

*Energy-Efficient
Prefetching and Caching*

Athanasios E. Papathanasiou

and

Michael L. Scott

University of Rochester

USENIX 2004

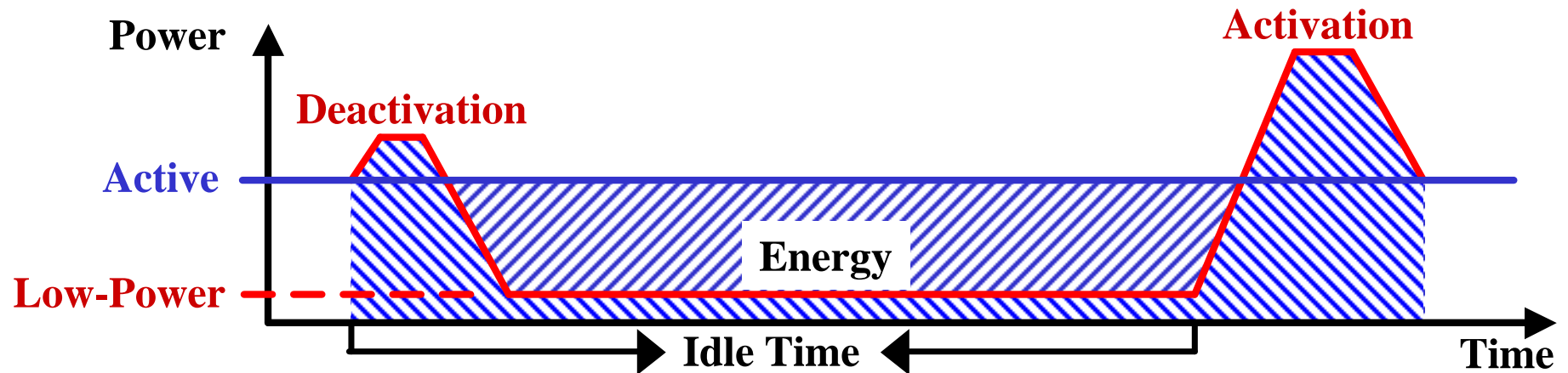
June 30, 2004

Prefetching and Energy Efficiency

- **Traditional prefetching and caching:**
 - Standard practice in modern operating systems
 - Improve performance (throughput and latency)
 - Spread requests smoothly over time
 - ✓ Avoid disk congestion
- **Shortcomings of traditional prefetching:**
 - Ignores and frustrates goal of energy efficiency
 - ✓ Important for mobile systems
 - Modern devices exploit **long idle intervals** to save energy
 - ✓ Disks, network interfaces, memory chips
 - **Smoothness** leads to **short idle intervals**

