Introduction to HPCC Systems®

*Powered by RELX Group*

Flavio Villanustre
VP, Technology

October 20, 2015
Section One: Data Centric Approach

Data Centric Approach

Data Flow Oriented Big Data Technology

Data Disambiguation & Linking Technology

Data Graph Processing Technology

Real-time Enabled Technology

Visualization & Workflow Technology

Introduction to HPCC Systems
The Data Centric Approach

A single source of data is insufficient to overcome inaccuracies in the data.

The holes are inaccuracies found in the data.

Our platform is built on the premise of absorbing data from multiple data sources and transforming them to a highly intelligent social network graphs that can be manipulated to extract the non-obvious value.

The holes in the core data have been eliminated.
Our Solutions Are Powered by HPCC at Their Core

Big Data

- Public Records
- Proprietary Data
- News Articles
- Unstructured Records
- Structured Records

Unstructured and Structured Content

- Over 4 petabytes of content
- 50 billion records
- 10,000 sources
- 7.5 billion unique name and address combinations

High Performance Computing Cluster Platform (HPCC)

- Grid computing
- Data-centric language (ECL)
- Integrated delivery system that offers data plus analytics

Analysis Applications

- Entity Resolution
- Link Analysis
- Clustering Analysis
- Complex Analysis

Open Source Components

- Multi-bureau/multi-source models and bureau roll-over support
- Extensive experience leveraging atomic level data, combining and leveraging disparate data
- Approximately 400 models deployed (custom and flagship)

Key Capabilities

- Data and analytics
- Identity verification and authentication
- Fraud detection and prevention
- Investigation
- Screening
- Receivables management

Introduction to HPCC Systems
Section Two: Data Flow Oriented Big Data Technology

Data Centric Approach

Data Flow Oriented Big Data Technology

Data Disambiguation & Linking Technology

Data Graph Processing Technology

Real-time Enabled Technology

Visualization & Workflow Technology
Data Flow Oriented Big Data Platform

**Thor**
- Shared Nothing MPP Architecture
- Commodity Hardware
- Batch ETL and Analytics

**ROXIE**
- Shared Nothing MPP Architecture
- Commodity Hardware
- Real-time Indexed Based Query
- Low Latency, Highly Concurrent and Highly Redundant

---

**ESP**
- Middleware Services
- Portal
- Batch Subscribers
- Batch Processed Data

---

Raw data from several sources

Batch requests for scoring and analytics

---

**ECL**
- Easy to use
- Implicitly Parallel
- Compiles to C++
Thor – The Batch Processing Analytics Engine

Massively Parallel Extract Transform and Load (ETL) engine
- Built from the ground up as a parallel data environment.
- Leverages inexpensive locally attached storage.
- Doesn’t require a SAN infrastructure.

Enables data integration on a scale not previously available
- Current LexisNexis person data build process generates 350 billion intermediate results at peak.

Suitable for:
- Massive joins/merges
- Massive sorts and transformations
- Any N2 problem
ROXIE – The Real-Time Analytics Delivery Engine

A massively parallel, high throughput, structured query response engine.

Ultra fast due to its read-only nature.

Allows indices to be built onto data for efficient multi-user retrieval of data.

Suitable for:
- Volumes of structured queries
- Full text ranked Boolean search

- Raw data from several sources
- Batch reporting requests

Thor

ROXIE

ECL

Reporting

Batch Subscribers

Batch reporting requests
## Scalability Profiles – Thor vs. ROXIE

<table>
<thead>
<tr>
<th>Thor</th>
<th>ROXIE</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Scales to available memory</td>
<td>• Scales to available memory</td>
</tr>
<tr>
<td>• Spills to disk as needed</td>
<td>• Can run in a memory only model if required</td>
</tr>
<tr>
<td>• Optimal architecture minimizes disk spills</td>
<td></td>
</tr>
<tr>
<td>• CPU intensive</td>
<td>• Not typically CPU intensive, but complex processing can be performed as needed because of ECLs power</td>
</tr>
<tr>
<td>• Maximize available CPU cores by optimal slave node configuration</td>
<td></td>
</tr>
<tr>
<td>• Sequential reads and writes</td>
<td>• Random disk read</td>
</tr>
<tr>
<td>• Not heavily IO bound</td>
<td>• Heavily IO bound</td>
</tr>
<tr>
<td>• SAS drives are typical</td>
<td>• SAS drives are typical. SSD provide the best performance</td>
</tr>
<tr>
<td>• Disk spills occur as needed</td>
<td></td>
</tr>
<tr>
<td>• Network bandwidth matters only when processing jobs that exchange a lot of data between nodes</td>
<td>• Network bandwidth is required for fulfilling a number of requests as well as exchange of data between nodes</td>
</tr>
</tbody>
</table>
ECL – The Data Flow Oriented Programming Language

