Received: 2 October 2016 | Revised: 29 January 2017 | Accepted: 13 March 2017
DOI: 10.1002/cpe.4157

WILEY

SPECIAL ISSUE PAPER

Adaptive parallel Delaunay triangulation construction with dynamic pruned binary tree model in Cloud

Jiaxiang Lin[1,2] | Riqing Chen[1] | Liping Wu[1] | Zhaogang Shu[1] | Changcai Yang[1]

[1]College of Computer and Information Sciences, Fujian Agriculture and Forestry University, Fuzhou 350002, China
[2]Institute of Geographic Sciences and Natural Resources Research, CAS, Beijing 100101, China

Correspondence
Riqing Chen and Jiaxiang Lin, College of Computer and Information Sciences, Fujian Agriculture and Forestry University, Fuzhou 350002, China.
Email: riqing.chen@fafu.edu.cn; linjx@fafu.edu.cn

Funding information
National Natural Science Foundation of China, Grant/Award Number: 41401458 and 61501120; Key Laboratory of Spatial Data Mining and Information Sharing of Ministry of Education, Fuzhou University, Grant/Award Number: 2017LSDMIS01; Key Program of Science and Technology of Fujian Province, Grant/Award Number: 2015H0015; Natural Science Foundation of Fujian Province of China, Grant/Award Number: 2014J05045, 2016J05148 and 2016J01753; Science and Technology Foundation of The Education Department of Fujian Province of China, Grant/Award Number: JA12375; Youth Foundation of Fujian Agriculture and Forestry University, Grant/Award Number: k13xjj05a

Summary
The paper illustrates a parallel and distributed scheme for computing a planar Delaunay triangulation using a divide-and-conquer strategy in Cloud environment, which combines the incremental insertion algorithm and the divide-and-conquer method. The proposed hybrid algorithm for Delaunay triangulation construction is easy to be parallelized due to the dynamic pruned characteristic of the binary tree model used. Moreover, the Cloud platform decreases the communication overhead and improves data locality by making use of a data partitioning and integrating scheme offered by the map-reduce architecture. The implementation of the parallel and distributed version of the algorithm relied on a robust data structure called quad-edge, which implies the geometric relationship among the edges and vertexes adjacent. More importantly, the data are serialized easily and transmitted efficiently between different Cloud nodes; the algorithm is executed conveniently on PC clusters. We tested the parallel version of the algorithm on GeoKSCloud, a geographical knowledge service Cloud developed by our research team. Experimental results show that the proposed hybrid algorithm is efficient and competitive; it can be easily migrated and deployed in distributed and parallel computing environment, such as grid and Cloud. The parallel implementation of the hybrid algorithm has a good speed-up, while data communication is the crucial factor for the efficiency of the parallel version. Overall, the parallel version outperforms both the sequential divide-and-conquer algorithm and the sequential incremental insertion algorithm.

KEYWORDS
Delaunay TIN, distributed and parallel, cloud services, divide and conquer, incremental insertion

1 | INTRODUCTION

Delaunay triangulation (D-TIN) and its geometric dual Voronoi diagram are two significant data structures in computational geometry, which are widely used in many industry applications, such as geographical information system, digital elevation model, terrain analysis, computer graphics and computer vision, finite element analysis, and surface reconstruction.[1] With the booming development of data collection technology and computer hardware, parallel and distributed computation-based Delaunay algorithm[2–4] is becoming a hot research area in recent years. But the shortage of multicore CPU computers seriously limits the usage of parallel D-TIN. The superiority of Cloud computing over other parallel and distributed environment makes parallel D-TIN construction in Cloud become a promising way to promote the efficiency of algorithm when processing large-scale data in compute-intensive applications. In addition, in many real situations, datasets are tend to distribute in different geographical locations, so the construction of D-TIN on local and global datasets is usually required simultaneously by some practical applications.[5,6]

There are several categories of algorithms to construct D-TIN. Among which, the incremental insertion algorithm is competitive for several reasons when compared with the others.[7,8] First, it is much easier to implement the incremental algorithm (IA) because of its simplicity, and it is applicable to various datasets especially when their sizes increase dynamically. Second, if the site set is sampled with a uniform probability distribution, the expected time for each insertion is small and roughly independent of n; the processing time is then dominated by site location. For the simple walk algorithm,[9,10] to locate a site in an existing triangulation, the expected time is roughly $O(n^{\frac{1}{2}})$ for each site, so the total computational time is $O(n^{\frac{3}{2}})$.[11] As a result, the performance can be quite acceptable in many real-world applications, even when the sites are all given in advance. Third, in many practical situations, successive sites tend to be close to each other. Consequently, the edge returned in the previous call can be used by the location process as its starting point; each insertion may take roughly a constant time. In such cases, the incremental insertion method may perform better than the divide and conquer even for a large number of sites.[12,13]

However, the incremental insertion algorithm can not be parallelized, and the datasets in many practical D-TIN applications are usually very large and may be stored in different geographical distributed nodes, and the D-TIN can not be computed in a single machine, and even in a high-performance computing (HPC) server, the traditional D-TIN algorithm is incapable of figuring out the problem, which urges us to combine the incremental insertion algorithm with another existing algorithm that is capable of parallel computing. Considering the high-efficiency, autonomy characteristics of the divide and conquer algorithm for D-TIN, this paper focuses on a hybrid algorithm (HA) that combines the incremental insertion algorithm and the divide-and-conquer algorithm, which has the advantages of both, and is high efficiency and simple to be migrated and deployed in a distributed and parallel computing environment.

The rest of this paper is organized as follows: Section 2 briefly describes some closely related work on D-TIN and the architecture of a Cloud test-bed, GeoKSCloud, a geographical knowledge service Cloud developed by our project team. Then, a HA to construct D-TIN is introduced in Section 3. In section 3.1, the design and implementation of a distributed and parallel D-TIN service in GeoKSCloud platform is described in detail. Section 4 presents experimental results with contrast evaluation of the HA on centralized and Cloud environment. Finally, we conclude in Section 5.

2 | RELATED WORK

For a dataset with *n* sites, the Voronoi Diagram is a subdivision of the plane into *n* regions; each region corresponds to one site. The region for a given site consists of that portion of the plane closer to it than to any other sites. Both D-TIN and its dual Voronoi diagram have been widely used in several domains, such as geography, meteorology, biology, anthropology, archeology, astronomy, geology, physics, metallurgy, and statistics.[14–16] As a consequence, they have attracted numerous researchers from various disciplines, and a large number of algorithms have been proposed and implemented.

