

JavaOne™

Sun's 2004 Worldwide Java Developer Conference™

Garbage Collection in the Java HotSpot™ Virtual Machine

Choices and Trade-Offs

Peter Kessler

Jon Masamitsu

John Coomes

Sun Microsystems, Inc.

<http://www.sun.com>

java.sun.com/javaone/sf



Improve Java™ Platform Performance

Choosing a garbage collector

Learn about different garbage collection choices in the Java HotSpot™ virtual machine and the consequences of those choices

Have a little fun

Agenda

Welcome to “GC Today”

Introductions

What’s New in GC

Parallel Collector

Concurrent Collector

Ergonomics

Choosing a Collector

Futures

Questions From “Our Listeners”

Where Is GC Today?

Choice of collectors

- Serial collector
 - Copying young generation
 - Mark/Sweep/Compact old generation
- Incremental (train) collector
 - Incremental old generation
- Parallel collector
 - Parallel copying young generation
- Mostly concurrent collector
 - Parallel copying young generation
 - Concurrent old generation

Introductions

Today's guest speakers

- John Coomes
 - Concurrent garbage collection
- Jon Masamitsu
 - Ergonomics, parallel garbage collection

What's Coming in GC?

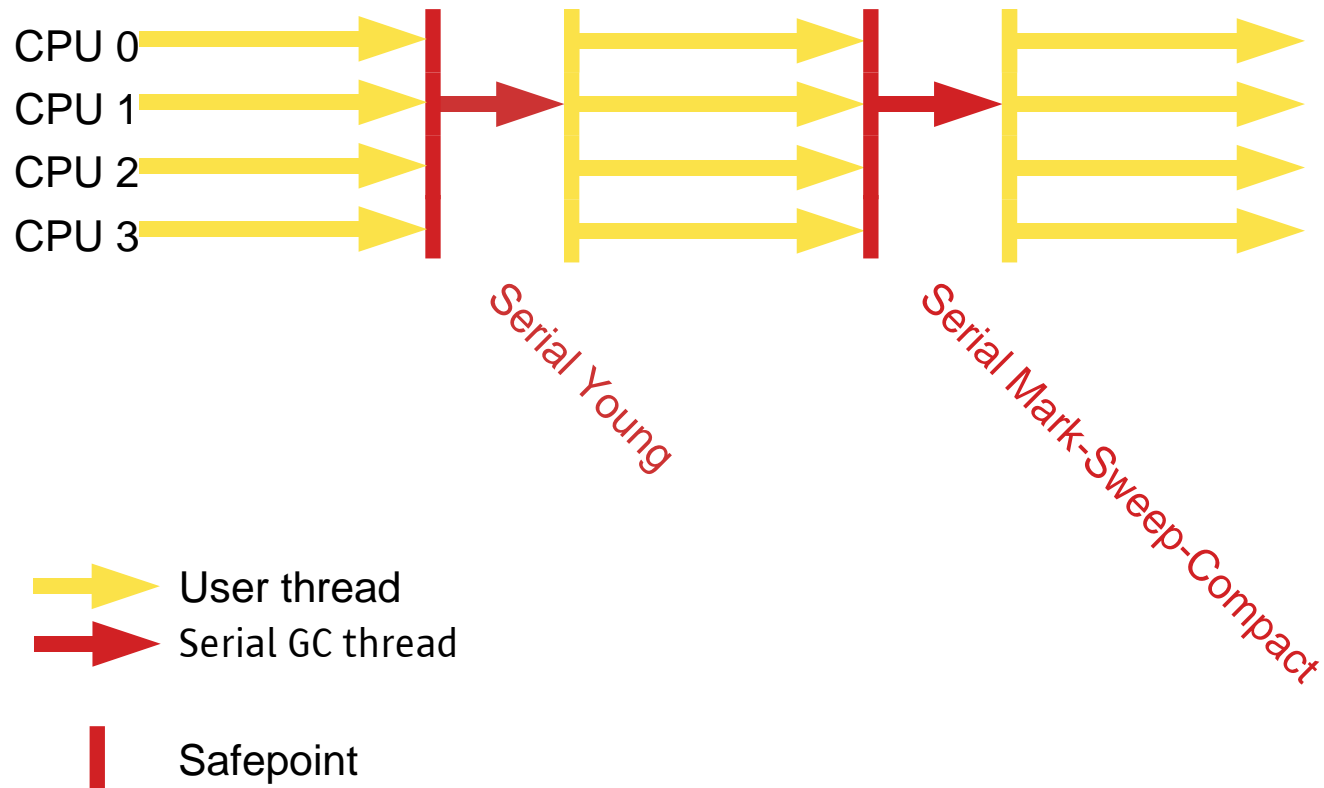
- Parallel collector
 - Behavior-based tuning
 - Dynamic generation sizing
- Mostly concurrent collector
 - More parallelism
 - Dynamic scheduling
 - Promotion failure handling
- Incremental collector deprecated
 - –Xincgc now selects concurrent collector
- Thread local allocation buffers
 - Resizing based on usage

