Light-Weight Contexts

An OS Abstraction for Safety and Performance

James Litton

Anjo Vahldiek-Oberwagner

Eslam Elnikety

Deepak Garg

Bobby Bhattacharjee

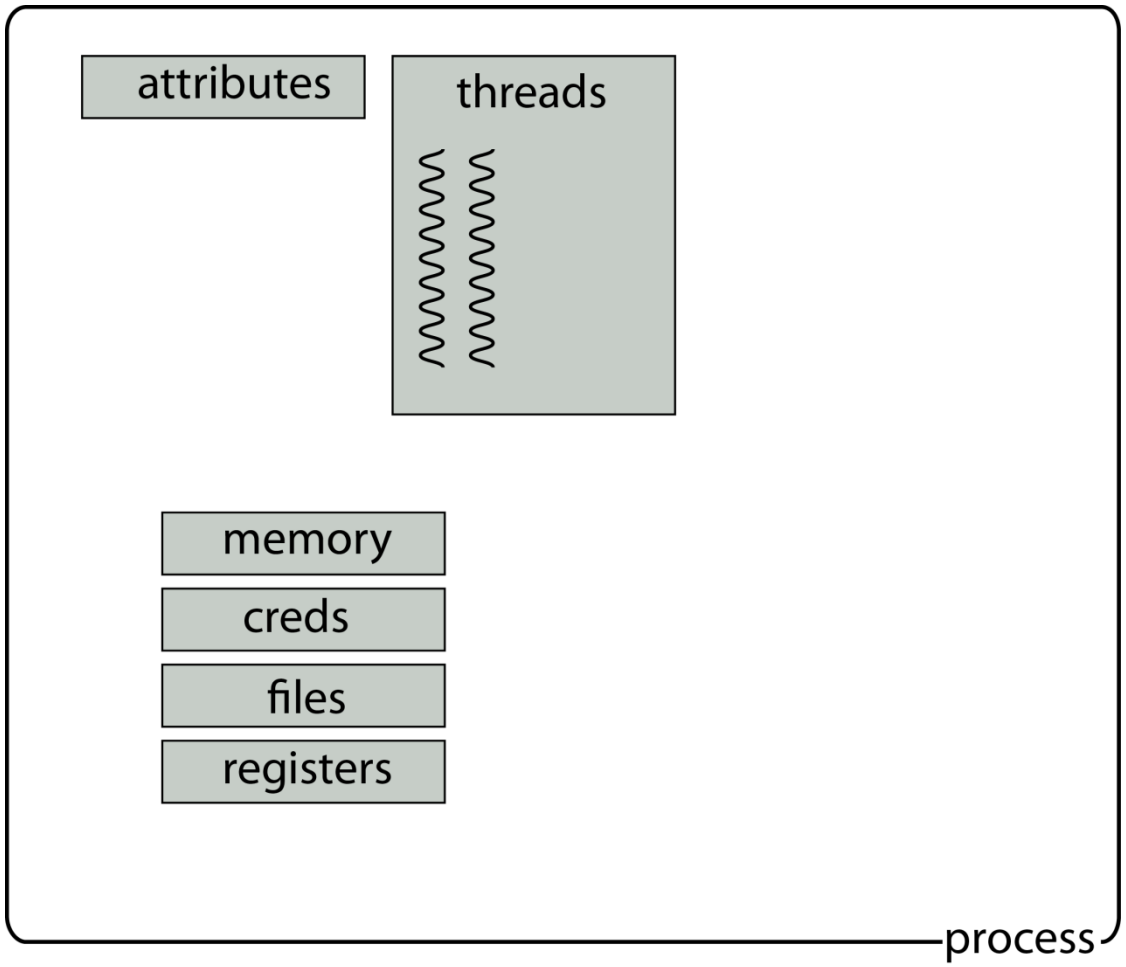
Peter Druschel



Max
Planck
Institute
for
Software Systems

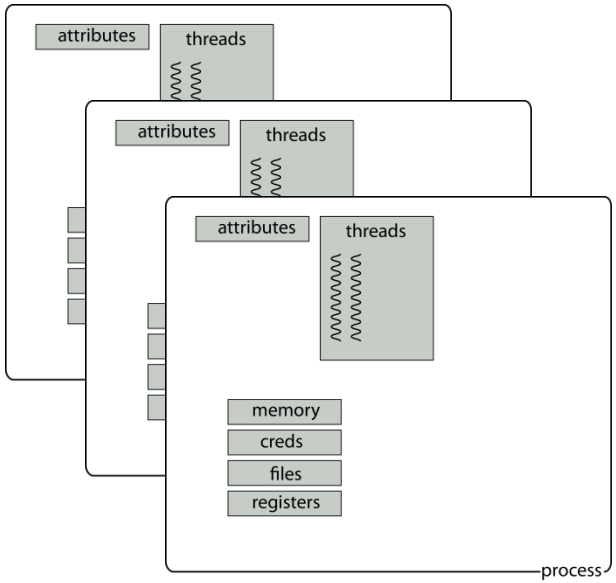


Process



Unit of isolation, privilege separation, and program state

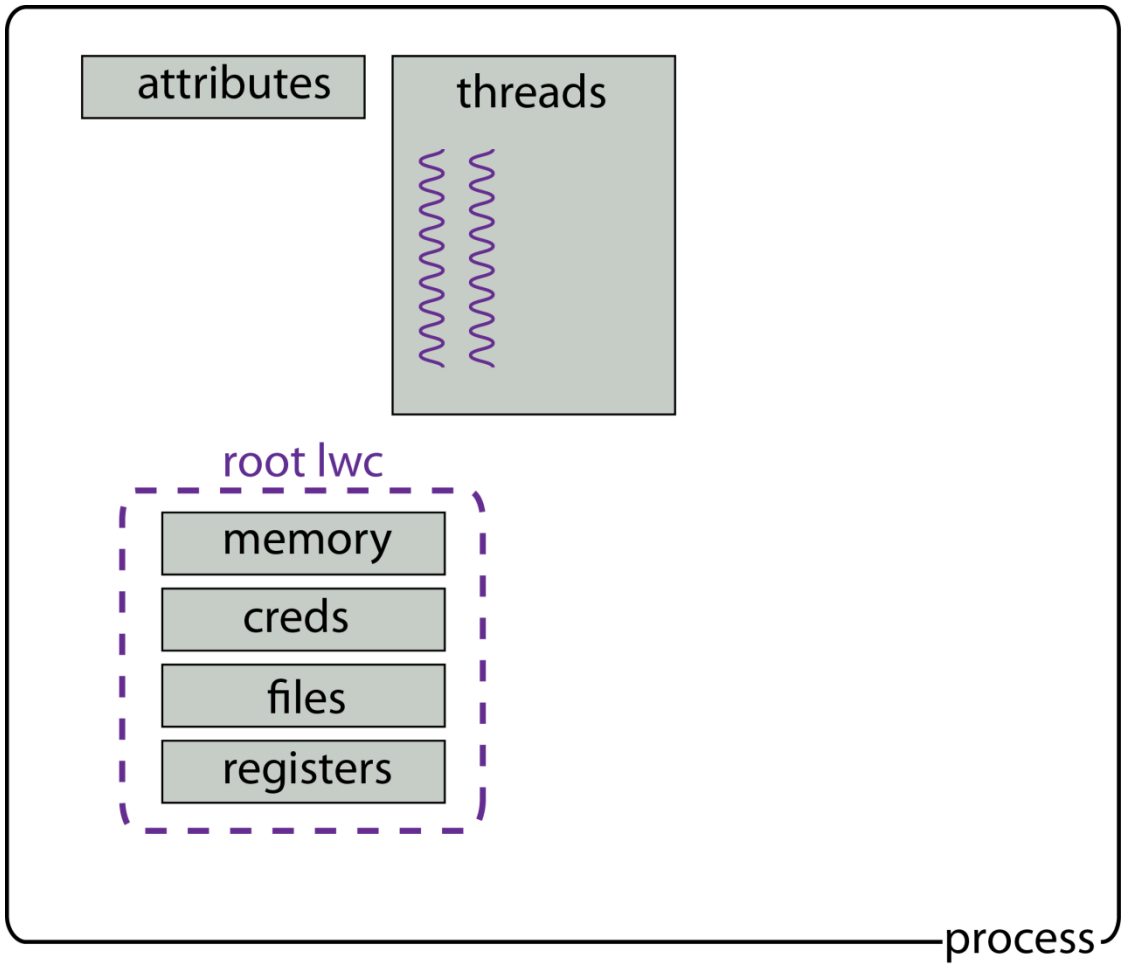
Fork



Used to achieve distinct isolation, privilege separation, and program state

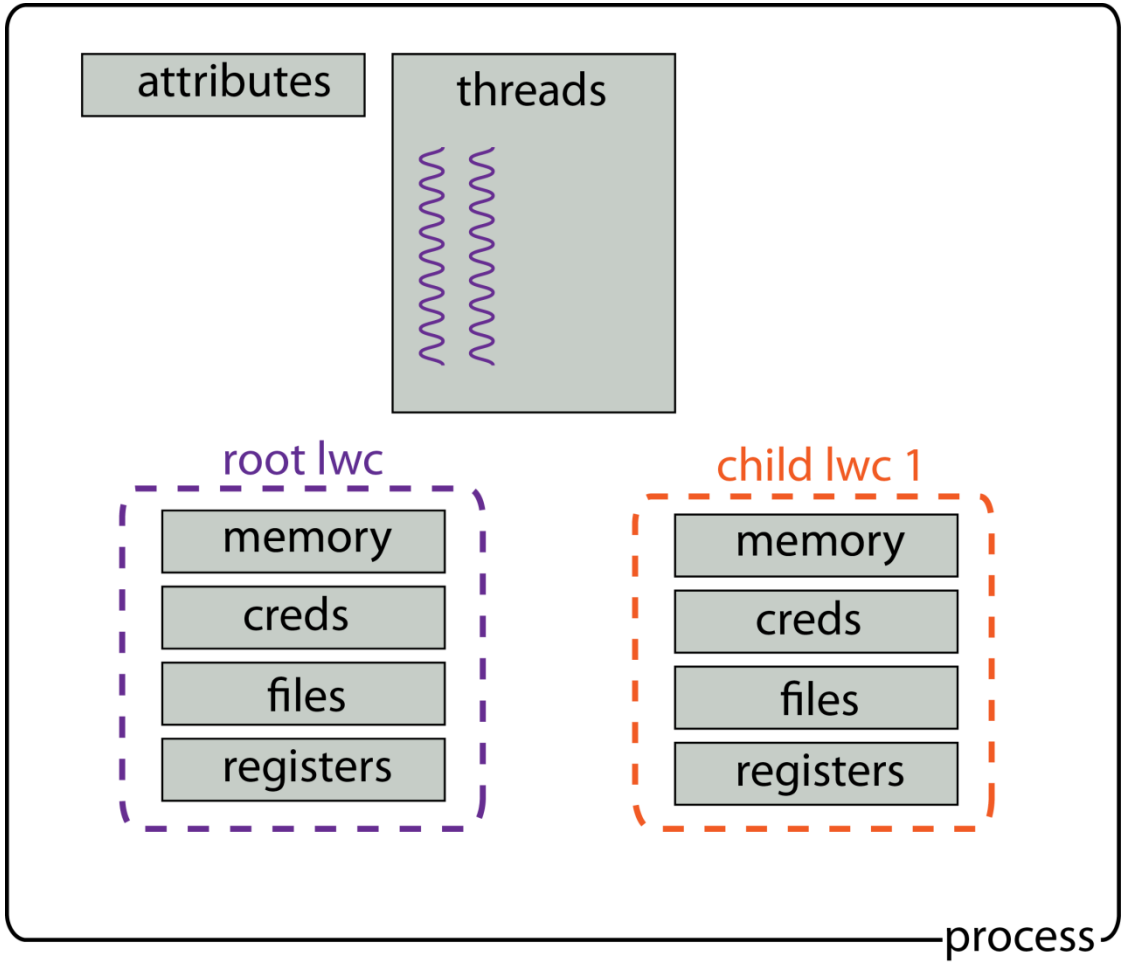
Imposes scheduling and coordination overhead

