2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum

MapReduce Skyline Query Processing with A New Angular Partitioning Approach

Liang Chen
Zhejiang University
Hangzhou, China
cliang@zju.edu.cn

Kai Hwang
University of Southern California
Los Angeles, USA
kaihwang@usc.edu

Jian Wu
Zhejiang University
Hangzhou, China
wujian2000@zju.edu.cn

***Abstract*—Fast skyline selection of high-quality web services is of critically importance to upgrade e-commerce and various cloud applications. In this paper, we present a new MapReduce Skyline method for scalable parallel skyline query processing. Our new angular partitioning of the data space reduces the processing time in selecting optimal skyline services. Our method shortens the Reduce time significantly due to the elimination of more redundant dominance computations. Through Hadoop experiments on large server clusters, our method scales well with the increase of both attribute dimensionality and data-space cardinality.**

We define a new performance metric to assess the local optimality of selected skyline services. By experimenting over 10,000 real-life web service applications over 10 performance attribute dimensions, we find that the angular-partitioned MapReduce method is 1.7 and 2.3 times faster than the dimensional and grid partitioning methods, respectively with a higher probability to reach the local optimality. These results are very encouraging to select optimal web services in real-time out of a large number of web services.

***Keywords*-Web services; skyline query processing; MapReduce; Hadoop programming**

I. Introduction

In recent years, we see an increasing demand of personalized web services and cloud computing applications. Typically, there may be a large number of providers that respond to the same service request. For example, the search result of Seekda! [1] may be serviced by one or more out of 100 weather forecast providers or from 200 stock-query answering providers. In other words, many providers are competing for the similar services.

Users demand the *quality of service* (QoS) in making their selection of the service. How to select an appropriate or the best service from many alternative offerings becomes a major concern from the vast user communities [1][2][3][4]. In another front, the dynamically of web service environment poses a challenge to the efficiency of QoS-based service selection [5][6].

Ever since 2001, the skyline operators [7] and their extensions have been advocated for QoS-based service selection by many authors [8][9][10]. The skyline approach is especially attractive to achieve optimal or semi-optimal service selection in a multi-attribute decision-making process. We have adopted the skyline approach to solving the QoS problem in several previous works [11][12]. However, two critical issues are not considered carefully in previous work, which are important to the generation of QoS-guaranteed skyline solution in real-life web/cloud service applications.

- **Exponential growth of the Skyline complexity in the selection space**. The skyline selection complexity increases exponentially with both the number of attribute dimensions and the cardinality of the skyline sample data space. With the blooming of the service industry, the skyline solution space may become too complex to optimize in real-time on a conventional computer.
- **How to assure the QoS in skyline selected services.** The QoS of selected service may get degraded rapidly, when the Internet traffic becomes saturated or jammed with bottlenecks. This may prevent the skyline solution from achieving the desired level of QoS. The situation may be compounded badly, if both selection complexity and network traffic get deteriorated at the same time.

In this paper, we attempt to solve both of the above problems. We extend the the MapReduce model [13] for solving the Skyline selection problem on cloud platforms. The approach is to accelerate the Skyline selection process by exploring high degree of distributed parallelism in automated datacenters or public computing clouds.

In this approach, the service search space is first partitioned into subspaces, then the local skylines of each partition are computed in parallel, and the global skyline services are finally computed by merging all local skyline choices. It should be noted that a local skyline choice may not be necessarily globally optimal.

We generate three MapReduce versions of the BNL (Block Name Label) Skyline algorithm [7], based on three data space partitioning schemes. Our skyline algorithms are denoted as *MR-Dim*, *MR-Grid*, and *MR-Angle* in Table 1. Specifically, in MR-Angle, we propose to use a novel angular partitioning algorithm to divide the data space, which improves the efficiency of MapReduce based skyline query process. The Hadoop experiments show that MR-Angle method is 1.7 and 2.3 times faster than the other two methods, when the service search space is very large. Besides the processing time, we propose another metric,

[1]A web service search engine, http://www.webservices.com

978-0-7695-4676-6/12 $26.00 © 2012 IEEE
DOI 10.1109/IPDPSW.2012.279

Table I
KEY PARAMETERS AND MAPREDUCE SKYLINE QUERY PROCESSING ALGORITHMS

Notation	Definition and Brief Description
Data Space, S	An n-dimensional space of N data points
Skyline Set,	Set of skyline data points in the space S
Processing Time, T	CPU time to generate the skyline set over the data space S
MR-Dim	MapReduce Skyline selection using dimensional partition over S space
MR-Grid	Grid-partitioned MR Skyline selection using grid partition
MR-Angle	Angular partitioned MR Skyline selection using the angular partitioning

optimality of local skyline selection, to evaluate the QoS of local skyline services in different partitions. The MR-Angle outperforms the other two methods in terms of *local skyline optimality*.

In particular, the main contribution of this paper is summarized as follows:

1) We extend the MapReduce application to select optimal web services with QoS assurance.
2) A novel angular partitioning algorithm is adopted in skyline query processing
3) Report large-scale MapReduce skyline benchmark experimental results on large clusters over large web-service datasets.

For the convenience of our readers, Table 1 summarizes important notations and algorithms used in this paper. Notations for three MapReduce skyline methods are given.

