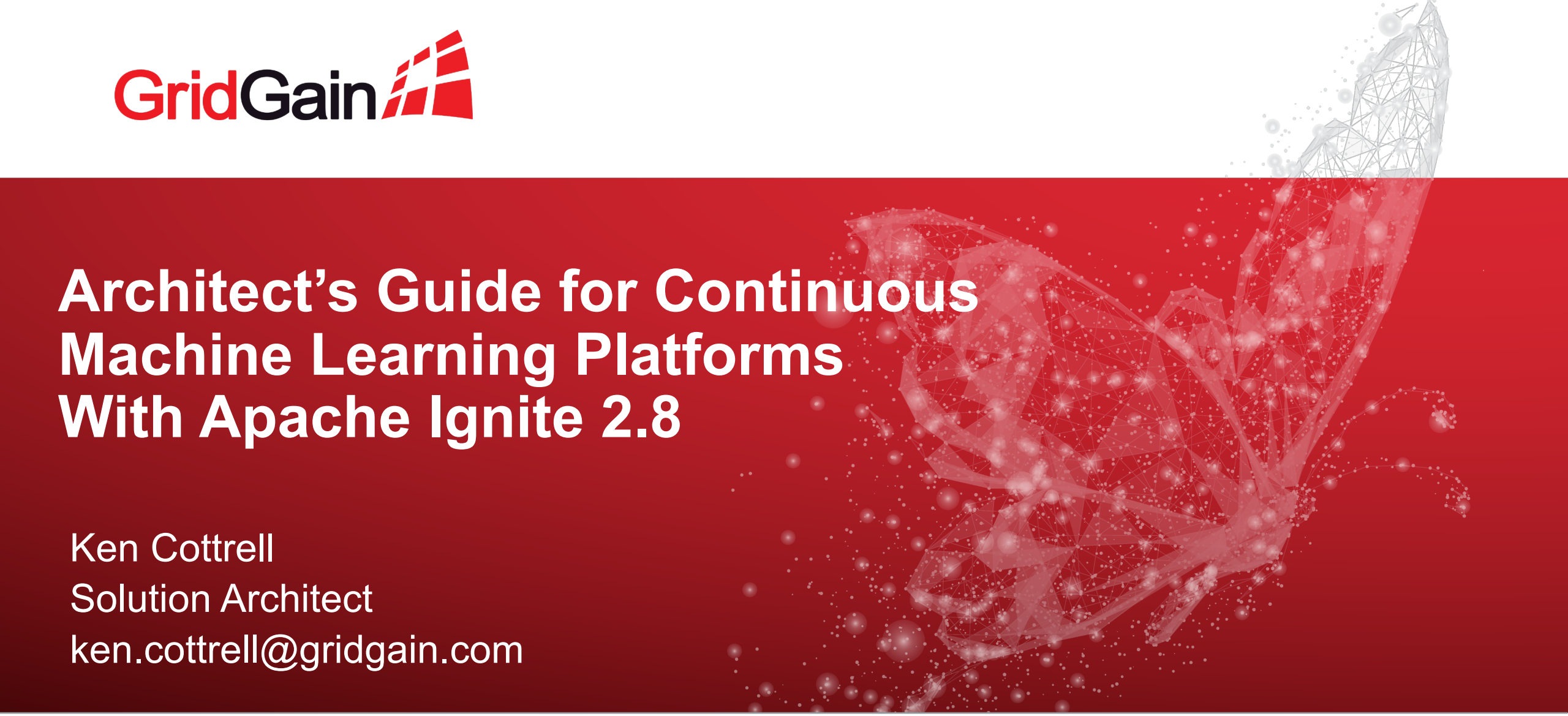


Architect's Guide for Continuous Machine Learning Platforms With Apache Ignite 2.8

Ken Cottrell
Solution Architect
ken.cottrell@gridgain.com

Agenda

Architect's Guide for Continuous Machine Learning With Apache Ignite 2.8

- Introduction to ML/DL Architecture Patterns: Continuous Machine Learning at Scale
- Apache Ignite 2.8 ML/DL Capabilities and New Features
- Apache Ignite 2.8 ML/DL Code Examples / Demos
- Q & A

2019 © GridGain Systems

Apache Ignite 2.8

Over 1,900 upgrades and fixes that enhance almost all platform components

- Production Support Improvements to better support high 24x7 workloads
  - Persistence compaction, Baseline topology autoadjustment
- Support for Next-Generation Monitoring, User-defined Alerts, and Event Tracing
  - APIs for JMX, SQL, OpenCensus
  - Take a look at the new Gridgain Control Center product: https://control.gridgain.com/
- Machine Learning Enhancements
  - A new pipelining API gives you added ML workflow flexibility
  - Additional ensemble methods, which allows you to combine ML training algorithms into more agile training workflows
  - Model parsers to leverage outputs from such leading ML technologies as Apache Spark and XGBoost models

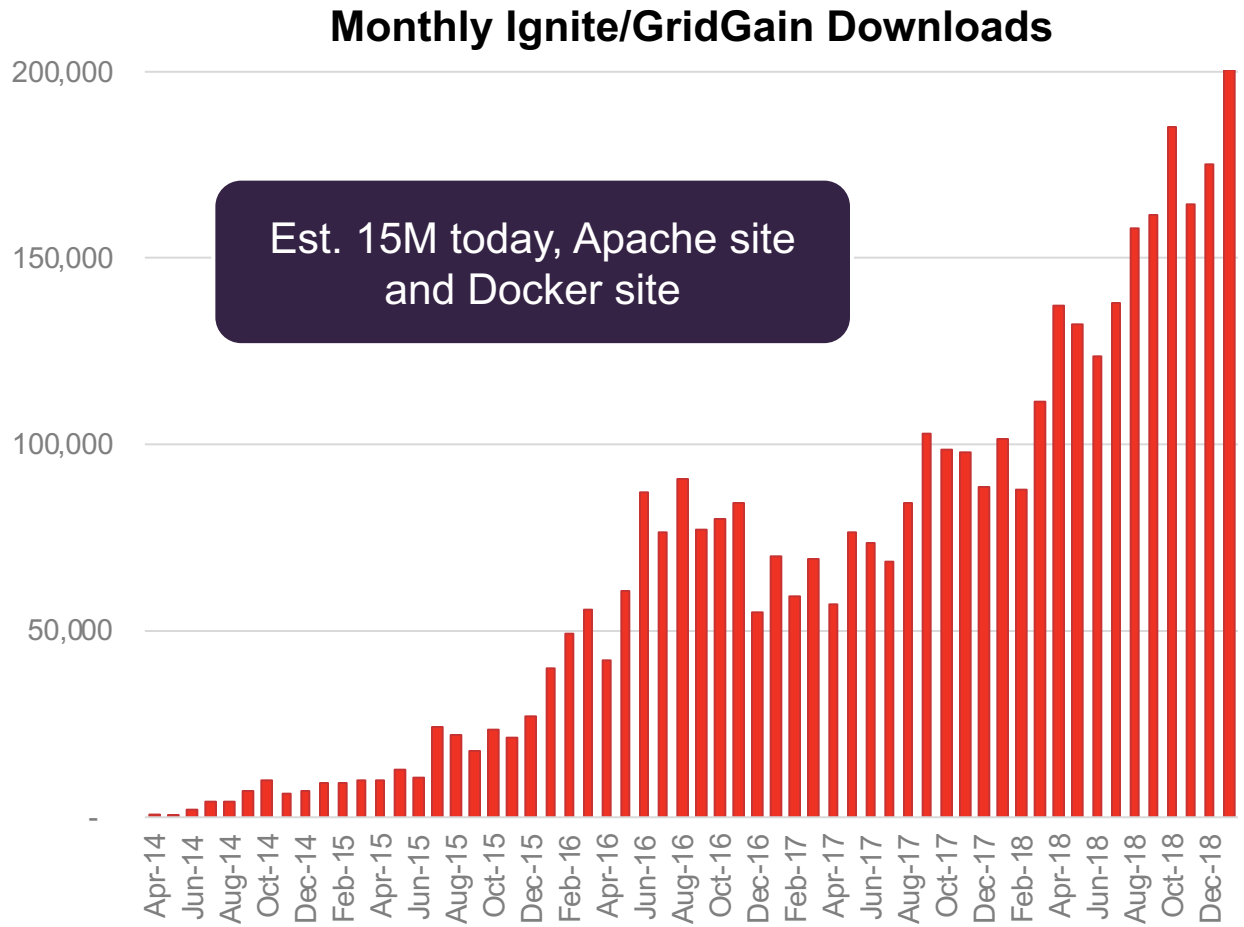
2019 © GridGain Systems

Apache Ignite Is a Top 5 Apache Project

A Top 5 Apache Software Foundation Project

Top 5 Dev Mailing Lists	Top 5 User Mailing Lists
1. beam	1. Flink
2. Apache Ignite	2. Lucene
3. kafka	3. Apache Ignite
4. Apache Tomcat	4. cassandra
5. James Enterprise Mail Server	5. kafka

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



2019 © GridGain Systems

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

Software/Cloud

ServiceNow, Brocade, Microsoft, VMware, Marketo An Adobe Company

Logistics & Transportation

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

Telecom & Mobile

KODIAK a Motorola Solutions Company, Huawei, RingCentral, Reliance Jio, intelligentpipe

eCommerce & Retail

newegg, 24 HOUR FITNESS, TUI Travel PLC, СТРОЙДВОР СТРОИТЕЛЬНЫЕ МАТЕРИАЛЫ, JacTravel

