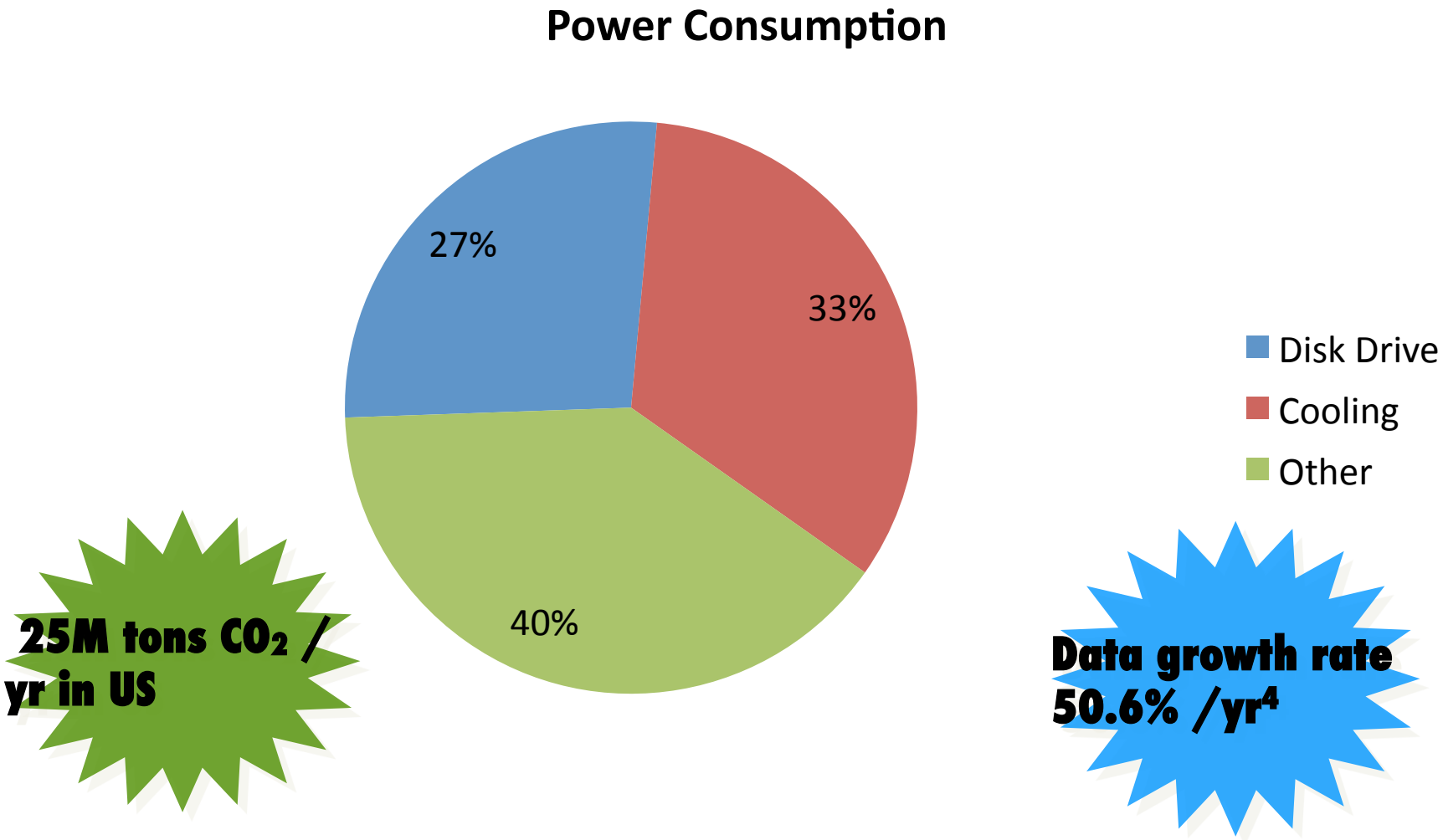
Data Centers, Disks & Power Consumption

Guy Hugot-Derville

Motivation

Annual Data Center Electricity Costs worldwide = $7.2B (2005)



Solution?

Mode[6]	Disk Rotation	Head Movement	R/W interface	Power (W)
Active	On	On	On	12.8
Idle	On	Off	On	7.5
Standby	Off	Off	On	1.5
Sleep	Off	Off	Off	< 1

5-10s ↑

↓ **Whenever disks are idle, spin them down?**

Other Solutions?

- Use multiple rotational speed disks
  - IBM Ultrastar 36Z15
  - At 3,000RPM: 6.1W
  - At 15,000RPM: 13.5W
- Do we save power?
  - Remember: Energy = Power X Time
  - We want to minimize Energy
  - Works for a 20% workload
  - We need to predict the workload!

Plan

- Introduction
- Predicting the writes
  - Hibernator
- Directing the writes
  - LFS-based solution
  - Write Off-Loading
- Evaluation
- Conclusion

Hibernator

- Formalization
  - Constraints
  - Poisson Distribution
  - How do we get E_{ij}?
  - How do we get R_{ij}?
- Adaptative Layout
  - Small Scale Reorganization
  - Large Scale Reorganization

Large Scale Reorganization 1

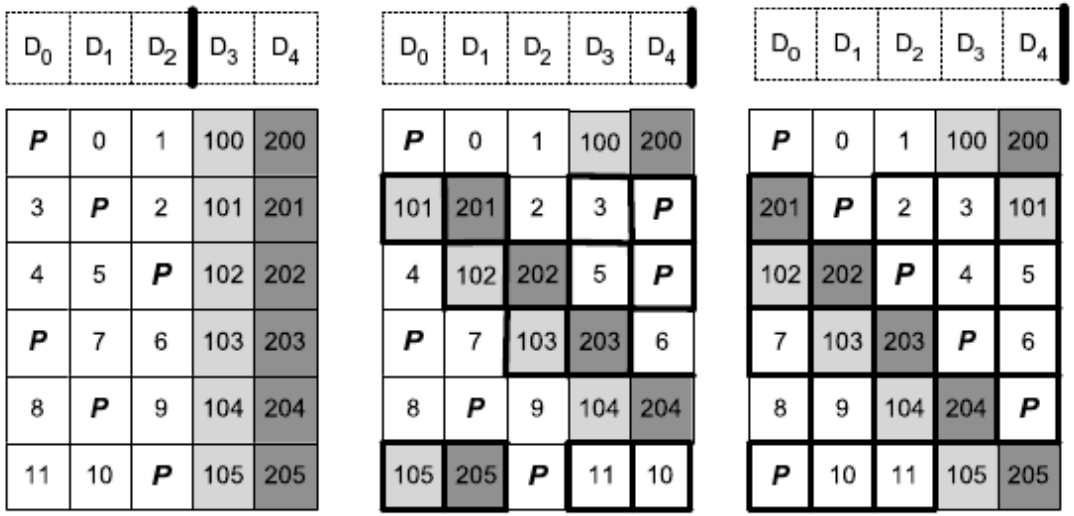


Figure 4: Large-scale reorganization solutions: PS and SS.

