Design of an Instant Messaging System Based on the IaaS Cloud Platform

Miaofan Sun, Shengsheng Wang, Zhiyi Fang, and Mengjiao Zhang
College of Computer Science and Technology Jilin University, Changchun, 130012, P.R. China
Email: fangzy@jlu.edu.cn

Abstract— Instant messaging product is one of the most important social network products. It promotes the development of communication, business and mobile networks. However, with the development of instant messaging system, some of the disadvantages gradually are showed out, such as the proprietary is too strong, repetitive development cycle is too long, and the safety is not ideal. In order to better solve these problems, we designed an instant messaging system based on the cloud platform. Firstly, we built IaaS (Infrastructure as a Service) cloud platform of the instant messaging service and implemented the function interface. Secondly, we built the Opnifre server which is a Real Time Collaboration (RTC) server licensed under the Open Source Apache License, realized the data conversion between XMPP (Extensible Messaging and Presence Protocol) and HTTP (HyperText Transfer Protocol), and developed the extension function plug-ins. Finally, we implemented the Android client based on the above platform. The Instant messaging system based on IaaS has a good performance which can be independently introduced to other business areas, extends the function of Openfire server and shortens development cycle.

Index Terms—instant messaging, Infrastructure as a Service (IaaS), openstack, openfire, android

I. Introduction

Instant Messaging (IM) system [1], [2] has rapid development in recent years, however, IM system at present has some limitations. It cannot be introduced to other areas of commerce, and cannot guarantee safety and maintainability. IM which is implemented based on Infrastructure as a Service (IaaS) [3], [4] cloud platform can solve the above problems.

IaaS [5] is an infrastructure and service. It provides a service which includes using of all computing infrastructure to consumers, such as CPU, memory, storage, networks, and other basic computing resources. The consumer does not manage or control any cloud computing infrastructure, but can choose operating system, storage space, deployment of application, and can also possible to obtain the limit control of the network components. OpenStack [6], [7] is a cloud computing management software which was researched and developed cooperatively by the U.S. National Aeronautics and Space Administration (NASA). On IaaS platform, Openstack mainly includes three major projects [8]: operation project of Nova, oriented object data storage project of swift and image file transfer service of glance. Application development of IM based on IaaS platform has become a new research direction in recent years. IM system based on IaaS cloud platform can be independent to other areas of Commerce, which can solve the IM system of proprietary strong problem. Expansion function of Openfire on open source servers can widen the domain functionality. And the development of IM based on the platform has a shorter development cycle.

In this paper our main contribution is as follows.

1. We built a private Iaas cloud platform with OpenStack. On the platform we implemented and encapsulated IM function interfaces which can solve the existing problem of IM system.
2. Based on the platform, we built Openfire server which implemented the communication between clients and the Openfire server.
3. We implemented data conversion between XMPP and HTTP protocol on Openfire server.
4. On the Openfire, we developed extension plug ins which made up the insufficient function of the Openfire server.
5. We implemented the Android client on the platform and verified the interface instances.

The remainder of this paper is structured as follows. In Section 2 we introduce the related work about IM system. In Section 3 we describe the system requirement analysis. In Section 4 we design the system. In Section 5 we implement and evaluate the system. In Section 6 we present the conclusions of this work and the next step work in future.

II. Related Work

In recent years, IM system technique is mainly as follows. Loesing Karsten *et al.* [9] proposed an IM system, which is explicitly designed to protect user presence without the need of a trusted central registry. Zhang weize *et al.* [10] employed an open instant messaging protocol called XMPP to build a web instant messaging system in browser/server structure for distance education. Bin Zhang *et al.* [11] designed and integrated a secure add-in seamlessly into the MSN client utilizing interfaces of the MSN client. Gao yunxiang *et al.* [12] proposed a new service model and implementation mechanism based on Jabber/XMPP. Zhao Sheng *et al.* [13]

Manuscript received April 15, 2015; revised September 9, 2015.
Corresponding author email: fangzy@jlu.edu.cn.
doi:10.12720/jcm.10.9.734-739

proposed a universal architecture that supports broadcasting of messages to users in a particular physical region. Liu Xiufeng *et al*. [14] presented a new approach of location service system which realizes location-based mobile IM chatting functions. Wei Ya Kun *et al*. [15] Applied Java language, MySQL database to develop an IM software on the basis of cross-platform Qt development framework and Android operating system. Wang Xiao-Yuan *et al.* [16] described the implementation in C# of a Client/Server pattern, and the design of an instant message application in LAN. Neviarouskaya Alena *et al*. [17] research addresses the tasks of recognition, interpretation and visualization of affect communicated through text messaging.

