**NIST Big Data**

**Technology Roadmap**

**Version 1.0**

**Input Listing: M0087**

**Technology Roadmap Subgroup**

**NIST Big Data Working Group (NBD-WG)**

**September, 2013**

[Executive Summary 4](#_Toc386537155)

[1 Purpose, Background, and Vision 4](#_Toc386537156)

[1.1 NIST Technology Roadmap Purpose 4](#_Toc386537157)

[1.1.1 Technology Roadmap Subgroup 5](#_Toc386537158)

[1.2 Big Data Background 6](#_Toc386537159)

[1.3 NIST Big Data Technology Roadmap Stakeholders 7](#_Toc386537160)

[1.4 Guiding Principles for Developing the NIST Big DataTechnology Roadmap 9](#_Toc386537161)

[2 NIST Big Data Definitions and Taxonomies (from Def. & Tax. Subgroup) 9](#_Toc386537162)

[3 Big Data Requirements (from Requirements & SecNPrivacy Subgroups) 10](#_Toc386537163)

[4 Big Data Reference Architecture (from RA Subgroup) 11](#_Toc386537164)

[5 Big Data Security and Privacy (from SecNPrivacy Subgroup) 12](#_Toc386537165)

[6 Features and Technology Readiness 14](#_Toc386537166)

[6.1 Technology Readiness 14](#_Toc386537167)

[6.1.1 Types of Readiness 15](#_Toc386537168)

[6.1.2 Scale of Technological Readiness 15](#_Toc386537169)

[6.2 Organizational Readiness and Adoption 16](#_Toc386537170)

[6.2.1 Types of Readiness 16](#_Toc386537171)

[6.2.2 Scale of Organizational Readiness 16](#_Toc386537172)

[6.2.3 Scale of Organizational Adoption 17](#_Toc386537173)

[6.3 Features Summary 18](#_Toc386537174)

[6.4 Feature 1: Storage Framework 23](#_Toc386537175)

[6.4.1 Physical Storage Frameworks 24](#_Toc386537176)

[6.4.2 Logical Data Distribution 26](#_Toc386537177)

[6.5 Feature 2: Processing Framework 36](#_Toc386537178)

[6.5.1 Batch Frameworks 38](#_Toc386537179)

[6.6 Feature 3: Resource Manager Framework 40](#_Toc386537180)

[6.7 Feature 4: Infrastructure Architecture 41](#_Toc386537181)

[6.8 Feature 5: Information Framework 42](#_Toc386537182)

[6.9 Feature 6: Standards Integration Framework 43](#_Toc386537183)

[6.10 Feature 7: Application Framework 44](#_Toc386537184)

[6.11 Feature 8: Business Operations 44](#_Toc386537185)

[7 Big Data Related Multi-stakeholder Collaborative Initiatives 47](#_Toc386537186)

[7.1.1 Characteristics supported by standards 48](#_Toc386537187)

[7.1.2 Information and Communications Technologies (IT) Standards Life Cycle 49](#_Toc386537188)

[7.2 Data Service Abstraction 49](#_Toc386537189)

[7.2.1 Data Provider Registry and Location service 49](#_Toc386537190)

[7.2.2 Data Provider Interfaces 51](#_Toc386537191)

[7.2.3 Data Sources 51](#_Toc386537192)

[7.3 Usage Service Abstraction 54](#_Toc386537193)

[7.4 Capability Service Abstraction 54](#_Toc386537194)

[7.4.1 Security and Privacy Management 54](#_Toc386537195)

[7.4.2 System Management 60](#_Toc386537196)

[7.5 Standards Summary 60](#_Toc386537197)

[8 Big Data Strategies 60](#_Toc386537198)

[8.1 Strategy of Adoption 60](#_Toc386537199)

[8.1.1 Identify and include stakeholders 60](#_Toc386537200)

[8.1.2 Identify potential road blocks 60](#_Toc386537201)

[8.1.3 Define Achievable Goals 61](#_Toc386537202)

[8.1.4 Define “Finished” and “Success” at the beginning of the project 61](#_Toc386537203)

[8.2 Strategy of Implementation 62](#_Toc386537204)

[8.3 Resourcing 64](#_Toc386537205)

[9 Concerns and Assumptions Statement 65](#_Toc386537206)

# Executive Summary

Provide executive level overview of the Technology Roadmap, introduce the vision of the document.

Author: Tech Writer – Leaving blank until October

[Content Goes Here]

# Purpose, Background, and Vision

## NIST Technology Roadmap Purpose

Author: Wo/Carl

There is a broad agreement among commercial, academic, and government leaders about the remarkable potential of “Big Data” to spark innovation, fuel commerce, and drive progress. Big Data is the term used to describe the deluge of data in our networked, digitized, sensor-laden, information driven world. The availability of vast data resources carries the potential to answer questions previously out of reach. Questions like: How do we reliably detect a potential pandemic early enough to intervene? Can we predict new materials with advanced properties before these materials have ever been synthesized? How can we reverse the current advantage of the attacker over the defender in guarding against cybersecurity threats?

However there is also broad agreement on the ability of Big Data to overwhelm traditional approaches. The rate at which data volumes, speeds, and complexity are growing is outpacing scientific and technological advances in data analytics, management, transport, and more.

Despite the widespread agreement on the opportunities and current limitations of Big Data, a lack of consensus on some important, fundamental questions is confusing potential users and holding back progress. What are the attributes that define Big Data solutions? How is Big Data different from the traditional data environments and related applications that we have encountered thus far? What are the essential characteristics of Big Data environments? How do these environments integrate with currently deployed architectures? What are the central scientific, technological, and standardization challenges that need to be addressed to accelerate the deployment of robust Big Data solutions?

At the NIST Cloud and Big Data Forum held in January 15-17, 2013, the community strongly recommends NIST to create a public working group for the development of a Big Data Technology Roadmap. This roadmap will help to define and prioritize requirements for *interoperability*, *portability*, *reusability*, and *extensibility* for big data usage, analytic techniques and technology infrastructure in order to support secure and effective adoption of Big Data.

