International Journal of Advance Research in Science and Engineering
Vol. No.6, Issue No. 07, July 2017
www.ijarse.com

IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346

Fluid computing: Principles, Applications, Future Directions

Manas Kumar Yogi[1], Lakkamsani Yamuna[2], K.Chandrasekhar[3]

[1,2,3]Dept. of CSE, Pragati Engineering College (Autonomous), Surampalem, A.P.(India)

ABSTRACT

Pervasive computing is the next level of computing in modern technological era where user is much bothered about the flexibility of working on an application without having to bother about the nature of devices. The user may change the devices according to his portable needs and the handoff of the application state must be near perfect. To achieve this near perfect synchronization the concept of fluid computing and its implementation resulting into a robust middle architecture forms the main crux of our paper. We present the basic fundamentals needed to under the fluid computing concept and its applications which are already sparking off an revolution in wireless technological world.

***Keywords:* Batch mode , Fluid Computing, Handoff, Trickle mode, Synchronization**

I. INTRODUCTION

As the advent of distributed systems brings about never imagined advancements still mankind hopes for a better technology at its dispense. Few years back cloud computing paradigm had taken the whole world by storm with its infinite ability it provides various services on demand. With its boon for virtualization a well secured architecture was put into practice. the essential properties of cloud namely on-demand self-service, broad network access, resources-pooling, rapid elasticity, measured services more talking point of every potential cloud user But due to security issues while cloud data storage a shift to new technology paradigm called as “fog computing; came into existence. In fog computing also known as edge computing, proximity of data to end users is more .It is medium in terms of computing power. as said the inherent scope of development still exists which has given birth to mist computing which refers to a lightweight and primitive form of computing power which resides directly inside the network fabric consisting of microcomputer and micro controllers to feed into fog nodes thus the fog nodes are responsible to inject the same forward cloud computing platform. Mist, fog computing are emerging as contenders against cloud in terms of connectivity bandwidth, latency, cost & security challenges imposed by cloud architectures. So far everything was seemingly alright but human nature of operational simplicity forced, towards a principle of distributed computing architecture where design was to achieve most cost effective, reliable, scalable architecture which was never possible before, The operating technology was coined a term “fluid” and fluid computing become famous from last couple of years.

II. LITERATURE SURVEY

2.1 Chisel

Adaptable software for varying contexts was a challenge met by researches when they produced a dynamic framework named chisel[1] .chisel worked on the principle of adaptation of changing work execution environmental taking inputs from low-level or high level knowledge sources. The need for chisel came into existence due to change in users requirements in terms of resources in highly unpredictable manner. Chisel had policy based control mechanism. Chisel had control for adaptation –Manager in distributed nature .It resulted in more flexible & scalable behavior of chisel .It had provisions for user to supply more information to guide the adaptation manager for leveraging the adaptation behavior chisel had specification documents containing the policies which were a form of declarative representations of policy rules. These policy rules were simple to handle for users as well as programmers. It also composed of service objects which provided multiple non-functional behaviors .Iguana/jmetabyte[2] that can be statically dynamically connected as well as disconnected with service objects was considered for this task at any time if policy had to be changed it could be done in chisel due to its flexible architecture. The overall control was in hands of a meta level adaptation manager which could interpret policy to be adopted s per changes made in the context chisel had yet another notable component named context manager context managers to reset a relevant rule, policy manager was primarily responsible to track any changes made in the policy declaration file the policy language had two different part the first part corresponds to the adaptation rule .the second part represents the definition of new events dynamically in mobile computing chisel has been actively used to implement context aware applications. It has been concluded by recent survey reports the chisel has reached the high expectations from users of mobile network community in terms of efficiency & performance.

2.2. Eye Dentify

It’s a smart phone application which is deployed on the top of a java based high performance distributed middleware named IBIS. The main objects is to implement cyber foraging in cyber foraging users can take help for computing by resources provided by surrogates which are part of the distributed server application. The user need not worry about low computing power of the client mobile and depends on mobile in the vicinity to perform efficient functionality eyedentify application can perform object recognition using the camera sensor of the smart phone to offload the computational tasks while doing so the issues faced are responsiveness energy usage & accuracy. Apart from these challenges it uses lightweight communication protocol and should efficiently handle time intervals of no connectivity and no sensors access .the main component of the ibis distributed deployment system is the java Grid application toolkit(javaGAT).This toolkit included library files for functionalities like remote file management, remote job submission, monitoring and steering. The design is so much flexible that using the middleware adapter it can bind to any API. On android any remote application on any machine can be activated using SSH. On top of this java GAT the graphical user interface (GUI) is used to deploy the Ibis applications. The deployment is a series of subtasks which starts with the activity of copying the application libraries inputs files to determine needed resources.

