Cloud Client Software VINCA ProUsage Post-print

Authors: Wang Jing, Weilong Ding, Zhao Shuan, Li Feng

Date: 2017-03-10T00:00:00+00:00

Abstract

With the establishment of cloud infrastructure based on open network environments and the development of the XaaS service delivery model, end users can consume and utilize resources in cloud environments in a "use without owning" , "on-demand" , and "ubiquitous" manner. However, while enjoying the benefits brought by centralized economies of scale, the demand for user autonomous control has become increasingly prominent. How to enable users to effectively utilize cloud infrastructure while also possessing autonomous control capabilities such as personalized application customization, diversified terminal utilization, and private data protection has become one of the key issues in cloud client research. This paper first provides an overview of related work on cloud clients, and then introduces the architecture and principles of VINCA ProUsage —a novel cloud client software developed by the authors.

Full Text

Preamble

VINCA ProUsage: A Cloud Client Software
Jing Wang, Weilong Ding, Shuan Zhao, Feng Li

Abstract

With the establishment of cloud infrastructure based on open network environments and the development of XaaS (Everything as a Service) delivery models, end users can now consume and utilize resources in cloud environments in an "on-demand," "ubiquitous," and "use-without-owning" manner. However, while enjoying the benefits of centralized economies of scale, users' demand for autonomous control has become increasingly prominent. How to enable users to effectively leverage cloud infrastructure while retaining autonomous control capabilities such as personalized application customization, diversified terminal

utilization, and private data protection has become a key research issue in cloud clients. This paper first provides an overview of related work on cloud clients, then introduces VINCA ProUsage—a new type of cloud client software we have developed—describing its architecture and key principles.

Keywords: cloud client, user-driven, end-user programming, local resource integration

1. Introduction

Driven by the need for cross-domain information sharing and application collaboration, content-centric cloud "resource centers"have gained increasing attention. By establishing Internet-based information processing infrastructure, a logical center for resources—including computing, storage, data, and software—can be provided for upper-layer applications. This approach openly incorporates network resources, optimizes their management and scheduling, and delivers services through the XaaS model. Under this paradigm, users consume and utilize cloud resources in a "use-without-owning," "on-demand," and "ubiquitous" fashion [1]. Users no longer need to build their own IT infrastructure or deploy software; instead, professional operators handle operations and maintenance, while users access resources on-demand through a "pay-as-you-go" model. Compared with traditional models, this significantly reduces users' upfront capital investment and risk, lowers management costs, and enhances business flexibility. From the operator' s perspective, this centralized operations and maintenance model effectively leverages economies of scale to reduce average costs. Cloud computing is a typical product of scale economy drivers. Operators achieve cost and energy consumption reductions through centralized resource management and optimized scheduling, maximizing resource sharing and improving overall utilization. Clearly, this centralized management approach creates a win-win situation for both users and operators.

However, alongside the benefits of centralized economies of scale, users' demand for autonomous control has become increasingly prominent, manifesting in three main aspects:

Personalized User Requirements: The openness and massive, diverse user base of cloud computing environments pose challenges for application construction. Traditional models relying on IT professionals to build systems based on complete specifications can no longer satisfy the spontaneous, dynamic, and immediate application needs of large Internet user populations. The application construction paradigm is gradually shifting from IT personnel analyzing requirements to end-user-centered, on-demand dynamic application construction. Users need to be able to construct personalized applications that meet their immediate needs through client-side programming means such as on-demand customization or configuration.

Diversified Terminals: Terminals serve as the gateway for users to access cloud computing environments. With advances in hardware technology and mo-

bile Internet, terminals are no longer limited to traditional PCs but now include mobile phones, tablets, smart TVs, and other device types. These rich terminal devices support users in accessing cloud resources anytime and anywhere, unrestricted by location. User environments have also evolved beyond traditional web browsers to include "rich client" software that can comprehensively utilize both cloud and local resources, combining efficient, user-friendly local interaction with the powerful processing capabilities of cloud environments to deliver a better user experience.

Private Data Protection: While users want to leverage cloud services to reduce the costs and risks of purchasing, developing, and maintaining their own infrastructure, they also wish—due to policy, legal, and intellectual property considerations—to retain certain core data and business processing on the client side. This allows them to maintain autonomous control over core operations and avoid the risk of private resource leakage that could result from transmitting private data over the network.

