Shining Rays of Light & Savings on Cloud Portfolios: An Important Advance!

Richard J. Krempasky - richard.krempasky@gmail.com

Kenneth S. Rhodes - KRhodes@technomics.net

R. Alex Wekluk - AWekluk@technomics.net

International Cost Estimating and Analysis Association (ICEAA) Conference

5/16/2023

Abstract

Existing cloud cost estimating and pricing tools have at least two significant drawbacks: 1) they require many assumptions, therefore a detailed understanding of current/future architectures and 2) they do not provide insights required to manage costs. Technomics developed a groundbreaking alternative that will revolutionize how cloud costs are estimated and managed. This paper describes our parametric-based toolset, which enables cloud lifecycle cost estimation, cost reduction and efficiency analysis.

Track: IT & Cloud Computing Track (IT06)

Keywords: Cloud Computing, Parametric Estimating, Cost Optimization

Table of Contents

Table of Figures

1. Introduction

Survey feedback from the cloud user community [9] consistently highlights two top cloud adoption challenges facing organizations: 1) accurate cost forecasting of the requirements and 2) engineers optimizing cost. The tools and techniques presented in this paper provide a framework for program managers, cost analysts and engineers to fully optimize the cost, and therefore overall mission performance, of their cloud migration, adoption, and usage. This paper builds on a 2022 Joint IT and SW Cost Forum (JITSWCF) presentation [15] and addresses practical methods for forecasting and optimizing cloud costs at any stage in the lifecycle.

As shown in Figure 1, the selection of a cloud cost estimating, forecasting, or optimization method is highly dependent on the phase within the development (Dev) and operations/sustainment (Ops) cycle, which in turn determines the data available to employ the methods.

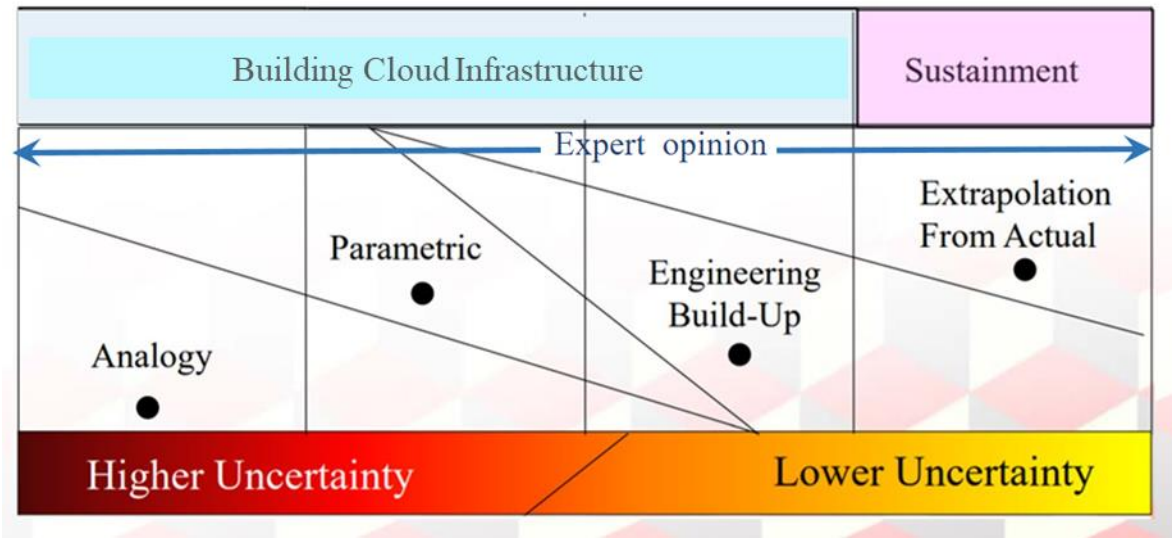

Figure 1: Cloud Cost Estimating Methods

Early adoption of cloud infrastructure based on initial requirements results in higher uncertainty or variability in the range of potential outcomes in development; whereas estimates based on extrapolation of actual cloud purchases in operations produce forecasts with lower uncertainty [7]. The best approach to cloud cost estimating depends on when the estimate is needed and the maturity of the functional requirements.

This paper discusses two tools: Multi-Service Cloud Estimating (MUSCLE) model and Framework for Lifecycle Execution of Cloud Systems (FLECS). MUSCLE, which is designed to address scenarios where a parametric approach is most relevant, is a modular and scalable cloud estimating model that enables evaluation of infinite cost scenarios, trades, and technical baselines. FLECS is an integrated set of tools that cohesively optimizes and manages cloud environments and solutions in Ops. In recognition of the differences inherent in earlier versus later in the lifecycle cloud analysis, FLECS is designed to exploit actuals-to-date via the extrapolation cost estimating technique, which necessarily requires analysis of real-time cost and consumption measures.

It is important to note that this paper does not address cloud environment set-up and initial implementation

2. MUSCLE and FLECS Differentiation

MUSCLE and FLECS deliver differentiated capabilities.

Let us discuss MUSCLE first. Other cloud cost estimating and pricing tools require a detailed understanding of current and future architectures, including assumptions for 20-150+ technical parameters/inputs. In contrast, MUSCLE's main input requirement is a standard sizing metric related to Key System Attributes or Key Performance Measures. The tool's parametric approach to estimating cloud lifecycle costs leverages known programmatic requirements (e.g., apps, data/video feeds, satellite connectivity), facilitating lifecycle cost management. Through multi-variate regression analysis and a program requirements assessment, the cost model provides a linkage between customer technical/programmatic requirements and vendor pricing. In other words, MUSCLE produces a cost estimate that is directly tied to requirements and available early in the software development process before technical design details are available. By analyzing the key cost drivers in overall cloud architectures and pricing models, MUSCLE provides cost estimators a data-driven method for estimating cloud costs.

Next, let us discuss FLECS's differentiated capability. Subsequent to cost estimating activities during cloud migration or initial adoption, the operational phase also provides

ample opportunity for improved cost forecasting. In execution, cloud usage is often managed with a stagnant budget based on the original specification and is not optimized for most efficient utilization as program requirements change. FLECS consists of a cloud analytics dashboard designed to enable insights and understanding that facilitate cost reduction and efficiency. FLECS's supporting tools include backend data ingest via automated workflows and data storage to inform decisions related to usage and utilization by account/project, engendering increased resource efficiency. Informative dashboards provide utilization and usage metrics, advancing continual real-time optimization by monitoring expenditures and managing cloud costs within budget allocations. Linking the usage, utilization and pricing data together provides unique cloud management insights and deeper understanding of the cloud framework, enabling decisions that scale smartly to accommodate growth.

