MAPLE: Model Aggregation and Prediction for Learned Ecosystem

Chetan Phalak
Tata Consultancy Services
India
chetan1.phalak@tcs.com

Dheeraj Chahal
Tata Consultancy Services
India
d.chahal@tcs.com

Aniruddha Sen
University of Massachusetts Amherst
USA
aniruddhasen@umass.edu

Mayank Mishra
Tata Consultancy Services
India
mishra.m@tcs.com

Rekha Singhal
Tata Consultancy Services
India
rekha.singhal@tcs.com

ABSTRACT

Many Artificial Intelligence (AI) applications are composed of multiple machine learning (ML) and deep learning (DL) models. Intelligent process automation (IPA) requires a combination (sequential or parallel) of models to complete an inference task. These models have unique resource requirements and hence exploring cost-efficient high performance deployment architecture especially on multiple clouds, is a challenge. We propose a high performance framework MAPLE, to support the building of applications using composable models. The MAPLE framework is an innovative system for AI applications to (1) recommend various model compositions (2) recommend appropriate system configuration based on the application's non-functional requirements (3) estimate the performance and cost of deployment on cloud for the chosen design.

CCS CONCEPTS

• **General and reference** → **Performance**; **Estimation**.

KEYWORDS

Performance and cost estimation, cloud deployment

ACM Reference Format:
Chetan Phalak, Dheeraj Chahal, Aniruddha Sen, Mayank Mishra, and Rekha Singhal. 2022. MAPLE: Model Aggregation and Prediction for Learned Ecosystem. In *Companion of the 2022 ACM/SPEC International Conference on Performance Engineering (ICPE '22 Companion), April 9–13, 2022, Bejing, China.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3491204.3527497

1 INTRODUCTION

Complex AI applications are composed of numerous models linked to each other as a directed acyclic graph (DAG). The output and input from each of these models are processed using transformation functions. An inference request flows in such workflows through multiple models and transformation functions. One example of such workflows is an information extraction application which is composed of models for extracting data from documents available in different formats such as tables, hand-written text,logos, etc. Each of these models has unique resource requirements. Finding optimal architecture for deploying such workflows on cloud is a tedious and challenging process as it involves experimenting with numerous cloud services and instance configurations. InferLine [4] is one such system for the provisioning and management of an ML inference pipeline. Although InferLine recommends optimal hardware configurations, it does not consider characteristics of cloud services and their unique cost models. We propose a tool called MAPLE for estimating the performance and cost of deployment of complex AI workflows on cloud. Some of the salient features and our contributions through this tool are as follows:

- Estimating the performance and cost of deployment of these workflows on multiple clouds without deploying application
- Designing optimal architectures for deploying AI workflows that honor Service Level Agreements (SLAs)
- It allows users to build and visualize complex AI workflows. Also, *what-if* and *if-what* analysis is possible by varying the configuration of instances used for deploying workflows

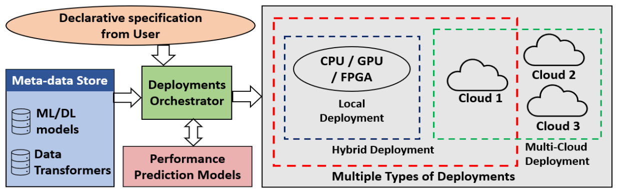
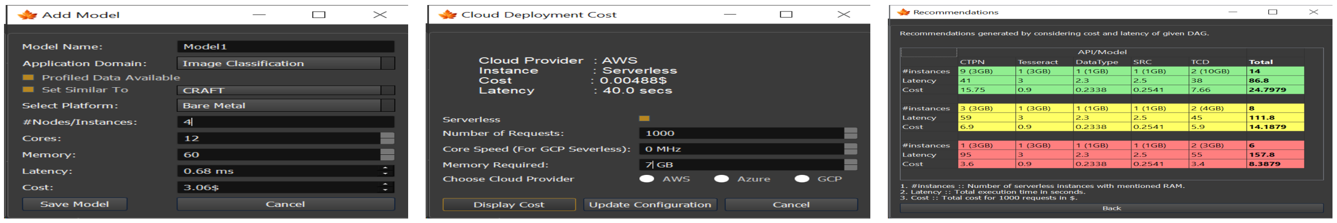