In response to these requirements, this paper first outlines the current state of research, then introduces VINCA ProUsage—a new type of cloud client software we have developed—presenting its principles and key technologies.

2. Overview of Cloud Client Research

Building upon the three issues discussed above, this section provides an overview of related research from three perspectives: end-user programming, integrated terminal utilization, and private data protection.

2.1 End-User Programming

End-user programming provides non-professional software developers with techniques to create or modify software products to a certain extent [2,3]. With the development of service computing, the application domain of end-user programming has evolved from early desktop software to cloud-based applications built through service composition in cloud environments. Services, as domain-oriented reusable components, can more directly reflect user business requirements compared to objects or components, and their implementation technologies are cross-platform and cross-programming-language. Therefore, service-based reusable components are easier to integrate. End-user-driven service composition supports end users in autonomously assembling reusable service components within their domain on-demand to obtain software entities that satisfy their personalized needs. In service environments, on one hand, users can perform composition programming on a set of larger-granularity components that are closer to their needs, reducing programming complexity; on the other hand, the "use-without-owning" characteristic of services provides users with more open choices while avoiding the burden of service maintenance and upgrades.

Based on different user concerns, end-user-driven service composition can be categorized into data-oriented, process-oriented, and interface-oriented approaches.

Data-Oriented Service Composition: Users obtain required data by composing different data-providing services. During this process, data of interest is often presented directly to users through the interface, who then operate on the data to obtain explicit results. This approach is also known as direct data programming [3]. Key technologies required include: user-oriented data models, data operations, mapping between global and local models, and change maintenance. Depending on how data is organized and presented, such work can be divided into tree-structured data services, spreadsheet-structured data service composition, and nested-table-structured data service composition. Representative works include MashMaker [4], SheetMusiq [5], AMICO [6], and SpreadATOR [7].

Process-Oriented Service Composition: Users compose services with different functions by constructing workflows to describe composition logic. Such work typically provides users with a visual construction environment where they can complete workflow construction through direct manipulation such as dragging and linking. This approach can compensate for the shortcomings of data-oriented service composition, where program logic is generated by the system through recording and inferring user behavior—users lack understanding of the program logic and find it difficult to identify potential defects in the system' s inference process. In process-oriented service composition, program logic expressed through visual means is presented directly to users, facilitating observation and modification. Such work can be divided into dataflow-based and control-flow-based service composition. Representative works include Yahoo Pipes [8], IBM Damia [9], and Marmite [10].

Interface-Oriented Service Composition: The user interface serves as the interface between users and computer operations. Interface-oriented end-user service composition aims to support end-users in designing and creating applications through interface-layer composition. From the user' s perspective, what they see are interface components, while service interface information is hidden behind the presentation layer. This approach can provide users with a "what-you-see-is-what-you-get" experience. Key technologies in this area include service component interface generation, interface interaction, and legacy application interface integration. Such work can be categorized into client-side interface component integration, browser interface component integration, and Web application interface integration. Representative works include FAST Gadgets Visual Storyboard [11] and ServFace Builder [12].

2.2 Integrated Terminal Utilization

Reviewing the history of network computing reveals that user environment architectures have undergone multiple iterations and spiral evolution between client and server sides. Early computer systems were generally bulky and expen-

sive, with resources and services concentrated on mainframes, necessitating thin clients. Users logged into mainframes through terminals like the 3270 to perform business operations. Starting from the mid-1980s, traditional mainframes and dumb terminals were challenged by PC-based microcomputer networks, and client/server (C/S) gradually became the mainstream model. Client applications could provide users with rich interface elements and fast local computing capabilities. With Internet development and gradually increasing bandwidth, the thin client model regained momentum, with typical representatives including Network Computers (NC) and browser/server (B/S) architectures. While reducing client requirements and management maintenance costs, the thin client model increased server dependency and suffered from disadvantages such as requiring constant online connectivity, large network transmission volumes, and vulnerability to single-point failures or bottlenecks on the server side.

