



# INTERNATIONAL JOURNAL OF COMPUTERS AND THEIR APPLICATIONS

---

## TABLE OF CONTENTS

	Page
<b>Guest Editorial: Special Issue from ISCA Fall-2016 Conference</b> .....	207
<i>Gongzhu Hu and Takaaki Goto</i>	
<b>Evaluating an Array of Heterogeneous Disks</b> .....	208
<i>Andras Fekete and Elizabeth Varki</i>	
<b>Evaluation and Generalization of Trust Modes in P2P Networks</b> .....	216
<i>Wei Li</i>	
<b>Chinese Characters Ontology and Induced Distance Metrics</b> .....	223
<i>Antoine Bossard and Keiichi Kaneko</i>	
<b>Data Lossless Compression Using Improved GFC Algorithm with Multiple GPUs</b> .....	232
<i>Rui Wu, Muhanna Muhanna, Sergiu M. Dascalu, Lee Barford, Frederic C. Harris, Jr.</i>	
<b>Index</b> .....	242

\* "International Journal of Computers and Their Applications is abstracted and indexed in INSPEC and Scopus."

# International Journal of Computers and Their Applications

*A publication of the International Society for Computers and Their Applications*

## EDITOR-IN-CHIEF

Dr. Frederick C. Harris, Jr., Professor  
Department of Computer Science and Engineering  
University of Nevada, Reno, NV 89557, USA  
Phone: 775-784-6571, Fax: 775-784-1877  
Email: Fred.Harris@cse.unr.edu, Web: <http://www.cse.unr.edu/~fredh>

## ASSOCIATE EDITORS

**Dr. Hisham Al-Mubaid**  
University of Houston-Clear Lake,  
USA  
[hisham@uhcl.edu](mailto:hisham@uhcl.edu)

**Dr. Antoine Bossard**  
Advanced Institute of Industrial  
Technology, Tokyo, Japan  
[abossard@aiit.ac.jp](mailto:abossard@aiit.ac.jp)

**Dr. Mark Burgin**  
University of California,  
Los Angeles, USA  
[mburgin@math.ucla.edu](mailto:mburgin@math.ucla.edu)

**Dr. Sergiu Dascalu**  
University of Nevada, USA  
[dascalus@cse.unr.edu](mailto:dascalus@cse.unr.edu)

**Dr. Sami Fadali**  
University of Nevada, USA  
[fadali@ieee.org](mailto:fadali@ieee.org)

**Dr. Vic Grout**  
Glyndŵr University,  
Wrexham, UK  
[v.grout@glyndwr.ac.uk](mailto:v.grout@glyndwr.ac.uk)

**Dr. Yi Maggie Guo**  
University of Michigan,  
Dearborn, USA  
[magyigu@umich.edu](mailto:magyigu@umich.edu)

**Dr. Wen-Chi Hou**  
Southern Illinois University, USA  
[hou@cs.siu.edu](mailto:hou@cs.siu.edu)

**Dr. Ramesh K. Karne**  
Towson University, USA  
[rkarne@towson.edu](mailto:rkarne@towson.edu)

**Dr. Bruce M. McMillin**  
Missouri University of Science and  
Technology, USA  
[ff@mst.edu](mailto:ff@mst.edu)

**Dr. Muhanna Muhanna**  
Princess Sumaya University for  
Technology, Amman, Jordan  
[m.muhamna@psut.edu.jo](mailto:m.muhamna@psut.edu.jo)

**Dr. Mehdi O. Owrang**  
The American University, USA  
[owrang@american.edu](mailto:owrang@american.edu)

**Dr. Xing Qiu**  
University of Rochester, USA  
[xqiu@bst.rochester.edu](mailto:xqiu@bst.rochester.edu)

**Dr. Abdelmounaam Rezgui**  
New Mexico Tech, USA  
[rezgui@cs.nmt.edu](mailto:rezgui@cs.nmt.edu)

**Dr. James E. Smith**  
West Virginia University, USA  
[James.Smith@mail.wvu.edu](mailto:James.Smith@mail.wvu.edu)

**Dr. Shamik Sural**  
Indian Institute of Technology  
Kharagpur, India  
[shamik@cse.iitkgp.ernet.in](mailto:shamik@cse.iitkgp.ernet.in)

**Dr. Ramalingam Sridhar**  
The State University of New York at  
Buffalo, USA  
[rsridhar@buffalo.edu](mailto:rsridhar@buffalo.edu)

**Dr. Junping Sun**  
Nova Southeastern University, USA  
[jps@nsu.nova.edu](mailto:jps@nsu.nova.edu)

**Dr. Jianwu Wang**  
University of California  
San Diego, USA  
[jianwu@sdsc.edu](mailto:jianwu@sdsc.edu)

**Dr. Yiu-Kwong Wong**  
Hong Kong Polytechnic University,  
Hong Kong  
[eykwong@polyu.edu.hk](mailto:eykwong@polyu.edu.hk)

**Dr. Rong Zhao**  
The State University of New York  
at Stony Brook, USA  
[rong.zhao@stonybrook.edu](mailto:rong.zhao@stonybrook.edu)

ISCA Headquarters ••••• 64 White Oak Court, Winona, MN 55987 ••••• Phone: (507) 458-4517  
E-mail: [isca@ipass.net](mailto:isca@ipass.net) • URL: <http://www.isca-hq.org>

Copyright © 2016 by the International Society for Computers and Their Applications (ISCA)  
All rights reserved. Reproduction in any form without the written consent of ISCA is prohibited.

## Guest Editorial: Special Issue from ISCA Fall--2016 Conferences

This Special Issue of IJCA is a collection of four refereed papers selected from the CAINE 2016: 29th International Conference on Computer Applications in Industry and Engineering, held during September 26-28, 2016, in Denver, Colorado, USA,

Each paper submitted to the conference was reviewed by at least two members of the International Program Committee, as well as by additional reviewers, judging the originality, technical contribution, significance and quality of presentation. After the conferences, four best papers were recommended by the Program Committee members to be considered for publication in this Special Issue of IJCA. The authors were invited to submit a revised version of their papers. After extensive revisions and a second round of review, the four papers were accepted for publication in this issue of the journal.

The papers in this special issue cover a wide range of research interests in the community of computers and applications. The topics and main contributions of the papers are briefly summarized below.

ANDRÁS FEKETE and ELIZABETH VARKI of University of New Hampshire, USA, presented RAIDX in their paper “*RAID on a Heterogeneous Disk Array.*” It is a new type of heterogeneous RAID that is an extended and enhanced version of RAID. Based on RAID, by modifying the RAID structure and adding a lookup table, RAIDX supports heterogeneous disk array. So, different types of storage devices can be configured in RAIDX.

WEI LI of Nova Southeastern University, USA, in the paper “*A Comparative Evaluation of Trust Models in P2P Networks.*” presented a comparative study on three major trust models (Eigentrust, PeerTrust, and R<sup>2</sup>Trust) in P2P networks, proposed a generalized model for trust computation, and provided an analysis of how existing approaches can be encompassed in this model.

ANTOINE BOSSARD of Kanagawa University, Japan, and KEIICHI KANEKO of Tokyo University of Agriculture and Technology, Japan, discussed a scientific approach to Chinese characters in their paper “*A Scientific Approach to Chinese Characters: Rationale, Ontology and Application.*” They presented an ontology from information science point of view, including phonetic, morphological and graphic. Multi-lingual aspect of Chinese characters are considered in their study, as well as an algorithm for calculation of the distance between characters.

RUI WU, SERGIU M. DASCALU, LEE BARFORD and FREDERICK C. HARRIS, Jr of University of Nevada Reno, USA, and MUHANNA MUHANNA of Princess Sumaya University for Technology, Jordan, introduced three methods to improve the performance of the traditional GFC compression algorithm in their paper “*Data Lossless Compression Using Improved GFC Algorithm with Multiple GPUs.*” Their methods are using clzll to count the leading zeros, remove if-else statements in program, and use multiple Graphics Processing Units. Experimental results showed that the proposed methods improve the compression speed by more than 10-fold.

As guest editor's we would like to express our genuine appreciation for the encouragement and support from the editor-in-chief of IJCA, Frederick C. Harris, Jr. We also owe many thanks to the authors and program committees of the conferences these papers were selected from.

We hope you enjoy this special issue of the IJCA and we look forward to seeing you at a future ISCA conference. More information about ISCA Society can be found at <http://www.isca-hq.org>.

Guest Editors:

*Gongzhu Hu*, Central Michigan University, USA, CAINE 2016 Conference Chair

*Takaaki Goto*, Ryutsu Keizai University, Japan, CAINE 2016 Program Chair

November 2016

# Evaluating an Array of Heterogeneous Disks

Andras Fekete\* and Elizabeth Varki\*

University of New Hampshire, Durham, NH, USA

## Abstract

RAID assumes homogeneous disks. When a disk in RAID fails, it may be replaced by a larger disk, but the extra space in the new disk is wasted. To address this problem, this paper proposes RAIDX, RAID eXtended for heterogeneous disks. Similar to RAID, RAIDX bundles data across its disks; the stripes of RAID and RAIDX are constructed differently. In RAID, a stripe is a row of stripe units, one per disk; the stripe units are at identical locations on each disk. In RAIDX, a bundle is a row of chunks, with at most one chunk per disk; the chunks in a bundle need not be on the same location on each disk. Chunks of a bundle may be relocated dynamically when new disks are added to or removed from RAIDX. RAIDX requires a lookup table to map block numbers to chunk locations. This table is at worst only 0.3% of the total array size. The lookup tables add overhead to RAIDX, however, experiments demonstrate that RAID and RAIDX have comparable speeds when the array consists of similar disks. RAIDX supports arrays that contain a mix of hard disks and solid state disks. This combination can be used to increase the access speed of the array by directing traffic to the faster disks. RAIDX is compared to RAID by experiments that attempt to exercise similar functionality on the same hardware. We show that the proposed lookup table design adds only a little computational overhead and RAIDX approaches the speed of a traditional RAID array.

**Key Words:** RAID, heterogeneous disks, storage.

## 1 Introduction

Redundant Array of Inexpensive Disks (RAID) consists of several disks logically bound together to form a single storage unit, capable of higher performance than each individual disk. Hardware RAID uses RAID cards between the storage devices and the motherboard while software RAID uses the system CPU and memory to achieve the same function. Additionally, RAID offers redundancy to prevent data loss in the event of a drive failure. While the redundancy causes computational overhead, there is a net gain to this organization of storage.

There are several RAID configurations, of which RAID5,

RAID10, and RAID6 are the most popular. All configurations of RAID assume that the disks are identical. When disks fail, they are usually replaced by identical disks. Disks have an increasing Mean Time to Failure (MTTF), and they are growing in size. The combination of these two facts makes it plausible that when a disk fails in an array, it is likely replaced by a larger disk. Over time, a homogeneous array gets replaced by a completely new array with the old hardware discarded, a costly solution. A second solution is to replace the failed disks by new large disks, but the additional space in the new disks are not utilized. A third option is to extend the second solution by using the additional space as a separate storage unit, which interferes with the operation of RAID. All these solutions are ad-hoc, wasteful, and expensive. This paper addresses this issue by evaluating a RAID configuration called RAIDX [4], for heterogeneous disks and is an expansion of our prior work [5].

RAIDX - RAID eXtended for heterogeneous disks - configures different types of disks into a single array. For example, SSD and HDD could be combined into a single RAIDX array. The configuration of RAIDX ensures that the speed of the array approaches that of the faster disk in the array. Moreover, when disks of several sizes are placed in a RAIDX array, the additional space in the larger disks becomes available. RAIDX also supports all the traditional RAID levels (striping, mirroring, and parity). RAIDX, designed for heterogeneity of disk sizes and speeds, provides optimal performance regardless of homogeneous or heterogeneous disks.

This paper describes the RAIDX system architecture. The traditional, homogeneous RAID organizes storage data into stripes that span the disks uniformly. Each stripe (row) consists of stripe units, one per disk; the stripe units corresponding to a stripe at the same location on each disk. RAIDX has a completely different organization since disks vary in size. The bundles in RAIDX consist of chunks (not stripe units), which are not necessarily at the same location of each disk. Moreover, bundles need not span all the disks; larger disks participate in more bundles than smaller disks. In traditional RAID, stripes are just rows of storage data that logically follow each other. In RAIDX, bundles are organized for maximizing storage utilization of different sized disks.

Like in traditional RAID, the chunk size is selectable at initialization. With a smaller chunk size, the number of bundles increases thus increasing the space requirements for the lookup

\*College of Engineering and Physical Science. Email: afekete@wildcats.unh.edu and varki@cs.unh.edu

table both in memory as well as on disk. Nevertheless, the size of the lookup table in the worst case is less than 0.3% percentage of the array size. If the chunk size increases, the lookup table size decreases. The lookup table still offers an  $O(1)$  retrieval of chunk addresses. This is because once the array is assembled, the lookup table no longer changes.

In this paper we compare RAIDX performance to that of traditional RAID. We take into account one of the simpler RAID types, namely mirroring as a comparison. While there is a difference between the speeds achieved by the two implementations, there are many things that can still improve the speed of RAIDX. We have shown a 10 times speed increase over an individual drive in our 7 disk array. Mirroring RAIDX outperformed traditional RAID in terms of writes, but fell short on transactions with mostly reads. The crossover point occurs in transactions where at least 60% of the access is a read. A simple remedy for this would be to add a cache layer which will keep most frequently accessed bundles in memory.

## 2 Related Work

RAID [10] calls a piece of data or parity a stripe unit. Multiple stripe units are organized into stripes. Stripe units at the same logical location are in the same stripe. Stripe units are the smallest blocks that a RAID system can access.

Although other RAID algorithms exist, typically, RAID10, RAID5 and RAID6 is most commonly used. RAID10 is a combination of mirroring and striping across the SUs where RAID5 and RAID6 are single and double parity algorithms respectively. A parity is calculated by taking the XOR of the data SUs in the parity. For RAID5, these are the SUs in the stripe other than the parity stripe unit. In RAID6, the first parity can be the same that of RAID5, where the second is a myriad of possibilities [7, 9, 15, 16].

The basic rule in making an algorithm redundant is that each piece of the data and parity must be kept on a separate disk. The only way this is possible on differently sized disks is to treat all disks equal to the size of the smallest disk in the array. One trick that has been done is to place a second array on top of the remaining space on the disks. This process repeats until there are only two disks with free space which is then combined with a RAID1 array. The trouble this causes is that if both RAID arrays are accessed simultaneously there is a steep performance degradation. In any disk access, it is well known that the throughput is heavily influenced by the seek time required to access the data [11]. Therefore, any rotational disk that has multiple RAIDs on it will suffer from this.

Many solutions to this have revolved around obfuscating the underlying storage devices such as Logical Volume Management [14] (LVM). This divides up a physical volume (disk) to physical extents which can be allocated to logical extents in a logical volume. LVM leads to waste by adding a layer of abstraction in addition to sub-optimal utilization of hardware by treating all disks as identical. Others have tried to create virtual arrays [13] and combine them together into a

single system.

RAIDX combines the layer which creates an abstraction of the underlying heterogeneous storage with the layer that provides redundancy and speed improvements over a single drive. Being aware of the low level storage, RAIDX can make optimizations on the ordering of how transactions get executed.

## 3 RAIDX

Our current implementation is done entirely in software, but there is no restriction that would prevent a hardware implementation. Using the Linux network block device module, it is possible to create a block-level user space environment called BUSE [3]. This creates a block-level device that can represent the RAID array. By avoiding kernel modules, a much simpler implementation can be accomplished using object oriented languages. Using direct file access, we can force the kernel to retrieve data directly to the userspace memory and avoid unnecessary memory copies.

### 3.1 RAIDX Instantiation

In RAIDX there are some additional steps to assembling the RAID. When creating a new RAIDX set, one must define the number of chunks per bundle. The smaller the number, the closer to full utilization of the disks there is, but the larger the lookup table will be. This is because for a smaller chunks per bundle, there are more possible ways of spreading the chunks across the disks to optimize the layout. Each disk in the array can only store one chunk in a bundle to maintain redundancy. A disk with more than one chunk out of a bundle would be a weak link in the array where only a single disk failure is tolerated. If the chunks per bundle is equal to the number of disks in the array, then we have a traditional RAID layout and the number of stripes is determined by the smallest disk in the array. RAIDX is similar in that once the RAID is assembled, the chunks per bundle cannot be easily changed. The difference exists in the number of bundles. RAIDX allows for bundles to be added as well as removed based on the number of available chunks in the array. This means that as the array ages and transforms, it is possible to further increase the storage capacity. With modern filesystems, it is possible to resize the filesystem to follow the size of the underlying device. Many filesystems also allow to do this change on a live system.

The minimum necessary chunks per bundle for traditional RAIDs are shown in Table 1. The largest traditional RAID requires 4 chunks. In our experiments, where we have several different sized disks, we can see that this still produces a RAID with nearly the full size of the physical array. With a different set of disk sizes this same phenomenon occurs. The larger the number of disks in the array, the better the ability to make use of the full disk space.

To initialize the array, the number of chunks on each disk is calculated and these become the number of free chunks. It is possible to estimate the maximum number of bundles (see

Table 1: RAID level and minimum chunks per bundle

RAID Level	0	1	5	6	10
chunk per bundle	1	2	3	4	4

equation 1), but not every configuration of physical disk sizes allows to have all the disks used. A simple example is taking an array with a 1TB disk and two 250GB drives. There is no possible way to make full use of the 1TB array in a redundant array. If there were two chunks per bundle, the size of the array is at most 750GB using equation 1, but because of the large disparity between disk sizes and the small number of disks in the array, the array size is going to be only 500GB. If we added two more 250GB (or one 500GB) disks then we would reach the maximum size of the array.

$$numStripes = \frac{totalArraySize}{chunksPerStripe * chunkSize} \quad (1)$$

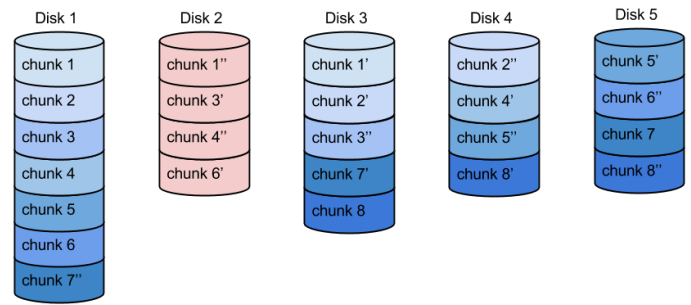
In traditional RAID, a hot spare is a drive that is put into the array as a backup drive that is normally unused, but in the event of a disk failure it becomes activated and the array fails over to that drive. In RAIDX, the concept of hot spare becomes unnecessary, because it is better to have the added performance of an additional disk in parallel. It is possible to put a smaller filesystem on the RAIDX array, so that if there is a disk failure, the RAIDX array can reorganize the chunk locations without needing to modify the filesystem.

### 3.2 RAIDX Lookup Table

As all chunks in a bundle can be arbitrarily allocated on the disk, we must keep a lookup table to later determine the selected locations. The table on the disk is much different than what is kept in memory. The disks store only information about those chunks that are stored on it. The size of the table on the disk is constant, because the number of chunks a disk can store is based on the size of the disk and the size of a chunk.

Figure 1 shows an example. The labels within each drive show the lookup table on each disk. The table at the bottom of the figure is representative of the lookup table stored in memory. Note that if a disk fails there is no use in having knowledge about which chunks were stored on it. In RAIDX a replacement disk may not be the same size as the failed disk, therefore any knowledge of previous chunks stored provides nothing. Each disk stores the UUID of the RAIDX array it is a part of, how many total stripes the RAIDX array has, the number of chunks per stripe, the size of a chunk, and the table of offsets of chunks on the disk. Individually, each disk knows nothing of other disks in the array or how many disks are part of the array.

The size of the lookup table depends linearly on the size of the chunks in the array. The smaller the chunks, the more that fit on the disks thus larger the table. This is true for both on-disk and in-memory tables. The lookup table only changes when there is a disk added or removed. Once a table is read from disk, there is no need to make any updates. The in-memory lookup table



	Chunk 1	Chunk 2	Chunk 3	Chunk 4	Chunk 5	Chunk 6	Chunk 7	Chunk 8
Piece 1	Disk 1	Disk 1	Disk 1	Disk 1	Disk 1	Disk 1	Disk 5	Disk 3
Piece 2		Disk 3		Disk 4	Disk 5		Disk 3	Disk 4
Piece 3	Disk 3	Disk 4	Disk 3		Disk 4	Disk 5	Disk 1	Disk 5

Figure 1: RAIDX lookup table

is intended to provide an O(1) lookup for the location of each chunk. Loss of the in-memory table is not a problem, because it can always be reconstructed from the tables stored on the disk.

One of the many advantages of RAIDX is that it is possible for the array to return to a fully redundant state after a disk failure but before that disk is replaced. This is because with a lookup table, the bundles can get redistributed across the remaining disks. With most modern filesystems, it is possible to reduce the filesystem size as long as the new size is greater than the size of the data stored on the filesystem. Once the filesystem has been resized, the RAIDX array can be restructured to remove the additional bundles and update the lookup table with the new locations of the chunks of a bundle. After the restructure, the RAIDX array is now fully redundant and can handle an additional disk failure. Conversely, when the data storage on the RAIDX array has become full, it is possible to add an additional disk to increase the physical storage capacity. This additional disk can be incorporated into the array by expanding the lookup table and reshuffling the chunks across the array.

### 3.3 RAIDX1

Using RAIDX, we looked at mirroring RAID. With chunks scattered across the disks, mirroring does not occur like it does on a traditional RAID. It is more akin to a RAID10 implementation. Figure 2 shows an example with two chunks per bundle that would be used for a mirrored RAID setup. Notice that if bundles 1-4 are requested, all the disks could be utilized. This does not necessarily mean that they should. Depending on the speed of the disks, a better solution might be to use only one disk to access the data. Recall that the time that takes the longest on any magnetic disk is the seek time. So if all four disks must seek to the same location to retrieve a small chunk, then in essence, each of the slower disks in the set are trying to keep up with the fastest disk. Disk scheduling methods are nothing new [12], what is new is how to select the best disk

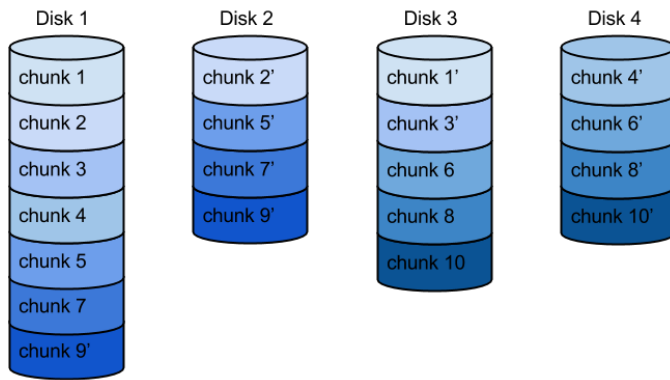


Figure 2: RAIDX array showing a 2 chunk per bundle setup

among a set of different disks. For this, we tested 4 different algorithms. The first (alg1), iterates through all disks, and only considers disks that are either twice as fast as the selected, or is idle while the selected is busy, or the distance the disk is estimated to seek is less than the currently selected one. If any of the considerations is true, then the disk under consideration is set as the selected one. The second algorithm (alg2) considers all disks independent of the speed. The third algorithm (alg3) is like the second, except it omits the busy consideration. The fourth algorithm (alg4) only looks at the seek distances.

