Java Concurrency in Practice

Advanced Topics

Chapter-16

The Java Memory Model

Upcode Software
Engineer Team

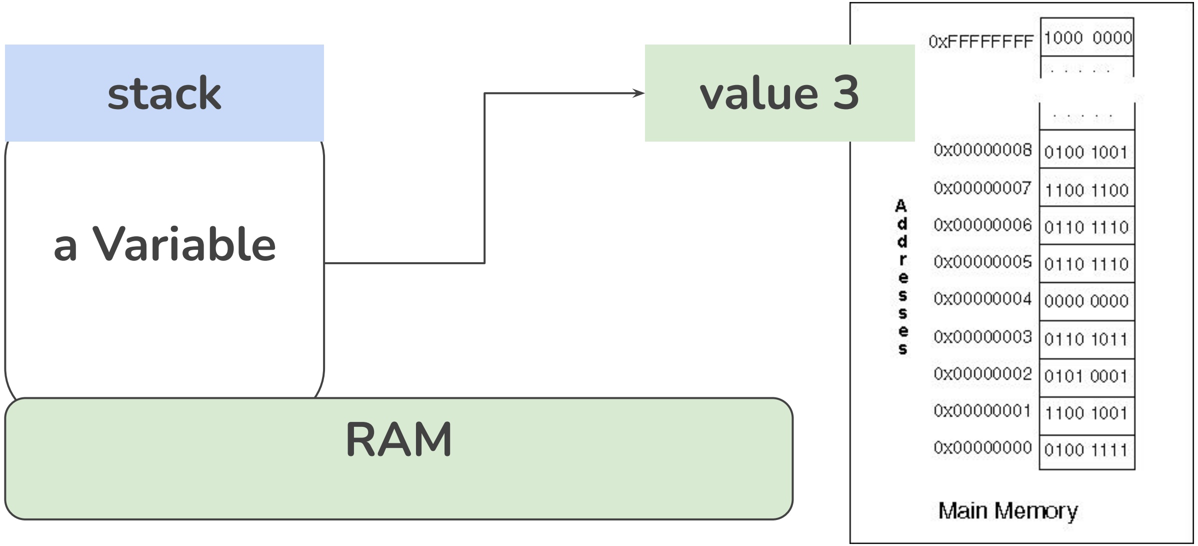
-2023-

CONTENT

1. What is memory model ?
2. Why would I want one ?
3. Safe publication
4. Solution
5. Reference

What is memory model (1/n)

A memory model addresses the question “Under what conditions does a thread that reads a Variable see the value 3?”

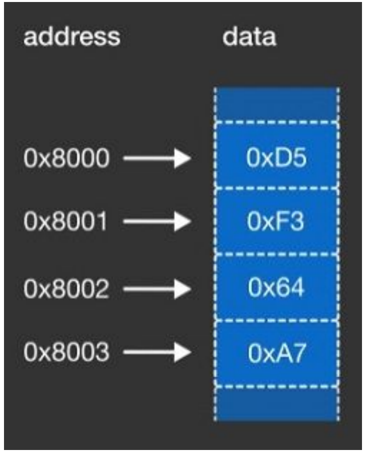


0xFFFFFFFF	1000 0000
	. . . . .
	. . . . .
0x00000008	0100 1001
0x00000007	1100 1100
0x00000006	0110 1110
0x00000005	0110 1110
0x00000004	0000 0000
0x00000003	0110 1011
0x00000002	0101 0001
0x00000001	1100 1001
0x00000000	0100 1111

Main Memory

What is memory model (2/n)

