Architecting Distributed Domain Reducer in Cloud Environment

Umar Khalid Farooqui
Manav Bharti University
Solan,H.P, India

Mohammad Hussain
M.G Institute of Management &Technology ,G.B.T.U, India

Ashish Kumar Trivedi
Manav Bharti University
Solan, H.P, India

ABSTRACT

Clouds computing is a recent technology used to represent a different way to architect and remotely manage computing resources; it is sharing resources/information as-a –service using internet. It 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. Cloud computing also describes applications that are extended to be accessible through the internet.

Here in this we would like to propose an architecture which reduce huge data set into smaller one. Huge data set could be list of document urls given by web services which is to be filtered according to user specific search criterion. It run jobs in parallel environment and also schedule jobs automatically. Also we will propose a technique by which it will clean up the data and reduce the complexity. Hope that it will be more efficient and effective.

Keywords

Cluster Container, Map Reduce, Persistence Storage, De-provisioning, Cleanup.

1. INTRODUCTION

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. [2]

The cloud applications are extended to be accessible through the internet. It uses large data centers and powerful servers that host web application and web services. Any one with suitable internet connection and a standard web browser can access a cloud application [2].

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[2].

A cloud is more than a collection of computer resources because a cloud provides a mechanism to manage these resources, management includes provisioning, change request, workload rebalancing, de provisioning and monitoring.

Cloud computing infrastructures can allow enterprises to achieve more efficient use of their IT hardware and software investments. Cloud computing permits management of the group of system as a single entity.

The infrastructure can be a cost efficient model for delivering information services, reducing IT management complexity, promoting innovations and increasing responsiveness through real time workload balancing.

Application built on cloud architecture runs in the cloud where the physical location of the infrastructure is determined by the provider, They take advantage of simple APIs of internet accessible services that scale on demand ,that are industrial strength ,where the complex reliability and scalability logic of the underlying service remains implemented and hidden inside the cloud.

We would like to design an architecture that allows user to filter the "Million Search Result" and hence reducing the Huge Data Set into smaller one. The application runs in parallel environment while the jobs are scheduled automatically, we also provide a technique which will cleanup the data and reduce the complexity and hope that it will be more efficient and effective.

2. BACKGROUND

Cloud computing automate management of group of systems as a single entity, cloud computing also describes applications that are extended to be accessible through the internet .Greg Boss, Linda et.al. Proposed architecture that focuses on the core backend of the cloud computing platform it does not address the user interface [2].

Aneka is a platform for deploying clouds, developing applications, it provides a runtime environment and a set of API that allows developers to build .NET applications that leverage their computation on either public or private clouds.

They propose a customizable and extensible service oriented run time environment represented by a collection of software container connected together [1]

Eucalyptus is an open source software framework for cloud computing that implements Infrastructure as a service (IaaS) system that gives user the ability to run and

control entire virtual machine instances developed across a verity of physical resources. It is a portable modular and simple to use infrastructure common to academics [4].

The Google File System is a scalable distributed file system for large distributed data-intensive applications. It provides fault tolerance while running on inexpensive commodity hardware, and it delivers high aggregate performance to a large number of clients. While sharing many of the same goals as previous distributed file systems, their design was driven by observations of their application workloads and technological environment, both current and anticipated that reflect a marked departure from some earlier file system assumptions. They present file system interface extensions designed to support distributed applications, discuss many aspects of their design, and report measurements from both micro-benchmarks and real world use. [5]

Map Reduce is a programming model and an associated implementation for processing and generating large data sets. Users specify a map function that processes a key/value pair to generate a set of intermediate key/value pairs, and a reduce function that merges all intermediate values associated with the same intermediate key. Many Real world tasks are expressible in this model [6].

3. METHODOLOGY

Map Reduce is a software framework introduced by the google in 2004 to support distributed computing on large datasets on clusters of computers. The framework is inspired by Map and Reduce functions commonly found in functional programming.

Map Reduce libraries have been written in c++, c#, Erlang, java, perl, pythen, and other programming language.

