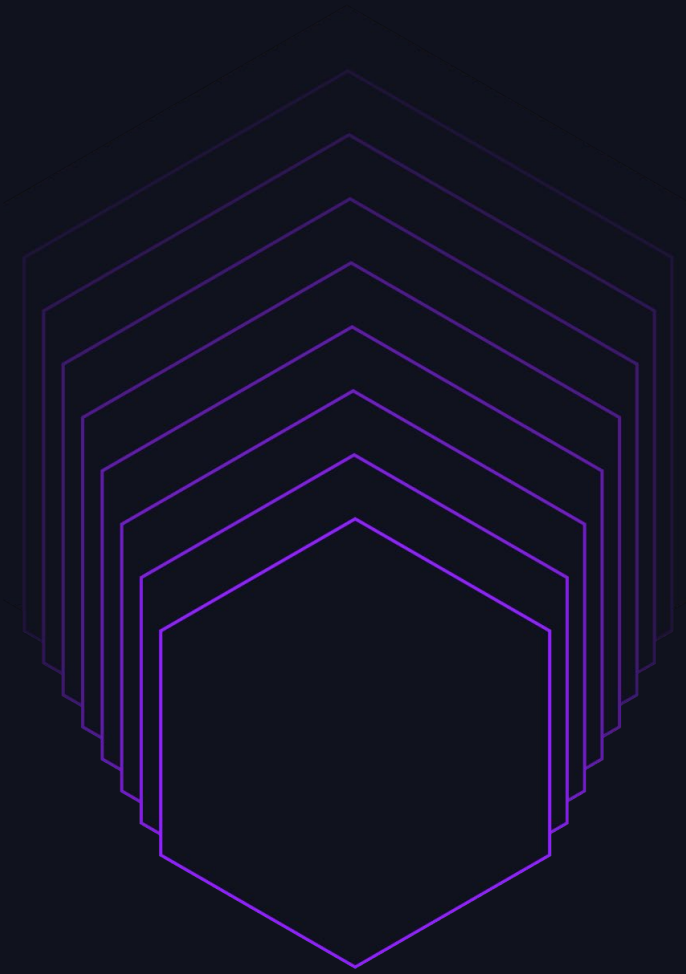
DATA+AI SUMMIT BY databricks

AutoML Time Series Forecasting for Infrastructure Resources



Heping Liu
June 10-13, 2024

Introduction to Workday Inc.

Workday, Inc., is an international cloud-based on-demand human capital management, financial management and student information system software vendor. Workday was founded in 2005.

Type	Public
Traded as	• Nasdaq: WDAY (Class A) • Nasdaq-100 component • Fortune 500
Industry	• Cloud computing • Enterprise software
Founded	March 2005; 19 years ago
Founders	• Dave Duffield • Aneel Bhusri
Headquarters	Pleasanton, California
Area served	Worldwide
CEO	• Carl Eschenbach
Revenue	US$7.26 billion (2024 fiscal year)
Number of employees	18,800 (2024)
Website	workday.com

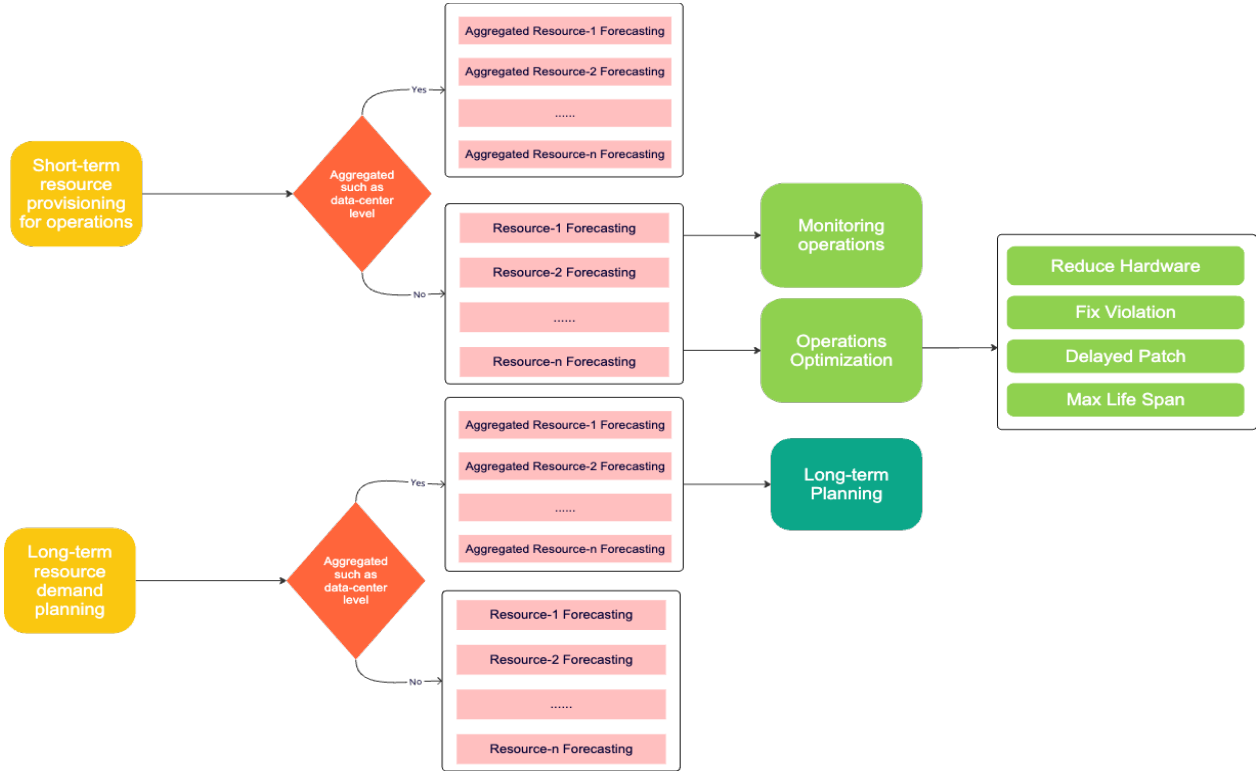
CONTENT

- Why Time Series Forecasting for Infrastructure Resources
- Architecture of Recipe-managed AutoML Time Series Forecasting
- Big Data ETL Pipeline Engine
- Recipe-manageded AutoML Time Series Forecasting Engine
- Application Cases
- Conclusions
- Q & A

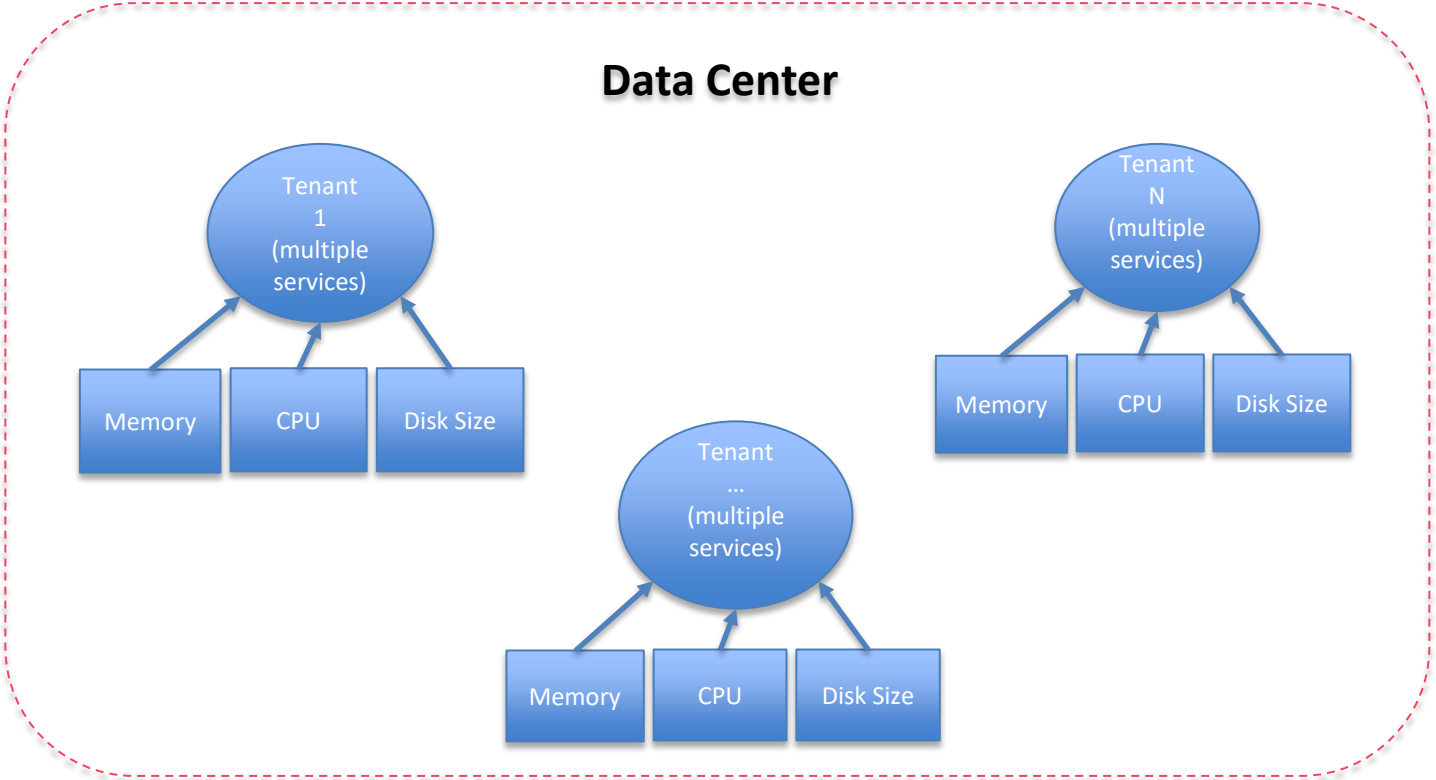
Why Time Series Forecasting for Infrastructure Resources

- The infrastructure resources include storage and computation resources such as physical server, MySQL server, JVM, memory, disk size, CPU, etc.
- The operation monitoring and demand planning require to project the future demand so that these resources can be appropriately provisioned and optimized.
- This talk introduces a recipe-managed AutoML forecasting framework which uses Spark to automate and linearly scale up the fitting and forecasting operations of millions of time series in parallel.
- The forecasting framework can be used to address not only the daily dynamic provision of resources to satisfy the fluctuating transactions, but also the long-term planning to achieve and maintain an effective supply equilibrium.
- The forecasting framework can be extended to many other time series forecasting applications.

