A Cost-Effective and Scalable Processing of Heavy Workload with AWS Batch

Nagresh Kumar[1] and Sanjay Kumar Sharma[2]

[1]Department of Computer Science, Banasthali Vidyapith, Rajasthan, India, nagresh@gmail.com
[2]Department of Computer Science, Banasthali Vidyapith, Rajasthan, India, skumar2.sharma@gmail.com

***Corresponding Author:** Nagresh Kumar ; E-mail: nagresh@gmail.com

ABSTRACT- Recent technological advancements in the IT field have pushed many products and technologies into the cloud. In the present scenario, the cloud service providers mainly focus on the delivery of IT services and technologies rather than throughput. In this research paper, we used a scalable cost-effective approach to configure AWS Batch with AWS Fargate and CloudFormation and implemented it in order to handle a heavy workload. The AWS service configuration procedure, GitHub repository, and Docker desktop applications have been clearly described in this work. A cost-effective configuration and architecture of AWS Batch processing are given to provide high throughput. The processing of heavy workload by AWS Batch is represented in terms of execution time and the result shows that the concurrent execution reduces the execution time. To enhance the throughput heavy workload using batch processing an "Amazone FSx for Lustre" can also be used.

General Terms: Data Science, Machine Learning and Artificial Intelligence

Keywords: AWS, CloudFormation, Fargate, GitHub, Docker, Amazon Web service (AWS)

ARTICLE INFORMATION

Author(s): Nagresh Kumar and Sanjay Kumar Sharma

Special Issue Editor: Dr. Vikash Yadav
Received: 21/03/2022; **Accepted**: 21/04/2022; **Published**: 22/05/2022;
e-ISSN: 2347-470X;
Paper Id: 0422SI-IJEER-2022-01;
Citation: 10.37391/IJEER.100216
Webpage-link:
https://ijeer.forexjournal.co.in/archive/volume-10/ijeer-100216

crossref member CROSSREF.ORG THE CITATION LINKING BACKBONE

This article belongs to the Special Issue on **Recent Developments in Communication Technology using Machine Learning Techniques**.

Publisher's Note: FOREX Publication stays neutral with regard to Jurisdictional claims in Published maps and institutional affiliations.

1. INTRODUCTION

Modern science and technology require robust computation to process a high volume of data, retrieve, and discover meaningful and valuable information. In the traditional method, scientists and researchers used High-Performance Computing (HPC) Centre to process the work. The conventional method of distributing tasks and data among HPC is costly and difficult to manage. Cloud computing is an alternative to HPC, where scientists and researchers can process the high volume of data and pay only for what they use.

AWS Batch is a compute service on AWS where scientists, researchers, and developers quickly and efficiently execute thousands of computing jobs in a batch. AWS Batch dynamically provisions compute resources based on the volume and worker resources required for the specific submitted batch. AWS Batch, plans, schedules, and runs batch compute workloads against AWS compute services and capabilities, such as AWS Fargate, Amazon EC2, and spot instances. AWS Batch has no additional charges. The customer only pay for AWS resources created to store and run jobs in batches. With AWS batch computing, customers can execute many "tasks" on one or more compute resources without any intervention. Hence, users can create a job or an array of jobs, schedule and sequence them as needed, and configure compute resources [1].

In machine learning, discipline batch processing is a common method to process, analyze, and prediction of information from data sets of video, image, text, etc. Batch processing is a common need in Big-data analysis, data mining, and genomics research.

1.1 AWS Batch Features

A. Dynamic compute resource provisioning and scaling.
When we need to use AWS Fargate, FSx, Cloud Formation with AWS Batch, we only need to stop compute environment, job queue, and job definition. Complete compute infrastructures are available and there is no need to manage any infrastructure [2, 3].

B. AWS Batch with Fargate
AWS Fargate with AWS Batch provides managed server-less architecture for the execution of jobs. In this environment, every job receives the exact amount of CPU cycle and memory requested by the Jobs, and there is no wastage of resources.

C. Support for tightly-coupled HPC workloads
AWS batch enables to execute a single job having multiple tasks on multiple EC2 instances or multiple Jobs on multiple EC2. The AWS batch facility helps us to execute heavy workloads efficiently like Distributed graphics processing unit (GPU) model training or tightly coupled high-performance computing.

D. Priority-based job scheduling
AWS batch provides a facility to configure multiple queues with priority scheduling. There is the facility to assign priority to the jobs in the queue. We can configure job definition and job queue by AWS batch linked with different compute facilities. Each job executes according to given priority with optimal compute facility.

