IJCC Editorial Board

Editors-in-Chief
Hemant Jain, University of Wisconsin–Milwaukee, USA
Rong Chang, IBM T.J. Watson Research Center, USA

Associate Editor-in-Chief
Bing Li, Wuhan University, China

Editorial Board
Danilo Ardagna, Politecnico di Milano, Italy
Janaka Balasooriya, Arizona State University, USA
Roger Barga, Microsoft Research, USA
Viraj Bhat, Yahoo, USA
Rajdeep Bhowmik, Cisco Systems, Inc., USA
Jiannong Cao, Hong Kong Polytechnic University, Hong Kong
Buqing Cao, Hunan University of Science and Technology, China
Keke Chen, Wright State University, USA
Haopeng Chen, Shanghai Jiao Tong University, China
Malolan Chetlur, IBM India, India
Alfredo Cuzzocrea, ICAR-CNR & University of Calabria, Italy
Ernesto Damiani, University of Milan, Italy
De Palma, University Joseph Fourier, France
Claude Godart, Nancy University and INRIA, France
Nils Gruschka, University of Applied Sciences, Germany
Paul Hofmann, Saffron Technology, USA
Ching-Hsien Hsu, Chung Hua University, Taiwan
Patrick Hung, University of Ontario Institute of Technology, Canada
Hai Jin, HUST, China
Li Kuang, Central South University, China
Grace Lin, Institute for Information Industry, Taiwan
Xumin Liu, Rochester Institute of Technology, USA
Shiyong Lu, Wayne State University, USA
J.P. Martin-Flatin, EPFL, Switzerland
Vijay Naik, IBM T.J. Watson Research Center, USA
Surya Nepal, Commonwealth Scientific and Industrial Research Organisation, Australia
Norbert Ritter, University of Hamburg, Germany
Josef Schiefer, Vienna University of Technology, Austria
Jun Shen, University of Wollongong, Australia
Weidong Shi, University of Houston, USA
Liuba Shrira, Brandeis University, USA
Kwang Mong Sim, University of Kent, UK
Wei Tan, IBM T.J. Watson Research Center, USA
Tao Tao, IBM T. J. Watson Research Center, USA
Kunal Verma, Accenture Technology Labs, USA
Raymond Wong, University of New South Wales & NICTA, Australia
Qi Yu, Rochester Institute of Technology, USA
Jia Zhang, Carnegie Mellon University – Silicon Valley, USA
Gong Zhang, Oracle Corporation, USA
Call for Articles
International Journal of Cloud Computing

Mission
Cloud Computing has become the de facto computing paradigm for Internet-scale service development, delivery, brokerage, and consumption in the era of Services Computing, fueling innovative business transformation and connected human society. 15 billion smart devices would be communicating dynamically over inter-connected clouds by 2015 as integral components of various industrial service ecosystems. The technical foundations of this trend include Service-Oriented Architecture (SOA), business & IT process automation, software-defined computing resources, elastic programming model & framework, and big data management and analytics. In terms of the delivered service capabilities, a cloud service could be, among other as-a-service types, an infrastructure service (managing compute, storage, and network resources), a platform service (provisioning generic or industry-specific programming API & runtime), a software application service (offering email-like ready-to-use application capabilities), a business process service (providing a managed process for, e.g., card payment), a mobile backend service (facilitating the integration between mobile apps and backend cloud storage and capabilities) or an Internet-of-things service (connecting smart machines with enablement capabilities for industrial clouds).

The International Journal of Cloud Computing (IJCC) aims to be a reputable resource providing leading technologies, development, ideas, and trends to an international readership of researchers and engineers in the field of Cloud Computing. IJCC only considers extended versions of conference papers published at reputable conferences such as IEEE International Conference of Cloud Computing.

Topics
The International Journal of Cloud Computing (IJCC) covers state-of-the-art technologies and best practices of Cloud Computing, as well as emerging standards and research topics which would define the future of Cloud Computing. Topics of interest include, but are not limited to, the following:

- ROI Model for Infrastructure, Platform, Application, Business, Social, Mobile, and IoT Clouds
- Cloud Computing Architectures and Cloud Solution Design Patterns
- Self-service Cloud Portal, Business Dashboard, and Operations Management Dashboard
- Autonomic Process and Workflow Management in Clouds
- Cloud Service Registration, Composition, Federation, Bridging, and Bursting
- Cloud Orchestration, Scheduling, Autoprovisioning, and Autoscaling
- Cloud Enablement in Storage, Data, Messaging, Streaming, Search, Analytics, and Visualization
- Software-Defined Resource Virtualization, Composition, and Management for Cloud
- Security, Privacy, Compliance, SLA, and Risk Management for Public, Private, and Hybrid Clouds
- Cloud Quality Monitoring, Service Level Management, and Business Service Management
- Cloud Reliability, Availability, Serviceability, Performance, and Disaster Recovery Management
- Cloud Asset, Configuration, Software Patch, License, and Capacity Management
- Cloud DevOps, Image Lifecycle Management, and Migration
- Cloud Solution Benchmarking, Modeling, and Analytics
- High Performance Computing and Scientific Computing in Cloud
- Cloudlet, Cloud Edge Server, Cloud Gateway, and IoT Cloud Devices
- Cloud Programming Model, Paradigm, and Framework
- Cloud Metering, Rating, and Accounting
- Innovative Cloud Applications and Experiences
- Green Cloud Computing and Cloud Data Center Modularization
- Economic Model and Business Consulting for Cloud Computing
Table of Contents

EDITOR-IN-CHIEF PREFACE
Hemant Jain, University of Wisconsin–Milwaukee, USA
Rong Chang, IBM T.J. Watson Research Center, USA

RESEARCH ARTICLES

1 Cellcloud: Towards a Cost Effective Formation of Mobile Cloud Based on Bidding Incentive
Shahid A. Noor, Department of Computer and Information Sciences University of Alabama at Birmingham
Ragib Hasan, Department of Computer and Information Sciences University of Alabama at Birmingham
Md Haque, Department of Computer and Information Sciences University of Alabama at Birmingham

16 End-to-End Big Data Processing Protection in Cloud Environment Using Black Boxes - An FPGA Approach
Lei Xu, University of Houston, Houston, TX, USA
Khoa Dang Pham, University of Houston, Houston, TX, USA
Hanyee Kim, Korea University, Seoul, Korea
Weidong Shi, University of Houston, Houston, TX, USA
Taeweon Suh, Korea University, Seoul, Korea

30 Learning-based High-Throughput Dispatching for Trajectory Streams
Xin Zhang, Automation Department of Tsinghua University; IBM Research China
Guoqiang Hu, IBM Research China
Ning Duan, IBM Research China
Peng Gao, IBM Research China
Weishan Dong, IBM Research China
Jun Zhu, IBM Research China
Rong Chang, IBM Research, USA & China

45 Call for Papers: IEEE CLOUD/ICWS/SCC/MS/BigData/SERVICES 2015
Call for Articles: International Journal of Services Computing (IJSC)
Call for Articles: International Journal of Big Data (IJBD)
Welcome to the inaugural issue of International Journal of Cloud Computing (IJCC), the first open access on-line journal on cloud computing. The increasing importance of cloud computing is evidenced from the rapid adoption of this technology in businesses around the globe. The cloud computing is redefining the business model of various industries from video rental (Netflix is enabled by cloud) to small start-up companies (companies can be started with very little investment using cloud infrastructure). The potential of cloud computing is even more promising. The cloud computing combined with developments like internet of things can significantly change the life as we know today. However, to deliver on these promises and to prevent cloud computing from becoming a passing fad significant technical, economic, and business issues need to be addressed. IJCC is designed to be an important platform for disseminating high quality research on above issues in a timely manner and provide an ongoing platform for continuous discussion on research published in this journal. We aim to publish high quality research that addresses important technical challenges, economics of sustaining this environment, and business issues related to use of this technology including privacy and security concerns, legal protection, etc. We seek to publish original research articles, expanded version of papers presented at high quality conferences, key survey articles that summarizes the research done so far and identify important research issues, and some visionary articles. We will make every effort to publish articles in a timely manner.

This inaugural issue collects the extended version of five IEEE CLOUD 2013 articles in the general area of managing Cloud computing environment.

The first article titled, *QOS-Based Resource Allocation Framework for Multi-Domain SLA Management in Clouds* by Lu, Yahyapour, Wieder, Kotsokalis, Yaqub, and Jehangiri tackles the issue of downtime and service unavailability due to live migration. They present an OpenStack based implementation of a cloud resource allocation framework, named Generic SLA Manager, that supports downtime-aware VM selection and allocation during live migration of VMs. A simulation based evaluation of the proposed framework is reported as well.

The second article titled, "Rapidly Alternating Bottlenecks: A Study of Two Cases in N-Tier Applications" by Wang, Kanemasa, Li, Shimizu, Matsubara, Kawaba, and Pu reveals the importance of identifying the location of performance bottlenecks when scaling n-tier applications in computing clouds. They propose a bottleneck detection method that could be used to rapidly detect alternating bottlenecks. Experimental evaluation results for the proposed method are reported via two use cases.

The third article titled, "Cross Cloud MapReduce: A Result Integrity Check Framework on Hybrid Clouds" by Wang, Wei, and Srivatsa tackles the trust issue in adopting large-scale MapReduce on public clouds. They present a framework, named Cross Cloud MapReduce (CCMR), which overlays the MapReduce computation on a hybrid cloud where a master ensures correct result. A result integrity check scheme is also presented for accuracy and performance. Both theoretical and experimental analyses are reported.

The fourth article titled, "Implementation and Empirical Assessment of A Web Application Cloud Deployment Tool" by Sampaio, Costa, Mendonça, and Filho tackles the time-consuming issue in migrating applications to an IaaS cloud via application-specific VM images. They present an automated application deployment approach that requires less cataloged VM images. The approach can be supported via a tool, called TREXCLOUD, and an empirical evaluation of the tool is reported.

The fifth article titled, "Implementation and Empirical Assessment of A Web Application Cloud Deployment Tool" by Sampaio, Costa, Mendonça, and Filho tackles the time-consuming issue in migrating applications to an IaaS cloud via application-specific VM images. They present an automated application deployment approach that requires less cataloged VM images. The approach can be supported via a tool, called TREXCLOUD, and an empirical evaluation of the tool is reported.
The fifth article titled, “A Queuing Model to Achieve Proper Elasticity for Cloud Cluster Jobs” by Salah tackles the issue of achieving proper elasticity for parallelized jobs running on cloud clusters. Based on finite queuing systems, the article presents an analytical model that can be used to determine the minimal number of cloud resources needed to satisfy the SLO requirements with constraints. Discrete Event Simulation is reported to verify the correctness of the proposed model.

We would like to thank the authors for their effort in delivering those five quality articles. We would also like to thank the reviewers as well as the Program Committee of IEEE CLOUD 2013 for their help with the review process. Finally, we are grateful for the effort Jia Zhang and Liang-Jie Zhang made in giving birth to this inaugural issue of International Journal of Cloud Computing (IJCC).

About the Editors-in-Chief

Dr. Hemant Jain is the Interim Director of Biomedical and Health Informatics Research Institute, Roger L. Fitzsimonds Distinguished Scholar and Professor of Information Technology Management at University of Wisconsin–Milwaukee. Dr. Jain specializes in information system agility through web services, service oriented architecture and component based development. His current interests include development of systems to support real time enterprises which have situational awareness, can quickly sense-and-respond to opportunities and threats, and can track-and-trace important items. He is also working on issues related to providing quick access to relevant knowledge for cancer treatment and to providing medical services through a virtual world. Dr. Jain is an expert in architecture design, database management and data warehousing. He teaches courses in database management, IT infrastructure design and management, and process management using SAP. Dr. Jain was the Associate Editor-in-Chief of IEEE Transactions on Services Computing and is Associate Editor of Journal of AIS, the flagship journal of the Association of Information Systems.

Dr. Rong N. Chang is Manager & Research Staff Member at the IBM T.J. Watson Research Center. He received his Ph.D. degree in computer science & engineering from the University of Michigan at Ann Arbor in 1990 and his B.S. degree in computer engineering with honors from the National Chiao Tung University in Taiwan in 1982. Before joining IBM in 1993, he was with Bellcore researching on B-ISDN realization. He is a holder of the ITIL Foundation Certificate in IT Services Management. His accomplishments at IBM include the completion of a Micro MBA Program, one IEEE Best Paper Award, and many IBM awards, including four corporate-level Outstanding Technical Achievement Awards and six division-level accomplishments. He is an Associate Editor of the IEEE Transactions on Services Computing and the International Journal of Services Computing. He has chaired many conferences & workshops in cloud computing and Internet-enabled distributed services and applications. He is an ACM Distinguished Member/Engineer, a Senior Member of IEEE, and a member of Eta Kappa Nu and Tau Beta Pi honor societies.
**CELLCLOUD: TOWARDS A COST EFFECTIVE FORMATION OF MOBILE CLOUD BASED ON BIDDING INCENTIVE**

Shahid A. Noor, Ragib Hasan, Md Haque  
Department of Computer and Information Sciences  
University of Alabama at Birmingham  
{shaahid, ragib, mhaque}@cis.uab.edu

**Abstract**

In recent years, cloud computing has become one of the most dominant computing paradigms. Researchers have explored the possibility of building clouds out of loosely associated mobile computing devices. However, most such efforts failed due to the lack of a proper incentive model for the mobile device owners. In this paper, we propose CellCloud — a practical mobile cloud architecture, which can be easily deployed on existing cellular phone network infrastructures. CellCloud is based on a novel reputation-based economic incentive model in order to compensate the mobile device owners for the use of their phones as cloud computing nodes. CellCloud offers a practical model for performing cloud operations, with lower costs compared to a traditional cloud. We provide an elaborate analysis of the model with security and economic incentives as the major focus. Along with presenting a cost equation model, we perform extensive simulations to evaluate the performance and also analyze the feasibility of our proposed model. Our simulation results show that CellCloud creates a win-win scenario for all three stakeholders (client, cloud provider, and mobile device owners) to ensure the formation of a successful mobile cloud architecture.

**Keywords:** mobile cloud; bidding; challenges; trustworthiness; cost model;

---

1. **INTRODUCTION**

Cloud computing is a well-known computing model thanks to its competitive price, performance, and expandability. However, there is a tradeoff associated with cloud computing — hidden costs make it infeasible compared to private hosting [1], [2]. From the operational and structural point of view, the fixed structure of cloud data centers can cause underutilization of resources if there is a rapid decrease in clients’ demands for cloud services. Recently, chief providers of cloud service like Amazon and Microsoft did not succeed in earning the revenues they initially expected to get, because of the unforeseen shutdown of the government budget [3]. One of the main reasons for this deficit in the expected revenue is due to the inflexibility of these organizations to contract and expand the resources based on the client requirements. We argue that, a cloud service can be designed that is not subject to underutilization of resources, if the servers themselves could be outsourced from the cloud service providers to individuals with excess resources. In this model, using mobile devices, it is possible to form a highly scalable ad hoc mobile cloud with low infrastructure set up cost and time.

Using mobile devices, it is possible to create a highly scalable, ad hoc mobile cloud with an easy to avail infrastructure, low budget, and short time frame. For this reason, mobile cloud computing is introduced where unlike a traditional cloud, a virtualized interface is formed using mobile devices.

Researchers have defined mobile cloud from two aspects. According to the first aspect, mobile cloud computing is an infrastructure where mobile users use backend cloud system for storing and processing data required to run an application [4]. The second aspect contends that, mobile cloud computing enhances the storage and computational power of the cloud system by using the unused resources of mobile devices [5]. In this paper, we would like to shed light on the second aspect of mobile cloud. A few applications utilize mobile sensed data, which is both lengthy and costly in sending to the traditional cloud for processing. A better approach would be to process data on a local basis using the mobile cloud as explained in the second aspect. Another benefit of the second aspect of mobile cloud is the accessibility of millions of unused mobile devices. The survey of Lockout Inc. data show that around 20, 16, and 19 percent of the people have respectively one, two, and more than two unused mobile devices [6]. We argue that the computing capability of such devices can and should be utilized.

There are many benefits of choosing a mobile cloud over traditional cloud. The *first* benefit is that a mobile cloud requires low set up and maintenance costs compared to traditional cloud. The *second* benefit is that a mobile cloud can be expanded to suitably keep pace of the growing demands of a client. A *third* benefit is that tasks can be effortlessly distributed and transferred among mobile devices as needed since the infrastructure of mobile network is already available in most places. Finally, according to Alzain et al. [7], Bendahmane et al. [8], and Lagar et al. [9], one can summarize that...
maintaining higher redundancy in task computation guarantees more accurate results. For this reason, there is a higher possibility of obtaining legitimate results in a mobile cloud than a traditional cloud since there are more unutilized mobile devices that can be used for the repetitive computation of a single task.

Most of the research performed up to now focus on the first aspect of mobile clouds which utilizes the cloud in the backend to enhance the storage and computational power, battery longevity, safety, and security of mobile device [10], [11], [12], [13]. Most researchers have not foreseen a more innovative approach where the use of mobile devices plays an integral part of a cloud [14], [15]. Low storage and computing power of the unused mobile devices were the biggest stumbling blocks for researchers to exploit the opportunity of forming mobile clouds with these devices. However, with the advance of technology, researchers have just started considering the second aspect of mobile clouds where mobile devices are used as integral part of a cloud [14], [15]. Though these models [14], [15] include architecture for forming a mobile cloud, the absence of appropriate cost/incentive model fails to motivate the mobile device owners to participate. These models also do not address the feasibility of utilizing these unused mobile devices from the provider’s perspective. To solve this, we introduce CellCloud – a practical mobile cloud architecture, which can be easily deployed on existing cellular phone network infrastructures. In CellCloud, we address these issues by exploring a bidding strategy for providing incentives to mobile device owners, and also quantify the benefits achieved by cloud providers by using mobile devices as cloud nodes.

In CellCloud, mobile devices known as bidders are hired following a bidding process. During the bidding process, each bidder is offered monetary incentive based on their available resource and rating point. The rating point determines the trustworthiness of the bidder for a particular task. Based on client requirement, CellCloud provider hires the required number of bidders for a task, divides the task into smaller subtasks, and distributes them among bidders for computation. The CellCloud provider uses a MapReduce based scheme for its computations. The overall model proves to be cost effective for both cloud providers and mobile owners creating a win-win situation for all the stakeholders.

Initial results from CellCloud were presented in [16]. In this paper, we provide a comprehensive discussion of our expanded model, and present extensive and new simulation results from additional benchmarks of various aspects of CellCloud’s performance in order to demonstrate the feasibility of this model.

Contributions:

1) To the best of our knowledge, CellCloud is the very first attempt to form a mobile cloud using the network of the mobile operators and unused mobile phones. We introduce a bidding process for forming the mobile cloud where mobile device owners can submit their free resources during bidding and get incentives for their resources.

2) We introduce the novel concept of rating points associated with the bidders (mobile devices) to ensure trustworthiness and reliability of the service in hand. Moreover, our model enables users to choose different service costs based on the rating level of the computing nodes, thus providing a unique opportunity to the clients.

3) We provide simulation results on an LTE network topology using the NS-3 simulator to demonstrate the feasibility of the CellCloud model.

The rest of the paper is organized as follows. Section 2 describes the motivation of our research work. In section 3, we discuss some of the related works on mobile cloud. Section 4 introduces our mobile cloud architecture. We describe the strategies for assigning rating points, measuring the cost of the operation, and selecting base stations and bidders in section 5, 6, 7, and 8 respectively. Section 9 elaborates possible challenges in mobile cloud. We define some policy for the client and the bidder in section 10 followed by experimental results in section 11. Some possible applications of CellCloud are mentioned in section 12. Finally, we conclude with discussion and future directions in sections 13 and 14 respectively.

2. Motivation

Mobile cloud computing requires a significantly smaller amount of initial set up cost as compared to a traditional cloud system. The primary reason of this low set up cost is that, it utilizes the unused mobile devices to create a cloud platform. A study by Lockout found that, approximately 52% mobile users intend to donate their old sets for charitable use [6]. Moreover, it is also observed that, mobile devices are in idle state for 89% of the time in a day and during that time the devices use less than 11% of total CPU power [17]. Therefore, with no obvious harms and associated economic benefit, we believe that it would be very easy to motivate the mobile device owners to share their unused mobile devices on a cloud platform.

