Time-Triggered Garbage Collection

Robust and Adaptive Real-Time GC Scheduling for Embedded Systems

Sven Gestegård Robertz and Roger Henriksson

(sven|roger)@cs.lth.se

Department of Computer Science

Lund University, Sweden

LUND INSTITUTE OF TECHNOLOGY
Lund University

Outline

- Problem statement
- Background
- Time-triggered GC
- Adaptive GC scheduling
- Summary

Problem statement

From an engineering perspective:

Real-time run-time system for Java

Handle run-time scheduling and schedulability analysis separately

If a set of processes is schedulable, the run-time system should ensure real-time performance.

Problem statement

GC properties

- Predictable and non-intrusive
- Transparent to the application developer

Incremental GC

- GC work scheduled at allocations
- Increment size proportional to object size
- Ensuring sufficient progress

$$w \geq W_{max} \cdot \frac{a}{F_{min}}$$

GC performed in-line with application code may cause long delays

Allocation-triggered GC

Issues:

- Bursty allocation
- GC work metric concerns

GC work metrics

How to express GC work

- Based on known quantities
- Model the temporal behaviour of the GC
- Feasible to calculate at run-time

Example

The evacuation pointer metric

$$W = \Delta B$$

$$W_{max} = E_{max}$$

Problem: A small increment of the metric may take very long time to perform

An improved metric

[Henriksson 98]

$$W = \alpha \cdot roots + \beta \cdot \Delta S + \Delta B + \gamma \cdot \Delta P$$

$$W_{max} = \alpha \cdot roots_{max} + \beta \cdot E_{max} + E_{max} + \gamma \cdot M_{HP}$$

- Requires tuning of α, β and γ
- Still not perfect

Concurrent GC

- No GC penalty at allocations
- Problem: ensuring sufficient progress

Time-triggered GC

- Use time instead of allocation to trigger GC work
- Calculate GC cycle time that ensures sufficient progress
- $T_{GC} = f(H, L_{max}, \{a_p\})$

Time-triggered GC

Properties:

- GC rate independent of application behaviour
- GC can be scheduled as a normal thread
- GC scheduling independent of work metric

One parameter: the cycle time

Adaptive GC scheduling

Manual tuning of GC scheduling parameters

- requires detailed analysis of both GC and application
- is based on worst case analysis
- is not possible if the run-time configuration or platform is unknown

GC cycle time auto-tuning

GC cycle time auto-tuning

A simple model:

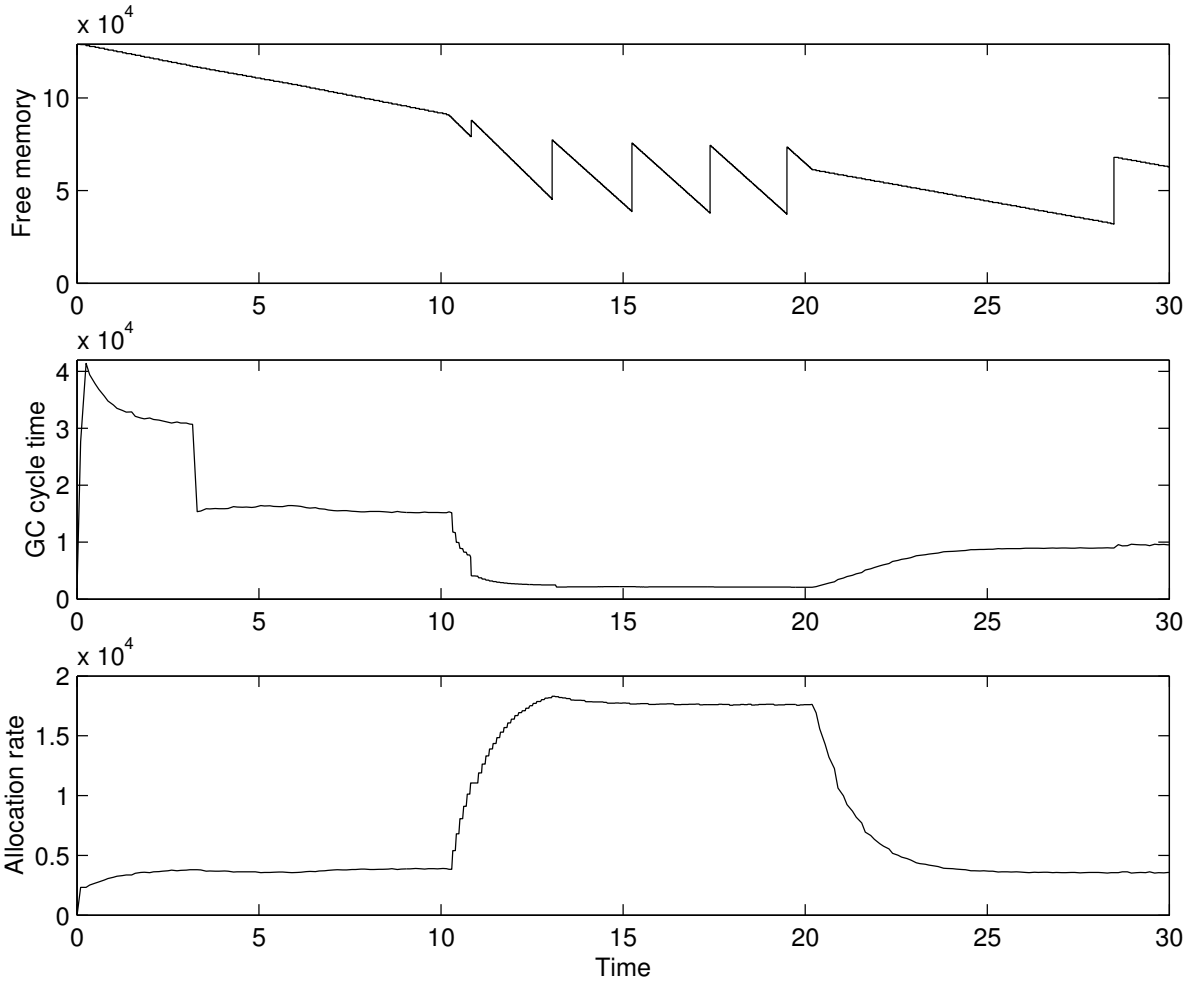
$$T_{remaining\ this\ cycle} = \frac{F}{\dot{a}}$$

$$T_{GC} = \frac{F}{\dot{a}} + T_{elapsed}$$

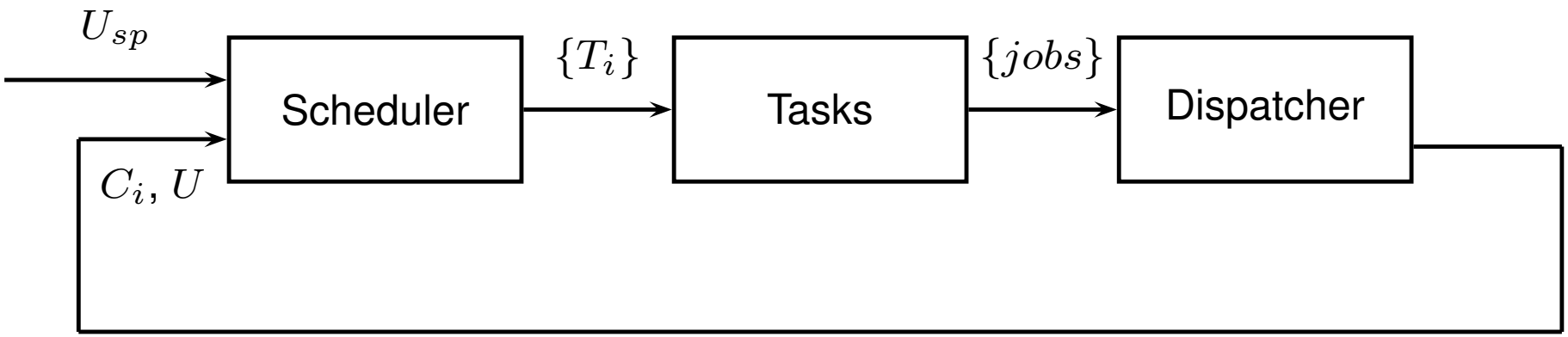
Taking floating garbage into account:

