Correlation Using Pair-wise Combinations of Multiple Data Sources and Dimensions at Ultra-Large Scales

October 24-28, 2011

Jonathan Larson, Kathleen Lossau, Dale Walsh

Potomac Fusion, Inc.
1515 S. Capital of Texas, Ste 500
Austin, TX 78746

The MITRE Corporation
7515 Colshire Drive
McLean, VA 20122

ABSTRACT

Large scale data poses significant challenges in locality, transmission, and partitioning of data. Using open source cloud technologies such as HDFS, Map/Reduce, and Hive processing massive correlation matrices is now becoming a tractable option for today's intelligence systems. The cloud allows us to distribute the computational and storage load into a parallel and scalable environment.

Data warehousing technologies have long relied on denormalization techniques to enable quick analysis. For these denormalizations, a dimensional model is employed and a 'projection' is built out on all or a subset of the stored features. Until large scale computing, it has not been practical to implement large scale pair-wise computations due to the massive disk space required to pivot on the many features (dimensions) in the data. Most systems today use a well informed subject matter expert to define a model before-hand to restrict what dimensions are combined. This process may restrict and ultimately hide important relationships between dimensions of interest. Instead we have implemented a system that calculates all dimensional combinations in a distributed architecture. This effectively enables an automated correlation capability that can uncover non-obvious correlations that would otherwise be left undiscovered.

Using these techniques alongside existing feature extraction / enrichment analytics helps to create a flexible platform for the development and implementation of new capabilities that work at scale. In this manner, the raw data is used to build analytic projections recursively, so that each new projection is built from an existing set of projections.

Analytics can be built to calculate combinatorics en masse by pair-wise combinations of many different data sources and dimensions on a large scale. The initial results of these analytics, which range into the trillions of comparisons, have provided groundbreaking results that prove the worth of applying massive computational power to discover previously hidden patterns.

1.0 Introduction

A critical step in intelligence analysis is looking for previously undiscovered patterns and anomalies in the data. Data analysis in ultra large data stores is a targeted process that leverages a subject matter expert's (SME) familiarity with the data. Using this expertise, a few features (dimensions) are chosen for a suggested analysis. This modeling process also creates a bias because the data is only analyzed within the context of the SME's understanding. While this is an effective approach, it will miss patterns that a computationally aggressive approach would have detected. This is where the massive computational capability of a cloud can be put to good use. Instead of comparing a few pre-defined combinations of dimensions, we use a distribution technique that facilitations discovering patterns within the data without introducing any modeling bias.

For example: we may detect an increase in significant hostile activities in areas where there are blue forces and an increase in the availability of a specific type of food. This, previously hidden pattern, quickly emerges from the pair-wise computations. It could be certain cultures eat specific foods and their presence can be discovered by looking for increase sales of that food. The importance and reason for the correlation is something that requires an analyst to determine (i.e., is the correlation useful?). In this simple example, the increase in reporting could be due to the increase in blue forces. An additional calculation to determine the pattern in blue forces to significant activity counts could easily remove this particular bias. Finding the hidden patterns enables the analysts to focus on the importance and relevancy of the correlations rather than trying to find hidden patterns.

We can leverage simple techniques, such a the classic correlation co-efficient and apply it across the data at a massive scale (e.g. trillions of comparisons). To implement this capability we rely on several cloud-based data warehousing techniques paired with batch processing operations to distribute the analytic load. This allows for the data de-normalization (a massively IO intensive process) to be distributed across the computers within the cloud. This paper will discuss the architectural approach we used to implement this capability.

The benefit of these techniques is to provide a rich basis for analysis of large-scale, heterogeneous graphs. The Army has begun to implement a concept called the Global Graph, which today provides for an interrelated graph structure between over a dozen different classes of entity type (people, facilities, events, pieces of equipment, societal elements, etc.). As it is initially fused, the Global Graph will only contain entity interrelationships that are observed/reported (two cell phones communicate); discovery of "closer" relationships between entities (two people talking) can best be done by the pair-wise analytic techniques described in this paper.

