Garbage Collection in Smalltalk

* By John M McIntosh
  - Corporate Smalltalk Consulting Ltd.
  - http://www.smalltalkconsulting.com
  - johnmci@smalltalkconsulting.com

    *
    * *Maintainer of the Squeak Macintosh VM.*
    * *Squeak TK4 VM support {Ask me later about TK4}*
    * *Trip reports for OOPSLA, Camp Smalltalk, etc.*

For ESUG 2004

© copyright 1997-2004 John M McIntosh, all rights reserved.

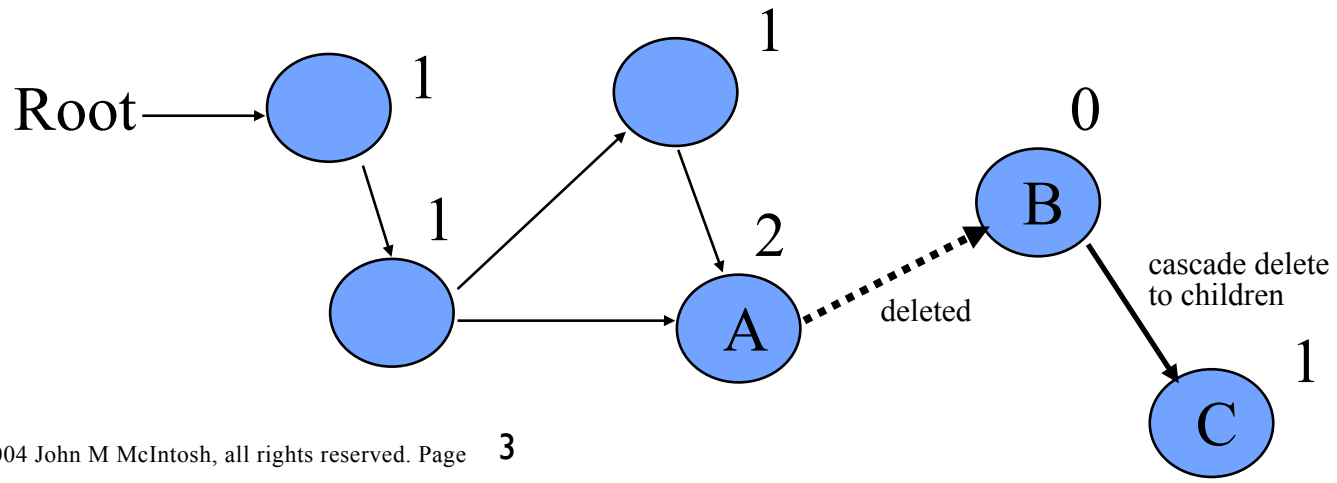
Garbage Collection - a worldly view

* Internet group postings in the last year:
  - “I eventually 'killed' the thing by enumerating over it and using 'become' to transform it into a string.”
  - “What happens to Threads when they die? Do they go to heaven? Or does the garbage collector take them away?”
  - “Just a thought that there is a dead object seating (sic) in my memory listening to events and doing funny things without my knowledge is scary and can produce hard to debug behavior.”

© copyright 1997-2004 John M McIntosh, all rights reserved.

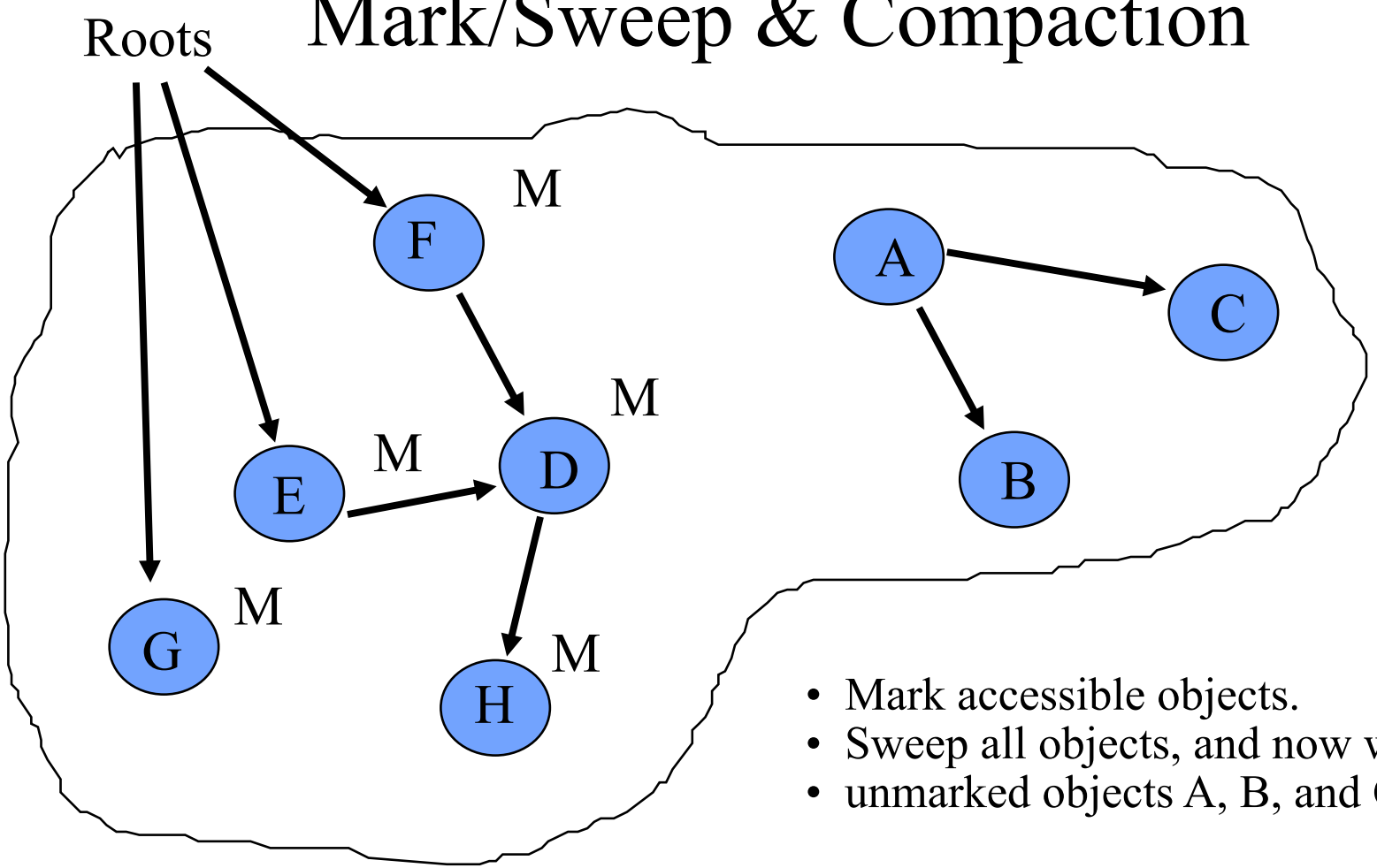
Reference Counting

* Each object has a counter to track references.
* As references to an object are made/deleted, this counter is incremented/decremented.
* When a reference count goes to *zero*, the object is <usually> deleted and any referenced children counters get decremented.



© copyright 1997-2004 John M McIntosh, all rights reserved.

Mark/Sweep
Mark/Sweep & Compaction

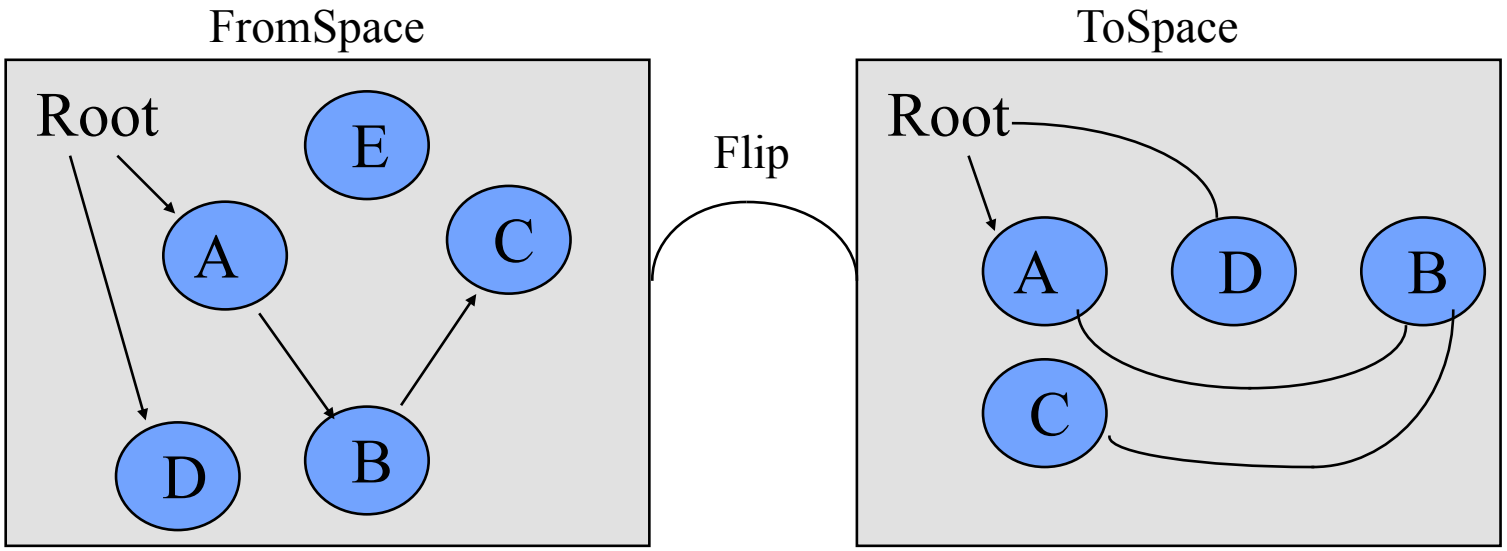


- Mark accessible objects.
- Sweep all objects, and now we realize,
- unmarked objects A, B, and C are free

Expensive optional compaction event clumps results together making free space one big block.

