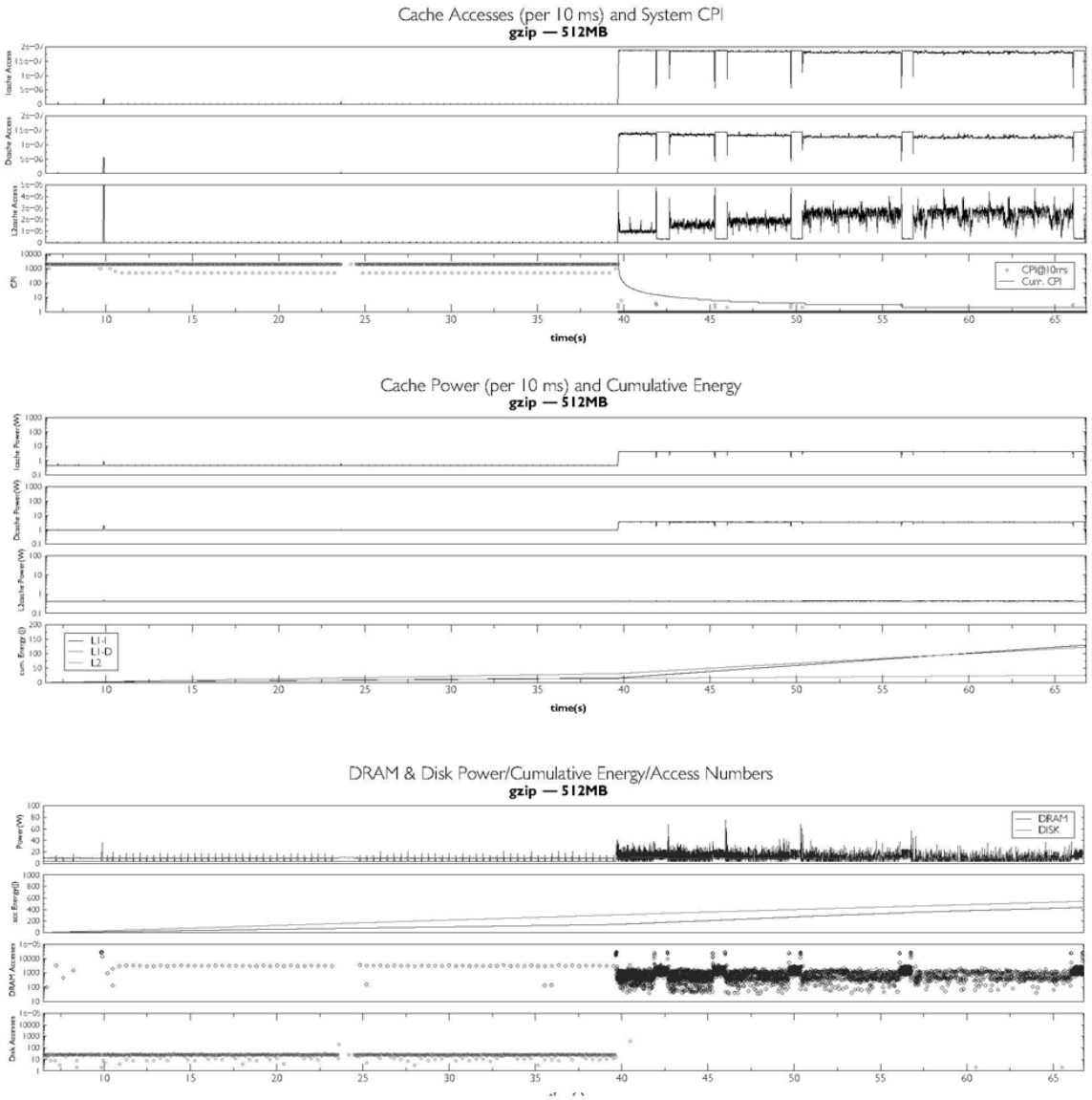

FIGURE 0v.27: Full execution of Gzip. The figure shows the entire run of gzip. System configuration is a 2-GHz Pentium processor with 512 MB of DDR2-533 FB-DIMM main memory and a 12k-RPM disk drive with built-in disk cache. The figure shows the interaction between all components of the memory system, including the L1 instruction and data caches, the unified L2 cache, the DRAM system, and the disk drive. All graphs use the same x-axis, which represents execution time in seconds. The x-axis does not start at zero; the measurements exclude system boot time, invocation of the shell, etc. Each data point represents aggregated (not sampled) activity within a 10-ms epoch. The CPI graph shows two system CPI values: one is the average CPI for each 10-ms epoch, and the other is the cumulative average CPI. A duration with no CPI data point indicates that no instructions were executed due to I/O latency. During such a window the CPI is essentially infinite, and thus, it is possible for the cumulative average to range higher than the displayed instantaneous CPI. Note that the CPI, the DRAM accesses, and the disk accesses are plotted on log scales.

DOCKET ALARM
Find authenticated court documents without watermarks at docketalarm.com.

more compute-intensive phase in which the CPI asymptotes down to the theoretical sustainable performance, the single-digit values that architecture research typically reports.
- By the end of execution, the total energy consumed in the FB-DIMM DRAM system (a half a kilojoule) almost equals that of the energy consumed by the disk, and it is twice that of the L1 data cache, L1 instruction cache, and unified L2 cache combined.

Currently, there is substantial work happening in both industry and academia to address the latter issue, with much of the work focusing on access scheduling, architecture improvements, and data migration. To complement this work, we look at a wide range of organizational approaches, i.e., attacking the problem from a parameter point of view rather than a system-redesign, component-redesign, or new-proposed-mechanism point of view, and find significant synergy between the disk cache and the memory system. Choices in the disk-side cache affect both system-level performance and system-level (in particular, DRAM-subsystem-level) energy consumption. Though disk-side caches have been proposed and studied, their effect upon the total system behavior, namely execution time or CPI or total memory-system power including the effects of the operating system, is as yet unreported. For example, Zhu and Hu [2002] evaluate disk built-in cache using both real and synthetic workloads and report the results in terms of average response time. Smith [1985a and b] evaluates a disk cache mechanism with