Services Computing for Big Data: Challenges and Opportunities

Gang Huang(✉)

Key Lab of High-Confidence Software Technologies, MoE, China, Peking University, No.5, Yiheyuan Road, Haidian District, Beijing 100871, China
hg@pku.edu.cn

Abstract. In the era of Big Data, data becomes a type of resource like the material and energy. However, a huge volume of deep data resides in the billions of information islands whose data cannot be open to the third parties easily and naturally. In this keynote, we discuss why and how Services Computing can be the silver bullet to the grand challenge of opening and sharing such deep data.

Keywords: Big data · Services computing · Client-Cloud Convergence

1 Information Island Crisis in the Era of Big Data

In the era of big data, data becomes a type of resource like the material and energy. There are many types and sources of data while the main body is in the billions of IT systems connected by the Internet. The well known data in the Internet is the surface data from the World Wide Web, which is information centric, and can be retrieved by standard web crawlers or search engines such as Google, Baidu, Bing, etc. Till June 2016, it is reported that there have been 4.5+ million web sites with 200+ billion web pages. However, such surface data accounts for very small portion of all data that resides on the Web, because so much data can be fetched only in a "service-oriented" way. Such data is stored deeply in enterprise information systems, desktop applications, mobile apps and embedded systems. These systems use some private technologies and do not follow the standard web technologies. As a result, they cannot be accessed via crawlers.

Indeed, deep data is the core competitive advantage of big data. However, collecting deep data is quite challenging because of the information islands, which cannot open their internal data, functions and workflows in an intuitive and easy way.

Opening the information islands raises various challenging issues. Typically, a Web system consists of three tiers, back-end DB, application logic that can be hosted by middleware, and the front-end applications. Considering the type of front-end, there can be three types, i.e., Client/Server, Browser/Server, and App/Server. To collect the deep data and enable the connection between information islands, various specific or ad-hoc solutions for different levels and scenarios have been proposed. Typical technologies include DB exporter/importer,

S. Wang and A. Zhou (Eds.): CollaborateCom 2016, LNICST 201, pp. 690–696, 2017.
DOI: 10.1007/978-3-319-59288-6_69

crawler, refactoring, package interception on network, and so on. However, these solutions are quite ad-hoc, and heavily depend on the application infrastructure, e.g., hardware, OS, security policies. As a result, they are of high difficulty, risk, and cost, but error-prone and difficult-to-generalize.

2 Services Computing: The Silver Bullet for Opening Information Island

Given the urgent requirements of collecting deep data, it calls for promising and feasible solutions to address the information island crisis. Essentially, connecting the information islands indicates the deep and on-demand resource sharing. In the US NSF "Grand-Challenge" Project (2008–2012), tens of professors from Stanford, UC Berkeley, and so on, proposed the programmable open mobile Internet (POMI). McKeown *et al.* proposed the concept of "*Software-Defined Network*" along with software APIs to decompose the management of network hardware and connect isolated hardware and share network resources [1], and finally promotes the emergence and popularity of SDN as well as its related huge industry. Such a success exactly brings "silver bullet" to open the hardware islands of computer networks.

However, compared to the hardware, opening the software islands is more complex. Google In-App search can search some app contents only when those apps implement some pre-defined interfaces, but such manually re-engineered approach is infeasible to the millions of apps in Google Play today. Besides the pre-defined interfaces approach, Apple iOS spotlight can search the data preloaded by apps but such cached data is always out-of-time and partial if the apps are seldom opened by the user. Still in the POMI project, Klemmer *et al.* [2] proposed "*Design Pattern Mining*" to open and share dynamic Web data. More specifically, they create APIs and make record-and-replay to recover the structure as well as the associated data.

Indeed, these preceding work has made the preliminary efforts to open information islands. Essentially, we can infer that such an **API-based** solution can be a promising way, **which is the core of services computing**. Services computing is one of the most important computing paradigms for the Internet computing environment. From software system perspective, services computing aims to build software application systems by connecting various independently developed services, or more specifically, APIs, and thus enables the dynamic and flexible adaptation to the emergent requirements.

Indeed, services computing faces a long-term debates of the unsuccessful and unexpected adoption of Web services technologies such as SOAP, WSDL, and UDDI. However, services computing itself is not limited to the underlying technologies and platforms, but provides a programing paradigm to establish the ubiquitous and pervasive fabric and realize the interoperability between isolated "building blocks" [3].

3 Our Research Practice: Client-Cloud Convergence

We position services computing as a "Silver Bullet" to open the information islands. In the past ten years, our research group in Peking University, has made a systematic solution in this direction. Due to page limit, we then summarize the efforts as follows.

3.1 Deep Control of Web Pages

The first effort of our silver bullet is to control the web pages for opening information islands. To this end, we choose the services mashups as the target application. Services mashups essentially integrate data from multiple sources such as web pages and web services. To enable services mashup, the first research question is that very few services mashup components can be found on the Internet. In our opinion, the web pages can provide a large amount of "deep" data, but without APIs.

