

Smart Separated Software Execution System for Software as a Service

Su-Min Jang, Won-Hyuk Choi, Won-Young Kim

Cloud Computing Research Department, Electronics and Telecommunications Research Institute, Daejeon, Korea

jsm@etri.re.kr, whchoi@etri.re.kr, wykim@etri.re.kr

Abstract— In this paper, we propose a separated software execution system (SSES) for software as a service (SaaS). The main difference of the proposed SSES and virtual desktop solutions such as Citrix XenDesktop is that 3D rendering commands or graphic commands do not be executed at the server part but they are delivered to client via the Internet. This mechanism brings to reduces server's overhead occurred from multi-users service on 3D rendering applications. In performance experiment, the proposed SSES gets a more stable frame rate than existing virtual desktop solutions on multiple clients. In particular, the SSES shows very high performance in term of frequent screen changes. Also, as another issue, we propose a smart mechanism that servicing software list is auto-updated by processing software usage data.

Keywords— software as a service, on-demand service software, cloud computing.

I. INTRODUCTION

Most of recent business applications use delivery model of software as a service (SaaS), because it has many strong points. Main benefits of the SaaS are as follows.

- When a company subscribes to a SaaS application, its fee will depend on total used time without losing a large initial investment.
- Traditional licensed software typically has to wait for the next release. However, SaaS application can start using it, as soon as a new feature appears in the application.
- SaaS application ensures a convergence of interest between customer and service provider.
- The subscription of SaaS application can be canceled or changed at any time.
- SaaS application needs shorter deployment time than a traditional one, because the SaaS application is already up and running on the service server.

We know that virtual desktop solution is a well-known service delivery model and is used at various applications. However, it is not efficient as multiple user service. The reason is that server has a high overhead on 3D rendering service for multiple users, because the server executes all related data processing commands and graphic commands for 2D and 3D and additionally sends the screens data of their results. To solve this problem, we propose a new system denoted as separated software execution system (SSES) for software as a service. The proposed SSES can stably support multiple users at the same time, through executing graphic commands at client part.

In addition, most of service software offered from SaaS server to client computers selected by the service provider or system administrator and the service software are manually updated by them. However, mid or large scale companies or groups need to use the service software reflected their feedback. Therefore, we propose a new smart updating mechanism for solving the problem.

This paper is organized as follows. Section 2 presents the related work. Section 3 proposes separated software execution system (SSES). Section 4 presents smart auto-updating mechanism for software as a service. Section 5 contains various experimental evaluations. Section 6 concludes the paper with directions for future works.

II. RELATED WORKS

In recent, with technic development of cloud computing and virtual desktop, related software market is extended. The commercial software are Citrix XenDesktop [1], Microsoft RDP[2], VNC[3] and THINC[4]. We often use Microsoft Remote Desktop Protocol (RDP) for delivery model of Software as a Service (SaaS). The RDP provides remote display and input capabilities over network connections for Windows-based applications running on a server. The RDP supports different types of network topologies and a multiple-channel capable protocol that allows for separate virtual channels for carrying device communication and server screen. It delivers encrypted client mouse and keyboard information to server. Also, RDP provides an extensible base and supports separated channels for data transmission and provisions for multipoint transmission.

Well-known another solution of SaaS is using the Citrix XenDesktop. Fig. 1 shows the overall workflow of Citrix XenDesktop. All service applications are installed and are run on its server. The server sends screens of application to various client terminals such as phone, desktop PC, notebook, PDA and mouse clicks and keystrokes occurred from client terminal are delivered to the server. The merit of Citrix XenDesktop is that applications can be accessed from most any device and covering both wired network and wireless network. In particular, the HDX (High Definition User Experience) of Citrix XenDesktop enables to optimize the performance of graphics-intensive 2D and 3D and media rich applications. This capability offers the ability to leverage both software and hardware-based rendering to assist with compression, performance, and network efficiency.