The rest of the paper is organized as follows: Section 2 presents the background of Skyline selection process for web services, and the MapReduce model specially-tailored for skyline query processing. Section 3 specifies three MapReduce skyline algorithms including our new method. Section 4 analyzes the complexity of our new angular-partitioned MapReduce method. Section 5 presents the experimental setting and reports Hadoop experimental results obtained. The analysis and evaluation of the optimality of local skyline selection is given in Section 6. Finally, we summarize our technical contributions and make some suggestions for further research challenges that can be extended from this work.

II. MAPREDUCE SKYLINE QUERY PROCESSING

In this section, we introduce skyline query and describe how to link the skyline approach to solving the problem of QoS-based web service selection. Given a set Q of data points in d-dimensional QoS space, each dimension represents a performance attribute with values properly ordered. The lower-valued points are better than the higher-valued ones. A data point P_j is dominated by P_i , if P_i is better than or equal to P_j in all dimensions. Furthermore, P_i must be better than P_j in at least one dimension [8].

Given two services represented by two service data points s_1 and s_2 in the QoS space Q. The service s_1 dominates service s_2, if s_1 is better than or equal to s_2 in all attribute dimensions of the space Q. Furthermore, s_1 must be better than s_2 in at least one attribute dimension of Q. The subset S of services form the skyline in the QoS space Q , if all service points on the skyline are better than or equal to other services along all attribute dimensions in the space Q. In other words, all skyline services are not dominated by any other service in space Q.

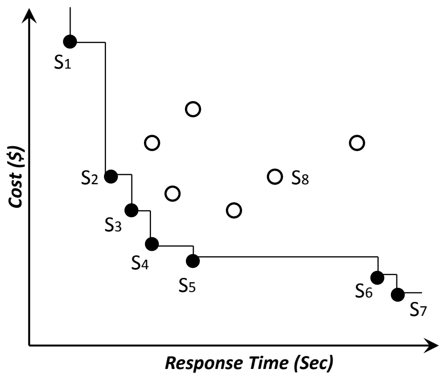


Figure 1. A 2-dimensional data space, where the skyline choices along the contour have lower service cost and shorter response time, compared with all service points in either attribute dimensions

Figure 1 shows a 2-dimensional QoS space Q consisting of many service data points. Each service is characterized by two performance attributes: namely the *response time* (x-axis) and the *service cost* (y-axis). The coordinate of each data point corresponds to the pair of values: (*response time*, *service cost*) of a particular service.

For example, the service point s_3 is a skyline point denoted by shown by solid circles, because it is not dominated by any other service points. Thus s_3 has shorter response time and lower cost than all service points to its right or above s_3 in the Q space. Similar superiority can be established to qualify other skyline services. Thus, we obtain the skyline set $S = \{ s_1, s_2, s_3, s_4, s_5, s_6, s_7\}$.

In general, we should expect the relationship $S \ll Q$. Many of the non-skyline or unselected data points are towards the upper right corner of the 2-D space. This process essentially needs to compare all data points in a pairwise

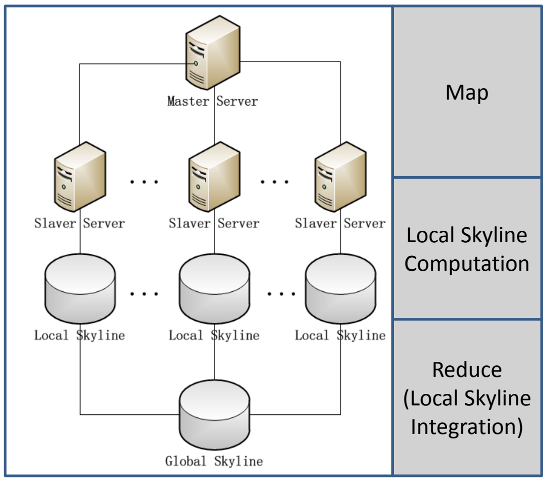


Figure 2. MapReduce model for computing skyline tasks to achieve optimal QoS selection

fashion. When the QoS space Q is very large, this is a very time-consuming process. That is why we need parallel comparison in a MapReduce cluster or cloud.

All hollow circles in Figure 1 are non-skyline service points. In particular, we show that service s_8 is not a skyline service, because it is dominated by services s_2, s_3, s_4, and s_5 along the x-axis and s_8 has no competitors along the y-axis. The Skyline is visualized by linking lines among the selected data points in set S. Usually, the skyline is a contour at the lower left corner of the data space Q towards the origin.

Inspired by distributed parallel processing, we propose to apply MapReduce technology to upgrade the computing efficiency with scalable performance in large-scale Skyline query processing. We propose a variant of MapReduce approach by adding a process between Map and Reduce. The idea is shown in Figure 2 in 3 steps.

1) **The Map Process**. The service data points are partitioned by the master server (e.g. UDDI) into multiple data blocks based on the QoS demand. The data blocks are dispatched to many slaver servers for parallel processing.
2) **Local Skyline Computation**. This process is used to generate the local skylines from service data points in subdivided data blocks.
3) **The Reduce Process**. In this process, local skylines generated by all slaver servers are merged and integrated into a global skyline, which applies to all services being evaluated.

The MapReduce skyline model handles the pairwise evaluation of existing services and the addition of new services. Given a new service which is added into UDDI, traditional approach has to compute the global skyline again. With the MapReduce approach, the new service is first mapped into a group and added into the local skyline computation. Then all local skylines are integrated into the global skyline at the Reduce stage.

