ISSN 2278-3091
Volume 9, No.4, July – August 2020
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse200942020.pdf
https://doi.org/10.30534/ijatcse/2020/200942020

Cloud Data Pipeline Automation Framework with Integrated Budget Management

Sumant Kulkarni[1], Dr. Shashank Joshi[2], Dr. Manjusha S. Joshi[3]
[1]Data Analyst and Cloud developer, Freelancer, Pune, India, sumant.shilpa@gmail.com
[2]Professor, Dean Faculty of Engineering and Technology, BVDU, Pune, India
[3]Asst Professor, SKNCOE, Vadgaon, Pune, India

ABSTRACT

Today is the era of Big Data and Cloud Computing. Organizations, Universities and Governments are investing heavily in big data analysis which provide strategically important insights. The cloud platforms, with virtually infinite amount of on demand and scalable resources, are used to execute the required data transformations and pipelines. As the scope of the data analysis activities widen, organizations need a framework to schedule the data pipelines and achieve the objectives of Budget management, Data pipeline prioritization and Reuse encouragement. This paper presents a "Cloud Data Pipeline Automation Framework" with several checks and controls. The framework was evaluated on Google Cloud Platform and the results establish the effectiveness of the framework. By using this automation approach, organizations would realize the true benefits offered by the "pay-as-you-use" cloud platforms.

Key words: Cloud Budget Management, Data Pipeline Automation, Data Pipeline Prioritization, Effort optimization, Scheduling Algorithm.

1 INTRODUCTION

Big data analysis provides valuable insights to organizations. Availability of on demand, scalable cloud resources have further boosted its adoption. However, as more and more data analyses are performed on cloud, a few unique challenges have emerged as described below:

A. Imagine a healthcare organization analyzing the patient care and revenue data to distill useful information. There can be several use cases, e.g.
   (i) Assess regulatory compliance of health care quality.
   (ii) Identify the trends in the patient service quality.
   (iii) Identify most / least effective treatment strategies.
   (iv) Profitability assessment of facilities and services.
   (v) Project future needs and emergency preparedness.

   Organizations need to choose and prioritize these use cases and apportion the budget to them.

B. The base data used for this type of analysis need not be static. As new data becomes available on daily/weekly/monthly basis, some of these analyses will need to rerun and refreshed periodically.

C. Some of the analyses can be shared with customers and third parties and may generate direct revenues. Organizations need to be prepared to respond in timely manner for any such ad-hoc requests.

D. Many analyses are used for improving process / quality / revenue for the existing services / products and are funded from internal budgets. Organizations need to ensure adherence and proper utilization of the internal budgets.

E. Organizations need to budget for multiple iterations of the analysis to improve the outcomes.

F. The Analysis is performed as a sequence of data transformations which are tied together in a data pipeline. Reusing the transformation components across multiple pipelines can save significant time and effort of the data engineers and is also less error prone. Organizations need to provide a structure to encourage such reuse.

G. Cloud platforms provide virtually unlimited on demand resources. If a data pipeline erroneously starts consuming too many resources or takes unusually long time; significant wastage may result. Organizations need to safeguard against this.

To address the above challenges, we propose a "Cloud Data Pipeline Automation Framework". The rest of the paper is organized as follows. Section 2 gives an overview of the related works. Section 3 lists the objectives of the proposed framework. Section 4 describes the approach taken to achieve the objectives using cloud services and tools. Section 5 presents the "Cloud Data Pipeline Automation Framework". Section 6 establishes the effectiveness of the framework based on evaluation results. Section 7 has the concluding remarks.

2 RELATED WORKS

Addressing the challenges related to Big data processing on cloud is a subject of much interest for the academia. Several efforts aim at resource, time, and cost optimizations on the cloud platforms. Heuristic workflow scheduling algorithms with various constraints are described in [2], [3], [5], [22], and [23]. A summary of various workflow scheduling techniques and their classification based on their objectives and execution models is presented in [13]. Review

of the approaches and techniques available for orchestrating big data workflow in the cloud is provided in [14]. All these solutions are aimed at cloud providers rather than cloud user organizations. Cloud platform improvisations are readily available to all cloud users. However, these solutions do not cater for cloud user side data pipeline scheduling challenges described in the Introduction above.