To address the cloud adoption challenges, MUSCLE and FLECS aim to actively monitor cloud costs to measure performance and reduce waste. The tools' outputs dashboards enable decision makers to understand costs and their relationship to requirements. In fact, early implementation successes of both methods throughout the DevOps cycles suggest that integration of tools and data to better understand and manage cost drivers greatly improves out-year and annual budget forecasting.

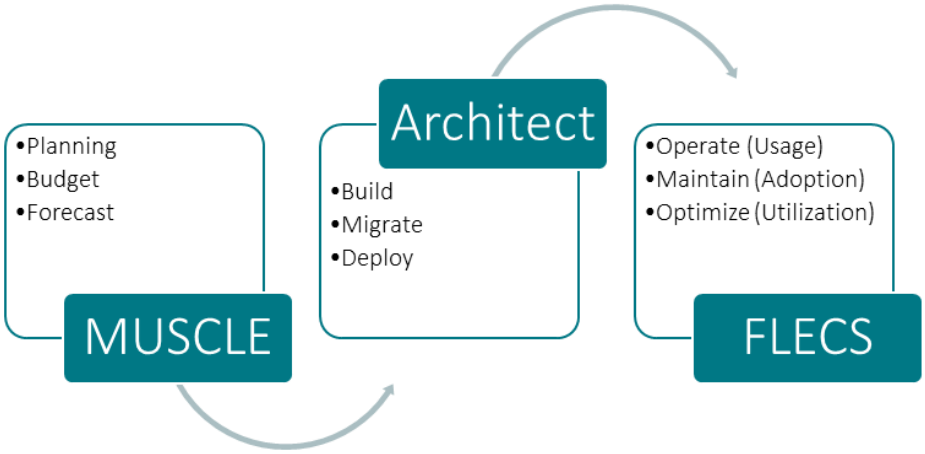

Figure 2: MUSCLE and FLECS Feedback Loop

Figure 2 depicts the feedback loop of operational data/metrics (FLECS) back into forecasting tools (MUSCLE) to further refine and improve the parametric approach. Using a combination of realistic cost predictions and active monitoring, cloud usage can maximize mission effectiveness by identifying specific actions that can be taken reduce costs or inefficient usage.

Figure 3 visualizes a scenario where operational costs exceed initial cost estimates. With active monitoring of cloud costs, FLECS can identify under-utilized resources for immediate cost savings, as well as capture real-time cloud pricing for the baseline architecture that can improve the forecasting capabilities of MUSCLE.

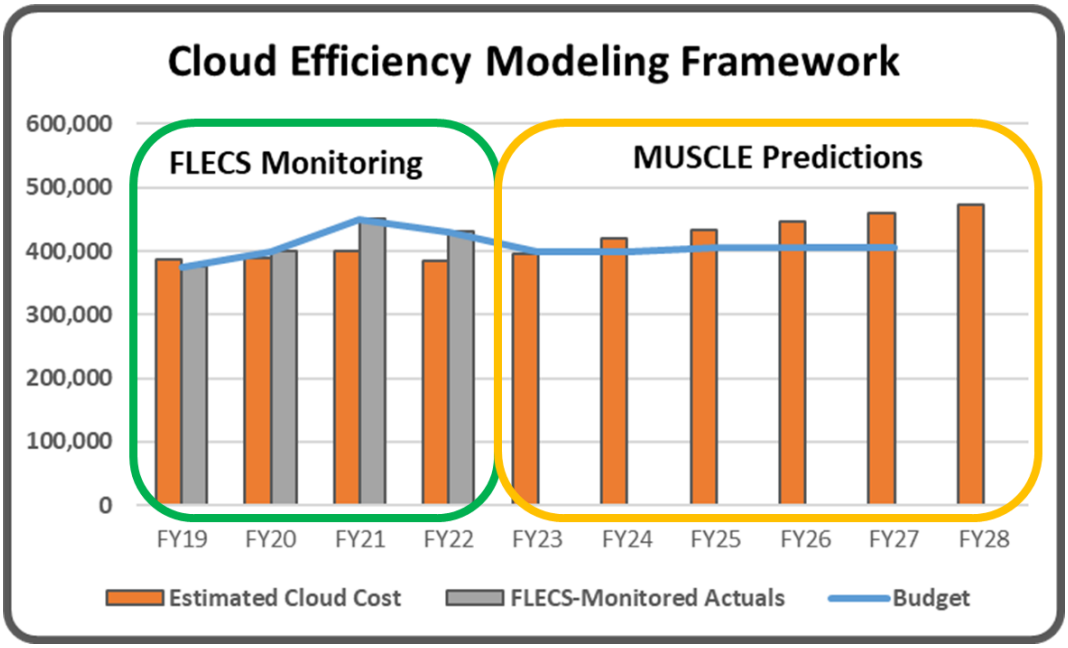

Figure 3: Cloud Efficiency Modeling Framework

The feedback loop and improved cloud cost forecasting of MUSCLE and FLECS address a capability gap with existing tools available online that require mature program definition.

In contrast to other cloud cost estimating tools, MUSCLE produces long-term estimates based on programmatic requirements (ex. # of Galleries, duration of custody), is flexible to accommodate requirements changes over time, and accommodates wide-range of

program applications given historical data on analogous systems. FLECS provides cost optimization (i.e., cost management) that uncovers possible opportunities to reduce costs by scaling resources, eliminating unused resources, using automation, and improving processes. Efficiencies gained by analyzing service and cost limits, volumes, usage and efficiency goals can result in common policies and governance that optimize the operating environment.

3. MUSCLE Model – Estimating Costs in the Future

3.1. Philosophy

Cloud usage is predicted to grow to nearly $600B in 2023, more than 20% growth from 2022 [(1)]. With this huge and bourgeoning market, good cost estimating takes on a critical role in decision-making. Survey after survey identified reducing waste and inefficiency as the key focus areas for businesses, meaning proper scaling and forecasting is of paramount importance [(10)(11)]. Technomics' clients expressed a need for a reliable prediction of cost, and in many cases had multiple scenarios for how they would execute. Most of the cloud calculators on the market tend to estimate for the current month, and require a very exact technical solution or they will not run. This technical solution includes tens of inputs that may or may not be known at the time. Amazon Web Services (AWS) pricing calculator requires 70+ inputs for their estimator [(2)]. Alternatively, independent cost and execution models (e.g., Gartner, Government Cost Groups) are more flexible, accommodating requirements growth, pricing changes, and enterprise efficiencies based on the use of vendor-agnostic cloud pricing data collected over a long period of time. Despite requiring fewer inputs, the independent cost models necessarily rely on mature technical definition (i.e. inputs) of the key cost drivers (Compute (EC2), Relational Database Service (RDS), and Storage (S3, S3-IA, Glacier, EBS)).

