Segmented Threading to Cluster Explored Data

Mr. Rajesh K M[1], Dr. Manjunatha Rao L[2]
[1]Research Scholar, CMJ University, Meghalaya, India

Abstract

Many developers will have read Herb Sutter's article "The Free Lunch is Over" which talks about the future speed increases of CPUs. When software distributed shared memory systems provide multithreading to exploit cluster of symmetric multiprocessors (SMPs), a challenge is how to preserve memory consistency in a thread-safe way. In particular, the various method using the whole address space to the application. Hyperthreading is one which allows a single processor core to execute two threads in parallel, and also allow many threads to execute truly independently. Intel is already talking about chips with over a hundred cores. The number of possible news servers will vary from one application to another, but all programs in this class can download one or more files in one or more threads, from two or more different news-servers simultaneously. The vulnerability of boundary error when generating the "Connecting" log message for HTTP downloads is exploited to cause a stack-based buffer overflow

---------- ◆ ----------

Introduction:

Multi-thread Segmented Downloading System is a downloading tool which is used to download files from net in rapid time. It helps to download the files more faster than the existing technology. This is based on the concept of processes and threads. Here two or more threads for a single file to be downloaded is created and downloading is made. Finally integration of threads is performed. This technique helps to download the files at faster rate than existing technology. With the advance in time and technology there is a need for faster dissemination of information. Connected, personalized, intelligent information appliances are becoming increasingly important in our business and private lives. These appliances include devices such as cell phones, Computers and Internet. Although broadband Internet access is getting more and more common, people's demand for faster file downloads has never stopped. It is this desire for speed that gave birth to Multithread downloading system, which accelerate file downloads for people. In a real world scenario, such as college campus, information in the form of online magazine, e-learning tools is spread among the users.

Nowadays, the widely used internet technology and the reduction of the price of the Internet, the average computer system is possible to deliver multimedia data. Combined use of the Internet and the capabilities in computer; computers can exchange multimedia data in the real world. Multi-threading is the ability for an application to perform more than one execution simultaneously. When used properly, it can greatly improve the responsiveness and efficiency of an application. The worldwide infrastructure of computers and networks creates exciting opportunities for collecting vast amounts of data and for sharing computers and resources on an unprecedented scale Therefore this paper aims to provide a convenient environment for downloading in computer users to perform fast downloading. This program is functional, stable, and user-friendly. Our application is a most needed tool for those looking impatiently for the next download. The application is developed to manage downloads, pause and resume downloads, queue downloads, or search for downloads. This software comes with an adaptive download accelerator, dynamic file segmentation, and multipart downloading technology to considerably improve the download process.

Literature Survey:

Yang-Suk Kee a, Jin-Soo Kim b, Soonhoi Ha ; When software distributed shared memory (SDSM) systems provide multithreading to exploit cluster of symmetric multiprocessors (SMPs), a challenge is how to preserve memory consistency in a thread-safe way, which is known as ‘‘atomic page update problem’’. In this , we show that this problem can be solved by creating two independent access paths to a physical page and by assigning different access permissions to them. Especially, we propose three new methods using System V shared memory inter-process communication (IPC), a new mdup() system call, and a fork() system call in addition to a known method using file mapping. The main contribution of this is to introduce various solutions to the atomic page update problem and to compare their characteristics extensively. Experiments carried out on a Linux-based cluster of SMPs and an IBM SP Night Hawk system show that the proposed methods overcome the drawbacks of the file mapping method such as high initialization cost and buffer cache flushing overhead. In particular, the method using a fork() system call preserves the whole address space to the application.[1]

S. Williams, J. Shalf, L. Oliker, S. Kamil.; This article describes a performance comparison between different multi-core architectures while processing an image segmentation algorithm which is often used in computer vision and medical imaging. STI Cell processor, Graphic Processing Units (GPU) and SIMD architectures were benchmarked. Key bottlenecks of the application have been evaluated for each architecture showing which kind of approach best fits each one of them. GPU implementation was made through fragment pipeline in a data-parallel paradigm using NVidia's Cg language. Cell's implementation used SIMD, parallelism and double-buffering techniques. Mean segmentation times for a

256x256 RGB were: 3.3 ms in a CPU, 2.4 ms in a CPU with SIMD extensions, 1.0 ms in a GPU (8 pixel shaders), 0.87 ms in a PS3 Cell processor (6 SPE's) and 0.4 ms in another GPU (32 pixel shaders), which encourages the adoption of parallel approaches in this application. The main bottleneck has been identified as memory transfers.[2]