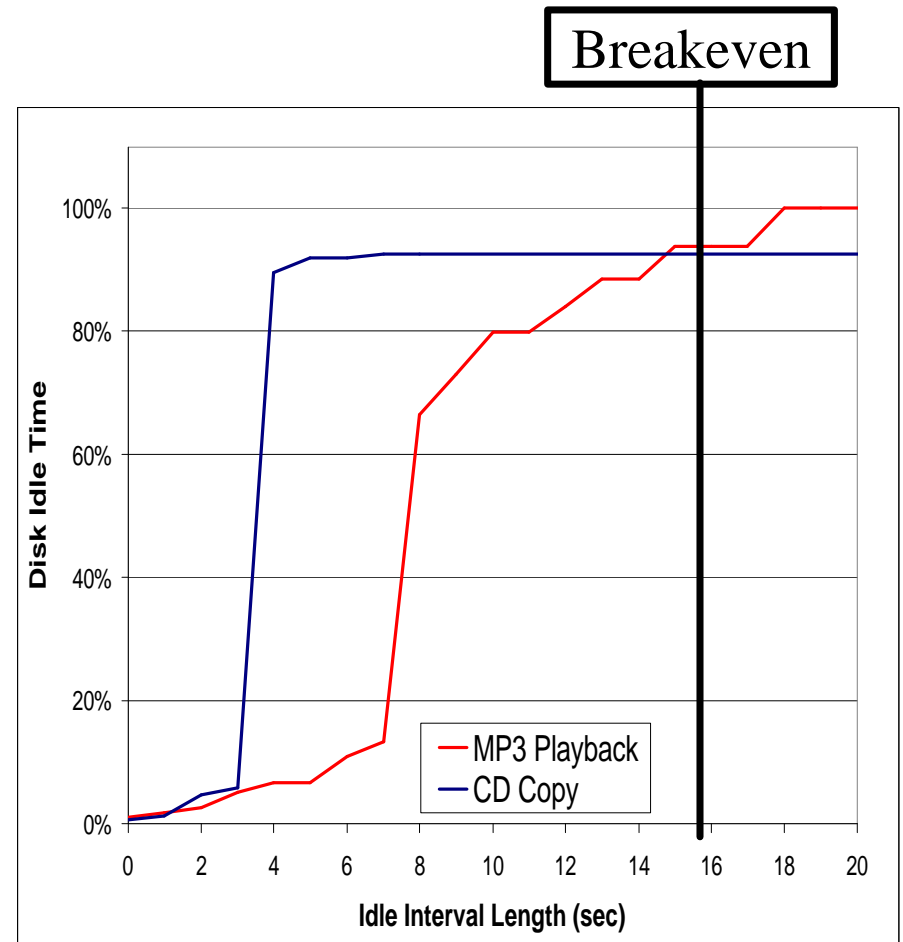
Hard Disk Power States

- **Modern disks: up to five low-power modes**
- **Lower modes consume less power**
 - Lowest modes: **an order of magnitude** lower power than active
- **Significant time and energy to return to active**
- **Breakeven time**
 - Lowest modes: at the order of **several seconds**



File System Behavior: Examples

- **MP3 playback (300 sec)**
 - Disk Idle Time: **291 seconds**
 - 66%: shorter than 8 seconds
 - 10% only: longer than 16 second breakeven time
- **CD copy (1359 sec)**
 - Disk Idle Time: **1191 seconds**
 - 92%: shorter than 5 seconds



Traditional vs. Energy-reducing Prefetching

- **Objective of traditional prefetching**
 - Hide disk delays from applications to improve performance
- **Objective of energy-reducing prefetching**
 - Maximize length of disk idle intervals to save energy

Energy-reducing Prefetching Challenges

- **Prefetching has to be very aggressive**
 - Fetch as many future references as possible
- **Prefetching has to be very speculative**
 - An avoided power-up operation justifies lots of unnecessary requests
- **I/O activity across applications has to be coordinated**
 - Uncoordinated requests break idle time into short intervals
- **Prefetching has to be latency-sensitive**
 - Take into account delays of disk activation and congestion

Rules for Optimal Prefetching for Performance (Pei Cao'95)

- **Optimal prefetching**

- Fetch next page

- **Optimal replacement**

- Discard page whose reference is farthest in the

Time	Application	Disk	Cache		
1		F(A)			
2	Access(A)	P(B)	A		
3	Access(A)	P(C)	A	B	
4-11	Access(A)	idle	A	B	C

Idle time appears in intervals of **9** time units on average

- **DO NO HARM**

- **First opportunity**

- Prefetch when a fetch completes
- Prefetch after reference

13-21	Access(B)	idle	D	B	C
22	Access(C)	P(E)	D	B	C
23-31	Access(C)	idle	D	E	C
32	Access(D)	P(F)	D	E	C
33-41	Access(D)	idle	D	E	F

Energy Efficient Prefetching

- **Substitute rule 4 for:**
 - Maximize disk utilization
 - ✓ Always initiate a prefetch after a fetch if possible
 - Respect idle time
 - ✓ Never interrupt an idle

Time	Application	Disk	Cache		
1		F(A)			
2	Access(A)	P(B)	A		
3	Access(A)	P(C)	A	B	
4-11	Access(A)		A	B	C

Idle time appears in intervals of **27** time units on average

- **Dynamically switch between “first opportunity” and “just-in-time” based on disk state**

