Dresden Realtime Operating System (DROPS)

Hermann Härtig, Robert Baumgartl, Martin Borriss,
Claude Hamann, Michael Hohmuth, Lars Reuther,
Sebastian Schönberg, Jean Wolter

Dresden University of Technology

Objectives

- Realtime and *Timesharing*
  - on one system
  - in one application
- *Standard* Hardware Architectures
- A usable system

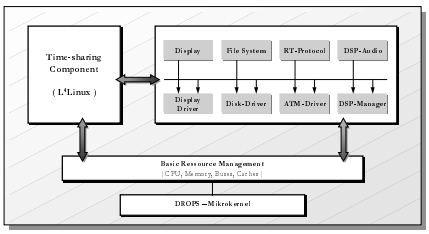
Realtime and Timesharing

- Separation, reduce interference
- Reservation + Adaptation
- Application areas:
  - media applications
  - measurement systems

Today's standard HW is

- Optimized for the average case
- Influenced by
  - memory and bus hierarchies (e.g. caches)
  - intelligent devices
- Hardly predictable

DROPS' Architecture



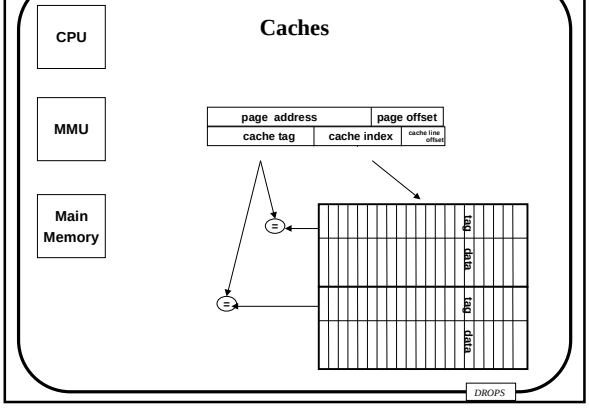
OS Controlled Cache Predictability for Realtime Systems

(Jochen Liedtke, IBM/Watson, Michael Hohmuth, Hermann Härtig)

Objectives

- partition caches to minimize interference
- OS-controlled
  - transparent to applications
  - not relying on cooperation of applications

Caches



DROPS

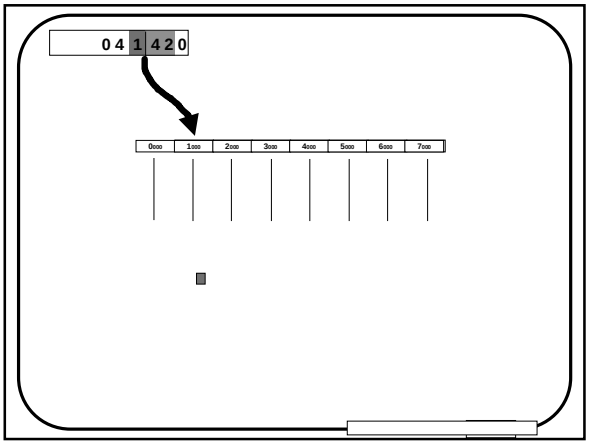
Cache Partitioning

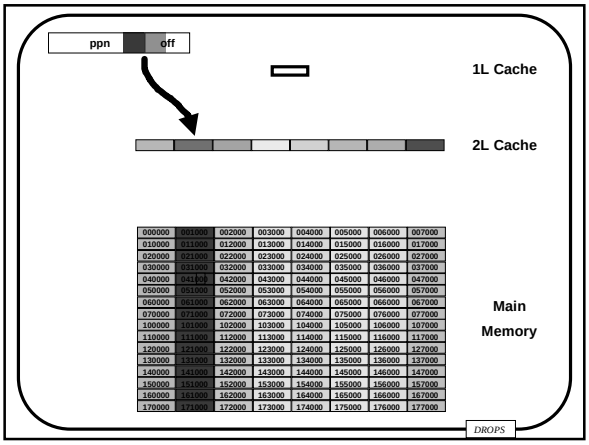
Address regions
that map to a given set of cachelines (a partition)
are used only by one task, thread, etc.

Works with virtually, physically tagged caches

Supported by tools (linkers etc) (Wolfe et. al.)

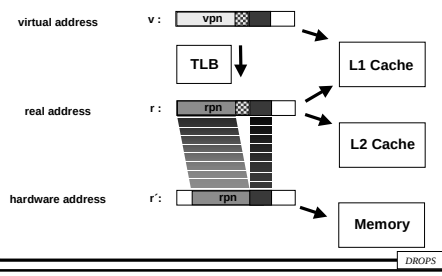
DROPS





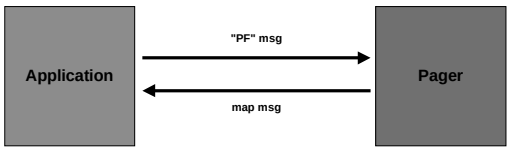
DROPS

Colorable Memory



DROPS

Page Fault Handling



DROPS

A simple Pager

```
DO
    wait for request (requestor, virtual address) ;
    virtual page := virtual address DIV pagesize
    IF requestor is real-time
        THEN source page := 2*virtual page  + 1
        ELSE source page := 2*virtual page
    FI ;
    send (requestor, map (source page * pagesize))
OD
```


Experiment 1

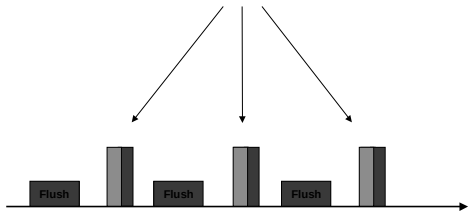
Scenario:

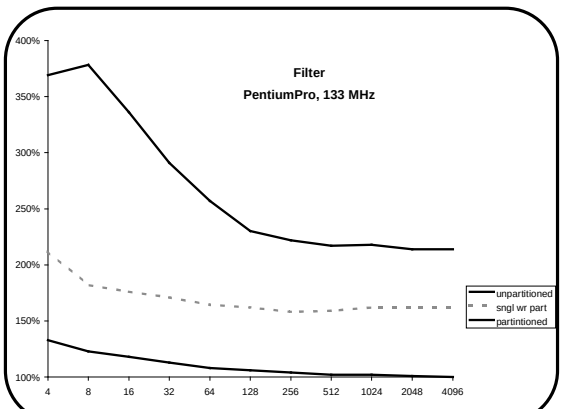
- high prio realtime task
- running in frequent short intervals

Which worst case delays are caused by prior cache uses in other tasks and the os?

Example: Filter of varying size.

Filter





Experiment 2

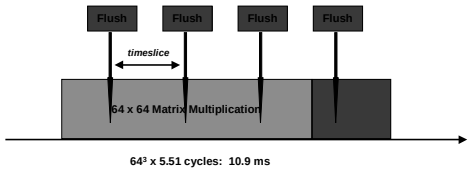
Scenario

- low priority task
- interpupted frequently

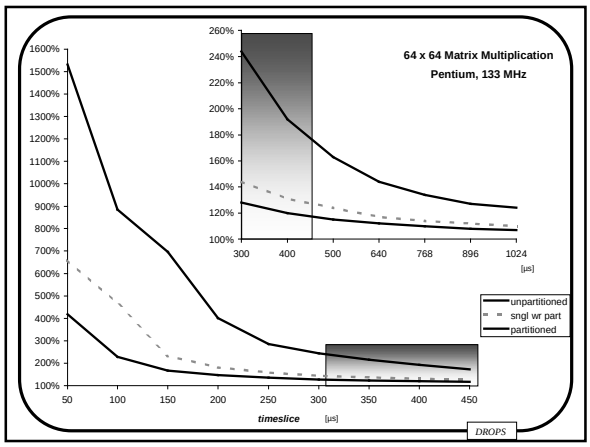
What is the effect?

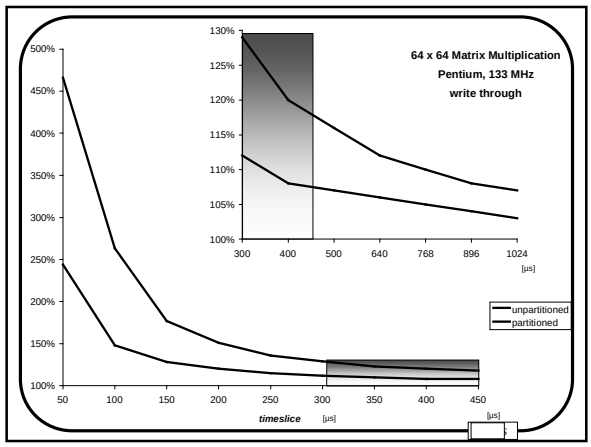
Example: matrix multiplication.

Matrix Multiplication



64^3 x 5.51 cycles: 10.9 ms

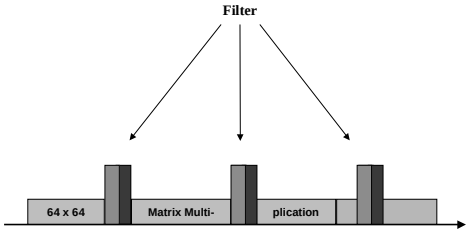




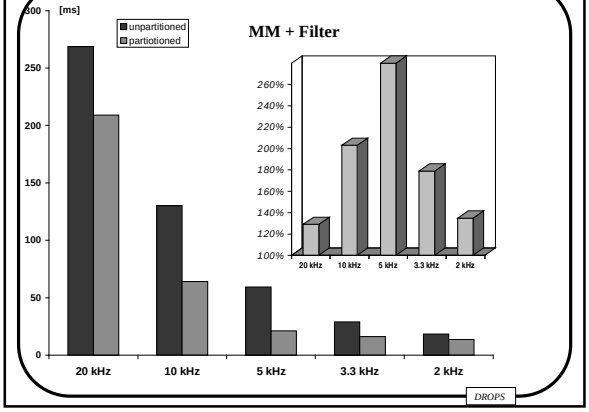
Experiment 3

Combination:
Low priority Task interrupted by known high prio task

DROPS



DROPS



DROPS

Other Penalty Sources

Interrupts
- must be known
- controlled by disabling and scheduling

TLBs
- limited by working set

Memory Refresh Cycles

DMA (-> Schoenberg)

DROPS

Conclusions

?

Effect on more complex tasks?

Interference with other penalties, e.g. bus sharing.

Taming Linux

(Michael Hohmuth, Jean Wolter, Hermann Härtig)

Problems with L4Linux in DROPS:

- grabs all resources
- drives all devices
- disables interrupts
- cache partioning

Resource manager

- Starts system servers besides L4Linux
- controls and hands out system resources
  - RAM
  - interrupt request lines
  - task numbers
- configurable via scripts

Soft Interrupts

- Linux uses cli(), sti() for critical sections
- not acceptable in rt environment

=>

- “soft” interrupts: never disable hardware interrupts
- new implementation of cli(), sti() using queued locks
- exeption: PIC programming

Device Drivers

- Linux drivers do not support QoS

=>

- new drivers
- Linux drivers replaced by stubs (using ipc)
- currently: SCSI, RTFS, ATM

Conclusions

- Linux can be tamed
- L4 => Fiasco (rt/C kernel with L4 interface)