DYNAMIC ENERGY RESOURCE ALLOCATION FOR PARALLEL TASK BASED APPLICATION IN CLOUD COMPUTING

[1]Abdullah Chaudhary,[2]Ameya Ahir,[3]Moinuddin Ansari, [4]Yaseen Jamadar,
[1-4]Graduate Students
[1-4]Computer Engineering Department,
[1-4]Pillai HOC College of Engineering & Technology, Mumbai University,Rasayani,Maharashtra, India

Abstract : Green Computing is a recent trend in computer science, which tries to reduce the energy consumption and carbon footprint produced by computers on distributed platforms such as clusters, grids, and clouds. we propose a real-time dynamic scheduling system to execute efficiently task- based applications on distributed computing platforms in order to minimize the energy consumption.
A Chatbot is a virtual person that can effectively talk to any human being with the help of interactive conversion textual skill. Now a days there are many cloud-based platforms available for developing and deploying the chatbot such as Microsoft bot framework, IBM Watson, Kore, AWS lambda, Microsoft Azure bot service, Chatfuel, Heroku and many more but all those techniques has some drawbacks such as built-in Artificial Intelligence, NLP, conversion service, programming etc

IndexTerms - :-Distributed Computing ,NLP ,Multi-hueristic Computing ,Task Based Application ,Artificial Intelligence

I. INTRODUCTION

Recent studies [1,2] have estimated that around 1.5%–2.0% of the total energy consumption is consumed by data centers, and this energy demand is growing extremely fast due to the popularization of Internet services and distributed computing platforms such as clusters, grids, and clouds. Regarding the efficiency of data centers, studies have concluded that, in average, around 55% of the energy consumed in a data center is consumed by the computing system.

"CHATBOT". It is a computer program designed to interact with users via textual or auditory methods us- ing artificial intelligence. We can also call it as a Personal digital assistant. Now we are living in an era where everything is available through the internet. We can get all type of information but this can be possible by typing for that the chatbots are invented. Depending on the application use such as shopping, customer services, food order, news updates, reservation etc. we can develop the chatbot. The popularity of messaging apps suggests people will happily talk to chatbots. Basically, chatbot requires chat interface or chat win- dow which takes user inputs and provide a response message to the user.

The bots can be developed for entertainment or business purpose. Depending on the way we developed the bots, we can categorize the bot in two way one is Command-based bots and the other is smart bots. Command-based bots are manually programmed by a developer with the help of user inputs. The functionality of com-mand-line bots is limited as they are not using the cognitive services to programmed the bots. Smart bots depend on artificial intelligence to interact with users. Instead of going through the predefined answer, smart bots predict response message based on the context and previous message.

This paper presents the introduction of cloud-platform which can be used to develop the chatbot. There are many cloud-platforms such as Microsoft Azure bot service, IBM Watson, Chatfuel, Heroku, Kore, AWS lambda etc. which provides bot services and built-in cognitive services which are easy to configure chatbot. All the cloud-platforms differs with their own features and functionalities that I will com- pare in analysis part. Azure Bot Service[2] provides an integrated environment in which we can deploy, build, test, and connect with different channels that interact naturally wherever your users are interacting. With the IBM Watson Conversation service[3], we can create chatbot application that combines cognitive approaches build and train a chatbot using intents and entities and constructing dialog to simulate conversation. AWS Lambda[4] is not a Cloud Chat bot platform, but it uses lambda functions which facilitate easy creation and deployment of Chatbots. Kore[5] is a complete enterprise grade platform-as-a-service (PaaS) that enables companies and developers to design, create and deliver superior, highly intelligent, Natural Language Processing (NLP) enabled bots for use in many commu- nication channels. Another chatbot service is Chatfuel[6] which is very simple to develop chatbot with no programming language skill is required. It can be easily integrated with Facebook and other social channels. Heroku[7] is cloud platform that enables develop- ers to build, run, and test chatbot which supports built-in artificial intelligence as well as programming languages.

1.1. Front End

Front end can be any channel which provides the chat interface to interact with the bot. The popular messenger such as Facebook, Telegram, Skye, Slack and much more provide chat interface to the users to interact with bot user-friendly. There are some programming languages such as Node.js is one example offers its own real time

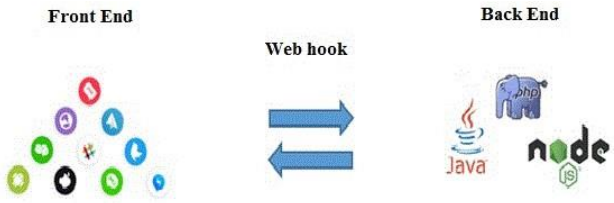

Figure 1: working of chatboat

1.1. Back End

Chatbots can be built on any cloud platform which may support programming language that allows you to make a web API. Program- ming languages such as Node.js or PHP, which are commonly used to build a chatbot but Java and Python provide inbuilt bot libraries as well. The backend is responsible for receiv-ing user input, pro- cess the user message using some external APIs or Algorithms, and generate the response message.