The above papers mainly concerning the function and structure of IM system, however, how to design an ideal platform, overcome the proprietary of IM system, avoid long repetitive development cycle, and improve the efficiency of the communication is still to be researched.

III. SYSTEM REQUIREMENT ANALYSIS

In order to implement IM system based on IaaS platform, we should build a cloud platform, develop the IM function, and deploy the IM function to the cloud platform which includes good external interfaces. On the client, the system should have the ability of processing local storage and data transmission. On the server, the system should have the ability of accepting connections, parsing protocol and connecting to the cloud platform.

According to the functional requirements, we planned seven interface modules. In the system, these seven modules of Openfire correspond to the interfaces of IaaS platform. The specific design of the seven modules is shown in IV section.

IV. SYSTEM DESIGN

A. *System Target*

The target of IM system based on IaaS platform is as follows:

- Our cloud platform server is build based on OpenStack, and the interfaces of IM function should be encapsulated.
- The multi-clients communicate normally through connecting the server. And the client's account and password should be safe and reliable.
- Openfire should be optimized. In view of requirement analysis of IM function, we should expand its functions, realize the protocol conversion on the server, and make up the deficiency of Openfire function.
- The communication between Openfire and cloud platform should be implemented and the platform interfaces should be called correctly.

B. *System framework*

IM system based on the IaaS platform is divided into three parts: the cloud platform, Server and Clients. The architecture of the IM system is shown in Fig. 1.

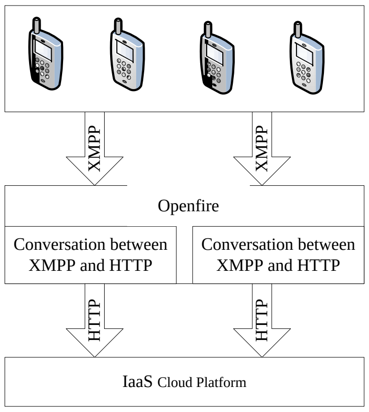

XMPP
XMPP
Openfire
Conversation between XMPP and HTTP
Conversation between XMPP and HTTP
HTTP
HTTP
IaaS Cloud Platform

Fig. 1. The architecture of the IM system.

- Cloud platform: We build a cloud platform based on OpenStack, and at the same time, we implemented and encapsulated the function interfaces of IM system. The composition of OpenStack project is as shown in Fig. 2. Openstack compute is cloud computing controller which provides a management tool for the cloud platform. It can run virtual machine instances, control project and user access to the cloud, and manage network at al. OpenStack Object Storage is an object storage system which is extensible. It can store a variety of data. Image service (Glance) is an virtual machine image service which can storage, query and detect on the virtual machines.

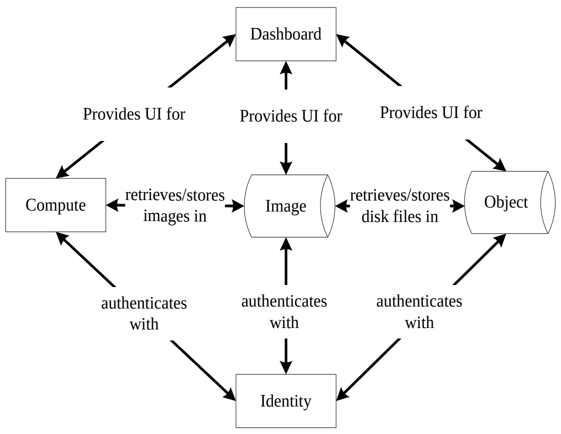


Fig. 2. The composition of OpenStack project

The step of building the cloud platform based on OpenStack is as follows.

