Distributed Machine Learning with the HPCC Systems Platform

Presenter: Dr. Flavio Villanustre
Lead of the HPCC Systems Initiative

http://hpccsystems.com
FAU Seminar

Distributed Machine Learning with the HPCC Systems Platform
Dr. Flavio Villanustre, VP Technology & Product - LexisNexis

Agenda

11:00am – 11:15am: Welcome
11:15am – 12:15pm: Presentation
12:15pm – 1:00pm: Q&A / Open discussion, Trivia & Raffle

Twitter event hashtag:
#hpccmeetup
Big Data in Gartner’s Hype Cycle

Big Data reaching Plateau in 2 to 5 years

Source: Gartner
Real Life Graph Analytics

Scenario

This view of carrier data shows seven known fraud claims and an additional linked claim.

The Insurance company data only finds a connection between two of the seven claims, and only identified one other claim as being weakly connected.
Real Life Graph Analytics

Task

After adding the LexID to the carrier Data, LexisNexis HPCC technology then explored 2 additional degrees of relative separation.

Result

The results showed **two family groups interconnected on all of these seven claims**.

The links were much stronger than the carrier data previously supported.
Property Transaction Risk

Three core transaction variables measured

• Velocity

• Profit (or not)

• Buyer to Seller Relationship Distance (Potential of Collusion)
Property Transaction Risk

±700 mill Deeds

Data Factory Clean

Collusion Graph Analytics

Chronological Analysis of all property Sales

Derived Public Data Relationships from +/- 50 terabyte database

Large Scale Suspicious Cluster Ranking

Historical Property Sales Indicators and Counts

Person / Network Level Indicators and Counts
Suspicious Equity Stripping Cluster
Results

Large scale measurement of influencers strategically placed to potentially direct suspicious transactions.

• All BIG DATA on one supercomputer measuring over a decade of property transfers nationwide.

• BIG DATA Products to turn other BIG DATA into compelling intelligence.

• Large Scale Graph Analytics allow for identifying known unknowns.

• Florida Proof of Concept
  – Highest ranked influencers
    ▪ Identified known ringleaders in flipping and equity stripping schemes.
    ▪ Typically not connected directly to suspicious transactions.
  – Known ringleaders not the Highest Ranking.

• Clusters with high levels of potential collusion.
• Clusters offloading property, generating defaults.
• Agile Framework able to keep step with emerging schemes in real estate.
Medicaid Case Study

Scenario
Proof of concept for Office of the Medicaid Inspector Generation (OMIG) of large Northeastern state. Social groups game the Medicaid system which results in fraud and improper payments.

Task
Given a large list of names and addresses, identify social clusters of Medicaid recipients living in expensive houses, driving expensive houses.

Result
Interesting recipients were identified using asset variables, revealing hundreds of high-end automobiles and properties.

Leveraging the Public Data Social Graph, large social groups of interesting recipients were identified along with links to provider networks.

The analysis identified key individuals not in the data supplied along with connections to suspicious volumes of “property flipping” potentially indicative of mortgage fraud and money laundering.
Social Graph and Prescriptions

Scenario
Healthcare insurers need better analytics to identify drug seeking behavior and schemes that recruit members to use their membership fraudulently.
Groups of people collude to source schedule drugs through multiple members to avoid being detected by rules based systems.
Providers recruit members to provide and escalate services that are not rendered.

Task
Given a large set of prescriptions. Calculate normal social distributions of each brand and detect where there is an unusual socialization of prescriptions and services.

Result
The analysis detected social groups that are sourcing Vicodin and other schedule drugs. Identifies prescribers and pharmacies involved to help the insurer focus investigations and intervene strategically to mitigate risk.
Social Graph Analytics - Collusion

• LexisNexis Public Data Social Graph (PDSG)
  • Public Data relationships.
  • High Value relationships for Mapping trusted networks.
• Large Scale Data Fabrication and Analytics.
  • Thousands of data sources to ingest, clean, aggregate and link.
  • 300 million people, 4 billion relationships, 700 million deeds.
  • 140 billion intermediate data points when running analysis.
• HPCC Systems from LexisNexis Risk Solutions
  • Open Source Data Intensive high performance supercomputer.
    (http://hpccsystems.com)
• Innovative Examples leveraging the LexisNexis PDSG
  • Healthcare.
    • Medicaid\Medicare Fraud.
    • Drug Seeking Behavior
  • Financial Services.
    • Mortgage Fraud.
    • Anti Money Laundering.
    • “Bust out” Fraud.
• Potential Collusion (The value in detecting non arms length transactions)
Network Traffic Analysis in Seconds