- An easy to use, data-centric programming language optimized for large-scale data management and query processing
- **Highly efficient** — automatically distributes workload across all nodes
- Industry analysts: “80% more efficient than C++, Java and SQL — 1/3 reduction in programmer time to maintain/enhance existing applications”
- Benchmark against SQL (5 times more efficient) for code generation
- Automatic parallelization and synchronization of sequential algorithms for **parallel and distributed processing**
- Large library of built-in modules to handle common data manipulation tasks

**Declarative programming language ... powerful, extensible, implicitly parallel, maintainable, complete and homogeneous**
Component View of Capabilities

- Extraction & Loading
- Data Hygiene
- Data Profiling
- Standardization
- Linking & Clustering
- Modeling
- Batch Scoring
- Index Build
- Real-Time Query and Scoring

Raw data from several sources

Batch requests for scoring and analytics

Thor

ETL & Analytics System

ROXIE

Data Delivery System

Access Controls and Permissible Use Query

ESP: Middleware Services

Portal

Batch Subscribers

Batch Processed Data
Layer View of Platform Capabilities

Data Science Portal
- Dashboard Creator
- Data Composition Builder

Analytics Tools
- ETL
- In Memory Ad-Hoc BI
- Entity Disambiguation (SALT)
- Graph Analytics (KEL)
- Machine Learning

Common Data Centric Language
- ECL

MPP Platform
- Thor
- Delta Store (Cassandra)
- ROXIE

Introduction to HPCC Systems
Section Three: Data Disambiguation & Linking Technology

Data Centric Approach

Data Flow Oriented Big Data Technology

Data Disambiguation & Linking Technology

Data Graph Processing Technology

Real-time Enabled Technology

Visualization & Workflow Technology
An Introduction to SALT (Scalable Automated Linking Technology)

From disparate data, to clustering, to showing relationships
Data Disambiguation and Linking Technology

- A process that links together all records common to a single individual
- Assigns a unique identifier to those records
- Not rules based — Calculates a weight (the specificity) for each value of each field, relative to all of the other values for that field found within the data
- Includes fuzzy matching and other capabilities to help find the best matches

This linking technology (SALT) powers many of our products and solutions through:

- Linking
- Data Enhancement
- Clustering
- Search Processes
- Analytics

LexID: 123456
John Doe
SALT Increases Developer Productivity

- 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

Data Sources

1. Data Preparation Processes (ETL)
   - Profiling
   - Parsing
   - Cleansing
   - Normalization
   - Standardization

2. Record Linkage Process
   - Matching Weights & Threshold Computation
   - Blocking/Searching
   - Weight Assignment & Record Comparison
   - Record Match Decision
   - Add’l Data Ingest
   - Linking Iterations

42 Lines of SALT
3,980 Lines of ECL
482,410 Lines of C++
SALT’s Superior Linking Technology

SALT eliminates **FALSE NEGATIVES** using probabilistic learning

1. Flavio Villanustre, Atlanta
2. Javio Villanustre, Atlanta

**SALT**— the system has learnt that “Villanustre” is specific because the frequency of occurrence is small and there is only one present in Atlanta

**ERROR**— because the rules determine that “Flavio” and “Javio” are not the same

**MATCH** — the system has learnt that “Villanustre” is specific because the frequency of occurrence is small and there is only one present in Atlanta

**NO MATCH** — because the rules determine that “Flavio” and “Javio” are not the same

SALT eliminates **FALSE POSITIVES** using probabilistic learning

1. John Smith, Atlanta
2. John Smith, Atlanta

**ERROR**— because the rules determine that “John Smith” and the city for both the records match

**MATCH** — because the rules determine that “John Smith” and the city for both the records match

**NO MATCH** — the system has learnt that “John Smith” is not specific because the frequency of occurrence is large and there are many present in Atlanta

**MATCH** — because the rules determine that “John Smith” and the city for both the records match

**NO MATCH** — the system has learnt that “John Smith” is not specific because the frequency of occurrence is large and there are many present in Atlanta
### An Entity Disambiguation Example

Do these two records belong to the same person?

