International Conference on Computational Intelligence: Modeling Techniques and Applications (CIMTA) 2013

SaaS Oriented Generic Cloud Compiler

Chandan Banerjee \textsuperscript{a,c,*}, Anirban Kundu \textsuperscript{b,c}, Rana Dattagupta \textsuperscript{d}

\textsuperscript{a}Netaji Subhash Engineering College, Kolkata 700152, India
chandanbanerjee1@gmail.com

\textsuperscript{b}Kuang-Chi Institute of Advanced Technology, Shenzhen 518057, P.R.China
anirban.kundu@kuang-chi.org

\textsuperscript{c}Innovation Research Lab (IRL), Howrah, West Bengal 711103, India
anik76in@gmail.com

\textsuperscript{d}Jadavpur University, Kolkata 700032, India
rdattagupta@cse.jdvu.ac.in

Abstract

In this paper, we have proposed cloud compiler as software-as-service (SaaS), which means compiler transforms source code written in a programming language (source language) into machine language in cloud computing environment. In our application, even if the client machine has no specific compilers installed, the user can write/upload a program and submit it to cloud system. Cloud system can provide different compilers for various source languages as a service, like C, C++, Java or any other languages. Compiler as a cloud service is the creation of several cloud clusters which corresponds to the various compilers that can be accommodated by the cloud service. Each cloud cluster has its own specific captive server which returns the compiled source code to the main server. The program is processed in our cloud architecture and the error/output is returned back to the client. We have proposed serial and parallel program allocation from server to backend tiers and also compared average Turnaround Time and Maximum Turnaround Time to complete all programs with two and four backend tiers.

© 2013 The Authors. Published by Elsevier Ltd. Open access under CC BY-NC-ND license.
Selection and peer-review under responsibility of the University of Kalyani, Department of Computer Science & Engineering

Keywords: Cloud Compiler; SaaS; IIS server; Front Tier; Middle Tier; Back Tier

1. Introduction

Cloud computing is cost-efficient and flexible usage of IT services. The services are offered just-in-time over the

\* Corresponding author. Tel.: +91-9830127266.
E-mail address: chandanbanerjee1@gmail.com
Cloud Computing is broadly classified into three services: "Software", "Platform" and "Infrastructure" [1, 2]. Each service serves a different purpose and offers different products for businesses and individual people around the world. The first type of service in Cloud Computing is Software as a Service (SaaS) [3, 4] which provides an application on the basis of users requirement. It is a type of model for software deployment in which an application is hosted as a service provided to the user across the network. There is no need to install and run the application on the user’s own computer. User does not have to purchase resources; just has to pay for what he/she uses during a particular session. SaaS reduces the user’s burden of software maintenance and support. The second type of service in Cloud Computing is Platform as a Service (PaaS). The idea of PaaS is that some users can provide the specific hardware and a certain amount of application software as a foundation by which other users can build their applications. This service is termed as an integrated solution over the clouds. User needs not to bother about the internal architecture of machine, operating system usage, so on. Google App Engine (GAE) [5] is an example of PaaS. The third type of service is Infrastructure as a Service (IaaS), which includes computing control and storage. Amazon Web Services (AWS) [6] is one of the pioneers of such an offering.

A compiler, which is transforms source code from a higher level language to a lower, machine level language. This is mainly done in order to create executable files which can then be ‘run’ in order to execute the program and its instructions.

In this paper, we are going to propose compiler as service, which comes under software-as-service (SaaS). Section 2 shows the related work of our approach. Section 3 presents system design framework. Experimental results are presented in Section 4. Conclusion is depicted in Section 5.

2. Related Work

Cloud compiler implies a service oriented architecture, reduced information technology overhead for the end-user great flexibility, reduced cost and on demand services. In [7], the authors proposed an online Java compiler using cloud system. In [7] the client machine doesn’t having java development kit, but still users can run Java program from his/her machine. To describe an online compiler which helps to reduce the problems of portability and storage space by making use of the concept of cloud computing have discussed. Emre Kiciman et.al [11] a system that separates an Internet service’s logical functionality from the architectural decisions made to support performance, scalability and reliability. In [11], described several program transformations in the performance optimization space which show how, through the use of restricted programming model and runtime tracing, optimization decisions can be automated.

