LOCKDEP

Runtime lock dependency correctness checker

Byungchul Park max.byungchul.park@gmail.com

LOCKDEP

Runtime lock dependency correctness checker

Byungchul Park max.byungchul.park@gmail.com

- What What Lockdep does
- How How Lockdep works
- Why Why we use Lockdep

- Config How to enable Lockdep
- Practice How to use Lockdep
- Appendix Enhanced Lockdep

Contents

What

Thread X	Thread Y
spin_lock A	
	spin_lock B
spin_lock B	
	spin_lock A
A_data = A_data + B_data	
	B_data = A_data * B_data
spin_unlock B	
	spin_unlock A
spin_unlock A	
	spin_unlock B

Deadlock ?

Thread X	Thread Y
spin_lock A	
	spin_lock B
spin_lock B	
	spin_lock A
A_data = A_data + B_data	
	B_data = A_data * B_data
spin_unlock B	
	spin_unlock A
spin_unlock A	
	spin_unlock B

Deadlock ?

Thread X	Thread Y
spin_lock A // Acquired successfully	
	spin_lock B
spin_lock B	
	spin_lock A
A_data = A_data + B_data	
	B_data = A_data * B_data
spin_unlock B	
	spin_unlock A
spin_unlock A	
	spin_unlock B

Deadlock ?

Thread X	Thread Y
spin_lock A // Acquired successfully	
	spin_lock B // Acquired successfully
spin_lock B	
	spin_lock A
A_data = A_data + B_data	
	B_data = A_data * B_data
spin_unlock B	
	spin_unlock A
spin_unlock A	
	spin_unlock B

Deadlock ?

Thread X	Thread Y
spin_lock A // Acquired successfully	
	spin_lock B // Acquired successfully
spin_lock B // Wait for B to be unlocked	
	spin_lock A
A_data = A_data + B_data	
	B_data = A_data * B_data
spin_unlock B	
	spin_unlock A
spin_unlock A	
	spin_unlock B

Deadlock ?

Thread X	Thread Y
spin_lock A // Acquired successfully	
	spin_lock B // Acquired successfully
spin_lock B // Wait for B to be unlocked	
	spin_lock A // Wait for A to be unlocked
A_data = A_data + B_data	
	B_data = A_data * B_data
spin_unlock B	
	spin_unlock A
spin_unlock A	
	spin_unlock B

Deadlock ?

Thread X	Thread Y
spin_lock A // Acquired successfully	
	spin_lock B // Acquired successfully
spin_lock B // Wait for B to be unlocked	
	spin_lock A // Wait for A to be unlocked
A_data = A_data + B_data	
	B_data = A_data * B_data
spin_unlock B	
	spin_unlock A
spin_unlock A	
	spin_unlock B

Obvious Deadlock !

Byungchul Park max.byungchul.park@gmail.com

Thread X	Thread Y
spin_lock A	
	spin_lock B
spin_lock B	
	spin_lock A
A_data = A_data + B_data	
	B_data = A_data * B_data
spin_unlock B	
	spin_unlock A
spin_unlock A	
	spin_unlock B

What if ?

Thread X	Thread Y
spin_lock A	
spin_lock B	
A_data = A_data + B_data	
spin_unlock B	
spin_unlock A	
	spin_lock B
	spin_lock A
	B_data = A_data * B_data
	spin_unlock A
	spin_unlock B

What if ?

Thread X	Thread Y
spin_lock A	
spin_lock B	
A_data = A_data + B_data	
spin_unlock B	
spin_unlock A	
	spin_lock B
	spin_lock A
	B_data = A_data * B_data
	spin_unlock A
	spin_unlock B

What if ?

Thread X	Thread Y
spin_lock A // Acquired successfully	
spin_lock B	
A_data = A_data + B_data	
spin_unlock B	
spin_unlock A	
	spin_lock B
	spin_lock A
	B_data = A_data * B_data
	spin_unlock A
	spin_unlock B

What if ?

Thread X	Thread Y
spin_lock A // Acquired successfully	
spin_lock B // Acquired successfully	
A_data = A_data + B_data	
spin_unlock B	
spin_unlock A	
	spin_lock B
	spin_lock A
	B_data = A_data * B_data
	spin_unlock A
	spin_unlock B

What if ?

Thread X	Thread Y
spin_lock A // Acquired successfully	
spin_lock B // Acquired successfully	
A_data = A_data + B_data	
spin_unlock B	
spin_unlock A	
	spin_lock B
	spin_lock A
	B_data = A_data * B_data
	spin_unlock A
	spin_unlock B

What if ?

Thread X	Thread Y
spin_lock A // Acquired successfully	
spin_lock B // Acquired successfully	
A_data = A_data + B_data	
spin_unlock B // Released successfully	
spin_unlock A	
	spin_lock B
	spin_lock A
	B_data = A_data * B_data
	spin_unlock A
	spin_unlock B

What if ?

Thread X	Thread Y
spin_lock A // Acquired successfully	
spin_lock B // Acquired successfully	
A_data = A_data + B_data	
spin_unlock B // Released successfully	
spin_unlock A // Released successfully	
	spin_lock B
	spin_lock A
	B_data = A_data * B_data
	spin_unlock A
	spin_unlock B

What if ?

Thread X	Thread Y
spin_lock A // Acquired successfully	
spin_lock B // Acquired successfully	
A_data = A_data + B_data	
spin_unlock B // Released successfully	
spin_unlock A // Released successfully	
	spin_lock B // Acquired successfully
	spin_lock A
	B_data = A_data * B_data
	spin_unlock A
	spin_unlock B

What if ?

Thread X	Thread Y
spin_lock A // Acquired successfully	
spin_lock B // Acquired successfully	
A_data = A_data + B_data	
spin_unlock B // Released successfully	
spin_unlock A // Released successfully	
	spin_lock B // Acquired successfully
	spin_lock A // Acquired successfully
	B_data = A_data * B_data
	spin_unlock A
	spin_unlock B

What if ?

Thread X	Thread Y
spin_lock A // Acquired successfully	
spin_lock B // Acquired successfully	
A_data = A_data + B_data	
spin_unlock B // Released successfully	
spin_unlock A // Released successfully	
	spin_lock B // Acquired successfully
	spin_lock A // Acquired successfully
	B_data = A_data * B_data
	spin_unlock A
	spin_unlock B

What if ?

Thread X	Thread Y
spin_lock A // Acquired successfully	
spin_lock B // Acquired successfully	
A_data = A_data + B_data	
spin_unlock B // Released successfully	
spin_unlock A // Released successfully	
	spin_lock B // Acquired successfully
	spin_lock A // Acquired successfully
	B_data = A_data * B_data
	spin_unlock A // Released successfully
	spin_unlock B

What if ?

Thread X	Thread Y
spin_lock A // Acquired successfully	
spin_lock B // Acquired successfully	
A_data = A_data + B_data	
spin_unlock B // Released successfully	
spin_unlock A // Released successfully	
	spin_lock B // Acquired successfully
	spin_lock A // Acquired successfully
	B_data = A_data * B_data
	spin_unlock A // Released successfully
	spin_unlock B // Released successfully

What if ?

Thread X	Thread Y
spin_lock A // Acquired successfully	
spin_lock B // Acquired successfully	
A_data = A_data + B_data	
spin_unlock B // Released successfully	
spin_unlock A // Released successfully	
	spin_lock B // Acquired successfully
	spin_lock A // Acquired successfully
	B_data = A_data * B_data
	spin_unlock A // Released successfully
	spin_unlock B // Released successfully

No Problem ?

Thread X	Thread Y
spin_lock A // Acquired successfully	
spin_lock B // Acquired successfully	
A_data = A_data + B_data	
spin_unlock B // Released successfully	
spin_unlock A // Released successfully	
	spin_lock B // Acquired successfully
	spin_lock A // Acquired successfully
	B_data = A_data * B_data
	spin_unlock A // Released successfully
	spin_unlock B // Released successfully

Problematic Code !

```
spin_lock A
spin_lock B
A_data = A_data + B_data
spin_unlock B
spin_unlock A
                    spin_lock B
                    spin_lock A
                    B_data = A_data * B_data
                    spin_unlock A
                    spin_unlock B
```

```
                    spin_lock B
spin_lock A
                    spin_lock A
spin_lock B
                    B_data = A_data * B_data
                    spin_unlock A
                    spin_unlock B
A_data = A_data + B_data
spin_unlock B
spin_unlock A
```

```
spin_lock A
                    spin_lock B
spin_lock B
                    spin_lock A
A_data = A_data + B_data
spin_unlock B
spin_unlock A
                    B_data = A_data * B_data
                    spin_unlock A
                    spin_unlock B
```

```
                    spin_lock B
                    spin_lock A
                    B_data = A_data * B_data
                    spin_unlock A
                    spin_unlock B
spin_lock A
spin_lock B
A_data = A_data + B_data
spin_unlock B
spin_unlock A
```

Problematic Code !

Byungchul Park max.byungchul.park@gmail.com

```
spin_lock A
spin_lock B
A_data = A_data + B_data
spin_unlock B
spin_unlock A
                    spin_lock B
                    spin_lock A
                    B_data = A_data * B_data
                    spin_unlock A
                    spin_unlock B
```

```
                    spin_lock B
spin_lock A
                    spin_lock A
spin_lock B
                    B_data = A_data * B_data
                    spin_unlock A
                    spin_unlock B
A_data = A_data + B_data
spin_unlock B
spin_unlock A
```

DEADLOCK

```
spin_lock A
                    spin_lock B
spin_lock B
                    spin_lock A
A_data = A_data + B_data
spin_unlock B
spin_unlock A
                    B_data = A_data * B_data
                    spin_unlock A
                    spin_unlock B
```

```
                    spin_lock B
                    spin_lock A
                    B_data = A_data * B_data
                    spin_unlock A
                    spin_unlock B
spin_lock A
spin_lock B
A_data = A_data + B_data
spin_unlock B
spin_unlock A
```

Problematic Code !

Byungchul Park max.byungchul.park@gmail.com

```
spin_lock A
spin_lock B
A_data = A_data + B_data
spin_unlock B
spin_unlock A
                    spin_lock B
                    spin_lock A
                    B_data = A_data * B_data
                    spin_unlock A
                    spin_unlock B
```

```
                    spin_lock B
spin_lock A
                    spin_lock A
spin_lock B
                    B_data = A_data * B_data
                    spin_unlock A
                    spin_unlock B
A_data = A_data + B_data
spin_unlock B
spin_unlock A
```

```
spin_lock A
                    spin_lock B
spin_lock B
                    spin_lock A
A_data = A_data + B_data
spin_unlock B
spin_unlock A
                    B_data = A_data * B_data
                    spin_unlock A
                    spin_unlock B
```

DEADLOCK

```
                    spin_lock B
                    spin_lock A
                    B_data = A_data * B_data
                    spin_unlock A
                    spin_unlock B
spin_lock A
spin_lock B
A_data = A_data + B_data
spin_unlock B
spin_unlock A
```

Problematic Code !

Byungchul Park max.byungchul.park@gmail.com

```
spin_lock A
spin_lock B
A_data = A_data + B_data
spin_unlock B
spin_unlock A
                    spin_lock B
                    spin_lock A
                    B_data = A_data * B_data
                    spin_unlock A
                    spin_unlock B
```

```
                    spin_lock B
spin_lock A
                    spin_lock A
spin_lock B
                    B_data = A_data * B_data
                    spin_unlock A
                    spin_unlock B
A_data = A_data + B_data
spin_unlock B
spin_unlock A
```

```
spin_lock A
                    spin_lock B
spin_lock B
                    spin_lock A
A_data = A_data + B_data
spin_unlock B
spin_unlock A
                    B_data = A_data * B_data
                    spin_unlock A
                    spin_unlock B
```

```
                    spin_lock B
                    spin_lock A
                    B_data = A_data * B_data
                    spin_unlock A
                    spin_unlock B
spin_lock A
spin_lock B
A_data = A_data + B_data
spin_unlock B
spin_unlock A
```

Problematic Code !

Byungchul Park max.byungchul.park@gmail.com

Lockdep should detect and report,

- Not only obvious deadlock.
- But also problematic code.

What Lockdep Does

Lockdep should detect and report,

- Not only obvious deadlock.
- But also problematic code.

What Lockdep Does

Lockdep should detect and report,

and… works in kernel

- Not only obvious deadlock
- But also problematic code.

What Lockdep Does

Byungchul Park max.byungchul.park@gmail.com

spin_lock A

…

…

…

…

…

…

…

…

…

…

…

…

spin_unlock A

Kernel Lockdep

Byungchul Park max.byungchul.park@gmail.com

spin_lock A

Interrupt

spin_lock A

…

spin_unlock A

…

…

…

…

…

…

…

spin_unlock A

Kernel Lockdep

Byungchul Park max.byungchul.park@gmail.com

spin_lock A

Interrupt

spin_lock B

…

spin_unlock B

…

…

…

…

…

…

…

spin_unlock A

Kernel Lockdep

Byungchul Park max.byungchul.park@gmail.com

spin_lock A

Interrupt

spin_lock B

…

spin_unlock B

bottom-half

spin_lock A

…

spin_unlock A

…

…

…

spin_unlock A

Kernel Lockdep

Byungchul Park max.byungchul.park@gmail.com

spin_lock A

Interrupt

spin_lock B

…

spin_unlock B

bottom-half

spin_lock C

…

spin_unlock C

…

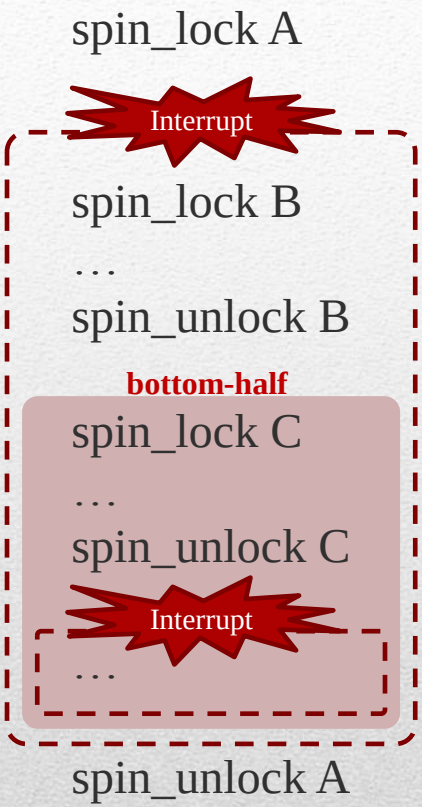
…

…

spin_unlock A

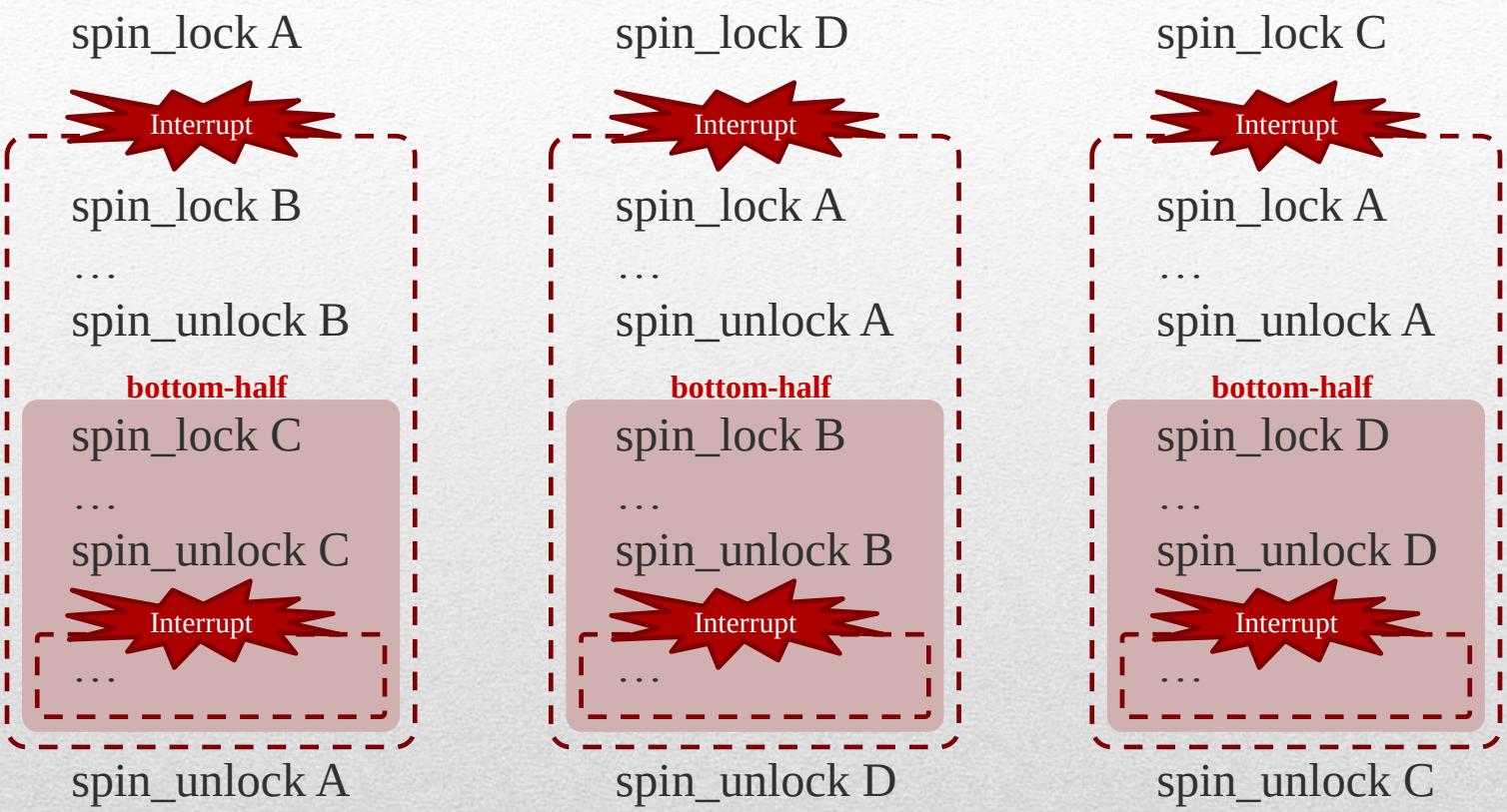
Kernel Lockdep

Byungchul Park max.byungchul.park@gmail.com



Kernel Lockdep

Byungchul Park max.byungchul.park@gmail.com



Kernel Lockdep

Byungchul Park max.byungchul.park@gmail.com

Kernel Lockdep,

- considers all type of contexts,
- considers all possible scenarios,
- detects and reports problematic code.

What Kernel Lockdep Does

How

Incorrect relationship between dependencies causes a deadlock.

Deadlock Comes From

Byungchul Park max.byungchul.park@gmail.com

Incorrect relationship between dependencies causes a deadlock.

Deadlock Comes From

Byungchul Park max.byungchul.park@gmail.com

Incorrect relationship between dependencies causes a deadlock.

Deadlock Comes From

Byungchul Park max.byungchul.park@gmail.com

lock A

lock B (while holding A.)

= A depends on B. (A → B)

Define Dependency

```
spin_lock A
spin_lock B
…
spin_unlock B
spin_unlock A
```

Example

spin_lock A
spin_lock B
…
spin_unlock B
spin_unlock A

Example

spin_lock A
spin_lock B
…
spin_unlock B
spin_unlock A

A → B

Example

Global Space

Build Dependency Graph

Byungchul Park max.byungchul.park@gmail.com

spin_lock A
spin_lock B
…
spin_unlock B
spin_unlock A

A → B

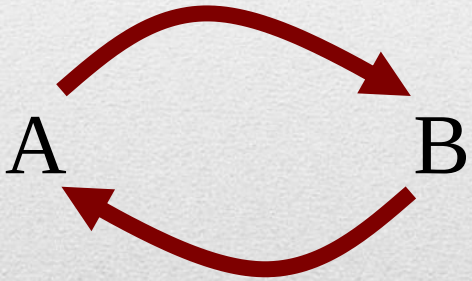
Example

spin_lock B
spin_lock A
…
spin_unlock A
spin_unlock B

B → A

Example

Global Space



Build Dependency Graph

Byungchul Park max.byungchul.park@gmail.com

Circular Dependencies

Byungchul Park max.byungchul.park@gmail.com

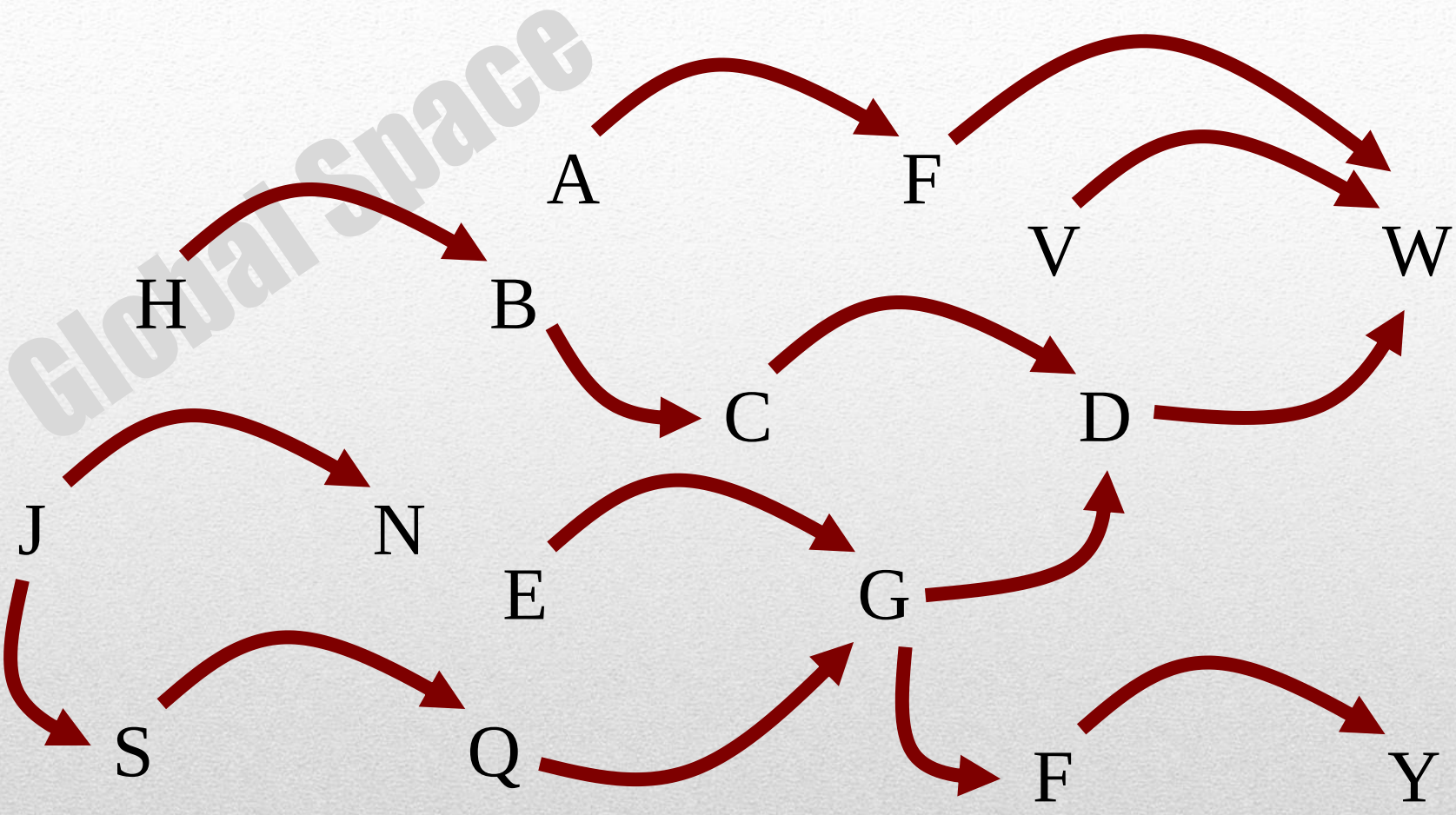
Global Space

Report it !

