Journal of Network and Computer Applications 93 (2017) 27–37

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Distributed asteroid discovery system for large astronomical data

Chi-Sheng Huang, Meng-Feng Tsai*, Po-Hsuan Huang, Li-Ding Su, Kuei-Sheng Lee

Department of Computer Science and Information Engineering, National Central University, No. 300, Zhongda Rd., Zhongli District, Taoyuan City 32001, Taiwan, ROC

ARTICLE INFO

Keywords:
Big data
Cloud computing
Distributed system
Asteroid track
Hough transform algorithm

ABSTRACT

Thanks to the advanced technology in astronomical observations, astronomers have collected massive data sets. One of the research tasks is to discover asteroids based on observational data. However, with the fast-growing volume of astronomical data sets, this task becomes enormously relied on computing power. Therefore, the cloud-enable distributed computation and super powerful computation power can offer a good solution to this field. We adopt the Hough transform to link the sequential DETECTIONs of asteroids, and also design a distributed algorithm to process this task on distributed cloud environment for flexible and efficient calculation. Our designs are developed on MapReduce and Spark frameworks. We also utilizes HDFS and HBase in order to reduce disk I/O overhead and increase reliability and scalability. The system architecture is built on the OpenStack which allocates hardware resources flexibly. Our experiment results show significant improvements of calculation time. We believe this will help astronomers to interactively access and analyze data to discover asteroids. The system can also incrementally update DETECTIONs linking with new observed data. It also provides the visual interface to inspect the linked DETECTIONs of asteroids.

1. Introduction

With the rapid development of observation technology in astronomical research projects, the Astronomical Survey Data has been soon amassed petabyte-level scale (Carilli and Rawlings, 2004; Huijse et al., 2014; Stephens et al., 2015). Thus, utilizing big data analytics becomes one of the most recent trends of astronomical research.

Astronomy is a quite varied research for natural science, from asteroid detection to cosmology. To discover asteroids, especially those potential Earth impactors, is one of the important field of astronomy. Asteroids are small objects following the Keplerian law to orbit around the Sun. Some of them are resourceful, which possess valuable materials (e.g., noble metal, water...etc.) (Ross, 2001; Elvis, 2014; Andrews et al., 2015), and some of them are dangerous, which could possible impact the Earth and cause catastrophic damages (i.e., potential hazardous asteroids, PHAs). However, the known asteroids are only a small portion of all asteroids. It is an important task to discover them and especially to identify those PHAs.

Astronomers constantly monitor the sky by taking images and measuring the brightness, positions (i.e., equatorial coordinate; Right Accession, RA, and Declination, Dec) of celestial objects. Therefore, it records these information as a function of time. Instead of staying in the same positions on the sky as stars do, asteroids move on the sky along their own tracks to create certain patterns of position on the sky as time moves on. Astronomers therefore can look for these position patterns as a function of time. Such position patterns of asteroids are typically a short straight line within a short period of time.

However, with the advance in the technology for astronomical observation, the volume of accumulated data increases very rapidly. The old-fashion way (i.e., single server computing and storage) to search for such position pattern is no longer able to deal with these avalanched data. Therefore, we adopt cloud computing (distributed computation) and greater data storage strategy to accelerate astronomers' research and overcome the main drawback of single-server computing as well as limited memory capacity.

Progressive computer networking technology also helps the rapid development of distributed systems. The concept of cloud computing is based on distributed systems. The cloud computing services such as OpenStack (2010), Hadoop (2007), Spark (2012), Storm (2012) etc. allow clients using virtualization technology to process large data and access resources efficiently through the networks of centralized management structure (Corradi et al., 2015; Yang and Chen, 2015; Díaz et al., 2016; Madni et al., 2016). Xie et al. (2010) proposed the performance problem of the Hadoop DataNode with different HDFS on heterogeneous clusters. Mesmoudi et al. (2016) designed a benchmark for structured data from astronomy where a set of defined SQL queries

* Corresponding author.
E-mail addresses: vcshuang@csie.ncu.edu.tw (C.-S. Huang), mftsai@csie.ncu.edu.tw (M.-F. Tsai), 102522006@cc.ncu.edu.tw (P.-H. Huang), 104522077@cc.ncu.edu.tw (L.-D. Su), 995402017@cc.ncu.edu.tw (K.-S. Lee).

http://dx.doi.org/10.1016/j.jnca.2017.03.013
Received 30 October 2016; Received in revised form 1 February 2017; Accepted 14 March 2017
Available online 16 March 2017

on Hive and HadoopDB. Hadoop MapReduce (Dean and Ghemawat, 2008), Spark and Storm are popular cloud computing frameworks. Each of these frameworks has its own merits. MapReduce and Spark perform batch processing of lager data in most cases. And Storm allows processing streams of data for real-time computation (Gu and Li, 2013; Basanta-Val et al., 2015, 2016). Therefore, it is proper to utilize the distributed file system in cloud computing environments to ensure continuity of data processing, considering the volume-growing astronomical data size.

In this paper, we designed an efficient system to discover asteroids from the linkage of their sequential DETECTIONs based on Hough Transform algorithm (Duda and Hart, 1972). The fundamental concept is the simplified process in comparison and possible segments alignment in Hough space. (A "DETECTION" is a signal detected in the image, it contains position, magnitude and time when it was observed.) Since original Hough Transform algorithm has high computational complexity, and storage capacity is limited on single server, it results in time-consuming and disk I/O overhead. On the other hand, both coordinates and transformed data can be partitioned and distributed calculated, so the distributed computation can take best advantage of backend cloud architecture. For solving these low performance issues, in Chen et al. (2008), Satzoda et al. (2008), Sathyanarayana et al. (2009), Chen et al. (2012), Zhou et al. (2013) and Lu et al. (2013), they proposed parallel computing methods on Hough Transform algorithms.

Instead of the parallel solution, this study proposes the Distributed Asteroid Discovery System (DADS). After preprocessing the raw data of positions of celestial objects, it uniformly partitions these data into many blocks according to its content, and link the sequential DETECTIONs of asteroids, respectively, with Hadoop MapReduce and Spark frameworks. In addition, we use Apache HBase (2009) and Hadoop Distributed File System (HDFS) which provides data storage built on the OpenStack which allocates hardware resources flexibly. With the flexibility, we are able to adjust and process different observational datasets with advanced advices in the future.

The major contribution of this research : .

(1) Adopt a distributed Hough Transform method with big data analysis for cloud computing technique to discover asteroid systematically and efficiently.
(2) Construct an open source cloud computing environment to discover asteroid using the Intermediate Palomar Transient Factory (PTF) (2008) astronomical data set, which contains the positions of celestial objects as a function of time. The system can incrementally update the new data from continuously observations, and also provides the visual interface to inspect the position-linkage for the discovered asteroids.

The paper is organized as follows. The pertinent literature and related work are reviewed in Section 2. The additive property of linear Hough Transform to look for the position-linkage of asteroids is briefly described in Section 3. The formulation of our method is given in Section 4. The experimental results are presented in Section 5. Conclusions and the future works are given in Section 6.

2. Background and related work

2.1. Discovery of Asteroid

Asteroids are dangerous, but also resourceful. Some asteroids pose threaten by impacting earth (i.e., PHAs), and however, some asteroids can be the ideal place to provide replenishment of valuable resources for space exploration (i.e., asteroid mining) (Ross, 2001; Elvis, 2014; Andrews et al., 2015). Compare to the entire asteroid population, the known asteroids are only a very small portion. Therefore, to discover asteroids is one of the most important topics in astronomy.

