GridGain

Continuous Machine and Deep Learning at Scale With Apache Ignite

Ken Cottrell
Solution Architect
ken.Cottrell@gridgain.com

Agenda

- Continuous Machine Learning / Deep Learning Introduction
- Overview of Apache Ignite Continuous ML/DL Capabilities
- Demo & API discussion
- Q & A

Why Machine Learning at Scale?

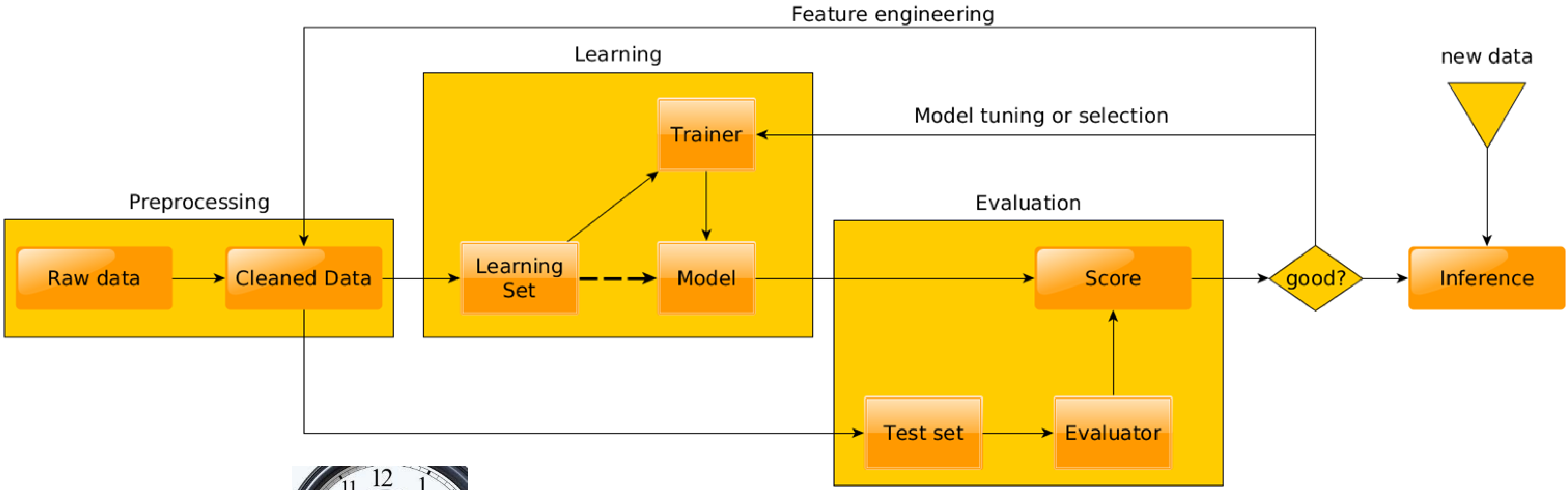
Scalability

- Data exceed capacity of single server
- Burden for development and business

Models trained & then deployed in different systems

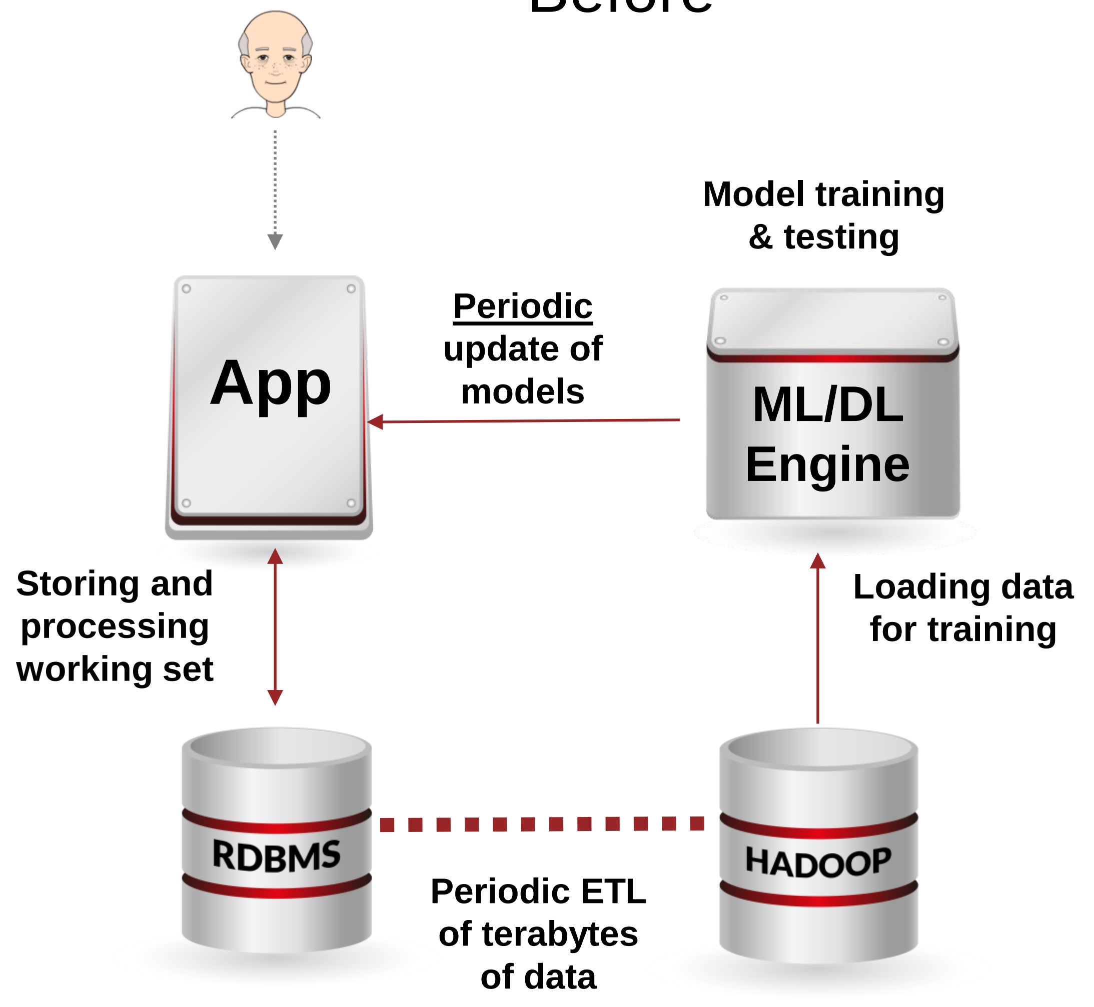
- Move data out for training
- Wait for training to complete
- Redeploy models in production

Machine Learning Pipelines: where is the time spent?

