How to build a multi PB Data Hub in less than 6 months

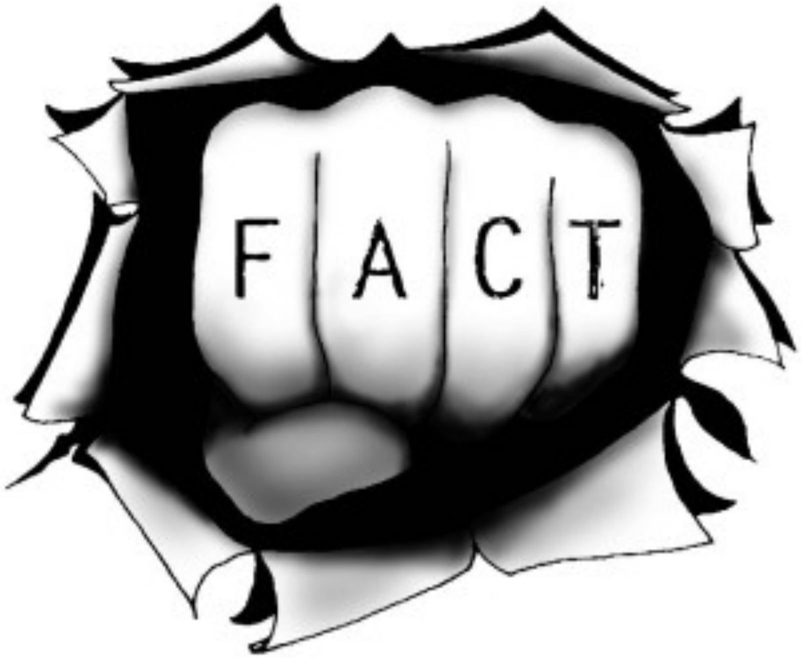
Oz Levi, CTO, MatrixBI

Gmail

Google Search   I'm Feeling Lucky

Google.co.il offered in: עברית العربية

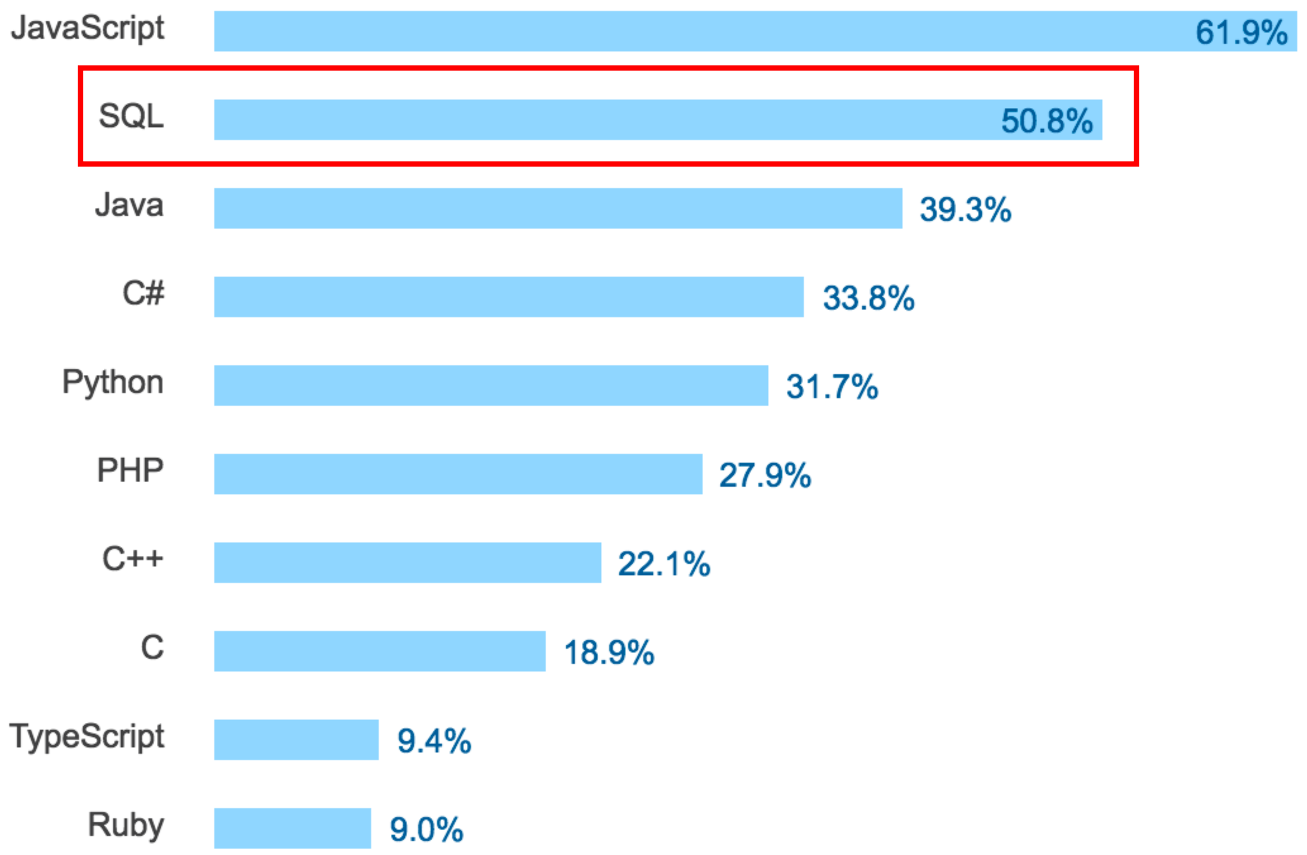
Start from the facts



matrixBi

No# 1 - SQL

2nd Most-Common *



matrixBi

* Stack overflow 2017

No# 1 - SQL

Over 130 Databases

334 systems in ranking, October 2017

Rank Oct 2017	Rank Sep 2017	Rank Oct 2016	DBMS	Database Model	Score Oct 2017	Score Sep 2017	Score Oct 2016
1.	1.	1.	Oracle	Relational DBMS	1348.80	-10.29	-68.30
2.	2.	2.	MySQL	Relational DBMS	1298.83	-13.78	-63.82
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1210.32	-2.23	-3.86
4.	4.	↑ 5.	PostgreSQL	Relational DBMS	373.27	+0.91	+54.58
5.	5.	↓ 4.	MongoDB	Document store	329.40	-3.33	+10.60
6.	6.	6.	DB2	Relational DBMS	194.59	-3.75	+14.03
7.	7.	↑ 8.	Microsoft Access	Relational DBMS	129.45	+0.64	+4.78
8.	8.	↓ 7.	Cassandra	Wide column store	124.79	-1.41	-10.27
9.	9.	9.	Redis	Key-value store	122.05	+1.65	+12.51
10.	10.	↑ 11.	Elasticsearch	Search engine	120.23	+0.23	+21.12
11.	11.	↓ 10.	SQLite	Relational DBMS	111.98	-0.05	+3.41

- 6 out of the top 10 are SQL Based!
- Vertica ranked at No# 26 **overall** and #15 in **Relational DBMS** ranking

matrixBi

Statistics from db-engines.com

No# 1 - SQL

47 Years old

It is the oldest programming language in use today!!

1970 - "A Relational Model of Data for Large Shared Data Banks”

Followed By:

- C (1972)
- C++ (1983)

matrixBi

* Stack overflow 2017

LIKE IT OR NOT

SQL IS HERE TO STAY!

No# 2 - Data

90% of the data in the world was created in the last 2 years

(Exceeding moors law)

matrixBi

WHY ARE WE USING 1990'S MTHODOLIGIES?

matrixBi

No# 3 - Usage

Users want more data!

