Locks, TM, or DCS?

Carole-Jean Wu

Parallel Programming Model

- Traditional Synchronization Operation
  - Introduces complexity
- Transactional Memory (TM)
  - Requires non-local reasoning
- Data-Centric Synchronization (DCS)
  - Colorama

Challenges for Locking

- Fine-grained vs. Coarse-grained locking

- Complexity
  - Priority inversion
  - Convoying
  - Deadlocks
  - Conservative/Unnecessary Synchronization

Transactional Memory

- Programmers specify sequences of operations that should be executed atomically
  - Avoid traditional locking → simple
  - No worry for fine-tune critical section

Challenges for TM

- Concurrency
  - limited by dependencies
- Programmability
  - required programmers to reason codes non-locally
  - hard to define transaction boundaries

Colorama

Architectural Support for DCS

Data-Centric Synchronization (DCS)

- Use local reasoning to assign synchronization constraint
- Focus on which sets of data structures remain consistent with each other
  - Assign to the same critical section region

Data-Centric Synchronization (DCS)

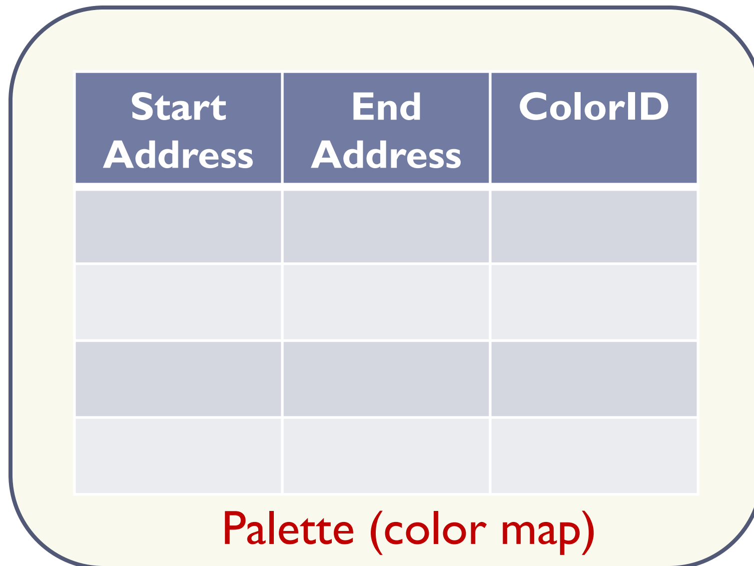
- System automatically infers critical section
  - programmers associates synchronization constraints with data structures when they are declared or allocated
- Then, insert synchronization operations

Hardware Data-Centric Synchronization

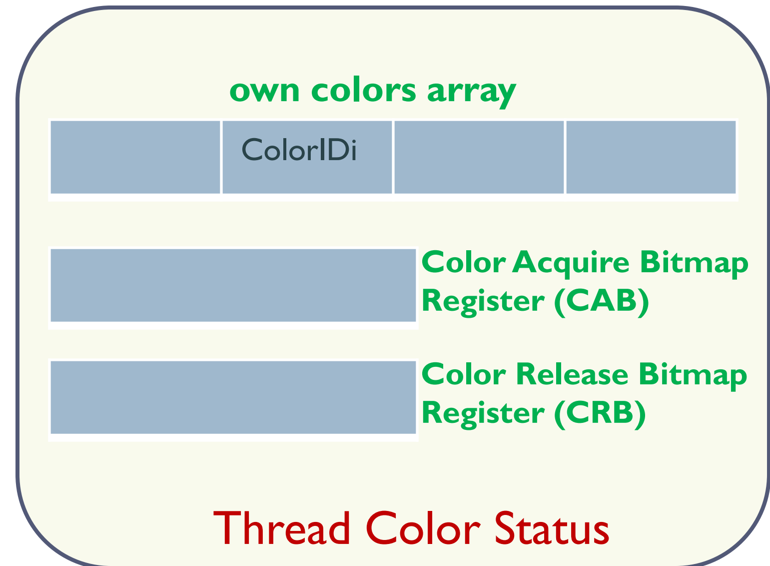
- Hardware primitive to specify "data consistency domain"
  - One that monitors all memory accesses to determine the start of a critical section
  - One that triggers the exit of a critical section

- All executions of critical sections of the same color by different threads are serializable

Colorama Architecture



Shared



Per Thread

Colorama Architecture (TM-based)

load
0xC0AD

Start Address	End Address	ColorID
0x0000	0x6876	RED
0xC000	0xFFFF	BLUE
0x9000	0x9999	PURPLE

Palette (color map)

Shared

own colors array

PURPLE	RED

Color Acquire Bitmap Register (CAB)

Color Release Bitmap Register (CRB)

Thread Color Status

Per Thread

Colorama Architecture (TM-based)

load
0xC0AD

Start Address	End Address	ColorID
0x0000	0x6876	RED
0xC000	0xFFFF	BLUE
0x9000	0x9999	PURPLE

Palette (color map)

Shared

own colors array

PURPLE	RED	

Color Acquire Bitmap Register (CAB)

Color Release Bitmap Register (CRB)

Thread Color Status

Per Thread

Colorama Architecture (TM-based)

load
0xC0AD

Start Address	End Address	ColorID
0x0000	0x6876	RED
0xC000	0xFFFF	BLUE
0x9000	0x9999	PURPLE

Palette (color map)

Shared

own colors array

PURPLE	RED	BLUE

Color Acquire Bitmap Register (CAB)

Color Release Bitmap Register (CRB)

Thread Color Status

Per Thread

Colorama Architecture (TM-based)

Transaction starts!!