FinTech

THE GLUE FINTECH SOLUTIONS, markit, FINASTRA, SS&C ADVENT, KINETIX TRADING SOLUTIONS, mosaic SMART DATA

IoT

thingworx, NTT DATA, Itron

AdTech / Media / Entertainment

adalyser, DreamWorks, Sizmek

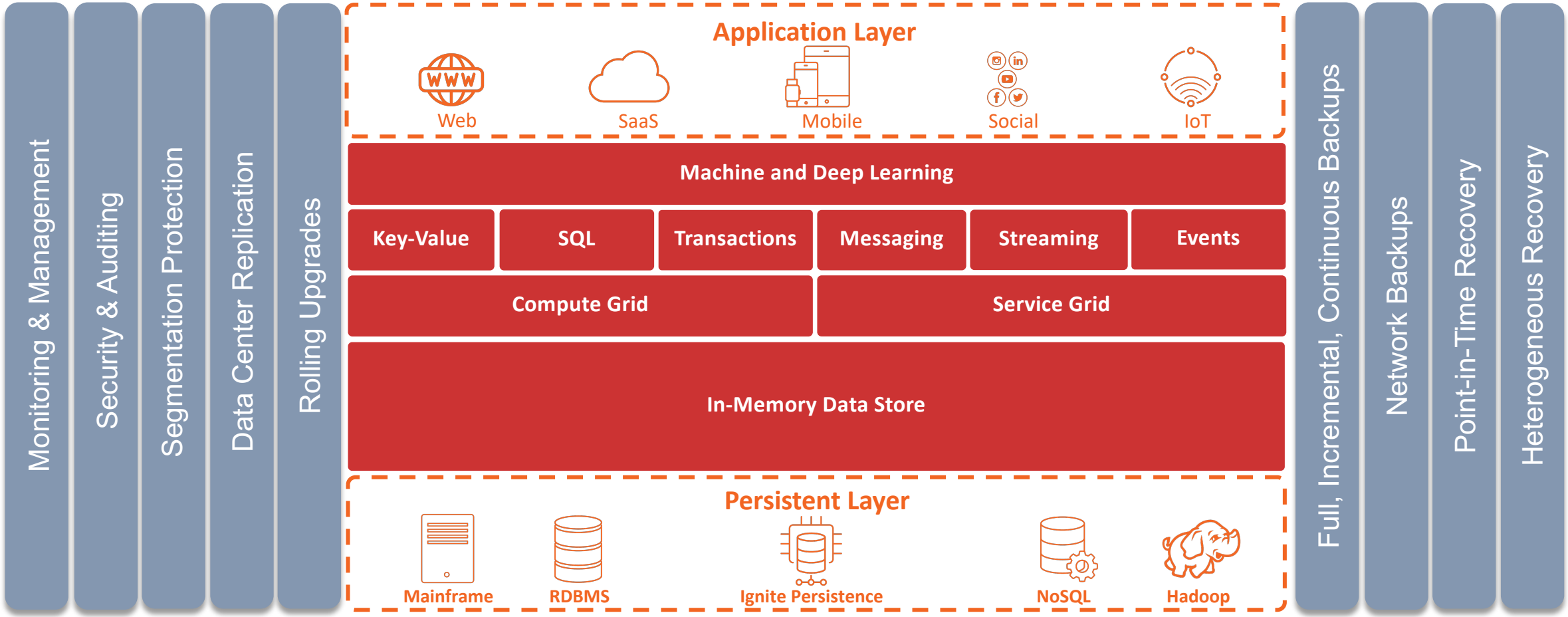
Pharma & Healthcare

PRIME THERAPEUTICS, Agilent, UnitedHealthcare, MultiPlan, e-Therapeutics plc

2019 © GridGain Systems

GridGain

Apache Ignite In-Memory Computing Platform



Apache Ignite Features

GridGain Enterprise Features

2019 © GridGain Systems

GridGain

ML/DL Architectures that deliver Continuous Machine Learning at Scale

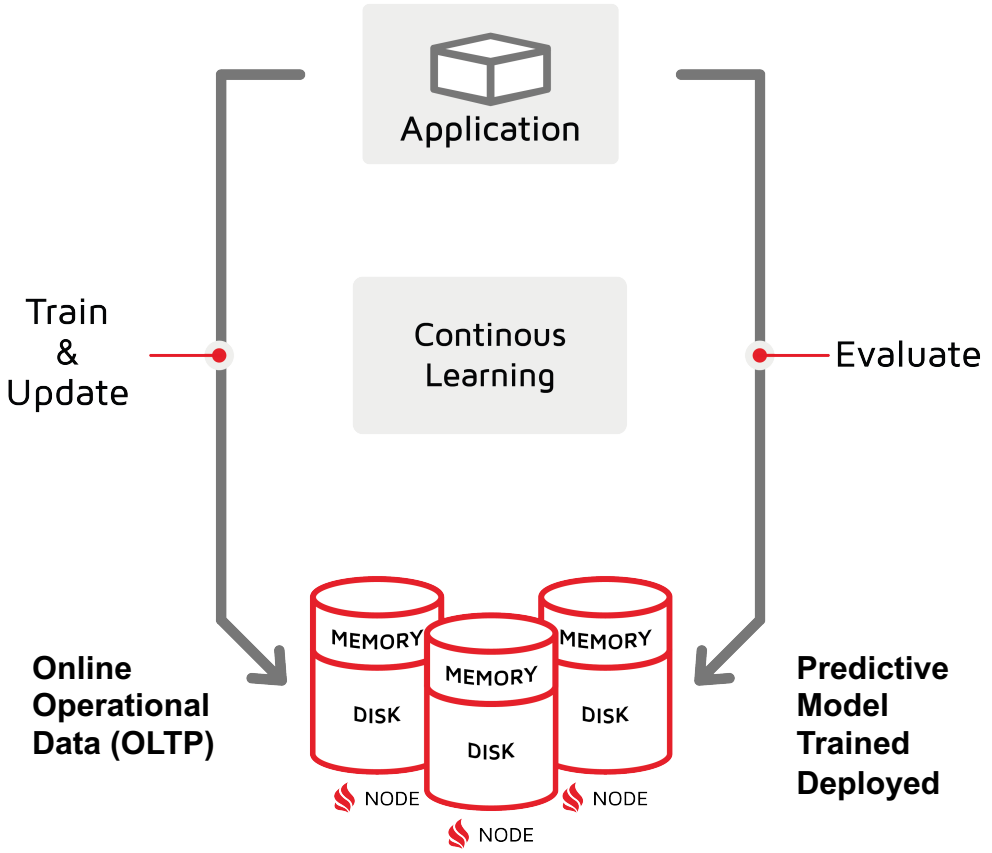
2019 © GridGain Systems

GridGain

Continuous Machine Learning At Scale:
Both Real-time and Batch Data, Together

- New “model update” APIs let refine an existing model on the fly with new data samples

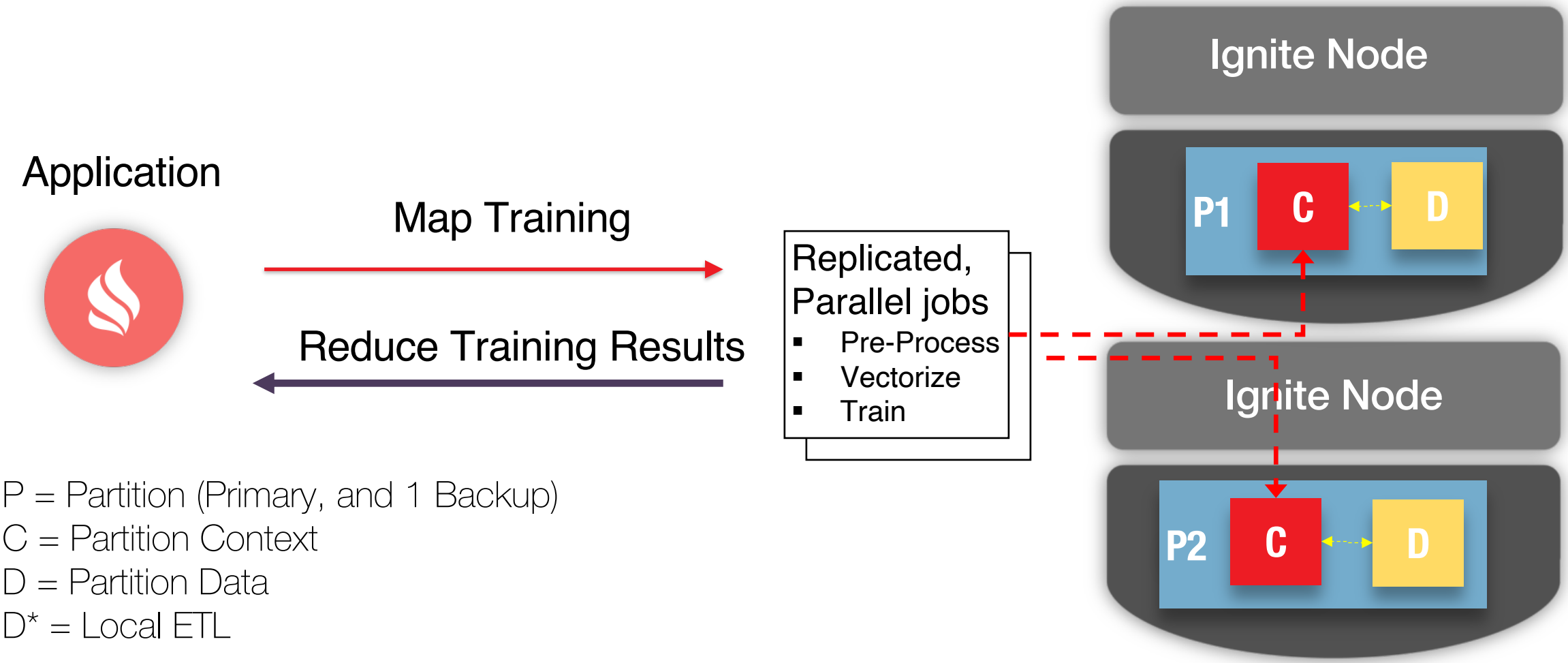
- Ignite ML is deeply integrated into Ignite multi-tiered storage:
  - Optional ETL and no data shuffling during training
  - (Re-)Train across petabytes of in-memory and on-disk data



