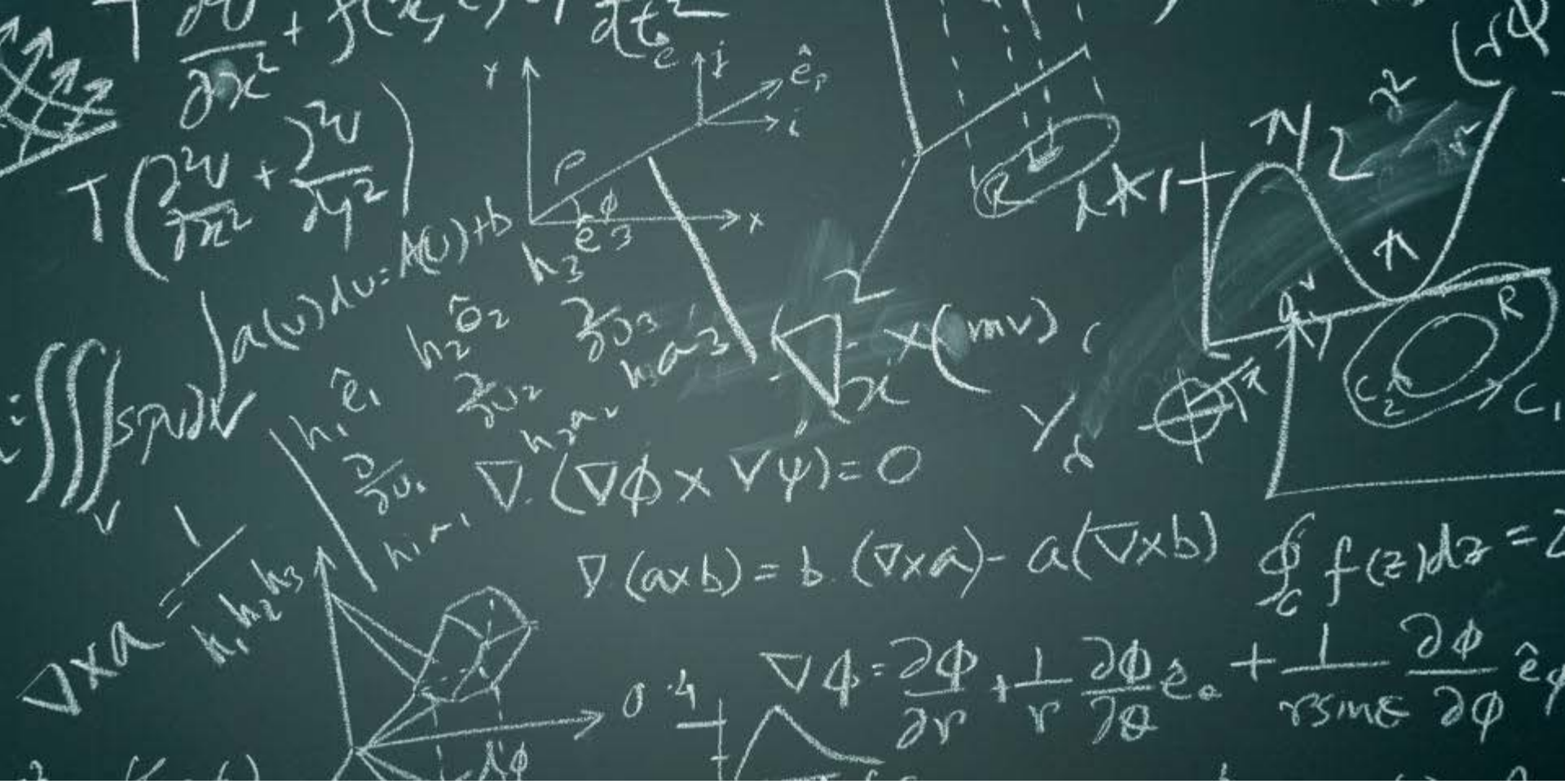


Distributed Machine Learning with the HPCC Systems Platform

Presenter: Dr. Flavio Villanustre

Lead of the HPCC Systems Initiative

LexisNexis®

HPCC Systems

http://hpccsystems.com

Risk Solutions

Welcome

KSU Seminar

Distributed Machine Learning with the HPCC Systems Platform

Dr. Flavio Villanustre, VP Technology & Product - LexisNexis

Agenda

11:00am – 11:15am: Welcome

11:15am – 12:30pm: Presentation

12:30pm – 1:00pm: Q&A / Open discussion, Trivia & Raffle

Twitter event hashtag:

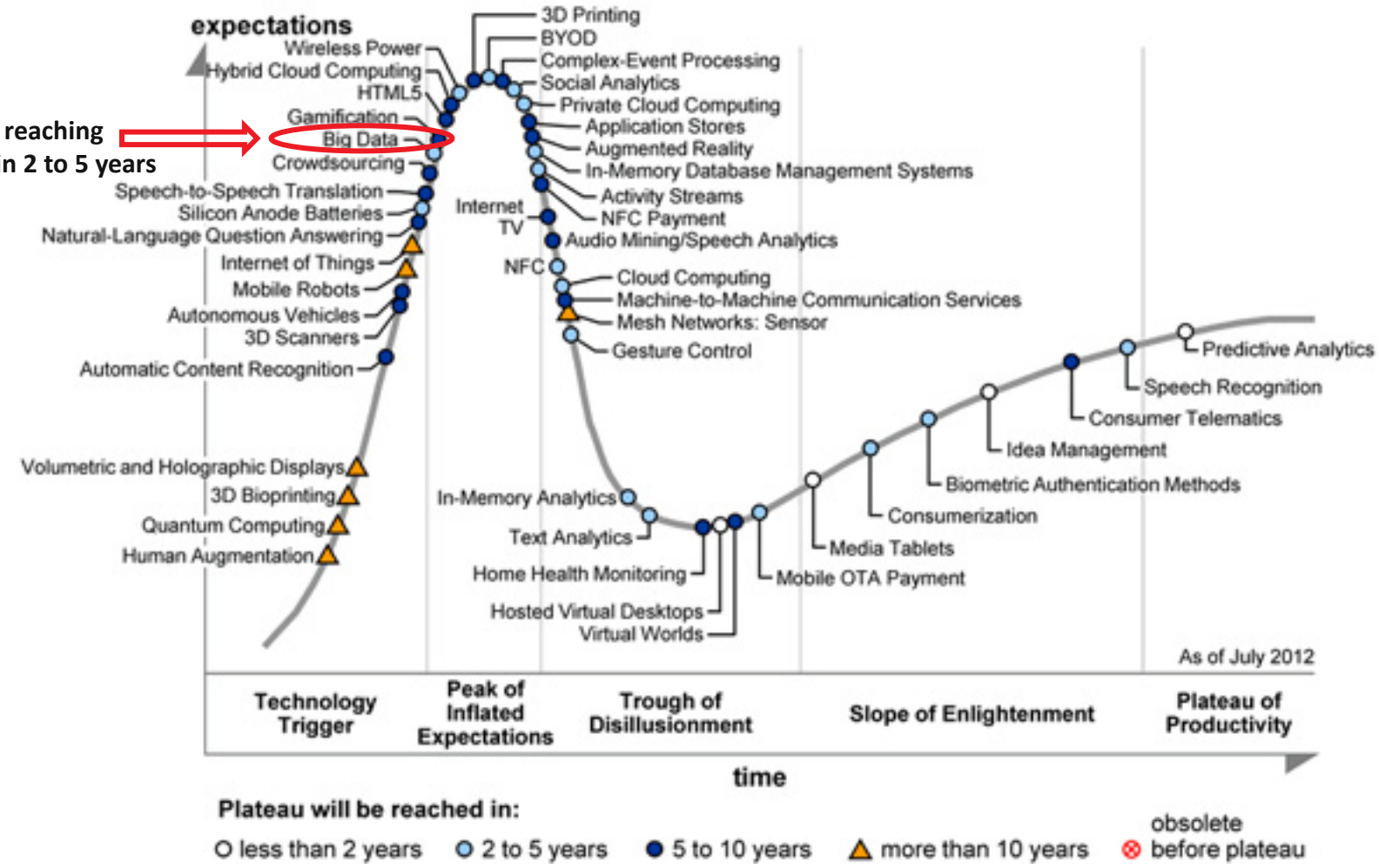
#hpccmeetup

hpccsystems.com

Big Data in Gartner's Hype Cycle

HPCC Systems
http://hpccsystems.com

Big Data reaching Plateau in 2 to 5 years



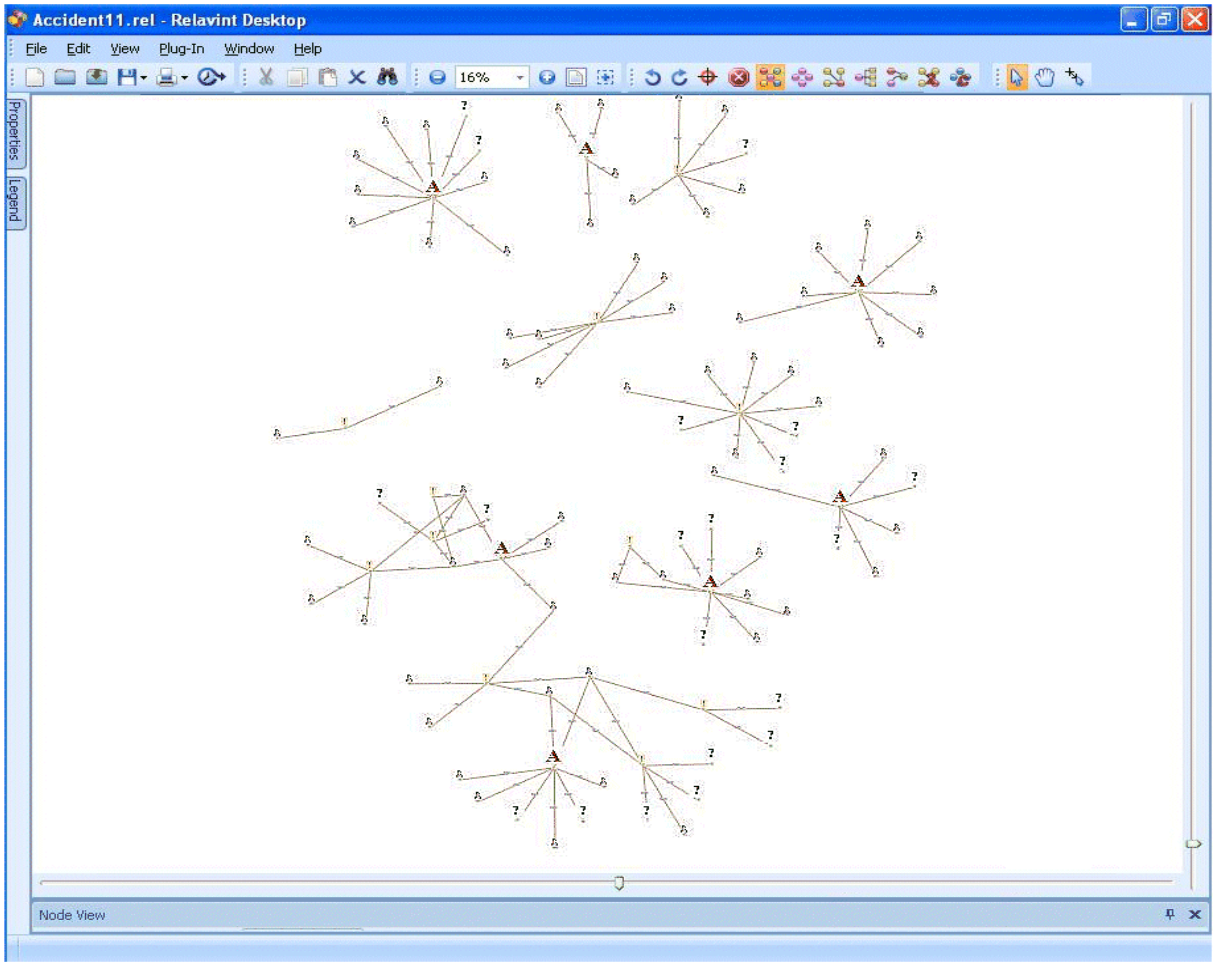
Source: Gartner

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

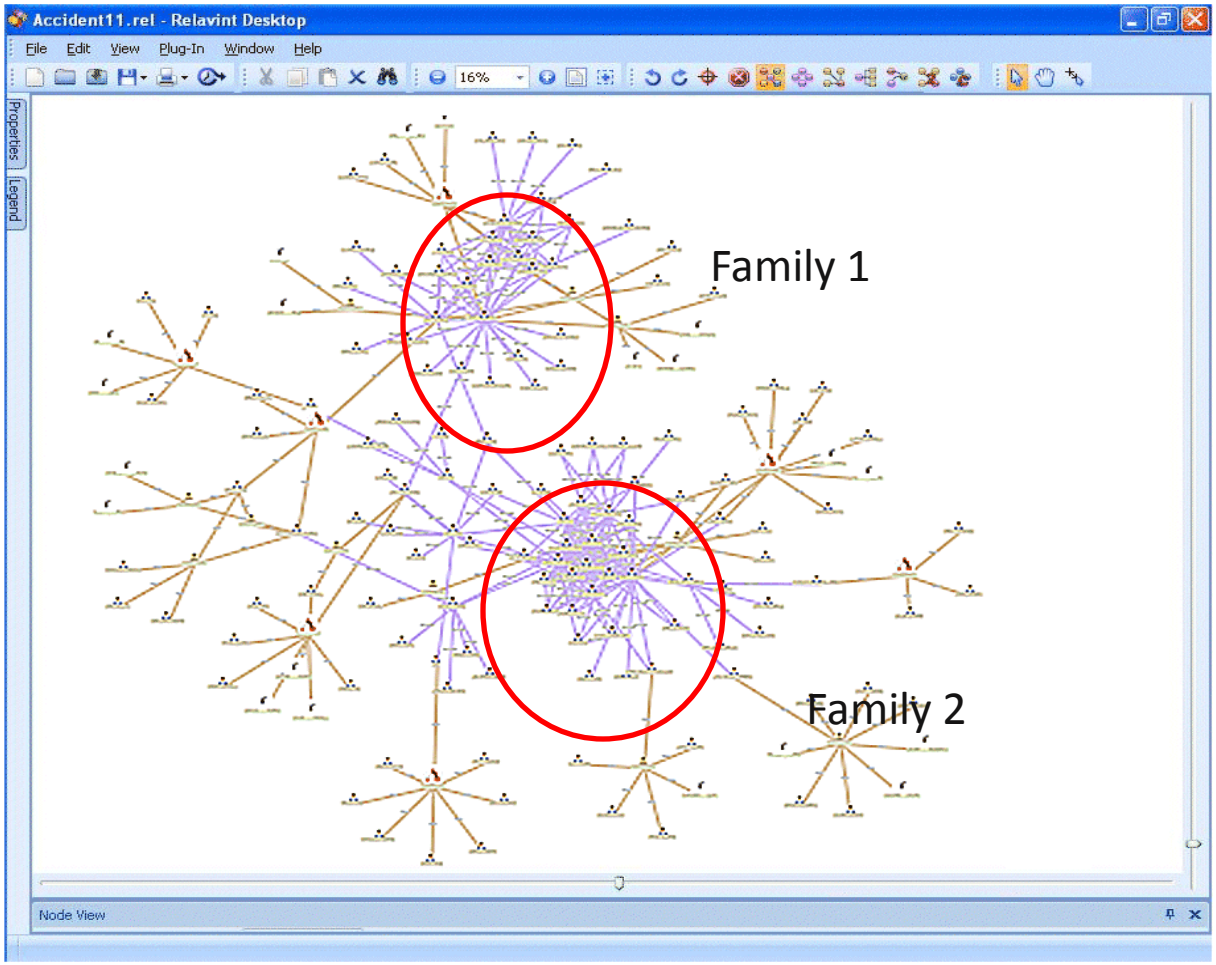
Task

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**.

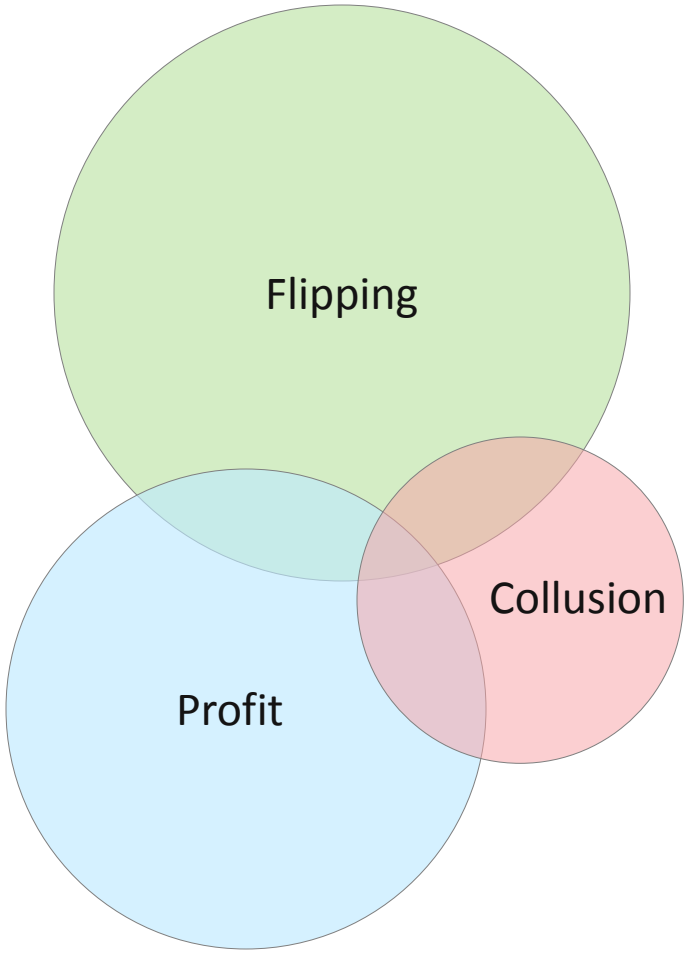
The links were much stronger than the carrier data previously supported.



Property Transaction Risk

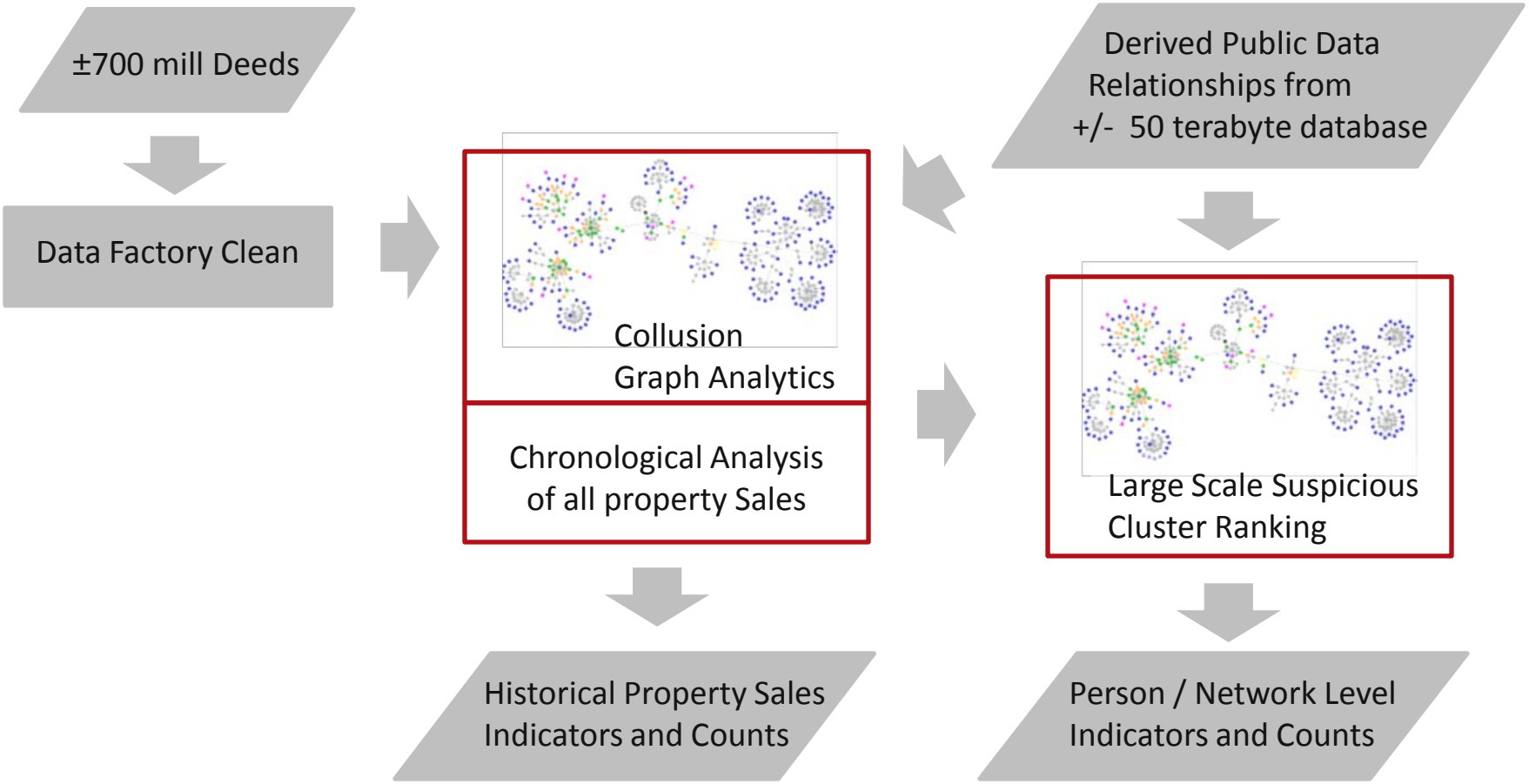
Three core transaction variables measured

- Velocity
- Profit (or not)
- Buyer to Seller Relationship Distance (Potential of Collusion)



Property Transaction Risk

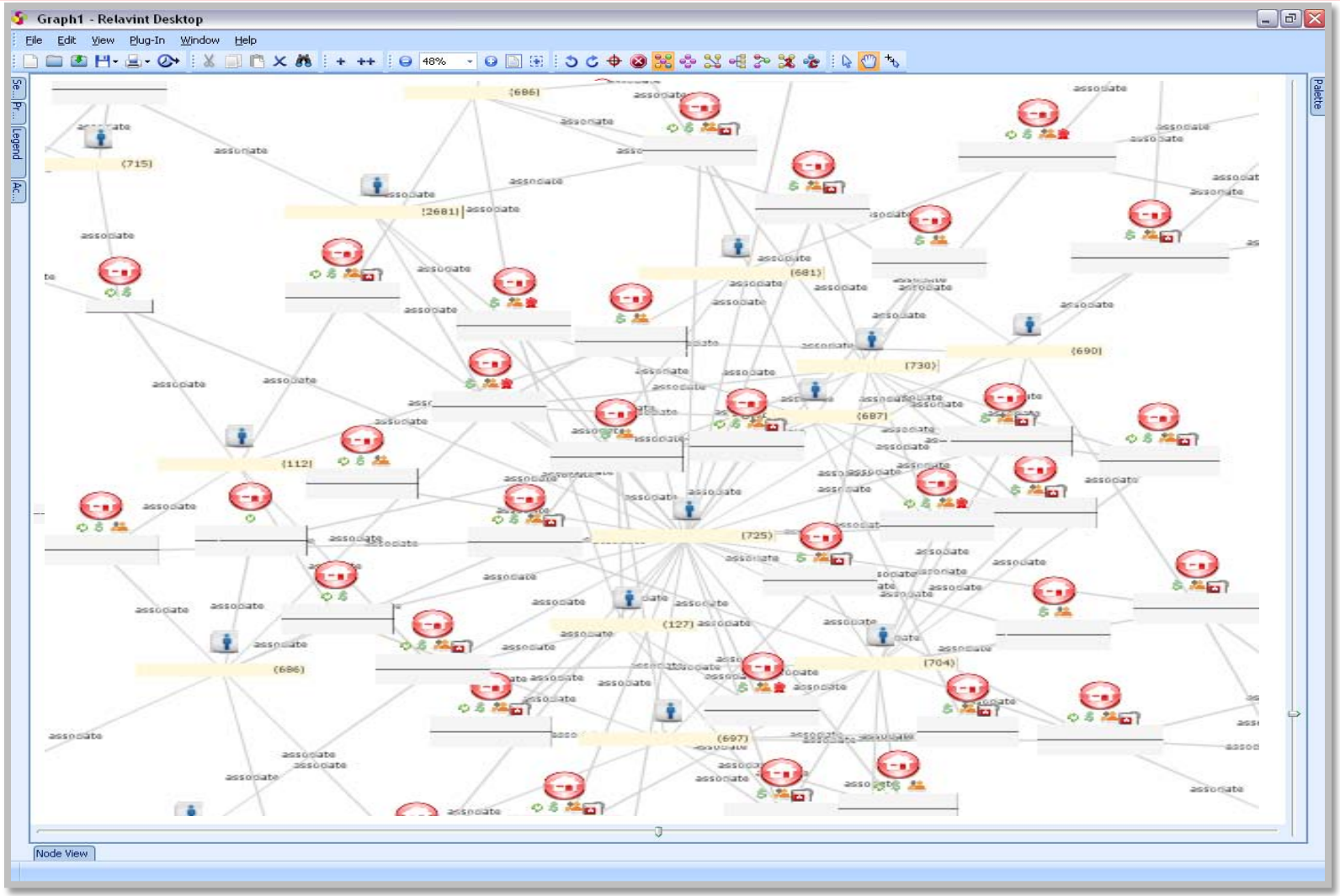
HPCC Systems
http://hpccsystems.com



HPCC Systems

Suspicious Equity Stripping Cluster

HPCC Systems
http://hpccsystems.com



HPCC Systems

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

HPCC Systems
http://hpccsystems.com

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

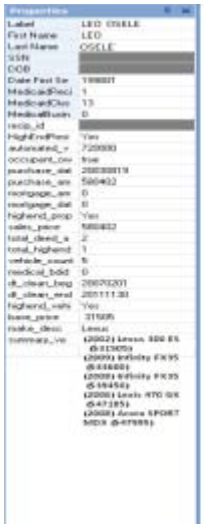
Given a large list of names and addresses, identify social clusters of Medicaid recipients living in expensive houses, driving expensive houses.

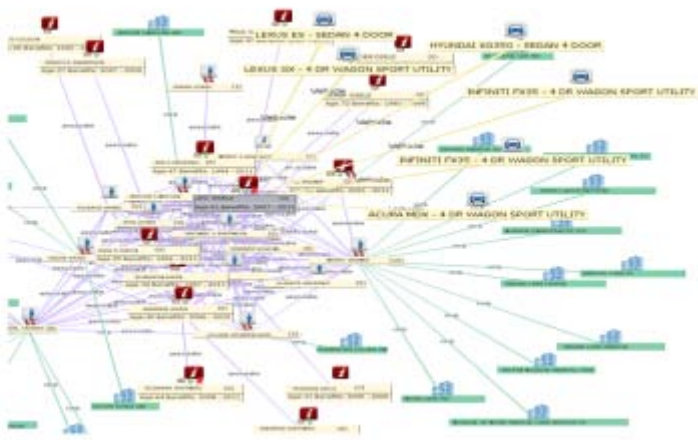
Result

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





Make Description	#	Make Description	#
Mercedes-Benz	46	Chevrolet	2
Lexus	41	Hummer	2
BMW	27	Jeep	2
Infiniti	13	Nissan	2
Acura	9	Toyota	2
Lincoln	8	Aston Martin	1
Audi	7	Bentley	1
Land Rover	7	Cadillac	1
Porsche	6	GMC	1
Jaguar	5	Honda	1
Mercedes Benz	3	Volkswagen	1
Saab	3	Volvo	1

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.

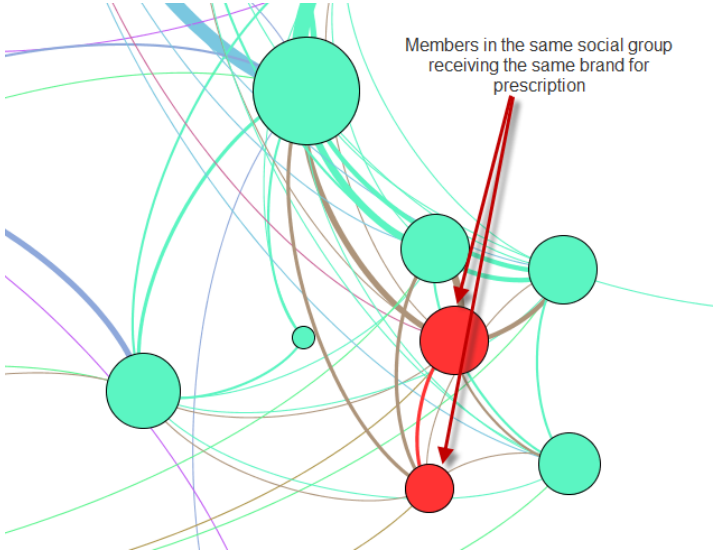
Providers recruit members to provide and escalate services that are not rendered.

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.

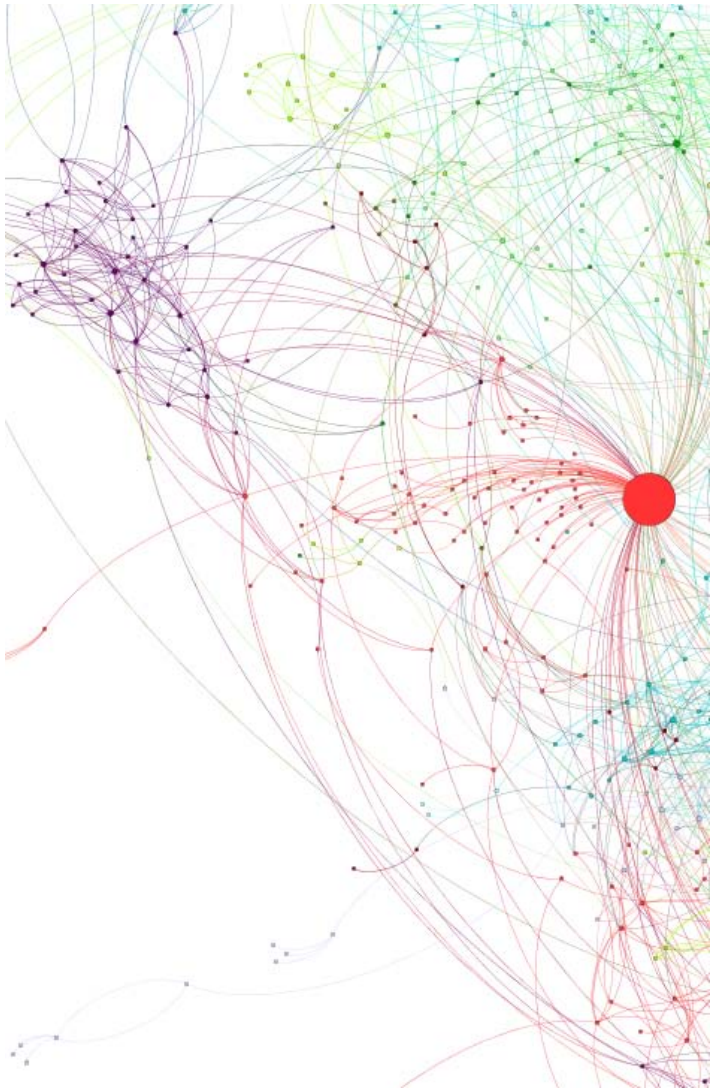
Result

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

HPCC Systems
http://hpccsystems.com



- 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

HPCC Systems
http://hpccsystems.com

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

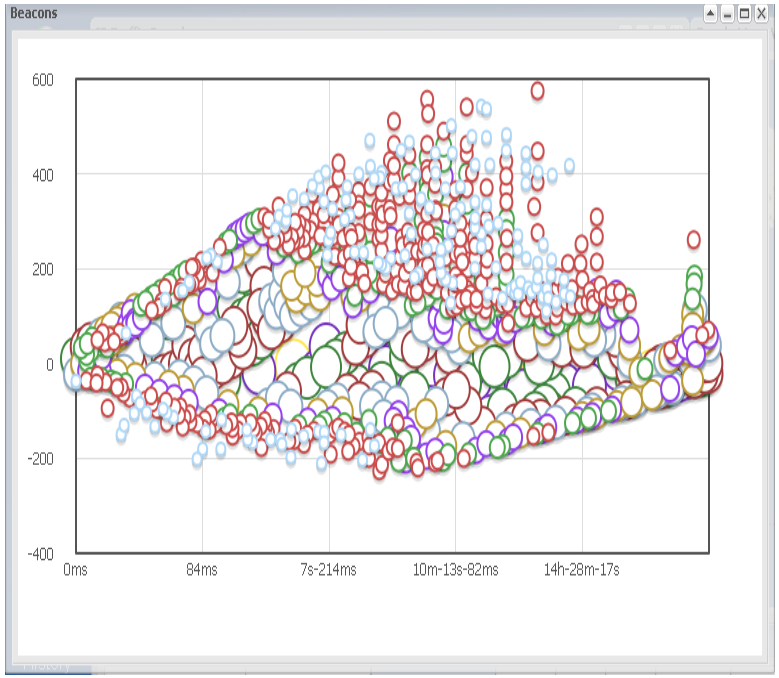
Task

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

Wikipedia Graph Demo

HPCC Systems
http://hpccsystems.com

Scenario

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

Task

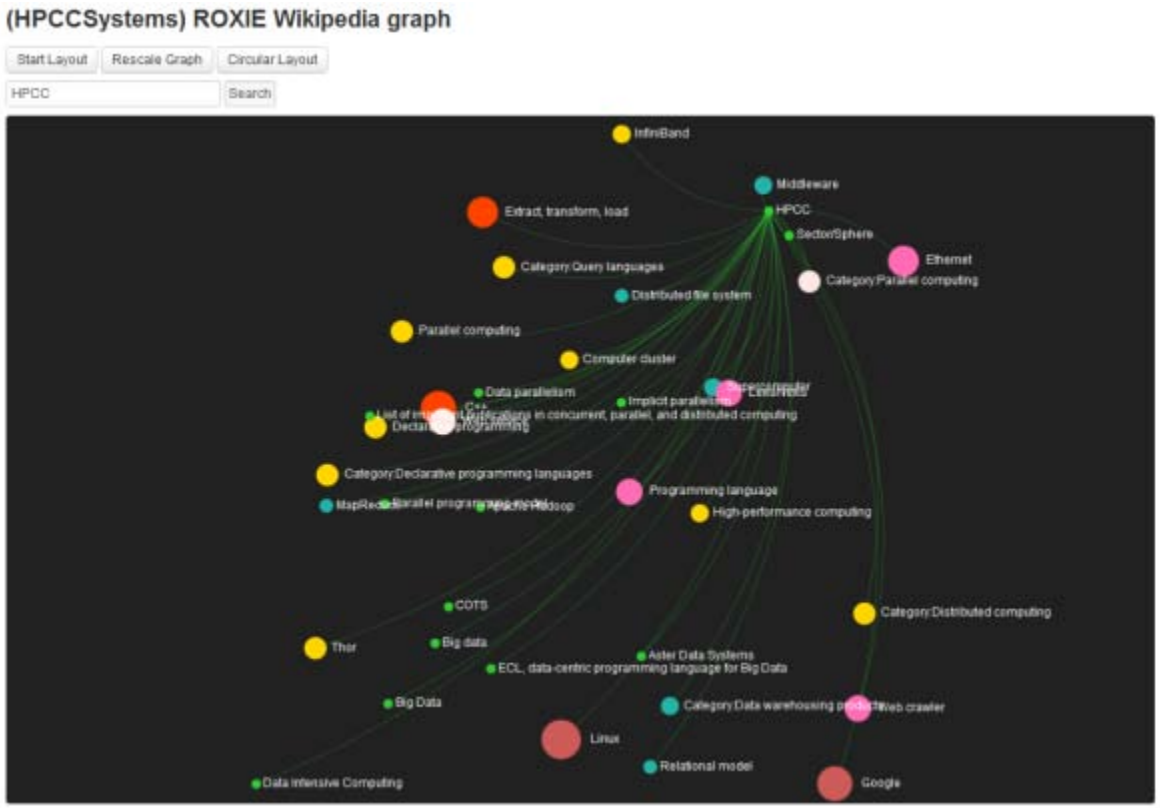
Load the 75GIG English Wikipedia XML snapshot. Strip page links from all pages and run 20 iterations of Google Page Rank Algorithm. Generate indexes and Roxie query to support visualization.

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

HPCC Systems
http://hpccsystems.com

Scenario

21 Billion Rows of Wikipedia Hourly Page view Statistics for a year.

Task

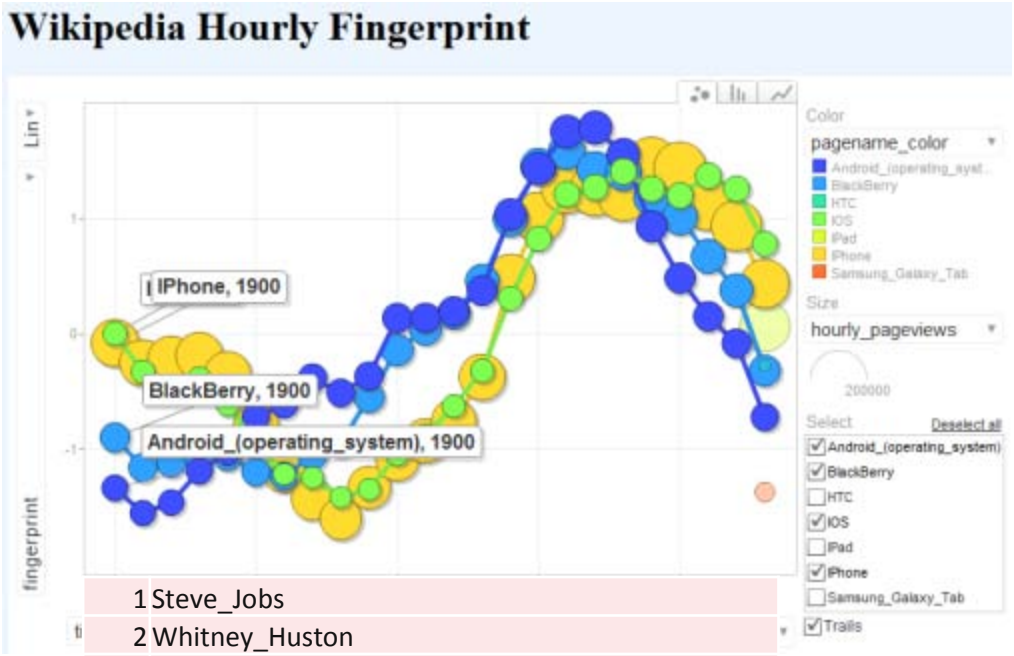
Generate meaningful statistics to understand aggregated global interest in all Wikipedia pages across the 24 hours of the day built off all English Wikipedia page view logs for 12 months.

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.

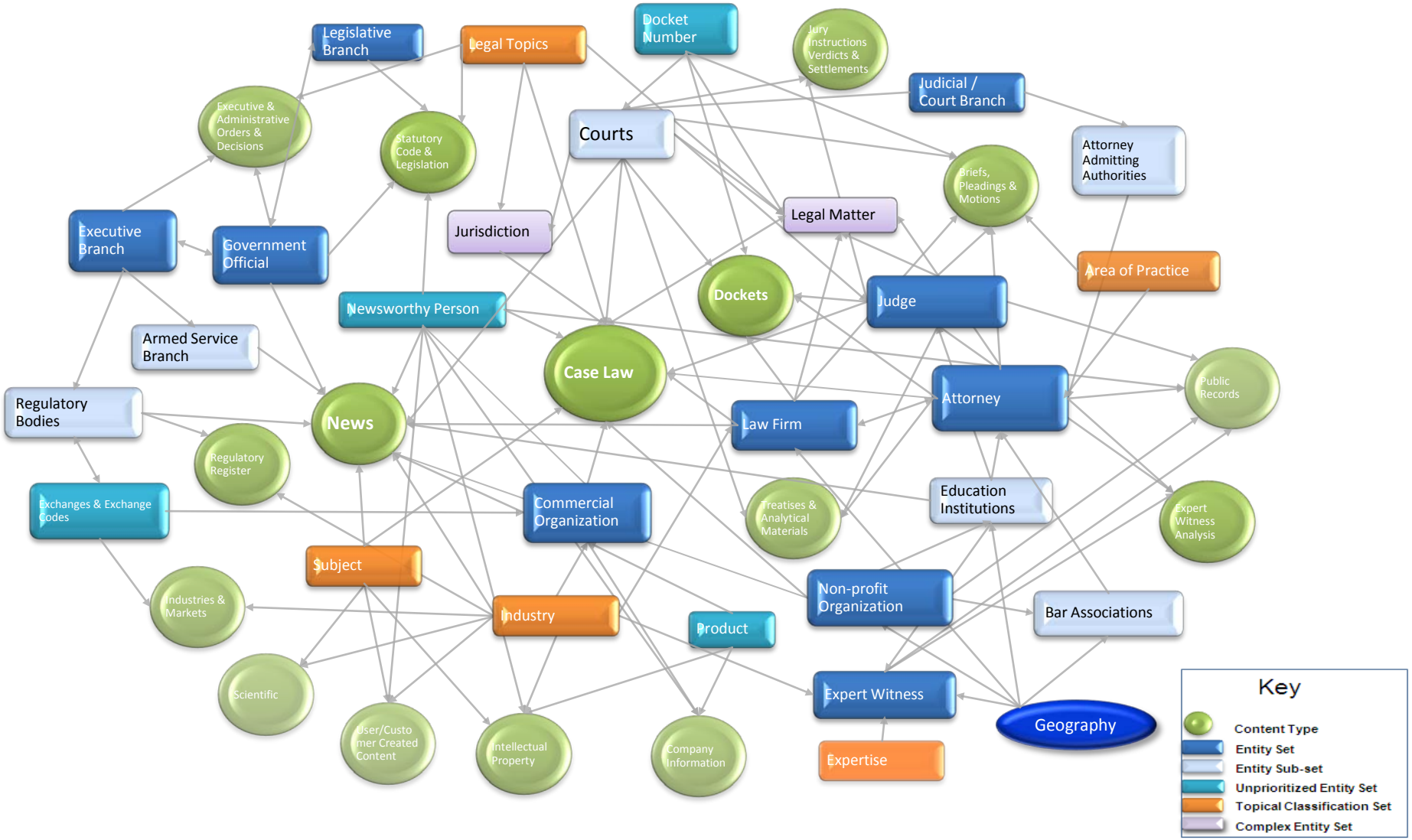
This result can be leveraged with Machine Learning to cluster pages with similar 24hr Fingerprints.



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

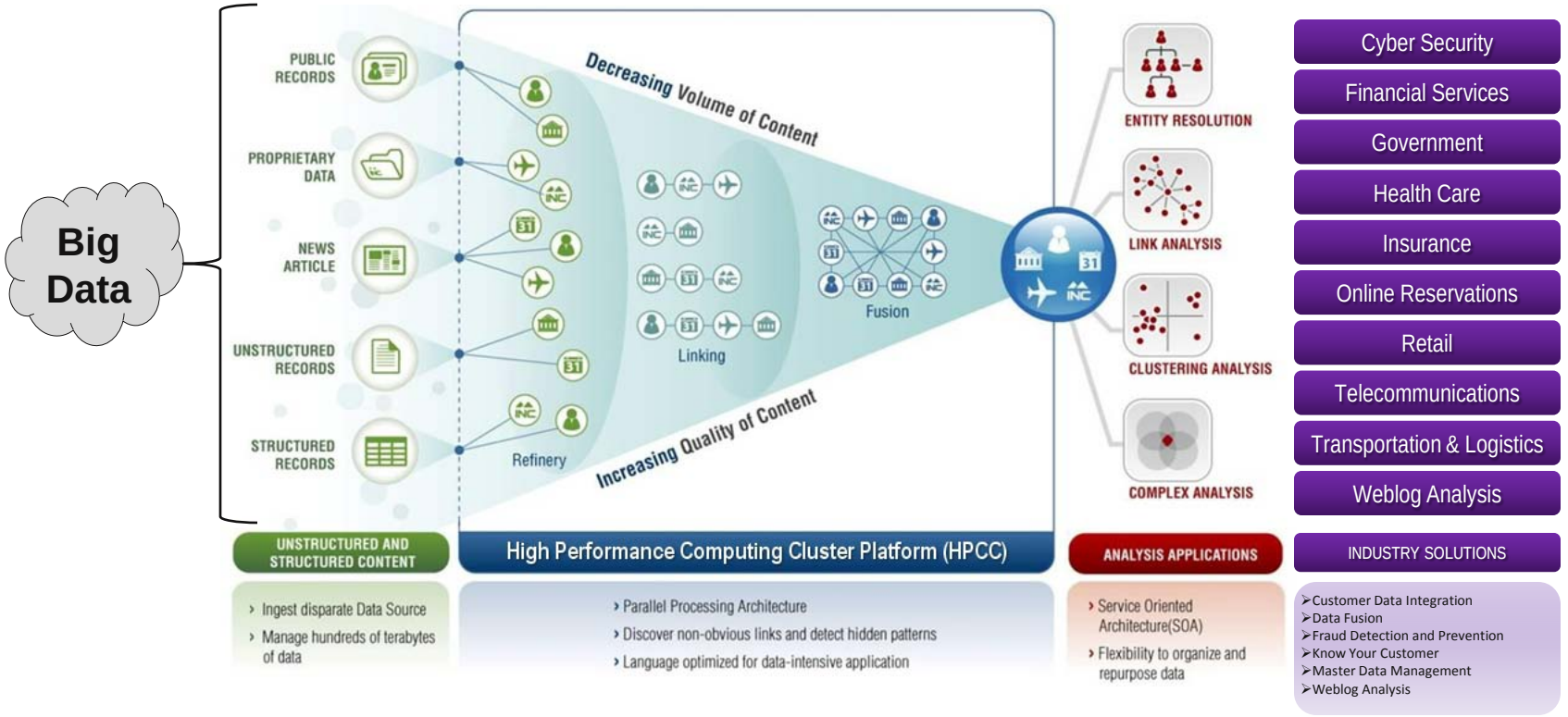
Connected Legal Content Universe

HPCC Systems
http://hpccsystems.com



The Data/Information flow

HPCC Systems
http://hpccsystems.com



- **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

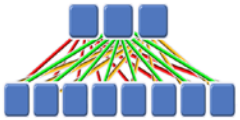
The Three HPCC components

HPCC Systems
http://hpccsystems.com

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)