Data access is treated as an individual chunk being the smallest block of data accessible. If a transaction requires multiple chunks, the best disk is decided on a per-chunk basis starting with the chunk on the first logical bundle. In our tests, we always had the bundles organized in ascending order. When disks have been added and removed from the array, this may not always be the case. It depends on how the algorithm that adds in new disks to a previously degraded array places the chunks. For the present experiments, we need not concern about this scenario.

#### 4 Simple RAID1

A simplified non-traditional RAID1 was implemented on the heterogeneous disk set. In this case, the heterogeneity is the speed of the disks. The goal is to understand how different speed disks in a RAID affects performance and if there is a different algorithm to take advantage of the variation in speed.

With this implementation, the disks were treated as identical sized storage devices. Several different algorithms were created to test different ways of using the write buffers.

Since the data on all the disks is identical, all writes to the array were issued to all the disks simultaneously. There were different algorithms for reads. They are discussed in the following sections. While many of the algorithms use asynchronous calls, the read only completes when all asynchronous calls have completed.

#### 4.1 Fastest Idle Read

This algorithm uses always the fastest disk from the set of idle disks to execute the read. It is an attempt to ensure that the RAID is not held back by waiting for a slow disk when a faster one is available. When all disks are busy, it will wait for the first one to become idle. We also use a synchronous read as there is no need to have the overhead of an asynchronous call for the particular data.

#### 4.2 All Idle Read Async

All Idle Read makes an asynchronous call to each disk in the set of idle disks to simultaneously read a portion of the requested blocks. The presumption is that on a set of disks, it is better to utilize all disks that are waiting idle.

#### 4.3 Fastest Read

This procedure always issued all read requests to a single disk which was determined to be the fastest at system startup. The assumption is that there may be a really fast disk in the array that can be able to keep up with the writes in addition to executing the reads.

#### 4.4 All Read

This algorithm attempts to simulate the traditional RAID1 such that all read operations are distributed evenly across all disks.

#### 4.5 Optimized Parallelism

Optimized Parallelism is a cross between Fastest Idle Read and All Idle Read Async. There are small transactions that would not make sense to issue to multiple disks. Thus, the Fastest Idle Read is used in that case, otherwise the All Idle Read Async method is used.

### 5 Experimental Setup

To test the speed of the array implementations, the standard testing tool called flexible I/O tester (fio) [1] was used. Fio is a tool for any arbitrary I/O throughput testing. It is commonly used as a comparison tool by many researchers [2, 8] for arbitrary I/O traffic generation. It is possible to generate random reads and writes with a desired percentage being reads. The main drawback is that it takes a long time to get the results as the transactions have to be executed on the given hardware.

In our tests we use fio to issue random reads and writes, starting with all writes and incrementing by 10% reads to 100% reads. This can show how the RAID performs with different types of loads. In initial tests, the size of each read or write is a constant 10MB. Subsequent tests were done with a random size ranging from 1KB to 10MB. Each test iteration was run for approximately 60 seconds. An iteration consisted of creating constant transactions at a set reads to writes ratio from

4 separate thread sources. This produced a very large number of transactions to exhaust any kind of system buffer or cache device and produce a reliable average transaction rate. The speed of the disk was recorded as the amount of data transferred in the actual amount of time.

A decision regarding the chunk size needs to be made when allocating a RAID storage device. A multiple of the disk block size is used for an optimal disk interface. In 2011, sector sizes have been defined to be 4096 bytes on all commercial drives [6]. In prior years, the sector size has been 512 bytes, but the increase in drive capacity allowed for a block size increase to make transactions more efficient.

In the RAIDX instance, the size of a chunk was varied across tests. The tested sizes included: 16K, 32K, 64K, 128K, 256K and 512K bytes. These are all the typical stripe unit sizes for traditional RAID.

Our control in this experiment was using the standard Linux MD RAID implementation on the same drives. For both the control as well as RAIDX1, we used the same 7 drives that were described in Section 3.1. All the drives were magnetic platter disks. The speeds of the disks ranged from 10.7MB/s to 21MB/s for reads and 2.2MB/s to 21.6MB/s for writes. The average speed was 16.6MB/s for reads and 10.25MB/s for writes.

Our experiments had four 148.5GB, one 139.2GB, one 297.5GB and one 74GB drives for a total of 1104GB. When the chunks per bundle equals the number of disks, this forces the total available space to be a multiple of the smallest disk in the array. In our case, that makes seven 74GB drives for a total of 518GB. Had there been a traditional RAID on this set of drives, over half of the physical storage would be unattainable to the array. Using a chunk size of one, two or three yields a total of 1104GB storage space for this set of drives. Therefore, a RAID1 or RAID5 can be easily placed on these disks and have the total capacity of the array while also being redundant. Using a double disk redundancy technique where there are four chunks per bundle like RAID6 would allow 1076GB of storage space. In RAIDX, a double disk redundancy becomes unnecessary. The reason for this is the possibility of dynamic array resizing discussed in Section 3.2.

## 6 Results

This section describes the results of the experiments run with both RAIDX and traditional RAID. First, an analysis of the base structure of the RAID is examined. Many attributes that apply to traditional RAID also applies to RAIDX. Subsequent sections show how RAIDX compares to traditional RAID.

It is important to examine the trade-offs in chunks per bundle and chunk size selection. We can easily determine that the chunk size has a linear correlation with the amount of memory the lookup table will consume and the allocation time of the lookup table. This is because the smaller the chunk size, the more bundles that fit on a disk. Smaller chunk sizes result in more efficient transfers of small files, but would cause slowdowns in larger files.

To understand how RAIDX performs, it is important to have a baseline comparison. This test used the Linux multi-disk module to create the RAID1 array with the parameters all set to the default. The RAID1 was constructed to use all the available space on the disks (see Figure 3). The total space available on the RAID was equal to the smallest disk. After the RAID was assembled, we ran the test routine to determine the baseline speed. This scenario would never be realized in a system as it would be too cost prohibitive. The reason for this test was to provide a baseline for what kinds of results we should be expecting with existing methods.

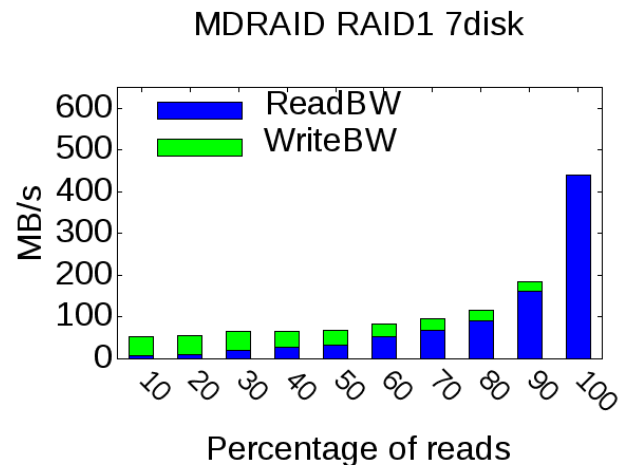


Figure 3: Using traditional RAID1 on a 7 disk array

**6.0.1 Simplified RAID1.** Using our simplified RAID1 implementation, we tested the performance of write buffers and multi-disk transactions on disks with different speeds. Figures 4 and 5 illustrates the simple RAID1 algorithm's transaction rate when the same 10MB block transactions were issued across 7 disks. Each of the graphs have the same axis limits to make comparisons easier to see. The All Read algorithm which is the traditional RAID1 algorithm performs really well because of the added parallelism. All Idle Read Async and Optimized Parallelism showed similar improvement with more parallelism. In fact, they performed better than the MD RAID system. The total bandwidth was around 550-600MB/s in each of those tests. In practice, this configuration would not be practical, but it does show that the process scales very well.

Fastest Idle Read and Fastest Read were most responsive to disk speeds. In tests where only two disks were used, when a magnetic disk was replaced with an SSD, the array had almost a 10x speedup in transaction speed. We can also observe that those algorithms that excelled with greater parallelism did not perform well in a situation where there were different speeds in disks.

Overall, it makes most sense to use Fastest Read when there is a disk that is orders of magnitude faster (like in the case of an SSD), but otherwise the All Idle Read Async and All Read



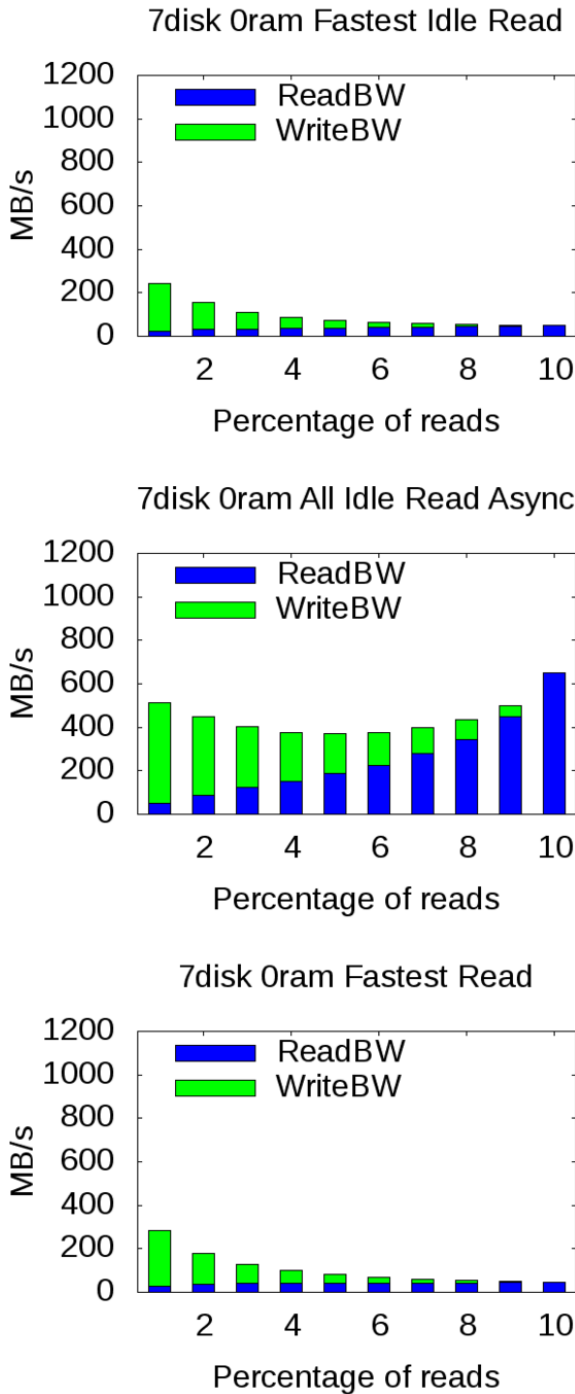


Figure 4: RAIDX1 throughput results for 7 disks with CPS=2

algorithms give the best performance.

### 6.1 RAIDX Performance

RAIDX also provides a set of write buffers to increase the bandwidth of the disks by keeping the disks continuously active during burst writes. These write buffers have been observed to increase the write intensive workload speeds by up to 40% in our

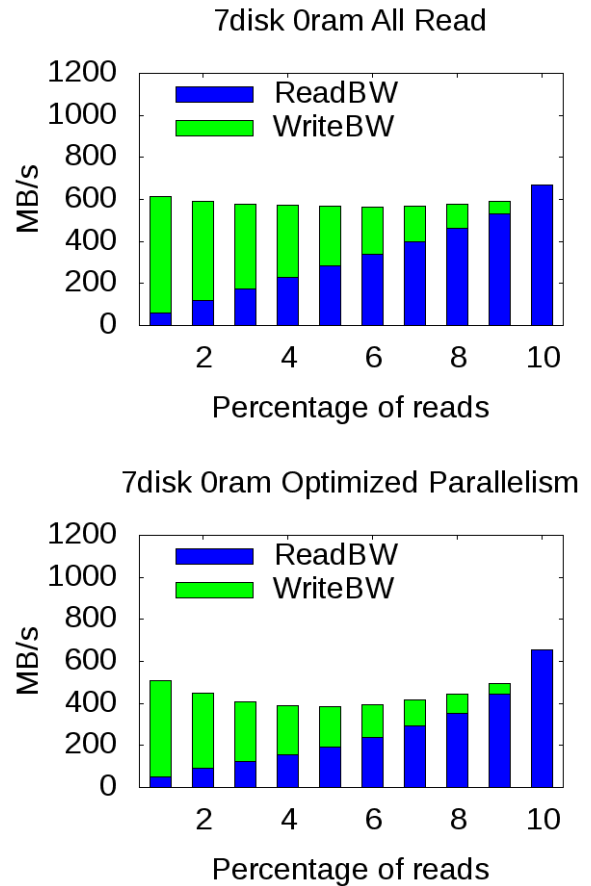


Figure 5: RAIDX1 throughput results for 7 disks with CPS=2

experiments. The concept is that in a hardware implementation, these write buffers would be placed in a non-volatile RAM so even in the case of power loss, the data will have been written to the RAID device and can resume writing when started up again.

Without the write buffers, the disk speeds suffer. Larger chunk size arrays have the largest impact. In these cases, what happens is that when a write transaction is taking place, it is blocking any future write transactions. The larger the chunk size, the larger the minimum size that a transaction must be to be split across multiple disks.

With smaller sequential transactions (1KB to 10MB versus a constant 10MB size), it is natural that the throughput shrinks on magnetic disks. Even still, the RAID array performs better than an individual disk due to the parallelism that is available. The transactions are likely to be carried out by only a single disk which leaves the rest of the disks idle.

When the same experiment was run with write buffers, the transaction speeds gained a 30% boost for 512KB chunk sizes, but only a 7.7% gain for 16KB chunk sizes. The buffers allowed more disks to be processing writes at the same time, but since the 16KB chunk size already split the transactions across more disks, it didn't see as much of an improvement. Consider that if a transaction is 64KB and chunk size is 16KB, then the

transaction will likely be split across 4 separate disks, whereas if the chunk size was 512KB, then the transaction will fall on a single disk unless it is on a chunk boundary where it will be split across two disks. This reason is why write buffers help the array with larger chunk sizes.

RAIDX1 on a set of heterogeneous disks is able to store more data than a traditional RAID1 because of the different layout. Traditionally, in RAID1, each two disks form a mirrored set. Thus, if the data being requested is largely in a certain logical location, then only two drives will have that data. The other disks in the array would be sitting idle. With RAIDX1, there wouldn't be any mirrored sets, as the bundles would be distributed evenly across all of the disks. Therefore, there is greater parallelism in a RAIDX1 set.

We find that the best algorithm is alg4 where we strictly look at the distance between where the head of the disk was last and where the next transaction needs to be.

## 7 Conclusions

RAIDX, a new type of heterogeneous RAID was developed and tested on a simple striping and mirroring RAID. RAIDX is different in that it uses bundles which are arranged on the disks in a fashion that is determined by the sizes of the disks. While this requires the use of lookup tables to keep track of where the bundles are, it does perform on par with traditional RAID and allows for additional features (such as RAID size extension) that can't be done with traditional RAID. We have shown that write speeds 10 times the speed of an individual drive in the array are attainable and sustainable. While read speeds are not as fast, it is possible to add a cache and prefetch layer to improve it, like it is done on most systems. To also help enhance reads on disks, we looked into how RAID1 will perform on a simplified implementation treating unequal sized disks as equal, but at different speeds. We then took this and created a RAIDX1 implementation using a subset of these algorithms and compared it to the traditional RAID1.

In this work, the main concentration was to ensure fast writes to an array of heterogeneous disks. In the current implementation of the algorithm, when several bundles are requested, each chunk is requested on the disk individually. Combining physically sequential chunk requests to disks have been shown to give significant improvements in our simplified tests.

In the future we will also consider RAIDX5, with the added advantage, that it may be possible to use the parity blocks on faster disks rather than data blocks to optimize the throughput. For example, given a bundles with chunks on various speed disks, it may be better to retrieve only those chunks that are on the faster disks and calculate the parity than to always retrieve the data blocks.

## References

- [1] J. Axboe. "fio - Flexible I/O Tester Synthetic Benchmark." URL <https://github.com/axboe/fio> (Accessed: 2015-06-13).
- [2] D. Bhagwat. "A Practical Implementation of Clustered Fault Tolerant Write Acceleration in a Virtualized Environment." *Proceedings of the 13th USENIX Conference on File and Storage Technologies*, pp. 287–300. 2015.
- [3] A. Fekete. "BUSE-CPP." URL <https://github.com/bandi13/BUSE-CPP> (Accessed: 2015-09-08).
- [4] A. Fekete and E. Varki. "RAID-X : RAID eXtended for Heterogeneous Arrays." "30th International Conference on Computers and Their Applications," pp. 157–162. URL <http://bit.ly/2g3W61H>. 2015.
- [5] A. Fekete and E. Varki. "RAID on a Heterogeneous Disk Array." "CAINE," Denver, CO. 2016.
- [6] IDEMA. "The Advent of Advanced Format." URL [http://www.idema.org/?page\\_{\\_}id=2369](http://www.idema.org/?page_{_}id=2369) (Accessed: 2015-06-13). 2013.
- [7] C. Jin, H. Jiang, D. Feng, and L. Tian. "P-Code: A New RAID-6 Code with Optimal Properties." "23rd International Conference on Supercomputing," URL <http://dl.acm.org/citation.cfm?id=1542326>. 2009.
- [8] H. K. Lee. "Minimizing Consistency-Control Overhead with Rollback-Recovery for Storage Class Memory." 2015.
- [9] X. Luo and J. Shu. "Generalized X-code." *ACM Transactions on Storage*, 8(3):pp. 1–16. ISSN 15533077. doi:10.1145/2339118.2339121. URL <http://dl.acm.org/citation.cfm?doid=2339118.2339121>. sep 2012.
- [10] D. A. Patterson, G. Gibson, and R. H. Katz. "A Case for Redundant Arrays of Inexpensive Disks (RAID)." "ACM SIGMOD Record," vol. 17, pp. 109–116. URL <http://dl.acm.org/citation.cfm?id=50214>. 1988.
- [11] S. Schlosser, J. Schindler, S. Papadomanolakis, M. Shao, A. Ailamaki, C. Faloutsos, and G. Ganger. "On Multidimensional Data and Modern Disks." *on File and Storage*, pp. 225–238. doi:10.1016/j.ejogrb.2011.11.018. URL [http://static.usenix.org/events/fast05/tech/schlosser/schlosser\\_{\\_}.html/](http://static.usenix.org/events/fast05/tech/schlosser/schlosser_{_}.html/). 2005.
- [12] M. Seltzer, P. Chen, and J. Ousterhout. "Disk Scheduling Revisited." 1990.

- [13] A. Thomasian. “Disk Arrays with Multiple RAID Levels.” *ACM SIGARCH Computer Architecture News*, 41(5):pp. 6–24. URL <http://dl.acm.org/citation.cfm?id=2641364>. 2014.
- [14] L. Vanel, R. V. D. Knaap, D. Foreman, K. Matsubara, and A. Steel. “AIX Logical Volume Manager from A to Z- Introduction and Concepts.” 1999.
- [15] P. Xie, J. Huang, Q. Cao, X. Qin, and C. Xie. “A New Non-MDS RAID-6 Code to Support Fast Reconstruction and Balanced I/Os.” *Computer Journal*, 58:pp. 1811–1825. doi:10.1093/comjnl/bxv006. 2015.
- [16] P. Xie, J. Huang, Q. Cao, and C. Xie. “Balanced P-Code: A RAID-6 Code to Support Highly Balanced I/Os for Disk Arrays.” “9th IEEE International Conference on Networking, Architecture, and Storage,” pp. 133–137. IEEE. ISBN 978-1-4799-4087-5. doi:10.1109/NAS.2014.29. URL <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6923172>. aug 2014.



**András Fekete** is a PhD candidate in Computer Science at the University of New Hampshire. He received his M.Sc. in Electrical Engineering in 2009, and B.Sc. in Computer Science in 2006. His research focuses on storage arrays consisting of different types of disks.



**Elizabeth Varki** received a PhD in Computer Science from Vanderbilt University in 1997. She is an Associate Professor in the Department of Computer Science at the University of New Hampshire. Her research and teaching interests are distributed systems, storage devices, and performance modeling.

# Evaluation and Generalization of Trust Models in P2P Networks

Wei Li\*

College of Engineering and Computing  
Nova Southeastern University, Fort Lauderdale, FL 33314 USA

## Abstract

Peer to Peer (P2P) networks have been widely used recently in various applications such as file sharing, content distribution, and e-commerce. At the same time, there were a number of attacks on the reputation mechanisms in these P2P networks. These attacks try to manipulate or misuse the reputation systems so that ratings on certain peers are biased, changed, or ignored. Many approaches have tried to defend against these attacks. This paper first provides a comparative study on trust models for P2P networks, with a focus on three major trust models: EigenTrust, PeerTrust, and R<sup>2</sup>Trust. We then propose a generalized trust model, show its parameters, and present a detailed analysis on how existing approaches can be encompassed in this model. In addition, we discuss issues related to existing models, and indicate a few potential areas for future research.

**Key Words:** Peer to peer (P2P) network, trust, reputation system, attack, network security.

## 1 Introduction

Peer to peer (P2P) networks, sometimes referred to as P2P overlay networks, have gained increasing popularity during the past two decades. In these networks, a peer (or node, user) can choose to join in or leave arbitrarily, and is able to connect to other peers for point to point communications. The overlay networks are built on top of existing TCP/IP network protocols and can better utilize limited network resources such as bandwidth or computational power. Due to their dynamic and distributed nature, these networks provide desirable features such as self-organization, robust routing, efficient searching, redundant storage, massive scalability, inherent anonymity, and fault tolerance, among others [1]. Different from the traditional client-server model of the Internet, each peer can act as the content provider (server) and content consumer (client) simultaneously. P2P networks have been used widely in applications such as e-commerce, file sharing, multimedia streaming. It was estimated that P2P network traffic constitutes more than half of today's Internet traffic volume [1].

Despite all these features, P2P networks face a number of security challenges. These challenges range from traditional

issues such as access control, denial of service (DoS) attacks, man-in-the-middle attacks, to distinctive issues such as routing disruption, collusions among multiple peers, and sharing of malware. One fundamental issue in P2P network is, which and to what extent peers can be trusted. Over the past decade, many trust/reputations based systems have been proposed to address this issue [2-9, 11]. These systems generally rely on the aggregation or estimation of ratings from peers in the same network, and may integrate different parameters such as social context, distance measurements, etc.

This paper is intended to perform a comparative study on a number of reputation models built for unstructured P2P networks (no centralized control once the peer to peer communication starts) and provide some directions for future research. On the other hand, structured P2P networks are tightly controlled and contents are distributed at specified locations to make subsequent queries more efficient [1].

The rest of the paper is organized as follows. In Section 2, we provide a general discussion on common risks and attacks on trust/reputation models in P2P networks. Section 3 presents a comparative review on three widely recognized trust models: EigenTrust, PeerTrust, and R<sup>2</sup>Trust. Section 4 shows a generalized trust model and discusses its parameters. Section 5 discusses issues related to existing models, and shows a few potential areas for future research. Section 6 shows the summary and conclusion of the paper.