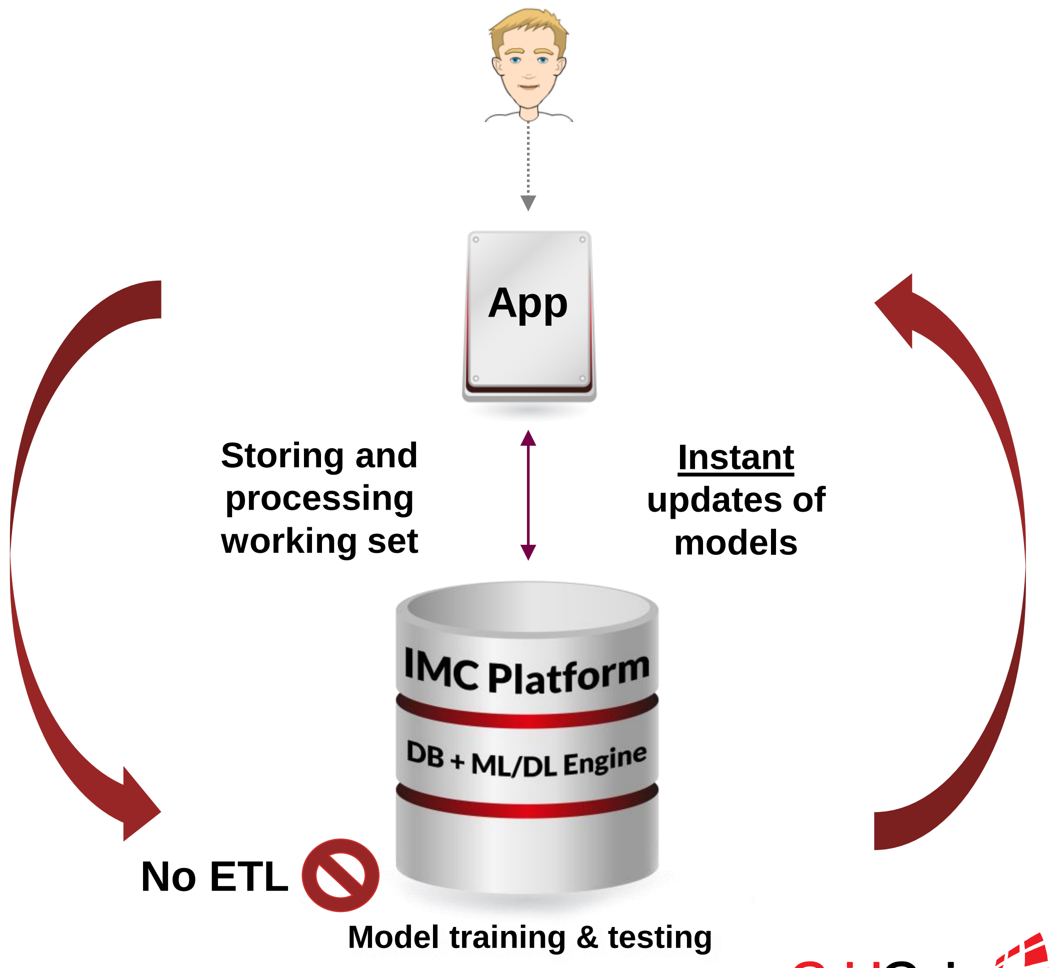


Continuous Machine Learning at Scale

Before



After (With CL)



Apache Ignite Is a Top 5 Apache Project

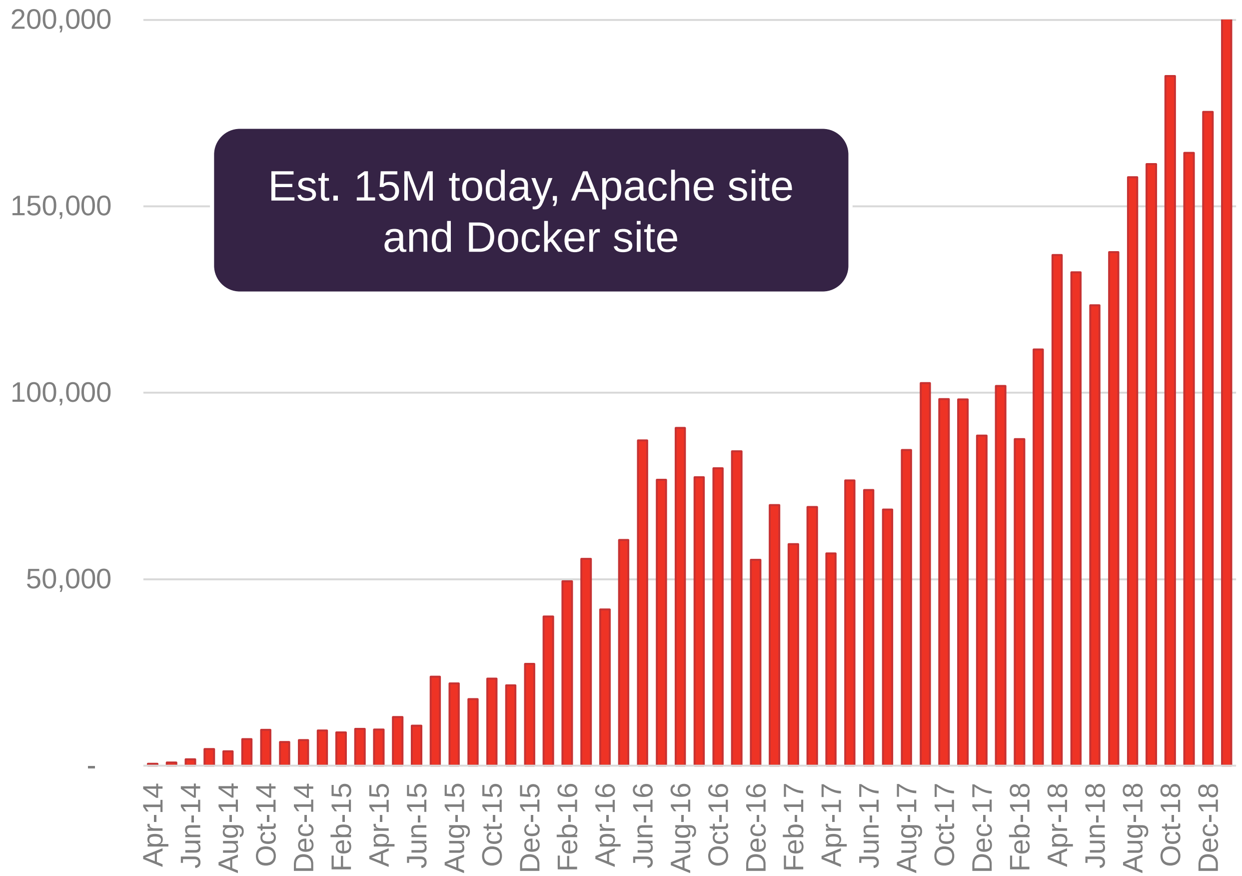
A Top 5 Apache Software Foundation Project

Top 5 Dev Mailing Lists	
1.	beam
2.	apache Ignite
3.	kafka
4.	Apache Tomcat
5.	James Enterprise Mail Server

Top 5 User Mailing Lists	
1.	Flink
2.	Lucene
3.	apache Ignite
4.	cassandra
5.	kafka

From January 1, 2019 Apache Software Foundation Blog Post:
"Apache in 2018 – By The Digits"

Monthly Ignite/GridGain Downloads



GridGain

Apache Ignite Users

Financial Services

- American Express
- citibank
- Wellington Management
- ING
- Macquarie
- Sberbank
- Raymond James
- 中国平安 PING AN
- Barclays
- Jefferies
- Societe Generale

FinTech

- The Glue FinTech Solutions
- markit
- Finastra
- SS&C Advent
- Kinetix Trading Solutions
- mosaic smart data

Software/Cloud

- servicenow
- Brocade
- Microsoft
- vmware
- Marketo An Adobe Company

Telecom & Mobile

- Kodiak a Motorola Solutions Company
- Huawei
- RingCentral
- Reliance Jio
- intelligentpipe