Greg Beech; Many developers will have read Herb Sutter's article "The Free Lunch is Over" which talks about the future speed increases of CPUs. The good news is that they will get significantly faster, but the bad news is that you won't see all of the possible performance gains unless you write your application to take advantage of them. Over the last few years the increase in clock speeds has slowed down, and chip manufacturers are focussing more and more on concurrent execution of code. Hyperthreading was the first step, which allows a single processor core to execute two threads in parallel, but the future is multi-core chips which will allow many threads to execute truly independently. Intel is already talking about chips with over a hundred cores, so if your code is single-threaded you may only be using one hundredth of the available processing power! This Charteris White introduces the fundamentals that will allow you to begin writing concurrent applications, and is intended to be easier to read than much of the material on this subject. It is targeted at developers who have some experience of developing on the .NET Framework, and ideally in C# as this is my language of choice for the code samples. No prior knowledge of concurrent programming or multi-threading is needed, though even experienced developers in this area may find some new and surprising information.[3]

Johan Powelse; Even though many P2P file-sharing systems have been proposed and implemented, only very few have stood the test of intensive daily use by a very large user community. The BitTorrent file-sharing system is one of these systems. Measurements on Internet backbones indicate that BitTorrent has evolved into one of the most popular networks. In fact, BitTorrent traffic made up 53 per cent of all P2P traffic in June 2004. As BitTorrent is only a file-download protocol, it relies on other (global) components, such as websites, for finding files. The most popular website for this purpose is suprnova.org. There are different aspects that are important for the acceptance of a P2P system by a large user community. First, such a system should have a high availability. Secondly, users should (almost) always receive a good version of the content they request (no fake files). Thirdly, the system should be able to deal with flashcards. Finally, users should obtain a relatively high download speed. In this we present a detailed measurement study of the combination of BitTorrent and Suprnova. This measurements study addresses all four mentioned aspects. Our measurement data consist of detailed traces gathered over a period of 8 months (Jun'03 to Mar'04) of more than two thousand global components. In addition, for one of the most popular files we followed all 90,155 downloading peers from the injection of the file until its disappearance (several months). In a period of two weeks we measured the bandwidth of 54,845 peers downloading over a hundred newly injected files. This makes our measurement effort one of the largest ever conducted.[4]

J. D. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Krüger; This means that more than one file can be downloaded at one time, not *sequentially,* but in *parallel.* Different applications in this class have differing capabilities in that regard, For example: NewsBin Pro Version 4.x/5.x has the ability to download as many as ten (10) files simultaneously. Forté Agent (to v3.2) is multi-threaded because it is capable of downloading two files simultaneously, from the same server. See the Agent Setup page to learn how to make it do that. Many people report a significant increase in the overall download speed when a multi-threaded download is taking place. Here you can enter more than one news-server from which to download both headers and files, and that the client will use multiple news-servers at the same time. The number of possible news servers will vary from one application to another, but all programs in this class can download one or more files in one or more threads, from two or more different news-servers simultaneously. The practical advantages of this multi-threaded multi-server approach are several: By using multiple threads it is often

possible to obtain better overall download speeds than with a single thread. By using multiple news servers the application can obtain the various parts of files from different servers and combine them. A file which has 8 of 9 parts complete on server "A", may have only 4 of 9 parts complete on server "B", but may have the part that is missing on server "A", this allows you to download a complete file without needing a fill.[5]

J.P. Farrugia, P. Horain, E. Guehenneux, and Y. Alusse ;

"Orbit Downloader, leader of download manager revolution, is devoted to new generation web (web2.0) downloading, such as video/music/ streaming media from Myspace, YouTube, Imeem, Pandora, Rapidshare, support RTMP and to make general downloading easier and faster". Secunia Research has discovered a vulnerability in Orbit Downloader, which can be exploited by malicious people to compromise a user's system. The vulnerability is caused due to a boundary error when generating the "Connecting" log message for HTTP downloads. This can be exploited to cause a stack-based buffer overflow by e.g. tricking a user into downloading from a malicious HTTP server or opening a specially crafted HTTP URL containing an overly long host name. Successful exploitation allows execution of arbitrary code.[6]

Methodology

This application is a most needed tool for those looking impatiently for the next download. The application is developed to manage downloads, pause and resume downloads, queue downloads, or search for downloads. This software comes with an adaptive download accelerator, dynamic file segmentation, and multipart downloading technology to considerably improve the download process. This application is a simple and reliable download manager with excellent features like, a) segmented downloads for acceleration b) simultaneous downloads c) Supersonic increase where download speed by finding multiple sources. d) Accelerates downloads by up to 30% by using multiple download threads. e) Can resume broken download.

