RESEARCHERID
THOMSON REUTERS

[COTII- 2018]

ISSN 2348 – 8034
Impact Factor- 5.070

GLOBAL JOURNAL OF ENGINEERING SCIENCE AND RESEARCHES

EVENT DIMINISHMENT APPROACH FOR CLOUD ARCHITECTURE

Ashish Kumar Trivedi[1], Rajiv Pandey[2] & P. K. Bharti[3]
[1]Research Scholar, MUIT, Lucknow, India
[2] Amity University, Lucknow, India
[3]MUIT, Lucknow, India

ABSTRACT

As of late distributed computing is advancing as a key processing stage for sharing assets. It guarantees that assets like registering limit and capacity or administrations like databases or informing framework can quickly be obtained and discharged in light of the present necessities of the application. Expansive scale, shared IT framework accessible over the web, is changing the way corporate IT administrations are conveyed and overseen. Distributed computing is an exceedingly adaptable and financially savvy framework for running HPC, venture and Web applications. So as to plan arrangements in Cloud Computing, profound examination of Cloud is required as for their energy productivity. In this propose work we break down and engineer distributed computing on a few Key elements which empower to bring down the impact of distributed computing in the shape vitality utilize and carbon discharge. To address the above said challenges, we will build up a proficient and compelling Event decreasing model for Cloud Architecture that prompts an answer of the past difficulties. In this model we utilize Event diminishment approaches, similar to occasion grouping, to lessen the excess and copy occasions from private cloud and afterward give these occasions to the Event Orchestration System to get to administrations from open cloud. Accordingly benefits accessible to the clients in a productive and powerful way with least load on servers and server farms. Here in this work we propose a model that decreases Event in cloud engineering. In this model we utilize Event Clustering to decrease the repetitive and copy occasions originating from private mists and after that forward these occasions to the Event Orchestration System whose duty is to find benefits in the general population cloud, encourage these administrations at client level. Thus benefit accessibility to the client is ensured in a proficient and compelling way.

Keywords: *HPC, Event reducing, Event Clustering, public cloud, Private Cloud, Orchestration engine*.

I. INTRODUCTION

Recently cloud computing is advancing as a key computing platform Use of Enterprise Applications (EA) is extremely prominent among organizations in light of the fact that are ended up being a decent worldview for correspondence and interoperability between applications. These Applications give consistency of data all through associations. Improvement of EA depends on Enterprise processing and developed hugely in finished the most recent couple of years. As an ever increasing number of associations with a bunch of utilization s turn into, the capacity to develop and coordinate existing applications ends up critical. The web benefit engineering with its open measures gives a promising method to coordinate divergent applications. The Service Oriented Architecture (SOA) and Web Service approach have been used for integration of Enterprise Applications but this approach, introduces another level of complexity; i.e., the services have to be complete, readily available and cannot be changed once in use. The real power of the services model appears when Event Driven Architecture (EDA) is combined with orchestration. Orchestration allows a configuration to define where the services are, how and when they are consumed, other services that are executed, and how data are combined. The orchestration engine calls a service and passes the context to it. Once the service has been completed, the context is handed back to the orchestration engine that executes the process. The orchestration engine is commanding services to execute and control the overarching flow [6].

Development of applications by using orchestrations to combine services provides several benefits. Application development with higher level languages allows developers to combine existing services into a process. While using

orchestrations to call independent services over the Internet has been very successful, but its use in EA is very problematic. Performance, control, life span, access control and several other issues need to be solved. Using services in EAs is much more complex and introduces challenges for developer [6].

There are several challenges like Dependencies of services, response time, linking of implementation and services and rigidity of services exist in Enterprise Applications. To address the above said challenges, in this paper we proposed an efficient and effective Event Driven Service Oriented Architecture based on Cloud Computing that leads to an answer of the previous challenges. In this architecture we use Event Clustering to reduce the redundant and duplicate events from private cloud and then give these events to the Event Orchestration System to access services from public cloud. As a result services available to the user s in an efficient and effective manner.