E. Support for GPU scheduling
GPU scheduling enables to configure the number of accelerators and type of accelerators needed during job definition. AWS Batch scales up according to the required instance based on job definition and GPU configuration. AWS batch also isolates accelerators according to instance required so that specified containers can access them.

F. Support for popular workflow engines
AWS Batch service can also be used with the open-source and commercial workflow engine. AWS step function integrates AWS Batch with the workflow engine.

G. Integration with EC2 Launch Templates
AWS Batch supports EC2 startup templates that allow creating custom models for compute resources and scaling instances in order to meet the requirements.

H. Resource allocation strategies
AWS batch provides three allocation strategies of resource provisioning to consider throughput and pricing.

Best Fit: AWS Batch prioritizes the cheapest combination of jobs and instance types and then chooses the instance type that best suits for the minimum cost. This process of customization strategy reduces the costs but can also reduce scalability.

Best Fit Progressive: In this strategy AWS batch select additional instance types with minimum cost, which are large enough to execute the job. If these selected instances are not available then AWS Batch selects new instances.

Spot Capacity Optimized: In this allocation strategy, AWS Batch selects one or more than one spot instances which are large enough to execute the jobs.

1.2 Components of AWS Batch
AWS Batch is a compute service that facilitates the execution of batch jobs on multiple resources provisioned on different availability zone of a single region. Now, a compute environment for AWS Batch under a particular virtual private cloud (VPC) is created. After that, a job queue is associated with an active compute environment. Now, we can define a job that refers to a container image. Container images are a set of programs to perform a particular task.

Jobs
A Job is the smallest unit of work, a script or image of a script available through Docker container image submitted to AWS batch. It runs on configured EC2 instance of AWS Fargate using specified parameters by job definition.

Job Definitions
Job definition is the main component of AWS batch configuration. It is also called the blueprint of resources on which a job is provisioned to execute. Here Jobs supplied with predefined (identity and access management) IAM roles to access the resources. The user can define CPU requirement, memory, container image, timeout, and other environment variables.

Job Queues
When the user submits a job, it resides in the Jobs queue. Job scheduler schedules the Jobs from the job queue to the resources provisioned. The priority value for Jobs across job queues is also assigned.

Compute Environment
A compute environment is a configured computing resource in AWS. Instance type is an important parameter to create compute environment. Users can select a particular instance type as per execution workload like c4.xlarge, m5.10xlarge, etc. AWS Batch can be easily configured, launched, manage and terminate as per need. The working architecture of AWS Batch is described in *figure-1*.

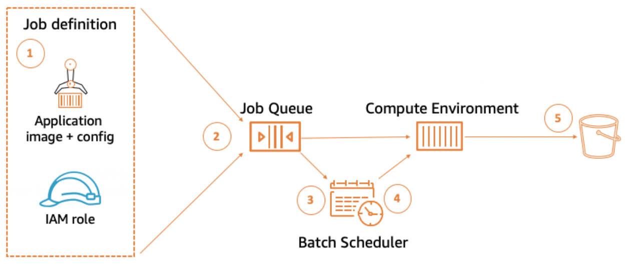

Figure 1: AWS Batch Architecture

1. The user creates the job definition
2. The user submits jobs to managed AWS Batch job queue
3. AWS Batch evaluate the CPU, Memory, and GPU requirement of the jobs in the queue and launch the compute resources in the compute environment
4. Job exists with status and writes results to user-defined storage

The major contribution of this research paper is providing architecture for high performance computing with AWS resources and services. *Section-1* describes the concepts and architecture of AWS Batch service. *Section-2* describe the related works. *Section-3* describe the steps to configure AWS Batch to handle heavy workloads. In *Section-4*, we have shown our architecture to handle heavy workload. *Section-5* is about execution output and result discussion. *Section-6* describe the conclusion and future scope.

2. RELATED WORKS
In the present cloud computing scenario, the core issues are heavy processing workload, scientific workflow, and resource provisioning. AWS batch plays an important role in improving workflow scheduling, resource allocation, scheduling, and cost optimization.

Kyle M. et al. [4] used a high throughput AWS batch to see the capability of AWS batch in executing a heavy scientific workload. The author develops a testing suite. A workflow has been applied to configure AWS batch infrastructure by increasing the number of jobs, job size, job queue, etc. The author found that when there is a significant delay in sending job to AWS Batch and running AWS EC2, there is a little overhead in the actual running of the Jobs.