Light-Weight Context

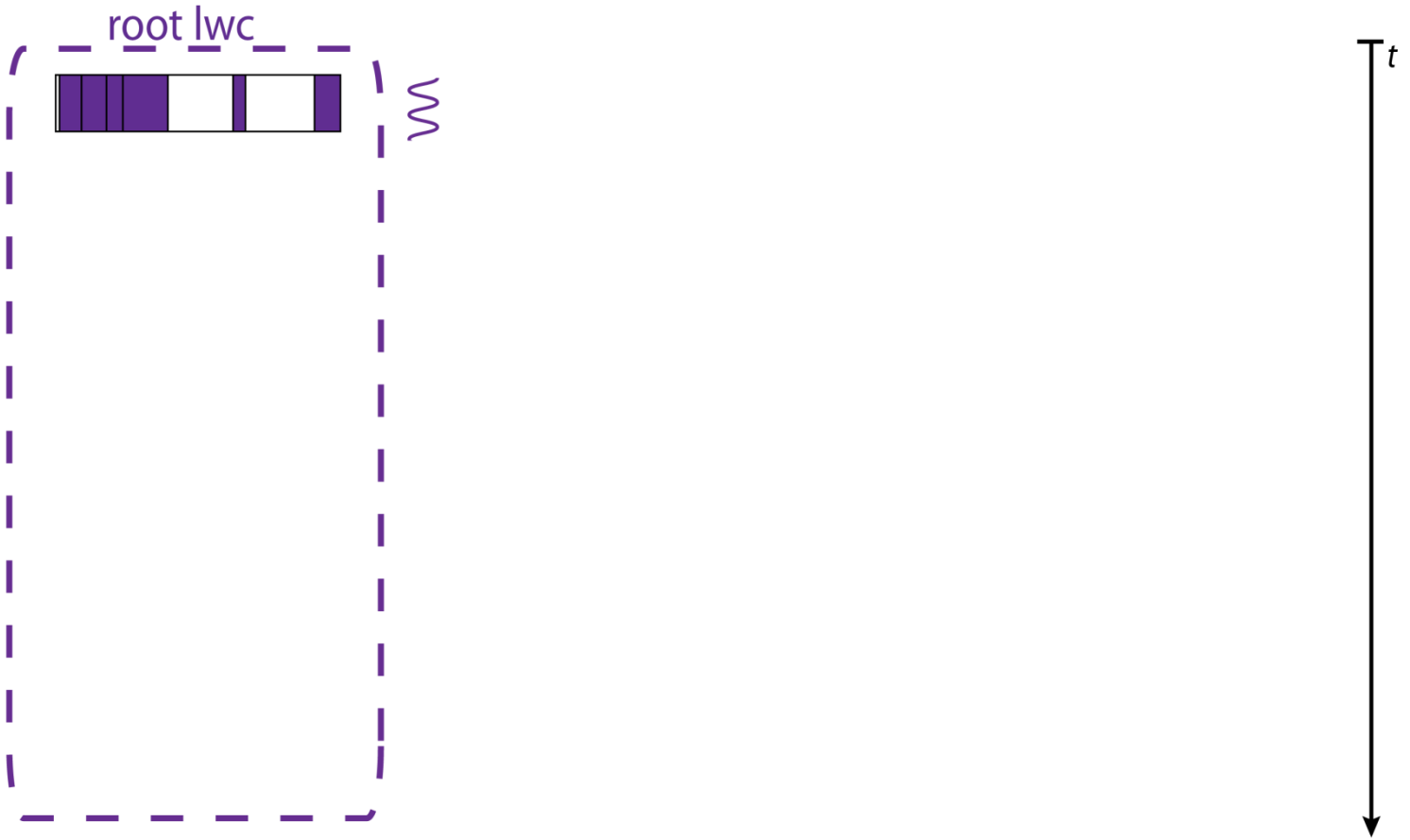


Intra-process isolation of system resources

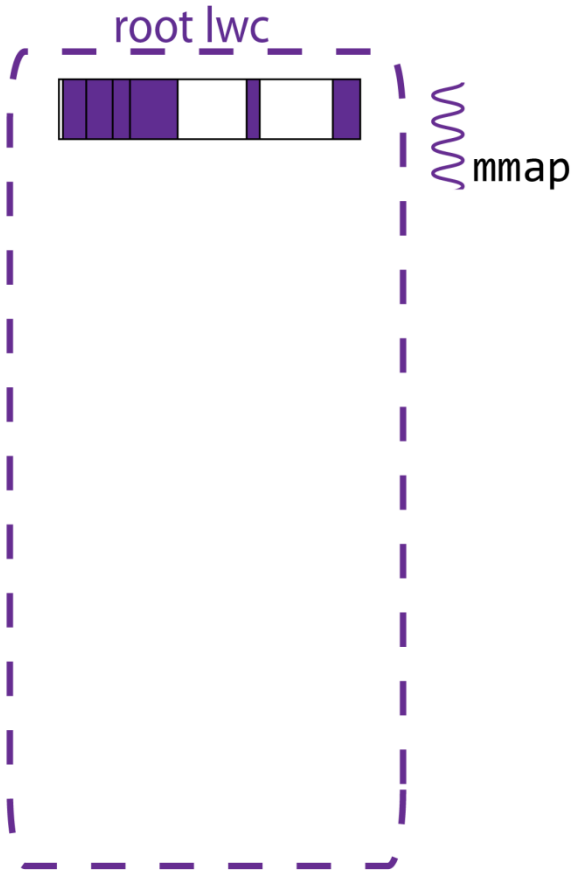
Light-Weight Context

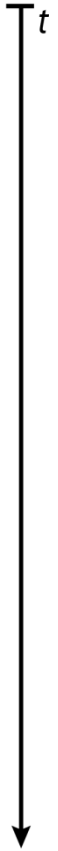


Creating lwCs

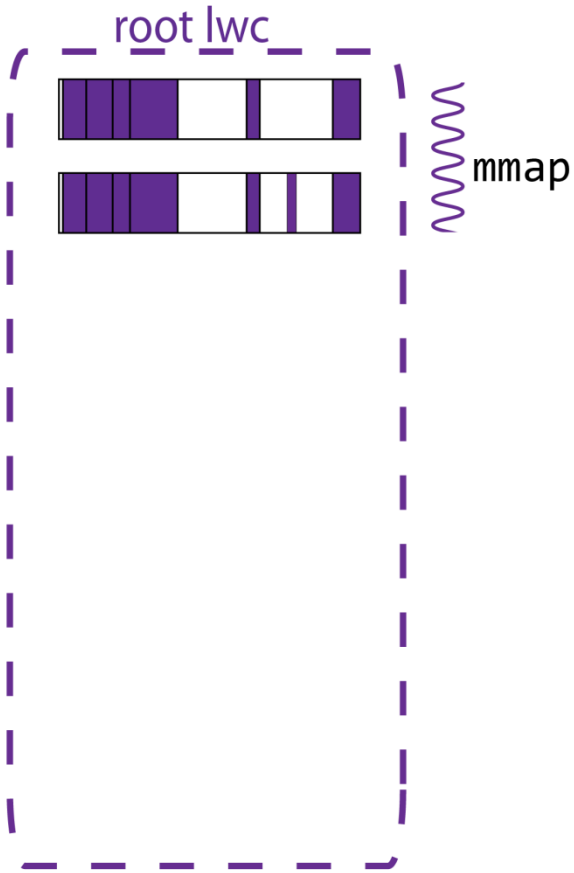


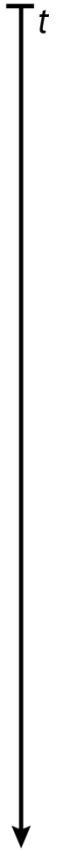
lwCreate





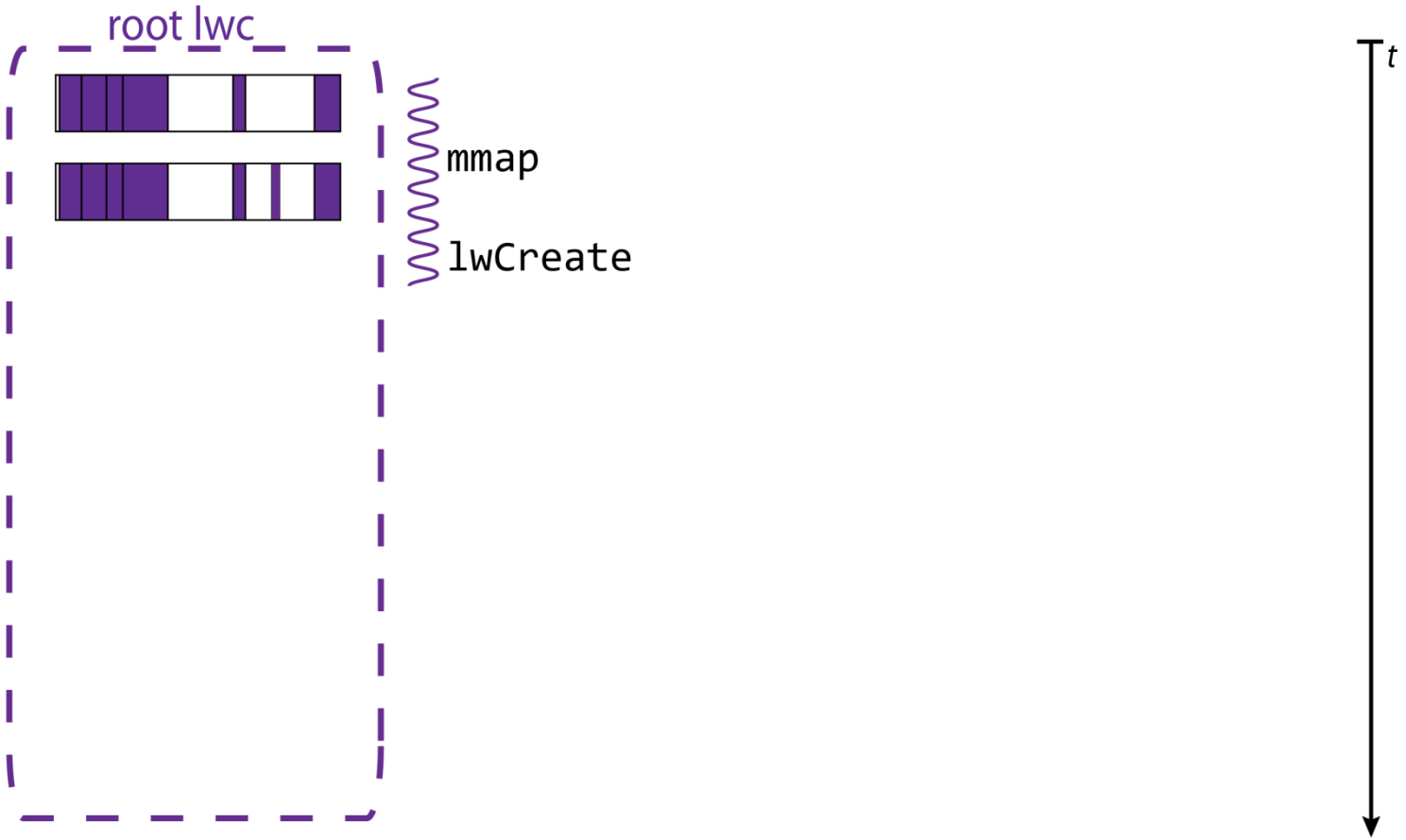
lwCreate





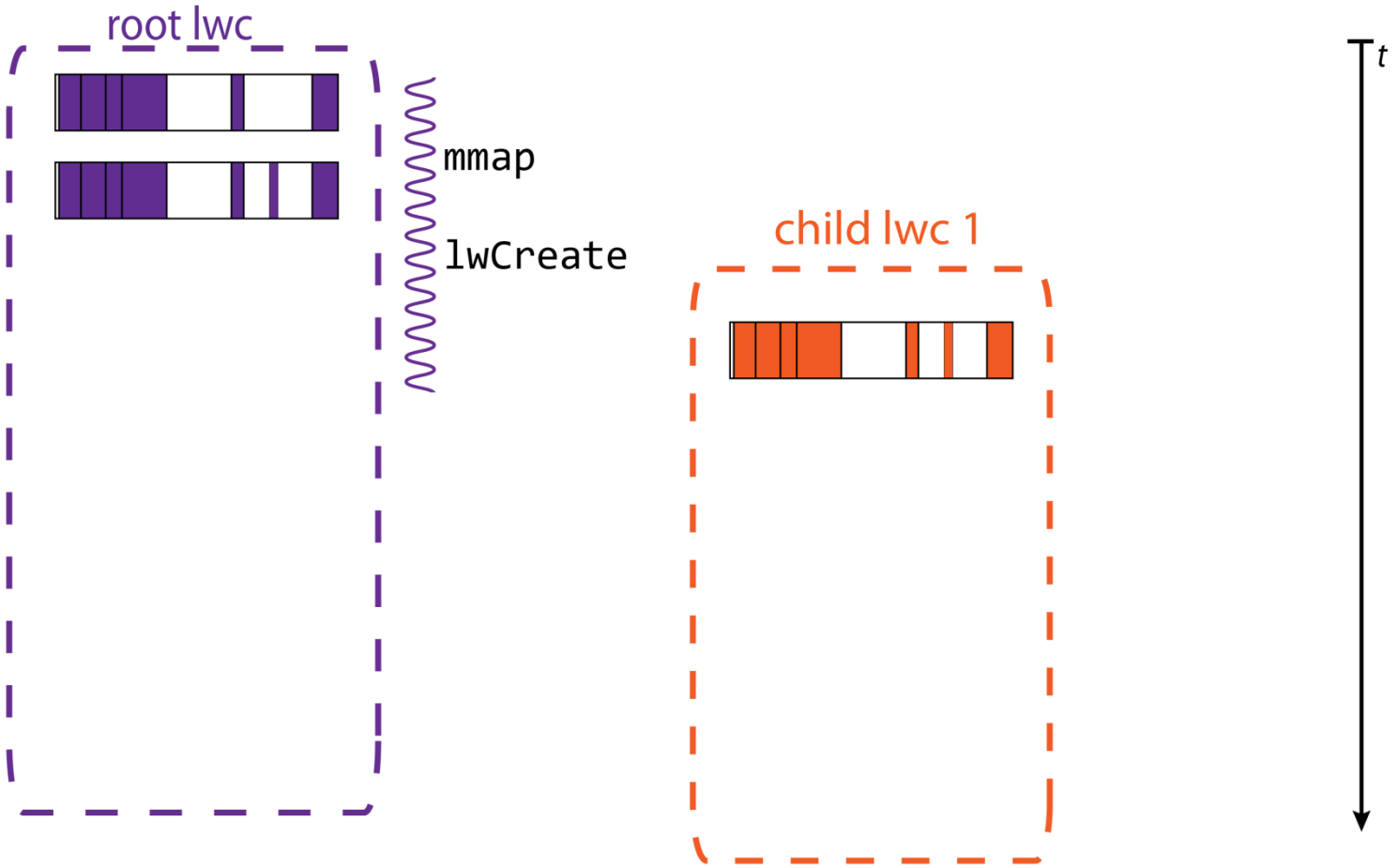
lwCreate

```
new,caller,args ← lwCreate(spec, options)
```



lwCreate

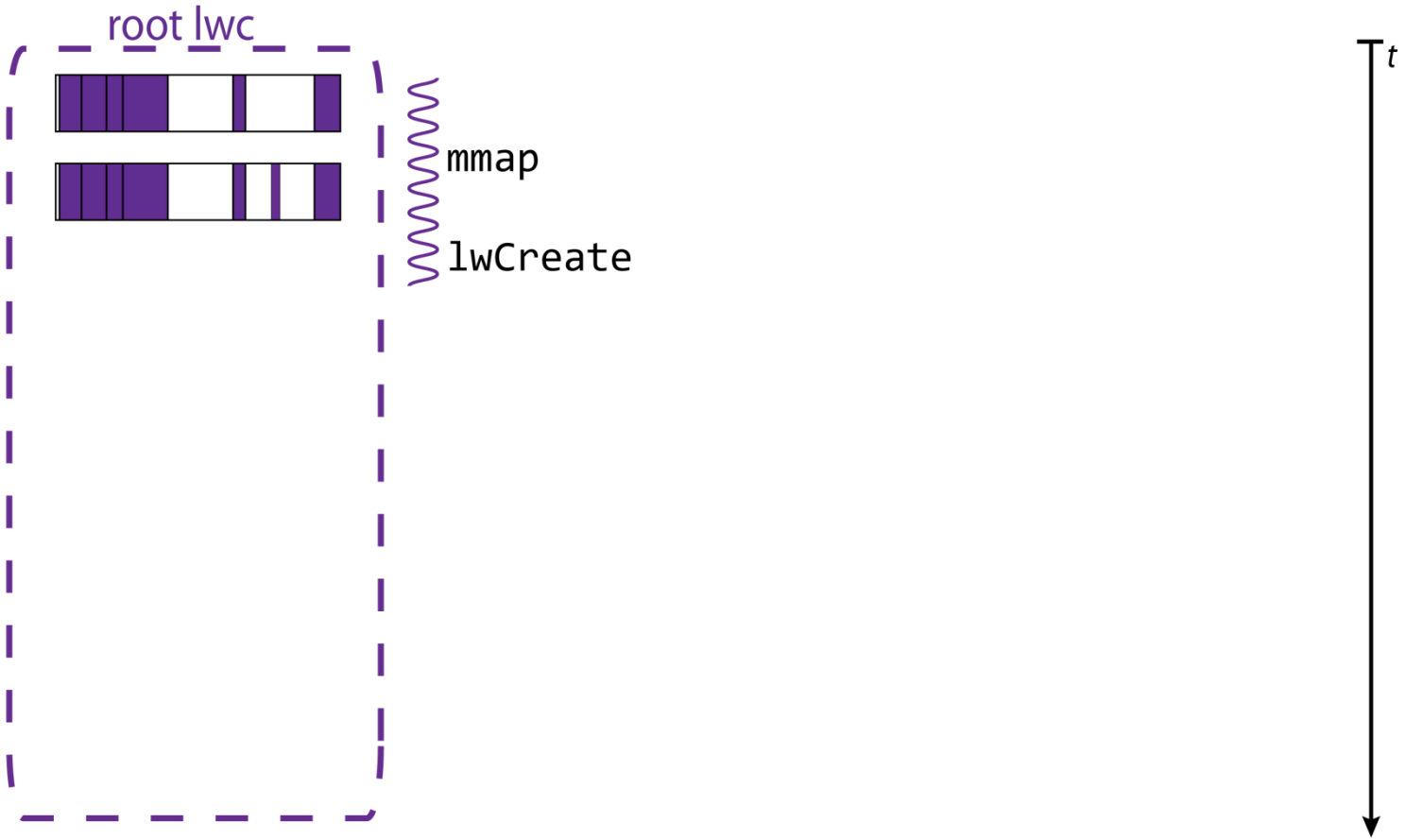
```
new,caller,args ← lwCreate(spec, options)
```



The child is a copy (via COW) of the parent by default

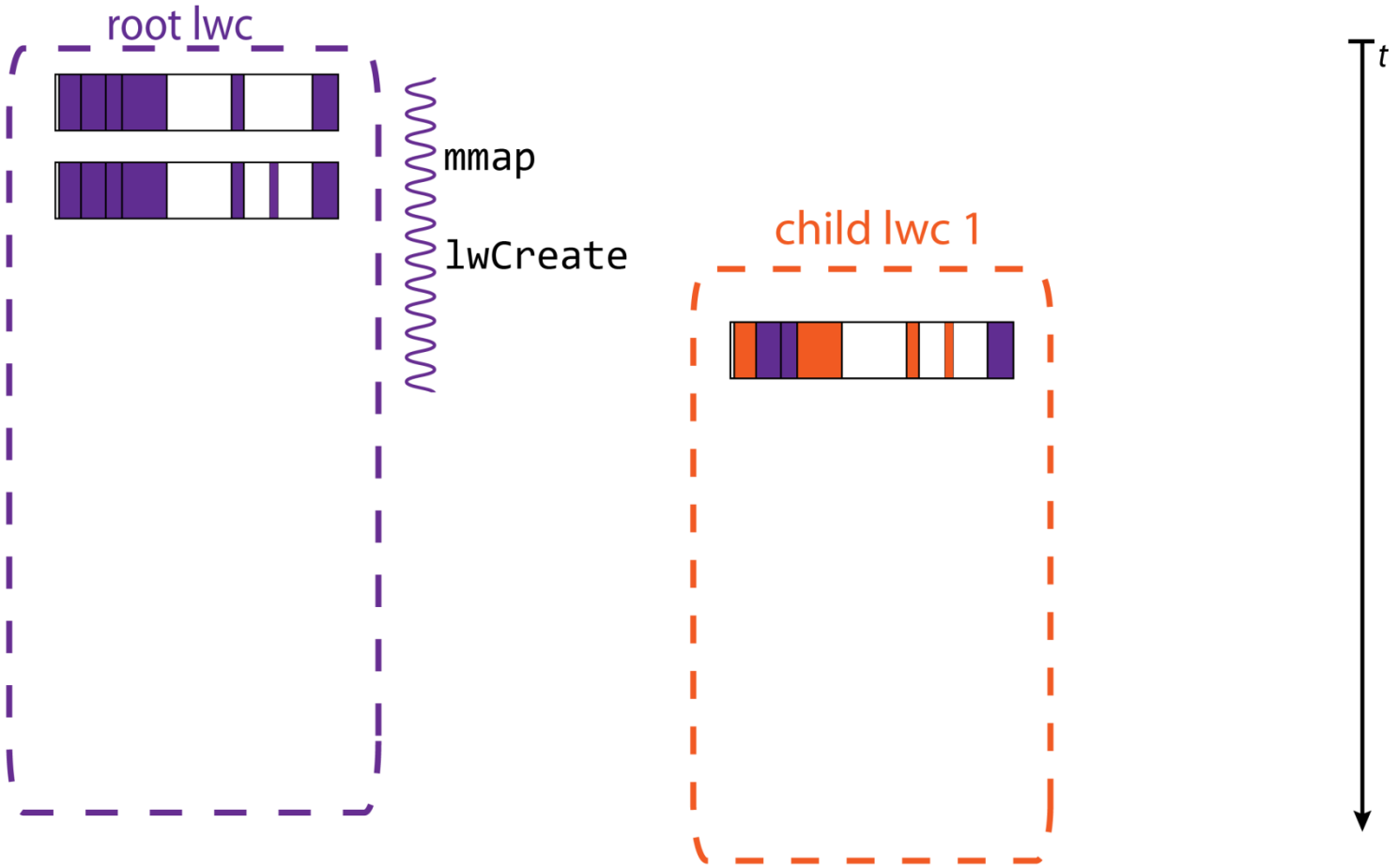
lwCreate

```
new,caller,args ← lwCreate(spec, options)
```



lwCreate

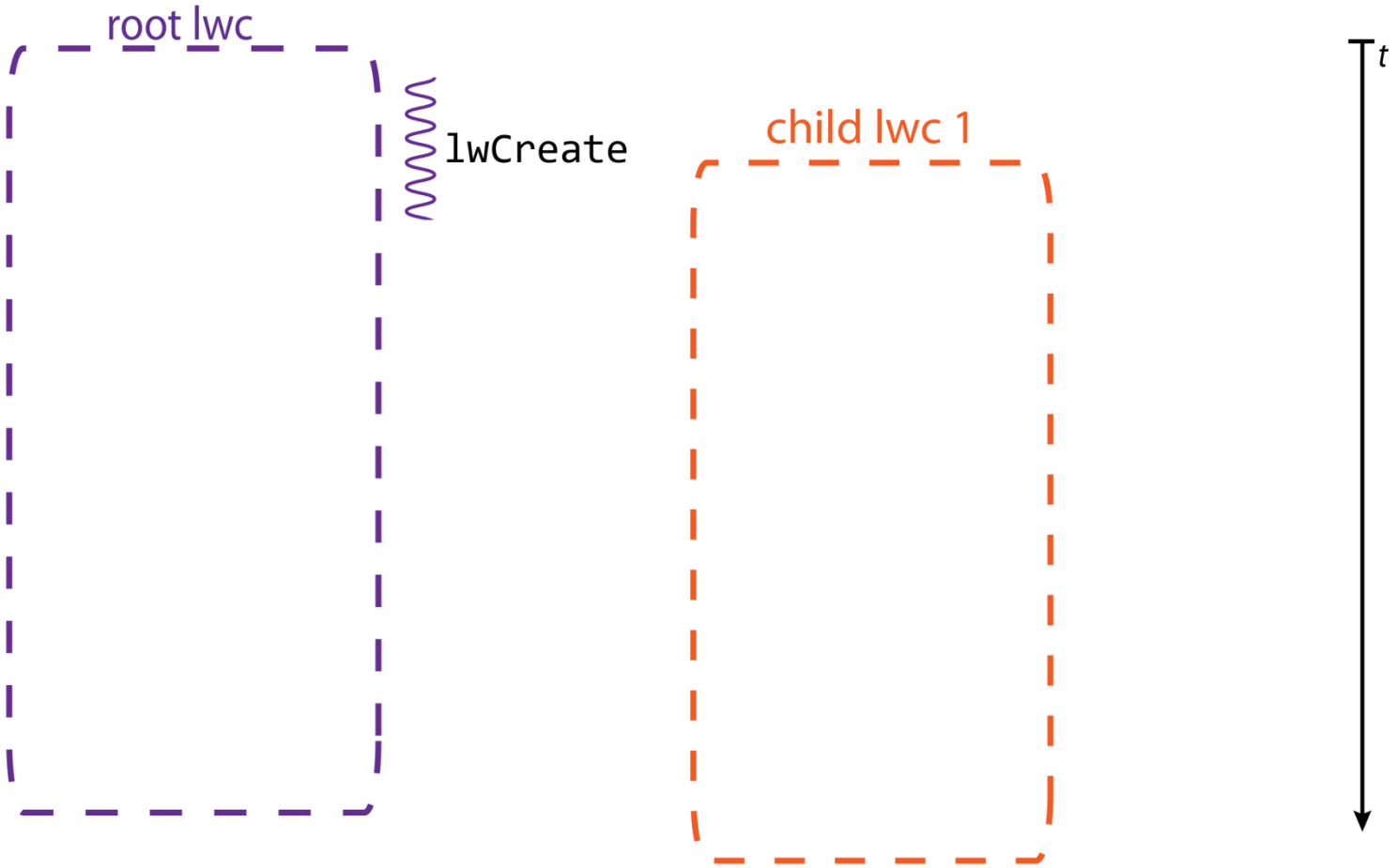
```
new,caller,args ← lwCreate(spec, options)
```



The child may share or omit parent resources

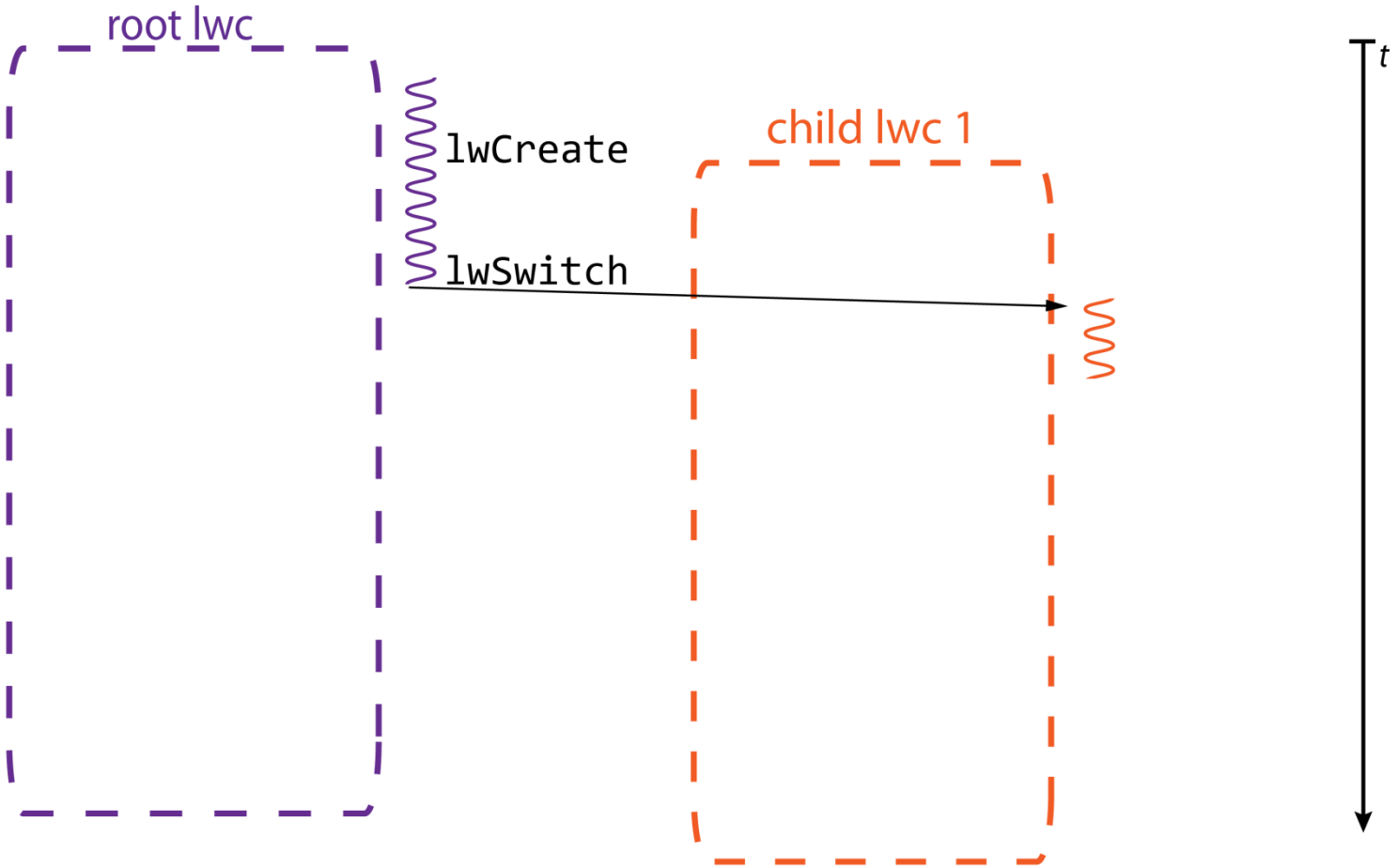
lwSwitch

```
caller,args ← lwSwitch(target, args)
```