3. System Design Framework

The main reason for creating our system is to provide a distributed compiling scheme. Also, it would act as a centralized repository for all the codes written. Our cloud system can provide different compilers for various source languages as a service, like C, C++, Java or any other high languages. There would be no need to maintain separate compilers at the client side. Authentication and personalized task distribution would be made possible. Figure 1 shows, the overall system design of our cloud compiler system. The functionality of the cloud system is divided into 3 major tiers- a) Front Tier, b) Middle Tier and c) Back Tiers.

The front tier is the user interface and the database of the cloud compiler system. The user interface is basically a webpage hosted on the IIS server [8] and is used by the users to input their program. Programs submitted by clients are taken by the IIS server and inserted into the database. The database is provided by Microsoft SQL Server [9]. The database forms an integral part of the front tier. Programs are controls and handle resource allocation decisions [10]. The middle tier acts as the central centre of the cloud compiler system because it is interacting actively with the front tier and managing the back tier and hence houses the Compiler Control Centre. To manage its various activities with the back-tier the middle tier uses a Scheduler which can ping, receive acknowledgement and send data. The middle tier also has a receive output server which receives the output data packet and stores it in the database. The Compiler Control Centre has basically three parts. The Compiler Server Management enables the addition of new back tier compilation servers, viewing the status of existing compiler servers and removal of compiler servers. Figure 2 shows the complete cloud compiler system. Working principle of the serial scheduler is discussed in algorithm 1.
Figure 1. Overall system design

Figure 2. Complete program execution flowchart
Algorithm 1: working principle of the serial scheduler (allocation of server to backend tiers).

Step 1: Get list of all the programs from the database
Step 2: Get next program on the list
Step 3: If End of Program List is reached, goto Step 1
Step 4: Check to see whether the program is allocated a compiler server but the output is not received within a specific time.
If TRUE goto Step 5(a), else goto Step 6
Step 5(a): Decrement the allocated Compiler Server Pending Status.
Step 5(b): Increment The Compiler Server's Priority by 2 (that is, for the next 2 programs, the server will be at a lower priority)
Step 5(c): Mark the corresponding program as unscheduled then goto Step 2
Step 6: If program sent but not allocated a compiler server then goto step 2 else get all compiler servers from the database.
Step 7: Sort according to priority in a priority list
Step 8: Get next compiler server from the priority list
Step 9: If End of Priority list is reached then goto step 2 else goto step 10
Step 10: If server_status=running & server_lang=program_lang and program not scheduled then goto step 11 else goto step 8
Step 11: Connect to server and send the program
Step 12: IF: reply= program rejected then goto step 13 else Step 14
Step 13: Mark program as not scheduled Increment server priority by 2 and goto step 8
Step 14: Increment compiler server pending priority status by 1 Set program.serverid=Compiler server id Set program sent flag=1 and goto step 2.

The scheduler sends fetches the un-compiled program from the database and sends the program datagram to the back-tier compiler server. The receive output server receives the compiled program packets sent by the back-tier compilers and then stores them in the database. The IIS server in the Front-Tier fetches the output from the database and sends them to the web-client/browser. The most important feature of the receive output server is that it receives the compiled program data packet from the back tier compiler. The back tier consists of “n” number of compiler servers which are used to compile and execute the programs like C, C++, Java etc. The back tier compiler requires the IP Address and port number which must be specified. We can also check the system usage which gives us the CPU usage and the available RAM.

We have applied Algorithm 2, where program has been allocated parallel way from server to backend tiers.
All program submitted to scheduler server at the same time.

Algorithm 2: Parallel allocation of server to backend tiers.

Step 1: Get list of all the programs from the database
Step 2: Get next program on the list
Step 3: If End of Program List is reached, goto Step 1
Step 4: Check to see whether the program is allocated a compiler server but the output is not received within a specific time.
If True goto Step 5(a), else goto Step 6
Step 5(a): Decrement the allocated Compiler Server Pending Status.
Step 5(b): Increment The Compiler Server's Priority by 2 (that is, for the next 2 programs, the server will be at a lower priority)
Step 5(c): Mark the corresponding program as unscheduled.
Step 5(d): Goto Step 2
Step 6: If Program is not allocated a Compiler Server (that is unscheduled) and a Thread is not allotted for the
program then goto Step 7 else goto Step 2
Step 7: While Number of allocated Threads exceeds the Thread Limit, if (while condition is true) then continue; else (while condition is false) then goto Step 8
Step 8: Allot a Thread to the Program and increase the Thread Count and goto Step 2.