27-30	Access(C)		D	B	C
31	Access(C)	P(D)	D	B	C
32	Access(D)	P(E)	D	E	C
33	Access(D)	P(F)	D	E	C
34-41	Access(D)	idle	D	E	F

Design Challenges

- **When to prefetch?**
- **What to replace?**
- **What to prefetch?**

Deciding When to Prefetch: Epoch-based Extensions

- **Active phase**
 - Predict future data and metadata accesses
 - Estimate memory size for prefetching
- **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
 - Low system memory

Deciding What to Replace

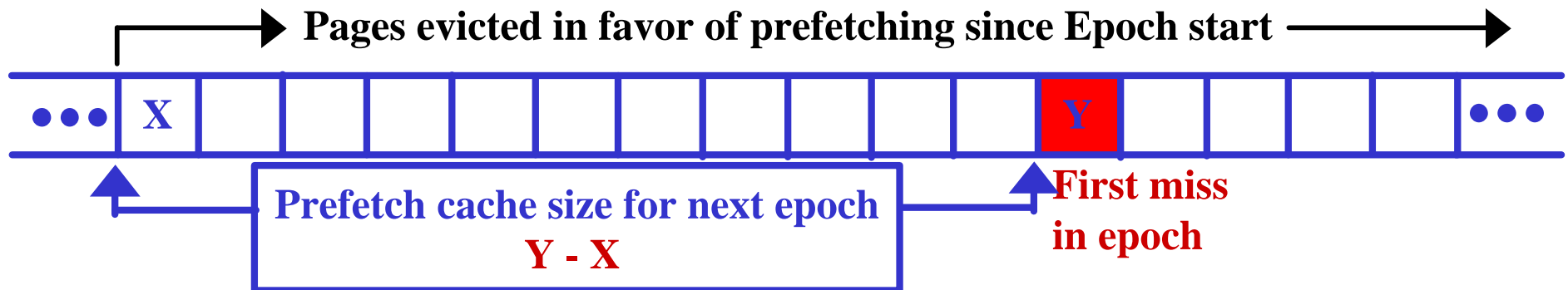
- **Goal: maximize time to first miss**
- ***Prefetch Cache***
 - Large enough to contain all predicted data accesses
 - Without causing eviction of useful data
- **First miss determines prefetching depth for next Epoch**
 - *Compulsory, Prefetch and Evictions Misses*
- **Keeping track of eviction history: *Eviction Cache***
 - Stores metadata of evicted pages
 - Identifies pages erroneously evicted in favor of prefetching

Prefetch Cache Size Estimation

- *Compulsory Miss*: No change
- *Prefetch Miss*: Increase by constant
- *Eviction Miss*:

X, Y: Eviction Numbers

Eviction Cache



Deciding What to Prefetch: Data Prediction

- **Common application characteristics**

- Long running
- Significant disk idle time
- BUT short idle intervals

- **Three classes of applications:**

- Sequential accesses to large files
 - ✓ Decoding, encoding, data transferring
- Accesses to multiple files
 - ✓ Compilations, version control, indexing
- Random accesses to large files
 - ✓ SPHINX-II speech recognition

