JAIST Repository
https://dspace.jaist.ac.jp/

Title	Comparative analysis of overall energy consumption of storage architectures in home media players
Author(s)	ウィ, クリスティアント
Citation	
Issue Date	2011-09
Type	Thesis or Dissertation
Text version	author
URL	http://hdl.handle.net/10119/9928
Rights	
Description	Supervisor: Associate Professor Xavier Defago, 情報科学研究科, 修士

JAIST
JAPAN
ADVANCED INSTITUTE OF
SCIENCE AND TECHNOLOGY

Japan Advanced Institute of Science and Technology

Comparative analysis of overall energy consumption of storage architectures in home media players

By Christianto Oeij

A thesis submitted to
School of Information Science,
Japan Advanced Institute of Science and Technology,
in partial fulfillment of the requirements
for the degree of
Master of Information Science
Graduate Program in Information Science

Written under the direction of
Associate Professor Xavier Défago

September, 2011

Comparative analysis of overall energy consumption of storage architectures in home media players

By Christianto Oeij (9210202)

A thesis submitted to
School of Information Science,
Japan Advanced Institute of Science and Technology,
in partial fulfillment of the requirements
for the degree of
Master of Information Science
Graduate Program in Information Science

Written under the direction of
Associate Professor Xavier Défago

and approved by
Associate Professor Xavier Défago
Professor Mizuhito Ogawa
Associate Professor Kiyofumi Tanaka

August, 2011 (Submitted)

Copyright © 2011 by Christianto Oeij

Abstract

In home entertainment, reducing energy is highly desirable. Our motivations stem from the important factors like the running cost, noise, heat, and energy consumption from the storage as the main component of it. Home theater PC is a good choice of home entertainment since it can provide high quality video output and be affordable in term of cost. We investigate the total energy consumption in systems built around a Home theater PC. There are several architecture choices that could potentially play a significant role on total energy consumption. In particular, we identify three choices:

- **HDD vs. SSD**: HDD can provide bigger capacity with affordable price. Even though, SSD is much more expensive, it is better in speed performance with less energy consumption compared to HDD. In term of energy consumption, is it true in all cases?

- **Local Storage vs. Network-Attached Storage (NAS)**: Local storage might be better in media playback performance and energy efficiency compared to NAS. However, NAS can provide benefits like centralized data storage and no extra space wasted for duplicate files.

- **Prefetching**: Typical Home theater PC might have big size of system memory installed for the cheap price of RAM memory. It is an opportunity for energy savings by applying *aggressive prefetching* to extend the disk idle time for saving energy.

We evaluated experimentally three different storage architectures by measuring important aspects, namely, energy consumption, hardware cost, playback performance, and maintenance effort. To do so, we have built and instrumented a benchmark environment, as well as a microcontroller-based device to monitor and record energy consumption directly from power cables.

We considered important key factors like the characteristics of storage medium, system memory size, prefetching size as well as the workload parameters like the size of multimedia data, duration of playback and also application parameters of the system application to explore the possibility of better energy efficiency and performance.

Interestingly, we have found that by applying aggressive prefetching, the energy consumption of HDD can be reduced to the same levels as SSD for video playback. This means that, with appropriate system support in the media player, SSD is not yet ready to replace HDD, even on the ground of energy-efficiency. For NAS, we found another interesting result that using HDD as local storage will consume less energy compared to using SSD. Based on the analysis of the experimental results, we identified the energy and cost efficient storage architectures in home media players.

Keywords: Storage, energy efficiency, prefetching, embedded system, home media players

Acknowledgements

Being able to come to Japan and study at JAIST is very valuable life experience for me. I have been lucky and honorable to be a member of Dependable Distributed System Group (DDSG) lab in School of Information Science at JAIST. I really appreciated my supervisor, Professor Xavier Défago, to let me join his lab from 2009. I am very grateful to my supervisor for advising and guiding me throughout my research. During my research, I often got a lot of encouragement and inspiration from him. Also, I enjoyed the discussions with him because he has incredible source of knowledge and experiences. I would also like to thank him for the time to patiently guide and advise me many times during my research. Thank you for the opportunity and guidance for me to have my research accepted at APSYS 2011 and be able to give poster presentation in Shanghai. Furthermore, I really enjoyed the atmosphere of the lab, which provides motivating research environment.

I would also like to thank my seniors, Daiki Higashihara and Shintaro Hosoai for assisting me not only in study but also daily lives need. With them, I enjoyed the discussion and conversation about the research, culture difference, etc.

Beside that, I also appreciate the help of François Bonnet, the post-doc researcher in our lab for all his advices, encouragement and help for my study.

I am also very glad to have all the friends or colleagues now for their care to me.

I would like to express my gratitude to Rotary Yoneyama Memorial Foundation who supports me the monthly scholarship from April, 2010 up to September, 2011. Thank you for also letting me to understand the Japanese culture through the meetings.

Lastly, I would like to thank my parents and dedicate all my works to them. They always support me throughout my life and are always my motivation to do my best in my life. And also to my brother and sister who always cheers me up in difficulties. I love you all.

— JAIST School of Information Science

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Nowadays, people spend more time on consuming media like television, movies, music, games and browsing the Internet than the past. Because of the high availability and faster access to media, we can enjoy entertainment more.

Starting from the earliest media player like traditional televisions, radios, tape players, nowadays we can have one media player with many functionality like playing music, movie, radio, taking picture, etc. Also, compared to the past times, the price of media players have become cheaper mainly for the decreased price of the hardware. The media players can be the portable ones like portable audio or video players as well as non-portable ones like the entertainment units we have in our homes. Currently, media player has even become like one part of our daily needs, which means that almost everyone has their own media player.

Figure 1.1: Home theater

1.1 Home media players

In home, people also need entertainment, which can be realized by having home media players. For home media players, many young and old people have embraced the use of home theatre pc for several reasons.

It is basically an old computer, which is used to provide entertainment functionality like watching movies and playing music. When connected to other devices like display and speakers, it can provide high quality of movie playback with nice sounds.

Another reason is that it is easier to upgrade to match any development that occurs. It is different with other dedicated service gadgets, which are more difficult to upgrade. In most cases, we need to replace the old ones with the new ones. Beside that, we can customized it to provide more functionality according to our needs like internet connection, radio or games since it is basically a PC machine.

1.2 Storage architectures

For home media players, the most important asset is the media data, which is stored inside it. Thus, storage plays the important role in home media player like home theatre PC. Before people decided to buy or build a home media player, there are several factors they would consider such as the cost, data storing capacity and maintenance complexity. Storage, as the main component of it, would directly relate to the consideration points mentioned.

For storage, we may have different options like local and network storage. For the local storage, the common options are HDD (Hard Disk Drive) and SSD (Solid State Drive). However, each of them offers different pros and cons. And for the network storage, people use NAS (Network-Attached Storage) as the architecture option. In fact, there are still more storage architecture options like CD/DVD and flash memory.

1.3 Energy consumption

Reducing energy consumption has always been an important issue. The rising cost of energy and increased public awareness surrounding the environment and sustainability has prompted even more attention to reducing the power consumption of household.

Concerning the important role of entertainment takes in our daily lives nowadays; we also need to consider the energy efficiency of entertainment media equipment, which we use in home. For home entertainment, home media players would be the main equipment, which consumes much energy compared to the others. Therefore, getting to know how much energy is being consumed and trying to reduce the energy consumption is essential especially to reduce part of house energy consumption cost.

1.4 Objectives

Our main (principle) motivation is to consider the energy consumption of storage architectures in home media players. In particular, home media player is commonly used for watching high quality movies, which will take bigger capacity size.

Our work considers mainly the energy consumption during video playback of home media players. In the experiment, we analyze the energy consumption by using different storage architectures commonly used. We also consider the overall energy consumption for the NAS (Network-Attached Storage) architecture in home media player.

Our research goal is to provide comparative analysis of total energy consumption of storage architectures in home media players. As an important factor in reducing energy consumption, we also include prefetching as one of the architecture choices in the experiment.

1.5 Contributions

The contribution in this dissertation is the investigation of energy consumption of storage architectures in home media players with the comparative analysis. We showed that HDD is not less energy-efficient than SSD for media application.

1.6 Organization

The rest of the dissertation is structured as follows:

- **Chapter 2** discusses the background of home media players, storage architectures and also the operating system used in home media players.
- **Chapter 3** states the problem that needs to be addressed in home media players.
- **Chapter 4** discusses prefetching and its effect in term of storage energy efficiency.
- **Chapter 5** states the approach of comparative analysis methodology used throughout this dissertation.
- **Chapter 6** presents the results of the experiments with the analysis.
- **Chapter 7** summarizes the major results of this work and outline future work directions.

Chapter 2

Home Media Players and Storage Architectures

With the growing entertainment consumption nowadays, the usage of home media players has also increased. Home entertainment has become a part of daily needs. Even in home, people would also like to have entertainment like movies, music, games, etc. Home media players could offer the entertainment as wanted. However, there are various kinds of home media players and architecture options to consider. Each one of them might offer different cost/capacity tradeoffs.

2.1 Home media players

Commonly known home entertainment is home cinema. The purpose of home cinema is to provide home theatre, which are home entertainment set-ups that seek to reproduce movie theater video and audio feeling in a private home. For the set-ups, it certainly must include home media players to provide the service needed. Currently, there are many options in building home cinema like buying expensive high quality equipment. Among all these options, there is actually a great home entertainment options, which is by using our old PC as home theatre.

2.1.1 HTPC

A Home Theatre PC (HTPC) is a convergence device that combines some or all the capabilities of a personal computer with a software application that supports video, photo, and music playback, and sometimes, digital video recorder and time shifting functionality. It integrates many or all components of a home theater into a single unit co-located with a home entertainment system. Since it is designed for a home entertainment, it typically has a remote control and bigger user interface design so that it can be viewed from certain distance from the display. An HTPC can either be purchased pre-configured with the required hardware and software needed, or can be combined together from components essential to build a HTPC.

Stored media is kept either on a local hard drive or on a network attached storage. Some software is capable of doing other tasks, such as finding news or checking weather forecast from the Internet.

2.2 Storage architectures

Deciding the storage architectures in home media players is essential since it directly relates to the cost, capacity and maintenance complexity. Of course, the ideal architecture is that we can bigger capacity with inexpensive cost and simple maintenance effort. But in reality, different storage architectures offer different tradeoffs. In this research, we consider local storage as well as network-attached storage and analyze the energy consumption as the main metric.

2.2.1 Local Storage

With local storage, we can simply manage all the media files (such as video files) directly into the local storage. It does not need complex management. Obviously, for media players, bigger storage capacity is preferred. Currently, we have two options for local storage such as HDD (Hard Disk Drive) and SSD (Solid State Drive). With HDD, we can get bigger capacity with cheaper price if we compared it with SSD. But in term of energy efficiency, SSD performs better than HDD. In this experiment, we analyze the energy consumption of HDD and SSD during video playback in home media players.

2.2.2 Network-Attached Storage

With Network-Attached Storage, we can get another advantages like centralized data management, which might offer efficiency in term of space usage. In home media players, when there are several users, there is a possibility that they might have the same media files. By putting all the media files into the same storage in NAS, it will eliminate the situation when there is the same file, which is copied in different storage such as in local storage. However, with Network-Attached Storage, the maintenance is becoming more complicated since we also need to consider the networking part like switch and NAS maintenance. For the storage option, it is much reasonable to use HDD since it provides bigger capacity with cheaper cost.

2.3 Operating System and Applications

Inside home media players like HTPC, we need software to run it. HTPC options exist for each of the major operating systems: Microsoft Windows, Mac OS X and Linux. The software is sometimes called “Media Center Software”. Beside that, there also exist some media center oriented embedded systems.

2.3.1 Standard OS for media center

For GNU/Linux, we can directly use the existing Linux OS like Ubuntu, Fedora, RedHat, Knoppix, etc. And there are also few options of customized linux-based OS like MythUbuntu, KnoppMyth, SageTV and Boxee.

For MAC OS X, some HTPC functionality is built into the operating system itself. Specifically, the programs Front Row and Cover Flow, utilized in conjunction with the Apple Remote, let users easily browse through and enjoy any multimedia content stored in their Macs.

For Microsoft Windows, a common approach is to install a version that contains the Windows Media Center (Home Premium, Professional or Ultimate for Windows 7, Home Premium or Ultimate for Windows Vista, or the older Windows XP Media Center Edition). Windows Media Center includes additional software that covers the PVR functions of the proposed HTPC, including the free program guide information and automatic program recording. Windows 7, Windows Vista Home Premium and Windows Vista Ultimate already include an MPEG2 decoder. Only Windows XP MCE does not provide an MPEG2 codec, that can be purchased from Intel, or is alternatively included in Intervideo's WinDVD shareware, or with DVD Decoder packages such as Nvidia's PureVideo and Sonic's CinePlayer.

2.3.2 Embedded OS for media center

Instead of using standard OS like Windows, MAC OS X or Linux, there are available options like media center oriented embedded systems. It will provide smaller size of operating system, which means that we can have empty space for storing data. There are some available free media center embedded operating systems such as GeeXboX [3], OpenELEC and Element OS. They are all Linux based distributions.

In this research, we are interested in using Linux based embedded OS in the experiment because of two main reasons. The first one is that it is open source so that we can analyze the code and script inside for further optimization. The second reason is that the size of the operating system is much smaller, so that users can put more media files inside the storage as compared if they use the standard OS.

Chapter 3

Problem Statement: Energy Consumption and Capacity

In home media players, people prefer big storage capacity with inexpensive price. Having high definition movies inside the home media players are preferred. Nevertheless, it also means that we need for bigger space for store the movies. As an example, the size of one complete high definition movie needs more than 4GB while the size of lower definition movie might need only about 1.5 GB. Even for higher quality lossless (BluRay) movie, it might need about 13GB to 40GB for only one file. This tells us the importance of bigger capacity of storage in home media players.

Also, the energy efficiency of the appliances is preferred. Since storage is the main component of home media players, we would like to analyze the energy consumption of it. As we know that SSD offer better energy efficiency than HDD, at first it seems attractive to just choose SSD as the storage architecture options. But in fact, current price of SSD is still way too high if we compared it to the price of HDD.

Therefore, it seems that we might not be able to have all the ideal conditions since different storage architectures would offer different tradeoffs. In this research, we would like to analyze the energy consumption of storage architectures and explore the way to optimize the energy efficiency in home media players.

3.1 HDD

A hard disk drive (HDD) is a non-volatile, random access digital data storage device. It features rotating rigid platters on a motor-driven spindle within a protective enclosure. Data is magnetically read from and written to the platter by read/write heads that float on a film of air above the platters.

Many of the hard drive companies are now producing Green Drives that require much less power and cooling. Many of these Green Drives spin slower (less than 5,400 rpm compared to 7,200, 10,000 or 15,000 rpm), thereby generating less heat. Parking the drive heads when the disk is not in use by reducing friction, adjusting spin speeds, and disabling internal components when not in use can also reduce power consumption. Most

hard disk drives today support some form of power management, which uses a number of specific power modes that save energy by reducing performance. When implemented an HDD will change between a full power mode to one or more power saving modes as a function of drive usage. Recovery from the deepest mode, typically called Sleep, may take as long as several seconds.

Several works have been published for low power HDD. In [11], it is reported that the mechanical parts incur large overheads of power consumption, especially when the HDD starts to run the spindle and head. Also, the spin-up energy can vary between HDDs by an order of magnitude.

Several methods for HDD power management have been presented. In [14], the authors present the quantitative comparison of existing dynamic power management policies (DPM) used for shutting down hard drive into low power states when there is no I/O access. The inter-session delay is predicted and exploited to make HDD enter into low power states. In [9], a time-out based DPM policy is presented. In this work, if there is no new access during the time-out, a low power state is entered. The time-out is determined adaptively based on the accuracy of previous time-out predictions. In [8], the authors present a machine learning-based method to determine the best policy among a set of DPM policies by adapting changes in the system workloads. In [7], the DPM scheme based on idle period clustering and adaptive learning trees are presented. The other method presented for the Dynamic Power Management is the Program Counter-Based Prediction Techniques in [10] which dynamically learns the access patterns of applications and predicts when an I/O device can be shut down to save energy.

There are several studies to HDD idle time proactively in order to save energy. In [15], prefetching is applied to prolong the idle time of HDDs, enabling them to stay in low power states for longer periods. The other method to make the HDD idle time longer is by providing power-aware cache management as presented in [4, 6, 13, 19].

There is also another disk power consumption reduction technique which was done by having the disk drives to enter acoustic modes which reduces the instantaneous power consumption presented in [5].

3.2 SSD

A solid-state drive (SSD) is a data storage device that uses solid-state memory to store persistent data with the intention of providing access in the same manner of a traditional block I/O hard disk drive. SSDs are distinguished from traditional hard disk drives (HDDs), which are electromechanical devices containing spinning disks and movable read/write heads. SSDs, in contrast, use microchips, which retain data in non-volatile memory chips and contain no moving parts. Compared to electromechanical HDDs, SSDs are typically less susceptible to physical shock, are silent, and have lower access time and latency, but are more expensive per gigabyte (GB) and typically support a limited number of writes over the life of the device. SSDs use the same interface as hard disk drives, thus easily replacing them in most applications.

Flash memory has the advantages of high performance, low power consumption, and

high reliability compared with a conventional hard disk drive (HDD). High performance is the main reason that flash memory is adopted, especially for faster booting and application loading. The advantages come from the fact that flash memory is based on electronic functions, e.g., program, read and erase, whereas an HDD is based on mechanical ones, e.g., servo and spindle motors and arms. Driving those mechanical parts entails significant latency (on the order of milliseconds) and power consumption, especially when the HDD enters the active mode from a low power mode.

