Integrated IoT and Cloud Environment for Fingerprint Recognition

Ehsan Nadjaran Toosi[1], Adel Nadjaran Toosi[1], Reza Godaz[2], and Rajkumar Buyya[1]

[1] Cloud Computing and Distributed Systems (CLOUDS) Laboratory
School of Computing and Information Systems
The University of Melbourne, Australia
{enadjaran,anadjaran,rbuyya}@unimelb.edu.au
[2] Department of Software Enegineering
Islamic Azad University of Mashhad
rgodaz@mshdiau.ac.ir

Abstract. Big data applications involving the analysis of large datasets becomes a critical part of many emerging paradigms such as smart cities, social networks and modern security systems. Cloud computing has developed as a mainstream for hosting big data applications by its ability to provide the illusion of infinite resources. However, harnessing cloud resources for large-scale big data computation is application specific to a large extent. In this paper, we propose a system for large-scale fingerprint matching application using Aneka, a platform for developing scalable applications on the Cloud. We present the design and implementation of our proposed system and conduct experiments to evaluate its performance using resources from Microsoft Azure. Experimental results demonstrate that matching time for biometric information such as fingerprints in large-scale databases can be reduced substantially using our proposed system.

1 Introduction

Big data applications are getting popular in many fields due to the quick expansion of the Internet, smart cities, Internet of Things (IoT) devices in producing data [1]. In most of the cases, the data requires being processed and structured for further procedures [2]. Processing big data is often very time-consuming while it could be decreased by increasing the computation power. One of the most preferred approaches to speed up big data processing is cloud computing [3].

Cloud computing can provide an infinite amount of computing, storage, and network resources which suits big data challenges. The data could also be stored entirely in a local infrastructure and only transferred to public infrastructure for more computation power while the trade-off between data transfer and computation power need to be considered.

In this paper, we propose a design and implementation of a big data and security-based application for searching and finding a matched fingerprint as a

biometric information among a massive database of fingerprints records. The main aim of the application is to find personal information attached to the matched fingerprint. For instance, we suppose that a police department is responsible for finding the information of a person whose fingerprint has been found in a crime scene rapidly in a massive database of records. However, there are two challenges against this goal. The first is that the local computation power is limited and the number of records is enormous. The second is to compare a pair of fingerprints, the features of fingerprints are needed to be extracted and compared which is a cumbersome computational task.

Our proposed implementation aims at utilizing the computation power of hybrid or multi-cloud. In this regard, we utilize a middleware framework, named Aneka [4], which is a Platform-as-a-Service (PaaS) solution and provides Application Programming Interfaces (APIs) for the developers to deploy their applications. We build and deploy a finger matching application on Azure cloud resources using Aneka Software Development Kit (SDK). For fingerprint verification, we use a framework presented in [5].

The rest of the paper is organized as follows. Section 2 discusses the related work. Section 3 represents the system architecture and how it works. Section 4 defines the main modules and interface of the system in detail. We evaluate and analyze the performance of our system in section 5. Finally, section 6 is dedicated to the conclusion and future works.

2 Related Work

A similar work to ours is fast fingerprint identification for large databases [6] where authors proposed a distributed framework for fingerprint matching for large databases. Similarly, their framework is also flexible to any kind of fingerprint matching algorithms. Le et al. [7] and Cappelli et al. [8] benefit from graphics processing unit (GPU) computing implementation of a fingerprint matching algorithm to speed up their system performance on large databases. These algorithms have been implemented completely using CUDA [9] where they have used different parallelization approaches. CUDA is a parallel computing framework which enables software developers to use GPU for general purpose processing. This approach is called General-Purpose computing on Graphics Processing Units (GPGPU). Even though the performance gain is high, the implementation of the parallel algorithms in CUDA is relatively hard and cumbersome. In this paper, we used Aneka platform and its easy-to-use and high-level APIs [4] which give us the power to concentrate on task distributions and make it flexible to use different fingerprint matching algorithms.

