Fiji Priority Rollback Protocol

Lukasz Ziarek

Fiji Systems Inc.

Filip Pizlo, Ethan Blanton, and Jan Vitek

Fiji

Systems LLC

Communication between mixed-criticality partitions

Situational Awareness

Flight System

Communication between mixed-criticality partitions

Situational Awareness

Middleware

Flight System

Fiji
Systems LLC

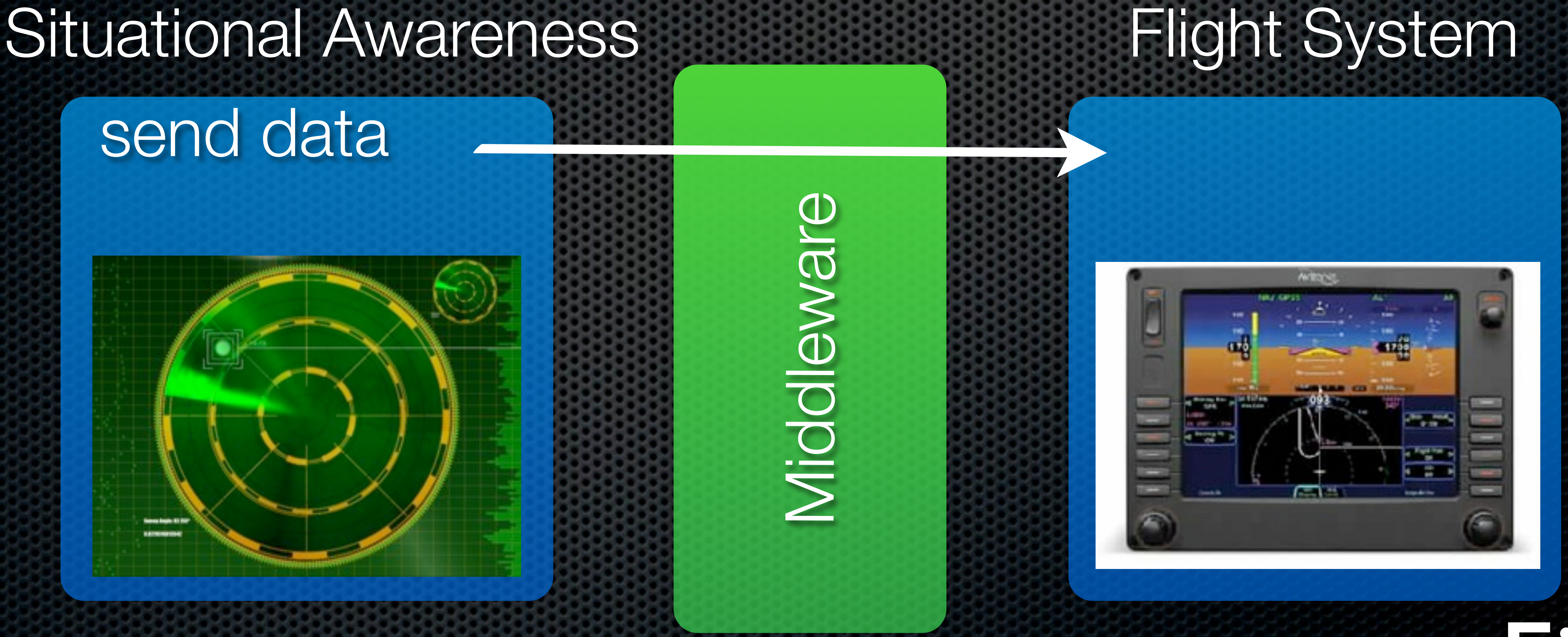
Communication between mixed-criticality partitions

Situational Awareness package data

Middleware

Flight System

Communication between mixed-criticality partitions



Situational Awareness

send data

Middleware

Flight System

Fiji

Systems LLC

Communication between mixed-criticality partitions

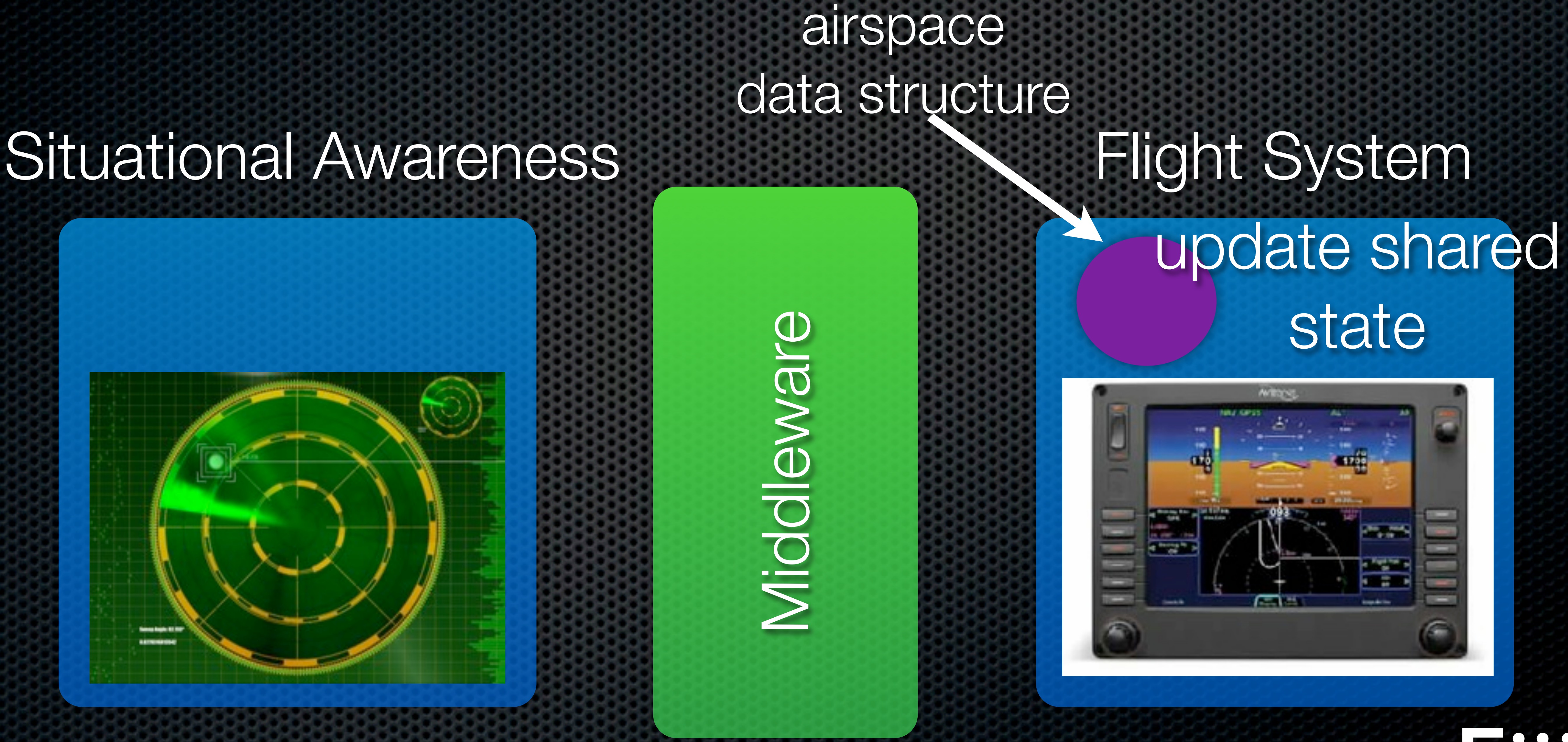
Situational Awareness

Middleware

Flight System

unpack data

Communication between mixed-criticality partitions



airspace
data structure

Situational Awareness

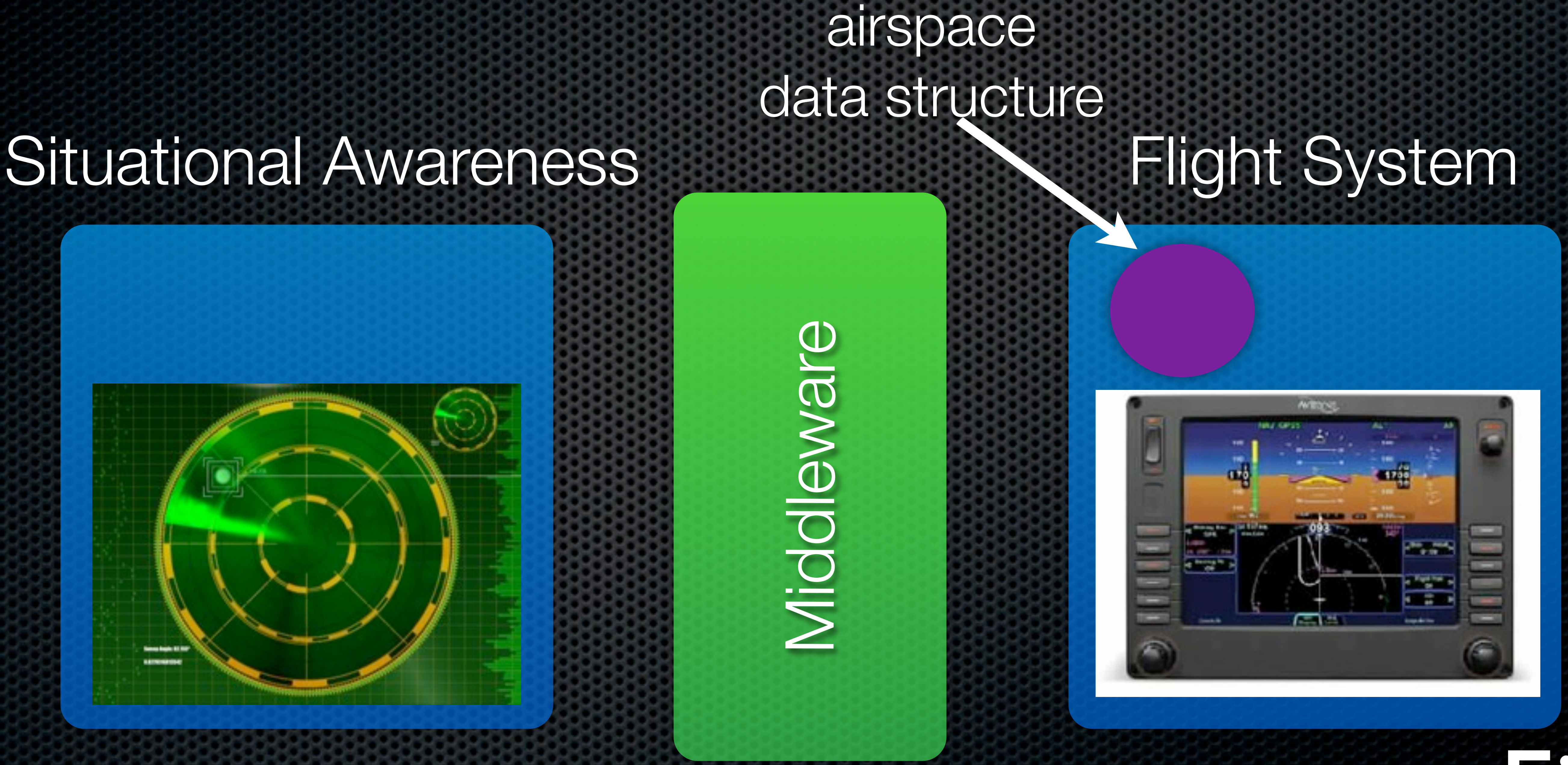
Middleware

Flight System

update shared
state

Fiji
Systems LLC

Communication between mixed-criticality partitions



airspace
data structure

Situational Awareness

Middleware

Flight System

typically a communication protocol

Fiji
Systems LLC

Communication between mixed-criticality partitions

Situational Awareness

Flight System

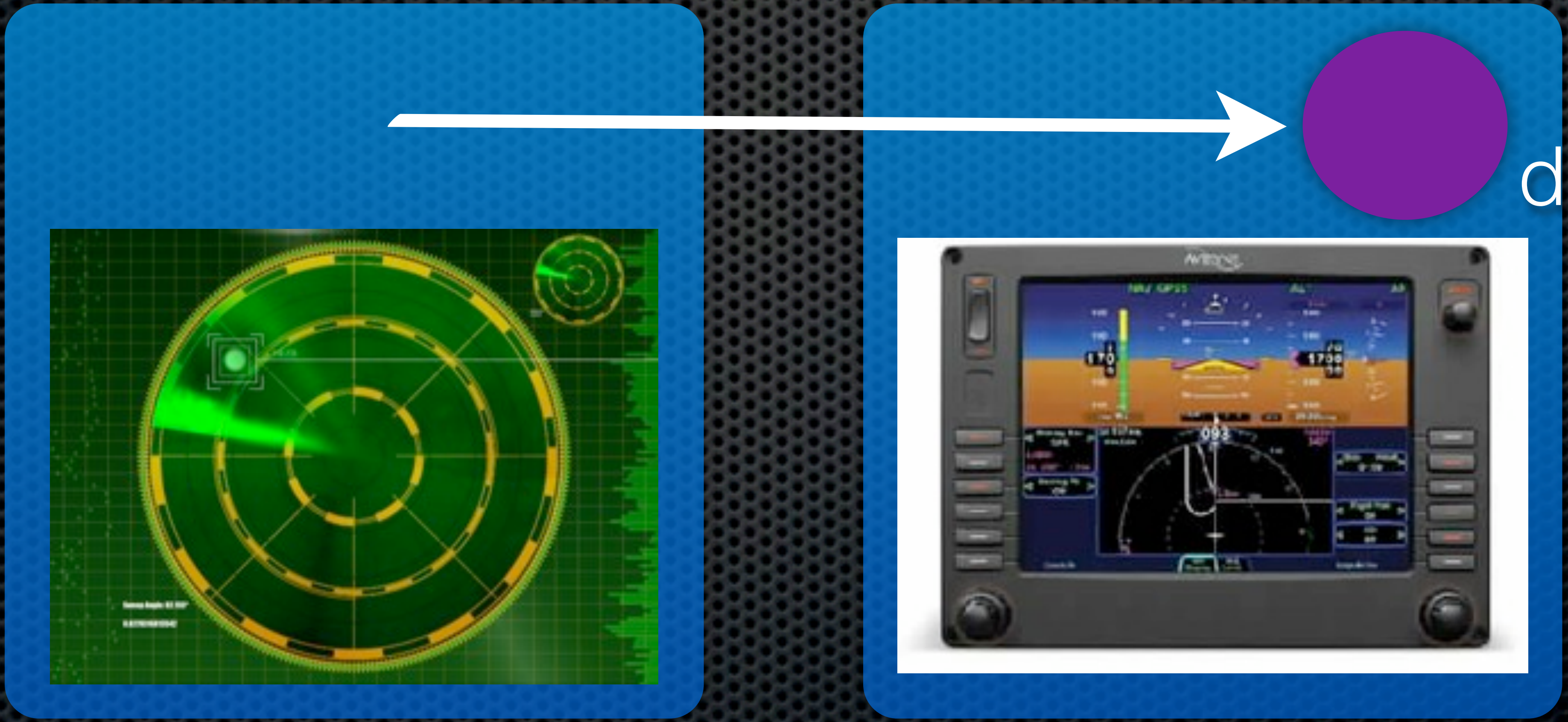
airspace data structure

Communication between mixed-criticality partitions

can we allow direct access?

Situational Awareness

Flight System



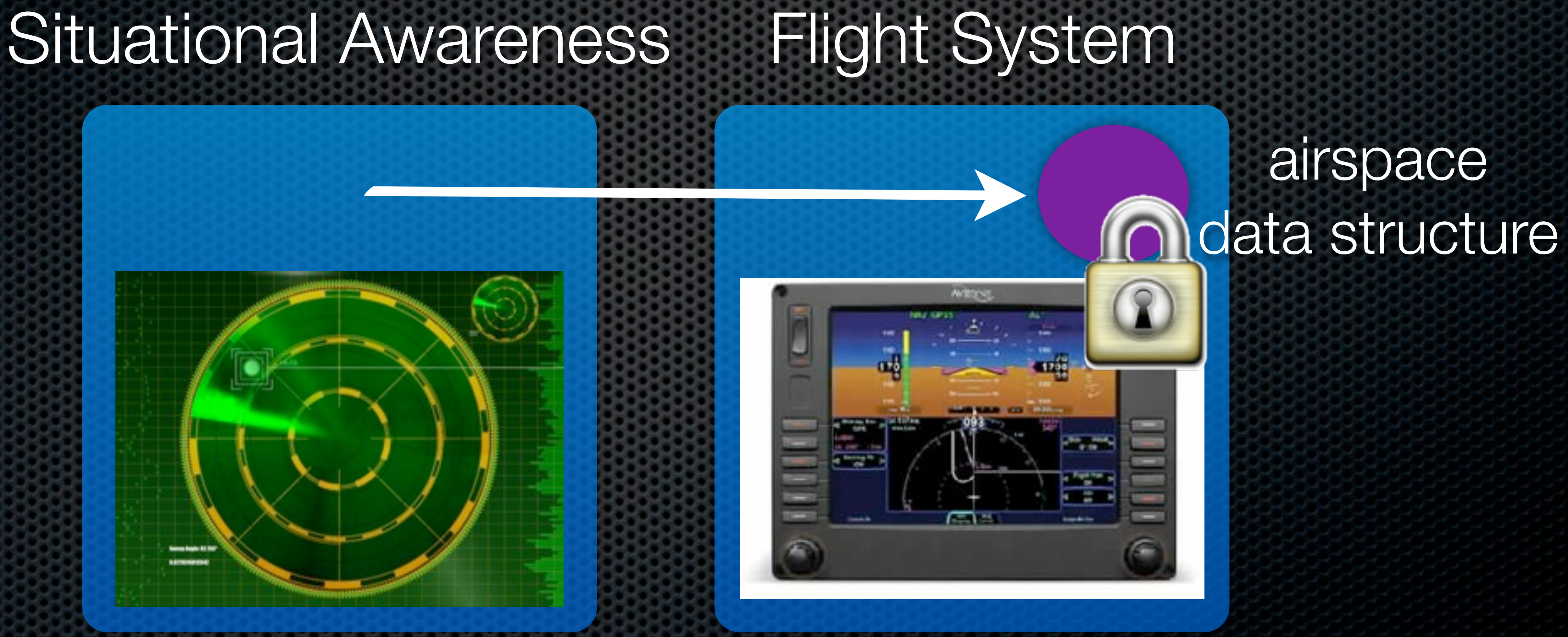
airspace data structure

Fiji

Systems LLC

Communication between mixed-criticality partitions

mediate access via a lock



Situational Awareness

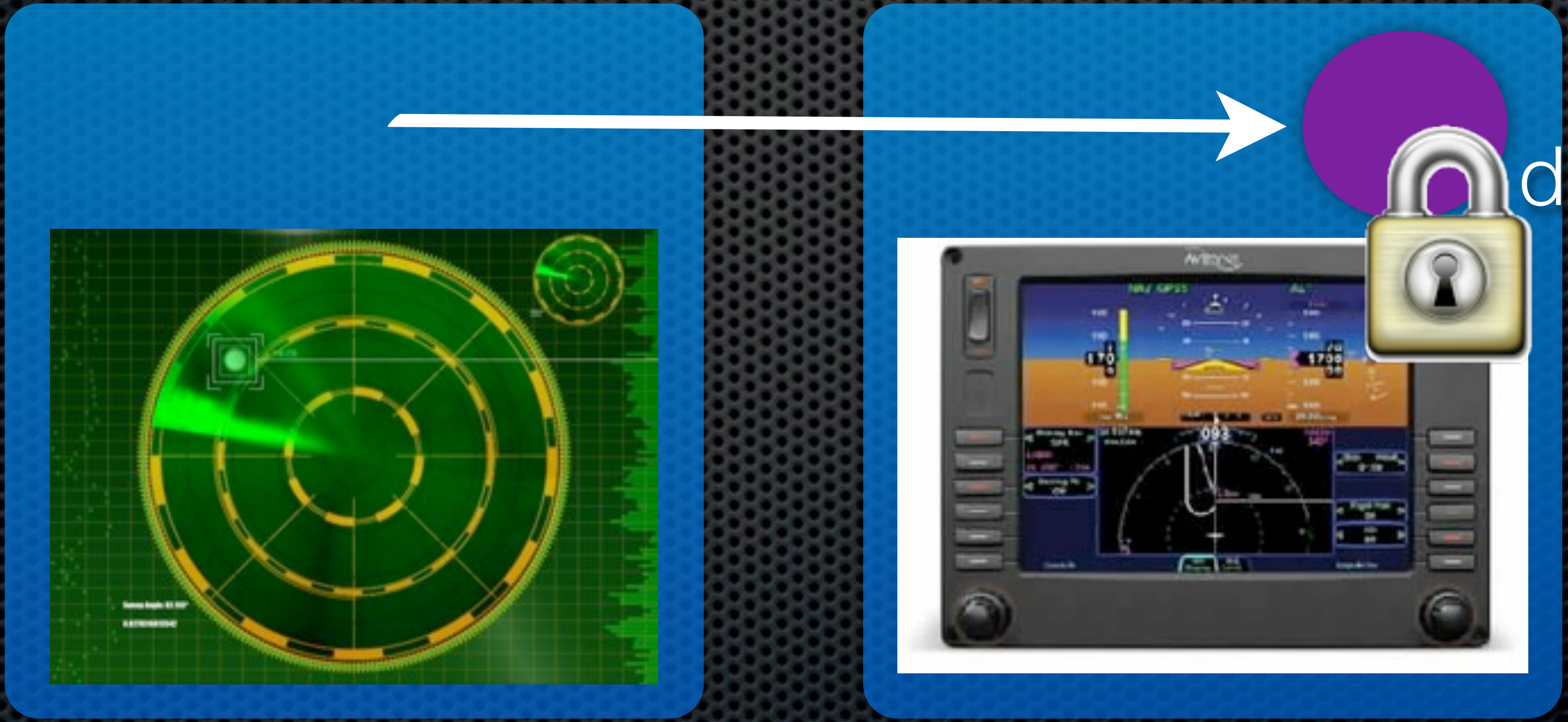
Flight System

airspace data structure

Communication between mixed-criticality partitions

Situational Awareness

Flight System



airspace data structure

how do we guarantee responsiveness of higher criticality partitions?

Communication between mixed-criticality partitions

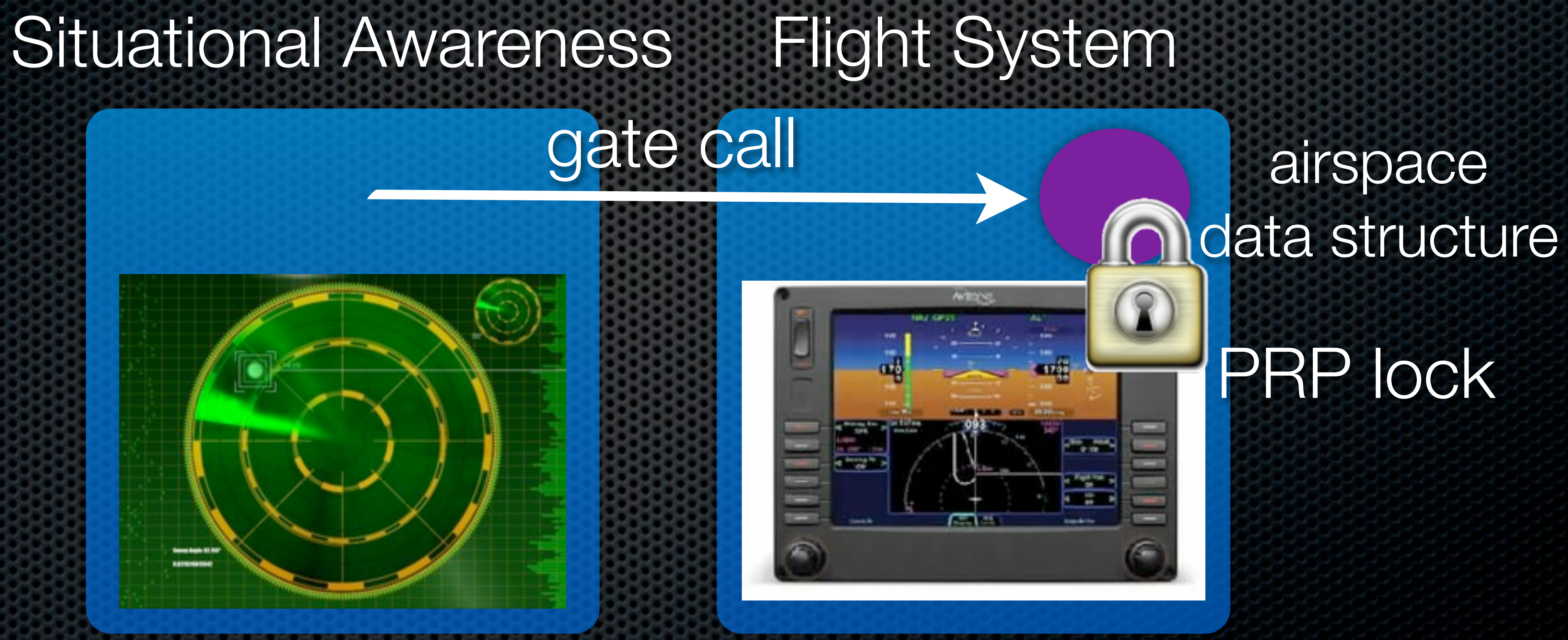
Situational Awareness

Flight System

airspace data structure

Fiji Systems LLC

Communication between mixed-criticality partitions



Situational Awareness

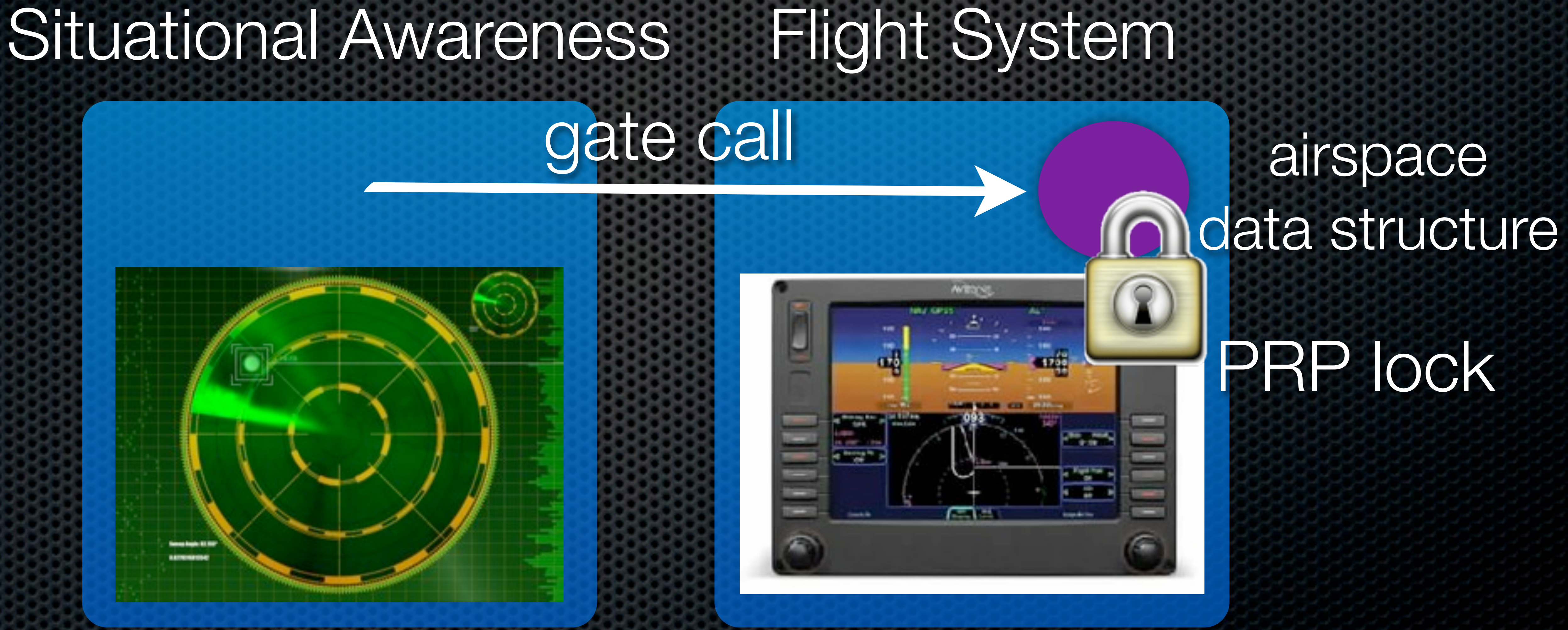
Flight System

gate call

airspace data structure

PRP lock

Communication between mixed-criticality partitions



Situational Awareness

Flight System

gate call

airspace data structure

PRP lock

PRP allows for priority aware, criticality aware, safe, reliable, shared memory between partitions