Among flash-based storage, SSD is becoming a major storage device, replacing the HDD in smart phones (e.g. iPhone 3GS with 32 GB flash memory) and net books as well as notebook PCs and servers.

One of the main reasons that SSD is favored over HDD is performance. The SSD offers higher performance than the HDD via parallel accesses; it utilizes relatively low speed flash devices in parallel. For instance, in order to achieve a throughput higher than 240 MB/s, we can utilize 8 flash devices with 33 MBps each in parallel. Recently, in order to obtain further performance improvement, high speed flash interface specifications have been presented, e.g. ONFI and toggle NAND. By adopting flash memories with high I/O bandwidth, the SSD performance can be improved significantly.

However, the price of SSD is still very expensive comparing to the price of HDD. Concerning to this, people still prefer HDD to have big capacity of storage like the multimedia storage.

3.3 Network-Attached Storage

A NAS unit is a computer connected to a network that only provides file-based data storage services to other devices on the network. NAS uses file-based protocols such as NFS (popular on UNIX systems), SMB/CIFS (Server Message Block/Common Internet File System) (used with MS Windows systems), or AFP (used with Apple Macintosh computers). NAS units rarely limit clients to a single protocol. Some advantages of NAS are:

- Users running different types of machines (PC, Apple iMac, etc.) and running different types of operating systems (Windows, Unix, Mac OS, etc.) can share files.
- NAS appliances are “plug-and-play” meaning that very little installation and configuration is required beyond connecting them to the LAN.
- Less administration overhead than that required for a Unix or NT file server.
- Centralized storage, which makes the data easier to manage and share. Incidentally, centralized storage is more expensive than local disks on byte cost basis, but users have to do tasks such as backups and restores on their own.

However, NAS might require much more power consumption since it needs to work on LANs. Also, for home users, it might require more maintenance effort for the NAS and the networking parts. For the energy consumption side, it means additional consumption

at the NAS server itself and also the networking switch/hub in the case of home media players.

Chapter 4

Prefetching

In prefetching, the operating system tries to predict the pages a process will need and to preload them when memory space is available. If the system is able to make correct decisions about future page use, the process's total runtime can be reduced.

4.1 Prefetching in Linux

Many disk accesses are sequential. Regular files are stored in disk in large groups of adjacent sectors, so that they can be retrieved quickly with few moves of the disk heads. When a program reads or copies a file, it often accesses it sequentially, from the first byte to the last one. Therefore, many adjacent sectors on disk are likely to be fetched when handling a series of a process's read requests on the same file.

Prefetching in Linux (or commonly called Read-ahead) consists of reading several adjacent pages of data of a regular file or block device file before they are actually requested. In most cases, read-ahead significantly enhances disk performance, because it lets the disk controller handle fewer commands, each of which refers to a larger chunk of adjacent sectors. Moreover, it improves system responsiveness. A process that is sequentially reading a file does not usually have to wait for the requested data because it is already available in RAM.

However, read-ahead is of no use when an application performs random accesses to files; in this case, it is actually detrimental because it tends to waste space in the page cache with useless information. Therefore, the kernel reduces or stops read-ahead when it determines that the most recently issued I/O access is not sequential to the previous one.

The readahead inside Linux 2.6 adopts dual windows to achieve readahead pipelining: while the application is walking in the `current_window`, I/O is underway in the `ahead_window`. For the purpose of pipelining, the I/O is issued for the next readahead before the not-yet-consumed readahead pages fall under a threshold, lookahead size. A value of `lookahead_size` = 0 disables pipelining, whereas `lookahead_size` = `readahead_size` opens full pipelining.

We put the source code of on-demand readahead algorithm[17, 18] in Linux 2.6 inside

Appendix A. It is composed of a list of condition-action blocks. Each condition tests for a specific case (Table 4.1), and most actions merely fill the readahead state with proper values.

case	description	condition
initial	read on start of file	`!offset`
oversize	random oversize read	`!page && !sequential && size > max`
random	random read	`!page && !sequential`
lookahead	lookahead hit	`offset == ra->lookahead_index`
readahead	readahead hit	`offset == ra->readahead_index`
miss	sequential cache miss	`!page`
interleaved	lookahead hit with no context	`page`

Table 4.1: Cases for detecting access patterns

The cases considered in the algorithm:

- **Random:** A small, stand-alone read. Take it as a random read, and read as is.
- **Lookahead:** It is readahead time indicated by the readahead state, so ramp up the size quickly and do the next readahead.
- **Readahead:** It is readahead time indicated by the readahead state. We can reach here if the lookahead mark was somehow ignored (queue congestion) or skipped (sparse read). Do the same readahead as in lookahead time.
- **Initial:** First read on start of file. It may be accessing the whole file, so start readahead.
- **Oversize:** An oversize read. It can not be submitted in one huge I/O, so do it progressively as a readahead sequence.
- **Miss:** A sequential cache miss. Start readahead.
- **Interleaved:** A lookahead hit without a supporting readahead state. It can be some interleaved sequential streams that keep invalidating each other's read-ahead state. The lookahead page indicates that the new readahead will be at least the second one in the readahead sequence. So get the initial readahead size and ramp it up once.

4.2 Energy-efficient Prefetching

Prefetching and caching are standard practice in modern file systems. They serve to improve performance that is 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. This strategy results in relatively short intervals of inactivity. It 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 states 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 energy efficient prefetching [15] is to create bursty access patterns for devices with non-operational low-power states, increasing the average length of idle intervals and maximizing utilization when the device is active, without compromising performance.

Typical hard disks support at least four power states: Active, Idle, Standby, and Sleep. The disk only works in the Active state. In the Idle state the disk is still spinning, but the electronics may be partially unpowered, and the heads may be parked or unloaded. In the Standby state, the disk is spun down. The Sleep state powers off all remaining electronics; a hard resets is required to return to higher states. Individual devices may support additional states. The IBM TravelStar, for example, has three different idle sub-states.

One to three seconds are typically required to transition from Standby to Active state. During that spin-up time, the disk consume 1.5-2X as much power as it does when Active. The typical laptop disk must therefore remain in Standby state for a significant amount of time on the order of 5-16 seconds for current laptop disks to justify the energy cost of the subsequent spin-up. The energy savings in idle state approaches that of Standby state, particularly in very small form factor devices, and the time and energy to move from Idle to Active state are minimal. Hence, even modest increases of the disk's idle interval can lead to significant energy savings.

In order to understand how prefetching can help reduce the energy consumption of hard drive, we show the concept of power management first in Figure 4.1.

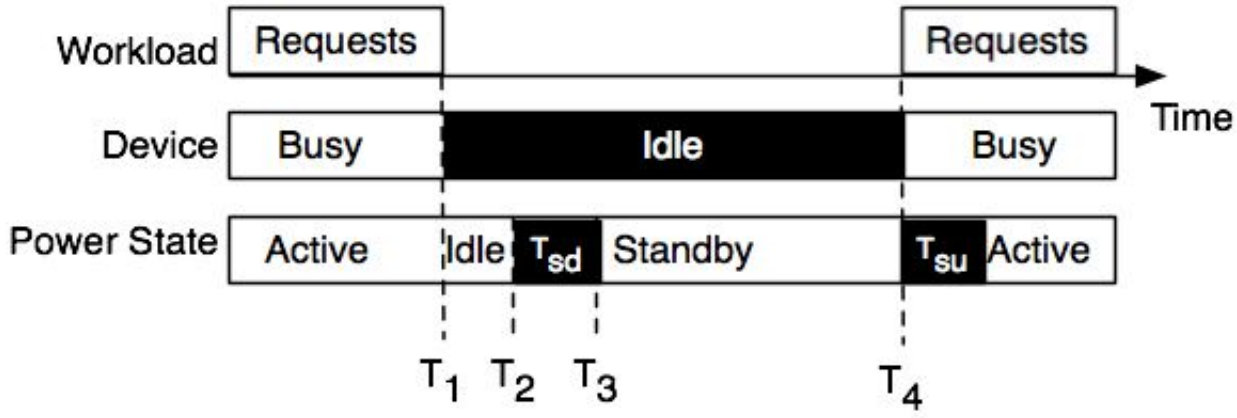

Figure 4.1: Power management from the workload, device, and power state points of view

A workload consists of many requests. The requests are read and write commands. When there are requests, the device is busy. Otherwise, it is idle. When hard drive is busy serving for requests, its power state is active. If there is no request and the device is idle, then it will enter idle mode. After certain period of time (T_1 to T_2), the disk will enter Standby mode. To enter standby mode, disk needs to spin down and it needs T_{sd} time. When there is new request again, disk will spin up and for this, it need T_{su} time. As mentioned above that lower power state consumes less power, the longer the disk stays in standby mode, the less power being consumed.

However, the standby period (T_2 to T_4) needs to be long enough to compensate for the overhead of power state changes (spin up and spin down). The minimum length of the period to save power is called the break-even time (T_{be}). It depends on individual devices and is independent of requests. Suppose its power in the active and standby states is P_a and P_s. On the left of Figure 4.2, the device is kept in the working state; on the right, the device is shut down. The break-even time has to be larger than the transition delay. Namely, $P_a \times T_{be} = (E_{sd} + E_{su}) + P_a \times (T_{be}$ - $(T_{sd} + T_{su}))$. Therefore, $T_{be} = max[(E_{sd} + E_{su}) - P_s \times (T_{sd} + T_{su})/(P_a$ - $P_s), T_{sd} + T_{su}]$.

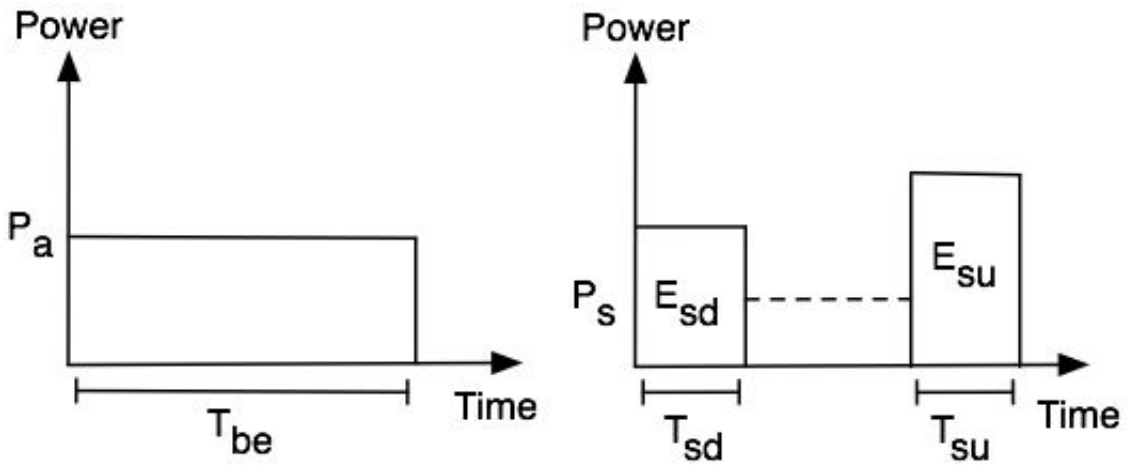

Figure 4.2: Breakeven time

In order to reduce energy consumption, the standby period needs to be longer than the breakeven time. Energy-efficient prefetching does this by making the workload access during the busy time of the device to be bursty so that all the needed request data available in the page cache and the device can be put into standby mode for longer time.

4.3 More Energy-savings with Aggressive Prefetching

Traditional OS-level prefetching strategies have tended to be conservative, fetching only those data that are likely to be needed according to some simple heuristic, and only just in time for them to arrive before the first access. More aggressive policies which might speculate more about which data to fetch, or fetch them earlier in time have typically not been considered a prudent use of computational, memory, or bandwidth resources. However, the technological trends and emerging system design goals have dramatically reduced the potential costs and dramatically increased the potential benefits of highly aggressive prefetching policies.

Published studies have shown that aggressive prefetching [16] has the potential to improve I/O performance for a variety of workloads and computing environments, either by eliminating demand misses on pages that a conservative system would not prefetch, or by avoiding long delays when device response times are irregular. Most modern operating systems, however, still rely on variants on the standard, conservative sequential read-ahead policy. Linux, for example, despite its reputation for quick adoption of promising research ideas, prefetches only when sequential access is detected, and by default to a maximum of only 128KB.

Technology and market forces have led to dramatic improvements in processing power, storage capacity, and to a lesser extent I/O bandwidth. One of the obvious improvements is the cheaper price for large size memory. As memory sizes and disk bandwidths continue to increase, and as multimedia applications continue to proliferate, the performance benefit of aggressive prefetching will surpass that of caching policies.

Chapter 5

Comparative analysis methodology

After all the required background and previous works was presented in previous chapters, this chapter explains the methodology used for the comparative analysis. The storage architecture options of home media player that we consider in this experiment are:

- HDD versus SSD
- Local storage versus Network-Attached Storage (NAS)
- Prefetching

We applied the systemic approach[12] for the comparative analysis by firstly defining the project plan before starting the study. The methodology steps are defined in detail in the sections following.

5.1 Systemic Approach to Evaluation

The study is based on the following plan:

1. **System definition:** The goal of the study is to compare the overall energy consumption of home media players using different storage architectures. The key component under the study is storage architectures. The storage architectures are HDD, SSD or NAS. However, comparing directly the energy consumption of local storage and NAS is not fair, thus we measure the overall energy consumption of NAS and make the comments of the comparison between local storage and NAS later. The system consists of home media player with storage medium. All the media like movies and music are stored in the storage medium. Only the subsets of the media player system that offer media playback services is considered to be part of the system. The study will be conducted so that the effect of components outside the system is minimized.
2. **Services:** The services offered by the system are to provide video playback with different storage architectures. The resources used by home media player system

depend upon the storage medium and system parameters being used. In this case study, media playback is chosen as the application and the media will be classified by the size (small, large) and quality (low, medium, high) depending upon the type of data being played by the system. In other words, the system offers services: small sized video (with low, medium, high quality) playback and large sized video (with low, medium, high quality) playback.

3. **Metrics:** The study will be limited to correct operation only. For each service, we observed the mainly the energy consumption of the system. The resources are the media player system and the storage medium. This leads to the following performance metrics:

   - Energy consumption of the storage medium per video playback.

4. **Parameters:** The system parameters that affect the performance and energy consumption of the given application and data size are the following:

   - Type of storage architecture
   - Speed of CPU
   - Size of memory
   - Speed of storage data transfer
   - Speed of network (for NAS architecture)
   - Size of prefetching
   - Correctness of predicted future data of prefetching
   - Operating system overhead for interfacing with the application
   - Operating system overhead for interfacing with the networks
   - Reliability of the network

   The workload parameters that affect the main metric, energy consumption, are the following:

   - Number and sizes of movie file
   - Playback duration of the movie
   - Type of media player system
   - Parameters setting of the playback application

5. **Factors:** The key factors chosen for this study are the following:

   - Type of storage architectures. For local storage, two types, HDD, SSD are compared. For NAS, two types, HDD as local storage with NAS and SSD as local storage are compared.
   - Number and sizes of the data. One big high definition movie file will be used.

- Size of prefetching. Different values of system maximum prefetching size will be used. Two values – default (128KB) and large (512MB).
- Size of memory. System memory size: 884 MB.

6. **Evaluation technique:** Since the media player, system and workload are ready to be used, measurements will be used for the evaluation. Analytical modeling will be used to justify the consistency of measured values for different parameters.
7. **Workload:** The workload will consist of media player application playing video from different storage architectures. The disk power measurement device will monitor the power measured and record the measured results.
8. **Experimental design:** We used full factorial experimental design was used for the initial study and then apply this experimental design after repeated number of experiments after considering the impact and time needed for completing each experiment as shown in Table 5.1 and 5.2

Factor	**Levels**
Storage	HDD, SSD
Data size	Big file size (4.7 GB)
Prefetching	Default size (128KB), Big size (512MB)
Memory	System memory size (884MB)

Table 5.1: Experimental design for local storage

Factor	**Levels**
Storage	HDD as local with NAS, SSD as local with NAS
Data size	Big file size (4.7 GB)
Prefetching	Default size (128KB)
Memory	System memory size (884MB)

Table 5.2: Experimental design for NAS

9. **Data analysis:** Analysis of variance will be used to quantify the effects of the factors.
10. **Data presentation:** The final results will be plotted as a function of power consumption in Watts.

5.2 Monitors

In this experiment, the main metric is the energy consumption. The monitor used in the experiment should not interfere with the system to make sure that the correctness of the result obtained.

For measuring the power consumption, we measure directly from the power supply lines of the disk drives. In order to achieve it, we have built the measurement device to measure directly from the power cables and record all the collected data into a memory card.

5.2.1 Disk Power Measurement Device

We have developed a custom measurement device that provides direct, online measurements of disk drive power consumption. We configured our device to capture two inputs; the 5-Volt and 12-Volt supply lines of a connected drive at 12.5 Hz. The measurement device was built by using a microcontroller, Arduino [2] (board model: UNO) and two current sensors (ACS712). Each current sensor will capture the current value from the supply lines and input them into the microcontroller. The microcontroller will process the data and record them into a MicroSD card as shown in Figure 5.2.

The frequency of sample collection is about 12.5 Hz. Inside the current sensor, 1.2 m Ω internal conductor resistance is used to intercept with the supply lines, leading to a small, but detectable, drop in potential across these resistors. Our measurement instrumentation is powered separately and has a negligible effect on the actual power consumption of the drive under test. Pictures of the measurement setup and also the electronic diagram are provided in the Figure 5.1 and Figure 5.3, respectively.

