Scalable Online Learning for Flink: SOLMA Library

Hamid Bouchachia
Bournemouth University
Department of Computing and Informatics
United Kingdom
abouchachia@Bournemouth.ac.uk
1. CONTEXT
2. SCALABLE ONLINE MACHINE LEARNING
3. SOLMA
4. IMPLEMENTATION CONSIDERATIONS
5. CONCLUSIONS
Work conducted in the context of a H2020 project, called PROTEUS

Big Data platform, Flink

Why new machine learning library for Flink?
- Flink is equipped with FlinkML, a machine learning library dedicated for offline learning
- FlinkML supports optimised routines for handling sparse and dense matrix as well as BLAS (Basic Linear Algebra Subprograms)-compliant operations.
- Currently FlinkML supports the following algorithms:
  - Supervised Learning: SVM, multiple linear regression
  - Unsupervised Learning: KNN
  - Data Pre-processing: Polynomial Features, Standard Scaler, MinMax Scaler
  - Recommendation: Alternating Least Squares
  - Outlier selection: Stochastic Outlier Selection
  - Utilities: Distance Metrics, Cross Validation
Context: What is Apache Flink?

- Apache Flink is a **Big Data open source platform** for scalable batch and stream data processing.

- Started in 2009 by the Berlin-based database research groups (Stratosphere project).

- Accepted as Apache Incubator project in April 2014 and became **Apache Top-Level project** in December 2014.

- About 120 contributors, highly active community.
Apache Flink comparison

### Batch processing

<table>
<thead>
<tr>
<th></th>
<th>Flink</th>
<th>Spark</th>
<th>Storm</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>API</strong></td>
<td>low-level</td>
<td>high-level</td>
<td>high-level</td>
</tr>
<tr>
<td><strong>Data Transfer</strong></td>
<td>batch</td>
<td>batch</td>
<td>pipelined &amp; batch</td>
</tr>
<tr>
<td><strong>Memory Management</strong></td>
<td>disk-based</td>
<td>JVM-managed</td>
<td>Active managed</td>
</tr>
<tr>
<td><strong>Iterations</strong></td>
<td>file system cached</td>
<td>in-memory cached</td>
<td>streamed</td>
</tr>
<tr>
<td><strong>Fault tolerance</strong></td>
<td>task level</td>
<td>task level</td>
<td>job level</td>
</tr>
<tr>
<td><strong>Good at</strong></td>
<td>massive scale out</td>
<td>data exploration</td>
<td>heavy backend &amp; iterative jobs</td>
</tr>
<tr>
<td><strong>Libraries</strong></td>
<td>many external</td>
<td>built-in &amp; external</td>
<td>evolving built-in &amp; external</td>
</tr>
</tbody>
</table>

### Streaming processing

<table>
<thead>
<tr>
<th></th>
<th>Flink</th>
<th>Spark</th>
<th>Storm</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>API</strong></td>
<td>low-level</td>
<td>high-level</td>
<td>high-level</td>
</tr>
<tr>
<td><strong>Fault tolerance</strong></td>
<td>tuple-level ACKs</td>
<td>RDD-based (lineage)</td>
<td>coarse checkpointing</td>
</tr>
<tr>
<td><strong>State</strong></td>
<td>not built-in</td>
<td>external</td>
<td>internal</td>
</tr>
<tr>
<td><strong>Exactly once</strong></td>
<td>at least once</td>
<td>exactly once</td>
<td>exactly once</td>
</tr>
<tr>
<td><strong>Windowing</strong></td>
<td>not built-in</td>
<td>restricted</td>
<td>flexible</td>
</tr>
<tr>
<td><strong>Latency</strong></td>
<td>low</td>
<td>medium</td>
<td>low</td>
</tr>
<tr>
<td><strong>Throughput</strong></td>
<td>medium</td>
<td>high</td>
<td>high</td>
</tr>
</tbody>
</table>
Massive parallel data flow engine with **unified batch and stream processing**
- Batch (DataSet) and Stream (DataStream) APIs on top of a streaming engine

**Rich set of operators (including native iteration)**
- Map, Reduce, Join, CoGroup, Union, Iterate, Delta Iterate, Filter, FlatMap, GroupReduce, Project, Aggregate, Distinct, Vertex-Update, Accumulators, ...

**Programming APIs for Java and Scala (Python upcoming)**

**Flink Optimizer**
- Inspired by optimizers of parallel database systems
- Physical optimization follows cost-based approach

**Memory Management**
- Flink manages its own memory
- Never breaks the JVM heap
SOLMA, Flink & the BD Ecosystem

- **SOLMA**: Scalable Online machine Learning and data Mining Algorithms.
Scalable Online Machine Learning

- **ML challenge: Data Streams**
  - Current state of the art of ML algorithms for BDD is dominated by offline learning algorithms that process *data-at-rest*.
  - Plenty of current data sources are streaming (*online, data-in-motion*): sensors, social networks, clickstream, etc.
  - In online learning, the algorithms see the data only once. The traditional meaning of *online* is that data is processed sequentially one by one but for many epochs.
    - Evolutionary vs. revolutionary
    - No memory vs. partial memory
    - Low computational time
Scalable Online Machine Learning

Online learning

For $t=1, 2, \ldots, T$
- Receive an instance $x_t$
- Predict its class label $\hat{y}_t = \text{sgn}(f_t(x_t))$
- Receive the true class label $y_t$
- Suffer loss $\ell(y_t, f_t(x_t))$
- Update the prediction model $f_t(x) \rightarrow f_{t+1}(x)$

Goal: To minimize the total loss suffered:

$$\sum_{t=1}^{T} \ell(y_t, f_t(x_t))$$

- No prior knowledge about the characteristics of data
- Model is self-adaptive – captures and reacts to changes (dynamic environments)
- Model compares well against offline counterpart
Online learning at scale

- Big Data
  - Flood of data available
  - Internet, Smartphones, IoT, etc.