**Simple
Automatic detection**

**Speculative
Aggressive
Hints for accuracy**

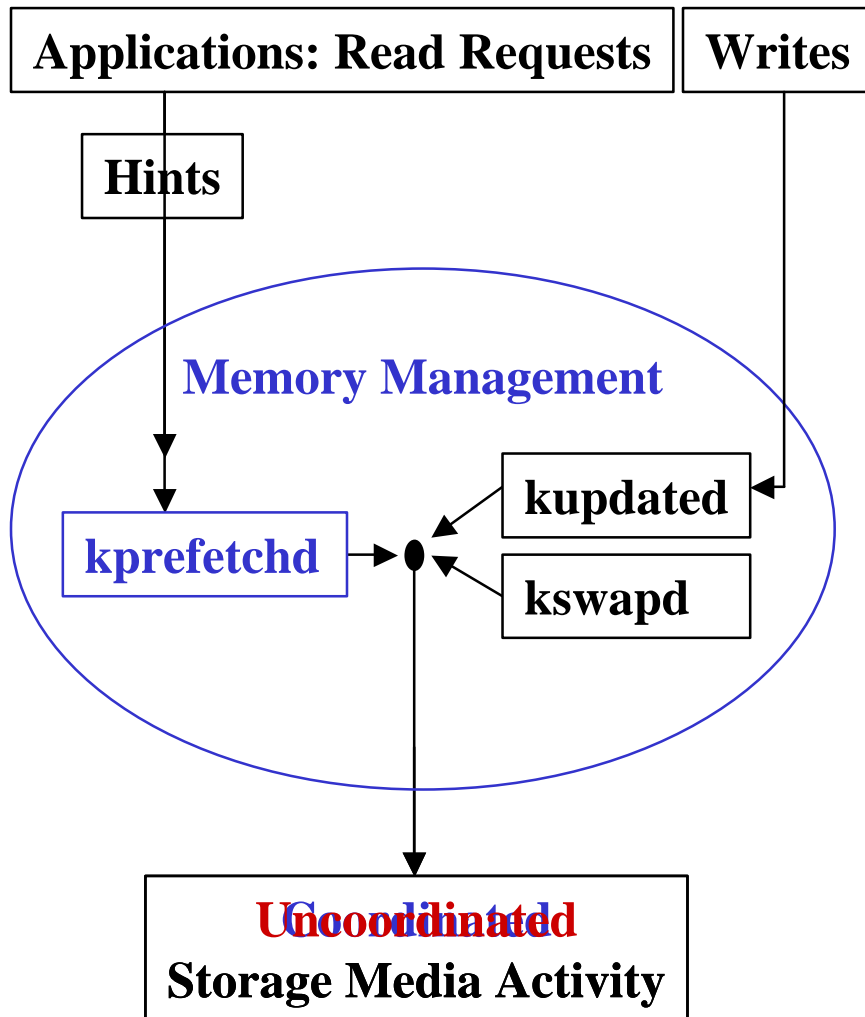
Automatic Hint Generation: Monitor Daemon

- **Trace process and I/O syscalls**
- **Analyze trace offline**
 - Create DB, indexed by
 - ✓ Application pathname
 - ✓ Current working directory
 - ✓ Application arguments
- **Use DB to generate runtime hints**
 - On behalf of application at execution time

Trace Analysis Details

- **Associated applications**
 - Accesses attributed to **process group leader**
- **Multiple file accesses**
 - One hint for each file in DB
- **Random accesses**
 - Identify “hot blocks”

Coordinating across Applications: The Prefetch Thread



Additional goal of prefetch thread

- Monitors rate of progress for each application
- Splits prefetch cache based on application rate

Prefetch Cache



**Fast
Application**

**Slow
Application**

- Applications run out of in-memory data within **the same small window of time**

Handling Write Activity

- **Dirty buffers flushed once per minute**
- **Optional: modified `open` system call**
 - Postpone write-behind until **`close`** or **`exit`**
 - Use with applications without strict reliability constraints
 - ✓ Examples: encoding operations, copying, compilations
- **Write side effects: **read** accesses for file system metadata**
 - Directory structure
 - Free disk block bitmap
 - Inode information

Experimental Evaluation

Experimental Platform

- **Dell Inspiron Laptop**
 - 512MB of memory
 - Hitachi DK23DA disk
- **Power measurements:**
 - Instrumentation of the hard disk's supply lines

