TransOS: a transparent computing-based operating system for the cloud

Yaoxue Zhang and Yuezhi Zhou*

Key Laboratory of Pervasive Computing,
Ministry of Education,
Tsinghua National Laboratory
for Information Science and Technology,
Department of Computer Science and Technology,
Tsinghua University,
Beijing 100084, China
E-mail: zyx@moe.edu.cn
E-mail: zhouyz@mail.tsinghua.edu.cn
*Corresponding author

Abstract: Cloud computing has become a hot topic recently. Among these research issues, cloud operating systems have attracted extensive attention. However, to date, there is no answer to such issues as what a cloud operating system is and how to develop one. This paper proposes a cloud operating system, TransOS, from the viewpoint of transparent computing, in which all traditional operating system codes and applications are centrally stored on network servers, and an almost bare terminal dynamically schedules the necessary codes selected by users from the network server, and runs them mostly with the terminal’s local resources. The TransOS manages all the resources to provide integrated services for users, including traditional operating systems. This paper first introduces the concept of transparent computing as a background and presents TransOS and its main characteristics. It then gives a layered structure-based designation of TransOS and finally illustrates one example of its implementation.

Keywords: transparent computing; cloud operating system; cloud computing.


Biographical notes: Yaoxue Zhang received his PhD in Computer Science in 1989 from Tohoku University, Japan. He is a member of the Chinese Academy of Engineering and the President of Central South University, China. Currently, he is also a Professor at the Department of Computer Science and Technology in Tsinghua University, China. His research interests include transparent computing, cloud computing, pervasive computing, and active services.

Yuezhi Zhou received his PhD in Computer Science in 2004 from Tsinghua University, China. Currently, he is an Associate Professor at the Department of Computer Science and Technology in Tsinghua University. His research interests include cloud computing, transparent computing, distributed systems, and computer networks.
1 Introduction

As a new computing paradigm first proposed by Google and other companies in 2008, cloud computing is an emerging hot topic in computing research and related industrial development (Armbrust et al., 2010). The defining characteristic of this new computing paradigm is that the computation and storage of software, originally done on a stand-alone computer, are now separated and completed on a client terminal and network server, respectively. That is, software [including operating system (OS) programmes, applications, and data] is stored on the network server, while the client terminal only manages user interaction and access to services. The computation that provides services for the user can be completed either on a network computing server, or at a client terminal.

Companies like Google, IBM and Amazon have already launched their own cloud computing technologies and products (Chen and Zheng, 2009). Google, in particular, has developed its Google file system (GFS) (Ghemawat et al., 2003), a storage structure for a large-scale distributed file system and MapReduce (Dean and Ghemawat, 2004), an algorithm for fast searching. However, so far there have been no reports of progress in the research and development of a network OS for a cloud computing system. It is true that Google launched its Chrome OS (Mediati, 2009) in 2009, but it is essentially an OS that provides web services for terminals and does not provide any effective management of networked software and hardware resources.

Focusing on user services and management of networked software and hardware resources, an OS for the cloud, TransOS, is proposed in this paper. TransOS treats traditional OSs, applications and files/data as resources, implements scheduling, distribution, recycling and control of such resources on/via the network, and manages storage and computation of such resources separately. Since TransOS runs in a distributed manner underlying various traditional OSs, users may choose an OS (Windows or Linux) and its respective applications on TransOS. Thus any OSs developed by different companies or based on different standards can be running on the same client terminal. This paradigm may reduce storage load and simplify resource management at the client so as to enhance the accessibility and security of user services.

In Section 2, we discuss the genetic concept and development of OSs and explain why we need a new cloud OS. Section 3 introduce the concept and core ideas of transparent computing as a background knowledge. We present the concept, main characteristics, and design of TransOS in Section 4 and gives one of its implementation example in Section 5. The related works are discussed in Section 6 and a conclusion of our study is made in Section 7.

