

Energy Efficiency through Burstiness

Athanasios E. Papathanasiou

and

Michael L. Scott

University of Rochester

WMCSA 2003

October 9, 2003

Smoothness or Burstiness

- **Smoothness:**

- Commonly used by traditional resource management policies
- Improves overall throughput and latency by minimizing contention
- **BUT:** Can hurt energy efficiency

- **Burstiness:**

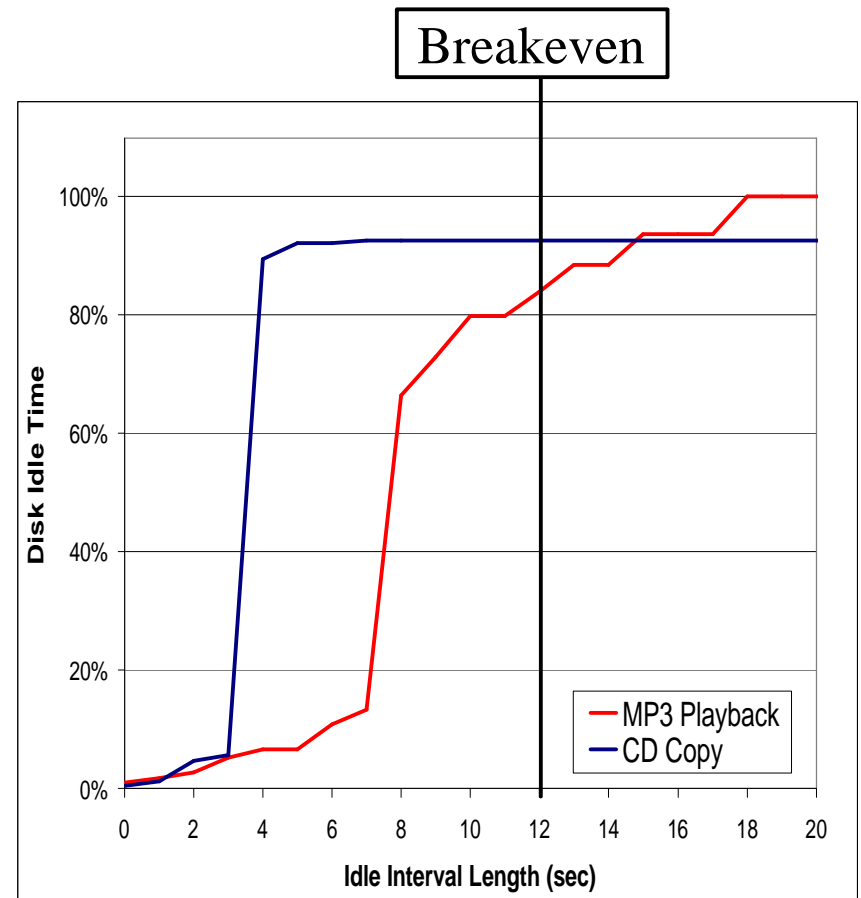
- Can improve energy efficiency in important cases
- Without a significant impact on performance
- Example: Bursty disk access patterns...

Energy-Efficient Devices and Modern File Systems

- **Power-efficient devices:**
 - Support low power modes
 - Save energy by exploiting idle time
 - Require long idle intervals (**tens** of seconds)
 - Remain in low power state for a min. period: **breakeven** point
- **Modern operating systems:**
 - Maximize throughput
 - Minimize latency
- **What about energy?**
 - Idle times: too short to exploit
 - Even under light workload

