

## Energy Efficient Prefetching and Caching

Athanasios E. Papathanasiou and Michael L. Scott

*University of Rochester*

{papathan,scott}@cs.rochester.edu

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

Prefetching and caching are standard practice in modern file systems. They serve to improve performance—to increase throughput and decrease latency—by eliminating as many I/O requests as possible, and by spreading the requests that remain as smoothly as possible over time, resulting in relatively short intervals of inactivity. This strategy ignores the goal of energy efficiency so important to mobile systems, and in fact can frustrate that goal. Magnetic disks, network interfaces, and similar devices provide low power modes that save energy only when idle intervals are relatively long. A smooth access pattern can eliminate opportunities to save energy even during such light workloads as MPEG and MP3 playback.

The aim of our work is to create bursty access patterns for devices with non-operational low power modes, increasing the average length of idle intervals and maximizing utilization when the device is active. At present we are focusing on hard disks. We have modified the memory management and file systems of the Linux 2.4.20 kernel, extending them with novel algorithms and data structures to:

- Quickly identify the working set of the executing job mix and dynamically control the amount of memory used for aggressive prefetching and buffering of dirty data.
- Coordinate the generation of I/O requests among concurrently running applications, so that they are serviced by the device during the same small window of time.

Our kernel extensions are “epoch” based. Each epoch consists of two phases: a *request generation* phase and an *idle* phase. During the request generation phase the operating system attempts to load into memory all data that will be accessed soon. It can afford in this attempt to be highly aggressive: the energy saved by “spinning down” a typical laptop disk is large enough to justify fetching a substantial amount of data “just in case”. To reduce the chance that something important is missed, prefetching accuracy is improved through the use of hints. Newly written applications can provide these hints explic-

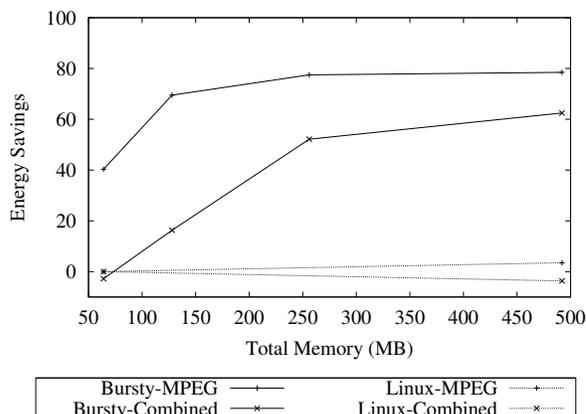


Figure 1: Disk energy savings as a function of total memory size. An 8-fold increase in memory size leads to less than 3% energy savings under standard Linux.

itly. Older applications are supported by a monitoring system that predicts future accesses based on past behavior.

To coordinate prefetching requests across all running applications we introduce a centralized *prefetch daemon* that is responsible for generating prefetching requests for all running applications. During the idle phase the daemon monitors the progress of each application and the status of the file system cache. It predicts the time of the next request and, if that request is far enough in the future, moves the device to a low power mode.

To evaluate our system we use applications with execution times longer than one minute that generate a significant amount of file system activity. Such applications include MPEG and MP3 playback and encoding, data transfer operations (copying), and large-scale compilations. Figure 1 compares the disk energy savings of our *Bursty* policy to that of the standard Linux kernel as total system memory increases. Results are shown for two experimental workloads, MPEG playback (*MPEG*) and concurrent execution of MPEG playback and MP3 encoding (*Combined*). The full poster will include similar results for both clean and incremental rebuilds of the Linux kernel itself.