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.
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

Historical Property Sales Indicators and Counts

Derived Public Data Relationships from +/- 50 terabyte database

Large Scale Suspicious Cluster Ranking

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.
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.
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
**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.

*Interactive Visualization uses calculated page rank to define size and color of the nodes.*
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.

Wikipedia Hourly Fingerprint

1 Steve_Jobs
2 Whitney_Huston
3 Wikipedia:SOPA_initiative/Learn_more
4 List_of_HTTP_status_codes%231xx_Informational
5 Adele_(singer)
6 Bruno_Mars
7 Kim_Jong-il
8 Jeremy_Lin
9 Heavy_D
10 Christopher_Columbus
11 Murder_of_Meredith_Kercher
12 Eli_Manning
13 Jorge_Luis_Borges
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 Open Source 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 end-to-end Big Data workflow management, scheduler, integration tools, etc.
- Originally designed circa 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
   - Programmable using ECL

2. HPCC Data Delivery Engine (Roxie)
   - A massively parallel, high throughput, query engine
   - Low latency, highly concurrent and highly available
   - Allows multipart (compound) indices for efficient retrieval
   - 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

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

**Powerful:** High level data primitives as JOIN, TRANSFORM, PROJECT, SORT, DISTRIBUTE, MAP, etc. are available.

**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, without side effects and with efficient encapsulation, programs are more succinct, reliable and easier to troubleshoot

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

**Homogeneous:** One language to express data algorithms across the entire HPCC platform, data integration, analytics and high speed delivery
Enterprise Control Language (ECL)

- The ECL optimizer generates an optimal execution plan
- Subgraphs run activities in parallel, automatically persist the results and spill to disk as needed
- Lazy evaluation avoids re-computing results when data and code haven’t changed
- Graphs are dynamic and display the number of records traversing the edges and data skew percentages
Machine Learning on HPCC

- Extensible Machine Learning Library developed in ECL
- Fully distributed across the cluster
- Supports PB-BLAS (Parallel Block BLAS)
- 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, for numeric optimization and clustering)
- Open Source and available at: http://hpccsystems.com/ml
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
ECL-ML: extensible ML on HPCC (ii)

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
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!
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(Wo IN Actor(_Actor){UID}){What})
- QUERY:FindCostars(_Actor) <= Actor(UID IN CoStar(Wo IN Actor(_Actor){UID}){WhoElse})
- QUERY:FindAll(_Actor) <= Actor(_Actor),Movie(UID IN Appearance(Wo IN _1{UID}){What}),Actor(UID IN CoStar(Wo IN _1{UID}){WhoElse})
Assorted examples

- Data integration using HPCC Flow
- Record linkage, entity disambiguation, entity resolution using SALT
- Data Visualization
- Exploratory data analysis
- Recommendation systems
- Search and retrieval
- Full text classifiers
- Scoring/statistical models
- Attribute manager
- Sentiment Analysis example
- Graph traversal problem handling (IMDB/Kevin Bacon example)
- Smart View
- SQL/JDBC integration
- Data Streaming/Kafka
- R integration
- Java integration
- Python integration
Data Integration using HPCC Flow

Drag and drop interface to data workflows on HPCC
Generates ECL code so it can be used also as a learning tool

Simple workflow, with the ECL Primitives shown on Design
Data Integration using HPCC Flow

A parent job calling a child job
Data Integration using HPCC Flow

Get the source code: https://github.com/hpcc-systems/spoon-plugins

Child job called by a parent job
Record linkage, entity disambiguation, entity resolution
Scalable Automated Linking Technology (SALT)

- The acronym stands for “Scalable Automated Linking Technology”
- Template based ECL code generator
- Provides for automated data profiling, QA/QC, parsing, cleansing, normalization and standardization
- Sophisticated specificity based linking and clustering
- Significant productivity boost!
Rules based vs. probabilistic record linkage

Thoroughly employing statistical algorithms, SALT has the two significant advantages over rules:

