Technical report: A4TR-2007.1

This page is intentionally left blank.

How migrating 0.0001% of address space saves 12% of energy in hybrid storage

Berend Jan van der Zwaag, Mohammed G. Khatib, Pieter H. Hartel

11 September 2007

Abstract – We present a simple, operating-system independent method to reduce the number of seek operations and consequently reduce the energy consumption of a hybrid storage device consisting of a hard disk and a flash memory. Trace-driven simulations show that migrating a tiny amount of the address space (0.0001%) from disk to flash already results in a significant storage energy reduction (12%) at virtually no extra cost. We show that the amount of energy saving depends on which part of the address space is migrated, and we present two indicators for this, namely sequentiality and request frequency. Our simulations show that both are suitable as criterion for energy-saving file placement methods in hybrid storage. We address potential wear problems in the flash subsystem by presenting a simple way to prolong its expected lifetime.

1 Introduction

To increase performance and to decrease energy consumption, hard-disk manufacturers are moving toward hybrid storage solutions as an enhancement of hard-disk-based storage devices.

In a hybrid storage device the main storage unit is still a hard disk, but enhanced with a flash memory unit. The flash memory has lower latency than the hard disk and consumes less energy. However, because flash memory currently has a higher (25–60 times) cost per gigabyte storage than hard disk storage, its capacity in a hybrid storage device is typically a few orders of magnitude smaller than that of the disk.

If a computer system contains a hybrid storage device, it is not straightforward what is the best way to make use of the hybridity so as to improve performance and reduce energy consumption, while minimising cost. We believe that file placement has a significant impact on performance and energy consumption. We show by trace-driven simulation that migrating a tiny fraction (0.0001%) of the address space from the disk subsystem to the flash subsystem already results in a reduction in storage energy by 12%. This is five orders of magnitude larger than the relative size of the migrated address space.

An additional feature of our file placement methodology is that it does not need to be implemented inside the operating system: it is in principle independent from the operating system, i.e., it may be implemented in any operating system without the operating system directly influencing the method itself. Other solutions to reduce traffic to and from the disk often require implementation in the operating system kernel, which is only possible if the operating system is open source or if operating system vendors include such solutions in their products.

Furthermore, our methodology allows a simple solution to handle flash wear issues. We will show that we can prolong the predicted lifetime of the flash subsystem in our hybrid storage system up to its desired lifetime, using redundancy.

In the next section we present the problem and our methodology in more detail. Section 3 gives the background for our methodology along with related work from literature. Our experimental setup and models are explained in section 4, followed by results from our simulations in section 5. Section 6 shows how we can prolong the predicted lifetime of the flash subsystem and section 7 concludes.

2 Problem statement

We consider our system environment to be a mobile computer with limited energy budget and with a hybrid storage system consisting of two storage subsystems, where the total logical address space is spread over all storage subsystems. In our case study we consider a hybrid storage system consisting of cheap high-capacity hard-disk storage and more expensive but low-latency flash memory.

The problem can then be described as follows: what

are the relations between the size of the flash memory (relative to the total storage capacity), the energy consumption of the total hybrid storage device, and its cost?

2.1 Road to solution

To solve the above-stated problem, we first need to build some models. Firstly, we require a model of the mapping of requested logical addresses to physical sectors on the hybrid storage device. Secondly, we need a model of how much energy a given request would consume on the hard disk and how much energy it would consume on the flash memory. Thirdly, we need a model for the cost of the total hybrid storage system, depending on the relative size of the flash memory compared to the disk subsystem.

Finally, we require a placement algorithm to decide to which storage subsystem – disk or flash – a given storage I/O request should be sent. We study file placement methodologies using a modern and realistic workload trace. Implementations of file placement algorithms can be applied offline, e.g., based on a recorded trace, or online, in which case the algorithm is dynamically optimising the address migration.

Our current study is in fact a case study based on a real-world one-month-long workload trace of storage I/O requests to a hard disk in a notebook (Chang and Kuo 2005). By taking out particular requests (decided by the placement algorithm using two criterions that we present in this paper, namely sequentiality and request frequency) and moving them to a flash memory, we can obtain an indication for potential improvements in energy consumption, based on our models and the workload trace. The cost model then gives an indication of the related cost.

Because at this stage we are only interested in the effect of file placement on energy consumption, we do not make any optimisation in scheduling or any optimisation in the separate storage subsystems. We consider requests coming in one after the other (basically, scheduling on a first-come-first-serve basis, i.e., the NOOP scheduler) and without any smart buffering, which would improve performance and energy consumption of the disk subsystem (e.g., burst-mode with gaps between the bursts large enough to spin down the disk). We ignore recorded time stamps and feed the requests to our system in a continuous manner, so as to keep the system busy all the time until the end of the trace. Effectively, our study is a worst-case scenario, therefore our results will provide a lower bound for seek energy reduction.

This way we can study the pure effect of data placement in hybrid storage compared to disk storage, without the results being influenced by other optimisations. A later experiment will include more optimisations and should therefore produce more realistic results, but, without studying the effect of single optimisation methods, we will not be able to state how much of the composite improvement is the result of a particular optimisation method.

3 Background

In this section we give some background for hybrid storage systems. For instance, what did hybrid storage solutions look like in the past and what types of hybrid storage systems are currently available. We illustrate these questions with existing solutions from literature.

3.1 Inspiration from the past

In 1974 IBM implemented Hierarchical Storage Management (HSM) in the IBM 3850 (Johnson 1975). It stored frequently accessed files on disk and automatically migrated less frequently used files to tape. The total mass storage system appeared to the CPU as a disk. Basically, the disk subsystem was used as a cache for the tape subsystem. The tape subsystem actually was a tape library holding many tape cartridges (each with a capacity of 50 MB) that were accessed automatically by a robot and placed into a reader/writer unit. The 3850 can be regarded as the precursor to modern tape libraries.

In those days hard-disks were expensive and tapes the cheaper alternative with high latency. Nowadays hard-disks are considered cheap with high latency (i.e., for random access) and a (more expensive) flash memory can be used as a cache to lower the latency. Anyway, the principle remains the same, regardless of the exact implementation of hierarchical storage in a hybrid storage system.

3.2 Saving energy in conventional disk storage systems

In current operating systems many optimisations have been implemented to save seek energy and/or total storage energy, and more solutions exist in literature. Most of these deal with a reorganisation of metadata files. For instance, Mullender and Tanenbaum (1984) propose to store the first part of a file in its inode (Unix equivalent of a metadata file) to enhance performance. Ganger and Kaashoek (1997) enhance this idea using embedded inodes and explicit

grouping. A similar idea has also been implemented in NTFS, where small files and folders (typically 1,500 bytes or smaller) are entirely contained within their respective metadata records,[1] thereby saving the extra seek to and transfer of the files themselves.

Such existing enhancements will also have their merits in hybrid storage systems, but using the hybridity explicitly additional improvements can be achieved, as we will show later.

3.3 Hybrid storage systems

A block storage device such as a hard disk is interfaced to the system as a contiguous linear storage device. Every block on the storage device is assigned a logical number called its logical block address (LBA). Every LBA is transformed into a physical block address (PBA) on the storage device by mapping the corresponding cylinder/track/sector to be located by the read/write head.

