ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 7, July 2014

A LIGHTWEIGHT MESSAGING QUEUE IN CLOUD

Rupinder Kaur[1], Sahil Vashist[2]

Student, Computer Science, Chandigarh Engineering College, Landran, Mohali, India [1]

Assitant Professor, Computer Science, Chandigarh Engineering College, Landran, Mohali, India [2]

Abstract: In cloud computing environments guarantees, consistency mechanisms, state and transactions are frequently listed for scalability, measurability and performance. Based on this challenge we tend to presents CMQ, a UDP-based inherently asynchronous message queue to orchestrate messages, events and processes within the cloud. CMQ's light-weight edge-to-edge style, that is somewhat almost like Unix Pipes, makes it terribly composable. By presenting our work, we tend to hope to initiate discussion on a way to implement lightweight electronic communication paradigms that are aligned with the general architectures and goals of cloud computing.

Keyword: Cloud Paradigm, CMQ, UDP Protocol, Message-Oriented-Middleware

I.INTRODUCTION

Cloud computing has 2 perspectives: initially, an outwardlooking perspective that embodies Associate in embodies an elastic application executed in an exceedingly secure instrumentation and accessible over the internet, as seen by developers and finish users, secondly, an inward looking perspective that describes the massive scale distributed cloud computing platform and its middleware as enforced and operated by the supplier [1]. CMQ presents a message passing model (that could be a middleware abstraction) enforced in Haskell that addresses the reality of each views. The inward-looking perspective

The physical cloud nodes in computing clouds are organized into "Points of Delivery" and interconnected via equipment that either switches at line rate, or that uses lossless local area network fabric technologies. In switched information center networks, all local area network (RFC 894) based mostly networking protocols are switched while not discrimination. Congestion and packet loss are very unlikely in such information center networks However, each protocols need specialised hardware such as network cards, change gear that's not in line with the trend to make clouds from artifact hardware [2], and accept occasional failures instead of preventing failure at any price [3]. In CMQ, UDP is employed because the transport protocol for the following reasons:

• it's connectionless.

• it's the protocol that adds the smallest amount overhead to the ethernet network, it adds solely twenty eight bytes overhead to every packet (20 computer memory unit IPv4c header + eight computer memory unit UDP header).

• it's accessible to guest systems on hypervisors and clouds and is instantly out there to (Haskell) developers via commonplace modules.

Above all, the look of the UDP protocol fits the abovementioned notion of cloud computing well, that is, to accept occasional failure and manage it, instead of troubled to prevent it.

A.OUTWARDLOOKING PERSPECTIVE

The guest systems in clouds usually need to deal with the suboptimal network conditions caused by package devices, a tangle that the VEPA common place tried to unravel in 2009. The package devices, like vswitches and vrouters, for regulation network traffic inside the cloud nodes and ar guest systems themselves. Depending on the virtualization magnitude relation, one virtual switch could be chargeable for up to sixty four guest systems. Guest systems oftentimes need to deal with packet loss [3] that, when using TCP/IP prices several central processor cycles on systems that ar themselves beaked in step with the out there central processor cycles. Packet loss in TCP/IP will simply cause guest systems to grind to a halt. Therefore, it's cheap to assume that future cloud services, most of which can be obsessed with one of the seven trillion wireless devices, need protocols that ar considerably higher than communications protocol within the presence of errors.

II. RELATED WORK

According to [1], "the cloud demands obedience to overarching style goals", and "failing to stay the broader principles in mind" results in a disconnection of cloud computing analysis from universe computing clouds. Furthermore, scientists "seem to be guilty of fine-tuning specific solutions while not adequately brooding about the context that which they're used and also the real has to which they respond". One over arching style goal is to avoid sturdy synchronization provided by protection services. Wherever doable, all building blocks of a computing cloud ought to be inherently asynchronous. CMQ, being designed to fulfill the important desires of cloud computing, is strictly asynchronous and is that the combined analysis result from

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 7, July 2014

many alternative analysis fields, as well as network and data- center style, network protocols, message oriented middleware and practical programming languages.