1. SALT sufficiently explore all credible matches

   **INPUT**
   - Flavio Villanustre, Atlanta  
   - Javio Villanustre, Atlanta

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

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

2. SALT effectively minimizes false matches

   **INPUT**
   - John Smith, Atlanta
   - John Smith, Atlanta

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

   **RULES**
   - MATCH, because the rules determine that “John Smith” and the city for both the records match
Another purpose of SALT is to perform record linking.
Data Visualization
Access a Roxie query to display a dynamic data dashboard from within Excel:

Function refreshdata()
    Application.EnableEvents = False
    Call [Turnover].QueryTable.Refresh
    Application.EnableEvents = True
End Function

The Excel Visual Basic code to retrieve data:
Data Visualization: Excel Integration...

Selecting the published Roxie Query in Excel:
Copy the HPCC JDBC Driver in directory `\report-designer\lib\jdbc` and specify the following values:

**Connection Type:** Generic Database

**Custom Connection URL:**
```
jdbc:hpcc:ServerAddress=99.999.999.9;Cluster=default:WsECL
DirectPort=8008;EclResultLimit=100:QuerySet=thor:LazyLoad=true:PageSize=100:TraceLevel=ALL:TraceToFile=true:
```

**Custom Driver Class Name:**
```
org.hpccsystems.jdbcdriver.HPCCDriver
```

Reports/Charts can be created by writing a custom query or by calling a Roxie query.

**Adhoc SQL Query:**
```
select quarter, store_sales, unit_sales, store_cost, the_year, 
month_of_year 
from foodmart::agg_c_10_sales_fact_1997 where quarter = ${pQtr}
```

**Roxie Query:**
call roxie::all_cancers_for_state(GA)
Data Visualization: Pentaho Report Designer

Data Visualization: Mondrian using Saiku UI

Real-time graphical online Analytical Processing (OLAP) on HPCC
Users explore business data by drilling into and cross-tabulating information with speed-of-thought response times to complex analytical queries.

Download: http://meteorite.bi/saiku
Data Visualization: BIRT

Eclipse-based open source reporting system with HPCC

BIRT Data Source Types:
- JDBC (for AdHoc queries or Roxie queries)
- Web Service (Roxie queries)
Data Visualization: BIRT Example Output

Download: http://download.eclipse.org/birt/downloads/
Data Visualization: Result Cell Formatting

Visualizations can be used to:
- Understand new data
- Demonstrate data/products
- Summarize data.

In the ECL Watch Tech Preview, users can specify specific output columns to contain “HTML” or “JavaScript”, rather than the default plain text. See example ECL code snippet below.

This allows the user to perform some “Cell Formatting” of the results, such as:
- Add textual formatting, e.g.: `<b>Text</b>` (bold text)
- Cell color: Make a cell Red/Yellow to alert the viewer that certain rows are important.
- Embed Hyperlinks: Embed a link to a preformatted Google search, or a link to a Google map with lat. and long. etc.

```ecl
import SampleData.Cars;
import Bundles.d3;

r := record
    dataset(Cars.CarsRecord) cars;
    varstring pc__javascript;
end;

d3Chart := d3.Chart('cars', 'name', 'unused');
d := dataset([[Cars.CarsDataset, d3Chart.ParallelCoordinates]], r);
d;
```

Name the column with a trailing “__html” or “__javascript”
Data Visualization: Result Cell Formatting Output
Data Visualization: Result Cell Formatting – Basic Chart
Data Visualization: Result Cell Formatting – Graphs
Exploratory Data Analysis
Exploratory Data Analysis

The EDA tool set will be tools added to HPCC Flow that allow analytics on big data. The EDA tools will not provide all the functionality of existing tools so an additional benefit is the ability to reduce a “Big Data” problem down to a statistically significant subset of random records to be used in existing analytic tools such as R and SAS.

