Isla Vista Heap Sizing:

Using Feedback to Avoid Paging

Chris Grzegorczyk
Sunil Soman
Rich Wolski
Chandra Krintz

Dept. of Computer Science
University of California, Santa Barbara

State of Affairs

- Managed Runtime Environment (MRE)
  - ↳Garbage Collector (GC): automatic memory management
  - ↳Manage heap storage: reclaim dead, unreachable objects
  - ↳Impacts application execution performance
- Operating System (OS)
  - ↳Memory Manager (MM): automatic storage allocation
  - ↳Arbitrate allocation of physical memory to competing apps
  - ↳Reclaim pages unlikely to be used soon
- Observation: MM & GC have diff. objectives
  - ↳Potential for conflict and negative impact on application

Questions

- When do GC and MM conflict?
  - ↳ What GC actions cause degraded performance?
    - Large heaps result in **page faults** during GC
  - ↳ What level of performance degradation results from conflict?
    - Page faulting can be **dominating factor** in application performance
- Can controlling heap size alleviate conflict?
- An additional constraint: Non-intrusiveness
  - ↳ MM and GC are complex and sensitive to changes
    - Critical to **stability and performance** of OS/MRE
  - ↳ Simplicity and portability result in practical impact

3

Avoid or Cooperate?

- Who is responsible for heap residency?

1. Cooperation: Intertwine MM and GC [Hertz PLDI05]
    - Communicate page out/in events to GC
    - Try to keep heap resident by freeing pages
    - Track connectivity of swapped pages, avoid them during full GC

2. Avoidance: MRE avoids swapping [Yang ISMM04,Yang OSDI06]
    - Modify memory manager to support approx real mem. availability
    - Use available memory info to resize heap

4

Outline

- Understanding GC Memory Access Patterns
  - ↳ Visualizing the working set of GC
  - ↳ Identifying GC and heap sizing triggers
- Reclaiming Pages in the MM
  - ↳ Handling a Memory Shortfall
- Solution: Isla Vista Heap Sizing
  - ↳ Heap Resizing using MM events as Feedback
  - ↳ Measuring the cost of GC induced Paging
  - ↳ Evaluating IV Heap Sizing
  - ↳ Examining the Non-Intrusive Design

5

GC Mem. Access Patterns

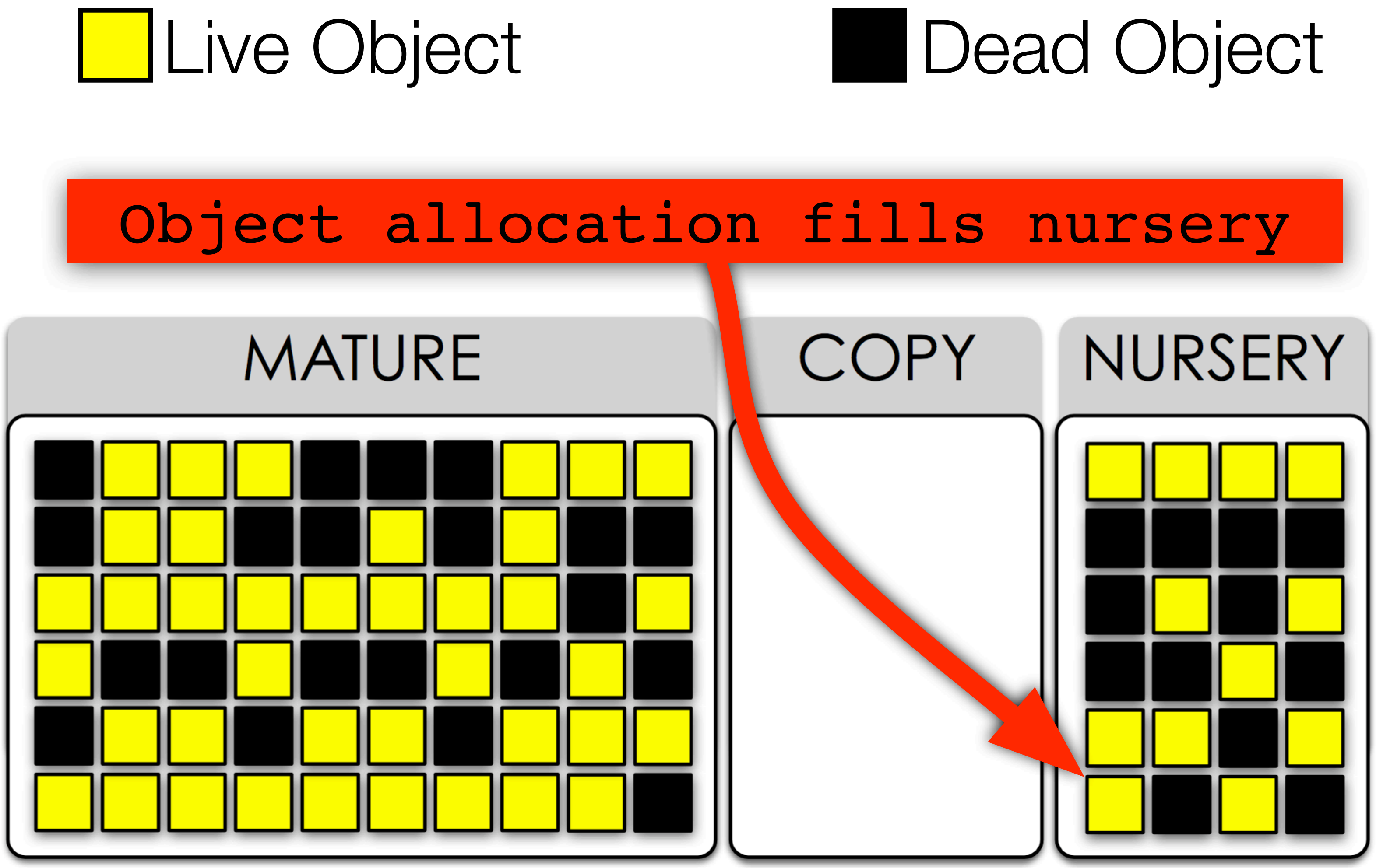
- GenMS: Gen. Mark-Sweep [Blackburn ICSE04]
  - ↳ Heap divided into 3 spaces:
    - **Nursery**: allocate objects here
    - **Copy**: used for copying live objects during GC
    - **Mature**: objects having survived more than one GC, “old”
  - ↳ Two kinds of collections:
    - **Nursery GC**: copying collection of nursery into copy
    - **Full GC**: marking traversal of live mature, then sweep dead
- Why GenMS? Best GC in JikesRVM

