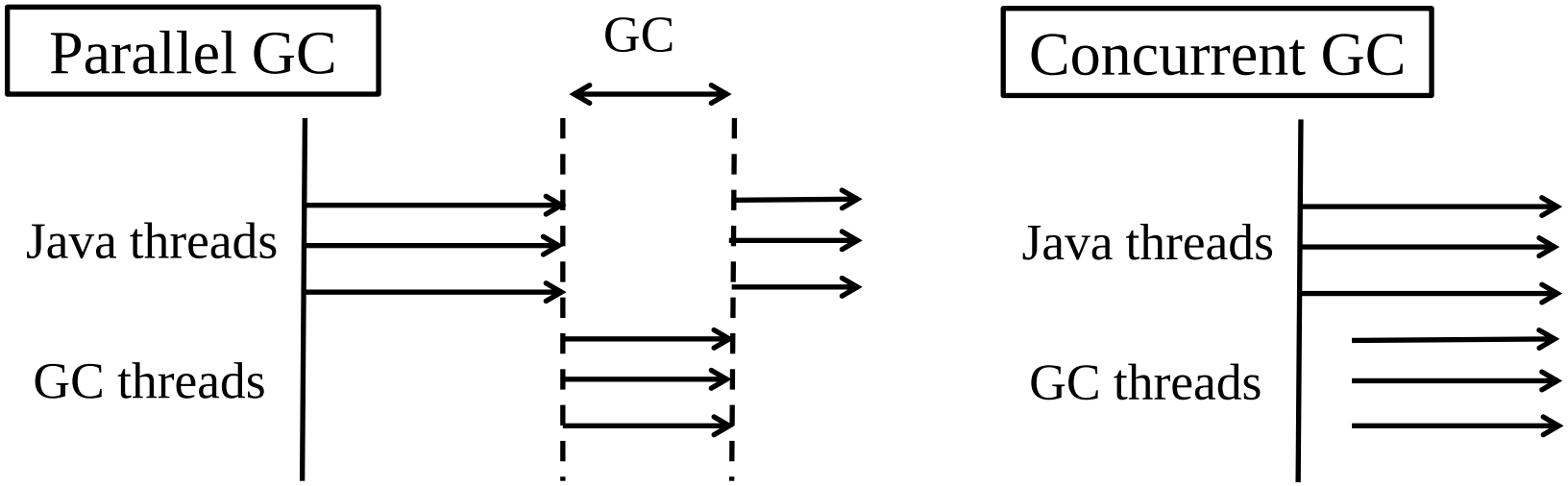
Concurrent Mark&Sweep

第6回JVMソースコードリーディングの会(OpenJDK)

中村 実
nminoru@nminoru.jp
nminoru1975@gmail.com
Twitter @nminoru_jp

Parallel GCとConcurrent GC

- Stop the world(STW)型GC
  - Javaスレッド(Mutator)を止めてからGC
  - Parallel GCはSTWした後のGC処理を複数のスレッドが分担
- Concurrent GC
  - JavaスレッドとGCスレッドが同時に動く

