Transactional Memory: An Overview (part II)

Written by Harris et al.

Another Example

a = 20, b = 50, c = 0

T1

```
...
down(mutex);
a = a + 20;
b = b - 10;
c = c - b;
up(mutex);
...
```

T2

```
...
down(mutex);
b = b + 20;
c = c + b;
up(mutex);
...
```

Another Example

a = 20, b = 50, c = 0

T1

```
...
down(mutex);
a = a + 20;
b = b - 10;
c = c - b;
up(mutex);
...
```

T2

```
...
down(mutex);
b = b + 20;
c = c + b;
up(mutex);
...
```

if T1 before T2
a = 40
b = 60
c = 20

if T2 before T1
a = 40
b = 60
c = 10

Another Example

a = 20, b = 50, c = 0

T1

```
...
begin TX
a = a + 20;
b = b - 10;
c = c - b;
end TX
...
```

T2

```
...
begin TX
b = b + 20;
c = c + b;
end TX
...
```

if T1 commits before T2
a = 40
b = 70
c = 70

if T2 commits before T1
a = 40
b = 40
c = -50

Another Example (eager)

a = 20, b = 50, c = 0

```
T1

...
begin TX
a = a + 20;
b = b - 10;
c = c - b;
end TX
...
```

```
T2

...
begin TX
b = b + 20;
c = c + b;
end TX
...
```

Another Example (eager)

a = 20, b = 50, c = 0

```
T1

...
begin TX
a = a + 20;
b = b - 10;
c = c - b;
end TX

...
```

RS
a = 20

WS
a = 40

```
T2

...
begin TX
b = b + 20;
c = c + b;
end TX

...
```

RS

WS

Another Example (eager)

a = 20, b = 50, c = 0

```
T1

...
begin TX
a = a + 20;
b = b - 10;
c = c - b;
end TX

...
```

RS

a = 20

b = 50

WS

a = 40

b = 40

```
T2

...
begin TX
b = b + 20;
c = c + b;
end TX

...
```

RS

WS

Another Example (eager)

a = 20, b = 50, c = 0

```
T1

...
begin TX
a = a + 20;
b = b - 10;
c = c - b;
end TX

...
```

RS
a = 20
b = 50
b = 40
c = 0

WS
a = 40
b = 40
c = -40

```
T2

...
begin TX
b = b + 20;
c = c + b;
end TX

...
```

RS
b = 40

WS

Another Example (eager)

a = 20, b = 50, c = 0

T1

...

begin TX

a = a + 20;

b = b - 10;

c = c - b;

end TX

...

RS

a = 20

b =

b =

c = 0

WS

a = 40

b = 40

c = -40

Should we abort T2?

T2

begin TX

b = b + 20;

c = c + b;

end TX

...

RS

b = 40

WS

Another Example (eager)

a = 20, b = 50, c = 0

T1

```
...
begin TX
a = a + 20;
b = b - 10;
c = c - b;
end TX
...
```

RS

a = 20

b = 50

b = 40

c = 0

WS

a = 40

b = 40

c = -40

T2

```
...
begin TX
b = b + 20;
c = c + b;
end TX
...
```

RS

b = 40

WS

b = 60

Another Example (eager)

a = 20, b = 50, c = 0

```
T1

...
begin TX
a = a + 20;
b = b - 10;
c = c - b;
end TX

...
```

RS

a = 20
b = 50
b = 40
c = 0

WS

a = 40
b = 40
c = -40

```
T2

...
begin TX
b = b + 20;
c = c + b;
end TX

...
```

RS

b = 40
b = 60
c = -40

WS

b = 60
c = 20

Another Example (eager)

- T1 commits first so the result in T2 is fine.
- What happen to both transactions if T2 commits first?

Another Example (eager)

a = 20, b = 50, c = 0

T1

...

begin TX

a = a + 20;

b = b - 10;

c = c - b;

end TX

...

RS

a = 20

b = 50

WS

a = 40

slightly behind T1

T2

...

begin TX

b = b + 20;

c = c + b;

end TX

...

RS

b = 50

WS

Another Example (eager)

a = 20, b = 50, c = 0

T1

```
...
begin TX
a = a + 20;
b = b - 10;
c = c - b;
end TX
...
```

RS

a = 20

b = 50

WS

a = 40

b = 40

T2

```
...
begin TX
b = b + 20;
c = c + b;
end TX
...
```

RS

b = 50

WS

Another Example (eager)

a = 20, b = 50, c = 0

T1

```
...
begin TX
a = a + 20;
b = b - 10;
c = c - b;
end TX
...
```

RS
a = 20
b = 50

WS
a = 40
b = 40

Abort T2

T2

```
...
begin TX
b = b + 20;
c = c + b;
end TX
...
```

RS
b = 50

WS

Another Example (lazy)

a = 20, b = 50, c = 0

```
T1

...
begin TX
a = a + 20;
b = b - 10;
c = c - b;
end TX

...
```

RS
a = 20
b = 50
b = 40
c = 0

WS
a = 40
b = 40
c = -40

```
T2

...
begin TX
b = b + 20;
c = c + b;
end TX

...
```

RS
b = 50

WS
b = 70

Another Example (lazy)

a = 20, b = 50, c = 0

```
T1

...
begin TX
a = a + 20;
b = b - 10;
c = c - b;
end TX
...
```

RS
a = 20
b = 50
b = 40
c = 0

WS
a = 40
b = 40
c = -40

Commit T1

```
T2

...
begin TX
b = b + 20;
c = c + b;
end TX
...
```

RS
b = 50
b = 70
c = 0

WS
b = 70
c = -70

Another Example (lazy)

a = 20, b = 50, c = 0

T1

```
...
begin TX
a = a + 20;
b = b - 10;
c = c - b;
end TX
...
```

RS
a = 20
b = 50
b = 40
c = 0

WS
a = 40
b = 40
c = -40

T2

```
...
begin TX
b = b + 20;
c = c + b;
end TX
...
```

RS
b = 50
b = 70
c = 0

WS
b = 70
c = -70

Abort T2

Hardware TM

- Minimalist
  - modifying cache consistency protocol
  - extending instruction set architecture
    - keep speculative state in a buffer

Hardware TM

- ISA support
  - delimiter instructions (STR and ETR)
  - special load and store (TLD and TST)
  - abort and validation (ABR and VLD)
    - VLD is used for eager versioning

Hardware TM

- Buffer or cache modifications
  - store speculative states in hardware buffer or extended cache
    - word level or cache-line level

Hardware TM

- Herlihy and Moss
  - read set and write set in data cache
  - transactional cache
  - two additional bits per cache line
    - discard pre-transaction values or discard speculative values

Software TM

- Two approaches
  - separation of ordinary data and transactional data
  - all data are ordinary but separate metadata structure for transactional data

Software TM

- Transactional data
  - store in object headers
  - special methods (openforread, openforwrite) to dynamically build read set and write set
  - private shadow copy of each object for each transaction

Software TM

- Metadata for transactional objects
    - special methods (openforreading, openforwriting) to track transactional accesses to ordinary objects

Software TM

- Detecting conflicts
  - two-phase locking
    - acquire lock at the beginning of transaction and relinquish lock at the end
  - hybrid
    - lock on write, version control on read

Summary

- Relieve the programmer's burden of coordinating parallelism
  - offload the responsibility to runtime systems
    - conflict detection and resolution
- Can be implemented in hardware and software
- More details to follow in subsequent meetings