2 Operating systems

An OS is a set of programmes, i.e., system software which stay between hardware and applications and manage computer hardware resources and provide common services for
TransOS: a transparent computing-based operating system

application software (Wikipedia, 2011). Its development can be roughly classified into two technological categories:

- **Single machine-based OS.** It is a single machine-based platform, which is built on the von Neumann architecture, for resource management and running programmes. It includes the batch processing system developed at the very beginning, as well as the multiprogramming system, time-sharing system, and OS with networking functions developed later. Some characterising representatives of this kind of OSs are Windows, Linux and Symbian, etc. Since the resource management on this kind of OSs is only involved in a stand-alone computer, with increasing demands from various applications, it will inevitably lead to the problems of increasing complex functions, heavy load of programmes and increasing security risks. The result is that they are less user-friendly and have difficulties in management and maintenance.

- **Network OS.** It adds protocols, networking functions, and shared resource management software to a single machine-based OS. This kind of OSs makes network as a platform for resources management and running programmes. Typical network OSs include Windows 7 (Nash, 2008), Android (Hall and Anderson, 2009) and Chrome OS (Mediati, 2009), etc. Compared with the single machine-based OSs, the front-end functions for the network OSs can be light-weighted, virtualised, mobile, and managed in a centralised manner. Therefore, they are more user-friendly in management and maintenance and have enhanced user accessibility and security.

It is fact that both kinds of OSs cannot run on across hardware and software platforms. For example, the OS supporting desktop software does not support mobile software, and the applications running on Windows cannot be running on Linux and Android. If we would like them to run on across hardware and software platforms, the system would become extremely complex and overloaded. This would entail some inconvenience and security risks for users to access various kinds of services and conduct management on the network and would also bring about high cost and power consumption. Therefore, they are not suitable for the cloud computing paradigm which involves management across sprawling, heterogeneous computation and storage environments and cannot provide active services as required by users.

Although Chrome OS tries to tackle the above problems, it is aimed at users who spend most of their time on the web and provides the only application, a browser incorporating a media player and a file manager (Wikipedia, 2011). It features a fast start-up speed as well as a quick access to the huge back-end database and computing services via the web browser, but cross-platform operations and access are not supported. Therefore, Chrome OS cannot be regarded as a final solution of cloud OS.

3 Transparent computing

Transparent computing is a computing paradigm that is aimed at providing active services for users (Zhang and Zhou, 2006). That is to say, users do not need to be aware
or well informed of the minute details of the technology applied in the system; instead, they need to only care about the services they want, and the quality of the services.

In order to achieve this goal, a transparent computing-based system consists of a network server and one or more client terminal. The client terminal is rather simple and light-weighted, almost like a bare computer. It only stores the underlying basic input output system (BIOS) and a small fraction of protocol and management programmes. On contrast, OSs, applications and user data are all regarded as software resources and stored in the network server. Moreover, computation and storage are spatio-temporally separated through the network. The OSs and applications required in the user services are dynamically scheduled from the network server to run on the client terminal through a distributed super OS in a buffer-enabled block or streaming way. Such a paradigm enables the client terminal to be diverse, light-weighted, secure, and easy to manage and maintain. Besides, the network server can be used to store various OSs and numerous applications, which can form a so called ‘cloud’ or link to a number of ‘clouds’ via the network. Therefore, users are able to freely access the services needed on networks across heterogeneous software and hardware platforms.

The core ideas of transparent computing include:

1. separation of computation, storage, and management
2. enabling across-terminal and across-OS operations and enabling users to select the services they need
3. instructions are exchanged in ‘data streams’ or ‘block scheduling’ between the network server and client terminal, and the super OS will conduct integrated management of all kinds of resources on the network
4. programmes are dynamically scheduled to run on specific terminals or servers in a streaming way (Zhou and Zhang, 2009).

Based on the theory of transparent computing, we proposed a transparent computing architecture 4VP++, and developed the client terminal and related systems accordingly in our previous study (Zhang and Zhou, 2007). A cooperated research project was established with Intel Corporation by exploiting and combining the new-generation BIOS, extensible firmware interface (EFI) (Unified EFI Forum, 2011) with transparent computing. It is expected to develop the next-generation of transparent computing systems (Tsinghua University, 2007).