© copyright 1997-2004 John M McIntosh, all rights reserved.

Copying - The Flip or **Scavenge**



* When FromSpace is full we *flip* to ToSpace:
  - Copy roots of the world to ToSpace.
  - Copy root accessible objects to ToSpace.
  - Copy objects accessible from root accessible objects to ToSpace.
  - Repeat above until done.
* Cost is related to number of accessible objects in FromSpace.

Notice the change of Placement and that E isn't copied.

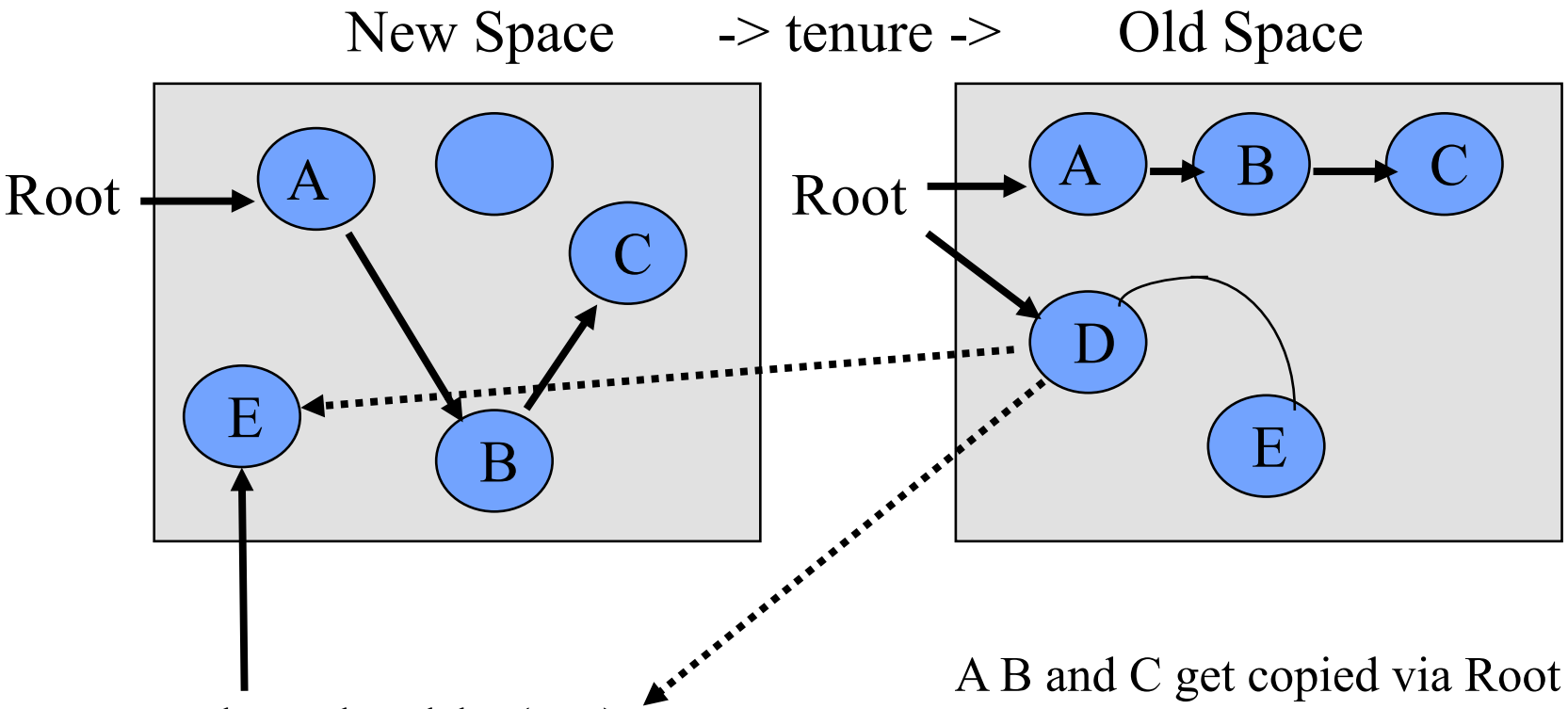
© copyright 1997-2004 John M McIntosh, all rights reserved.

Generational Garbage Collector

* Separate objects by age into two or more regions.
  - For example: Tektronix 4406 Smalltalk used seven regions, 3 bits Pat Caudill (1945-2001)
* Allocate objects in *new* space, when full then copy accessible objects to *old* space. This is known as a *scavenge* event.
* Movement of objects to old space is called *tenuring*.
* Objects must have a high death rate and low old to young object references. (Eh?). . . Both very important issues, I'll explain in a few minutes.

© copyright 1997-2004 John M McIntosh, all rights reserved.

Generational Scavenge Event



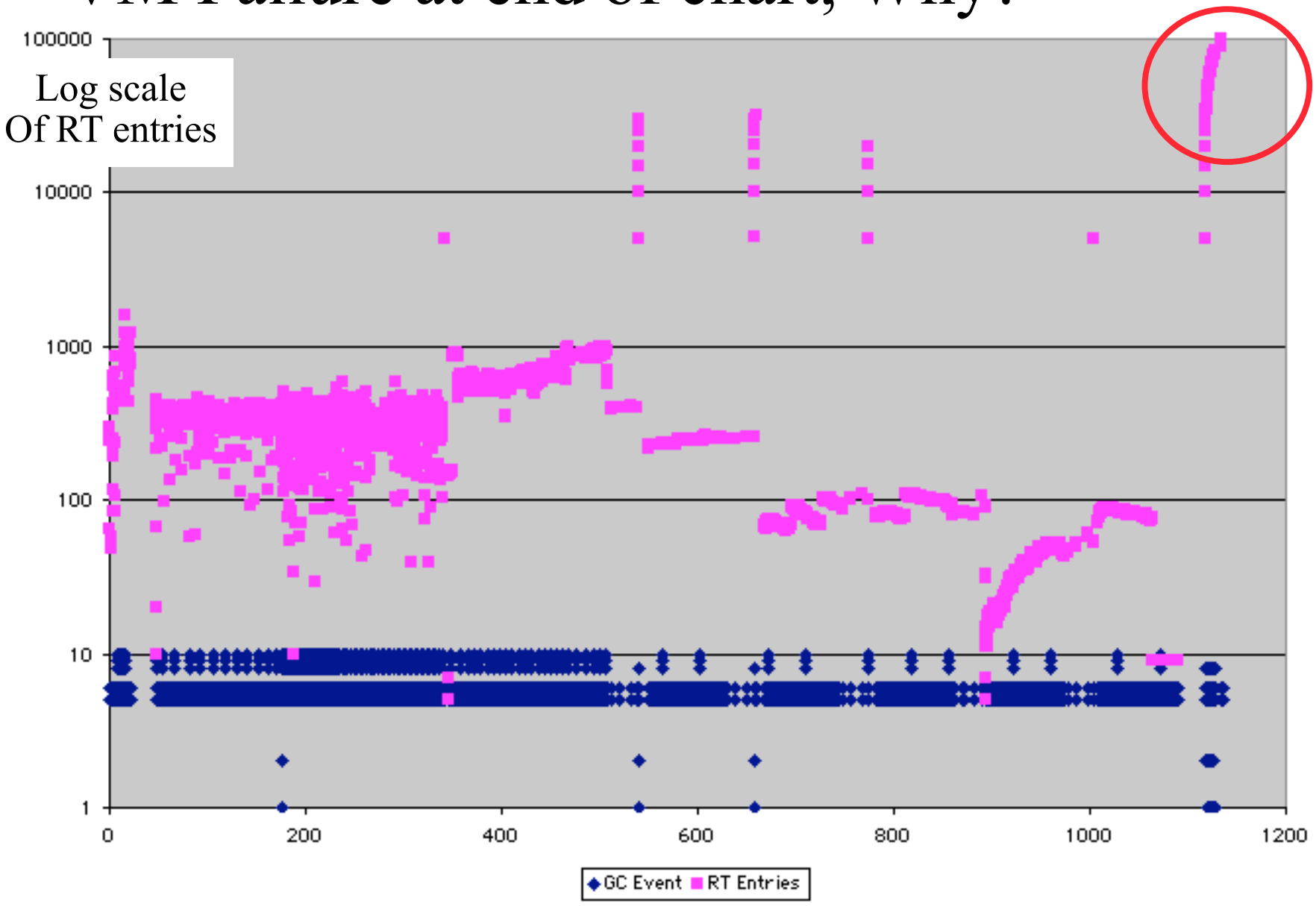
Remembered Table (RT)

InterGenerational References

A B and C get copied via Root reference. *E is copied via OldSpace reference from D, which is remember by being stored in the Remember Table.*

© copyright 1997-2004 John M McIntosh, all rights reserved.

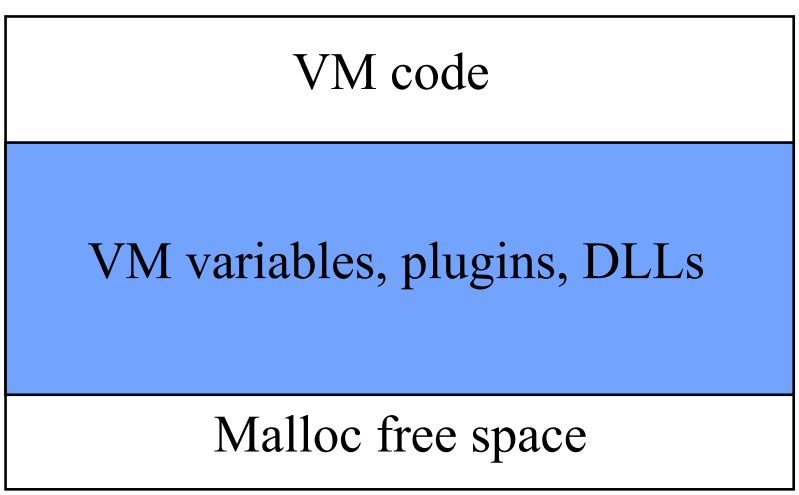
VM Failure at end of chart, Why?



