Pervasive Detection of ~~Thread~~ **Process Races** In Deployed Systems

Columbia University
Oren Laadan
Nicolas Viennot
Chia-Che Tsai
Chris Blinn
Junfeng Yang
Jason Nieh

```
ps aux | grep pizza
```

ps aux | grep pizza outputs how many lines:

A) 0
B) 1
C) it depends
D) I can't think, you made me hungry with the pizza thing

ps aux | grep pizza outputs how many lines:

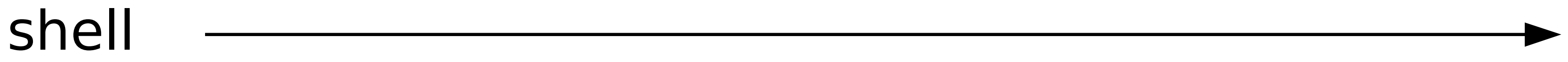
A) 0

B) 1

C) it depends

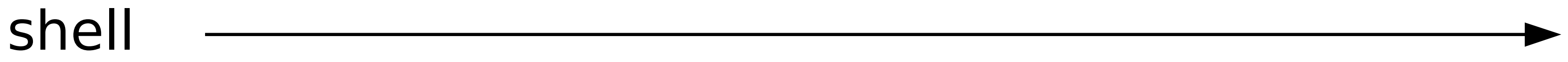
D) I can't think, you made me hungry with the pizza thing