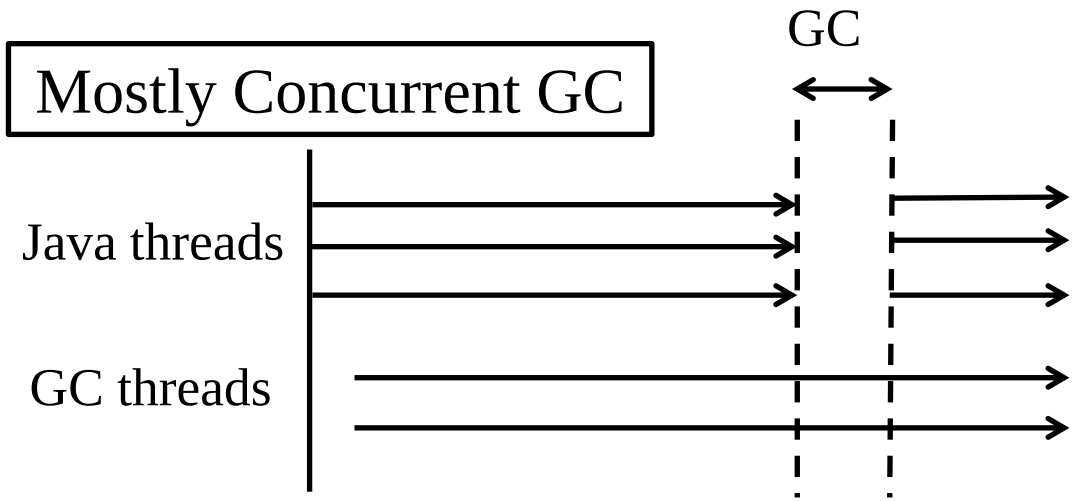


Concurrent GC

- メリット
  - Javaが常に動けるのでGCによる停止がない。
- デメリット
  - 機構が複雑になり、JavaスレッドとGCスレッドが邪魔しあうのでスループット性能が落ちる。

Hotspot VMのConcurrent GC

- Mostly Concurrent Mark & Sweep(CMS)
  - ほとんどconcurrentだが一部停止するGC
  - Tony Printezis, David Detlefs. A Generational Mostly-concurrent Garbage Collection (ISMM 2000)

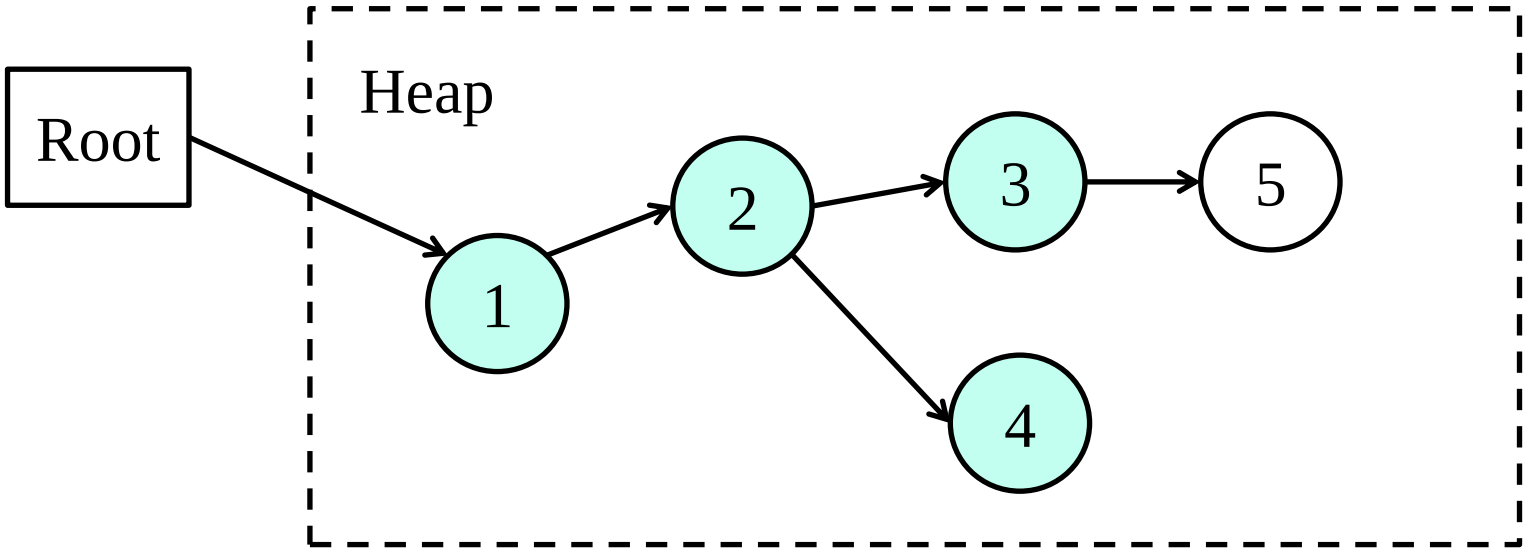


CMS

- Mark&Sweep
  - Marking phase
    - Concurrent marking
    - Serial marking(Hotspot VMの用語でfinal marking)
  - Sweeping phase
    - Concurrent sweeping