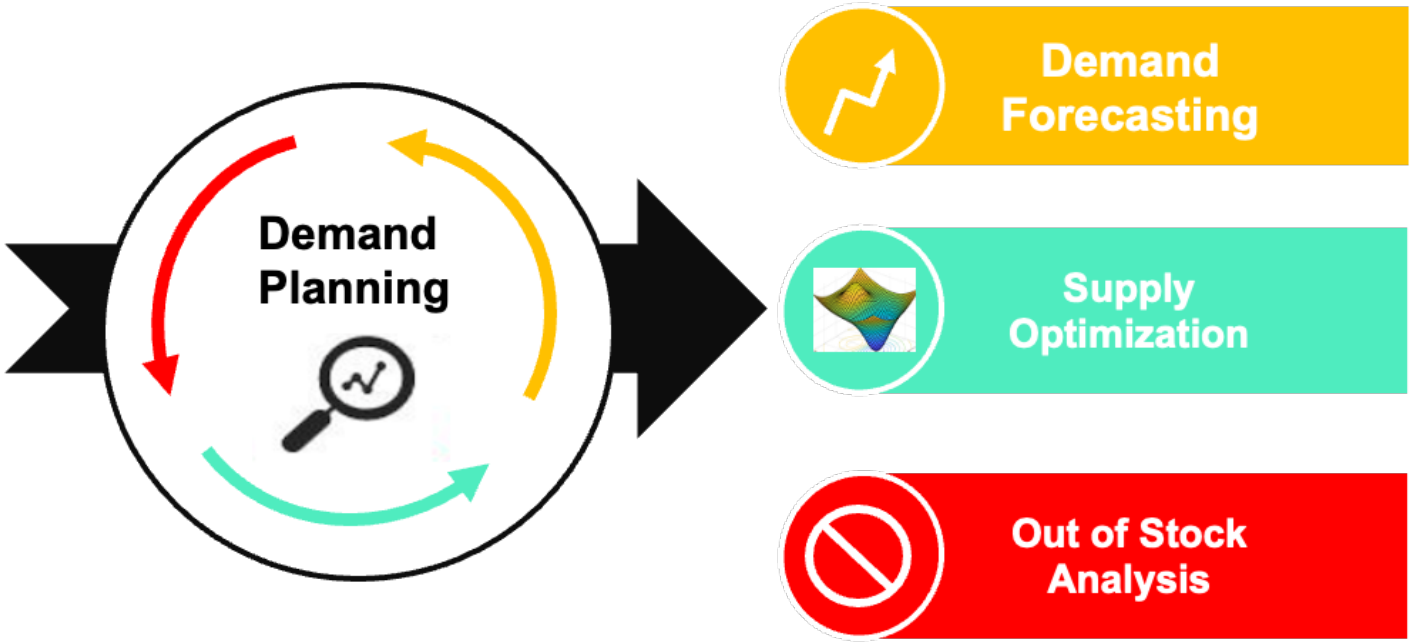
Time Series Forecasting under Different Scenarios



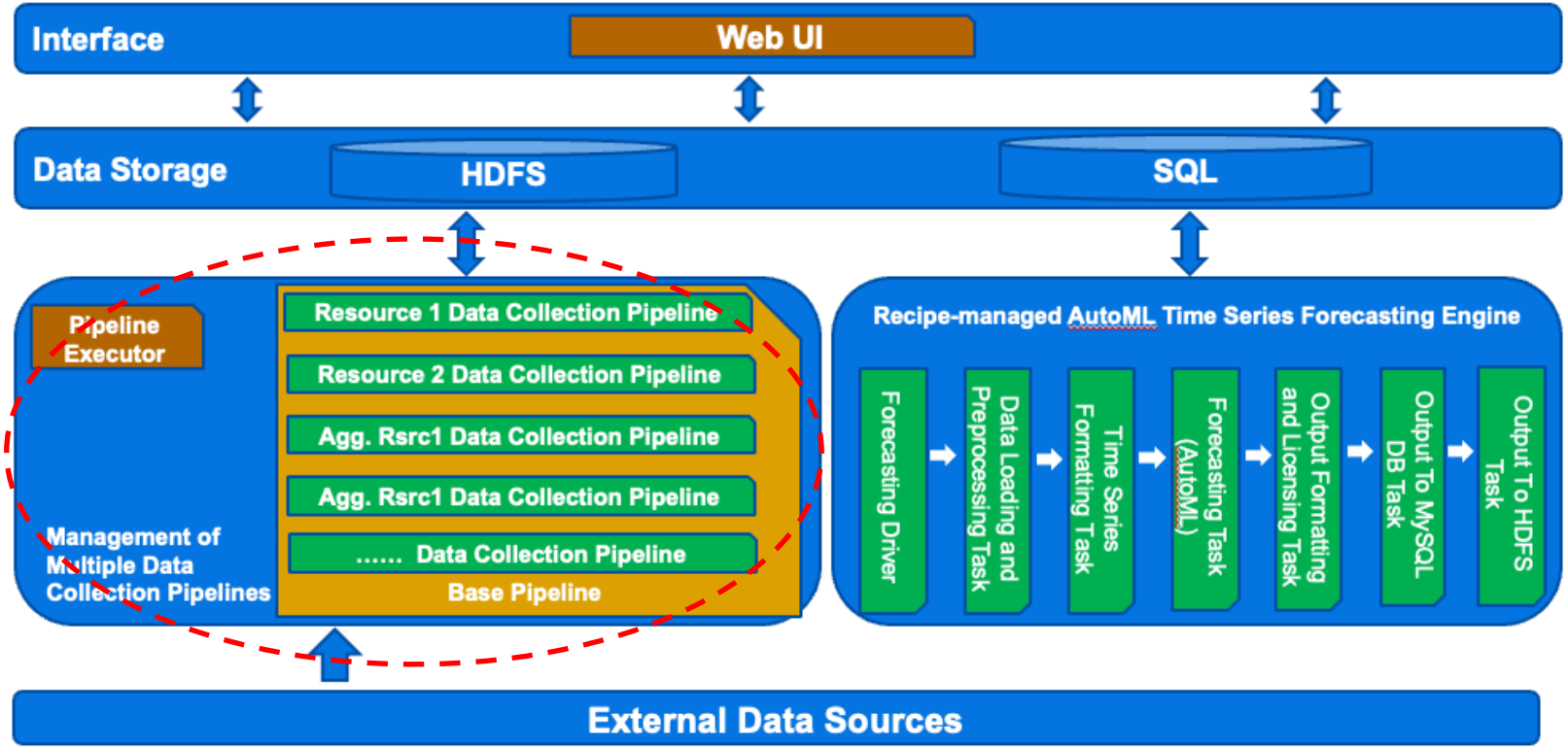
Time Series Forecasting to Provision Resources for Daily Operations



Time Series Forecasting for Long-term Demand Planning



Architecture of Recipe-managed AutoML Time Series Forecasting



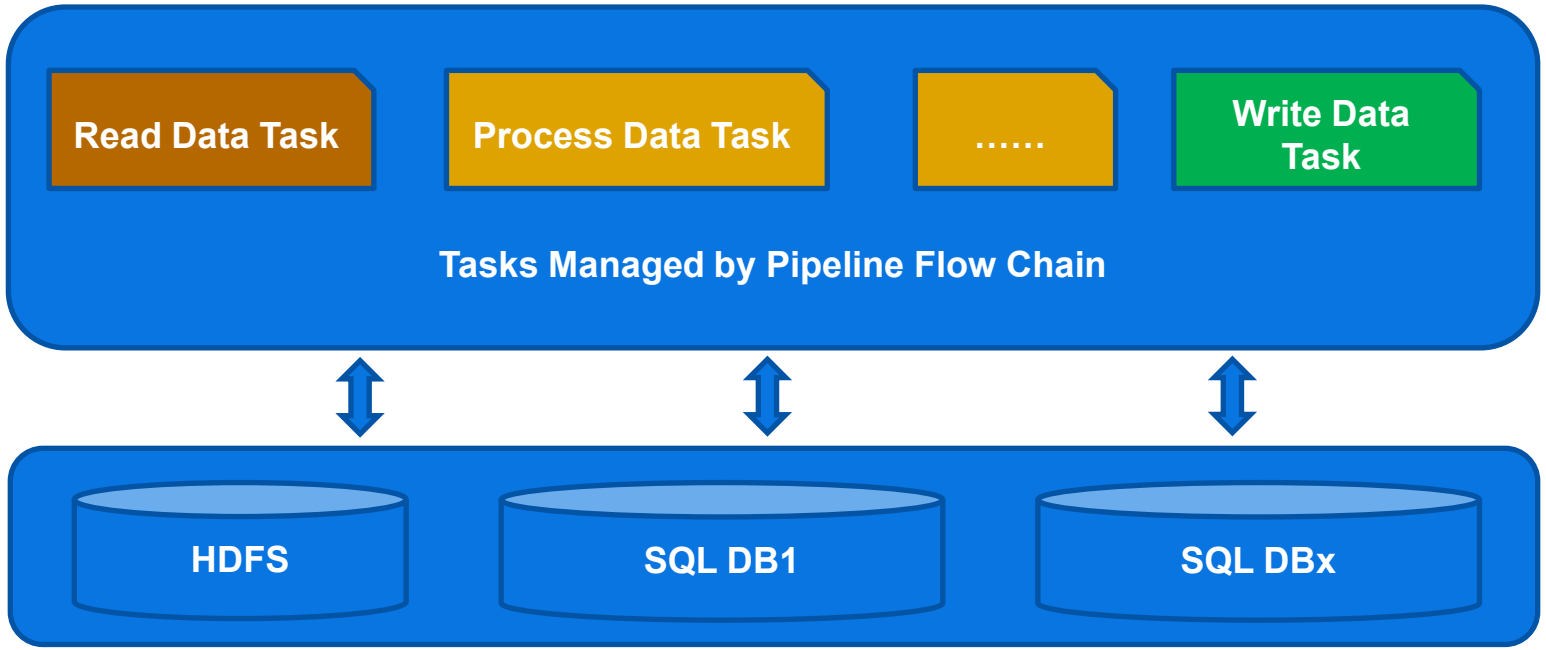
Why Need Big Data ETL Framework

- Data collection has been done by different engineers and the codes may sit in the different repositories, which makes the management and maintenance of codes very difficult.
- The significant number of codes in the different data collections is same or similar and it becomes unnecessary to repeat them.
- A lightweight big data ETL framework is designed, developed and used to collect and process data from the different sources with the different data formats.
- Data collection built on top of the big data ETL framework becomes an important and integral part of the whole forecasting cycle.