Figure 5.1: Measurement setup

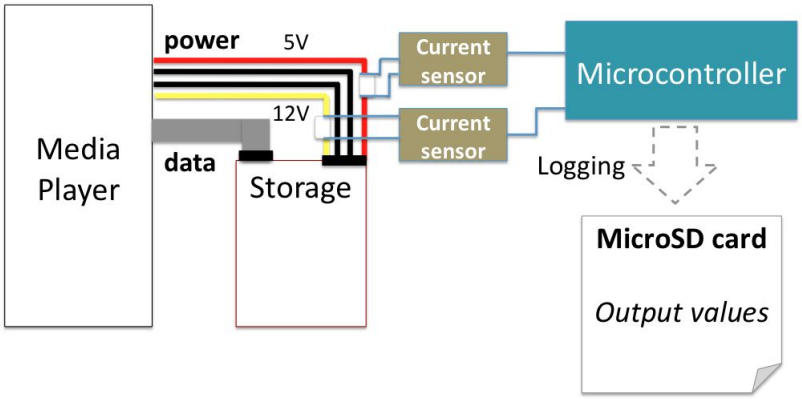

Figure 5.2: Logical diagram of the disk power measurement device

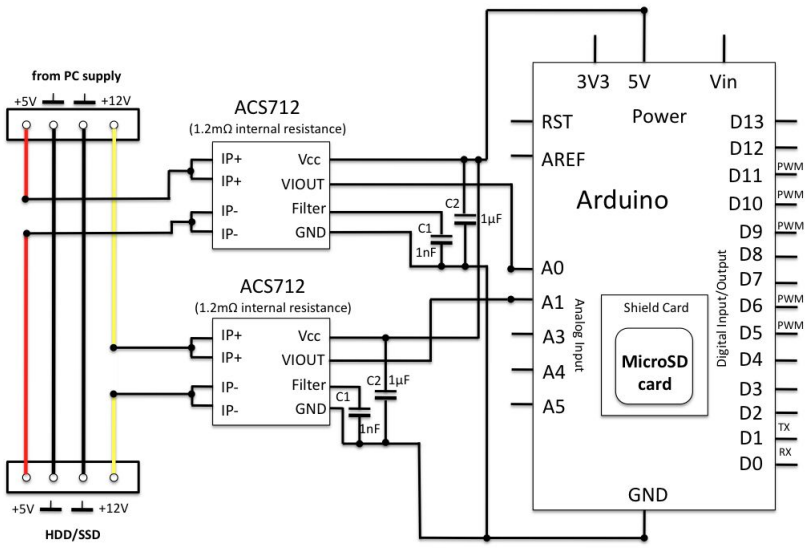

Figure 5.3: Electrical diagram of the disk power measurement device

5.2.2 Analysis and Interpretation of Log Data

The data collected by the measurement device is the current value taken from the power supply lines. The amount of data being taken depends on the duration of the experiments. As mentioned above that the sampling frequency of the device is about 12.5 Hz, we calculated the average power consumption at that frequency. From the obtained average power consumption, we calculated the energy consumption of the disk based on the duration of playback.

5.3 Benchmarking Environment

The test is comprised of a media player with HDD or SSD and also one unit of NAS server. For each experiment, the test setup is set based on the setup requirement. The experimental setup is shown in Figure 5.4

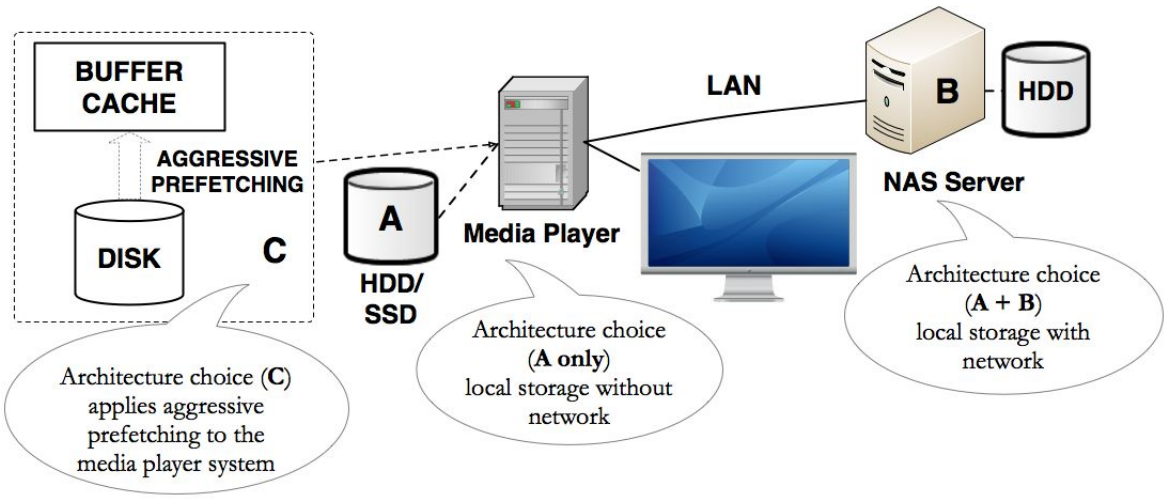

Figure 5.4: Experimental Setup of Storage Architectures

Machine setup:

- **Media player:**
  - CPU: 2.66 GHz Intel Core 2
  - Memory: 884 MB
  - OS: GeeXboX
  - Local storage:
    - HDD:
      - Model: Seagate SATA 2.5 inch ST9250315AS
      - Capacity: 250 GB

        * SSD:
            · Model: Intel X25-M SATA 2.5 inch
            · Capacity: 120 GB

- **Network-attached storage (NAS):**
    - Machine: QNAP TS210 TurboNAS
    - Storage:
        * HDD:
            · Model: Western Digital WD3200AAKS
            · Capacity: 320 GB

Workload setup:

- **Application:**
    - MPlayer
- **Data:**
    - Movie file:
    - Size: 4.7 GB
    - Quality: High definition (.MKV file)
    - Duration: 1 hour 45 minutes

5.4 Discussion

As mentioned above, we applied the systemic approach for the comparative analysis. One of important thing in the experiment is to make sure the correctness of the result obtained. In the experiment, we started by simple experiments first instead of running the whole experiments at a time. Then, by the results obtained, we analyzed the correctness of it and improved the methodology being used.

The measurement device is one of the core components in this experiment. The device was improved by modifying the configuration to provide more accurate results. As the initial step, we tried to calibrate the device to make sure the results measured are correct enough by comparing the results with the values from the data sheets of the hardware.

Chapter 6

Result and analysis

In this section, we present the results and analysis of testing the storage architectures in home media player. Before starting the comparison, it is also important to view and understand the power consumption based on the disk drive characteristics used in the experiment. After that, we present the results and the comparative analysis of energy consumption of local storage, HDD and SSD in media player. Then, the results and analysis between local storage and network storage is presented.

Since the energy consumption of electrical devices in home depends on the users' usage behavior, we consider several user scenarios and present the analysis of overall energy consumption of the home media player.

6.1 Basic power consumption analysis of storage in the experiment

For HDD, we observed and analyzed the power consumption of the drive's operating modes: sleep, standby, idle and active. Meanwhile, since the SSD drive we use in the experiment is NAND-based flash memory and will consume approximately the same power consumption after it is powered on, we observed and analyzed its power consumption after media player is turned on. In the data sheet specification of the SSD we used in the experiment, it mentions that the SSD also has idle mode by DIPM (Device Initiated Power Management). However, it must be supported by specific chipsets (usually in laptop). Since the media player system we used in the experiment does not support it, we do not include it in this experiment. All the results measured in the same system of the home media player (GeeXboX).

In the measurement analysis, we did repeated experiment of each case to include the variances of data and summarize the data obtained by a certain amount of confidence interval.

6.1.1 Basic power consumption analysis of HDD

HDD has different power consumption for different modes, which consist of standby, idle and active. In the experiment, we did the real measurement and also include the power consumption of the same modes provided by the vendor for reference. In the experiment, we take samples from the repeated experiments and make the analysis. The hard drive we use in the experiment is Seagate SATA HDD 2.5 inch ST9250315AS. It only consumes the power from the 5V power supply cable.

Standby Mode

The standby mode measurements were conducted by issuing the ACPI standby command and recording the power consumption. Multiple samples were collected to expose any variation the measurements. Disk is put into standby mode after the disk was in idle mode for a certain period of time.

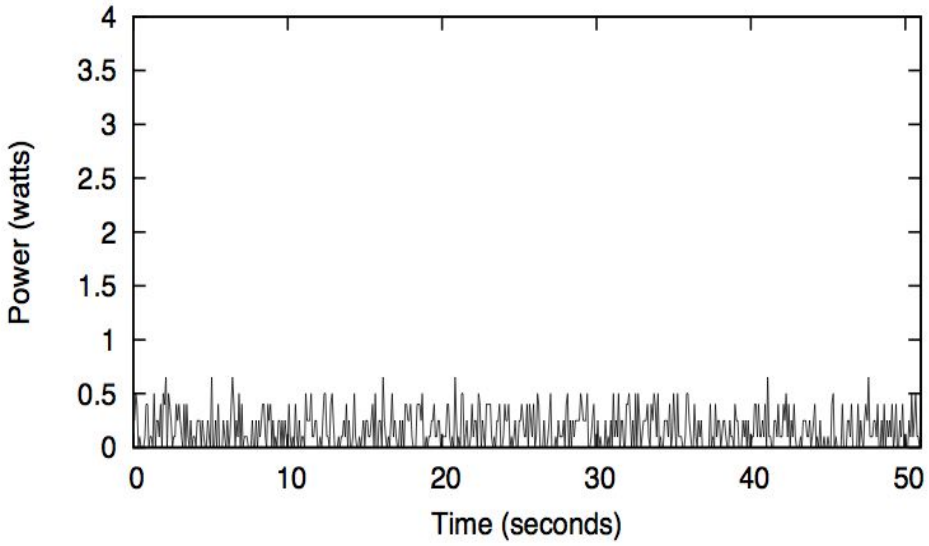

Figure 6.1: HDD Standby Power Consumption

Figure 6.1 shows a sample of power consumption during idle state. We took 100 samples of the power consumption of the idle state and calculated the mean value is 0.1615 Watts with standard deviation value 0.1677. From the results obtained, we can say that at 99% confidence interval for the mean is (0.118,0.205). From the vendor's data sheet, the power consumption of standby mode of the same HDD is about 0.20 watts. It is almost near to the value provided by the vendor.

Idle Mode

The idle mode measurements were conducted by letting disk idle without any workload. Multiple runs were collected to expose any variation the measurements. The Figure 6.2 below shows the power consumption during idle mode of the disk. In idle mode, the drive is not servicing any requests, but the platters are spinning.

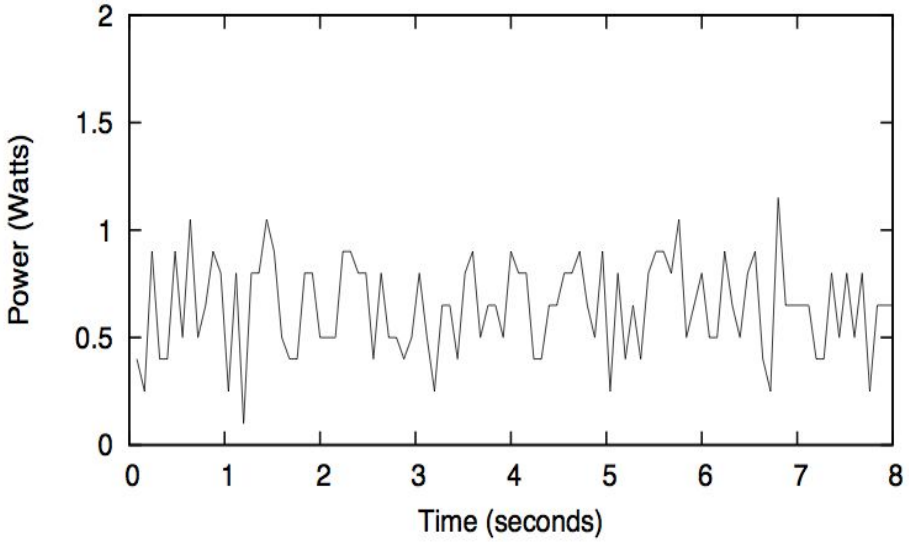

Figure 6.2: HDD Idle Power Consumption

We took 100 samples of power consumption during disk idle mode and measured that the mean value during idle mode is about 0.641Watts with standard deviation value 0.219. From the results obtained, we can say that at 99% confidence interval for the mean is (0.585, 0.697). The measured value for the idle mode is nearly the same as the reference value provided by the vendor, 0.67 watts.

Active Mode

Active mode power consumption is measured by capturing data when the disk is servicing I/O request (data seek/read) during the movie playback. This is the mode, which consume the highest power compared to the other modes. The Figure 6.3 below shows the power consumption of many seek operations.

We took 100 samples of power consumption of seek operations and calculated the mean value of it is about 1.747 Watts with standard deviation 0.389. From the results obtained, we can say that at 99% confidence interval for the mean is (1.647, 1.847). The measured active mode power consumption is a little bit higher than the value provided by vendor, which is 1.54 Watts.

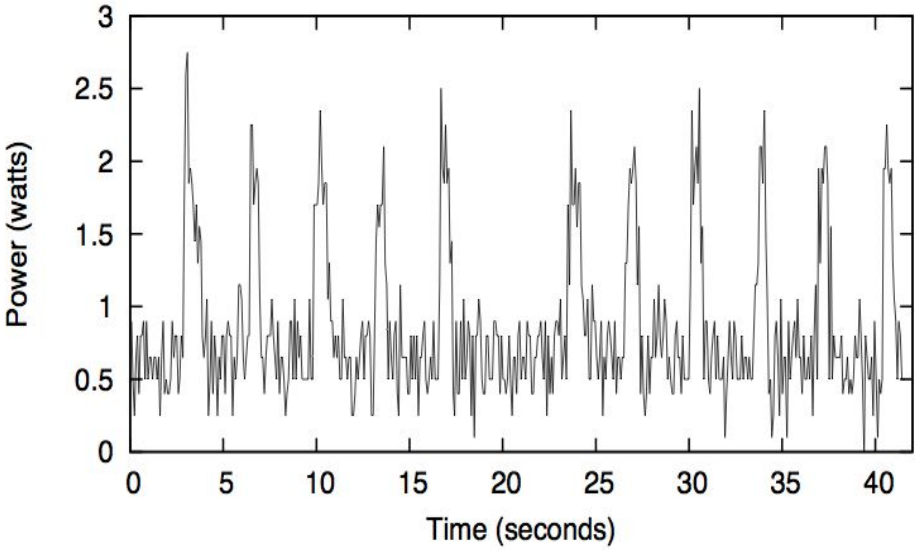

Figure 6.3: HDD Active (seek) power consumption

Sleep Mode

Sleep mode is the mode when the machine is powered-off so there is no power consumption at that time.

Spin-up Transition

When the disk is in standby mode and need to be accessed when the system needs disk I/O, the disk needs to be spin-upped in order to be active for the operation. This spin-up transition takes high power consumption and a certain period of time. We measured the power and time needed for the spin-up transition by capturing the data when the disk is triggered after entering standby mode for a period of time. The Figure 6.4 shows the graph of power consumption characteristic of HDD during spin-up transition.

For the spin-up transition, the duration needed for the transition is essential so we measured and analyzed it by taking 35 samples. The spin-up duration measured is shown in the Figure 6.5.

The mean value of the measured spin-up duration is 1.34 seconds with variance value of 0.0086. From the obtained data, we can say that at 99% confidence interval for the mean is between 1.212 and 1.468.

To analyze the energy consumption needed for the disk spin-up transition, we measured by taking 35 samples. The measured energy consumption of HDD spin-up transition from repeated experiments is shown in the Figure 6.6.