ps aux | grep pizza



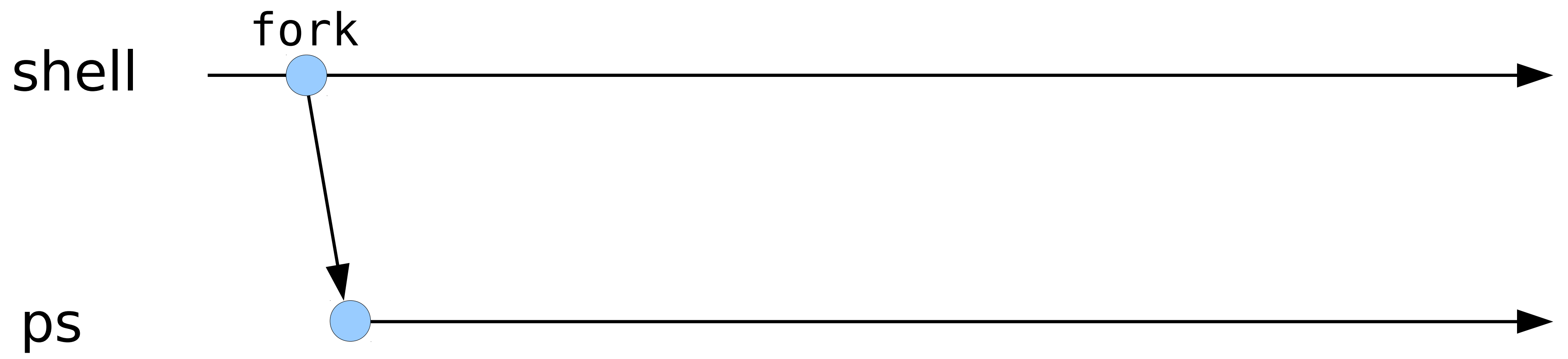
```
$
```

ps aux | grep pizza



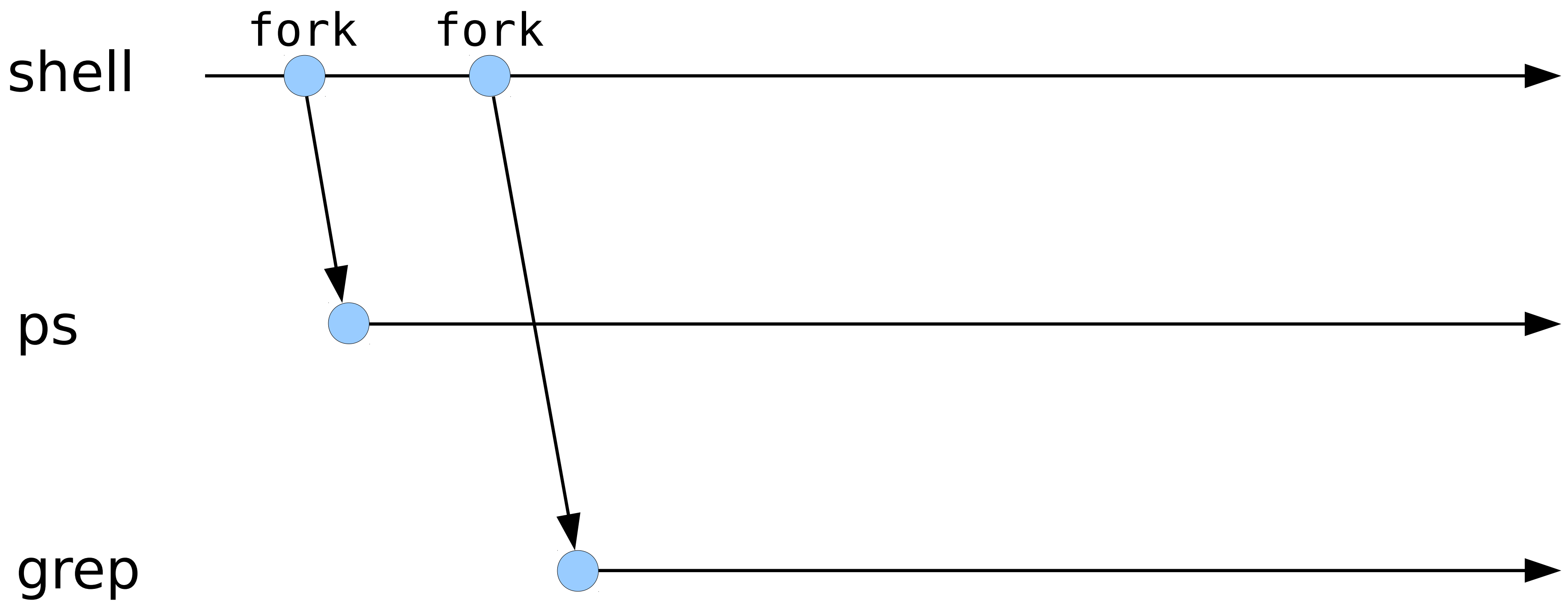
```
$ ps aux | grep pizza
```

ps aux | grep pizza



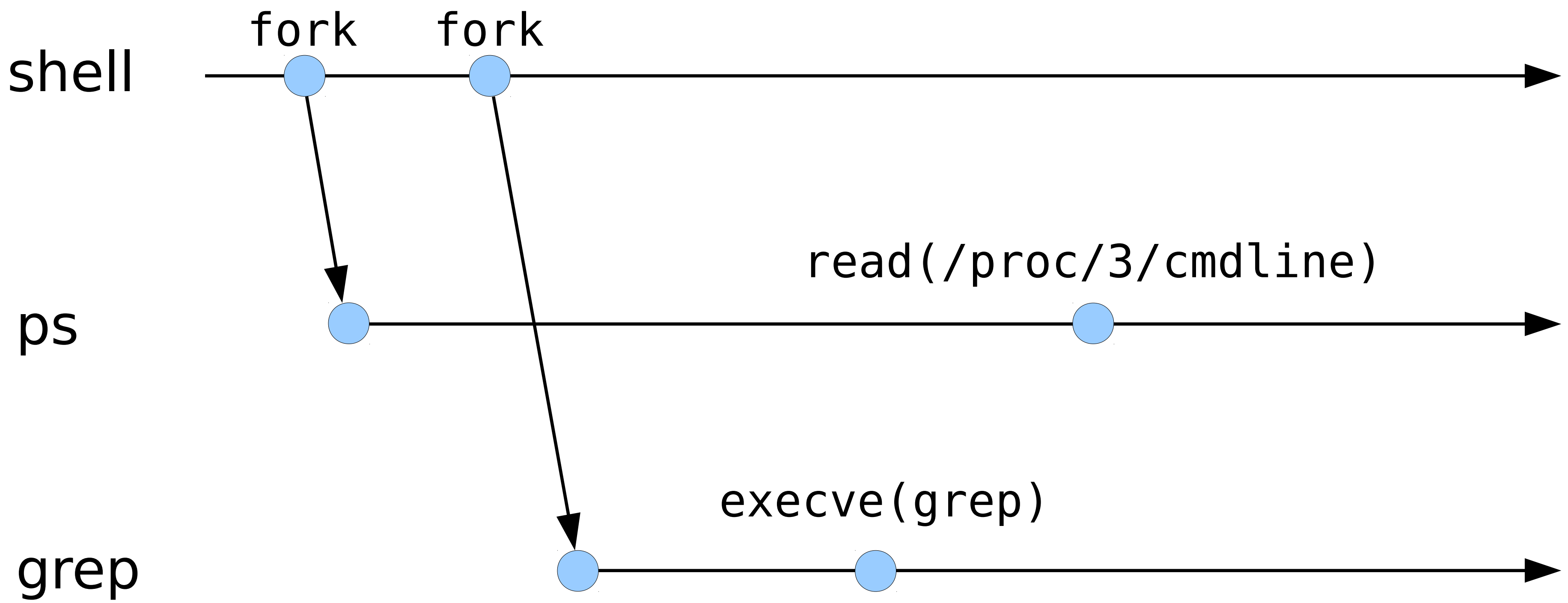
```
$ ps aux | grep pizza
```

ps aux | grep pizza



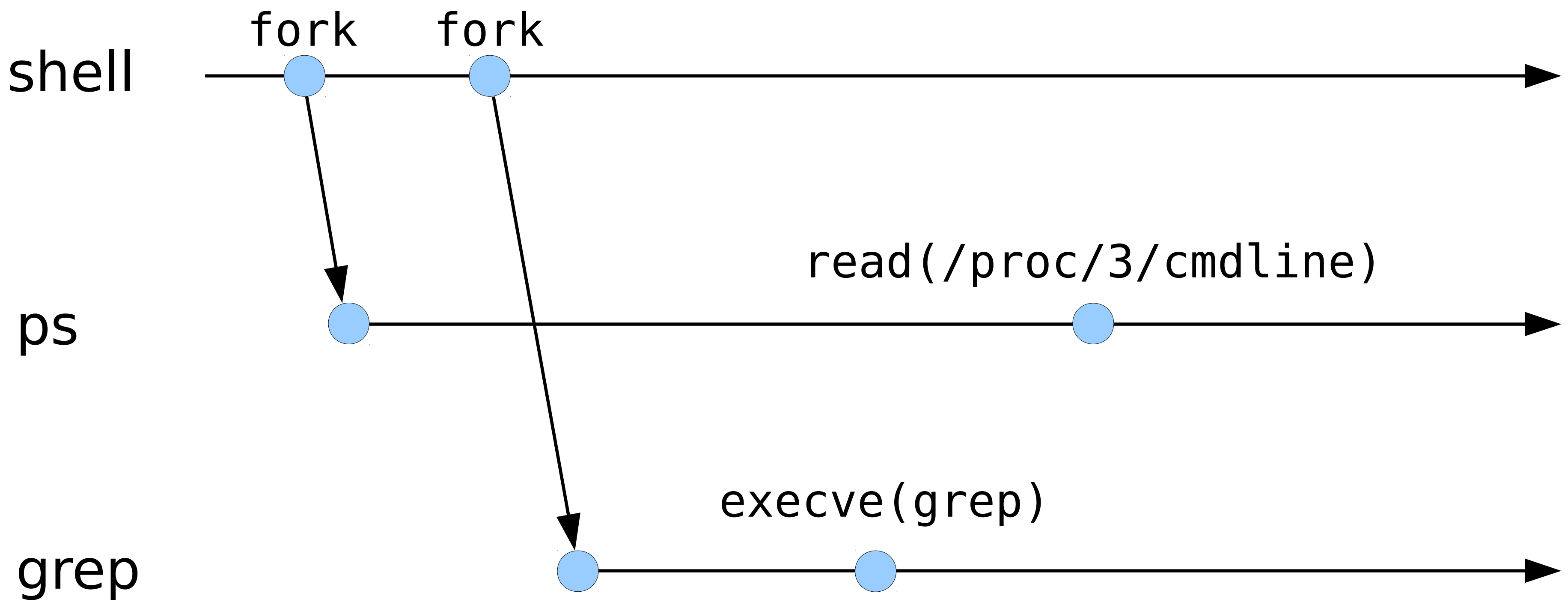
```
$ ps aux | grep pizza
```

ps aux | grep pizza



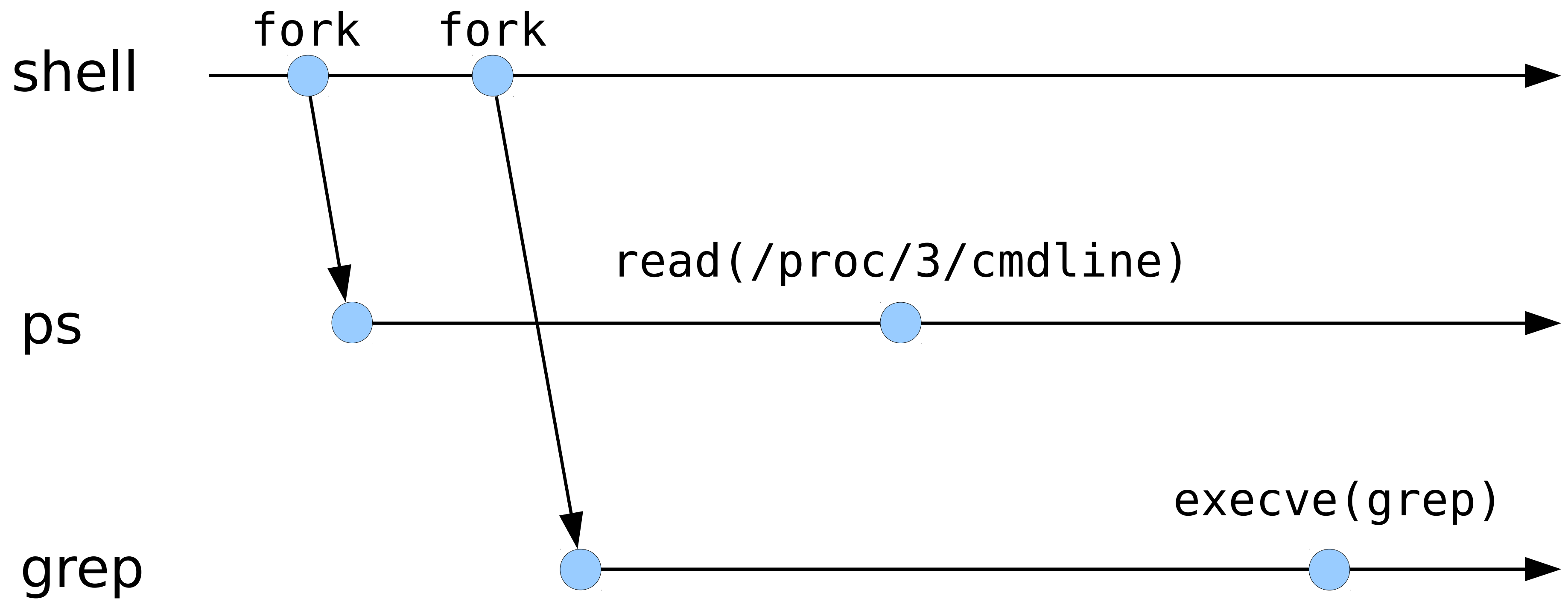
```
$ ps aux | grep pizza
```

ps aux | grep pizza



```
$ ps aux | grep pizza
nviennot  3 ... S+ 13:30  0:00 grep pizza
$
```

ps aux | grep pizza



```