Scalability of the client tasks is another big reason to choose mobile cloud than traditional cloud systems. Sometimes, there can be more demand for resources than expected. In general, a good amount of resources remain unused in a public cloud, as the providers want to be on the safe side. Therefore, if at any time there is a certain decrease in client demand, some resources will be unused. Recently, Amazon, Microsoft, and some other cloud services failed to earn expected revenues due to the unexpected shutdown of government budget due to the stalemate in the US legislature [3]. On the other hand, in our CellCloud architecture, we always hire bidders on an
on-demand basis. Since the infrastructure of the mobile phone network is already deployed and available, the cost for keeping continuous connection between the bidders and the base stations is almost negligible. We can always have a good number of reserve bidders. Therefore, our CellCloud architecture will be able to handle both the sudden increase and decrease of clients without any financial downfall.

A mobile cloud can reduce the computational cost significantly as compared to the traditional backend cloud system. Mahesri and Vardhan showed that the average power consumption of a personal computer with a 1.3 GHz processor in idle state is 13.13W [18] whereas, that of a mobile phone with a 400 MHz processor is only 268.8mW [19]. Hence, the total power consumption in a mobile cloud is much less than a traditional cloud. Therefore, we can include more bidders in our CellCloud to achieve the same computing capabilities as provided by a traditional cloud. This comparatively higher power consumption issue also plays a big role in higher operational cost for traditional cloud services.

Ensuring trustworthiness during computation is another major reason to choose a mobile cloud over the traditional clouds. Bendahmene et al. discussed two popular methods for ensuring the authenticity in cloud computation: majority based voting and m-first voting system [8]. However, both of these methods use multiple virtual devices for computing a single task. We use a similar approach in our CellCloud for enhancing trustworthiness. Since the level of trust is associated with the level of redundancy in task computation, availability of larger number of free bidders will help us in getting more precise results. In CellCloud, we always have a large number of unused bidders, which can be used for redundant computation.

3. RELATED WORK

Mobile computing is used for developing several applications in the field of distributed computing. Beberg et al. proposed Folding@home, a distributed computing methodology, for reproducing biophysical methods [20]. As indicated by their building design, volunteer users with an interest to participate in the computation allow their personal computers to be used. The server sends the users the data about a specific work unit, which is the mixture of some input files needed for completing a job inside a certain time period. A header is connected within a work unit to set the core type while the version inside a work unit is used to download and process the core. A core can be conceived as an executable file, which contains input files and generates output. The output created by the designated computational core is sent to the server. Since cores are autonomous from customer in this manner, this methodology can do any kind of processing and the overhaul of centers does not require reinstallation of any software. The significant downside of this methodology is that once a task is given, client has to follow and download the essential core type manually. Therefore, it is unlikely that they would like the complexity associated with the operation. In contrary, in our CellCloud architecture, the inputs are provided along with the requirements during the time a bidder participates on bidding. Based on the instruction provided by cloud, the bidder will just perform computation.

Condor is a very famous project that is initiated by the researchers of the University of Wisconsin Madison [21]. It manages, circulates, and maintains a wide scope of computing systems. The project has the flexibility to match any input request for resource, assigning checkpoint along with migration facility. Moreover, it allows remote system calls to run jobs remotely. The jobs with higher priority are executed first without any interruption. The lower priority jobs are executed while the CPU is in idle and transferred to another one as soon as the CPU becomes busy. The user job is taken by the agent associated with the Condor kernel, while the resources are monitored by the matchmaker. As soon as a match is found, the job is transferred to that specific resource. One major problem of Condor project is that when a higher priority job comes and the system does not have enough resource, the lower priority jobs are interrupted. Hence, estimating the task completion time is very difficult for the client. On the contrary, jobs are independent of each other in our CellCloud architecture. Resources are allocated for a client as soon as a job is accepted. If there is not enough resource for satisfying the client requirements, then CellCloud will not accept the task. Therefore, it is easier for client to anticipate the time and cost of task completion in CellCloud.

Miluzzo et al. proposed the concept of mClouds where group of mobile devices, known as mDevs, are brought together to form a cloud computing platform [15]. Whenever a mobile device wishes to compute a task, which requires larger resources than it currently has, the device broadcasts a solicitation message to inform the other mobile devices to join its mCloud formation. The mobile device divides and distributes the task among the mCloud devices. If the task is too large to finish even after forming the mCloud, then the device uses the backend cloud system for the remaining subtasks, as no mDevs are available. In mCloud system, mobile users have to decide when to form the mCloud and whether joining on mCloud will be beneficial or not. The primary problem in mCloud system is to ensure security since anyone having a mobile device can join without verification. Therefore, the presence of a rogue device can lead to wrong output. On the other hand, all the devices used in our CellCloud are verified by the operator providing cloud service before they are included on cloud. Therefore, we can ensure more accurate and trustworthy results in CellCloud than the mCloud system.
Researchers from the Space Science Laboratory of University of California created the SETI@home project to explore the presence of life in the universe [22]. In their project, they consolidate immense computing power distributed all around the world to examine radio telescope signals come from space. A large number of data are broken into smaller chunks and distributed among a large number of computers for processing. Result obtained from each computer is organized by central repository. Compared to the typical distributed systems, SETI@home uses more variety of resources distributed among diverse locations. However, the major problem in SETI@home project is that the tasks are distributed only if the resources are in idle state. Therefore, it rarely provides any real time solution for a task. Some tasks might need to be finished within a specific deadline. On the other hand, in our CellCloud architecture, client can inform the deadline for their task and CellCloud selects the resources based on the client requirements.

Dashti et al. introduces an approach to use smartphone sensors for effectively and efficiently detecting the effect of an earthquake [23]. In their approach, smart phones are placed into an artificially shaking table. The shaking table can move in every direction and can horizontally accelerate up to 1.5 g even to a 100,000 lb object. The time needed for the smartphones to send their data, along with the position and orientation, is determined and used during prediction. A shaking meter continuously detects shakes and only starts recording while the meter drops below a threshold. The emergency responders can summarize the result sent by those sensors and take the necessary steps based on the summary after an earthquake. Their approach does not produce sparse result unlike the result produced by the existing methods. Direct human observation during the earthquake might help in detecting the actual situation of earthquake but it is highly related to how fast the user responses. Moreover, untrained users might provide misleading information. Therefore, using smartphone sensors for observing real time earthquake data as proposed in their research work will be both inexpensive and efficient as they are now easily available and the sensors perform quite well in sensing the environment.

Mednis et al. discuss the uses of mobile sensors to detect pothole on the road. The first approach Z-THRESH compares the accelerometer reading values with a threshold value and any value exceeding the threshold indicates the possibilities of pothole. In second method, the difference between two values above a threshold is used to determine the pothole. The third method first determines the standard deviation of the acceleration on vertical X-axis, which is compared with the threshold to detect pothole. The fourth method considers threshold values in the entire three axes for measuring pothole. One problem on their proposed work is, there can be several values above a threshold. But it is unclear which two values will be taken from those values. Similarly the authors never defined clearly the term sliding window. So its hard to imagine the effect of the size of sliding window on determining true positives and true hit. From their experimental result it is also seen that the performance of all the methods other than Z-DIFF is not convincing for detecting drain pits.

Angin et al. proposed a Context-aware blind navigation system, which uses the resources of cloud for complex computations [25]. In their method, a cameral is attached with the sunglass, which captures the visual records and sends it to the mobile device via Bluetooth. The mobile device senses the voices and its integrated camera determines the position. The mobile device processes the data for identifying the context. The positioning and direction is precisely determined by using the combination of GPS, Wi-Fi access point, and cell tower triangulation along with a compass. The authors used android as a mobile platform and Amazon EC2 as a cloud platform. For detecting the 3D real-time images, they choose time-of-flight range cameras. Their proposed method, in addition to reaching the destination properly, also supports blind people to effectively track their personal belongings along with the identifying people surrounded by them. In addition, it provides several advantages as compared to the normal stereo cameras such as, better detection of actual dense, simpler and efficient processing etc. Moreover it can guide during outdoor navigation such as how to cross an intersection in urbane area, detecting an obstacle, locating a bus stop along with the pedestrian signal.

Hoang et al. proposed a mobile cloud architecture that composed of sensor and mobile agent component which are responsibility to collect and manage data from the environment [26]. They included several features to maximize the utilization ratio. For example their asyynchronous message mode (AMM) activates a mobile application, which sends a request to cloud server and wait until a response comes from the server. For implementing AMM they uses a push mechanism, which helps to run application in the background so that the energy can be saved. A database is embedded with their system, which relocates the relevant data when the connection is lost. The context aware mobile and
middleware section of their proposed architecture used the contextual information such as quality of session, condition of network, temperature, humidity location, etc. during processing the information. They use an intelligent learning model for creating and updating their context repositories. This contextual form of data can be used to reduce the energy consumption of a specified application. A middleware layer is placed between data acquisition layer and back end cloud layer to handle the power of the context aware data along with the activation of emergency if required and dynamic allocation of the back end cloud resources. However, It is not clear from their work that how does they handle the power or network failure. No algorithm is mentioned for handling such unwanted situations.

All the aforementioned approaches have some common problems such as, the motivation to participate in cloud architecture on part of the device owners is overlooked, the requirements of client for a task is not mentioned, and no cost benefit analysis is provided for client to take decision on joining cloud system. On the contrary, in CellCloud, the cloud service provider hires bidders by providing incentives. As a result, we can expect to get a large number of bidders for tasks. Clients can also submit their requirements for a task and the CellCloud provider provides the estimated cost of that task. Therefore, clients can analyze their benefits before giving any task to CellCloud.

Moreover, CellCloud provides a pricing chart for the client from where they can get the idea of the possible time and cost for their task completion. Clients can easily verify whether assigning a task to our mobile cloud will be beneficial for them or not.

4. **CellCloud Architecture**

The CellCloud architecture consists of a cloud central system (CCS), which is the central part of the operator cloud system. During the cloud set up, CCS sends a message to all of its base stations to inform the mobile users under their coverage area to initiate bidding. CCS determines the price for hiring a bidder based on the rating point of the bidder. The interested bidders submit the information regarding their available resources to the corresponding base stations. Each base station maintains a table consisting of the bidder id, rating point, and available resources. The high level architecture of CellCloud is shown in figure 1. However, the following two scenarios can be possible while a bidder wants to participate in a task:

**Scenario 1:** If the bidder is not registered and sends a request to the corresponding base station for the first time, the base station adds a new entry into its bidder information table. The base station requests CCS to assign an id for the new bidder. The id is stored into the id field of the table. A default initial rating point of 0.5 is assigned for the bidder. The bidder receives its id from the base station and stores it along with the id of the current base station.

**Scenario 2:** If an existing bidder requests to the corresponding base station for the participation of a task, it needs to send its id along with the id of last base station it visited for a task computation. The base station collects the information of that bidder from that last visited base station. The current base station updates its bidder table while the previous base station deletes the entry associated with that bidder.

![Figure 1. CellCloud Architecture](image-url)
amount of available resources. The operational unit divides the subtask again into several un-uniform subtasks and distributes among the bidders based on their resources. The process of mapping a task inside an operational unit is depicted in figure 2.

Each bidder sends its result to the reducer after computation. Each reducer continuously collects results from the bidders and performs reducing operation on them. Once the reducer computes the final result after reducing all the results obtained from the bidders, it sends the result to CCS. CCS collects the reduced result from each base station inside an operational unit and starts reducing the result. The final result is sent to the client. The process of reducing the result is shown in figure 3.

Once a subtask is finished, base station measures the performance of each bidder under its coverage area. Based on the performance, the base station re-evaluates and updates the rating point for each bidder.

5. Rating Point Calculation

The rating point denotes the level of trust the cloud provider has on a bidder of the cloud system. The rating point of each bidder ranges from 0 to 1. A bidder with a higher rating point is considered to be more trustable compared to a lower rating point bidder. Initially each bidder is assigned a rating point of 0.5. Upon the successful completion of a task, we provide a reward that will increase bidder’s rating point. On the other hand, we will penalize if the bidder fails to finish the task within deadline. However, the rate of penalize will be more as compared to the rate of reward. Before sending a task to a bidder, base station will estimate the possible task completion time \( t_0 \) based on its resources. In addition, base station will define two more time. The max time \( t_u \) and the min time \( t_l \) which are 10 percent larger and smaller respectively than the original estimated time. We assign a reward of 1, 0.7 and 0.5 for completing the task within the time \( t_u \), \( t_0 \) and \( t_l \) respectively. On the other hand, a bidder will not be given any reward if it does not finish the task before \( t_l \). The new rating point will be the average of the current rating point and the reward point. In other word, we always emphasize the most recent task during the calculation of the new rating point. Thus, if the current rating point of the users is \( P_c \) and the reward for the most recent task is \( P_r \), then the new rating point will be,

\[
P_n = \frac{P_c + P_r}{2}
\]

6. Measurement of Cost and Time

For calculating the cost, we consider the cost associated with hiring the bidders and the cost associated with using network devices. The hiring time of a bidder is the addition of the time taken by the bidder to receive the task; the time bidder is used for task computation, and the time taken by the bidder to send the result. However, the consumed power in each mobile device during sending and receiving time is proportional to the rate at which the mobile device transfers and receives data. Suppose the power drop for sending and receiving at a rate of 1 unit/s is \( P_x \) and \( P_y \) respectively. If the bidder sends and receives data at a rate of \( x \) unit/s and \( y \) unit/s then the power drop for sending and receiving is \( P_{x,s} \) and \( P_{y,r} \) respectively. If the bidder sends data for \( t_s \) times and receives data for \( t_r \) times, then power consumption for receiving a task and sending result will be,

\[
P_{en} = t_s P_{x,s} + t_r P_{y,r}
\]

Let us assume that the power drop during task computation is \( P_t \) in each unit of time. If the bidder takes \( t_t \)
times for computing a task then the consumed power during computing a task will be,
\[ P_{task} = tP_i \]

Suppose the cost for consuming each unit of power is \( C_u \) thus the cost for total power consumption will be,
\[ C_p = C_u (P_u + P_{task}) \]

Let us assume that bidder with rating point 0.1 receives \( m\% \) profit of total cost for power consumption.

If the rating point of the bidder is \( R_b \) then the cost for hiring a bidder will be,
\[ C_h = (100+m) \times C_p \times R_b \]

For calculating the cost associated with the use of network devices, we need to know the cost of mobile operator for transferring each unit of data within the network. Let us assume that the cost of mobile operator is \( C_d \) for transferring each unit of data. Therefore, if the bidder receives \( p \) unit and sends \( q \) unit of data the cost of the operator will be \( C_d(p+q) \). If we assume that the mobile operator makes \( r\% \) profit on its total cost for sending and receiving data then the cost associated with network will be,
\[ C_n = C_d(p+q) \times (100+r)/100 \]

Hence, the cost for completing a t unit of task by a single bidder will be,
\[ C_b = C_n + C_h \]

If \( n \) bidders \( B_1, B_2, ..., B_n \) are hired for a task with the cost of \( C_1, C_2, ..., C_n \) respectively then the total cost for task completion will be,
\[ C = C_1 + C_2 + ... + C_n \]

For each base station, we consider the following times during a task computation:
- Time to divide the task among \( n \) segments \( (t_d) \)
- Time to send the segments of a task to bidders \( (t_s) \)
- Maximum task completion time among all bidders \( (t_{max}) \)
- Time to receive the results by base station from the bidders \( (t_r) \)
- Time to reduce the results \( (t_{rd}) \)

Suppose \( n \) number of bidders \( B_1, B_2, ..., B_n \) participate on computation where each bidder takes \( x_i \) size of task as input and produces \( y_i \) size of result. If the data transfer rate between the bidder and the base station is \( BW \) then the time that will be taken by each base station for computing its assigned task will be,
\[ t_{bs} = t_r + \frac{\sum_{i=1}^{n}(x_i + y_i)}{BW} + t_{max} + t_{rd} \]

Suppose there are \( n \) base stations \( b_1, b_2, ..., b_n \) inside an operational unit and each take \( t_{bs1}, t_{bs2}, ..., t_{bsn} \) time to finish the task. If the CCS takes \( t_{div} \) and \( t_{red} \) time to distribute the task and reducing the result then the total time for a task computation will be,
\[ T = \text{maxtime}(t_{bs}) + t_{div} + t_{red} \]

Here maxtime is the maximum task completion time among all the base stations.

7. **BASE STATION SELECTION STRATEGY**

In the Base Station bidding strategy, the reliability scales from 0 to 1. The client submits his required reliability \( R \) along with the deadline \( T \). CCS broadcasts a message to all the base stations and ask for total free resources it can provide with an average rating point of \( R \).

Suppose \( p \) out of \( n \) base stations reply with the resource amount \( r_1, r_2, ..., r_p \) respectively. Based on the task size and the deadline, CCS estimates total resources required for finishing the tasks. CCS divides and distributes the task into several smaller subtasks in such a way that each of these \( p \) base stations receives equal load on their available resources. If \( M \) is the total resources required for finishing the task of size \( S \) and CCS selects \( r_1', r_2', ..., r_p' \) resources from base stations \( b_1, b_2, ..., b_p \) respectively then,
\[ r_i' = M \times \frac{r_i}{\sum_{j=1}^{p} r_j} \]

8. **BIDDER SELECTION STRATEGY**

Suppose a base station receives a request for resource \( M \) from CCS with an average rating point of \( R \). The base station uses the procedure as described in Algorithm 1, to select the bidders from its available bidder list.

**Algorithm 1 Bidder Selection**

1: set selectedbidders=\null
2: sort bidders into ascending order \( b_1, b_2, ..., b_m \) based on the rating point \( p_1, p_2, ..., p_m \)
3: find \( k \) such that \( p_i \geq R \forall i \geq k \) and \( p_i \lt R \) otherwise
4: set upperpointer=k and lowerpointer=k-1
5: push \( b_{upper pointer} \) into selectedbidders
6: set upperpointer=upperpointer+1
7: find resourcesum which is the sum of the resources of selectedbidders
8: if resourcesum \geq M \) then
9: \quad return selectedbidders
10: else
11: if average rating point of selected bidders \geq target rating point then
12: \quad push \( b_{lower pointer} \) into selectedbidders
13: \quad set lowerpointer=lowerpointer-1
14: else
15: \quad push \( b_{upper pointer} \) into selectedbidders
16: \quad set upperpointer=upperpointer+1
17: \quad end if
18: \quad go to step 7
19: \quad end if
whether the average rating point of currently selected bidders are less than or greater than the target rating point. Base station repeats the process of selecting bidders until the required amount of resources for the task is achieved.

9. Challenges in Bidding Strategy

We listed several unwanted circumstances in our proposed model, which are described below.

9.1 Finishing Task on Deadline.

One of the major challenges in our architecture is to ensure that, each task will be finished before its deadline. In our CellCloud architecture, CCS only accepts a task if the cloud has sufficient resources. Task completion may be delayed due to some unwanted circumstances such as, network failure or bidder negligence. However, the bidder knows that it will receive more incentives if it has higher rating point and only finishing the task on deadline could enhance its rating point. Therefore, the bidder will sincerely try to complete the task on time.

9.2 Insufficient Battery Power.

For finishing the tasks on time, mobile devices need to remain switched on. Unexpected power failure can stop computational process or result in loss of computed data. Therefore, it is very important for CCS to know the current power status of the mobile device so that it can collect the result so far computed in case of possible power failure and distributes the remaining tasks to the backup bidders. We can use the agent SystemSens in each computational device to monitor battery power [27]. A threshold level $P_{th}$ is defined and if the power goes below that threshold then the agent communicates with the CCS to inform possible power loss. Depending on the user’s requirement, the base station may consider some of the bidders as reversed bidders and may contact them when required. Base station forwards the incomplete tasks to one of those reserved bidders on the cloud system. Base station sends a confirmation to both the current bidder and the new bidder when the incomplete tasks are transferred to the new bidder.