In addition, the use of a framework will have significant advantages:

- Robustness – ability to work at scale in a cloud-type environment by use of map/reduce and other techniques
- Adaptability – ability to provide new and unpredicted "feature projections" that will allow analysts to "cast new nets" for interrelationships beyond those mined for in previous analyses
- Modularity – ability to bring new analytic results across a non-static dataspace, and the ability to persist those findings back into an architecture for provisioning and dissemination (and follow-on analysis)

In short, these techniques provide the ability to "see what might otherwise not be seen". Even against an adaptive opponent, the ability to see with deep multi-dimensionality can allow discovery of trends and traits, including those the opponent may not even realize they have.

2.0 Large Data Analytic Framework

Large scale data pose significant challenges in locality, transmission, partitioning of data and processing analytics. Many analytics involve batch processing and staging of the data (i.e., not real-time). However, it is increasingly important to be able to operate in a real-time analytic framework. Emerging technologies like BRISK (DataStax's Cassendra/Hive implementation) support this type of processing and need to be further explored.

Raw data is transformed and indexed into denormalized data stores to build analytic projections recursively, such that each new projection is built off of an existing set of projections. Effectively, this creates an ad-hoc development environment, in which many analytics can be tested forensically across historical data, while also enabling a pipeline for pre-defined analytics that are optimized for real-time operations and feedback.

Drawing on experience and techniques that have been implemented and refined in various DoD intelligence projects, analytics can be built to calculate combinatorics en masse by pair-wise combinations of many different data sources and dimensions on a large scale. This effectively formulates an automated correlation capability that can uncover otherwise undiscovered correlations. The initial results of these analytics, which range into the trillions of comparisons, have provided groundbreaking results that prove the worth of applying massive computational power to discover previously hidden patterns. We will describe several classes of statistical anomaly detection algorithms, aggregate analysis, and rasterization techniques that are optimized to operate on a large scale and can be applied to new problem sets. These analytics are given a final polish by using a set of multi-dimensional visualization utilities to fine-tune the analytic output.

The basic framework leverages Hadoop / HDFS / Hive as a base mechanism for storing the data structures (Apache Pig is another option to replace Hive, but Hive is well suited for its strong support of storing schemas alongside the data). Using these constructs, raw data is stored in a basic table within Hive (which is then automatically distributed across HDFS). Transformations to that raw data are then built up on demand through Map / Reduce batch processes to generate a new de-normalized view of the data (an index). All of this is normal cloud data warehousing up to this point. The next optimization step, however, is where the huge performance gains can be made. Once all of the indices have been built up, some will be elevated into Map / Reduce job's memory using Hive's compatibility with Hadoop streaming. This optimization technique speeds up subsequent index operations dramatically – especially for N^2 operations wherein all of the data must be analyzed against itself (as in a correlation co-efficient matrix calculation).

The next step is to take what we have developed and put it into a large data analytic framework to enable additional data sources and analytics to be added, developing and sharing new projections and visualizations of the data.