4 TransOS, an OS for the cloud

4.1 The concept of TransOS

TransOS is a distributed system software, which stays between the server, network, and client terminal hardware (with BIOS) and the traditional OS and applications, manages all the networked resources and dynamically schedules the centrally stored codes and data of traditional OS and applications to run mostly with the local resources (including CPU, memory, and other devices) of the client terminal. The concept is shown in Figure 1. Generally speaking, the TransOS client runs on the client terminal devices and the TransOS server runs on the network server. The TransOS client lies just above the
TransOS: a transparent computing-based operating system

BIOS of the terminal, but beneath the virtual running traditional OS and applications. Here, the virtual running means that the traditional OS and applications are running in a virtual environment created by TransOS client and their codes and data are fetched from the network server in a streaming way. The dynamic scheduling and running of traditional OS and application is carried out with the cooperation between the TransOS server and client. The TransOS server manages the network servers to implement the central storage and management of traditional OS and data and respond to the requests issued by TransOS client.

Figure 1 Concept of TransOS (see online version for colours)

As mentioned above, the single machine-based OSs are only able to manage limited resources in the stand-alone computer, and the network OSs available up to now only add networking support to single machine-based OSs, so that they cannot provide active services to users as required. In the cloud computing environment, a super OS is needed that can manage networked software and hardware resources and provide active services to users. Therefore, based on transparent computing, we have designed a new layered OS, TransOS, for the cloud with distributed kernels. Unlike single machine-based OSs and existing network OSs, TransOS is able to conduct an integrated management of the networked resources consisted of various heterogeneous terminals, servers, network and traditional OSs and provide services as required by users freely and actively. The characteristics of TransOS are outlined as follows.

4.1.1 Traditional OSs as a kind of resources

Three main kinds of resources are under the management of TransOS. They are the hardware platform resource, the virtualised resource, and the software and data resource. The hardware platform resource consists of terminal(s), server(s) and the network, the virtualised resource is formed through device drivers and virtualisation based on the hardware resource, and the software/data resource includes traditional OSs, applications, statuses, and data. Therefore, different from traditional OSs which only manage and schedule applications and data, while in TransOS, traditional OSs, such as Linux and Windows, are regarded as software resource and subject to management and scheduling.
Y. Zhang and Y. Zhou

TransOS client is a core component of TransOS for the cloud. Once a preferred traditional OS is specified explicitly or implicitly by a user, the traditional OS will be scheduled by TransOS client to be executed on the client terminal via the TransOS server. One main task of TransOS is to perform the integrated management and scheduling of the above mentioned three kinds of resources, in which the TransOS client plays an important role in transparent computing-based TransOS for the cloud computing paradigm. For clarification purposes, some related terms are defined below:

- Networked resources: A networked resource is any physical or virtual device or component of limited availability within a networked computer system. In this context, networked resources refer to hardware resources to be managed by TransOS, including computation and storage devices such as terminal(s) and server(s) and network devices such as routers, switches and base stations, and software or data resources such as traditional OSs, development and management tools, applications, structured and non-structured data, etc.

- Autonomous network system: It is a collection of connected routing and network switches under the control of one or more network operators according to certain predefined protocols and policy. In this context, it refers to the network system under the control and management of TransOS which runs on connected hardware devices as terminal(s), server(s) and network switch.

- Instance OS: It means to create an instance of one of the traditional OSs such as Windows, Linux, Symbian, Android and Windows Mobile, etc. In this context, TransOS manages the traditional OSs and can create an instance OS upon the request from a client terminal.

4.1.2 Distributed kernels and buffer-enabled dynamic scheduling of instructions

The TransOS kernels are distributed on the client terminals and servers. As is shown in Figure 2, only a small fraction of the TransOS kernels reside in the client BIOS, and all other kernel modules reside in the server. It is necessary to point out that only the one-to-one client and server case was shown in Figure 2, however, it can be extended to any many-to-many cases. After the booting of client terminal, the TransOS kernels residing in the client BIOS will take over control of the client terminal, communicate with the TransOS server to fetch more kernel codes needed in running the system, and load them to the client for execution in a block-streaming way. Block-streaming is a way of scheduling and execution of a small fraction of codes or data each time on demand from the network server to the client terminal. That is to say, a small block of codes forms as a stream of codes flows from the network server to the client terminal and is executed. It is worthwhile to mention that the instance OSs, applications and data running on the TransOS are also dynamically scheduled to the client terminal in the block-streaming way.
Figure 2  Distributed kernels to be dynamically scheduled for execution (see online version for colours)

Figure 3  The relationship between TransOS, BIOS and traditional OS (see online version for colours)

