Introduction

Idle periods can be used to do work that will improve overall system performance

Need to know:

- ❏ when idle periods (will) happen
- ❏ how long they will last

Want to be able to say why one detection mechanism is better than another

Introduction

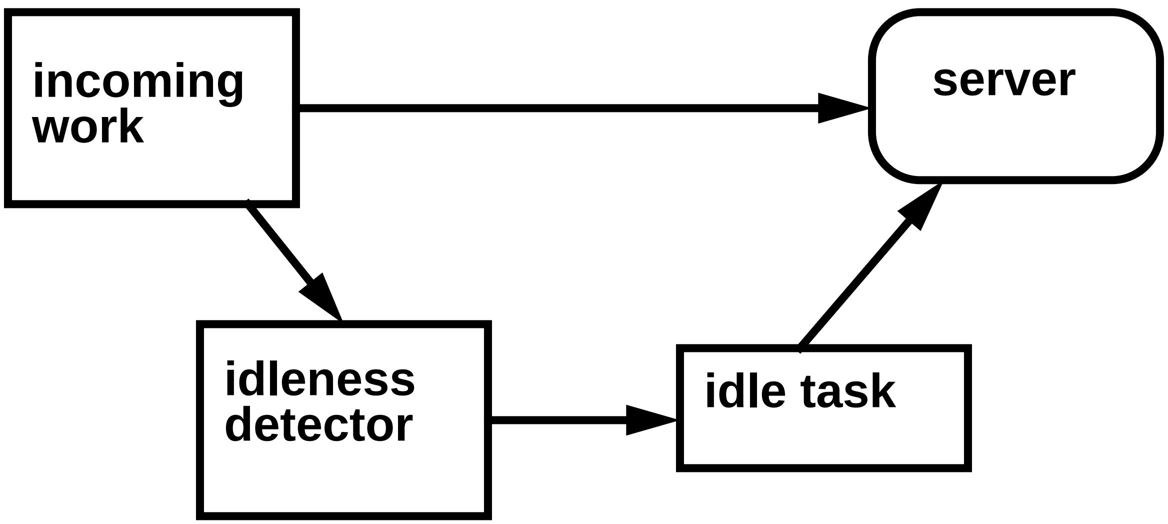
Our approach to using idleness

Medium-term scheduling problem:

Build a *detector* that watches the system

Emits a stream of predictions *(start, duration)*

Use these to schedule *idle tasks*



Introduction

What is different here?

Durations:

- ❏ anticipate when new work will arrive
- ❏ can adjust work to the expectation

Unlike background processing:

- ❏ adding and removing tasks from a system
- ❏ complement each other

Unlike real-time scheduling:

- ❏ no guarantees—best-effort only
- ❏ use little knowledge of other activities

Idle tasks

Some examples

Delay ordinary work

- ❑ delaying writes

Eager work

- ❑ readahead, compilation, cache flushing

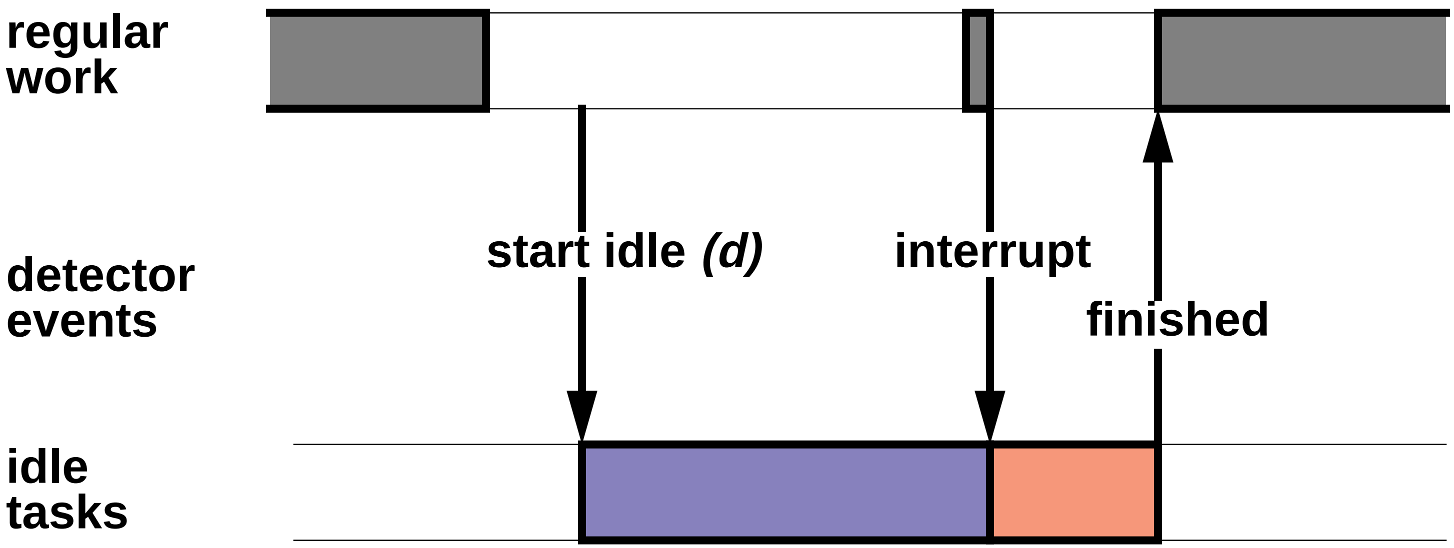
Improve system behavior

- ❑ cache coherence, rebuilding indexes

Load balancing

- ❑ determine lightly-used resources, CPU versus bandwidth trade-offs

Characterizing idle tasks



- ❑ Interruptability *(run to completion, stop early)*
- ❑ Work loss *(redo, undo, checkpoint)*
- ❑ Resource use *(exclusive, shared)*

Characterizing idle tasks

Detailed examples

Spinning down a disk

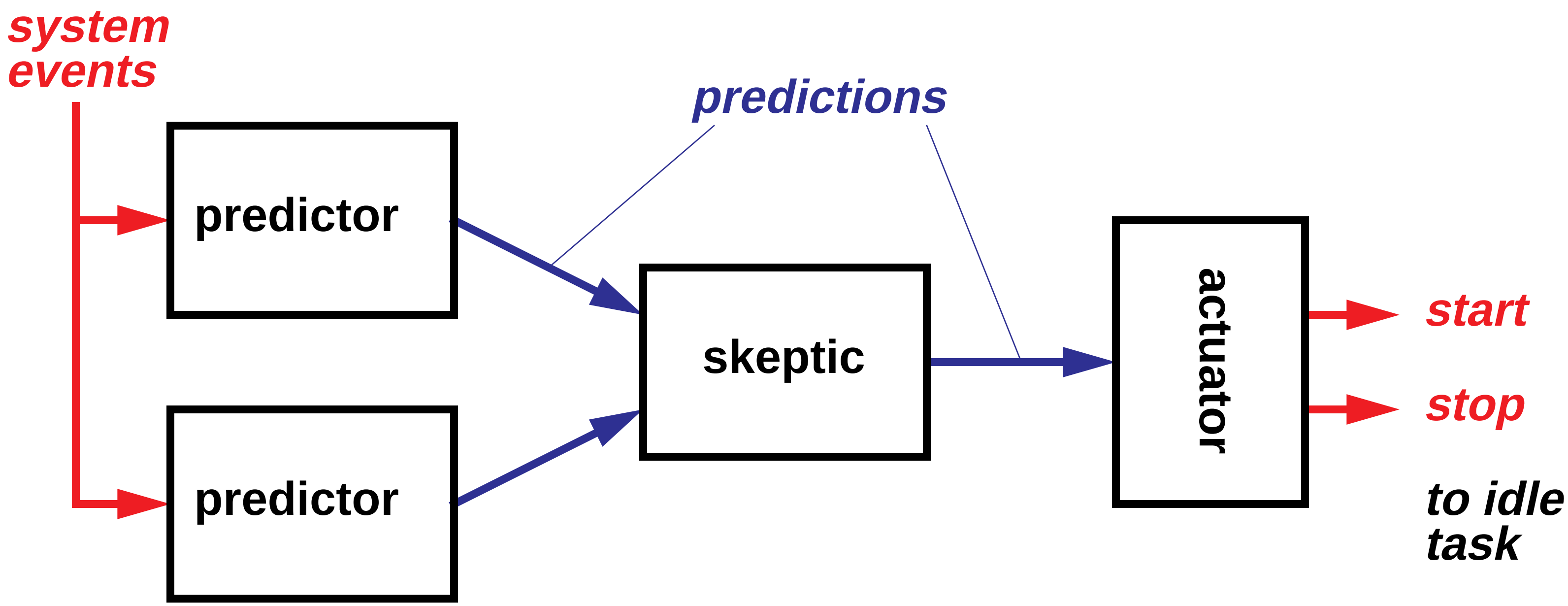
- ❑ task: spin disk down, then wait
- ❑ recovery: spin disk back up
- ❑ “interruptible”, excludes other disk activity

File system reorganization

- ❑ task: reorganize one “chunk”
- ❑ may be interruptible, with loss of work
- ❑ other operations can proceed

Detecting idle time

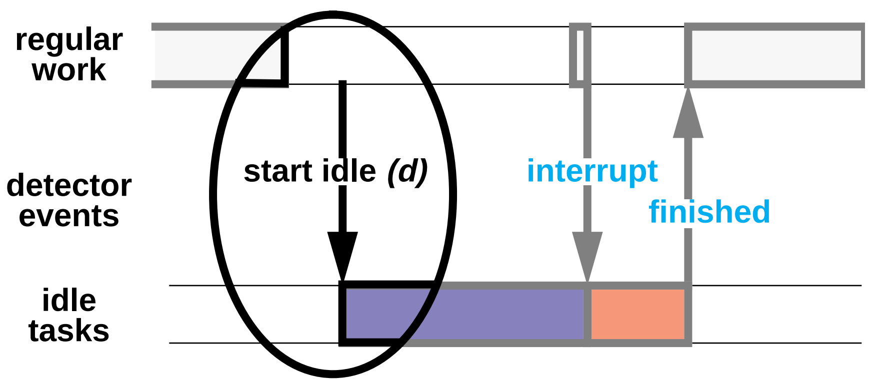
An architecture



Detecting idle time

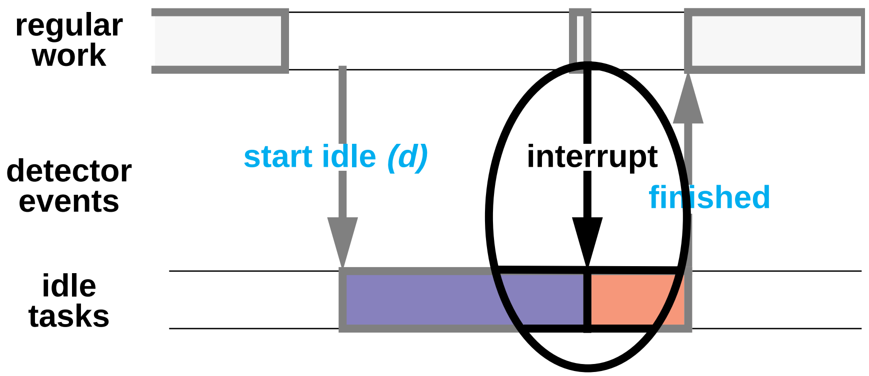
When to start

- ❑ Timer
- ❑ Rate-based
- ❑ Periodic
- ❑ Pattern recognition
- ❑ Adaptive versus static



Detecting idle time
Duration

- ❑ Fixed
- ❑ Moving average
- ❑ Adaptive increase/decrease
- ❑ Pattern recognition
- ❑ “At least” versus “exactly”



Detecting idle time

Using skeptics to improve predictions

Filtering the stream:

- ❑ time-of-day
- ❑ shut off when performing poorly
- ❑ special cases

Combining multiple predictions:

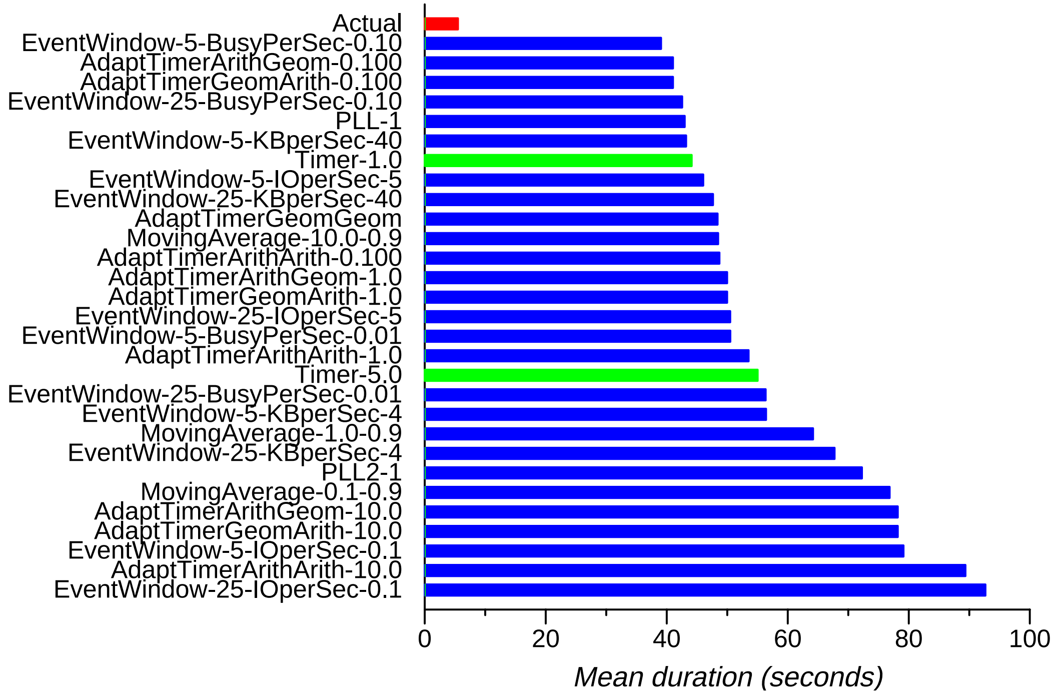
- ❑ quorum voting
- ❑ binomial weighting algorithm

Evaluating idle detectors

Mean idle duration

Only consider start time

Measure *duration* from start to next operation



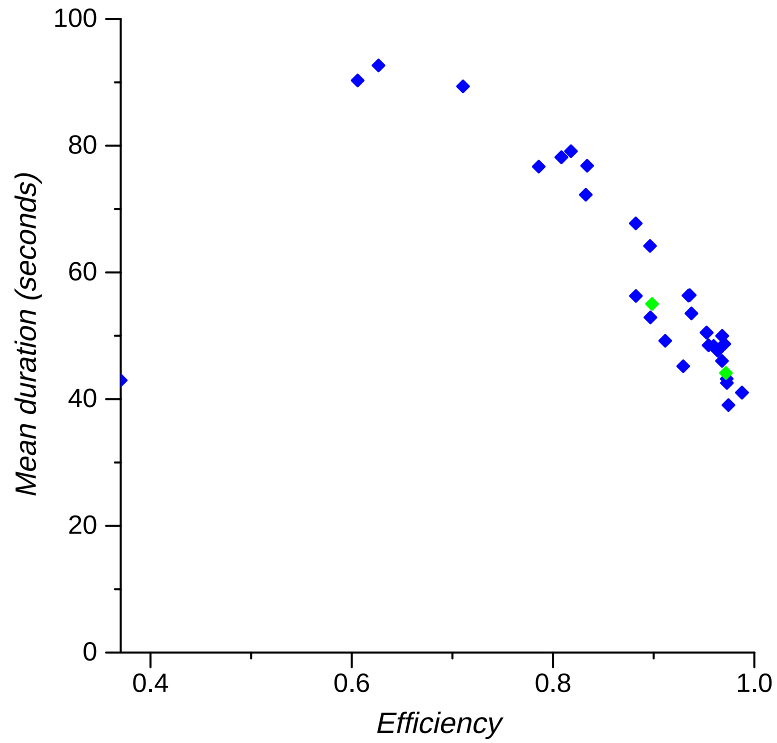
HEWLETT PACKARD

Evaluating idle detectors

Efficiency

For the same data set, compute *efficiency:*

efficiency = predicted idle time / actual idle time

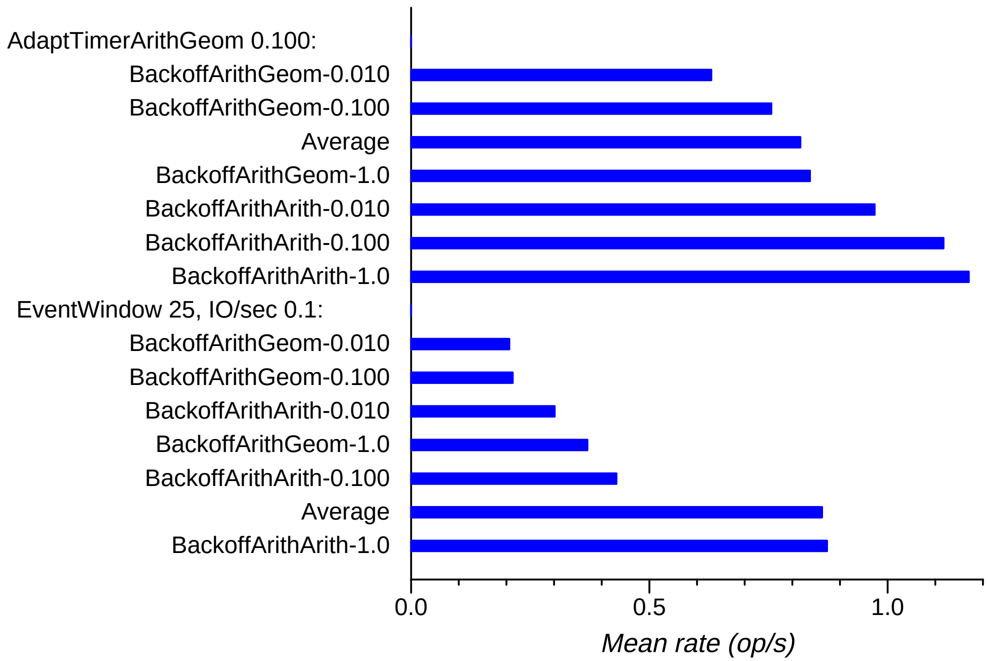


Evaluating idle detectors

How many operations are affected?

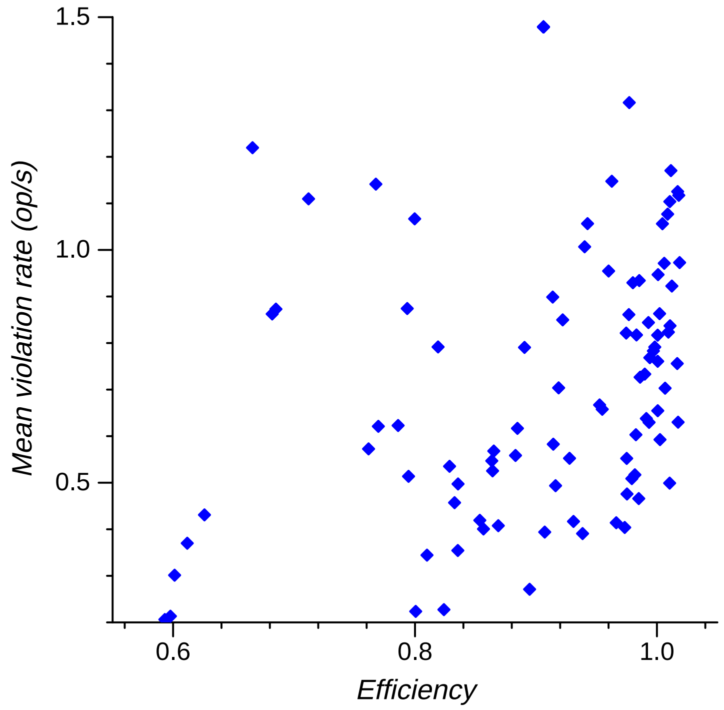
Add duration predictions (and follow them)

Count *violations*



HEWLETT PACKARD

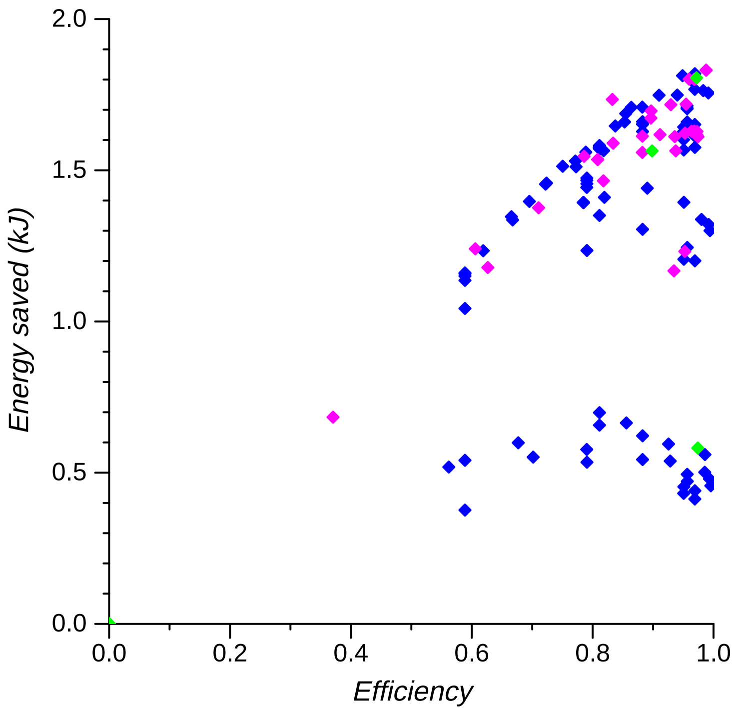
Evaluating idle detectors



Using the detectors for spin-down

Energy savings

Hypothesis: energy savings related to efficiency

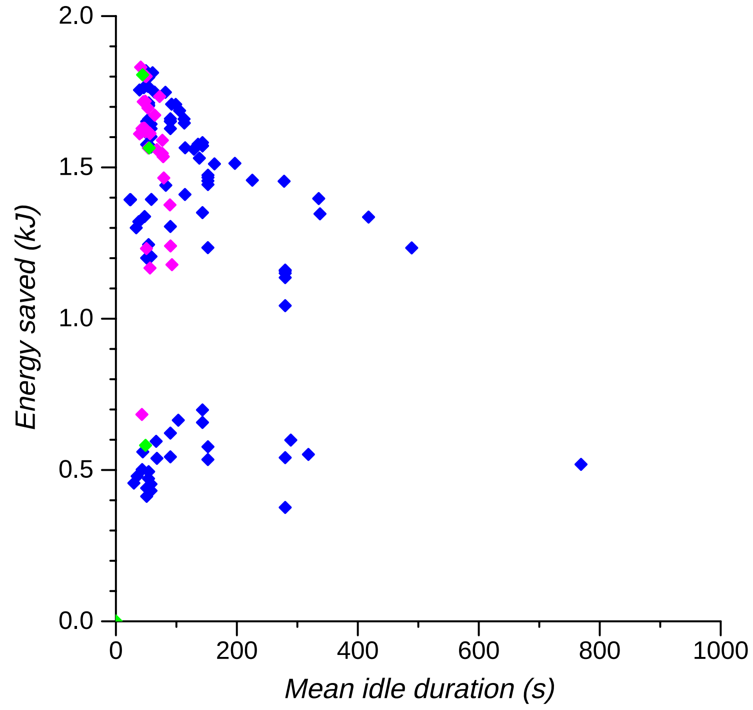


HEWLETT PACKARD

Using the detectors for spin-down

Energy savings

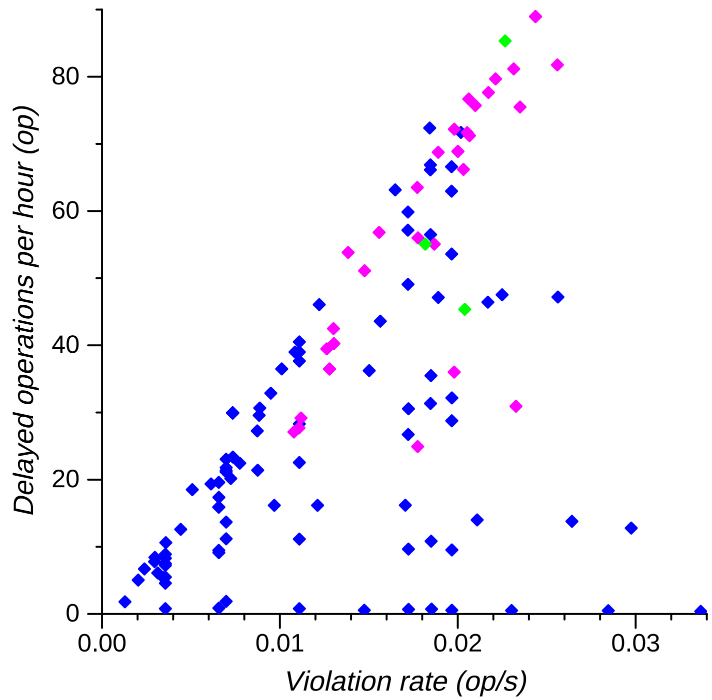
Hypothesis: related to mean idle duration



Using the detectors for spin-down

Number of delayed operations

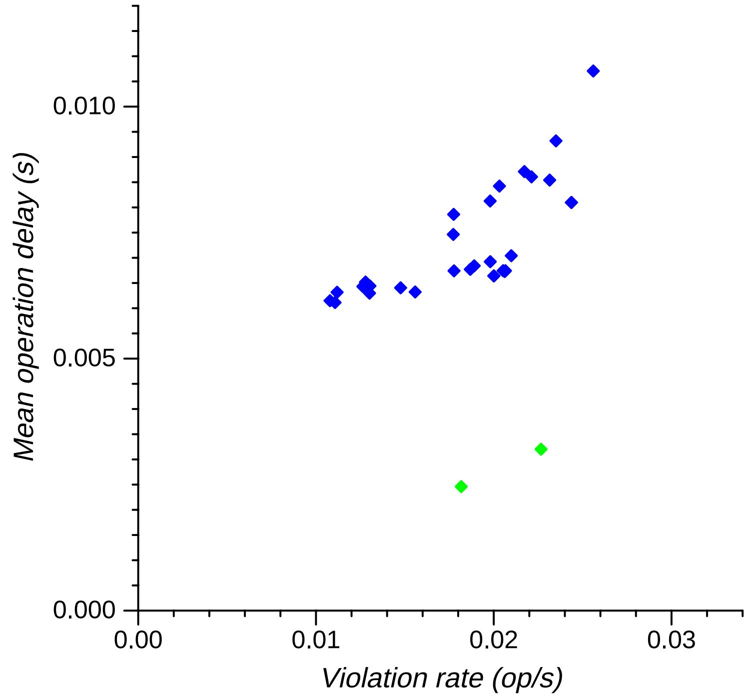
Hypothesis: related to violation rate



HEWLETT PACKARD

Using the detectors for file system reorganization

Hypothesis: intrusiveness related to violation rate



HEWLETT PACKARD

Idleness is not sloth

Conclusions

- ❑ Many opportunities for using idle time productively
- ❑ Taxonomy of idle time helped guide analysis
- ❑ Taxonomy of detection methods helped us find new methods
- ❑ The detectors can be used to schedule realistic idle tasks, and we can evaluate how well they work

Contact: golding@hpl.hp.com

1995 Winter Usenix, New Orleans

Idleness is not sloth

*Richard Golding, Peter Bosch,**
Carl Staelin, Tim Sullivan, and John Wilkes

Hewlett-Packard Laboratories
** Universiteit Twente*

19th January 1995

HEWLETT PACKARD

HEWLETT PACKARD

Slides for — Idleness is not sloth

Richard Golding, Peter Bosch,
Carl Staelin, Tim Sullivan,
and John Wilkes

Concurrent Computing Department
Hewlett-Packard Laboratories

HPL–CCD–95–1
19 January 1995

Slides presented at the Winter Usenix conference in New Orleans from 16–20th January 1995.

This presentation is an overview of our work on using idle time productively, introducing our approach and presenting a few important results. A fuller account can be found in the paper published with the proceedings.