## 2 Risks of Trust Models in P2P Networks

Risks in unstructured P2P networks are inherent in nature due to its absence of tight control. Peers in these networks can misuse the reputation systems either in isolated or in collaborative ways. The attacks are specific to the P2P network in which the reputation system was built. Broadly speaking, reputation attacks can generally be classified into three categories: unfair recommendations (peers spread unfair ratings), inconsistent behaviors (peers strategically misbehave that leads to an incorrect estimate of their reputation), and identity management related attacks (misuse identity scheme permitted by the P2P systems such as using multiple IDs with different ratings) [7]. The first category has attracted more focus than others in the research literature due to its direct impact on reputation systems, more specifically, on the calculation of ratings. Some typical unfair recommendations include the following [6,7].

---

\* 3301 College Avenue. Email: lwei@nova.edu.

- Collusion attacks. These attacks occur when a group of malicious peers collude to subvert the reputation system. In most cases the malicious peers are compromised or hijacked by a misbehaved peer. This is one of the most dangerous attacks because it is very difficult to track down the attack if they function correctly in the short term. In these attacks, malicious cooperate to spread bad ratings of other peers (*badmouthing*) while promoting each other's rating [7]. These peer attacks compromise the anonymity of peers.
- Forgery attacks. These attacks work by tampering the transmitted rating data. As a result, the ratings from a reputation system is not as trustworthy as it should be. These attacks compromise the confidentiality and integrity of reputation systems.
- Eclipse attacks. Sometimes referred to as membership attacks, these attacks work by controlling part of the overlay network to drop or reroute messages sent from legitimate peers, so that the ratings from the legitimate peers no longer count in the final ratings.
- Sybil attacks. These attack work by compromising the reputation mechanism. More specifically, an attacker may create a number of entities, and then use them together to defeat the reputation system. The target can be a single peer or a group of peers.
- Omission attacks. These attacks occur when the reputation mechanism is compromised. All ratings submitted by certain peers are ignored.
- Pollution attacks. These attacks are performed by sending a large volume of fake data.

Other than the attacks shown above, P2P reputation systems are also susceptible to traditional attacks such as Denial of Service (DoS) attacks, where malicious peers send excessive amounts of requests or ratings intended to overwhelm other peers; man-in-the-middle attacks, where intermediate nodes tamper messages they are supposed to deliver; or attacks against the P2P networks themselves. The intention is either to subvert the whole reputation system, or to create an unfair reputation so that ratings on certain peers are biased or ignored.

### 3 Reputation-Based Models in P2P Networks

As an active research field, a number of reputation-based models have been proposed during the past decade to address the security concerns shown above [2-4]. A survey on current status of reputation systems can be found in [5-8]. Due to the large number of publications in this field, we focus on a subset of representative approaches that tried to address the attacks with "unfair recommendations" shown above, namely, the PeerTrust [2], Eigentrust [3], and R<sup>2</sup>Trust [4].

There have been various definitions on reputation and trust systems. In this paper, we adopt the one proposed in [5], "A reputation system works by facilitating the collection, aggregation and distribution of data about an entity that can, in turn, be used to categorize and predict that entity's future

actions". Despite the fact that different reputation systems might have differences on how data is collected, how data is integrated, how data is used, and how the systems are deployed, they all provide a source of trust from which peers' future actions can be regulated. It should also be noted that there are delicate differences on the definitions of reputation and trust. Reputation refers more to the character others think someone has (the perception). On the other hand, trust focuses more on the measured dependence and reliability. Trust can be established through reputation, and a better reputation can lead to greater trust [5]. In this paper, we use the two terms interchangeably.

#### 3.1 Eigentrust

One of the most cited trust/reputation systems is Eigentrust [3]. It was designed to aggregate local trust values for a P2P file-sharing network, and the transactions consist of uploading and downloading tasks. Because of the distributed nature, these systems do not require a centralized storage and management system. Each peer maintains only trust values to its neighbors. The motivation behind Eigentrust was called "transitive trust" – a peer will have a high opinion of other peers who have provided authentic files, and is likely to trust the opinions of these peers. In other words, peers who are honest about the files they have are also likely to be honest in reporting their local trust values.

The local trust value  $s_{ij}$  is defined as follows.

$$s_{ij} = sat(i, j) - unsat(i, j)$$

In this definition,  $sat(i, j)$  is the number of satisfactory transactions (e.g., downloads, uploads) peer  $i$  has had with peer  $j$ ,  $unsat(i, j)$  is the number of unsatisfactory peer  $i$  has had with peer  $j$ .

Local trust values are then normalized according to the following equation.

$$c_{i,j} = \frac{\max(s_{ij}, 0)}{\sum_j \max(s_{ij}, 0)}$$

It is clear that  $0 \leq c_{i,j} \leq 1$ . If  $\sum_j \max(s_{ij}, 0) = 0$ , then  $c_{i,j}$  is undefined. To address this issue, the concept of a priori notion of trust was introduced, which indicates that some known peers (e.g., those established in the network at the beginning) are trustworthy. Let  $P$  be the set of peers that are known to be trusted,  $p_i = 1/|P|$  where  $i \in P$ , and  $p_i = 0$  otherwise. Normalized local trust values can then be rewritten as:

$$c_{i,j} = \begin{cases} \frac{\max(s_{ij}, 0)}{\sum_j \max(s_{ij}, 0)} & \text{if } \sum_j \max(s_{ij}, 0) \neq 0 \\ p_j & \text{otherwise} \end{cases}$$

Normalized local trust values are then aggregated according to the following equation.

$$t_{ik} = \sum_j c_{ij} c_{jk}$$

Here  $t_{ik}$  represents the trust peer  $i$  places in peer  $k$  by querying trust from its friends. Let  $C$  be the matrix  $[c_{ij}]$ ,  $\mathbf{t}_i$  be the vector containing the values  $t_{ik}$ , and  $\mathbf{c}_i$  be the vector containing the values  $c_{ij}$ , then we have  $\mathbf{t}_i = C^T \mathbf{c}_i$  where  $C^T$  is the transpose of  $C$ . At this point, the trust values stored by peer  $i$  contains only the experience of  $i$  and its neighbors. To get a broader view,  $i$  can continue to ask its friends' friends ( $\mathbf{t} = (C^T)^2 \mathbf{c}_i$ ), and the process will go on ( $\mathbf{t} = (C^T)^n \mathbf{c}_i$ ). It has been shown that if  $n$  is large, the vector  $\mathbf{t}_i$  will finally converge to the same vector for every peer  $i$ . In other words, it will converge to the left principle eigenvector of  $C$ . Here  $\mathbf{t}$  is a global trust vector and its elements  $\mathbf{t}_j$  contains how much trust the system as a whole places on peer  $j$ .

To protect the trust system from malicious collective attacks (more broadly, collusions and Sybil attacks), the global trust values is re-defined as

$$\mathbf{t}^{(k+1)} = (1 - \alpha)C^T \mathbf{t}^{(k)} + \alpha \mathbf{p}$$

where  $\alpha$  is a constant less than 1 and  $\mathbf{t}^0 = \mathbf{p}$ ,  $\mathbf{p}$  is the start vectors. Using this sliding-window equation, a peer crawling the network by the previous probabilistic model is unlikely to get stuck in a malicious collective as it has a probability to crawl to a pre-trusted network. The larger the value  $\alpha$  is, the better chance that peer will place more trust in the pre-trusted network. In a number of attack scenarios, this approach has shown its effectiveness. It also supports distributed and scalable computing [3]. The trust model was widely cited and compared against, and was extended in a number of research efforts such as [11].

### 3.2 PeerTrust

Another frequently cited work on reputation systems is called PeerTrust [2]. The approach targets on distributed e-commerce applications and reputation are calculated based on transactions, but it can also be adapted to other domains. It supports two main features. One is the inclusion of three basic trust parameters (feedback a peer receives from other peers, total number of transactions a peer performs, and the credibility of the feedback sources) and two adaptive factors (transaction context factor and the community context factor) in trust computation. The other feature was the definition of a general trust metric to combine these parameters.

The approach has the following parameters.

- $I(u, v)$ : total number of transactions performed by peer  $u$  with peer  $v$ ;
- $I(u)$ : total number of transactions performed by peer  $u$  with all other peers;
- $p(u, i)$ : other participating peers in  $u$ 's  $i$ th transaction;
- $S(u, i)$ : normalized amount of satisfaction peer  $u$  received from  $p(u, i)$  during its  $i$ th transaction;

- $Cr(v)$ : credibility of the feedback submitted by  $v$ ;
- $TF(u, i)$ : adaptive transaction context factor for peer  $u$ 's  $i$ th transaction;
- $CF(u, t_k, t)$ : adaptive community context factor for peer  $u$  during the period of  $t_k$  and  $t$ .

The trust value of peer  $u$  during the period of time  $t_k$  and  $t$  can then be defined as

$$T(u) = \alpha * \sum_{i=1}^{I(u)} S(u, i) * Cr(p(u, i)) * TF(u, i) + \beta * CF(u)$$

where  $\alpha$  and  $\beta$  denote the normalized weight factors for collective evaluation and the community context factors. The above equation can be simplified by manipulating the weight factors. When  $\beta = 0$  and  $\alpha = 1$ , it will result in a "basic metric" where community context is excluded from consideration.

To evaluate the credibility of feedback, PeerTrust includes two credibility measures. The first is to use a function of trust value of a peer as its credibility factor. It implies that feedback from trustworthy peers are considered more credible. Consequently, they are weighted more than those from untrustworthy peers. The metric can be defined as follows.

$$Cr(p(u, i)) = \frac{T(p(u, i))}{\sum_{i=1}^{I(u)} T(p(u, i))}$$

Another credibility measure is used for peer  $w$  to rate the credibility of another peer  $v$  through  $w$ 's personal experience, and will affect the feedback by  $v$  on other peers. It contains the following parameters.

- $IS(v)$ : set of peers that have interacted with peer  $v$ ;
- $IJS(v, w)$ : common set of peers that have interacted with both peer  $w$  and  $v$ , thus  $IJS(v, w) = IS(v) \cap IS(w)$ .

Subsequently, the measure is defined as

$$Cr(p(u, i)) = \frac{Sim(p(u, i), w)}{\sum_{i=1}^{I(u)} Sim(p(u, i), w)}$$

Where

$$Sim(v, w) = 1 - \sqrt{\frac{\sum_{x \in IJS(v, w)} \left( \frac{\sum_{i=1}^{I(x, v)} S(x, i)}{I(x, v)} - \frac{\sum_{i=1}^{I(x, w)} S(x, i)}{I(x, w)} \right)^2}{|IJS(v, w)|}}$$

Here  $Sim(v, w)$  is the similarity between the two feedback vectors. The rooted mean square measure is used to compute the similarity. Intuitively it implies that value as feedback from similar raters are given more weight. The design has the potential to defend against potential collusions, as the feedback

similarity between a peer in and a peer outside the collision group tend to be low. As a result it will filter out dishonest peers [2]. The transaction factor  $TF(u, i)$  can be based on transaction contexts such as the size, the category or time stamps. Recent transactions are generally assigned higher weight than other older transactions. The community context factor  $CF(u)$  can be defined as  $CF(u) = F(u)/I(u)$  where  $F(u)$  represents the total number of feedback peer  $u$  gives to others, as a building incentive to users who perform transactions or providing ratings [2].

### 3.3 R<sup>2</sup>Trust

A more recent model called R<sup>2</sup>Trust was proposed [4]. It integrated both the reputation and risk information into the model, and claimed to be able to handle malicious attacks, collusive attacks, and strategic attacks. It has a few distinctive features. First, the concept of risk was introduced in the computation of a peer's trust value. The risk represents various malicious behaviors, such as misuse of trust. Second, the recommender's credibility is updated quantitatively to minimize the effects from collusive peers. Third, the approach considers quality of service as probabilistic ratings in the interval [0, 1], not necessarily binary as appeared in most approaches [2].

In R<sup>2</sup>Trust, the overall trust value of peer  $j$  at peer  $i$  can be represented as

$$TV_{ij} = \alpha * T_{ij} - \beta * RV_{ij} \quad \text{where } 0 \leq \alpha, \beta \leq 1$$

In this equation,  $T_{ij}$  and  $RV_{ij}$  represent the trust value and risk value over peer  $j$ ,  $\alpha, \beta$  represent the weights for  $T_{ij}$  and  $RV_{ij}$ , and are set based on the optimistic perception of peer  $j$ . The computation of  $T_{ij}$  relies on two parts: direct trust and reputation value. The computation of direct trust relies on cases when a peer has direct transactions with other peers. In this part, the approach introduces a timing discount function, with the assumption that most recent ratings are more accurate to reflect a peer's reputation in the near future. The computation of reputation value relies on the aggregation of all the "referrals" from other peers. The reputation also considers the "credibility" of peers to identify peers that might be involved in collusive cheating attacks. The computation of risk value  $RV_{ij}$  is based on the "interaction-derived information" (local view of a peer on the whole network) by utilizing the concept of entropy. It was claimed that the use of risk evaluation provides a better chance to identify misbehaved peers. Despite the complex computations in this approach, the experimental results showed that R<sup>2</sup>Trust performed better than similar systems with a given set of parameters and has the potential to identify misbehaved peers [4].

### 3.4 Other Trust Models

Other than the trust models discussed above, there were quite a few other approaches proposed. In PowerTrust [12], a trust overlay network was used to model trust relationships.

The system dynamically chose most reputable nodes (the "power" nodes) based on a distributed ranking mechanism. Local trust scores were computed based on Bayesian inference. A distributed hash table (DHT) was used to propagate trust values among peers. The calculation of global trust was expedited by using look-ahead random walk strategy. The approach was able to adapt to situations with nodes dynamically joining and leaving the P2P network, and was able to withstand malicious users based on experiments with eBay data of more than 10,000 users [12]. However, it has been indicated by other researchers that in PowerTrust, subjective opinions from each node may be ignored as all participating nodes in the whole P2P network may assume the same trust value [13].

Some other models, such as SFTrust [14], have also been used for trust management in unstructured P2P networks, where there is no strict control over network topology, and there is no direct relationship between network topology and trust storage. Unlike most other approaches where trust was evaluated as a single metric, in SFTrust, trust values are categorized into two groups – service trust and feedback trust. It implies that a peer that provides high quality service may not necessarily provide high quality ratings, and vice versa. In other words, a peer may be used for service purposes even if feedback on other peers may not be trustworthy. Because of this, a double trust metric was used to evaluate trust values. Trust storage was implemented by using a topology adaptation protocol [14]. The approach has been criticized for lack of consideration on transactions with time variance and on the quality of transaction [13].

In a more recent model, a neighbor similarity trust measure was proposed to defend against specific attacks such as the Sybil Attack [15]. The approach assumes that 1) Sybil attack peers have relatively loose connections with the rest of the network; and 2) the attacks tend to use graph analysis techniques to estimate connection graph topology. Sybil attacks happen when a malicious peer creates multiple non-existent peers with different identities. To detect this, a peer is evaluated by referencing to its trustworthiness and the similarity to the neighbors. If the peer does not have the same trust/similarity data as its neighbors, it is considered as having multiple identities. In a small simulation network with 40 peers, the approach has shown to have better performance than Eigentrust in the detection of Sybil attacks [15]. However, the performance seemed to degrade in a sparsely connected network, and the similarity determination was based on threshold values, which in some cases might require expert opinions.

## 4 A Generalized Model for Trust Computation

In addition to the reputation models discussed above, there were various models proposed as extensions – especially those based on Eigentrust and PeerTrust. Here we want to consolidate the three models and develop a generic version, so that future researchers may look into their common features. It is also helpful for the research community to "think out of the box" of existing models and propose innovative solutions.

Most (if not all) P2P reputation models proposed can be generalized into the following generic form. A trust  $Trust_{ij}$  is defined as a two-way relationship that peer  $i$  places on peer  $j$ .

$$Trust_{ij} = f(Dir(i,j), InDir(j,k), Env(i)) \quad (1)$$

In this equation,  $Dir(i,j)$  means the direct relationship between peer  $i$  and peer  $j$ , which can be interpreted differently in different P2P systems – when there are direct transactions [2,3], when  $i$  downloaded contents from  $j$ , or when there is a direct connection between  $i$  and  $j$  in the overlay network. In this relationship, a peer  $j$  can simply have a binary direct impact on  $i$  (e.g., there is a transaction or no transaction), some type of numerical-value impact on  $i$  (e.g., there is a certain probability that a transaction might be successful), or an accumulated value such as the number of transactions.  $InDir(j,k)$  indicates an indirect relationship with peer  $k$  which is the set of peers that have an indirect relationship with  $j$ . The set of  $k$  can be interpreted as  $i$ 's neighbors' neighbors in a P2P overlay network. Note that both  $Dir(i,j)$  and  $InDir(j,k)$  can be as simple as a single value, positive or negative values, discrete or continuous values [4], or complex functions that take into account past transaction history.

$Env(i)$  denote the environmental factors in the network where peer  $i$  resides – such as the community context factors used in [2]. These environmental factors may include, but are not limited to, a pre-defined subset of trustworthy peers, how much weight should be applied to past and current transactions or credibility values, network topology, application areas, and could be updated dynamically. For example, a peer may not be initially included in the set of trustworthy peers. However, after a number of transactions and the peer showed its credibility, it can be included. Similarly, a peer may be excluded from the set due to malicious or irregular behaviors.

The weights can also be adjusted according to the context of transactions, and specific type of P2P networks. In e-commerce applications, the focus should be on the volume of successful transactions and on the monetary amount these transactions involve. On the other hand, in P2P streaming applications [6], more focus should be on the volume of content that a peer can successfully deliver.

The  $f()$  function defines how the direct impact, the indirect impact, and the environmental factors are aggregated. Considering the computational overhead, most approaches adopt a linear combination (e.g., summation) of these elements [2-4]. A reputation aggregation approach called FuzzyTrust was proposed in the literature, which has comparable performance with Eigentrust in experimenting with transaction data from eBay [9]. In this context, the  $f()$  function is defined as fuzzy inference rules and can be reasoned using the inference engine. The fuzzy inference engine has the distinctive benefit to handle imprecise linguistic terms. In addition to these efforts, with the adoption of compact data structures, there might be other approaches that may better reflect the trust relationship between peer  $i$  and peer  $j$  with similar or comparable computational overheads.

Figure 1 illustrates relationships among factors shown in Equation (1). Here  $i$  and  $j$  are directly connected peers, and  $InDir(j,k)$  consists of a number of indirect relationships between  $i$  and  $j$ 's neighbors  $k_1, k_2, \dots, k_n$ .

Trust computation is not a one-step process in distributed networks. These values are dynamic and can evolve as more transactions/activities accumulate over time. More importantly, due to the distributed nature of P2P networks, the initial computation of  $Trust_{ij}$  is solely based on peer  $i$ 's local perspective on the P2P network. The trust values need to propagate so that at one point, the values will converge. As a result, most reputation systems define the following (generic) function for trust propagation.

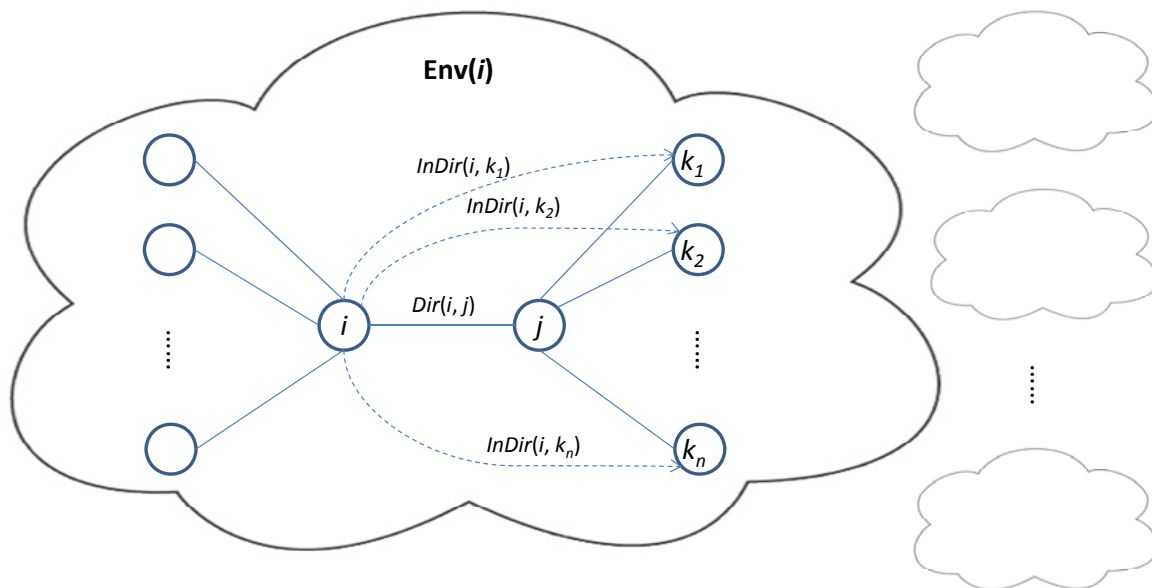


Figure 1: Generalized model for Trust Computation in P2P networks



$$Trust_{ij}^{(t+1)} = d(Trust_{ij}^{(t)}, c)$$

The equation denotes that the trust peer  $i$  places on peer  $j$  at time  $(t + 1)$  depends on the trust obtained at time  $t$ , and other factors  $c$  (e.g., the start vector in [3]). The  $d()$  function defines how these factors are aggregated. It can be a sliding window function [3], entropy based function [4], or other functions.

It should be noted that the generalized model hides many computational issues incurred by various functions. In real implementations, they may vary greatly. Despite this, the generic model should shed some light on how a generic trust/reputation model works, and how to develop future reputation models as improvements.

## 5 Discussion

Trust models (not necessarily in P2P networks) has been utilized in a number of e-commerce websites such as eBay, Amazon, Digg, ePoints, and Yelp. The models were also used in special purpose web page rankings, such as PageRank used by Google, Slashdot, StackOverFlow, etc. A wider usage is on P2P file-sharing networks such as Napster, YourBittorrent, Kazaa, Gnutella, and eDonkey. Many reputation systems are built with potential malicious behaviors in mind, but they still suffer a number of challenges shown below.

### 5.1 Basic Assumption

Any existing trust model implies a fundamental assumption that most peers in a P2P network behave honestly, and the misbehaved peers are relatively rare. The assumption is reasonable, but may not always be valid especially in large-scaled attacks such as DoS.

### 5.2 Aggregation Function

The aggregation function used in most approaches are based on a linear combination of trust or rating scores from peers, which by nature can be misused or manipulated. There also needs to be a mechanism to adjust the number of peers included in aggregation. Complex data fusion techniques (e.g., non-linear algorithms) can also be used here.

### 5.3 Convergence

The coverage refers to whether the trust function  $Trust_{ij}^{(t)}$  is able to converge after a number of iterations. Existing approaches did not provide a formal analysis, and a convergence is not guaranteed. Empirically, it has been shown that the EigenTrust algorithm adopted a simple evolution function, which will converge after 100 query cycles for a network of 1000 peers [3]. For a general convergence case, the algorithm does not provide an upper bound on  $n$  – the number of peers. The issue is more complex in a dynamic network.