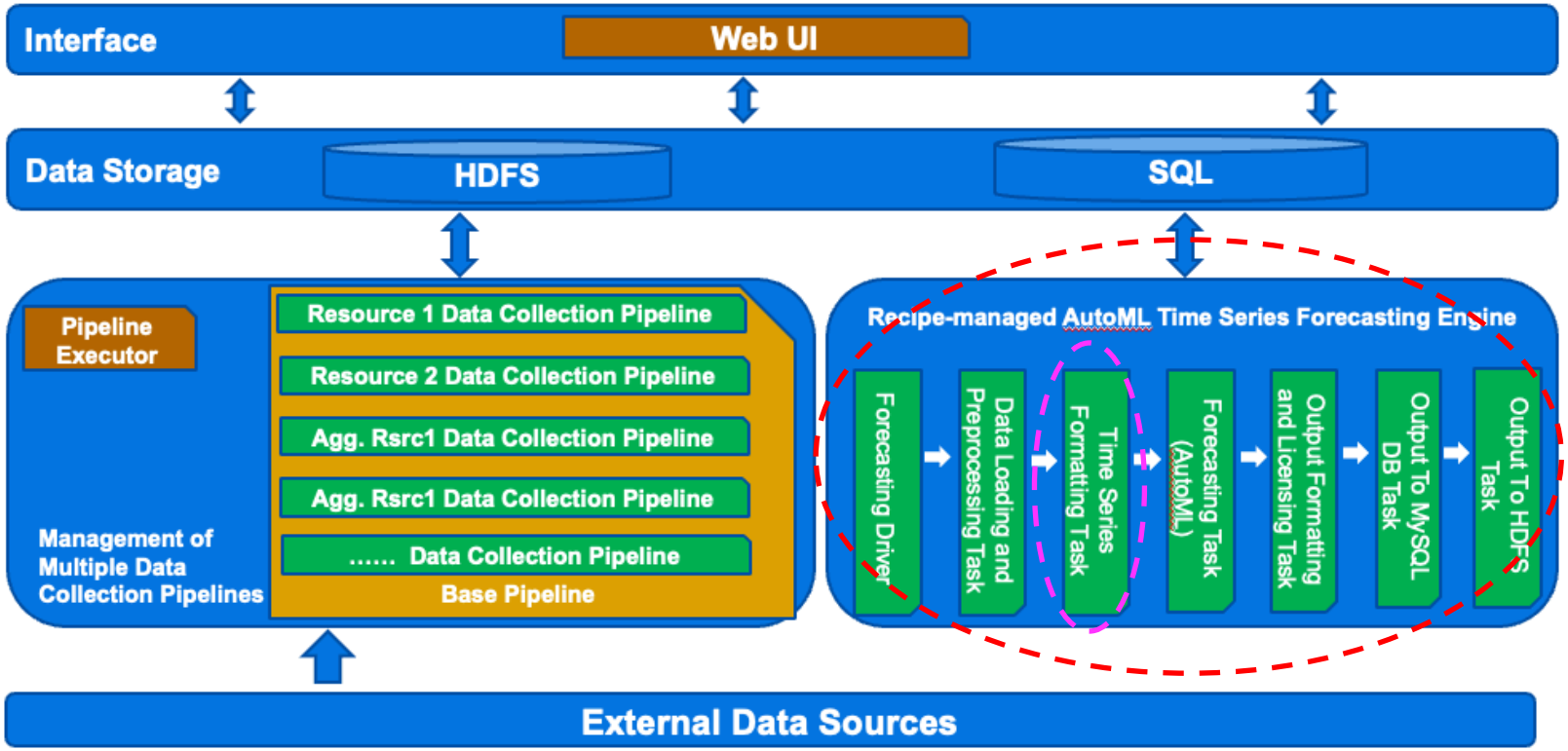
Critical Features of Big Data ETL Framework

- Spark based big data ETL framework which can process/collect the big data and feed it to the forecasting engine.
- Lightweight pipeline framework which can manage multiple heterogenous data collections / processing operations.
- The big data ETL operations are divided into a few independent and modularized tasks, and these modularized tasks are logically managed by a pipeline workflow chain.
- Typesafe based configuration management which can manage the multiple environment configurations and properties of data pipelines.

Architecture of Big Data ETL Framework



Architecture of AutoML-Powered Time Series Forecasting

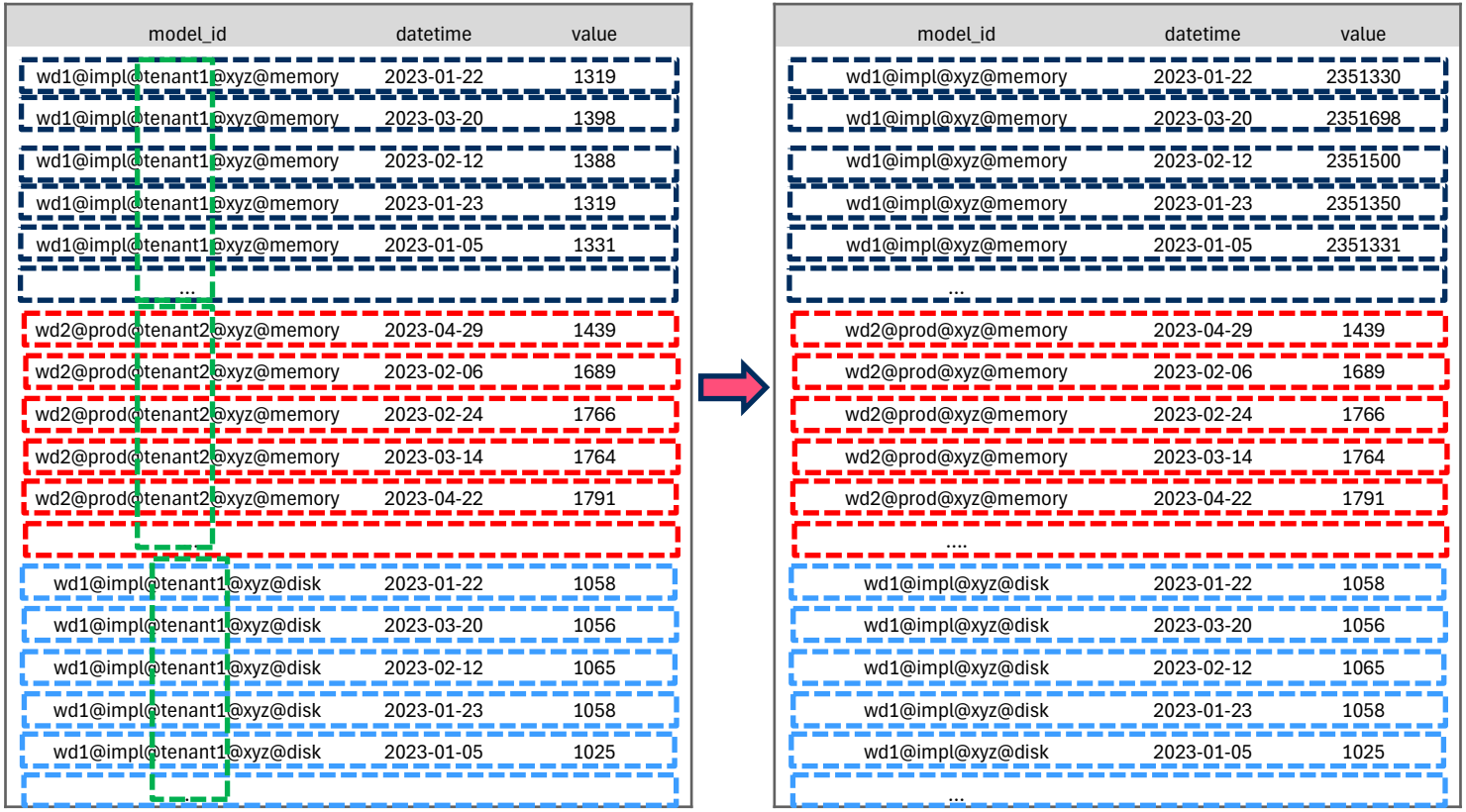


How Tenant Time Series Are Generated?

id	tnt_name	data_center	environment	datetime	service	memory	disk
1	tenant1	WD1	IMPL	2023-01-22	xyz	1319	1058
2	tenant2	WD2	PROD	2023-04-29	xyz	1439	5420
3	tenant3	WD1	IMPL	2023-01-04	xyz	1284	
4	tenant2	WD2	PROD	2023-02-06	xyz	1689	5370
5	tenant1	WD1	IMPL	2023-03-20	xyz	1398	1056
6	tenant4	WD1	IMPL	2023-03-02	xyz	1380	
7	tenant1	WD1	IMPL	2023-02-12	xyz	1388	1065
8	tenant2	WD2	PROD	2023-02-24	xyz	1766	
9	tenant2	WD2	PROD	2023-03-14	xyz	1764	5372
10	tenant6	WD1	IMPL	2023-04-10	xyz	1409	
11	tenant2	WD2	PROD	2023-04-22	xyz	1791	5390
12	tenant1	WD1	IMPL	2023-01-23	xyz	1319	1058
13	tenant2	WD2	PROD	2023-04-04	xyz	1769	5396
14	tenant1	WD1	IMPL	2023-01-05	xyz	1331	1025
15	tenant2	WD2	PROD	2023-01-16	xyz	1672	5368
...							

model_id	datetime	value
wd1@impl@tenant1@xyz@memory	2023-01-22	1319
wd1@impl@tenant1@xyz@memory	2023-03-20	1398
wd1@impl@tenant1@xyz@memory	2023-02-12	1388
wd1@impl@tenant1@xyz@memory	2023-01-23	1319
wd1@impl@tenant1@xyz@memory	2023-01-05	1331
...		
wd2@prod@tenant2@xyz@memory	2023-04-29	1439
wd2@prod@tenant2@xyz@memory	2023-02-06	1689
wd2@prod@tenant2@xyz@memory	2023-02-24	1766
wd2@prod@tenant2@xyz@memory	2023-03-14	1764
wd2@prod@tenant2@xyz@memory	2023-04-22	1791
....		
wd1@impl@tenant1@xyz@disk	2023-01-22	1058
wd1@impl@tenant1@xyz@disk	2023-03-20	1056
wd1@impl@tenant1@xyz@disk	2023-02-12	1065
wd1@impl@tenant1@xyz@disk	2023-01-23	1058
wd1@impl@tenant1@xyz@disk	2023-01-05	1025
...		