2019 © GridGain Systems

GridGain

Continuous Machine Learning At Scale:
Enabled with Partitioned Datasets

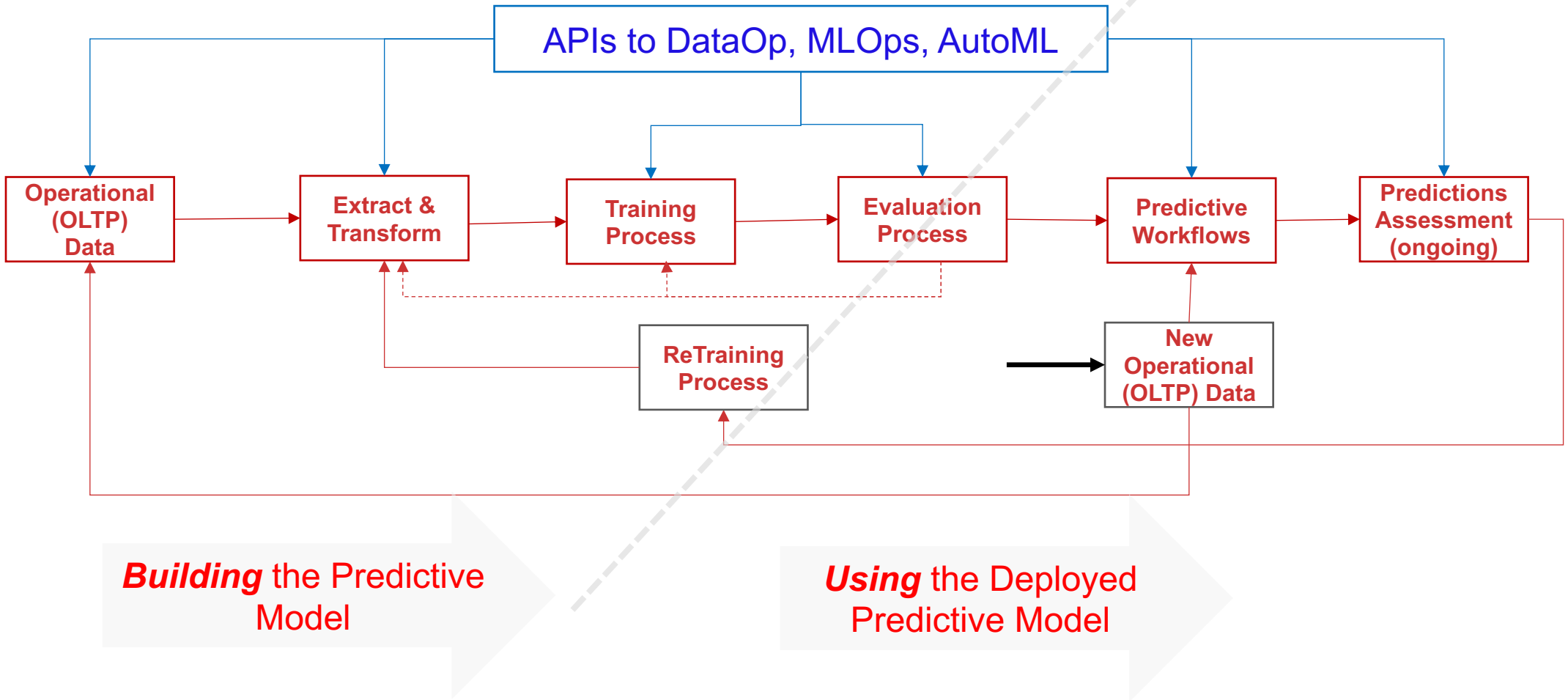


P = Partition (Primary, and 1 Backup)
C = Partition Context
D = Partition Data
D* = Local ETL

2019 © GridGain Systems

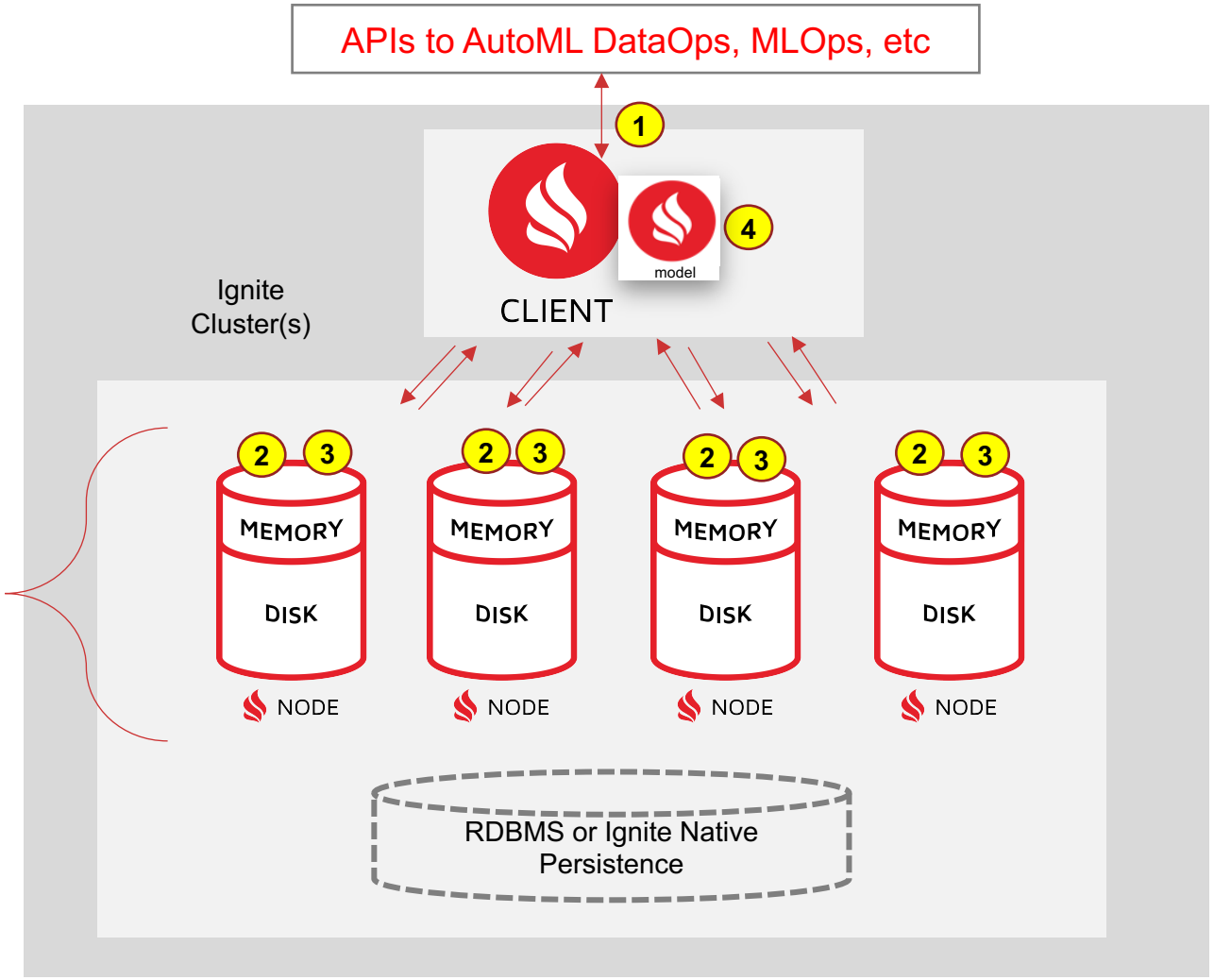
GridGain

Continuous ML at Scale:
Example Deployment Patterns



2019 © GridGain Systems

Deployment Pattern: Building Predictive Models



Co-Located in same partitions

- OLTP data (historical and/or online)
- ML compute tasks

Batch training on large datasets

Latency constraint:

- Batch run
- Approx. 1 / Node count time reduction

Pattern: ML tasks performed where OLTP data already resides

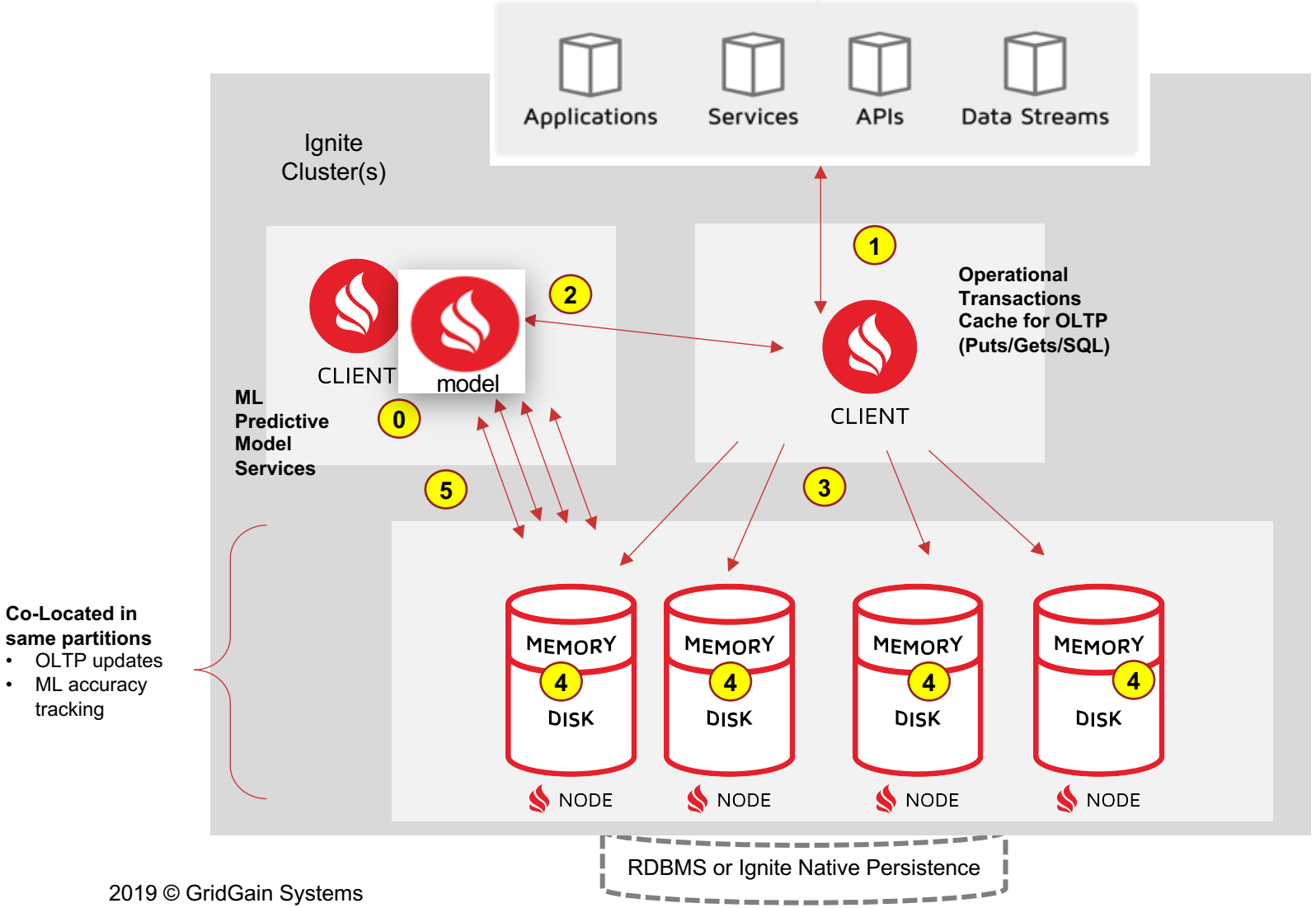
- Extract OLTP into ML dataset
- Preprocess into ML format
- Train w/ selected algorithms
- Evaluate for accuracy

Output:

- Trained ML predictive Model(s) collected in client node(s)
- Select models using evaluation data and other criteria
- Models can be saved and loaded back to other clients

2019 © GridGain Systems

Deployment Pattern: Using the Predictive Models
Real-Time predictions



Latency constraint:

- Per OLTP commit: Application needs to get prediction first before performing commit

Pattern:

- Predictive Model forward deployed close to OLTP application for fast response
- Options include: Same JVM, Same Node, Same cluster

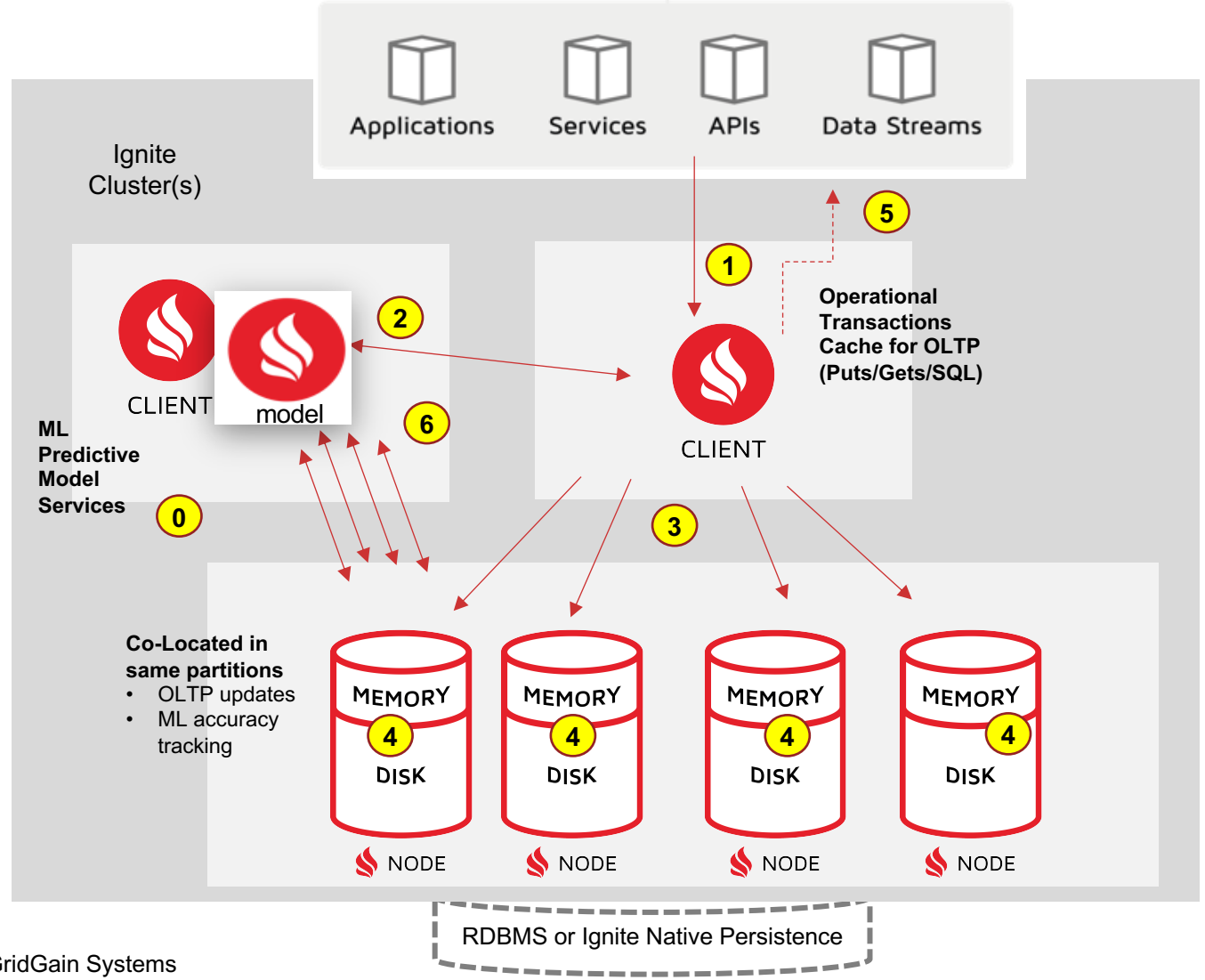
Outputs / results:

- OLTP writes performed in 1/N scale-out
- OLTP writes include both Predictions + Actuals for accuracy tracking

2019 © GridGain Systems

GridGain

Deployment Pattern: Using the Predictive Models
"Near Real-Time" predictions



Latency constraint:

- Prediction doesn't need to be sent back to application client before write to cache; predictions can be done in reasonable size batches
- Workflow prioritization post commit: Claims processing for example

Pattern:

- Predictive Model forward deployed close to OLTP application
- Options include: Same JVM, Same Node, Same cluster

