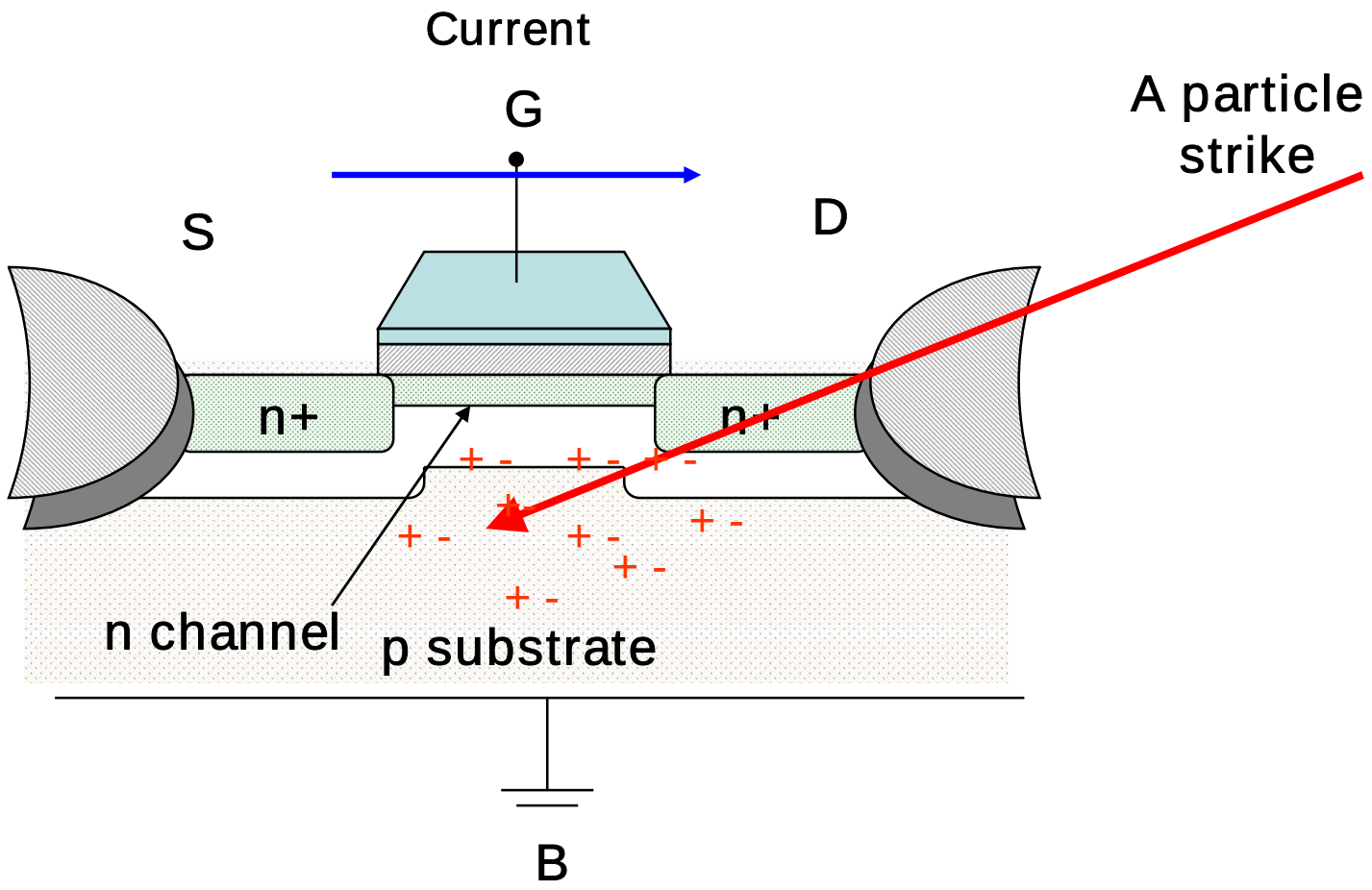
Object Duplication for Improving Reliability

G. Chen, G. Chen, M. Kandemir, N. Vijaykrishnan, M. J. Irwin

Department of Computer Science and Engineering

The Pennsylvania State University

Soft Errors



Soft errors or transient errors are circuit errors caused due to excess charge carriers induced primarily by external radiations

Soft Errors

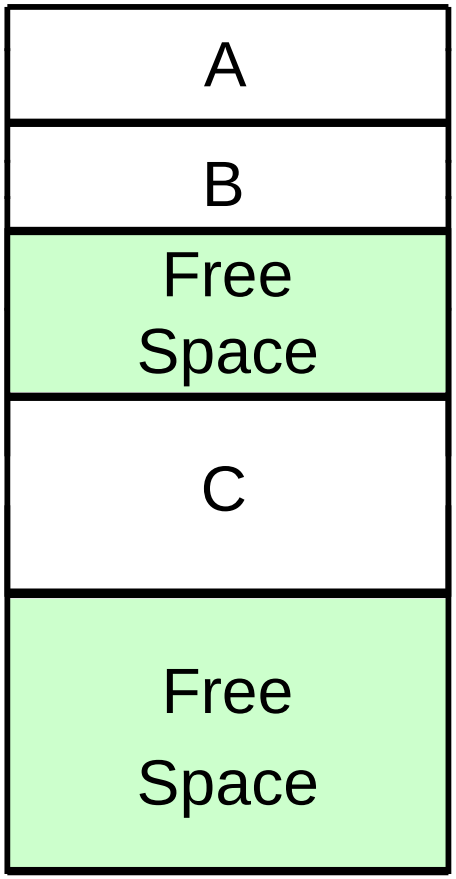
- Why soft errors become problematic?
  - As CMOS device sizes decrease, the charge stored at each node decreases
  - Potentially leads to a much higher rate of soft errors
  - Low-power operating modes accentuates the soft error problem

Challenges for Soft Error Protection

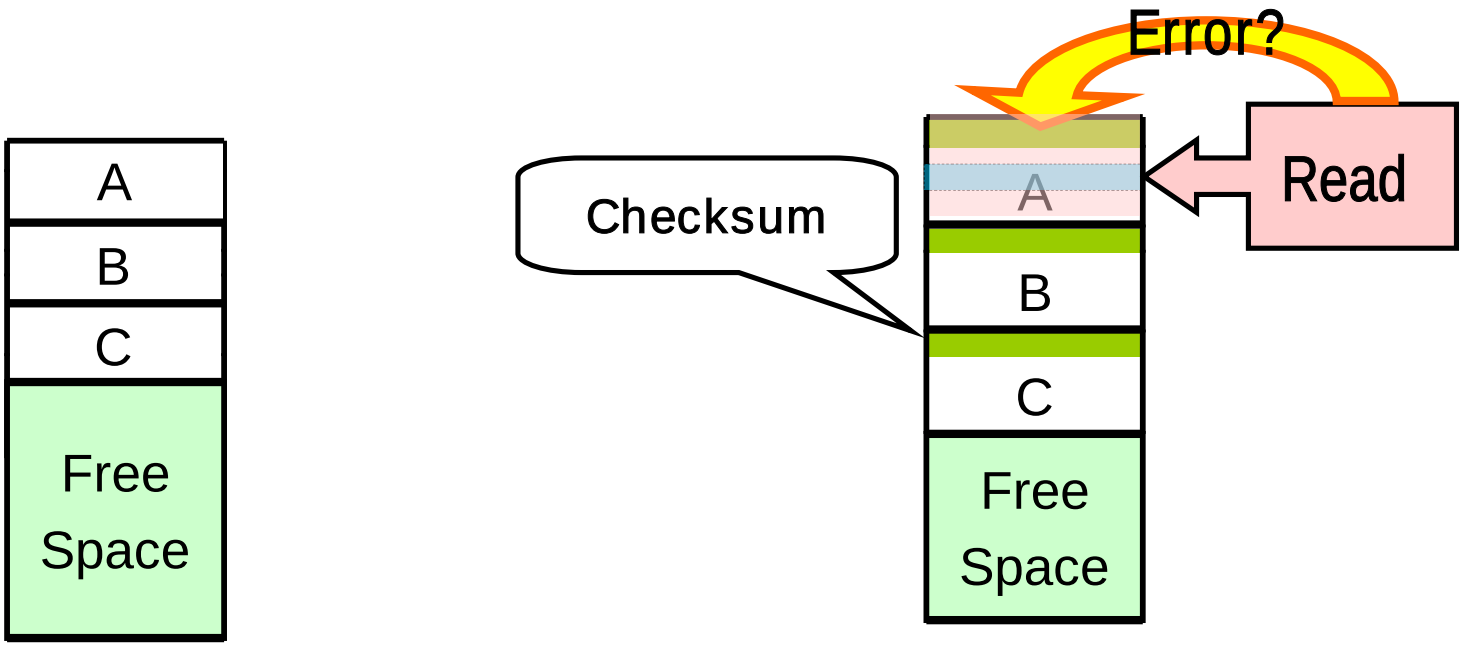
- Balance several factors
  - Cost
    - Extra hardware required
  - Flexibility
    - Different requirements of multiple applications
  - Overhead
    - Performance, power, memory space
  - Reliability