Parallel Collector Updates

Specify desired behavior

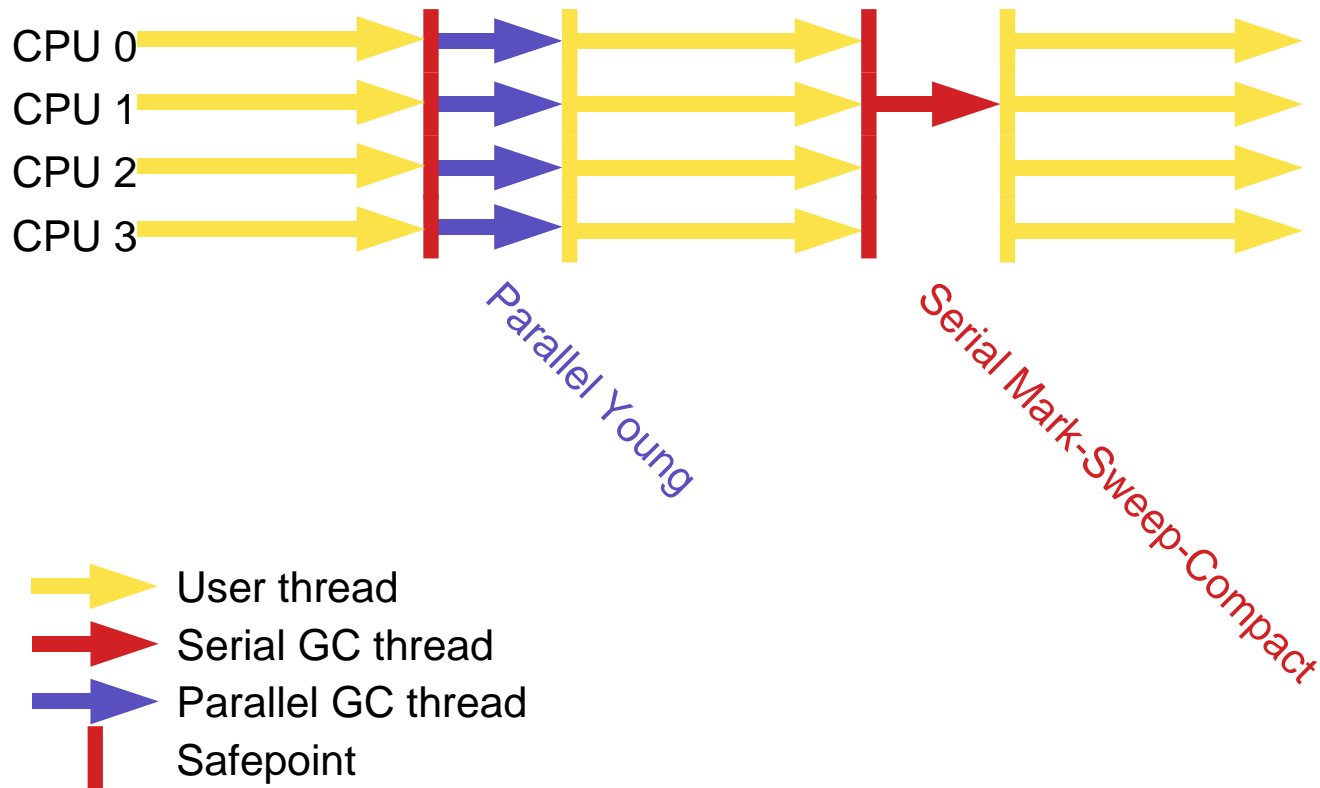
- Pause time goal
 - Desired maximum pause
- Throughput goal
 - Proportion of time in GC
- Minimize footprint
 - When other goals are satisfied
- GC time limit
 - What is “out of memory”?