$ ps aux | grep pizza
$
```

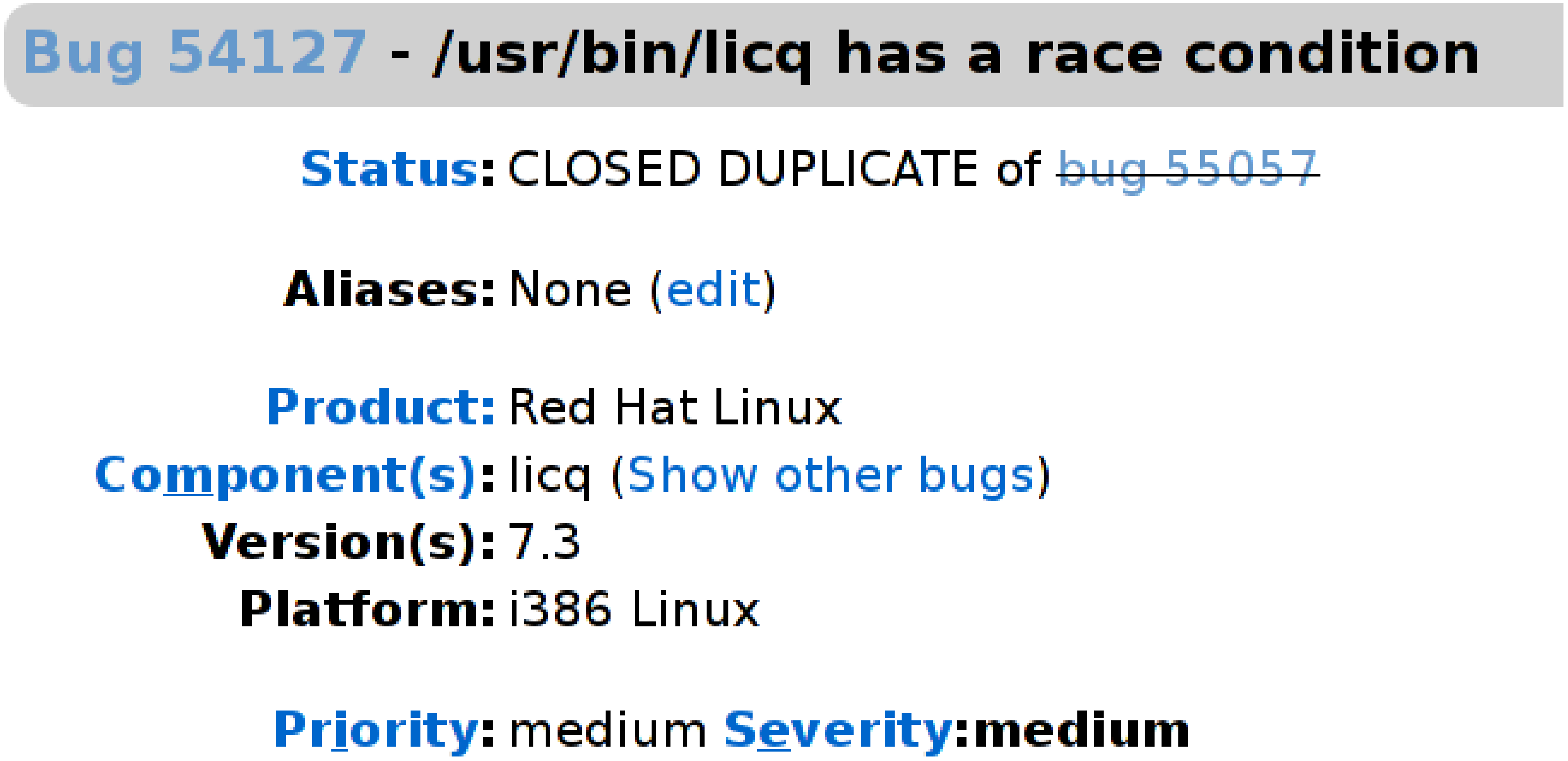
That's a process race

Process Races

- **Process races** occur when multiple processes access shared resources (such as files) without proper synchronization

- Examples:
  - parallel make (`make -j`) failure
  - ps aux | grep pizza

ps aux | grep xxx



Bug 54127 - /usr/bin/licq has a race condition

Status: CLOSED DUPLICATE of ~~bug 55057~~

Aliases: None (edit)

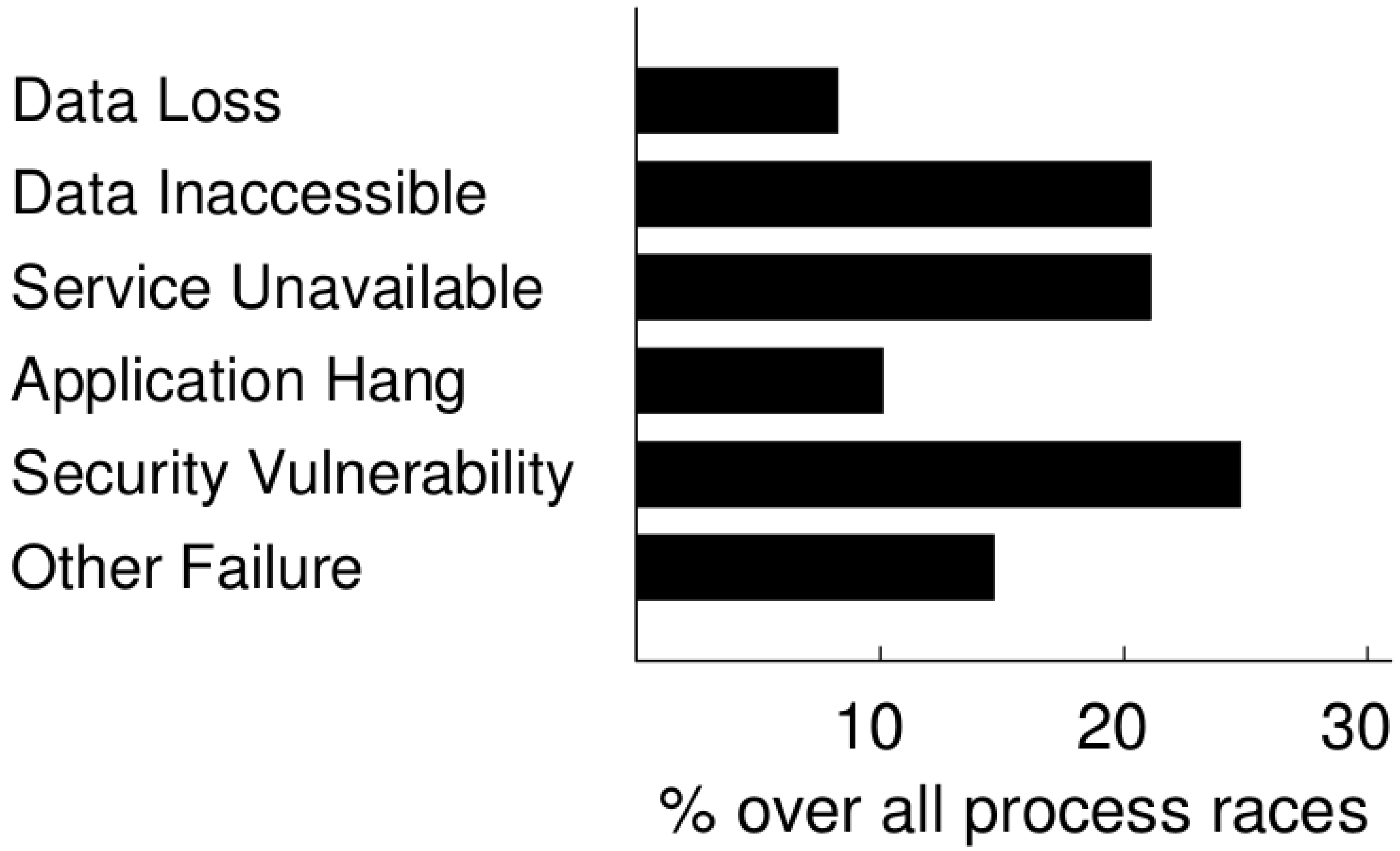
Product: Red Hat Linux
Component(s): licq (Show other bugs)
Version(s): 7.3
Platform: i386 Linux