Step 1: Three networks are defined in Virtual box. NET1 is management network, NET2 is public network, and NET3 is object storage network.

Step 2: Three virtual machines are created and started. One is controller virtual machine, and the others are fuel virtual machines.

Step 3: The nodes are configured as cloud computing nodes and the control nodes.

- Client: Client implements the interface functions of the cloud platform, which including user login management, friend management, one to one chat,

chat room, state management, footprint sharing and other modules. We propose specific design in Section IV-D.

- Server: According to the requirement of the platform, server extends functions, encapsulates the protocol interface of HTTP, and implements conversion between HTTP and XMPP [18]. The implement of protocol conversion is shown in Section V-D.

C. Communication Protocol of IM System

- Communication architecture of XMPP protocol: XMPP [19] which is client-server architecture does not combine with the specific network. The abstract of the architecture is shown as follows (The symbol "–" means XMPP communication, and the symbol "=" means arbitrary communication protocol).

$C1-S1-S2-C3$

$C2-+-G1=FN1=FC1$

$C1,C2,C3=Client\,of\;XMPP$

$S1,S2=Server\,of\;XMPP$

$G1=A\,Gateway\,between\,XMPP\,and\,an\,external\,message\,networ$

$FN1=An\,external\,message\,network$

$FC1=An\,Client\,of\;external\,message\,network$

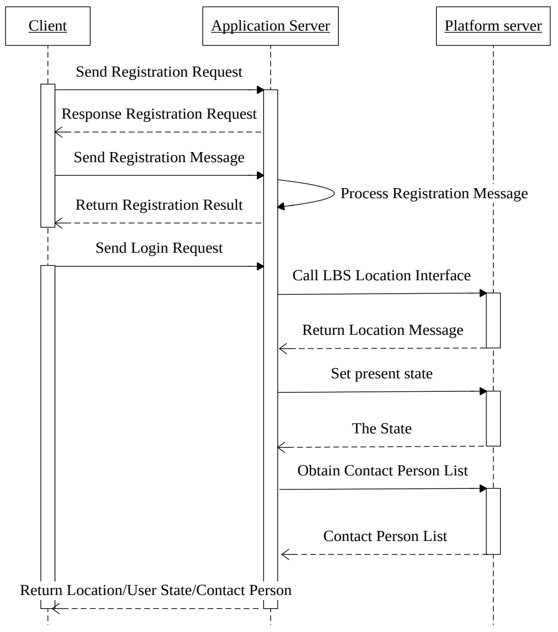
XMPP address structure: An entity in the network structure is considered to be a node. It has a unique identifier JID that is entity address which is used to identify a user or the other content, such as a chat room. An effective JID should include some elements such as domain identifier, node identifier, and resource identifier.

- XMPP message format: XMPP includes three top-level XML elements: Message, Presence, and IQ. Message represents the transmission message. When the user sends a message, the Message element will be inserted in the context of the flow. Presence represents the state of the user, such as online, offline, etc. When the user changes his state, a Presence element will be inserted in the context of the data flow. IQ represents a request-response mechanism which an entity sends a request, and another entity accepts the request and responds.

D. The Design of the Function Modules

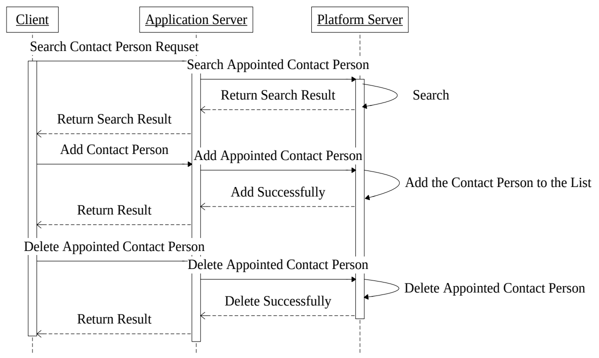
- User management: it includes operation of login, registration, and exit. We wrote the server domain name on Openfire [20]. When the user registers, the module determines whether the domain name of the server can be connected to the Openfire. If the domain name can be connected, the registration information will be sent to Openfire. Then Openfire will receive the XML format messages, and parse the message into an HTTP data flow which is sent to the cloud platform. The cloud platform will check processing, and encrypt the password which will be stored to the database. Then, the cloud platform returns successfully registered information to Openfire. Finally, protocol conversion will be finished, and sent to Client. User management interaction diagram is shown in Fig. 3.

