CLEMSON UNIVERSITY

High Performance Computing Cluster

Collection – Structured, unstructured and semi-structured data from multiple sources

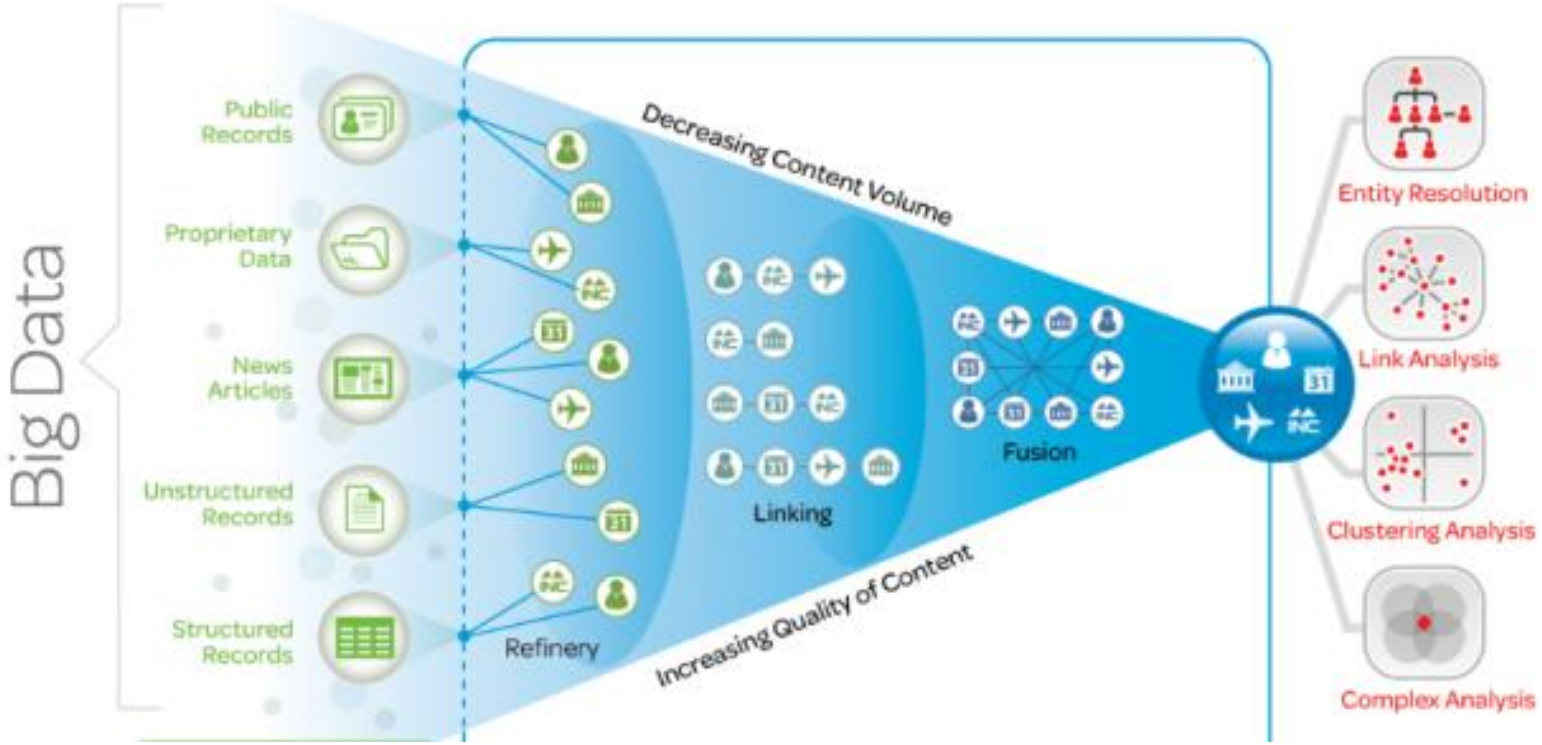
Ingestion – loading vast amounts of data onto a single data store

Discovery & Cleansing – understanding format and content; clean up and formatting

Integration – linking, entity extraction, entity resolution, indexing and data fusion

Analysis – Intelligence, statistics, predictive and text analytics, machine learning

Delivery – querying, visualization, real time delivery on enterprise-class availability

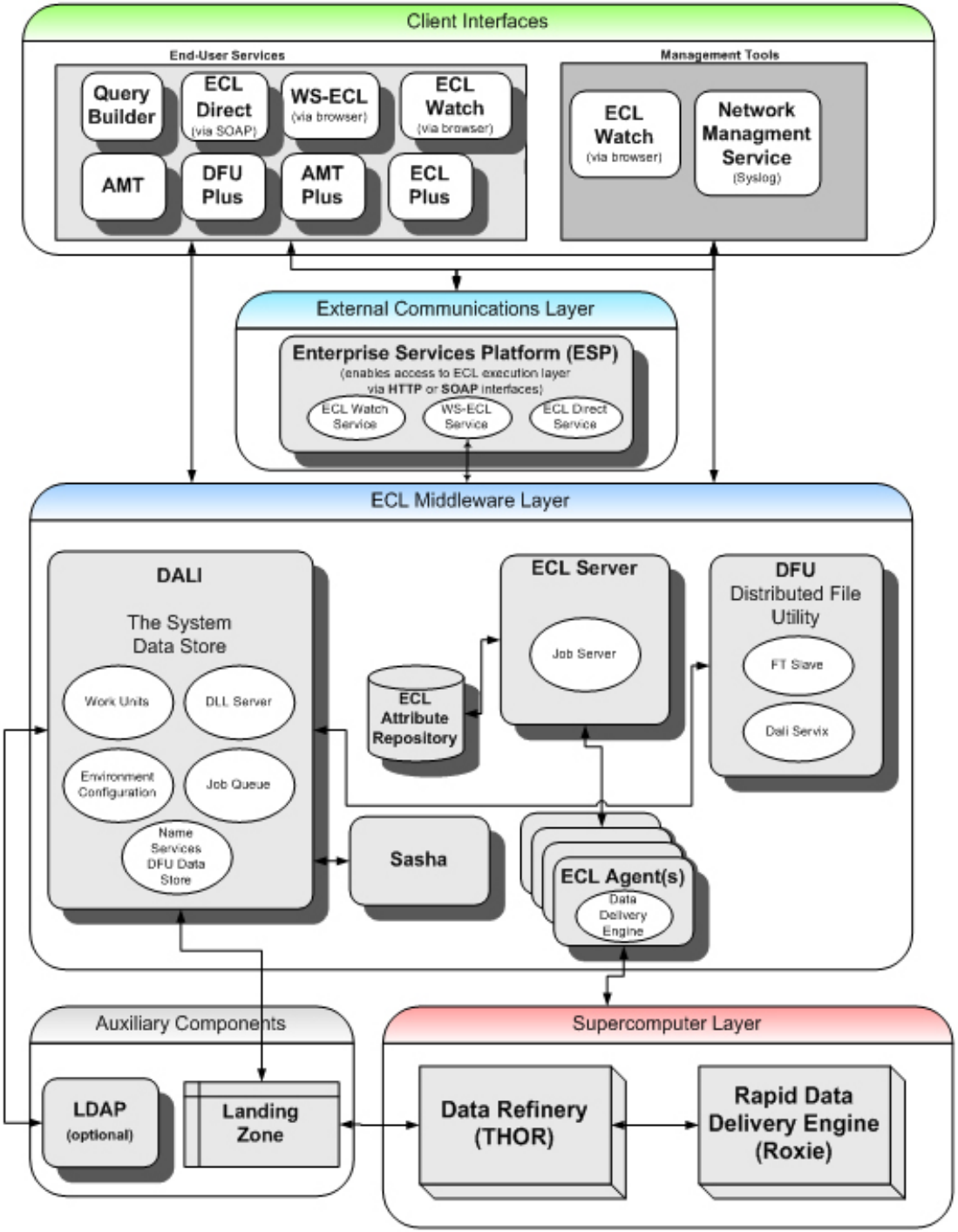

Big Data
Public Records
Proprietary Data
News Articles
Unstructured Records
Structured Records
Refinery
Decreasing Content Volume
Linking
Fusion
Increasing Quality of Content
Entity Resolution
Link Analysis
Clustering Analysis
Complex Analysis

High Performance Computing Cluster (HPCC)

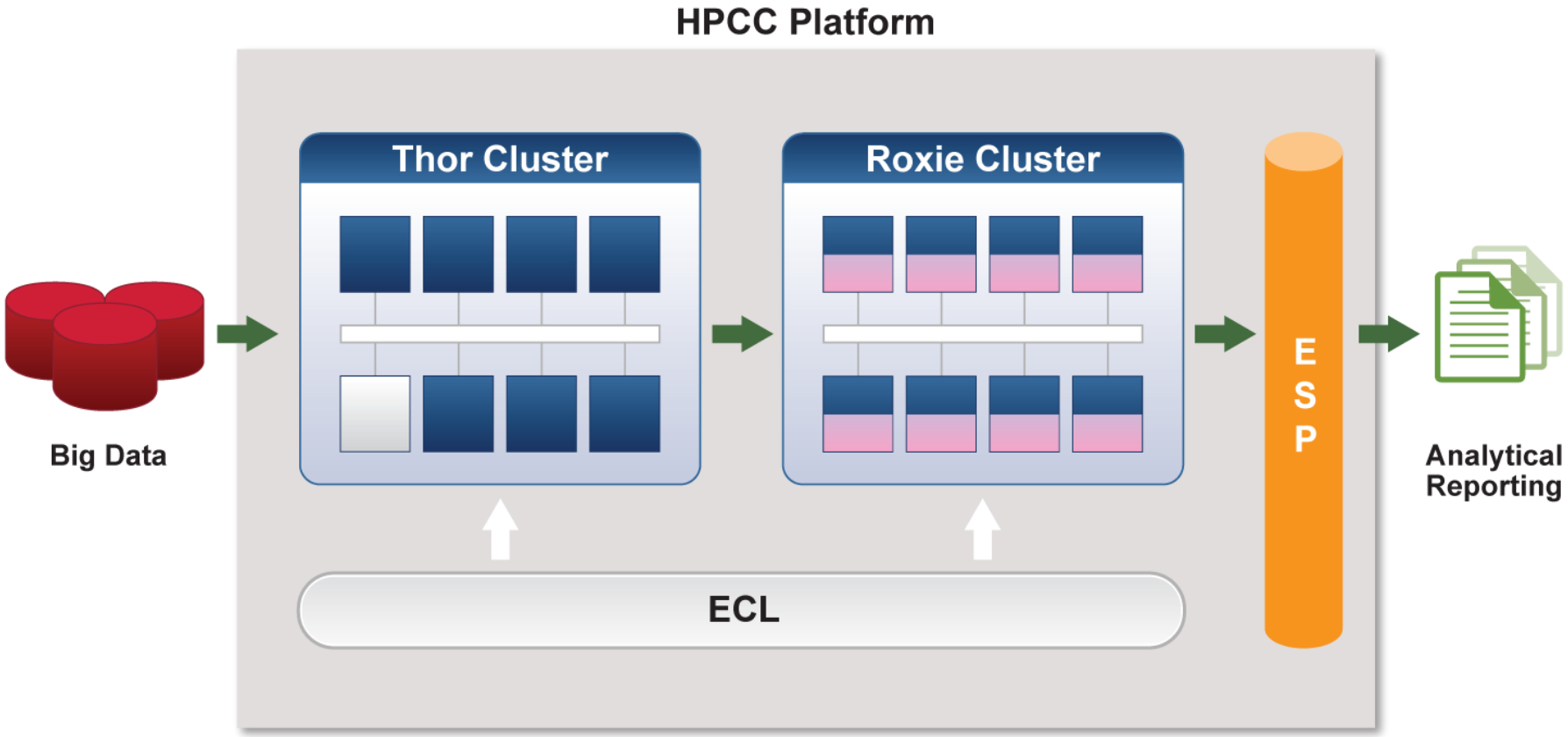
- Large-scale data storage and analytics platform
- Developed by LexisNexis Risk Solutions in the early 2000s
- Released as open-source in 2011
- Configurations to support both parallel batch data processing and online query applications using indexed data files
- Enterprise Control Language (ECL)

High Performance Computing Cluster
HPCC

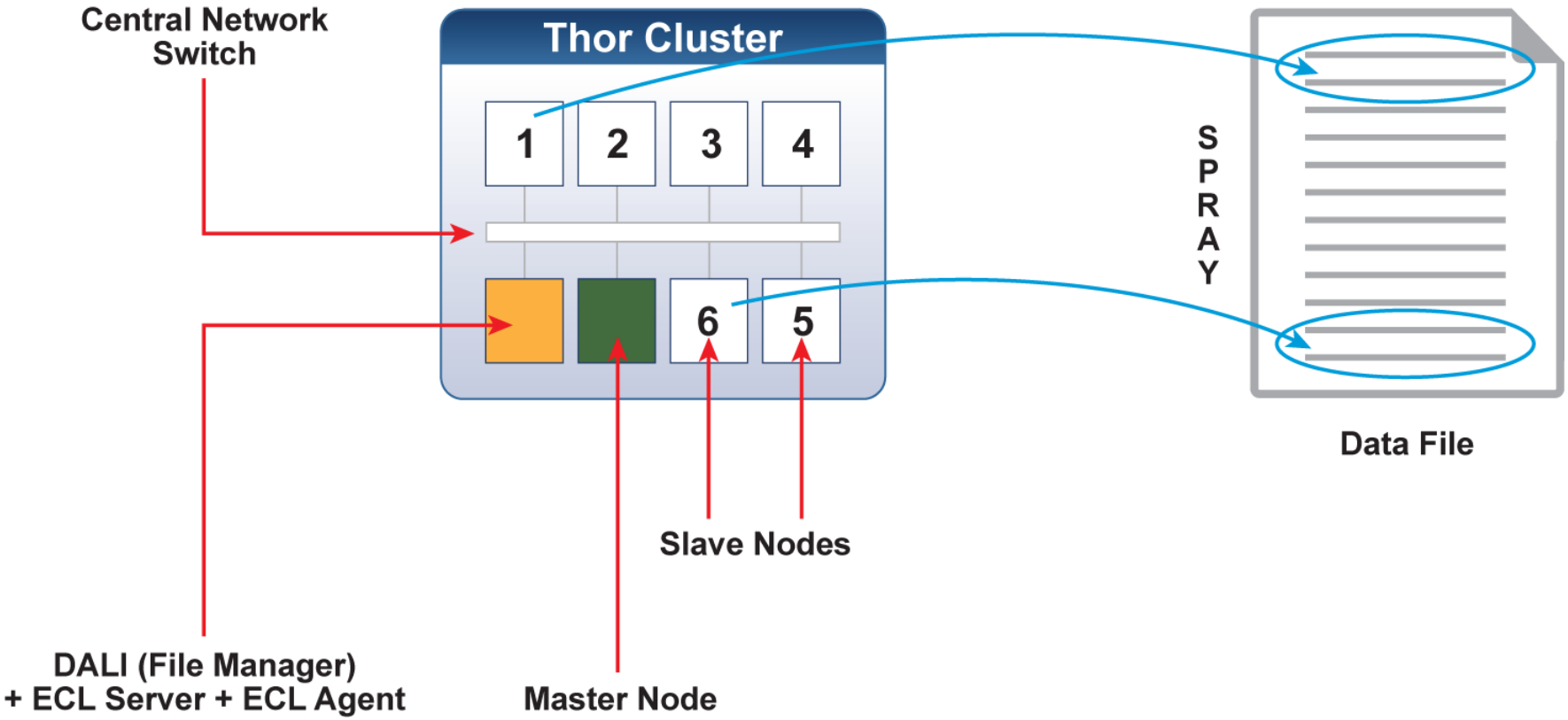
- Consists of two processing environments
  - Thor (Data Refinery)
  - Roxie (Data Delivery Engine)
- Software and Middleware
- Enterprise Control Language (ECL)
- Client Interfaces
  - ECL Watch
  - ECL IDE
  - ECL Plugin for Eclipse


Client Interfaces
End-User Services
Query Builder
ECL Direct (via SOAP)
WS-ECL (via browser)
ECL Watch (via browser)
AMT
DFU Plus
AMT Plus
ECL Plus
Management Tools
ECL Watch (via browser)
Network Managment Service (Syslog)
External Communications Layer
Enterprise Services Platform (ESP)
(enables access to ECL execution layer via HTTP or SOAP interfaces)
ECL Watch Service
WS-ECL Service
ECL Direct Service
ECL Middleware Layer
DALI
The System Data Store
Work Units
DLL Server
Environment Configuration
Job Queue
Name Services DFU Data Store
ECL Attribute Repository
ECL Server
Job Server
DFU Distributed File Utility
FT Slave
Dali Servix
Sasha
ECL Agent(s)
Data Delivery Engine
Auxiliary Components
LDAP (optional)
Landing Zone
Supercomputer Layer
Data Refinery (THOR)
Rapid Data Delivery Engine (Roxie)

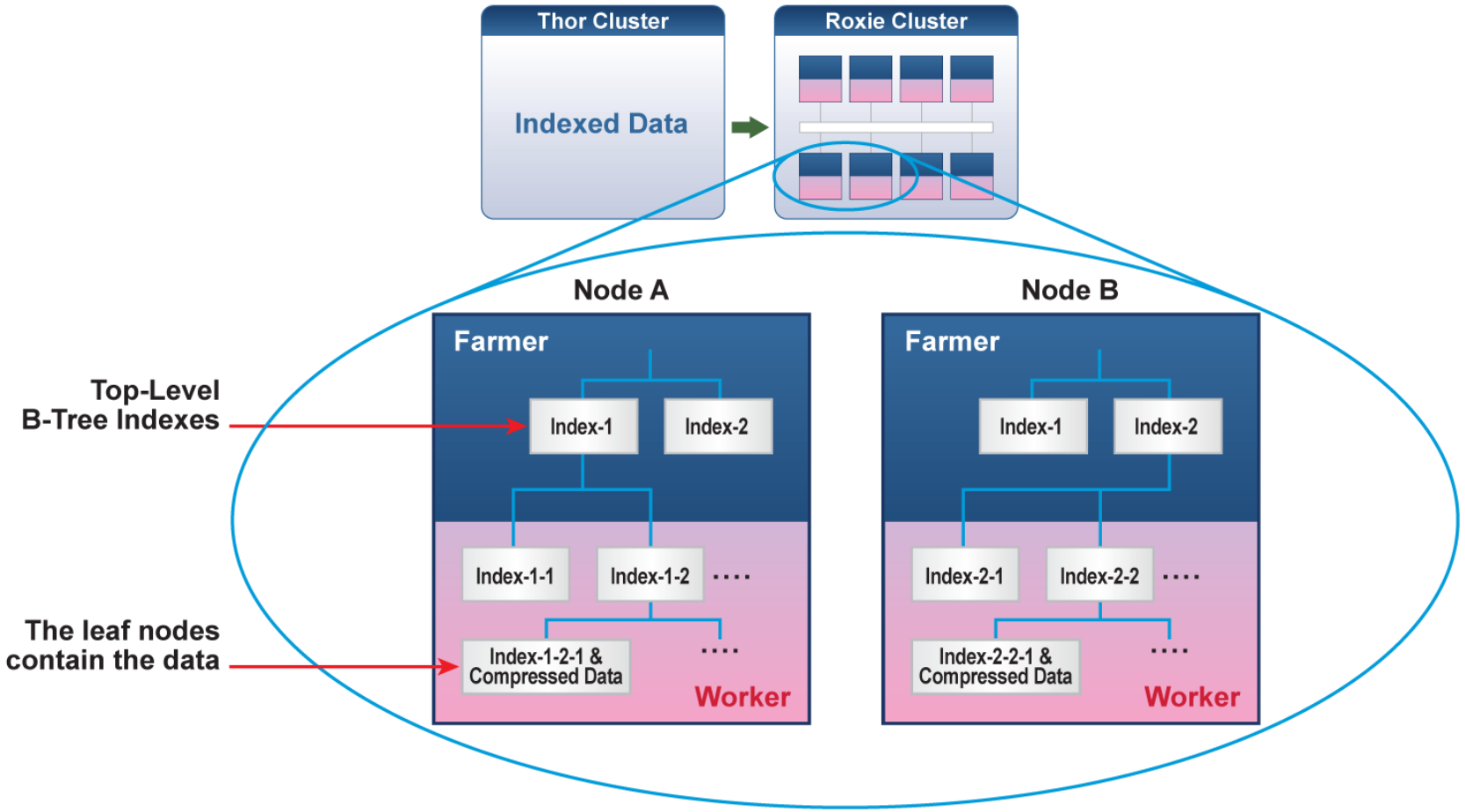
HPCC Systems Architecture



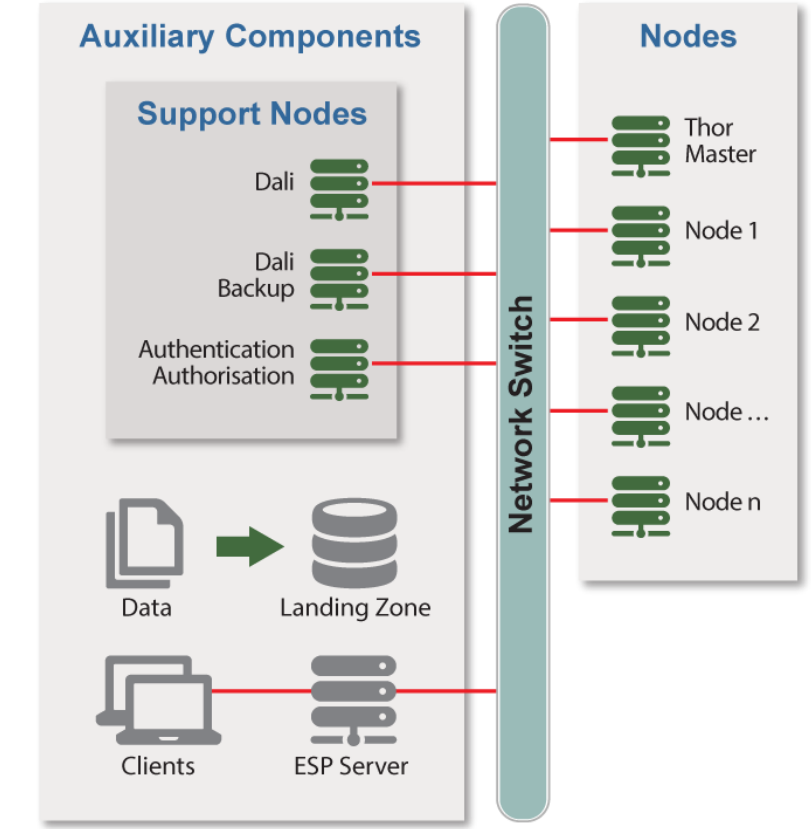
Thor Logical Architecture



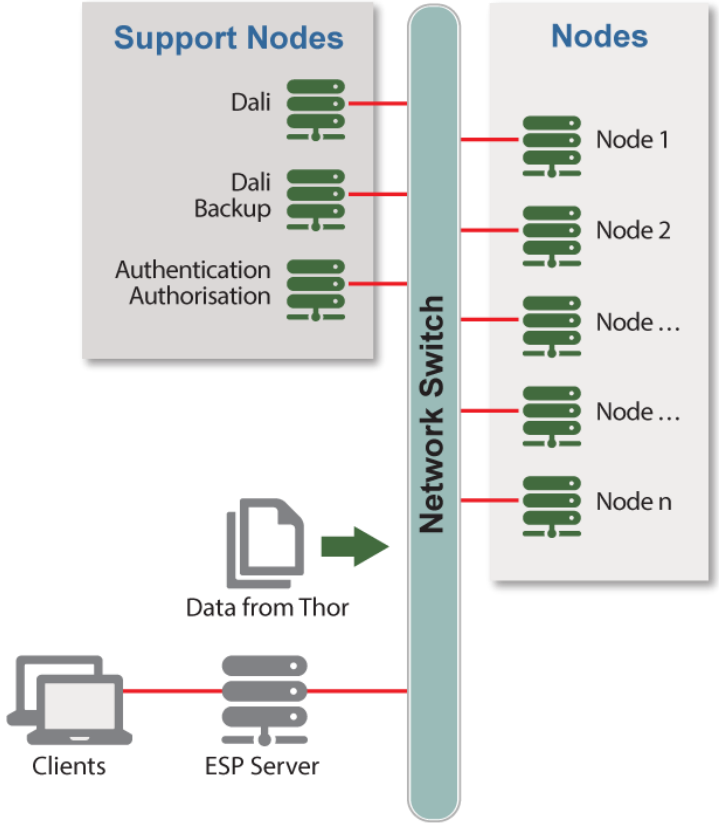
Roxie Logical Architecture



Thor/Roxie Physical Architecture



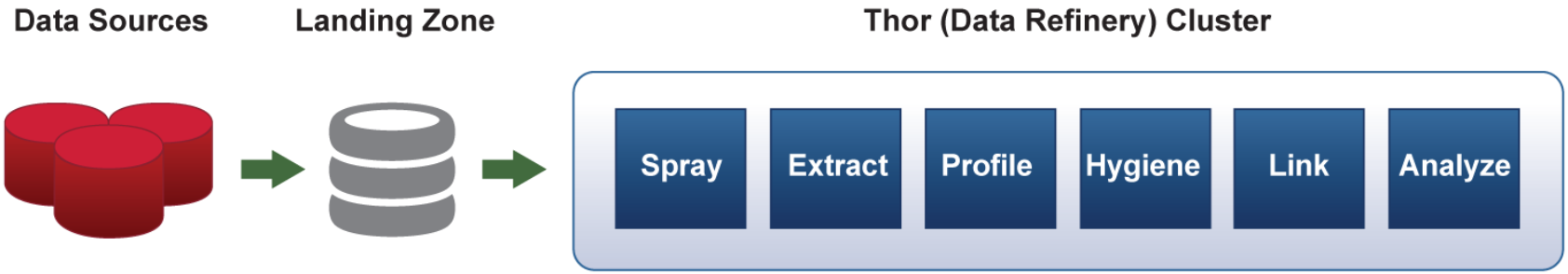
Thor
(Batch Job Execution Engine + DFS)
Physical Layout Schematic Diagram



Roxie
(Rapid Data Delivery Engine)
Physical Layout Schematic Diagram

CLEMSON UNIVERSITY

Data Refinery Process



Enterprise Control Language (ECL)

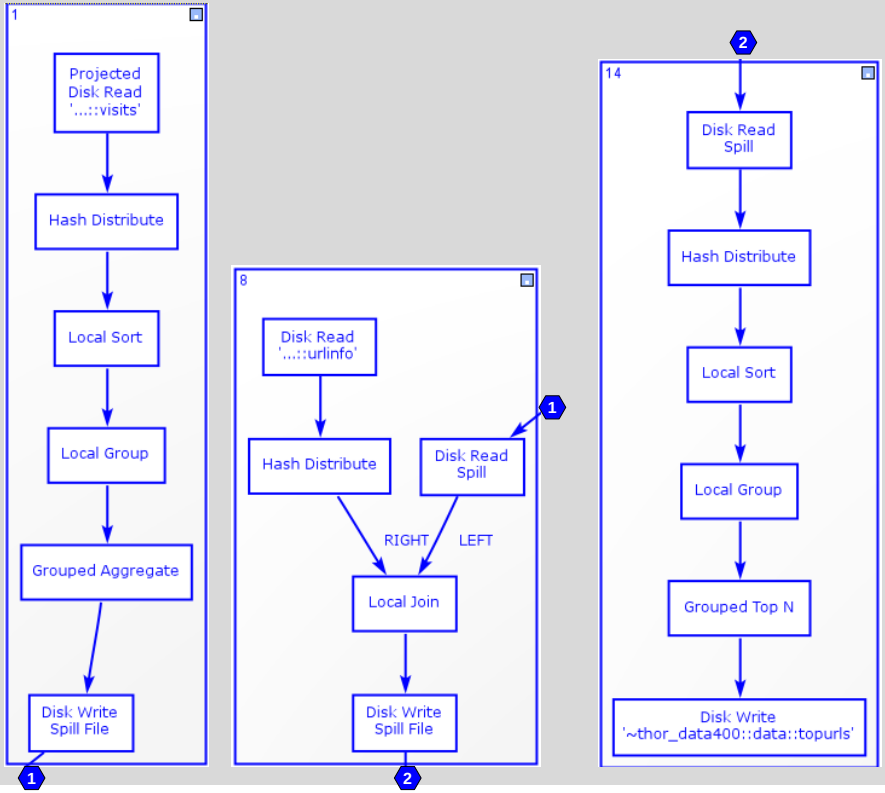
- Declarative programming language
- Powerful
- Extensible
- Implicitly parallel
- Maintainable
- Homogeneous

```
// 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)

- 1 line of ECL is roughly equivalent to 120 lines of C++
- ECL primitives that act upon datasets include:
  - SORT,
  - ROLLUP,
  - DEDUP,
  - ITERATE,
  - PROJECT,
  - JOIN,
  - NORMALIZE,
  - DENORMALIZE,
  - PARSE,
  - DISTRIBUTE
  - Etc.

```
Go                                   Queue: dev_eclserver_  Cluster: thor400_88_de  More
1  // Sample ECL Code
2  layout_visits := RECORD string user; string url; string time; END;
3  visits := DATASET('~thor_data400::data::visits',layout_visits,FLAT);
4
5  layout_urlInfo := RECORD string url; string category; string pRank; END;
6  urlInfo := DATASET('~thor_data400::data::urlinfo',layout_urlInfo,FLAT);
7
8  // Distribute Visits by URL, Count vists by URL
9  layout_visitCounts := RECORD visits.url; visits_cnt := COUNT(GROUP); END;
10 visitCounts := TABLE(DISTRIBUTE(visits,HASH(url)),layout_visitCounts,url,LOCAL);
11
12 // Distribute Category by URL, Join category to URLs
13 visitCountsCat := JOIN(visitCounts,DISTRIBUTE(urlInfo,HASH(url)),LEFT.URL=RIGHT.URL,LOCAL);
14
15 // Distribute and Group by Category, Output top 10 URLs for each category
16 topUrls := TOPN(GROUP(DISTRIBUTE(visitCountsCat,HASH(category)),category,ALL,LOCAL),10, -visits_cnt);
17 OUTPUT(topUrls,,'~thor_data400::data::topurls',OVERWRITE);
```



CLEMSON UNIVERSITY

Enterprise Control Language (ECL)

- 1 line of ECL is roughly equivalent to 120 lines of C++
- ECL primitives that act upon datasets include: SORT, ROLLUP, DEDUP, ITERATE, PROJECT, JOIN, NORMALIZE, DENORMALIZE, PARSE, CHOSEN, ENTH, TOPN, DISTRIBUTE

Enterprise Control Language (ECL)

```
// First declare a dataset with one column containing a list of strings
// Datasets can also be binary, CSV, XML or externally defined structures

D := DATASET([{'ECL'},{'Declarative'},{'Data'},{'Centric'},{'Programming'},{'Language'}],{STRING Value;});
SD := SORT(D,Value);
output(SD)
```

- OUTPUT(SD)
  - What is an SD?
- SD := SORT(D,Value);
  - SD is a D that has been sorted by ‘Value’
  - What is a D?
- D := DATASET([{'ECL'},{'Declarative'},{'Data'},{'Centric'},{'Programming'},{'Language'}],{STRING Value;});
  - D is a dataset with one column labeled ‘Value’ and containing the following list of data.

HPCC vs HADOOP

Hadoop Name/Term	ECL equivalent	Comments
MAPing within the MAPper	PROJECT/TRANSFORM	Takes a record and converts to a different format; in the Hadoop case the conversion is into a key-value pair
SHUFFLE (Phase 1)	DISTRIBUTE(,HASH(KeyValue))	The records from the mapper are distributed dependent upon the KEY value
SHUFFLE (Phase 2)	SORT(,LOCAL)	The records arriving at a particular reducer are sorted into KEY order
REDUCE	ROLLUP(,Key,LOCAL)	The records for a particular KEY value are now combined together

HPCC vs HADOOP

Feature	HPCC	Hadoop
Distributed File System	Thor DFS/Roxie DFS	Hadoop DFS
Database Capability	native to DFS	HBase
Data Warehouse	Roxie	Hive
Programming Languages	ECL	Java/Pig

Using HPCC Systems to Manage Academic Data

CLEMSON UNIVERSITY

Managing the Academic Data Lifecycle

- Single platform for everything
- Supports Parallelization
- Supports Scalability
- Embedded support for external languages and libraries for specialized capability

Managing the Academic Data Lifecycle

- Data required for scholarly data research
  - Many different sources
  - Various formats

- Aggregating these sources into a cohesive structure requires a tool that supports a data-intensive approach for:
  - Preprocessing
  - Integration
  - Analysis

Managing the Academic Data Lifecycle

Managing the Academic Data Lifecycle

Detailed list of articles by discipline with abstracts, references, and disambiguated authors. (**XML**)

Enrollment **(multi-sheet Excel)**
Financial **(multi-sheet Excel)**
Faculty **(multi-sheet Excel)**

Institutional Information with Carnegie's Research Classifications **(multi-sheet Excel)**

Detailed Award **(XML)**
Federal Funding **(tab-delimited)**
Expenditures **(tab-delimited)**
Institution Patent **(Excel)**
Degrees Conferred **(Excel)**

Institutions from States with EPSCoR status **(tab-delimited)**

NIH Award Data **(CSV/Excel)**

Top500 Supercomputer affiliated with academic institutions **(XML)**

CLEMSON UNIVERSITY BDSLab LexisNexis™ CLEMSON UNIVERSITY

Managing the Academic Data Lifecycle
Ingesting Tabular Data

```
Ipeds_Layout := RECORD

   STRING   UNITID;
   STRING   INSTNM;
   STRING   ADDR;
   STRING   CITY;
   STRING   STABBR;
   STRING   ZIP;
   STRING   CONTROL;
   STRING   HBCU;
   STRING   MEDICAL;
   STRING   TRIBAL;
   STRING   CCBASIC;

END;

EXPORT Ipeds := DATASET('ipeds::institutional_
characteristics', Ipeds_Layout, CSV(HEADING(1)));
```

```
IMPORT $.IPEDS;

COUNT IPEDS;
OUTPUT(IPEDS, {unitid, instnm}, NAMED('ID_NAME'));
```

Managing the Academic Data Lifecycle
Ingesting XML Data

```
ProgramsXML:= RECORD
  UNICODE program {XPATH('')};
END;


AwardXML:= RECORD

  UNICODE awardNumber         {XPATH('AwardNumber')};
  UNICODE amountToDate        {XPATH('AwardedAmountToDate')};
  UNICODE title               {XPATH('Title')};
  UNICODE awardInstrument     {XPATH('AwardInstrument')};
  UNICODE startDate           {XPATH('StartDate')};
  UNICODE expirationDate      {XPATH('ExpirationDate')};
  UNICODE directorate         {XPATH('NSFDirectorate')};
  UNICODE nsfOrganization     {XPATH('NsfOrganization')};
  DATASET (ProgramsXML) programs {XPATH('Program')};
  UNICODE abstract            {XPATH('Abstract')};

END;

EXPORT NsfAwards := DATASET('nsfdata::nsfxml',AwardXML,XML('AwardsList/Award',NOROOT));
```

Managing the Academic Data Lifecycle
Cleaning Data

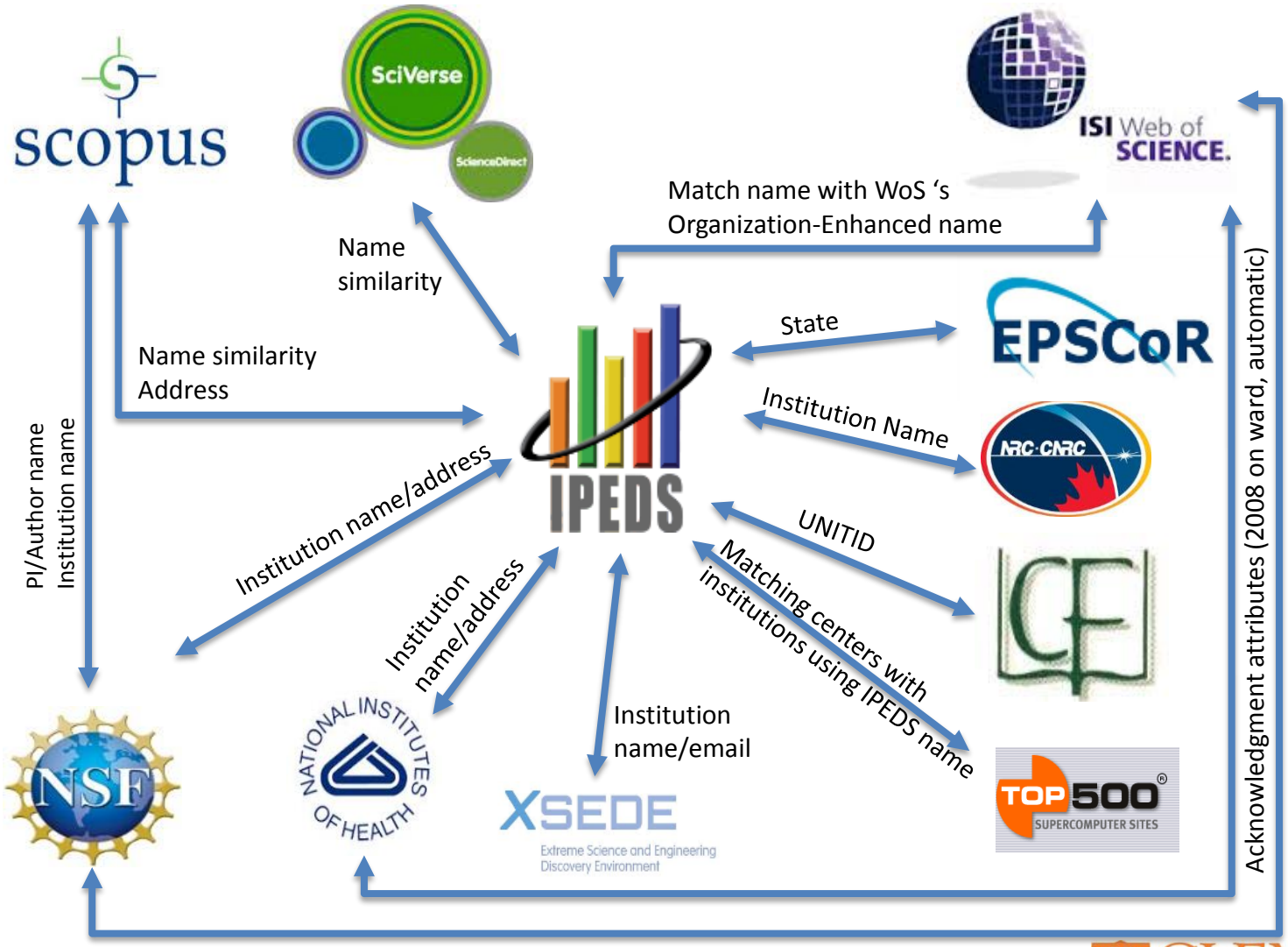
```
IMPORT Std;

EXPORT CleanForTokens(STRING s) := FUNCTION
  sRestrictChars       :=  ' '+REGEXREPLACE('[^- A-Z0-9\']',Std.Str.ToUpperCase(s),' ')+' ';
  sStripPunctEnds      :=  REGEXREPLACE('( -)|(- )|( \')|(\' )',sRestrictChars,' ');
  sRemoveNumberOnly    :=  REGEXREPLACE(' [-0-9\']+(?=[ ])',sStripPunctEnds,'');
  sCompressSpaces      :=  REGEXREPLACE('[ ]+',sRemoveNumberOnly,' ');
  sNormalizePossessives:= REGEXREPLACE('\'S ',sCompressSpaces,' ');
  sSplitContraction01  :=  REGEXREPLACE('\'RE ',sNormalizePossessives,' ARE ');
  sSplitContraction02  :=  REGEXREPLACE('\'LL ',sSplitContraction01,' WILL ');
  sSplitContraction03  :=  REGEXREPLACE(' I\'M ',sSplitContraction02,' I AM ');
  RETURN sSplitContraction03;
END;
```

```
IMPORT $.Ipeds;
IMPORT $.CleanForTokens;

$.Ipeds Standardize($.Ipeds Le) := TRANSFORM

   SELF.INSTNM := $.CleanForTokens(Le.INSTNM);
   SELF.ADDR   := $.CleanForTokens(Le.ADDR);
   SELF.CITY   := $.CleanForTokens(Le.CITY);
   SELF.STABBR := $.CleanForTokens(Le.STABBR);
   SELF.ZIP    := Le.ZIP[..5];
   SELF        := Le;

END;

EXPORT Standard_IPEDS := PROJECT($.Ipeds,Standardize(LEFT));
```

Managing the Academic Data Lifecycle
Linking Data

```
IMPORT $;

NsfIpedsRec := RECORD

      STRING unitid;
      UNSIGNED awardnumber;

END;

NsfIpedsRec JoinThem($.Awards Le, $.IPEDS Ri) := TRANSFORM

      SELF.unitid := Le.unitid;
      SELF.awardnumber := Ri.awardnumber;

END;
EXPORT Nsfipeds := JOIN($.IPEDS,$.Awards, LEFT.name = RIGHT.institution, JoinThem(LEFT,RIGHT));
```

Managing the Academic Data Lifecycle
Examples of Scholarly Data Links



CLEMSON UNIVERSITY BDSLab LexisNexis™ CLEMSON UNIVERSITY