IoT

- thingworx
- NTT Data
- Itron

AdTech / Media / Entertainment

- adalyser
- DreamWorks
- Sizmek

Logistics & Transportation

- American Airlines
- MercuryGate TMS that delivers.
- DriveNow
- Nederlandse Spoorwegen

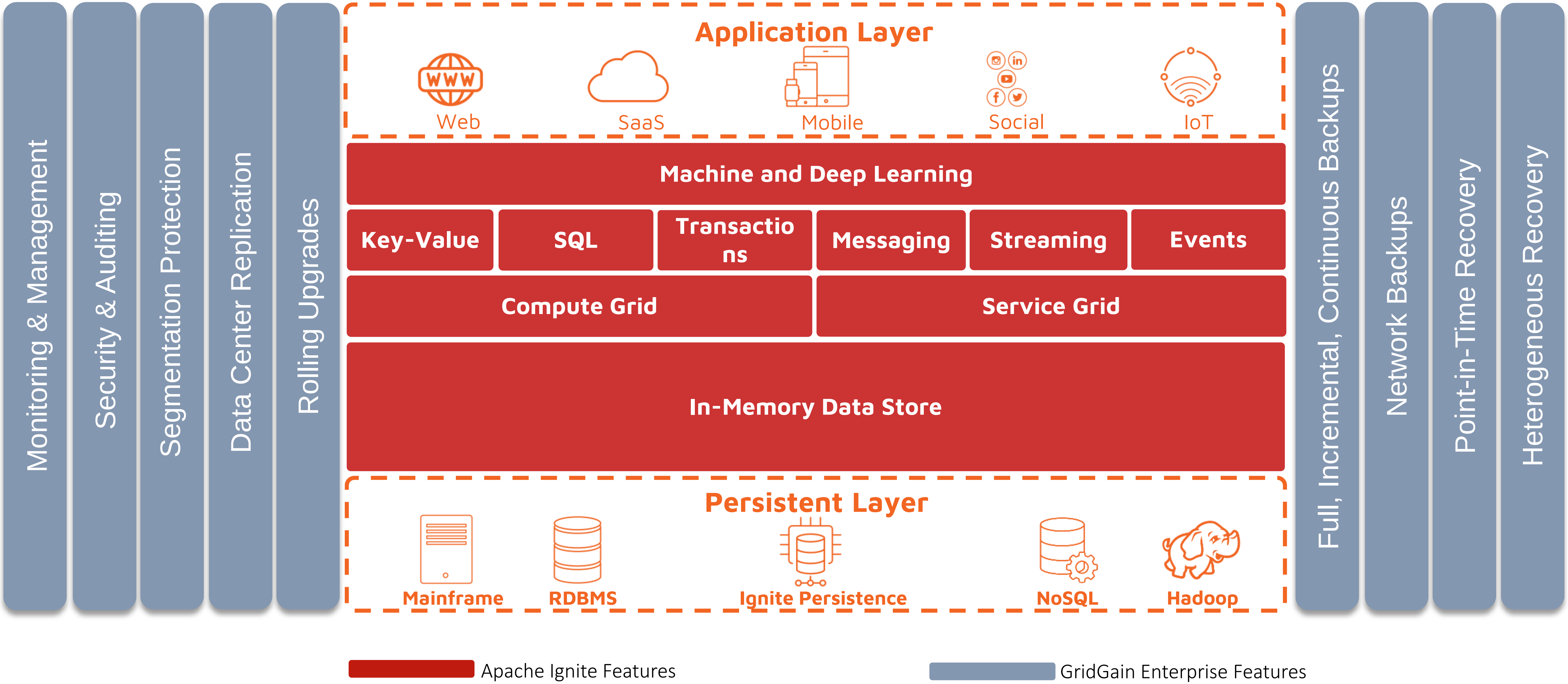
eCommerce & Retail

- newegg
- 24 Hour Fitness
- TUI Travel PLC
- СТРОЙДВОР строительные материалы
- JacTravel

Pharma & Healthcare

- Prime Therapeutics
- Agilent
- UnitedHealthcare
- MultiPlan
- e-Therapeutics plc

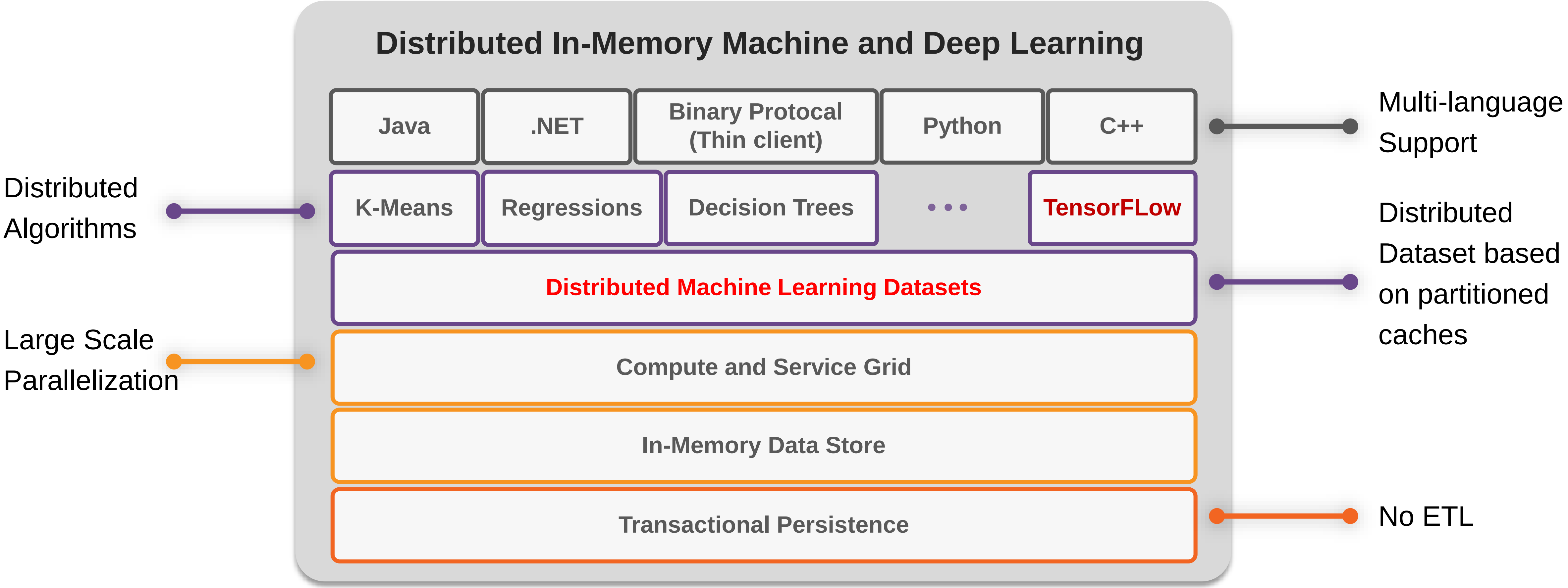
Apache Ignite In-Memory Computing Platform



