Applying Priorities to Memory Allocation

Sven Gestegård Robertz

Department of Computer Science

Lund University

Sweden

LUND INSTITUTE OF TECHNOLOGY
Lund University

Background

Embedded systems

- Small memory
- Real-time applications
- Robustness

Background

Memory is a global resource

- Out-of-memory errors have serious consequences
- Great responsibility on programmers

LUND INSTITUTE OF TECHNOLOGY
Lund University

Background

Not all of the code is critical

- Critical parts must always be executed
- Non-critical parts may be skipped if there is not enough memory to run them safely
- Critical and non-critical “aspects”

The basic idea

Prevent system from running out of memory by limiting the amount of non-critical allocations.

- Traditionally done manually
- Run-time system support

Priorities for memory allocations!

Key points

- Guarantee that all critical high priority allocations succeed w/o delay
- Prevent non-critical memory allocations in high priority processes from starving low priority processes.
- Memory priority and CPU priority are orthogonal
- Worst case analysis only needed for the critical parts

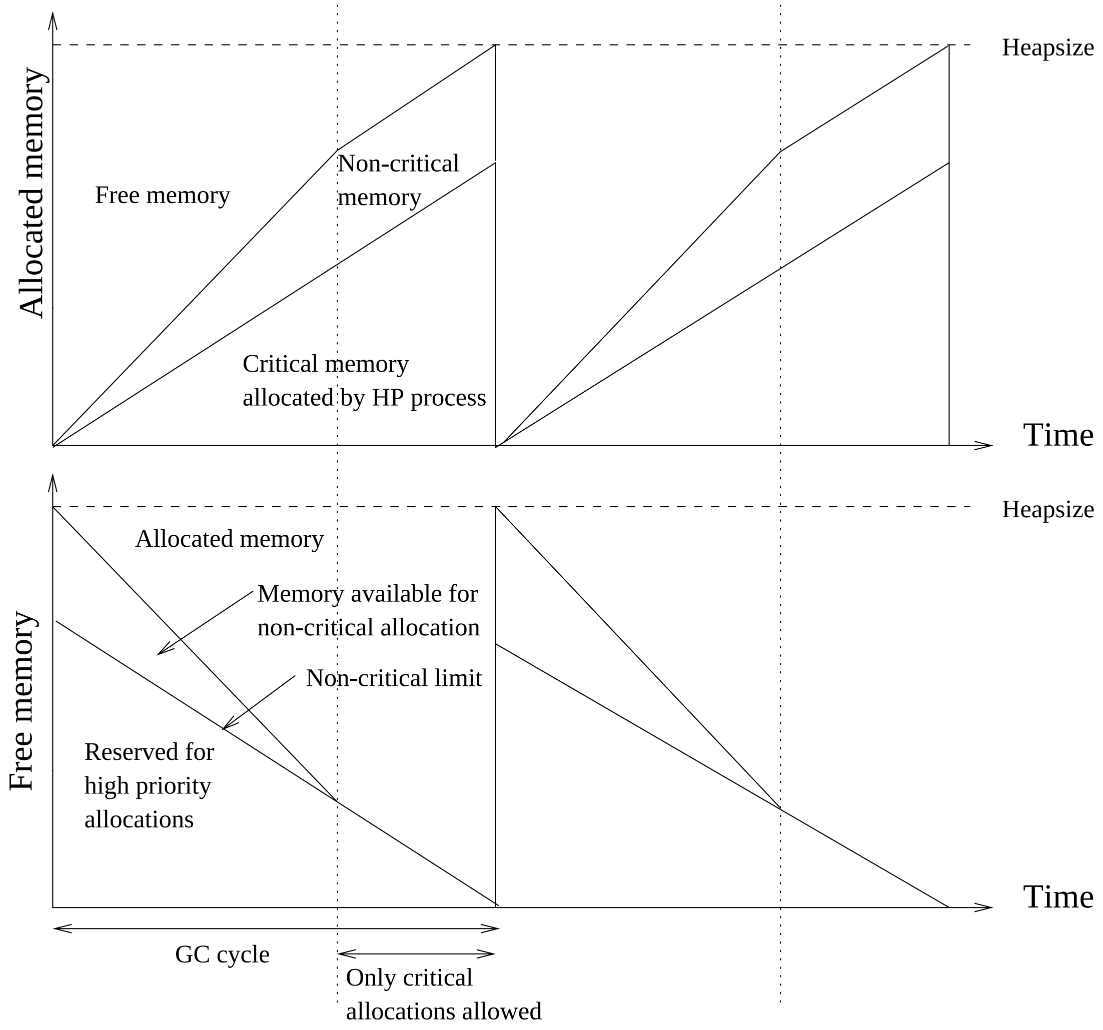
Non-critical limit

Keep the amount of live, non-critically allocated memory below a safe limit

or

Keep the amount of allocatable memory above the safe level

Non-critical limit


Allocated memory
Heapsize
Free memory
Non-critical memory
Critical memory allocated by HP process
Time
Free memory
Heapsize
Allocated memory
Memory available for non-critical allocation
Non-critical limit
Reserved for high priority allocations
Time
GC cycle
Only critical allocations allowed

LUND INSTITUTE OF TECHNOLOGY
Lund University

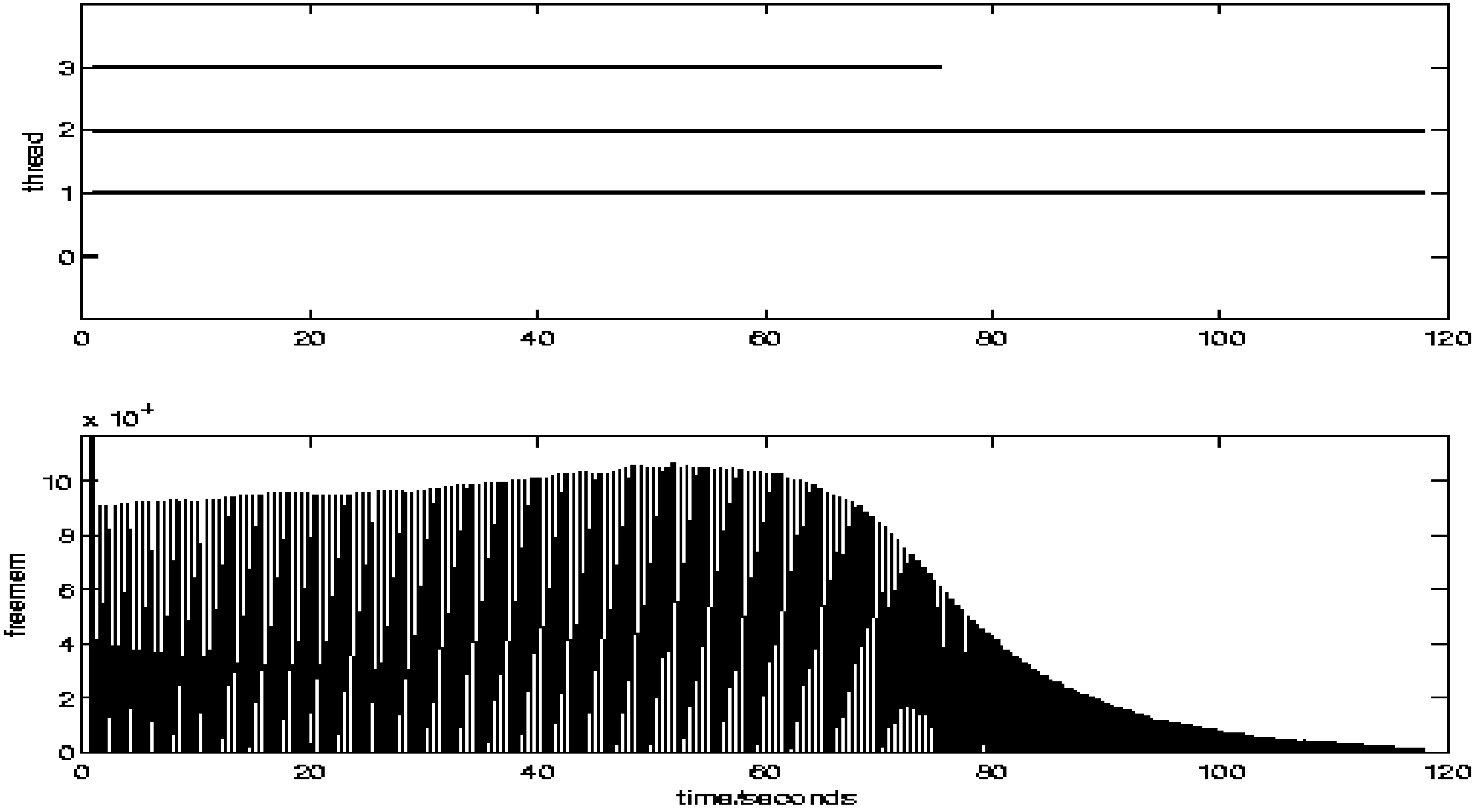
Experimental results:

Simple control application with logging

- Control – critical
- Logging – non-critical

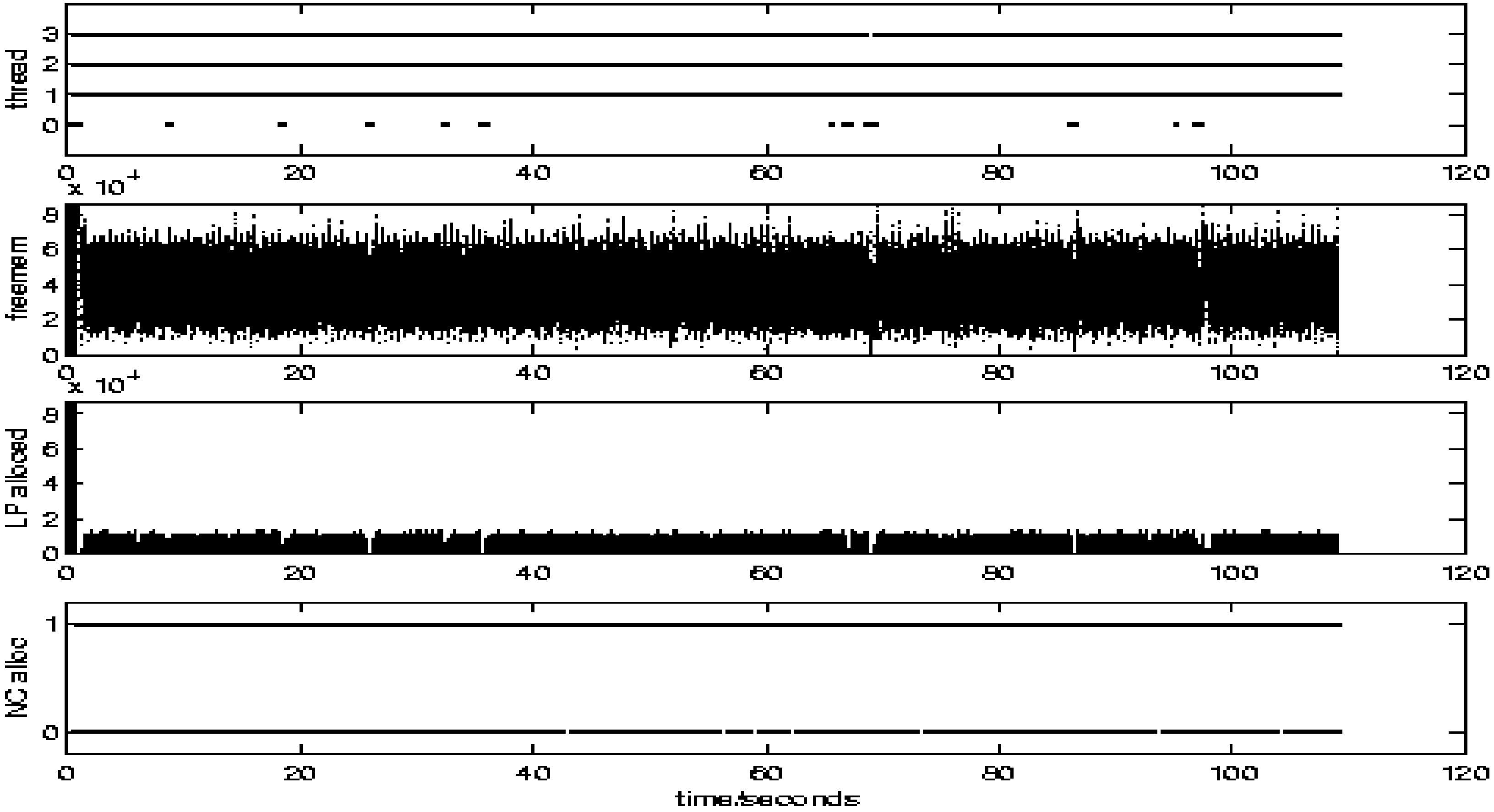
```
void control(){
    calculateControlSignal();
    updateState();
    try{
        deliverLogData();
    } catch(NoNonCriticalMemoryException e) {
        // not enough memory to safely allocate log data
    }
}
```

