The Illusion Of Execution

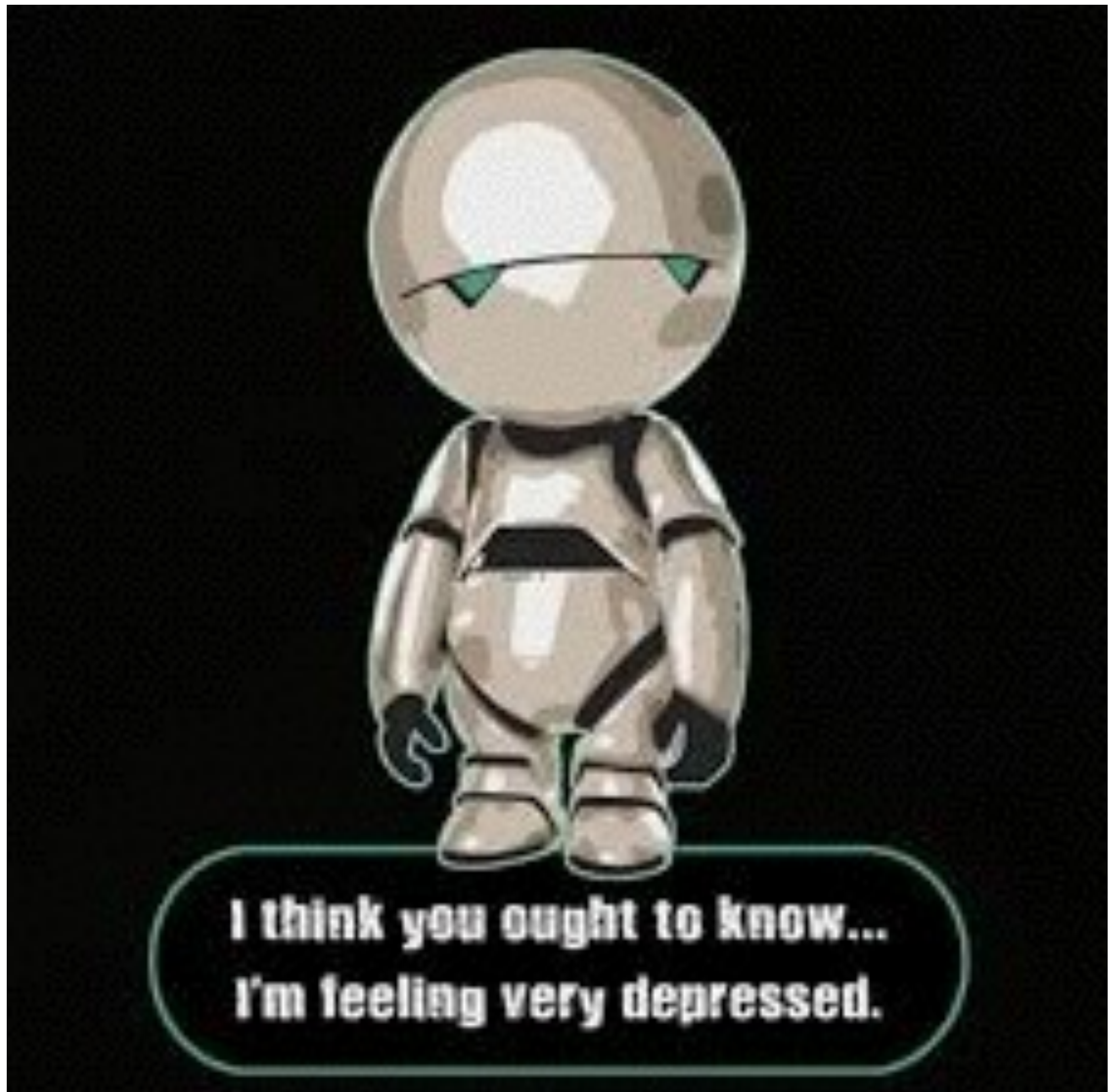
Nitsan Wakart (@nitsanw)
Lead Performance Engineer, Azul Systems
http://psy-lob-saw.blogspot.com

Thanks!

“Time is an illusion, and lunch time doubly so.”

– Ford Prefect, Hitchhiker’s Guide To The Galaxy

DON'T PANIC!!!



Write Once, Run?

- Developer writes Java code
- Compile/Pack/Deploy
- … MAGIC! …
- JVM executes the code on some hardware?