Faster and closer to source format!

matrixBi

LONG DESIGN AND DEVELOPMENT PROCCESSESES ARE JUST NOT RELEVANT ANYMORE

How to build a multi PB DWH in less then 6 months?

matrixBi

Design

Execute

Evolve

Step 1

Design

Define Objectives

What roles should the system fill (Data hub, Searching, Reporting etc.)

Understand Context

How does the system interact with existing platforms?

What sources, volumes and integration requirements exists?

Determine Platform

Understand the types and volume of data your Big Data Application

Hadoop is not always the default selection

Step 2

Execute

Start SMALL; graduate slowly

Avoid boiling the ocean!

Example: Start with offline & Batch and mature in Real time stream processing

Collect Data

Design generic data collection modules that allows schedule, continues and event driven data collection

Close the loop

Feed data sets that are processed by system to users and sources!

Step 3

Evolve

Machine Learning

Use behavioral analysis and un-supervised models to find the gold nuggets

Evaluate Real Time models

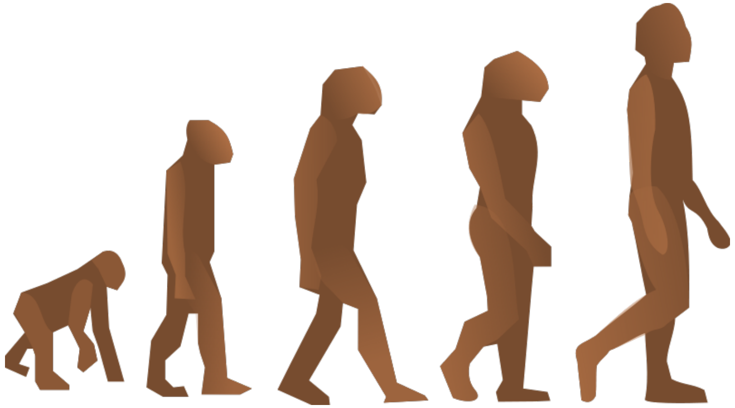
Provide real-time access to the models developed
In Example: Asses user Life Time Value upon signup

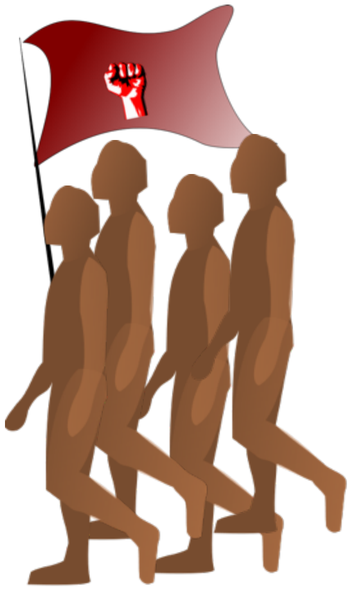
Continuously Refine

Evaluate and measure result while refining the process

Design

Technology





Evolution vs. Revolution

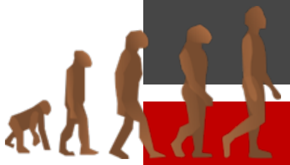
Evolution

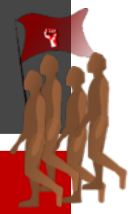
- Massively Parallel Processing DBMSs
- Mostly Relational
- Consistency model is ACID
- SQL Interface

* Some use Special Hardware

Revolution

- Distributed processing systems
- File / Object store
- Consistency model is BASE
- New languages, usually JAVA interface





COMMON ARCHITECTRE PATTERNS

BASE
Basically **A**vailable
Soft State system
With **E**ventual consistency

*See brewers CAP theorem

matrixBi

It's the consistency stupid!

ACID:

- Strong consistency.
- Less availability.
- Pessimistic concurrency.
- Complex.

VERTICA

Fully ACID compliant

BASE:

- Availability is the most important thing. Willing to sacrifice for this (CAP).
- Weaker consistency (Eventual).
- Best effort.
- Simple and fast.
- Optimistic.

DW's Need high consistency

Hadoop is not the default answer

matrixBi

But I have really Big Data

Large Scale DW implementation

Largest (public) production Vertica deployments

- **300+ Nodes – 6+ Petabyte**
- Facebook – More than 2 Petabyte
- Zynga – 3.2 Petabyte

OUTDATED

Design

Data Warehouse V2.0

The Data Vault

matrixBi

Old Challenges, New Consideration

DW's Still deliver	The problem space now contains
Data integration of multiple systems	Real Time Data
Accuracy, completeness and auditability	Shorter time to access / Real 'Self Service'
Reporting	Larger amounts of data
Clean Data	Many more systems
A "single version of the truth"	

What is best practice today?

A modern, best in class data warehouse:

- Is designed for **scalability**, ideally using **MPP** & Cloud architecture
- Uses a **bus-based**, lambda architecture for Data Loading
- Has a federated data model for structured and unstructured data
- Uses an agile data model like **Data Vault** *
- Is built using **code automation**
- Processes data using **ELT, not ETL**

What is the Data Vault model?

The Data Vault Model is a **detail oriented**, **historical** tracking and uniquely **linked set of normalized tables** that support one or more functional areas of business. It is a **hybrid approach** encompassing the best of breed between 3rd normal form (3NF) and star schema.

The design is **flexible, scalable, consistent and adaptable** to the needs of the enterprise

Execute

Data Architecture

Columns Store tradeoffs

Weakness	Solution
No PK / FK integrity enforced on write	Design using calculated keys (e.g. hashes)
Slow on DELETE, UPDATE	Build ETLs that use COPY / TRUNCATE
Sl[illegible]	Use specific key/Value OLTP
Op[illegible] for limited concurrency but big queries; only a few users can use at a time	Optimize data structures for common queries and leverage big, slow disks to create denormalized tables

Vertica overcomes some of these!!