There are many possible configurations of hybrid storage systems, using different storage subsystems. For instance, three different configurations using NOR flash, NAND flash, and battery-backed SDRAM (but without a hard disk) are investigated by Lee and Chang (2005).

However, all hybrid storage systems can be categorised into two main categories; we distinguish between serial hybridity and parallel hybridity. In serial hybrid storage systems a small but fast (and expensive) storage subsystem is used as a cache for a large (and cheap) storage subsystem which holds the total address space to itself. In parallel hybrid storage systems the address space is spread over all storage subsystems combined.

3.3.1 Serial hybridity

Most works about hybrid or heterogeneous secondary storage systems investigate systems where one storage type acts as a buffer or a cache for the other. For instance, Khatib et al. (2007*a*) interpose flash as a buffer between disk drive and DRAM to save up to 40% energy in streaming applications, while at the same time reducing the demand for DRAM capacity.

Many power saving strategies are discussed by Lorch and Smith (1998), among which flash memory as a disk cache and flash memory as a replacement for disk. For the first case – flash as a cache for disk – some old figures (Marsh et al. 1994) show that energy consumption of the secondary storage system for mobile computers can be decreased by 20–40% while at the same time improving the I/O response time by 30–70%.

Other figures from the same year show a reduction in power consumption of 60–90% when a disk is replaced by a flash memory (Douglis et al. 1994). These figures are obtained by simulations on mobile computers but not hand-held computers, for which similar numbers can be found (in our experiments on an HP iPAQ, a Compact Flash storage card uses up to ten times less energy than a Microdrive of the same capacity and format factor (Khatib et al. 2007*b*)).

The new Microsoft operating system Windows Vista includes features that use hybrid storage to boost performance and decrease energy consumption.[2] Windows ReadyBoost uses external flash memory devices, such as USB sticks, as a cache for the hard-disk to boost performance. Increased responsiveness is achieved also by an improved prefetching algorithm called Windows SuperFetch.[2]

3.3.2 Parallel hybridity

One paper on parallel hybridity (Edel et al. 2004) concentrates on the compression of small files (such as file metadata) in a hybrid storage system consisting of a disk as well as non-volatile RAM. The authors do not assume the use of any particular kind of non-volatile memory technology (battery-backed SDRAM, flash or MRAM, to name a few), but assume that it can be mapped directly into the system address space. Their results indicate that a hybrid file system including a compressed non-volatile memory component offers a significant speed improvement over a typical disk-only file system, while at the same time requiring significantly fewer resources than hybrid file systems that do not take advantage of compression. How to handle on-disk allocation was left open, however. We address this issue by identifying the most frequently accessed sectors and migrate those from disk to flash.

An area where file allocation has been addressed already, is in storage architectures that contain multiple disks (e.g. in a RAID configuration), which, in a sense, can also be considered as hybrid storage systems, in particular if the disks differ in capacity, performance and/or energy consumption. Improvements are then possible by applying conservation techniques, such as Popular Data Concentration (PDC), which migrates frequently accessed data to a subset of the disks (Pinheiro and Bianchini 2004), or the Massive Array of Idle Disks (MAID), which relies on temporal locality to place copies of files on a

[1] http://www.microsoft.com/technet/prodtechnol/windows2000serv/reskit/prork/prdf_fls_xkhv.mspx

[2] http://www.microsoft.com/windowsvista/features/foreveryone/performance.mspx

subset of the disks (Colarelli and Grunwald 2002).

Furthermore, Kim et al. (2006) propose a file placement technique based on the request type: it adapts an existing data concentration technique by separating frequent read and frequent write I/O requests. This differs from our approach, as we use access frequency and sequentiality (see section 4.3.1) as criteria for separating storage I/O requests. Later research may show that a combination of both approaches may lead to even better results, but we will study the merit of simple placement criteria before moving on to more complicated approaches.

Another study (Zheng et al. 2003) looks at performance and energy impacts of three different storage technologies for mobile hand-held computers: a Microdrive, a flash card and a wireless LAN card (for remote storage), but not at any combinations of these. Nevertheless, Zheng et al. conclude their paper with the conjecture that a judicious combination of the different technologies may play an important role in mobile storage systems. In our paper, we explore this conjecture for a combination of hard disk and flash memory.

Apart from the features mentioned in the previous section, i.e., Windows ReadyBoost and Windows SuperFetch, Microsoft Windows Vista offers another new feature called Windows ReadyDrive, which also uses hybrid storage systems to boost performance and reduce energy consumption. It stores key system data in the flash memory subsystem and uses it as a write cache for the hard-disk subsystem as well. In this regard, it can be seen as a mixture of serial and parallel hybridity. As a result of a cooperation between Microsoft and Samsung, the first hybrid hard drives have become available on the market by early 2007.[3]

4 Experimental setup

The work described in this paper is of an exploratory nature to investigate the merit of file placement in a hybrid storage device. Using trace-driven simulation we indeed confirm our expectation that energy consumption of a hybrid storage device can be reduced by a proper division of the address space.

In the following subsection we describe the workload trace on which we base our simulations. Section 4.2 presents the models that we use for our simulations along with a description of our simulation tool. In section 4.3 we present two criteria that can be used in file placement algorithms: one based on request sequentiality and one based on request frequency.

[3] http://www.samsung.com/global/business/hdd/

4.1 Workload trace

The workload trace that we use in our case study was supplied by Chang and Kuo, the authors of a paper on efficient management for large-scale flash memory storage systems (Chang and Kuo 2005). In their work, Chang and Kuo use the trace for comparing two different approaches for flash storage management. In the remainder of this paper, we refer to this trace as the CK04 trace.

Quoting the authors, the trace is one of a "typical workload of common people," as observed on an IBM-X22 ThinkPad, a notebook with a 20 GB harddisk[4], NTFS as its file system and Microsoft Windows XP as its operating system. Recorded are for each storage I/O request the time stamp (i.e., time of day), the requested logical address, the request size and the request type, i.e., READ or WRITE. The trace was recorded over a period of one month, of which 26 days show activity in the trace, i.e., there are 25 requests that have a smaller time stamp than the immediately preceding request in the trace.

The activities on the notebook consisted of web surfing, e-mail sending and receiving, movie downloading and playing, document typesetting, gaming, and background activities such as various operating system-specific activities and scanning for viruses. Unfortunately, no record was kept of which application was running at what time.

4.2 Simulation models and tools

We have carried out a number of experiments by means of simulation, using the CK04 trace as a starting point.

The simulated hard disk in the hybrid system has the same capacity in each experiment, namely 20 GB, i.e., the same capacity as the hard disk used for the CK04 trace. For the simulated disk subsystem, we use disk parameters of a popular notebook hard disk with the same capacity, the 2.5 inch IBM Travelstar 20GN (IBM Storage Technology Division 2000).

The capacity of the flash memory subsystem within the hybrid storage system varies throughout the experiments, and ranges from 20 MB to 2 GB, or 0.1% to 10% of the hard disk capacity, which itself remains unchanged. The addresses that are migrated to flash are left unused on the disk. Following these experiments, which essentially form a feasibility study, we test one configuration in which the capacity of the flash memory subsystem is just 20 KB.