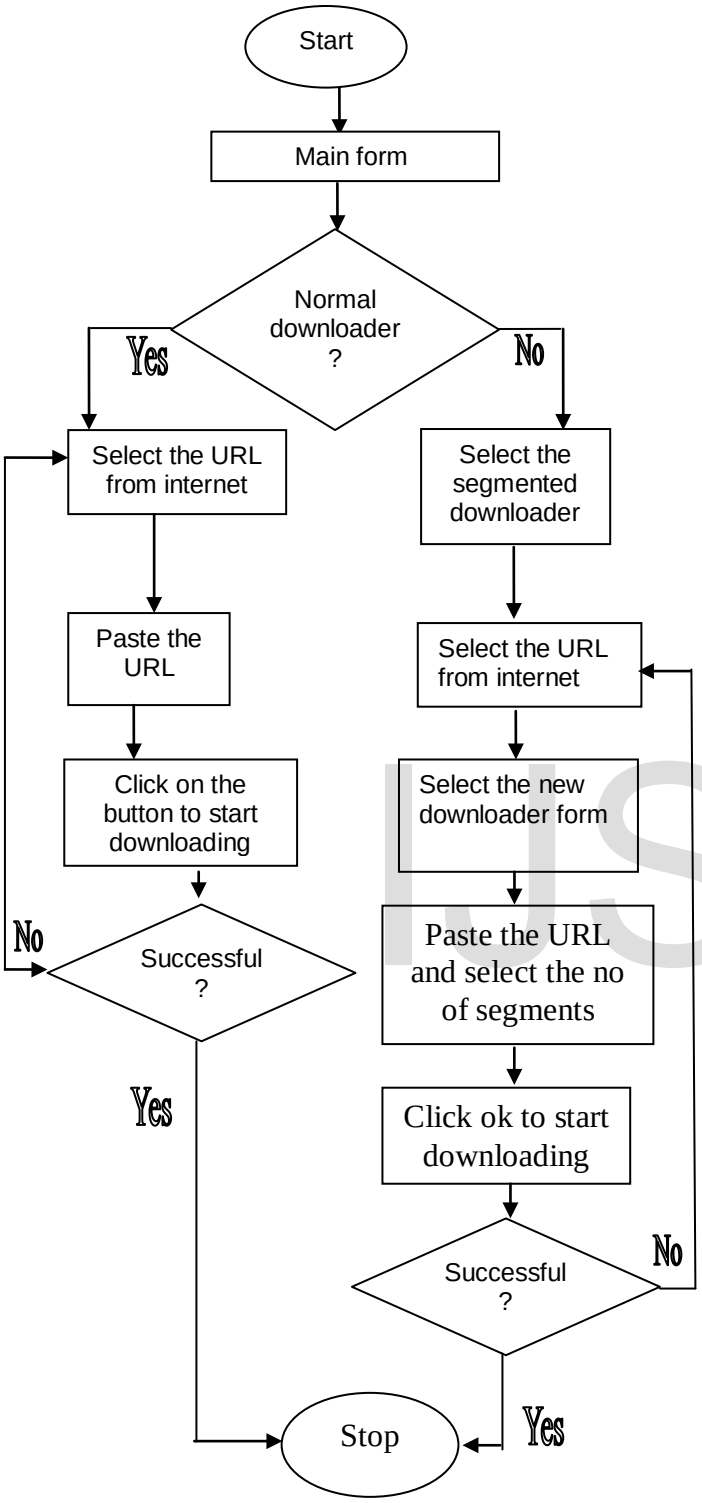
To accelerate downloads, this application splits a large file into multiple segments and simultaneously downloads as many segments as possible within some constraints such as were available network bandwidth, any speed limits that we have set (explained below) and server side controls that may exist occasionally during peak demands. This application is enabled by default and is set to make the most of wer available bandwidth. If we are in an environment where we need to share bandwidth with others and would like to set a limit on the bandwidth consumed or disables the download acceleration altogether, we can do so from dialog.

File Validation after Download; This application validates the file integrity after downloading each file, before considering the download as complete. The progress of this file validation is displayed on the FTM status bar. In a transfer that consists of multiple files, the validation occurs after each file is downloaded before the download of next file begins. Downloading files over the network could be a cumbersome task. Trying to implement other features like auto-resume and support for proxy servers, etc. makes it even harder. Using this component can make things a lot easier. Anyone can download files over proxies, password protected sites or entrusted SSL connections simply by setting a few properties. The example that is provided shows how to do the job and how to handle events over a simple GUI. Since the download operation takes place in a worker thread asynchronously, we don't need to worry about downloading large files or blocking the user interface.

