Practical Concurrency

Copyright 2007 SpringSource. Copying, publishing or distributing without express written permission is prohibited.

Agenda

- Motivation
- Java Memory Model Basics
- Common Bug Patterns
- JDK Concurrency Utilities
- Patterns of Concurrent Processing
- Testing Concurrent Applications
- Concurrency in Java 7

Copyright 2007 SpringSource. Copying, publishing or distributing without express written permission is prohibited.

Motivation

Copyright 2007 SpringSource. Copying, publishing or distributing without express written permission is prohibited.

Multicore

- Multi-core architecture changes everything
  - Parallel execution is more common
  - Memory model is more important

Copyright 2007 SpringSource. Copying, publishing or distributing without express written permission is prohibited.

Understand program execution

- Know what the VM is doing with your application
- Understand how best to express concurrent semantics

Copyright 2007 SpringSource. Copying, publishing or distributing without express written permission is prohibited.

Java Memory Model Basics

Copyright 2007 SpringSource. Copying, publishing or distributing without express written permission is prohibited.

Core Principles

- The JMM defines:
  - Ordering
  - Atomicity
  - Visibility

Copyright 2007 SpringSource. Copying, publishing or distributing without express written permission is prohibited.

Basic Guarantees

- Ordering
  - Within thread as-if-serial
- Atomicity
  - Read and writes to all fields except long and double
- Visibility
  - Within thread only

Copyright 2007 SpringSource. Copying, publishing or distributing without express written permission is prohibited.

Improving these guarantees

- Synchronized
  - Define atomic blocks
  - Constrain ordering
- Volatile
  - Control cross-thread visibility
  - Constrain ordering
  - Atomic read/write to long and double
- Final

Copyright 2007 SpringSource. Copying, publishing or distributing without express written permission is prohibited.

Understanding *happens-before*

- Program order rule
- Monitor lock rule
- Volatile variable rule
- Thread start rule
- Thread termination rule

Copyright 2007 SpringSource. Copying, publishing or distributing without express written permission is prohibited.

Mystery of Visibility

Copyright 2007 SpringSource. Copying, publishing or distributing without express written permission is prohibited.

Safe publication

- Final
- Volatile
- Locking

Copyright 2007 SpringSource. Copying, publishing or distributing without express written permission is prohibited.

Common Bug Patterns

Copyright 2007 SpringSource. Copying, publishing or distributing without express written permission is prohibited.

Common Bug Patterns

- Shared, non-volatile primitives
- One-sided synchronization
- Mixed synchronization
- Incorrect encapsulation

Copyright 2007 SpringSource. Copying, publishing or distributing without express written permission is prohibited.

Demo

Common Bug Patterns

Copyright 2007 SpringSource. Copying, publishing or distributing without express written permission is prohibited.

JDK Concurrency Utilities

Copyright 2007 SpringSource. Copying, publishing or distributing without express written permission is prohibited.

Building Blocks

- Atomics
- Latches
- Semaphores
- Locks and Conditions

Copyright 2007 SpringSource. Copying, publishing or distributing without express written permission is prohibited.

Atomics

- Common operations on common types handled atomically
- Support for manipulating references and primitives
- Supports updates of fields reflectively

Copyright 2007 SpringSource. Copying, publishing or distributing without express written permission is prohibited.

Semaphores

- Manage access to resources
- Provide some degree of ordering

Copyright 2007 SpringSource. Copying, publishing or distributing without express written permission is prohibited.

Latches

- Provide control across task execution
- Await a condition

Copyright 2007 SpringSource. Copying, publishing or distributing without express written permission is prohibited.

Locks and Conditions

- Lock interface as an alternative to synchronized
- Condition interface as an alternative to wait/notify
- Prefer inbuilt operations where possible
  - Use when read/write locking can offer a real benefit

Copyright 2007 SpringSource. Copying, publishing or distributing without express written permission is prohibited.

Demo

Building Blocks

Copyright 2007 SpringSource. Copying, publishing or distributing without express written permission is prohibited.

Concurrent Collections

- ConcurrentHashMap
- CopyOnWrite(List|Set)
- ConcurrentSkipList(Map|Set)
- Improvement over synchronized Collections
  - Permit interleaved read/write

Copyright 2007 SpringSource. Copying, publishing or distributing without express written permission is prohibited.