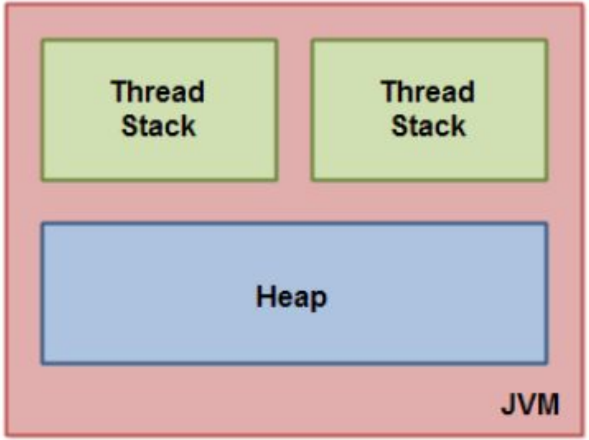
- **The main memory (or simply the memory)** is where variables and other information are stored while a program runs.
- From the perspective of a program, **the computer's memory is a collection of *bytes*, each with an integer *address*.** For example, there is a byte with address 1, another with address 2, etc., up to a very large number.
- A program can **fetch the current contents of the byte at a given memory address** and it can store a given value into that byte.



address	data
0x8000 →	0xD5
0x8001 →	0xF3
0x8002 →	0x64
0x8003 →	0xA7

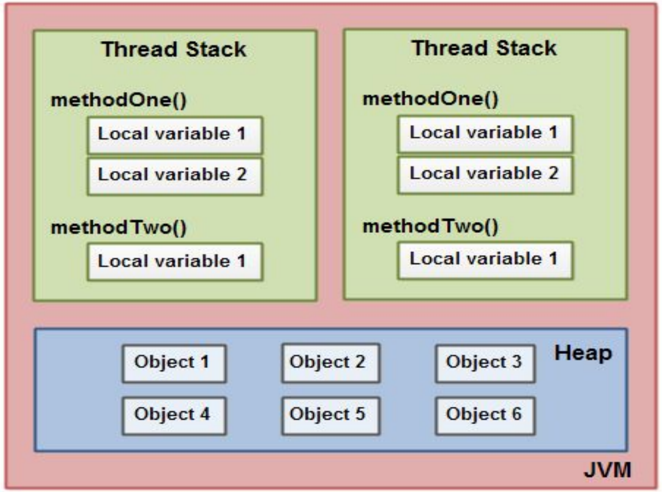
What is memory model (3/n)

- the JVM to maintain **within-thread as-if-serial semantics:** as long as the program has the same result.
- The JMM specifies the minimal guarantees **the JVM must make about when writes to variables become visible to other threads**. It was designed to **balance the need for predictability and ease of program development** with the realities of implementing high-performance **JVMs on a wide range of popular processor architectures.**