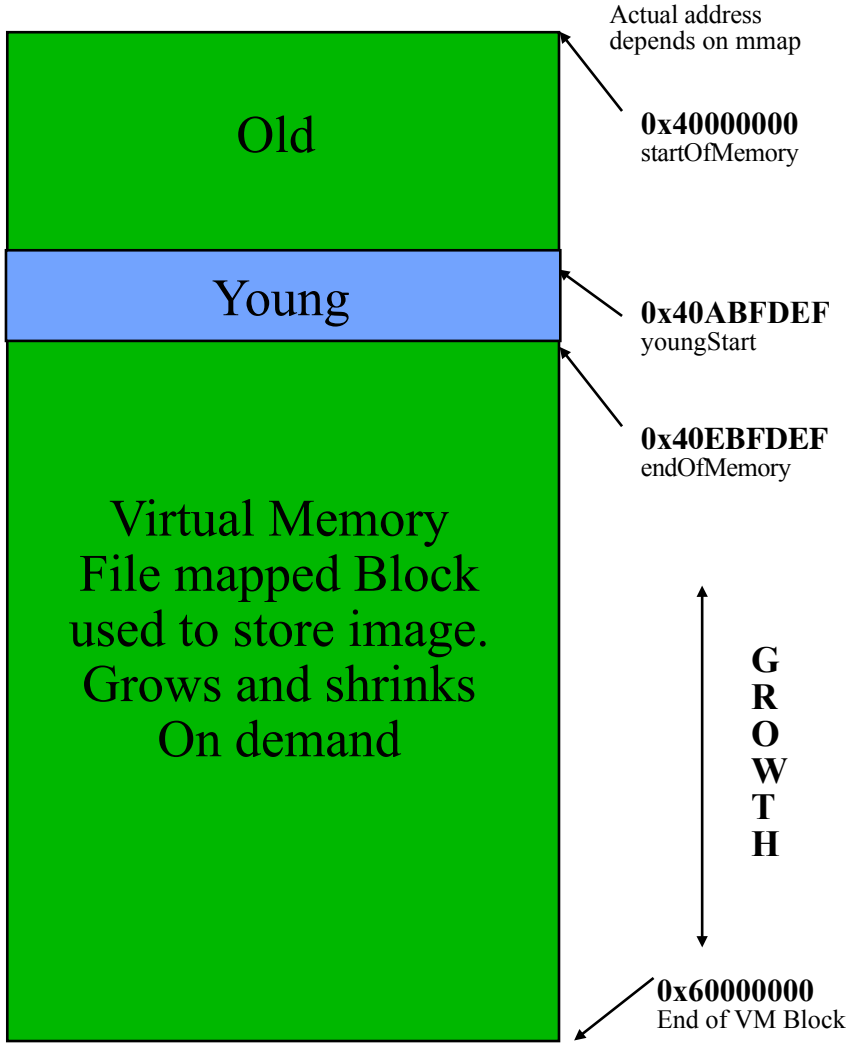
© copyright 1997-2004 John M McIntosh, all rights reserved.

Squeak Memory Layout

Generational Mark/Sweep Compacting



+

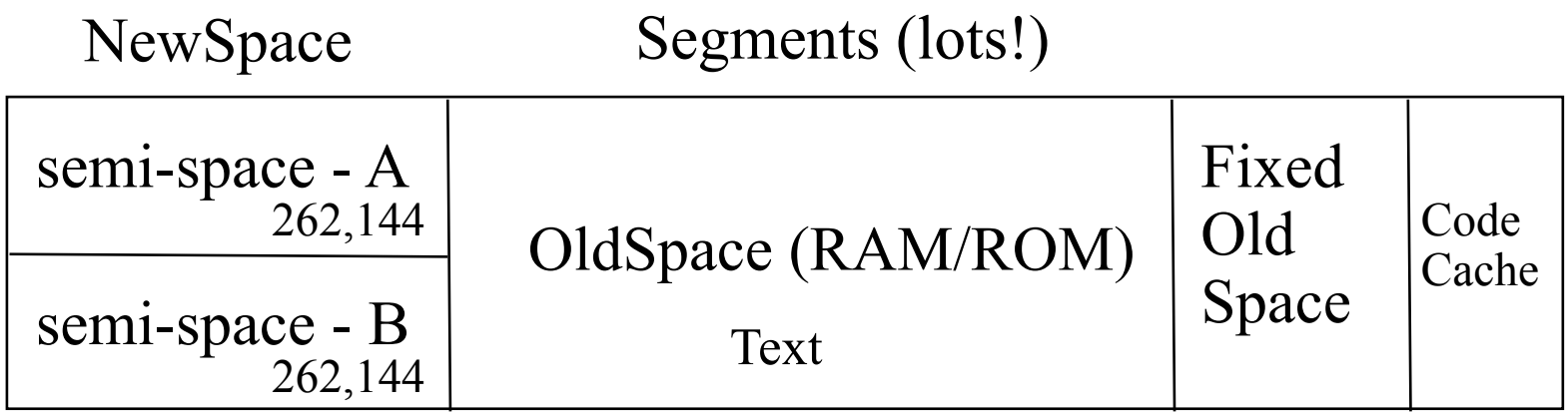


Applies to Squeak VM that support Virtual Memory file mapping and ability to grow/shrink image space (win32, some unix, mac os-9/os-x) Issues with non 32bit clean code limit object space to first 2GB of Address space.

© copyright 1997-2004 John M McIntosh, all rights reserved.

IBM VisualAge Memory Layout v5.5.1

Generational Copying (compacting optional)



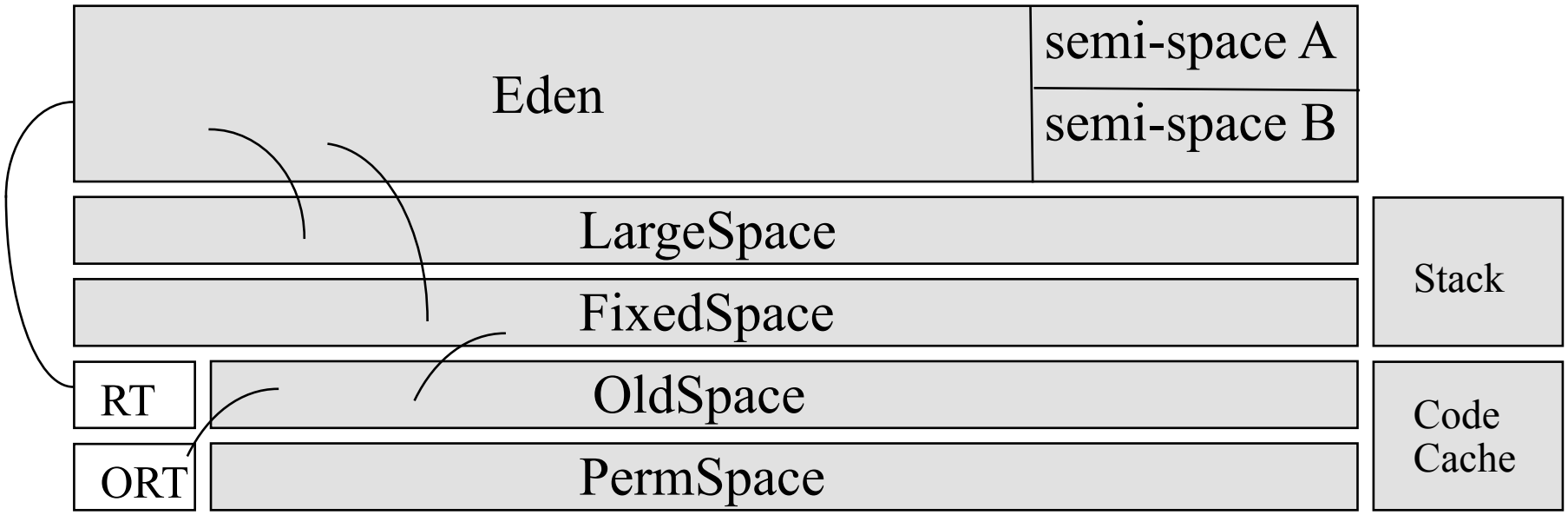
* Generational Copy Garbage Collector & Mark/Sweep.
* Copy between semi-spaces until full
* Then tenure some objects to current OldSpace segment
* Object loader/dumper can allocate segments (supports ROM)
* EsMemorySegment activeNewSpace

* To set current NewSpace semi-spaces size. Default is 262,144 in size
* abt -imyimage.im -mn###### (Start with ### byte in NewSpace)

© copyright 1997-2004 John M McIntosh, all rights reserved.