- Lighter = smaller = hotter
- Permutational Shuffing
  - Newly added disks to old ones
  - Few relocated blocks
  - Load uneven
- Sorting Shuffing
  - Blocks first sorted
  - Rotational shuffing
  - Big overhead
  - Load even

Large Scale Reorganization 1

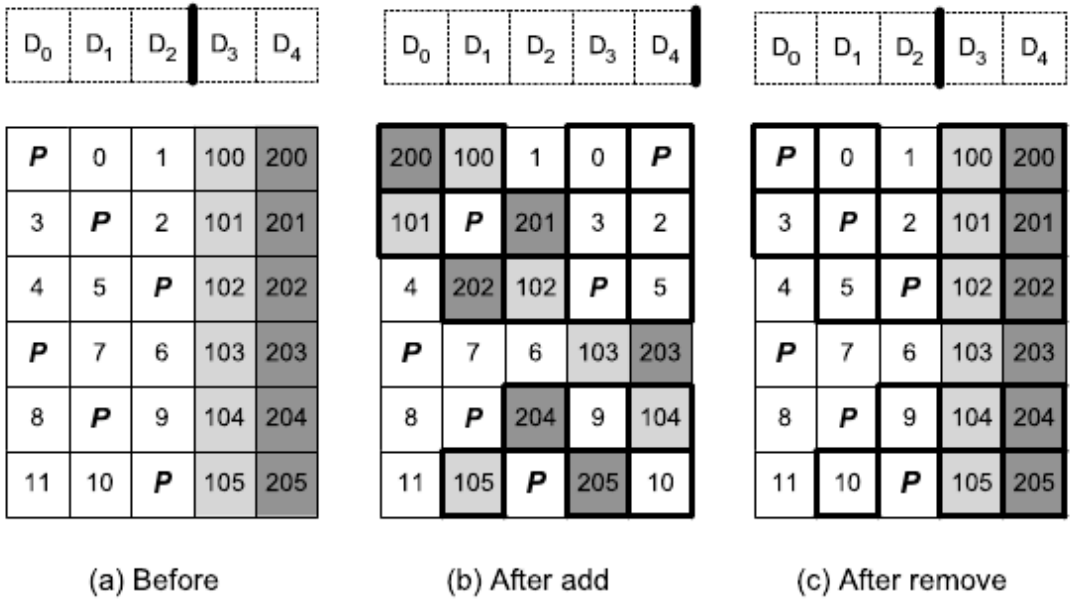


Figure 5: Randomized Shuffling. The graphics have the same

- Randomized Shuffing
  - Fixes both problem
  - 2m migrated blocks for m stripes as in PS
  - Load even because random

Small Scale Reorganization

- Avoiding Hot Spots
- Data into fixed-size relocation blocks (RB)
- Temperature of each RB is maintained
- RB are moved down or up a tier depending on their relative temperature

Setting the speed of disks

- Disk speed is adapted
  - We know the previous disk utilization
  - We predict the future disk utilization
- Coarse-grain Response (CR)
  - Avoid frequent spin up and down
  - T_{epoch}: fixed time during which speed is constant
- Trade-off
  - Responsive
  - MTTL and power cost amortization of changing speed

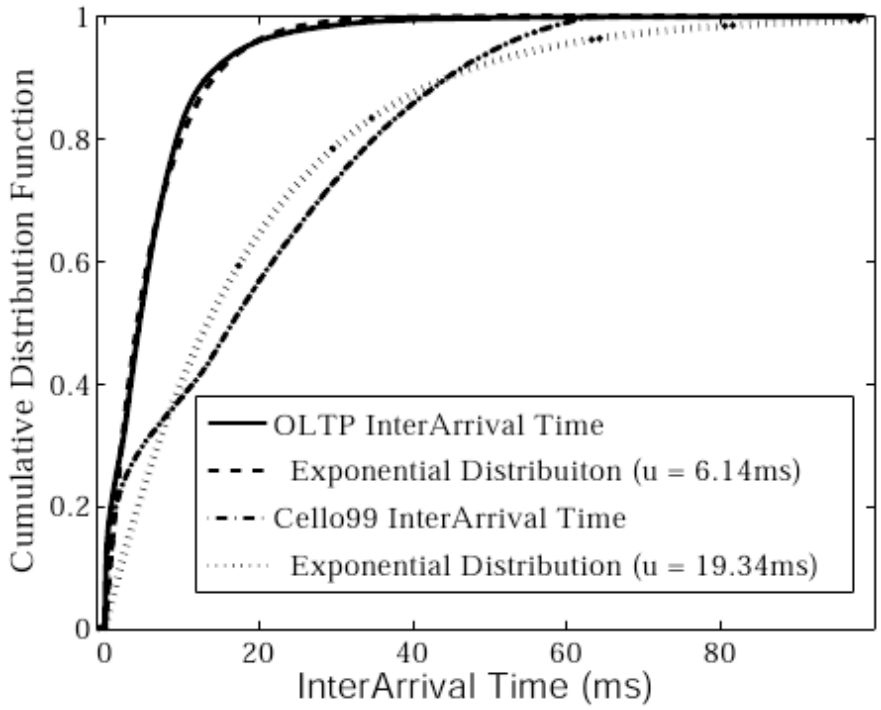
Constraints

- Energy
  - The less energy we spend, the better
  - Total energy is the sum of all the disk energy
- Response time
  - Mean response time inferior to a given limit: R_{limit}
  - Average weighted by the number of request number on each disks

$$\textbf{minimize} \qquad \sum_{i=0}^{n-1} E_{ij}$$

$$\textbf{subject to} \qquad \sum_{i=0}^{n-1} (N_i \times R_{ij})/N \leq R_{limit}$$

Notations



- Poisson distribution
- t_{ij} service time
- $Exp(t_{ij})$: average
- $Var(t_{ij})$: variance
- α_i: request arrival rate
- $\rho_{ij} = \alpha_i Exp(t_{ij})$: disk utilization

How do we get E_{ij}?

- Three terms
  - Remember: Energy = Power * Time
  - Active: servicing requests
  - Idle: no requests
  - Transition between two speeds

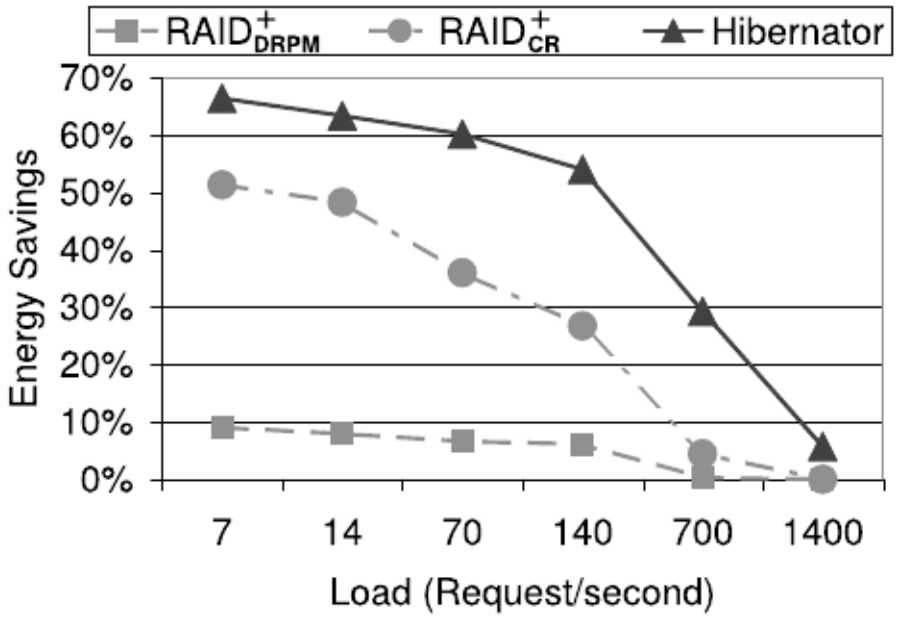
$$E_{ij} = P'_{ij} \times T_{epoch} \times \rho_{ij} + P''_{ij} \times (T_{epoch} - T_{epoch} \times \rho_{ij} - T_i) + P'''_{ij} \times T_i$$

How do we get R_{ij}?