Serial Collector



Parallel Collector

“Throughput Matters”



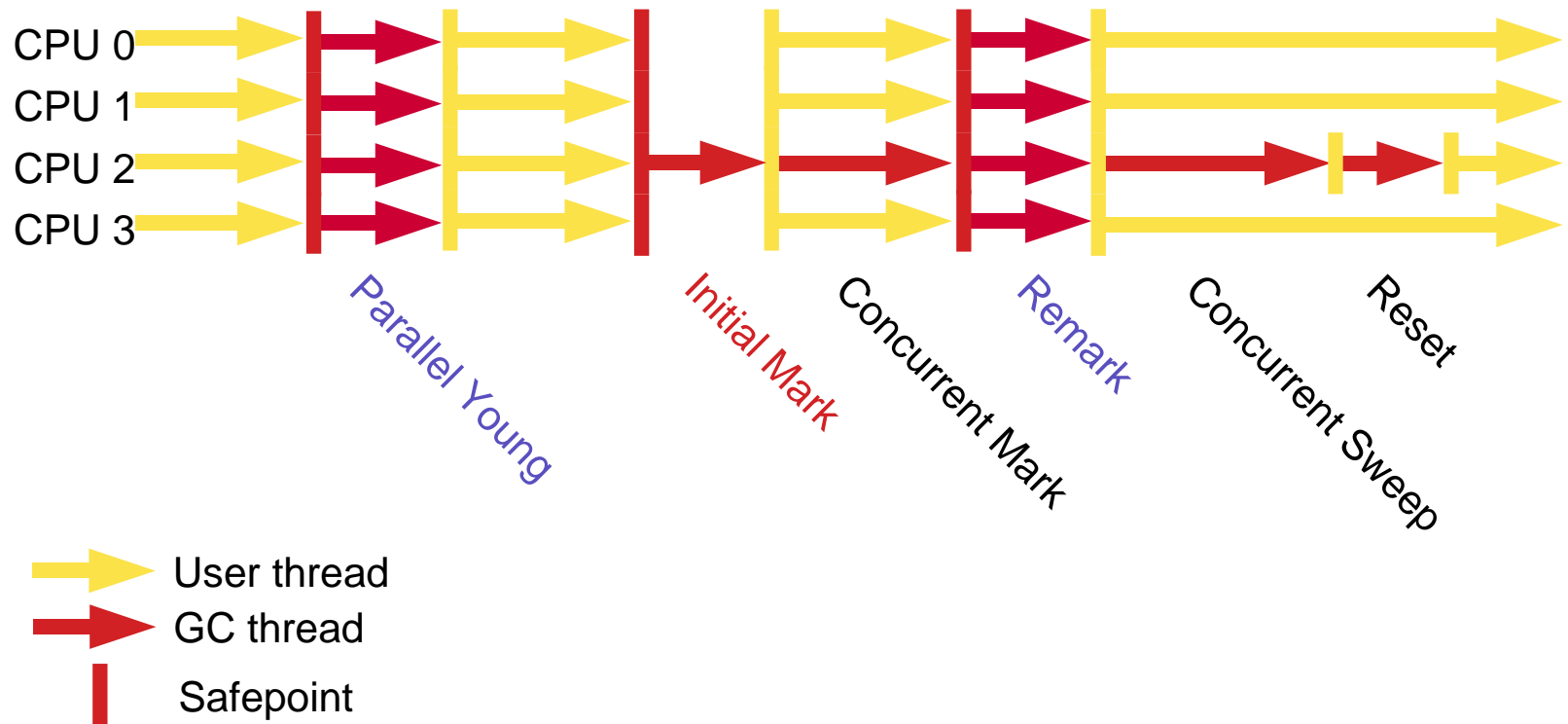
Parallel Collector Updates

Specify desired behavior

- Pause time goal
 - Desired maximum pause
- Throughput goal
 - Proportion of time in GC
- Minimize footprint
 - When other goals are satisfied
- GC time limit
 - What is “out of memory”?

