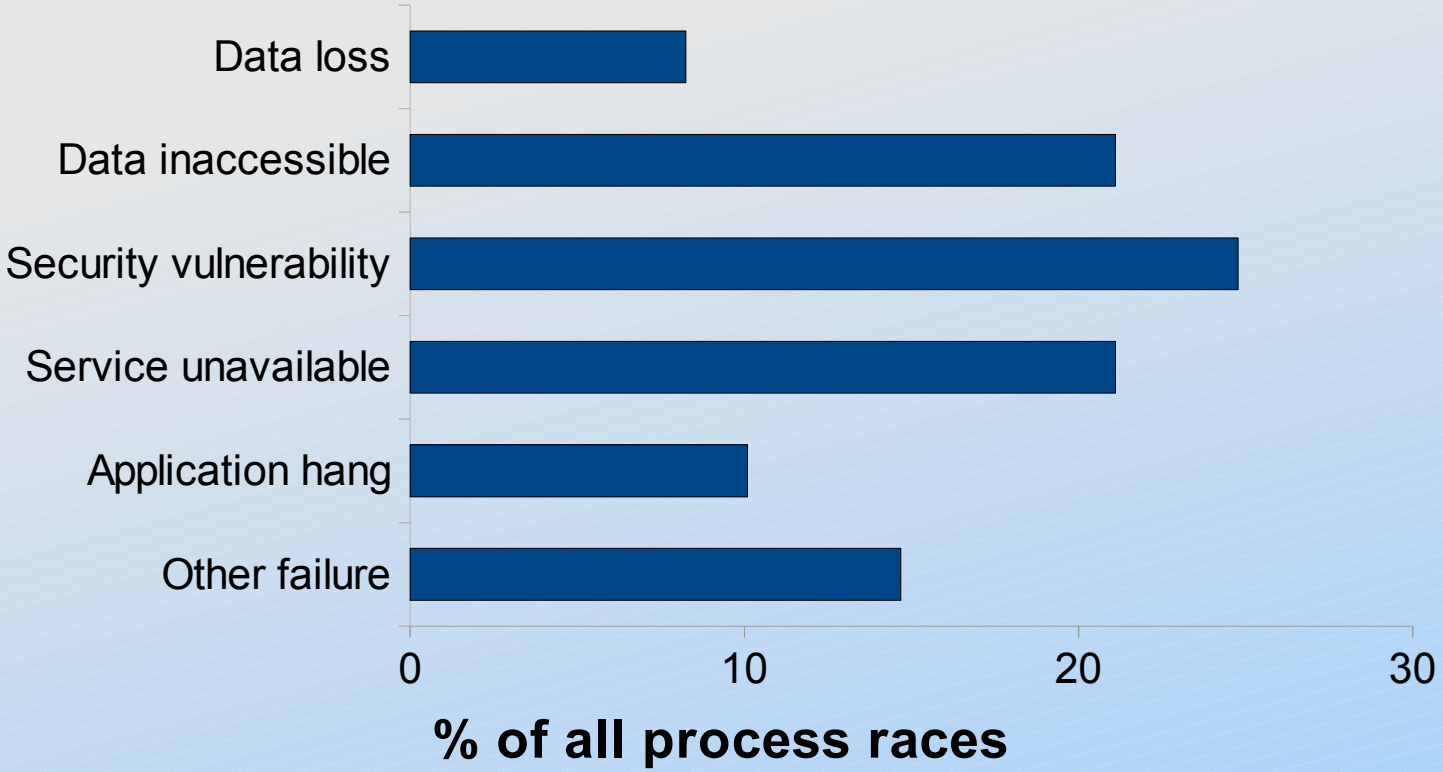
- So, are they really a problem ?
- If yes, how do we tackle them ?

Process Races... Numerous



(* out of sampled race reports)

Process Races... Dangerous



Mid-Talk Trivia

- In last three OSDI, SOSP, PLDI, ASPLOS, how many papers published on:
  - on thread races ?
  - on process races ?

Mid-Talk Trivia

- In last three OSDI, SOSP, PLDI, ASPLOS, how many papers published on:
  - on thread races ? **30+**
  - on process races ?

Mid-Talk Trivia

- In last three OSDI, SOSP, PLDI, ASPLOS, how many papers published on:
  - on thread races ? **30+**
  - on process races ? **2**

