Introduction

Processor-Memory performance gap
Many hardware mechanisms have been proposed to reduce memory access time
• Pipelining, cache memory,prefetching, etc.
Software mechanisms are also needed to efficiently manage the available memory.

- Kernel: Slab allocator, Page cache, etc.
- Userspace: Memory allocators, garbage collectors, etc.

Those mechanisms are complex and difficult to debug.

Our goal is to provide tools and algorithms able to evaluate the efficiency of those mechanisms and to detect performance problems.
Agenda

- Investigations
  - Dynamic Trace-based Sampling Algorithm for Memory Usage Tracking
  - Performance Analysis of Garbage collection algorithms in Java Virtual Machines
Dynamic Trace-based Sampling Algorithm for Memory Usage Tracking
Dynamic Trace-based Sampling Algorithm for Memory Usage Tracking

Motivation and challenges

- The high frequency of memory operations makes the trace file huge
- Basic trace filtering techniques are not possible, since physical memory releasing can be done in the context of other processes
Investigations
Dynamic Trace-based Sampling Algorithm for Memory Usage Tracking

Proposed architecture
Investigations

Dynamic Trace-based Sampling Algorithm for Memory Usage Tracking

Virtual memory monitoring

Memory-related system calls `mmap()`, `munmap()`, and `sbrk()` as a triggers for the Kernel Counters Reader

![Diagram showing memory allocation and system calls](image)
Investigations
Dynamic Trace-based Sampling Algorithm for Memory Usage Tracking

Physical memory monitoring:

An event is triggered if memory variability exceeds a certain threshold
Dynamic Trace-based Sampling Algorithm for Memory Usage Tracking

Investigations

Physical memory monitoring:

```
startTimer(rate)
if Timer tick then
    processes ← ListSystemProcesses()
    for process in processes do
        trace_memory(process)
        variability[process] ← 0
    end
   _restartTimer()
end

// probe
if memory page allocated/released then
    process ← getCurrentProcess()
    variability[process] += PAGE_SIZE * direction
    if variability[process] exceeds the threshold then
        trace_memory(process)
    end
end
```

Algorithm 1: Dynamic sampling Algorithm

- Implemented as a Kernel module.
- Configurable through the `proc` file system (sampling rate, variability threshold)
- Lock-free data structures are used to provide a good scalability
Investigations

Dynamic Trace-based Sampling Algorithm for Memory Usage Tracking

Firefox memory usage at start-up using Massif / Dynamic Sampling Algorithm

Video player memory usage using Massif / Dynamic Sampling Algorithm
Investigations

Dynamic Trace-based Sampling Algorithm for Memory Usage Tracking

Performance  (20 GB memory workload)

Execution Time

![Graph showing performance over buffer sizes](image)
Dynamic Trace-based Sampling Algorithm for Memory Usage Tracking

Performance (20 GB memory workload)

Trace Size

<table>
<thead>
<tr>
<th>Buffer Size</th>
<th>100 Mb</th>
<th>250 Mb</th>
<th>500 Mb</th>
<th>1 G</th>
<th>2 G</th>
<th>4 G</th>
<th>5 G</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reduction factor</td>
<td>7.4</td>
<td>5.5</td>
<td>4.3</td>
<td>3.57</td>
<td>3.45</td>
<td>68</td>
<td>59</td>
</tr>
</tbody>
</table>
Performance Analysis of Garbage collection algorithms in Java Virtual Machines
Investigations
Performance Analysis of Garbage collection algorithms in Java Virtual Machines

Explicit memory management problems: dangling references, memory leaks, etc.

Automatic memory management

Garbage collection

• Reclaims unused memory by automatically removing unreachable objects
Characteristics of garbage collectors

- Serial / Parallel
- Concurrent / Stop-the-world
- Compacting / Non-Compacting

Performance metrics

- Throughput
- Overhead
- Pause time
- Frequency
Investigations
Performance Analysis of Garbage collection algorithms in Java Virtual Machines

- Minor collections must be fast
- Major collections must be space efficient
Performance Analysis of Garbage collection algorithms in Java Virtual Machines

It is difficult to find a good configuration of the garbage collector
Investigations

Performance Analysis of Garbage collection algorithms in Java Virtual Machines

A misconfiguration can cause of a very bad performance
Investigations

Performance Analysis of Garbage collection algorithms in Java Virtual Machines

We instrumented the garbage collector of the Hotspot virtual machine in order to recover useful performance metrics
Investigations

Performance Analysis of Garbage collection algorithms in Java Virtual Machines

- Minor collections
- Major collections

Minor / Major size (MB)

02 / 30
05 / 27
10 / 22
15 / 17
20 / 12
25 / 7
30 / 2
Thank You!