To open the web pages, we try to encapsulate any web page as a web mashup component which can expose any data from a web page. We propose the service component model that can parse a web page and extract any data from the page [4]. By our service component model, we can open any data from a web page and release it as an API (service interface actually). Based on such a model, we further implement a browser-based runtime to enable the flexible interaction between every single "componentized" web pages.

3.2 Deep Control of Web Browser

The second part of our silver bullet is that we control the web browser for opening information islands. Web browser plays as the runtime infrastructure of web pages, and thus controls the behavior of web pages such as requesting data from servers, parsing the data, and rendering them on visualized UI. To further support the web pages that do not well comply with our service component model, we choose to control the web browser, which is the runtime of web pages, and enable data extraction. We make an in-depth study of the popular Chrome web browser (10 + Million LoC) and perform model checking of its runtime behavior for understanding the whole browser-based services mashups. More specifically, we redefine the security mechanisms of standard web, i.e., sand-box and single-origin, so that they can interact with each other. It should be noted that, such a solution essentially plays as the similar role of HTML5 *postmessage()*, but our idea was drawn three years earlier.

3.3 Deep Control of Browser/Server Interaction

We then control the interaction between web browser and web server for opening information islands. In practice, it is well known that loading web pages is quite slow, and the situation becomes even worse for services mashups. We then focus

on whether standard Browser/Server interactions unfit services mashup. To this end, we find the imperfectness of current web cache, and redefine the cache mechanism with fine-grained service interfaces as well as the underlying browser facility, so that developers can easily control the cache strategies of HTTP and HTML. In this way, the cache performance can be improved with 58% [5].

Additionally, rich services mashups not only suffer from slow page load, but further cannot work well on mobile devices such as smartphones. For example, Chess games, 3D Graphics, RPGs related mashups have severely poor user experiences. Hence, we control and save web-page-based services mashups with less computing resources and energy. To this end, we propose the first approach to leveraging the cloud-side resources and making the H5 programs offloading from mobile browser to cloud. We use very advanced techniques to analyze the constraints of JavaScript programs, partition the code, and offload those computation-intensive code to cloud. With our approach, we gain 49x page load time improvement and 92% energy saving [6].

3.4 Client-Cloud Convergence

These preceding efforts, indeed, provide the preliminary foundations to enable the opening and sharing data from isolated information islands. To further synthesize these results, we propose a vision, namely Client-Cloud-Convergence [7], which means any part of data, computation, control flow, and even user interaction can be on demand distributed, executed and collaborated in any devices and clouds.

We extend the classical Model-View-Controller programming model with a new Service element so that the Browser/Server applications, the Client/Server applications and the mobile App/Server applications can be smoothly transformed to service oriented applications. We name such a systematic support as SM@RT, which is not limited to web systems. For example, we re-implement the SM@RT approach in Java systems. This model helps to automatically analyze the Java bytecode and offload the computation-intensive code to the cloud, re-direct the Java invocation, and finally reduce the execution time and battery energy consumption of the Android apps drastically [8].

3.5 Summary

In a short summary, our 10 years research makes substantial breakthrough of opening the deep data of all C/S, B/S, and A/S in current networked software architecture, as illustrated in Fig. 1. The core idea is to automatically analyze the executable code (rather than source code) and runtime data of legacy systems, and then automatically generate a Service-Model-View-Controller (SMVC) model instance which is causally connected with the running legacy system. The causal connection means any runtime change of the legacy system will immediately lead to the corresponding change in the SMVC model, and vice versa. The SMVC model instance exposes a set of Application Programming Interfaces to

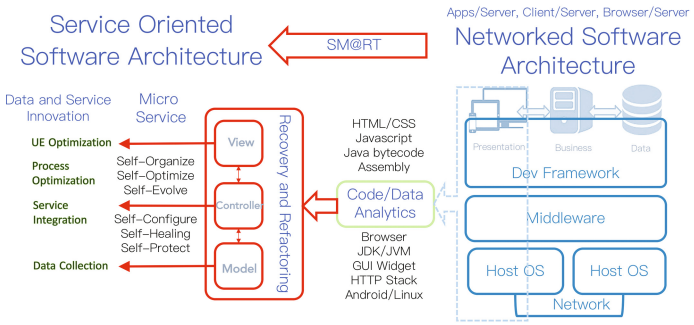

Fig. 1. Technical architecture for Client-Cloud Convergence

the third parties, so that the data, function and workflow of the legacy systems can be read and written in an open and real-time manner.

4 Our Industrial Practice: The YanCloud System

Beyond the academia results, we also develop our commercial system, called YanCloud for Data-as-a-Service, as illustrated in Fig. 2, and gain various successful applications in real-world industrial practices.