Then the Ibis server is started to form an overlay network .Further, middleware specific job description are constructed then the jobs statuses are developed along with jobs development when the jobs are completed output files are retrieved final task consist of cleaning u the remote file system for programming ibis application the system has ibis portability

Layer (IPL)[3].It is a powerful & efficient primitives for communication. The IPL ports have robust support for one to one ,one to many, many to many connections. For each port the most efficient communication implementation which satisfies the user requirement can be chosen by the programmer. The notable observation in this scenario is that efficient object serialization can be done for fault tolerance as well as dynamic addition & deletion of computational resources .eyedentify works in two operational modes namely the learning mode recognition mode. In learning mode, user enters the object its name in internal database. In recognition mode, the user inputs objects image under different viewpoint and lighting condition & eyedentify outputs best match from the database of learned objects. The computing multimedia library provides the object recognition algorithm for eyedentify. The object recognition algorithm works as follows. It creates receptive fields around the image centre and builds a color histogram for each different color model .The output parameters from all histograms are integrated into a single feature vector resulting into a condensed description of the image scene object is recognized by determining the closest neighboring point in a high dimensional space. Observational records on implantation eyedentify have shown that ibis provides easy and powerful multimedia cyber foraging tasks for smart phones[4].

2.3. TOTA (Tuples on the air)

Used as an efficient middleware support in pervasive mobile computing environmental .Here spatially distributed tuples are propagated across a network on basis of application specific rules[5] .This rules denote the inherent contextual data along with uncoupled interaction across application components. The tuples in this application are not specific to a node of networks. But they are diffused in a network to propagate as patterns. These tuples have ability to transmit messages across the network each node save addresses to limited set of neighboring nodes In case of new nodes attached in network or failure in any node it results in dynamic change in the topology.GPS like localization mechanism & RADAR like mechanism find the location of an agent with respect to another agent maintenance rules depends on localization information by verifying the tuple propagation rule check tuples may or may not propagate. The main components of TOTA middleware are TOTA API which is responsible for providing a transparent interface between the application agents the middleware[6]. Primary concern is injection of new tuples, retrieval of tuples, placement of tuple related and network related subscriptions in the event interface .The second component is EVENT INTERFACE which controls the event arises due to subscribed events, final component is the TOTA ENGINE which maintains the TOTA network by opening closing the communication sockets to interchange tuple ,tuples ids are generated by adding a unique number to the MAC address along with a progressive counter for tuples diffused in the node. The hash based tuples ids are quick to access the scheme of the tuples. The researchers have developed a java based emulator to put into action the functionality of TOTA in a MANET scenario .It allows adaptation of interface between emulator the application layers so that its highly beneficial to test applications first in emulator and then upload then directly into a network of real components. TOTA emulator are integrated successfully with third party simulators in diverse domain like video games urban traffic control robotics.

III. PRINCIPLES

3.1 Design issues with fluid computing

a. Operational consistency: In fluid computing architecture an application should be able to flow from one application to another based on users situation & ability of the interface being used, thus we say that application state flows as a fluid between devices. The main issue is means to handle the state of operation with design of synchronization- middleware. The middleware should be able to replicate data on many devices and this
Replication state should operate autonomously across all the devices involved.

b. Synchronization protocol: It's the set of rules which can keep the replicas consistent, based on quality of the network connectivity available. Hence it is obvious that high quality network bandwidth will be a safe design necessity to maintain the replica consistency.

There are 2 operation modes for the fog computing synchronization protocol:

Model 1: Batch Mode: In this operational mode ,incase
connectivity is lost and then gained after sometime, all updates accumulated during the disconnection are exchanged between the devices. This exchange helps in maintenance of consistent state among all devices which host the application.

Model 2: Trickle mode :In this mode replication happens as long as there is some connectivity, it spreads real-time updates as soon as generation of state occurs at their respective replicas.

c. Reliability: It's improved by using hypervisor redundancy usage of hypervisor redundancy places need of only one virtual appliance such that complexity. Configuration of redundancy of devices is removed due to downtime. Some implementations of fluid computing is deployed on the top of RAID cloud architecture.

d. Scalability: Virtual appliance configuration helps improve the scalability by simply changing it which is quick too. The above stated design issues are taken into active consideration to develop the fluid computing architecture.

3.2 The fluid middleware architecture is as follows

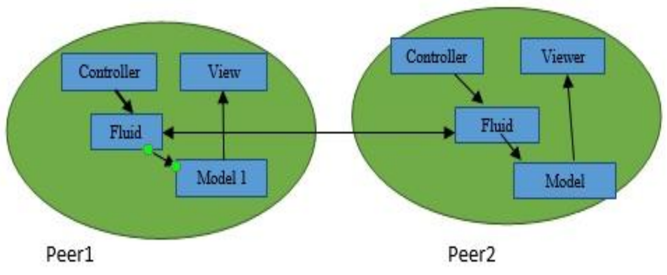

Figure 2.Fluid middleware architecture

As discussed in design issues the fluid middleware should ideally consist of essential features like auto transaction between connection & disconnection, different connectivity degrees should be supported, diverse devices should besupported, and complex data structures should be supported. In above figure the model entity stores the state in a suitable data structure together with application logic and facilitates an interface to perform changes also. View element is responsible for displaying information present in the model and controller accepts input from the user, understands it and calls the suitable method of model.

Fluid is represented as a thin layer which denotes itself between the model & MVC controller. Controller can modify the model. Model is responsible for registering the views and they can be changed with help of events which are sent by the model after it performs modifications.