Priority: medium **Severity:** **medium**

Process Races Are Numerous

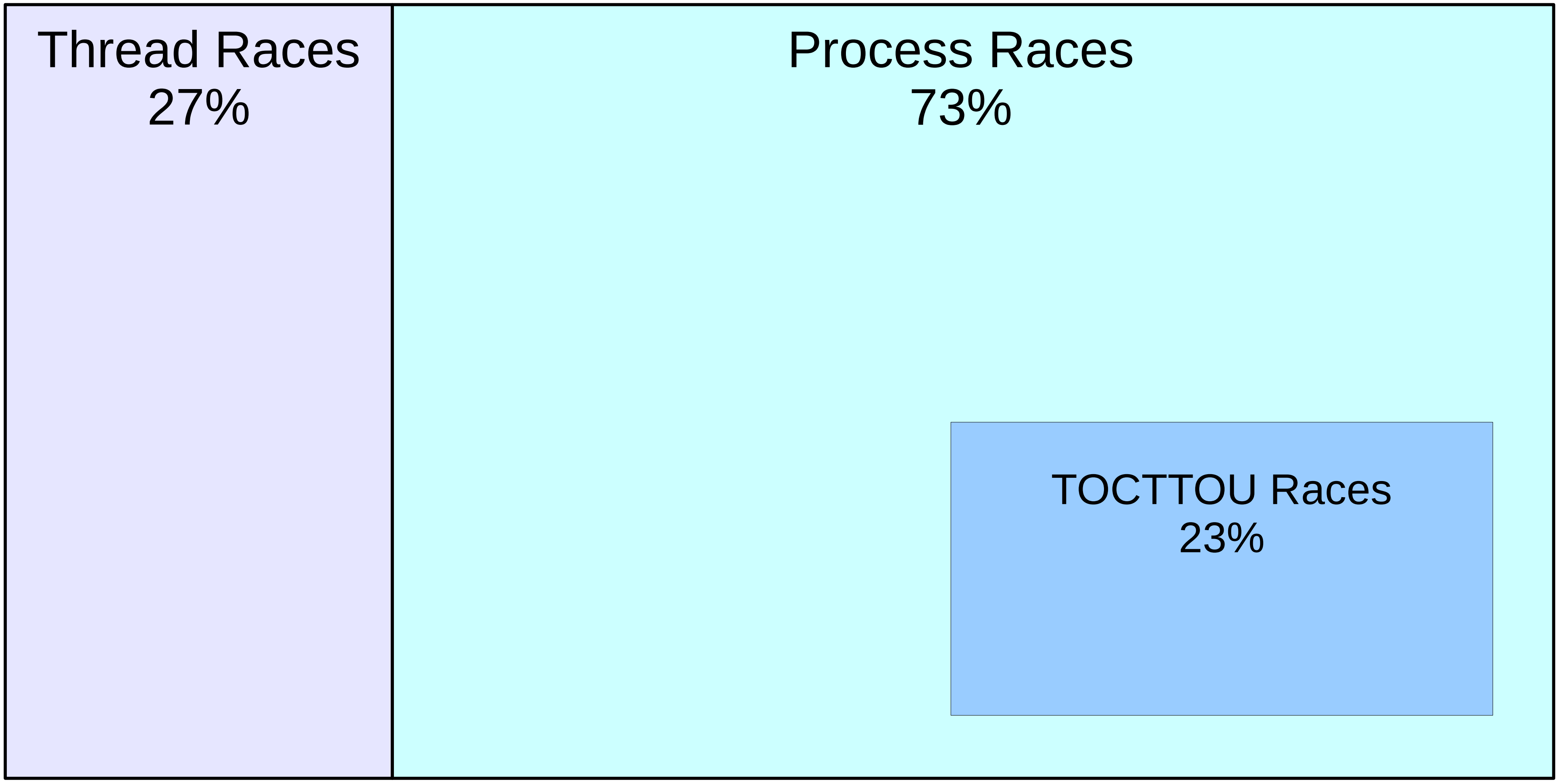
- Searched for “race” in the distro bug trackers
  (Ubuntu, Redhat/Fedora, Gentoo, Debian, CentOS)
- 9000+ results
- Sampled 500+ of them
- 109 unique bugs due to process races

Process Races Are Dangerous



Source: samples from Ubuntu, Redhat, Fedora, Gentoo, Debian, CentOS bug trackers

Process Races Are Hard To Detect



Thread races may be underrepresented in linux distributions bug trackers

General process races
cannot be detected
using existing race detectors

Not so surprising

- Different programs, written in different languages
- Access many different resources
- Syscalls semantics are a bit obscure
- Depends on user configuration, specific environment

Racepro

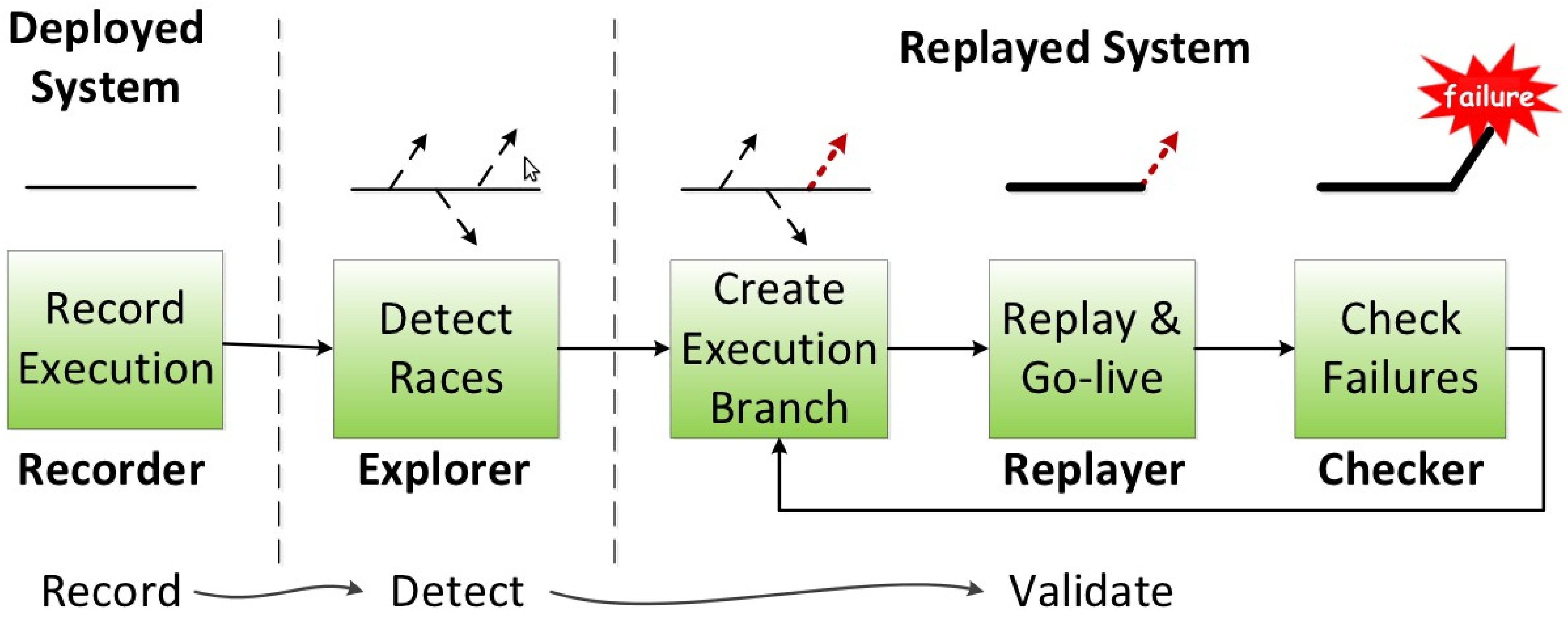
The *first* generic process race detection framework

“It's Amazing”
Nicolas Viennot

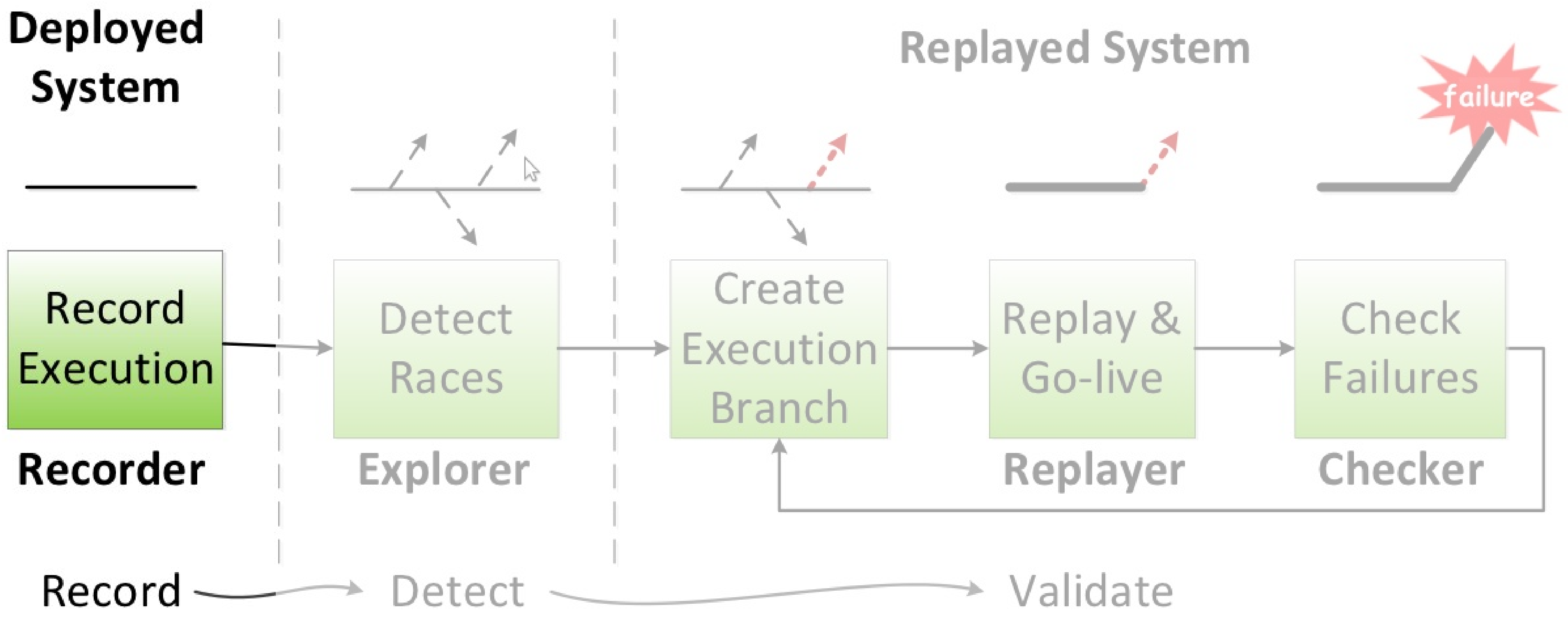
Racepro

- Detect generic process races
- Check deployed systems in-vivo
- Low overhead
- Transparent to applications
- Detected previously known and **unknown** bugs

