International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 06 | June 2020 www.irjet.net p-ISSN: 2395-0072

An Extensive Study of Infrastructure Automation using Elastic Cloud and Compiler Optimization

Kunal Bhandari[1], Geetha. V[2]

[1]Student, Department of Information Science, RV College of Engineering, Bengaluru, Karnataka, India
[2]Assitant Professor, Department of Information Science, RV College of Engineering, Bengaluru, Karnataka, India

---

Abstract - *In the highly competitive market, companies have many software's running on their distributed systems which are on premise but as demands increases, capital costs spent on infrastructure also skyrockets. This leads to unbalanced use of resources implying resource utilization is not balanced efficiently. People needing resource utility may not get what they need leading to inefficiencies in productivity. The only possible way is to pool resource and provision them virtually on cloud. Provisioning Resources on cloud manually can lead to service outages and huge downtime which can affect business of customers running their critical services for their customers. It also affects reputation for a cloud company. Therefore choosing path to automate the infrastructure provision is the go. It will have no human intervene. The models are automated to provision resources based on the requests of the customers. This will also eliminate any scope of human errors. The manual provision by engineers can cause human burnouts and can annoy them due to performing similar tasks and bring down their work efficiency. The Automation is suitable to avoid these challenges to have continuous integration and deployment. The Automation provide resources in the form of virtual machines (VMs) used to run software for business purpose. The advantage of running it on a virtual machine gives it a dedicated environment to run it without any other dependency and also gets the advantage of running it at desired optimal speed. The software running has an auditing tool to check whether the usage of software is correct. This is analogous to compilers verifying their usage like syntax and semantics. A lot of cloud compilers have been developed and embedded on software's running on cloud but still are in developing phase. Usually compiler are embedded with software for auditing completeness but they tend to take large resources hence optimization of compilers give results with low latency and balanced high resource utilization factor.*

***Key Words*: Cloud Compiler, Infrastructure Automation, Virtual Machine, Infrastructure as a code**

1. INTRODUCTION

The Infrastructure Automation is a powerful step to automate the need of resources by automated scripts generated or run as service created by Infrastructure as a code. The DevOps team have people from both the background of development as wells as operation to work up in sync to run a smooth automation service. It has ability to come up on its own if the service goes down hence not requiring human intervention and the time taken is less than what would have been if manually done. It can self-heal to isolate the outage areas which can be verified by monitoring logs of automation controlled by system engineer. This implementation provides a deliverable software which creates infrastructure in real-time according to the parameters set by user. The advantage of this model is scalability because of high resource utilization factor and it is elastic a user can scale up or down the computing, storage or networking resources which is provisioned from pool of resources. The virtual appliances created by automation is used for business purposes for testing and running traffic to develop a feature. The configurations done on these appliances are verified by a compiler as service running on cloud. The optimization of compiler are with respect to internal data-structures used. It looks on efficient way for internal processing of these data-structures. Since compiler have to sort out many dependencies, this is usually in the form of acyclic graph structure. It has focus on such mechanism which do not take much memory and time, hence it produces intermediate representation in the form of binaries because they can be loaded fast, processed fast, takes less time and are in low-level language which are easily understood by machines. Thus there are many optimizations which are to be discovered. The next section elaborates on these systems end-to-end.

2. LITERATURE SURVEY

The Infrastructure Automation has seen in last couple of years of growing importance of DevOps (Development and Operations), but there is more to this. The Infrastructure as a code service provisions computing data centers through configuration files rather than manually connecting the network hardware.

TOCSA [1] is Topology and Orchestration Specification for Cloud Applications. It is a modelling system based on node system. TOSCA has some nodes which run automation scripts and other nodes which are classified for deployment purpose. There is some nice distinction between the jobs classified to run on. The modelling node system forms a graph system to proceed with correct ordering of automation services satisfying all dependencies. This system is necessarily acyclic tree (graph) forming a topological approach weeding out circular dependencies.

MORE [2] as name suggests is model-driven approach for infrastructure automation for cloud IT services that concentrates on automating the deployment of cloud service and applications. It has the provision to change the configuration file dynamically. It defines a topology which is converted into executable code model to reach a desired state. The dynamic changes when updated re-compiles only updated parts and its dependents and dependencies following recursively eliminating the approach complete recompilation. This also leads to detect changes intelligently and paves way for incremental automation with phased approach.