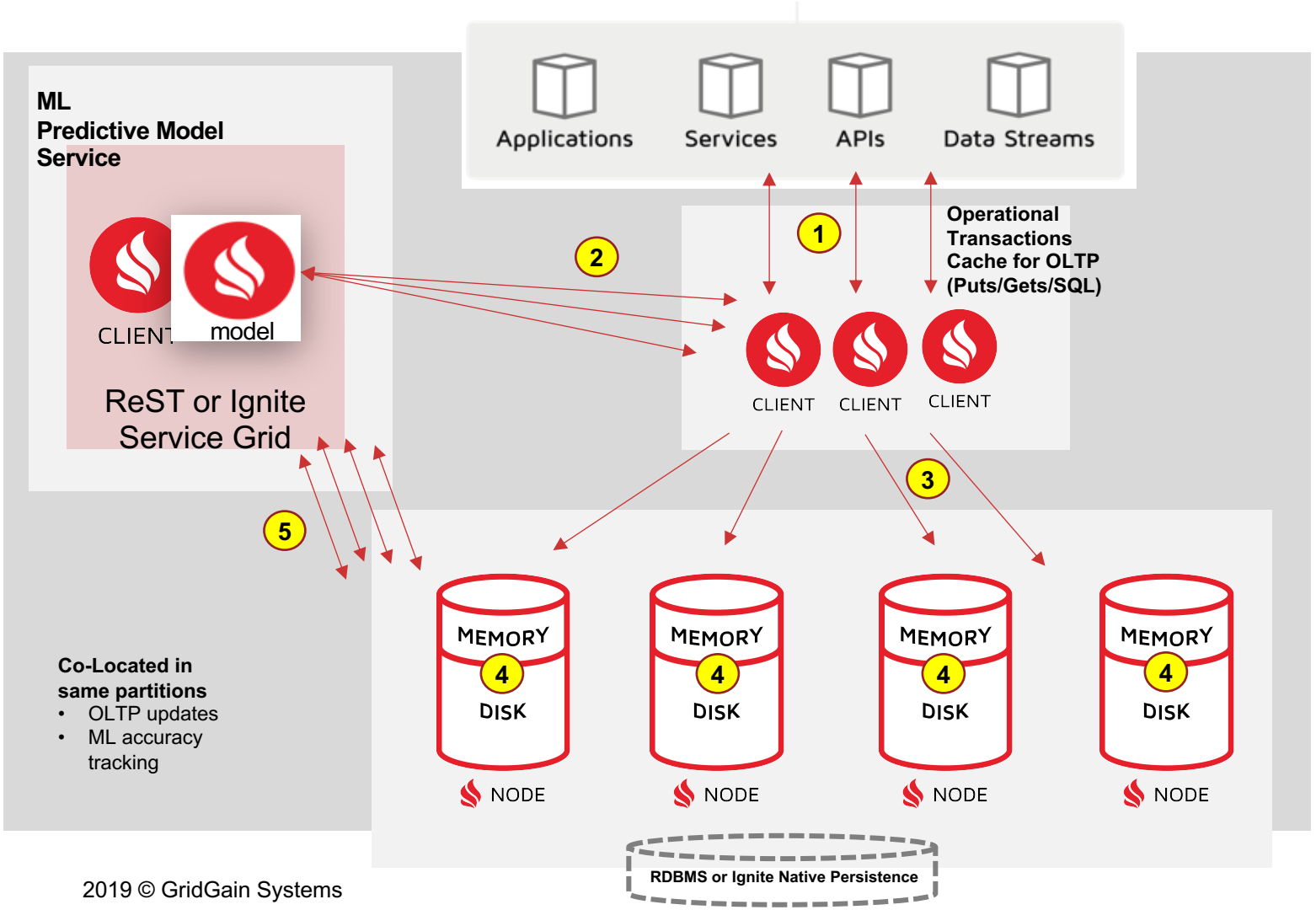
Outputs / results:

- OLTP writes performed in 1/N scale-out
- OLTP writes include both Predictions + Actuals for accuracy tracking at each node

2019 © GridGain Systems

GridGain

Deployment Pattern: Using the Predictive Models
Shared Model as a service



Pattern:

- Replicated OLTP client nodes that share Model service
- Predictive Model forward deployed close to OLTP application, if you need you need predicts prior to commits
- Wrap service for governance (security, metering, routing / transformation)

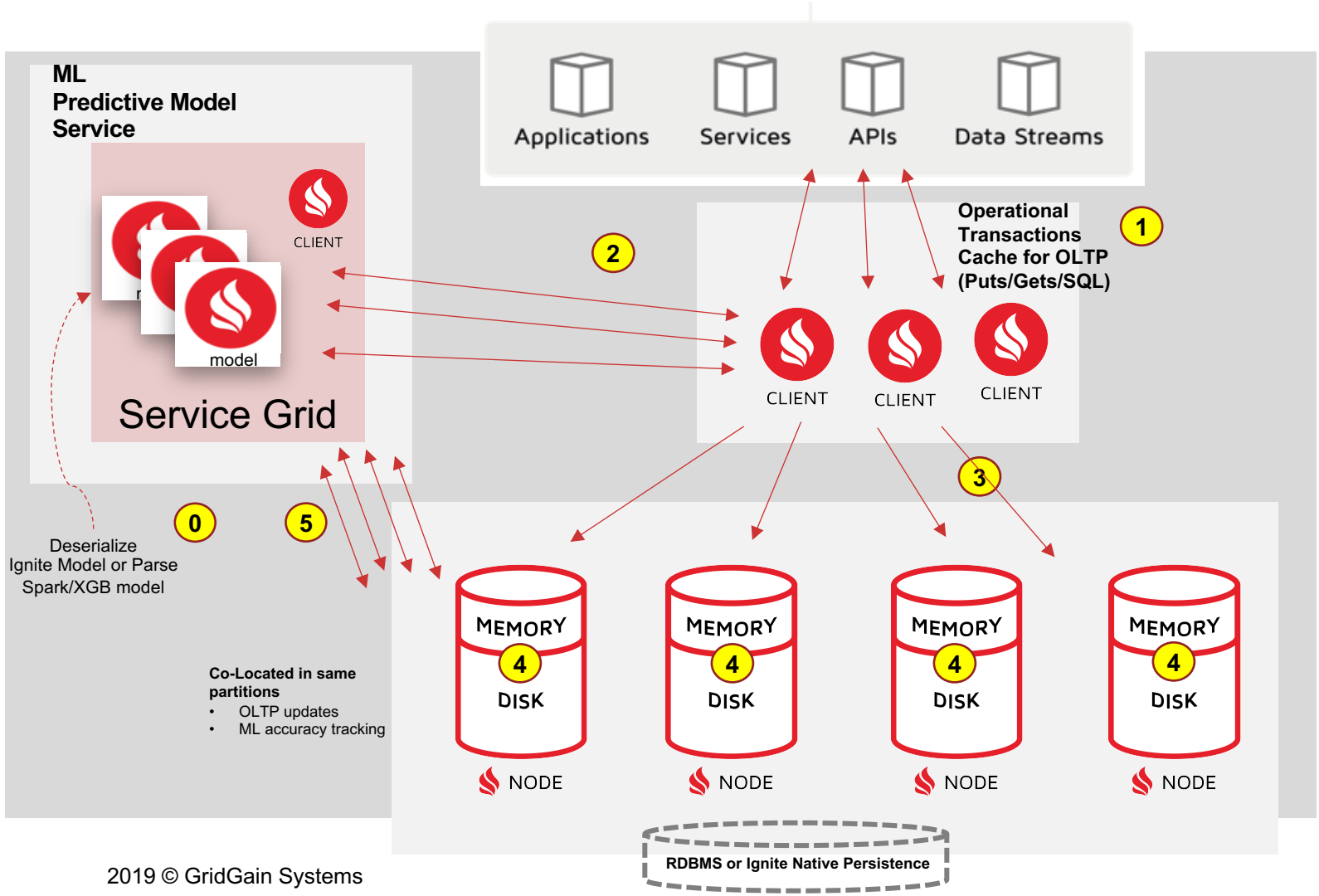
Outputs / results:

- OLTP writes performed in 1/N scale-out
- OLTP writes include both Predictions + Actuals for accuracy tracking

2019 © GridGain Systems

GridGain

Deployment Pattern: Using the Predictive Models
Replicating the Model service



Latency constraint:

- Can supply prediction to application prior to OLTP commit

Pattern:

- Predictive Model forward deployed but now load-balanced
- Identical Model is replicated (Deserialized or Parsed) into each Model node.
- Model needs to be loaded from one source so identical in each instance