VisualWorks Memory Layout V5.i2

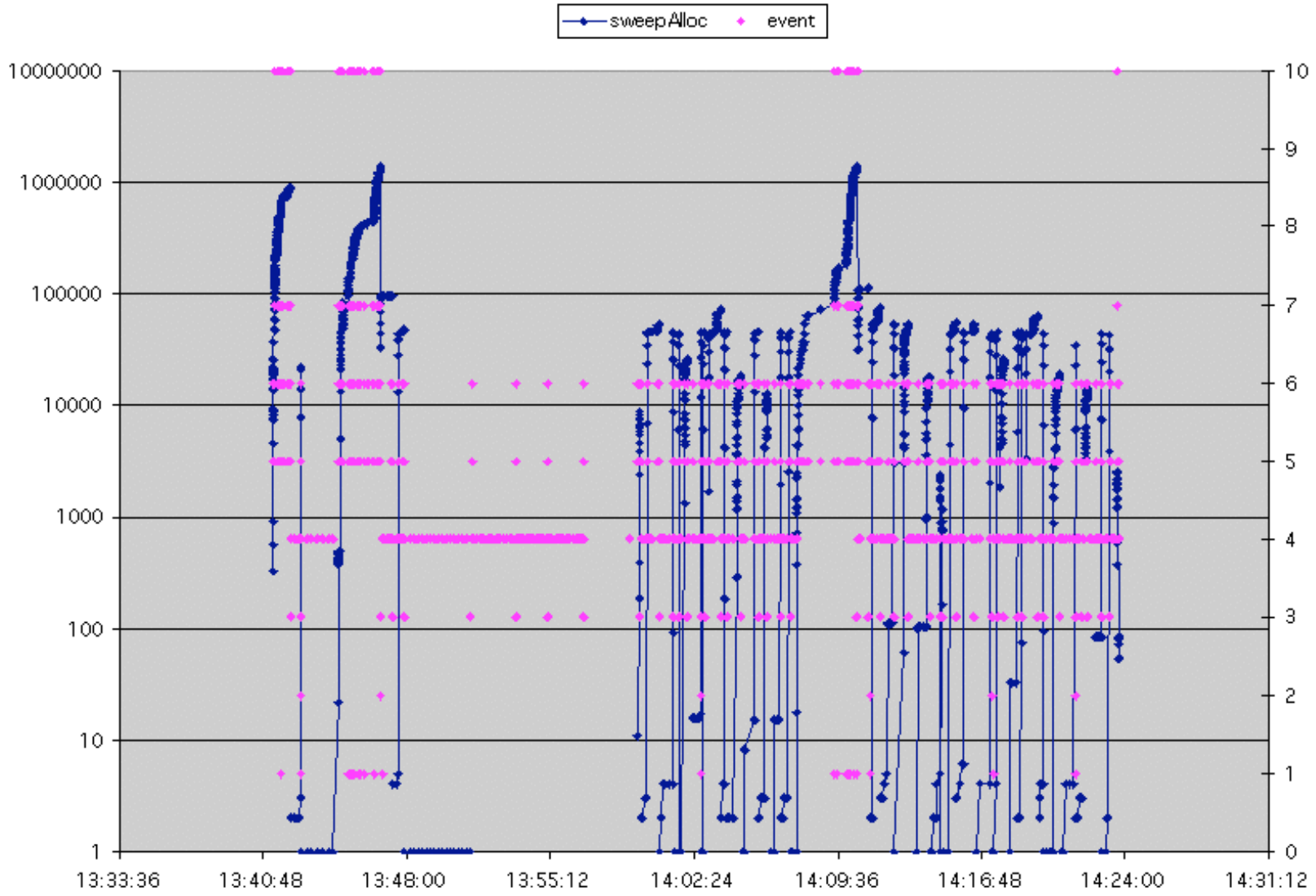
Generational Copying, + Incremental Mark/Sweep (Compacting optional)



* Allocate objects or headers in Eden (bodies in Eden, Large, or Fixed).
* Full? Copy Eden and active semi-space survivors to empty semi-space.
* When semi-space use exceeds threshold, tenure some objects to OldSpace.
* Once in OldSpace, use a Incremental Mark/Sweep GC to find garbage.

© copyright 1997-2004 John M McIntosh, all rights reserved.

SurvivorSpace small, watch how objects get tenured



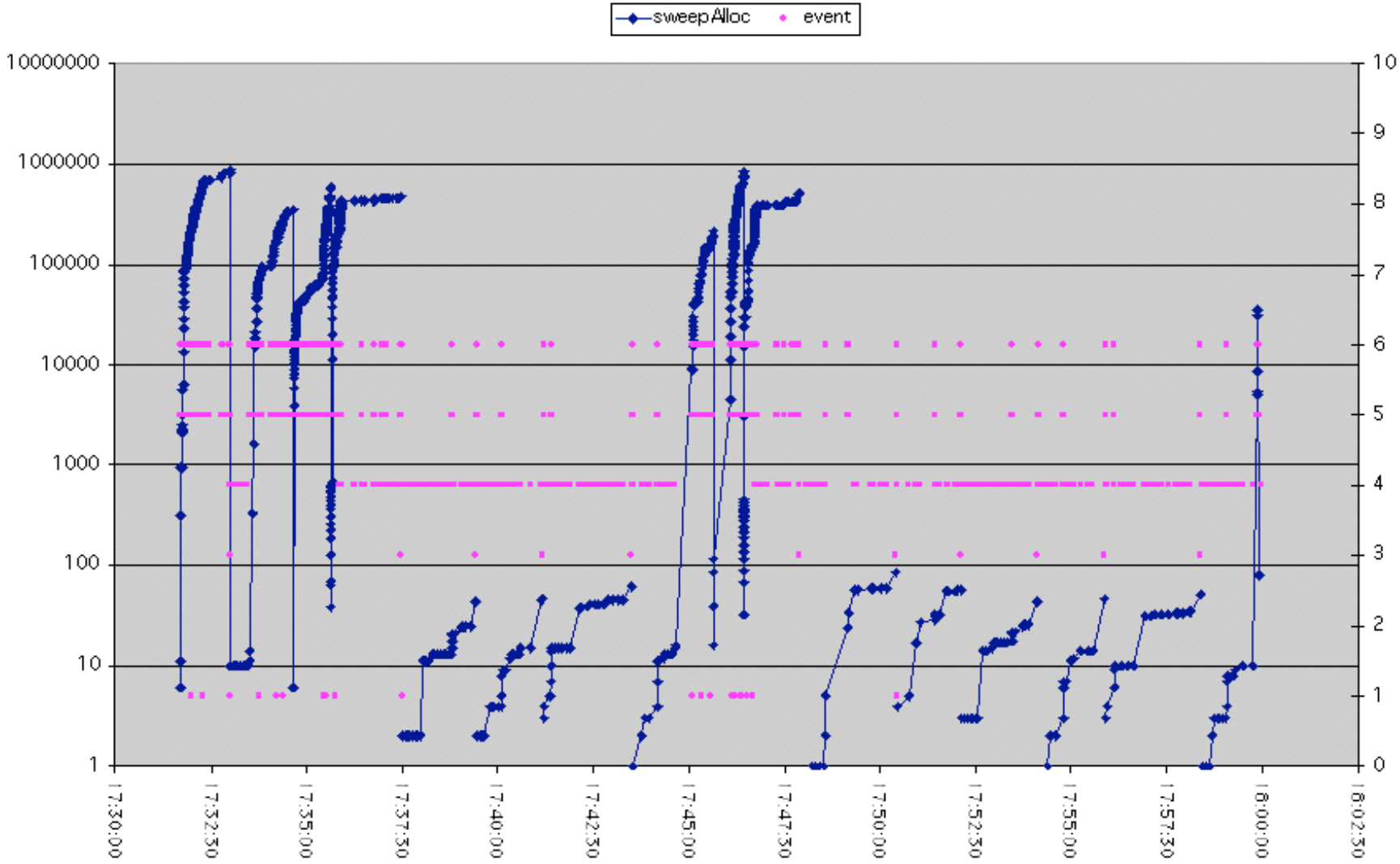
© copyright 1997-2004 John M McIntosh, all rights reserved.

Key to GC Events

1	Compact GC event: Full mark/sweep/compact OldSpace
2	Compacting decision has been made
3	IGC justified, interruptible, full cycle via idle loop call
4	Idle Loop Entered
5	Low Space Action Entered via VM trigger
6	*Incremental GC, (work quotas) attempt to cleanup OldSpace*
7	Request grow; Grow if allowed
8	LowSpace and we must grow, but first do aggressive GC work: Finish IGC, do OldSpace Mark/Sweep GC, if required followup with OldSpace Mark/Sweep/Compact
9	Grow Memory required
10	Grow Memory attempted, may fail, but usually is granted

