Application –Aware Big Data Duplication In Cloud Environment

Baie Sreelatha , A. Lizi M.C.A
RCR Institutes of Management and Technology

***Abstract*- Deduplication has become a widely deployed technology in cloud data centers to improve IT resources efficiency. However, traditional techniques face a great challenge in big data deduplication to strike a sensible tradeoff between the conflicting goals of scalable deduplication throughput and high duplicate elimination ratio. We propose *AppDedupe*, an application-aware scalable inline distributed deduplication framework in cloud environment, to meet this challenge by exploiting application awareness, data similarity and locality to optimize distributed deduplication with inter-node two-tiered data routing and intra-node application-aware deduplication. It first dispenses application data at file level with an application-aware routing to keep application locality, then assigns similar application data to the same storage node at the super-chunk granularity using a hand printing-based stateful data routing scheme to maintain high global deduplication efficiency, meanwhile balances the workload across nodes. AppDedupe builds application-aware similarity indices with super-chunk handprints to speed up the intra-node deduplication process with high efficiency. Our experimental evaluation of AppDedupe against state-of-the-art, driven by real-world datasets, demonstrates that AppDedupe achieves the highest global deduplication efficiency with a higher global deduplication effectiveness than the high-overhead and poorly scalable traditional scheme, but at an overhead only slightly higher than that of the scalable but low duplicate-elimination-ratio approaches.**

I. INTRODUCTION

Recent technological advancements in cloud computing, internet of things and social network, have led to a deluge of data from distinctive domains over the past two decades. Cloud data centers are awash in digital data, easily amassing petabytes and even exabytes of information, and the complexity of data management escalates in big data. However, IDC data shows that nearly 75% of our digital world is a copy. Data deduplication, a specialized data reduction technique widely deployed in disk-based storage systems, not only saves data storage space, power and cooling in data centers, also decreases significant administration time, operational complexity and risk of human error. It partitions large data objects into smaller parts, called chunks, represents these chunks by their fingerprints, replaces the duplicate chunks with
transfers or stores the unique chunks for the purpose of improving communication and storage efficiency. Data deduplication has been successfully used in various application scenarios, such as
backup system, virtual machine storage, primary storage, and WAN replication.

Big data deduplication is a highly scalable distributed deduplication technique to manage the data deluge under the changes in storage architecture to meet the service level agreement requirements of cloud storage. It is generally in favour of source inline deduplication design, be-cause it can immediately identify and eliminate duplicates in datasets at the source of data generation, and hence significantly reduce physical storage capacity requirements and save network bandwidth during data transfer. It performs in a typical distributed deduplication framework to satisfy scalable capacity and performance requirements in massive data. The framework includes inter-node data assignment from clients to multiple deduplication storage nodes by a data routing scheme, and independent intra-node redundancy suppression in individual storage nodes.

Unfortunately, this chunk-based inline distributed de-duplication framework at large scales faces challenges in both inter-node and intra-node scenarios. First, for the inter-node scenario, different from those distributed de-duplication with high overhead in global match query, there is a challenge called deduplication node in-formation island. It means that deduplication is only per-formed within individual nodes due to the communication overhead

considerations, and leaves the cross-node redundancy untouched. Second, for the intra-node scenario, it suffers from the chunk index lookup disk bottleneck. There is a chunk index of a large dataset, which maps each chunk's fingerprint to where that chunk is stored on disk in order to identify the replicated data. It is generally too big to fit into the limited memory of a deduplication node, and causes the parallel deduplication performance of multiple data streams to degrade significantly due to the frequent and random disk index I/O s.

II. EXISTING SYSTEM

There are several existing solutions that aim to tackle the above two challenges of distributed deduplication by exploiting data similarity or locality. Locality means that the chunks of a data stream will appear in approximately the same order again with a high probability. Locality-only based approaches distribute data across deduplication servers at coarse granularity to achieve scalable deduplication throughput across the nodes by exploiting locality in data streams, but they suffer low duplicate elimination ratio due to high cross-node redundancy.