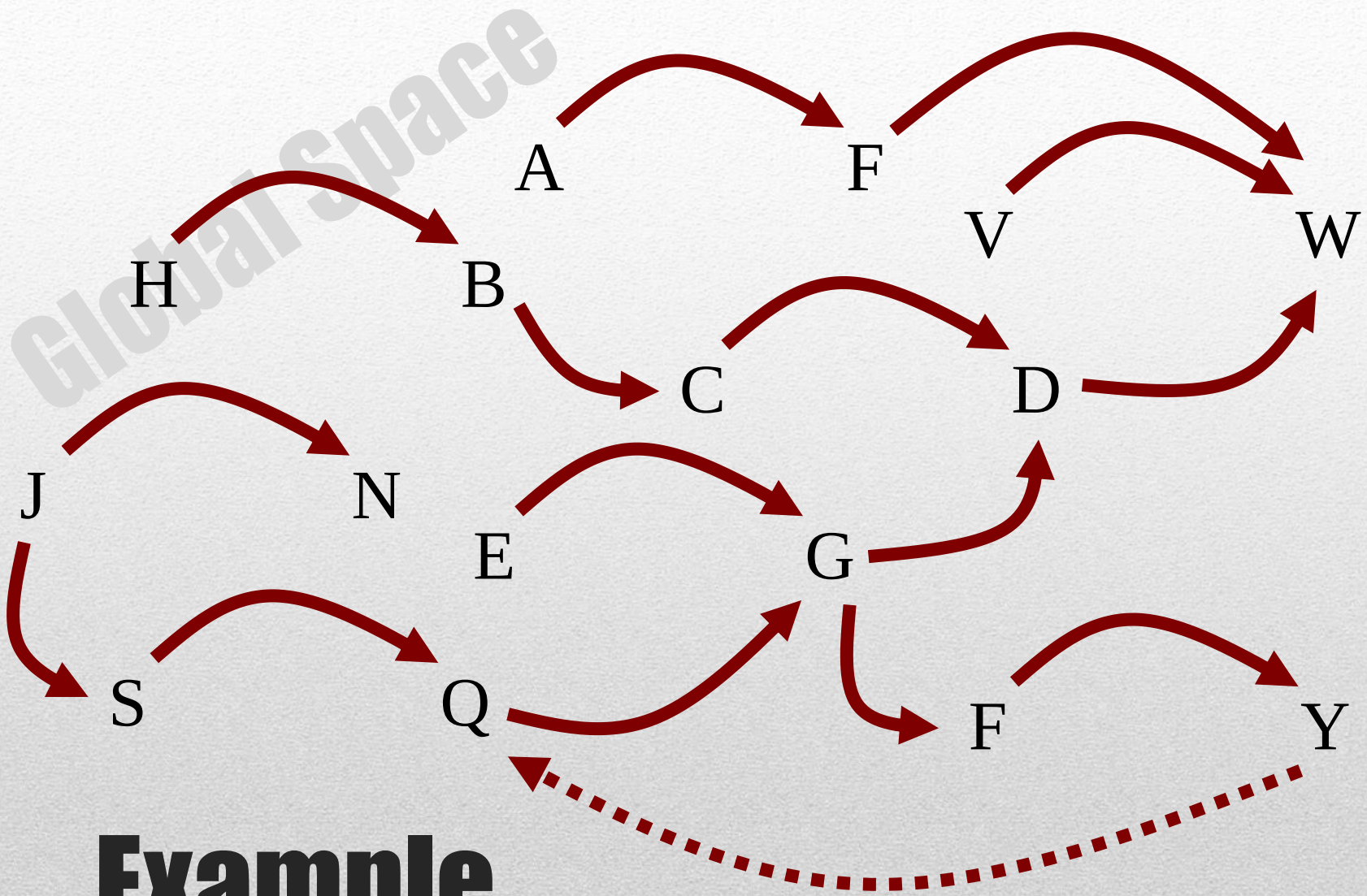
A B

Circular Dependencies

Byungchul Park max.byungchul.park@gmail.com

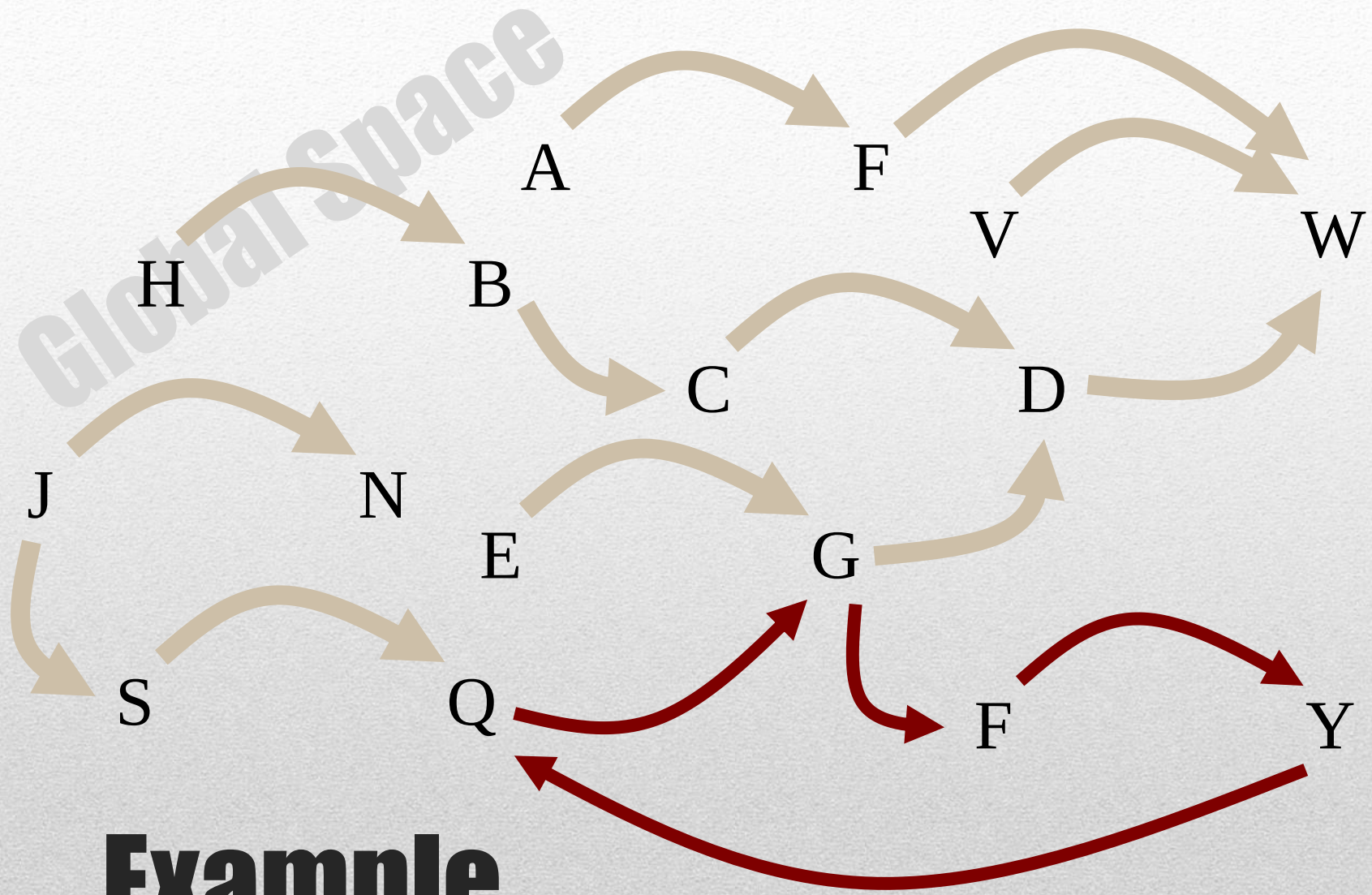


Example

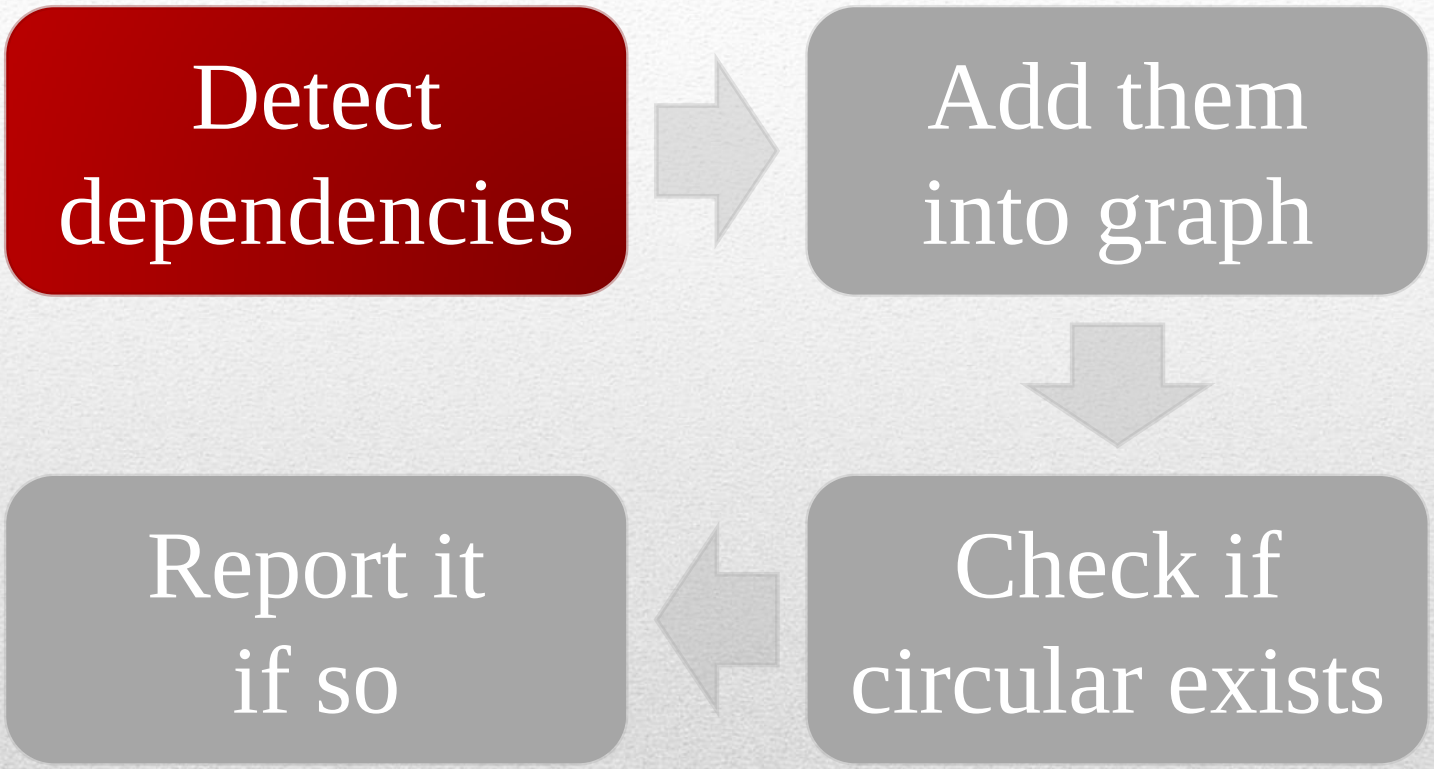


Example

Byungchul Park max.byungchul.park@gmail.com

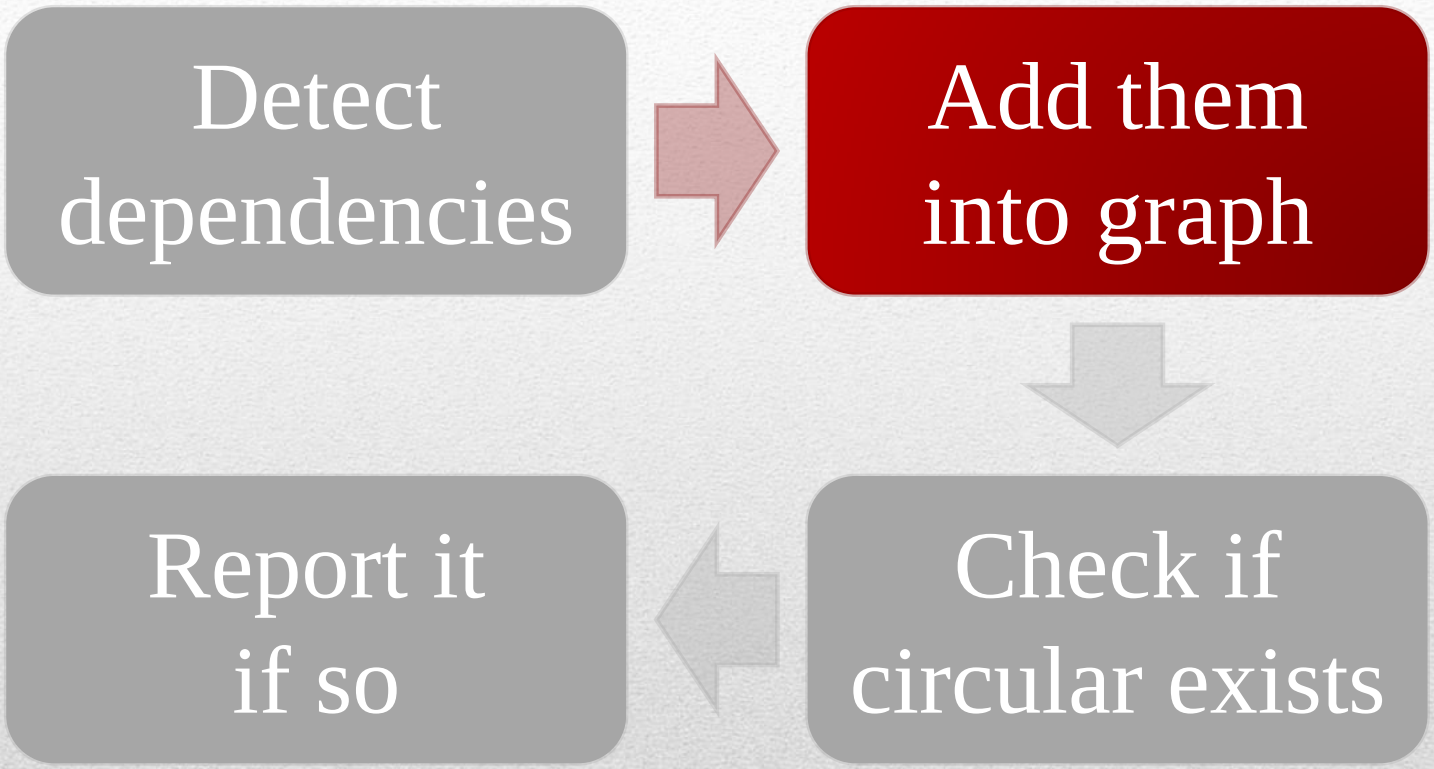


Example



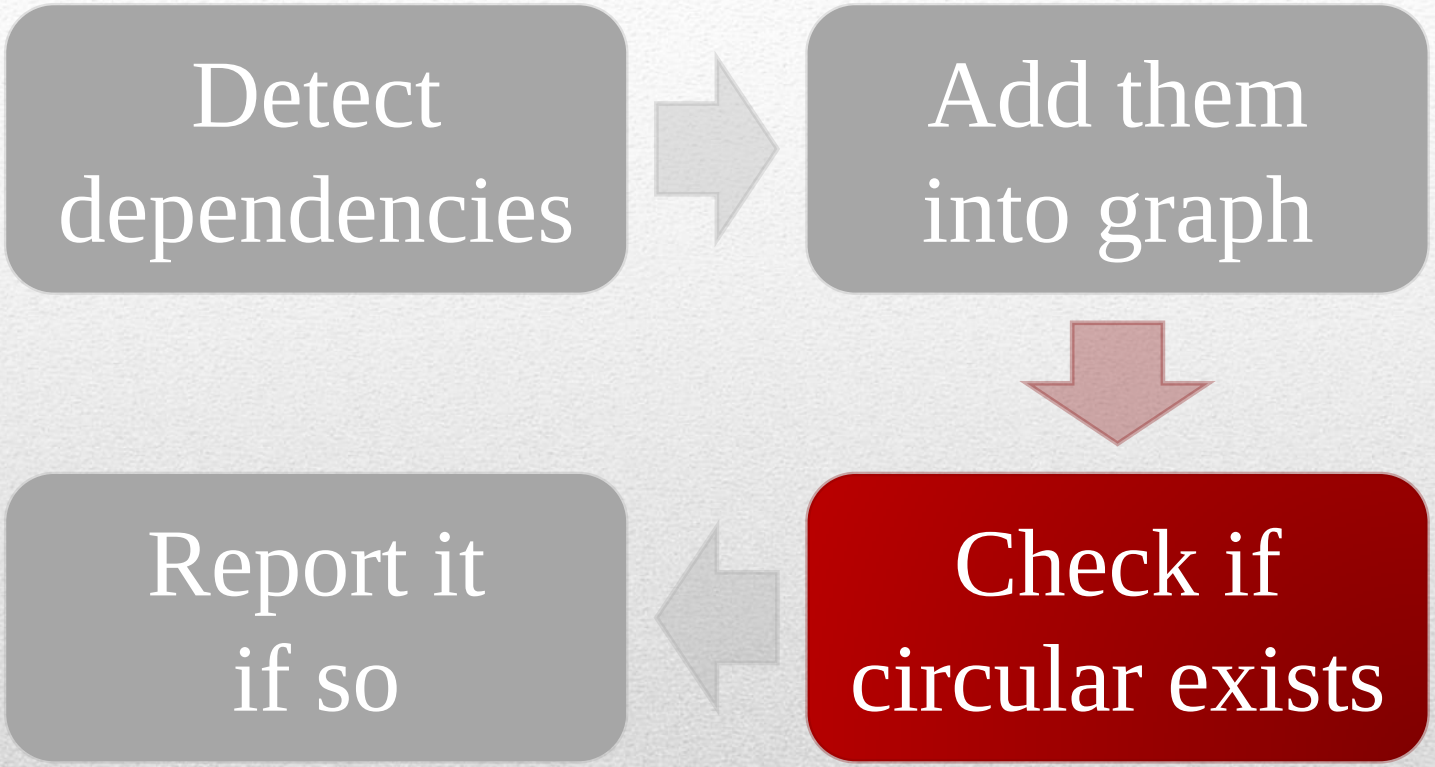
How Lockdep Works

Byungchul Park max.byungchul.park@gmail.com



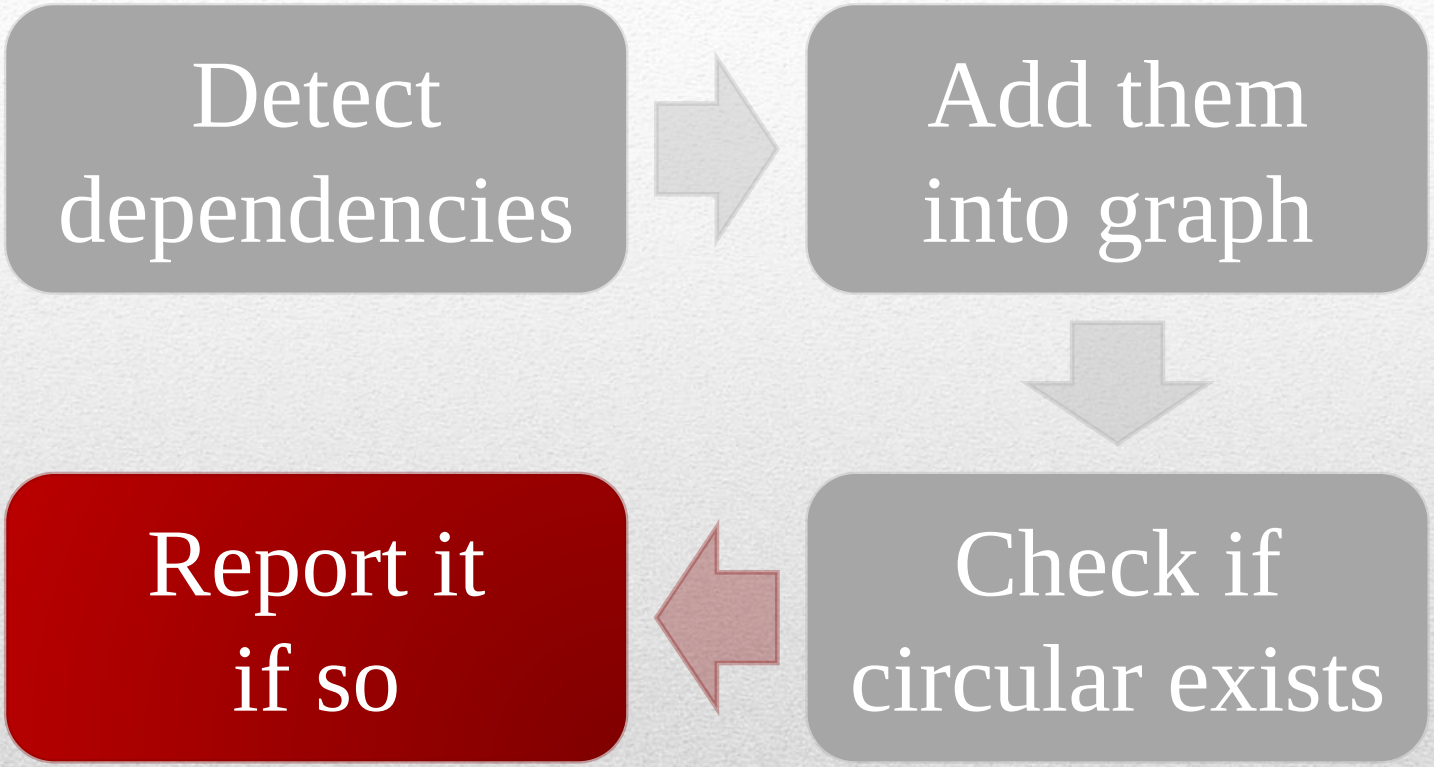
How Lockdep Works

Byungchul Park max.byungchul.park@gmail.com



How Lockdep Works

Byungchul Park max.byungchul.park@gmail.com



How Lockdep Works

Byungchul Park max.byungchul.park@gmail.com

spin_lock A

…

spin_lock A

…

spin_unlock A

…

spin_unlock A

Example

AA Deadlock

Byungchul Park max.byungchul.park@gmail.com

spin_lock A

…

spin_lock A

…

spin_unlock A

…

spin_unlock A

Example

spin_lock A

…

…

…

…

…

spin_unlock A

Example

spin_lock A

Interrupt

spin_lock A

…

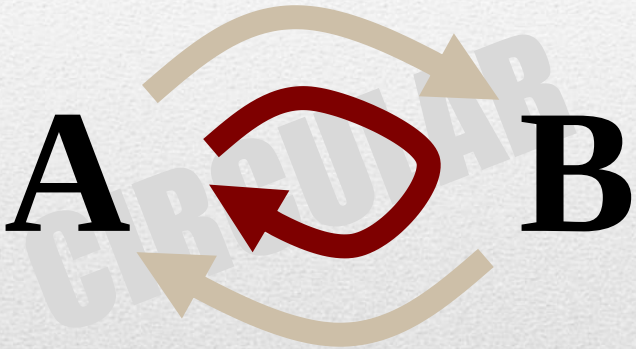
spin_unlock A

spin_unlock A

Example

spin_lock A	spin_lock B
…	…
spin_lock B	spin_lock A
…	…
spin_unlock B	spin_unlock A
…	…
spin_unlock A	spin_unlock B