On contrary to the stationary sources (i.e., stars, galaxies…etc.), the displacements of asteroids can be noticed within a short period of time (i.e., minutes to hours), and typically they can be approximated by a straight line. Therefore, asteroids can be discovered by finding the DETECTIONs moving along a certain straight line in sequential images (i.e., images of the same sky area observed in different time).

Currently the most comprehensive automatic asteroid detecting software package is the Pan-STARRS Moving Object Processing System (MOPS) (Kubica et al., 2007; Denneau et al., 2013). The MOPS uses the "kd-tree" nearest neighborhood search to look for sets of DETECTIONs in sequential images that satisfy two conditions: 1. the set of DETECTIONs are nearby to each other spatially and temporally, and 2. the distances between sequential DETECTIONs are consistent with an asteroid moving at constant speed. The Pan-STARRS and the MOPS have already observed more than 240,000 asteroids during the first 15 months of its 3-year all-sky survey mission (Vere et al., 2015), and the total number of asteroids observed is continually increasing.

2.2. The Palomar Transient Factory

The Palomar Transient Factory (PTF) (2008) (Law et al., 2009; Rau et al., 2009) is an international astronomical collaboration to explore the transient and variable sky synoptically. The intermediate PTF employs the Palomar 48-in. Oschin Schmidt Telescope equipped with an 11-chip mosaic CCD camera to create a field of view of ~7.26 deg^2, which is powerful to collect time-serious astronomical data. In 2014 Feb 20–23, Chang et al. (2015) used the intermediate PTF to collect asteroid light curves (i.e., brightness of asteroid as a function of time) to measure their rotation periods. The observation was carried out with a cadence of 10 min and had a total sky coverage ~86 deg^2. In such high cadence observation, the positions of a single asteroid observed at different times would line up approximately as an obvious line. Therefore, this data set is very suitable to discover asteroids using linear Hough Transform.

2.3. The Hough Transform

The Hough Transform algorithm (Duda and Hart, 1972) is a shape detection method used to detect lines, curves, and objects for image analysis, computer vision, and digital image processing. It maps each of points in a two-dimensional spatiotemporal domain into a parameter space which used polar coordinate system. In detail, it recognizes specific patterns in a bilevel image by determining maximal accumulated crossover point in a transformed parameter space, i.e., Hough space.

In this research, we adopt the linear Hough Transform (LHT) to link the sequential DETECTIONs of asteroids obtained from images of astronomical observations. The equation of a straight line in x-y plane corresponding to the polar coordinate plane is:

$$\rho = \chi \cos\theta + y \sin\theta \quad (1)$$

It is used to detect collinear points represented by many sinusoidal curves in the Hough space as shown in Fig. 1. Points N_1, N_2 and N_3, lying on the same line segment, use the same formula of ρ-θ combination, $(\rho_\lambda, \theta_\lambda)$, which denotes that the sinusoidal curves of the collinear points cross in the polar coordinate plane.

However, the LHT is very time-consuming for the images of large size (Ambs et al., 1986; Hollitt, 2013). Accordingly, many efficient accelerating methods developed successively. In Chen et al. (2008), it proposed parallel executing the LHT implementing on multi-core processors. Satzoda et al. (2008) and Sathyanarayana et al. (2009) studied the additive property of the LHT and introduced a grid-based framework for parallel computing the LHT through dividing images uniformly. Additionally, in Chen et al. (2012), Zhou et al. (2013) and Lu et al. (2013), utilizing embedded application parallelized the LHT implementing on the FPGA architecture.

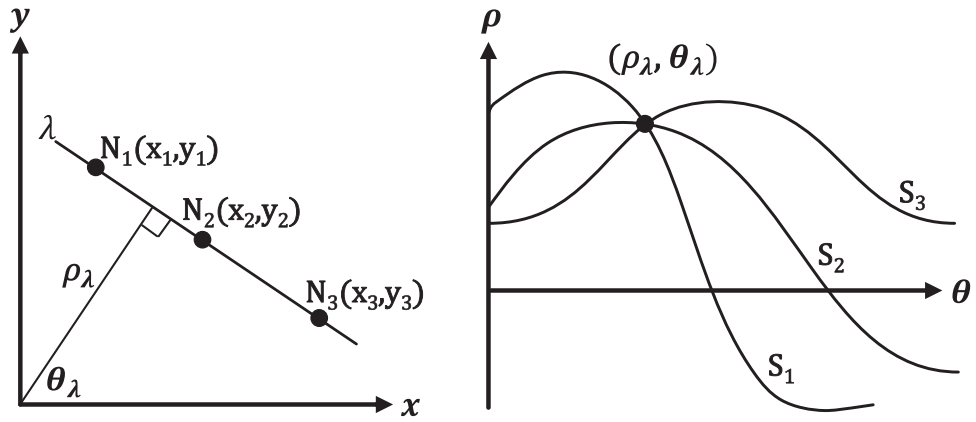

Fig. 1. Line Hough Transform.

3. Methodology

We are going to explain our methodology. Since the LHT algorithm has high computational complexity, Satzoda et al. (2008), Sathyanarayana et al. (2009) discussed the additive property of the LHT for parallel computing. Therefore, we extend the framework of distributed computing to link the lining up sequential DETECTIONs in a straight line to discover asteroids using the LHT.

Basically, two or more points can be considered as parts of one track candidate, if each segment formed by those points has the same perpendicular distance and angle. And the two parameters can be directly compared based on the transformed value. Based on this property, all possible segments can be compared and linked.

In the view of the LHT algorithm, as seen in Fig. 2, $\rho_O(A, \theta)$ represents the distance from the origin O perpendicular to the line L2 passing through the point A and makes an angle of θ with the positive x-axis. Concerning the point B, the distance of the orthogonal projection onto L2 from B is $\rho_B(A, \theta)$. Consider the line L1 passing through B and paralleling with L2, then $\rho_O(B, \theta)$ is the perpendicular distance to L1 from the origin O. According to the properties of parallel lines, it's apparent that the corresponding angles are equal for all projections namely $\rho_O(A, \theta)$, $\rho_B(A, \theta)$and$\rho_O(B, \theta)$. Consequently, the following equation can be derived:

$$\rho_O(\mathbf{A}, \boldsymbol{\theta}) = \rho_B(\mathbf{A}, \quad \boldsymbol{\theta}) + \rho_O(\mathbf{B}, \boldsymbol{\theta}) \tag{2}$$

We anticipate a huge number of possible segments to be calculated, so we will try to partition the data set and use distributed cloud architecture to provide greater computation power. The kernel issue of designing a distributed algorithm is the additive property as explained below:

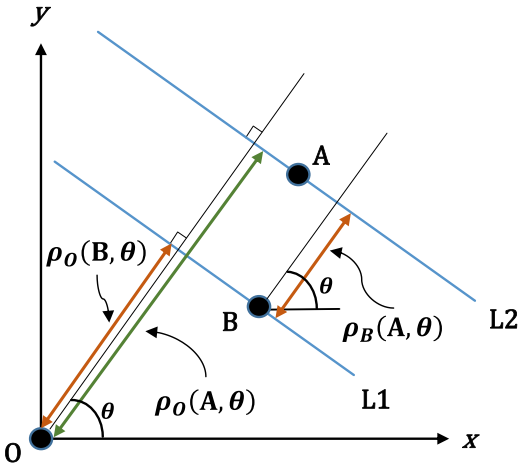

Fig. 2. Illustrating the additive property of the LHT.