On June 19, 2013, the NIST Big Data Public Working Group (NBD-PWG) was launched with overwhelmingly participation from industry, academia, and government across the nation. The scope of the NBD-PWG is to form a community of interests from all sectors including industry, academia, and government, with the goal of developing a consensus in definitions, taxonomies, secure reference architectures, and a technology roadmap. Such a consensus would therefore create a vendor-neutral, technology and infrastructure agnostic framework which would enable Big Data stakeholders to pick-and-choose best analytics tools for their processing and visualization requirements on the most suitable computing platform and cluster while allowing value-added from Big Data service providers.

Currently NBD-PWG has created five subgroups namely the Definitions and Taxonomies, Use Case and Requirements, Security and Privacy, Reference Architecture, and Technology Roadmap. These subgroups will help to develop the following set of preliminary consensus working drafts by September 27, 2013:

1. Big Data Definitions
2. Big Data Taxonomies
3. Big Data Requirements
4. Big Data Security and Privacy Requirements
5. Big Data Reference Architectures White Paper Survey
6. Big Data Reference Architectures
7. Big Data Security and Privacy Reference Architectures
8. **Big Data Technology Roadmap**

Due to time constraints and dependencies between subgroups, the NBD-PWG hosted two hours weekly telecon meeting from Mondays to Fridays for the respective subgroups. Every three weeks, NBD-PWG called a joint meeting for progress reports and document updates from these five subgroups. In between, subgroups co-chairs met for two hours to synchronize their respective activities and identify issues and solutions.

### Technology Roadmap Subgroup

The focus of the NBD-WG Technology Roadmap Subgroup was to form a community of interest from industry, academia, and government, with the goal of developing a consensus vision with recommendations on how Big Data adoption should move forward by performing a good gap analysis through the materials gathered from all other NBD subgroups. This included setting standardization and adoption priorities through an understanding of what standards are available or under development as part of the recommendations. The primary tasks of the Technology Roadmap Subgroup included:

* Gather input from NBD subgroups and study the taxonomies for the actors’ roles and responsibility, use cases and requirements, and secure reference architecture.
* Gain understanding of what standards are available or under development for Big Data
* Perform a thorough gap analysis and document the findings
* Identify what possible barriers may delay or prevent adoption of Big Data
* Document vision and recommendations

## Big Data Background

Author: Dave

There is an old saying that everything is about perspective. The fundamental characteristic of Big Data is that it is too big (volume), or arrives too fast(velocity) or is too diverse (variety) to be processed within a local computing structure without using additional approaches/techniques to make the data fit or provide a result in an acceptable time frame. If we look at big data from a time perspective what was considered extremely large even five years ago can be handled easily today on portable and mobile platforms. The use of swapping and paging from ram to disk or other media was one of the very first techniques employed to deal with what was thought of as big data years ago. What will be considered big five years from now may likely depend on how well Moore’s Law continues to hold. From a connectivity perspective what is considered big is determined by how long it would take to retrieve/move the data to get an answer, or in some cases if it would be even possible to move the data. A high resolution image from a sensor would likely not be considered big when being retrieved and processed within a data center or even office networking environment.

However, to a soldier on dismounted patrol in the mountains of Afghanistan it is not even practical to being to transfer him that data in its native form. From a variety or complexity perspective even our cell phones today process a wide variety of web based content to include text, video, URLs easily. On the other side, there is too much variety in that data for the processor to reason about it or turn it into relevant information and knowledge without a human reviewing it. Even, large data centers struggle to align and reason about diverse data and the long term vision of a semantic web must deal heavily with the diverse domains and multiple semantic meanings assigned to common terms and concepts.

The total scale of data is well described by the NSA an organization that has dealt with Big Data type problems for decades as follows "According to figures published by a major tech provider, the Internet carries 1,826 Petabytes of information per day. In its foreign intelligence mission, NSA touches about 1.6% of that. However, of the 1.6% of the data, only 0.025% is actually selected for review.

\*\*\*\*EDITORIAL COMMENT\*\*\*\*\*

The net effect is that NSA analysts look at 0.00004% of the world's traffic in conducting their mission - that's less than one part in a million. Put another way, if a standard basketball court represented the global communications environment, NSA's total collection would be represented by an area smaller than a dime on that basketball court."

The problem with Big Data is that at some point in time, applying more and bigger, does not result in an uptick in performance, based on the investment.

there is a threshold at which for a certain set of data and specific application at which simply using a faster processor, more memory, more storage, or other traditional data management techniques (scaling vertically, data Organization/indexing, algorithms) cannot produce an answer in an acceptable timeframe and requires approaches that distribute the data across multiple processing nodes (scale horizontally) to meet the application requirements.

## NIST Big Data Technology Roadmap Stakeholders

Who should read this Tech Roadmap, what should they plan to takeaway from reading this document. Define stakeholders and include a stakeholder matrix that relates to the remaining sections of this document. This should likely also include a RACI matrix (RACI == Dan’s section)

This document is target towards any individual or actor supporting or involved in the adoption or implementation of Big Data technologies or systems. As documented in the NIST Big Data Taxonomy there are a wide range of actors who support various roles and thus have a stake in the adoption of Big Data technology within an organization. The following roles and associated actors are defined in the taxonomy as shown in Table 1.3-1 below.

|  |  |
| --- | --- |
| Role | Actors |
| System Orchestrator | Business Leadership  Consultants  Data Scientists  Information Architects  Software Architects  Security & Privacy Architects  Network Architects |
| Big Data Security and Privacy | Corporate Security Officer  Security Specialist  Security and Privacy Architects  Internal Auditor  External Auditor |
| System Management | In-house Staff  Data Center Management  Cloud Providers |
| Data Consumer | End Users  Researchers  Other Applications & Systems |
| Data Provider | Enterprises  Other Applications & Systems  Public Agencies  Researchers & Scientists  Search Engines  End Users  Network Operators |
| Big Data Application Provider | Application Specialists  Platform Specialists  Consultants |
| Big Data Framework Providers | System Architects  Information Architects  Software Architects  Network Architects  Cloud Providers |