The Compiler Optimization are based on registry [3] model storing internal values and how they are loaded internally without much load-store operations. The load-store reduced results in optimizations of faster reads and ruling out unnecessary write which results in faster execution upto 3x. These frequent load-store are written to temporary registry where they are accessed and only after final update or commit operation , they are written back to cache or backing store. Examples could be iterators of for loop where looping variable i and other frequent variables in body of the loop are stored temporarily in registry and after the complete loop iterations, variable i may be destroyed and other frequent variables are stored in cache or backing store with dirty bit on to carry out concurrency.

The software signature [4] to the code blocks are assigned to the model to not allow any illegal branching or code execution. It plans to take an efficient data path to execute model plan which results in low latency and optimal use of spatial memory. The intermediate representation are developed for this purpose. Sometimes the executable file can be corrupted with minor changes resulting in illegal code executions or branching. The software signature assigned are internal to code blocks so that even if illegal branching would occur it fails software signature test because it is linked to its correct signature with dependency on previous branch on code block making only correct branch to proceed further with execution and aborting program if illegal branching occurs.

3. STUDY ON AUTOMATION AND COMPILER OPTIMIZATION

3.1 Infrastructure Automation

In [5], authors presented on workflow management of automation infrastructure. The paper talks about the systematic approach to design the workflow of automation modules with the modules in the higher ups having a tree structure and modules in lower parts acting as a child having dependency on the above modules described. It also discusses about techniques of cloud computing to set up different virtual instances by pooling of resources centrally enabling shared tenancy and supporting metered usage. It also sheds light about managing the networking part and maintenance of operations on the cloud and configuring various services on the network as cloud services. The complete workflow is automated, where just initialization of service token automates the complete pipeline of work without having to initialize intermediate actions. It just gives final action of output.

In [6], authors presented on scaling of automation infrastructure. The paper talks about scalability of application services hosted in cloud and methods to remove the bottlenecks. It throws light on deployment of services in Multi-hybrid Cloud Platform. It talks about deployment specifications ready for orchestration at a large scale and how to solve dependencies. It proposes micro-service architecture having minimal dependency solved through endpoints published to interact with other service. We can load only the service we need without running the complete application given that it doesn't have any dependency. The kubernet helps in orchestrating the deployment in order such that satisfies dependencies. Lastly it explains the importance thrown on automation unit testing, integration testing and end-to-end system testing and maintenance of development operations.

In [7], authors presented on inter-cloud applications for enabling services on hybrid cloud. The paper discusses use of 2 state architecture which requires cloud applications on different servers to interact with each other in real-time to service the request. It requires both service to be up and running. The interactions can be one-way or two-way. Only dependencies to be solved is to know the service end points of the other service needed to talk. If those are published, it is the statefull state because it is aware of whom to contact for what, but limiting factor is even other service needs to statefull. If not, it is stateless state, where service is running only for micro-service architecture. This clear distinction between statefull vs stateless is of the binary format representation where service is on (1) or off (0), therefore simplifying representation of services. The log is more classified where we can dig deeper into modules having the desired vs undesired state for debugging any discrepancies.

3.2 Compiler Optimization

In [8], authors presented on analysis on parsing technique and performance on compiler applications. This paper talks about Compiler as the need to design and connectivity between the hardware and software process. Programs written in a high level programming language to be translated to object code before going to be executed. This paper is about brief information of compiler on how the source program gets evaluated and translated, explain the concept of pre-processors, translators, linkers and loaders and procedure to generate target code and from which sections source code has to pass and parse in order to generate target code as output. It specifies a roadmap of what to process in which order aiming to satisfy all constraints and dependencies. The design to load the object

and dynamic changes should be coherent with spatial and temporal memory.

In [9], authors presented on cloud compiler applications. This paper talks about computers becoming limitation when it comes to local installation of compilers and it results mostly in numerous portability and compatibility problems. Compilers and their versions are hugely dependent on the specifications of target computers resulting in difficulties to programmers in compiling such programs. The solution is given in the form of cloud native compiler as a service. The cloud native approach have compilers on cloud with users compiling a program online sending that program unit or folder as input. The compilation takes places on the cloud. The response is sent back as error or as executed output. It deals with optimizations as learnings from machine learning with deep networks through errors and user-defined action and pattern. It talks about compilers on remote networks and architecture for scaling such solutions by orchestrating them on cloud.