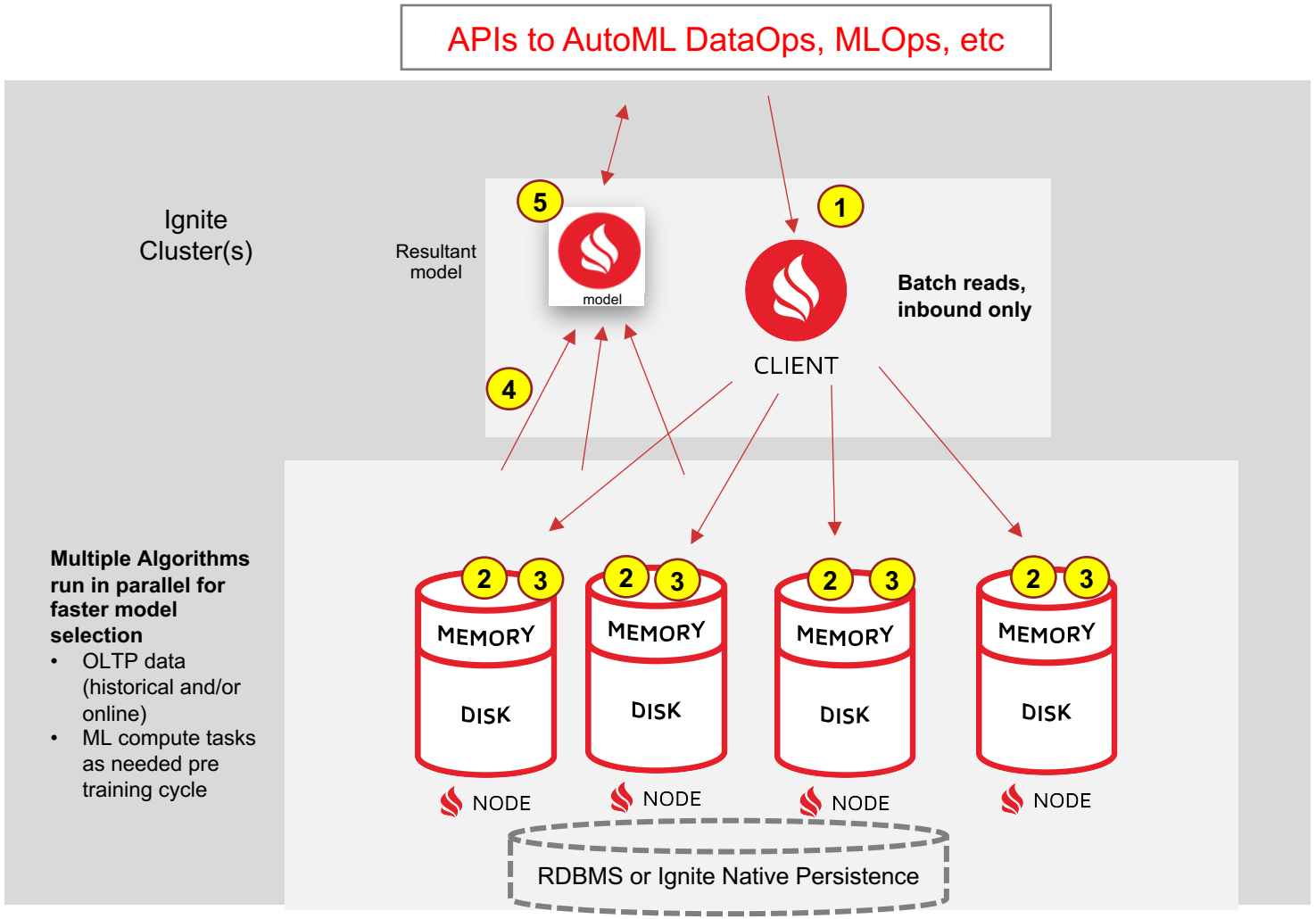
Outputs / results:

- OLTP writes performed in 1/N scale-out
- OLTP writes include both Predictions + Actuals for accuracy tracking

2019 © GridGain Systems

GridGain

Deployment Pattern: Using the Predictive Models
Batch process



Latency constraints

- orchestrating repeat batch runs: no per-record round trips to Application on the way to cluster
- **Not always** a "predict" problem per-se since you might be looking for unknowns
- Examples: clustering for unlabeled population health, customers, network behavior

Deployment Pattern

- Forward deployed Model or co-located Model with data, depends: May require interactive 3rd party AutoML orchestrations via API

Outputs / Results

- Model results collected on client (i.e. Entropy evaluation if Clustering) Parallel processing with co-located data
- Extract, preprocess, train, evaluate, rerun with new data, training, evaluation, reporting…..

15
2019 © GridGain Systems

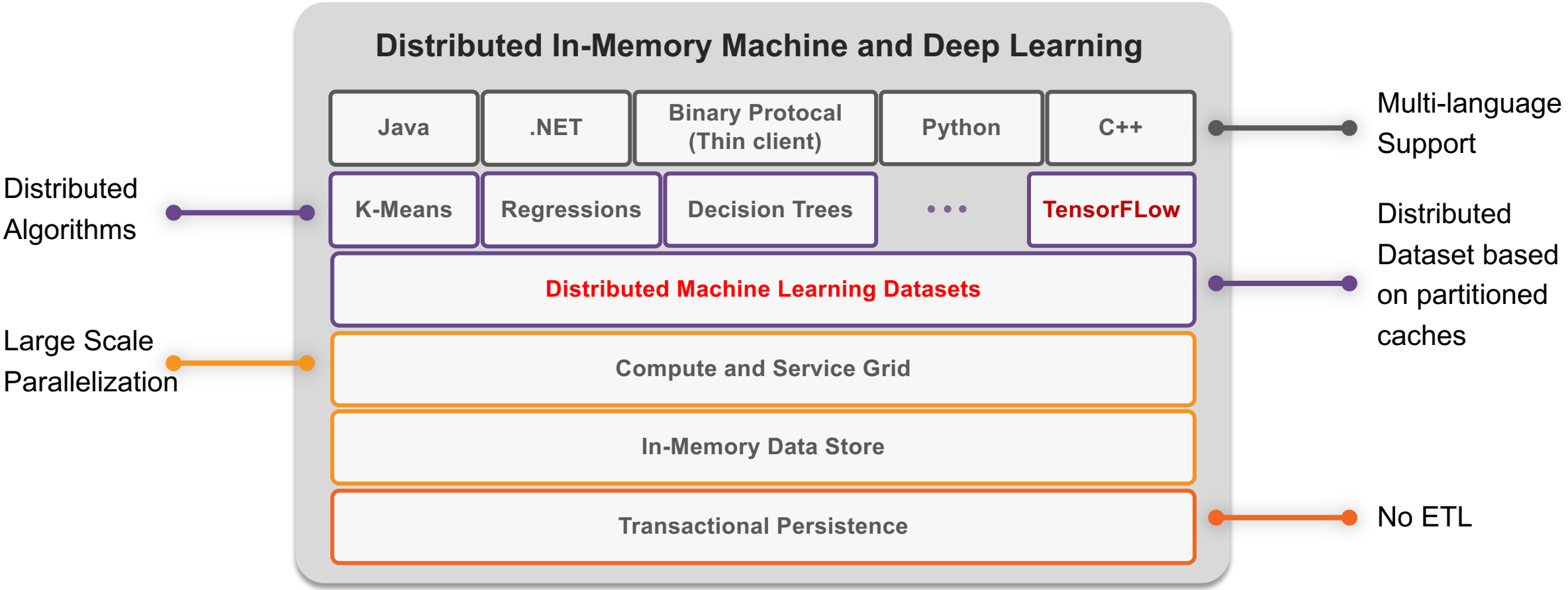
Apache Ignite 2.8 ML/DL

Pipeline APIs, Algorithms

2019 © GridGain Systems

GridGain

Apache Ignite Continuous Learning framework



2019 © GridGain Systems

GridGain

Apache Ignite 2.8 ML Out-Of-The-Box

Algorithms for every ML Pipeline use case

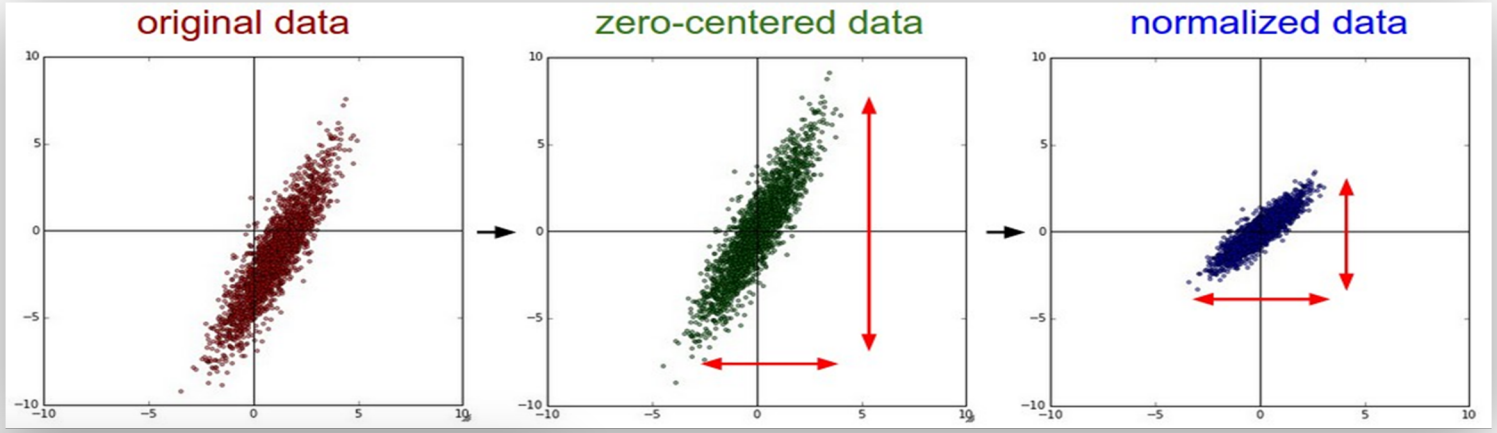
Pipeline Stage	Output
Data Extraction **Preprocessing**	• Feature Vectorization (i.e. columns of interest in ML training) • Handling the major column types: string, categoricals, numbers • Normalization of variables • Preventing overfitting (Cross Validate, Test/Train Split)
Training **Evaluation** (before deployment)	• Regression labeled data for numeric estimation • Classifying labeled data for predicting output classes • Importing from 3rd party work benches XGBoost, Spark, H20 … • Loading previously trained & saved Ignite models
Prediction **Performance** **Retraining**	• Extensive set of predictive Algorithms including newly released Ensembles • Transaction & Batch workloads • Deployment flexibility: Either front-deployed near your Application client or co-located with OLTP cache data

