Open Access
Full Length Article

ORIGINAL ARTICLE

Distributed IDE on High-speed Internet Android Dandled Devices

Muhammad Munwar Iqbal[1], Ghulam Dastagir[2], Yasir Saleem[3],

[1]Assistant Professor, Department of Computer Science, Universtiy of Engineering and Technology, Taxila, Pakistan
[2] MS Scholar, Department of Computer Science, University of Engineering and Technology, Taxila Pakistan
[3]Associate Professor, Department of Computer Science, Universtiy of Engineering and Technology, Lahore, Pakistan

ABSTRACT

Distributed Computing is an efficient technology for internet services and entire IT world. Developments in today's world, distributed computing provides software, hardware, and information as services that make IT development cost-effective. Often developers face a tough time in setting up their development tools on their local computer. This problem can be solved if we provide those tools and environment as a service in the Cloud. Both cloud and distributed computing in a crossway, these technologies will support to form an efficiently, powerful distributed cloud. This paper will demonstrate the architecture of a Distributed IDE across the network with a cloud host. It has explained those challenges that come across while implementation and provide a brief solution. In the final section, we have explained the messages and listeners behaviors, and how can actually carry out the whole process. It has used a hybrid approach to attain a full distributed cloud IDE with one cloud server that entertains the different jobs from different devices in the network.

Keywords: IDE, cloud, distributed cloud, security, Java, Android IDE, JVM

Author`s Contribution
[1] Data analysis, interpretation and manuscript writing, Active participation in data collection
[2] Conception, synthesis, planning of research and manuscript writing
[3] Interpretation and discussion, Data Collection

Address of Correspondence
Muhammad Munwar Iqbal
Email: munwariq@gmail.com

Article info.
Received: Feb 27, 2017
Accepted: May 30, 2017

Cite this article: *Iqbal MM, Dastagir G, Saleem Y. Distributed IDE on high-speed Internet Android Dandled Devices J. inf. commun. technol. robot. appl.2017; 8(1):31-36.*

Funding Source: *Nil*
Conflict of Interest: *Nil*

INTRODUCTION

In today's era, distributed computing is a common technology that is efficient not only for internet services but for the whole IT world. On the other hands, cloud computing is using remote servers for data processing, computing, and storage. The use of both cloud and distributed computing in a crossway, both of these technologies will support to form an efficiently distributed cloud. Cloud as distribution is the application of cloud computing technologies to join data and applications served from multiple physical locations. Distributed, in information technology (IT) context, means that something is common among multiple systems which may also be in different locations. Distributed cloud speeds up the inter-process communications for global services and enables more open communications for specific regions. The distributed cloud computing is a growing technology. It fills several parts, allocating data for use by distributed applications, for data safety, and for other reasons. We

have seen an upsurge in the number of distributed applications. Non-distributed applications lack the reliability that is required to work within a cloud, whereas distributed jobs add a certain amount of reliability to an application. Depending upon their architecture, the lack or failure of one part of a distributed application will never shut down the entire application. The cloud computing has started to make an impact on the tools, techniques and support solutions that are running IT industry [1, 2]. This includes automation of datacenter, backups, recovery, configuration automation, and performance analysis [3]. Like we have seen cloud versions of middleware in the form of Platform-as-a-Service (PaaS), Agile solutions, code versioning of software versioning systems like Github, continuous integration of code and system testing.

Yet despite this transformation, there has been little confusion to the integrated development environment (IDE) world. Why hasn't the development environment moved to the distributed IDE on the cloud? Cloud solution of Integrated Development Environment became an immerging technology in recent past, but when an IDE becomes available as a service on the cloud, it becomes a headache for devices with low processing power and power resource [4, 5]. Building a couple of classes and then sending them on the cloud to compile is not a difficult thing to handle for an android device but when the code consists of a chunk of classes and different libraries, it becomes difficult to build the code and then receive output on the same device.

Another drawback was connection interruption during the process as if the device sends the code for compilation and server gets down or fails to establish a connection with the device. In this case, the user will be again sending the code for compilation which will result in more utilization of memory, more processing power to rebuild the code and more battery consumption. If we think of different scenarios that explain different drawbacks. Another unspoken downside to a single device hosted IDE is putting data on the application layer. When we talk about cloud, we have to keep bandwidth and packet size in mind. With thin clients and low processing power, it becomes difficult for the user to send a significant amount of precious form of text. It's been noticed that enterprises who have adopted the IDE as a service don't recommend its access from thin clients.