In implementing a Big Data system it is critical that the responsibilities of the actors supporting these various roles be well defined as they are all critical stakeholders to the effort as shown in the following RACI (Responsible, Accountable, Consulted & Informed) Matrix Table 1.3-2 below. Within this matrix is a potential alignment of core responsibilities to the roles shown above. In addition, to the key Big Data roles identified in the taxonomy we have added and Executive Leadership role to denote the overall decision maker the system will ultimately support. Organizations looking to implement a Big Data system are encouraged to decompose this matrix down to the specific actors supporting each of the roles within the organization or problem domain. This decomposition will help direct those actors to the critical parts of this document that deal with their roles. This strategy is designed to aid these stakeholders in performing their roles by providing a framework to evaluate organizational readiness to implement Big Data and the readiness of the Big Data technologies to meet organizational needs along with a feature structure that will allow the stakeholder to evaluate the value and applicability of Big Data technology features to their business problems. See Appendix A, one example for applying a RACI Matrix for Security & Privacy within a BigData/Cloud Eco-system.

Author: Carl

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Executive Stakeholders | System Orchestrator | Security and Privacy | System Management | Data Consumer | Data Provider | Big Data Application Providers | Big Data Framework Provider |
| Organizational Adoption and Business Strategy | A | R | C | C | C | I | C | C |
| Infrastructure and Architecture | I | R | A | A | I | I | C | C |
| Complex analytics, reporting, and business intelligence | I | A | C | C | C | I | R | C |
| Programming paradigms and information management | I | A | C | C | C | C | R | C |
| Deployment, administration, and maintenance | I | R | R | R/A | I | C | R | R |

## Guiding Principles for Developing the NIST Big DataTechnology Roadmap

This document was developed to in accordance with the overall Big Data Working Group mission in order to create a vendor-neutral, technology and infrastructure agnostic framework to enable Big Data stakeholders from industry, government and academia to plan for, adopt, and create the technologies and approaches appropriate to their problem domain. In addition, every attempt was made to align this document with the deliverable artifacts from the other subgroups.

# NIST Big Data Definitions and Taxonomies (from Def. & Tax. Subgroup)

The Definitions and Taxonomy subgroup focused on identifying the concepts involved in Big Data, and defining terms in both the concepts needed to describe this new reality, and to define the terms used in the reference architecture.

For *managers* the terms will distinguish the concepts needed to understand this changing field

For *procurement officers* this will provide the framework for discussing organizational needs, and distinguishing among offered approaches

For *marketers* this document will provide the means to promote the characteristics of solutions and innovations

For the *technical* community it will provide a common language to better differentiate the specific offerings

What makes “Big Data” big, or how large does a dataset have to be in order to be called big data? The answer is an unsatisfying “it depends”. Data is considered “big” if the use of the new scalable architectures provides a business efficiency over other relational data model, in other words the functionality cannot be achieved in a traditional relational database platform.

Big data focuses on the self-referencing viewpoint that data is big because it requires scalable systems to handle the volume quickly, and architectures with better scaling have come about because of the need to handle the volume and velocity of data within Big Data. The following is the core definition adopted for Big Data by the group:

***Big Data*** *consists of extensive datasets, primarily in the characteristics of volume, velocity and/or variety, that require a scalable architecture for efficient storage, manipulation, and analysis.*

For a detailed discussion of the definitions and taxonomy the reader should refer to the published NIST documents.

# Big Data Requirements (from Requirements & SecNPrivacy Subgroups)

The Requirements Subgroup collected 51 use cases organized into nine broad areas [application domains] as follows, with the number of associated use cases in parentheses:

* Government Operation (4)
* Commercial (8)
* Defense (3)
* Healthcare and Life Sciences (10)
* Deep Learning and Social Media (6)
* The Ecosystem for Research (4)
* Astronomy and Physics (5)
* Earth, Environmental and Polar Science (10)
* Energy (1)

These use cases were then decomposed into detailed requirements and then in turn boiled up into general requirements in the 7 broad areas shown below.

|  |
| --- |
| **Data Sources** |
| **General Requirement** 1. needs to support reliable real time, asynchronize, streaming, and batch processing to collect data from centralized, distributed, and cloud data sources, sensors, or instruments 2. needs to support slow, bursty, and high throughput data transmission between data sources and computing clusters. 3. needs to support diversified data content ranging from structured and unstructured text, document, graph, web, geospatial, compressed, timed, spatial, multimedia, simulation, instrumental data. |
| **Transformation** |
| **General Requirement** 1. needs to support diversified compute intensive, analytic processing and machines learning techniques 2. needs to support batch and real time analytic processing 3. needs to support processing large diversified data content and modeling 4. needs to support processing data in motion (streaming, fetching new content, tracking, etc.) |
| **Capability** |
| **General Requirement** 1. needs to support legacy and advance software packages (subcomponent: SaaS) 2. needs to support legacy and advance computing platforms (subcomponent: PaaS) 3. needs to support legacy and advance distributed computing cluster, co-processors, I/O processing (subcomponent: IaaS) 4. needs to support elastic data transmission (subcomponent: networking) 5. needs to support legacy, large, and advance distributed data storage (subcomponent: storage) 6. needs to support legacy and advance programming executable, applications, tools, utilities, and libraries |
| **Data Consumer** |
| **General Requirement** 1. needs to support fast search (~0.1 seconds) from processed data with high relevancy, accuracy, and high recall 2. needs to support diversified output file formats for visualization, rendering, and reporting 3. needs to support visual layout for results presentation 4. needs to support rich user interface for access using browser, visualization tools 5. needs to support high resolution multi-dimension layer of data visualization 6. needs to support streaming results to clients |
| **Security & Privacy** |
| **General Requirement** 1. Must support the protection and preserve the security and privacy of sensitive data 2. Supports multi-level policy-driven, sandbox, access control, authentication on protected data  3. Supports digital forensics.  3. Supports internal and external audits. |
| **Lifecycle** |
| **General Requirement** 1. Supports data quality curation including pre-processing, data clustering, classification, reduction, format transformation 2. Supports dynamic updates on data, user profiles, and links 3. Supports data lifecycle and long-term preservation policy including data provenance 4. Supports data validation 5. Supports human annotation for data validation 6. Supports prevention of data loss or corruption 7. Supports multi-sites archival 8. Supports persistent identifier and data traceability 9. Supports standardize, aggregate, and normalize data from disparate sources |
| **Others** |
| **General Requirement** 1. Supports rich user interface from mobile platforms to access processed results 2. Supports performance monitoring on analytic processing from mobile platforms 3. Supports rich visual content search and rendering from mobile platforms 4. Supports mobile device data acquisition, integrity, confidentiality and availability 5. Supports security across mobile devices, platforms, OS’s and Carriers.  6. Support privacy across mobile devices, platforms, OS’s and Carriers. |