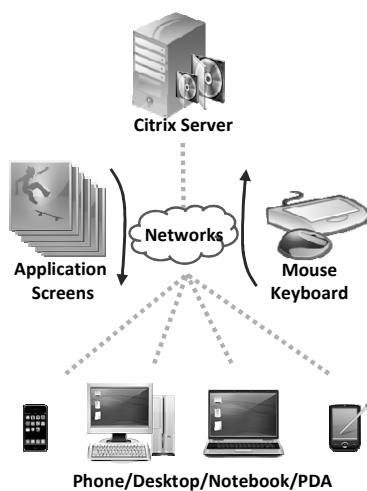


Figure 1. The workflow of Citrix XenDesktop

In other related services, OnLive[5] is specialized SaaS for game application called as gaming-on-demand platform. Since the game is executed on the OnLive server, the games are synchronized, rendered, and stored on remote servers and its screens just delivered to client computer via the Internet. Because OnLive needs a 3 Mbit/s connection meets the minimum system requirements, low-performance computer, as long as it can play video, may be used to play any kind of game.

Above-mentioned services, to be seen 3D rendered screens in low performance client, compress and send result screens executed in server [6][7][8]. In other words, they use exclusively the GPU of graphic card of the server part for processing 3D rendering commands. Such mechanism is hard to service and manage extending many users.

III. SEPARATED SOFTWARE EXECUTION SYSTEM

The difference of the proposed SSES and previous related services is that the SSES executes 3D graphic rendering commands and graphic commands at the client computer. In other words, graphic commands do not execute on remote server but deliver via the Internet to client. Frequent screen

changes or complex 3D rendering will cause an overload on the server.

Fig. 2 shows the workflow of separated software execution system (SSES). Its example is the status of calling CreateWindowEx() command from the application. Detail order for processing the CreateWindowEx() command in SSES is following steps.

1. SSES server hooks graphic commands (CreateWindowEx()) of application.
2. SSES server delivers them via the Internet to SSES client.
3. SSES client execute received graphic commands (CreateWindowEx()).
4. SSES client sends mouse and keyboard information (return value of executed graphic command, other needed data) to SSES server.
5. A virtual window on SSES server is created by CreateWindowEx() command.
6. Return virtual handle for real application of SSES server.

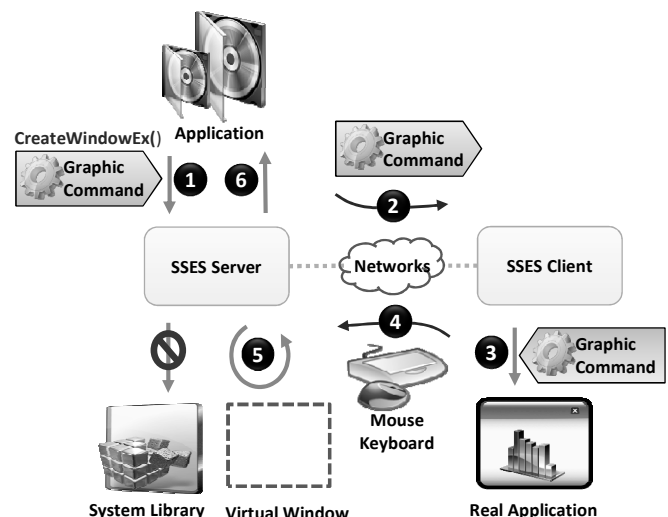


Figure 2. The workflow of SSES

Fig. 3 shows the hooking mechanism of separated software execution system (SSES). It is a type of using wrapper libraries. When creating a wrapper, we make our own version of a library that an application loads, with all the same functionality of the original library that it will replace. When executed application on SSES server calls graphic commands such as CreateWindowEx(), Direct3DCreate9 and D3D10CreateDevice, our SSES server hooking module executes our wrapper library instead of original library.