2.1. Multidimensional Databases

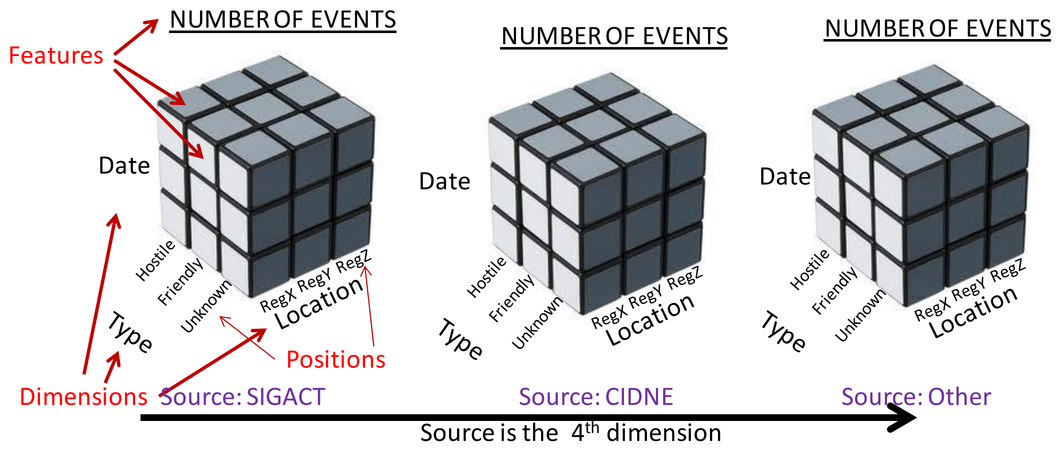
A multidimensional database (MDD) is a repository optimized for online analytical processing (OLAP) and is structured to answer questions related to patterns and trends in the data. An MDD is used to aggregate or summarize the data so that patterns and anomalies in the patterns can be more easily extracted. A multidimensional view of the data provides the ability to do multi-dimensional analysis; multi-dimensional database are best used when dealing with highly interrelated dataset types (i.e., not sparse datasets), even when those inter-relationships are not well understood.

The core of any OLAP system is an MDD or hypercube. It consists of facts (features or measures) that are categorized by *dimensions*. The features are derived from the rows in the fact table and the dimensions (aggregations) are defined from the dimension tables. An MDD following a standard abstract representation of a star schema creates two types of tables: Fact and Dimension.

- Fact tables have many rows (potentially millions) and contain the values or measurements.
- Dimension tables contain the meta-data and are used to define the aggregations or summarizations of the data.

Event	Type	Source	Location
Ev1	Hostile	SIGACT	regionx
EV2	Friendly	CIDNE	regionx
EV3	Hostile	Other	regiony

- Find number of events in a region or by source or by type
- Easily dd new dimensions



In order to address the multi-precision detection requirement we will use the multidimensional database to derive rollups of multiple dimensions. These abstractions, clusters, and summarizations of information are representations of the same information at different scales. Examples of different precision levels can be represented as follows:

- **Time** – hour, day, week, month, etc.
- **Location** – point, region, city, state, etc.
- **Group** – individual, group, organization
- **Activity** – single event, set of events, sequence of events

Indices are built for each level of precision as they are needed in further dependent analytics. The highest level of precision is always available from the raw data storage.

2.2. Planes of Enrichment and Data Indexing

With the availability of massive hardware and disk space available on the cloud, the option to denormalize data en masse becomes a very attractive prospect. Thus, instead of just storing the raw data tables and dynamically calculating analytics on the fly, everything is serialized into staged indices to optimize retrieval. The result are trees of derivative indices or "planes of enrichment" that transform, enrich, and otherwise manipulate the data into a new form. Often, this means aggregation to a lower precision (IE: if the timestamps of a records are to the millisecond, we might aggregate all records on the same day instead), which greatly reduces the computational load for specific analytics. Applying this principal all of the way through the system, indices may be built all the way up to support specific visualization API calls as well.

These de-normalized planes of enrichment don't need to be restricted just to the output of MDD. They can also be the product of graph manipulations as well. Thus, an MDD (which is inherently bad at graph style operations) might project into a graph format for a relationship based analytic. Similarly, a graph (which is not well suited for efficient aggregate analytics) might project back out into a tabular MDD style table. Thus, the two different data structures (each with its own specialty in terms of optimized execution style) can use projections as an intermediary staging point between analytic stages.