There are many "cloud based" proprietary data pipeline (ETL) tools available for data processing like AWS Glue, Fivetran, Blendo, Matillion, SnapLogic, Confluent, and XPlenty. Likewise, many open source ETL tools like Stitch, Talend, Apache Airflow, Apatche Nifi, KNIME [1] are also available for the organizations to use. Many of the tools have easy to use web interface for the users to define their data pipelines. They provide connectivity to all public cloud platforms as well as a host of data sources. Most of them have built in transformations for data transfer, cleanup, and integration. Some of the tools like Airflow [24], support workflow automation and scheduling. PipeFlow [6] is a tool that allows the user to write a stream processing script in a higher level data flow language which can run on any streaming engine like Spark or Storm. ShareInsights [7] is a data pipeline platform which uses a custom language, library of components and collaboration tools to accelerate data pipeline development. Apache Zeppelin [11] is a web-based notebook that enables data-driven, interactive data analytics on a wide range of technologies.

However, to the best of our knowledge, none of the "of the shelf" tools provide the facility of Budget management and prioritization while scheduling the data pipelines. Jürgen Cito et al. [8] have concluded in their systematic study on how software developers build applications for the cloud that "Cloud costs are deemed as important but are not tangible to developers." Syed Karimunnisa et al. [21] have mentioned "Scheduling for Resource Optimization" as one of the current research trend and issue in Cloud computing. Hence, it is imperative that budget management, use case prioritization and reuse be integrated with the data pipeline scheduling framework. Our framework aims to achieve the same.

3 PROBLEM DEFINITION

Organizations employing large scale data analysis activities using cloud platforms, need a data pipeline scheduling framework to achieve following objectives:

A. Define Data Transformations
B. Build Data Pipelines using Transformations
C. Repository of Transformations and Pipelines for reuse
D. Scheduling Data Pipelines
E. Prioritizing Data Pipelines
F. Monitoring Data Pipelines
G. Budget Allocation and Monitoring.

4 RESEARCH APPROACH

We present a new "Cloud Data Pipeline Automation Framework" which addresses above challenges. We have built a prototype tool using the framework and tested it on Google Cloud Platform. We compared the performance before and after using the tool and have presented the results. The results confirm the effectiveness of the framework vis a vis the stated objectives.

4.1 Key Design Considerations

- The budget allocations for data processing activities need to be drilled down to the data pipeline level by the business. It is possible to set up pipelines with unlimited budget.
- To support fluctuations in business, we have kept the data pipeline prioritization in the hands of the user and have avoided automation of the same.
- The scheduling of the pipeline is based on "Priority Scheduling" algorithm. The algorithm is enhanced with several Checks and Controls.
- Apache Airflow [24] is used to build the transformation DAGs. It provides support for a large set of operators including custom operators, so that any underling big data technology can be used for the data transformations. The DAGs also serve as a Data transformations repository.
- The Data Pipeline design is kept simple and is made of a fixed sequence of Data Transformations. Any conditional flow control is handled inside the Airflow DAG. This eliminates the need for a rich and complex GUI for defining the data pipelines.
- The framework has provision for building data pipeline schedules in advance, to aid planning and reduce delays.
- Since the challenges are common to all cloud platforms, the framework is cloud platform independent. The architecture is described using GCP tools and services. But similar and equivalent tools in AWS and Azure cloud platforms are mentioned in Table 3.
- The framework itself is on cloud platform as it is convenient for cloud users.
- Security is also built-in as it is always crucial in cloud environments.

4.2 Google Tools and Services used in the framework

- Cloud Function [25] – It is a serverless execution environment for building and connecting cloud services and responding to events and triggers.
- Cloud Scheduler [26] – It is a fully managed enterprise-grade cron job scheduler.
- Cloud Pub / Sub [27] – It is a fully-managed real-time messaging service that is used to send and receive messages between independent applications.
- Cloud Datastore [28] – It is a NoSQL document database built for automatic scaling.
- Cloud Composer [29] – It is a managed workflow orchestration service that is built on Airflow.
- Cloud Storage [30] – It is a globally unified, scalable, and durable object storage.