II. BACK GROUND

Web services are self contained, self describing, modular applications that can be published, located, and invoked across the web. Web services perform functions, which can be anything from simple requests to complicated business processes Web services are proving to be a good paradigm for communication and interoperability between applications. Web services are platform-independent, low-cost, performing and – perhaps most important - correct interfaces for humans, who enact complex application workflows communicating with Web services. To develop Web applications, several research groups suggests the use of declarative design languages ([1], [4] …), by shifting focus from source code level development to abstract and platform-independent models, to enable fast prototyping, automatic code and documentation generation, and model-level correctness checking. [5].

Service Oriented Architecture (SOA) is an architectural style for organizing and utilizing distributed capabilities that may be under the control of different ownership domains and implemented using various technology stacks. In Service Oriented Architectures (SOA), an Enterprise's Architecture is developed in a Service Driven approach, where each service is autonomous. Loosely coupled Services are orchestrated into business processes that support customer goals and/or organization's business goals. More than just a component of reusable code, a service is considered as a part of a running program that can be invoked by a client without having to incorporate the code itself. A service by definition is reusable and replaceable, that is, service is reused again and again by other services for the functionality it provides and service provider implementation can be replaced by another provider's implementation. In SOA the services can be categorized into basic/foundation services, management services, security services, business services, portal services, etc. It should also be noted that a service is offering a specific functionality for an Enterprise and transcends projects; service is only implemented once in enterprise architecture and can be reused across projects that deal with delivering content, services, multi-media, etc. In SOA the communication flow is closed to new unforeseen inputs once the communication flow has started, i.e., they are typically well defined and the boundaries well set. [7]

Use of SOA has various complexities i.e., the services have to be complete, readily available and cannot be changed once in use. To overcome these problems events driven services are introduced. Since events are huge, asynchronous, and unpredictable. Therefore **Event Driven Architecture (EDA)** emerges to create efficient and agile applications and it embraces mechanisms for coordinating the callers and providers of service, producers and consumers of data, sensors and responders of software events with variable level of communication coupling, with variable spectrum of message correlation and with variable options to deliver quality of service. EDA engenders a network that listens to thousands of incoming events from different sources, wherein complex event processing results in intended system response. EDA supports dynamic, parallel, asynchronous flows of messages and hence reacts to external inputs that can be unpredictable in nature. An EDA can coordinate synchronously or asynchronously between software endpoints, and possibly provide both synchronous and asynchronous access between the same participants. In EDA, simultaneous streams of execution can run independently of each other to fulfill a customer request or system responsibility, typically an event bus serves as a platform to manage integration and/or choreograph a larger process [7].

In EDA an event is actually service request and is used to control the components which act autonomous, and are fully decoupled. The basis of event processing is that events are sent via middleware application to all the users who act on it. The other components decide themselves which type of event they will subscribe or on which they will respond [8].

The use of SOA in building EAs was supposed to solve the problems of dependency and soft-ware complexity. Services are the driving force for better organized codes. Integration becomes a very simple task with access to functionality readily available. Therefore applications could deliver substantial benefits by leveraging process management tools and the internal application code. Web Services, work very well in the distributed environment of the Internet where the amount of data to be transferred is relatively small and users are willing to wait a second or two for a response, and where the services have a very clear function and purpose. Using services in EAs is much more complex and introduces challenges for developers [6].

Components are used to overcome these challenges of SOA. Component-based software development provides a variety of advantages; it manages complexity more effectively, reduces development time, produces better quality software, and increases reusability. **Component Based Architecture (CBA),** the basis for distributed component architecture which binds mechanisms and techniques for developing coarse components that is environment/container aware, which decomposes a service, into multiple pluggable and distributable parts associated with presentation logic, business logic, resource access logic, integration logic, network event logic, security logic and more. Basis of a component based approach evolves are object oriented design principles (encapsulation, polymorphism, inheritance, etc.), and gives a foundation set of development and infrastructure technologies (based on J2EE/J2ME/JAIN or .NET/OSA/Parlay) for seamless integration. Component Based Architecture (CBA) fundamentally modularizes monolithic systems into multiple coarse-grained, durable and reusable units that could be implemented by multiple vendors yet integrated into larger enterprise applications. CBA typically engenders designing the internals of these coarse-grained, business-aligned component boundaries, through finer-grained object orientation. [7].