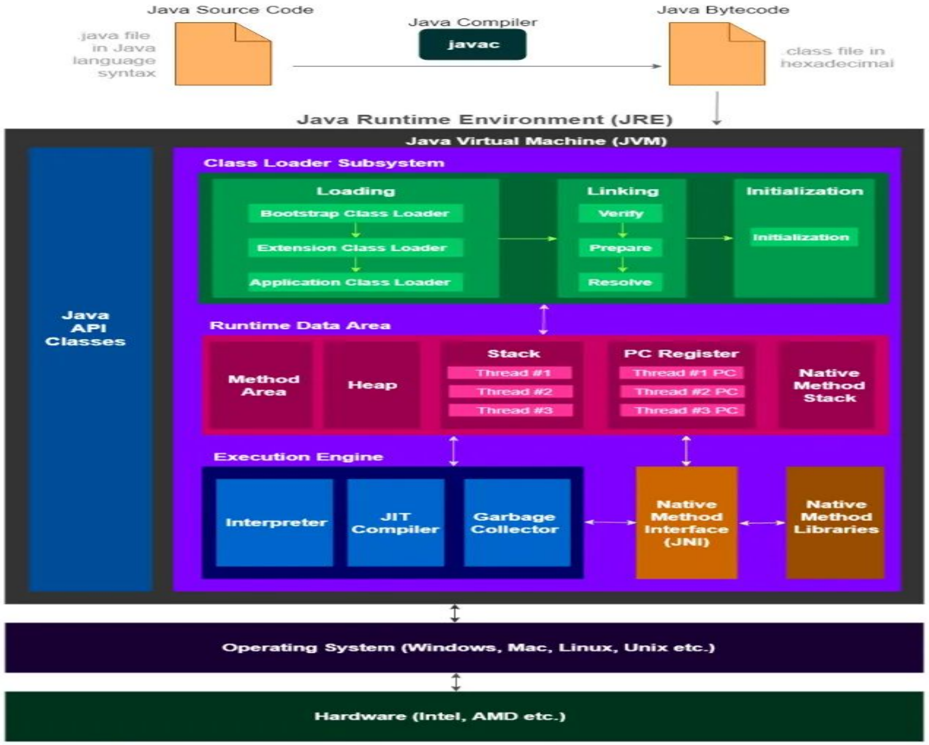


What is memory model (4/n)

- Some aspects of the JMM may be disturbing at first
- if you are not familiar with the tricks used by modern processors and compilers to squeeze extra performance out of your program.



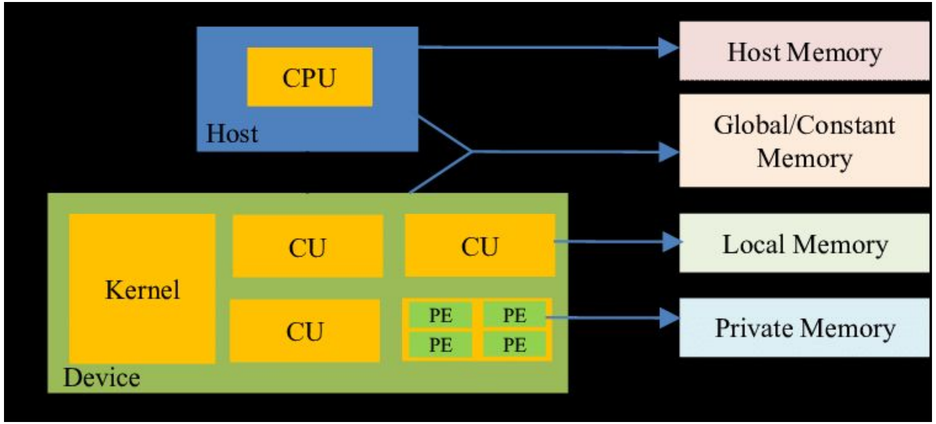
What is memory model (5/n)



JVM Architecture
(Image: PlatformEngineer.com)

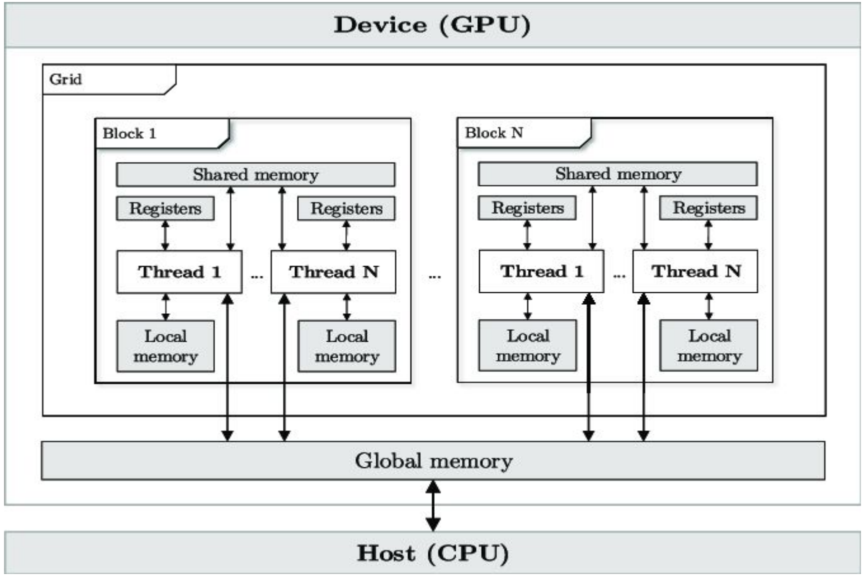
What is memory model (6/n)

- **The Java Memory Model (JMM) defines the allowable behavior of multithreaded programs**
- therefore describes when such **reorderings are possible**.
- It places **execution-time constraints on the relationship between threads and main memory in order to achieve consistent and reliable Java applications**.