PRP Locks

- Flight System’s partition is guaranteed fast
- Bound on preemption based on data structure size
- Situational awareness' partition access is slightly slower but is still bounded in time

Inspiration from Transactions

- Atomic replacement for locks
- Automatic serializability detection
- Runtime monitoring
- Aborts - ability to rollback

Fiji
Systems LLC

Inspiration from Transactions

- Atomic rep✗cement for locks
- Automatic serializability detection
- Runtime monitoring
- Aborts - ability to rollback

New programming model

Inspiration from Transactions

- ~~Atomic replacement for locks~~
- ~~Automatic serializability detection~~
- Runtime monitoring
- Aborts - ability to rollback

New programming model

Unpredictable

Inspiration from Transactions

- Atomic replacement for locks ✗
- Automatic serializability detection ✗
- Runtime monitoring ✓
- Aborts - ability to rollback

New programming model

Unpredictable

Fiji Systems LLC

Inspiration from Transactions

- Atomic replacement for locks ✗
- Automatic serializability detection ✗
- Runtime monitoring ✓
- Aborts - ability to rollback ✓

New programming model

Unpredictable

Fiji
Systems LLC

PRP: Two Options

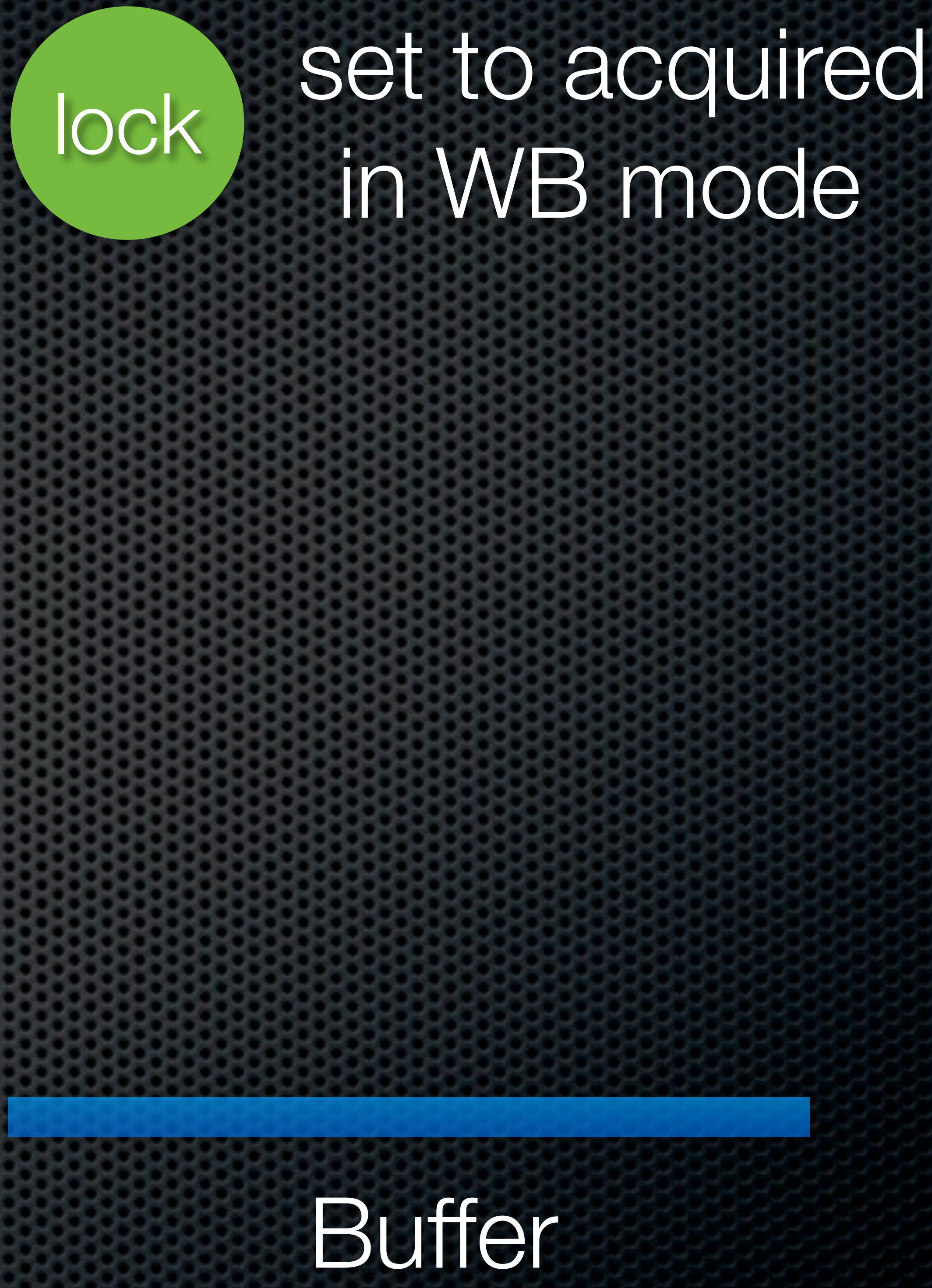
- Write Buffering
  - All updates buffered : memory is always consistent
- Write Logging
  - Updates to shared memory : undo log allows reversion to consistent state of memory

Write Buffering

```
synchronized(lock){
    foo.a = x;
    foo.b = y;
    foo.c = z;
    foo.a = w;
    if(foo.b+4 > foo.a)
        ...
}
```

Write Buffering

```
synchronized(lock){
    foo.a = x;
    foo.b = y;
    foo.c = z;
    foo.a = w;
    if(foo.b+4 > foo.a)
        ...
}
```

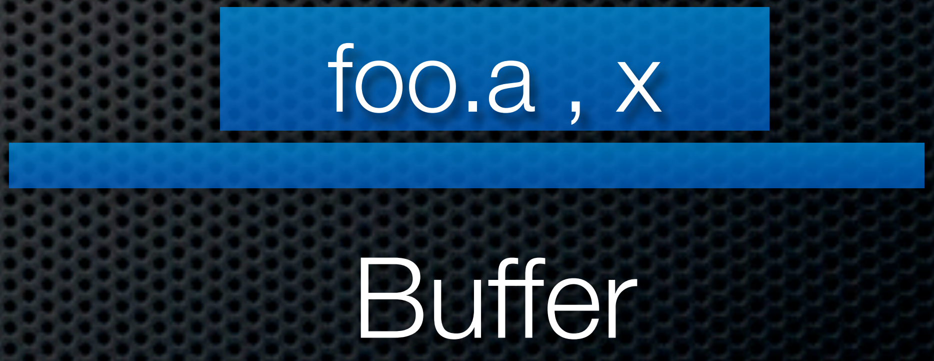


Write Buffering

lock

set to acquired
in WB mode

```
synchronized(lock){
  foo.a = x;
  foo.b = y;
  foo.c = z;
  foo.a = w;
  if(foo.b+4 > foo.a)
      ...
}
```

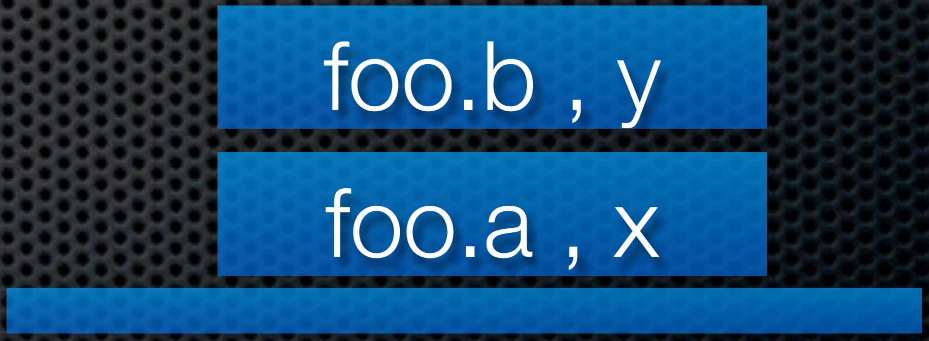


Write Buffering

lock

set to acquired
in WB mode

```
synchronized(lock){
   foo.a = x;
   foo.b = y;
   foo.c = z;
   foo.a = w;
   if(foo.b+4 > foo.a)
      ...
}
```



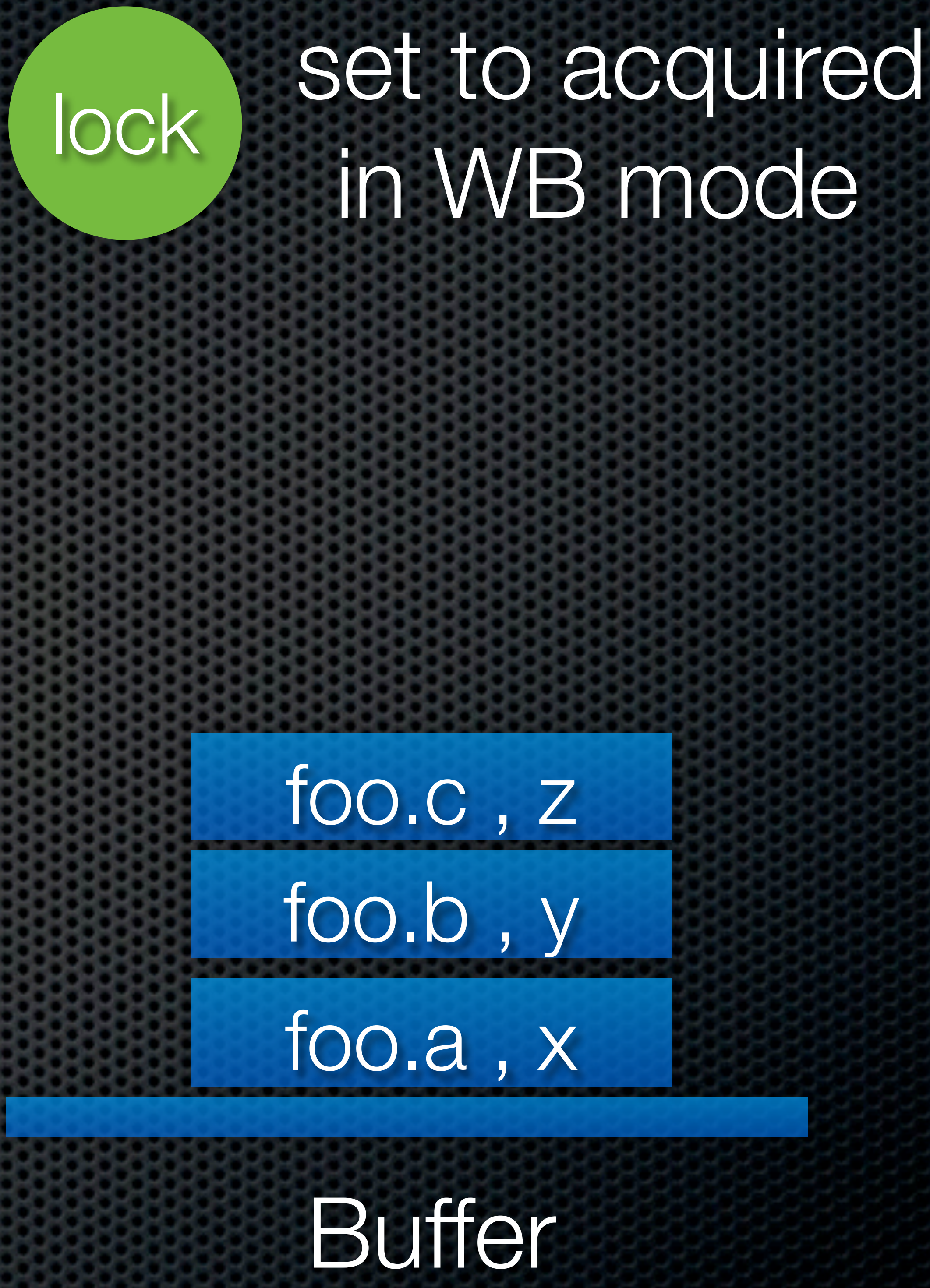
foo.b , y

foo.a , x

Buffer

Write Buffering

```
synchronized(lock){
  foo.a = x;
  foo.b = y;
  foo.c = z;
  foo.a = w;
  if(foo.b+4 > foo.a)
      ...
}
```



lock

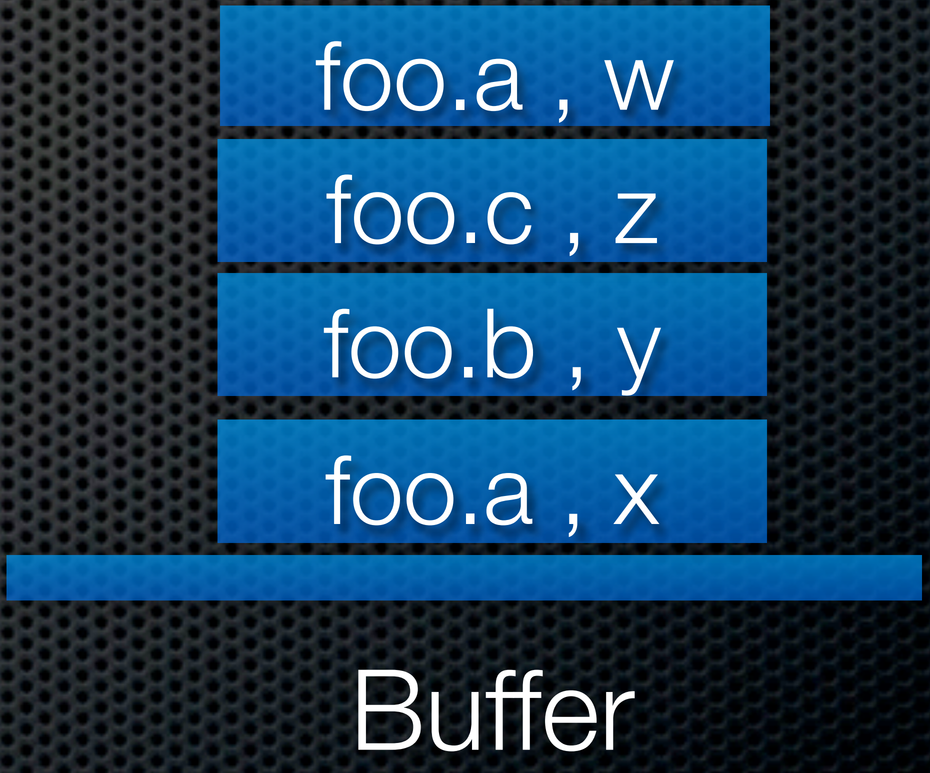
set to acquired in WB mode

Write Buffering

lock

set to acquired
in WB mode

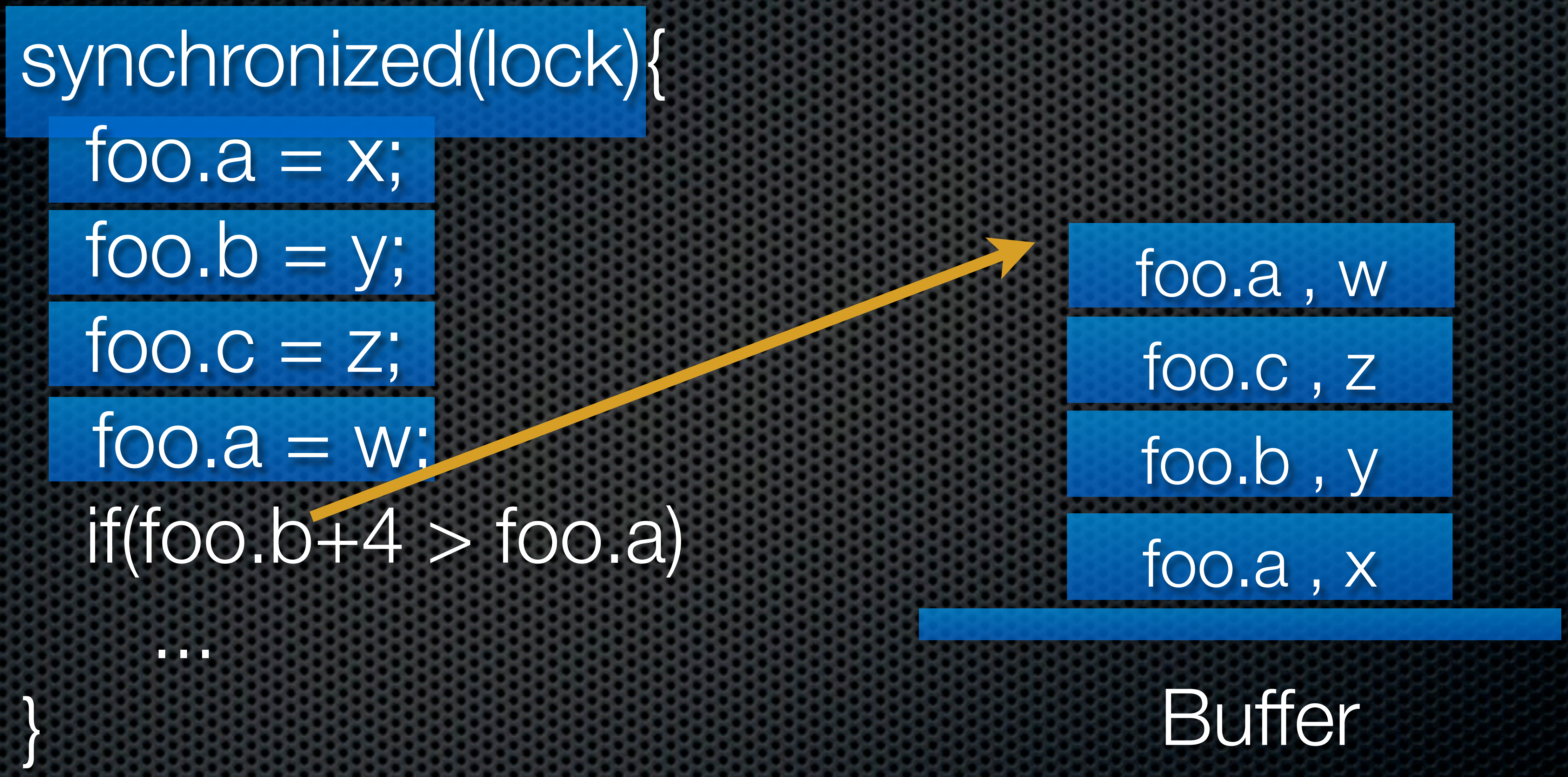
```
synchronized(lock){
    foo.a = x;
    foo.b = y;
    foo.c = z;
    foo.a = w;
    if(foo.b+4 > foo.a)
        ...
}
```



Write Buffering

lock

set to acquired
in WB mode



```
synchronized(lock){
  foo.a = x;
  foo.b = y;
  foo.c = z;
  foo.a = w;
  if(foo.b+4 > foo.a)
     ...
}
```

foo.a , w
foo.c , z
foo.b , y
foo.a , x

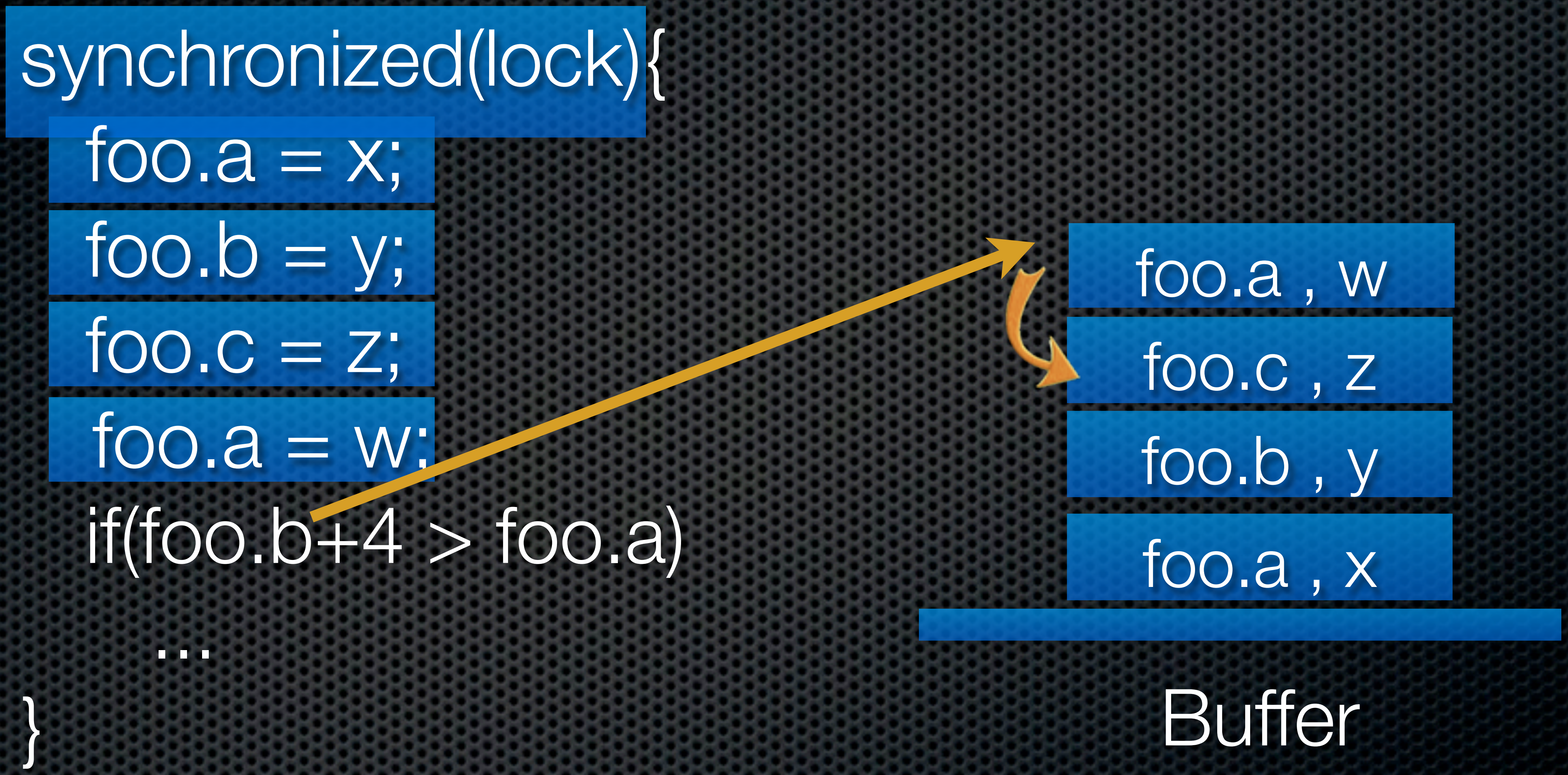
Buffer

Fiji
Systems LLC

Write Buffering

lock

set to acquired
in WB mode



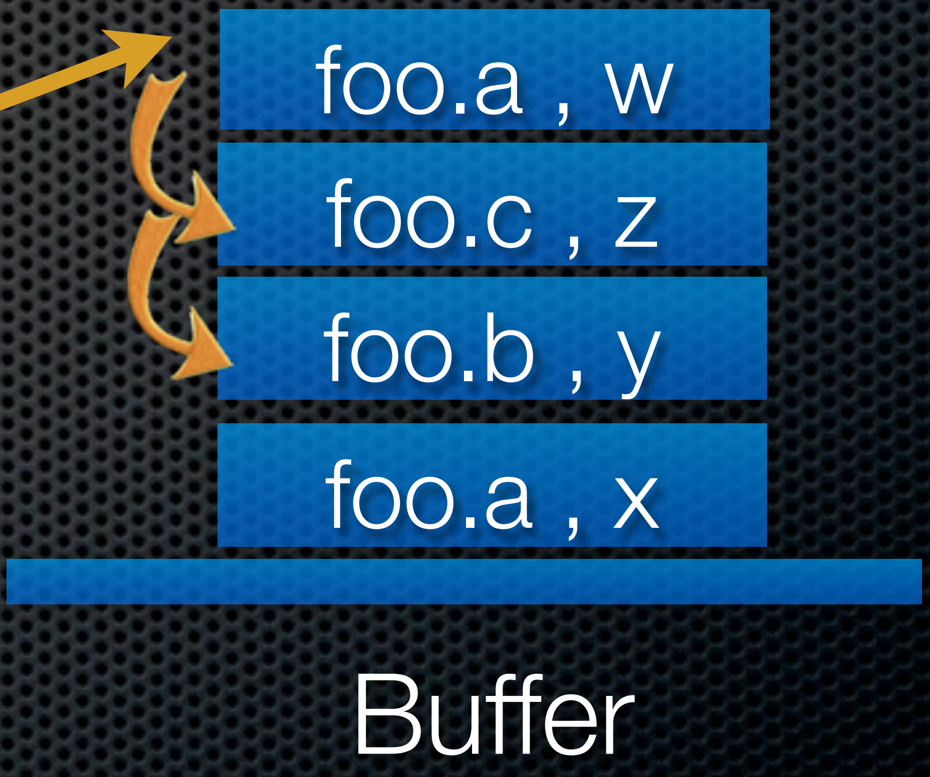
```
synchronized(lock){
   foo.a = x;
   foo.b = y;
   foo.c = z;
   foo.a = w;
   if(foo.b+4 > foo.a)
      ...
}
```