<table>
<thead>
<tr>
<th>Record</th>
<th>First Name</th>
<th>Last Name</th>
<th>Address</th>
<th>City</th>
<th>State</th>
<th>Zip</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>MARCIA</td>
<td>MARSUPIAL</td>
<td>6035 JONES ST</td>
<td>ARVADA</td>
<td>CO</td>
<td>80004</td>
</tr>
<tr>
<td>2</td>
<td>KAREN</td>
<td>KANGAROO</td>
<td>5865 W OHIO AVE</td>
<td>LAKEWOOD</td>
<td>CO</td>
<td>80226</td>
</tr>
</tbody>
</table>
An Entity Disambiguation Example

Among the thousands of data sources, LexID found a series of related records

<table>
<thead>
<tr>
<th></th>
<th>First</th>
<th>M.I.</th>
<th>Last</th>
<th>St #</th>
<th>dir</th>
<th>St Name</th>
<th>suffix</th>
<th>city</th>
<th>st</th>
<th>zip</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matched record 1</td>
<td>MARCIA</td>
<td>K</td>
<td>MARSUPIAL</td>
<td>6035</td>
<td>JONES</td>
<td>ST</td>
<td>ARVADA</td>
<td>CO</td>
<td>80004</td>
<td></td>
</tr>
<tr>
<td>Moved</td>
<td>MARCIA</td>
<td>K</td>
<td>MARSUPIAL</td>
<td>9170</td>
<td>W</td>
<td>14TH</td>
<td>AVE</td>
<td>LAKEWOOD</td>
<td>CO</td>
<td>80226</td>
</tr>
<tr>
<td>Changed Last Name</td>
<td>MARCIA</td>
<td>K</td>
<td>KANGAROO</td>
<td>9170</td>
<td>W</td>
<td>14TH</td>
<td>AVE</td>
<td>LAKEWOOD</td>
<td>CO</td>
<td>80226</td>
</tr>
<tr>
<td>Used Middle Initial as First Name</td>
<td>K</td>
<td>MARCIA</td>
<td>KANGAROO</td>
<td>9170</td>
<td>W</td>
<td>14TH</td>
<td>AVE</td>
<td>LAKEWOOD</td>
<td>CO</td>
<td>80226</td>
</tr>
<tr>
<td>Moved &amp; matched record 2</td>
<td>KAREN</td>
<td>MARCIA</td>
<td>KANGAROO</td>
<td>5865</td>
<td>W</td>
<td>OHIO</td>
<td>AVE</td>
<td>LAKEWOOD</td>
<td>CO</td>
<td>80226</td>
</tr>
</tbody>
</table>

Thus these two seemingly unrelated records were matched for the response analysis!

* Example is real scenario from actual campaign analysis with the names changed to protect the customers information.
Section Four: Data Graph Processing Technology

- Data Centric Approach
- Data Flow Technology
- Real-time Enabled Technology
- Visualization & Workflow Technology
- Data Graph Processing Technology
- Data Disambiguation & Linking Technology
- Oriented Big Data Technology
Graph Processing Technology

KEL — an abstraction for network/graph processing

- Declarative model: describe what things are, rather than how to execute
- High level: vertices and edges are first class citizens
- A single model to describe graphs and queries
- Leverages Thor for heavy lifting and ROXIE for real-time analytics
- Compiles into ECL (and ECL compiles into C++, which compiles into assembler)

<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 <strong>human design</strong></td>
<td>Thor/ROXIE split by <strong>system design</strong></td>
</tr>
<tr>
<td>Solid, reliable and mature</td>
<td>R&amp;D</td>
</tr>
</tbody>
</table>
KEL is a graph representation language used to represent:

1. **Entities**
   
   ```plaintext
   Person := ENTITY(FLAT(UID=id, STRING fname, STRING mname, 
   STRING lname, STRING sex, STRING bdate, INTEGER age, 
   INTEGER income, STRING address1, STRING address2, STRING city, STRING state));
   Vehicle := ENTITY(FLAT(UID=id, INTEGER myear, STRING make=name.make, STRING model=name.model, 
   STRING body=name.body, STRING vin, INTEGER msrp, STRING purchaseprice, INTEGER value));
   Claim := ENTITY(FLAT(UID=id, INTEGER amount, DATE claim_date, STRING totalled));
   ```