Data Vault Model

Methodology

- Consistent
- Repeatable
- Pattern Based

Architecture

- Multi-Tier support
- Supports C-Store & NoSQL
- Scalable

Model

- Flexible
- Scalable
- Hub & Spoke

Automation

Evolution

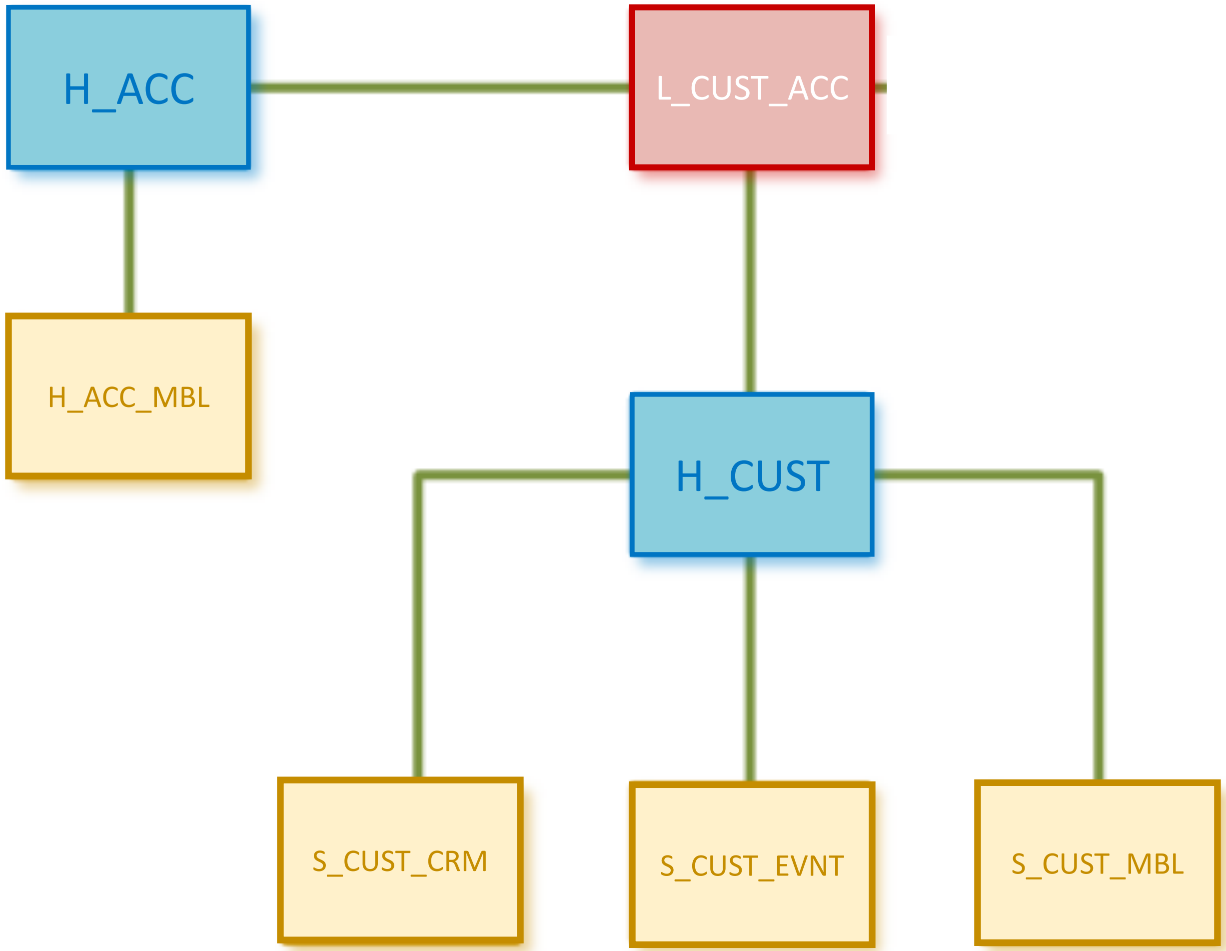
Scalability

Data Vault 2.0

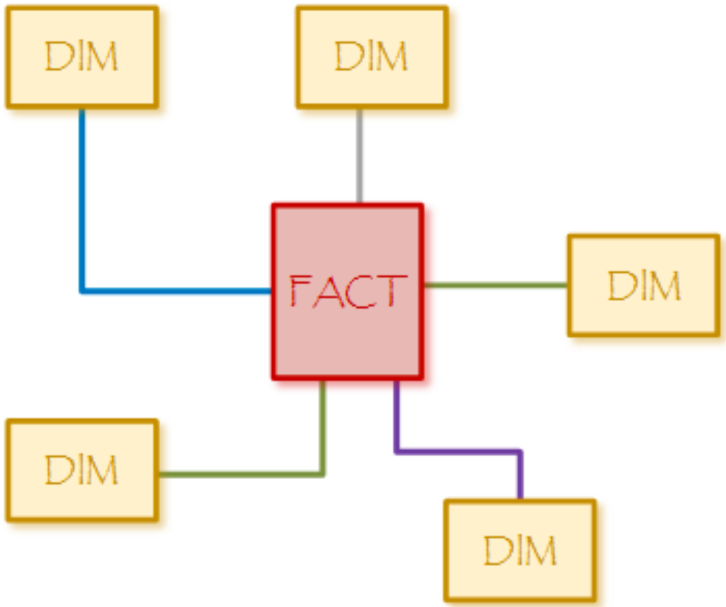
Data model designed for simple automated loading of data with repetitive entity based design built on the Hub & Spoke paradigm

HUB	A table with Keys and static entity data
Satellite	A table containing entity data and the key from the Hub
Link	Essentially an M2M table designed to link Hubs
Reference	A Static ENUM table

Data Vault Model



Data Vault



Star Schema

matrixBi

Real World Example MPP – 3 Node HP Vertica database

Time: First fetch (10 rows): 153.675 ms.

All rows formatted: 153.733 ms
somedb=> select count(*) from h_events;
 count

 247,915,500,000
(1 row)

```
SELECT /* +label(c4q0022) */ computer_id,dt , count(distinct process_name) process_count
From h_events
WHERE dt =  2017-06-07  and tm between  14:41:30  and  21:01:45
Group by computer_id, dt
ORDER BY process_count DESC
 limit 10;
```

Time: First fetch (10 rows): 16715.003 ms.
All rows formatted: 16715.060 ms

This is what we have after the where to the group by and distinct. - 18.3B rows out of 248B