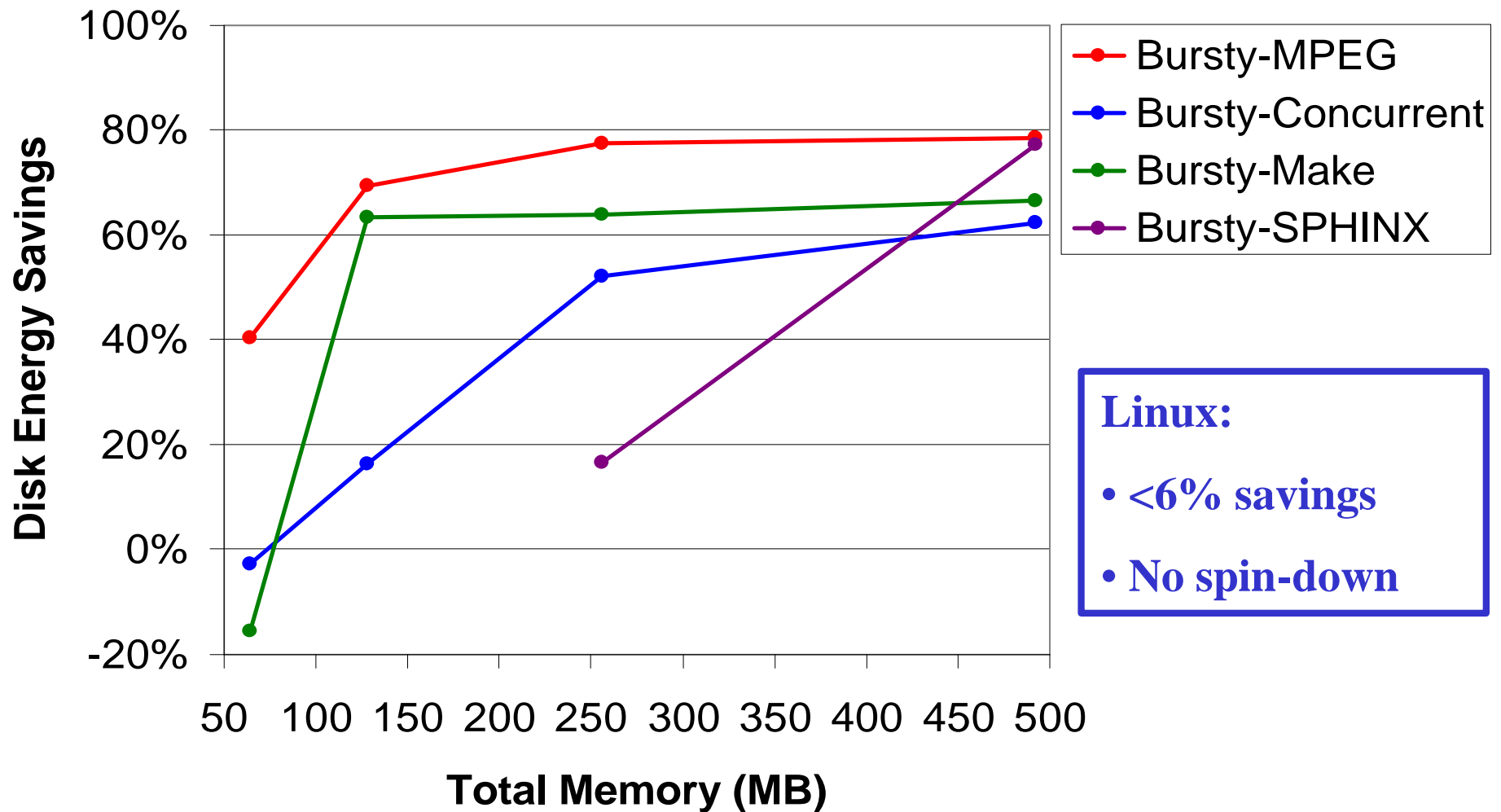


Workload Scenarios

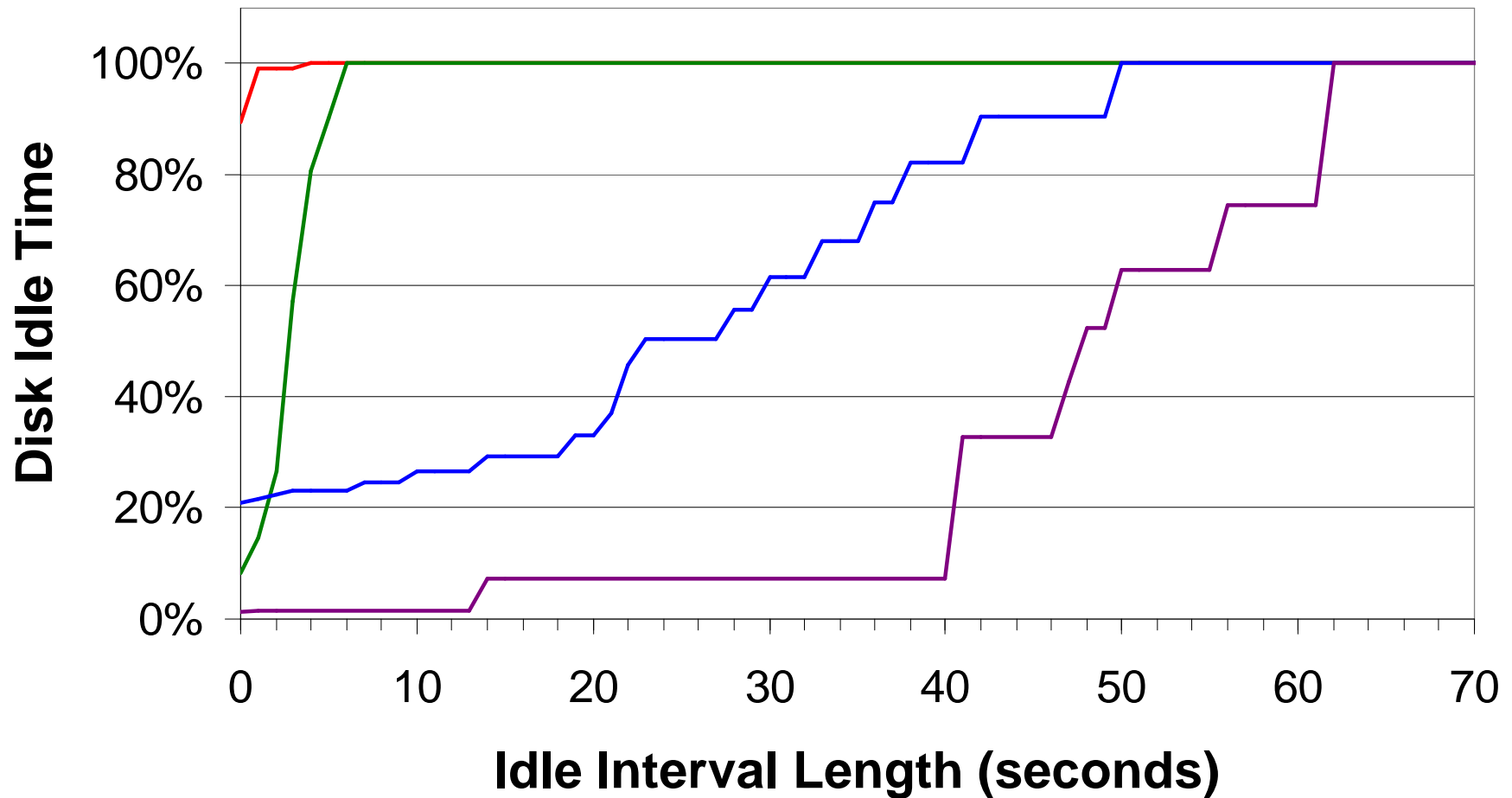
Workload Characteristics

- **MPEG playback** → Sequential accesses
- **Concurrent**
 - MPEG playback
 - MP3 encoding} Multitasking workload
Request coordination
- **Linux Kernel Compilation** → Multiple files
Associated tasks
- **SPHINX-II Speech Recognition**
 - ✓ Random accesses to 128MB file} Random Access