Figure 1: MAPLE Architecture

2 OVERVIEW OF MAPLE

Major components of MAPLE architecture (Figure 1) are as follows: ***Use specification:*** This component allows users to specify the budget and expected performance in terms of latency and throughput.
Model meta-data: The tool maintains the repository of performance data (latency and throughput) of well-known models for different application domains such as computer vision, image classification, speech-to-text, etc. The performance database is built by crowdsourcing and also by conducting experiments in the lab.



Add Model

Model Name:	Model1
Application Domain:	Image Classification
▨ Profiled Data Available	
▨ Set Similar To	CRAFT
Select Platform:	Bare Metal
#Nodes/Instances:	4
Cores:	12
Memory:	60
Latency:	0.68 ms
Cost:	3.06$

(a)

Cloud Deployment Cost

```
Cloud Provider  : AWS
Instance        : Serverless
Cost            : 0.00488$
Latency         : 40.0 secs
```

Serverless	▨
Number of Requests:	1000
Core Speed (For GCP Severless):	0 MHz
Memory Required:	7 GB
Choose Cloud Provider	● AWS ● Azure ● GCP

(b)

Recommendations

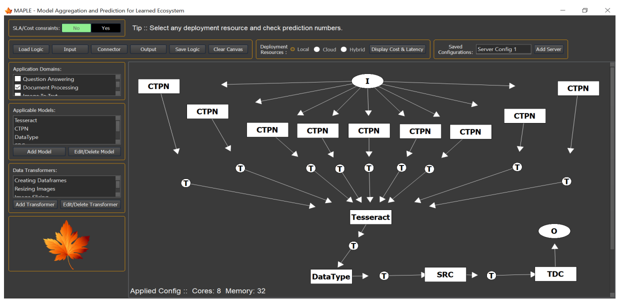
Recommendations generated by considering cost and latency of given DAG.

	API/Model					
	CTPN	Tesseract	DataType	SRC	TCD	**Total**
#instances	9 (3GB)	1 (3GB)	1 (1GB)	1 (1GB)	2 (10GB)	**14**
Latency	41	3	2.3	2.5	38	**86.8**
Cost	15.75	0.9	0.2338	0.2541	7.66	**24.7979**
#instances	3 (3GB)	1 (3GB)	1 (1GB)	1 (1GB)	2 (4GB)	**8**
Latency	59	3	2.3	2.5	45	**111.8**
Cost	6.9	0.9	0.2338	0.2541	5.9	**14.1879**
#instances	1 (3GB)	1 (3GB)	1 (1GB)	1 (1GB)	2 (3GB)	**6**
Latency	95	3	2.3	2.5	55	**157.8**
Cost	3.6	0.9	0.2338	0.2541	3.4	**8.3879**

1. #instances :: Number of serverless instances with mentioned RAM.
2. Latency :: Total execution time in seconds.
3. Cost :: Total cost for 1000 requests in $.

(c)

Figure 2: Some screenshots from MAPLE (a) Adding a new model to the data store (b) Cloud deployment cost display (c) Cost and execution time for processing documents in Deep Reader with different serverless configurations



MAPLE - Model Aggregation and Prediction for Learned Ecosystem

SLA/Cost consraints: [✓ No] Yes

Tip :: Select any deployment resource and check prediction numbers.

Deployment Resources : [✓ Local] Cloud Hybrid

Saved Configurations: Server Config 1

Application Domains: ☐ Question Answering ☑ Document Processing

Applicable Models: Tesseract, CTPN, DataType

Data Transformers: Creating Dataframes, Resizing Images

Applied Config :: Cores: 8 Memory: 32

Figure 3: MAPLE Tool Screen

Performance and cost model: Maple has in-built performance and cost extrapolation models. These models are used to estimate the performance and cost of inference when performance data is not available in the data store.