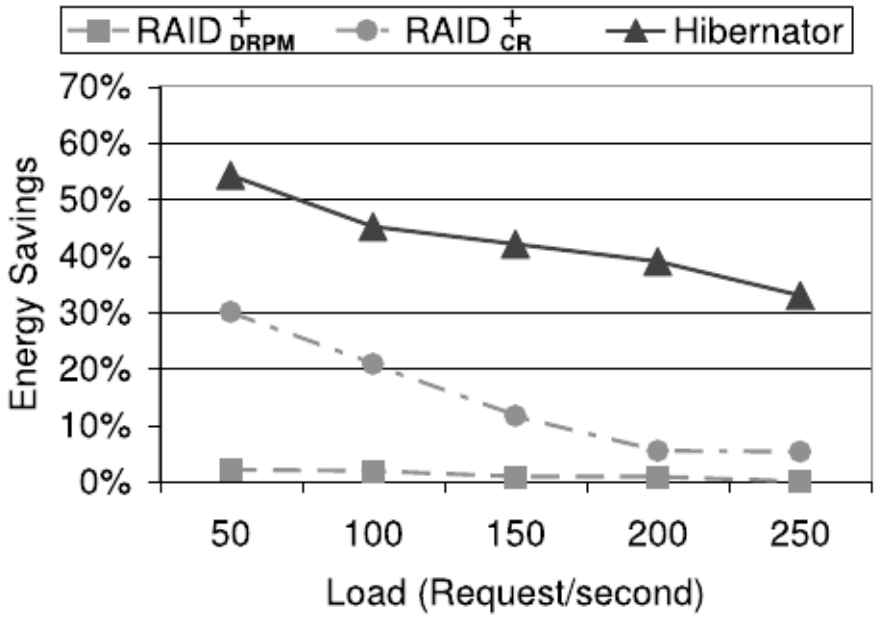
- Two terms:
  - Disks are spinning up:
    long delays $$R'_{ij} = \frac{T_i}{2} + \frac{\alpha_i T_i Exp(t_{ij})}{2} = \frac{T_i}{2}(1+\rho_{ij})$$
  - Normal usage:
    short delays $$R''_{ij} = \alpha_i Exp(t_{ij}) + \frac{\alpha_i^2 (Exp^2(t_{ij}) + Var(t_{ij}))}{2(1-\alpha_i Exp(t_{ij}))}$$
- We do the weighted average of both terms

$$R_{ij} = \frac{\alpha_i T_i R'_{ij} + \alpha_i (T_{epoch} - T_i) R''_{ij}}{\alpha_i T_{epoch}}$$