There are several categories of algorithms to construct D-TIN, including some direct methods and the indirect method. A direct method computes and obtains D-TIN directly, while the indirect method construct Voronoi diagram first and then gain the geometric dual D-TIN subsequently. Typical direct algorithms for D-TIN include the divide-and-conquer algorithm, the incremental insertion algorithm (which is also called dynamic algorithm[17]), the triangle expanding algorithm, the plane sweep line algorithm, the gift wrapping algorithm, and the convex hull based algorithm.[1] For *n* sites in the plane, the optimal time complexity for the divide-and-conquer algorithm, the plane sweep line approach and the convex hull based algorithm can be as predominance as $O(n\log n)$, while the other ones[7] have time complexity of $O(n^2)$. In a word, each of the D-TIN constructing algorithm has some specific characteristic, which can fulfill the requirements of different application scenarios.

Lee and Schachter[18] first proposed an indirect approach to construct D-TIN, then Guibas and Stolfi[19] gave more details and designed a very robust implementation to demonstrate the efficiency and advantages of their quad-edge data structure for representing subdivisions. Besides that, typical D-TIN algorithms have been introduced: the incremental insertion algorithm, the divide-and-conquer method, and the plane sweep-line approach over the past 2 decades.

Unfortunately, most of the existing D-TIN algorithms and their dual Voronoi diagram often tend to neglect the handling of the numerical stability of interesting complex bisector manipulation, and some special cases such as cocircular sites and sites lying on a line. Moreover, some distributed and parallel D-TIN mechanisms on Cloud environment do not scale well and present poor performance for compute-intensive network-based applications.[20,21]

There are many strategies to compute D-TIN for a set *P* of *n* sites. These strategies can be grouped into 2 categories: direct and indirect methods. The indirect method is to construct the Voronoi diagram ($Vor(P)$) first and then obtains its dual Delaunay TIN $T(P)$; while the direct methods compute D-TIN directly, typical direct methods include the randomised IA, the divide-and-conquer method, and the plane sweep-line approach.[22] Time complexity of the incremental insertion algorithm for inserting *n* sites is approximately $O(n^2)$, and both the divide-and-conquer method and the plane sweep-line approach have intrinsic predominance in term of time complexity $O(n\log n)$. However, as described in Section 1, the incremental insertion method may perform better than the divide-and-conquer algorithm in some special circumstances; that is the reason why we proposed a HA that combines the divide-and-conquer algorithm and the incremental insertion algorithm, while the HA is efficient and has a time complexity of $O(n\log n)$, and what is more, the HA algorithm can be easily parallelized in a distributed and parallel computing environments.

The emergence of distributed shared infrastructures and Cloud technologies provide new paradigms for dealing with spatial data mining applications, and various sources are available for collaborative computing in Cloud and Cloud environment.[5] The problem of poor efficiency of traditional algorithms can be solved by distributed and parallel computing mechanisms.[23,24] We believe that Cloud technology will greatly improve the distributed data-mining techniques in both the ability to provide on-line and on-demand knowledge creation/discovery services and collaborative composition of the knowledge service itself.

A geographical knowledge service Cloud test-bed (GeoKSCloud)[25,26] is developed by our research team, which is an internet-based, intelligent Cloud platform. GeoKSCloud provides the main functions of distributed spatial data mining and spatial decision-support based on the SOA architecture and Cloud middleware Hadoop. The ultimate goal is to provide an intelligent, efficient, and collaborative geo-spatial problem-solving environment, which will greatly promote the deep transformation of web sharing of spatial information, from the simplest information exchange and interoperability to the spatial data mining and decision-support services.

GeoKSCloud currently includes all the features owned by computational Cloud and Cloud. Once a knowledge service is deployed, it can be monitored, discovered, shared, and called by every node of the Cloud. After introducing large amounts of spatial data-mining services and spatial decision-support services, GeoKSCloud has different kinds of knowledge service capabilities, such as spatial clustering, spatial association rule mining, spatial outlier detection, and urban air pollutant dispersion simulation that are needed by various applications. Through years of research on spatial data-mining algorithms and the mechanisms of Cloud service interoperability, several main function modules have reached maturity and provide better and reliable distributed spatial knowledge services with map-reduce architecture. Typical modules include Cloud resource center, Cloud information center, execution management center, knowledge service center, and Cloud platform management center with a uniform Cloud portal interface. Moreover, GeoKSCloud provides a novel problem-solving environment that will enable the in-depth study of distributed D-TIN algorithms in Cloud environment.[26]

In recent years, Cloud system and platform are widely used in distributed computing and data-mining practices.[5,27,28] This paper conducts distributed and parallel D-TIN computing research in a geographical knowledge service Cloud, named GeoKSCloud, which is developed and tested in our laboratory.

3 | HYBRID ALGORITHM FOR PARALLEL D-TIN

In this section, we introduce a hybrid technique for performing D-TIN. The technique combines the divide-and-conquer and the incremental insertion algorithms.

Considering a site set P with n points in the plane, θ is the threshold that whether the subproblem needs to be further segmented. The kernel idea of the HA is that if the scale of problem P is smaller than θ, then IA is used to compute the D-TIN. Otherwise, the problem P is segmented into 2 approximately equal size of subproblems, namely, left subproblem (L) and right subproblem (R), and the corresponding D-TIN $T(L)$ and $T(R)$ of L and R are computed, respectively. That is, the divide strategy of GuibasStolfi's divide-and-conquer algorithm[3] is applied to the problem whose scale is over θ.

During the process of problem segmentation, the subproblems are divided top-down and recursively until the scale of subproblems (L_i) and (R_i) is less than θ. For the leaf nodes in the segmentation tree, IA is adopted to construct D-TINs, and the result sub-DTINs are merged bottom-up on the basis of the binary tree until the global D-TIN is obtained. Thus, all the sequential steps can be processed in parallel by using multitasking parallel technology. Hence, the parallel version of the algorithm is presented in the section subsequent.

The principle of problem decomposition and subtriangulations merging for the HA is shown in Figure 1.

Main steps of HA algorithm include the divide-and-conquer and the incremental insertion stages, which are more adequate to run on distributed platforms and mainly for very large Delaunay decomposition problems. Through seamless combination of these 2 steps, the new approach inherits the advantages of high-adaptability, simplicity of the incremental insertion algorithm and the advantages of high-efficiency, autonomy of the divide-and-conquer algorithm.[29] At the same time, it overcomes the fatal drawback of a large number of recursions involved in the divide-and-conquer algorithm without a significant reduction in the efficiency. In particular, in a distributed environment, it avoids a large number of network communications and hence improves its performance.