Many studies in cloud computing have addressed the expansion of local infrastructure capacity by using public cloud resources. Toosi et al. [1] focus on data-intensive applications and consider the impact of data transfers in their decision for using local or public infrastructure. Mateescu et al. [10] propose a High-Performance Computing (HPC) infrastructure architecture to execute scientific applications. Assunção et al. [11] examine the usage cost and the per-

formance of different public cloud resource provisioning algorithms. Belgacem and Chopard [12] conduct an empirical study of running a massively parallel MPIs application over an existing HPC infrastructure and bursting into Amazon EC2 clusters if the need arises. They aim to evaluate the overhead of using public cloud resources. Yuan et al. [13] intend to utilize the temporal variation of prices public clouds to maximize profit in a hybrid cloud environment.

3 System Architecture

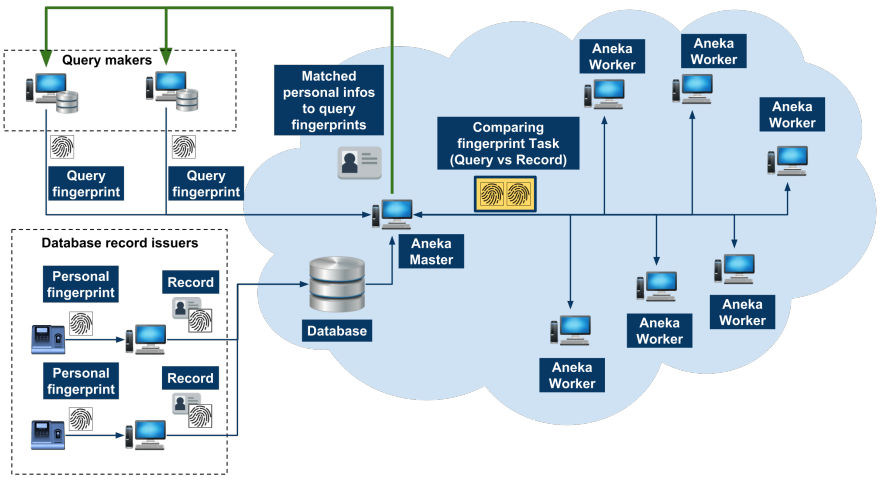

Fig. 1: System architecture.

This section provides a general overview of how the entire system works. Figure 1 visualizes the system architecture. The system contains three components which are database record issuers, query makers, and cloud-based fingerprint searching.

The *database record issuers* is the part of the system which provide the input data for the system. The peoples' fingerprints are binded with their personal information (i.e personal photo and name and etc.) and stored in the database. For the simplicity of the system, we use a file-system to store all records.

The second part of the system is the *query makers*. Queries are requested to find the matched fingerprints to the query fingerprint of interest. In this regard, requests are sent to the main component of the system which is *cloud-based fingerprint searching*.

The cloud-based fingerprint searching is controlled by Aneka cloud platform. Considering that the number of records in the database and the number of

queries can be huge, the importance of parallel searching in the database is obvious. The requests are given to the Aneka master (main node) which is responsible to make and distribute the comparison tasks among Aneka workers. Each comparison task consists all query fingerprints and a single fingerprint record in the database. The total number of comparison task is equal to the number of database records and independent to the number of queries. Aneka workers return matched similarity index between the fingerprint database record and query fingerprints to Aneka master. Aneka master gathers all the similarity indexes computed by Aneka workers, finds the maximum similarity, retrieves the personal information of matched fingerprints to query fingerprints and returns them to the query makers.

Two main components of the used framework for fingerprint recognition which are computationally heavy are 1) extracting features from fingerprint images and 2) comparing fingerprint features. To avoid re-extracting feature of both query and database fingerprints in the searching procedure, Aneka master extracts the feature of query fingerprints, and Aneka workers extract the features of database fingerprint records and compare them against the query fingerprint features.