### 5.4 Constant Factors in the Trust Function

Many trust systems use some type of constant values in their computation, for example, to what extent the trust should rely on the start vector ( $\alpha$  as defined in EigenTrust [3]), the community context ( $\alpha$  and  $\beta$  as defined in PeerTrust [2]), relative values between trust and risk ( $\alpha$  and  $\beta$  as defined in R<sup>2</sup>Trust [4]), and on recent/distant transactions. Although different algorithms define mechanisms to update these values based on empirical studies, constant factors still play a major role in trust computation, and may lead to inaccuracies.

### 5.5 Attacks Against Trust Systems as a Whole

Although some mechanisms are used to detect collusion/Sybil/Eclipse attacks, the attacks can only be detected for obvious versions. The elusive attacks are hard to detect since each rating seems legitimate individually. A recent attack, RepHi, has shown to be effective to subvert the rating systems [10]. As a result, we expect that these attacks will remain active in P2P networks for a long time.

### 5.6 Computational Overhead

In almost any reputation systems, there was a lack of formal analysis on how much computational overhead they may involve. More specifically, we wish to see an asymptotic analysis in the form of  $O(Trust_{ij}^{(t)})$ . It is understandable that such a formal analysis is difficult for any distributed algorithm, however even the formal upper bounds on key resources (memory, storage, and bandwidth) on a number of factors of computation will be helpful. It should be noted that the reputation systems are built on top of existing P2P networks, and in many cases, the peer needs a quick decision to choose peers to communicate. It is not desirable when computational overhead degrades the performance of overall P2P networks.

### 5.6 Alternative Approaches/Future Research Directions

The original trust issue in P2P networks can be attributed to a pattern recognition problem: *given a local view of each peer in a dynamic P2P network, how to develop a robust and efficient trust mechanism so that peers can rely upon for future transactions?* This is a difficult issue due to the following constraints: lack of centralized monitoring and management; limited view of each peer over the entire network; limited transactions among peers; convergence; computational overhead, among others. The following research directions may be worth future efforts: 1) more detailed analysis on real-world attacks against trust systems in P2P networks; 2) the use of compact digital signatures for validation of peers; 3) the use of compact data structure for storage; 4) machine learning techniques applied to trust computation; and 5) network simulation and performance evaluation of large-scale P2P networks.

## 6 Summary and Conclusions

Peer to Peer (P2P) networks have been widely adopted in recent years in various of applications such as file sharing, content distribution, and e-commerce. At the same time, there were a number of attacks on the reputation mechanisms in these P2P networks. These attacks intend to manipulate or misuse the reputation systems so that ratings on certain peers are biased or ignored. Many approaches have been proposed in the academia to defend against these attacks. This paper provided a comparative study on three major trust models in the research literature: EigenTrust, PeerTrust, and R<sup>2</sup>Trust. Key parameters of each model was shown and discussed in detail. We then proposed a generalized trust model, shown its parameters, and presented a detailed analysis how existing approaches work. In addition, we discussed issues related to existing models, and shown several potential areas for future research.

## References

- [1] Do-sik An, Byong-lae Ha, and Gi-hwan Cho, "A Robust Trust Management Scheme against the Malicious Nodes in Distributed P2P Network", *International Journal of Security and Its Applications*, 7(3):317-326, 2013.
- [2] Jingyu Feng, Yuqing Zhang, Shenglong Chen, and Anmin Fu, "Rephi: A Novel Attack against P2P Reputation Systems", 2011 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPs), IEEE, pp. 1088-1092, 2011.
- [3] Gabriela Gheorghe, Renato Lo Cigno, and Alberto Montresor, "Security and Privacy Issues in P2P Streaming Systems: A Survey", *Peer-to-Peer Networking and Applications*, 4(2):75-91, 2011.
- [4] Ferry Hendriks, Kris Bubendorfer, and Ryan Chard, "Reputation Systems: A Survey and Taxonomy", *Journal of Parallel and Distributed Computing* 75 (2015): 184-197.
- [5] Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina, "The EigenTrust Algorithm for Reputation Management in P2P Networks", *Proceedings of the 12th International Conference on World Wide Web*, ACM, pp. 640-651, 2003.
- [6] Eleni Koutrouli and Aphrodite Tsalgatidou, "Taxonomy of Attacks and Defense Mechanisms in P2P Reputation Systems—Lessons for Reputation System Designers", *Computer Science Review*, 6(2):47-70, 2012.
- [7] Xiong, Li, and Ling Liu, "Peertrust: Supporting Reputation-Based Trust for Peer-to-Peer Electronic Communities", *IEEE Transactions on Knowledge and Data Engineering*, 16(7):843-857, 2004.

- [8] Eng Keong Lua, Jon Crowcroft, Marcelo Pias, Ravi Sharma, and Steven Lim, "A Survey and Comparison of Peer-to-Peer Overlay Network Schemes", *IEEE Communications Surveys & Tutorials*, 7(2):72-93, 2005.
- [9] Félix Gómez Mármol and Gregorio Martínez Pérez, "State of the Art in Trust and Reputation Models in P2P Networks", *Handbook of Peer-to-Peer Networking*, Springer, US, pp. 761-784, 2010.
- [10] Shen Rao, Yong Wang, and Xiao-ling Tao, "The Comprehensive Trust Model in P2P Based on Improved EigenTrust Algorithm", *2010 International Conference on Measuring Technology and Mechatronics Automation*, IEEE, 3:822-825, 2010.
- [11] Shanshan Song, Kai Hwang, Runfang Zhou, and Y-K. Kwok, "Trusted P2P Transactions with Fuzzy Reputation Aggregation", *IEEE Internet computing*, 9(6):24-34, 2005.
- [12] Chunqi Tian, and Baijian Yang, "R<sup>2</sup>Trust, a Reputation and Risk Based Trust Management Framework for Large-Scale, Fully Decentralized Overlay Networks", *Future Generation Computer Systems*, 27(8):1135-1141, 2011.
- [13] Guojun Wang, Felix Musau, Song Guo, and Muhammad Bashir Abdullahi, "Neighbor Similarity Trust against Sybil Attack in P2P E-Commerce", *IEEE Transactions on Parallel and Distributed Systems*, 26(3):824-833, 2015.
- [14] Yunchang Zhang, Shanshan Chen, and Geng Yang, "SFTrust: A Double Trust Metric Based Trust Model in Unstructured P2P System", *IEEE International Symposium on Parallel & Distributed Processing, 2009, IPDPS 2009*, IEEE, pp. 1-7 2009.
- [15] Runfang Zhou and Kai Hwang, "Powertrust: A Robust and Scalable Reputation System for Trusted Peer-to-Peer Computing", *IEEE Transactions on Parallel and Distributed Systems*, 18(4):460-473, 2007.



member of ACM.

**Wei Li** is a Professor in the College of Engineering and Computing at Nova Southeastern University. His research interests include attack modeling and simulation, intrusion detection, firewall management, role-based access control, and the application of AI techniques in various security problems. He has published over two dozen papers in referred journals and conferences. He is a senior member of IEEE and a

# Chinese Characters Ontology and Induced Distance Metrics\*

Antoine Bossard<sup>†</sup>

Kanagawa University, Hiratsuka 259-1293, JAPAN

Keiichi Kaneko<sup>‡</sup>

Tokyo University of Agriculture and Technology, Koganei 184-8588, JAPAN

## Abstract

Saying that writing systems based on Chinese characters are perceived as difficult to master is no overstatement. At the same time, Chinese characters are ubiquitous across a large part of Asia, being used in several countries such as China and Japan. Hence, various methodologies supporting the learner memorization process have been proposed. Given the huge number of characters involved, memorization is a herculean, never ending task. At the difference of most previous methods, we detail in this paper a scientific approach to Chinese characters. Building on our previous research results, we conduct ontological discussion regarding Chinese characters from an information science point of view. Aiming at maximizing the versatility of such an information model, we shall consider multi-lingual properties of these characters. We subsequently review an important application of this foundation work by introducing the notion of *distance* between any two Chinese characters. Being a critical component for the pedagogical method of *character chaining*, several distance metrics are proposed in addition to a character chain construction algorithm.

**Key Words:** Linguistics; model; relation; natural; language; script.

## 1 Introduction

Due to the huge number of characters involved, memorization of Chinese characters is an extremely demanding task. This can be easily assessed by looking at the curriculum of Japanese elementary schools: a large part of curricula for successive years is occupied by Chinese characters studies. It is thus *a fortiori* very challenging for non-native learners of such a script, like the ones used by the Japanese and Chinese languages, amongst a few others such as Vietnam's *chữ nôm*.

The first objective of the this paper is the description of an ontology for Chinese characters, and from an information

science point of view. We thus aim at proposing an information model as complete as possible, for that considering Chinese characters from various point of views (e.g. graphical, phonetic, morphological, etc.) and various languages (scripts). This research work should therefore be considered as a foundation for scientifically approaching Chinese characters. Effectively, by providing a highly detailed information model, we would ideally gather all the information available for theoretical work with respect to Chinese characters, and with applications including for example automatic character processing, like assembling a character database. To the knowledge of authors, this is unprecedented work, although critical for numerous applications. The second objective is to use the previously introduced model to define the notion of *distance* between any two Chinese characters. A metric has important applications such as the automatic construction of character chains which facilitate character memorization.

Several pedagogic methods have been discussed in the literature aiming at easing the character memorization burden on the learner. A few examples of such works are recalled here. Always interesting are the short stories provided by Heisig for characters in his famous approach he applied to both Chinese [8, 9] and Japanese [7] scripts. In her classification work, Castelain has described an innovative method to lookup Chinese characters [4], thus by extension applicable to memorization as well. The phonetic approach has been explored amongst others by Vaccari and Vaccari [16]. In addition, the work by Henshall [10] provides a good overview of the classical approach to Chinese characters. Lastly, with our own work introducing the premises of an algebra of Chinese characters [2] (and related work by Sproat [15] describing a theoretical approach to writing systems in general), as well as the seminal works on writing systems by Sampson [13], Coulmas [5] and De Francis [6], we have briefly reviewed a rather large spectrum of what can be expected in the field.

The rest of this paper is organized as follows. In Section 2, the motivations for this research are further discussed. Next, preliminary results reused in this paper are briefly recalled in Section 3. The proposed ontology is described in Section 4 and examples are detailed in Section 5. Then, we focus in Section 6

\* An extended abstract of this paper has been published in [3].

<sup>†</sup>Graduate School of Science. Email: abossard@kanagawa-u.ac.jp.

<sup>‡</sup>Graduate School of Engineering.

on an interesting application of the proposed character ontology: by using the defined characters properties and relations, the notion of *distance* between characters is introduced. Finally, this paper is concluded in Section 7.

## 2 Rationale: Informal Discussion

We discuss in this section one important part of the motivation behind our approach. We start by making the link with philosophy, which is relevant here as we conduct this work of ontological discussion.

One does not learn philosophy, but rather learns, better *practices*, philosophy, that is to *philosophize*. Since extremely close to philosophy (precisely, focusing on say the technicality of philosophy), it is reasonable to assume that the same can be said about mathematics. One does not learn mathematics, but rather practices how to use them. Yet, this fundamental and essential aspect is lost, at least partially, in most other disciplines. Why? Possibly amongst other reasons, because the learner has to comply with various constraints, for instance material constraints for the physicist, natural ones for the biologist, and communication-related ones for the linguist. So, somehow, one “simply” learns biology, a language, etc.

Although unaware at first, it became clear to the authors that adopting a scientific approach to Chinese characters was a way to regain freedom lost by such constraints. When considering Chinese characters from a logical, algebraic point of view, we free ourselves from the communication barriers and bonds such as grammar, ruling how to communicate, and in other words being a kind of establishment in place for this Chinese characters topic. This is just one example, and it can be generalized to, say, other scripts and languages, and possibly to other unrelated subjects.

With such an algebraic approach, thus anchoring back to mathematics, we recover much freedom and set our minds free to not only learn characters (conventionally), but to *practice* them, considering these characters *for instance* as elements of a large set, and with relations between these elements – “for instance” is purposely emphasized here to show that with such an approach, it is entirely up to the individual to decide from where, in other words with what, to start. This aspect usually hidden to the language learner should be seen as another, novel and very natural way to approach and become used to Chinese characters.

Moreover, when considering natural languages, and more precisely sets including the various objects, like glyphs, defining such language, it is very common to discover punctual relations between these objects. Since reflecting natural languages, and thus empirically assembled, it is improbable that such sets originally include logical order or structure. But because of the applications mentioned previously, it is extremely interesting to try to formalize the possibly existing relations, potentially defining new ones, and eventually obtaining a logical structure that is much easier to use: simplified processing by computers and facilitated acquisition by learners are two examples.

## 3 Preliminaries

Let us recall that Chinese character decomposition operations have been introduced in previous works [2, 15]. For the sake of clarity, we recall in this section the two main operations introduced in [2].

Introduced initially for the subset  $\mathbb{J}$  of the Chinese characters used in Japanese, but nonetheless applicable to Chinese characters in general, the following two decomposition operations have been defined, amongst others – it is effectively important to note that the research conducted in this paper is applicable no matter the sorts and numbers of decomposition operations considered.

**Definition 1.** [2] *The operation  $+$  realizes the horizontal combination of the left operand with the right operand.*

$$+ : \mathbb{J} \times \mathbb{J} \rightarrow \mathbb{J}$$

$$\boxed{a} + \boxed{b} \mapsto \boxed{a b}$$

Let us illustrate this  $+$  operation with the following example. Consider the three Chinese characters 木, 南, 楠  $\in \mathbb{J}$  (“tree”, “south” and “camphor tree”, respectively); the equality 楠 = 木 + 南 holds.

**Definition 2.** [2] *The operation  $\times$  realizes the vertical combination of the left operand on top of the right operand.*

$$\times : \mathbb{J} \times \mathbb{J} \rightarrow \mathbb{J}$$

$$\boxed{a} \times \boxed{b} \mapsto \boxed{\begin{matrix} a \\ b \end{matrix}}$$

For example, we consider the three characters 山, 石, 岩  $\in \mathbb{J}$  (“mountain”, “stone” and “rock”, respectively). The equality 岩 = 山  $\times$  石 holds.

Finally, it is worth mentioning that these two character decomposition operations  $+$  and  $\times$  have the same evaluation priorities, and that they are applicable to a large majority of Chinese characters.

## 4 Proposal of a Chinese Character Ontology

In this section, we propose a detailed information model of Chinese characters. As we focus on a natural language, and especially its writing, it is a difficult task to formalize considered objects and their relations. The described ontology aims at addressing this issue. In order to facilitate the model description, we rely on the UML class diagram standard to represent the identified objects and their mutual relations. The diagram is given in Figure 1.

At the center of the diagram is the *Character* class, with its associations to multiple other classes. Hence, a Chinese character is described by the whole diagram, not only the *Character* class. In total, we have introduced seven classes (*Character*, *Radical*, *Script*, *Pronunciation*, *Meaning*,

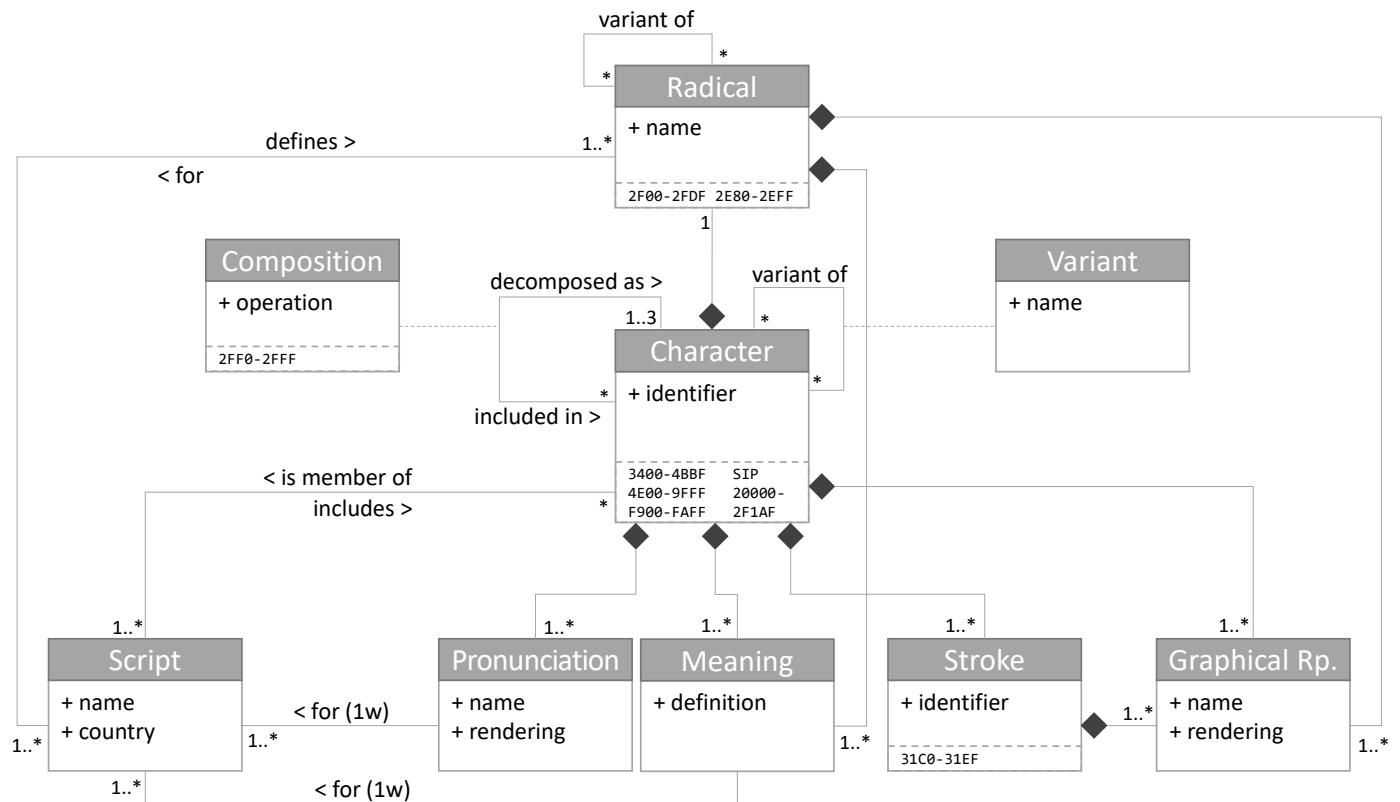


Figure 1: Class diagram illustrating the proposed Chinese character ontology. (*1w* stands for *one-way association*.)

*Stroke*, *Graphical Representation*) plus two association classes (*Composition*, *Variant*).

First, we discuss the *Character* class. An instance of this class, i.e. corresponding to a Chinese character, has one identifier, which can be conveniently derived from the Unicode standard or any other similar code (the JIS standard [11] is another example), and may be decomposed into several other characters according to a decomposition operation. Reversely, a character may be included in another character as a “sub-character”. This explains the *Composition* self-association.

In addition, the *Character* class has a second self-association: it enables the identification of character variants. As illustrated in Section 5, a same ideogram might be writable with several different characters, characters which are thus variants of that same ideogram (i.e. variants of each other). There exist several character variant sorts; they are distinguished by their names, such as “old form”, “simplified form”, etc.


A character has one unique radical, yet such radicals can have variants. This explains the self-association for the *Radical* class. A radical is identified by its name, some may have several, has at least one meaning and one graphical representation. The number, shape, etc. of radicals may vary slightly from one script to another (e.g. depending on the language or country); this explains the association to the *Script* class.

A character is part of at least one script, and conversely a script includes several characters. Script examples include simplified Chinese, traditional Chinese, Korean, etc. A character obviously

has at least one graphical representation, for instance the one based on the *seal* character style; more details are given in Section 5.

A character has at least one meaning as well as at least one pronunciation (i.e. phonetic information). These two character properties depend on the script considered for a particular character; this explains the two one-way associations from the *Pronunciation* and *Meaning* classes to the *Script* class. Regarding the pronunciation information, the reason is straightforward: a same character is expressed phonetically in a different manner considering for instance the Japanese and Mandarin Chinese languages. Justification is less obvious regarding the meaning of a character: even though originating from the same roots, and thus from a same meaning, the meaning for one character may have evolved over time, and may thus today possibly differ from one language to another.

Finally, a character consists of one or several strokes. There exist 36 different strokes which are combined to represent Chinese characters; each stroke is usually identified by a name. In the diagram, several classes include at the bottom additional information under a dotted line: this is the Unicode range (i.e. characters, glyphs) applicable to the corresponding class. For example, the *Stroke* objects actually match the glyphs described by the Unicode standard in the range 31C0–31EF, and whose graphical representation can be: 一 丨 丿 ㇀ ㇁ ㇂ ㇃ ㇄ ㇅ ㇆ ㇇ ㇈ ㇉ ㇊ ㇋ ㇌ ㇍ ㇎ ㇏ ㇐ ㇑ ㇒ ㇓ ㇔ ㇕ ㇖ ㇗ ㇘ ㇙ ㇚ ㇛ ㇜ ㇝ ㇞ ㇟ ㇠ ㇡ ㇢ ㇣ ㇤ ㇥ ㇦ ㇧ ㇨ ㇩ ㇪ ㇫ ㇬ ㇭ ㇮ ㇯ ㇰ ㇱ ㇲ ㇳ ㇴ ㇵ ㇶ ㇷ ㇸ ㇹ ㇺ ㇻ ㇼ ㇽ ㇾ ㇿ ㇺ ㇻ ㇼ ㇽ ㇾ ㇿ ㇺ ㇻ ㇼ ㇽ ㇾ ㇿ (36 glyphs in total; the last 12

glyphs of the range remain unused). Similarly, the *Composition* association, which is based on the formally introduced decomposition operations  $+$  and  $\times$  (see Section 3), matches the glyphs described in the Unicode range 2FF0–2FFF:  (12 glyphs in total; the last 4 glyphs of the range remain unused).

As a next step, several concrete and non-trivial examples of object and association instances will be discussed in Section 5 below.

## 5 Object Instance Examples

We illustrate in this section the information model described in Section 4 by giving object instance examples of the defined classes and associations. These examples will be given in a classic object-oriented programming writing style (dotted notation), as in C++ and Java. In addition, it is assumed in the following notations that members of an object instance are implicitly instantiated. For example, a *Character* object has an instance radical of the *Radical* class.

First, we give as example a possible way to instantiate a *Character* object. The radical name given in this example is the one used in Japanese.

```
Character c
c.meaning.definition = {belief, trust}
c.radical.name = ninben
c.graphicalRp.rendering = {信}
```

Then, the *Composition* association between several characters is illustrated. As mentioned previously, this is related to previous work focused on the algebraical approach to Chinese characters [2, 15].

```
Character c1(相) // simplified construction
Character c2(木), c3(目), c4(湘)

Composition co(c1)
co.operation = <+> // meaningful for decomposedAs only
co.decomposedAs = {c2, c3}
co.includedIn = {c4}
```

Next, we present an instance example of the *Variant* association. Attention should be paid to the order used to declare such relation between two characters.

```
Character c1(学), c2(學)

Variant v(c2)
v.name = old_form
v.variantOf = c1 // c2 is an old-form variant of c1
```

We continue by illustrating the relation between characters of different languages and scripts.

```
Character c1(业), c2(業)
c1.script.name = Simplified_Chinese
```

```
c1.script.country = {China, Singapore}
c2.script.name = Japanese
c2.script.country = {Japan}
```

```
Variant v(c1)
v.name = simplified_shape
v.variantOf = c2 // c1 is a simplified-form variant of c2
```

Effectively, in our approach, it is meaningful to consider that a simplified Chinese character (业 above) is a *variant* of the corresponding traditional character (業 above), rather than just a different graphical representation. Indeed, a graphical representation is not related to a script or language. Refer to the additional examples given below with respect to graphical representations.