3.1 | Parallel D-TIN computing in Cloud

The parallel D-TIN construction in Cloud consists of 2 atomic services, namely, the divide-and-conquer and the incremental services. The 2 services are encapsulated and deployed independently as atomic Cloud services on different Cloud nodes in GeoKSCloud. Jointly, called `D-TIN service`, the 2 services constitute an integrated distributed D-TIN services. Once a Cloud service is successfully started in any Cloud node, it is ready for the client to invoke a Cloud service everywhere. More detail information about the mechanism of parallel and distributed computing in GeoKSCloud can be found in several studies.[30,31] The key steps of a distributed version of the HA in a Cloud environment are as follows:

1. Sort the sites in ascending order with planar scanning sequence, that is, $(x_i, y_i) < (x_{i+1}, y_{i+1})$ if and only if $x_i < x_{i+1}$, or $x_i = x_{i+1}$ and $y_i < y_{i+1}$. After this process, data sites are divided into several nonoverlapping subsets along the x- and y-axis directions. Each data cell is allocated roughly equal number of sites.
2. Find an available Cloud computing node to conduct the construction of D-TIN for siteset P. Note that this initial Cloud computing node should be able to invoke the hybrid D-TIN service.
3. If the problem size is smaller than a specified threshold, compute the D-TIN directly and return the result of D-TIN to the user.
4. If the problem size is greater than the threshold, partition the problem into 2 adjacent subproblems of approximately equal

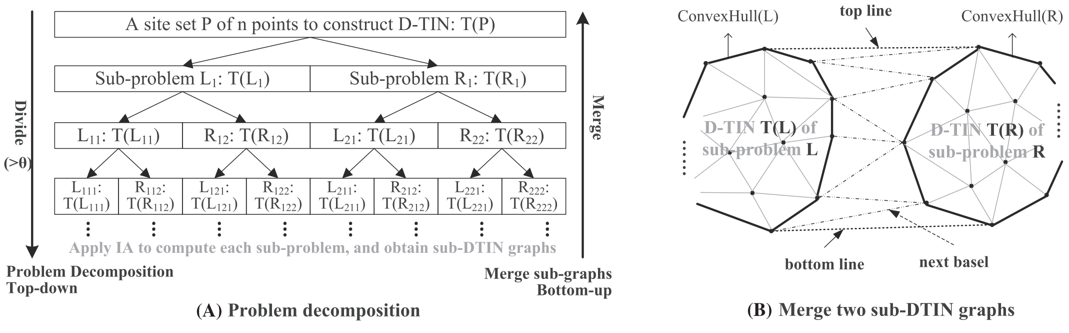

A site set P of n points to construct D-TIN: T(P)
Sub-problem L₁: T(L₁)
Sub-problem R₁: T(R₁)
L₁₁: T(L₁₁)
R₁₂: T(R₁₂)
L₂₁: T(L₂₁)
R₂₂: T(R₂₂)
L₁₁₁: T(L₁₁₁)
R₁₁₂: T(R₁₁₂)
L₁₂₁: T(L₁₂₁)
R₁₂₂: T(R₁₂₂)
L₂₁₁: T(L₂₁₁)
R₂₁₂: T(R₂₁₂)
L₂₂₁: T(L₂₂₁)
R₂₂₂: T(R₂₂₂)
Apply IA to compute each sub-problem, and obtain sub-DTIN graphs
Divide (>θ)
Merge
Problem Decomposition Top-down
Merge sub-graphs Bottom-up
(A) Problem decomposition
ConvexHull(L)
top line
ConvexHull(R)
D-TIN T(L) of sub-problem L
D-TIN T(R) of sub-problem R
bottom line
next basel
(B) Merge two sub-DTIN graphs

FIGURE 1 Problem decomposition and sub-Delaunay triangulations (D-TINs) merging in hybrid algorithm

size, and find 2 available Cloud nodes to deal with them. Go to step 3.

5. Once the division of the initial problem into small subproblems has been finished, the subproblems are performed in parallel on their corresponding Cloud nodes. Each subproblem will generate a subgraph. This partitioning forms a binary tree, in which the leaves represent the subproblems.
6. Merge every 2 adjacent subgraphs starting from the bottom (leaf nodes) of the tree using the merge algorithm of the divide and conquer. This is repeated at every level of the tree until we reach the root of the tree, which represents the initial Cloud node with the whole site set P.

3.2 | GeoKSCloud processes

The execution and scheduling flowchart of the distributed and parallel D-TIN in GeoKSCloud is shown in Figure 2. When a D-TIN is submitted and registered to the Cloud platform, the execution management center (refers to GRAM by Globus) will find and invoke an available hybrid D-TIN service to perform the task.

In practical D-TIN construction applications, after a job has been submitted to the Cloud platform, a hybrid D-TIN service can be invoked. If the problem scale is smaller than the threshold, the execution management center will invoke a Cloud server to deal with the problem. If the scale of the problem is greater than the threshold, 2 Cloud servers are invoked, 1 for each subproblem.

As shown in Figure 2, the merge step keeps waiting until all the 2 subproblems have been solved and the subgraphs have been returned to their parents. To optimize the distributed version of the HA, for both computation and communication times, the Cloud platform will allocate only an extra node instead of 2 to complete the 2 subtasks. Therefore, one needs to find a Cloud server to deal with 1 subproblem while the other subproblem is processed by the local node. In this way, not only it reduces the response time due to data transmission between nodes but also it reduces the application requirements for resources.

3.3 | System architecture

The architecture of the system is shown in Figure 3 in more details. The 2 atomic services of the HA are registered to the information center and deployed on the sever gird nodes. The execution management center will conduct services discovery, auto-matching and execution optimization for the distributed D-TIN job submitted.