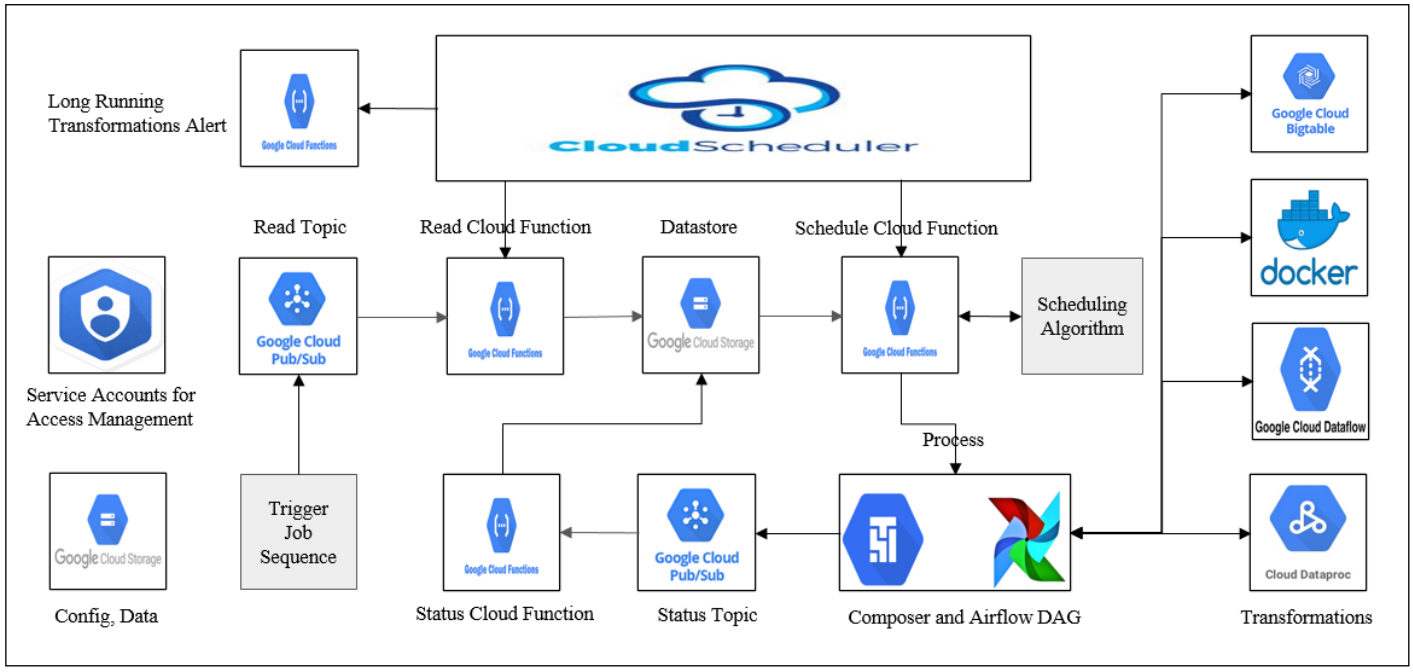

Figure 1: Cloud Data Pipeline Automation Framework Architecture

5 CLOUD DATA PIPELINE AUTOMATION FRAMEWORK

5.1 Overview (Please refer to Figure 1)

- Data processing involves executing several transformations on the data. Most of the transformations need to be processed one after the other in a Data Pipeline. This framework allows configuring of such Data Pipelines and Transformations.
- For each Transformation Type, a corresponding Airflow DAG needs to be defined. This DAG contains the exact Data Processing tasks to be performed for the Transformation.
- A Data Pipeline is triggered by the user using a python script. The trigger is received and stored in a "Read Pub Sub Topic".
- Cloud scheduler periodically triggers a "Read cloud function". This will read pending triggers in the Read Pub Sub Topic and store them in Datastore.
- Cloud scheduler periodically triggers a "Schedule Cloud Function". The Schedule Cloud Function will schedule the next Transformation pending in the Datastore, based on a Scheduling Algorithm. To schedule the Transformation, an appropriate request will be sent to Cloud Composer.
- The cloud composer will invoke corresponding Airflow Directed Acrylic Graph (DAG). The DAG in turn will invoke the appropriate underlined tasks configured for the transformation. This is where most of the Data processing takes place.
- On completion of the DAG, a Success or Failure message will be sent to "Status Pub Sub Topic".
- The "Status Cloud Function" will be triggered due to this message and it will update the Transformation status in the Datastore.

5.2 Checks and Controls

- The Data Pipelines which are made of Transformations are stored in a configurable repository. Thus, any number of Pipelines can be developed based on the need and can be integrated with configuration changes alone.
- The Transformation types are configurable. For each Transformation type a corresponding DAG needs to be developed. Many different types of Data processing tasks can be executed through the Airflow DAG.
- The Data pipeline requests have a Priority set from 1(Highest) to 7 (Lowest). This ensures that resources are made available to urgent tasks first. Low priority tasks are processed when the resources are less loaded so that there is no unnecessary ramp up or overloading of common infrastructure. The user can disable scheduling of tasks of any priority at run time based on business need.
- The Data Pipelines will have a budget assigned to them. If a request exceeds the allocated budget, it will be automatically skipped by the framework.
- Maximum number of simultaneous transformations of each type is configurable. This can be tuned with corresponding ramp up / ramp down in shared infrastructure resources like a data source. With a ramp up in infrastructure, more tasks can be run in

parallel reducing the turn-around time; however, this will entail additional infra cost. And vice versa for ramp down. Thus, the cost vs time adjustment can be done to achieve the desired optimization.