9.3 Trusted Computing.

Another major concern for the provider is to ensure that any malicious program inside a mobile device cannot manipulate computation or compromise data inside the mobile device. Using a trusted cloud computing platform (TCCP), we can prohibit malware to access input and output data, or stop interfering during computation [28]. To establish a TCCP, a trusted virtual machine monitor (TVMM) [29] is installed in a mobile device if the platform inside the mobile device satisfies the specification defined by the trusted computing group (TCG). TVMM prohibits even the privileged user from observing or altering the data during computation. A trusted coordinator (TC) inside TCCP certifies a platform if it finds the platform secure for computation. A bidder only accepts input data and performs computation if TC certifies it.

10. Policies in CellCloud

The list of policies in our mobile cloud architecture includes:

10.1 Task Selection Policy.

We expect a large number of requests from client at a certain period of time. The clients will be served on first come first serve basis. A client will be served only if the cloud has enough resources to fulfill clients’ requirements. Otherwise, clients will be informed regarding the limitation of resources. The client might change its requirements and request for cloud service again. If two client come at the same time then the client which can be served with relatively less time will be served first so that more number of clients get the opportunity to use CellCloud.

10.2 Waiting Time Policy.

It is expected that each bidder will finish its task before the timeline defined by base station. However, there are still possibilities that a bidder fails to finish the task before deadline. Also, Bidders may loose the connection with base stations and therefore, fail to submit the result. Bidders can also refuse to do the task. Regardless of whether any bidder is working or not, mobile cloud always needs to ensure the client that their task will be finished within the deadline. We use the following steps for handling such unwanted situations:

Step-1: We include redundancy during task computation. The level of redundancy is proportional to the level of reliability client demands.

Step-2: Base station only considers the bidders who send their result before the original estimated time $t_r$.

Step-3: If no bidder sends its result before the original estimated time $t_r$, base station will still wait for receiving a result from a bidder until the min time $t_m$ reaches.

Step-4: If all the redundant bidders fail to submit the result within the min time $t_m$, base station finds alternate bidder and give the task to those bidders.

10.3 Price and Delay Policy.

We assume that mobile cloud will make 100% profits on the cost of computation. If the cost associated with computing a task is $C$ then the price for the client will be $2C$. In case of failure to finish the task within the deadline specified by the client, the mobile cloud will make 50% less profit than the profit made for successful completion of the clients’ requirements.
11. Evaluation

In this section, we present our implementation and evaluation results.

11.1 Experimental Setup.

In our experiment, we used three types of mobile devices as our bidders. The first one was a Google nexus 4 with quad-core Krait clocked at 1.5GHz processor along with 2GB of RAM, the second one was a Samsung Galaxy S Plus with a 1.4 GHz Scorpion processor and 512 MB RAM. The third one was a Samsung Galaxy S4 with 1.2 GHz quad-core processor and 2 GB RAM. These mobile devices have internal or external SD card where the task was stored before executing. Similarly, the result of the computation was also stored in the SD card before sending back to the corresponding base station. We solved all the benchmark problems for a input file of size 1MB to 10MB with 1MB increment on these mobile devices and measured the time and power consumption. We calculated the average task completion time and power consumption on each of those mobile devices for a task of 1MB and considered that task as our base task. However, the sorting problem required much longer time on mobile devices due to their limited resources. Therefore, we solved the sorting problem for a input file of size 1KB to 10KB with 1KB increment and measured the time and power consumption for those tasks. We calculated the average task completion time and power consumption on each of those mobile devices for a task of 1KB and considered that task as our base task. We repeat the whole steps for 10 times and measure the task completion time and the power consumption. From the sample run, we found that the task completion time on each type of mobile device is almost proportional to the task size for specific types of task. Hence, we simplified our model by comparing any task size with the base task size. For example, if the ratio between the current task size and base task size is $S$, and the base task completion time is $T_b$, then the current task completion time is $S \cdot T_b$.

The task completion time varies depending on the types of mobile devices. Therefore, instead of considering a fixed base task completion time, we considered a random time within a defined range as base task completion time for each bidder in our simulation. Similarly, we measured the average power drop in each second while solving the three-benchmark problems.

As part of the evaluation, we considered three benchmark problems: word count, intervened index, and sorting an array. We first compared the performance of our CellCloud with the traditional cloud system. We also considered different failure rate of bidder and measured the performance of our CellCloud system.

We built our CCS on a Desktop with 2.8 GHz Core™ Quad CPU along with 7.7 GB RAM. The Android Standard Development Kit (SDK) on Java eclipse platform was used for implementation and performance analysis of CellCloud Central System. Android SDK is proprietary software from Google, which is installed on the open source platform Eclipse for performing computation on android system. In our experimental set up, we considered that CellCloud can have a maximum of 100 base stations, where a maximum of 1000 bidders can participate under each base station. Each bidder can have any of the three types of mobile devices. The rating point of each bidder ranges between 0.1 to 1. Each bidder is assigned a failure value ranging from 0.1 to 1. Moreover, the bidders base task completion time is also initialized by generating a random number within a pre-defined range that we derived from the sample run on actual devices.

For network setup, we assumed that the CellCloud is implemented on LTE network structure [30]. We used the NS-3 LTE module for simulating the distribution and transfer of the task to the bidders and receiving the result from them. We used the standard LTE format of 1460 bytes packet size. In LTE, each of the 3 sectors of a base station can transfer data at a rate of 3.3 Gbit/s [31]. We considered a 60-meter cell size, where the UE nodes (bidders) are distributed and served by an end node (Base Station). We considered 1500 bytes maximum transfer unit (MTU) along with propagation delay or channel delay of 0.010 seconds. Interval between two consecutive packets was considered as 0.006 seconds. For simplicity, we assumed that the bidders are not moving.

We also set up a private backend cloud system by creating micro-instances from Amazon EC2 cloud on pay as you go basis, which costs 1.3 cents/hour for storage and 1 cent/GB for data transfer. For our convenience, we referred both the rented bidders and computers as nodes.

11.2 Experimental Results.

Task Completion Time and Cost. First, we selected a word count task of size 1 MB. Initially, we considered that the private cloud is made of only a single node and calculated the task completion time. In amazon pay as you go service, a PC has to be rented for at least an hour. Amazon charges 1.3 cents/hour for a micro instance. However, in our experiment, we considered two scenarios. In the first scenario, we calculated the cost based on the actual time a PC is rented. Therefore, if a PC is rented for 10 minutes, the cost will be 0.13 cents instead of 1.3 cents. In the second scenario, no matter how long a PC is rented, the cost will be calculated on hourly basis. Thus, the cost of hiring a PC for 90 minutes will be 2.6 cents. In each iteration, we added a PC until the task completion time reached the lowest value.

Next, we ran the same task in our CellCloud and measured the task completion time. We used the power meter to determine the power drop on a node in every ms while computing a task. We found that the power drop is approximately 1000 mw. For each node, we calculated the total power drop based on the time that the node is used.
for computation. From the experimental result of Huang et al. [32], we knew that in LTE technology, the power drop for receiving data at a rate of 1 Mbps is 1340.01 mw in every ms while sending data at the same rate has power drop of 1726.43 mw/ms. We calculated the power drop in every ms for this varying data rate. Based on the time a node is used for sending and receiving data, the total power drop in every node was measured. On an average, the cost of electricity in USA is 8.75-cent/KW hour. Considering the above facts, we measured the cost of hiring bidders for the whole task. From the survey of Brain et al., we found that in best case, the operators can make 200% profit on their total investment if they provide internet services with just 1.9 cents per gigabyte [33]. However, in the worst case, they need to provide Internet services with 8.3 cents per gigabyte to make 200% profits. Considering the amount of data sent and received during the mapping and reducing process, we measured the cost in both the best and worst condition. The resulting cost was computed by adding up this network cost with the bidder hiring cost. We repeated the whole process for 10 times and took the average. We repeated the whole process by increasing the file size by 1 MB in each iteration until the file size is reached 20 MB. Applying the similar procedure, we computed the task completion time for inverted index and sorting problem on both CellCloud and traditional cloud. The time for different task size for each of the three benchmarks is depicted on Figures 4, 5, and 6 respectively. The corresponding cost for task completion is shown in Figures 7, 8, and 9 respectively. We used a logarithmic Y-axis of base 2 for our convenience.

From Figures 4, 5, and 6, we notice that the on an average, the task completion time in CellCloud for word count problem is 15 times more than traditional Cloud. For inverted index problem, CellCloud takes much higher around 30 times more than traditional cloud. The sorting problem takes approximately 20 times more in CellCloud than traditional cloud. Moreover, according to the first scenario of traditional cloud, Figure 7, 8, and 9 show that the cost is approximately half compared to even the best case scenario of CellCloud for word count and inverted index problem. For sorting problem, traditional cloud is 10 times less expensive than the best case scenario of CellCloud.
On the other hand, Figure 7 and 8, and 9 depict that the cost for word count, inverted index, and sorting problem, according to the second scenario of traditional cloud, is approximately 270, 150, and 90 times more than the cost of CellCloud in the worst case. However, if we observe the pricing policy of the major cloud providers, then the second scenario is more practical. Therefore, our CellCloud system might not ensure lowest time but it is far less expensive than traditional cloud. Let us consider the ultimate gain as the ratio of gain in cost to gain in time. The ultimate gain with CellCloud is 5 times more compared to traditional cloud for both inverted index and sorting problem and that of word count problem is approximately 15 times.

**Computations with Node Failures.** In our next experiment, we assumed that the bidder can fail anytime during task execution. The probability of failure for each bidder can varies from 0% to 100%. During our simulation, we assigned a failure value for each node under a base station. We also fixed a threshold for failure and assumed that the nodes whose assigned failure value is below the threshold will fail to perform the task on time. Each base station has the information of possible task completion time for each bidder. If the base station does not receive result within that time it will find another bidder and send the task to that bidder. The bidder that fails will not receive money for the task. The variation of task completion time for different failure rate is depicted in Figures 10, 11, and 12.
From Figures 10, 11, and 12, we notice a very little increase in task completion time with increasing failure rate. For word count and inverted index problem, the rate of increment in task completion time is 1 and 2 percent respectively from failure value 0 to failure value 0.5. The increasing rate is small because we are distributing the task among a large number of bidders. Hence, each bidder receives a very small portion of the task. Therefore, failure of a bidder does not affect much on overall task completion time. However, for sorting problem, we identified approximately 200% increase in time from failure rate 0 to 0.5. Each node in sorting problem takes much larger time even for a very small portion of the task. Moreover, the input and output size is the same in sorting problem. Thus, the base station has to wait for much longer time before transmitting the task again to a different node. However, considering failure rate of 0.5, the increase in time is still much smaller as compared to the overall task completion time.
On the other hand, if we consider Figures 13, 14, and 15, we notice that the cost increases very slightly with increasing rate of failure. From failure rate 0 to 0.5, in best case, the cost increases approximately 18%, 10%, and 2% for word count, inverted index, and sorting problem respectively. The increase rate is approximately 24%, 12%, and 7% for word count, inverted index, and sorting problem respectively. In our CellCloud architecture, the bidders will not receive any money if they fail to perform the task on time. Hence, when a bidder fails we just need to consider the network cost for sending the task to the bidder.

12. FUTURE OF CELLCLOUD

12.1 SPEECH RECOGNITION.

In several scenarios we might need to recognize the speech [34]. For example if we move to a place where people speak different languages then we might feel difficulty to communicate with them. In such case we might want our mobile device to act as a translator. In addition, sometimes we meet a person after long time and forget the name of that person. Using speech recognizer, we can match the voice of the person with our previously recorded voice and identify that person. However, it requires a substantial computational power and a battery power for running the tool. Instead, our mobile device can record the voice and transmit the speech signal to the cloud. The cloud returns the recognition result after processing.

12.2 IMAGE ANALYSIS.

Image analysis might require when we want to know the details of a visible image. For example, often we visit a museum where we see different types of arts. We are always enthusiasm to gather information about those arts. We might not want to run a complex image analysis tool on our limited power mobile devices. The picture of those arts can be sent to the cloud for analyzing. The cloud sends the detail information of the art back to the user after the processing.

12.3 REAL TIME INFORMATION GATHERING.

People might be interested to gather information quickly about a specific area with minimal amount of cost. For example, we might want to go to a party and want to know the suitable dress for the party. Moreover, we might be also interested to know the number of people so far in that party. We can easily get that information using CellCloud. The CCS can ask the bidders on that party to collect and process the required information.

12.4 FACE RECOGNITION.

Face recognition is a widely used for applications such as surveillance, airport security, law enforcement, and border patrol [35]. First, an input image is taken for analysis by the face recognition algorithm. From the result of the analysis, we extract several information such as, size, shape, and position etc. of various facial features. These extracted features are compared with the images existed on an already existing facial database for a match. However, this complex face recognition algorithm is hard to run on a user limited mobile resource. The user can acquire the image and send it to the cloud for processing.

13. DISCUSSION

From our simulation result, we see that the task completion time in CellCloud is much higher than traditional cloud. This might lead to a question why will we use CellCloud? If clients want immediate result of a problem, then CellCloud will not be the right choice for them. However, sometimes clients want service where waiting time does not matter much. In fact, clients are more concern about the cost and complexity. In such cases, CellCloud could be a better option for them than traditional cloud. They are already connected to a mobile network and therefore, requesting for cloud service from the same mobile network provider is much easier than login to traditional cloud provider and ask for service. Moreover, they are getting the service with much cheaper rate than using the traditional cloud.

If we further analyze the task completion time of our CellCloud then we can see that among the total task completion time 80% time is required for sending the task to the bidders and receive the result back to CCS. Even though the data transfer rate in 4G networks is very high, it is still much smaller than the data transfer rate inside the traditional cloud such as Amazon, Google etc. The task completion time in CellCloud and Traditional Cloud will be approximately same if we do not consider the task transfer time or if we consider the similar data rate in both cases. With the advancement of technology, the data transfer rate in mobile network is increasing every couple of years. Researchers have already started working of the improvement of mobile network that will soon converge to the 5G technology [36]. According to the specification defined in the blueprint of 5G technology, the data transfer rate will be much higher around 10Gb/s as compared to 3Gb/s 4G technology. Moreover, 5G technology will have very small amount of latency and response time, and will consume much smaller power than the currently using 4G technology. If we deploy our CellCloud according to the specification defined in 5G technology, the task completion time will be reduced to approximately 1/3 of the current task completion time. In that case, CellCloud can also serve the task, which
requires immediate result with significantly smaller amount of cost than traditional cloud.

14. CONCLUSION AND FUTURE WORK

The current trend of rapidly growing number of smart phone users along with the tendency of switching to new phones in every couple of years is creating a big pile of unused mobile devices. To the best of our knowledge, CellCloud is the first protocol that attempts to reshape the definition of mobile cloud by incorporating these unused but available resources. Along with the detailed architecture of such a system, we have developed a cost model to analyze the benefits from both mobile owners’ and provider’s point of view. CellCloud features, such as, facilitating different pricing options for different deadlines and level of reliability, providing money to the mobile owners for sharing their unused resources, and lessening operational cost compared to traditional cloud for the cloud provider ensure that such a model can create a win-win situation for all the parties. Currently, we are trying to build a model by which, the CellCloud provider can provide an estimation of task completion time and the associated cost before accepting a task from the client. For doing this, we are planning to train our system with sample tasks of various sizes. Based on the result obtained from the training, we will develop a map between task size and the required amount of resource. We also plan to deploy the architecture in small scale in real life mobile infrastructure to analyze the feasibility of the model.

15. REFERENCES

Authors

Shahid Al Noor is a PhD student and a graduate teaching assistant at Computer and Information Science at UAB. His research interest is in Mobile Cloud Computing, Mobile Device Security, and Cloud Security. He received his B.Sc. and M.Sc. in Information and Communication Engineering from University of Rajshahi, Bangladesh in 2006 and 2007 respectively.

Dr. Ragib Hasan is a tenure-track Assistant Professor at the Department of Computer and Information Sciences at the University of Alabama at Birmingham. Prior to joining UAB, He received his Ph.D. and M.S. in Computer Science from the University of Illinois at Urbana Champaign in October, 2009, and December, 2005, respectively, and was an NSF/CRA Computing Innovation Fellow post-doc at the Department of Computer Science, Johns Hopkins University. Hasan has received multiple awards in his career, including the 2014 NSF CAREER Award, 2013 Google RISE Award, and 2009 NSF Computing Innovation Fellowship. His research interest lies in the area of efficient and secure systems including secure clouds, mobile devices, and financial computing systems.

Md Munirul Haque is currently working as Research Scientist in Regenstrief Center for Healthcare Engineering, Purdue University. He received his Ph.D. from Marquette University, USA in 2013 before joining SECURe and Trustworthy Computing Lab (SECRETLab) at the University of Alabama at Birmingham as post-doctoral fellow. His academic studies and training have made him an expert in the area of mobile based real life applications. He is the recipient of the Ross Fellowship Award for outstanding Ph.D. student and several best paper (COMPSAC 2007, CHI 2012, RACS 2013) and poster awards. His field of interest encompasses security and privacy in pervasive health, m-Health, mobile cloud, and HCI (Human Computer Interaction).
END-TO-END BIG DATA PROCESSING PROTECTION IN CLOUD ENVIRONMENT USING BLACK BOXES - AN FPGA APPROACH

Lei Xu1, Khoa Dang Pham1, Hanyee Kim2, Weidong Shi1, and Taeweon Suh2
1 University of Houston, Houston, TX, USA 2 Korea University, Seoul, Korea
{lxx13, pdkhoa, wshi3}@central.uh.edu, {hanyeemy, suhtw}@korea.ac.kr

Abstract
Privacy is one of the critical concerns that hinders the adoption of public cloud. For simple application, like storage, encryption can be used to protect user's data. But for outsourced data processing, i.e., big data processing with MapReduce framework, there is no satisfying solution. Users have to trust the cloud service providers that they will not leak users' data. We propose adding black boxes to the public cloud for critical computation, which are tamper resistant to most adversaries. Specifically, FPGAs are deployed in the public cloud environment as black boxes for privacy preserving computation, and proxy re-encryption is used to support dynamic job scheduling on different FPGAs. In FPGA cloud, cloud is not necessarily fully trusted, and during outsourced computation, user's data is protected by a data encryption key only accessible by trusted FPGA devices. As an important application of cloud computing, we apply FPGA cloud to the popular MapReduce programming model and extend the FPGA based MapReduce pipeline with privacy protection capabilities. Finally, we conduct experiments and evaluation for k-NN protected by a data encryption key only accessible by trusted FPGA devices. As an important application of cloud computing, big data attracts attentions from both the industrial and academic communities. Researchers also developed more efficient techniques for special applications such as database operations. There has been work on supporting search on cipher-texts (Boneh, Crescenzo, Ostrovsky & Persiano, 2004; Bellare, Boldyreva & O'Neil, 2007; Song, Wagner & Perrig, 2000), order preserving encryption (OPE) scheme (Boldyreva, Chenette, Lee & O’Neil, 2009; Boldyreva, Chenette & O’Neil, 2011). A cloud database system focusing on managing numerical data was developed based on the OPE scheme (Curino et al., 2011). Hacigümüş, Iyer, Li and Mehrrota (2002) proposed a more general bucketization method to process SQL queries for outsourced database and protect the privacy of the database. Liu, Kantarcioğlu, and Thuraisingham (2009) designed a privacy preserving decision tree mining scheme with perturbed data. All these techniques are not fully satisfied because they either support limited application scenarios or suffer from high computation/storage cost. Furthermore, there has been little work in supporting strong privacy preservation for cloud based parallel data analytics such as enabling strong privacy protection for the popular MapReduce programming model. Field programmable gate arrays (FPGAs) receive much attention in recent years for data analytical applications.

1. INTRODUCTION

Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction (Mell & Grance, 2009). These features of cloud computing make it attractive for many applications, e.g., database system, customer relationship management, and call center. Especially, as an emerging technology, big data attracts attentions from both the industrial and academic communities.