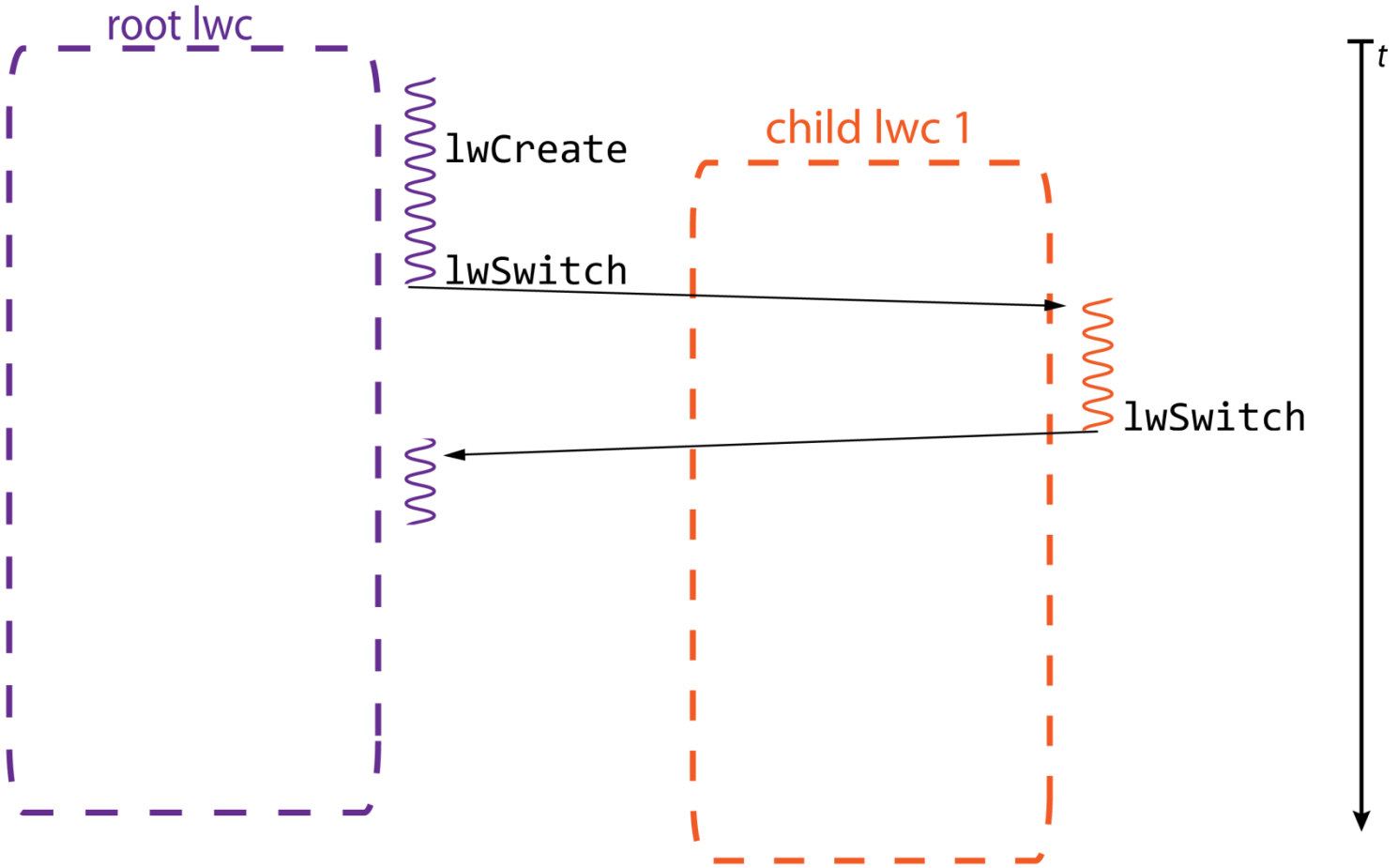
lwSwitch

```
caller,args ← lwSwitch(target, args)
```



lwSwitch

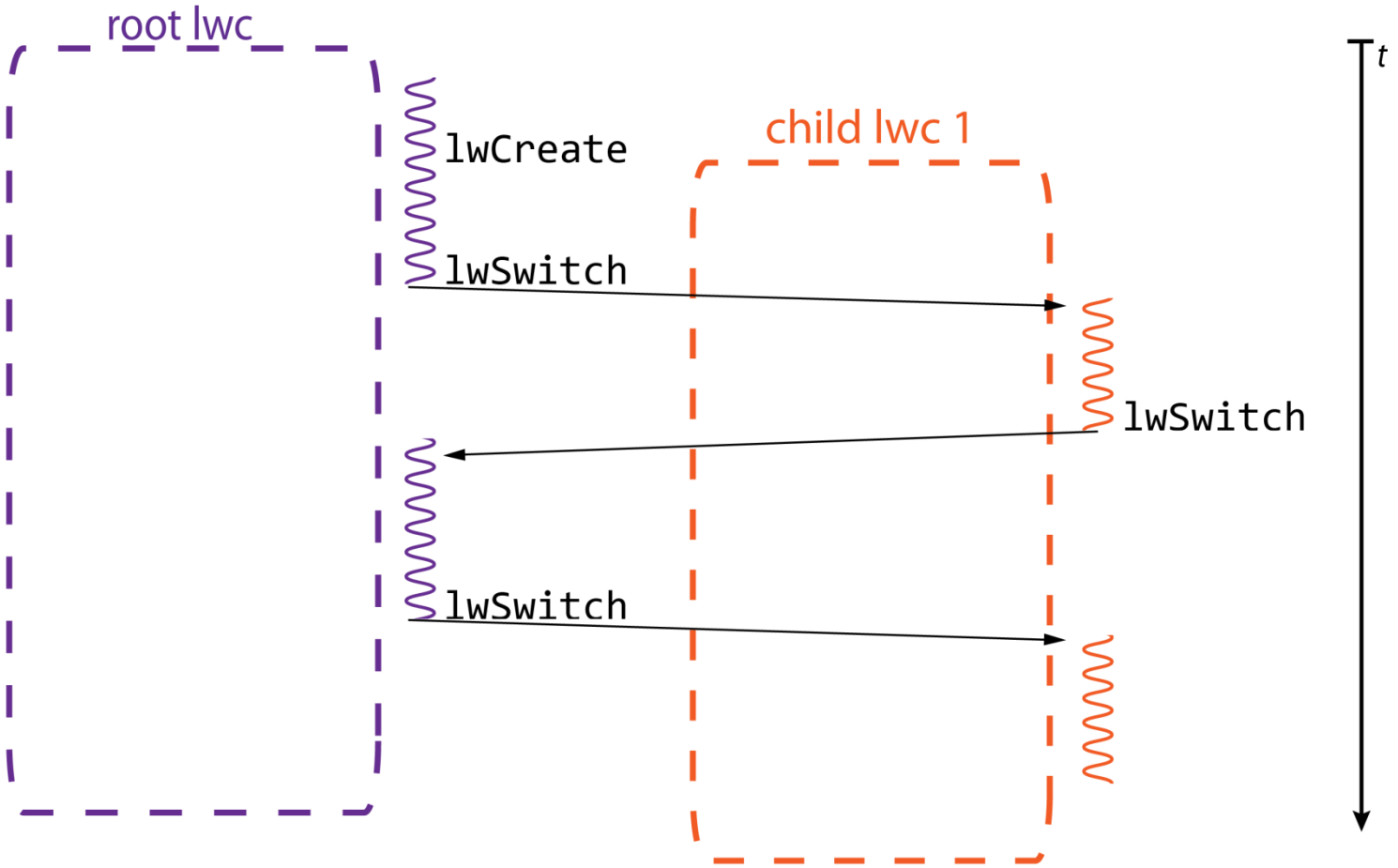
```
caller,args ← lwSwitch(target, args)
```



lwSwitch provides coroutine semantics

lwSwitch

```
caller,args ← lwSwitch(target, args)
```



lwSwitch provides coroutine semantics

API Summary

Create lwC		
new,caller,args	←	lwCreate(spec, options)
Switch to lwC		
caller,args	←	lwSwitch(target, args)
Resource Access		
status	←	lwRestrict(lwc, spec)
status	←	lwOverlay(lwc, spec)
status	←	lwSyscall(target, mask, syscall, args)

Common uses

- Session isolation

Common uses

- Session isolation
- Sensitive data isolation

Common uses

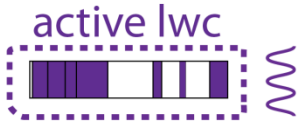
- Session isolation
- Sensitive data isolation
- Snapshot and rollback

Common uses

- Session isolation
- Sensitive data isolation
- Snapshot and rollback
- Reference monitoring

Snapshots

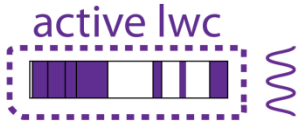
```
1:  function SNAPSHOT()
2:      new,caller,arg = lwCreate(. . . )
3:      if caller = -1 then
4:          return new
5:      else
6:          close(caller)
7:          return snapshot()
8:  function ROLLBACK(snap)
9:      lwSwitch(snap, 0)
10: function MAIN()
11:     ...
12:     snap = snapshot()
13:     ...
14:     rollback(snap)
```





Snapshots

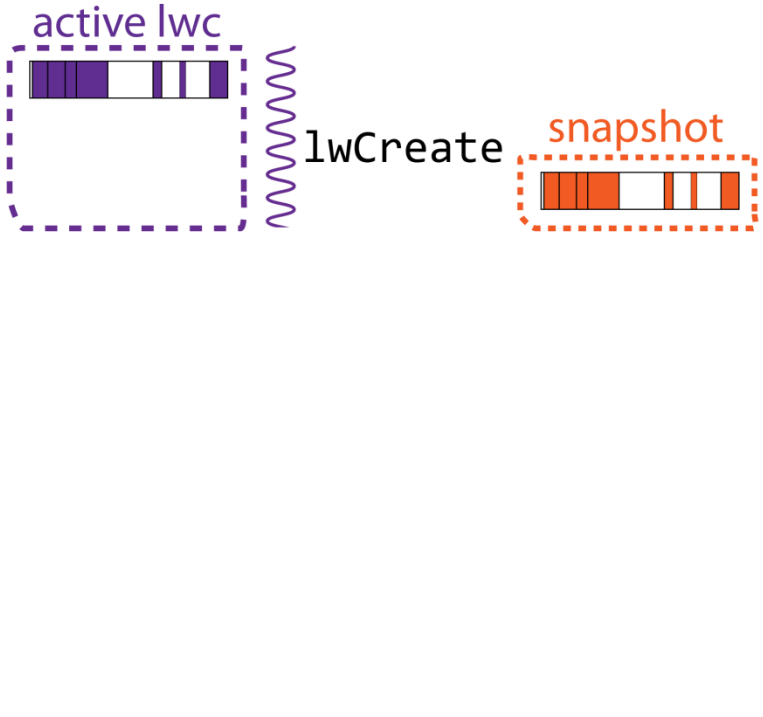
```
1:  function SNAPSHOT()
2:      new,caller,arg = lwCreate(...)
3:      if caller = -1 then
4:          return new
5:      else
6:          close(caller)
7:          return snapshot()
8:  function ROLLBACK(snap)
9:      lwSwitch(snap, 0)
10: function MAIN()
11:     ...
12:     snap = snapshot()
13:     ...
14:     rollback(snap)
```





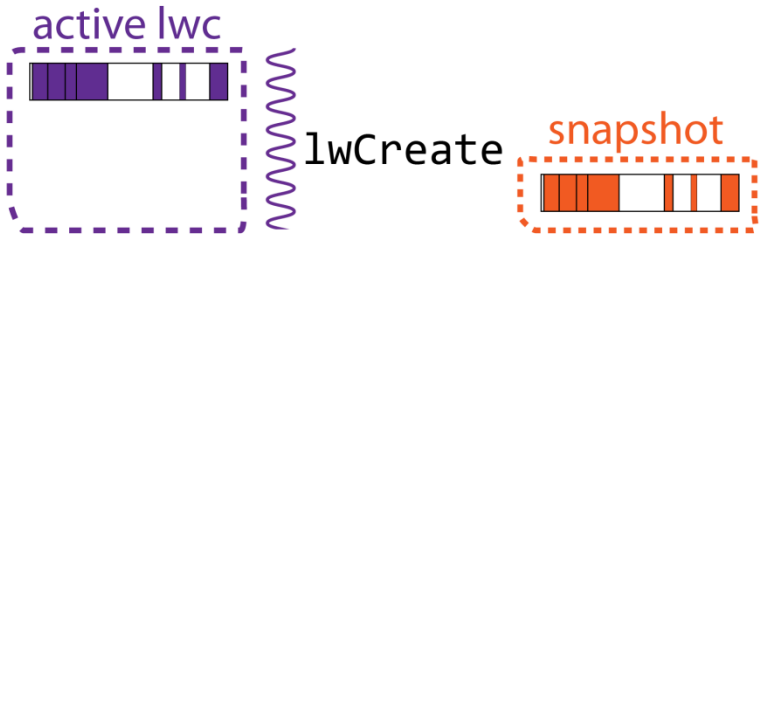
Snapshots

```
1:  function SNAPSHOT()
2:      new,caller,arg = lwCreate(...)
3:      if caller = -1 then
4:          return new
5:      else
6:          close(caller)
7:          return snapshot()
8:  function ROLLBACK(snap)
9:      lwSwitch(snap, 0)
10: function MAIN()
11:     ...
12:     snap = snapshot()
13:     ...
14:     rollback(snap)
```



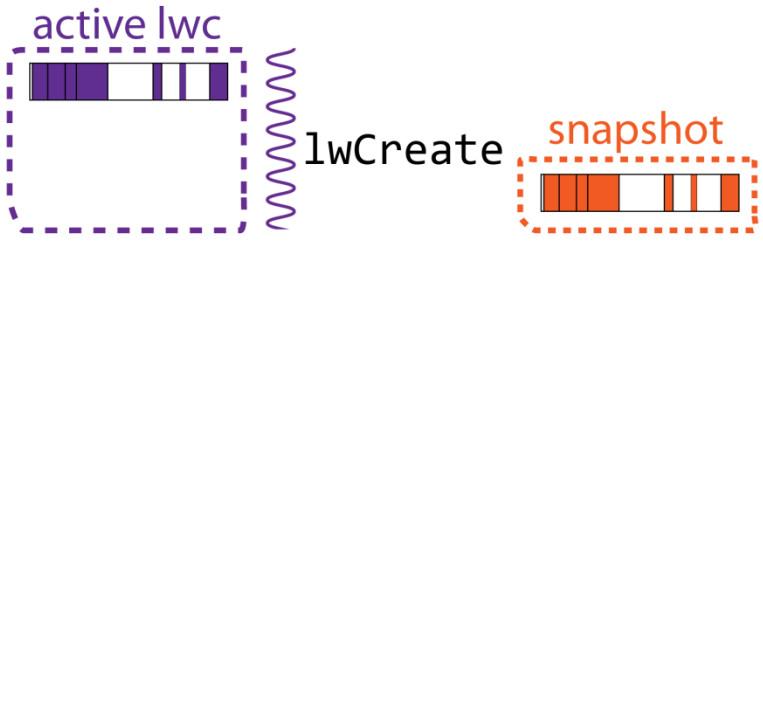
Snapshots

```
1:  function SNAPSHOT()
2:      new,caller,arg = lwCreate(...)
3:      if caller = -1 then
4:          return new
5:      else
6:          close(caller)
7:          return snapshot()
8:  function ROLLBACK(snap)
9:      lwSwitch(snap, 0)
10: function MAIN()
11:     ...
12:     snap = snapshot()
13:     ...
14:     rollback(snap)
```



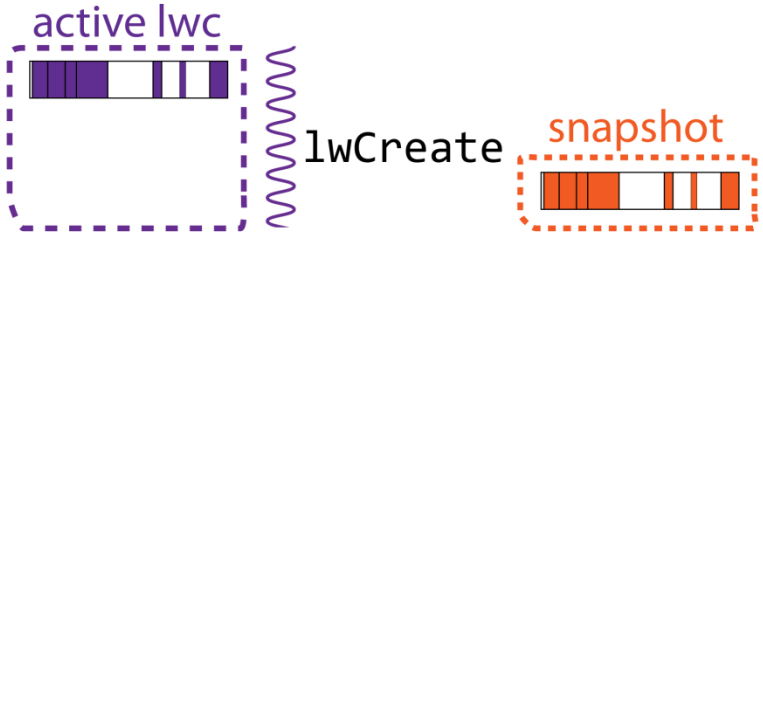
Snapshots

```
1:  function SNAPSHOT()
2:      new,caller,arg = lwCreate(...)
3:      if caller = -1 then
4:          return new
5:      else
6:          close(caller)
7:          return snapshot()
8:  function ROLLBACK(snap)
9:      lwSwitch(snap, 0)
10: function MAIN()
11:     ...
12:     snap = snapshot()
13:     ...
14:     rollback(snap)
```



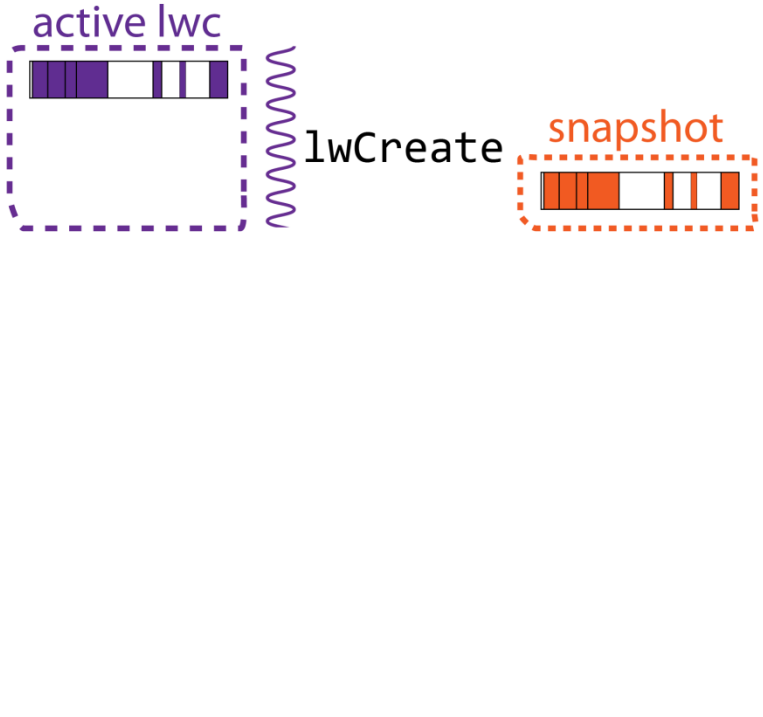
Snapshots

```
1:  function SNAPSHOT()
2:      new,caller,arg = lwCreate(...)
3:      if caller = -1 then
4:          return new
5:      else
6:          close(caller)
7:          return snapshot()
8:  function ROLLBACK(snap)
9:      lwSwitch(snap, 0)
10: function MAIN()
11:     ...
12:     snap = snapshot()
13:     ...
14:     rollback(snap)
```



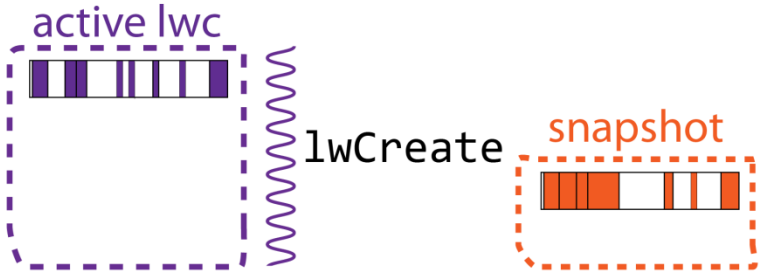
Snapshots

```
1:  function SNAPSHOT()
2:      new,caller,arg = lwCreate(...)
3:      if caller = -1 then
4:          return new
5:      else
6:          close(caller)
7:          return snapshot()
8:  function ROLLBACK(snap)
9:      lwSwitch(snap, 0)
10: function MAIN()
11:     ...
12:     snap = snapshot()
13:     ...
14:     rollback(snap)
```



Snapshots

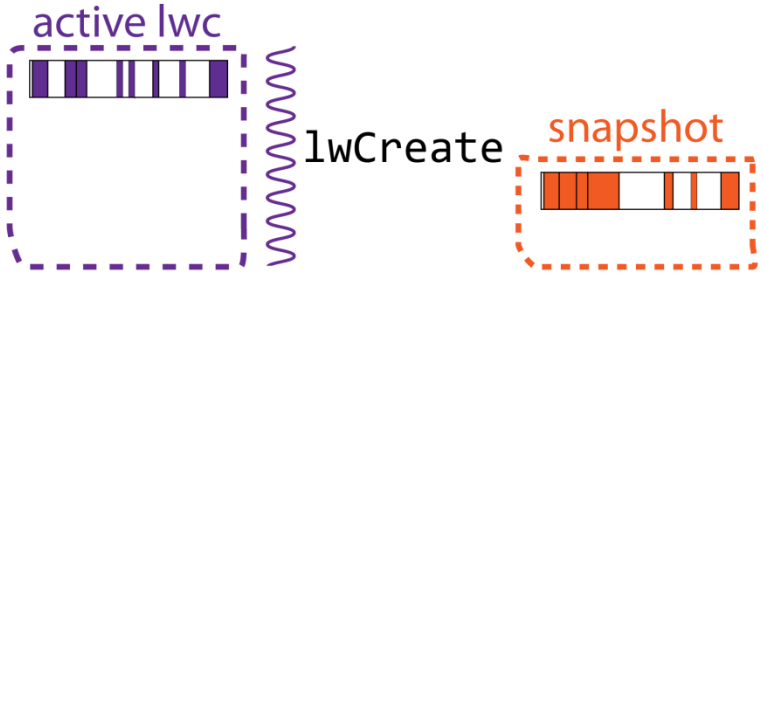
```
1:  function SNAPSHOT()
2:      new,caller,arg = lwCreate(...)
3:      if caller = -1 then
4:          return new
5:      else
6:          close(caller)
7:          return snapshot()
8:  function ROLLBACK(snap)
9:      lwSwitch(snap, 0)
10: function MAIN()
11:     ...
12:     snap = snapshot()
13:     ...
14:     rollback(snap)
```





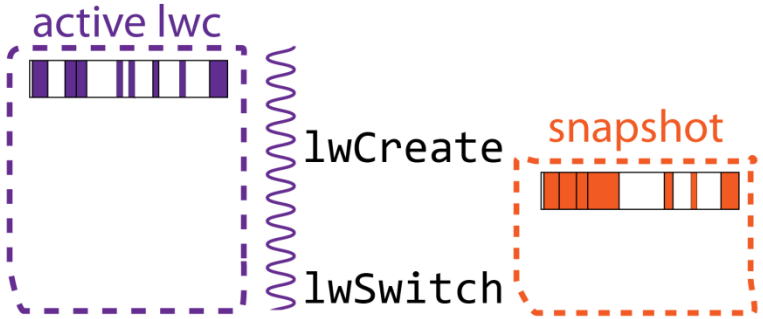
Snapshots

```
1:  function SNAPSHOT()
2:      new,caller,arg = lwCreate(...)
3:      if caller = -1 then
4:          return new
5:      else
6:          close(caller)
7:          return snapshot()
8:  function ROLLBACK(snap)
9:      lwSwitch(snap, 0)
10: function MAIN()
11:     ...
12:     snap = snapshot()
13:     ...
14:     rollback(snap)
```