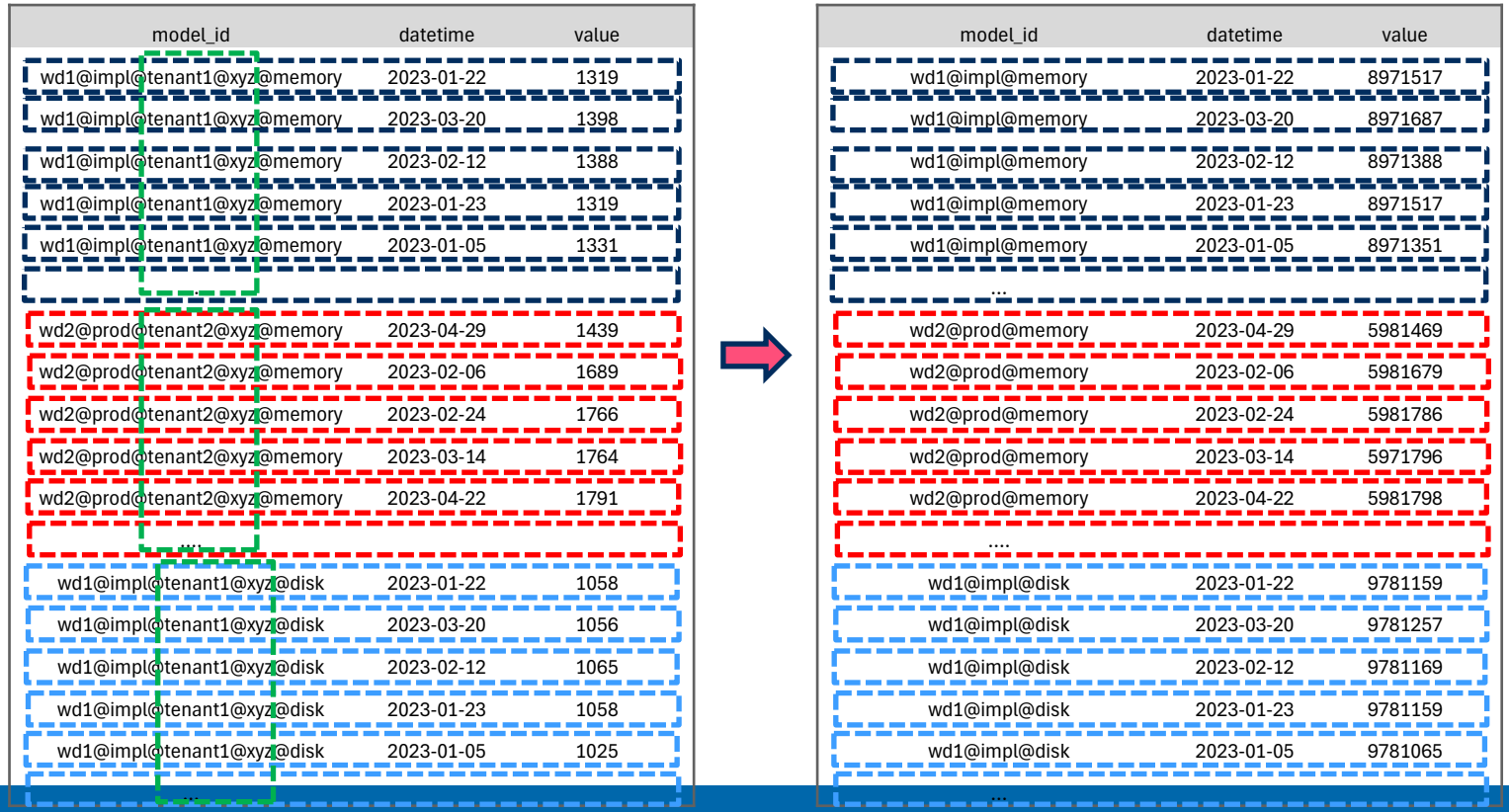
How Service Time Series Are Generated?



model_id	datetime	value
wd1@impl@tenant1@xyz@memory	2023-01-22	1319
wd1@impl@tenant1@xyz@memory	2023-03-20	1398
wd1@impl@tenant1@xyz@memory	2023-02-12	1388
wd1@impl@tenant1@xyz@memory	2023-01-23	1319
wd1@impl@tenant1@xyz@memory	2023-01-05	1331
...		
wd2@prod@tenant2@xyz@memory	2023-04-29	1439
wd2@prod@tenant2@xyz@memory	2023-02-06	1689
wd2@prod@tenant2@xyz@memory	2023-02-24	1766
wd2@prod@tenant2@xyz@memory	2023-03-14	1764
wd2@prod@tenant2@xyz@memory	2023-04-22	1791
wd1@impl@tenant1@xyz@disk	2023-01-22	1058
wd1@impl@tenant1@xyz@disk	2023-03-20	1056
wd1@impl@tenant1@xyz@disk	2023-02-12	1065
wd1@impl@tenant1@xyz@disk	2023-01-23	1058
wd1@impl@tenant1@xyz@disk	2023-01-05	1025

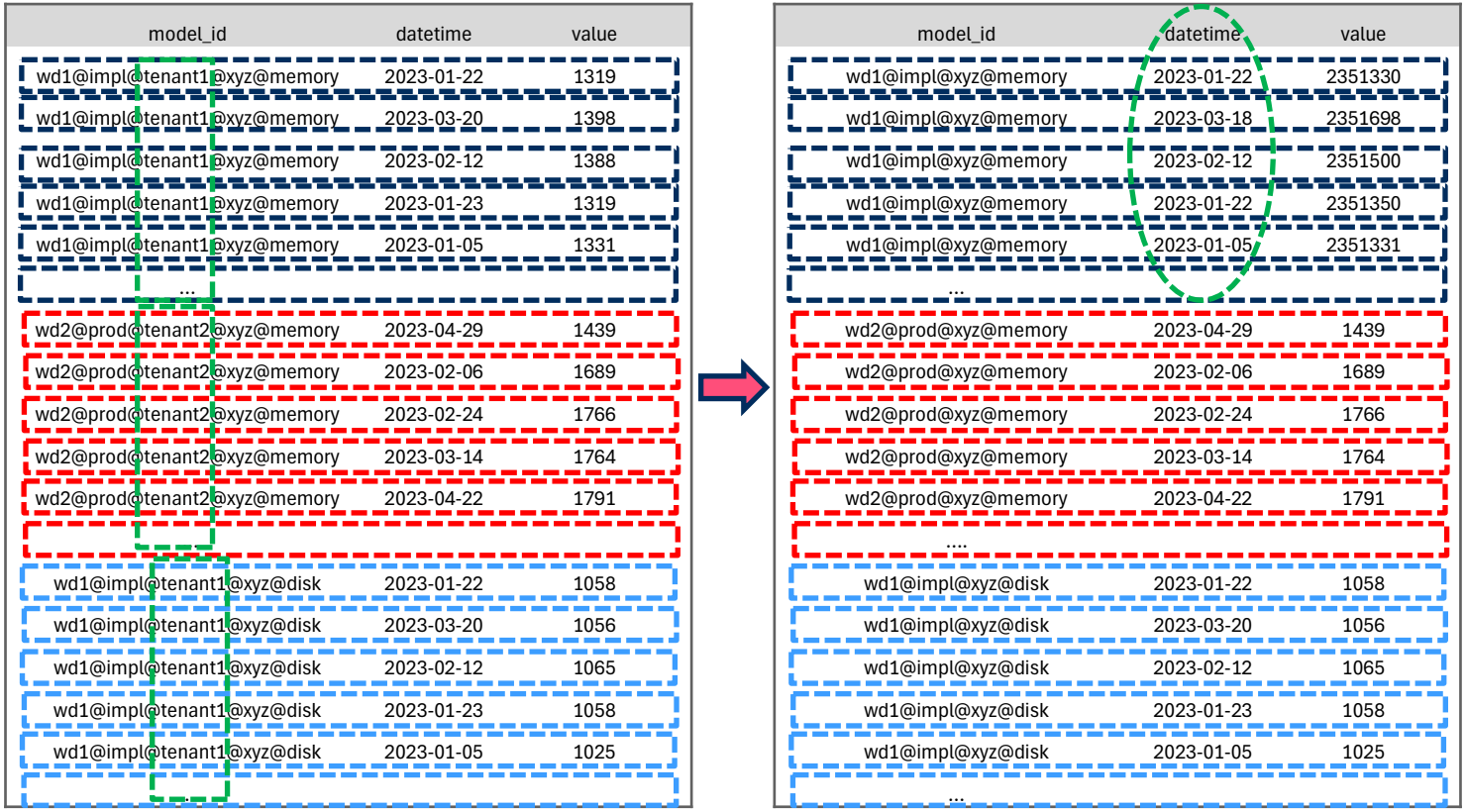
model_id	datetime	value
wd1@impl@xyz@memory	2023-01-22	2351330
wd1@impl@xyz@memory	2023-03-20	2351698
wd1@impl@xyz@memory	2023-02-12	2351500
wd1@impl@xyz@memory	2023-01-23	2351350
wd1@impl@xyz@memory	2023-01-05	2351331
...		
wd2@prod@xyz@memory	2023-04-29	1439
wd2@prod@xyz@memory	2023-02-06	1689
wd2@prod@xyz@memory	2023-02-24	1766
wd2@prod@xyz@memory	2023-03-14	1764
wd2@prod@xyz@memory	2023-04-22	1791
....		
wd1@impl@xyz@disk	2023-01-22	1058
wd1@impl@xyz@disk	2023-03-20	1056
wd1@impl@xyz@disk	2023-02-12	1065
wd1@impl@xyz@disk	2023-01-23	1058
wd1@impl@xyz@disk	2023-01-05	1025
...		

How Data Center Time Series Are Generated?