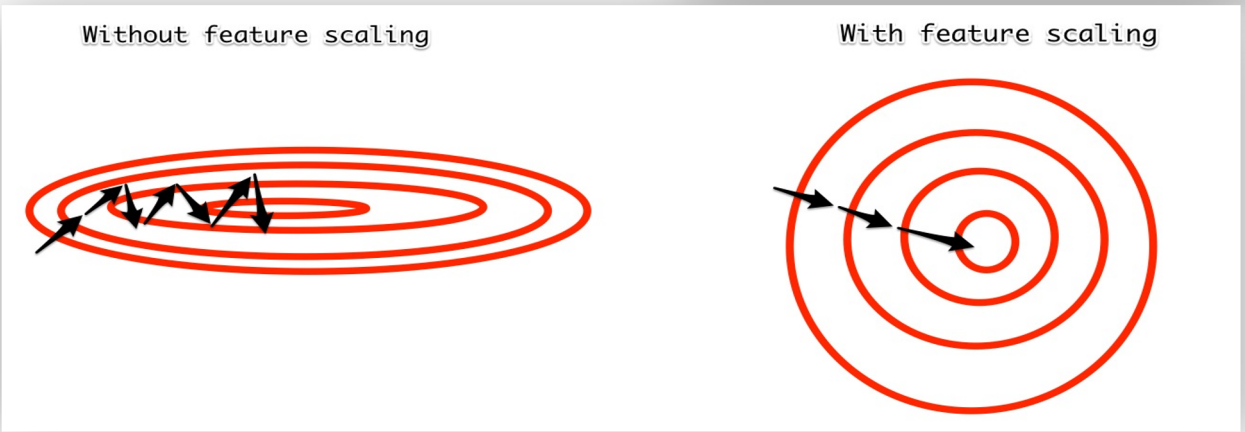
2019 © GridGain Systems

Apache Ignite Distributed PreProcessing:
Normalization & Scaling

Improved Model training velocity and accuracy

- Prevent large magnitude variables from overwhelming smaller ones
- Speed up processing by re-centering

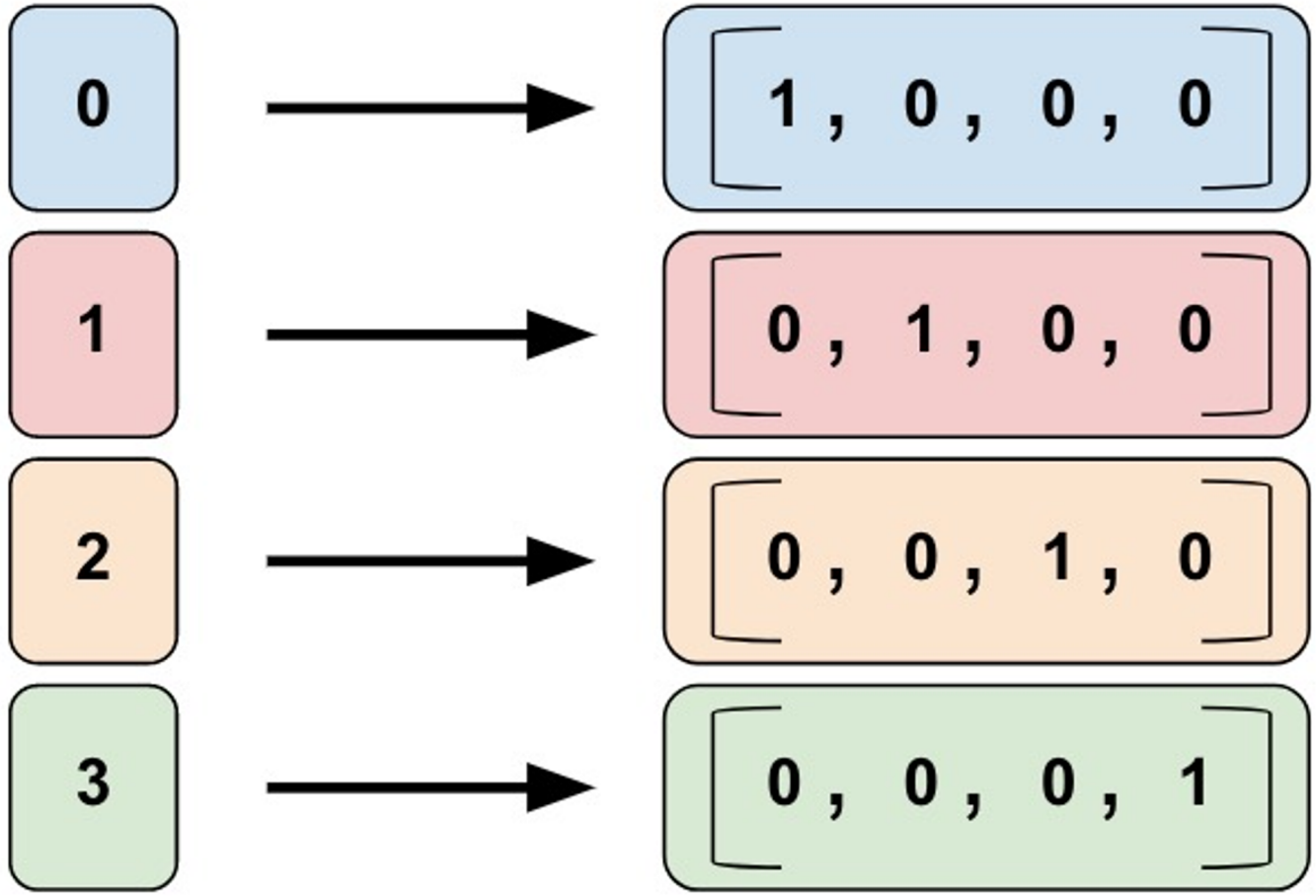




https://medium.com/@nsethi610/data-cleaning-scale-and-normalize-data-4a7c781dd628

2019 © GridGain Systems

Apache Ignite Distributed preprocessing:
Categorical values: One-Hot encoders



* Also included:
String Encoding

2019 © GridGain Systems

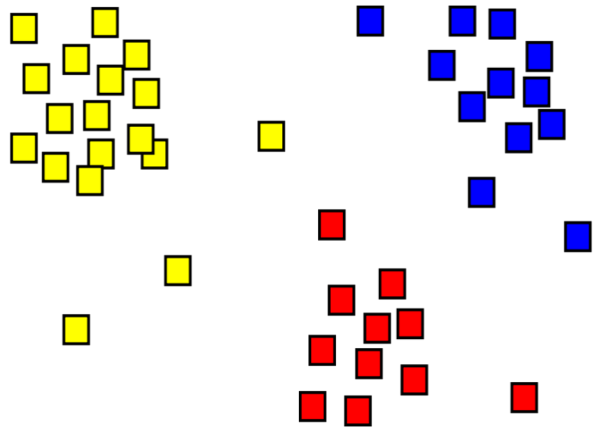
GridGain

Apache Ignite Distributed Training: Clustering

For “unlabeled” data – the data appears continuous but are there actually classes like “high risk”, “medium risk”, “low risk”?

Examples:

- Population Segmentation (health, retail, genomics)
- Anomaly Detection
- Network throughput characterization



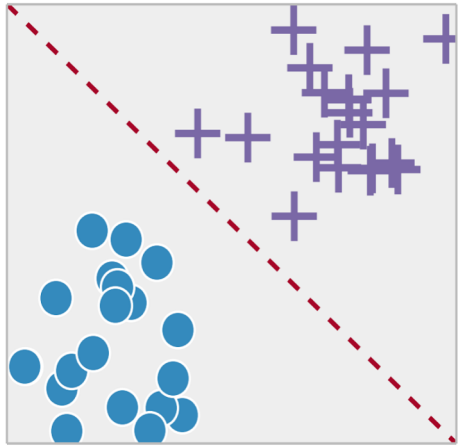
K-means , GMM

2019 © GridGain Systems

Apache Ignite Distributed Training:
Classification & Regression

Predicting output classes

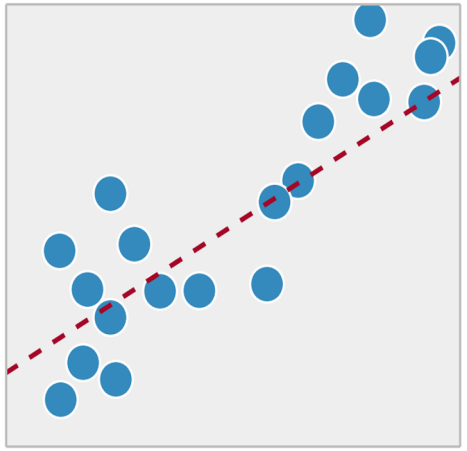
- Fraud detection
- Credit Card Scoring
- Clinical Trials
- Customer Segmentation