Memory is infinite!

```
SocketChannel accepted = serverSocket.accept();
Connection c = new Connection(accepted, messageSize);
```

Computation is Infinite!

```
SocketChannel accepted = serverSocket.accept();
Connection c = new Connection(accepted, messageSize);
new Thread(c, "Echo " + accepted.getRemoteAddress().toString()).start();
```

"When I program, I program like a god, as if resources are infinite!"

–Some Idiot I once worked with

The Way Down

- Your code…
- Is run by a JVM…
- Which is a process of an OS…
- Which is running on some hardware…

Hardware is FINITE!

Hardware Illusions

Elastic Computing Power

- How many cores?
  - Physical/Logical?
  - Hyper Threading on/off (BIOS settings)
- What is the frequency of a given core?
  - C/P-State (`intel_idle.max_cstate=0, cpufreq`)
  - Turbo Boost
  - Temperature

Hierarchical Memory

- How long does it take to read a value?
  - Registers
  - L1/L2/…/LLC
  - NUMA
- How can the hardware help?
  - Prefetch
  - Branch prediction

So What?

- Know what you measure
- Consider configuration options that reflect your requirements (disable C states/fix frequency)
- Performance counters(perf/likwid…):
  - cache misses
  - instruction counts

OS Illusions

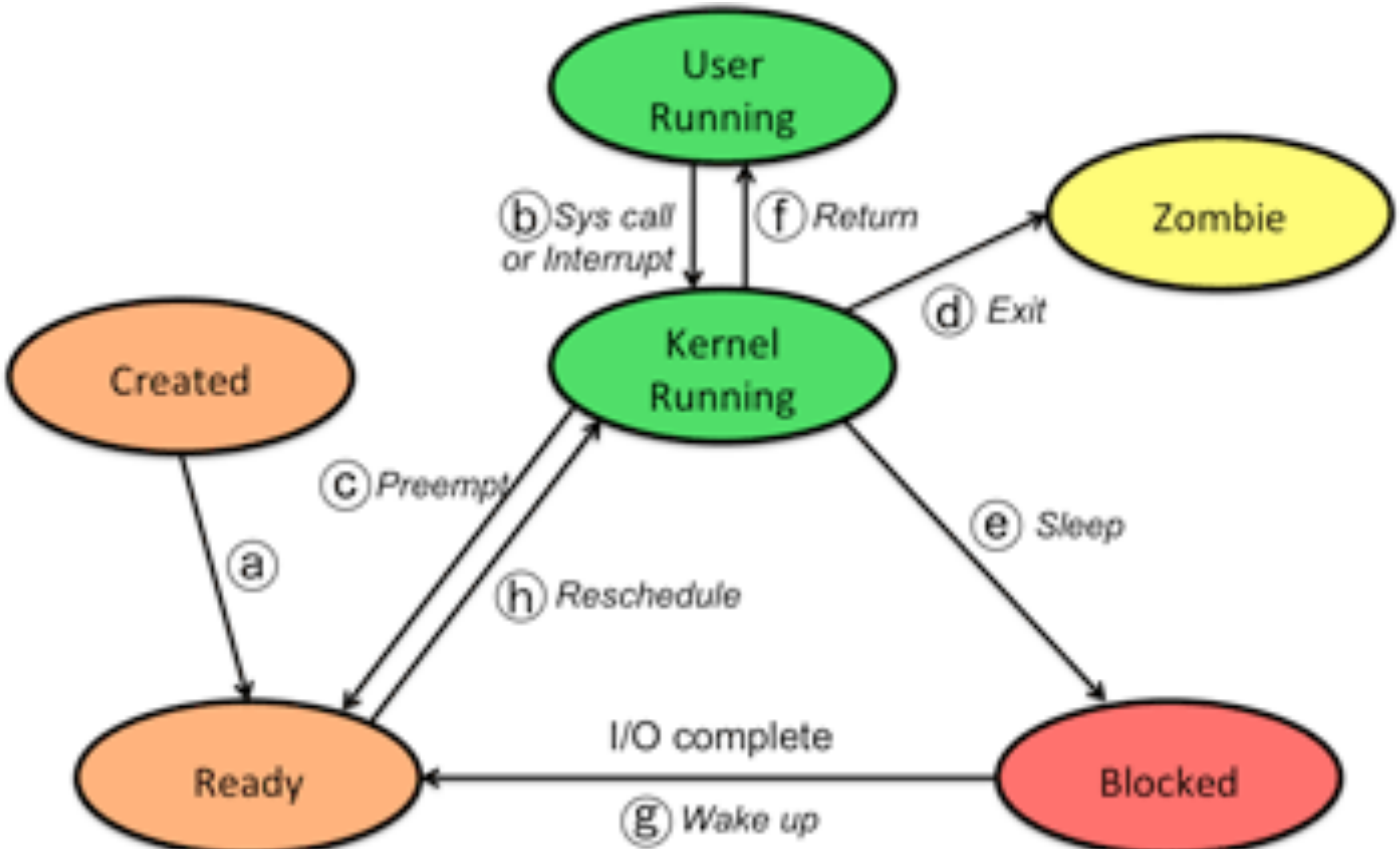
Multi-Tasking

- More processes than cores
  - Not an issue…
  - … unless they all want to run at the same time
- Scheduling and interrupts
  - Fairness
  - Context Switching
  - **Next instruction is arbitrarily delayed**