Mostly Concurrent Collector

“Latency Matters”



Mostly Concurrent Collector Updates

- More parallelism during pauses
 - Young generation scanning
 - Reference object processing (soft, weak, etc.)
- Dynamic scheduling
 - Start of old generation collection
 - Pauses during old generation collection
- Promotion failure handling
 - Better use of heap space
 - Fewer full collections

Agenda

Welcome to “GC Today”

Introductions

What’s New in GC

Parallel Collector

Concurrent Collector

Ergonomics

Choosing a Collector

Futures

Questions From “Our Listeners”

Parallel Collector

“Throughput Matters”

- Maximize application time
 - Reduce time in garbage collection
- No pause time requirements
 - Or generous pause bounds
- Examples:
 - Back-office processing
 - Scientific computing

Point: Collector for Throughput

Parallel collector

- Parallel young generation collection
 - Efficient
 - Scalable
- Tuned for large servers
 - 2 CPUs to dozens of CPUs
- Faster than serial collector on 2 or more CPUs
- Serial old generation collection

Point: Parallel Collector Throughput Goal

Automatic sizing

- Throughput goal
- Measure throughput
 - Time spent in GC
 - Time spent outside of GC
- Grow generation sizes to meet goal
 - Larger generations mean more time between GCs
 - Both generations grow
- Adapt to changing application behavior

Counterpoint: Parallel Collector Throughput Goal

Automatic sizing

- Why not use maximum heap settings?
 - Xmx and –Xms set to the same high value
- What about rapid changes in application behavior?
- What about compilation time?

Point: Parallel Collector Pause Goal

Automatic sizing

- Maximum GC pause time goal
 - Best effort
 - Average + variance
- Measure young and old generation pauses
 - Measured separately
- Shrink generation size to meet the goal
 - A smaller generation usually collected more quickly
 - Not always possible to shrink the heap

Counterpoint: Parallel Collector Pause Goal

Automatic sizing

- Can the goal always be met?
 - Performance can suffer
- What if only one generation pause is too long?
 - Goal is not being met
- What if both generation pauses are too long?
 - One generation shrinks at a time

Summary on Parallel Collector

“Throughput Matters”

- Scales to higher numbers of CPUs
- Automatic sizing
- Occasional long pauses
 - Not designed for latency constraints

Agenda

Welcome to “GC Today”

Introductions

What’s New in GC

Parallel Collector

Concurrent Collector

Ergonomics

Choosing a Collector

Futures

Questions From “Our Listeners”

Concurrent Collector

“Latency Matters”

- Latency affected by pause times
 - “Typical” GC stops all user threads to do work
 - Response time suffers
- Areas where pauses matter
 - User interaction
 - Systems with work queues
 - Time-outs cause extra work or retries
- Examples:
 - Web servers/application servers
 - Telecommunications switching
 - Integrated development environments

Point: Low Pause Times

Concurrent collector

- Young generation collections
 - Parallelism reduces pause times
- Old generation collections
 - Most work done concurrently
 - Application continues to run
 - One CPU used for GC work
 - Two short pauses
 - Parallelism reduces pause times
- Pauses more uniform

Counterpoint: Low Pause Times

Concurrent collector

- Are spare CPUs needed?
 - GC takes processing power from application
 - Performance on 1-CPU platform
- Is a larger heap required?
 - Fragmentation
 - Does not compact the old generation
 - “Floating garbage”

Point: Scheduled Pauses

Concurrent collector

- Young generation collection
 - Application stopped for entire collection
- Old generation collection
 - Application stopped briefly
 - “Initial mark” and “remark”
- Young and old generation pauses independent
 - Cannot occur simultaneously
 - Otherwise, no restriction

Point: Scheduled Pauses

Concurrent collector

- Problem:
 - Young generation pause occurs
 - Immediately followed by old generation pause
 - Application sees one long pause
- Solution: schedule old generation pauses
 - Mid-way between young generation collections

Counterpoint: Scheduled Pauses

Concurrent collector

- Does it reduce total pause time?
- What about rapid changes in allocation rate?
- What is the cost of scheduling the pauses?

Point: Concurrent Collection Start

Concurrent collector

- Dynamically starts a concurrent collection
 - Gathers statistics
 - Collection duration
 - How fast old generation is being filled
 - Starts old generation collection “just in time”
- Adjusts as application changes
- Uses space more fully
 - Less need for reserve

Counterpoint: Concurrent Collection

Concurrent collector

- What happens if it starts too late?
 - Must do a stop-the-world collection
 - Much longer pause
- Can the collection start too early?
 - Yes, but rarely
 - Next collection will have better statistics

Summary on Concurrent Collector

“Latency Matters”

- Low latencies are achievable:
 - 500, 200, even 100 milliseconds maximum pause
 - Not guaranteed
 - Depends on application, heap size, hardware, etc.
- Some automatic adjustments
 - Default size of young generation
 - When to start old generation collection
 - Scheduling of old generation collection pauses
- Not fully “ergonomic” (yet)
 - Must specify heap sizes instead of pause time