For each distance from the origin O perpendicular to all lines through the point A that makes different angles counterclockwise from the positive x-axis, it could generalize the relation on the set, namely the additive property of the LHT:

$$\mathbf{LHT}(\mathbf{A}, \mathbf{O}) = \mathbf{LHT}(\mathbf{A}, \mathbf{B}) + \mathbf{LHT}(\mathbf{B}, \mathbf{O}) \tag{3}$$

The PTF astronomical observed data set is recorded each entry in DETECTION units including observed time (mjd), equatorial coordinate system (RA, Dec), etc. Every equatorial coordinate (RA, Dec) could be treated as a position of a point (x, y) in two-dimensional spatio-temporal space. In Fig. 3, we divide all DETECTIONs into n-by-n partitions of the same size (e.g. $P_0 \sim P_{n^2-1}$) and then divide each partition into several cells depending on how the precision of (RA, Dec) values recorded. The equatorial coordinate (RA, Dec) could specify which cell and partition the DETECTION located in. Hence, according to the additive property of the LHT (3), for each equatorial coordinate of DETECTIONs relative to the global origin O, LHT(DETECTION,O_{global}), it could use substitution to obtain the relation:

$$\mathbf{LHT}(\mathbf{DETECTION}, \mathbf{O_{global}}) = \mathbf{LHT}(\mathbf{DETECTION}, \mathbf{O_{local}}) + \mathbf{LHT}(\mathbf{O_{local}}, \mathbf{O_{global}}) \tag{4}$$

O_{global} denotes the global origin of the whole partitions. In this example, that is the local origin of the most bottom-left partition (P_0). O_{local} denotes the most bottom-left cell of each partition of DETECTIONs. For instance, in Fig. 3, the LHT that DETECTION U is at the cell of P_5 with respect to its local origin O_5, is equivalent to the

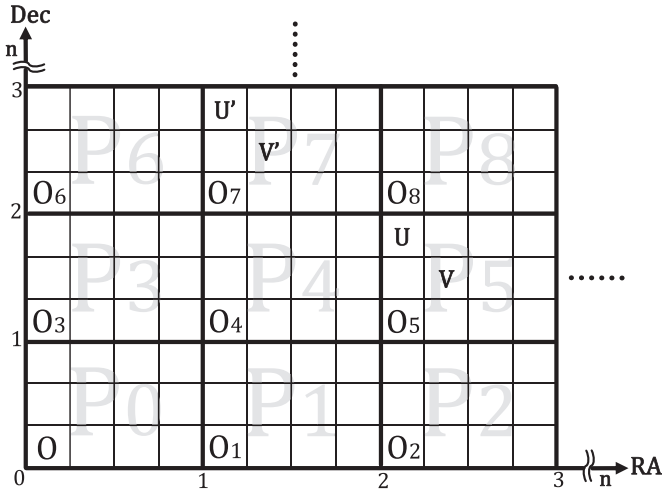

Fig. 3. The n-by-n partitions of DETECTIONs by RA-Dec coordinate.

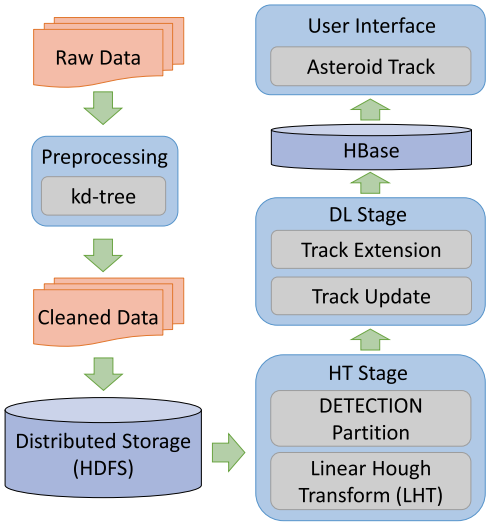

Fig. 4. The DADS flow diagram.

LHT that DETECTION U′ is at the cell of P_7 with respect to its local origin O_7. For the same reason, as V is to O_5, so is V′ to O_7.

Consequently, it needs to calculate the LHT of all cells in a partition with respect to its local origin only once as a lookup table, and then calculate the LHT of all local origin of each partitions with respect to the global origin. Taking advantage of the additive property of the LHT, it could compute the LHT of all DETECTIONs with less computational costs.

4. DADS

We propose the DADS to deal with large observed asteroid tracks. Fig. 4 briefly expresses the flow diagram of the proposed system.

The astronomical data set (PTF) records the information of celestial objects, including position and brightness, as a function of time. The position (RA, Dec) is in equatorial coordinate system, and the observing time is usually expressed as modified Julian day (mjd). Before to link sequential DETECTIONs of asteroids, the DETECTIONs of stationary sources are removed from the data set first. The left panel of Fig. 5 shows the DETECTIONs of non-stationary sources obtained from sequential images, where we see the DETECTIONs of asteroids lining up along straight lines with time-varied color points. However, we also see noise DETECTIONs randomly distributed in the left panel of Fig. 5, therefore, we apply kd-tree involving nearest neighbor searches (NNS) to filter out these noise DETECTIONs to increase the efficiency of the LHT linkage. The right panel of Fig. 5 shows the sequential DETECTIONs after filtering out the noise DETECTIONs (hereafter, the clean DETECTIONs), and the lining-up sequential DETECTIONs of asteroids become more obvious. All the cleaned DETECTIONs are then stored in distributed storage (HDFS).

In the Hough Transform Stage (HT Stage), all cleaned DETECTIONs from HDFS is partitioned via RA and Dec of each DETECTION. We then utilize additive property of the LHT to distributedly perform LHT algorithm to obtain all perpendicular distances and angles of the all DETECTIONs. In the DETECTION-Linking Stage (DL Stage), we classify all DETECTIONs into different tasks determined by their common perpendicular distances and angles. According to the time, possible tracks are extended and updated. Finally, the calculated track results are stored into NoSQL database of HBase. Astronomers could further query and research celestial events through the user interface.

4.1. Preprocessing by the kd-tree

We utilize kd-tree to filter out noise DETECTIONs. Algorithm 1 shows the procedure of the kd-tree. First, we construct the kd-tree by every (RA, Dec) pairs. Then, we use the NNS for every DETECTION to do range query and find the other DETECTIONs within the range and record them. We set the pair_range, which is a lower bound on the possible asteroid moving distance, between any pair of DETECTIONs. If a DETECTION does not reach the least_neighbor_threshold (empirical setting) within the lowest limit of range or more than one DETECTION within the stationary_range (the DETECTION within stationary_range means that it does not belong to the asteroid belt, or it may be the noise), we can determine that the DETECTION is not generated by an asteroid and we can remove it.