Snapshots

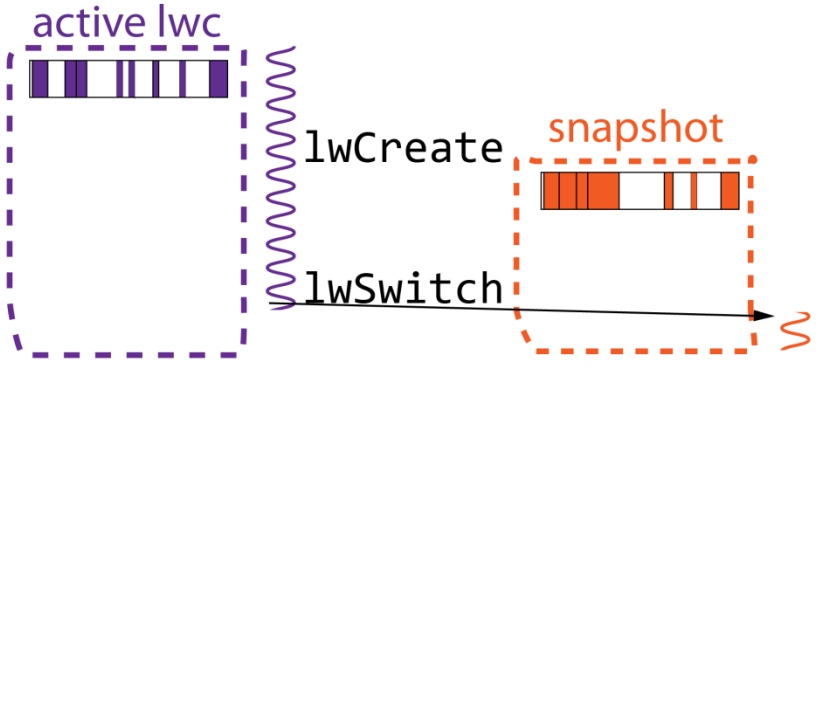
```
1:  function SNAPSHOT()
2:      new,caller,arg = lwCreate(...)
3:      if caller = -1 then
4:          return new
5:      else
6:          close(caller)
7:          return snapshot()
8:  function ROLLBACK(snap)
9:      lwSwitch(snap, 0)
10: function MAIN()
11:     ...
12:     snap = snapshot()
13:     ...
14:     rollback(snap)
```





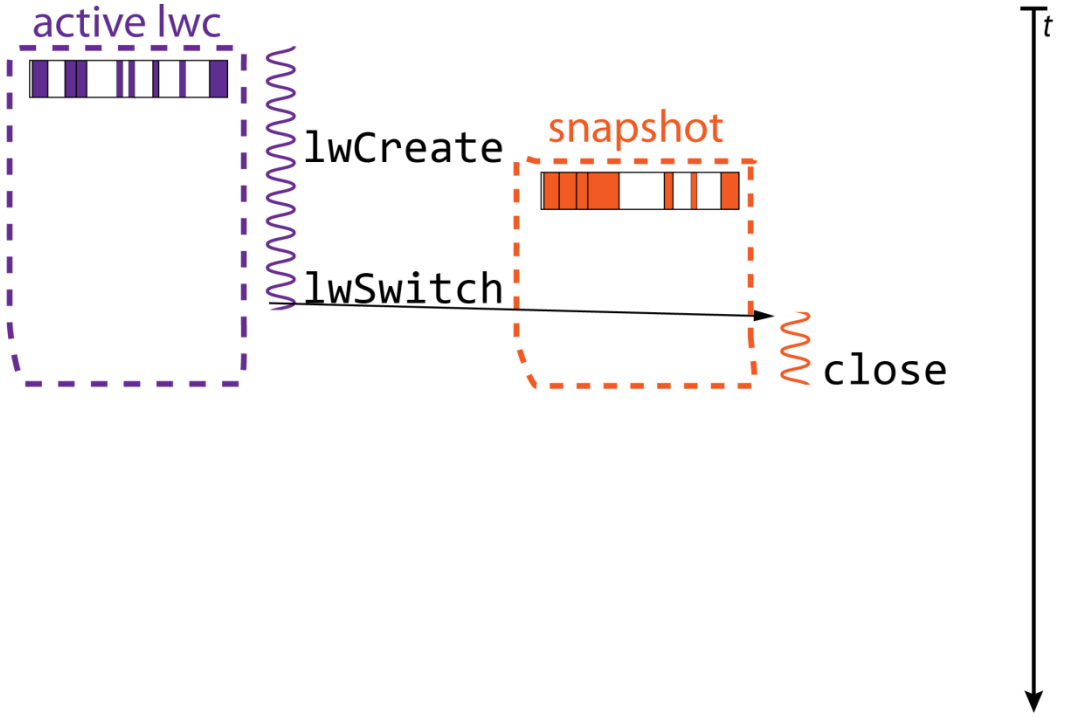
Snapshots

```
1:  function SNAPSHOT()
2:      new,caller,arg = lwCreate(...)
3:      if caller = -1 then
4:          return new
5:      else
6:          close(caller)
7:          return snapshot()
8:  function ROLLBACK(snap)
9:      lwSwitch(snap, 0)
10: function MAIN()
11:     ...
12:     snap = snapshot()
13:     ...
14:     rollback(snap)
```



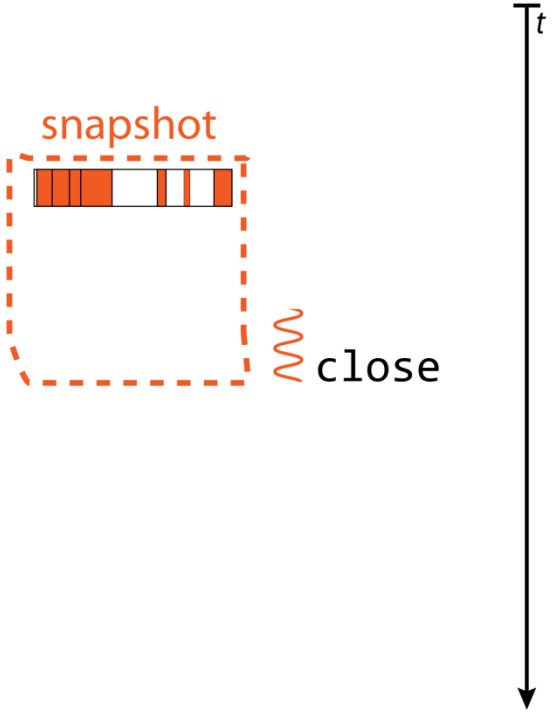
Snapshots

```
1:  function SNAPSHOT()
2:      new,caller,arg = lwCreate(...)
3:      if caller = -1 then
4:          return new
5:      else
6:          close(caller)
7:          return snapshot()
8:  function ROLLBACK(snap)
9:      lwSwitch(snap, 0)
10: function MAIN()
11:     ...
12:     snap = snapshot()
13:     ...
14:     rollback(snap)
```



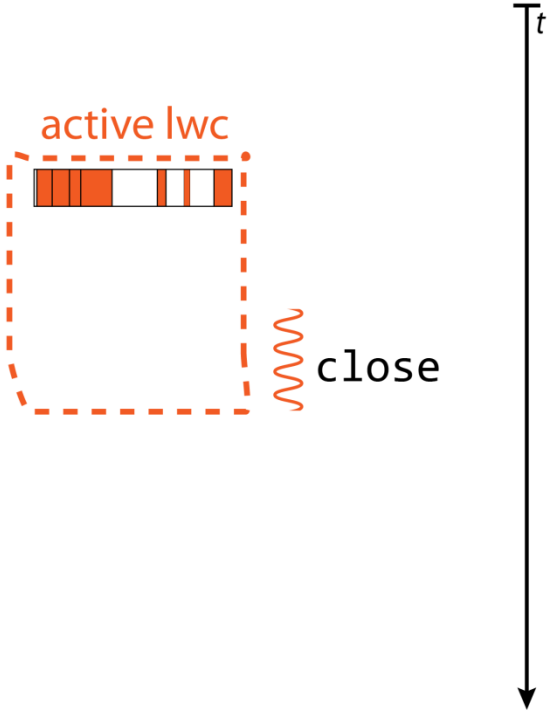
Snapshots

```
1:  function SNAPSHOT()
2:      new,caller,arg = lwCreate(...)
3:      if caller = -1 then
4:          return new
5:      else
6:          close(caller)
7:          return snapshot()
8:  function ROLLBACK(snap)
9:      lwSwitch(snap, 0)
10: function MAIN()
11:     ...
12:     snap = snapshot()
13:     ...
14:     rollback(snap)
```



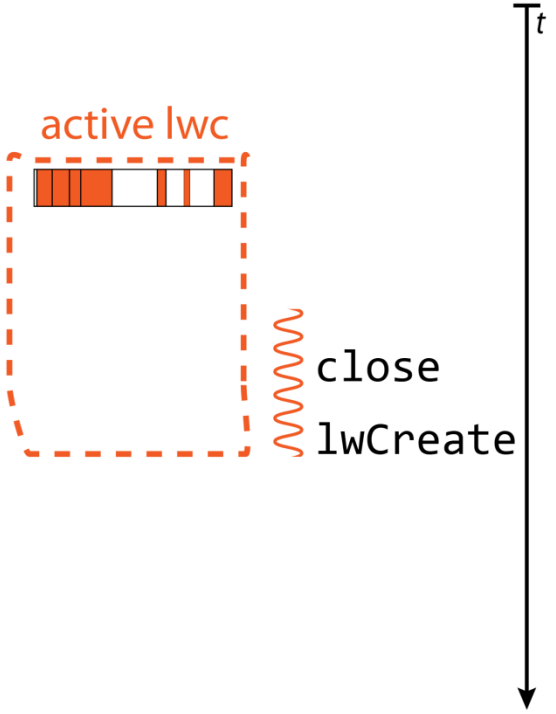
Snapshots

```
1:  function SNAPSHOT()
2:      new,caller,arg = lwCreate(...)
3:      if caller = -1 then
4:          return new
5:      else
6:          close(caller)
7:          return snapshot()
8:  function ROLLBACK(snap)
9:      lwSwitch(snap, 0)
10: function MAIN()
11:     ...
12:     snap = snapshot()
13:     ...
14:     rollback(snap)
```



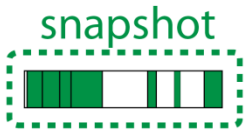
Snapshots

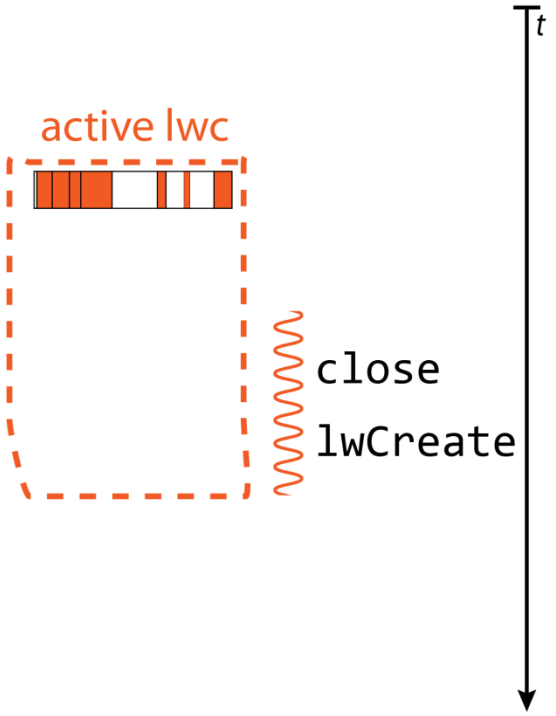
```
1:  function SNAPSHOT()
2:      new,caller,arg = lwCreate(...)
3:      if caller = -1 then
4:          return new
5:      else
6:          close(caller)
7:          return snapshot()
8:  function ROLLBACK(snap)
9:      lwSwitch(snap, 0)
10: function MAIN()
11:     ...
12:     snap = snapshot()
13:     ...
14:     rollback(snap)
```



Snapshots

```
1:  function SNAPSHOT()
2:      new,caller,arg = lwCreate(...)
3:      if caller = -1 then
4:          return new
5:      else
6:          close(caller)
7:          return snapshot()
8:  function ROLLBACK(snap)
9:      lwSwitch(snap, 0)
10: function MAIN()
11:     ...
12:     snap = snapshot()
13:     ...
14:     rollback(snap)
```

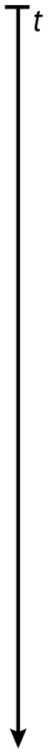




Reference Monitor

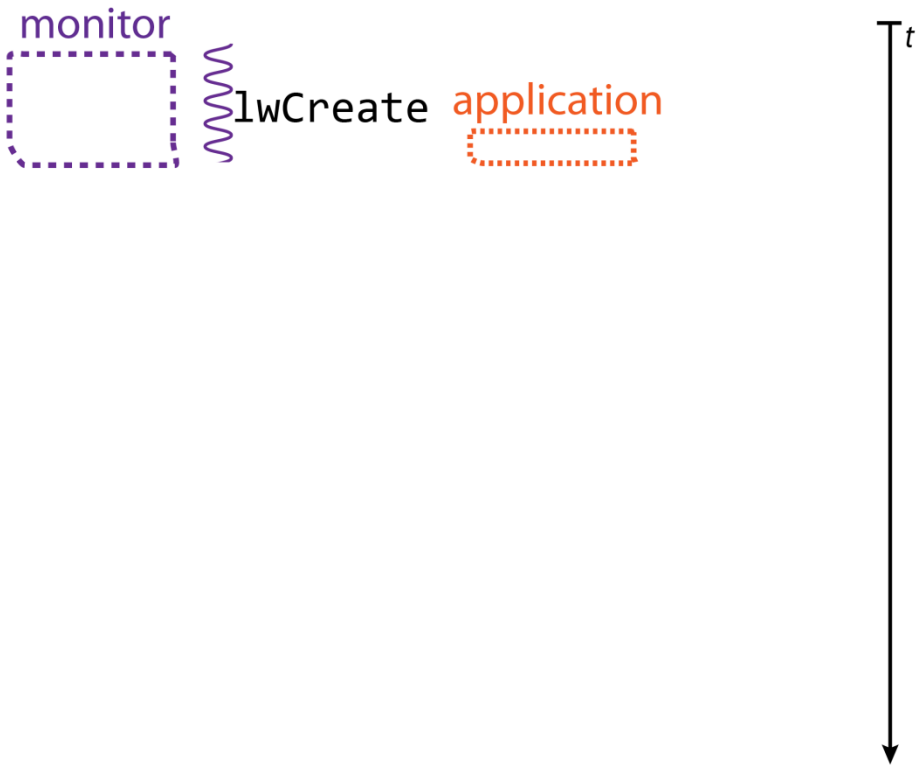
```
1:  function MONITOR(child)
2:      _,call = lwSwitch(child, NULL)
3:      loop
4:          if is_allowed(call) then
5:              specs = { ... }
6:              rv = lwSyscall(child, specs,
                                call.num, call.params)
7:              out.err,out.rv = errno, rv
8:          else
9:              out.err,out.rv = EPERM, -1
10:         _,call = lwSwitch(child, out)
11: function MAIN
12:     specs = { ... }
13:     child,c,_ = lwCreate(specs, LWC_SYSTRAP)
14:     if c = -1 then
15:         monitor(child)
16:     privdrop() && run()
```





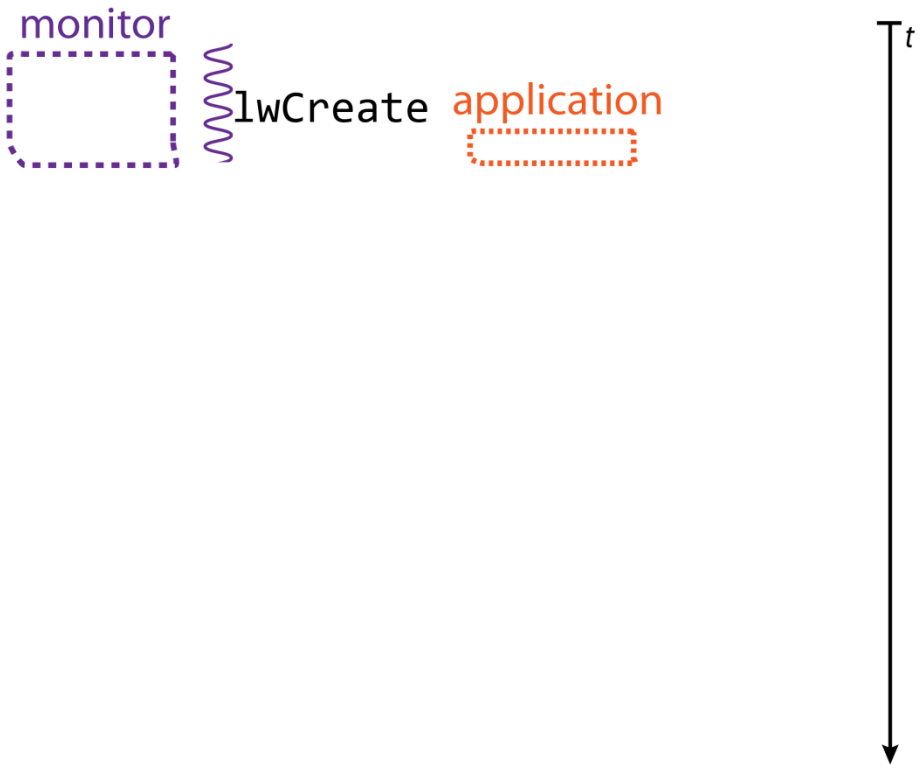
Reference Monitor

```
1:  function MONITOR(child)
2:      _,call = lwSwitch(child, NULL)
3:      loop
4:          if is_allowed(call) then
5:              specs = { ... }
6:              rv = lwSyscall(child, specs,
                                call.num, call.params)
7:              out.err,out.rv = errno, rv
8:          else
9:              out.err,out.rv = EPERM, -1
10:         _,call = lwSwitch(child, out)
11: function MAIN
12:     specs = { ... }
13:     child,c,_ = lwCreate(specs, LWC_SYSTRAP)
14:     if c = -1 then
15:         monitor(child)
16:     privdrop() && run()
```



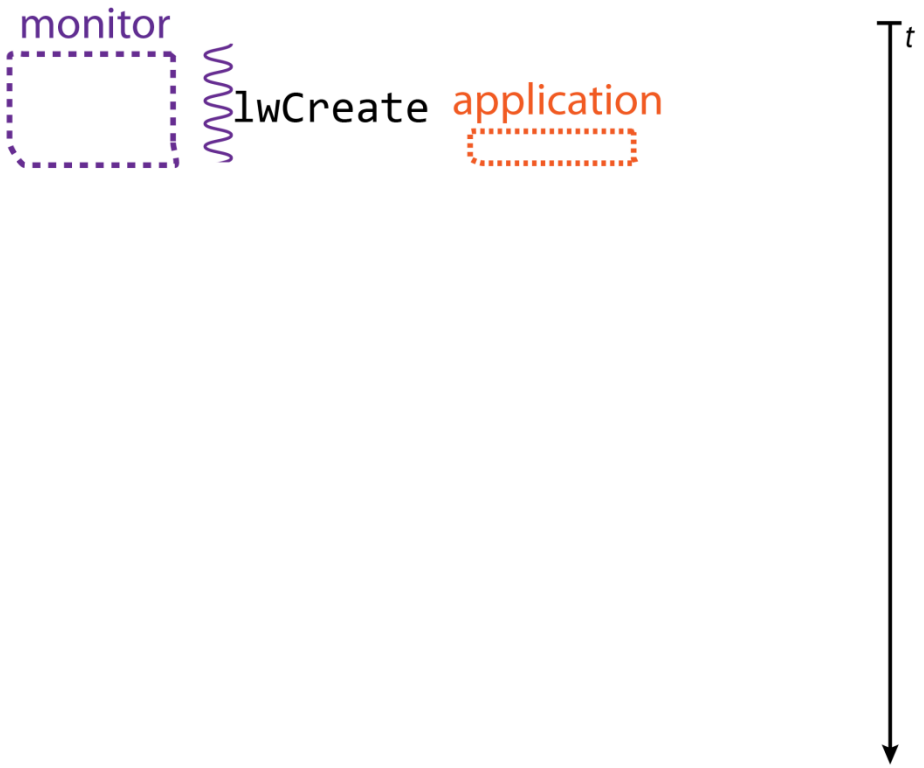
Reference Monitor

```
1:  function MONITOR(child)
2:      _,call = lwSwitch(child, NULL)
3:      loop
4:          if is_allowed(call) then
5:              specs = { ... }
6:              rv = lwSyscall(child, specs,
                               call.num, call.params)
7:              out.err,out.rv = errno, rv
8:          else
9:              out.err,out.rv = EPERM, -1
10:         _,call = lwSwitch(child, out)
11: function MAIN
12:     specs = { ... }
13:     child,c,_ = lwCreate(specs, LWC_SYSTRAP)
14:     if c = -1 then
15:         monitor(child)
16:     privdrop() && run()
```



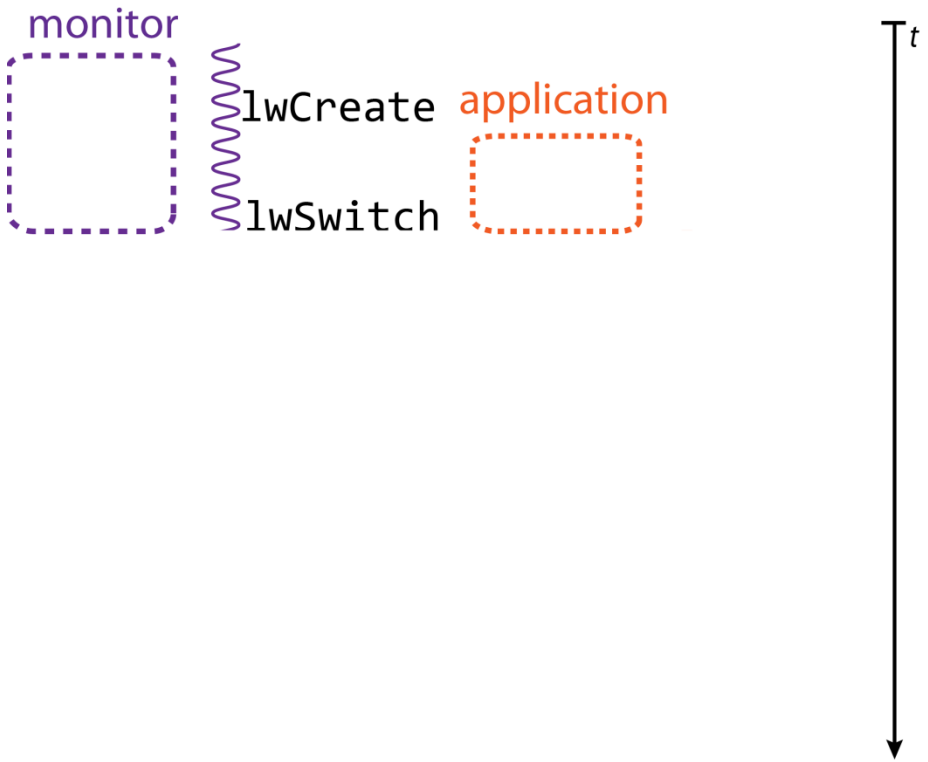
Reference Monitor

```
1:  function MONITOR(child)
2:      _,call = lwSwitch(child, NULL)
3:      loop
4:          if is_allowed(call) then
5:              specs = { ... }
6:              rv = lwSyscall(child, specs,
                                call.num, call.params)
7:              out.err,out.rv = errno, rv
8:          else
9:              out.err,out.rv = EPERM, -1
10:         _,call = lwSwitch(child, out)
11: function MAIN
12:     specs = { ... }
13:     child,c,_ = lwCreate(specs, LWC_SYSTRAP)
14:     if c = -1 then
15:         monitor(child)
16:     privdrop() && run()
```