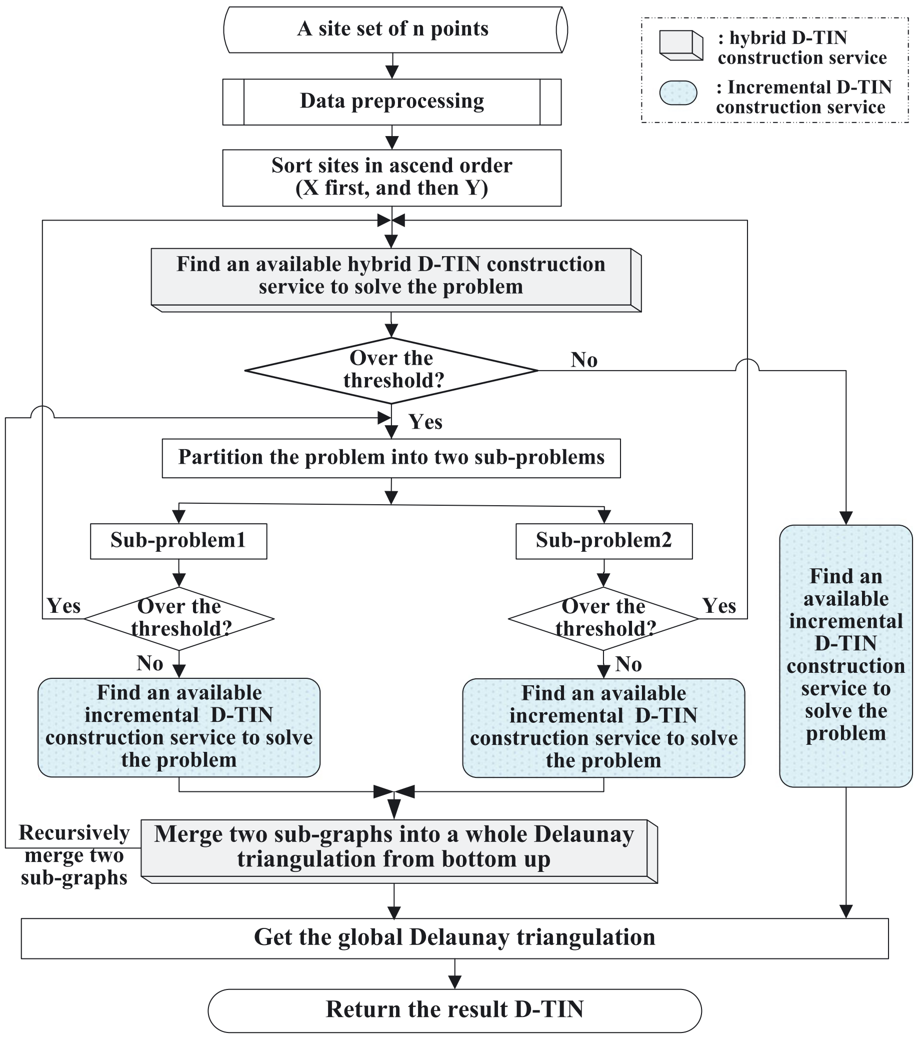

A site set of n points
: hybrid D-TIN construction service
: Incremental D-TIN construction service
Data preprocessing
Sort sites in ascend order (X first, and then Y)
Find an available hybrid D-TIN construction service to solve the problem
Over the threshold?
No
Yes
Partition the problem into two sub-problems
Sub-problem1
Sub-problem2
Yes
Over the threshold?
No
Over the threshold?
Yes
No
Find an available incremental D-TIN construction service to solve the problem
Find an available incremental D-TIN construction service to solve the problem
Find an available incremental D-TIN construction service to solve the problem
Recursively merge two sub-graphs
Merge two sub-graphs into a whole Delaunay triangulation from bottom up
Get the global Delaunay triangulation
Return the result D-TIN

FIGURE 2 Distributed Delaunay triangulation (D-TIN) construction in Cloud environment

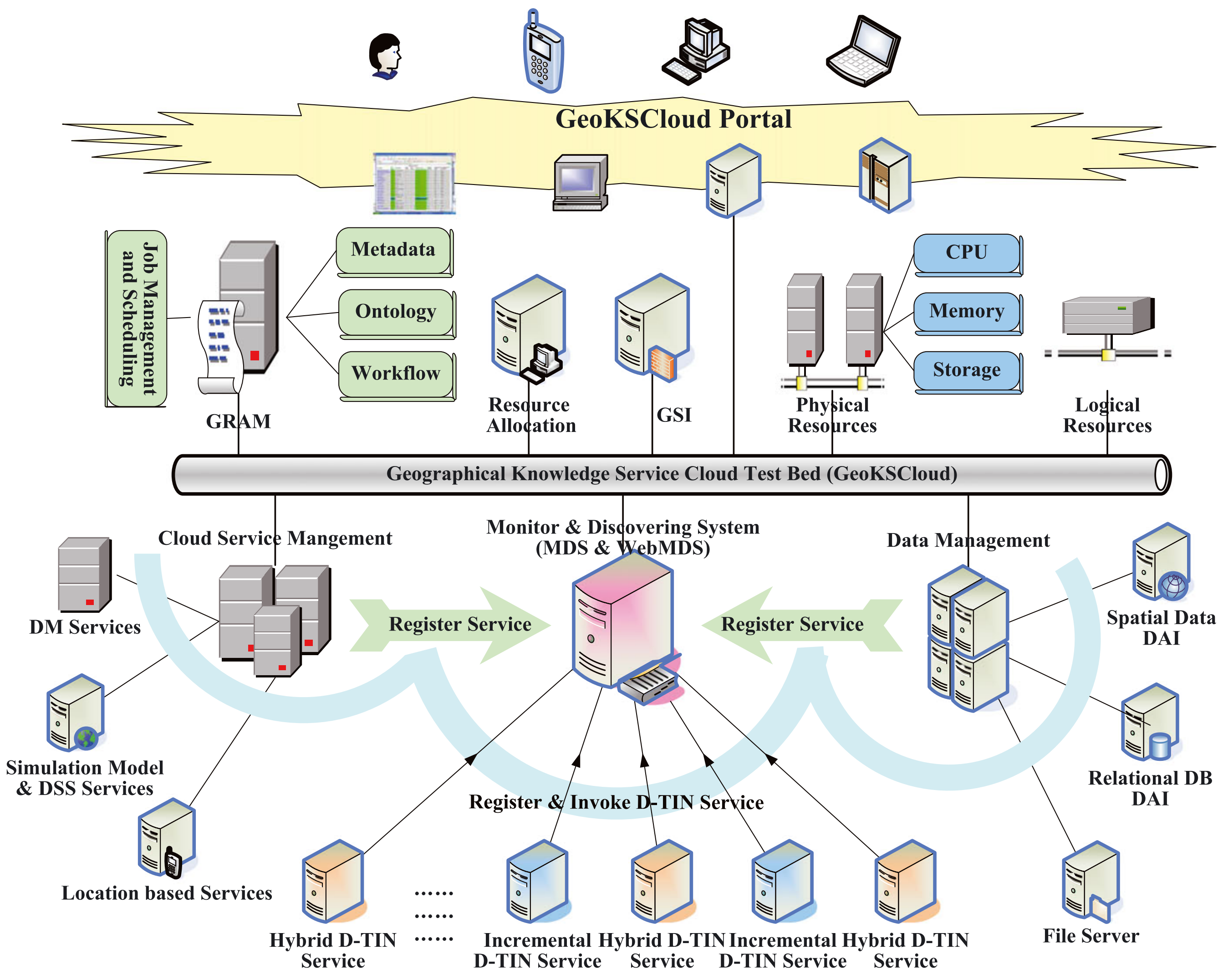


FIGURE 3 Network topology of distributed Delaunay triangulation (D-TIN) service in GeoKSCloud

A detailed process of Cloud service management and scheduling by the execution management center is shown in Figure 3. In GeoKSCloud, all services are registered and monitored by the information center MDS (refers to the Cloud service monitor and discovery service in Cloud). For instance, for hybrid D-TIN services, the execution management center will access the MDS center for the incremental insertion D-TIN services and their corresponding data services when the task is performed. Essentially, all Cloud services registered with MDS keep their stubs in the information center. Once an application requires a specified Cloud service, it requests its corresponding stub from MDS and invokes the service in Cloud servers simultaneously. At the same time, the data set involved in the application, referred to as a data service, is sent to the Cloud server. In GeoKSCloud, the data management is implemented by means of data movement service, such as CloudFTP (Cloud file transfer protocol) and RTF (reliable file transfer), or data replica service, such as RLS (replica location service). In a word, all the cloud services related to data are managed by GRAM (the Global Resource Allocation Manager of Cloud), and the organization of cloud data is with DAI infrastructure (the Data Access and Integration model). So each node will serialize the data locally and then transmit the parameters to the Cloud server. At the Cloud server, the data will be deserialized and used by the process of D-TIN construction. Without doubt, all operations in cloud is under GSI (the Global Security Infrastructure of Cloud), to ensure the safety of data and service functions of Cloud.