Agenda

Welcome to “GC Today”

Introductions

What’s New in GC

Parallel Collector

Concurrent Collector

Ergonomics

Choosing a Collector

Futures

Questions From “Our Listeners”

Ergonomics

“Let the VM choose”

- Select the garbage collector
 - Chose a collector to fit the needs
- Select the heap size
 - One size does not fit all
- Select the runtime compiler
 - `-server` versus `-client`

Selections in the Java 2 Platform, Standard Edition (J2SE™) 1.4.x and Earlier

The same for all applications

- Serial collector
- Smaller heap settings
 - Initial heap: ~4MB
 - Maximum heap: 64MB
- Client runtime compiler

Ergonomic Selections in the J2SE 1.5.0 Platform

Better for some applications

- On server-class machines
 - Parallel collector
 - Larger heap settings
 - Initial heap: 1/64 physical memory up to 1GB
 - Maximum heap: 1/4 physical memory up to 1GB
 - Server runtime compiler
- On client machines
 - Same as the J2SE 1.4.x platform

Server-class Machines

How we make the decision

- Server-class machines have:
 - 2 CPUs (or more)
 - 2 GB memory (or more)
- Why base decision on machine type?
 - Keep it simple, easy to explain
 - Often indicates type of application
 - Currently, choice must be made at startup
- Ideally, select based on desired behavior
 - E.g., maximum pause time, desired throughput
 - Adapt at runtime
 - Future release

Choosing a Collector

- Is GC a problem?
 - If not, let the VM choose
- Small heap?
 - 20MB to 30MB or less, let the VM choose
- Pause time or response time limits?
 - 3 seconds or less: start with concurrent collector
 - 10 seconds or more: try parallel collector

Second Thoughts on Choosing the Concurrent Collector

- Allocation rate versus collection rate
 - Most collection work done with one thread
 - More than eight to twelve active user threads
 - Concurrent collector may not be able to keep up
- Only one or two CPUs
 - Concurrent work takes 1 CPU from application
- One sample application:
 - Four CPUs, 1GB heap, 20MB young generation
 - 100–200 millisecond pauses
 - Concurrent work
 - About 10–15 seconds per old generation collection
 - Old generation collections every few minutes

Agenda

Welcome to “GC Today”

Introductions

What’s New in GC

Parallel Collector

Concurrent Collector

Ergonomics

Choosing a Collector

Futures

Questions From “Our Listeners”

Futures

- Enhance ergonomics
 - Trade space between generations
 - Extend to concurrent collector
 - Select collector based on desired behavior
- Parallel old generation collector
 - Improve scalability, efficiency
- Garbage collecting old collectors
 - Train collector (use concurrent collector)
 - Serial collector (let the VM choose)

Command Line Flags

See the J2SE platform 1.5.0 release notes for details

- Select an old generation collector
 - XX:+UseParallelGC
 - XX:+UseConcMarkSweepGC
 - XX:+UseSerialGC
- Request performance characteristics
 - XX:GCTimeRatio=
 - XX:MaxGCPauseMillis=
- Change features to suit
 - Xms , -Xmx
 - XX:+UseAdaptiveSizePolicy

For More Information

- Meet the Java HotSpot™ VM Development Team (BOF-2520)
- Java™ Platform Performance (TS-1218)
- Performance of Java™ 2 Platform, Standard Edition (J2SE™), Java™ 2 Platform, Enterprise Edition (J2EE™) Web Server, Web Services, and Portals (BOF-1217)
- J2SE 1.4.2 platform GC tuning guide
 - <http://java.sun.com/docs/hotspot/gc1.4.2/>
 - An update is planned for J2SE 1.5.0
- J2SE 1.5.0 platform release notes

Agenda

Welcome to “GC Today”

Introductions

What’s New in GC

Parallel Collector

Concurrent Collector

Ergonomics

Choosing a Collector

Futures

Questions From “Our Listeners”

Q&A

John Coomes
Jon Masamitsu
Peter Kessler



JavaOne™

Sun's 2004 Worldwide Java Developer Conference™

Garbage Collection in the Java HotSpot™ Virtual Machine

Choices and Trade-Offs

Peter Kessler

Jon Masamitsu

John Coomes

Sun Microsystems, Inc.

<http://www.sun.com>

java.sun.com/javaone/sf