The mean value of the spin-up energy consumption is 3.87 Joules. From the obtained data, we can say that at 99% confidence interval for the mean is between 3.498 and 4.240. In the data sheet, the vendor only provide the maximum current needed during the spin-

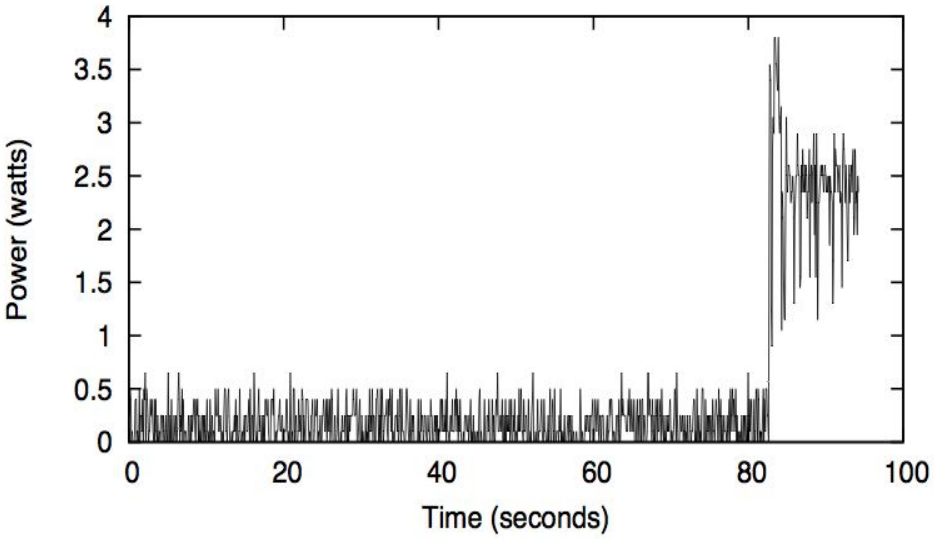

Figure 6.4: HDD Standby-to-Active Transition

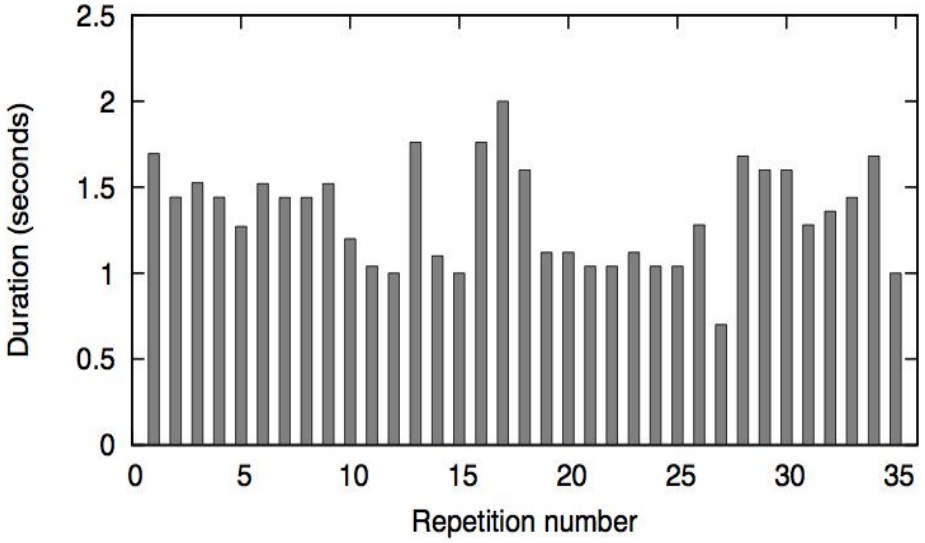

Figure 6.5: HDD Spin-up Duration from 35 samples

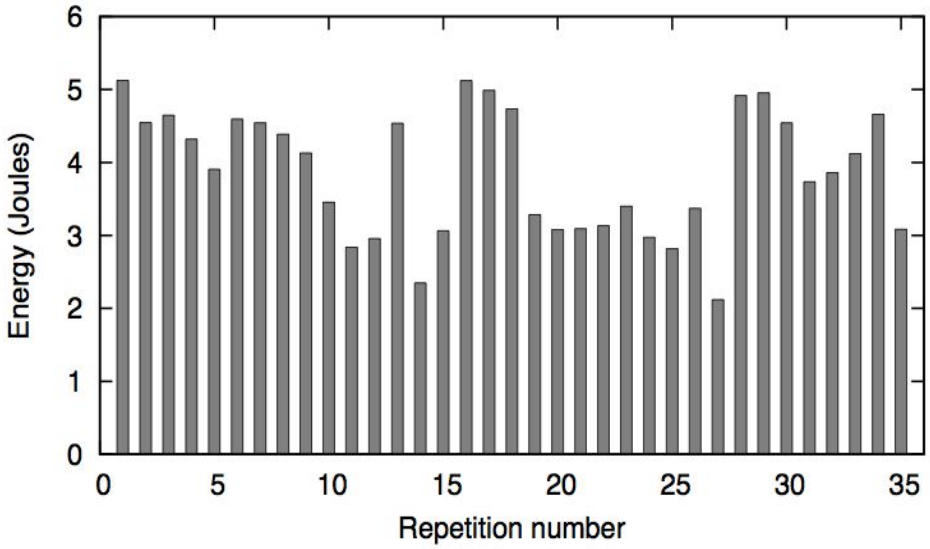

Figure 6.6: HDD Spin-up Energy Consumption from 35 samples

up transition that is 1 Ampere. We measured the average current is about 0.6 ampere, which is still reasonable if we take the value from the data sheet as reference.

6.1.2 Basic power consumption analysis of SSD

Since the SSD used in the experiment only support one mode, active so we measured and did the analysis based on the data. The SSD we use in the experiment is Intel X25 SATA SSD 2.5 inch 120 GB. It only consumes the power from the 5V power supply cable.

In the experiment, we found no big differences of power consumption of SSD whether it is servicing I/O operation or not. The power consumption was measured by taking multiple samples to analyze the variance. The Figure 6.7 below shows the power consumption of SSD. From the data we obtained, we observed small variances in the power consumption but with the same average value. It is different with HDD because SSD is NAND-based flash storage and the power consumption depends on the chips operating inside. When there is operation, there will be chips operating in parallel.

We took 100 samples and calculated its mean value of power consumption to be about 0.54 Watts with standard deviation value 0.199. From the results obtained, we can say that at 99% confidence interval for the mean is (0.487, 0.589). The measured power consumption of SSD is higher than the value provided by the vendor. The possible reason is that the vendor uses different operating system with specific benchmarking software.

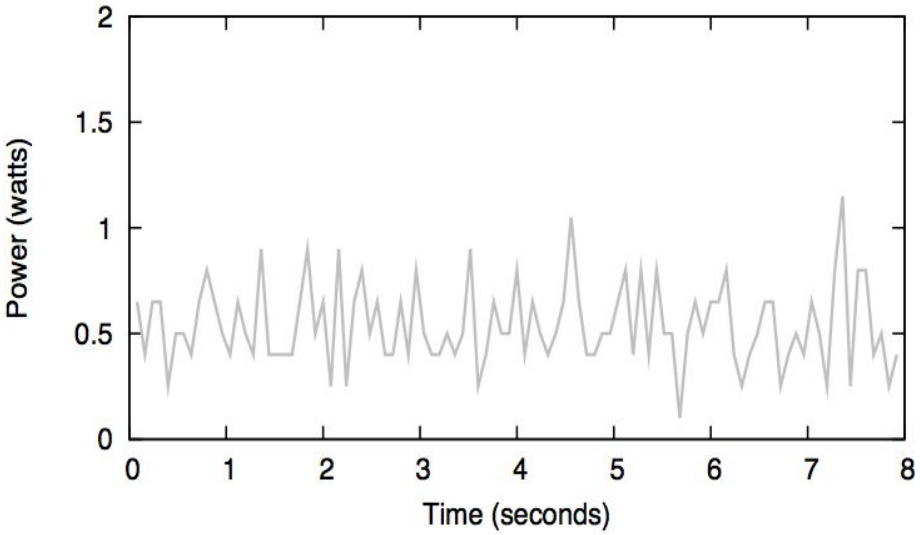

Figure 6.7: SSD Active Power Consumption

6.1.3 Basic power consumption of HDD inside NAS

Since NAS has its own system and there might be a lot factors affecting, we measured the basic power consumption of the HDD inside NAS with different operating system, Linux Fedora 13. The hard drive we use inside NAS is Western Digital WD3200AAKS 3.5 inch with 320GB capacity. The reason is because HDD offers bigger capacity with cheaper price, so it is more reasonable to use it as storage inside NAS.

Mode	Measured power (watts)	Data sheet based power (watts)
Standby	1.25	1
Idle	5.02	7.5
Active	7.7	7.75
Sleep	0	0

Table 6.1: Measured power consumption of HDD in NAS

The power consumption of the HDD is higher because it uses both of the power supply 5V and 12V. It is different if we compared it with the 2.5 inch HDD as we presented before since 2.5 inch HDD or SSD consume only from the power supply 5V.

6.2 Comparison between local storage (HDD and SSD)

In the section, we present the results and analysis for the local storage architecture inside home media player. For comparison, we only change the storage used inside the media

player with the same system, workload and settings. We measured the energy consumption of local storage with the default setting as the baseline and then add prefetching to see the affect on the energy consumption between these two different storage. We have several different testing setup for comparison:

1. HDD with default prefetching
2. SSD with default prefetching
3. HDD with aggressive prefetching
4. SSD with aggressive prefetching

6.2.1 Experiment 1 (HDD with default prefetching)

As mentioned before that Linux has already implemented the prefetching mechanism inside but with default maximum prefetching size of 128KB. We measured this experiment by running workload (movie playback with 105 minutes duration) with system default setting. To be able to observe the variance, we run and get the results for several repeated experiments. The Figure 6.8 below shows the power consumption characteristic graph for the whole movie playback.

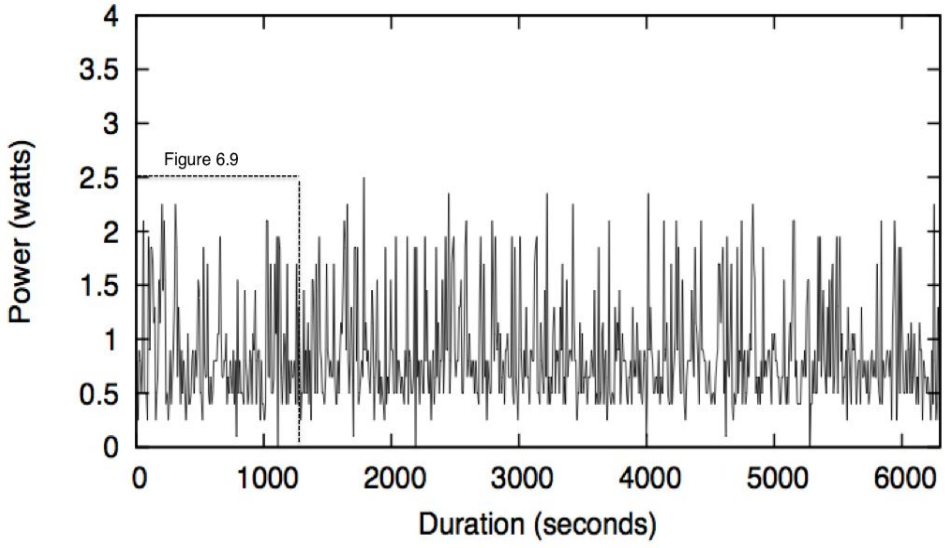

Figure 6.8: HDD Power Consumption during the whole movie playback

From the playback starting time, we observed that the disk is in active mode most of the time. During the movie playback, the disk needs to service with the data almost all the time with small idle interval. The disk seek/read operation dominates. The reason behind this is because the small prefetching size of the system which needs frequent disk

access. The Figure 6.9 below shows a small part of the power consumption during the playback. Since the prefetching size is 128KB and the movie needs to be loaded in time during the playback, the system needs to make disk access many times while the system page cache is never full. The size of page cache in the system is the same as the total memory size available for the system, which is 884MB.

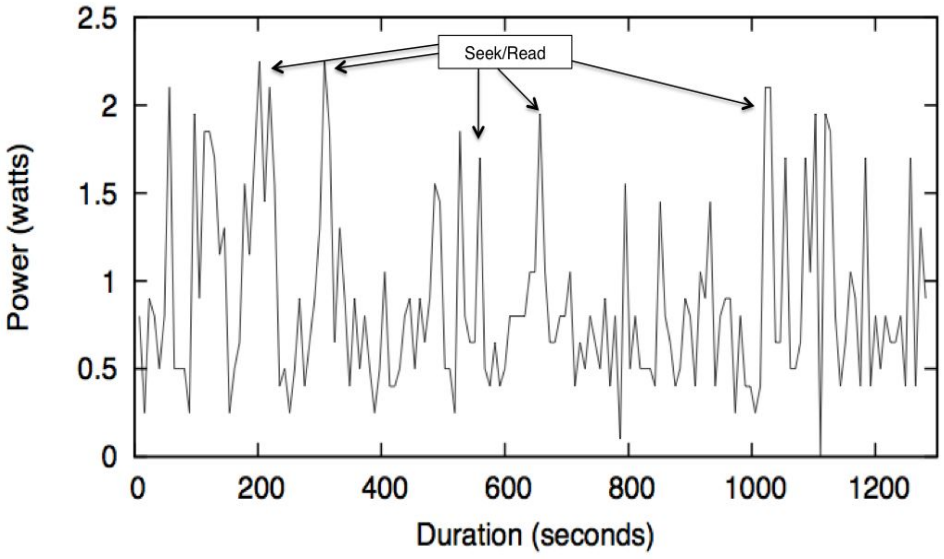

Figure 6.9: HDD Power Consumption from small part of the playback

From this experiment, we identify that the power consumption is mostly consumed by the disk doing seek and read operations, which is actually higher compared to other power modes.

We measured the energy consumption of it by repeated experiments to observe the variance as shown in Figure 6.10

From the obtained data, the mean value of the energy consumption of the experiments is about 6211.37 Joules with standard deviation value of 559.11. We can say that at 99% confidence interval for the mean is (5060.18, 7362.56).

6.2.2 Experiment 2 (SSD with default prefetching)

For comparison, this experiment uses the same setting and workload as experiment 1 but using SSD as the storage inside the tested home media player. To be able to observe the variance, we run and get the results for several repeated experiments. The Figure 6.11 below shows the power consumption characteristic graph for the whole movie playback.

From the Figure 6.11, we see that the power consumption of SSD during the whole playback has nearly the same average power consumption with little variance compared to HDD in experiment 1.

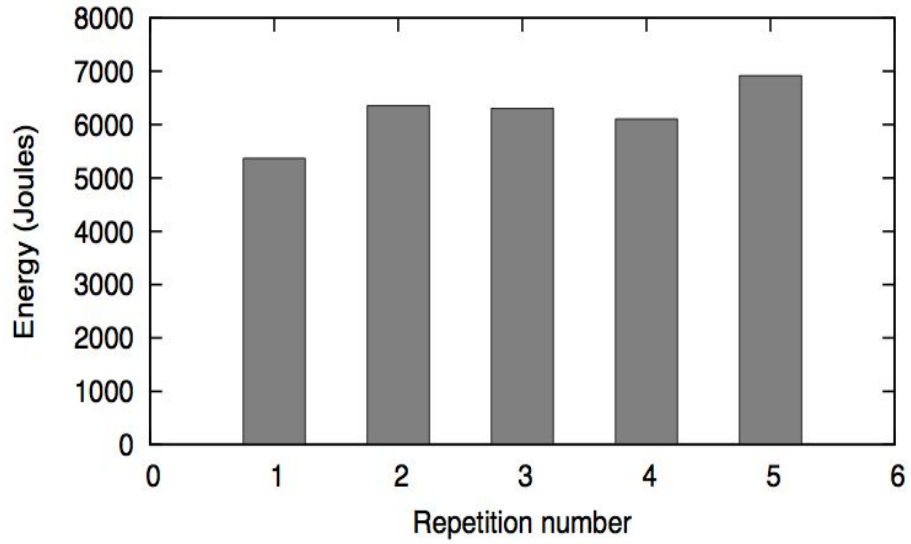

Figure 6.10: HDD Movie Playback Energy Consumption of repeated experiments

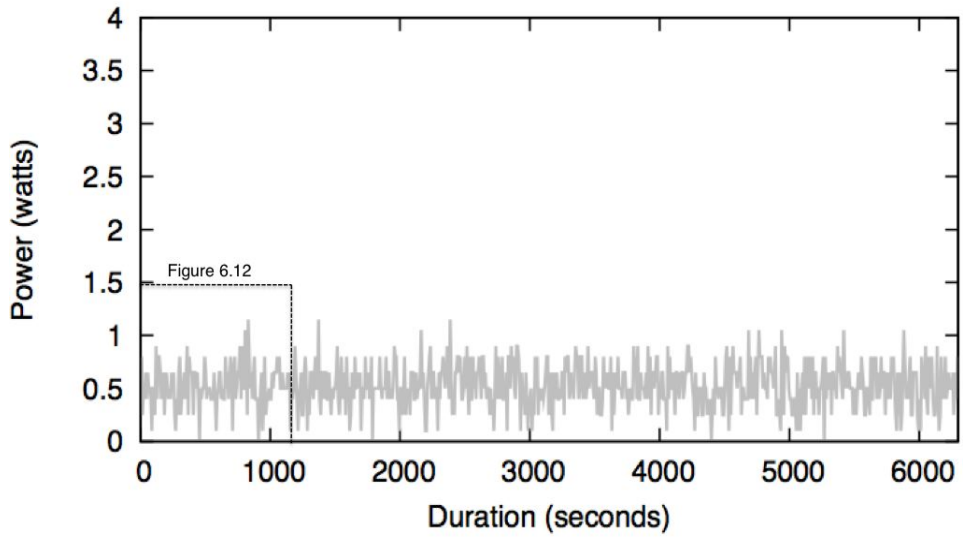

Figure 6.11: SSD Power Consumption during the whole movie playback

We present the power consumption of small part of the playback to observe the characteristics shown in Figure 6.12. The SSD always stay active all the time and the variance happens depends on the operation and number of chips being powered-on inside SSD. The system page cache situation is the same as experiment 1 which never get fully filled because of the small prefetching size.

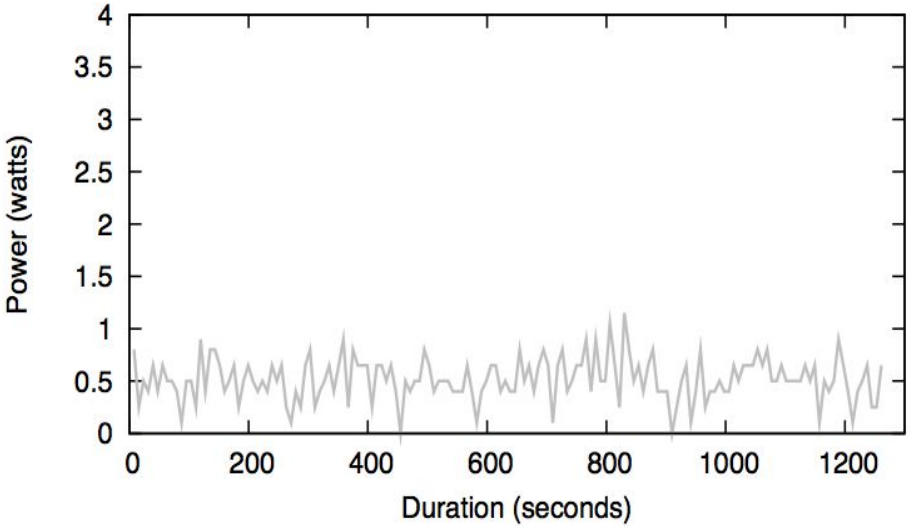

Figure 6.12: SSD Power Consumption from small part of the playback

We measured the energy consumption of it by repeated experiments to observe the variance as shown in Figure 6.13

From the obtained data, the mean value of the energy consumption of the experiments is about 3180.49 Joules with standard deviation value of 431.05. We can say that at 99% confidence interval for the mean is (2292.97, 4068.01).