Algorithm 1. : Preprocessing by the kd-tree.

```
Input: DETECTIONs
Output: clean DETECTIONs
1:  Tree = kdtree(RA[ ], Dec[ ])
2:  pair[DETECTION] = Tree.range_query(pair_ range)
3:  stationary[DETECTION] = Tree.range_query(stationary_range)
4:      bool mask[length(RA)]
5:      for i=0 to length(RA):
6:          if length(pair[i]) > least_neighbor_threshold and length(stationary[i])=1
7:            keep the DETECTION
8:          else
9:            clean the DETECTION
10:         endif
11:     endfor
12: output clean DETECTION
```

4.2. HT Stage

The initial Hough transform is conducted in this stage. All cleaned DETECTIONs were used to compute maximum boundary. We split all DETECTIONs into n-by-n partitions (n is initialized empirically) of the same size and then divide each partition into cells depending on the precision of positions (i.e., RA, Dec). To reduce the following computational costs, it needs to calculate the LHT of all cells in a partition with respect to its local origin once, and prepare a corresponding lookup table. The table is stored in the DistributedCache.

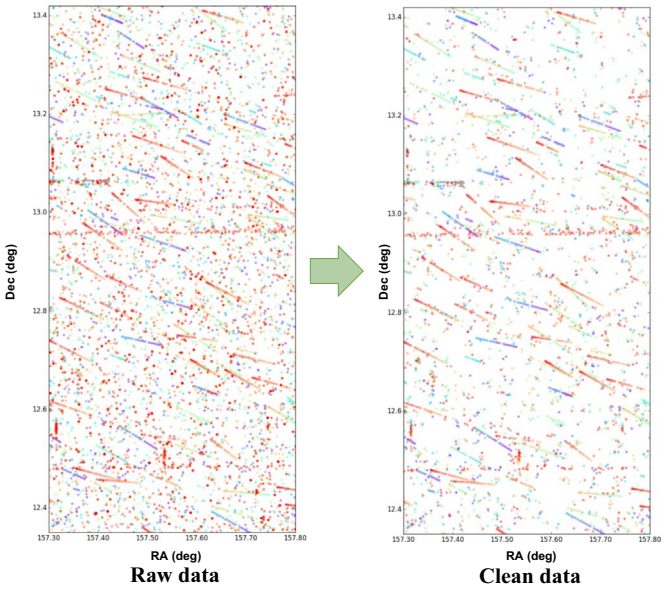

Fig. 5. The outputs of preprocessing by the kd-tree.

After preparing DETECTIONs partitioning and the lookup table of the LHT for local cells in a partition, it executes the additive LHT procedure of DETECTIONs, as we show in Algorithm 2. First, compute the LHT of O_{local} in the input partition with respect to the O_{global} on specific range of angles (RT_{lg}). Then, read values of the LHT of each DETECTION in the partition with respect to its local origin from the DistributedCache (RT_{dl}), and compute the additive LHT of these DETECTIONs on specific range of angles by adding both RT_{lg} and RT_{dl}. In addition, it defines the Line ID with coded information. The coded information include the perpendicular distances of these DETECTIONs to O_{global} as well as each specific angle. Finally, output each Line ID and its corresponding DETECTION as a tuple respectively. We will further utilize the results to sort all DETECTIONs in the next stage.

Algorithm 2. : HT Stage.

Input: <DETECTIONs>: a set of DETECTIONs in a certain partition, set of angle, O_{local}, O_{global}
Output: < key: Line ID, value: DETECTION >
1: Array[angle.size] RT_{lg}
2: Array[angle.size] RT_{dl}
3: RT_{lg} = LHT (O_{local}, O_{global}, angle)
4: **while** DETECTIONs has next **then**
5: d = DETECTIONs.next()
6: RT_{dl} = get LHT(d, O_{local}, angle) from DistributedCache
7: **for**(i = 0, i < angle.size, i++)
8: LHT = RT_{lg} [i] + RT_{dl} [i]
9: convert LHT to Line ID
10: **output** < Line ID, d >
11: **endfor**
12: **endwhile**

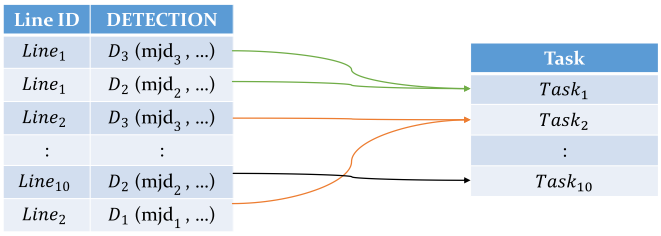

Fig. 6. Distribute the tasks to proceed.

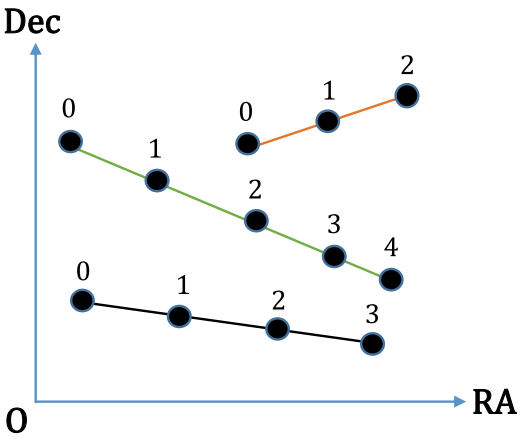

Fig. 7. The result of the DL Stage.

LinkagesMetadata. The table Linkages contains the DETECTIONs with their Line ID and the index represented the position with respect to its linkage line. The table LinkagesMetadata contains the Line IDs and the amount of the DETECTIONs on their linkage respectively.

Each task would conduct the Algorithm 3. At first, the DETECTIONs are arranged by the observed time (mjd) in ascending order. If the tuple with the Line ID exists in the table LinkagesMetadata, it is going to update the track Line ID; otherwise, it would extend the track Line ID. For the part of the track extending, it specifies the serial numbers to the DETECTIONs on the track in accordance with the order of time. Then, it constructs the form of < key, value > tuples. The key named LineID_SN is a string combining the Line ID with the serial number of DETECTIONs on the track, and the mapping value of the corresponding DETECTION. It stores these tuples into the table Linkages and update the value of the amount of the DETECTIONs on the track Line ID in the table LinkagesMetadata.

For the part of the tracks updating, we obtain the amount of DETECTIONs on the track Line ID from the table LinkagesMetadata in

4.3. DL Stage

In the DL Stage, all track segments are compared and merged if possible. We categorize the outputs of the HT Stage into different tasks according to their Line ID. The DETECTIONs belonging to the same line (i.e., same Line ID) will be distributed to the same task as shown in Fig. 6, and then, they are extended and updated. There are two tables we created in HBase: the table Linkages and the table

the first place. It then reads all DETECTIONs on this track by the LineID_SN in ascending order of time (mjd) from the table Linkages. To save time, instead of querying the entire table, the task will be performed distributedly. We would determine, on the track Line ID, the correct positions of the newer DETECTIONs which we want to update to the table Linkages by using the binary search algorithm via comparison of time. Nevertheless, it would ignore the newer DETECTION when its mjd collides. Finally, update all LineID_SN, and then store these tuples into the table Linkages. On the same time, update the amount of the DETECTIONs on the track Line ID in the table LinkagesMetadata. Fig. 7 shows that the results of the DL Stage. The asteroids would extend along the tracks of the same Line ID.