[4] Even though in their paper Chang and Kuo (2005) mention that the harddisk had been replaced with flash-memory devices, this was in fact not the case when the trace was generated [L.-P. Chang, personal communication]

Table 1: Parameters used in our models

parameter	symbol	value	source
number of sectors		40 million	CK04 trace (Chang and Kuo 2005)
min. seek time	$\min(t_{\text{seek}})$	2.5 ms	IBM Travelstar 20GN specs (IBM Storage Technology Division 2
max. seek time	$\max(t_{\text{seek}})$	23.0 ms	IBM Travelstar 20GN specs (IBM Storage Technology Division 2
idle spinning power	P_{rot}	1.85 W	IBM Travelstar 20GN specs (IBM Storage Technology Division 2
seek power	P_{seek}	2.30 W	IBM Travelstar 20GN specs (IBM Storage Technology Division 2
disk read/write power	P_{diskRW}	2.05 W	IBM Travelstar 20GN specs (IBM Storage Technology Division 2
flash read/write power	P_{flashRW}	0.20 W	our measurements
time to R/W one sector	$t_{\text{RW}}(1 \text{ sector})$	25 µs	avg. media transfer rate of 160 Mb/s

4.2.1 Modelling logical to physical block address mapping

In each experiment, we move part of the total address space as recorded in the disk trace, to a hypothetical flash memory. The combination of the flash memory and the remainder of the address space on disk is our hypothetical hybrid storage device.

We assume that the disk is fairly new and that we can therefore assume a linear one-to-one mapping of the logical block addresses (LBAs) to the physical block addresses (PBAs), which is also what most operating systems assume. This assumption becomes less valid when a hard disk is ageing, as more and more bad blocks will appear and have to be remapped, out-of-order. However, this is outside of the scope of our investigation.

4.2.2 Modelling the cost

Although flash memory generally has lower power consumption than a hard disk, it does come at a cost: currently, a flash memory with the same capacity of a typical modern hard-disk (a few hundred GB) would cost at least 25 times as much as the hard disk, based on the lowest prices per GB for current devices: about 6 €/GB of flash (May 2007 spot price for flash chips around 0.75 €/Gbit) and about 0.24 €/GB of hard disk (e.g., 250 GB for 60.00 €). In practice, prices for high-capacity flash-based solid state disks (SDDs) are even higher (max. 128 GB SDD for about 2000 €, or 65 times the price of an equally-sized hard disk).

Even though prices per gigabyte of flash are falling by 30 to 40 percent per year (Goldstein 2006), the prices per gigabyte of hard disk are falling at the same rate (40 to 50 percent per year according to Thompson and Best (2000)). Therefore, for our experiments, we assume the flash memory subsystem to cost 25 times as much per gigabyte as the hard disk subsystem. So, if we replace the hard disk with a hybrid storage system containing 4% flash memory, the price will double ($25 \times 4\% = 100\%$ extra) compared to the original disk.

This simple cost model does not account for extra overhead costs, such as more complex controllers, data channels, and other hardware. On the other side, we also ignore the resulting reduction in battery cost due to the energy reduction that we achieve. In practice, the battery is likely to remain the same anyway, but with a longer service time per charge as a bonus.

4.2.3 Modelling seek energy

Table 1 lists the values of the disk parameters that we use. Figure 1 shows the resulting seek energy profile for the disk subsystem in our hybrid storage system, as a function of the number of sectors between two consecutive requests. The underlying seek model is detailed in the appendix. The seek energy is mainly consumed by the spinning of the disk for the duration of the seek operation, as can also be concluded from the difference in power between idle spinning and seeking (table 1 shows that this holds for read and write power as well). The seek profile of figure 1 matches measured seek energy profiles of hard disks, such as reported by Ruemmler and Wilkes (1994), Stadlander (2004) and Zedlewski et al. (2003).

4.2.4 Simulation tool

We have developed a simulation package in Matlab, which contains tools for trace analysis as well as for trace-driven storage system simulation. The analysis tools are able to extract and to graphically present a large variety of statistics from disk traces and the simulator uses the parameterised models described above to simulate timing and energy aspects of a storage system (containing disk, flash, or both), using a workload trace as input.

Because it uses models at the system level rather than at the physical level, the simulator is able to

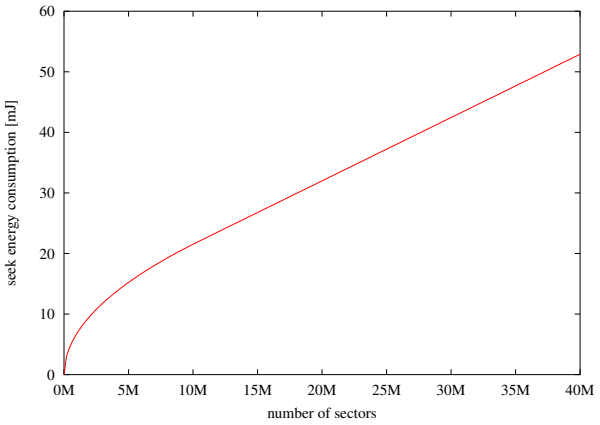

Figure 1: Energy consumption model of seek operations (see Appendix for more details)

process many (more than one million) trace records (i.e., requests) in parallel instead of processing them one by one. This makes it fast and enables multiple simulation runs (e.g., with different settings per run) in a relatively short time.

4.3 Methods for file placement

To decide to which storage subsystem – disk or flash – a given storage I/O request should be sent, we require a file placement algorithm. We study criterions for two such placement algorithms, namely one based on request sequentiality and one based on request frequency. The placement algorithm determines which part of the logical address space is mapped onto the flash memory and consequently which part remains mapped onto the hard disk.

4.3.1 Sequentiality

Because seek times depend largely on the address differences between successive requests, we expect that workloads with many sequential requests (i.e., sequential in the address space as well as in time) will have shorter average seek times than workloads with many non-sequential requests. As a consequence, we expect the seeks generated by workloads with high sequentiality to consume less energy than the ones generated by workloads with the same number of requests, but with lower sequentiality.

To find out the relation between average seek time (hence seek energy consumption) and sequentiality of requests, we use the following definition:

Definition 1 *A request Q_n is said to be sequential if its first requested sector is right next in the address space to the last requested sector of the immediately preceding request Q_{n-1}:*

$$seq(Q_n) := (addr(Q_n) = addr(Q_{n-1}) + size(Q_{n-1}))$$

The overall sequentiality of a workload trace is then defined as the number of sequential requests as a percentage of all requests in the trace.

In the ideal case, if requests are sequential in the logical address space, they are also sequential in the physical address space. This means that sequential requests do not incur seeks, apart from a seek to the first request of a sequence or in case of a track switch. Unfortunately, this ideal case is not always true for real hard-disks. The number of bad sectors on a disk increases with the age of the disk (as an indication Pinheiro et al. (2007) report that 9% of the disks in a large population of disks between 0 and 5 years old have a bad sector reallocation count of more than 0). Each track has a number of spare sectors, that can be used for remapping bad sectors. When a track has more bad sectors than spare sectors, remapping to different areas of the disk is applied. This has an impact on the performance of the disk, as more seeks will be necessary, in particular when sequential logical sectors contain remapped sectors in the physical address space. Nevertheless, our simple model assumes a purely linear mapping of the logical address space to the physical address space.