Example



ABBA Deadlock

Byungchul Park max.byungchul.park@gmail.com

spin_lock A	spin_lock B
…	…
spin_lock B	spin_lock A
…	…
spin_unlock B	spin_unlock A
…	…
spin_unlock A	spin_unlock B

Example

spin_lock A	spin_lock B
…	…
spin_lock B	…
…	…
spin_unlock B	…
…	…
spin_unlock A	spin_unlock B

Example

spin_lock A	spin_lock B
…	Interrupt
spin_lock B	spin_lock A
…	…
spin_unlock B	spin_unlock A
…	
spin_unlock A	spin_unlock B

Example

spin_lock A	spin_lock B
…	…
…	spin_lock A
…	…
…	spin_unlock A
…	…
spin_unlock A	spin_unlock B

Example

spin_lock A	spin_lock B
Interrupt	…
spin_lock B	spin_lock A
…	…
spin_unlock B	spin_unlock A
	…
spin_unlock A	spin_unlock B

Example

spin_lock A	spin_lock B
…	…
…	…
…	…
…	…
…	…
spin_unlock A	spin_unlock B

Example

spin_lock A

Interrupt

spin_lock B

…

spin_unlock B

spin_unlock A

spin_lock B

Interrupt

spin_lock A

…

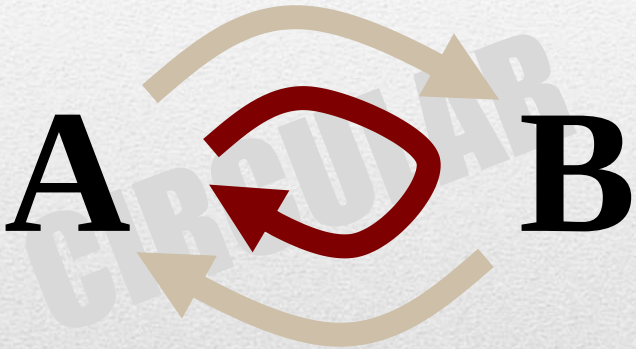
spin_unlock A

spin_unlock B

Example

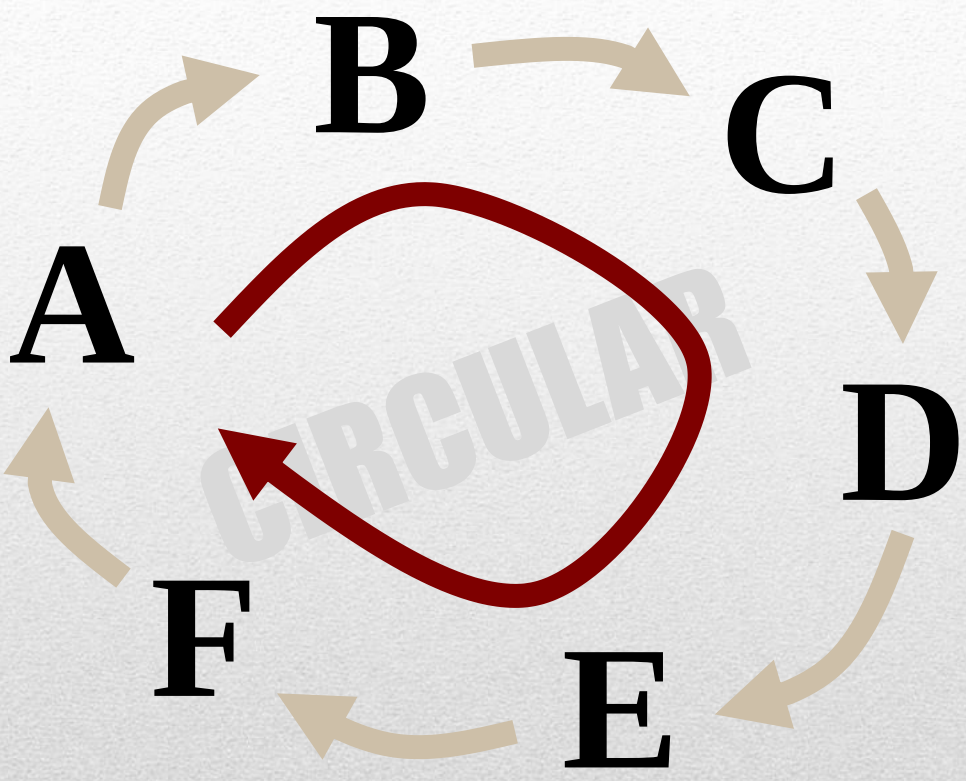
AA Deadlock

Byungchul Park max.byungchul.park@gmail.com



ABBA Deadlock

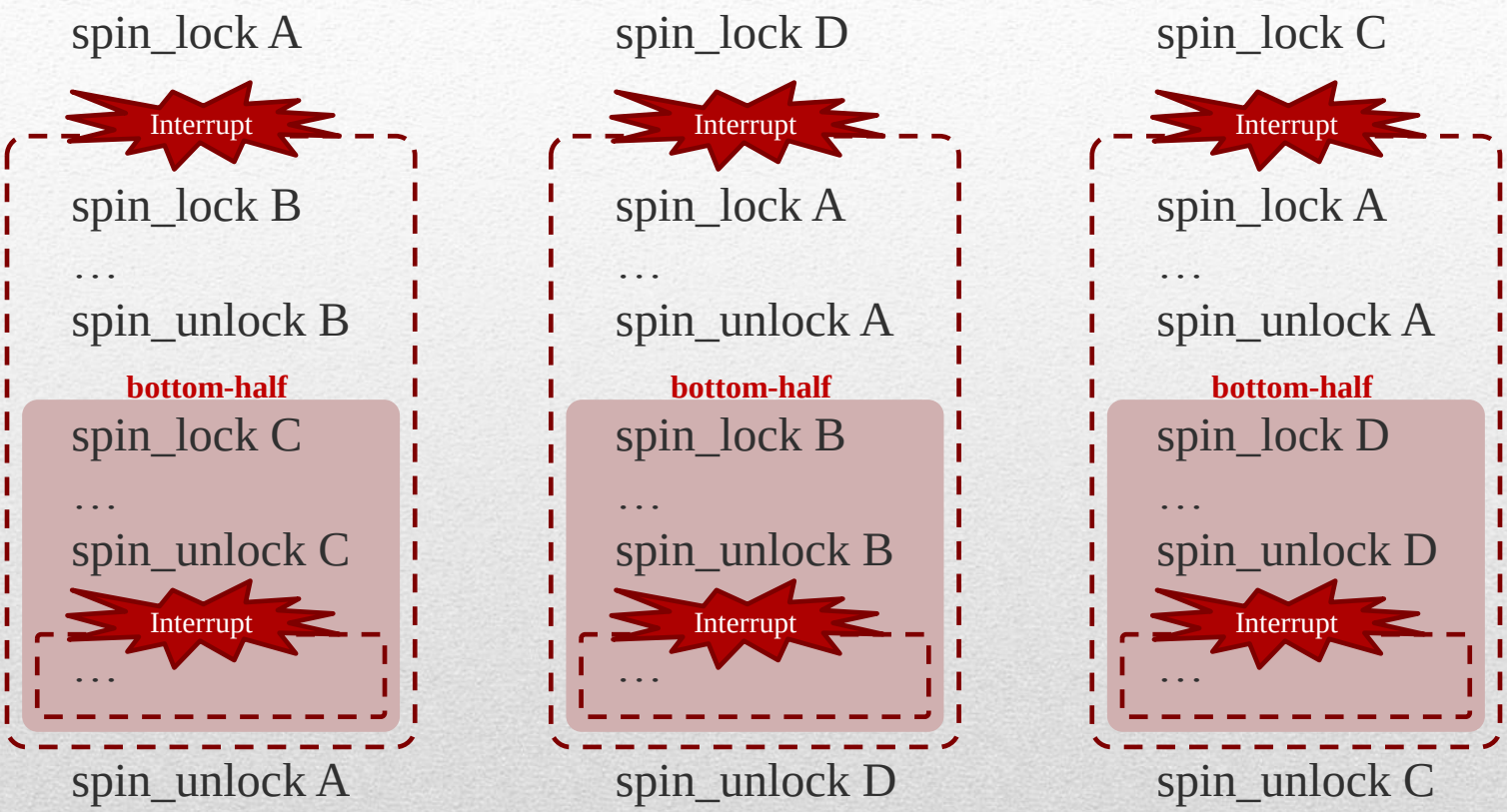
Byungchul Park max.byungchul.park@gmail.com



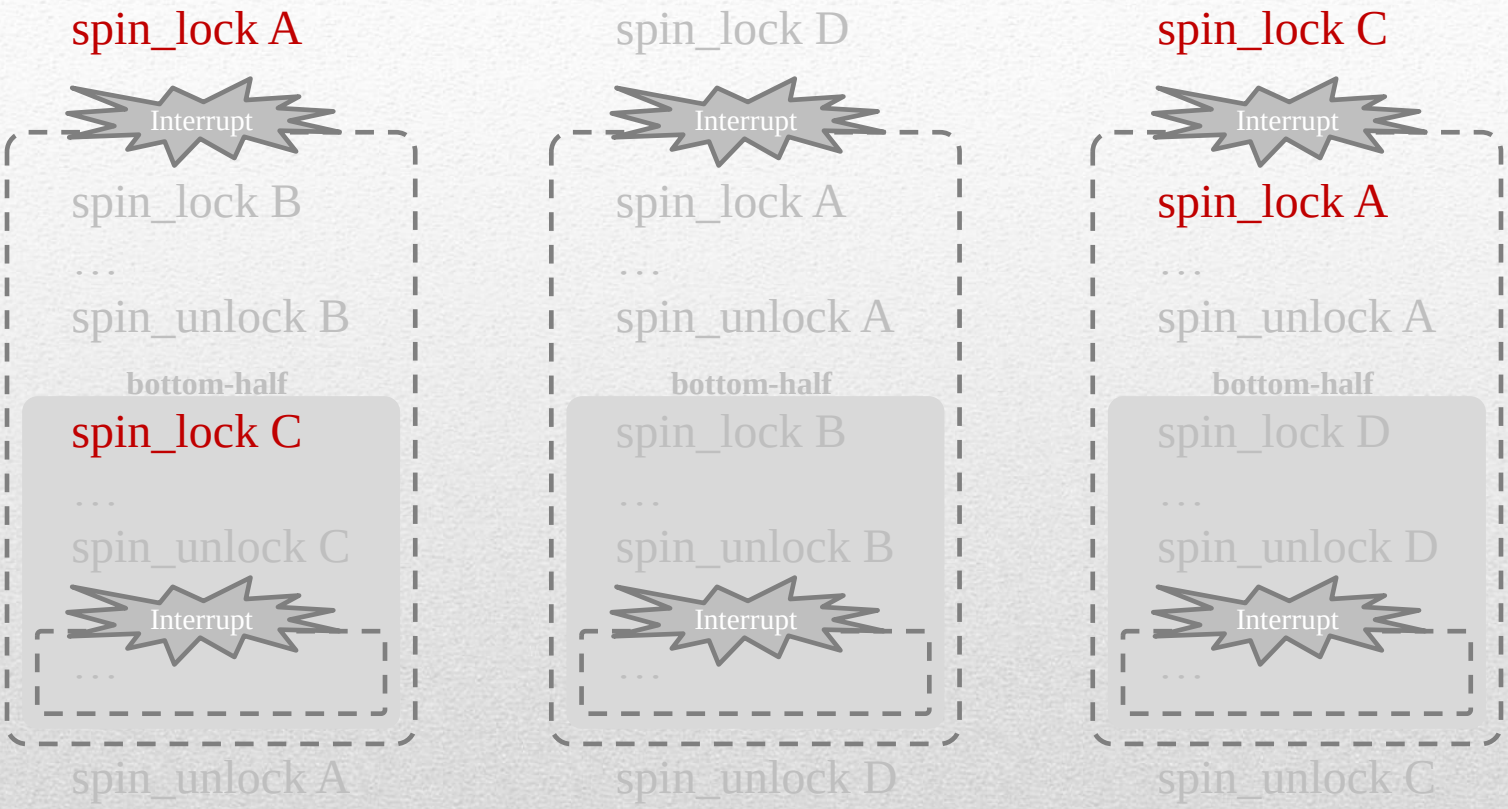
General Deadlock

Byungchul Park max.byungchul.park@gmail.com

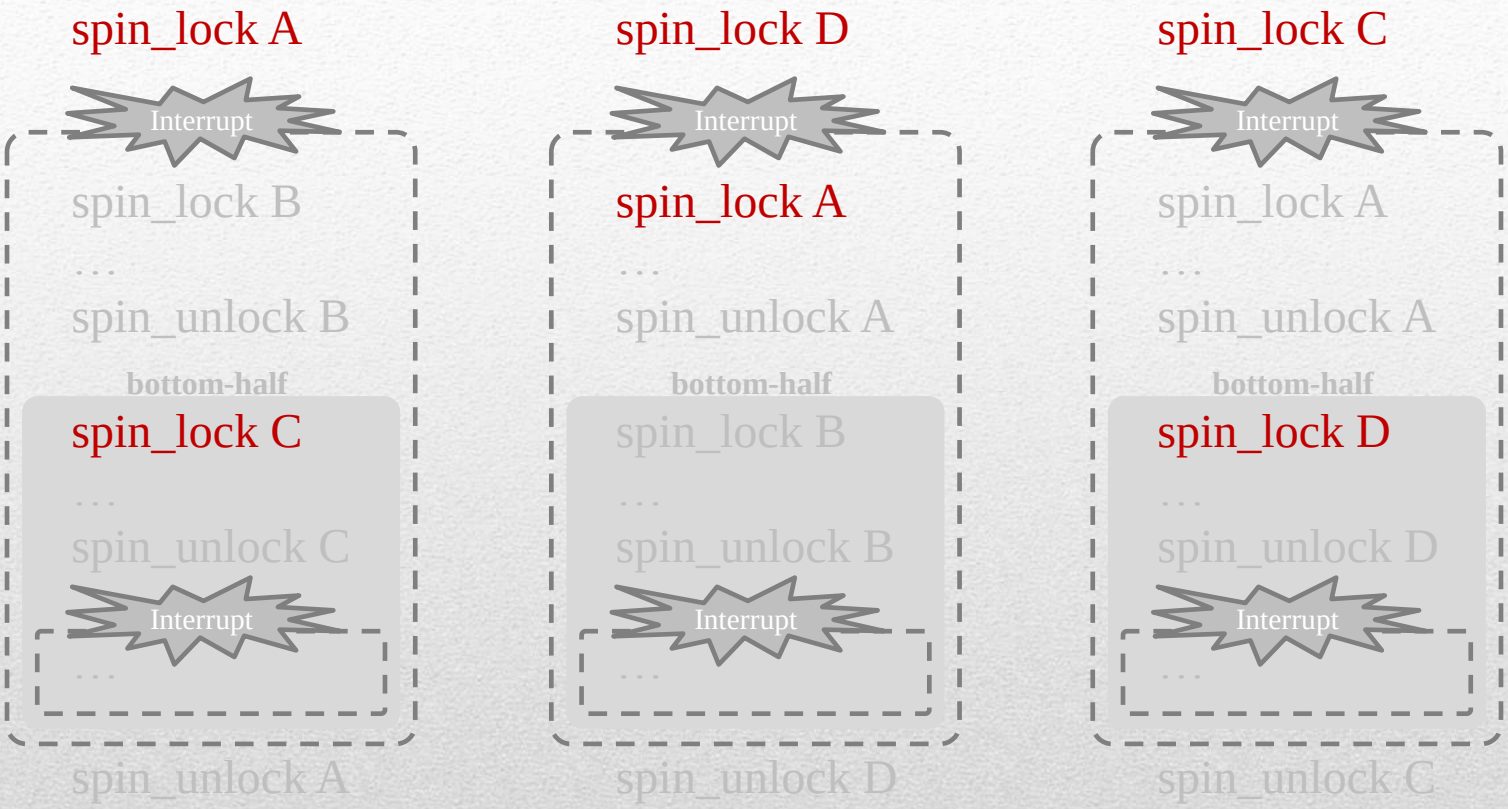
Why


spin_lock A
Interrupt
spin_lock B
…
spin_unlock B
bottom-half
spin_lock C
…
spin_unlock C
Interrupt
…
spin_unlock A
spin_lock D
Interrupt
spin_lock A
…
spin_unlock A
bottom-half
spin_lock B
…
spin_unlock B
Interrupt
…
spin_unlock D
spin_lock C
Interrupt
spin_lock A
…
spin_unlock A
bottom-half
spin_lock D
…
spin_unlock D
Interrupt
…
spin_unlock C

Deadlock ?


spin_lock A
Interrupt
spin_lock B
...
spin_unlock B
bottom-half
spin_lock C
...
spin_unlock C
Interrupt
...
spin_unlock A
spin_lock D
Interrupt
spin_lock A
...
spin_unlock A
bottom-half
spin_lock B
...
spin_unlock B
Interrupt
...
spin_unlock D
spin_lock C
Interrupt
spin_lock A
...
spin_unlock A
bottom-half
spin_lock D
...
spin_unlock D
Interrupt
...
spin_unlock C

Deadlock !



Deadlock !

Byungchul Park max.byungchul.park@gmail.com

```
printk("something…");
  console_lock();
    spin_lock(&sem->lock);
    spin_unlock(&sem->lock);
  console_unlock();
    spin_lock(&sem->lock);
    wake_up_process(task);
      spin_lock(&task->pi_lock);
        spin_dump(&task->pi_lock, "WARN");
          printk("WARN");
            console_lock();
              spin_lock(&sem->lock);
    spin_unlock(&sem->lock);
```

Deadlock ?

```
printk("something…");
  console_lock();
    spin_lock(&sem->lock);
    spin_unlock(&sem->lock);
  console_unlock();
    spin_lock(&sem->lock);
    wake_up_process(task);
      spin_lock(&task->pi_lock);
        spin_dump(&task->pi_lock, "WARN");
          printk("WARN");
            console_lock();
              spin_lock(&sem->lock);
    spin_unlock(&sem->lock);
```

Deadlock !

```
int count;
spin_lock_t a;

void print_genius(void)
{
   spin_lock(&a);
   printk("I am %dth genius!\n", ++count);
   spin_unlock(&a);
}

void kthread_mycode(void)
{
   print_genius();
}
```

Deadlock ?

```
int count;
spin_lock_t a;

void print_genius(void)
{
   spin_lock(&a);
   printk("I am %dth genius!\n", ++count);
   spin_unlock(&a);
}

void kthread_mycode(void)
{
   print_genius();
}
```

```
…

extern void print_genius(void);

void who_am_I(void)
{
   print_genius();
}

void foo_interrupt_handler(void)
{
   who_am_I();
}
```

Deadlock ?

```
int count;
spin_lock_t a;

void print_genius(void)
{
   spin_lock(&a);
   print[illegible]ius!\n", ++count);
   spin_unlock(&a);
}

void kthread_mycode(void)
{
   print_genius();
}
```

Interrupt

```
…

extern void print_genius(void);

void who_am_I(void)
{
   print_genius();
}

void foo_interrupt_handler(void)
{
   who_am_I();
}
```

Deadlock !

```
void task_migration(struct q *a,
                    struct q *b)
{
   spin_lock_irq(&a->lock);
   spin_lock_irq(&b->lock);
   …
   spin_unlock_irq(&b->lock);
   spin_unlock_irq(&a->lock);
}
```

```
void load_balance(void)
{
   …
   if (condition 1)
      task_migration(&q0, &q1);
   else if (condition 2)
      task_migration(&q1, &q0);
   …
}
```

Deadlock ?

```
void task_migration(struct q *a,
                    struct q *b)
{
  spin_lock_irq(&a->lock);
  spin_lock_irq(&b->lock);
  …
  spin_unlock_irq(&b->lock);
  spin_unlock_irq(&a->lock);
}
```

```
void load_balance(void)
{
  …
  if (condition 1)
    task_migration(&q0, &q1);
  else if (condition 2)
    task_migration(&q1, &q0);
  …
}
```

Deadlock ?

```
void task_migration(struct q *a,
                    struct q *b)
{
  spin_lock_irq(&a->lock);
  spin_lock_irq(&b->lock);
  …
  spin_unlock_irq(&b->lock);
  spin_unlock_irq(&a->lock);
}
```

```
void load_balance(void)
{
  …
  if (condition 1)
    task_migration(&q0, &q1);
  else if (condition 2)
    task_migration(&q1, &q0);
  …
}
```

q0->lock

q1->lock

Deadlock ?

```
void task_migration(struct q *a,
                    struct q *b)
{
  spin_lock_irq(&a->lock);
  spin_lock_irq(&b->lock);
  …
  spin_unlock_irq(&b->lock);
  spin_unlock_irq(&a->lock);
}
```

```
void load_balance(void)
{
  …
  if (condition 1)
    task_migration(&q0, &q1);
  else if (condition 2)
    task_migration(&q1, &q0);
  …
}
```

Deadlock ?

```
void task_migration(struct q *a,
                    struct q *b)
{
  spin_lock_irq(&a->lock);
  spin_lock_irq(&b->lock);
  …
  spin_unlock_irq(&b->lock);
  spin_unlock_irq(&a->lock);
}
```

```
void load_balance(void)
{
  …
  if (condition 1)
    task_migration(&q0, &q1);
  else if (condition 2)
    task_migration(&q1, &q0);
  …
}
```

q0->lock

q1->lock

Deadlock ?

```
void task_migration(struct q *a,
                    struct q *b)
{
  spin_lock_irq(&a->lock);
  spin_lock_irq(&b->lock);
  …
  spin_unlock_irq(&b->lock);
  spin_unlock_irq(&a->lock);
}
```

```
void load_balance(void)
{
  …
  if (condition 1)
    task_migration(&q0, &q1);
  else if (condition 2)
    task_migration(&q1, &q0);
  …
}
```

Deadlock ?

```
void task_migration(struct q *a,
                    struct q *b)
{
  spin_lock_irq(&a->lock);
  spin_lock_irq(&b->lock);
  …
  spin_unlock_irq(&b->lock);
  spin_unlock_irq(&a->lock);
}
```

```
void load_balance(void)
{
  …
  if (condition 1)
    task_migration(&q0, &q1);
  else if (condition 2)
    task_migration(&q1, &q0);
  …
}
```

q0->lock

q1->lock

Deadlock !

```
void task_migration(struct q *a,
                    struct q *b)
{
  spin_lock_irq(&a->lock);
  spin_lock_irq(&b->lock);
  …
  spin_unlock_irq(&b->lock);
  spin_unlock_irq(&a->lock);
}
```

```
void load_balance(void)
{
  …
  if (condition 1)
    task_migration(&q0, &q1);
  else if (condition 2)
    task_migration(&q1, &q0);
  …
}
```

Deadlock !

