KNIME Big Data Workshop

Björn Lohrmann

KNIME
Variety, Velocity, Volume

- **Variety:**
  - Integrating heterogeneous data...
  - ... and tools

- **Velocity:**
  - Real time scoring of millions of records/sec
  - Continuous data streams
  - Distributed computation

- **Volume:**
  - From small files...
  - ...to distributed data repositories
  - Moving computation to the data
Variety
The KNIME Analytics Platform: Open for Every Data, Tool, and User

Data Scientist

Business Analyst

External Data Connectors

Native Data Access, Analysis, Visualization, and Reporting

Distributed / Cloud Execution

External and Legacy Tools

Microsoft Azure

mongoDB

aws

TIBCO Spotfire

ACTUATE

BIRT

python

SSAS

AN IBM COMPANY

Open for Innovation
Data Integration

<table>
<thead>
<tr>
<th>Sepal length</th>
<th>Sepal width</th>
<th>Petal length</th>
<th>Petal width</th>
<th>Species</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>3.5</td>
<td>1.4</td>
<td>0.2</td>
<td><em>l. setosa</em></td>
</tr>
<tr>
<td>4.9</td>
<td>3.0</td>
<td>1.4</td>
<td>0.2</td>
<td><em>l. setosa</em></td>
</tr>
<tr>
<td>4.7</td>
<td>3.0</td>
<td>1.3</td>
<td>0.2</td>
<td><em>l. setosa</em></td>
</tr>
<tr>
<td>4.6</td>
<td>3.1</td>
<td>1.5</td>
<td>0.2</td>
<td><em>l. setosa</em></td>
</tr>
<tr>
<td>5.0</td>
<td>3.6</td>
<td>1.4</td>
<td>0.2</td>
<td><em>l. setosa</em></td>
</tr>
<tr>
<td>5.4</td>
<td>3.9</td>
<td>1.7</td>
<td>0.4</td>
<td><em>l. setosa</em></td>
</tr>
<tr>
<td>4.6</td>
<td>3.4</td>
<td>1.4</td>
<td>0.3</td>
<td><em>l. setosa</em></td>
</tr>
<tr>
<td>5.0</td>
<td>3.4</td>
<td>1.5</td>
<td>0.2</td>
<td><em>l. setosa</em></td>
</tr>
</tbody>
</table>
Integrating R and Python
Modular Integrations

---

R Snippets
- R Snippet
- R View (Table)
- R Source (Table)

R Workspace Manipulation
- R Source (Workspace)
- R To R
- R View (Workspace)
- R Model Reader
- Table to R
- Add Table To R
- R to Table
- R Model Writer
- R Learner
- R Predictor
- R To PMML

Python Snippets
- Python Script
- Python View
- Python Edit Variable
- Python Script (2:1)
- Python Source

Python Snippets
- Python Learner
- Python Script (DB)
- Python Script (Hive)
- Python Object Reader
- Python Predictor
- Python Object Writer
Other Programming/Scripting Integrations

Java Integrations
- Java Snippet
- Java Snippet Row Filter
- Java Snippet Row Splitter

JavaScript Integrations
- Generic JavaScript View
- JavaScript Table View
- JavaScript Scatter Plot
- JavaScript Lift Chart
- JavaScript Box Plot
- JavaScript ROC Curve
- JavaScript Line Plot
- JavaScript Bar Chart
- JavaScript Pie/Donut Chart

Misc Integrations
- External Tool
- External SSH Tool
- Generic Web Service Client
- SAS7BDAT Reader
Velocity
Velocity

• High Demand Scoring/Prediction:
  – Scoring using KNIME Server
  – Scoring with the KNIME Managed Scoring Service

• Continuous Data Streams
  – Streaming in KNIME
What is High Demand Scoring?

• I have a workflow that takes data, applies an algorithm/model and returns a prediction.
• I need to deploy that to hundreds or thousands of end users
• I need to update the model/workflow periodically
How to make a scoring workflow

- Use JSON Input/Output nodes
- Put workflow on KNIME Server -> REST endpoint for scoring
KNIME Managed Scoring Service

• The KNIME Managed Scoring Service is a hosted service that allows provisioning and consuming of scoring workflows via REST.

