International Journal of Research
Available at https://edupediapublications.org/journals
e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 04 Issue 14
November 2017

Innovative And Intelligent Data Retrieval Model Using Semantic Oriented Frameworks

C. Parvathi Poojitha & C. Viswanadh
Sir Vishveshwariah Institute Of Science And Technology

(Asst.Professor) Sir Vishveshwariah Institute Of Science And Technology

viswanadh.chinna@gmail.com poojitha291@gmail.com

ABSTRACT

The challenges of handling the explosive growth in data volume and complexity cause the increasing needs for semantic queries. The semantic queries can be interpreted as the correlation-aware retrieval, while containing approximate results. Existing cloud storage systems mainly fail to offer an adequate capability for the semantic queries. Since the true value or worth of data heavily depends on how efficiently semantic search can be carried out on the data in (near-) real-time, large fractions of data end up with their values being lost or significantly reduced due to the data staleness. To address this problem, we propose a near-real-time and cost-effective semantic queries based methodology, called FAST. The idea behind FAST is to explore and exploit the semantic correlation within and among datasets via correlation-aware hashing and manageable flat-structured addressing to significantly reduce the processing latency, while incurring acceptably small loss of data-search accuracy. The near-real-time property of FAST enables rapid identification of correlated files and the significant narrowing of the scope of data to be processed. FAST supports several types of data analytics, which can be implemented in existing searchable storage systems. We conduct a real-world use case in which children reported missing in an extremely crowded environment (e.g., a highly popular scenic spot on a peak tourist day) are identified in a timely fashion by analyzing 60 million images using FAST. FAST is further improved by using semantic-aware namespace to provide dynamic and adaptive namespace management for ultra-large storage systems. Extensive experimental results demonstrate the efficiency and efficacy of FAST in the performance improvements.

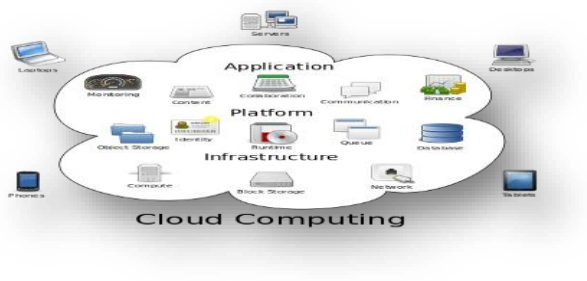
INTRODUCTION

What is cloud computing?

Cloud computing is the use of computing resources (hardware and software) that are delivered as a service over a network (typically the Internet). The name comes from the common use of a cloud-shaped symbol as an abstraction for the complex

infrastructure it contains in system diagrams. Cloud computing entrusts remote services with a user's data, software and computation. Cloud computing consists of hardware and software resources made available on the Internet as managed third-party services.

These services typically provide access to advanced software applications and high-end networks of server computers.



How Cloud Computing Works?

The goal of cloud computing is to apply traditional supercomputing, or high-performance computing power, normally used by military and research facilities, to perform tens of trillions of computations per second, in consumer-oriented applications such as financial portfolios, to deliver personalized information, to provide data storage or to power large, immersive computer games.

The cloud computing uses networks of large groups of servers typically running low-cost consumer PC technology with specialized connections to spread data-processing chores across them. This shared IT infrastructure contains large pools of systems that are linked together. Often, virtualization techniques are used to maximize the power of cloud computing.

International Journal of Research
Available at https://edupediapublications.org/journals
e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 04 Issue 14
November 2017

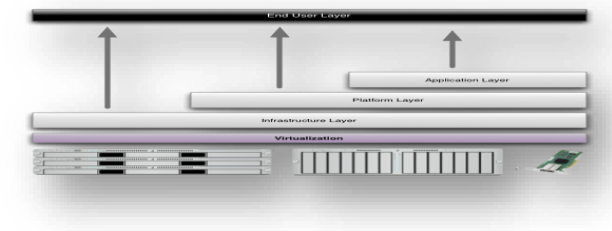
Characteristics and Services Models:

The salient characteristics of cloud computing based on the definitions provided by the National Institute of Standards and Terminology (NIST) are outlined below:

- **On-demand self-service**: A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service's provider.
- **Broad network access**: Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, laptops, and PDAs).
- **Resource pooling**: The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. There is a sense of location-independence in that the customer generally has no control or knowledge over the exact location of the provided resources but may be able to specify location at a higher level of abstraction (e.g., country, state, or data center). Examples of resources include storage, processing, memory, network bandwidth, and virtual machines.
- **Rapid elasticity**: Capabilities can be rapidly and elastically provisioned, in some cases automatically, to quickly scale out and rapidly released to quickly scale in. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be purchased in any quantity at any time.
- **Measured service**: Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts). Resource usage can be managed, controlled, and reported providing transparency for both the provider and consumer of the utilized service.