Security has been considered as one of the critical concerns that hinder the wide adoption of public cloud, especially for the enterprises and the government market. In reality, it is very unlikely that the companies like Amazon, Microsoft, and Google who run the cloud service will try to access the users’ data without permission. The main threats come from malicious users and administrators. Due to the virtualization technology used in cloud computing, malicious users may cross the boundary to access others’ data (Ristenpart, Tromer, Shacham & Savage, 2009). Administrators usually have higher privileges and they may abuse this ability to learn users’ data without permission (like the Snowden case (Toxen, 2014)). For simple cloud service like data storage, the user can protect the data from these threats with encryption. More complex techniques are designed to support applications like sharing and efficient retrieval (Thuraisingham, Khadilkar, Gupta, Kantarcioğlu & Khan, 2010). Applications that are dependent on both the storage and computation capability of cloud need a more complex solution for the security concern. FHE (fully homomorphic encryption, (Brakerski & Vaikuntanathan, 2011; Gentry, 2009; van Dijk, Gentry, Halevi & Vaikuntanathan, 2010)) is a powerful tool for privacy preserving computation outsourcing. Because FHE supports operations on cipher-texts, a user only needs to send encrypted data to the cloud for computation. The drawback of FHE based solution is that the existing FHE schemes are very inefficient and it is not practical to use them for meaningful computation tasks such as signal processing and data analysis. Researchers also developed more efficient techniques for special applications such as database operations.
because its capability to support customized parallel processing. FPGAs are considered to be powerful computation devices and suited as accelerators for big data analytics. It is common for many applications to employ FPGA based devices to accelerate their performance (e.g., (Court, Gu & Herbordt, 2004; Ronan, Óigeartaigh, Murphy, Scott & Kerins, 2006; Woods & VanCourt, 2008)). FPGAs are also deployed in the cloud environment as accelerators for large scale data processing such as MapReduce (Shan, Wang, Yan, Wang, Xu & Yang, 2010). Another line of FPGA research focuses on protection of the FPGA bitstream (compiled FPGA netlist). In this case, the main concern is to protect the intellectual property of FPGA bitstream developers. Many commercial FPGA devices support bitstream encryption for this purpose. However, to the best of our knowledge, no one has tried to leverage such mechanisms to protect the privacy of the data that is processed by the FPGAs in the public cloud setting.

Since FPGAs are powerful computation devices with certain unique security features that are not available on today’s commercial general purpose processors, our motivation is to enable practical privacy preserving solutions in general cloud based data analytics by leveraging the unique security properties of FPGAs. To achieve these goals, we develop a privacy preserving FPGA cloud, a new framework of cloud computation with FPGAs as part of the computation infrastructure provided by a cloud service provider. By taking advantage of the unique security features of FPGAs, one can enable such computing scenario that when privacy sensitive data is processed on the cloud side using FPGAs, the user’s data is not disclosed to the service provider who hosts the FPGA cloud infrastructure. As FPGA is a general purpose computation hardware, it enables the user to do many types of computation (e.g., signal processing, data mining, pattern matching, data analytics, and database operations). In a public cloud environment, a user typically doesn’t know or care which machine will be assigned to process his/her data. In order to enable the cloud to manage FPGA resources and dynamically allocate arbitrary number of FPGA devices to process user’s data, we develop a cryptographic solution based on proxy re-encryption, which supports cipher-text re-encryption. The solution protects cloud user’s data key without disclosing it to the proxy. We also discuss how to apply FPGA cloud to MapReduce programming model, which is widely used in the cloud environment for big data processing. In summary, our contributions in this paper include:

- We propose a framework of privacy preserving FPGA cloud, which enables privacy protection in outsourced big data analytics (e.g., MapReduce) by leveraging the existing hardware supports for bitstream IP protection.
- We apply FPGA cloud to MapReduce and extend the FPGA based MapReduce pipeline with privacy protection of user’s data against untrusted service providers.
- We design a proxy re-encryption approach and MapReduce key-value transformation that allow cloud service providers to parallelize and manage MapReduce tasks without disclosing user’s private data.
- We conduct security and performance analysis, and the results confirm that the FPGA cloud is practical.

The rest of the paper is organized as follows: In Section 2 we give a short review of the techniques used in the proposed solution. Section 3 discusses the security assumptions and threat models. Section 4 provides a detailed description of our solution, followed by security evaluation and enhancement in Section 5. In Section 6, we describe applying FPGA cloud for MapReduce jobs and give a concrete example of k-NN clustering in Section 7. Section 8 discusses related work and Section 9 concludes the paper.

2. BACKGROUND

In this section, we give a brief review of FPGA security features and proxy re-encryption schemes.

2.1 FPGA AND BITSTREAM ENCRYPTION.

FPGAs combine some advantages of software (fast development, low non-recurring engineering costs) with those of hardware (performance, relative power efficiency). These advantages have made FPGAs an important fixture, especially for those applications that are computation intensive. Particularly, a growing body of literature has focused on applying FPGAs to accelerate MapReduce based data analytics (e.g., (Shan et al., 2010)).

![Figure 1. Bitstream encryption for FPGA. The bitstream is protected by AES and HMAC. The keys are also stored in](image-url)
have a bitstream encrypted and decrypted each time when it is loaded into the FPGA (McNeil, 2012; Microsemi, 2013). Many commercial FPGAs support such bitstream encryption/decryption. Specifically, each FPGA is assigned a symmetric key $k$ and only the legitimate user knows this information. The legitimate user encrypts the bitstream with $k$ and can upload it to FPGA. The encrypted bitstream is often stored in a flash memory that comes with an FPGA based system. After that, the FPGA can decrypt and run the function contained in the bitstream. Figure 1 depicts the bitstream encryption mechanism. Because the FPGA is tamper resistant, malicious attackers cannot access the bitstream. The symmetric key is protected and cannot be accessed by end users of the FPGA system (e.g., cloud service provider). However, this bitstream IP protection feature cannot be leveraged directly for privacy protection in the cloud. It is unrealistic to distribute the symmetric keys of FPGAs to many different cloud customers securely. To make the matter more complex, users’ computation tasks may be dynamically scheduled to different physical FPGA devices in the cloud environment.

2.2 Proxy Re-Encryption.

Proxy re-encryption plays an important role in our framework to enable dynamic scheduling. Proxy re-encryption was firstly proposed by Blaze, Bleumer and Strauss (1998), which gave a positive answer to the open problem whether it is possible to convert a cipher-text encrypted using one key to cipher-text encrypted using another key without decryption.

There are usually three parties for a proxy re-encryption scheme, user $A$, user $B$, and the proxy. User $A$ and $B$ possess a pair of keys $(p_k_A, s_k_A)$ and $(p_k_B, s_k_B)$ respectively. With the key information, a re-encryption key $r_{k_{AB}}$ can be calculated. The re-encryption key is usually sent to the proxy. For any cipher-text $c$ that is encrypted with user $A$’s public key $p_k_A$, the proxy can use $r_{k_{AB}}$ to re-encrypt $c$ to get another cipher-text $c’$. Then user $B$ can use its own private key $s_k_B$ to decrypt the cipher-text $c’$. During the re-encryption process, the proxy learns nothing about the corresponding plain-text.

Proxy re-encryption schemes with different properties are proposed since the invention of the first scheme (Canetti & Hohenberger, 2007; Libert & Vergnaud, 2008; Tang, 2008). For our purpose, we require that the proxy re-encryption scheme should have the following properties (Ateniese, Fu, Green & Hohenberger, 2006): 1. Unidirectional.Delegation from user $A$ to user $B$ does not allow re-encryption from $B$ to $A$; 2. Non-interactive. Re-encryption key from user $A$ to $B$ can be generated by $A$ using $B$’s public key, no trusted third party or interaction is required; 3. Collusion safe. Even if a proxy holds a re-encryption key $r_{k_{A-B}}$ and colludes with user $B$, they cannot recover user $A$’s private key; 4. Non-transitive. The proxy alone, cannot re-delegate decryption rights, e.g., from re-encryption key $r_{k_{A-B}}$ and $r_{k_{B-C}}$, the proxy cannot produce $r_{k_{A-C}}$. We will explain the importance of these properties for FPGA cloud in Section 5.

3. System Overview and Threat Model

In this section, we give an overview of FPGA cloud and discuss the security assumptions for different parties involved in the framework.

![Figure 2. Overall framework of end-to-end protection of cloud based big data processing. FPGA vendor collaborates with the cloud to initialize the FPGAs, which are managed by the cloud. The users send encrypted data to cloud and the cloud processes the data using FPGAs. In the processing procedure, the proxy is set up for cipher-texts conversion. At the final step, the cloud returns the encrypted results to the users.](image)

3.1 System Overview.

As shown in Figure 2, four parties are involved in FPGA cloud: cloud user, FPGA vendor, cloud service provider, and proxy.

1) Cloud user: A cloud user possesses a public/private key pair. When the user outsources the computation work to the cloud, the user first encrypts his/her data with a symmetric encryption scheme such as AES using a data key. The data key is further encrypted with user’s public key. The user coordinates with the proxy for computing re-encrypted data keys;

2) FPGA vendor: FPGA vendor manufactures FPGA based computing devices that are ready to be deployed in the cloud. The vendor is also responsible for embedding keys into the FPGAs. The keys can be embedded in the FPGA bitstream that is encrypted and protected with the existing FPGA IP protection mechanisms. As mentioned in Section 2, each FPGA has two types of keys, one is a public/private key pair that is embedded into the bitstream, the other is a symmetric key, which is used to encrypt the bitstream. The symmetric key is usually stored in tamper-proof storage of the FPGA so that no one outside can access this key. Besides releasing the hardware to the cloud
service provider, a vendor does not need to interact with any other parties to fulfil its designated role;

3) Cloud service provider: A cloud service provider possesses and manages a pool of FPGA devices besides usual computing hardware that one can find in cloud such as CPUs, memories, and disks. For privacy sensitive tasks that are designed to run on the FPGAs, the cloud provider provisions user’s data (protected with a secret data key inaccessible by the service provider) to the FPGA device memory and instructs the FPGA to process the user data. After finishing the computation, the FPGA returns encrypted results to the cloud;

4) Proxy: Proxy is responsible for cipher-texts conversions and re-encryption key management. It will interact with both the cloud service provider and the cloud user.

3.2 THREAT MODEL.

We assume that cloud user is fully trusted and the user will be diligent in having all the key materials properly protected for his/her own interests.

The cloud service provider is semi-trusted, i.e., it will work fairly by following pre-defined protocols and polices but may try to learn the user’s data from what it can access. The cloud service provider cannot be fully trusted due to reasons from two perspective: (i) Risks from malicious users. It is common that multiple users reside in the same cloud infrastructure, or even the same physical machine due to virtualization. Malicious users may cross the boundary to access others’ data (Ristenpart et al., 2009); (ii) Risks from malicious administrators. Administrators usually have higher privileges and malicious administrators may abuse these privileges to learn users’ data (Kandias, Virvilis, & Gritzalis, 2013; Toxen, 2014).

As proxy only supports limited functions and has simple storage structure, it is easy to be well protected. So we assume the proxy is trusted.

We further assume that the vendor is trusted. The bitstream is assumed to be free from backdoors or Trojans. The vendor will not collude with the cloud service provider or the proxy to compromise user’s data privacy. It is worth pointing out that without colluding with the cloud service provider, the vendor itself generally has no access to the user’s encrypted data or encrypted data key. Thus, the vendor cannot compromise the privacy of user’s data by itself.

3.3 GOALS OF FPGA CLOUD.

The goals of FPGA cloud can be summarized as follows:

1) Data security. User’s data sending to the cloud for processing are well protected, i.e., no one except the user has access to both the original data and the final processed results;

2) Reasonable cost and generality. FPGA cloud should be able to support different applications like big data processing with real scale.

4. PRIVACY PRESERVING FPGA CLOUD

In this section we provide detailed design of FPGA cloud.

4.1 PROXY RE-ENCRYPTION SCHEME USED FOR FPGA CLOUD.

For cloud computing one key feature is dynamic job scheduling, and proxy re-encryption plays an important role for this capability. In this paper, we use the proxy re-encryption scheme proposed in (Ateniese et al., 2006). This scheme satisfies our requirements such as unidirectional, non-interactive, conclusion safe, and non-transitive. We give a short review of this scheme. Suppose $G_1, G_2$ are two groups of prime order $q$ with a bilinear pairing map $e : G_1 \times G_1 \rightarrow G_2$, $g \in G_1$ is a random generator and let $Z = e(g, g)$. The scheme works as follows:

- Key Generation. User A’s key pair is of the form $p_k A = g^a, s_k A = a$, where $a \in Z_q$;

- Re-encryption Key Generation. If user A wants to delegate the decryption right to user B, he computes the re-encryption key $r k_{A\rightarrow B} = g^{b/a}$.

- Encryption. To encrypt a message $m \in G_2$ with public key $p_kA$, a user first chooses a random number $k \in Z_q^*$ and computes the cipher-text as $c_A = (g^{ak}, mZ^k)$;

- Re-encryption. For cipher-text $c_A = (g^{ak}, mZ^k)$ , the proxy with re-encryption key $r k_{A\rightarrow B} = g^{b/a}$ computes $e(g^{ak}, g^{b/a}) = Z^{bk}$ and publishes the new cipher-text $c_B = (Z^{bk}, mZ^k)$;

- Decryption. Given cipher-text $c_A = (g^{ak}, mZ^k)$, user A who possesses private key $s_k A = a$ can compute $e(g^{ak}, g^{1/a}) = Z^k$ and then recover the plain-text $mZ^k/Z^k = m$ . For FPGA cloud, the Decryption algorithm is not used;

- Re-encryption Decryption. Given re-encrypted cipher-text $c_B = (Z^{bk}, mZ^k)$ , user B who possesses private key $s_k B = b$ first computes $(Z^{bk})^{1/b} = Z^k$ and then recovers the plain-text $m$.

4.2 BITSTREAM EXTENSION.

Figure 3. The extended bitstream structure. To support privacy preserving computation, extra components are added into the bitstream, e.g., the AES encryption/decryption module for input decryption and output encryption, public/private key and corresponding proxy re-encryption module for data key processing.
In order to utilize FPGA for privacy preserving computation in public cloud environment, we need to add some new features into the bitstream. Figure 3 depicts the extended bitstream structure. Specifically, a public/private key pair related to proxy re-encryption scheme and corresponding functions are embedded into the bitstream.

4.3 Outsourcing Computation to FPGA Cloud.

In this section, we provide detailed description of using FPGA cloud to protect privacy of user’s data while offering processing capacity. We assign an identity to each function type so that cloud can schedule tasks to the proper FPGAs. The FPGA cloud consists of five protocols.

System Initialization. As mentioned in Section 4.2, a pair of keys is embedded into the bitstream. For security consideration, we require that all the key pairs be unique. In other words, each FPGA $i$ has its own key pair, denoted as $(pk_i^f, sk_f^i)$. Each user $j$ is also given a key pair, denoted as $(pk_j^U, sk_U^j)$. All the public keys are published. The proxy maintains a database of proxy keys for the FPGAs and users. Table 1 gives an example of the proxy key database. This database stores re-encryption keys that are used for converting user encrypted data to proper form for the designated FPGA to process. The database is managed by the proxy in a lazy manner, i.e., an entry of the database is computed and saved only when the value saved in this entry is needed to perform data processing. The disadvantage of this strategy is that user has to remain online in order to cooperate with the proxy after submitting his/her task.

Data Uploading. Before uploading any data to the cloud, user $j$ randomly generates a symmetric key $dek$ to encrypt the data that needs to be processed in the FPGA cloud. $dek$ is further encrypted with user $j$’s public key $pk_j^U$. Denote the encrypted data as $c_{data}$ and the cipher-text of $dek$ as $c_{dek}$. User $j$ sends $c_{data}$, $c_{dek}$, his/her identity $j$, and the function type identity. User also records the relationship between the symmetric key and the task.

Task Scheduling. One advantage of cloud computation is that the cloud service provider possesses a resource pool and can allocate resources to the users on demand. After receiving the user’s request, the cloud has to schedule certain FPGA(s) for the task. It first checks the function type to decide which group of FPGAs can be used for this task, then schedule the job to these FPGAs. The service provider may consider different factors when choosing the FPGA, e.g., the physical location, load balancing, etc. For each chosen FPGA, the cloud service provider asks the proxy for re-encryption of the user’s data for this FPGA. The proxy will query the database for the FPGAs and users. If the re-encryption key does not exist, then the proxy re-encrypts $c_{dek}$ to $c'_{dek}$ and sends $c'_{dek}$ to the cloud provider. Then, the FPGA decrypts it with the private key embedded into its bitstream. If the re-encryption key does not exist, then the proxy interacts with the user to generate the re-encryption key, saves the key in the database, and performs the above operations. If multiple FPGAs are selected, then the cloud performs the same work for each of them. After receiving response from the proxy, the cloud service provider sends $c_{data}$ and $c'_{dek}$ to the FPGA for data processing.

Data Processing. With the decryption function and key embedded into the bitstream, an FPGA can decrypt $c_{dek}$ to get $dek$. Then it can further decrypt $c_{data}$ and does the required processing procedure. Note that all the decryptions are carried out inside the FPGAs and it is not possible for the cloud service provider to find out and disclose user’s data. Thus, user’s data privacy cannot be compromised. After finishing the data processing, the FPGAs encrypt the results with $dek$ and then the cloud will return them to the user.

Results Retrieval. Results retrieval is very simple. As the user knows the symmetric key $dek$, he/she can decrypt the results to get the plain-text of the result.

5. Security Evaluation and Enhancement of FPGA Cloud

In this section we evaluate the security property of FPGA cloud and discuss potential risks.

5.1 Security of FPGA Cloud

![Security of user’s data](image)

Figure 4. Security of user’s data. The user uses his/her public key to encrypt $dek$ before sending to the cloud. The proxy only has access to the re-encryption key to help on converting $c_{dek}$ to $c'_{dek}$, but cannot learn any information about $dek$. The only place where $dek$ exists in plain-text form is inside the FPGA. The tamper resistant property of FPGA assures the security of $dek$. 
According to the protocol given in Section 4, except inside the tamper resistant FPGA, user’s original inputs and final results are always encrypted using a symmetric key $dek$ and $dek$ is protected by either the user’s public key or the FPGA’s public key. The proxy will converting the cipher-text of $dek$, and the security property of the proxy re-encryption scheme assures the proxy cannot learn anything about $dek$. So security of the user’s data only depends on the security of the private keys of the user/FPGA.

The user will not leak his/her private key to any parties. The re-encryption key generation involves user’s private key but this procedure is finished on the user side. The FPGA private key is protected by the bitstream encryption key and depends on the security of the bitstream encryption key. Figure 4 summarizes the security relationship between these components.

5.2 Potential Risk of FPGA Cloud and Enhancement

Initial key generation. If the symmetric keys used for bitstream protection leak to adversaries, the whole solution fails. Our solution makes an assumption that the FPGA vendor is responsible for embedding keys into FPGAs, and the FPGA vendor becomes a potential vulnerability. To overcome this challenge, Xu and Shi (2014) proposed a method to enable the FPGA vendor and the cloud service provider work together to generate the keys, so the bitstream protection keys are at risk only when both parties are compromised.

Side channel attacks. One key feature of the proposed solution is to utilize FPGAs as black boxes in the cloud. Every security solution utilizing hardware black boxes faces the threat of side channel attacks. By collecting side channel information like energy consumption (Messerges, Dabbish & Sloan, 1999, 2002), radiation (Gandolfi, Mourtel & Olivier, 2001), time (Renauld, Staendaert & Veyerat-Charvillon, 2009), or even sound information (Genkin, Shamir & Tromer, 2013), the attacker can recover the secret information stored inside the hardware. Specified side channel attacks target at FPGAs are also studied (e.g., Moradi, Barenghi, Kasper & Paar, 2011; Moradi, Oswald, Paar & Swierczynski, 2013), but these attacks generally have special and rigorous requirements for the attacking environment. For example, Moradi et. Al (2011) proposed power analysis attack against Xilinx FPGA, which requires connecting the FPGA board to a special communication module and use digital oscilloscope to collect the side channel information.

In the cloud environment, users have no physical access to the data center and devices deployed there. So it is hard for malicious users to launch side channel attacks. Even though administrators usually have more control over the cloud computing system than general users, they also have very limited physical access to the infrastructure (Steiner, 2012). Thus, it is improbable for administrators to attach special devices to collect side channel information. The effectiveness of side channel attacks is heavily dependent on the hardware/software implementation of a system. Careful design and implementation of the FPGA board can make side channel attacks harder (Bogdanov, Moradi & Yalcin, 2012; Canetti & Hohenberger, 2007; Güneysu & Moradi, 2011; Poschmann et al., 2011).