6

GC Mem. Access Patterns

- GenMS: Gen. Mark-Sweep [Blackburn ICSE04]
  - ↳ Heap divided into 3 spaces:
    - **Nursery**: allocate objects here
    - **Copy**: used for copying live objects during GC
    - **Mature**: objects having survived more than one GC, “old”
  - ↳ Two kinds of collections:
    - **Nursery GC**: copying collection of nursery into copy
    - **Full GC**: marking traversal of live mature, then sweep dead
- Why GenMS? Best GC in JikesRVM

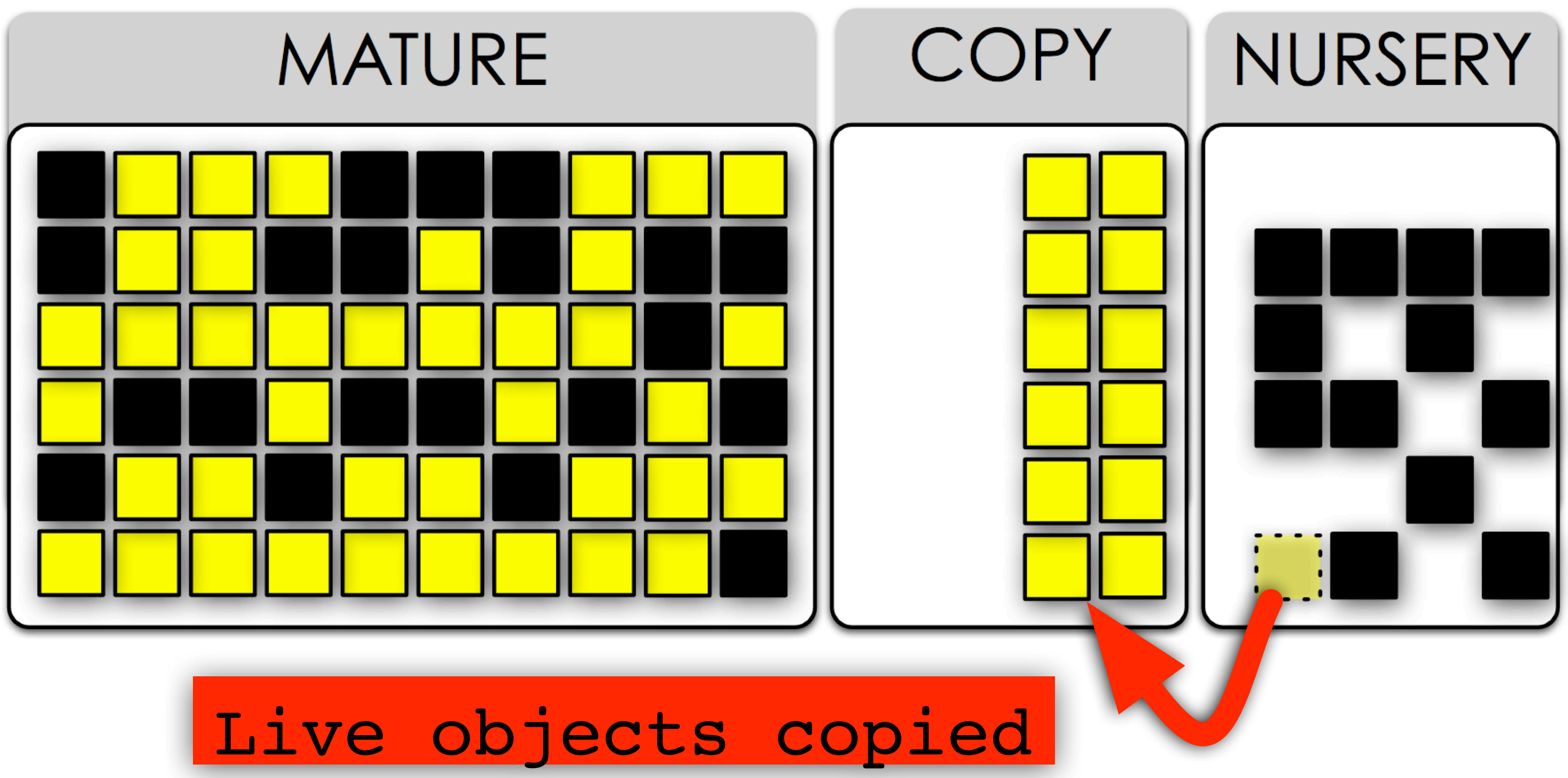
Nursery GC


Live Object
Dead Object
Object allocation fills nursery
MATURE
COPY
NURSERY

8

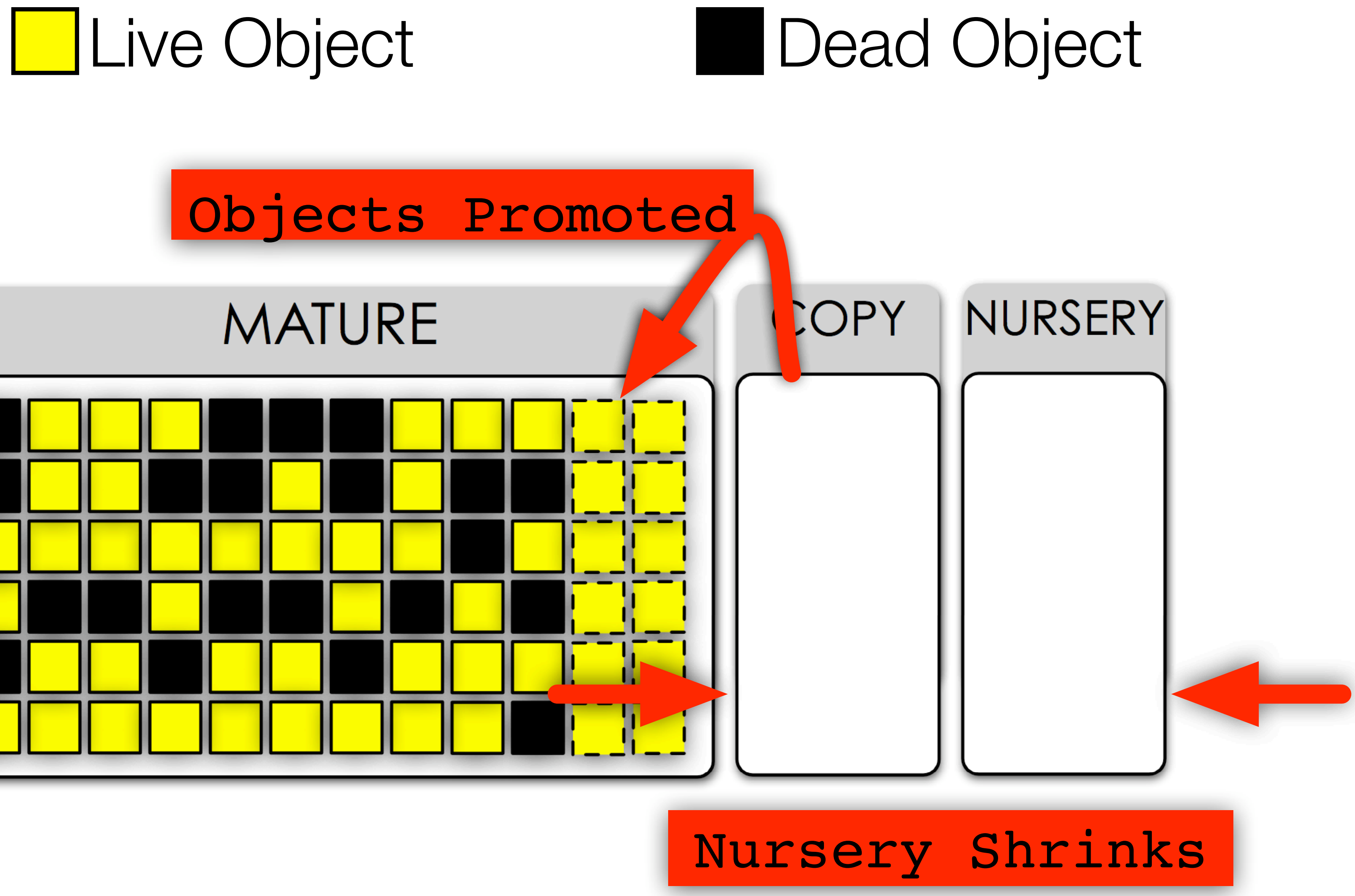
Nursery GC: Copy

Live Object   Dead Object



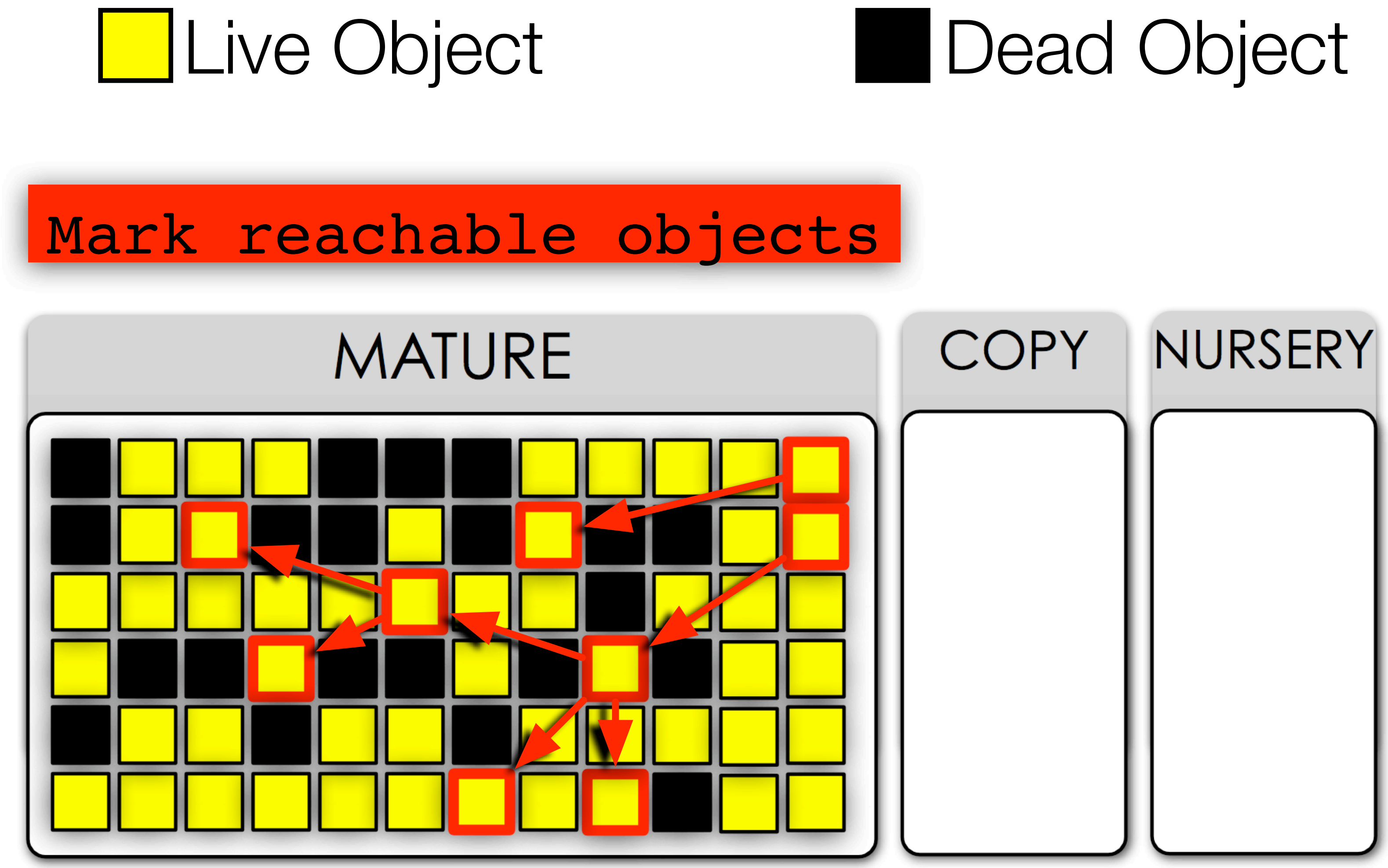
9

Nursery GC: Promotion



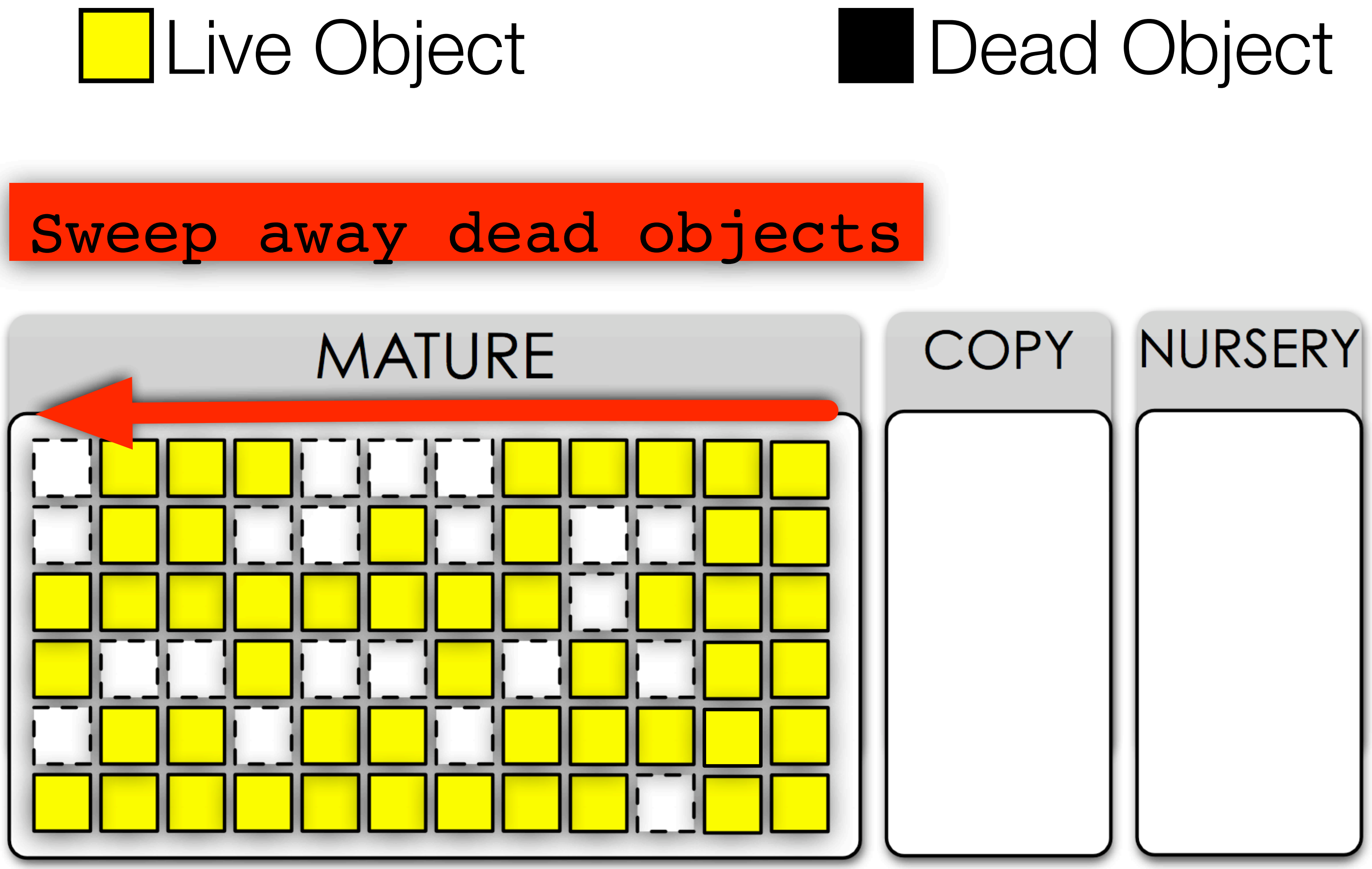
10

Full GC: Mark



11

Full GC: Sweep



12

Reclaiming Pages in MM

- Page Replacement: 2Q [Johnson VLDB94]