Write Buffering

lock

set to acquired
in WB mode

```
synchronized(lock){
   foo.a = x;
   foo.b = y;
   foo.c = z;
   foo.a = w;
   if(foo.b+4 > foo.a)
       ...
}
```



foo.a , w

foo.c , z

foo.b , y

foo.a , x

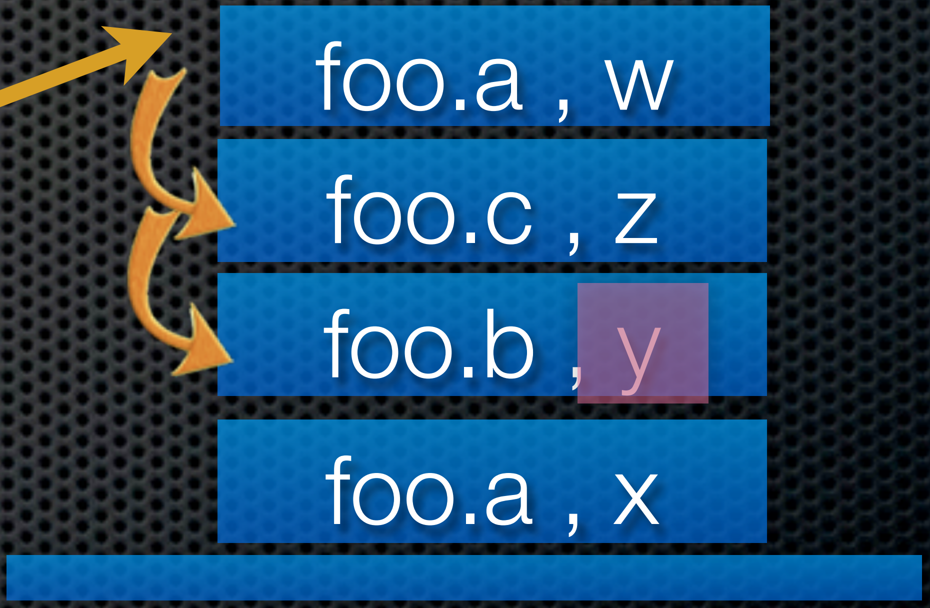
Buffer

Write Buffering

lock

set to acquired in WB mode

```
synchronized(lock){
   foo.a = x;
   foo.b = y;
   foo.c = z;
   foo.a = w;
   if(foo.b+4 > foo.a)
      ...
}
```



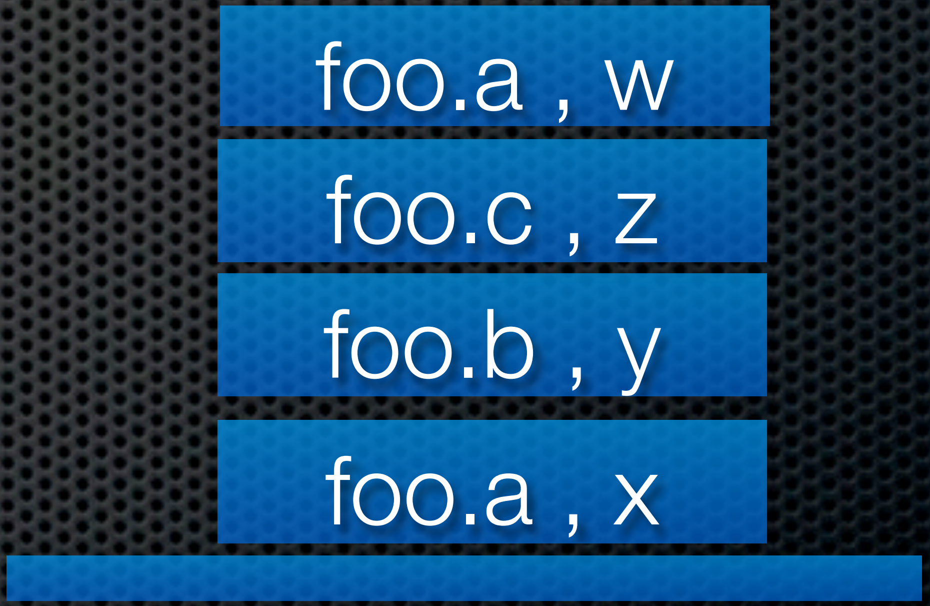
Buffer

Write Buffering

lock

set to acquired
in WB mode

```
synchronized(lock){
    foo.a = x;
    foo.b = y;
    foo.c = z;
    foo.a = w;
    if(foo.b+4 > foo.a)
        ...
}
```



foo.a , w

foo.c , z

foo.b , y

foo.a , x

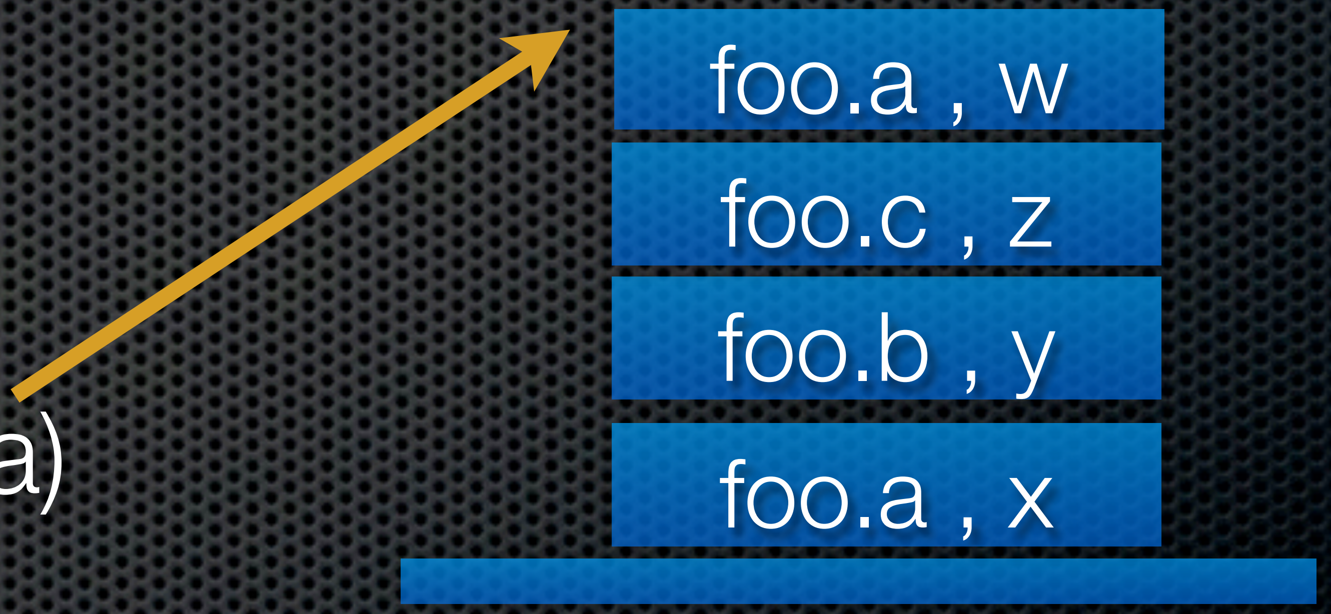
Buffer

Write Buffering

lock

set to acquired
in WB mode

```
synchronized(lock){
  foo.a = x;
  foo.b = y;
  foo.c = z;
  foo.a = w;
  if(foo.b+4 > foo.a)
      ...
}
```



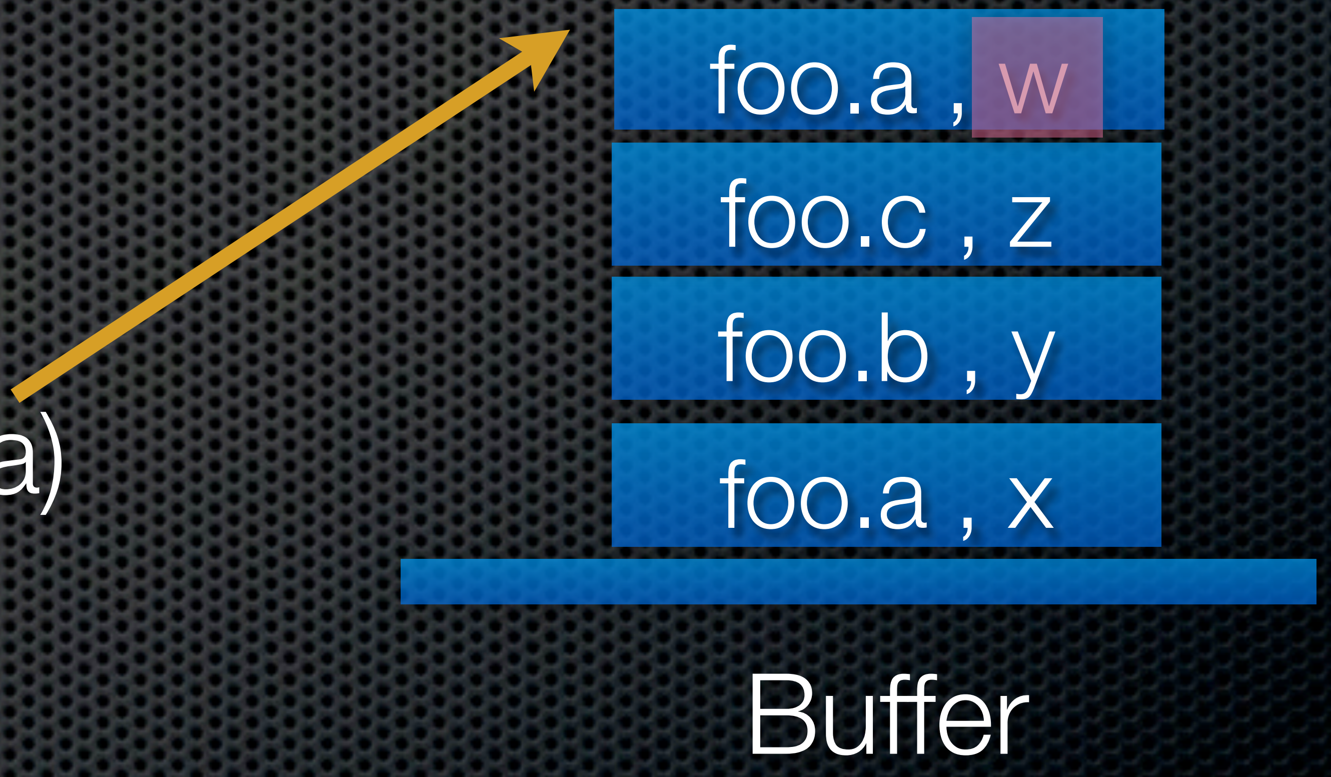
Buffer

Write Buffering

lock

set to acquired
in WB mode

```
synchronized(lock){
  foo.a = x;
  foo.b = y;
  foo.c = z;
  foo.a = w;
  if(foo.b+4 > foo.a)
     ...
}
```



foo.a , w

foo.c , z

foo.b , y

foo.a , x

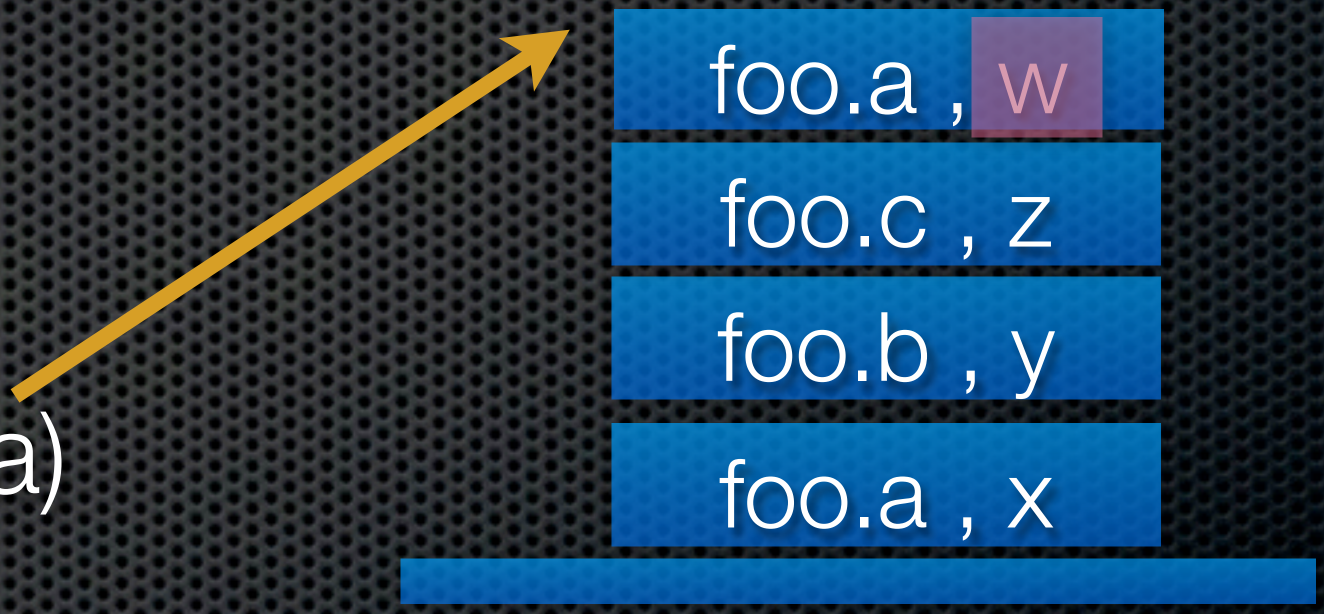
Buffer

Write Buffering

lock

set to acquired
in WB mode

```
synchronized(lock){
  foo.a = x;
  foo.b = y;
  foo.c = z;
  foo.a = w;
  if(foo.b+4 > foo.a)
     ...
} Commit
```



foo.a , w
foo.c , z
foo.b , y
foo.a , x

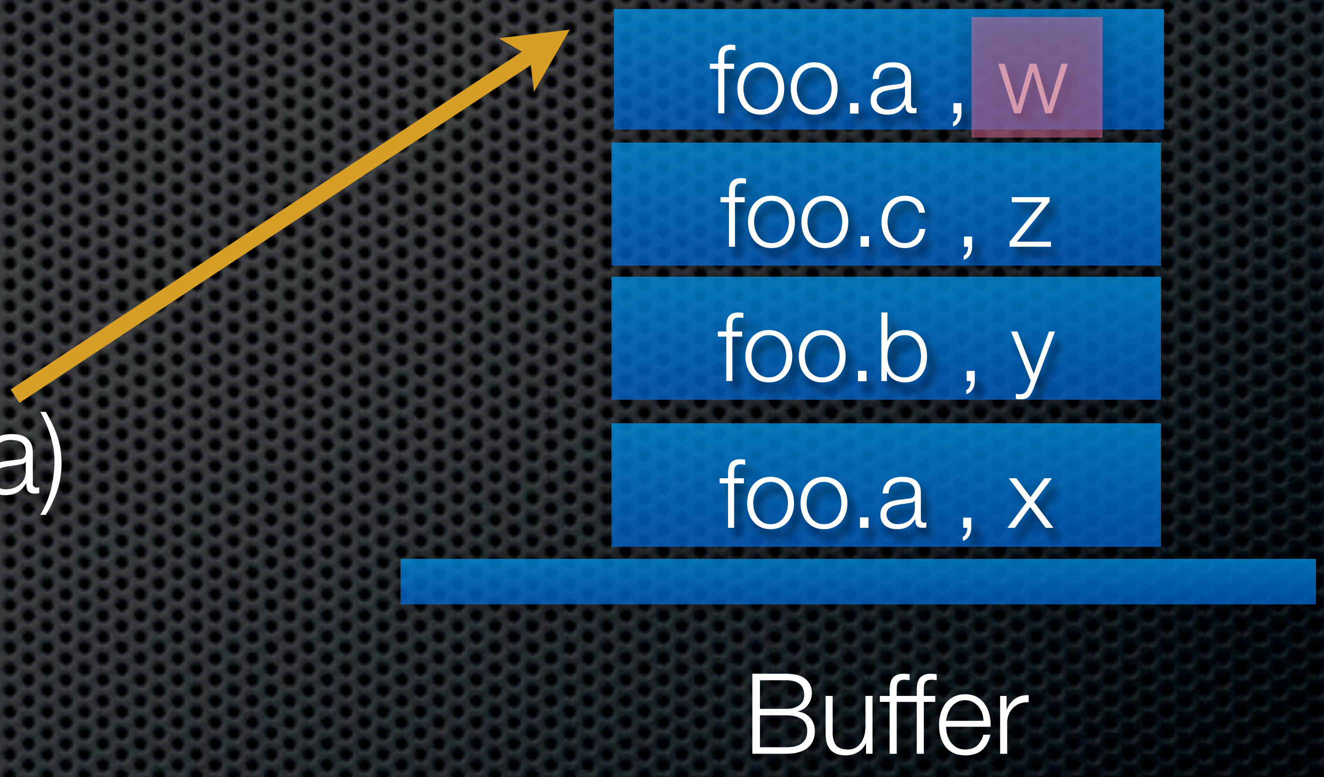
Buffer

Write Buffering

lock

set to acquired
in WB mode

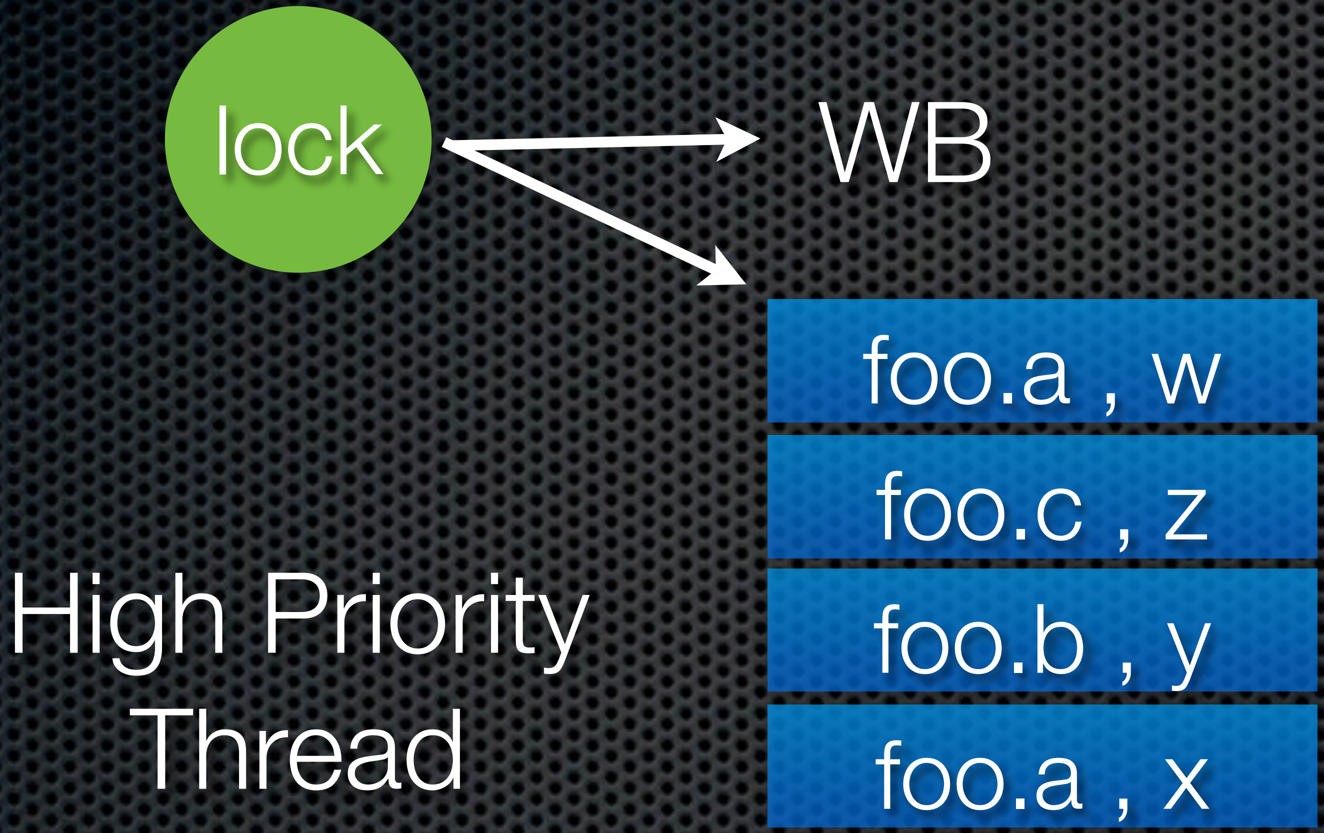
```
synchronized(lock){
   foo.a = x;
   foo.b = y;
   foo.c = z;
   foo.a = w;
   if(foo.b+4 > foo.a)
      ...
} Commit
```