2. **Associations (or Relations)**
   
   ```plaintext
   Owns := ASSOCIATION(FLAT(Person owner, Vehicle veh));
   ClaimFor := ASSOCIATION(FLAT(Vehicle veh, Claim claim));
   ```

3. **Attributes**
   
   ```plaintext
   Claim => AgeMo := MONTHSBETWEEN (claim_date, TODAY());
   Person: => ClaimAmountCountLess100 := ClaimFor.claim (amount<=100, AgeMo<=12):Count;
   ```
Section Five: Real-time Enabled Technology

Data Centric Approach

- Data Flow Oriented Big Data Technology
- Data Disambiguation & Linking Technology
- Data Graph Processing Technology
- Real-time Enabled Technology

Visualization & Workflow Technology
Enhanced Architecture to Support IoT

- Real-time and/or stream based data collection
- Real-time rules
- Real-time alerts
- Historic trends and analytics
- Massively scalable platform
- Real-time search and business intelligence
ROXIE
• REST
• SOAP
• Websocket
• IPv6
• 6LoWPAN
• UDP
• uIP
• DTLS
• MQTT
• CoAP
• ROLL
• XMPP-IoT
• Mihini/M3DA

Cassandra
• AMQP
• DDS
• LLAP
• LWM2M
• SSI
• IOTDB
• SensorML
• IPSO
• Telehash
• TSMP
• NanoIP
• ONS 2.0

Thor

Adapter

memcached

Kafka

Index Updates

HPCC: Internet of Things Architecture
Section Six: Visualization & Workflow Technology

Data Centric Approach

Data Flow Oriented Big Data Technology

Data Disambiguation & Linking Technology

Data Graph Processing Technology

Real-time Enabled Technology

Visualization & Workflow Technology
Provide a Powerful Visual Tool for Managing Data Workflows and Visualization

- Build complex relationship graph easily using a drag and drop interface
- Gain insight into hidden relationships using powerful visualization techniques
A Plugin Architecture to Developing Data Workflows
Presenting the Side Effects Paradigm
Powerful Dashboard Creation for End Users

Data Science Portal

Browse Dashboards | Demo Flight Data

Arrival Delay By State

Arrival Delay By Carrier

Departure Delay By State

Departure Delay By Carrier

Copyright © 2015 LexisNexis
Network Graph
A natural way to solve hard big data problems — Representing data as a set of relationships

1. Clearly identify entities of interest
2. Accurately represent relationship between the entities
3. Perform queries using degrees of separation
4. Build attributes using filters based on relationships and degrees of separation
5. Develop accurate learning models based on a social footprint
### Example #1: Understanding People Relations Helps Us Predict Risk

<table>
<thead>
<tr>
<th>Count</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>16.5 B</td>
<td>Consumer records</td>
</tr>
<tr>
<td>8.9 B</td>
<td>Unique name/address combos</td>
</tr>
<tr>
<td>4.0 B</td>
<td>Property records</td>
</tr>
<tr>
<td>3.7 B</td>
<td>Motor vehicle registrations</td>
</tr>
<tr>
<td>417 M</td>
<td>Criminal records</td>
</tr>
<tr>
<td>269 M</td>
<td>Auto and home claim records</td>
</tr>
<tr>
<td>188.5 M</td>
<td>Unique cell phones</td>
</tr>
<tr>
<td>37 M</td>
<td>Unique businesses</td>
</tr>
</tbody>
</table>

- Collect largest, broadest, deepest, most accurate, up-to-date repository of public record and contributory data
- Clean and standardize the data
- Identify unique entities using sophisticated learning techniques
- Create the social relationships
Example #2: Network Intrusion Detection by Understanding the Relationships Between IP Addresses, Country, Botnets, etc.