Stopping a download operation that is running is tricky. We should call the DownloadThread.Abort() method on the downloader thread to stop the work, but this is not the recommended way. The main reason for this is the Thread.Abort throws an exception which can occur at any time during where work and leaves where running state inconsistent. The easier way is to check for a Boolean field and terminate the operation when the field is set, or run the job on an entirely different process. To accelerate downloads, this application splits a large file into multiple segments and simultaneously downloads as many segments as possible within some constraints such as were available network bandwidth, any speed limits that we have set (explained below) and server side

controls that may exist occasionally during peak demands.

Flow chart:



Flow chart of Multi Thread Segmented Downloading System

Whenever an end-user wants to download file or application from the internet, the first page that appears to the end-user is the Home page. Here two options will be displayed i.e,

- Normal download
- Segmented download

Whenever the user selects normal download the entire file or application gets downloaded at a stretch and finally the downloading ends.

Suppose the user selects segmented download then the end-user will be asked to enter the number of segments. Based on the number of segments the entire file gets segmented and all these segments gets downloaded simultaneously. At last the integration of these segments takes place and finally the downloading ends.

Design of Start page: To design the start page a Picture box and a Button is used. It has been designed such that on clicking the button the end-user enters into the page containing two options one is for normal downloading and the other is for segmented downloading.

Browse page: This page is used to browse the net in order to get the URL of the file or an application that needs to be downloaded.

Design of Normal downloader form: To design the normal downloader form a text box and a button has been used. The URL that has been searched from the browse page is pasted in the text box, then on clicking the button beside the text box the downloading starts. The progress of the downloading is indicated in terms of percentage, which is displayed beside the file name that is being downloaded.

Implementation:

The basic implementation has been done in prototype development style. Where at each stage, a prototype for the required problem is built and tested for the sample inputs if the prototype is giving the correct

output then this prototype is embedded with the system with proper interfaces built to it.

The prototype development has many advantages they are, It is a rapid development process

Even the additional requirements can be got from the prototype designed. If there are any differences in the requirement specification and the actual system required then it can be sorted out at very early stages of the software development process. The change in one particular prototype inserted in the system will not cause the change in the whole system, because the interface to this prototype will be same as before. It can be known whether the system requirement specifications can be satisfied or not at the initial stages of the development process.

Creating Threads: The starting point for such a thread is void method with no parameters. Thread creation is done in two steps: a) Create a delegate object, initialized with our method. b) Use this object as an initialization parameter when creating a new Thread object.

Segmentation of the file to be downloaded: Downloads can be segmented because both HTTP and FTP protocols allow the client to specify the start position of the stream. First, application performs a request to the server to discover the file size.

After that, for calculates the segment we use

Segment size = min ((file size / number of segments), Minimum allowed segment size)

With the segment size, application creates another request specifying the start position of the stream. In this way, we can have multi-requests for the same files running in parallel using multi-threading techniques. This technique speeds up the transfer rate even more if we are using mirrors.

Programmatically starting download pausing and then restarting again:

1. Typically send a RETRIEVE (RETR) command for file to start download. Store data received from server to a temp file for example *Temp_File.txt*. So any time download is aborted, size of temp file is the actual useful data transferred so far of *File.txt*.

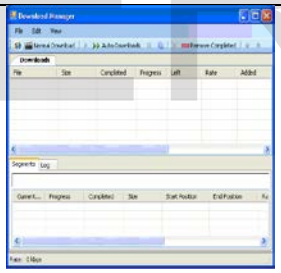
2. If all of a sudden ISP connection is broken or server is down or manually stopped the download, next time when we want to restart download, read the size of temp file (i.e. useful data downloaded so far), and seek to the end of the temp file so further download will append the data from end.

