ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Special Issue 6, July 2017

Data Extraction using Chunking approach on Decentralized Mixture Network

Saurabh Walhekar, Pooja Gawade, Ashwini Kad, Suryakant Tripathi.

Department of Computer Engineering, SCOE Sudumbare, Maharashtra, India

ABSTRACT: - In this paper, we develop an efficient file sharing system based on a mixture distribution model working in the BitTorrent like p2p networks. The Bit-Torrents built-in unchunking mechanism delays the initial file sharing process for newly joined peers as well as brings the problem of free-riding that peers only download from others but never contribute to the network. We demonstrate a file sharing mechanism for BitTorrent. File will be shared in chunks using BSW(Basic Sliding Window) algorithm also providing the support of Multi-threading which allows load balancing. By multi-threading we will be exploiting all the available CPU cores to gain the X4(in case of good core) increase in the chunking performance. In dense distributed computing condition, we will be using RR algorithm to avoid the failure of the TCP clocking mechanism in case of large no of request. RR algorithm will switch the service providing between the active and the inactive users. The active user will be given top priority while the inactive user will also done the same if they become active else not.
General Terms- Data Distribution, Chunking, BSW

KEYWORDS: P2P, Mixture distribution, File sharing mechanism, BitTorrent, Free-riding, Multithread, Content-based chunking, TCP download performance

I. INTRODUCTION

File Sharing is the one of the most widely used concept in today's digital world. Associated to the File Sharing are also the Peer-to-Peer (p2p) applications. Among all the p2p applications, file sharing is the most popular one. The traditional client/server file sharing lacks the few key benefits when compared to the Peer-to-Peer file sharing, one of which is Scalability. On facing the increase in the number of clients the performance of the traditional client/server file sharing decreases rapidly. In case of well-built Peer-to-Peer File Sharing System the condition is well handled, since the load is distributed to all participating peers.

BitTorrent is the main application for the Peer-to-Peer File Sharing. BitTorrent is a popular peer-to-peer _le-sharing protocol that has been shown to scale well to very large peer populations. With BitTorrent, content (e.g., a set of _les) is split into many small pieces, each of which may be downloaded from different peers. The content and the set of peers distributing it are usually called a *torrent*. A peer that only uploads content is called a *seed* (Peers that have the whole file), while a peer that uploads and downloads at the same time is called a *leecher*. The connected set of peers participating in the piece exchanges of a torrent is referred to as a *swarm*. In a BitTorrent network, a single file is shared by many users. The main feature of BitTorrent is that the shared file is divided into many pieces (the default size of a piece is 256KB), so a peer could start to serve others even if it does not have a complete file. While other peers with partial or none of the file is called downloaders, when a peer first joins the network, it connects to a central server called tracker to get a list of peers. The new peer then connects to those peers to request for pieces and those peers become the neighbors of the new peer. Once the new peer obtains at least one piece, it can start to contribute to the network uploading to others. The peer then exchanges pieces with its neighbors until it obtains all pieces and becomes a seed. The distributed property of file sharing process makes the BitTorrent scalable.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Special Issue 6, July 2017

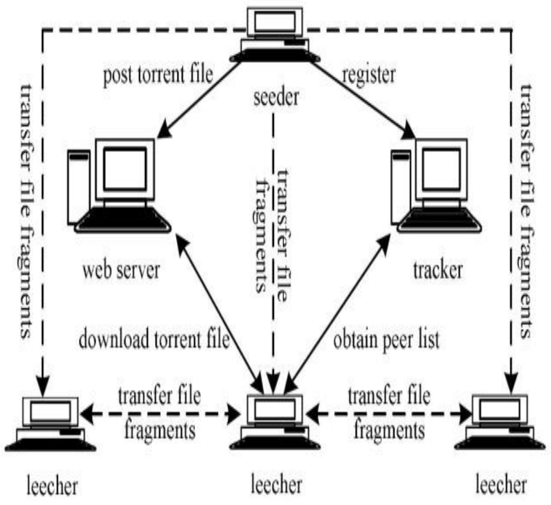


Figure 1: BitTorrent Overview

The goal of this paper is to solve the inefficiency caused in Bit-Torrent like P2P networks, one of such inefficiency is the performance depletion by randomly generated load which is not balanced very well and further leads to scalability issue.This paper mainly designed a mixture distribution model for P2P networks and a file sharing mechanism that handles the randomly generated load efficiently