The Android phone's battery has a tough job to do all the time with plenty of services and applications running in the background. When some lines of code are built on a device, and service is invoked to compile that code on the cloud, it takes enough battery power if we look ideally [6]. Then a response is generated on the cloud to show output on the screen of the android device. This mechanism involves virtualization of some resources on an Android device like allocation of some memory for the output processes [7]. This kind of mechanism clearly consumes enough battery power and results in the bad user experience of cloud IDEs.

During the process of sending the code to the cloud and getting outputs back on the device, it takes some time, depends on the bandwidth of the internet connection. Throughout the process, Android device stays idle, and all the threads are active until cloud makes a response [8]. This results in the utilization of significant processing power and usage of cache memory. Imagine a processor converting source code into bytecode, and some runtime exceptions occur and that output needs to be again converted into bytecode to put on application layer so it can be displayed on the android screen, but it will be again converted to plain text or some UI [9]. I think that would be too unfair with a processor to do in a couple of seconds. Such stumbling blocks pave the way for distributed cloud IDE. Distribute cloud-based IDEs are integrated, making it easy to share. Developers can co-edit, co-build, or co-debug, changing the entire nature of pair programming. Cloud can propose improvements in system competence and compactness, giving each individual workspace a configurable portion of the accessible memory and calculate resources.

The security is another serious issue that needs to be part of the debate. With each passing day, we hear about the cloud theft and cloud data loses and that makes the cloud customer adopt conventional ways. When a service becomes available in the cloud, special surveillance needs to be developed to secure the cloud. If the cloud is distributed and its resources are not centralized, that can prevent the cloud from malicious attacks. If someone somehow breaks into the system and starts using the service, then the server may not be responding to him back. Instead, it will generate a message on some user-

defined end that will help to identify an attack.

LITERATURE REVIEW

Due to the perks of cloud computing, many desktop applications have been transferred to the cloud. Lots of IDEs are also moved into the cloud because they can bring developers a lot of suitability for their developing process, but some threatening problems are still not touched [10].

During past decade, big amount of web-based IDEs has been released to facilitate the software developers. Most of them just help to compile some chunk of code, but some of them can actually develop and deploy complete web applications. Like Arvue, it's a browser-based IDE that enables simple development of web applications. Arvue applications are created in the browser using a UI designer and a unified code editor program. The source code is stored in a predefined VCS (version control system) provided by Arvue and that can be published on the cloud server [11-13]. Arvue runs on a dedicated Jetty Webserver and applications created and published using Arvue IDE are hosted on Amazon Web Services cloud to get high scalability. It offers all the tools needed for creating, publishing and hosting the applications[14].

Some cloud-based IDEs are developed for embedded systems entirely based on open-source solutions that have prospects to ease some conventional-IDE heaps. In contrast to existing solutions, these IDEs support flashing and debugging of a target board connected to the host computer while the complete development takes place in a cloud environment[15, 16].

At Microsoft research, a web-based IDE Touch Develop has been developed that enables programming on thin clients especially mobile phone devices [17]. Programs can be modified and debug on the device without a secondary device. Transitioning between programming on one mobile device and resuming on another device is unified. Moreover, this web application also works offline. It delivers a green-field platform to explore IDE and designs of programming language, as well as runtime techniques and distributed data storage concepts [18].

RESEARCH METHODOLOGY

Our proposed work includes a server that acts as a bridge between Android devices while code compilation is in process. There could be more than one android devices in a cloud distributed network and a server gets the code from a client device and shows the output on a different android device specified by the device that initiated the debug process.

These issues and drawbacks endorse the need for Distributed Integrated Development Environment. It will distribute the integral parts of the system among Android devices, i.e. build, source code, output and will speed up the old thin client IDEs by pre-allocating of functions that are needed to be performed by the device.