With advances in computer hardware technology and new-generation software technology, the utilization of client-side computing capabilities has once again gained attention. In network computing architecture, a trend has emerged to shift processing tasks originally on the server side to the client side. One current typical model eliminates the boundary between client and server, namely the peer-to-peer (P2P) computing model; another typical model re-divides processing tasks between client and server, what Giga Information Group calls "Return to Rich Client" and Gartner Group calls High Fidelity Client or Smart Client. This rich client model addresses the shortcomings of traditional B/S architecture by redefining the deployment and communication methods of the presentation, business logic, and data layers, transferring some processing tasks originally on the server side to the client side. Applications built under this model are called Rich Internet Applications (RIA) [5], with representative technologies including AJAX, Flex, JavaFX, Smart Client, and SilverLight.

With the development of rich client technology and the increasing variety and processing capabilities of terminal devices, user environments have evolved beyond traditional web browsers into RIA-type applications that can comprehensively utilize both cloud and client resources [13]. The role of the client includes not only receiving user input and displaying pages but also asynchronously communicating with the server based on user requests, maintaining data locally, and supporting offline processing and data synchronization. All these capabilities can operate without relying on continuous network connectivity between client and server, thereby delivering not only better interactive experiences but also being particularly suitable for various mobile terminal devices in mobile Internet environments.

Based on which side serves as the control center, RIA-type applications can be divided into two categories: server-side control center and client-side control center.

Server-Side Control Center: In this scenario, users' local data and programs need to be registered on the server side for unified management and control. To support client-side resource utilization and offline processing under changing

network conditions, key enabling technologies are required, including client-side data caching, network monitoring and automatic offline/online switching, data synchronization and consistency assurance, and client-side program automatic installation and update maintenance mechanisms. Representative work includes Mokafive in the virtual desktop field [14], which provides a "desktop-as-a-service" platform embedded with the open-source VMware Player. The server centrally stores virtual machine images, which users can run offline or online. Images can be accessed via Web login or downloaded for use, and can also be pre-installed on USB devices, mobile phones, or laptops, supporting seamless upgrades and automatic installation/uninstallation of virtual machines. Other work supports browser-based offline Web applications, such as Google Calendar, Docs, and Gmail, which utilize Google Gears technology [15] to enable Web applications to access client-side local SQLite databases through JavaScript APIs, allowing data operations to be performed locally and providing users with faster response times and better offline support.

Client-Side Control Center: In this scenario, application management and control are completed in the user' s local environment. At runtime, applications are decomposed into distributable execution fragments, which are allocated to either local or network environments for execution. In addition to client-side offline usage technologies common with server-side control center approaches, this category of work focuses on application partitioning strategies, performance monitoring and prediction techniques, and optimization scheduling mechanisms. Based on when application partitioning occurs, related work can be divided into static partitioning and dynamic partitioning. The former includes research by Fan Yang [16] on partitioning methods for data-driven Web applications to optimize user response time; the latter includes CloneCloud developed by Intel Labs [17], which comprehensively considers device processing capabilities, network conditions, cloud services, and workload to compute partitioning schemes and dynamically adapts adjustments based on runtime environmental changes.

2.3 Private Data Protection

In cloud computing environments, data privacy protection faces severe challenges. Since user data is stored and processed on operators' machines beyond user control, privacy concerns constitute a major factor affecting user adoption of cloud computing solutions [18]. Due to policy, legal, and intellectual property considerations, users often wish to keep certain core data and business processing on the client side to retain autonomous control over core operations and avoid the risk of private resource leakage that could result from transmitting private data to network environments.

Current research on privacy protection can be divided into two categories based on the object of operation: service resource-oriented and data resource-oriented. Service resource-oriented privacy protection involves selecting and scheduling resources based on service trust levels, such as CCOF [19] and GridSec [20]. Data resource-oriented privacy protection can be further divided into atomic

privacy and constraint privacy based on whether relationships between data resources are considered.