Collusion between different parties. If one or more parties do not act according to its protocol, the privacy of user’s data may be at risk. One potential risk is collusion between the cloud service provider and the proxy. Suppose that a user uploads data to the cloud where the data is protected with a data encryption key $dek$. Instead of using the public key of an FPGA, the cloud service provider generates a public/private key pair $(pk_{fake},sk_{fake})$ with the same public parameters of the deployed proxy re-encryption scheme. Then the proxy interacts with the user and asks for a re-encryption key $rk_{fake}$. If the user responds to this request and generates the re-encryption key for the proxy, the proxy can use this key to convert the cipher-text of $dek$ to another cipher-text that can be decrypted using $sk_{fake}$. Then the cloud service provider can learn the plain-text of $dek$ and use this key to decrypt user’s data. This risk can be mitigated by introducing PKI, i.e., each public key is associated with a certificate that is signed by a CA. When a user receives a request of re-encryption key generation, he/she can verify whether this public key is valid or not, and only respond to these valid public keys.

6. FPGA Cloud for K-NN Clustering with MapReduce

In this section, we describe using the FPGA cloud for big data processing with MapReduce framework.

6.1 Short Review of MapReduce Framework

MapReduce is a powerful tool for big data processing (Dean & Ghemawat, 2008), and is popular for cloud based data analytics. As the name implies, MapReduce contains two main functions: map function and reduce function. The MapReduce process can be broken down into a pipeline as shown in Figure 5 (Zhang, Cherkasova & Loo, 2013).

![Figure 5](image_url)

**Figure 5.** Original MapReduce processing pipeline. The results of mappers go through three steps and stored as intermediate results. Then the intermediate results are shuffled to different reducers.

Map task consists of five phases:

1) **Read.** Reading a data block (e.g., 64 MB) from distributed file system (e.g., Hadoop distributed file system);
2) **Map.** Applying the map function to each record in the input file and generating the map-output data (intermediate data);
3) **Collect.** Buffering the map phase outputs into memory;
4) **Spill.** Sorting and partitioning the intermediate data across different reduce tasks;
5) **Merge.** Merging different spill files into a single spill file for each reduce task.

Reduce task has the following three phases:
1) **Shuffle.** Transferring the intermediate data from map tasks to reduce tasks and merge-sorting them together;
2) **Reduce.** Applying the reduce function on the input key and all the values corresponding to it to produce the final output data;
3) **Write.** Writing the reduce output to the distributed file system.

### 6.2 MAPREDUCE TASK IN FPGA CLOUD.

A MapReduce task can be divided into three parts in FPGA cloud framework, as Figure 6 depicted.

- FPGAs for the map function are responsible for **read** and **map**;
- FPGAs for the reduce function are responsible for **reduce** and **write**;
- General purpose processors (CPUs) are responsible for computations related to other scheduling jobs, e.g., collect, spill, merge, and shuffle.

Compared with the standard MapReduce process without privacy preserving support, the scheduling jobs carried out by the cloud service provider using CPUs are the same, and the differences are all related to the FPGA jobs. Figure 7 summarizes the whole processing procedure. In the following description, key may refer to either part of the (key, value) pair in MapReduce framework or an encryption key.

1) **Input data.** The input data is encrypted by a user using a randomly generated data encryption key \( d_{e_k} \), then \( d_{e_k} \) is encrypted using the user’s public key, and the cipher-text \( c_{d_{e_k}} \) is stored together with the encrypted data. Here, the data encryption algorithm should be compatible with the data division strategy, i.e., when the cipher-text of the data is divided into blocks, each block can be decrypted properly independently. For example, when AES is used for encryption, the data can be encrypted block by block, other than encrypted as a whole;

2) **Read.** When an FPGA for running the map function reads a block, it also reads the cipher-text of the related data encryption key. Following the protocol mentioned in Section 4, the FPGA interacts with the proxy to get a re-encrypted cipher-text of the data encryption key. Then it can decrypt to get \( d_{e_k} \) with its own private key. After that, the data block can be decrypted;

3) **Map.** After the decrypted data block is processed, the FPGA for the map function gets a set of pairs (key, value), which are the same as the standard MapReduce process. However, before sending these pairs as output, the FPGA for the map function encrypts the original pair (key, value) with \( d_{e_k} \), denoted as \( c_{cipher_{pair}} \), and applies a transformation to key to get another index hashcode. The cloud service provider uses hashcodes for scheduling as replacement of the original reduce key. We will discuss details of this transformation in Section 6.3;
4) **Reduce.** After receiving a pair \((\text{hashcode}, \text{cipher}_{\text{pair}})\), an FPGA for running the reduce function interacts with the proxy and asks for a re-encrypted cipher-text of \(dek\). Then the reducer FPGA decrypts to get \(dek\), and uses \(dek\) to decrypt \(\text{cipher}_{\text{pair}}\). After that, the reduce function is carried out and the final result is calculated;

5) **Write.** Before writing the result to the distributed file system, the reducer FPGA encrypts the result with \(dek\). User who uploads the task knows \(dek\) so the user can decrypt and extract the final result.

6.3 **Statistic Information Leakage and Counter Measures.**

For MapReduce framework with FPGA cloud, the mapper FPGA cannot encrypt the whole \((key, value)\) pairs directly because the cloud service provider needs this information of keys to schedule these pairs to reducer FPGAs. The basic principle of MapReduce is that output pairs of mappers with the same key should be dispatched to the same reducer FPGA. Encryption of the value part of each pair is not enough, as key usually reflects some property of value, and attacker may gain useful information even if all values are encrypted. Take the application of word count as an example, the output of mapper is in the form \((key, 1)\), where key is the word appeared in the data being processed. It is easy to see that encrypting the value part does not help us to protect the data being processed. Hash function can be applied to key to hide the original word but it does not help for the protection of the distribution. Attackers who can access the intermediate result may easily obtain the word frequency information of the input data. Another potential risk is that if the domain of value is small and key is generated from value without adding any secret, one may use the dictionary attack to recover plain-text of value with the key.

In order to prevent such risks, we propose using the truncated HMAC to generate a hashcode from the original key, and \(dek\) is used as secret key to HMAC. As a secret key is involved in the process of calculating hashcode, dictionary attack is prevented. To verify the effectiveness of our method against frequency attack, we apply truncated HMAC to a data set that contains more than 5 million url items (Leskoveca, Langb, Dasgupta & Mahoneya, 2009) in simulated MapReduce PageRank test and keep the first 6 bits of the HMAC values as hashcode. Figure 8 shows the frequency of different hashcode s. It can be seen that although several hashcodes appear more than others, the overall distribution is even, which means an attacker cannot learn much from observing the frequency information. Furthermore, because we use HMAC to generate hashcodes, there are little restrictions of the input format, which makes our method universal.

For certain data sets and applications, after applying our hashcode generation method, the distribution of the result hashcode s could be uneven and still leak some information. In this case, the mapper FPGA can choose to add pseudo key value pairs into the real pairs to make the distribution even and mitigate the possibility of statistical attack.

Another issue is the length of the cipherpair. If the original pairs \((key, value)\) have different sizes, the corresponding cipher-text also have different lengths, which may cause potential information leakage. A straightforward solution to solve this problem is to add some constant data to the original pair before encryption so that the cipher-texts always have the same length.

6.4 **Security Analysis and Evaluation of MapReduce with FPGA Cloud.**

Combining the hashcode protection method together with FPGA cloud, user’s data is properly protected: the plaintexts of user’s data only appear inside the FPGAs for map and reduce, and the cloud service provider has no access to the data. Furthermore, application of hashcode makes it is hard for an adversary to learn statistics information about user’s data.

By applying the hashcode protection method, the

![Figure 8. 6 bits hashcode distribution for a dataset of URLs given in (Leskoveca et al., 2009).](image)

![Figure 9. Execution time under different number of reduce tasks (Zhang et al., 2013).](image)
freedom degree of scheduling is affected. Specifically, the bit number of hashcode sets an upper bound of potential FPGAs as reducers. For example, if the bit number is 6, then no more than $2^6 = 64$ FPGAs can be used as reducers. In practice, this restriction will not affect the overall MapReduce performance very much (Zhang et al., 2013), because MapReduce execution time does not decrease linearly with the increasing number of FPGA reducers. Figure 9 shows the relationship between execution time and number of reducer on different size datasets. From this figure, we can see that 64 reduce tasks (6 bits of hashcode) is a good choice in practice for reasonable size of data. When more reducers are needed, one can raise the upper bound by increasing the hashcode length.

7. K-NN Clustering with FPGA Cloud and MapReduce

In this section we describe applying FPGA cloud for k-NN, a concrete application of MapReduce. Implementation details and experimental evaluation are also provided.

7.1 Short Review of K-NN Clustering

The k-nearest neighbors algorithm (k-NN) is a nonparametric method used for classification. Specifically, k-NN finds a group of $k$ objects in the training set that are closest to the test object, and bases the assignment of a label on the predominance of a particular class in this neighborhood (Wu et al., 2008). Algorithm 1 describes the basic version of k-NN clustering algorithm.

\begin{algorithm}[h]
\caption{k-NN clustering}
\begin{algorithmic}[1]
\State Input: The training objects set $T = \{(x, y)\}$ ($x$ is the data point and $y$ is the class label), the test data point $x'$, and the parameter $k$ ($|D| \geq k$);
\State Output: the class label of $x'$;
\State $D \leftarrow \emptyset$;
\For {each $(x, y) \in T$, where $x$ is the data point and $y$ is the class label}
\State $(d, y) \leftarrow \text{dis}(x, x')$;
\State $D \leftarrow D \cup (d, y)$;
\EndFor
\State Find $D' \subset D$ such that $|D'| = k$, and every $(d', y') \in D'$ satisfies $d' < d$ where $(d, y) \in D - D'$;
\State Find class label $y'$ that appears most in $D'$;
\State return $y'$;
\end{algorithmic}
\end{algorithm}

k-NN clustering algorithm can be ported to MapReduce framework:

- Map function. For a test data point, the map function calculates the distance between the test data point and all the data points in the training set. The test data is used as key, value contains the distance information and the class label;
- Reduce function. For a given key, the reduce function first selects $k$ key-value pairs that have the smallest distance. Then the reduce function checks the k class labels.

7.2 Implementation of K-NN with FPGA Cloud

The implementation of k-NN clustering uses ZedBoard, which is based on Xilinx Zynq-7000 SoC (xc7z020clg484-1). ZedBoard mainly consists of two modules, the FPGA component and an ARM core, which can run Linux. To convert C-based functions of mapper and reducer to hardware modules, Xilinx Vivado High Level Synthesis (HLS) is utilized. We set up PIPELINE directive in Vivado HLS to archive a higher throughput and a lower latency by adding pipeline stages in hardware.

Key related implementation. Proxy re-encryption scheme in (Ateniese et al., 2006) is used to support dynamic job scheduling, and bilinear pairing is the most expensive part of this scheme. Bilinear pairing is usually constructed with elliptic curve points group and calculated using Miller’s (2004) algorithm. The parameters we use here are as follows: the elliptic curve is $E: y^2 = x^3 - x + 1$, which is super-singular. The underlining finite field is $GF(3^{97})$, where the irreducible polynomial is $x^{97} + x^{12} + 2$. Here $G_1$ is a subgroup of the points on $E$, and $G_2$ is a cyclic subgroup of $GF(3^{582})$. The reason we choose these parameters is that in terms of bandwidth efficiency, it is more efficient to use elliptic curves in characteristic three for systems based on bilinear pairing such as Weil or Tate pairing (Galbraith, 2001), and lots of work has been done to optimize bilinear pairing computation with finite field with character three (Beuchat et al., 2008; Kerins, Marnane, Popovici & Barreto, 2005; Iwan & Hyang-Sook, 2003).

On the user and proxy side, encryption and re-encryption functions are implemented based on Miracol library (http://www.certivox.com/miracol). The FPGA only needs to support re-encryption decryption, which equals to one division in $GF(3^{582})$. We adopt the implementation of (Page & Smart, 2003) for this function.

Key-value pair protection/processing implementation. The key-value pair protection consists two components: AES encryption/decryption and HMAC evaluation. An open IP core from the OpenCore.org is used for the AES encryption/decryption. This implementation takes around 30 cycles to encrypt/decrypt a 128-bit data block (operating at 100MHz). HMAC is used for hashcode generation. Like AES, FPGA implementations of HMAC have been studied a lot in the past and our implementation is based on the HMAC module of (Julianto & Gebotys, 2011). The resources consumption of these two modules is summarized in Table 2.
For data processing, the number of clusters does not affect to resource consumption of map function, while dimensions of data point does not cause any change in reducer’s hardware resource, as shown in Table 3.

Table 2. Resources allocated for AES with key length of 128 bits (both encryption and decryption) and HMAC using SHA2-256.

<table>
<thead>
<tr>
<th></th>
<th>DSP48E</th>
<th>FF</th>
<th>LUT</th>
<th>BRAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>AES</td>
<td>0</td>
<td>6482</td>
<td>4548</td>
<td>50</td>
</tr>
<tr>
<td>HMAC</td>
<td>0</td>
<td>1818</td>
<td>3950</td>
<td>2</td>
</tr>
</tbody>
</table>

TABLE 3. Resources allocated for map and reduce function with different parameters. k is the number of clusters, and d is the dimension of data point.

<table>
<thead>
<tr>
<th></th>
<th>DSP48E</th>
<th>FF</th>
<th>LUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>map (d = 4)</td>
<td>4</td>
<td>7</td>
<td>111</td>
</tr>
<tr>
<td>map (d = 8)</td>
<td>4</td>
<td>79</td>
<td>115</td>
</tr>
<tr>
<td>map (d = 16)</td>
<td>4</td>
<td>80</td>
<td>119</td>
</tr>
<tr>
<td>reduce (d = 32)</td>
<td>0</td>
<td>118</td>
<td>213</td>
</tr>
<tr>
<td>reduce (d = 64)</td>
<td>0</td>
<td>121</td>
<td>219</td>
</tr>
<tr>
<td>reduce (d = 128)</td>
<td>0</td>
<td>124</td>
<td>225</td>
</tr>
</tbody>
</table>

In our implementation, a pair of AES encryption/decryption with map/reduce cores and an HMAC core are coupled (Figure 10 shows the map function). We denote the map function coupled with AES and HMAC map slice, and reduce function with AES and HMAC reduce slice. Because the map core runs 2 to 3 times faster than the AES core, we allocate multiple AES cores per map core in order to get higher throughput for the system as a whole. Resource consumptions for the map/reduce slices are presented in the Table 4. The column “Slices per ZedBoard” shows the number of map/reduce slices we can allocate on a ZedBoard based on resource availability.

TABLE 4. Resources allocated for map and reduce slices with AES and HMAC cores.

<table>
<thead>
<tr>
<th></th>
<th>DSP48E</th>
<th>FF</th>
<th>LUT</th>
<th>BRAM</th>
<th>Slices / board</th>
</tr>
</thead>
<tbody>
<tr>
<td>map (d = 4)</td>
<td>4</td>
<td>27824</td>
<td>22253</td>
<td>204</td>
<td>1</td>
</tr>
<tr>
<td>map (d = 8)</td>
<td>4</td>
<td>27825</td>
<td>22257</td>
<td>204</td>
<td>1</td>
</tr>
<tr>
<td>map (d = 16)</td>
<td>4</td>
<td>27826</td>
<td>22261</td>
<td>204</td>
<td>1</td>
</tr>
<tr>
<td>reduce (k = 32)</td>
<td>0</td>
<td>14900</td>
<td>13259</td>
<td>104</td>
<td>2</td>
</tr>
<tr>
<td>reduce (k = 64)</td>
<td>0</td>
<td>14903</td>
<td>13265</td>
<td>104</td>
<td>2</td>
</tr>
<tr>
<td>reduce (k = 128)</td>
<td>0</td>
<td>14906</td>
<td>13271</td>
<td>104</td>
<td>2</td>
</tr>
</tbody>
</table>

Figure 10. Secure Map function core for k-NN executed on Zynq FPGA device: the ARM Linux is responsible for everything except running the Map function kernel. Encrypted test points and training objects are stored in the input BRAM using DMA transactions. The Map function core decrypts the data and calculates the distances. Output key-value pairs are encrypted and stored in the output BRAM. A cryptographic hash key is computed using the output key as input. The ARM CPU will copy the encrypted output data from BRAM to DDR using DMA transactions. Temporary results of the Map function logic are stored in LUT-RAM which is invisible to the ARM. There are multiple DDR memory existing transactors.

TABLE 5. Storage cost for re-encryption key management.¹

<table>
<thead>
<tr>
<th>#FPGA</th>
<th>1k</th>
<th>5k</th>
<th>10k</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4 MB</td>
<td>20 MB</td>
<td>40 MB</td>
</tr>
<tr>
<td>500</td>
<td>20 MB</td>
<td>100 MB</td>
<td>200 MB</td>
</tr>
<tr>
<td>1000</td>
<td>40 MB</td>
<td>200 MB</td>
<td>400 MB</td>
</tr>
</tbody>
</table>

¹ For simplicity, we use 194 bits for an element of GF(3^19).

7.3 EXPERIMENTAL RESULTS OF k-NN WITH FPGA CLOUD

Key related performance. The proxy stores all the re-encryption keys for job scheduling. Table 5 shows the relationship between the storage cost and the number of users/FPGAs in the system. The storage cost given in Table 5 is the upper bound as it assumes there is a re-encryption key for each user and FPGA. For moderate parameters, the re-encryption keys can be easily put into the proxy’s memory.
For a k-NN task, the user may use a single data key \(dek\) for all the data so each FPGA only needs to decrypt \(c_{dek}\) once, and time cost is negligible. The main time consuming parts are cipher-text re-encryption (on the proxy side) and re-encryption key generation (on the user side). Figure 11 depicts the cost of these two parts. The burden on user side will decrease as more re-encryption keys are established.

Key-value pair protection/processing performance. The performance of k-NN with FPGA cloud is affected by two parameters: the data point dimension \(d\) and the number of neighbors \(k\). Figure 12 provides latency/throughput information of map and reduce module. As the map/reduce functions of k-NN are very simple, AES module becomes the performance bottle neck. As the ZedBoard we use for experiments only has limited resources, we cannot deploy more AES modules. Otherwise more AES modules can be coupled with the map/reduce module to achieve lower latency and higher throughput.

8. Related Work

One research area of cloud computing that has received a lot of attention recently is to utilize cryptographic tools to provide secure outsourced computation. Some research efforts focus on specialized applications such as outsourced databases (Hacigumiš et al., 2002). Others attempt to enable pattern matching and/or text searching on cipher-text (e.g., (Boneh et al., 2004; Bellare et al., 2007; Song et al., 2000)). By now, the most powerful tool in terms of cryptographic protection is fully homomorphic encryption (Gentry, 2009) that supports arbitrary operations on cipher-text. Many of these approaches have significant theoretical contributions. However, it is infeasible to apply them in the real world because they either support limited application scenarios or suffer from high computation/storage cost. There is a great need to create practical privacy preserving solutions for big data analytical applications such as those based on the parallel MapReduce programming model. Such a need has not been met by the existing solutions in the literature.

From the perspective of secure FPGA, a lot of research has been done to protect the FPGA bitstream itself. Some researchers have proposed using physical unclonable functions (PUFs) to protect the FPGA bitstream (Guajardo, Kumar, Schrijen & Tuyls, 2007; Kumar, Guajardo, Maesyz, Schrijen & Tuyls, 2008). Other techniques such as watermarking and signature are also used to protect the FPGA IPs (Kahng, Lach & Mangione-Smith, 2001; Lach, Mangione-Smith & Potkonjak, 1999; Schmid, Ziener & Teich, 2008). The main objective of these prior research efforts is to protect the intellectual property of the FPGA bitstream developers, which is to make sure that the bitstream cannot be either reverse-engineered or illegally duplicated. These IP protection techniques cannot be applied directly to solve the problem of data privacy in the cloud.

There are recent efforts on applying FPGAs as accelerators for parallel data analytics (Court et al., 2004; Ronan et al., 2006; Woods & VanCourt, 2008). Shan et al. present a framework to use FPGAs to accelerate MapReduce processing in (Shan et al., 2010). All these related efforts do not consider the problem of data privacy.