Because it is definitely worth paying attention to radicals, a non-trivial example is given below. Once again, the radical names given in this example are the ones used in Japanese.

```
Radical r1, r2
r1.name = mizu
r1.meaning.definition = {water}
r1.graphicalRp.rendering = {水}
r2.name = sanzui
r2.graphicalRp.rendering = {彡}

RadicalVariant rv(r2)
rv.variantOf = r1 // r2 is a radical variant of r1

Character c(洪)
c.radical = r1 // NB: r2 also valid here since variant of r1
```

Finally, *Graphical Representation* objects are illustrated. The character shapes used in this example originate from the *Ancient Chinese characters project* [17] and are in the public domain. For the sake of clarity, this example uses the name: `rendering` syntax to denote the instantiation of a *Graphical Representation* object, thus directly assigning in practice the name of a rendering (i.e. here, a graphical style) to the rendering itself (i.e. image information, such as a bitmap).

```
Character c;
c.meaning.definition = {horse}
c.graphicalRp = {
  oracle bone: ,
  bronze: ,
  big seal: ,
  seal: ,
  clerical: ,
  regular: 
}
```

## 6 Application: Character Distance

In this section, we propose the definition of a distance metric between any two Chinese characters. This is a direct application of the character ontology presented previously since we rely entirely on the identified character properties and relations. Defining such a distance metric between characters is indeed meaningful in the pedagogical context. Effectively, it enables the creation of *character chains*, which are in practice sequences of characters with the least possible changes between two consecutive characters in such a sequence. Here is an example of such a sequence: 單 → 單 → 戰 → 蟬 → 虫 → 蟲; additional sample sequences can be found in various reference works such as [14]. As a result, memorization is highly facilitated for the learner when relying on such character chains for memorization – refer to [1] in which is addressed the closely related character layerization and cartography concept. By formally defining such a character distance metric, we enable the automatic generation of character chains.

In search of a suitable and coherent distance metric, we shall propose and discuss hereinafter several such measures. Each metric (except the last one,  $d$ ) can be seen as a morphological and semantic distance between two characters since we simultaneously rely on morphological information for a character such as decomposition matters, as well as semantic information such as variants and radicals for characters. Informally, the calculated distance is a real number that gets larger if the two characters have little in common, and conversely a real number that gets smaller if the two characters share several attributes.

### 6.1 The $\delta$ Metric

In this section, we describe the distance  $\delta(a, b)$  between any two Chinese characters  $a$  and  $b$ . This distance  $\delta(a, b)$  is expressed as a positive real number. To start, let us consider several specific properties for a character. For instance, we consider the following two character properties – note that more or fewer such properties could be similarly treated. First, the *variant* property, which is satisfied if and only if the character  $a$  is a variant of the character of  $b$ . Second, the *radical* property, which is satisfied if and only if the two characters  $a$  and  $b$  share the same character radical.

Let  $p$  represent the number of satisfied properties between the two characters  $a$  and  $b$ . Since we have considered in this example two such character properties, we have  $0 \leq p \leq 2$ . Then, as recalled in Section 3, we shall rely on decomposition operations. Let  $o_{a,b}$  (resp.  $o_{b,a}$ ) be the number of decomposition operation levels required for  $a$  (resp.  $b$ ) until finding a common element (i.e. a sub-character) for  $a$  and  $b$ . Thus, the value of  $o_{a,b}$  (resp.  $o_{b,a}$ ) is minimal at any time. In the case the characters  $a$  and  $b$  share no common element, define  $o_{a,b} + o_{b,a} = \Omega$ , with  $\Omega \in \mathbb{R}$  a large positive constant.

As example, for the characters  $a = 峠$  and  $b = 雫$ , we have the two decompositions  $a = 山 + (上 \times 下)$  and  $b = 雨 \times 下$ . It takes two levels of decomposition for  $a$ , and one single decomposition

level for  $b$  before finding the first common element, here precisely 下. Hence, we have  $o_{a,b} = 2$  and  $o_{b,a} = 1$ .

This number of decomposition operations is further refined by excluding the radical as follows. Given two characters  $a$  and  $b$  of radicals  $r_a$  and  $r_b$ , respectively, in the case both  $r_a = r_b$ ,  $a \neq b$ ,  $a \neq r_a$ ,  $b \neq r_b$  hold, let  $\tilde{o}_{a,b}$  (resp.  $\tilde{o}_{b,a}$ ) be the number of decomposition operation levels required for  $a$  (resp.  $b$ ) until finding a common element for  $a$  and  $b$  (i.e. a sub-character) *other than the radical*  $r_a (= r_b)$ . If the characters  $a$  and  $b$  share no common element except their radical, define similarly  $\tilde{o}_{a,b} + \tilde{o}_{b,a} = \Omega$ . Otherwise, that is in the case either  $r_a \neq r_b$ ,  $a = b$ ,  $a = r_a$  or  $b = r_b$  is satisfied, simply define  $\tilde{o}_{a,b}$  (resp.  $\tilde{o}_{b,a}$ ) as  $o_{a,b}$  (resp.  $o_{b,a}$ ). For example, given the two characters  $a = 涪$  and  $b = 沼$  of same radical  $r = 氵$ , we have  $a = 涪 = r + (十 \times 口)$ , which induces  $\tilde{o}_{a,b} = 2$ , and we have  $b = 沼 = r + (刀 \times 口)$ , which induces  $\tilde{o}_{b,a} = 2$ . Also, for any character  $a$ , we have  $\delta(a, a) = 0$ .

**Definition 3.** For any two characters  $a$  and  $b$ , their distance  $\delta(a, b) \in \mathbb{R}$  is defined as

$$\delta(a, b) = \frac{\tilde{o}_{a,b} + \tilde{o}_{b,a}}{p + 1}$$

One should note that if the distance  $\delta(a, b)$  depends on  $\Omega$ , it will necessarily be of the form  $\Omega/n$  with  $n \in \mathbb{N}^*$ , thus allowing for total ordering of distances. Several additional examples are given in Table 1. The “variant” and “radical” columns are Boolean values respectively meaning “ $a$  is a variant of  $b$ ” and “ $a$  and  $b$  share the same radical”, or their opposites.

Table 1: Examples of character distance calculations with the  $\delta$  metric

$a$	$b$	variant	radical	$p$	$\tilde{o}_{a,b}$	$\tilde{o}_{b,a}$	$\delta(a, b)$
洪	浜	no	yes	1	2	2	2
榎	夏	no	no	0	1	0	1
桜	櫻	yes	yes	2	2	2	4/3
峠	雫	no	no	0	2	1	3
湘	眼	no	no	0	2	1	3
木	林	no	yes	1	0	1	1/2
木	水	no	no	0	$\Omega$		$\Omega$
沐	浴	no	yes	1	$\Omega$		$\Omega/2$

Hence, given a set of characters and one starting character, a character chain can be established rather simply as follows. For example, consider the character set  $E = \{\text{蟲, 戰, 蟬, 虫, 單, 單}\}$  used as example at the beginning of this section, and 單  $\in E$  the starting current character of the character chain to be obtained. By iterating the set  $E$  repetitively to find the character which is at a closest distance from the current character, subsequently updating the current character, we have the following steps:

1. 單 :  $\{(\text{蟲}, \Omega), (\text{戰}, 1), (\text{蟬}, 1), (\text{虫}, \Omega), (\text{單}, 2/3)\}$
2. 單 :  $\{(\text{蟲}, \Omega), (\text{戰}, 3), (\text{蟬}, 3), (\text{虫}, \Omega)\}$

3. 戦 :  $\{(\text{蟲}, \Omega), (\text{蟬}, 2), (\text{虫}, \Omega)\}$
4. 蟬 :  $\{(\text{蟲}, 1), (\text{虫}, 1/2)\}$
5. 虫 :  $\{(\text{蟲}, 1/3)\}$

where a pair  $(e \in E, \gamma \in \mathbb{R})$  means that the current character  $c$  (on the left of the colon), and the character  $e$  are at distance  $\delta(c, e) = \gamma$ . Therefore, we obtain the character chain 単  $\rightarrow$  單  $\rightarrow$  戦  $\rightarrow$  蟬  $\rightarrow$  虫  $\rightarrow$  蟲, which indeed matches the chain given as example previously. For additional details, the pseudo-code of this chaining algorithm is given in Algorithm 1.

---

**Algorithm 1:** CHAINING( $E, c$ )

---

**Input:** An unordered set of characters  $E$ ; a starting character  $c \in E$ .  
**Result:** A character chain corresponding to  $E$ .  
**if**  $|E| = 1$  **then**  
     $\{e\} := E$ ;  
    **return**  $e$   
**else**  
     $E' := \arg \min_{e \in E \setminus \{c\}} \delta(c, e)$ ;  
     $\{e'_1, e'_2, \dots, e'_{|E'|}\} := E'$ ; //  $c$  equidistant to  $e'_i$   
    **return**  $e'_1 \rightarrow \text{CHAINING}(E \setminus \{c\}, e'_1)$   
**end**

---

## 6.2 Property Independence and the $\delta'$ , $\delta''$ Metrics

The  $\delta$  metric described previously includes in its definition  $p + 1$  as denominator so as to take into account the properties shared between characters. Yet, this denominator may legitimately seem unnatural. So here, we refine the  $\delta$  metric by handling shared properties differently.

As with the  $\delta$  metric, let us consider the two *radical* and *variant* properties; more would be handled similarly. Let  $\varepsilon$  be the symbolic constant associated with the *radical* property;  $0 < \varepsilon < 1$  holds. Similarly, let  $\varphi$  be the symbolic constant associated with the *variant* property;  $0 < \varphi < 1$  holds. Such constants shall be subtracted from the calculated character distance in case the corresponding property is satisfied: a shared property induces a shorter distance.

Moreover, considered properties may not always be independent. Hence, for two properties  $p$  and  $q$  of respective symbolic constants  $c_p$  and  $c_q$ , if  $p \Rightarrow q$  holds, then  $c_p > c_q$  is induced, and  $c_p$  only is subtracted (not subtracting both  $c_p$  and  $c_q$ ). For instance, in the case of the two constants  $\varepsilon$  and  $\varphi$ , respectively corresponding to the *radical* and *variant* properties, when satisfied the *variant* property induces the *radical* property (character variants simply signify appearance changes, thus retaining the same radical), hence  $0 < \varepsilon < \varphi < 1$ . So, when calculating the distance between two characters that are variants,  $\varphi$  is subtracted.

Therefore, in Definition 4 below, we can assume without loss of generality that the shared properties  $p_1, p_2, \dots, p_k$  of any two characters are independent, and we introduce the following definition.

**Definition 4.** For any two characters  $a$  and  $b$  sharing independent properties  $p_1, p_2, \dots, p_k$ , their distance  $\delta'(a, b) \in \mathbb{R}$  is defined as

$$\delta'(a, b) = \bar{o}_{a,b} + \bar{o}_{b,a} - \sum_{i=1}^k c_i$$

and their distance  $\delta''(a, b) \in \mathbb{R}$  as

$$\delta''(a, b) = o_{a,b} + o_{b,a} - \sum_{i=1}^k c_i$$

with  $c_i$  the symbolic constant corresponding to  $p_i$ .

The sample distances calculated in Table 1 are expressed with the  $\delta'$  and  $\delta''$  metrics in Table 2: the two dependent properties  $p_1, p_2$  corresponding to *variant* and *radical* (thus  $p_1 \Rightarrow p_2$ ) and of symbolic constants  $c_1 = \varphi$ ,  $c_2 = \varepsilon$ , respectively, are considered.

Table 2: Examples of character distance calculations with the  $\delta'$  and  $\delta''$  metrics

$a$	$b$	$p_1$	$p_2$	$\bar{o}_{a,b}$	$\bar{o}_{b,a}$	$\delta'(a,b)$	$o_{a,b}$	$o_{b,a}$	$\delta''(a,b)$
洪	浜	no	yes	2	2	$4 - \varepsilon$	1	1	$2 - \varepsilon$
榎	夏	no	no	1	0	1	1	0	1
桜	櫻	yes	yes	2	2	$4 - \varphi$	1	1	$2 - \varphi$
峠	雫	no	no	2	1	3	2	1	3
湘	眼	no	no	2	1	3	2	1	3
木	林	no	yes	0	1	$1 - \varepsilon$	0	1	$1 - \varepsilon$
木	水	no	no	$\Omega$		$\Omega$	$\Omega$		$\Omega$
沐	浴	no	yes	$\Omega$		$\Omega - \varepsilon$	1	1	$2 - \varepsilon$

## 6.3 Discussing the $\delta$ , $\delta'$ and $\delta''$ Metrics

First, the original  $\delta$  metric makes use of the  $p + 1$  denominator, which may seem unnatural. Hence, the introduction of refined metrics  $\delta'$  and  $\delta''$ .

Second, with the  $\delta''$  metric, as shown in Table 2 the equality  $\delta''(\text{洪}, \text{浜}) = \delta''(\text{沐}, \text{浴})$  holds, which induces some incoherency given that the two sub-characters 共, 兵 are closer (i.e. more related) than the two sub-characters 木, 谷.

Third, with the  $\delta'$  metric, as shown in Table 2 the equality  $\delta'(\text{沐}, \text{浴}) = \Omega - \varepsilon$  holds. In other words, even though the two characters 沐, 浴 share the same radical (氵), their distance involves  $\Omega$  and is thus large. Even though this may seem disturbing at first glance, coherence is retained. Yet, one possibly concerning aspect of this  $\delta'$  metric, is that it may not treat character variants fairly enough, that is, not reducing the distance value enough in case the two characters are variants of each other. For instance, we have  $\delta'(\text{桜}, \text{櫻}) = 4 - \varphi$  for the two variant characters 桜, 櫻, while for instance we have  $\delta'(\text{峠}, \text{雫}) = 3$  for the two, not variants, characters 峠, 雫, with  $\delta'(\text{桜}, \text{櫻}) > \delta'(\text{峠}, \text{雫})$  thus holding. Again, even though this may seem disturbing at first glance given that



桜, 櫻 are variants while 峠, 霰 are not, some coherence is retained. For comparison, the original,  $\delta$  metric induced distances  $\delta(\text{桜}, \text{櫻}) = 4/3$  and  $\delta(\text{峠}, \text{霰}) = 3$ . This issue is mitigated with the  $\delta''$  metric since we have  $\delta''(\text{桜}, \text{櫻}) = 2 - \varphi$ , and  $\delta''(\text{峠}, \text{霰}) = 3$ .

### 6.4 The $d$ Metric

The previously defined  $\delta$ ,  $\delta'$  and  $\delta''$  distance metrics are not distances in the mathematical sense. Effectively, it is easy to show that they do not satisfy the triangle inequality. In this section, we propose another distance metric, this time satisfying the requirements for a mathematical distance, most notably the symmetry property and the triangle inequality. Each character is assumed without loss of generality to be either a combination of sub-characters, or a canonical element [2], the  $\emptyset$  symbol representing the empty character and thus being canonical. Distinguishing between canonical characters and others is not trivial [2]; for the sake of clarity it is simplified in this discussion. The proposed distance  $d$  has some relation to the Levenshtein distance (a.k.a. edit distance) [12]. It is defined as follows.

**Definition 5.** For any two characters  $a$  and  $b$ , their distance  $d(a, b) \in \mathbb{R}$  is defined as

$$d(a, b) = \begin{cases} 0 & a, b \text{ canonical, } a = b \\ 1 & a, b \text{ canonical} \\ \min\{d(a, b_1) + d(\emptyset, b_2), \\ \quad d(\emptyset, b_1) + d(a, b_2)\} & a \text{ canonical} \\ \min\{d(a_1, \emptyset) + d(a_2, b), \\ \quad d(a_1, b) + d(a_2, \emptyset)\} & b \text{ canonical} \\ \min\{d(a_1, b_1) + d(a_2, b_2), \\ \quad d(a_1, b_2) + d(a_2, b_1), \\ \quad d(a_1, \emptyset) + d(a_2, b), \\ \quad d(a_1, b) + d(a_2, \emptyset), \\ \quad d(a, b_1) + d(\emptyset, b_2), \\ \quad d(\emptyset, b_1) + d(a, b_2)\} & \text{otherwise} \end{cases}$$

where  $a$  (resp.  $b$ ) is either canonical or of the form  $a = a_1 \bullet a_2$  (resp.  $b = b_1 \bullet b_2$ ), with  $\bullet$  a character decomposition operation (see Definitions 1 and 2).

**Proposition 1.** The  $d$  metric is a mathematical distance.

**Proof.** Consider any two characters  $a$  and  $b$ . Obviously,  $d(a, b) \geq 0$ . First, we show by recurrence that  $d(a, b) = 0 \Leftrightarrow a = b$ , which becomes thus our induction hypothesis. We show that this holds in the case  $a, b$  canonical; this is the base case of the recursion. Assume  $a = b$ . Since  $a$  canonical, we have  $d(a, b) = d(a, a) = 0$  directly. Assume  $d(a, b) = 0$ . Since  $a, b$  canonical, by definition we directly have  $a = b$ .

We show in the general case that  $d(a', b') = 0 \Leftrightarrow a' = b'$  holds with  $a' = a \bullet \tilde{a}$ ,  $b' = b \bullet \tilde{b}$  and  $\tilde{a}, \tilde{b}$  canonical. The case  $a' = \tilde{a} \bullet a$ ,  $b' = \tilde{b} \bullet b$  is shown similarly. By definition, and as per our

hypothesis, we have  $d(a', b') = d(a, b) + d(\tilde{a}, \tilde{b}) = d(\tilde{a}, \tilde{b})$ . So clearly,  $d(a', b') = 0 \Leftrightarrow d(\tilde{a}, \tilde{b}) = 0$  and  $d(\tilde{a}, \tilde{b}) = 0 \Leftrightarrow \tilde{a} = \tilde{b}$ .

Next, the equality  $d(a, b) = d(b, a)$  trivially holds since all the sub-character pair combinations are systematically exhausted by definition. Hence,  $a$  and  $b$  can be swapped freely.

Finally, we show that the triangle inequality  $d(a, c) \leq d(a, b) + d(b, c)$  holds for any characters  $a, b, c$ . We proceed by recurrence. We show that this holds in the case  $a, b, c$  canonical; this is the base case of the recursion. If  $a = b = c$ ,  $d(a, c) = d(a, b) = d(b, c) = 0$  and the hypothesis  $d(a, c) \leq d(a, b) + d(b, c)$  is satisfied. If  $a = b$  and  $a \neq c$ ,  $d(a, c) = d(b, c) = 1$  and  $d(a, b) = 0$ , and the hypothesis holds. If  $a = c$  and  $a \neq b$ ,  $d(a, b) = d(b, c) = 1$  and  $d(a, c) = 0$ , and the hypothesis holds. If  $a \neq b \neq c$ ,  $d(a, c) = d(a, b) = d(b, c) = 1$  and the hypothesis holds.

We show in the general case that  $d(a', c') \leq d(a', b') + d(b', c')$  holds with  $a' = a \bullet \tilde{a}$ ,  $b' = b \bullet \tilde{b}$ ,  $c' = c \bullet \tilde{c}$  and  $\tilde{a}, \tilde{b}, \tilde{c}$  canonical. The case  $a' = \tilde{a} \bullet a$ ,  $b' = \tilde{b} \bullet b$ ,  $c' = \tilde{c} \bullet c$  is shown similarly. First,  $0 \leq d(\tilde{a}, \tilde{b}) \leq 1$  since  $\tilde{a}, \tilde{b}$  canonical. Thus,  $d(a', b') \geq d(a, b) + d(\tilde{a}, \tilde{b})$  and  $d(a', b') - d(\tilde{a}, \tilde{b}) \geq d(a, b)$  for any  $a, b$ . Hence, we have by induction hypothesis

$$\begin{aligned} d(a', c') - d(\tilde{a}, \tilde{c}) &\leq d(a, c) \\ &\leq d(a, b) + d(b, c) \\ &\leq d(a', b') - d(\tilde{a}, \tilde{b}) + d(b', c') - d(\tilde{b}, \tilde{c}) \\ d(a', c') &\leq d(a', b') + d(b', c') \\ &\quad + d(\tilde{a}, \tilde{c}) - d(\tilde{a}, \tilde{b}) - d(\tilde{b}, \tilde{c}) \end{aligned}$$

It is recalled that  $0 \leq d(\tilde{a}, \tilde{c}), d(\tilde{a}, \tilde{b}), d(\tilde{b}, \tilde{c}) \leq 1$  since  $\tilde{a}, \tilde{b}, \tilde{c}$  canonical. Hence,  $d(\tilde{a}, \tilde{b}) = d(\tilde{b}, \tilde{c}) = 0$  holds if and only if  $\tilde{a} = \tilde{b} = \tilde{c}$ , and thus  $d(\tilde{a}, \tilde{b}) = d(\tilde{b}, \tilde{c}) = 0 \Rightarrow d(\tilde{a}, \tilde{c}) = 0$ . Therefore,  $d(\tilde{a}, \tilde{c}) - d(\tilde{a}, \tilde{b}) - d(\tilde{b}, \tilde{c}) \leq 0$  holds, and we have

$$d(a', c') \leq d(a', b') + d(b', c')$$

□

One should note that the properties shared between characters as distinguished previously in this section are ignored by the metric  $d$  so as to satisfy the triangle inequality. The sample distances calculated in Table 1 are expressed this time with the  $d$  metric in Table 3.

We now discuss the  $d$  metric. First, it is important to note that even though coherence is retained, the meaning of this distance metric is different from that of the previous ones: mainly, since the number of decompositions is counted, two unrelated canonical characters have distance 1, while they would be at distance  $\Omega$  with the previous metrics. This is a change of point of view: the less the number of decomposition operations, the shorter the distance.

Next, it should be noted that the distance expressed with this  $d$  metric is less “sharp” than when expressed with the previous metrics. Effectively, in comparison, many more instances of character pairs will have the same distance value. This is also symptomatic of the exclusion of character properties such as *radical* from the distance calculation.