In other words, we only need to compare the new service with the services in a subdivided group. Specifically, the number of local skyline services is largely smaller than the one of services in UDDI. If the number of services for skyline computation becomes too large, the MapReduce solution can be even applied iteratively using the Twister HLL language support [14].

A. Mapping of Partitioned Skyline Tasks

Given a set of N slave servers, the quality of the selected skyline services depends on the efficiency of the local skyline computation and the performance of the integration process. Thus, the efficiency and QoS of the MapReduced skyline process depends mainly on how to explore the distributed parallelism among the N servers, to accelerate the Map stage.

The efficiency of the mapping depends on data space partitioning. The service data points are partitioned into divided regions. The goal is to achieve load balancing, to fit into the local memory, and to avoid repeated computations when old services are dropped and new services are added dynamically.

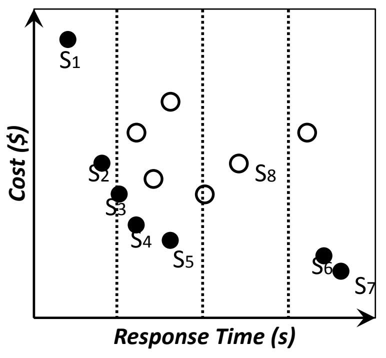
B. Merging in Reduce Computations

Before the process of Reduce, we introduce a middle process (local skyline computation) at Step 2. The reason is that computing skyline services is expensive if the number of candidate services is extremely large. By introducing the middle process, only local skyline services are delivered to the Reduce process at step 3. This will decrease largely the number of services to be processed at the Reduce stage.

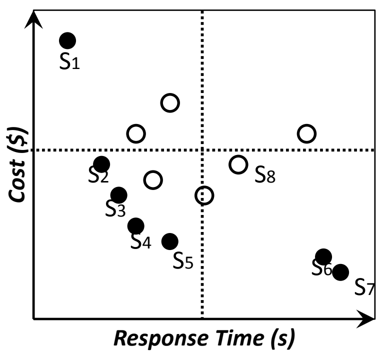
For instance, the local skyline services in Figure 3 involve only $\{s_1, s_2\}$, $\{s_3\}$, $\{s_4, s_5\}$, and $\{s_6, s_7\}$, respectively. This requires only a small percent of all services in the entire data space. Meanwhile, the computing efficiency of skyline services is significantly improved by computing local skyline services in parallel. We choose the BNL algorithm [7] at Step 2 for its simplicity. In Reduce process, we merge all local skyline services and prune the dominated skyline services to get the global skyline services.

III. ANGULAR PARTITIONING OF SKYLINE DATA SPACE

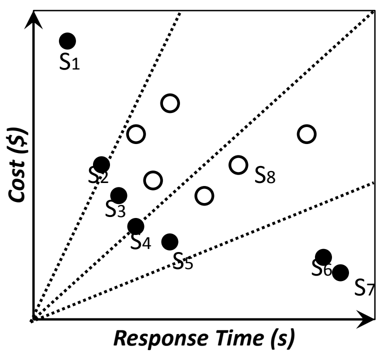
Three MapReduce skyline algorithms are given below based on the three data partitioning schemes. The MR-Dim Algorithm is the simplest one to implement, based on one-dimensional partitioning [15]. The MR-Grid Algorithm is based on grid partitioning, which was first introduced by Wang, et al [16] and Wu, et al [17], separately. The MR-Angle method is a new MapReduce Skyline method first-time proposed here using the angular partitioning as demonstrated in Figure 3(c).



(a) Dimensional partitioning for MR-Dim method



(b) Grid partitioning for MR-grid method



(c) Angular partitioning for MR-angle method

Figure 3. Three data space partitioning methods for three MapReduce skyline query processing schmes

A. *MR-Dim Skyline Algorithm*

The MR-Dim algorithm contains two stages: (1) Partitioning Job, in which we divide the data space into some disjoint subspaces and compute the local skyline of each subspace; 2) Merging Job, in which we merge all local skylines to compute the global skyline. Empirically, the number of partitions is set as (2 $\times$ number of nodes) in the MR-Dim algorithm.

Specially, in the process of implementation, the range of each partition in dimension d is equal to $\frac{V_{max}}{N_p}$, where V_{max} is the maximum value in dimension d, and N_p is the number of partitions. After data space partitioning, the local skyline of each partition is computed using BNL algorithm. In the Merging job, all local skyline services are given the same key in the *Map* process, while they are merged to compute the global skyline in the *Reduce* process.

In MR-Dim, only the QoS parameter values in one dimension are used to do the partitioning [11]. For example, we separate the data space into 4 blocks according to the response time of each service. This approach is easy to implement, while the redundant computations still exist in this case. In addition, this method needs to balance load in the Reduce process.

B. *MR-Grid Skyline Algorithm*

Different from MR-Dim algorithm, multi dimensions are utilized to partition space in MR-Grid algorithm. In the simplest case, two dimensions are utilized (e.g., response time, and cost), and the 2-dimensional data space is divided into 4 partitions by setting the range of partition in each dimension is the half value of the maximum one.