D. Cui, et al. [5] proposed a multi-objective optimization to solve workflow scheduling based on a heterogeneous distributed deep learning strategy. AWS Batch is a compute service on AWS where scientists and developers can easily and efficiently execute the processing of thousands of batch jobs.

Due to the increasing demand for interactive supercomputing, big-data analysis, machine learning algorithms, and fast processing of thousands of jobs is essential in reasonable throughput [B C]. C. Byun et al. [8] proposed a pre-emptive approach to implement spot jobs on the MIT super cloud system to utilize batch for the long-running batches job. This new approach separates job pre-emption and scheduling operation. This approach achieves 100 times faster performance than the standard scheduler with automatic priority pre-emption.

Most existing workflow scheduling algorithms are tightly bounded to compute resources. The compute resources in this environment are not released or provisioned on-demand until execution is completed. Such an environment directly affects the efficiency of workflow scheduling. C. Lin et al. [9] proposed SHEFT algorithms optimize the workflow time scheduling and also elastically provisioned the resources.

A multi-core system can run more than one user program concurrently. When the number of user programs exceeds the number of core, oversubscription occurs. In the oversubscription technique programs are executed in a time-shared manner using the resources. In the traditional method, oversubscription is used to enhance system utilization. V. March et al. [10] suggested a batch scheduler for HPC with virtualization technology.

Cloud computing is an advance distributed technology. In this technology, there are various features like heterogeneity, scalability, and virtualization of resources that are most suitable for scientific workflow for processing thousands of jobs and vast amounts of data. Many scheduling algorithms have been proposed for scientific workflow. V. Vinothina et al. [11] have presented an IWSACO with variance in WFSACO. The author shows that IWF performs better after the implementation and analysis of results.

J. Agarkhed et al. [12] have presented a multi-objective optimization technique for resource provisioning to handle the workload. A batch job is placed on an idle workstation in an opportunistic scheduling algorithm to improve system performance.

J. Abawajy et al. [13] have presented a new opportunistic scheduling algorithm using three policies known as job rotation, scheduling policy used in condor, and round-robin policy. The result shows that the algorithm performs better than the existing scheduling algorithm.

To enhance the throughput and system performance the parallel programming environment is most suitable to handle a heavy workload. As per other parallel processing environments, time elasticity is insufficient for high throughput. D. Kumar et al. [14] examined a runtime elasticity environment for high throughput.

S. Rana et al. [15] investigated various workflow scheduling algorithms to model high-performance throughput. The author also identifies multiple challenges that need to be addressed to utilize resources efficiently. They also analyze pros and cons as well as list numerous research issues in workflow scheduling algorithms.

Y. T. Chou et al. [16] also raises a core issue in solving the workflow scheduling. The author proposed Search Economics for Working Scheduling Algorithm (SEWSA) and experimentally showed that the proposed algorithm performance is better than the existing algorithm concerning cost and makes span.

In high-performance scientific workflow scheduling, many processes execute on different processors. This reduces the system performance due to job dependency, frequent communication among processes, and synchronization. To handle these issues R. Balasubramanian et al. [17] proposed a co-scheduling technique along with batch scheduling and gang scheduling where resources are shared. The author concludes that this technique works well when jobs are distributed parallel on the entire machine.

3. PROCEDURE

Currently, batch processing is a common method to execute a heavy workload. AWS Batch is one of the best-managed services to perform the task efficiently. AWS Batch allows researchers, developers, and scientists to focus on solving problems, analyzing methods used rather than managing the resources. But before that, we need to complete the following steps in setting up with AWS Batch according to the guidelines given below [18-23].

Step 1- Create the following accounts
(a) AWS Console account
(b) GitHub account
(c) Docker hub account
Step 2- Install and configure AWS Command line Interface (CLI)
Step 3- Install Docker on AWS
Step 4- Create the AWS CloudFormation stack
(a) Log in to AWS CloudFormation console
(b) Create a cloud formation stack using the following steps
  (i) Start Create Stack wizard
  (ii) Select a stack template

(iii) Specify stack parameters
(iv) Set the AWS CloudFormation stack options
(v) Review the stack

(c) Once stack creation is complete, select outputs tab to view identifiers for the resources created.