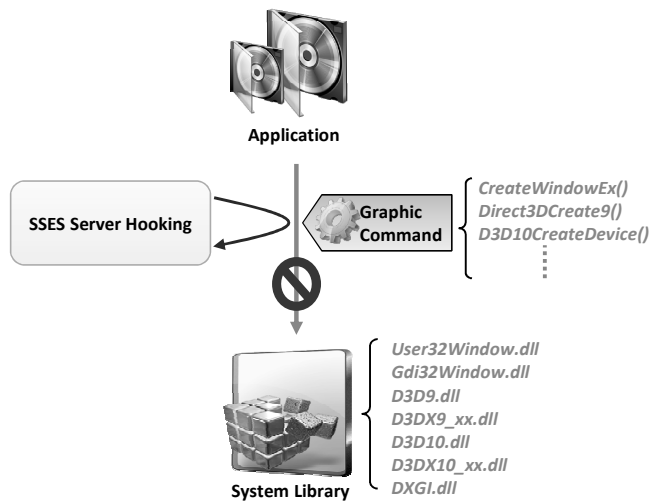


Figure 3. The hooking mechanism on SSES server

There are two working processes for separated software execution. One process is for processing data-intensive commands at server part and another process is for processing graphic commands at client part. For it, each process uses independent objects such as Windows handles or interfaces and callback functions, but the independent objects have been managed with a same handle value in order to interact between two processes like Fig. 4.

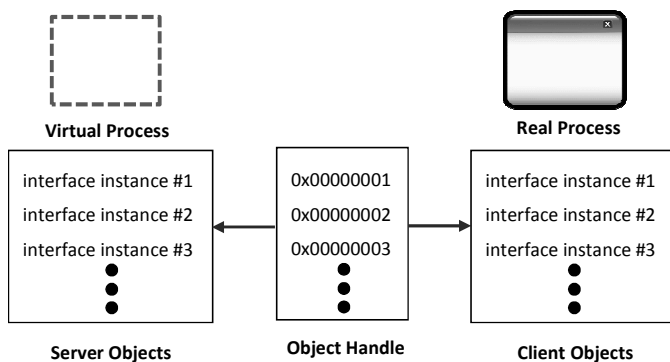


Figure 4. The management mechanism on common objects

The SSES has various modules such as hooking module, common object management module, processing callback function module, network protocol module, etc. Like this, because the 3D rendering commands or graphic commands are executed at the client part, the SSES reduces server's overhead on 3D rendering applications.

IV. SMART UPDATING MECHANISM

Another issue of this paper is smart updating mechanism for SSES. Most of SSES or similar service systems are composed of several servers for supporting a lot of users. The server computers of them are placed at the special area such as internet data centers (IDC), because many users demand the quality-of-service (QoS) of the service software, but it is hard that the service provider continuously selects appropriate service software at the appropriate time and updates them.

Even if service software of SSES were initially well chosen, there is a problem which unwanted software potentially remains. To solve the problems, we collect user data on software usage and we use the collected user data in order to select appreciated software. To determine best service software, we use skyline query [10] on multiple attributes of the collected user data such as user usage time, user usage count.

Fig. 5 shows the example of a skyline query. A general example is a list of hotels, each of which has two attributes such as the distance from some beach and the price of a hotel for on-line hotel booking system. Clients may request skyline query for hotel booking, which is as cheap as possible while also near the beach. The hotel H_1 dominates the three hotels, H_2 , H_3 , and H_6 since H_1 has smaller coordinates than them in both dimensions. In addition, H_1 , H_4 , and H_7 not dominated by any other hotels are the skyline for the requested skyline query. Note that the hotels of the retrieved skyline points are more meaningful than the dominated hotels, but the skyline points do not have priority. In case of high-dimensional data, as the number of dimensions increases, the number of skyline points increases substantially [20], [21]. The reason is that the high dimensions reduce the probability that some points dominate another point. When there are too many skyline points, they are not useful in recommendation services.