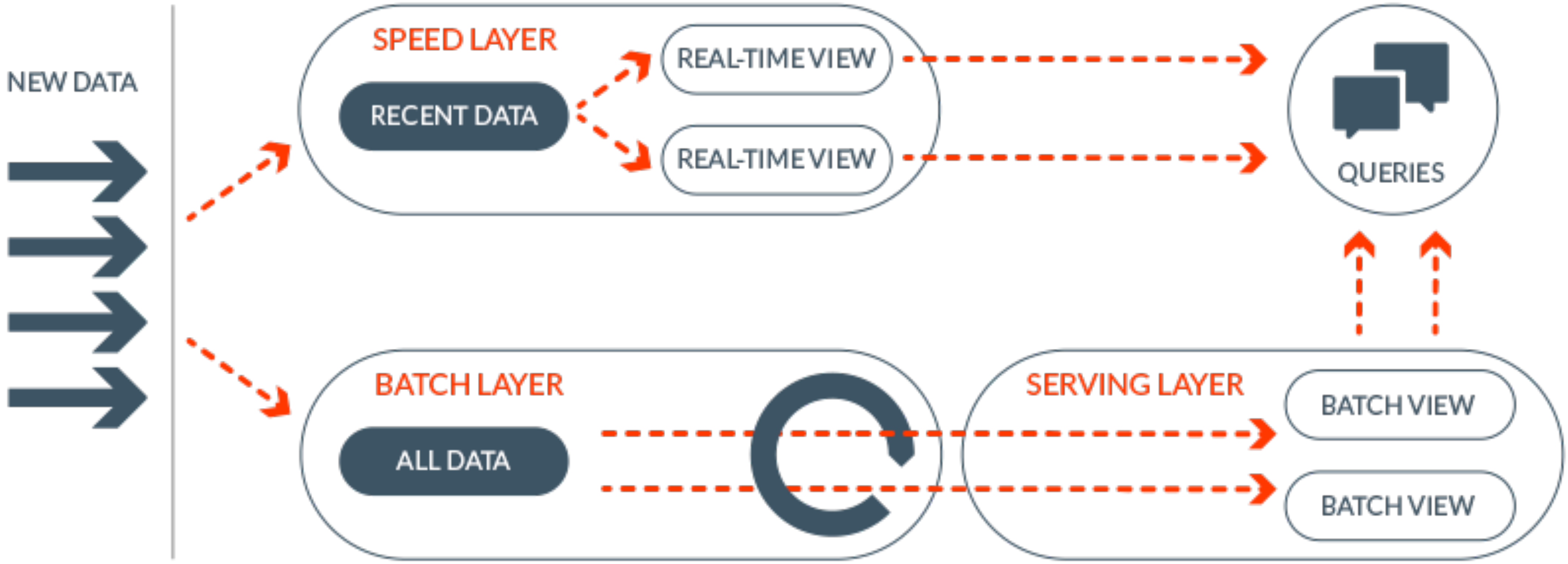
matrixBi

Execute

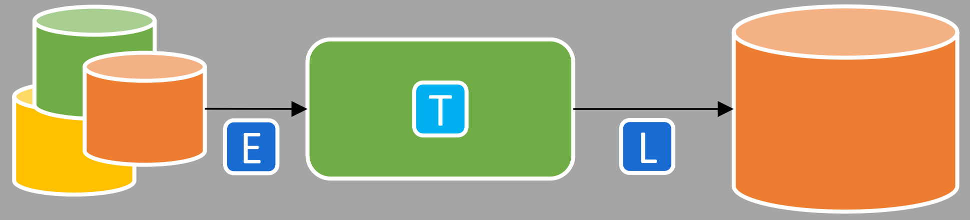
Data Load

Bus based architecture

Lambda Architecture

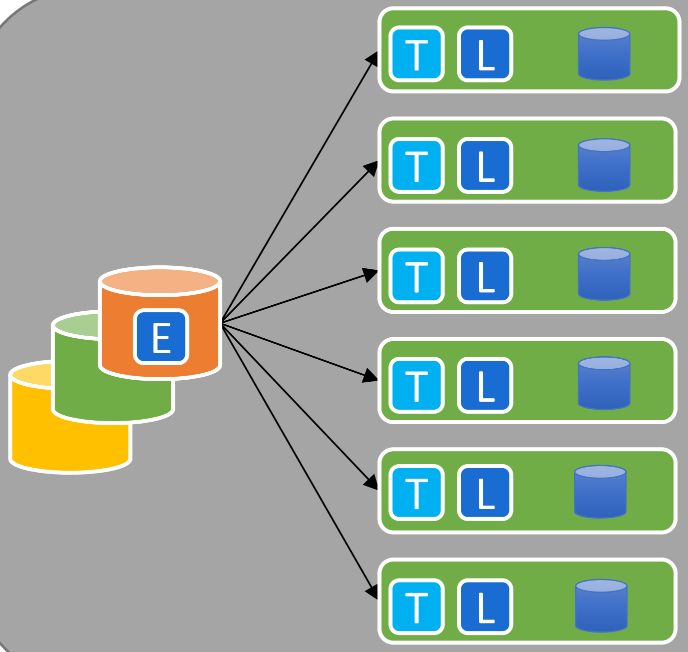


ETL vs ELT



Transform is a separate ETL Server

- Proprietary Engine
- Poor Performance
- High Costs



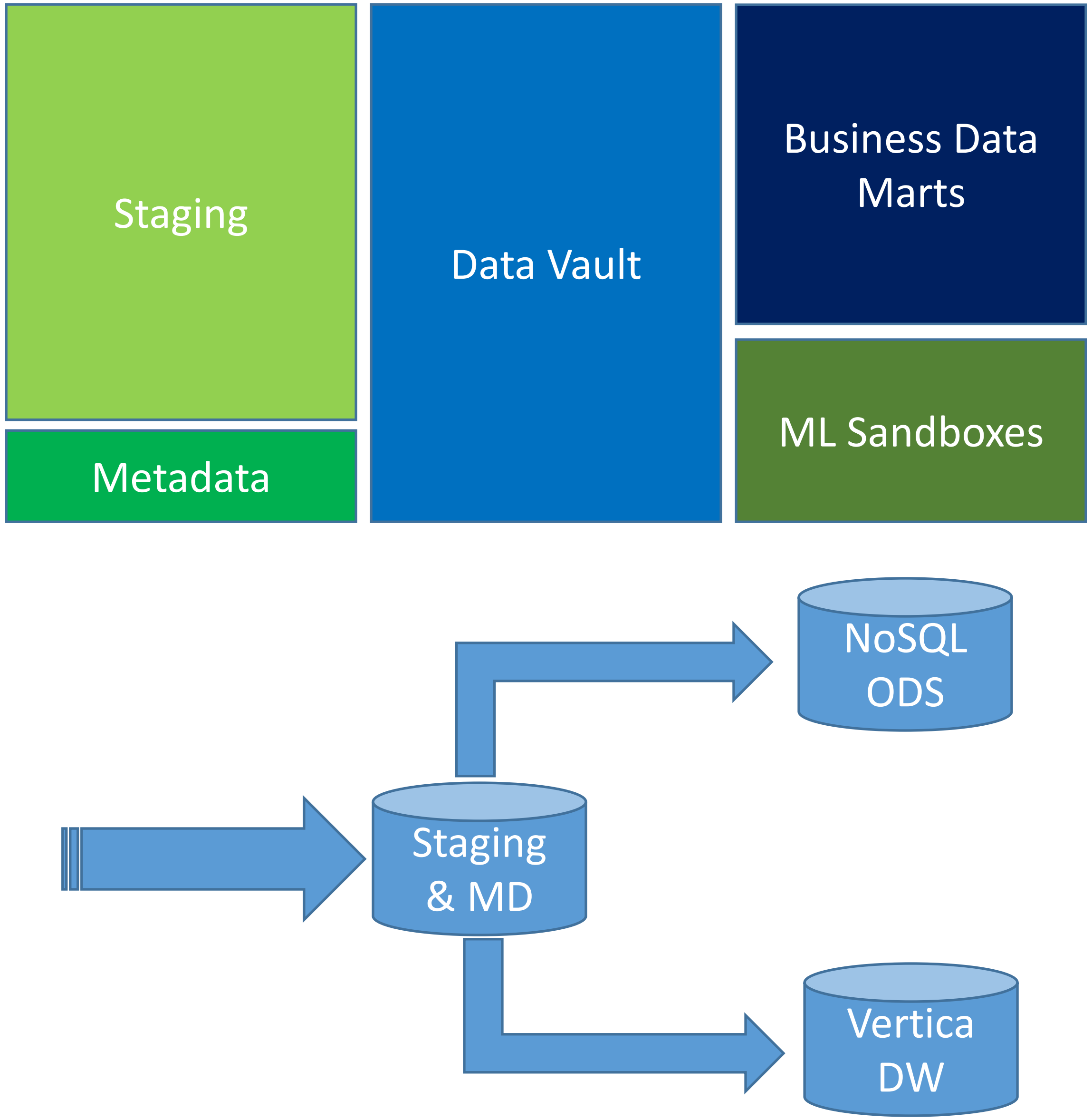
Transform in Database

- Leverage distributed resources
- Efficient
- High Performance

Benefits

- Optimal Performance & Scalability
- Easier to Manage & Lower Cost

Loading Data to the Vault



1. **Staging area** – Create Hashed Keys, Re-Use GKs (Can be a different DB like PGSQL, MYSQL etc.) or a **Vertica FlexZone table**.
2. **Metadata** – A reference area containing source MD, configuration, Management Tables etc.
3. **Data Vault** – The main Vault structure
4. **Business DMs** – Aggregation and pre processed tables that hold business data.
5. **ML Sandboxes** – An area to create feature vectors, training data sets etc.

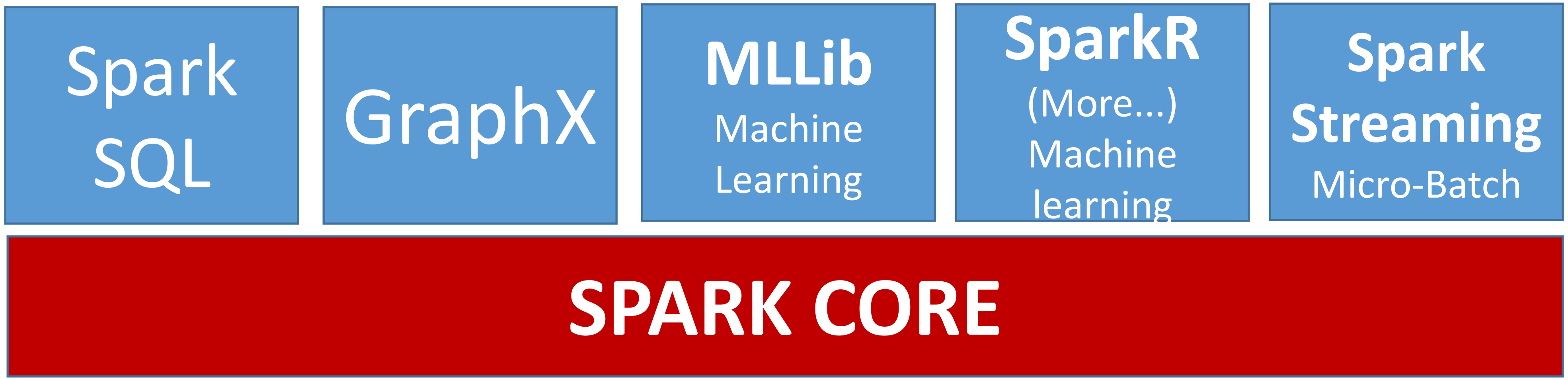
Some implementations adds ODS & OPS Marts with constant keys generated in the staging and MD Area.