4.1.3 Running above BIOS and beneath instance OSs

As shown in Figure 3, TransOS client may support various client hardware architectures, such as x86, ARM and MIPS, through standardised BIOS interfaces, such as EFI. When booted by the BIOS, TransOS client contacts the server and lists the available instance OSs for a user to select. Once selected by the user, the instance OS is scheduled by TransOS to the client terminal for execution in a block-streaming way.
4.1.4 Integrated management and scheduling of distributed, networked resources

As shown in Figure 4, TransOS manages an autonomous network system, which is an across-network, across-terminal and across-server platform. The management and maintenance of the networked software and hardware resources distributed in different locations is carried out via the network communication protocols. The resources including the client hardware, server hardware, networks, instance OSs, applications and data, etc., and all other related resources are scheduled upon the user’s specific needs, are automatically scheduled for the later use.

Figure 4 Integrated management of distributed network resources (see online version for colours)

4.2 The design of TransOS

In order to conduct an integrated management and scheduling of distributed networked software and hardware resources including instance OSs, the TransOS is designed as a layered modular structure as shown in Figure 5. It consists of seven layered modules and each module may be distributed on one or more devices as required but they jointly accomplish a function.

4.2.1 The networked interrupt handling module

It handles the interrupts from virtual devices (they are hardware devices of a single machine and can be used by other machines). This module may receive and handle networked interrupts from any one computer in the autonomous network system, and accomplish the functions requested by such interrupts. Figure 6 describes the difference between the interrupt handling of TransOS and that of a traditional single machine-based OS.
As seen in Figure 6, in a traditional single machine-based OS, an occurred interrupt, such as a page fault or a keyboard interrupt is handled locally. The CPU will save the current running state in the local storage and begin to run proper interrupt handlers. Once the interrupt is handled, the original running state will be restored and the running process is resumed. However, in TransOS, an occurred interrupt is not handled locally by the TransOS client. Instead, it is sent by a virtual device on the TransOS client terminal to the TransOS server via network and handled by the TransOS server. Once the interrupt is handled, the results will be returned to the TransOS client via networks, and then the CPU of the TransOS client will resume the running process.

4.2.2 Physical/virtual resource driver module

This module drives and manages the physical/virtual resources based on the networked interrupt handling. Especially, this module accomplishes the following three functions:

1. driving physical hardware devices, such as the network card and display card, and driving all hardware devices in the autonomous network system.
establishing, driving and managing virtual device resources, including virtual CPU, virtual memory and virtual storage, etc.

supporting the storage of large-scale software/data, which is similar to the functions of GFS and Bigtable (Chang et al., 2006) developed by Google Inc., accomplishing network storage and searching of large volumes of data.

4.2.3 The software/data resource storage module

This module is for organising, storing and managing the instance OSs, applications and data for effective access by the TransOS client and user side.

4.2.4 Resource management and scheduling module

The core function of TransOS involves management and scheduling of networked resources, including the scheduling, distributing and recycling of various network resources, which is accomplished by the resource management and scheduling module. Its function is similar to the kernel scheduling function of traditional single machine-based OSs. In order to meet the needs of scheduling and management requested by different networked resources, it is necessary to develop various scheduling, distributing and recycling algorithms based on different strategies and methods. Only if an elastic flow of resources are possible to form to meet heterogeneous requirements from different network resources.

4.2.5 Interfaces module

Similar to the single machine-based OS, TransOS also has its interface module, which can be used to interact with the applications and a user, such as receiving and handling requests sent by the applications or by the user.

4.2.6 Other modules

Different from the single machine-based OS, TransOS has two special modules, networked booting and optimisation module, and network communication and service protocol module. The client of the networked booting and optimisation module captures the startup interrupts at the client terminal and sends requests to the network server for the later handling of the interrupts. The server of the networked booting and optimisation module is for responding to the request sent from the client, assigning a network address to the client, performing user authentication, and returning the list of instance OSs supported by the autonomous network system to the user. Once the user selects an instance OS, it initiates relevant transmission protocols and loads some part of the instance OS on the memory of the client terminal to be ready for running, which is so called the booting of the instance OS. After the booting of the instance OS, the network communication and service protocol module transmits other necessary instance OS modules and application codes and data to the memory of the client terminal in a block streaming way. Different from the traditional OSs in which the communications and protocol module facilitates only communications
between applications, the communication protocols in TransOS transmit both data and codes as well, and they can start functioning even before the booting of the instance OS.