Java Virtual Machine (JVM)

- Java is widely used in embedded devices
- JVM has details about application's status and semantics
  - error detection and recovery may be easier
- Heap is the dominant space consumer in JVM
  - We focus on heap objects

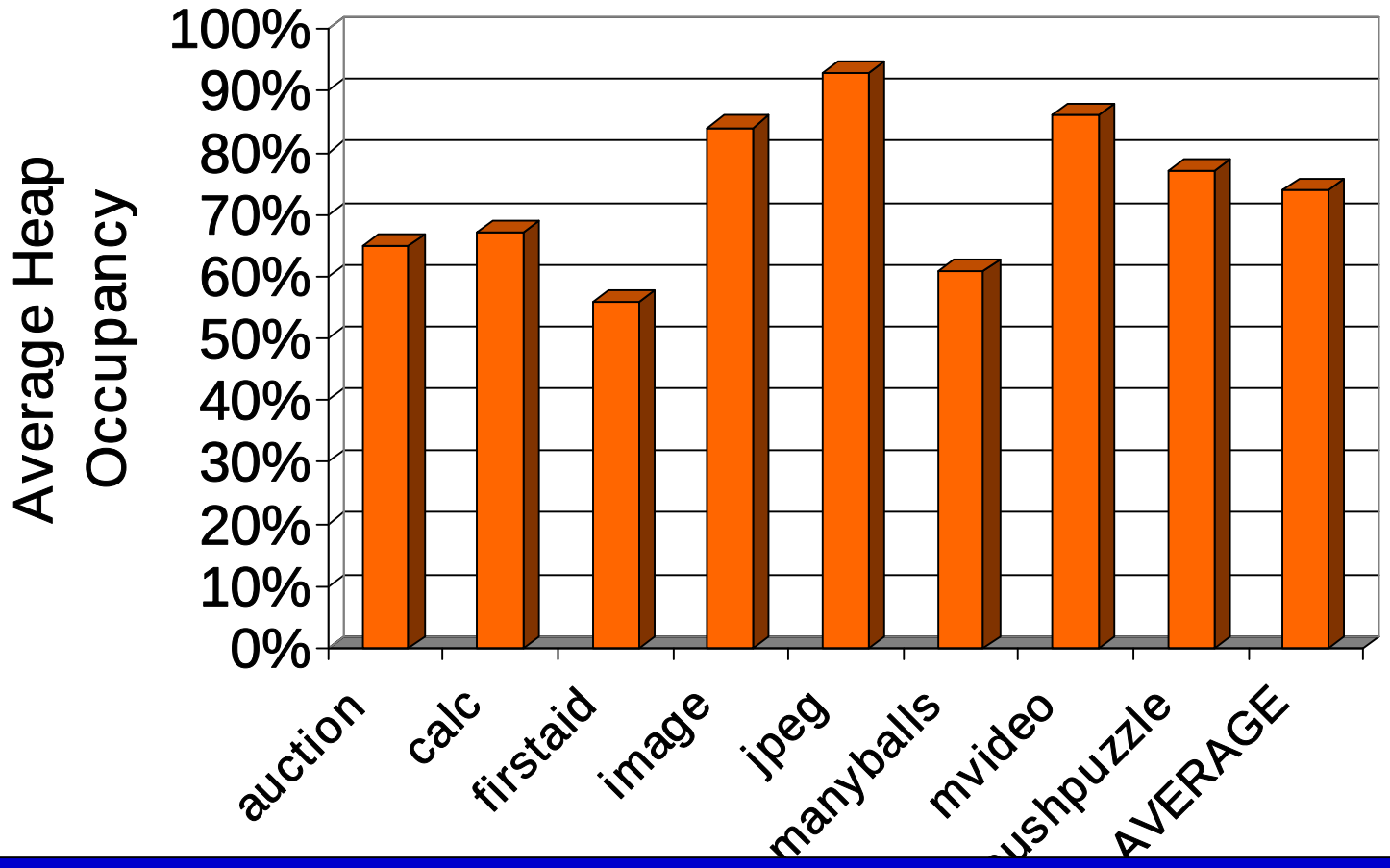


Checksum-based Schemes



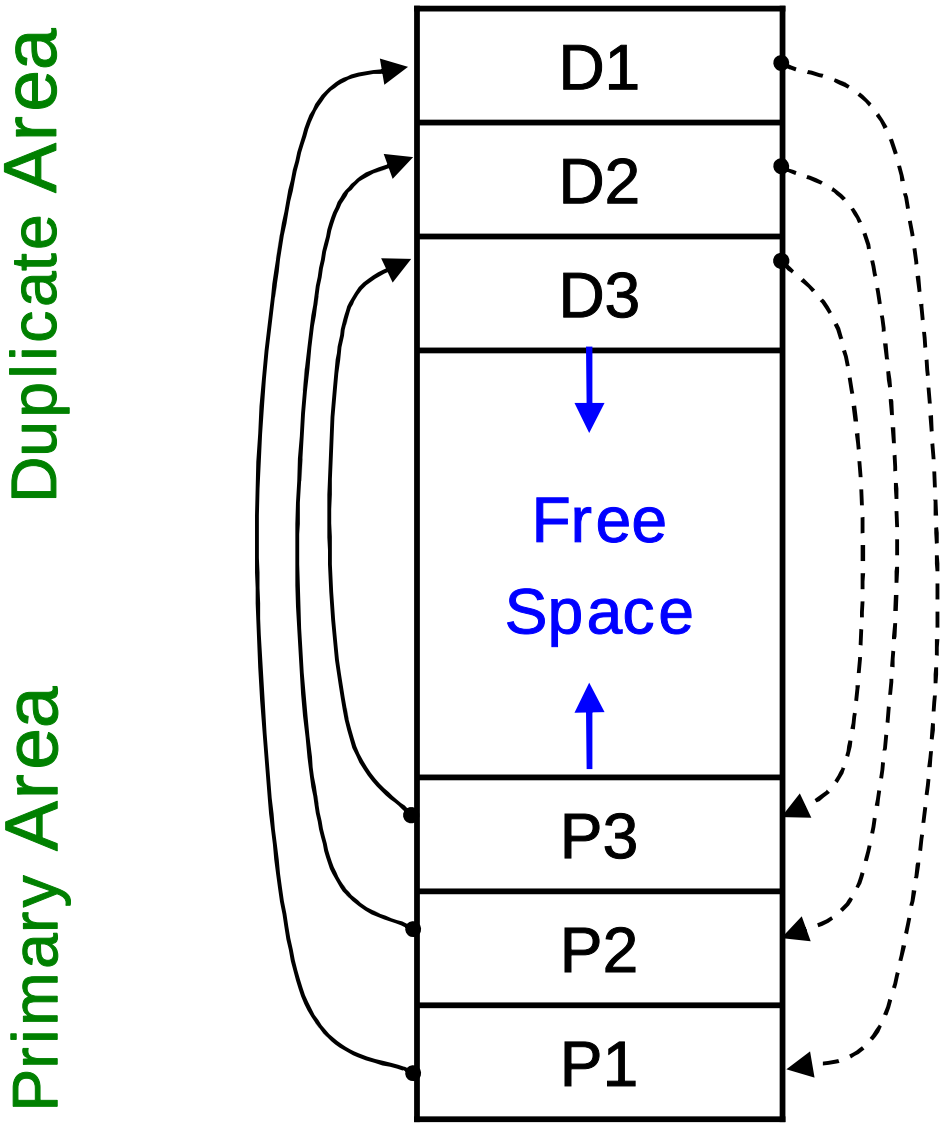
- A checksum for each field or each object
- Updated at write operations
- Checking is required for each read operation
- Error detection, but no error correction