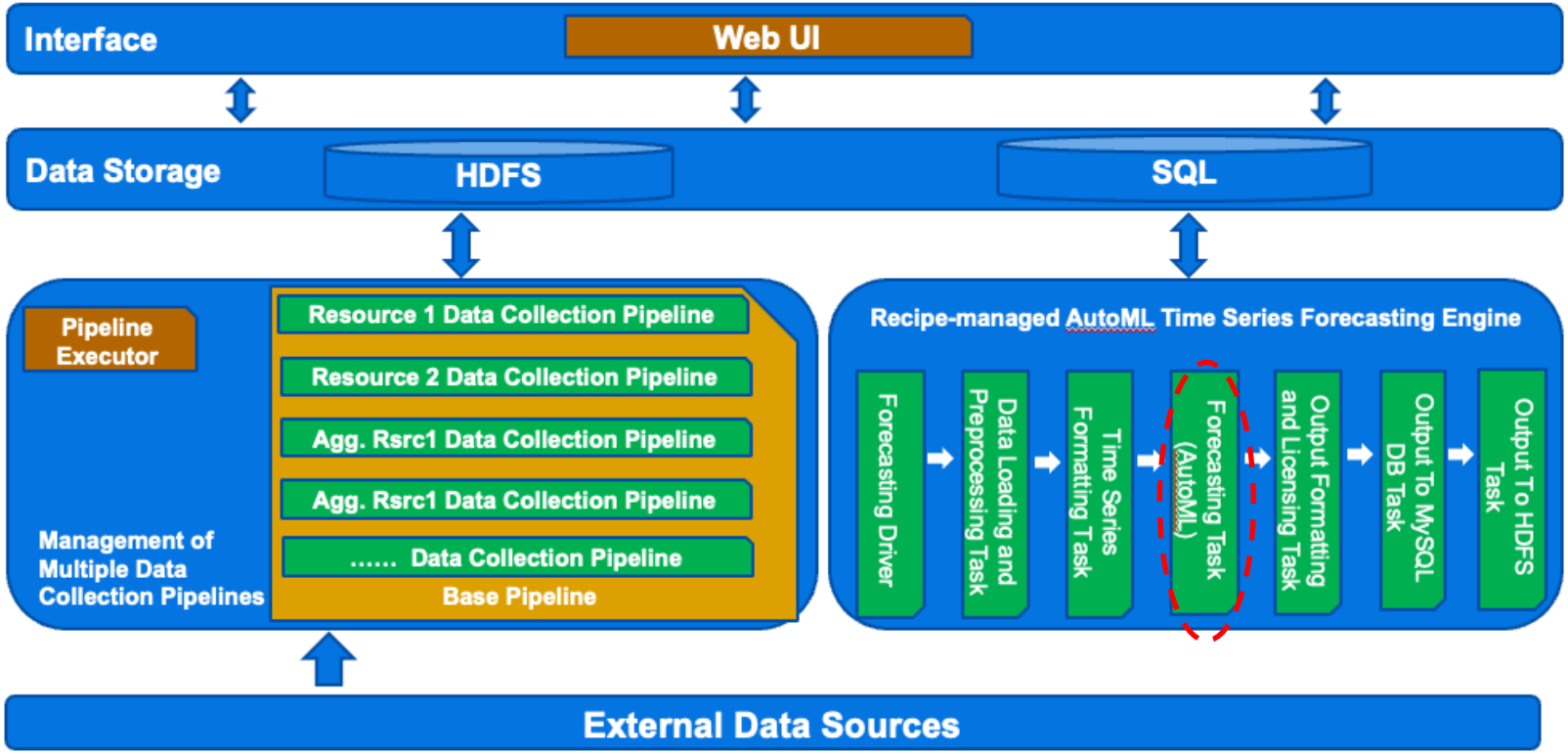


workday.

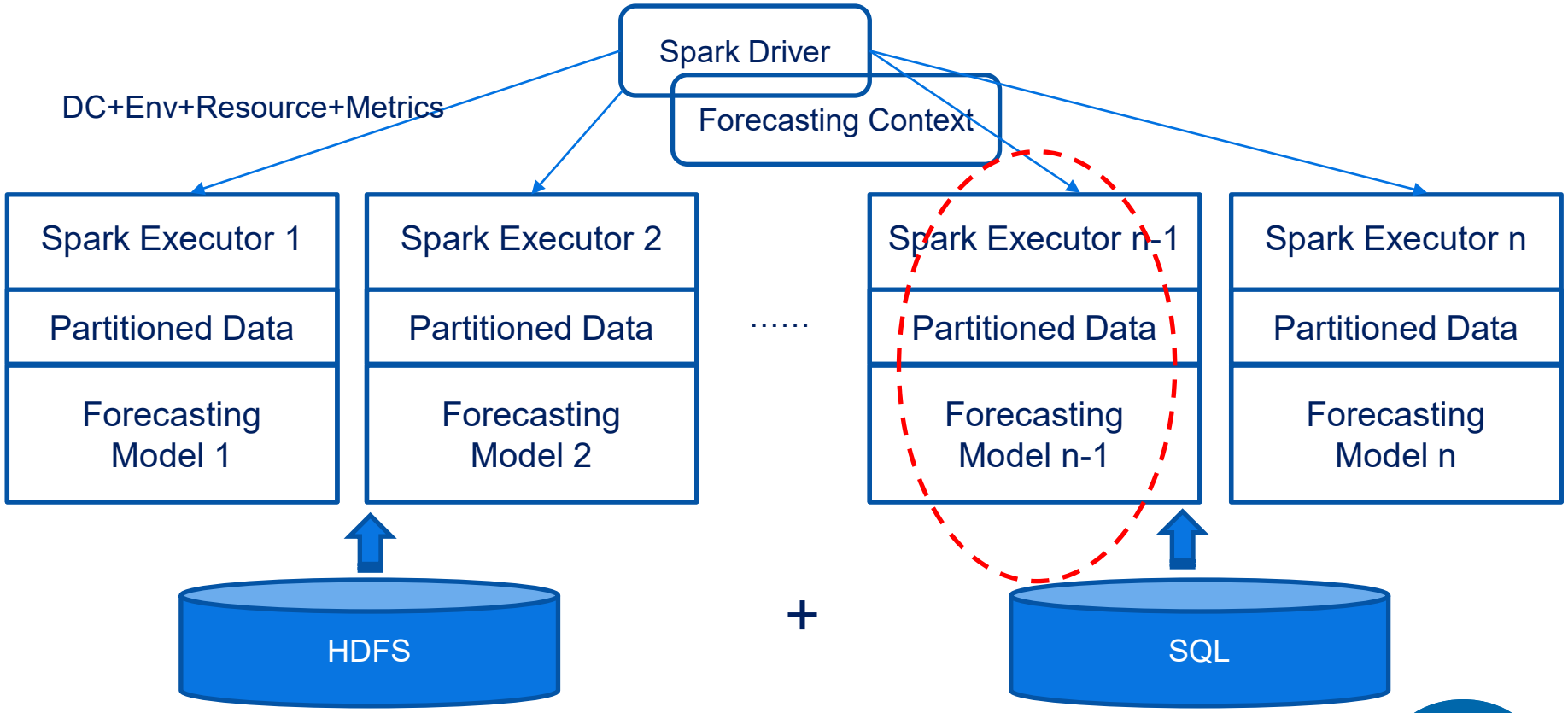
How Time Series with Different Frequencies Are Generated?



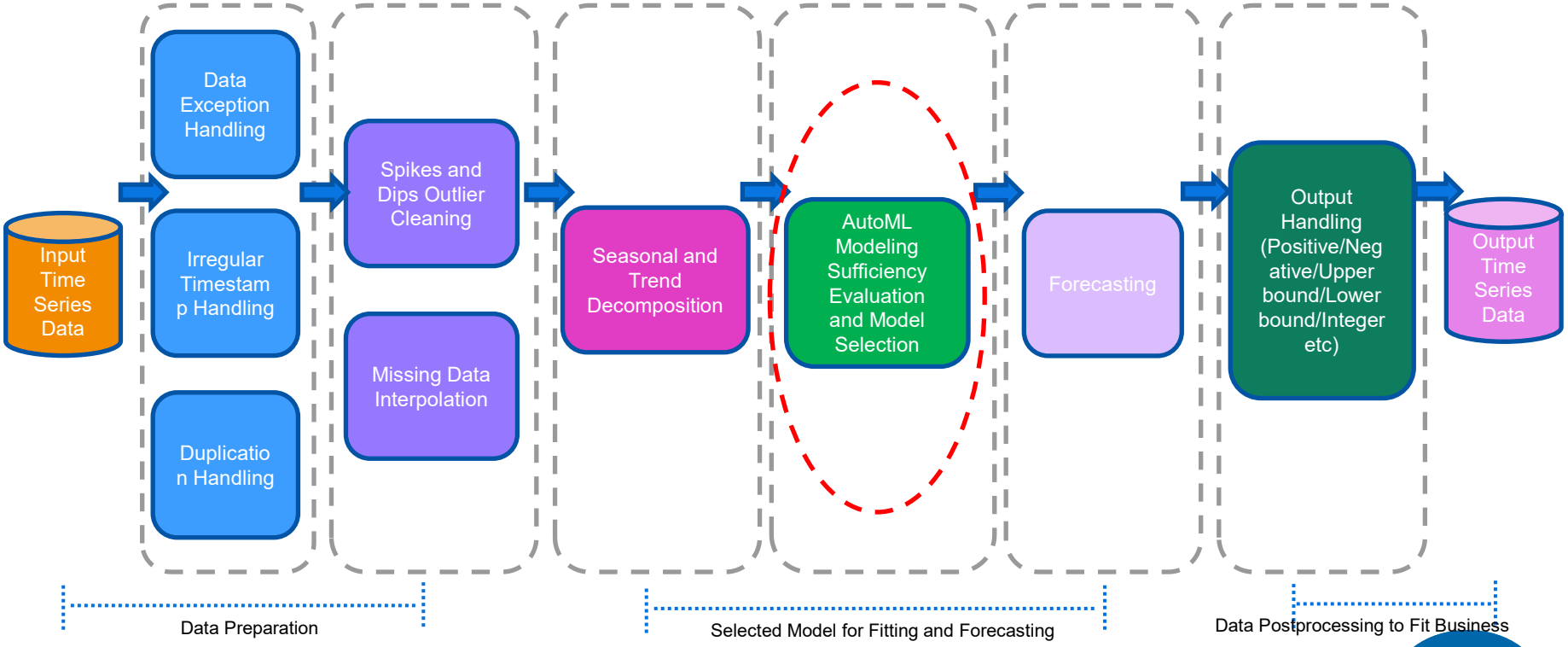
Architecture of AutoML-Powered Time Series Forecasting



Data and Model Scalability: Spark Based Distribution Forecasting Engine



AutoML Time Series Forecasting Modeling



ARIMA Time Series Modeling Used in AutoML

$$x_t = \sum_{i=1}^{p} \varphi_i x_{t-i} + \sum_{j=1}^{q} \theta_j \omega_{t-j}$$

- *ARIMA(p, d, q)* -models are a mix of *autoregressive (AR)* and *moving average (MA)* models. AR-models relate the current value x_t of a process to a finite, linear combination of previous values of the process and a random noise ω. They get abbreviated as *AR*(*p*), where *p* describes the order. Instead, MA-models represent x_t linearly dependent on a finite number *q* of previous random noise ω's . *d* is a difference factor to remove the unit root and make the data stationary.
- Time series data may have the seasonal pattern. Differencing the data at lag *h* can help decompose the seasonal effect.
- The model may introduce some exogenous variables to make the forecasting results more accurate.