Atomic Privacy: This approach does not consider relationships between data resources and performs privacy policy description and protection on individual atomic data resources. It is relatively easy to implement but imposes high requirements on users, who must describe privacy policies for each atomic resource and ensure policy consistency. Related technologies for atomic privacy protection are divided into information-hiding-based data protection and policy-based data protection. Information-hiding-based data protection processes data through encryption, hashing, or anonymization techniques, representing privacy protection under the premise of data transmission. Policy-based data protection matches requests and resources before data transmission to determine whether private data can be sent to resources. For example, W3C' s P3P (Platform for Privacy Preferences) [21] is used to determine whether a Web site' s privacy protection policy matches user preferences, with the protection object being personal information related to user Web browsing. Reference [22] proposes a situation-aware access control mechanism that determines what information to provide during service matching based on context, controlling the visibility of information requested by services.

Constraint Privacy: This approach considers relationships between data resources, including descriptions of data relationships in the policy model and constraints that these relationships impose on privacy. It can simplify users' privacy policy expressions and provide richer constraint requirements. Related technologies for constraint privacy protection include: privacy policy models that support data relationship description, policy consistency determination, and application scheduling mechanisms that maintain policy consistency. Reference [23] considers data dependency relationships and proposes a privacy-preserving service composition framework that annotates application data with sensitivity levels and matches them with service models to determine whether privacy policies would be violated. This work describes dependency relationships between service input and output parameters through service models and propagates policies through data dependencies during service matching. Reference [24] considers interest conflicts between competing organizations in distributed workflow execution and proposes a distributed workflow "Chinese Wall" [6] model that protects sensitive workflow control data and provides secure and trustworthy distributed workflow execution. This work considers the impact of workflow control data on workflow execution flow and restricts access to workflow control data by organizations with conflicting interests. Reference [25] addresses separation of duties by grouping services with similar functions and scheduling services to maximize separation of duties. Separation of duties divides roles into mutually exclusive sets—once a user is assigned one role, they cannot possess other mutually exclusive roles. Similar work includes reference [26].

3. VINCA ProUsage

VINCA ProUsage provides users with a convenient and easy-to-use cloud client programming and usage environment. Its distinguishing feature is its user-centric design. On one hand, it can connect to the VINCA service community, providing users with a unified window for accessing and utilizing both cloud environment resources and local client resources. On the other hand, it supports users in flexibly composing resources to construct personalized applications that meet immediate needs, thereby reducing costs and improving application development efficiency. While enabling effective utilization of cloud computing infrastructure, it also preserves user-side control capabilities.

As shown in [Figure 1: see original paper], VINCA ProUsage primarily consists of two major parts: the client side and the cloud side. The client' s local environment includes three modules: the integrated resource view, the client programming Integrated Development Environment (IDE) [7], and the lightweight client engine. The cloud side comprises the client connection management module deployed on the service community and hosting environment foundation.

The main functions of each module are:

1. **Integrated Resource View:** This module implements interaction with the service community and presents the service resource view on the client side. It can simultaneously connect to multiple service communities and integrate the presentation of both remote resources and local client resources. Users can transparently use remote and local resources without concerning themselves with implementation-level details such as physical deployment or invocation interfaces. It allows users to customize service resources of interest within their service communities, forming a user-customized personalized resource view that lays the foundation for application construction.

2. **Client Programming IDE:** This module provides a user-driven, convenient, and flexible visual application integration development environment. It supports users in autonomously customizing personalized applications and making on-demand adjustments. It provides application correctness assurance mechanisms that can instantly verify users' programming operations and intelligently recommend service resources matching the current context.

3. **Lightweight Client Engine:** This module treats the client as the control center for application execution to organize and schedule resources. This model can be viewed as building a secure and transparent application execution environment for users, capable of automatically decomposing, scheduling, coordinating, and synthesizing applications while isolating and protecting local resources during optimized resource utilization.

4. **Client Connection Management:** This module implements interaction functions such as establishing connections between client and server,

session maintenance and migration, execution request submission, and result reception. It supports seamless connection and utilization of server-side resources through diverse terminal devices and across various network environments.