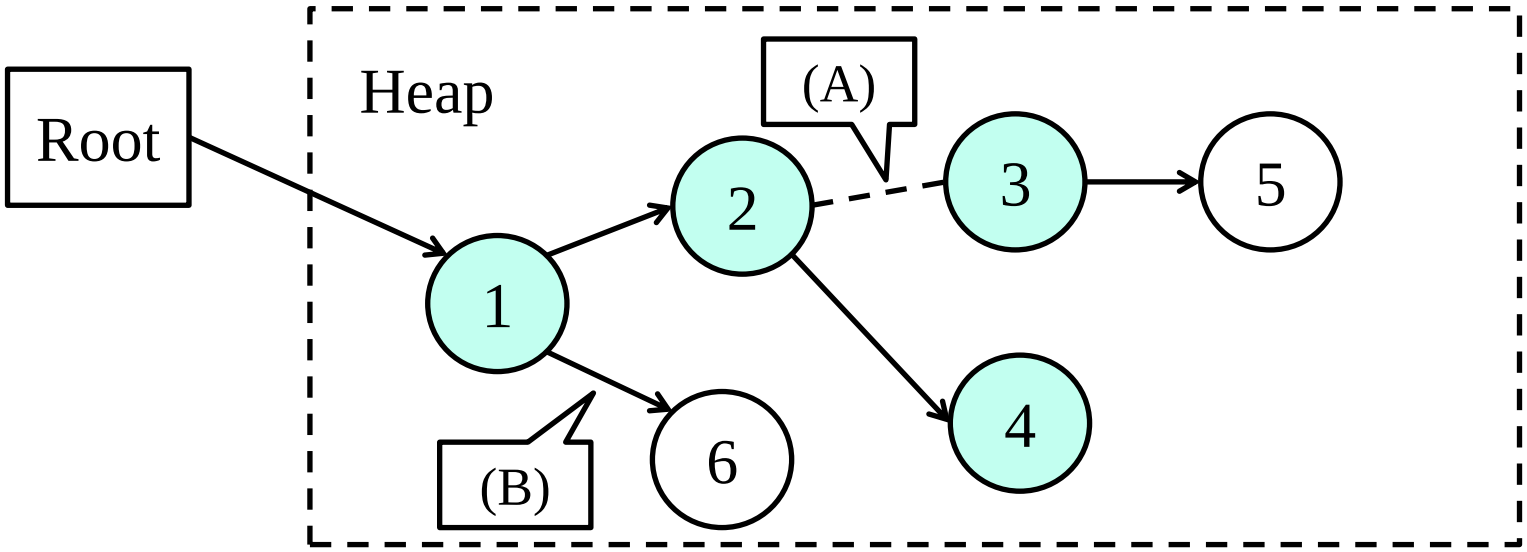
Concurrent Marking(1/3)

- Marking threadはrootから初めてオブジェクトを巡回しマーキングする(着色したオブジェクトはマークをあらわす)
- Javaスレッドはその間も動いて、オブジェクトを生成・参照の変更を行う。



Concurrent Marking(2/3)

- Marking threadとJavaスレッドが同時に動くと衝突が発生
  - (A) 余分なマーク
    - 3をマークし後に2→3の参照が切れた。
  - (B) マーク漏れ
    - 1をマークした後に6が作られた。
- (A)はGCが回収する量が減るだけだが、(B)は許されない。



Concurrent Marking(3/3)

- Write barrier
  - Javaスレッドがputfield/putstaticなどを実行した時に、そのrecieverにライトバリアを付ける。
    (A.field = B なら A にライトバリア)
- Serial marking
  - Concurrent markingの後にSTWを起こしてライトバリアがついている場所から残りのmarkingを行う。