Scenario
Conventional network sensor and monitoring solutions are constrained by inability to quickly ingest massive data volumes for analysis
- 15 minutes of network traffic can generate 4 Terabytes of data, which can take 6 hours to process
- 90 days of network traffic can add up to 300+ Terabytes

Task
Drill into all the data to see if any US government systems have communicated with any suspect systems of foreign organizations in the last 6 months
- In this scenario, we look specifically for traffic occurring at unusual hours of the day

Result
In seconds, the HPCC sorted through months of network traffic to identify patterns and suspicious behavior

Horizontal axis: time on a logarithmic scale
Vertical axis: standard deviation (in hundredths)
Bubble size: number of observed transmissions

LexisNexis
**Scenario**
Calculate Google Page Rank to be used to rank search results and drive visualizations.

**Task**
Load the 75GIG English Wikipedia XML snapshot. Strip page links from all pages and run 20 iterations of Google Page Rank Algorithm. Generate indexes and Roxie query to support visualization.

**Result**
Produces +- 300 million links between 15 million pages. Page Rank allows for ranking results in searching and driving more intuitive visualizations.

**Advanced**
Lays a foundation for advanced graph algorithms that combine ranking, Natural Language Processing and Machine Learning in scale.
Wikipedia Pageview Demo

Scenario

Task
Generate meaningful statistics to understand aggregated global interest in all Wikipedia pages across the 24 hours of the day built off all English Wikipedia page view logs for 12 months.

Result
Produces page statistics that can be queried in seconds to visualize which key times of day each Wikipedia page is more actively viewed.
Helps gain insight into both regional and time of day key interest periods in certain topics.
This result can be leveraged with Machine Learning to cluster pages with similar 24hr Fingerprints.
The Data/Information flow

- **High Performance Computing Cluster Platform (HPCC)** enables data integration on a scale not previously available and real-time answers to millions of users. Built for big data and proven for 10 years with enterprise customers.
- **Offers a single architecture**, two data platforms (query and refinery) and a consistent data-intensive programming language (ECL)
- **ECL Parallel Programming Language** optimized for business differentiating data intensive applications
The HPCC Systems platform

• Open Source distributed data-intensive computing platform
• Shared-nothing architecture
• Runs on commodity computing/storage nodes
• Binary packages available for the most common Linux distributions
• Provides for end-to-end Big Data workflow management services
• Originally designed around 1999 (predates the original paper on MapReduce from Dec. ‘04)
• Improved over a decade of real-world Big Data analytics
• In use across critical production environments throughout LexisNexis for more than 10 years
Components

• The HPCC Systems platform includes:
  • Thor: batch oriented data manipulation, linking and analytics engine
  • Roxie: real-time data delivery and analytics engine

• A high level declarative data oriented language: ECL
  • Implicitly parallel
  • No side effects
  • Code/data encapsulation
  • Extensible
  • Highly optimized
  • Builds graphical execution plans
  • Compiles into C++ and native machine code
  • Common to Thor and Roxie

• An extensive library of ECL modules, including data profiling, linking and Machine Learning
The Three HPCC components

1. HPCC Data Refinery (Thor)
   - Massively Parallel Extract Transform and Load (ETL) engine
   - Enables data integration on a scale not previously available:
     - Suitable for:
       - Massive joins/merges
       - Massive sorts & transformations
   - Programmable using ECL

2. HPCC Data Delivery Engine (Roxie)
   - A massively parallel, high throughput, structured query response engine
   - Low latency, highly concurrent and highly available
   - Allows compound indices to be built onto data for efficient retrieval
   - Suitable for:
     - Volumes of structured queries
     - Full text ranked Boolean search
     - Real time analytics
   - Programmable using ECL

3. Enterprise Control Language (ECL)
   - An easy to use, declarative data-centric programming language optimized for large-scale data management and query processing
   - Highly efficient; automatically distributes workload across all nodes.
   - Automatic parallelization and synchronization of sequential algorithms for parallel and distributed processing
   - Large library of efficient modules to handle common data manipulation tasks