Table 3: Examples of character distance calculations with the  $d$  metric

$a$	$b$	$d(a,b)$
洪	浜	$d(\dot{\gamma}, \dot{\gamma}) + d(\text{共}, \text{兵}) = d(\text{艹}, \text{丘}) + d(\text{八}, \text{八}) = 1$
榎	夏	$d(\text{木}, \emptyset) + d(\text{夏}, \text{夏}) = 1$
桜	櫻	$d(\text{木}, \text{木}) + d(\text{艹}, \text{嬰}) = d(\text{艹}, \text{嬰}) + d(\text{女}, \text{女}) = d(\text{艹}, \text{貝}) + d(\emptyset, \text{貝}) = d(\text{艹}, \text{目}) + d(\emptyset, \text{八}) + d(\emptyset, \text{目}) + d(\emptyset, \text{八}) = 4$
峠	雫	$d(\text{山}, \text{雨}) + d(\text{卩}, \text{下}) = 1 + d(\text{上}, \emptyset) + d(\text{下}, \text{下}) = 2$
湘	眼	$d(\dot{\gamma}, \text{艮}) + d(\text{相}, \text{目}) = 1 + d(\text{木}, \emptyset) + d(\text{目}, \text{目}) = 2$
木	林	$d(\text{木}, \text{木}) + d(\emptyset, \text{木}) = 1$
木	水	1
沐	浴	$d(\dot{\gamma}, \dot{\gamma}) + d(\text{木}, \text{谷}) = 1$

For example, the equality  $d(\text{洪}, \text{浜}) = d(\dot{\gamma}, \dot{\gamma}) + d(\text{共}, \text{兵}) = d(\text{共}, \text{兵})$  is satisfied: rather than counting the first horizontal combination operation and then subtracting some constant since these two characters share the same radical, the first decomposition operation directly results in no impact on the final distance value since involving two same sub-characters.

## 7 Conclusions and Future Works

Chinese characters are challenging for various reasons, their huge number being the most prominent. In this paper, we have first discussed the motivations behind our scientific approach to these characters. Then, in order to make this approach as versatile as possible, we have proposed an information model for Chinese characters, a rather innovative perspective in this field. Several points of view have been taken into account, including the phonetic, morphological and graphic ones. Importantly, we have considered the multi-lingual aspect of Chinese characters in our work. This information model has then been used to propose a novel concept: the definition of distance between any two characters, enabling further important applications such as character chains. An algorithm for character chain computation has been given and illustrated with examples. Nevertheless, defining such a distance metric with respect to Chinese characters is a complex topic, thus several metrics have been successively proposed and discussed. As each of these metrics has its own pros and cons, the choice of a distance metric would depend on the application considered.

As for future works, it would be meaningful to conduct a similar ontological discussion in the case of characters of other, unrelated scripts. For instance, Korea's Hangul script is one possible candidate, yet being much simpler to handle since being a phonic writing system, not using ideograms. Other script candidates include cuneiform writing as well as Egyptian hieroglyphs.

## References

- [1] Antoine Bossard, "Japanese Characters Cartography for Efficient Memorization," *International Journal of Computers and Their Applications*, 21(3):170–177, 2014.
- [2] Antoine Bossard, "Premises of an Algebra of Japanese Characters," *Proceedings of the Eighth International C\* Conference on Computer Science & Software Engineering*, Yokohama, Kanagawa, Japan, pp. 79–87, July 13–15, 2015.
- [3] Antoine Bossard and Keiichi Kaneko, "A Scientific Approach to Chinese Characters: Rationale, Ontology and Application," *Proceedings of the 29th International Conference on Computer Applications in Industry and Engineering*, Denver, CO, USA, pp. 111–116, September 2016.
- [4] Anne Castelain, *Direct Access Instant Kanji Dictionary*, Nichigai Associates, Tokyo, 1998.
- [5] Florian Coulmas, *Writing Systems of the World*, Wiley-Blackwell, Hoboken, NJ, 1991.
- [6] John DeFrancis, *Visible Speech: The Diverse Oneness of Writing Systems*, University of Hawai'i Press, Honolulu, 1989.
- [7] James W. Heisig, *Remembering the Kanji, Volume 1: A Complete Course on How Not to Forget the Meaning and Writing of Japanese Characters*, University of Hawai'i Press, Honolulu, 2010.
- [8] James W. Heisig and Timothy W. Richardson, *Remembering Simplified Hanzi, Volume 1: How Not to Forget the Meaning & Writing of Chinese Characters*, University of Hawai'i Press, Honolulu, 2012.
- [9] James W. Heisig and Timothy W. Richardson, *Remembering Traditional Hanzi, Volume 1: How Not to Forget the Meaning & Writing of Chinese Characters*, University of Hawai'i Press, Honolulu, 2012.
- [10] Kenneth G. Henshall, *A Guide to Remembering Japanese Characters*, Tuttle Publishing, 1988.
- [11] Japanese Industrial Standards Committee (JISC), *7-Bit and 8-Bit Coded Character Sets for Information Interchange (7ビット及び8ビットの情報交換用符号化文字集合)*, in Japanese, 1969.
- [12] Vladimir I. Levenshtein, "Binary Codes Capable of Correcting Deletions, Insertions, and Reversals," *Soviet Physics Doklady*, 10(8):707–710, 1966.
- [13] Geoffrey Sampson, *Writing Systems*, Equinox Publishing, Sheffield, 2015.
- [14] Shizuka Shirakawa, *Creating a Dictionary (字書を作る, in Japanese)*, Heibonsha, Tokyo, 2002.
- [15] Richard Sproat, *A Computational Theory of Writing Systems*, Cambridge University Press, Cambridge, United Kingdom, 2000.
- [16] Oreste Vaccari and Enko Elisa Vaccari, *Pictorial Chinese-Japanese Characters: A New and Fascinating Method to Learn Ideographs*, C. E. Tuttle Co., Tokyo, 1958.

- [17] Wikimedia Commons, *Ancient Chinese Characters Project*, [https://commons.wikimedia.org/wiki/Commons:Ancient\\_Chinese\\_characters\\_project](https://commons.wikimedia.org/wiki/Commons:Ancient_Chinese_characters_project), last accessed December 2016.



**Antoine Bossard** is an Assistant Professor of the Graduate School of Science, Kanagawa University, Japan. He received the B.E. and M.E. degrees from Université de Caen Basse-Normandie, France in 2005 and 2007, respectively, and the Ph.D. degree from Tokyo University of Agriculture and Technology, Japan in 2011. His research is focused on graph theory, interconnection networks and dependable systems. For several years, he has also been conducting research regarding Chinese characters and their memorization. He is a member of ACM, ACIS and ISCA.



**Keiichi Kaneko** is a Professor at Tokyo University of Agriculture and Technology. His main research areas are dependable systems, interconnection networks, functional programming, parallel and distributed computation, partial evaluation and educational systems. He received the B.E., M.E. and Ph.D. degrees from the University of Tokyo in 1985, 1987 and 1994, respectively. He is a member of IEEE, ACM, IEICE, IPSJ and JSSST.

# Data Lossless Compression Using Improved GFC Algorithm with Multiple GPUs

Rui Wu\*

University of Nevada Reno, Reno, Nevada USA

Muhanna Muhanna<sup>†</sup>

Princess Sumaya University for Technology, Amman, JORDAN

Sergiu M. Dascalu\*, Lee Barford<sup>\*‡</sup>, Frederick C. Harris, Jr\*

University of Nevada Reno, Reno, Nevada USA

## Abstract

Compression is widely used in both scientific research and industry. The most common use is that people compress the backup data and infrequently used data to save space. Compression is significantly meaningful for big data because it will save a lot of resources with the help of a good compression algorithm. There are two criteria for a good compression algorithm—compression ratio and time consumption. GFC is one of the fastest compression algorithms with a mediocre compression ratio, which is designed for real-time compression with the help of Graphics Processing Units (GPU). This paper introduces three methods to increase the speed of GFC algorithm by using the `clzll` function, removing if-else statements, and using multi-GPUs. The first and third methods improve the original algorithm performance. However, the if-else-removal method cannot always guarantee better results. The final compression speed is more than 1,000 gigabits/s, which is much faster than 75 gigabits/s—the original GFC algorithm speed.

**Key Words:** GFC; lossless compression; high-speed; floating-point data.

## 1 Introduction

Big data and its management is a hot topic for both businessmen and scientists. The digital era brings us many opportunities and also tons of problems. Almost every device keeps generating data all the time. For example, the Large Synoptic Survey Telescope (LSST) needs to manage over 100 PB of data [4]. The Facebook warehouse stores upwards of 300 PB with a daily incoming rate around 600 TB [16]. There are 300 hours of video material uploaded to YouTube every minute [6]. However, it is hard to manage and analyze big data. To uncover the “gold mines” buried in these datasets,

researchers hold many conferences to resolve these hard big data problems, such as XLDB [15].

Compression is one of the keys to manage big data and it helps businessmen and scientists save resources. One of the most common rules is that the data management system will compress data if the data is not frequently used. If a compression algorithm compresses original data 20% smaller than before, it means people can save 20% more space, which means a lot for petabyte-scale datasets. Therefore, a good compression algorithm is significant to a big data project. Also, compression is very significant for some big data web-based application. Dr. Holub and his colleagues introduced a method about how to transmit HD, 2K, and 4K videos with the low-latency network in their paper [7]. The core idea of this project is to compress and decompress JPEG efficiently with the help of GPUs. Figure 1 displays a simplified network diagram of the pilot deployment of their project [7].

There are many mature and good CPU compression algorithms. Some of them are designed for image compressions, such as JPEG [17], some of them are designed for audio and video compression, such as MPEG [10], and some of them are for general use, such as LZ4 [3]. Also, some scientists tried to take advantage of GPU to increase the speed of CPU compression algorithms. For example, [2] tried to improve the Huffman compression algorithm using GPU.

GPU is short for Graphics Processing Unit. It is originally designed for computer graphics and image processing, and it is very popular in high-performance computing today. Also, there is a trend that scientists use multi-GPUs, instead of a single GPU to improve performances of different algorithms. However, GPU is not suitable for all kinds of algorithms. If an algorithm is not parallelizable or highly divergent, it is better not to use GPU.

Here are some reasons that we chose GFC instead of other algorithms. First, GFC is one of the fastest existing lossless compression algorithms. The original algorithm is 75 gigabits/s [14]. It is gigabit, instead of gigabyte, because the core ideas of GFC algorithm are based on bitwise operations. The speed is much faster than most other compression algorithms.

For example, LZ4 is around 14.56 gigabits/s [3], which is much slower than wide-band network speed. If we do not choose a fast algorithm for high-speed web-based

\* Department of Computer Science and Engineering. Email: {rui, dascalus, fred.harris}@cse.unr.edu.

<sup>†</sup>Department of Computer Graphics. Email: m.muhananna@psut.edu.jo.

<sup>‡</sup> Keysight Laboratories, Keysight Technologies, Reno, NV. Email: lee\_barford@ieee.org.

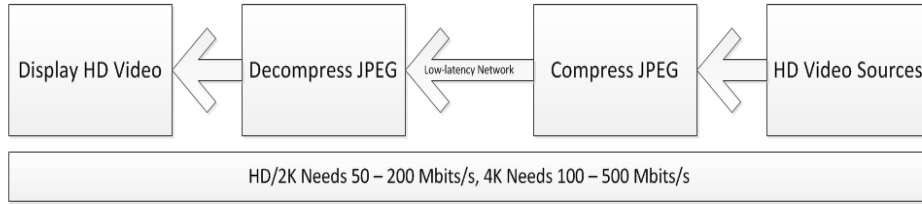


Figure 1: Transfer HD videos with slow network by compressing each frame in the server side and uncompressing the frame in the client side

applications, the algorithm will slow down the throughput of these applications. Second, GFC is designed for GPU directly. To contrast to GFC, most of the GPU algorithms are converted from CPU algorithms, which means some compromises have to be made and it will have a negative impact on the algorithm performance most of the time. Third, GFC aims to compress large datasets, which is critical for both business and scientific uses.

Some basic concepts about GPU, such as grid, block, warp, and thread can be found in the paper [12] and Figure 2 dis-

plays a common GPU structure, which presents the relations between threads, blocks, and grids. Different GPU video card structures may be different from each other, but they all share some common features: if users want their GPU algorithms to perform best, they have to use all the threads in a warp; if different threads, in the same block need to communicate with each other, programmers can use shared memory; if different threads, in different blocks need to communicate with each other, programmers can use global memory.

The rest of this paper is organized as follows in the remain-

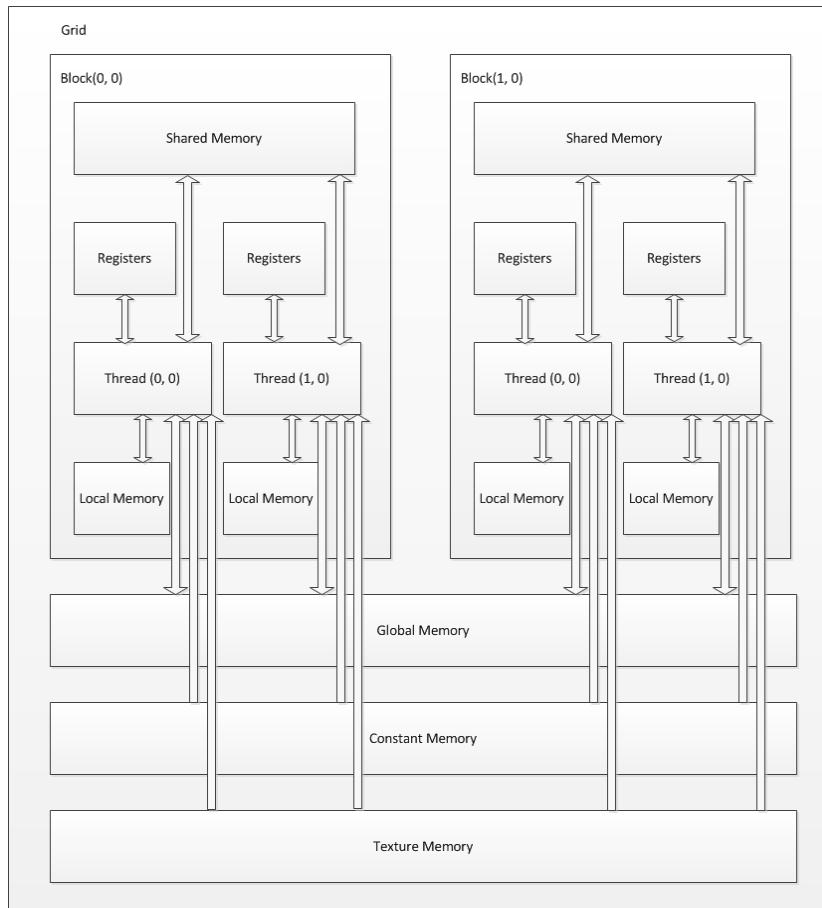


Figure 2: GPU structure. Threads in different blocks should try to avoid communicating with each other because it cannot use local memory and performance is not good

remaining part: Section 2 introduces the original GFC algorithm; Section 3 introduces our three methods to improve GFC algorithm; Section 4 introduces the results and our opinions about these results; Section 5 concludes the main ideas of this paper.

### 2 Original GFC Algorithm

GFC is a lossless double-precision floating-point data compression algorithm. It is designed for GPU specifically. By using [9], GFC algorithm replaces 64-bit floating-point values with 64-bit integers. Therefore, GFC needs only integer operations, although it compresses floating-point datasets.

Overview of warp, block and chunk assignment of GFC is displayed in Figure 3. The uncompressed data is separated into  $r$  chunks and each chunk contains 32 doubles. Each chunk is processed by one warp in the GPU. After all warps finish compressing the assigned chunk, GFC combines all the results together, which is compressed data. The reason that each chunk contains 32 doubles is that there are 32 threads in each warp for most of GPU video cards and it is most effective when a program uses all the threads in a warp.

Figure 4 presents the details about GFC compression algorithm. According to GFC, we need to subtract  $p$ , which is in the previous chunk, from  $i$ , which is in the current chunk, and  $p = 32 - (dim - 1 \% dim)$  [14]. Dim means “dimension” in this equation. If the subtraction is negative, we need to use operation—negate to make it positive. The magic part of GFC is the rectangle named residual in the bottom part of Figure 4. By counting the leading zeros of this part,

removing these zeros, and adding the leading zeros metadata, GFC compresses the original datasets. The most significant theory behind GFC algorithm is that most scientific datasets interleave values from multiple dimensions [14]. For example, weather temperature will follow a pattern each year for most of the time, which means temperature scientific data can have many leading zeros by using GFC compression algorithm. Users need to find the interleave orders, gets the maximum leading zeros and removes them to have the highest compression ratio.

It is possible that the compressed data is larger than the original data using GFC compression algorithm if we choose a bad interleave dimensionality. For example, all the eight bytes of residuals are non-zeros and it results in the output sub-chunk being 16 bytes larger than the original chunk, which is 6% larger than the original part [14]. Before users use GFC compression with their data, it is better to preprocess their data and find out the suitable data interleave dimensionality to obtain the best performance.

O’Neil and Burtcher created GFC and published this algorithm in [14]. They avoided using long if-else statements and assigned datasets reasonably according to the structure of GPU to improve the performance of their algorithm. If-else statements can slow down a program, especially a GPU program. This is because of the structure of video cards. Each warp has 32 threads (for most video cards) and all these threads (in the same wrap) must execute the same instruction in one cycle [12]. When these threads execute If-else statements, some threads may fulfill the if statement and execute that part of the code, and the remaining threads will.

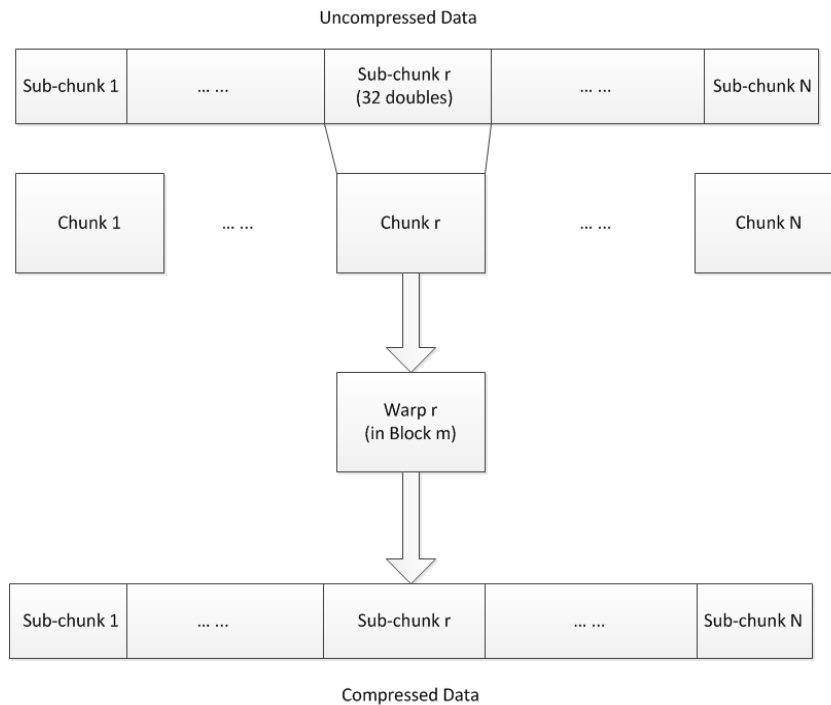


Figure 3: Overview of GFC algorithm warp, block, and chunk assignment. Each warp is assigned 32 doubles because there are usually 32 threads in each warp

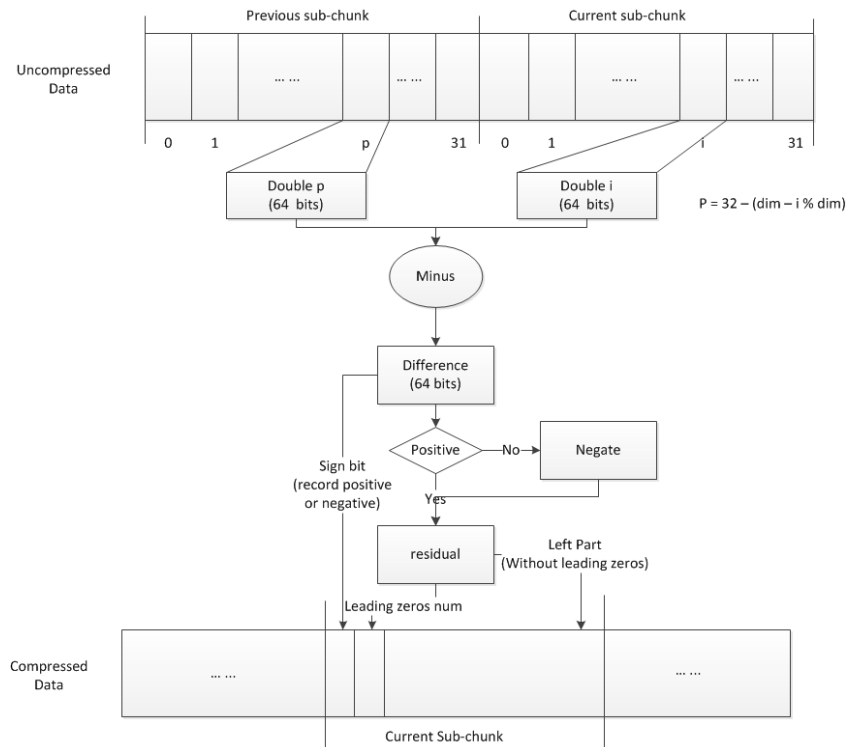


Figure 4: GFC compression algorithm  
The original file is shrunk by removing the leading zeros.

stay idle, which means threads are not fully used. Therefore, GFC avoids using long if-else statements

The line chart is not always above zero. This means “if-else-removal” method cannot always improve the performance.

### 3 Improved GFC Algorithm

We tried to improve the performance of GFC algorithm with three methods: 1) using `clzll` to count the leading zeros; 2) removing if-else statements in the program; 3) using multi-GPUs.

#### 3.1 Clzll

In the summary and conclusions part of [14], the authors mentioned that they wrote their own function to count the leading zeros, because their video card was GTX-285 and it does not support `clzll`, which is used to count the number of consecutive leading zeros bits, starting at the most significant bit (bit 63) of `x` [13]. They believe GFC could be improved by using `clzll` to count the leading zeros to replace their code. We agree with their idea because professional programmers in Nvidia know secrets of their video cards. Therefore, it is not strange that their GPU functions are more suitable to the structure of video cards and more effective than our codes. The results in Section 5 also prove this idea is right.

#### 3.2 If-Else-Removals

In our opinion, if-else statements can slow down programs, especially for GPU programs. Because if-else statements will make some of the threads in a warp idle, when these threads cannot fulfil the if-else statement. Here is an example presented in Figure 5:

```

if a < 3 then
    a = 7
else
    if a >= 3 then
        a = 5
    end if
end if
    
```

Figure 5: If-else Statement Example

Each warp has 32 threads (for most current video cards). Only the threads that fulfil the condition,  $a > 3$ , they will execute  $a = 7$ . Other threads will be idle till the whole warp goes through this if-statement.

There are some materials, such as [11], proving long if-else statements will also have a negative impact on the performance of normal programs. Therefore, we tried to remove if-else statements in GFC algorithm by using bitwise operations. Here is an example, as Figure 6 displays:

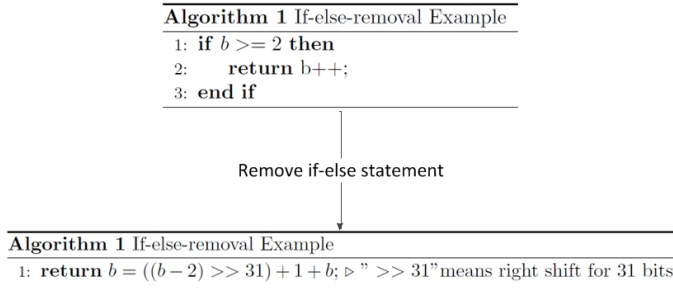


Figure 6: If else-removal example, less lines but more complex

“>>31” means a right shift for 31 bits. For most cases, signed integers have 32 bits and the left most bit is used for a sign (positive or negative).  $(b - 2) \gg 31$  is -1 when  $b - 2$  is negative and it is 0 when  $(b - 2) \gg 31$  is positive. Therefore, the two statements are the same in Figure 6.