- There is provision for automatic reattempts of Failed Transformations for a configured number of times.
- There is provision to Cancel, Pause or Restart Data pipelines, if some problem is detected during processing.
- The speed of scheduling tasks can be adjusted by changing the periodicity of invoking "Read" and "Schedule" cloud functions in the Cloud Scheduler.
- Running tasks are periodically scanned to check if any task is taking longer than expected and is flagged of in an alert email.

5.3 Process Flow

- Triggering a Data Pipeline - The user uses a python based script to trigger the data pipeline. This will send a message to the "Read Pub sub Topic" using Google cloud API for Pub Sub. The message will contain:
  - Priority – 1 (Highest) to 7 (Lowest)
  - Data Pipeline Type – From a configured list of Data Pipeline types.
  - Command - Start / Cancel / Pause / Restart
  - Data Pipeline Id – Not required for Start Command, it will be system generated.
  - Data Pipeline Configuration - Json Payload corresponding to the Data Pipeline
  - Estimated Cost and Duration
  - Earliest Start Datetime – For prebuilt schedules
  - Logging info like username, current time
- To control access, a service account is created which has access to the "Read Pub sub Topic". The trigger script user provides the credential json file of this service account to send the request. This ensures proper authorization.
- Read a Data Pipeline request - Cloud scheduler triggers the "Read Cloud Function" periodically. The function reads and acknowledges pending messages in the "Read pub Sub Topic". The message details will be stored in datastore table "Data Pipeline" with Status = Submitted. If there is no Data Pipeline Id provided, the system will generate and assign a unique Data Pipeline Id.
- Schedule a Data Pipeline - Cloud scheduler triggers the "Schedule Cloud Function" periodically. It reads the status of all transformations and schedules them for execution. The decision regarding which transformations to schedule is done using a Scheduling Algorithm described in Table 1. To schedule a transformation, appropriate request is sent to Cloud composer. The Cloud Composer has a DAG configured and deployed for each transformation type. The "Schedule Cloud Function" invokes this DAG to schedule the processing of a transformation. The DAG request also includes the transformation configuration json.

Table 1: Scheduling Algorithm

Get the list of Data Pipelines where Command = Cancel / Pause and Status = Submitted
For each Cancel / Pause Command
Get the Data Pipeline with the given Data Pipeline Id Change the Data Pipeline Status to = Cancelled / Paused Change the Command Request Status to = Processed
Get the list of Data Pipelines where Command = Restart and Status = Submitted
For each Restart Command
Get the Data Pipeline with the given Data Pipeline Id Change the Data Pipeline Status to = Processing Change the Command Request Status to = Processed
Set Priority (P) = 1
While P <= 7 and the Priority level is not disabled
Get the list of Data Pipelines where Priority = P and Status = Submitted/Processing and "Earliest Start Datetime" after current datetime (Ordered by Submission time)
For each Data Pipeline ID (DPID)
If the Pipeline exceeds Budgeted cost, skip the pipeline
Get the Transformation where Data Pipeline ID = DPID and status = Processing / Failed
If Failed Transformation is found and number of attempts are below "maximum allowed" and number of transformations for this type are within configured limits
Schedule above transformation by invoking its DAG Update the Transformation Status to = Processing Increment the no of attempts by 1
If No Transformation is found
Get the next Transformation in the Data Pipeline
If number of jobs for this transformation type are within configured limits
Create an entry for the next Transformation with status = Submitted Schedule above transformation for processing by invoking its DAG Update the Transformation Status to = Processing Increment the number of attempts by 1
If this is first transformation in the Data Pipeline
Set the Data Pipeline Status = Processing
Increment the Priority (P) by 1

- Process a transformation - The Cloud Composer runs the invoked DAG in Airflow. In Airflow, a DAG (Directed Acyclic Graph) is a collection of all the tasks you want to run, organized in a way that reflects their relationships and dependencies. Airflow provides operators for many common tasks, including: Bash command, Python function, Sending Email, HTTP Request, Execute SQL commands on various databases like Cloud Bigtable. It can also run many more specific tasks like Docker, Dataproc job and Dataflow job. All these tasks are included in the pre-designed DAG for each transformation. On completion of the DAG, a success or failure message is sent to the "Status Pub Sub Topic".