MUSCLE is architected to be a multi-service estimating model, allowing estimates for multiple cloud providers, and is also a multi-*phase* model allowing for estimates before or during Phase A, Phase B or Phase C in a program lifecycle.

MUSCLE was designed and built to help all layers the organization: from engineers up to enterprise decision-makers. The tool is a modular and scalable cloud estimating model, allowing infinite cost scenarios, trades and technical baselines to be examined in one place. For the team to really help drive change in the industry, we wanted to highlight the functionality and features of MUSCLE to show what is possible. It is our hope that by socializing and collaborating, we can help other help other model builders find effective ways of forecasting their agencies' costs.

3.2. Model Description

MUSCLE is architected to be maximally informative to decision makers, but can also speak the language of the cloud engineer. The team deconstructed cloud services into independent variables and ran multi-variate regressions. Those regressions yielded credible, defensible algorithms that, as discussed earlier, require much fewer inputs than other cloud models. MUSCLE also provides the capability to select specific known cloud services and calibrate those to accounting data, if available for an existing program. Our team also decomposed technical requirements on several programs to create algorithms with programmatic inputs, such as number of images, number of hosted image galleries needing to be processed, or technical specifications of an image chain for Intelligence, Surveillance and Reconnaissance (ISR) collectors. This linkage to programmatic requirements helps our team communicate the trade space to program management and decision makers in their vernacular.

The model also provides the capability to phase cost estimates into the future via consideration of: inflation, cost improvement from technology maturation and selectable compounding usage increases over time. Cost estimates can be calibrated to available historical cost or pricing quote data, saving the time required to specify a technical solution that already exists, which today entails an estimator selecting values for the seventy inputs required by online calculators.

Potential cost scenarios that can be modeled in MUSCLE are predictive pricing by year (or potentially by month, as desired), varying schedules for each instance or batch of instances, management of an entire portfolio of programs, Pareto analysis of a frontier of solutions, and even cost as an independent variable.

Regression analysis, along with a modular and scalable architecture, discriminate MUSCLE from other existing solutions, and helps our team to rapidly and consistently model trade space and future costs.

3.3. Cost Data Analysis

In parallel with model development, the team analyzed several large sets of pricing data from AWS and Azure. Many independent variables were assessed to determine the

cost drivers that influence the greatest proportion of cost, with the goal of minimizing user-required inputs to design a cloud architecture. Figure 4 includes a snapshot of some of the independent variables tested for compute. Obvious-looking outliers are generally explainable by some other variable. For example, in the Independent Variable (Ind Var) 2: vCPUs graph, the three stratified points around 450 vCPUs have three different memory allocations. The team had 430 data points for compute in the initial analysis run, and 135 for database. In that dataset, location ended up being much less significant a driver than memory and vCPUs. A perfect algorithm representing all variables would end up requiring 70+ input parameters and an order of magnitude more data points; but, the goal was not to recreate service provider calculators. Instead, the goal was to predict the cost of a technical solution and quantify the uncertainty at that phase of the program.

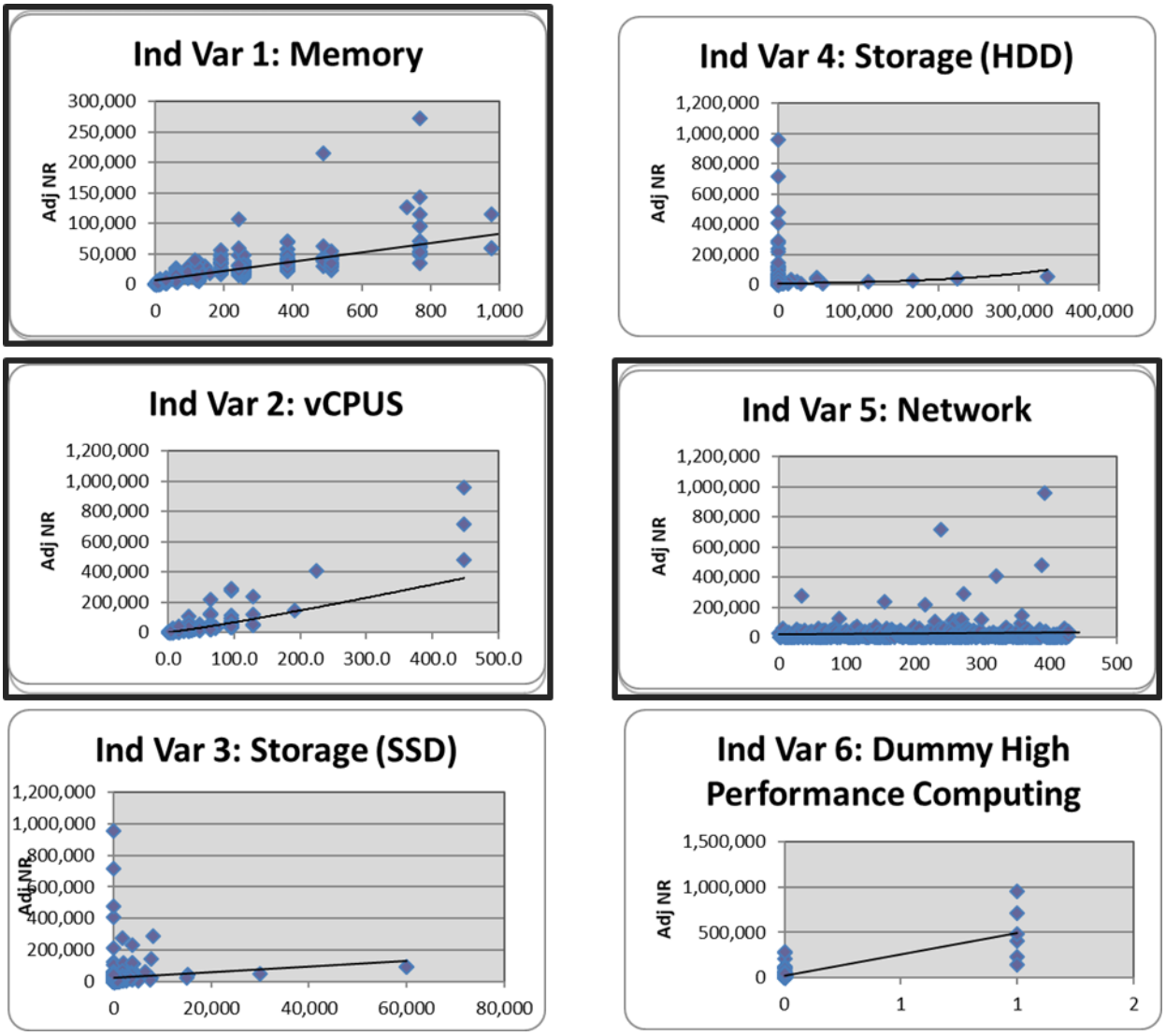

Figure 4. Cloud cost data analysis showing six different independent variables.