Disadvantages:

1. Finding the data deduplication among several data centers.
2. Data has to be distributed on multiple locations.

III. PROPOSED SYSTEM

We propose AppDedupe, an application-aware scalable inline distributed deduplication framework in cloud environment, to meet this challenge by exploiting application awareness, data similarity and locality to optimize distributed deduplication with inter-node two-tiered data routing and intra-node application-aware deduplication. It first dispenses application data at file level with an application-aware routing to keep application locality, then assigns similar application data to the same storage node at the super-chunk granularity using a hand printing-based stateful data routing scheme to maintain high global deduplication efficiency, meanwhile balances the workload across nodes.

Advantages:

1. Stores the similar type of data in a single location.
2. So that data duplication will be found out easily.

IV. SYSTEM REQUIREMENTS

H/W System Configuration:-

Processor - Pentium –III
RAM - 256 MB (min)
Hard Disk - 20 GB
Key Board - Standard Windows Keyboard
Mouse - Two or Three Button Mouse
Monitor - SVGA

S/W System Configuration:-

Operating System : Windows family
Application Server : Tomcat5.0/6.X

Front End : HTML, Jsp
Scripts : JavaScript.
Server side Script : Java Server Pages.
Database : MySQL 6.0
Database Connectivity : JDBC

Software Environment

Java Technology

Java technology is both a programming language and a platform.

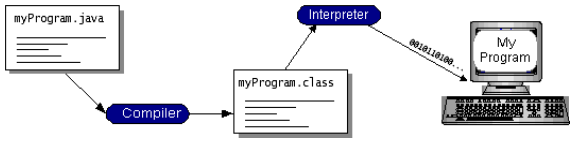
The Java Programming Language

The Java programming language is a high-level language that can be characterized by all of the following buzzwords:

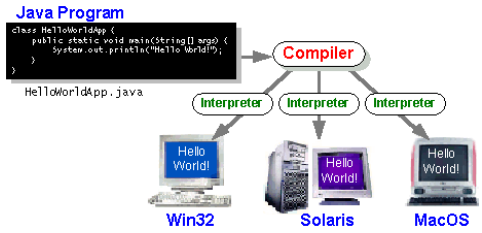
- Simple
- Architecture neutral
- Object oriented
- Portable
- Distributed
- High performance
- Interpreted
- Multithreaded
- Robust
- Dynamic
- Secure

With most programming languages, you either compile or interpret a program so that you can run it on your computer. The Java programming language is unusual in that a program is both compiled and interpreted. With the compiler, first you translate a program into an intermediate language called *Java byte codes* —the platform-independent codes

interpreted by the interpreter on the Java platform. The interpreter parses and runs each Java byte code instruction on the computer. Compilation happens just once; interpretation occurs each time the program is executed. The following figure illustrates how this works.



You can think of Java byte codes as the machine code instructions for the *Java Virtual Machine* (Java VM). Every Java interpreter, whether it's a development tool or a Web browser that can run applets, is an implementation of the Java VM. Java byte codes help make "write once, run anywhere" possible. You can compile your program into byte codes on any platform that has a Java compiler. The byte codes can then be run on any implementation of the Java VM. That means that as long as a computer has a Java VM, the same program written in the Java programming language can run on Windows 2000, a Solaris workstation, or on an iMac.



The Java Platform

A *platform* is the hardware or software environment in which a program runs. We've already mentioned some of the most popular platforms like Windows 2000, Linux, Solaris, and MacOS. Most platforms can be described as a combination of the operating system and hardware. The Java platform differs from most other platforms in that it's a software-only platform that runs on top of other hardware-based platforms.

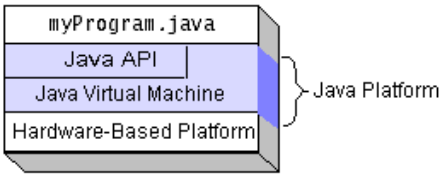
The Java platform has two components:

- The *Java Virtual Machine* (Java VM)
- The *Java Application Programming Interface* (Java API)

You've already been introduced to the Java VM. It's the base for the Java platform and is ported onto various hardware-based platforms.

The Java API is a large collection of ready-made software components that provide many useful capabilities, such as graphical user interface (GUI) widgets. The Java API is grouped into libraries of related classes and interfaces; these libraries are known as *packages*. The next section, What Can Java Technology Do? Highlights what functionality some of the packages in the Java API provide.

The following figure depicts a program that's running on the Java platform. As the figure shows, the Java API and the virtual machine insulate the program from the hardware.