OpenJDKでの CMSの実装

OpenJDKのGCの種類

	オプション	ヒープのクラス	空間のクラス
逐次GC	-XX:+UseSerialGC	GenCollectedHeap	DefNewGeneration TenuredGeneration
並列GC	-XX:+UseParNewGC	GenCollectedHeap	ParNewGeneration TenuredGeneration
並列GC	-XX:+UseParallelGC	ParallelScavengeHeap	
コンカレントGC	-Xconcgc or -XX:+UseConcMarkSweepGC	GenCollectedHeap	ParNewGeneration ConcurrentMarkSweep
インクリメンタルGC	-Xincgc (コンカレントGCに-XX:+CMSIncrementalModeをつけたもの)	GenCollectedHeap	同上
G1GC	-XX:+UseG1GC	G1CollectedHeap	

CMSのオプション

オプション	説明
-Xconcgc or -XX:+UseConcMarkSweepGC	Concurrent GCを有効にする。
-XX:ConcGCThreads=<n>	Concurrent GCスレッド数を指定する。 デフォルトは0でConcurrent GCスレッドは1となる。
-XX:+CMSParallelRemarkEnabled	CMS Full GCをマルチスレッドで実行する。 マルチCPU環境ではデフォルトで指定される。
-XX:CMSWaitDuration=<n>	同期型のGCが起きた後に<n>ミリ秒経った後にCMSを開始する。デフォルトは2000ミリ秒。
-XX:+PrintGC -XX:+PrintGCDetails	GCの詳細を表示する。
-XX:+TraceCMSState	Concurrent Mark&Sweepに関係する出力を増やす(デバッグ版のみ)。

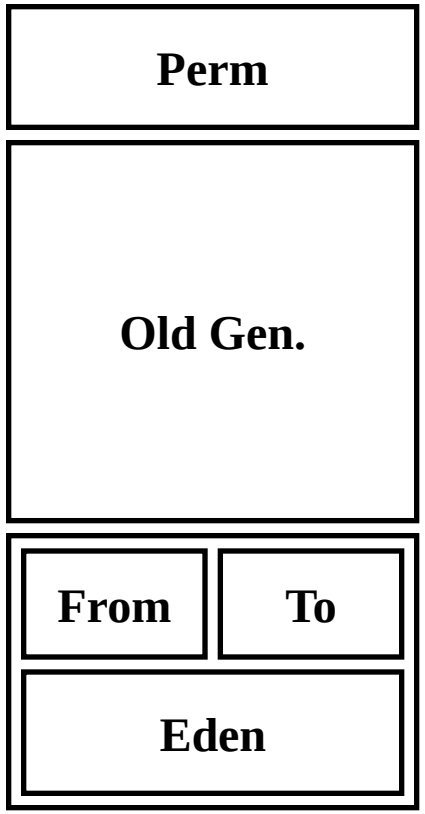
hotspot/src/share/vm/runtime/globals.hppに多数存在。

Hotspot VMのCMSの概要

- ConcurrentMarkSweepThreadスレッドが全体を制御
- ヒープ空間は新旧2世代
- Write barrierはCard markingで代替
- CMSのマーキングはビットマーキング