9. Conclusion

This paper proposes an FPGA cloud framework that supports privacy preserving computation outsourcing. The FPGA cloud uniquely integrates proxy re-encryption with FPGAs so that users can utilize the capabilities of cloud computing while keeping the privacy of their data. Compared with existing solutions, the FPGA cloud achieves its security goal at reasonable cost. Besides straightforward applications such as signal processing, the paper discusses applying FPGA cloud to support MapReduce. Furthermore, we conduct experiments and evaluation to demonstrate the practicability and effectiveness of the FPGA cloud with k-NN clustering.
10. ACKNOWLEDGEMENT

The authors would like to thank all the reviewers for their valuable comments and suggestions to improve the quality of the paper.

11. REFERENCES


Authors

Lei Xu received the B.Sc degree in Applied Mathematics from Hebei University, China, in 2004, and the Ph.D. of Computer Science from Institute of Software, Chinese Academy of Sciences, in 2011. He is currently a postdoctoral researcher at University of Houston. From 2011 to 2013, he worked as a research engineer at the Central Research Institute, Huawei Technologies Co. Ltd. His research interests include cloud computing and big data security, applied cryptography, and algebraic algorithms.

Khoa Dang Pham is working for the Department of Computer Science, University of Houston as a Research Assistant. With a 2-year experience on digital system and FPGA design, his expertise spreads from serial communication design, digital filter design, complex algorithm implementation to abstractions for reconfigurable fabric development. He received Bachelor degree in the Mechanical Engineering from the Ho Chi Minh City University of Technology, Vietnam in 2008 and Master degree in the School of Computer Engineering from Nanyang Technological University, Singapore in 2014 respectively.

Hanyee Kim received the B.S. degree in computer science education from Korea University, Seoul, Korea, in 2012. He is a graduate student in the combined M.S and Ph.D. program at Korea University. His research interests include embedded systems, computer architecture, parallel...
computer architecture and programming, many-core, and computer science education.

Weidong Shi received his Ph.D. of Computer Science from Georgia Institute of Technology where he did research in computer architecture and computer systems. He was previously a senior research staff engineer at Motorola Research Lab, Nokia Research Center, and co-founder of a technology startup. Currently, he is employed as an assistant professor by University of Houston. In the past, he contributed to design of multiple Nvidia platform products and was credited to published Electronic Art console game. In addition, he authored and co-authored over publications covering research problems in computer architecture, computer systems, multimedia/graphics systems, mobile computing, and computer security. He has multiple issued and pending USPTO patents.

Taeweon Suh is an associate professor in the Graduate School of Information Security, Korea University. Prior to joining academia, he was a systems engineer at Intel Corporation in Hillsboro, Oregon, USA. His research interests include embedded systems, computer architecture, multiprocessor and virtualization. He has a BS in Electrical Engineering from the Korea University, Korea, and an MS in Electronics Engineering from the Seoul National University, Korea, and PhD in Computer Engineering from the Georgia Institute of Technology, USA.
LEARNING-BASED HIGH-THROUGHPUT DISPATCHING FOR TRAJECTORY STREAMS

Xin Zhang1,2, Guoqiang Hu2, Ning Duan2, Peng Gao2, Weishan Dong3, Jun Zhu2, Rong Chang3
1Automation Department of Tsinghua University
z-xin07@mails.tsinghua.edu.cn
2IBM Research – China
{xin, hugq, duanning, bjgaop, dongweis, zhujun}@cn.ibm.com
3IBM Research, USA & China
rong@us.ibm.com

Abstract
With the development of Internet-of-Things (IoT) technologies, large-volume sensor data streams are sent to cloud in near real-time which raise an important requirement for high performance sensor big data analytics. This paper focuses on the scalability challenges for analyzing large-volume mobile data streams, as mobile data streams convey valuable spatio-temporal information (termed as trajectory or semantic trajectory). A framework for high performance trajectory streams processing is proposed together with a learning method for workload assignment optimization and a dispatching method for high-throughput geo-message dispatching. By taking the spatial connectivity implied by trajectory streams into consideration, a trajectory preserving partitioning method is proposed for improving the quality of geo-partitioning. Based on the optimized geo-partitioning, a novel Geohash Tree based dispatching method is developed for achieving high-throughput geo-message dispatching. Via mobility localization formalism, we demonstrate that an implementation of our geo-spatial partition algorithms could balance workload and minimize cross-node communication. And experimental evaluations using real world and simulated data also validate the performance of the proposed methods.

Keywords: service scalability, mobile data streams, graph partitioning, trajectory, geohash tree

1. INTRODUCTION
Mobile sensors (e.g., built-in sensors on mobile phones or portable devices, and in-vehicle sensors) are pervasively available nowadays. Information such as location, speed, temperature, fuel consumption, and driving operation, can be continuously sensed and uploaded to back-end cloud environment in the form of mobile data streams through mobile network. Hence providing cloud services analyzing large volume mobile data streams becomes a key capability in many application domains including connected vehicles, smart traffic and logistic, mobile commerce, and telematics insurance.

The main payload of mobile data streams is the trajectory information. A trajectory is a sequence of spatio-temporal data records describing a journey of a moving object together with the sensed information along that journey. In recent years, trajectory analytics has got increasingly research interest (Bu, 2009), (Liu 2011), (Davics, 2006) along with the booming volume of available trajectory data. And there is also increasing requirement for analyzing trajectory streams with low latency in many applications. In real-time traffic applications that fuse sensed vehicle speed on the same road to estimate road traffic condition, Map Matching (Newson, 2009) infers a vehicle’s traversing roads from a sequence (usually 5 to 20) of sensed GPS records from that vehicle. And in telematics insurance services, driving behavior is expected to be analyzed from a sequence of sensed driving operations (e.g., decelerate-turn-accelerate) immediately for alerting driving risk in-time. And also to enable online trajectory prediction in mobile commerce applications, the recent location records of a moving object, in addition to the current location, shall be considered for accurate prediction. Hence processing trajectory streams in a sliding window approach is a common scheme in many mobility analytics services. And with the growing volume of mobile clients and mobile sensors, throughput and scalability are critical issues for many cloud services analyzing trajectory streams.

In recent years, stream computing platforms (e.g., Storm, InfoSphere Streams (Gedik, 2008), and S4 (Neumeyer, 2010)) become popular for scalable and real-time data stream processing. And there is also solid research outcome on scalable streaming, such as (Andrade, 2009), (Schneider, 2009), (Khandekar, 2009), and stream data mining, such as (Gaber, 2005), (Domingos, 2000), (Chen, 2002). However, to the best of our knowledge, there is little research addressing the scalability challenge of trajectory streams processing. In existing location-based services (LBS), usually the static geo-spatial continuity is considered in geo-partitioning (i.e., partition a geo-space into multiple sub geo-areas). The partitioning result serves as rules for dispatching workload to different servers for parallel processing. While for cloud services analyzing trajectory
streams, geo-partitioning shall also take the geo-spatial continuity implied by the trajectory streams into consideration. Otherwise it would cause large amount cross-server trajectories and incur cross-server data access which will lower parallelization throughput. And service scalability will be impacted.

In this paper, we extend the trajectory preserving partitioning method (Zhang, 2014) into a scalable trajectory streams processing framework. By analyzing the unique requirement in high performance trajectory streams processing, we come up with the scalable framework and major performance measurements. Key algorithms are developed to optimize the workload partitioning and improve the efficiency of service dispatching.

The major research contributions of this paper are:

1. A framework for high performance trajectory streams analysis is proposed which covers both the offline optimal partition learning and online message dispatching.

2. A novel geo-partitioning optimization method is proposed and key algorithms are implemented for optimizing geo-partitioning taking mobility localization and workload into consideration.

3. A Geohash Tree approach and related algorithms are devised to achieve high performance geo-message dispatching.

4. The effectiveness of the proposed framework and methods is validated through experiments on both real-world and simulated data.

The remainder of this paper is organized as follows. Section 2 discusses the trajectory streams features and presents the framework for scalable trajectory streams processing. The mobility localization principle and measurements are also introduced. In section 3, the geo-partitioning optimization method is depicted and related enabling algorithms are introduced. Section 4 presents the devised algorithm for high performance trajectory streams dispatching. Experiments are described in section 5. And related work is given in section 6. Section 7 concludes the paper and highlights the future work.

2. FRAMEWORK FOR SCALABLE TRAJECTORY STREAMS PROCESSING

A general scalability strategy in location-based services is geo-partitioning that enables dispatching data records (or requests) according to their geo-locations. For the benefit of computational efficiency, spatially near-by data records are dispatched to the same processing node (Jensen, 2007), (Mouratidis, 2005). So partitioning the whole geo-space into multiple geo-areas according to the geo-spatial continuity is a common practice in location-based services. After that, computation nodes are associated with partitioned geo-areas to handle corresponding data workload. Through this approach, geo-spatial message can be dispatched to corresponding computation node according to its geo-location and scalable parallel processing can be achieved on cloud.

While for mobility analytics services, beside the static geo-spatial continuity, geo-partitioning shall also take the geo-spatial relationship and workload implied by the trajectory streams into consideration. Otherwise there could be massive communication overhead caused by cross-server data access and service scalability will be degraded. Figure 1 displays an illustrating sample. Two schemes are applied for handling workload on a geo-space with 3 computation nodes. The dark dot lines are the trajectories of moving objects. The rectangles represent partitioned geo-areas and arrowed lines indicate the assignment between computation nodes and partitioned geo-areas. Intuitively scheme 1 (Upper side of Figure 1.) incurs a lot of cross-server data communication as mobility is not considered during geo-partitioning. And the data workload is unbalanced among partitions. While scheme 2 successfully suppresses cross-server communication as it manages most of trajectories within individual geo-areas. And the workload is evenly distributed among partitions. With the capability of reducing cross-node communication and keeping workload balancing, parallelization efficiency will be improved hence scalability can achieved. In real-world services, where there are large volume of moving objects and trajectories and a geo-area usually has irregular shape, determining the optimized partition boundary is not a simple task.

Figure 1. Assignment Schemes.

Motivated by the unique characteristic of trajectory streams, a framework for scalable trajectory streams processing is proposed and shown in Figure 2. In general, the framework consists of the offline part and the online part. The offline part learns from large volume of historical trajectory data and generates optimized geo-partitions as the base for geo-message dispatching rules. The mobility preserved geo-partitioning service associates the trajectory data with map and mines the optimal geo-area boundaries. And the geo-dispatching rule builder processes the partitioned geo-areas and generates rules for geo-message dispatching. The online part has two layers: the messaging layer and the streaming layer. The messaging layer serves as the front-end server handling massive concurrent connections and continuous network messages from mobile
devices. As a core component in the streaming layer, the geo-message dispatching service is responsible for dispatching the mobile data messages to the right computation nodes according to dispatching rules. Throughput is a critical performance measurement for the dispatching service and is one important indicator of service scalability. Modern messaging services usually take the pub-sub schema to decouple message producers and message consumers. The lightweight MQTT protocol (Locke, 2010) enables efficient message transmitting interface between dispatching service and computation nodes. At the streaming layer, computation nodes are a cluster of processing units connecting to dispatching service with high network bandwidth. Each computation node (or a set of nodes) is responsible for processing data located in a certain geo-area. Obviously preserving trajectories locally within individual geo-areas is much more efficient than splitting trajectories onto multiple geo-areas, which incurs cross-server remote access and would significantly impact the processing latency and throughput. So a good architectural strategy is to localize a node’s computation and reduce the cross-node data access or data migration as much as possible. And the more the cross-node communication exists, the less scalable the service is. Therefore geo-partitioning service is critical for service scalability. And the efficiency of dispatching service is also important to the overall runtime performance and to avoid being the bottleneck of cloud services.

In (2.1), \( \{Trj\} \) is the whole set of trajectories used for partitioning. Usually a big volume of historical trajectory data is used representing the statistical mobility pattern of the geo-space to be partitioned. \( P_i \) is the i-th partition and \( t_i \) is a trajectory which has all of its discrete location \( l \) in one partition. And correspondingly we define the Mobility Delocalization Index as \( I_{md} \), which is closely related with cross server communication. \[ I_{md} = 1 - I_d \] (2.2)

Besides the mobility localization, workload balance is another measurement for partitioning quality. In existing geo-partitioning, usually some static factors are regarded for balancing, e.g., the weighted number of road links or the size of spatial area. In the trajectory streams processing context, trajectory streams exactly represent the computation workload. For example, more vehicles are in cities than rural areas. So it is more reasonable to balance the workload according to vehicle data distribution than to cut the city map into sub areas with even geo-spatial size. So in addition to the mobility localization, trajectory-load-based balance is another measurement for partition performance. For the balance constraint, a ratio of imbalance \( R \) (Karypis, 1999) is defined as:

\[
R = \max \left\{ \frac{W_{run}/(W_{run}/k)}{W_{run} \text{ for } 1 \leq i \leq k} \right\}
\] (2.3)

\( k \) is the total number of partitions, and \( W_{run} \) denotes the sum of node weight in the i-th resulting partition, i.e., weight of \( P_i \). For trajectory-load-based balance, the node weight is specifically modeled according to the distribution of trajectories related with the node. \( W_{run} \) denotes the sum of node weight of the whole graph. So \( (W_{run}/k) \) is the average partition weight, which is constant when the graph and the partition number \( k \) are given. A balance constraint can thus be defined as:

\[
R \leq 1 + \varepsilon
\] (2.4)

Where \( \varepsilon \geq 0 \) is a user-defined parameter indicating the tolerance of imbalance. And \( \varepsilon = 1 \) implies the weight of any partition shall not exceed twice of the average partition weight. Users can choose a proper \( \varepsilon \) according to application need to control the imbalance level.

Besides the mobility localization index and ratio of imbalance, the Messaging throughput \( (T_m) \) is another important performance measurement. It is defined as the number of geo-messages dispatched to computation nodes per unit time.

\[
T_m = \frac{n_m}{t}
\] (2.5)

Messaging throughput measures the online geo-message dispatching efficiency and evaluates when the dispatching service may get to be the bottleneck. It is a preferable measurement on general platform performance than the end to end throughput which is application specific. The end to end throughput depends on the computation node configuration and application specific computation complexity, which is beyond the scope of this paper.
3. Trajectory Preserving Partitioning Method and Algorithms

For achieving mobility localization, the spatial connectivity implied by trajectory data has to be leveraged in geo-partitioning. While some spatial grid based approach and geometric partitioning methods, e.g., (Simon, 1991), (Farhat, 1993), are available for geo-partitioning, the rigid grid shape and the limit of spatial distance metric can significantly impact the partitioning result. A recent enhancement to spatial partitioning is road network distance based partitioning (Ventresque, 2012). The road network-based partitioning can more correctly reflect the localized relationship in many applications. For example, two persons who are 200 meters away spatially actually need more than 1 kilometer to meet each other in terms of road network distance. However the focus of (Ventresque, 2012) is on static network based partitioning, without taking data continuity into consideration. And the connection between road network partitioning and geo-partitioning is not addressed.

![Diagram](image.png)

Figure 3. Component Diagram of Trajectory Preserving Partitioning.

In this section a trajectory preserving partitioning method is proposed and corresponding algorithms are developed in following sub-sections. The general idea of trajectory preserving partitioning method is to overlay the trajectory information on top of road network. And then a graph model can be built and graph partitioning method can be applied. After that, the road network partitioning result needs to be projected back into geo-partitions (for the sake that geo-partitions enable much faster dispatching than road network partitions since it’s costly to compute the road link of each data record at the point of message dispatching). Figure 3 shows the component diagram of the trajectory preserving partitioning method. The map road network and historical trajectory data are the input of the method. And the partitioned geo-areas are the output. The partitions here imply that the union of the geo-areas covers the whole geo-space and there is no overlap between any pair of geo-areas. As the first step, the Road Network Transformer transforms the map road network from node graph into link graph.

Section 3.1 will discuss the transformation process. Map Matching is a well-studied algorithm to associate trajectories with road networks. It takes trajectory data as input and output the sequence of road links that the trajectory traverses. Interested audiences can refer to (Lou, 2009), (Newson, 2009) for Map Matching technology. As a front step before graph partition, the Constructor builds a trajectory weighted graph model according to the transformed road network and trajectory data matched on road network, which will be depicted in section 3.2. Then graph partitioning can be performed. Graph partitioning is a classic mathematical technique with abundant applications. In the domain of traffic network analysis, graph partitioning methods are applied in partitioning large-scale road to speedup shortest path search (Delling, 2009), (Delling, 2011) and distributed transportation simulation (Xu, 2012). In this paper, the METIS toolkit (Karypis, 1999) is adopted for partitioning the weighted road network. METIS takes a heuristic partitioning approach and has been widely referred in research and practice for its proven performance. The trajectory preserving road network partitioning algorithm is described in section 3.3. After that an algorithm is developed to project from road network partitions back to geo-partitions, which will be introduced in section 3.4.

3.1 Road Network Transformation.

In most existing road network partitioning methods, the link cut scheme is applied to partition road network by splitting road links connecting different part of sub road networks. It take a straightforward mapping between road network model and graph model, i.e., each node of road network (e.g., road junction or endpoints) is modeled as a vertex of a graph, and each link(connecting two nodes on road network) is modeled as a graph edge connecting two corresponding vertexes. Thus a graph partitioning problem is defined by minimizing the number of road links cut during the partitioning. At the same time, balance constraint can be imposed on the optimization procedure so that the resulting sub-graphs are balanced in terms of the sum of node weight.

For the trajectory preserving partitioning, it turns out to be a different situation. Cutting on road links leaves the issue of which partitions the cut road links shall join. Since road links have their own geo-shape (could be as long as several kilometers), different joining choices can significantly impact the boundary of geo-areas and could lead to a non-optimum solution. On the contrast it won’t be
an issue if partitioning is performed on the nodes of road network: As in road network, nodes are physically modeled as points with no geo-shape. So no matter which partition a cut node joins, it makes no difference to the geo-area boundary. To enable the node cutting, the road network needs to be transformed into a “link graph”. A link graph takes each road link on the road network as its vertex (i.e., link vertex) and builds the edges connecting vertexes based on the connectivity between links. In this way, the partitioning on nodes of road network is enabled.

3.2 Trajectory Weighted Graph Construction.

To build the edges connecting vertexes, and more importantly, to define the weights on the edges, both the road network connectivity and the trajectory information needs to be considered. For each adjacent pair of links, the weight on the edge connecting corresponding link vertexes is increased with a certain value \( w_j \). And for each trajectory traversing a sequence of links, each adjacent pair of links in the trajectory will have a certain value \( w_t \), contributing to the corresponding edge of the graph. The ratio between \( w_t \) and \( w_j \), instead of their absolute value, is important for balancing between the static connectivity and the dynamic connectivity. The static connectivity is more stable with good coverage, while the trajectory is more live through may not cover the full spectrum of network. Merely weighting on static connectivity would lead to non-optimum situation as illustrated in Figure 1. While, in another extreme, purely trajectory based weighting may lead to disconnected graph. As trajectory usually provides many more instances of connectivity than road network (as a pair of links could be traversed by thousands of trajectories), it needs a factor to balance the relative importance of two weights. The principle for weight balance used in this paper is to ensure equal contribution of total weights. For presetting weights \( w_t \) and \( w_j \), a tuning factor, \( r_w \), is calculated reflecting the ratio of total static weight to total dynamic weight.

\[
\begin{align*}
r_w &= \frac{\sum_{\text{link\ pairs}} w_j}{\sum_{\text{traversed\ pairs}} w_t} \\
\end{align*}
\]

(3.1)

Then the \( w_t \) can be tuned by multiplying \( r_w \):

\[
\begin{align*}
w_t' &= w_t \cdot r_w \\
\end{align*}
\]

(3.2)

Static and dynamic weight can be integrated automatically by this approach. One could also adjust \( r_w \), manually if prior knowledge is available on application scenario or data quality.

Trajectory data also serves as more precise workload information than static network or geo-spatial elements. As a direct model of workload, each link vertex in the link graph has the weight that equals to the number of trajectories traversing that link.

3.3 Trajectory Preserving Road Network Partitioning Algorithm.

The trajectory preserving partitioning algorithm is presented in Figure 43. It takes road network and trajectory data as input together with two parameters: \( k \) is the expected number of partitions and \( ubR \) is the upper boundary of imbalance ratio \( R \). It outputs the partitioned sub road networks, each of which contains a list of road links. Step 1-4 of the algorithm build the link graph and estimate the static edge weight \( e.w_l \), the dynamic edge weight \( e.w_d \), and the node workload weight \( v.w_w \). Step 5 calculates the weight tuning factor \( f^w \). And step 6 integrates the static weight with dynamic weight automatically according to \( f^w \).