Power Consumption – Hibernator 1

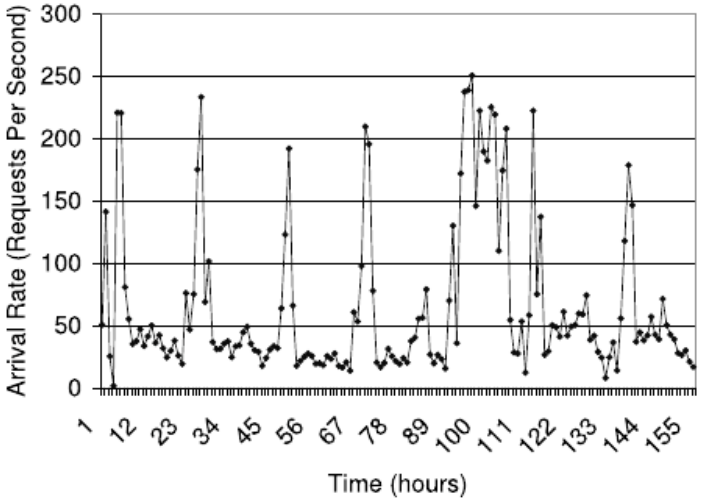


(a) OLTP

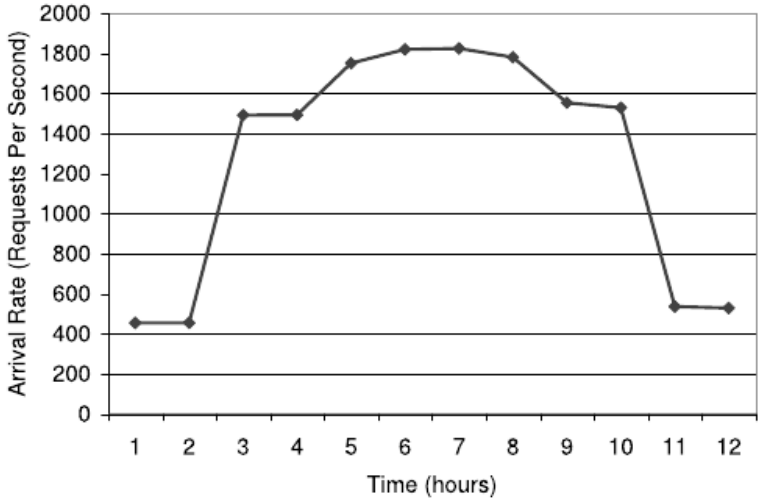


(b) Cello99

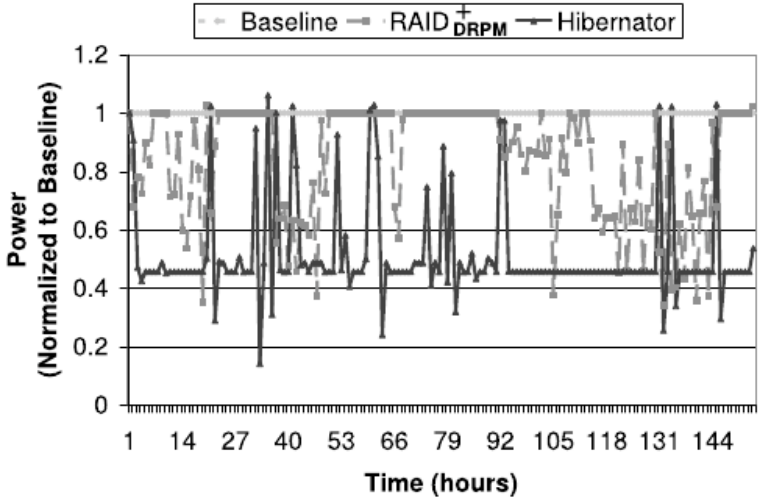
Power Consumption – Hibernator 2

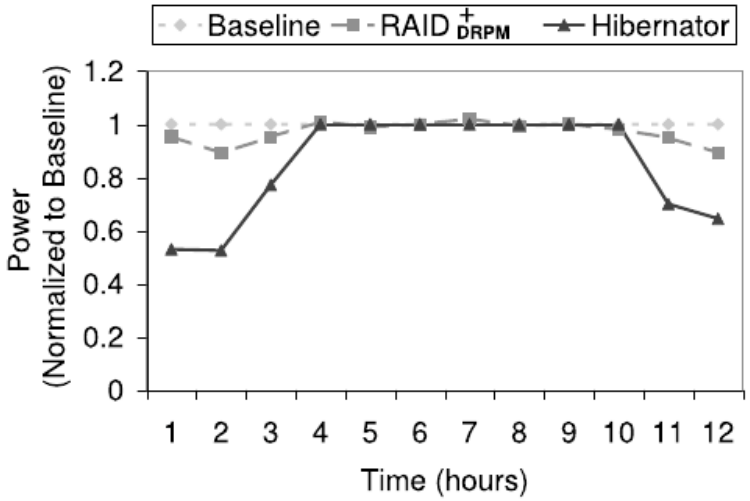


(a) Cello99



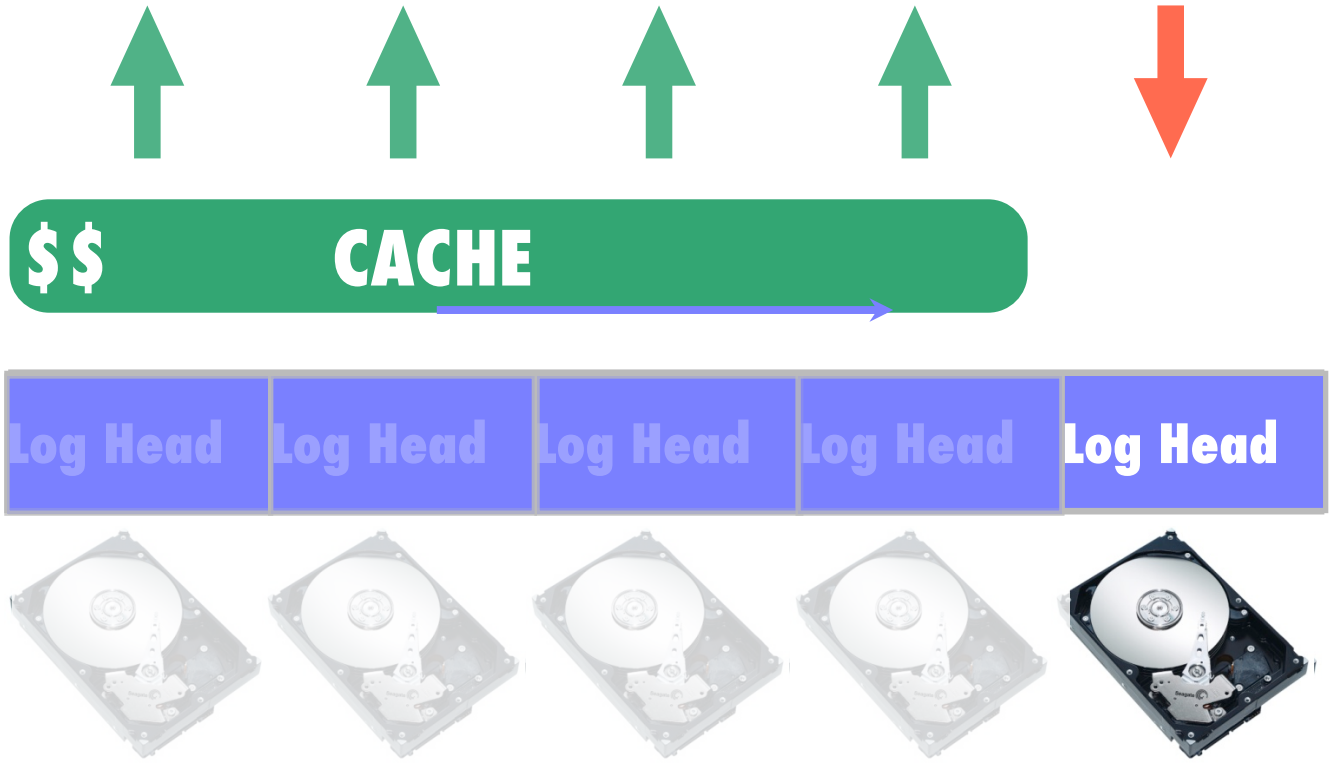
(b) OLTP





Log-structured File System

We don't predict write accesses, we *know*



We predict writes but not reads

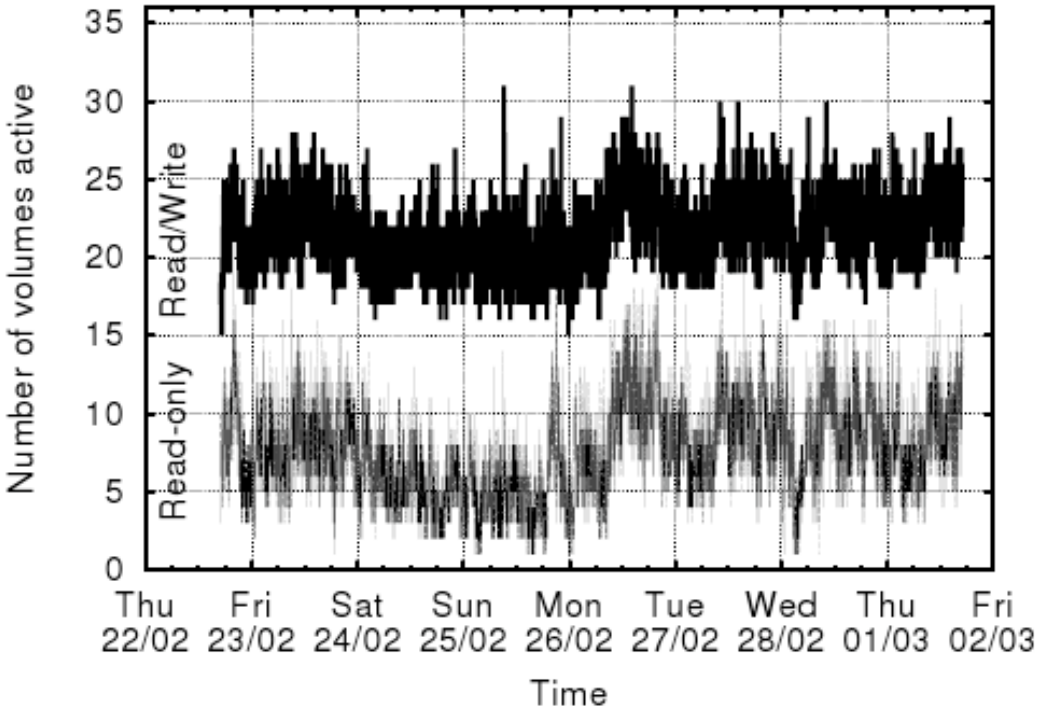
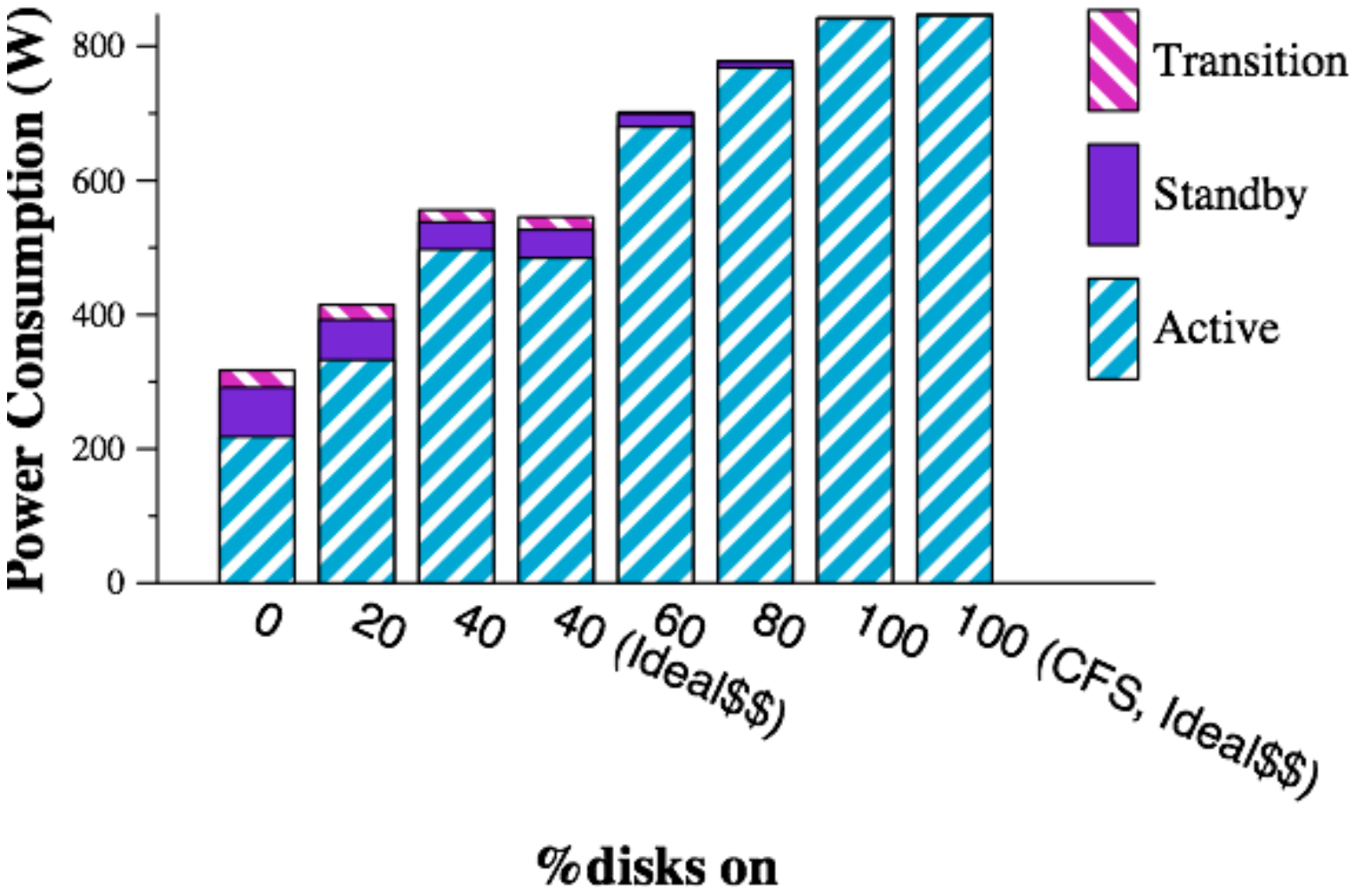