Algorithm 3. : DL Stage.

```
Input: < D_Set >: a set of DETECTIONs with same Line ID, Table Linkages, Table
       LinkagesMetadata
Output: Table Linkages, Table LinkagesMetadata
Tuple:< LineID_SN , DETECTION >
List < Tuple >Track
Sort < D_Set > by DETECTION.mjd in ascending order
If Table LinkagesMetadata is empty then
    Put Line ID and D_Set.size as Size into Table LinkagesMetadata
    for(i = 0, i < D_Set.size, i++)
        tuple.DETECTION = D_Set[i]
        tuple.LineID_SN = strcat(Line ID, i)   // strcat(): concatenate two strings
        Track.add(tuple)
    endfor
else
    size = get from Table LinkagesMetadata by Line ID
    for(i = 0, i < size; i++)
        LineID_SN = strcat(Line ID, i)
        Tuple t = get tuple from Table Linkages by LineID_SN
        Track.add(t)
    endfor
    for(i = 0, i < D_Set.size, i++)
        index = BinarySearch(D_Set[i], Track)
        tuple.DETECTION = D_Set[i]
        tuple.LineID_SN = strcat(Line ID, index)
        Track.insert(index, tuple)
    endfor
    Update all LineID_SN in List Track
    Update Line ID and Track.size as Size into Table LinkagesMetadata
endif
output Track into Table Linkages
```

5. Experiments

5.1. System architecture of DADS

This work proposes the DADS to calculate asteroid track of the large observational data and help astronomers to understand some astronomical phenomena. It divides astronomical data into many blocks uniformly according to its content and asteroid tracks are computed respectively. The system is built with Hadoop MapReduce (ver. 2.6.0) and Spark (ver. 1.6.0) frameworks on YARN (Vavilapalli et al., 2013) which schedules jobs and manages resources. In addition, we use the Apache HBase (ver. 0.98.12.1) and HDFS which provides data storage and we also build multi-node OpenStack virtual computing clusters, as Fig. 8 shows.

The Apache Hadoop project develops open-source software and implements MapReduce in Java. Hadoop is a two-layered structure composed of HDFS (data storage layer) and Hadoop MapReduce Framework on YARN (data processing layer). The Apache Spark is an open-source cluster computing framework. It provides the fundamental data abstraction called Resilient Distributed Dataset (RDD) (Zaharia et al., 2012). Spark uses HDFS and HBase as its input source and output destination without using its own distributed file system.

5.2. Experimental environments

Our experiments are performed on two environments —single machine and OpenStack distributed clusters. The single machine is equipped with Intel Core i7-2600 3.40 GHz CPU, 16 GB RAM, 1 TB HDD, and running Microsoft Windows 7 Enterprise.

The distributed clusters run the OpenStack (JUNO) platform and their equipment listed below:

Controller Node: Intel Core i7-870 2.93 GHz processor at 8-cores, 6 GB RAM and 500 GB HDD, connected through one 1000 Mbps LAN adapter.

Network Node: Intel Core 2 Quad Q9500 2.83 GHz processor at 4-cores, 4 GB RAM and 500 GB HDD, connected through three 1000 Mbps LAN adapters.

Compute Node: The physical and virtual environments are shown in Tables 1 and 2. Each physical machine is connected through two 1000 Mbps LAN adapters. Each Hadoop node consists of a virtual machine (VM) equipped with 2VCPU, 8 GB RAM and 100 GB HDD. We use 11 Hadoop Nodes, one of which is the NameNode as the resource manager, and the others are DataNodes as the task computing.

Both physical and virtual machines run Linux Ubuntu 14.04.

This work employed experimental data sets (1.2 million records)

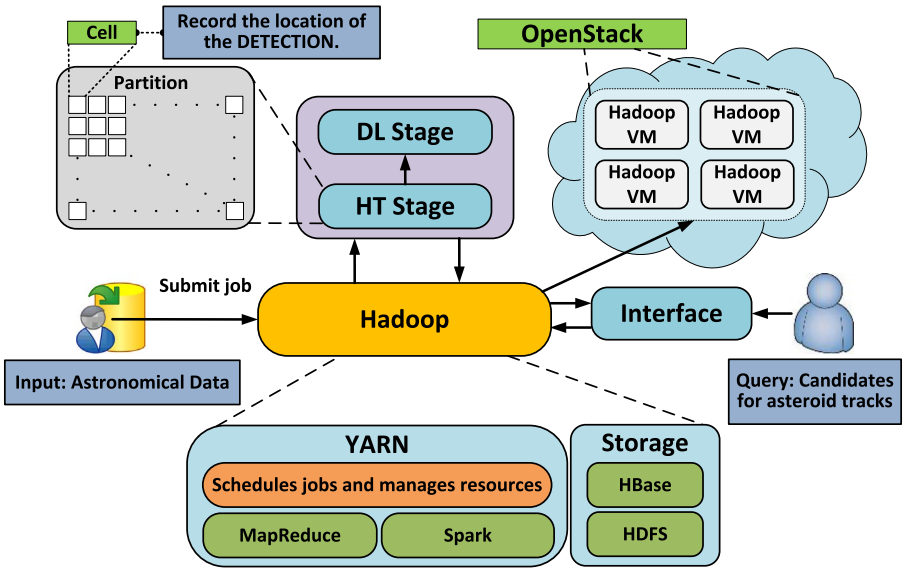

Fig. 8. System architecture of DADS.

Table 1
Equipment of Compute Node.

OpenStack Compute Node			
Node name	**Compute1**	**Compute2**	**Compute3**
CPU	Intel i7–3770	Intel i7–3770	Intel i7–4790
Memory	32 GB	32 GB	32 GB
HDD	500 GB	500 GB	1TB

Table 2
Equipment of Hadoop Node.

Hadoop node (VM)		
Framework	**MapReduce (Each Node)**	**Spark (Each Node)**
CPU	2 VCPU	2 VCPU
Memory	8 GB	8 GB
HDD	100 GB	100 GB

extracted from the Palomar Transient Factory (PTF) astronomical observed data. We split the original data in three kinds of scales such as one-third, one-half and all of the original data for experiments, i.e. |D|=0.4 million, 0.6 million and 1.2 million, respectively, where |D| denotes the size of the experimental data set. We compare four different numbers of Hadoop DataNodes (4 Nodes、6 Nodes、8 Nodes、10 Nodes) on three different size of data sets.

5.3. Experimental results

In order to implement track extension of asteroid in the DADS, we need to calculate each tracks through HT Stage and DL Stage, then store the results of asteroid tracks into MySQL while we implement our experiments in single machine. For Hadoop distributed computing framework like MapReduce and Spark, we use HDFS and HBase for storage.

In Fig. 9(a), the total execution time of single machine for operating the HT Stage and the DL Stage with 1.2 million records is 10076.72 min. In Fig. 10, for the comparison of execution time with 10 nodes, it took 214.65 min to operate both stage using MapReduce, but it only need 112.11 min using Spark. Applying both MapReduce and Spark outperform single machine by 47x and 90x respectively. The most efficient case to calculate the data using Spark is with 8 nodes. MapReduce performs well in 6 nodes through 0.6 million records. For the other sets of records, however, it leads to good performance in 8 nodes.

Implementing the additive LHT in the DADS to estimate the asteroid track must include the execution time of both HT Stage and DL Stage, as well as the extra time of network transmission through different nodes. In order to estimate the efficiency of additive LHT implemented on MapReduce and Spark frameworks, we perform a set of experiments which compare the execution time of four different numbers of Hadoop DataNodes on three different size of data sets, as shown in Fig. 11. In Fig. 9(b), the total execution time of single machine for operating HT Stage with 1.2 million records is 227.30 min. In Fig. 11, the execution time using Spark decrease by around 30 min in comparison with those using MapReduce in condition of 10 nodes.

