Processes in KaffeOS: Isolation, Resource Management, and Sharing in Java

Godmar Back
Wilson Hsieh   Jay Lepreau
School of Computing
University of Utah

Motivation

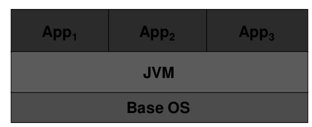
- Java Applications
  - Untrusted code: possibly malicious or buggy
  - Multiple applications on behalf of multiple users
- Efficient use of resources
  - Resource-constrained environments: “small” systems (handhelds, embedded systems)
  - Increased scalability

What is KaffeOS

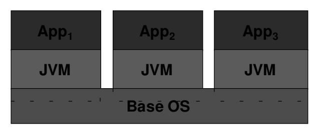
- A Java operating system:
  - Enhanced JVM that provides operating system functionality
- Features:
  - Separation
  - Resource management
  - Direct sharing
- Focus on memory (GC) and CPU management
- Architectural concepts taken from OS
- Mechanisms taken from garbage collection

Current Options

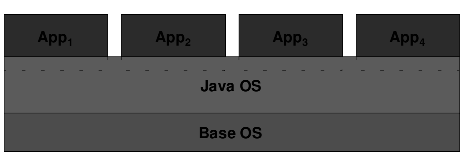
- Multiple apps in one JVM



- One app per JVM in different OS processes



Java Operating System



\+ Good separation
\+ Good resource management
\+ Allows some direct sharing

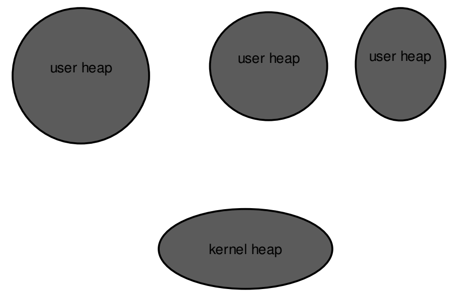
KaffeOS Design Principles

- Process separation
- Full reclamation of memory
- Precise memory and CPU accounting
- Direct sharing
- Safe termination of processes
- Hierarchical memory management

Process Separation

- **Protecting access to a process's objects**
  - **Via Java type safety, via Java name spaces**
- **Define a user/kernel boundary**
  - **Protect integrity of kernel**
  - **Guarantee safe termination [HotOS '99]**
- **Separation of resources**
  - **Memory, CPU, garbage collection**

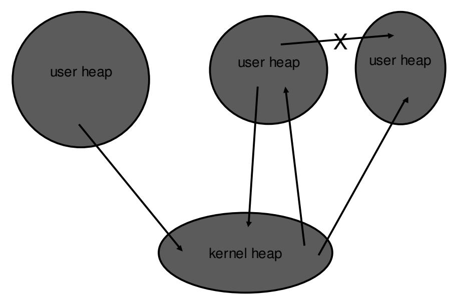
Heap Structure (user heaps)



Full Reclamation

- **Assumption: do not have MMU to unmap memory**
  - **Rely on garbage collection to free objects**
- **References can make objects unreclaimable**
  - **Define and enforce set of rules for writes that create references**
- **Define heap structure**

Heap Structure (user heaps)



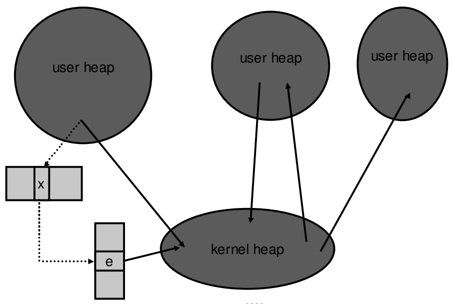
Enforcing Heap Structure

- **Certain cross-heap pointers are illegal**
  - **User-user pointers are illegal**
- **Use GC "write barrier" to enforce**
  - **Check heaps on every write**
  - **Throw SegmentationViolationError if illegal**
- **Small cost on legal write (common case)**
  - **Microbenchmarks (tight loop, hot cache)**
    - » **41 cycles without memory overhead**
    - » **25 cycles with slight memory overhead**

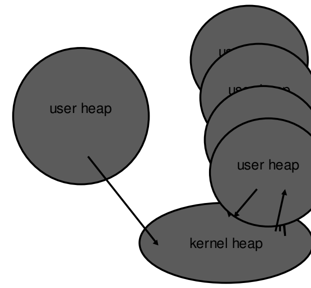
Memory and CPU Accounting

- **Accurate and complete per-process accounting**
  - **Minimize number of objects on kernel heap**
  - **Minimize amount of time spent in kernel code**
- **Separate garbage collection**
  - **Minimize unaccounted resource use**
  - **Avoid priority inversion**
  - **Maintain isolation**