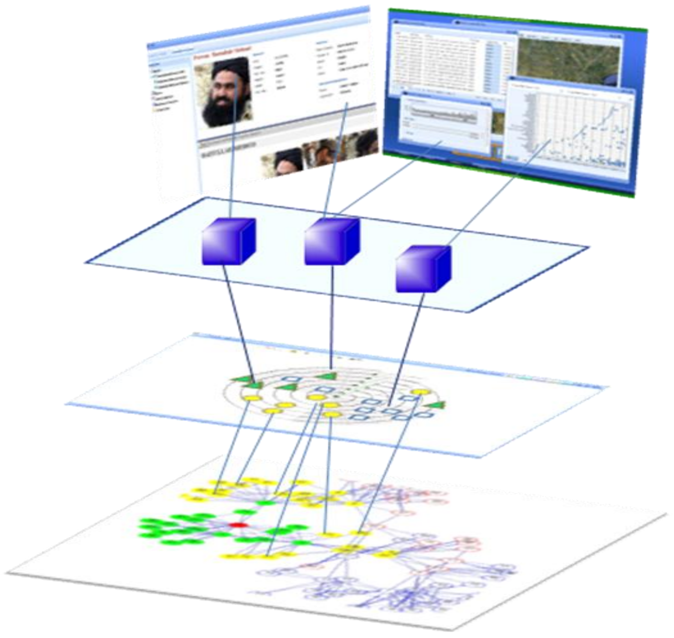


Figure 1: An MDD with Planes of Enrichment

2.3. Pairwise Combinatorics

One kind of analytic function is analytic combinatorics [Flajolet, 2008]. The combinatorics use combinatorical classes to generate functions that operate over the data to discover correlations and patterns. We have explored how using pairwise combinatorics can help discover patterns within the data. This class of analytics often led to intractable computation time, but can now be addressed thanks to advances in cloud computing.

One example of a typical function to apply combinatorically is the correlation co-efficient. This is an important mathematical construct used when comparing two features [Buda, 2010]. This requires a vector for each feature and often includes a lag variable, meaning many combinations that need to be computed. Using even simple correlation algorithms such as Pearson's correlation co-efficient, we are able to do a measure of the correlation (linear dependence) between pairwise variables that is permuted across many dimensions. This results in innumerable combinations and results. However, as each result is calculated, it is filtered to retain only those combinations that produce a correlation strength of interest. IE, analyzing trillions of pairwise combinations and comparisons to get a few thousand good results. However, when we do get those ~.6% of results that come back with a strong correlation coefficient, we know that there is a pattern worth exploring. Instead of having a SME specify that two dimensions are related in a certain way, the data is telling the SME what dimensions are actually related allowing the SME to spend his time exploring the relevancy of the correlation to the mission at hand.

Taking a cue from the financial markets and their approach to arbitrage, we enhance this process using Pearson sliding windows[Bäcker and Cassenaer] . Using these, you can watch things go into correlation and then leave correlation - again providing extremely interesting insights. We used this method to detect changes between two localized areas market places selling the same good. When the correlations deviated, there were other detected differences in the stability of the local area. This type of combinatorial analytic relies on “brute force”: it is so computationally intensive that it requires a cloud and extensive hardware, especially as we cross-correlate across data sources.

The “low-hanging fruit” of covariance / correlation co-efficient has proved very useful just by itself and applying it in mass computation. The results are often easy for our military analysts to explain and use.

As a very simple example, suppose you have a stream of documents coming in for a specific region with a built in alerts system that detects documents with specific keywords. You have 5000 keywords that will raise an alert. Each keyword thus has a trend over time for the number of documents occurrences. Running a simple correlation coefficient function between all 5000 keyword trends over time with many lag window settings could very quickly identify leading / lagging indicators. For example, you might find that a bomb goes off 5 days after a particular name shows up in town. While trivial to calculate, the combinatorics of such a system (especially with many different lag variable settings) becomes intractable on a single system.

We have also looked at other derivative purpose-built analytics such as a variety of other distance metrics between feature vectors. Taking into account all of these distance metric calculations, it is then simple to construct a dimensional visualization to view the top relationship pairings.

Complicating matters further is the issue of combinatorics of the actual features. Using readily available commodity price data, we produced the following example:

One year of data included approximately 300K unique data sets. Pairwise correlations between all dimensions on a sliding 30 day window required hundreds of millions of comparisons.