As shown in Figure 4, steps 1 to 12 are the main procedures for a distributed D-TIN scheduling. The steps A to D are executed by the information center that referring to an incremental D-TIN service; and substeps a and b depict the distributed and parallel mechanism for the HA and their scheduling processes in a Cloud environment. If the problem size is smaller than the threshold, it will request and invoke an incremental D-TIN server. Otherwise, it will request and invoke another hybrid D-TIN server.

3.4 | Implementation

The most important task of the proposed distributed and parallel D-TIN construction service in a Cloud is to implement 2 atomic D-TIN services: the Hybrid D-TIN service and the incremental insertion D-TIN service.

Guibas and Stolfi[32] proposed an robust quad-edge data structure for representing graph embedding on 2-dimensional manifolds, which simultaneously represents a structure and its dual. In this paper, this idea is adopted to represent D-TIN. The implementation of the divide-and-conquer algorithm, IA, and the HA benefitted hugely from

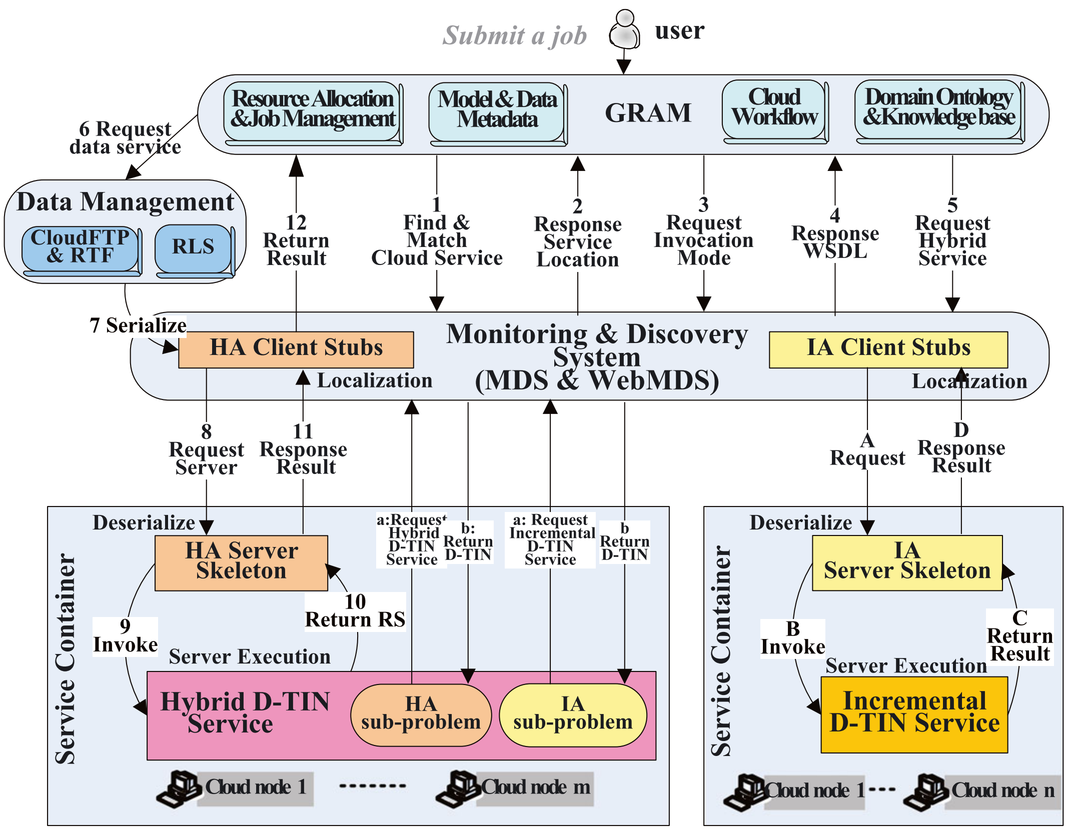


FIGURE 4 Execution and scheduling of distributed Delaunay triangulation (D-TIN) construction in GeoKSCloud. CloudFTP, Cloud file transfer protocol; HA, hybrid algorithm; IA, incremental algorithm; MDS, monitor and discovery service; RLS, replica location service; RTF, reliable file transfer

quad-edge data structure and mainly the integration of the incremental insertion algorithm into the divide-and-conquer algorithm.

The pseudocode for the implementation of hybrid D-TIN construction algorithm in Cloud environment is shown in Algorithm 1, which greatly exhibits the idea of the divide and conquer. The 2 subproblems L and R are processed in parallel when the scale of the problem is over the threshold.

Algorithm 1 Pseudo-code of Hybrid D-TIN Service.

```
Input: A site set s[ ] in ascending order.
    A Specified threshold $\theta$ for distributed D-TIN construction
Output: Delaunay Triangulation for the site s[ ].
Procedure CreateDTINbyDistributedHybrid(s[ ]) return D-TIN
    //if problem scale $\leqslant \theta$, find an incremental service to tackle with the job
    If (size $\leqslant \theta$)    //invoke an incremental D-TIN service
        Return invokeCloudService(S, "Incremental");
    //problem scale > $\theta$, divide S into 2, and find 2 hybrid service to treat them
    D-TIN LeftDtin = invokeCloudService(L, "Hybrid");    // for L
    D-TIN RightDtin = invokeCloudService(R, "Hybrid");   // for R
```

A crucial part of the parallel computing of D-TIN in Cloud lies in the construction of subtriangulations correspondent to the branch nodes in the binary tree model, while the internal nodes uses the divide-and-conquer service and the final leaf nodes of the binary tree adopted the incremental service to construct D-TIN; the encapsulation of the *IA* Cloud service can be seen in Algorithm 2.

Algorithm 2 Pseudo-code of Incremental D-TIN Service.

```
Input: A site set s[ ]
Output: Delaunay Triangulation for the site s[ ].
Procedure CreateDTINbyIncremental(s[ ]) return D-TIN
    Return getDTINbyInsertion(s);    //create D-TIN with incremental insertion service
```

3.5 | Cost model

Assume that the size of D-TIN computing problem is n, and the threshold for distributed and parallel D-TIN computing is θ. Let k be the number of nodes needed to process subproblems of size $m \leqslant \theta$. Then, the number of subdivisions is easy to calculate according to the mechanism of the HA. On the basis of the algorithm described above, the division of the problem consists of a binary tree. Therefore, k is of the form $k = 2^{\omega}$, such that $2^{\omega-1} < \lceil \frac{n}{\theta} \rceil \leqslant 2^{\omega}$.