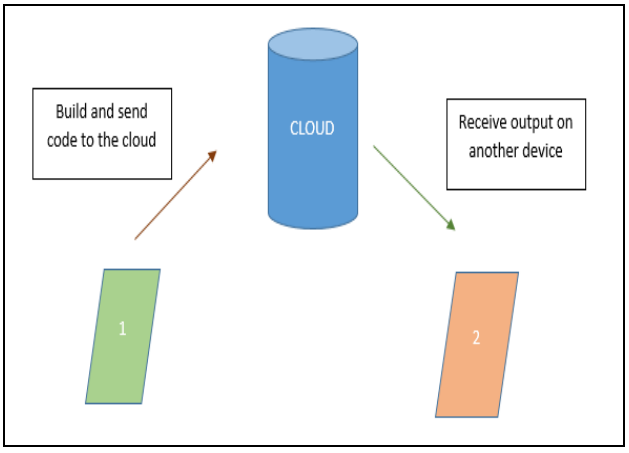

Fig 0.1 Basic architectural diagram of distributed cloud IDE

RESULTS AND DISCUSSION

The biggest challenge in a Distributed application that implements cloud-based IDE is a secure connection. A well-established connection would be the backbone of our system. The system comprises of different libraries and APIs that help the client devices perform actions in the cloud. How to control the already and new running instances of compilation [19, 20]. How to insert breakpoints on remote devices access? These would be some challenging questions that we would try to answer in this section. A class of core package will be dealing Threads among server and client devices. An initiated thread can only be in one particular state at a given point in unit time. States can be new, blocked, runnable, terminated or waiting. These states are solely states of a virtual machine which do not imitate any thread states of

the operating system.

The second most important task to carry out will be control of references among different nodes. Concerned class or interface sets up all of the appropriate cross-references between components at runtime. Almost all of the functionality of Distributed Cloud IDE is hidden in components with extremely high-level interfaces. This class ensures that each device component is registered with all other components on a cloud server that need to call its methods. In essence, this class manages associations of all the components in the system.

Startup class will consist of purely static members. Main method will read the <systemIdentifier>.jar file on the cloud to get the critical information to start the built or compile process depends on the request. Other utility classes like Code Navigator, File Node, Project Profile, Compiler Listener, etc. will be the part of the mobile module. Every thin client will have an interface to generate output or show any activity that is generated against a request. Like, an interface or class to represent source errors and warnings generated by the compiler running on a cloud.

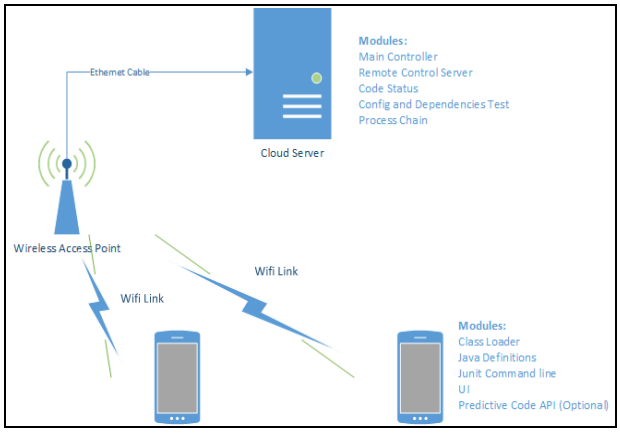

Fig 0.2 Network flow diagram

An interface will be tracking all event listeners to an Interactions Model that has the capability to notify them of some event when invoked on any device available in the network. This class has been assigned a specific role of managing Interactions Listeners in the system. Other classes will be using alike or slightly different code to execute the same function for other event interfaces, e.g., JavadocEventNotifier and GlobalEventNotifier. Such classes or interfaces implement the suitable interface definition in such a way that they can be used transparently as composite packaging for a specific listener interface (will be implemented next) on the specific Android device. Components which might achieve their own list of listeners use EventNotifiers instead of simplifying their internal implementation as we do in our Java applications. Notifier interfaces should, therefore, be considered a private implementation of device module and should never be used straight outside of the local host component.

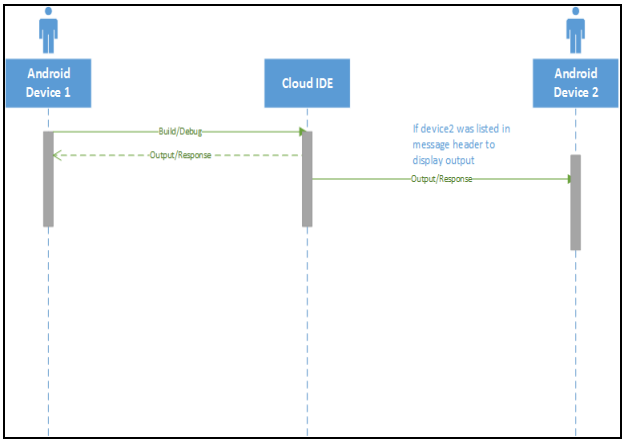

Fig 0.3 Sequence diagram of messages between nodes

All interfaces in the system must use the synchronization methods provided by java read/write streams. It guarantees that multiple notifications (reads) can occur concurrently, but only one thread can be adding or removing listeners (writing) at a unit time, and no “reads” can occur during a “write.” We will not be using any traditional Java synchronization methods here to avoid bad or untraceable compilation results.