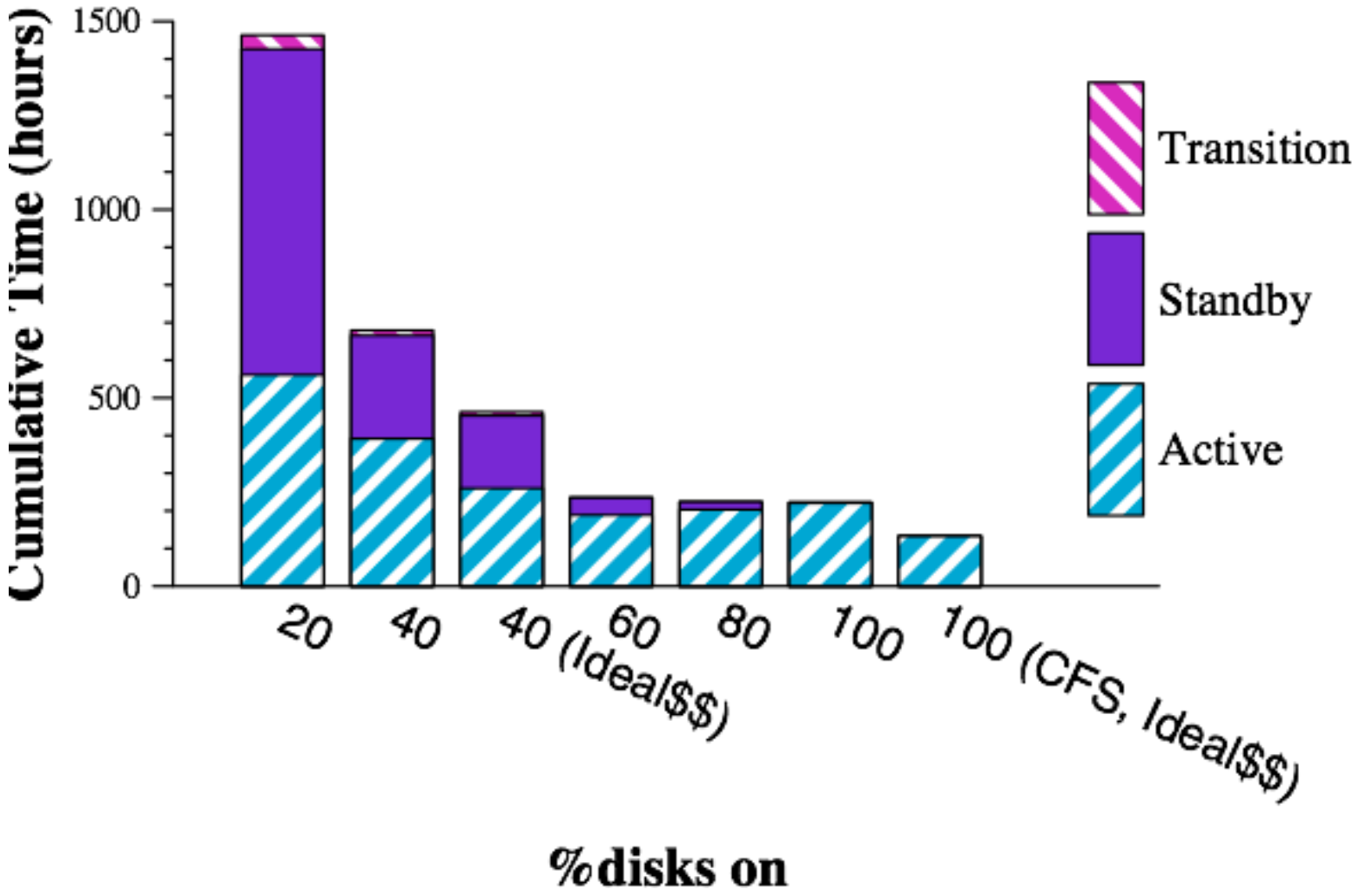
Figure 3: Number of active volumes over time

- Writes:
  - Heavy load
  - BUT one disk
- Reads:
  - ALL disks
  - BUT cache => soft load
- 10% of disks need to be up

Power Consumption



Time of run


Cumulative Time (hours)
1500
1000
500
0
20
40
40 (Ideal$$)
60
80
100
100 (CFS, Ideal$$)
%disks on
Transition
Standby
Active

Write Off-Loading

- Idea:
  - Split the log across all the disks
  - Better write performances
- Design
  - Loggers
    - Temporarily stores blocks on behalf of other disks
    - On each disks
  - Managers
    - Intercept all Read/Write requests
    - Control Off-loading of blocks
    - Consistency & Failure recovery
- Consistency & Failures

Design - Loggers

- Four operations
  - WRITE: data + meta-data (LogicalBlockNr + version)
  - READ: latest stored version
  - INVALIDATE: mark a version as invalid, garbage collected
  - RECLAIM: like read, for any block
- INVALIDATE and RECLAIM: background process
  - Not latency critical
- WRITE and READ : latency critical
  - Reads are rare
  - Optimized for writes: log

Managers

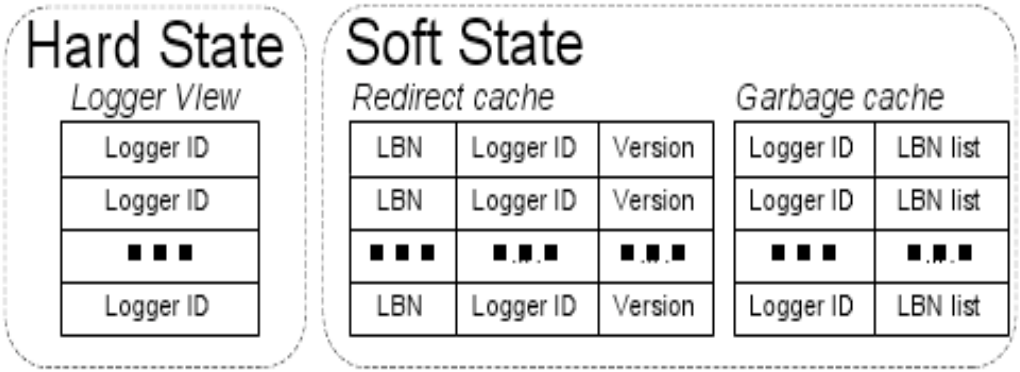


Figure 4: Manager data structures.