Different Models Can Be Used in AutoML

Prophet (Additive model for time series forecasting)

Formula:

- Model Equation: $y(t) = g(t) + s(t) + h(t) + \epsilon_t$
  Where:
  - $g(t)$ is the trend component.
  - $s(t)$ is the seasonal component.
  - $h(t)$ is the holiday component.
  - ϵ_t is the error term.

Kalman Filter (State space model)

Formula:

- State Prediction: $\hat{x}_{t|t-1} = A\hat{x}_{t-1|t-1} + Bu_t$
- Covariance Prediction: $P_{t|t-1} = AP_{t-1|t-1}A^T + Q$
- Kalman Gain: $K_t = P_{t|t-1}H^T(HP_{t|t-1}H^T + R)^{-1}$
- State Update: $\hat{x}_{t|t} = \hat{x}_{t|t-1} + K_t(y_t - H\hat{x}_{t|t-1})$
- Covariance Update: $P_{t|t} = (I - K_tH)P_{t|t-1}$

LightGBM (Gradient boosting framework by Microsoft)

Formula:

- Predicted Value: $\hat{y} = \sum_{k=1}^{K} f_k(x)$
  Where $f_k(x)$ is the k-th tree in the ensemble.
- Objective Function: Minimize the loss function (e.g., Mean Squared Error) with added regularization term.

Holt-Winters (Exponential smoothing model with trend and seasonality)

Formula:

$\hat{y}_{t+h} = (L_t + hT_t)S_{t+h-m(k+1)}$

Where:

- $\hat{y}_{t+h}$ is the forecast for h periods ahead.
- L_t is the level component.
- T_t is the trend component.
- S_t is the seasonal component.
- m is the length of the seasonal cycle.
- k is the integer part of $(h-1)/m$.

Update Equations:

- Level: $L_t = \alpha\frac{y_t}{S_{t-m}} + (1-\alpha)(L_{t-1} + T_{t-1})$
- Trend: $T_t = \beta(L_t - L_{t-1}) + (1-\beta)T_{t-1}$
- Seasonality: $S_t = \gamma\frac{y_t}{L_t} + (1-\gamma)S_{t-m}$

Different Models Can Be Used in AutoML

Croston TSB (Croston's method with demand interval probability updated every period)

Formula:

$\hat{y}_{t+1} = \hat{z}_t \hat{p}_t$

Where:

- $\hat{y}_{t+1}$ is the forecast for the next period.
- $\hat{z}_t$ is the estimate of the demand size.
- $\hat{p}_t$ is the estimate of the probability of a demand occurring.

Update Equations:

- Demand Size Update: $\hat{z}_t = \alpha z_t + (1 - \alpha)\hat{z}_{t-1}$
- Probability Update: $\hat{p}_t = \beta I_t + (1 - \beta)\hat{p}_{t-1}$

Croston (Exponential smoothing model used for intermittent demand forecasting)

Formula:

$\hat{y}_{t+1} = \frac{\hat{z}_t}{\hat{p}_t}$

Where:

- $\hat{y}_{t+1}$ is the forecast for the next period.
- $\hat{z}_t$ is the estimate of the demand size.
- $\hat{p}_t$ is the estimate of the interval between non-zero demands.

Update Equations:

- Demand Size Update: $\hat{z}_t = \alpha z_t + (1 - \alpha)\hat{z}_{t-1}$
- Demand Interval Update: $\hat{p}_t = \beta p_t + (1 - \beta)\hat{p}_{t-1}$

LSTM Time Series Modeling in Forecasting Platform

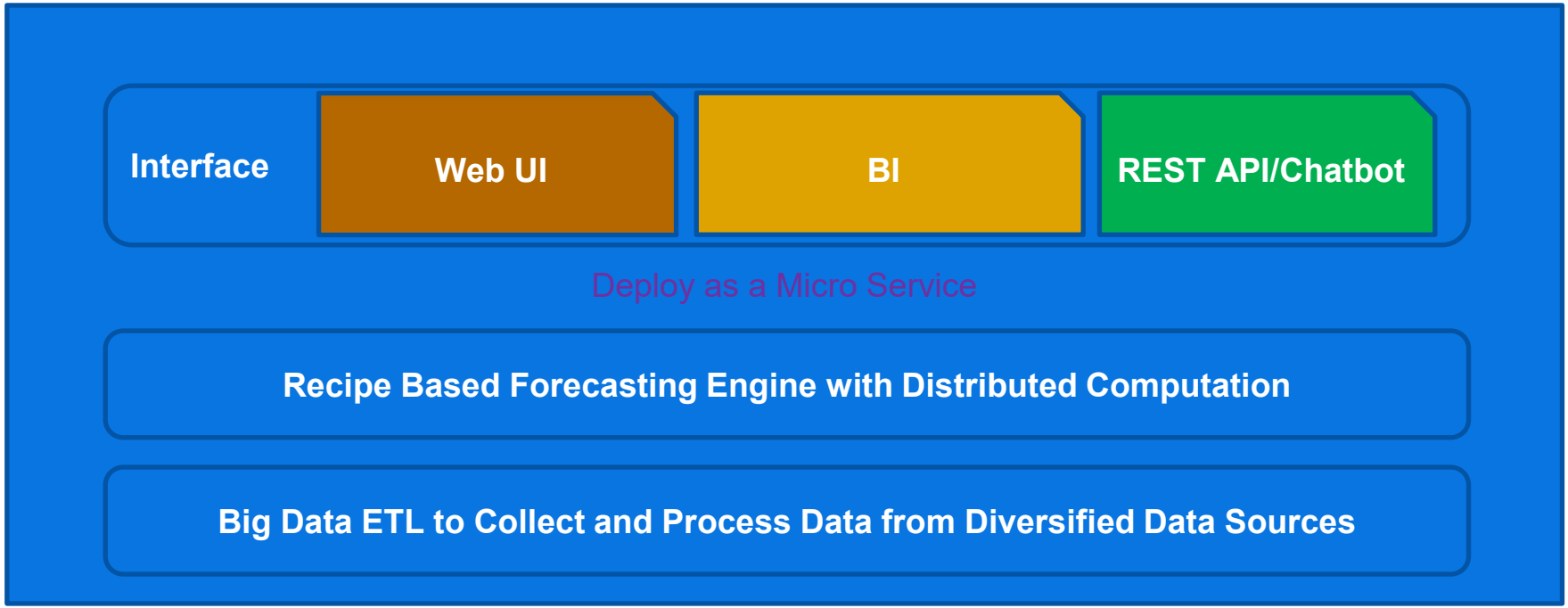
Title: LSTM Time Series Modeling in Forecasting Platform

Content:

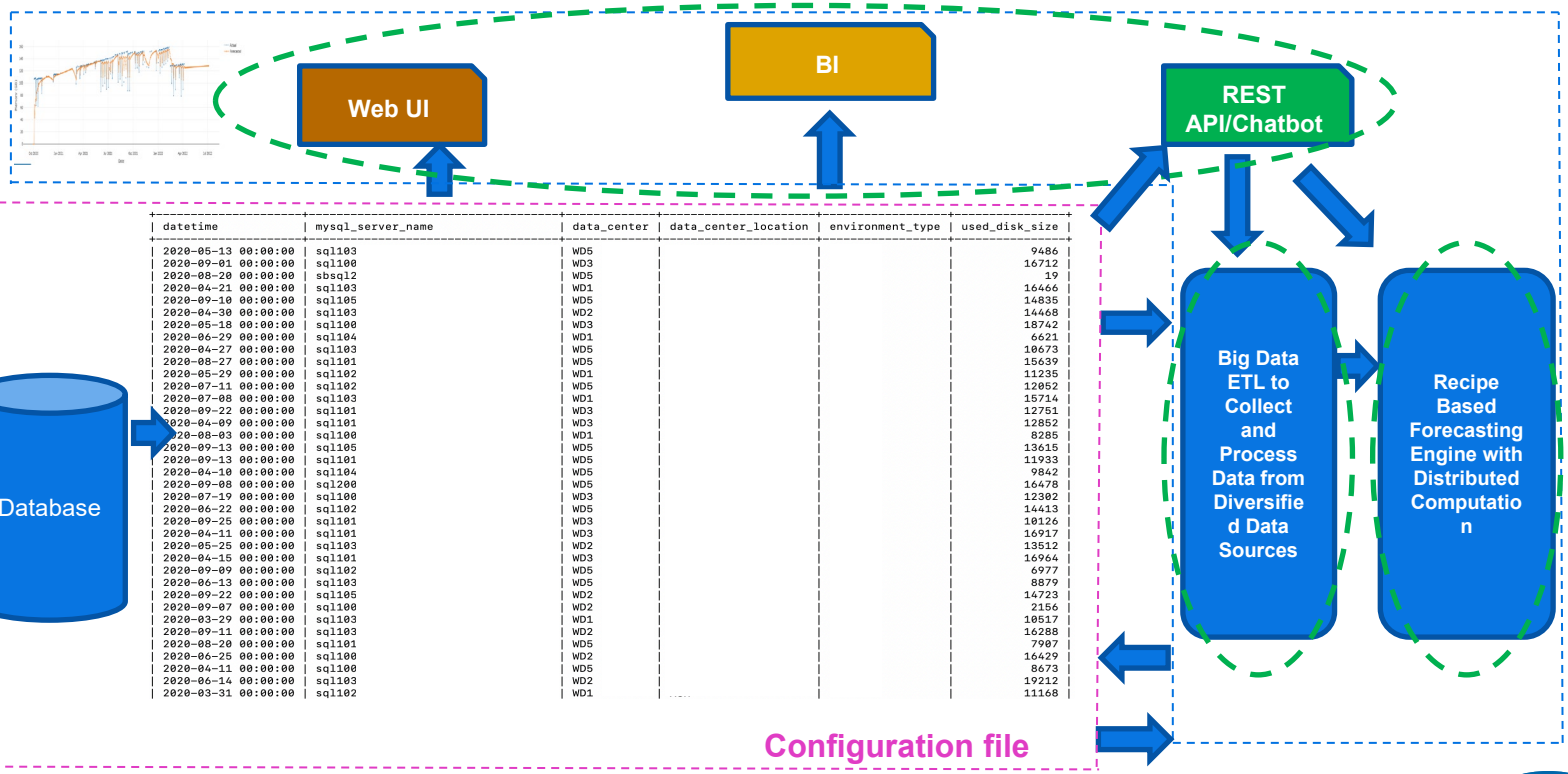
- **Definition:** Long Short-Term Memory (LSTM) is a type of Recurrent Neural Network (RNN) designed to model temporal sequences and long-range dependencies.
- **Components:**
  - **Cell State:** Maintains the memory of the network.
  - **Gates:**
    - **Forget Gate:** Decides what information to discard from the cell state.
    - **Input Gate:** Decides what new information to store in the cell state.
    - **Output Gate:** Decides the output based on the cell state.
- **Application:** LSTMs are particularly effective for time series forecasting, especially when there are long-term dependencies in the data.
- **Model Equations:**
  - Forget Gate:
    $f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$
  - Input Gate:
    $i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$
  - Cell State:
    $\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$
  - Cell Update:
    $C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$
  - Output Gate:
    $o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$
  - Hidden State:
    $h_t = o_t * \tanh(C_t)$
- **Extensions:** Includes Bidirectional LSTMs and Stacked LSTMs.