Native code is code that after you compile it, the compiled code runs on a specific hardware platform. As a platform-independent environment, the Java platform can be a bit slower than native code. However, smart compilers, well-tuned interpreters, and just-in-time byte code compilers can bring performance close to that of native code without threatening portability.

What Can Java Technology Do?

The most common types of programs written in the Java programming language are *applets* and *applications*. If you've surfed the Web, you're probably already familiar with applets. An applet is a program that adheres to certain conventions that allow it to run within a Java-enabled browser.

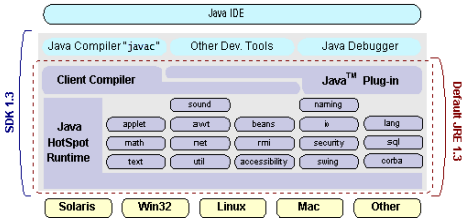
However, the Java programming language is not just for writing cute, entertaining applets for the Web. The general-purpose, high-level Java programming language is also a powerful software platform. Using the generous API, you can write many types of programs.

An application is a standalone program that runs directly on the Java platform. A special kind of application known as a *server* serves and supports clients on a network. Examples of servers are Web servers, proxy servers, mail servers, and print servers. Another specialized program is a *servlet*. A servlet can almost be thought of as an applet that runs on the server side. Java Servlets are a popular choice for building interactive web applications, replacing the use of CGI scripts. Servlets are similar to applets in that they are runtime extensions of applications. Instead of working in browsers, though, servlets run within Java Web servers, configuring or tailoring the server.

How does the API support all these kinds of programs? It does so with packages of software components that provides a wide range of functionality. Every full implementation of the Java platform gives you the following features:

- **The essentials**: Objects, strings, threads, numbers, input and output, data structures, system properties, date and time, and so on.
    - **Applets**: The set of conventions used by applets.
    - **Networking**: URLs, TCP (Transmission Control Protocol), UDP (User Data gram Protocol) sockets, and IP (Internet Protocol) addresses.
    - **Internationalization**: Help for writing programs that can be localized for users worldwide. Programs can automatically adapt to specific locales and be displayed in the appropriate language.
    - **Security**: Both low level and high level, including electronic signatures, public and private key management, access control, and certificates.
    - **Software components**: Known as JavaBeans™, can plug into existing component architectures.
    - **Object serialization**: Allows lightweight persistence and communication via Remote Method Invocation (RMI).
    - **Java Database Connectivity (JDBC™)**: Provides uniform access to a wide range of relational databases.

The Java platform also has APIs for 2D and 3D graphics, accessibility, servers, collaboration, telephony, speech, animation, and more. The following figure depicts what is included in the Java 2 SDK.



How Will Java Technology Change My Life?

We can't promise you fame, fortune, or even a job if you learn the Java programming language. Still, it is likely to make your programs better and requires less effort than other languages. We believe that Java technology will help you do the following:

- **Get started quickly**: Although the Java programming language is a powerful object-oriented language, it's easy to learn, especially for programmers already familiar with C or C++.
- **Write less code**: Comparisons of program metrics (class counts, method counts, and so on) suggest that a program written in the Java programming language can be four times smaller than the same program in C++.
- **Write better code**: The Java programming language encourages good coding practices, and its garbage collection helps you avoid memory leaks. Its object orientation, its JavaBeans component architecture, and its wide-ranging, easily extendible API let you reuse other people's tested code and introduce fewer bugs.

- **Develop programs more quickly**: Your development time may be as much as twice as fast versus writing the same program in C++. Why? You write fewer lines of code and it is a simpler programming language than C++.
- **Avoid platform dependencies with 100% Pure Java**: You can keep your program portable by avoiding the use of libraries written in other languages. The 100% Pure Java™ Product Certification Program has a repository of historical process manuals, white papers, brochures, and similar materials online.
- **Write once, run anywhere**: Because 100% Pure Java programs are compiled into machine-independent byte codes, they run consistently on any Java platform.
- **Distribute software more easily**: You can upgrade applets easily from a central server. Applets take advantage of the feature of allowing new classes to be loaded "on the fly," without recompiling the entire program.

