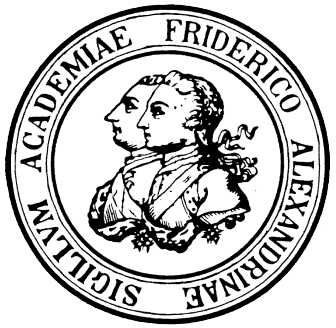
Self-Learning Hard Disk Power Management for Mobile Devices

Andreas Weissel
weissel@cs.fau.de, http://www4.cs.fau.de

Department of Computer Sciences 4
Distributed Systems and Operating Systems
Friedrich-Alexander University of Erlangen-Nuremberg



Frank Bellosa
bellosa@ira.uka.de, http://i30www.ira.uka.de

System Architecture Group
University of Karlsruhe

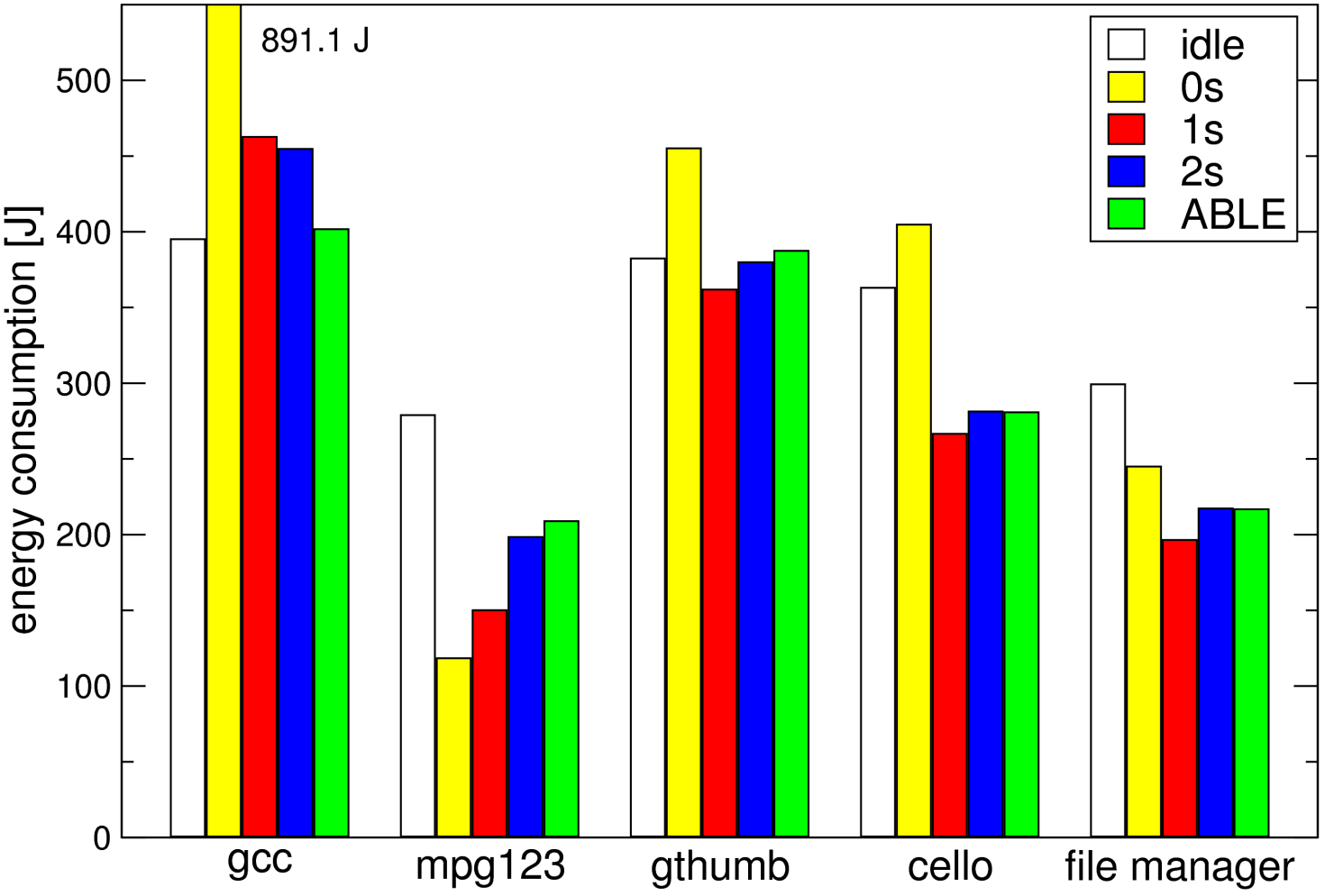
Motivation

- Hard disk power management
  - use low-power operating modes, e. g. *standby mode* (spin down drive motor) if drive is idle
  - mode transitions cause **overhead** in energy and time (up to several seconds)
  - minimum idle period necessary to achieve energy savings
- Spin-down policies
  - fixed or variable time-out before mode transition (filter out short idle periods)
  - adaptive policies predict length of upcoming idle interval based on past hard disk accesses

Motivation: Adaptive Policies

- Example: **Adaptive Battery Life Extender (ABLE)**