Would you like to proceed with creating the PowerPoint slides for these models, or do you need any modifications or additional information?

Forecasting as a Service (SEER)



Forecasting as a Service (SEER)



datetime	mysql_server_name	data_center	data_center_location	environment_type	used_disk_size
2020-05-13 00:00:00	sql103	WD5			9486
2020-09-01 00:00:00	sql100	WD3			16712
2020-08-20 00:00:00	sbsql2	WD5			19
2020-04-21 00:00:00	sql103	WD1			16466
2020-09-10 00:00:00	sql105	WD5			14835
2020-04-30 00:00:00	sql103	WD2			14468
2020-05-18 00:00:00	sql100	WD3			18742
2020-06-29 00:00:00	sql104	WD1			6621
2020-04-27 00:00:00	sql103	WD5			10673
2020-08-27 00:00:00	sql101	WD5			15639
2020-05-29 00:00:00	sql102	WD1			11235
2020-07-11 00:00:00	sql102	WD5			12052
2020-07-08 00:00:00	sql103	WD1			15714
2020-09-22 00:00:00	sql101	WD3			12751
2020-04-09 00:00:00	sql101	WD3			12852
2020-08-03 00:00:00	sql100	WD1			8285
2020-09-13 00:00:00	sql105	WD5			13615
2020-09-13 00:00:00	sql101	WD5			11933
2020-04-10 00:00:00	sql104	WD5			9842
2020-09-08 00:00:00	sql200	WD5			16478
2020-07-19 00:00:00	sql100	WD3			12302
2020-06-22 00:00:00	sql102	WD5			14413
2020-09-25 00:00:00	sql101	WD3			10126
2020-04-11 00:00:00	sql101	WD3			16917
2020-05-25 00:00:00	sql103	WD2			13512
2020-04-15 00:00:00	sql101	WD3			16964
2020-09-09 00:00:00	sql102	WD5			6977
2020-06-13 00:00:00	sql103	WD5			8879
2020-09-22 00:00:00	sql105	WD2			14723
2020-09-07 00:00:00	sql100	WD2			2156
2020-03-29 00:00:00	sql103	WD1			10517
2020-09-11 00:00:00	sql103	WD2			16288
2020-08-20 00:00:00	sql101	WD5			7907
2020-06-25 00:00:00	sql100	WD2			16429
2020-04-11 00:00:00	sql100	WD5			8673
2020-06-14 00:00:00	sql103	WD2			19212
2020-03-31 00:00:00	sql102	WD1			11168

Forecasting as a Service (SEER)

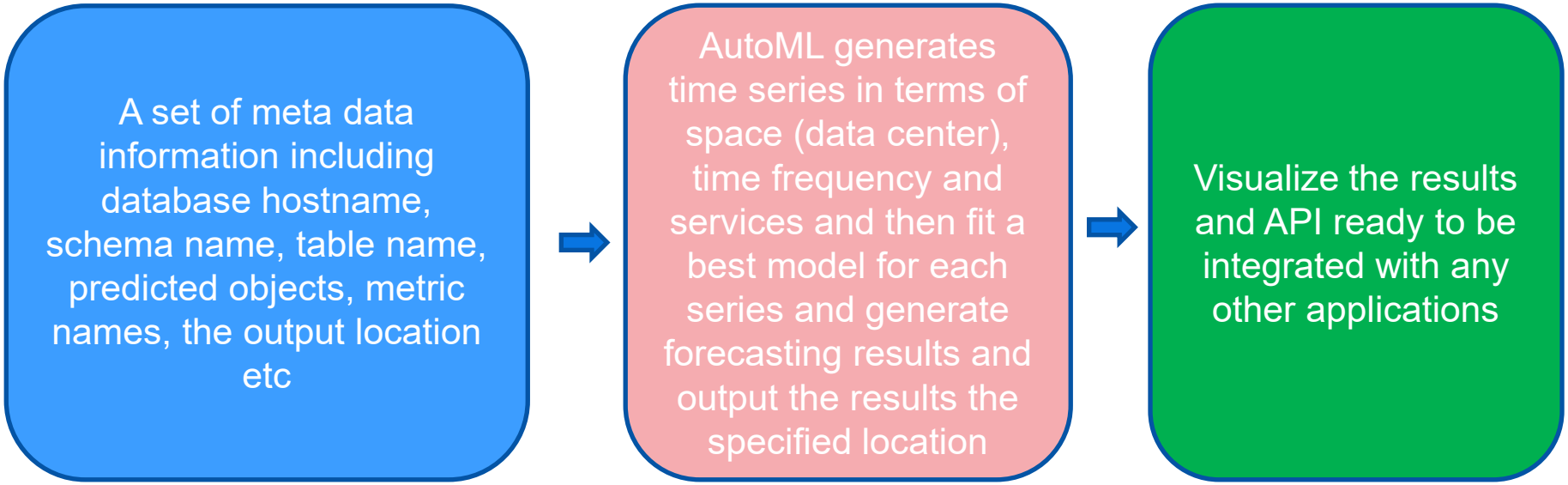
```
<recipe id="10200">
    <name>ForecastingMySQLServer</name>
    <module id="1"...>
    <module id="2"...>
    <module id="3"...>
    <module id="4"...>
    <module id="5"...>
    <module id="6"...>
</recipe>
```

```
        <value>weekly</value>
<!--        <value>daily</value>-->
    </parameter>
    <parameter id="10">
        <name>Forecasting_Step</name>
        <value>12</value>
    </parameter>
    <parameter id="11">
        <name>Forecasting_Difference_Value</name>
        <value>forecasting.difference.value.mysqlserver-useddisksize,forecasting.difference.value.mysqlserver-datadisksize,forecasting.difference.value.mysqlserver-logdisksize</value>
    </parameter>
    <parameter id="12">
        <name>Forecasting_Difference_Number</name>
        <value>forecasting.difference.number.mysqlserver-useddisksize,forecasting.difference.number.mysqlserver-datadisksize,forecasting.difference.number.mysqlserver-logdisksize</value>
    </parameter>
    <parameter id="13">
        <name>Model_Parameter</name>
        <value>arima-3-1-0</value>
    </parameter>
    <parameter id="14">
        <name>Model_Run_Time</name>
        <value>5</value>
    </parameter>
    <parameter id="15">
        <name>Forecasting_Input_Adjustment_Conditions</name>
        <value>mysqlserver-useddisksize:min,mysqlserver-useddisksize:positive|max|outlier,mysqlserver-datadisksize:min|positive|max|outlier,mysqlserver-logdisksize:min|positive|max|outlier</value>
    </parameter>
    <parameter id="16">
        <name>Forecasting_Output_Adjustment_Conditions</name>
        <value>mysqlserver-useddisksize:min,mysqlserver-useddisksize:positive|max|outlier,mysqlserver-datadisksize:min|positive|max|outlier,mysqlserver-logdisksize:min|positive|max|outlier</value>
    </parameter>
    <parameter id="17">
        <name>Input_Outlier_SD_Threshold</name>
        <value>input.value.outlier.threshold.mysqlserver-useddisksize,input.value.outlier.threshold.mysqlserver-datadisksize,input.value.outlier.threshold.mysqlserver-logdisksize</value>
    </parameter>
    <parameter id="18">
        <name>Output_Outlier_SD_Threshold</name>
        <value>output.value.outlier.threshold.mysqlserver-useddisksize,output.value.outlier.threshold.mysqlserver-datadisksize,output.value.outlier.threshold.mysqlserver-logdisksize</value>
    </parameter>
</module>
```