3.4. Regressions of Cost Data to Build CER Library

The backbone of MUSCLE is a robust and adaptable Cost Estimating Relationship (CER) library. This library allows almost any form without limits on number of independent variables. With that in mind, the team ran multi-variate regression analysis on the EC2 and on RDS pricing data to create a family of CERs. Over two hundred

regressions were run on that batch of data using different combinations of variables. Some of the variables considered were: quantity of vCPUs, solid-state storage capacity, hard disk storage capacity, memory, operating system, and network speed. Additionally, dummy variables were considered to stratify the data for subjective criteria like high performance computing. Figure 5 shows the actuals versus estimate plot for one method. It's easy to see some of the extreme outliers at higher costs, likely evidence of heteroskedasticity. This paper is intended to be a proof of concept, so we will not delve into acceptance criteria for the individual methods.

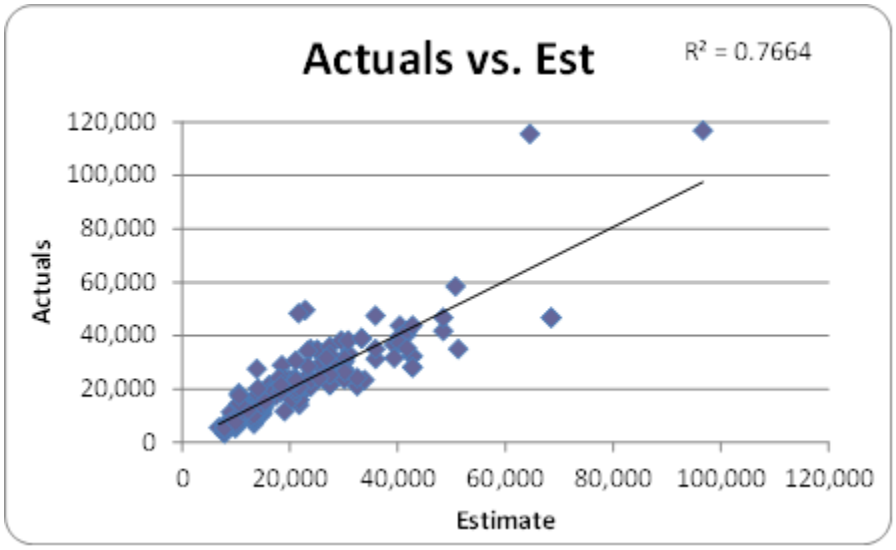

Figure 5. Actuals versus estimates plot for one of the regressed methods.

The team binned the data with histograms on all of the independent variables to identify potential outliers that may need additional explanatory variables. The method library size is not limited in any way, so creating additional CERs to represent niche instance types for any of the service providers is a good way to address outliers. The team derived multiple CERs with Adjusted R^2 of greater than 75% and Coefficient of Variation of less than 30%. These and other regression statistics (e.g., Degrees of Freedom, number of data points, Standard Error) are maintained in the CER library in addition to the algorithms so Monte Carlo analysis might be run in the future. The method

development process run is done with a person ***in the loop*** and an emphasis on causation instead of just correlations, in contrast to machine learning algorithms.

The team continues to identify more data from additional service providers for use in expanding the CER library to support more comprehensive future estimates.

3.5. Model Schematic and Order of Operations

As indicated in Figure 6, MUSCLE is divided into four main sections: 1) model assumptions, global inputs and parameters; 2) the Cloud Service Cost Model; 3) Enterprise Loaders; and 4) Outputs.

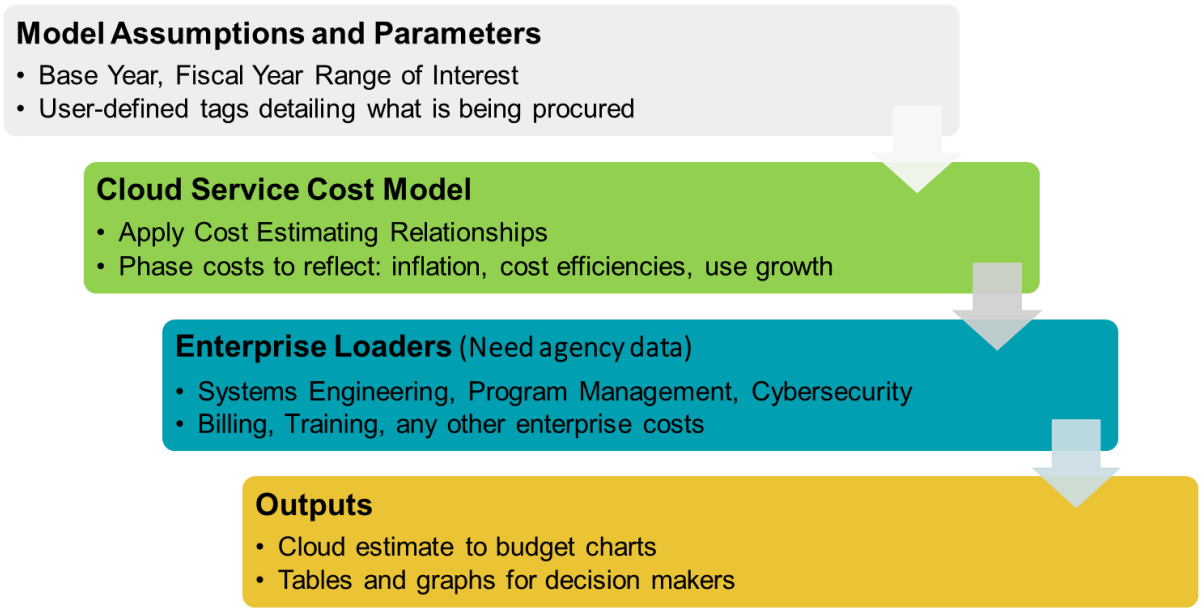

Figure 6. MUSCLE model schematic.

Model assumptions include the base year of the estimate and input data, the Fiscal Year range of interest, and user-defined tags. User-defined tags include organizational elements, such as a Work Breakdown Structure (WBS), program or domain. Other user tags are for the cost scenario; for example, a primary and secondary cost scenario or some kind of trade space describing an alternative procurement strategy.

The Cloud Service Cost Model includes user definition of requirements and/or what instance types are being procured. CERs are then applied to generate estimates for each end item or instance. Costs can be calibrated to actuals or to an online calculator if an exact point solution is already determined. This section of the model includes phasing that reflects inflation, cost efficiencies over time and service usage increases. The model is modular and scalable, and individual end items or services can be copied and changed to generate cost scenarios or to determine sensitivity to input variables.