GridGain

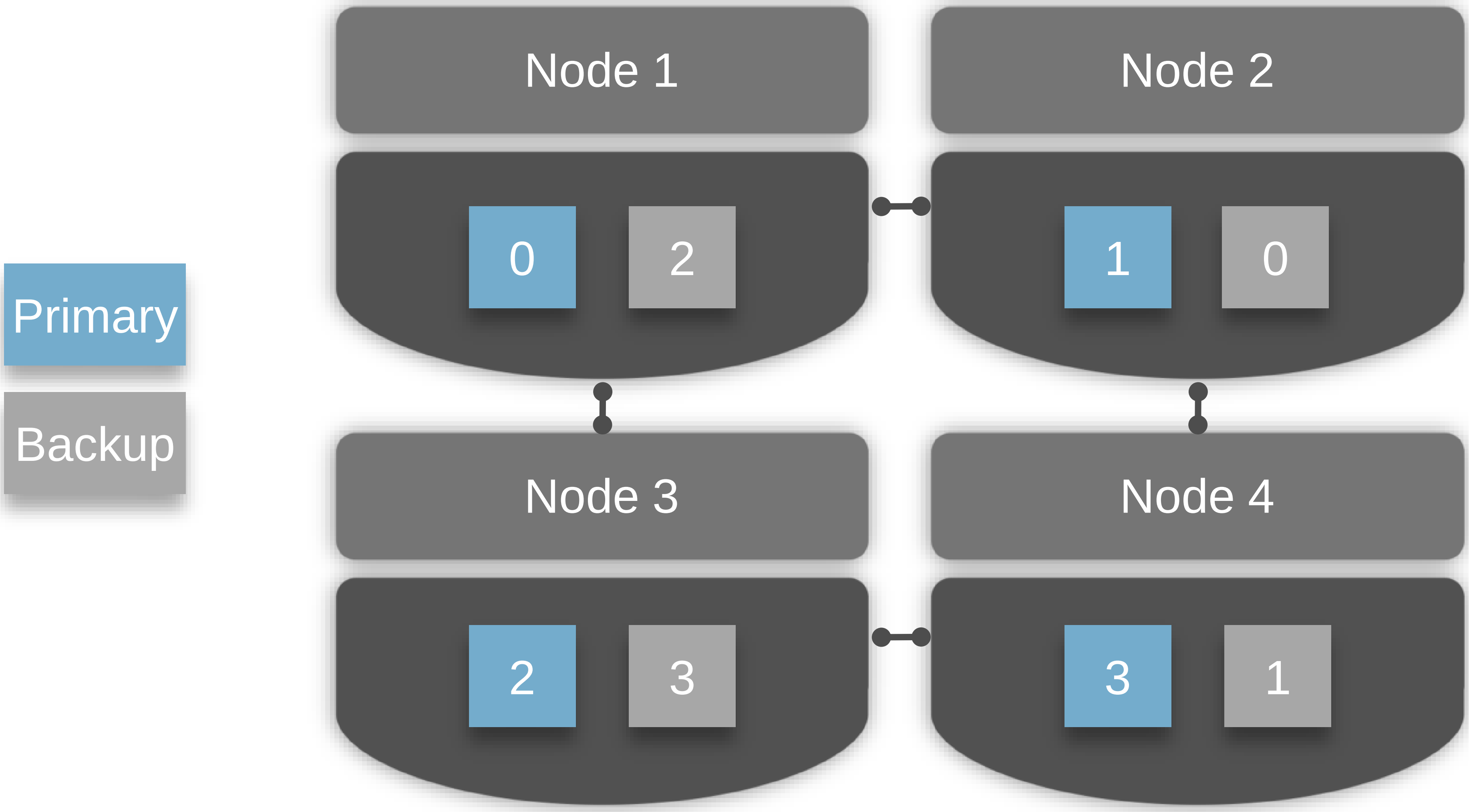
Overview of Apache Ignite Continuous ML/DL Capabilities

Apache Ignite Continuous Learning framework



GridGain

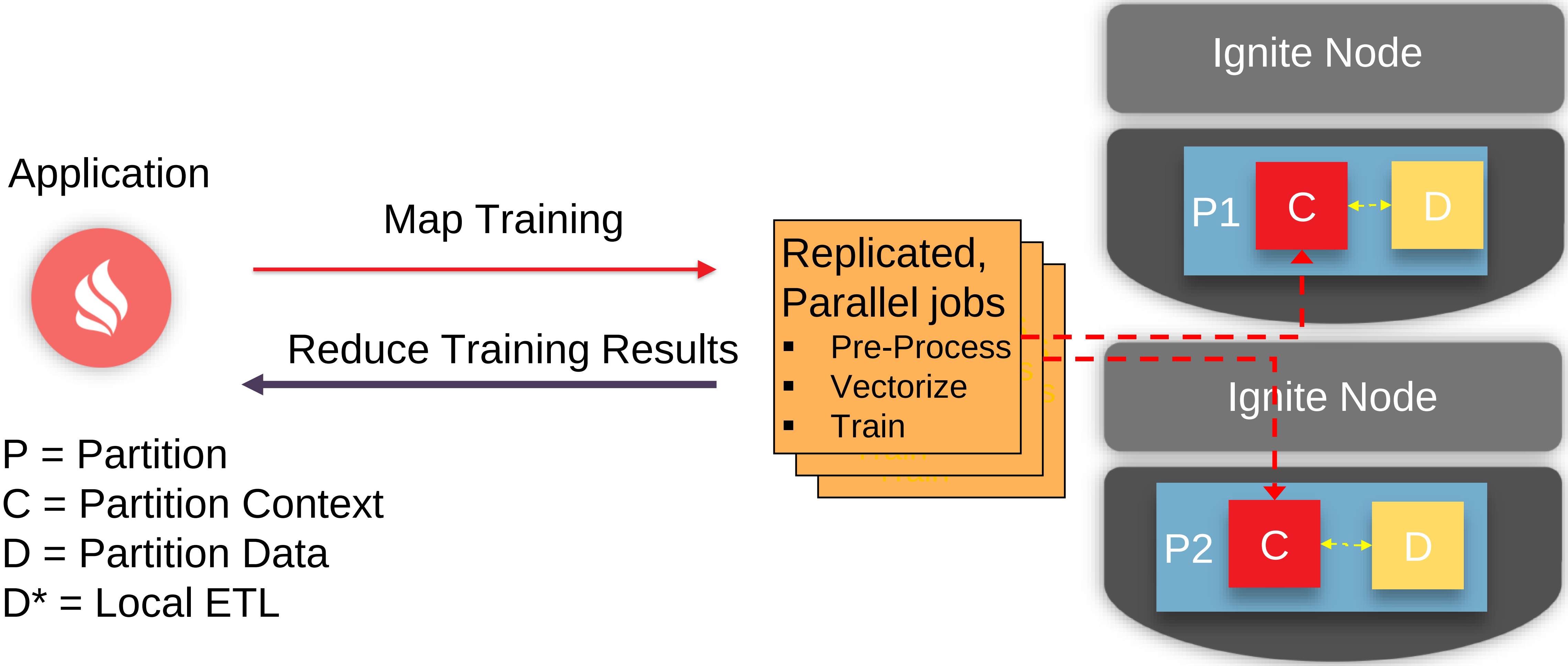
Partitions Distribution and Replication



Co-Located by Partition:

- Transactional Data
- Vectorized Data
- Training context data
- Other Computation functions

