Design and Implementation of Cache in the Cloud for Cashing

Rajesh Babu Chebrolu¹, Durga Prasad Kondisetty², Betam Suress³

¹persuing M.Tech(CSE) Vikas Group of Institutions (Formerly known as Mother Theresa Educational Society Group of Institutions), Affiliated to JNTU-Kakinada, Nunna, Vijayawada, Andhra Pradesh, India.

²working as an Asst. Professor at Vikas Group of Institutions (Formerly known as Mother Theresa Educational Society Group of Institutions), Affiliated to JNTU-Kakinada, Nunna, Vijayawada, Andhra Pradesh, India.

³working as an HOD with Vikas Group of Institutions (Formerly known as Mother Theresa Educational Society Group of Institutions), Affiliated to JNTU-Kakinada, Nunna, Vijayawada, Andhra Pradesh, India.

Abstract: From the past several years cloud computing technology getting boom in the current world, most of the organization using cloud computing service for storing and processing their data. In Cloud computing there are number of distributed database connected through a common channel and cloud manager take look on it. In a simple web architecture, a client or user making his request to server for specific content. Responsibility of server is that server have to take his request and process this request for generating the response content. On server for each client, it have to process all request and respond the content every time each content is same for many time. It is overhead on server to provide same content number of time. Solution of this problem is cache. From several year cache becoming a vital technology for improving the performance of service. Quality of service is one of the major concern in any application. Caching technique is better way to improve the QoS. Here we proposing a Cache server system with association with clouds. A cache server is a computer or network device that has been setup to store web pages that have been accessed by users on the network. If any user was trying to access the web page that has already been stored on the cache server will be sent the stored version instead of downloading the related web page again individually. This mainly helps to reduce network traffic and Internet traffic congestion as well as saves the company on bandwidth costs. Implementation of cache in cloud for cashing in place of typical service is more beneficial for both user and service provider. We will check the feasibility of Cache service on elastic system and also check it for practical use. We will show here that using this service help us to leverage the cloud economy greatly in that extra user cost for input output performance gain is minimal if ever exits and also the provider’s profit will increase due to server consolidation resulting from that performance gain. Here the same cache technology is we are going to implement at server side and here the request for a particular file is given to server then it fetches the data from database server and if the request is given for same data again then we say that it is time consuming process to fetch again the same resource and here in this place we are introducing the concept of cache and it can improves the performance of server and automatically the burden on server will reduce and we are offering this as a cloud service that is Cache as a service (Caas).

Index Terms—Cloud Computing, Cache as a Service, Remote Memory, Cost Efficiency, QoS.

I. Introduction

The main goal of a cloud computing in the real scenario is to apply traditional supercomputing, high performance computing power, normally used by military and research facilities, to perform tens of trillions of computations per second, in consumer-oriented applications such as financial portfolios, to deliver personalized information, to provide data storage or to power large, immersive computer games. To do this, cloud computing uses networks of large groups of servers typically running low-cost consumer PC technology with specialized connections to spread data-processing in them. This type of shared IT infrastructure may contains large pools of systems that may link together. Also, virtualization techniques may be used to maximize the power of cloud computing. A cloud is a pool of number of distributed database which are linked together in a distributed environment. There is cloud owner or cloud manager which having control over all databases, these databases known as cloud. Cloud computing provide a huge benefit to user. Cloud computing provides facilities for user to develop and manage their own applications on the cloud, this enhance the concept of virtualization of resources. Through virtualization resources are managed by themselves. The implementation of cloud computer increased widely in organization. Cloud computing has started to obtain mass appeal in corporate data centers as it enables the data center to operate like the Internet through the process of enabling computing resources to be accessed and shared as virtual resources in a secure and scalable manner.

As the cloud service increasing world widely the resource redundancy also increasing in same manner for offering the services. For example many cloud service provider virtualize the resources such as processor, storage and network devices and offer them as a service on demand, this is known as Infrastructure as a Service (IaaS) which is main concerning...
focus of this paper. One of the basic implementation of Iaas is Virtual machine. As we know that VM is an computing plateform which is able to run multiple application on single plateform. It is known from the study that VM is solely dedicated to single application, because of that we using the expression VM application have to interchange themselves on that. Cloud Computing services virtualization making an virtual structure of unlimited resource capacity. This elasticity of cloud service brings cost effectiveness which is one of the primary driving force behind the cloud. Here we seeing that making cost efficiency on higher priority than the cost effectiveness is more useful and beneficial for both cloud user and cloud provider.

