SCIENCE CHINA
Information Sciences

• PERSPECTIVE •

December 2017, Vol. 60 126101:1–126101:3
doi: 10.1007/s11432-017-9240-4

Understanding "software-defined" from an OS perspective: technical challenges and research issues

Hong MEI[1,2]

[1]*Beijing Institute of Technology, Beijing* 100081*, China;*
[2]*Key Laboratory on High-Confidence Software Technologies (MOE), Peking University, Beijing* 100871*, China*

Received August 29, 2017; accepted September 13, 2017; published online November 8, 2017

Citation Mei H. Understanding "software-defined" from an OS perspective: technical challenges and research issues. Sci China Inf Sci, 2017, 60(12): 126101, doi: 10.1007/s11432-017-9240-4

What is "software-defined"? Software-defined has become one of the hottest buzzwords in the information technology (IT) community. It is used to describe a family of technologies, including software-defined networking (SDN), software-defined storage (SDS), and software-defined data centers (SDDC). These are part of a broader trend that is referred to as software-defined everything (SDX). The movement toward software-defined infrastructure is focused on decoupling hardware layers that execute data transactions and computations from the software layers that manage them.

The term "software-defined" originated from software-defined networking (SDN), which became popular after the release of OpenFlow [1]. As one of the most important research outcomes of POMI 2020, an NSF Expedition project led by Stanford University that began in 2009, OpenFlow has been widely adopted in the cloud and in data centers. Although the term SDN has been frequently linked to OpenFlow, the technical essence of SDN can be traced back to as early as the 1990s, where early software-defined prototypes, including AT&T's GeoPlex, emerged.

The core technologies of software-defined can be summarized by two concepts [2]. The first is the virtualization of hardware resources, meaning that all hardware resources can be abstracted as virtual resources, which can be managed by OS routines or the control plane. The second core technology is the programmability of management functionalities, meaning that users can write programs to access the services provided by virtualized resources. The functionality of virtualized hardware resources should be programmable, such that it can be managed by software. From the perspective of programming, this means that one can use software to define the functionality of a system, thereby allowing it to be called a "software-defined system".

Software is far more flexible than hardware and can be changed much more easily. Although firmware can also be updated in legacy devices, it is still much less flexible than today's software-defined environments. With SDN, SDS, and software-defined servers, we can construct large software-defined environments, such as SDDC, which can be controlled, managed, and maintained much more flexibly. The technical and economic benefits brought about by using software-defined technology have been overwhelming. Software-defined is not only changing technology and products, but also changing the entire technological infrastructure and ecosystem.

Understanding software-defined from an OS perspective. Although software-defined has been promoted as a new technological breakthrough since SDN was adopted, we argue that the concept of "software-defined" has actually existed for many years prior to the advent of SDN. Consider a

Email: meih@pku.edu.cn

personal computer (PC) as an example. Although most PCs use standard hardware components (CPU, memory, motherboard), each PC exhibits unique behaviors in the hands of each user because operating systems (OSs), such as Windows and Linux, enable the installation of various personalized software and applications.

In retrospect, we can argue that an OS actually offers software-defined capabilities for a PC, making one computer system different from another, even if they have the same hardware. The earliest computers did not include any operating systems, which limited their capabilities to certain types of tasks, which were mostly scientific computation problems [3]. Only after mainstream operating systems, such as UNIX and Windows, emerged, did computers became powerful enough to expand into business and personal applications, eventually touching every corner of our world and every aspect of our lives.

As technology and time have advanced, most computer-related products have seen a decreasing number of hardware options, with major features and functions that are enabled through software comprising the main differences. In this sense, OSs were the earliest technology to embrace the software-defined concept. The main functionality of an OS is to abstract hardware resources (CPU, memory, storage, networking) and provide flexible services to the applications running on it. From this perspective, every modern computer running a modern OS, such as Windows or Linux, can be called a "software-defined computer".

Software-defined represents a new class of products where software is the focus and software, rather than hardware, is used to provide solutions. From an OS perspective, we can view all software-defined products as providing an OS-like software layer to manage their corresponding hardware. For example, SDN actually provides an OS for networking hardware, meaning all networking functionality can be virtualized and managed or programmed using software. Creating an SDDC entails constructing an operating system for a data center. However, the operating system for a data center runs above existing operating systems (such as Linux) because it manages all the servers within the data center as a whole, rather than individually managing each hardware component within each server.

Ultimately, SDX is focused on using software to bridge the technological and organizational gaps between discrete IT silos to create an underlying infrastructure that can be managed using software abstractions, such as operating systems.

Technical challenges and research issues. Although SDX, especially SDN, has been widely accepted in the technology community and industry, it still faces many difficult challenges.