Step 5- Upload input dataset to AWS simple storage service (S3)

Step 6- Link GitHub repository containing necessary script and template to Docker hub

Step 7- Create, and upload the Docker image

Step 8- Link Elastic Container Registry (ECR) to Docker

Step 9- Submit the sample AWS Batch jobs using the process given below

(a) Create/select compute environment
(b) Create job queue
(c) Create job definition
(d) Submit job

4. ARCHITECTURE

As per the architecture shown in *figure-1*, we use the following parameters in configuration.

1. **Region-** N. Virginia
2. **Compute-** c4.xlarge
3. **Storage-** S3 standard
4. **CloudFormation-** We modeled the AWS resources through CloudFormation. This resource can also be configured through the AWS management console or AWS CLI.
5. **GitHub Repository (private)** - The necessary scripts are available here.
6. **Docker desktop application-** Image from Github repository is pushed in Docker desktop application. In place of the Docker desktop Application, ECR can also be used.
7. **Array Job-** An array job is an application under AWS Batch service which is used to share standard parameters like memory, vCPUs, and job definition. It is used to execute a set of different jobs with common parameters by distributing concurrently and parallel across multiple resources. A single job is divided into child jobs as per array size declared during job creation. These child jobs span over multiple EC2 instances to execute concurrently [24].

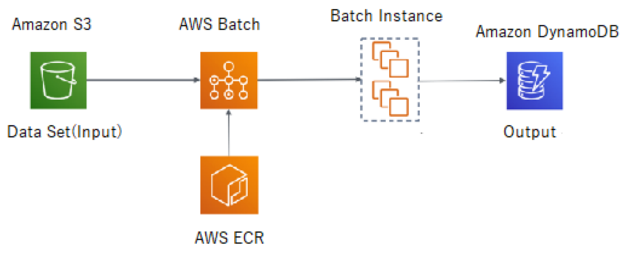

Figure 2: AWS Batch workload process

Now, our data sets are available in the S3 bucket in .csv format. Since we configured these arrangements to process the heavy workload, the AWS Batch will fetch the image of our scripts from the ECR/Docker desktop application. Now our script process the data sets according to different configurations of AWS Batch. Parallel execution of compute environment is configured with a different scenario of job queue and array size.

4.1 Cost of Resources

AWS generally charges for the resources named compute, storage, and data transfer. There are no charges for the configuring or architecting of resources. AWS charges based on the "pay as you go" principle.

Here we have used S3 standard storage and c4.xlarge compute services. Costs of these resources are shown in the tables given below [25, 26].

Table 1. Cost of c4.xlarge EC2 instance

Instance Name	Region	vCPU	RAM (MiB)	Storage (GB)	Price / hour (USD)
c4.xlarge	N. Virginia	4	7680	EBS only	0.199

Table 2. Storage Cost

S3 Standard/Per Month	Cost/GB (USD)
Up to 50 TB	$0.023
51TB - 500 TB	$0.022
Over 500 TB	$0.021

To process the heavy workloads like scientific workload, machine learning algorithm, and big data [6, 7] processing, we need high-performance computing (HPC) resources but it is not always feasible for researchers, scientists, and developers. Using AWS service charges mention in *table-1* and *table-2*, and our AWS Batch configuration shown in *figure-1*, it becomes cost-effective and feasible for every individual. The execution process of *scenario-1* and the *scenario-2* process is given below.

5. RESULT ANALYSIS

Scenario-1: We created 200 copies of the data set in S3 and our scripts execute this data set with a different cluster of compute. We used 5 instances of c4.xlarge. The execution time in different scenarios is captured, calculated, and represented in tabular and graphical form.

Table 3. Array size Vs Total execution Time

Array size	Input files	Execution time
0	200	25m 14s
2	200	12m 08s
4	200	5m 52 s
8	200	2m 38 s
10	200	2m 4s

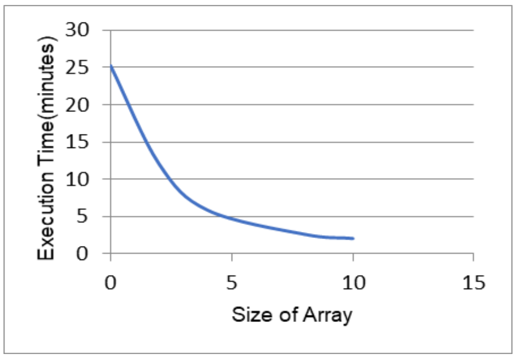

Figure 3: Array size Vs Total execution Time

Result discussion of scenario-1: In this scenario, we used 5 instances of c4.xlargge where each instance has 4 vCPUs. It means a total of 20 vCPUs are available to execute our scripts in parallel to process the dataset. When we take array size zero means a single task executed 200 times sequentially that is not a better utilization of the resources. When we take array size 4 means 4 scripts are executed concurrently on 20 vCPUs. Similarly, when the array size is 10, then 10 scripts are executed concurrently on 20 vCPUs. *Table-1* and *figure-3* shows that as the array size increases the execution time decreases.