Energy Savings vs. Memory Size



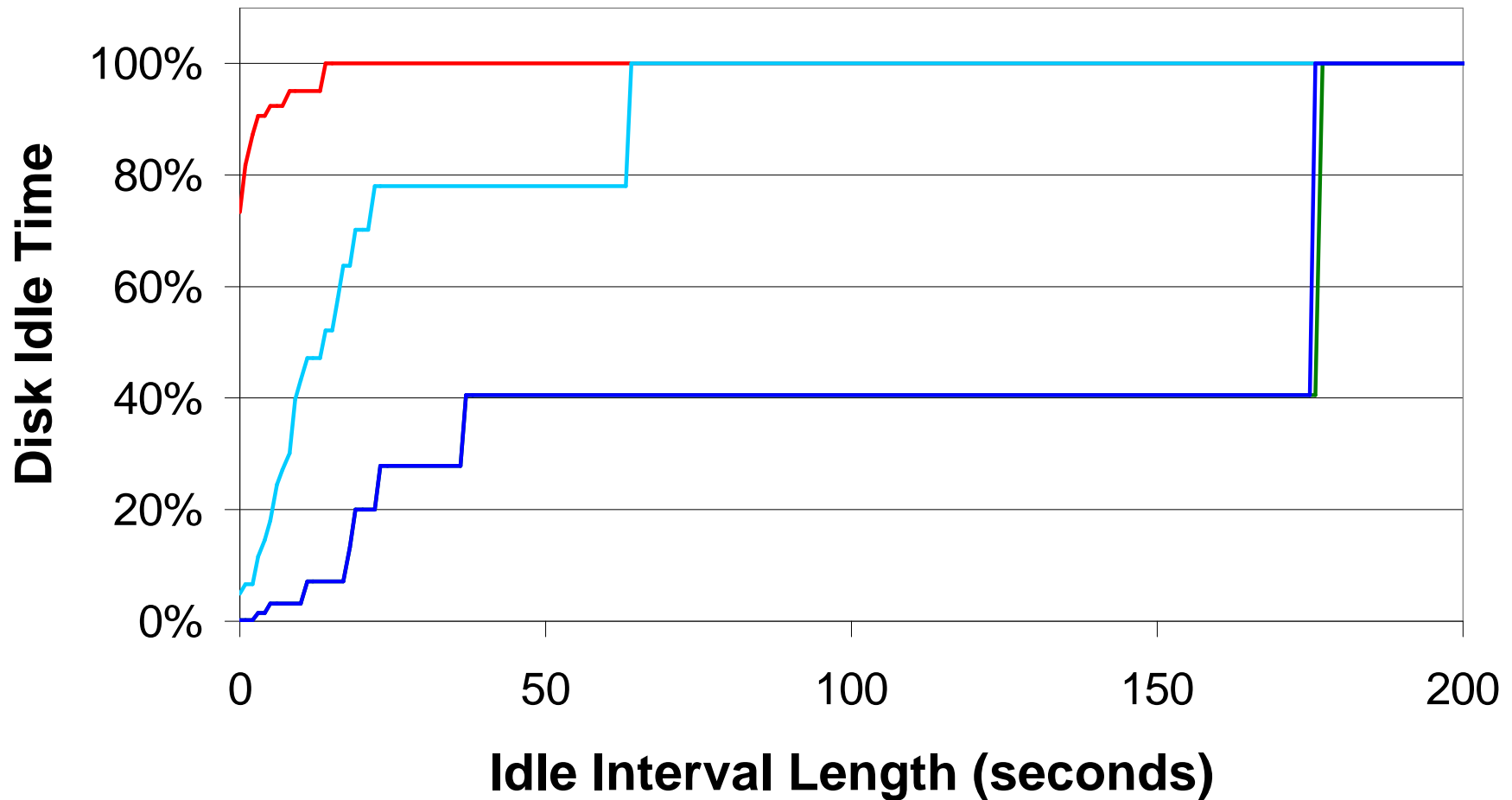
Idle Interval Length Distribution Concurrent Workload