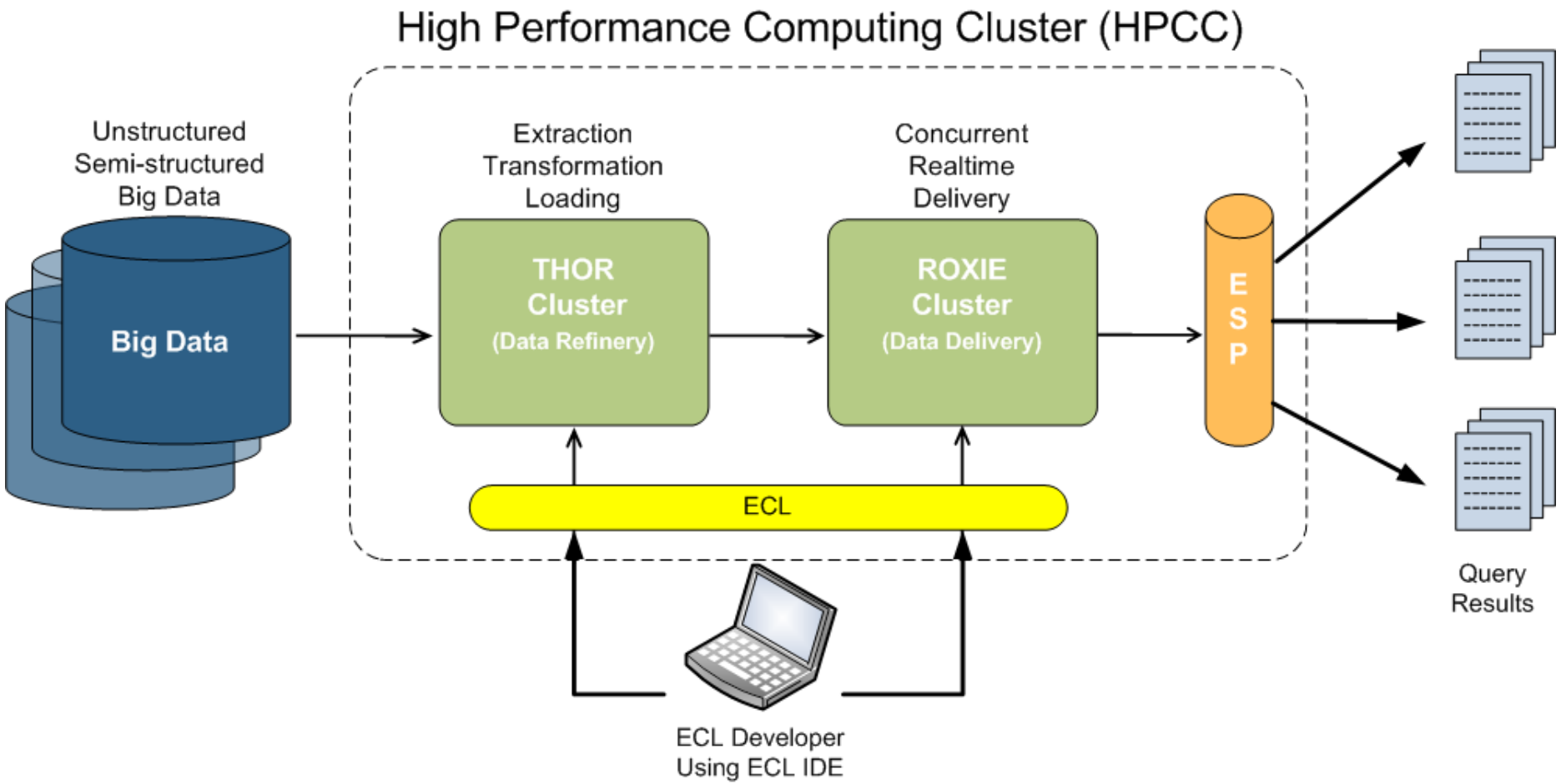
- An easy to use, declarative data-centric programming language optimized for large-scale data management and query processing
- Highly efficient; automatically distributes workload across all nodes.
- Automatic parallelization and synchronization of sequential algorithms for parallel and distributed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

Conclusion: End to End solution

- No need for any third party tools

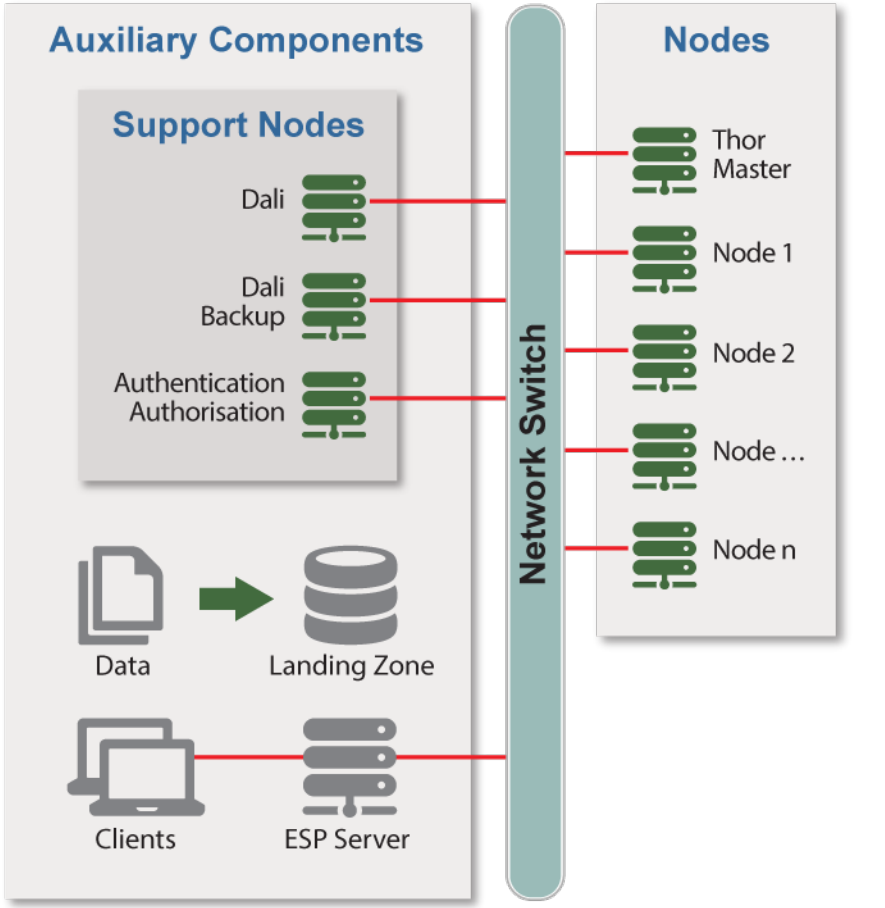
The HPCC Systems platform

HPCC Systems
http://hpccsystems.com

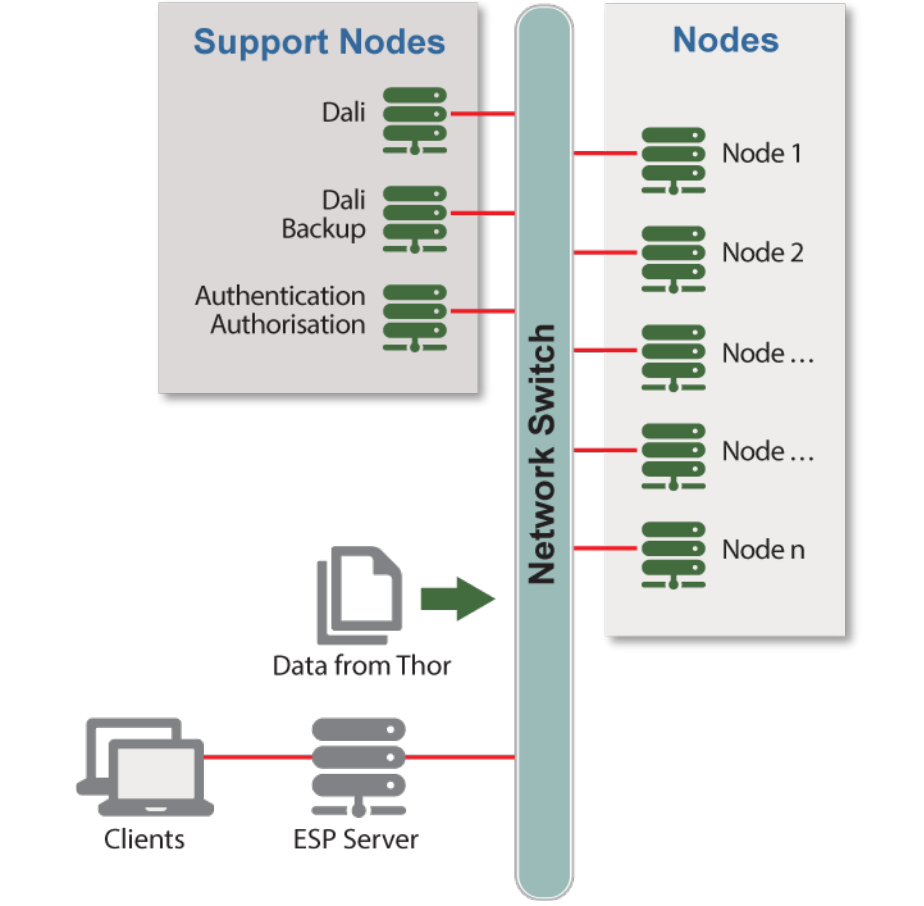


Detailed HPCC Architecture

HPCC Systems
http://hpccsystems.com



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



Roxie
(Rapid Data Delivery Engine)
Physical Layout Schematic Diagram

Enterprise Control Language (ECL)

HPCC Systems
http://hpccsystems.com

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

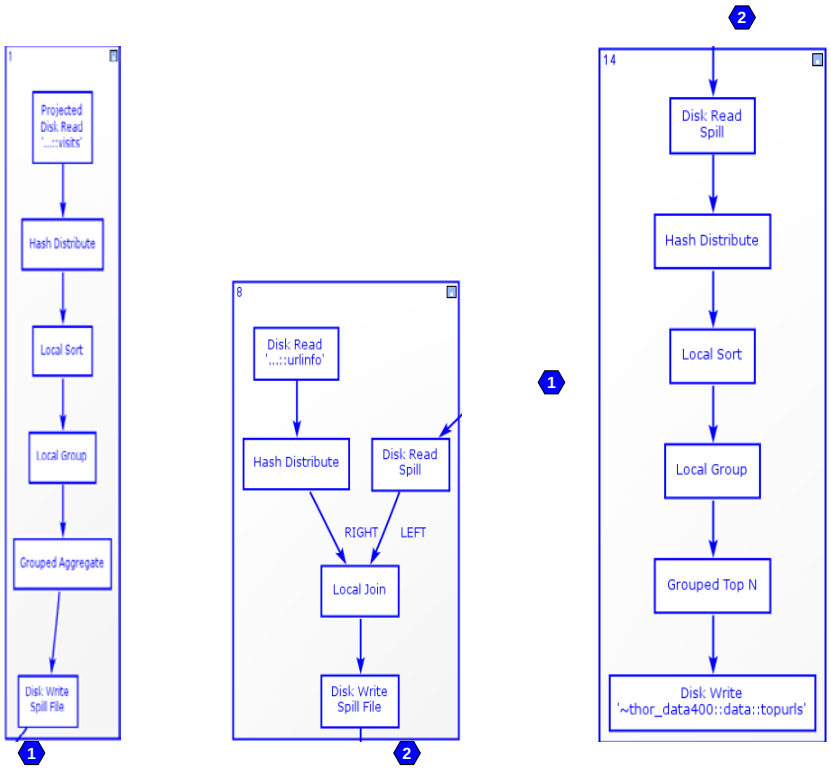
```
// Initialize output log
log_out_init := project(log_init,
                        transform(layout_logo
                        self := left,
                 self := []));

// Create error log
outerrorfile := 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({ric
                     self := left),
```

Enterprise Control Language (ECL)

HPCC Systems
http://hpccsystems.com

- ECL is a declarative, data-centric, programming language which can 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.