4 System Design and Implementation

Figure 2 shows a layered view of our system's key components. It also provides the data flow in the system. The layered system design contains three layers.

The top layer belongs to the fingerprint recognition application and its main functionalities. The user is able to make a new record to store in the database or search a fingerprint through the database to retrieve the information of the person who is matched with the query fingerprint. Upon the receipt of a request for fingerprint matching, the features of the query fingerprint are extracted. Task Manager is the part of the system which tells how many database fingerprint records are needed to be packed in a comparison task. For the sake of simplicity, in this paper, we only pack one record in each comparison task. Task builder is the component that makes comparison tasks. It uses Aneka ITask interface to prepare the comparison task for the master. Later on, master submits Tasks to the workers for the execution.

The middle layer belongs to the Aneka cloud which performs as a middleware providing computational resources to the fingerprint matching application. Aneka tasks which are created previously in the application layer are buffered in the task queue in the Aneka master. Aneka master submits Aneka tasks to the Aneka workers for comparing record and query fingerprints. After receiving Aneka tasks by Aneka workers, they start to extract the features of record in the task, and compare it to the query fingerprint features. Finally, they return the matching similarity index to the master node to aggregate results.

The bottom layer provides computational resources for the Aneka platform to execute its tasks. Resources residing in this layer are obtained and managed

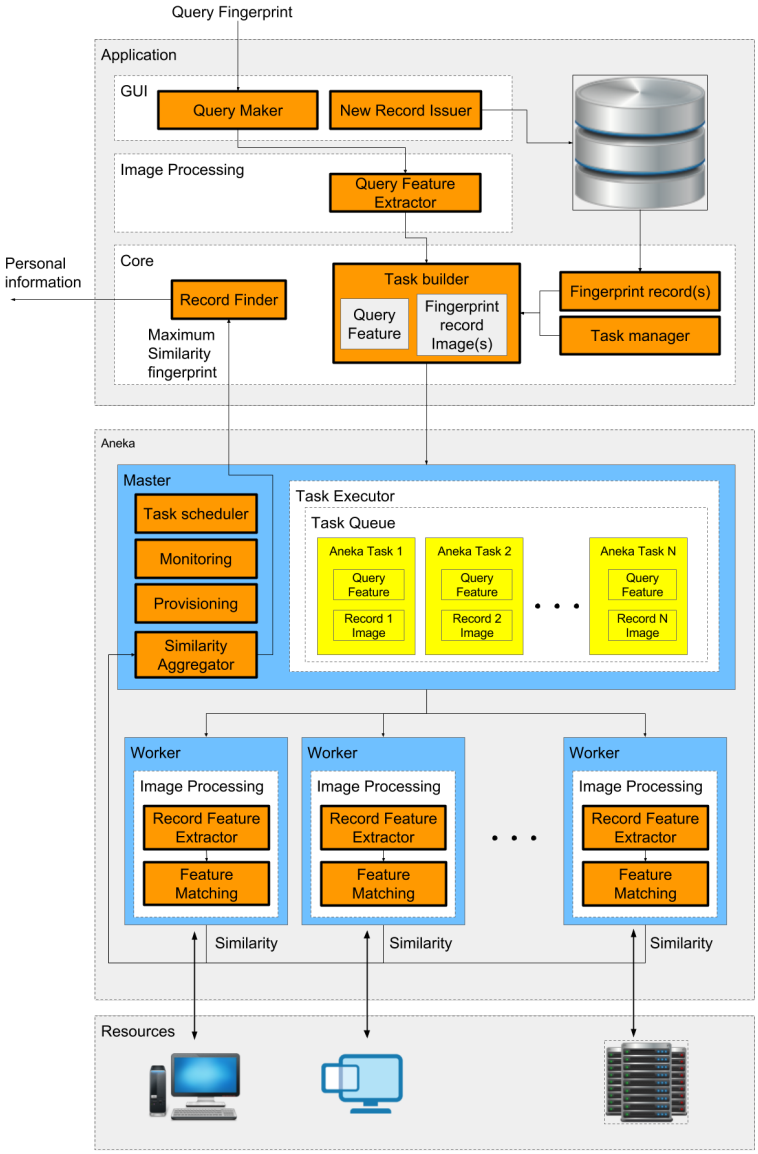

Fig. 2: System design and implementation.