Scenario-2:

Table 4. Array size Vs Total execution Time

Array size	Input files	Execution time
0	200	26m 06s
2	200	12m 29s
4	200	6m 21 s
8	200	8m 14 s
10	200	11m 13s

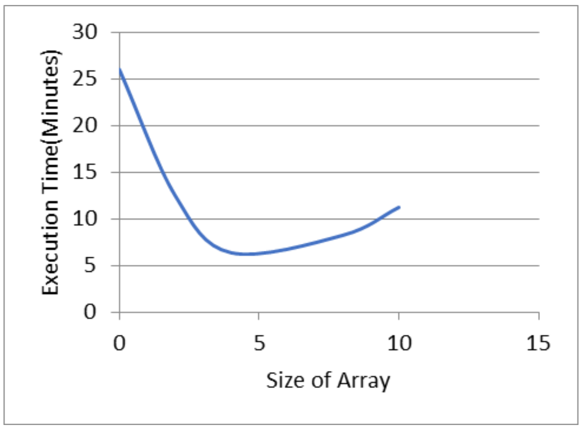

Figure 4: Array size Vs Total execution Time

Result discussion of scenario-2: In this scenario, we used only one instance of c4.xlarge. All other parameters are the same as in *scenario-1*. In this only one instance with 4 vCPUs are available. When the array size increases up to 4 the execution time decreases and when the array size is greater than 4, then more than 4 scripts are ready to execute concurrently on vCPUs. This increases the waiting time as well as communication overhead and hence execution time increases.

It is clear from *scenario-1* and *scenario-2* that there is a relation between array size and execution time. So in architecting the resources, CloudFormation plays an important role. As per architecture, array size must be taken carefully for cost-effective batch processing for heavy work.

6. CONCLUSION

In this research paper, we focus on the architecture and configuration of AWS Batch service for cost-effective processing of heavy workload with high throughput. As per implementation results shown in *table 3*, we found that the concurrent execution of jobs reduces the execution time. The *table 4* shows that first, the execution time decreases and then increases because sufficient resources are not available as per the size of the array and therefore the jobs are in a waiting state also increases the execution time. So the configuration of resources and array size must be taken carefully. As a future direction, too many architecture with AWS Batch and other AWS services may be possible. To enhance the fast accessing and throughput of batch processing Amazon FSx for Lustre" can also be used. Selection of instances, storage and other AWS services to configuration of AWS Batch for high performance computing (HPC) must be taken carefully.

7. ACKNOWLEDGMENTS

We would like to express our appreciation to all those who have supported us during our research and study in the department of Computer Science at Banasthali Vidyapith, Rajasthan, India.

REFERENCES

[1] AWS documentation on AWS Batch. Accessed on: Feb. 20, 2022 [Online]Available: https://docs.aws.amazon.com/batch/latest/userguide/what-is-batch.html

[2] AWS documentation on AWS Batch Features. Accessed on: Feb. 20, 2022 [Online] Available: https://aws.amazon.com/batch/features/

[3] Chandrajeet Yadav, Vikash Yadav et al, "Authentication, Access Control, VM Allocation and Energy efficiency towards Securing Computing Environments in Cloud Computing", Annals of the Romanian Society for Cell Biology, Association of Cell Biology Romania Publication, ISSN 1583-6258, Vol. 25, No. 6, pp. 17939-17954, June 2021.

[4] Kyle M. D. Sweeney and Douglas Thain, 2018. Early Experience Using Amazon Batch for Scientific Workflows. In Proceedings of the 9th Workshop on Scientific Cloud Computing (ScienceCloud'18). Association for Computing Machinery, New York, NY, USA, Article 5, 1–8.

[5] D. Cui et al., "Cloud Workflow Task and Virtualized Resource Collaborative Adaptive Scheduling Algorithm Based on Distributed Deep Learning," 2020 IEEE International Conference on Advances in Electrical Engineering and Computer Applications(AEECA), 2020, pp. 137-14.