Motivation for Object Duplication



Remaining heap can be used for other purposes, or example, storing object duplicates

Object Duplication

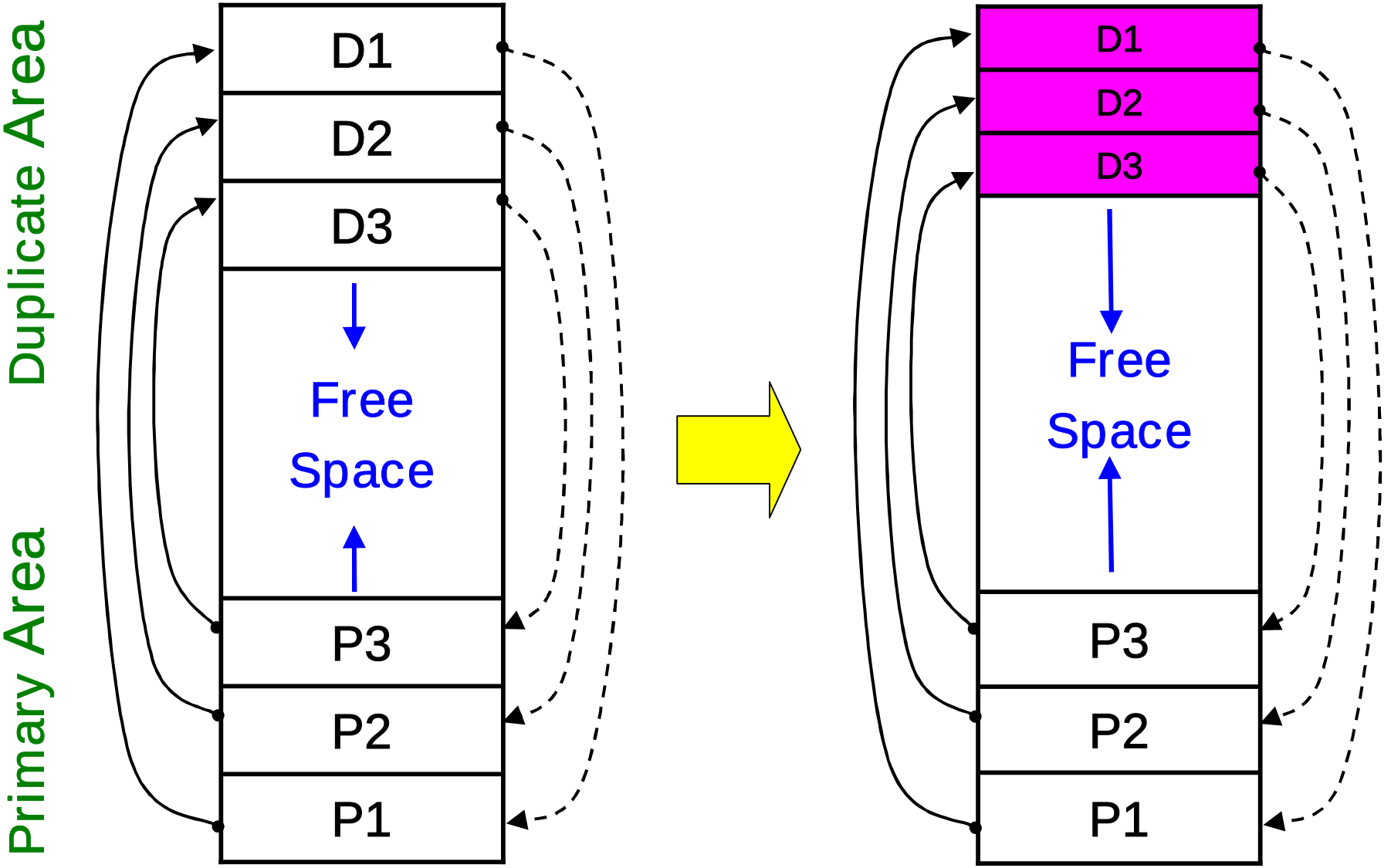


- Duplicate is updated on each write
- Duplicate is read when there is an error in the primary

Full Duplication

- Performance overhead
  - Extra accesses on writes
  - Read accesses when error happens
  - Cache conflicts
  - Performance overhead is very low
- Memory space requirement doubled
- Two ways to reduce memory space overhead
  - Compression
  - Selective duplication