- Hard/Soft State
- Reads
  - Check Red Cache for latest version
  - Fallback: home
- Write
  - Choose best logger
  - When write acknowledged: invalidate older versions
  - Writes are reclaimed in idle mode

Consistency & Failures

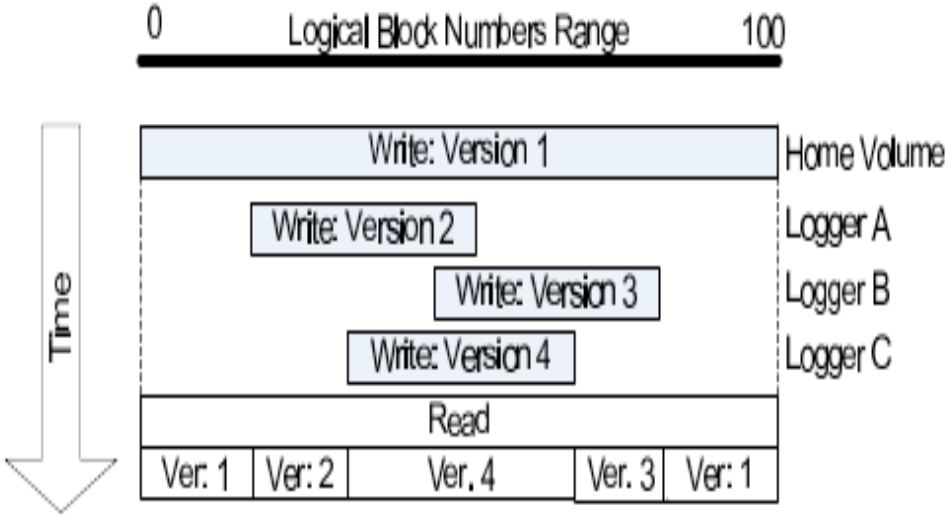
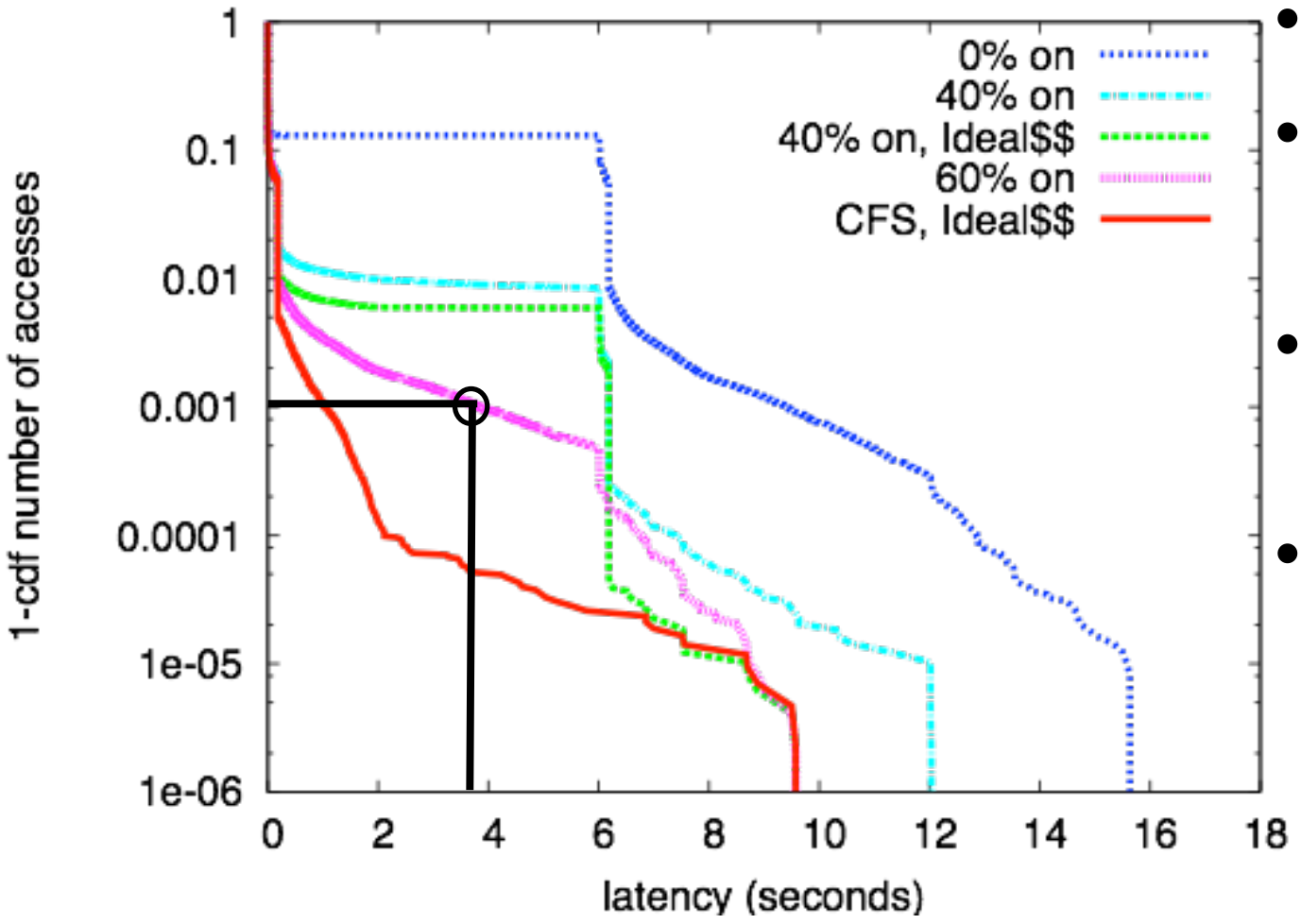


Figure 5: Consistency across loggers example

- Consistency
  - Always knows where the last block is
- Failures
  - Loggers: reconstruct soft state from the log
  - Managers: reconstruct soft state from Logger View and Loggers