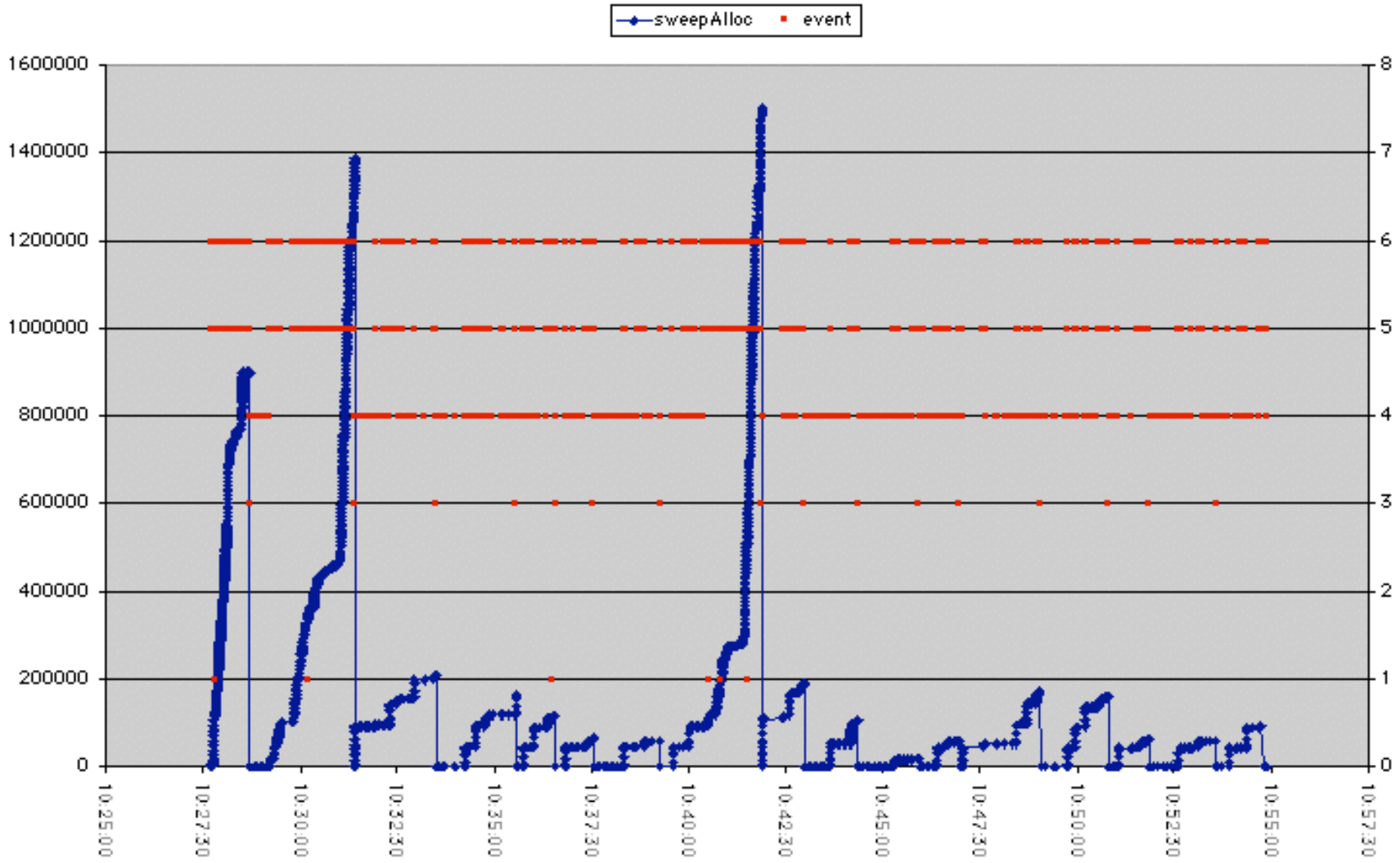
© copyright 1997-2004 John M McIntosh, all rights reserved.

SurvivorSpace larger, less objects get tenured



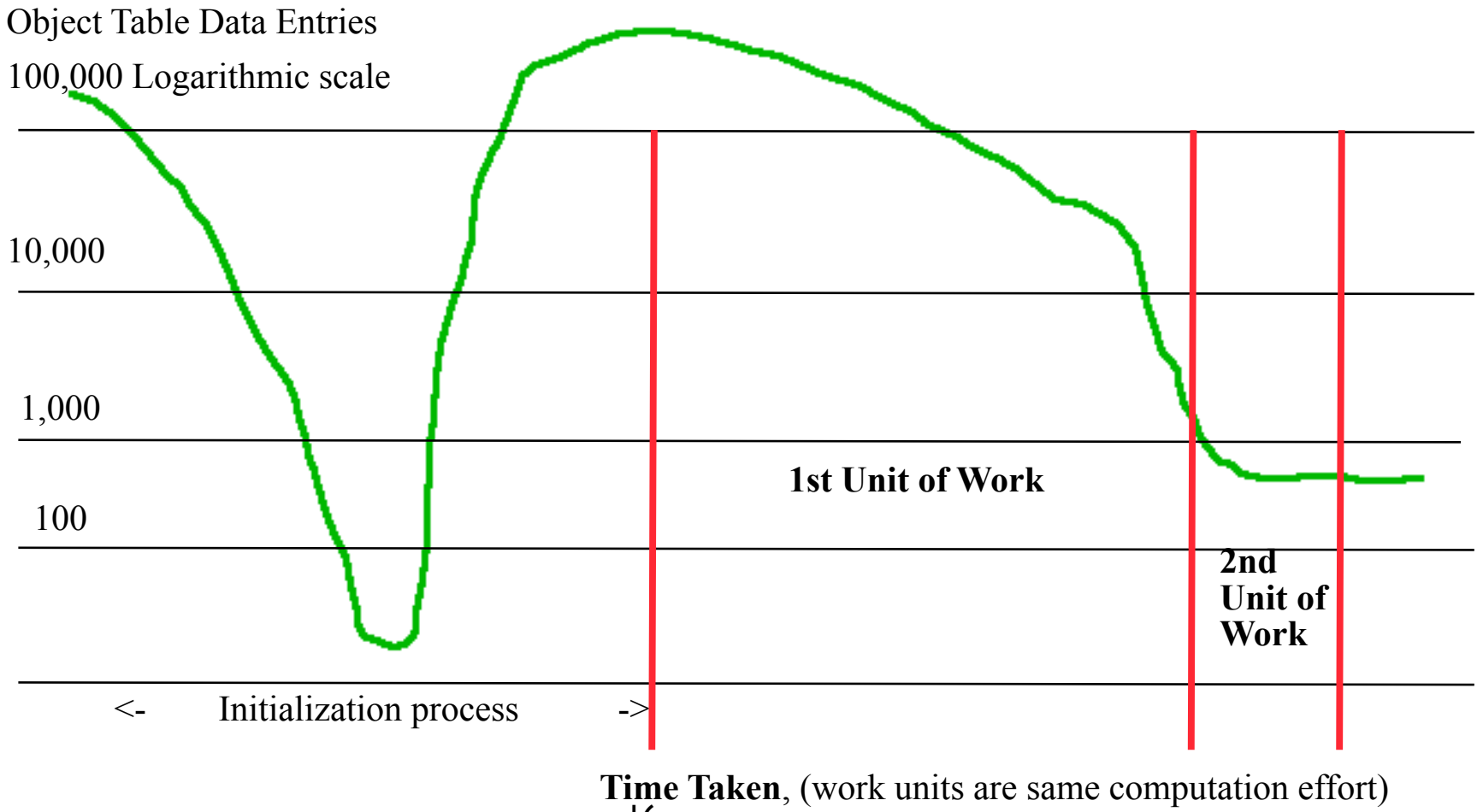
© copyright 1997-2004 John M McIntosh, all rights reserved.

Beware trade-off is complex, bigger SurvivorSpace =? Poor Performance



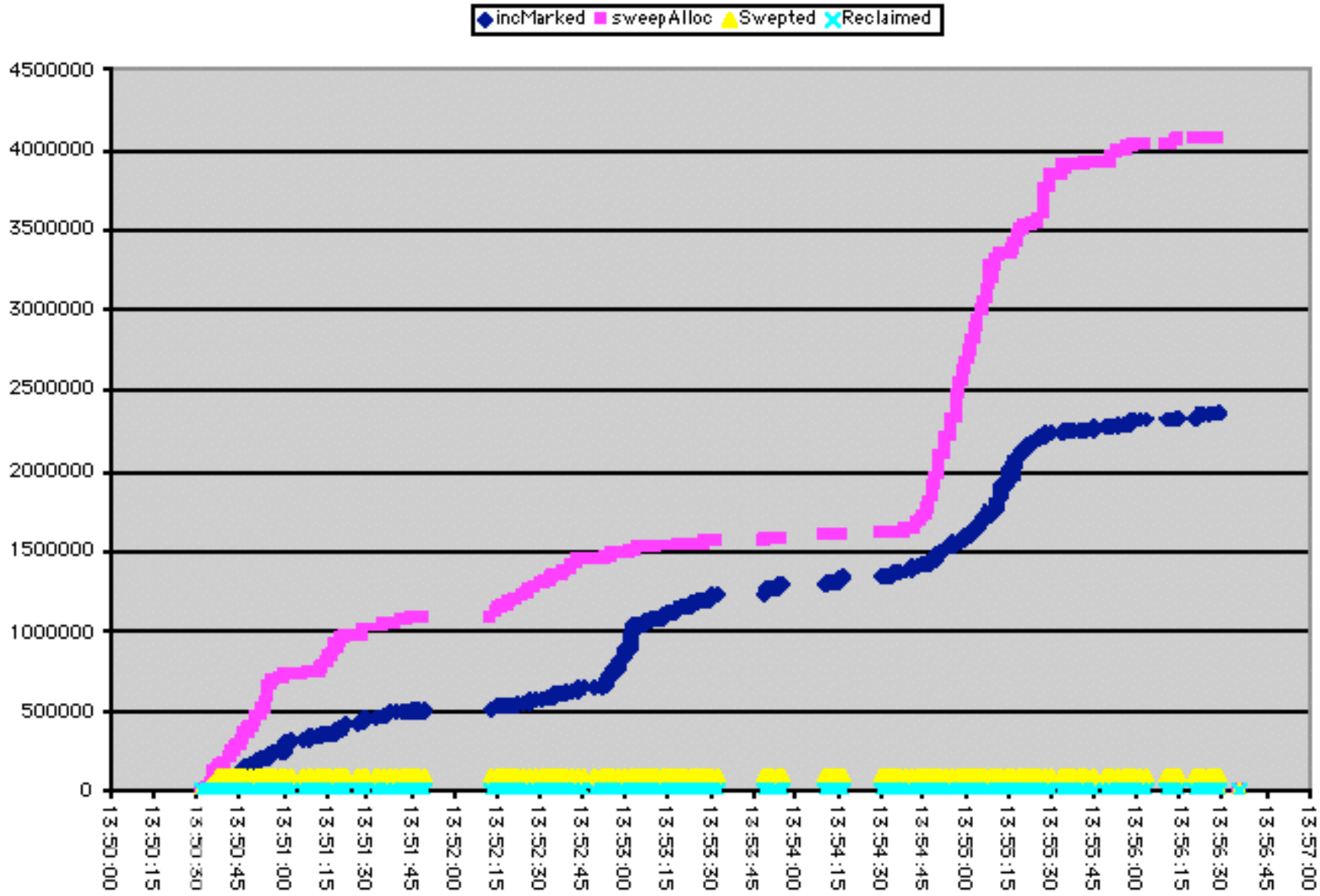
© copyright 1997-2004 John M McIntosh, all rights reserved.