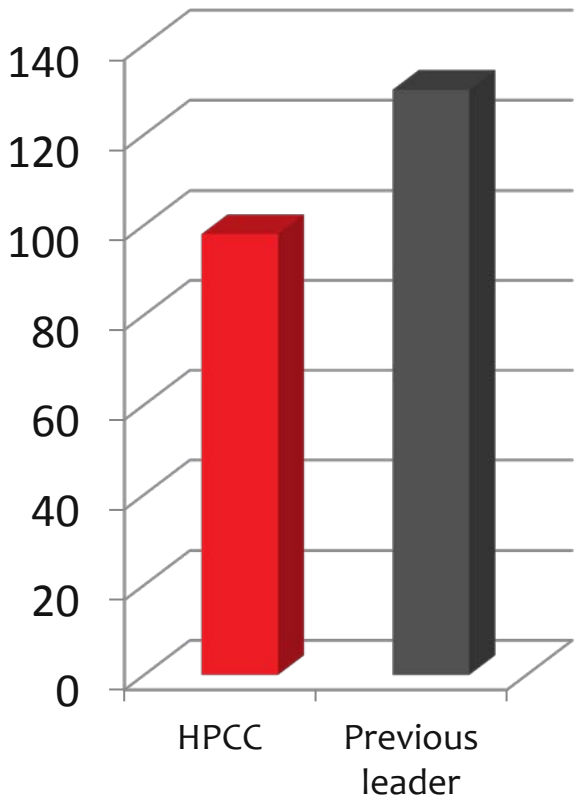
```
// Sample ECL Code
layout_visits := RECORD string user; string url; string time; END;
visits := DATASET('~thor_data400::data::visits',layout_visits,FLAT);

layout_urlInfo := RECORD string url; string category; string pRank; END;
urlInfo := DATASET('~thor_data400::data::urlinfo',layout_urlInfo,FLAT);

// Distribute Visits by URL, Count vists by URL
layout_visitCounts := RECORD visits.url; visits_cnt := COUNT(GROUP); END;
visitCounts := TABLE(DISTRIBUTE(visits,HASH(url)),layout_visitCounts,url,LOCAL);

// Distribute Category by URL, Join category to URLs
visitCountsCat := JOIN(visitCounts,DISTRIBUTE(urlInfo,HASH(url)),LEFT.URL=RIGHT.URL,LOCAL);

// Distribute and Group by Category, Output top 10 URLs for each category
topUrls := TOPN(GROUP(DISTRIBUTE(visitCountsCat,HASH(category)),category,ALL,LOCAL),10, -visits_cnt);
OUTPUT(topUrls,,'~thor_data400::data::topurls',OVERWRITE);
```



Terasort Benchmark results

HPCC Systems
http://hpccsystems.com

Execution Time (seconds)



Productivity

3 Lines of ECL

```
// Perform global terasort
rec := record
      string10  key;
      string10  seq;
      string80  fill;
      end;
in := DATASET('~thor::terasort1',rec,FLAT);
OUTPUT(SORT(in,key,UNSTABLE),,'nhtest::terasort1out',overwrite);
//End
```

700+ Lines of Java MapReduce Code

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

/**
 * An inner trie node that contains 256 children based on the next
 * character.
 */
static class InnerTrieNode extends TrieNode {
  private TrieNode[] child = new TrieNode[256];

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

Space/Cost

HPCC

Previous leader

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

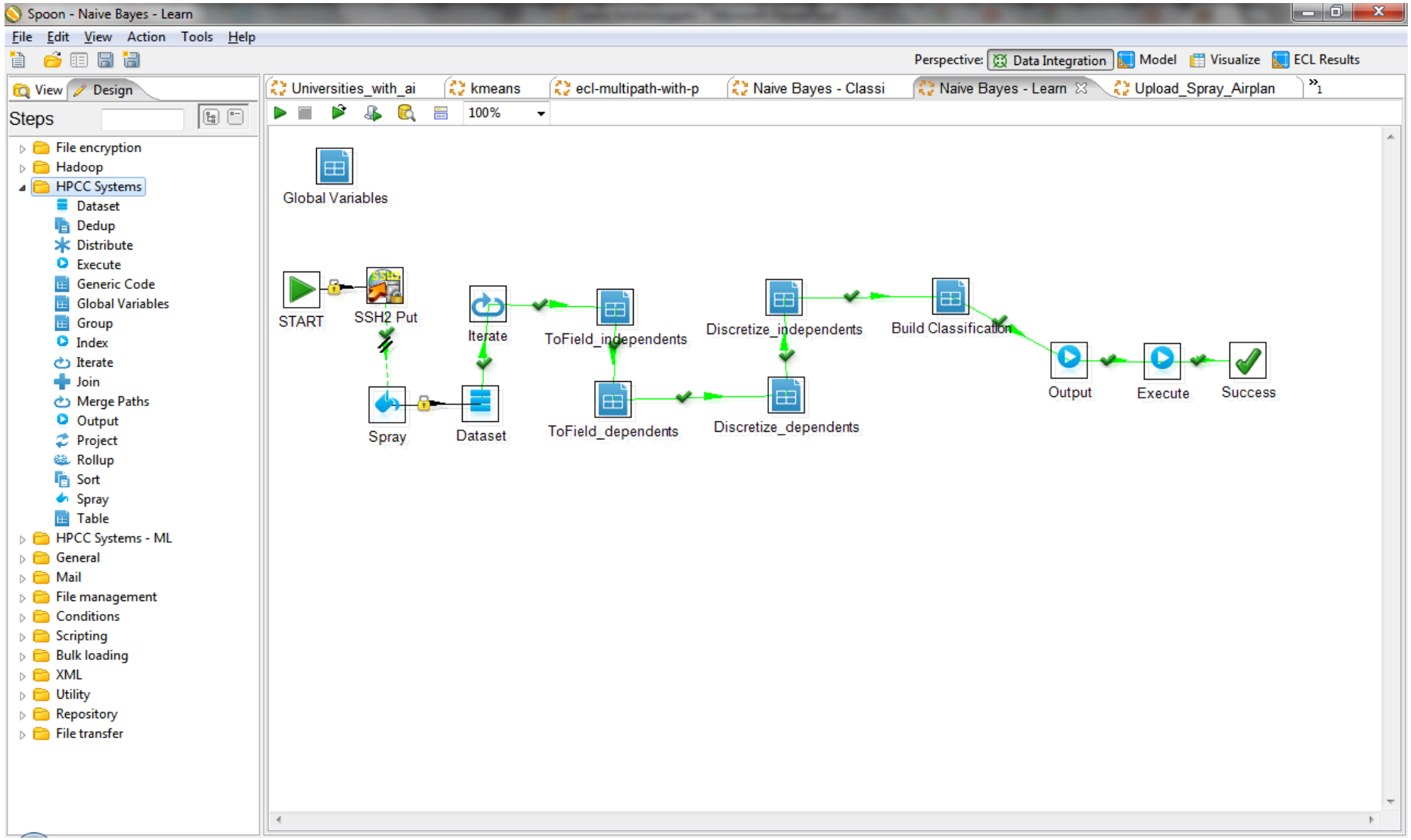
ECL-ML: extensible ML on HPCC (ii)

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

Kettle: a GUI to ECL-ML

HPCC Systems
http://hpccsystems.com



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

Integration code of PaperBoat with ECL

The goal of this section is to give more details about the integration mechanism of PaperBoat. As mentioned in *C++ code overview* the key point is to override some methods of the `WorkSpace` class.

All the files for connecting PaperBoat with ECL can be found inside the directory

```
ecl-pb/ecl-pb-glue
```

Let's start with the `workspace.h` file.

We define a new `WorkSpace` class that inherits from the `fl::ws::WorkSpace`. We need to provide new methods that translate an HPCC Dataset to a PaperBoat Table and loads it in the workspace. We also need to provide a method that does the opposite, exports a PaperBoat Table into an HPCC Dataset.

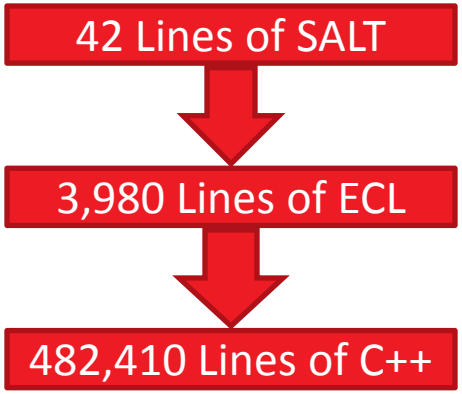
```
namespace fl { namespace hpcc {

class WorkSpace : public fl::ws::WorkSpace {
  public:

    // We have some type definitions that we need to ommit for
    // the shake of simplification
    ......

    /**
     * @brief This method reads an HPCC Dataset, converts it to
     *        a PaperBoat Table and stores it inside the workspace
     *        The function is templatized over the precision T of the Dataset.
     */
    template<typename T>
    void LoadDenseHPCCDataSet(const std::string &name,
        index_t n_attributes,
        index_t n_entries,
        const void *in_data);

    /**
     * @brief This method exports a PaperBoat Table residing in the workspace
     *        to an HPCC Dataset. The function is templatized over the
     *        precision T of the HPCC Dataset.
     */
    template<typename T>
```