- Friend management: It includes adding, deleting, finding friends and friends list. Client sends the request to Openfire which sends the request to the cloud platform through the interface. Cloud platform based on the Client request for processing. The Friend management interaction diagram is shown in Fig. 4.



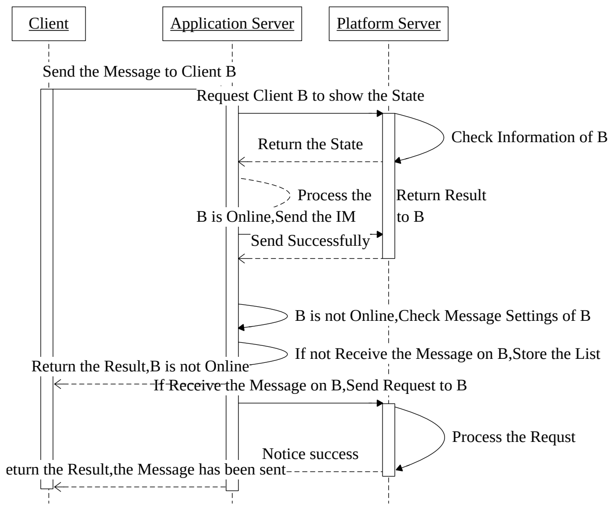
Client
Application Server
Platform server
Send Registration Request
Response Registration Request
Send Registration Message
Process Registration Message
Return Registration Result
Send Login Request
Call LBS Location Interface
Return Location Message
Set present state
The State
Obtain Contact Person List
Contact Person List
Return Location/User State/Contact Person

Fig. 3. User management interaction diagram



Client
Application Server
Platform Server
Search Contact Person Requset
Search Appointed Contact Person
Return Search Result
Search
Return Search Result
Add Contact Person
Add Appointed Contact Person
Add the Contact Person to the List
Add Successfully
Return Result
Delete Appointed Contact Person
Delete Appointed Contact Person
Delete Appointed Contact Person
Delete Successfully
Return Result

Fig. 4. Friend management interaction diagram



Client
Application Server
Platform Server
Send the Message to Client B
Request Client B to show the State
Check Information of B
Return the State
Process the
B is Online,Send the IM
Return Result
to B
Send Successfully
B is not Online,Check Message Settings of B
If not Receive the Message on B,Store the List
Return the Result,B is not Online
If Receive the Message on B,Send Request to B
Process the Requst
Notice success
eturn the Result,the Message has been sent

Fig. 5. One-to-one chat interaction diagram

- One-to-one chat: It is a communication between two clients. “A” sends the instant messages to “B”, at this time, “A” sends the request to Openfire which receives the request, formats the data, and sends the request to the cloud platform. After receiving the message, the cloud platform judges whether “B” is online. If “B” is online, the cloud platform will directly send the message to “B”, and “B” will receive the message to process. If “B” is not online, then the cloud platform will store the message to the offline message list, send the message to “B” when “B” is online, and delete the offline records. One to one chat interaction diagram is shown in Fig. 5.
- Group management: It mainly includes creating groups, updating group information, adding and deleting group members. The Group management interaction diagram is similar to figure4, so it not to be detailed here.
- Chat room: It is a process of a one-to-many chat. If a client speaks, Openfire will send the message to others clients in the chat room. This client sends the message to Openfire which receives the message, formats the data and sends the message to the cloud platform. After receiving the message, the cloud platform sends the message to the contact person, and this is one-to-one processing. After the clients return the feedback information, Openfire sends the information back to the user. The chat room interaction diagram is shown in Fig. 6.

