DATA+AI SUMMIT
BY databricks

Albertsons Companies

NEAR REAL-TIME INFENRECE WITH DATABRICKS SERVERLESS COMPUTE

Amit Adiraju , Diana Adam

©2024 Databricks Inc. — All rights reserved

Amit Adiraju
Sr. ML Engineer

Diana Adam
Sr. Director Data Science

©2024 Databricks Inc. — All rights reserved

FORECASTING AT ALBERTSONS

Corporate Overview

As one of the largest food and drug retailers in the United States, Albertsons Companies operates stores to be locally great while being nationally strong. The Company's omnichannel approach and commitment to innovation are making it easier and more convenient for customers to shop, paving the way for profitable, sustainable growth.

Company Profile

2,269 Retail Stores	**22** distribution centers	**19** manufacturing plants
$79.2 B in sales (FY 2023)	**$4.3 B** adj. EBITDA (FY 2023)	**285,000 jobs** One of the largest retail employers

Company Banners

SAFEWAY, Albertsons, VONS, Jewel Osco, ACME, shaw's, Tom Thumb, United supermarkets, United Express, Randalls, Albertsons Market, PAVILIONS, star market, KINGS, Market Street, Haggen, CARRS, Andronico's Community Markets, Balducci's, Amigos, Lucky, Albertsons Market Street

©2024 Databricks Inc. — All rights reserved

FORECASTING USE CASE REQUIREMENTS

- A future looking view into the estimated performance of our products is critical for the short- and long-term strategy planning of **What, Where, When** and for **How Much** to sell.

- Forecasting needs to be done at the **relevant granularity** to support these goals and have a **high degree of accuracy**.

- Business users need the ability to "**simulate**" in **near real time** the **impact** of different strategies on the forecast in based on various objectives to pick the best one.

Geography + Business Strategy + Product Hierarchy = Forecasting Keys

Total Number of Forecasting Models	~124, 000
Algorithms Used	FB Prophet, Moving Average, Temporal Fusion Transformer and GPVAR
Registry	Mlflow (model registry, as well as experiment tracking)
Inference & Serving	Run simulations for different inputs in less than 15 seconds per request (~1400 individual forecast models per request)

©2024 Databricks Inc. — All rights reserved

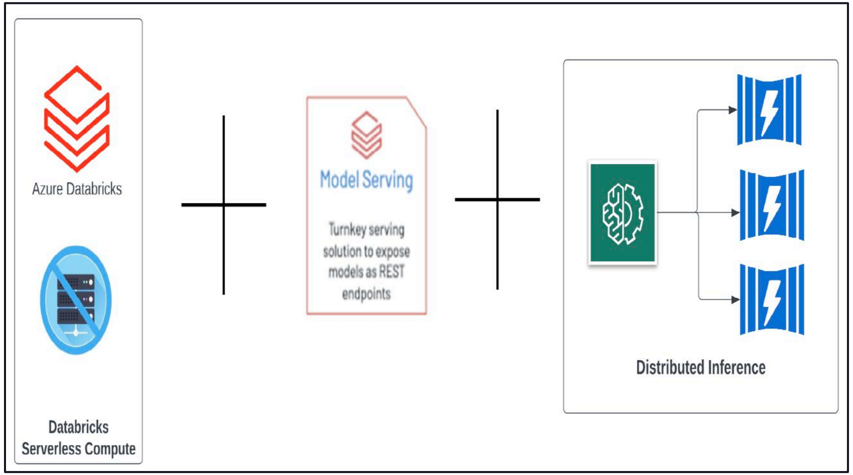
TYPICAL APPROACHES

Approach	Details	Challenges
1. On Demand Batch Inference Job with group of models.	1. User Request -> FAST API on AKS -> Trigger Job Cluster for Inference.	1. Cluster Startup time = 3 mins + model inference time = ~ 2 mins.
2. One REST end point per model.	2. Deploy one prophet model, per REST endpoint, and query group of endpoints based on user's request.	2. Not scalable as no. of models grow, expensive and high operational complexity. Also involves multiple network calls.