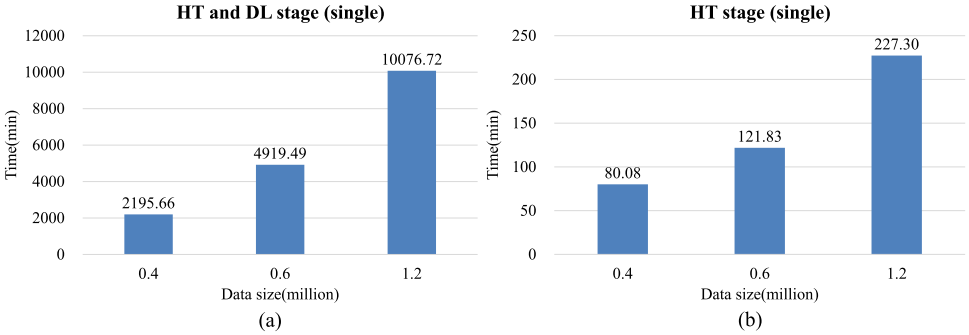

Fig. 9. Experiments of single machine.

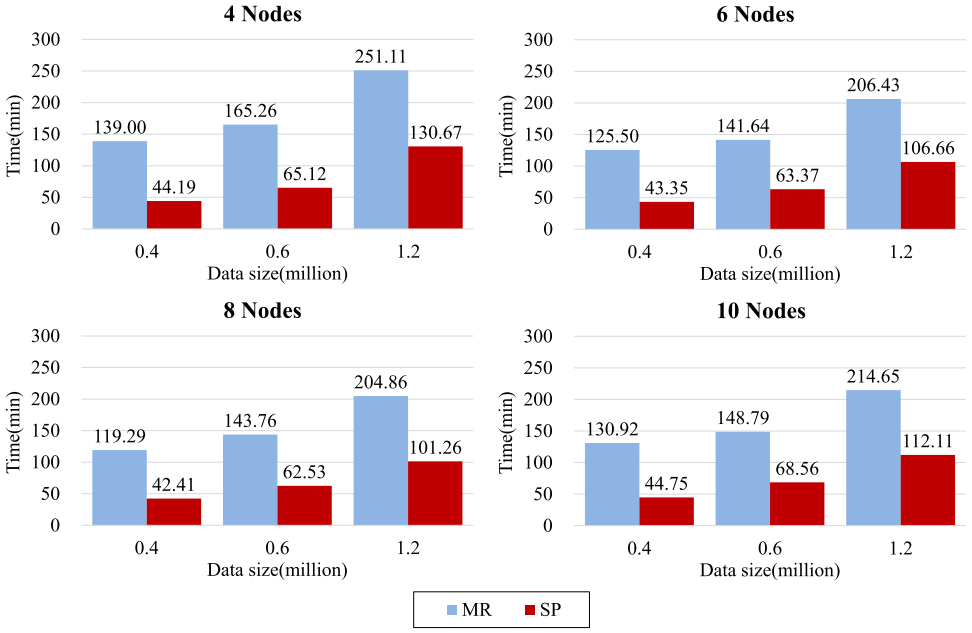

Fig. 10. Experiments the HT Stage and the DL Stage of MapReduce and Spark frameworks.

In this stage, both MapReduce and Spark outperform single machine by 6x and 32x respectively. In the case of Spark in 10 nodes, the execution time is the lowest. MapReduce gets the best result in 8 nodes through 0.4 million records and it results in good performance in 6 nodes using the other sets of records.

The total execution time of tracks updating in DL Stage is the addition of the tracks extension time and updating time in HBase. To estimate the efficiency of calculation in track updating, we set 0.4 million records for its database in HBase after track extension, then input 0.6 and 1.2 million records to compare the original ones respectively and discuss the efficiency of asteroid track updating. In tracks updating, we must extract the existed track data with sorted DETECTIONs in HBase, then insert the new DETECTION by position. The position is accessed by binary searching via comparison of mjd. In Fig. 12, the efficiency of track updating in 8 Nodes outperforms the other numbers of Nodes on any size of data sets. Since RDDs are less suitable for asynchronous finegrained updates to shared state, we cannot implement track updating in HBase through Spark (Zaharia et al., 2012).

In the experimental results of HT Stage and DL Stage, we can see that the processing speed of Spark outperforms both MapReduce and single machine in different numbers of Hadoop Nodes and data sizes. In Fig. 10, due to restricted hardware resources, the efficiency of tracks updating in 8 Nodes outperforms 10 Nodes. When the numbers of Hadoop Nodes increase, the needs to read and write data records simultaneously also increase. It may cause longer execution time and increasing data reloading from disk during the operations. Although the management of virtualization technology in cloud provides a flexible use of system resources, the system I/O processing and virtual node computing are still the primary causes of delay. Similar situations are also discussed in (Ibrahim et al., 2009; Moon et al., 2014; Kralevska et al., 2016). In the experimental results of the HT Stage, Spark

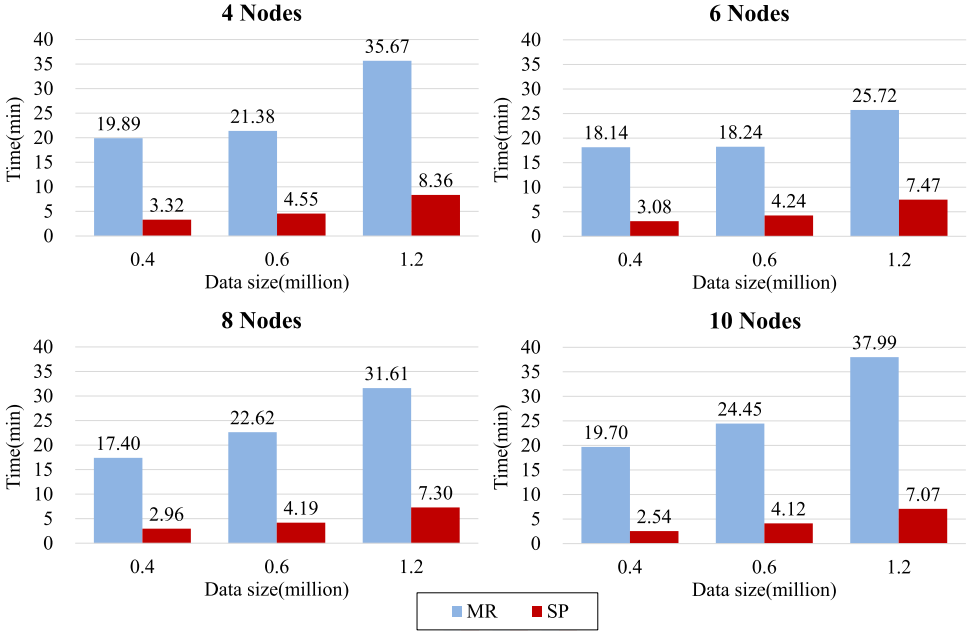

Fig. 11. Experiments the HT Stage of MapReduce and Spark frameworks.