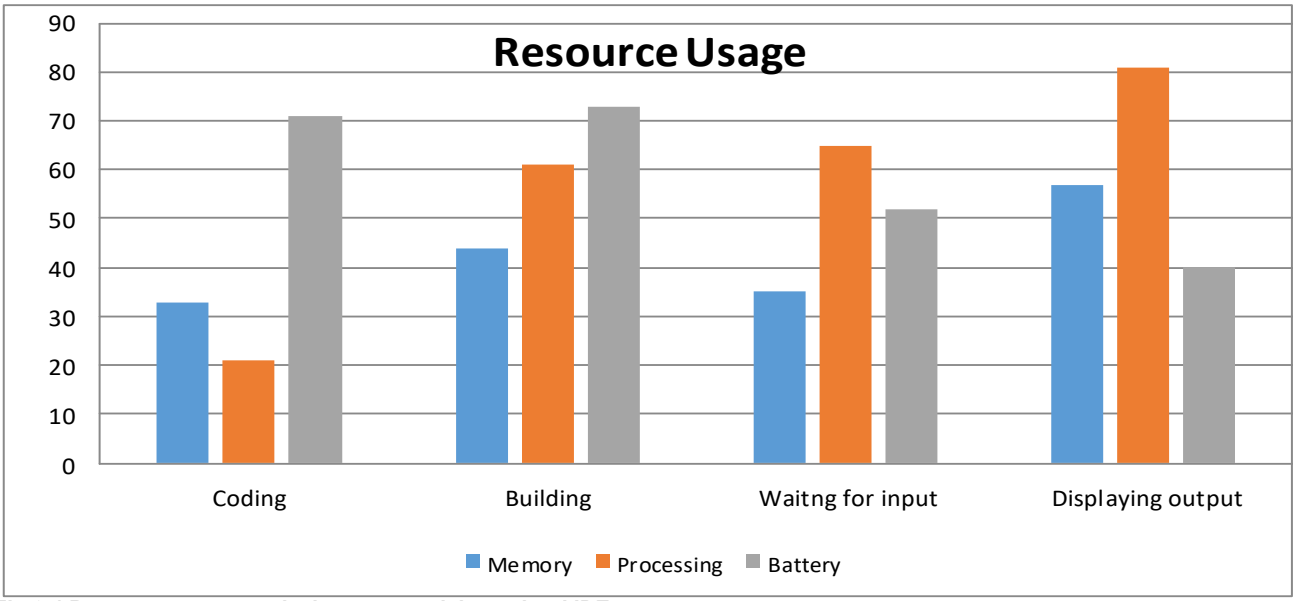

Fig 0.4 Resource usage analysis on a standalone cloud IDE

CONCLUSION

Any service that pays off a significant number of loads on processing devices can be migrated to a distributed architecture to help utilize the more processing speed and power resource. Apparently, it's a tough job for every developer to have the same environment wherever they are. Cloud-based compilers will run on thin clients will surely help every developer to develop everywhere by utilizing their devices in an efficient way. More than one developers can make their own network and carry out development process and can share their code through the cloud. The coder can keep on writing the code and then occasionally builds it to show the results on the team lead's phone screen. Mobile phone comes handy when people have to get their hands-on software development in countries where power issues are dominant. Plenty of tasks are handled by the cloud server which plays a role of the server in distributed cloud IDE environment. Devices consume fewer power resources as they are idle during the code compilation process. Keeping the architecture of this system in mind, in future we can eliminate the usage of cloud server and distributed, its tasks in devices with better processing power and more memory resources. A distributed cloud IDE will surely help thin client users to makes use of their devices in an efficient way for development purposes.