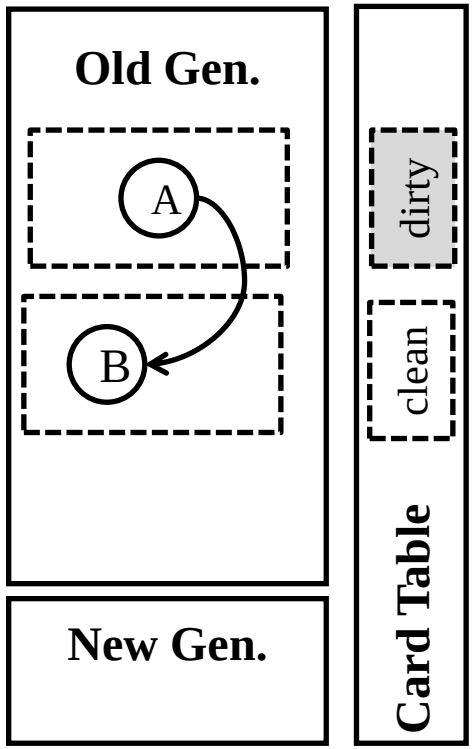
ヒープ構成

- 空間配置
  - 新世代はParNewGC(parallel copying GC)用。
  - 旧世代はCMS用でフリーリスト管理されている。
- GC
  - 最初に新世代にParNewGCを起こす(STW型)
  - 次に旧世代にCMS GCを起こす。
  - それでも不足の場合はCMS Full GCを起こす(STW型)



Card Makingの流用

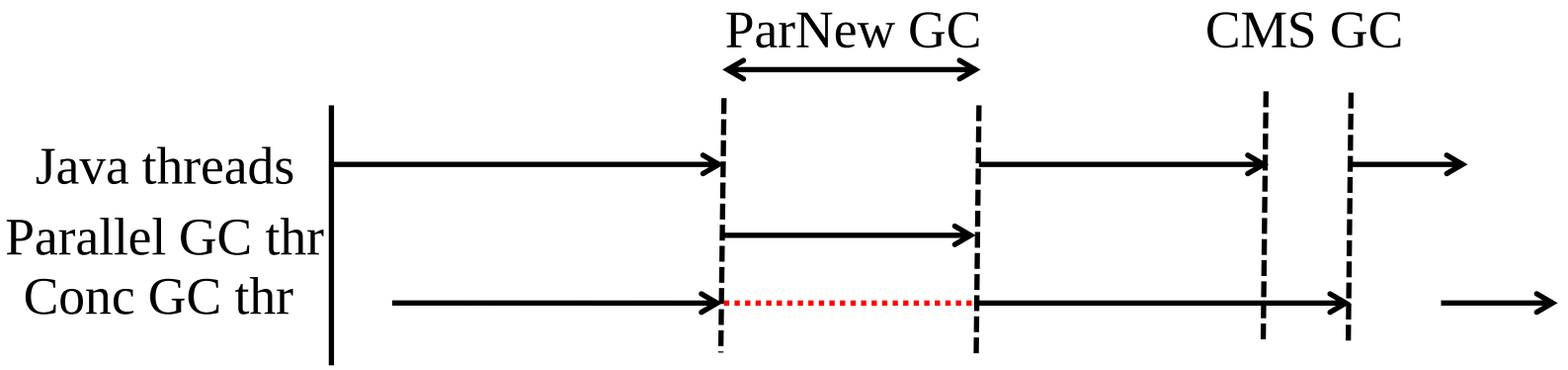
- Concurrent markingのためにはライトバリアが必要。でも世代別GCには似たような処理があるので流用
- Card Table
  - 512バイト単位に1バイトのバイトテーブル
  - Clean cardは0xFF、dirty cardは0
  - 本来は旧世代から新世代を指すオブジェクトの位置を記録
- putfield/putstatic/aastoreにフック
  - A.field = B実行時になら



CardTableModRefBS::byte_map_base [uintptr_t(A) >> CardTableModRefBS::card_shift] = 0

CMS GCとParNewGCの競合

- マーキング領域
  - CMSのマーキングは専用のビットマップを使う(CMSBitMap)ので問題なし。
- GC
  - ConcurrentMarkSweepThreadスレッドが動いている最中にParNewGCが起きることがある。
    その場合はConcurrentMarkSweepThreadが一時停止してから再開する。



CMSのフェーズ

- ConcurrentMarkSweepThread::run at concurrentMarkSweepThread.cpp:128
  - CMSCollector::collect_in_background at concurrentMarkSweepGeneration.cpp:2246