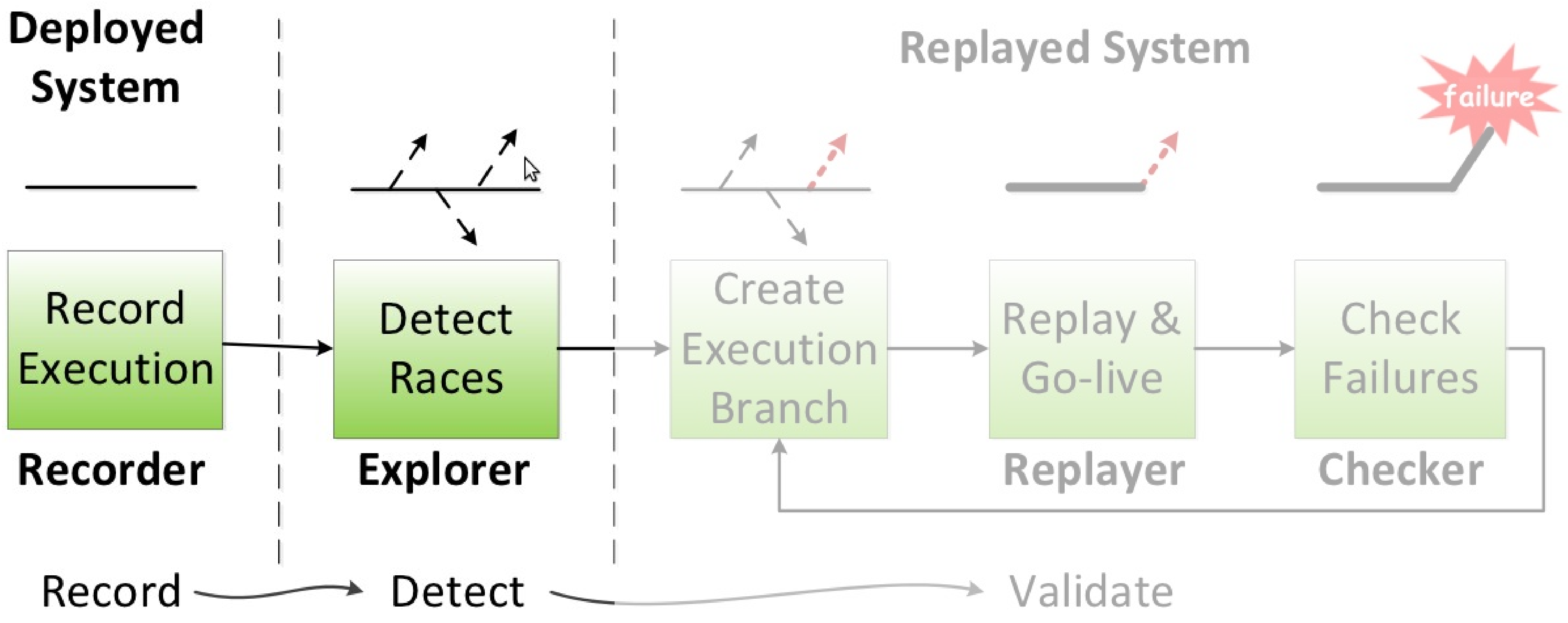
Racepro Workflow



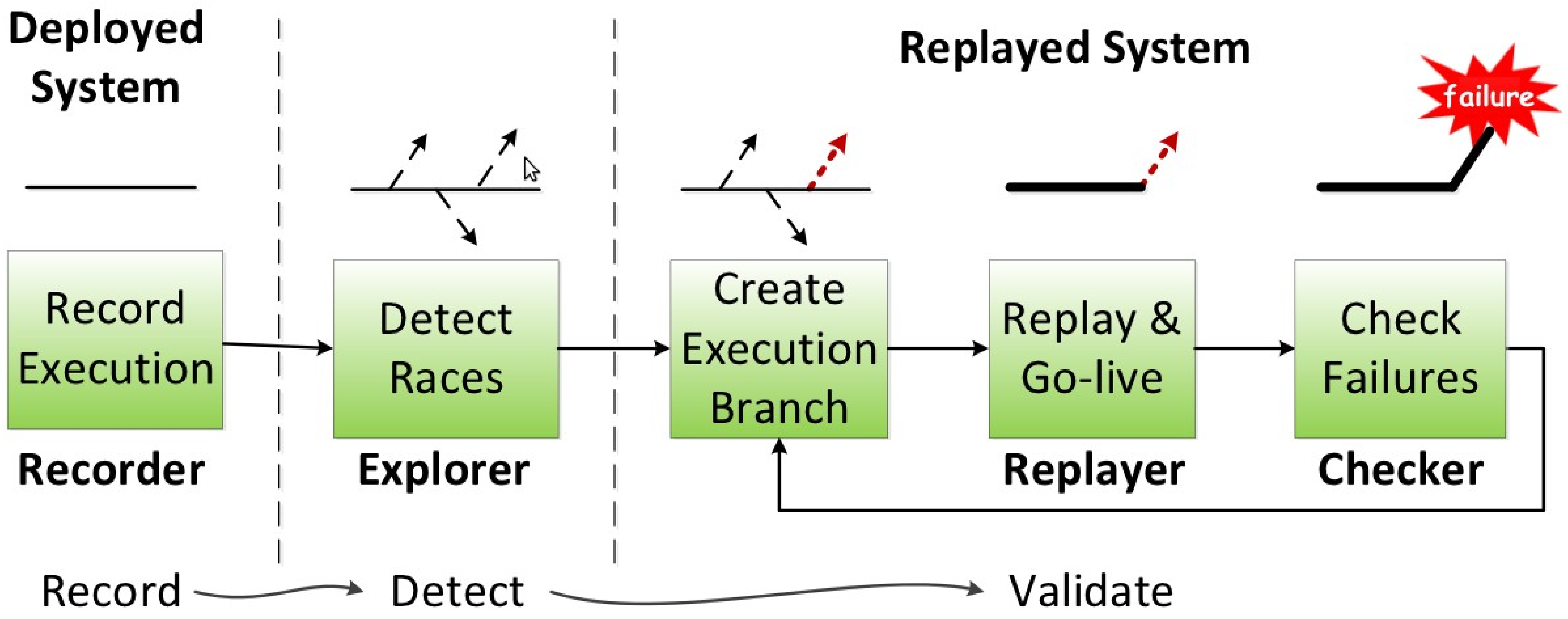
Racepro Workflow



Racepro Workflow



Racepro Workflow



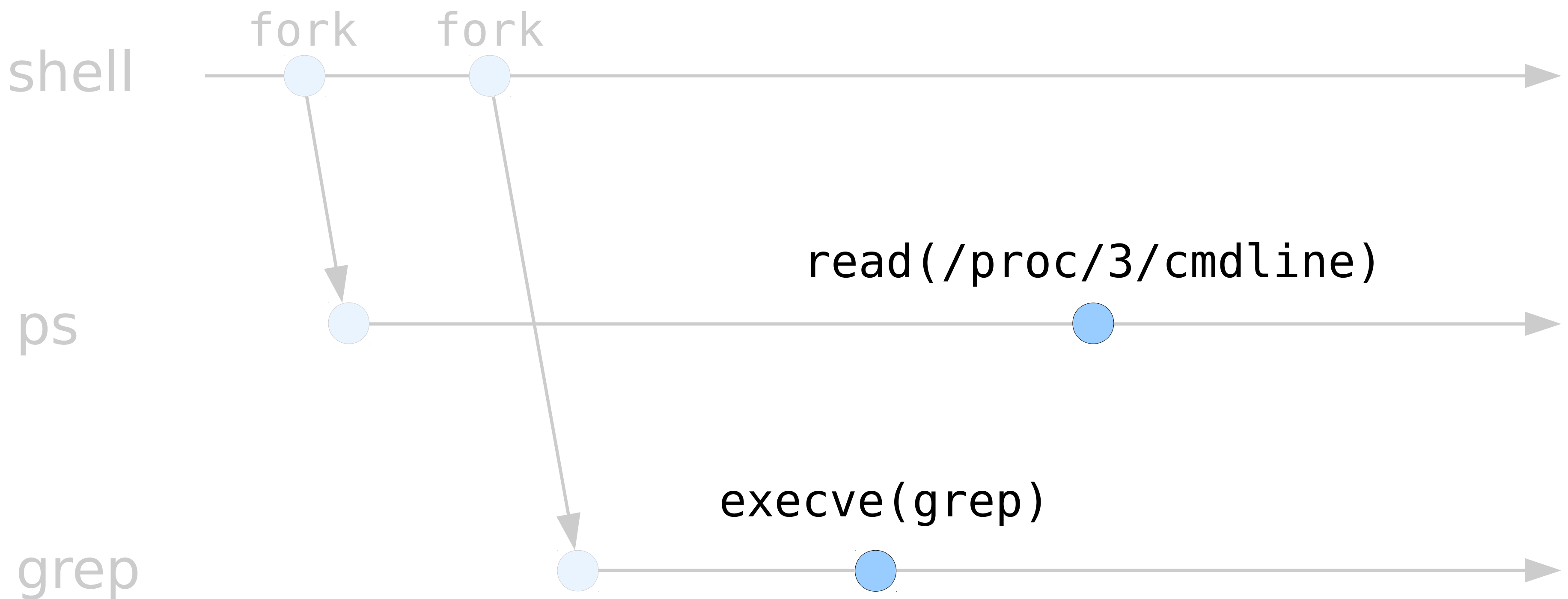
Recorder

- Builds on Scribe (Sigmetrics 2010)
- Lightweight kernel-level recorder
- Rendez-vous points:
  - Partial ordering of system calls
- Sync points:
  - Convert asynchronous events to synchronous events to track signals and shared memory

Benefits

- Tracks kernel object accesses
- Allows deterministic replay
- Enables transition to live execution
- Runs on commodity hardware, SMP friendly
- Low overhead
- Transparent to applications

ps aux | grep pizza



Log File Content

```
[2] read() = 11
[2]   read files_struct, id = 41, serial = 157
[2]   write file, id = 152, serial = 0
[2]   read pid, id = 40, serial = 17


[3] execve() = 0
[3]   write pid, id = 40, serial = 8
[3]     read inode, id = 1, serial = 0
[3]     read inode, id = 11, serial = 0
[3]     read inode, id = 1, serial = 0
[3]     read inode, id = 6, serial = 0
[3]     read inode, id = 13, serial = 0
[3]     read inode, id = 6, serial = 0
[3]     write futex, id = 51, serial = 0
```

Log File Content

```
[2] read() = 11
[2]   read files_struct, id = 41, serial = 157
[2]   write file, id = 152, serial = 0
[2]   read pid, id = 40, serial = 17


[3] execve() = 0
[3]   write pid, id = 40, serial = 8
[3]     read inode, id = 1, serial = 0
[3]     read inode, id = 11, serial = 0
[3]     read inode, id = 1, serial = 0
[3]     read inode, id = 6, serial = 0
[3]     read inode, id = 13, serial = 0
[3]     read inode, id = 6, serial = 0
[3]     write futex, id = 51, serial = 0
```

Log File Content

```
[2] read() = 11
[2]   read files_struct, id = 41, serial = 157
[2]   write file, id = 152, serial = 0
[2]   read pid, id = 40, serial = 17


[3] execve() = 0
[3]   write pid, id = 40, serial = 8
[3]     read inode, id = 1, serial = 0
[3]     read inode, id = 11, serial = 0
[3]     read inode, id = 1, serial = 0
[3]     read inode, id = 6, serial = 0
[3]     read inode, id = 13, serial = 0
[3]     read inode, id = 6, serial = 0
[3]     write futex, id = 51, serial = 0
```

