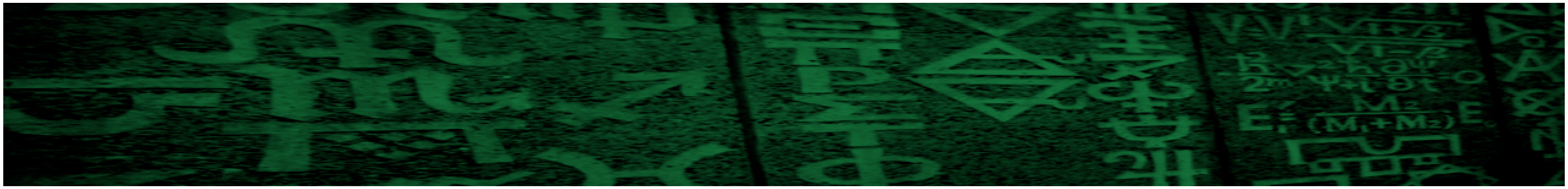
# Parallel and Concurrent Real-time Garbage Collection

## Part I: Overview and Memory Allocation Subsystem

David F. Bacon

IBM

T.J. Watson Research Center



# What It Does

(Demo)

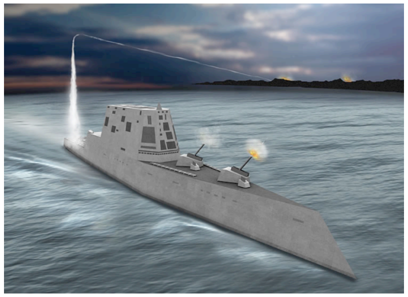
http://www.youtube.com/user/ibmrealtime

# What it Is

- A production garbage collector that is
  - Real-time (450us worst-case latencies)
  - Multiprocesing (uses multiple CPUs)
  - Concurrent (can run in background)
  - Robust (within and across JVMs)

# Why It’s Important

Telco SIP Switch



DDG-1000 Destroyer

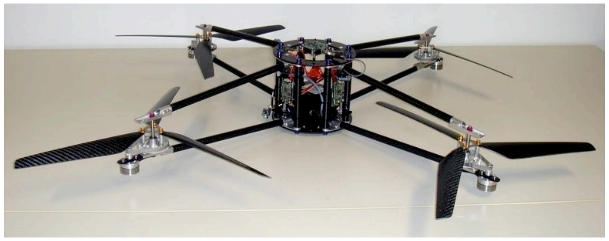
Trade Execution

Playstation/Xbox etc

Automotive Electronics

Java-based
Synthesizer

JAviator
(w/ Salzburg)



Air Java
(w/ Berkeley CE)

# Who and When

Recycler
(1999-2001)

*Dick Attanasio*
*David Bacon*
*V.T. Rajan*
*Steve Smith*

*Han Lee*

Metronome
(2001-2004)

*David Bacon*
*Perry Cheng*
*V.T. Rajan*

*Martin Vechev*

WebSphere Realtime
(2004-2007)

*Josh Auerbach*
*David Bacon*
*Perry Cheng*
*Dave Grove*

*5 Developers*
*10 Testers*
*5 Salespeople*
…

# Digression: Keys to Success

- Intelligence
- Collaboration
- Problem Selection

# Perspectives

- Concurrent garbage collection is
  - A key language runtime component
  - A challenging verification problem
  - A multi-faceted concurrent algorithm