II. RELATED WORK

The basic concept of P2P file sharing network is that the peers takes part in an application level overlay network and contribute their content in a cooperative manner. For BitTorrent, there is a noticeable result in many aspects, including design of system, measurement, and improvement of performance, and incentive mechanisms. Indeed, many works involve the modeling of BitTorrent-like networks in varying degrees of complexity. Over the few years, the branching process for BitTorrent has given special importance on the performance modeling and measurement. Markov chain model is always considered to study the steady state of the network. Properties like scalability to sustain a large number of peers, efficiency in sharing a file and robustness against oscillation of the network are studied
in these measurements. A stochastic model is applied in to concern the variability of the peers in the network when incorporated with realistic scenarios like the peer appearance, removal and the upload/download rates. The study also proves that the mechanism built into the BitTorrent can encourage peers to upload to some degree. In the starting stage of the file sharing process is concerned to study the evolution of peers. The fairness problem has always been an issue for p2p file sharing network. Most p2p networks try to build in some incentives to deter peers from free-riding. For Bit-Torrent, the built-in file sharing mechanisms which apply the TFT strategy have not solved the problem ideally the existence of the problem further degrades the survivability of the network. Develops a model to represent the different stages of peers in which each stage corresponding to the completeness of a download job. The author also proposes a new TFT strategy to encourage the peers to stay longer which finally prolongs the lifetime of the network. In, a game-theoretic framework for understanding the conflict and cooperation between peers is presented to deter free-riders. In , it is shown that the TFT incentive mechanism is generally not robust to strategic clients. A strategic client can take advantage of TFT and hurt the performance of other peers.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

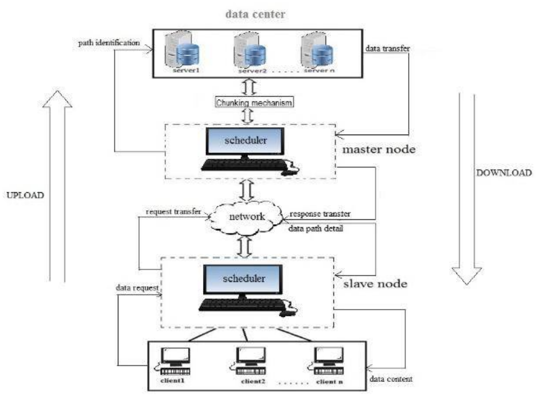
International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Special Issue 6, July 2017

III. SYSTEM ARCHITECTURE



1. System Model

A. Architecture

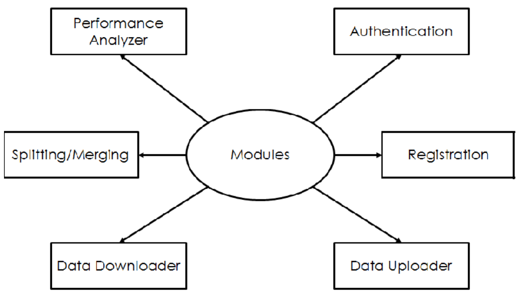


Figure 2: Chunking Level/Type

B. Chunking

Chunking is a process to partition entire file into small pieces of chunks. For any data de-duplication system, chunking is the most time consuming processes since it has to traverse entire file without any exception. The process time of chunking totally depends on how the chunking algorithms break a file. Moreover, the smaller the size of a chunk has the better result a de-duplication system has. Increasing the number of chunks, however, results in increasing the processing time for both schemes which we presented in previous section. For the hash-based de-duplication systems, increasing the number of chunk also means increasing the size of lookup table, and then the systems need to spend more time to perform the comparisons. The worst case is that the systems cannot load entire lookup table into memory when the size of the lookup table becomes very huge. In this case, the systems have to pay most expensive costs in disk I/O. The content-aware systems face the similar tradeoffs while performing the block-to-block comparisons. In other words, a good chunking algorithm has to satisfy certain conditions such as minimizing the processing time, balancing the scalability and de-duplication ratio, and controlling the variations of chunk-size.
According to how to break a file, there are three different chunking categories as shown in Figure 3.