Strongly correlated pairs were used to determine networks between certain cities and identify “stability systems”. Stability systems, where stimulating one part (a market) impacted other parts, were found to be influenced by events “outside” the system. For example, the potato markets between two cities maintained a strong correlation in price until border activity (unclassified SIGACT report) happened on a road between the two cities.

Stability system behavior can, under some circumstances, also be predictive in nature. Using combinatorics on commodity price data clearly demonstrates the potential value of applying such an approach to substantially increase the value of data collection.

Not all problems are naturally parallelizable. This particular problem is inherently NOT a parallelizable due to the need for data locality of all data across all nodes. Since locality of data across all nodes is a requirement, a copy of all the data (or shards of the data) across all nodes, but we elevate all of the data (or the shards of it) entirely in memory. With this in place, we then use Map / Reduce to feed in the combinations that each node is to calculate for the correlation matrix between variables. By doing this, we can effectively distribute the workload of the calculations across the nodes. This is particularly effective for problems that have many features, but fewer rows of data on which to run the correlation (such that the data can be elevated into memory).

2.4. Combining Analytics over Real-Time Data

Cloud computing requires large scale data ingestion combined with real-time access to and processing of that data. However, most analytics operate by batch processing (e.g., MapReduce). Thus, current cloud analytic processing works like this: first, analytics that operate using a batch mechanism stage the data; they then cast multiple projections (views) of the data on which more real-time analytics can operate.

We need to develop a framework in which analytic processing, real-time ingestion, and real-time user access can occur seamlessly. Analytics that operate in this batch mechanism stage the data. They cast multiple projections (views) of the data where more real time analytics operate. This is the cloud analytic processing environment we have today. HoweverTo accomplish this, we are looking at emerging technologies such as Brisk (Apache’s Cassendra and HIVE real-time processing) environments.

The LDAF architecture is particularly suitable to processing massive amounts of intelligence data. There are some unique challenges to this domain, including large-scale streaming data – especially as it relates to data locality and transmission.

Data management, including representation, indexing, and retrieval is an essential part of the overall framework/architecture. Also important is the ability to deliver an ad-hoc development environment (e.g., Hypercube / forensic) along with a real-time delivery capability from within the cloud infrastructure. Building and indices and projections for layers and layers of analytics are essential to performing higher-order analysis on the data (producing patterns and trends). These dimensional projections are called "hyperslices" of the cube. Auto-normalization techniques of the data as well as index / hyperslice management are essential LDAF components to support both a forensic and real-time capability.

The use of common hardware and software, as well as adherence to open standards, is fundamental to developing long term solutions for the DoD community. PFI has built a powerful analytic development environment and useful analytics. In addition, exploring emerging concepts like Brisk has given us the technological expertise necessary to integrate analytics into a real-time environment. We believe that one solution is to use a cloud deployment of a multi-dimensional manipulable space along with real time ingestion of large amounts of data. Data locality, partitioning, and performance tuning are all essential to having both real-time functions and analytics operate in the same space. Aggressive indexing of data before distribution and elevation into memory for sharding is a key performance enhancer.