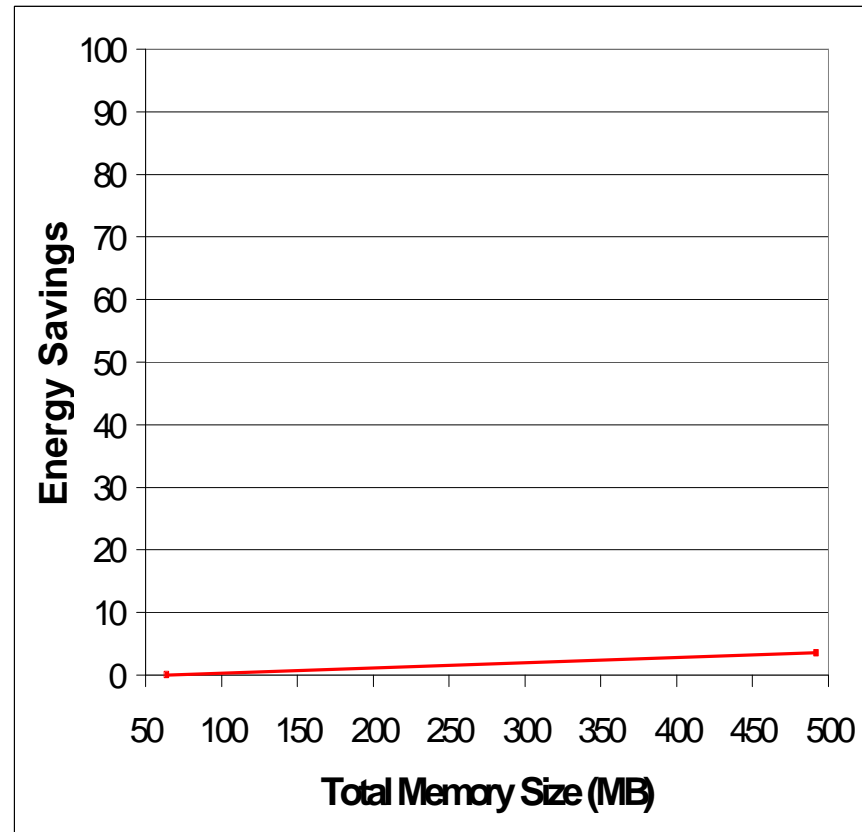
File System Behavior: Examples

- **MP3 playback (300 sec)**
 - Disk Idle Time: **291 seconds**
 - 66%: shorter than 8 seconds
 - 12% only: longer than 12 second breakeven time
- **CD copy (1359 sec)**
 - Disk Idle Time: **1191 seconds**
 - 92%: shorter then 5 seconds
- **Sustainable disk bandwidth: 19MB/s**
 - Much larger than **16KB/s** required for MP3 playback
 - Much Larger than CD Bandwidth



Disk Energy Savings vs. Memory Size

- Intuitively *increased* system memory should lead to *reduced* disk energy consumption
- **BUT:** this is NOT the case...
- **Example: MPEG playback**
 - **8-fold** memory increase
 - **3.4%** disk energy savings

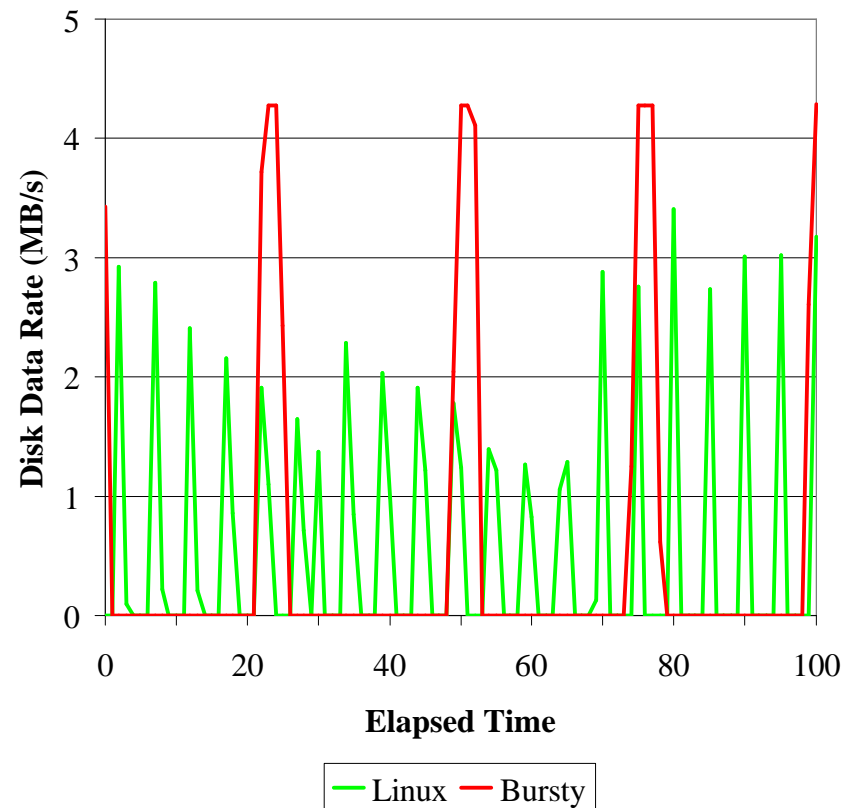


New Design Goal:

Increase Burstiness

- **Maximize energy efficiency**
 - Increase idle interval length to allow a power state transition
 - Operate at max disk bandwidth when disk is active
 - Decrease number of transitions

Disk Usage Pattern



Design Guidelines for Energy Efficiency

- **Maximize idle phases**
 - Aggressive, Speculative Prefetching
 - ✓ With hints to improve accuracy
 - Bursty Periodic Update
 - ✓ Increased variance in time-to-stable-storage
 - ✓ Respect reliability constraints
- **Coordinate I/O across applications**
 - Arrange for all applications to run out of data at the same time
- **Preactivate the disk in anticipation of next use**
 - Maintain interactive responsiveness

Epoch-based Extensions:

Two Phases per Epoch

- **Request generation phase**
 - Predict and prefetch future data and metadata accesses
 - Estimate memory size for prefetching and buffering
- **Idle phase**
 - Estimate time to next request
 - Power-down disk if predicted idle time is long enough
 - Schedule disk preactivation.
- **New epoch triggered by:**
 - Initiation of a new prefetching cycle
 - Demand miss or expiration of one or more dirty buffers
 - Low system memory

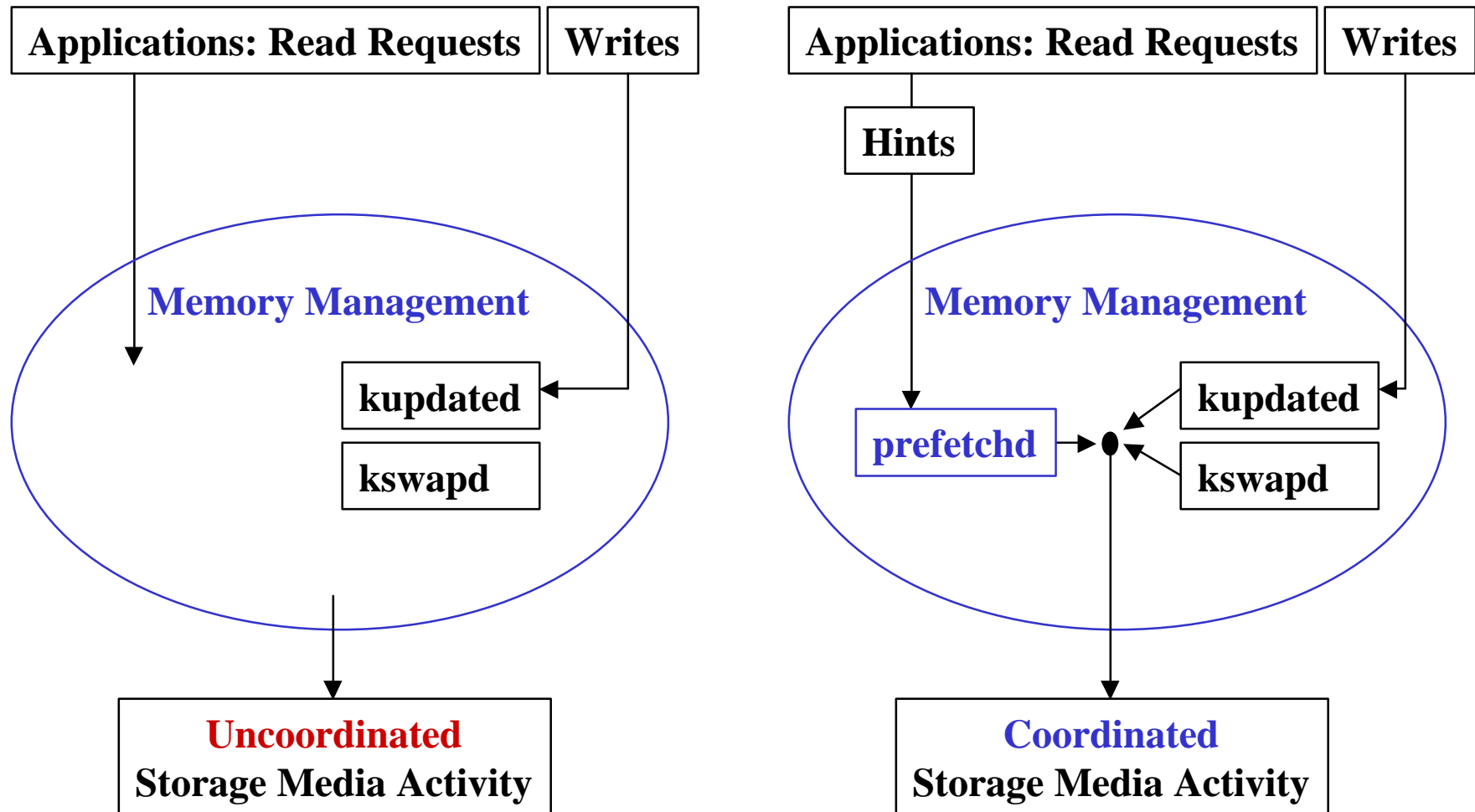
Deciding What to Prefetch

- **Sequential access is relatively easy**
 - MP3, MPEG, encoding, copying, etc.
 - Notice pattern and rate; predict accordingly
- **Applications with random or multiple-file accesses:**
 - Focus of our current research
 - System call for application-provided hints
 - On-line monitoring system supports existing applications
 - ✓ Analyzes past execution instances
 - ✓ Provides hints on behalf of applications
 - Prefetching can be **very speculative**
 - ✓ Resembles **prefetching for disconnected operation**

Estimating Memory for Prefetching The Prefetch Cache

- **LRU VM extended with Prefetch Cache**
- **Prefetch Cache size**
 - Large enough to contain all predicted data accesses
 - Without causing eviction of useful data
- **Type of first miss determines Prefetch Cache size for next Epoch**
 - *Compulsory Miss*: No change
 - *Prefetch Miss*: Increase by constant
 - *Eviction Miss*: Decrease by number of useful pages evicted for prefetching
- **Keeping track of eviction history: *Eviction Cache***
 - Stores metadata of recently evicted pages with increasing serial number
 - *Epoch's starting eviction number*: Serial number of first evicted page.

Coordinating across Applications: The Prefetch Thread



Update Policy

- **Dirty buffers flushed once per minute**
- **Modified *open* system call**
 - Provides direction to postpone write-behind until *close*
 - Useful for application without strict reliability constraints
 - ✓ Examples: encoding operations, copying, compilations!