Enterprise loaders are costs to the enterprise that tend to scale based on the amount of services performed. Loaders vary based on agency, and typically cover Systems Engineering, Program Management, Cybersecurity and also billing, training or enterprise architecture. These are difficult to characterize universally, since strategies for cloud services varies so greatly by agency. Some procure all services together to save costs, some have programs buy individual services as needed.

The final section of the model is the Outputs, which are provided via a standardized dashboard, as well as customizable tables and graphs for decision makers. Outputs also can be connected to FLECS, discussed in later sections, to characterize current usage.

Underpinning the entire model are the cloud pricing data, historical data from agencies and the robust and growing CER Library.

3.6. Inputs Required for Cost Modeling

MUSCLE versatility comes from a modular and scalable design. The scalability required inclusion of robust configuration management. For example, the model includes the ability to manage schedules at a top level, so it is easy to maintain a large number and types of instances. The model makes it obvious when a standard configuration is overridden in favor of a tailored solution.

Figure 7 depicts how inputs are used in the model. First, define an instance as compute, database or storage. Next, define the schedule and high-level requirements. The instance is further specified if it is Structured Query Language (SQL) for databases or, in the case of storage, glacier, including identifying access requirements. Lastly,

determine the independent variables such as amount of storage, quantity of vCPUs and memory. If programmatic CERs are selected, those would replace any technical variables.

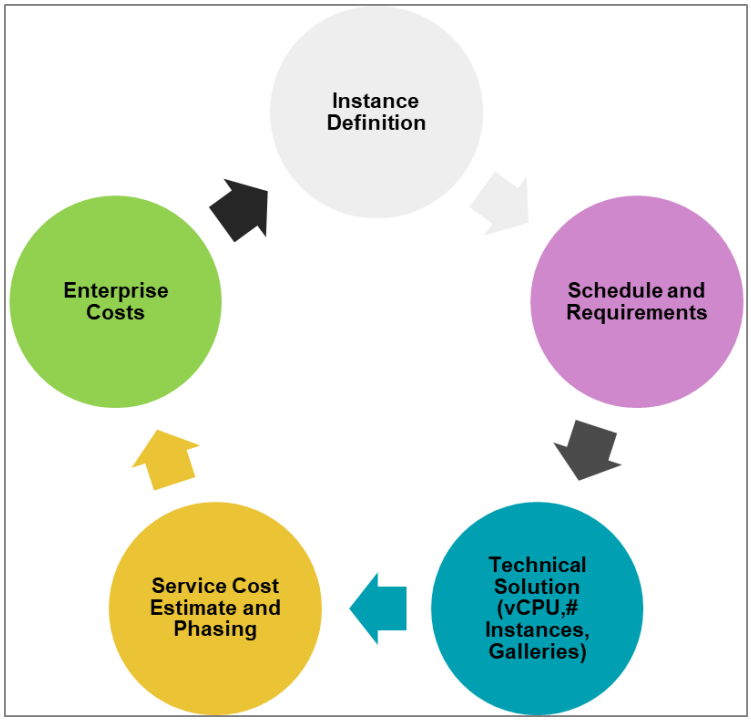

Figure 7. Model inputs and logic flow.

Based on these selections, a point solution is established and the model attempts to automatically time-phase the estimate via consideration of inflation, usage increase and cost efficiencies over time.

Finally, enterprise costs are added to the service costs. Any or all of the instances can be replicated easily and quickly for cost ‘what if’ scenarios.

3.7. MUSCLE Sample of Inputs Required

To date, the model has been used for about five estimates, and the framework was adapted to create a comprehensive integration model for hardware and software IT estimates. This versatility is enabled by the robust CER library, VBA code, and the modularity inherent in the design. Figure 8 shows a portion of the model, demonstrating

how easily inputs can be edited in the yellow cells. The screenshot shows compute instances, but it is just as easy to estimate storage, database, or virtually any hardware/software type. Independent variables are populated in the same cells regardless of instance type

			Cost Estimating Relationship (CER)	Instance X-Check		Schedule		Independent Var 1		Independent Var 2	
Item	Cost Type	Instance Qty	CER Default	Instance	Yrly Cost	Start Date	End Date	Var 1	Input	Var 2	Input
Production Controllers	Compute	1	EC2 General	c5.4xlarge	5,957	10/1/2020	9/30/2025	Memory (GB/vCPU)	2	vCPUs	16
Production Templates	Compute	4	EC2 General	c5.2xlarge	2,978	10/1/2020	9/30/2031	Memory (GB/vCPU)	2	vCPUs	8
Testing Controllers	Compute	3	EC2 General	c5.4xlarge	5,957	10/1/2020	9/30/2031	Memory (GB/vCPU)	2	vCPUs	16
Testing Templates	Compute	1	EC2 General	c5.2xlarge	2,978	10/1/2020	9/30/2031	Memory (GB/vCPU)	2	vCPUs	8

Figure 8. Sample of model inputs and user interface.

Our team also translated cloud requirements into unique customer-specific variables in the vernacular of the program manager as opposed to the cloud engineer. For example, in one project, program management understood how many pictures would be processed, based on number of users. Our team helped translate the cloud technical requirements for number of users to the quantity and type of cloud services. This helped management see the cost of their solutions based on real-world scenarios that tied to programmatic documentation, engendering much greater insight and understanding than is possible with technical documents such as engineering interface designs.

3.8. MUSCLE Outputs

Figure 9 shows a snapshot of an automatically-generated model dashboard for a sanitized, real-world use-case. The dotted line depicts a cost scenario showing an uptick in travelers for a system after COVID-19 restrictions are lifted. The outputs shown are easily reconfigurable for any sponsor or agency standards. We believe the parametric approach to the model minimizes error, since typical engineering buildups often neglect unknown costs and risks.

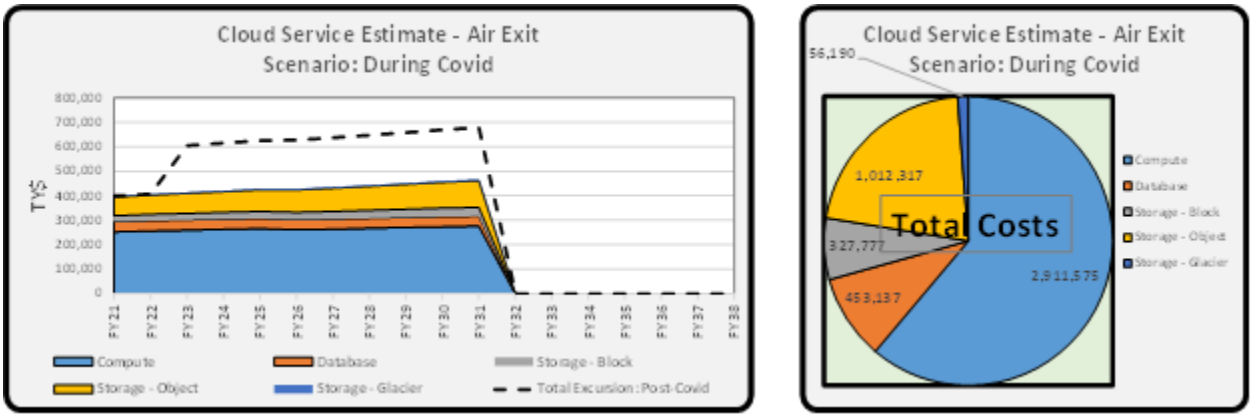

Figure 9. Sample MUSCLE dashboard.