In each of our experiments we will calculate the sequentiality of that part of the trace that contains requests to the address space that is still mapped onto the disk subsystem and determine the average seek time of all requests to the disk storage subsystem. As indicated above, we expect to find a correlation between this sequentiality and the average seek time.

Hypothesis 1 *The higher the sequentiality of a (sub)trace, the lower the average seek time of the requests in that (sub)trace.*

4.3.2 Request frequency

If some part of the address space is much more frequently visited than the rest of the address space, than we can save energy for reading and writing if that part is relocated to the flash memory subsystem, because reading and writing from/to flash consumes less energy than reading and writing from/to disk, as already mentioned above. Less straightforward is the expectation that this may also reduce the average seek time (and hence seek energy) for the remaining requests on disk. However, if the relocated part of the address space is frequently visited throughout the entire duration of the trace, there will be many

seek operations from and to this part of the address space. So, if it is relocated to flash, those seeks are avoided, and the average seek time decreases. This is true in particular if the relocated addresses are at either end of the address space, because the maximum seek distance in the remaining disk address space will be reduced.

Hypothesis 2 *The higher the request frequency (or hotness) of the relocated part of the address space, the lower the average seek time of the requests in the remaining part of the address space.*

5 Results

This section describes the main results of our simulations. We show that energy consumption can be reduced if even the tiniest amount of addresses is remapped from the hard disk subsystem to the flash memory subsystem of a hybrid storage system. The actual energy reduction depends on the size of the block, where it is located in the address space, and the nature of the requests to those addresses (i.e., sequential or not). The larger the flash memory subsystem we take, the more energy we can save, but the more we need to pay for it as well. So, a trade-off will arise. We will have a look at this after discussing the results in more detail below.

5.1 Migrating a chunk of LBA space from disk to flash

As a feasibility study, we first conduct a series of simulation experiments for a flash memory subsystem with a capacity varying from 0.1% to 10% of the 20 GB disk capacity, i.e. ranging from 20 MB to 2 GB. For each capacity, we calculate the consumed storage energy as a function of the original location of the migrated block of addresses. Figure 2 shows the resulting energy consumption for the best case and the worst case per series of up to 1000 experiments. If the moved addresses are infrequently requested or if many of the requests to these addresses are sequential, there is no significant improvement in energy consumption, regardless of the size of the chunk of LBA space that is moved from disk to flash. However, if the moved addresses are frequently requested or if many of the requests to these addresses are non-sequential, there is a significant reduction in energy consumption, ranging from 10% energy reduction if a block of 0.1% of all addresses is moved, to 32% energy reduction if a block of 10% of all addresses is moved from disk to flash.

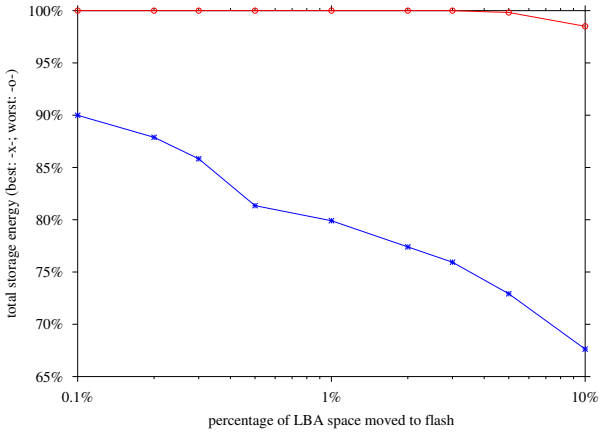

Figure 2: Storage energy consumption as a function of the relative size of the flash memory subsystem

To show the importance of selecting which blocks are moved from the disk subsystem to the flash subsystem – even without looking at the characteristics of the requests other than their addresses – we show the worst and best results when 10% of the address space is moved to flash in table 2.

The worst result is obtained when all requests to addresses located between 60.7% and 70.7% into the address space are migrated from disk to flash. Table 2 shows that this decreases the total seek energy consumption of the hybrid storage system by only 0.8%, even though 2.5% of all requests are redirected to flash. Even if we include read/write energy consumption, the reduction is only 1.5%, despite the fact that reading and writing in flash consumes 90% less energy than on disk. It appears that many requests to the selected block of 10% of the address space are sequential requests, which are typically handled by the disk subsystem more efficiently than non-sequential requests. The sequentiality decreases by about 6%, as table 2 shows.

In contrast, the sequentiality of the remaining requests to the disk subsystem is improved when all requests to addresses located between 5.7% and 15.7% into the address space are moved from disk to flash. Most of the requests to this 10% block of addresses are non-sequential requests, which are typically inefficient for disk storage. Therefore, the total storage energy reduction of 32.37% is more than we would expect purely based on the amount of migrated requests, namely 23.52% of all requests. The sequentiality of the remaining requests increases by almost 24%, so, not only is this 10% block of addresses frequently accessed, but also are most of the requests non-sequential indeed, leading to additional energy

Table 2: Effect on the seek energy consumption of the hybrid storage system when a block of 10% of the address space is located on the flash memory subsystem, and the influence of which 10% block of the original address space is relocated

parameter	old	new, worst case		new, best case	
no. of requests to disk [$\times 10^6$]	3.14	3.06	(−2.5%)	2.40	(−23.5%)
no. of sequential requests [$\times 10^6$]	.824	.760	(−7.9%)	.781	(− 5.3%)
sequentiality [%]	26.2	24.8	(−5.5%)	32.5	(+23.8%)
average seek time on disk [ms]	3.97	4.04	(+1.8%)	3.26	(−18.0%)
total seek energy [kJ]	28.7	28.5	(−0.8%)	18.0	(−37.3%)
total storage energy [kJ]	34.9	34.4	(−1.5%)	23.6	(−32.4%)

reduction. This confirms our hypothesis of section 4.3.1.

Summarising, table 2 shows that it is important which part of the address space is moved to the flash subsystem.

5.1.1 Tradeoff between energy reduction and cost price

Unfortunately, the reduction in energy consumption of hybrid storage has a price tag, mainly caused by the high cost of flash memories, as explained in section 4.2.2. According to our model and our simulations, in order to save one third on storage energy consumption, the cost of the hybrid storage system will be three times as high as the cost of just a hard disk with the same capacity. So, in most situations a trade-off is necessary.

This is expressed by the receiver operating curve (ROC) in figure 3, where the ideal point would be in the bottom left corner – outside the shown area, at (100%, 0%) –, i.e., maximum energy consumption reduction at no extra cost. However, this is not an option. The available options lie on the line in the graph. Furthermore, the trade-off graph shows that the first 10% of energy consumption reduction can be obtained at virtually no extra cost, a 20% energy reduction costs about 25% extra, and a 30% reduction costs about three times as much as the original disk storage system.

5.1.2 Relation between sequentiality and average seek time

As we have already briefly mentioned, there is a clear relation between the sequentiality of a (sub)trace and its energy consumption. We have plotted the sequentiality calculated during our experiments against the average seek times in figure 4. This confirms what we already expected in section 4.3.1, namely that there is a correlation between them (the correlation coefficient is −0.81). As can be seen in the graph, our measurements confirm hypothesis 1: more sequentiality leads to lower seek times.