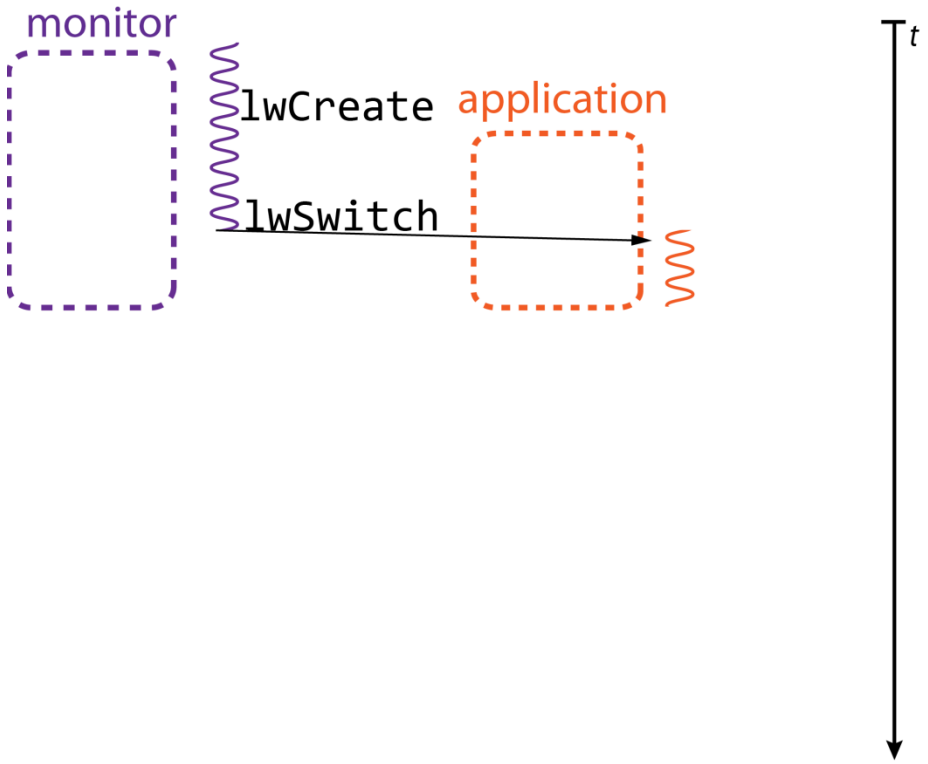
Reference Monitor

```
1:  function MONITOR(child)
2:      _,call = lwSwitch(child, NULL)
3:      loop
4:          if is_allowed(call) then
5:              specs = { ... }
6:              rv = lwSyscall(child, specs,
                                call.num, call.params)
7:              out.err,out.rv = errno, rv
8:          else
9:              out.err,out.rv = EPERM, -1
10:         _,call = lwSwitch(child, out)
11: function MAIN
12:     specs = { ... }
13:     child,c,_ = lwCreate(specs, LWC_SYSTRAP)
14:     if c = -1 then
15:         monitor(child)
16:     privdrop() && run()
```



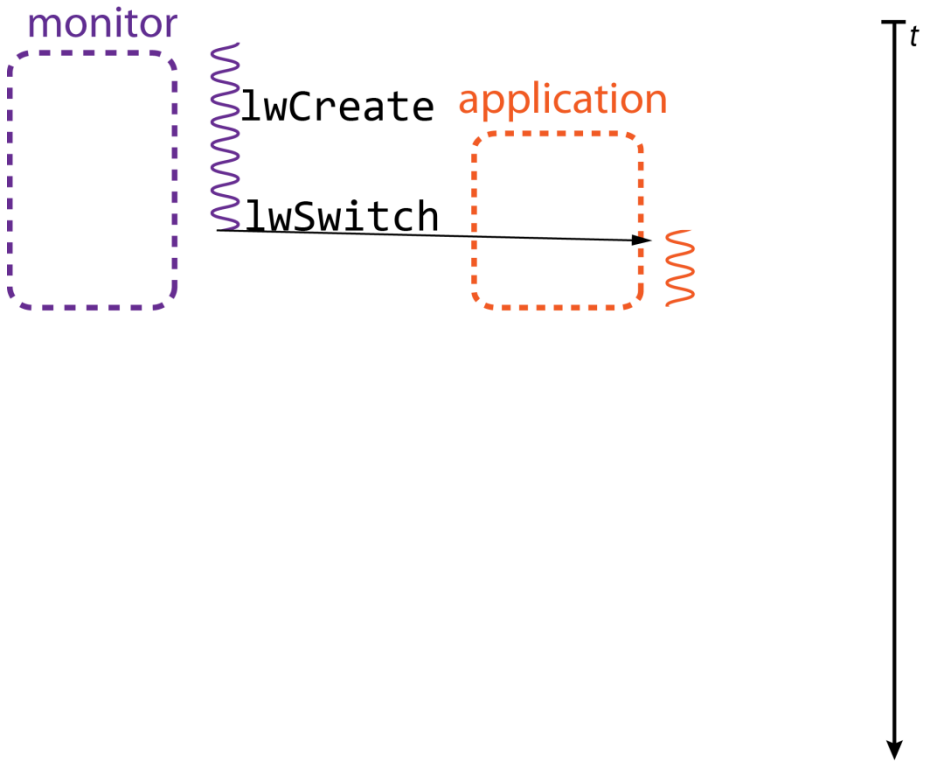
Reference Monitor

```
1:  function MONITOR(child)
2:      _,call = lwSwitch(child, NULL)
3:      loop
4:          if is_allowed(call) then
5:              specs = { ... }
6:              rv = lwSyscall(child, specs,
                               call.num, call.params)
7:              out.err,out.rv = errno, rv
8:          else
9:              out.err,out.rv = EPERM, -1
10:         _,call = lwSwitch(child, out)
11: function MAIN
12:     specs = { ... }
13:     child,c,_ = lwCreate(specs, LWC_SYSTRAP)
14:     if c = -1 then
15:         monitor(child)
16:     privdrop() && run()
```



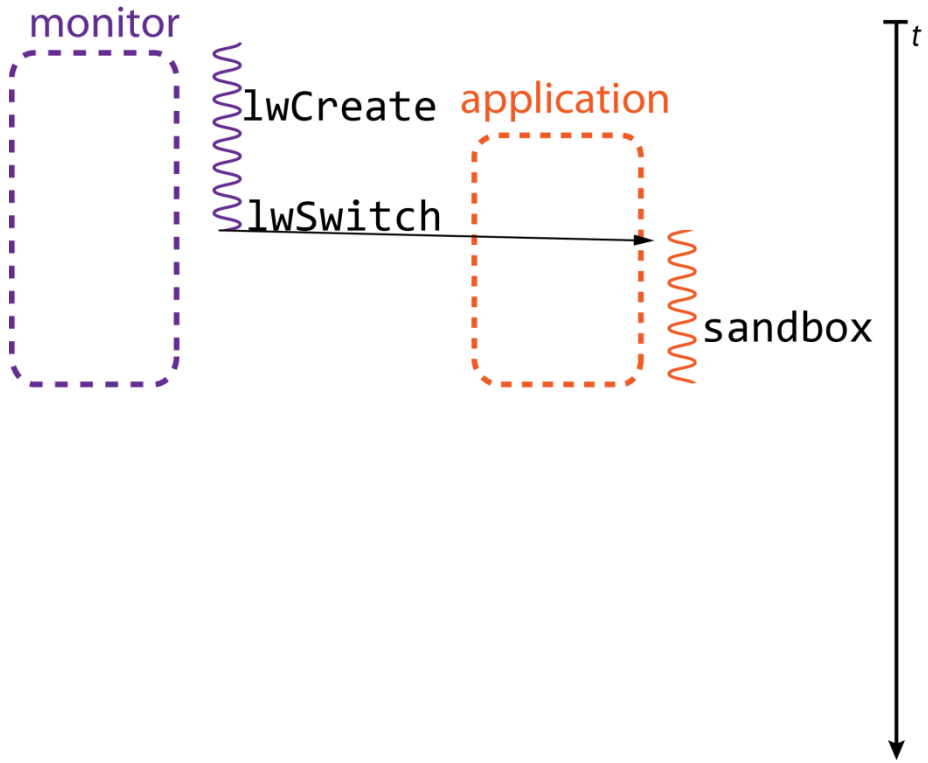
Reference Monitor

```
1:  function MONITOR(child)
2:      _,call = lwSwitch(child, NULL)
3:      loop
4:          if is_allowed(call) then
5:              specs = { ... }
6:              rv = lwSyscall(child, specs,
                               call.num, call.params)
7:              out.err,out.rv = errno, rv
8:          else
9:              out.err,out.rv = EPERM, -1
10:         _,call = lwSwitch(child, out)
11: function MAIN
12:     specs = { ... }
13:     child,c,_ = lwCreate(specs, LWC_SYSTRAP)
14:     if c = -1 then
15:         monitor(child)
16:     privdrop() && run()
```



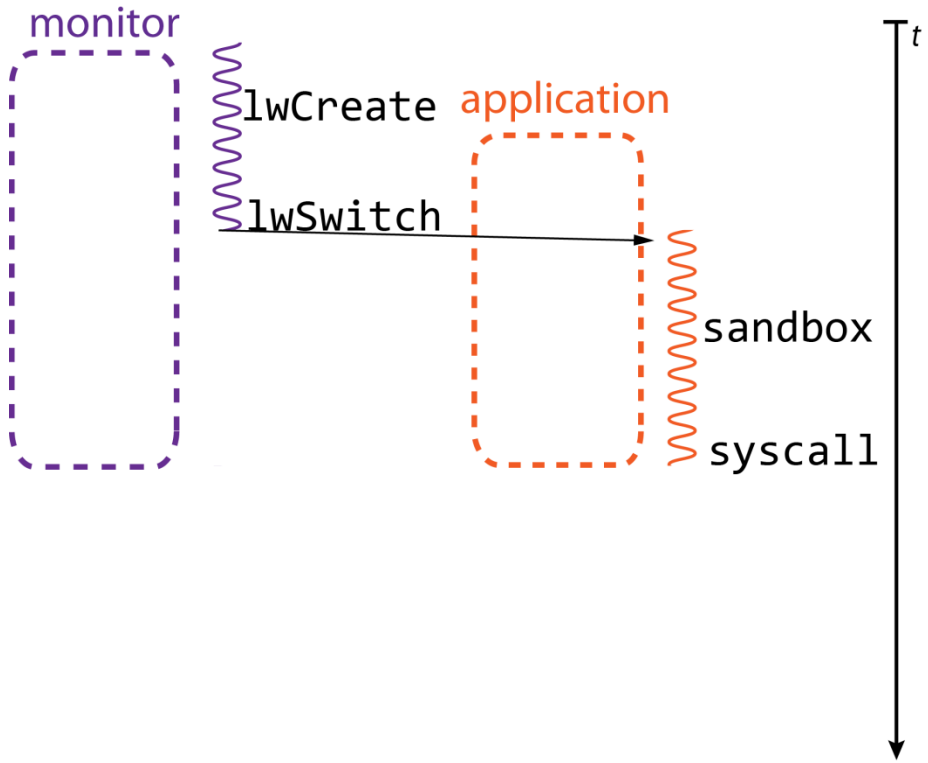
Reference Monitor

```
1:  function MONITOR(child)
2:      _,call = lwSwitch(child, NULL)
3:      loop
4:          if is_allowed(call) then
5:              specs = { ... }
6:              rv = lwSyscall(child, specs,
                               call.num, call.params)
7:              out.err,out.rv = errno, rv
8:          else
9:              out.err,out.rv = EPERM, -1
10:         _,call = lwSwitch(child, out)
11: function MAIN
12:     specs = { ... }
13:     child,c,_ = lwCreate(specs, LWC_SYSTRAP)
14:     if c = -1 then
15:         monitor(child)
16:     privdrop() && run()
```



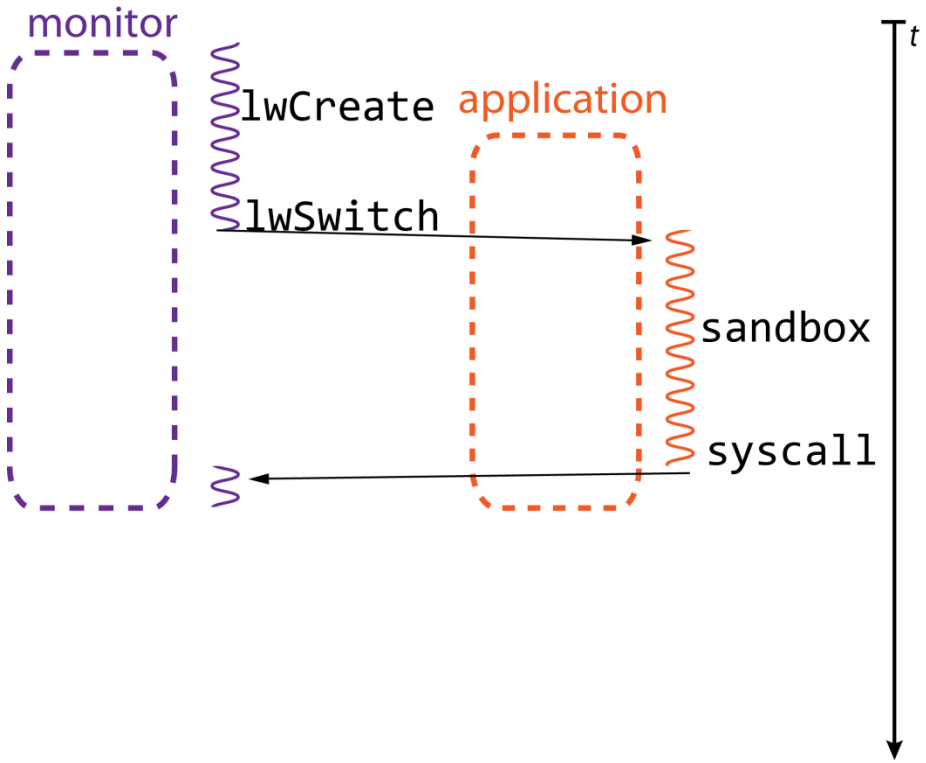
Reference Monitor

```
1:  function MONITOR(child)
2:      _,call = lwSwitch(child, NULL)
3:      loop
4:          if is_allowed(call) then
5:              specs = { ... }
6:              rv = lwSyscall(child, specs,
                               call.num, call.params)
7:              out.err,out.rv = errno, rv
8:          else
9:              out.err,out.rv = EPERM, -1
10:         _,call = lwSwitch(child, out)
11: function MAIN
12:     specs = { ... }
13:     child,c,_ = lwCreate(specs, LWC_SYSTRAP)
14:     if c = -1 then
15:         monitor(child)
16:     privdrop() && run()
```



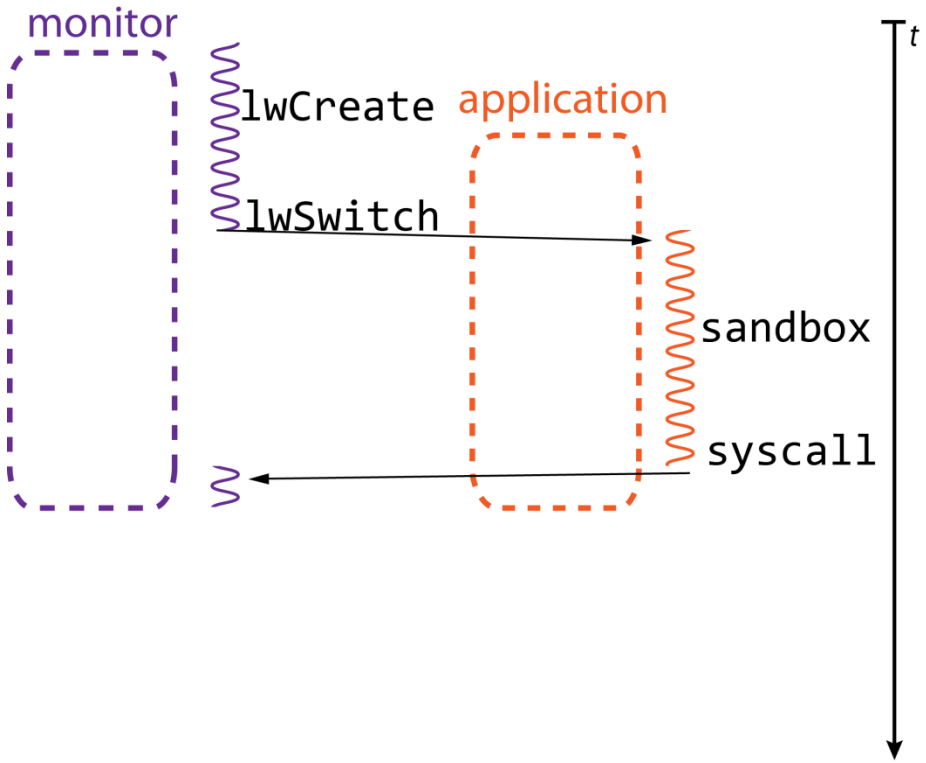
Reference Monitor

```
1:  function MONITOR(child)
2:      _,call = lwSwitch(child, NULL)
3:      loop
4:          if is_allowed(call) then
5:              specs = { ... }
6:              rv = lwSyscall(child, specs,
                               call.num, call.params)
7:              out.err,out.rv = errno, rv
8:          else
9:              out.err,out.rv = EPERM, -1
10:         _,call = lwSwitch(child, out)
11: function MAIN
12:     specs = { ... }
13:     child,c,_ = lwCreate(specs, LWC_SYSTRAP)
14:     if c = -1 then
15:         monitor(child)
16:     privdrop() && run()
```



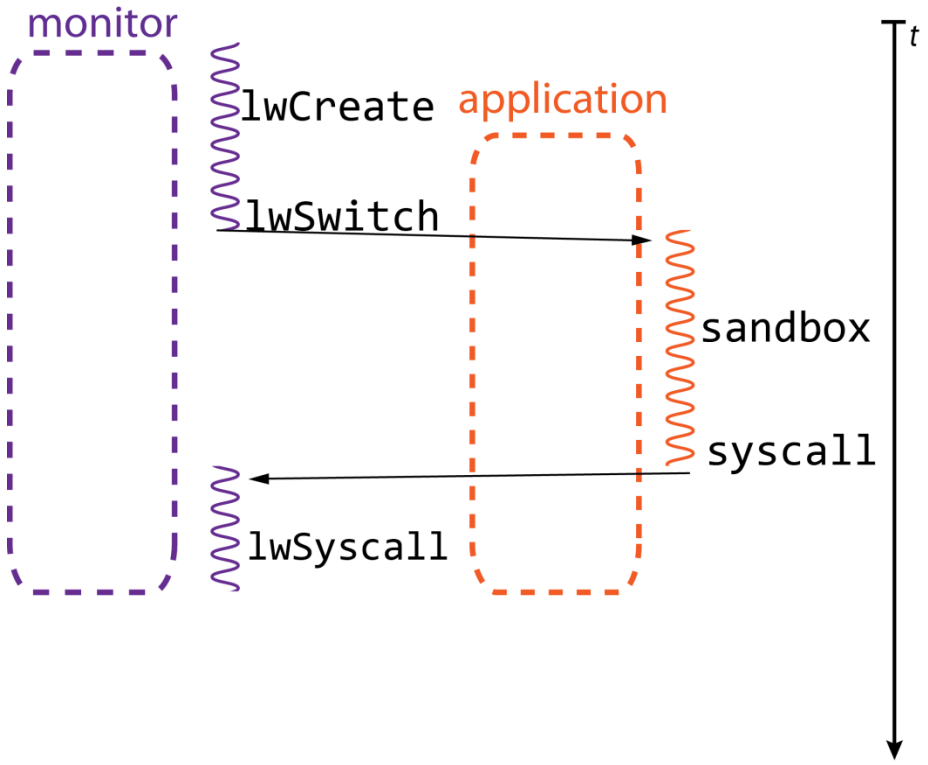
Reference Monitor

```
1:  function MONITOR(child)
2:      _,call = lwSwitch(child, NULL)
3:      loop
4:          if is_allowed(call) then
5:              specs = { ... }
6:              rv = lwSyscall(child, specs,
                               call.num, call.params)
7:              out.err,out.rv = errno, rv
8:          else
9:              out.err,out.rv = EPERM, -1
10:         _,call = lwSwitch(child, out)
11: function MAIN
12:     specs = { ... }
13:     child,c,_ = lwCreate(specs, LWC_SYSTRAP)
14:     if c = -1 then
15:         monitor(child)
16:     privdrop() && run()
```