Virtual Memory

- How much memory can a process use?
    - Resident vs Virtual Memory
    - Swap
- When is memory disk?
    - Page faults
- When is disk memory?
    - Memory mapped files
    - Page cache

Thread States - Linux



From: http://www.cs.rutgers.edu/~pxk/416/notes/

So What?

- Avoid saturation
  - More executing threads than cores (long run queue)
  - More memory used than available (page faults, swap)
  - Set swappiness=0, buy more memory
- Consider controlling resource allocation
  - Use taskset/numactl/isocpus to reduce contention

JVM Process Illusions

The JVM Process: Threads

- How many threads for an application?
  - Application Java threads (Main, Thread etc)
  - Application native threads (native lib)
  - JVM Threads (GC, Compiler, JMX, RMI…)

Threads Example

(Oracle JDK8u20, on i5/dual core laptop, no args)

- Application threads
- 4 GC Threads (ParallelGC)
- 3 Compiler threads (1 C1 + 2 C2)
- … and others (Reference Handler/Finalizer/JVM Service…)
- 15 threads reported by jstack, 19 by OS

The JVM Process: Memory

- How much memory used if -Xmx1g?
  - Heap
  - Stack
  - JVM
  - Unmanaged

ZST - Zing System Tools

- Memory management module for fast page mapping
- Java memory 'taken' upfront
- Blurring the JVM/OS line

So What?

- Avoid swap! configure for existing resources!
- Consider JVM threads/memory in estimates
- Configure GC/Compiler thread counts
- Monitor full process memory (not just heap)

Java Runtime Illusions

Code & JVM: Symbiosis

- Memory is managed!
  - Reference accounting
- Hotspot compilation!
  - Code mutation
- Managed Execution!
  - The occasional pause…

Java Hidden Symbols

```
void copyPoint(Point p1, Point p2) {
    p1.x = p2.x;
}

void copyPoint(oop p1, oop p2) {
    address a1 = readBarrier(p1);
    address a2 = readBarrier(p2);
    oop x = getObject(a2+xFieldOffset);
    putObject(a1+xFieldOffset, x);
    writeBarrier(a1, x);
    safepoint_poll();
}
```

Safepoint

(noun.)

- A thread state
  - `Waiting/Idle/Blocked -> @Safepoint`
  - `Running Java code -> !@Safepoint`
  - `Running native code -> @Safepoint`

http://blog.ragozin.info/2012/10/safepoints-in-hotspot-jvm.html
http://psy-lob-saw.blogspot.com/2014/03/where-is-my-safepoint.html
http://chriskirk.blogspot.ru/2013/09/what-is-java-safepoint.html

When at a safepoint…

- Heap is not accessed
  - GC time?
- Java code is not executed
  - Code change time?

Global Safepoint

- All threads are @Safepoint -> no Java code is running
- JVMs use it for:
  - Some GC phases
  - Deoptimization
  - Stack trace dump
  - Lock un-biasing
  - Class redefinition

You could cause a Safepoint…

- On normal allocation (Young Gen exhausted)
- On large object allocation (Old Gen exhausted)
- On synchronized block (unbiasing)
- Profiler sampling
- Hitting cold code

TTSP - Time To Safepoint

- To bring JVM to global safepoint:
  - Raise Safepoint ‘flag’
  - Wait for ALL threads to reach Safepoint and stop
- Not included in GC Time
  - -XX:+PrintGCApplicationStoppedTime

Where would sir like his Safepoint?

- Safepoint poll inserted at:
    - While loop back edge
    - Method exit

Safepoint poll implementation: OpenJDK

- Read from a special page:

```
test    DWORD PTR [rip+0xfffffffffe690e53],eax
```

- JVM Sets the page to protected, polling threads trap a SEGV and go to safepoint

- Look for `{poll}` or `{poll_return}` in the assembly comments

Safepoint poll implementation: Zing

- Read the thread local safe point flag:

```
gs:cmp4i [0x40 tls._please_self_suspend],0

jnz 0x500a0186; Where the safepoint code be
```

- JVM Sets the thread flag to 1, polling threads hop to

- Look for **`tls._please_self_suspend`**

How far to the nearest Safepoint?