  - internal, adaptive algorithm of IBM/Hitachi hard disks
  - the drive intelligently manages the transition between its operating modes depending on the current access pattern
  - the optimal low-power mode and the time before the mode transition are determined dynamically
  - decision based on command history and energy costs associated with each mode
- Deepest power-saving mode can be configured by the user
  - limit impact on performance

Motivation: Energy Savings



Observations

- Different, **optimal** spin-down policies (with respect to energy) for different tasks or applications

- **Application-specific** trade-off between energy savings and performance
  - interactive tasks: delays due to mode transitions may irritate the user

➔ The need for adaptive, application-specific power management

Outline

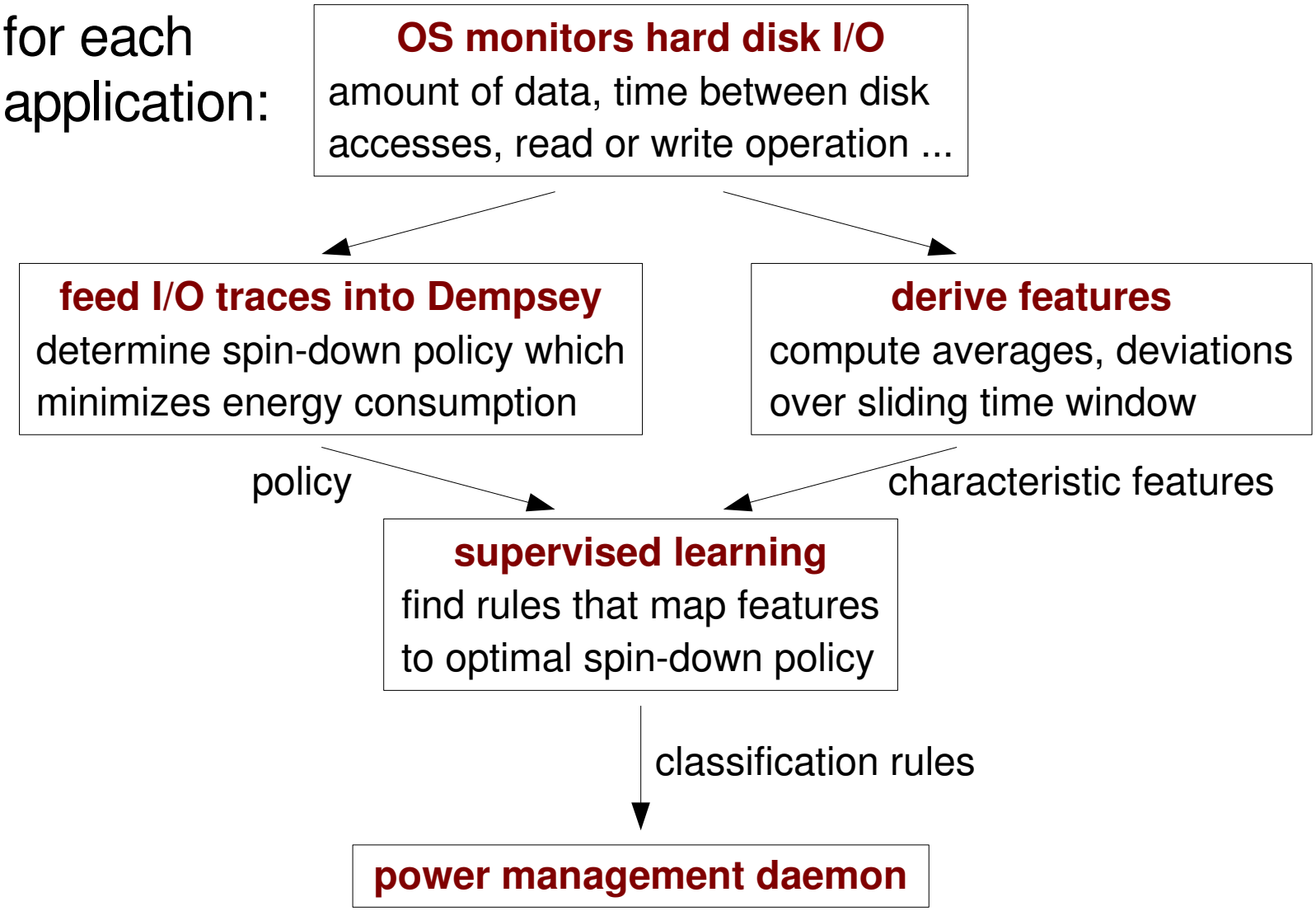
- **Self-Learning Hard Disk Power Management**
- Implementation in Linux
- Energy Estimation Using *Dempsey*
- Evaluation
- Conclusion