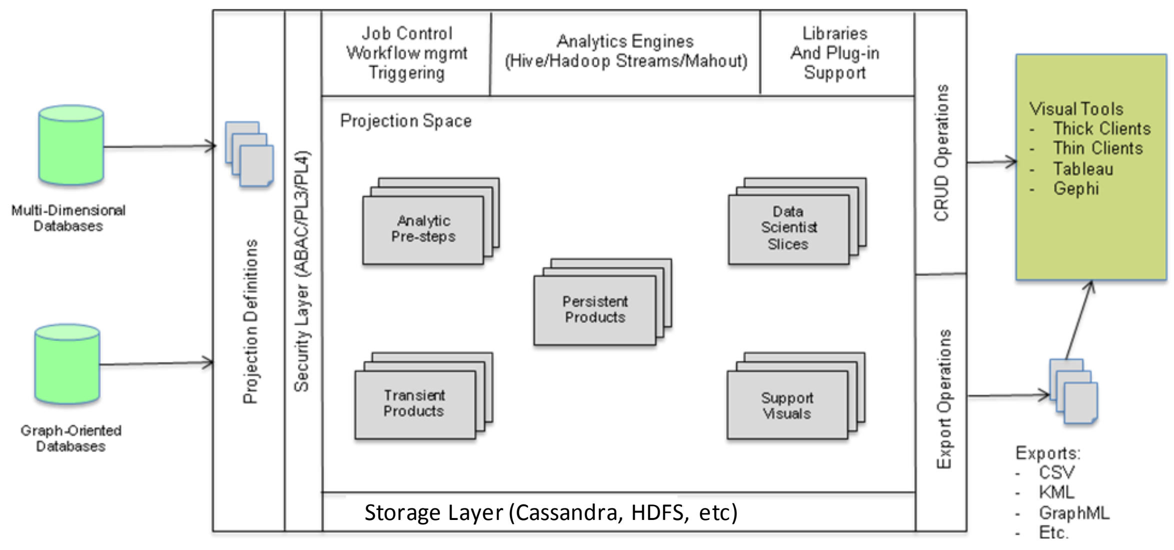


Figure 2: LDAF Architecture Components

The LDAF architecture allows for real-time ingestion, export, and CRUD operations on the data, while analytic engines can operate in batch or streaming process to create additional projections of data – higher-order analytics.

3.0 Future Issues

One of the biggest challenges that we will need to address in our DoD MDD is how to map the data types to a Semantic Meta-Model, and thus to the DoD cloud infrastructure, in order to enable analytics to operate. Many of the analytics are focused on summarizing higher order data: for example, the collection of low-precision statistical measurements and observations regarding regions, rather than specific entities. This information can be represented with the defined MDD structure with no problems, but we need to think carefully about the planes of enrichment that are built from these structures. In particular, we must consider the performance impact of certain schema layouts for what are ultimately statistical roll-ups of the facts and dimensions. Ultimately, the hardest task will be to map the ground truth that these analytics refer to into the existing semantic meta model (schema) constructs.

In conclusion, we have made the argument that feature correlation across multiple data sources, at ultra-large scales, is a tractable problem when performed in a framework context on scalable architectures. The ability to do such analyses across large dataspaces of heterogeneous data (such as a Global Graph) can bring robust, adaptable, and modular capabilities to the newest and most aggressive types of modern intelligence analysis problems.

4.0 References

- Gray, Jim; Chaudhuri, Surajit; Layman, Andrew; Reichart, Hamid; Venkatrao; Pellow; Pirahesh (1997). "Data Cube: {A} Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals". *J. Data Mining and Knowledge Discovery* **1** (1): 29–53.
- Bach Pedersen, Torben; S. Jensen, Christian (December 2001). "Multidimensional Database Technology". *Distributed Systems Online* (IEEE): 40–46. ISSN 0018-9162
- Ilkka Kivim, Krista Lagus, Ilari T. Nieminen, Jaakko J, and Timo Honkela (2010) "Using correlation dimension for analysing text data". In Proceedings of the 20th international conference on Artificial neural networks: Part I (ICANN'10), *368-37*
- O'Brien & Marakas, Introduction to Information Management Systems, 2009, 177-178
- Williams, C., Garza, V.R., Tucker, S, Marcus, A.M. (1994, January 24). Multidimensional models boost viewing options. InfoWorld, 16(4)
- Philippe Flajolet and Robert Sedgewick, Analytic Combinatorics, Cambridge University Press, 2008, ISBN 0521898064
- DataStax, Real-Time and Analytic Data Management for the Enterprise, 2011
- Lakshman, Malik, Cassandra - A Decentralized Structured Storage System, 2009
- A. Buda and A.Jarynowski (2010) *Life-time of correlations and its applications vol.1*, Wydawnictwo Niezalezne: 5–21, December 2010, ISBN 978-83-915272-9-0
- Alex Bäcker and Stijn Cassenaer, Asymmetric sliding-window cross-correlation, www.alexbacker.com/CNS/pub/Asymmetricslidingwindowx-correlation.pdf