Conclusion: End to End solution
- No need for any third party tools
The HPCC Systems platform

High Performance Computing Cluster (HPCC)

Unstructured Semi-structured Big Data

Big Data

Extraction Transformation Loading

THOR Cluster (Data Refinery)

Concurrent Realtime Delivery

ROXIE Cluster (Data Delivery)

ESP

Query Results

ECL

ECL Developer Using ECL IDE
Detailed HPCC Architecture

- **Auxiliary Components**
  - Support nodes
    - Dali
    - Dali Backup
    - Authentic/Authorize
  - Data
  - Landing Zone
  - Clients
  - ESP Server

- **Nodes**
  - Thor Master
  - Node 1
  - Node ...
  - Node n

- **Support nodes**
  - Dali
  - Dali Backup
  - Authentic/Authorize

- **Thor**
  (Batch Job Execution Engine + DFS)
  Physical Layout Schematic Diagram

- **Roxie**
  (Rapid Delivery Engine)
  Physical Layout Schematic Diagram

http://hpccsystems.com
Enterprise Control Language (ECL)

**Declarative programming language**: Describe what needs to be done and not how to do it

**Powerful**: Unlike Java, high level primitives as JOIN, TRANSFORM, PROJECT, SORT, DISTRIBUTE, MAP, etc. are available. Higher level code means fewer programmers & shortens time to delivery

**Extensible**: As new attributes are defined, they become primitives that other programmers can use

**Implicitly parallel**: Parallelism is built into the underlying platform. The programmer needs not be concerned with it

**Maintainable**: A high level programming language, no side effects and attribute encapsulation provide for more succinct, reliable and easier to troubleshoot code

**Complete**: ECL provides for a complete data programming paradigm

**Homogeneous**: One language to express data algorithms across the entire HPCC platform, including data ETL and high speed data delivery

```c
// Initialize output log
log_out_init := project(log_init,
    transform(layout_logout),
    self := left,
    self := []);

// Create error log
outererrorfile := join(log_seq,
    log_out_init,
    left.linenum = right.linenum,
    transform(recordof(log_seq),
        self := left),
    left only,
    hash);

// Denormalize key value pairs
outlogfile := sort(denormalize(distribute(log
    sort(distribute(key)
    left.linenum = right.linenum,
    transform(layout_logout,
        self.keyvals := left
        row({rid
        self := left),
```
Enterprise Control Language (ECL)

- ECL is a declarative, data-centric, programming language which can be expressed concisely, parallelizes naturally, is free from side effects, and results in highly-optimized executable code.

- ECL is designed for a specific problem domain (data-intensive computing), which makes resulting programs clearer, more compact, and more expressive. ECL provides a more natural way to think about data processing problems for large distributed datasets.

- Since ECL is declarative, execution is not determined by the order of the language statements, but from the sequence of dataflows and transformations represented by the language statements. The ECL compiler determines the optimum execution strategy and graph.

- ECL incorporates transparent and implicit parallelism regardless of the size of the computing cluster and reduces the complexity of parallel programming increasing the productivity of application developers.

- The ECL compiler generates highly optimized C++ for execution.

- ECL provides a comprehensive IDE and programming tools including an Eclipse plugin.

- ECL is provided with a large library of efficient modules to handle common data manipulation tasks.
Terasort Benchmark results

Execution Time (seconds)

Productivity

Space/Cost

```
// Perform global terasort
rec := record
  string8 key;
  string8 seq;
  string8 fill;
end;
in := :FILE('terasort1',rec,FLAT);
OUTPUT(OUTPUT,red,y,WHITENBLI),'nhst:terasort1out',overwrite);
//end
```

```
abstract int findPartition(Text key);
abstract void print(PrintStream stm) throws IOException;
int getLevel() {
  return level;
}

/*
 * An inner trie node that contains 26 children based on the next
 * character.
 */
static class InnerTrieNode extends TrieNode {
  private TrieNode[] child;
  new TrieNode(26);

  InnerTrieNode(int level) {
    super(level);
    int findPartition(Text key) {
      int level = getLevel();
      if (key.getLength() <= level )
        return child[0].findPartition(key);
      return child[key.getLength() & (int)(findPartition(key));
```
Benefits

Speed
- Scales to extreme workloads quickly and easily
- Increase speed of development leads to faster production/delivery
- Improved developer productivity