Self-Learning Power Management

- Several spin-down policies or low-power modes are available
- Identify optimal, task-specific policy at run-time
  - minimum energy consumption for current workload: determined by simulation environment **Dempsey** (Zedlewski et al., FAST '03)
  - limit on performance degradation
- Infrastructure for workload- or task-specific power management
  - apply techniques from machine learning
  - account for user preferences (sensitivity to transition delays)

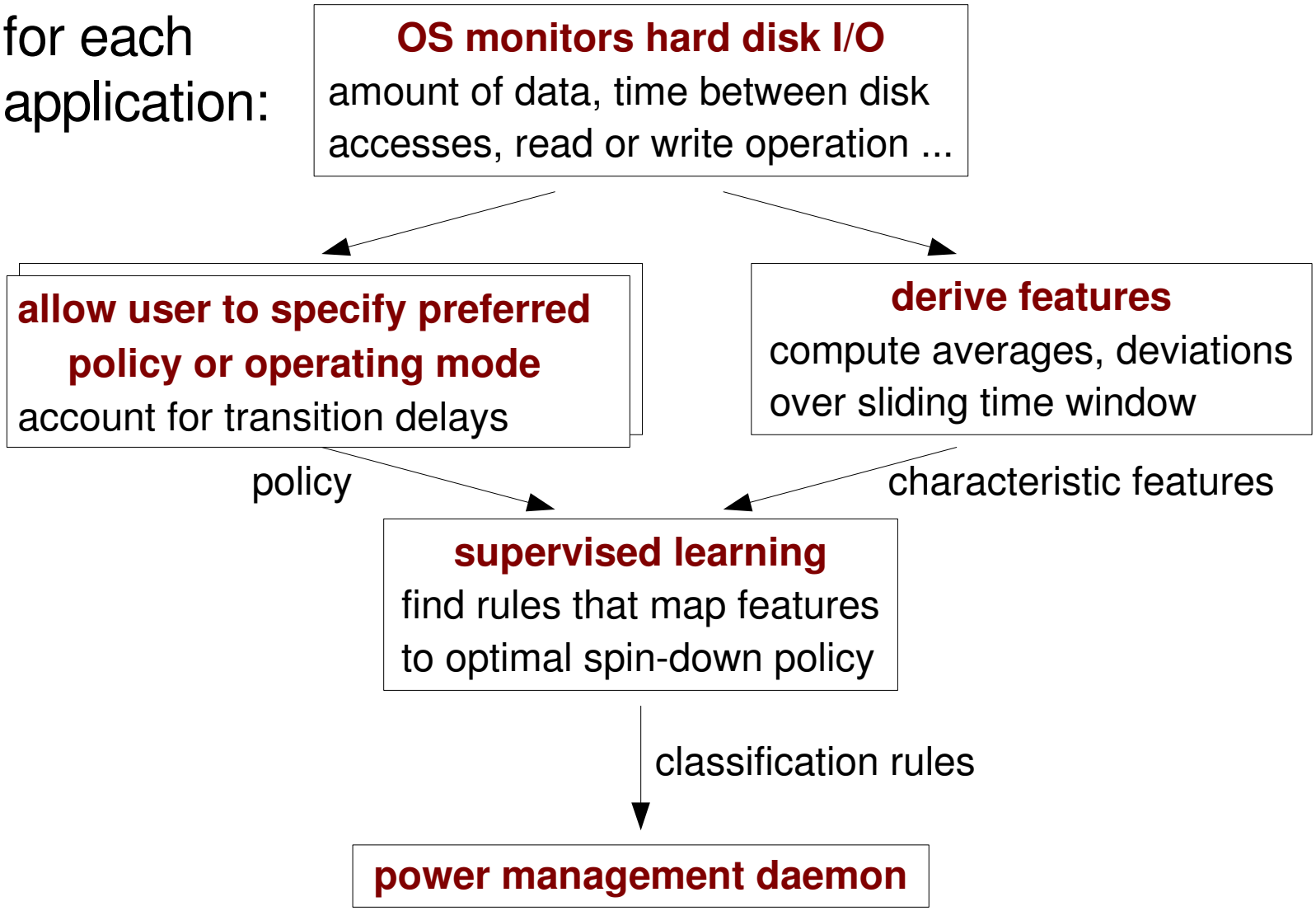
Overview: Training

- for each application:

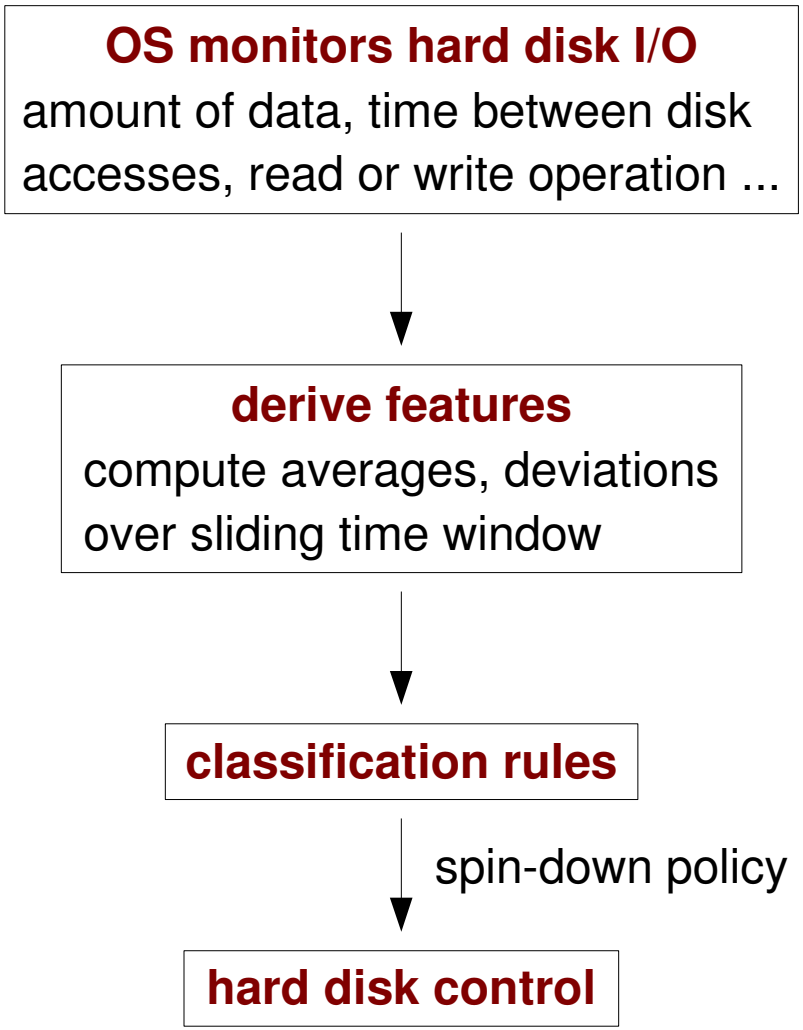


Overview: Training

- for each application:



Overview: Classification



Outline

- Motivation
- Self-Learning Hard Disk Power Management
- **Implementation in Linux**
- Energy Estimation Using *Dempsey*
- Evaluation
- Conclusion

Implementation in Linux

- IDE driver supports different spin-down algorithms
  - currently only policies with fixed time-out
- Operating system monitors hard disk I/O
  - added hooks to I/O-related system calls (`read`, `write`)
  - record time between I/O requests
  - record amount of data read and written (block device driver switch, `generic_make_request`)
  - store data in small ring buffers
  - extension: monitor hard disk I/O per process
- Power management daemon in user space
  - periodically retrieves disk access patterns from kernel, computes features and performs classification

Implementation: Features

- Subset of features that can be used for classification

Number of disk accesses
Number of disk reads
Number of disk writes
Amount of data read or written
Amount of data read (kbytes)
Amount of data written (kbytes)
Number of syscall invocations to read or write data
Number of syscall invocations to read data
Number of syscall invocations to write data
Average time between two hard disk accesses
Average time between two read operations
Average time between two write operations

  - computed over a time window of 10 seconds
  - most significant features are automatically determined by training algorithm

Implementation: Training

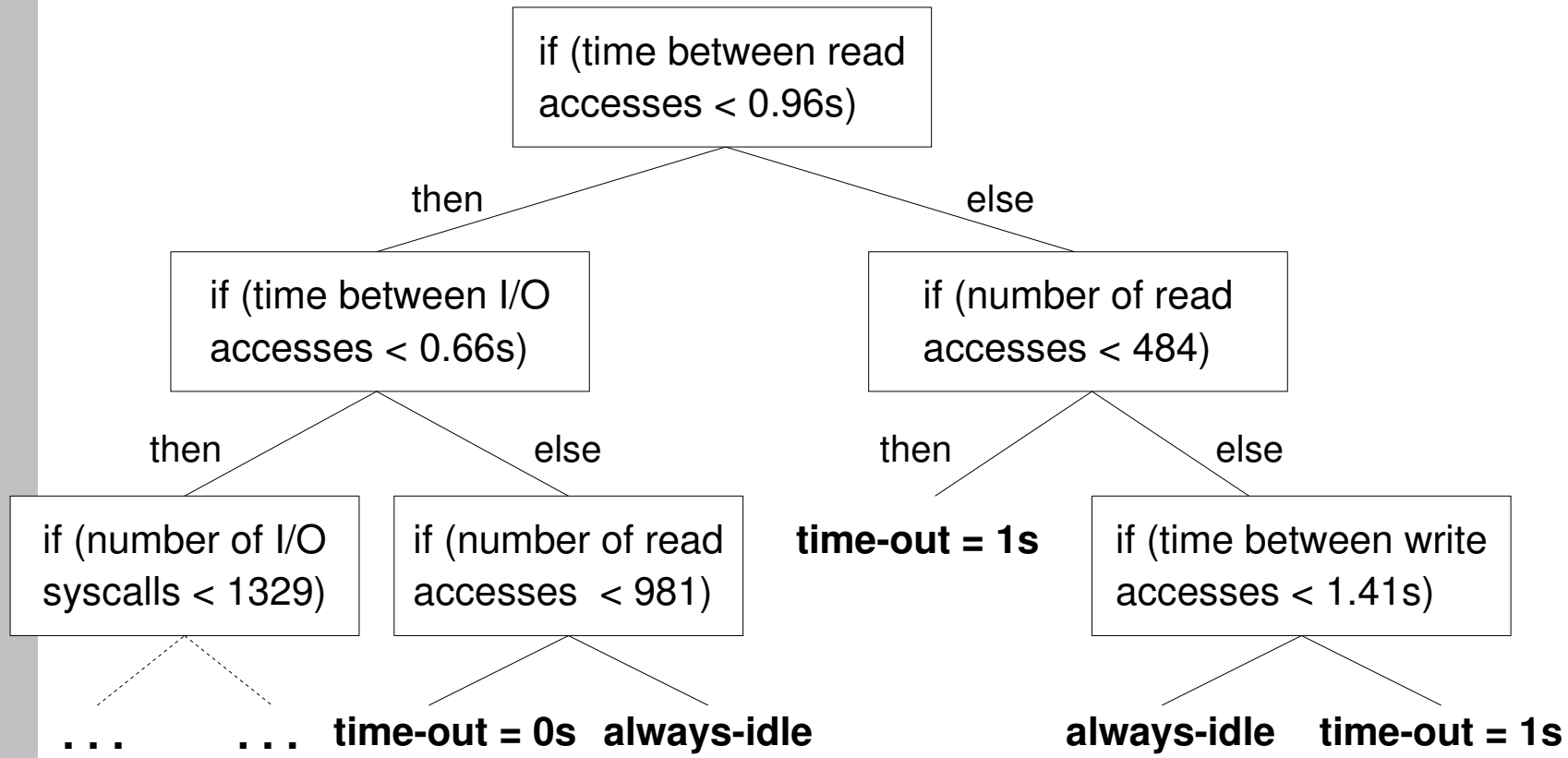
- Edinburgh Speech Tools Library (C++ class library)