©2024 Databricks Inc. — All rights reserved

TO ADDRESS STARTUP TIME OF COMPUTE

Databricks Serverless Compute + Mlflow Serving

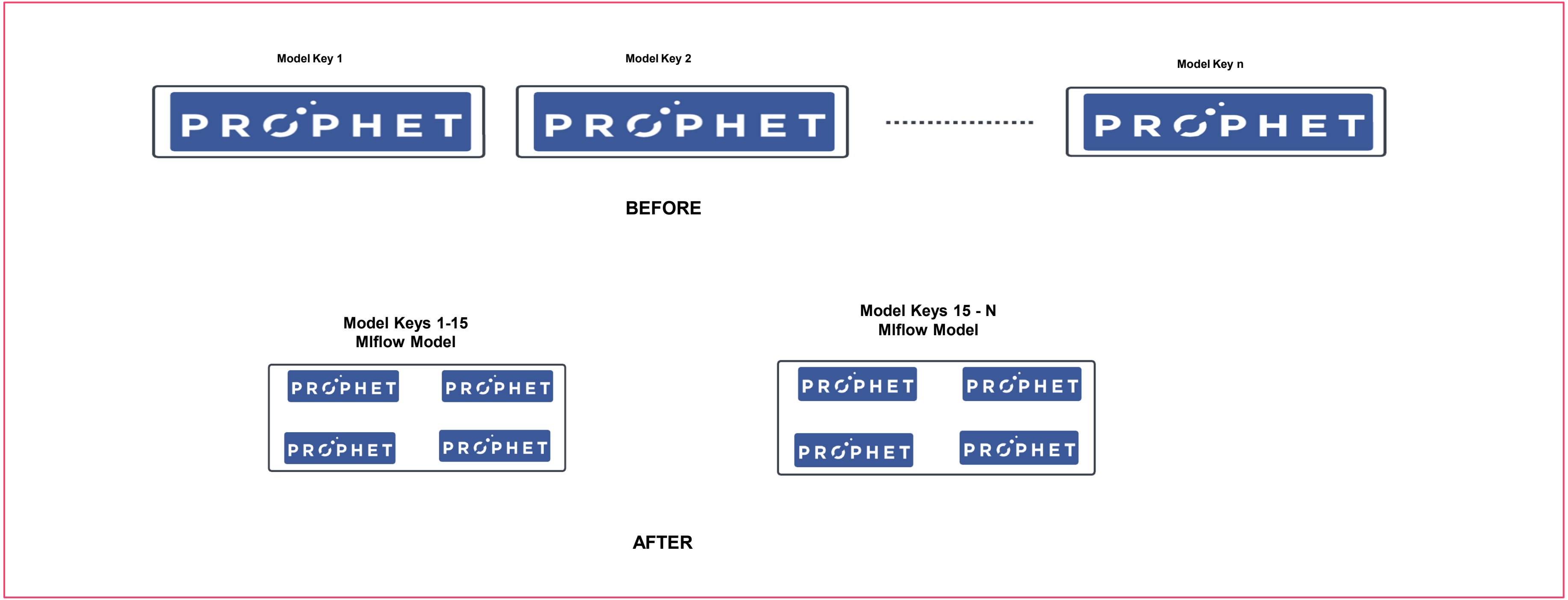


- Databricks Serverless Compute can spin up inference ready compute nodes in range of 5–30 s.
- Mlflow Serving + Serverless compute can help in providing on-demand inference in less than 30s on average.

©2024 Databricks Inc. — All rights reserved

N/W CALL REDUCTION

MLFLOW PYFUNC TO LOAD & INFER MULTIPLE PROPHET MODELS



©2024 Databricks Inc. — All rights reserved

TO ADDRESS MULTIPLE N/W CALLS

Mlflow Pyfunc Class

PYTHON	PYTHON

<pre>class ModelPackager(mlflow.pyfunc.PythonModel):
     def __init__(
          self,
          model_artifact_manager
                   ):
        # dictionary to hold model_names and model artifacts.
        self.models = { }
 	
        # Contains logic on how to load serialized models.
        self.model_manager = model_artifact_manager
 	
     def load_context(self, context):
         # load models from storage in key-value format
         models = self.load_models(context.artifact_path)
 	
         # optionally serialize, compress / quantize models
 	
        # optionally, persist serialized dictionary items in dbfs location</pre>
	<pre>….
	
	def predict(self, context, input_pandas_df, params = None):
	    results = …. # dictionary to store model results by key
	
	     for model_key, single_model_data in
	                                inp_pd_df.groupby(“model_key”).group.items():
	