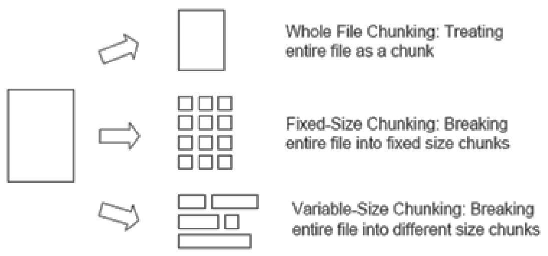

Whole File Chunking: Treating entire file as a chunk
Fixed-Size Chunking: Breaking entire file into fixed size chunks
Variable-Size Chunking: Breaking entire file into different size chunks

Figure 3: Three Different Chunking Categories.

These categories have been studied and researched. Generally speaking, whole-file chunking is the simplest and fastest, but it has the worst results regarding de-duplication ratio. The de-duplication ratio of the fixed-size chunking is totally depending on what the fixed-size is. The smaller the fixed size is, the better de-duplication ratio has. Again, the fixed-size chunking faces the tradeoffs in balancing the capacity scalability and the de-duplication ratio.

C. Mixture Distribution Model

Note that the shared file is divided into many pieces and each peer can serve others when it has partial pieces of the file regardless of how many portions it owned. In the BitTorrent network, the pieces of a file can be regarded as a uniform distribution where the probabilities to obtain each piece are the same. In our model, we distribute each piece selection with different probabilities

Beta distribution is a family of continuous probability distributions defined on the interval, which represents a continuous random tendency and ensures that the total probability integrates to 1. The distributions, like uniform distribution, linear distribution, are the special cases of beta distribution. Due to these properties, we apply Beta distribution
to our model.

$$\text{Beta}(a, b) = \frac{(\ +\ )}{(\ )\ (\ )} -1(1 - \cdot)^{\cdot} -1 \quad (1)$$

Where Γ(x) is the gamma function:

$$\Gamma(x) = \int_0^\infty \quad ^{-1} \ ^{-} \cdot \cdot$$

The parameters *a* and *b* control the distribution of *x*. To simplify the model, we use the mean *c* and variance *v* of the function to substitute *a* and *b*. Then the function can be presented as *B(c,v)*, where *c* controls the “center” of the distribution and *v* controls how “fat” the distribution is. Meanwhile, the *v* also represents the uniqueness of a file.

In the model, for a specific file *k*, it has a unique distribution $B(c,v_k)$across the interval [0, 1]. Once a file is published in the network, the unique distribution of the file is generated. However, the distribution $B(c,v_k)$ should not be a uniform style, meanwhile, the distribution should be assigned smoothly to make sure the pieces with least probabilities are not approaching to 0. Reasons on this will be shown in section IV. We concentrate on the model and mechanism in sharing the file between peers. Typically, we demonstrate them with the setting of one file.

D. The Basic Sliding Window (BSW) Algorithm

The BSW algorithm was proposed for a low bandwidth network file (LBFS) system.