Capacity
- Enables massive joins, merges, sorts and data transformations
- State of the art Big Data workflow management
- Increases business responsiveness
- Accelerates creation of new services via rapid prototyping capabilities
- Offers a platform for collaboration and innovation leading to better results

Cost Savings
- Commodity hardware and fewer people can do much more in less time
- Uses IT resources efficiently via sharing and higher system utilization
Machine Learning on HPCC

- Extensible Machine Learning Library developed in ECL
- Fully distributed across the cluster
- General statistical functions
- Supports supervised, semi-supervised and unsupervised learning methods
- Document manipulation, tokenization and statistical Natural Language Processing
- A consistent and standard interface to classification ("pluggable classifiers")
- Efficient handling of iterative algorithms (for example, k-means)
- Open Source and available at: http://hpccsystems.com/ml
Machine Learning on HPCC

- ML on a general-purpose Big Data platform means effective analytics in-situ

- The combination of Thor and Roxie is ideal when, for example, training a model on massive amounts of labeled historical records (Thor), and providing real-time classification for new unlabeled data (Roxie)

REMEmBER! When applying Machine Learning methods to Big Data: data profiling, parsing, cleansing, normalization, standardization and feature extraction represent 85% of the problem!
ECL-ML: extensible ML on HPCC

General aspects
- Based on a distributed ECL linear algebra framework
- New algorithms can be quickly developed and implemented
- Common interface to classification (pluggable classifiers)

ML algorithms
- Linear regression
- Several Classifiers
- Multiple clustering methods
- Association analysis

Document manipulation and statistical grammar-free NLP
- Tokenization
- CoLocation

Statistics
- General statistical methods
- Correlation
- Cardinality
- Ranking
Linear Algebra library

- Support for sparse matrices
- Standard underlying matrix/vector data structures
- Basic operations (addition, products, transpositions)
- Determinant/inversions
- Factorization/SVD/Cholesky/Rank/UL
- PCA
- Eigenvectors/Eigenvalues
- Interpolation (Lanczos)
- Identity
- Covariance
- KD Trees
Kettle: a GUI to ECL-ML
HPCC-PaperBoat Integration

- Completed in about 2 weeks by one person without prior ECL experience
- Represents about 100,000 lines of existing C++ libraries
- Uses embedded C++ wrappers, ECL macros and ECL functions
- Seamless from an ECL programming standpoint
HPCC-PaperBoat ML

**Available now**
- All nearest neighbors
- Kernel Density Estimation and non-parametric Bayes Classifier
- Linear Regression
- LASSO
- Support Vector Machine
- Non-Negative Matrix Factorization
- Singular Value Decomposition

**Coming soon**
- Non-parametric regression
- Decision trees
- Principal Component Analysis
- Orthogonal Range Search
- Mean Shift
- Ensemble Singular Value Decomposition
- Multi-Time Series Prediction
- Maximum Variance Unfolding
- 3D Tensor Factorization
- Graph Formation/Diffusion
Beyond ECL: SALT

- The acronym stands for “Scalable Automated Linking Technology”
- Templates based ECL code generator
- Provides for automated data profiling, parsing, cleansing, normalization and standardization
- Sophisticated specificity and relatives based linking and clustering

42 Lines of SALT
3,980 Lines of ECL
482,410 Lines of C++
Beyond ECL: SALT (ii)

- Calculates record matching field weights based on term specificity and matching weights
- What is the chance that two records for “John Smith” refer to the same person? How about “Flavio Villanustre”?
- It also takes into account transitive relationships
- What if these two records for “John Smith” were already linked to “Flavio Villanustre”? How many “John Smiths” does “Flavio Villanustre” know? Are these two “John Smith” records referring to the same person now?
SALT Demo

Distributed Machine Learning with the HPCC Systems Platform
### SALT Demo

Distributed Machine Learning with the HPCC Systems Platform
SALT Demo

Distributed Machine Learning with the HPCC Systems Platform

<table>
<thead>
<tr>
<th>Column</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>sec range</td>
<td>76</td>
</tr>
<tr>
<td>city</td>
<td>36</td>
</tr>
<tr>
<td>state</td>
<td>18</td>
</tr>
<tr>
<td>zip</td>
<td>40</td>
</tr>
<tr>
<td>zip4</td>
<td>197</td>
</tr>
<tr>
<td>county</td>
<td>25</td>
</tr>
<tr>
<td>msa</td>
<td>15</td>
</tr>
<tr>
<td>phone</td>
<td>23</td>
</tr>
<tr>
<td>fein</td>
<td>4</td>
</tr>
</tbody>
</table>