Continuous Learning enabled with Partitioned Datasets



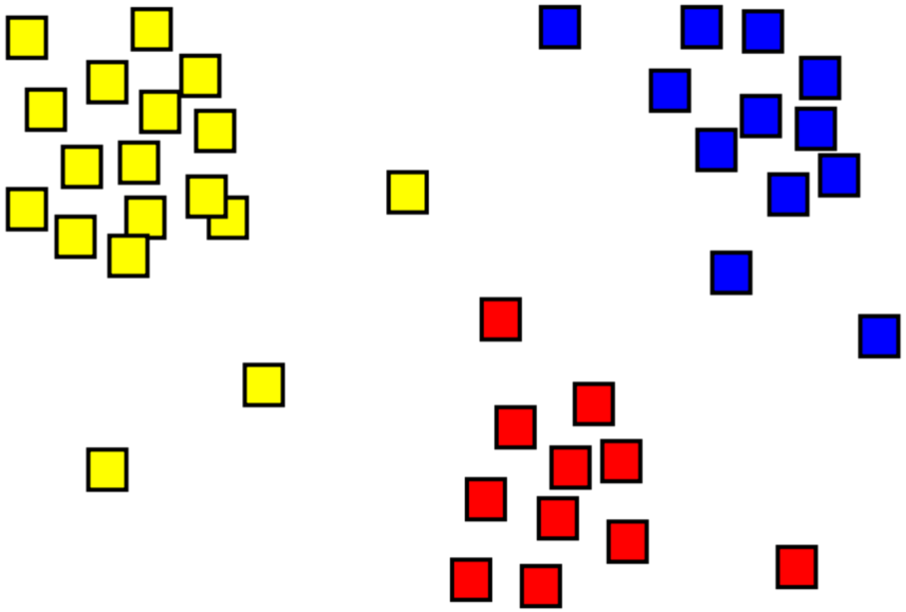
P = Partition
C = Partition Context
D = Partition Data
D* = Local ETL

GridGain

Apache Ignite Distributed Training: Clustering

- K-means (Centroid Mean)
- GMM (Centroid Mean + Variance)

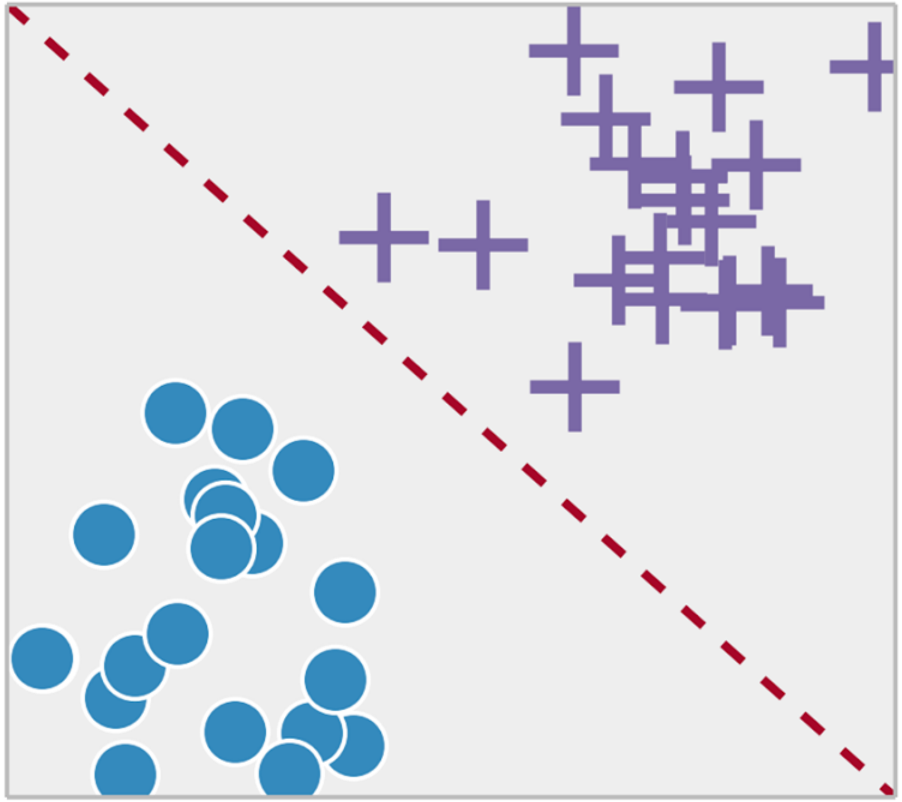
- Use Cases - OLTP and other tabular data that need to be Labeled
  - Customer Segmentation
  - Anomaly Detection
  - Network throughput characterization



Apache Ignite Distributed Training: Classification

- Logistic Regression & Naive Bayes
- SVM, KNN, ANN
- Decision trees & Random Forest

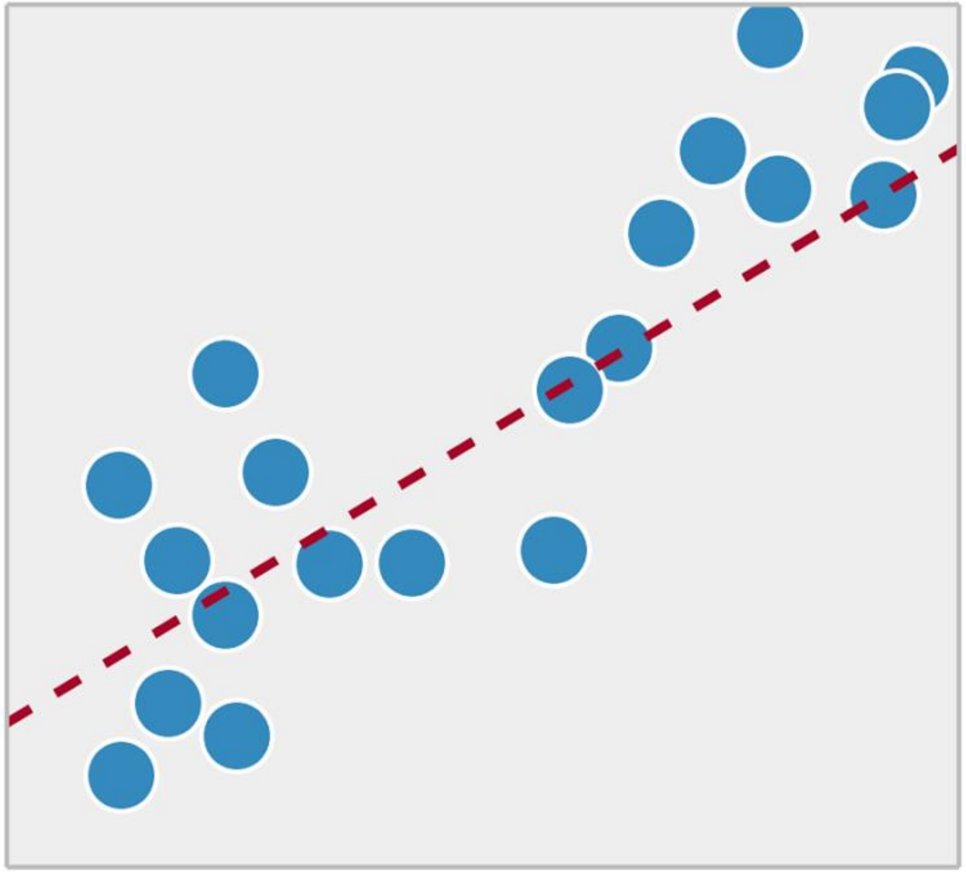
- Use cases - Operational (OLTP) data prediction:
  - Fraud detection
  - Credit Card Scoring
  - Clinical Trials
  - Customer Segmentation