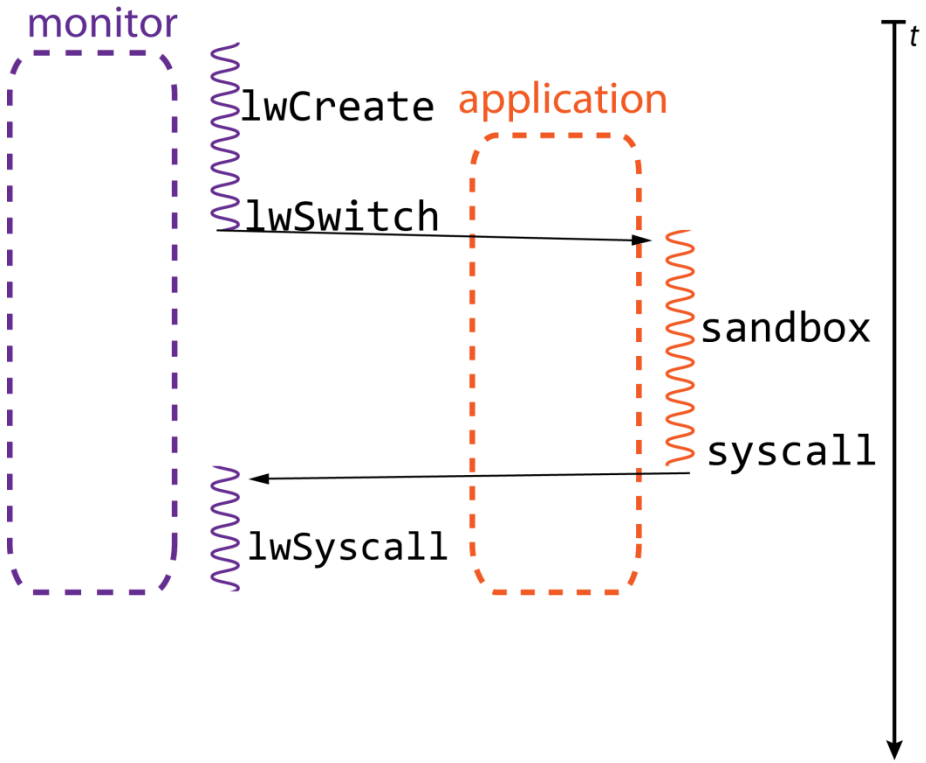
Reference Monitor

```
1:  function MONITOR(child)
2:      _,call = lwSwitch(child, NULL)
3:      loop
4:          if is_allowed(call) then
5:              specs = { ... }
6:              rv = lwSyscall(child, specs,
                               call.num, call.params)
7:              out.err,out.rv = errno, rv
8:          else
9:              out.err,out.rv = EPERM, -1
10:         _,call = lwSwitch(child, out)
11: function MAIN
12:     specs = { ... }
13:     child,c,_ = lwCreate(specs, LWC_SYSTRAP)
14:     if c = -1 then
15:         monitor(child)
16:     privdrop() && run()
```



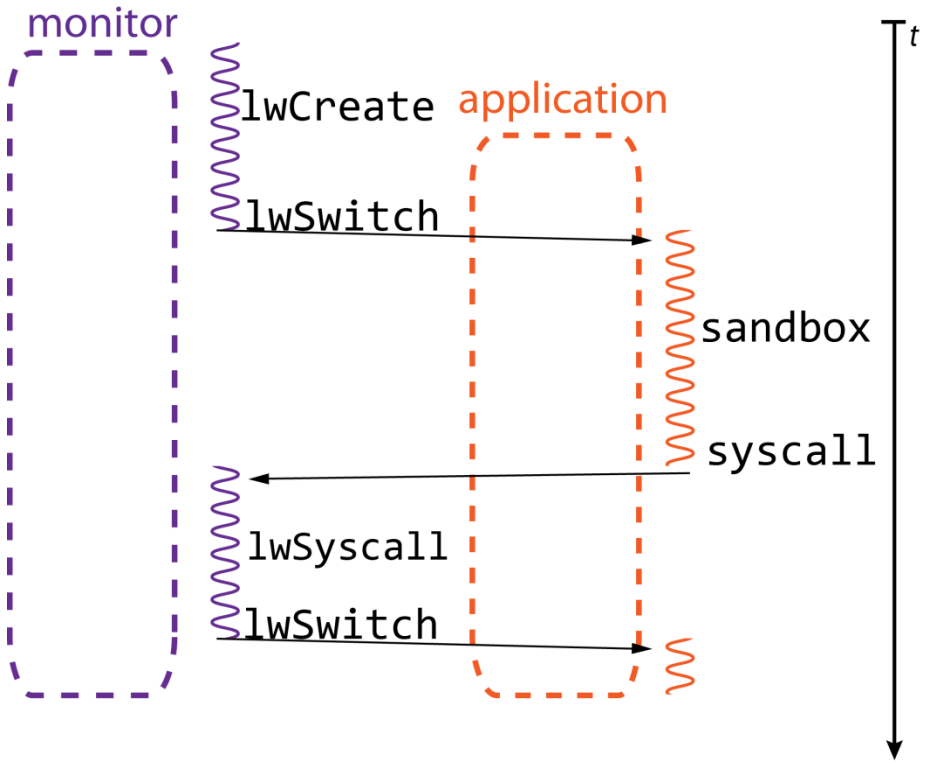
Reference Monitor

```
1:  function MONITOR(child)
2:      _,call = lwSwitch(child, NULL)
3:      loop
4:          if is_allowed(call) then
5:              specs = { ... }
6:              rv = lwSyscall(child, specs,
                               call.num, call.params)
7:              out.err,out.rv = errno, rv
8:          else
9:              out.err,out.rv = EPERM, -1
10:         _,call = lwSwitch(child, out)
11: function MAIN
12:     specs = { ... }
13:     child,c,_ = lwCreate(specs, LWC_SYSTRAP)
14:     if c = -1 then
15:         monitor(child)
16:     privdrop() && run()
```



Reference Monitor

```
1:  function MONITOR(child)
2:      _,call = lwSwitch(child, NULL)
3:      loop
4:          if is_allowed(call) then
5:              specs = { ... }
6:              rv = lwSyscall(child, specs,
                               call.num, call.params)
7:              out.err,out.rv = errno, rv
8:          else
9:              out.err,out.rv = EPERM, -1
10:         _,call = lwSwitch(child, out)
11: function MAIN
12:     specs = { ... }
13:     child,c,_ = lwCreate(specs, LWC_SYSTRAP)
14:     if c = -1 then
15:         monitor(child)
16:     privdrop() && run()
```



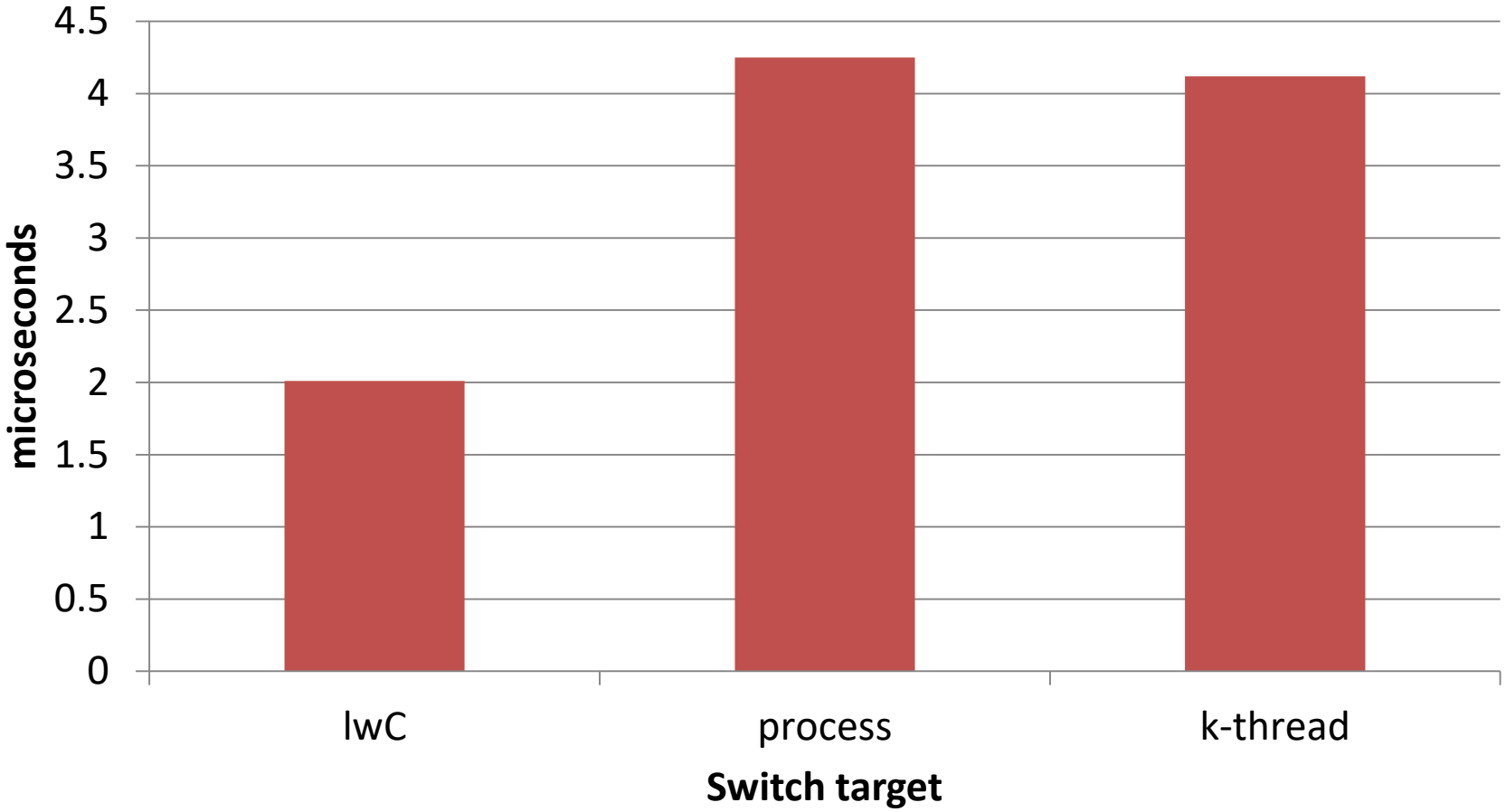
Evaluation

- Microbenchmarks
  - lwCreate
  - lwSwitch
  - Reference monitoring (systrap)
- Apache (v. 2.4.18)
  - Session isolation
  - Reference monitoring
- Nginx (v. 1.9.15)
  - Session isolation
  - Reference monitoring
  - SSL private key isolation
- PHP (v. 7.0.11)
  - Fast startup via snapshots

Creation/Destruction Cost

- No pages dirtied
  - 87.7 microseconds
- Additional COW cost per page dirtied
  - ~ 3.4 microseconds

Switching Cost



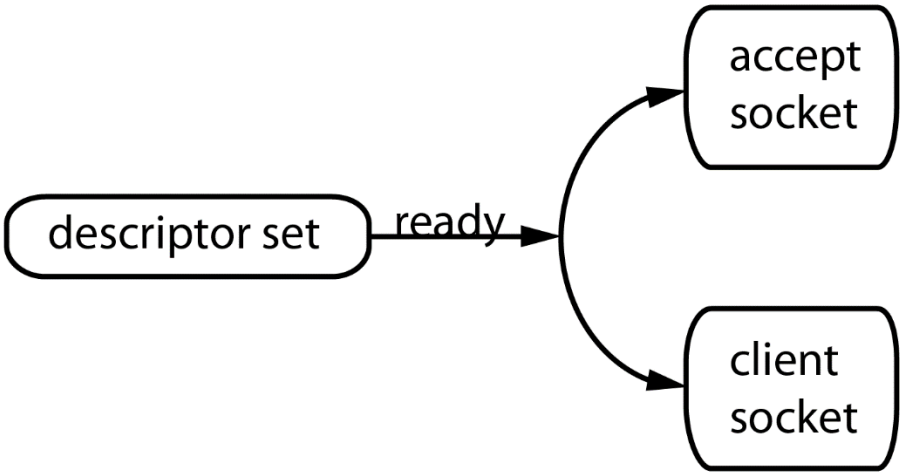
Evaluation – nginx

- High performance web server
  - Asynchronous & event-driven
  - No session isolation