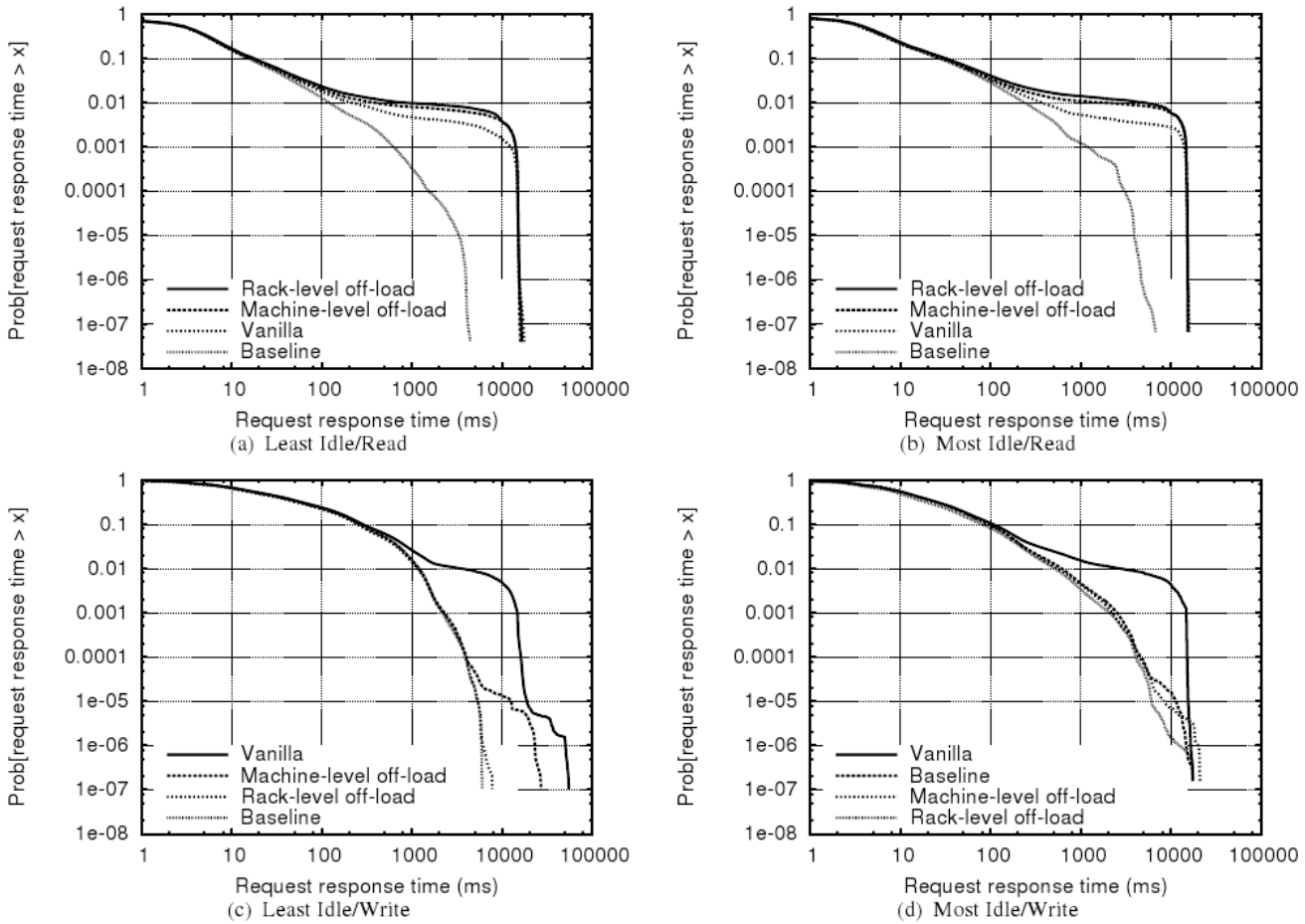
Evaluation

Performance - LFS-based Solution



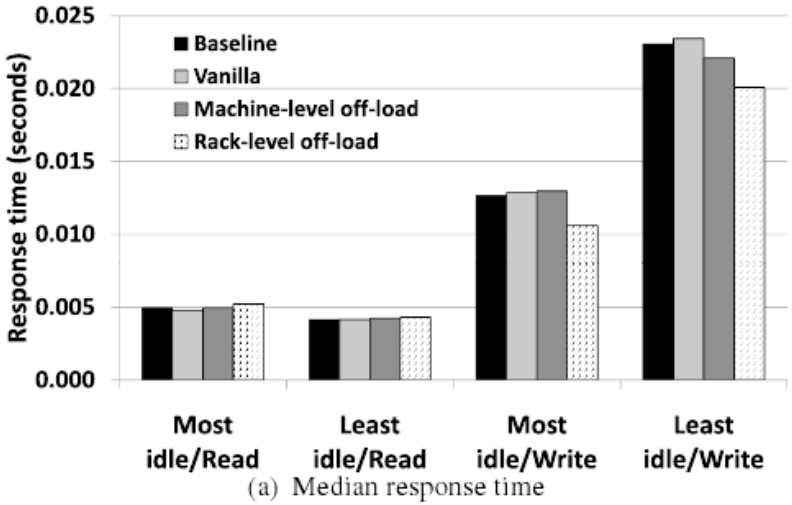
- y: log-scale
- Long-tail distribution
- Cache miss => disks spin up
- 99.9% accesses take <= 3.7s

Performance – Write Off-Loading 1

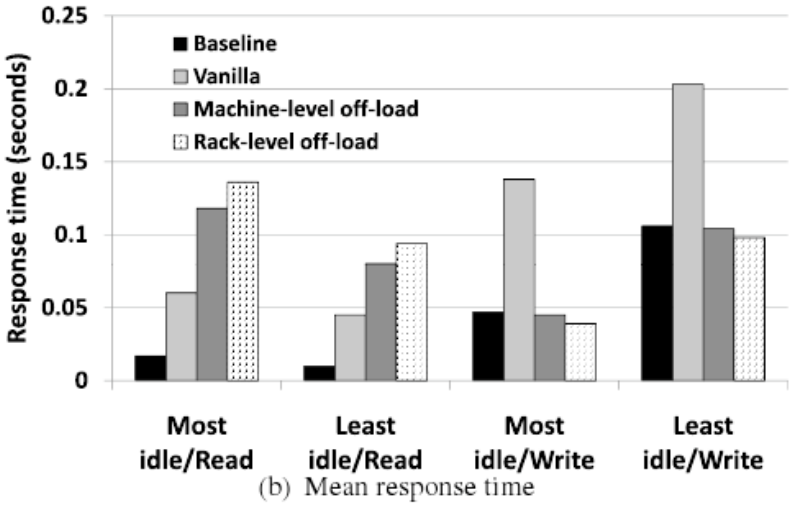


- Same graph
- Left/right = Least/most idle
- Top/Bot = Read/Write
- Read:Cache Miss
- Write:Cache Overflow