• No need to think about servers, hosting, building services, etc...

Work in progress. Public beta available soon!
KNIME Managed Scoring Service

KNIME handles this...

Application Load Balancer

Scaling Metric

Scale up/down With demand

Client application

KNIME Scoring Agent

KNIME Scoring Agent

KNIME Scoring Agent

KNIME Scoring Agent
Velocity

• High Demand Scoring/Prediction:
  – High Performance Scoring using generic Workflows
  – High Performance Scoring of Predictive Models

• Continuous Data Streams
  – Streaming in KNIME
Streaming in KNIME

Streamable nodes are all contained in wrapped node.

Stream

WrappedNode Input Punctuation Erasure N Chars Filter Stop word Filter Snowball Stemmer WrappedNode Output

Node 1
Node 2
Node 12
Volume
Moving computation to the data
Volume

- Database Extension
- Introduction to Hadoop
- KNIME Big Data Connectors
- KNIME Extension for Apache Spark
- KNIME H2O Sparkling Water Integration
- KNIME Workflow Executor for Apache Spark
Database Extension

• Visually assemble complex SQL statements
• Connect to almost all JDBC-compliant databases
• Harness the power of your database within KNIME
In-Database Processing

- Operations are performed within the database
Tip

- SQL statements are logged in KNIME log file
Database Port Types

- Database JDBC Connection Port (light red)
  - Connection information

- Database Connection Port (dark red)
  - Connection information
  - SQL statement

Database Connection Ports can be connected to Database JDBC Connection Ports but not vice versa.
Database Connectors

- Nodes to connect to specific Databases
  - Bundling necessary JDBC drivers
  - Easy to use
  - DB specific behavior/capability
- Hive and Impala connector
- General Database Connector
  - Can connect to any JDBC source
  - Register new JDBC driver via preferences page
Register JDBC Driver

Open KNIME and go to File -> Preferences

Register single jar file JDBC drivers

Register new JDBC driver with companion files

Increase connection timeout for long running database operations
Query Nodes

- Filter rows and columns
- Join tables/queries
- Extract samples
- Bin numeric columns
- Sort your data
- Write your own query
- Aggregate your data
Database GroupBy – Manual Aggregation

Returns number of rows per group
Database GroupBy – DB Specific Aggregation Methods

SQLite 7 aggregation functions

PostgreSQL 25 aggregation functions
Volume

- Database Extension
- Introduction to Hadoop
- KNIME Big Data Connectors
- KNIME Extension for Apache Spark
- KNIME H2O Sparkling Water Integration
- KNIME Workflow Executor for Apache Spark
Apache Hadoop

- Open-source framework for distributed storage and processing of large data sets
- Designed to scale up to thousands of machines
- Does not rely on hardware to provide high availability
  - Handles failures at application layer instead
- First release in 2006
  - Rapid adoption, promoted to top level Apache project in 2008
- Spawned diverse ecosystem of products
Hadoop Ecosystem

Access
- HIVE

Processing
- MapReduce
- Tez
- Spark

Resource Management
- YARN

Storage
- HDFS
HDFS

• Hadoop distributed file system
• Stores large files across multiple machines
HDFS – Data Replication and File Size

Data Replication

• All blocks of a file are stored as sequence of blocks

• Blocks of a file are replicated for fault tolerance (usually 3 replicas)
  – Aims: improve data reliability, availability, and network bandwidth utilization
YARN

• Cluster resource management system
• Two elements
  – Resource manager (one per cluster):
    • Knows where worker nodes are located and how many resources they have
    • Scheduler: Decides how to allocate resources to applications
  – Node manager (many per cluster):
    • Launches application containers
    • Monitor resource usage and report to Resource Manager
MapReduce

Map applies a function to each element

For each word emit: word, 1

Reduce aggregates a list of values to one result

For all equal words sum up count
Hive

- **SQL-like database** on top of files in HDFS
- Provides data summarization, query, and analysis
- Interprets a set of files as a database table (schema information to be provided)
- Translates SQL queries to MapReduce, Tez, or Spark jobs
- Supports various file formats:
  - Text/CSV
  - SequenceFile
  - Avro
  - ORC
  - Parquet
Hive

```
select * from table
```