- Higher performance of computers
  - Larger memory and cheap computer clusters
  - Greater computational power for calculating [parallel and distributed computing]

- Growing interest in online learning and stream processing over the recent years

- Increasing support from industries
  - Web data, sensor networks, traffic, videos, images, webpages, click data, blogs, financial transactions, stock exchange data, telephony data, health data, communication networks, electricity consumption, computational biology and chemistry, astronomy, physics, industry, etc.
Hence, the motivations of this research:

1. The need for scalable online learning algorithms to cope with:
   - Huge volumes of data
   - high-velocity streams

2. Lack of such algorithms for Flink as a novel big data platform.
   - Flink brings both batch and the stream processing together in the same environment
   - SOLMA offers a response to the stream analytics needs for such big data platform
SOLMA Library


- SOLMA covers two classes of algorithms, basic and advanced:
  - Basic streaming routines
    - Online moments: simple mean, simple variance, weighted mean, weighted variance, exponentially weighted mean and variance, moving average, aggregation algorithm.
    - Online sampling: Simple reservoir sampling, weighted reservoir sampling and adaptive reservoir sampling
    - Online frequent directions
    - Incremental principal component analysis.
SOLMA Library

Advanced ML algorithms

- Classification
  - Online support vector machines (OSVM)
  - Online bi-level stochastic gradient for support vector machines (OBSG-SVM)
  - Online passive-aggressive algorithms (PA)

- Regression
  - Online ridge regression (ORR)
  - Online shrinkage via limit of Gibbs sampling (OSLOG)
  - Aggregating algorithm for regression (AAR)
  - Competitive online iterated ridge regression (COIRR)

- Drift handling and anomaly detection
  - Online weighted averaging passive-aggressive algorithm (OWAPA)
  - Online normalised least mean square regression (ONLMSR)
  - Anomaly detection using incremental PCA (IPCA-AD)
SOLMA Library

- Algorithms under development
  - Active learning
    - Bi-criteria online active learning
  - Clustering and topic modelling
    - Latent Dirichlet based (LDA) on stochastic variational inference (SVI)
    - Gaussian mixture models based on stochastic variational inference
Implementation considerations

- SOLMA followed mainly *FlinkML* pipeline
- SOLMA’s API is inspired by *sklearn*. It makes use of 3 interfaces: *Estimator*, *Transformer*, and *Predictor*.
  - *Estimator* is the base class from which *Transformer* and *Predictor* inherit.
  - *Estimator* defines a *fit* method which performs the actual training of the model (classification, clustering, regression: training the model)
  - *Transformer* defines a transform method (like scaling the input, pre-processing, feature reduction and selection, mappings from one feature space to another).
  - *Predictor* defines a predict method: classification, clustering, regression (predicting class/membership/independent variable)

- Pipeline = chaining together one or more Transformers and the final link in a pipeline can be a *Predictor* or another *Transformer*.

- Pipelines that end with Predictor cannot be chained any further.

- SOLMA has been developed considering Flink as the main deployment environment under PROTEUS scope.
Implementation considerations

- The technical challenge in the design of distributed algorithms is how to achieve lossless parallelism for the online learning algorithms.

- Parallelism is achieved by either data parallelism or model parallelism.
  
  - Data parallelism requires replicating the model over different machines.
  
  - The replicas of the model on each machine synchronise the model parameters after a fixed number of presentations (synchronous mode).
  
  - The model replicas may be trained over multiple machines, but each replica is trained over a chunk of data.
  
  - All the distributed algorithms in SOLMA are implemented as Flink programs which are inherently parallel and distributed.
  
  - Model parallelism is partitioned in sub-models that correspond to different tasks.
  
  - Providing model parallelism in existing applications is non-trivial.
    
    - It requires modifying ML algorithms to ensure that the model is split such that the communication costs are limited within each data presentation.
Implementation considerations

- Master-Slave architecture
  
  - The stream is distributed among the available workers and the master receives parameters from each worker, the master updates the parameter and sends the updated parameters to each worker.
  
  - Updates of parameters are asynchronous when the communication between the master and the workers is not synchronised, possibly due to the delay in the arrival of the stream and the converse refers to the synchronous case.
  
  - The implementation of the parameter server allows synchronisation of the model parameters between the workers and the master.
Implementation considerations

- Master-slave architecture
Conclusion

- SOLMA as a library for streaming data developed for Flink.

- It is being populated by basic streaming routines and advanced machine learning algorithms.

- Currently it contains a number of scalable online algorithms.

- In the near future, we aim to cover other algorithms, especially in the area of clustering, semi-supervised and active learning.

- The library is open source and we hope it can be adopted by end-users and extended by contributors.
Thanks for your attention!

www.proteus-bigdata.com

Contact us:

- Prof Hamid Bouchachia
  - abouchachia@bournemouth.ac.uk
  - http://staffprofiles.bournemouth.ac.uk/display/abouchachia