Impact of pre VW 5.x single free chain performance issue
First unit of work takes 10x longer than 2nd unit of work. Why?



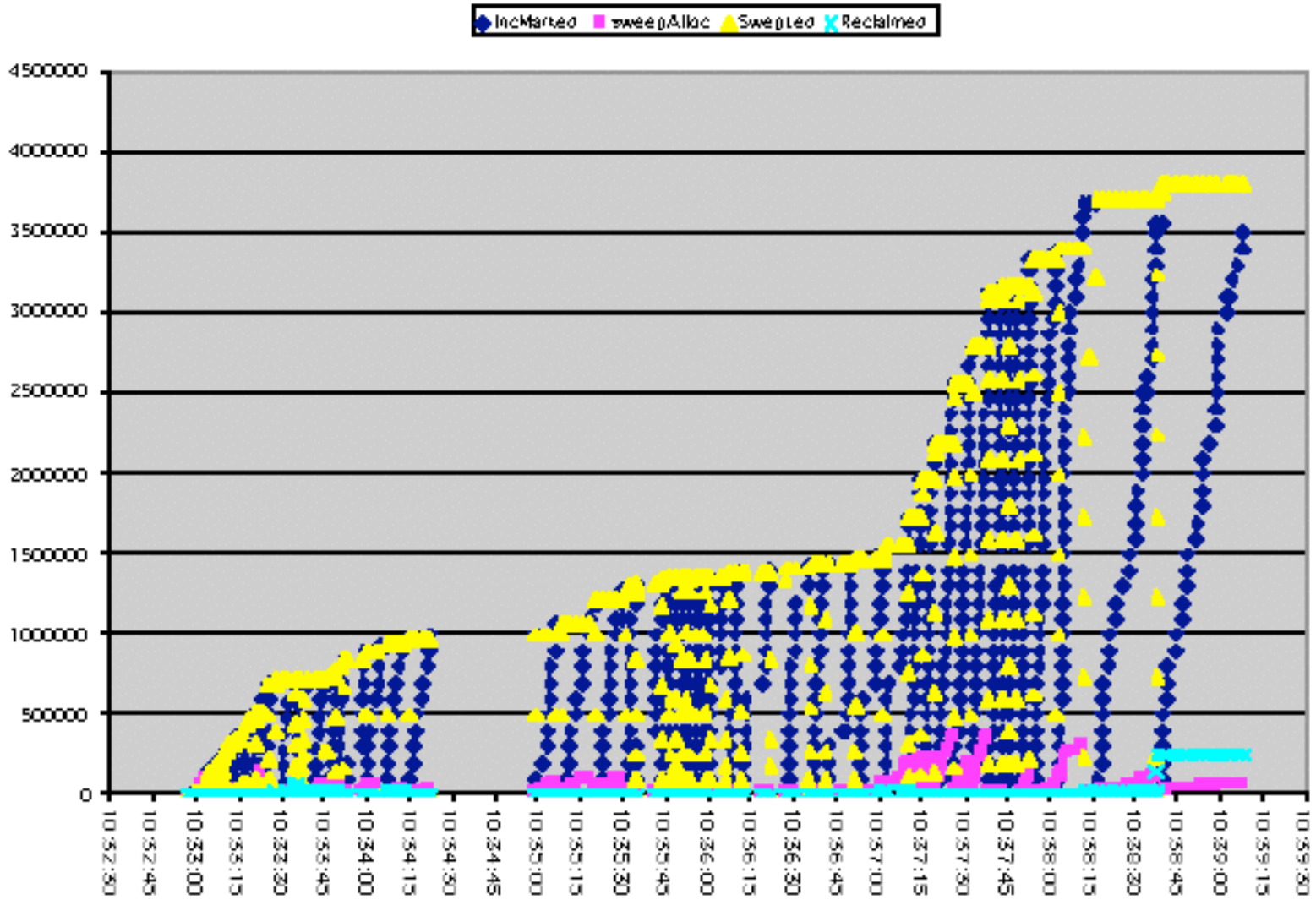
© copyright 1997-2004 John M McIntosh, all rights reserved.

Incorrect IGC work loads, IGC never completes



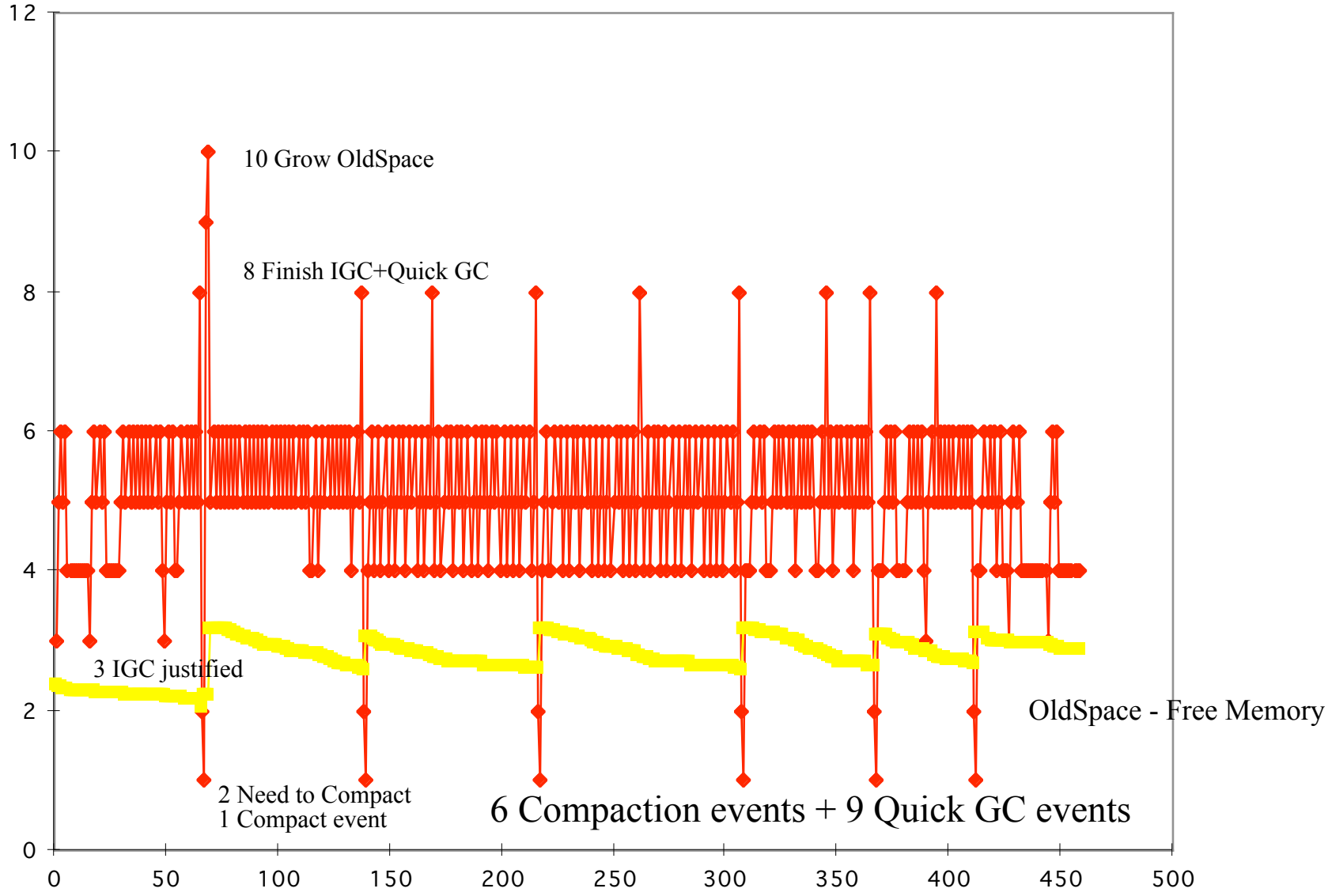
© copyright 1997-2004 John M McIntosh, all rights reserved.

Better IGC work loads, IGC always completes



© copyright 1997-2004 John M McIntosh, all rights reserved.