- Inlining removes end of method safepoints
- Safepoints can be delayed by:
  - Long counted loops
  - Large memory copies (System.arrayCopy/Unsafe.copyMemory)
  - Interrupted threads
  - Page Faults

So what?

- High TTSP -> Long STW Pauses
- Global SP ‘Cost’ = threads * TTSP
- Global TTSP = MAX(Thread TTSP)
- Mind the gap:
  - Mapped files write/read
  - Big memory copy operations
  - Very large counted loops

What’s an OOP?

- Ordinary Object Pointer
- Java: Object reference -> JVM: OOP
- Pointers to managed data on the heap

Memory Barrier

(not the JMM kind)

"…a block [of code] on reading from or writing to certain memory locations by certain threads or processes."

Memory Management Reference:
http://www.memorymanagement.org/glossary/b.html#term-barrier-1

Compressed OOPs

-XX:+UseCompressedOops

- Want large heaps (> 4G)
- Want 32bit OOPs
- Objects aligned to -XX:ObjectAlignmentInBytes=A (default is 8), K power of 2 (8 -> K=3)
- Can compress OOP by dropping last K bits (>>K)
- Must decompress address to use it (<<K)
- Can use heap base to extend referable range (BASE + OOP<<K)
- Max referenced heap size is now 4G * A

https://wikis.oracle.com/display/HotSpotInternals/CompressedOops

Compressed Oops Example(x86):

JAVA:

```
long v = this.l.longValue();
```

-XX:-UseCompressedOops:

```
mov r10, QWORD PTR [rsi+0x18]          ; r10= this.l
mov r10, QWORD PTR [r10+0x10]          ; r10= l.value
```

-XX:+UseCompressedOops:

```
mov r11d, DWORD PTR [rsi+0x10]         ; r11d= this.l
mov r10,  QWORD PTR [r12+r11*8+0x10]; r10= l.value
```

CompressedOops is a Read Barrier

(arguably)

- Must be decompressed before read ‘through’
- Can be copied without decompression
- Can be compared without decompression

LVB - Zing Read Barrier

- Will not fit in this talk, but…
- Looks like this

```
test8 rax,[gc_phase_trap_mask]; GC phase changed?

jnz 0x500d639b; GOTO LVB cold path
```

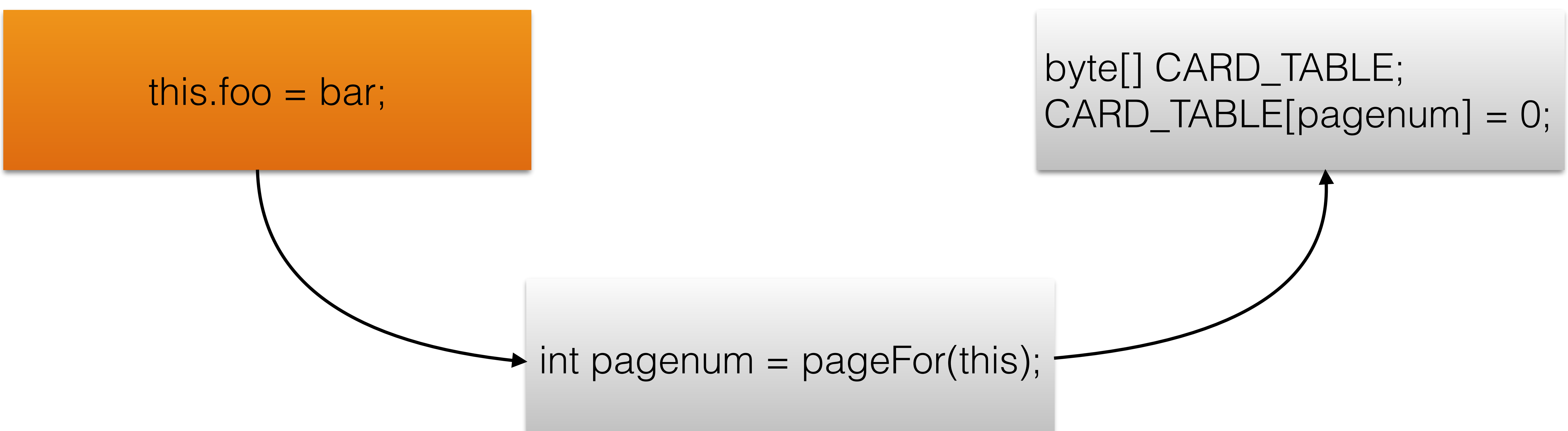
- Cold path: value has relocated
  - Mutator will fix up the loaded value
  - ‘Self healing’ - mutator participates in relocation

http://www.azulsystems.com/sites/default/files/images/c4_paper_acm.pdf

http://www.javaworld.com/article/2078661

Card Marking

"The JVM maintains a card map, with one bit (or byte, in some implementations) corresponding to each card in the heap. Each time a pointer field in an object in the heap is modified, the corresponding bit in the card map for that card is set."