all allocations critical



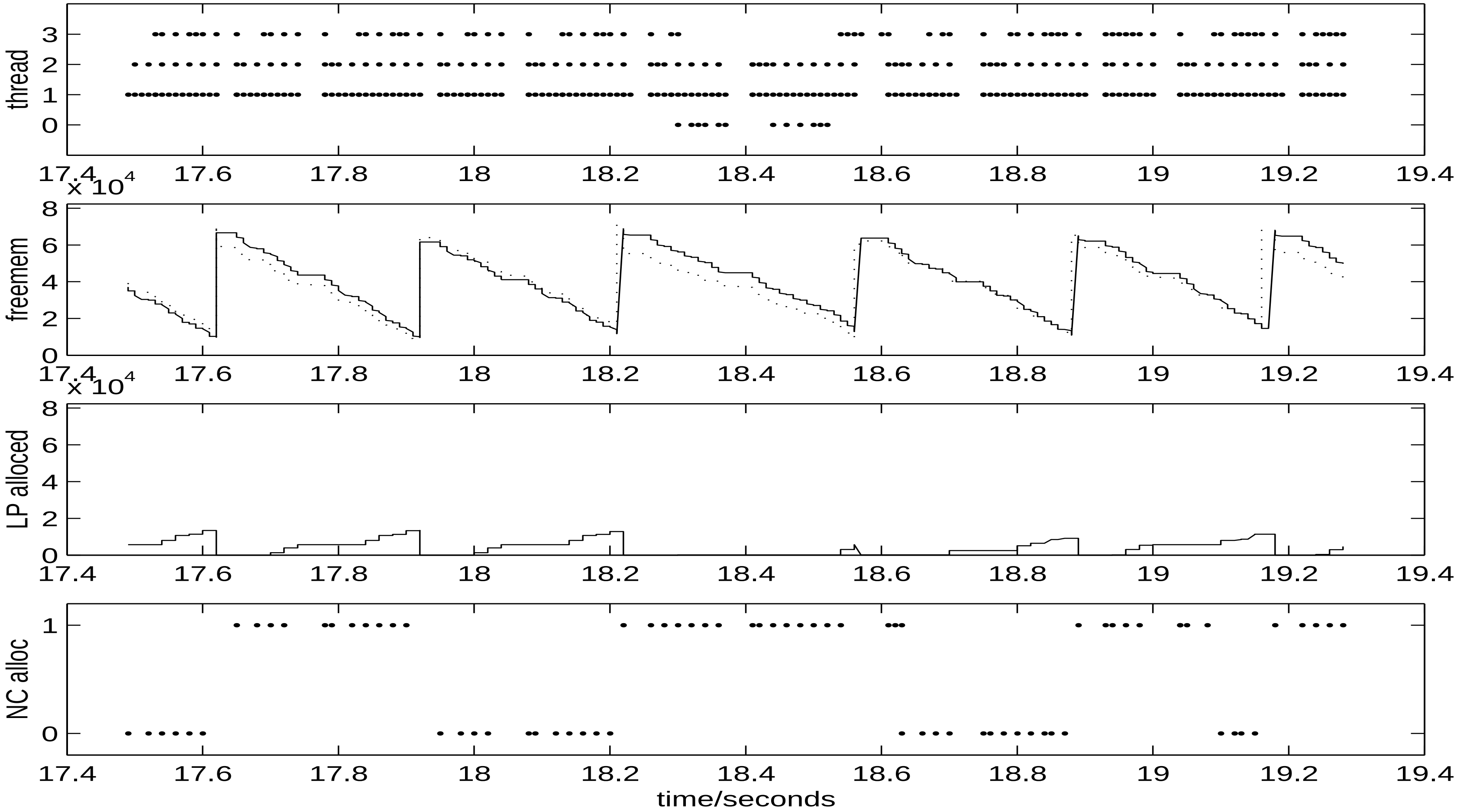
LUND INSTITUTE OF TECHNOLOGY
Lund University