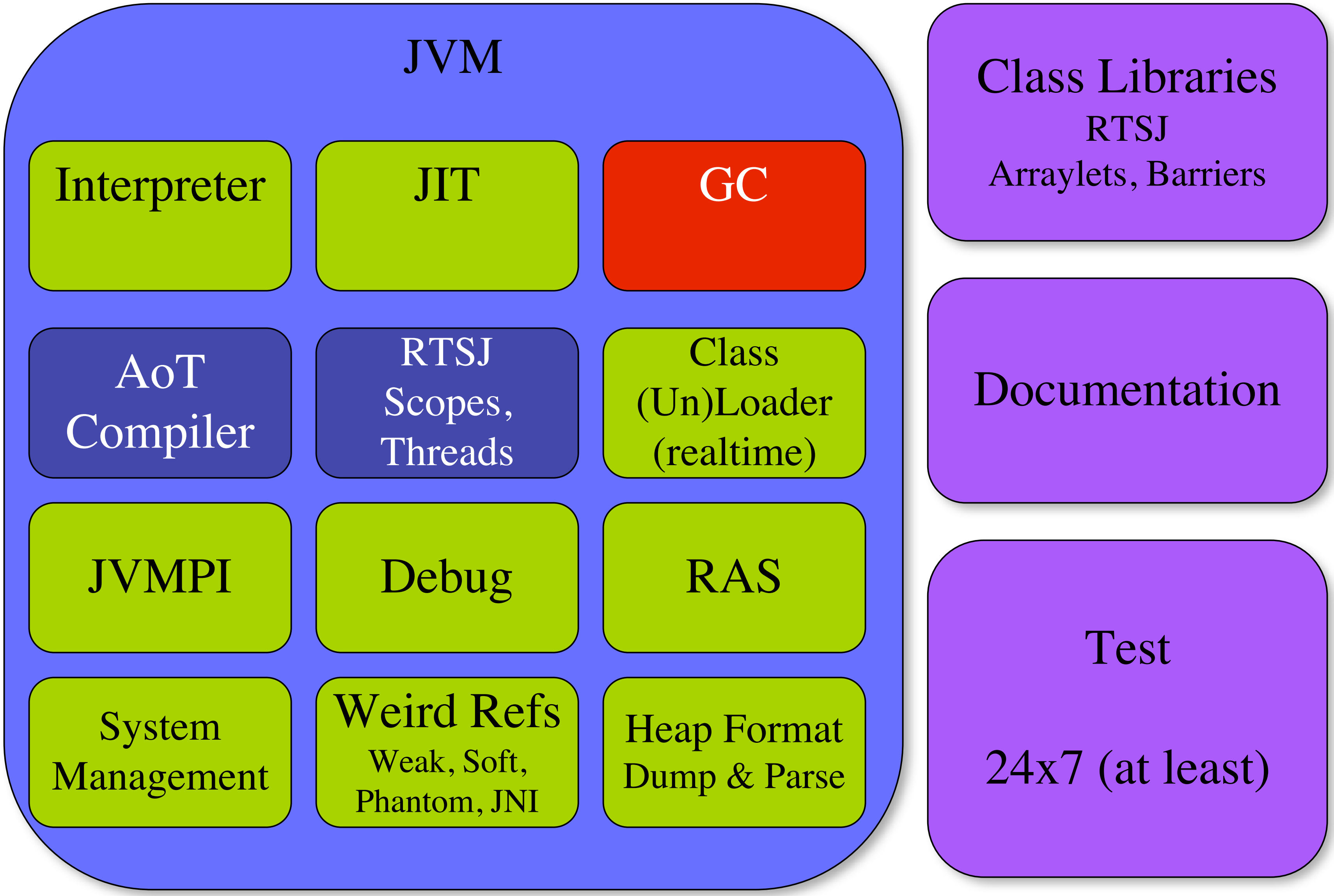
# Goals

- Learn how to bridge:
  - from abstract design…
  - …to concrete implementation

- Learn how to combine different
  - algorithms…
  - …and implementations…
  - …into a complete system

- Gain deep understanding
  - highly complex, real-world system
  - apply lessons to your problems

# Where it Fits In



JVM

Interpreter

JIT

GC

AoT Compiler

RTSJ Scopes, Threads

Class (Un)Loader (realtime)

JVMPI

Debug

RAS

System Management

Weird Refs
Weak, Soft, Phantom, JNI

Heap Format Dump & Parse

Class Libraries
RTSJ
Arraylets, Barriers

Documentation

Test

24x7 (at least)