However, we found when if-else statement is short (for example, there is just one line of statement under “if”), the replacement of if-else statements with bitwise operations will slow down the program. We think it may be because something undisclosed in the compiler to optimize the program. The authors of [14] also tried to avoid long if-else statements in their program, except one part in the decompress kernel. Therefore, we replaced that part with bitwise operations as Figure 7 shows.

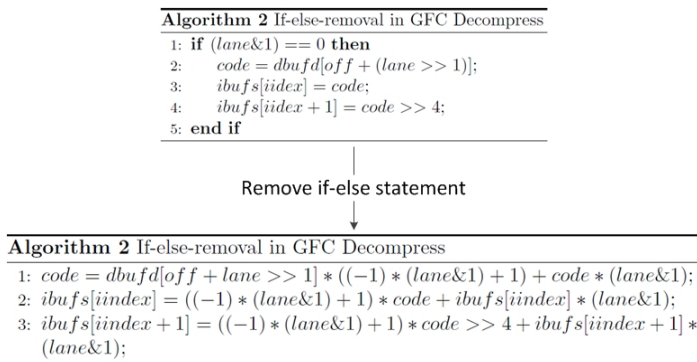


Figure 7: If-else-removal in GFC decompress

But, the method cannot guarantee better results all the time. Figure 8 displays the delta time between the original algorithm and the improved algorithm for a dataset named `obs_info`. When the line is above zero, it means the improved algorithm is faster. Even if the improved algorithm is better, the improvement is not really obvious. Therefore, we don’t apply this method in the final improved algorithm. In our opinion, the reasons that this method does not improve the performance are that each thread needs to spend more time than before because the code is more complex and the total time consumption is worse, even if there are no idle threads in the wrap.

### 3.3 Multi-GPUs

After reading some GPU technique papers, we found that

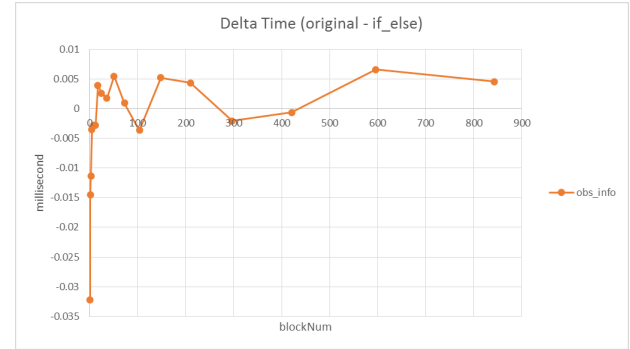


Figure 8: If-else-removal time delta

some authors try to improve the performance of an algorithm by parallelizing the algorithm and others try to enhance an algorithm by parallelizing tasks. For example, in [8], the author proposed to separate strings and assign a thread for each segment to increase the speed of Boyer-Moore algorithm. We also found there was a trend that scientists used multi-GPUs instead of a single GPU to improve their algorithms.

We found the task—compression is parallelizable. “Parallelizable” means that we can separate the task into several parts and each part can be processed independently. GFC is a GPU algorithm and it uses both blocks and threads. Therefore, we need to assign a GPU for every segment to enhance the performance. So we tried to use multi-GPUs instead of single GPU and the basic idea is displayed in Figure 9. The uncompressed dataset is separated into  $N$  chunks, each chunk is processed by a GPU, and each GPU processes the assigned data with GFC algorithm. After all the GPUs finish their jobs, a CPU will combine the results together, which is the compressed data.

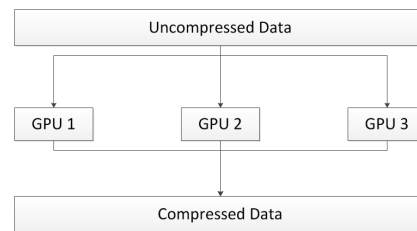


Figure 9: Multi-GPUs method

## 4 Results

We did experiments with a Cubix machine, which has eight GeForce GTX 780 video cards, Intel(R) Xeon(R) CPU E5-2620 @ 2.00GHz, and PCI 3.0.

All the flowing experiment datasets are offered by Martin Burtscher, who is one of the authors of [14]. The datasets can be downloaded in [1]. From our experiences about GPU programming, the best results of different problems need different numbers of blocks and threads. After experiments with four of these datasets, we found that we need to use all the threads in the chosen number of blocks to get the best results (throughputs). Therefore, we only did experiments to find the



best number of blocks for each dataset and used all the threads. All the experiments were ran 11 times and we chose the median value of these 11 results to be the final result. For example, in multi-GPUs part, we tested different numbers of blocks for a dataset named obs\_info. We did the same experiment 11 times and finally found we should use 51 blocks and all the threads in these blocks to get the maximum throughput 1073.376 gigabits/s.

Because [14] mentioned that PCIe bus is too slow for GFC (compression speed is limited to 8GB/s [5]), O'Neil and Burtscher did not record the time of transferring data from CPU to GPU. Therefore, we did not do that for all the following experiments. We also compared decompressed files with original files to make sure that our methods do not change files.

#### 4.1 Clzll

The first improvement is to use `__clzll()`, which is used to count the number of consecutive leading zeros bits, starting at the most significant bit (bit 63) of `x` [13]. The results are presented in Figure 10.

In Figure 10, we subtracted original GFC's throughput from improved GFC's throughput. And we found most of the time, the deltas are above zero, which means the improved algorithms' throughput are better. This proves the idea that is introduced in Section 3.1.

#### 4.2 Multi-GPUs

We did the experiments with one, two, four, and eight GPUs to study the relation between the number of GPUs and the speedup. We recorded time consumptions of each GPU and used the maximum time to be the final time consumption. For example, we used 8 GPUs and GPU1 spent  $T_1, \dots, GPU_2$  spent

$T_2 \dots GPU_8$  spent  $T_8$ . The final time consumption was  $\text{Max}(T_1, T_2, \dots, T_8)$ . We used the maximum time for the final time because we set up a synchronizing point, which resulted in GPUs waiting for others until all the GPUs finish their jobs. Table 1 displays the throughputs (gigabits/s) of a dataset named num\_plasma. To save time, we did not do the experiment with block number from 1 to 1024. The step of BlockNum in Table 1 is  $\text{int}(\sqrt{2})$ .

Table 1: Num\_plasma throughputs

BlockNum	8-GPU	4-GPU	2-GPU	1-GPU
1	159.26	81.40	41.18	21.25
2	304.68	158.78	81.51	42.00
3	436.46	233.39	120.21	62.06
5	668.01	376.61	196.12	102.86
8	987.06	572.33	304.24	159.44
12	1,233.31	804.09	438.55	233.19
17	1,214.70	768.86	420.02	219.24
25	1,219.77	715.74	386.17	202.43
36	1,212.85	803.42	438.75	233.02
51	1,268.61	815.37	465.57	250.08
73	1,365.97	955.67	541.73	261.71
104	1,381.89	876.36	481.61	258.48
148	1,312.64	871.23	523.82	264.98
210	1,266.23	860.80	500.14	274.39
297	1,214.87	838.03	496.44	274.89
421	1,170.78	818.49	480.72	266.15
596	1,140.80	743.96	457.01	264.19
843	1,079.34	715.57	439.54	253.56

Table 2 presents the maximum throughputs of different number of GPUs. From this table, we can tell that the speedup is better with more GPUs. However, the relationship between the speedup and the GPU number is not linear. For example, 8-GPU speedup does not equal eight times 1-GPU speedup. In

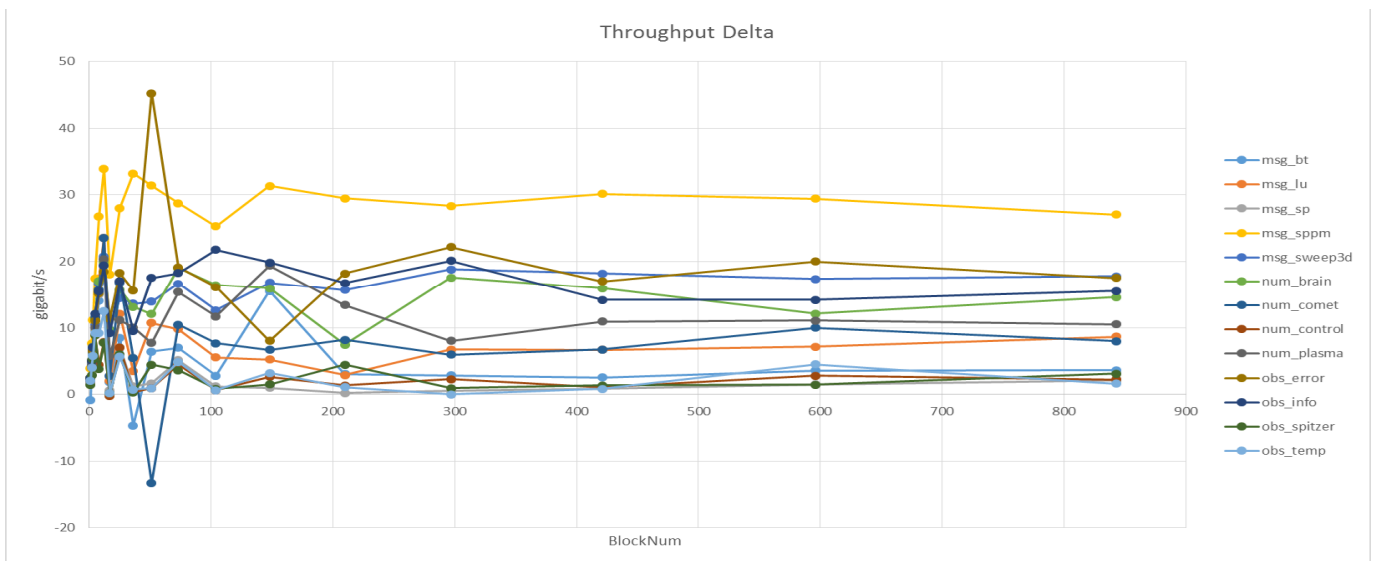


Figure 10: Clzll throughput delta.

Most cases on the line charts are above zero. This means "Clzll" function can improve the performances.

our opinions, this is because of the more GPUs we have, the more segment file will be generated (our program will separate the original file into N parts and each GPU is in charge of a segment). Our program needs to combine all the segment files together to be the final compressed file in the last compression step, which is done by a CPU sequentially. This step will use more time if we have more segment files.

Table 2: Maximum throughput

Name	Max Throughput (gigabits/s)	BlockNum	Speedup
8-GPU	1,381.89	104	5.03
4-GPU	955.67	73	3.48
2-GPU	541.73	73	1.97
1-GPU	274.89	297	1.00

Figure 11 visualizes the relation between the throughputs of each number of GPUs with a line chart. For each line in Figure 11, we found they went up first and then went down, which means that too many blocks will reduce the throughputs (gigabit/s) after a certain threshold. When the blocks number is small, N GPUs will increase the throughput almost N times. However, when the blocks number is increased, the speedup is less than N times. We think it may be because of the impact of blocks, as we just discussed. This negative impact will reduce the gap between each of the multi-GPUs results. Therefore, the final results are less than N times, when the blocks number is large.

### 4.3 Final Improved GFC Algorithm

Finally, we combined two methods—clzll and multi-GPUs together to improve GFC. We did experiments to datasets from [1] and obtained speedup results (the improved GFC algorithm over the original GFC algorithm) as Figure 12 presents.

The maximum speedup of the improved GFC algorithm is 8.705 and the maximum throughput of the improved GFC algorithm is 2454.603 gigabits/s, which is much faster than original GFC throughputs in [10]. Of course, the good result is partially because we used better hardware than the original GFC paper.

### 5 Conclusion and Future Work

In this paper, we introduced three methods to increase the speed of a lossless compression algorithm named GFC. These three methods are: 1) using clzll to count the leading zeros; 2) replacing if-else statements with bitwise operations in the program; 3) using multi-GPUs instead of a single GPU.

After some experiments with datasets downloaded from [15], we found 1) and 3) were effective and the maximum speedup is 8.705 and the maximum throughput of the improved GFC algorithm is 2,454.60 gigabits/s, by using 1) and 3) together. However, 2) cannot guarantee good results all the time.

In the future, we want to do more experiments to find out the rules between the performance and number of blocks and GPUs. For example, an equation can obtain the number of blocks and GPUs for a specific problem to get the best results done sequentially using a CPU core. We have designed a new

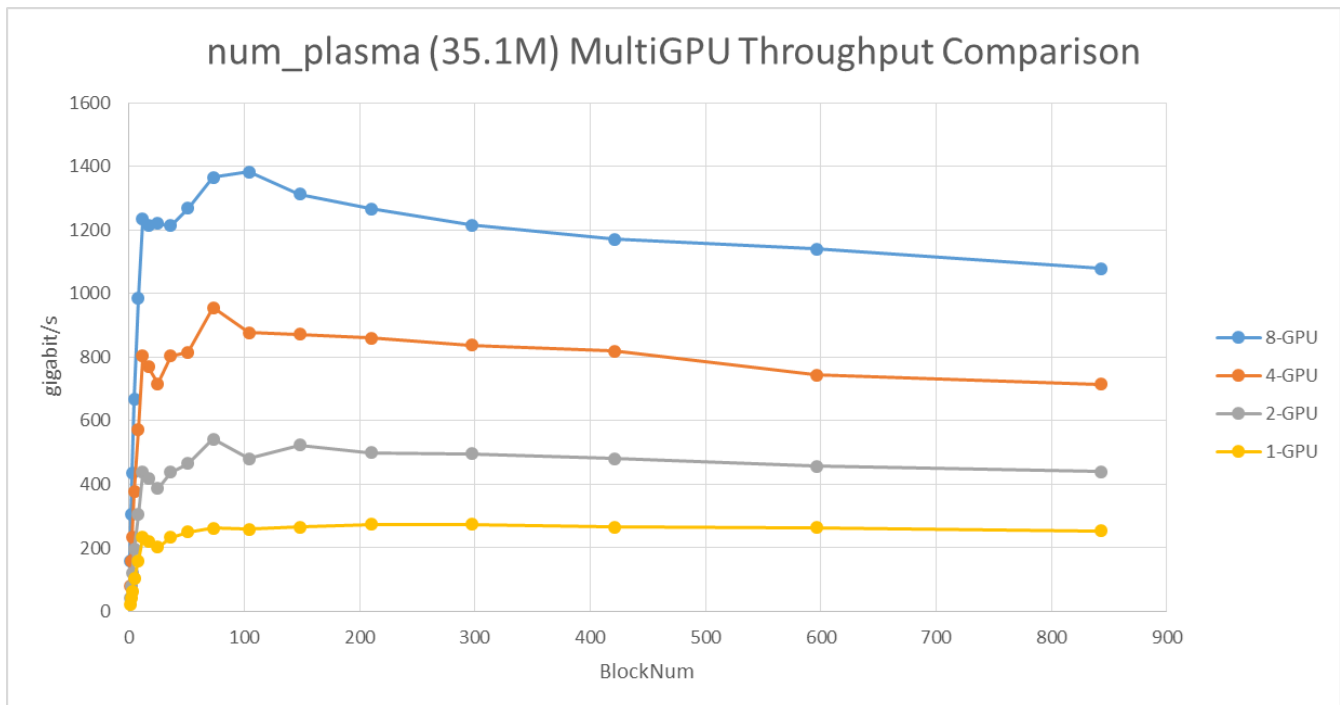


Figure 11: Multi-GPUs throughput of num\_plasma

(throughputs). The last step of our method is to combine all the compressed file segments into the final compressed file. This is method to do it parallel using multiple CPU cores. Figure 13 presents the details of this method. The basic idea is to use one CPU core to combine two compressed file segments.

Therefore, we can use N CPU cores to combine 2N file segments in one step. We also want to extend our previous work introduced in [18, 19, 20] with the improved data compression algorithm.



Figure 12: Speedup of improved GFC algorithm  
The speedups of most cases are above 4 for all the datasets

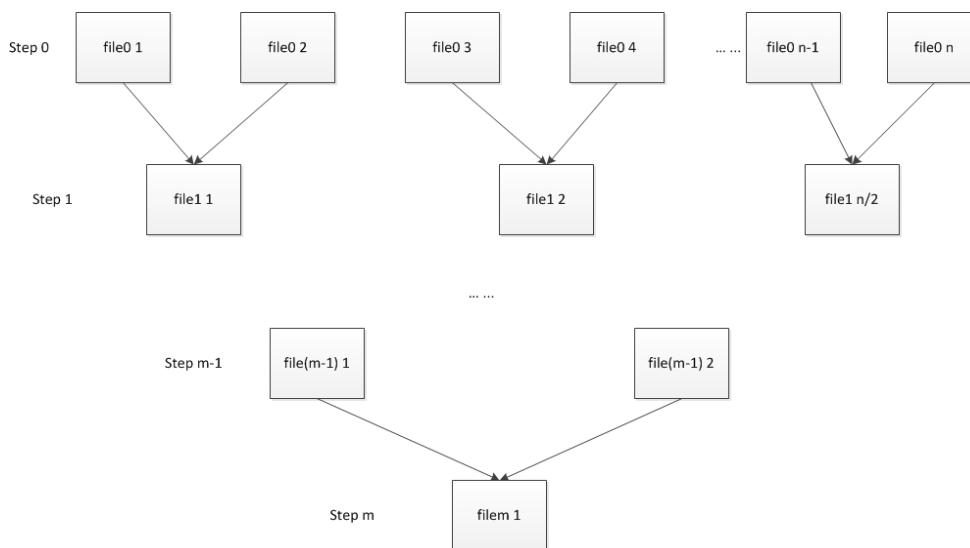


Figure 13: Segment files combination in parallel

### Acknowledgements

The authors of this paper acknowledge the help from O'Neil and Burtscher. They kindly answered some hard questions about GFC by email and offered us test datasets.

This material is based upon work supported in part by The National Science Foundation under grant numbers IIA-1301726 and IIA-1329469, and by Cubix Corporation through use of their PCIe slot expansion hardware solutions and HostEngine.

Any opinions, finds, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation or Cubix Corporation.

### Reference

- [1] M. Burtscher, "Martin Burtscher/FPdouble," <http://cs.txstate.edu/~burtscher/research/datasets/FPdouble/>, (accessed 5/5/2015).
- [2] R. L. Cloud, M. L. Curry, H. L. Ward, A. Skjellum, and P. Bangalore, "Accelerating Lossless Data Compression with GPUs," *arXiv*, 3:26-29, 2009.
- [3] Y. Collet, "LZ4-Extremely Fast Compression Algorithm," <https://code.google.com/p/lz4/>, (accessed 5/4/2015).
- [4] P. Cudré-Mauroux, H. Kimura, K. T. Lim, J. Rogers, R. Simakov, E. Soroush, P. Velikhov, D. L. Wang, M. Balazinska, J. Becla, D. DeWitt, B. Heath, D. Maier, S. Madden, J. Patel, M. Stonebraker and S. Zdonik, "A Demonstration of SciDB: A Science-Oriented DBMS" *Proceedings of the VLDB Endowment*, 2(2):1534-1537, 2009.
- [5] A. Eirola, Lossless Data Compression on GPGPU Architectures," *arXiv preprint arXiv: 1109.2348*, 2011.
- [6] Google Inc., "Statistics—YouTube," <https://www.youtube.com/yt/press/statistics.html>, (accessed 5/4/2015).
- [7] P. Holub, M. Šrom, M. Pulec, J. Matela, and M. Jirman, "GPU-Accelerated DXT and JPEG Compression Schemes for Low-Latency Network Transmissions of HD, 2K, and 4K Video," *Future Generation Computer Systems*, 29(8):1991-2006, 2013.
- [8] M. Jaiswal, "Accelerating Enhanced Boyer-Moore String Matching Algorithm on Multicore GPU for Network Security," *International Journal of Computer Applications*, 97(1):30-35, 2014.
- [9] W. Kahan, Lecture Notes on the Status of IEEE Standard 754 for Binary Floating-Point Arithmetic." *Manuscript*, 30 pp., May 1996.
- [10] D. Le Gall, "MPEG: A Video Compression Standard for Multimedia Applications," *Communications of the ACM*, 34(4), 46-58, 1991.
- [11] S. Loinel, "Does a Lot of "if ... else" Statements Slow Down the Code?" <https://software.intel.com/en-us/forums/topic/283268>, (accessed 5/5/2015).
- [12] J. Luitjens and S. Rennich, "CUDA Warps and Occupancy," *GPU Computing Webinar*, 11:2-19, 2011.
- [13] NuDoq. "NuDoq – CUDAFy.NET," <http://www.nudoq.org/#!/Packages/CUDAFy.NET/Cudafy.NET/IntegerIntrinsicsFunctions/M/clzll>, (accessed 5/5/2015).
- [14] M. A. O'Neil and M. Burtscher, "Floating-Point Data Compression at 75 Gb/s on a GPU," *Proceedings of the Fourth Workshop on General Purpose Processing on Graphics Processing Units*, ACM, p. 7, 2011.
- [15] M. Stonebraker, J. Becla, D. J. DeWitt, K. T. Lim, D. Maier, O. Ratzesberger, and S. B. Zdonik, "Requirements for Science Data Bases and SciDB," *Proceedings of the Fourth Biennial Conference on Innovative Data System*, 7:173-184, January 2009.
- [16] P. Vagata. and K. Wilfong, "Scaling the Facebook Data Warehouse to 300 PB," <https://code.facebook.com/posts/229861827208629/scaling-the-facebook-data-warehouse-to-300-pb/>, (accessed 5/4/2015).
- [17] G. K. Wallace, "The JPEG Still Picture Compression Standard," *Communications of the ACM*, 34(4):30-44, 1991.
- [18] M. Zhang, T. Yang, and R. Wu, "Space-Efficient Multiple String Matching Automata," *International Journal of Wireless and Mobile Computing*, 5(3):308-313, 2012.
- [19] R. Wu., S. Dascalu, and F. Harris, (2015) Environment for Datasets Processing and Visualization Using SciDB. *Proceedings of the 24th International Conference on Software Engineering and Data Engineering (SEDE 2015)*, San Diego, CA, pp. 223-229, October 12-14, 2015.
- [20] R. Wu, C. Chen, S. Ahmad, J. Volk, C. Luca, F. Harris, and S. Dascalu, "A Real-time Web-Based Wildfire Simulation System," *Proceedings of the 2016 IEEE Industrial Electronics Conference (IECON 2016)*, Florence, Italy, Oct 24-27, 2016.



**Rui Wu** is a Ph.D. student in the Department of Computer Science and Engineering at the University of Nevada, Reno, USA. He started the Ph.D. program in Spring 2014 after obtaining in 2013 a Bachelor's degree in Computer Science and Technology from Jilin University, China. His main research interests are in data analysis, data visualization, and software engineering.