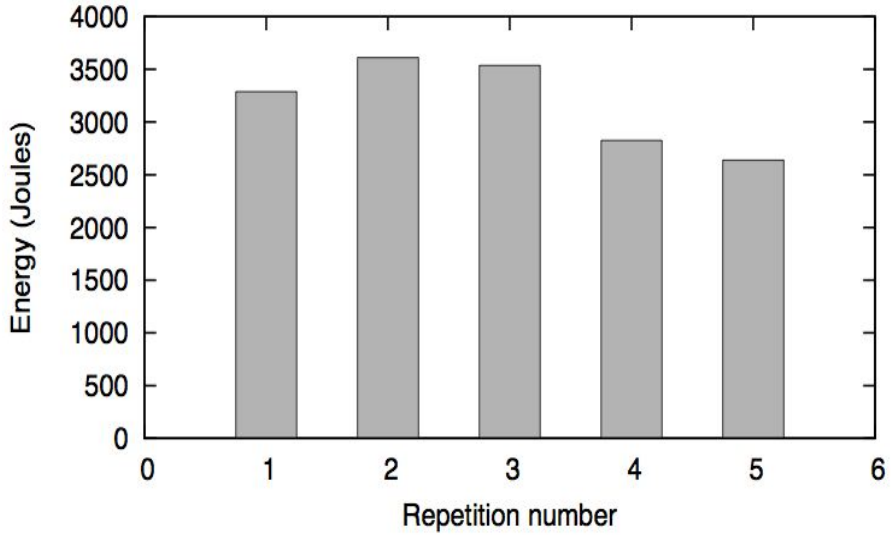

Figure 6.13: SSD Movie Playback Energy Consumption of repeated experiments

6.2.3 Experiment 3 (HDD with aggressive prefetching)

In this experiment, we present the energy consumption of the same movie playback with aggressive prefetching. Since Linux already has the prefetching mechanism, we set the prefetching size to be 512MB, which is large enough compared to the default prefetching size, 128KB. To achieve this, we set the system parameter through ACPI interface[1] (hdparm -a1048576 deviceid) before running the workload.

The Figure 6.14 below shows the power consumption graph during the whole playback.

From the graph, we can see the disk seek/read I/Os have been reduced compared with experiment 1 and also the disk entered standby mode many times. For better view, we show small part of the graph to see the difference as Figure 6.15.

We can see that there are only several disk seek/read operations with longer idle intervals and eventually entering standby mode. After the disk enters standby mode and the system needs to access data from the disk, the disk will enter spin-up transition. The reason is because we allow the system to preload large data from the disk into the page cache so that the disk can be put into rest and enter idle/standby mode. Since the system is a home media player, it only run the data of the workload that is the movie file., thus we can make sure that the disk I/O will be less since all the needed chunks of data had already been loaded into the memory.

We measured the energy consumption of it by repeated experiments to observe the variance as shown in Figure 6.16

The mean value of the energy consumption from the repeated experiments is 3757.62 with standard deviation 807.609. From the data obtained, we can say that at 99% confidence interval for the mean is (2094.78, 5420.46).

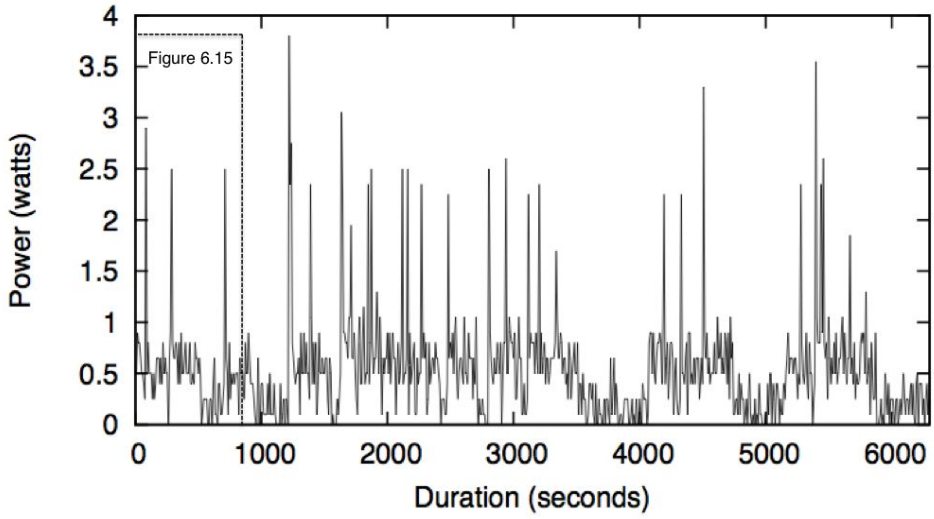

Figure 6.14: HDD Power Consumption with aggressive prefetching during the whole movie playback

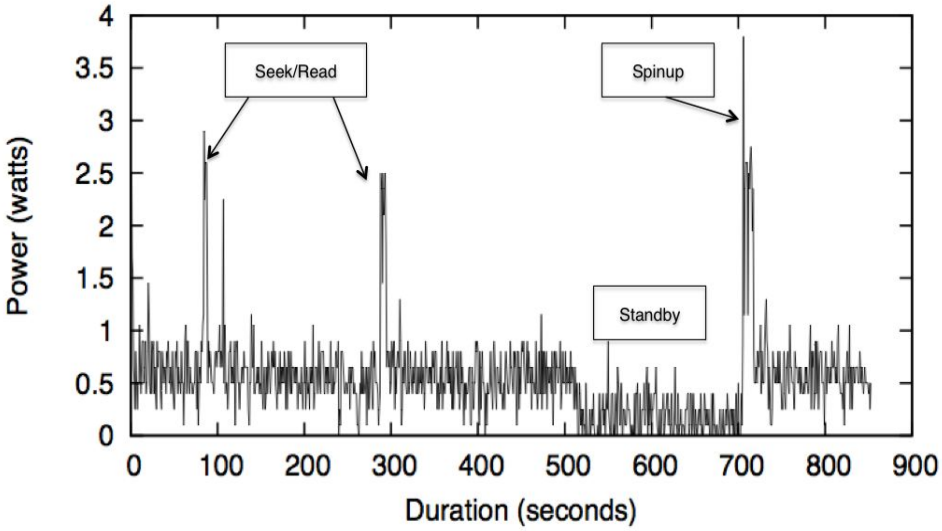

Figure 6.15: HDD Power Consumption with aggressive prefetching from small part of the playback

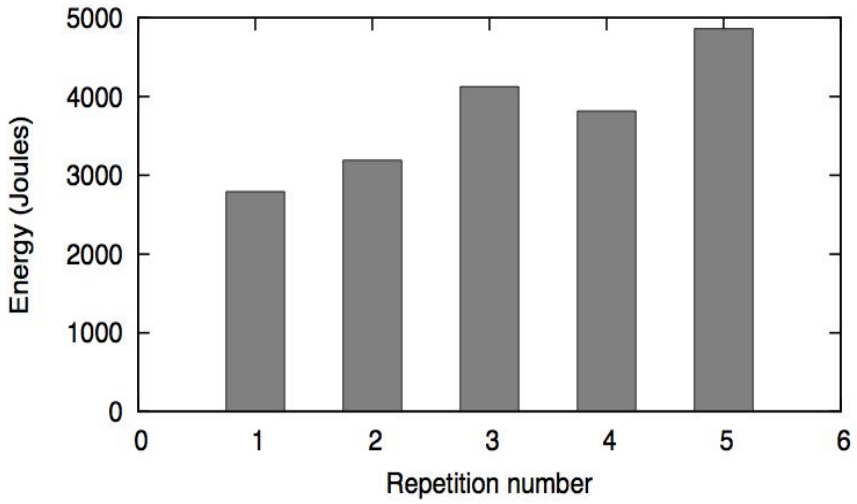

Figure 6.16: HDD Movie Playback Energy Consumption with aggressive prefetching of repeated experiments

6.2.4 Experiment 4 (SSD with aggressive prefetching)

This experiment uses the same setting as experiment 3 but with SSD as the storage inside the media player. Here, we would like to analyze the impact of applying aggressive prefetching to SSD. The system prefetching size is set to be 512MB.

The figure below shows the power consumption graph during the whole movie playback.

The power consumption graph shows not much difference if we compare it with SSD with default prefetching size.

To analyze the variance of the energy consumption, we collected the data from repeated experiments as shown in Figure 6.18

The mean value of the energy consumption is about 3448.19 Joules with standard deviation value of 617.58. We can say that at 99% confidence interval for the mean is (2176.61, 4719.77).

6.2.5 Comparison analysis

In this section, we present the comparison analysis with all the data by using t-test method to compare between two different systems. T-test is a method of unpaired observations of two systems.

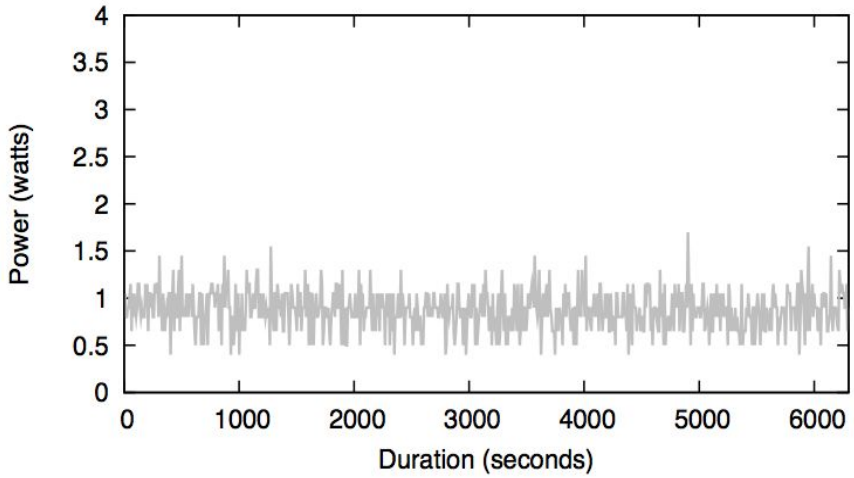

Figure 6.17: SSD Power Consumption with aggressive prefetching during the whole movie playback

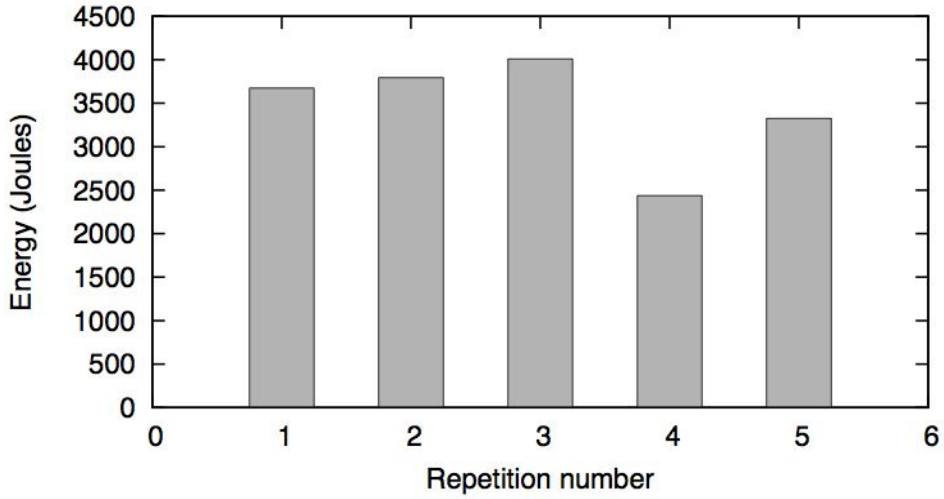

Figure 6.18: SSD Movie Playback Energy Consumption with aggressive prefetching of repeated experiments

Comparison between HDD with default prefetching (experiment 1) and HDD with aggressive prefetching (experiment 3)

We present the statistical analysis to show the difference of energy consumption between HDD with default prefetching and HDD with aggressive prefetching. We called the system with HDD with default prefetching and the system with HDD with aggressive prefetching as system 1 and system 2, respectively.

For system 1:

- Mean: 6211.37
- Variance: 312606.64
- Total sample: 5

For system 2:

- Mean: 3757.62
- Variance: 652232.94
- Total sample: 5

By t-test method, we calculated that 99% confidence for difference between these two systems is between 1431.99 and 3475.51. The confidence interval shows that system 1 indeed consumes much more power than system 2. It means that the impact of aggressive prefetching on energy consumption is big enough in HDD with 99% confidence interval that the difference of energy consumption is between 1431.99 and 3475.51.

Comparison between SSD with default prefetching (experiment 2) and SSD with aggressive prefetching (experiment 4)

Here, we provide the analysis to show the impact of aggressive prefetching on energy consumption of SSD. For simplicity, we called the system with SSD with default prefetching and the system with SSD with aggressive prefetching as system 1 and system 2, respectively.

For system 1:

- Mean: 3180.49
- Variance: 185802.84
- Total samples: 5

For system 2:

- Mean: 3448.19
- Variance: 381404.87

- Total samples: 5

By t-test method, we calculated that 99% confidence for difference between these two systems is between -1362.33 and 826.93.

The confidence interval includes zero. Therefore, at this confidence level the two systems are not different. It means that we can say that with 99% confidence interval, there is no different on the energy consumption of SSD with default prefetching and aggressive prefetching.

Comparison between HDD with aggressive prefetching (experiment 3) and SSD with aggressive prefetching (experiment 4)

From the comparison result before, we notice that there is no difference of energy consumption between SSD with default and aggressive prefetching. Here, we want to know the difference of energy consumption between HDD with aggressive prefetching and SSD with aggressive prefetching. For simplicity, we called the system with HDD with aggressive prefetching and SSD with aggressive prefetching as system 1 and system 2, respectively.

For system 1:

- Mean: 3448.19
- Variance: 381404.87
- Total samples: 5

For system 2:

- Mean: 3757.62
- Variance: 652232.94
- Total samples: 5

By t-test method, we calculated that 99% confidence for difference between these two systems is between -1787.29 and 1168.06.

The confidence interval includes zero. Therefore, at this confidence level the two systems are not different. It means that we can say that at 99% confidence interval, there is no different in term of energy consumption between HDD with aggressive prefetching and SSD with aggressive prefetching. This tells us that by applying aggressive prefetching, the energy consumption can be further reduced to be the same level as SSD.

6.2.6 Summary of energy consumption of local storage

From the comparative analysis above, we observed that by default, the energy consumption of HDD is indeed bigger than SSD. Before we did the experiment, we knew this intuitively. Then, after applying aggressive prefetching, we notice the interesting result that the energy consumption of HDD can be reduced to the same level as SSD. On the other side, we also noticed that there is no impact of aggressive prefetching in term of energy consumption is SSD as listed in Table 6.2.

Architecture options	Mean energy consumption (Joules)
HDD with default prefetching	6211.37
SSD with default prefetching	3180.49
HDD with aggressive prefetching	3448.19
SSD with aggressive prefetching	3757.62

Table 6.2: Measured energy consumption of local storage in home media player

6.3 Energy consumption analysis of network-attached storage (NAS)

In this section, we present the results and analysis of energy consumption for the home media player architecture with NAS. Prefetching is also added in the experiment to see its impact in the energy consumption. We measured the overall energy consumption of local storage inside media player with energy consumption of HDD inside NAS. For prefetching, we set the read size of the network prefetching for the NFS sharing at client side. We have several different testing setup for comparison:

- HDD as local storage and NAS
- SSD as local storage and NAS

6.3.1 Experiment 5 (HDD as local storage and NAS)

In this experiment, we use HDD as the local storage in home media player and doing playback for the movie located inside a NAS server. Here, we measured the energy consumption of local storage (HDD, in this case) and the HDD inside NAS. We use NFS connection as the file sharing protocol inside the home media player system to connect to the NAS server. All the settings used are default ones.

The Figure 6.19 below shows the power consumption graph of the HDD as local storage of the home media player.

We can see from the power consumption graph that the disk always stayed in standby mode during the movie playback. It is because the movie being played was accessed from NAS so the hard drive as the local storage could be put into rest (standby mode) which resulted in much less energy consumption.

To be able to observe the variances happened during the measurements, we measured the energy consumption of HDD (local storage) during the whole playback for repeated number of experiments as shown as Figure 6.20 below.

From the data we obtained, the mean energy consumption value is about 1525.95 Joules with standard deviation value of 329.09. Beside that, we can also say that at 99% confidence interval, the mean value is between 848.37 and 2203.52.

Beside the energy consumed at the local storage of home media player, we also measured the energy consumption of HDD inside the NAS server. The Figure 6.21 below shows the power consumption of HDD inside NAS server during the whole movie playback.