<table>
<thead>
<tr>
<th>EDA Procedures</th>
<th>EDA Visualization</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Frequency</td>
<td>• Scatter Plots</td>
</tr>
<tr>
<td>• Mean, Median, Mode, Standard Deviation, Percentiles, Min, Max (Essentially Proc Univariate)</td>
<td>• Line Plots</td>
</tr>
<tr>
<td>• Rank descending and ascending</td>
<td>• Histograms</td>
</tr>
<tr>
<td>• Tabulate procedure</td>
<td>• Pie charts</td>
</tr>
<tr>
<td>• Random number generators</td>
<td>• GUI will allow user to define which data is plotted against the X and Y axis</td>
</tr>
<tr>
<td>• Simple math ops such as x/y for each observation</td>
<td>• Graph options will include Line, Bar, Pie and Histogram</td>
</tr>
<tr>
<td>• Options to easily add columns of data created by doing math on one or more existing columns of data</td>
<td></td>
</tr>
<tr>
<td>• Correlation statistics between two variables</td>
<td></td>
</tr>
</tbody>
</table>
Sentiment Analysis
HPCC Systems and its ECL-ML library includes an extensible set of fully parallel Machine Learning (ML) and Matrix processing algorithms to assist with business intelligence; covering supervised and unsupervised learning, document and text analysis, statistics and probabilities, and general inductive inference related problems.

1. Download the ML Library
   http://hpccsystems.com/ml

2. Extract the contents of the zip file to the ECL IDE source folder.

3. Reference the library in your ECL source using a import statement as shown in the example:

   ```eclipse
   IMPORT * FROM ML;
   IMPORT * FROM ML.Cluster;
   IMPORT * FROM ML.Types;

   x2 := DATASET([ {1, 1, 1}, {1, 2, 5}, {2, 1, 5}, {2, 2, 7},
                   {3, 1, 8}, {3, 2, 1}, {4, 1, 0}, {4, 2, 0}, {5, 1, 9}, {5, 2, 3},
                   {6, 1, 1}, {6, 2, 4}, {7, 1, 9}, {7, 2, 4} ], NumericField);

   c := DATASET([ {1, 1, 1}, {1, 2, 5}, {2, 1, 5}, {2, 2, 7},
                  {3, 1, 9}, {3, 2, 4} ], NumericField);

   x3 := Kmeans(x2,c);

   OUTPUT(x3);
   ```
ML.Classify tackles the problem: “given I know these facts about an object; can I predict some other value or attribute of that object.” For example, can I predict whether sentiment of a tweet is positive or negative?

Using a classifier in ML involves three logical steps:

1. Learning the model from a training set of data that has been classified externally.
2. Testing. Getting measures of how well the classifier fits.
3. Classifying. Apply the classifier to new data in order to give it a classification

```ecl
IMPORT ML;

/* Use ML.Docs module to pre-process tweet text and generate classification training set: IndependentDS, ClassDS

dRaw  - collection of positive and negative sentiment tweets
dLexicon := ML.Docs.Tokenize.Lexicon(dTokens);
ML.Docs.Trans(ML.Docs.Tokenize.ToO(dTokens,dLexicon)).WordsCounted;
*/
BayesModule := ML.Classify.NaiveBayes;
// Learning the model
Model := BayesModule.LearnD(IndependentDS, ClassDS);
// Testing
TestModule := BayesModule.TestD(IndependentDS, ClassDS);
OUTPUT(TestModule);
//Classifying
Results := BayesModule.ClassifyD(IndependentDS, Model);
OUTPUT(Results);
```
Sentilyze uses HPCC Systems and its ML-Library to classify tweets with positive or negative sentiment.