For a site set of n points, the HA divides the problem recursively into 2 approximately equal subproblems. Let $T_1(n)$ be the response time for the sequential version and $T_k(n)$ be the response time for the distributed version with k processors. $T_1(n)$ would be the sum of the computation time by the incremental insertion algorithm and the merging time of the 2 subproblems, while $T_k(n)$ would be the sum of the computation time by the incremental insertion algorithm, the merging time of the 2 subproblems and the communication time between the server and the clients. Let $T_{IA}(m)$ be the computation time of IA for a problem of size m, $T_{merge}(m)$ be the time for merging 2 subproblems of size $\frac{m}{2}$ each and $T_{comm}(m)$ be the time for exchanging dataset of size m among the servers and clients. So the response time for the sequential version $T_1(n)$ and the distributed version $T_k(n)$ would be

$$T_1(n) = kT_{IA}(m) + \sum_{i=1}^{\omega} \frac{k}{2^i} T_{merge}(2^i m), \tag{1}$$

$$T_k(n) = T_{IA}(m) + \sum_{i=1}^{\omega} \left(T_{merge}(2^i m) + T_{comm}(2^{i-1} m) \right), \tag{2}$$

because D-TIN can be computed in $O(n\log n)$ by using the divide-and-conquer algorithm and in $O(n^2)$ by using the incremental insertion algorithm. Therefore, time complexity for D-TIN construction in the leaf node with m points would be $O(m^2)$ by the incremental insertion algorithm, and the corresponding time for merging 2 subproblems L and R in the internal nodes would be $O(2m)$ by the HA.

As for the communication time in the distributed version of D-TIN construction, a data structure *QuadEdge* is adopted to fully express the spatial topology of the D-TIN and its dual, so the main source of data transmission between the clients and the servers is a series of *QuadEdge*. For a site set of n points, there are $3n + 6$ edges at most in the corresponding D-TIN. As a result, the communication time depends on the number of *QuadEdge*.

To sum up, the response time for the sequential version T_1 is km $(m + \omega)$,

Similarly, the response time for the distributed and parallel version T_k is $m(m + 3(k - 1))$, because the subproblems with the same level in the binary tree structure are computed in parallel.

It can be found that the distributed version, $T_k(n)$, outperforms the sequential version, $T_1(n)$, by a factor k, especially when the problem size becomes larger and enough Cloud nodes are available for the distributed version of D-TIN computing. Moreover, according to the principle of the HA, it has 2 special cases: (1) it can degenerate into the divide-and-conquer algorithm if the threshold θ is smaller than 1. In this case, we need as many processing nodes as the size of the problem. Therefore, the communication time may dominate the whole execution, which is not effective way of distributing the algorithm. (2) The HA can degenerate into the incremental insertion algorithm if the threshold $\theta \geqslant n$. In this case, the number of processing nodes will be $k = 1$, and the response time will be equal to the sequential version's response time.

4 | EXPERIMENTS AND ANALYSIS

Some experiments are performed to compare the performance of the HA in a single machine (sequential implementation) with the distributed version (deployed in GeoKSCloud platform). In the experiments, different problem sizes were tested with threshold taken as $\theta \in [\frac{n}{8}, \frac{n}{4}]$. Following the process of distributed scheduling of the hybrid D-TIN service, the problem is divided recursively into 2 subproblems, and $\frac{k}{2}$ different Cloud nodes with hybrid D-TIN service and k Cloud nodes with the incremental insertion D-TIN service are found and invoked to finish the parallel construction work. Figure 5 shows an example of the scheduling process with a site set of 10 000 points and the threshold is 2000. The rectangle object denotes the hybrid D-TIN service, while the rounded rectangle object denotes the incremental insertion D-TIN service. Both the hybrid and the incremental insertion D-TIN services are deployed over the GeoKSCloud nodes.

The number of sites and the corresponding executions time are listed in the Table 1. The experimental results indicate that the distributed and parallel D-TIN construction service based on the proposed HA can be used to compute Delaunay TIN in Cloud environment in an efficient way. One can notice from the results that the distributed version outperforms the sequential version, especially when m is becoming larger and larger. This is not really a surprise since more processing nodes are expected to perform better than one, as it is expected in the theoretical model.

Moreover the performance issue is not in the computation time, but in the communication time and the way that the algorithm was parallelized on k processing nodes. Therefore, the speed-up of the distributed version of the algorithm can be affected heavily by the overheads due to the communication times and other system activities. One can notice that for smaller sizes of the problem, the distributed version performs poorly against the sequential version, as the communications count for large part of the response time. From the last 3 columns of Table 1, we can conclude that the communication time of the distributed version counts for a significant proportion of the global response time.

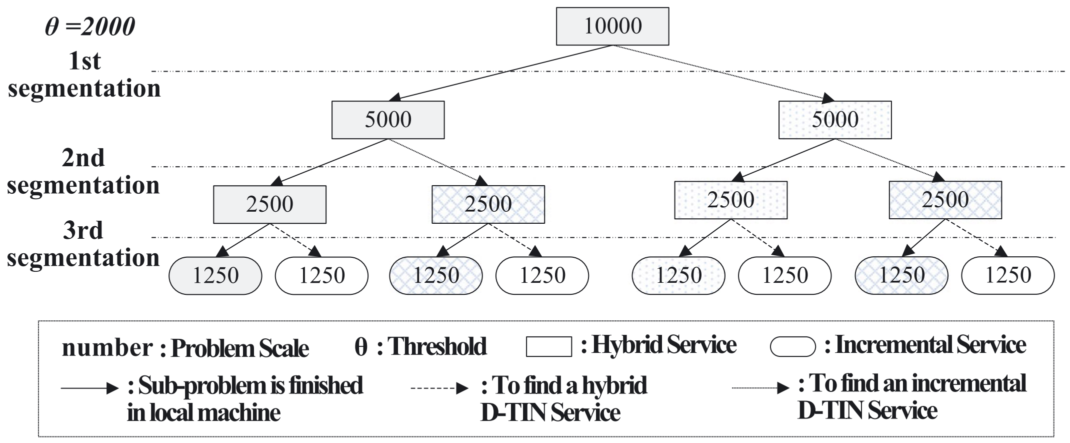
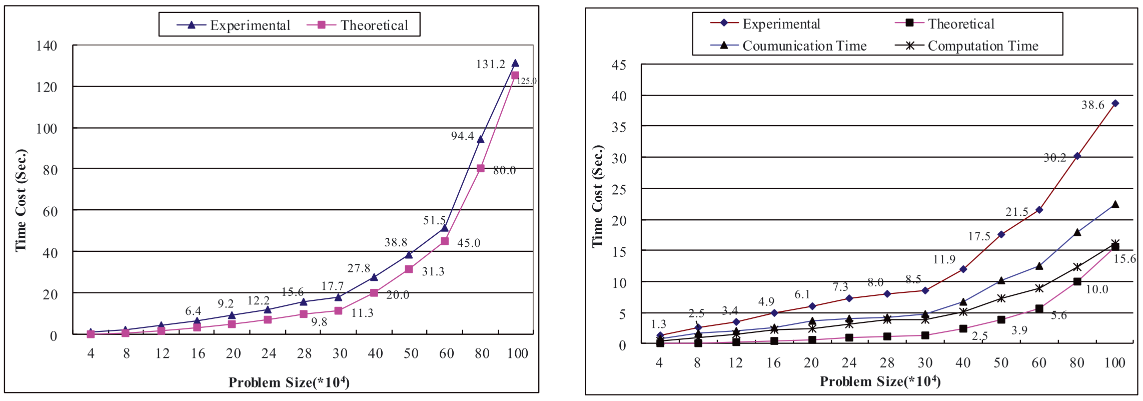