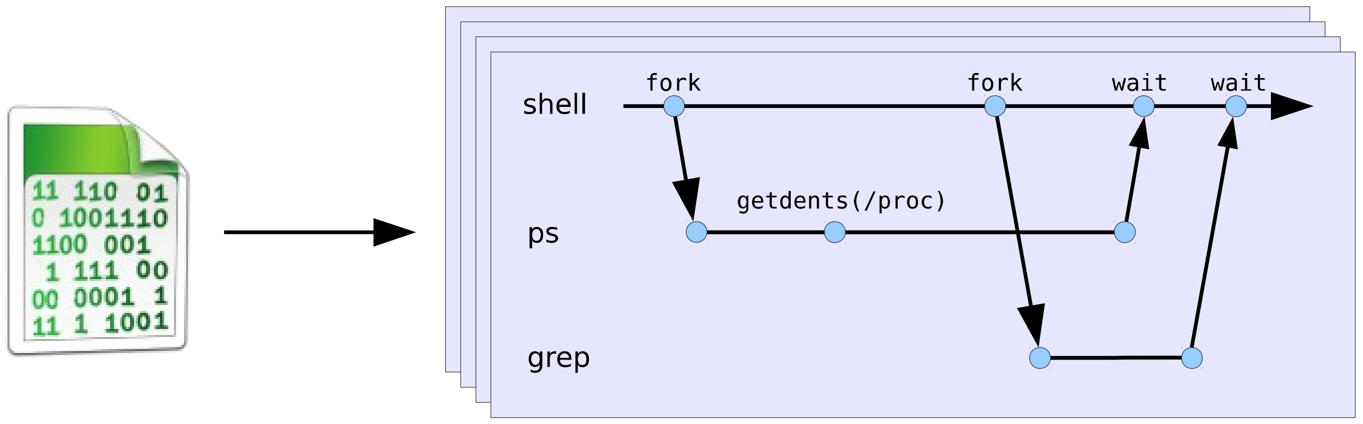
Log File Content

```
[2] read() = 11
[2]   read files_struct, id = 41, serial = 157
[2]   write file, id = 152, serial = 0
[2]   read pid, id = 40, serial = 17


[3] execve() = 0
[3]   write pid, id = 40, serial = 8
[3]     read inode, id = 1, serial = 0
[3]     read inode, id = 11, serial = 0
[3]     read inode, id = 1, serial = 0
[3]     read inode, id = 6, serial = 0
[3]     read inode, id = 13, serial = 0
[3]     read inode, id = 6, serial = 0
[3]     write futex, id = 51, serial = 0
```

Log File Content

```
[2] read() = 11
[2]   read files_struct, id = 41, serial = 157
[2]   write file, id = 152, serial = 0
[2]   read pid, id = 40, serial = 17


[3] execve() = 0
[3]   write pid, id = 40, serial = 8
[3]     read inode, id = 1, serial = 0
[3]     read inode, id = 11, serial = 0
[3]     read inode, id = 1, serial = 0
[3]     read inode, id = 6, serial = 0
[3]     read inode, id = 13, serial = 0
[3]     read inode, id = 6, serial = 0
[3]     write futex, id = 51, serial = 0
```

Step 2: Detection



Log file

Races

Model

System calls are translated to *load/store* micro-operations

Micro-operations

```
[2] read() = 11
[2]   read files_struct, id = 41, serial = 157
[2]   write file, id = 152, serial = 0
[2]   read pid, id = 40, serial = 17


[3] execve() = 0
[3]   write pid, id = 40, serial = 8
[3]     read inode, id = 1, serial = 0
[3]     read inode, id = 11, serial = 0
[3]     read inode, id = 1, serial = 0
[3]     read inode, id = 6, serial = 0
[3]     read inode, id = 13, serial = 0
[3]     read inode, id = 6, serial = 0
[3]     write futex, id = 51, serial = 0
```

Micro-operations

```
[2]   read files_struct, id = 41, serial = 157
[2]   write file, id = 152, serial = 0
[2]   read pid, id = 40, serial = 17
```

```
[3]   write pid, id = 40, serial = 8
[3]     read inode, id = 1, serial = 0
[3]     read inode, id = 11, serial = 0
[3]     read inode, id = 1, serial = 0
[3]     read inode, id = 6, serial = 0
[3]     read inode, id = 13, serial = 0
[3]     read inode, id = 6, serial = 0
[3]     write futex, id = 51, serial = 0
```

Micro-operations

```
[2]   read files_struct, id = 41, serial = 157
[2]   write file, id = 152, serial = 0
[3]   write pid, id = 40, serial = 8
[3]   read inode, id = 1, serial = 0
[3]   read inode, id = 11, serial = 0
[3]   read inode, id = 1, serial = 0
[3]   read inode, id = 6, serial = 0
[3]   read inode, id = 13, serial = 0
[3]   read inode, id = 6, serial = 0
[3]   write futex, id = 51, serial = 0
[2]   read pid, id = 40, serial = 17
```

Micro-operations

```
[2]    load 41
[2]    store 152
[3]    store 40
[3]    load 1
[3]    load 11
[3]    load 1
[3]    load 6
[3]    load 13
[3]    load 6
[3]    store 51
[2]    load 40
```

Micro-operations

```
[2]    load 41
[2]    store 152
[3]    store 40
[3]    load 1
[3]    load 11
[3]    load 1
[3]    load 6
[3]    load 13
[3]    load 6
[3]    store 51
[2]    load 40
```

You can now run your favorite thread race algorithm !

Micro-operations

```
[2]    load 41
[2]    store 152
[3]    store 40
[3]    load 1
[3]    load 11
[3]    load 1
[3]    load 6
[3]    load 13
[3]    load 6
[3]    store 51
[2]    load 40
```

Racy Instructions !

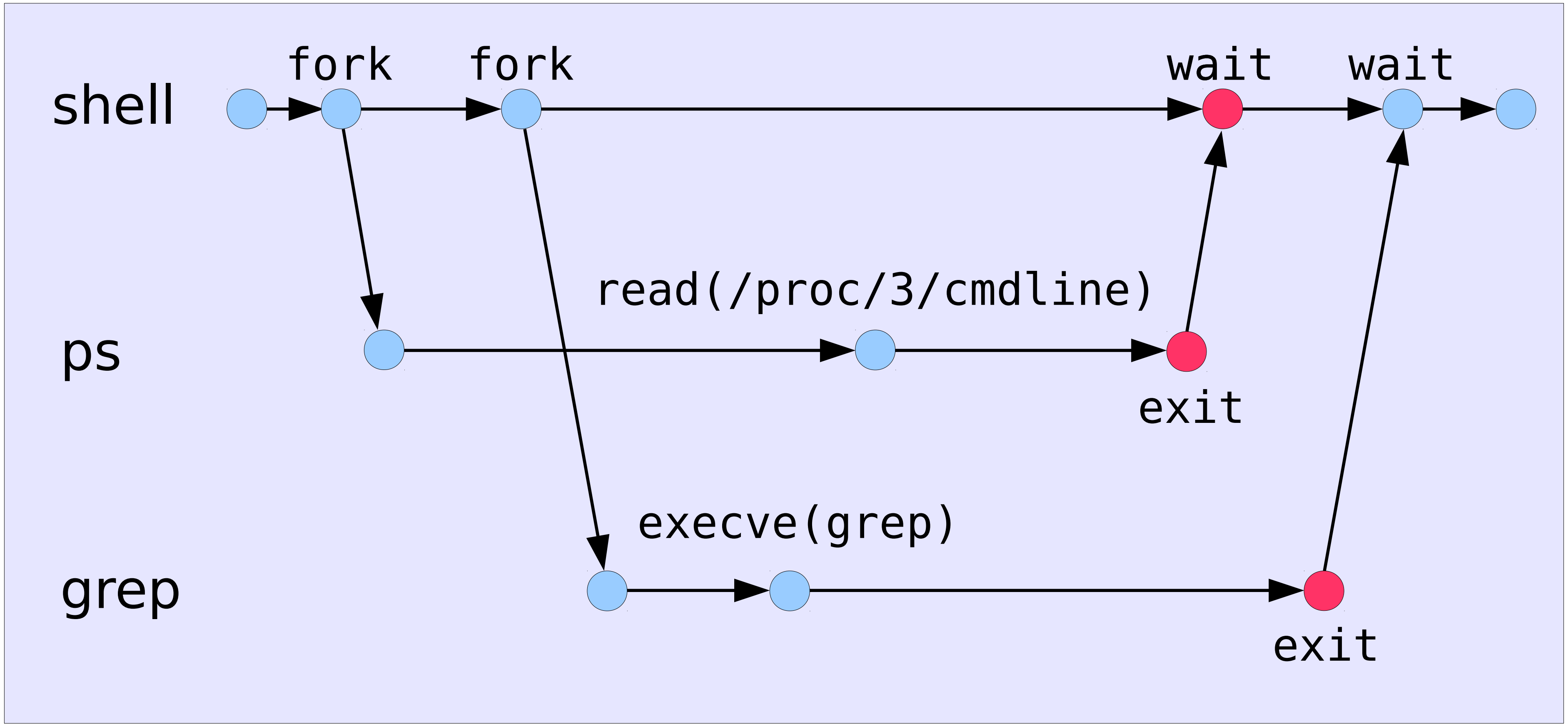
You can now run your favorite thread race algorithm !

Other kinds of races...

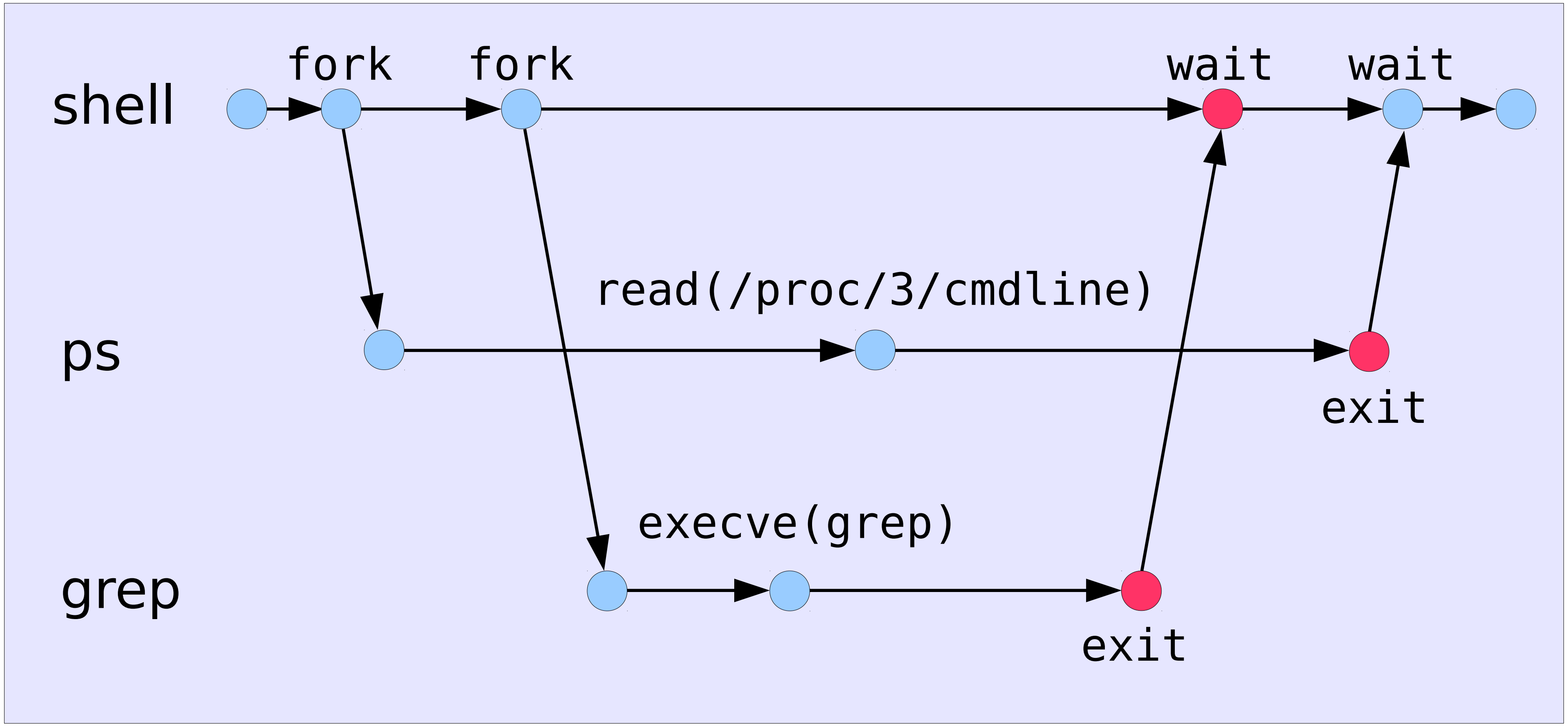
Wait-Wakeups Race

- A waiting syscall can be woken up by many matching wakeup syscalls
- Only Racepro detect such races
- Example:
  - read() on pipe can be woken by any writers
  - waitpid() can be woken by any children