5 An implementation example of TransOS

There are many approaches to implement TransOS, and here is an implementation example based on EFI BIOS that is built on the 4VP+ (four virtual layers, two protocols) structure on common computer boards. With this implementation, the functions of the TransOS client and the TransOS server are accomplished on the four virtual layers and two protocols respectively, as shown in Figure 7.

The two protocols are multi-OS remote booting protocol (MRBP) and network service access protocol (NSAP). MRBP initialises the network environment and acquires booting interrupt signals; then it captures the instance OSs list stored in the server, downloads several OS kernel modules and the NSAP client. Therefore, the MRBP client codes must be resident on the BIOS of terminal mainboard beforehand. Since the MRBP may start its functioning before the booting of an instance OS, the MRBP client, unlike traditional protocols, shall have its own network device driver and network protocol stack instead of using the network communication protocol of the OS. NSAP transmits TransOS as well as the instructions, data, interrupts and responses of the instance OS.

The four virtual layers in 4VP+ are virtual network I/O management (VNIOM), virtual network device management (VNDM), virtual network file management (VNFM) and virtual network user management (VNUM).

VNIOM. The major functions in this layer include receiving the I/O requests from the instance OS, analysing the interrupts and activating proper handling programmes to handle them. Different from the single machine-based computer environment, the VNIOM in the client does not handle the I/O requests but it sends them to the server and then receives the handling results from the server. Since the server needs to handle the I/O requests from different client terminals,
the VNIOM in the server needs to manage multiple priority-based I/O queues and respond accordingly.

2 VNDM. The functions in this layer include allocating and reclaiming of virtual devices, maintaining and managing the mapping from the client virtual devices to the virtual image on the server, and completing the virtual exchanges and scheduling between client and server.

3 VNFM. This layer performs the storage, access and searching of large-scale distributed file system, which is similar to those functions done by Google GFS. It also provides some auxiliary functions such as allocation and management of user file space and file directories, user file access redirection and virtualisation, ensuring the consistency of file access, and provision of access control and searching of user files.

4 VNUM. Functions in this layer include adding and deleting users, and registering and configuring their information, conducting user authentication, and supervising user’s operations.

We have already implemented 4VP+ in LAN environments based on desktop terminals and PC server clusters, and have promoted its industrialisation and actual deployment. We are currently conducting the research of TransOS based on tablet terminals and mobile phones as well as mobile internet. Besides, TransOS can also be applied to home appliances and other devices.

6 Related work

In recent years, there have been a lot of efforts on promoting cloud computing. Google Inc. has developed GFS (Ghemawat et al., 2003), BigTable (Chang et al., 2006) and MapReduce (Dean and Ghemawat, 2004); Amazon has developed EC2 (Amazon, 2011a) and S3 (Amazon, 2011b), and other companies and research institutions have developed their own cloud computing technologies and products (Chappell, 2011; Borthakur, 2011). However, all these are developed on the distributed system based on traditional single machine-based OS, and up to now little attention has been paid to the cloud OS so far. Recently, there are many emerging technologies related to cloud storage and cloud virtualisation (Calder et al., 2011; Mahajan et al., 2011; Zhang et al., 2011), these works focus on the improvement of current cloud technologies, but not concerning the cloud OS issues, such as resource sharing, scheduling, and management of a cloud system.

Table 1 gives detailed comparisons between TransOS and Chrome OS. Chrome OS (Mediati, 2009), as an example of cloud OS, is developed and running on the network platform, and provides services for users with the light-weighted front-end interface and the powerful back-end ‘cloud’. In a word, it focuses on a light-weighted front-end and powerful back-end support. TransOS adopts a similar idea and strategy. However, Chrome OS adds a browser to the local light-weighted OS kernel, and downloads the web applications via the local browser, which is still a development approach based on a traditional single machine-based OS. Different from Chrome OS, TransOS tries to use the open BIOS technology and a light-weighted firmware module, and schedules the needed instance OSs and applications stored on the network to run locally in a block streaming way from the network server to the client terminal. It can be
seen that TransOS adopts a development approach based on a network OS. Moreover, Chrome OS only supports web applications, whereas TransOS may support multiple OSs and heterogeneous applications.