# Big Data Reference Architecture (from RA Subgroup)

The NIST Big Data Reference Architecture (RA) shown on Figure 4 represents an agnostic big data system comprised of logical functional blocks interconnected by interoperability interfaces (a.k.a., services). The blocks represent functional roles in the Big Data ecosystem and are called “Providers” to indicate that they provide or implement a specific technical function within the system.

According to the Big Data taxonomy, a single actor can play multiple roles, and multiple actors can play the same role. This functional RA doesn’t specify the business boundaries between the participating stakeholders or actors, indicating that such roles can reside within the same business entity or can be implemented by different business entities. As such, the RA is applicable to a variety of business environments including tightly-integrated enterprise systems, as well as loosely-coupled vertical industries that rely on the cooperation by independent stakeholders.

Note: As a result, the notion of internal vs. external functional blocks or roles doesn’t apply to this RA. However, for a specific use case, once the roles are associated with specific business stakeholders, the functional blocks would be considered as internal or external - subject to the use case’s point of view.

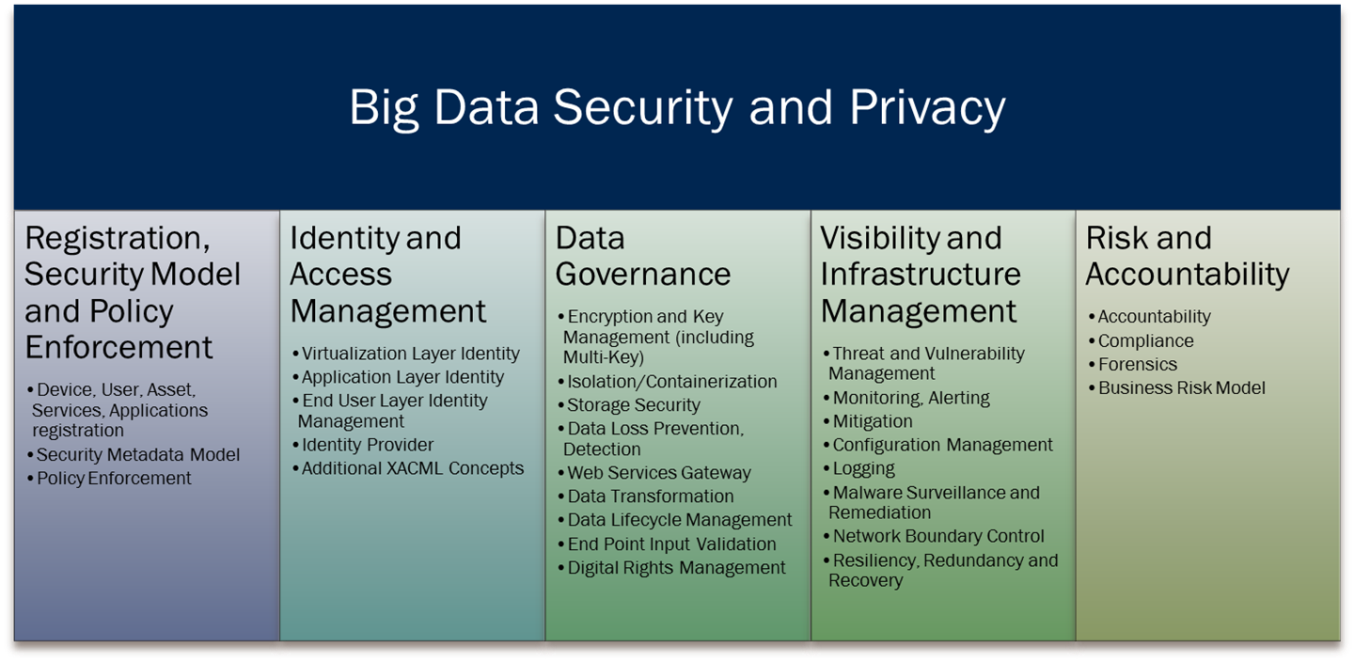


# Big Data Security and Privacy (from SecNPrivacy Subgroup)

The Security and Privacy subgroup attempted to identify security and privacy issues particular to Big Data. Variety, Volume and Velocity that are key elements to Big Data, and, where possible, aspects of these properties directed their attention. That aside in general Big Data Security and Privacy primarily requires that security and privacy implementations scale to meet the Variety, Volume, and Velocity of Big Data.

Certain security and privacy concerns are concentrated in Big Data systems. Big Data projects often encompass heterogeneous components in which a single security scheme has not been designed from the outset. While most security and privacy methods have been designed for batch or online transaction processing systems, Big Data projects increasingly involve one or more streamed data sources, used in conjunction with data at rest, creating unique security and privacy scenarios. Approaches to de-identify personally identifiable information (PII) that were satisfactory prior to Big Data may no longer be adequate. In increased reliance on sensor streams, such as that anticipated with the Internet of Things is expected to create vulnerabilities that were more easily managed before amassed to Big Data scale. Certain types of data thought not to be “machine readable,” such as geospatial and video imaging, will become commodity Big Data sources, often without prior security and privacy protections in place. Issues of veracity, provenance and jurisdiction are greatly magnified in Big Data. Multiple organizations, stakeholders, legal entities, governments and far more members of the citizenry will find data about themselves included in Big Data analytics. Lastly, Big Data conceives of data as permanent *by default*. Security is a fast-moving field with multiple attack vectors and countermeasures. Data may be preserved beyond the lifetime of the security measures designed to protect it.

