PalArch's Journal of Archaeology of Egypt / Egyptology

SERVERLESS ARCHITECTURE FOR BIG DATA PROCESSING IN ENTERPRISE DATA HUB

Dr. Murugan A – Associate Professor and **Ganesan S** – Research Scholar,
PG & Research Dept. of Computer Science, Dr. Ambedkar Govt. Arts College,
University of Madras, Chennai, India

Dr. Murugan A , Ganesan S , Serverless architecture for big data processing in Enterprise Data Hub-Palarch's Journal Of Archaeology Of Egypt/Egyptology 17(7), ISSN 1567-214x

Abstract— In the era of modern utility computing theory, Serverless architecture is the cloud platform concept to hide the server usage from the development community and runs the code on-demand basis. Key objective is to abstract the infrastructure complexities of server management and scaling, but metered payment for processed requests only. This paper provides an algorithm to handle enterprise data hub using efficient serverless architecture for the complex big data query processing. This paper depicts about the experimental advantage of execution time optimization with agile availability and cost reduction of the underlying infrastructure, using new design of Serverless architecture.

Index Terms— Serverless, Utility computing, Big Data, Scaling, Metered payment, Enterprise Data Hub, etc.

I. INTRODUCTION

Data is foundation of the computer industry. In the rapid transformation of Information Technology, enterprise data growth is phenomenal. Enterprise Data Hub (EDH) is a solution to build and maintain the golden records of any enterprise as shared trustable enterprise data. With the big data Map Reduce algorithm, EDH is easily built using the traditional computing model. In recent industry days, serverless computing is highly adopted due to favorable small, self-contained units of computation, which makes the big data process easier to manage and scale in the cloud. Serverless computing programming model is highly influenced with inherit model of non-maintenance state of the underlying application. This paper depicts about the efficient way of utility computing to handle the data in the most effective logic. Key logic of this paper, is to build the serverless computing for EDH design.

II. LITERATURE REVIEW

A. *Enterprise Data Hub*

Enterprise Data Hub is a solution to build and maintain the golden records of any enterprise as shared trustable enterprise data. It is the proposed solution which is a central data repository for any enterprise with open sourced Big Data tools and techniques like HCatalog (Reference Data), Data Wrangler (cleansing), Hadoop Distribution File System, Map Reduce, Hive (distributed process). The final output of this framework is Master data.

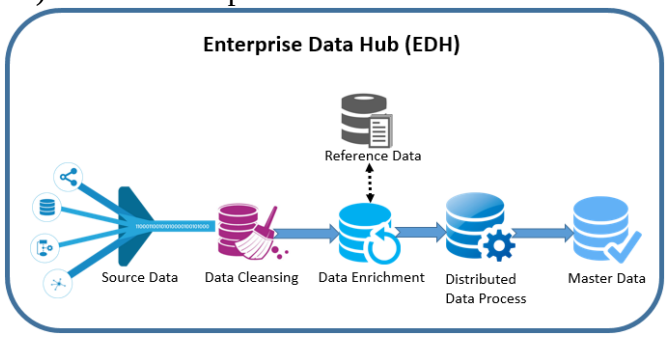

Fig. 1. Reference Architecture of EDH

B. *Shift in Serverless Architecture*

The word 'Serverless' doesn't literally mean as 'no server'. It leads to emerge Function as a Service (FaaS) concept. It allows small pieces of code represented as functions to run for limited amount of time on demand basis in the cloud.

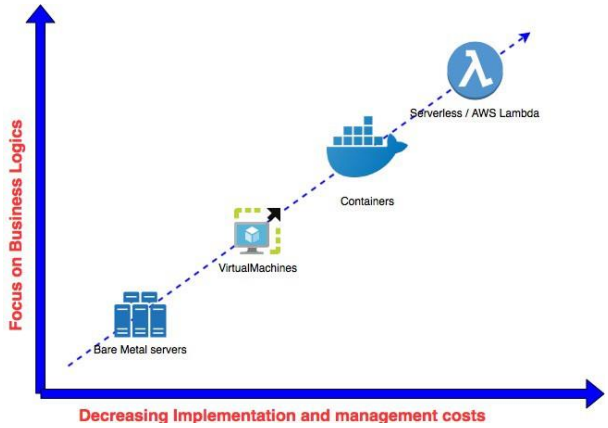

Fig. 2. Architecture Evolution of Serverless Computing

As depicted above, the physical machines took months to deploy any system in production with the life cycle in years. On evolution of VM and containers concepts, serverless computation deploy the production ready code with self-contained small units in the cloud.

III. SERVERLESS ARCHITECTURE

A. Definition

Serverless Architecture is disruptive design pattern to support the modern utility computing. It incorporates the custom written code along with third party services, in managed and ephemeral containers. This concept is terms as "Functions as a Service" (FaaS) platform. In a nutshell, FaaS executes the backend code with own short lived server application and defined as Serverless Architecture.

The ephemeral and stateless nature of the serverless micro services and functions that make up the modern distributed application is great for agility and scalability.

B. Industry Use Case

Allied Market Research report estimated the global serverless architecture market was worth ~$3.1 billion in 2017, and forecasts to ~$22 billion by 2025 with annual growth rate of 28%

Here is an industry use case to implement serverless architecture in an online shopping store.

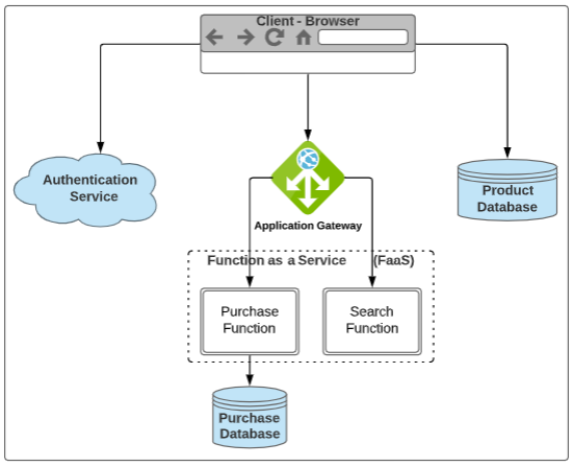

Fig. 3. Online shopping store

This design helps to parallelize into independent units of work in asynchronous and concurrent mode without worrying about the underlying infrastructure. In terms of changing business requirements, it is highly dynamic for accelerated developer velocity.

C. Message Driven Design - Illustration

Message driven design is the best suit for the serverless computing. In theory of computer architecture, message driven design is built based on the asynchronous communication. A message is a simple data transfer object (DTO) with message name and details. On arrival/dispatch of any message, the system triggers a function to execute. It is the fundamental of message driven application.

The context is usage of message driven design in FaaS. Let us consider a real

life application – Advertisement (Ad) server. Traditionally, Ad Server is designed synchronously with the response to the user click operation in a channel. In terms of operation, the user clicks on the advertisement content in the browser and the server collects the relevant information for further processing.

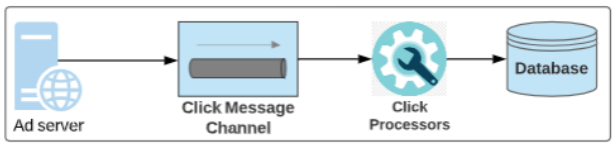

Fig. 4. **FaaS Message Driven Design**

As illustrated in the above design, it is redesigned with message consumer model using FaaS function of asynchronous message processing is a very popular use case for serverless architecture. This function runs within the event-driven context the vendor provides. FaaS is distinct to process several messages in parallel by instantiating multiple copies of the function code. Programmer doesn't need to worry about the underlying infrastructure as it is taken care by serverless technologies.

D. Benefits of Serverless computing

Scalability: Cloud's scalability is one of the powerful automatic mechanism to increase the infrastructure capacity dynamically. As serverless technologies handle the scaling demand seamlessly, software developers don't have to worry about the infrastructure policies.

Simplified server execution: FaaS is the core design of the serverless computing. It perform the single purpose function independently like general API call by abstracting the execution complexity.

Time to Market: Serverless architecture makes the availability of the program instantly and so it significantly reduces the software deployment cycle. As the result, the software launching time to market is quite fast.

Cost advantage: Traditional computing needs an upfront capital cost to purchase the physical servers well in advance. Serverless computing requires only the operational cost, no upfront capital. The rate of cost advantage is depicted as below for the serverless computing.

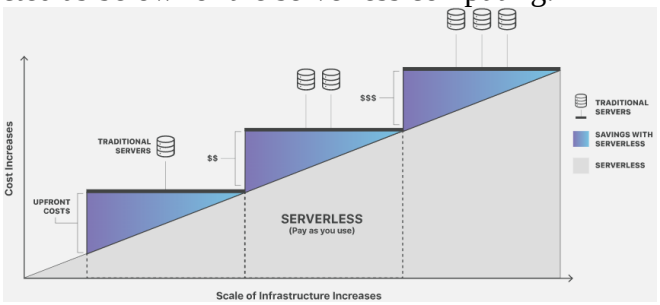

Fig. 5. **Cost Advantage of Serverless Computing**

E. Limitation of Serverless computing

No long term task: By design, serverless computing is intentionally ephemeral. As the execution is sustained for few minutes in the cloud, it doesn't fit for long running tasks. Also, it doesn't retain any stateful data of the previous run.

IV. TECHNICAL IMPLEMENTATION

A. Top-k Pricing algorithm using Map Reduce

Map Reduce is an algorithm to process and generate big data result with parallel and distributed computing concepts. Its implementation consists of a:

1. Map function that performs filtering and sorting of the given big data set, and a
2. Reduce function that performs a summary operation on the output of the Map function

Top-k share pricing search is one of the key functionalities in Enterprise Data Hub. Source data is ingested from New York Stock Exchange web site into Hadoop file system. Perceived historical data is queried by the user with Top-k stock details based on its closing price.

Pseudo code is drafted with parallel top-k query processing using Google's Map Reduce programming model.

Mapper processes the key value pair at a time and writes them as intermediate output on local disk. To do that, whole block is processed with key value pairs to find top-k result.

Algorithm: Map algorithm to process the big data

Input: Raw data to be processed

Output: Intermediate output on local disk

---

begin

Get the input of Object as key

Get the input of Text as value

G et the input of Context as context

function Type[] map

begin

Let tokens = value.split("\t");

Let price = tokens[0];

Let topPrice = parse (tokens[1]);

*LettopPrice = (-1) * topPrice;*

Write context as topPrice, price;

end

end

Reducer processes the key value pair at a time and writes them as final output on Hadoop Distributed File System. Pseudo code of Reducer function is depicted here.

Algorithm: Reduce algorithm to produce the final data

Input: Intermediate output on local disk

Output: Final processed big data result

---

begin

Get the input of LongWritable as key

Get the input of Iterable Text as value

Get the input of Context as context

function Type[] reduce

begin

Let price = null;

*Let topPrice = (-1) * key.get();*

for each (Text val in values)

begin

Let price = val;

end

if (count < k)

begin

Write the context as topPrice;

count ++;

end

end

end

In the above map reduce algorithm, it is evident to implement the efficient process of all key value pairs with lot of data movement in asynchronous functions.

B. ServerlessFaaS Design

Function-as-a-Service (FaaS) is the core design of serverless computing. It consists of smaller functions, which are managed by the cloud provided infrastructure. This new architecture model helps the various application patters like event handler, invocation patterns and compute-intensive big data process.

The abstraction level of FaaS architecture is depicted as below. It contains front end layer named Edge and back end layer named Master. On receipt of the various front end hits, Event Queue orchestrates the incoming request to Dispatcher component. By design, FaaS has multiple instances of the smaller functions as Worker. Dispatcher maintains and routes the job request to the distributed worker list in the Master layer.

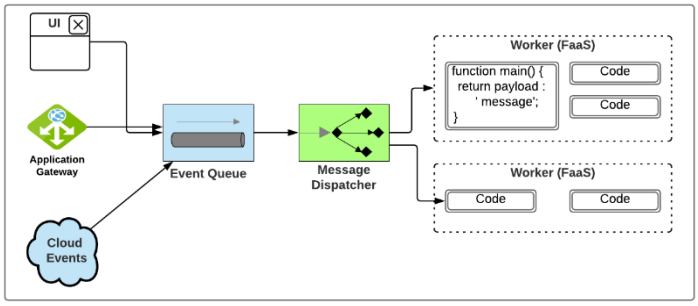

Fig. 6. Serverless FaaS Architecture

Typically, FaaS programming model consists of two major primitives namely Action and Trigger.

1. **Action:** Stateless function to execute arbitrary code.

2. **Trigger:** Events class from a variety of sources.

Actions can be invoked asynchronously either via REST API or based on trigger. A single event can trigger the multiple functions for parallel invocations. In contrast, the result of an action can trigger another function as sequential invocations.

function main (params, context)

begin

return { payload:

'stock name ' + params.name +

'top price ' + params.price

}
end

In the above serverless code, main function takes a dictionary using JSON object as input and produces output in dictionary format. By design, serverless functions don't maintain the state between the executions. So, the program has to handle to retrieve and update any needed state with context object.

C. ServerlessFaaS based Top-k algorithm

This serverlessFaaS based map reduce top-k algorithm has three key components namely

1. load processor
2. mapper
3. reducer

Initially, stock price data files is stored in the landing folder. It triggers the first task named load processor, which picks up the data file to process and load into mapper function of the algorithm. Mapper processes the input data to produce key value pairs. Reducer picks up the output of the Mapper to aggregate and produce the desired results.

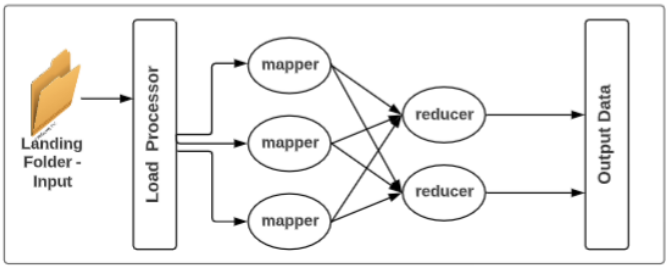

Fig. 7. Serverless Map Reduce Components

Mapper function splits the input of load processor into key/value pairs to proceed further. Mathematically, the size of the data chunk to be processed by each mapper is:

$$chunck\ size = \frac{total\ data\ size}{N_{mappers}}$$

where dividend is total data size of the input content and divisor is number of Mapper function.

Prior to pass the input from load processor to mapper function, it verifies 'chunk size' to satisfy the below conditions:

[1] If chunk size is smaller than the minimum block size specified by the user, chunk size will be set to Min block size. By doing so, the number of mapper function is calculated as:

$$N_{mappers} = int\left(\frac{total\ data\ size}{Min\ block\ size}\right) + 1$$

[2] If chunk size is higher than the calculated safe memory size of the underlying system, then safe memory size is calculated as a percentage of the memory assigned to the mapper functions. To fit the data in mapper function, the number of mapper function is calculated as:

$$N_{mappers} = int\left(\frac{total\ data\ size}{Safe\ memory\ size}\right) + 1$$

[3] In case of chunk size is higher than the maximum block size, it will be set to Max block size, which leads to calculate the number of mapper function as:

$$N_{mappers} = int\left(\frac{total\ data\ size}{Max\ block\ size}\right) + 1$$

The algorithm covered the possible conditions to recalculate the number of mappers function. In all scenarios, the load processor always adds one extra mapper with the responsibility to process the residual data chunk. The size of the residual data is calculated as:

$$residual\ data = total\ data\ size - \left(N_{mappers} - 1\right) * chunk\ size$$

This approach prevents any mapper to process its corresponding chunk plus the mentioned residual data, what could cause an additional overload in that mapper.

It is efficient to distribute the load across multiple reducers if the keys are not well distributed in the input data. To do so, it is vital to hash the keys using numbers and extract the residual part of the integer division with the total number of reducers. Mathematically, it was represented as:

$$reducer\ size = hash(key)\ \%N_{reduce}$$

Thus, map reduce algorithm is extended with serverless architecture for processing top-k pricing data.

V. EXPERIMENTATION RESULT

A. System Environment

To evaluate the proposed serverless algorithm, 5 nodes of virtual Linux servers are created in Amazon cloud environment. Sample data volume of 90 million rows with 6.39 GB sized content. Python 2.7 and Node.js software are used to develop the program. The system uses Amazon cloud's Lambda in conjunction with Amazon S3 to build a Map Reduce framework that can process data stored in S3.

Amazon Lambda provides the cloud platform for the execution of high throughput job. Each serverless Lambda function provides the service of memory between 128 to 3,008 MB, disk storage of 75 GB, concurrent execution of 1,000 tasks and timeout of 15 minutes.

B. Execution Time Optimization

Serverless big data map reduce top-k algorithm is benchmarked to measure the response time with meaningful queries – scan & aggregation. The function is executed against disk based algorithms – Hive &Serverless map reduce with three handful use cases.

Table- I: Experimented Result – Data Retrieval Time

Disk based Algorithms	Execution (in seconds)		
	Scan-1000	**Scan-100**	**Aggrega-tion**
Hive	52.4	61.5	741.1
Serverless map reduce	38.3	48.2	202

The execution data set contains 90 million rows and approximately 636 GB of data volume. Scan queries are built to select Top-k price value where k > z, with the constraint of z = {1000, 100}. The two scan queries are tested and execution time is captured against disk based hive and serverless map reduce algorithms.

Aggregation is the most complex query in any database management system. Serverless map reduce algorithm is more efficient than hive execution time. The data set 775 million rows with approximately 127GB disk size. The aggregate query covers the pricing field in Group By condition.

Experimented execution results are marked in the above table to demonstrate the time efficiency. Data points are graphically represented as below.

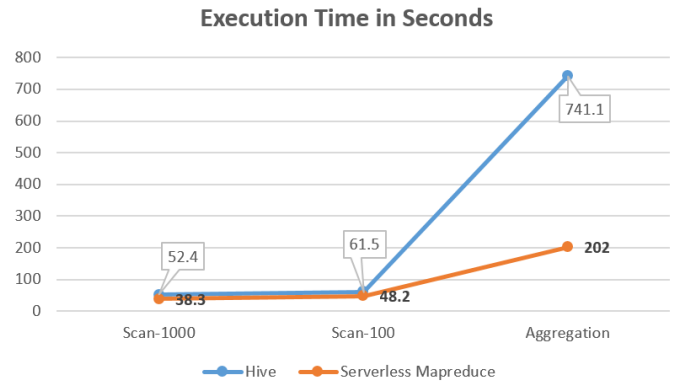

Fig. 8. Comparision chart of Scan & Aggregation queries

Result shows the huge improvement of big data retrieval execution time between hive and serverless map reduce algorithms. When the rate of data volume and complexity of queries increase, the execution time efficiency is improved as shown in the experimented result graph.

VI. CONCLUSION

This paper describes an improved version of big data map reduce algorithm, by leveraging the emerging serverless architecture. Amazon Lambda framework is used to implement serverless architecture in this paper. The core logic is to invoke Function as a Service in response to the distributed data based on the number of simultaneous mapper function. On comparing the performance of serverless architecture against Hive, it has an impact on the execution time efficiency of big data map reduce queries.

This research paper concludes the technical strength of new serverless architecture, using the experimental results on big data queries. Thus, enterprise data hub can leverage the state of the most efficient architecture to execute the big data processing Top-k algorithm.

REFERENCES

[1]. Ganesan S. and Murugan A., “Top-k Equities Pricing Search in the Large Historical data set of NYSE”, International Journal of Computational Intelligence Research, Volume 13 Number 1 pp 161-173, 2017

[2]. Bernstein D., “Containers and cloud: From LXC to Docker to Kubernetes”, IEEE Cloud Computing 1, 3 (Sept. 2014), 8184

[3]. Baldini I., “Serverless computing: Current trends and open problems”, Research Advances in Cloud Computing, Springer, 2017, 120

[4]. CNCF Serverless White Paper, https://github.com/cncf/wg-serverless, whitepaper

[5]. Yan M., Castro P., Cheng P. and Ishakian V., "Building a chatbot with serverless computing", In Proceedings of the 1st Intern Workshop on Mashups of Things, 2016
[6]. Ye W., Khan A.I. and Kendall E.A., "Distributed network file storage for a serverless (P2P) network", In Proceedings of the 11th IEEE Intern. Conf. on Networks, 2003, 343347
[7]. Wang L., Li M., Zhang Y., Ristenpart T. and Swift M., "Peeking behind the curtains of serverless platforms", In Proceedings of USENIX Annual Technical Conf., 2018, 133146. USENIX Association
[8]. Lin W-T, Krintz C., Wolski R., Zhang M., Cai X., Li T. and Xu W., "Tracking causal order in AWS lambda applications", In Proceedings of the IEEE Intern. Conf. on Cloud Engineering, 2018
[9]. Baldini I., Castro P., Cheng P., Fink S., Ishakian V., Mitchell N., Muthusamy V., Rabbah R., Suter P., "Cloud-native, event-based programming for mobile applications", In Proceedings of the Intern. Conf. on Mobile Software Engineering and Systems, 2016, 287288. ACM, New York, NY.
[10]. Etzioni O. and Niblett P., "Event Processing in Action", Manning Publications Co., Greenwich, CT, 2010
[11]. Kilcioglu C. Rao, J.M. Kannan and McAfee R.P., "Usage patterns and the economics of the public cloud", In Proceedings of the 26th Intern. Conf. World Wide Web, 2017
[12]. Oakes E., Yang L., Houck K., Harter T., Arpaci-Dusseau A.C. and Arpaci-Dusseau R.H., "Pipsqueak: Lean Lambdas with large libraries", In Proceedings of 2017 IEEE 37th Intern. Conf. on Distributed Computing Systems Workshops, 395400
[13]. Lee H., Satyam K. and Fox G.C., "Evaluation of production serverless computing environments", In Proceedings of IEEE Cloud Conf. Workshop on Serverless Computing (San Francisco, CA, 2018)