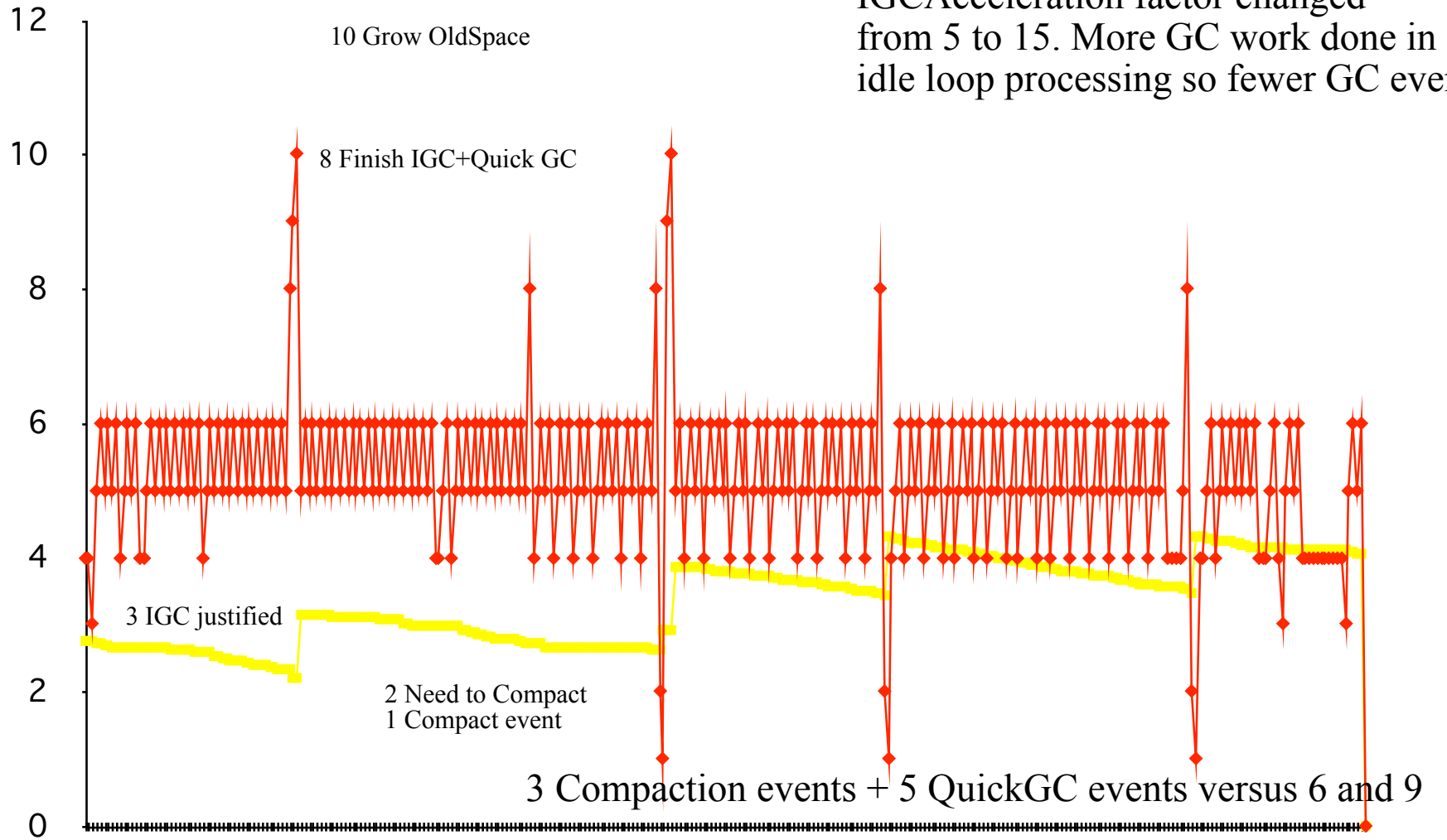
Case 1: User interaction test case before changes



© copyright 1997-2004 John M McIntosh, all rights reserved.

Case 1: User interaction test case after changes,

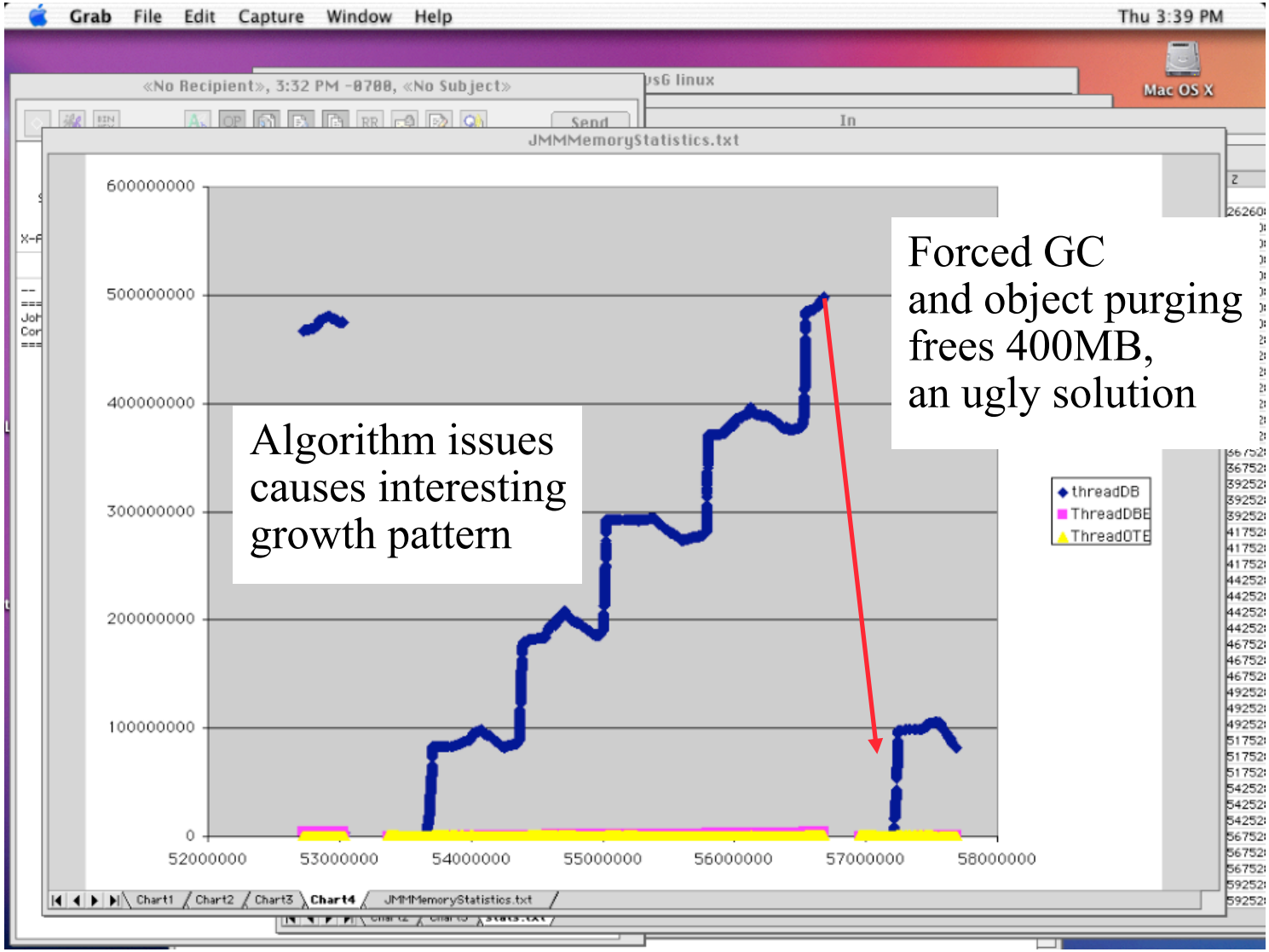
IGCAcceleration factor changed from 5 to 15. More GC work done in idle loop processing so fewer GC events



© copyright 1997-2004 John M McIntosh, all rights reserved.

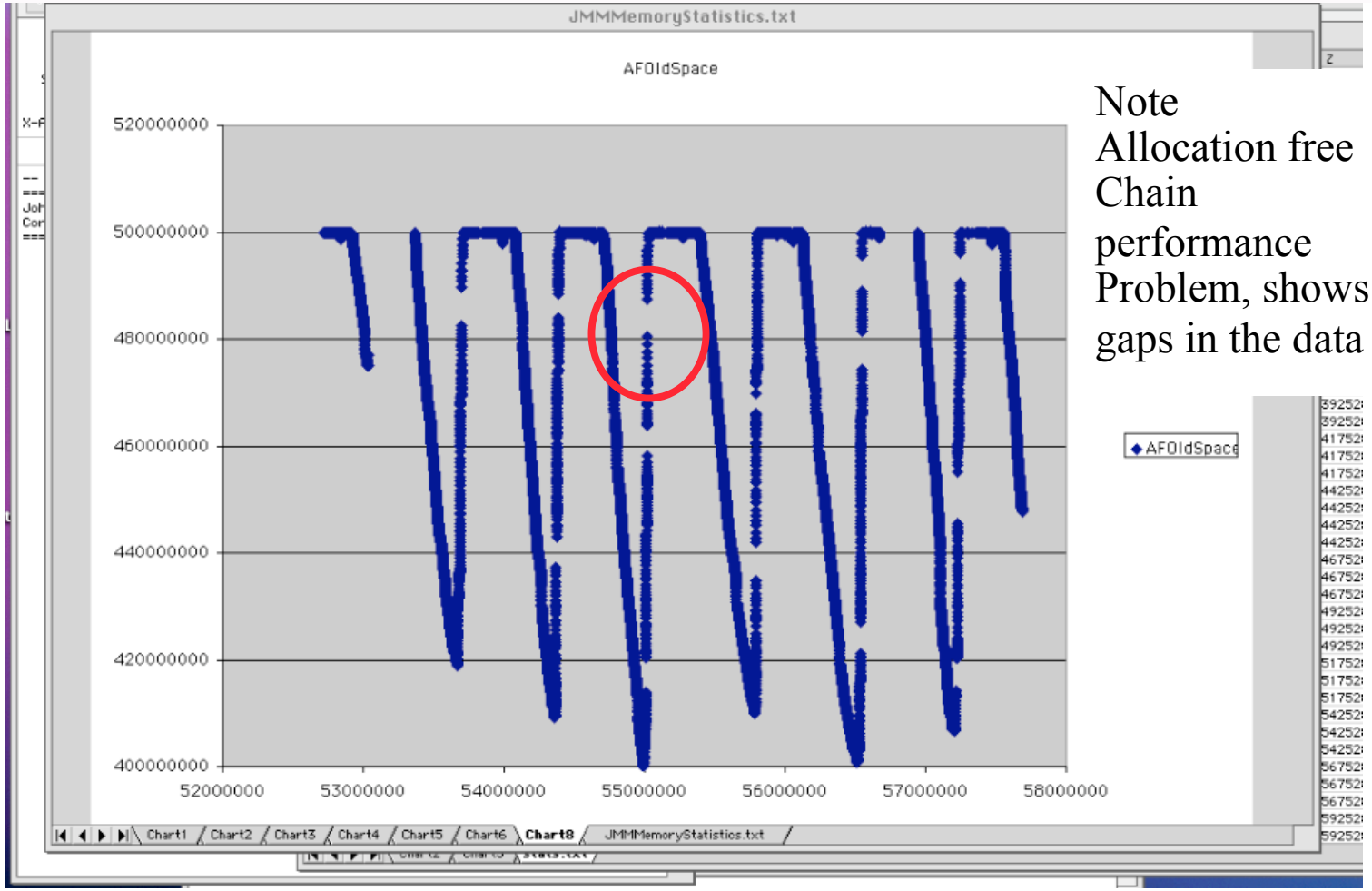
Old Case was ~460 seconds, we save 130 secs

500-800MB server application, issues with algorithms



© copyright 1997-2004 John M McIntosh, all rights reserved.