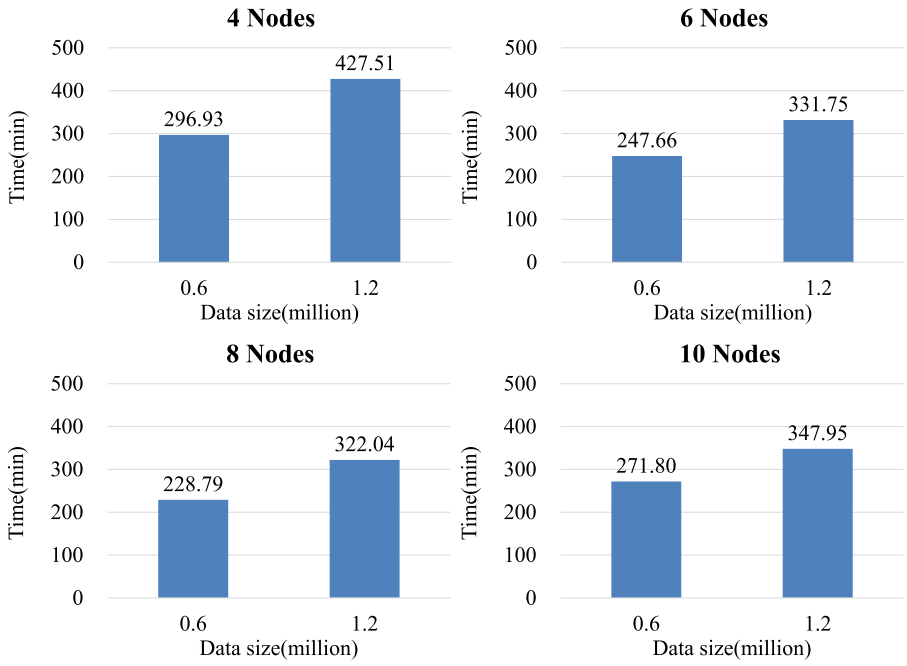

Fig. 12. Execution time of HT Stage and DL Stage with tracks update on MapReduce framework.

outperforms MapReduce. So we calculate the records in the HT Stage based on Spark in DADS. Though we cannot implement track updating in the DL Stage based on Spark, we use MapReduce to extend and update the tracks of the asteroid as a part of an efficient system.

5.4. Visual user interface

After processing PTF astronomical observed data set through both HT stage and DL Stage, we can extract approximately 0.45 million asteroid track candidates from inputs of 1.2 million records in the data set. In DADS user interface, each asteroid track has its own Line ID to help user finding the asteroid through its track. The upper left of user interface, input Line ID (or r: perpendicular distance and theta: angle) and the length threshold which determines the existence of a track when the amount of DETECTIONs on it is larger than the threshold. Fig. 13 shows the results of querying asteroid tracks on the interface. Each asteroid track will be represented in different coloured line segment consisted of many DETECTION points on the RA-Dec plane. We can check the DETECTION of point in each asteroid track (clicks the point and show the box). The DADS will support the astronomers to query and research asteroid tracks via this user interface.

6. Conclusions

We designed an efficient system called DADS to calculating possible asteroid tracks. The design is based on Hough Transform algorithm and implemented on cloud computing environments. Both MapReduce and Spark frameworks are adapted to manipulate a large amount of astronomical data to solve the problem of insufficient memory.

Comparing to the method of single machine, the experiment results show that our method can improve the efficiency significantly. With this system, astronomers can interactively access and analyze data from asteroid tracks. The new observed data can be incrementally updated and the system also provides the visual interface for observing the

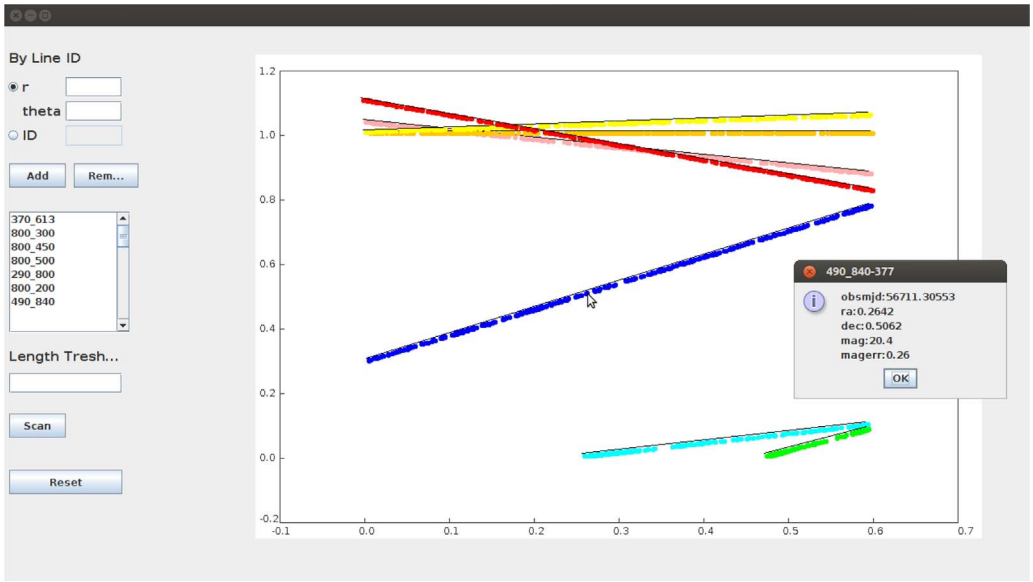

Fig. 13. Visual user interface.

moving of asteroid. Due to the characteristic of cloud computing environments, we can increase or decrease computing nodes flexibly. The DADS consists of three operation stages: Preprocessing, HT Stage and DL Stage. In response to the variable requirements and the development of the new distributed computing platforms in the future, we could improve the methods in each stage individually. It provides better maintainability through the way of stratification in our system.

In future work, we found that the frequency of data access through I/O in the DL Stage will lead to serious time-consuming processes while we extend and update asteroid tracks. To solve this problem, we can implement a distribution file system for storage indexed by data locality to achieve high efficiency. In a different aspect, we are considering crossmatching different asteroid track data of astronomical observatories to integrate and analyze the tracks of asteroid entirely.

Acknowledgements

This research was sponsored by the Ministry of Science and Technology, Taiwan under the Contract No. MOST 104-2221-E-008-076. We are grateful to the Palomar Transient Factory (PTF) for providing Astronomical Survey Data. Also, a special thanks goes to the researchers, Chan-Kao Chang and Hsing-Wen Lin, in the Institute of Astronomy, National Central University, Taiwan for supporting this study.

References

Ambs, P., Lee, S.H., Tian, Q., Fainman, Y., 1986. Optical implementation of the Hough transform by a matrix of holograms. Appl. Opt. 25, 4039–4045.

Andrews, D.G., Bonner, K.D., Butterworth, A.W., Calvert, H.R., Dagang, B.R., et al., 2015. Defining a successful commercial asteroid mining program. Acta Astronaut. 108, 106–118.

Apache HBase, 2009. [Online]. Available: ⟨http://hbase.apache.org/⟩.

Basanta-Val, P., Fernández García, N., Wellings, A.J., Audsley, N.C., 2015. Improving the predictability of distributed stream processors. Future Gener. Comput. Syst. 52, 22–36.

Basanta-Val, P., Audsley, N.C., Wellings, A.J., Gray, I., Fernandez-Garcia, N., 2016. Architecting Time-Critical Big-Data Systems. IEEE Trans. Big Data 2, 310–324.

Carilli, C.L., Rawlings, S., 2004. Science with the square kilometre array: motivation, key science projects, standards and assumptions. New Astron. Rev. 48, 979–984.