One technique also introduced by implementing a new device called as VMM-Bypass which will eliminate the data transfer between the guest operating system and give it accessibility to directly use the service as a native machine. With the increasing use of VM many hardware vendors providing virtualization as a service as a feature of hardware. We implementing the cache service with the virtualization feature.CPU cache is one of the basic example for checking the impact of caching in input –output functionality. When the processor needs to read from or write to a location in the main memory, it mainly checks that whether a copy of the data is in the cache or not. If it so happened, then the processor immediately starts reading from or else writes to the cache memory, which is more fastest than reading it from or writing data to main memory. In present scenario, most modern desktops and server CPUs may had atleast three independent caches: an instruction cache available to speed up executable instruction fetch, a data cache is used to speed up data fetch and store into the memory, and a translation look aside buffer (TLB) used to speed up virtual-to-physical address translation for both executable instructions and data that is available. The data cache is usually had been organized as a hierarchy of more cache levels. The proportion of accesses that is available in a cache memory to hit is known as that of hit rate of data, and it can be a measure of the effectiveness of the cache for a given program or algorithm. Read misses delay execution because they require data to be transferred from memory much more slowly than the cache by itself. Writing misses may occur without in such penalties, since the processor may continue execution while data is copied to main memory in the background. In order to make room for the new entry on the cache missed, then the cache may be have to choose one of the existing entries in it. The idea is that it uses for choosing the entry to eviction is called replacement policies. The fundamental problem here is with any replacement policies from that it must predicted which existing cache entry will be least likely to be used in our future. Predicting the future is difficult, so there is no perfect way to choose among

II- IMPACT OF CACHE WITH VIRTUALISATION

From the past decade virtualization becoming a famous area of discussion in point of providing services on a single machine in multiple manner. Number of research and study has been done in respect of judging the performance efficiency and effectiveness of VM. Virtual machine uses the concept of virtualization which enables the resources should be multiplexed on physical machine and they should isolated from multiple guests OS. By using VM we are able to run multiple operating system on a single machine. When we using VM ,one thing for care is that input –output functionality should be a safe and efficient manner. It is known from the several studies that the virtualization of input –output is one of the major problem in software due to its performance overhead. One thing coming out in that there is a difference between the performance when a native and virtual machine when they using the hardware resource. Her number of efforts has been done for narrowing the gap between virtual and native performance. In one research concept of page flipping came across in order to making good performance for virtual machine. In this technique they proposed that ,page flipping can be simply replaced by thememory function for avoiding the side effect. Another research has been come with the concept of Virtual machine Monitor global page and super page mapping which will optimize the input output performance of virtual machine.
the variety of replacement policies available. One popular replacement policy, least-recently used (LRU), replaces the least recently accessed entry. Marking some memory ranges as non-cacheable can improve performance by avoiding caching of memory regions that may rarely re-accessed in it. This may avoids the overhead that of loading something into the cache memory, without that having any reuse.

### III-SYSTEM ARCHITECTURE

For implementing the cache in the cloud for cashing, we make here a system architecture. Here we are going to implement Cacheonix for getting cashing scheme in clouds. Here we will see that implementing cashing through cacheonix will fulfill requirement of all component which require for implementing cache as a service like elastic cache system for architectural foundation and second one is pricing scheme as economical foundation. It is well known that every service have to full fill QoS, in which the two main fundamentals are effectiveness and efficiency. Cacheonix fulfilling all QoS required for architectural implementation and for economical implementation. Now here we are going to present the system architecture of cacheonix.