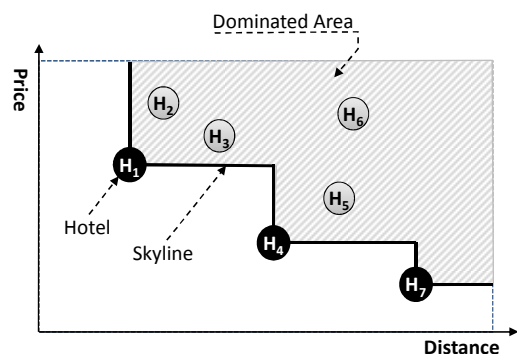


Figure 5. The example of a skyline query

We use top- n skyline query for it. The top- n skyline query is to find more interesting n points among skyline results retrieved from multiple attributed data. Some researchers have studied to get more important or representative data from skyline points according to user preference. X. Lin et al. [18] investigated the problem of computing the top- n representative skyline points. This was the first attempts to develop efficient and scalable algorithms to solve the problem.

It presents a dynamic programming based algorithm for a 2d-space, but it is difficult to handle the problem for space with dimensionality $d \geq 3$. Yufei Tao et al. [19] proposed to solve the problem which the formulation of top- n representative skyline points [18] sometimes cannot capture the skyline contour accurately. It is the distance-based representative skyline, and it also develops several algorithms for computing distance based representative skylines. However, it addressed that the optimal representative skyline computation for higher dimensional spaces is NP-hard. Chee-Yong Chan et al. [11, 19] introduced skyline frequency as a measure of interestingness for points in the data set. The skyline frequency of a point measures the number of subspaces in which the point is a skyline. To get the top- n skyline query, we use well-known k -dominant skyline query [11]. The notation and related terms for k -dominant skyline are defined as follows.

TABLE 1. NOTATIONS AND DEFINITIONS

Notation	Definition
S	The data set (Data points)
p, q	data points in S
D	The dimension set (Attributes of data points)
d_i	i th dimension ($1 \leq i \leq D $)
$p(d_i)$	i th dimension value of data point p

Dominate

Let p and q be two data points, where we denote the values of p and q on dimension d_i as $p(d_i)$ and $q(d_i)$. For a dimension set D , p dominates q if $\forall d_i \in D, p(d_i) \leq q(d_i)$ and $\exists d_j \in D, p(d_j) < q(d_j)$ ($1 \leq i, j \leq |D|$).

Skyline points

The skyline points are made up of those points that are not dominated by any other points.

k -Dominate

A point p k -dominates another point q if p is better than or as good as point q and is better in at least one of k -dimensional subspaces ($k \leq |D|$).

k -Dominant skyline

The k -dominant skyline is defined as those points that are not k -dominated by any other points.

The k -dominant skyline includes points that are not dominated by any other points in terms of any k attributes. As the value k is smaller, the k -dominant skyline means more important and meaningful data. Using this principle of k -dominant skyline helps to determine more important and meaningful software. In recent years, recommendation services have explosively grown with the development of the world-

wide-web and the emergence of e-commerce, and they have been used in various applications. Many people use a top- n query in order to find or select more important and meaningful data from very large amounts of data. The most used query in the recommendation systems is top- n query. Note that several attribute values of multidimensional (multiple-attributed) data should be converted into one measure parameter using the weight of each attribute, in order to generate the ranks. In step 4, service software list updater module finally updates or deletes old service software. Software provider module of Software providing site needs to install automatically the selected software to SSES server.

Fig. 6 shows the workflow of smart updating mechanism on SSES server. We can see that users of SSES service can generally use two types of software such as normal software and service software. The service software of SSES client is offered by software provider of SSES server. On the other hand, its normal software is downloaded from software providing sites and it is directly installed and executed by users. To collect and save user software usage information, SSES Client includes two modules which are the user software usage information collector module and its sender module which saves them on the data server. We save software name, software usage time, software install-time, software uninstall-time and user profile. Also, we reprocess the saved basic data to get average usage time, total usage time, maximum usage time and minimum usage time of the software, and save them on the data server too. In step 3 of Fig. 6, the role of service software list selector module is to select top- n software using multiple attributed data of stored data server.