The Basic Sliding Window Algorithm [3] avoids boundary shifting by making chunk boundaries dependent on the local contents of the local contents of file, rather than distance from the beginning of the file.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Special Issue 6, July 2017

The main purpose of this algorithm is to reduce the network bandwidth requirements by avoiding the boundary shifting problem. In following sections, we discuss the concepts of BSW algorithm

E. *High Performance Computing (HPC)*

"Multicore System of HPC framework" technique improves the performance of data distribution algorithm which uses sequential processing they are modified using the theory of parallel processing. Using the HPC there is chance of performance improvement of chunking algorithm by using Parallel Processing.
High Performance Computing (HPC)

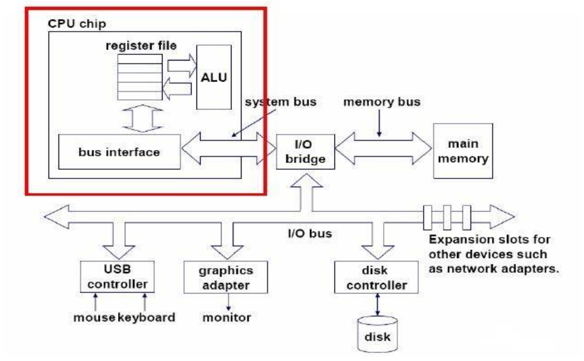
High-performance computing (HPC) is the use of super computers and parallel processing techniques for solving complex computational problems. HPC technology focuses on developing parallel processing algorithms and systems by incorporating both administration and parallel computational techniques.

High-performance computing is typically used for solving advanced problems and performing research activities through computer modeling, simulation and analysis. HPC systems have the ability to deliver sustained performance through the concurrent use of computing resources.

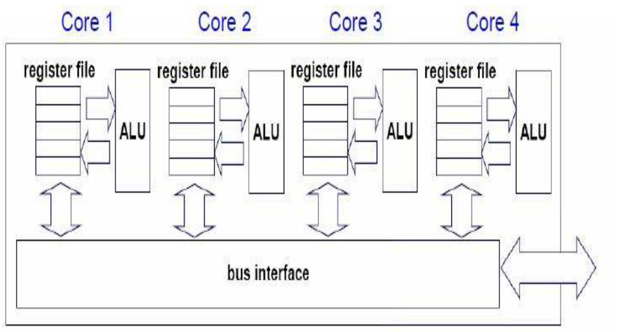
The terms high-performance computing and supercomputing are sometimes used interchangeably.

F. *Multi core*

Single Core Architecture:



Multi-Core Architecture:



In Multi-Core instead of using a single processor 2 or more processors are assigned. They sound similar to parallel processors but differ from them as they are integrated on same chip circuit. A multi core processor implement message passing or shared memory inter core communication methods in multiprocessing. If the number of threads are less than or equal to the number of cores, separate core is provided to each thread and threads run independently on multiple cores. (Figure 1) If the number of threads is more than the number of cores, the cores are distributed among the threads. Any application that can be threaded can be mapped effortlessly to multi-core, but the improvement in performance gained by the usage of multi core processors depends on the portion of the program that can be parallelized.

2. Data Distribution Mechanism

In the file sharing network, each peer is responsible for attempting to maximize its own download rate and obtain new pieces quickly. In BitTorrent, peers achieve this by downloading from whoever they can and deciding which peers to upload to via the TFT strategy. Our system differs from BitTorrent in which peers can only make a request to others for downloading.
Meanwhile they should contribute to others according to the choking algorithm, so that they can upgrade their contribution coefficient to help them get new pieces. We propose the file sharing mechanism which includes the choking algorithm for peers to share file and the peer selection algorithm to decide which peers to unchoke.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Special Issue 6, July 2017

I. Boundary Shifting Problem

Both whole-file chunking and fixed-size chunking face the boundary shifting problem due to the data modifications. When users only insert or delete one byte, the whole file chunking will result in two different hash values between the modified file and the original file, even if most of the data remain unchanged.

In same situation, after one-byte modification happens, the fixed-size chunking will generate totally different results for all the subsequent chunks even though most of the data in the file areunchanged. This problem is called as the boundary shifting problem. We show anexample in Figure 4. In Figure 4, (a) shows the case when we break entire file intofixed-size chunks, and (b) shows after inserting 2 bytes string - "*12*" in the *c2* position ofthe original file, we obtain the totally different results, even other data remain the same.