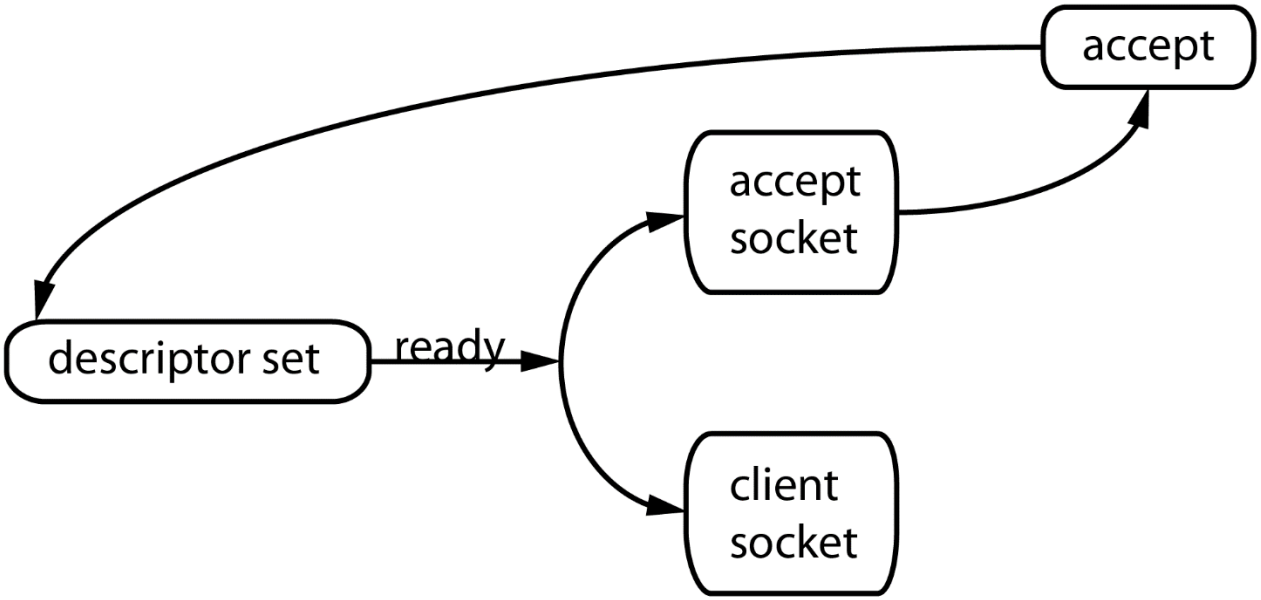
nginx execution



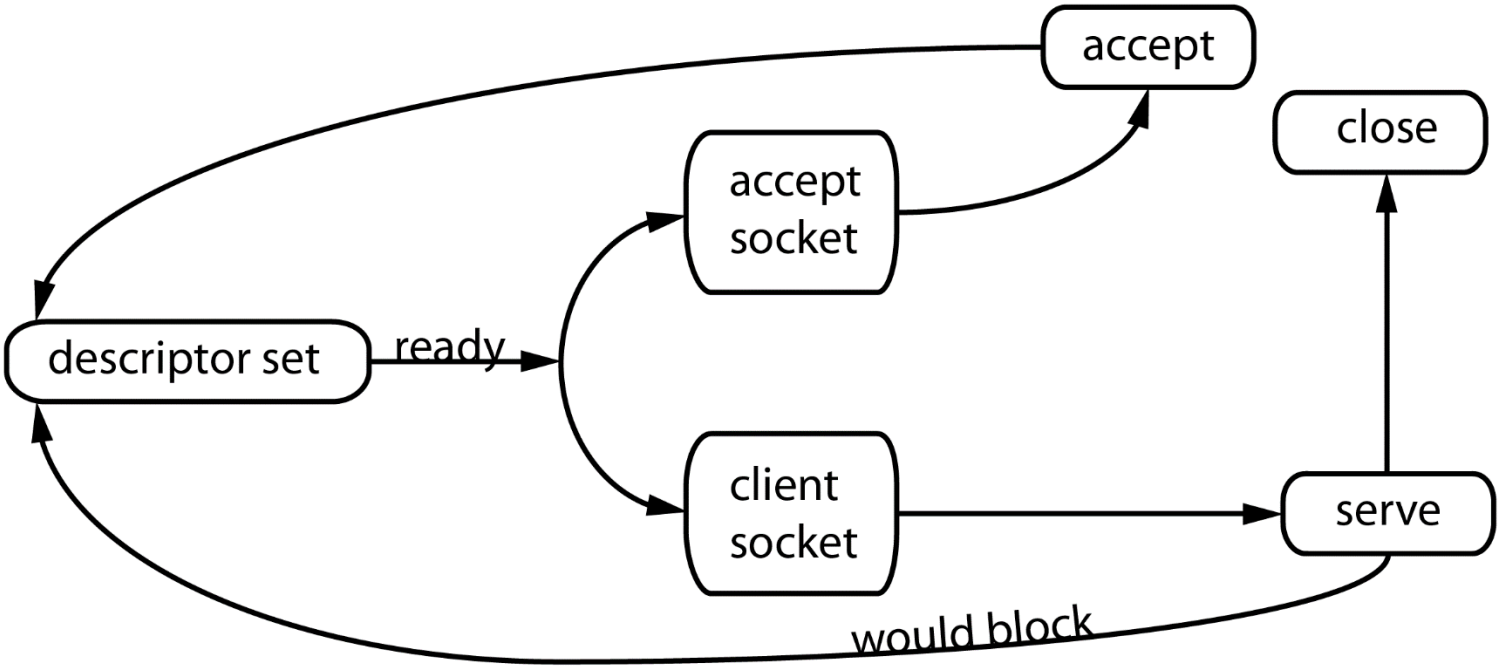
nginx execution



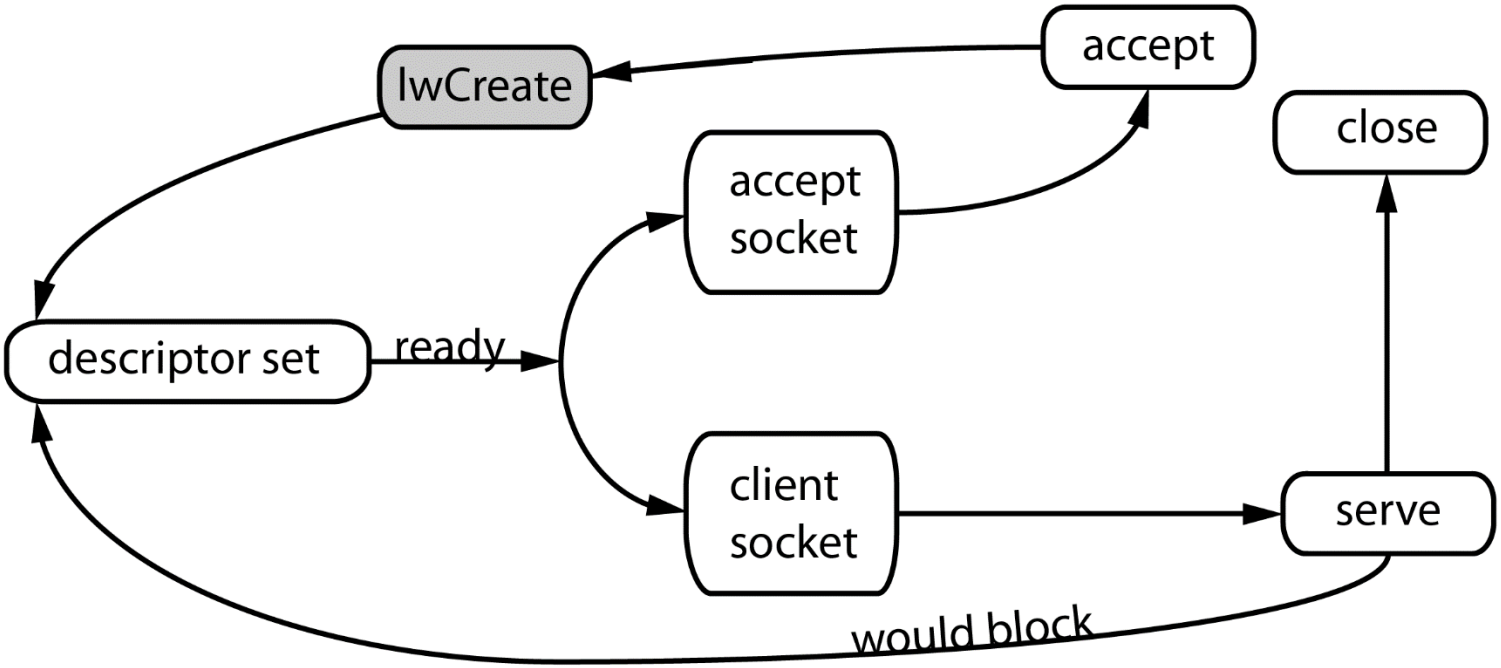
nginx execution



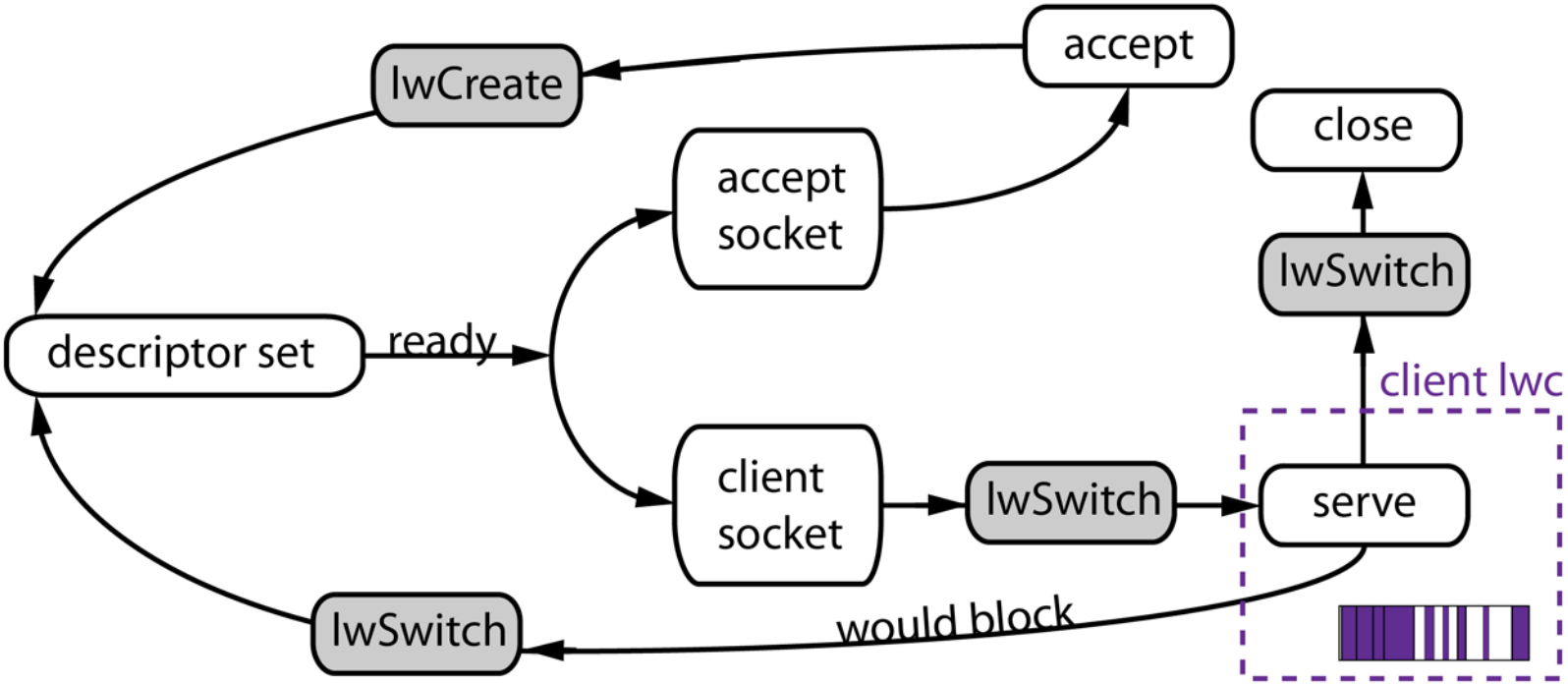
nginx execution



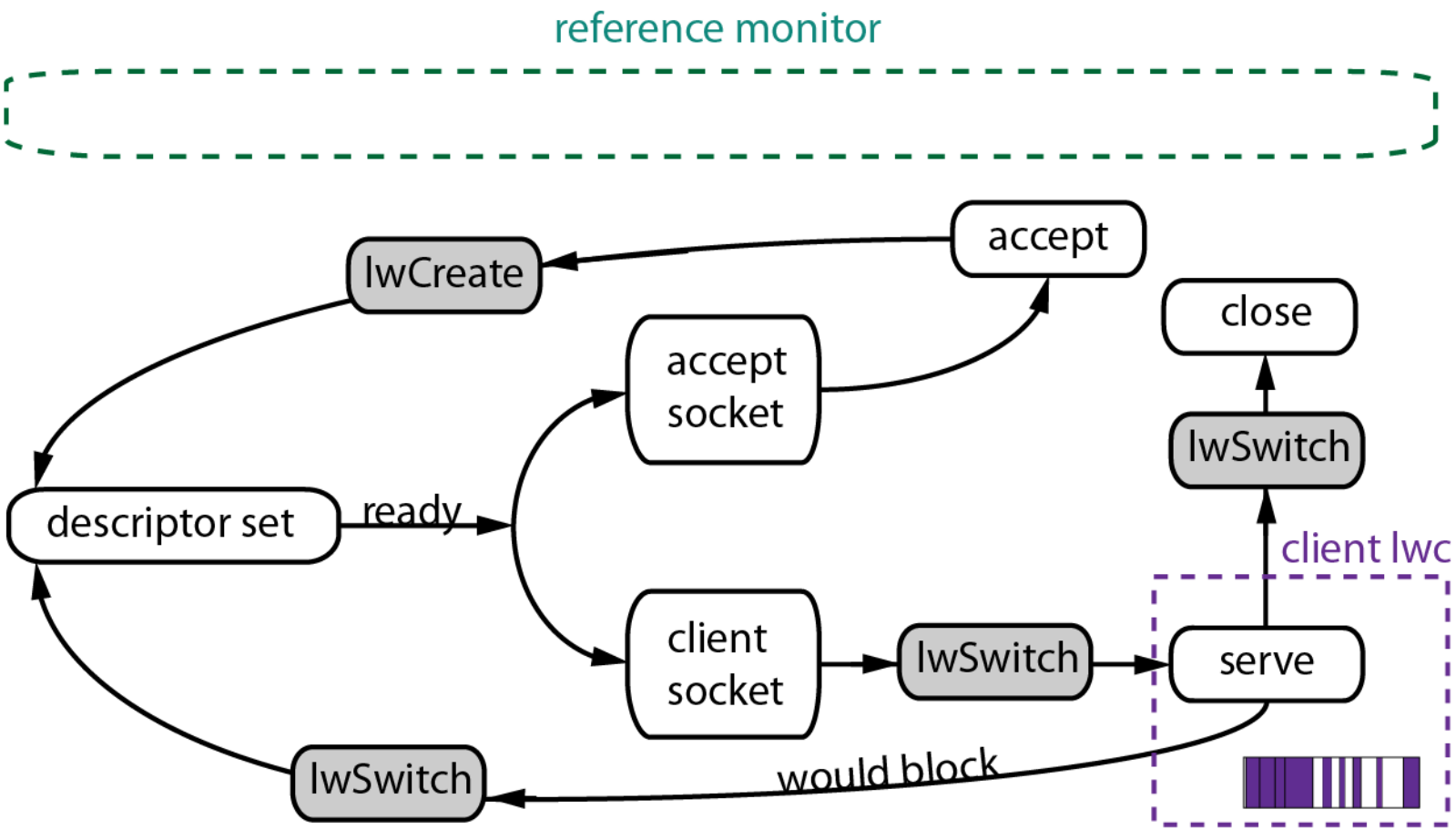
nginx execution



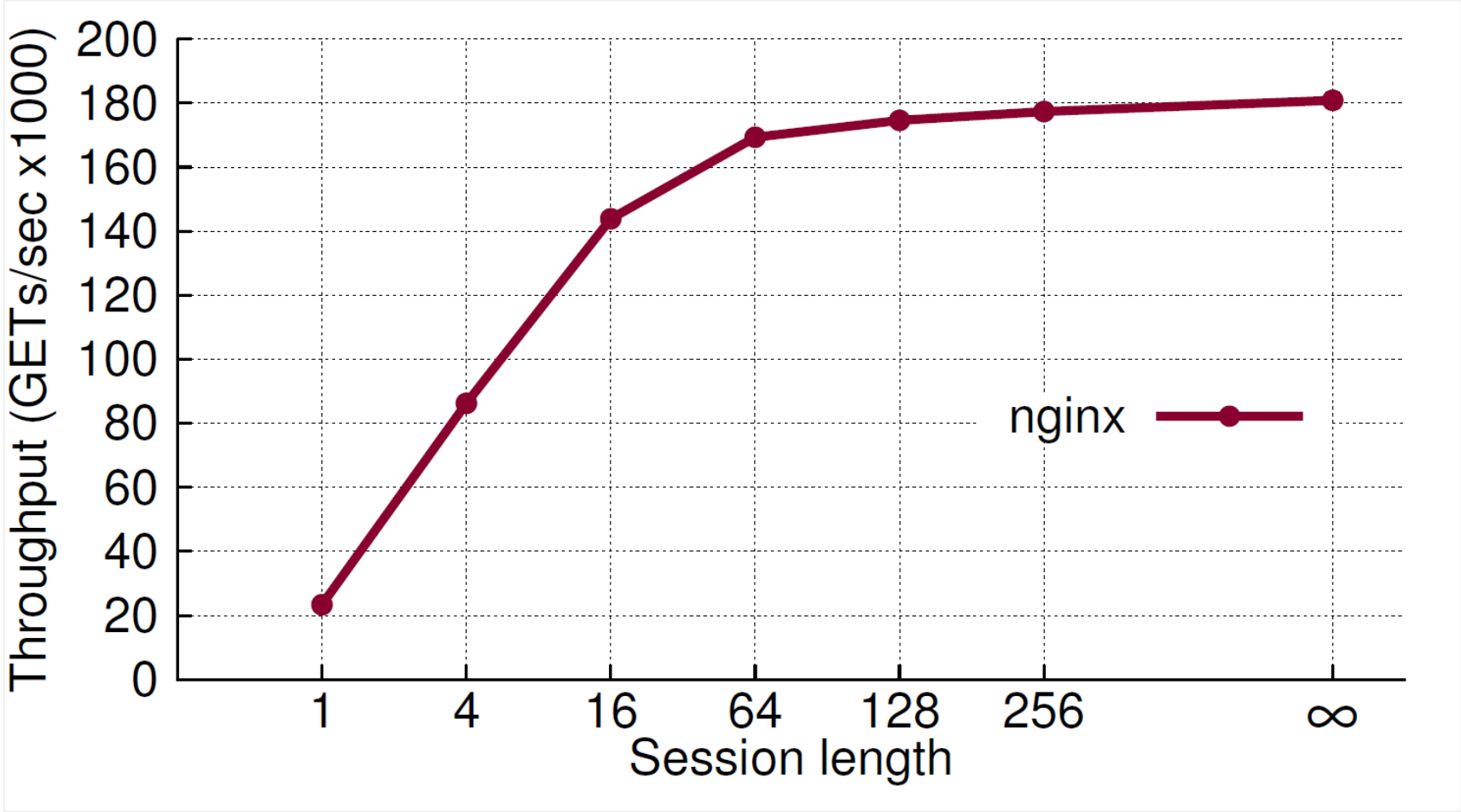
nginx execution



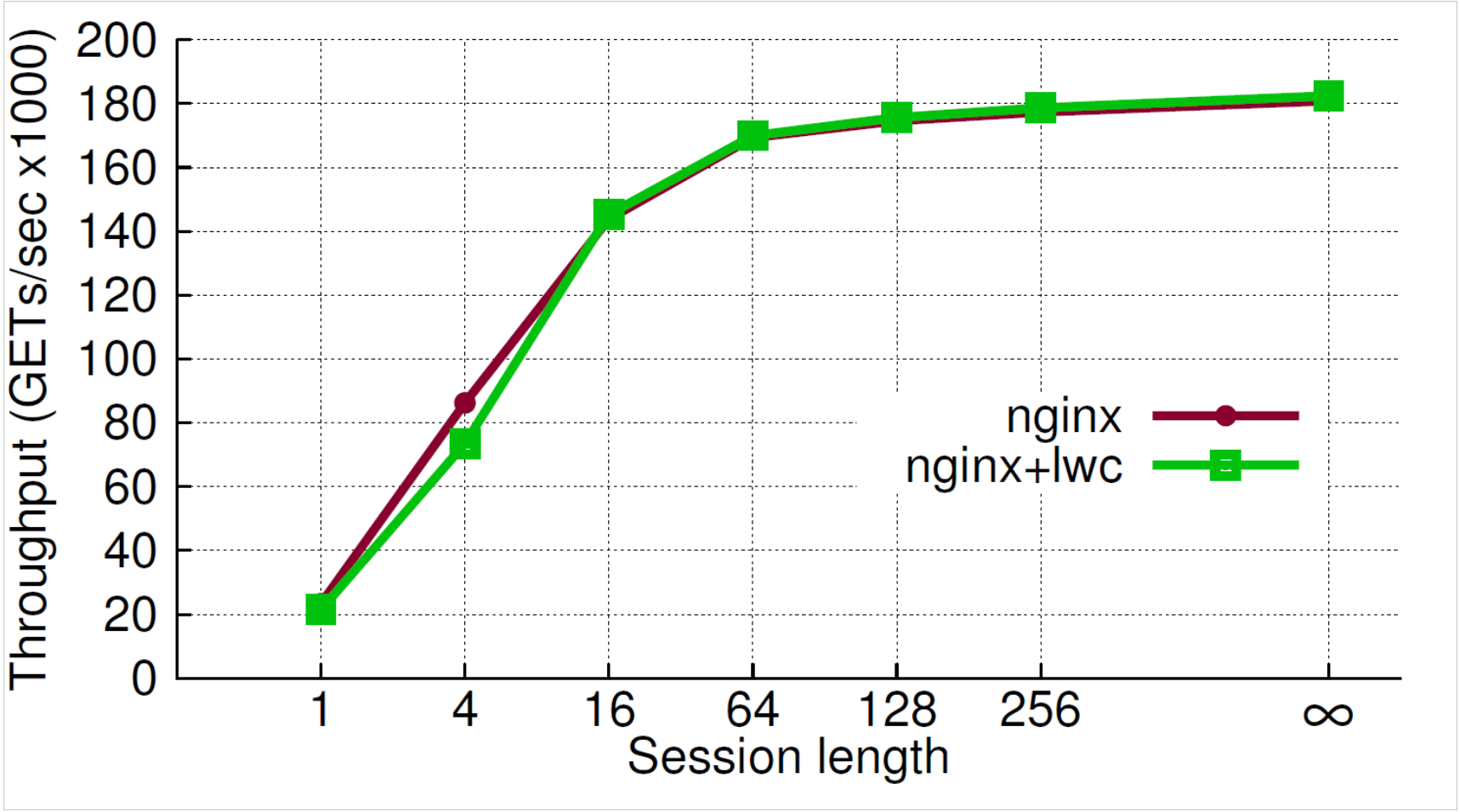
nginx execution



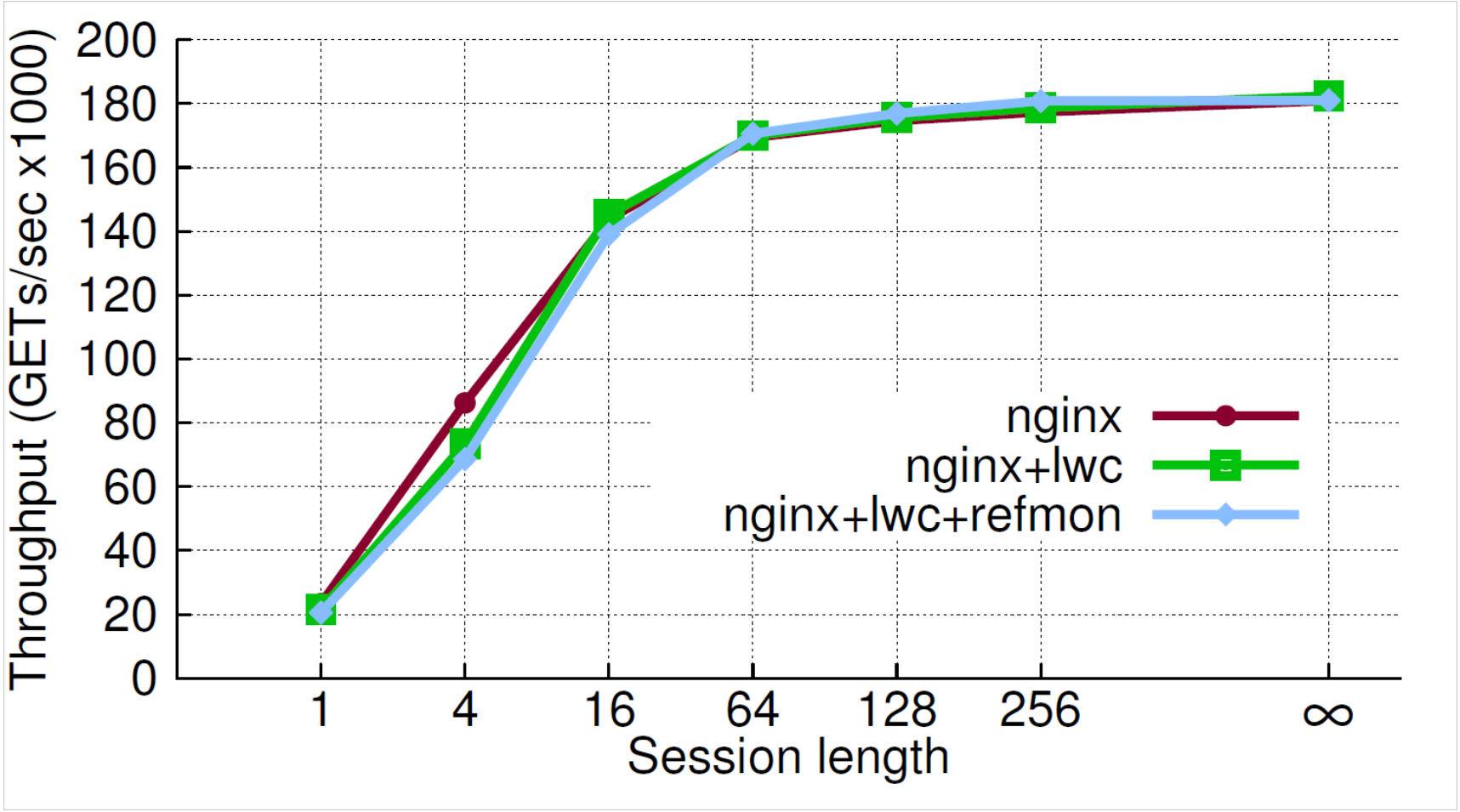
Evaluation - nginx



Evaluation - nginx



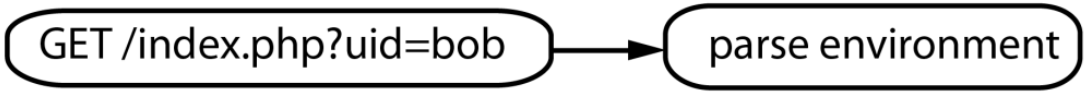
Evaluation - nginx



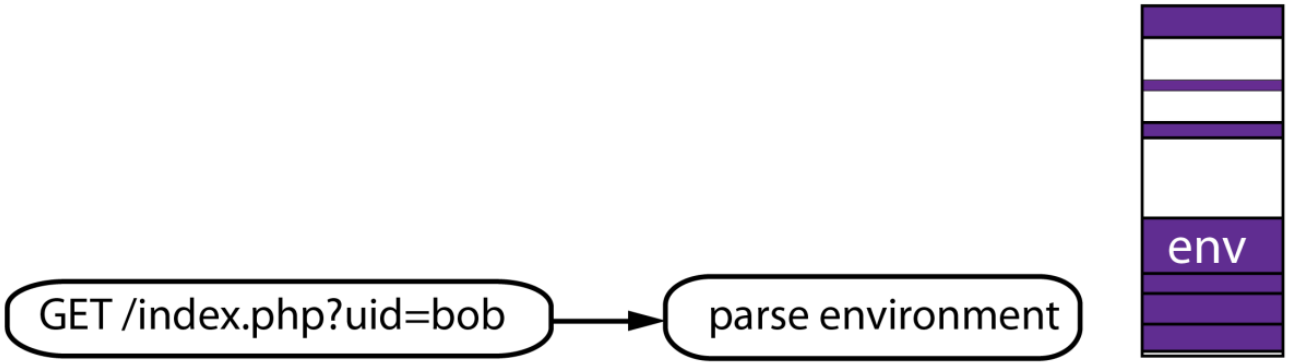
Evaluation - PHP-FCGI

- Use lwC snapshots to speed up PHP