log data is non-critical

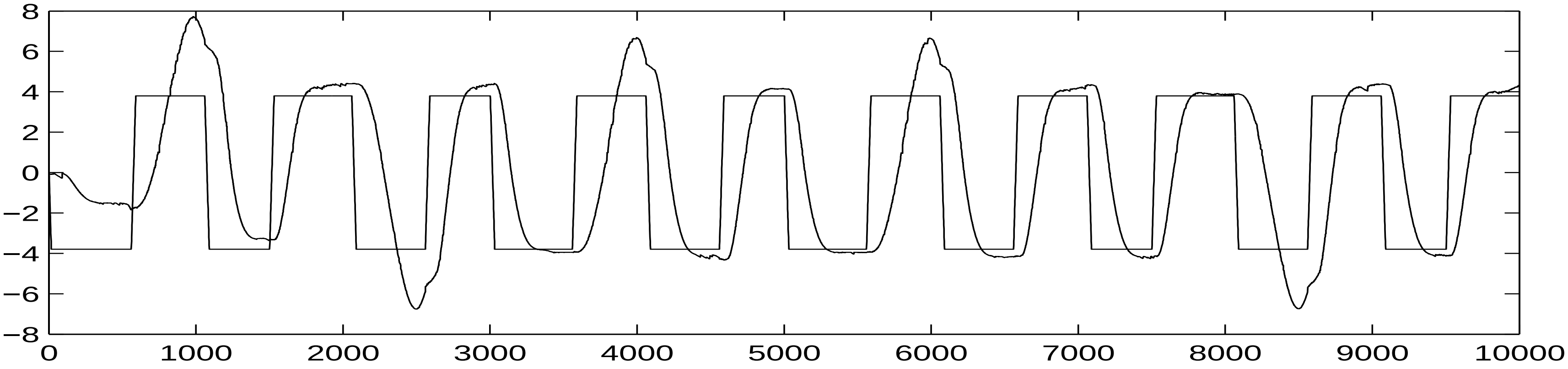


LUND INSTITUTE OF TECHNOLOGY
Lund University

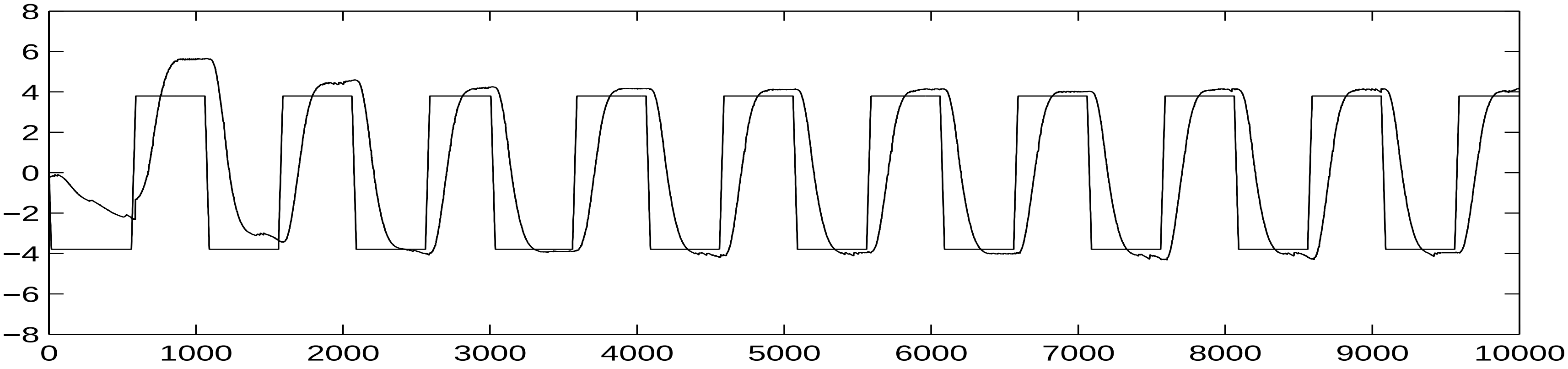
close-up


thread
3
2
1
0
17.4
17.6
17.8
18
18.2
18.4
18.6
18.8
19
19.2
19.4
x 10^4
freemem
8
6
4
2
0
x 10^4
LP alloced
NC alloc
time/seconds

Performance



a) log data objects are always allocated



b) allocation of log data is non−critical

Summary

- Memory requirements can be separated into “critical” and “non-critical”
- Separate memory and CPU time priorities

  Not all of the allocations in a HP process are critical
- Run-time system support
- Improves robustness and performance
- Worst case analysis only needed for critical parts