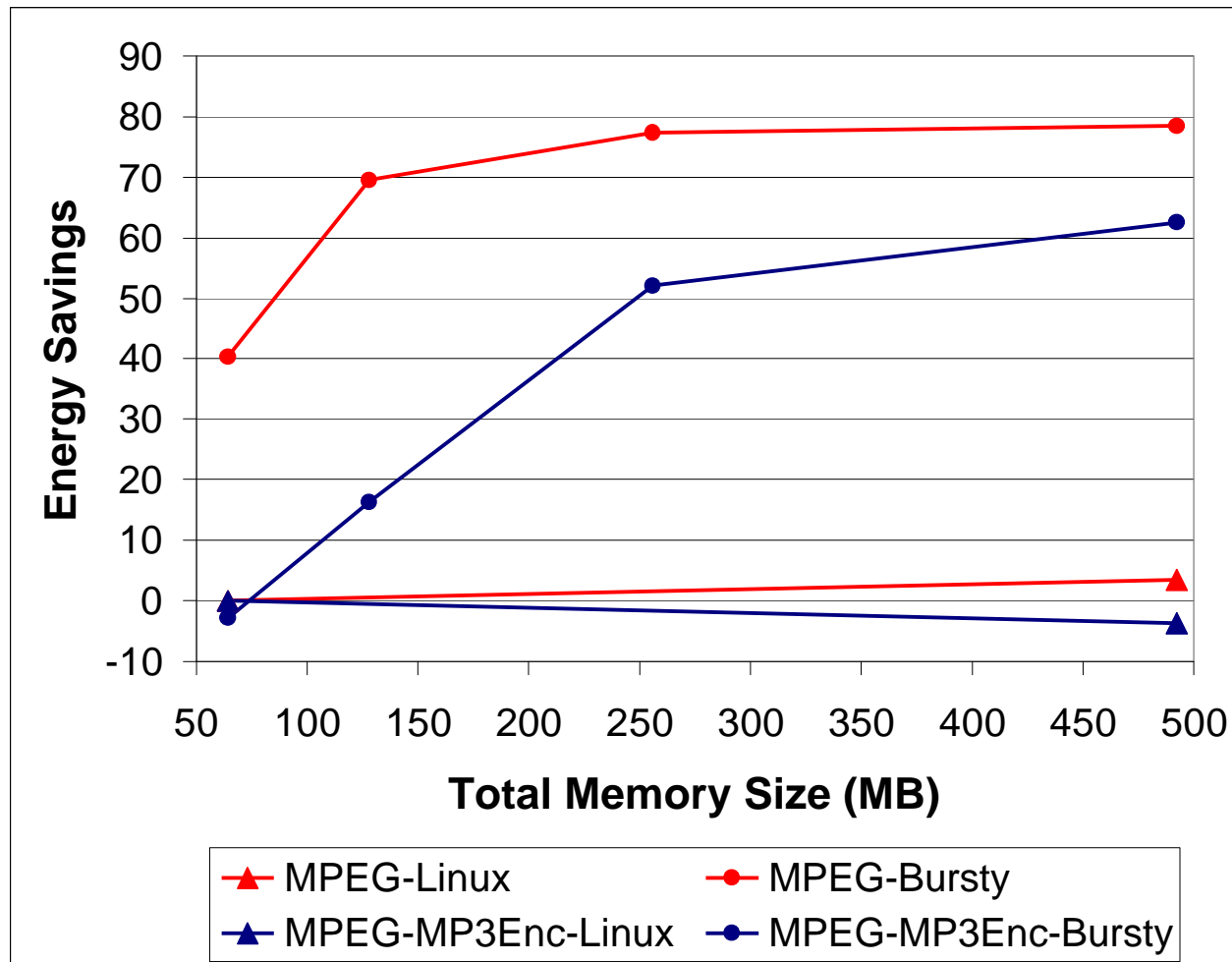
Maintaining Responsiveness

- **Responsiveness may be decreased because of**
 - Increased disk congestion due to burstiness
 - Latency penalty for disk power-up
- **Solution: Preactivate disk**
 - Monitor application progress and file system cache state
 - ✓ Data consumption and production rates
 - Initiate prefetch cycle before applications run out of data
- **For interactive responsiveness:**
 - Prioritized disk queues:
 - ✓ Required to service quickly unpredicted demand misses during periods of high disk congestion