Figure 5 below categorizes the key security and privacy general requirements by their operational area.



Security and Privacy considerations form a fundamental aspect of the Big Data Reference Architecture. This is geometrically depicted by having a Security and Privacy fabric around the reference architecture components, since it touches all of the components. This way the role of S&P is depicted in the right relation to the components and at the same time does not explode into finer details, which may be more accurate but are best relegated to a more detailed Security Reference Architecture. In addition to the Application and Framework Providers, we also decided to include the Data Provider and Data Consumer into the fabric since at the least they have to agree on the security protocols and mechanisms in place.

Figure 6 goes into details on Security and Privacy requirements at each of the component interfaces, as well as inside the IT Provider.

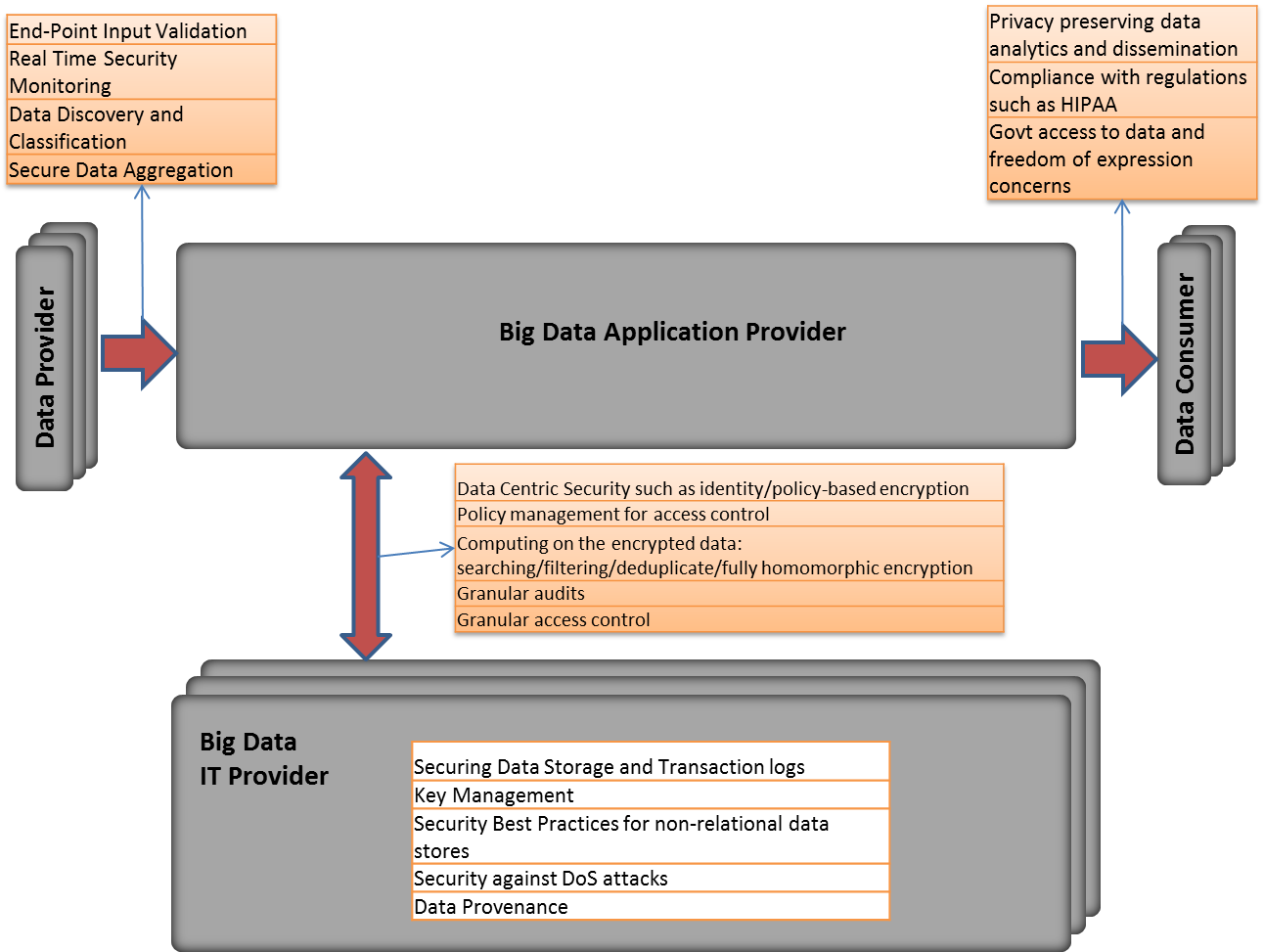


Figure 6. Big Data Security Reference Architecture

# Features and Technology Readiness

## Technology Readiness

Author: Dan

The technological readiness for Big Data serves as metric useful in assessing both the overall maturity of a technology across all implementers as well as the readiness of a technology for broad use within an organization. Technology readiness evaluates readiness types in a manner similar to that of technology readiness in Service-Oriented Architectures (SOA). However, the scale of readiness is adapted to better mimic the growth of open source technologies, notably those which follow models similar to the Apache Software Foundation (ASF). Figure 1 provides a superimposition of the readiness scale on a widely recognized "hype curve." This ensures that organizations which have successfully evaluated and adopted aspects of SOA can apply similar processes to assessing and deploying Big Data technologies.