Map reduce is not a data storage or management system, it is an algorithmic technique for the distributed processing of large amount of data (Google web crawler is a real life example).

Hadoop Map reduce is an open source distributed processing framework that allows computation of large datasets by splitting the datasets into manageable chunks, spreading it across a fleet of machines and managing the overall process by launching jobs, process it and at the end aggregate the job output in to a final result .

It works in three phases

1) Map Phase
2) Combine Phase
3) Reduce Phase

A map phase transforms the input into an intermediate representation of key value pairs.
A combine phase combines and sorts by the keys. And a reduce phase recombines the intermediate representation into final output. Developer implements two interfaces Mapper and Reducer, while hadoop takes care of all the distributed processing (automatic parallelization, job scheduling, job monitoring, and result aggregation).

In hadoop there is a master process running on one node to oversee a pool of slave processes (workers) running on separate nodes .Hadoop splits the input into chunks. These chunks are assigned to slaves, each slave perform the map task(logic specified by the user) on each pair found in the chunk and writes the result locally and inform the master of the completed status .hadoop combines all the results and sorts the results by the keys the master then assigns keys to the reducers. The reducer pulls the result using an iterator, runs the reduce task (logic specified by the user), and sends the final output back to distributed file system.

The Map and Reduce functions of Map Reduce are both defined with respect to data structured in (key, value) pairs. Map takes one pair of data with a type in one data domain, and returns a list of pairs in a different domain:

Map $(K_1, V_1) \rightarrow$ list (k_2, v_2)

The Map function is applied in parallel to every item in the input dataset. This produces a list of (K_2, V_2) pairs for each call. After that, the Map Reduce framework collects all pairs with the same key from all lists and groups them together, thus creating one group for each one of the different generated keys.

The Reduce function is then applied in parallel to each group, which in turn produces a collection of values in the same domain:

Reduce $(K_2,$ list $(V_2)) \rightarrow$ list (V_3)

Each Reduce call typically produces either one value v3 or an empty return, though one call is allowed to return more than one value. The returns of all calls are collected as the desired result list.

Thus the Map Reduce framework transforms a list of (key, value) pairs into a list of values. These behaviors is different from the typical functional programming map and reduce combination, which accepts a list of arbitrary values and returns one single value that combines all the values returned by map.

It is necessary but not sufficient to have implementations of the map and reduce abstractions in order to implement Map Reduce. Distributed implementations of Map Reduce require a means of connecting the processes performing the Map and Reduce phases. This may be a distributed file system. Other options are possible, such as direct streaming from mappers to reducers, or for the mapping processors to serve up their results to reducers that query them.

3.1 Algorithm

The web service which crawl massive data creates a super set of URL's, the super set of URL's be represented as-
Super Set= {A, B, C, D…….Z},
Where each A, B...and so on, are it self a set of URL's, which can be represented as-

A= $\{a_1, a_2, a_3 \ldots\ldots\ldots a_n\}$,

B= $\{b_1, b_2, b_3 \ldots b_n\}$'

.

.

Z= $\{z_1, z_2, z_3 \ldots\ldots\ldots z_n\}$.
Here each a_1, $b_1 \ldots$ are individual URL.

The set of URL's are then passed to the proposed application as first input and the request pattern as second input, the I/O container will be responsible for maintaining a compressed file as input which keeps these sets (A to Z) in ordered fashion and hence create a list of sets and assign a unique "LocationPointer" to each set, Now each arbitrary set and its corresponding locationpointer is applied as (key, value) pair to a distributed map/reduce function.

The map function return these (key, value) pairs in a different domain.

Map (k_1 , v_1) ⟶ List (k_2 , v_2)

Map function is distributed and in parallel environment provisioned by Cluster Container, it produce various list of k_2 , v_2, these lists are then combined (intermediate result) and passes to reduce function, the reduce function finally aggregate these intermediate results and produce final result. Distributed map/reduce operation is pictorially represented in Fig 1.