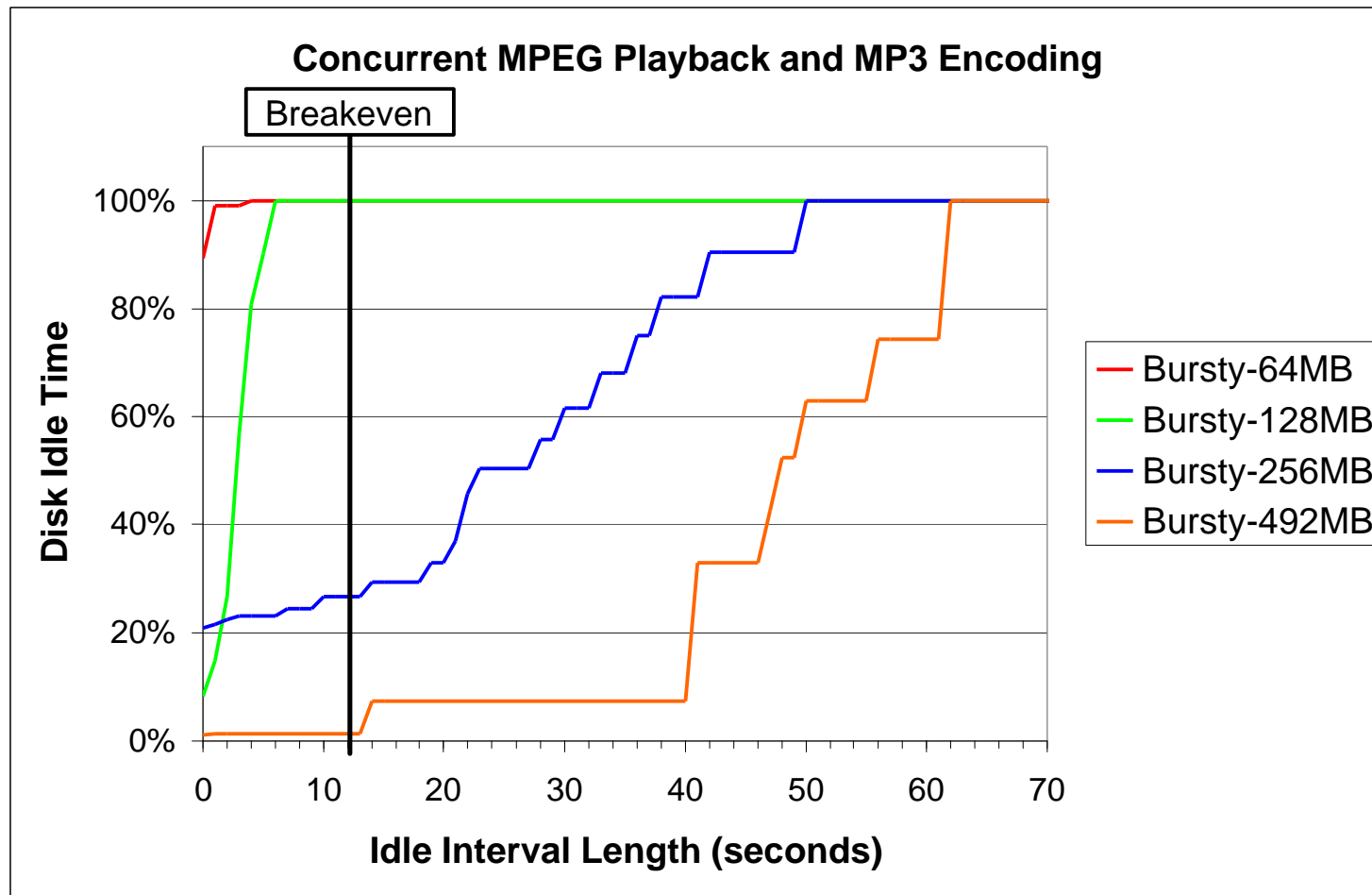
Platform and Workloads

- **Dell Inspiron 4100 Laptop**
 - 512MB of memory
 - Hitachi DK23DA hard disk
- **Power measurements collected through instrumentation of the hard disk's supply lines**
- **Workload scenarios**
 - MPEG Playback (two 70MB files)
 - Concurrent MPEG Playback and MP3 encoding
 - ✓ MPEG Player Input: two 70MB files
 - ✓ MP3 Encoder Input: 10 WAV files (626MB)
 - ✓ MP3 Encoder Output: 10 MP3 files (42.9MB)
- **Power management policy:**
 - **Linux:** 10 second fixed threshold (degenerates to No-Spin-Down)
 - **Bursty:** Predictive algorithm that monitors application progress and file system cache state