Step 7 invokes graph partitioning algorithm and gets the partitioned vertex sets. Then in step 8 the partitioned vertex sets are mapped back into sub road networks denoting as lists of road links, which are returned as algorithm output in step 9.

Trajectory Preserving Road Network Partitioning Algorithm

Input:

- \( RN(N,L) \): Road network consists of node set \( N \) and link set \( L \).
- \( Trj = \{ t_i \} \): Trajectory data set. (where \( t_i = (l_{i1},...l_{ik}) \) \( l_{ij} \in L \))
- \( k \): Number of partitions to be generated.
- \( ubR \): Upper boundary of imbalance ratio \( R \).

Output:

- \( \{ subRN_i \} \): each \( subRN_i \) is denoted by \( L_i = \{ l_j \in L \mid l_j \in subRN_i \} \) and \( \bigcup_i L_i = L \land \forall i \neq j L_i \cap L_j = \emptyset \) holds.

BEGIN

\[ w_t = w_j = 1 \], without loss of generality.

1. Create an empty link graph \( G(V,E), V,E=\emptyset \).
2. For each \( l_j \in L \)
   a) Create a link vertex \( v_j \) and add into vertex set \( V \leftarrow V \cup v_j \)
   b) Initialize the workload weight of \( v_j \):
      \( v_j.w_w \leftarrow 0 \)
3. For each \( n_i \in N \)
   a) For each pair of links \( (l_i,l_j) \) connected at \( n_i \)

\[ \text{For clarity of presentation, the multi-link (multiple links between two nodes) handling is omitted.} \]
i. Create an edge $e_{ij}$ and add into edge set $E \leftarrow E \cup e_{ij}$

ii. Set the static weight of $e_{ij}$: $e_{ij}.w_{s} \leftarrow w_{i}$

iii. Initialize the dynamic weight of $e_{ij}$: $e_{ij}.w_{d} \leftarrow 0$

4. For each $t_{j} \in \{T_{j}\}$
   a) Parse each adjacent pair of links $(l_{s}, l_{l})$ of $t_{j}$
      i. Update dynamic weight of $e_{i}^{n}$: $e_{i}^{n}.w_{d} \leftarrow e_{i}^{n}.w_{d} + w_{i}$
      ii. Update workload weight of corresponding vertexes $v_{i}^{n}.w_{w} \leftarrow v_{i}^{n}.w_{w} + 1, i = x, y$

5. $r_{w} = (\sum_{i,j} e_{ij}.w_{s}) / (\sum_{i,j} e_{ij}.w_{d})$

6. For each $e_{ij} \in E$
   a) $e_{ij}.w_{s} \leftarrow e_{ij}.w_{s} + r_{w} * e_{ij}.w_{d}$
   b) For each $v_{j} \in sV_{j}$
      i. Add corresponding link to $L_{j}$: $L_{j} \leftarrow L_{j} \cup l_{j}$

9. Return $\{L_{j} \mid j = 1 \cdots k\}$

END

Figure 4. Trajectory Preserving Road Network Partitioning Algorithm.

The computational complexity of the algorithm depends on the step 4 and step 7. Step 4 has the time cost $O(n_{j})$ (i.e., linear to the number of trajectories in training set $n_{j}$). And Step 7 has the approximate complexity of $O(n + m + klog(k))$ according to the author of METIS (Karypis, 1998), where $n$ is the number of nodes, $m$ is the number of edges, and $k$ is the number of expected partitions. So the algorithm’s computational complexity is approximately $O(n_{j} + n + m + klog(k))$.

3.4 Margin Maximzed Geo-Partitioning Algorithm.

The trajectory preserving road network partitioning algorithm introduced in section 3.3 generates the road network partitions as output. Each road network partition contains a list of road links. For geo-messages in trajectory streams, the location information is generally presented in terms of geo-spatial coordinates (e.g., longitude and latitude). Since associating coordinates with road links relies on Map Matching which is computationally costly, road network partition is not an ideal structure for online geo-message dispatching. In contrast to a road network partition defined as a list of road links, a geo-partition is defined in terms of a geo-area having a polygon as its boundary. Computationally matching coordinates with polygons is much more effective than associating coordinates with road links. Hence there is a need to transform from road network partitions into geo-spatial partitions for effective geo-message dispatching.

Developing one-one mapping between road network partitions and geo-partitions (represented by a polygon as the boundary) has the following requirements. Basically each polygon shall include all road links allocated to the corresponding road network partition and exclude any road links of other partitions. And there shall be no overlapping or missing coverage between the geo-partitions. Moreover, the position of boundary shall maximize its distance from the data of adjacent partitions as much as possible, so as to be resilient to data noise (if the boundary is close to data, noisy data could float into the other partition and cause wrong dispatching). This raises the need to maximize the margin between polygon boundary and road links near polygon boundary.

To fulfill the above requirements, the Margin Maximzed Geo-Partitioning Algorithm is developed and shown in Figure 5. The algorithm determines a compact hull for each road network partition $subRN_{i}$ and extends those compact hulls into a geo-partition solution in three steps. Firstly, based on the detected compact hulls, the algorithm discovers buffer zones. Then for each buffer zone, it generates separating polylines maximizing the margin in buffer zone. After that the compact hulls are extended into geo-partitions by replacing some lines of compact hull with separating polylines (and corresponding nodes as well). The algorithm assumes that a road network can be modeled as a planar graph, which is valid for most road networks.

**Margin Maximzed Geo-Partitioning Algorithm**

**Input:**
- $\{subRN_{i}\}$: each $subRN_{i}$ is denoted by a list of road links $L_{i} = \{l_{i} \mid l_{i} \in L \land l_{i} \in subRN_{i}\}$.
- $\{pn_{i}\}$: set of partitioned nodes across $subRN_{i}$ s.

**Output:**
- $\{GeoArea_{i}\}$: each $GeoArea$ is represented by a polygon as its boundary.

**BEGIN**

1. Generate a compact convex hull $H_{all} = (H_{n}, H_{l})$ for the whole road network $\bigcup subRN_{i}$. $H_{n}$ and $H_{l}$ are the boundary nodes and boundary lines respectively.

2. Clockwise traverse the boundary polygon and mark the direction of each line on the boundary the same as traversing direction.

3. Generate a compact hull $H_{i} = (H_{sn_{i}}, H_{sl_{i}})$ for each sub road network $\{subRN_{i}\}$. $H_{sn_{i}}$ and $H_{sl_{i}}$ are the boundary nodes and boundary lines of $subRN_{i}$ respectively.
4. Clockwise traverse the boundary polygon of each sub road network and mark the direction of each line on the boundary the same as traversing direction.
5. Initialize buffer zone set \( \{ Z_b \} \leftarrow \emptyset \)
6. For each node in the partitioning node set \( n \in \{ p_{n_j} \} \)
   a) Discover buffer zones \( z_j \) starting from \( n \) by searching in all hull elements \( H_m \cup H_j \).
   b) \( \{ Z_b \} \leftarrow \{ Z_b \} \cup z_j \)
7. Initialize separating polyline set \( \{ P_s \} \leftarrow \emptyset \)
8. For each \( z_j \leftarrow \{ Z_b \} \)
   a) Find the separating polylines \( \{ p_k \} \) to maximize the margin in the buffer zone \( z_j \).
   b) \( \{ P_s \} \leftarrow \{ P_s \} \cup \{ p_k \} \)
9. For each \( \text{subRN}_j \)
   a) Extend \( \text{subRN}_j \)'s hull \( H_j = \{ H_{sn_j}, H_{sl_j} \} \) to margin maximized boundary \( B_j = \{ B_{sn_j}, B_{sl_j} \} \) by replacing some of its boundary lines with those separating polylines \( p_i \in \{ P_s \} \) which have both end nodes belong to \( H_{sn_j} \). Maximized margin geo-partition for \( \text{subRN}_j \) is bounded by \( B_j \).
10. Return margin maximized boundary set \( \{ B_j \} \)

END

Figure 5. Margin Maximized Geo-Partitioning Algorithm.

To determine the compact hull (i.e., spatial outer borders) of a road network can be realized by constructing a minimal hull to encompass all nodes of the road network. Since each edge of the road network stands for a road segment modeled by a straight line, the resulting hull encompasses all the network edges as well. Algorithms are available for the hull generation (e.g., (DeBerg, 2000)). By representing each edge using its vertex points, hull generation algorithm can output a polygon as the boundary of a list of network edges. And each vertex of the polygon is a boundary node of the road network and each edge stands for an outer border. To integrate the border information into the road network model, each edge of the polygon (i.e., line) is added to the road network as a new edge, if there is no overlap with existing network edge. After compact hulls are identified, all nodes and lines which are not in the compact hulls \( H_m \cup H_j \) are regarded as internal elements and can be ignored in the following operations. And it’s obvious that all partitioned nodes are kept in compact hulls’ node set \( H_m \cup H_{sn_j} \).

A buffer zone can be specified by its border (i.e., a polygon composed by lines in compact hulls \( H_m \cup H_{sl_j} \), and with no road network node in it). A sample is shown in Figure 6. The discovery of each buffer zone starts from a boundary node between two sub road networks (i.e., partitioned node), e.g., the boundary node V. And then it traverses on \( H_m \cup H_{sl_j} \), following the link direction generated in step 2 and step 4 of Figure 5 (i.e., clockwise direction of each individual polygon). When getting into another boundary node share by two sub network (e.g., boundary node U in Figure 6), it switches to the boundary of another hull and continues the traversing. The process stops when the starting boundary node is revisited. Then the traversed lines form a polygon as the border of a buffer zone.

Figure 6. Sample of Discovering Buffer Zone.

Figure 7. Sample of Finding Separating Polylines Maximizing Margin in Buffer Zone.

For finding the separating polylines to maximize the margin in the buffer zone, a triangulated approach is implemented. The buffer zone is triangulated first and the division lines are obtained by chaining the centroid of triangle and connecting further to the boundary nodes. The resulting division polylines become the border lines of final geo-spatial partitions. A sample is depicted in Figure 7 which finds the separating polyline in a buffer zone involving two sub road network boundaries. When a buffer zone border involves multiple sub road networks, similar approach can be applied and multiple polylines will be output. For the interest of paper length, the detail is the omitted here. The Margin Maximized Geo-Partitioning Algorithm has the complexity...
of $O\left(k \times n \times \log(m) + n^p \times n^b \times \log(m_b)\right)$, where $n$ is number of nodes, $m$ is the number of edges, $k$ is the number of partitions, $n_p$ is the size of $\{pn\}$, $n_b$ is the size of $Hn \bigcup Hsn$, and $m_b$ is the size of $Hl \bigcup Hsl$.

With the method proposed in this section, optimized geo-partitions can be generated from road network partitions. On top of the geo-partitions, high performance geo-message dispatching can be enabled, which will be introduced in the next section.

4. High Performance Geo-Message Dispatching

For dispatching geo-messages, the dispatching service needs to compare the location information in each geo-message with boundary of geo-areas to identify the right geo-area the message located. Then the message will be routed to the computation node responding for the workload of that geo-area. Comparing to key based or attribute-value based dispatching in ordinary data streams, geo-message dispatching is still heavier in terms of computation: calculating the geo-spatial relationship is in general more complex than value matching. So the dispatching service is prone to scalability bottleneck.

In this section, based on the Geohash technology (Fox, 2013), a Geohash Tree scheme is proposed to codify the dispatching rules for high performance geo-message dispatching. The Geohash Tree gains the performance advantage by transforming geo-spatial matching into less expensive Geohash code matching. And moreover, Geohash Tree takes a hierarchical and flexible layered structure comparing with grid approach: when a node has its bound within a single partitioned geo-area, the node is marked as leaf node and needs not to be split further. Therefore both the storage space can be saved and levels of comparison can be reduced. Geohash Tree has the similar idea of spatial index tree such as R+-tree (Sellis, 1987). While a major difference in a Geohash Tree is that each node has its Geohash code by inheriting the Geohash code from its parent node and extending with one character. The character uniquely differentiates the node from other nodes with the same parent and maps to a specific sub-area of the parent’s area. The number of children that a node can have equals to the cardinality of the Geohash character set. In practice, a Geohash Tree with its character set cardinality of 32 can reach to meter level resolution on its node in the 10th level of depth. On the one hand, Geohash tree shares the same advantage as R+-tree on effective indexing and space saving comparing to grid index scheme (by enabling leaf node on high level of the tree and avoiding massive low level pieces). On the other hand, Geohash tree replaces R+-tree’s geo-spatial matching with character matching, which is computationally much more effective.

Figure 8 gives the algorithm pseudo-code for generating a Geohash Tree. And Figure 9(b) illustrates a sample instance of Geohash Tree and Figure 9(a) visualizes the relationship between Geohash codes and spatial areas.

**Dispatching Rule Generation Algorithm**

**Input:**
- $\{B_i\}$: Boundary polygons of geo-partitions.

**Output:**
- $\text{GHTree}$: Geohash Tree. Each node on the $\text{GHTree}$ has the attributes of $(\text{code, isleaf, \{geo_area\}, bound, \{child\}})$. $\text{code}$ is the Geohash code of the tree node; $\text{isleaf}$ is true if the node is leaf node (with no children nodes); $\text{\{geo_area\}}$ is the areas the node maps to; $\text{bound}$ is the geo-spatial boundary of the node in terms of up-left limit point and down-right limit point; $\text{\{child\}}$ contains the children nodes if it’s not a leaf node.

BEGIN
1. Initial a node queue: $q_{node} \leftarrow \emptyset$
2. Calculate bound of the whole area and generate the Geohash code $\text{code}$ to cover the bound.
3. Create a root node: $\text{root} \leftarrow \text{(code, FALSE, [ALL\_GEO\_AREAS], bound, NULL)}$
4. $\text{GHTree.root} \leftarrow \text{root}$
5. Push the root node into the node queue: $q_{node} \leftarrow \text{push(root)}$
6. While ($q_{node}$ is not empty)
   a) Poll a node from queue: $n_d \leftarrow q_{node}.\text{poll()}$
   b) If ($\|n_d.\text{geo_area}\|_b \leq 1$)
      ii. $n_d.\text{isleaf} \leftarrow \text{TRUE}$
      ii. $n_d.\text{\{child\}} \leftarrow \text{NULL}$
   c) Else
      ii. $n_d.\text{\{child\}} \leftarrow \emptyset$
      ii. For each option char $\text{cr}$ of Geohash code
         1. Compute the corresponding sub boundary $b_i$ in $\text{nd.bound}$.
         2. Find the overlapped areas $\text{ga}$ between $b_i$ and $n_d.\text{\{geo_area\}}$.
         3. If ($\|\text{ga}\|_b = 0$) // out of scope.
            a) Continue;
         4. Else if ($\|\text{ga}\|_b = 1$)
            a) New a leaf node: $n_d \leftarrow \text{(code + cr.\text{TRUE, ga.b_i, NULL})}$
            b) Add $n_d$ to nd ’s children list.
    5. Else $//$ may need to break down
       a) If reaches max tree level
          i. $n_d.\text{isleaf} \leftarrow \text{TRUE}$
          ii. $n_d.\text{\{child\}} \leftarrow \text{NULL}$
       b) Else

END
i. New a node:
   \[ nd, \leftarrow \{ \text{code, cr, false, go, b, NULL} \} \]
ii. Add \( nd \) to \( nd \)'s children list.
iii. Add to queue: \( q_{node}.push(nd,) \)

7. Return \( GHTree \).

END

Figure 8. Dispatching Rule Generation Algorithm.

The Dispatching Rule Generation Algorithm adopts a breadth-first approach to explore and build the tree. It initializes a root node and sets its Geohash code covering the whole geo-area of interest. Then the root node is pushed into queue as the starting seed. Step 6 is the main body of the algorithm. In the loop of step 6, nodes are polled out of the queue in the same sequence as they enter the queue. By evaluating the relationship between the polled node’s boundary and the geo-areas, actions are taken on the node to either mark the node as leaf or explore the children of the node further. In the action of further exploration of children nodes (i.e., step 6.c) of Figure 8), a child node could be either added to tree as leaf node (if it maps to a single geo-area), or discarded (if it maps to no geo-area of interest), or pushed into queue for further exploration (if it maps to multiple geo-areas). In the example in Figure 9, we explore the space from very beginning to ‘f2j’ and then explore its grandchildren ‘f2j2p’, ‘f2j2pc’, etc. The ‘f2j2p’ node needs to be explored further since it covers 3 geo-areas (as Figure 9(a) shows). So ‘f2j2p’ is further split into nodes ‘f2j2p4’, ‘f2j2pd’, ‘f2j2pz’, ‘f2j2pr’, ‘f2j2pe’, ‘f2j2pt’, etc. Since each node of ‘f2j2p’, ‘f2j2p4’ and ‘f2j2pt’ covers only one partition area, they are marked as leaf node and will not be further drilled down. On the contrast, ‘f2j2pd’, ‘f2j2pe’ and ‘f2j2pz’ has to be explored further as each of them covers multiple geo-areas. Ideally each leaf node on the resulting Geohash Tree shall map to exactly one geo-area, if the depth of tree is unrestricted. While in practice, the depth of tree stops at a level (mostly between 6 and 11) for storage and precision consideration. So there would be a small portion of leaf nodes across multiple geo-areas when these leaf nodes are over the partition boundaries. Only in those cases, geo-spatial calculation is needed for determine the right geo-area for a geo-message. The Dispatching Rule Generation Algorithm is relatively time consuming. In worst case it takes \( d^{d+1} \times k \) times of geo-relationship calculation if all leaf nodes are at the most depth of the tree and there are no pre-ending branches in the intermediate levels of the tree. Here \( c \) is the cardinality of code, \( d \) is the maximum levels of tree, and \( k \) is the number of partitions. Fortunately that will not happen as Geohash Tree can always function and cut branches radically at early stage. In practice, it takes less than a minute to generate a Geohash Tree for a 100,000 km² and 6 partitions, which is affordable for offline processing.

Figure 10 gives the detail of Geo-message dispatching algorithm which takes the Geohash Tree as its dispatching rule.

Geo-Message Dispatching Algorithm

Input:
- \( m \): a geo-message which is a tuple with the following attributes \((id, lon, lat, ts, message_body)\).
- \( id \) is the moving object’s id; \( lon \) and \( lat \) are the location information in longitude and latitude; \( ts \) is the timestamp of the message; \( message_body \) has information that needs to be dispatched to computation node.

Output:
- GeoArea id: the GeoArea id (eventually, the responsible computation node id) that the geo-message shall be dispatched to.

BEGIN
1. Get Geohash Tree Root Node \( nd \leftarrow GHTree.root \).
2. Generate Geohash code for \( m \):
   \( m.code \leftarrow \text{geohash.gen}(m.lon, m.lat) \)
3. If \( m \) is within the area of interest: \( has\_prefix(m.code, nd.code) \).
   a) While (not \( nd.isleaf \))
      i. Find \( nd \)'s child node \( nd \), that
      \( has\_prefix(m.code, nd.code) \).
      ii. Get down a level: \( nd \leftarrow nd \),
   b) If (\( nd.(geo\_area) \mid_k = 1 \))
      i. Return \( nd.geo\_area.id \)
   c) Else
      i. Do geo-spatial comparison between \( m \)'s location with the geo-areas \( nd \) covers.
      ii. Return the matched geo-area id.
4. Else \( m \) is out of the scope.
   a) Return Out-Of-Map-Scope error.

END

Figure 10. Geo-message Dispatching Algorithm.
generation operation and several operators on tree node access for dispatching a geo-message, which is very efficient.

5. EXPERIMENT

In this section, the performance of proposed methods and accompany algorithms are evaluated by experiments using both real world trajectory data collected from GPS equipped vehicles and simulated trajectory data streams. The performance measurements are the Mobility Localization Index \((I_m)\), the ratio of imbalance \((R)\), and the Messaging Throughput \((T_m)\) which are introduced in section 2. A set of experiments are conducted on partition performance for validating the soundness of trajectory preserving partition method. The experiment context and result will be introduced in section 5.1. Another set of experiments are perform for real-time messaging dispatching performance, which will be introduced in section 5.2.