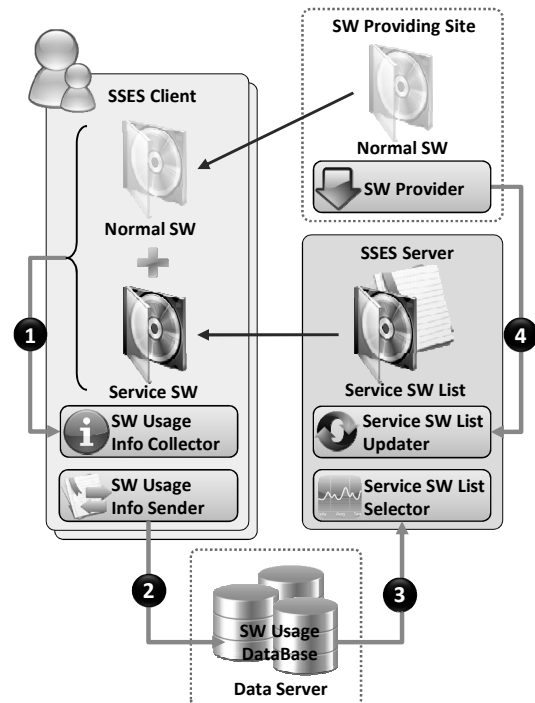


Figure 6. The workflow of smart updating mechanism for SSES

V. PERFORMANCE EVALUATION

In order to show the superiority of the proposed SSES, we compared it with Microsoft RDP and HDX of Citrix XenDesktop. We use SoftParticle, HDAO and GPUBoids sample program as test programs. Fig. 7 shows screenshots of SoftParticle, HDAO and GPUBoids sample program. GPUBoids sample program create the illusion of flocks of birds or bugs swarming around a light or fallen comrade. HDAO sample program presents an innovative technique for achieving high-definition ambient occlusion (HDAO). It utilizes making use of the new shader model and it is used for the condition of fixed screens. On the other hand, SoftParticle sample program is a common program for creating volumetric effects in games. They can be used to simulate such things as fire, clouds, smoke, dust, glowing projectiles, magic spells, and so on. It is used for the condition of frequent change screens.

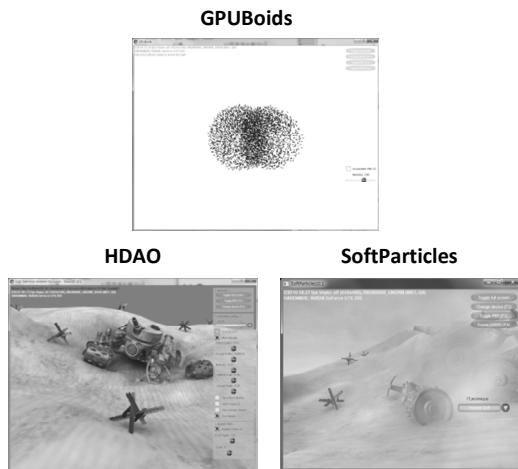


Figure 7. The test programs for performance evaluation

We evaluate FPS (frame per second) of RDP and SSES in term of single user like Fig. 8.