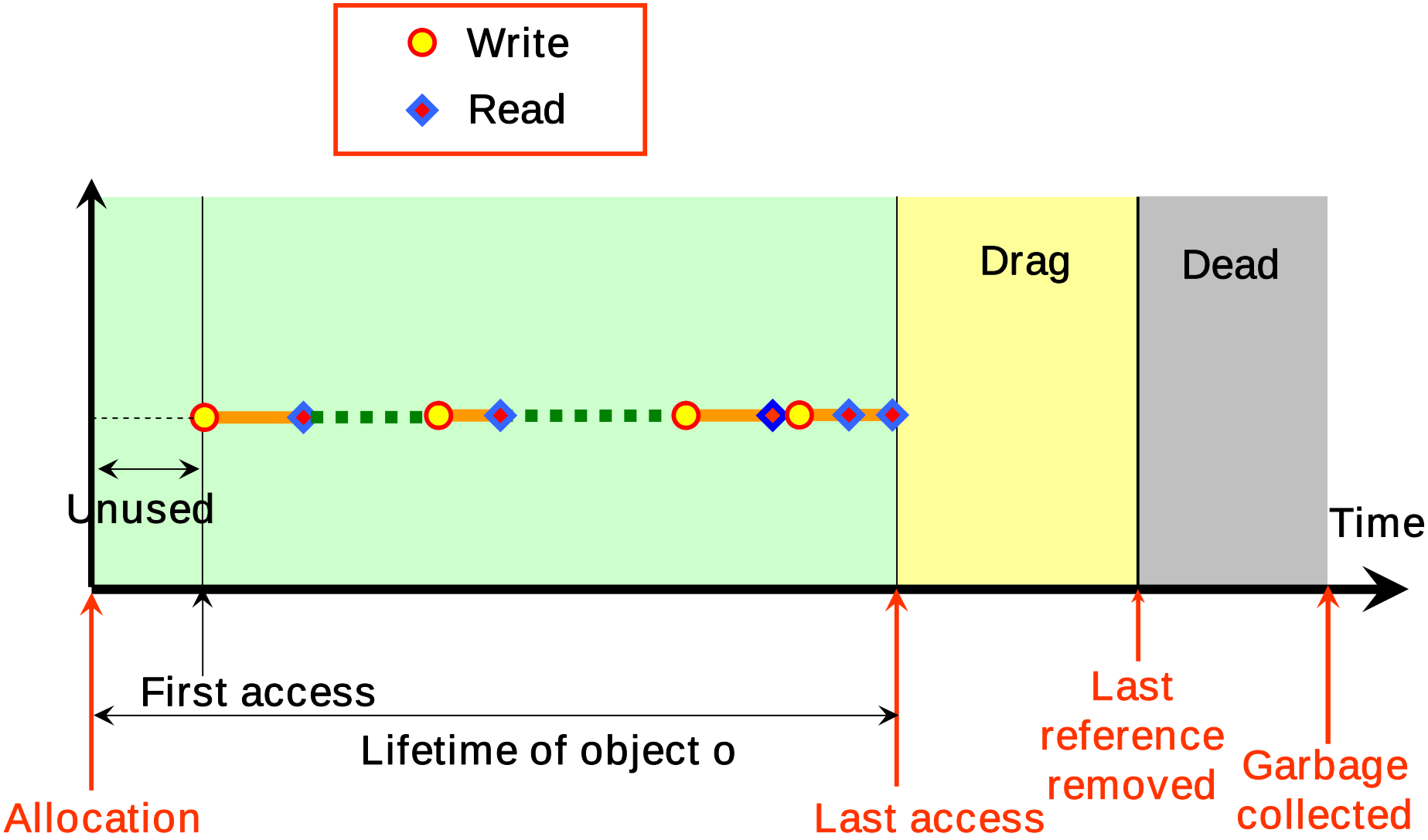
Compression Based Duplication



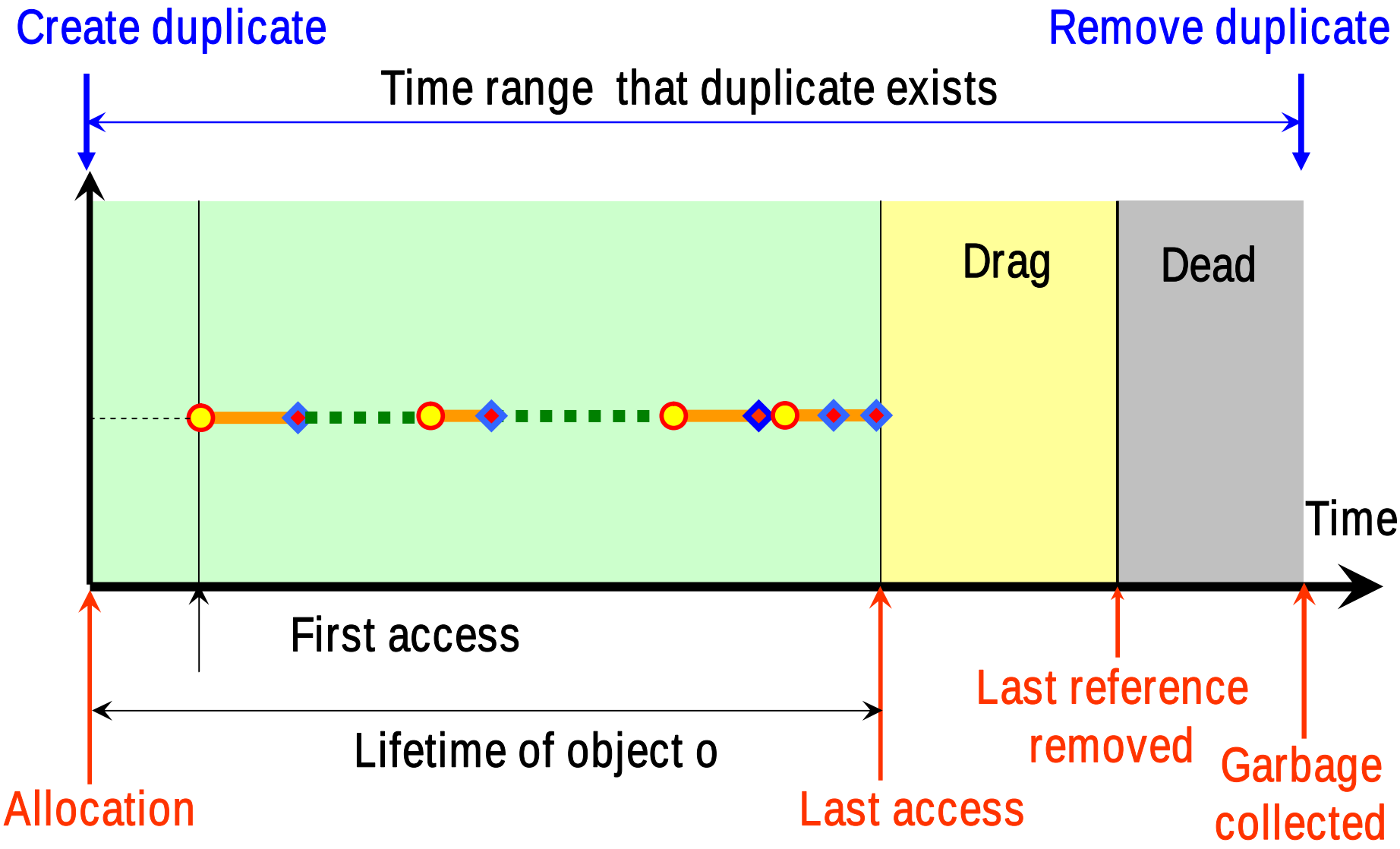
Compression Based Duplication

- All objects have compressed duplicates
- Performance overhead
  - Compression on writes
  - Decompression when primary copy corrupted
  - Little overhead if the object is rarely updated
- Zero-removal compression/decompression
  - Fast
  - Java objects contains a lot of zero bytes