1.2. Connecting the two

The connection between bot and the chat interface is done using Web Hook- URL based connection. A web hook conveys user message to other applications as it happens, meaning you get data immediately. If we want to integrate our bot to the Facebook messenger, initially we have to set up web hook then we can interact with bot using Facebook messenger.

II. LITERATURE SURVEY:-

There are many techniques and other services such as Speech-to-Text, Text-to-Speech, and natural language processing etc. where the bot can be interactive. Kader et al.[9] presented the design techniques for developing interac-tive chatbots. They used Natural language processing techniques such as NLTK which can be used to analyze speech and make the bot response intelligent. They have done the survey of nine selected studies and also discussed the comparison between the chatbot design techniques. The authors in [10] discussed the different chatbot strategies and also compared the conversion techniques such text-based conversion and speech based conversion. They also discussed some parameters which affect human-computer interaction quality in conversational systems which can be used to design web interface. Nowa-days many different chatbots are found through the web. Pereira and Coheur[11] described their own chatbot platform "Just.Chat" which can be used to process the information for developing the chatbots. They also discussed "Ed-gar" platform which is designed for answering natural language questions. Based on the filter such as Domain filter, Personal filter, and Blacklist Filter, they identified the interaction and created chatbots knowledge bases.

Cloud services are separating its application from its hardware and software dependencies. There are many Cloud service providers including Google, Microsoft and Amazon Web Services etc. Gandhi and Kumbharana[12] presented a comparison between Amazon Web Service and Microsoft Azure platform for choosing the cloud ser-vices. They have compared AWS and Microsoft Azure based on some parameter such as Base plan price, virtual CPU core, RAM, Disk space, IDE support, server OS type and much more. Selection of cloud service provider depends on the application requirement and the cloud services that are necessary to develop the application.

Figure : Chatbot Architecture

III. METHODOLOGY

With the recent increase in the popularity of chatbot, we need to think about the interaction between the user and the bot. We need to analyze user input message, and the bot response as well[13]. Figure 2Shows the general chatbot architecture. The interaction begins with user input message what user is talking about. Then the User input message will be processed through two modules intent classification and entity recognition. Intent classification module checks the user input message and identifies the purpose user message. Based on the number of intents and context of the input message it identifies the intents. Entity recognition module recognizes user message struc- tured and extracts the main keyword from the bits of information, for example, The Airline bot can extract city and date. Both Intent classification and entity recognition modules are very important to find out the intents and entities throughout the interaction between the user and a bot. Next im-portant module in chatbot architecture is response generator. It uses some external APIs and algorithm to generate the response. The response generator use intent and entities, as well as the context of the conversation, extracted from the last user message.

Now-a-days many cloud-platforms provide the bot services where we can develop the bot and deploy to any one of the cloud. There are some cloud-platforms which provide different services apart from the bot service such as built-in artificial intelligence, Cognitive services etc. In this paper, we have used the Microsoft Azure cloud platform to develop the chatbot. Microsoft bot framework consists of Bot builder, bot Connector, and bot directory. It also has an emulator where we can test the working of the bot. Bot builder SDKs for Node.Js, Net are available to build the bot. If we want our bot to be more interactive, we can incorporate Microsoft cognitive services such as Language Understanding Intelligence Service (LUIS) [14]. Figure3Shows Microsoft Bot Framework which describes how the users communicate[15, 16, 17, 18, 19] with the bot using the mes- senger apps. In Microsoft bot framework, the role of bot connector is very im-portant. The responsibility of bot connector is to connect with the different channels some of which are Facebook, Skype,