foo.a , w
foo.c , z
foo.b , y
foo.a , x
Buffer

Set lock mode to commit

Fiji
Systems LLC

Acquisition by higher priority thread



lock

WB

foo.a , w

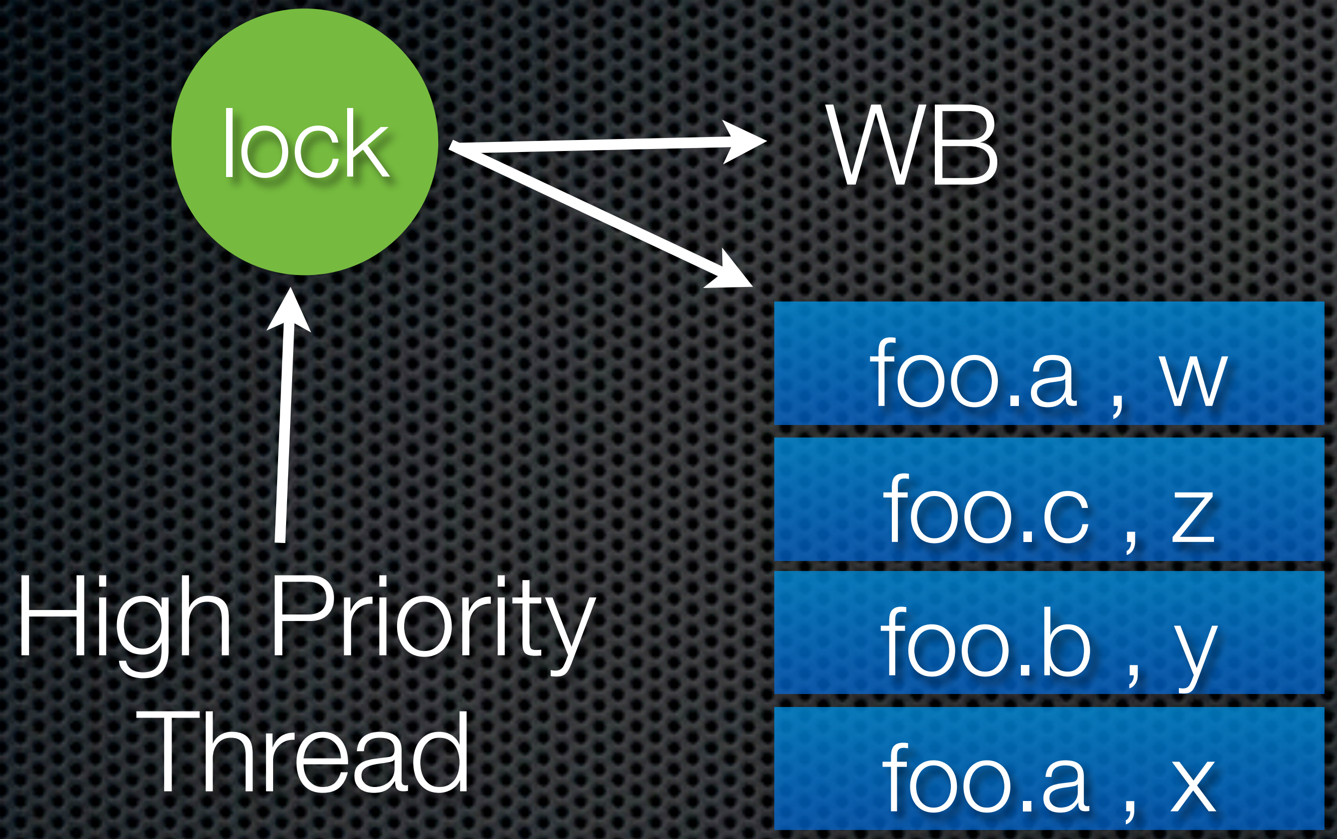
foo.c , z

foo.b , y

foo.a , x

High Priority Thread

Acquisition by higher priority thread



lock

WB

foo.a , w

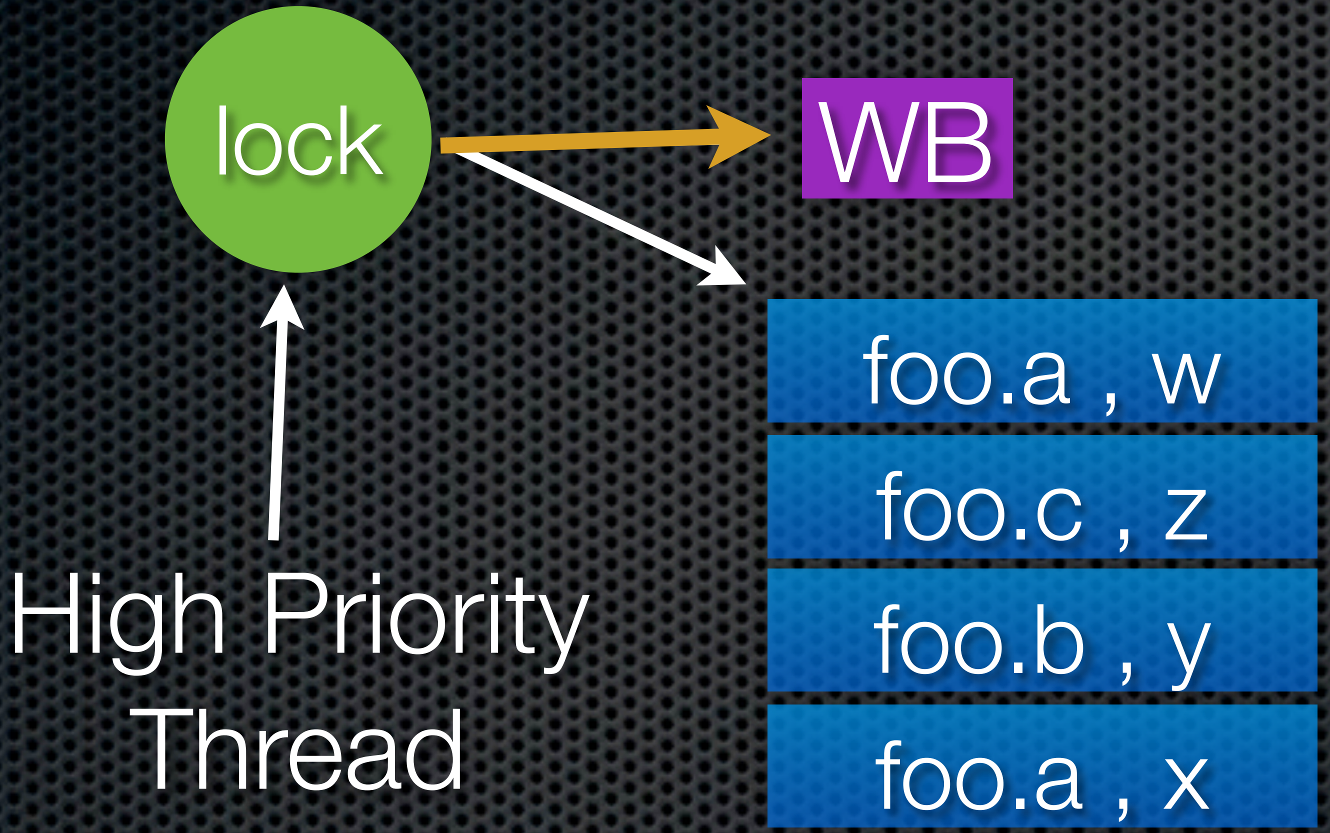
foo.c , z

foo.b , y

foo.a , x

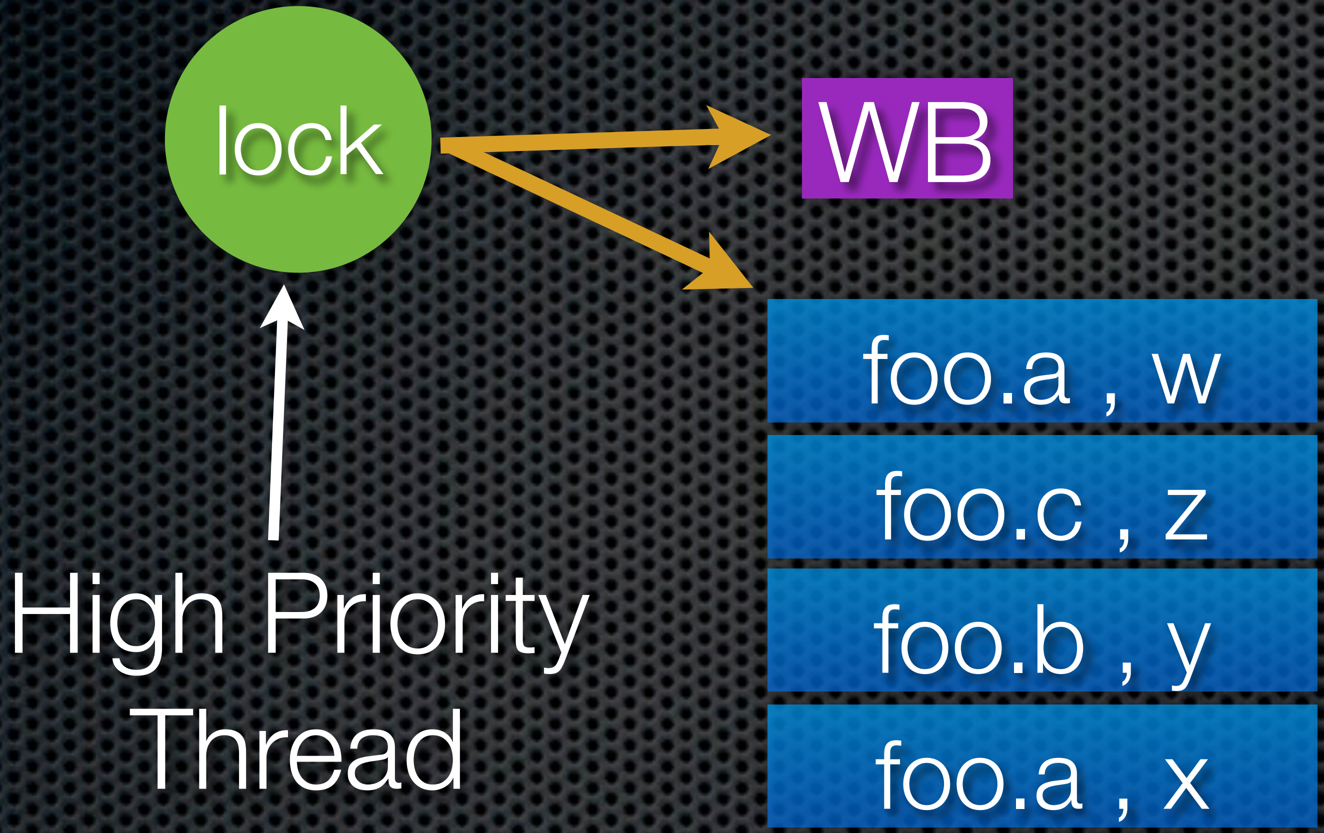
High Priority Thread

Acquisition by higher priority thread



Fiji
Systems LLC

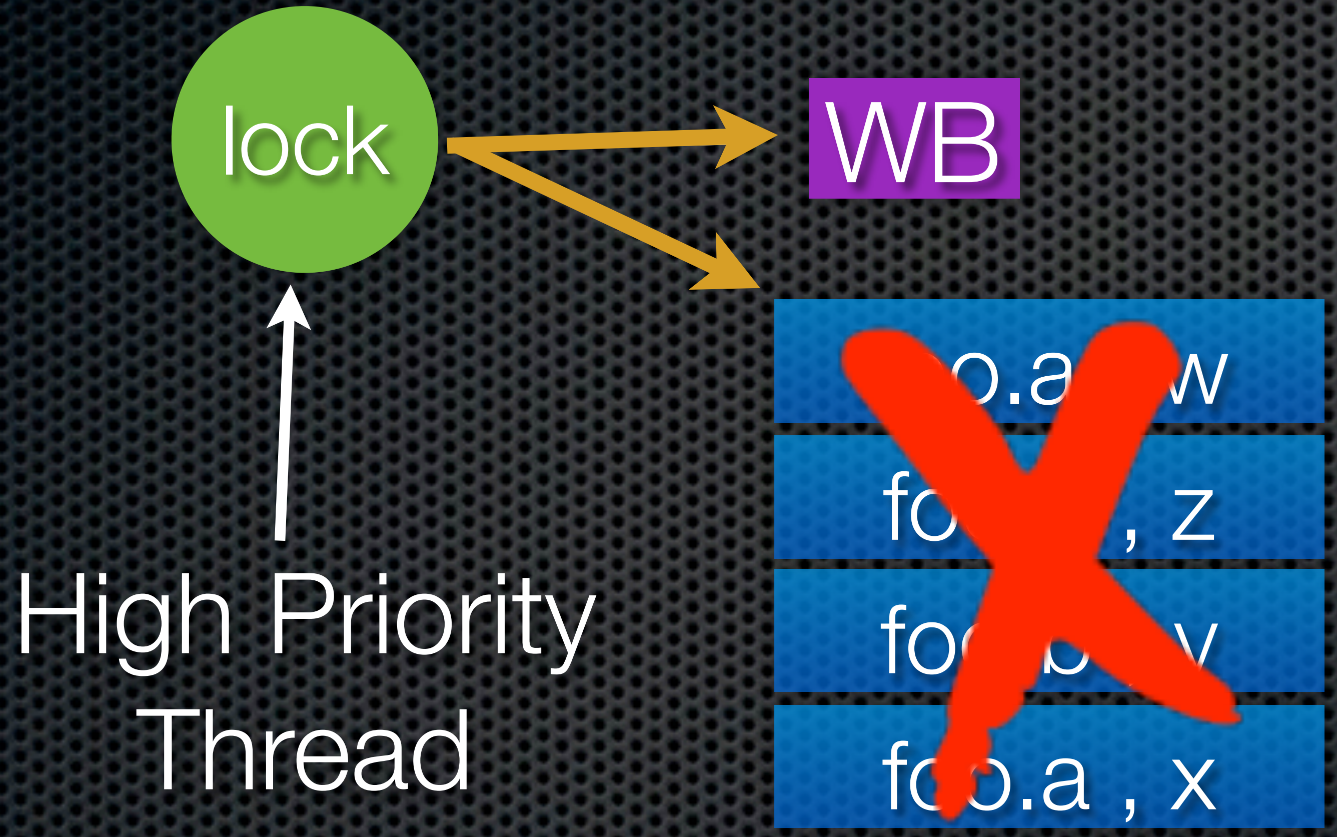
Acquisition by higher priority thread



Fiji

Systems LLC

Acquisition by higher priority thread



lock

WB

o.a w

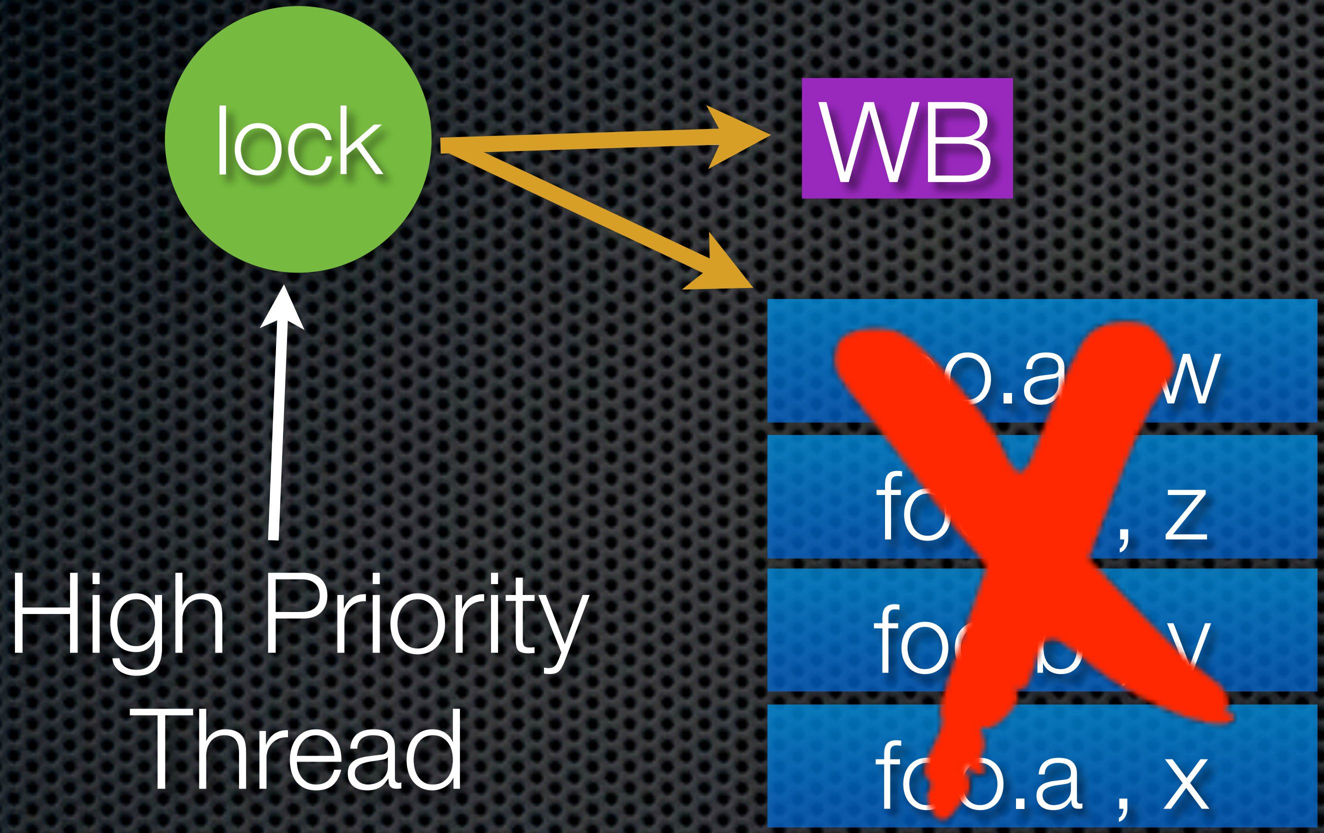
fo , z

fo b y

fo o.a , x

High Priority Thread

Acquisition by higher priority thread



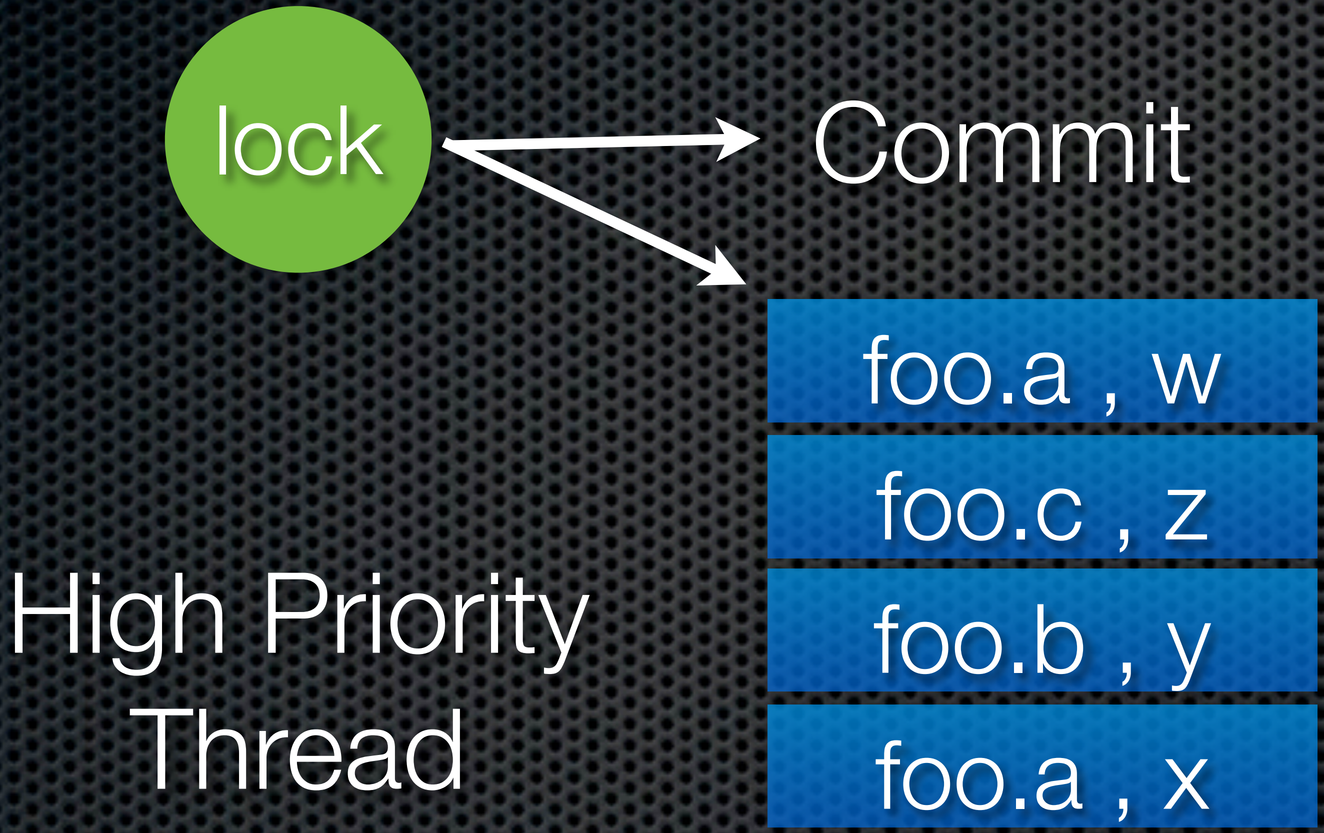
No priority boosting!