V. ODBC

Microsoft Open Database Connectivity (ODBC) is a standard programming interface for application developers and database systems providers. Before ODBC became a *de facto* standard for Windows programs to interface with database systems, programmers had to use proprietary languages for each database they wanted to connect to. Now, ODBC has made the choice of the database system almost irrelevant from a coding perspective, which is as it should be. Application developers have much more important things to worry about than the syntax that is needed to port their program from one database to another when business needs suddenly change.

Through the ODBC Administrator in Control Panel, you can specify the particular database that is associated with a data source that an ODBC application program is written to use. Think of an ODBC data source as a door with a name on it. Each door will lead you to a particular database. For example, the data source named Sales Figures might be a SQL Server database, whereas the Accounts Payable data source could refer to an Access database. The physical database referred to by a data source can reside anywhere on the LAN.

The ODBC system files are not installed on your system by Windows 95. Rather, they are installed when you setup a separate database application, such as SQL Server Client or Visual Basic 4.0. When the ODBC icon is installed in Control Panel, it uses a file called ODBCINST.DLL. It is also possible to administer your ODBC data sources through a stand-alone program called ODBCADM.EXE. There is a 16-bit and a 32-bit version of this program and each maintains a separate list of ODBC data sources. From a programming perspective, the beauty of ODBC is that the application can be written to use the same set of function calls to interface with any data source, regardless of the database vendor. The source code of the application doesn't change whether it talks to Oracle or SQL Server. We only mention these two as an example. There are ODBC drivers available for several dozen popular database systems. Even Excel spreadsheets and plain text files can be turned into data sources. The operating system uses the Registry information written by ODBC Administrator to determine which low-level ODBC drivers are needed to talk to the data source (such as the interface to Oracle or SQL Server). The loading of the ODBC drivers is transparent to the ODBC application program. In a client/server environment, the ODBC API even handles many of the network issues for the application programmer.

The advantages of this scheme are so numerous that you are probably thinking there must be some catch. The only disadvantage of ODBC is that it isn't as efficient as talking directly to the native database interface. ODBC has had many detractors make the charge that it is too slow. Microsoft has always claimed that the critical factor in performance is the quality of the driver software that is used. In our humble opinion, this is true. The availability of good ODBC drivers has improved a great deal recently. And anyway, the criticism about performance is somewhat analogous to those who

said that compilers would never match the speed of pure assembly language. Maybe not, but the compiler (or ODBC) gives you the opportunity to write cleaner programs, which means you finish sooner. Meanwhile, computers get faster every year.

VI. JDBC

In an effort to set an independent database standard API for Java; Sun Microsystems developed Java Database Connectivity, or JDBC. JDBC offers a generic SQL database access mechanism that provides a consistent interface to a variety of RDBMSs. This consistent interface is achieved through the use of “plug-in” database connectivity modules, or *drivers*. If a database vendor wishes to have JDBC support, he or she must provide the driver for each platform that the database and Java run on.

To gain a wider acceptance of JDBC, Sun based JDBC’s framework on ODBC. As you discovered earlier in this chapter, ODBC has widespread support on a variety of platforms. Basing JDBC on ODBC will allow vendors to bring JDBC drivers to market much faster than developing a completely new connectivity solution.

JDBC was announced in March of 1996. It was released for a 90 day public review that ended June 8, 1996. Because of user input, the final JDBC v1.0 specification was released soon after.

The remainder of this section will cover enough information about JDBC for you to know what it is about and how to use it effectively. This is by no means a complete overview of JDBC. That would fill an entire book.

for **H**yper **T**ext **M**arkup **L**anguage, which is the most widely used language on Web to develop web pages. HTML was created by Berners-Lee in late 1991 but "HTML 2.0" was the first standard HTML specification which was published in 1995. HTML 4.01 was a major version of HTML and it was published in late 1999. Though HTML 4.01 version is widely used but currently we are having HTML-5 version which is an extension to HTML 4.01, and this version was published in 2012.

- Hypertext refers to the way in which Web pages (HTML documents) are linked together. Thus the link available on a webpage is called Hypertext.
- As its name suggests, HTML is a Markup Language which means you use HTML to simply "mark up" a text document with tags that tell a Web browser how to structure it to display.

Originally, HTML was developed with the intent of defining the structure of documents like headings, paragraphs, lists, and so forth to facilitate the sharing of scientific information between researchers.