Start Address	End Address	ColorID
0x0000	0x6876	RED
0xC000	0xFFFF	BLUE
0x9000	0x9999	PURPLE

Palette (color map)

Shared

own colors array

PURPLE	RED	BLUE

Color Acquire Bitmap Register (CAB)

Color Release Bitmap Register (CRB)

Thread Color Status

Per Thread

Colorama Architecture (TM-based)

Start Address	End Address	ColorID
0x0000	0x6876	RED
0xC000	0xFFFF	BLUE
0x9000	0x9999	PURPLE

Palette (color map)

Shared

own colors array

PURPLE	RED

Color Acquire Bitmap Register (CAB)

Color Release Bitmap Register (CRB)

Thread Color Status

Per Thread

Colorama Architecture (TM-based)

Start Address	End Address	ColorID
0x0000	0x6876	RED
0xC000	0xFFFF	BLUE
0x9000	0x9999	PURPLE

Palette (color map)

Shared

own colors array

PURPLE

Color Acquire Bitmap Register (CAB)

Color Release Bitmap Register (CRB)

Thread Color Status

Per Thread

Colorama Architecture (TM-based)

Transaction commits!!

Start Address	End Address	ColorID
0x0000	0x6876	RED
0xC000	0xFFFF	BLUE
0x9000	0x9999	PURPLE

Palette (color map)

Shared

own colors array

clear color bits	Color Acquire Bitmap Register (CAB)
clear color bits	Color Release Bitmap Register (CRB)

Thread Color Status

Per Thread

Colorama Architecture (Lock-based)

- Lock contention
  - Critical sections run until the end of subroutine

    → larger critical section size
- Deadlocks
  - Possible deadlock occurrence if nested TM is supported
  - Possible data race if failing to assign colors in exclusive manners

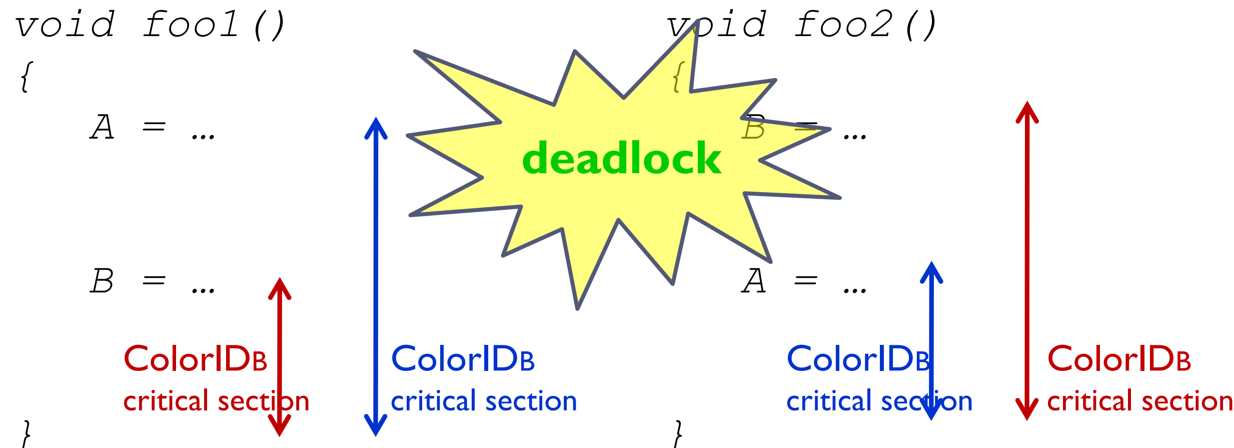
```
void foo1()
{
    A = ...


    B = ...


}
```

```
void foo2()
{
    B = ...


    A = ...


}
```

Colorama Architecture (Lock-based)

- Lock contention
  - Critical sections run until the end of subroutine

    → larger critical section size
- Deadlocks
  - Possible deadlock occurrence if nested TM is supported
  - Possible data race if failing to assign colors in exclusive manners



Drawbacks of HDCS

- Limited program knowledge
  - hard to identify accesses to data structures belonging to a domain
  - exit policies cause possible deadlocks for lock-based implementation
- Coloring structures for which the accesses do not need to be constrained (e.g. threads' private variable)
  - Limited concurrency

Comparison

Parallel Programming Models	Approach	Programmability	Critical Section Boundary	Issues
Traditional Synchronization	Code-Centric Synchronization	Non-local reasoning	Programmers insert lock acquire/release	Possible deadlocks
Transactional Memory	Code-Centric Synchronization	Non-local reasoning	Programmers insert transaction start/end	No deadlocks
Data-Centric Synchronization	Data-Centric Synchronization	Local reasoning	System automatically infers critical sections	Possible deadlocks