FIGURE 5 Example of distributed Delaunay triangulation (D-TIN) construction with $n = 10000$, $\theta = 2000$

TABLE 1 Time consumption of Delaunay triangulation construction with different conditions (time, sec)

Prob.Size	**Sequential Impl**		**Para & Dist Impl ($\theta = \lceil \frac{n}{8} \rceil$)**		
(n)	$\theta = \log_2 n$	$\theta = \lceil \frac{n}{8} \rceil$	**Comm time**	**Comp time**	**Total time**
40 000	1.422	0.879	0.837	0.454	1.292
80 000	2.797	2.291	1.603	0.939	2.541
120 000	4.349	4.137	1.985	1.457	3.442
160 000	6.265	6.428	2.590	2.291	4.880
200 000	7.677	9.203	3.662	2.410	6.073
240 000	9.563	12.164	4.053	3.213	7.266
400 000	16.411	27.793	6.724	5.182	11.906
600 000	26.349	51.533	12.480	8.994	21.474
800 000	35.693	94.398	17.888	12.274	30.162
1 000 000	44.505	131.182	22.462	16.163	38.625



(A) Time cost of sequential D-TIN **(B) Time cost of parallel D-TIN**

FIGURE 6 Experimental and theoretical execution time of A, sequential and B, parallel Delaunay triangulation (D-TIN)

However, the distributed implementation is much better when the scale of the problem increases gradually.

Figure 6 shows the experimental and theoretical time complexity of sequential and parallel versions of D-TIN based on the experimental results reported in Table 1 and the cost model developed in Section 3.5. Figure 6A,B follows exactly what was expected in theory. Particularly, the communication time between nodes in the distributed version is well inline with the experimental results. The difference between the theoretical and the experimental results is due to some other system activities, such as the scheduling time of Cloud services. Moreover, we considered in our theoretical model that the communication time is proportional to the size of the problem (linear progression), which is not really the case on Cloud platforms due to the network traffic and heterogeneity of their resources. This is also reflected in these results. Overall results of the distributed version of the HA are promising and mainly for very large size of the problem.

5 | CONCLUSIONS

In this paper, an adaptive parallel D-TIN construction algorithm with dynamic pruned binary tree model is presented, which combines the divide-and-conquer algorithm and the incremental insertion algorithm, and has the advantages of both algorithms. Moreover, the proposed HA is high efficiency and simple to be migrated and deployed in a distributed and parallel computing environment.

In addition, a parallel implementation of the hybrid D-TIN construction algorithm is introduced as well. We implement both the stand-alone version and the parallel version of the HA with Java/J2EE in our GeoKSCloud test-bed. For the parallel version, the divide-and-conquer part and the incremental insertion part are encapsulated as 2 atomic Cloud services, and they are deployed in several Cloud-computing nodes, respectively. Then, the integrated use 2 kinds of Cloud-computing nodes to construct D-TIN in parallel in a dynamic pruned binary tree model is narrated, namely, the distributed and parallel version of D-TIN computing, which is the center work of the paper.

Finally, the effectiveness and efficiency of the HA and the parallel version are verified through both theoretical analysis and experiments on stand-alone and Cloud environment. Theoretically, the incremental insertion algorithm for D-TIN construction has a time complexity of $O(n^2)$, while the divide-and-conquer algorithm is $O(n\log n)$. The HA is a combination of the incremental insertion algorithm and the divide-and-conquer algorithm, whose time complexity is also $O(n\log n)$, which is optimal for D-TIN construction in stand-alone model. Although, there is not any improvement in the efficiency of D-TIN construction, but the HA has some advantages over the incremental insertion and the divide-and-conquer algorithms. First, the HA own the superiority of both algorithms. That is, it takes the characteristics of high-adaptability, simple to understand and implement by incremental insertion algorithm, and has the characteristics of high-efficiency, autonomy by the divide-and-conquer algorithm simultaneously. Second, the HA can be easily migrated and deployed as a parallel version to realized the distributed and parallel computing of D-TIN.

Besides that, some experimentations are conducted, to compare the time consumption of the HA with the IA and the divide-and-conquer algorithm, and to compare the cost of the stand-alone HA with the parallel version, results show that the HA outperforms the incremental insertion algorithm, but it is a bit weaker than the divide-and-conquer algorithm. In a word, the HA and its parallel version for D-TIN construction are effective and competitive in 2 aspects. First, without taking into consideration the time consumption of data transfer among different distributed and parallel computing nodes, when divide a problem and distribute the subproblems to the related computing nodes, and leave out the cost of scheduling of the computing nodes, the algorithm has a time complexity of $O(n\log n)$, which is optimal for D-TIN construction. Second, the datasets in many practical applications are usually very large and may be stored in different geographical distributed nodes, and the D-TIN can not be computed in a single machine and even in a HPC server, the traditional stand-alone algorithm is incapable of figuring out the problem. At this very moment, the proposed HA and its parallel version can be used to compute D-TIN in a distributed and parallel network environment with a satisfiable efficiency.

Furthermore, this paper implements and tests the hybrid D-TIN computing algorithm in GeoKSCloud, which is developed by our research team. Among which, GeoKSCloud is only a network environment that sustains 2 atomic services (divide-and-conquer and incremental insertion parts) of the HA, helps to manage, transfer, allocate, and schedule of the data and computer powers, to enable the distributed and parallel computing of D-TIN. The HA can be easily migrated and deployed in another distributed and parallel computing environment, such as grid, HPC, or exascale computing platform, with PCs or clusters.

ACKNOWLEDGMENTS

This research was supported in part by the National Natural Science Foundation of China (No.41401458, No.61501120); the Key Laboratory of Spatial Data Mining and Information Sharing of Ministry of Education, Fuzhou University (No. 2017LSDMIS01); the Key Program of Science and Technology of Fujian Province (No.2015H0015); the Natural Science Foundation of Fujian Province of China (No.2014J05045, No.2016J05148, No.2016J01753); the Science and Technology Foundation of The Education Department of Fujian Province of China (No.JA12375); and Youth Foundation of Fujian Agriculture and Forestry University (No.k13xjj05a).