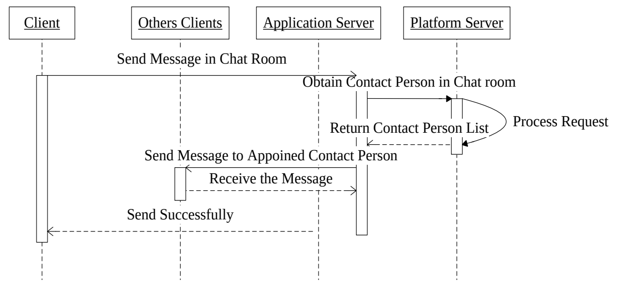

Client
Others Clients
Application Server
Platform Server
Send Message in Chat Room
Obtain Contact Person in Chat room
Process Request
Return Contact Person List
Send Message to Appoined Contact Person
Receive the Message
Send Successfully

Fig. 6. Chat room interaction diagram

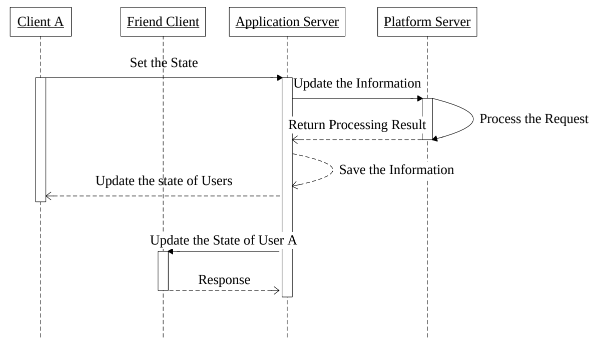


Fig. 7. State management interaction diagram

- State management: Users have their own login status which is online, off-line, and busy, etc. State management can modify the state of the users. If Openfire receives the request of modification, the request will be sent to the cloud platform which will modify and store the user’s state. Then, cloud platform will return back the results to the Openfire, and Openfire will push the state to the others clients by broadcasting. The State management interaction diagram is shown in Fig. 7.
- Footprint sharing: When the user shares the micro-blog information to platform, Openfire will receive the request and send the request to the cloud platform. Cloud platform will return the processing result to Openfire according to the request. The Footprint sharing interaction diagram is shown in Fig. 8.

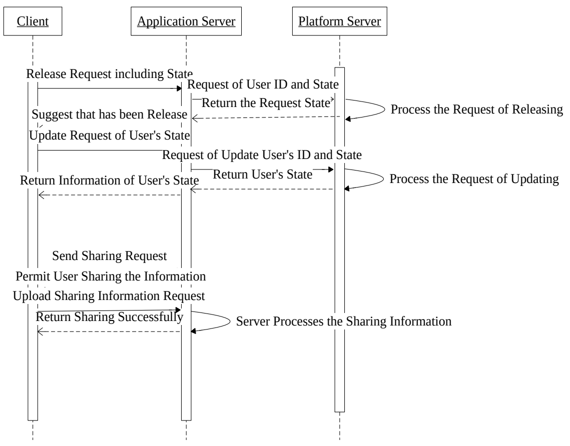

Client
Application Server
Platform Server
Release Request including State
Request of User ID and State
Return the Request State
Process the Request of Releasing
Suggest that has been Release
Update Request of User's State
Request of Update User's ID and State
Return User's State
Process the Request of Updating
Return Information of User's State
Send Sharing Request
Permit User Sharing the Information
Upload Sharing Information Request
Return Sharing Successfully
Server Processes the Sharing Information

Fig. 8. Footprint sharing interaction diagram

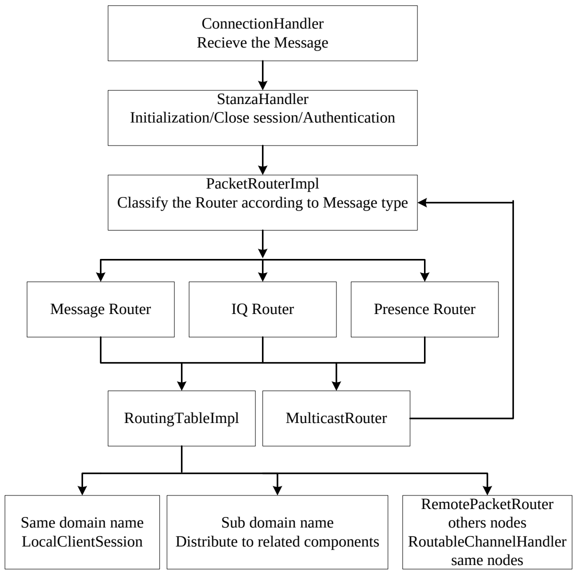


Fig. 9. Data flows parsing of Openfire

E. *Openfire Sever Design*

The design steps of Openfire sever is as follows:

Step 1: Openfire source code deployment. We packed the plug-ins of Openfire , Set running data and configured database.