(1) The architectural design of SDX. Although SDX has been widely accepted in many areas, it has only been in development for a relatively short period of time. As a result, many aspects of its architectural design have not been studied extensively. One of the key design considerations in SDX architectures is determining the granularity of software-defined components in order to balance the trade-off between granularity and efficiency. Virtualization is another important challenge because new systems may require completely new methods of implementing virtualization, which is one of the key enablers of SDX.

(2) Quality improvement of SDX. Similar to many other systems, SDX faces the typical challenges in performance, energy consumption, and dependability. How do we ensure that a software-defined system has comparable performance to the original system? How do we guarantee that a software-defined system does not consume more energy compared to the original system? Is there a way to implement a systematic process to ensure that a software-defined system is dependable? These issues must be investigated in order to guarantee the efficient construction and deployment of software-defined systems.

(3) Security of SDX. Software systems are more vulnerable to security threats compared to hardware systems. With SDX, software becomes the control center of a system or environment, making it the main target of attackers. However, few efforts have been invested into making SDX more secure. In the future, new security mechanisms and security protocols should be embedded into SDX technology, such as SDN. Security has become an ever-present challenge to all information systems that are supported by SDX.

(4) Lightweight virtualization. Virtualization is one of the key enablers of SDX. While cloud computing has been pushed to the edge, such as in smartphones and IoT devices, we must investigate lightweight virtualization technologies in order to provide efficient software-defined capabilities and support software-defined edge computing.

(5) Smooth transition to SDX. Although SDX has already been adopted in many areas of industry, many enterprises are still hesitant to make the transition. One of the key obstacles is that the transition to SDX requires a complete overhaul of current infrastructure, necessitating the installation of new hardware and software management systems. Therefore, it is very important to help organizations to make the transition to SDX in a

smooth and economically sound manner.

(6) Generalization of SDX. Since SDN became popular, many software-defined systems, such as software-defined storage and software-defined cloud computing, have been proposed. However, current types of SDX have been largely limited to hardware, such as traditional IT infrastructure, including networking or data centers. We believe that software-defined may be generalized in many different ways.

- Although SDX has largely been applied to hardware resources, such as networking or storage devices, it may also be applied to higher-level software resources, such as platforms, data, and applications. In the future, we may see software-defined being applied to full resource stacks, including hardware and software, to aid in connecting various IT silos and supporting new software-defined applications. In addition to traditional hardware and software, higher-level concepts, such as data, can also be software-defined. For example, it is possible to offer software-defined data opening and sharing capabilities between many legacy systems without source code or documentation, which will facilitate the emergence of a new generation of IT systems in various areas, including government, enterprises, and corporations.
- SDX can also be generalized to areas other than computers or data centers. For example, consider a software-defined enterprise or software-defined city. To this end, research and development work has already been proposed in Urban OS (software-defined city)[1], HomeOS (software-defined home) [4], CampusOS (software-defined campus) [5], and Internetware OS (software-defined Internet) [6, 7]. As one of the major driving forces behind "Made in China 2025", it is inevitable that software-defined will be applied to the modernization of manufacturing businesses (i.e., software-defined manufacturers). Additional research efforts are needed to investigate how these new types of SDX can be created and managed efficiently, and how to put them into practice.

Acknowledgements This work was supported in part by National Natural Science Foundation of China (Grant No. 61421091). The author wants to thank Professor Yao Guo and anonymous reviewers for insightful comments and suggestions that help improve this work.

References

1 McKeown N, Anderson T, Balakrishnan H, et al. OpenFlow: enabling innovation in campus networks. SIGCOMM Comput Commun Rev, 2008, 38: 69–74
2 Mei H, Huang G, Cao D G, et al. Understanding software-defined from the perspectives of software researchers (in Chinese). Commun CCF, 2015, 11: 68–72
3 Mei H, Guo Y. Network-oriented operating systems: status and challenges (in Chinese). Sci Sin Inform, 2013, 43: 303–321
4 Dixon C, Mahajan R, Agarwal S, et al. An operating system for the home. In: Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation (NSDI), San Jose, 2012
5 Yuan P F, Guo Y, Chen X Q. Towards an operating system for the campus. In: Proceedings of the 5th Asia-Pacific Symposium on Internetware, Changsha, 2013
6 Mei H, Guo Y. Development and present situation of Internetware operating systems (in Chinese). Sci Technol Rev, 2016, 34: 33–41
7 Mei H, Huang G, Lan L, et al. A software architecture centric self-adaptation approach for Internetware. Sci China Ser F-Inf Sci, 2008, 51: 722–742

1) Living PlanIT SA. http://living-planit.com/.