The technical features of VINCA ProUsage are:

1. **User-Driven Business Application Construction:** Aiming to support end-user programming, VINCA ProUsage supports both process-oriented and data-oriented service composition application requirements. It proposes three different end-user service composition technologies: exploratory service orchestration, business service-based business process modeling, and nested-table-based Mashup application construction. The first two approaches are process-oriented: exploratory service orchestration 主要针对 dynamic and immediate application needs, providing a flexible process modeling approach of "define-as-you-execute"; business service-based business process modeling 主要针对 typical and stable business process management needs, providing a modeling approach that enables business users to autonomously define executable business processes. At the system-assisted support level for user programming, VINCA ProUsage provides context-sensitive proactive service recommendation and automatic association mechanisms that can provide intelligent suggestions to users, achieving resources that "appear on demand"to assist user decision-making. It also provides instant verification mechanisms that can check and predict whether user-built applications conform to existing system constraints (consistency, conflicts, etc.), measure inconsistent programs, and feed back verification results and adjustment suggestions to users, improving the quality and efficiency of user programming.

2. **Client-Centric Application Scheduling and Execution:** VINCA ProUsage adopts a client-centric computing model. The client serves as the control center for application scheduling and execution, organizing and utilizing both local and network resources while coordinating execution between the client and remote nodes. The lightweight client engine performs operations such as business application decomposition, scheduling, coordinated execution, and data synthesis under comprehensive consideration of network conditions, client and server capabilities, and other factors. The client actively participates in application execution as a control center rather than passively relying on server operations. A crucial function of the client engine is application rescheduling. When service status or network environment changes, the client engine dynamically reschedules and deploys yet-to-be-executed application fragments in real-time according to rescheduling strategies to ensure uninterrupted application execution and achieve optimal execution efficiency.

3. **Private Data Protection:** VINCA ProUsage provides a client-side private data protection mechanism that supports users in describing privacy policies for individual data resources as well as constraint relationships

such as dependencies and mutual exclusions between data. Policies allowing or prohibiting disclosure of individual data resources can be determined before application runtime; policies described through constraint relationships often depend on runtime information and require runtime determination. The system can perform consistency verification and monitoring on policy sets. Through application scheduling mechanisms that combine static analysis and dynamic adjustment, it can not only detect private resource leakage risks during application design but also determine at runtime whether conflicts will occur and mediate conflicts promptly, ensuring that applications do not violate user-defined privacy protection policies for local data. Through this private data protection mechanism, users' private data and programs can be used directly in the local environment without the cumbersome steps of packaging them into services and registering them in network registries, while avoiding private resource leakage issues.

4. Summary

With the development of cloud computing infrastructure and terminal devices, computing itself is gradually shifting from "machine-centric" to "user-centric." To support end users in fully utilizing cloud infrastructure, cloud client research has garnered widespread attention from both industry and academia. This paper analyzes the autonomous control issues of cloud clients, decomposing them into three aspects: supporting users' personalized needs, diversified terminal utilization, and private data protection. It discusses the current state of research and then introduces the development of VINCA ProUsage cloud client software. VINCA ProUsage provides users with a convenient and easy-to-use cloud client programming and usage environment that supports user-driven business application construction, client-centric application scheduling and execution, and private data protection. It can balance effective utilization of cloud computing infrastructure with client-side control capabilities, providing a better user experience and helping promote the widespread adoption and application of cloud computing.

References

[1] Yanbo Han, Guiling Wang, Chen Liu, et al. *Principles and Practice of Internet Computing: Exploring the Essential Issues and Key Technologies Behind Grid, Cloud and Web X.0.* Beijing: Science Press, 2010.

[2] Eisenberg M. End-user programming. *Handbook of Human-Computer Interaction*. Amsterdam: Elsevier Science, 1997.

[3] Jones C. End user programming. *Computer*, 1995, 28(9): 68-70.

[4] Ennals R, Gay D. User-friendly functional programming for web mashups. *Proceedings of the 12th ACM SIGPLAN International Conference on Functional*

Programming, 2007: 223-234.

[5] Liu B, Jagadish H. A spreadsheet algebra for a direct data manipulation query interface. *VLDB*, 2009: IEEE Computer Society Washington, DC, USA.

[6] Obrenović Ž, Gašević D. End-User Service Computing: Spreadsheets as a Service Composition Tool. *IEEE Transactions on Services Computing*, 2008, 1: 229-242.

[7] Kongdenfha W, et al. Rapid development of spreadsheet-based web mashups. *Proceedings of the 18th International Conference on World Wide Web*, 2009, Madrid, Spain: ACM.

