Tuning Garbage Collection in an Embedded Java Environment

G. Chen, R. Shetty, M. Kandemir, N. Vijaykrishnan, M. J. Irwin, and M. Wolczko*
Penn State University
Sun Microsystems*

http://www.cse.psu.edu/~mdl
Outline

- Background: KVM and Garbage Collection
- Banked memory architecture and Garbage Collection controlled bank turn off
- Parameter tuning:
  - Frequency
  - Compaction
  - Allocation Strategy
- Conclusions and Future Work
Part I
Introduction to KVM and Garbage Collection
What is KVM?

From KVM White Paper (Sun Microsystems)
Resource Constraints on KVM

- **Small memory budget**
  - Typically 512 KB. “K” stands for “kilo”
  - Lots of research has been carried out

- **Limited energy supply**
  - Powered by batteries
  - Not yet well exploited

- **Requires**: redesign of all the components of virtual machine. Our work is focused on **Automatic Memory Management Subsystem (Garbage Collector)**
Garbage Collection in KVM

- **Why?**
  - To reclaim memory occupied by objects no longer needed by the application

- **When?**
  - Not enough free space in the heap for new object (conventionally)

- **How? - 3 Phases:**
  - Mark - find out garbage
  - Sweep - free the memory occupied by garbage
  - Compact - combine fragments (optional)
Mark Phase – detection of garbage

▲ Roots: internally defined by the KVM implementation

▲ Live Objects: reachable from the roots

▲ Garbage (Dead Objects): not reachable from the roots, not accessible to the application
Compaction Phase

Why?

- Heap fragmentation wastes heap memory, increases allocation cost due to longer free list scanning

How?

1. Slide live objects to one end of the heap and free blocks to the other end.
2. Combine free blocks into one contiguous free area
3. References to the moved objects are updated
Mark / Sweep / Compaction

Before GC

After Mark

After Sweep

After Compact

Live  Garbage  Unknown  Free

root

A

B

C

D

E

F

G

root

A

B

C

D

E

F

G

root

B

E

G

root

B

E

G
Part II

Banked Memory Architecture

&

GC Controlled Bank Turn off
System Energy Distribution

![Energy Distribution Graph]

- **Core**: Represents the core energy consumption.
- **Heap (Dynamic)**: Indicates dynamic heap usage.
- **Runtime Stack (Leakage)**: Shows leakage in the runtime stack.
- **Runtime Stack (Dynamic)**: Displays dynamic runtime stack usage.
- **ROM (leakage)**: Indicates leakage in the ROM.
- **ROM (Dynamic)**: Shows dynamic ROM usage.
System Energy Distribution (Average)

Heap is a major consumer of system energy (45%)
Live Object Size Oscillations

Manyballs (max: 20682 bytes; avg: 13276 bytes)

- Heap memory requirement varies during execution
- Turning off part of the heap memory when memory requirement is low can save energy
Banked Memory

1. Each bank can be turned on/off by software independently
2. When a bank is turned off, it consumes very little energy
3. If a bank is turned off, data in it is lost
## Energy Characteristics of Memory Banks

<table>
<thead>
<tr>
<th>Mode</th>
<th>Dynamic</th>
<th>Leakage</th>
<th>Data Lost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read (ON)</td>
<td>yes</td>
<td>yes</td>
<td>Not lost</td>
</tr>
<tr>
<td>Write (ON)</td>
<td>yes</td>
<td>yes</td>
<td>Not lost</td>
</tr>
<tr>
<td>No access (ON)</td>
<td>no</td>
<td>yes</td>
<td>Not lost</td>
</tr>
<tr>
<td>OFF</td>
<td>no</td>
<td>no</td>
<td>Lost</td>
</tr>
</tbody>
</table>

Note: turning on a currently off bank incurs 350 cycles’ penalty (resynchronization cost)
Embedded System Configuration

PROCESSOR CORE

DCACHE

ICACHE

Optional

RAM

Heap

Runtime Stack

KVM Code

Class Libraries

RAM

ROM

Bank
## GC Controlled Bank Turn Off

<table>
<thead>
<tr>
<th>Time (T)</th>
<th>BANK1</th>
<th>BANK2</th>
<th>BANK3</th>
<th>BANK4</th>
</tr>
</thead>
<tbody>
<tr>
<td>T=0</td>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
</tr>
<tr>
<td>T=1</td>
<td>A</td>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
</tr>
<tr>
<td>T=50</td>
<td>A B</td>
<td>C D</td>
<td>E F</td>
<td>G</td>
</tr>
<tr>
<td>T=100</td>
<td>A B</td>
<td>C D</td>
<td>E F</td>
<td>G</td>
</tr>
<tr>
<td>T=200</td>
<td>A</td>
<td>C D</td>
<td>OFF</td>
<td>G</td>
</tr>
<tr>
<td>T=500</td>
<td>A</td>
<td>C D</td>
<td>OFF</td>
<td>G</td>
</tr>
<tr>
<td>T=800</td>
<td>A C G</td>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
</tr>
</tbody>
</table>

- **Live**: Areas marked as "Live" indicate active elements.
- **Garbage**: Areas marked as "Garbage" indicate inactive elements.