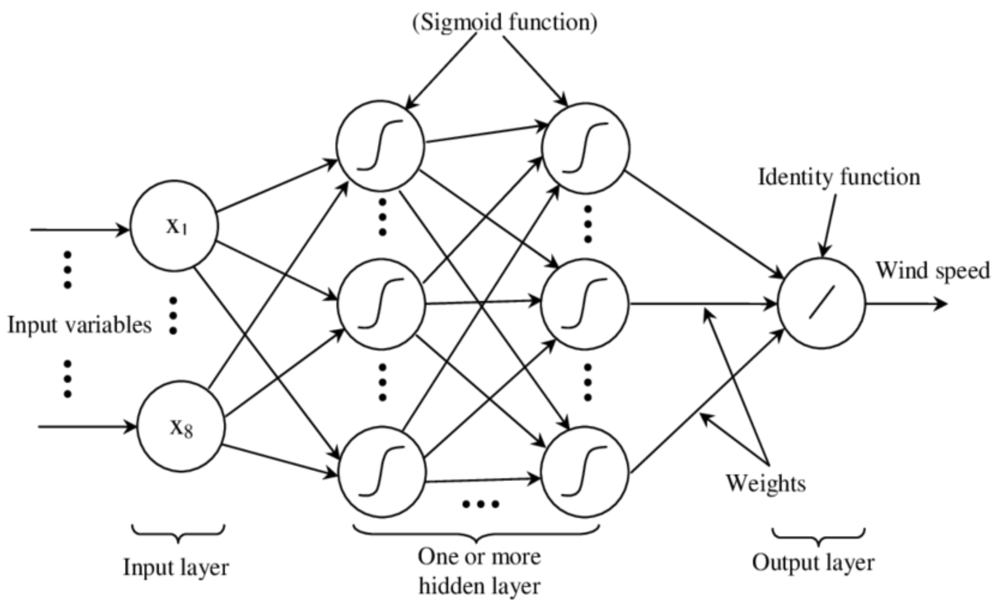
Apache Ignite Distributed Training: Regression

- KNN & Linear Regressions
- Decision tree regression
- Random forest regression
- Gradient-boosted tree regression

- Use cases - Operational data (OLTP) predictions
  - Trend analysis
  - Financial forecasting
  - Time series prediction
  - Response modeling (pharma etc)



Apache Ignite: TensorFlow Integration



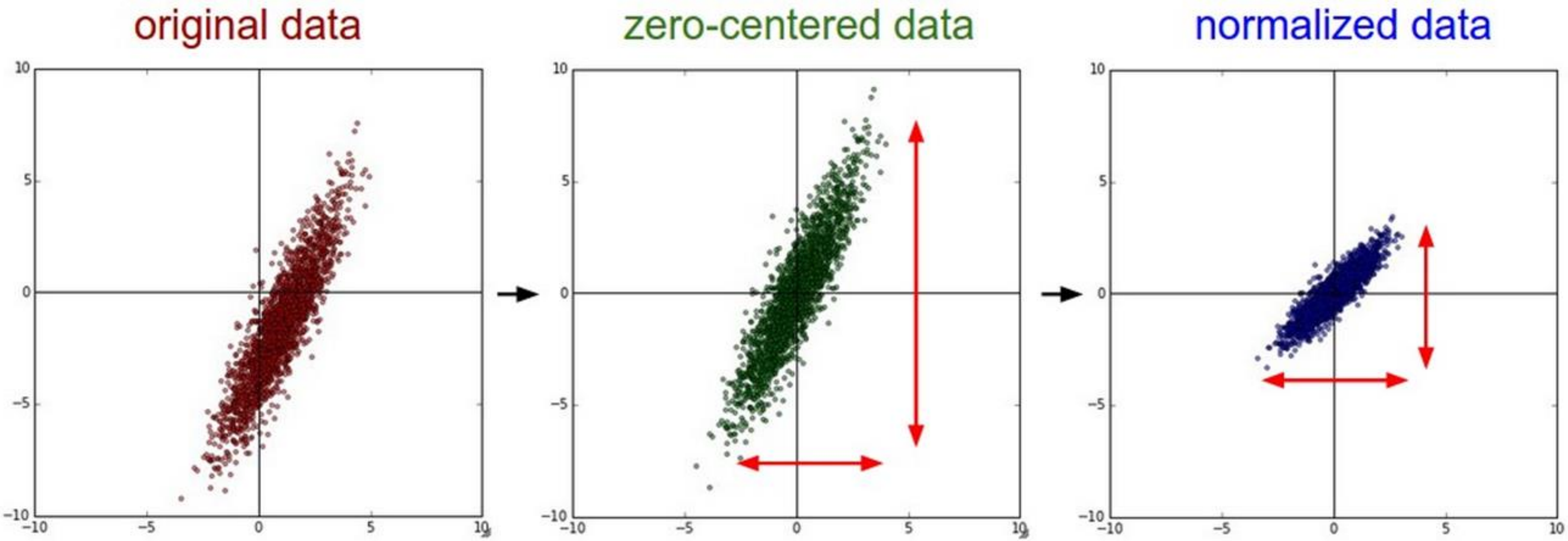

```
>>> import tensorflow as tf
>>> from tensorflow.contrib.ignite import IgniteDataset
>>>
>>> dataset = IgniteDataset(cache_name="SQL_PUBLIC_KITTEN_CACHE")
>>> iterator = dataset.make_one_shot_iterator()
>>> next_obj = iterator.get_next()
>>>
>>> with tf.Session() as sess:
>>>   for _ in range(3):
>>>     print(sess.run(next_obj))

{'key': 1, 'val': {'NAME': b'WARM KITTY'}}
{'key': 2, 'val': {'NAME': b'SOFT KITTY'}}
{'key': 3, 'val': {'NAME': b'LITTLE BALL OF FUR'}}
```

Use Cases - Operational data "High dimension" data (Images, Text, Audio, speech)