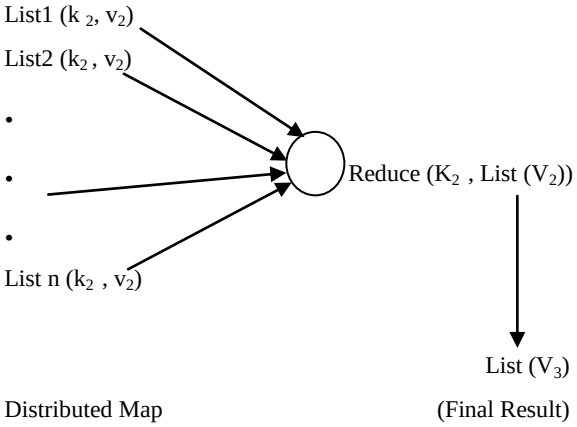

Fig 1: Pictorial Representation of Map/Reduce

3.2 Pseudo-codes

Void Map (LocationPointer, URLSET);

//URLSET is a sub set of huge data set from I/O container

//LocationPointer serves the basis for key, it is provided by //I/O container

For each URL in URLSET

EmitIntermidiate ("RequestPattern", URL);

Void Reduce ("RequestPattern", URLSET)

For each "RequestPattern" in URLSET

Append (URL, NEWURLSET);

Emit (NEWURLSET);

3.3 Our approach

The context diagram is shown in fig2. The web service that crawls web produce a huge data set (i.e. list of sorted document URLs), this huge data set is given as primary input to this application and another input is given by the user as 'request patterns'.

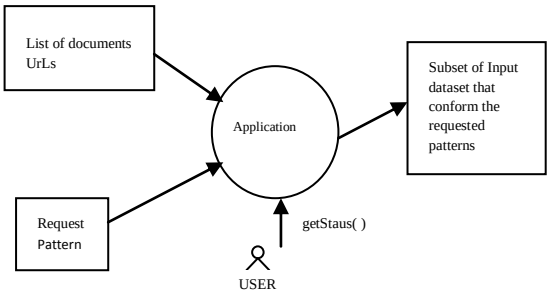

Fig 2: Context Diagram

It will then return a filtered sub set of document links as final output. Since the overall process is asynchronous, user can get the status of their job using getStatus( ).

The approach is to build an application that scales with demand, while keeping the cost of upfront investment down and to get response in a reasonable amount of time. It is important to distribute the job in to multiple tasks and to perform a distributed search application that run those tasks on multiple nodes in parallel. The application is modular it does its processing in four phases as shown in fig 3.

The initialization phase is responsible for validating and initiating the processing of a user request, starts all the job process initiating master and slave clusters.

The service phase is responsible for monitoring the clusters, perform map reduce, and checking for success and failure. The de provisioning phase is responsible for billing and de provisioning all the processes and instances.

The cleanup phase is responsible for deleting the transient data from the persistent storage. This application is modular and use following components

I/O Container: For retrieving input data sets and for storing output data sets.

Buffers: For durable buffering requests and to make the entire controller asynchronous.

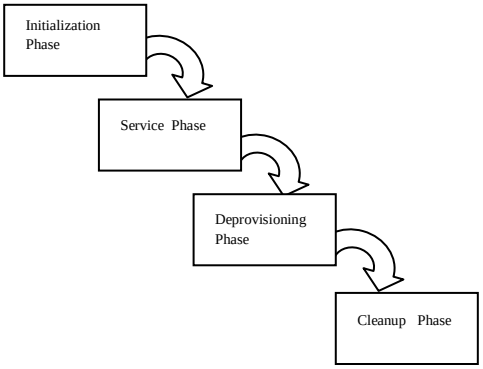

Fig 3: Phases of Application

Persistence Storage: For storing intermediate status, logs, and user data.

Cluster Container: For distributed processing, automatic parelellization and job scheduling.

The detailed work flow is shown in fig 4.

As application starts, buffers are created if not already created and all the controller threads are started. Each controller thread starts polling their respective buffer for any message.

When a start message is received an initialization message is post in the init buffer, the initialization controller thread pickup the initialization message and execute the task, update the status and timestamps in the persistent storage it also post a new message in the service buffer and deletes the message from the init buffer after processing.