Table 1  Similarity and difference between TransOS and Chrome OS

<table>
<thead>
<tr>
<th></th>
<th>TransOS</th>
<th>Chrome OS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware platform</td>
<td>x86, ARM, MIPS, can be used on PC, server, netbook, tablet computer, and mobile phones</td>
<td>x86 and ARM, can be used on netbook only</td>
</tr>
<tr>
<td>Network support</td>
<td>LAN, WAN, 3G, etc.</td>
<td>WAN</td>
</tr>
<tr>
<td>Type</td>
<td>A light-weighted operating system, with distributed kernels/components + web browser</td>
<td>A light-weighted operating system, with operating system kernels + web browser</td>
</tr>
<tr>
<td>What to be transmitted through network</td>
<td>The operating system environment (including the operating system, applications, data, status, etc.)</td>
<td>Applications</td>
</tr>
<tr>
<td>Storage location of the OS</td>
<td>Network</td>
<td>Local</td>
</tr>
<tr>
<td>Storage location of applications</td>
<td>Network</td>
<td>Network</td>
</tr>
<tr>
<td>Types of applications supported</td>
<td>Local and web applications</td>
<td>Web applications</td>
</tr>
<tr>
<td>Maintenance and management of the OS</td>
<td>Updated remotely and conducting integrated management</td>
<td>Read-only locally</td>
</tr>
<tr>
<td>Maintenance and management of the applications</td>
<td>Centralised management</td>
<td>Centralised management</td>
</tr>
<tr>
<td>Objects of management Programming</td>
<td>Operating systems</td>
<td>Applications</td>
</tr>
<tr>
<td>Backward compatibility</td>
<td>Supporting both traditional applications and web applications</td>
<td>Only support web applications</td>
</tr>
<tr>
<td>User experience</td>
<td>Maintaining the original user experience</td>
<td>Enhancing user experience</td>
</tr>
</tbody>
</table>

7  Conclusions and future development

Cloud OS is one of the core issues in the field of cloud computing technology. The OSs that have been developed so far, however, can only manage single machine-based software and hardware resources. In the cloud computing paradigm, which involves diffuse and heterogeneous computing devices, users want to run any service needed on their handy devices without concerning themselves with the underlying technologies. Trying to tackle this problem, this paper has proposed a cloud OS, TransOS, based on the concept of transparent computing. The main idea is to separate the computation, storage, and management of a traditional OS, applications, and services. To satisfy
this goal, all the codes of a traditional OS and its applications are centrally stored and managed on a network server. Users can select any OS and application they want to run. The codes of the selected OS and applications are dynamically scheduled on demand by streaming from the network server and run mostly with the client terminal’s local resources. This lightweight way can reduce the storage load and simplify client terminal management and maintenance and allow users to freely select and run whatever resources they desire. The TransOS manages all the networked and virtualised hardware and software resources, including traditional OS, physical and virtualised underlying hardware resources, and enables users can select and run any service on demand. The kernels of TransOS are distributed across the autonomous network system, enabling integrated management of various networked software and hardware resources including traditional OSs, and can schedule resources on demand from a user. With the support of super OS like TransOS, users, a lightweight terminal, can receive powerful back-end support from the cloud and various other active services. We compared TransOS with Chrome OS in several ways, showing that TransOS is a more efficient, backward-compatible, and user-friendly solution for cloud operating systems.

Future development in this field may include

1. exploring how to extend the applications of TransOS to even more devices (such as digital home appliances) and networks (such as 3G and LTE networks)
2. establishing a cloud operating system architecture and relevant interface standards
3. enhancing relevant theories and technologies based on actual implementation and application.

Acknowledgements

This work is partially supported by the National Core-High-Base Major Project of China (Grant No. 2010ZX01045-001-005-4), the National High Technology Research and Development Programme of China (Grant No. 2011AA01A203), and the research fund of Tsinghua – Tencent Joint Laboratory for Internet Innovation Technology. The authors would like to acknowledge the helpful comments of anonymous reviewers, guest editors and the participants of the transparent computing project.

References