by Aneka from a variety of sources, including private and public clouds, clusters, grids, and desktop grids.

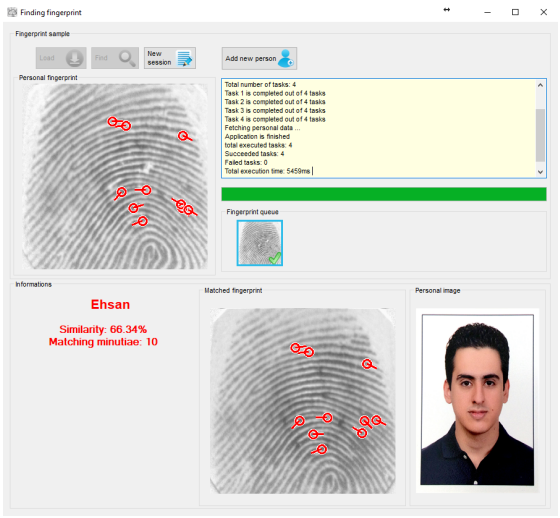

Fig. 3: Application GUI

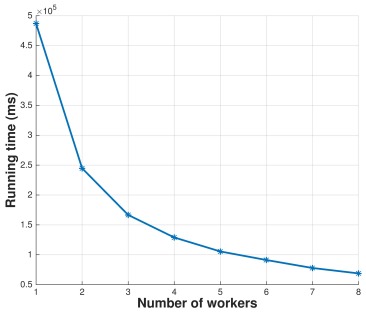

Fig. 4: Running time vs number of workers

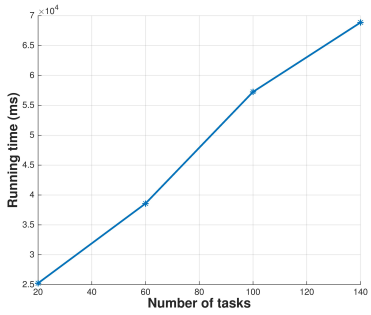

Fig. 5: Running time vs number of tasks

5 Performance Evaluation

In this section, we evaluate the performance of the system in terms of running time using two different experiments. For this purpose, we used the GUI of our application shown in Figure 3. Query fingerprints are queued for finding the matched person. The personal information of the matched person along with the matched fingerprint are displayed. There is also a console showing the progress of fingerprint searching such as tasks distribution, total running time, etc.

Both experiments are run on a master and a set of worker machines. The master runs on a desktop machine residing at the University of Melbourne and workers are provisioned from the Microsoft Azure Australia Southeast region. The master machine is an Intel Core i5-430M (2 Cores and 4 Logical Processors)

at 2.27GHz, 8GB main memory and runs under Microsoft Windows 10 Pro operating system. Worker machines are single core Azure Instances (Standard DS1) with a 2.4GHz processor and 3.5GB main memory running Windows Server 2012 as the operating system. We configured our application task manager to create an Aneka task (comparing task) per each fingerprint in the database.

In the first experiment, the number of tasks is fixed to 140 and on the other hand, the running time of the application is evaluated by varying the number of workers from 1 to 8. As expected, increasing the number of workers reduces the running time of the system. For instance, Figure 4 shows that the running time of the system for a single and two workers are almost 500 and 250 seconds, respectively, where the running time is reduced to half. Eventually, the running time reaches about 69 seconds when the number of workers is increased to 8.