- Scheduling Algorithm - The algorithm which is used to decide which transformations are to be scheduled for processing by the "Schedule Cloud Function" is described in Table 1
  The scheduling algorithm is a key component of the framework. The algorithm is based on "Priority Scheduling" and uses several checks and controls.
  - It schedules the pipelines based on Priority 1 (highest) to Priority 7 (lowest). Within the priorities it uses FCFS (First come first serve).
  - Priorities disabled by user are skipped.
  - Pipelines scheduled for future are skipped. These are pre-built schedules.
  - Pipelines exceeding budget are skipped. Revenue earning pipelines can be setup with unlimited budget.
  - Maximum number of jobs for each transformation type is restricted through configuration to avoid system overloading.
- Update transformation status - The "Status Cloud Function" will be triggered whenever a message on the "Status Pub Sub Topic" is received. This cloud function will update the transformation status in the Datastore.
  - If the transformation is successful, its status is set to = Success
  - If the transformation is successful and it is the last transformation in a Data Pipeline, the status of the Data Pipeline is set to = Success
  - If the transformation is failed, its status is set to = Failed
  - If the transformation is failed and number of attempts are equal to maximum allowed, the status of the Data Pipeline is set to = Failed
- Long Running Transformations Alert – Cloud scheduler triggers the "Monitor Cloud Function" periodically. It checks the duration of running tasks against estimated duration and sends an alert email for long running jobs.
- Transformations Status Dashboard –
  The status of all Data Pipeline requests, and their underlining transformations can be viewed on the Datastore page of the Google cloud console. Simple queries can also be run to filter the transformations based on its attributes like priority, status.
  A web based customized UI dashboard can also be developed to view this status. It can also be used for maintenance and backend updating of the Transformation and Data Pipeline status in the datastore by the system administrator.

5.4 Repositories for Reuse

- Data Transformation Repository - A DAG is defined for each Data Transformation. The DAGs are python scripts which run in Airflow environment. A name is assigned to the DAG and its related information is stored in a json configuration file. This file along with the corresponding airflow DAG script, forms the repository of all Data Transformations. The data engineers will search through this repository before creating any new transformation to maximize reuse.

- Data Pipeline Repository – A Data pipeline has a simple structure and is made of a fixed sequence of Data Transformations. All conditional flows are handled inside the Data Transformations. The list of all Data pipelines is stored in a json configuration file. The data engineers add new entries to this file whenever a new data pipeline is required.

5.5 Data Model

Datastore kinds (tables) required for the framework are mentioned in Table 2.

Table 2: Datastore Table Fields

Data Pipeline Table	Transformation Table
Data Pipeline Id	Data Pipeline Id
Data Pipeline Type	Transformation Type
Status	Status
Command	Number of Attempts
Data Pipeline Configuration	Transformation Configuration
Estimated Cost	Estimated Cost
Estimated Duration	Estimated Duration
Submission Datetime	Submission Datetime
Last Update Datetime	Last Update Datetime
Updated By	Updated By
Earliest Start Datetime	

5.6 Public Cloud Compatibility

The architecture above is described using GCP. Similar and equivalent tools and services available in AWS and Azure platforms are mentioned in Table 3. This framework can be implemented in AWS and Azure using these services.

Table 3: Equivalent Tools and Services

GCP	AWS	Azure
Cloud Function	AWS Lambda	Azure Function
Cloud Scheduler	AWS Lambda Rule	Function Schedule
Pub / Sub	Pub / Sub	Azure Service Bus
Cloud Composer + Airflow	Airflow webserver + AWS EMR	Bitnami Apache Airflow
Datastore	DynamoDB	Cosmos DB
Cloud Storage	Amazon S3	Blob storage
Service Accounts	Service Accounts	Service Principal

5.7 Security

- Service Accounts are used to control access to the various parts of the system. For triggering a request on the "Read Pub sub Topic" a Service Account is created. And only this service account can send message to the Topic. The access credential json file of this service account is shared with the users who are authorized to submit these requests.
- Another service account is created for the Cloud functions (Read, Schedule, Status and Monitor) used in the automation. These cloud functions are setup to run under the service account. All required accesses are provisioned to this service account. This ensures that users cannot misuse the access provided for running the cloud functions.
- Access Control Lists (ACLs) are used to control access to various components of the GCP like Datastore, Buckets, Cloud Scheduler, Pub Sub, etc. E.g. The buckets are used for various purposes like ingesting data, job configurations, storing intermediate outputs, maintenance etc. Need based ACLs are created in Cloud Identity and Access Management (IAM) and appropriate access is granted to them for the buckets.
- Cloud Firestore (Datastore) Security Rules allow control of access to documents and collections in the database. The flexible rules syntax allows creation of rules that match anything, from all writes to the entire database to operations on a specific document.

