In-Memory Performance
Durability of Disk
Scalable Machine and Deep Learning with Apache Ignite

Denis Magda
Apache Ignite PMC Chair
GridGain Director of Product Management
Agenda

• Why Machine Learning at Scale?
• Ignite Machine Learning
• Genetic Algorithms
• TensorFlow Integration
• Demo
• Q&A
Why Machine Learning at Scale?

1. Models trained and deployed in different systems
   - Move data out for training
   - Wait for training to complete
   - Redeploy models in production

2. Scalability
   - Data exceed capacity of single server
   - Burden for developers
Memory-Centric Storage

- Predictable memory consumption
- Fully Transactional (Write-Ahead Log)
- Instantaneous Restarts
- Automatic Defragmentation
- Off-heap Removes noticeable GC pauses
- Stores Superset of Data

Server Node

IN-MEMORY

ON-DISK
Machine Learning

- Multi-Language Support
- Distributed Core Algebra
- Dense and Sparse Algebra
- No ETL
- Large Scale Parallelization
- Distributed Algorithms

Languages:
- R
- C++
- Python
- Java
- Scala
- REST
Record to Node Mapping

Key → Partition → Server Node
ON-DISK
Caches and Partitions

Partition 1
- K1, V1
- K2, V2
- K3, V3
- K4, V4

Partition 2
- K5, V5
- K6, V6
- K7, V7
- K8, V8
- K9, V9
Training Failover

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

Ignite Node 1

Ignite Node 2
# Algorithms and Applicability

<table>
<thead>
<tr>
<th></th>
<th>Classification</th>
<th>Regression</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Identify to which category a new observation belongs, on the basis of a training set of data</td>
<td>Modeling the relationship between a scalar dependent variable ( y ) and one or more explanatory variables ( x )</td>
</tr>
<tr>
<td><strong>Applicability</strong></td>
<td>spam detection, image recognition, credit scoring, disease identification</td>
<td>drug response, stock prices, supermarket revenue</td>
</tr>
<tr>
<td><strong>Algorithms</strong></td>
<td>nearest neighbor, decision tree classification, neural network</td>
<td>linear regression, decision tree regression, nearest neighbor, neural network</td>
</tr>
</tbody>
</table>
### Algorithms and Applicability

<table>
<thead>
<tr>
<th></th>
<th>Clustering</th>
<th>Preprocessing</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Grouping a set of objects in such a way that objects in the same group are more similar to each other than to those in other groups</td>
<td>Feature extraction and normalization</td>
</tr>
<tr>
<td><strong>Applicability</strong></td>
<td>customer segmentation, grouping experiment outcomes, grouping shopping items</td>
<td>transform input data, such as text, for use with machine learning algorithms</td>
</tr>
<tr>
<td><strong>Algorithms</strong></td>
<td>k-means</td>
<td>Normalization preprocessor</td>
</tr>
</tbody>
</table>
Linear Regression

• Ordinary Least Squares
• Linear Regression Trainer
  • QR Decomposition
  • Gradient Descent

// y = bx + a
LinearRegressionModel model = trainer.train(trainSet);
double prediction = model.predict(sampleObject);

// Prepare trainSet
...

// QR Decomposition
LinearRegressionQRTrainer trainer = new LinearRegressionQRTrainer();
LinearRegressionModel mdl = trainer.train(trainSet);

// Gradient Descent
LinearRegressionSGDTrainer trainer = new LinearRegressionSGDTrainer(1000, 1e-6);
LinearRegressionModel mdl = trainer.train(trainSet);
Decision Trees

- Data stored by features
- Related data on same node
- Features
  - Continuous
  - Categorical

```java
// Train the model
DecisionTreeModel mdl = trainer.train(
    new BiIndexedCacheColumnDecisionTreeTrainerInput(
        cache, new HashMap<>(), ptsCnt, featCnt));

// Estimate the model on the test set
IgniteTriFunction<Model<Vector, Double>,
    Stream<IgniteBiTuple<Vector, Double>>,
    Function<Double, Double>,
    Double> mse = Estimators.errorsPercentage();

Double accuracy = mse.apply(mdl, testMnistStream.map(
    v -> new IgniteBiTuple<>(v.viewPart(0, featCnt), v.getX(featCnt))),
    Function.identity());

System.out.println(">>> Errs percentage: " + accuracy);
```
Demo: Fraud Detection
Genetic Algorithms

F = Fitness Calculation
C = Crossover
M = Mutation

F = F1 + F2
C = C1 + C2
M = M1 + M2

Biological Evolution Simulation

Collocated Computation

Chromosome and Genes Cluster

Ignite Cluster

IN-MEMORY
ON-DISK

F1, C1, M1
F2, C2, M2
TensorFlow Integration: Benefits

- Ignite as distributed data source
  - Perfect fit for distributed TF training

- Less ETL
  - TF nodes deployed together with Ignite nodes
  - In-machine data movement only

- TF tasks execution in-place in Ignite
  - Roadmap
TensorFlow Integration: Main Features

- Distribution of user tasks written in Python
- Automatic creation and maintenance of TF cluster
- Minimization of ETL costs
- Fault tolerance for both Ignite and TF instances
Demo: TensorFlow and Ignite
Summary: Apache Ignite Benefits

- Massive scalability
  - Horizontal + Vertical
  - RAM + Disk

- Zero-ETL
  - Train models and run algorithms in place

- Fault tolerance and continuous learning
  - Partition-based dataset
Resources

- Apache Ignite ML Documentation:
  - https://apacheignite.readme.io/docs

- ML Blogging Series:
  - Genetic Algorithms with Apache Ignite
  - Introduction to Machine Learning with Apache Ignite
  - Using Linear Regression with Apache Ignite
  - Using k-NN Classification with Apache Ignite
  - Using K-Means Clustering with Apache Ignite
  - Using Apache Ignite’s Machine Learning for Fraud Detection at Scale
Among Top 5 Apache Projects

Top 5 Developer Mailing Lists
1. Ignite
2. Kafka
3. Tomcat
4. Beam
5. James

Top 5 User Mailing Lists
1. Lucene/Solr
2. Ignite
3. Flink
4. Kafka
5. Cassandra

Top 5 by Commits
1. Hadoop
2. Ambari
3. Camel
4. Ignite
5. Beam

Over 1M downloads per year
Apache Ignite – We’re Hiring ;)

- Very Active Community
- Great Way to Learn Distributed Computing
- How To Contribute:
  - [https://ignite.apache.org/](https://ignite.apache.org/)
Any Questions?

Thank you for joining us. Follow the conversation.

http://ignite.apache.org

@ApachIgnite
@gridgain
@denismagda