Services Models:

Cloud Computing comprises three different service models, namely Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS). The three service models or layer are completed by an end user layer that encapsulates the end user perspective on cloud services. The model is shown in figure below. If a cloud user accesses services on the infrastructure layer, for instance, she can run her own applications on the resources of a cloud infrastructure and remain responsible for the support, maintenance, and security of these applications herself. If she accesses a service on the application layer, these tasks are normally taken care of by the cloud service provider.



Benefits of cloud computing:

1. **Achieve economies of scale** – increase volume output or productivity with fewer people. Your cost per unit, project or product plummets.
2. **Reduce spending on technology infrastructure.** Maintain easy access to your information with minimal upfront spending. Pay as you go (weekly, quarterly or yearly), based on demand.
3. **Globalize your workforce on the cheap.** People worldwide can access the cloud, provided they have an Internet connection.

4. **Streamline processes.** Get more work done in less time with less people.
5. **Reduce capital costs.** There's no need to spend big money on hardware, software or licensing fees.
6. **Improve accessibility.** You have access anytime, anywhere, making your life so much easier!
7. **Monitor projects more effectively.** Stay within budget and ahead of completion cycle times.
8. **Less personnel training is needed.** It takes fewer people to do more work on a cloud, with a minimal learning curve on hardware and software issues.
9. **Minimize licensing new software.** Stretch and grow without the need to buy expensive software licenses or programs.
10. **Improve flexibility.** You can change direction without serious "people" or "financial" issues at stake.

Advantages:

1. **Price:** Pay for only the resources used.
2. **Security**: Cloud instances are isolated in the network from other instances for improved security.
3. **Performance:** Instances can be added instantly for improved performance. Clients have access to the total resources of the Cloud's core hardware.
4. **Scalability:** Auto-deploy cloud instances when needed.
5. **Uptime:** Uses multiple servers for maximum redundancies. In case of server failure, instances can be automatically created on another server.
6. **Control:** Able to login from any location. Server snapshot and a software library lets you deploy custom instances.
7. **Traffic:** Deals with spike in traffic with quick deployment of additional instances to handle the load.

The word *distributed* in terms such as "distributed system", "distributed programming", and "distributed algorithm" originally referred to computer networks where individual computers were physically distributed within some geographical area. The terms are nowadays used in a much wider sense, even referring to autonomous processes that run on the same physical computer and interact with each other by message passing. While there is no single definition of a distributed system, the following defining properties are commonly used:

- There are several autonomous computational entities, each of which has its own local memory.
- The entities communicate with each other by message passing.

In this article, the computational entities are called *computers* or *nodes*. A distributed system may have a common goal, such as solving a large computational problem.[1] Alternatively, each computer may have its own user with individual needs, and the purpose of the distributed system is to coordinate the use of shared resources or provide communication services to the users. Other typical properties of distributed systems include the following:

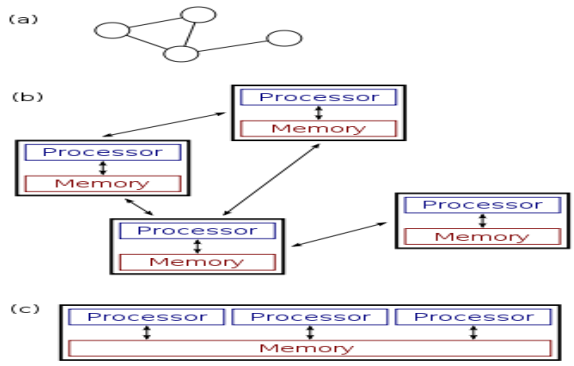
- The system has to tolerate failures in individual computers.
- The structure of the system (network topology, network latency, number of computers) is not known in advance, the system may consist of different kinds of computers and network links, and the system may change during the execution of a distributed program.
- Each computer has only a limited, incomplete view of the system. Each computer may know only one part of the input.

Distributed systems are groups of networked computers, which have the same goal for their work. The terms "concurrent computing", "parallel

International Journal of Research
Available at https://edupediapublications.org/journals
e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 04 Issue 14
November 2017

computing", and "distributed computing" have a lot of overlap, and no clear distinction exists between them. The same system may be characterized both as "parallel" and "distributed"; the processors in a typical distributed system run concurrently in parallel. Parallel computing may be seen as a particular tightly coupled form of distributed computing, and distributed computing may be seen as a loosely coupled form of parallel computing. Nevertheless, it is possible to roughly classify concurrent systems as "parallel" or "distributed" using the following criteria:

- In parallel computing, all processors may have access to a shared memory to exchange information between processors.
- In distributed computing, each processor has its own private memory (distributed memory). Information is exchanged by passing messages between the processors.



The figure on the right illustrates the difference between distributed and parallel systems. Figure (a) is a schematic view of a typical distributed system; as usual, the system is represented as a network topology in which each node is a computer and each line connecting the nodes is a communication link. Figure (b) shows the same distributed system in more detail: each computer has its own local memory, and information can be exchanged only by passing messages from one node to another by using the available communication links. Figure (c) shows a parallel system in which each processor has a direct access to a shared memory.

The situation is further complicated by the traditional uses of the terms parallel and distributed *algorithm* that do not quite match the above definitions of parallel and distributed *systems*; see the section Theoretical foundations below for more detailed discussion. Nevertheless, as a rule of thumb, high-performance parallel computation in a shared-memory multiprocessor uses parallel algorithms while the coordination of a large-scale distributed system uses distributed algorithms.

IMPLEMENTATION

System Construction Module

- In the first module we develop the System Construction module, to evaluate and implement a near-real-time and cost-effective semantic queries based methodology, called FAST. For this purpose we develop User and Admin entities. In User entity, a user can upload a new images, view all uploaded images and a user can search a images of other users images by using content based image retrieval.
- In the admin entity, the admin privileged access is provided and then admin monitor the user's details and users uploaded images.
- To implement FAST and examine the efficiency and efficacy of the proposed methodology, we leverage "Finding Missing Children" as a use case to elaborate the FAST design and evaluate its performance. A missing child is not only devastating to his/her family but also has negative societal consequences. Although existing surveillance systems are helpful, they often suffer from the extremely slow identification process and the heavy reliance on manual observations from overwhelming volumes of data.

Semantic-Aware Namespace

- By leveraging semantic aggregation, FAST is able to improve entire system scalability. The semantics embedded in file attributes and user access patterns can be used to reveal the potential correlation of file in a large and distributed storage system. These files are thus aggregated into the same or adjacent groups by using the semantic-aware per-file namespace.

- In order to offer smart namespace in FAST, we need to manage the file system namespace in an intelligent and automatic way. In FAST's namespace, we identify semantic correlations and data affinity via lightweight hashing schemes.
- In order to accurately represent the namespace, FAST makes use of multi-dimensional, rather than single-dimensional, attributes to identify semantic correlations. FAST hence obtains the accuracy and simplicity in namespace for large-scale file systems.

Features of Images

- To perform reliable and accurate matching between different views of an object or scene that characterize similar images, we extract distinctive invariant features from images. Feature-based management can be used to detect and represent similar images to support correlation-aware grouping and similarity search. Potential interest points are identified by scanning the image over location and scale.
- We propose to use a crowd-based aid, i.e., personal images that can be openly accessed, to identify helpful clues. People often take many similar pictures on a famous scenic spot, which actually are the snapshots of those locations in a given period of time. High-resolution cameras offer high image quality and multiple angles. Repeatedly taking pictures can further guarantee the quality of snapshots.
- Given the convenient and easy access to the cloud, these images are often uploaded and shared on the web instantaneously (e.g., by smartphones). We can therefore leverage these publicly accessible images made possible in part by the crowdsourcing activities to help find the images that are correlated with a given missing child.