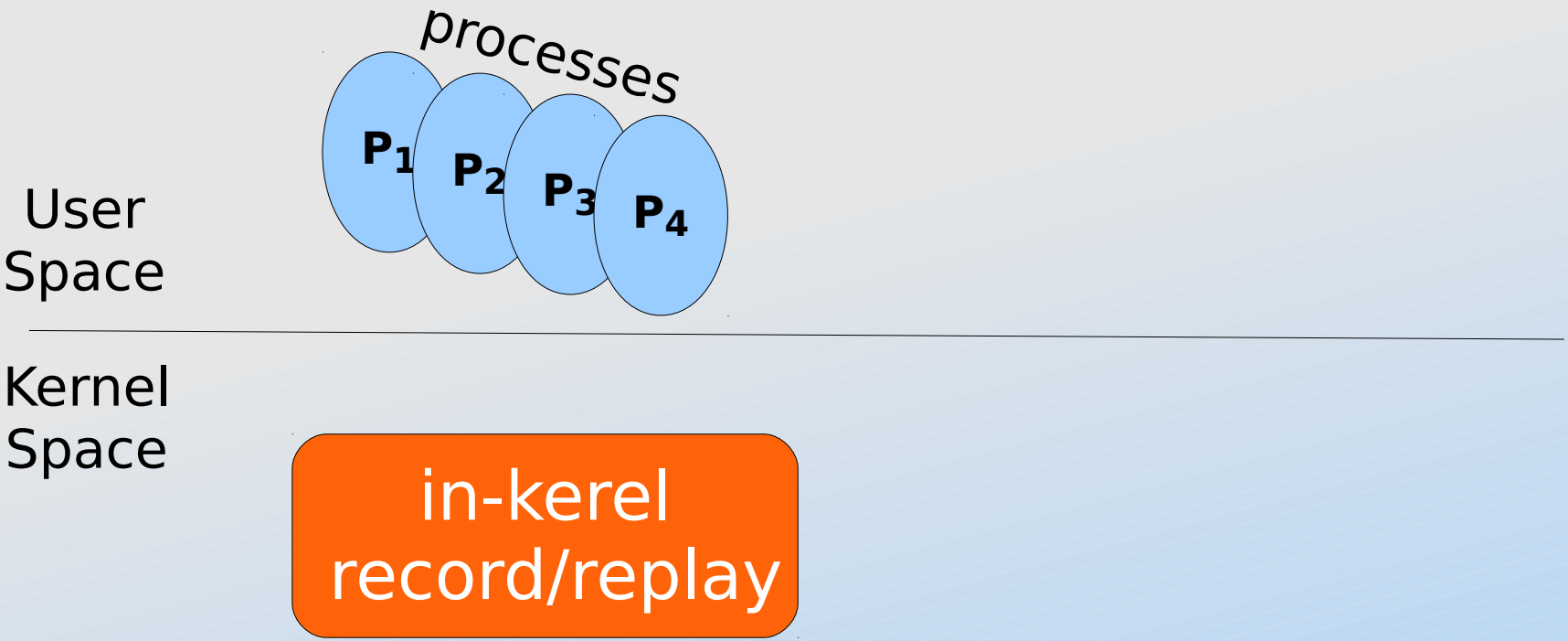
  none on process race detectors;

- Now, let's do something about that!

Process Races Challenges

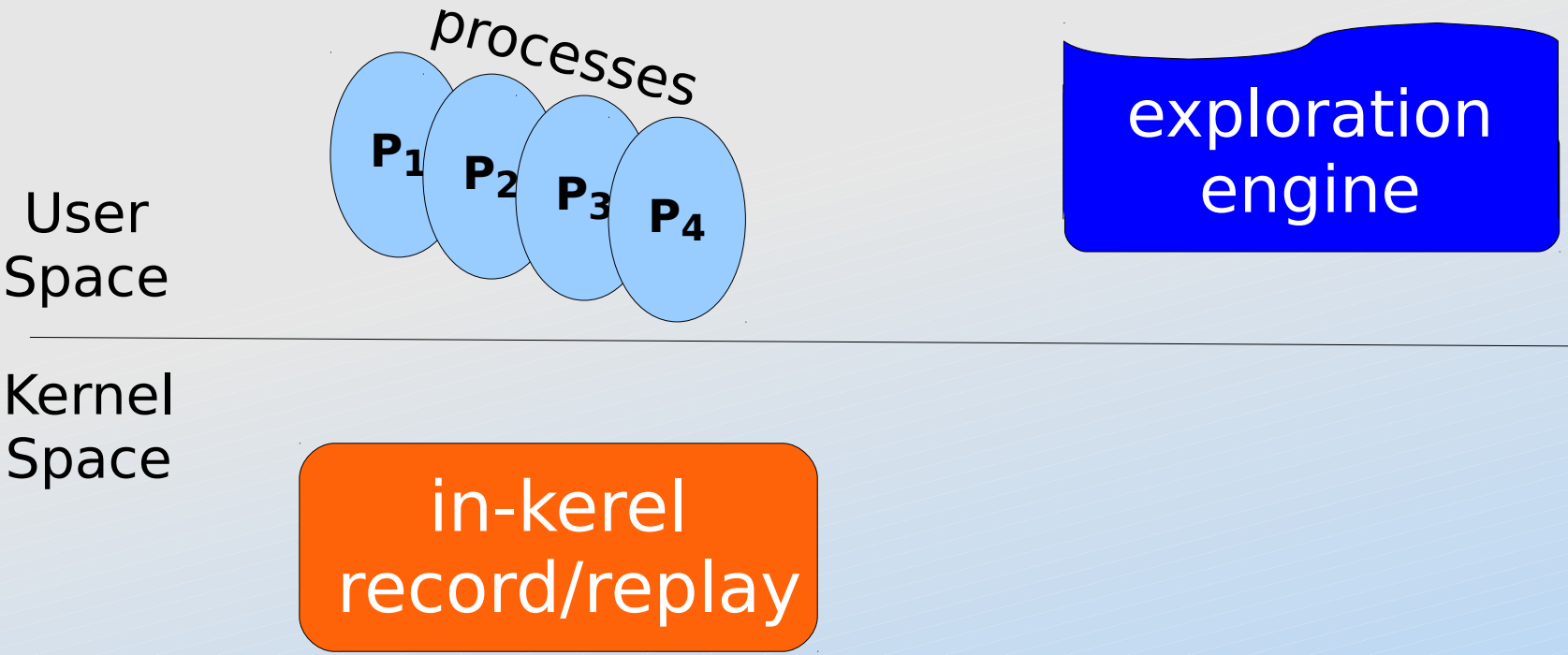
- Scope
  - diverse languages, programs, resources
- Coverage
  - depends on timing, environment, usage
- Algorithm
  - complex interactions of syscalls/resources
- False Positives
  - may produce too many false positives