real traces collected in real IBM mainframes on a disk cache simulator and reports the results in terms of miss rate. Huh and Chang [2003] evaluate their RAID controller cache organization with a synthetic trace. Varma and Jacobson [1998] and Solworth and Orji [1990] evaluate destaging algorithms and write caches, respectively, with synthetic workloads. This study represents the first time that the effects of the disk-side cache can be viewed at a system level (considering both application and operating-system effects) and compared directly to all the other components of the memory system.

We use a full-system, execution-based simulator combining Bochs [Bochs 2006], Wattch [Brooks et al. 2000], CACTI [Wilton & Jouppi 1994], DRAMsim [Wang et al. 2005, September], and DiskSim [Ganger et al. 2006]. It boots the RedHat Linux 6.0 kernel and therefore can capture all application behavior, and all operating-system behavior, including I/O activity, disk-block buffering, system-call overhead, and virtual memory overhead such as translation, table walking, and page swapping. We investigate the disk-side cache in both single-disk and RAID-5 organizations. Cache parameters include size, organization, whether the cache supports write caching or not, and whether it prefetches read blocks or not. Additional parameters include disk rotational speed and DRAM-system capacity.

We find a complex trade-off between the disk cache, the DRAM system, and disk parameters like rotational speed. The disk cache, particularly its write-buffering feature, represents a very powerful tool enabling significant savings in both energy and execution time. This is important because, though the cache's support for write buffering is often enabled in desktop operating systems (e.g., Windows and some but not all flavors of Unix/Linux [Ng 2006]), it is typically disabled in enterprise computing applications [Ng 2006], and these are the applications most likely to use FB-DIMMs [Haas & Vogt 2005]. We find substantial improvement between existing implementations and an ideal write buffer (i.e., this is a limit study). In particular, the disk cache's write-buffering ability can offset the total energy consumption of the memory system (including caches, DRAMs, and disks) by nearly a factor of two, while sacrificing a small amount of performance.

Ov.4.2 Fully Buffered DIMM: Basics

The relation between a traditional organization and a FB-DIMM organization is shown in Figure Ov.28, which motivates the design in terms of a graphics-card organization. The first two drawings show a multi-drop DRAM bus next to a DRAM bus organization typical of graphics cards, which use point-to-point soldered connections between the DRAM and memory controller to achieve higher speeds. This arrangement is used in FB-DIMM.