Card Marking is a Write Barrier

- An optimisation for young collections
- Reduce the impact of OldGen size on scan time
- Introduce a small overhead
- Introduce false sharing? (-XX:+UseCondCardMark)
- Comes in different flavours!

CardMarking v1 (default)

```
; rsi is 'this' address
; rdx is setter param, reference to bar
; this.foo = bar
mov    QWORD PTR [rsi+0x20],rdx
; r10 = rsi = this
mov    r10,rsi
; r10 = r10 >> 9;
shr    r10,0x9
; r11 is base of card table, byte[] CARD_TABLE
mov    r11,0x7ebdfcff7f00
; Mark 'this' card as dirty
; CARD_TABLE[this address >> 9] = 0
mov    BYTE PTR [r11+r10*1],0x0
```

CardMarking v1 (default)

```
this.bar = foo;
CARD_TABLE[addressOf(this) >> 9] = 0;
```

CardMarking v2 (-XX:+UseCondCardMark)

```
; rsi is 'this' address
; rdx is setter param, reference to bar
; r10 = this
mov    r10,rsi
; r10 = r10 >> 9
shr    r10,0x9
; r11 = CARD_TABLE
mov    r11,0x7f7cb98f7000
; r11 = CARD_TABLE + (this >> 9)
add    r11,r10
; r8d = CARD_TABLE[this >> 9]
movsx  r8d,BYTE PTR [r11]
test   r8d,r8d
; if(CARD_TABLE[this >> 9] == 0) goto 0x00007fc4a1071d7d
je     0x00007fc4a1071d7d
; CARD_TABLE[this >> 9] = 0
mov    BYTE PTR [r11],0x0
0x00007fc4a1071d7d:
mov    QWORD PTR [rsi+0x20],rdx  ; this.foo = bar
```

CardMarking v2 (-XX:+UseCondCardMark)

```
if (CARD_TABLE[addressOf(this) >> 9] != 1){
  CARD_TABLE[addressOf(this) >> 9] = 0;
}
this.bar = foo;
```

CardMarking v3 (-XX:+UseG1GC)

```
 movsx  edi,BYTE PTR [r15+0x2d0] ; read GC flag
  cmp    edi,0x0; if (flag != 0)
  jne    0x00000001066fc601; GOTO OldValBarrier
Label WRITE:
  mov    QWORD PTR [rsi+0x20],rdx; this.foo = bar
  mov    rdi,rsi; rdi = this
  xor    rdi,rdx; rdi = this XOR bar
  shr    rdi,0x14; rdi = (this XOR bar) >> 20
  cmp    rdi,0x0; If this and bar are not same gen
  jne    0x00000001066fc616; GOTO NewValBarrier
Label EXIT:
;…
Label OldValBarrier:
  mov    rdi,QWORD PTR [rsi+0x20]
  cmp    rdi,0x0; if(this.foo == null)
  je     0x00000001066fc5dd; GOTO WRITE
  mov    QWORD PTR [rsp],rdi ; setup rdi as parameter
  call   0x000000010664bca0  ;   {runtime_call}
  jmp    0x00000001066fc5dd; GOTO WRITE
Label NewValBarrier:
  cmp    rdx,0x0; bar == null
  je     0x00000001066fc5f5 goto Exit
  mov    QWORD PTR [rsp],rsi
  call   0x000000010664bda0  ;   {runtime_call}
  jmp    0x00000001066fc5f5  ; GOTO exit;
```

CardMarking v3 (-XX:+UseG1GC)

```
oop oldFooVal = this.foo;
if (GC.isMarking != 0 && oldFooVal != null){
  g1_wb_pre(oldFooVal);
}
this.foo = bar;
if ((this ^ bar) >> 20) != 0 && bar != null) {
  g1_wb_post(this);
}
```

So What?

- References mean extra work (but usually not much)
- Impact can change by option/GC/JVM
- ‘Normalized’ data structures can help
    - Inheritance vs. Composition
- Value Types might help(Java 9)

```
while(Q){
  A();
}
return;
```