Solution: RacePro



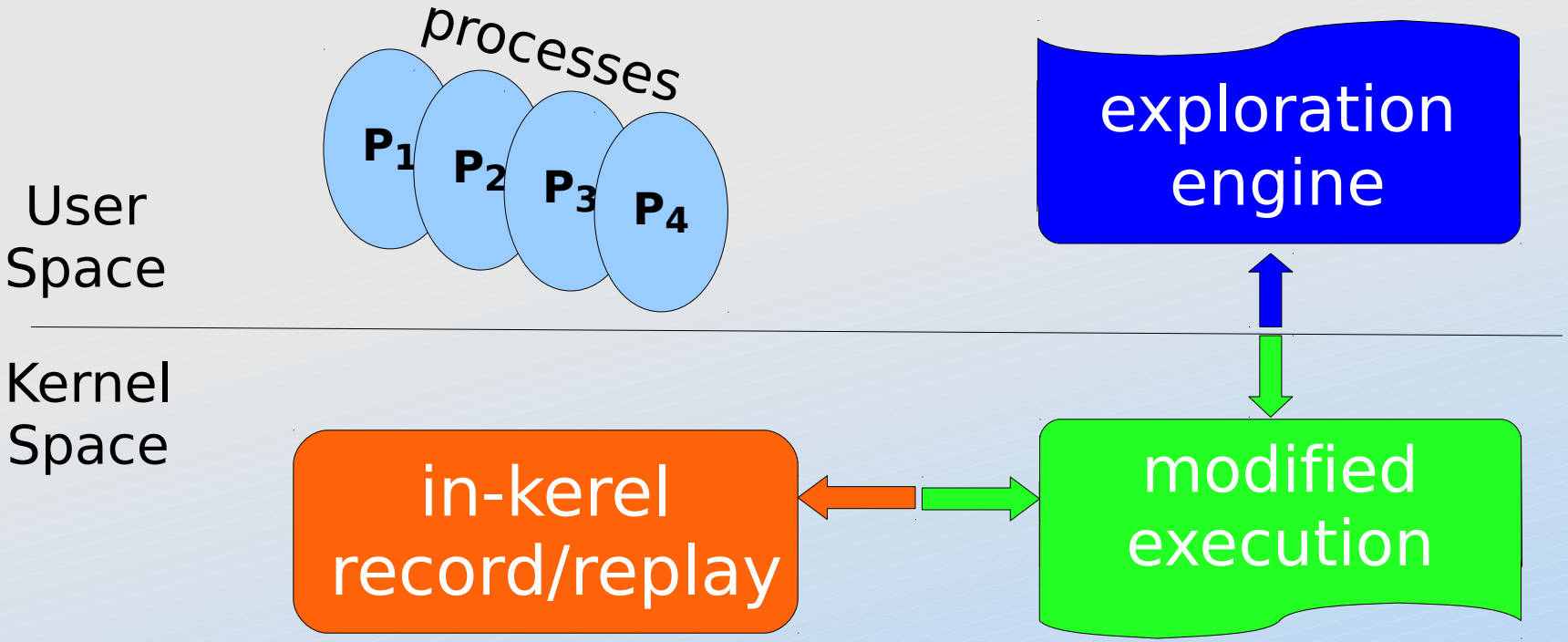
- online: record execution in-vivo

Solution: RacePro



- online: record execution in-vivo
- offline: analyze record to detect races

Solution: RacePro



- online: record execution in-vivo
- offline: analyze record to identify races
- offline: validate via real execution

Solution: RacePro

- Record execution in-vivo [AfterSight, USENIX '08]
  - capture interaction at syscall/resource level with low overhead
- Identify potential races offline
  - map resources to shared memory and syscalls to load/store operations
- Validate via real execution [RaceFuzzer PLDI '08]
  - replay modified recording to force races and check their effect

Results

- Detected 14 races:
  - 4 data-loss
  - 5 crash
  - 5 security
- Validation is crucial:
  - most races are benign or pruned
  - only 3-10% are proved harmful