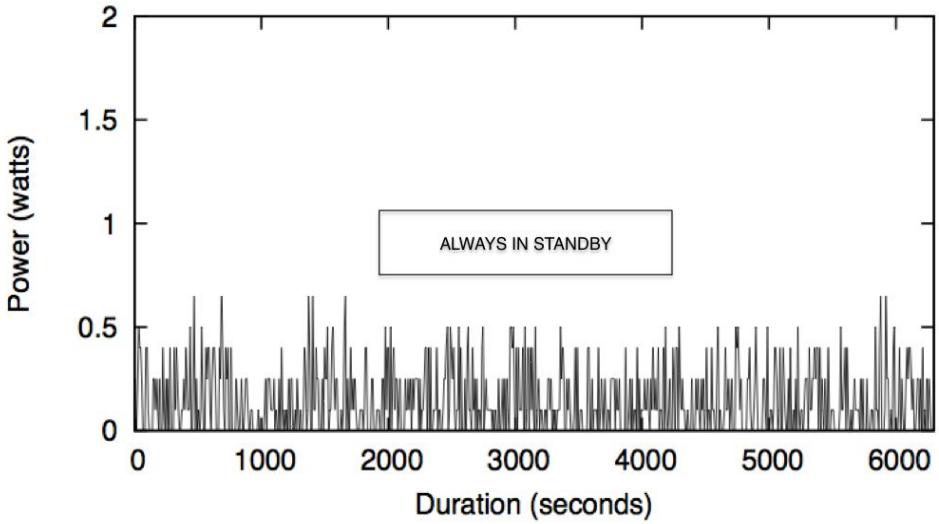

Figure 6.19: HDD (Local Storage) Power Consumption during the whole movie playback with NAS

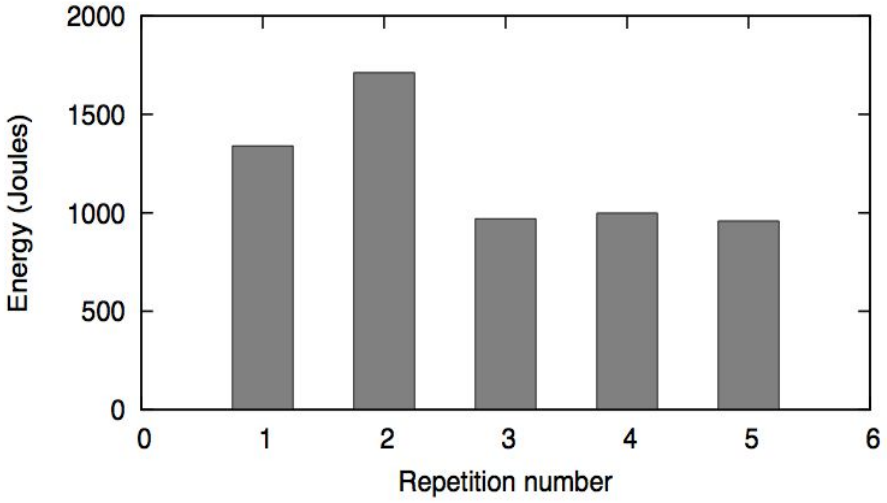

Figure 6.20: HDD (Local Storage) Energy Consumption with NAS of repeated experiments

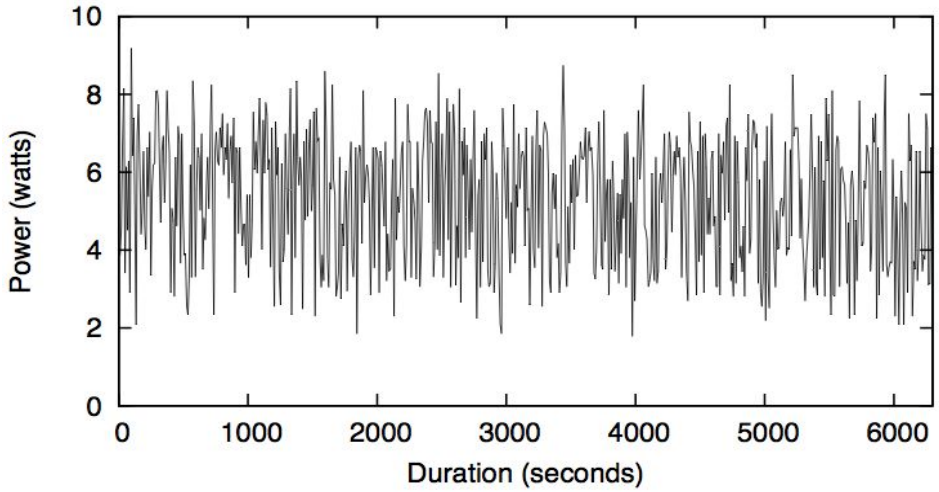

Figure 6.21: HDD (inside NAS) Power Consumption during the whole movie playback with NAS

From the graph, we see that the hard drive is in active mode all the time during the movie playback. This behavior is normal because a NAS server is supposed to serve many requests from clients in the LAN. To observe the variances in the energy consumption, we did repeated number of experiments and plotted it in the Figure 6.22.

From the data we obtained, the mean value of energy consumption of the hard drive inside NAS server is about 32312.6 Joules with standard deviation value 1809.642. At 99% confidence interval, we can say that the mean is between 28586.6 and 36038.6.

6.3.2 Experiment 6 (SSD as local storage and NAS)

In this experiment, we use SSD as the local storage for the home media player with NAS server as the storage for the movie files. Here, we measured the energy consumption of local storage (SSD, in this case) and the SSD inside NAS. We use NFS connection as the file sharing protocol inside the home media player system to connect to the NAS server. All the settings used are default ones.

The Figure 6.23 below shows the power consumption graph of the SSD as local storage of the home media player.

The power consumption shows no much difference with the case of the movie file being played directly from the SSD (in experiment 2 and 4).

We repeated the experiments several times to observe the variances happened. The graph 6.24 below shows the energy consumption of SSD as the local storage during the whole movie playback for repeated number of experiments.

The mean value of the energy consumption is about 3192.2 Joules with standard devi-

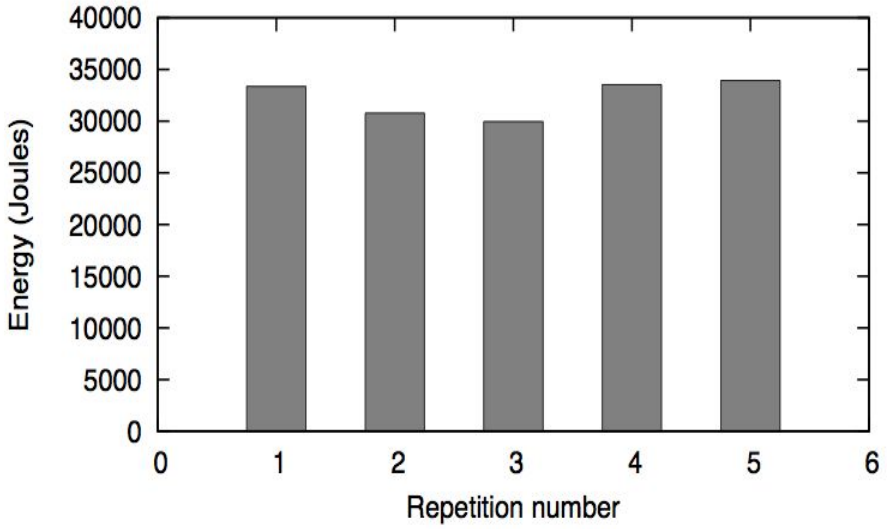

Figure 6.22: HDD (inside NAS) Energy Consumption with NAS of repeated experiments

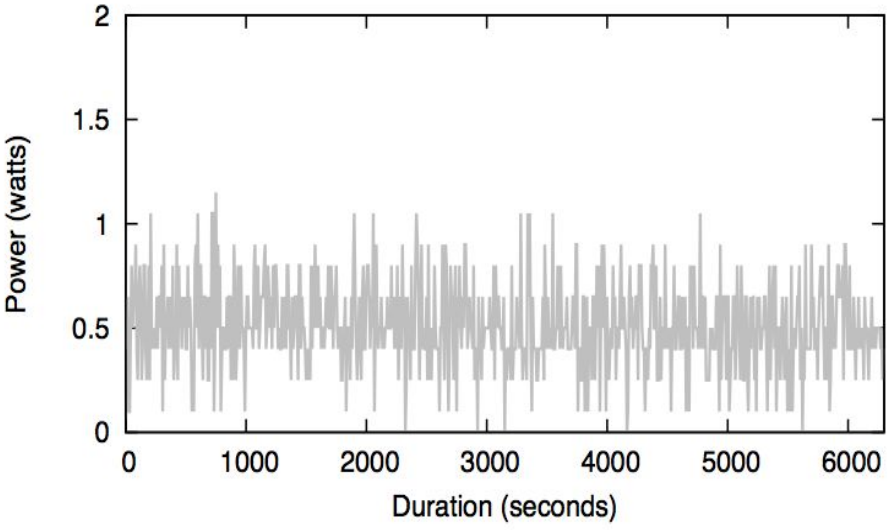

Figure 6.23: SSD (Local Storage) Power Consumption during the whole movie playback with NAS

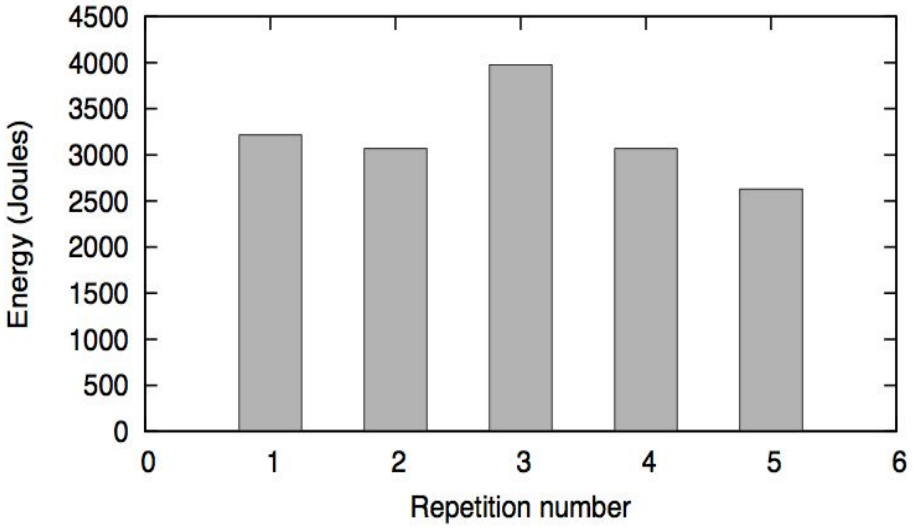

Figure 6.24: SSD (Local Storage) Energy Consumption with NAS of repeated experiments

ation value 491.12. At 99% confidence interval, the mean is between 2181 and 4203.4.

For the power consumption of the hard drive inside NAS, we show it in the Figure 6.25.

To observe the energy consumption of the hard drive inside NAS, we repeated several times and plot the data as shown in Figure 6.26.

The mean value of the energy consumption of the hard drive inside NAS during the whole movie playback is about 32004.8 Joules with standard deviation value 4015.72. At 99% confidence interval, the mean of the energy consumption is between 23736.55 and 40273.05.

6.3.3 Summary of energy consumption of network-attached storage architecture

From the experiments results and by doing t-test, we see that using as the local storage inside the home media player, HDD would consume less energy compared to using SSD because the hard drive would be put in standby mode all the time since there is no need for disk access during the movie playback. Since SSD runs only in active mode, it will consume the same amount of energy in any case as listed in Table 6.3

Architecture options	**Local disk energy (Joules)**	**HDD energy of NAS (Joules)**
HDD as local storage with NAS	1525.95	32312.6
SSD as local storage with NAS	3192.2	32004.8

Table 6.3: Measured energy consumption of local storage in home media player with NAS

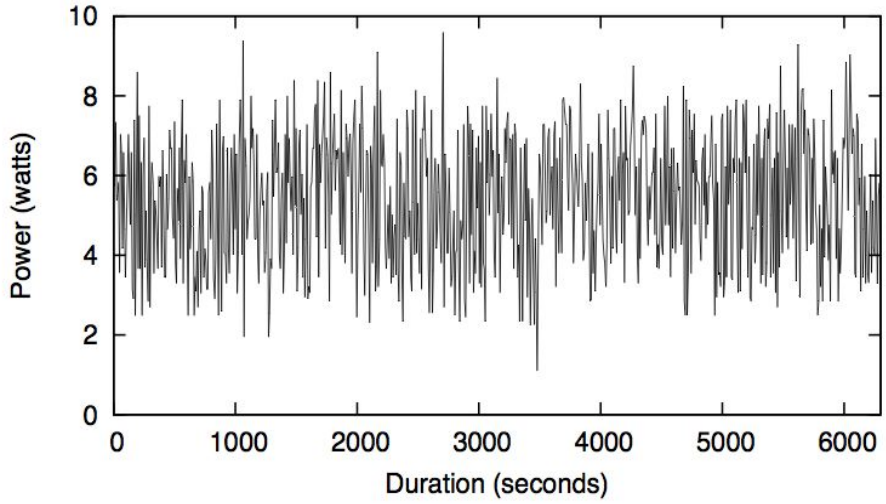

Figure 6.25: HDD (inside NAS) Power Consumption during the whole movie playback with NAS

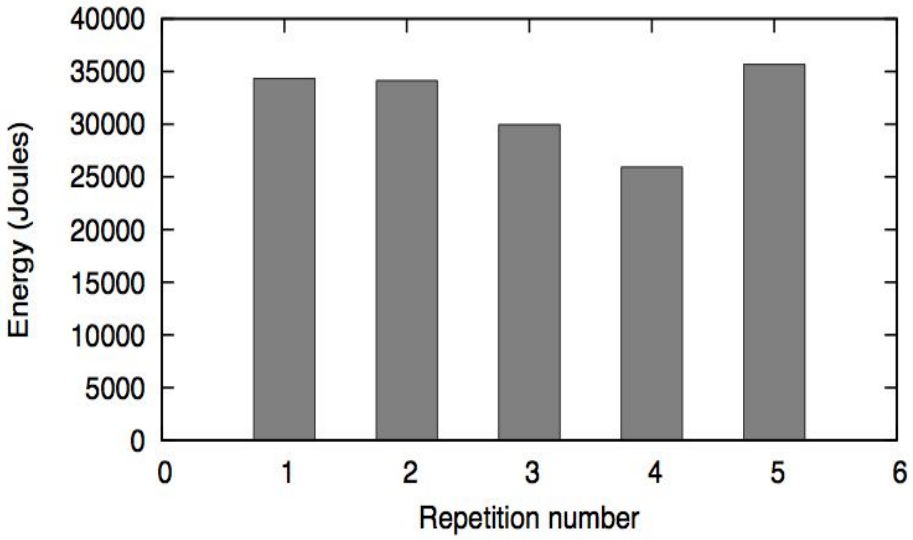

Figure 6.26: HDD (inside NAS) Energy Consumption with NAS of repeated experiments

Beside that, we noticed that the energy consumption of HDD inside NAS is nearly the same for experiment 5 and 6. In addition, we also measured the power consumption of the hard drive of the NAS and found that the hard drive is never put into standby mode (by the company setting). Thus, we can say that the energy consumption by the scenario of experiment 5 and 6 is already the optimized one in term of energy efficiency for the same energy consumption of NAS even during idle time.

6.4 More experiments for energy comparison between SSD and HDD with aggressive prefetching

From the experiment, we analyzed that the energy consumption of HDD with aggressive prefetching is close enough to the energy consumption of SSD. However, we noticed that the confidence interval is too large especially for HDD with aggressive prefetching concerning the variances of energy consumption between the same repeated experiments. So, it is necessary to do more experiments and collect more samples to support our statement that HDD is not less energy efficient than SSD for media playback. In the previous experiments, we did five repeated experiments for each case. Furthermore, we added the experiments to be 15 repeated experiments for the case of SSD and HDD with aggressive prefetching.

For HDD with aggressive prefetching, the energy consumptions for 15 repeated experiments is shown in Figure 6.27. The mean value of energy consumption is about 2649.18 Joules with 99% confidence that the mean value is between 1915.56 and 3427.79. The range of the mean value becomes smaller compared to the previous sample sets.

For SSD, the energy consumptions for 15 repeated experiments is shown in Figure 6.28. The mean value of energy consumption is about 3298.87 Joules with 99% confidence that the mean value is between 2812.88 and 3784.86.

After doing more experiments and analyzing the results, we can be more confident to say that HDD is not less energy-efficient than SSD for media playback with the system support of aggressive prefetching.

6.5 Discussion

In this chapter, we have presented the experiment results with the analysis of the energy consumption of storage architectures in home media player. The main metric we analyzed in the experiment is the energy consumption in the context of home media player. We did the experiment by firstly investigating the energy consumption on default system setting and then explored prefetching as the method for optimizing energy efficiency in HDD. From the results, we obtained some interesting findings such as the big impact of aggressive prefetching to reduce energy consumption of HDD to reach the same level as SSD. By this, we can say that in home media player, SSD is not a better choice to replace HDD even in the context of energy efficiency.