Step 2: Openfire source codes parse. When Openfire running, it receives messages by ConnectionHandler, and processes the message by StanzaHandler. Openfire parses the message according to the nodes of XML’s byte flows. The result of parsing is sent to PacketRouterImpl which classify the messages. The detailed data flow of parsing is shown in Fig. 9.

V. System Implememntation and Evaluation

A. Openstack Build

In this study, we used Openstack to build cloud platform which need to use three PC. If conditions are limited, the environment can also be instead of the virtual machine. The detailed environment deployment configuration: at least for dual core processor, memory more than 4G, and disk more than 200G. We used fuel file to deploy the Openstack.

B. Instant Messaging Interface Implementation

In order to implement the instant messaging interface, we wrote the compute function modules by secondary development. The steps are as follows.

Step 1: We created instant messaging interface file. In the interface file, we verified the sending data from the client and stored the data in the database.

Step 2: Call the interface needs a display page. We added the layout of the page and the display of obtained data according the requirement.

Step 3: We created the access address of the interface and configuration of the router. We created the interface resource, and determined the interface for GET or POST.

Through the above steps, we implemented and encapsulated the instant messaging interface. If the interface is updated on openstack, we should restart the Nova.

C. Mobile Client Implemention

The logic structure of application implementation module for client is shown in Fig. 10. We took the footprint sharing module as an example. And the others modules are not to be detailed here. On Openfire, the footprint sharing is an extension function which client can send message and share their tracks. On the map, the client can also show the location and the content of the release information. The user interface of footprint sharing is shown in Fig. 11, the user interface of chat room is shown in Fig. 12. The others user interfaces are not to be displayed here.

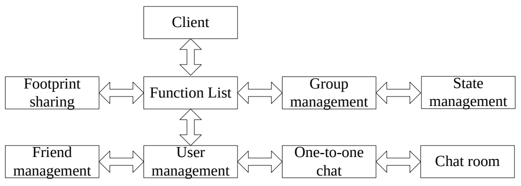


Fig. 10. Logic structure of application module for client

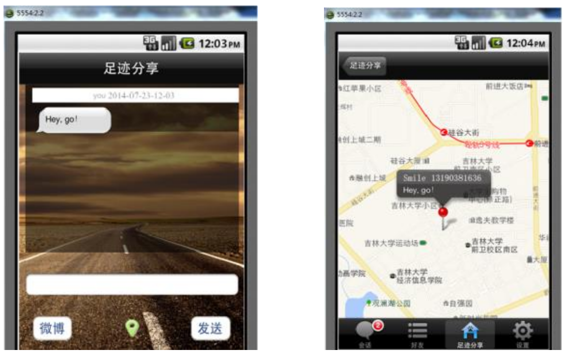
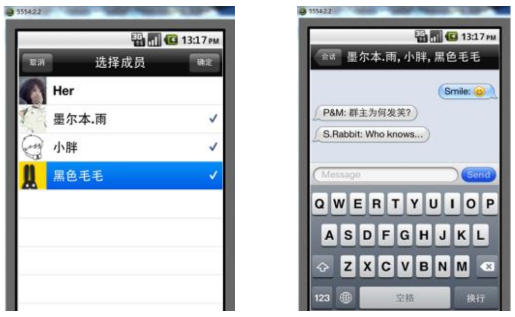


Fig. 11. The user interface of footprint sharing



Smile:
P&M: 群主为何发笑?
S.Rabbit: Who knows...

Fig. 12. The user interface of group member and chat room

D. Openfire Protocol Conversion Implementation

On Openfire, data flow received from the client is XML data format. However, IaaS platform olny uses the HTTP protocol. So the Openfire needs conversion between XMPP and HTTP protocol. We implemented the interface of protocol conversion. The steps are as follows.