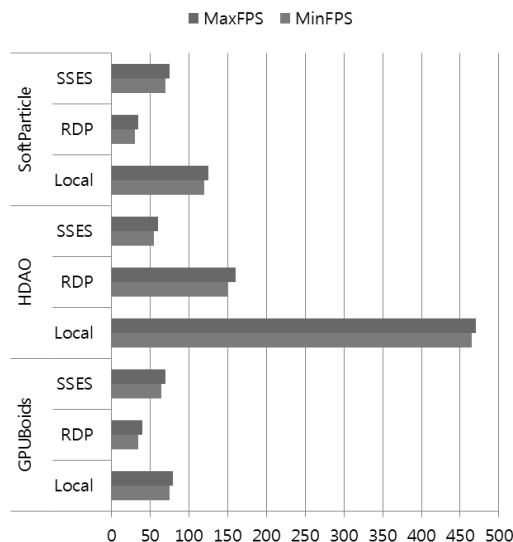


Figure 8. FPS evaluation of SSES vs RDP / Local

For it, we execute client and server program in same one computer. The specification of the used computer is Intel Core i7 3.20GHz, Memory 4G, NVIDIA GeForce GTX 480. When frequent change screens or complex 3D rendering jobs are needed at GPUBoids and SoftParticle, the FPS of SSES is better than that of RDP. On the other hand, we can see that RDP is better than SSES at HDAO (the condition of fixed screens). Local evaluation means that each sample program locally executes without additional modules.

Fig. 9 shows the FPS evaluation for multiple users. In this evaluation, we separately execute client program and server program in different computers. The specification of server computer is Intel Core i7 2.8GHz, memory 4G, NVIDIA GeForce GTS 250 and that of three client computes is Intel Core i7 3.20GHz, memory 12G, NVIDIA GeForce GTX 480.

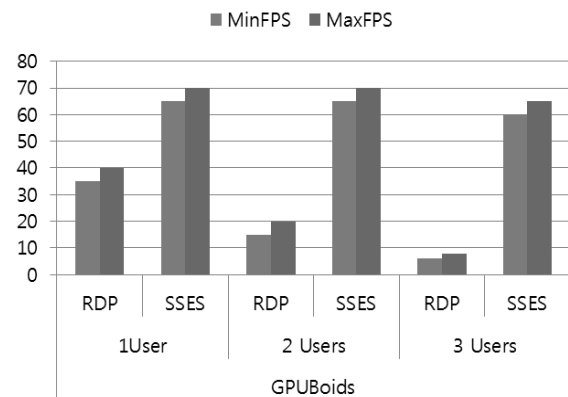


Figure 9. FPS evaluation of SSES vs RDP for multiple users

In Fig. 10, we evaluate FPS of HDX and SSES in terms of separately executing client program and server program. HDX server is Intel Core i7 2.93GHz, memory 16G, NVIDIA Quadro FX 3800 (\$750~\$840). On the other hand, SSES server is Intel Core i7 2.8GHz, memory 4G, NVIDIA GeForceGTS 250 (\$160~\$200). Each client of HDX and SSES is Intel Core i7 2.8GHz, memory 4G, NVIDIA GeForceGTS 250. The FPS of SSES is better than that of HDX in GPUBoids (the condition of frequent change screens). In case of SoftParticle, the FPS of SSES is similar to that of HDX, despite the condition that SSES server computer (\$160~\$200) has low specifications.

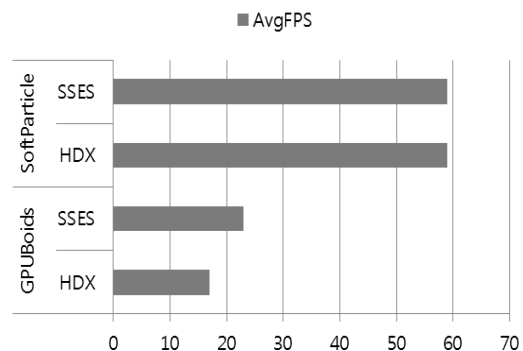


Figure 10. FPS evaluation of SSES vs HDX

VI. CONCLUSION

Software as a Service (SaaS) is emerged as a new technique for more convenient and economical way of using software. Therefore, many business applications use the delivery model of SaaS. To improve the capability of the SaaS related applications, we have devised the smart separated software execution system (SSES) which is more effective on service for multiple users. In performance experiment, the proposed SSES shows a good scalability which it gets more stable frame rate than existing virtual desktop solutions under various conditions. As another issue, we design smart updating mechanism for SSES and we convince ourselves that it helps to automatically provide better service software for users. In the near future, the proposed smart updating mechanism will be applied to various real applications and will be evaluated in various environments.