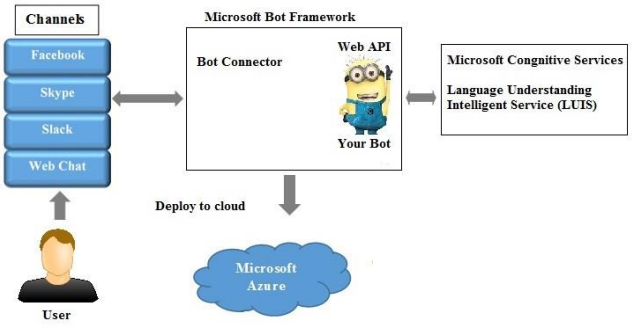

Figure 2: Microsoft Bot Framework

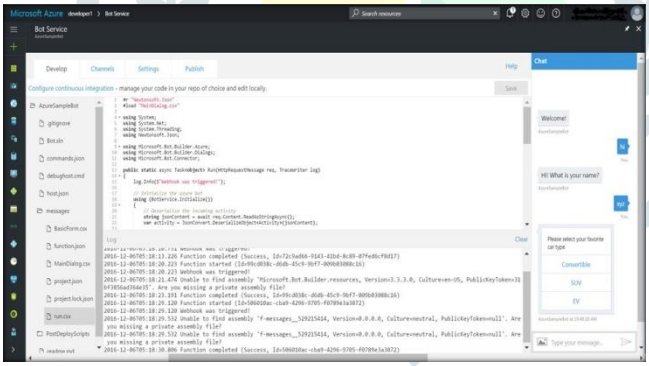

Figure 3: Microsoft Azure Integrated Environment

IV. RESULTS AND ANALYSIS:-

1.1. Microsoft bot framework:

As we have already discussed the supported cloud platform to build a chatbot in the First section. There are many advantages and disad- vantages of chatbot cloud platforms based on their function-alities and features. In this section, we compare all the bots devel-oped in different cloud environments and their results. There are many cloud platforms, but here we discuss most commonly used platforms these includes Microsoft Azure bot service, IBM Watson, Heroku etc. We show the integrated environment as well as analysis table which are used to develop the chatbot based on user requirement. Figure4. shows the integrated environment of Mi-crosoft azure where we can code, test, deploy, and publish the bot. Table1depicts the features, pros, and cons of Microsoft Azure platform.

1.2. Heroku

Figure5. Shows the integrated environment of Heroku cloud plat- form. It is also similar to Microsoft Azure, but it doesn't have in-built Artificial intelligent so it is that much popular than Microsoft Azure platform. Table2depicts the features, pros, and cons of Heroku platform where we can analysis the result of chatbot.

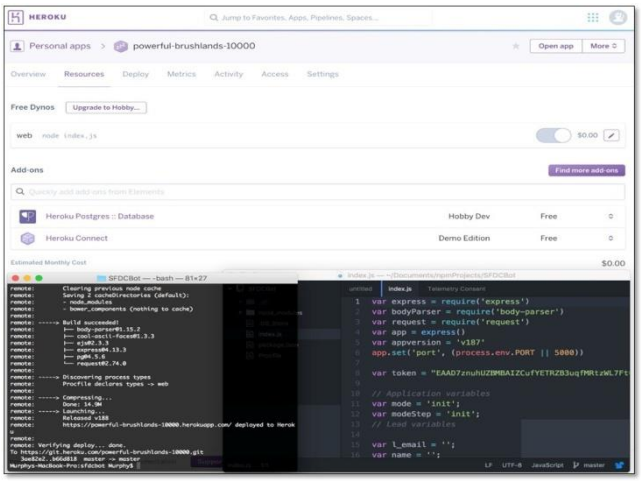

Figure 4: Heroku Bot environment

Figure6. Shows basis flow of IBM Watson conversion service. This is very simple to design the dialog, but the main disadvantage is we cannot design dynamic flow of conversions. The Watson Conversion service consists of Intents, Entities, and dialog through which we can build interactive smart bot. Table3depicts the features, pros,

1.3. IBM Watson

Figure7. Shows the comparison between all the cloud platform by comparing some parameter some of they are AI Built-In, pro-gramming need, time to setup. Supported IDE etc. The below table also shows the pros and cons of all the cloud platform. Chat bot development is varied depends on who is going to develop the bot. If the developer is not from a coding background, he/she can choose KORE and Chatfuel platform to build the chatbot, but in order to build interactive chatbot, it should have in-built AI. Below table shows cloud platforms which have the build-in AI such as KORE, Microsoft bot framework, and IBM Watson. We can use different IDE to develop chatbot and some of the cloud platform has its own built-in IDE some of these are KORE, Chatfuel, Microsoft, Azure, AWS Lambda and IBM Watson. By this analysis table, anyone can choose the cloud platform in order to build and deploy the chatbot.