1	InitialMarking	(STW)Rootからのマーキング	VM_CMS_Initial_Mark::doit()
2	Marking	通常のconcurrent marking	CMSCollector::markFromRoots()
3	Precleaning	弱参照系のconcurrent marking	CMSCollector::preclean()
4	AbortablePreclean		CMSCollector::abortable_preclean()
5	FinalMarking	(STW)Markingの残り	VM_CMS_Final_Remark::doit()
6	Sweeping	回収	CMSCollector::sweep
7	Resizing	空間のサイズ調整	CMSCollector::compute_new_size
8	Resetting	ビットマップのクリアなど	CMSCollector::reset
9	Idling	CMSWaitDurationミリ秒待機	

3と4の処理は省略されることがある。

InitialMarking

- Rootからのマークはstwで行う。
- ConcurrentMarkSweepThreadスレッドが指示を出すが、実際の処理はVMThreadが行う。

```
#0  CMSBitMap::mark() at globalDefinitions.hpp:418
#1  MarkRefsIntoClosure::do_oop() at concurrentMarkSweepGeneration.cpp:6586
#2  MarkRefsIntoClosure::do_oop_work<oopDesc*>(oopDesc**)  at
hotspot/src/share/vm/utilities/globalDefinitions.hpp:418
#3  MarkRefsIntoClosure::do_oop() at concurrentMarkSweepGeneration.cpp:6590
#4  Universe::oops_do() at hotspot/src/share/vm/memory/universe.cpp:208
#5  SharedHeap::process_strong_roots() at hotspot/src/share/vm/memory/sharedHeap.cpp:139
#6  GenCollectedHeap::gen_process_strong_roots() at hotspot/src/share/vm/memory/genCollectedHeap.cpp:741
#7  CMSCollector::checkpointRootsInitialWork() at concurrentMarkSweepGeneration.cpp:3570
#8  CMSCollector::checkpointRootsInitial() at concurrentMarkSweepGeneration.cpp:3489
#9  CMSCollector::do_CMS_operation() atconcurrentMarkSweepGeneration.cpp:6306
#10 VM_CMS_Initial_Mark::doit() at concurrentMarkSweep/vmCMSOperations.cpp:140
#11 VM_Operation::evaluate() at hotspot/src/share/vm/runtime/vm_operations.cpp:65
```

Marking

```
#0  CMSBitMap::mark() at globalDefinitions.hpp:418
#1  PushOrMarkClosure::do_oop() at concurrentMarkSweepGeneration.cpp:7555
#2  PushOrMarkClosure::do_oop_work<oopDesc*>(oopDesc**) at globalDefinitions.hpp:418
#3  PushOrMarkClosure::do_oop_nv(oopDesc**) () at hotspot/src/share/vm/runtime/thread.hpp:1826
#4  instanceKlass::oop_oop_iterate_nv() at hotspot/src/share/vm/oops/instanceKlass.cpp:1825
#5  instanceRefKlass::oop_oop_iterate_nv() at hotspot/src/share/vm/oops/instanceRefKlass.cpp:285
#6  oopDesc::oop_iterate() at /globalDefinitions.hpp:418
#7  MarkFromRootsClosure::scanOopsInOop() at concurrentMarkSweepGeneration.cpp:7215
#8  MarkFromRootsClosure::do_bit() at concurrentMarkSweepGeneration.cpp:7113
#9  BitMap::iterate() at hotspot/src/share/vm/utilities/bitMap.cpp:512
#10 BitMap::iterate() at globalDefinitions.hpp:418
#11 CMSBitMap::iterate() at globalDefinitions.hpp:418
#12 CMSCollector::do_marking_st() at concurrentMarkSweepGeneration.cpp:4325
#13 CMSCollector::markFromRootsWork() at concurrentMarkSweepGeneration.cpp:3692
#14 CMSCollector::markFromRoots() at concurrentMarkSweepGeneration.cpp:3629
#15 CMSCollector::collect_in_background() at concurrentMarkSweepGeneration.cpp:2288
#16 ConcurrentMarkSweepThread::run() at concurrentMarkSweepThread.cpp:128
```