For the allotted Thread's Function

Step 9: Get a list of all compiler servers from the database according to priority
Step 10: Get the next compiler server on the list
Step 11: If end of Compiler Server List is reached, goto Step 2
Step 12: Check whether the Compiler Server Status = Running and Corresponding Program Language = Compiler Server Language If True goto Step 13 else goto Step 10
Step 13: Connect to the Compiler Server and send the program
Step 14: Check whether the Reply = Program Accepted. If True goto Step 15(a) else goto Step 16(a)
Step 15(a): Increment Compiler Server's Pending Status
Step 15(b): Increment Compiler Server's Priority by 1 (that is lower priority for the next program)
Step 15(c): Set Program Server ID= Compiler ServerID
Step 15(d): Mark the program as scheduled
Step 15(e): Decrement Thread count and goto Step 2
Step 16(a): Increment Compiler Server's Priority Status by 2 (that is lower the priority for next 2 programs)
Step 16(b): Goto Step 10

4. Experimental Results

In this section, experimental results have been shown as per our proposed approach. The characteristics of the machines, considered in our experiments, are as follows:

(i) Server Configuration
   1) Intel Core i7 CPU @ 3GHz
   2) 8 GB RAM
   3) 500 GB HDD
   4) Windows 7 Home Premium

(ii) Back Tiers
   1) Intel Core 2 Duo
   2) 2 GB RAM
   3) 320 GB HDD
   4) Windows XP-SP3

Software Used: Compulsory Requirement

   1) VMware Workstation 6.0.0
   2) Microsoft Visual Studio 8
   3) Microsoft Sql Server 2005
   4) Microsoft .NET framework 3.5
   5) Jdk 1.6
   6) MinGW for C and C++
   7) Microsoft Sql Server Management Studio

Figure 3 shows maximum turnaround time with four backend tiers and Figure 4 shows average turnaround time with four backend tiers using serial scheduling respectively. Serial scheduling principle has been discussed in Algorithm 1. We have used a simple ten seconds delay program in Java for testing purpose in serial and parallel allocation. Figure 5 shows using Algorithm 2, average time taken to complete all programs with two backend tiers and Figure 6
shows average time taken to complete all programs with four backend tiers. Using Algorithm 2, Figure 7 and Figure 8 shows maximum time taken to complete all programs with two backend tiers and maximum time taken to complete all programs with four backend tiers respectively.

Network is one basis of the Cloud, so an unavoidable issue is that network bottlenecks often occur when large data is transferred. Bottleneck burden clearly shown from nature of curves in our experimental results in Figure 3 and Figure 4. Delay in transmission over TCP/IP network. The delay typically occurs when a system’s bandwidth cannot support the amount of information being relayed at the speed it is being processed. There are, however, many factors that can create a bottleneck in a system. Bottlenecks affect network performance by slowing down the flow of information transmitted across networks. In Figure 5 to 8 shows, the effect of the Parallel allocation of programs from server to backend tiers.
Figure 5. Average Turnaround Time with Two Backend Tiers using parallel allocation

Figure 6. Average Turnaround Time with Four Backend Tiers using parallel allocation
Figure 7. Maximum Turnaround Time with Two Backend Tiers using parallel allocation

Figure 8. Maximum Turnaround Time with Four Backend Tiers using parallel allocation
5. Conclusion

In this paper, a cloud compiler has successfully designed in form of a SaaS based on a scheduling algorithm. The scheduling algorithm based on memory usage and available RAM has been tested in the compiler control centre and does its scheduling of programs to the appropriate servers based on the said algorithm. Apart from this all the requirements of the system design are met and successfully implemented in a distributed architectural system. Back-tier machines successfully correspond and communicate in an asynchronous manner with the middle-tier elements. The Front-tier sends the data to the database which stores it. Output cannot be greater than 8KB in size. System has a restriction to execute programs which has runtime greater than a specific time period. Different compilers have been implemented unlike [7]. This would eliminate the need to install compilers separately. Another advantage of our cloud compiler system is that whenever the compiler package is to be upgraded it can be done easily without re-installing it on each machine.

References

5. www.appengine.google.com
6. www.aws.amazon.com