These are some basic tags of html

VII. SYSTEM DESIGN UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The UML uses mostly graphical notations to express the design of software projects.

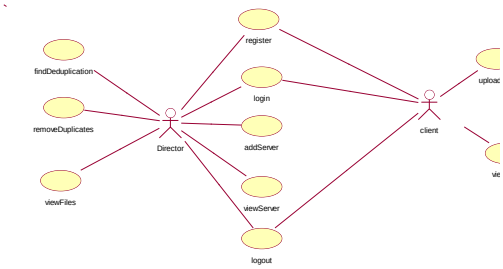
GOALS

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

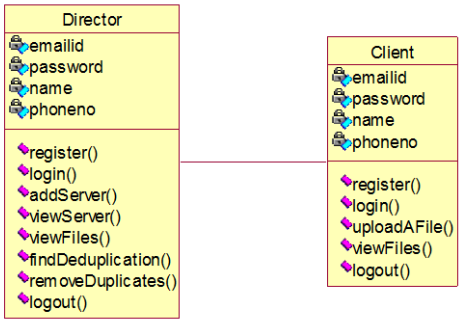
USE CASE DIAGRAM

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.



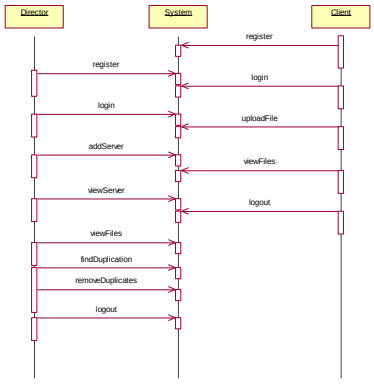
CLASS DIAGRAM

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.



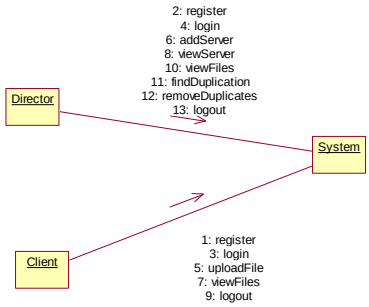
SEQUENCE DIAGRAM

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



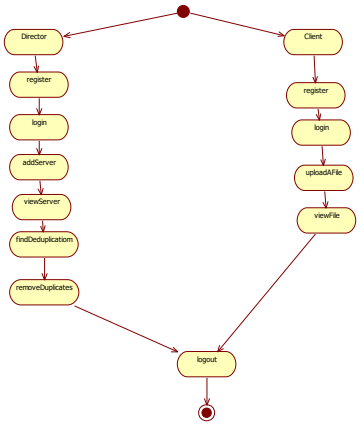
Collaboration Diagram:

In collaboration diagram the method call sequence is indicated by some numbering technique as shown below. The number indicates how the methods are called one after another. We have taken the same order management system to describe the collaboration diagram. The method calls are similar to that of a sequence diagram. But the difference is that the sequence diagram does not describe the object organization where as the collaboration diagram shows the object organization.



ACTIVITY DIAGRAM:

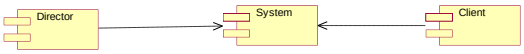
Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.



COMPONENTDIAGRAM

Component diagrams are used to describe the physical artifacts of a system. This artifact includes files, executables, libraries etc. So the purpose of this diagram is different, Component diagrams are used during the implementation phase of an application. But it is prepared well in advance to visualize the implementation details. Initially the system is designed using different UML diagrams and then

when the artifacts are ready component diagrams are used to get an idea of the implementation.



DEPLOYMENT DIAGRAM

Deployment diagram represents the deployment view of a system. It is related to the component diagram. Because the components are deployed using the deployment diagrams. A deployment diagram consists of nodes. Nodes are nothing but physical hardwares used to deploy the application.