The team determined the uptick in travelers for the system based on program guidance, then translated that guidance into an acceptable technical solution, as stated in the previous section. The sole independent variable for that CER was the number of travelers.

3.9. Lessons Learned from Market Survey and Designing MUSCLE

It is our hope that the methods and approaches presented in this paper will be adopted across the cost community. Our team learned a great deal during MUSCLE development – from ways to estimate cloud costs, to model shortcomings gleaned from our market survey that should be avoided. Any models for estimating cloud costs should include the ability to:

- Run cost scenarios, adding modularity to the system.
- Include unlimited instance types, each estimated at their appropriate level, adding scalability to the model. These instance types should be easily replicable to improve modularity.
- Run different types of CERs, some with many independent variables for well-known instances, but also some with few independent variables for poorly-defined or pre-Phase A instances.
- Employ CERs that can translate requirements to independent variables; for example, number of end users rather than a quantity of vCPUs. This becomes

critical for moving up the leadership decision chain and can facilitate enterprise-level leaders' use of informed non-technical assumptions.

- Include a scheduling component to turn on/off services as needed, preferably in a consolidated place in the model. Our team relied upon a user-defined default schedule for services that could be overridden; overrides call immediate attention to any differences from defaults.

Some shortcomings that we encountered in other models that should be avoided are including too many inputs and not including a forecasting component.

We discussed the drawbacks of requiring so many inputs, and how requiring over-specification can be detrimental to completing an estimate. Scheduling takes on a critical importance when tying estimates to requirements and for tuning existing cloud procurements, as monitored and optimized by FLECS.

4. FLECS – Active Monitoring to Get the Most Out of Cloud Services and Budgets

4.1. FLECS Overview

Cloud providers such as Amazon Web Services (AWS), Microsoft Azure, Oracle, and Google enable their customers to have access to not only the latest technologies but the ability to scale at a previously unheard-of speed and size. This ability to scale can quickly get out of hand and lead to explosions in costs and usage. However, users have the ability to access minute details in their cloud deployments and monitor every aspect of their accounts. Users need to implement a framework to ingest, analyze and act on the data for their cloud accounts to fully control their technology within budget. Usage and cost should be monitored constantly, and those costs can be fed back into MUSCLE to help tune budgets as the technology progresses. The same data that goes into MUSCLE also helps inform day-to-day operations as well as overall architectural reviews of user accounts.

4.2. Data Ingest

Cloud accounts generate massive amounts of data on each transaction. Each individual instance can be billed, in some cases, down to the second. A single user can have multiple applications on one account plus multiple accounts within their purview. Since gathering all the data can be an arduous task, application of the proper tools is paramount.

Figure 10 shows the cost query flow. To start, scripting languages such as Python or JavaScript provide a quick tool to programmatically access each account and pull the necessary data from them. As an example, AWS utilizes the Boto3 Software Development Kit (SDK) to create, configure, and manage AWS services, such as Amazon Elastic Compute Cloud (Amazon EC2) and Amazon Simple Storage Service (Amazon S3). The SDK provides an object-oriented Application Programming Interface (API) as well as low-level access to AWS. The idea of an SDK is not unique to AWS and is available from all of the other cloud providers. Limitations with scripting come in

the form of automation and scale. Typically, scripts are configured and run on a unique computer with a unique environment. To enable the scale and automation needed to work with multiple accounts over multiple environments different Extract, Transform, and Load (ETL) software is needed. One such software package is the open-source platform Apache Airflow, which orchestrates batches of workflows that can chain together ETL tasks and execute them at scale. One downside to utilizing software such as Apache Airflow is that it requires provisioning and maintaining servers hosting the software on top of programming and maintaining the workflows as well. Managed solutions such as Azure Data Factory, AWS Glue, Oracle Data Integrator or Google Cloud Data Fusion help aid in this task by managing installation, provisioning, and maintenance of the ETL Software so the analyst can focus on ingesting the data at hand. An example of a cloud native ETL process is utilizing AWS Glue to crawl through an S3 bucket to ingest an AWS Cost and Usage Report. Now that the data has been processed it is able to be queried using standard SQL utilizing Amazon Athena.

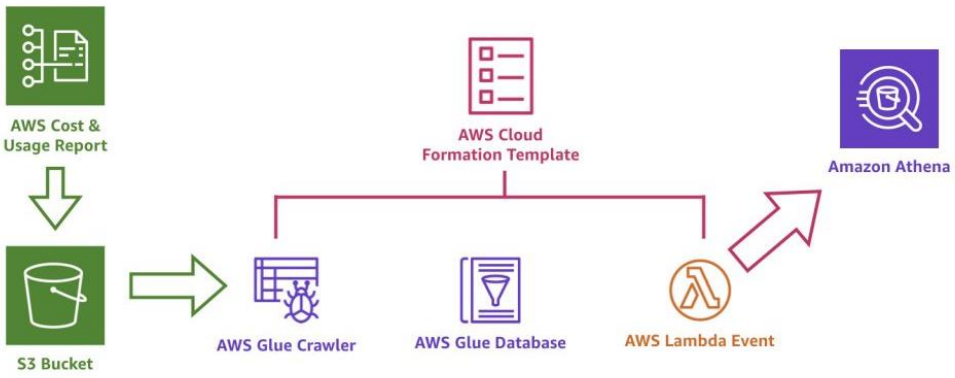

Figure 10. AWS cost query data flow. Source: AWS[3].

4.3. Analysis of Ingested Data

After building data pipelines, users are thrust into another key aspect in building the cloud framework through a thorough analysis of the recently ingested data. By utilizing easy to read dashboards, standard behaviors can be identified and thresholds can be set based on expected behaviors. One such widely-available software, Grafana, can connect to multiple cloud vendors and data sources to generate views of the information to better serve the user. Figure 11 shows the average CPU Utilization and network traffic for the last six hours of an instance.