Loading Data to the Vault

And More...

Spark is a General Purpose distributed data processing framework

Core engine with libraries for streaming, SQL Machine learning, Graph processing and more...

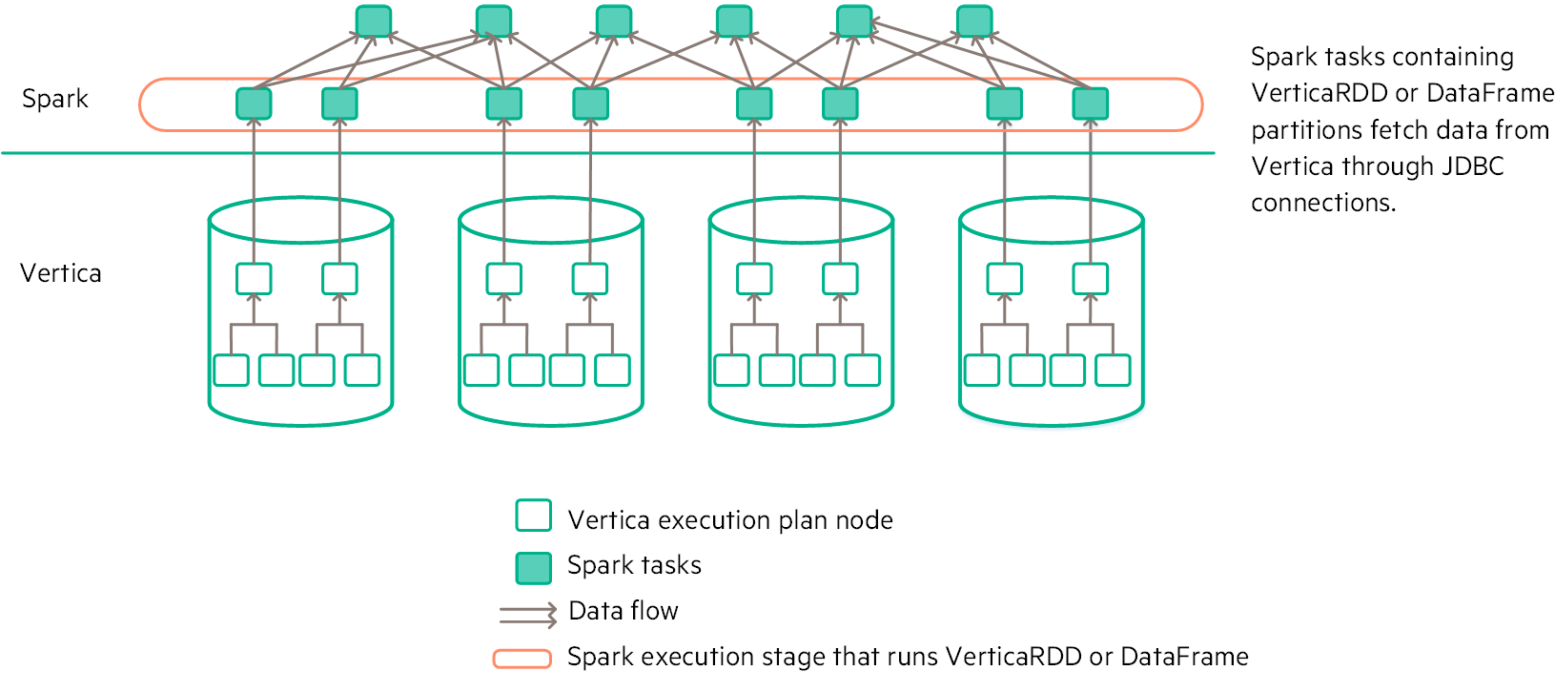


VERTICA

APACHE

Spark™

matrixBi



Spark
Vertica
Spark tasks containing VerticaRDD or DataFrame partitions fetch data from Vertica through JDBC connections.
Vertica execution plan node
Spark tasks
Data flow
Spark execution stage that runs VerticaRDD or DataFrame

Spark-Vertica connector in a nutshell

- 2 Way connector
- Locality aware partitions
- Locality aware query
  - Query pruning
- Computation push-down
  - Filters
  - Projections – Count(*)
  - Joins
  - Aggregations

Machine Learning
Graph Processing
Distributed Data Processing
Streaming / Micro-batching

Geospatial analytics
Sentiment analysis
Sessionization of event streams
Time series pattern matching
(Many more)...

Additional Methodologies

1	**Multi-Temperature Data Management** Hierarchical Storage	Cost effectiveness over extremely large data volumes
2	**Polymorphic File System** Multi data base usage	Native Storage in a form most suitable for processing
3	**Late Binding** With Data access capabilities that go beyond traditional	Usable by data scientists who does not need to be a computer scientist