- Use millions of application transaction log records per day
- Combine information from network routers, DNS servers, etc.
- Use third party sources to identify blacklisted information
- Apply learning techniques to match the digital footprint to identify the virtual identities
- Prevent intrusions by detecting anomalies
Real World Examples on Using the Social Graph to Solve Problems

1. Ability to query the graph in real time is critical
2. Attribute calculation at various degrees of separation helps to identify non obvious clusters
Example #1: Graph Analysis
Bust-Out Fraud

SCENARIO

• Perpetrators typically apply for credit 4 to 24 months before busting out

• Hard to predict at an individual level

Apply for Credit  Build good credit history  Obtain additional credit  Draw down all available credit  Disappear

More than $2 billion in annual losses
Example #1: Graph Analysis
The Solution Highlights

SCENARIO

- Financial company hypothesized that organized groups were targeting them and desired to tackle bust-out fraud at a social level
- Although some risk was not individually large, the company wanted to ascertain where that risk was growing without them realizing the connections
Example #1: Graph Analysis
The Solution Highlights

DATA & ANALYSIS

• 5 million accounts flagged with active, known fraud, charge offs and preemptively closed tags
• We investigated every address, asset and business connection attached to the accounts

OUR APPROACH

• Standardized input, applied LexID and joined to our Social Graph (5 million into 4 billion relationships)
• Calculated grouped social aggregates, scored and rank ordered the result
### Example #1: Graph Analysis

#### Results Overview

<table>
<thead>
<tr>
<th>Known Fraud</th>
<th>Charge Off</th>
<th>Preemptively Closed</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong># in 1 degree</strong></td>
<td><strong># in 2 degrees</strong></td>
<td><strong># in 1 degree</strong></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td><strong>3</strong></td>
<td><strong>2</strong></td>
<td><strong>212</strong></td>
</tr>
<tr>
<td><strong>31 associated with 3 known fraud accounts</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Known Fraud

- 31 accounts associated with 3 known fraud accounts

#### Charge Off

- 212 accounts associated with 2 charge accounts
- 2 charge accounts

#### Preemptively Closed

- 3 accounts associated with 15 preemptively closed accounts
- 15 preemptively closed accounts
Example #1: Graph Analysis

Aggregated group behavior variables for a single account

- Flatten the graph
- Calculate aggregate group behavior measurements
- Drive predictive analytics at a granular level using graph variables

LexID: 312873
Jack Johnson
Cluster ID: 1214379

<table>
<thead>
<tr>
<th># Network Neighborhood</th>
<th>81</th>
</tr>
</thead>
<tbody>
<tr>
<td># 1st Degrees</td>
<td>8</td>
</tr>
<tr>
<td># 2nd Degrees</td>
<td>72</td>
</tr>
<tr>
<td>Cohesivity</td>
<td>1.67</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th># Active Accounts</th>
<th>Total</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Degree 0</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Degree 1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Degree 2</td>
<td></td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th># Fraud Accounts</th>
<th>&lt;=2 Degrees</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Degree 0</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>Degree 1</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>Degree 2</td>
<td></td>
<td>6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th># Charge Off Accounts</th>
<th>&lt;=2 Degrees</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Degree 0</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>Degree 1</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>Degree 2</td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th># Preemptive Close Accounts</th>
<th>&lt;=2 Degrees</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Degree 0</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>Degree 1</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>Degree 2</td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

**SCORE** 6.6

Small network

Fairly tightly connected

3 active accounts within the network

9 accounts within that network that are known fraud

1 account within the network that’s been preemptively closed
Example #2: Graph Analysis
Insurance Collusion

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
Example #2: Graph Analysis
Insurance Collusion

TASK
• After adding the Link ID to the carrier data, our HPCC technology then added 2 additional degrees of relatives

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
Questions?

- Open Source HPCC Systems Platform: [http://hpccsystems.com](http://hpccsystems.com)
- Free Online Training: [http://learn.lexisnexis.com/hpcc](http://learn.lexisnexis.com/hpcc)
- SALT: [http://hpccsystems.com/salt](http://hpccsystems.com/salt)
- KEL: [https://hpccsystems.com/download/free-modules/kel-lite](https://hpccsystems.com/download/free-modules/kel-lite)
- Machine Learning portal: [http://hpccsystems.com/ml](http://hpccsystems.com/ml)
- The HPCC Systems blog: [http://hpccsystems.com/blog](http://hpccsystems.com/blog)
- Community Forums: [http://hpccsystems.com/bb](http://hpccsystems.com/bb)
- Source Code: [https://github.com/hpcc-systems](https://github.com/hpcc-systems)
- WIKI: [https://wiki.hpccsystems.com](https://wiki.hpccsystems.com)