Cloud Computing is a most promising and emerging paradigm to overcome the challenges of SOA. Cloud Computing holds first place in Gartners's top 10 strategic technologies list[12].It promises that resources like computing capacity and storage or services like databases or messaging systems can rapidly be acquired and released based on the current requirement s of deployed application. Basis of cloud computing is a combination of several well-known technologies like virtualization and concepts like service oriented architecture [8].

Cloud computing describes both a platform and type of application .A cloud computing platform dynamically provisions configures, reconfigures and de-provision servers as needed. Servers in the cloud can be physical machines or virtual machines. Advanced cloud typically includes other computing resources such as storage area network (SAN), network equipments, firewalls and other security devices. [10]

The cloud applications can be accessible through the internet. It exploits large data centers and powerful servers that host web application and web services. Anyone can access a cloud application by using suitable internet connection and a standard web browser [10].

A cloud can be viewed as a pool of virtualized computer resources, it can-

- Host a variety of different workloads including batch style back end jobs and interactive user facing applications.
- Allows workloads to be deployed and scaled out quickly through the rapid provisioning of virtual machines or physical machines.
- Support redundant self recovering, highly scalable programming models that allow workloads to recover from much unavoidable hardware/software failure.
- Monitor resource use in real time to enable rebalancing of allocations when needed [10].

ISSN 2348 – 8034
Impact Factor- 5.070

A cloud is considered as a collection of computer resources. A cloud provides a mechanism to manage these resources; management includes provisioning, change request, workload rebalancing, de provisioning and monitoring [9].

III. METHODOLOGY

Architecture
The proposed architecture (Fig. 1) comprises of following components

User Interface
With user interface a user can interact with the system and get response.

Private Cloud
A private cloud is a cloud dedicated to specific purpose/person. A private cloud can do following-

- Publish user event asynchronously
- Provides user Centricity
- Follow and Remember user interest
- Keep user data
- Initially invoke events of various nature as required
- Forward user events to event cluster engine
- Provides local data storage
- Provides security

Event Cluster Station
The event cluster station is responsible for following

- It preliminary identify nature of events
- Grouping of similar events
- Replacing redundant occurrence
- Finally reduce the event domain

Event Orchestration Engine
The Orchestration engine is responsible for following

- Allows a configuration to define where the services are(locating services)
- How and when services are consumed
- Other services that are executed
- How data are combined(Service integration)
- The orchestration engine calls a service and passes the context to it.
- Once the service has been completed, the context is handed back to the orchestration engine that executes the process. This pattern has been called "command and control". The orchestration engine is commanding services to execute and controls the overarching flow.

Public Cloud
The public cloud can be viewed as service repository and is responsible for following

- External data centers
- Provide services of different nature
- On request call to other cloud
- Can scale inclemently on demand and provide services as pay per use basis

ISSN 2348 – 8034
Impact Factor- 5.070

IV. OUR APPROACH

In an enterprise level application massive number of users interact simultaneously and generate events of their interest, these events are actually service request, the events are invoked through private clouds and passed to event clustering system which reduces the occurrence of redundant event, group events of similar nature and produce a reduced domain of events which is then passed to Event Orchestration Engine. The Event Orchestration Engine allows a configuration to define where the services are, how and when they are consumed, other services that are executed, and how data are combined. The orchestration engine calls a service and passes the context to it. Once the service has been completed, the context is handed back to the orchestration engine that executes the process. This pattern has been called "command and control". The orchestration engine is commanding services to exe-cute and controls the overarching flow.

The public cloud can be thought as service repository ,it receives call to various services from Event Orchestration engine, provide services on pay per use basis, it can scale incrementally as and when required, can call to other clouds, act as are source provisioning system.