Byungchul Park max.byungchul.park@gmail.com

```
void task_migration(struct q *a,
                    struct q *b)
{
  if (a < b) {
    spin_lock_irq(&a->lock);
    spin_lock_irq(&b->lock);
  else {
    spin_lock_irq(&b->lock);
    spin_lock_irq(&a->lock);
  }
  …
  spin_unlock_irq(&a->lock);
  spin_unlock_irq(&b->lock);
}
```

Deadlock ?

```
void task_migration(struct q *a,
                    struct q *b)
{
  if (a < b) {
    spin_lock_irq(&a->lock);
    spin_lock_irq(&b->lock);
  else {
    spin_lock_irq(&b->lock);
    spin_lock_irq(&a->lock);
  }
  …
  spin_unlock_irq(&a->lock);
  spin_unlock_irq(&b->lock);
}
```

```
void foo_tasks(struct q *a,
               struct q *b)
{
  if (a->q_id < b->q_id) {
    spin_lock_irq(&a->lock);
    spin_lock_irq(&b->lock);
  else {
    spin_lock_irq(&b->lock);
    spin_lock_irq(&a->lock);
  }
  …
  spin_unlock_irq(&a->lock);
  spin_unlock_irq(&b->lock);
}
```

Deadlock ?

```
void task_migration(struct q *a,
                    struct q *b)
{
  if (a < b) {
    spin_lock_irq(&a->lock);
    spin_lock_irq(&b->lock);
  else {
    spin_lock_irq(&b->lock);
    spin_lock_irq(&a->lock);
  }
  …
  spin_unlock_irq(&a->lock);
  spin_unlock_irq(&b->lock);
}
```

```
void foo_tasks(struct q *a,
               struct q *b)
{
  if (a->q_id < b->q_id) {
    spin_lock_irq(&a->lock);
    spin_lock_irq(&b->lock);
  else {
    spin_lock_irq(&b->lock);
    spin_lock_irq(&a->lock);
  }
  …
  spin_unlock_irq(&a->lock);
  spin_unlock_irq(&b->lock);
}
```

Deadlock ?

```
void task_migration(struct q *a,
                    struct q *b)
{
  if (a < b) {
    spin_lock_irq(&a->lock);
    spin_lock_irq(&b->lock);
  else {
    spin_lock_irq(&b->lock);
    spin_lock_irq(&a->lock);
  }
  …
  spin_unlock_irq(&a->lock);
  spin_unlock_irq(&b->lock);
}
```

a->lock

b->lock

```
void foo_tasks(struct q *a,
               struct q *b)
{
  if (a->q_id < b->q_id) {
    spin_lock_irq(&a->lock);
    spin_lock_irq(&b->lock);
  else {
    spin_lock_irq(&b->lock);
    spin_lock_irq(&a->lock);
  }
  …
  spin_unlock_irq(&a->lock);
  spin_unlock_irq(&b->lock);
}
```

Deadlock ?

```
void task_migration(struct q *a,
                    struct q *b)
{
  if (a < b) {
    spin_lock_irq(&a->lock);
    spin_lock_irq(&b->lock);
  else {
    spin_lock_irq(&b->lock);
    spin_lock_irq(&a->lock);
  }
  ...
  spin_unlock_irq(&a->lock);
  spin_unlock_irq(&b->lock);
}
```

a->lock

b->lock

```
void foo_tasks(struct q *a,
               struct q *b)
{
  if (a->q_id < b->q_id) {
    spin_lock_irq(&a->lock);
    spin_lock_irq(&b->lock);
  else {
    spin_lock_irq(&b->lock);
    spin_lock_irq(&a->lock);
  }
  spin_unlock_irq(&a->lock);
  spin_unlock_irq(&b->lock);
}
```

a->lock

b->lock

Deadlock ?

```
void task_migration(struct q *a,
                    struct q *b)
{
  if (a < b) {
    spin_lock_irq(&a->lock);
    spin_lock_irq(&b->lock);
  else {
    spin_lock_irq(&b->lock);
    spin_lock_irq(&a->lock);
  }
  ...
  spin_unlock_irq(&a->lock);
  spin_unlock_irq(&b->lock);
}
```

```
void foo_tasks(struct q *a,
               struct q *b)
{
  if (a->q_id < b->q_id) {
    spin_lock_irq(&a->lock);
    spin_lock_irq(&b->lock);
  else {
    spin_lock_irq(&b->lock);
    spin_lock_irq(&a->lock);
  }
  ...
  spin_unlock_irq(&a->lock);
  spin_unlock_irq(&b->lock);
}
```

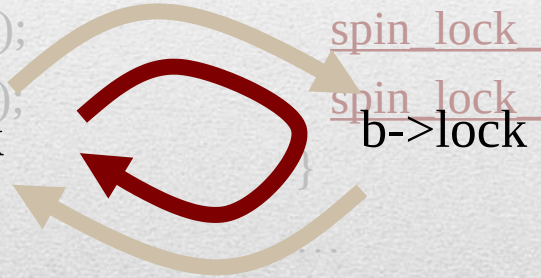
a->lock

b->lock

Deadlock !

```
void task_migration(struct q *a,
                    struct q *b)
{
  if (a < b) {
    spin_lock_irq(&a->lock);
    spin_lock_irq(&b->lock);
  else {
    spin_lock_irq(&b->lock);
    spin_lock_irq(&a->lock);
  }
  ...
  spin_unlock_irq(&a->lock);
  spin_unlock_irq(&b->lock);
}
```

```
void foo_tasks(struct q *a,
               struct q *b)
{
  if (a->q_id < b->q_id) {
    spin_lock_irq(&a->lock);
    spin_lock_irq(&b->lock);
  else {
    spin_lock_irq(&b->lock);
    spin_lock_irq(&a->lock);
  }
  ...
  spin_unlock_irq(&a->lock);
  spin_unlock_irq(&b->lock);
}
```



Deadlock !

```
…
spin_lock (&rq);
…
spin_lock (&task);
…
```

```
…
spin_lock (&console);
…
spin_lock (&terminal);
…
```

```
…
spin_lock (&sem);
…
spin_lock (&rq);
…
```

```
…
spin_lock (&console);
…
spin_lock (&sem);
…
```

```
…
spin_lock (&terminal);
…
spin_lock (&serial);
…
```

```
…
spin_lock (&task);
…
spin_lock (&console);
…
```

Deadlock ?

```
…
spin_lock (&rq);
…
spin_lock (&task);
…
```

rq

task

```
…
spin_lock (&console);
…
spin_lock (&terminal);
…
```

```
…
spin_lock (&sem);
…
spin_lock (&rq);
…
```

```
…
spin_lock (&console);
…
spin_lock (&sem);
…
```

```
…
spin_lock (&terminal);
…
spin_lock (&serial);
…
```

```
…
spin_lock (&task);
…
spin_lock (&console);
…
```

Deadlock ?

…
spin_lock (&rq);
…
spin_lock (&task);
…

…
spin_lock (&console);
…
spin_lock (&sem);
…

…
spin_lock (&console);
…
spin_lock (&terminal);
…

…
spin_lock (&terminal);
…
spin_lock (&serial);
…

…
spin_lock (&sem);
…
spin_lock (&rq);
…

…
spin_lock (&task);
…
spin_lock (&console);
…

rq → task

console → terminal

Deadlock ?

…
spin_lock (&rq);
…
spin_lock (&task);
…

rq → task

…
spin_lock (&console);
…
spin_lock (&sem);
…

…
spin_lock (&console);
…
spin_lock (&terminal);
…

console → terminal

…
spin_lock (&terminal);
…
spin_lock (&serial);
…

…
spin_lock (&sem);
…
spin_lock (&rq);
…

sem → rq

…
spin_lock (&task);
…
spin_lock (&console);
…

Deadlock ?

…
spin_lock (&rq);
…
spin_lock (&task);
…

…
spin_lock (&console);
…
spin_lock (&terminal);
…

…
spin_lock (&sem);
…
spin_lock (&rq);
…

…
spin_lock (&console);
…
spin_lock (&sem);
…

…
spin_lock (&terminal);
…
spin_lock (&serial);
…

…
spin_lock (&task);
…
spin_lock (&console);
…

Deadlock ?

…
spin_lock (&rq);
…
spin_lock (&task);
…

…
spin_lock (&console);
…
spin_lock (&sem);
…

…
spin_lock (&console);
…
spin_lock (&terminal);
…

…
spin_lock (&terminal);
…
spin_lock (&serial);
…

…
spin_lock (&sem);
…
spin_lock (&rq);
…

…
spin_lock (&task);
…
spin_lock (&console);
…

Deadlock ?

…
spin_lock (&rq);
…
spin_lock (&task);
…

rq → task

…
spin_lock (&console);
…
spin_lock (&terminal);
…

console → terminal

…
spin_lock (&sem);
…
spin_lock (&rq);
…

sem → rq

…
spin_lock (&console);
…
spin_lock (&sem);
…

console → sem

…
spin_lock (&terminal);
…
spin_lock (&serial);
…

terminal → serial

…
spin_lock (&task);
…
spin_lock (&console);
…

task → console

Deadlock ?

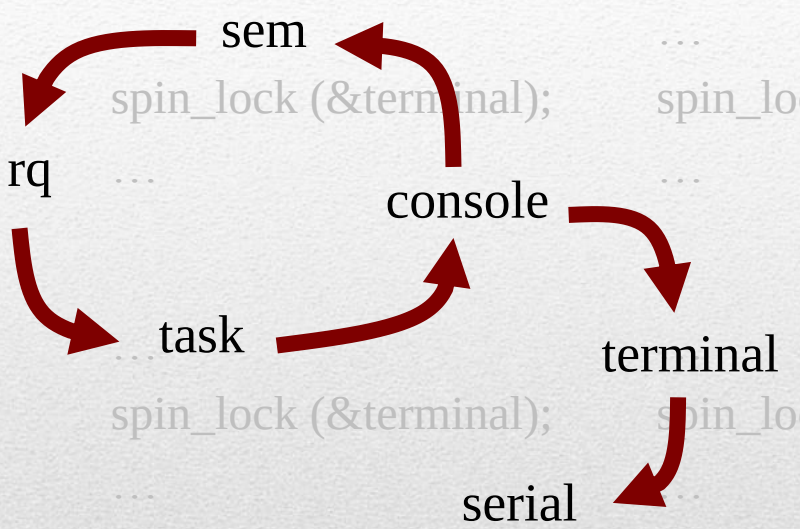
…
spin_lock (&rq);
…
spin_lock (&task);
…

…
spin_lock (&console);
…
spin_lock (&sem);
…

…
spin_lock (&console);
…
spin_lock (&terminal);
…

…
spin_lock (&sem);
…
spin_lock (&rq);
…

…
spin_lock (&terminal);
…
spin_lock (&task);
…
spin_lock (&console);
…



Deadlock !

Byungchul Park max.byungchul.park@gmail.com

Terribly hard to find out,

problematic code in head.

Why Use Lockdep

Byungchul Park max.byungchul.park@gmail.com

Config

CONFIG_TRACE_IRQFLAGS_SUPPORT

CONFIG_STACKTRACE_SUPPORT

CONFIG_LOCKDEP_SUPPORT

(Depending on architectures)

CONFIG_DEBUG_KERNEL

Kernel hacking

Kernel debugging

CONFIG_PROVE_LOCKING

Kernel hacking

Lock Debugging (spinlocks, mutexes, etc…)

Lock debugging: prove locking correctness

Kconfig

Practice

```
[    0.030236] =================================
[    0.030732] [ INFO: inconsistent lock state ]
[    0.031000] 4.9.0+ #15 Not tainted
[    0.031000] ---------------------------------
[    0.031000] inconsistent {IN-HARDIRQ-W} -> {HARDIRQ-ON-W} usage.
[    0.031000] swapper/0/1 [HC0[0]:SC0[0]:HE1:SE1] takes:
[    0.031000]  (&rq->lock){?.....}, at: [<ffffffff8108cc0e>] wake_up_process+0x2e/0x60
{IN-HARDIRQ-W} state was registered at:
[    0.031000]   [    0.031000] [<ffffffff810ace95>] __lock_acquire+0xaf5/0x1220
[    0.031000]   [    0.031000] [<ffffffff810ad9c0>] lock_acquire+0xb0/0x1d0
[    0.031000]   [    0.031000] [<ffffffff819e2e9c>] _raw_spin_lock+0x2c/0x40
[    0.031000]   [    0.031000] [<ffffffff8108e26c>] scheduler_tick+0x3c/0xc0
[    0.031000]   [    0.031000] [<ffffffff810d2e52>] update_process_times+0x42/0x50
[    0.031000]   [    0.031000] [<ffffffff810e158a>] tick_periodic+0x2a/0xc0
[    0.031000]   [    0.031000] [<ffffffff810e1640>] tick_handle_periodic+0x20/0x70
[    0.031000]   [    0.031000] [<ffffffff8101e930>] timer_interrupt+0x10/0x20
[    0.031000]   [    0.031000] [<ffffffff810bff47>] __handle_irq_event_perq+0x37/0x300
[    0.031000]   [    0.031000] [<ffffffff810c022e>] handle_irq_event_perq+0x1e/0x50
[    0.031000]   [    0.031000] [<ffffffff810c0294>] handle_irq_event+0x34/0x60
[    0.031000]   [    0.031000] [<ffffffff810c3353>] handle_level_irq+0x83/0xf0
[    0.031000]   [    0.031000] [<ffffffff8101e376>] handle_irq+0xa6/0x130
[    0.031000]   [    0.031000] [<ffffffff8101da2e>] do_IRQ+0x5e/0x120
[    0.031000]   [    0.031000] [<ffffffff819e40c9>] ret_from_intr+0x0/0x19
[    0.031000]   [    0.031000] [<ffffffff810c1f30>] __setup_irq+0x440/0x5e0
[    0.031000]   [    0.031000] [<ffffffff810c211f>] setup_irq+0x4f/0xc0
[    0.031000]   [    0.031000] [<ffffffff81f8b442>] setup_default_timer_irq+0x1e/0x20
[    0.031000]   [    0.031000] [<ffffffff81f8b45b>] hpet_time_init+0x17/0x19
[    0.031000]   [    0.031000] [<ffffffff81f8b41d>] x86_late_time_init+0xa/0x11
[    0.031000]   [    0.031000] [<ffffffff81f84f7f>] start_kernel+0x389/0x438
[    0.031000]   [    0.031000] [<ffffffff81f8458c>] x86_64_start_reservations+0x2a/0x2c
[    0.031000]   [    0.031000] [<ffffffff81f84678>] x86_64_start_kernel+0xea/0xed
[    0.031000] irq event stamp: 837
[    0.031000] hardirqs last  enabled at (837): [<ffffffff810b03dc>] __raw_spin_lock_init+0x1c/0x50
[    0.031000] hardirqs last disabled at (836): [<ffffffff810b03dc>] __raw_spin_lock_init+0x1c/0x50
[    0.031000] softirqs last  enabled at (588): [<ffffffff81064cde>] __do_softirq+0x32e/0x430
[    0.031000] softirqs last disabled at (583): [<ffffffff81065065>] irq_exit+0xb5/0xc0
[    0.031000]
[    0.031000] other info that might help us debug this:
[    0.031000]  Possible unsafe locking scenario:
[    0.031000]
[    0.031000]       cpu0
[    0.031000]       ----
[    0.031000]   lock(&rq->lock);
[    0.031000]   <Interrupt>
[    0.031000]     lock(&rq->lock);
[    0.031000]
[    0.031000]  *** DEADLOCK ***
[    0.031000]
[    0.031000] 2 locks held by swapper/0/1:
[    0.031000]  #0:  (cpu_hotplug.dep_map){.+.+.+}, at: [<ffffffff8105dd0f>] get_online_qs+0x1f/0x70
[    0.031000]  #1:  (smpboot_threads_lock){+.+.+.}, at: [<ffffffff810858e3>] smpboot_register...qmask+0x33/0x100
```

```
[    0.031000]
[    0.031000] stack backtrace:
[    0.031000] cpu: 0 PID: 1 Comm: swapper/0 Not tainted 4.9.0+ #15
[    0.031000] Hardware name: QEMU Standard PC (i440FX + PIIX, 1996), BIOS Bochs 01/01/2011
[    0.031000]  ffffc900000d3a00 ffffffff8138ab4e ffff88001e370000 ffffffff827a8aa0
[    0.031000]  ffffc900000d3a50 ffffffff81150aa2 0000000000000000 0000000000000001
[    0.031000]  0000000000000001 0000000000000000 ffff88001e370000 ffffffff810aa630
[    0.031000] Call Trace:
[    0.031000]  [<ffffffff8138ab4e>] dump_stack+0x67/0x99
[    0.031000]  [<ffffffff81150aa2>] print_usage_bug+0x1f2/0x203
[    0.031000]  [<ffffffff810aa630>] ? print_shortest_lock_dependencies+0x1c0/0x1c0
[    0.031000]  [<ffffffff810abdf2>] mark_lock+0x212/0x2a0
[    0.031000]  [<ffffffff810acd63>] __lock_acquire+0x9c3/0x1220
[    0.031000]  [<ffffffff810ad9c0>] lock_acquire+0xb0/0x1d0
[    0.031000]  [<ffffffff8108cc0e>] ? wake_up_process+0x2e/0x60
[    0.031000]  [<ffffffff81085570>] ? sort_range+0x20/0x20
[    0.031000]  [<ffffffff819e2e9c>] _raw_spin_lock+0x2c/0x40
[    0.031000]  [<ffffffff8108cc0e>] ? wake_up_process+0x2e/0x60
[    0.031000]  [<ffffffff8108cc0e>] wake_up_process+0x2e/0x60
[    0.031000]  [<ffffffff81081a88>] __kthread_create_on_node+0x128/0x220
[    0.031000]  [<ffffffff810aa6a0>] ? check_usage_backwards+0x70/0x120
[    0.031000]  [<ffffffff810aa6a0>] ? check_usage_backwards+0x70/0x120
[    0.031000]  [<ffffffff81081bb4>] kthread_create_on_node+0x34/0x40
[    0.031000]  [<ffffffff811ac750>] ? kmem_cache_alloc_node_trace+0x1c0/0x230
[    0.031000]  [<ffffffff81082449>] kthread_create_on_q+0x29/0x70
[    0.031000]  [<ffffffff81085825>] __smpboot_create_thread.part.3+0x65/0xf0
[    0.031000]  [<ffffffff81085950>] smpboot_register_perq_thread_qmask+0xa0/0x100
[    0.031000]  [<ffffffff81f9e1ac>] ? trace_event_define_fields_irq_handler_exit+0x6a/0x6a
[    0.031000]  [<ffffffff81f9e1e2>] spawn_ksoftirqd+0x36/0x40
[    0.031000]  [<ffffffff81000408>] do_one_initcall+0x38/0x150
[    0.031000]  [<ffffffff8102e0e3>] ? print_q_info+0x83/0xf0
[    0.031000]  [<ffffffff81f8516c>] kernel_init_freeable+0x13e/0x27e
[    0.031000]  [<ffffffff819dbb19>] ? kernel_init+0x9/0x100
[    0.031000]  [<ffffffff819dbb10>] ? rest_init+0x130/0x130
[    0.031000]  [<ffffffff819dbb19>] kernel_init+0x9/0x100
[    0.031000]  [<ffffffff819e3987>] ret_from_fork+0x27/0x40
```

Report Example 1

Byungchul Park max.byungchul.park@gmail.com

```
[    0.030236] =================================
[    0.030732] [ INFO: inconsistent lock state ]
[    0.031000] 4.9.0+ #15 Not tainted
[    0.031000] ---------------------------------
[    0.031000] inconsistent {IN-HARDIRQ-W} -> {HARDIRQ-ON-W} usage.
[    0.031000] swapper/0/1 [HC0[0]:SC0[0]:HE1:SE1] takes:
[    0.031000]  (&rq->lock){?.....}, at: [<ffffffff8108cc0e>] wake_up_process+0x2e/0x60
{IN-HARDIRQ-W} state was registered at:
[    0.031000]   [    0.031000] [<ffffffff810ace95>] __lock_acquire+0xaf5/0x1220
[    0.031000]   [    0.031000] [<ffffffff810ad9c0>] lock_acquire+0xb0/0x1d0
[    0.031000]   [    0.031000] [<ffffffff819e2e9c>] _raw_spin_lock+0x2c/0x40
[    0.031000]   [    0.031000] [<ffffffff8108e26c>] scheduler_tick+0x3c/0xc0
[    0.031000]   [    0.031000] [<ffffffff810d2e52>] update_process_times+0x42/0x50
[    0.031000]   [    0.031000] [<ffffffff810e158a>] tick_periodic+0x2a/0xc0
[    0.031000]   [    0.031000] [<ffffffff810e1640>] tick_handle_periodic+0x20/0x70
[    0.031000]   [    0.031000] [<ffffffff8101e930>] timer_interrupt+0x10/0x20
[    0.031000]   [    0.031000] [<ffffffff810bff47>] __handle_irq_event_perq+0x37/0x300
[    0.031000]   [    0.031000] [<ffffffff810c022e>] handle_irq_event_perq+0x1e/0x50
[    0.031000]   [    0.031000] [<ffffffff810c0294>] handle_irq_event+0x34/0x60
[    0.031000]   [    0.031000] [<ffffffff810c3353>] handle_level_irq+0x83/0xf0
[    0.031000]   [    0.031000] [<ffffffff8101e376>] handle_irq+0xa6/0x130
[    0.031000]   [    0.031000] [<ffffffff8101da2e>] do_IRQ+0x5e/0x120
[    0.031000]   [    0.031000] [<ffffffff819e40c9>] ret_from_intr+0x0/0x19
[    0.031000]   [    0.031000] [<ffffffff810c1f30>] __setup_irq+0x440/0x5e0
[    0.031000]   [    0.031000] [<ffffffff810c211f>] setup_irq+0x4f/0xc0
[    0.031000]   [    0.031000] [<ffffffff81f8b442>] setup_default_timer_irq+0x1e/0x20
[    0.031000]   [    0.031000] [<ffffffff81f8b45b>] hpet_time_init+0x17/0x19
[    0.031000]   [    0.031000] [<ffffffff81f8b41d>] x86_late_time_init+0xa/0x11
[    0.031000]   [    0.031000] [<ffffffff81f84f7f>] start_kernel+0x389/0x438
[    0.031000]   [    0.031000] [<ffffffff81f8458c>] x86_64_start_reservations+0x2a/0x2c
[    0.031000]   [    0.031000] [<ffffffff81f84678>] x86_64_start_kernel+0xea/0xed
[    0.031000] irq event stamp: 837
[    0.031000] hardirqs last  enabled at (837): [<ffffffff810b03dc>] __raw_spin_lock_init+0x1c/0x50
[    0.031000] hardirqs last disabled at (836): [<ffffffff810b03dc>] __raw_spin_lock_init+0x1c/0x50
[    0.031000] softirqs last  enabled at (588): [<ffffffff81064cde>] __do_softirq+0x32e/0x430
[    0.031000] softirqs last disabled at (583): [<ffffffff81065065>] irq_exit+0xb5/0xc0
[    0.031000]
[    0.031000] other info that might help us debug this:
[    0.031000]  Possible unsafe locking scenario:
[    0.031000]
[    0.031000]       cpu0
[    0.031000]       ----
[    0.031000]   lock(&rq->lock);
[    0.031000]   <Interrupt>
[    0.031000]     lock(&rq->lock);
[    0.031000]
[    0.031000]  *** DEADLOCK ***
[    0.031000]
[    0.031000] 2 locks held by swapper/0/1:
[    0.031000]  #0:  (cpu_hotplug.dep_map){.+.+.+}, at: [<ffffffff8105dd0f>] get_online_qs+0x1f/0x70
[    0.031000]  #1:  (smpboot_threads_lock){+.+.+.}, at: [<ffffffff810858e3>] smpboot_register...qmask+0x33/0x100
```

```
[    0.031000]
[    0.031000] stack backtrace:
[    0.031000] cpu: 0 PID: 1 Comm: swapper/0 Not tainted 4.9.0+ #15
[    0.031000] Hardware name: QEMU Standard PC (i440FX + PIIX, 1996), BIOS Bochs 01/01/2011
[    0.031000]  ffffc900000d3a00 ffffffff8138ab4e ffff88001e370000 ffffffff827a8aa0
[    0.031000]  ffffc900000d3a50 ffffffff81150aa2 0000000000000000 0000000000000001
[    0.031000]  0000000000000001 0000000000000000 ffff88001e370000 ffffffff810aa630
[    0.031000] Call Trace:
[    0.031000]  [<ffffffff8138ab4e>] dump_stack+0x67/0x99
[    0.031000]  [<ffffffff81150aa2>] print_usage_bug+0x1f2/0x203
[    0.031000]  [<ffffffff810aa630>] ? print_shortest_lock_dependencies+0x1c0/0x1c0
[    0.031000]  [<ffffffff810abdf2>] mark_lock+0x212/0x2a0
[    0.031000]  [<ffffffff810acd63>] __lock_acquire+0x9c3/0x1220
[    0.031000]  [<ffffffff810ad9c0>] lock_acquire+0xb0/0x1d0
[    0.031000]  [<ffffffff8108cc0e>] ? wake_up_process+0x2e/0x60
[    0.031000]  [<ffffffff81085570>] ? sort_range+0x20/0x20
[    0.031000]  [<ffffffff819e2e9c>] _raw_spin_lock+0x2c/0x40
[    0.031000]  [<ffffffff8108cc0e>] ? wake_up_process+0x2e/0x60
[    0.031000]  [<ffffffff8108cc0e>] wake_up_process+0x2e/0x60
[    0.031000]  [<ffffffff81081a88>] __kthread_create_on_node+0x128/0x220
[    0.031000]  [<ffffffff810aa6a0>] ? check_usage_backwards+0x70/0x120
[    0.031000]  [<ffffffff810aa6a0>] ? check_usage_backwards+0x70/0x120
[    0.031000]  [<ffffffff81081bb4>] kthread_create_on_node+0x34/0x40
[    0.031000]  [<ffffffff811ac750>] ? kmem_cache_alloc_node_trace+0x1c0/0x230
[    0.031000]  [<ffffffff81082449>] kthread_create_on_q+0x29/0x70
[    0.031000]  [<ffffffff81085825>] __smpboot_create_thread.part.3+0x65/0xf0
[    0.031000]  [<ffffffff81085950>] smpboot_register_perq_thread_qmask+0xa0/0x100
[    0.031000]  [<ffffffff81f9e1ac>] ? trace_event_define_fields_irq_handler_exit+0x6a/0x6a
[    0.031000]  [<ffffffff81f9e1e2>] spawn_ksoftirqd+0x36/0x40
[    0.031000]  [<ffffffff81000408>] do_one_initcall+0x38/0x150
[    0.031000]  [<ffffffff8102e0e3>] ? print_q_info+0x83/0xf0
[    0.031000]  [<ffffffff81f8516c>] kernel_init_freeable+0x13e/0x27e
[    0.031000]  [<ffffffff819dbb19>] ? kernel_init+0x9/0x100
[    0.031000]  [<ffffffff819dbb10>] ? rest_init+0x130/0x130
[    0.031000]  [<ffffffff819dbb19>] kernel_init+0x9/0x100
[    0.031000]  [<ffffffff819e3987>] ret_from_fork+0x27/0x40
```