In MR-Grid, there are dominance relationships between partitions. For example, the bottom-left partition dominates the up-right partition in the above simple case, as all services in the up-right partition are dominated by any service in the bottom-left partition. Therefore, we do not have to compute the local skyline of the up-right partition in Step 2, i.e., *Local Skyline Computation.*

In this case, the efficiency of step 2 is improved 25%. However, as the comparability decreases with the increase of the dimension, the improvement of MR-grid in step 2 is limited when the number of dimension is large. When the number of dimension reaches 10, the improvement is even less than 11.08% [18].

C. *MR-Angle Skyline Algorithm*

In this paper, we propose a new angular partitioning method shown in Figure 3. Apparently, this angle-based partitioning reduces many redundant computations and balances the workload, because each subdivided data block involves both high-quality and low-quality data points in the data space.

For instance, each partitioned block (an angular sector) involves some global skyline services: $\{s_1, s_2\}$,$\{s_3\}$,$\{s_4, s_5\}$, and $\{s_6, s_7\}$. The angular partitioning process contains two steps: (1) Mapping the Cartesian coordinate space into a hyperspherical space and (2) Dividing the data space into N sectors according to the angular coordinates.

Consider an n-dimensional QoS data space. The cartesian coordinates of a service is represented by a vector $s = \{v_1, v_2, \ldots, v_n\}$, where v_i refers the value of s in the i^{th} attribute dimension. The hyperspherical coordinate of the service s is specified by a radial coordinate r and *(n-1)* angular coordinates $\{\varnothing_1, \varnothing_2, \ldots, \varnothing_{n-1}\}$ as follows:

$$
\begin{aligned}
r &= \sqrt{v_n^2 + v_{n-1}^2 + \ldots + v_1^2} \\
tan(\varnothing_1) &= \frac{\sqrt{v_n^2 + v_{n-1}^2 + \ldots + v_2^2}}{v_1} \\
&\ldots \\
tan(\varnothing_{n-2}) &= \frac{\sqrt{v_n^2 + v_{n-1}^2}}{v_{n-2}} \\
tan(\varnothing_{n-1}) &= \frac{\sqrt{v_n^2}}{v_{n-1}} \qquad (1)
\end{aligned}
$$

We divide the data space into N sectors by utilizing the angular coordinates $\o_i$. We modify the grid partitioning over the n-1 subspaces defined in Eq.(1). For the 2-dimensional data space in Figure 3, a point s=(x, y) is expressed by

$$r = \sqrt{x^2 + y^2}, tan(\o) = \frac{y}{x} \tag{2}$$

Algorithm 1 MR-Angle for Skyline Query Processing

```
Input: the original data set S
Output: the skyline of S
1: // Generation of local skyline points within each partitioned subspace
2: for all service $s_n$ in dataset S do
3:     compute the coordinates of $s_n$ using Eq.(1)
4:     compute the partition $P_i$ that $s_n$ belongs to based on the service $s_n$'s coordinate value
5:     output $(P_i, s_n)$
6: end for
7: for all partitioned sectors $P_i$ do
8:     compute local skyline $LS_i$ using BNL
9:     output ($P_i$,$LS_i$) in file st
10: end for
11: // Merging of Many Skyline subsets
12: for all service $s_i$ in file st do
13:     output(null,$s_i$)
14: end for
15: compute the global skyline GS using BNL
16: output(GS)
```

The hyperspherical coordinate is used in the *Map* process at Step 1. The Cartesian coordinate is used in the *local skyline computation* at Step 2 and in the *Reduce* process at Step 3. Different from MR-Dim and MR-Grid algorithm, the original Cartesian coordinate-based data should be transformed into hyperspherical coordinate-based data in MR-Angle Algorithm (line 3). The detailed transforming equation is as Equation(1) shows. After getting the hyperspherical coordinate-based data, we get divide the data space according to the angle values of each service (line 4). The local skyline of each partition is then computed by using BNL method (line 7-10). Finally, we merge all local skylines to compute the global skyline (line 12-15).

IV. Complexity Analysis of Angular MapReduce Skyline Method

In this section, we analyze the complexities of MR-Grid and MR-Angle algorithms. As Fig.4 shows, service s_4 is the nearest one to the axes. It can be shown that the first nearest neighbor, i.e., s_4 , is part of the skyline. On the other hand, all the points in the dominance region of s_4 (gray region) can be pruned from further consideration. In this way, the dominated region is pruned, and the first skyline service s_4 is selected. Then, the left regions are computed recursively.

The dominance ability of skyline services is critical to the efficiency of skyline computation. For example, if the dominance ability of s_4 is stronger, more services will be pruned, which leads to higher efficiency. Therefore, dominance ability is selected as the evaluation metric of algorithm complexity. The dominance ability of skyline service s_i is defined by the ratio $D_{s_i} = \frac{Num_{s_i}}{Num_{all}}$, where Num_{s_i} is the number of services dominated by s_i, and Num_{all} means the number of all services. In this paper, to make the comparison between two algorithms more intuitive, the dominance ability of s_i is defined by the ratio $D_{s_i} = \frac{Area_{s_i}}{Area_{all}}$, where $Area_{s_i}$ is the area dominated by service s_i in the partition that s_i belong to, and $Area_{all}$ is the area of the partition that s_i belongs to.