  - ↳ Why? Linux 2.6.16.9 uses approx. 2Q
  - ↳ Maintains lists to track recency of reference: Active, Inactive
    - **Active**: recently used, hot pages
    - **Inactive:** candidates for reclaim (i.e. swap out, free)

- Tracks fast simple performance counters
    - **2Q**: page [de]activations, page in/out, inactive refills, allocation stalls
    - **Demand Paging**: major/minor faults

13

Reclaiming Pages in MM

- MM triggered by memory shortfall
  - ↳ Two thresholds for number of free pages
    - **low**: starting to run out of memory for allocations
    - **min**: allocations stopped, page laundering required to allocate

- How pages are reclaimed:
  1. Refill inactive list
     - unreferenced pages from active list
  2. Scan inactive list
     - free unused or start I/O
  3. Scan inactive list *again*
     - wait for I/O to finish

14

Handling a Shortfall

- Two Possible Reclaim Paths:

1. **Passive Reclaim**: free < `low`
   - Wake up `kswapd`, kernel thread for freeing memory
   - `kswapd` **asynchronously** flushes pages to disk (`pageout`)

2. **Direct Reclaim**: free < `min`
   - Allocations halted until free > `low`
   - Trying to allocate memory executes reclaim code (`allocstall`)
   - Process does the work of `kswapd` **synchronously** (`pageoutS`)

- Recall: Full GC touches **all** heap pages

15

Solution: IV Heap Sizing

- Objective: Profitably trade GC for page faults
  - ↳ Shrink heap inducing GC; avoiding paging
- **`allocstall`** as predictor of future GC-induced paging
- Controlling heap size in JikesRVM
  - ↳ Determined by 1 variable: heap size (=mature+copy+nursery)
  - ↳ Resize the heap after each GC, including nursery
- Isla Vista Heap Sizing
  - ↳ Sample feedback only during GC
    - No **`allocstall`**: Grow heap linearly
    - **`allocstall`**: Shrink heap multiplicatively