6 EVALUATION AND RESULTS

A prototype tool using the framework was implemented on Google Cloud Platform for an organization.

- Before the tool - The organization followed monthly budget allocation to different sub-teams. The performance of the data analysis activities before the use of the tool was recorded for 3 months. The team was using Airflow on Cloud composer for data orchestration. The data transformation tasks within the data pipelines were submitted on cloud manually. There were no priorities, budgets, and schedules set for the data pipelines.
- Using the tool - In the subsequent 3 months, the performance using the tool was recorded. The existing transformations built in Airflow were used as it is, to ensure exact comparison. Data pipelines were defined using these transformations and budgets were assigned to them. Priority 1 was reserved for external revenue earning requests, while priority 2 to 5 were used for internal data pipelines. There was no use case for priority 6 and 7. For routine runs, schedules were pre-built by sending advance schedule requests.

The performance evaluation results are summarized in Table 4, Table 5, and Figure 2. Following points (A to E) based on Table 4 indicate the effectiveness of the new framework.

A. Failed runs came down from 90 per month to single digit. The failures came down mainly due to –
   i. Avoiding overloading of shared cloud resources by the transformations
   ii. Reuse of tested transformations across pipelines
B. The Budgeted cost overrun which was in the range 8% to 15%; came down to 9% "balance" in 3rd month using new framework. Main contributors were –
   i. Reduced failed runs
   ii. Budget restrictions on Pipeline types
   iii. Priority setting for data pipelines
C. Average turn-around time for revenue earning P1 requests halved due to prioritization.
D. The P1 runs increased from average 10 per month to 35. This was due to –
   i. Reduced load on data engineers due to reduced total runs
   ii. Improved turn-around time resulted in additional requests from customers
E. Backlog of 4-5 days' worth runs came down to half days' worth runs. This can be attributed to prebuilt schedules.

Table 4: Automation Framework Evaluation

Parameter	Without Framework			With Framework		
Month	M1	M2	M3	M1	M2	M3
Cost Overrun (%) (P2-P5)	10	15	8	-1	-5	-9
Failed Runs	82	91	80	23	10	8
Successful Runs (P2-P5)	402	415	395	413	408	393
Successful Revenue Earning Runs (P1)	10	9	12	12	27	35
Total Runs	494	515	487	448	445	436
Avg Turnaround time (P1) (Business Hours)	16	17	16	8	8	9
Avg Daily Backlog (P2-P5)	95	89	84	9	10	5

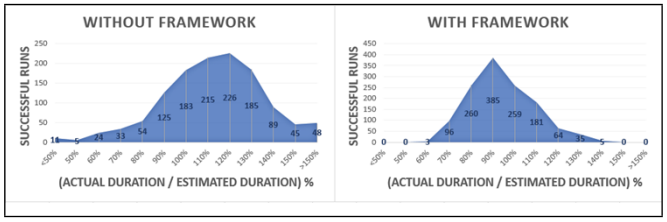

Figure 2: Duration Variance of Successful Runs

Table 5: Duration Variance Evaluation

Parameter	Without Framework	With Framework
Median	118 %	97 %
Mean	117 %	98 %
Standard Deviation	23.98	14.75
Coefficient of Variance	20.44 %	14.98 %

F. As seen in Figure 2 and Table 5 –

i. The median duration variance (Actual duration / Estimated duration), which was at 118% "without framework", has come down to 97% "with framework". This indicates better adherence to the duration estimates.

ii. The Coefficient of Variance "without framework" is 20.44 % indicating higher level of dispersion, while "with framework" is 14.98 % indicating lower level of dispersion. Hence the predictability of actual duration is better "with framework".

iii. The range "without framework" is from 30% to 197%. However, "with framework" range narrows down from 62% to 148%.

Out of 22 data pipelines defined, 8 have shared transformations with other pipelines. All transformations are added to the reuse repository which is a set of airflow DAGs. Building new data pipelines will be faster and less error prone, due to the reuse of these transformations

7 CONCLUSION AND FUTURE WORK

In this paper we have presented a "Cloud Data Pipeline Automation Framework" with several checks and controls to address challenges related to budget management, use-case prioritization, and reuse of transformations. The framework was evaluated on GCP and the results establish the effectiveness of the framework. Organizations may build in house applications using this framework or the Data Pipeline tools may consider incorporating this framework in their tooling.By using this automation approach, organizations would realize the true benefits offered by the "pay-as-you-use" cloud platforms. In future, we plan to verify the framework on AWS and Azure cloud platforms. We also aim to improve the estimation process for the Data transformations and pipelines as estimation accuracy continues to be a challenge.