Life Cycle of an Object



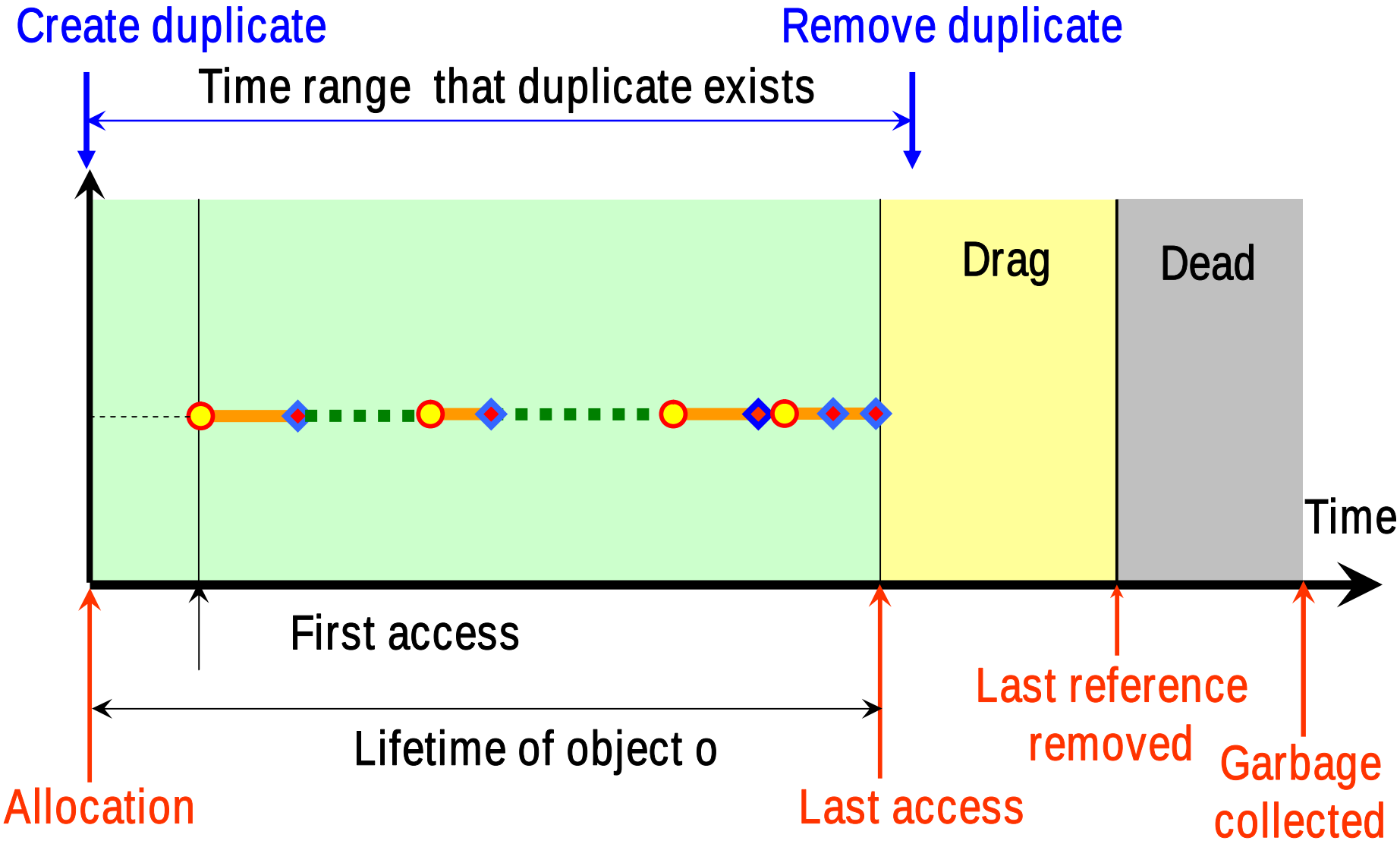
Full Duplication



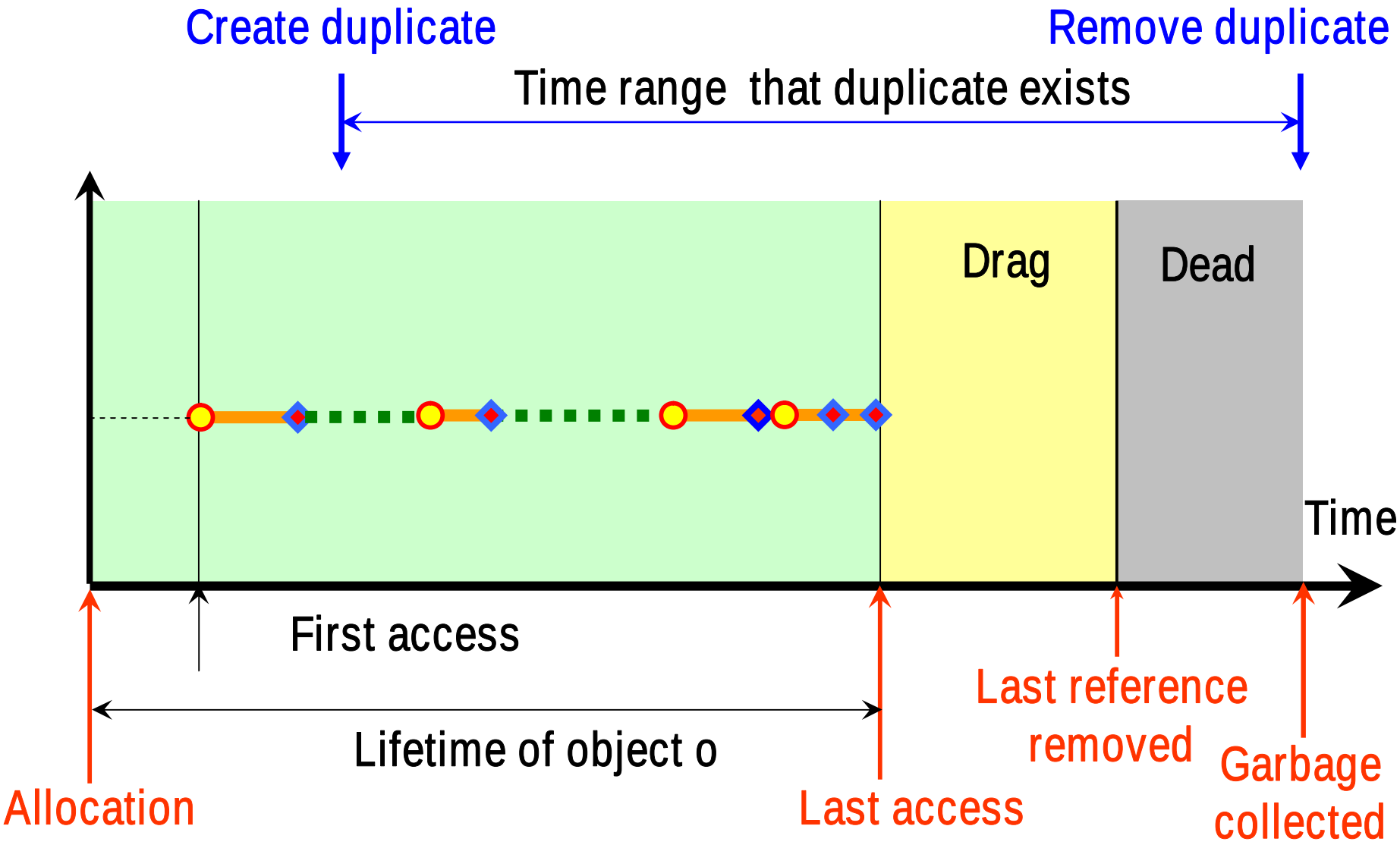
Selective Duplication

- Early termination based duplication
- Lazy duplication
- Allocation site based

Early Termination Based



Lazy Duplication



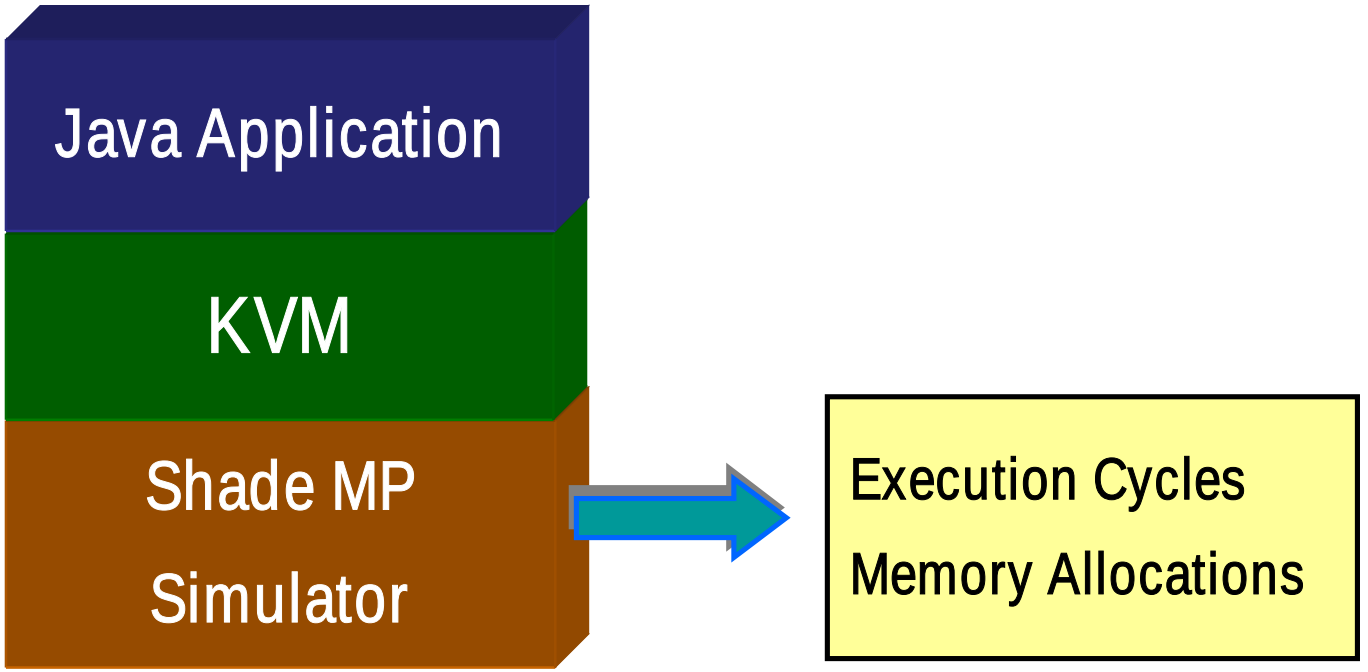
Allocation Site Based

- Based on profiling
- Sort allocation sites according to the access frequency of the object accessed
  - Create duplicate at allocation sites leading to higher access frequency
  - No duplicate creation at allocation sites leading to lower access frequency

Discussion of Selective Schemes

- Reliability is lower than full duplication
- Object life time
  - Early termination based scheme favors short-living objects
  - Lazy duplication and allocation site based schemes favor long-living objects
- Static vs dynamic
  - Early termination based and lazy duplication schemes are dynamic
  - Allocation site based scheme is static