REFERENCES

1. Su T, Wang W, Lv Z, Wu W, Li X. Rapid Delaunay triangulation for randomly distributed point cloud data using adaptive hilbert curve. *Comput Graph*. 2016;54(8):6574.
2. Jacobsen DW, Gunzburger M, Ringler T, Burkardt J, Peterson J. Parallel algorithms for planar and spherical Delaunay construction with an application to centroidal Voronoi tessellations. *Geosci Model Dev*. 2013;06(04):1353-1365.
3. Wu WZ, Rui YK, Su FZ, Cheng L, Wang JC. Novel parallel algorithm for constructing Delaunay triangulation based on a twofold-divide-and-conquer scheme. *Gisci Remote Sens*. 2014;51(05): 537-554.
4. Feng D, Tsolakis C, Chernikov AN, Chrisochoides NP. Scalable 3D hybrid parallel Delaunay image-to-mesh conversion algorithm for distributed shared memory architectures. *Procedia Eng*. 2015;124:18-30.
5. Zhang Y, Gao Q, Gao L, Wang C. iMapReduce: A distributed computing framework for iterative computation. *J Grid Comput*. 2012;10(01): 47-68.
6. Shen J, Guo L, Qi L, Zhu W. Delaunay triangulation parallel construction method and its application in map generalization. *ISPRS - Int Archives of the Photogrammetry, Remote Sensing and Spatial Inform Sci*. 2012;XXXIX-B2:23-28.
7. Zadravec M, Zalik B. An almost distribution-independent incremental Delaunay triangulation algorithm. *Visual Comput*. 2005;21(06): 384-396.
8. Boissonnat J-D, Devillers O, Hornus S. Incremental construction of the Delaunay triangulation and the Delaunay graph in medium dimension. *2009 Twenty-fifth Annual Symposium On Computational Geometry (SCG'09)*, Aarhus, Denmark; 2009:208-216.
9. Mostafavi MA, Gold C, Dakowicz M. Delete and insert operations in Voronoi/Delaunay methods and applications. *Comput Geosci*. 2003;29(04):523-530.
10. Zhao H, Bikdash M. Algorithm to locate points in a Delaunay triangulation. *2006 38th Southeastern Symposium on System Theory*, Cooksville, TN, Canada; 2006:211-215.
11. Chen C, Chen L, Shi M. A highly solid model boundary preserving method for large-scale parallel 3D Delaunay meshing on parallel computers. *Comput-Aided Design*. 2015;58:73-83.
12. Laarhoven JW, Ohlmann JW. A randomized Delaunay triangulation heuristic for the Euclidean Steiner tree problem in R_d. *J Heuristics*. 2011;17(04):353-372.
13. Chen RJ, Gotsman C. Localizing the Delaunay triangulation and its parallel implementation. *Transactions on Computational Science: Special Issue on Voronoi Diagrams and Their Applications*, Lecture Notes in Computer Science, vol. 8110: Springer Berlin Heidelberg; 2013:39-55.
14. Mulchrone KF. Application of Delaunay triangulation to the nearest neighbour method of strain analysis. *J Struct Geol*. 2003;25(05): 689-702.
15. Wong W-C, Chan SHG. Improving Delaunay triangulation for application-level multicast. *2003 IEEE Global Telecommunications Conference (GLOBECOM'03)*, San Francisco, CA; 2003:2835-2839.
16. Rana O, Walker D, Li M, Lynden S, Ward M. PaDDMAS: parallel and distributed data mining application suite. *2000 14th International Conference on Parallel and Distributed Proeessing Symposium (IPDPS 2000)*, Cancun, Mexico; 2000:387-392.
17. Zalik B, Kolingerova I. An incremental construction algorithm for Delaunay triangulation using the nearest-point paradigm. *Int J Geogr Inf Sci*. 2003;17(02):119-138.
18. Lee DT, Schachter BJ. Two algorithms for constructing the Delaunay Triangulation. *Int J Parallel Prog*. 1980;09(03):219-242.
19. Guibas L, Stolfi J. Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. *ACM T Graphic*. 1985;04(04):74-123.
20. Cannataro M, Talia D, Trunfio P. Distributed data mining on the grid. *Future Gener Comp Sy*. 2002;18(08):1101-1112.
21. Chen M-B, Chuang T-R, Wu J-J. Efficient parallel implementations of near Delaunay triangulation with High Performance Fortran. *Concurr Comp-Pract E*. 2004;16(12):1143-1159.
22. Buchin K, Mulzer W. Delaunay triangulations in $O(sort(n))$ time and more. *J ACM (JACM)*. 2011;58(02):1-27.
23. Chen M-B, Chuang T-R, Wu J-J. Parallel divide-and-conquer scheme for 2D Delaunay triangulation. *Concurr Comp-Pract E*. 2006;18(12): 1595-1612.

24. Cesario E, Talia D. Distributed data mining models as services on the grid. *2008 IEEE International Conference on Data Mining Workshops(ICDMW '08)*, Pisa, Italy; 2008:486-495.
25. Lin JX, Chen CC, Wu XZ, Wu JW, Wang WB. GeoKSGrid: A geographical knowledge grid with functions of spatial data mining and spatial decision. *IEEE International Conference on Spatial Data Mining and Geographical Knowledge Services (ICSDM 2011)*, Fuzhou, China; 2011:143-148.
26. Chen C, Lin J, Wu X, Wu J. Parallel and distributed spatial outlier mining in grid: algorithm, design and application. *J Grid Comput*. 2015;13(02): 139-157.
27. Kim S, Kim J, Weissman JB. A security-enabled grid system for MINDS distributed data mining. *J Grid Comput*. 2014;12(03):521-542.
28. Cuzzocrea A. Models and algorithms for high-performance data management and mining on computational grids and clouds. *J Grid Comput*. 2014;12(03):443-445.
29. Rui Y, Wang J, Qian C, Liu J, Li X. A new compound algorithm study for Delaunay triangulation construction. *2007 15th International Conference on Geoinformatics: Cartographic Theory and Models*, Nanjing, China; 2007:B7510-B7510.
30. Lin JX, Chen CC, Ye DY, Wang WB. Research on grid based spatial outliers mining and its implementation. *2008 3rd International Conference on Intelligent System and Knowledge Engineering (ISKE 2008)*, Xiamen, China; 2008:324-328.
31. Lin JX, Chen CC, Wang QM, Wang WB, Wu JW. Distributed spatial data mining in geospatial knowledge service grid. *2010 2nd International Conference on Advanced Geographic Information Systems, Applications, and Services (GEOProcessing 2010)*, St. Maarten, Netherlands Antilles; 2010:80-87.
32. Guibas LJ, Knuth DE, Sharir M. Randomized incremental construction of Delaunay and Voronoi diagrams. *Algorithmica*. 1992;07(01): 381-413.

How to cite this article: Lin J, Chen R, Wu L, Shu Z, Yang C. Adaptive parallel Delaunay triangulation construction with dynamic pruned binary tree model in Cloud. *Concurrency Computat: Pract Exper*. 2017;29:e4157. https://doi.org/10.1002/cpe.4157