### Types of Readiness

* **Architecture**: Capabilities concerning the overall architecture of the technology and some parts of the underlying infrastructure
* **Deployment**: Capabilities concerning the architecture realization infrastructure deployment, and tools
* **Information**: Capabilities concerning information management: data models, message formats, master data management, etc.
* **Operations, Administration and Management**: Capabilities concerning post-deployment management and administration of the technology

### Scale of Technological Readiness

1. **Emerging**

* Technology is largely still in research and development
* Access is limited to the developers of the technology
* Research is largely being conducted within academic or commercial laboratories
* Scalability of the technology is not assessed

1. **Incubating**

* Technology is functional outside laboratory environments
* Builds may be unstable
* Release cycles are rapid
* Documentation is sparse or rapidly evolving
* Scalability of the technology is demonstrated but not widely applied

1. **Reference Implementation**

* One or more reference implementations are available
* Reference implementations are usable at scale
* The technology may have limited adoption outside of its core development community
* Documentation is available and mainly accurate

1. **Emerging Adoption**

* Wider adoption beyond the core community of developers
* Proven in a range of applications and environments
* Significant training and documentation is available

1. **Evolving**

* Enhancement-specific implementations may be available
* Tool suites are available to ease interaction with the technology
* The technology competes with others for market share

1. **Standardized**

* Draft standards are in place
* Mature processes exist for implementation
* Best practices are defined

## Organizational Readiness and Adoption

Technological readiness is useful for assessing the maturity of the technology components which make up Big Data implementations. However, successful utilization of Big Data technologies within an organization strongly benefits from an assessment of both the readiness of the organization and its level of adoption with respect to Big Data technologies. As with the domains and measures for the Technology Readiness scale, we choose definitions similar to those used for SOA.

### Types of Readiness

#### Organizational Readiness Domains

* **Business and Strategy:** Capabilities that provide organizational constructs necessary for Big Data initiatives to succeed. These include a clear and compelling business motivation for adopting Big Data technologies, expected benefits, funding models etc.
* **Governance:** The readiness of governance policies and processes to be applied to the technologies adopted as part of a Big Data initiative. Additionally, readiness of governance policies and processes for application to the data managed and operated on as part of a Big Data initiative.
* **Projects, Portfolios, and Services:** Readiness with respect to the planning and implementation of Big Data efforts. Readiness extends to quality and integration of data, as well as readiness for planning and usage of Big Data technology solutions.
* **Organization:** Competence and skills development within an organization regarding the use and management of Big Data technologies. This includes, but is not limited to, readiness within IT departments (e.g., service delivery, security, and infrastructure) and analyst groups (e.g. methodologies, integration strategies, etc.).

### Scale of Organizational Readiness

1. **No Big Data**

* No awareness or efforts around Big Data exist in the organization

1. **Ad Hoc**

* Awareness of Big Data exists
* Some groups are building solutions
* No Big Data plan is being followed

1. **Opportunistic**

* An approach to building Big Data solutions is being determined
* The approach is opportunistically applied, but is not widely accepted or adopted within the organization

1. **Systematic**

* The organizational approach to Big Data has been reviewed and accepted by multiple affected parties.
* The approach is repeatable throughout the organization and nearly-always followed.

1. **Managed**

* Metrics have been defined and are routinely collected for Big Data projects
* Defined metrics are routinely assessed and provide insight into the effectiveness of Big Data projects

1. **Optimized**

* Metrics are always gathered and assessed to incrementally improve Big Data capabilities within the organization.
* Guidelines and assets are maintained to ensure relevancy and correctness

### Scale of Organizational Adoption

1. **No Adoption**

* No current adoption of Big Data technologies within the organization

1. **Project**

* Individual projects implement Big Data technologies as they are appropriate

1. **Program**

* A small group of projects share an implementation of Big Data technologies
* The group of projects share a single management structure and are smaller than a business unit

1. **Divisional**

* Big Data technologies are implemented consistently across a business unit

1. **Cross-Divisional**

* Big Data technologies are consistently implemented by multiple divisions with a common approach
* Big Data technologies across divisions are at an organizational readiness level of Systematic or higher

1. **Enterprise**

* Big Data technologies are implemented consistently across the enterprise
* Organizational readiness is at level of Systematic or higher

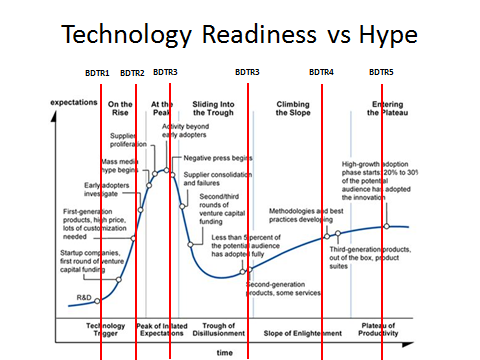


Figure Technology Readiness levels visualized along Gartner's "hype curve."