A.UDP PROTOCOL

The more and more wide adaptation of the UDP protocol indicates the suitability of the UDP protocol as associate economical transport protocol for supporting distributed applications. For example, UDP is employed for information transportation in Network classification system (NFS) and for state and event transportation in huge Multiplayer on-line Games (MMOGs) [9,10]. Ever Quest, town of Heroes, Asheron's Call, Ultima on-line, Final Fantasy XI, etc. area unit among several MMOGs that use UDP as its transport protocol. The fact that MMOG applications area unit naturally of enormous, however elastic scale create them ideal customers for IaaS and PaaS offerings. By exploitation cloud computing and storage facilities, not solely value and risks, that area unit typically connected to assembling new MMOGs, reduced [4], however conjointly over-provisioning MMOG hardware to air standby for peak times are avoided. However, whether or not computing clouds will fulfil the demanding period needs of MMOGs continues to be an open issue [4].

B. MESSAGE ORIENTED MIDDLEWARE

If we tend to read computing clouds from the inward-looking perspective mentioned higher than, it is seen that the cloud framework itself could be a distributed application that successively supports distributed guest applications, for the explanations listed below.

• Computing clouds have associate inherent distributed character.

• The cloud framework permits snap, and parallelization (through distribution) of guest applications.

The guest application ought to be distributable in order that it's the flexibleness to be distributed to different resources once the bounds of the current out there cloud resources area unit reached. The connexion of message passing for computing clouds stems from the distributed programming model that's chosen to code either the cloud platform or the guest application.

C.FUNCTIONAL-PROGRAMMING LANGUAGES

The practical programing language Haskell is chosen as the programing language to implement CMQ because of the subsequent reasons:

• it's freelance from any third-party platform or runtime (e.g., Clojure and Scala area unit designed on high of JVM, NET platform).

• it's being actively researched associated has an ever increasing giant analysis community.

• It supports a large vary of concurrency paradigms .

• it's powerful in list manipulation. Lists area unit used in CMQ to supply light-weight organization that holds messages in sequence.

III. CMQ IMPLEMENTATION

CMQ may be a light-weight message queue enforced in Haskell. CMQ provides a polymorphic information kind Cmq a wherever a is that the content form of the queue. CMQ has currently 3 primitives: newRq (to initialize the queue and information structures), cwPush (to push a message into the queue), and cwPop (to pop a message from the queue).

cwPush

Messages for remote processes area unit known by a key tuple consisting of the scientific discipline address of the remote system and an number that is reserved for future use, for instance, it will be wont to specify the inflammatory disease of the remote method. When a message is pushed with cwPush 2 things happen:

• The key-tuple and therefore the information area unit keep in an exceedingly map, implemented exploitation the Haskell library information. Map (a dictionary that's enforced as a balanced binary tree).

• The key-tuple and therefore the creation time area unit keep in an exceedingly priority search queue (PSQ), enforced exploitation the Haskell library information. PSQueue and used as a pointer to the corresponding binding within the map.

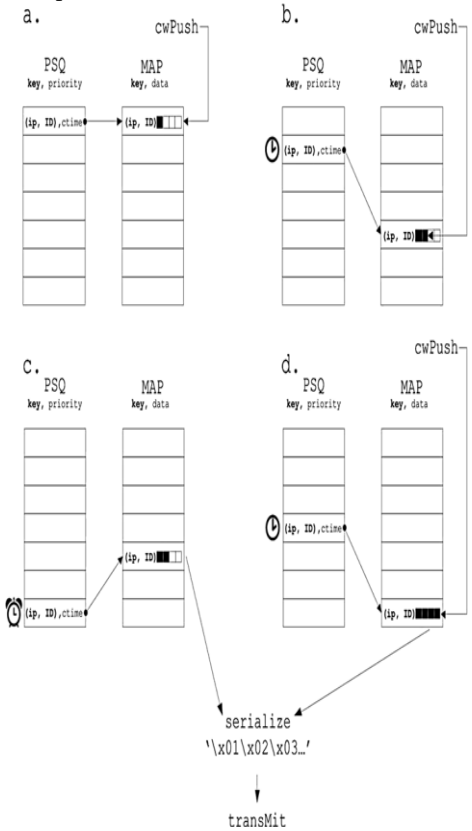

Fig 1 Map and PSQ pointer in CMQ

a). cwPush is called when the key for the recpient process is not present
b) cwPush is called when the key for the recipient process is already present
c) timeout for a key has been reached
d) cwPush is called when the key for the recipient process is present and the data length amounts to qthresh

.