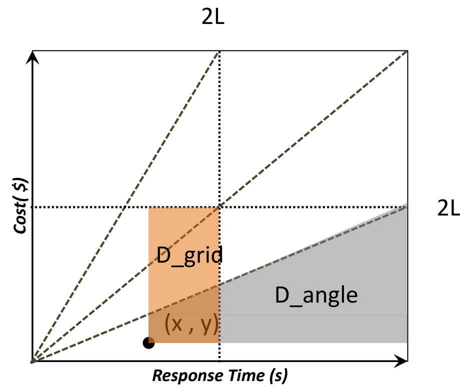


Figure 4. MR-Grid and MR-Angle partitioning

In Fig.4, we compare MR-Grid and MR-Angle approach to divide the data space into 4 partitions, respectively. The data space is a square with $2L$ long sides. The area of the overall data space is $4L^2$, while the area of each partition is L^2. Given a skyline service s with coordinate (x,y), and it belongs to the partition close to the axes as the most case.

Theorem 1: If MR-Angle approach is employed to do partition, the dominance ability of service *s* is as follows:

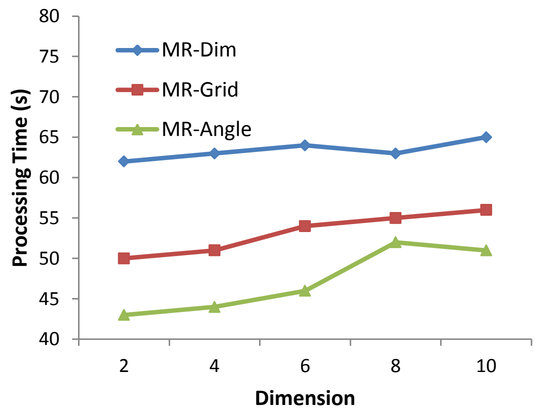
$$D_s^{angle} = \frac{L^2 - \frac{x^2}{4} - (2L - x)y}{L^2} \tag{3}$$

Proof: The dominance region of service s is the gray region if MR-Angle is employed. The area of the gray region can be computed by using the area of a partition to subtract the area of non-dominance regions. It can be presented as $L^2 - \frac{x^2}{4} - (2L - x)y$. And the area of partition that s belongs to is still L^2 while using MR-Angle approach. Therefore, the dominance ability is $\frac{L^2 - \frac{x^2}{4} - (2L-x)y}{L^2}$. ■

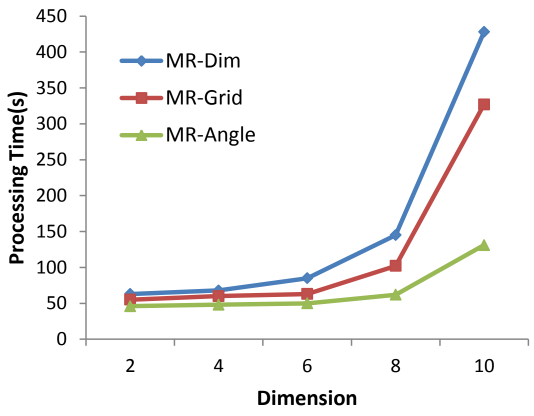
Theorem 2: The dominance ability of MR-Angle outperforms the MR-Grid method by the following quality:

$$\Delta D = D_s^{angle} - D_s^{grid} \geq \frac{x}{2L^2}(L - \frac{x}{2}) \tag{4}$$

Proof: Similar to Theorem 1, the dominance of MR-Grid method is $\frac{(L-x)(L-y)}{L^2}$. To compare the dominance



(a) Small cardinality (N=1000)



(b) Large cardinality (N=100,000)

Figure 5. Processing time of angular MapReduce Skyline method against the dimensional and grid portioned methods

ability of MR-Grid and MR-Angle, we introduce another symbol ΔD which is presented as $D_s^{angle} - D_s^{grid}$. Because $y \leq \frac{x}{2}$,we have

$$\begin{aligned}\Delta D &= \frac{L^2 - \frac{x^2}{4} - (2L-x)y}{L^2} - \frac{(L-x)(L-y)}{L^2} \\ &= \frac{1}{L^2}(-\frac{x^2}{4} - yL + xL) \\ &\geq \frac{1}{L^2}(-\frac{x^2}{4} - \frac{x}{2}L + xL) \\ &= \frac{x}{2L^2}(L - \frac{x}{2})\end{aligned}$$

■

From Theorem 2, the dominance ability of MR-Grid approach is always weaker than the one of MR-Angle approach. This result is also the same in high-dimensional case. Therefore, we can draw the conclusion that the performance of MR-Angle approach is better than MR-Grid approach.

V. HADOOP EXPERIMENTS AND RESULTS

In this section, we evaluate the performance of three MapReduce based skyline algorithms which are used for skyline services computation. Then, the scalability of our proposed MR-Angle method is evaluated.

A. Experiment Setup and Evaluation Metrics

We have conducted our experiments based on one real dataset. It is based on the QWS (details in http://www.uoguelph.ca/~qmahmoud/qws/index.html) dataset [19], which comprises measurements of nine QoS attributes over 10,000 real-world Web services. The majority of Web services were obtained from public sources on the Web including UDDI, search engines and service portals.

Considering the rapidly development of Web services, we extend the size of QWS dataset by randomly generating QoS values which are limited to a narrow range following the distribution of the QWS dataset. The number of services is finally extended to 100,000 and 10 QoS attributes are selected for our experiments.