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

Numerical Estimation

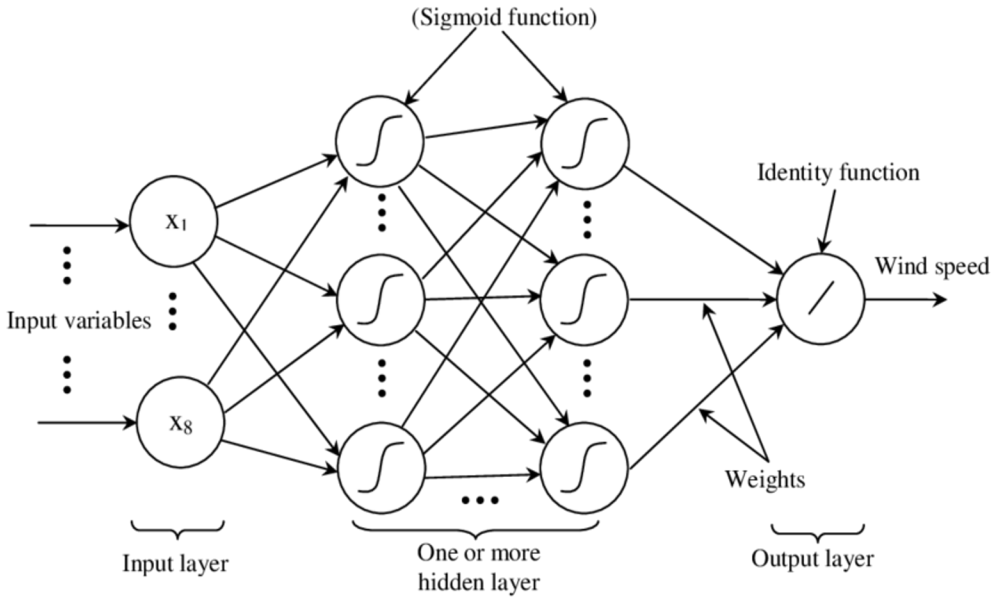
- Trend Analysis
- Financial Forecasting
- Time Series Prediction
- Response Modeling (Pharma, etc)



- KNN & Linear Regressions
- Decision tree regression
- Random forest regression
- Gradient-boosted tree regression
- Import XGBoost, Spark

2019 © GridGain Systems

Apache Ignite: TensorFlow Integration



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

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

```
>>> 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'}}
```


2019 © GridGain Systems

To Summarize: Apache Ignite for Continuous Learning at Scale

Massive Scale for Memory, Storage, (ML) Computation

- Massive Throughput with minimal (or even eliminated) ETL data movements
- Massive operational data sizes + in-place parallel processing
- Faster cycle times for OLTP transactions, ML/DL operations

Integrates with Existing ML / DL operations

- Low-level Distributed APIs to integrate with Auto ML and other Data Science workflows
- Apache Ignite integrations to leverage Spark, TensorFlow pipelines; including Model imports from other tool sets such as XGBoost, Spark, others

2019 © GridGain Systems

Demonstrations, Code Walkthroughs

2019 © GridGain Systems

APIs: Building and Using the Predictive Model

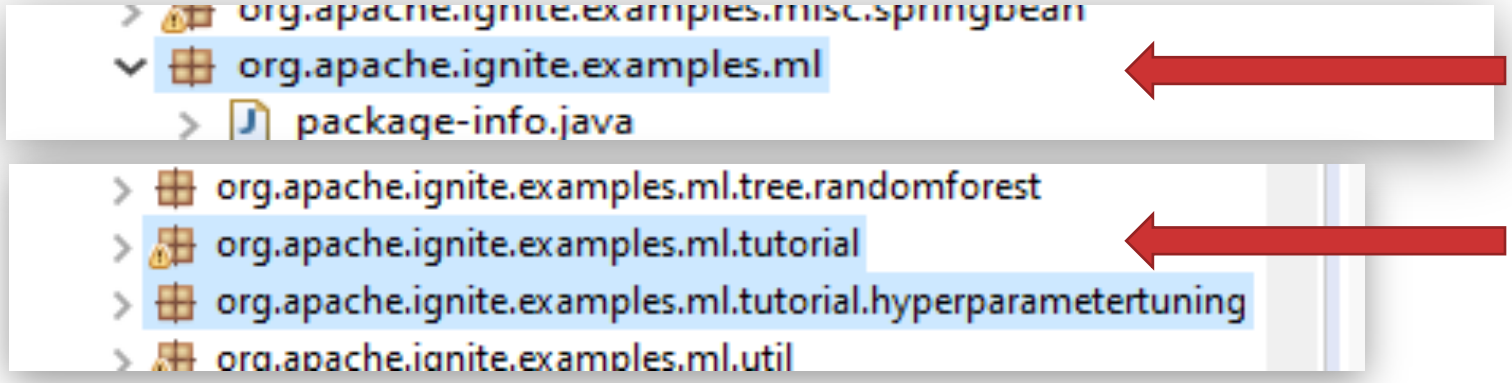
Some representative API call signatures on typical Model object	Pipeline Stage comments

`MyIgniteModel = My<AlgorithmSpecific>Trainer.fit(. . .)`
`MyXGBModel = MyModelBuilder.build(. . . , myXGBParser, . . . )`
`MySparkModel = (MyAlgorithm)SparkModelParser.parse(. . . )`
	Create an instance of the Model: MyModel created either by fit() call (distributed training on the algorithm specific trainer) or parsing from Spark, XGBoost, other from saved Ignite Model . **Note:** Use of the New Pipeline calls means you can replace fit() calls on individual algorithms with a single API if desired

`Accuracyvalue = MyEvaluator.evaluate(. . . , MyModel, . . .)`	**Test the model before you use it:** Accuracy value is a number (double). MyModel passed as one of the inputs for accuracy checking. Several accuracy measures available including: accuracy, F-measure, Precision, Recall, Specificity, MAE …..
`Predictedvalue = MyModel.predict(input field values)`	**Predict outputs for incoming transaction:** "Predicted value" is a double returned from the deployed model to an external application. You should probably wrap this call in a service to enforce things like security, latency, etc
Measuring Predictions versus Actuals performance has no standard API per se. You'll need to capture both values and then do comparisons periodically…	**Track Prediction accuracy over time**: Examples: For **Regression**, a common measure for error is MAE (Mean Absolute Error) or MSE (Mean Squared Error). For **Classification**, (Correct # guesses / total # guesses)
`MyModelVersion2 = MyModelVersion1.update(...,` `newTransactionsCache, MyModelVersion1, . . . )`	**When model accuracy drifts over time, update with latest data sample:** Input arguments include a reference to the older model plus a reference to the new set of data

2019 © GridGain Systems

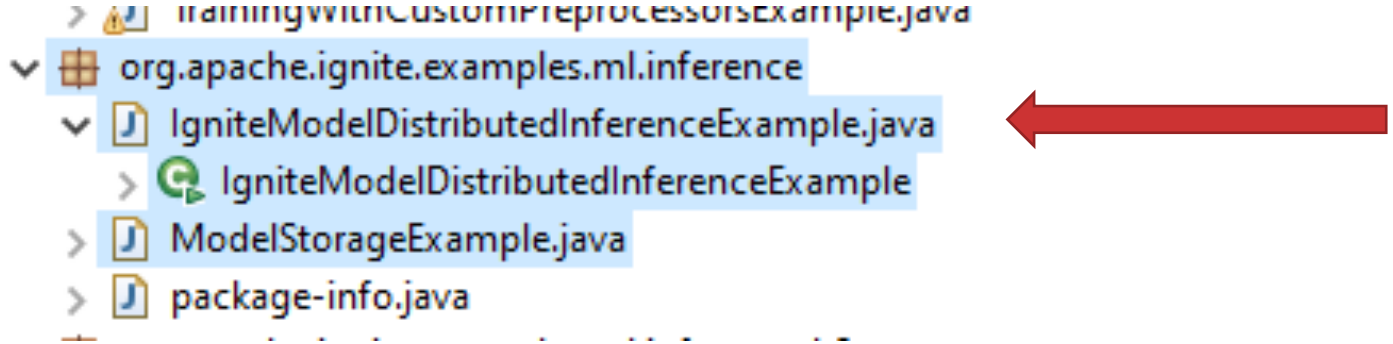
Building and Using the Predictive Model: try out these examples, many more….



All ML examples contained in this package

Building Models Tutorial package provides best overall overview (Enhanced for 2.8)

- More algorithms added, including hyperparameter tuning steps
- Complete pipeline shows accuracy comparisons for each algorithms: pipeline aggregrator, missing values processor, categoricals processor, scaling, test-train split, cross validation, several trainers…



Using a Model Deploying then invoking `output = mdl.predict(inputrecord)`

- Also show loading to reuse a saved Model from storage

2019 © GridGain Systems

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

2019 © GridGain Systems

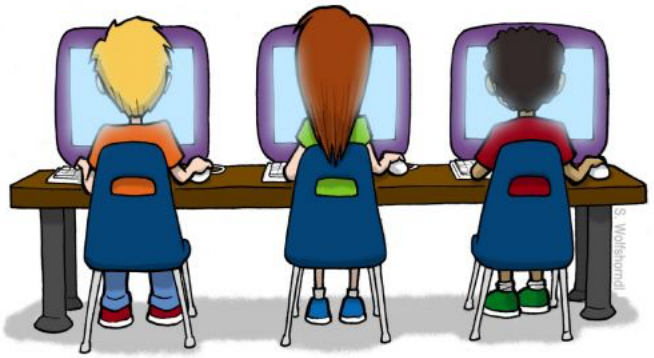
Advanced Toolkit for Machine Learning Experts

- Pipelining APIs
  - https://apacheignite.readme.io/docs/pipeline-api
- Evaluators
  - https://apacheignite.readme.io/docs/evaluator
- Model cross-validation
  - https://apacheignite.readme.io/docs/cross-validation
- Models ensembling
  - https://apacheignite.readme.io/docs/ensemble-methods

2019 © GridGain Systems

Join Apache Ignite Community!

- Rapidly Growing Engineering Community
- Great Way to Learn Distributed Systems, Computing, SQL, ML, Transactions
- How To Contribute:
  - https://ignite.apache.org/community/contribute.html
- Join Ignite Meetups:
  - https://ignite.apache.org/meetup-groups.html



2019 © GridGain Systems

Q & A

2019 © GridGain Systems

GridGain