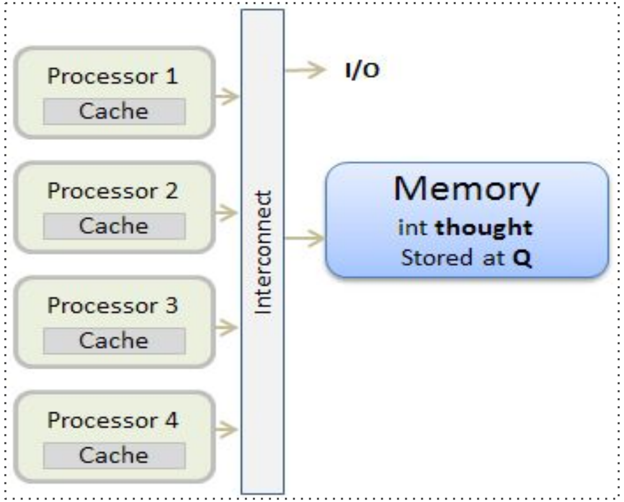
What is memory model (7/n) -Platform memory

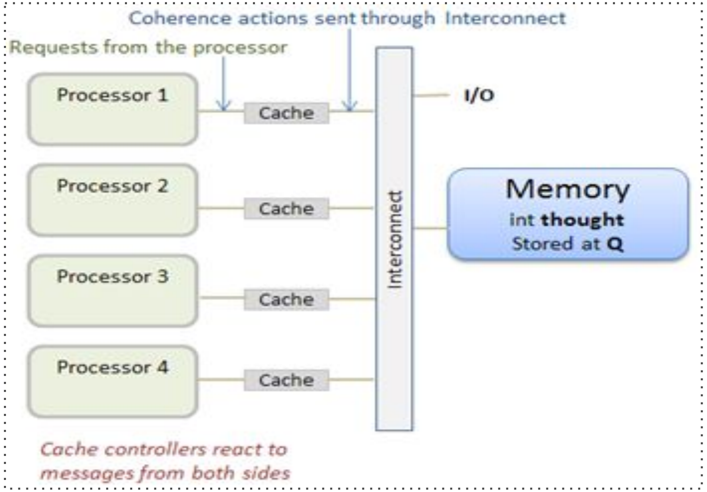
- You must have used some of the following **JVM memory configurations** when running resource-intensive Java programs.
- Processor architectures provide varying degrees of *cache coherence*; some provide minimal guarantees that allow different processors to see different values for **the same memory location** at virtually any time.



What is memory model (8/n) -Platform memory

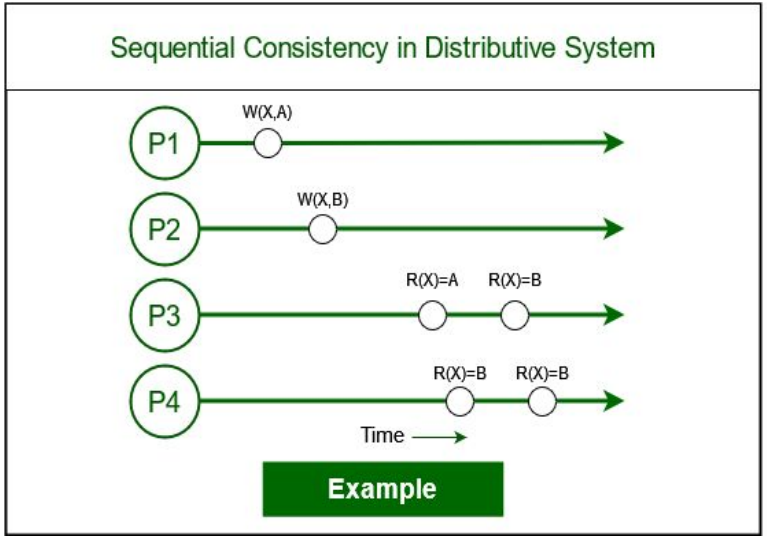
- Most of the time this information is not needed, so processors relax their **memory-coherency guarantees to improve performance.**
- In order to shield **the Java developer from the differences between memory models across architectures**, Java provides its own memory model, and the J**VM deals with the differences between the JMM and the underlying platform's memory model by inserting memory barriers at the appropriate places.**