ACKNOWLEDGMENT

This work was supported by the Technology Innovation Program [10035185, Development of a Separated Software Execution Technology for Software Service] funded by the Ministry of Knowledge Economy (MKE, Korea)

REFERENCES

- [1] Citrix XenDesktop (<http://www.citrix.com/xendesktop>)
- [2] Brian Craig Cumberland, Gavin Carius and Andrew Muir, "Microsoft Windows NT Server 4.0, Terminal Server Edition: Technical Reference." Microsoft Press, Redmond, Wash, 1999.
- [3] Virtual Network Computing (<http://www.realvnc.com>)
- [4] Ricardo A. Baratto, Leonard N. Kim, Jason Nieh, "THINC: a virtual display architecture for thin-client computing," Proceedings of the twentieth ACM symposium on Operating systems principles, October 23-26, 2005, Brighton, United Kingdom
- [5] OnLive (<http://www.onlive.com>)
- [6] Fabrizio Lamberti and Andrea Sanna, "Streaming-based solution for remote visualization of 3D graphics on mobile devices," IEEE Transactions on Visualization Computer Graphics, 13 (2) (2007), pp. 247-260
- [7] David Koller, Michael Turitzin, Marc Levoy, Marco Tarini, Giuseppe Crocchia, Paolo Cignoni, Roberto Scopigno, "Protected interactive 3D graphics via remote rendering," ACM Transactions on Graphics (TOG), v.23 n.3, August 2004
- [8] Lars Ailo Bongo, Grant Wallace, Tore Larsen, Kai Li, and Olga Troyanskaya, "Systems support for remote visualization of genomics applications over wide area networks," In Proc. of GCCB'06. LNBI 4360, 2006.
- [9] Micah Dowty, Jeremy Sugerman, "GPU Virtualization on VMware's Hosted I/O Architecture." In First Workshop on I/O Virtualisation, 2008.
- [10] S. Borzsonyi, D. Kossmann, and K. Stocker, "The skyline operator", *Proc. ICDE*, 2001, pp. 421-430.
- [11] C. Y. Chan, H. V. Jagadish, K.-L. Tan, A. K. Tung, and Z. Zhang, "Finding k-dominant skylines in high dimensional space", *Proc. SIGMOD*, 2006, pp. 503-514.
- [12] J. Chomicki, P. Godfrey, J. Gryz, et al., "Skyline with Presorting", *Proc. ICDE*, 2003, pp. 816-825.
- [13] D. Kossmann, F. Ramsak, and S. Rost, "Shooting Stars in the Sky: an Online Algorithm for Skyline Queries," *Proc. VLDB*, 2002, pp. 275-286.
- [14] D. Papadias, Y. Tao, G. Fu, et al., "An Optimal and Progressive Algorithm for Skyline Queries," *Proc. SIGMOD*, 2003, pp. 467-478.
- [15] D. Papadias, Y. Tao, G. Fu, et al., "Progressive Skyline Computation in Database Systems," *TODS*, vol. 30, no.1, 2005, pp. 41-82.
- [16] W.-T. Balke, U. Guntzer, and X. Zheng, "Efficient Distributed Skylining for Web Information Systems," *Proc. EDBT*, 2004, pp. 573-574.
- [17] X. Lin, Y. Yuan, W. Wang, et al, "Stabbing the Sky: Efficient Skyline Computation Over Sliding Windows," *Proc. ICDE*, 2005, pp. 502-513.
- [18] X. Lin, Y. Yuan, Q. Zhang, et al., "Selecting Stars: The k-Most Representative Skyline Operator," *Proc. ICDE*, 2007, pp. 86-95.
- [19] Y. Tao, L. Ding, X. Lin, et al., "Distance-based Representative Skyline," *Proc. ICDE*, 2009, pp. 892-903.
- [20] C.-Y. Chan, H. V. Jagadish, K.-L. Tan, et al., "On High Dimensional Skylines," *Proc. EDBT*, 2006, pp. 478-495.