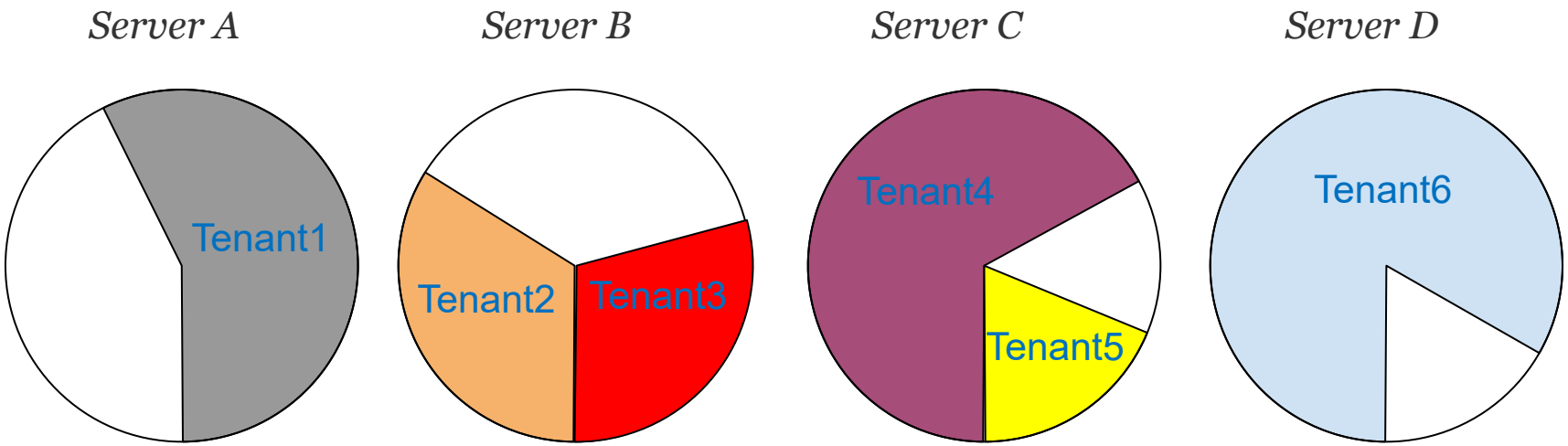
```
<module id="1">
    <name>LoadMysqlServerUsageDataFromMySQLDBTask</name>
    <class>LoadMysqlServerUsageDataFromMySQLDBTask</class>
    <parameter id="1">
        <name>data_source</name>
        <value>mysqlserver-datadisksize:useddisksize:logdisksize</value>
    </parameter>
    <parameter id="2">
        <name>Data_Base_Time_Frequency</name>
        <value>daily</value>
    </parameter>
    <parameter id="3">
        <name>Forecasting_Frequency</name>
        <value>daily</value>
    </parameter>
    <parameter id="4">
        <name>Starting_Time</name>
        <value>2021-04-01 01:15:21</value>
    </parameter>
    <parameter id="5">
        <name>Ending_Time</name>
        <value>2022-05-31 15:10:00</value>
<!--         <value>now</value>-->
    </parameter>
    <parameter id="6">
        <name>Forecasting_Range</name>
        <value>data_center_range:WD1|WD2|WD3|WD5</value>
    </parameter>
    <parameter id="7">
        <name>Data_Source_Query_Parameter</name>
        <value>database_name:tam_forecasting</value>
    </parameter>
    <parameter id="8">
        <name>Forecasting_Return_Rate</name>
        <value>false</value>
    </parameter>
    <parameter id="9">
        <name>Forecasting_Seasonality</name>
        <value>weekly</value>
```

workday.

Forecasting as a Service (SEER)



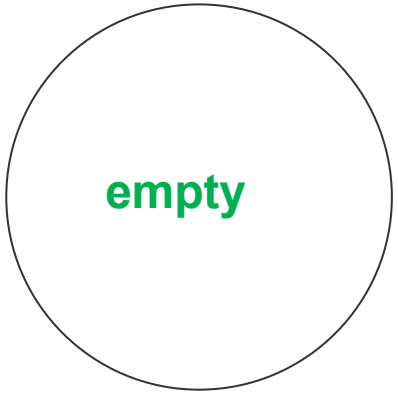
Application 1: Short-term Resource Provisioning



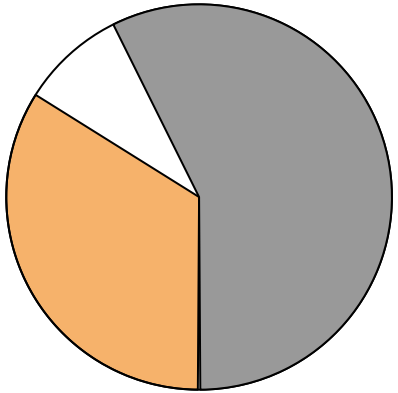
Problem: *can we host these six tenants by using fewer servers in the future 6 weeks?*

Application 1: Short-term Resource Provisioning

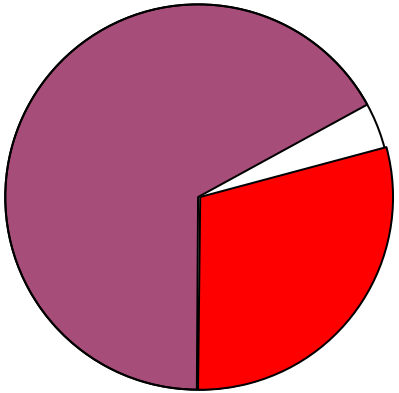
Server A



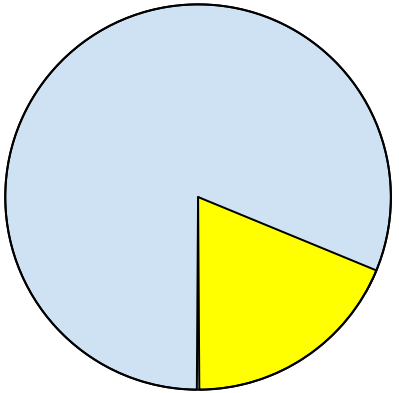
Server B



Server C



Server D



Use 0-1 integer programming to optimize the assignments and save one server.

Application 1: Short-term Resource Provisioning

$Minimize: The\ total\ resource\ usage\ by\ assigning\ tenants\ to\ servers$

Subject:

- *capacity constraints*
  e.g.: $\sum_t STATIC\_MEM_{tp} * x_{ts} \leq \delta^{RAM} * SVCRAM_s * y_s \quad \forall s \in S, \forall p \in P$

- *policy constraints*
  e.g.: $\sum_t x_{ts} \leq MaxTenants_s * y_s \quad \forall s \in S$

- *structural constraints*
  e.g.: $\sum_s x_{ts} = 1 \quad \forall t \in T$

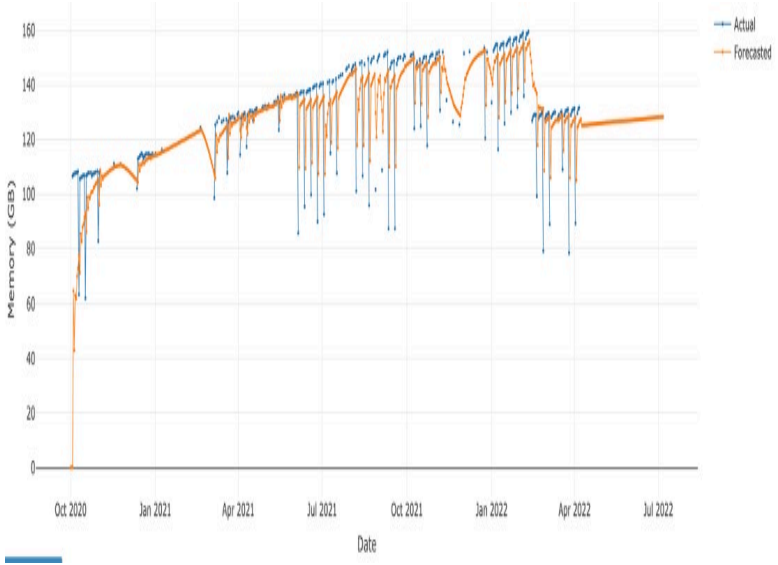
Application 1: Short-term Resource Provisioning

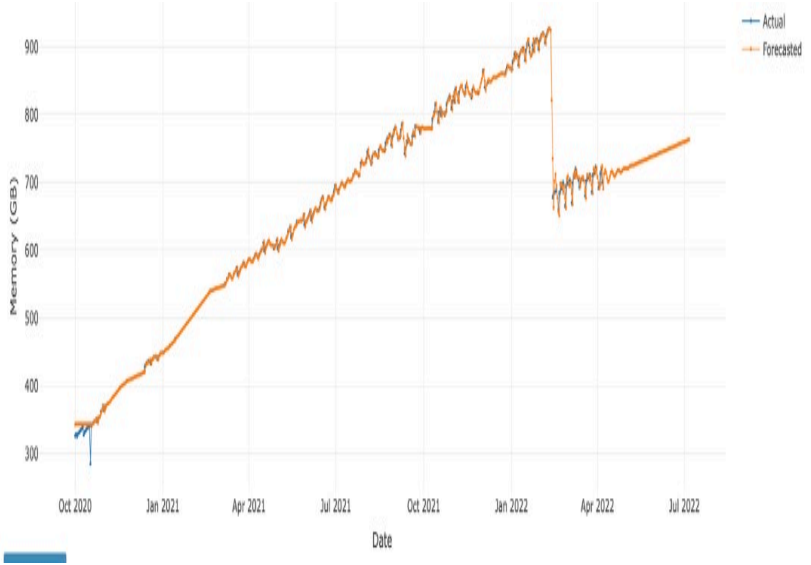
- **Reduce-hardware strategy**
  - Pack tenants from low-loaded servers and generate empties
- **Fix-infeasibility/violation strategy**
  - Heal environment by fixing infeasibilities (e.g. reduce overloaded servers)
- **Delay-patch strategy**
  - Support skip patching N cycles by taking the memory growth rates of tenants into consideration while making tenant assignment decisions
- **Maximized life span of resource usage strategy**
  - Maximize the life span of given resources by considering the growth of resource usage while making tenant assignment decisions

Application 1: Short-term Resource Provisioning

- Hundreds of thousands of forecasting models are automatically built in a Spark cluster every day and used to forecast the usage of tenant resources in the future 2-6 weeks.
- The 0-1 integer programming model is then used to allocate the resources to individual tenants to satisfy their daily operations. Three types of optimization models are built on top of forecasting output: daily optimization, weekly optimization, and delayed patch optimization (usually 2-6 weeks).
- The weekly operations optimization can be a reduce-hardware strategy and/or fix-violation strategy for one environment (such as IMPL, PROD, SANDBOX, PREVIEW) within one data center.
- Assume that there are 5 data centers, 4 environments per data centers, 5000 tenants per data and environments, and 3 metrics. In total, there are 5 x 4 x 5000 x 3 = 300,000 time series models.

Application 1: Short-term Resource Provisioning





Application 2: Long-term Resource Demand Planning

- In the long-term resource demand planning, thousands of monthly time series models can be automatically built every month for the different types of servers and/or the aggregated usage of server resources at the data center level and used to forecast the resource demand in the future 3-5 years.
- This type of forecasting is very helpful for determining the monthly and quarterly procurement and the quarterly and annual budgets in demand planning.

Conclusions

- This talk presents a framework for AutoML time series forecasting powered by Spark, focusing on time series metrics of infrastructure resources. The framework supports ARIMA (autoregressive integrated moving average) and various other time series models. A key innovation is the use of Spark-based AutoML and big data distributed computation to automate and scale the fitting and forecasting operations of millions of time series models in parallell.
- Two empirical applications are presented. In these applications, hundreds of thousands of time series models are automatically built and used to support short-term operations and plan long-term resource demand. The applications demonstrate that the AutoML time series forecasting framework is a high-level, innovative solution that is efficient, easy to implement, and capable of addressing multiple scenarios of time series forecasting for infrastructure resources.
- The framework can be applied to the time series forecasting in the different application areas.

Q & A

Thank you!