- **Classification and Regression Trees**
  - decisions on answers to binary questions
  - questions on elements of feature vector, e. g.:
    ```
    if (average number of disk reads per time window) < 5
    ```
  - questions are ordered in a tree structure
  - use **purity** of a set for ordering, splitting & pruning the tree:
    a set is pure if all of its elements belong to the same class
  - for classification, the questions are processed until a leaf
    is reached: spin-down policy

- Classification tree as sequence of if-clauses
  - implementation as Perl module

Implementation: Classification

- Generated **classification tree**



Dempsey

- Based on **DiskSim** simulator (Ganger et al.)
- Implementation of spin-down policies
- Input
  - configuration file with properties of low-power modes (power consumption, energy and time overhead)
  - trace file of hard disk I/O (time stamp, drive, sector, number of blocks, read or write op.)
- Output: for each spin-down policy
  - energy estimation (error < 10 %)
  - execution time

Outline

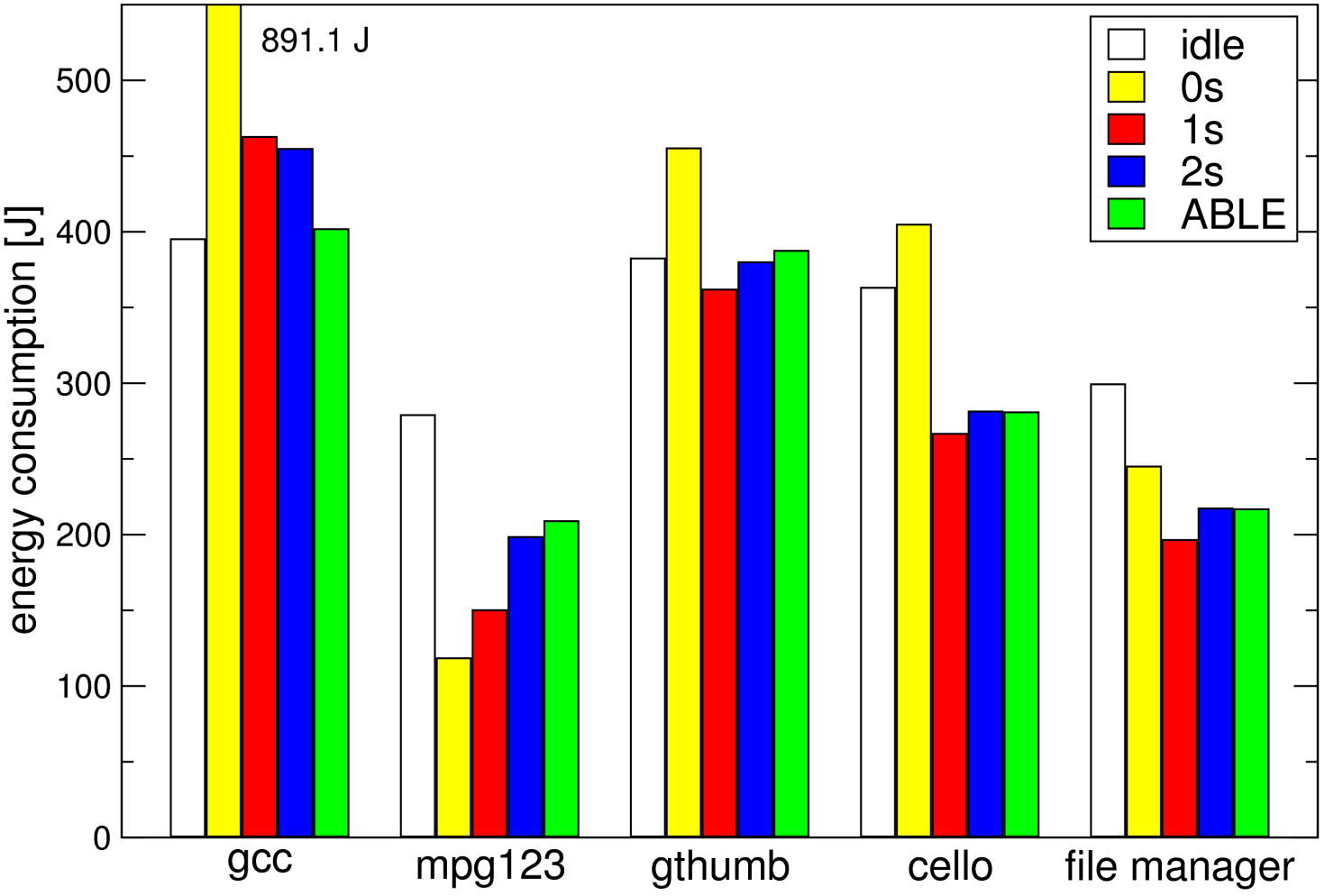
- Motivation
- Self-Learning Hard Disk Power Management
- Implementation in Linux
- Energy Estimation Using *Dempsey*
- **Evaluation**
- Conclusion