Experiment Setup

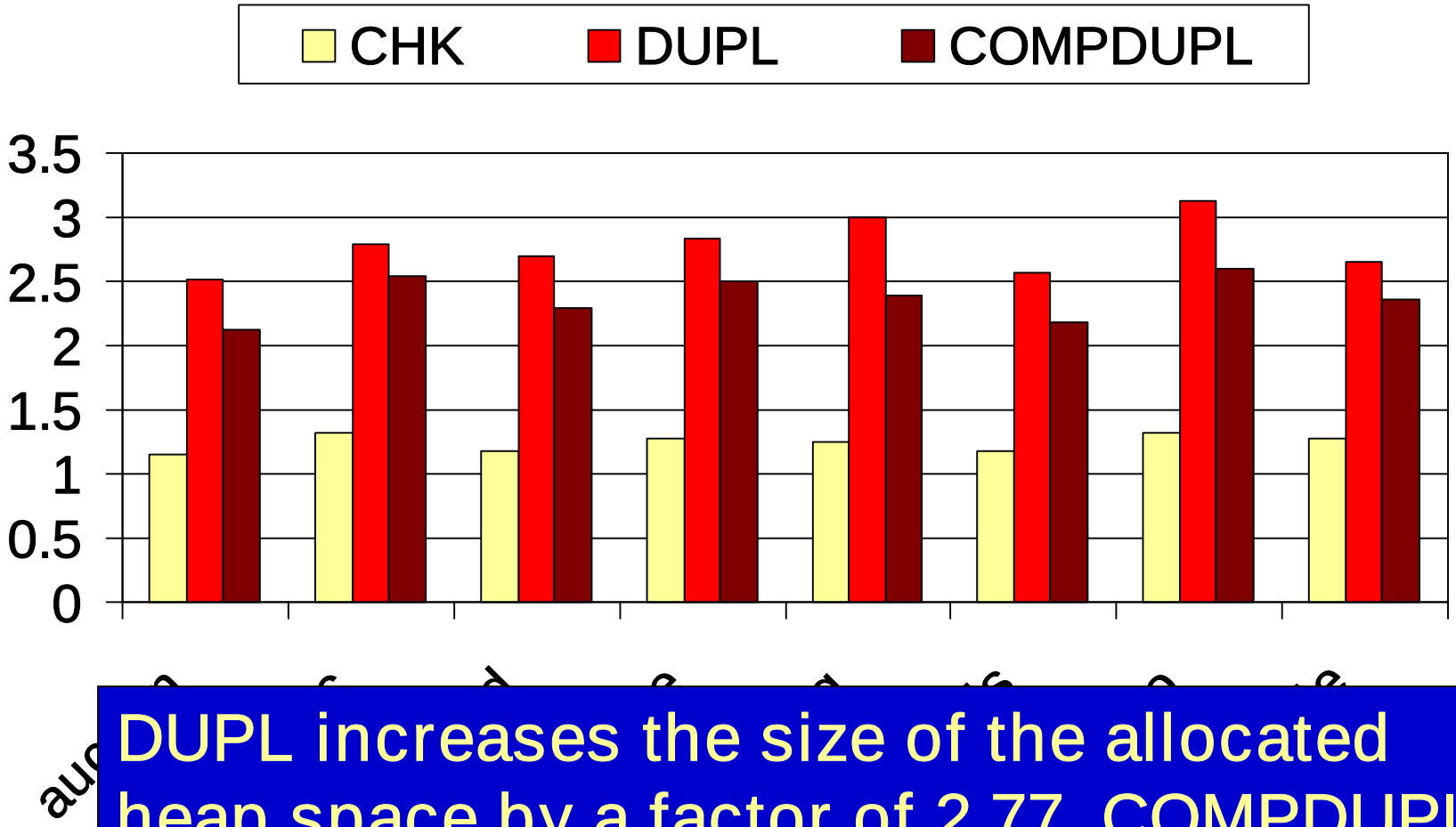


- Three types of statistics
  - Heap space
  - Error resilience
  - Performance

Benchmarks

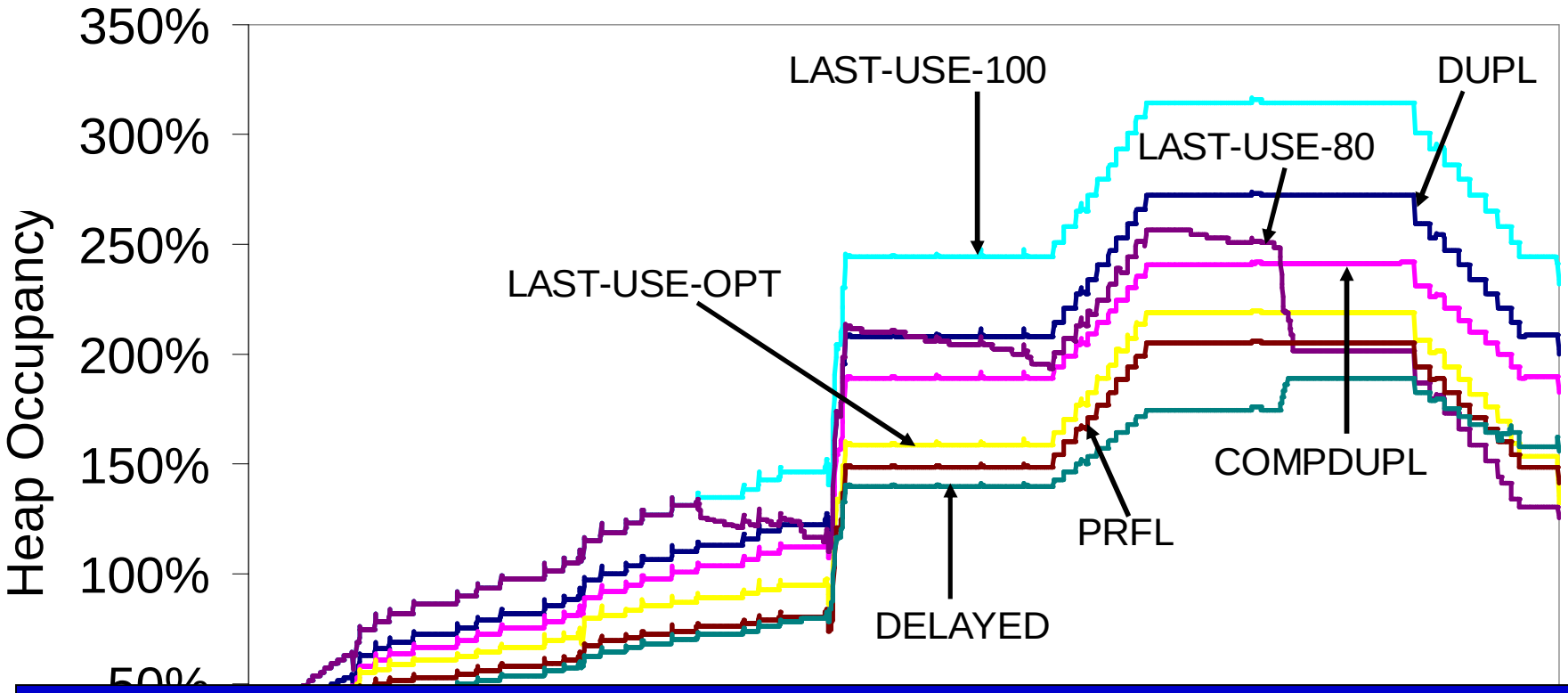
Benchmark	Description
auction	Ticket auction
calc	Calculator
firstaid	Firstaid information