Report Example 1

Byungchul Park max.byungchul.park@gmail.com

swapper/0/1 [HC0[0]:SC0[0]:HE1:SE1] takes:

(&rq->lock){?.....}, at: [<ffffffff8108cc0e>] wake_up_process+0x2e/0x60

{IN-HARDIRQ-W} state was registered at:

[<ffffffff8108e26c>] scheduler_tick+0x3c/0xc0 (CALL STACK)

cpu0

---

lock(&rq->lock);

<Interrupt>

lock(&rq->lock);

Report Example 1

Byungchul Park max.byungchul.park@gmail.com

swapper/0/1 [HC0[0]:SC0[0]:HE1:SE1] takes:

(&rq->lock){?.....}, at: [<ffffffff8108cc0e>] wake_up_process+0x2e/0x60

{IN-HARDIRQ-W} state was registered at:

[<ffffffff8108e26c>] scheduler_tick+0x3c/0xc0 (CALL STACK)

cpu0

---

lock(&rq->lock);

<Interrupt>

lock(&rq->lock);

Report Example 1

Byungchul Park max.byungchul.park@gmail.com

```
[    0.030236] =================================
[    0.030732] [ INFO: inconsistent lock state ]
[    0.031000] 4.9.0+ #15 Not tainted
[    0.031000] ---------------------------------
[    0.031000] inconsistent {IN-HARDIRQ-W} -> {HARDIRQ-ON-W} usage.
[    0.031000] swapper/0/1 [HC0[0]:SC0[0]:HE1:SE1] takes:
[    0.031000]  (&rq->lock){?.....}, at: [<ffffffff8108cc0e>] wake_up_process+0x2e/0x60
{IN-HARDIRQ-W} state was registered at:
[    0.031000]   [    0.031000] [<ffffffff810ace95>] __lock_acquire+0xaf5/0x1220
[    0.031000]   [    0.031000] [<ffffffff810ad9c0>] lock_acquire+0xb0/0x1d0
[    0.031000]   [    0.031000] [<ffffffff819e2e9c>] _raw_spin_lock+0x2c/0x40
[    0.031000]   [    0.031000] [<ffffffff8108e26c>] scheduler_tick+0x3c/0xc0
[    0.031000]   [    0.031000] [<ffffffff810d2e52>] update_process_times+0x42/0x50
[    0.031000]   [    0.031000] [<ffffffff810e158a>] tick_periodic+0x2a/0xc0
[    0.031000]   [    0.031000] [<ffffffff810e1640>] tick_handle_periodic+0x20/0x70
[    0.031000]   [    0.031000] [<ffffffff8101e930>] timer_interrupt+0x10/0x20
[    0.031000]   [    0.031000] [<ffffffff810bff47>] __handle_irq_event_perq+0x37/0x300
[    0.031000]   [    0.031000] [<ffffffff810c022e>] handle_irq_event_perq+0x1e/0x50
[    0.031000]   [    0.031000] [<ffffffff810c0294>] handle_irq_event+0x34/0x60
[    0.031000]   [    0.031000] [<ffffffff810c3353>] handle_level_irq+0x83/0xf0
[    0.031000]   [    0.031000] [<ffffffff8101e376>] handle_irq+0xa6/0x130
[    0.031000]   [    0.031000] [<ffffffff8101da2e>] do_IRQ+0x5e/0x120
[    0.031000]   [    0.031000] [<ffffffff819e40c9>] ret_from_intr+0x0/0x19
[    0.031000]   [    0.031000] [<ffffffff810c1f30>] __setup_irq+0x440/0x5e0
[    0.031000]   [    0.031000] [<ffffffff810c211f>] setup_irq+0x4f/0xc0
[    0.031000]   [    0.031000] [<ffffffff81f8b442>] setup_default_timer_irq+0x1e/0x20
[    0.031000]   [    0.031000] [<ffffffff81f8b45b>] hpet_time_init+0x17/0x19
[    0.031000]   [    0.031000] [<ffffffff81f8b41d>] x86_late_time_init+0xa/0x11
[    0.031000]   [    0.031000] [<ffffffff81f84f7f>] start_kernel+0x389/0x438
[    0.031000]   [    0.031000] [<ffffffff81f8458c>] x86_64_start_reservations+0x2a/0x2c
[    0.031000]   [    0.031000] [<ffffffff81f84678>] x86_64_start_kernel+0xea/0xed
[    0.031000] irq event stamp: 837
[    0.031000] hardirqs last  enabled at (837): [<ffffffff810b03dc>] __raw_spin_lock_init+0x1c/0x50
[    0.031000] hardirqs last disabled at (836): [<ffffffff810b03dc>] __raw_spin_lock_init+0x1c/0x50
[    0.031000] softirqs last  enabled at (588): [<ffffffff81064cde>] __do_softirq+0x32e/0x430
[    0.031000] softirqs last disabled at (583): [<ffffffff81065065>] irq_exit+0xb5/0xc0
[    0.031000]
[    0.031000] other info that might help us debug this:
[    0.031000]  Possible unsafe locking scenario:
[    0.031000]
[    0.031000]       cpu0
[    0.031000]       ----
[    0.031000]   lock(&rq->lock);
[    0.031000]   <Interrupt>
[    0.031000]     lock(&rq->lock);
[    0.031000]
[    0.031000]  *** DEADLOCK ***
[    0.031000]
[    0.031000] 2 locks held by swapper/0/1:
[    0.031000]  #0:  (cpu_hotplug.dep_map){.+.+.+}, at: [<ffffffff8105dd0f>] get_online_qs+0x1f/0x70
[    0.031000]  #1:  (smpboot_threads_lock){+.+.+.}, at: [<ffffffff810858e3>] smpboot_register...qmask+0x33/0x100
```

```
[    0.031000]
[    0.031000] stack backtrace:
[    0.031000] cpu: 0 PID: 1 Comm: swapper/0 Not tainted 4.9.0+ #15
[    0.031000] Hardware name: QEMU Standard PC (i440FX + PIIX, 1996), BIOS Bochs 01/01/2011
[    0.031000]  ffffc900000d3a00 ffffffff8138ab4e ffff88001e370000 ffffffff827a8aa0
[    0.031000]  ffffc900000d3a50 ffffffff81150aa2 0000000000000000 0000000000000001
[    0.031000]  0000000000000001 0000000000000000 ffff88001e370000 ffffffff810aa630
[    0.031000] Call Trace:
[    0.031000]  [<ffffffff8138ab4e>] dump_stack+0x67/0x99
[    0.031000]  [<ffffffff81150aa2>] print_usage_bug+0x1f2/0x203
[    0.031000]  [<ffffffff810aa630>] ? print_shortest_lock_dependencies+0x1c0/0x1c0
[    0.031000]  [<ffffffff810abdf2>] mark_lock+0x212/0x2a0
[    0.031000]  [<ffffffff810acd63>] __lock_acquire+0x9c3/0x1220
[    0.031000]  [<ffffffff810ad9c0>] lock_acquire+0xb0/0x1d0
[    0.031000]  [<ffffffff8108cc0e>] ? wake_up_process+0x2e/0x60
[    0.031000]  [<ffffffff81085570>] ? sort_range+0x20/0x20
[    0.031000]  [<ffffffff819e2e9c>] _raw_spin_lock+0x2c/0x40
[    0.031000]  [<ffffffff8108cc0e>] ? wake_up_process+0x2e/0x60
[    0.031000]  [<ffffffff8108cc0e>] wake_up_process+0x2e/0x60
[    0.031000]  [<ffffffff81081a88>] __kthread_create_on_node+0x128/0x220
[    0.031000]  [<ffffffff810aa6a0>] ? check_usage_backwards+0x70/0x120
[    0.031000]  [<ffffffff810aa6a0>] ? check_usage_backwards+0x70/0x120
[    0.031000]  [<ffffffff81081bb4>] kthread_create_on_node+0x34/0x40
[    0.031000]  [<ffffffff811ac750>] ? kmem_cache_alloc_node_trace+0x1c0/0x230
[    0.031000]  [<ffffffff81082449>] kthread_create_on_q+0x29/0x70
[    0.031000]  [<ffffffff81085825>] __smpboot_create_thread.part.3+0x65/0xf0
[    0.031000]  [<ffffffff81085950>] smpboot_register_perq_thread_qmask+0xa0/0x100
[    0.031000]  [<ffffffff81f9e1ac>] ? trace_event_define_fields_irq_handler_exit+0x6a/0x6a
[    0.031000]  [<ffffffff81f9e1e2>] spawn_ksoftirqd+0x36/0x40
[    0.031000]  [<ffffffff81000408>] do_one_initcall+0x38/0x150
[    0.031000]  [<ffffffff8102e0e3>] ? print_q_info+0x83/0xf0
[    0.031000]  [<ffffffff81f8516c>] kernel_init_freeable+0x13e/0x27e
[    0.031000]  [<ffffffff819dbb19>] ? kernel_init+0x9/0x100
[    0.031000]  [<ffffffff819dbb10>] ? rest_init+0x130/0x130
[    0.031000]  [<ffffffff819dbb19>] kernel_init+0x9/0x100
[    0.031000]  [<ffffffff819e3987>] ret_from_fork+0x27/0x40
```

Report Example 1

Byungchul Park max.byungchul.park@gmail.com

```
[    0.030236] =================================
[    0.030732] [ INFO: inconsistent lock state ]
[    0.031000] 4.9.0+ #15 Not tainted
[    0.031000] ---------------------------------
[    0.031000] inconsistent {IN-HARDIRQ-W} -> {HARDIRQ-ON-W} usage.
[    0.031000] swapper/0/1 [HC0[0]:SC0[0]:HE1:SE1] takes:
[    0.031000]  (&rq->lock){?.....}, at: [<ffffffff8108cc0e>] wake_up_process+0x2e/0x60
{IN-HARDIRQ-W} state was registered at:
[    0.031000]   [    0.031000] [<ffffffff810ace95>] __lock_acquire+0xaf5/0x1220
[    0.031000]   [    0.031000] [<ffffffff810ad9c0>] lock_acquire+0xb0/0x1d0
[    0.031000]   [    0.031000] [<ffffffff819e2e9c>] _raw_spin_lock+0x2c/0x40
[    0.031000]   [    0.031000] [<ffffffff8108e26c>] scheduler_tick+0x3c/0xc0
[    0.031000]   [    0.031000] [<ffffffff810d2e52>] update_process_times+0x42/0x50
[    0.031000]   [    0.031000] [<ffffffff810e158a>] tick_periodic+0x2a/0xc0
[    0.031000]   [    0.031000] [<ffffffff810e1640>] tick_handle_periodic+0x20/0x70
[    0.031000]   [    0.031000] [<ffffffff8101e930>] timer_interrupt+0x10/0x20
[    0.031000]   [    0.031000] [<ffffffff810bff47>] __handle_irq_event_perq+0x37/0x300
[    0.031000]   [    0.031000] [<ffffffff810c022e>] handle_irq_event_perq+0x1e/0x50
[    0.031000]   [    0.031000] [<ffffffff810c0294>] handle_irq_event+0x34/0x60
[    0.031000]   [    0.031000] [<ffffffff810c3353>] handle_level_irq+0x83/0xf0
[    0.031000]   [    0.031000] [<ffffffff8101e376>] handle_irq+0xa6/0x130
[    0.031000]   [    0.031000] [<ffffffff8101da2e>] do_IRQ+0x5e/0x120
[    0.031000]   [    0.031000] [<ffffffff819e40c9>] ret_from_intr+0x0/0x19
[    0.031000]   [    0.031000] [<ffffffff810c1f30>] __setup_irq+0x440/0x5e0
```

```
[    0.031000]   [    0.031000] [<ffffffff81f8b41d>] x86_late_time_init+0xa/0x11
[    0.031000]   [    0.031000] [<ffffffff81f84f7f>] start_kernel+0x389/0x438
[    0.031000]   [    0.031000] [<ffffffff81f8458c>] x86_64_start_reservations+0x2a/0x2c
[    0.031000]   [    0.031000] [<ffffffff81f84678>] x86_64_start_kernel+0xea/0xed
[    0.031000] irq event stamp: 837
[    0.031000] hardirqs last  enabled at (837): [<ffffffff810b03dc>] __raw_spin_lock_init…
[    0.031000] hardirqs last disabled at (836): [<ffffffff810b03dc>] __raw_spin_lock_init…
[    0.031000] softirqs last  enabled at (588): [<ffffffff81064cde>] __do_softirq+0x32e/0x430
[    0.031000] softirqs last disabled at (583): [<ffffffff81065065>] irq_exit+0xb5/0xc0
[    0.031000]
[    0.031000] other info that might help us debug this:
[    0.031000]  Possible unsafe locking scenario:
[    0.031000]
[    0.031000]        cpu0
[    0.031000]        ----
[    0.031000]   lock(&rq->lock);
[    0.031000]   <Interrupt>
[    0.031000]     lock(&rq->lock);
[    0.031000]
[    0.031000]  *** DEADLOCK ***
[    0.031000]
[    0.031000] 2 locks held by swapper/0/1:
[    0.031000]  #0:  (cpu_hotplug.dep_map){.+.+.+}, at: [<ffffffff8105dd0f>] get_online_qs…
[    0.031000]  #1:  (smpboot_threads_lock){+.+.+.}, at: [<ffffffff810858e3>] smpboot_register...
```

```
[    0.031000] stack backtrace:
[    0.031000] cpu: 0 PID: 1 Comm: swapper/0 Not tainted 4.9.0+ #15
[    0.031000] Hardware name: QEMU Standard PC (i440FX + PIIX, 1996), BIOS…
[    0.031000]  ffffc900000d3a00 ffffffff8138ab4e ffff88001e370000 ffffffff827a8aa0
[    0.031000]  ffffc900000d3a50 ffffffff81150aa2 0000000000000000 0000000000000001
[    0.031000]  0000000000000001 0000000000000000 ffff88001e370000 ffffffff810aa630
[    0.031000] Call Trace:
[    0.031000]  [<ffffffff8138ab4e>] dump_stack+0x67/0x99
[    0.031000]  [<ffffffff81150aa2>] print_usage_bug+0x1f2/0x203
[    0.031000]  [<ffffffff810aa630>] ? print_shortest_lock_dependencies+0x1c0/0x1c0
[    0.031000]  [<ffffffff810abdf2>] mark_lock+0x212/0x2a0
[    0.031000]  [<ffffffff810acd63>] __lock_acquire+0x9c3/0x1220
[    0.031000]  [<ffffffff810ad9c0>] lock_acquire+0xb0/0x1d0
[    0.031000]  [<ffffffff8108cc0e>] ? wake_up_process+0x2e/0x60
[    0.031000]  [<ffffffff81085570>] ? sort_range+0x20/0x20
[    0.031000]  [<ffffffff819e2e9c>] _raw_spin_lock+0x2c/0x40
[    0.031000]  [<ffffffff8108cc0e>] ? wake_up_process+0x2e/0x60
[    0.031000]  [<ffffffff8108cc0e>] wake_up_process+0x2e/0x60
[    0.031000]  [<ffffffff81081a88>] __kthread_create_on_node+0x128/0x220
[    0.031000]  [<ffffffff810aa6a0>] ? check_usage_backwards+0x70/0x120
[    0.031000]  [<ffffffff810aa6a0>] ? check_usage_backwards+0x70/0x120
[    0.031000]  [<ffffffff81081bb4>] kthread_create_on_node+0x34/0x40
[    0.031000]  [<ffffffff811ac750>] ? kmem_cache_alloc_node_trace+0x1c0/0x230
[    0.031000]  [<ffffffff81082449>] kthread_create_on_q+0x29/0x70
```

```
[    0.031237] =======================================================
[    0.031946] [ INFO: possible circular locking dependency detected ]
[    0.032000] 4.9.0+ #16 Not tainted
[    0.032000] -------------------------------------------------------
[    0.032000] swapper/0/1 is trying to acquire lock:
[    0.032000]  (&p->pi_lock){.....}, at: [<ffffffff8108cc1b>] wake_up_process+0x3b/0x70
[    0.032000] but task is already holding lock:
[    0.032000]  (&rq->lock){-.....}, at: [<ffffffff8108cc10>] wake_up_process+0x30/0x70
[    0.032000] which lock already depends on the new lock.
[    0.032000]
[    0.032000]
[    0.032000] the existing dependency chain (in reverse order) is:
[    0.032000]
-> #1 (&rq->lock){-.....}:
[    0.032000]        [    0.032000] [<ffffffff810ad471>] __lock_acquire+0x10c1/0x1220
[    0.032000]        [    0.032000] [<ffffffff810ad9d0>] lock_acquire+0xb0/0x1d0
[    0.032000]        [    0.032000] [<ffffffff819e2e9c>] _raw_spin_lock+0x2c/0x40
[    0.032000]        [    0.032000] [<ffffffff8109ba57>] task_fork_fair+0x27/0xf0
[    0.032000]        [    0.032000] [<ffffffff8108d9fd>] sched_fork+0x24d/0x3d0
[    0.032000]        [    0.032000] [<ffffffff8105ad02>] copy_process.part.47+0x622/0x1e10
[    0.032000]        [    0.032000] [<ffffffff8105c6d2>] _do_fork+0xe2/0x6e0
[    0.032000]        [    0.032000] [<ffffffff8105ccf4>] kernel_thread+0x24/0x30
[    0.032000]        [    0.032000] [<ffffffff819db9fe>] rest_init+0x1e/0x130
[    0.032000]        [    0.032000] [<ffffffff81f85021>] start_kernel+0x42b/0x438
[    0.032000]        [    0.032000] [<ffffffff81f8458c>] x86_64_start_reservations+0x2a/0x2c
[    0.032000]        [    0.032000] [<ffffffff81f84678>] x86_64_start_kernel+0xea/0xed
[    0.032000]
-> #0 (&p->pi_lock){.....}:
[    0.032000]        [    0.032000] [<ffffffff810ab4f3>] check_prev_add+0x723/0x730
[    0.032000]        [    0.032000] [<ffffffff810ad471>] __lock_acquire+0x10c1/0x1220
[    0.032000]        [    0.032000] [<ffffffff810ad9d0>] lock_acquire+0xb0/0x1d0
[    0.032000]        [    0.032000] [<ffffffff819e3017>] _raw_spin_lock_irqsave+0x37/0x50
[    0.032000]        [    0.032000] [<ffffffff8108cc1b>] wake_up_process+0x3b/0x70
[    0.032000]        [    0.032000] [<ffffffff81081a88>] __kthread_create_on_node+0x128/0x220
[    0.032000]        [    0.032000] [<ffffffff81081bb4>] kthread_create_on_node+0x34/0x40
[    0.032000]        [    0.032000] [<ffffffff81082449>] kthread_create_on_q+0x29/0x70
[    0.032000]        [    0.032000] [<ffffffff81085825>] __smpboot_create_thread.part.3+0x65/0xf0
[    0.032000]        [    0.032000] [<ffffffff81085950>] smpboot_register_perq_thread_qmask+0xa0/0x100
[    0.032000]        [    0.032000] [<ffffffff81f9e1e2>] spawn_ksoftirqd+0x36/0x40
[    0.032000]        [    0.032000] [<ffffffff81000408>] do_one_initcall+0x38/0x150
[    0.032000]        [    0.032000] [<ffffffff81f8516c>] kernel_init_freeable+0x13e/0x27e
[    0.032000]        [    0.032000] [<ffffffff819dbb19>] kernel_init+0x9/0x100
[    0.032000]        [    0.032000] [<ffffffff819e3987>] ret_from_fork+0x27/0x40
[    0.032000]
[    0.032000] other info that might help us debug this:
[    0.032000]
[    0.032000]  Possible unsafe locking scenario:
[    0.032000]
[    0.032000]        cpu0                    cpu1
[    0.032000]        ----                    ----
[    0.032000]   lock(&rq->lock);
[    0.032000]                                lock(&p->pi_lock);
[    0.032000]                                lock(&rq->lock);
[    0.032000]   lock(&p->pi_lock);
[    0.032000]
[    0.032000]  *** DEADLOCK ***
[    0.032000]
[    0.032000] 3 locks held by swapper/0/1:
[    0.032000]  #0:  (cpu_hotplug.dep_map){.+.+.+.+}, at: [<ffffffff8105dd0f>] get_online_qs+0x1f/0x70
[    0.032000]  #1:  (smpboot_threads_lock){+.+.+.+}, at: [<ffffffff810858e3>] smpboot_register…qmask+0x33/0x100
[    0.032000]  #2:  (&rq->lock){-.....}, at: [<ffffffff8108cc10>] wake_up_process+0x30/0x70
[    0.032000]
[    0.032000] stack backtrace:
[    0.032000] q: 0 PID: 1 Comm: swapper/0 Not tainted 4.9.0+ #16
[    0.032000] Hardware name: QEMU Standard PC (i440FX + PIIX, 1996), BIOS Bochs 01/01/2011
[    0.032000]  ffffc900000d39b8 ffffffff8138ab5e ffff88001e370870 ffffffff827a8aa0
[    0.032000]  ffffc900000d39f8 ffffffff811506fa ffffc900000d3a20 ffff88001e370870
[    0.032000]  ffff88001e3708a0 ffff88001e370000 a579d3845e226952 ffffffff810a8dd0
[    0.032000] Call Trace:
[    0.032000]  [<ffffffff8138ab5e>] dump_stack+0x67/0x99
[    0.032000]  [<ffffffff811506fa>] print_circular_bug+0x2f0/0x2fe
[    0.032000]  [<ffffffff810a8dd0>] ? graph_unlock+0x80/0x80
[    0.032000]  [<ffffffff810ab4f3>] check_prev_add+0x723/0x730
[    0.032000]  [<ffffffff810ad471>] __lock_acquire+0x10c1/0x1220
[    0.032000]  [<ffffffff810a8dd0>] ? graph_unlock+0x80/0x80
[    0.032000]  [<ffffffff810ad9d0>] lock_acquire+0xb0/0x1d0
[    0.032000]  [<ffffffff8108cc1b>] ? wake_up_process+0x3b/0x70
[    0.032000]  [<ffffffff81085570>] ? sort_range+0x20/0x20
[    0.032000]  [<ffffffff819e3017>] _raw_spin_lock_irqsave+0x37/0x50
[    0.032000]  [<ffffffff8108cc1b>] ? wake_up_process+0x3b/0x70
[    0.032000]  [<ffffffff8108cc1b>] wake_up_process+0x3b/0x70
[    0.032000]  [<ffffffff81081a88>] __kthread_create_on_node+0x128/0x220
[    0.032000]  [<ffffffff810aa6b0>] ? check_usage_backwards+0x70/0x120
[    0.032000]  [<ffffffff810aa6b0>] ? check_usage_backwards+0x70/0x120
[    0.032000]  [<ffffffff81081bb4>] kthread_create_on_node+0x34/0x40
[    0.032000]  [<ffffffff811ac760>] ? kmem_cache_alloc_node_trace+0x1c0/0x230
[    0.032000]  [<ffffffff81082449>] kthread_create_on_q+0x29/0x70
[    0.032000]  [<ffffffff81085825>] __smpboot_create_thread.part.3+0x65/0xf0
[    0.032000]  [<ffffffff81085950>] smpboot_register_perq_thread_qmask+0xa0/0x100
[    0.032000]  [<ffffffff81f9e1ac>] ? trace_event_define_fields_irq_handler_exit+0x6a/0x6a
[    0.032000]  [<ffffffff81f9e1e2>] spawn_ksoftirqd+0x36/0x40
[    0.032000]  [<ffffffff81000408>] do_one_initcall+0x38/0x150
[    0.032000]  [<ffffffff8102e0e3>] ? print_q_info+0x83/0xf0
[    0.032000]  [<ffffffff81f8516c>] kernel_init_freeable+0x13e/0x27e
[    0.032000]  [<ffffffff819dbb19>] ? kernel_init+0x9/0x100
[    0.032000]  [<ffffffff819dbb10>] ? rest_init+0x130/0x130
[    0.032000]  [<ffffffff819dbb19>] kernel_init+0x9/0x100
[    0.032000]  [<ffffffff819e3987>] ret_from_fork+0x27/0x40
```