In [10], authors presented on online compiler as cloud service and automation. This paper talks about having different compilers and version for compiling program in specific language. This creates problems on storage as well as portability. Moving on these solutions on cloud and orchestrate them can solve problems. The request will be handled based on the server load balancing capacity. The server assigns the special spawned processor based on request to compile that language based program configuration or specification. The request send the response back which would be output after compiling the program. The output may also be error if program has lexical, syntactic or semantic errors. These cloud models are trained on machine learning model to learn from user defined patterns and errors to incorporate such things for compiling it to executable object to so that user gets the understandable output format and message according to the context of the program. These learnings are modelled to learn automatically.

4. CONCLUSIONS

The ever increasing use of infrastructural resources calls for a system to be in place for assigning them, maintaining a queued list and prioritizing such allocation of resources. It is essential that this system should be automated so that it can be efficiently be carried without human prone error but instead to be monitored by humans. It saves time from manual configuration and eradicates long waiting time hence increasing productivity of both developers needing such resources and system automation engineers automating resource allocation. This eliminates manual laborious work which could cause human burnouts. It is essential to allocate resources with priority to get done with important business projects which require essential resources without starving other process. The automated resources also needs to work efficiently internally. The verification of configuration on any virtual appliance can be done by compiler. Based on resource available and overhead involved, these compilers can be on cloud if it is single and large or embedded on the appliance if small and multiple versions exist for different tools.

FUTURE SCOPE

To get all desired features such a availability, reliability, efficiency, confidentiality, integrity etc. is only an ideal situation which we can strive to achieve but in real world only some important objectives (ex:- high confidentiality affects availability and vice-versa) are achieved guided by business rules but cannot compromise on important features such as security. Future goal is to accommodate all features with an efficient working end-to-end system in place.

REFERENCES

[1] Matej Artac, Tadej Borovsak, “Topology and Orchestration Specification for Cloud Applications (TOSCA), DevOps: Introducing Infrastructure-as-Code”, in processings of 2017 IEEE/ACM 39th IEEE International Conference on Software Engineering Companion.

[2] N. Ferry, A. Rossini, F. Chauvel, B. Morin, and A. Solberg, “Towards Model-Driven Provisioning, Deployment, Monitoring, and Adaptation of Multi-cloud Systems”, in proceeding of Sixth IEEE International Conference on Cloud Computing, pp. 887-894, 2013.

[3] Md. Alomgir Hossain, Rihab Rahman , Md. Hasibul Islam, Mahabub Azam,” A Study on Language Processing Policies in Compiler Design”, American Journal of Engineering Research (AJER), 2019.

[4] Christopher Monsanto, Nate Foster, Rob Harrison, David Walker,” A Compiler and Run-time System for Network Programming Languages”, European Journal of Networking Research (AJER), 2017.

[5] Dimitrios Georgakopoulos and MarkHornick, “An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure “, in proceedings of Kluwer Academic Publishers, Boston, Volume 3, pp 119-153, 2015.

[6] Sushil Deshmukh , Sweta Kale, "Automatic Scaling Of Web Applications For Cloud Computing Services: A Review", in proceedings of International Journal of Research in Engineering and Technology, Vol. 03, pp. 2321-7308, 2014.

[7] N. Asthana, T. Chefalas, A. Karve, A. Segal, M. Dubey and S. Zeng, "A declarative approach for service enablement on hybrid cloud orchestration engines," in proceedings of IEEE/IFIP Network Operations and Management Symposium, Taipei, pp. 1-7, 2018

[8] C.h. Raju, Thirupathi Marupaka, Arvind Tudigani "Analysis of Parsing Techniques & Survey on Compiler Applications", International Journal of Computer Science and Mobile Computing, 2013.

[9] S C Suryawanshi, Akshay Bankar , Akshay Agrawal , Aneesh Ashtikar, Pranesh Meher, "Cloud Compiler techniques", International Journal of Advanced Research in Computer and Communication Engineering,2017.

[10] Arnab Paul, Arjun Datta, "Online Compiler as cloud service and automation", International Conference on Advance Communication Control and Computing Technologies (ICACCCT), 2014.