REFERENCES

1. Michael R. Berthold, Nicolas Cebron, Fabian Dill, Thomas R. Gabriel, Tobias Kötter, Thorsten Meinl, Peter Ohl, Kilian Thiel, and Bernd Wiswedel. 2009. **KNIME - the Konstanz information miner: version 2.0 and beyond.** SIGKDD Explor. Newsl. 11, 1 (June 2009), 26–31. DOI:https://doi.org/10.1145/1656274.1656280
2. Maria A. Rodriguez and Rajkumar Buyya. 2017. **Budget-Driven Scheduling of Scientific Workflows in IaaS Clouds with Fine-Grained Billing Periods.** ACM Trans. Auton. Adapt. Syst. 12, 2, Article 5 (May 2017), 22 pages. DOI:https://doi.org/10.1145/3041036
3. L. Zeng, V. Bharadwaj, and X. Li, **"Scalestar: Budget conscious scheduling precedence-constrained many-task workflow applications in cloud,"** in Int. Conf. on Advanced Information Networking and Applications, vol. 0, Los Alamitos, CA, USA, 2012, pp. 534–541.
4. Karthik Raman, Adith Swaminathan, Johannes Gehrke, and Thorsten Joachims. 2013. **Beyond myopic inference in big data pipelines.** In Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '13). Association for Computing Machinery, New York, NY, USA, 86–94. DOI:https://doi.org/10.1145/2487575.2487588
5. Fei Cao, Dabin Ding, Dunren Che, Michelle M. Zhu, and Wen-Chi Hou. 2013. **Scheduling data processing flows under budget constraint on the cloud.** In Proceedings of the 2013 Research in Adaptive and Convergent Systems (RACS '13). Association for Computing Machinery, New York, NY, USA, 69–74. DOI:https://doi.org/10.1145/2513228.2513250
6. Omran Saleh and Kai-Uwe Sattler. 2015. **The pipeflow approach: write once, run in different stream-processing engines.** In Proceedings of the 9th ACM International Conference on Distributed Event-Based Systems (DEBS '15). Association for Computing Machinery, New York, NY, USA, 368–371. DOI:https://doi.org/10.1145/2675743.2776774
7. Mukund Deshpande, Dhruva Ray, Sameer Dixit, and Avadhoot Agasti. 2015. **ShareInsights: An Unified Approach to Full-stack Data Processing.** In Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data (SIGMOD '15). Association for Computing Machinery, New York, NY, USA, 1925–1940. DOI:https://doi.org/10.1145/2723372.2742800
8. Jürgen Cito, Philipp Leitner, Thomas Fritz, and Harald C. Gall. 2015. **The making of cloud applications: an empirical study on software development for the cloud.** In Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering (ESEC/FSE 2015). Association for Computing Machinery, New York, NY, USA, 393–403. DOI:https://doi.org/10.1145/2786805.2786826
9. Tarek M. Ahmed, Farhana H. Zulkernine, and James R. Cordy. 2016. **Proactive auto-scaling of resources for stream processing engines in the cloud.** In

Proceedings of the 26th Annual International Conference on Computer Science and Software Engineering (CASCON '16). IBM Corp., USA, 226–231.