	         # predicts based on row-batching, for chosen model
	         single_model_predictions =
	self.models[model_key].predict(single_model_data)
	
	         # appends result by model key
	         results[model_key] = single_model_predictions
	
	    return results</pre>

©2024 Databricks Inc. — All rights reserved

TO ADDRESS MODEL MEMORY CONSUMPTION

- For Prophet models, pruning ”history” attribute is one way to reduce model size, but may impact model performance.
- Compressing Prophet model dictionary as-is , through approaches like LZMA, Pickle is simple, efficient and preserves model performance.
- Tensor flow, Pytorch and Sklearn type of models can be optimized for memory through various ”Model Quantization” techniques.

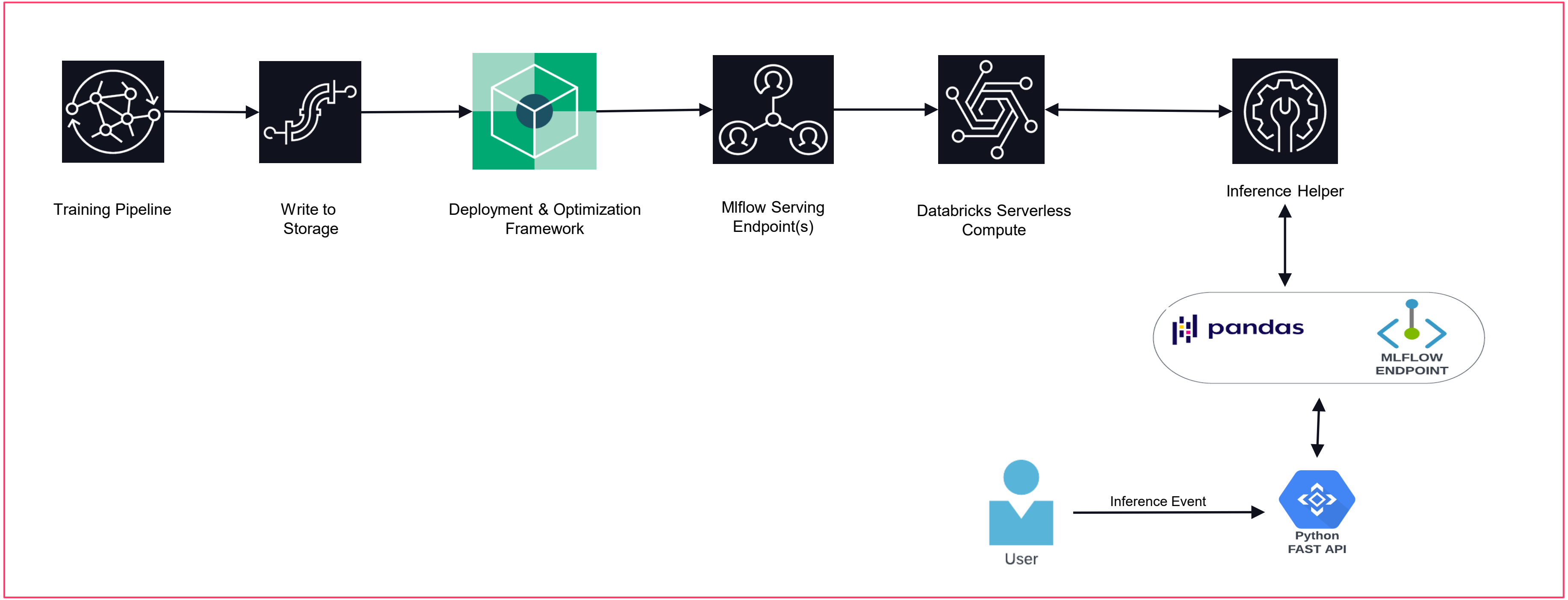
```
]: df_comp.sort_values(by=['bytes'])
```

]:

	Compression	Load_Time_Sec	bytes	perc_save_mem	MB
3	LZMA + Pickle	0.022012	28420	99.045613	0.028420
4	Brotli Compression	0.006791	34888	98.828408	0.034888
1	GZIP + Pickle	0.021448	58977	98.019463	0.058977
2	BZ2 + Pickle	0.033048	62351	97.906158	0.062351
5	Blosc Compression	0.019898	355948	88.046724	0.355948
0	Regular Pickle	0.020988	2977828	0.000000	2.977828