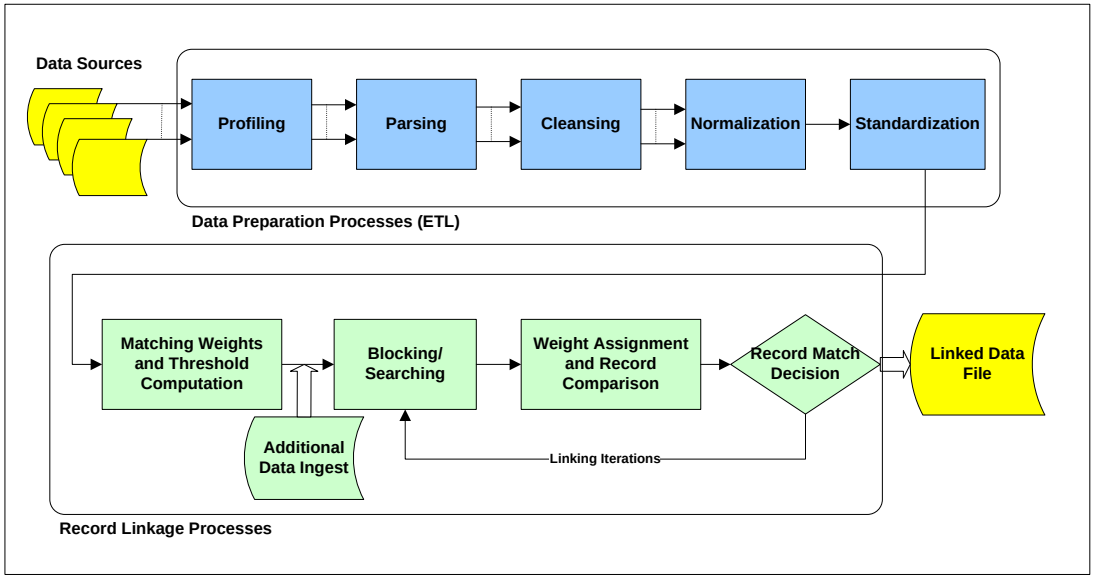
HPCC-PaperBoat ML

- **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





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”?
- What if these two records for “John Smith” were already linked to “Flavio Villanustre”? How many “John Smith” does “Flavio Villanustre” know? Are these two “John Smith” records referring to the same person now?

```
1  OPTIONS:-ga
2  MODULE:MyModule
3  FILENAME:Sample
4  IDFIELD:EXISTS:BDID
5  RIDFIELD:rcid
6  RECORDS:9000000
7  POPULATION:1000000
8  NINES:3
9  FIELDTYPE:DEFAULT:LEFTTRIM:NOQUOTES("'):
10 FIELDTYPE:NUMBER:ALLOW(0123456789):
11 FIELDTYPE:ALPHA:CAPS:ALLOW(ABCDEFGHIJKLMNOPQRSTUVWXYZ):
12 FIELDTYPE:WORDBAG:CAPS:ALLOW(ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789'):SPACES( <>{}[]-^=!+&,./):ONFAIL(CLEAN):
13 FIELDTYPE:CITY:LIKE(WORDBAG):LENGTHS(0,4..):ONFAIL(BLANK):
14 FIELD:source:CARRY:
15 FIELD:vendor_id:CARRY:
16 FIELD:dt_first_seen:RECORDDATE(FIRST):
17 FIELD:dt_last_seen:RECORDDATE(LAST):
```

msa	populated	maxlength	avelength	populated	maxlength	avelength	populated	maxlength	avelength
85.6									

	txt	numberofrecords	populated source pcnt	maxlength source	avelength source	populated vendor id pcnt	maxlength vendor id	avelength vendor id
1	Data Profiling Summa	7959271	100.000000	2	1.787971	81.667705	34	15.146526

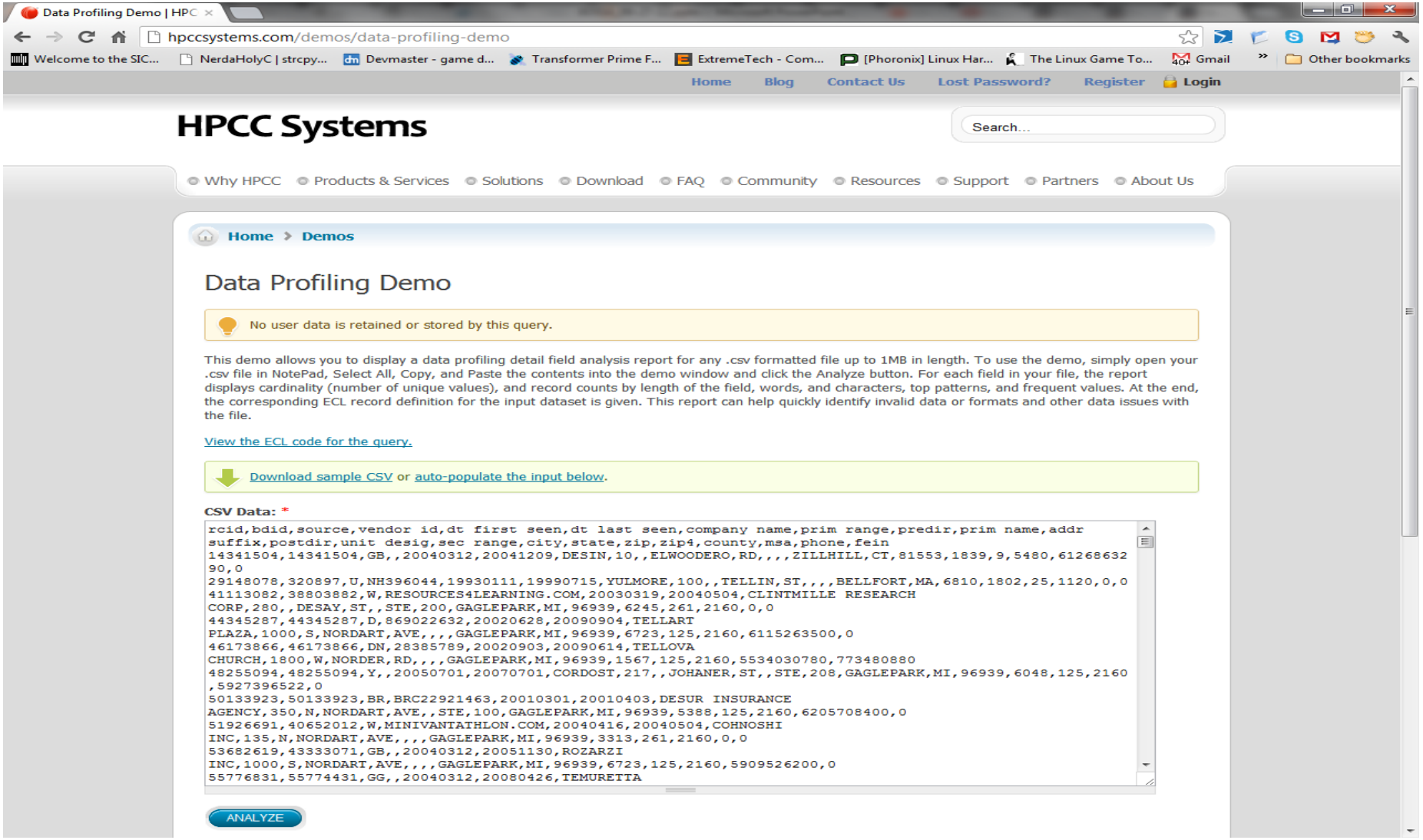
```
26
27
28 FIELD:zip:LIKE(NUMBER):11,001
29 FIELD:zip4:LIKE(NUMBER):CONTEXT(zip):11,007
30 FIELD:county:6,001
```

fldno	fieldname	cardinality	len	cnt	words	cnt	characters	cnt	patterns	cnt	frequent terms	cnt
1	19	phone	970479	1	4946302	1	7959271	0	8485069	9	4946302	0	4946302
				10	2970565			2	3325862	9999999999	2970565	8002882020	979
				7	42044			3	3182462	9999999	42044	9146408100	833

	fields	cnt
1	:company_name:prim_range:prim_name:addr_suffix:city:state:zip:zip4:Phone	239398
2	:company_name:prim_range:prim_name:addr_suffix:city:state:zip:zip4	236589
3	:company_name:prim_range:predir:prim_name:addr_suffix:city:state:zip:zip4:Phone	99631
4	:company_name:prim_range:predir:prim_name:addr_suffix:city:state:zip:zip4	80250
5	:company_name:city:state	55903
6	:company_name:Phone	40800
7	:company_name:prim_range:prim_name:addr_suffix:unit_desig:sec_range:city:state:zip:zip4:Phone	24342
8	:company_name:prim_range:prim_name:addr_suffix:postdir:city:state:zip:zip4:Phone	22128
9	:company_name:prim_name:city:state:zip:zip4:Phone	18771
10	:company_name:prim_range:prim_name:addr_suffix:postdir:city:state:zip:zip4	17454
11	:company_name:prim_range:prim_name:addr_suffix:city:state:zip:zip4:fein	16296
12	:company_name:prim_range:prim_name:city:state:zip:zip4	12038
13	:company_name:prim_range:predir:prim_name:addr_suffix:unit_desig:sec_range:city:state:zip:zip4:Phone	11745
14	:company_name:prim_range:prim_name:city:state:zip:zip4:Phone	11670
15	:company_name:prim_range:prim_name:addr_suffix:unit_desig:sec_range:city:state:zip:zip4	9883