Deployment orchestrator: The deployment orchestrator uses model data store as well as performance and cost models to evaluate the cost of deployment on heterogeneous architectures (CPU, GPU) and services (VMs instances, serverless etc.) over multiple clouds such as AWS, Azure, and Google Cloud Platform. The cost is evaluated by fetching the cost specifications available on the cloud vendor's website. In case user specifications are provided, appropriate hardware configuration and architecture are suggested for the deployment.

The MAPLE GUI has the following features:

SLA specification: Using a toggle button, a user can specify whether there are specific SLAs to meet or not.

Deployment option selection: MAPLE supports local (data center), cloud (includes AWS, GCP, Azure), and Hybrid (includes a mix of supporting clouds and local servers) deployment options. Users can select any one option and instance to get the end-to-end cost and latency of designed workflows on the provided configurations.

Server Configuration: This palette gets enabled once user selects "local" or "hybrid" as a deployment resource. Users can add any number of server configurations and use them to design a logic.

Application Domain selection: MAPLE has a few predefined domains listed in this palette. User is allowed to choose or add one or more domains that suit the application.

Applicable Model Selection: As per the selection of appropriate applicable domains in the palette mentioned above, the ML/DL models which fall under respective domains are populated in this palette. Users can select any number of models and pull on the canvas for design application logic. Users can add to this list (figure 2(a))

Transformer Function Selection: The end-to-end latency of an application consisting of ML/DL models is affected by the pre-processing of input data and post-processing of output data. MAPLE provides the creation and application of such data transformers.

3 USECASE

We demonstrate the use of MAPLE for estimating the performance of an information extraction application called Deep Reader [2, 3] on the AWS serverless platform Lambda. The application consists of multiple models for text detection (e.g. CTPN [5] or CRAFT [1]), text recognition (e.g. Tesseract citetes), and text labeling along with transformation functions for data processing. Users can choose the models of their choice using the information extraction application in the tool. A complete workflow is designed using the required models as shown in figure 3. Using the characterization data available in the data store for the models, the end-to-end cost and performance are evaluated for various configurations of the serverless instances chosen as shown in figure 2(b) and 2(c).

4 CONCLUSION

We proposed a tool for visualizing AI workflows and estimating the performance and cost of deploying these workflows on multiple clouds. Additionally, tools features are demonstrated using an information extraction application called Deep Reader.

REFERENCES

[1] Youngmin Baek, Bado Lee, Dongyoon Han, Sangdoo Yun, and Hwalsuk Lee. 2019. Character region awareness for text detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 9365–9374.

[2] Dheeraj Chahal, Ravi Ojha, Manju Ramesh, and Rekha Singhal. 2020. Migrating Large Deep Learning Models to Serverless Architecture. In *2020 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*. 111–116. https://doi.org/10.1109/ISSREW51248.2020.00047

[3] Dheeraj Chahal, Manju Ramesh, Ravi Ojha, and Rekha Singhal. 2021. High Performance Serverless Architecture for Deep Learning Workflows. In *2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*. IEEE, 790–796.

[4] Daniel Crankshaw, Gur-Eyal Sela, Xiangxi Mo, Corey Zumar, Ion Stoica, Joseph Gonzalez, and Alexey Tumanov. 2020. InferLine: Latency-Aware Provisioning and Scaling for Prediction Serving Pipelines *(SoCC '20)*. Association for Computing Machinery, New York, NY, USA, 477–491. https://doi.org/10.1145/3419111.3421285

[5] Zhi Tian, Weilin Huang, He Tong, Pan He, and Yu Qiao. 2016. Detecting Text in Natural Image with Connectionist Text Proposal Network, Vol. 9912. 56–72. https://doi.org/10.1007/978-3-319-46484-8_4