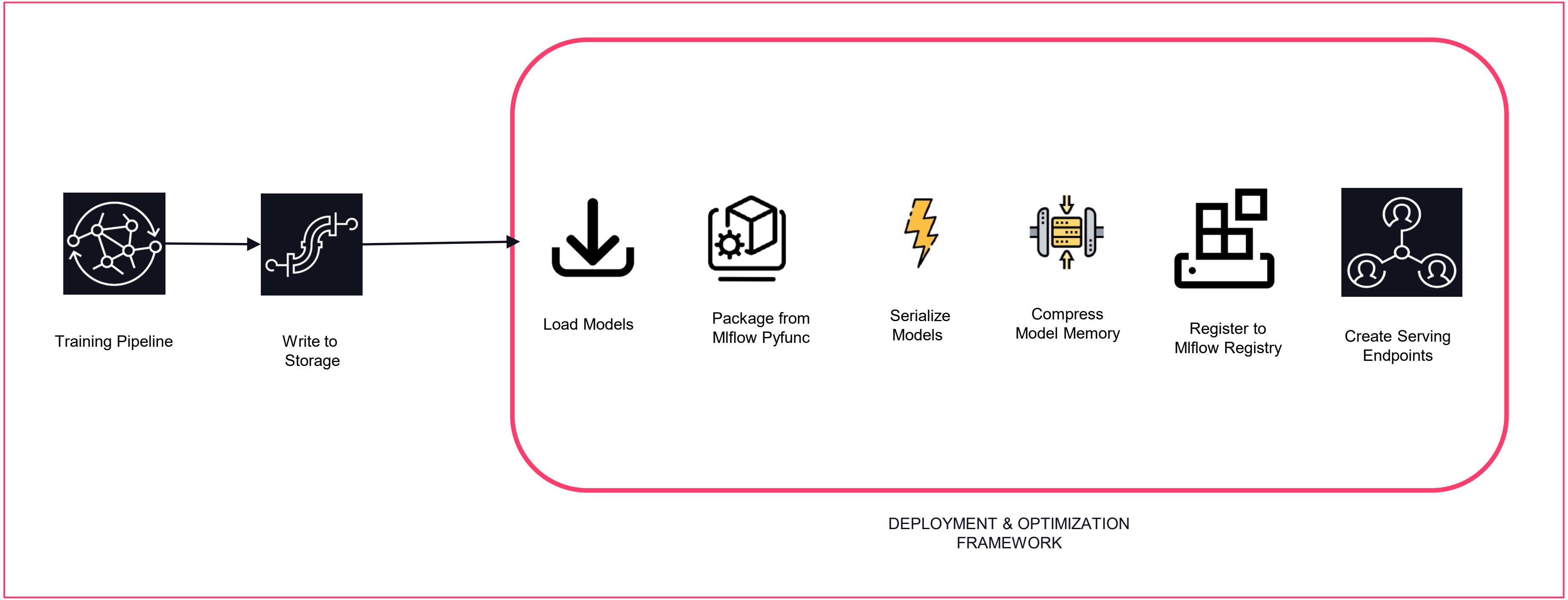
©2024 Databricks Inc. — All rights reserved

OVERALL ARCHITECTURE



©2024 Databricks Inc. — All rights reserved

DEPLOYMENT & OPTIMIZATION FRAMEWORK



©2024 Databricks Inc. — All rights reserved

DATA FORMAT OF INFERENCE REQUEST

- Column "model_key" uniquely identifies which model this record belongs to.
- The inference dataset is then batched both by model and num of models in an individual request.
- HTTP API request is used to call the Databricks serverless API concurrently.