A few things can be understood by reviewing this simple chart. First, the software

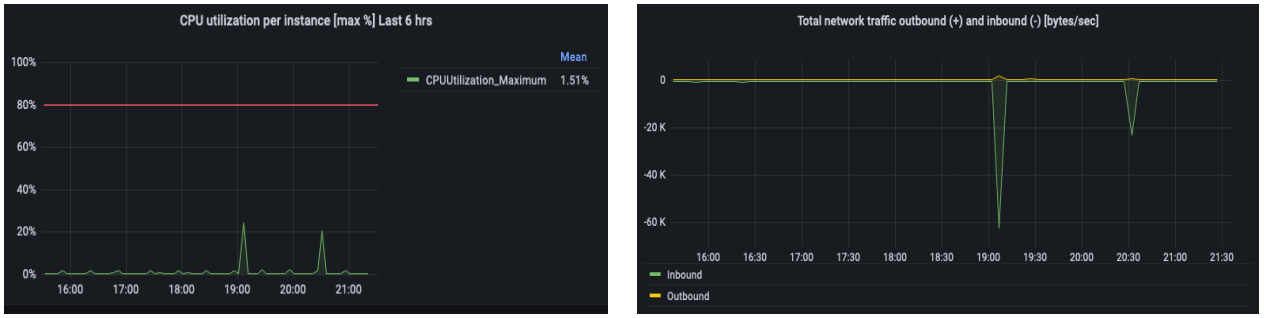

Figure 11. Compute Metrics showing CPU utilization and network traffic per instance.

running on the system is clearly nowhere near maxing out the compute power and it can be flagged as an opportunity to downsize the system and save money on hourly rental costs. In some cases, though, CPU Utilization might not be the whole story. The software load might not be taxing the system, but the volume of network traffic might demonstrate a need to upscale or downscale the network bandwidth, or find a more suitable technology in order to meet the need.

The reality of cloud technologies is that while compute efficiencies are necessary for operational deployments the cost of these technologies are calculated, in some cases, down to the second. Figure 12 shows a sample of cloud services being procured for an account. Being able to parse through costs to a highly detailed level is paramount in controlling cloud deployments because of the potential velocity of deployments and their resultant costs. From this workflow, resulting analysis produces alerts generated by either the cloud vendor systems themselves or third-party tools to notify account holders

of a bad state, underutilized resources, or costs exceeding a budget. All of the major cloud vendors generate metrics, allowing for analysis and data visualization that produce deeper insights and ensure proper management of the systems being hosted in the user accounts.

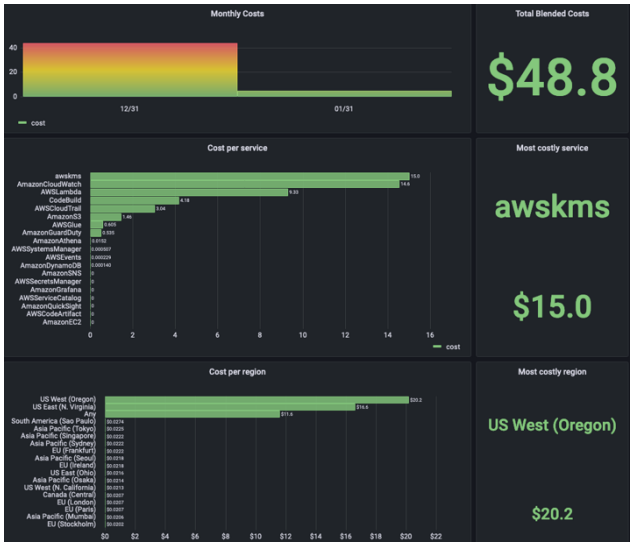

Figure 12. Sample various costs associated with each service in the account.

The Appendix provides additional figures depicting Technomics-designed dashboard views for FLECS. The visualizations show the level of traffic and usage in a well-architected and monitored system for EC2, EBS and other instances.

4.4. Architecture Reviews

A key aspect of optimizing cloud is running well architected reviews of each project. Each review consists of identifying key aspects (ex. cost drivers, requirement, operational metrics, etc.) that need to be assessed and performing a detailed assessment of performance. The key to these reviews is not only understanding the intent of the applications being developed on cloud vendors' technologies but also the art of the possible with the ever-changing landscape of technology. Figure 13 shows various architecture reviews from the major cloud providers, all of which have striking similarities.

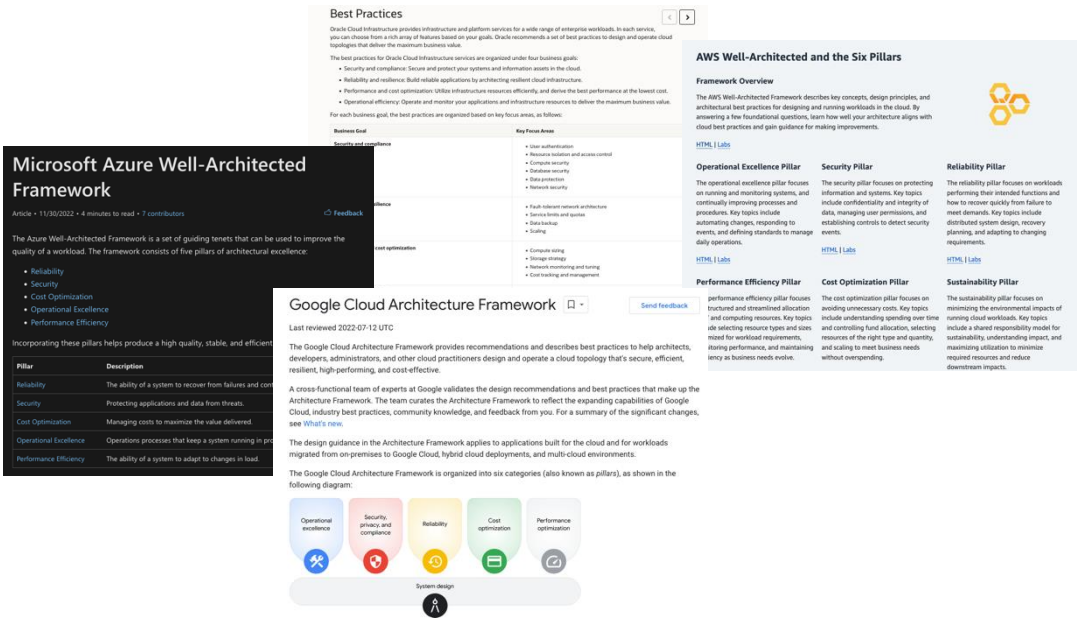

Figure 13. Sample of unique but similar architectural reviews from cloud providers.

Each cloud vendor focuses primarily on five tenets: reliability, security, cost optimization, operational excellence, and performance. Reliability is defined as the ability for a system to recover form failure and continue to function. Security is defined as maintaining customer trust by protecting user applications and data from threats. Cost optimization is defined as managing costs in order to deliver the maximum value. Operational excellence is defined as maintaining operational processes that keep a system running in production. Performance is defined as focusing on streamlining allocation of IT and compute resources to meet the need of the end applications. All of these tenants are intertwined; however, focusing an architectural review for each one individually gives the user the ability to evaluate the complete trade space from a specific viewpoint.