- **Energy wasted in bank3**: Indicates energy loss in bank3 at T=200.
- **Compaction increases turn off potential**: Indicates increased potential for turning off bank3 at T=500.
Part III

Tuning KVM's Garbage Collection
Experiment Settings

- Shade Simulator
- KVM
- Java Application

Profile:
- Instruction Counts
- Memory Accesses

Energy Parameters

Energy Value
## Benchmarks

<table>
<thead>
<tr>
<th>Application</th>
<th>Description</th>
<th>Application</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crypto</td>
<td>Cryptograph Lib</td>
<td>MathFP</td>
<td>Fixed-point math lib</td>
</tr>
<tr>
<td>Calculator</td>
<td>Arithmetic calculator</td>
<td>Kwml</td>
<td>WML browser</td>
</tr>
<tr>
<td>Kshape</td>
<td>E-map on KVM</td>
<td>Scheduler</td>
<td>Personal scheduler</td>
</tr>
<tr>
<td>Elite</td>
<td>3D rendering</td>
<td>Kvideo</td>
<td>KPG (MPEG for KVM) decoder</td>
</tr>
<tr>
<td>ManyBalls</td>
<td>Game</td>
<td>Missiles</td>
<td>Game</td>
</tr>
<tr>
<td>Dragon</td>
<td>Game</td>
<td>StarCruiser</td>
<td>Game</td>
</tr>
</tbody>
</table>
Influence of GC Controlled Bank turn off

On average, 30% heap energy is saved
Influence of GC Frequency

- More frequent GC
  - Earlier detection of banks containing only garbage
  - More memory accesses
  - More CPU operations
  - Longer execution time

More frequent GC

Calculator Dragon Kvideo Kml Missiles

% Heap Energy

100 90 80 70 60 50 40 30 20 10 0

Out of Memory
Influence of Compaction

% Heap Energy

- Calculator
- Dragon
- Kvideo
- Kwml
- Missiles

No Compact
Compact
Influence of Allocation Strategy

Allocation policy of objects also influences energy consumption

- **First Bank**: Allocate objects from the first bank having enough space
- **Active Bank**: Try to allocate objects from currently active banks. If fails, turn on a new bank
- **Active+GC**: In addition to Active, perform GC to find space in active banks before turning on new bank
First/Active Bank Allocation

Before Allocation

<table>
<thead>
<tr>
<th>BANK1</th>
<th>BANK2</th>
<th>BANK3</th>
<th>BANK4</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>OFF</td>
<td>OFF</td>
</tr>
</tbody>
</table>

First Bank Allocation

<table>
<thead>
<tr>
<th>BANK1</th>
<th>BANK2</th>
<th>BANK3</th>
<th>BANK4</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>D</td>
<td>OFF</td>
</tr>
</tbody>
</table>

Active Bank Allocation

<table>
<thead>
<tr>
<th>BANK1</th>
<th>BANK2</th>
<th>BANK3</th>
<th>BANK4</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>OFF</td>
<td>C</td>
</tr>
</tbody>
</table>

Influence of Allocation Strategy

![Bar Chart]

- Calculator
- Dragon
- Kvideo
- Kwml
- Missiles

% Heap Energy

- First
- Active
- Active+GC
Conclusions

- Banked memory with GC controlled turn off reduces heap energy (30% on average)

- GC invocation frequency critically impacts system energy (optimal frequency depends on the application)

- Object allocation and compaction schemes affect the potential of memory bank turn-off

- Strategies can be applied to design energy aware memory management software
Future Work

- **Adaptive GC**
  - Automatically find out the optimal GC frequency

- **Object Collocation**
  - Put objects having same life time in a same bank

- **More memory bank states**
  - Active, Stand-by, Sleep, Power-off
Adaptive GC

Missiles

<table>
<thead>
<tr>
<th>Energy</th>
<th>10</th>
<th>40</th>
<th>75</th>
<th>100</th>
<th>250</th>
<th>Out of Memory</th>
<th>adpt</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LHEAP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DHEAP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LSTACK</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSTACK</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LROM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DROM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Thank you!
Appendixes
ROM Energy Consumption

% ROM Energy

Calculator | Dragon | Kvideo | Kwml | Missiles

- 10
- 40
- 75
- 100
- 250
- Out of Memory

Out of Memory

Calculator Dragon Kvideo Kwml Missiles
Performance (1)

![Graph showing the execution time percentage for Missiles with different numbers of Out of Memory instances. The x-axis represents the number of Missiles (10, 40, 75, 100, 250, Out of Memory), and the y-axis represents the execution time percentage ranging from 0 to 100%.]
Performance (2)

![Kvideo](image-url)

Execution time vs. Out of Memory for different Kvideo values.
Influence of Cache

4KB I-Cache + 4KB D-Cache
2-Way Association

% Heap Energy

Calculator  Dragon  Kshape  Kvideo  Missiles

M&S, Out of Memory  M&S, Active+GC