The initialization task starts clusters and starts Map/Reduce task, it run map tasks on slave nodes in parallel. Each map task takes files (multithreaded in

background) from I/O container and writes the match result along with a description of up to 5 matches locally and then the combine/reduce task combines and sorts the result and consolidates the output .The final result are stored on the I/O Container's output bucket.

The service controller thread pickup this message, validates the status/error in the persistent storage and execute the task updates the status in the database post a new message in the de provisioning buffer and billing buffer and deletes the message from service buffer after processing.

During service phase it checks for the cluster status (job tracker success/failure) in regular intervals, updates the persistence storage (database) items with status/error and output file in the I/O container.

The de provisioning controller thread pickup this message from the de provisioning buffer and executes the de provisioning task, updates the status and timestamps in persistent storage, deletes the message from the de provisioning buffer after processing, this kills the process, terminates the instances of clusters and finally disposes of the infrastructure.

The billing task gets information of usage calculates the billing and passes it to billing service.

The cleanup phase archive the persistent storage's (the database's) data with user info.

User can execute getStatus ( ) on the service endpoint to get the status of the overall system and download the filtered result from the I/O container after completion.

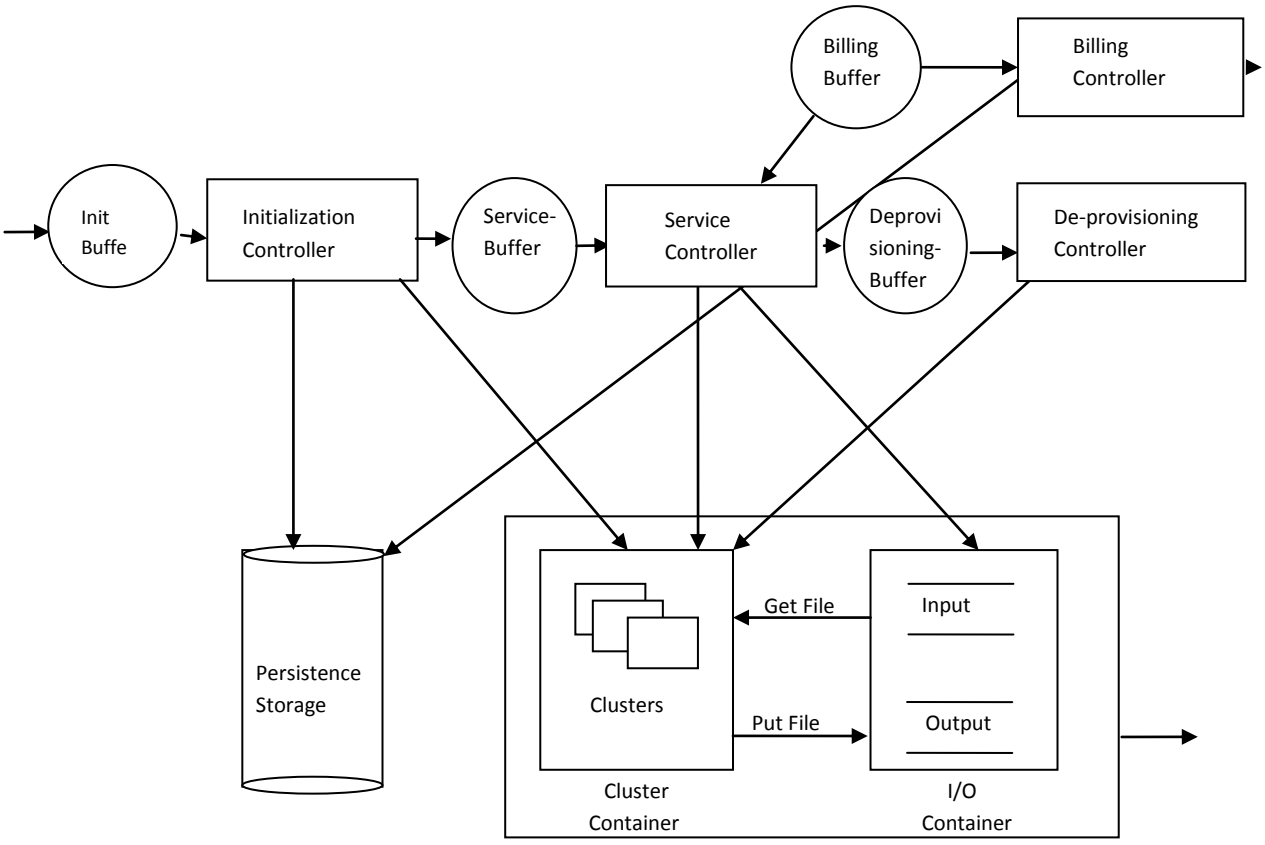

Fig 4: Architectural View

4. CONCLUSION

The proposed architecture reduces the huge data set(i.e., Search result given by web services) in to smaller one(i.e., user centric data set) using map /reduce operation, performed in distributed and parallel environment**.** In this architecture every phase is responsible for its own scalability; they are loosely coupled because of intermediate buffering and hence independent to each other. The map/reduce operation for filtering the result is performed parallel using cluster container services. Number of clusters/nodes can be decreased or increased on demand on pay per use basis; the billing controller is responsible for it.