The process of an architectural review begins by breaking down complex components into modular parts that are reusable and scalable after assessing how operations are run through the various cloud environments, how changes are made within the environment and how efficient operations are run utilizing automation procedures. Users and development teams are guided to institute practices that align with operational

curiosity and goals of continual improvement. Development teams will be encouraged to approach operations with an anticipation of application failure and, as such, to do an effective analysis of different failure states to avoid catastrophic instances. The entire architectural review process is intended to be a collaborative exercise between users, development teams and the vendor.

5. Framework Established

A user's cloud journey begins with a clear skies and endless opportunities. It is essential to begin any cloud migration or implementation process with estimates backed by strong CERs as we have done with MUSCLE: informed by past executions and current cloud pricing data from all vendors. Once cloud solutions are initiated, users should build a thorough and holistic understanding of operations through constant monitoring, as we have accomplished with FLECS. The goal of the entire framework is simple yet powerful: continuous improvement of the architecture to enhance the mission by optimizing budget and execution. Combined with pre-deployment planning utilizing best of breed cloud cost estimation techniques the light can shine through the fog of the unknown and provide the customer with an exception product.

6. Next Steps in Cloud Monitoring and Optimization

On separate projects, Technomics analysts developed MUSCLE to improve forecasting of cloud cost estimates for program budgets, and FLECS to help the customer understand exactly what they were buying and where they could save money while improving mission performance. The implementation of FLECS cost account utilization dashboards in one program office delivered the insights and recommendations needed to turn-off unused tools and right-size existing solutions to requirements, resulting in significant cost savings. Combined, the MUSCLE and FLECS framework provides even more accurate predictions, greater cost savings and direct linkage from program requirements to operational cloud deployment.

The tools discussed in this paper are intended for use in generating data visualization products that inform the cloud user, cost account manager, and program manager.

Armed with these tools, analysts will have the ability to assume ownership of a programmatic cost driver and implement best practices for cloud cost forecasting and management, positively and directly impacting mission effectiveness and overall cloud financial management. Our objective for sharing the ideas behind the tools is to improve the broader community's ability to maximize their business value by leveraging the principles of active cloud account monitoring and waste reduction.

As new services and vendors emerge and prices continually change, we will continue to expand our cloud price database for the purpose of expanding the CER library to address a wider range of programmatic requirements (e.g., Images per day, # of users). The need to accurately forecast future years' cloud consumption budgets gets increasingly complex with additional service providers, Government vs Commercial vs Hybrid resources, and domain firewalls. Continuous refinement and expansion of methods for translating program requirements to Cloud cost estimates is imperative. We will continue to use FLECS and MUSCLE to help program managers make data-driven decisions and leverage user feedback to implement required improvements.

7. Appendix

7.1. Dashboard Visualization Examples

This section includes visualizations created as part of a FLECS monitoring dashboard. These dashboards are useful for keeping usage cost-efficient and are be used to right-size instance types.

Figure 14 shows inbound and outbound network traffic on EC2, which can show the level of traffic a particular application has.

Figure 14. EC2 Network Traffic.

Figure 15 shows the volume of bytes written and read to EBS volumes attached to EC2 instances.

Figure 15. EBS Data Volume.

Figure 16 shows the number of times particular functions had been called and details of their operations.

Figure 16. Lambda Metrics.

Figure 17 shows more dashboard views for usage, including throttling and error rates.

Figure 17. Additional Dashboard Views.

8. Works Cited

1. Accenture. (2023, January 15). Retrieved from accenture.com: https://www.accenture.com/us-en/cloud/insights/cloud-computing-index
2. Amazon Web Services. (2022, October 1). *AWS Pricing Calculator*. Retrieved from https://calculator.aws/#/addService?nc2=h_ql_pr_calc
3. AWS Cloud Financial Managemetn. (2023, February 1). *Querying your AWS Cost and Usage Report using Amazon Athena*. Retrieved from https://aws.amazon.com/blogs/aws-cloud-financial-management/querying-your-aws-cost-and-usage-report-using-amazon-athena/
4. Braxton, P. J. (2021). Inherent Risk and Uncertainty of Self-Similar Sizing Scales in Agile Software Development, or "Does This T-Shirt Make My Estimate Look Big?". *JITSWCF Proceedings.* Washington, DC: DHS.
5. Braxton, P. J. (2022). Uncertainty of Expert Judgment in Agile Software Sizing. *ICEAA Conference Proceedings.* Pittsburgh, PA: ICEAA.
6. Braxton, P. J., & Coleman, R. L. (2012). Teaching Pigs to Sing: Improving Fidelity of Assessments from Subject Matter Experts (SMEs). *ICEAA Chapter Luncheon Proceedings.* Washington, DC: ICEAA Washington Chapter.
7. Braxton, P. J., & Sayer, L. H. (2013). Probability Distributions for Risk Analysis. *ICEAA Conference Proceedings.* New Orleans, LA: ICEAA.
8. *Cost Estimating Body of Knowledge (CEBoK).* (2013). Annandale, VA: ICEAA.
9. FinOps Foundation. (2022, August 16). *The State of FinOps Report 2022.* Retrieved from https://data.finops.org/
10. Flexera. (2023, January 5). *State of the Cloud Report*. Retrieved from https://info.flexera.com/CM-REPORT-State-of-the-Cloud
11. Gartner. (2022, October 31). *gartner.com*. Retrieved from https://www.gartner.com/en/newsroom/press-releases/2022-10-31-gartner-

forecasts-worldwide-public-cloud-end-user-spending-to-reach-nearly-600-billion-in-2023

12. Gellatly, W., Braxton, P. J., Brown, D. H., Jones, L. F., & Wekluk, R. A. (2022). Dynamic Software Effort Estimation: How SWEET It Is! *ICEAA Conference Proceedings.* Pittsburgh, PA: ICEAA.

13. Kahneman, D. (2011). *Thinking, Fast and Slow.* New York, NY: Farrar, Straus and Giroux.

14. Kosmakos, C., & Brown, D. H. (2022). Are We Agile Enough to Estimate Agile Software Development Costs? *ICEAA Conference Proceedings.* Pittsburgh, PA: ICEAA.

15. Krempasky, R. R. (2022). Cloud Cost Estimating & Optimization Framework. *JITSWC.* Virtual: Department of Homeland Security.

16. Livio, M. (2008). *The Golden Ratio: The Story of PHI, the World's Most Astonishing Number.* Crown.

17. Radigan, D. (2022, February 24). *Story points and estimation*. Retrieved from Atlassian Agile Coach: https://www.atlassian.com/agile/project-management/estimation