Performance – Write Off-Loading 2



(a) Median response time



(b) Mean response time

Median Response Time

Mean Response Time

Performance – Hibernator 1

- Focus set on MEAN response time

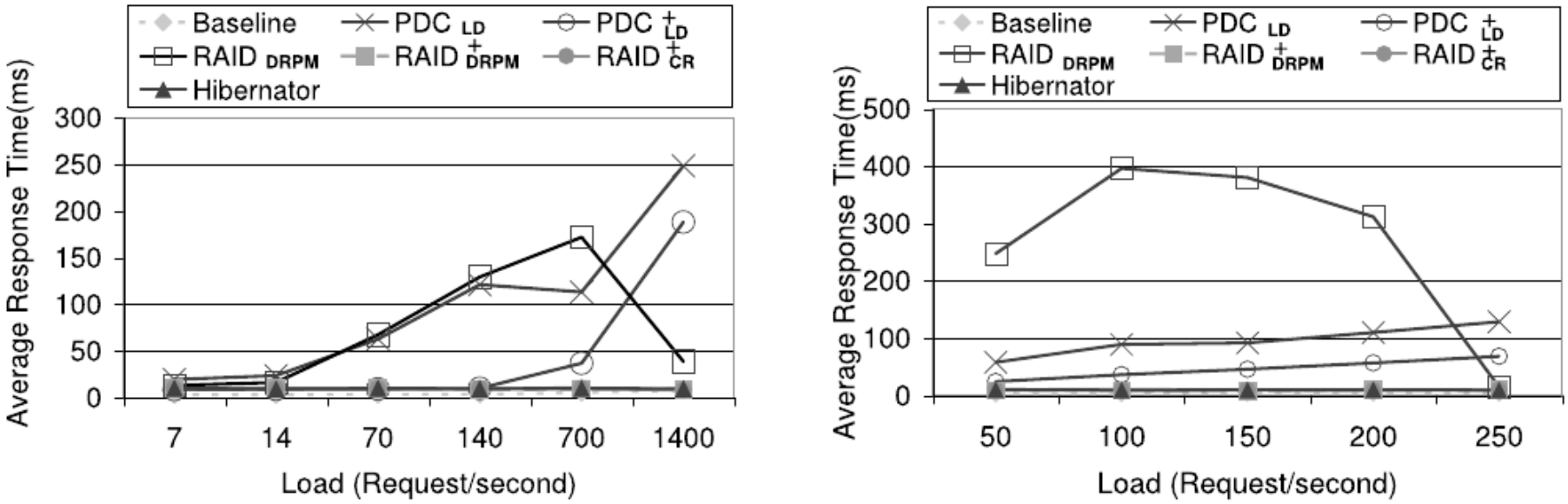
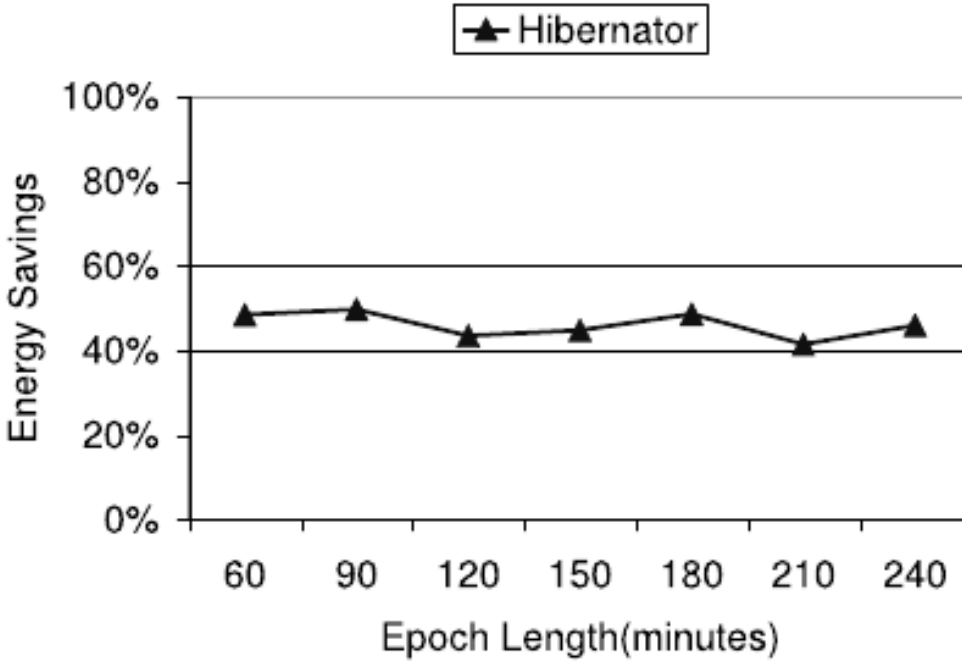
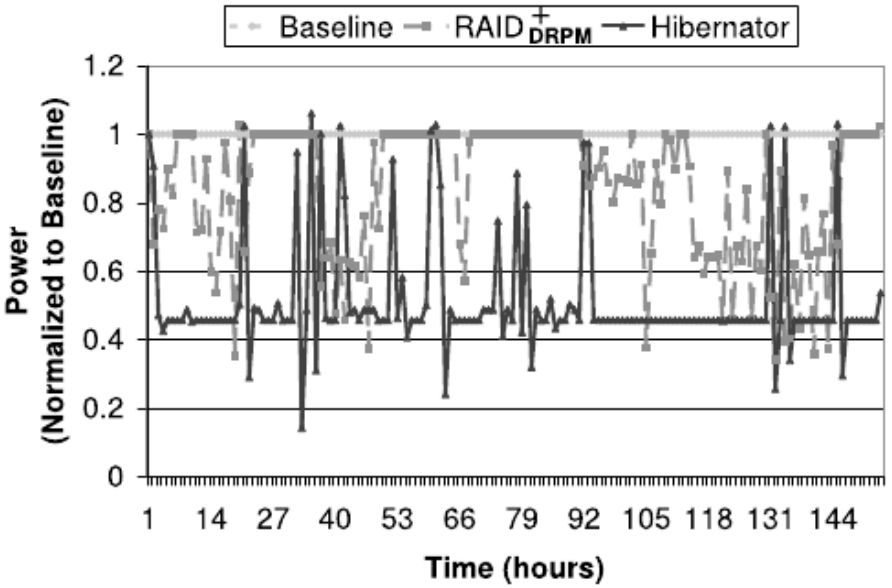


Figure 7: Average response time for different schemes under various stable loads. (Note the x-axis scale in (a) is not linear.)

Performance – Hibernator 2

- Do we still have a long tail distribution?
- Yes: speed transitions need to restart disks
- It can be good: 15s/(240*60s) = 10^-3
- It can be catastrophic





Power Consumption – Write Off-Loading

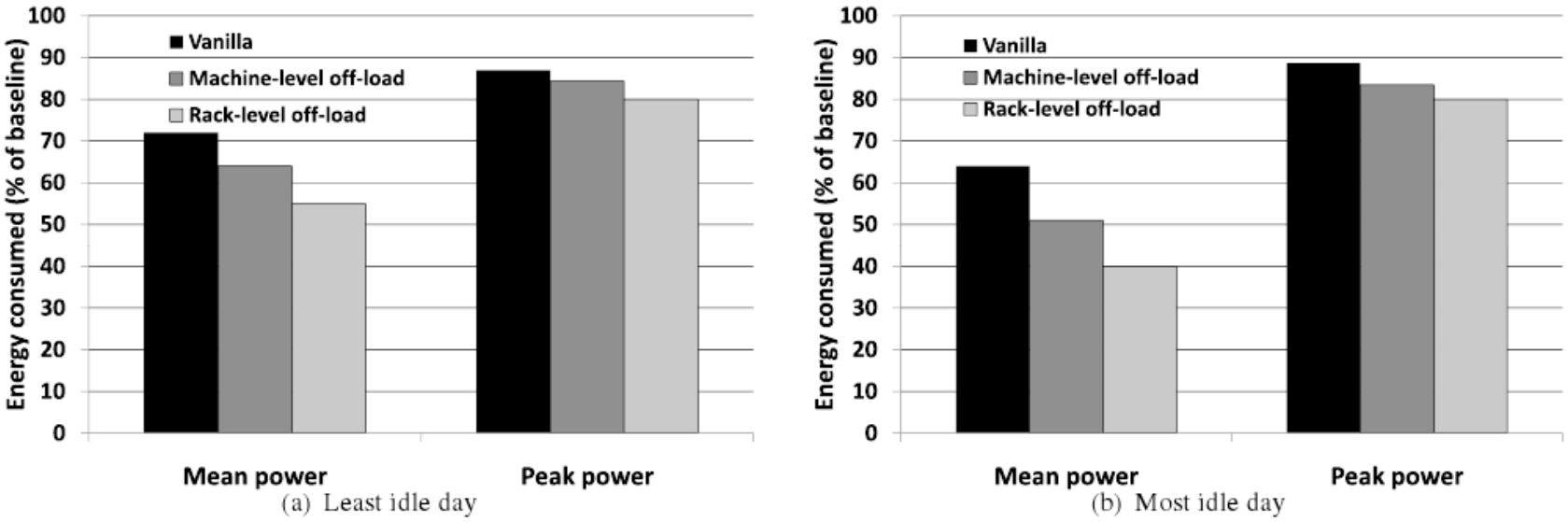


Figure 7: Total power consumption as percentage of baseline

Conclusion

- Substantial power saving can be achieved
- Two solutions
  - Predict the writes
  - Direct the writes
- A trade-off has to be considered:
  - What performance impact can I accept,
  - For what power gain?