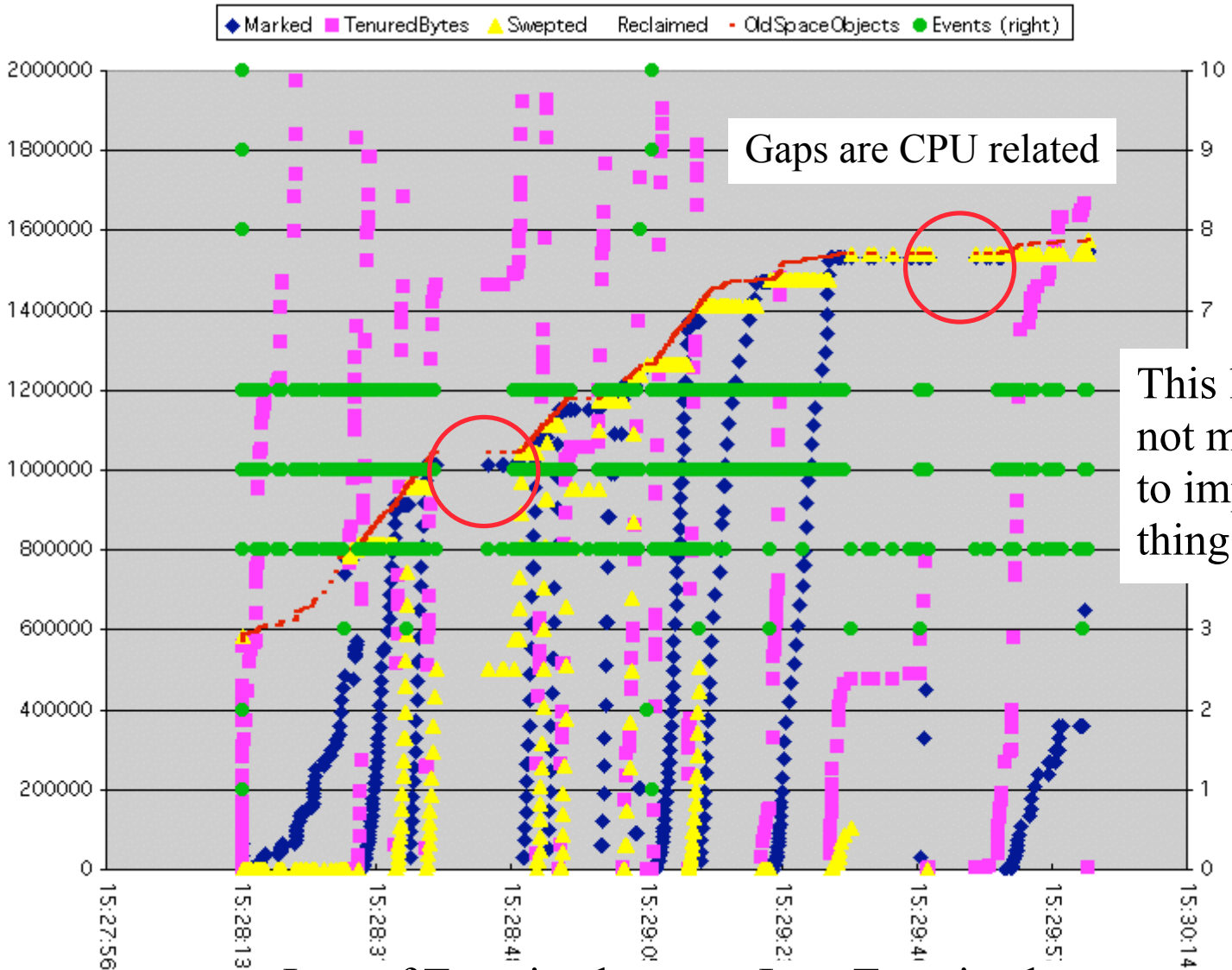
Old space is forced to shrink, then it grows!



Note Allocation free Chain performance Problem, shows as gaps in the data

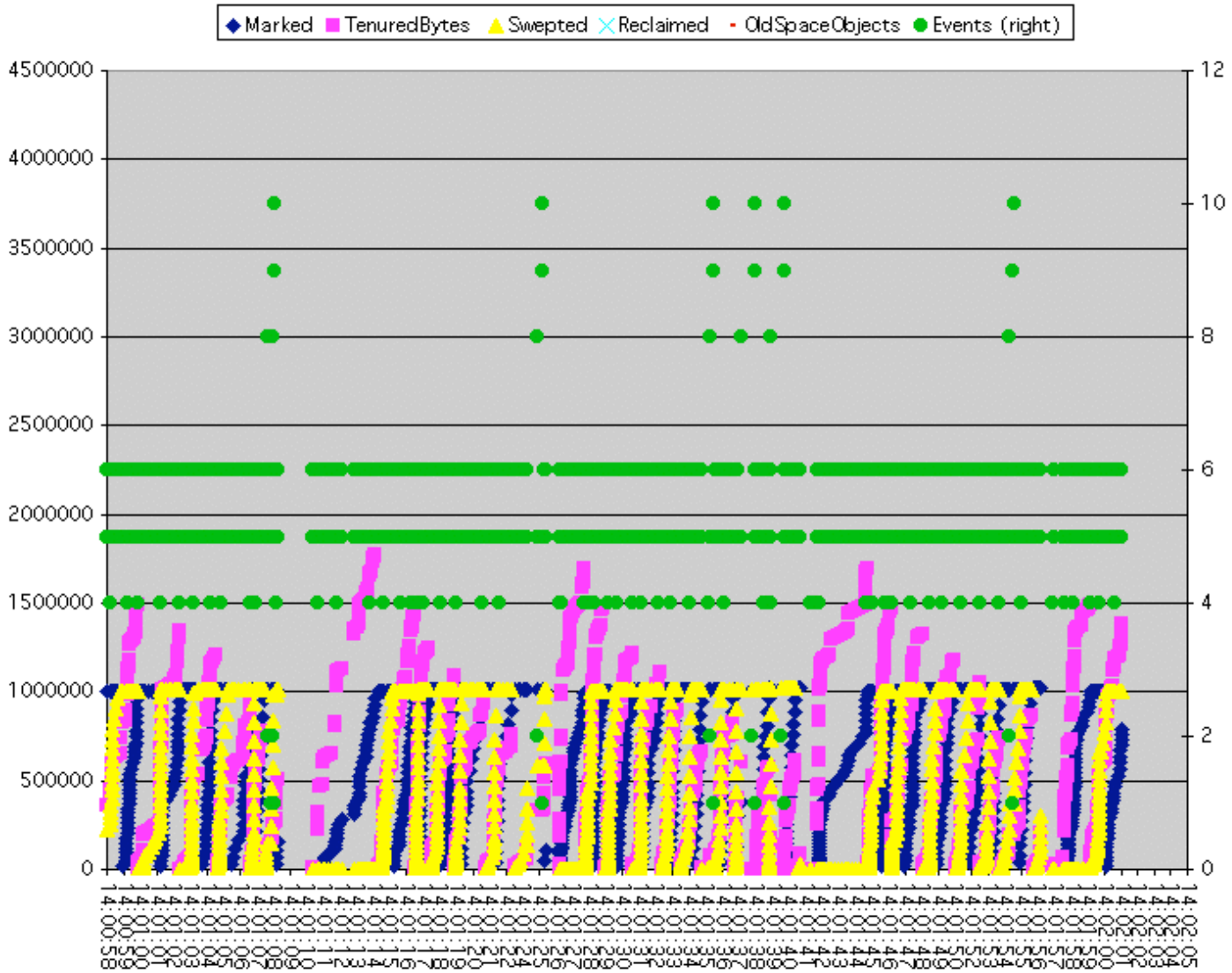
© copyright 1997-2004 John M McIntosh, all rights reserved.

Different VW 7.x 300 MB client application, note Mark/Sweep completions



© copyright 1997-2004 John M McIntosh, all rights reserved.

500 MB un-tuned vw7 server app, note IGC ceilings and excessive GC activity



© copyright 1997-2004 John M McIntosh, all rights reserved.

Mission Critical VW 3.x 400-500MB application
Before & After, shows what serious tuning can do

Before any tuning we saw:

Event	Total
1 Compact	33
2 Compact Need	6
3 Full ILIGC	46
4 Idle Loop	260
5 Low Space	1921
6 Run IGC	1793
7 Request Grow	129
10 Grow	129
Grand Total	4317

After tuning effort:

Event	Total
3 Full ILIGC	10
4 Idle Loop	177
5 Low Space	1224
6 Run IGC	1224
Grand Total	2635

Zero code tuning was done. Application improved a few % for each SUnit test case. Keyboard responsiveness became “snappy”. Got rid of mystery lock-ups, a core dump (or two), and reduced the number of GC cursor events.

© copyright 1997-2004 John M McIntosh, all rights reserved.

Garbage Collection in Smalltalk

* By John M McIntosh
  - Corporate Smalltalk Consulting Ltd.
  - http://www.smalltalkconsulting.com
  - johnmci@smalltalkconsulting.com

    * *Maintainer of the Squeak Macintosh VM.*
    * *TK4 VM support {Ask me later about TK4}*
    * *Trip reports for OOPSLA, Camp Smalltalk, etc.*

For ESUG 2004

© copyright 1997-2004 John M McIntosh, all rights reserved.