CopyOnWrite

- Ideal where read greatly outweighs write
  - Event listeners for example

Copyright 2007 SpringSource. Copying, publishing or distributing without express written permission is prohibited.

ConcurrentHashMap

- Default ConcurrentMap
  - Supports a configurable level of throughput
    - Ideally match to expected number of threads
  - Provides ‘weakly-consistent’ views
  - Iteration does not throw ConcurrentModificationException

Copyright 2007 SpringSource. Copying, publishing or distributing without express written permission is prohibited.

ConcurrentSkipList*

- Concurrent replacement for sorted Map and sorted Set
- Typically lower throughput that ConcurrentMap
- Some operations are not constant time
  - size() requires traversal of all elements

Copyright 2007 SpringSource. Copying, publishing or distributing without express written permission is prohibited.

Weak Consistency

- Iterators on ConcurrentHashMap and ConcurrentSkipListMap
- Reflect the state of the Map at or since the creation of the iterator
- CopyOnWrite gives **fully consistent** iterators

Copyright 2007 SpringSource. Copying, publishing or distributing without express written permission is prohibited.

ConcurrentMap interface

- putIfAbsent
  - Add an item if not present and return previously mapped value
- remove
  - Only if mapped to supplied value
- replace
  - Only if mapped to supplied value

Copyright 2007 SpringSource. Copying, publishing or distributing without express written permission is prohibited.

Why composite operations?

Thread A	Thread B
Object val = get("key");	
someProcess(val); // invalidates "key"	
	set("key", new Object()); // key in now valid
remove("key"); // incorrect!	

Copyright 2007 SpringSource. Copying, publishing or distributing without express written permission is prohibited.

Task Execution

- Abstraction around execution of concurrent tasks
- Based on a thread pool
- Highly configurable
- Supports
  - Delays
  - Prioritization
  - Concurrent result processing

Copyright 2007 SpringSource. Copying, publishing or distributing without express written permission is prohibited.

Demo

Task Execution

Copyright 2007 SpringSource. Copying, publishing or distributing without express written permission is prohibited.

Patterns of Concurrent Processing

Copyright 2007 SpringSource. Copying, publishing or distributing without express written permission is prohibited.

Concurrent Caches

- Use ConcurrentHashMap

Copyright 2007 SpringSource. Copying, publishing or distributing without express written permission is prohibited.

Updating Data Structures

- CopyOnWrite Collections for Lists/Sets
- Thread confinement for simple data structures

Copyright 2007 SpringSource. Copying, publishing or distributing without express written permission is prohibited.

Encapsulation

- Encapsulate access to shared data
- Avoid synchronized methods
- Keep locking internal

Copyright 2007 SpringSource. Copying, publishing or distributing without express written permission is prohibited.

Demo

Caches and Data Structures

Copyright 2007 SpringSource. Copying, publishing or distributing without express written permission is prohibited.

Handling Interruption

- Use interruption to stop task execution
- Never ignore an InterruptedExecution
  - Propagate the state
  - Or handle the interrupt

Copyright 2007 SpringSource. Copying, publishing or distributing without express written permission is prohibited.

Demo

Handling Interruption

Copyright 2007 SpringSource. Copying, publishing or distributing without express written permission is prohibited.

Client Locking Protocols

- Allowing clients to participate in your locking protocol
- Avoid where possible
  - Can lead to deadlock
- Expose composite operations or consider concurrent structures

Copyright 2007 SpringSource. Copying, publishing or distributing without express written permission is prohibited.

Alien Calls

- Calling methods outside your control whilst holding a lock
- Avoid at all costs
  - Easily leads to deadlock
  - Can block indefinitely