High Priority Thread Acquires Lock

Fiji Systems LLC

Acquisition by higher priority thread

Main Memory



lock

Commit

foo.a , w

foo.c , z

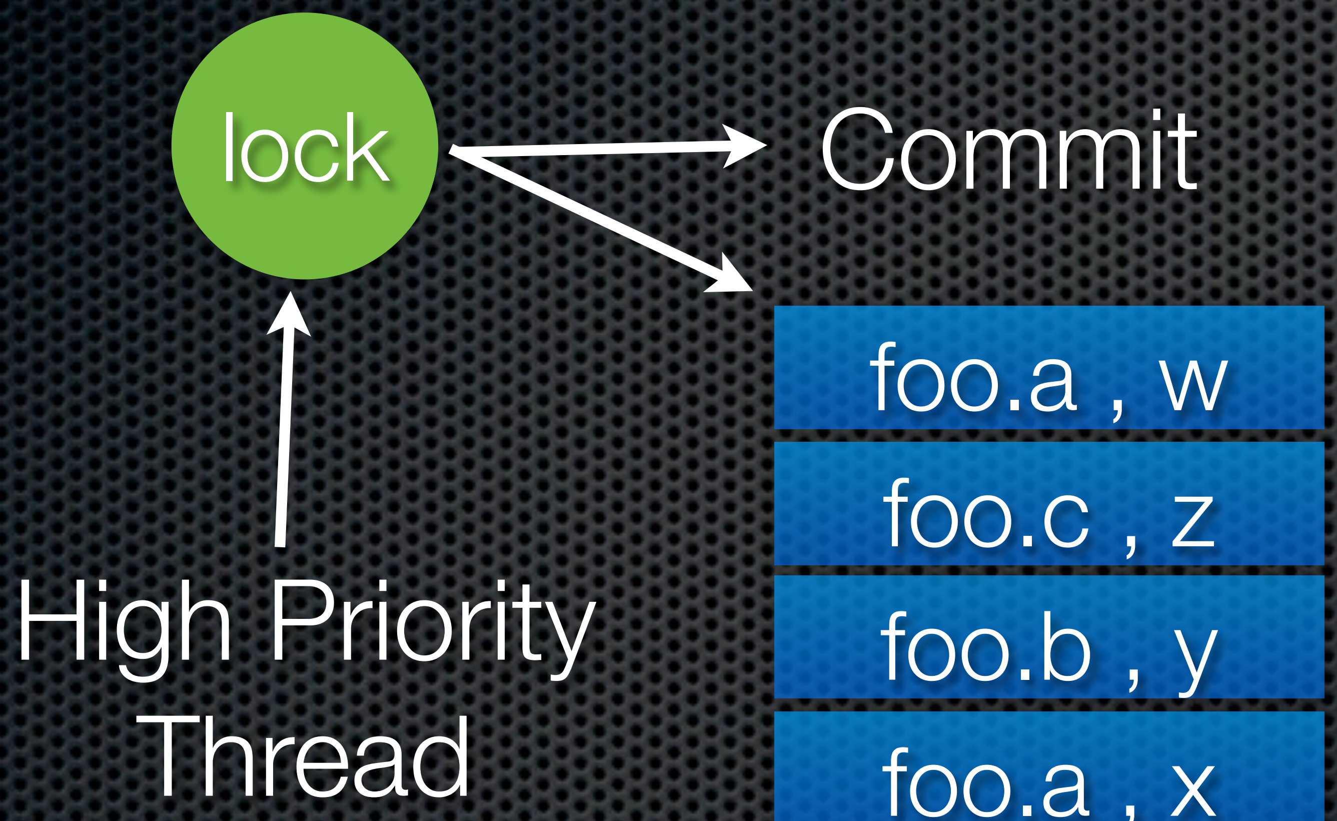
foo.b , y

foo.a , x

High Priority Thread

Acquisition by higher priority thread

Main Memory



lock

Commit

foo.a , w

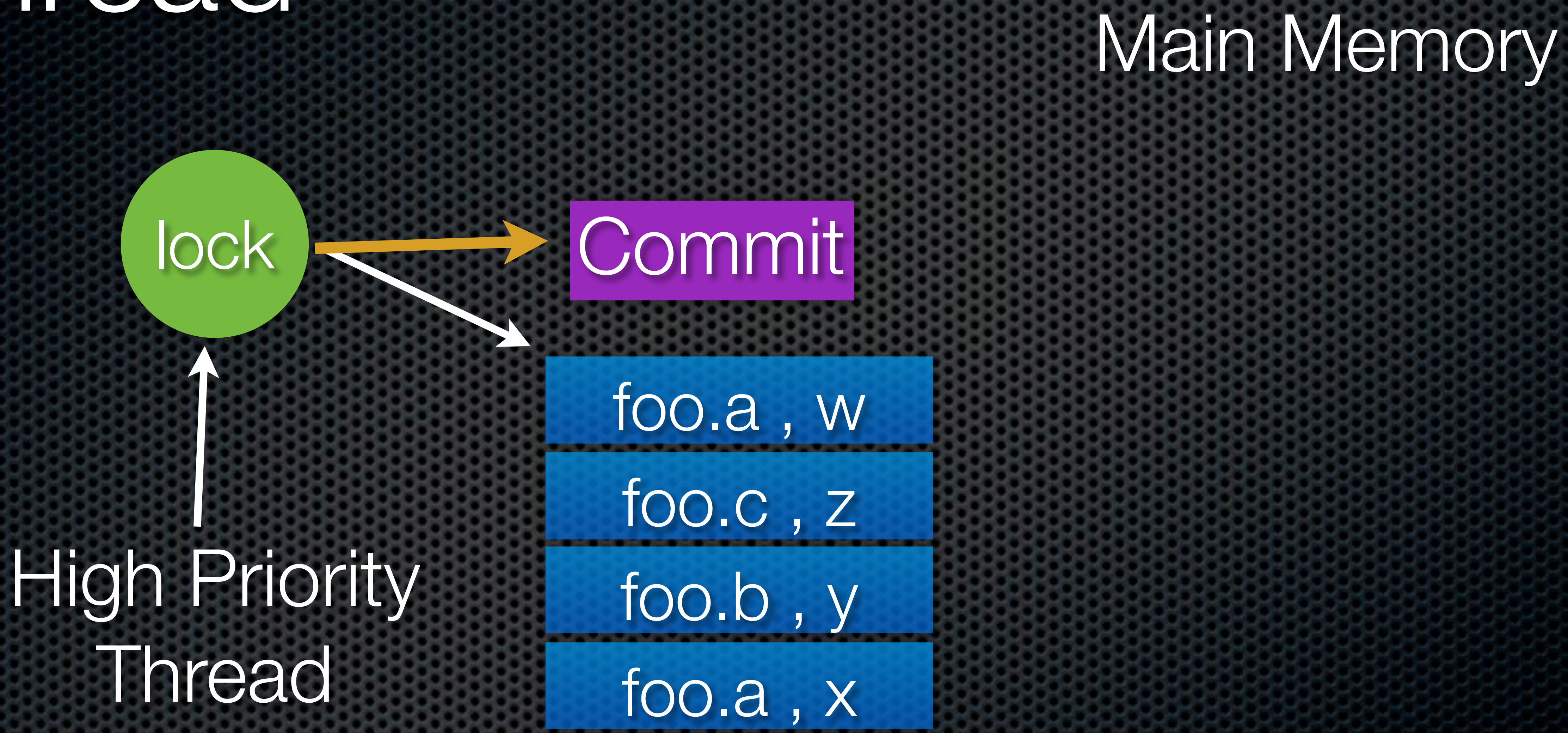
foo.c , z

foo.b , y

foo.a , x

High Priority Thread

Acquisition by higher priority thread



Main Memory

lock

Commit

foo.a , w

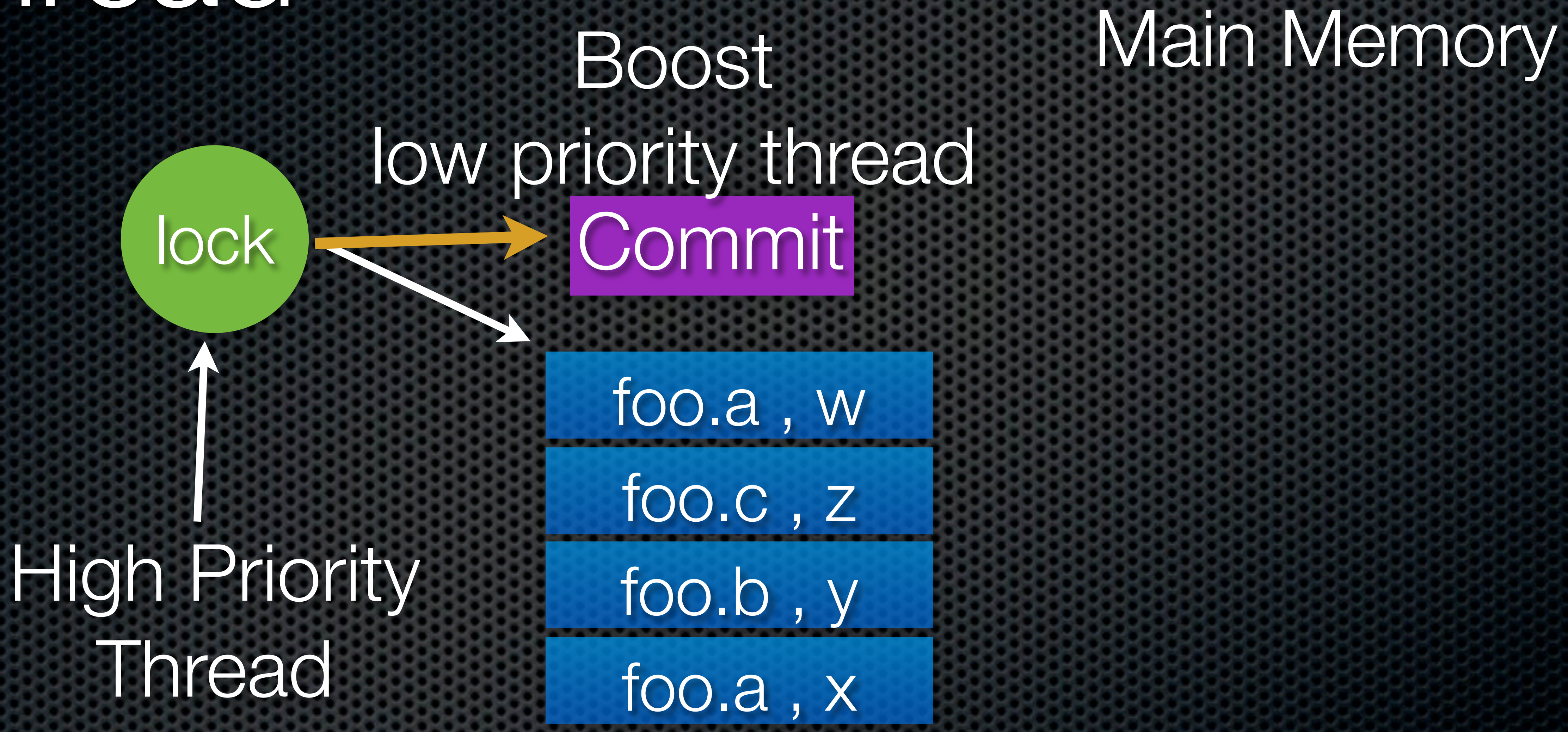
foo.c , z

foo.b , y

foo.a , x

High Priority Thread

Acquisition by higher priority thread



Boost

low priority thread

Main Memory

lock

Commit

foo.a , w

foo.c , z

foo.b , y

foo.a , x

High Priority Thread

Fiji

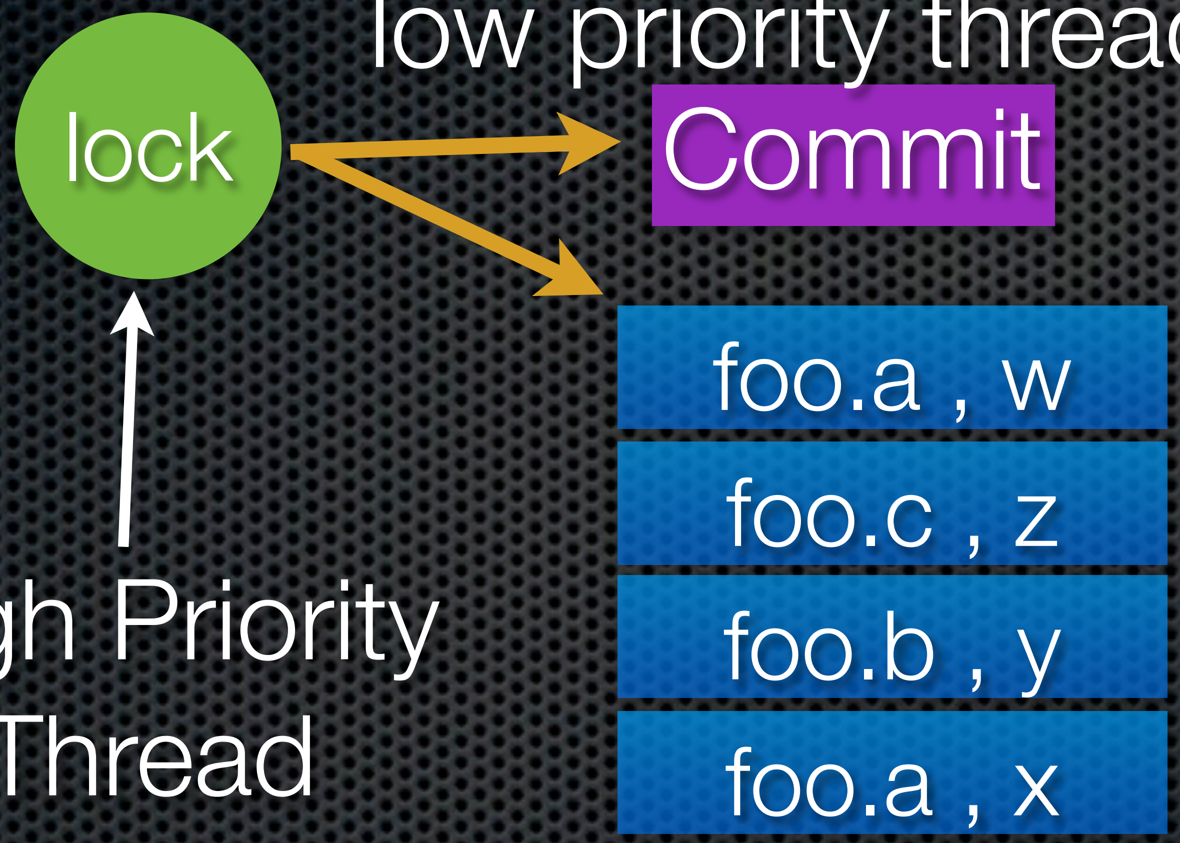
Systems LLC

Acquisition by higher priority thread

Main Memory

Boost

low priority thread



lock

Commit

foo.a , w

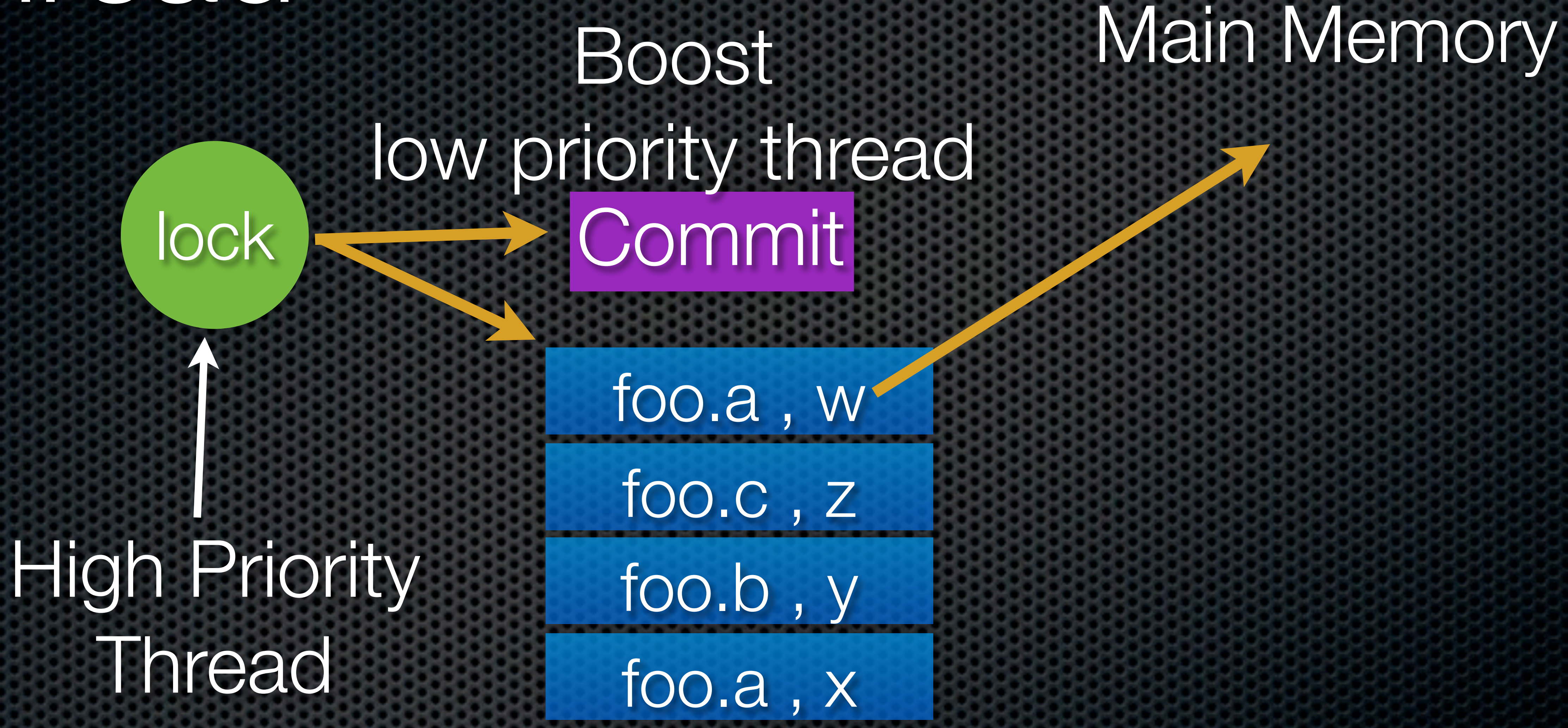
foo.c , z

foo.b , y

foo.a , x

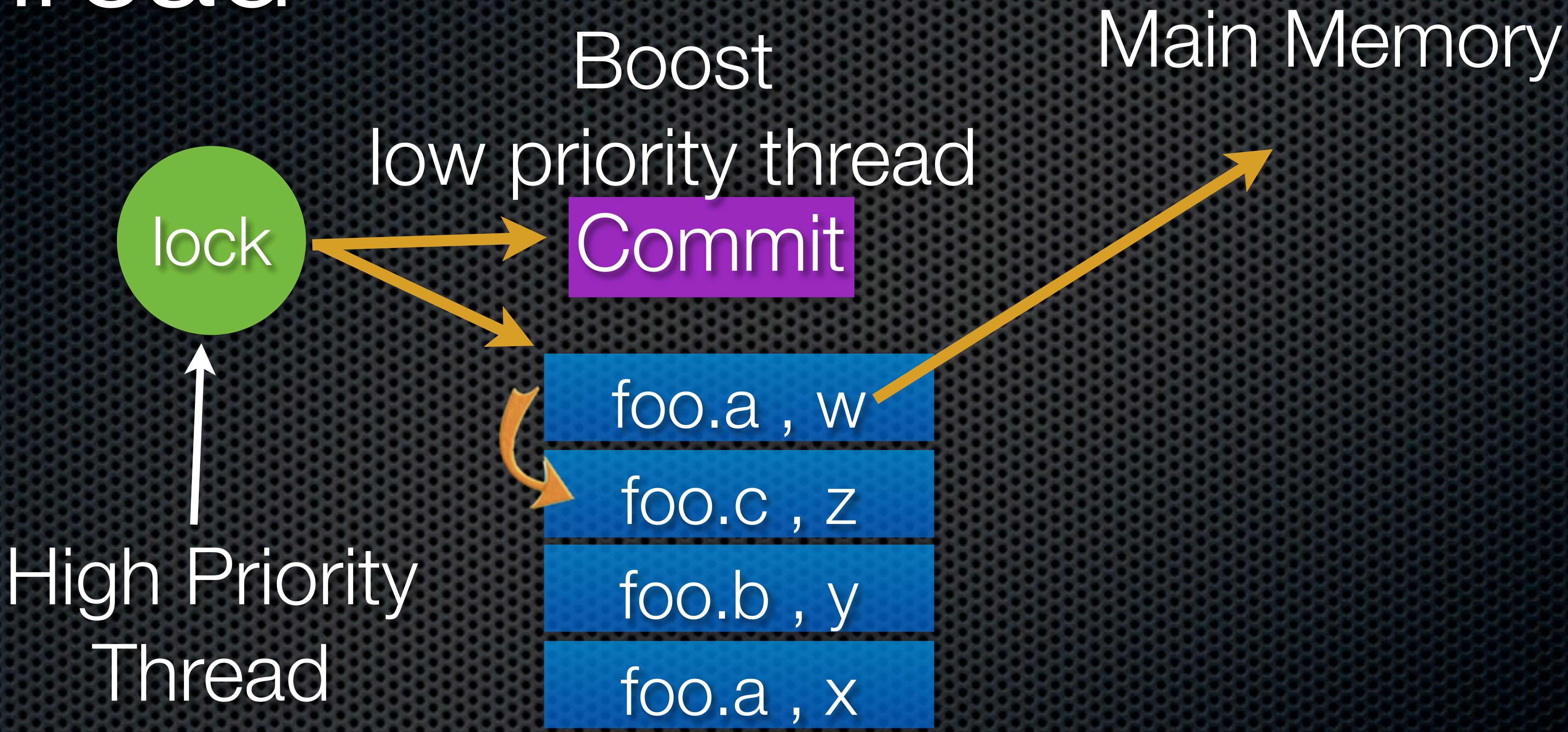
High Priority Thread

Acquisition by higher priority thread



Fiji
Systems LLC

Acquisition by higher priority thread



Boost

low priority thread

Main Memory

lock

Commit

foo.a , w

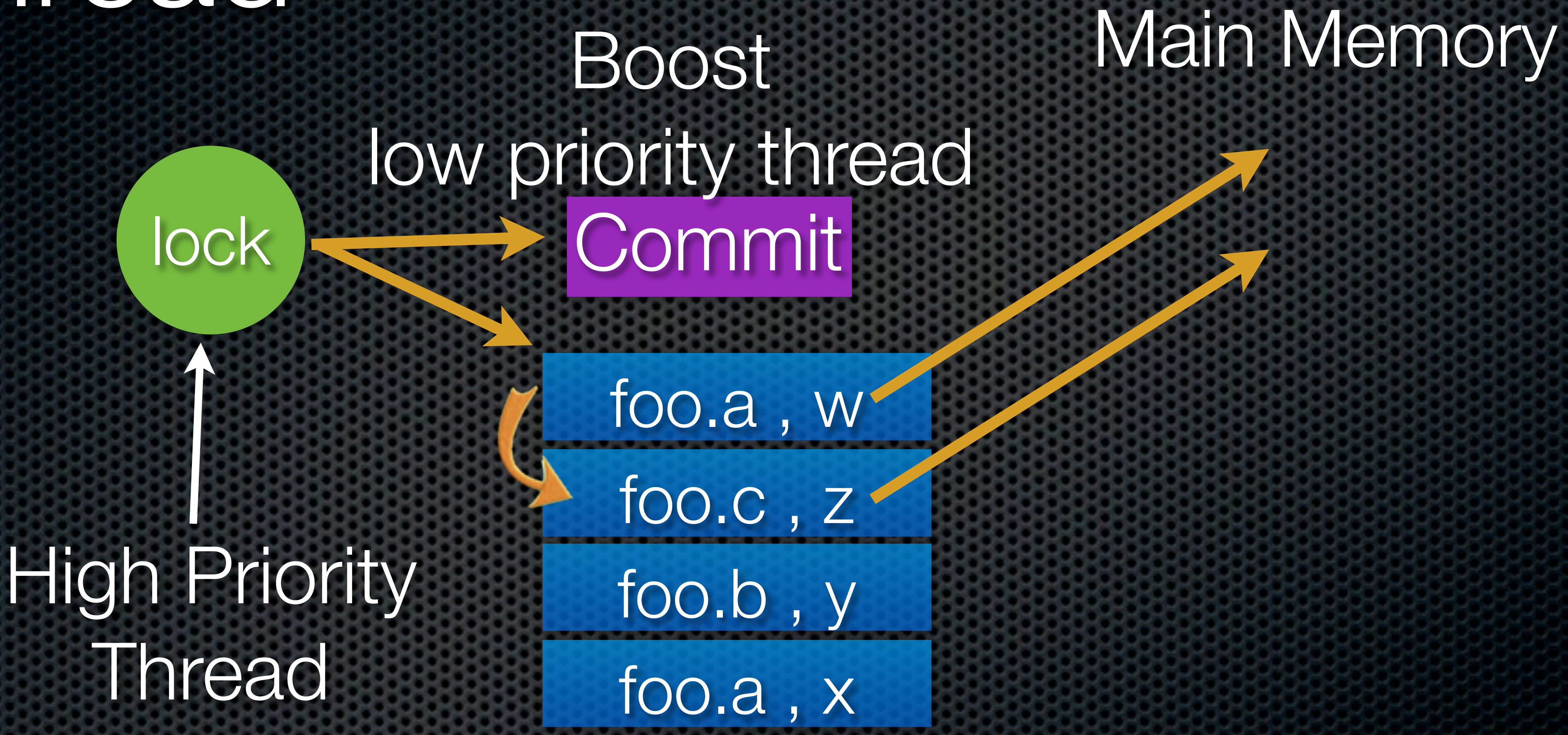
foo.c , z

foo.b , y

foo.a , x

High Priority Thread

Acquisition by higher priority thread



Boost

low priority thread

Main Memory

lock

Commit

foo.a , w

foo.c , z

foo.b , y

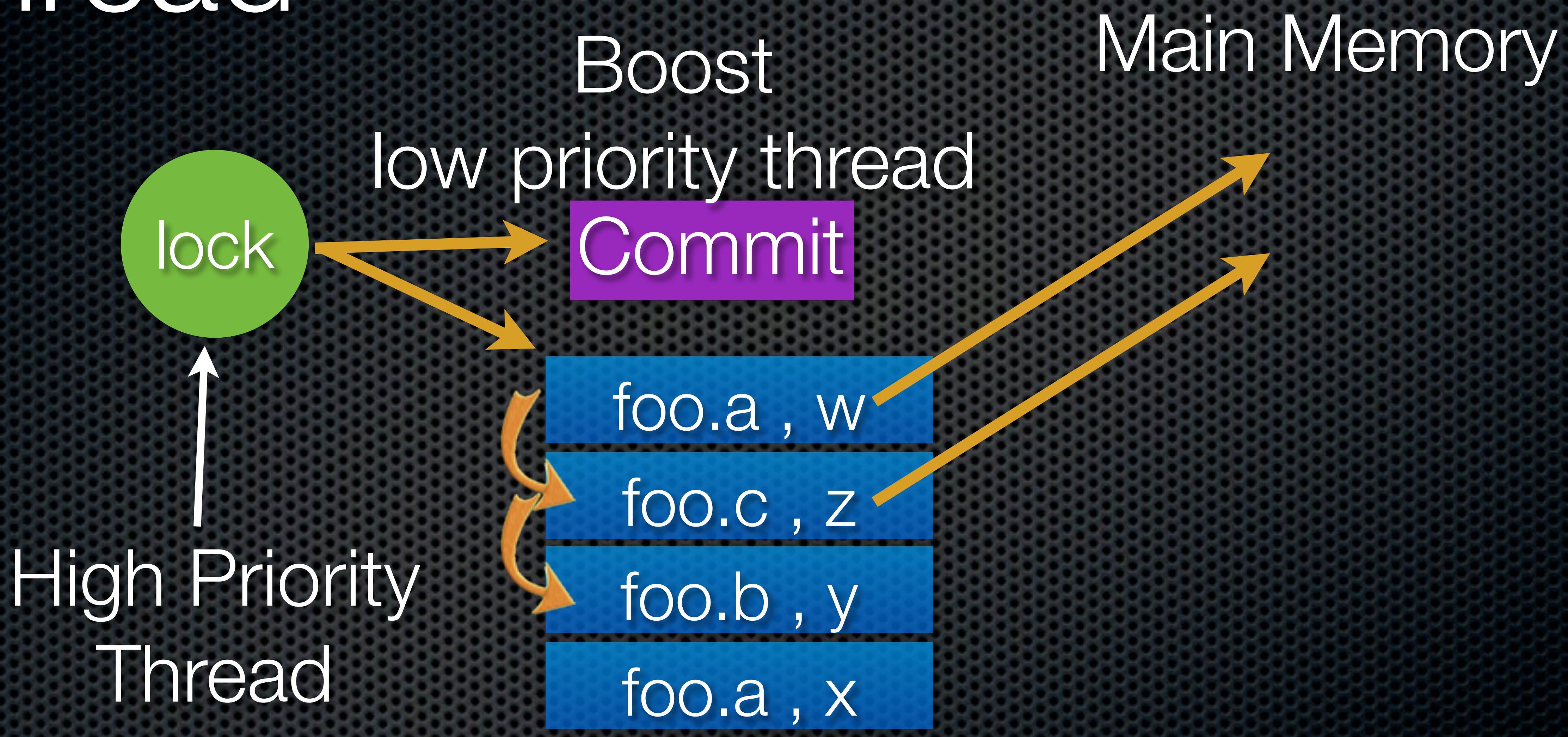
foo.a , x

High Priority Thread

Fiji

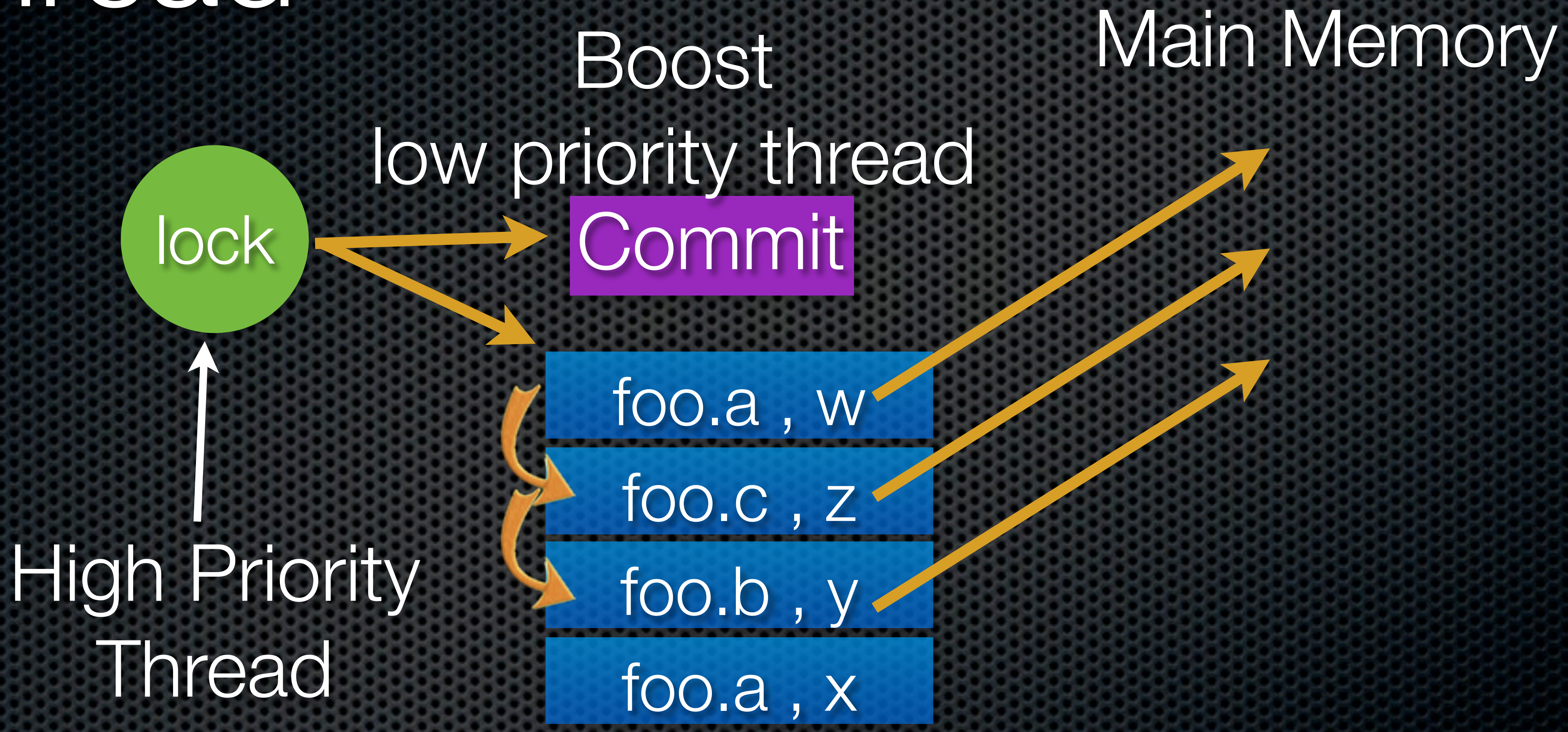
Systems LLC

Acquisition by higher priority thread



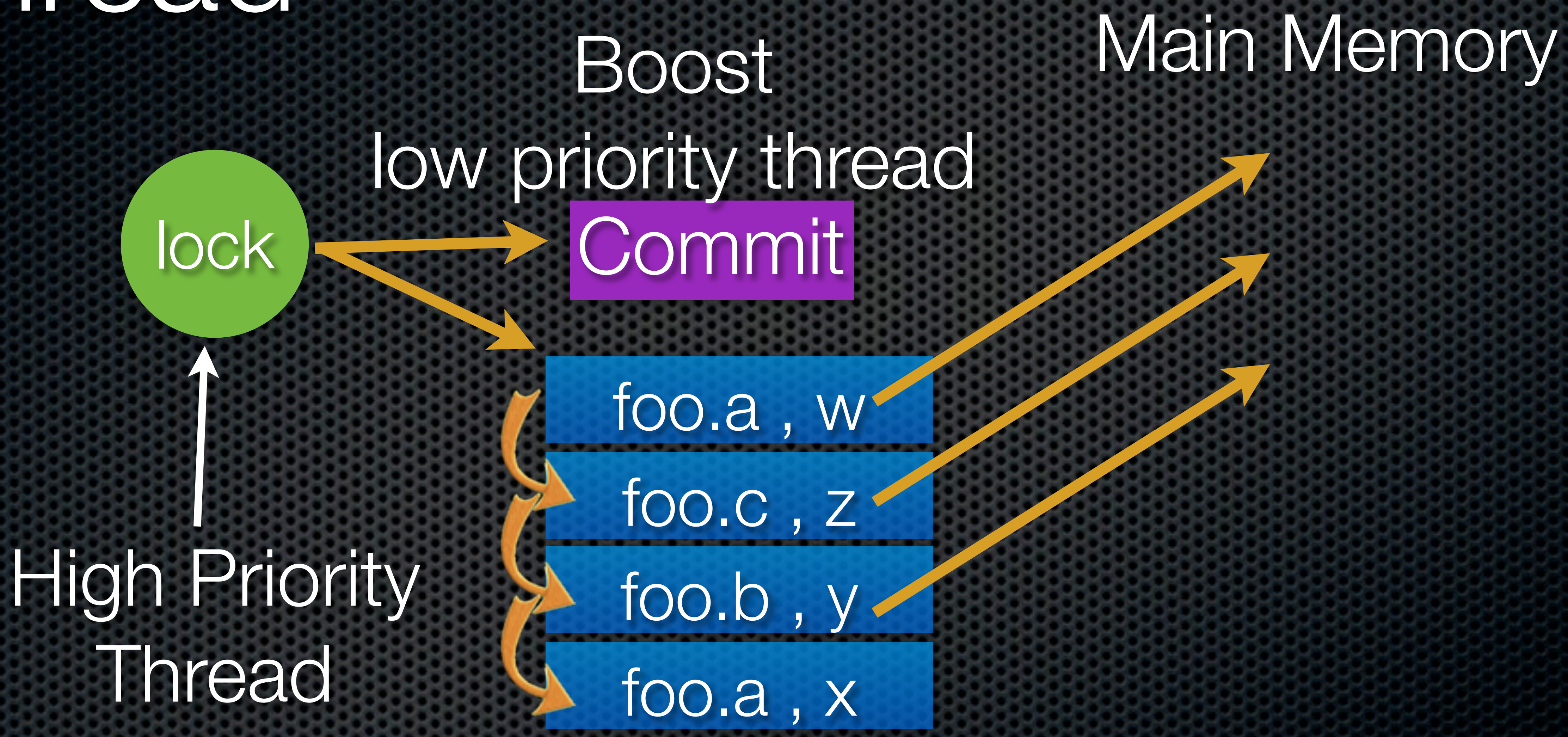
Fiji
Systems LLC

Acquisition by higher priority thread



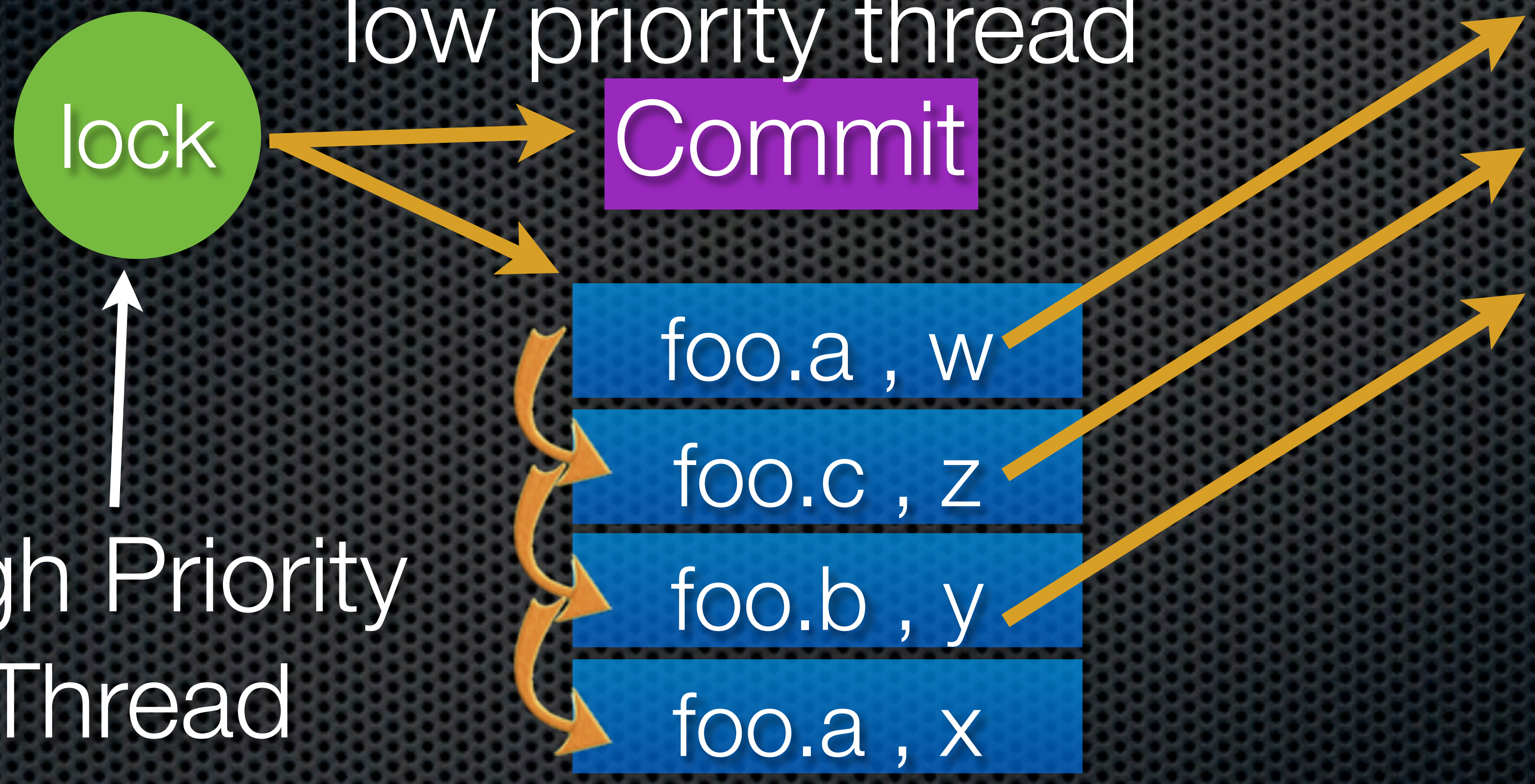
Fiji
Systems LLC

Acquisition by higher priority thread



Fiji
Systems LLC