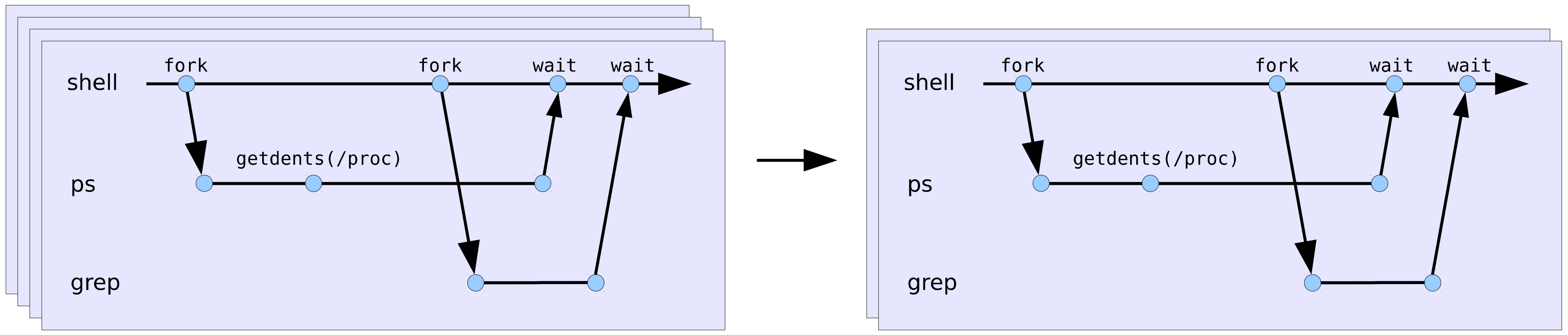
Wait-Wakeups Race Example



Wait-Wakeups Race Example



Step 3: Validation



Races

Harmful Races

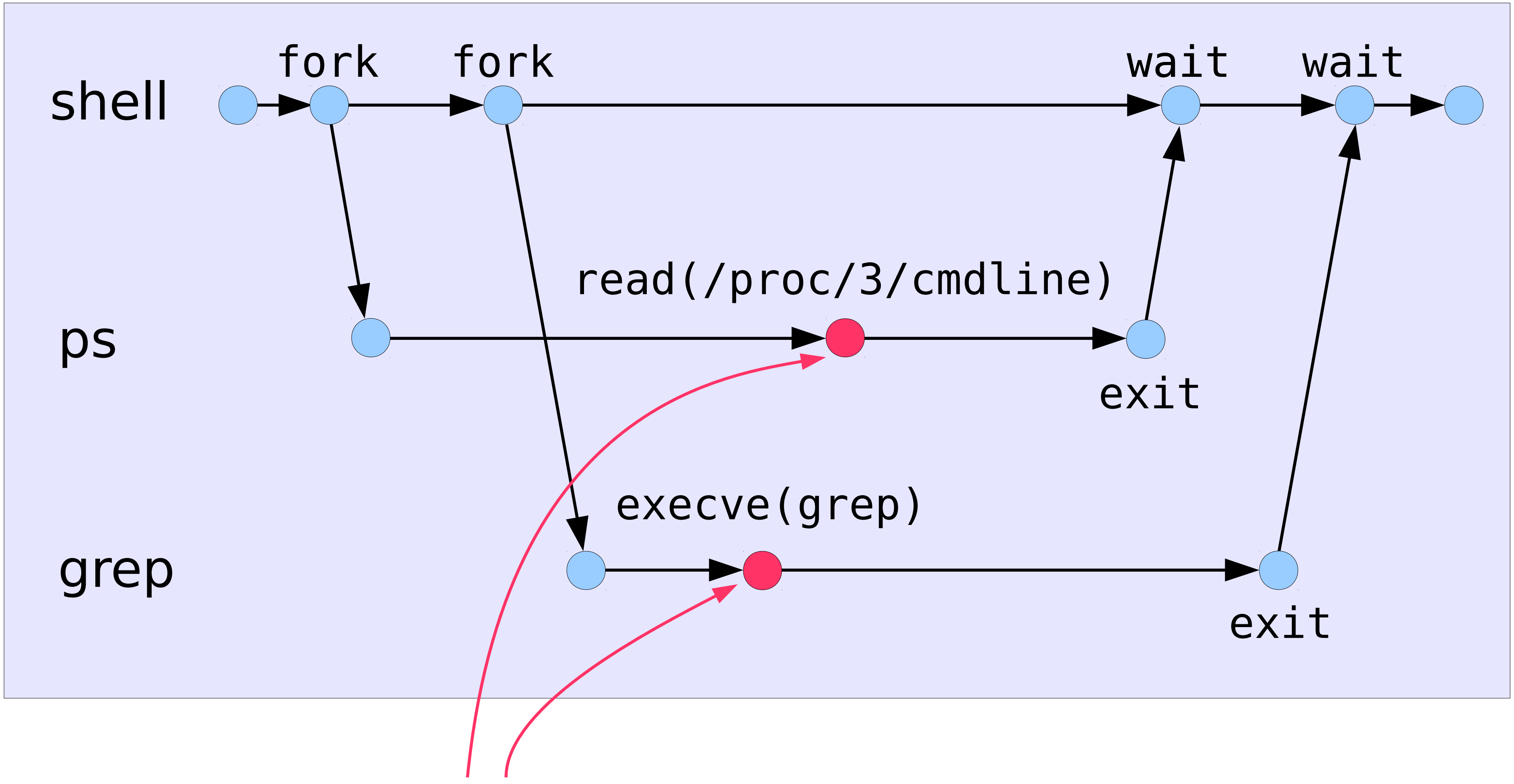
Validation Overview

- Create execution branch: **Modified** version of the original execution that makes the race occur by changing the order of system calls
- Problem: change in the middle of the recording can make the replay diverge
- Solution: truncate the log file after the modification and transition to live execution

Validation Steps

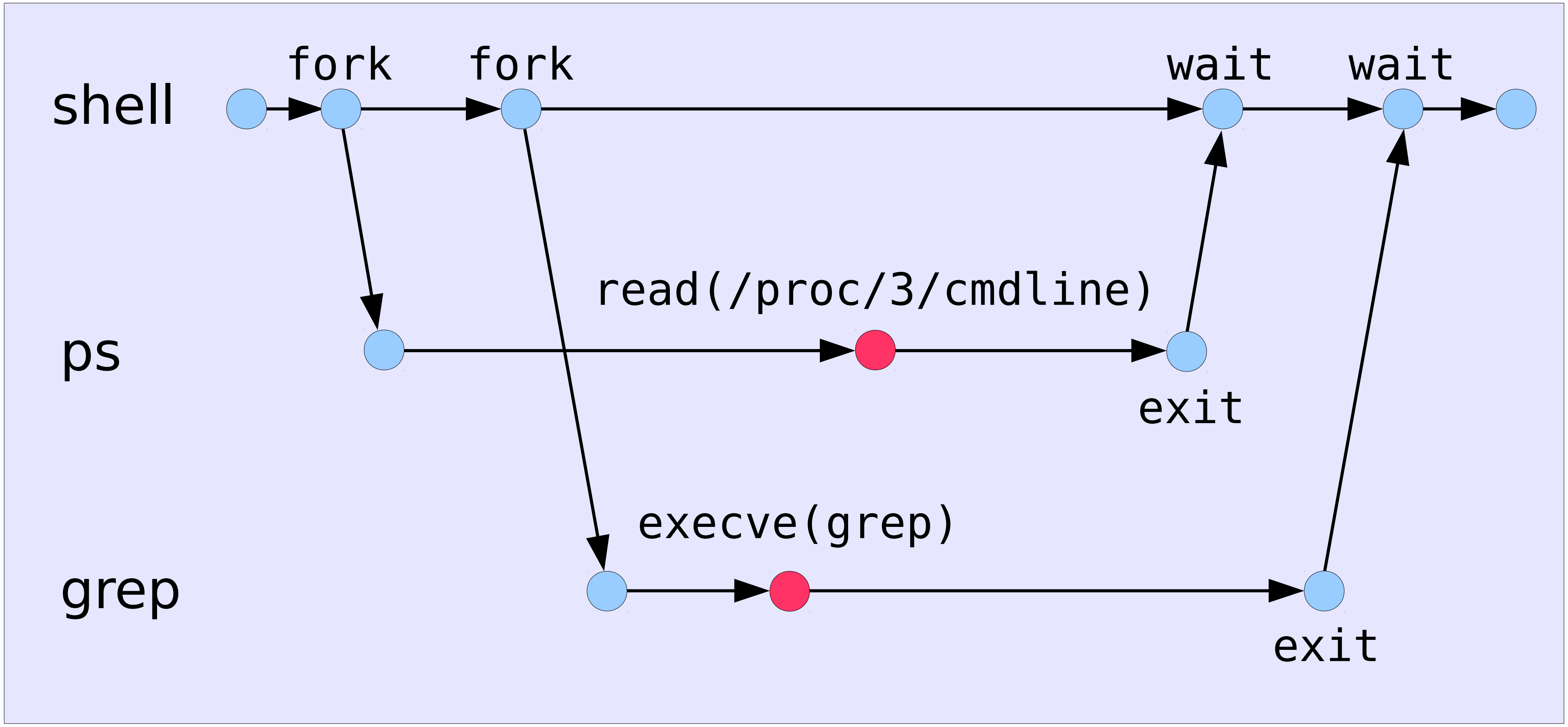
- Deterministic replay until race occurs, including replaying internal kernel state
- Replay the reordered racy system calls
- Transition to live execution
- Run built-in or custom checkers

Validation

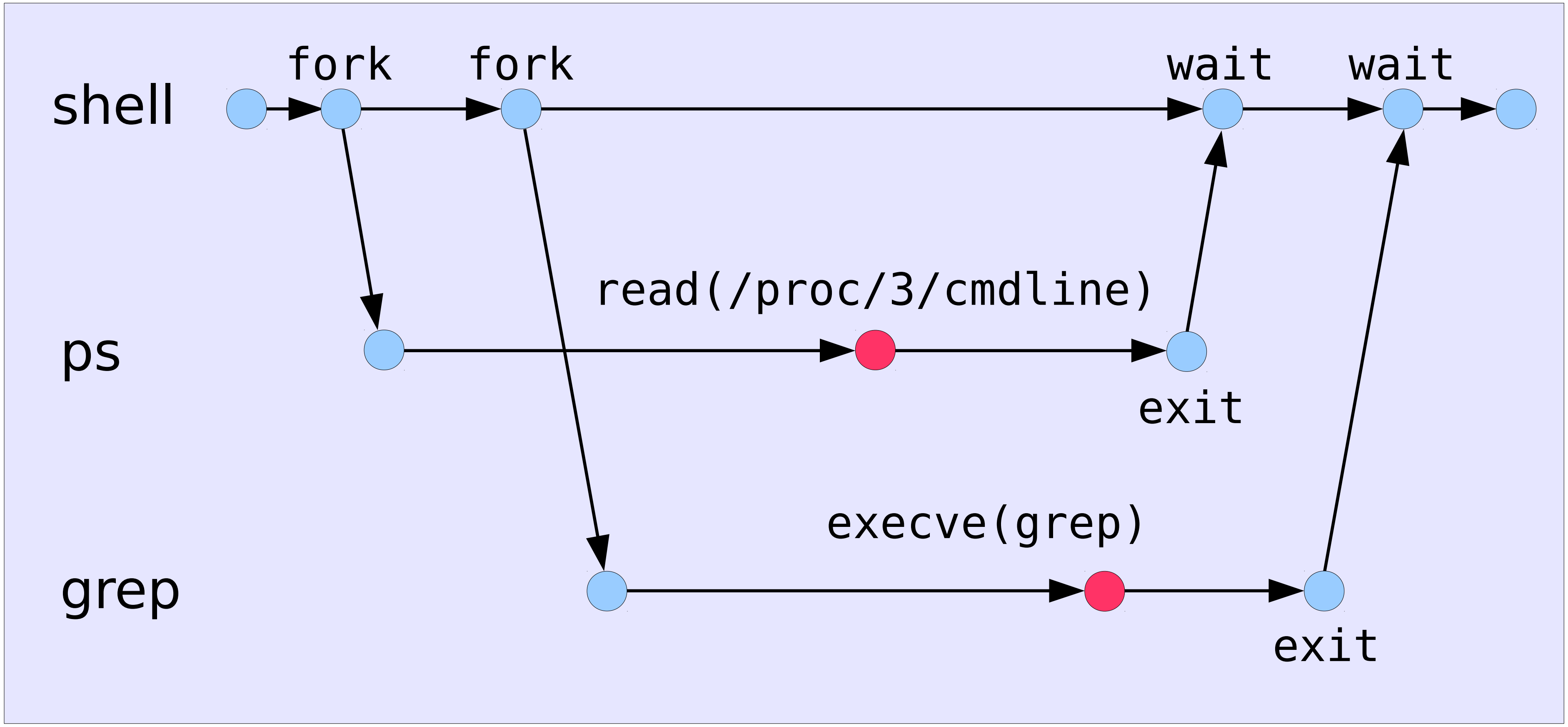


Is this race harmful or not ?