Report Example 2

Byungchul Park max.byungchul.park@gmail.com

```
[    0.031237] =====================================================
[    0.031946] [ INFO: possible circular locking dependency detected ]
[    0.032000] 4.9.0+ #16 Not tainted
[    0.032000] -------------------------------------------------------
[    0.032000] swapper/0/1 is trying to acquire lock:
[    0.032000]  (&p->pi_lock){.....}, at: [<ffffffff8108cc1b>] wake_up_process+0x3b/0x70
[    0.032000] but task is already holding lock:
[    0.032000]  (&rq->lock){-.....}, at: [<ffffffff8108cc10>] wake_up_process+0x30/0x70
[    0.032000] which lock already depends on the new lock.
[    0.032000]
[    0.032000]
[    0.032000] the existing dependency chain (in reverse order) is:
[    0.032000]
-> #1 (&rq->lock){-.....}:
[    0.032000]        [    0.032000] [<ffffffff810ad471>] __lock_acquire+0x10c1/0x1220
[    0.032000]        [    0.032000] [<ffffffff810ad9d0>] lock_acquire+0xb0/0x1d0
[    0.032000]        [    0.032000] [<ffffffff819e2e9e>] _raw_spin_lock+0x2e/0x40
[    0.032000]        [    0.032000] [<ffffffff8109ba57>] task_fork_fair+0x27/0xf0
[    0.032000]        [    0.032000] [<ffffffff8108d9fd>] sched_fork+0x24d/0x3d0
[    0.032000]        [    0.032000] [<ffffffff8105ad02>] copy_process.part.47+0x622/0x1e10
[    0.032000]        [    0.032000] [<ffffffff8105c6d2>] _do_fork+0xe2/0x6e0
[    0.032000]        [    0.032000] [<ffffffff8105ccf4>] kernel_thread+0x24/0x30
[    0.032000]        [    0.032000] [<ffffffff819db9fe>] rest_init+0x1e/0x130
[    0.032000]        [    0.032000] [<ffffffff81f85021>] start_kernel+0x42b/0x438
[    0.032000]        [    0.032000] [<ffffffff81f8458c>] x86_64_start_reservations+0x2a/0x2c
[    0.032000]        [    0.032000] [<ffffffff81f84678>] x86_64_start_kernel+0xea/0xed
[    0.032000]
-> #0 (&p->pi_lock){.....}:
[    0.032000]        [    0.032000] [<ffffffff810ab4f3>] check_prev_add+0x723/0x730
[    0.032000]        [    0.032000] [<ffffffff810ad471>] __lock_acquire+0x10c1/0x1220
[    0.032000]        [    0.032000] [<ffffffff810ad9d0>] lock_acquire+0xb0/0x1d0
[    0.032000]        [    0.032000] [<ffffffff819e3017>] _raw_spin_lock_irqsave+0x37/0x50
[    0.032000]        [    0.032000] [<ffffffff8108cc1b>] wake_up_process+0x3b/0x70
[    0.032000]        [    0.032000] [<ffffffff81081a88>] __kthread_create_on_node+0x128/0x220
[    0.032000]        [    0.032000] [<ffffffff81081bb4>] kthread_create_on_node+0x34/0x40
[    0.032000]        [    0.032000] [<ffffffff81082449>] kthread_create_on_q+0x29/0x70
[    0.032000]        [    0.032000] [<ffffffff81085825>] __smpboot_create_thread.part.3+0x65/0xf0
[    0.032000]        [    0.032000] [<ffffffff81085950>] smpboot_register_perq_thread_qmask+0xa0/0x100
[    0.032000]        [    0.032000] [<ffffffff81f9e1e2>] spawn_ksoftirqd+0x36/0x40
[    0.032000]        [    0.032000] [<ffffffff81000408>] do_one_initcall+0x38/0x150
[    0.032000]        [    0.032000] [<ffffffff81f8516c>] kernel_init_freeable+0x13e/0x27e
[    0.032000]        [    0.032000] [<ffffffff819dbb19>] kernel_init+0x9/0x100
[    0.032000]        [    0.032000] [<ffffffff819e3987>] ret_from_fork+0x27/0x40
[    0.032000]
[    0.032000] other info that might help us debug this:
[    0.032000]
[    0.032000]  Possible unsafe locking scenario:
[    0.032000]
[    0.032000]        cpu0                    cpu1
[    0.032000]        ----                    ----
[    0.032000]   lock(&rq->lock);
[    0.032000]                                lock(&p->pi_lock);
[    0.032000]                                lock(&rq->lock);
[    0.032000]   lock(&p->pi_lock);
[    0.032000]
[    0.032000]  *** DEADLOCK ***
[    0.032000]
[    0.032000] 3 locks held by swapper/0/1:
[    0.032000]  #0:  (cpu_hotplug.dep_map){.+.+.+.+}, at: [<ffffffff8105dd0f>] get_online_qs+0x1f/0x70
[    0.032000]  #1:  (smpboot_threads_lock){+.+.+.+}, at: [<ffffffff810858e3>] smpboot_register…qmask+0x33/0x100
[    0.032000]  #2:  (&rq->lock){-.....}, at: [<ffffffff8108cc10>] wake_up_process+0x30/0x70
[    0.032000]
[    0.032000] stack backtrace:
[    0.032000] q: 0 PID: 1 Comm: swapper/0 Not tainted 4.9.0+ #16
[    0.032000] Hardware name: QEMU Standard PC (i440FX + PIIX, 1996), BIOS Bochs 01/01/2011
[    0.032000]  ffffc900000d39b8 ffffffff8138ab5e ffff88001e370870 ffffffff827a8aa0
[    0.032000]  ffffc900000d39f8 ffffffff811506fa ffffc900000d3a20 ffff88001e370870
[    0.032000]  ffff88001e3708a0 ffff88001e370000 a579d3845e226952 ffffffff810a8dd0
[    0.032000] Call Trace:
[    0.032000]  [<ffffffff8138ab5e>] dump_stack+0x67/0x99
[    0.032000]  [<ffffffff811506fa>] print_circular_bug+0x2f0/0x2fe
[    0.032000]  [<ffffffff810a8dd0>] ? graph_unlock+0x80/0x80
[    0.032000]  [<ffffffff810ab4f3>] check_prev_add+0x723/0x730
[    0.032000]  [<ffffffff810ad471>] __lock_acquire+0x10c1/0x1220
[    0.032000]  [<ffffffff810a8dd0>] ? graph_unlock+0x80/0x80
[    0.032000]  [<ffffffff810ad9d0>] lock_acquire+0xb0/0x1d0
[    0.032000]  [<ffffffff8108cc1b>] ? wake_up_process+0x3b/0x70
[    0.032000]  [<ffffffff81085570>] ? sort_range+0x20/0x20
[    0.032000]  [<ffffffff819e3017>] _raw_spin_lock_irqsave+0x37/0x50
[    0.032000]  [<ffffffff8108cc1b>] ? wake_up_process+0x3b/0x70
[    0.032000]  [<ffffffff8108cc1b>] wake_up_process+0x3b/0x70
[    0.032000]  [<ffffffff81081a88>] __kthread_create_on_node+0x128/0x220
[    0.032000]  [<ffffffff810aa6b0>] ? check_usage_backwards+0x70/0x120
[    0.032000]  [<ffffffff810aa6b0>] ? check_usage_backwards+0x70/0x120
[    0.032000]  [<ffffffff81081bb4>] kthread_create_on_node+0x34/0x40
[    0.032000]  [<ffffffff811ac760>] ? kmem_cache_alloc_node_trace+0x1c0/0x230
[    0.032000]  [<ffffffff81082449>] kthread_create_on_q+0x29/0x70
[    0.032000]  [<ffffffff81085825>] __smpboot_create_thread.part.3+0x65/0xf0
[    0.032000]  [<ffffffff81085950>] smpboot_register_perq_thread_qmask+0xa0/0x100
[    0.032000]  [<ffffffff81f9e1ac>] ? trace_event_define_fields_irq_handler_exit+0x6a/0x6a
[    0.032000]  [<ffffffff81f9e1e2>] spawn_ksoftirqd+0x36/0x40
[    0.032000]  [<ffffffff81000408>] do_one_initcall+0x38/0x150
[    0.032000]  [<ffffffff8102e0e3>] ? print_q_info+0x83/0xf0
[    0.032000]  [<ffffffff81f8516c>] kernel_init_freeable+0x13e/0x27e
[    0.032000]  [<ffffffff819dbb19>] ? kernel_init+0x9/0x100
[    0.032000]  [<ffffffff819dbb10>] ? rest_init+0x130/0x130
[    0.032000]  [<ffffffff819dbb19>] kernel_init+0x9/0x100
[    0.032000]  [<ffffffff819e3987>] ret_from_fork+0x27/0x40
```

Report Example 2

Byungchul Park max.byungchul.park@gmail.com

swapper/0/1 is trying to acquire lock:

(&p->pi_lock){.....}, at: [<ffffffff8108cc1b>] wake_up_process+0x3b/0x70

but task is already holding lock:

(&rq->lock){-.....}, at: [<ffffffff8108cc10>] wake_up_process+0x30/0x70

which lock already depends on the new lock.

the existing dependency chain (in reverse order) is:

-> #1 (&rq->lock){-.....}:

[<ffffffff8109ba57>] task_fork_fair+0x27/0xf0 (CALL STACK)

-> #0 (&p->pi_lock){.....}:

[<ffffffff8108cc1b>] wake_up_process+0x3b/0x70 (CALL STACK)

cpu0	cpu1
----	----
lock(&rq->lock);	
	lock(&p->pi_lock);
	lock(&rq->lock);
lock(&p->pi_lock);	

Report Example 2

Byungchul Park max.byungchul.park@gmail.com

swapper/0/1 is trying to acquire lock:

(&p->pi_lock){.....}, at: [<ffffffff8108cc1b>] wake_up_process+0x3b/0x70

but task is already holding lock:

(&rq->lock){-.....}, at: [<ffffffff8108cc10>] wake_up_process+0x30/0x70

which lock already depends on the new lock.

the existing dependency chain (in reverse order) is:

-> #1 (&rq->lock){-.....}:

[<ffffffff8109ba57>] task_fork_fair+0x27/0xf0 (CALL STACK)

-> #0 (&p->pi_lock){.....}:

[<ffffffff8108cc1b>] wake_up_process+0x3b/0x70 (CALL STACK)

cpu0	cpu1
----	----
lock(&rq->lock);	
	lock(&p->pi_lock);
	lock(&rq->lock);
lock(&p->pi_lock);	

Report Example 2

Byungchul Park max.byungchul.park@gmail.com

```
[    0.031237] =======================================================
[    0.031946] [ INFO: possible circular locking dependency detected ]
[    0.032000] 4.9.0+ #16 Not tainted
[    0.032000] -------------------------------------------------------
[    0.032000] swapper/0/1 is trying to acquire lock:
[    0.032000]  (&p->pi_lock){.....}, at: [<ffffffff8108cc1b>] wake_up_process+0x3b/0x70
[    0.032000] but task is already holding lock:
[    0.032000]  (&rq->lock){-.....}, at: [<ffffffff8108cc10>] wake_up_process+0x30/0x70
[    0.032000] which lock already depends on the new lock.
[    0.032000]
[    0.032000]
[    0.032000] the existing dependency chain (in reverse order) is:
[    0.032000]
-> #1 (&rq->lock){-.....}:
[    0.032000]        [    0.032000] [<ffffffff810ad471>] __lock_acquire+0x10c1/0x1220
[    0.032000]        [    0.032000] [<ffffffff810ad9d0>] lock_acquire+0xb0/0x1d0
[    0.032000]        [    0.032000] [<ffffffff819e2e9c>] _raw_spin_lock+0x2c/0x40
[    0.032000]        [    0.032000] [<ffffffff8109ba57>] task_fork_fair+0x27/0xf0
[    0.032000]        [    0.032000] [<ffffffff8108d9fd>] sched_fork+0x24d/0x3d0
[    0.032000]        [    0.032000] [<ffffffff8105ad02>] copy_process.part.47+0x622/0x1e10
[    0.032000]        [    0.032000] [<ffffffff8105c6d2>] _do_fork+0xe2/0x6e0
[    0.032000]        [    0.032000] [<ffffffff8105ccf4>] kernel_thread+0x24/0x30
[    0.032000]        [    0.032000] [<ffffffff819db9fe>] rest_init+0x1e/0x130
[    0.032000]        [    0.032000] [<ffffffff81f85021>] start_kernel+0x42b/0x438
[    0.032000]        [    0.032000] [<ffffffff81f8458c>] x86_64_start_reservations+0x2a/0x2c
[    0.032000]        [    0.032000] [<ffffffff81f84678>] x86_64_start_kernel+0xea/0xed
[    0.032000]
-> #0 (&p->pi_lock){.....}:
[    0.032000]        [    0.032000] [<ffffffff810ab4f3>] check_prev_add+0x723/0x730
[    0.032000]        [    0.032000] [<ffffffff810ad471>] __lock_acquire+0x10c1/0x1220
[    0.032000]        [    0.032000] [<ffffffff810ad9d0>] lock_acquire+0xb0/0x1d0
[    0.032000]        [    0.032000] [<ffffffff819e3017>] _raw_spin_lock_irqsave+0x37/0x50
[    0.032000]        [    0.032000] [<ffffffff8108cc1b>] wake_up_process+0x3b/0x70
[    0.032000]        [    0.032000] [<ffffffff81081a88>] __kthread_create_on_node+0x128/0x220
[    0.032000]        [    0.032000] [<ffffffff81081bb4>] kthread_create_on_node+0x34/0x40
[    0.032000]        [    0.032000] [<ffffffff81082449>] kthread_create_on_q+0x29/0x70
[    0.032000]        [    0.032000] [<ffffffff81085825>] __smpboot_create_thread.part.3+0x65/0xf0
[    0.032000]        [    0.032000] [<ffffffff81085950>] smpboot_register_perq_thread_qmask+0xa0/0x100
[    0.032000]        [    0.032000] [<ffffffff81f9e1e2>] spawn_ksoftirqd+0x36/0x40
[    0.032000]        [    0.032000] [<ffffffff81000408>] do_one_initcall+0x38/0x150
[    0.032000]        [    0.032000] [<ffffffff81f8516c>] kernel_init_freeable+0x13e/0x27e
[    0.032000]        [    0.032000] [<ffffffff819dbb19>] kernel_init+0x9/0x100
[    0.032000]        [    0.032000] [<ffffffff819e3987>] ret_from_fork+0x27/0x40
[    0.032000]
[    0.032000] other info that might help us debug this:
[    0.032000]
[    0.032000]  Possible unsafe locking scenario:
[    0.032000]
[    0.032000]        cpu0                    cpu1
[    0.032000]        ----                    ----
[    0.032000]   lock(&rq->lock);
[    0.032000]                                lock(&p->pi_lock);
[    0.032000]                                lock(&rq->lock);
[    0.032000]   lock(&p->pi_lock);
[    0.032000]
[    0.032000]  *** DEADLOCK ***
[    0.032000]
[    0.032000] 3 locks held by swapper/0/1:
[    0.032000]  #0:  (cpu_hotplug.dep_map){.+.+.+.+}, at: [<ffffffff8105dd0f>] get_online_qs+0x1f/0x70
[    0.032000]  #1:  (smpboot_threads_lock){+.+.+.+}, at: [<ffffffff810858e3>] smpboot_register…qmask+0x33/0x100
[    0.032000]  #2:  (&rq->lock){-.....}, at: [<ffffffff8108cc10>] wake_up_process+0x30/0x70
[    0.032000]
[    0.032000] stack backtrace:
[    0.032000] q: 0 PID: 1 Comm: swapper/0 Not tainted 4.9.0+ #16
[    0.032000] Hardware name: QEMU Standard PC (i440FX + PIIX, 1996), BIOS Bochs 01/01/2011
[    0.032000]  ffffc900000d39b8 ffffffff8138ab5e ffff88001e370870 ffffffff827a8aa0
[    0.032000]  ffffc900000d39f8 ffffffff811506fa ffffc900000d3a20 ffff88001e370870
[    0.032000]  ffff88001e3708a0 ffff88001e370000 a579d3845e226952 ffffffff810a8dd0
[    0.032000] Call Trace:
[    0.032000]  [<ffffffff8138ab5e>] dump_stack+0x67/0x99
[    0.032000]  [<ffffffff811506fa>] print_circular_bug+0x2f0/0x2fe
[    0.032000]  [<ffffffff810a8dd0>] ? graph_unlock+0x80/0x80
[    0.032000]  [<ffffffff810ab4f3>] check_prev_add+0x723/0x730
[    0.032000]  [<ffffffff810ad471>] __lock_acquire+0x10c1/0x1220
[    0.032000]  [<ffffffff810a8dd0>] ? graph_unlock+0x80/0x80
[    0.032000]  [<ffffffff810ad9d0>] lock_acquire+0xb0/0x1d0
[    0.032000]  [<ffffffff8108cc1b>] ? wake_up_process+0x3b/0x70
[    0.032000]  [<ffffffff81085570>] ? sort_range+0x20/0x20
[    0.032000]  [<ffffffff819e3017>] _raw_spin_lock_irqsave+0x37/0x50
[    0.032000]  [<ffffffff8108cc1b>] ? wake_up_process+0x3b/0x70
[    0.032000]  [<ffffffff8108cc1b>] wake_up_process+0x3b/0x70
[    0.032000]  [<ffffffff81081a88>] __kthread_create_on_node+0x128/0x220
[    0.032000]  [<ffffffff810aa6b0>] ? check_usage_backwards+0x70/0x120
[    0.032000]  [<ffffffff810aa6b0>] ? check_usage_backwards+0x70/0x120
[    0.032000]  [<ffffffff81081bb4>] kthread_create_on_node+0x34/0x40
[    0.032000]  [<ffffffff811ac760>] ? kmem_cache_alloc_node_trace+0x1c0/0x230
[    0.032000]  [<ffffffff81082449>] kthread_create_on_q+0x29/0x70
[    0.032000]  [<ffffffff81085825>] __smpboot_create_thread.part.3+0x65/0xf0
[    0.032000]  [<ffffffff81085950>] smpboot_register_perq_thread_qmask+0xa0/0x100
[    0.032000]  [<ffffffff81f9e1ac>] ? trace_event_define_fields_irq_handler_exit+0x6a/0x6a
[    0.032000]  [<ffffffff81f9e1e2>] spawn_ksoftirqd+0x36/0x40
[    0.032000]  [<ffffffff81000408>] do_one_initcall+0x38/0x150
[    0.032000]  [<ffffffff8102e0e3>] ? print_q_info+0x83/0xf0
[    0.032000]  [<ffffffff81f8516c>] kernel_init_freeable+0x13e/0x27e
[    0.032000]  [<ffffffff819dbb19>] ? kernel_init+0x9/0x100
[    0.032000]  [<ffffffff819dbb10>] ? rest_init+0x130/0x130
[    0.032000]  [<ffffffff819dbb19>] kernel_init+0x9/0x100
[    0.032000]  [<ffffffff819e3987>] ret_from_fork+0x27/0x40
```