Flat-Structured Addressing

- The near-real-time property of FAST enables rapid identification of correlated files and the significant narrowing of the scope of data to be processed. FAST supports several types of data analytics, which can be implemented in existing searchable storage system. FAST consists of two main functional modules, i.e., big data processing and semantic correlation analysis. FAST is able to improve entire system scalability. FAST is designed to be compatible with or orthogonal to existing file systems. We hence implement FAST as a middleware between user applications and file systems. For the file system stacks, FAST is transparent, thus being flexibly used in most file systems to significantly improve system performance.
- The namespace serves as a middleware in the file systems by offering an optional semantic-aware function. To be compliant with conventional hierarchical file systems, the user level client contains two interfaces, which can be decided by the application requirements. If working with the conventional systems, the proposed namespace bypasses the semantic middleware and directly links with the application, like existing file systems.
- Users can access file systems via the existing POSIX interfaces. Otherwise, the namespace is used via the enhanced POSIX I/O in the user space. By exploiting the semantic correlations existing in the files' metadata, FAST is able to support efficient semantic grouping and allow users to carry out the read/write operations on files via the enhanced POSIX I/O interfaces

LITERATURE SURVEY

A comparative study of high performance computing on the cloud

The popularity of Amazon's EC2 cloud platform has increased in recent years. However, many high-performance computing (HPC) users consider dedicated high-performance clusters, typically found in large compute centers such as those in national laboratories, to be far superior to EC2 because of significant communication overhead of the latter. Our view is that this is quite narrow and the proper metrics for comparing high-performance clusters to EC2 is turnaround time and cost.

In this paper, we compare the top-of-the-line EC2 cluster to HPC clusters at Lawrence Livermore National Laboratory (LLNL) based on turnaround time and total cost of execution. When measuring turnaround time, we include expected queue wait time on HPC clusters. Our results show that although as expected, standard HPC clusters are superior in raw performance, EC2 clusters may produce better turnaround times. To estimate cost, we developed a

pricing model---relative to EC2's node-hour prices---to set node-hour prices for (currently free) LLNL clusters. We observe that the cost-effectiveness of running an application on a cluster depends on raw performance and application scalability.

Evaluating the usefulness of content addressable storage for high-performance data intensive applications

Content Addressable Storage (CAS) is a data representation technique that operates by partitioning a given data-set into non-intersecting units called chunks and then employing techniques to efficiently recognize chunks occurring multiple times. This allows CAS to eliminate duplicate instances of such chunks, resulting in reduced storage space compared to conventional representations of data. CAS is an attractive technique for reducing the storage and network bandwidth needs of performance-sensitive, data-intensive applications in a variety of domains. These include enterprise applications, Web-based e-commerce or entertainment services and highly parallel scientific/engineering applications and simulations, to name a few.In this paper, we conduct an empirical evaluation of the benefits offered by CAS to a variety of real-world data-intensive applications. The savings offered by CAS depend crucially on (i) the nature of the data-set itself and (ii) the chunk-size that CAS employs. We investigate the impact of both these factors on disk space savings, savings in network bandwidth, and error resilience of data. We find that a chunk-size of 1 KB can provide up to 84% savings in disk space and even higher savings in network bandwidth whilst trading off error resilience and incurring 14% CAS related overheads. Drawing upon lessons learned from our study, we provide insights on (i) the choice of the chunk-size for effective space savings and (ii) the use of selective data replication to counter the loss of error resilience caused by CAS.

SmartEye: Real-time and efficient cloud image sharing for disaster environments

Rapid disaster relief is important to save human lives and reduce property loss. With the wide use of smartphones and their ubiquitous easy access to the Internet, sharing and uploading images to the cloud via smartphones offer a nontrivial opportunity to provide information of disaster zones. However, due to limited available bandwidth and energy, Smartphone-based crowdsourcing fails to support the real-time data analytics

SYSTEM ANALYSIS

EXISTING SYSTEM:

- ISABELAQA is a parallel query processing engine that is designed and optimized for analyzing and processing spatiotemporal, multivariate scientific data. Mix Apart uses an integrated data caching and scheduling solution to allow Map Reduce computations to analyze data stored on enterprise storage systems.
- The frontend caching layer enables the local storage performance required by data analytics. The shared storage back-end simplifies data management.
- Spyglass exploits the locality of file namespace and skewed distribution of metadata to map the namespace hierarchy into a multi-dimensional K-D tree and uses multilevel versioning and partitioning to maintain consistency.