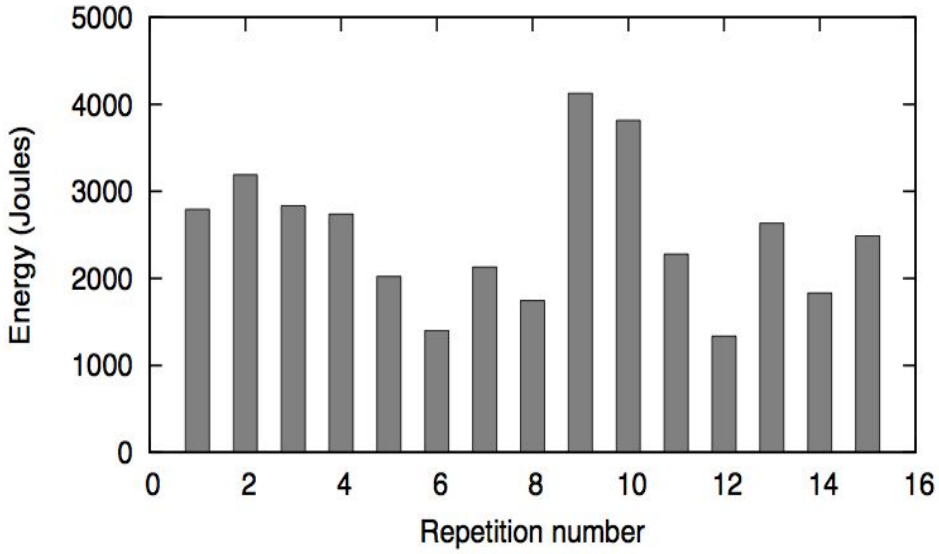

Figure 6.27: Energy consumptions of HDD with aggressive prefetching with 15 repeated experiments

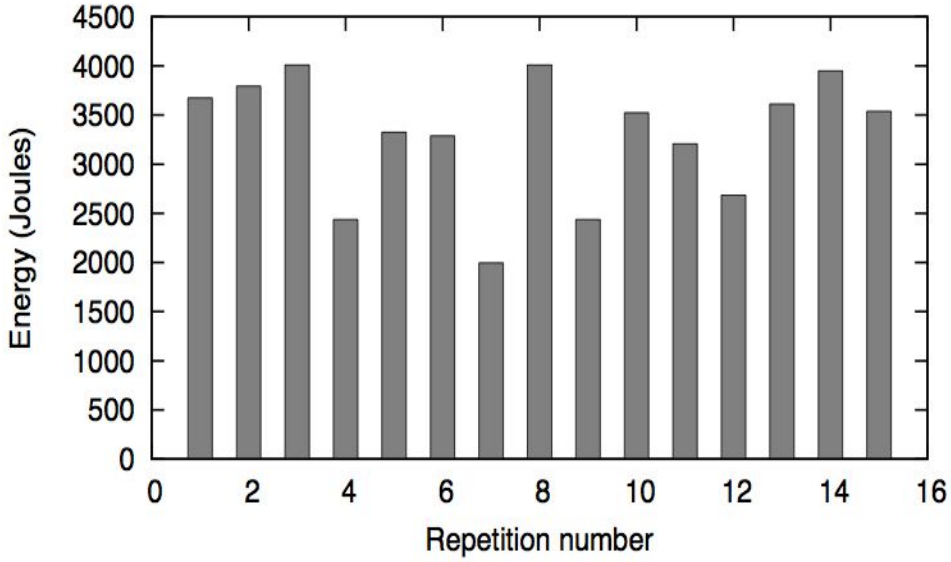

Figure 6.28: Energy consumptions of SSD with 15 repeated experiments

Chapter 7

Energy consumption of storage architectures in home media players

In the previous chapter, we presented the experimental results with all the analysis. From the data analyzed, we had some findings. However, they are only data, so we present our insights based on the findings in this chapter.

We measured and analyzed the energy consumption of storage architectures such as HDD, SSD as well as NAS in a home media player with video playback as the workload. As we know that reducing energy in home entertainment system is highly desirable, we consider the energy consumption of the storage architectures as the main metric. Beside that, the other important aspects are hardware cost, storage capacity, playback performance, and maintenance effort. The data analyzed before is valuable because it gives us insights to identify more energy and cost efficient home entertainment systems.

7.1 Energy and cost efficient home media players with HDD as the storage

One of the interesting finding we have found is that by applying aggressive prefetching, the energy consumption of HDD can be reduced to the same levels as SSD for video playback. In the analysis, we compared the energy consumption between HDD with aggressive prefetching and SSD during the video playback and found this result. This fact tells us that for home media players, using HDD as the storage is still a good choice even in term of energy efficiency as compared to SSD. It means we can have home entertainment system with bigger capacity of storage, cheaper price and less energy consumption. All of these will affect directly on the overall annual running cost of home media players.

To clearly present the impact of the finding on home media players, we use a real use case scenario as shown in Figure 7.1 to describe the overall picture. The use case scenario of home media players in a house is considered as below:

- Total users: 3 people
- The frequency of watching movies for each user is three times a week

- Each user store 1 new file per week
- They like to watch only high-definition (HD) movies which file size is about 4.7 GB and the average playback duration is 105 minutes
- They prefer to store every movie files into their home media player so that they can watch again anytime in the future
- Supposed that the average cost of electricity per Kwh is 0.12 USD

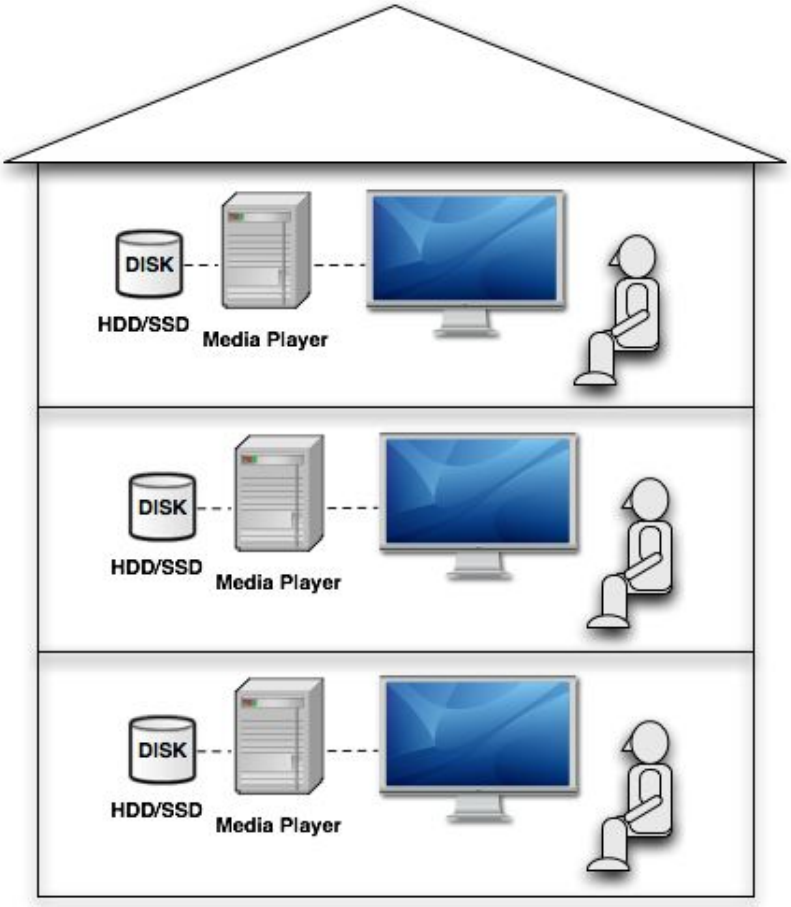

Figure 7.1: Use case scenario diagram for home media players using local storage

For the purpose of this scenario, we measured the overall power consumption of a mini pc, ZOTAC ZBOX, which is targeted as a home media player.

Power consumption of the mini pc during the playback measured with SSD as the local storage is about 19.38 Watts. Assuming that the power consumed by SSD is the same as

in the system we used in our experiment, we can get the power consumption of the mini pc without storage to be about 18.87 Watts.

For showing the impact clearly, we consider two different cases. The first one is the case where all users use HDD as their local storage and the second case is to use SSD as the local storage.

From the experiment, we knew that HDD with aggressive prefetching consumes nearly the same energy level as SSD, so we assume that the use has its system optimized with aggressive prefetching. The average power consumption by the mini pc (ZOTAC ZBOX) during the playback with HDD as local storage and aggressive prefetching applied is about 19.47 Watts. Based on the given scenario, the annual electricity cost for the home media players in the house is about 1.91 USD. We calculated the amount of movies being stored in a year, as the capacity needed for the home media players to be about 244.44 GB. For the total cost, we consider storage cost and electricity cost. For the storage cost, we calculated it as the price needed to have the required capacity in a year. It costs 244.44 GB x 0.11 USD (HDD cost as of 3/17/2011) = 26.88 USD. Since there are 3 users, then the total storage cost is estimated about 80.65 USD. In total, they need to spend about 82.57 USD per year for the running cost.

In the second case, we consider that all users use SSD for their local storage. The average power consumption by the mini PC (ZOTAC ZBOX) during the playback with SSD as the local storage is calculated to be about 19.37 Watts. Based on the given scenario, the annual electricity cost for the home media player is about 1.9 USD. The capacity needed for storing movie files in a year is 260 GB. We consider the total of storage cost and electricity cost as the total running cost. For the storage cost, it costs 244.44 GB x 1.80 USD (SSD cost as of 3/7/2011) = 439.99 USD. Since there are 3 users in total, the total storage cost is estimated to be about 1,319.98 USD. In total, they need to spend about 1,321.88 USD per year.

We put these numbers on the Table 7.1 with the other important aspects to consider in choosing storage type like noise, heat, playback performance and maintenance complexity. Noise and heat of each storage type are commonly known. The criterion for playback performance is given based on the user experience during the experiments.

Local Storage	**Electricity cost (USD)**	**Hardware cost (USD)**	**Noise**	**Heat**	**Playback Performance**	**Maintenance complexity**
HDD with aggressive prefetching	1.91	80.65	Small	Very little	Acceptable	Easy
SSD	1.9	1319.98	None	Extremely little	Good	Easy

Table 7.1: Cost and other aspects consideration of HDD and SSD as local storage

The annual running costs of using HDD and SSD as the local storage of home media player are 82.56 USD/year and 1321.88 USD/year. With our finding that shows by using HDD is not less energy-efficient than SSD in home media player and furthermore by the scenario above, we can save about 1239.31 USD/year for the running cost by using HDD as the local storage option.

7.2 Energy and cost efficient home media players with NAS as the storage

With NAS as the storage architecture option, the advantages that can be obtained are the centralized data management and saved space for duplicate files. From the experiment, we got to know that overall energy consumption of home media player with NAS is quite big. However, if there are many duplicate files stored by each user, then it is more cost efficient to use NAS.

To give clearer picture on the energy and cost efficient impact of using NAS in home media players, we use the same use case scenario as before. Here, we add another assumption that each user has exactly the same movie files being stored every week. Instead of accessing the movie directly from the local storage, all users store and access movie files from a NAS server as shown in Figure 7.2.

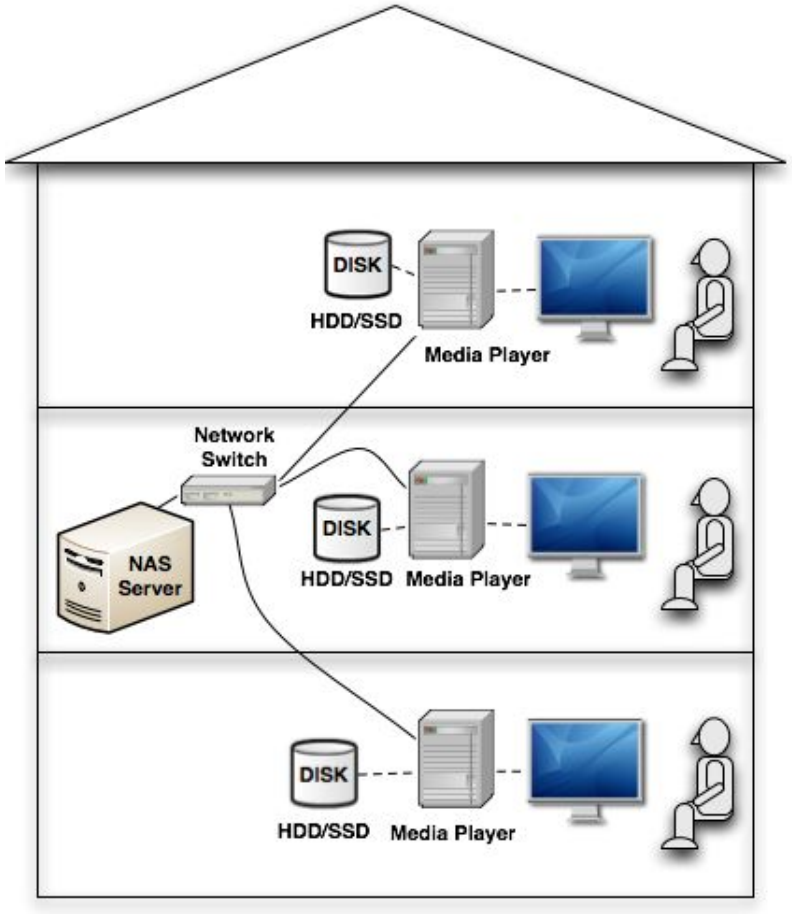

Figure 7.2: Use case scenario diagram for home media players using NAS server

The average power consumption by the mini pc (ZOTAC ZBOX) during the playback

with HDD as local storage with NAS is about 19.11 Watts. In this architecture, we need to consider additional power consumption of NAS server and the network switch, which are 14 Watts and 3.4 Watts, respectively as referred from the vendors' data sheet. Supposed that the NAS server and the hard drive inside is never turned off, then the annual electricity cost of NAS and the network switch itself is about 18.39 USD. The electricity cost of the home media players is estimated to be about 1.88 USD per year. Thus, the total electricity cost by this architecture is about 20.27 USD per year. The capacity needed to store the movie files for one year is 244.44 GB. Since HDD is used as the storage inside the NAS server, then the cost needed for the storage is about 26.89 USD per year. Therefore, they spend about 47.16 USD per year for the total running cost.

Storage option	**Electricity cost (USD)**	**Hardware cost (USD)**	**Noise**	**Heat**	**Playback Performance**	**Maintenance complexity**
HDD with aggressive prefetching	1.91	80.65	Small	Very little	Acceptable	Easy
NAS	20.27	26.89	Small	Very little	Acceptable	Difficult

Table 7.2: Cost and other aspects consideration of HDD as local storage and NAS as another storage option

The annual costs of running HDD and NAS as the storage option of home media player are 82.57 USD/year and 47.16 USD/year, respectively. So we can save about 35.41 USD if we use NAS server in this scenario. In the case that there are many duplicate files stored by each user, the architecture option to use NAS as the storage is more cost-efficient but it is more difficult to manage since it deals with management of NAS and LAN access.

7.3 Discussion

In this chapter, we presented our insights to summarize/identify our findings from all the experiments and results we analyzed before. Since reducing energy is important in home entertainment, we found that we can also have energy and cost efficient home media players with HDD as the storage by applying aggressive prefetching into the system. This is different from our initial intuitive, which assumed that HDD always consumes more energy compared to SSD in any case. Meanwhile, if users in the home have many duplicate files, then using NAS server as the storage could be better option in term of overall running cost.

Chapter 8

Conclusion

8.1 Research Assessment

In this research, we have investigated the energy consumption of storage architectures in home media player since storage is the main component and important asset of it. For each of the storage architecture, we have measured and analyzed the energy consumption by the system default setting for the same movie playback workload. By investigating the energy consumption of storage during the playback had led us to understand the power consumption characteristics of different storage and to explore the possibility to reduce energy consumption of HDD by applying aggressive prefetching.

From the analysis, we found that by default, the energy consumption of using HDD as the local storage in home media player is much bigger than SSD. However, by applying aggressive prefetching, the energy consumption of HDD can be reduced to be about the same level of SSD. For the architecture of home media player with network-attached storage (NAS), we thought that it could not be compared directly in term of the energy consumption since it would be unfair. From the data analyzed, we presented our insights to identify cost and energy efficient of storage architectures in home media players. Since each storage architecture offer different pros and cons, it depends on users to decide which architectures to apply by considering the tradeoffs.

In all, this research project has led to one contribution. Our contribution is the investigation of energy consumption of storage architectures in home media player with the comparative analysis and we showed that HDD is not less energy-efficient than SSD for media application.

8.2 Future Research Directions

In our experiment, we noticed that the power consumption of SSD in any case is about the same average value with small variance since it is always operating in active mode. However, some SSD vendors provide other mode like idle mode, which is stated to consume less power. To enable this mode, the machine must have the supporting chipsets. This idle mode is called DIPM (Device Initiated Power Management). The interesting question

is that how big the impact of this DIPM on the overall energy consumption of SSD.

Study more about the effective power-aware prefetching method for video playback in HDD.

Explore more for various use cases of home media player to see if further energy efficiency use case model exists for example in the case of using NAS as the multimedia storage.

Bibliography

[1] Advanced configuration and power interface (acpi) specification. `http://www.acpi.info`.

[2] Arduino open-source electronics prototyping platform. `http://www.arduino.cc/`.

[3] Geexbox embedded linux media center distribution. `http://www.geexbox.org/`.

[4] G. H. Cao. Proactive power-aware cache management for mobile computing systems. *IEEE Transactions on Computers*, 51(6), June 2002.

[5] D. Chen, G. Goldberg, R. Kahn, R. I. Kat, and K. Meth. Leveraging disk drive acoustic modes for power management. *IEEE Mass Storage System and Technologies*, 2010.

[6] F. Chen and X. Zhang. Caching for bursts (c-burst): Let hard disks sleep well and work energetically. In *Proceeding of the 13th international symposium on Low power electronics and design*, 2008.

[7] E. Y. Chung, L. Benini, and G. D. Micheli. Dynamic power management using adaptive learning tree. *ACM International Conference on Computer-Aided Design*, 1999.

[8] G. Dhiman and T. S. Rosing. Dynamic power management using machine learning. *International Conference on Computer-Aided Design*, 2006.

[9] F. Douglis, P. Krishnam, and B. Bershad. Adaptive disk spin-down policies for mobile computers. *2nd Symposium on Mobile and Location-Independent Computing*, pages 121–137, 1995.

[10] C. Gniady, A. R. Butt, Y. C. Hu, and Y. H. Lu. Program counter-based prediction techniques for dynamic power management. *IEEE Transactions on Computers*, 55(6), June 2006.

[11] A. Hylick, A. Rice, B. Jones, and R. Sohan. Hard drive power consumption uncovered. *ACM SIGMETRICS Performance Evaluation Review*, 35(3):54 – 55, December 2007.