jpeg	Jpeg viewer
image	Photo album
manyballs	Bounding balls
mvideo	Video player
pushpuzzle	Puzzle game

Allocated Heap Space



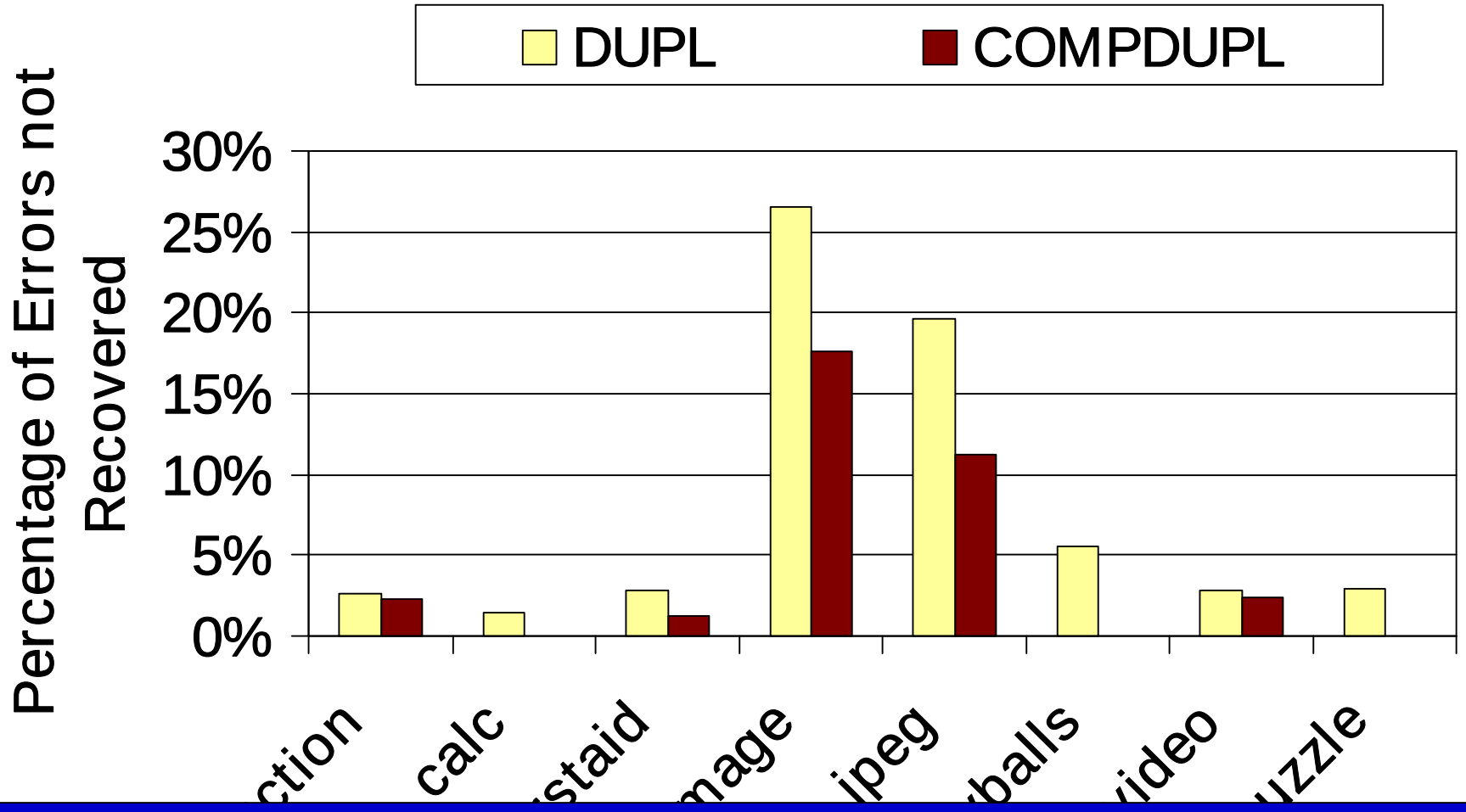
DUPL increases the size of the allocated heap space by a factor of 2.77. COMPDUPL reduces it to 2.37.

Heap Occupancy for Selective Schemes



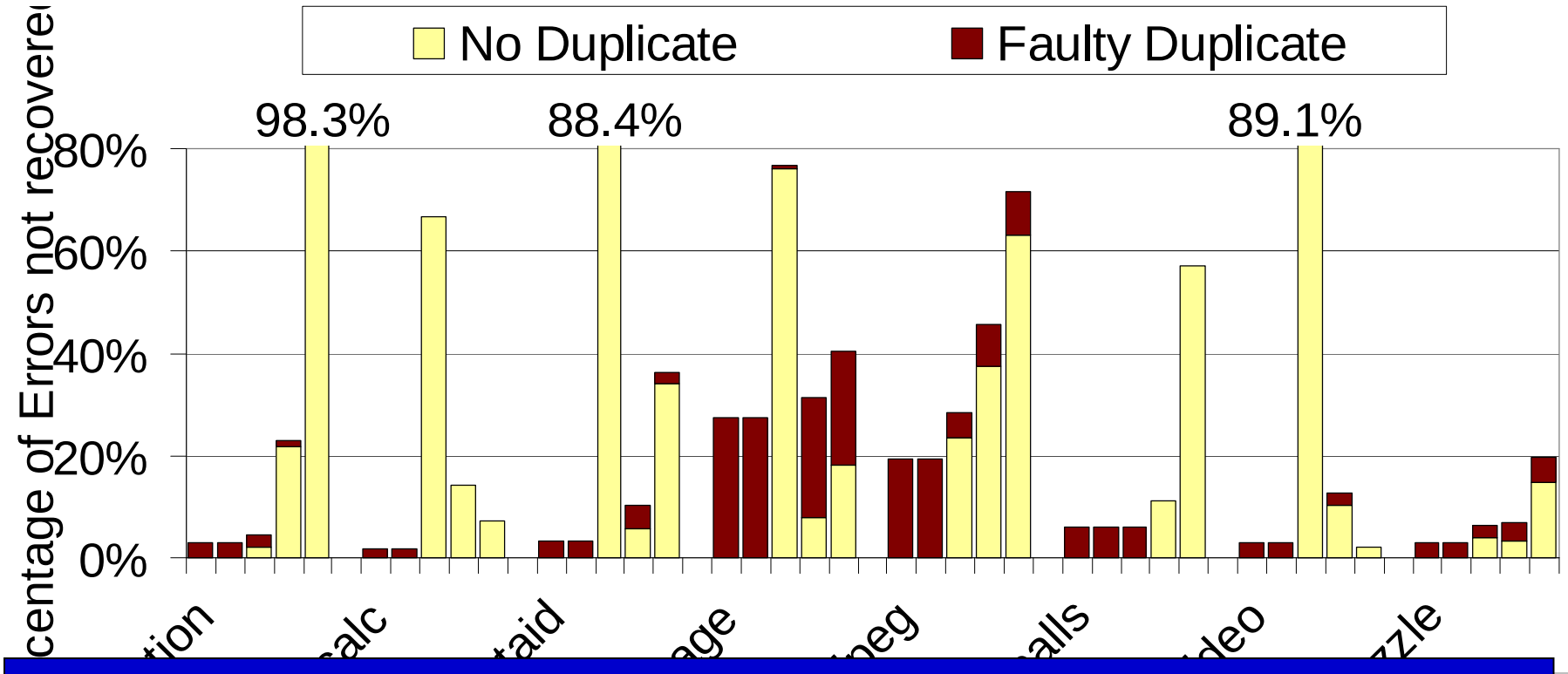
Selective schemes performs well in terms of heap space consumption. The ideal LAST-USE-OPT performs very well. PRFL and DELAYED are better than COMPDUPL.

Non-correctable Error Rates



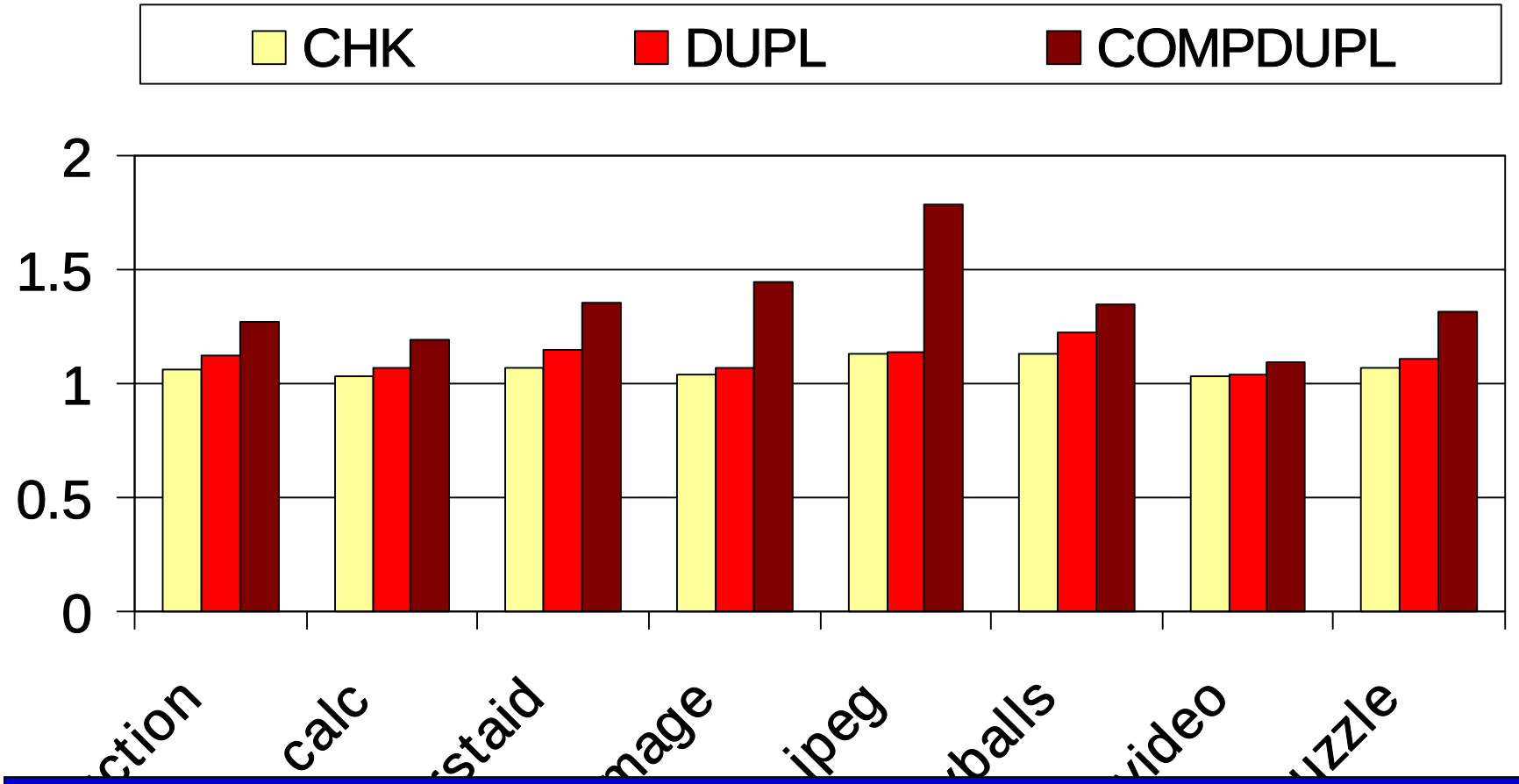
Average non-correctable error rate of DUPL and COMPDUPL are 8.2% and 4.3%, respectively.

Non-correctable Error Rates



Average non-correctable error rate of LAST-USE-OPT, LAST-USE-100, LAST-USE-80, PRFL, and DELAYED are 8.2%, 8.2%, 50.2%, 19.1% and 41.6%, respectively.

Execution Cycles



Average execution cycles increase of DUPL and COMPDUPL are 11.3% and 34.9%, respectively.

Conclusion

- Object duplication improves data integrity and has good error coverage
- Compression reduces heap occupancy but increases execution cycles significantly
- Selective schemes help tradeoff heap space consumption, error recover rate, and performance

Thank you!