Example code of a dataset of tweets that are classified using both Keyword Count and Naïve Bayes sentiment classifiers:

```ecl
IMPORT Sentilyze;
Tweets := DATASET('~SENTILYZE::TWEETS',Sentilyze.Types.TweetType.CSV);
rawTweets := Sentilyze.PreProcess.ConvertToRaw(Tweets);
processTweets := Sentilyze.PreProcess.RemoveAnalysis(rawTweets);
kcSentiment := Sentilyze.KeywordCount.Classify(processTweets);
nbSentiment := Sentilyze.NaiveBayes.Classify(processTweets);
OUTPUT(kcSentiment,NAMED('TwitterSentiment_KeywordCount'));
OUTPUT(nbSentiment,NAMED('TwitterSentiment_NaiveBayes'));
```
Graph Traversal Problem Handling
Graph Traversal Problem Handling

Data analysis techniques using HPCC Systems ECL can be applied to social graph traversal problems for finding customer insight and detection of important data patterns and trends.

Get the example code:
http://hpccsystems.com/download/docs/six-degrees

---

/* *********************************************************/
ATTRIBUTES:
Produce a series of sets for Actors and Movies that are:
- distance-0 away (KBacon's Direct movies),
- distance-1 away KBacon's CoStars Movies,
- distance-2 away - Movies of Costars of Costars etc

The nested attributes below are shown here together for the benefit of the reader.

Notes on variable naming convention used for costars and movies

KBMovies : Movies Kevin Bacon worked in (distance 0)
KBCoStarMovies : Movies worked in by KBCoStars (distance 1)
except KBMovies (distance 1)
KBCo2Stars : Stars(Actors) who worked in KBCoStarMovies (distance 2)
except KBCoStarMovies (distance 2)
KBCo2StarMovies : Movies worked in by KBCo2Stars (distance 2)
except KBCo2StarMovies (distance 2)
KBCo3Stars : Stars(Actors) who worked in KBCo2StarMovies (distance 3)
except KBCo2StarMovies (distance 3)

etc..

*************************************************************/

IMPORT Std;
IMPORT IMDB;

EXPORT KevinBaconNumberSets := MODULE
// Constructing a proper name match function is an art within itself
// For simplicity we will define a name as matching if both first and last name
// are found within the string
NameMatch(string full_name, string fname,string lname) :=
Std.Str.Pind(full_name,fname,1)>0 AND
Std.Str.Pind(full_name,lname,1)>0;

----- Get KBacon Movies
AllKBMovies := IMDB.ActorsInMovies(NameMatch(actor,'Kevin','Bacon'));
EXPORT KBMovies := DEGUP(AllKBMovies, movie, ALL); // Each movie should ONLY occur once

----- Get KBacon CoStars
KBCoStars := IMDB.ActorsInMovies(Movie IN SET(KBMovies,Movie));
EXPORT KBCoStarMovies := DEGUP( KBCoStars[actor <> 'Kevin Bacon'], actor, ALL);
Smart View™ (SV) generates comprehensive profiles of ‘entities’ – people, places, and organizations – from a ‘perfect storm’ of big data (ie, disparate, complex and massive datasets requiring fusion for time and mission-critical applications). The combination of SV’s intuitive user interface, along with its leveraging of a patented, statistically-based clustering technique, make it a compelling option for the data scientist or business analyst faced with complex entity resolution challenges.

This Smart View screenshot contains a visualization depicting how 12 individual records were fused together across 4 iterations – ultimately formulating an in-depth profile of the identity being analyzed.
The more comprehensive your entity profiles are, the greater the probability that non-obvious linkages between them will be uncovered. Smart View’s link analysis feature allows the end-user to define a series of relationships, then uncover such relationships through an iterative, statistical process (similar to how SV’s entity resolution problem is solved). To aide the data scientist in interpreting what their link analysis results signify, Smart View includes a set of relationship visualizations that display dynamically according to ‘point’ and ‘click’ commands.

How many individuals are linked to Kevin Bacon through 3 degrees of separation? For those actors with one degree of separation, how ‘strong’ are the linkages? These are just a sample of the many questions SV’s link analysis feature can help users address (for Kevin Bacon, or any other entity for that matter 😊)
SQL/ JDBC Integration
SQL/JDBC Integration

The HPCC JDBC Driver provides SQL-based access to HPCC Systems data files and published queries.