**Muhanna Muhanna** is an Assistant Professor in the Department of Computer Graphics at Princess Sumaya University for Technology, Jordan, which he joined in 2011 after receiving his Ph.D. in Computer Science and Engineering from the University of Nevada, Reno earlier that year. In 2007, he received his M.S. in Computer Science from the University of Nevada, Reno as well. His main

research interests are in human-computer interaction, user experience, and software engineering. Moreover, he has been the Assistant President for Accreditation and Quality Assurance at Princess Sumaya University for Technology since 2013.



**Sergiu Dascalu** is a Professor in the Department of Computer Science and Engineering at the University of Nevada, Reno, USA, which he joined in 2002. In 1982 he received a Master's degree in Automatic Control and Computers from the Polytechnic University of Bucharest, Romania and in 2001 a Ph.D. in Computer Science from Dalhousie University, Halifax,

NS, Canada. His main research interests are in the areas of software engineering and human-computer interaction. He has published over 140 peer-reviewed papers and has been involved in numerous projects funded by industrial companies as well as federal agencies such as NSF, NASA, and ONR.



**Lee Barford** is a Fellow at Keysight Laboratories and Professor of Computer Science and Engineering (adjunct) at the University of Nevada, Reno, Nevada. He leads research to identify and apply emerging technologies in software, applied mathematics, and statistics to enable new kinds of measurements and increase measurement accuracy and speed. Lee's work has been used to improve R&D productivity and reduce manufacturing cost in the leading companies in the technology and transportation industries, including Apple, Boeing, Cisco, Ford, HP, Microsoft, and NASA. Dr. Barford has given invited talks at universities worldwide, including MIT, Cambridge, Stanford, and Tsinghua. Previously, he managed a number of research projects at Agilent Laboratories and Hewlett-Packard Laboratories, for example in visible light and X-ray imaging systems, calibration methods for non-linear and dynamical disturbances, and fault isolation from automatic test equipment results. He is the author of over 50 peer-reviewed publications and inventor of approximately 60 patents.



**Frederick C. Harris, Jr.** is a Professor in the Department of Computer Science and Engineering and the Director of the High Performance Computation and Visualization Lab and the Brain Computation Lab at the University of Nevada, Reno, USA. He received his B.S. and M.S. degrees in Mathematics and Educational Administration from Bob Jones University in 1986 and 1988 respectively, and his M.S. and Ph.D. degrees in Computer Science from Clemson University in 1991 and 1994 respectively. He is a Senior Member of ACM and ISCA, and a member of IEEE. His research interests are in parallel computation, computational neuroscience, computer graphics and virtual reality.

## Index

## Authors

## A

- Akour, Mohammed**, see Banitaan, Shadi; *IJCA v23 n1 March 2016 29-34*
- Anderson, John W.**, see Carthen, Chase; *IJCA v23 n3 Sept 2016 195-207*
- Arad, Behnam S.**, see Durai, Maththaiya; *IJCA v23 n2 June 2016 87-95*
- Assefi, Mehdi**, Guangchi Liu, Mike P. Wittie, and Clemente Izurieta; Measuring the Impact of Network Performance on Cloud-Based Speech Recognition; *IJCA v23 n1 March 2016 19-28*

## B

- Badri, Linda**, see Flageol, William; *IJCA v23 n1 March 2016 69-76*
- Badri, Mourad**, see Flageol, William; *IJCA v23 n1 March 2016 69-76*
- Banitaan, Shadi**, Kevin Daimi, Mohammed Akour, and Yujun Wang; Test Suite Selection in Junit Testing Environment Based on Software Metrics; *IJCA v23 n1 March 2016 29-34*
- Barford, Lee**, see Wu, Rui; *IJCA v23 n4 Dec 2016 232-241*
- Bossard, Antoine** and Keiichi Kaneko; Chinese Characters Ontology and Induced Distance Metrics; *IJCA v23 n4 Dec 2016 223-231*
- Bossard, Antoine** and Les Miller; Guest Editor's Note; *IJCA v23 n2 June 2016 77*
- Burfield, Nolan P.**, see Carthen, Chase; *IJCA v23 n3 Sept 2016 195-207*

## C

- Carthen, Chase**, Thomas J. Rushton, Nolan P. Burfield, Donna Delparte, Tucker Chapman, W. Joel Johansen, Roger Lew, Nicholas R. Wood, Mathew Ziegler, John W. Anderson, Sergiu M. Dascalu, and Frederick C. Harris, Jr.; Virtual Watershed Visualization for the WC-WAVE

Project; *IJCA v23 n3 Sept 2016 195-207*

- Chapman, Tucker**, see Carthen, Chase; *IJCA v23 n3 Sept 2016 195-207*
- Codabux, Zadia**, see Deo, Ajay K.; *IJCA v23 n1 March 2016 35-56*

## D-G

- Daimi, Kevin**, see Banitaan, Shadi; *IJCA v23 n1 March 2016 29-34*
- Dascalu, Sergin M.**, see Carthen, Chase; *IJCA v23 n3 Sept 2016 195-207*
- ...see Wu, Rui; *IJCA v23 n4 Dec 2016 232-241*
- Delparte, Donna**, see Carthen, Chase; *IJCA v23 n3 Sept 2016 195-207*
- Deo, Ajay K.**, Zadia Codabux, Kazi Zakia Sultana, and Byron J. Williams; Assessing Software Defects Using Nano-Patterns Detection; *IJCA v23 n1 March 2016 35-56*
- Ding, Tao**, see Xu, Weifeng; *IJCA v23 n3 Sept 2016 141-159*
- Duri, Maththaiya**, Behnam S. Arad; A Pipelined Implementation of Hash Stream1-Synthetic Initialization Vector Encryption Algorithm; *IJCA v23 n2 June 2016 87-95*
- El Ariss, Omar**, see Xu, Weifeng; *IJCA v23 n3 Sept 2016 141-159*
- El-Kadi, Amr**, see Sobh, Karim; *IJCA v23 n2 June 2016 124-139*
- Etschmaier, Maximilian M.** and Gordon Lee; Defining the Paradigm of a Highly Automated System that Protects Against Human Failures and Terrorist Acts and Application to Aircraft Systems; *IJCA v23 n1 March 2016 4-11*
- Fekete, Andras** and Elizabeth Varki; Evaluating an Array of Heterogeneous Disks; *IJCA v23 n4 Dec 2016 208-215*
- Feng, Wenying** see Hu, Gongzhu; *IJCA v23 n1 March 2016 2-3*
- Flageol, William**, Mourad Badri, and Linda Badri; Investigating the Relationships between Use Cases Attributes and Source Code Size; *IJCA v23 n1 March 2016 69-76*
- Goto, Takaaki**, See Hu, Gongzhu; *IJCA v23 n1 March 2016 2-3*

see Hu, Gongzhu; *IJCA v23 n4 Dec 2016 207*

- Gray, Jeff**, see Yue, Songqing; *IJCA v23 n1 March 2016 57-68*

## H-J

- Hanaki, Hidenobu**, see Yokoyama, Michio; *IJCA v23 n1 March 2016 12-18*
- Harris, Jr., Frederick C.**; Editor's Note: March 2016; *IJCA v23 n1 March 2016 1*
- see Hu, Gongzhu; *IJCA v23 n1 March 2016 2-3*
- see Carthen Chase; *IJCA v23 n3 Sept 2016 195-207*
- ...see Wu, Rui; *IJCA v23 n4 Dec 2016 232-241*
- Hasegawa, Kazuki** and Kiyofumi Tanaka; Server Mechanisms for Guaranteeing Schedulability with RTOS Processing and Improving Application Responsiveness by Slack Reclaiming; *IJCA v23 n2 June 2016 116-123*
- Hesson, Aaron**, see Carthen, Chase; *IJCA v23 n3 Sept 2016 195-207*
- Hu, Gongzhu** and Takaaki Goto, Frederick C. Harris, Jr., Yan Shi, and Wenying Feng; Guest Editorial: Special Issue from ISCA Fall-2015 Conference; *IJCA v23 n1 March 2016 2-3*
- and Takaaki Goto; Guest Editorial: Special Issue from ISCA Fall-2016 Conference; *IJCA v23 n4 Dec 2016 207*
- Izurieta, Clemente**, see Assefi, Mehdi; *IJCA v23 n1 March 2016 19-28*
- Johansen, W. Joel**, see Carthen, Chase; *IJCA v23 n3 Sept 2016 195-207*
- Johnson, Christine M.**, see Carthen, Chase; *IJCA v23 n3 Sept 2016 195-207*

## K-L

- Kaneko, Keiichi**, see Bossard, Antoine; *IJCA v23 n4 Dec 2016 223-231*
- Katsumata, Kaori**, see Kinoshita, Toshiyuki; *IJCA v23 n2 June 2016 78-86*
- Kinoshita, Toshiyuki**, Matrazali

- Noorafiza, and Kaori Katsumata; *IJCA v23 n2 June 2016 78-86*
- Lee, Gordon** see Etschmaier, Maximilian M.; *IJCA v23 n1 March 2016 4-11*
- Lew, Roger**, see Carthen, Chase; *IJCA v23 n3 Sept 2016 195-207*
- Li, Wei**; Evaluation and Generalization of Trust Models in P2P Networks; *IJCA v23 n4 Dec 2016 216-222*
- Liu, Guangchi**, see Assefi, Mehdi; *IJCA v23 n1 March 2016 19-28*

### M-O

- Miller, Les**, Bossard, Antoine; *IJCA v23 n2 June 2016 77*
- Mizunuma, Mitsuru**, see Yokoyama, Michio; *IJCA v23 n1 March 2016 12-18*
- Muhanna, Muhanna**, see Wu, Rui; *IJCA v23 n4 Dec 2016 232-241*
- Negishi, Takumi**, see Yokoyama, Michio; *IJCA v23 n1 March 2016 12-18*
- Nielson, Daniel**, see Carthen, Chase; *IJCA v23 n3 Sept 2016 195-207*
- Nishimura, Kozo**, see Yokoyama, Michio; *IJCA v23 n1 March 2016 12-18*
- Noorafiza, Matrazali**, see Kinoshita, Toshiyuki; *IJCA v23 n2 June 2016 78-86*
- Oladunni, Timothy** and Sharad Sharma; Predicting Fair Housing Market Value: A Machine Learning Investigation; *IJCA v23 n3 Sept 2016 160-175*
- Otani, Kazuya**, see Yokoyama, Michio; *IJCA v23 n1 March 2016 12-18*

### P-Q

- Periyasamy, Kasi** and Karamveer Yadav; Consolidation of Data in Multiple Databases; *IJCA v23 n2 June 2016 96-104*

### R-S

- Rushton, Thomas J.**, see Carthen, Chase; *IJCA v23 n3 Sept 2016 195-207*
- Sharma, Sharad**, see Oladunni, Timothy; *IJCA v23 n3 Sept 2016 160-175*

- Shi, Yan**, see Hu, Gongzhu; *IJCA v23 n1 March 2016 2-3*
- Sobh, Karim** and Amr El-Kadi; A Unified Cloud Metering Framework; *IJCA v23 n2 June 2016 124-139*
- Stanchev, Lubomir**; Creating a Probabilistic Model for WordNet; *IJCA v23 n3 Sept 2016 176-194*
- Sultana, Kazi Zakia**, see Deo, Ajay K.; *IJCA v23 n1 March 2016 35-56*

### T-V

- Tanaka, Kiyofumi**, see Hasegawa, Kazuki; *IJCA v23 n2 June 2016 116-123*
- Varki, Elizabeth**, see Felete. Andras; *IJCA v23 n4 Dec 2016 208-215*

### W

- Wang, Yujun**, see Banitaan, Shadi; *IJCA v23 n1 March 2016 29-34*
- Williams, Byron J.**, see Deo, Ajay K.; *IJCA v23 n1 March 2016 35-56*
- Wittie, Mike P.**, see Assefi, Mehdi; *IJCA v23 n1 March 2016 19-28*
- Wood, Nicholas R.**, see Carthen, Chase; *IJCA v23 n3 Sept 2016 195-207*
- Worrell, Bryan**, see Carthen, Chase; *IJCA v23 n3 Sept 2016 195-207*
- Wu, Rui**, Muhanna Muhanna, Sergiu M. Dascalu, Lee Barford, Frederick C. Harris, Jr.; Data Lossless Compression Using Improved GFC Algorithm with Multiple GPUs; *IJCA v23 n4 Dec 2016 232-241*

### X-Z

- Xu, Dianxiang**, see Xu, Weifeng; *IJCA v23 n3 Sept 2016 141-159*
- Xu, Weifeng**, Tao Ding, Dianxiang Xu, and Omar El Ariss; Mining Decision Trees as Test Oracles for Java Bytecode; *IJCA v23 n3 Sept 2016 141-159*
- Yadav, Karamveer**, see Periyasamy, Kasi; *IJCA v23 n2 June 2016 96-104*
- Yokoyama, Michio**, Takumi Negishi, Mitsuru Mizunuma, Kazuya Otani, Hidenobu Hanaki, and Kozo Nishimura; Multiple Regression Analysis and Learning System for Estimation of Blood Pressure Variation Using Photo-Plethysmograph Signals; *IJCA v23 n1 March 2016 12-18*
- Yue, Songqing** and Jeff Gray; Transforming C Applications with Meta-Programming; *IJCA v23 n1 March 2016 57-68*
- Ziegler, Mathew**, see Carthen, Chase; *IJCA v23 n3 Sept 2016 195-207*

**Key Words****A****Abstraction***IJCA v23 n1 March 2016 57-68***Aircraft safety and security***IJCA v23 n1 March 2016 4-11***Attack***IJCA v23 n4 Dec 2016 216-222***Autonomous cloud metering objects***IJCA v23 n2 June 2016 124-139***B-C****Biological sound processing***IJCA v8 n1 March 2001 7-12***Blood pressure***IJCA v23 n1 March 2016 12-18***Blood pressure estimation***IJCA v23 n1 March 2016 12-18***Causality***IJCA v8 n1 March 2001 23-32***Central server model***IJCA v23 n2 June 2016 78-86***Cloud computing***IJCA v23 n2 June 2016 124-139***Cloud metering***IJCA v23 n2 June 2016 124-139***Cloud metering markup language***IJCA v23 n2 June 2016 124-139***Cloud speech recognition***IJCA v23 n1 March 2016 19-28***Computational reflection***IJCA v23 n1 March 2016 57-68***Computer security***IJCA v23 n2 June 2016 105-115***Computer system performance***IJCA v23 n2 June 2016 78-86***Constraint matching***IJCA v23 n2 June 2016 96-104***Content matching***IJCA v23 n2 June 2016 96-104***Correlation***IJCA v23 n1 March 2016 12-18***D****Decision tree***IJCA v23 n3 Sept 2016 141-159***Defect detection***IJCA v23 n1 March 2016 35-56***Deferrable sever***IJCA v23 n2 June 2016 116-123***Domain-specific language***IJCA v23 n1 March 2016 57-68***E-F****Encryption***IJCA v23 n2 June 2016 87-95***Finite input source***IJCA v23 n2 June 2016 78-86***Floating-point data***IJCA v23 n4 Dec 2016 232-241***G-H****Geospatial***IJCA v23 n3 Sept 2016 195-207***GFC***IJCA v23 n4 Dec 2016 232-241***Hardware accelerator***IJCA v23 n2 June 2016 87-95***Heterogeneous disks***IJCA v23 n4 Dec 2016 208-215***High-speed***IJCA v23 n4 Dec 2016 232-241***Housing prices prediction***IJCA v23 n3 Sept 2016 160-175***Human factors***IJCA v23 n1 March 2016 4-11***Human failures***IJCA v23 n1 March 2016 4-11***Human-machine symbiosis***IJCA v23 n1 March 2016 4-11***Hydrology***IJCA v23 n3 Sept 2016 195-207***I-J****Infrared LED sensor***IJCA v23 n1 March 2016 12-18***Jimple***IJCA v23 n3 Sept 2016 141-159***K-L****Kernel level transport layer***IJCA v23 n2 June 2016 124-139***K-NN***IJCA v23 n3 Sept 2016 160-175***Linear regression***IJCA v23 n3 Sept 2016 160-175***Language***IJCA v23 n4 Dec 2016 223-231***Learning system***IJCA v23 n1 March 2016 12-18***Linguistics***IJCA v23 n4 Dec 2016 223-231***Lossless compression***IJCA v23 n4 Dec 2016 232-241***M****Machine learning***IJCA v23 n3 Sept 2016 160-175***Markov logic network for representing WordNet data***IJCA v23 n3 Sept 2016 176-194***Meta-object protocol***IJCA v23 n1 March 2016 57-68***Metering framework***IJCA v23 n2 June 2016 124-139***Metrics***IJCA v23 n1 March 2016 69-76***Mining***IJCA v23 n3 Sept 2016 141-159***Model***IJCA v23 n4 Dec 2016 223-231***Model data***IJCA v23 n3 Sept 2016 195-207***Multicollinearity***IJCA v23 n1 March 2016 12-18***Multiple regression analysis***IJCA v23 n1 March 2016 12-18***MVC***IJCA v23 n3 Sept 2016 160-175***N-O****Nano-patterns***IJCA v23 n1 March 2016 35-56***Natural***IJCA v23 n4 Dec 2016 223-231***Netfilter hooks***IJCA v23 n2 June 2016 124-139***Network security***IJCA v23 n4 Dec 2016 216-222***Neural network***IJCA v23 n3 Sept 2016 160-175***P-Q****Peer to peer (P2P) network***IJCA v23 n4 Dec 2016 216-222***Performance evaluation***IJCA v23 n2 June 2016 78-86***Photo-plethysmography***IJCA v23 n1 March 2016 12-18***Pipelining***IJCA v23 n2 June 2016 87-95***Polling Server***IJCA v23 n2 June 2016 116-123***Polynomial regression***IJCA v23 n3 Sept 2016 160-175***Probability-based semantic similarity and distances***IJCA v23 n3 Sept 2016 176-194*



**Proc filesystem***IJCA v23 n2 June 2016 124-139***Program transformation***IJCA v23 n1 March 2016 57-68***Purposeful systems***IJCA v23 n1 March 2016 4-11**IJCA v23 n2 June 2016 105-115***Quality of experience***IJCA v23 n1 March 2016 19-28***Queuing network***IJCA v23 n2 June 2016 78-86***Queuing theory***IJCA v23 n2 June 2016 78-86***R****RAID***IJCA v23 n4 Dec 2016 208-215***Real Estate***IJCA v23 n3 Sept 2016 160-175***Real-time scheduling***IJCA v23 n2 June 2016 116-123***Real-time systems***IJCA v23 n1 March 2016 19-28***Relation***IJCA v23 n4 Dec 2016 223-231***Relationships***IJCA v23 n1 March 2016 69-76***Reputation system***IJCA v23 n4 Dec 2016 216-222***RTOS***IJCA v23 n2 June 2016 116-123***S****Schema matching***IJCA v23 n2 June 2016 96-104***Script***IJCA v23 n4 Dec 2016 223-231***Semantic similarity benchmarks for****WordNet***IJCA v23 n3 Sept 2016 176-194***Semantic similarity***IJCA v23 n3 Sept 2016 176-194***Simulation***IJCA v23 n2 June 2016 87-95***Slack***IJCA v23 n2 June 2016 116-123***Software development effort***IJCA v23 n1 March 2016 69-76***Software measurement***IJCA v23 n1 March 2016 19-28***Software metrics***IJCA v23 n1 March 2016 29-34***Software patterns***IJCA v23 n1 March 2016 35-56***Software quality***IJCA v23 n1 March 2016 35-56***Software testing***IJCA v23 n1 March 2016 29-34**IJCA v23 n3 Sept 2016 141-159***Source code size***IJCA v23 n1 March 2016 69-76***Storage***IJCA v23 n4 Dec 2016 208-215***Streaming Media***IJCA v23 n1 March 2016 19-28***Synthesis***IJCA v23 n2 June 2016 87-95***System security***IJCA v23 n1 March 2016 4-11***System Verilog***IJCA v23 n2 June 2016 87-95***Systolic/diastolic blood pressure***IJCA v23 n1 March 2016 12-18***T****Terrain***IJCA v23 n3 Sept 2016 195-207***Test case selection***IJCA v23 n1 March 2016 29-34***Test Oracle***IJCA v23 n3 Sept 2016 141-159***Traceable patterns***IJCA v23 n1 March 2016 35-56***Trust***IJCA v23 n4 Dec 2016 216-222***U-Z****UML***IJCA v23 n3 Sept 2016 160-175***Unit testing***IJCA v23 n1 March 2016 29-34***Use cases***IJCA v23 n1 March 2016 69-76***Use case points***IJCA v23 n1 March 2016 69-76***Virtual watershed client***IJCA v23 n3 Sept 2016 195-207***Virtual watershed platform***IJCA v23 n3 Sept 2016 195-207***Visualization***IJCA v23 n3 Sept 2016 195-207*

## Instructions for Authors

---

The International Journal of Computers and Their Applications is published multiple times a year with the purpose of providing a forum for state-of-the-art developments and research in the theory and design of computers, as well as current innovative activities in the applications of computers. In contrast to other journals, this journal focuses on emerging computer technologies with emphasis on the applicability to real world problems. Current areas of particular interest include, but are not limited to: architecture, networks, intelligent systems, parallel and distributed computing, software and information engineering, and computer applications (e.g., engineering, medicine, business, education, etc.). All papers are subject to peer review before selection.

---

### A. Procedure for Submission of a Technical Paper for Consideration

1. Email your manuscript to the Editor-in-Chief, Dr. Fred Harris, Jr., Fred.Harris@cse.unr.edu.
2. Illustrations should be high quality (originals unnecessary).
3. Enclose a separate page (or include in the email message) the preferred author and address for correspondence. Also, please include email, telephone, and fax information should further contact be needed.

### B. Manuscript Style:

1. The text should be **double-spaced** (12 point or larger), **single column** and **single-sided** on 8.5 X 11 inch pages.
2. An informative abstract of 100-250 words should be provided.
3. At least 5 keywords following the abstract describing the paper topics.
4. References (alphabetized by first author) should appear at the end of the paper, as follows: author(s), first initials followed by last name, title in quotation marks, periodical, volume, inclusive page numbers, month and year.
5. Figures should be captioned and referenced.

### C. Submission of Accepted Manuscripts

1. The final complete paper (with abstract, figures, tables, and keywords) satisfying Section B above in **MS Word format** should be submitted to the Editor-in-Chief.
2. The submission may be on a CD/DVD or as an email attachment(s) . **The following electronic files should be included:**
  - Paper text (required).
  - Bios (required for each author). Integrate at the end of the paper.
  - Author Photos (jpeg files are required by the printer, these also can be integrated into your paper).
  - Figures, Tables, Illustrations. These may be integrated into the paper text file or provided separately (jpeg, MS Word, PowerPoint, eps).
3. Specify on the CD/DVD label or in the email the word processor and version used, along with the title of the paper.
4. Authors are asked to sign an ISCA copyright form (<http://www.isca-hq.org/j-copyright.htm>), indicating that they are transferring the copyright to ISCA or declaring the work to be government-sponsored work in the public domain. Also, letters of permission for inclusion of non-original materials are required.

### Publication Charges

After a manuscript has been accepted for publication, the contact author will be invoiced for publication charges of **\$50.00 USD** per page (in the final IJCA two-column format) to cover part of the cost of publication. For ISCA members, \$100 of publication charges will be waived if requested.