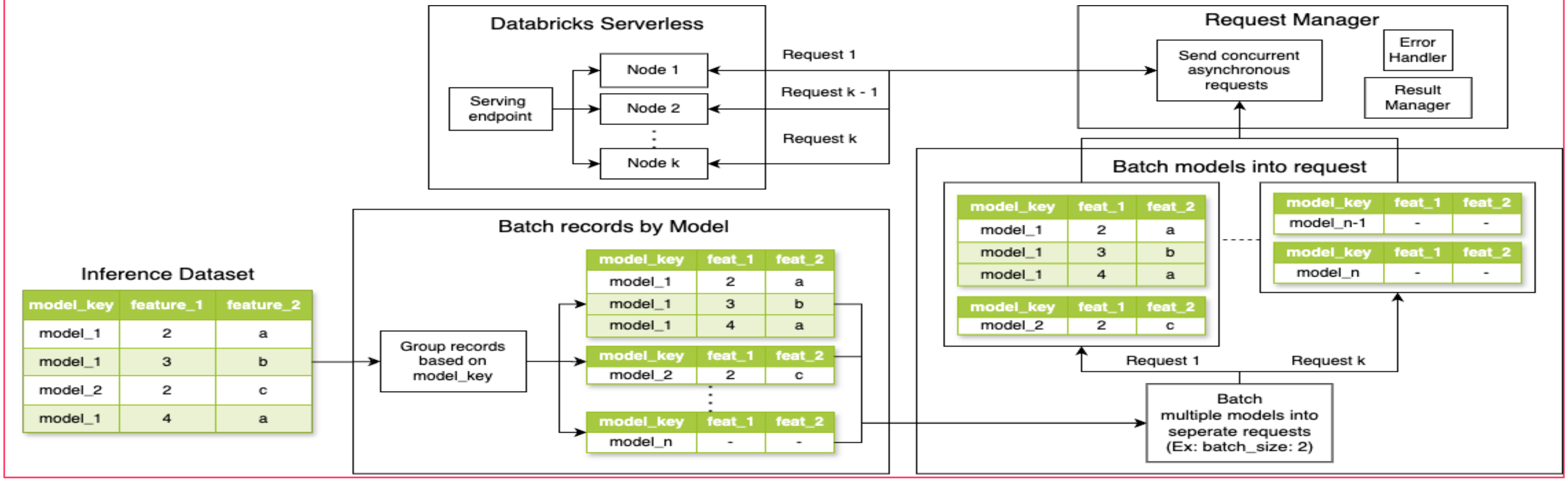
```
import requests
headers = {"Authorization": f"Bearer {token}", "Content-Type": "application/json"}
payload = {
    "params": {"inference_type": "native"},
    "dataframe_split": df.to_dict(orient="split")
}
df = requests.post(endpoint, json=payload, headers=headers).json()
```

Model Key	Feature 1	Feature 2
Model_1	1	34.2
Model_2	12	2.34
Model_1	2	9.75
Model_3	2	21.9

©2024 Databricks Inc. — All rights reserved

SEQUENCE OF INFERENCE REQUEST

Breakdown of Inference Request to Mlflow Serving Endpoint



©2024 Databricks Inc. — All rights reserved

HOW TO CREATE AND MANAGE ENDPOINTS

- One can manage Mlflow Serving endpoints with Databricks API programmatically.

- Supports:
    - Create serving endpoints.
    - Update permissions for serving endpoints.
    - Choose compute and concurrency limits.
    - Deleting serving endpoints.
    - List endpoints / extract metadata.
    - Extract logs for endpoints.

```python
user_permissions = {"username": "CAN_MANAGE"}
uc_model_uri = f"{catalog_name}.{schema_name}.{model_name}"

serving = DatabricksServing(
    token=databricks_token,
    host_url=databricks_host_url,
)
endpoint_info = serving.create_endpoint(
    model_name=uc_model_uri,
    endpoint_name=endpoint_name,
    model_version=model_version,
    workload_size=workload_size,
    tags=tags,
)
serving.update_user_permission_to_endpoint(
    endpoint_id=endpoint_info["id"],
    user_permissions=user_permissions,
)
```