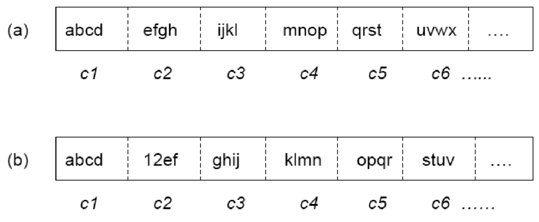


Figure 4: A Boundary Shifting Problem Example.
(a) The Original File.
(b) The File after Insertion.

Variable-size chunking is also called content-based chunking . In other words, chunking algorithms determine chunk boundaries depending on the content of the file. For example, content-based chunking algorithm may determine chunk boundaries by punctuation, each line, or every paragraph. In contrast with fixed-size chunking algorithms which determine chunk boundaries by the distance from the beginning of the file. Therefore when data modifications happen, most of the chunks remain unchanged. This is why variable-size chunking can avoid the boundary shifting problem .

II. Concept of the BSW Algorithm

In the BSW algorithm, there are three main parameters needed to be pre-configured

1. A fixed size of window *W*
2. an integer divisor –*D*
3. an integer remainder - *R*,

Where *R* <*D*. We describe how the BSW algorithm works as follows:

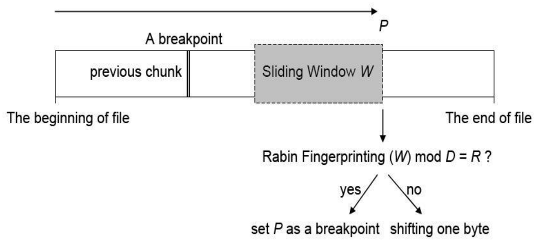


Figure 5: Concept of the BSW Algorithm.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Special Issue 6, July 2017

1. A fixed-size window *W* is shifting one byte at one time from the beginning of the file to end of the file.
2. At every position *p*, uses Rabin Fingerprinting algorithm to compute a hash value *h* for the content of current window.
3. If *h* mod *D* = *R*, the position *P* is a breakpoint for chunk boundary. Then the sliding window *W* starts at the position *P*and repeats the computation and comparison.
4. If *h* mod *D* ≠ *R*, the sliding window *W* keeps shifting one byte. And repeats the computation and comparison

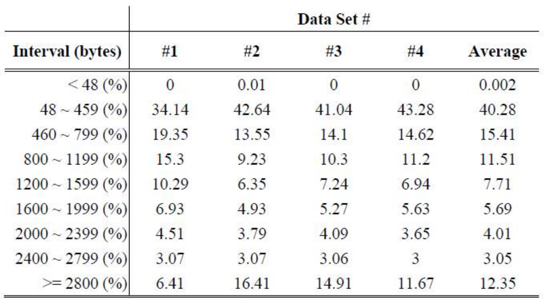
Since the BSW algorithm determines the chunk boundaries by the content of file, this approach has been proved to be success to avoid the boundary shifting problem

III. The Expected Chunk-Size

In practice, the parameter *D* plays the most important role in the BSW algorithm because it can be configured to make the chunk-size close to our expectancy. Since any integer divided by *D*, the remainder is between 0 and *D* – 1. The window shifts one byte at one time. In each shifting, the probability of *h mod D* = *R* is 1/*D*. In other words, we expect to find a breakpoint for chunk boundary at every *D* bytes. For example, if we expect the size of every chunk equals to1000 bytes, then we set the value of *D* as 1000 and the value of *R* is any integer where 0 ≦*R* ≦ 999. In the best case, we can always expect to find a matching for *h* mod *D* = *R* in every 1000 shifts, that is, 1000 bytes.

IV. Chunk-Size Distributions

In the last part of comparison, we consider the chunk-size distribution for two algorithms. According to the features of two algorithms, we take the chunk-size which is from 0 to 48bytes (window size) as the first interval for the BSW algorithm, and from 0 to 460 bytes (minimum threshold) as the first interval for TTTD algorithm. After that, we simply increase 400 bytes for each interval to group our statistical data.