[12] R. Jain. *The Art Of Computer Systems Performance Analysis*. John,Wiley Sons,Inc, 1991.

[13] M. Lee, E. Seo, J. Lee, and J. Kim. Pabc: Power-aware buffer cache management for low power consumption. *IEEE Transactions on Computers*, 56(4), April 2007.

[14] Y. L. Lu and G. D. Micheli. Comparing system-level power management policies. *IEEE Design and Test of Computers*, 2001.

[15] A. E. Papathanasiou and M. L. Scott. Energy efficient prefetching and caching. In *Proceedings of the annual conference on USENIX Annual Technical Conference*, 2004.

[16] A. E. Papathanasiou and M. L. Scott. Aggressive prefetching: An idea whose time has come. In *Proceeding of the 10th conference on Hot Topics in Operating Systems*, volume 10, 2005.

[17] F. Wu. *Advanced Operating Systems and Kernel Applications: Techniques and Technologies*, chapter 11 (Sequential File Prefetching in Linux), pages 218 – 261. Information Science Reference, 2010.

[18] F. Wu, H. Xi, J. Li, and N. Zhou. On the design of a new linux readahead framework. *ACM SIGOPS Operating Systems Review - Research and developments in the Linux kernel*, 42(5), July 2008.

[19] Q. Zhu, A. Shankar, and Y. Zhou. Pb-lru: A self-tuning power aware storage cache replacement algorithm for conserving disk energy. In *Proceedings of the 18th annual international conference on Supercomputing*, 2004.

Appendix A

Readahead (Prefetching) Source Code in Linux (2.6.38.3)

```
/*
 * mm/readahead.c - address_space-level file readahead.
 *
 * Copyright (C) 2002, Linus Torvalds
 *
 * 09Apr2002 Andrew Morton
 * Initial version.
 */

#include <linux/kernel.h>
#include <linux/fs.h>
#include <linux/gfp.h>
#include <linux/mm.h>
#include <linux/module.h>
#include <linux/blkdev.h>
#include <linux/backing-dev.h>
#include <linux/task_io_accounting_ops.h>
#include <linux/pagevec.h>
#include <linux/pagemap.h>

/*
 * Initialise a struct file's readahead state.  Assumes that the caller has
 * memset *ra to zero.
 */
void
file_ra_state_init(struct file_ra_state *ra, struct address_space *mapping)
{
ra->ra_pages = mapping->backing_dev_info->ra_pages;
ra->prev_pos = -1;
```

```
}
EXPORT_SYMBOL_GPL(file_ra_state_init);

#define list_to_page(head) (list_entry((head)->prev, struct page, lru))

/*
 * see if a page needs releasing upon read_cache_pages() failure
 * - the caller of read_cache_pages() may have set PG_private or PG_fscache
 *   before calling, such as the NFS fs marking pages that are cached locally
 *   on disk, thus we need to give the fs a chance to clean up in the event of
 *   an error
 */
static void read_cache_pages_invalidate_page(struct address_space *mapping,
     struct page *page)
{
if (page_has_private(page)) {
if (!trylock_page(page))
BUG();
page->mapping = mapping;
do_invalidatepage(page, 0);
page->mapping = NULL;
unlock_page(page);
}
page_cache_release(page);
}

/*
 * release a list of pages, invalidating them first if need be
 */
static void read_cache_pages_invalidate_pages(struct address_space *mapping,
      struct list_head *pages)
{
struct page *victim;

while (!list_empty(pages)) {
victim = list_to_page(pages);
list_del(&victim->lru);
read_cache_pages_invalidate_page(mapping, victim);
}
}

/**
 * read_cache_pages - populate an address space with some pages & start reads against
```

```
 * @mapping: the address_space
 * @pages: The address of a list_head which contains the target pages.  These
 *   pages have their ->index populated and are otherwise uninitialised.
 * @filler: callback routine for filling a single page.
 * @data: private data for the callback routine.
 *
 * Hides the details of the LRU cache etc from the filesystems.
 */
int read_cache_pages(struct address_space *mapping, struct list_head *pages,
int (*filler)(void *, struct page *), void *data)
{
struct page *page;
int ret = 0;

while (!list_empty(pages)) {
page = list_to_page(pages);
list_del(&page->lru);
if (add_to_page_cache_lru(page, mapping,
page->index, GFP_KERNEL)) {
read_cache_pages_invalidate_page(mapping, page);
continue;
}
page_cache_release(page);

ret = filler(data, page);
if (unlikely(ret)) {
read_cache_pages_invalidate_pages(mapping, pages);
break;
}
task_io_account_read(PAGE_CACHE_SIZE);
}
return ret;
}

EXPORT_SYMBOL(read_cache_pages);

static int read_pages(struct address_space *mapping, struct file *filp,
struct list_head *pages, unsigned nr_pages)
{
unsigned page_idx;
int ret;

if (mapping->a_ops->readpages) {
```

```
ret = mapping->a_ops->readpages(filp, mapping, pages, nr_pages);
/* Clean up the remaining pages */
put_pages_list(pages);
goto out;
}

for (page_idx = 0; page_idx < nr_pages; page_idx++) {
struct page *page = list_to_page(pages);
list_del(&page->lru);
if (!add_to_page_cache_lru(page, mapping,
page->index, GFP_KERNEL)) {
mapping->a_ops->readpage(filp, page);
}
page_cache_release(page);
}
ret = 0;
out:
return ret;
}

/*
 * __do_page_cache_readahead() actually reads a chunk of disk.  It allocates all
 * the pages first, then submits them all for I/O. This avoids the very bad
 * behaviour which would occur if page allocations are causing VM writeback.
 * We really don't want to intermingle reads and writes like that.
 *
 * Returns the number of pages requested, or the maximum amount of I/O allowed.
 */
static int
__do_page_cache_readahead(struct address_space *mapping, struct file *filp,
pgoff_t offset, unsigned long nr_to_read,
unsigned long lookahead_size)
{
struct inode *inode = mapping->host;
struct page *page;
unsigned long end_index; /* The last page we want to read */
LIST_HEAD(page_pool);
int page_idx;
int ret = 0;
loff_t isize = i_size_read(inode);

if (isize == 0)
goto out;
```

```
end_index = ((isize - 1) >> PAGE_CACHE_SHIFT);

/*
 * Preallocate as many pages as we will need.
 */
for (page_idx = 0; page_idx < nr_to_read; page_idx++) {
pgoff_t page_offset = offset + page_idx;

if (page_offset > end_index)
break;

rcu_read_lock();
page = radix_tree_lookup(&mapping->page_tree, page_offset);
rcu_read_unlock();
if (page)
continue;

page = page_cache_alloc_cold(mapping);
if (!page)
break;
page->index = page_offset;
list_add(&page->lru, &page_pool);
if (page_idx == nr_to_read - lookahead_size)
SetPageReadahead(page);
ret++;
}

/*
 * Now start the IO.  We ignore I/O errors - if the page is not
 * uptodate then the caller will launch readpage again, and
 * will then handle the error.
 */
if (ret)
read_pages(mapping, filp, &page_pool, ret);
BUG_ON(!list_empty(&page_pool));
out:
return ret;
}

/*
 * Chunk the readahead into 2 megabyte units, so that we don't pin too much
 * memory at once.
```

```
 */
int force_page_cache_readahead(struct address_space *mapping, struct file *filp,
pgoff_t offset, unsigned long nr_to_read)
{
int ret = 0;

if (unlikely(!mapping->a_ops->readpage && !mapping->a_ops->readpages))
return -EINVAL;

nr_to_read = max_sane_readahead(nr_to_read);
while (nr_to_read) {
int err;

unsigned long this_chunk = (2 * 1024 * 1024) / PAGE_CACHE_SIZE;

if (this_chunk > nr_to_read)
this_chunk = nr_to_read;
err = __do_page_cache_readahead(mapping, filp,
offset, this_chunk, 0);
if (err < 0) {
ret = err;
break;
}
ret += err;
offset += this_chunk;
nr_to_read -= this_chunk;
}
return ret;
}

/*
 * Given a desired number of PAGE_CACHE_SIZE readahead pages, return a
 * sensible upper limit.
 */
unsigned long max_sane_readahead(unsigned long nr)
{
return min(nr, (node_page_state(numa_node_id(), NR_INACTIVE_FILE)
+ node_page_state(numa_node_id(), NR_FREE_PAGES)) / 2);
}

/*
 * Submit IO for the read-ahead request in file_ra_state.
 */
```

```
unsigned long ra_submit(struct file_ra_state *ra,
       struct address_space *mapping, struct file *filp)
{
int actual;

actual = __do_page_cache_readahead(mapping, filp,
ra->start, ra->size, ra->async_size);

return actual;
}

/*
 * Set the initial window size, round to next power of 2 and square
 * for small size, x 4 for medium, and x 2 for large
 * for 128k (32 page) max ra
 * 1-8 page = 32k initial, > 8 page = 128k initial
 */
static unsigned long get_init_ra_size(unsigned long size, unsigned long max)
{
unsigned long newsize = roundup_pow_of_two(size);

if (newsize <= max / 32)
newsize = newsize * 4;
else if (newsize <= max / 4)
newsize = newsize * 2;
else
newsize = max;

return newsize;
}

/*
 *  Get the previous window size, ramp it up, and
 *  return it as the new window size.
 */
static unsigned long get_next_ra_size(struct file_ra_state *ra,
unsigned long max)
{
unsigned long cur = ra->size;
unsigned long newsize;

if (cur < max / 16)
newsize = 4 * cur;
```

```
else
newsize = 2 * cur;

return min(newsize, max);
}

/*
 * On-demand readahead design.
 *
 * The fields in struct file_ra_state represent the most-recently-executed
 * readahead attempt:
 *
 *                        |<----- async_size ---------|
 *     |------------------- size -------------------->|
 *     |==================#==========================|
 *     ^start             ^page marked with PG_readahead
 *
 * To overlap application thinking time and disk I/O time, we do
 * 'readahead pipelining': Do not wait until the application consumed all
 * readahead pages and stalled on the missing page at readahead_index;
 * Instead, submit an asynchronous readahead I/O as soon as there are
 * only async_size pages left in the readahead window. Normally async_size
 * will be equal to size, for maximum pipelining.
 *
 * In interleaved sequential reads, concurrent streams on the same fd can
 * be invalidating each other's readahead state. So we flag the new readahead
 * page at (start+size-async_size) with PG_readahead, and use it as readahead
 * indicator. The flag won't be set on already cached pages, to avoid the
 * readahead-for-nothing fuss, saving pointless page cache lookups.
 *
 * prev_pos tracks the last visited byte in the _previous_ read request.
 * It should be maintained by the caller, and will be used for detecting
 * small random reads. Note that the readahead algorithm checks loosely
 * for sequential patterns. Hence interleaved reads might be served as
 * sequential ones.
 *
 * There is a special-case: if the first page which the application tries to
 * read happens to be the first page of the file, it is assumed that a linear
 * read is about to happen and the window is immediately set to the initial size
 * based on I/O request size and the max_readahead.
 *
 * The code ramps up the readahead size aggressively at first, but slow down as
 * it approaches max_readhead.
```

```
 */

/*
 * Count contiguously cached pages from @offset-1 to @offset-@max,
 * this count is a conservative estimation of
 *  - length of the sequential read sequence, or
 *  - thrashing threshold in memory tight systems
 */
static pgoff_t count_history_pages(struct address_space *mapping,
   struct file_ra_state *ra,
   pgoff_t offset, unsigned long max)
{
pgoff_t head;

rcu_read_lock();
head = radix_tree_prev_hole(&mapping->page_tree, offset - 1, max);
rcu_read_unlock();

return offset - 1 - head;
}

/*
 * page cache context based read-ahead
 */
static int try_context_readahead(struct address_space *mapping,
 struct file_ra_state *ra,
 pgoff_t offset,
 unsigned long req_size,
 unsigned long max)
{
pgoff_t size;

size = count_history_pages(mapping, ra, offset, max);

/*
 * no history pages:
 * it could be a random read
 */
if (!size)
return 0;

/*
 * starts from beginning of file:
```

```
 * it is a strong indication of long-run stream (or whole-file-read)
 */
if (size >= offset)
size *= 2;

ra->start = offset;
ra->size = get_init_ra_size(size + req_size, max);
ra->async_size = ra->size;

return 1;
}

/*
 * A minimal readahead algorithm for trivial sequential/random reads.
 */
static unsigned long
ondemand_readahead(struct address_space *mapping,
   struct file_ra_state *ra, struct file *filp,
   bool hit_readahead_marker, pgoff_t offset,
   unsigned long req_size)
{
unsigned long max = max_sane_readahead(ra->ra_pages);

/*
 * start of file
 */
if (!offset)
goto initial_readahead;

/*
 * It's the expected callback offset, assume sequential access.
 * Ramp up sizes, and push forward the readahead window.
 */
if ((offset == (ra->start + ra->size - ra->async_size) ||
     offset == (ra->start + ra->size))) {
ra->start += ra->size;
ra->size = get_next_ra_size(ra, max);
ra->async_size = ra->size;
goto readit;
}

/*
 * Hit a marked page without valid readahead state.
```

```
 * E.g. interleaved reads.
 * Query the pagecache for async_size, which normally equals to
 * readahead size. Ramp it up and use it as the new readahead size.
 */
if (hit_readahead_marker) {
pgoff_t start;

rcu_read_lock();
start = radix_tree_next_hole(&mapping->page_tree, offset+1,max);
rcu_read_unlock();

if (!start || start - offset > max)
return 0;

ra->start = start;
ra->size = start - offset; /* old async_size */
ra->size += req_size;
ra->size = get_next_ra_size(ra, max);
ra->async_size = ra->size;
goto readit;
}

/*
 * oversize read
 */
if (req_size > max)
goto initial_readahead;

/*
 * sequential cache miss
 */
if (offset - (ra->prev_pos >> PAGE_CACHE_SHIFT) <= 1UL)
goto initial_readahead;

/*
 * Query the page cache and look for the traces(cached history pages)
 * that a sequential stream would leave behind.
 */
if (try_context_readahead(mapping, ra, offset, req_size, max))
goto readit;

/*
 * standalone, small random read
```

```
 * Read as is, and do not pollute the readahead state.
 */
return __do_page_cache_readahead(mapping, filp, offset, req_size, 0);

initial_readahead:
ra->start = offset;
ra->size = get_init_ra_size(req_size, max);
ra->async_size = ra->size > req_size ? ra->size - req_size : ra->size;

readit:
/*
 * Will this read hit the readahead marker made by itself?
 * If so, trigger the readahead marker hit now, and merge
 * the resulted next readahead window into the current one.
 */
if (offset == ra->start && ra->size == ra->async_size) {
ra->async_size = get_next_ra_size(ra, max);
ra->size += ra->async_size;
}

return ra_submit(ra, mapping, filp);
}

/**
 * page_cache_sync_readahead - generic file readahead
 * @mapping: address_space which holds the pagecache and I/O vectors
 * @ra: file_ra_state which holds the readahead state
 * @filp: passed on to ->readpage() and ->readpages()
 * @offset: start offset into @mapping, in pagecache page-sized units
 * @req_size: hint: total size of the read which the caller is performing in
 *            pagecache pages
 *
 * page_cache_sync_readahead() should be called when a cache miss happened:
 * it will submit the read.  The readahead logic may decide to piggyback more
 * pages onto the read request if access patterns suggest it will improve
 * performance.
 */
void page_cache_sync_readahead(struct address_space *mapping,
       struct file_ra_state *ra, struct file *filp,
       pgoff_t offset, unsigned long req_size)
{
/* no read-ahead */
if (!ra->ra_pages)
```

```
return;

/* be dumb */
if (filp && (filp->f_mode & FMODE_RANDOM)) {
force_page_cache_readahead(mapping, filp, offset, req_size);
return;
}

/* do read-ahead */
ondemand_readahead(mapping, ra, filp, false, offset, req_size);
}
EXPORT_SYMBOL_GPL(page_cache_sync_readahead);

/**
 * page_cache_async_readahead - file readahead for marked pages
 * @mapping: address_space which holds the pagecache and I/O vectors
 * @ra: file_ra_state which holds the readahead state
 * @filp: passed on to ->readpage() and ->readpages()
 * @page: the page at @offset which has the PG_readahead flag set
 * @offset: start offset into @mapping, in pagecache page-sized units
 * @req_size: hint: total size of the read which the caller is performing in
 *            pagecache pages
 *
 * page_cache_async_readahead() should be called when a page is used which
 * has the PG_readahead flag; this is a marker to suggest that the application
 * has used up enough of the readahead window that we should start pulling in
 * more pages.
 */
void
page_cache_async_readahead(struct address_space *mapping,
   struct file_ra_state *ra, struct file *filp,
   struct page *page, pgoff_t offset,
   unsigned long req_size)
{
/* no read-ahead */
if (!ra->ra_pages)
return;

/*
 * Same bit is used for PG_readahead and PG_reclaim.
 */
if (PageWriteback(page))
return;
```

```
ClearPageReadahead(page);

/*
 * Defer asynchronous read-ahead on IO congestion.
 */
if (bdi_read_congested(mapping->backing_dev_info))
return;

/* do read-ahead */
ondemand_readahead(mapping, ra, filp, true, offset, req_size);

#ifdef CONFIG_BLOCK
/*
 * Normally the current page is !uptodate and lock_page() will be
 * immediately called to implicitly unplug the device. However this
 * is not always true for RAID conifgurations, where data arrives
 * not strictly in their submission order. In this case we need to
 * explicitly kick off the IO.
 */
if (PageUptodate(page))
blk_run_backing_dev(mapping->backing_dev_info, NULL);
#endif
}
EXPORT_SYMBOL_GPL(page_cache_async_readahead);
```