10. Fábio Oliveira, Sahil Suneja, Shripad Nadgowda, Priya Nagpurkar, and Canturk Isci. 2017. **Opvis: extensible, cross-platform operational visibility and analytics for cloud.** In Proceedings of the 18th ACM/IFIP/USENIX Middleware Conference: Industrial Track (Middleware '17). Association for Computing Machinery, New York, NY, USA, 43–49. DOI:https://doi.org/10.1145/3154448.3154455
11. Yanzhe Cheng, Fang Cherry Liu, Shan Jing, Weijia Xu, and Duen Horng Chau. 2018. **Building Big Data Processing and Visualization Pipeline through Apache Zeppelin.** In Proceedings of the Practice and Experience on Advanced Research Computing (PEARC '18). Association for Computing Machinery, New York, NY, USA, Article 57, 1–7. DOI:https://doi.org/10.1145/3219104.3229288
12. Victor Giannakouris, Alejandro Fernandez, Alkis Simitsis, and Shivnath Babu. 2019. **Cost-Effective, Workload-Adaptive Migration of Big Data Applications to the Cloud.** In Proceedings of the 2019 International Conference on Management of Data (SIGMOD '19). Association for Computing Machinery, New York, NY, USA, 1909–1912. DOI:https://doi.org/10.1145/3299869.3320240
13. Mainak Adhikari, Tarachand Amgoth, and Satish Narayana Srirama. 2019. **A Survey on Scheduling Strategies for Workflows in Cloud Environment and Emerging Trends.** ACM Comput. Surv. 52, 4, Article 68 (September 2019), 36 pages. DOI:https://doi.org/10.1145/3325097
14. Mutaz Barika, Saurabh Garg, Albert Y. Zomaya, Lizhe Wang, Aad Van Moorsel, and Rajiv Ranjan. 2019. **Orchestrating Big Data Analysis Workflows in the Cloud: Research Challenges, Survey, and Future Directions.** ACM Comput. Surv. 52, 5, Article 95 (October 2019), 41 pages. DOI:https://doi.org/10.1145/3332301
15. El Kindi Rezig, Lei Cao, Michael Stonebraker, Giovanni Simonini, Wenbo Tao, Samuel Madden, Mourad Ouzzani, Nan Tang, and Ahmed K. Elmagarmid. 2019. **Data Civilizer 2.0: a holistic framework for data preparation and analytics.** Proc. VLDB Endow. 12, 12 (August 2019), 1954–1957. DOI:https://doi.org/10.14778/3352063.3352108
16. Simon Eismann, Johannes Grohmann, Erwin van Eyk, Nikolas Herbst, and Samuel Kounev. 2020. **Predicting the Costs of Serverless Workflows.** In Proceedings of the ACM/SPEC International Conference on Performance Engineering (ICPE '20). Association for Computing Machinery, New York, NY, USA, 265–276. DOI:https://doi.org/10.1145/3358960.3379133
17. Shirly Wang, Matthew B. A. McDermott, Geeticka Chauhan, Marzyeh Ghassemi, Michael C. Hughes, and Tristan Naumann. 2020. **MIMIC-Extract: a data extraction, preprocessing, and representation pipeline for MIMIC-III.** In Proceedings of the ACM Conference on Health, Inference, and Learning (CHIL '20). Association for Computing Machinery, New York, NY, USA, 222–235. DOI:https://doi.org/10.1145/3368555.3384469
18. D. Wu, L. Zhu, X. Xu, S. Sakr, D. Sun and Q. Lu, **"Building Pipelines for Heterogeneous Execution Environments for Big Data Processing,"** in IEEE Software, vol. 33, no. 2, pp. 60-67, Mar.-Apr. 2016 doi: 10.1109/MS.2016.35.
19. O. Dawelbeit and R. McCrindle, **"CloudEx: A Novel Cloud-Based Task Execution Framework,"** 2016 IEEE Globecom Workshops (GC Wkshps), Washington, DC, 2016, pp. 1-5. doi: 10.1109/GLOCOMW.2016.7848860.
20. Huiyan Cao and Chase Q. Wu. 2018. **Performance optimization of budget-constrained mapreduce workflows in multi-clouds.** In Proceedings of the 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid '18). IEEE Press, 243–252. DOI:https://doi.org/10.1109/CCGRID.2018.00039
21. Syed.Karimunnisa and Dr.Vijaya Sri Kompalli, 2019. **Cloud Computing: Review on Recent Research Progress and Issues.** In International Journal of Advanced Trends in Computer Science and Engineering, Volume 8, No. 2, March-April 2019, DOI: https://doi.org/10.30534/ijatcse/2019/18822019
22. Abhikriti Narwal and Sunita Dhingra, 2020. **Credit Based Scheduling with Load Balancing in Cloud Environment.** In International Journal of Advanced Trends in Computer Science and Engineering, Volume 9, No. 2, March-April 2020, DOI: https://doi.org/10.30534/ijatcse/2020/34922020
23. G. Kiruthiga, Dr. S. Mary Vennila, 2019. **An Enriched Chaotic Quantum Whale Optimization Algorithm Based Job scheduling in Cloud Computing Environment.** In International Journal of Advanced Trends in Computer Science and Engineering, Volume 8, No. 4, July-August 2019. DOI: https://doi.org/10.30534/ijatcse/2019/105842019
24. **Apache Airflow**, https://airflow.apache.org/
25. **"Cloud Function Documentation"** Google Cloud Platform https://cloud.google.com/functions/docs
26. **"Cloud Scheduler Documentation"** Google Cloud Platform https://cloud.google.com/scheduler/docs
27. **"Cloud Pub/Sub Documentation"** Google Cloud Platform https://cloud.google.com/pubsub/docs
28. **"Cloud Datastore Documentation"** Google Cloud Platform https://cloud.google.com/datastore/docs
29. **"Cloud Composer Overview"** Google Cloud Platform https://cloud.google.com/composer/docs/concepts/overview
30. **"Cloud Storage Documentation"** Google Cloud Platform https://cloud.google.com/storage/docs