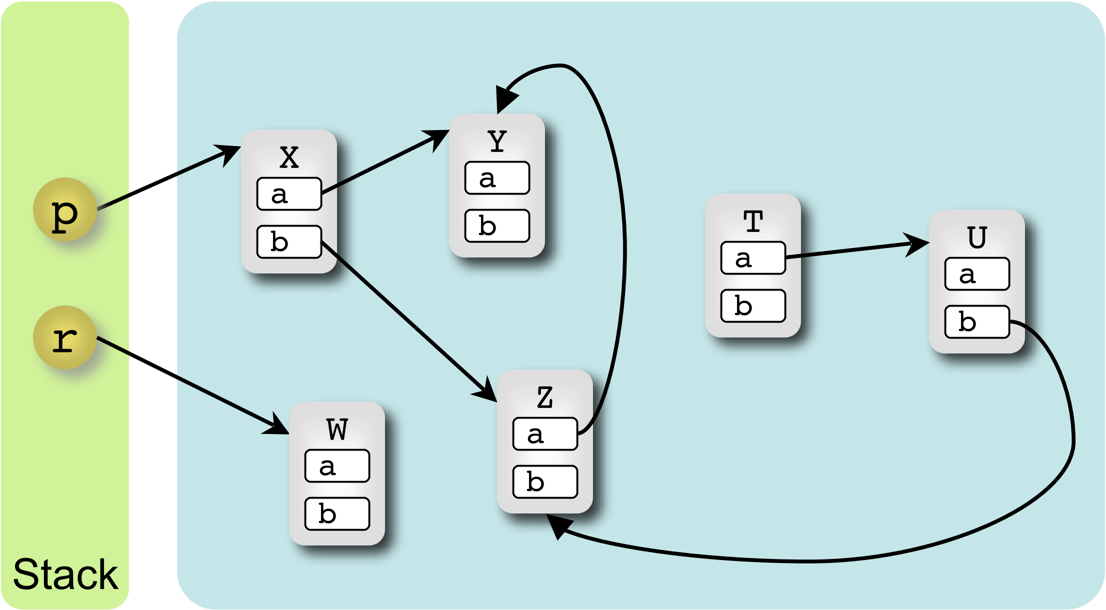
# Fundamental Issues

- Functional correctness (duh)
- Liveness
  - Timeliness (real-time bounds)
- Fairness
  - Priorities
- Initiation and Termination
- Contention
- Non-determinism

# Why is Concurrency Hard?

- Performance
  - – Contention
  - – Load Balancing
  - – Overhead -> Granularity
- “Inherent” Simultaneity
- Timing and Determinism

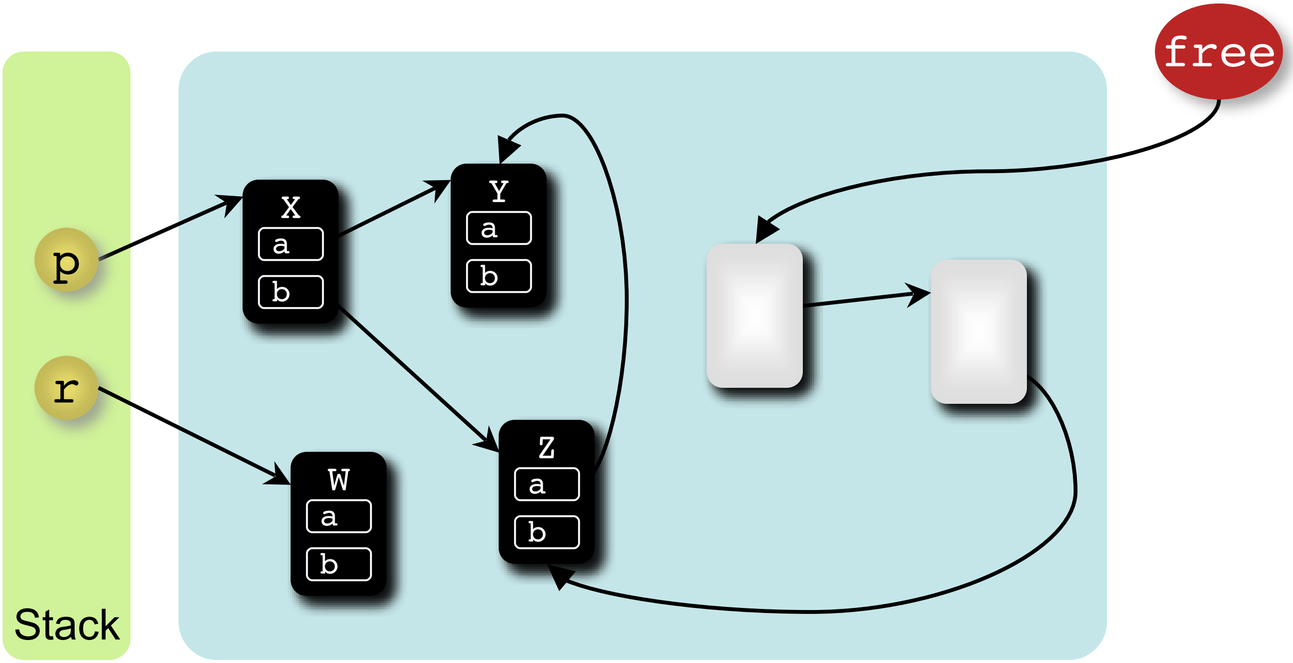
# GC: A Simple Problem (?)



```
Class Foo {
        Foo a;
        Foo b;
}
```

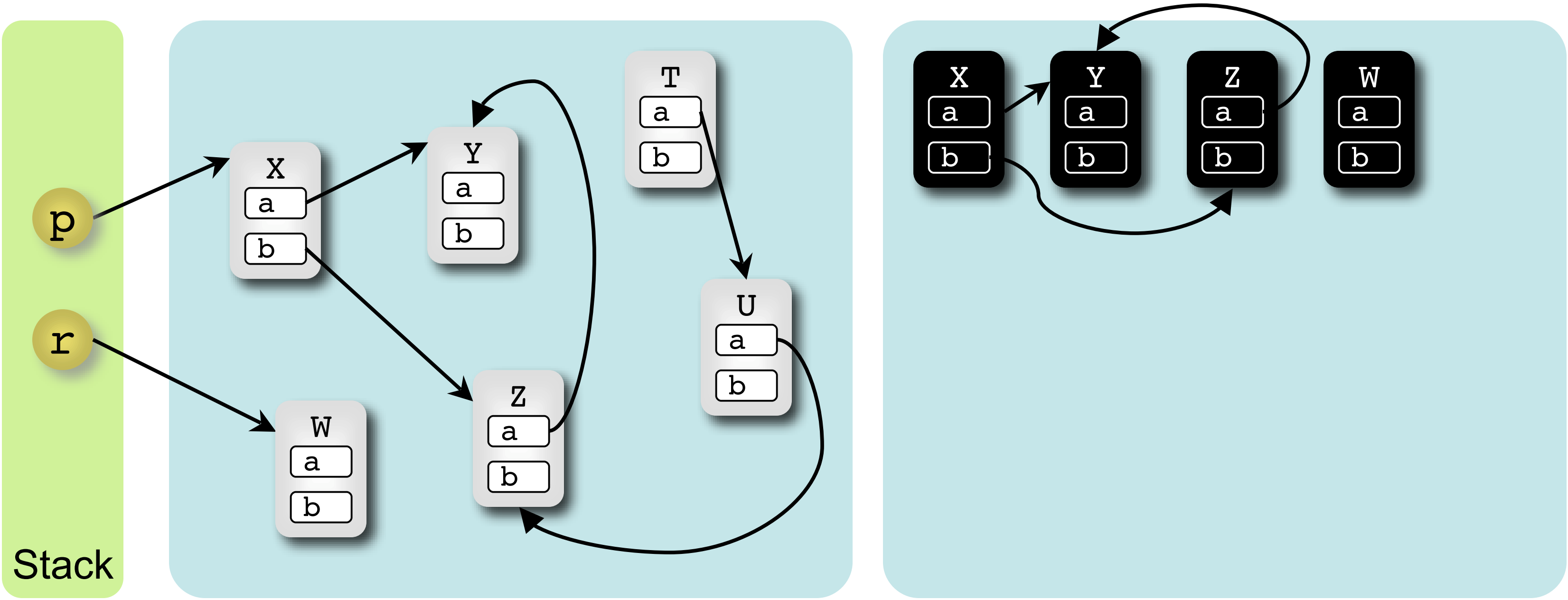
- Transitive Graph Closure

# Basic Approaches: Mark/Sweep



- *O(live)* mark phase but *O(heapsize)* sweep
- Usually requires no copying
- Mark stack is *O(maxdepth)*

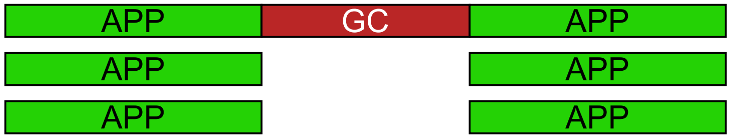
# Basics II: Semi-space Copying



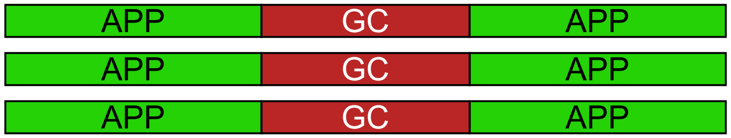
- *O(live)*
- If single-threaded, no mark stack needed
- Wastes 50% of memory

# Kinds of “Concurrent” Collection

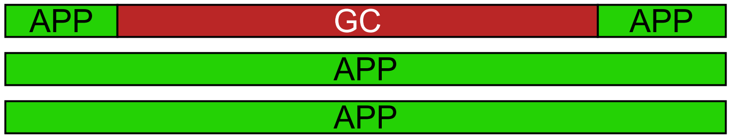
- “Stop the World”



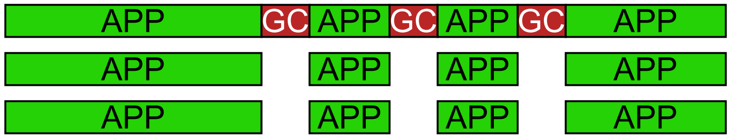
- Parallel



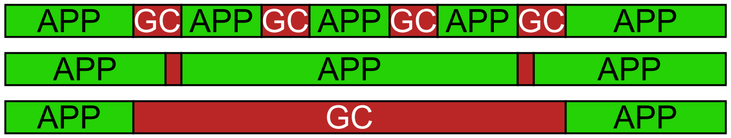
- Concurrent



- Incremental



# Our Subject: Metronome-2 System

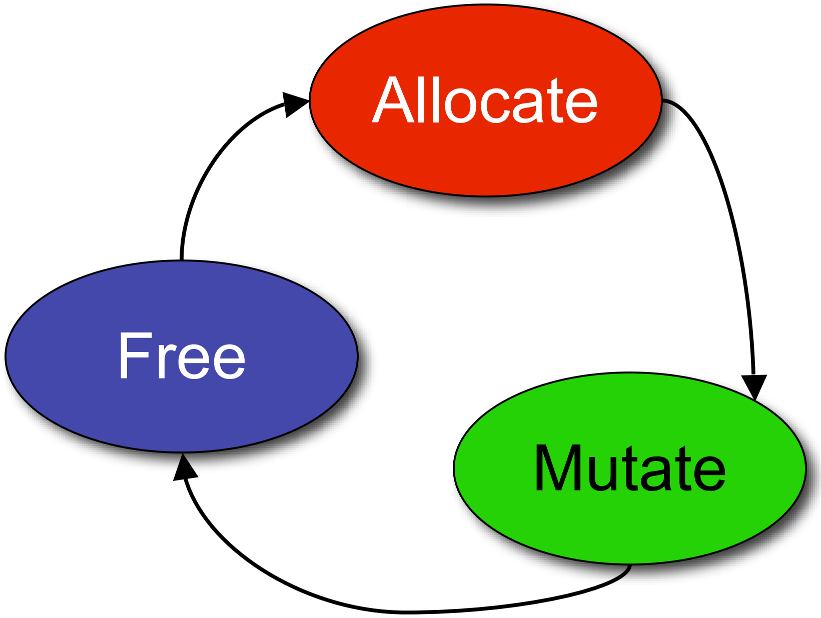


- Parallel, Incremental, and Concurrent
- No increment exceeds 450us
- Real-time Scheduling
- Smooth adaptation from under- to over-load
- Implementation in production JVM

# What Does “Real-time” Mean?

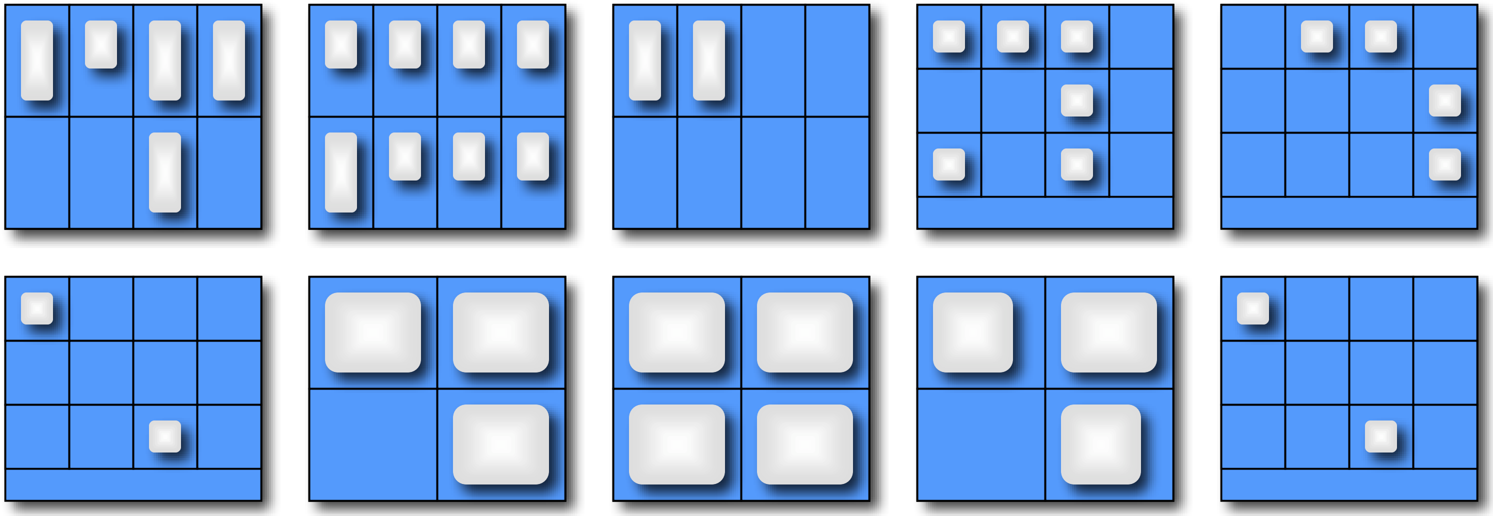
- Minimal, predictable interruption of application
- Collection finishes before heap is exhausted
- “Real space” - bounded, predictable memory
- Honor thread priorities
- Micro- or macro-level determinism (cf. CK)

# The Cycle of Life



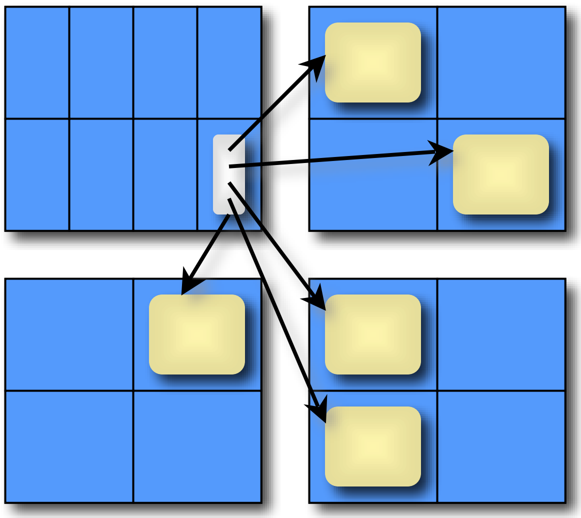
- Not really a “garbage collector”…
- … but a memory management subsystem

# Metronome Memory Organization



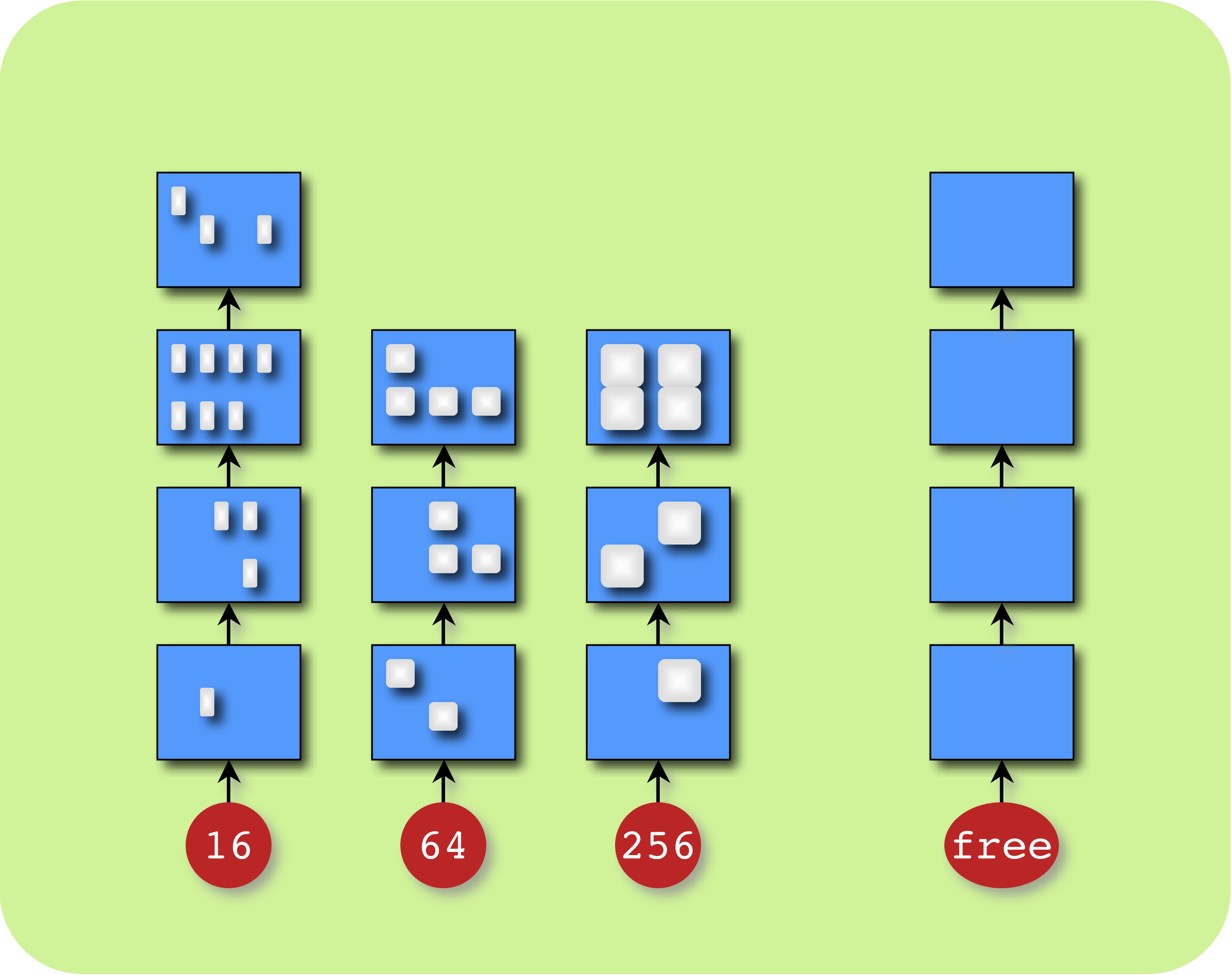
- Page-based
- Segregated free lists
- Ratio bounds internal & page-internal fragmentation

# Large Objects: Arraylets

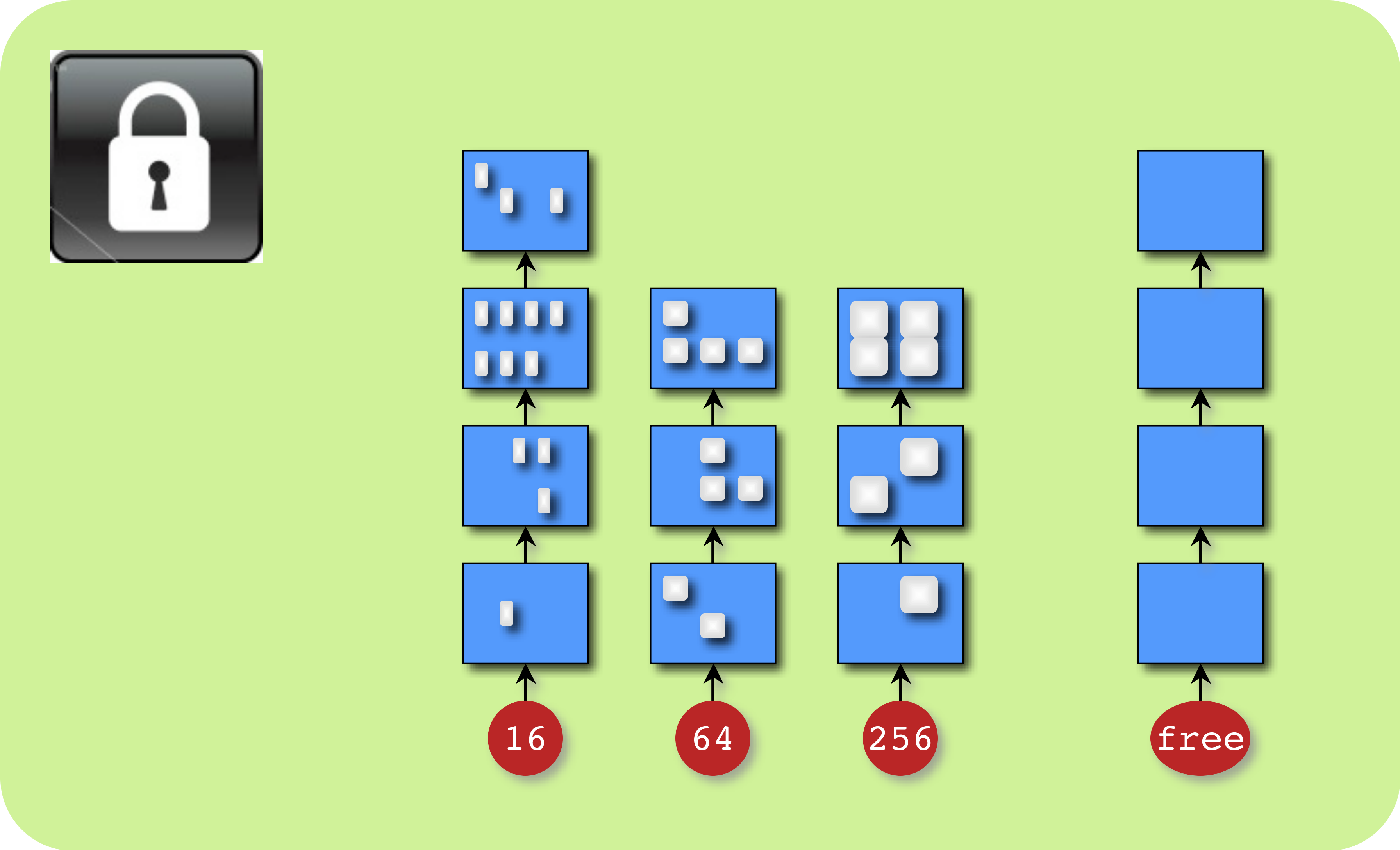


- (Almost) eliminates external fragmentation
- (Almost) eliminates need for compaction
- Very large arrays still need contiguous pages
- Extra indirection for array access