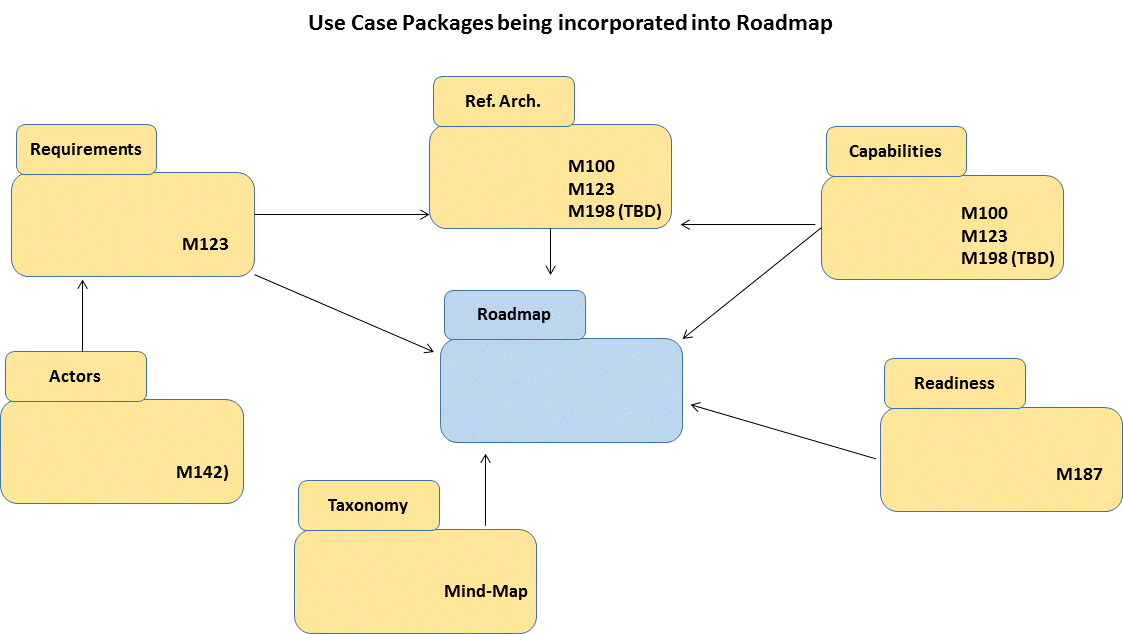
## Features Summary

Author: Bruno

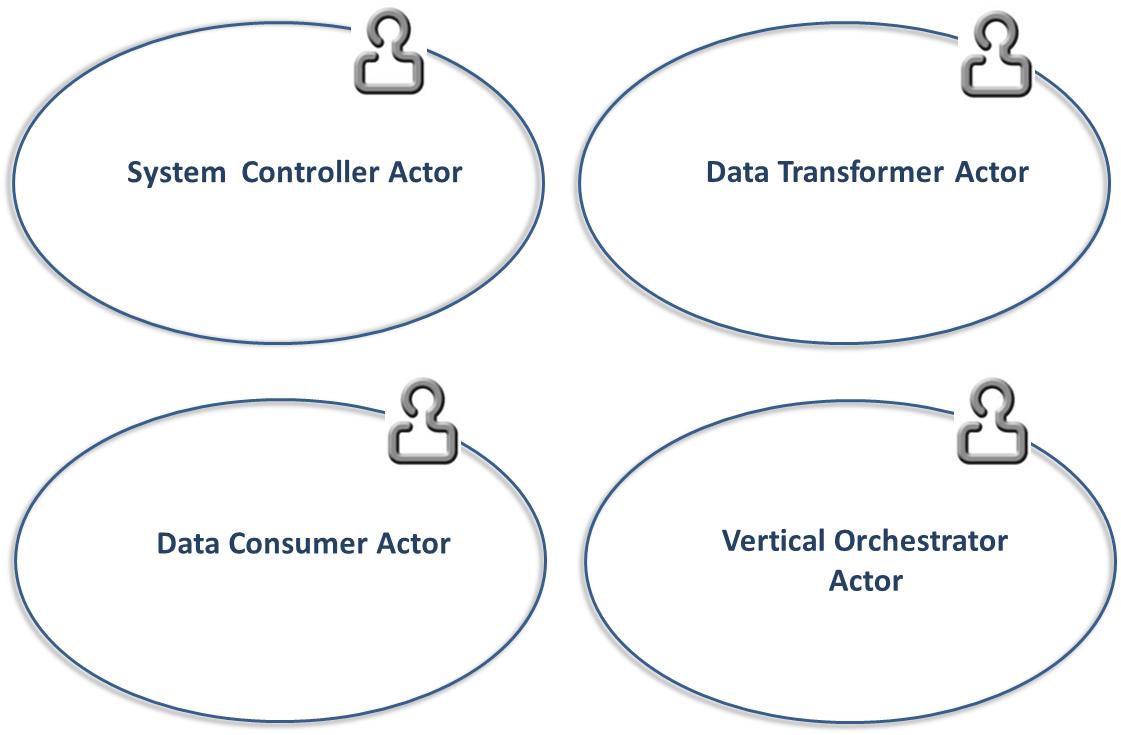
The NIST Roadmap is composed of features that outline the current and future state of Big Data. These features consists of four primary categories: 1) Data Services, 2) Usage Services, 3) Capabilities, and, 4) Vertical Orchestrator. These four categories also mirror the respective NIST Big Data Reference Architecture and Taxonomy. Within these two categories are the Top Nine Features.

What is the importance of having the Roadmap portray features? The ability for technical and business stakeholders to view the current and future state of features enables such individuals to better make decisions in using Big Data. The three ‘dials’ of technology, problems, and, end users is that all three are in constant change. In addition, they all interplay with one another. When one moves – the other are effected. Leaders must then pivot. The NIST Big Data Roadmap provides the best available snapshot in time of this moving target.

The reasoning of how the NIST Working Group arrived at these four categories and nine features is primarily due to the rationalizing of the Big Data landscape. The four categories were arrived at by aligning to the Reference Architecture. The nine Roadmap features were an output from the mind-share of the Reference Architecture, Requirements [Use Cases], [architectural] Capabilities, Actors, Taxonomy, and, [Big Data] Readiness. The below diagram displays the inputs and relationships between the NIST artifacts and the Roadmap.