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 7, July 2014

Fig 1 shows however CMQ works on the aspect of the sender. Once cwPush is termed to push a replacement message, a key-tuple k is made that consists of the scientific discipline address of the destination and a novel symbol (i.e. the PID). If the given secret's not already a member of the PSQ, then a new binding (k, p) is inserted wherever the priority p is that the creation-time of the binding. At an equivalent time a replacement key-value pair (k, a) is inserted into the map, where a may be a finite list that contains the pushed messages (Fig 1(a)). Message queues area unit keep within the map structure and therefore the map structure stores key-value pairs. the worth of every key-value pair may be a relevance a separate queue for a selected destination method. If at the time once a replacement message is pushed its key k is already a member of the PSQ, the new message is appended to the top of the queue that corresponds to k (Fig 1(b)). once the full quantity of messages in a queue (the gross length of all messages) for a selected key-tuple exceeds a group threshold (qthresh) then the whole queue are serialized and transmitted to the recipient (Fig 1(d)). so as to make sure that messages only keep within the queue for a brief time, a timeout threshold is employed. Once the timeout threshold is reached, all the messages within the queue are serialized and sent once the timeout threshold is reached (Fig 1(c)). The perform sendAllTo from the Haskell library Network.Socket. ByteString is employed to bring the UDP datagrams onto the wire. The perform sendAllTo guarantees that each one information is with success brought onto the wire which there have been no errors on the native network interface.

Since CMQ is enforced in an exceedingly pure purposeful programming language (Haskell), and pure purposeful information types area unit immutable , change a node by writing directly to memory isn't supported. the particular appending operation (++), that appends a replacement message to the top of a queue, doesn't update the tail node by dynamic its pointer so it points to the new accessorial message node, but recreates recursively every node within the queue, so that instead of writing a tiny low node and a pointer to memory, the perform returns, a whole new queue with the newly-added message returns [6]. This operation takes O(n) time. It is determined that, whereas the appending operation has time complexness O(n), adding a replacement message to the top of a queue, by consing (cons :) the new message node directly to the top of queue, takes O(1) time. therefore another methodology for the appending operation is to feature a new message node to the top of a queue rather than the tail. The queue created exploitation such a way maintains a reverse ordering of a FIFO queue.

IV. CONCLUSION AND FUTURE SCOPE

CMQ may be a light-weight message queue in Haskell. The idea to use UDP rather than TCP is intended by our understanding that, in Cloud Computing, present off-the-shelf technologies (both in hard- and software) are inspired, and if preventing errors from occurring becomes too expensive, handling the errors is also a stronger solution. This paper has incontestible the aptitude of victimization UDP for message queuing within the presence of errors, and has shown the steadiness of UDP electronic messaging in such conditions. strategies that take care of packet loss at the application level also are mentioned. The implementation of CMQ may be a Haskell Module that utilizes pure functional knowledge structures. The implementation of CMQ is available as module System.CMQ from the hackageDB at http://hackage.haskell.org/packages/ hackage.html, and also it are often put in mechanically via the Haskell package manager cabal on each Haskell Platform. Although CMQ may be a message queue oriented communication approach. CMQ will simply enough. It doesn't provide guarantees, therefore is extremely light-weight weight with low overhead and quick speed. though it will not provide guarantees, it seems to be stable within the presence of errors.

CMQ may be a start line for future analysis on distributed applications. One of the most goals of CMQ is to form the utilization of pervasive asynchronous correspondence simple and with stripped effort. Also, we have a tendency to also are fascinated by developing Associate in optional reliable layer that supports virtual connections, protocol-driven and/or application driven security mechanisms and knowledge to the target systems to enhance the performance and cut back the run-time distribution value.

REFRENCES

[1] Birman K., Van Ranesse R (2009), " Towards a cloud computing research agenda", SIGACT News 40(2): 68-80.
[2] Barraso L, Dean J, Holzle U(2003), " Web search : The Google Cluster architecture", IEEE Micro 23(2): 22-28.
[3] Wang GWT (2010), "The impact of virtualization on Network Performance of Amazon EC2 datacenter". IEEE INFO COM. Proceedings.pp 1-9,march 2010.
[4] Nae V, Prodan R, Fahringer T, Losup A (2009), "The impact of virtualization on the performance of massively multiplayer Online games, network and System support for games.
[5] Ren Y, Tang H (2010), "Performance comparison of UDP based protocol". Inf Technol J8(4); 600-604.