[8] Jones MC, Churchill EF, Twidale MB. Mashing up visual languages and web mash-ups. *IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC 2008)*, 2008.

[9] Altinel M, et al. Damia: a data mashup fabric for intranet applications. *Proceedings of the 33rd International Conference on Very Large Data Bases*, 2007, Vienna, Austria: VLDB Endowment.

[10] Wong J, Hong JI. Making mashups with marmite: towards end-user programming for the web. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2007, San Jose, California, USA: ACM.

[11] Hoyer V, Janner T, Delchev I, et al. "The FAST Platform: An Open and Semantically-Enriched Platform for Designing Multi-channel and Enterprise-Class Gadgets." *Service-Oriented Computing*, vol. 5900, Springer Berlin/Heidelberg, 2009, pp. 316-330.

[12] Feldmann M, Janeiro J, Nestler T, Hübsch G, Jugel U, Preußner A, Schill A. An integrated approach for creating service-based interactive applications. *Conference on Human-Computer Interaction (Interact)*, 2009.

[13] Höfer M, Howanitz G. *The Client Side of Cloud Computing.* University of Salzburg, Jul 1.

[14] MokaFive. http://www.mokafive.com/

[15] Google Gears. http://gears.google.com/

[16] Yang F, Gupta N, Gerner N, Qi X, Demers A, Gehrke J, Shanmugasundaram J. A unified platform for data driven web applications with automatic client-server partitioning. *WWW*, 2007.

[17] Chun BG, Maniatis P. Augmented smartphone applications through Clone Cloud execution. *HotOS*, 2009.

[18] Pearson S. Taking account of privacy when designing cloud computing services. *Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing*, 2009: 44-52.

[19] Zhao SY, Lo V. Result verification and trust-based scheduling in open peer-to-peer cycle sharing systems. Technical Report, University of Oregon, USA, 2005.

[20] Song SS, Kwok YK, Hwang K. Trusted job scheduling in open computational grids: security-driven heuristics and a fast genetic algorithm. *19th IEEE International Parallel & Distributed Processing Symposium (IPDPS-2005)*, Denver, CO, USA, IEEE Computer Society Press, Los Alamitos, CA, USA, April 4-8, 2005.

[21] Cranor L. The Platform for Privacy Preferences 1.1 (P3P1.1) Specification. W3C Working Draft, 2004. Available from: http://www.w3.org/P3P/

[22] Yau S, Liu J. A situation-aware access control based privacy-preserving service matchmaking approach for service-oriented architecture. *2007 IEEE International Conference on Web Services (ICWS 2007)*, 2007: 1056-1063.

[23] Xu W, Venkatakrishnan VN, Sekar R, Ramakrishnan IV. A framework for building privacy-conscious composite Web services. *4th IEEE International Conference on Web Services (ICWS' 06)*, Chicago, IL, September 2006: 655-662.

[24] Vijayalakshmi, C, Soon Ae, M Pietro. Chinese Wall security for decentralized workflow management systems. *Journal of Computer Security*, 2004, 12(6): 799-840.

[25] Hung PCK. From conflict of interest to separation of duties in WS-Policy for Web services matchmaking process. *Proceedings of the IEEE Thirty-Seventh Hawaii International Conference on System Sciences (HICSS-37)*, Hawaii, USA, 2004, 3: 30066b.

[26] Wang HJ, Leon Z. A formal approach to verification of process constraints. *16th Annual Workshop on Information Technologies & Systems (WITS)*, 2007. Paper available at SSRN: http://ssrn.com/abstract=1025600.

Author Biographies

Jing Wang: Assistant researcher and Ph.D. at the Service Computing Research Division, Institute of Computing Technology, Chinese Academy of Sciences. wangjing@ict.ac.cn

Weilong Ding: Ph.D. candidate at the Service Computing Research Division, Institute of Computing Technology, Chinese Academy of Sciences.

Shuan Zhao: Master' s student at the Service Computing Research Division, Institute of Computing Technology, Chinese Academy of Sciences.

Feng Li: Master' s student at the Service Computing Research Division, Institute of Computing Technology, Chinese Academy of Sciences.

Note: Figure translations are in progress. See original paper for figures.