matrixBi

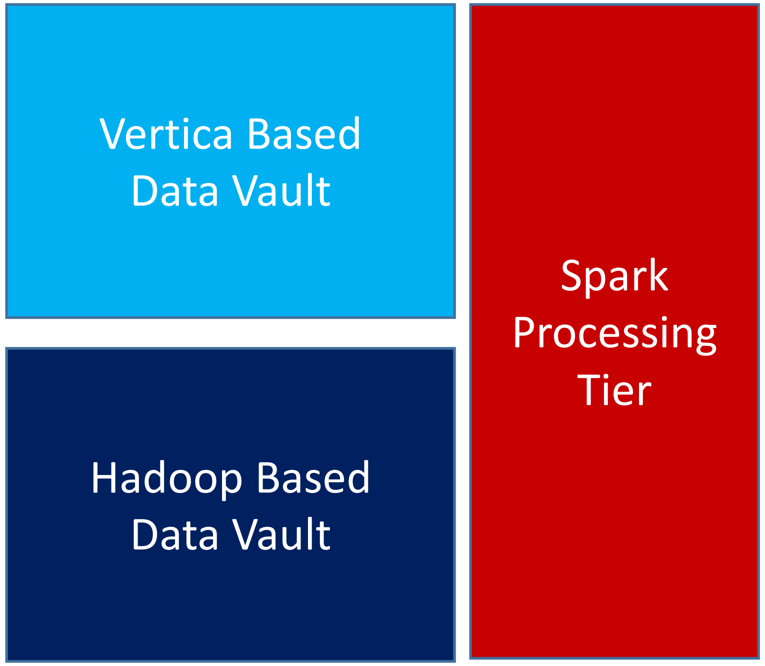
Multi temperature data Tiering approach

20% of EDW data is HOT

- Used frequently
- Recent Data

80% of the data is warm or cold

- Accessed infrequently
- History – months, years
- High granularity



matrixBi

Evolve

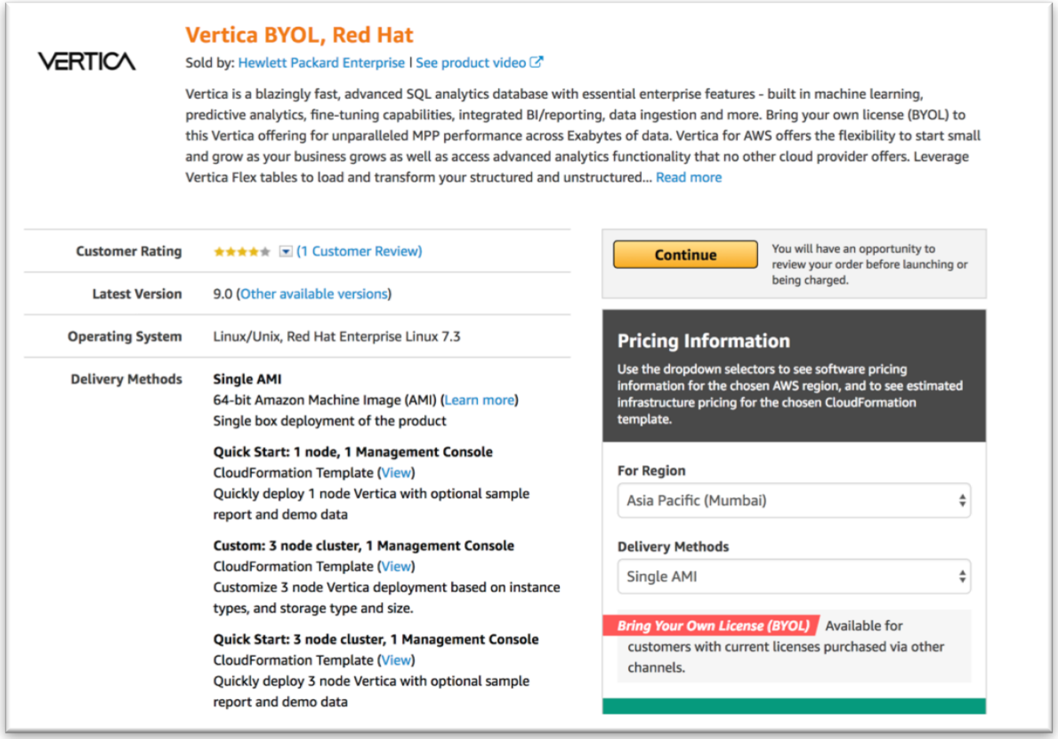
Data Load Automation
Machine Learning
More...

Success Factors

Start Quick

Try implementing a cloud first approach where Development is agile and fast.

Set up an On-Prem environment while you develop...