Evaluation

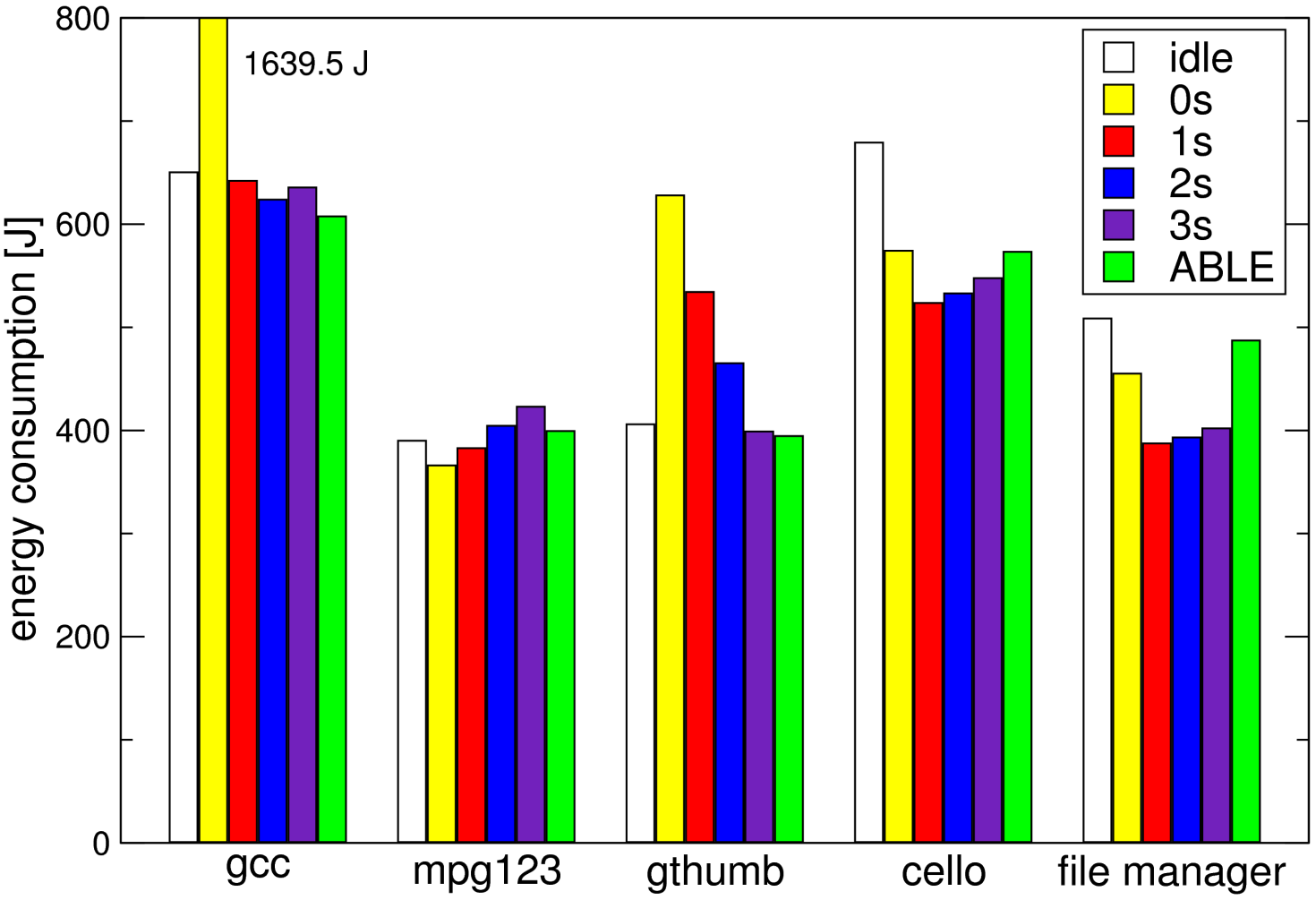
- Measurements of energy consumption using DAQ system
  - Desktop PC, sense resistor in 5 V power lines to hard disk
  - IBM/Hitachi Travelstar 40 GN (20 GB); Microdrive (1 GB)