This allows you to connect to the HPCC Systems platform through your favorite JDBC client and access your data without the need to write a single line of ECL!

- Analyze HPCC data using many JDBC Client User Interfaces
- Supports simple SQL SELECT or CALL syntax
  - Access HPCC data files as DB Tables
  - Access published queries as DB Stored Procedures
- Harnesses the full power of HPCC under the covers
  - Submitted SQL request generates ECL code which is submitted, compiled, and executed on your target cluster
  - Automatic Index fetching capabilities for quicker data fetches
- Creates entry-point for programmatic data access
- Leverage HPCC data and JDBC client functionality without need to learn and write ECL!
  - Opens the door for non ECL programmers to access HPCC data.
public class HpccConnect {
    public static void main(String[] args) {
        Connection con = null;
        try {
            //1. Initialize HPCC Connection
            Class.forName("org.hpccsystems.jdbcdriver.HPCCDriver");

            //2. Write the Query to select foodMart::currency table
            PreparedStatement pst = con.prepareStatement("select conversion_ratio,currency_id,currency from foodmart::currency");
            pst.clearParameters();
            ResultSet rs = pst.executeQuery();

            //3. Print the headers to the console
            System.out.println("conversion_ratio		currency_id		currency");

            //4. Loop through all the records and write the information to the console
            while (rs.next()) {
                System.out.println(rs.getObject(1) + "	" + rs.getString(2) + "	" + rs.getString(3));
            }

            //5. Close the connection
            con.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
Data Streaming
Apache Kafka is a distributed publish-subscribe messaging system. It is designed to support the following:

- Persistent messaging with $O(1)$ disk structures that provide constant time performance even with many TB of stored messages.
- High-throughput: even with very modest hardware Kafka can support hundreds of thousands of messages per second.
- Explicit support for partitioning messages over Kafka servers and distributing consumption over a cluster of consumer machines while maintaining per-partition ordering semantics.
- Horizontally Scalable
Data Streaming: How we are using Apache Kafka
rHPCC is an R package providing an interface between R and HPCC Systems.

# Creates an ECL "PROJECT" definition.
# The PROJECT function processes through all records in the recordset performing
# the transform function on each record in turn.
# Example ECL CODE: PROJECT(recordset,TRANSFORM(record,SELF := LEFT));

ECLProject <- setRefClass("ECLProject", contains="ECLDataset", fields = list(name = "character", inDataset="ECLDataset", outECLRecord="ECLRecord", def = "character"),
methods = list(
    getName  = function() {
        result <- name
    },

    addField  = function(id, value) {
        def <<- paste(def, " ", id, " := ", value, ";")
    },

    print = function() {
        result <- paste (name, " := PROJECT( ", inDataset$getName(), ", TRANSFORM(", outECLRecord$getName() , ",",def, " ));")
    })
)

rHPCC is an R package providing an interface between R and HPCC Systems.
ECL Code Snippet using EMBED(R)

IMPORT R;

EXAMPLE 1:

SET OF INTEGER2 sortArray ( SET OF INTEGER2 val) := EMBED(R)
  sort(val);  // R code here
ENDEMBED;

arr := sortArray([-111, 0, 113, 12, 45, 5]);
arr[3];  // Returns 5

EXAMPLE 2:

STRING cat(VARSTRING what, VARSTRING who) := EMBED(R)
  paste(what, who)  // R code here
ENDEMBED;

result := cat('Hello', 'World');
result;  // Hello World
Java Integration
Java Integration: Java ECL API

Java ECL API was developed along side the HPCC Flow tool building the necessary JAVA code base to handle generating the ECL code, calling the Cluster via the SOAP interface, and retrieving results and server status updates. A secondary goal to this project is to provide examples on using the HPCC Systems Platform from another programming language in general.