	Data Set #				
Interval (bytes)	#1	#2	#3	#4	Average
< 48 (%)	0	0.01	0	0	0.002
48 ~ 459 (%)	34.14	42.64	41.04	43.28	40.28
460 ~ 799 (%)	19.35	13.55	14.1	14.62	15.41
800 ~ 1199 (%)	15.3	9.23	10.3	11.2	11.51
1200 ~ 1599 (%)	10.29	6.35	7.24	6.94	7.71
1600 ~ 1999 (%)	6.93	4.93	5.27	5.63	5.69
2000 ~ 2399 (%)	4.51	3.79	4.09	3.65	4.01
2400 ~ 2799 (%)	3.07	3.07	3.06	3	3.05
>= 2800 (%)	6.41	16.41	14.91	11.67	12.35

fig: Chunk-Size Distributions of the BSW Algorithm.

IV. EXPERIMENTAL RESULT

We are considering three databases which are DB1,DB2,DB3:
The Dataset contain various types of file which are distributed among three database DB1,DB2,DB3 are as follows :
DB1 contains 1-6 type of file with their type & size
DB2 contains 7-15 type of file with their type & size
DB3 contains 16-20 type of file with their type & size

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Special Issue 6, July 2017

Dataset Table are as follow:

Documentation	**Type**	**Size**
MS-Word	.doc	3.20 MB
MS-Access	.accdb	7.10 MB
Adobe Reader	.pdf	8 MB
Application	.exe	28.70 MB
Notepad	.txt	2.3 MB
Image	.jpeg	5 MB
Video	.mp4	60 MB
Audio	.mp3	8 MB
MS power-point	.ppt	3.40 MB
MS-Excel	.xlt	6 MB
Photoshop	.psd	170 MB
Latex	.tex	0.15 MB
Torrent file	.torrent	0.51 MB
Compress file	.zip	51.30 MB

Fig: Dataset Table

We using 3 node as server,client1,client2 Specifications of nodes are as follows:

	Node-1 (Server)	Node-2 (Client1)	Node-3 (Client2)
Processor	i5-6th Gen	i3-5th Gen	i3-4th gen
RAM	8GB	4GB	4GB
Speed	2.24GHz	2.00GHz	2.40GHz
HardDisk	1 TB	1 TB	1TB
Java	Java 7	Java 7	Java 7
OS	Windows 10	Windows 10	Windows 7
Bandwidth	100mbps	100mbps	100mbps

Fig: Specifications

V. CONCLUSION

The BitTorrents built-in tit-for-tat unchoking mechanism delays the initial file sharing process for newly joined peers as well as brings the problem of free riding that peers only download from others but never contribute to the network. To solve these problems, we first develop a mixture distribution model that the pieces of a file are assigned with different distributions for different peers. Then we demonstrate a file sharing mechanism utilizing the historical contributions of peers which allows peers to share pieces according to different mixture distributions.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Special Issue 6, July 2017

REFERENCES

1. Wu J Q, Peng Y X. Speeding up Data Distribution of BitTorrent-Like P2P Systems[J]. Advanced Materials Research, 2011, 268: 2201-2207.
2. Ye L, Zhang H, Li F, et al. A measurement study on BitTorrentsystem[J]. Int'l J. of Communications, Network and System Sciences, 2010, 3(12): 916.
3. Dn G, Carlsson N. Dynamic swarm management for improved BitTorrent per-formance[C]//IPTPS. 2009, 9.
4. Levin D, LaCurts K, Spring N, et al. Bittorrent is an auction: analyzing and improving bittorrent's incentives[C]//ACM SIGCOMM Computer Communi-cation Review. ACM, 2008, 38(4): 243-254.
5. Wang M, Zhang Y, Meng X. A reputation based incentive mechanism for selfish BitTorrent system[C]//Global Communications Conference (GLOBE-COM), 2013 IEEE. IEEE, 2013: 1348-1353.