- Evaluated using Zend MVC Framework

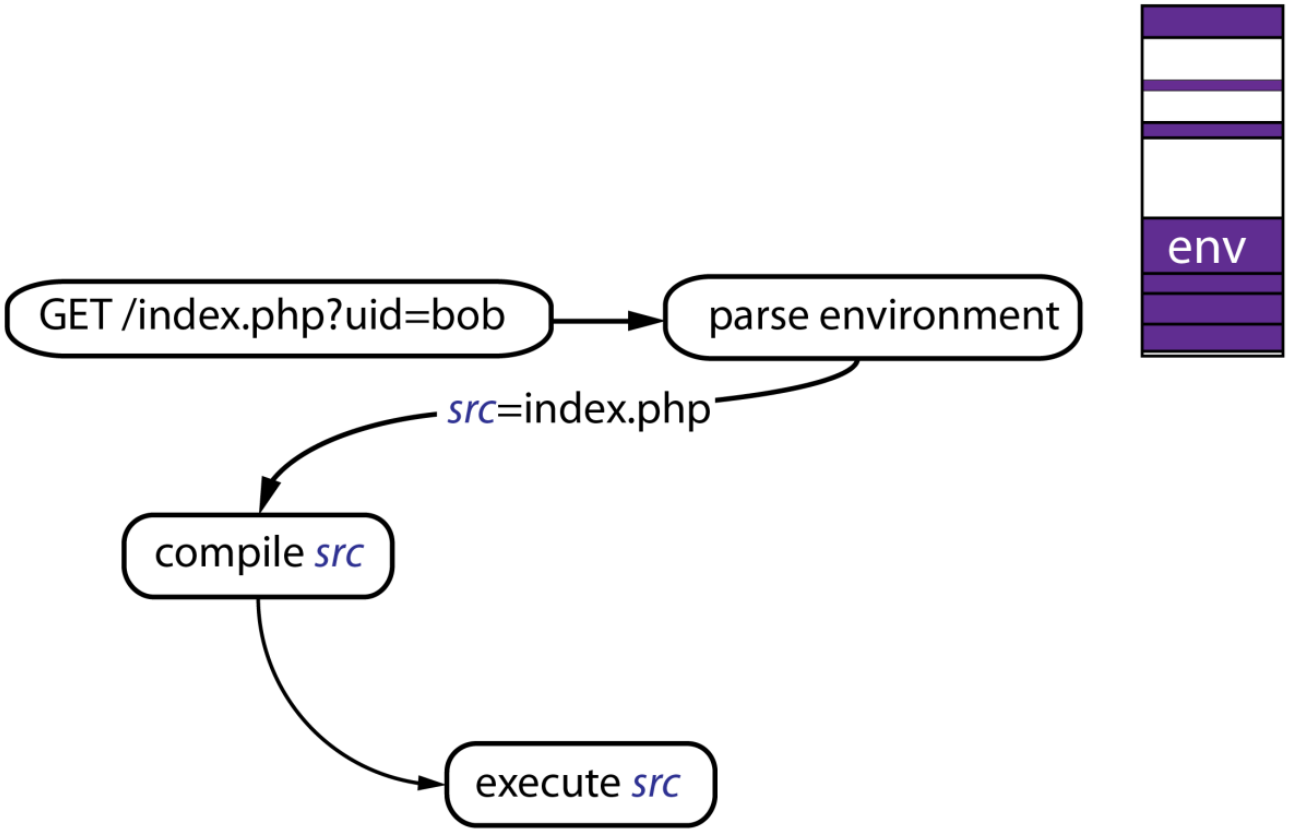
PHP Execution



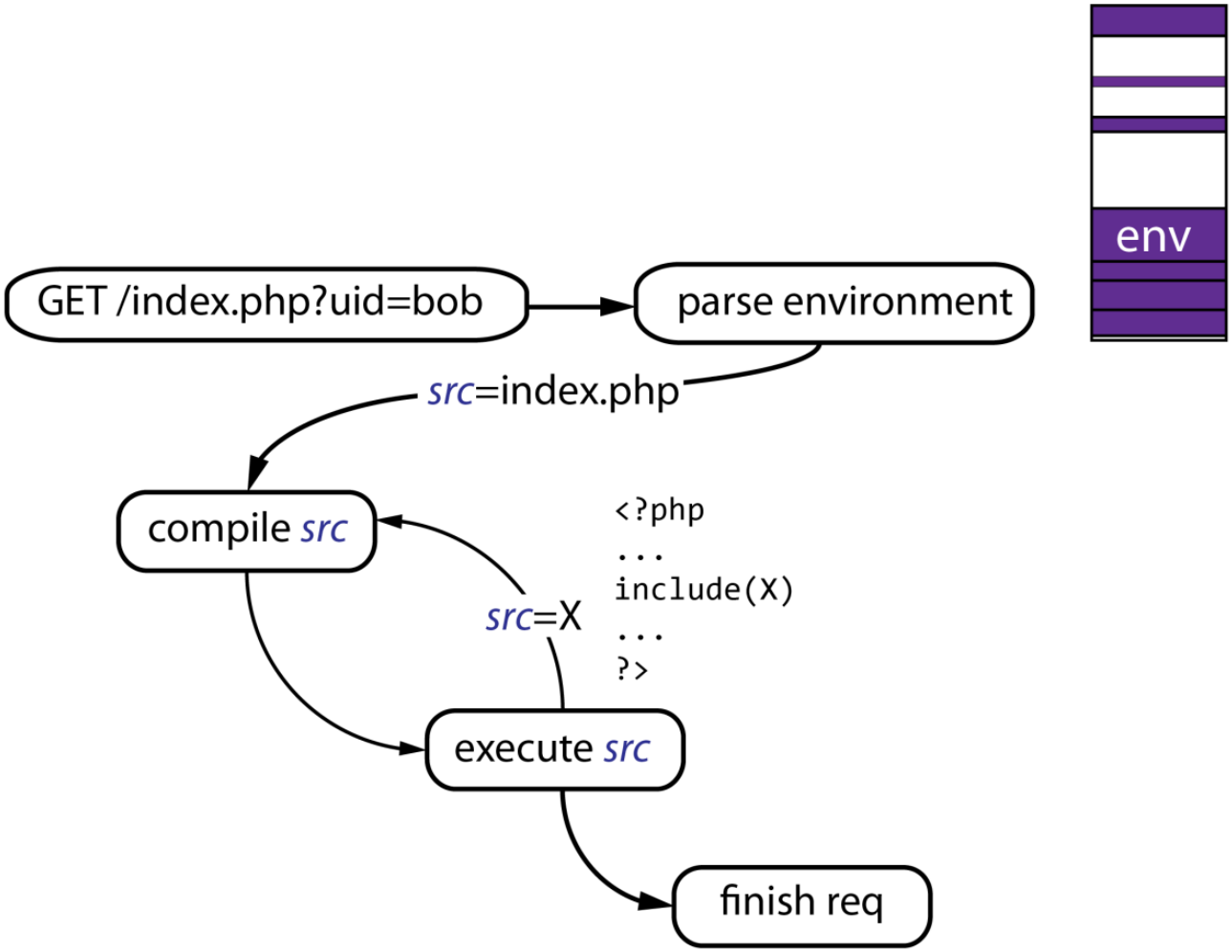
PHP Execution



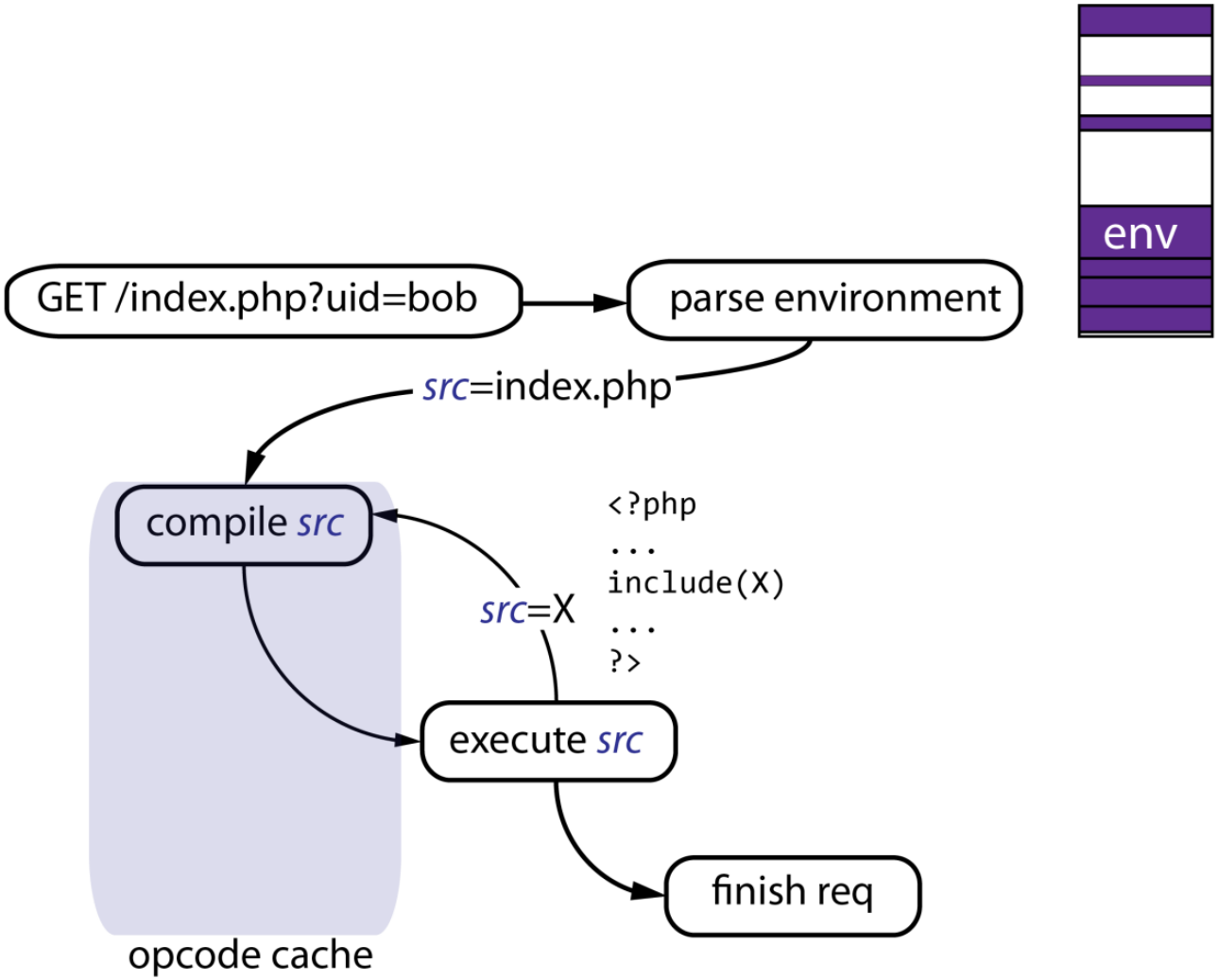
PHP Execution



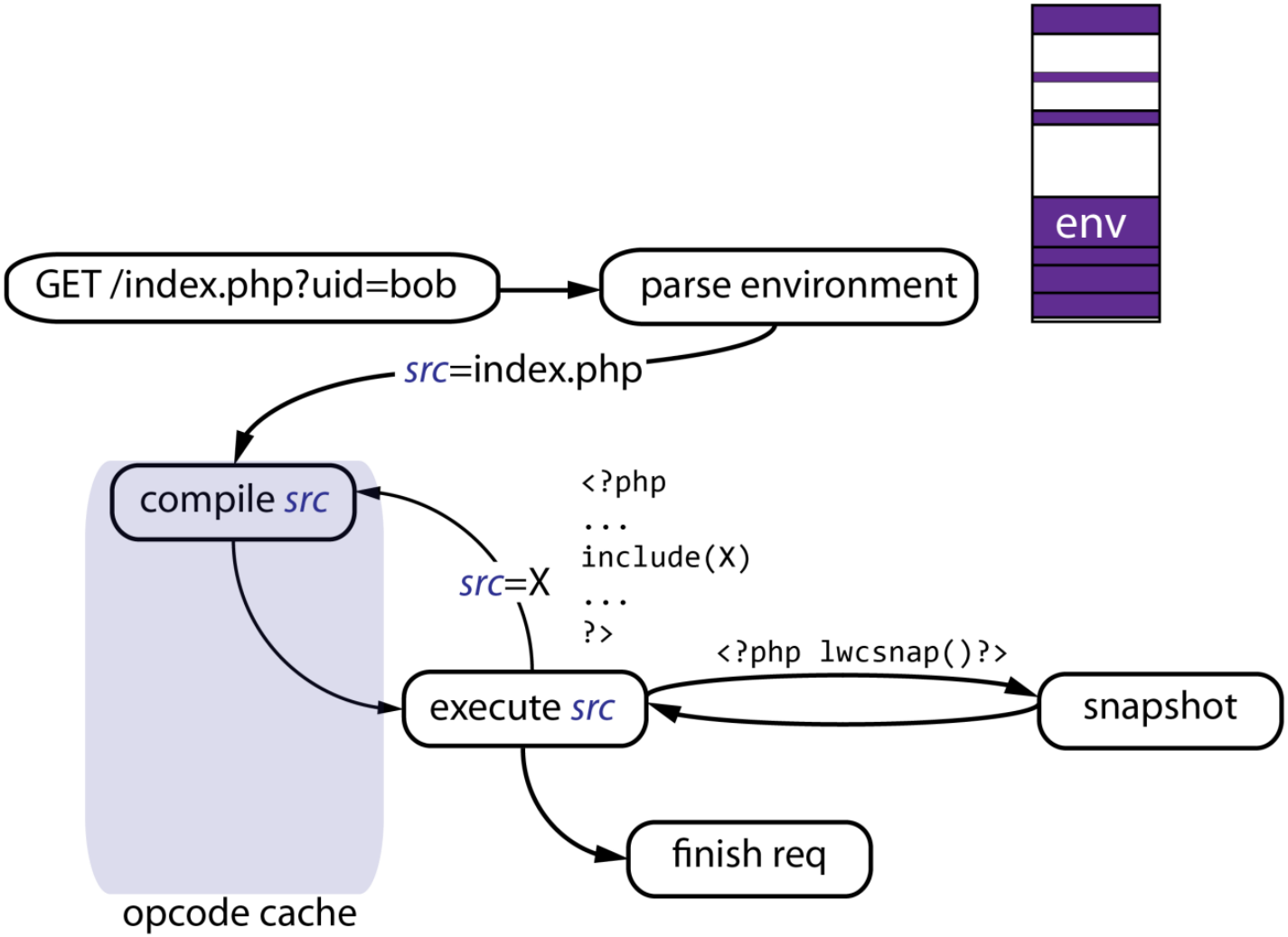
PHP Execution



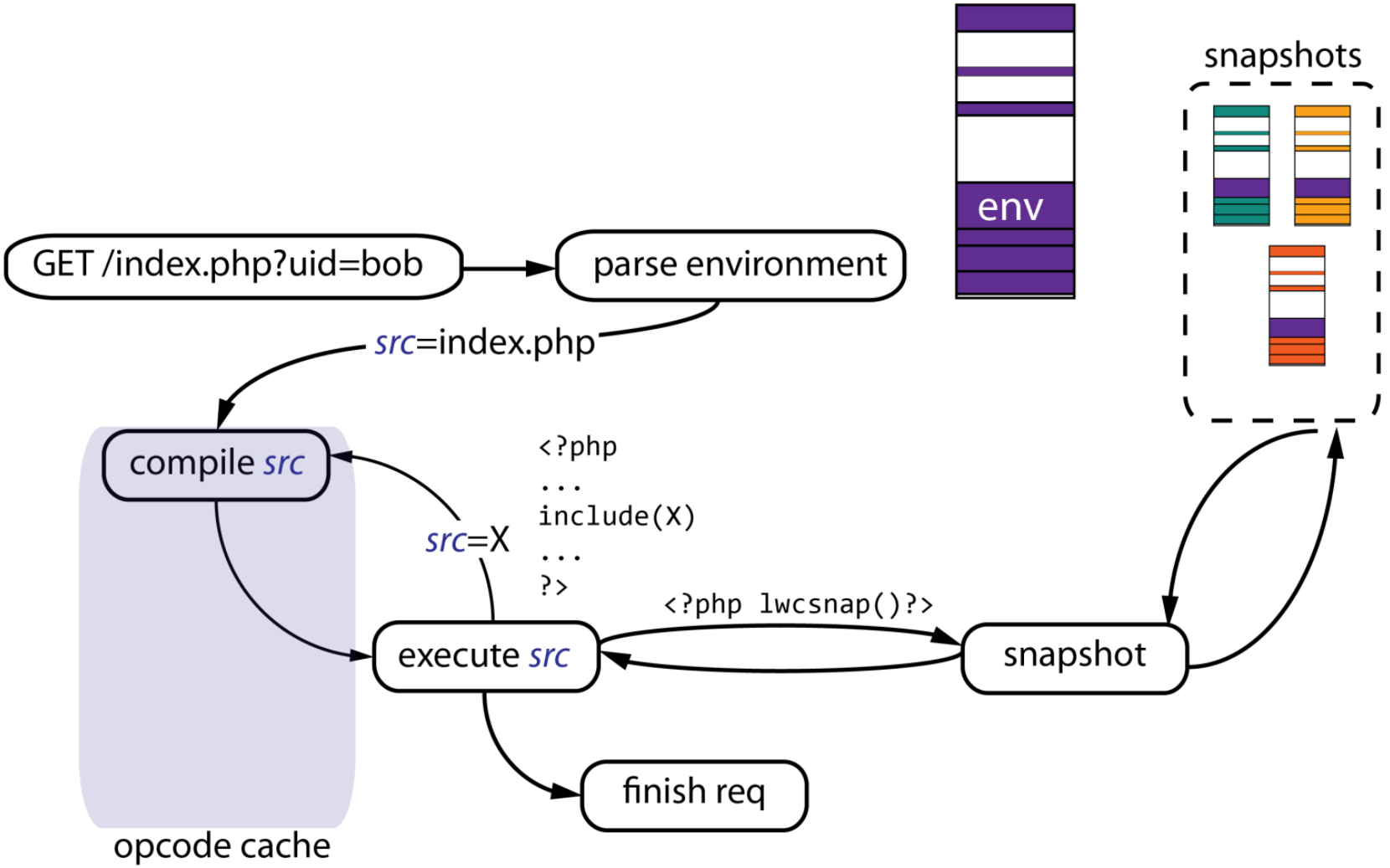
PHP Execution



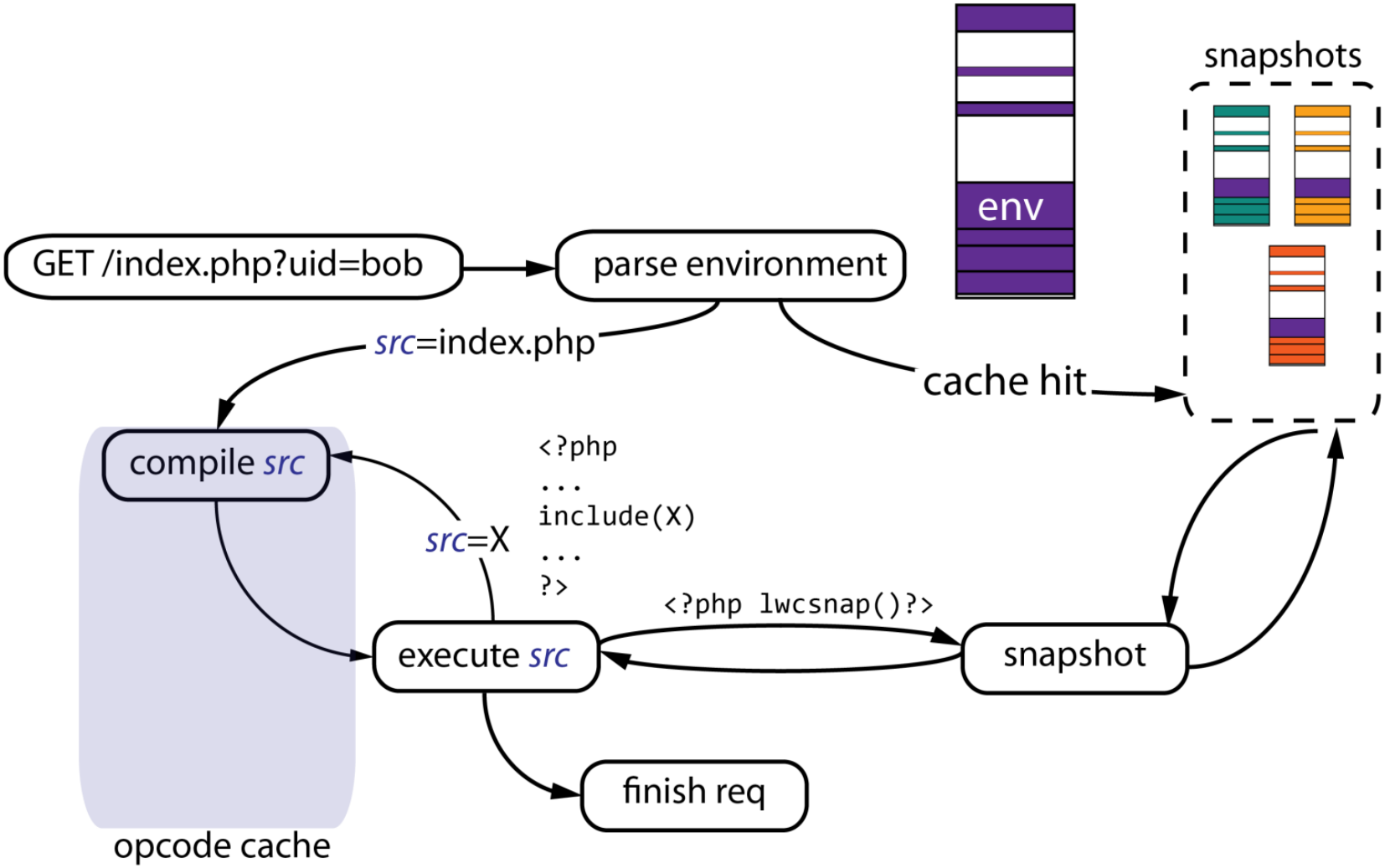
PHP Fast Startup



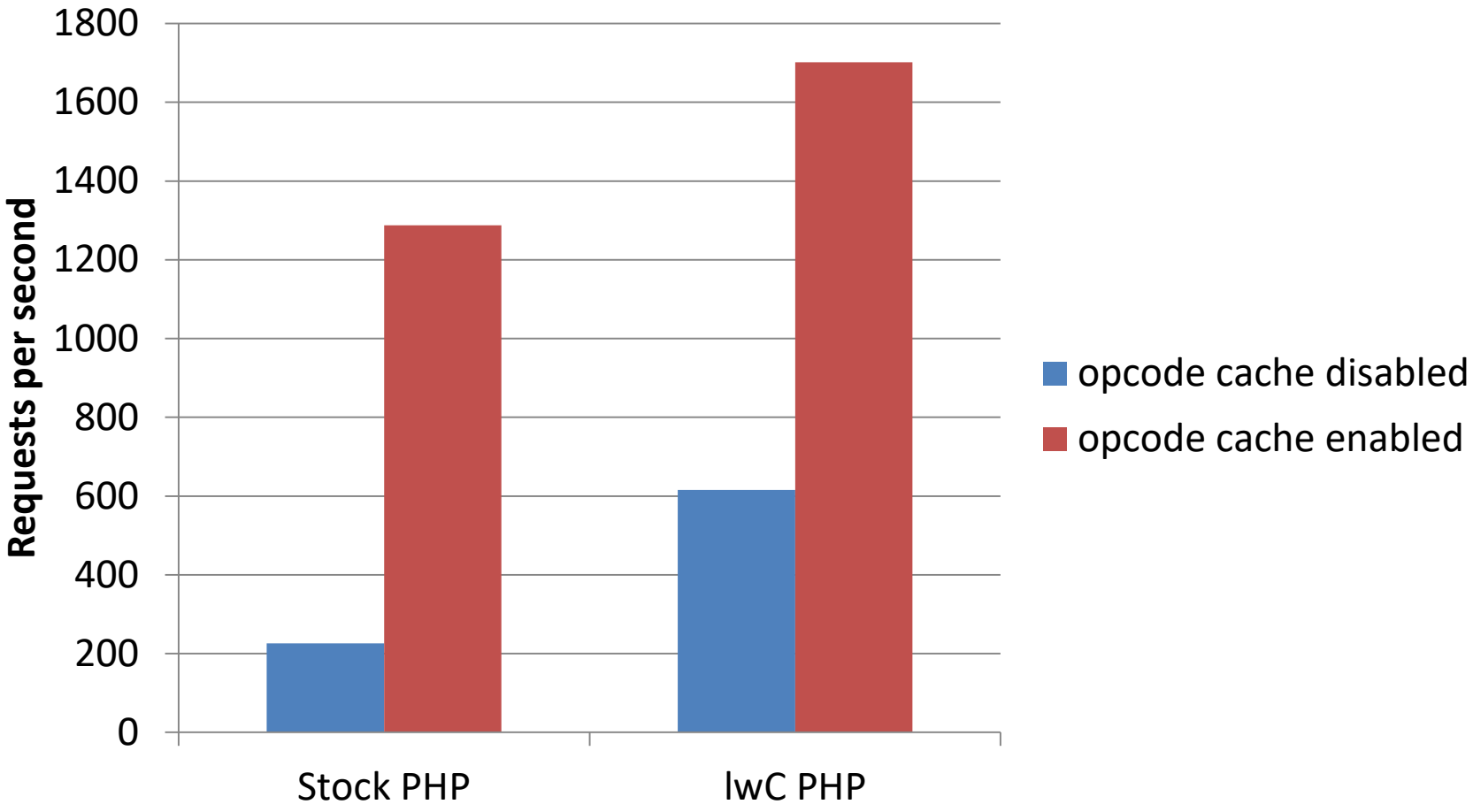
PHP Fast Startup



PHP Fast Startup



Evaluation - PHP-FCGI



Light-Weight Contexts

Source code will be available at
http://www.cs.umd.edu/projects/lwc/



Max
Planck
Institute
for
Software Systems