![Cacheonix Architecture](image)

**Fig 2-Cacheonix Architecture.**

Cacheonix can be defined as a reliable distributed shared associative memory. Each cache managed by Cacheonix carries a high-speed coherent local front cache, a partition and one or more replicas. Cacheonix provides continuous access to stored data even in presence of server failures and unpredictable network conditions by building on top of a highly-optimized cluster management protocol that ensures reliability and consistency of data access. Data in Cacheonix is addressed using a key. Each key has a value associated with it. The associative memory provided by Cacheonix is shared between a group of computers connected by a high-speed local network, or a cluster. All members of the cluster can read and write to Cacheonix. The content of the associative memory is evenly distributed (partitioned) among the members of the cluster. Cacheonix cluster is symmetrical. There is no primary node in the cluster. All nodes in the cluster are equal. Cacheonix automatically repartitions the data when a new node joins to the cluster. Each Cacheonix node maintains a coherent local cache that greatly speeds up access to the frequently used data. Cacheonix ensures that all nodes have a coherent view of the distributed shared memory while maintaining fast access to the cached data. Cacheonix uses multicast and point-to-point communications for cluster membership management and TCP sockets for the inter-node communications. Cacheonix fully supports clusters running on non-multicast networks by executing a proprietary TCP-based inter-node protocol.

### IV-RELATED WORK

**A.Fault tolerance:-** Fault tolerance is an ability to provide uninterrupted access to data in presence of a failure of a cluster node. Clustered applications have, despite the significant increase in hardware reliability, a very high probability of experiencing hardware failures due to the increased number of hardware components. Therefore, tolerating node failures is very important for long-running applications. Cacheonix is fault-tolerant. Cacheonix considers all failures as fail-stop failures. Cacheonix implements fault-tolerance by the means of creating back up copies of a value associated with a key. The number of back up copies is configurable and a range from 0 to N. Cacheonix is an N-tolerant system. It will continue to operate normally if Noperational - Nfailed >= Ntolerant. For 1-tolerant configuration the minimum number of nodes in the cluster is 3. While this works well for describing the static case when the failure occurs once, in real life nodes fail and repair multiple times throughout the life of the cluster. A node repair occurs when a previously failed node resumes operation or a new node joins. For the purposes of this discussion these are the same events. For Cacheonix to remain N-tolerant in a dynamic system, the following condition has to be satisfied: the rate of node failures should be lesser or equal than the rate of repairs by Ntolerant. In other words cluster nodes should repair at the same rate of faster then they fail. If this condition is violated, the cluster stops being N-tolerant until the condition is restored. The rate of repairs is a function of key-object size and network speed. Cacheonix has to create back up copies of the key-value pairs. The bigger the size of the key-value pair (KVsize), the longer it takes to restore N-tolerance. The higher the network speed (Nspeed), the shorter it takes to restore N-tolerance.

Time to repair for an unsaturated network can be calculated as

\[ KVsize*\text{Nbackup}/\text{Nspeed} = T_{\text{repair}}. \]
The repair rate can be calculated as:

\[ \frac{T_{\text{interval}}}{T_{\text{repair}}} = R_{\text{rate}}. \]

32-core machines with 64Gbytes of RAM (Big Fat Box) are not uncommon. A bigger machine means lesser maintenance. Unfortunately, this also means lesser fault tolerance. A big fat box can either pack up multiple small cluster nodes or a single larger node. Let's assume the MTBF does not depend on the size of the box. In other words, big fat boxes fail as often as the small ones. This means that to repair a single cluster node with 32 Gigabytes of heap will take 32 times longer then a node with 1Gigabyte of heap. Provide your Cacheonix-based application with a fast and, ideally, dedicated network infrastructure. Keep individual size of cached keys and objects reasonable Opt for small fast machines.

**B. Scalability:** Scalability is an ability of an application to process more requests by adding computational resources in it. Here we have mainly two types of scalability: vertical and horizontal. Vertical scalability is achieved by adding more computational resource, they are CPUs and RAM, to a single computer systems memory. Vertical scalabilities, in this while easy to achieve it, is limited memory. As although of this writing data, this single may be reasonably priced in computer system will be scaled only up to 64GB of RAM, 8 CPU cores, 10Gbit/s network I/O. As the demand grows, a Java web application needs more computing power and network bandwidth than single box will be provide to it. The application mainly has to scale horizontally in it. Horizontal scalability are of having ability to process more requests form users by adding computer systems to a set of computer systems that are running in the application and responsible for processing requests. This set of computer systems may also be known as a cluster. Applications architects are familiar with a phenomenon that adding more nodes to a cluster causes the ability of the application to process more requests to drop instead of growing. The application that may not respond with increasing ability for processing request by adding more server to the cluster is known as not scalable to it. An application that may does not having scale can only process request as fast as it slowest resource. The main source of scalability problems is an external bottleneck to it. A bottleneck may have a resource for that an access should be in serialized to it. When the bottlenecks are present in it, then the members of that clusters will only waiting for obtaining a right for overstating the resource because it may be accessed to only by a single cluster node at that time. Bottlenecks that are often affect web application are:

1. Database access.
2. File storage access.
3. Use of dedicated processing resources.

Distributed caching provides a solution for the problem of horizontal scalability by allowing a web application to avoid accessing the only slow serializable data source and that instead to get more data from the fast local memory to it. Because of this access to the external bottleneck will be removed from it, then the application may scale linearly, just by only adding more and more nodes to the given clusters. A Java web application using a distributed cache stores frequently accessed data such as results of requests to a transactional database in a large coherent cache memory. This may distributed cache that evenly spreads its own data across the clusters. Even though the distribution of cached information among the members of that cluster balances the load on every member. A distributed caching system maintains a fast and local cache, and also known as a front cache information, on each and every cluster member may tend to speed up the access to the cached data that may resides on every remote node and also to reduce use of network bandwidths. Cache coherence that may provided by only distributed by caching system ensures thats all members in the cluster have a consistent view in the data despite of local caching.

**C. Cache Eviction Policy:** A cache eviction policy is an algorithm according to which an existing element is removed from a cache when a new element is added in it. The eviction policies are applied to ensure that the size of the cache does not exceed a maximum size. Least Recently Used (LRU) is one of the most popular among a number of eviction policies in it. LRU may earned to its popularity and for being that best in capturing the temporal and spatial locality of data access. A minor disadvantage of LRU is its sensitivity to the full scanning system. The sensitivity that manifests itself for evicting accumulated by more frequently accessed in cache elements that when accessing the data that does not satisfy the requirement of temporal localities. This disadvantage is minor because of LRU recovers this from full scans quickly. A typical implementation of a cache with the LRU eviction policy consists of a map and a linked list array. Then the map stores cached elements in it. The linked list may keeps the track of the least recently used cache elements of it. When if any cache element was updated, it is removed from the list and also added to the top of that list array. Then the new elements may added to the top of the list as well as. If any of the cache grows bigger than its maximum size, an element will removed from that of the bottom from list and from the map also. By this way the least recently used elements are choose first from list. And While simple cache in Java may be implemented from a few lines of that code, it could be missing important features that are cache it needs to have in order to be usable from a real applications.

**V. CONCLUSION**

Finally we are going to provide a cache technique that can provide a good solution for improving your server performance and here how it is going to be increased means the no of round trips between database and server are reduced.
due to this cache and here the cache technique will act like a blood bank and here the cache is having frequently requested things and here if the requested term is there in cache then directly it gives that as response and so that is the reason why performance automatically increases and burden on server reduces. The performance improvement gained using cache services clearly leads to reducing the number of (active) physical machines the provided uses, increasing throughput, and intern results in profit increased. This profitability used improvement enabled the provided to adjust its pricing to attract more users. Cacheonix provides continuous access to stored data even in presence of server failures and unpredictable network conditions by building on top of a highly-optimized cluster management protocol that ensures reliability and consistency of data access.

REFERENCES

L. Wang, J. Zhan, and W. Shi ,


H. Kim, H. Jo, and J. Lee, “XHive: Efficient Cooperative Caching for Virtual Machines.”.


L. Cherkasova and R. Gardner, “Measuring CPU overhead for I/O processing in the Xen virtual machine monitor,” in

AUTHORS PROFILE

K Durga Prasad, is working as a assistant professor in Department of Computer Science Engineering at Vikas Group of Institutions (Formerly Mother Teresa Educational society Group of Institutions), Nunna, Vijayawada, Affiliated to JNTU-Kakinada, vizayawada, India

Betam Suresh, is working as an HOD, Department of Computer Science Engineering at Vikas Group of Institutions (Formerly Mother Teresa Educational society Group of Institutions), Nunna, Vijayawada, Affiliated to JNTU-Kakinada, vizayawada, India

Rajesh Babu Chebrolu, pursuing M.Tech in Computer Science Engineering from Vikas Group of Institutions (Formerly Mother Teresa Educational society Group of Institutions), Nunna, Vijayawada, Affiliated to JNTU-Kakinada, Vizayawada, India

AUTHORS PROFILE