Coherence actions sent through Interconnect
Requests from the processor
Processor 1
Cache
Processor 2
Cache
Processor 3
Cache
Processor 4
Cache
Interconnect
I/O
Memory
int **thought**
Stored at **Q**
Cache controllers react to messages from both sides

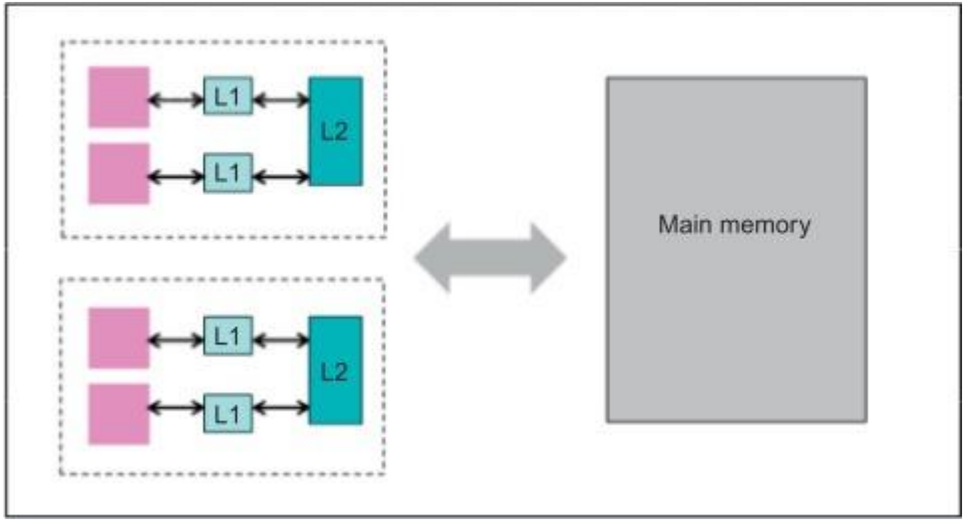
Sequential consistency

- Sequential consistency is **a conservative memory model that does not allow any instruction reordering on each core**.
- This prevents **many optimizations and degrades performance**. However, not all memory instructions on a single core need to preserve their program order



Sequential consistency

- **Sequential consistency** is a conservative memory model that does not allow any instruction reordering on each core.
- This prevents many optimizations and degrades performance.



The rules for happens-before

- **Program order rule**. Each action in a thread happens-before every action in that thread that comes later in the program order.
- **Monitor lock rule.** An unlock on a monitor lock happens-before every subsequent lock on that same monitor lock.
- **Volatile variable rule**. A write to a volatile field happens-before every subsequent read of that same field.
- **Thread start rule.** A call to Thread.start on a thread happens-before every action in the started thread.

The rules for happens-before

- **Thread termination rule.** Any action in a thread happens-before any other thread detects that thread has terminated, either by successfully return from Thread.join or by Thread.isAlive returning false.
- **Interruption rule.** A thread calling interrupt on another thread happens-before the interrupted thread detects the interrupt (either by having InterruptedException thrown, or invoking is Interrupted or interrupted).
- **Finalizer rule.** The end of a constructor for an object happens-before the start of the finalizer for that object.
- **Transitivity.** If A happens-before B, and B happens-before C, then A happens-before C

