Fiji Priority Rollback Protocol

Lukasz Ziarek
Fiji Systems Inc.

Filip Pizlo, Ethan Blanton, and Jan Vitek
Communication between mixed-criticality partitions

Situational Awareness  Flight System
Communication between mixed-criticality partitions

Situational Awareness

Middleware

Flight System
Communication between mixed-criticality partitions

Situational Awareness

package data

Middleware

Flight System
Communication between mixed-criticality partitions

Situational Awareness → Middleware → Flight System

send data
Communication between mixed-criticality partitions

Situational Awareness

Flight System

unpack data

Middleware
Communication between mixed-criticality partitions

Situational Awareness

Middleware

Flight System
update shared state

airspace
data structure
Communication between mixed-criticality partitions

Situational Awareness

Flight System

Middleware

airspace
data structure

typically a communication protocol

Wednesday, August 18, 2010
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
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

how do we guarantee responsiveness of higher criticality partitions?
Communication between mixed-criticality partitions

Situational Awareness  Flight System

airspace  data structure
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

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
Inspiration from Transactions

- Atomic replacement for locks
- Automatic serializability detection
- Runtime monitoring
- Aborts - ability to rollback
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 [X]
- Automatic serializability detection [X]
- 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
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)
        ...
}

Wednesday, August 18, 2010
Write Buffering

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

Buffer

set to acquired in WB mode
Write Buffering

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

Buffer

lock set to acquired in WB mode
Write Buffering

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

Buffer

set to acquired in WB mode
Write Buffering

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

Buffer

lock set to acquired in WB mode
Write Buffering

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

Buffer
Write Buffering

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

Buffer

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

Set to acquired in WB mode

Wednesday, August 18, 2010
Write Buffering

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

Buffer
Write Buffering

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

Buffer

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

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

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

Buffer

lock set to acquired in WB mode
Write Buffering

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

Buffer

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

Lock set to acquired in WB mode
Write Buffering

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

Buffer

lock set to acquired in WB mode

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

Wednesday, August 18, 2010
Write Buffering

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

Buffer

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

Commit

lock set to acquired in WB mode
Write Buffering

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

Commit

Set lock mode to commit
```

Buffer

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

Lock set to acquired in WB mode

Wednesday, August 18, 2010
Acquisition by higher priority thread

High Priority Thread

lock

WB

foo.a, w
foo.c, z
foo.b, y
foo.a, x
Acquisition by higher priority thread

lock

High Priority Thread

WB

foo.a, w
foo.c, z
foo.b, y
foo.a, x
Acquisition by higher priority thread

High Priority Thread

lock

WB

foo.a, w
foo.c, z
foo.b, y
foo.a, x
Acquisition by higher priority thread

High Priority Thread

lock

WB

- foo.a, w
- foo.c, z
- foo.b, y
- foo.a, x
Acquisition by higher priority thread

High Priority Thread

lock

WB

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

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

Wednesday, August 18, 2010
Acquisition by higher priority thread

High Priority Thread Acquires Lock

No priority boosting!
Acquisition by higher priority thread

Main Memory

lock

Commit

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

High Priority Thread

Wednesday, August 18, 2010
Acquisition by higher priority thread
Acquisition by higher priority thread

Main Memory

High Priority Thread

lock

Commit

foo.a, w
foo.c, z
foo.b, y
foo.a, x
Acquisition by higher priority thread

High Priority Thread

lock

low priority thread

Commit

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

Boost

Main Memory

Wednesday, August 18, 2010
Acquisition by higher priority thread

High Priority Thread

lock

Commit

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

low priority thread

Boost

Main Memory

Wednesday, August 18, 2010
Acquisition by higher priority thread

- Lock
- Commit
  - foo.a, w
  - foo.c, z
  - foo.b, y
  - foo.a, x

- Boost
- Main Memory

High Priority Thread

low priority thread

Wednesday, August 18, 2010