The terms associated with fluid middleware are as below:

a. Peer: device which is executing an instance of the fluid middleware.

b. Fluid session: Certain state corresponding to the temporary association of users, devices, application.

e. Model: ADT structuring the state application.

f. Model instance: Information which allows application state specific instance which can be recreated in a peer.

As shown in fig above, two peer devices peer1, peer2 create fluid session between each other. They will hare same logical Model instance & each will contain a local replica representing that logic model. If peer1 does any changes to the model, it's got hold by fluid instance 1, which does local modification and same time informs the fluid instance 2 of the update. Peer 2 when receiving the update applies same modifications to the local replica, consequently synchronizing both the replicas of peer1, peer2.

3.3 Inherent advantages of proposed architecture

The crucial benefit incurred is of complete transparency with respect to the single entities of a MVC pattern. Practically speaking the presence of the Fluid-instance offers the programmer the same API which is used in non-Fluid case. Moreover this middle has the flexibility to share API's too among the peers. Sometimes the user in one device may have difficulty in few functionalities due to unsupported API's forcing them to shift to other devices .The shared API concept is a very difficult operationality due to limitations of the language construct but nevertheless researchers in this area are working towards it. The main challenge is not technical in this API sharing principle but has business aspects attached to it. Few companies do not want to share the intrinsic features of the API which may leverage the performance of the devices if used by their competitive organizations working in the area of fluid computing.

IV. APPLICATION OF FLUID COMPUTING

In current technological market apple has incorporated in ios8 and OS XY emits updates the feature of fluid computing. Both are in beta stage as shown by researchers in WWDC conference. Google Microsoft are also pursing fluid computing. Citix Meta frame allows certain degree of fluidity by allowing users to move their session among devices but it has some inherent hardware related issues. Notwithstanding such challenges Blackberry has similar functionality name blend for its smartphones so as to send calls(or) text messages to windows & mac based OS liquid computing in these applications allow handoffs many times .In 2015,Nokia corporation presented its wonderful liquid computing application named cloud based radio-architecture at the mobile world congress. This application has ability to leverage the wireless network operability. In the demo it was show how proximity and location can connect subscribers and local points of interest for business purposes. For instance a base station which is located near a historic landmark can have ability to store data about it and subscribers of that devices will automatically receive a sms when they are passing near that landmark which boost the level of tourism in that geographical area. Other notable mentions may be using fluid applications for collaborative application design by freelancers who may work shared fluid machine and save appreciable amount of cost in developing an application. Also for testing purposes, fluid machines boot quickly and spin up a test cluster within few seconds.

IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346

V. CONCLUSION

In this paper we have presented the novel idea of flowing of the application state from device to device thereby creating a process of fluid among the devices. The main benefit is user can continue doing the work interruptible without bothering about the devices. Process synchronization is done by the middleware of the fluid computing architecture. Future work in this area is due to upcoming needs .This needs can be classified as reducing effects of variable connectivity and multi person application collaboration. Active research is going on in this area of pervasive computing so as to give the users maximum benefit of usability, security, robustness .All the major technological giants have started making steady progress in this area. Security in distributed computing is ever than challenging as before so distributed computing security features workable with the paradigm of fluid computing needs to be considered seriously. As said fluid computing, if sticks around will change the idea of how distributed applications are built .In our opinion ,both users and developers of fluid applications should work together in a developmental mode which actively considers relevance feedback from users while prototyping fluid applications. We sincerely conclude this paper stating that this computing is here to stay as well as flow.

REFERENCES

[1]. Daniela Bourges-Waldegg, Yann Duponchel, Marcel Graf, and Michael Moser. ”The fluid computing middleware: Bringing application fluidity to the mobile internet”. In SAINT ’05: Proceedings of the 2005 Symposium on Applications and the Internet (SAINT’05), pages 54–63, Washington, DC, USA, 2005. IEEE Computer Society.

[2]. Glenn E. Krasner and Stephen T. Pope. “A cookbook for using themodel-view controller user interface paradigm in smalltalk-“80. J. Object Oriented Program. , 1(3):26–49, 1988.

[3]. K. Petersen, M.J. Spreitzer, D.B. Terry, M.M. Theimer, A.J. Demers, "Flexible Update Propagation for Weakly Consistent explication", Proceedings of the 16th Symposium on Operating Systems Principles, October 1997.

[4]. D. B. Terry, M. M. Theimer, K. Petersen, A. J. Demers, M. J. Spreitzer, C. H. Hauser, "Managing Update Conflicts in Bayou a Weakly Connected Replicated Storage System", Proceedings of the 15th Symposium on Operating Systems Principles Copper Mountain Resort, December 1995.

[5]. E. Gamma, R. Helm, R. Johnson, J. Vlissides, "Design Patterns: Elements of Reuseable Object Oriented Software" in Addison Wesley, 1994.

[6]. M. Boulkenafed, V. Issarny, "A Middleware Service for Mobile Ad Hoc Data Sharing Enhancing Data Availability", Proceedings of Middleware 2003, June 2003.