**MapReduce / Tez / Spark**

```
MAP(...) REDUCE(...)  
```

<table>
<thead>
<tr>
<th>Var1</th>
<th>Var2</th>
<th>Var3</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>x1</td>
<td>z1</td>
<td>n1</td>
<td>y1</td>
</tr>
<tr>
<td>x2</td>
<td>z2</td>
<td>n2</td>
<td>y2</td>
</tr>
<tr>
<td>x3</td>
<td>z3</td>
<td>n3</td>
<td>y3</td>
</tr>
</tbody>
</table>

**DataNodes**

- DataNode: table_1.csv
- DataNode: table_2.csv
- DataNode: table_3.csv
Spark

• Cluster computing framework for large-scale data processing
• Keeps large working datasets in memory between jobs
  – No need to always load data from disk -> much (!) faster than MapReduce
• Programmatic interface
  – Scala, Java, Python, R
  – Functional programming paradigm: map, flatmap, filter, reduce, fold, ...
• Great for:
  – Iterative algorithms
  – Interactive analysis
Spark – Data Representation

**DataFrame:**

- *Table-like*: Collection of rows, organized in columns with names and types

- **Immutable**:
  - Data manipulation = creating new DataFrame from an existing one by applying a *function* on it

- **Lazily evaluated**:
  - Functions are not executed until an *action* is triggered, that requests to actually see the row data

- **Distributed**:
  - Each row belongs to exactly one *partition*
  - Each partition is held by a Spark Executor

<table>
<thead>
<tr>
<th>Name</th>
<th>Surname</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>Doe</td>
<td>35</td>
</tr>
<tr>
<td>Jane</td>
<td>Roe</td>
<td>29</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

**Note:**

- Earlier versions of KNIME and Spark used *RDDs* (resilient distributed datasets)
- In KNIME, DataFrames are always used in Spark 2 and later.
Spark – Lazy Evaluation

• Functions ("transformations") on DataFrames are not executed immediately
• Spark keeps record of the transformations for each DataFrame
• The actual execution is only triggered once the data is needed
• Offers the possibility to optimize the transformation steps
Spark Context

- Spark Context
  - Main entry point for Spark functionality
  - Represents connection to a Spark cluster
  - Allocates resources on the cluster
Volume

• Database Extension
• Introduction to Hadoop
• KNIME Big Data Connectors
• KNIME Extension for Apache Spark
• KNIME H2O Sparkling Water Integration
• KNIME Workflow Executor for Apache Spark
KNIME Big Data Connectors

- Package required drivers/libraries for specific HDFS, Hive, Impala access
- Preconfigured connectors
  - Hive
  - Impala
Hive Loader

• Upload a KNIME data table to Hive/Impala
• Part of the KNIME Big Data Connectors Extension
Hive Loader

- Partitioning influences performance

- Partition columns should not contain missing values

- Use columns that are often used in WHERE clauses

- Use only categorical columns with suitable value range, i.e. not too few distinct values (e.g. 2) and not too many distinct values (e.g. 10 million)
Hive Partitioning

Parent Table

- **Sales**
  - **Jan2017**
    - date ≥ 01-01-2017
    - date ≤ 01-31-2017
  - **Feb2017**
    - date ≥ 02-01-2017
    - date ≤ 02-29-2017
  - **Mar2017**
    - date ≥ 03-01-2017
    - date ≤ 03-31-2017

- **Europe**
  - region = Europe
- **Asia**
  - region = Asia
- **USA**
  - region = USA

Range Partition by date

List Sub-partition by region
HDFS File Handling

- New nodes
  - HDFS Connection
  - HDFS File Permission
- Utilize the existing remote file handling nodes
  - Upload/download files
  - Create/list directories
  - Delete files
HDFS – Access