Acquisition by higher priority thread



Main Memory

Boost

low priority thread

lock

Commit

foo.a , w

foo.c , z

foo.b , y

foo.a , x

High Priority Thread

High Priority Thread Acquires Lock

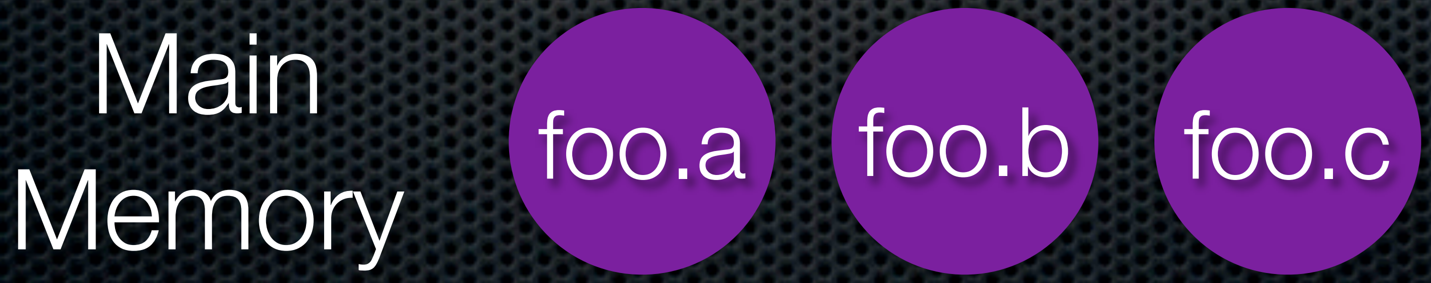
Complexity Costs - Write Buffering Low Priority Thread

- Reads: log(size of buffer) --- use RB tree
- Writes: log(size of buffer)
- Commit: size of buffer -- nested commit: n log(n)
- Acquisition: constant if thread not flushing buffer, size of buffer + context switches otherwise
- Memory: size of buffer

Fiji
Systems LLC

Write Logging

```
synchronized(lock){
  foo.a = x;
  foo.b = y;
  foo.c = z;
  foo.a = w;
  if(foo.b+4 > foo.a)
    ...
}
```



Write Logging

lock

set to acquired
in WL mode

```
synchronized(lock){
    foo.a = x;
    foo.b = y;
    foo.c = z;
     foo.a = w;
    if(foo.b+4 > foo.a)
        ...
}
```

Log

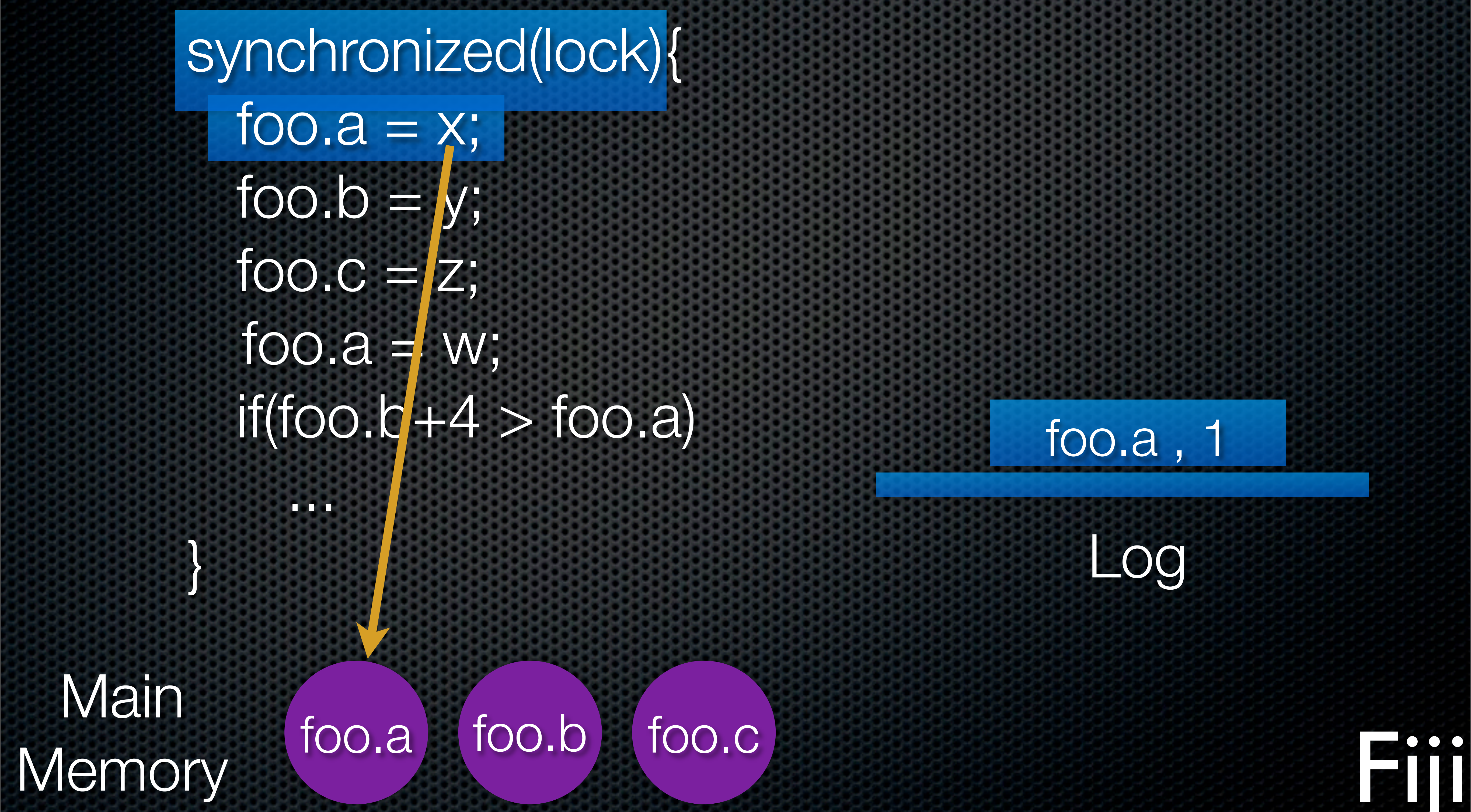
Main
Memory

foo.a foo.b foo.c

Write Logging

lock

set to acquired
in WL mode

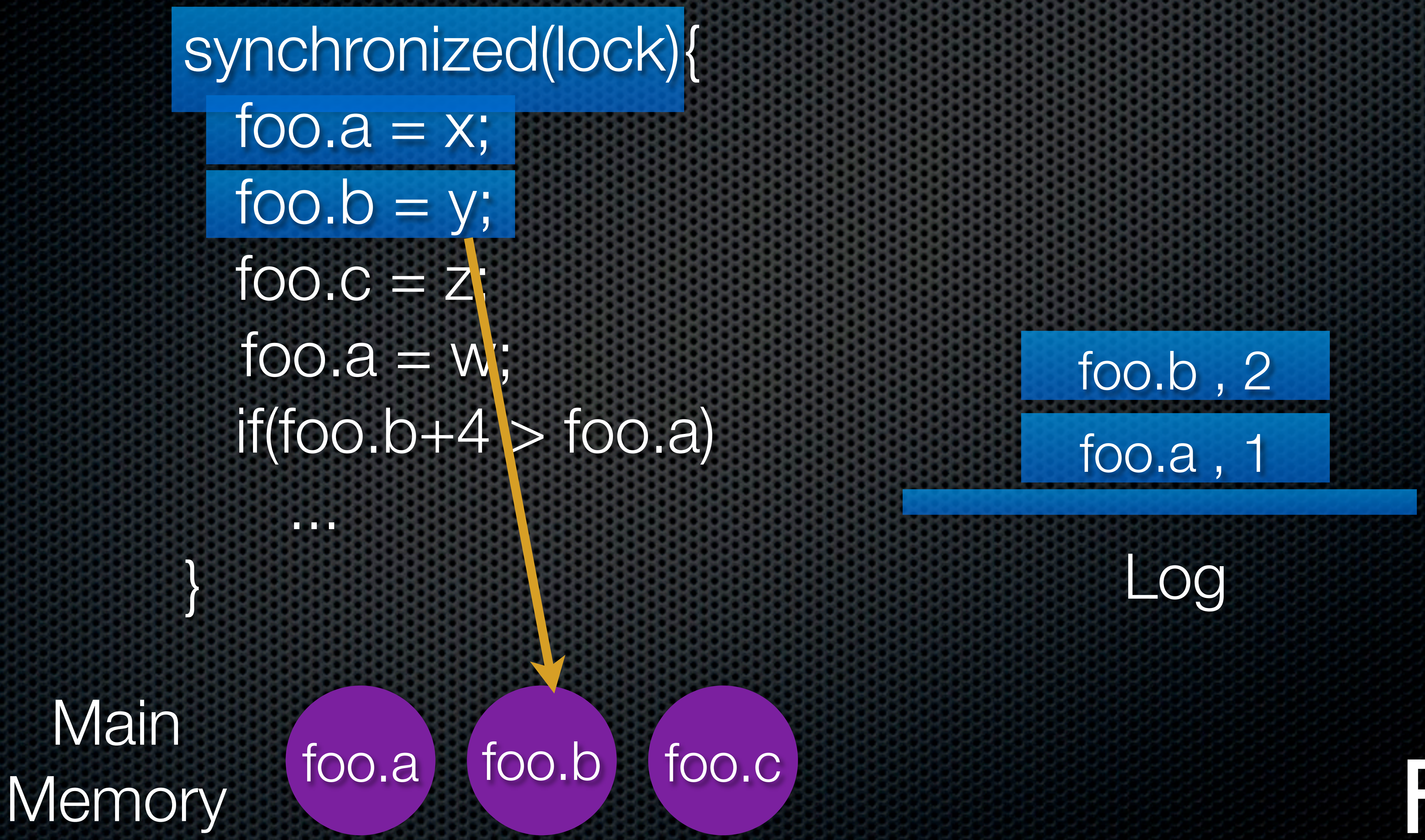


```
synchronized(lock){
  foo.a = x;
  foo.b = y;
  foo.c = z;
  foo.a = w;
  if(foo.b+4 > foo.a)
      ...
}
```

Write Logging

lock

set to acquired in WL mode



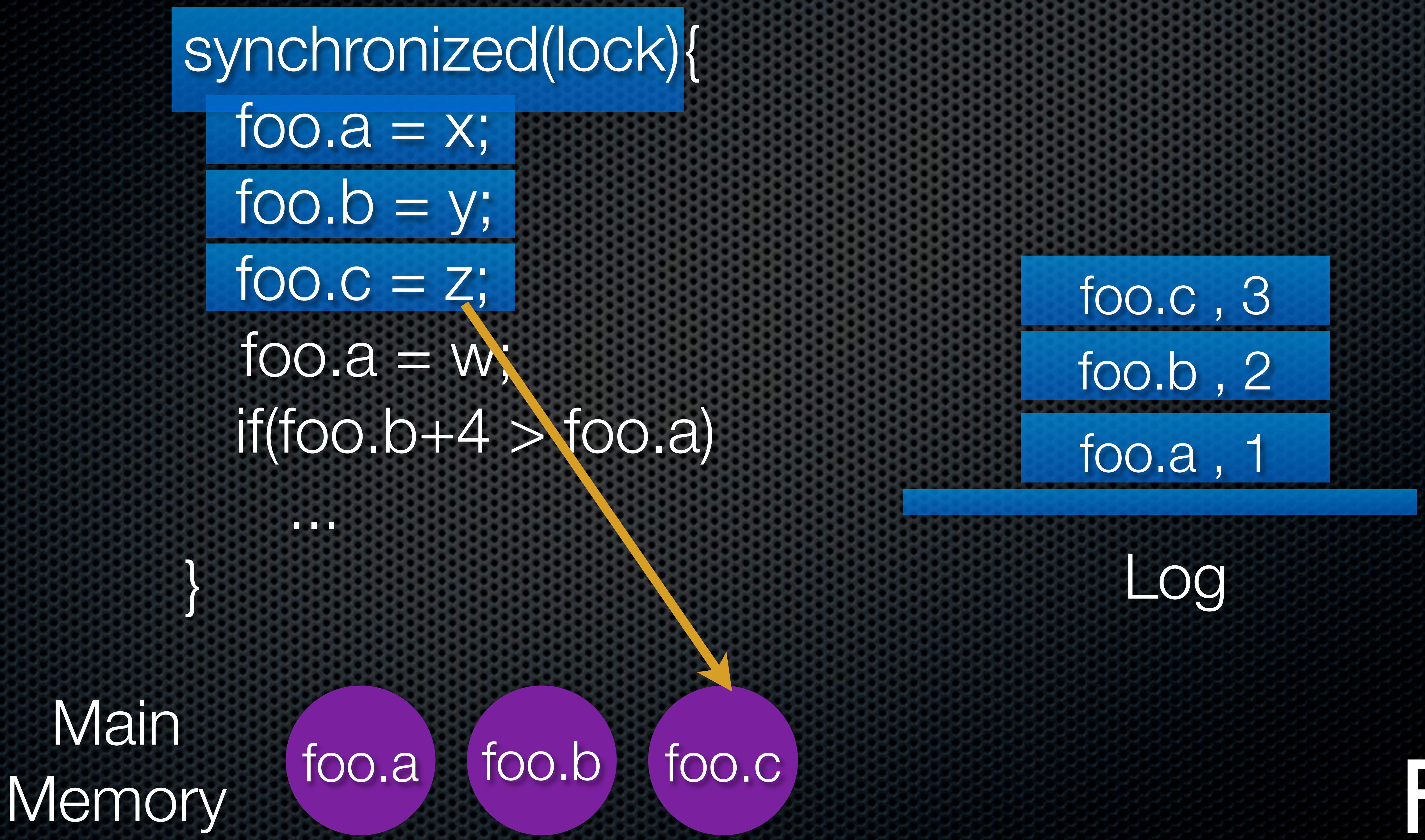
```
synchronized(lock){
   foo.a = x;
   foo.b = y;
   foo.c = z;
   foo.a = w;
   if(foo.b+4 > foo.a)
      ...
}
```

Fiji Systems LLC

Write Logging

lock

set to acquired
in WL mode



```
synchronized(lock){
  foo.a = x;
  foo.b = y;
  foo.c = z;
  foo.a = w;
  if(foo.b+4 > foo.a)
     ...
}
```

foo.c , 3

foo.b , 2

foo.a , 1

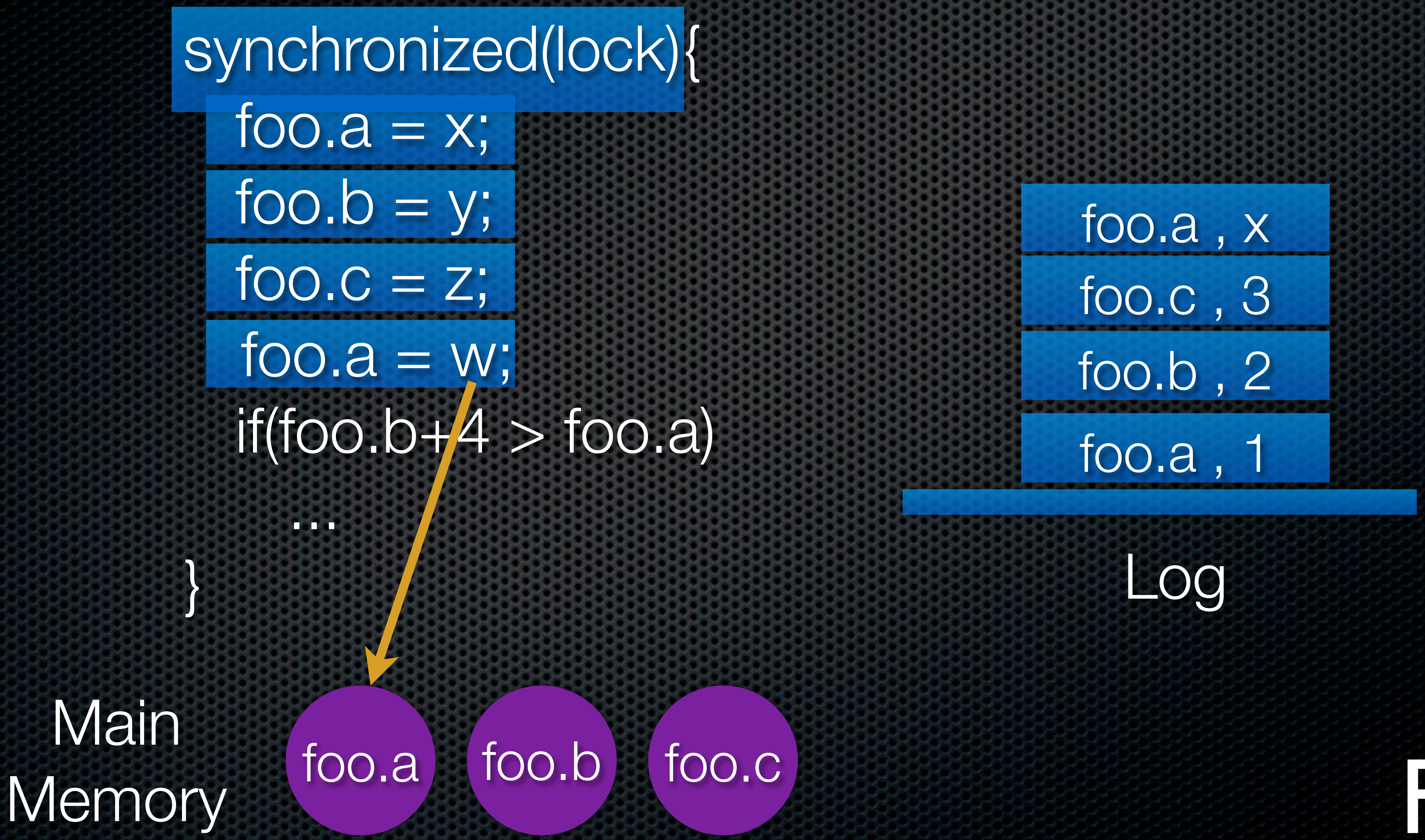
Log

Fiji
Systems LLC

Write Logging

lock

set to acquired in WL mode

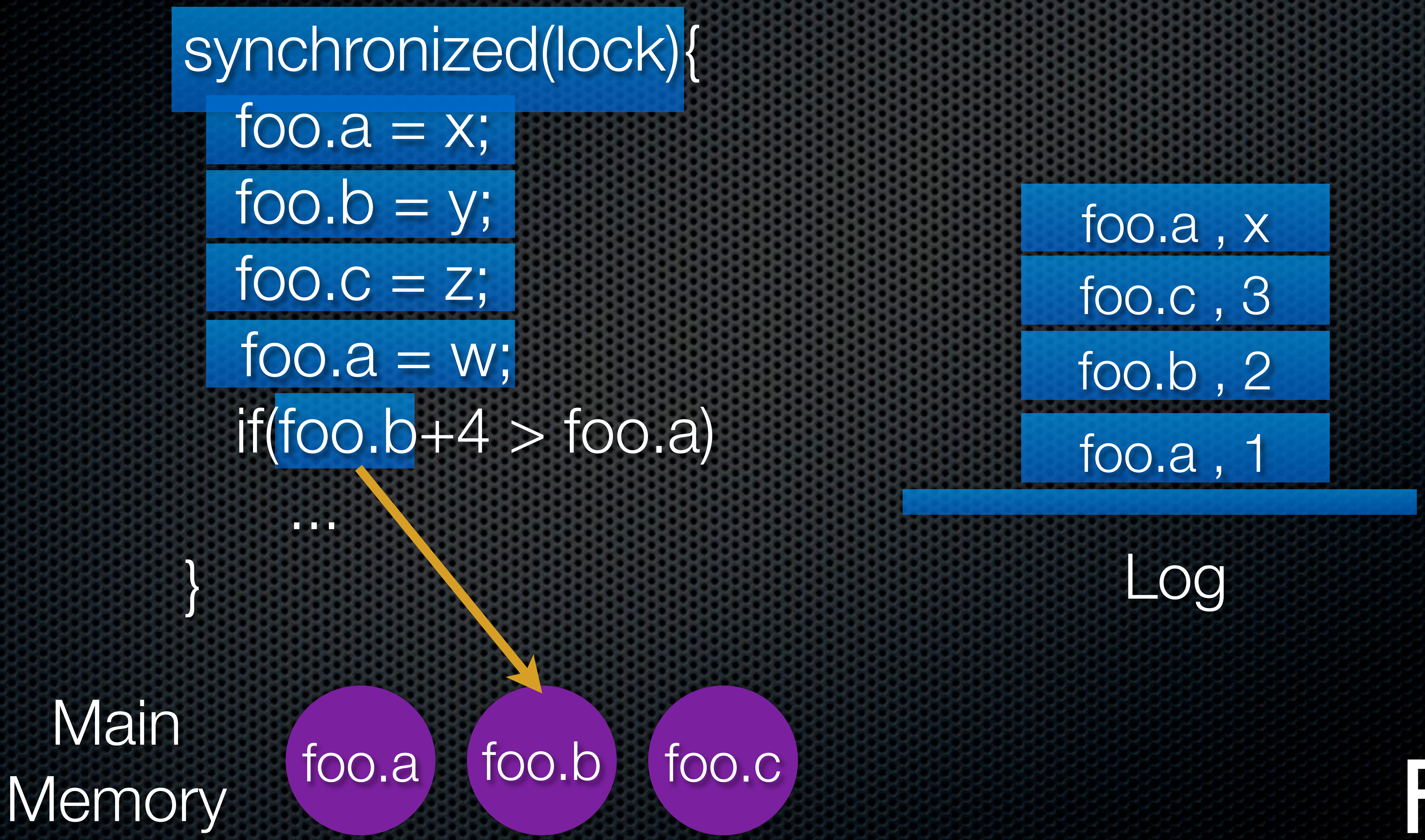


```
synchronized(lock){
   foo.a = x;
   foo.b = y;
   foo.c = z;
   foo.a = w;
   if(foo.b+4 > foo.a)
      ...
}
```