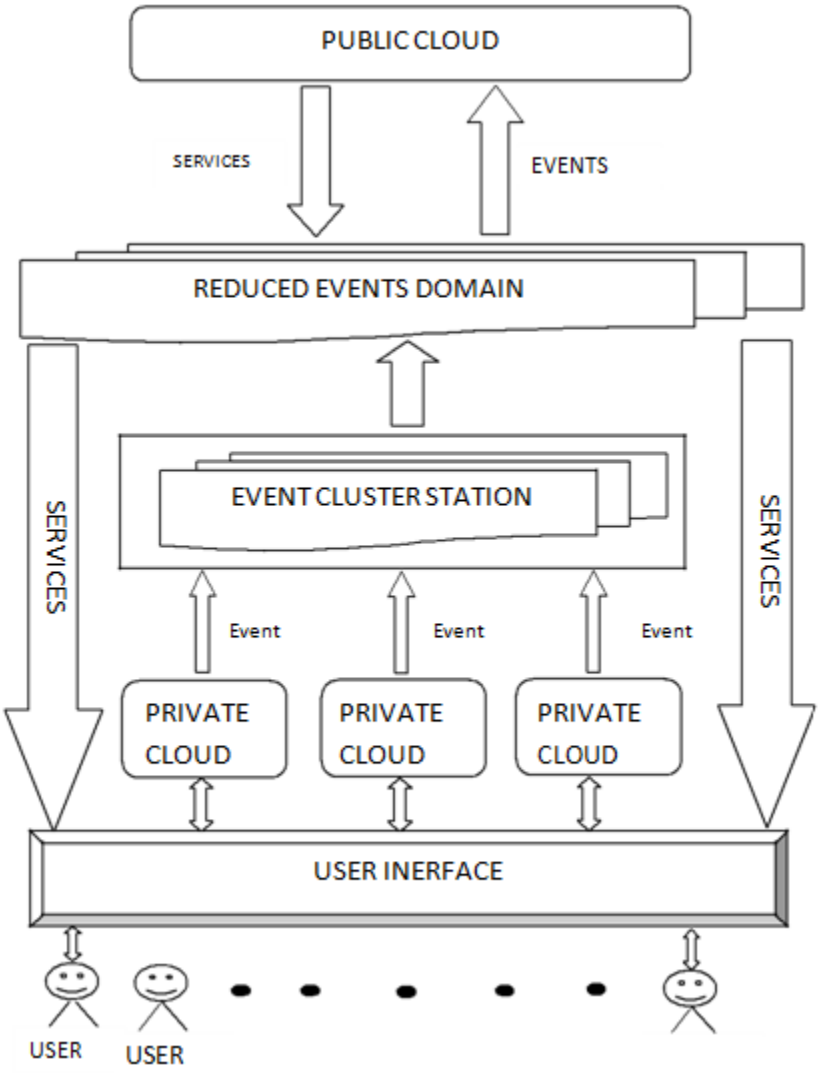

Fig 1: Architectural View

Event Clustering

Clustering is the process of grouping together of similar data items. Further clusters should reflect some mechanism at work in the domain from which instances or data points are drawn, a mechanism that causes some instances to bear a stronger resemblance to one another than they do to the remaining instances.

Let X ϵ R mxn a set of data items representing a set of m points xi in Rn. The goal is to partition X into K groups Ck such every data that belong to the same group are more alike than data in different groups. Each of the K groups is called a cluster. The result is an injective mapping X → C of data items Xi to clusters Ck.

For the efficient processing of massive amount of content each gathered piece of content represent an event that will be processed via a event processing network to correlate the new content with the relevant piece of information. This pre-processing of content reduces the overall amount to be processed by filtering out unusable or duplicate events and by deducting information from the events to create so called complex events representing the interlinking of several related raw events. This will reduce the amount of events that need to be processed.

In our approach we use events as data items. To reduce the redundant events, they are grouped into clusters to form event clusters. Event processing systems uses these event clusters to reduce the redundant events. The event processing system as a whole is distributed across the boundaries of system. In the first stage of event processing, agents that handle the massive amount of raw events are deployed in multiple nodes to realize the initial filtering and basic pre-processing. Event clusters are used to reduce the duplicate and redundant events by reducing the domains of events. Once the first stage is accomplished these nodes handover the reduced stream of events to Orchestration Engine which is again located in cloud environment.

V. CONCLUSION

This architecture uses Event driven service oriented approach based on cloud computing. In this architecture we use Event Clustering to reduce the redundant and duplicate events coming from private clouds and then forward these events to the Event Orchestration System whose responsibility is to locate services in the public cloud, facilitate these services at user level. Due to event clustering mechanism the domain of massive and redundant events reduced thus decreasing the overhead of Event Orchestration engine and improving the performance of the system. As a result service availability to the user is guaranteed in an efficient and effective manner.