16

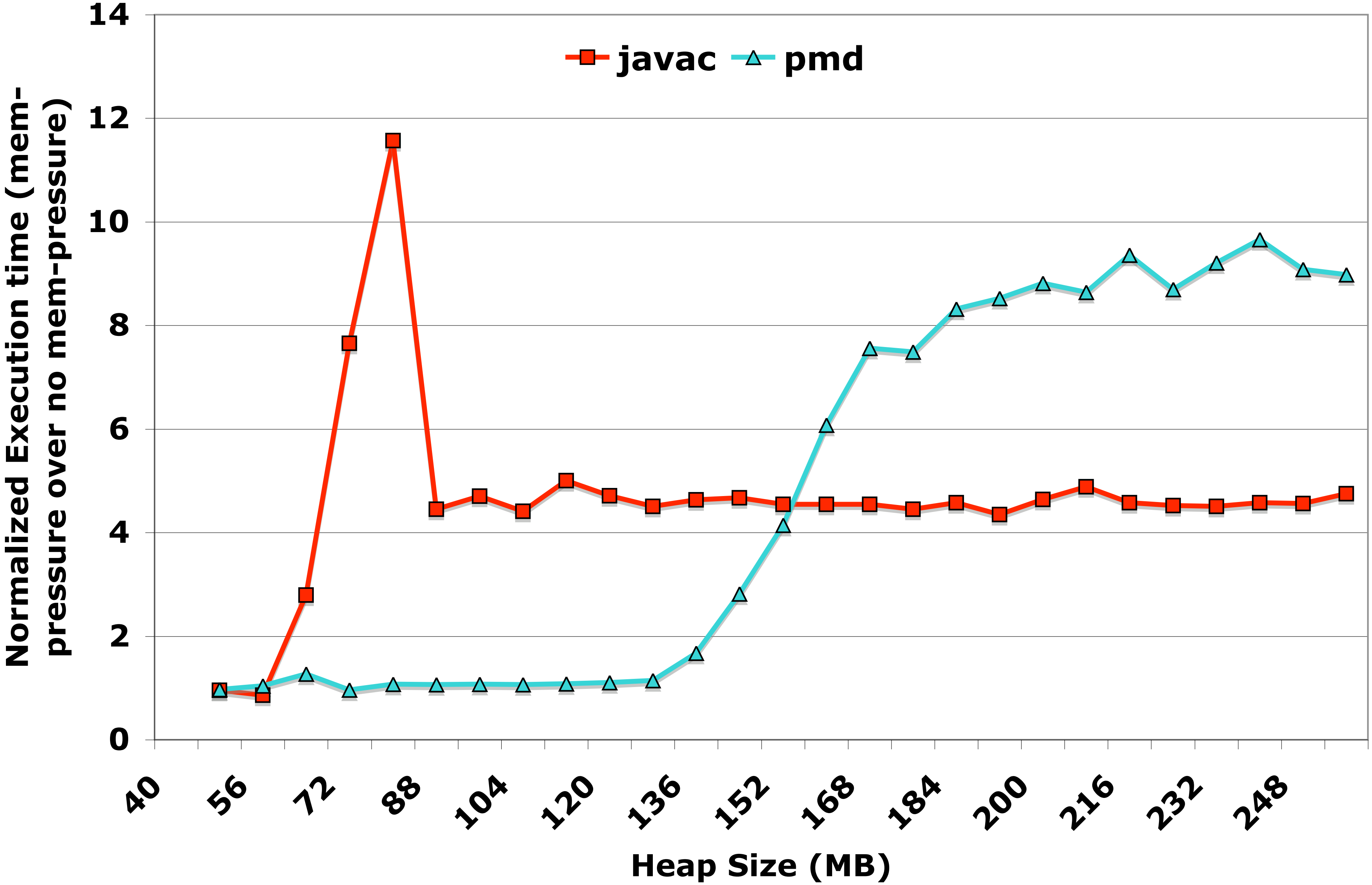
Evaluating IV Heap Sizing

- Evaluation Platform
  - ↳ Current & standard: Linux 2.6.16.9, JikesRVM CVS~2.4.6
  - ↳ SPECjvm98, dacapo, and SPECjbb2000
  - ↳ Hardware: 3.2Ghz Xeon, 896MB RAM, 5GB swap, 4k pages
  - ↳ Heap Sizes: 40MB-256MB
  - ↳ Induce mem. pressure w/ mlock(): 640MB(pmd),736MB(javac)

- Magnitude of GC induced paging
- How well does IV Heap Sizing mitigate problem
  - ↳ **Baseline (JikesRVM):** with pressure, and without pressure
  - ↳ **IV Heap Sizing:** with pressure, and without pressure