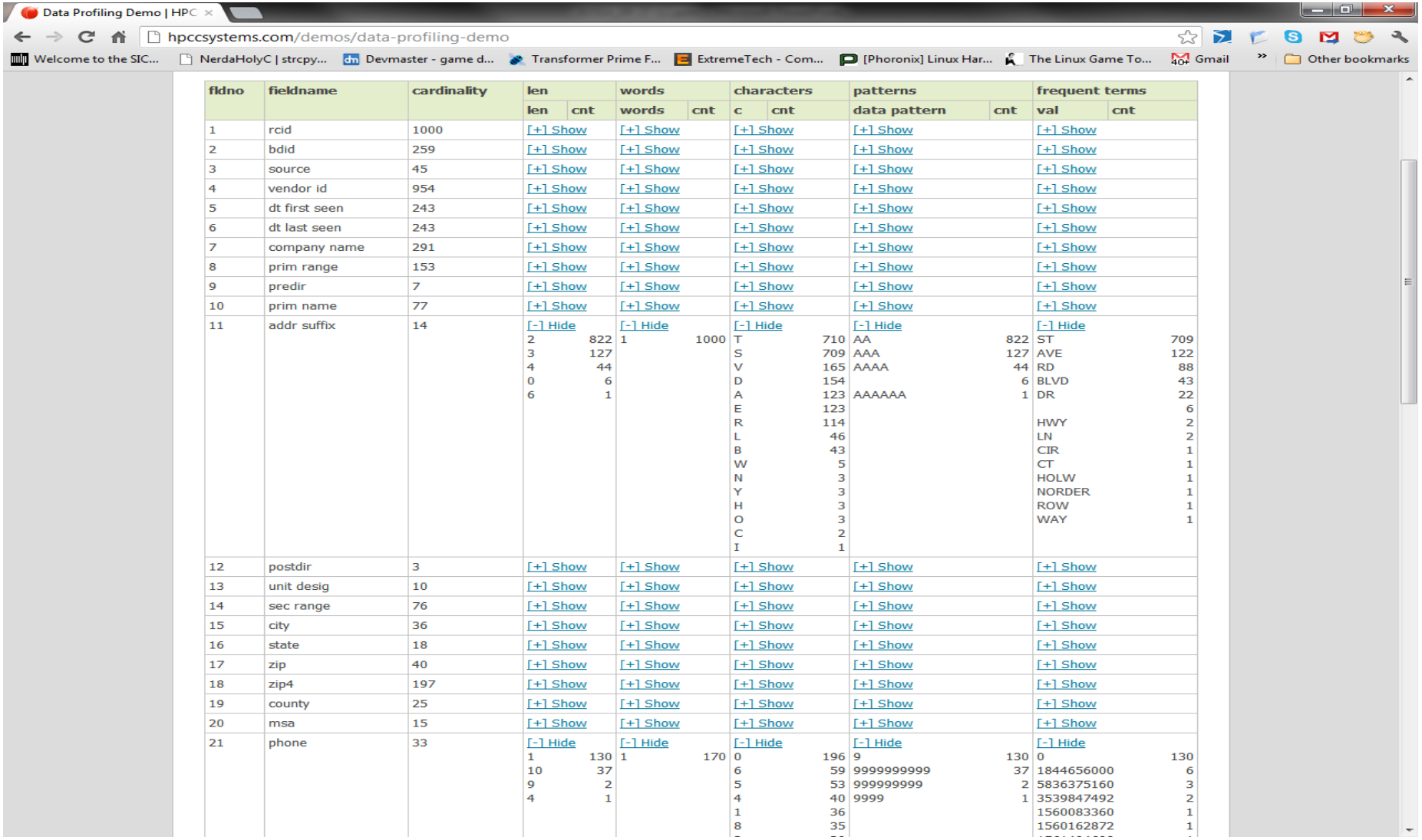
SALT Demo

HPCC Systems
http://hpccsystems.com



SALT Demo

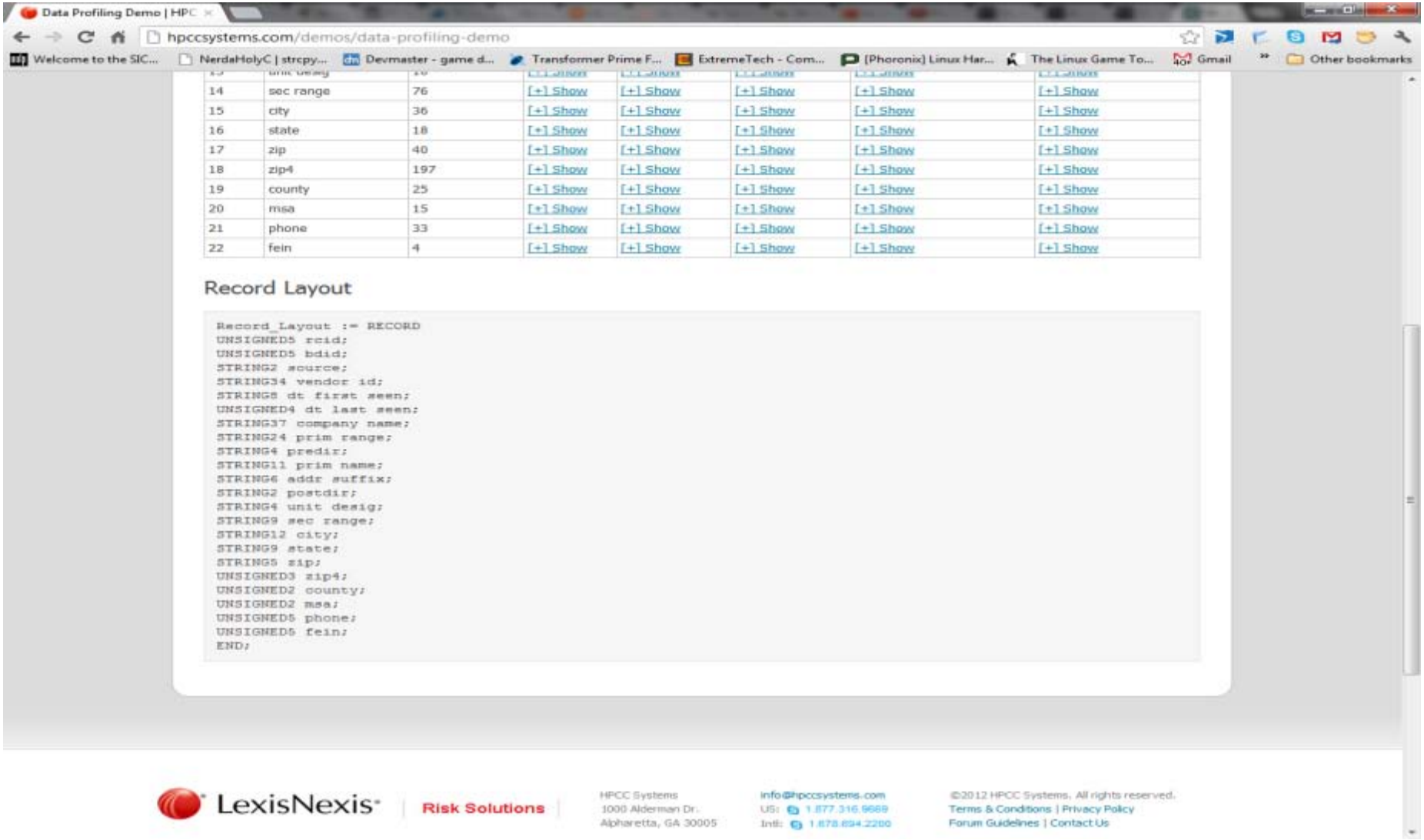
HPCC Systems
http://hpccsystems.com



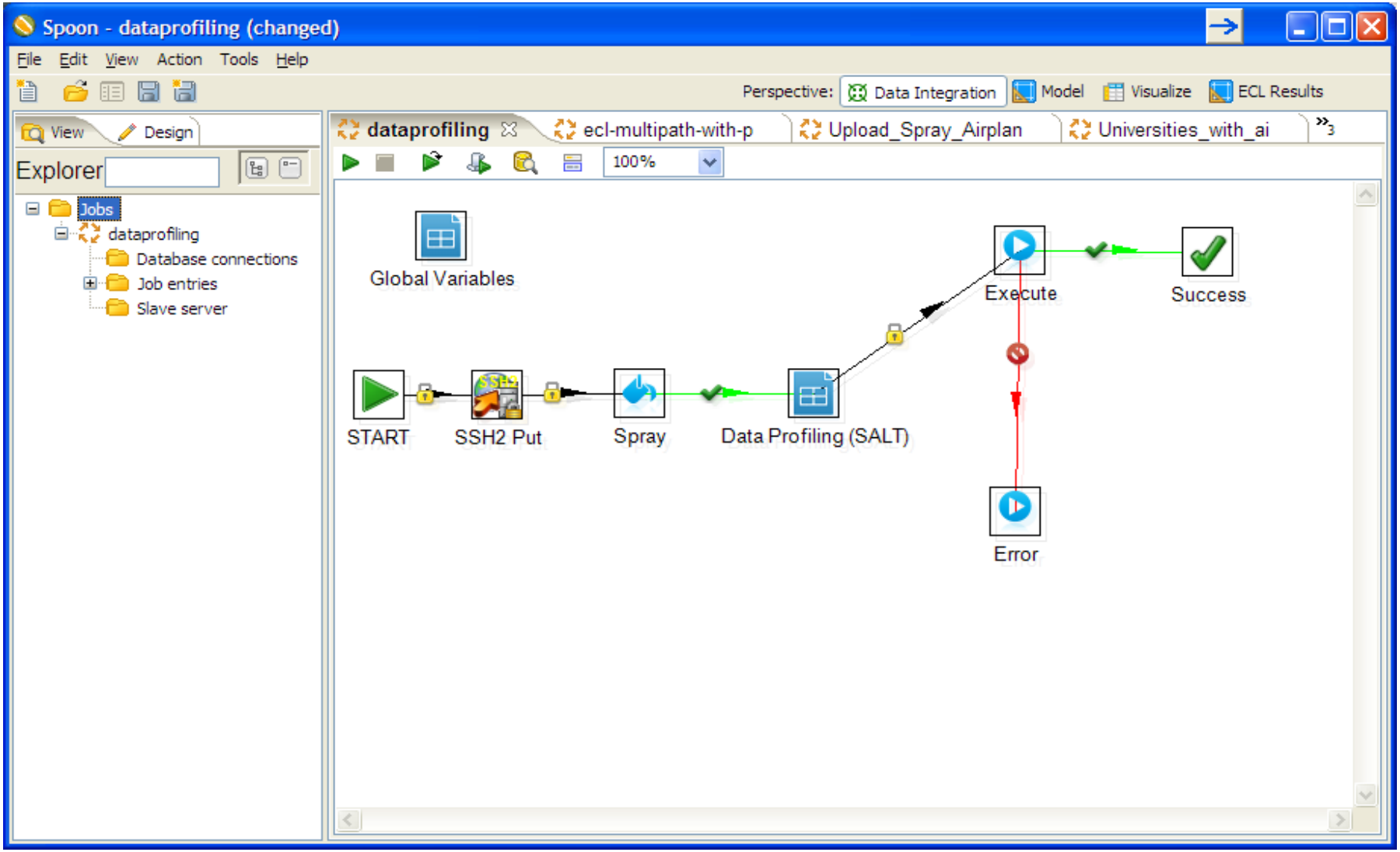
fldno	fieldname	cardinality	len		words		characters		patterns		frequent terms	
			len	cnt	words	cnt	c	cnt	data pattern	cnt	val	cnt
1	rcid	1000	[+] Show		[+] Show		[+] Show		[+] Show		[+] Show	
2	bdid	259	[+] Show		[+] Show		[+] Show		[+] Show		[+] Show	
3	source	45	[+] Show		[+] Show		[+] Show		[+] Show		[+] Show	
4	vendor id	954	[+] Show		[+] Show		[+] Show		[+] Show		[+] Show	
5	dt first seen	243	[+] Show		[+] Show		[+] Show		[+] Show		[+] Show	
6	dt last seen	243	[+] Show		[+] Show		[+] Show		[+] Show		[+] Show	
7	company name	291	[+] Show		[+] Show		[+] Show		[+] Show		[+] Show	
8	prim range	153	[+] Show		[+] Show		[+] Show		[+] Show		[+] Show	
9	predir	7	[+] Show		[+] Show		[+] Show		[+] Show		[+] Show	
10	prim name	77	[+] Show		[+] Show		[+] Show		[+] Show		[+] Show	
11	addr suffix	14	[-] Hide		[-] Hide		[-] Hide		[-] Hide		[-] Hide	
			2	822	1	1000	T	710	AA	822	ST	709
			3	127			S	709	AAA	127	AVE	122
			4	44			V	165	AAAA	44	RD	88
			0	6			D	154		6	BLVD	43
			6	1			A	123	AAAAAA	1	DR	22
							E	123				6
							R	114			HWY	2
							L	46			LN	2
							B	43			CIR	1
							W	5			CT	1
							N	3			HOLW	1
							Y	3			NORDER	1
							H	3			ROW	1
							O	3			WAY	1
							C	2				
							I	1				
12	postdir	3	[+] Show		[+] Show		[+] Show		[+] Show		[+] Show	
13	unit desig	10	[+] Show		[+] Show		[+] Show		[+] Show		[+] Show	
14	sec range	76	[+] Show		[+] Show		[+] Show		[+] Show		[+] Show	
15	city	36	[+] Show		[+] Show		[+] Show		[+] Show		[+] Show	
16	state	18	[+] Show		[+] Show		[+] Show		[+] Show		[+] Show	
17	zip	40	[+] Show		[+] Show		[+] Show		[+] Show		[+] Show	
18	zip4	197	[+] Show		[+] Show		[+] Show		[+] Show		[+] Show	
19	county	25	[+] Show		[+] Show		[+] Show		[+] Show		[+] Show	
20	msa	15	[+] Show		[+] Show		[+] Show		[+] Show		[+] Show	
21	phone	33	[-] Hide		[-] Hide		[-] Hide		[-] Hide		[-] Hide	
			1	130	1	170	0	196	9	130	0	130
			10	37			6	59	9999999999	37	1844656000	6
			9	2			5	53	999999999	2	5836375160	3
			4	1			4	40	9999	1	3539847492	2
							1	36			1560083360	1
							8	35			1560162872	1

SALT Demo

HPCC Systems
http://hpccsystems.com



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

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

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 3.10 of the HPCC Systems platform is out!
- Ongoing R&D (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.)
  - Embedding other languages (Python, JavaScript, Java and R, for starters)
  - Integration with other tools and systems (Pentaho Mondrian, R, etc.)

Useful links

- LexisNexis Open Source HPCC Systems Platform: http://hpccsystems.com
- Machine Learning portal: http://hpccsystems.com/ml
- PaperBoat integration: http://ismion.com/documentation/ecl-pb/index.html
- 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???