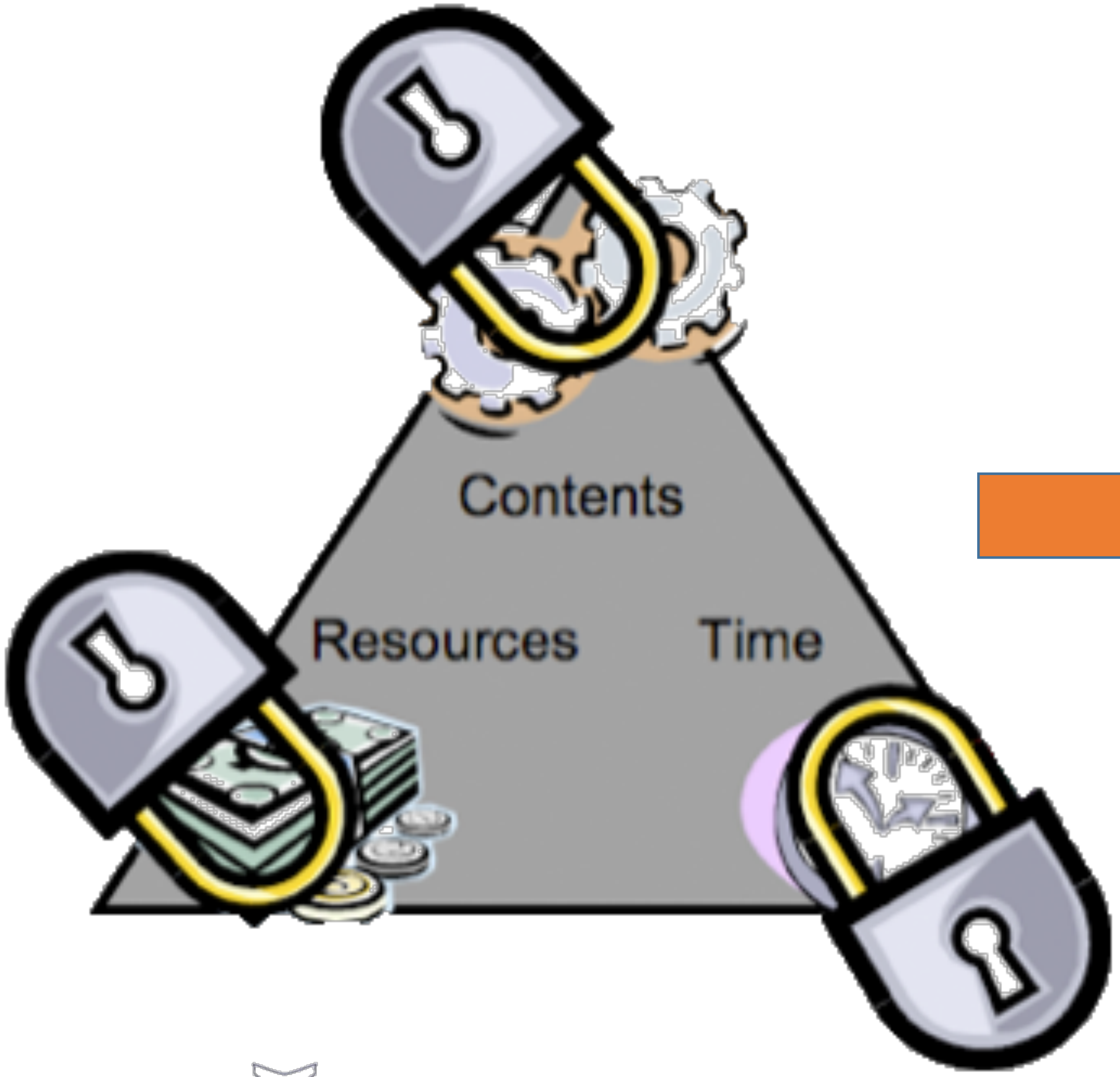


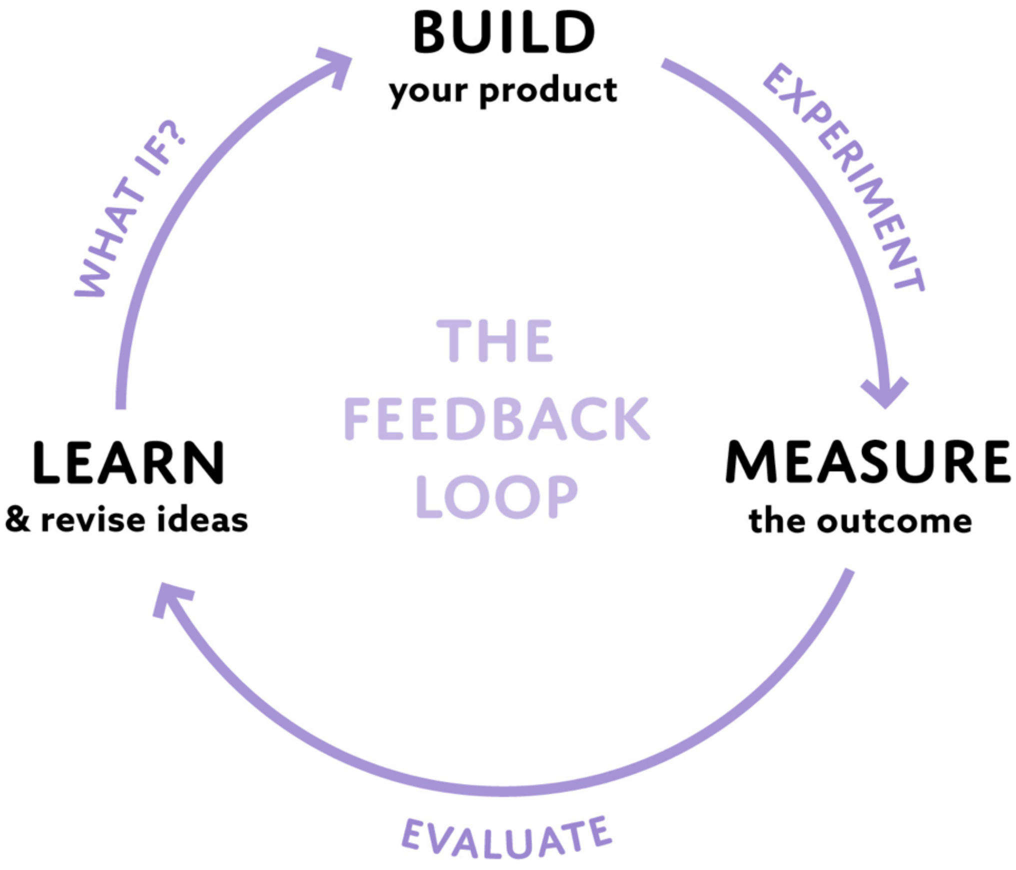
Think about data for the long term

Every data project should be started with consideration for the data's **reusability** in future applications.

By understanding that upcoming and future data needs are often **unknown**, you can prepare and utilize data accordingly.

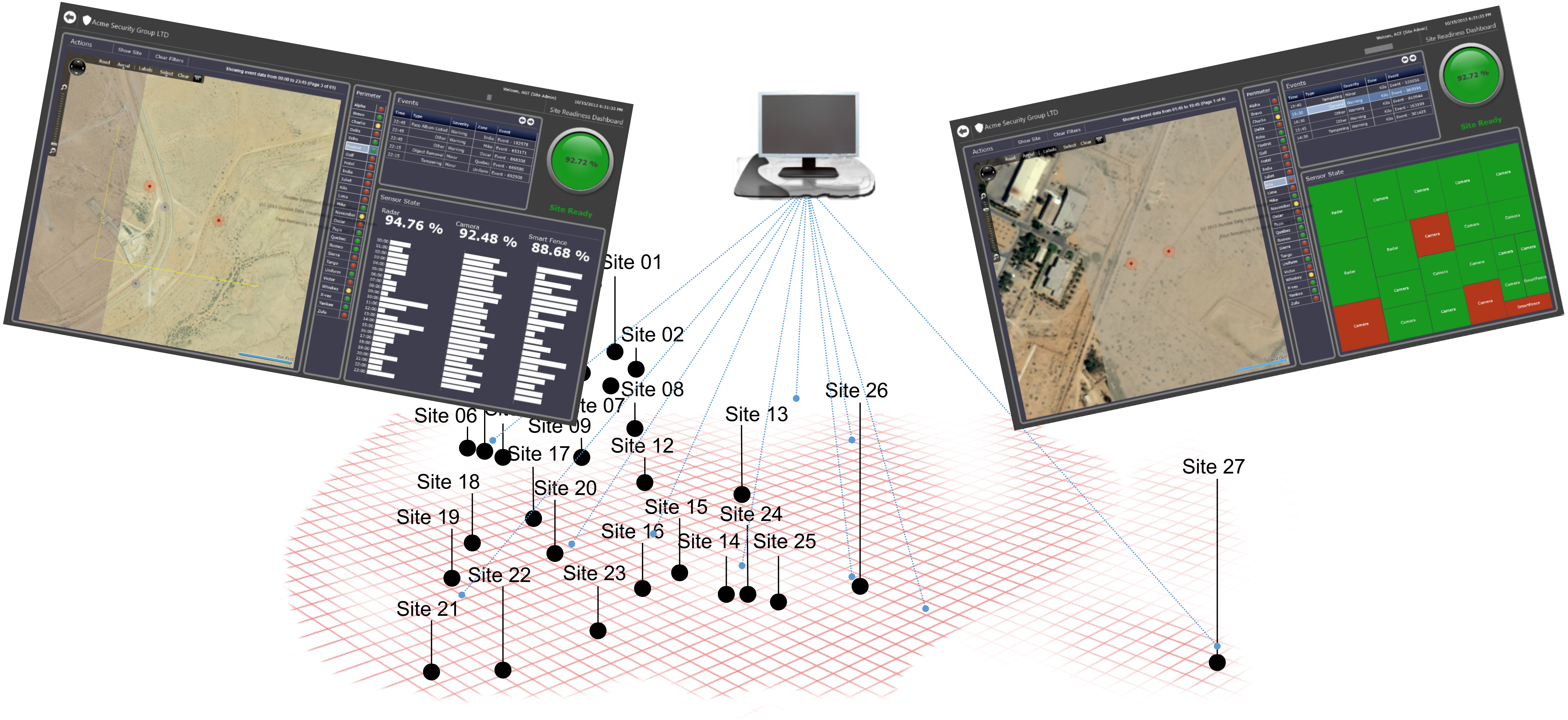
From fixed budget to Fast Iterations and Feedback loops





Use Case

Use case – Command and Control Intelligence



Challenges and Solutions

Challenge	Solution
Provide fast query response times for complicated reports	State of the art Columnar Datawarehouse design
Provide scalable infrastructure for future growth	Using Shared-Nothing architecture we achieve **Infinite Scalability**
Support loading and Processing of large amounts of structured data	Big Data & Modular Architecture
Process data from many different sites and systems geographically remote	Data Source Agnostic – Open Architecture
Provide easy and secure access for users located at many sites	Rich Web Architecture - Zero Footprint