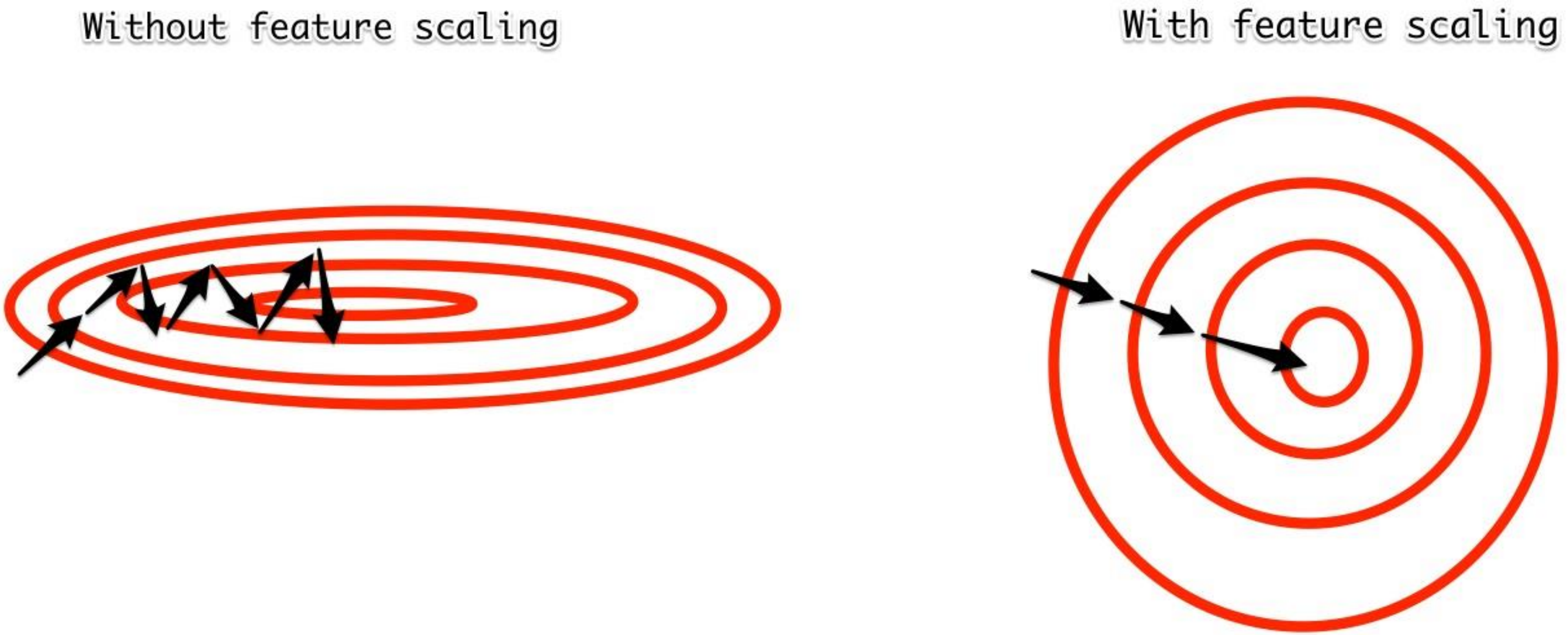
- Image data classification
- Natural Language Processing Clinical notes
- Document Classification, Free Form text

GridGain

Apache Ignite Distributed PreProcessing: Normalization

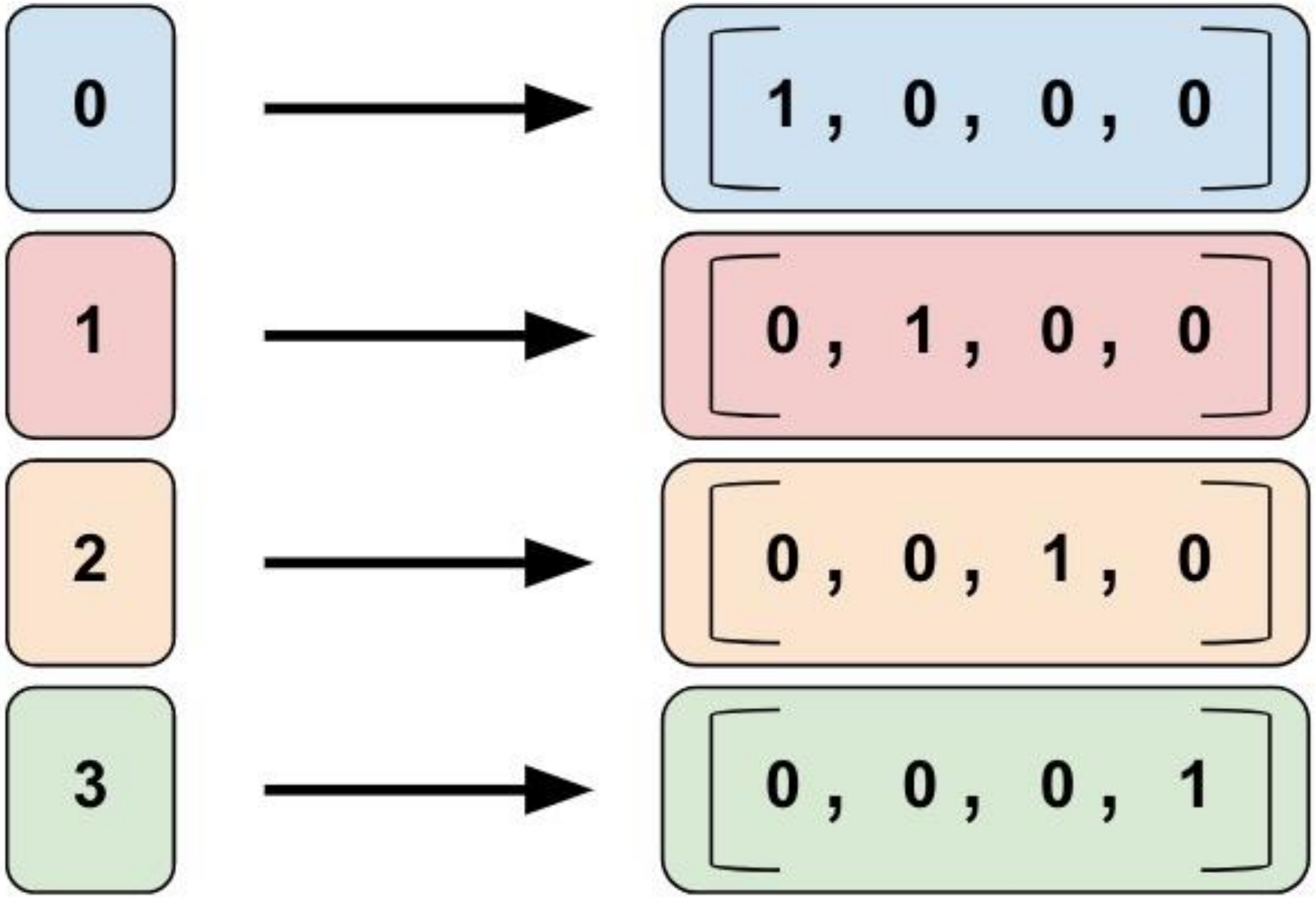


Apache Ignite Distributed Preprocessing: Scaling



https://medium.com/@nsethi610/data-cleaning-scale-and-normalize-data-4a7c781dd628

Apache Ignite Distributed preprocessing: One-Hot Encoder



* Also included: String Encoding

Achieving Continuous ML/DL at Scale:
Architectural Considerations / Trade-Offs

- Operational Data Models: from centralized to parallelized
  - De-normalization Data Affinity for parallel Loads, Queries, Updates, Joins
  - Horizontal scale-out

- Done locally in node: data partition + preprocessing + training + inferencing
  - Reduces data shuffling over the network between the cluster and application

- ML pipeline enhancements
  - Co-Located & Distributed processing of all ML steps: ingest to inferencing
  - ML model performance measured, and updatable, with nearby transaction data