In the second experiment, the number of workers is fixed to 8 and the running time is analyzed by changing the number of working tasks to 20, 60, 100 and 140. Figure 5 displays that the corresponding running time is nearly 25, 40, 57 and 68 seconds which is demonstrating a linear growth in time versus the number of tasks.

6 Conclusion and Future Work

In this paper, we demonstrated the benefits of using cloud computing for fingerprint recognition and matching for a large-scale database. We presented the design and implementation of our system including its architecture. We showed that how Aneka provides the required platform for scheduling and parallel execution of tasks on public cloud resources, e.g., Azure. A conducted performance evaluation showed that the fingerprint queries could be responded in a significantly lower timeframe using our proposed system.

As a future work, we are planning to devise a technique for dynamic resource provisioning based on the number of queries. The future direction will be to extend our system as a Software-as-a-Service (SaaS), one of the major categories of cloud computing, for the security-oriented organizations.

References

1. A. N. Toosi, R. O. Sinnott, and R. Buyya, "Resource provisioning for data-intensive applications with deadline constraints on hybrid clouds using aneka," *Future Generation Computer Systems*, vol. 79, no. Part 2, pp. 765 – 775, 2018.
2. C. P. Chen and C.-Y. Zhang, "Data-intensive applications, challenges, techniques and technologies: A survey on big data," *Information Sciences*, vol. 275, pp. 314–347, 2014.
3. R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Generation computer systems*, vol. 25, no. 6, pp. 599–616, 2009.

4. C. Vecchiola, R. N. Calheiros, D. Karunamoorthy, and R. Buyya, "Deadline-driven provisioning of resources for scientific applications in hybrid clouds with aneka," *Future Generation Computer Systems*, vol. 28, no. 1, pp. 58–65, 2012.
5. M. A. Medina-Pérez, O. Loyola-González, A. E. Gutierrez-Rodríguez, M. García-Borroto, and L. Altamirano-Robles, "Introducing an experimental framework in c# for fingerprint recognition," in *Mexican Conference on Pattern Recognition*. Springer, 2014, pp. 132–141.
6. D. Peralta, I. Triguero, R. Sanchez-Reillo, F. Herrera, and J. M. Benítez, "Fast fingerprint identification for large databases," *Pattern Recognition*, vol. 47, no. 2, pp. 588–602, 2014.
7. H. H. Le, N. H. Nguyen, and T. T. Nguyen, "A complete fingerprint matching algorithm on gpu for a large scale identification system," in *Information Science and Applications (ICISA) 2016*. Springer, 2016, pp. 679–688.
8. R. Cappelli, M. Ferrara, and D. Maltoni, "Large-scale fingerprint identification on gpu," *Information Sciences*, vol. 306, pp. 1–20, 2015.
9. J. Nickolls, I. Buck, M. Garland, and K. Skadron, "Scalable parallel programming with cuda," *Queue*, vol. 6, no. 2, pp. 40–53, 2008.
10. G. Mateescu, W. Gentzsch, and C. J. Ribbens, "Hybrid computingwhere hpc meets grid and cloud computing," *Future Generation Computer Systems*, vol. 27, no. 5, pp. 440–453, 2011.
11. M. D. de Assunção, A. di Costanzo, and R. Buyya, "A cost-benefit analysis of using cloud computing to extend the capacity of clusters," *Cluster Computing*, vol. 13, no. 3, pp. 335–347, 2010.
12. M. B. Belgacem and B. Chopard, "A hybrid hpc/cloud distributed infrastructure: Coupling ec2 cloud resources with hpc clusters to run large tightly coupled multiscale applications," *Future Generation Computer Systems*, vol. 42, pp. 11–21, 2015.
13. H. Yuan, J. Bi, W. Tan, and B. H. Li, "Temporal task scheduling with constrained service delay for profit maximization in hybrid clouds," *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 1, pp. 337–348, 2017.