DOCKET ALARM Find authenticated court documents without watermarks at docketalarm.com.

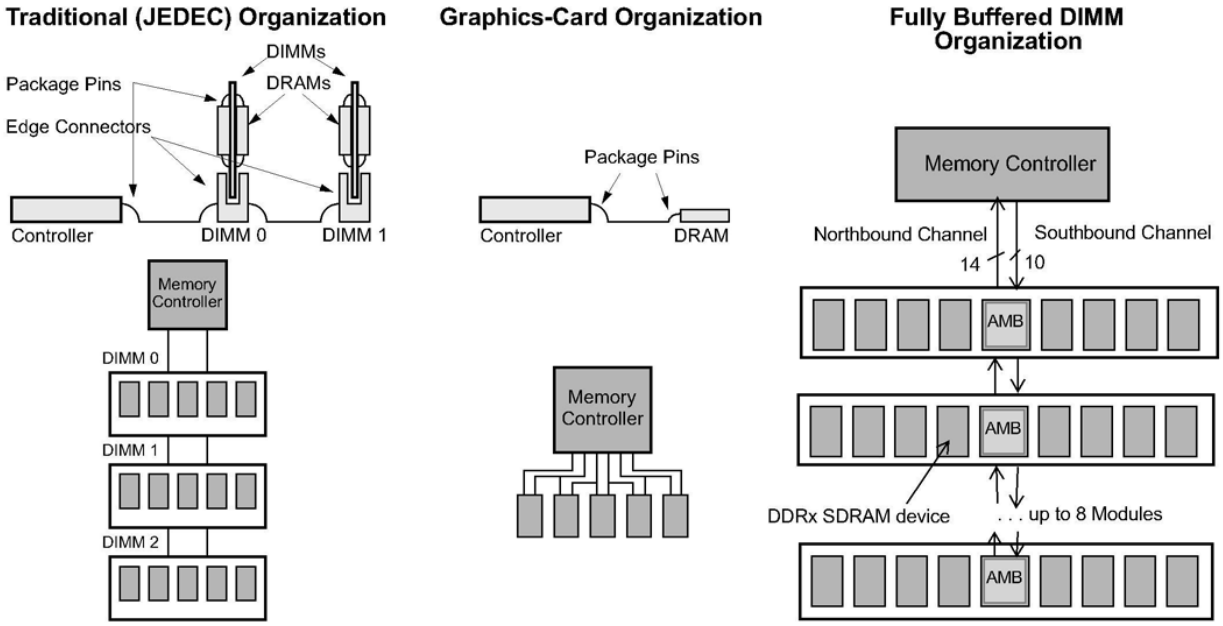

FIGURE 0v.28: FB-DIMM and its motivation. The first two pictures compare the memory organizations of a JEDEC SDRAM system and a graphics card. Above each design is its side-profile, indicating potential impedance mismatches (sources of reflections). The organization on the far right shows how the FB-DIMM takes the graphics-card organization as its *de facto* DIMM. In the FB-DIMM organization, there are no multi-drop busses; DIMM-to-DIMM connections are point to point. The memory controller is connected to the nearest AMB via two unidirectional links. The AMB is, in turn, connected to its southern neighbor via the same two links.

A slave memory controller has been added onto each DIMM, and all connections in the system are point to point. A narrow, high-speed channel connects the master memory controller to the DIMM-level memory controllers (called *Advanced Memory Buffers* or AMBs). Since each DIMM-to-DIMM connection is a point-to-point connection, a channel becomes a *de facto* multi-hop store and forward network. The FB-DIMM architecture limits the channel length to eight DIMMs, and the narrower inter-module bus requires roughly one-third as many pins as a traditional organization. As a result, an FB-DIMM organization can handle roughly 24 times the storage capacity of a single-DIMM DDR3-based system, without sacrificing any bandwidth and even leaving headroom for increased intra-module bandwidth.