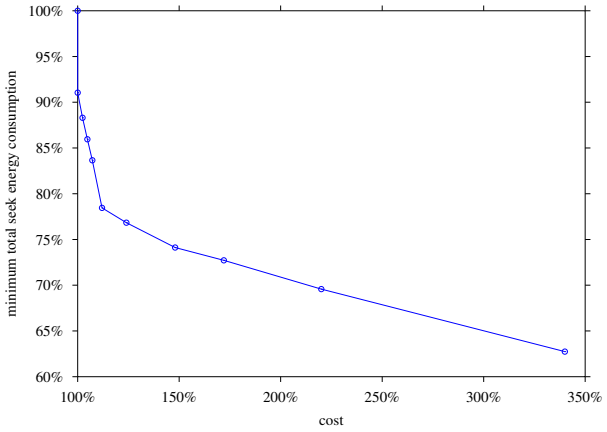

Figure 3: Receiver operating curve of storage cost and seek energy consumption

Therefore, it is beneficial to keep sequential data as much as possible on the disk storage subsystem, and move non-sequential data to the flash memory subsystem.

5.1.3 Relation between request frequency and average seek time

The results in table 2 indicate that a higher request count in the part of the address space that is mapped to the flash subsystem leads to a smaller average seek time for the remaining requests on disk. This is more clearly illustrated by figure 5, which indeed shows a correlation between them: the correlation coefficient is −0.78.

This confirms hypothesis 2: higher relocated request frequency leads to lower seek times. We will use this in the following section, where we relocate a very small part of the address space that has a very

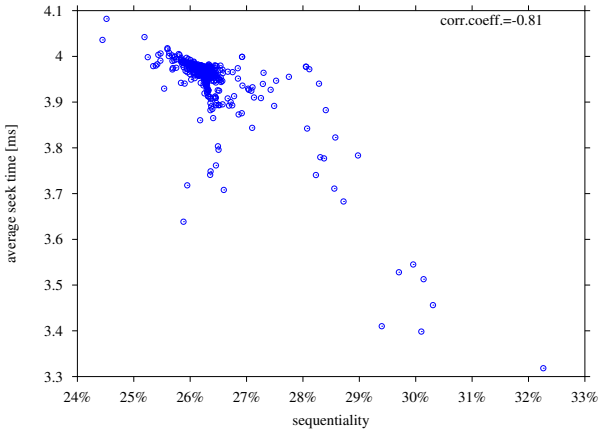

Figure 4: Average seek time per request to the disk subsystem plotted against the sequentiality of the requests to the disk

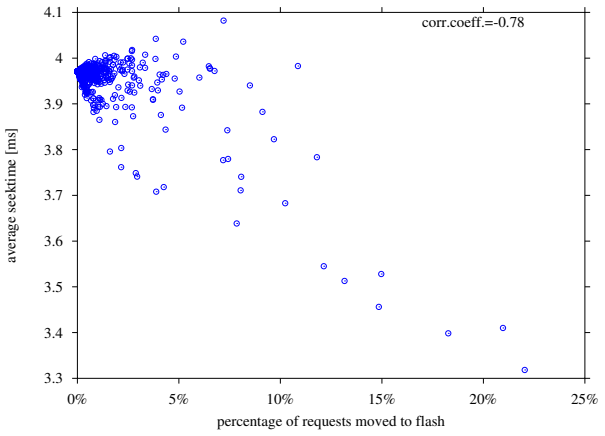

Figure 5: Average seek time per request to the disk subsystem plotted against the percentage of all requests relocated to the flash subsystem

high request frequency.

5.2 Migrating 0.0001% of LBA and saving 12% of energy

The results from the first series of experiments show that energy consumption of the storage system can be reduced by migrating part of the address space from disk to flash. However, they also show that the key to success is to locate the addresses that yield most energy reduction when migrated.

To stress this even more clearly in the following experiment, we use the above confirmation of hypothesis 2 and count how often each LBA is requested in the CK04 trace. The top 1000 most frequently requested LBAs are shown in figure 6. The peak in the middle is part of the Master File Table ($MFT), the area where the file system (NTFS) stores its meta data. This is the first area that one would migrate off disk to save energy.

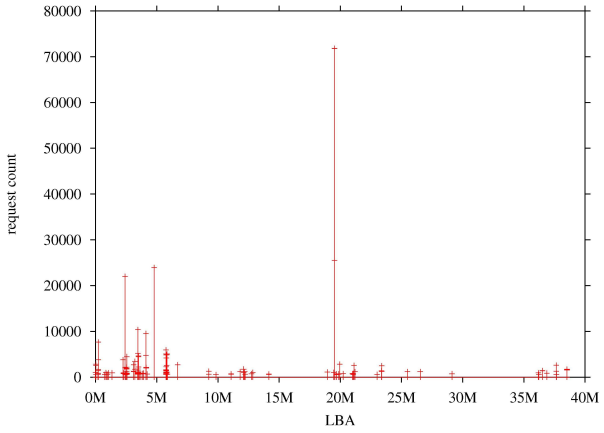

Figure 6: Request count of the 1000 most frequently requested addresses. Many of the spikes represent multiple addresses, as request sizes are often greater than one sector (typical request size is 8 sectors).

So, we choose 40 LBAs represented by the high peak in the middle (which is 32 sectors wide) and by several peaks in the lower address range and migrate those from disk to flash. These 40 sectors (of 512 bytes each) form a mere 20 kB, or 0.0001% of the total LBA space.

Out of the total of 3,142,725 requests, 270,572 requests (i.e. 8.6%) lie completely in the selected area.

Table 3 presents results based on the simple models described earlier. From these results, we can conclude that relocating a tiny fraction of the LBA space from hard disk to flash reduces the energy consumption for seek operations on the hard disk by about 14% and the total hybrid storage energy consumption by about 12% cf. table 3. This may not seem much, but we recall that only 0.0001% (in fact, just 20 kB) of the total LBA space has been moved to the flash memory subsystem.

The complete trace contains 824,442 sequential requests (i.e., the original sequentiality is 26.2%). After migration of the 40 sectors to flash, the absolute number of sequential requests remaining on disk increases to 833,853 (sequentiality 29.0%). This increase is caused by the migration of requests that previously interrupted sequential requests. Every time when request migration causes the (time-wise) surrounding requests to become (space-wise) sequential, two seeks are saved, namely one to and one from the