©2024 Databricks Inc. — All rights reserved

HOW TO SEND INFERENCE REQUESTS

Albertsons Companies

- Mlflow Serving endpoints have capability to support 4, 32 and 64 concurrent inference requests, per endpoint.
- Endpoints can be queried in batches, with delay between requests and other standard endpoint query formats like shown in code.

```
import asyncio
from aamp_model_wrapper_inference import config, inference_requests

async def make_inference_request(token: str, endpoint: str) -> pd.DataFrame:
    prediction_df = await inference_requests.run_inference(
        inference_dataset=inference_df,
        token=token,
        batch_size=config.BATCH_SIZE,
        num_concurrent_requests=config.NUM_CONCURRENT_REQUESTS,
        delay_between_concurrent_requests=config.DELAY_BETWEEN_CONCURRENT_REQUESTS,
        endpoint=endpoint,
    )
    return prediction_df

prediction_df = asyncio.run(make_inference_request(token=databricks_token, endpoint=endpoint))
```


©2024 Databricks Inc. — All rights reserved

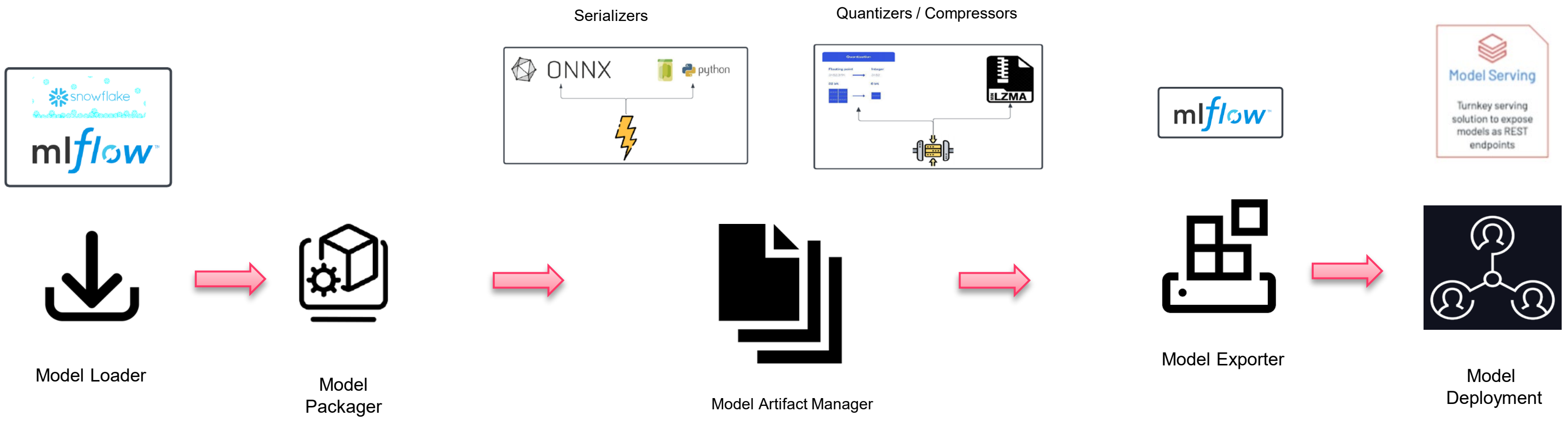
LATENCY BENCHMARKS

APPROACH	NO. OF BATCHED MODELS FOR INFERENCE	NO.OF REQUESTS	NO. OF MODELS TO BE INFERRED, PER REQUEST	NO. OF INFERENCE ROWS, PER MODEL	MEMORY OPTIMIZATION	MODEL INFERENCE TIME (IN SECONDS)
Job Cluster + Batch Inference	1400	1	1400	52	No	350.24
Serverless Compute	10	140	1400	52	No	48.18 (Includes Warm up Time)
Serverless Compute	20	70	1400	52	No	18.44 (Includes Warm up Time)
Serverless Compute	30	46	1400	52	No	18.24 (Includes Warm up Time)
Serverless Compute	10	140	1400	52	Yes	21.23
Serverless Compute	20	70	1400	52	Yes	16.32
Serverless Compute	30	46	1400	52	Yes	14.12

©2024 Databricks Inc. — All rights reserved

EXTENDING TO OTHER MODELS

More Serializers & Compressors



©2024 Databricks Inc. — All rights reserved

OUR TEAM

- Engineering Team

  Amit Adiraju. ; Kshitij, Karthick. ; Aravind, Chamakura ; Vijay, Sriram K ; Shijas, Abdulsalam.

- Data Science Team

  Diana, Adam. ;   Jonas, Krueger.

- Product Team

  Vijay, Nukala. ;   Bonnie, Sarmiento