VIII. CONCLUSION

In this paper, we describe AppDedupe, an application-aware scalable inline distributed deduplication frame-work for big data management, which achieves a tradeoff between scalable performance and distributed deduplication effectiveness by exploiting application awareness, data similarity and locality. It adopts a two-tiered data routing scheme to route data at the super-chunk granularity to reduce cross-node data redundancy with good load balance and low communication overhead, and employs application-aware similarity index based optimization to improve deduplication efficiency in each node with very low RAM usage. Our real-world trace-driven evaluation clearly demonstrates AppDedupe's significant advantages over the state-of-the-art distributed deduplication schemes for large clusters in the following important two ways. First, it outperforms the extremely costly and poorly scalable stateful tight coupling scheme in the cluster-wide deduplication ratio but only at a slightly higher system overhead than the highly scalable loose coupling schemes. Second, it significantly improves the stateless loose coupling schemes in the cluster-wide effective de-duplication ratio while retaining the latter's high system scalability with low overhead.

REFERENCES

[1] J. Gantz, D. Reinsel, "The Digital Universe Decade-Are You Ready?" White Paper, IDC, May 2010.

[2] H. Biggar, "Experiencing Data De-Duplication: Improving Efficiency and Reducing Capacity Requirements," White Paper, the Enterprise Strategy Group, Feb. 2007.

[3] K.R. Jayaram, C. Peng, Z. Zhang, M. Kim, H. Chen, H. Lei. "An Empirical Analysis of Similarity in Virtual Machine Images,"

ing: large scale, inline deduplication using sampling and locality," Proc. of the 7th Conf. on File and Storage Technologies (FAST '09), pages 111-123, Feb. 2009.

[15] B. Debnath, S. Sengupta, and J. Li. "ChunkStash: speeding up inline storage deduplication using flash memory." In Proceed-ings of the 2010 USENIX conference on USENIX annual tech-nical conference (2010), USENIX Association, p. 16.

[16] W. Xia, H. Jiang, D. Feng, Y. Hua, "Silo: a Similarity-locality based Near-exact Deduplication Scheme with Low RAM Overhead and High Throughput," Proc. of 2011 USENIX An-nual Technical Conference (ATC'11), pp. 285-298, Jun. 2011.

[17] A. Katiyar, J. Weissman. "ViDeDup: An Application-Aware Framework for Video Deduplication," Proc. of the 3rd USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage'11), pp. 31-35, 2011.

[18] C. Liu, Y. Lu, C. Shi, G. Lu, D. Du, and D.-S Wang, "ADMAD: Application-driven metadata aware de-deduplication archival storage systems," Proc. of the 5th IEEE International Work-shop on Storage Network Architecture and Parallel I/Os (SNAPI'08), pp.29-35, 2008.

[19] Y. Fu, H. Jiang, N. Xiao, L. Tian, F. Liu, L. Xu. "Application-Aware Local-Global Source Deduplication based Cloud Back-up Services for Personal Storage." In Proc. of IEEE Transac-tions on Parallel and Distributed Systems, 25(5): 1155-1165, 2014.

[20] L. Aronovich, R. Asher, E. Bachmat, H. Bitner, M. Hirsch, and S. T. Klein, "The design of a similarity based deduplication system," Proc. of 2nd ACM Annual Int. Systems and Storage Conf. (SYSTOR '09), pp. 6-6, Jun. 2009.

[21] D. Bhagwat, K. Eshghi, P. Mehra, "Content-based Document Routing and Index Partitioning for Scalable Similarity-based Searches in a Large Corpus," Proc. of the 13th ACM Interna-tional Conf. on Knowledge Discovery and Data Mining (SIGKDD'07), pp. 105-112, Aug. 2007.

[22] Y. Fu, H. Jiang, N. Xiao, L. Tian, F. Liu, "AA-Dedupe: An Ap-plication-Aware Source Deduplication Approach for Cloud Backup Services in the Personal Computing Environment," Proc. of the 13th IEEE Conf. on Cluster Computing (Cluster'11), pp. 112-120, Sep. 2011.

[23] Jaccard Index. http://en.wikipedia.org/wiki/Jaccard_index. 2012.

[24] A.Z. Broder, M. Charikar, A.M. Frieze, M. Mitzenmacher, "Min-wise Independent Permutations," Journal of Computer and System Sciences, vol. 60, no. 3, pp. 630–659, Jun. 2000, doi:10.1006/jcss.1999.1690.

[25] K. Eshghi, H.K. Tang, "A framework for Analyzing and Im-proving Content-based Chunking Algorithms," Technical Re-port, HPL-2005-30R1, HP Lab., Palo Alto, Sep. 2005.