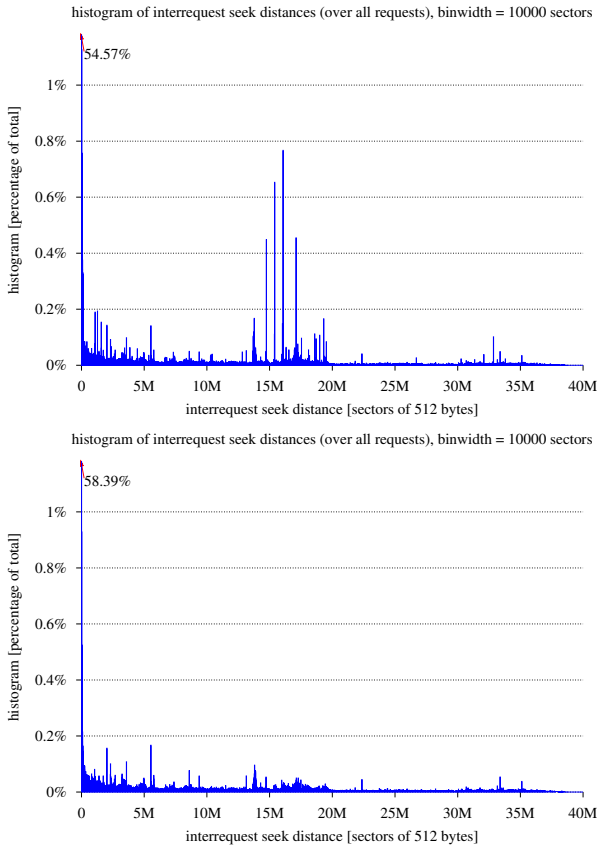

Figure 7: Distribution of seek distances in the complete CK04 trace (*top*) and the reduced version (*bottom*), where just 40 sectors have migrated from disk to flash. Note the difference in particular with respect to the peaks near the middle of the address space (i.e., about halfway the horizontal axis)

Table 3: Improvements when 40 of the most frequently accessed sectors are moved from disk to flash

parameter	old	new
no. of requests to disk [$\times 10^6$]	3.14	2.87 (− 8.6%)
no. of sequential requests [$\times 10^6$]	.824	.834 (+ 1.1%)
sequentiality [%]	26.2	29.0 (+10.7%)
average seek time on disk [ms]	3.97	3.73 (− 6.1%)
total seek energy [kJ]	28.7	24.6 (−14.2%)
total storage energy [kJ]	34.9	30.7 (−11.9%)

now remapped request.

In figure 7 we can see histograms of seek distances between requests to the disk. The peaks in the middle of the left histogram show, that, in the original situation, there are many seek operations that traverse about half of the address space, i.e., about half of the disk radius. Many of these seeks are to or from requests that lie in the narrow band of addresses that were remapped to the flash memory in our experiment, as we can conclude from the absence of most of these peaks from the right histogram. This saves a lot of seek energy (−14%) and consequently of total storage energy (−12%).

5.2.1 Reducing energy in an already enhanced system

Even after an optimisation of the operating system whereby file metadata is no longer as interruptive as in the CK04 trace, energy consumption can still be reduced. As indicated in section 3.2 such optimisation solutions are given in literature. For simplicity, suppose such a solution would effectively result in the removal of the 32 addresses that form the central peak in figure 6. We can then estimate the additional effect that our method would have by taking the reduced trace as a new reference.

From this reduced trace we select 40 of the remaining most frequently requested addresses and migrate those from disk to flash. The improvements in energy consumption compared to the metadata-reduced reference trace are given in table 4. Although the improvements are smaller than in the previous experiment, the storage system still uses less energy (6%) due to the migration of just 0.0001% of the address space from disk to flash. Thus, our method provides energy reduction in addition to improvements resulting from other methods.

Table 4: Improvements when 40 of the remaining most frequently accessed sectors are moved from disk to flash, after the central peak of figure 6 has been cleared by other means

parameter	old	new
no. of requests to disk [$\times 10^6$]	2.95	2.84 (−3.6%)
no. of sequential requests [$\times 10^6$]	.832	.834 (+0.3%)
sequentiality [%]	28.2	29.4 (+4.0%)
average seek time on disk [ms]	3.85	3.69 (−4.1%)
total seek energy [kJ]	26.1	24.2 (−7.6%)
total storage energy [kJ]	32.2	30.3 (−6.2%)

6 Dealing with wear issues

A well-known disadvantage of flash memory is the limited number of write operations before a flash cell wears out, typically in the order of 100,000-1,000,000 cycles. Its negative effect is normally minimised by wear levelling, which dynamically remaps the address space over the flash memory to spread the wear.

However, if the flash memory is small and the number of write operations high, then the expected lifetime will be short despite wear levelling. Taking our experiment of section 5.2 as an example we can predict the lifetime of the flash memory. Let us conservatively assume that a flash cell becomes unusable after 100,000 write operations. Figure 6 shows that the most frequently addressed LBAs are requested about 75,000 times over the trace duration of one month. Investigation of the trace reveals that most of these are actually write requests. So, if the flash memory would be just large enough to accommodate the migrated 20 KB, it would wear out soon after the first month. Therefore, we need extra space to allow for effective wear levelling.

So, suppose a lifetime of 10 years is required, then we can determine the amount of extra flash memory needed. Extrapolating the LBA count of the one-month trace, we can expect $120 \times 75,000 = 9,000,000$ write operations for the most frequently rewritten blocks over the course of 10 years. This is 90 times the maximum number of write cycles, so if we increase the size of the flash memory by a factor 90, there ought to be enough hardware redundancy for effective wear levelling. In other words, we will need 1.8 MB of flash memory to make the 20 KB migrated address space last for 10 years. This is still small (0.01% of the disk size) compared to the 12% energy reduction.

For larger migrated address spaces, the multiplication factor will be smaller, because the average number of write operations will decrease. (In fact, although we use the number of 75,000 above, this applies to only 16 out of the 40 addresses in our experiment; the average number of write cycles in our case is in fact about 40,000 per LBA, so that 1 MB of flash memory would already suffice for 10 years durability.)

However, the added cost for this hardware redundancy will cause the receiver operating curve of figure 3 to shift to the right. This may lead to a different trade-off point depending on the designer's requirements.

7 Conclusions and future work

Recalling our problem statement from section 2 – what are the relations between the size of the flash memory (relative to the total storage capacity), the energy consumption of the total hybrid storage device, and its cost? – we can draw the following conclusions.

Replacing a hard disk storage system by a hybrid storage system – effectively extending the hard disk storage system – with even the tiniest amount of flash memory, can already decrease the energy consumption of the storage device considerably (in our case, by 12%), at practically no extra cost, see section 5.2.

In general, when the amount of flash memory in the hybrid storage system is increased, more energy can be saved, although roughly every 10% reduction in the energy consumption requires a ten-fold increase in the size of the flash memory subsystem, as we have observed for flash memory sizes ranging from 0.1% to 10% of the total hybrid storage system capacity (section 5.1).

However, the associated additional cost increases accordingly. In most cases, a price of three times the original price is not acceptable in order to reduce seek energy consumption by one third. Therefore, in most cases a tradeoff is necessary. Depending on the weight of both decision factors – energy consumption reduction or cost – a point on the tradeoff graph has to be chosen (section 5.1.1, figure 3).

Also, it is important which part of the address space is mapped to the flash memory subsystem. Removing an arbitrary block of addresses may not lead to a reduction of energy consumption at all, as shown by table 2.

Addressing this problem, our study (section 5.1.2) presents two rules-of-thumb for choosing which part of the address space should be mapped onto the flash memory subsystem. (1) by looking at the sequentiality of the requests: the higher the sequentiality of the requests that remain on disk, the better the overall energy consumption. (2) by looking at the request frequency: the more requests are moved to flash, the lower the seek time and storage energy consumption.

Our solution allows extended flash lifetime by using a prediction of flash wear to determine the minimum size of the flash partition to extend its lifetime to some required minimum. This is illustrated by the example in section 6.

