XROOTD POPULARITY ON HADOOP CLUSTERS

Marco MEONI
Luca MENICHETTI
Domenico GIORDANO
Nicolò MAGINI
Tommaso BOCCALI

10/13/2016 – CHEP 2016 – San Francisco
In a Nutshell

- Dataset (DS) popularity is very important to CMS operations
- Current implementation in Oracle has scalability issue
  - Throughput limits in RDBMS clusters, relational constraints
- Migration to Hadoop, in harmony with CERN IT strategy
  - Grid monitoring and dashboard infrastructures
  - Hadoop parallelism optimized for Big Data
- DS popularity on Hadoop scales with data volume
CMS DS Popularity

- CMS data files are grouped in DS with common physics content
  - Average DS size: 350 GB, 100..1000+ files
  - ~500k DS containing 60M files from detector and simulations recorded since the beginning of CMS operations
  - Distributed among 70 PB of disk storages on WLCG computing centres
- Data distribution is based on popularity of the datasets
  - need to make optimal choice of replication to maximize data availability for processing and analysis
- Definition of “popularity” from several perspectives:
  - Data management: a DS “attracts” many accesses
  - Computing facility: lots of CPU hours spent processing a DS
  - User community: many users interested in analyzing a DS
- In this work, a DS is popular when used “often” in analysis jobs
CMS Xrootd Popularity Service

- Based on the monitoring infrastructure for the Xrootd servers
- File access on storage at WLCG sites for local and remote processing
- \textbf{In production since 2012 to monitor DS popularity on EOS storage at CERN}
- \textbf{O(Billions) raw data rows recorded in Oracle: scaling limitations, impractical reprocess to get new statistics}
Migrating from Oracle to Hadoop

• CERN IT Hadoop Service
  • 2 clusters, 52 nodes, Intel(R) Xeon(R) 4*8 cores
  • 416 total cores, 4.5PB SATA3 HDD, 3.4TB RAM

• Common strategy for Popularity
  • Implementing new version of Popularity aggregation service using Big Data tools to process RAW data on HDFS
  • AWG@CERN-IT and INFN/CMS@Pisa collaboration

• 2 orthogonal aspects
  • Big Data Analytics (handle massive data volumes)
  • Machine Learning (learn insights from data)
DATA INGESTION AND VALIDATION
Xrootd Popularity Service on Hadoop

- Streaming raw file access data into HDFS since March 2015
- Present work: implement popularity statistics aggregation with Spark jobs reproducing the old Oracle Materialized Views
Hadoop Aggregation

- Hadoop: re-processing of any time interval is fast
- Oracle: continuous running of incremental MV update, 5x speedup
Oracle vs Hadoop Deltas

- Example: aggregation by DS-name
- 3 metrics: numAccesses, readBytes, procTime
Pig vs Spark

- Spark offers better performance than MapReduce-based toolkits
- Resilient Distributed Dataset, Shared Memory, Persist(), etc…
Mobile Dashboard

- Site-driven UI for popularity data
PREDICTION OF DATASET POPULARITY
Mining DS Popularity On Hadoop

• What is the problem?
  • Predict the Dataset popularity

• Why is it important?
  • reactive: monitor historical info of DS usage (post-factum)
  • proactive: predict DS popularity using a model trained on metadata

• What is the contribution?
  • DS popularity prediction models based on Big Data technology
  • Evaluation on a large scale system (+ efficiency, - cost)

• … work in progress ....
Raw Data and Feature Selection

- Collect 2015’s raw data from heterogeneous sources (O(billions))
- Extract training features

<table>
<thead>
<tr>
<th>Source</th>
<th>#records</th>
<th>Type</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>EOS</td>
<td>786,934,116</td>
<td>structured</td>
<td>Disk storage system at CERN</td>
</tr>
<tr>
<td>AAA</td>
<td>1,682,509,226</td>
<td>structured</td>
<td>CMS XrootD federation for Grid data</td>
</tr>
<tr>
<td>CRAB</td>
<td>1,177,951</td>
<td>structured</td>
<td>Grid infrastructure for job submission</td>
</tr>
<tr>
<td>DBS3</td>
<td>5,193,522</td>
<td>structured</td>
<td>Global DS/fileblocks catalogue</td>
</tr>
<tr>
<td>Block-Replicas</td>
<td>805,614,541</td>
<td>structured</td>
<td>Global replica catalogue</td>
</tr>
<tr>
<td>PhEDEx</td>
<td>58,227,786</td>
<td>structured</td>
<td>Fileblock locator and export service</td>
</tr>
<tr>
<td>CADI</td>
<td>1,791</td>
<td>semi-struct</td>
<td>CMS Analysis database</td>
</tr>
</tbody>
</table>

Feature Selection

<table>
<thead>
<tr>
<th>Metric</th>
<th>Physics</th>
<th>Extra</th>
</tr>
</thead>
<tbody>
<tr>
<td>week</td>
<td>campaign</td>
<td>country</td>
</tr>
<tr>
<td>size</td>
<td>sub-campaign</td>
<td>conferenceID</td>
</tr>
<tr>
<td>nFiles</td>
<td>version</td>
<td>protocol</td>
</tr>
<tr>
<td>nBlocks</td>
<td>process</td>
<td></td>
</tr>
<tr>
<td>nSites</td>
<td>generator</td>
<td></td>
</tr>
<tr>
<td>nEvents</td>
<td>energy</td>
<td></td>
</tr>
<tr>
<td>luminosity</td>
<td>datatier</td>
<td></td>
</tr>
<tr>
<td></td>
<td>software</td>
<td></td>
</tr>
<tr>
<td></td>
<td>acquisitionEra</td>
<td></td>
</tr>
</tbody>
</table>
Popularity Cutoffs

- Train several classifiers with different cutoffs
  - Use threshold that splits popular and non-popular DSs with 1:10 ratio
Classifier Performance

- Rolling Forecast
  - Get new week, score the model, test accuracy, improve the model...
- Entirely developed in Spark with MLlib

<table>
<thead>
<tr>
<th>Classifier</th>
<th>auROC</th>
<th>Accuracy</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decision Tree</td>
<td>0.647</td>
<td>0.603</td>
<td>0.641</td>
<td>0.716</td>
<td>0.753</td>
</tr>
<tr>
<td>SVM</td>
<td>0.660</td>
<td>0.694</td>
<td>0.643</td>
<td>0.716</td>
<td>0.733</td>
</tr>
<tr>
<td>Logistic Regression</td>
<td>0.750</td>
<td>0.761</td>
<td>0.858</td>
<td>0.743</td>
<td>0.850</td>
</tr>
<tr>
<td>Random Forest</td>
<td>0.773</td>
<td>0.749</td>
<td>0.855</td>
<td>0.922</td>
<td>0.866</td>
</tr>
<tr>
<td>GBT</td>
<td>0.779</td>
<td>0.757</td>
<td>0.861</td>
<td>0.991</td>
<td>0.816</td>
</tr>
</tbody>
</table>
Conclusions

- XrootD DS popularity is very important to CMS operations
  - Current Oracle implementation has performance issues
- Implementation in Hadoop
  - Fast re-processing of any time interval, 5x speedup, scalable
- Prediction of DS popularity
  - First attempt on Big Data architecture
  - Train several models, compare performance, calculate accuracy