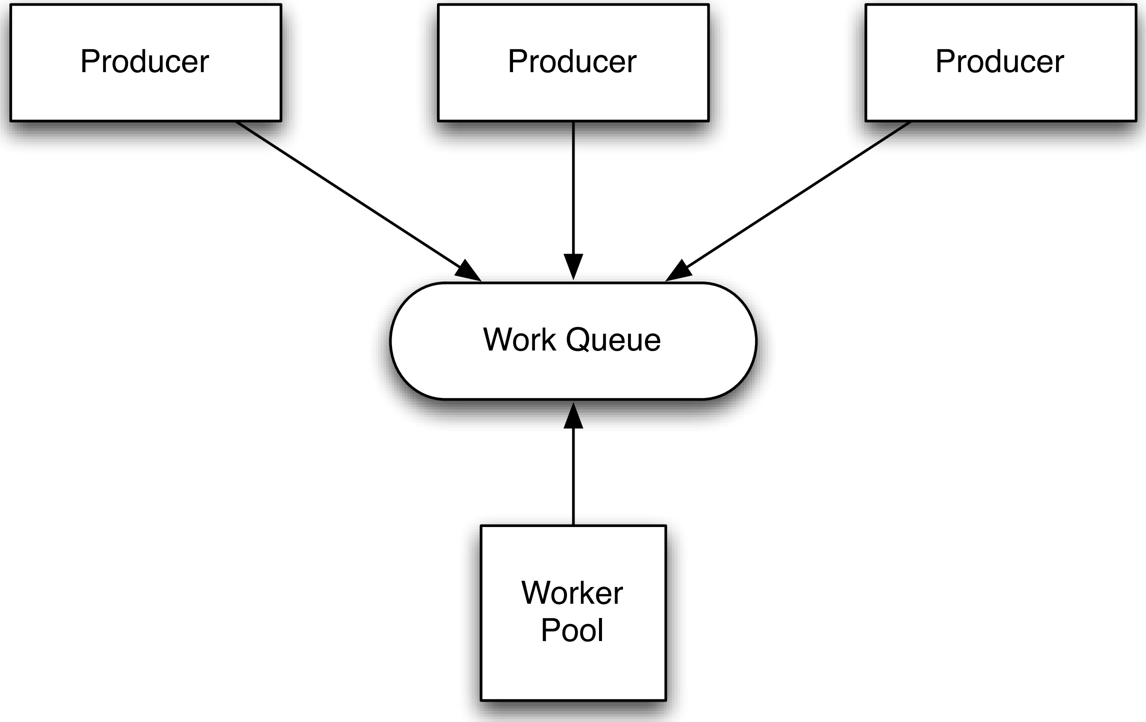
Copyright 2007 SpringSource. Copying, publishing or distributing without express written permission is prohibited.

Producer/Consumer

- Decouple the production of work items from their processing
- Build on ExecutorService and/or BlockingQueue
- Basic building block for a simple SEDA architecture

Copyright 2007 SpringSource. Copying, publishing or distributing without express written permission is prohibited.

Pattern



Copyright 2007 SpringSource. Copying, publishing or distributing without express written permission is prohibited.

Demo

Producer/Consumer

Copyright 2007 SpringSource. Copying, publishing or distributing without express written permission is prohibited.

Testing Concurrent Applications

Copyright 2007 SpringSource. Copying, publishing or distributing without express written permission is prohibited.

Testing

- MultithreadTC
  - http://www.cs.umd.edu/projects/PL/multithreadedtc/

Copyright 2007 SpringSource. Copying, publishing or distributing without express written permission is prohibited.

Demo

Testing

Copyright 2007 SpringSource. Copying, publishing or distributing without express written permission is prohibited.

Concurrency in Java 7

Copyright 2007 SpringSource. Copying, publishing or distributing without express written permission is prohibited.

Fork and Join

- Framework for parallel processing
- Recursively split processing into smaller chunks
- Details can be found at: http://g.oswego.edu/dl/concurrency-interest/

Copyright 2007 SpringSource. Copying, publishing or distributing without express written permission is prohibited.

Basic Pattern

```
Result solve(Problem problem) {
  if (problem is small)
    directly solve problem
  else {
    split problem into independent parts
    fork new subtasks to solve each part
    join all subtasks
    compose result from subresults
  }
}
```

Source: A Java Fork/Join Framework (Doug Lea)

Copyright 2007 SpringSource. Copying, publishing or distributing without express written permission is prohibited.

Summary

Copyright 2007 SpringSource. Copying, publishing or distributing without express written permission is prohibited.

Summary

- Concurrency is essential part of today’s applications
- Java has **excellent** supporting libraries
  - Use them
- Encapsulate your own shared data
- Keep an eye on Java 7

Copyright 2007 SpringSource. Copying, publishing or distributing without express written permission is prohibited.

Q&A

Copyright 2007 SpringSource. Copying, publishing or distributing without express written permission is prohibited.