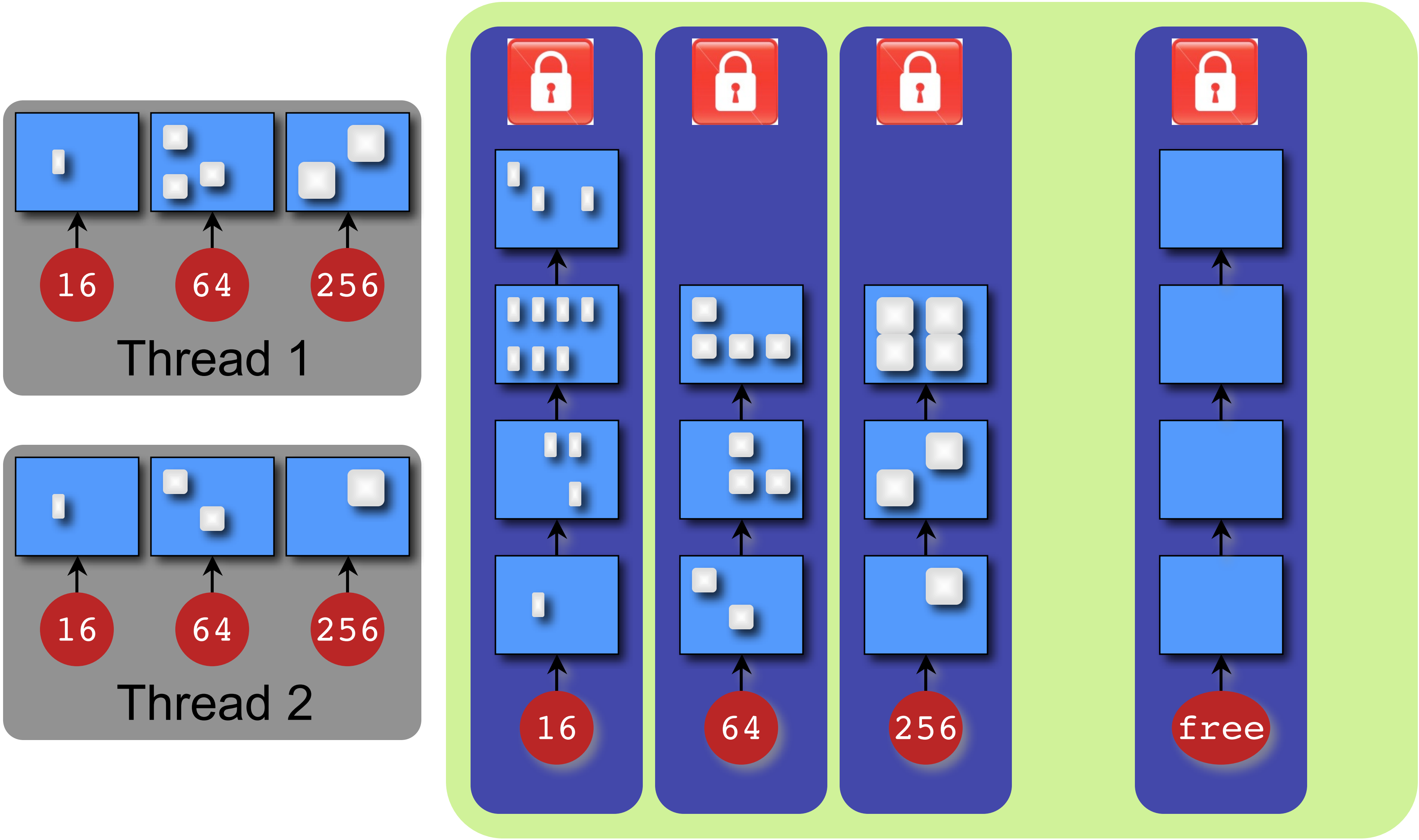
# Page Data Structures

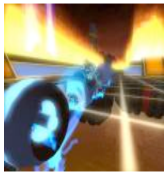


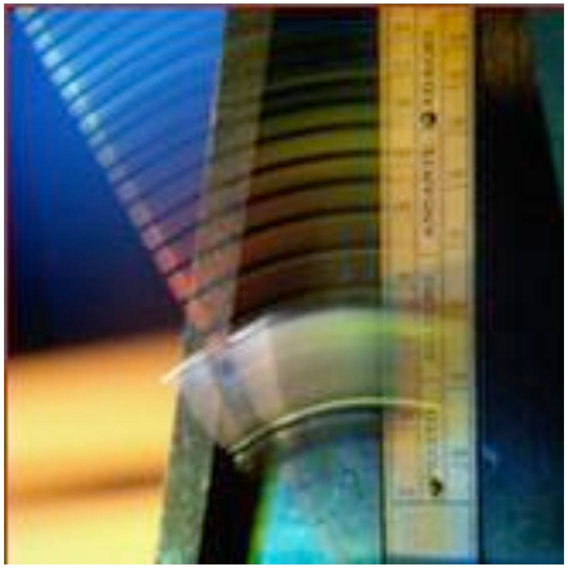
# Page Data Synchronization, Take 1



# Page Data, Take 2







http://www.research.ibm.com/metronome

https://sourceforge.net/projects/tuningforkvp