Write Logging

lock

set to acquired in WL mode



```
synchronized(lock){
  foo.a = x;
  foo.b = y;
  foo.c = z;
  foo.a = w;
  if(foo.b+4 > foo.a)
    ...
}
```

foo.a , x

foo.c , 3

foo.b , 2

foo.a , 1

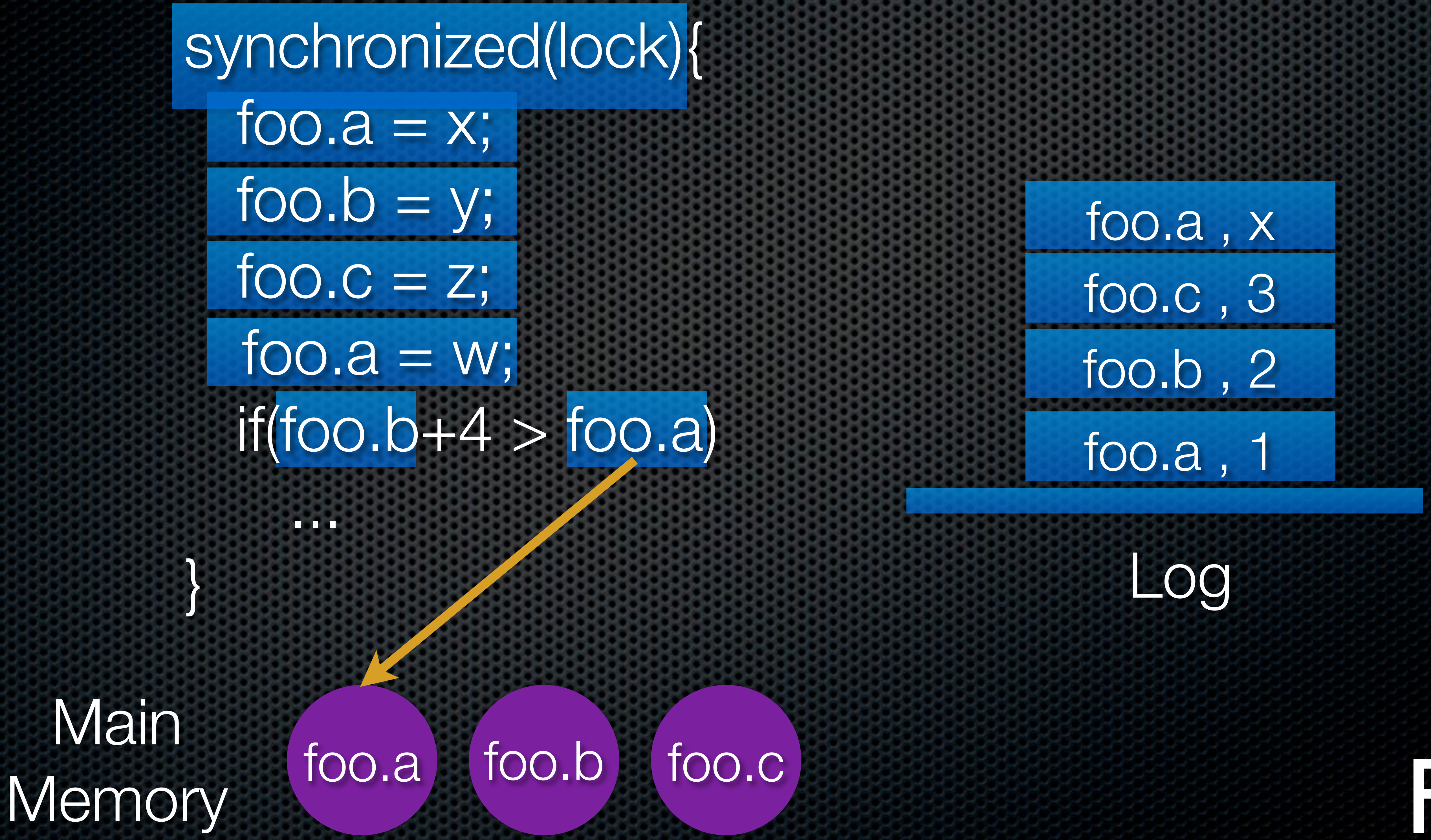
Log

Main Memory

Write Logging

lock

set to acquired
in WL mode



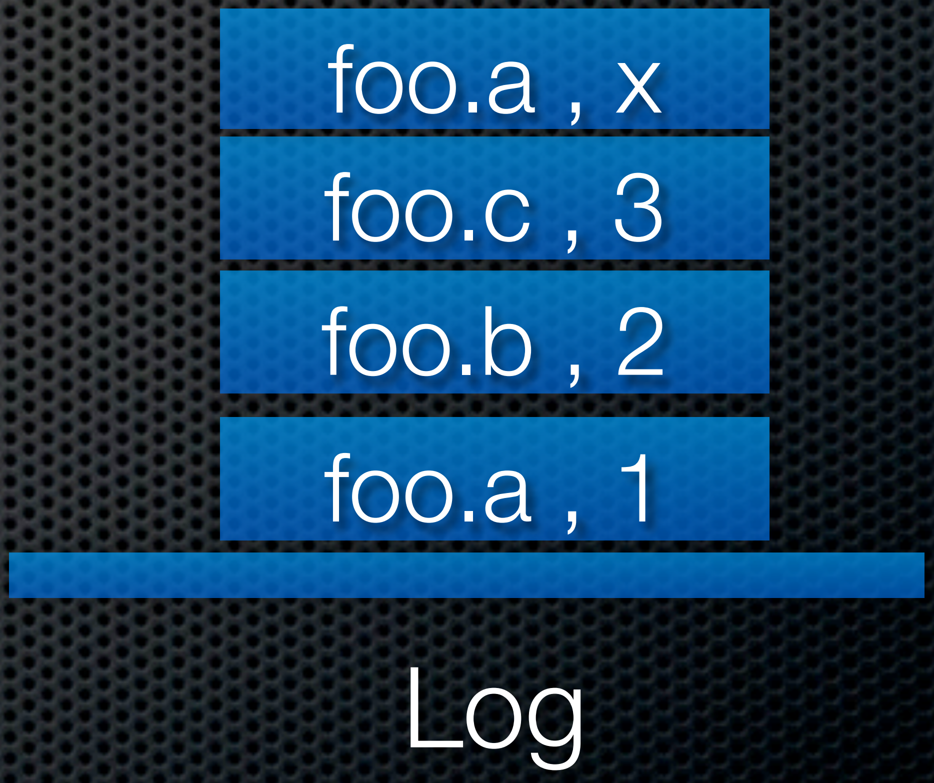
```
synchronized(lock){
   foo.a = x;
   foo.b = y;
   foo.c = z;
   foo.a = w;
   if(foo.b+4 > foo.a)
      ...
}
```

Fiji
Systems LLC

Write Logging

lock — set to acquired in WL mode

```
synchronized(lock){
   foo.a = x;
   foo.b = y;
   foo.c = z;
   foo.a = w;
   if(foo.b+4 > foo.a)
      ...
}  Commit
```