Preclean

#0 CMSDrainMarkingStackClosure::do_void() at concurrentMarkSweepGeneration.cpp:8638
#1 ReferenceProcessor::preclean_discovered_reflist() at
hotspot/src/share/vm/memory/referenceProcessor.cpp:1433
#2 ReferenceProcessor::preclean_discovered_references() at
hotspot/src/share/vm/memory/referenceProcessor.cpp:1345
#3 CMSCollector::preclean_work() at concurrentMarkSweepGeneration.cpp:4542
#4 CMSCollector::preclean() at concurrentMarkSweepGeneration.cpp:4376
#5 CMSCollector::collect_in_background() at concurrentMarkSweepGeneration.cpp:2300
#6 ConcurrentMarkSweepThread::run() at concurrentMarkSweepThread.cpp:128

FinalMarking

- Markingの最後はSTWして行う。
- Card markingによってdirty cardがある場所のみをマークする。

```
#0  CMSBitMap::mark_range() at hotspot/src/share/vm/memory/cardTableRS.hpp:150
#1  CardTableModRefBS::dirty_card_iterate() at
hotspot/src/share/vm/memory/cardTableModRefBS.cpp:617
#2  CMSParRemarkTask::do_dirty_card_rescan_tasks() at
concurrentMarkSweepGeneration.cpp:5322
#3  CMSParRemarkTask::work() at concurrentMarkSweepGeneration.cpp:5171
#4  GangWorker::loop() at hotspot/src/share/vm/utilities/workgroup.cpp:308
#5  GangWorker::run() at OpenJDK/hotspot/src/share/vm/utilities/workgroup.cpp:224
#6  java_start() at OpenJDK/hotspot/src/os/linux/vm/os_linux.cpp:852
```

Sweep(1/3)

- CMSCollector::sweep at concurrentMarkSweepGeneration.cpp:5976

```
void CMSCollector::sweep(bool asynch) {

  // PermGen の sweep が禁止されていた場合でも無効なオブジェクトを
  // _perm_gen_verify_bit_map BitMap に移す。
  MarkDeadObjectsClosure mdo(this, _permGen->cmsSpace(),
                             markBitMap(), perm_gen_verify_bit_map());

  _permGen->cmsSpace()->blk_iterate(&mdo);

  // Sweep 処理
  sweepWork(_permGen, asynch);
```

Sweep(2/3)

concurrentMarkSweepGeneration.cpp:9214

```
size_t MarkDeadObjectsClosure::do_blk(HeapWord* addr) {
        size_t res = _sp->block_size_no_stall(addr, _collector);
        if (_sp->block_is_obj(addr)) {
                if (_live_bit_map->isMarked(addr)) {
                } else {
                        _dead_bit_map->mark(addr);      // mark the dead object
                }
        }
        return res;
}
```

```
#0  MarkDeadObjectsClosure::do_blk() at concurrentMarkSweepGeneration.cpp:9214
#1  CompactibleFreeListSpace::blk_iterate() at concurrentMarkSweep/compactibleFreeListSpace.cpp:771
#2  CMSCollector::sweep() at concurrentMarkSweepGeneration.cpp:6000
#3  CMSCollector::collect_in_background() at concurrentMarkSweepGeneration.cpp:2333
#4  ConcurrentMarkSweepThread::run() at concurrentMarkSweepThread.cpp:128
```

Sweep(3/)

#0 SweepClosure::do_yield_work() at concurrentMarkSweepGeneration.cpp:8435
#1 SweepClosure::do_yield_check(HeapWord*) at
hotspot/src/share/vm/utilities/globalDefinitions.hpp:418
#2 SweepClosure::do_blk_careful() at concurrentMarkSweepGeneration.cpp:8017
#3 CompactibleFreeListSpace::blk_iterate_careful() at
concurrentMarkSweep/compactibleFreeListSpace.cpp:763
#4 CMSCollector::sweepWork() at concurrentMarkSweepGeneration.cpp:6201
#5 CMSCollector::sweep() at concurrentMarkSweepGeneration.cpp:6018
#6 CMSCollector::collect_in_background() at concurrentMarkSweepGeneration.cpp:2333
#7 ConcurrentMarkSweepThread::run() at concurrentMarkSweepThread.cpp:128