HDFS Connection WebHDFS Connection

HDFS/WebHDFS

HttpFS Connection

HttpFS

Hadoop

DataNode

DataNode

DataNode

PC

Client, e.g. KNIME

Hadoop

DataNode

DataNode

DataNode

PC

Client, e.g. KNIME

Hadoop

DataNode

DataNode

DataNode

Edge Node

Edge Node

Edge Node

© 2018 KNIME.com AG. All Rights Reserved.
Volume

- Database Extension
- Introduction to Hadoop
- KNIME Big Data Connector
- KNIME Extension for Apache Spark
- KNIME H2O Sparkling Water Integration
- KNIME Workflow Executor for Apache Spark
KNIME Extension for Apache Spark

• Based on Spark MLlib
• Scalable machine learning library
• Runs on Hadoop
• Algorithms for
  – Classification (decision tree, naïve bayes, …)
  – Regression (logistic regression, linear regression, …)
  – Clustering (k-means)
  – Collaborative filtering (ALS)
  – Dimensionality reduction (SVD, PCA)
Spark Integration in KNIME

- **Apache Spark**
  - IO
  - Database to Spark
  - Hive to Spark
  - Impala to Spark
  - Spark to Database
  - Spark to Hive
  - Spark to Impala
  - Read
    - Avro to Spark
    - CSV to Spark
    - JSON to Spark
    - ORC to Spark
    - Parquet to Spark
    - Table to Spark
    - Text to Spark
  - Write
    - Spark to Avro
    - Spark to CSV
    - Spark to JSON
    - Spark to ORC
    - Spark to Parquet
    - Spark to Table
    - Spark to Text
  - Persist Spark DataFrame/RDD
  - Unpersist Spark DataFrame/RDD
- **Column**
  - Convert & Replace
    - Spark Category To Number
    - Spark Column Rename
    - Spark Column Rename (Regex)
    - Spark Compiled Transformations Applier
    - Spark Normalizer
    - Spark Number To Category (Apply)
    - Spark Transformations Applier
  - Split & Combine
    - Spark Joiner
    - Split & Combine
    - Spark Missing Value
    - Spark Missing Value (Apply)
    - Spark Column Filter
- **Mining**
  - Clustering
    - Spark ClusterAssigner
    - Spark k-Means
  - Dimensionality Reduction
    - Spark PCA
    - Spark SVD
  - Item Sets / Association Rules
    - Spark Association Rule (Apply)
  - Spark Association Rule Learner
  - Spark Frequent Item Sets
- **PMML**
  - Spark Compiled Model Predictor
  - Spark MLlib to PMML
  - Spark PMML Model Predictor
- **Prediction**
  - Spark Decision Tree Learner
  - Spark Gradient-Boosted Trees Learner
  - Spark Linear Regression Learner
  - Spark Linear SVM Learner
  - Spark Logistic Regression Learner
  - Spark Naive Bayes Learner
  - Spark Predictor
  - Spark Random Forests Learner
  - Scoring
    - Spark Entropy Scorer
    - Spark Numeric Scorer
    - Spark Scorer
    - Spark Collaborative Filtering Learner
- **Misc**
  - Java Snippet
  - Spark DataFrame Java Snippet
  - Spark DataFrame Java Snippet (Sink)
  - Spark DataFrame Java Snippet (Source)
  - Spark RDD Java Snippet
  - Spark RDD Java Snippet (Sink)
  - Spark RDD Java Snippet (Source)
- **Row**
  - Spark Concatenate
  - Spark GroupBy
  - Spark Partitioning
  - Spark Pivot
  - Spark Row Filter
  - Spark Row Sampling
  - Spark Sorter
- **Statistics**
  - Spark Correlation Filter
  - Spark Correlation Matrix
  - Spark Linear Correlation
  - Spark Statistics
  - Create Local Big Data Environment
  - Create Spark Context (Jobserver)
  - Create Spark Context (Livy)
  - Destroy Spark Context
Spark Contexts: Creating

Three nodes to create a Spark context:

• Create Local Big Data Environment
  – Runs Spark locally on your machine (no cluster required)
  – Good for workflow prototyping

• Create Spark Context (Livy)
  – Requires a cluster that provides the Livy service
  – Good for production use

• Create Spark Context (Jobserver)
  – Requires a cluster that provides the Spark Jobserver service
  – Deprecated, not recommended anymore
Spark Contexts: Using, Destroying