Java ECL API Example with no client side compile check

```java
String eclCode = "output('Hello World');";
//getECLSoap() is a class used that sets up the server connection and returns a ECLSoap class
ECLSoap es = getECLSoap();
boolean isValid = false;
ArrayList dsList = null;
String outStr = "";

isValid = es.executeECL(eclCode);
String wuid = es.getWuid();
if(!isValid){
    this.error += "\r\nFailed to execute code on the cluster, please verify your settings\r\n";
}

To check the ECL code with a local compiler call:

boolean isError = false;
boolean isWarning = false;
String errorMessage = (es.syntaxCheck(eclCode)).trim();
if(es.getErrorCount() > 0){
    isError = true;
}
if(es.getWarningCount() > 0){
    isWarning = true;
}
```

Compile and submit to cluster
Java Integration: Java ECL API – Code Generators

<table>
<thead>
<tr>
<th>ECL Primitive</th>
<th>SALT</th>
<th>ML Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dataset</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dedup</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DeSpray</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Distribute</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FromField</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Group</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Index</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Iterate</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Join</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Limit</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Loop</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Merge</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Output</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Project</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rollup</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sort</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Spray</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Table</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ToField</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Profile Report</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Concepts</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specificities</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hygiene Report</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Internal Linking</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Associate</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Build Classify</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Classification</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Classify</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Discretize</td>
<td></td>
<td></td>
</tr>
<tr>
<td>KMeanes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Naïve Bayes</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Render ECL Example

```java
Join join = new Join();
join.setName("myRecSet");
join.setJoinCondition("left.theID" + "right.theID");
join.setJoinType("INNER");
join.setLeftRecordSet("MyLeftRecordSet");
join.setRightRecordSet("MyRightRecordSet");
String joinResults = join.ecl();
```

Resulting ECL Code

```ecl
myRecSet := join(MyLeftRecordSet,MyRightRecordSet,left.theID = right.theID, INNER);
```
Java Integration: Java ECL API – SALT Code Generator

Data Profile Example
Data Profile Example Result Set

Try the demo: http://hpccsystems.com/demos/data-profiling-demo
Get the source code: https://github.com/hpcc-systems/java-ecl-api
Python Integration
ECL Code Snippet using EMBED(Python)

IMPORT Python;

EXAMPLE 1:

SET OF INTEGER sortArrayOfIntegers (SET OF INTEGER val) := EMBED(Python)
  return sorted (val)  // Python code here
ENDEMBED;

arr := sortArray([-111, 0, 113, 12, 45, 5]);
arr[3];  // Returns 5

EXAMPLE 2:

SET OF STRING sortArrayOfString (SET OF STRING val) := EMBED(Python)
  return sorted (val)  // Python code here
ENDEMBED;

result := sortArrayOfString(['red','green','yellow']);
result[1];  // Returns ‘green’
What’s next?

- Version 5.0 Beta will be out soon.

- Ongoing R&D (5.2 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 (new abstractions, enhanced user interfaces)
  - Additional capabilities for embedded languages
  - More integration with other tools and systems
Useful links

- LexisNexis Open Source HPCC Systems Platform: [http://hpccsystems.com](http://hpccsystems.com)
- Free Online Training: [http://learn.lexisnexis.com/hpcc](http://learn.lexisnexis.com/hpcc)
- 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)
- Our GitHub portal: [https://github.com/hpcc-systems](https://github.com/hpcc-systems)
- JIRA: [https://track.hpccsystems.com](https://track.hpccsystems.com)

Upcoming Event!

Join us during Big Data Atlanta Week for a Meetup!

EDA Toolkit for Data Scientists
May 6, 2014 at 4:30pm
LexisNexis Alpharetta office,
1000 Alderman Drive

RSVP:
[http://www.meetup.com/Atlanta-Big-Data-Week-2014/events/169591702/](http://www.meetup.com/Atlanta-Big-Data-Week-2014/events/169591702/)
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???
Terasort Benchmark results

Execution Time (seconds)

Productivity

Space/Cost

3 Lines of ECL

700+ Lines of Java MapReduce Code
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 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](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)**