[6] Vikash Yadav et al, "Healthcare, IoT, and Big Data Support", "Empowering Artificial Intelligence through Machine Learning New

Advances and Applications", Print ISBN: 9781771889308, pp. 57-81 Published by "CRC Press & Apple Academic Press", July 2021.

[7] C. Byun et al., "Best of Both Worlds: High Performance Interactive and Batch Launching," 2020 IEEE High Performance Extreme Computing Conference (HPEC), 2020, pp. 1-7.

[8] C. Lin and S. Lu, "Scheduling Scientific Workflows Elastically for Cloud Computing," 2011 IEEE 4th International Conference on Cloud Computing, 2011, pp. 746-747.

[9] V. March, S. See, M. Garg, P. Gupta and T. Atrey, "Batch Scheduler for Personal Multi-Core Systems," in 2013 12th International Symposium on Distributed Computing and Applications to Business, Engineering & Science, Hong Kong, China, 2010 pp. 584-587.

[10] V. Vinothina, "Scheduling scientific workflow tasks in cloud using swarm intelligence," 2017 IEEE International Conference on Current Trends in Advanced Computing (ICCTAC), 2017, pp. 1-5.

[11] J. Agarkhed and R. Ashalatha, "Optimal workflow scheduling for scientific workflows in cloud computing," 2016 International Conference on Emerging Technological Trends (ICETT), 2016, pp. 1-6.

[12] J. Abawajy, "Job Scheduling Policy for High Throughput Computing Environments," in Proceedings of the Ninth International Conference on Parallel and Distributed Systems, Taiwan, China, 2002 pp. 605.

[13] D. Kumar, Z. Shae and H. Jamjoom, "Scheduling Batch and Heterogeneous Jobs with Runtime Elasticity in a Parallel Processing Environment," 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum, 2012, pp. 65-78. 10.1109/IPDPSW.2012.10

[14] S. Rana, A. Choudhary and K. J. Mathai, "A critical analysis of workflow scheduling algorithms in infrastructure as a Serivce Cloud and its research issues," 2016 IEEE Students' Conference on Electrical, Electronics and Computer Science (SCEECS), 2016, pp. 1-6.

[15] Y. -T. Chou, S. -J. Liu, T. -C. Wu, C. -L. Wu, C. -W. Tsai and M. -C. Chiang, "An Effective Algorithm for Cloud Workflow Scheduling," 2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2018, pp. 3603-3608.

[16] R. Balasubramonian and N. Madan, "Power Efficient Approaches to Redundant Multithreading" in IEEE Transactions on Parallel & Distributed Systems, vol. 16, no. 08, pp. 1066-1079, 2007.

[17] Docker Manuals on Docker Hub Quickstart. Accessed on: Feb. 20, 2022 [Online] Available: https://docs.docker.com/docker-hub/

[18] AWS documentation Installation of AWS CLI. Accessed on: Feb. 20, 2022 [Online] Available:https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html retrieve from www.aws.amazon.com

[19] AWS documentation on docker installation on Linux AMI. Accessed on: Feb. 20, 2022 [Online] Available: https://docs.aws.amazon.com/AmazonECS/latest/developerguide/docker-basics.html

[20] AWS documentation on AWS CloudFormation. Accessed on: Feb. 20, 2022 [Online] Available: https://docs.aws.amazon.com/cfn-guard/latest/ug/cfn-guard.pdf

[21] GitHub documentation on Publishing Docker images. Accessed on: Feb. 20, 2022 [Online] Available: https://docs.github.com/en/actions/publishing-packages/publishing-docker-images

[22] AWS documentation on AWS Batch API Reference. Accessed on: Feb. 20, 2022 [Online] Available:https://docs.aws.amazon.com/batch/latest/APIReference/batch-api.pdf

[23] AWS documentation on AWS Batch -Array Jobs. Accessed on: Feb. 20, 2022 [Online] Available: https://docs.aws.amazon.com/batch/latest/userguide/array_jobs.html

[24] Amazon EC2, On-Demand Pricing. Accessed on: Feb. 20, 2022 [Online] Available: https://aws.amazon.com/ec2/pricing/on-demand/

[25] Cloud Storage on AWS, Amazon S3 pricing. Accessed on: Feb. 20, 2022 [Online] Available: https://aws.amazon.com/s3/pricing/