- Tests (on Linux)
  - `gcc`: compile prototype Linux kernel (2.6.4) using `gcc` 3.4 (7-8 min)
  - `mpg123`: playback of MP3 audio file (128 kbit/s, 9 minutes)
  - `gthumb`: slide show of 140 digital camera pictures, 3s interval
  - `cello`: HP Labs trace file of hard disk accesses (April 18th, 1992), first 10 minutes
  - file manager: trace file of user session with gnome file manager `nautilus` (viewing PDF files, editing text documents, change file access rights), 10 minutes

Microdrive Tests



Travelstar Tests



Runtime Classification

- Train system with additional “idle trace” (with ABLE as the preferred policy)
- Tests with variations of program runs
  - `gcc` compiling Dempsey; different audio files; different slide show interval ...
  - `amp` instead of `mpg123` for audio playback
- Classification errors during start-up activity
  - time window (10s) has to be filled with characteristic values
- Accuracy > 90%
  - best results for compile job and audio playback (regular access patterns)

Applications Running in Parallel

- Extended task structure to maintain I/O statistics **per process**
- Identify appropriate spin-down policy for each task that issues hard disk requests
- Order policies (e. g., according to overhead)
- Successful test with `gcc` and `mpg123` running concurrently
  - hard disk is left in idle mode throughout the compile job

User-Specified Spin-Down Policies

- Influence of power management on application quality
  - no effect on MP3 playback
  - considerable delays for some interactive tasks (file manager test)
- Classification can be performed (partly) by the user
  - Dempsey generates configuration (text) file for training algorithm
  - can be edited by the user
  - future work: user interface
- Test with file manager `nautilus` and `mpg123`
  - always-idle as preferred policy for file manager
  - ➔ error rate ~5 %

Conclusion

- Adaptive hard disk power management
  - the system is automatically trained to identify different tasks and their optimal spin-down policy
  - account for trade-off between energy savings and delays due to mode transitions
- System services for application-specific, adaptive power management policies
  - correctly handle applications running concurrently
  - incorporate user preferences

Thanks for your attention!

Andreas Weissel
weissel@cs.fau.de, http://www4.cs.fau.de

Department of Computer Sciences 4
Distributed Systems and Operating Systems
Friedrich-Alexander University of Erlangen-Nuremberg

Frank Bellosa
bellosa@ira.uka.de, http://i30www.ira.uka.de

System Architecture Group
University of Karlsruhe