The rules for happens-before

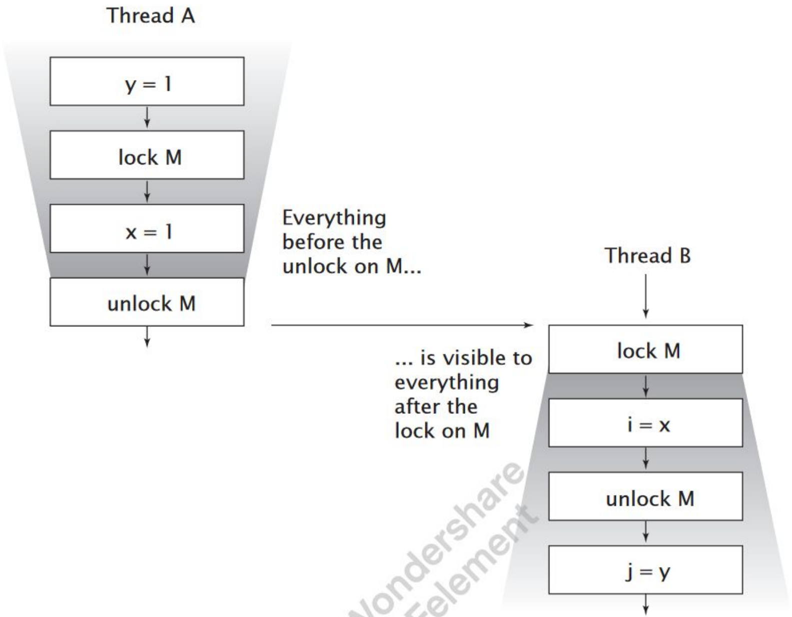

Thread A
y = 1
lock M
x = 1
unlock M
Everything before the unlock on M...
Thread B
lock M
... is visible to everything after the lock on M
i = x
unlock M
j = y

FIGURE 16.2. Illustration of *happens-before* in the Java Memory Model.

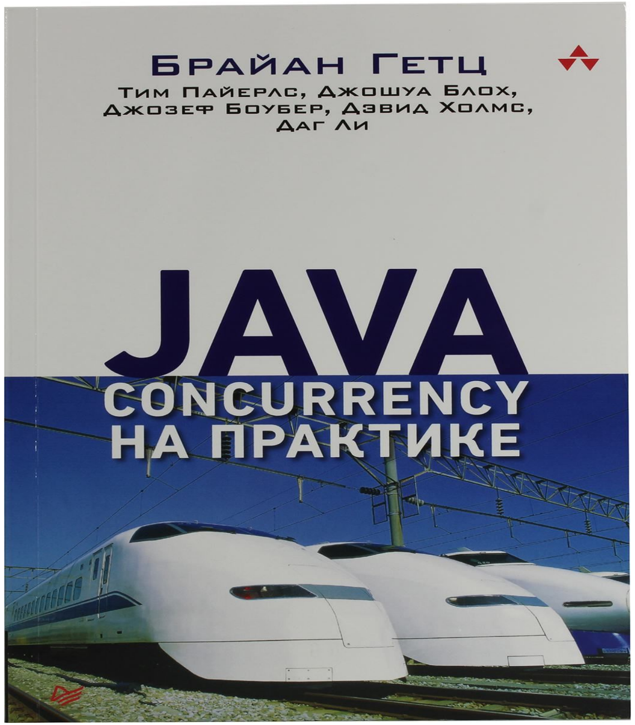
Wondershare PDFelement

Piggybacking on synchronization

Summary

- the most common reasons to use **threads** is to exploit **multiple processors**, in

Resources



Брайан Гетц

Тим Пайерлс, Джошуа Блох,
Джозеф Боубер, Дэвид Холмс,
Даг Ли

JAVA
CONCURRENCY
НА ПРАКТИКЕ

Reference

1. Java Concurrency book.
2. https://medium.com/platform-engineer/understanding-java-memory-model-1d0863f6d973
3. https://medium.com/@jojoooo/exploring-a-base-spring-boot-application-with-java-21-virtual-thread-spring-security-flyway-c0fde13c1eca
4. https://jenkov.com/tutorials/java-concurrency/java-memory-model.html

Thank you!

Presented by

Hamdamboy Urunov

(hamdamboy.urunov@gmail.com)