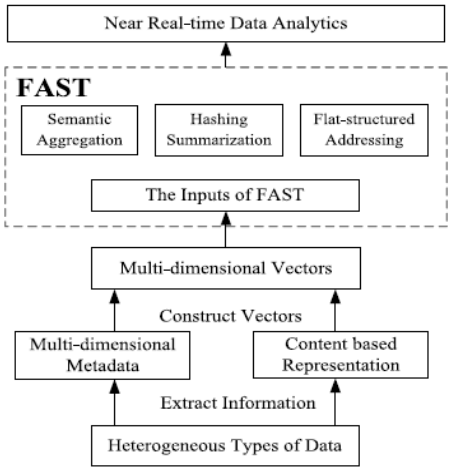
PROPOSED SYSTEM:

- In the context of this paper, searchable data analytics are interpreted as obtaining data value/worth via queried results, such as finding a valuable record, a correlated process ID, an important image, a rebuild system log, etc.
- We propose a novel near-real-time methodology for analyzing massive data, called FAST, with a design goal of efficiently processing such data in a real-time manner.
- The key idea behind FAST is to explore and exploit the correlation property within and among datasets via improved correlation aware hashing and flat-structured addressing to significantly reduce the processing latency of parallel queries, while incurring acceptably small loss of accuracy.

SYSTEM DESIGN

SYSTEM ARCHITECTURE:

International Journal of Research
Available at https://edupediapublications.org/journals
e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 04 Issue 14
November 2017



ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

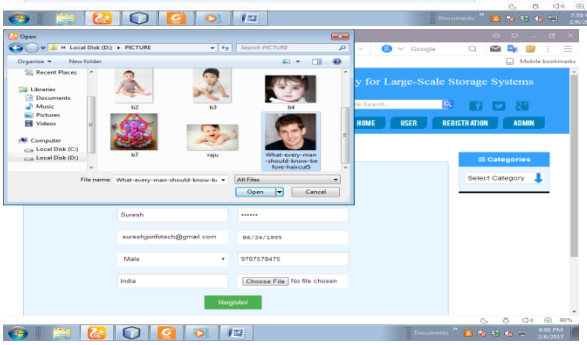
TECHNICAL FEASIBILITY

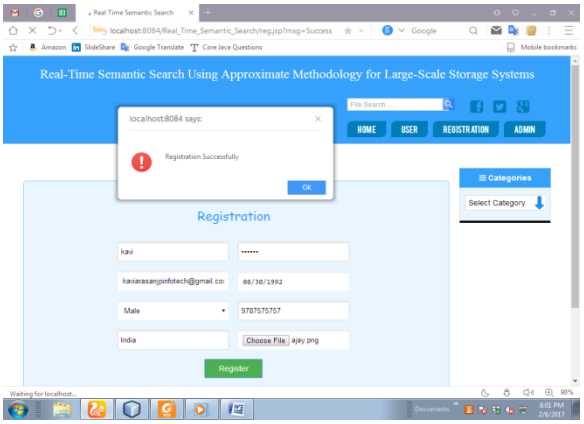
This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

SOCIAL FEASIBILITY

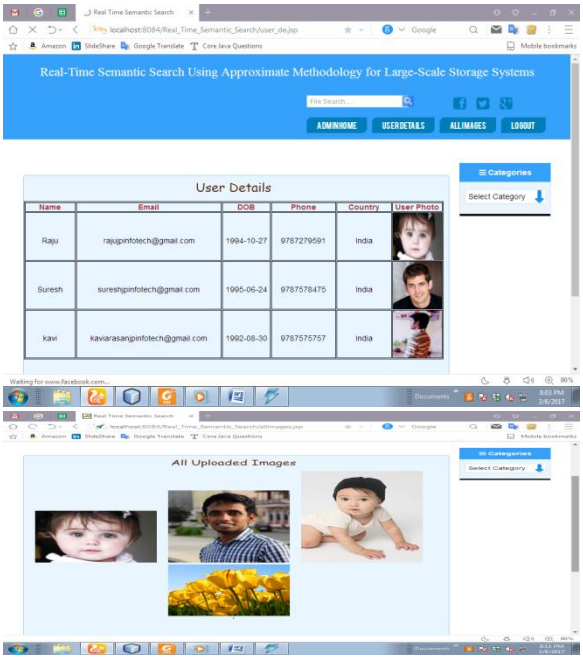
The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

Result





International Journal of Research
Available at https://edupediapublications.org/journals
e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 04 Issue 14
November 2017



CONCLUSION

This paper proposes a near real-time scheme, called FAST, to support efficient and cost-effective searchable data analytics in the cloud. FAST is designed to exploit the correlation property of data by using correlation-aware hashing and manageable flat-structured addressing. This enables FAST to significantly reduce processing latency of correlated file detection with acceptably small loss of accuracy. We discuss how the FAST methodology can be related to and used to enhance some storage systems, including Spyglass and Smart Store, as well as a use case. FAST is demonstrated to be a useful tool in supporting near real-time processing of real-world data analytics applications.