Step 1: We parsed the XML data flow, found the corresponding key information, and called protocol conversion interface which encapsulated the information to JSON format.

Step 2: When the interface is called, we used the get/post method in HTTP to call the corresponding interface of the IaaS platform. And the JSON data information is sent to the platform, then waiting for the feedback results of the authentication.

Step 3: The feedback results is HTTP format which are parsed and encapsulated for XML format to the client.

E. System Performence Evaluation

In order to test the performance of the system based on the IaaS cloud platform, we compared the processing time in a single machine environment and IaaS environment. According to the comparison, the advantage of IaaS platform is very obvious with the increase of the amount of data. The performance of comparison is shown in Fig. 13.

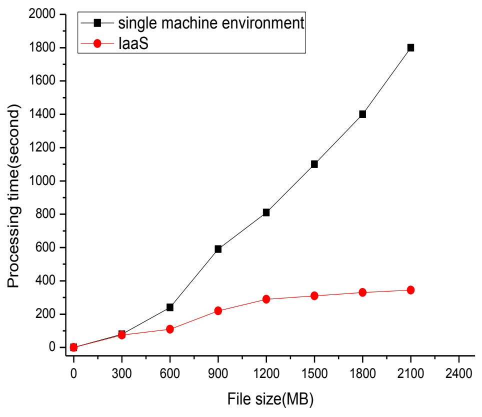


Fig. 13. The processing time of a single machine environment and IaaS

VI. Conclusions

In this paper, we designed and implemented an IM system based on IaaS cloud platform. We fist built a private Iaas cloud platform with OpenStack, and on this platform, we implemented and encapsulated IM function interfaces. Secondly, we built Openfire server which implements the communication between clients and the Openfire server, and implemented data conversion between XMPP and HTTP protocol on Openfire server. Thirdly, on the Openfire, we developed plug ins which made up the insufficient function of the Openfire server, and implemented the Android client on the platform. Finally, we compared the processing time in a single machine environment and IaaS environment. According to the comparison, the advantage of our IM system based on IaaS cloud platform is very obvious.

Our IM system based on the IaaS cloud platform has some follow-up work to do. In order to promote the performance of the system, we will optimize the performance of the cloud platform, continue to develop the IOS client and web version.

REFERENCES

[1] B. Y. Han, X. Y. Liu, J. Wang, *et al*., "Extending an instant messaging system with data services and mashups thereof," in *Proc. IEEE International Conference on Services Computing*, October 2014, pp. 848-849.

[2] X. F. Gu and L. Yan, "Instant messaging system based on Jabber," in *Proc. 10th International Computer Conference on Wavelet Active Media Technology and Information*, 2013, pp. 214-217.

[3] P. Xu, R. Hu, and S. Su, "Research on resource management in PaaS based on IaaS environment," *Communications in Computer and Information Science*, vol. 401, pp. 145-157, 2013.

[4] J. F. Jiang, X. S. Chen, and L. Chen, "A vulnerability scanning framework based on monitoring agents for IaaS Platforms," *Journal of Sichuan University (Engineering Science Edition)*, vol. 46, pp. 116-121, July 2014.

[5] C. A. Alexandra, D. Djawida, A. C. Orgerie, and P. Guillaume, "Towards energy-aware IaaS-PaaS Co-design," in *Proc. 3rd International Conference on Smart Grids and Green IT Systems*, 2014, pp. 203-208.

[6] K. R. Hassan, Y. Jukka, and A. A. Shohel, "OpenID authentication as a service in OpenStack," in *Proc. 7th International Conference on Information Assurance and Security*, 2011, pp. 372-377.

[7] W. C. David, S. Kristy, L. Craig, F. Yann, and G. Damien, "Adding federated identity management to openstack," *Journal of Grid Computing*, vol. 12, no. 1, pp. 3-27, March 2014.

[8] S. S. Suhas and S. S. Shilpa, "Comparing openstack and VMware," in *Proc. International Conference on Advances in Electronics, Computers and Communications*, January 2015.