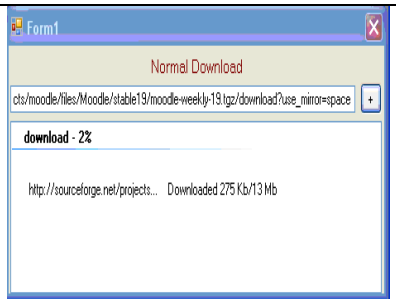
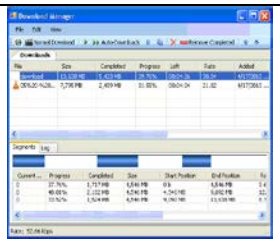
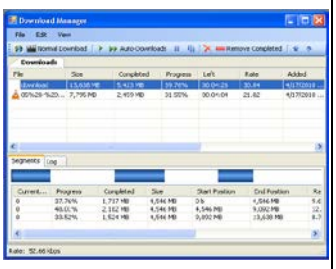
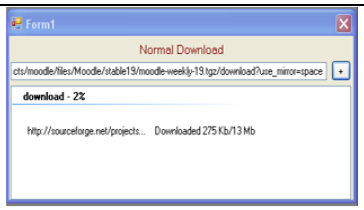
3. Now pass the file size of temp file as parameter, for example the size of the file is 250 bytes. Server will send a response stating "Restarting from byte 250 bytes.".

4. Now we can send RETRIEVE (RETR) again for file *File.txt* and server will start sending data from byte offset 251 onwards.

5. Once the transfer is complete rename the temp file to the actual file name on WINDOWS using C# it should typically look like RenameFile(Temp_File.txt, File.txt); Done.

Results:

Outcome	Description
	The Start page of Multi Thread Segmented Downloading System. On Clicking Enter Button the Segmented Downloader Form is opened.
	The Segmented Downloader Form consists of Menu strip and Tool strip which consists of certain options required for downloading. It also shows the segments and log files.

	This snapshot shows the browse page which appears on clicking the browser tool strip tab. This allows us to surf the internet for the required application that is of interest, in order to download
	This snapshot shows the normal download form which appears on clicking the normal download option. The normal downloader which downloads the applications from the internet using the existing technology.
	This snapshot shows the new download form. The new download form consists of the text-box's in order to paste the URL, select the target location to save the downloaded file in the local system. It allows the user to specify the number of segments.
	This snapshot shows the Download Manager form. The Download Manager form shows the progress of downloading a particular application from the internet
	Comparing segmented and normal downloader's.

Conclusion

The proposes a new method for downloading files from the internet. While the application utilizes the maximum resources of bandwidth while downloading. We have implemented basic structure of a multithreading downloading system. It is developed using c#. The implemented application download the files much faster than the existing system and supports lot of features which is not supported by the existing system. This application is a simple and reliable download manager with excellent features like, Segmented downloads for acceleration. Simultaneous downloads .Supersonic increase was download speed by finding multiple sources. Accelerates downloads by up to 30% by using multiple download threads. Can resume broken downloads

References:

1. Yang-Suk Kee, Jin-Soo Kim and Soonhoi Ha: Memory Management for Multi-Threaded Software DSM Systems. 26 August 2003

2. Amir Ehsan Khandani, Jinane Abounadi, Eytan Modiano, and Lizhong Zheng: "Cooperative Routing in Static Wireless Networks". IEEE

Transactions on Communications 55(11) : 2185-2192 (2007)

3. Shoaib Kamilyz, Cy Chany?, Leonid Olikery, John Shalfy, Samuel WilliamsyyCRD/NERSC, “An Auto-Tuning Framework for Parallel Multicore Stencil omputations”, Lawrence Berkeley National Laboratory Berkeley, CA 94720 zEECS Department, University of California at Berkeley, Berkeley, CA 94720 ?CSAIL, Massachusetts Institute of Technology, Cambridge, MA 02139

4. Greg Beech, Charteris White Paper Fundamentals of Concurrent Programming for .NET, 18 March 2005

5. J.A. Pouwelse, P. Garbacki, D.H.J. Epema, H.J. Sips, "The Bittorrent P2P File-sharing System: Measurements and Analysis", 4th International Workshop on Peer-to-Peer Systems (IPTPS'05), Feb 2005, (767KB .pdf file) {6 page version of technical report with improved analysis} [Chineese translation thanks to Lee Little]

6. JD Owens, D Luebke, N Govindaraju, M Harris, J Krüger, AE Lefohn, TJ Purcell Computer graphics forum 26 (1), 80-113 “A Survey of General-Purpose Computation on Graphics Hardware”

7. J. D. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Krüger, A. E. Lefohn, and T. J. Purcell. “A Survey of General-Purpose Computation on Graphics Hardware” Computer Graphics Forum 26, 1 (2007), 80 – 113.

8. J.P. Farrugia, P. Horain, E. Guehenneux, and Y. Alusse. “GPUCV: A Framework for Image Processing Acceleration with Graphics Processors.” Multimedia and Expo, 2006 IEEE International Conference On, (July, 2006), 585 – 588