REFERENCES

1. *Baresi, L., Garzotto, F., Paolini, P.: From Web Sites to Web Applications: New Issues for Conceptual Modeling. ERWorkshops 2000: 89-100.*
2. *Brambilla, M., Ceri, S., Comai, S., Fraternali, P.,Manolescu, I., Specification and design of workflow-drivenhypertexts, Journal of Web Engineering, 1(2) April, 2003.*
3. *Ceri, S., Fraternali, P., Bongio, A.: Web Modeling Language (WebML): a modeling language for designing Web sites.WWW9/Computer Networks 33(1-6): 137-157 (2000).*
4. *Ceri, S., Fraternali, P., Acerbis, R., Bongio, A., et al.:Architectural Issues and Solutions in the Development ofData-Intensive Web Applications, CIDR2003, Asilomar.*
5. *Brambilla, M., Ceri, S., Fraternali, P., di Milano, P., Acerbis, R., Bongio A., Model-driven Design of Service-enabled Web Applications.*
6. *Karina Hauser, Helgi S. Sigurdsson, Katherine M. Chudoba, EDSOA: An Event-Driven Service-Oriented Architecture Model For Enterprise Applications, International Journal of Management & Information Systems – Third Quarter 2010 Volume 14, Number 3 37*
7. *Badri Sriraman Lead IT Architect (Unisys), Rakesh Radhakrishnan Enterprise IT Architect (Sun Microsystems), Component Based Architecture Supplementing Service Oriented Architectures, March, 2005*
8. *Stella Gatzui grivas, Marc Schaaf, Michael Kaschesky, Guillaume Bouchard. Cloud-based Event-processing Architecture for Opinion Minning, 2011 IEEE World Congress on Services*

9. *Umar Khalid Farooqui ,Mohammad Hussain , Ashish Kumar Trivedi ,Architecting Distributed Domain Reducer in Cloud Environment ,International Journal of Computer Applications (0975 – 8887) Volume 40– No.12, February 2012.*
10. *Greg Boss, Padma Malladi, Dennis Quan, Linda Legregni, Harold Hall, "cloud computing", 15 Feb-2011. http://download.boulder.ibm.com/ibmdl/pub/software/dw/we s/hipods/Cloud_computing_wp_final8Oct.pdf.*
11. *Ceri, S., Fraternali, P., Bongio, A., Brambilla, M., Comai, S., Matera, M.: Designing Data-Intensive Web Applications,Morgan-Kaufmann, December 2002.*
12. *Fernandez, M. F., Florescu, D., Kang, J., Levy, A.Y., Suciu, D.: Catching the Boat with Strudel: Experiences with a Web-Site Management System. SIGMOD 1998: 414-425.*
13. *Gómez, J., Cachero, C., Pastor, O.: Conceptual Modeling of Device-Independent Web Applications. IEEE MultiMedia8(2): 26-39 (2001)*
14. *Manolescu, I., Brambilla, M., Ceri, S., Comai, S., Fraternali, P.: Model-Driven Design and Deployment of Service-Enabled Web Applications, TOIT, Volume 5, number 2 (May 2005).*
15. *Schwabe, D., Rossi, G.: An Object Oriented Approach to Web Applications Design. TAPOS 4(4): (1998).*
16. *Conallen, J., Building Web Applications with UML. Addison Wesley (Object Technology Series), 2000*
17. *Oleg y. Nickolayev, philip c. Roth and daniel a. Reed,"real-time statistical clustering for event trace reduction"*
18. *YongChul Kwon, Wing Yee Lee, Magdalena Balazinska "Clustering Events on Streams using Complex Context Information.", Data Mining Workshops, 2008. ICDMW '08. IEEE International Conference Dec 2008.*
19. *Rajender Kumar Trivedi, Rajani Sharma," Case Study on Environmental Impact of Cloud Computing", OSR Journal of Computer Engineering (IOSR-JCE) e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 16, Issue 2, Ver. VI (Mar-Apr. 2014), PP 81-86 www.iosrjournals.org*
20. *Muhammad Arif, Tariq Mahmood, "Cloud Computing and its Environmental Effects", International Journal of Grid Distribution Computing Vol.8, No.1 (2015), pp.279-286 http://dx.doi.org/10.14257/ijgdc.2015.8.1.26.*