The AMB acts like a pass-through switch, directly forwarding the requests it receives from the controller to successive DIMMs and forwarding frames from southerly DIMMs to northerly DIMMs or the memory controller. All frames are processed to determine whether the data and commands are for the local DIMM. The FB-DIMM system uses a serial packet-based protocol to communicate between the memory controller and the DIMMs. Frames may contain data and/or commands. Commands include DRAM commands such as row activate (RAS), column read (CAS), refresh (REF) and so on, as well as channel commands such as write to configuration registers, synchronization commands, etc. Frame scheduling is performed exclusively by the memory controller. The AMB only converts the serial protocol to DDRx-based commands without implementing any scheduling functionality.

The AMB is connected to the memory controller and/or adjacent DIMMs via unidirectional links: the southbound channel which transmits both data

DOCKET ALARM Find authenticated court documents without watermarks at docketalarm.com.

and commands and the northbound channel which transmits data and status information. The southbound and northbound datapaths are 10 bits and 14 bits wide, respectively. The FB-DIMM channel clock operates at six times the speed of the DIMM clock; i.e., the link speed is 4 Gbps for a 667-Mbps DDRx system. Frames on the north- and southbound channel require 12 transfers (6 FB-DIMM channel clock cycles) for transmission. This 6:1 ratio ensures that the FB-DIMM frame rate matches the DRAM command clock rate.

Southbound frames comprise both data and commands and are 120 bits long; northbound frames are data only and are 168 bits long. In addition to the data and command information, the frames also carry header information and a frame CRC (cyclic redundancy check) checksum that is used to check for transmission errors. A northbound read-data frame transports 18 bytes of data in 6 FB-DIMM clocks or 1 DIMM clock. A DDRx system can burst back the same amount of data to the memory controller in two successive beats lasting an entire DRAM clock cycle. Thus, the read bandwidth of an FB-DIMM system is the same as that of a single channel of a DDRx system. Due to the narrower southbound channel, the write bandwidth in FB-DIMM systems is one-half that available in a DDRx system. However, this makes the *total* bandwidth available in an FB-DIMM system 1.5 times that of a DDRx system.

Figure Ov.29 shows the processing of a read transaction in an FB-DIMM system. Initially, a command frame is used to transmit a command that will perform row activation. The AMB translates the request and relays it to the DIMM. The memory controller schedules the CAS command in a following frame. The AMB relays the CAS command to the DRAM devices which burst the data back to the AMB. The AMB bundles two consecutive bursts of data into a single northbound frame and transmits it to the memory controller. In this example, we assume a burst length of four corresponding to two FB-DIMM data frames. Note that although the figures do not identify parameters like t_CAS, t_RCD, and t_CWD, the memory controller must ensure that these constraints are met.

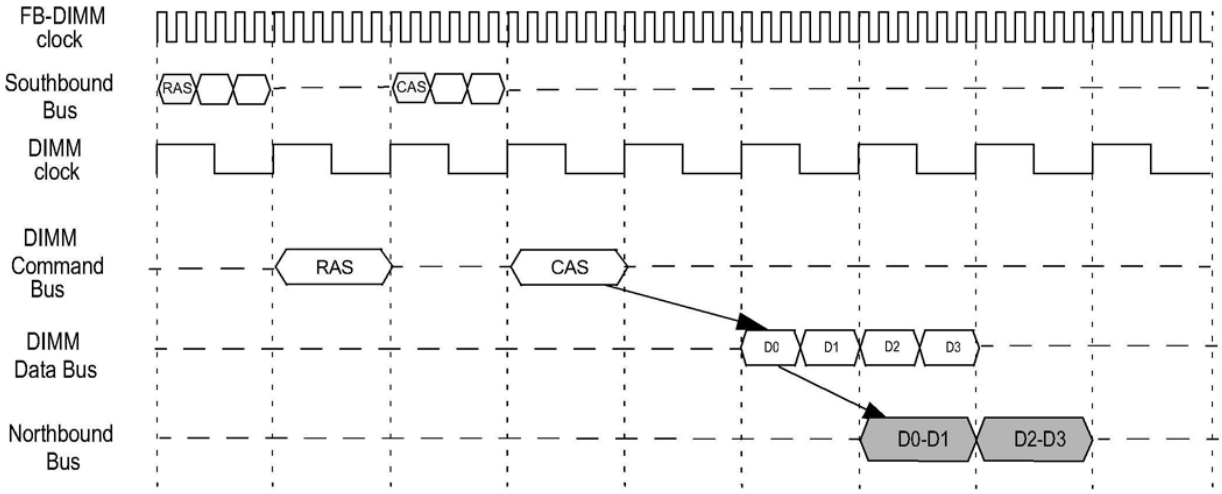

FIGURE Ov.29: Read transaction in an FB-DIMM system. The figure shows how a read transaction is performed in an FB-DIMM system. The FB-DIMM serial busses are clocked at six times the DIMM busses. Each FB-DIMM frame on the southbound bus takes six FB-DIMM clock periods to transmit. On the northbound bus a frame comprises two DDRx data bursts.