All experiments are implemented in Java, and the MapReduce related experiments are run on Hadoop 0.20.2 framework. Specifically, each server has a Intel Core Duo E7400 2.99GHz CPU, 3.25G main memory, with Ubuntu 10.9 OS and 1G memory allocated to JVM.

To study the performances of MapReduce based skyline algorithms for skyline services computation, we evaluate three different approaches:

- **MR-Dim**: MR-Dim skyline algorithm is adopted in this approach, details in Section 3.1. The implementation of MR-Dim can be found in [11].
- **MR-Grid**: Grid-based skyline algorithm is adopted in this approach.
- **MR-Angle**: Angle-based algorithm (Algorithm 1) is adopted in this approach. The difference among the above three approaches is mainly the partitioning approach.

B. MapReduce Skyline Query Processing Time

Figure 5 shows the processing time used for selecting the optimal or suboptimal skyline services, as the *service cardinality* (the number of candidate services) changes from rather small (1,000 in Fig5.(a)) to very large (100,000 in Fig.5(b)). In both cases, the *attribute dimension* (the number of QoS attributes being considered) increases from 2 to 10.

Figure 5(a) shows the results of three methods over 1,000 service choices. The processing times of the MR-grid and MR-dim methods are 6 $\sim$ 16% and 18 $\sim$ 45% higher, respectively, than using the MR-angle method. The other two methods increase rather flatly (within 12%), as the dimension increases. The MR-angle method shows at most 20% increase in processing time as more dimensions are involved in the evaluation process.

The advantage of MR-Angle increases much more sharply, as the cardinality increases to 100,000 services. As plotted in Fig 5(b), the MR-Angle method outperforms the the MR-grid method with 2% $\sim$ 170% reduction in processing time as the dimension increases from 2 to 10. The MR-dim method performs the worse with 3% $\sim$ 230% longer processing times than using the MR-angle method as the dimension increases.

In summary, with a large service cardinality of 100,000 and higher dimension of 10 attributes, our MR-angle method performs 1.7 and 2.3 times faster than using the MR-grid and MR-dim methods, respectively. These results clearly demonstrate the advantage of our MapReduce skyline method using the angular partitioning of a large data space over 10 attribution dimensions.

C. Breakdown of Processing Times in Map and Reduce Operations - Result on the Scalability

The number of servers used does affect the cluster performance greatly. We consider a large data space of 100,000 service data points. There are 10 dimensions of performance attributes. The server cluster used increases from 4 to 8, 12, 16, 20, 24, 28 and 32 servers. Figure 6 shows the processing time of the MR-Angle method plotted against the increase of servers used.

The processing time decreases sub-linearly with respect to more servers used. When the number of servers exceeds 24, the speedup improvement becomes saturated gradually. This processing time consists of mainly two parts: the Map Time and Reduce Time. The Map time reduces almost flatly with more servers used. The Map time doest get some speedup with more servers used. In other word, as more servers are used, the drop in Map time contributes the most to the scalability of the MR-angle method.

Compared with 4 servers used, the processing time is reduced only 10%, when 8 servers are used. The processing drops from 230 sec to 130 sec with 70% improvement as the number of servers used increases from 4 to 32. The sectioned bar diagram in Figure 6 clearly shows the scalability of using the MR-angle method to accelerate the Reduce process in skyline selection of optimal web services. The advantage increases with larger data space and more attribute dimensions adopted.

VI. OPTIMALITY OF LOCAL SKYLINE SELECTION

The major reduction in query processing time comes from lowering the Reduce time in our MR-angle method. The Reduce time is consumed mainly from merging the local skylines in various partitions to become the global skyline choices. In a given partition, if most local skyline services are also globally optimal, that partition is considered very efficient or effective in finding the optimal web service. We define a local skyline optimality to express the percentage of skyline services selected locally that are also qualified as a skyline selection in the global data space.

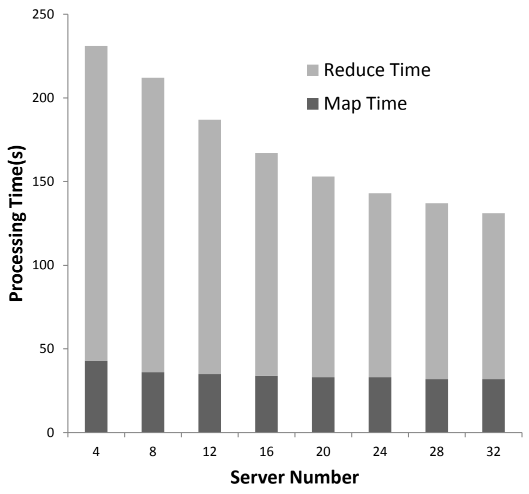


Figure 6. Breakdown of the processing time of MapReduce-angle method against increasing number of servers used in skyline selection of optimal services out of 100,000 web services

$$LocalSkylineOptimality = \frac{1}{N} \sum_{1<i<N} \frac{|sky_i \bigcap sky_{global}|}{|sky_i|} \tag{5}$$

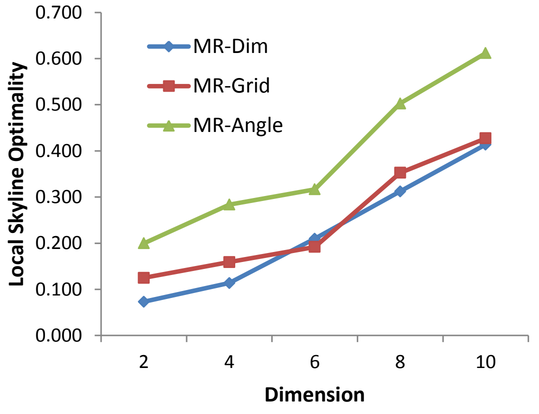
where the sky_i means the set of local skyline services in the partition, and the sky_{global} means the set of global skyline services. As the distribution of global skyline services in different partition is different, we use the average value of each partition to reflect local skyline optimality.