Report Example 2

Byungchul Park max.byungchul.park@gmail.com

```
[    0.031237] ======================================================
[    0.031946] [ INFO: possible circular locking dependency detected ]
[    0.032000] 4.9.0+ #16 Not tainted
[    0.032000] -------------------------------------------------------
[    0.032000] swapper/0/1 is trying to acquire lock:
[    0.032000]  (&p->pi_lock){......}, at: [<ffffffff8108cc1b>] wake_up_process+0x3b/0x70
[    0.032000] but task is already holding lock:
[    0.032000]  (&rq->lock){-.....}, at: [<ffffffff8108cc10>] wake_up_process+0x30/0x70
[    0.032000] which lock already depends on the new lock.
[    0.032000]
[    0.032000]
[    0.032000] the existing dependency chain (in reverse order) is:
[    0.032000]
-> #1 (&rq->lock){-.....}:
[    0.032000]       [    0.032000] [<ffffffff810ad471>] __lock_acquire+0x10c1/0x1220
[    0.032000]       [    0.032000] [<ffffffff810ad9d0>] lock_acquire+0xb0/0x1d0
[    0.032000]       [    0.032000] [<ffffffff819e2e9c>] _raw_spin_lock+0x2c/0x40
[    0.032000]       [    0.032000] [<ffffffff8109ba57>] task_fork_fair+0x27/0xf0
[    0.032000]       [    0.032000] [<ffffffff8108d9fd>] sched_fork+0x24d/0x3d0
[    0.032000]       [    0.032000] [<ffffffff8105ad02>] copy_process.part.47+0x622/0x1e10
[    0.032000]       [    0.032000] [<ffffffff8105c6d2>] _do_fork+0xe2/0x6e0
[    0.032000]       [    0.032000] [<ffffffff8105ccf4>] kernel_thread+0x24/0x30
[    0.032000]       [    0.032000] [<ffffffff819db9fe>] rest_init+0x1e/0x130
[    0.032000]       [    0.032000] [<ffffffff81f85021>] start_kernel+0x42b/0x438
```

```
[    0.032000]        [    0.032000] [<ffffffff81f84678>] x86_64_start_kernel+0xea/0xed
[    0.032000]
-> #0 (&p->pi_lock){.....}:
[    0.032000]        [    0.032000] [<ffffffff810ab4f3>] check_prev_add+0x723/0x730
[    0.032000]        [    0.032000] [<ffffffff810ad471>] __lock_acquire+0x10c1/0x1220
[    0.032000]        [    0.032000] [<ffffffff810ad9d0>] lock_acquire+0xb0/0x1d0
[    0.032000]        [    0.032000] [<ffffffff819e3017>] _raw_spin_lock_irqsave+0x37/0x50
[    0.032000]        [    0.032000] [<ffffffff8108cc1b>] wake_up_process+0x3b/0x70
[    0.032000]        [    0.032000] [<ffffffff81081a88>] __kthread_create_on_node+0x128/0x220
[    0.032000]        [    0.032000] [<ffffffff81081bb4>] kthread_create_on_node+0x34/0x40
[    0.032000]        [    0.032000] [<ffffffff81082449>] kthread_create_on_q+0x29/0x70
[    0.032000]        [    0.032000] [<ffffffff81085825>] __smpboot_create_thread.part.3…
[    0.032000]        [    0.032000] [<ffffffff81085950>] smpboot_register_perq_thread_qmask…
[    0.032000]        [    0.032000] [<ffffffff81f9e1e2>] spawn_ksoftirqd+0x36/0x40
[    0.032000]        [    0.032000] [<ffffffff81000408>] do_one_initcall+0x38/0x150
[    0.032000]        [    0.032000] [<ffffffff81f8516c>] kernel_init_freeable+0x13e/0x27e
[    0.032000]        [    0.032000] [<ffffffff819dbb19>] kernel_init+0x9/0x100
[    0.032000]        [    0.032000] [<ffffffff819e3987>] ret_from_fork+0x27/0x40
[    0.032000]
[    0.032000] other info that might help us debug this:
[    0.032000]
[    0.032000]  Possible unsafe locking scenario:
[    0.032000]
[    0.032000]        cpu0                    cpu1
```

```
[    0.032000]        cpu0                    cpu1
[    0.032000]        ----                    ----
[    0.032000]   lock(&rq->lock);
[    0.032000]                                lock(&p->pi_lock);
[    0.032000]                                lock(&rq->lock);
[    0.032000]   lock(&p->pi_lock);
[    0.032000]
[    0.032000]  *** DEADLOCK ***
[    0.032000]
[    0.032000] 3 locks held by swapper/0/1:
[    0.032000]  #0:  (cpu_hotplug.dep_map){.+.+.+}, at: [<ffffffff8105dd0f>] get_online_qs…
[    0.032000]  #1:  (smpboot_threads_lock){+.+.+.}, at: [<ffffffff810858e3>] smpboot_register…
[    0.032000]  #2:  (&rq->lock){-.....}, at: [<ffffffff8108cc10>] wake_up_process+0x30/0x70
[    0.032000]
[    0.032000] stack backtrace:
[    0.032000] q: 0 PID: 1 Comm: swapper/0 Not tainted 4.9.0+ #16
[    0.032000] Hardware name: QEMU Standard PC (i440FX + PIIX, 1996), BIOS Bochs…
[    0.032000]  ffffc900000d39b8 ffffffff8138ab5e ffff88001e370870 ffffffff827a8aa0
[    0.032000]  ffffc900000d39f8 ffffffff811506fa ffffc900000d3a20 ffff88001e370870
[    0.032000]  ffff88001e3708a0 ffff88001e370000 a579d3845e226952 ffffffff810a8dd0
[    0.032000] Call Trace:
[    0.032000]  [<ffffffff8138ab5e>] dump_stack+0x67/0x99
[    0.032000]  [<ffffffff811506fa>] print_circular_bug+0x2f0/0x2fe
[    0.032000]  [<ffffffff810a8dd0>] ? graph_unlock+0x80/0x80
```

```
[    0.032000] Call Trace:
[    0.032000]  [<ffffffff8138ab5e>] dump_stack+0x67/0x99
[    0.032000]  [<ffffffff811506fa>] print_circular_bug+0x2f0/0x2fe
[    0.032000]  [<ffffffff810a8dd0>] ? graph_unlock+0x80/0x80
[    0.032000]  [<ffffffff810ab4f3>] check_prev_add+0x723/0x730
[    0.032000]  [<ffffffff810ad471>] __lock_acquire+0x10c1/0x1220
[    0.032000]  [<ffffffff810a8dd0>] ? graph_unlock+0x80/0x80
[    0.032000]  [<ffffffff810ad9d0>] lock_acquire+0xb0/0x1d0
[    0.032000]  [<ffffffff8108cc1b>] ? wake_up_process+0x3b/0x70
[    0.032000]  [<ffffffff81085570>] ? sort_range+0x20/0x20
[    0.032000]  [<ffffffff819e3017>] _raw_spin_lock_irqsave+0x37/0x50
[    0.032000]  [<ffffffff8108cc1b>] ? wake_up_process+0x3b/0x70
[    0.032000]  [<ffffffff8108cc1b>] wake_up_process+0x3b/0x70
[    0.032000]  [<ffffffff81081a88>] __kthread_create_on_node+0x128/0x220
[    0.032000]  [<ffffffff810aa6b0>] ? check_usage_backwards+0x70/0x120
[    0.032000]  [<ffffffff810aa6b0>] ? check_usage_backwards+0x70/0x120
[    0.032000]  [<ffffffff81081bb4>] kthread_create_on_node+0x34/0x40
[    0.032000]  [<ffffffff811ac760>] ? kmem_cache_alloc_node_trace+0x1c0/0x230
[    0.032000]  [<ffffffff81082449>] kthread_create_on_q+0x29/0x70
[    0.032000]  [<ffffffff81085825>] __smpboot_create_thread.part.3+0x65/0xf0
[    0.032000]  [<ffffffff81085950>] smpboot_register_perq_thread_qmask+0xa0/0x100
[    0.032000]  [<ffffffff81f9e1ac>] ? trace_event_define_fields_irq_handler_exit+0x6a/0x6a
[    0.032000]  [<ffffffff81f9e1e2>] spawn_ksoftirqd+0x36/0x40
[    0.032000]  [<ffffffff81000408>] do_one_initcall+0x38/0x150
```

Appendix

Typical locks

e.g. spin lock, mutex

What Lockdep Covers

Byungchul Park max.byungchul.park@gmail.com

No more dependencies?

Enough ?

More dependencies exist!

Not Enough !

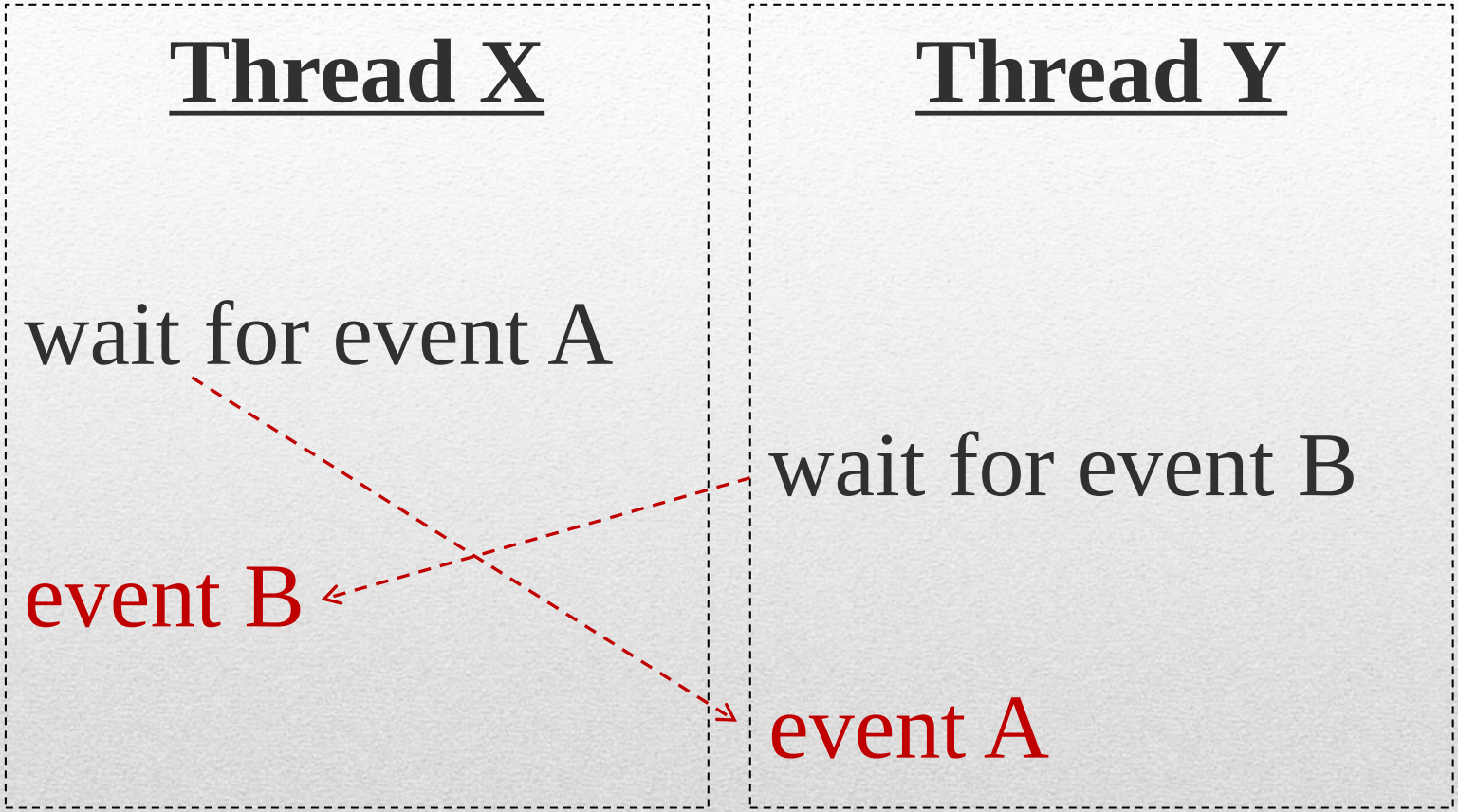
Not only locks but any waiters can also cause a deadlock.

What We Missed

Byungchul Park max.byungchul.park@gmail.com

Thread X	Thread Y
wait for event A	
	wait for event B
event B	
	event A

Example


Thread X
Thread Y
wait for event A
wait for event B
event B
event A

Example

lock A

lock B (while holding A.)

= A depends on B. (A → B)

Redefine Dependency

Byungchul Park max.byungchul.park@gmail.com

In case that two waiters for A and B exist, B should be triggered to trigger A.

= A depends on B. (A → B)

Redefine Dependency

Byungchul Park max.byungchul.park@gmail.com

In case that two waiters for A and B exist, B should be triggered to trigger A.

= A depends on B. (A → B)

Redefine Dependency

Byungchul Park max.byungchul.park@gmail.com

Thread X	Thread Y
mutex_lock A	mutex_lock A
…	…
wait_for_event B	mutex_unlock A
…	…
mutex_unlock A	event B

Example

mutex_lock A

…

wait_for_event B

…

mutex_unlock A

Dependency ?

mutex_lock A

…

wait_for_event B

…

mutex_unlock A

Dependency ?

mutex_lock A

…

wait_for_event B

…

mutex_unlock A

A → B

Dependency !

Byungchul Park max.byungchul.park@gmail.com

Global Space

Dependency Graph

Byungchul Park max.byungchul.park@gmail.com

mutex_lock A

…

mutex_unlock A

…

event B

Dependency ?

mutex_lock A

…

mutex_unlock A

…

event B

Dependency ?

mutex_lock A

…

mutex_unlock A

…

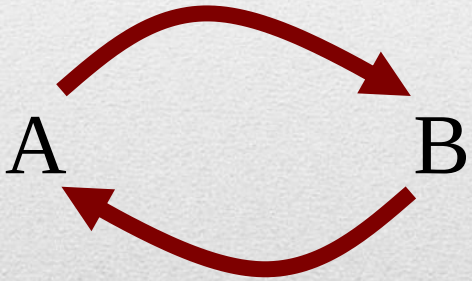
event B

B → A

Dependency !

Byungchul Park max.byungchul.park@gmail.com

Global Space



Dependency Graph

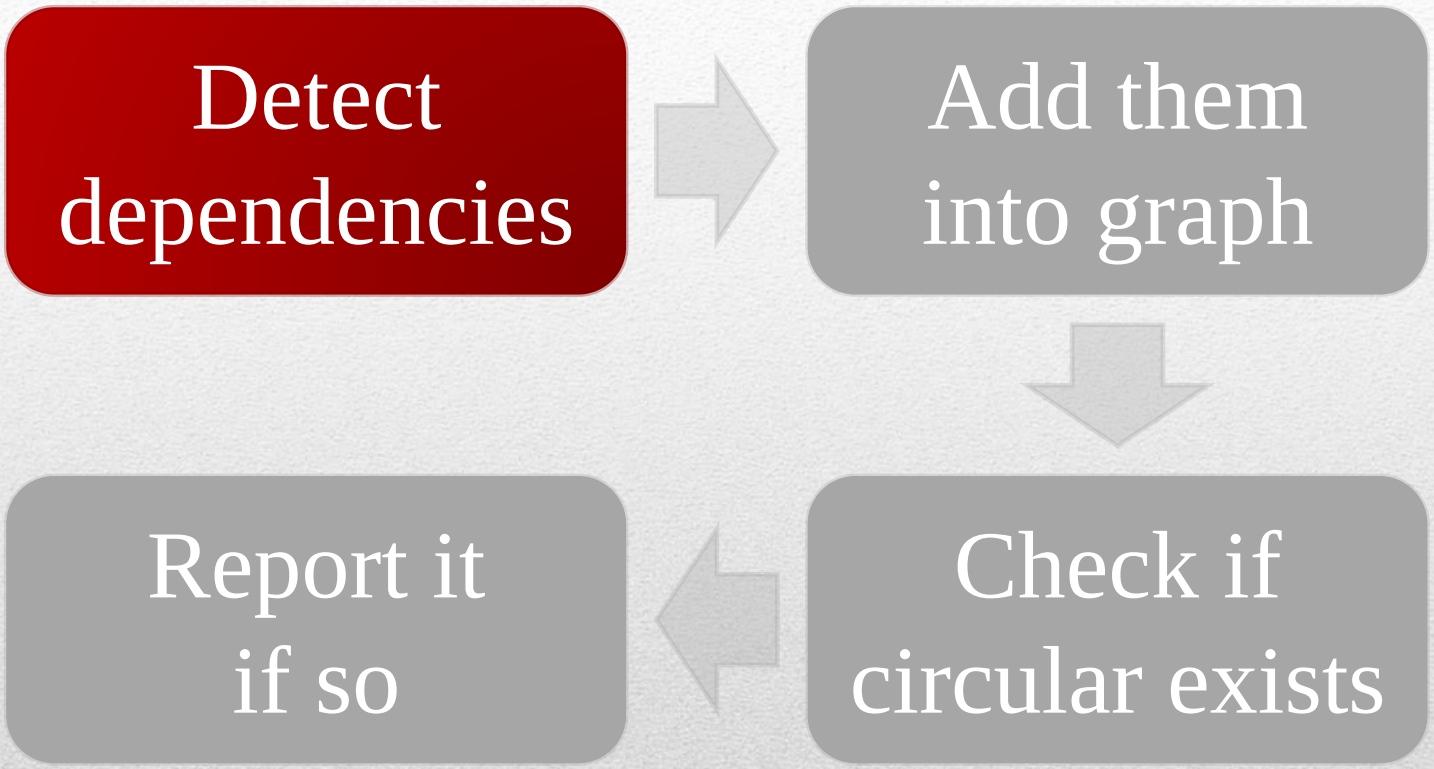
Byungchul Park max.byungchul.park@gmail.com

Circular Dependencies

Byungchul Park max.byungchul.park@gmail.com

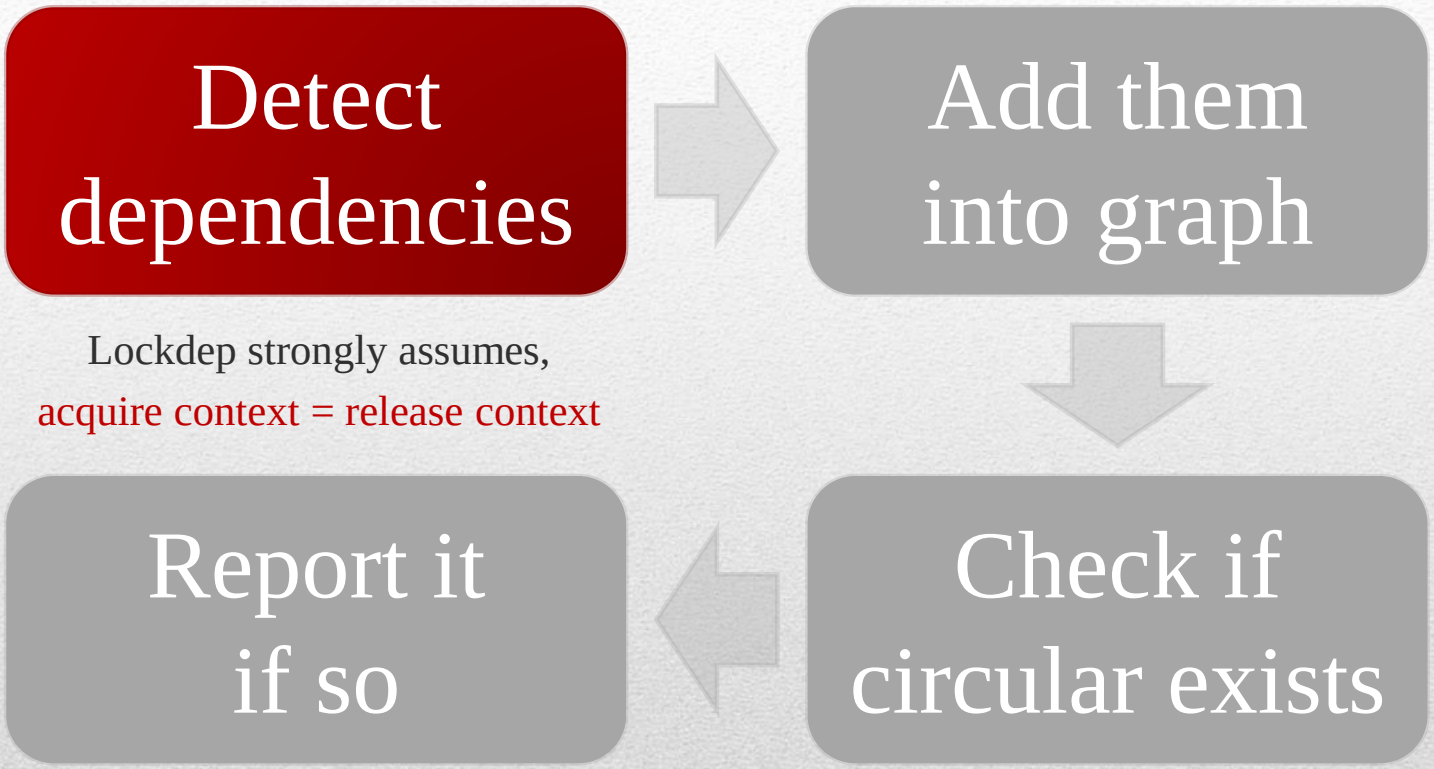
Why not?

Byungchul Park max.byungchul.park@gmail.com



How Lockdep Works

Byungchul Park max.byungchul.park@gmail.com



Lockdep strongly assumes,

acquire context = release context

How Lockdep Works

Byungchul Park max.byungchul.park@gmail.com

Thread X	Thread Y
mutex_lock A	mutex_lock A
…	…
wait_for_event B acquire context	mutex_unlock A
…	…
mutex_unlock A	event B release context

Example

acquire context = release context

acquire context ≠ release context

Generalize Lockdep

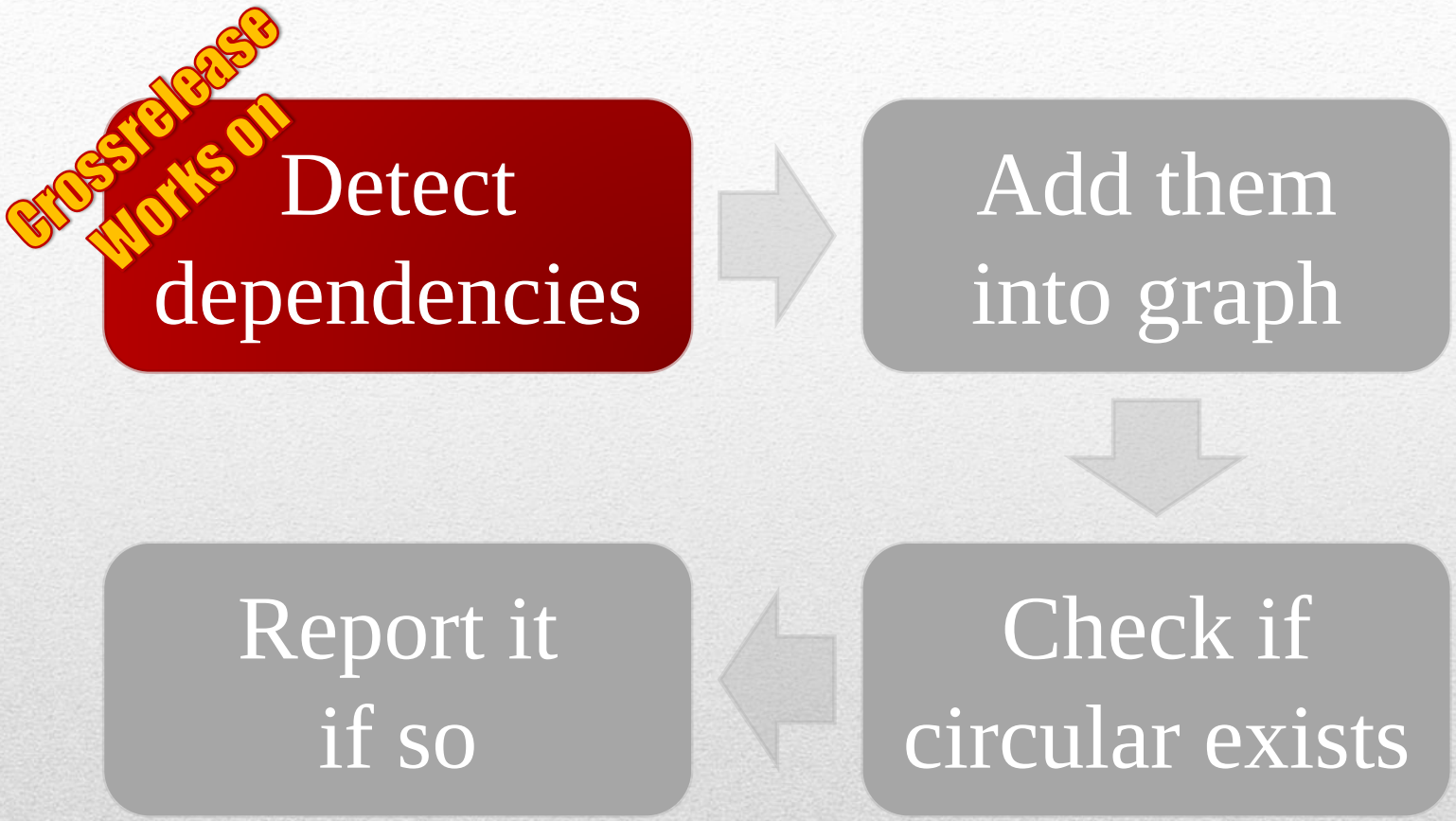
Byungchul Park max.byungchul.park@gmail.com

acquire context = release context

Crossrelease solves the issue!