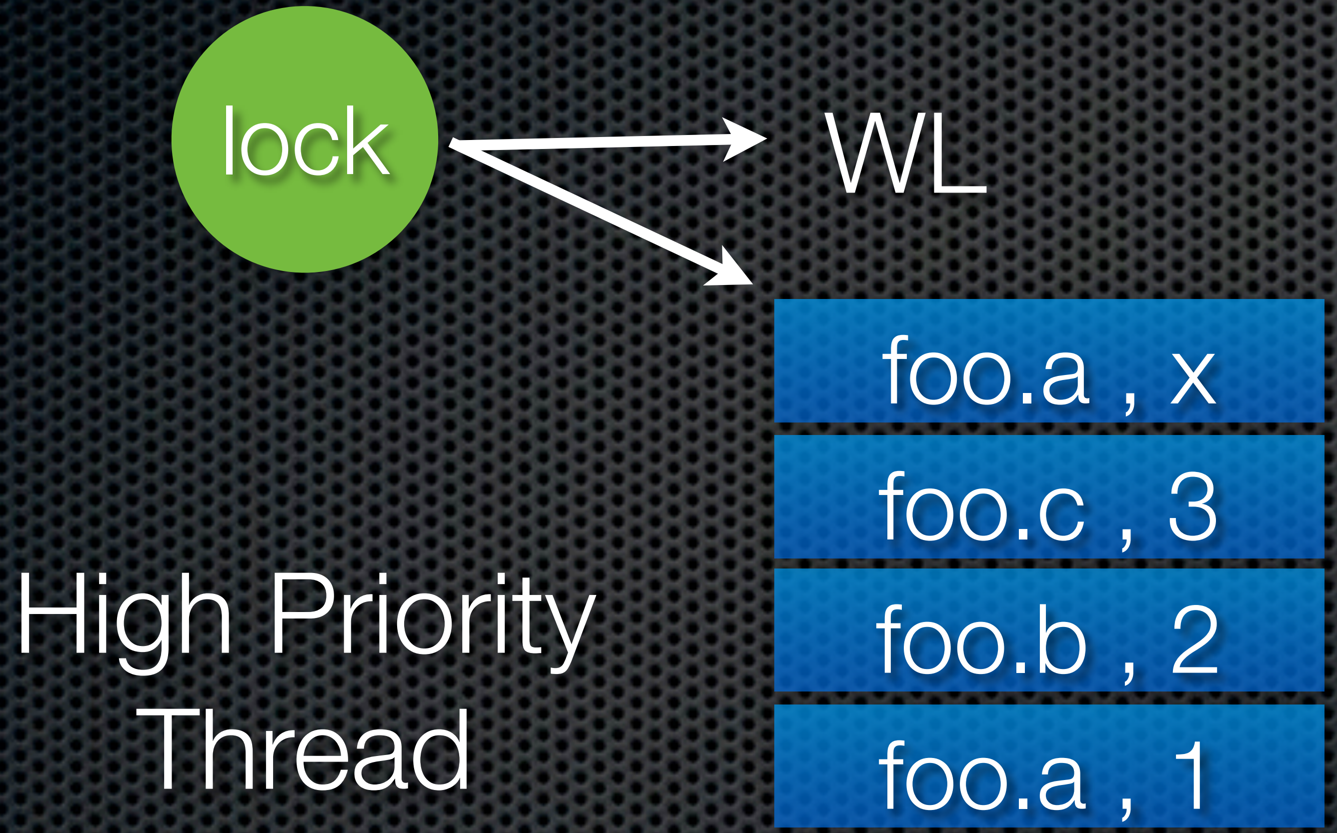
foo.a , x
foo.c , 3
foo.b , 2
foo.a , 1

Log

Main Memory
foo.a
foo.b
foo.c

Acquisition by higher priority thread

Main Memory



lock

WL

foo.a , x

foo.c , 3

foo.b , 2

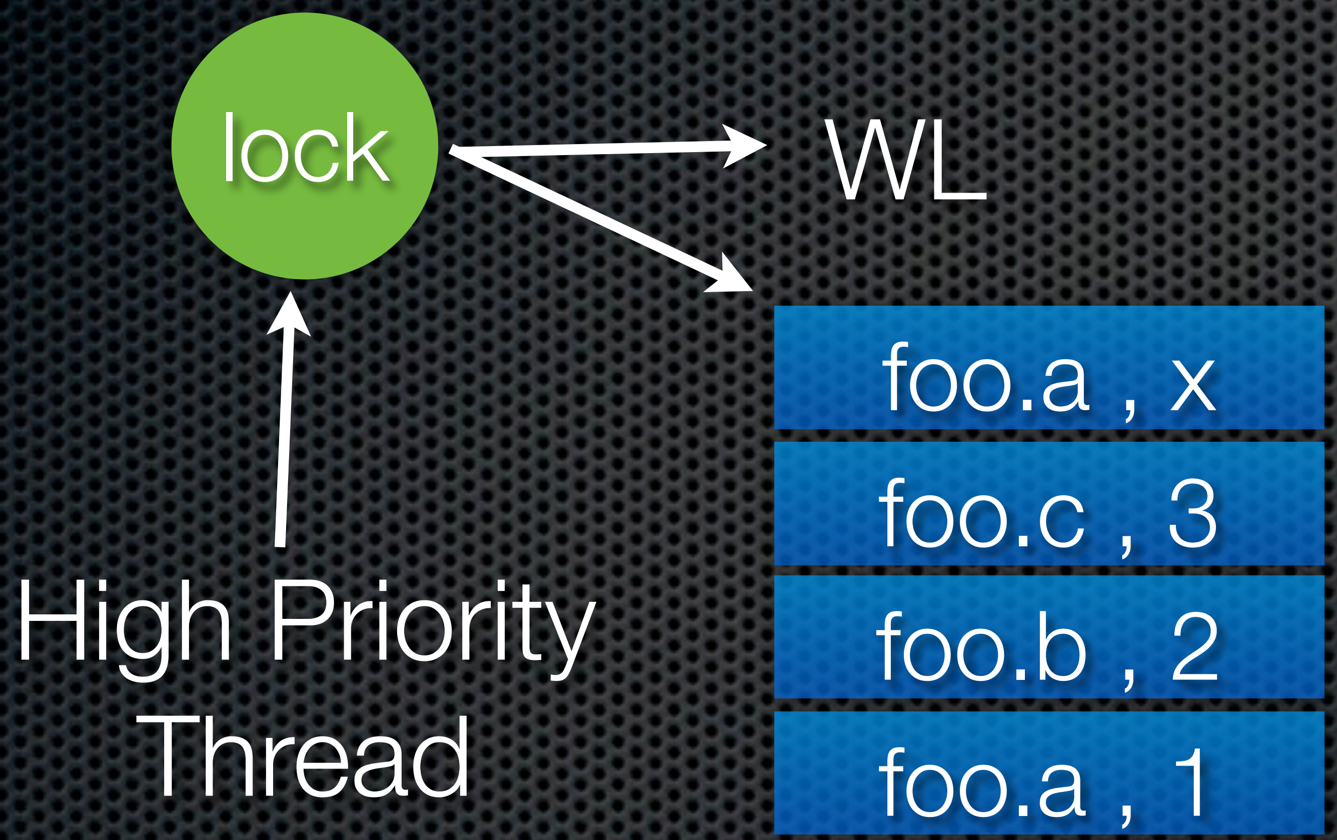
foo.a , 1

High Priority Thread

Fiji Systems LLC

Acquisition by higher priority thread

Main Memory



lock

WL

High Priority Thread

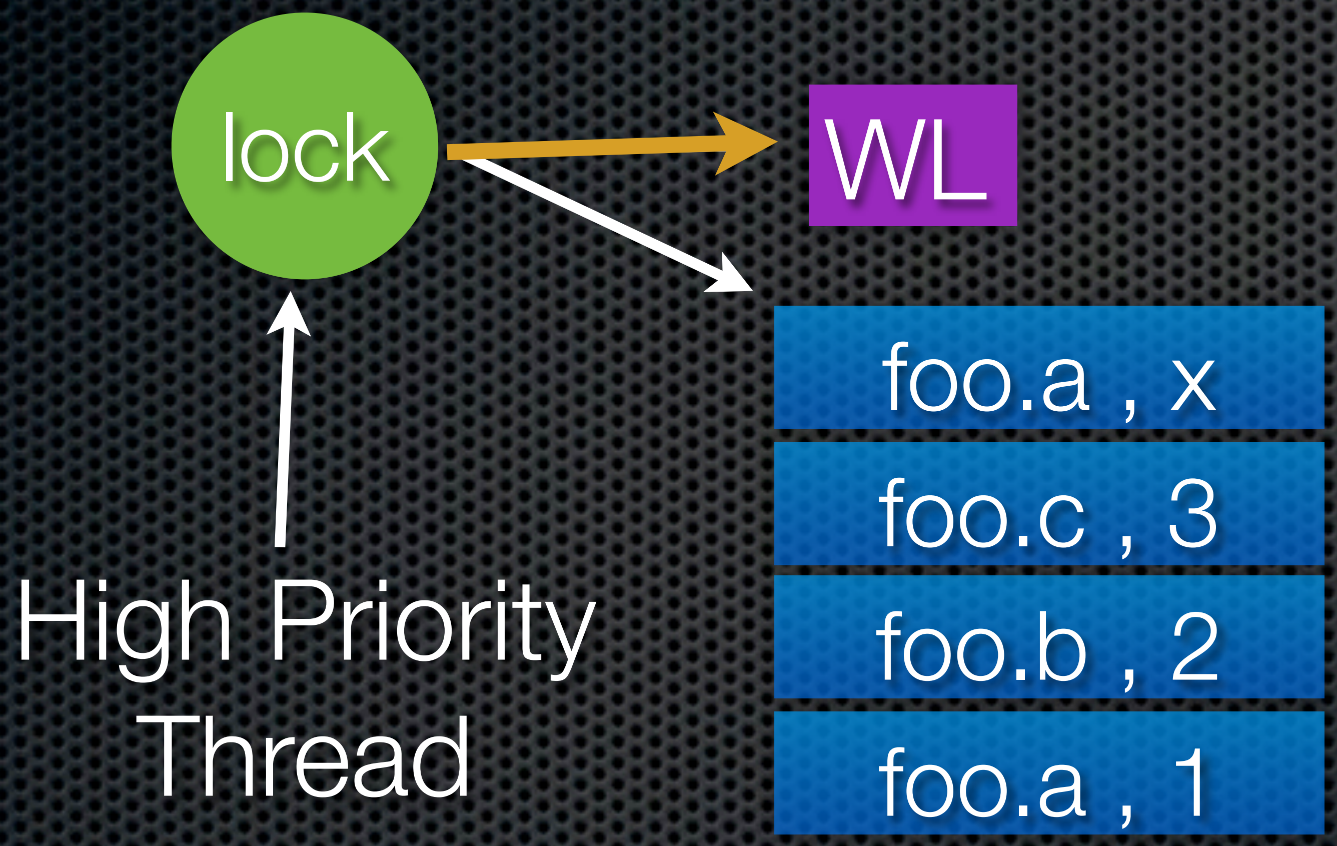
foo.a , x

foo.c , 3

foo.b , 2

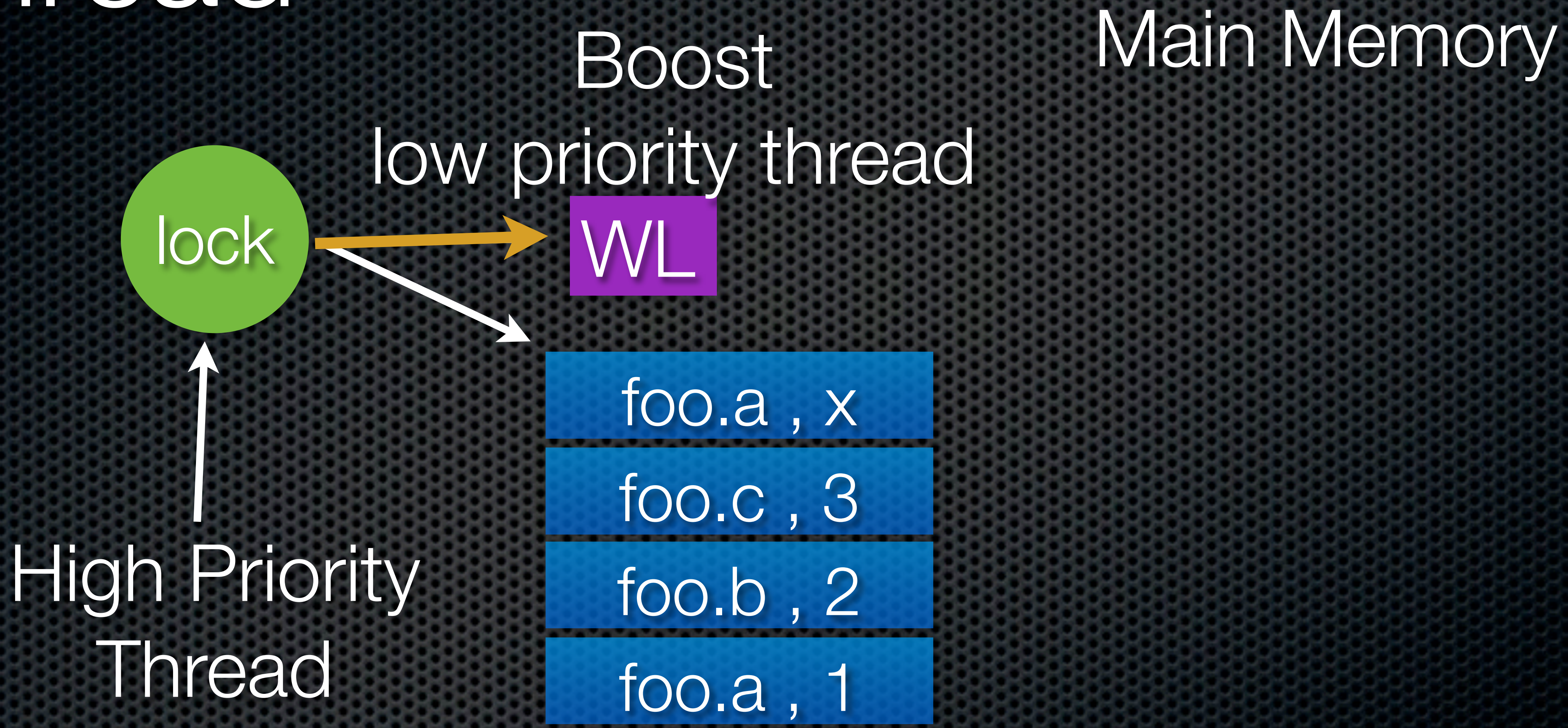
foo.a , 1

Acquisition by higher priority thread



Fiji Systems LLC

Acquisition by higher priority thread



Boost

low priority thread

Main Memory

lock

WL

foo.a , x

foo.c , 3

foo.b , 2

foo.a , 1

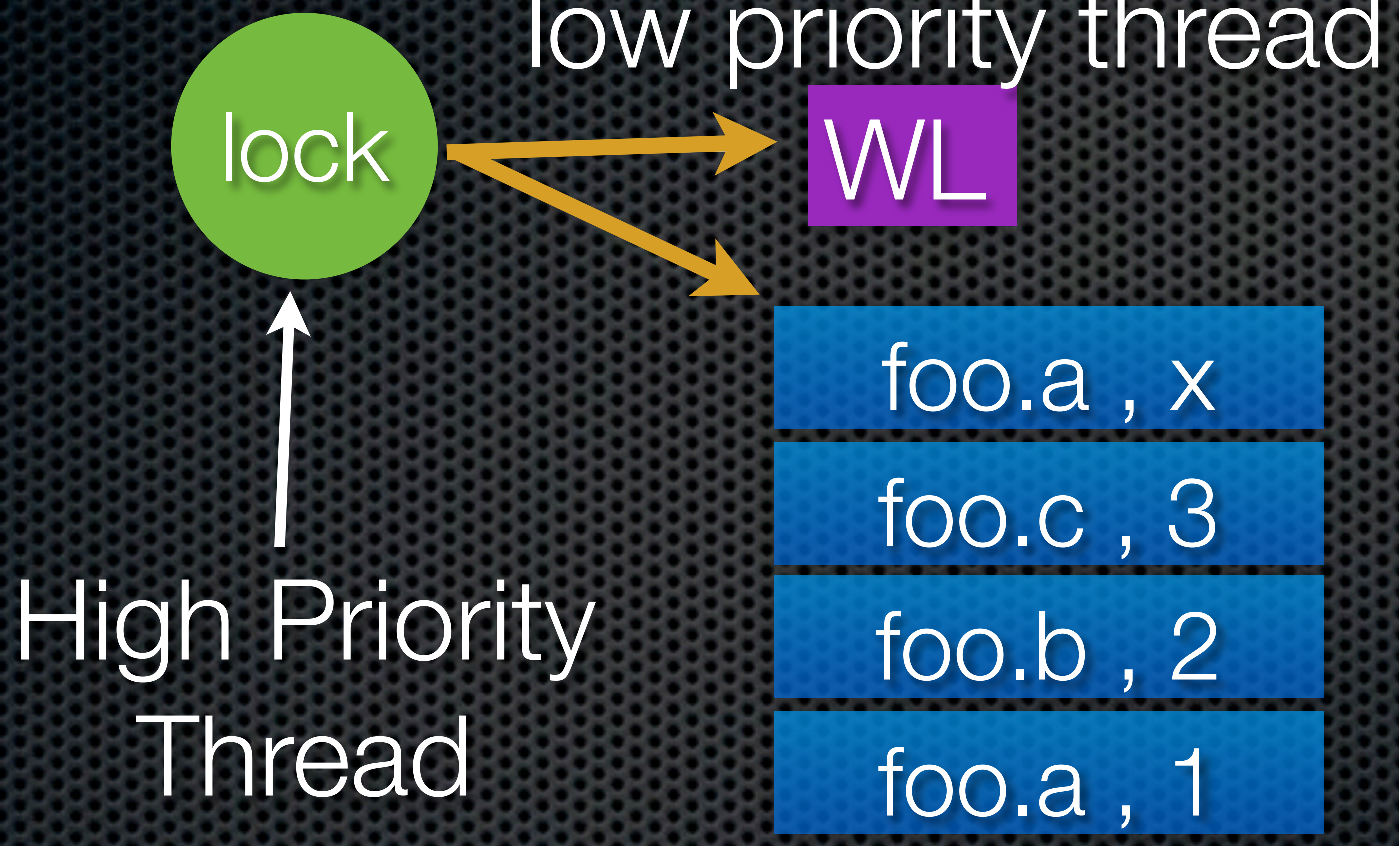
High Priority Thread

Fiji
Systems LLC

Acquisition by higher priority thread

Main Memory

Boost
low priority thread



lock

WL

foo.a , x

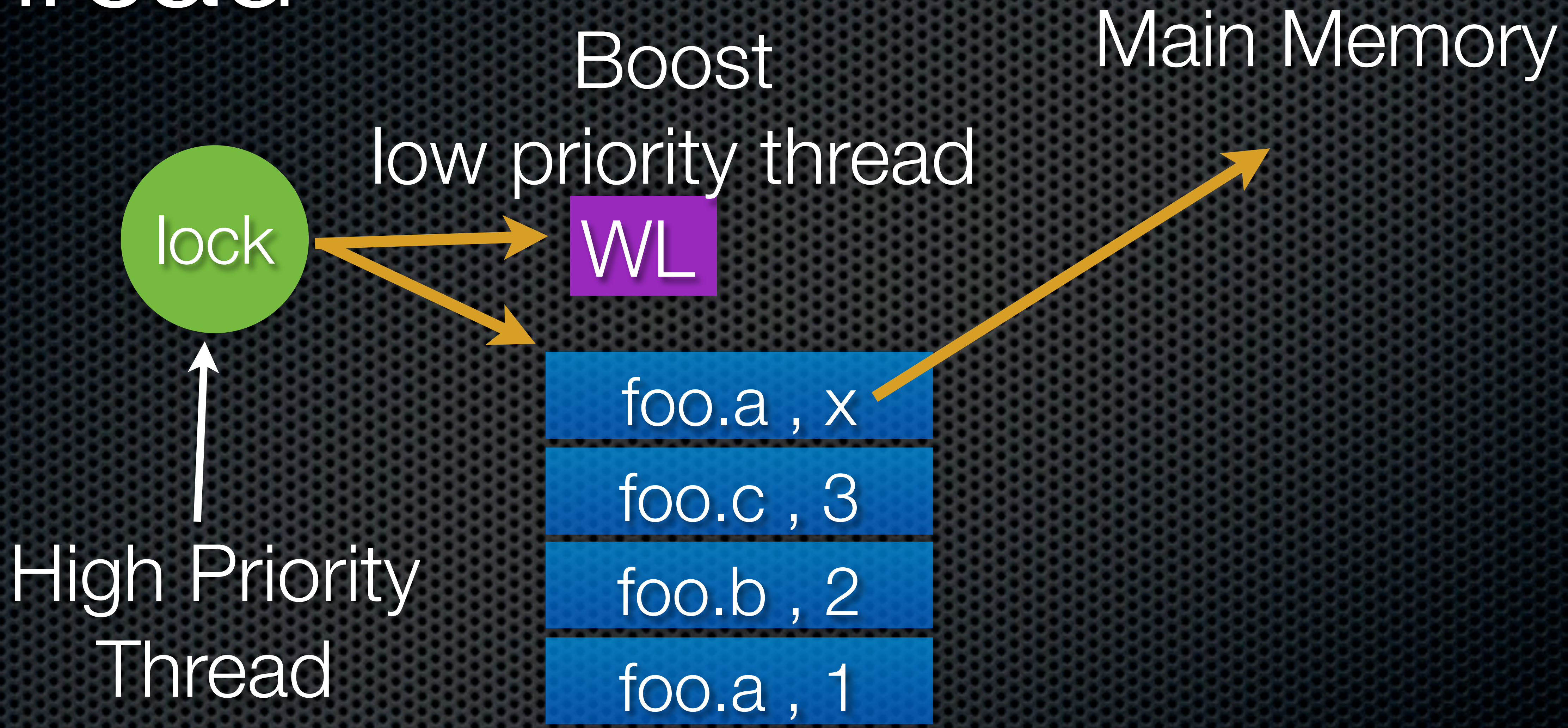
foo.c , 3

foo.b , 2

foo.a , 1

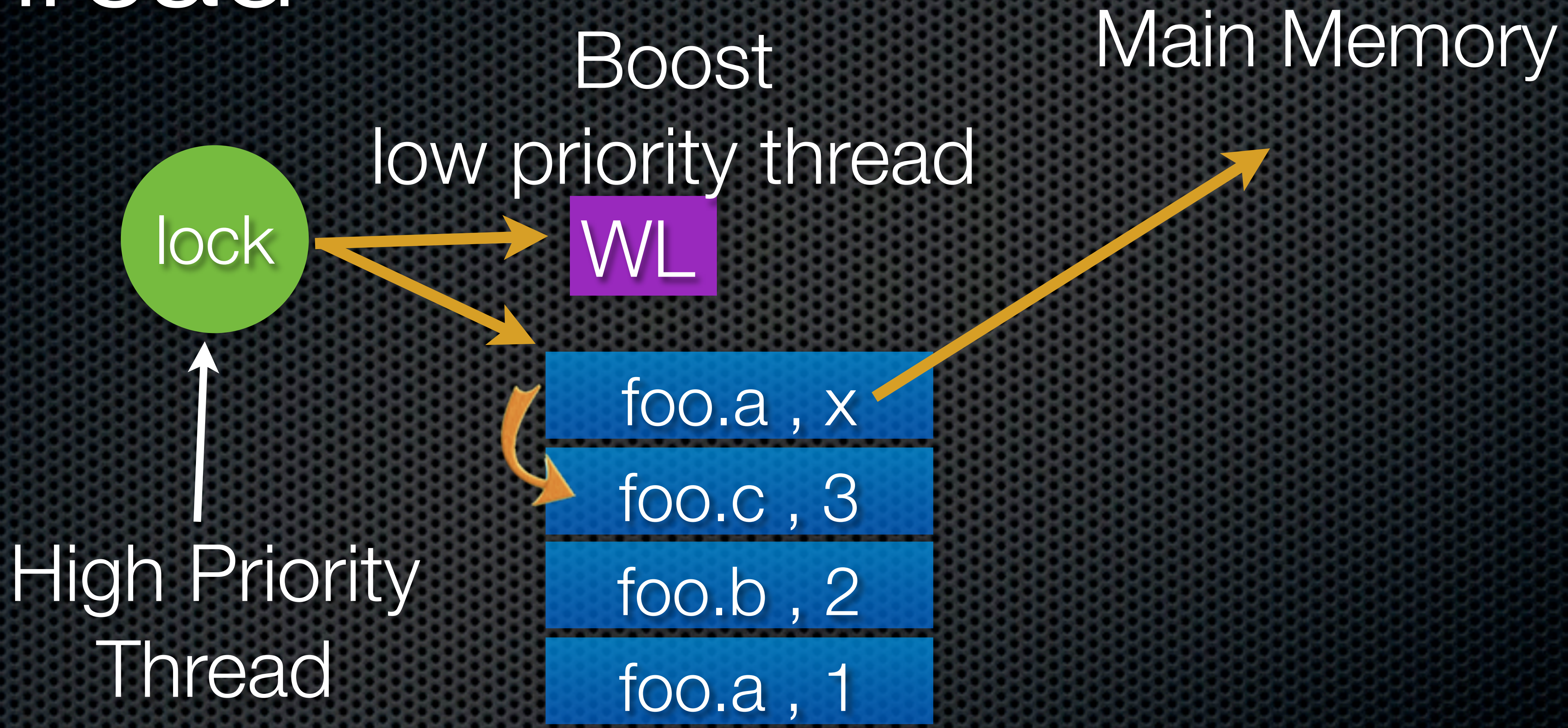
High Priority Thread

Acquisition by higher priority thread



Fiji
Systems LLC

Acquisition by higher priority thread



Boost

low priority thread

Main Memory

lock

WL

foo.a , x

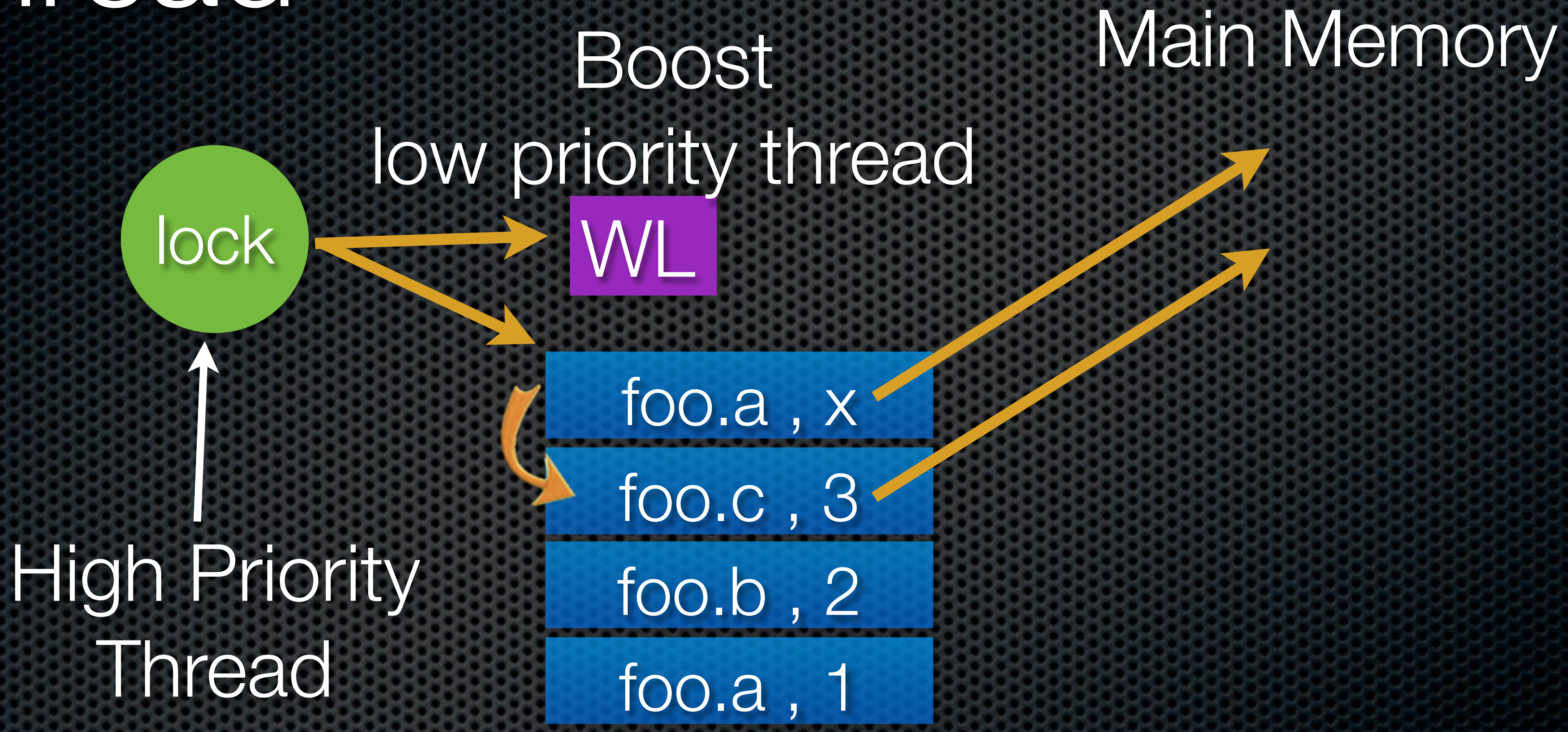
foo.c , 3

foo.b , 2

foo.a , 1

High Priority Thread

Acquisition by higher priority thread



Boost
low priority thread

Main Memory

lock

WL

foo.a , x

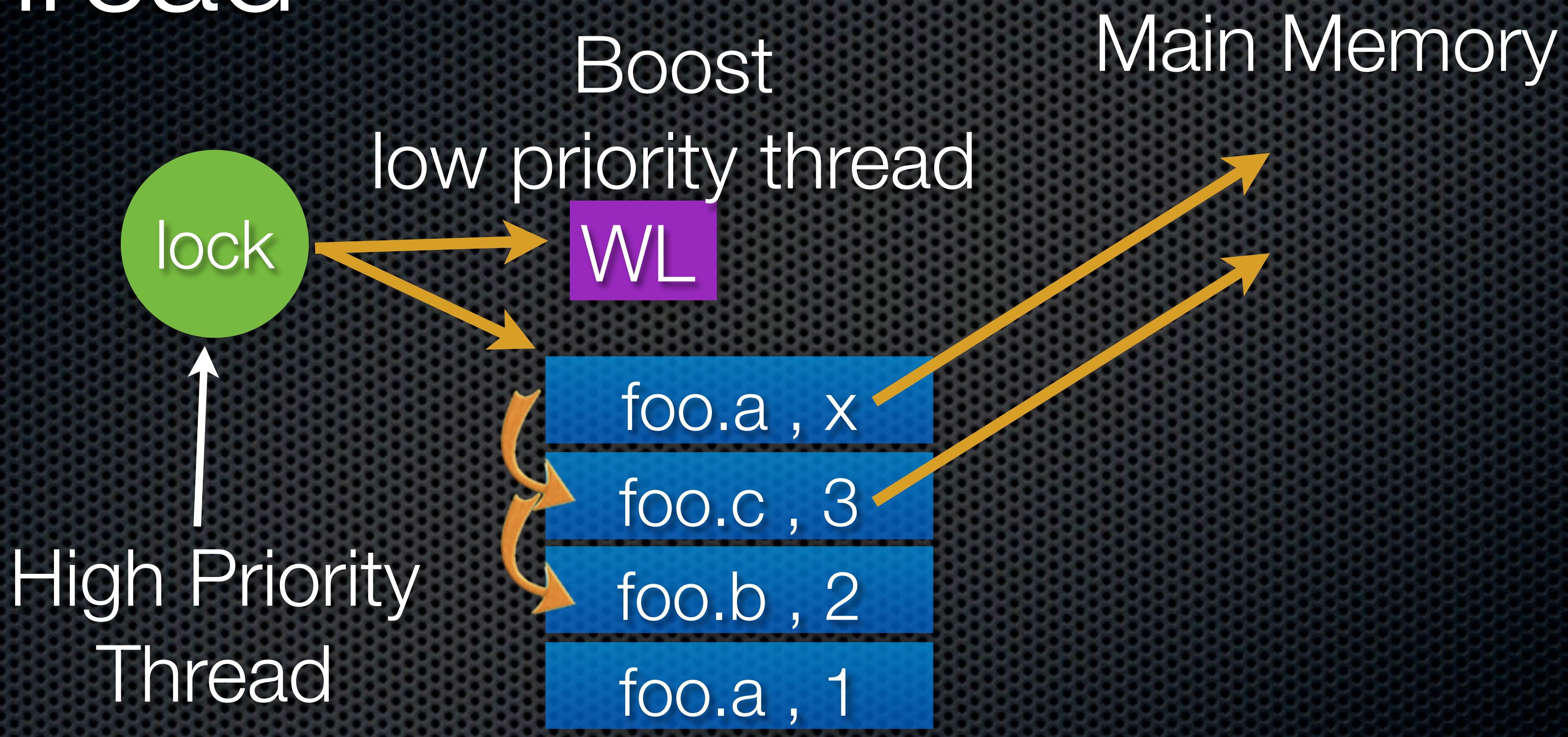
foo.c , 3

foo.b , 2

foo.a , 1

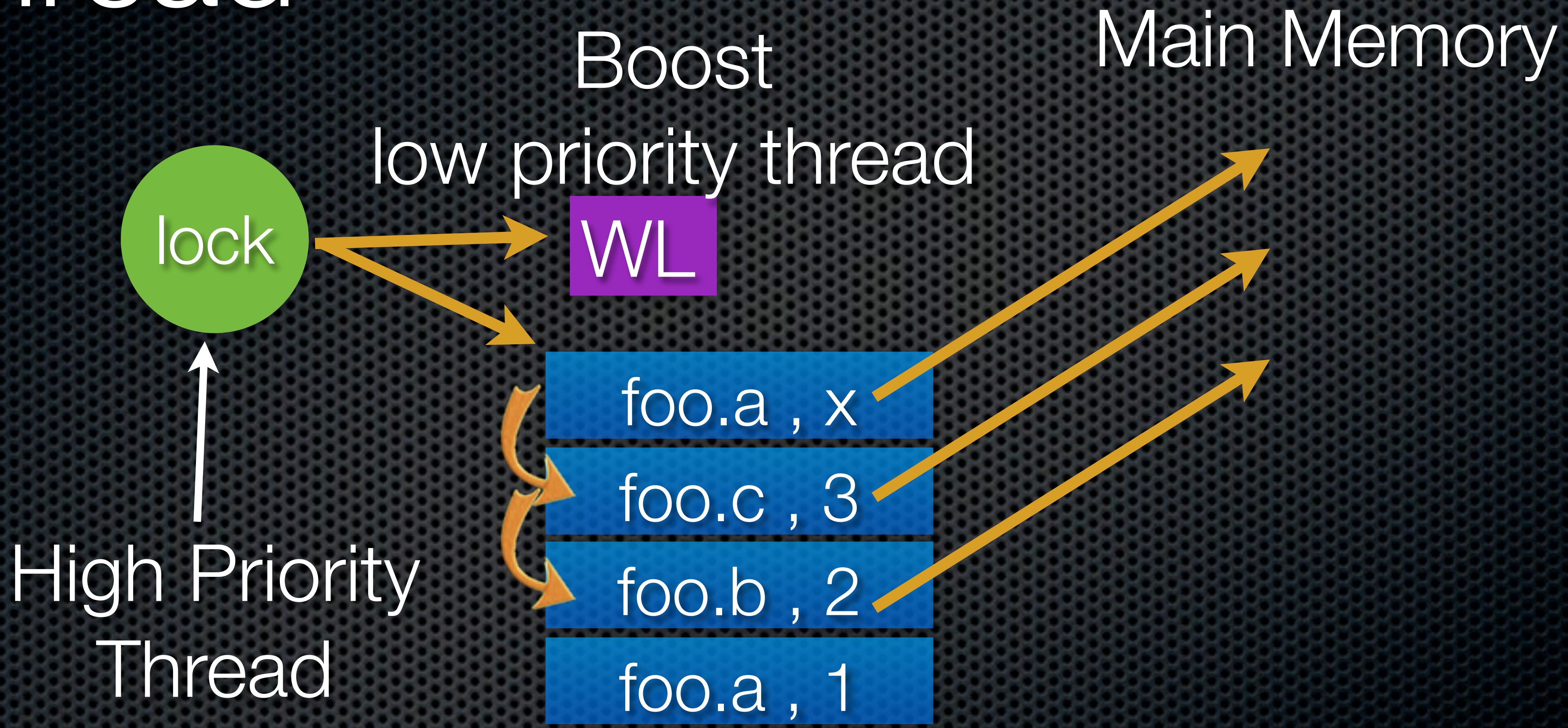
High Priority
Thread

Acquisition by higher priority thread



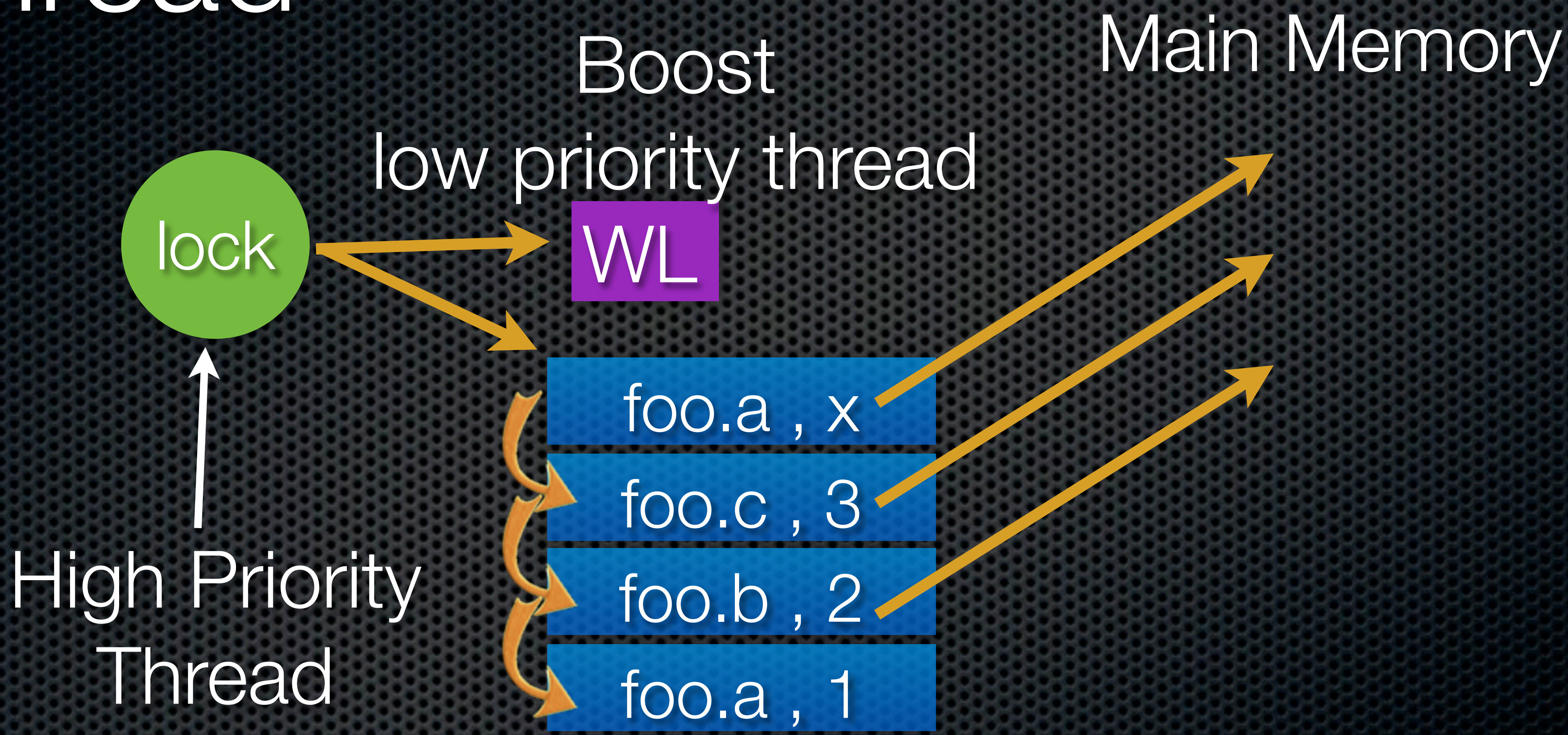
Fiji
Systems LLC

Acquisition by higher priority thread



Fiji
Systems LLC

Acquisition by higher priority thread



Boost

low priority thread

Main Memory

lock

WL

foo.a , x

foo.c , 3

foo.b , 2

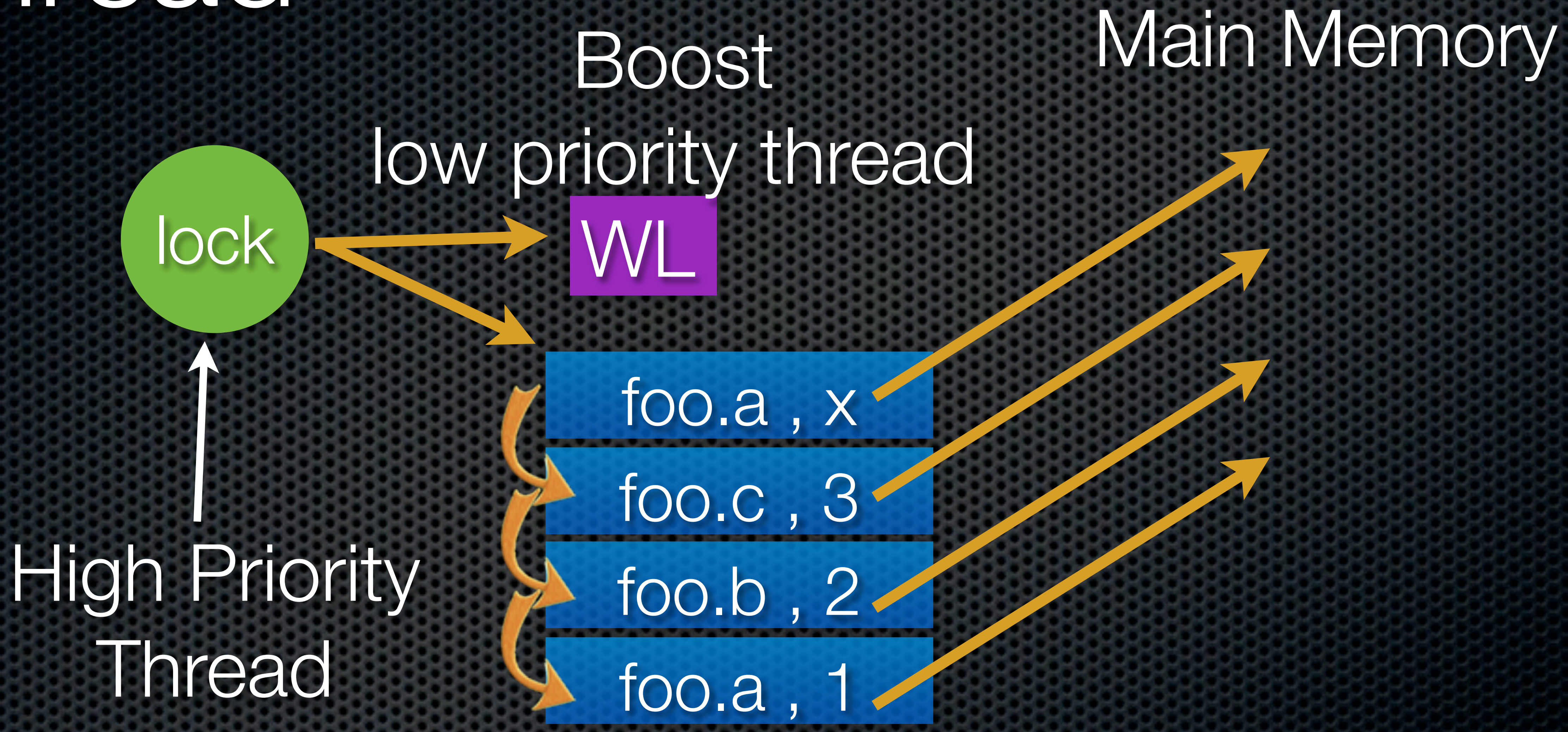
foo.a , 1

High Priority Thread

Fiji

Systems LLC

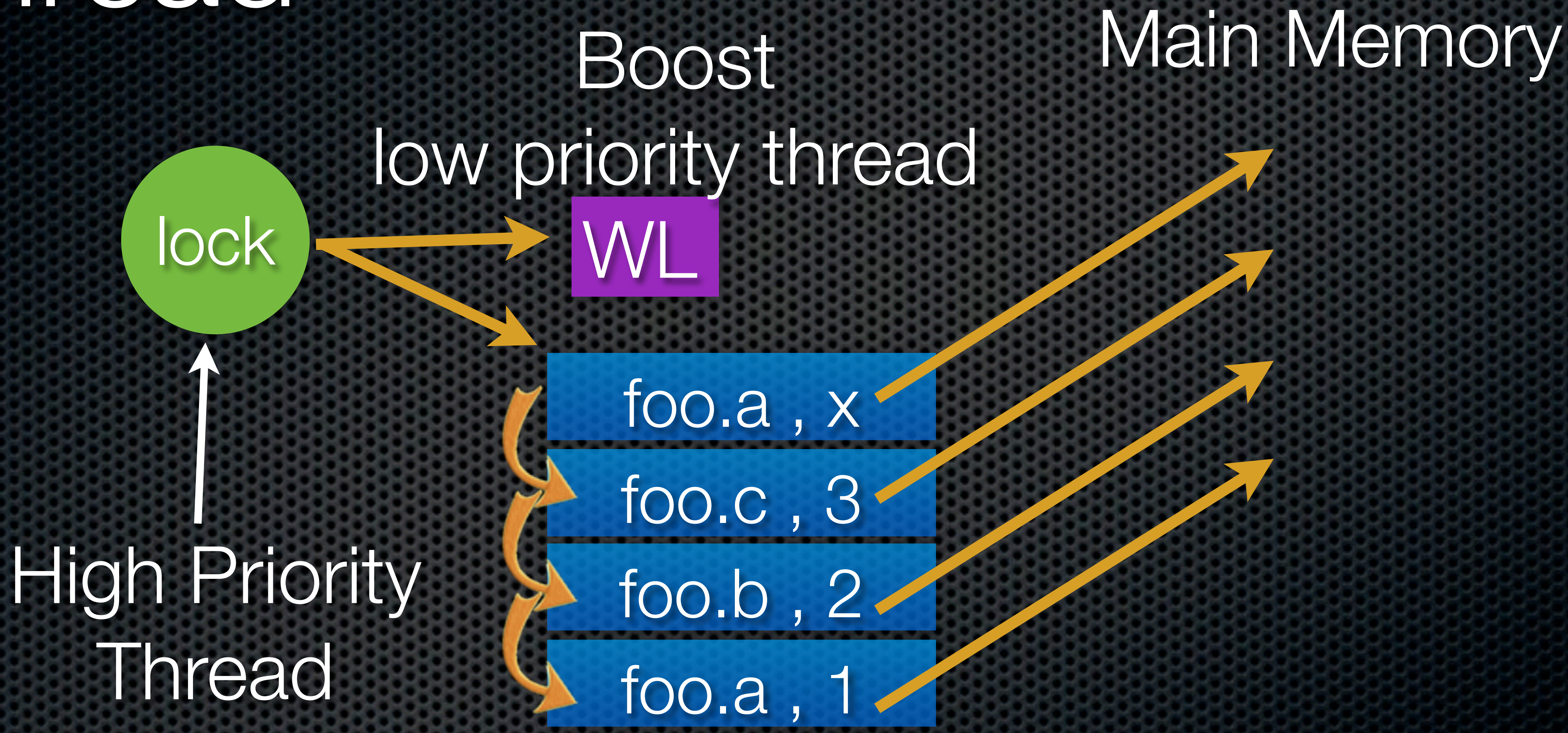
Acquisition by higher priority thread



Fiji
Systems LLC

Wednesday, August 18, 2010

Acquisition by higher priority thread



Boost

low priority thread

Main Memory

lock

WL

foo.a , x

foo.c , 3

foo.b , 2

foo.a , 1

High Priority Thread

High Priority Thread Acquires Lock

Complexity Costs - Write Logging Low Priority Thread

- Reads: constant
- Writes: read + log(size of write log)
- Commit: constant
- Acquisition: size of write log + context switches
- Memory: size of write log

Fiji
Systems LLC

Questions?

Fiji
Systems LLC