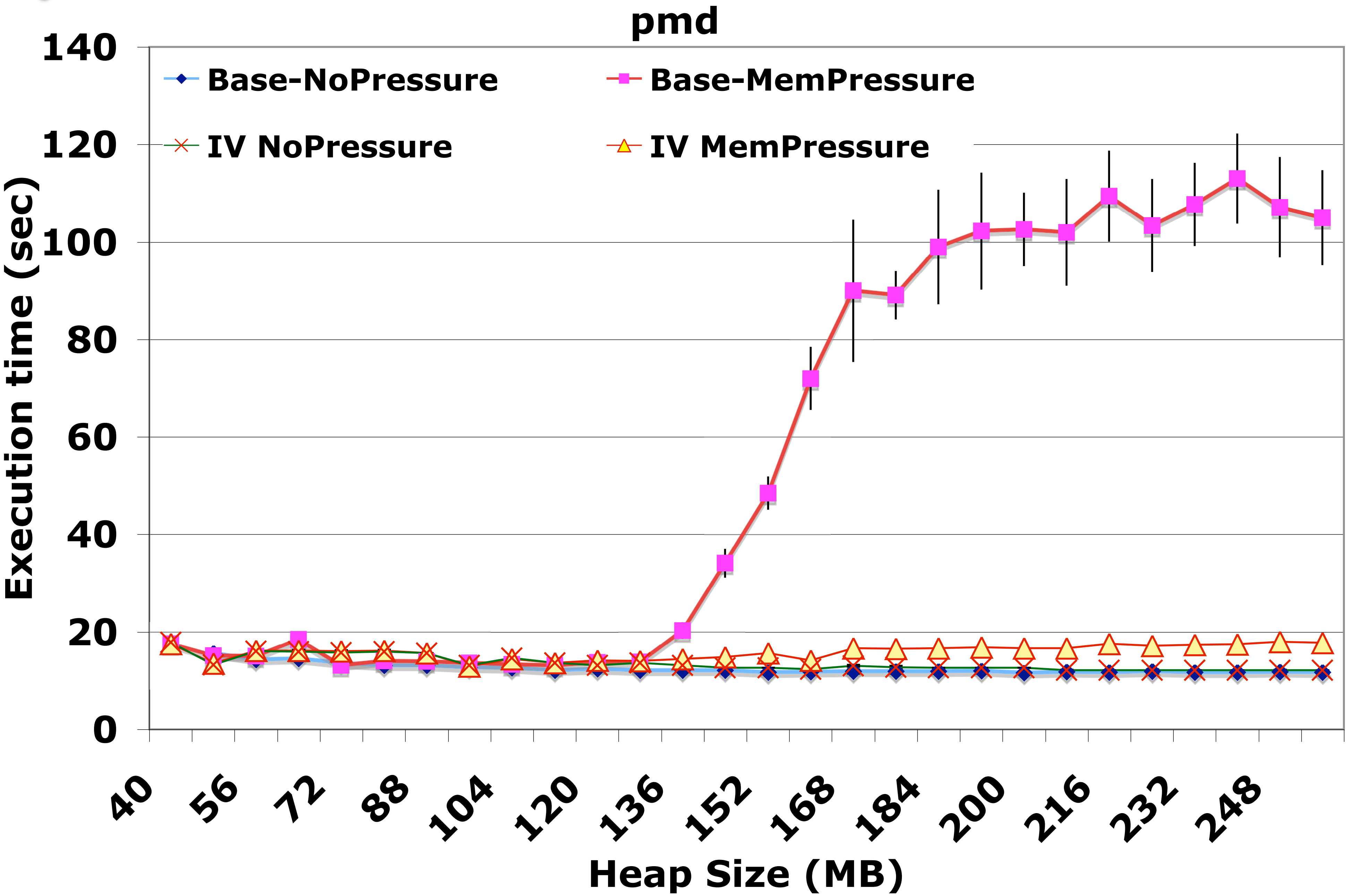
17

GC-induced Paging



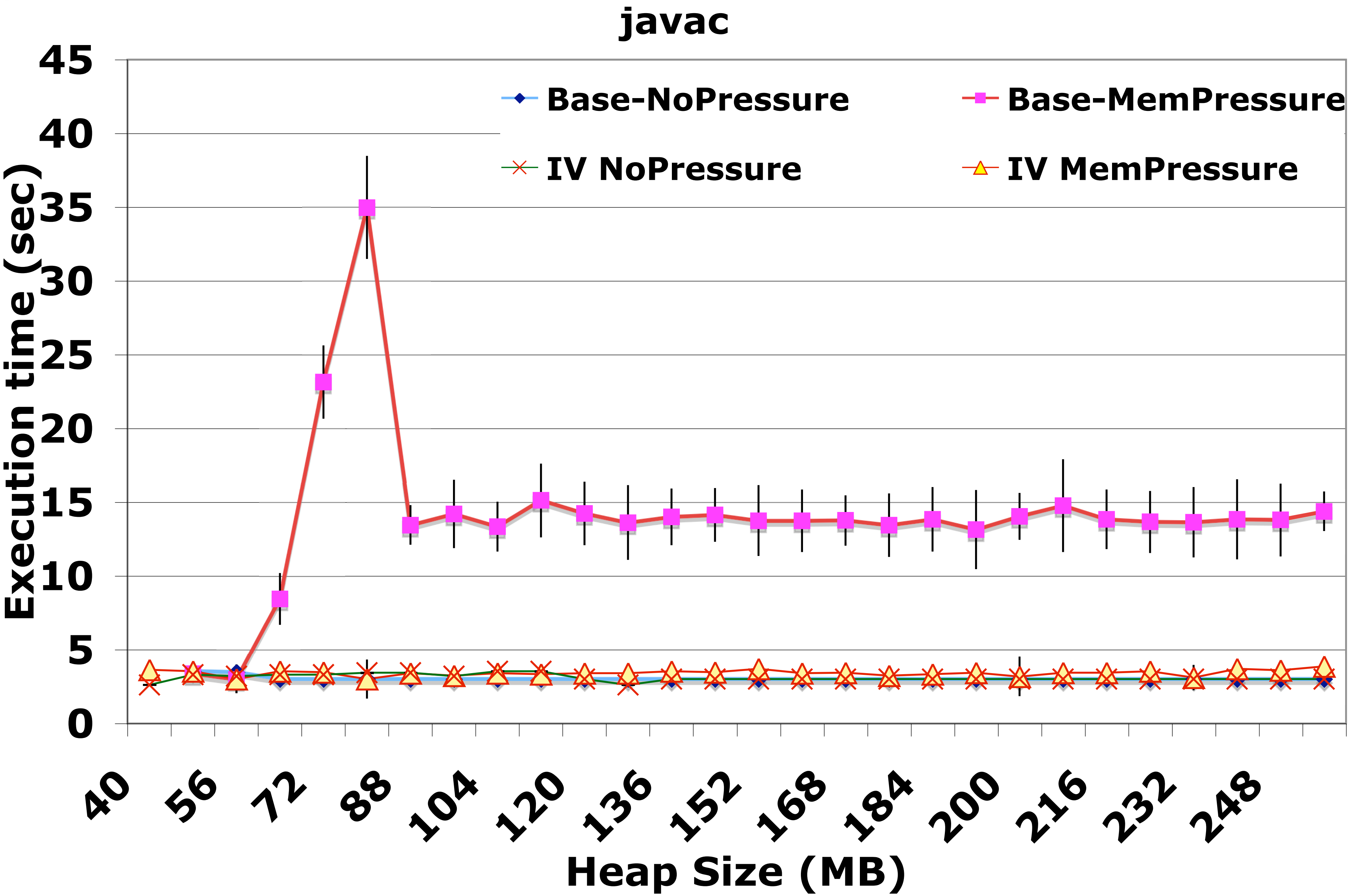
18

pmd: execution time



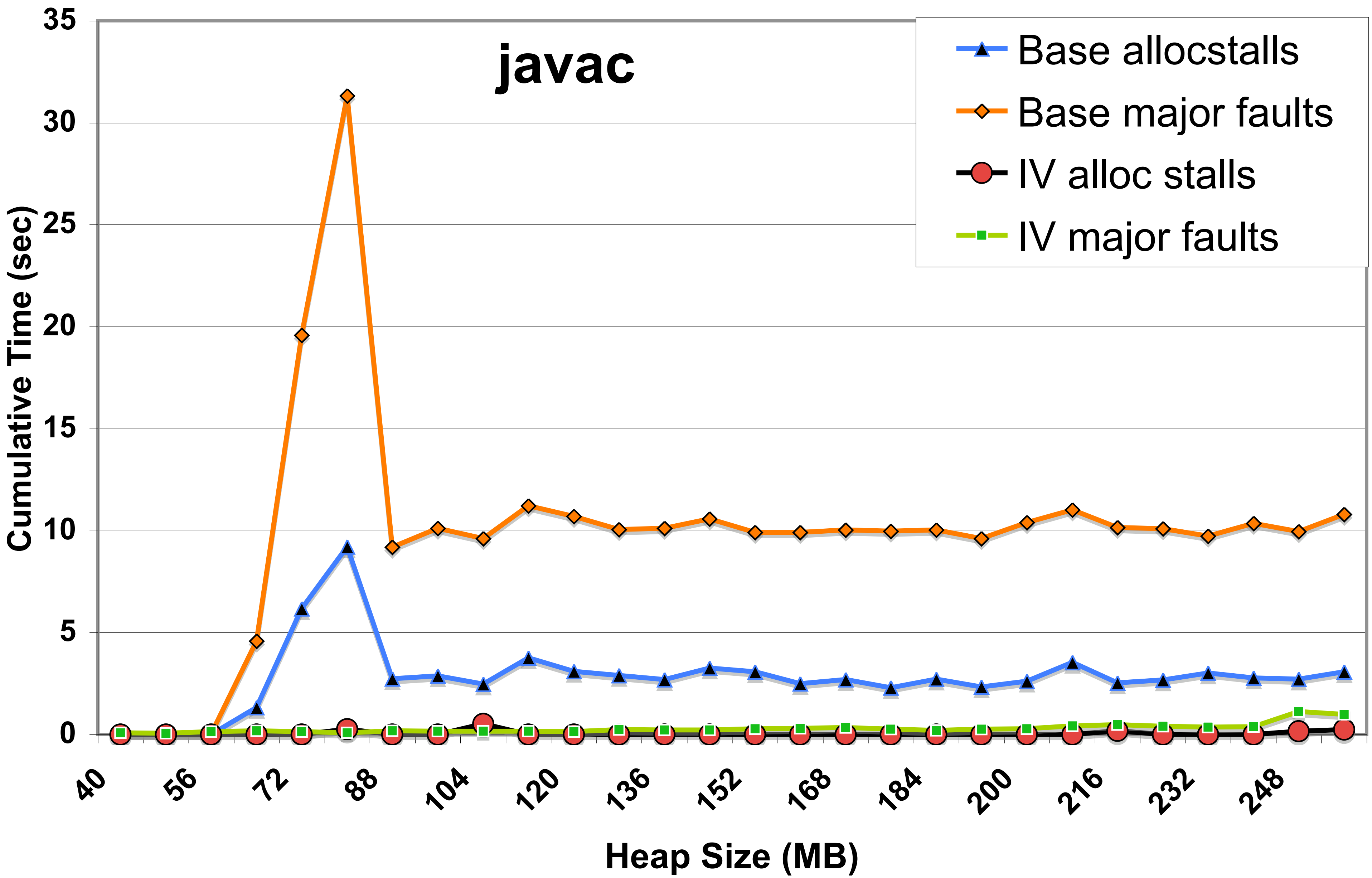
19

javac: execution time



20

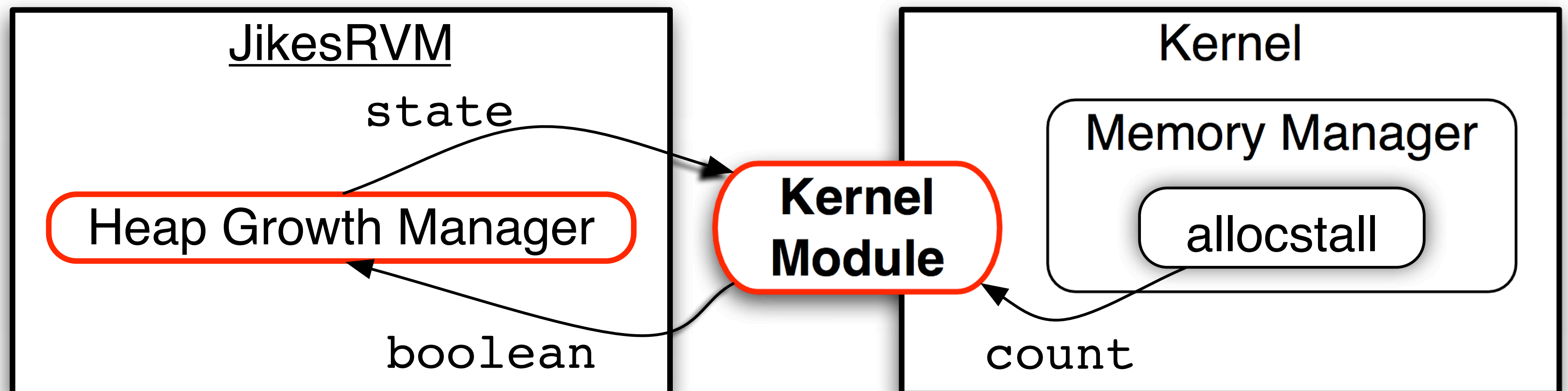
javac: allocstalls and faults



21

Design of IV Heap Sizing

- **Linux**: Insert kernel module
  - ↳Exports `allocstall` feedback
  - ↳No need to modify MM, ensures integrity
- **JikesRVM**: Modify Heap Sizing Policy
  - ↳Heap Sizing after every GC using feedback
  - ↳No need for new GC algorithm



22

Conclusions

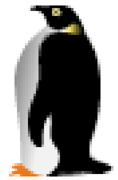
- Beneficial trade: GCs for Page Faults
  - ↳ Trigger GC by resizing the heap
  - ↳ Improves performance under memory pressure
  - ↳ Doesn't hurt when there is no pressure
- `allocstall`: effective predictor GC page faults
  - ↳ Best performing given constraints
- Non-Intrusive Design
  - ↳ No changes to the Memory Manager
  - ↳ No changes to the Garbage Collector

23

Conclusions

- Beneficial trade: GCs for Page Faults
  - ↳ Trigger GC by resizing the heap
  - ↳ Improves performance under memory pressure
  - ↳ Doesn't hurt when there is no pressure
- `allocstall`: effective predictor GC page faults
  - ↳ Best performing given constraints
- Non-Intrusive Design
  - ↳ No changes to the Memory Manager
  - ↳ No changes to the Garbage Collector
- **Source Code available under GPL**

24

Questions? 

Chris Grzegorczyk

grze@cs.ucsb.edu

http://www.cs.ucsb.edu/~grze/ivhs

25