Demo & API discussion

package org.apache.ignite.examples.ml

Adding your own Preprocessor and Algorithm to a Dataset

- **dataset**/AlgorithmSpecificDatasetExample.java

Passing custom preprocessor classes to the cluster

- **environment**/TrainingWithCustomPreprocessorsExample.java

TensorFlow data set , inferencing at the cluster nodes

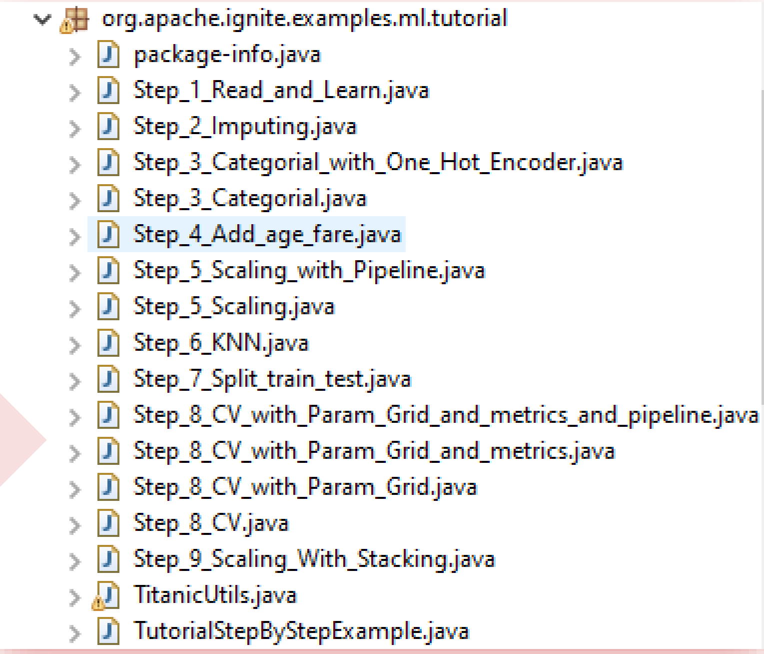
- **inference/**TensorFlowDistributedInferenceExample.java

Decision tree

- **tree/**FraudDetectionExample.java

End-to-End Model Prep & Training Tutorial (shows feature preprocessing, transformation, different algorithm comparisons, accuracy metrics, pipelines)

- **tutorial**/*.java   // pipeline to preprocess, train,
                  // & evaluate Titanic passenger data



Ignite ML API to Update the model

```
SVMLinearClassificationTrainer trainer = new
SVMLinearClassificationTrainer();

SVMLinearClassificationModel mdl1 =
trainer.fit(ignite, dataCache1, vectorizer);

SVMLinearClassificationModel mdl2 =
trainer.update(mdl1, ignite, dataCache2,
vectorizer);
```

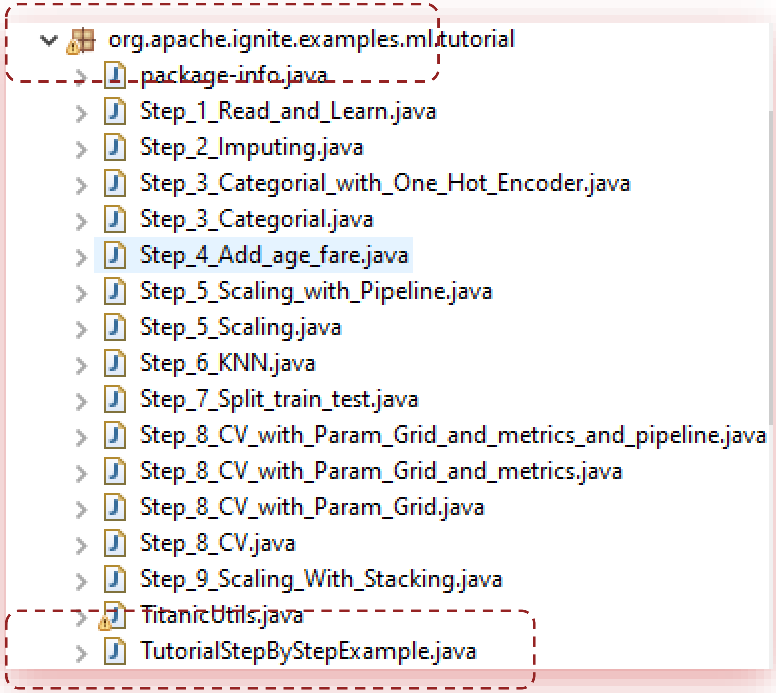
`DatasetTraininer` interface:

(Some Constraints according to the Algorithm)

Online / Online Batch with new data

- Centroid updates – KMeans, ANN
- Add new dataset - KNN
- Update with new Gradient – NN, Log Regression, Linear Regression
- Increment to Current state - SVM, GDB
- Decision Tree – retrain
- Random Forest – adds new DT, may discard other DTs for size management

Demo tutorial sample



To run this example:

- Import this directory with pom.xml into your favorite IDE as a Maven project
  - `<path>\apache-ignite-2.7.6-bin\`**`examples\pom.xml`**
- I'll run this job on a single node inside my laptop on Eclipse (normally you would run jobs on a cluster of nodes)
  - Each of these Steps can be run independently or all together with **`TutorialStepByStepExample.java`**
  - Widely used Titanic data set (we include it here)
- Discussion of how Apache Ignite API can be invoked by 3rd party Auto ML and other application wrappers
- Compare the Accuracy obtained different ML steps
  - Accuracy defined as % correct predictions versus ground truth
  - Different algorithms and different preprocessing
  - Effects of Test / Train split on Overfitting

GridGain

Apache Ignite Spark integration

```
Dataset<Row> peopleDF = spark.read().json(
        resolveIgnitePath("resources/people.json").getAbsolutePath());

System.out.println("JSON file contents:");

peopleDF.show();

System.out.println("Writing DataFrame to Ignite.");

peopleDF.write()
        .format(IgniteDataFrameSettings.FORMAT_IGNITE())
        .option(IgniteDataFrameSettings.OPTION_CONFIG_FILE(), CONFIG)
        .option(IgniteDataFrameSettings.OPTION_TABLE(), "people")
        .option(IgniteDataFrameSettings.OPTION_CREATE_TABLE_PRIMARY_KEY_FIELDS(),
                "id")
        .option(IgniteDataFrameSettings.OPTION_CREATE_TABLE_PARAMETERS(),
                "template=replicated")
        .save();
```

```
System.out.println("Reading data from Ignite table.");

Dataset<Row> peopleDF = spark.read()
        .format(IgniteDataFrameSettings.FORMAT_IGNITE())
        .option(IgniteDataFrameSettings.OPTION_CONFIG_FILE(), CONFIG)
        .option(IgniteDataFrameSettings.OPTION_TABLE(), "people")
        .load();

peopleDF.createOrReplaceTempView("people");

Dataset<Row> sqlDF = spark.sql("SELECT * FROM people WHERE id > 0 AND id < 6");
sqlDF.show();
```

Write to Ignite DataFrame from within Spark session

Read from same Ignite DataFrame from another Spark Session

- DF (and RDD) shared across sessions
- SQL with Indexing for faster queries
- Ignite DF are mutable

To Summarize: Apache Ignite for Continuous Learning at Scale

Massive Scale for Memory, Storage, Computation

- Massive Throughput with minimal ETL
- Massive operational data sizes + in-place parallel processing
- Faster cycle times from transactions, ML/DL dataset extraction, predictions

Integrates with Existing ML / DL operations

- Low-level Distributed APIs to integrate with Auto ML and other Data Science workflows
- For End-Users: Python API to manage Cache, Datasets, SQL, ML
- Apache Ignite integrations to accelerate Spark, TensorFlow pipelines; including Model imports from other tool sets

Resources

- Documentation:
  - https://apacheignite.readme.io/docs
- Python support
  - https://github.com/gridgain/ml-python-api
- Examples and Tutorials:
  - https://github.com/apache/ignite/tree/master/examples/src/main/java/org/apache/ignite/examples/ml
- Details on TensorFlow
  - https://medium.com/tensorflow/tensorflow-on-apache-ignite-99f1fc60efeb

Q & A