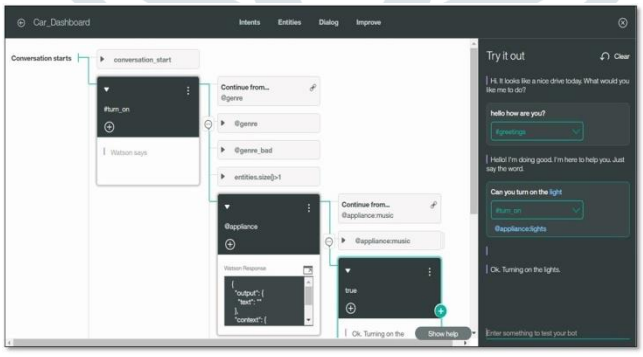

Figure 6: IBM Watson Bot Service

	KORE	Chatfuel	Microsoft Bot Framework	Microsoft Azure	HEROKU	AWS Lambda	IBM Watson
AI Built in	Yes	No	Yes	Yes	No	No	Yes
Programming needed?	No	NO	Yes	Yes	Yes	Yes	Yes
Complexity	High	Low	High	High	High	High	Medium
Time needed for setup	10 min	10 min	1 hour	15 min	2 hours	1 day	4 hours
Pre-study	8 hours	4 hours	8 hours	8 hours	8 hours	16 hours	16 hours
IDE	Build in	Build in	Visual Studio	Build in	Eclipse Atom	Build in Eclipse / command line	Build in Eclipse
Pro's	Extensive + all on platform	Plug'n'play	Most extensive	Integrated environment	Single development + deployment platform	Serverless deployment	High quality of interaction
Con's	Steep learning curve	Limited possibilities	Needs setup+ deployment	Only preview mode	Steep learning curve	Steep learning curve	Limited options to integrate with other services

Figure 7: Companion Table for all Cloud Platforms

Table 1: Analysis table for Microsoft Azure cloud platform

Channels	Programming Languages	Artificial Intelligenc	Integrates with	Pro's	Con's
Slack Facebook Messenger Skype GroupMe Telegram Twilio	Node.Js,C#	Built in - LUIS	Any API via programming	- No IDE required Integrated environment - Runs on Azure serverless architecture (Azure Functions) - Well documented	- Runs currently in preview mode - Required Azure subscription

Table 2: Analysis Table for Heroku Platform

Channels	Programming Languages	Artificial Intelligenc	Integrates with	Pro's	Con's
Slack Skype Web Mobile Email SMS	Node Ruby Java PHP Python Go Scala Clojure	None	SFDC as its part of Heroku ecosyste Any API via programming	- Secure (https) - Endpoint,Integrated with GIT so simple deployment., - Single platform for development & deployment	- Complex - Steep learning curve - Programming skills required.

Table 3: Analysis table for IBM Watson

Channels	Programming Languages	Artificial Intelligenc	Integrates with	Pro's	Con's
Facebook WeChat Telegram Kik Line Kakao	NodeJS Python Java Unity Android IOS	Built in Watson Conversation Other Watson Cognitive Services	Any API via programming	- Minimal number programming required - High Quality of Interaction - Proven solution	- Limited option to integrate various Watson services - Static Dialog

V. CONCLUSION

In this paper, we have presented an energy-aware scheduling system for task-based applications. The scheduler aims at minimizing a normalized bi-objective function. we introduced the chatbot concept and the different cloud platforms to develop chatbot. The developers, messenger apps as well as business become working together and building a new environment, because of the invention of chatbots. The chatbot can be developed for any purpose such as shopping, customer services, food order, news updates, reservation and much more.

All the cloud platform discussed in this paper has different features and function- alities and based on these we have developed some efficient and interactive chatbot and got some results. By using the analysis and results, anyone can choose the cloud platform to build the chatbot. In future research, we will work on how to train the bot using built-in artificial intelligence so that the user will fill like they are taking with another human being. Also, we can think that how the bot will response dynamically without using context, intents or entities.

REFERENCES

[1] J. Koomey, Growth in data center electricity use 2005 to 2010, A report by Analytical Press, completed at the request of The New York Times, 2011, p. 9.

[2] C. Pettey, Gartner estimates ict industry accounts for 2% of global co2 emissions, 14, 2007, p. 2013. Dostupno na: https://www.gartner.com/ newsroom/id/503867.

[3] R. Buyya, A. Beloglazov, J. Abawajy, Energy-efficient management of data center resources for cloud computing: a vision, architectural elements, and open challenges, 2010. arXiv preprint arXiv:1006.0308

[4] A. Miyoshi, C. Lefurgy, E. Van Hensbergen, R. Rajamony, R. Rajkumar, Critical power slope: understanding the runtime effects of frequency scaling, in: Proceedings of the 16th International Conference on Supercomputing, ACM, 2002, pp. 35–44.

[5] V. Tiwari, S. Malik, A. Wolfe, Compilation techniques for low energy: An overview, in: Low Power Electronics, Digest of Technical Papers., IEEE Symposium, IEEE, 1994, pp. 38–39.

[6] A. Beloglazov, J. Abawajy, R. Buyya, Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing, Future Gener. Comput. Syst. 28 (2012) 755–768.