Table 1. Experiment Real-World Datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>CityWideU (^a)</th>
<th>UrbanWideB (^b)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Road network size</td>
<td>62,810</td>
<td>31,434</td>
</tr>
<tr>
<td>(number of links)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Geo-message volume</td>
<td>80,297,139</td>
<td>57,102,570</td>
</tr>
<tr>
<td>(number of records)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Trajectory data</td>
<td>416,840</td>
<td>315,636</td>
</tr>
<tr>
<td>volume (number</td>
<td></td>
<td></td>
</tr>
<tr>
<td>of trajectories)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Average Length of</td>
<td>192.6</td>
<td>180.9</td>
</tr>
<tr>
<td>Trajectory (number</td>
<td></td>
<td></td>
</tr>
<tr>
<td>of records) c</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Dataset characteristics:

\(^a\) CityWideU is on a larger map with unbalanced trajectory distribution (trajectories on certain areas of the road network are much dense than those on other areas).

\(^b\) UrbanWideB is on a small map with near balanced trajectory distribution.

\(^c\) Mobile Data was sampled every 25 seconds/record in average.

Two real-world datasets were collected and used for experiments, as shown in Table 1. The trajectory data was generated from 3000 GPS equipped vehicles travelling in a large city in 7 days. Each raw trajectory is a sequence of GPS data records (and each record contains longitude, latitude, velocity, timestamp, and vehicle ID). After the preprocessing of associating GPS with road network, a trajectory of a vehicle can be expressed as a sequence of road links that the vehicle traverses. The first dataset, CityWideU, is a relatively large dataset which contains imbalance workload: central urban traffic is much heavier than that of sub-urban areas. From this perspective, the trajectory density is a good reflection of the urban roads popularity, but is biased when the suburban roads are concerned. The second dataset, UrbanWideB, has a smaller scale and a more balanced trajectory distribution. For evaluating geo-message dispatching throughput, simulated data was generated beside the real-world datasets. The detail will be introduced in section 5.2.

The experiments were conducted on a cloud environment using SoftLayer machines. The server for geo-spatial partition and message dispatching is a dedicated server. Its configuration is 4-CPU x86 Server with 16 G memory and Linux OS. The computation nodes are virtual machines with 4-CPU x86 Server with 8 G memory running map matching tasks. As formerly noted, since this paper focuses mainly on the message dispatching throughput instead of application dependent workload throughput, we ensure there are sufficient computation nodes and they would not be a source of bottleneck in the experiments.

5.1 Experiments on Partition Performance.

In the sub section, the Mobility Localization Index \((I_m)\) and the ratio of imbalance \((R)\) are evaluated for the partition performance using datasets described in Table 1. We denote the proposed trajectory preserving partition method and algorithms as TPP. And the road network-based partitioning method (Ventresque, 2012), denoting as RNP, is used as the baseline scheme for performance comparison.

Since the geo-partitioning is a learning process, the experiments follow the cross validation scheme to avoid over-fitting. Specifically ten-fold cross validation is applied and the final result is consolidated from the ten running output.

The number of expected partitions is the major parameter of the partition algorithms. In the experiments, different numbers of partitions (4, 6, 8, 10, 12 partitions respectively) are tested on both datasets. In the following the experiment result of both datasets will be presented.

Experiment Result on CityWideU. Table 2 summaries the overall partitioning result of both TPP and RNP on the CityWideU dataset with different setting of partition numbers. And the overall ratio of improvement is presented in the last column. In Table 2, the average number of cross-partition trajectories with various setting of partition numbers. And it numbers. And the overall ratio of improvement is presented as the base). Also TPP provides more balanced result and reduces the Ratio of Imbalance \((R)\) by 67.11% comparing with RNP.

Figure 11 displays the detail partitioning result on the number of cross-partition trajectories with various setting of partition numbers. And Figure 12 is the detail result on the Mobility Localization Index \((I_m)\). TPP gets higher \(I_m\) than RNP in every tested setting of partition numbers. And it stably keeps the \(I_m\) beyond 90% while RNP drops down to less than 80% when the partition number gets larger.
Table 2. Summary Result of Experiment on CityWideU

<table>
<thead>
<tr>
<th>Measurement</th>
<th>TPP</th>
<th>RNP</th>
<th>Ratio of Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Cross-Partition Trajectories</td>
<td>6</td>
<td>0</td>
<td>60.83%</td>
</tr>
<tr>
<td>Mobility Localization Index ($I_{ml}$)</td>
<td>%</td>
<td>%</td>
<td>12.32%</td>
</tr>
<tr>
<td>Average Ratio of Imbalance ($R$)</td>
<td>1.159</td>
<td>3.524</td>
<td>67.11%</td>
</tr>
</tbody>
</table>

The ratio of imbalance detail of CityWideU is shown in Figure 13. Considering the CityWideU dataset is characterized as the geospatially imbalanced distribution of trajectory workload, the ratio of imbalance would have challenge to control. The result of RNP algorithm with different numbers of partitions verified this. The ratio of imbalance always approaches or exceeds 3.0, indicating the workload of some partition is 3 times the average workload of all the partitions. While the TPP remarkably keeps the ratio of imbalance in the range of [1.03, 1.25], which means a nearly even workload distribution among partitions (i.e., the partition with the heaviest workload is less than 25% higher to the average workload).

Experiment Result on UrbanWideB. Table 3 summarizes the partitioning result on the UrbanWideB dataset. The UrbanWideB dataset is a smaller dataset than CityWideU, and is more connected with dense trajectories. In general it is more challenging to sustain the mobility localization. So it costs more in partitioning. This can be observed by comparing Table 3 with Table 2: both the average number of cut trajectories and the average mobility localization index of Table 3 are lower than those of Table 2. On the UrbanWideB dataset, TPP gets in average 19.74% of improvement on the $I_{ml}$ gets remarkably 44.54% improvement on the Ratio of Imbalance comparing to RNP. The details of the cross-partition trajectories and mobility localization comparison result on the UrbanWideB dataset are shown in Figure 14 and Figure 15 respectively. The $I_{ml}$ of partition result by TPP are mostly more than 90% (except the case that partition number is 12). And in each setting of partition number, TPP gets more than 10% higher on $I_{ml}$ than that of RNP.

Table 3. Summary Result of Experiment on UrbanWideB

<table>
<thead>
<tr>
<th>Measurements</th>
<th>TPP</th>
<th>RNP</th>
<th>Ratio of Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Cross-Partition Trajectories</td>
<td>30,001</td>
<td>93,765</td>
<td>68.00%</td>
</tr>
<tr>
<td>Mobility Localization Index ($I_{ml}$)</td>
<td>92.80%</td>
<td>77.51%</td>
<td>19.74%</td>
</tr>
<tr>
<td>Average Ratio of Imbalance ($R$)</td>
<td>1.195</td>
<td>2.154</td>
<td>44.54%</td>
</tr>
</tbody>
</table>

Figure 16 is the detail result of Ratio of Imbalance. RNP improves its performance since the UrbanWideB dataset is much more balanced than the CityWideU dataset. But it has still at least 0.8 higher than TPP on each of the cases.
5.2 Experiments on Dispatching Performance.

Messaging throughput is evaluated in this sub section. A grid based dispatching method is implemented for comparison. Grid index is a single layer structure. It’s very effective to locate a spatial point to a cell of the grid. The size of the grid cell is the major parameter that could affect indexing performance. Here we choose two grid sizes: Grid_DA50 is the grid dispatching algorithm with 50 meter cell configuration. And Grid_DA200 is the grid dispatching algorithm with 200 meter cell configuration. GHT_DA represents the Geohash tree based dispatching algorithm we proposed in section 4 (i.e., the Dispatching Rule Generation Algorithm and the Geo-Message Dispatching algorithm).

Figure 14. Cross-Partition Trajectories on Partitioning UrbanWideB.

Figure 15. Mobility Localization Index on Partitioning UrbanWideB.

Figure 16. Ratio of Imbalance on Partitioning UrbanWideB.

TPP has its computation time range from 52 seconds to 5 minutes generating an experiment result. RNP requires much less time for partitioning. It generates result within 30 seconds in any of the experiment cases. Since the partition is a learning process that can be done offline, the time is not a big concern. When there is a need to perform partitioning in online mode to reflect the trajectory pattern changes timely, the computation time will be an important measurement.

Figure 17. Geo-Message Dispatching Throughput on CityWideU.

Figure 18. Geo-Message Dispatching Throughput on UrbanWideB.

Figure 17 and Figure 18 are the messaging throughput result on CityWideU and UrbanWideB respectively. In the case of CityWideU, the throughputs of GHT_DA range from 0.8–1 million records/second in different configurations of partition numbers. The throughputs of Grid_DA50 and Grid_DA200 are in the range of [390k, 630k]. Grid_DA50 gets slightly better result than Grid_DA200. This attributes to the smaller grid size which reduces the probability for calculating geo-shape
relationship. The Figure 18 shows the similar result. In experiments on both datasets, GHT_DA has more than 30% improvement of the message dispatching performance.

For further validating the performance on large-scale map with more choices of partition numbers, a simulation is implemented on a map with 420,633 road segments. 6,955,046,200 geo-messages are generated and sent to server by 8 client machines in UDP protocol. The partition numbers are set to 4, 10, 20, 40, and 60 respectively. The dataset is denoted as Simu6B and the result is shown in Figure 19. The result is consistent with those on CityWideU and UrbanWideB and shows the promising generalization capability to apply the technology to large-scale map.

Figure 19. Geo-Message Dispatching Throughput on Simu6B

6. RELATED WORK

Providing high quality, scalable and real time analytics is one unprecedented challenge for the Internet-of-Things (Aggarwal, 2013). There have been research interests enabling scalable cloud services for IoT applications from resource virtualization, networking and architecture angles (Kovatsch, 2014), (Mukherjee, 2014), (Li, 2013). The research on scalable data stream processing has two main threads. The first thread takes the architectural perspective to enhance the programming model and develop optimization technologies for platform scalability. The second thread focuses on algorithm improvement for high performance distributed data stream analytics.

(Andrade, 2009) defined a streaming architectural pattern for the sensor-and-response application domains. Scalable application can be generated following the architectural pattern and corresponding programming model. (Khandekar, 2009) solved the assignment problem of processing elements (PEs) to processing hosts (PHs) in the context of high scalable distributed stream processing. It employed a graph partitioning method to minimize the inter-PH network communication while simultaneously balancing load across the PEs. In light of stream workload dynamics, (Schneider, 2009) proposed a technique to dynamically adjust the amount of computation of an operator. (Cherniack, 2003) introduced two stream processing systems and discussed the load (re-)partitioning issue in the context of generic workload. (Pietzuch, 2006) considered the dynamic network condition to relocate operator for higher stream processing performance. The existing work regarded mostly generic data streams and did not address the mobile characteristics.

There is also solid research outcome on scalable data streams analytics focusing on algorithm improvement (Gaber, 2005). (Domingos, 2000) proposed a general method, called VFML (Very Fast Machine Learning), for scaling up machine learning algorithms. In more recent years, the research on trajectory streams has it applications on anomaly detection (Bu, 2009), spatio-temporal causal interactions discovering (Liu, 2011), and map generation (Davies, 2006). This category of work forms the advanced analytics workload for scalable streams processing platform. Besides stream processing platform, graph partitioning technologies have been well studied and applied in parallel processing. Interest audiences can refer (Buluc, 2013) for recent advance of graph partitioning technologies as well as their applications in parallel processing.

7. CONCLUSIONS AND FUTURE WORK

In the IoT era, large-volume mobile data streams are generated from pervasive mobile sensors. This provides opportunities for precise mobility awareness and timely mobility insight analysis. This paper focuses on the scalability challenge of cloud services analyzing trajectory streams. With the proposed trajectory preserving partition method and Geohash Tree based dispatching method, high throughput streams processing cloud services can be achieved with high quality of balanced workload and reduced cross-server communication. The algorithms are developed and validated through experiments.

There are two interesting topics we would like to further pursue. Currently under the assumption that the workload pattern is stable and identical to that of historical data, the paper addresses the offline partitioning problem. While real world workload could dynamically shifts along with time. This raises the further requirement of online (re-)partitioning in face of the dynamics. Online repartitioning needs to take partition adjustment cost into consideration and requires incremental design on algorithm to avoid dramatic fluctuation. Besides that, the computational efficiency of online partitioning algorithm is also of interest. And this paper puts its focus mainly on the geo-message dispatching efficiency. In the future, with the maturity of the mobile data streams analytics workload patterns, end-to-end measurements of scalability performance with typical application workload patterns can be further investigated and validated.
8. REFERENCES


AUTHORS

Xin Zhang is with IBM Research – China and Automation Department of Tsinghua University, Beijing, China. He got his master degree in computer science and technology from BeiHang University, and joined IBM Research in 2000. His main research interest includes mobility big data analytics, mining data streams, graph model and graph-based optimization technology. In recent years, he led multiple projects in the domain of asset preventive maintenance analytics, railway transportation schedule optimization, IoT-based intelligent traffic systems, scalable dynamic map analysis, and big data analytics.

Guoqiang Hu received his B.S. Degree from the Shanghai Jiao-Tong University, China, in 1997 and the M.S. Degree in Information Technology from the University of Stuttgart, Germany, in 2002. From 2002 to 2007, he was with the Institute of Communication Networks and Computer Engineering, Stuttgart, Germany as a research staff and Ph.D. candidate. From 2008 to 2010, he worked as a research fellow in the Centre for Quantifiable Quality of Service in Communication Systems, Trondheim, Norway. He joined IBM Research - China in 2010 and researched in the domain of Internet-of-Things and Connected Vehicles. His research interests include distributed system architecture design, performance evaluation and large-scale data mining technologies.

Ning Duan is a research staff member of IBM Research, China. His research mainly focuses on the big moving object data mining and management. He has been in the area of data mining for almost 6 years. And apply big data mining and machine learning technology on connected vehicle area for more than 3 years. He received his Master degree in 2006 from XiDian University of China.

Peng Gao is a staff researcher of IBM Research, China. He received Ph.D. from Tongji University in 2010, and joined IBM subsequently. With a research experience of 10 years in Traffic & Transportation domain, he once served as a trustee of Traffic & Transportation Engineering Society of Tongji University. His research interests include: Complex Science, Intelligent Transportation System, Multi-agent System & Simulation, and Analytics & Optimization. He has over 10 publications and 12 patents.

Weishan Dong is a Research Staff Member from IBM Research, China. He acts as a research team leader working on big data processing, spatiotemporal analytics, and mobile computing. He received his B.E. degree in computer science and technology from the University of Science and Technology of China (USTC) in 2004, and his Ph.D. degree in pattern recognition and intelligent system from the Institute of Automation, Chinese Academy of Sciences in 2009. He joined IBM Research – China in 2009. His current research mainly focuses on data mining, especially mining big spatiotemporal data with addressing large scale and low latency. He is also interested in evolutionary computation and computer vision topics. Dr. Dong has more than 30 refereed publications in international journals and conferences and over 20 patents.

Jun Zhu is a Senior Technical Staff Member at the IBM Research, China in Shanghai and Senior Manager driving the Smarter Mobility research project. He joined IBM after graduation from Shanghai JiaoTong University in 2001, and has been involved in several research projects including model-driven business process analytics, cloud-based service delivery platform, data-driven testing planning & optimization, and connected vehicle service platform & data analytics solutions. Mr. Zhu was an IBM Master Inventor with more than 50 patents filed. He published 30+ papers.

Rong N. Chang is with IBM Research, USA & China, leading a global team creating innovative IoT cloud services technologies. He received his PhD degree in computer science and engineering from the University of Michigan at Ann Arbor in 1990 and his B.S. degree in computer engineering with honors from the National Chiao Tung University in Taiwan in 1982. Before joining IBM in 1993, he was with Bellcore researching on B-ISDN realization. He has won one IEEE Best Paper Award, received four IBM corporate-level Outstanding Technical Achievement Awards, held 30+ patents and published 40+ papers. He is Member of IBM Academy of Technology, ACM Distinguished Engineer, Chair of IEEE Computer Society Technical Committee on Services Computing (TCSVC), Chair of 2015 IEEE World Congress on Services, EIC of the Int. J. of Cloud Computing, and Associate Editor of IEEE Trans. on Services Computing and the Int. J. of Services Computing.
Call for Articles
International Journal of Services Computing

Mission
The International Journal of Services Computing (IJSC) aims to be a reputable resource providing leading technologies, development, ideas, and trends to an international readership of researchers and engineers in the field of Services Computing. To ensure quality, IJSC only considers extended versions of papers published at reputable international conferences such as IEEE ICWS.

From the technology foundation perspective, Services Computing covers the science and technology needed for bridging the gap between Business Services and IT Services, theory and development and deployment. All topics regarding Web-based services lifecycle study and management align with the theme of IJSC. Specially, we focus on: 1) Web-based services, featuring Web services modeling, development, publishing, discovery, composition, testing, adaptation, and delivery, and Web services technologies as well as standards; 2) services innovation lifecycle that includes enterprise modeling, business consulting, solution creation, services orchestration, services optimization, services management, services marketing, business process integration and management; 3) cloud services featuring modeling, developing, publishing, monitoring, managing, delivering XaaS (everything as a service) in the context of various types of cloud environments; and 4) mobile services featuring development, publication, discovery, orchestration, invocation, testing, delivery, and certification of mobile applications and services.

Topics
The International Journal of Services Computing (IJSC) covers state-of-the-art technologies and best practices of Services Computing, as well as emerging standards and research topics which would define the future of Services Computing. Topics of interest include, but are not limited to, the following:

- Services Engineering
- XaaS (everything as a service)
- Cloud Computing for Internet-based services
- Big Data services
- Internet of Things (IoT) services
- Pervasive and Mobile services
- Social Networks and Services
- Wearable services
- Web 2.0 and Web X.0 in Web services
- Service-Oriented Architecture (SOA)
- RESTful Web Services
- Service modeling and publishing
- Service discovery, composition, and recommendation
- Service operations, management, and governance
- Services validation and testing
- Service privacy, security, and trust
- Service deployment and evolution
- Semantic Web services
- Scientific workflows
- Business Process Integration and management
- Service applications and implementations
- Business intelligence, analytics and economics for Services
Call for Articles
International Journal of Big Data

Mission
Big Data has become a valuable resource and mechanism for the practitioners and researchers to explore the value of data sets in all kinds of business scenarios and scientific investigations. New computing platforms such as cloud computing, mobile Internet, social network are driving the innovations of big data. From government initiative perspective, Obama Administration in United States launched "Big Data" initiative that announces $200 Million in new R&D investments on March 29, 2012. European Union also announced "Big Data at your service" on July 25, 2012. From industry perspective, IBM, SAP, Oracle, Google, Microsoft, Yahoo, and other leading software and internet service companies have also launched their own innovation initiatives around big data.

The International Journal of Big Data (IJBD) aims to provide the first Open Access publication channel for all authors working in the field of all aspects of Big Data. Big Data is a dynamic discipline. One of the objectives of IJBD is to promote research accomplishments and new directions. Therefore, IJBD welcomes special issues in any emerging areas of big data.

Topics
IJBD includes topics related to the advancements in the state of the art standards and practices of Big Data, as well as emerging research topics which are going to define the future of Big Data. Topics of interest to include, but are not limited to, the following:

**Big Data Models and Algorithms** (Foundational Models for Big Data, Algorithms and Programming Techniques for Big Data Processing, Big Data Analytics and Metrics, Representation Formats for Multimedia Big Data)

**Big Data Architectures** (Cloud Computing Techniques for Big Data, Big Data as a Service, Big Data Open Platforms, Big Data in Mobile and Pervasive Computing)

**Big Data Management** (Big Data Persistence and Preservation, Big Data Quality and Provenance Control, Management Issues of Social Network enabled Big Data)

**Big Data Protection, Integrity and Privacy** (Models and Languages for Big Data Protection, Privacy Preserving Big Data Analytics Big Data Encryption)

**Security Applications of Big Data** (Anomaly Detection in Very Large Scale Systems, Collaborative Threat Detection using Big Data Analytics)

**Big Data Search and Mining** (Algorithms and Systems for Big Data Search, Distributed, and Peer-to-peer Search, Machine learning based on Big Data, Visualization Analytics for Big Data)