Heap Structure (separate GC)



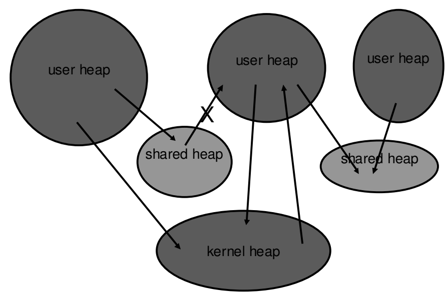
Heap Structure (resolving cycles)



Direct Sharing

- **As Interprocess Communication Mechanism**
- **Objects may contain pointers to other objects**
  - **Preserve Java model**
- **Accounting of shared objects**
  - **Sharers are all charged for shared heaps: double charging**
  - **Non-double charging does not work**
- **Reclamation**
  - **As soon as garbage collector detects that nothing on shared heap is referenced**

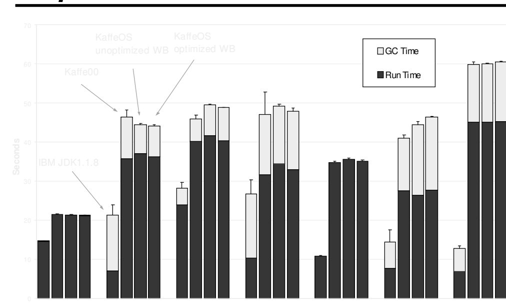
Heap Structure (shared heaps)



Performance Results

- **Overhead for well-behaved applications**
  - **SpecJVM98 benchmarks: 2-8%**
- **Comparing**
  - **IBM JDK 1.1.8**
  - **Base JVM: Kaffe (www.kaffe.org)**
    - » **Current version: Kaffe00 (June 2000)**
  - **KaffeOS with unoptimized write barrier**
  - **KaffeOS with optimized write barrier (memory overhead)**

SpecJVM Performance of KaffeOS

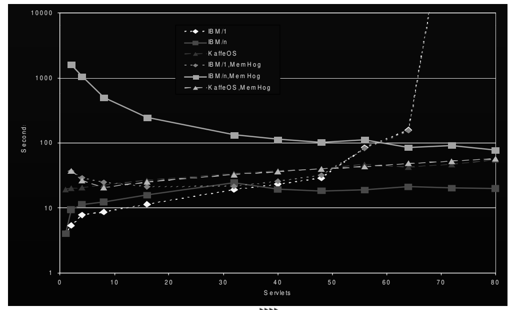


MemHog DoS Scenario

- Off-the-shelf Servlet engine
  - Linux 2.2.x w/ 256MB, Apache 1.3.12, JServ 1.1, JSDK 2.0
- How would we deal with this?
  - Run 1 servlet per IBM JVM (IBM/1)
  - Run all servlets on one IBM JVM (IBM/n)
  - Run each servlet in a KaffeOS process

IBM/1	IBM/N	KaffeOS
IBM/1, MemHog	IBM/N, MemHog	KaffeOS, MemHog

Service Under DoS Attack



Related Work

- Other Java Operating Systems
  - J-Kernel/JRes [Hawblitzel '98, Czajkowski '98], Alta [Tullmann '99], Java Nucleus [van Doorn '00]
- Java VM extensions
  - Multiprocess JVM [Balfanz '98], IBM [Dillenberger '00], Oracle 8i
- Java language extensions
  - Luna [Hawblitzel]
  - Java Realtime Extensions [Sun '00]
- Operating Systems
  - Opal [Chase '94]
  - Pilot [Redell et al. '80], Cedar [Swineheart et al. '86],
  - Spin [Bershad '95], Inferno [Dorward et al. '97]

Ongoing Work & Open Issues

- Running other DoS attacks
  - GarbageHog: allocate and abandon lots of memory
  - CPUHog: execute for too long
- Create bigger applications that use direct sharing model for IPC
- Demonstrate safe termination claim experimentally

Summary

- Java operating systems are necessary to support untrusted Java applications efficiently
- KaffeOS
  - Architectural concepts taken from operating systems
  - Mechanisms taken from garbage collection
    - » Distributed GC
    - » Write barriers
- Big issue: dealing with memory/GC
- Resource-based denial-of-service attacks can be stopped