- Spark Context port is required by all Spark nodes
- Destroying a Spark Context destroys all Spark DataFrames within the context
Create Spark Context (Livy)

- Allows to use Spark nodes on clusters with Apache Livy
- Out-of-the-box compatibility with:
  - Hortonworks (v2.6.3 and higher)
  - Amazon EMR (v5.9.0 and higher)
  - Azure HDInsight (v3.6 and higher)
Import Data to Spark

- From KNIME
- From CSV file in HDFS
- From Hive
- From other sources
- From Database
Modularize and Execute Your Own Spark Code: PySpark Script
MLlib Integration: Familiar Usage Model

- Usage model and dialogs similar to existing nodes
- No coding required
- Various algorithms for classification, regression and clustering supported
MLlib Integration: Spark MLlib Model Port

- MLlib model ports for model transfer
- Model ports provide more information about the model itself
MLlib Integration: Categorical features

- MLlib learner nodes only support numeric features and labels
- String columns (with categorical values) need to be mapped to numeric first
Mllib Integration: Categorical Values for Decision Tree Algorithms

• Tree algorithms have **optional** PMML input port
  – If connected, hints to Decision Tree algorithm which numeric columns are actually categorical in nature
  – Improves performance in some cases
Spark Predictor Node

- Spark Predictor assigns labels based on a given MLlib model
- Supports all **supervised** classification & regression MLlib models
Volume

- Database Extension
- Introduction to Hadoop
- KNIME Big Data Connector
- KNIME Extension for Apache Spark
- KNIME H2O Sparkling Water Integration
- KNIME Workflow Executor for Apache Spark
H2O Integration

- KNIME integrates the H2O machine learning library
- H2O: Open source, focus on scalability and performance
- Supports many different models
  - Generalized Linear Model
  - Gradient Boosting Machine
  - Random Forest
  - k-Means, PCA, Naive Bayes, etc. and more to come!
- Includes support for MOJO model objects for deployment
- Sparkling water = H2O on Spark
The H2O Sparkling Water Integration

Train Churn Prediction Model on Spark using H2O Sparkling Water

1. Create Local Big Data Environment
2. Create H2O Sparkling Water Context
3. Database Table Selector
4. Hive to Spark
5. Spark to H2O
6. H2O Partitioning
7. H2O Gradient Boosting Machine Learner
8. H2O Model to MOJO
9. H2O MOJO Writer
10. Save Churn Prediction Model
11. H2O Predictor (Classification)
12. H2O Binomial Scorer
13. Evaluate Model

Deploy Mojo Model on KNIME Server

1. H2O MOJO Reader
2. Read Churn Prediction Model
3. JSON to Table
4. H2O MOJO Predictor (Classification)
5. Columns to JSON
6. Container Output (JSON)
7. Result of REST call
8. Input from REST call
Volume

- Database Extension
- Introduction to Hadoop
- KNIME Big Data Connector
- KNIME Extension for Apache Spark
- KNIME H2O Sparkling Water Integration
- KNIME Workflow Executor for Apache Spark
KNIME Workflow Executor for Apache Spark
Use Cases & Limitations

• Each workflow replica processes the rows of one partition!

• **Good match for:**
  – KNIME nodes that operate row-by-row
    • Many pre- and postprocessing nodes
    • Predictor nodes
    • Nodes that are streamable
  – Parallel execution of standard KNIME workflows on “small” data
    • Hyper-parameter optimization

• **Bad match for:** Any node that needs all rows, such as
  – GroupBy, Joiner, Pivoting, ...
  – Model learner nodes
Big Data, IoT, and the three V

• Variety:
  – KNIME inherently well-suited: open platform
  – broad data source/type support
  – extensive tool integration

• Velocity:
  – High Performance Scoring of predictive models
  – Streaming execution

• Volume:
  – Bring the computation to the data
  – Big Data Extensions cover ETL and model learning
  – Distributed Execution of KNIME workflows
The KNIME® trademark and logo and OPEN FOR INNOVATION® trademark are used by KNIME.com AG under license from KNIME GmbH, and are registered in the United States.