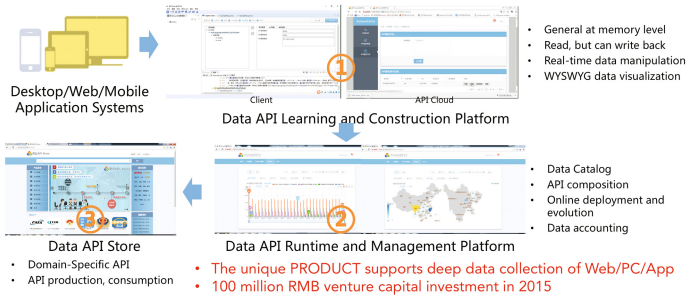

Fig. 2. YanCloud for data as a service

More specifically, the YanCloud consists of three major parts. The first is the Data API Learning and Construction Platform, which can generate data access interfaces of the given Desktop/Web/Mobile Application Systems. The second is Data API Runtime and Management Platform, which hosts and executes the SMVC model instance. The third is Data API Store, which is similar with mobile App Store but the App is "de-assembled" to APIs. With YanCloud DaaS, one can generate, deploy, operate, manage, buy, or sell APIs of any legacy systems.

In the past one year, YanCloud DaaS has been successfully applied in more than 500 systems covering e-Government, Smart City, education, finance, power-grid in 23 provinces across China. Compared to the traditional data collection solutions of our partners, including Digital China and other Top 100 software companies in China, YanCloud improves the engineering efficiency by 100 times while saves the labor cost by 90% on average. Since more and more deep data can be collected by YanCloud, we build up an API store for sharing and crowdsourcing of data, algorithms, applications and stakeholders, so that the entrepreneur based on big data becomes possible. In particular, we recently focus on a killer application of YanCloud DaaS, enabling emergent cooperation among mobile APPs. Recall that, YanCloud can open the data, functions and workflows of mobile apps. As a result, we can search and use the data and functions internal of mobile apps. We can also sense the user behavior of a mobile app and then recommend the data and functions of the other apps. In other words, the data and functions of a mobile app can cooperate with those of other mobile apps in an emergent fashion.

5 Future Outlook

Compared to the Web, the wide adoption of mobile apps have created much larger opportunities and challenges for opening the isolated information islands. The inter-organization deep-data sharing in Demand-Side Platform allows the data related to the advertisement to bid among mobile apps. Much more data can be shared through APIs in a situational way, which brings the phenomenon of API economy. For example, everyday there are billions of API requests of Google and Facebook, rather than their mobile apps. Obviously, compared to the manner of sharing the data as a static data set, sharing the data via APIs will be the best-of-the-breed. We believe the API economy will be one of the most important business models of big data.

However, in the context of API-economy and big data, services computing plays as fundamental "fabric". The era has changed, but some essences stay. We still need services description, discovery, invocation, and composition, but calls for entirely new technologies. We shall focus more on API economy by services computing, e.g., API generation and specification, API management, API operation, and API composition for situational requirements. In that sense, services computing will play a very important role in API economy and has unique values to the era of big data.

Acknowledgments. This work was supported by the High-Tech Research and Development Program of China under Grant No.2015AA01A203, and National Natural Science Foundation of China under Grant No.61421091, 61370020, 61528201. The author would like to thank colleagues from the System Research Lab in the Institute of Software, Peking University, and the supports from "Peking University-Digit China" Collaborative Innovation Center.

References

1. Martn, C., Michael, J.F., Justin, P., Jianying, L., Natasha, G., Nick, M., Scott, S.: Rethinking enterprise network control. IEEE/ACM Trans. Netw. **17**(4), 1270–1283 (2009)
2. Ranjitha K., Arvind S., Csar T., Maxine L., Salman A., Scott R. K., Jerry O.T.: Webzeitgeist: Design Mining the Web. In: 2013 ACM SIGCHI Conference on Human Factors in Computing Systems, pp. 3083–3092. ACM, New York (2013)
3. Qiao, X.Q., Chen, J.L., Tan, W., Schahram, D.: Service provisioning in content-centric networking: challenges, opportunities, and promising directions. IEEE Internet Comput. **20**(2), 26–33 (2016)
4. Liu, X.Z., Huang, G., Zhao, Q., Mei, H., Blake, M.B.: iMashup: a Mashup-based framework for service composition. Sci. China Inf. Sci. **57**(1), 1–20 (2014)
5. Liu, X.Z., Ma, Y., Liu, Y.X., Xie, T., Huang, G.: Demystifying the imperfect client-side cache performance of mobile web browsing. IEEE Trans. Mob. Comput. **15**(9), 2206–2220 (2016)
6. Wang, X.D., Liu, X.Z., Zhang, Y., Huang, G.: Migration and execution of JavaScript applications between mobile devices and cloud. In: Conference on Systems. Programming, and Applications: Software for Humanity, pp. 83–84. ACM, New York (2012)
7. Huang, G., Liu, X.Z., Ma, Y., Lu, X., Zhang, Y., Xiong, Y.F.: Programming situational mobile web applications with cloud-mobile convergence: an internetware-oriented approach. IEEE Trans. Serv. Comput. (to appear)
8. Zhang, Y., Huang, G., Liu, X.Z., Zhang, W., Mei, H., Yang, S.X.: Refactoring android Java code for on-demand computation offloading. In: 27th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications, pp. 233–248. ACM, New York (2012)