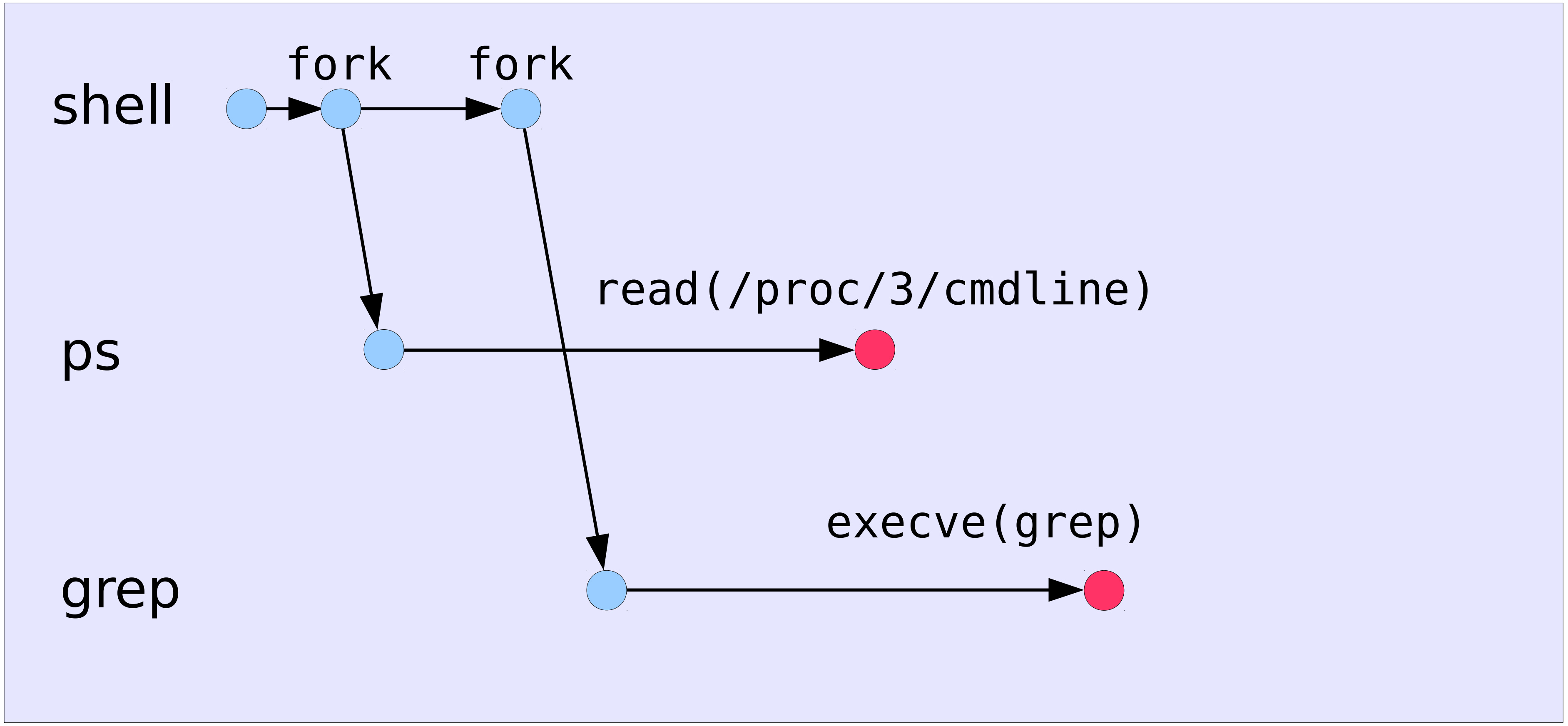
Validation


shell
fork
fork
wait
wait
ps
read(/proc/3/cmdline)
exit
grep
execve(grep)
exit

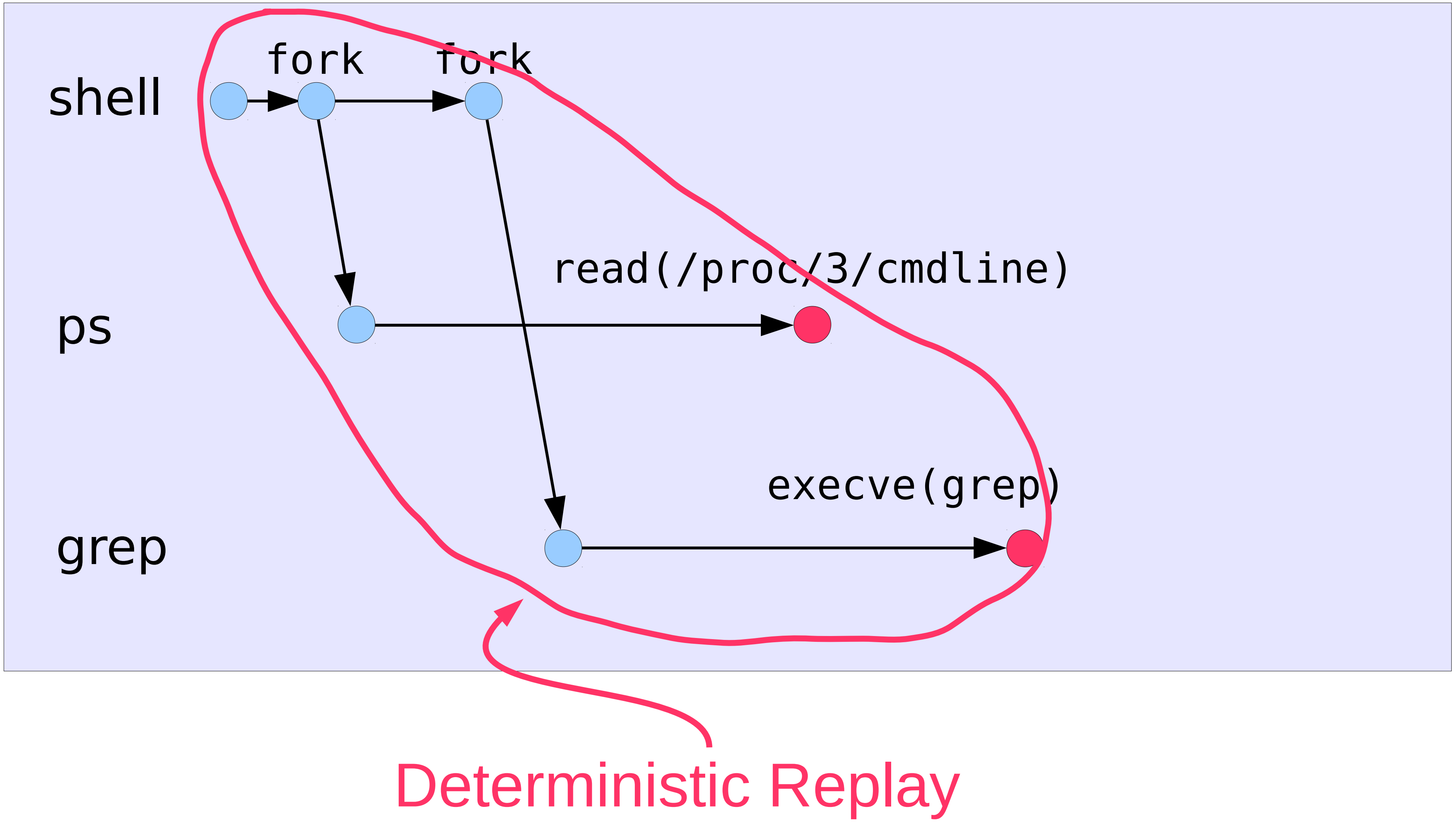
Validation


shell
fork
fork
wait
wait
ps
read(/proc/3/cmdline)
exit
grep
execve(grep)
exit

Validation

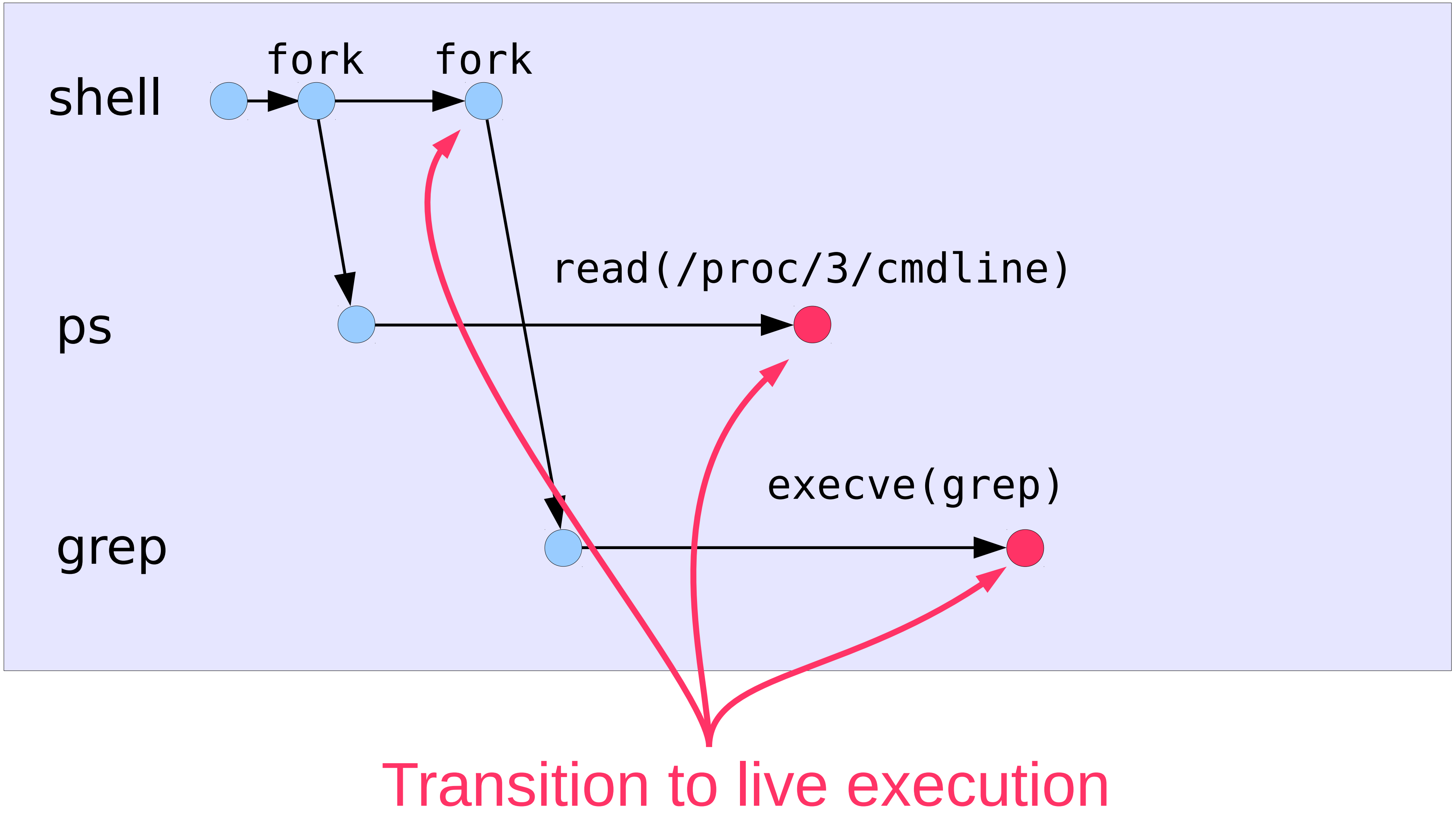


Validation



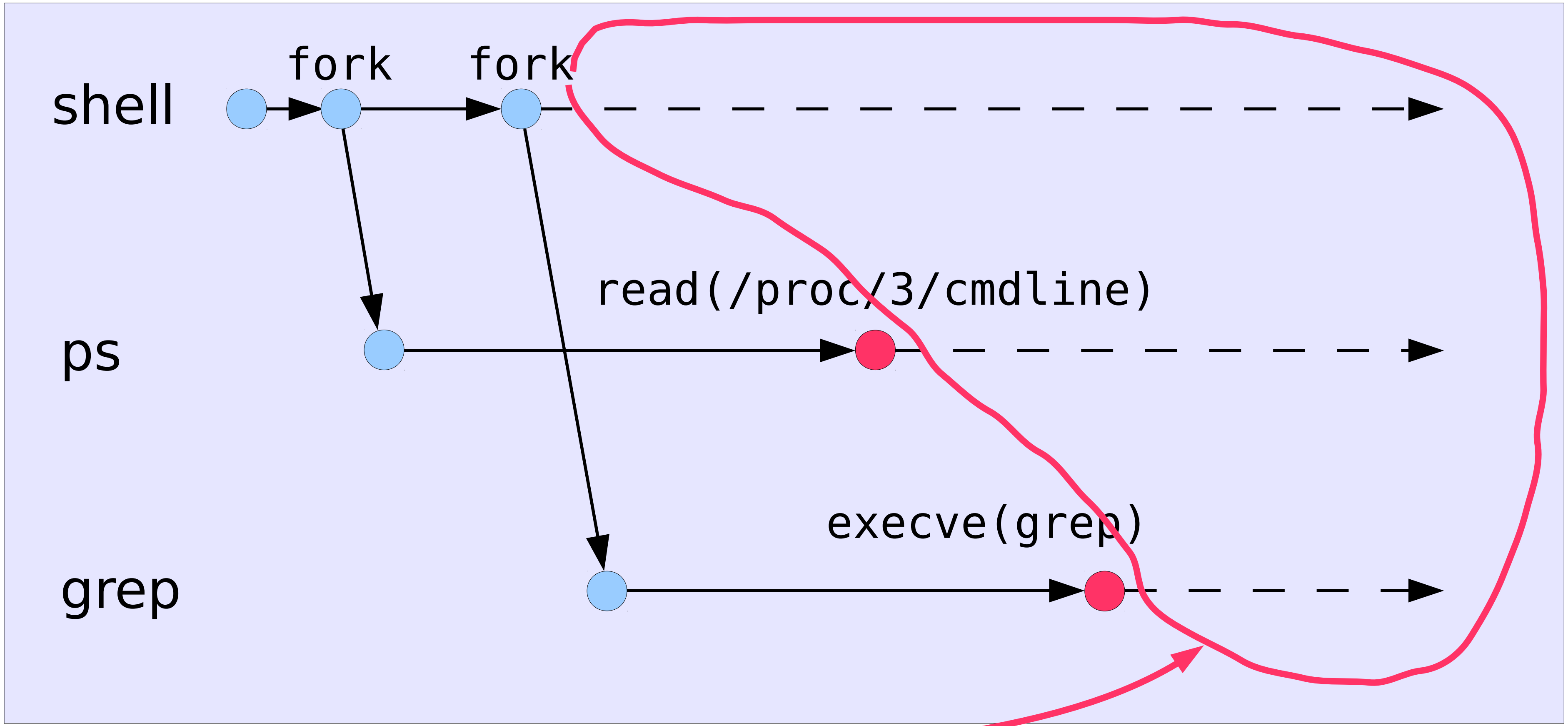
shell
ps
grep
fork
fork
read(/proc/3/cmdline)
execve(grep)
Deterministic Replay

Validation



Transition to live execution

Validation



Live execution
Watched with checkers

Results

- Detected previously known and unkown bugs
- Heavy inter-process interaction
- Validation is crucial
- Recording overhead is small

Bugs detected

Bug	Description
debian-294579	adduser: /etc/passwd corruption
debian-438076	mv: unlink target before calling rename
debian-399930	logrotate: create a file that may be observed by deamons without write permissions
redhat-54127	licq: ps | grep race causing the wrong interface to be loaded
launchpad-596064	upstart: does not wait until smdb creates a directory before spawning nmdb
launchpad-10809	bash: history file corruption
new-1	tcsh: history file corruption
new-2	updatedb: race with locate when saving the database
new-3	updatedb: concurrent updatedb may corrupt the database
new-4	abr2gbr: incorrect dependencies in the Makefile

Bugs detected

Bug	Description
debian-294579	adduser: /etc/passwd corruption
debian-438076	mv: unlink target before calling rename
debian-399930	logrotate: create a file that may be observed by deamons without write permissions
redhat-54127	licq: ps | grep race causing the wrong interface to be loaded
launchpad-596064	upstart: does not wait until smdb creates a directory before spawning nmdb
launchpad-10809	bash: history file corruption
new-1	tcsh: history file corruption
new-2	updatedb: race with locate when saving the database
new-3	updatedb: concurrent updatedb may corrupt the database
new-4	abr2gbr: incorrect dependencies in the Makefile

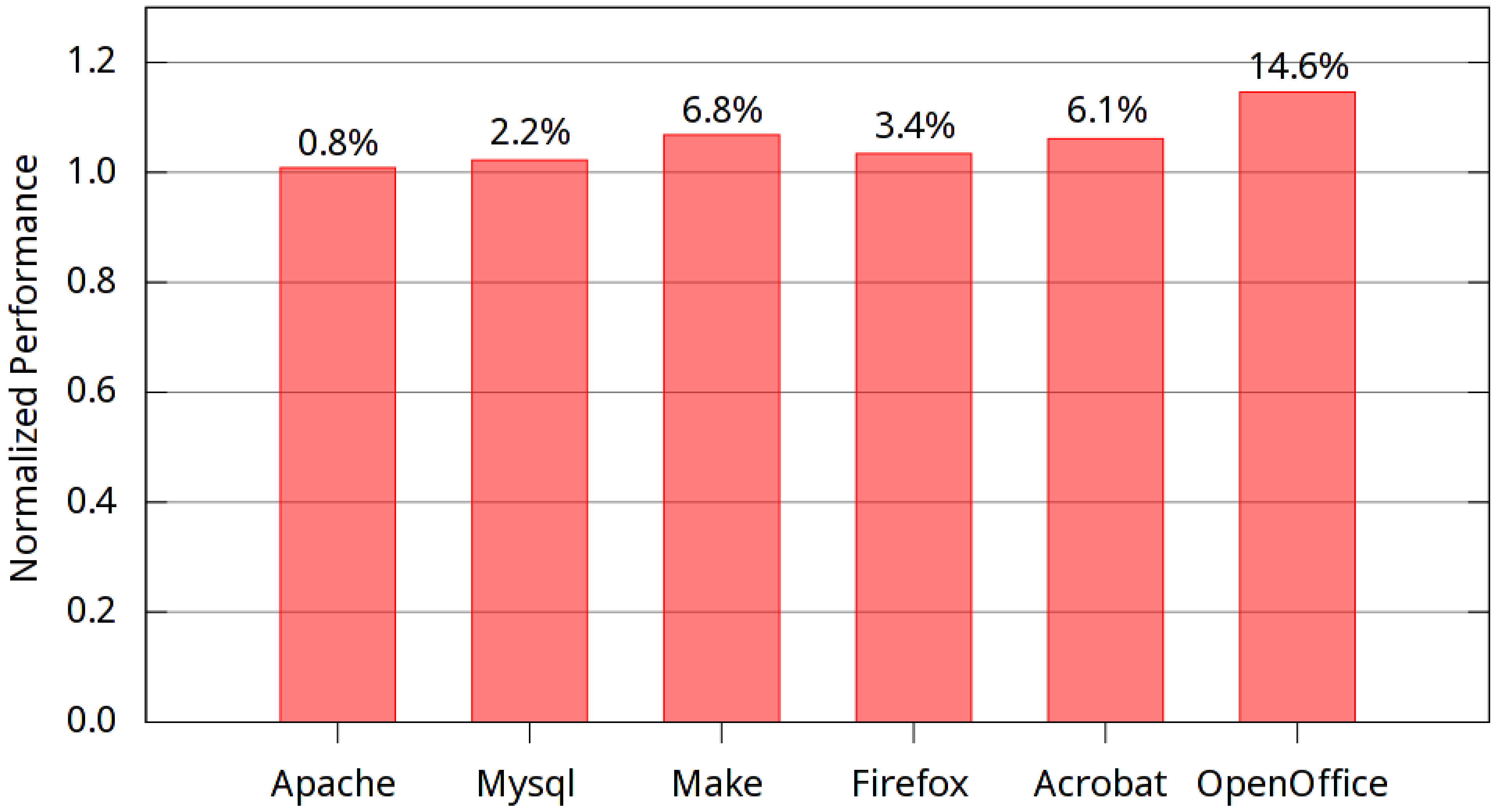
Detection

Bug	Processes	Syscalls	Resources
debian-294579	19	5275	658
debian-438076	21	1688	213
debian-399930	10	1536	279
redhat-54127	14	1298	229
launchpad-596064	34	5564	722
launchpad-10809	13	1890	205
new-1	12	2569	201
new-2	47	2621	467
new-3	30	4361	2981
new-4	19	4672	716

Validation

Bug	Detected	Harmful	Checker
debian-294579	4231	42	Custom
debian-438076	50	4	Default
debian-399930	17	4	Default
redhat-54127	35	4	Custom
launchpad-596064	272	2	Default
launchpad-10809	143	10	Custom
new-1	137	14	Custom
new-2	82	42	Default
new-3	17	4	Default
new-4	8	1	Default

Recording



Normalized Performance
0.0
0.2
0.4
0.6
0.8
1.0
1.2
0.8%
2.2%
6.8%
3.4%
6.1%
14.6%
Apache
Mysql
Make
Firefox
Acrobat
OpenOffice

Conclusion

- Racepro: the **first** generic process race detector
  - Record applications in production systems
  - Model system calls with load/store micro-ops
  - Validate by checking uncontrolled execution

- Detected previously known and **unknown** races
- Low recording overhead

For More Information

SOFTWARE SYSTEMS LABORATORY

systems.cs.columbia.edu

Scribe

github.com/nviennot/linux-2.6-scribe

Resources

Object	Description
inode	File, Directory, Socket, Pipe, TTY, Device
file	File handle of an opened file
file-table	Process file table
mmap	Process memory map
cred	Process credentials
global	System-wide properties (hostname, ...)
pid	Process ID
ppid	Parent process ID

Checkers

- Crash detection
- Application Hanging
- Check for error messages in log files
- Return value of application
- Linearized run (EuroSys11)