$$T_{GC} = \frac{1}{2}\left(\frac{F}{\dot{a}} + T_{elapsed}\right)$$

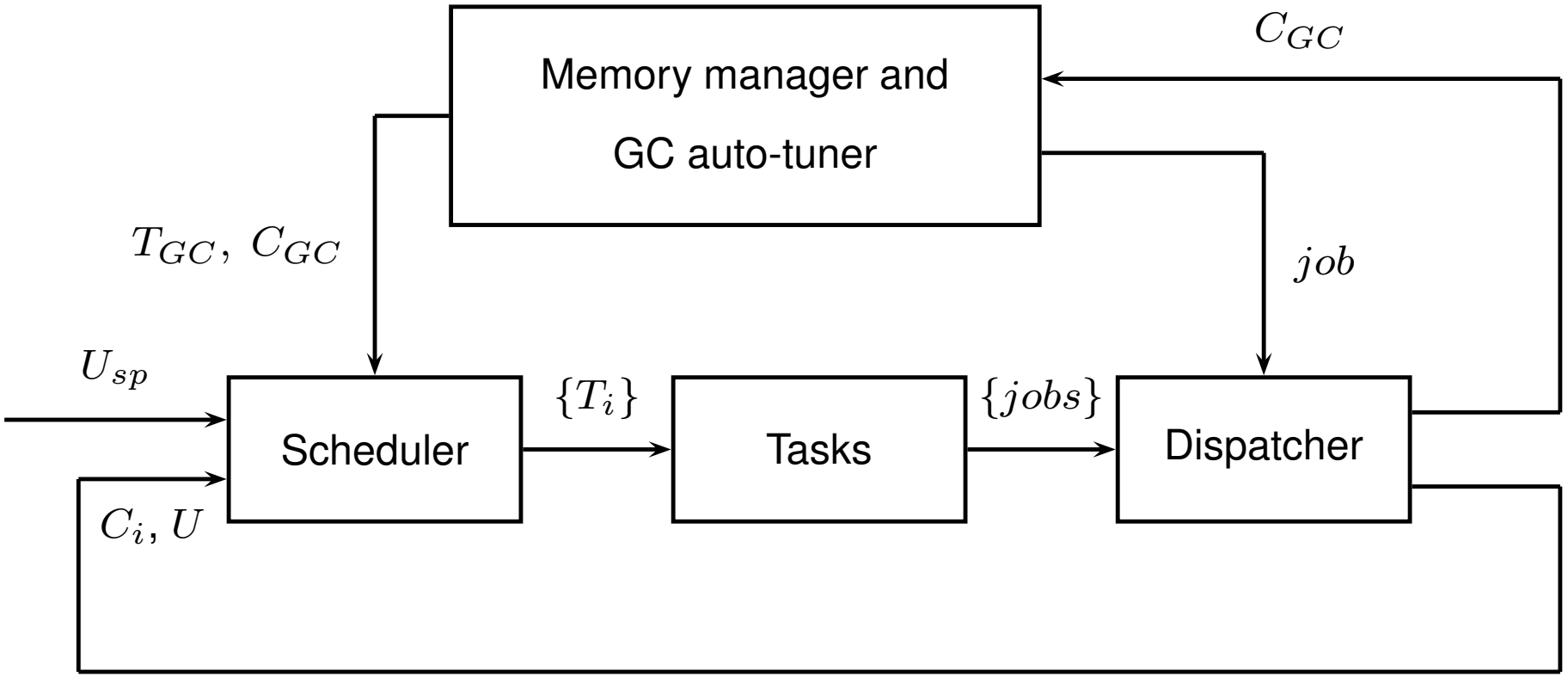
Experiment


x 10^4
Free memory
0
5
10
0
5
10
15
20
25
30
x 10^4
GC cycle time
0
1
2
3
4
0
5
10
15
20
25
30
x 10^4
Allocation rate
0
0.5
1
1.5
2
0
5
10
15
20
25
30
Time

Feedback scheduling



Feedback scheduling



Summary

Time-triggered GC scheduling

- Cycle-level view on GC scheduling
- Low-level scheduling decisions left to process scheduler
- Scheduling independent of GC work metric
- Non-intrusive GC with guaranteed progress under EDF
- Explicit scheduling parameters fits well into feedback scheduling and auto tuning systems
- Proof-of-concept implementation

Future Work

- High performance prototype
- Feedback scheduling
- Distributed systems, composability