REFERENCES

1. Khan, M. A., Iqbal, M. M., Ubaid, F., Amin, R., & Ismail, A. (2016). Scalable and secure network storage in cloud computing. International Journal of Computer Science and Information Security, 14(4), 545.
2. Iqbal, M. M., Saleem, Y., Naseer, K., & Kim, M. (2018). Multimedia-based student-teacher smart interaction framework using multi-agents in eLearning. Multimedia Tools and Applications, 77(4), 5003-5026.
3. Routray, R. R., Sharma, U., Uttamchandani, S. M., & Verma, A. (2012). U.S. Patent No. 8,121,966. Washington, DC: U.S. Patent and Trademark Office.
4. Schermann, G., Cito, J., & Leitner, P. (2018). Continuous Experimentation: Challenges, Implementation Techniques, and Current Research. IEEE Software, 35(2), 26-31.
5. Taherizadeh, S., Jones, A. C., Taylor, I., Zhao, Z., & Stankovski, V. (2018). Monitoring self-adaptive applications within edge computing frameworks: A state-of-the-art review. Journal of Systems and Software, 136, 19-38.
6. Missen Saad M, K.D., Aqib Mughees., A comparative study of diversity in search results of search engines. M. J. inf. commun. technol. robot. appl., 2017. 8((1)): p. 1-7.
7. Nair, S. R. (2018). U.S. Patent No. 9,910,689. Washington, DC: U.S. Patent and Trademark Office.
8. Pathak, A., Jindal, A., Hu, Y. C., & Midkiff, S. P. (2012, June). What is keeping my phone awake?: characterizing and detecting no-sleep energy bugs in smartphone apps. In Proceedings of the 10th international conference on Mobile systems, applications, and services (pp. 267-280). ACM.
9. Wang, C. L., Lam, K. T., & Ma, R. K. K. (2013). A computation migration approach to the elasticity of cloud computing. In Network and Traffic Engineering in Emerging Distributed Computing Applications (pp. 145-178). IGI Global.
10. Wu, L., Liang, G., Kui, S., & Wang, Q. (2011, July). CEclipse: An online IDE for programming in the cloud. In Services (SERVICES), 2011 IEEE World Congress on (pp. 45-52). IEEE.
11. Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R. H., Konwinski, A., ... & Zaharia, M. (2009). Above the clouds: A

berkeley view of cloud computing (Vol. 4, pp. 506-522). Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley.
12. Ardagna, D., Ciavotta, M., & Passacantando, M. (2017). Generalized nash equilibria for the service provisioning problem in multi-cloud systems. IEEE Transactions on Services Computing, 10(3), 381-395.
13. Ghose, M., Verma, P., Karmakar, S., & Sahu, A. (2017, December). Energy Efficient Scheduling of Scientific Workflows in Cloud Environment. In High-Performance Computing and Communications; IEEE 15th International Conference on Smart City; IEEE 3rd International Conference on Data Science and Systems (HPCC/SmartCity/DSS), 2017 IEEE 19th International Conference on (pp. 170-177). IEEE.
14. Ghazizadeh, A., & Olariu, S. (2018). Vehicular Clouds: A Survey and Future Directions. In Cloud Computing for Optimization: Foundations, Applications, and Challenges (pp. 435-463). Springer, Cham.
15. Frontoni, E., Mancini, A., Zingaretti, P., Contigiani, M., Di Bello, L., & Placidi, V. (2018). Design and test of a real-time shelf out-of-stock detector system. Microsystem Technologies, 24(3), 1369-1377.
16. Yee, J., Low, C. Y., Hanapiah, F. A., Zakaria, N. A. C., Mohammad, U., & bin Abd Rahman, R. (2017, September). System-level design of a cloud-based training device for upper limb spasticity rehabilitation. In Robotics and Manufacturing Automation (ROMA), 2017 IEEE 3rd International Symposium in (pp. 1-6). IEEE.
17. Ball, T., Burckhardt, S., de Halleux, J., Moskal, M., Protzenko, J., & Tillmann, N. (2015, May). Beyond open source: the TouchDevelop cloud-based integrated development environment. In Proceedings of the Second ACM International Conference on Mobile Software Engineering and Systems (pp. 83-93). IEEE Press.
18. Fähndrich, M. (2014, January). Lessons from a Web-based IDE and Runtime. In Proceedings of the ACM SIGPLAN 2014 Workshop on Partial Evaluation and Program Manipulation(pp. 1-2). ACM.
19. Dinh, H. T., Lee, C., Niyato, D., & Wang, P. (2013). A survey of mobile cloud computing: architecture, applications, and approaches. Wireless communications and mobile computing, 13(18), 1587-1611.
20. Roman, R., Lopez, J., & Mambo, M. (2018). Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges. Future Generation Computer Systems, 78, 680-698.