Use case – High End

Analytics & UI

MicroStrategy

ANGULAR

GIS

esri

ML

HP Vertica Distributed R

Big Data

VERTICA

hadoop

Future Phases

Configuration and Metadata

PostgreSQL

Data Integration

Servlet

J2EE

JMS

Java

talend*

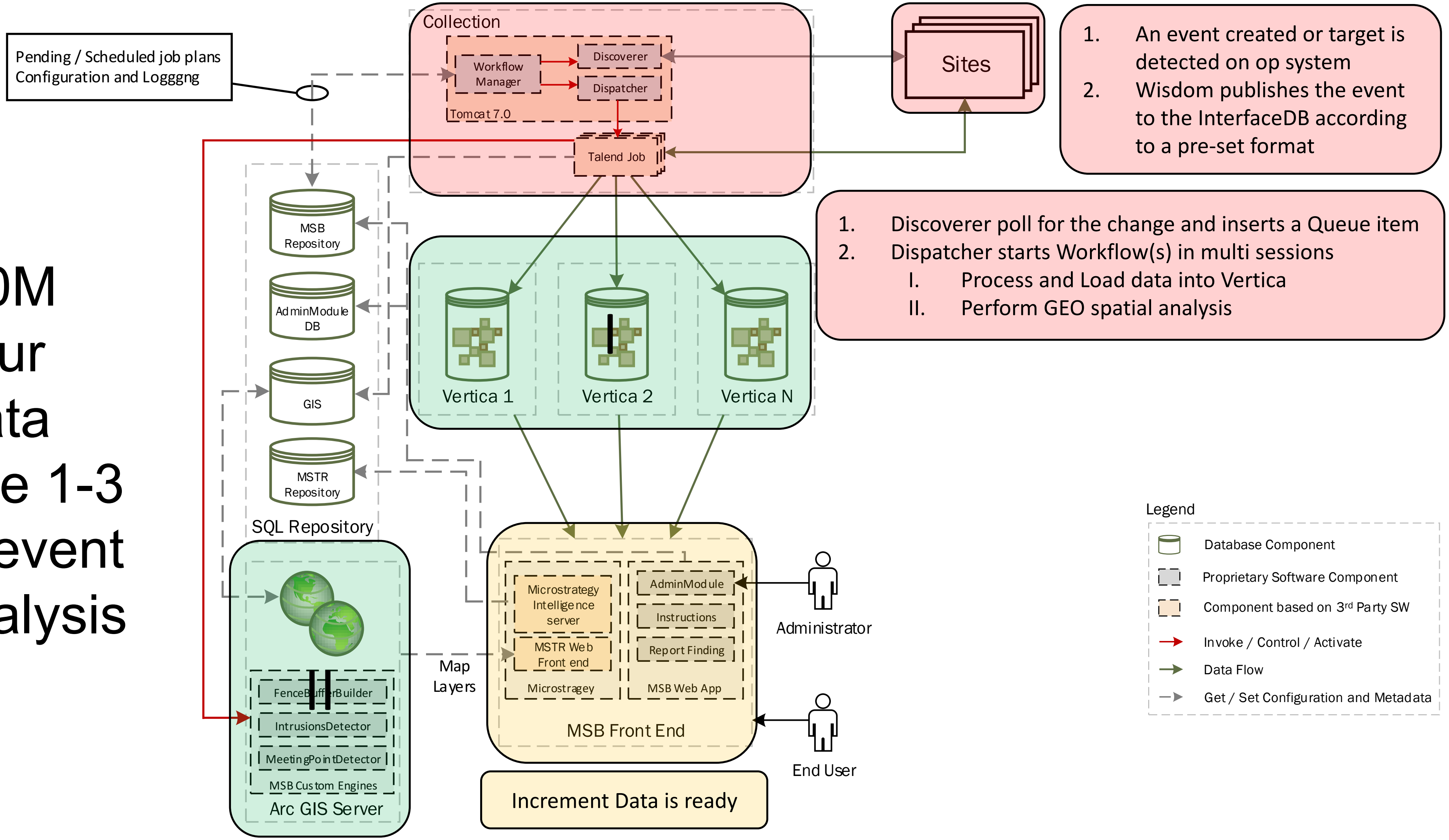
*open data solutions

matrixBi

Data flow

Capacity:

- Process 35-60M Events per hour
- End to End data availability time 1-3 minutes from event creation to analysis



matrixBi

Geospatial performance test

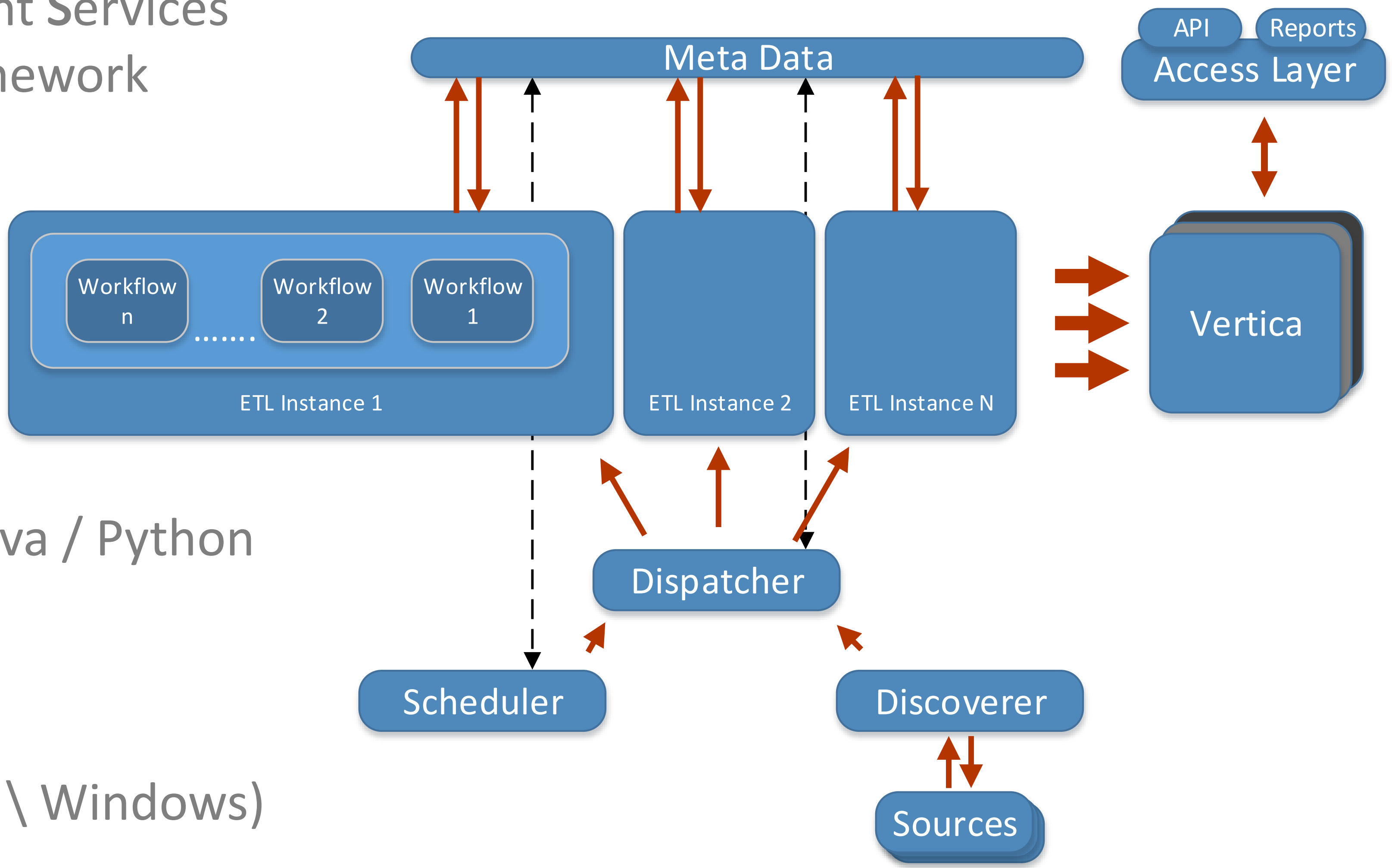
3 Node cluster and 4TB of Geospatial data

Use case	SQL (2014)	Oracle (12c)	Vertica (7.1)	Improvement (from Oracle)
1	52 Sec	9 Seconds	109 milliseconds	99%
2	6 Sec	3-6 Seconds	94 milliseconds	97%-98%
3	5:30 Min per week	54 Sec Per week	13 Sec Per week (29.5M Records)	76%
4	Over 13 Mins	1:40 to 3:00 Min	16 Sec	84%-89%
5	10 Seconds	13 seconds	300 milliseconds	98%

Components – DMS Architecture

MSB - (Distributed) **D**ata **M**ovement **S**ervices

- Micro batch data loading framework
- Guaranteed delivery!
  - Retry and self healing
  - High Availability
- Multi server architecture



- Flexibility (Can run Talend / Java / Python other executables)
- Platform independent! (Linux \ Windows)

matrixBi

Big journeys begin with small steps

matrixBi

Thank You!

Ozle@MatrixBI.co.il