Figure 7 shows the local skyline optimality of three MapReduce skyline methods. The higher is the optimality fraction, the better is the performance towards optimal choice. Figure 7(a) shows the experiment result over a small cardinality of 1,000 services, while the cardinality of dataset in Fig. 7(b) is 100,000. In general, the local skyline optimality of all three methods increases with the increase of dimension (the. number of QoS attributes), because the increase in dimensionality decreases the comparability between service pairs.

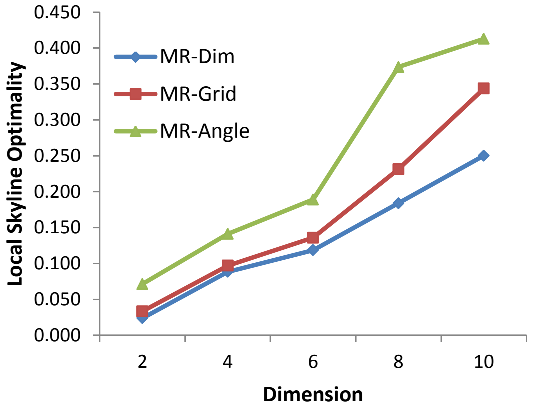
In Fig. 7(a), the local skyline optimality of MR-Angle outperforms MR-Grid and MR-Dim in all dimensions. The local skyline section optimality of MR-Angle achieves the maximum value 0.61 meaning 61% local skyline services are also the globally optimal. Again the MR-Dim method is the lowest in reaching optimality. The MR-grid method is only slightly better then the MR-Dim method. In Fig.7(b), the gaps between MR-Angle method and the other two are even greater.

VII. CONCLUSION

Skyline query processing is often used in database operation or in finding optimal web services with guaranteed



(a) Small cardinality (1000 services)



(b) Large cardinality (100,000 services)

Figure 7. The optimality of three MapReduce methods for Skyline query processing

QoS. However, the process is very time consuming when the service data space is very large with too many performance attributes to be evaluated. In this paper, we propose a new angular partitioning method to apply the MapReduce model for fast Skyline selection of optimal web services.

Our MapReduce angular method scales well with the increase of both attribute dimensionality and data-space cardinality. The newly defined performance metric is very powerful to assess the local optimality of selected skyline services. By experimenting over 10,000 real-life web service applications in 10 attribute dimensions, we find that the angular-partitioned MapReduce method is 1.7 and 2.3 times faster than the dimensional and grid partitioning methods, respectively.

Our method has a 60% higher probability to reach the local optimality than the other two methods. These results are very encouraging to select optimal web services in real-time out of a large number of web services. Our improved MapReduce skyline method is also applicable to cloud service application, which is under extended work by our joint research team at Zhejiang University and University of Southern California. The new MapReduce skyline algorithm and reported results are very encouraging to enable cloud computing and mashup services in real-time out of a large number of web services.

Acknowledgment

This research was supported partially by the 863 Research Program in China under Grant No. 2008AA01Z141, the National Natural Science Foundation of China under Grant No.60873224, Science and Technology Program of Zhejiang Province under Grant No. 2008C03007. Kai Hwang wants to acknowledge the support of this work by the Ministry of Science and Technology of China under National 973 Basic Research Grants No.2011CB302505 at Tsinghua University, Beijing. He is also grateful to the financial support by Cao Guang Biao Foundation at Zhejiang University during 2005-2007 and by the Intellectual Venture endowment at Tsinghua University.

References

[1] E. Al-Masri and Q. Mahmoud, "Qos-based discovery and ranking of web services," *Int'l Conf. on Computer Communications and Networks*, pp. 529–534, 2007.

[2] V. Cardellini, E. Casalicchio, V. Grassi, and F. L. Presti, "Flow-based service selection for web service composition supporting multiple qos classes," *Int'l Conference on Web Services*, pp. 743–750, 2007.

[3] V. Diamadopoulou, C. Makris, Y. Panagis, and E. Sakkopoulos, "Techniques to support web service selection and consumption with qos characteristics," *Journal of Network and Computer Applications*, vol. 31, no. 2, pp. 108–130, 2008.

[4] S. Ran, "A model for web services discovery with qos," *ACM SIGecom Exchanges*, pp. 1–10, 2003.

[5] Y. Liu, A. H. Ngu, and L. Zeng, "Qos computation and policing in dynamic web service selection," *International World Wide Web Conference (WWW)*, pp. 66–73, 2004.

[6] C. Makris, Y. Panagis, E. Sakkopoulos, and A. K. Tsakalidis, "Efficient and adaptive discovery techniques of web services handling large data sets," *Journal of Systems and Software*, vol. 79, no. 4, pp. 480–495, 2006.