Building application on fixed and rigid infrastructure is not fruitful for an organization, cloud architecture provides a new way to build applications on demand infrastructures. The proposed architecture depicts the way of building such applications. Therefore without having an upfront investment we are able to run a job massively distributed on multiple nodes in parallel and scale incrementally on demand ,without underutilizing the infrastructure and with in time.

5. REFERENCES

[1] Christian Vecchiola, Xingchen Chu, Rajkumar Buyya. In High Speed and Large Scale Scientific Computing (2010) Aneka: A Software Platform for .NET-based Cloud Computing

[2] Greg Boss, Padma Malladi, Dennis Quan, Linda Legregni, Harold Hall, "cloud computing", 15 Feb-2011. http://download.boulder.ibm.com/ibmdl/pub/software/dw/wes/hipods/Cloud_computing_wp_final_8Oct.pdf

[3] Deloitte: Cloud computing - A collection of working papers, released September 17, 2009 and published on July 31, 2009. http://www.bespacific.com/mt/archives/022426.h

[4] Daniel Nurmi, Rich Wolski, Chris Grzegorczyk et al. In Proceedings of *Cloud Computing* and Its Applications (October 2008)"The Eucalyptus: open-source cloud-computing system"

[5] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung, OSDI, "The Google File System", Published by ACM **...** Barbara Liskov ,SOSP- 2003, http://labs.google.com/papers/gfs-sosp2003.pdf

[6] Jeffrey Dean and Sanjay Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters", OSDI 2004, 6th Symposium on Operating Systems Design and Implementation. http://labs.google.com/papers/mapreduce-osdi04.pdf

[7] Junghoo Cho, Sridhar Rajagopalan "A Fast Regular Expression Indexing Engine", in Proceedings of the 18th International Conference on Data Engineering (ICDE.02)

[8] Junjie Peng, Xuejun Zhang , et.al,"Comparison of Several Cloud Computing Platforms" in Second International Symposium on Information Science and Engineering ,IEEE(2009).

[9] Liang-Jie Zhang and Qun Zhou "CCOA: Cloud Computing Open Architecture" in 2009 IEEE International Conference on Web Services.

[10] Huaglory, Tianfield , "Cloud Computing Architectures", in 2011, IEEE.

[11] V. Sarathy et al, "Next generation cloud computing architecture --enabling real-time dynamism for shared distributed physical infrastructure", 19th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE'10), Larissa, Greece, 28-30 June 2010, pp. 48-53.

[12] Mohamed Wahib, Masaharu Munetomo,et.al, "A Framework for Cloud Embedded Web Services Utilized by Cloud Applications",in 2011, IEEE - World Congress on Services.

[13] Huan Liu, Dan Orban, "Cloud MapReduce: a MapReduce Implementation on top of a Cloud Operating System",in 2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing.