Lambda Architecture for Cost-effective Batch and Speed Big Data processing

Mariam Kiran
School of Computer Science
University of Bradford
m.kiran@bradford.ac.uk

Peter Murphy, Inder Monga, Jon Dugan
Energy Sciences Network (ESnet)
{pmurphy, imonga, jdugan}@es.net

Sartaj Singh Baveja
Netaji Subhas Institute of Technology
New Delhi
sartaj.singh@nsitonline.in

***Abstract*—Sensor and smart phone technologies present opportunities for data explosion, streaming and collecting from heterogeneous devices every second. Analyzing these large datasets can unlock multiple behaviors previously unknown, and help optimize approaches to city wide applications or societal use cases. However, collecting and handling of these massive datasets presents challenges in how to perform optimized online data analysis 'on-the-fly', as current approaches are often limited by capability, expense and resources. This presents a need for developing new methods for data management particularly using public clouds to minimize cost, network resources and on-demand availability.**

This paper presents an implementation of the lambda architecture design pattern to construct a data-handling backend on Amazon EC2, providing high throughput, dense and intense data demand delivered as services, minimizing the cost of the network maintenance. This paper combines ideas from database management, cost models, query management and cloud computing to present a general architecture that could be applied in any given scenario where affordable online data processing of Big Datasets is needed. The results are presented with a case study of processing router sensor data on the current ESnet network data as a working example of the approach. The results showcase a reduction in cost and argue benefits for performing online analysis and anomaly detection for sensor data.

Keywords—big data processing, lambda architecture, Amazon EC2, sensor data analysis

I. Introduction

The Cloud computing paradigm is a promising environment delivering IT-as-a-service for industries and researchers to deploy their applications [4, 8]. These capabilities have laid foundations for more innovative research challenges in Big Data and Internet of Things projects, with a continuing growth of massive and diverse data volumes, along with the use of data intensive applications. These areas present a need to investigate effective means for data management in efficient and cost-effective ways. Forecasting a growth of $75 billion for small and medium-sized businesses using Clouds for data management applications, SAP industries argue lower costs, less installation needs, and ease of management of less IT resources as an attractive business model [14]. However, this technological innovation, comes with increased challenges such as network availability, security and reliability as biggest concerns for businesses world-wide .

Initiatives such as Smart City projects are highly reliant on the availability of various services to fulfil their aims of data collection, management and processing. Access to certain architectures and resources to enable users to conduct Big Data and Internet of things research, has raised a number of issues of availability, know-how and security [20]. A constant growth in devices such as smartphones, sensors, household appliances, RFID devices, are joining internet capabilities to produce global data traffic of massive volumes and varieties, presenting various challenges for the security and management of these data-as-a-service applications [20].

With this in mind, multiple vendors are delivering services for data processing such as Amazon Web Services (AWS), Rackspace hosting and Google Cloud, presenting a collection of tools for online data collection, cloud hosted databases and map reduce processing such as using Hadoop, Hive or Spark. By offering users virtual machines to host, compute and manage their data, users can use advantages such as elasticity, multi-tenancy and the pay-as-you-go cost model. For instance, cloud resources can be rented with current Amazon services priced for small data resources (i2.xlarge) for $0.853/hour and for large data resources (d2.8xlarge) for $5.520/hour for on-demand resources. Additional reserved instances can be rented from 1 to 3 year terms, but may prove expensive in the long run, especially if data needs are not as intensive at all times.

This paper presents a cost-optimised architecture for online and batch data processing for massive volumes of sensor data as an adaptation of the lambda architecture design pattern currently being used by companies such as Twitter and AWS [21]. The architecture combines both batch and stream processing capabilities for online processing and handling of massive data volumes in a uniform manner, reducing costs in the process. The paper presents a flexible data provisioning based on the user needs and achieves the following:

- Data capability to be collected in online and processed on-the-fly for real time analysis.
- Capability to perform massive batch processes on historical data sets to observe data patterns over longer period of time.
- Investigate cost-effective solutions using cloud services for deploying this architecture.

The paper has been organised as follows: section 2 presents related work and research pertaining to data processing architectures and the challenges still faced in them. This section also presents an overview of the lambda architecture and how it is currently being used with Apache Storm and Hadoop. Section 3 presents the proposed architecture and implementation challenges of porting similar data processing toolkits on AWS. These observations are supported by a case study presented in section 4 showing online data collection and processing for multiple router sensors sending data at a constant rate of 30 seconds. The results and conclusions are discussed in section 5 and 6 with further future extensions to the architecture to enable future smart city projects.

II. Related Work

A. Current Data Processing Solutions

Data analytics are essential to plan and create decision support systems for optimising the underlying infrastructure. This involves not only processing of the online data, in search for certain events, but also the historical data sources which may be needed to find data patterns which influence decisions. Cloud providers are paramount for the availability and durability to their resources but present various challenges. For instance, for availability, data is often replicated across multiple servers in different geographical locations, sometimes in untrustworthy locations [6]. There are also additional computational challenges in handling elasticity by allocating resources on-the-fly to handle increased demand.

Mian et al. [22] presented a cost effective model for virtual machine provisioning to execute dynamic data analytic workloads, at the same time trying to satisfy all service level agreement (SLA) constraints. The paper highlighted how an optimised infrastructure would be more reliant on the provider setting up experiments and would not be defined SLAs. Dobre et al [23] presented a context aware framework, specifically designed for handling multiple devices, mapping between components and caching or handling requests from multiple users. As a means to support intelligent data processing through contexts, the authors however did not discuss how the data is moved through multiple abstraction layers to aid with speed and cost of delivery.

Further projects such as M3 [24] proposed a disk communication layer between the mappers and reducers to allow dynamic rate-based load balancing and multi streaming of applications. Another version of the project Chameleon [25] used specific context based indexing to augment query for fast data delivery. Other concrete projects such as Yahoo's Pig [17], Microsoft's SCOPE [5] and Google's initiatives [9], are aiming to integrate declarative query constructs from the database community into MapReduce-like software to allow greater data independence, code reusability, and automatic query optimization. These projects approached the problem as a distributed model, however further work needs to explore hybrid solutions which consider resources, data models, varied queries in accordance with network traffic or cost.

Researchers have often merged techniques with other tools to develop field related solutions. Abouzied [26] discussed HadoopDB, a hybrid of MapReduce and DBMS technologies, to allow scalability and performance of massive data processing. The authors present the application for a biological protein analysis or for business warehousing. Another example of merging was for image analysis in medical fields [27]. Bruns [28] discussed how the current Twitter APIs were extended for third party researchers to deploy their own data analysis on twitter feeds in order to enhance business practices. However unique solutions that allow multiple users of varying backgrounds to write and deploy optimised data processing applications is still needed. However there is a need for tailored solutions for online and batch data processing which keeps in line non-functional attributes such as cost and network complexities.

Further work has used similar data processing toolkits in smart grid applications where it is important to forecast and redistribute resources on the fly [31]. Current industry focus of using Spark SQL have aided further faster processing reducing some of the weaknesses of the Hadoop processing model [30].

B. Lambda architecture

Presented as a software design pattern, the lambda architecture unifies online and batch processing within a single framework. The pattern is suited to applications where there are time delays in data collection and availability through dashboards, requiring data validity for online processing as it arrives. The pattern also allows for batch processing for older data sets to find behavioural patterns as per user needs [21].

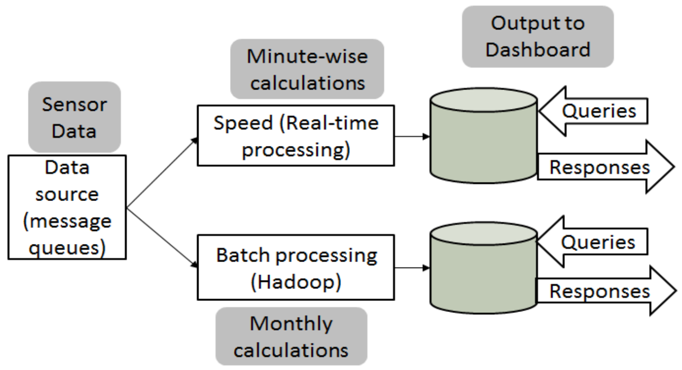

Fig. 1. Basic lambda architecture for speed and batch processing.

Figure 1 shows the basic architecture of how the lambda architecture works. It caters as three layers (1) Batch processing for precomputing large amounts of data sets (2) Speed or real time computing to minimize latency by doing real time calculations as the data arrives and (3) a layer to respond to queries, interfacing to query and provide the results of the calculations.

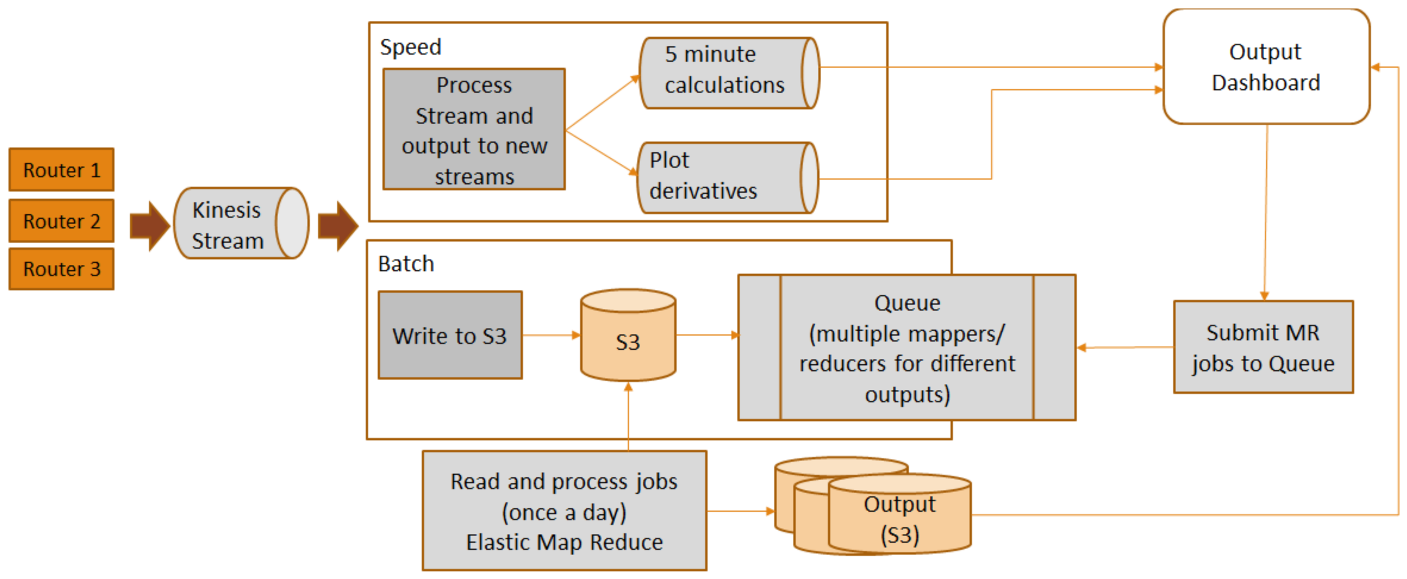
Speed
Process Stream and output to new streams
5 minute calculations
Plot derivatives
Output Dashboard
Router 1
Router 2
Router 3
Kinesis Stream
Batch
Write to S3
S3
Queue (multiple mappers/ reducers for different outputs)
Submit MR jobs to Queue
Read and process jobs (once a day) Elastic Map Reduce
Output (S3)

Fig. 2. Main lambda architecture implemented on Amazon web services.

Lambda architecture allows users to optimise their costs of data processing by understanding which parts of the data need online or batch processing. The architecture also partitions datasets to allow various kinds of calculation scripts to be executed on them [21]. However, a few critiques of the architecture have argued that the multiple set of projects that need to be maintained under the data branch to allow multiple data executions, requires more skills from the developers setting up the jobs to execute and produce results.

Despite of this, the architecture is well suited for big data processing problems with multiple kinds of analysis needed to study the online data arriving through sensors. The online stream can be used to detect data anomalies verifying whether it is accurate before processing it further. Verified data can then be stored into databases, which can have batch scripts performed once a day or a month to study data patterns over a time period. Users can reduce the costs of performing these scripts on larger data sets by breaking the problem down in manageable steps reducing cost and tailoring the data analysis routines to suit their needs. This architecture can be adapted for collecting and analysing online sensor data to find efficient solutions to process large data sets.

III. Proposed Architecture

In scenarios such as smart cities, involve working with large complex networks of sensors continuously fetching and recording data to a central repository for efficient decisions. Examples such as when to send garbage collection vans or when to grit the roads for better driving conditions can all be motivated through visual, motion and temperature sensor networks that already exist in city infrastructures.

Public clouds provide a number of services which could be employed for online and batch processing. Table 1 presents a comparison of Microsoft azure and Amazon AWS services offering similar capabilities. For the purpose of this paper, Amazon EC2 is chosen as a starting point for accessing multiple services. A comparison of the services presented in Table 1 shows that the online processing needs stream and batch processing which was easier to be performed in Amazon cloud rather than Azure services. The availability of services and cost plans for first time users of the Amazon infrastructure were also suitable for the project objectives.

TABLE I. Comparison of Cloud Services

Example services	***Microsoft Azure***	***Amazon web services Subhead***
Available Region	Azure Region	AWS Global Infrastructure
Compute Services	Virtual Machines (VMs)	Elastic Compute Cloud (EC2)
Storage Options	Azure Storage (Blobs, Tables, Queues, Files)	Amazon Simple Storage (S3)
Database Options	Azure SQL Database	Amazon Relational Database Service (RDS) Amazon Redshift
NoSQL Database Options	Azure DocumentDB Azure Managed Cache (Redis Cache)	Amazon Dynamo DB Amazon Elastic Cache
Data Orchestration	Azure Data Factory	AWS Data Pipeline
Administration & Security	Azure Active Directory	AWS Directory Service AWS Identity and Access Management (IAM)
Analytics	Azure Stream Analytics	Amazon Kinesis
Other Services & Integrations	Azure Machine Learning None None	None AWS Lambda AWS Config

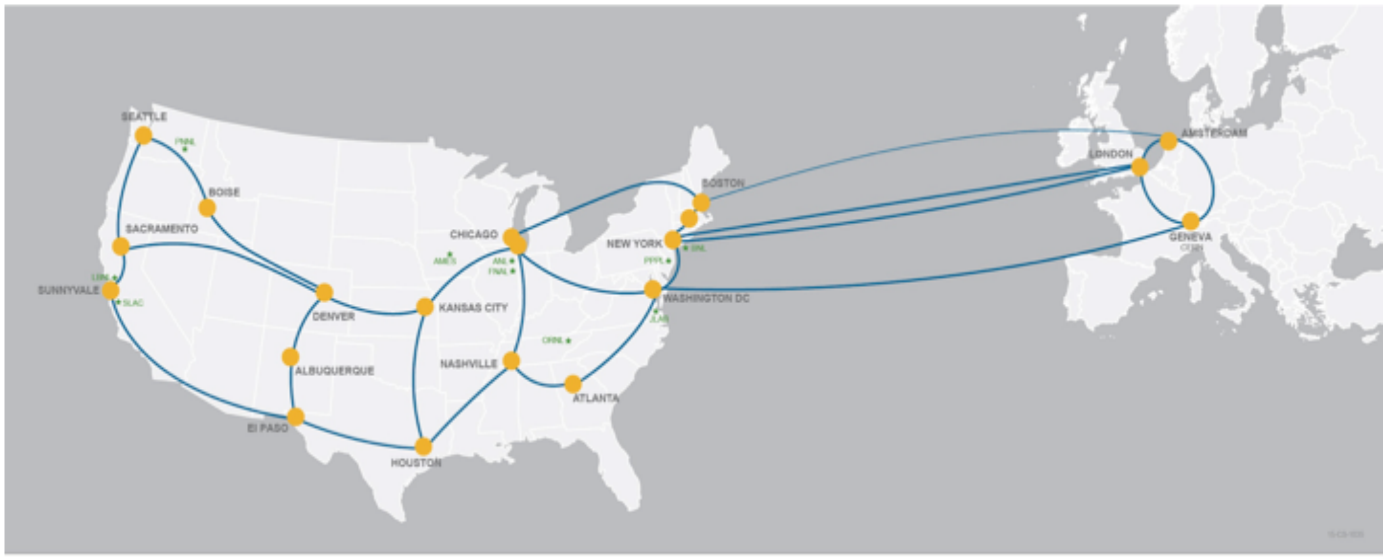

Fig. 3. ESnet router production network.

Amazon AWS offers a collection of services which could be used for different purposes, each differing in cost and time. Selection of the appropriate cloud service that maps onto the general architecture of lambda architecture was not obvious and required comparisons and study of performance, and cost. One of the decisions is showcased in Table 2, which presents a comparison of using either S3 or DynamoDB as a means to handle and process data. Although DynamoDB is much more expensive compared to S3, the speed of query processing would reduce the total effective cost as we plan for long-term use of DynamoDB rather than using S3.

TABLE II. COMPARING S3 AND DYNAMODB SERVICES

DynamoDB	**S3**
$0.02 per 100,000 transactions	$0.005 per 1000 requests
Storage costs vary. Maximum is $0.09 for storage	Storage costs vary. $0.03 per GB
Faster and DB	Blob

Similarly, a number of decisions had to be addressed in terms of cost and usefulness of the services. For the purposes of online processing of data, services such as Amazon Kinesis was chosen and merged with Amazon lambda for event-based processing of the data. Figure 2 describes the final processing architecture that was built on Amazon web services to read router data every 30 seconds and process it as it arrives and batch jobs.

IV. USE CASE: ESNET NETWORK SENSOR TESTBED

We used the entire ESnet router production network as the testbed to experiment with this architecture (shown Figure 3). An existing SNMP data collection software, ESxSNMP was used to collect router in and out bytes from every interface every 30 seconds.

Figure 2 describes the architecture that was built on Amazon web services to read router data every 30 seconds and process it in online and batch jobs. A recent report by Amazon [29] uses Apache spark and storm for processing the data stream. It also uses an event processing service which allowed processing scripts to be triggered when data arrives in the kinesis stream. In the architecture (figure 2) the event processing was omitted because in the use case, data was known to be arriving every 30 seconds making it less likely to have an event processing element. Having an event processing element also charges every time it is triggered, which would eventually charge more than the current architecture implemented.

The initial implementation report [29] also uses Spark SQL to perform batch processing for a fast query analysis. In figure 2, the basic elastic map reduce functions were implemented with Hadoop to perform map reduce processing jobs on hourly, daily and monthly bases in batches. The batch job could be triggered via cron jobs or through a job scheduler to run them once a day after the online data has been collected for the day. The map reduce jobs can filter and sort the data based on either hourly, 5 hourly or daily sorts.

A. *Real-time (online) or Speed processing*

The raw data arrives at 30second intervals from multiple router interfaces in the form of json files. These data sets were read and processed to calculate averages across minute intervals and the maximum values recorded. This has been explained below:

Arriving Json raw data: [router_id, interface_id, variable_id, timestamp, data_recorded]

5 minute aggregations: [router_id, interface_id, variable_id, 5_minute_avg, maximum_data_in_5_minutes]

The 5 minute aggregations were output to a new stream which could then be used to visualise the data while the data arrives.

B. Batch processing

Figure 4 shows the batch processing jobs on the raw data sets. Multiple map reduce jobs can be triggered to read the raw data sets and produce consolidated 1 day, 7 day and 90 day and 1 year aggregations. These batch files can only perform calculations on stored data sets.

Outputs for the calculated data sets can be read into output directory to visualise the averaged data sets. These outputs are also stored in separate S3 buckets.

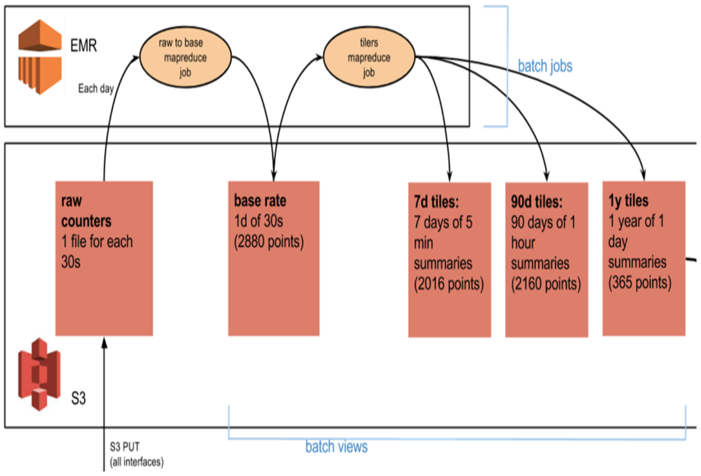
EMR
Each day
raw to base mapreduce job
tilers mapreduce job
batch jobs
raw counters 1 file for each 30s
base rate 1d of 30s (2880 points)
7d tiles: 7 days of 5 min summaries (2016 points)
90d tiles: 90 days of 1 hour summaries (2160 points)
1y tiles 1 year of 1 day summaries (365 points)
S3
S3 PUT (all interfaces)
batch views

Fig. 4. Batch processing on raw data sets.

The EMR scripts used c1.medium machines as master, core and task with the machine image version 3.8.0 and a Hadoop distribution of Amazon 2.4.0. The map reduce command used was as follows:

```
hadoop jar /home/hadoop/contrib/streaming/hadoop-
streaming.jar -files  s3://location-of-mapper/mapper.py,
s3://location-of-reducer/reducer.py  -libjars
/home/hadoop/CustomOutputFormats3.jar -outputformat
oddjob.hadoop.MultipleTextOutputFormatByKey -mapper
python mapper.py -reducer python reducer.py -input
s3n://location-of-inputs/jsons/ -output s3n://location-of-output-
job
```

The command, above, allows users to specify the location of mapper and reducer files, input files and where to produce outputs. Further java files can be passed as arguments to specify the format of the outputs generated as an optional step.

V. Results

The experiments were set up to run for 3 months executing certain jobs at specific times during the months. Figures 5-11 represent the statistics in terms of used hours, costs and type of services used during the time period.

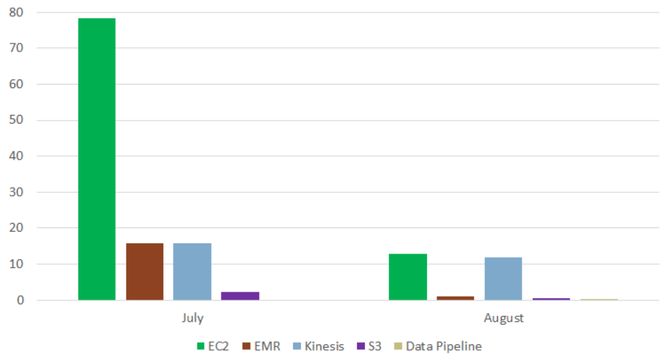

Fig. 5. Cost per service during two months (Month 1 and Month 2) of the experiment.

Figure 5 shows the cost variation between two months. Cloud services charge for the amount of usage. As shown in Figure 5, Month 1 used more demand on EC2 services as compared to the Month 2. This is reflected in Figure 6 and 7, where Month 3 used even lesser resources bringing the costs down from 80 dollars to 20 dollars.

Figure 7 shows that the cheapest services to use were micro machines and xlarge machines being the most expensive to use.

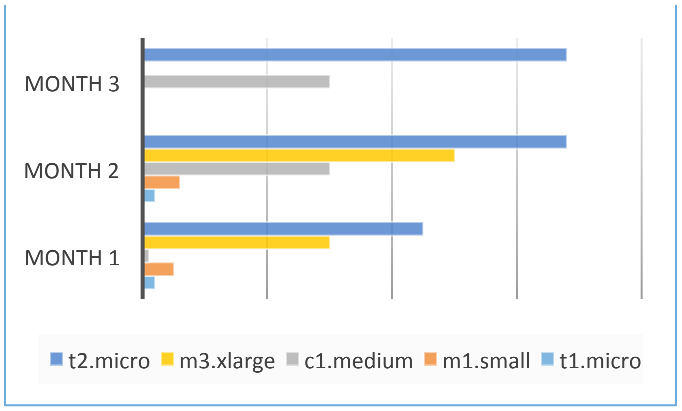

Fig. 6. Instance hours used over 3 months.

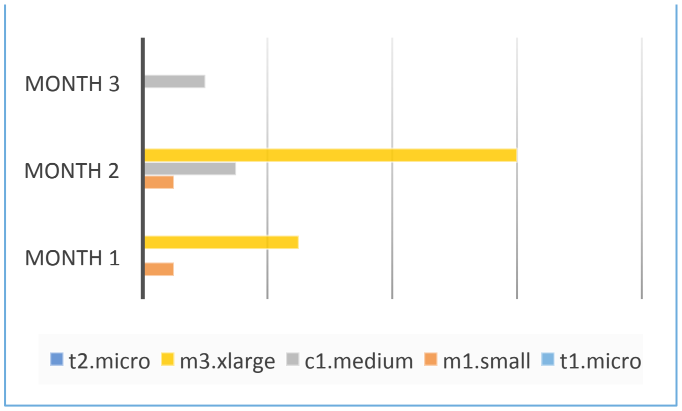

Fig. 7. Cost by instance type.

Further analysis on the usage of online kinesis stream and batch processing can be done as seen in Figure 8-11. Figure 8 and 9 show the distribution of requests used by the kinesis stream. Most of the kinesis cost arose from the stream hosting the data shown as shard active per hour. This shows that in the future cost could be optimised by limiting the amount of time the stream needs to be active by starting and deleting it, dynamically, when it is used.

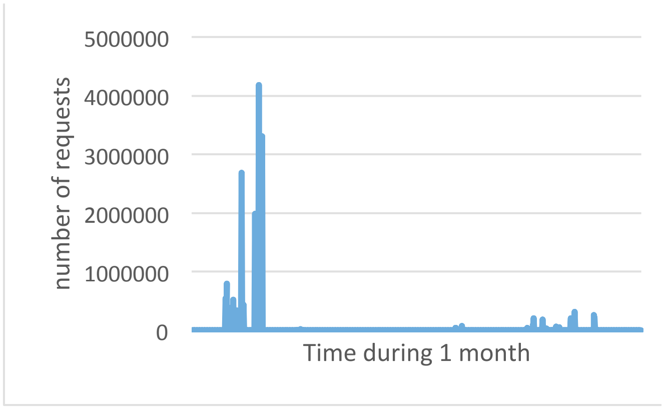

Fig. 8. Online stream: Kinesis usage during 1 month.

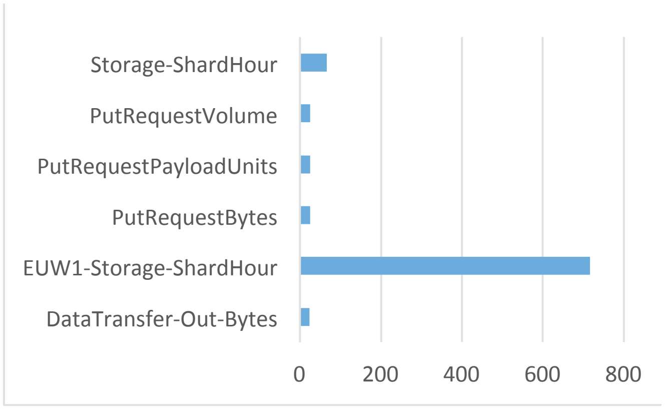

Fig. 9. Online Stream: Distribution of work in terms of requests for kinesis.

Figure 10 and 11 depict the usage of instances and the cost incurred by running batch jobs over the three month period using elastic map reduce. Using larges machine instances such as m3.xlarge machines produce more costs even if run a few times. These costs can be optimised by replacing larger machines with smaller machines such as c1.medium, to give similar results but at lower costs. Elastic map reduce involves creating a cluster of machines to act as master-slave to deploy and run the map reduce jobs. Therefore replacing these machines with smaller machines and more modes can help reduce the machine cost and also perform the same processing. The runtime of c1.medium cluster was approximately 10 minutes, which was slower (of 5 minutes) than the m3.large machine clusters.

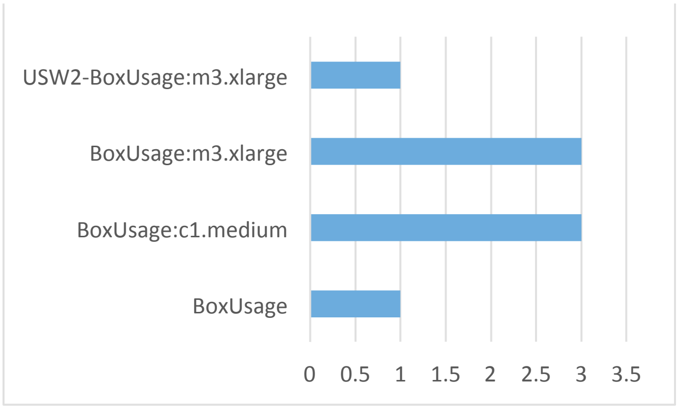

Fig. 10. Batch processing: usage of various instance types in EMR over 3 months

Figure 11 shows the reduction in cost by replacing the machine specifications for the map reduce clusters.

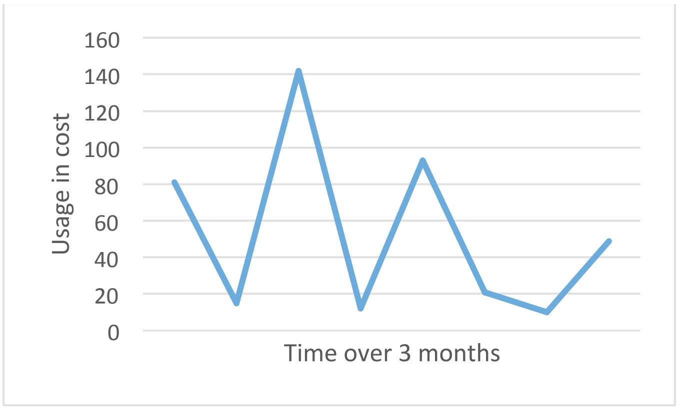

Fig. 11. Batch processing: price distribution of the EMR during 3 months

VI. Discussion and Conclusions

Lambda architecture allows multiple data processing scripts which are tailored to specific data sets. For online processing stream processing can be used to perform calculations as data arrives and batch processing scripts can be created to run on data stored from before. The above architecture was implemented on Amazon AWS utilising multiple resources and producing various results in terms of cost and usage. A number of lessons were learned while exploring the architecture on both batch and real time processing such as:

For batch processing:

- Performing local tests before deploying the scripts on EC2 helps to find code errors and manages in reducing the costs of failed clusters on EC2.
- Even if the cluster was executed for 2 minutes, Amazon cloud charges the machine as a full hour, which causes the steep increase in EMR costs shown in Figure 10. Which machines are used in the cluster cause an impact on the cost and also affects the time the service will take to execute the jobs.

- Future work can involve using spot instances to be requested, to allow optimising the costs even further but may introduce more delays in the processing being fulfilled.

For real-time or online processing:

- The kinesis stream read data as a last-in-first-out that meant data needed to be saved locally to calculate the 5 minute data aggregations.
- Amazon Cloud management allows roles and rights to be assigned to multiple members of a team. Multiple members may have rights for kinesis processing, but if the kinesis stream interacts with other services by moving data across would require the member to have rights to the other services as well. Software testing strategies such as try and catch exceptions need to be implemented in the code to prevent services to fail.

In terms of the processing time, the kinesis stream was able to process data in real-time while the EMR cluster used approximately 10 minutes to complete a job. Some of the services charge while they are active, and thus should only be dynamically started and stopped when being used in order to optimise costs.

In conclusion, the lambda architecture on Amazon AWS was able to provide a proof-of-concept for data processing in both as data arrives and if it is saved prior to the scripts. The tailored solutions allow users to perform cost-optimised processing. Both processes can produce data aggregations which are easier to plot and reduces time for data processing through the data visualisation interface.

Further work needs to be extended to explore issues of data security, availability zones and data replications for more complex operations of the services. Scripts which can perform data roll ups in case of data loss and machine-learning algorithms for performing online anomaly detection while the data arrives can be embedded into the stream, to check the data as it arrives. This would allow verification and validation of the data sets to be conducted as it arrives preventing loss in time and cost to run these scripts after the data has been saved and processed. However, the above experiment shows huge potential in processing of Big Data sets in cost-effective ways by implementing the lambda architecture design pattern and the architecture shows that it is particularly suited for cloud services where multiple resources are available for controlling and processing the data. This sort of architecture would prove extremely useful for processing sensor related data in Smart city research which is to be explored further in the future.

ACKNOWLEDGMENT

The work was supported by the NEMODE EPSRC grant (www.nemode.ac.uk) for Big Data analysis on Cloud services and further via ESnet resources and DOE support for Amazon Cloud services. ESnet is operated by Lawrence Berkeley National Laboratory, which is operated by the University of California for the U.S. Department of Energy under contract DE-AC02-05CH11231. This work was supported by the Directors of the Office of Science, Office of Advanced Scientific Computing Research, Facilities Division.

REFERENCES

[1] D. Abadi, Data Management in the Cloud: Limitations and Opportunities, IEEE Engineering Bulletin - DEBU , vol. 32, no. 1, pp. 3-12, 2009.

[2] R.M. Badia, M. Corrales, T. Dimitrakos, K. Djemame, E. Elmroth, A. Ferrer, N. Forgó, J. Guitart, F. Hernández, B. Hudzia, A. Kipp, K. Konstanteli, S. Nair, T. Sharif, C. Sheridan, J. Tordsson, T. Varvarigou, S. Wesner, W. Ziegler, C. Zsigri., Demonstration of the OPTIMIS Toolkit for Cloud Service Provisioning, Towards a Service-Based Internet LNCS vol 6994, 2011.

[3] J. Brown, Migration, Integration, Challenges in Government Cloud deployments, Govtech, 2015.

[4] R. Buyya, C. Yeo, S. Venugopal, J. Broberg, I. Brandic, Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility FGCS 25, 2009.

[5] R. Chaiken, B. Jenkins, P.-A. Larson, B. Ramsey, D. Shakib, S. Weaver, and J. Zhou. Scope: Easy and efficient parallel processing of massive data sets. In Proc. of VLDB, 2008

[6] M. Dikaiakos, G. Pallis, D. Katsaros, P. Mehra, A. Vakali, Cloud Computing: Distributed Internet Computing for IT and Scientific Research, IEEE Internet Computing, 2009.

[7] e-skills UK, Jan 2013 https://www.e-skills.com/research/research-themes/big-data-analytics/

[8] I. Foster, Y. Zhao, I. Raicu, S. Lu, Cloud Computing and Grid Computing 360-Degree Compared Proceedings of the IEEE Grid Computing Environments Workshop, pp. 1-10, 2008.

[9] J. N. Hoover. Start-Ups Bring Google's Parallel Processing to Data Warehousing. InformationWeek, August 29th, 2008.

[10] K. Jackson, K. Muriki, L. Ramakrishnan, K. Runge, R. Thomas, Performance and cost analysis of the Supernova factory on the Amazon AWS cloud, Scientific Programming 19, 2011.

[11] T. Jin, C. Tracy, M. Veeraraghavan, Characterization of high-rate large-sized flows, University of Virginia, Master's thesis, 2013, Online: http://www.cs.virginia.edu/events/colloquia/jin.html

[12] Z. Liu, M. Veeraraghavan, C. Tracy, J. Tie, I. Foster, J. Dennis, J. Hick, W. Yang, On using virtual circuits for GridFTP transfers, Int. Conf. for HPC, Networking, Storage and Analysis, pp. 81:1, 2012.

[13] Y. Lu, M. Wang, B. Prabhakar, F. Bonomi, ElephantTrap: A low cost device for identifying large flows, 15th Annual IEEE Symposium on High-Performance Interconnects, pp. 99–108, 2007.

[14] D. Menon, Benefits and challenges in deployment of Cloud solutions for SME, InsideSAP, 2014.

[15] C. Olston, B. Reed, U. Srivastava, R. Kumar, A. Tomkins. Pig latin: a not-so-foreign language for data processing, SIGMOD Conference, pgs 1099–1110, 2008.

[16] S. Sakr, A. Liu, D. M. Batista, M. Alomari, A Survey of Large Scale Data Management Approaches in Cloud Environments. IEEE Communication Surveys & Tutorials, vol 13, no 3, 2011.

[17] US DOE Office of Science ASCR, Terabit networks for extreme-scale science workshop report. Available Online: http://science.energy.gov/, 2014.

[18] S. Sarvotham, R. Riedi, R. Baraniuk, Connection-level analysis and modelling of network traffic, ACM SIGCOMM Internet Measurement Workshop, pp. 99–104, 2001.

[19] P. Xiong, Y. Chi, S. Zhu, H.J. Moon, C. Pu, H. Hacigumus, Intelligent Management of Virtualized Resources for Database Systems in Cloud Environment, In ICDE, 2011.

[20] N. Mitton, S. Papavassiliou, A. Puliafito, K. S Trivedi, Combining Cloud and sensors in a smart city environment, EURASIP Journal on Wireless Communications and Networking, December 2012, 2012:247

[21] N. Marz, J. Warren, Big Data: Principles and best practices of scalable realtime data systems. Manning Publications, 2013.

[22] R. Miana, P. Martina, J.L. Vazquez-Poletti, Provisioning data analytic workloads in a cloud, Future Generation Computer Systems 29 (2013) 1452–1458

[23] C. Dobre, F. Xhafa, Intelligent services for Big Data science, Future Generation Computer Systems 37 (2014) 267–281

[24] A.M. Aly, A. Sallam, B.M. Gnanasekaran, L. V. Nguyen-Dinh, W.G. Aref, M. Ouzzani, A. Ghafoor, M3: stream processing on main-memory MapReduce, in: Proceedings of the 2012 IEEE 28th International Conference on Data Engineering, ICDE '12, IEEE Computer Society, Washington, DC, USA, 2012, pp. 1253–1256.

[25] T.M. Ghanem, A.K. Elmagarmid, P.-A. Larson, W.G. Aref, Supporting views in data stream management systems, ACM Transactions on Database Systems 35 (2008) 1:1–1:47

[26] A. Abouzied, K. Bajda-Pawlikowski, J. Huang, D. J. Abadi, and A. Silberschatz. 2010. HadoopDB in action: building real world applications. In Proceedings of the 2010 ACM SIGMOD International Conference on Management of data. ACM, New York, USA

[27] Song X-W, Dong Z-Y, Long X-Y, Li S-F, Zuo X-N, Zhu C-Z, et al. (2011) REST: A Toolkit for Resting-State Functional Magnetic Resonance Imaging Data Processing. PLoS ONE 6(9): e25031. doi:10.1371/journal.pone.0025031

[28] A. Brun, Y. Liang, L. Eugene. Tools and methods for capturing Twitter data during natural disasters. First Monday, [S.l.], mar. 2012. ISSN 13960466.

[29] Amazon Web Services, Lambda Architecture for Batch and Stream Processing on AWS, May 2015

[30] Xin, Reynold; Rosen, Josh; Zaharia, Matei; Franklin, Michael; Shenker, Scott; Stoica, Ion, Shark: SQL and Rich Analytics at Scale, 2013

[31] Y. Simmhan, V. Agarwal, S. Aman, A. Kumbhare, S. Natarajan, N. Rajguru, I. Robinson, S. Stevens, W. Yin, Q. Zhou and V. Prasanna, Adaptive Energy Forecasting and Information Diffusion for Smart Power Grids , IEEE International Scalable Computing Challenge (SCALE 2012) , 2012