DOCKET ALARM
Find authenticated court documents without watermarks at docketalarm.com.

The primary dissipater of power in an FB-DIMM channel is the AMB, and its power depends on its position within the channel. The AMB nearest to the memory controller must handle its own traffic and repeat all packets to and from all downstream AMBs, and this dissipates the most power. The AMB in DDR2-533 FB-DIMM dissipates 6 W, and it is currently 10 W for 800 Mbps DDR2 [Staktek 2006]. Even if one averages out the activity on the AMB in a long channel, the eight AMBs in a single 800-Mbps channel can easily dissipate 50 W. Note that this number is for the AMBs only; it does not include power dissipated by the DRAM devices.

Ov.4.3 Disk Caches: Basics

Today's disk drives all come with a built-in cache as part of the drive controller electronics, ranging in size from 512 KB for the micro-drive to 16 MB for the largest server drives. Figure Ov.30 shows the cache and its place within a system. The earliest drives had no cache memory, as they had little control electronics. As the control of data transfer migrated from the host-side control logic to the drive's own controller, a small amount of memory was needed to act as a speed-matching buffer, because the disk's media data rate is different from that of the interface. Buffering is also needed because when the head is at a position ready to do data transfer, the host or the interface may be busy and not ready to receive read data. DRAM is usually used as this buffer memory.

In a system, the host typically has some memory dedicated for caching disk data, and if a drive is attached to the host via some external controller, that controller also typically has a cache. Both the system cache and the external cache are much larger than the disk drive's internal cache. Hence, for most workloads, the drive's cache is not likely to see too many reuse cache hits. However, the disk-side cache is very effective in opportunistically prefetching data, as only the controller inside the drive knows the state the drive is in and when and how it can prefetch without adding any cost in time. Finally, the drive needs cache memory if it is to support write caching/buffering.

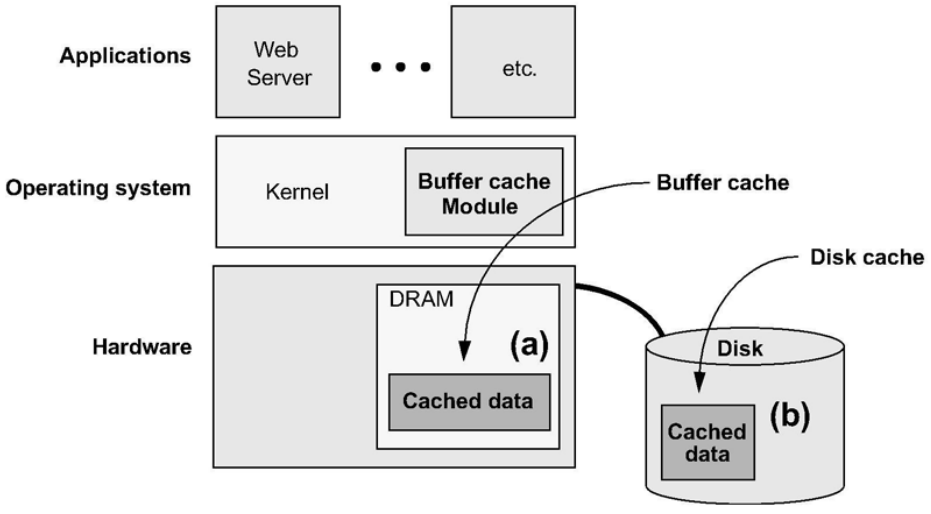

FIGURE Ov.30: Buffer caches and disk caches. Disk blocks are cached in several places, including (a) the operating system's *buffer cache* in main memory and (b), on the disk, in another DRAM buffer, called a *disk cache*.

DOCKET ALARM Find authenticated court documents without watermarks at docketalarm.com.

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.