Acquisition by higher priority thread

High Priority Thread

lock

boost

low priority thread

Commit

Main Memory

foo.a , x
foo.b , y
foo.c , z
foo.a , w
foo.a , x
Acquisition by higher priority thread

High Priority Thread

lock

low priority thread

Commit

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

Boost

Main Memory

Wednesday, August 18, 2010
Acquisition by higher priority thread

High Priority Thread

lock

Commit

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

Low Priority Thread

Boost

Main Memory
Acquisition by higher priority thread

- lock
- High Priority Thread
- Boost
- low priority thread
- Commit

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

Main Memory

Wednesday, August 18, 2010
Acquisition by higher priority thread

lock

High Priority Thread

low priority thread

Commit

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

Main Memory

Boost

Wednesday, August 18, 2010
Acquisition by higher priority thread

- Boost
- Commit

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

High Priority Thread

- Acquires Lock

Main Memory

Wednesday, August 18, 2010
Complexity Costs - Write Buffering Low Priority Thread

- Reads: $\log(\text{size of buffer})$  --- use RB tree
- Writes: $\log(\text{size of buffer})$
- 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
Write Logging

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

Main Memory

foo.a  foo.b  foo.c
Write Logging

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

Main Memory: foo.a, foo.b, foo.c

log set to acquired in WL mode
Write Logging

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

Main Memory

foo.a  foo.b  foo.c

Log

foo.a, 1

lock set to acquired in WL mode

Wednesday, August 18, 2010
Write Logging

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

Main Memory: foo.a, foo.b, foo.c

Log: foo.a, 1, foo.b, 2

lock set to acquired in WL mode
Write Logging

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

Set lock to acquired in WL mode

Log:
- foo.a, 1
- foo.b, 2
- foo.c, 3
- foo.a, 1

Main Memory:
- foo.a
- foo.b
- foo.c
Write Logging

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

Log

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

Main Memory
- foo.a
- foo.b
- foo.c
Write Logging

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

Main Memory

foo.a  foo.b  foo.c

Log

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

lock set to acquired in WL mode
Write Logging

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 WL mode

Main Memory

foo.a  foo.b  foo.c

Log

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

Wednesday, August 18, 2010
Write Logging

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

Commit

lock set to acquired in WL mode

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

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

Main Memory

foo.a
foo.b
foo.c
Acquisition by higher priority thread

Main Memory

High Priority Thread

lock

WL

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

Wednesday, August 18, 2010
Acquisition by higher priority thread

High Priority Thread

Main Memory

lock

WL

foo.a , x
foo.c , 3
foo.b , 2
foo.a , 1
Acquisition by higher priority thread
Acquisition by higher priority thread

High Priority Thread

lock

Boost

low priority thread

Main Memory

foo.a, 1

foo.b, 2

foo.c, 3

foo.a, x

foo.b, 2

foo.a, 1
Acquisition by higher priority thread

- Lock
- High Priority Thread
- Low Priority Thread
- Boost
- WL
- Main Memory

- foo.a, 1
- foo.b, 2
- foo.c, 3
- foo.a, x
- foo.b, 2
- foo.a, 1
Acquisition by higher priority thread

- High Priority Thread
- lock
- Boost
- low priority thread
- WL
- Main Memory
- foo.a, 1
- foo.b, 2
- foo.c, 3
- foo.a, x
- foo.a, 1
- foo.b, 2
- foo.c, 3

Wednesday, August 18, 2010
Acquisition by higher priority thread

High Priority Thread

lock

low priority thread

WL

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

Main Memory

boost

foo.a , 1
foo.b , 2
foo.c , 3
Acquisition by higher priority thread

lock

High Priority Thread

Boost

low priority thread

WL

Main Memory

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

Wednesday, August 18, 2010
Acquisition by higher priority thread

High Priority Thread

lock

Boost

low priority thread

WL

Main Memory

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

Wednesday, August 18, 2010
Acquisition by higher priority thread

High Priority Thread

lock

Boost

WL

Main Memory

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

foo.a, x
foo.c, 3
foo.b, 2
foo.a, 1
Acquisition by higher priority thread

- **lock**
- **High Priority Thread**
- **low priority thread**
  - foo.a, x
  - foo.c, 3
  - foo.b, 2
  - foo.a, 1

- **Main Memory**
- **Boost**
- **WL**

Wednesday, August 18, 2010
Acquisition by higher priority thread

High Priority Thread

lock

Boost

Low Priority Thread

WL

foo.a, 1

foo.a, x

foo.c, 3

foo.b, 2

foo.a, 1

Main Memory
Acquisition by higher priority thread

High Priority Thread

Acquires Lock

lock

Boost

WL

Main Memory

foo.a , 1

foo.b , 2

foo.c , 3

foo.a , x

foo.b , 2

foo.c , 3

foo.a , 1

High Priority Thread

Low Priority Thread

Wednesday, August 18, 2010
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
Questions?