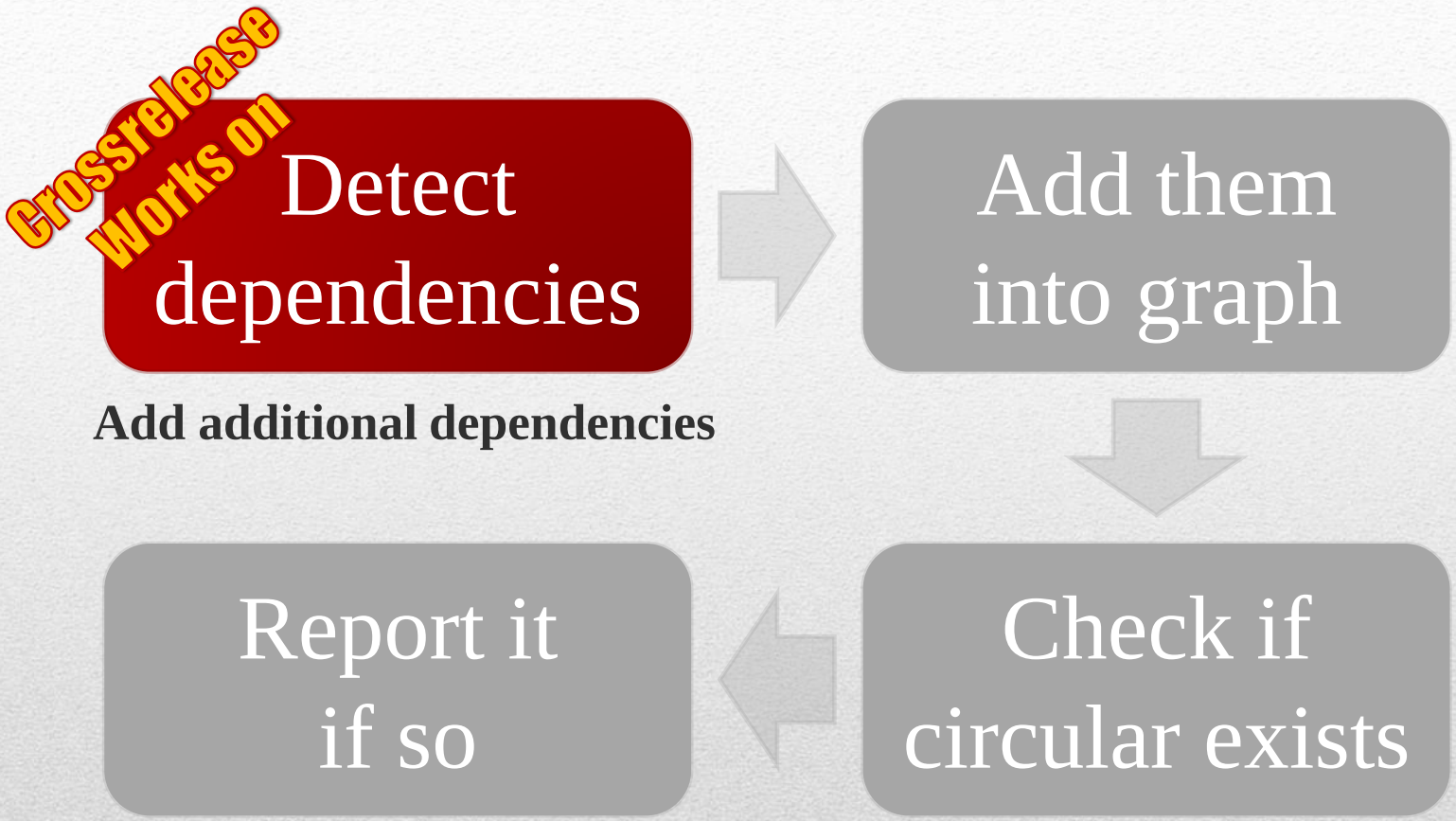
acquire context ≠ release context

Solution



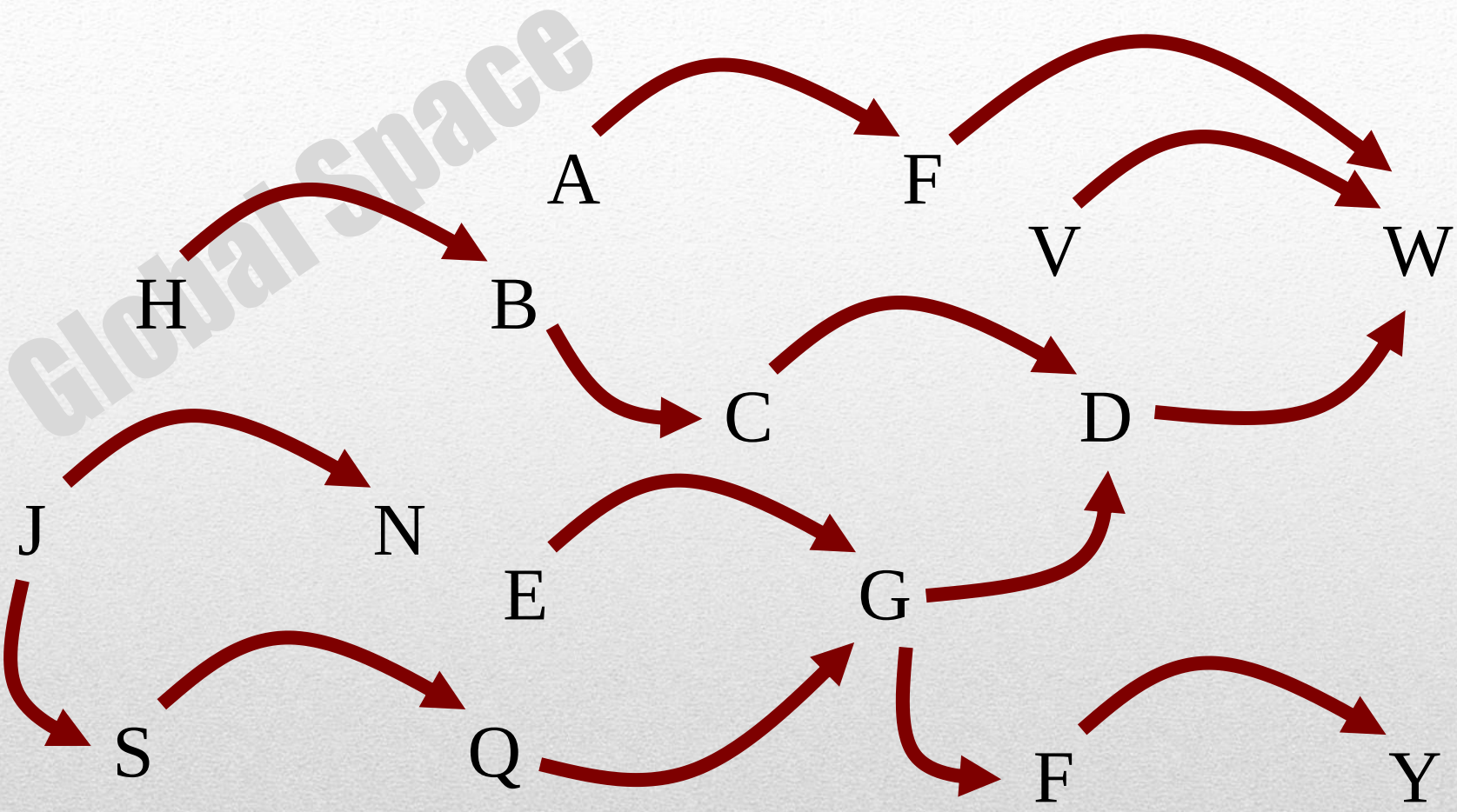
What Crossrelease Does

Byungchul Park max.byungchul.park@gmail.com



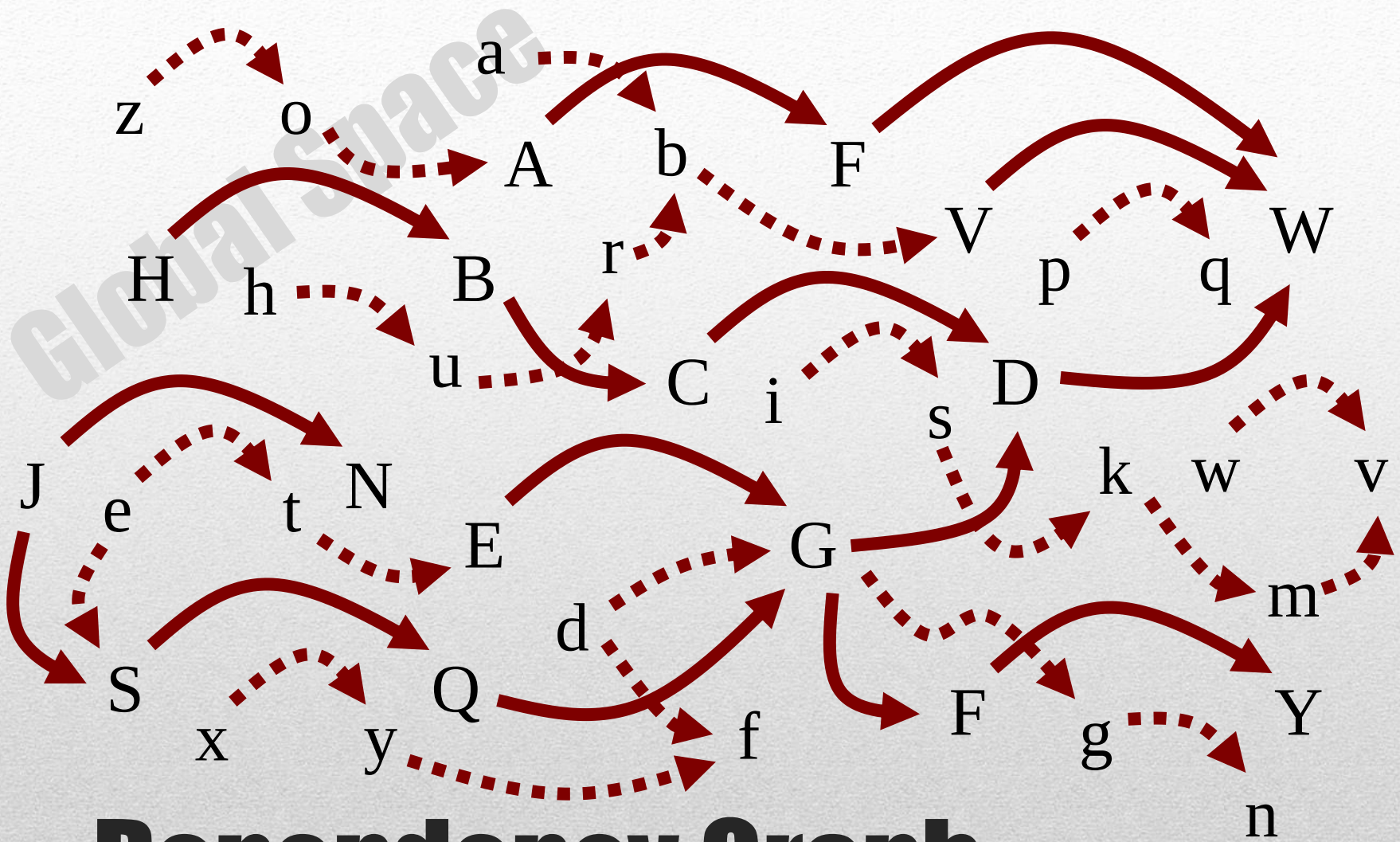
What Crossrelease Does

Byungchul Park max.byungchul.park@gmail.com



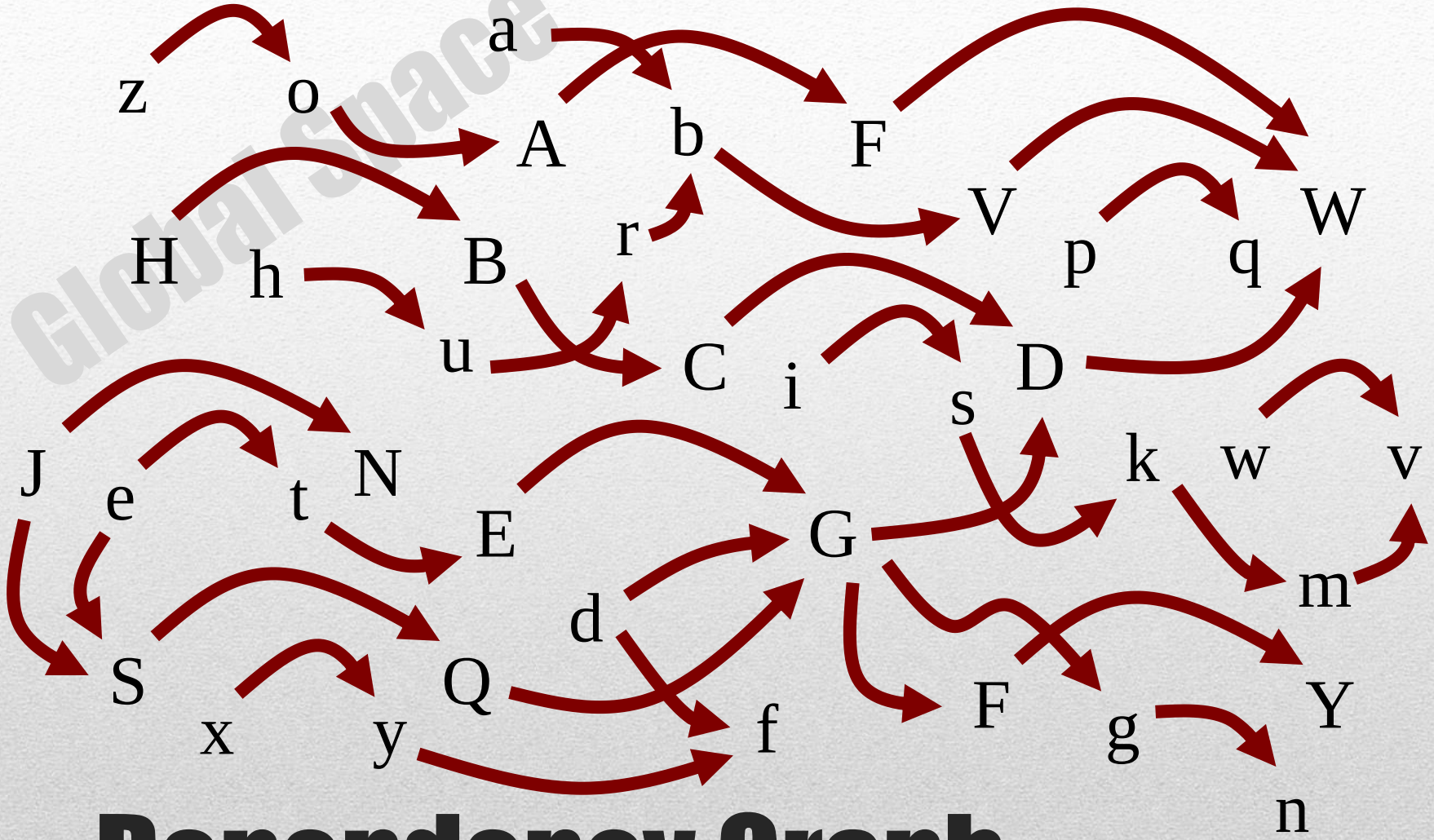
Dependency Graph

Byungchul Park max.byungchul.park@gmail.com



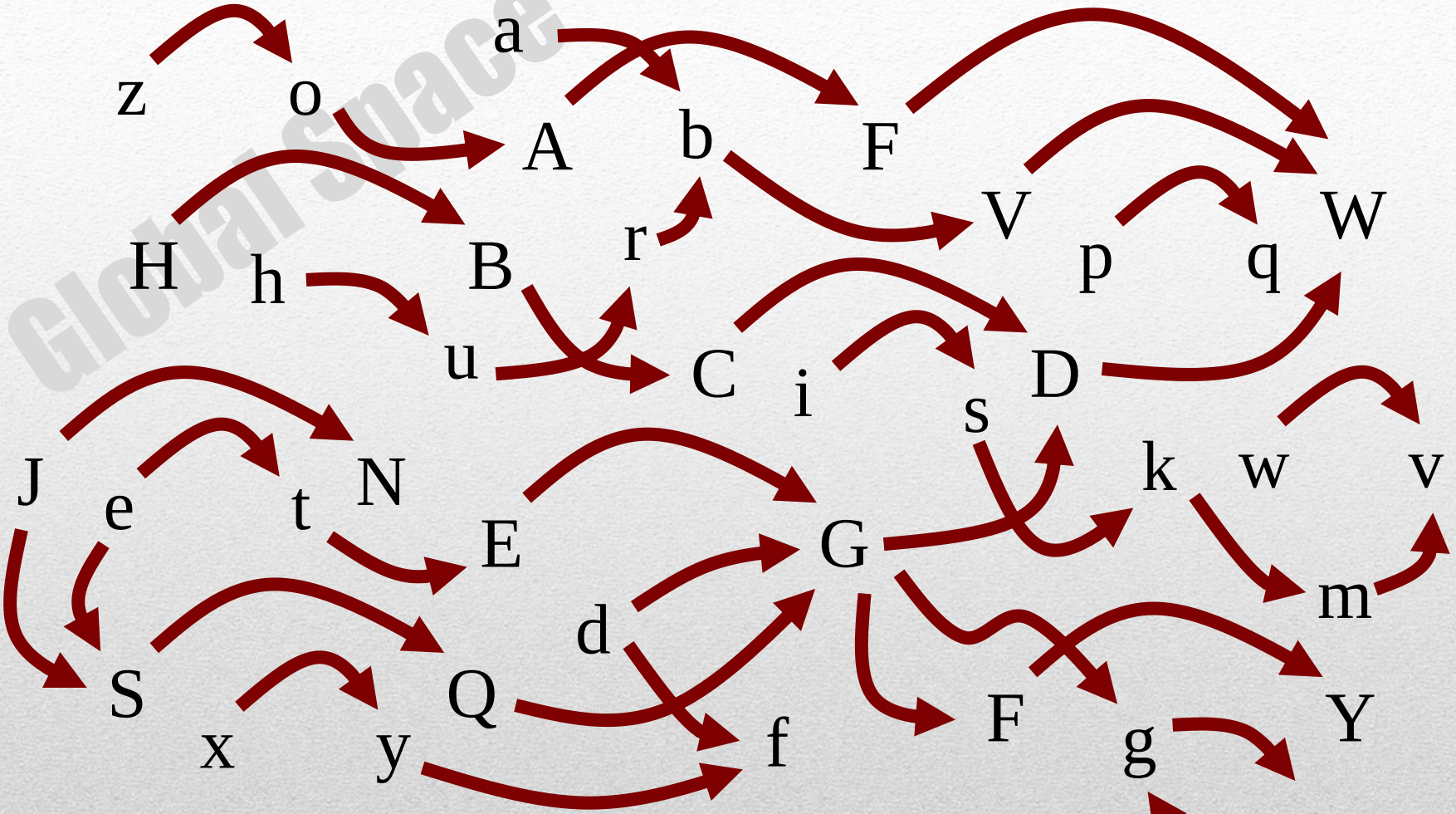
Dependency Graph

Byungchul Park max.byungchul.park@gmail.com



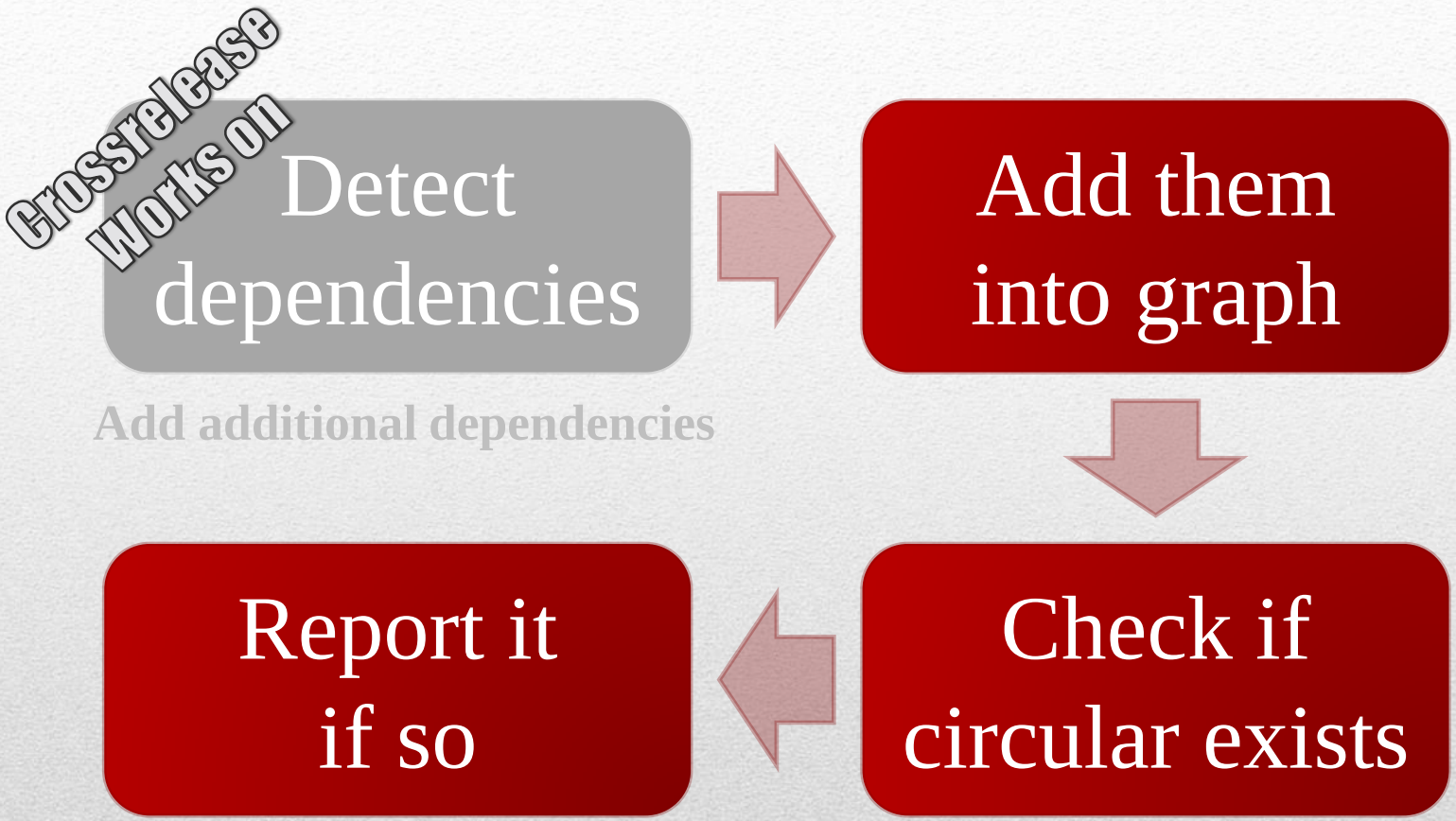
Dependency Graph

Byungchul Park max.byungchul.park@gmail.com



Dependency Graph

Byungchul Park max.byungchul.park@gmail.com



Original Lockdep Works

Byungchul Park max.byungchul.park@gmail.com

Makes Lockep stronger.

Why Crossrelease

Byungchul Park max.byungchul.park@gmail.com

	Without crossrelease	With crossrelease
Works with typical locks	Possible	Possible
Works with any waiters	Impossible	Possible
Works with page locks	Impossible	Possible
Overhead	Small	Large
Internal implementation	Simple	Complicated
What users should do	Nothing	Nothing

Before / After

Question