REFERENCES

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," Commun. ACM, vol. 53, no. 4, pp. 50–58, 2010.

[2] A. Marathe, R. Harris, D. K. Lowenthal, B. R. de Supinski, B. Rountree, M. Schulz, and X. Yuan, "A comparative study of highperformance computing on the cloud," in Proc. 22nd Int. Symp. High-Perform. Parallel Distrib. Comput., 2013, pp. 239–250.

[3] P. Nath, B. Urgaonkar, and A. Sivasubramaniam, "Evaluating the usefulness of content addressable storage for high-performance data intensive applications," in Proc. 17th Int. Symp. High-Perform. Parallel Distrib. Comput., 2008, pp. 35–44.

[4] Gartner, Inc., "Forecast: Consumer digital storage needs, 2010–2016," 2012.

[5] Storage Newsletter, "7% of consumer content in cloud storage in 2011, 36% in 2016," 2012.

[6] J. Gantz and D. Reinsel, "The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east," International Data Corporation (IDC) iView, Dec. 2012.

[7] Y. Hua, W. He, X. Liu, and D. Feng, "SmartEye: Real-time and efficient cloud image sharing for disaster environments," in Proc. INFOCOM, 2015, pp. 1616–1624.

[8] Y. Ke and R. Sukthankar, "PCA-SIFT: A more distinctive representation for local image descriptors," in Proc. IEEE Conf. Comput. Vis. Pattern Recog., 2004, pp. 506–513.

[9] Y. Ke, R. Sukthankar, and L. Huston, "Efficient near-duplicate detection and sub-image retrieval," in Proc. ACM Multimedia, 2004, pp. 869–876.

[10] J. Liu, Z. Huang, H. T. Shen, H. Cheng, and Y. Chen, "Presenting diverse location views with real-time near-duplicate photo elimination," in Proc. 29th Int. Conf. Data Eng., 2013, pp. 505–56.

[11] D. Zhan, H. Jiang, and S. C. Seth, "CLU: Co-optimizing locality and utility in thread-aware capacity management for shared last level caches," IEEE Trans. Comput., vol. 63, no. 7, pp. 1656–1667, Jul. 2014.

[12] P. Indyk and R. Motwani, "Approximate nearest neighbors: towards removing the curse of dimensionality," in Proc. 13th Annu. ACM Symp. Theory Comput., 1998, pp. 604–613.

[13] R. Pagh and F. Rodler, "Cuckoo hashing," in Proc. Eur. Symp. Algorithms, 2001, pp. 121–133.

[14] Y. Hua, H. Jiang, Y. Zhu, D. Feng, and L. Xu, "SANE: Semantic aware namespace in ultra-large-scale file systems," IEEE Trans. Parallel Distrib. Syst., vol. 25, no. 5, pp. 1328–1338, May 2014.

[15] (2011). Changewave research [Online]. Available: http://www.changewaveresearch.com

[16] Science Staff, "Dealing with data-Challenges and opportunities," Science, vol. 331, no. 6018, pp. 692–693, 2011.
[17] X. Tan, S. Chen, Z.-H. Zhou, and F. Zhang, "Face recognition from a single image per person: A survey," Pattern Recog., vol. 39, no. 9, pp. 1725–1745, 2006.
[18] T. Ahonen, A. Hadid, and M. Pietikainen, "Face description with local binary patterns: Application to face recognition," IEEE Trans. Pattern Anal. Mach. Intell., vol. 28, no. 12, pp. 2037–2041, Dec. 2006.
[19] X. Tan and B. Triggs, "Enhanced local texture feature sets for face recognition under difficult lighting conditions," IEEE Trans. Image Process., vol. 19, no. 6, pp. 1635–1650, Jun. 2010.
[20] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma, "Robust face recognition via sparse representation," IEEE Trans. Pattern Anal. Mach. Intell., vol. 31, no. 2, pp. 210–227, Feb. 2009.