[7] S. Borzsonyi, D.Kossmann, and K.Stocker, "The skyline operator," *International Conference on Data Engineering (ICDE)*, pp. 421–430, 2001.

[8] M. Alrifai, D. Skoutas, and T. Risse, "Selecting skyline services for qos-based web service composition," *Int'l Conf. on World Wide Web (WWW)*, pp. 11–20, 2010.

[9] H. Han, H. Jung, S. Kim, and H. Yeom, "A skyline approach to the matchmaking web service," *International Symposium on Cluster Computing and the Grid*, 2009.

[10] Q. Yu and A. Bouguettaya, "Computing service skyline from uncertain qows," *IEEE Transaction on Service Computing*, vol. 3, no. 1, pp. 16–29, 2010.

[11] L. Pan, L. Chen, and J. Wu, "Skyline web service selection with mapreduce," *International Conference on Computer Science and Service System*, 2011.

[12] L. Chen, J. Wu, S. Deng, and Y. Li, "Service recommendation: Similarity-based representative skyline," *IEEE World Congress on Services*, pp. 360–366, 2010.

[13] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[14] J. Ekanayake, H. Li, B. Zhang, T.Gunarathne, S.Bae, J.Qiu, and G. Fox, "Twister: A runtime for iterative mapreduce," *Proceedings of the First International Workshop on MapReduce and its Applications, in conjuction with ACM HPDC 2010*, pp. 810–818, 2010.

[15] W. tilo Balke, U. Gntzer, and J. X. Zheng, "Efficient distributed skylining for web information systems," *Extending Database Technology*, pp. 256–273, 2004.

[16] S. Wang, B. C. Ooi, A. K. H. Tung, and L. Xu, "Efficient skyline query processing on peer-to-peer networks," *International Conference on Data Engineering (ICDE)*, pp. 1126–1135, 2007.

[17] K. Deng, X. Zhou, and H. T. Shen, "Multi-source skyline query processing in road networks," *International Conference on Data Engineering*, pp. 796–805, 2007.

[18] B. Zhang, S. Zhou, and J. Guan, "Adapting skyline compuation to the mapreduce framework: Algorithms and experiments," *DASFAA Workshop, Lecture Notes in Computer Science Series 6637*, pp. 403–411, 2011.

[19] A. Eyhab and Q. H. Mahmoud, "Discovering the best web service," *International World Wide Web Conference*, pp. 1257–1258, 2007.

[20] Q. Y. Feng, K. Hwang, and Y. F. Dai, "Rainbow product ranking for upgrading e-commerce," *IEEE Internet Computing*, pp. 72–80, 2009.

[21] D. Kossmann and F. Ramsak, "Shooting stars in the sky: an online algorithm for skyline queries," *International Conference on Very Large Data Base (VLDB)*, pp. 275–286, 2002.

[22] H. T. Kung, F. Luccio, and F. P. Preparata, "On finding the maxima of a set of vectors," *Journal of the ACM*, vol. 22, no. 4, pp. 469–476, 1975.

[23] X. Lin, Y. Yuan, Q. Zhang, and Y. Zhang, "Selecting atars: The k most representative skyline operator," *International Conference on Data Engineering (ICDE)*, pp. 86–95, 2007.

[24] T. Nykiel, M. Potamias, and M. Mishra, "Mrshare: Sharing across multiple queries in mapreduce," *Proc of Very Large Data Bases*, vol. 3, no. 1, pp. 494–505, 2010.

[25] D. Papadias, Y. Tao, G.Fu, and B.Seeger, "An optimal and progressive algorithm for skyline queries," *International Conference on Management of Data (SIGMOD)*, pp. 467–478, 2003.

[26] K. S. Candan, P. Nagarkar, M. Nagendra, and R. Yu, "Rankloud: a scalable ranked query processing framework on hadoop," *International Conference on Extending Database Technology (EDBT)*, pp. 574–577, 2011.

[27] S. Sioutas, E. Sakkopoulos, L. Drossos, and S. Sirmakessis, "Balanced distributed web service lookup system," *Journal of Network and Computer Applications*, vol. 31, no. 2, pp. 149–162, 2008.

[28] Y. Taher, D. Benslimane, M. Fauvet, and Z. Maamar, "Towards an approach for web services substitution," *International Database Engineering and Applications Symposium*, pp. 166–173, 2006.

[29] K. Tan, P. Eng, and B. Ooi, "Efficient progressive skyline computation," *International Conference on Very Large Data Base*, pp. 301–310, 2001.

[30] Y. Wang and J. Vassileva, "Toward trust and reputation based web service selection: A survey," *International Transactions on Systems Science and Applications*, vol. 3, no. 2, pp. 118–132, 2007.

[31] T. Yu, Y. Zhang, and K. J. Lin, "Efficient algorithms for web services selection with end-to-end qos constraints," *ACM Trans. on the Web*, vol. 1, no. 1, pp. 1–26, 2007.

[32] L. Zeng, B. Benatallah, A. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "Qos-aware middleware for web services composition," *IEEE Trans. on Software Engineering*, vol. 30, no. 5, pp. 311–327, 2004.

[33] L. Zeng, Y. Ren, M. Li, and K. Sakurai, "Spse: A flexible qos-based service scheduling algorithm for service-oriented grid," *International Parallel & Distributed Processing Symposium (IPDPS)*, pp. 1–8, 2010.