— Bursty-64MB — Bursty-128MB — Bursty-256MB — Bursty-492MB

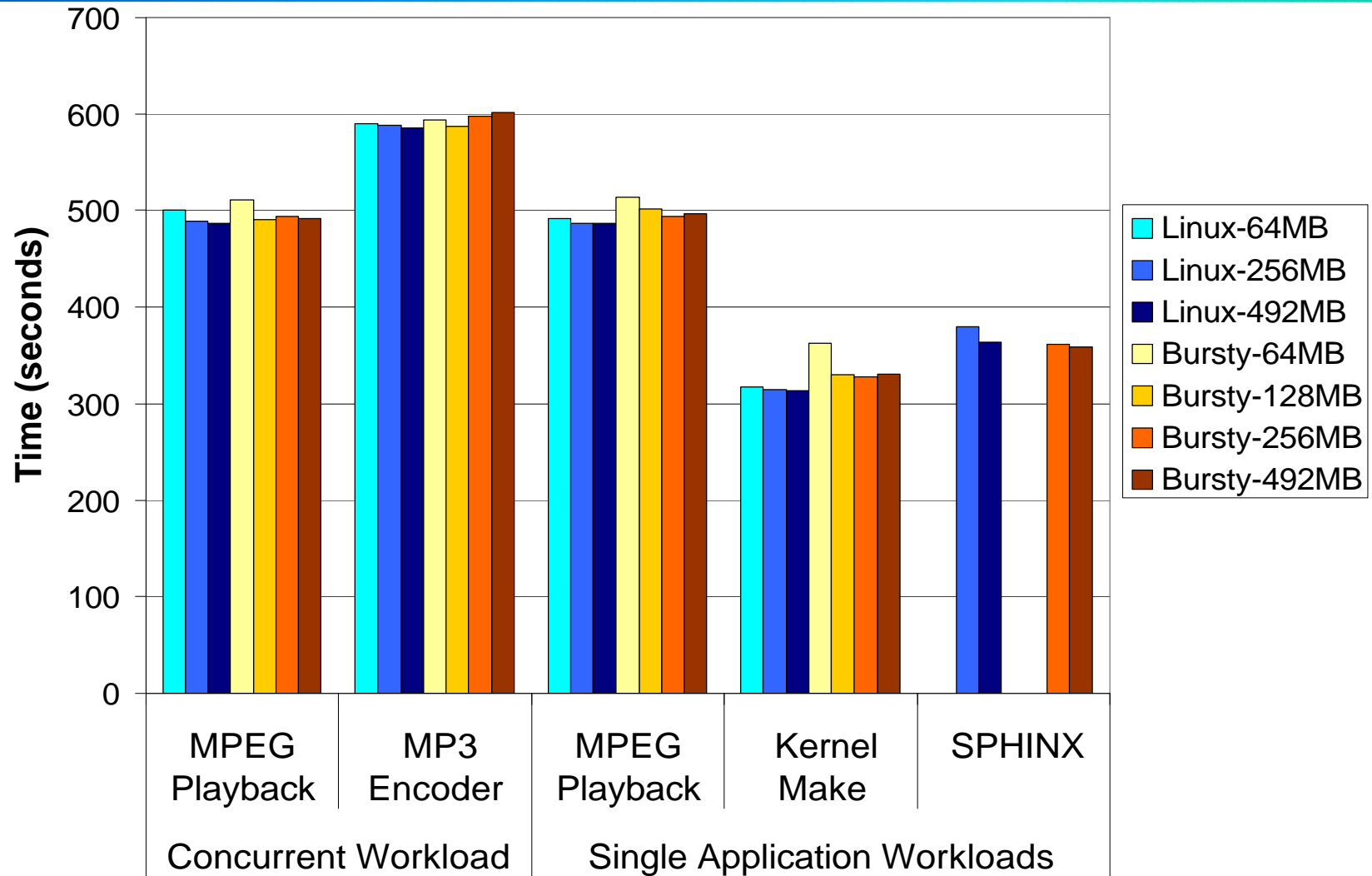
Idle Interval Length Distribution

Kernel Compilation



— Bursty-64MB — Bursty-96MB — Bursty-128MB — Bursty-256MB

Performance Impact



Contributions

- **Rules for prefetching and caching for energy efficiency**
- **Implementation in Linux 2.4.20**
 - Epoch-based algorithm with active and idle phases
 - Dynamic estimation of prefetch cache size
 - Automatic prediction of future file accesses for applications with
 - ✓ Sequential, multiple-file and random accesses
 - Access coordination of concurrently running applications
- **Energy savings of 60-80% with no or minimal performance loss**
 - Savings increase with total system memory

Energy-Efficient Prefetching and Caching

For more info visit:

www.cs.rochester.edu/~papathan/research/BurstyFS/

USENIX 2004

June 30, 2004