Finally, based on the experiment of section 5.2.1 we are confident that the trends of our results remain valid if additional optimisation algorithms are applied to the hybrid storage system or to either or both of its subsystems. Such additional optimisation algorithms could be burst mode disk access, improved scheduling, power management schemes (low power modes), or decision criteria that are more sophisticated than the simple *a posteriori* remapping of one chunk of the total address space to flash that we considered in this study. For instance, for practical applicability a runtime decision criterion would be needed. This will be the topic of further investigations.

7.1 Future work

We plan to validate our simulations in two ways: a hard validation and a soft one. The soft validation will itself be twofold as well, with improved energy models on the one hand and a wider variety of traces on the other.

Using improved energy models and a more detailed simulator, called DiskSim (Bucy et al. 2003), we hope to validate our simple energy model and simple simulator, thereby determining the accuracy of our results. We will then extend the number of experiments with new traces that we capture on a laptop and on a PDA, so that our results do not rely on a single-trace study.

The hard validation will be realised through the implementation of a daemon that will keep track of frequently used areas in LBA space and dynamically determine which LBAs to migrate to flash. The daemon will be tested in a conventional disk storage system and later be implemented in a hybrid storage system, where we can also measure the resulting energy consumption.

Along with those experiments, we will also test what the best location is for our method, i.e., where in the storage hierarchy do we implement the migration algorithm. Closely related to this is the development of good criteria to decide which frequently accessed blocks are expected to have the most impact when they are migrated.

Acknowledgments

We wish to thank Li-Pin Chang for sharing the CK04 trace with us and for detailing how he and his colleague Tei-Wei Kuo gathered the trace.

References

Bucy, J. S., Ganger, G. R. and Contributors (2003), The DiskSim simulation environment – version 3.0 reference manual, Technical Report CMU-CS-03-102, Carnegie Mellon University, Pittsburgh, PA, USA.

Chang, L.-P. and Kuo, T.-W. (2005), 'Efficient management for large-scale flash-memory storage systems with resource conservation', *ACM Transactions on Storage* **1**(4), 381–418.

Colarelli, D. and Grunwald, D. (2002), Massive arrays of idle disks for storage archives, *in* 'Supercomputing '02: Proceedings of the 2002 ACM/IEEE conference on Supercomputing', IEEE Computer Society Press, Los Alamitos, CA, USA, pp. 1–11.

Douglis, F., Cáceres, R., Kaashoek, M. F., Krishnan, P., Li, K., Marsh, B. and Tauber, J. A. (1994), Storage alternatives for mobile computers, *in* 'Operating Systems Design and Implementation (OSDI)', pp. 25–37.

Edel, N. K., Miller, E. L., Brandt, K. S. and Brandt, S. A. (2004), Measuring the compressibility of metadata and small files for disk/NVRAM hybrid storage systems, *in* 'Proceedings of the 2004 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS'04)'.

Ganger, G. R. and Kaashoek, M. F. (1997), Embedded inodes and explicit grouping: Exploiting disk bandwidth for small files, *in* 'Proceedings of the USENIX Annual Technical Conference', pp. 1–17.

Goldstein, H. (2006), 'Loser: Too little, too soon', *IEEE Spectrum* **43**(1), 28–29.

IBM Storage Technology Division (2000), *IBM Travelstar 32GH, 30GT, and 20GN 2.5-inch hard disk drives*.

Johnson, C. T. (1975), 'The IBM 3850: A mass storage system with disk characteristics', *Proceedings of the IEEE* **63**(8), 1166–1170.

Khatib, M., van der Zwaag, B., Hartel, P. and Smit, G. (2007*a*), Interposing flash between disk and dram to save energy for streaming workloads, *in* '5th IEEE Workshop on Embedded Systems for Real-Time Multimedia (ESTIMedia 2007)', Salzburg, Austria, pp. 7–12. `http://uspam.el.utwente.nl/w/images/8/8c/Flash_Buffer-ESTIMedia.pdf`.

Khatib, M., van der Zwaag, B., Hartel, P. and Smit, G. (2007*b*), Interposing flash between disk and DRAM to save energy for streaming workloads, Technical Report TR-CTIT-07-43, Centre for Telematics and Information Technology, University of Twente, Enschede, the Netherlands.

Kim, Y.-J., Kwon, K.-T. and Kim, J. (2006), Energy-efficient file placement techniques for heterogeneous mobile storage systems, *in* 'Proceedings of the 6th ACM & IEEE International Conference on Embedded Software (EMSOFT 2006)', pp. 171–177.

Lee, H. G. and Chang, N. (2005), 'Low-energy heterogeneous non-volatile memory systems for mobile systems', *Journal of Low Power Electronics* **1**(1), 52–62.

Lorch, J. R. and Smith, A. J. (1998), 'Software strategies for portable computer energy management', *IEEE Personal Communications Magazine* **5**(3), 60–73.

Marsh, B., Douglis, F. and Krishnan, P. (1994), Flash memory file caching for mobile computers, *in* 'Proceedings of the 27th Hawaii International Conference on System Sciences', Vol. 1, pp. 451–460.

Mullender, S. J. and Tanenbaum, A. S. (1984), 'Immediate files', *Software – Practice and Experience* **14**(4), 365–368.

Pinheiro, E. and Bianchini, R. (2004), Energy conservation techniques for disk array-based servers, *in* 'ICS'04', Malo, France, 26 June – 1 July.

Pinheiro, E., Weber, W.-D. and Barroso, L. A. (2007), Failure trends in a large disk drive population, *in* 'Proceedings of the 5th USENIX Conference on File and Storage Technologies (FAST'07)', pp. 17–28.

Ruemmler, C. and Wilkes, J. (1994), 'An introduction to disk drive modeling', *IEEE Computer* **27**(3), 17–29.

Stadlander, M. (2004), Hard disk power measurements, Master's thesis, University of Twente, Enschede, the Netherlands. `http://eprints.eemcs.utwente.nl/5281/`.

Thompson, D. and Best, J. (2000), 'The future of magnetic data storage technology', *IBM Journal of Research and Development* **44**(3), 311–322.

Zedlewski, J., Sobti, S., Garg, N., Zheng, F., Krishnamurthy, A. and Wang, R. (2003), Modeling hard-disk power consumption, *in* 'Proceedings of FAST '03: 2nd USENIX Conference on File and Storage Technologies', San Francisco, California, 31 March – 2 April, pp. 217–230.

Zheng, F., Garg, N., Sobti, S., Zhang, C., Joseph, R. E., Krishnamurthy, A. and Wang, R. Y. (2003), Considering the energy consumption of mobile storage alternatives, *in* 'Proceedings of the 11TH IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer Telecommunications Systems (MASCOTS'03)', pp. 36–45.

Appendix: Modelling seek energy

We use a simple model for the seek *distance* between two requests to the disk. Our model assumes that the physical seek distance $d_{\text{seek},n}$ from the last requested address of request $n-1$ to the first requested address of request n is linearly related with the number of addresses that lie between both requests:

$$d_{\text{seek},n} = \frac{|LBA_n - (LBA_{n-1} + size_{n-1})|}{\max(LBA)} \max(d_{\text{seek}}) \tag{1}$$

This means that for a sequential request m (that immediately follows request m-1 in the address space) $d_{\text{seek},m} = 0$.