Energy Savings vs. Memory Size

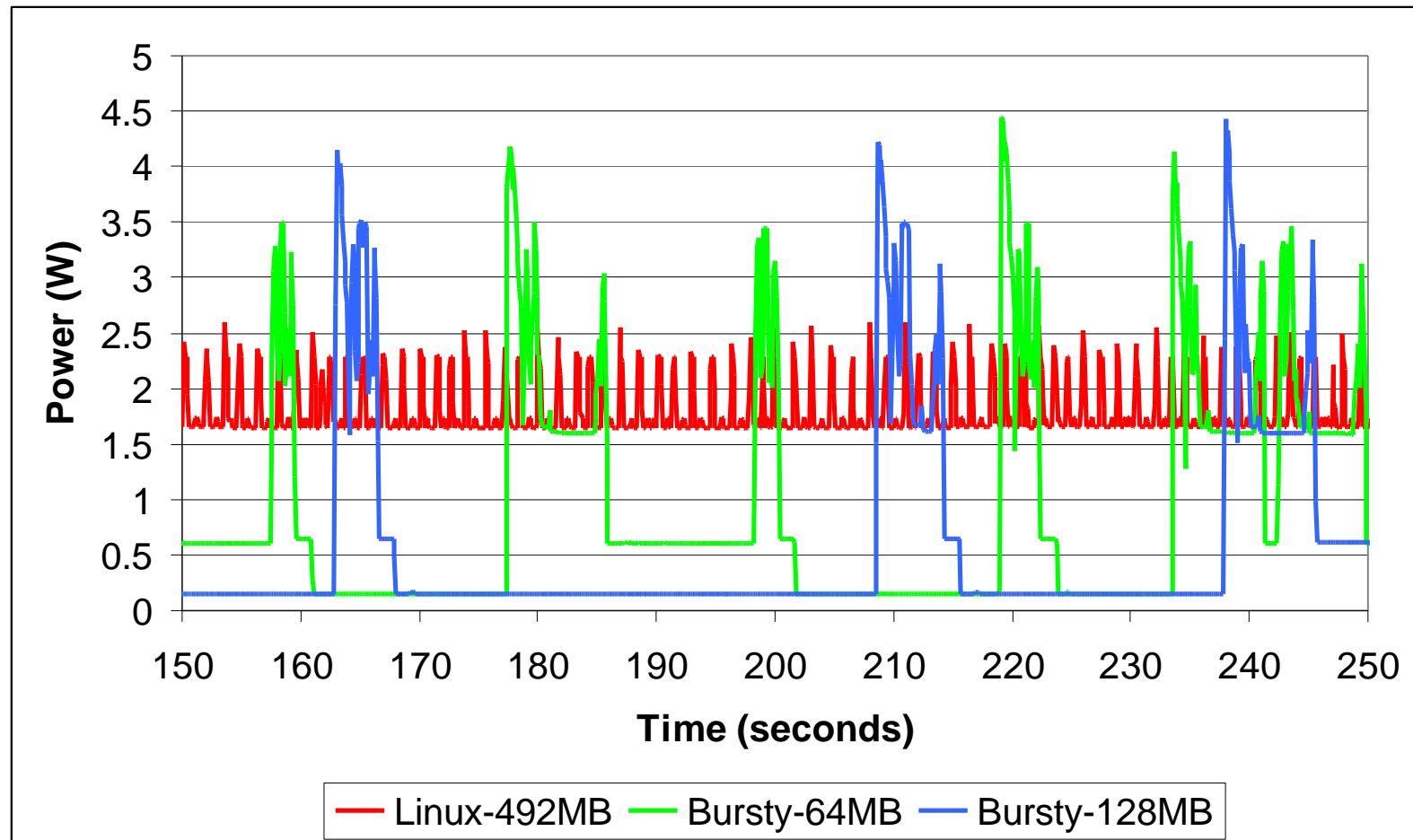


Distribution of Idle Time Intervals

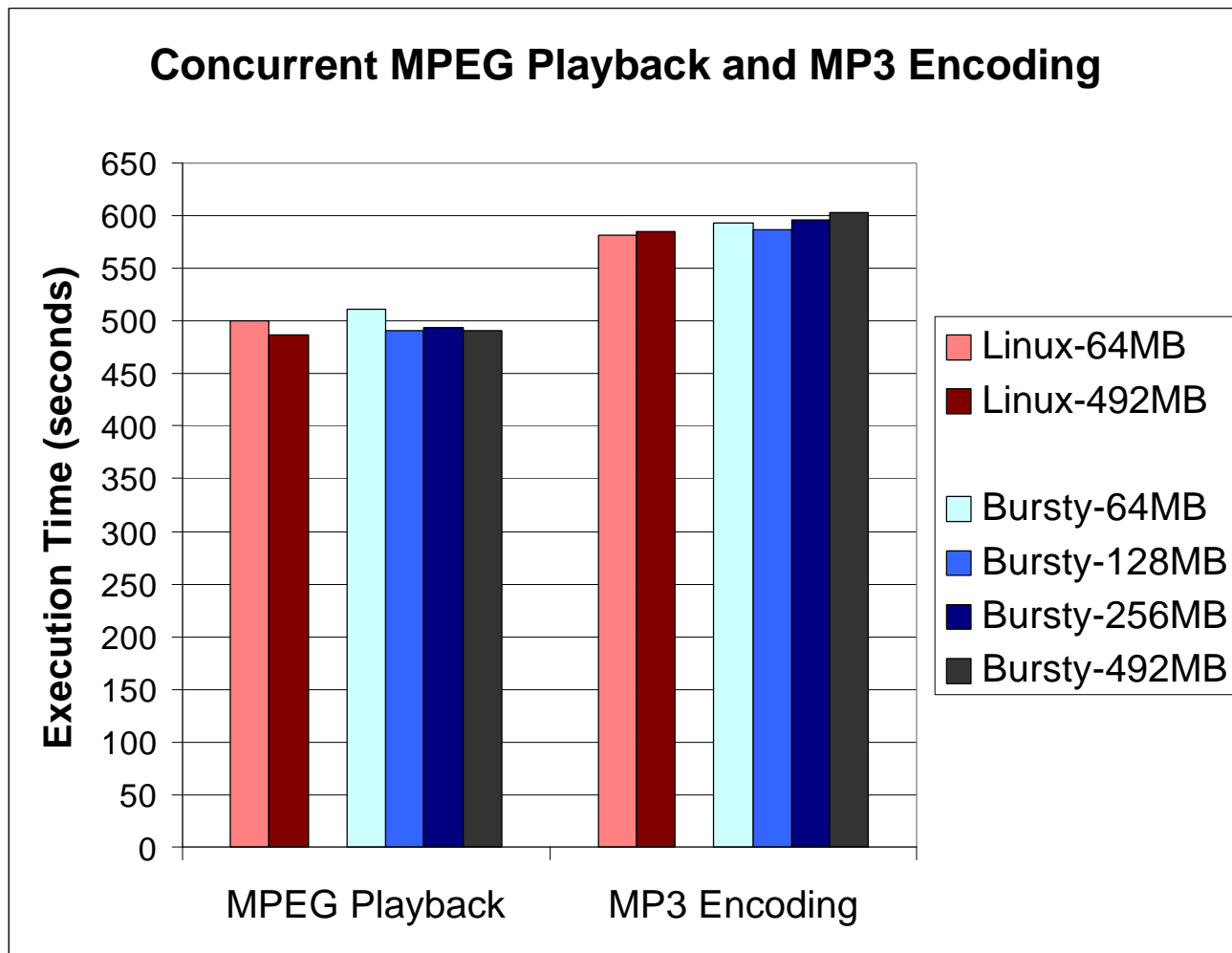


Disk Power Consumption Snapshot

MPEG Playback



Performance



Conclusion

- **Current Prototype**
 - Works well with predictable applications.
 - Energy savings scale with memory size.
 - Up to **78.5%** disk energy savings.
 - **Less than 5%** performance penalty across all workloads and memory sizes
- **For less predictable applications:**
 - Support for guiding the prefetching system is essential
 - ✓ New applications can provide hints explicitly
 - ✓ For older applications hints can be provided through an *on-line monitoring system*
 - Requires efficient access analysis & prediction techniques
 - Very speculative prefetching algorithms can be used to minimize “*false negatives*”
 - ✓ Even a high “false positive” to “true positive” ratio has a small negative effect on energy consumption.

Burstiness and...

- **Network interfaces**
 - Varying levels of broadcast power in addition to standby modes
 - Energy consumption depends on channel's quality
 - ✓ Communication bursts during periods of high quality
 - Accommodating externally initiated traffic
 - ✓ Impact on transport and physical layer protocols
- **Memory chips (RDRAM)**
 - Increase burstiness of accesses on one chip...
 - ✓ And idle interval lengths on the rest
- **Processor domain (future architectures)**
 - Save dynamic power by slowing down the integer unit in a floating point application **or**
 - Schedule instructions for burstiness and save both dynamic and static energy by gating off voltage to the integer unit

Energy Efficiency through Burstiness

For more info visit:

<http://www.cs.rochester.edu/~papathan/research/BurstyFS/>

WMCSA 2003

October 9, 2003