Chang, C.-K., Ip, W.-H., Lin, H.-W., Cheng, Y.-C., Ngeow, C.-C., et al., 2015. Asteroid Spin-rate Study Using the Intermediate Palomar Transient Factory. Astrophys. J. 219, 2.

Chen, Y., Li, W., Li, J., Wang, T., 2008. Novel parallel Hough Transform on multi-core processors. *Proceedings of 2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, p. 1457–1460.

Chen, Z.H., Su, A.W.Y., Sun, M.T., 2012. Resource-efficient FPGA architecture and implementation of hough transform. IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 20, 1419–1428.

Corradi, A., Foschini, L., Pipolo, V., Pernafini, A., 2015. Elastic provisioning of virtual Hadoop clusters in OpenStack-based clouds. *Proceedings of the IEEE International Conference on Communication Workshop (ICCW)*, p. 1914–1920.

Dean, J., Ghemawat, S., 2008. MapReduce: simplified data processing on large clusters. Commun. ACM 51 (1), 107–113.

Denneau, L., Jedicke, R., Grav, T., Granvik, M., Kubica, J., et al., 2013. The Pan-STARRS moving object processing System. Astron. Soc. Pac. 125, 357–395.

Díaz, M., Martín, C., Rubio, B., 2016. State-of-the-art, challenges, and open issues in the integration of Internet of things and cloud computing. J. Netw. Comput. Appl. 67, 99–117.

Duda, R., Hart, P., 1972. Use of the Hough transformation to detect lines and curves in pictures. Commun. ACM CACM Homepage Arch. 15, 11–15.

Elvis, M., 2014. How many ore-bearing asteroids? Planet. Space Sci. 91, 20–26.

Gu, L., Li, H., 2013. Memory or time: performance evaluation for iterative operation on hadoop and spark. *High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing (HPCC_EUC), 2013 IEEE In: Proceedings of the 10th International Conference on. IEEE, p.* 721–727.

Hollitt, C., 2013. A convolution approach to the circle Hough transform for arbitrary radius. Mach. Vis. Appl. 24 (4), 683–694.

Huijse, P., Estevez, P.A., Protopapas, P., Principe, J.C., Zegers, P., 2014. Computational intelligence challenges and applications on large-scale astronomical time series databases. IEEE Comput. Intell. Mag. 9, 27–39.

Ibrahim, S., Jin, H., Lu, L., Qi, L., Wu, S., Shi, X., 2009. Evaluating mapreduce on Virtual machines: the Hadoop case. Cloud Comput. Lect. Notes Comput. Sci. 5931, 519–528.

Kralevska K., Gligoroski D., Øverby H., 2016. Balanced locally repairable codes. *Proceedings of International Symposium on Turbo Codes and Iterative Information Processing (ISTC)*, p. 280-284.

Kubica, J., Denneau, L., Grav, T., Heasley, J., Jedicke, R., et al., 2007. Efficient intra-and inter-night linking of asteroid detections using kd-trees. Icarus 189, 151–168.

Law, N.M., Kulkarni, S.R., Dekany, R.G., Ofek, E.O., Quimby, R.M., et al., 2009. The Palomar Transient Factory: system Overview Performance, and First Results 121. Astronomical Society of the Pacific, 1395–1408.

Lu, X., Song, L., Shen, S., He, K., Yu, S., Ling, N., 2013. Parallel Hough transform-based straight line detection and its FPGA implementation in embedded vision. Sensors 13, 9223–9247.

Madni, S.H.H., Latiff, M.S.A., Coulibaly, Y., Abdulhamid, S.M., 2016. Resource scheduling for infrastructure as a service (IaaS) in cloud computing: challenges and opportunities. J. Netw. Comput. Appl. 68, 173–200.

Mesmoudi, A., Hacid, M.-S., Toumani, F., 2016. Benchmarking SQL on MapReduce systems using large astronomy databases. Distrib. Parallel Databases 34, 347–378.

Moon, S., Lee, J., Kee, Y.S., 2014. Introducing SSDs to the Hadoop MapReduce Framework. *Proceedings of the 2014 IEEE International Conference on Cloud Computing (CLOUD)*, p. 272-279.

OpenStack, 2010. [Online]. Available: ⟨https://www.openstack.org/⟩.

Rau, A., Kulkarni, S.R., Law, N.M., Bloom, J.S., Ciardi, D., et al., 2009. Exploring the Optical Transient Sky with the Palomar Transient Factory 121. Astronomical Society of the Pacific, 1334–1351.

Ross, S.D., 2001. Near-earth asteroid mining. Department of Control and Dynamical Systems.

Sathyanarayana, S.S., Satzoda, R.K., Srikanthan, T., 2009. Exploiting Inherent Parallelisms for Accelerating Linear Hough Transform. IEEE Trans. Image Process. 18, 2255–2264.

Satzoda, R.K., Suchitra, S., Srikanthan, T., 2008. Parallelizing the Hough Transform Computation. IEEE Signal Process. Lett. 15, 297–300.

Stephens, Z.D., Lee, S.Y., Faghri, F., Campbell, R.H., Zhai, C., et al., 2015. Big Data: astronomical or genomical? PLoS Biol., 13.

Vavilapalli, V.K., Murthy, A.C., Douglas, C., Agarwal, S., Konar, M., , et al.2013. Apache hadoop yarn: Yet another resource negotiator. *In: Proceedings of the 4th annual Symposium on Cloud Computing*.

Vere, P., Jedicke, R., Fitzsimmons, A., Denneau, L., Granvik, M., 2015. Absolute magnitudes and slope parameters for 250,000 asteroids observed by Pan-STARRS PS1-Preliminary results. Icarus 261, 34–47.

Xie, J., Yin, S., Ruan, X., Ding, Z., Tian Y., , et al.2010. Improving MapReduce Performance through Data Placement in Heterogeneous Hadoop Clusters. *Proceedings of 2010 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), p.* 1-9.

Yang, S.J., Chen, Y.R., 2015. Design adaptive task allocation scheduler to improve mapreduce performance in heterogeneous clouds. J. Netw. Comput. Appl. 57, 61–70.

Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., , et al.2012. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. *In: Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, p. 15-28.

Zhou, X., Tomagou, N., Ito, Y., Nakano, K., 2013. Efficient Hough transform on the FPGA using DSP slices and block RAMs. *Proceedings of Workshop on Advances in Parallel and Distributed Computational Models*, p. 771–778.

Chi-Sheng Huang, currently is a Ph.D. student at the Department of Computer Science and Information Engineering, National Central University, Taiwan. His research interests include data mining, big data analysis and cloud computation.

Meng-Feng Tsai, was born in Taipei, Taiwan, on Sept. 27, 1967. He received the Ph.D. degree in Computer Science from UCLA, USA, in February 2004. He then joined the Department of Computer Science and Information Engineering, National Central University, Taiwan, as Assistant Professor in 2004. His current research interests include data warehouse, database systems, data mining, scientific computation, distributed and parallel computation.

Po-Hsuan Huang, graduated from National Central University, Taiwan, Taiwan and received M.S. in Computer Science and Information Engineering in 2015. His research interests are cloud computing, big data analysis and management.

Li-Ding Su, currently is a graduate student at the Department of Computer Science and Information Engineering, National Central University, Taiwan. His research interests include data mining and distributed computing.

Kuei-Sheng Lee, a doctoral student of Department of Computer Science and Information Engineering, National Central University, Taiwan. He has an interesting in Big Data analysis and computing.