According to Ruemmler and Wilkes (1994) "a seek is composed of

- a *speedup*, where the arm is accelerated until it reaches half of the seek distance or a fixed maximum velocity,
- a *coast* for long seeks, where the arm moves at its maximum velocity,
- a *slowdown*, where the arm is brought to rest close to the desired track, and
- a *settle*, where the disk controller adjusts the head to access the desired location."

Because we do not have a detailed model of the hard disk that was used to produce the CK04 trace, we cannot determine which sectors lie in the same track. For that reason we do not explicitly model the settle time, but implicitly model it as part of the total seek time by taking sectors rather than tracks as the unit for our model of the seek distance.

So, on the one hand, if two non-sequential requests are in the same track, our model ignores the fact that there would be no seek at all and reports a small seek time that is more than zero but less than the typical settle time. On the other hand, however, if these requests would be in neighbouring tracks, the seek time would be approximately equal to the settle time, yet our model reports a seek time that is smaller than that. We assume that both situations will more or less even out in our model.

Based on this and on the above model for the seek distance, we obtain a model of the seek time, i.e., the time $t_{\text{seek},n}$ it takes to perform a seek with a given seek distance $d_{\text{seek},n}$. For large seeks we assume that the arm holding the read/write head first accelerates to its maximum velocity $v_{\max}$ at time t_0, then keeps moving with constant velocity $v_{\max}$, and finally at

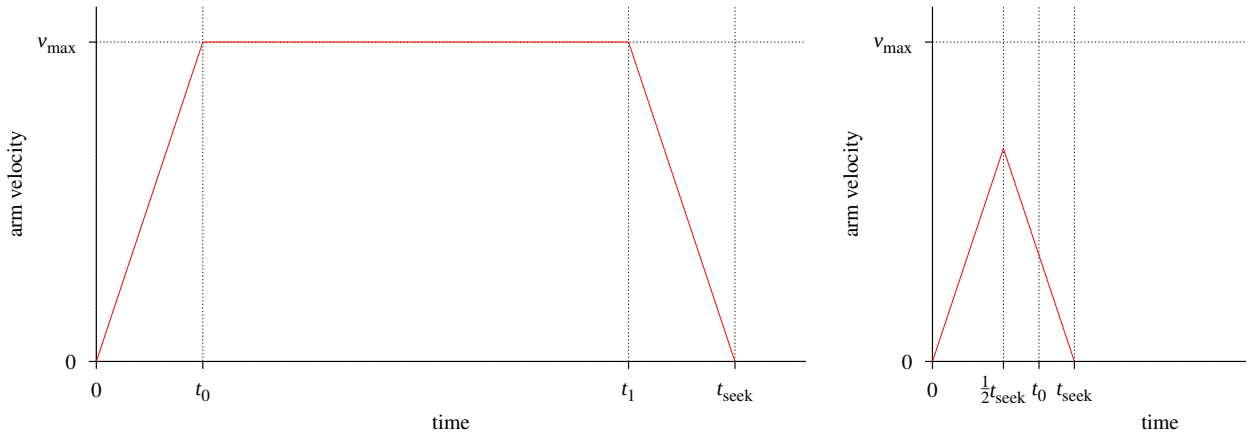

Figure 8: Model of the velocity of the arm of a disk during a seek operation for a large seek (*left*) and a small one (*right*)

time $t_1 = t_{\text{seek}} - t_0$ decelerates until it stops at the desired position at time t_{seek}, see figure 8 (left). We assume that maximum power is applied for accelerating/decelerating the arm, resulting in a constant angular acceleration a_{ang} during both the acceleration stage and the deceleration stage.

For small seeks, i.e., when $t_1 \leq t_0$, we assume that the arm holding the read/write head first accelerates until it is halfway at time $t_{\text{seek}}/2$, and then decelerates until it stops at the desired position at time t_{seek}. Thus, the arm does not reach its maximum velocity v_{max}, as also shown in figure 8 (right).

The velocity v of the read/write head, when it has not yet reached its maximum v_{max}, is given by

$$v(t) = r_{\text{arm}} a_{\text{ang}} t, \tag{2}$$

where r_{arm} is the distance from the arm's pivoting point to the read/write head, and a_{ang} is the (constant) angular acceleration.

The distance d that the read/write head travels during this time is then given by

$$d(t) = \int v(t)\mathrm{d}t = \frac{1}{2} r_{\text{arm}} a_{\text{ang}} t^2. \tag{3}$$

During the total seek time t_{seek} the read/write head will travel the total seek distance d_{seek}. This means for small seeks, where $t_{\text{seek}} \leq 2t_0$, the distance travelled during acceleration and deceleration, which is twice the distance travelled during acceleration:

$$\begin{aligned} d_{\text{seek}} &= 2d(\frac{t_{\text{seek}}}{2}) = 2 \cdot \frac{1}{2} r_{\text{arm}} a_{\text{ang}} \left(\frac{t_{\text{seek}}}{2}\right)^2 \\ &= \frac{1}{4} r_{\text{arm}} a_{\text{ang}} t_{\text{seek}}^2 \end{aligned} \tag{4}$$

and for large seeks, where $t_{\text{seek}} > 2t_0$, it is the sum of the distances travelled during acceleration until v_{max} is reached at t_0, during the time period between t_0 and $t_{\text{seek}} - t_0$ where the velocity v is constant (v_{max}), and during deceleration until the desired position is reached at time t_{seek}:

$$\begin{aligned} d_{\text{seek}} &= d(t_0) + v_{\text{max}}(t_{\text{seek}} - 2t_0) + d(t_0) \\ &= r_{\text{arm}} a_{\text{ang}} t_0^2 + v_{\text{max}}(t_{\text{seek}} - 2t_0), \end{aligned} \tag{5}$$

which, using equation 2, we can simplify to

$$\begin{aligned} d_{\text{seek}} &= v_{\text{max}} t_0 + v_{\text{max}}(t_{\text{seek}} - 2t_0) \\ &= v_{\text{max}}(t_{\text{seek}} - t_0). \end{aligned} \tag{6}$$

The model for the seek time t_{seek} as a function of the seek distance d_{seek} can now be found by inverting the above equations to:

$$t_{\text{seek}} = \begin{cases} 2\sqrt{\dfrac{d_{\text{seek}}}{r_{\text{arm}} a_{\text{ang}}}} & \text{for } d_{\text{seek}} \leq d_0 \\ \dfrac{d_{\text{seek}}}{v_{\text{max}}} + t_0 & \text{for } d_{\text{seek}} > d_0 \end{cases}, \tag{7}$$

where d_0 is the distance needed for the arm to reach its maximum velocity v_{max}:

$$d_0 = d(t_0) = \frac{1}{2} r_{\text{arm}} a_{\text{ang}} t_0^2 = \frac{v_{\text{max}}^2}{2 r_{\text{arm}} a_{\text{ang}}}. \tag{8}$$

Finally, because the energy needed to accelerate/decelerate the arm is much less than the energy consumed by the spinning disk, our model for the seek energy E_{seek} assumes a linear relationship with

the seek time t_{seek}, based on the power consumption P_{seek} of the spinning disk during a seek operation:

$$E_{\text{seek}} = P_{\text{seek}} t_{\text{seek}}, \tag{9}$$

which, together with the parameters listed in table 1, results in the seek energy profile presented in figure 1.