Record Layout

```
Record.Layout ::= RECORD
UNSIGNED5 ccid;
UNSIGNED5 bdid;
STRING2 source;
STRING34 vendor_id;
STRING5 dt_first_seen;
UNSIGNED4 dt_last_seen;
STRING37 company_name;
STRING24 prim_range;
STRING4 predir;
STRING11 prim_name;
STRING6 addr_suffix;
STRING2 postdir;
STRING4 unit_design;
STRING9 sec_range;
STRING12 city;
STRING9 state;
STRING5 zip;
UNSIGNED3 zip4;
UNSIGNED2 county;
UNSIGNED2 msa;
UNSIGNED5 phone;
UNSIGNED5 fein;
END
```
SALT/Kettle integration
The Future: Knowledge Engineering

At the end of the day

- Do I really care about the format of the data?
- Do I even care about the placement of the data?
- I do care (a lot!) about what can be inferred from the data
- The context is important as long as it affects my inference process
- I want to leverage existing algorithms

<table>
<thead>
<tr>
<th>ECL</th>
<th>KEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generates C++ (1-&gt;100)</td>
<td>Generates ECL (1-&gt;12)</td>
</tr>
<tr>
<td>Files and Records</td>
<td>Entities and associations</td>
</tr>
<tr>
<td>Detailed control of data format</td>
<td>Loose control of input format; none of processing</td>
</tr>
<tr>
<td>Can write graph and statistical algorithms</td>
<td>Major algorithms built in</td>
</tr>
<tr>
<td>Thor/Roxie split by human design</td>
<td>Thor/Roxie split by system design</td>
</tr>
<tr>
<td>Solid, reliable and mature</td>
<td>R&amp;D</td>
</tr>
</tbody>
</table>
KEL by example (WIP!)

- Actor := ENTITY( FLAT(UID(ActorName),Actor=ActorName) )
- Movie := ENTITY( FLAT(UID(MovieName),Title=MovieName) )
- Appearance := ASSOCIATION( FLAT(Actor Who,Movie What) )

- USE IMDB.File_Actors(FLAT,Actor,Movie,Appearance)

- CoStar := ASSOCIATION( FLAT(Actor Who,Actor WhoElse) )

- GLOBAL: Appearance(#1,#2) Appearance(#3,#2) => CoStar(#1,#3)

- QUERY:FindActors(_Actor) <= Actor(_Actor)
- QUERY:FindMovies(_Actor) <= Movie(UID IN Appearance(Who IN Actor(_Actor){UID}){What})
- QUERY:FindCostars(_Actor) <= Actor(UID IN CoStar(Who IN Actor(_Actor){UID}){WhoElse})
- QUERY:FindAll(_Actor) <= Actor(_Actor),Movie(UID IN Appearance(Who IN _1{UID}){What}),Actor(UID IN CoStar(Who IN _1{UID}){WhoElse})
What’s next?

- Version 3.8.6 of the HPCC Systems platform is out!

- Ongoing R&D (3.10, 4.0 and beyond):
  - Level 3 BLAS support
  - More Machine Learning related algorithms
  - Heterogeneous/hybrid computing (FPGA, memory computing, GPU)
  - Knowledge Engineering Language driving ML
  - General usability enhancements (GUI to SALT data profiling and linking, etc.)
  - Integration with other third party systems (R, for example)
Useful links

- LexisNexis Open Source HPCC Systems Platform: http://hpccsystems.com
- Machine Learning portal: http://hpccsystems.com/ml
- The HPCC Systems blog: http://hpccsystems.com/blog
- Our GitHub portal: https://github.com/hpcc-systems
- Community Forums: http://hpccsystems.com/bb
Trivia!

Answer the question and win a prize!

• What is the name of the data refinery engine that provides batch oriented data manipulation?
• What is the name of the data delivery engine that provides real-time analytics?
• What is the query processing language supporting the HPCC Systems platform?
• What can be used to generate ECL code for automating data profiling, parsing and cleansing?
• Name three machine learning algorithms supported in ECL-ML.
• What does KEL stand for?
Questions???