[9] L. Karsten, R. Maximilian, W. Christian, and W. Guido, "Implementation of an instant messaging system with focus on protection of user presence," in *Proc. 2th International Conference on Communication System Software and Middleware and Workshops*, 2007.

[10] W. Z. Zhang, "Distance education oriented web instant messaging system," *WIT Transactions on Information and Communication Technologies*, vol. 58, pp. 943-952, 2014.

[11] B. Zhang, M. Feng, H. R. Xiong, and D. Y. Hu, "Design and implementation of secure instant messaging system based on MSN," in *Proc. International Symposium on Computer Science and Computational Technology*, 2008, pp. 38-41.

[12] Y. X. Gao, C. B. Xiao, C. Q. Gao, and H. G. Chen, "A study on jabber-based instant messaging system for mobile networks," in *Proc. IEEE International Symposium on Knowledge Acquisition and Modeling Workshop Proceedings*, 2008, pp. 884-887.

[13] S. Zhao, X. Feng, Z. Chen, Z. Li, and J. H. Ma, "MobiMsg: A resource-efficient location-based mobile instant messaging system," in *Proc. 2nd International Conference on Cloud and Green Computing and 2nd International Conference on Social Computing and Its Applications*, 2012, pp. 466-471.

[14] X. F. Liu, L. G. Zhang, X. J. Zhan, and P. P. Chen, "Location-based mobile instant messaging system," in *Proc. 2nd International Conference on Consumer Electronics, Communications and Networks*, 2012, pp. 1886-1889.

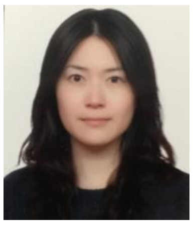
[15] Y. K. Wei, L. X. Zuo, C. B. Shao, Y. Fu, and Y. Wan, "The design of instant messaging system based on Qt-Android system," *Advanced Materials Research*, vol. 834-836, pp. 1915-1918, 2013.

[16] X. Y. Wang, X. Feng, and D. L. Wang, "Research for key technologies of internal instant messaging system development," in *Proc. International Conference on Mechatronic Science, Electric Engineering and Computer*, 2011, pp. 326-329.

[17] N. Alena, P. Helmut, and I. Mitsuru, "User study of affect IM, an emotionally intelligent instant messaging system," *Lecture Notes in Computer Science*, vol. 5208, pp. 29-36, 2008.

[18] X. F. Bai and M. Yang, "Design and implementation of web instant message system based on XMPP," in *Proc. IEEE 3rd International Conference on Software Engineering and Service Science*, 2012, pp. 83-88.

[19] F. G. Wang, X. W. Yang, L. Zhang, and Y. B. Zhang, "Design of environmental monitoring system based on XMPP," in *Proc. 2nd International Conference on Measurement, Information and Control*, vol. 1, pp. 509-512, 2013.



Miaofan Sun is currently a Ph.D. candidate in Computer science and Technology of Jilin University. Her research interests include the areas of distributed computing system and parallel computing.

Shengsheng Wang received his PhD degree in computer science from Jilin University in 2003. From 2004 to 2007, he worked with the Mathematics Institute of Jilin University as a postdoctoral researcher. From 2009 to 2010, he worked with the national lab NCGIA at the University of Maine in US as a visiting professor. His research interests include spatio-temporal reasoning, computer vision and pattern recognition. He published more than 80 papers including 12 indexed by SCI, nearly 70 indexed by EI. As a key member, he has accomplished 5 National 863 Projects and 5 NSFC projects (including one Major Research Program). He has received 1st prize of Science & Technology Progress Award of Jilin Province twice and 2nd prize of Science & Technology Progress Award of Jilin Province once.

Zhiyi Fang received the PhD degree in computer science from Jilin University, Changchun, China, in 1998, where he is currently a professor of computer science. He was a senior visiting scholar of the University of Queensland, Australia, from 1995 to 1996, and the University of California, Santa Barbara, from 2000 to 2001. He is a member of China Software Industry Association (CSIA) and a member of Open System Committee of China Computer Federation (CCF). His research interests include distributed/parallel computing system, mobile communication, and wireless network.