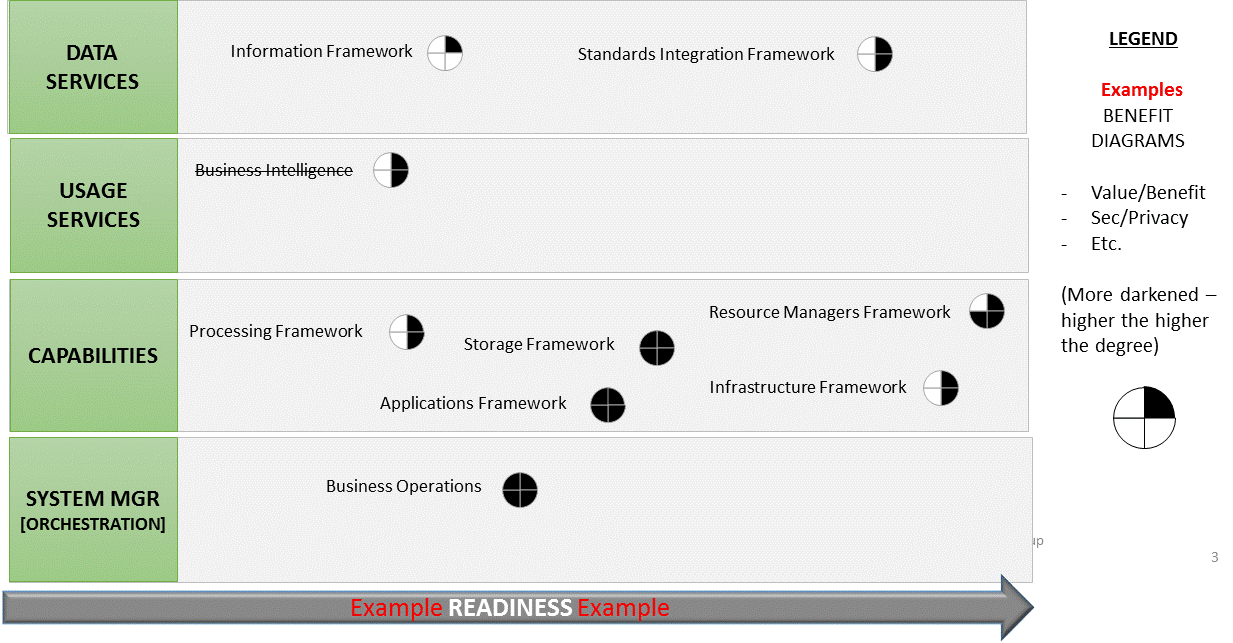
Realizing the centricity of the end-user is critical, the Roadmap has inherited the four actor groups from the NIST Big Data artifacts. These four abstract actors are: 1) System Controller, 2) Data Transformation, 3) Data Consumer, and, 4) Vertical Orchestrator (see diagram below). Very similar to Unified Modeling Language (UML) standards, the Actors are abstract and can be not only individuals but also systems. Actors can also fulfill more than just one role.



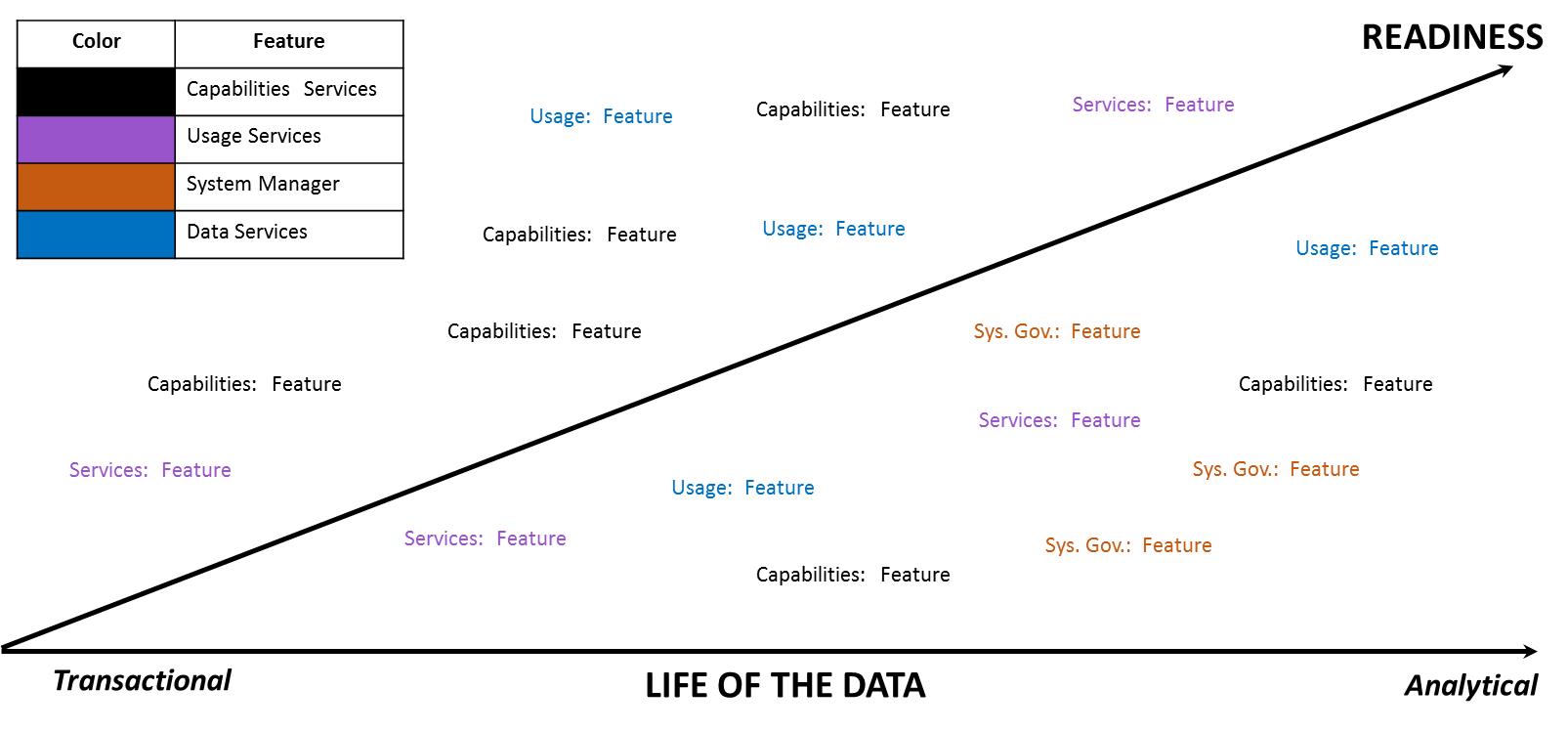
The below tables outlines the value statements for each of the Big Data features. In addition, the NIST Working Group has mapped each feature to technology and organizational readiness.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Feature | Value Statement | Roles | Readiness | RA Mapping |
| Storage Framework | Storage Framework defines how big data is logically organized, distributed, and stored. The volume and velocity of Big Data frequently means that traditional (file systems, RDBMS) solutions will not hold up to one or both of these attributes. | TBD | TBD | Capabilities |
| Processing Framework | Processing Frameworks defines how data is operated on in the Big Data environment. The volume or velocity of Big Data often means that analysis of the data requires more resources (memory, cpu cycles) than are available on a single compute node and that the processing of the data must be distributed and coordinated across many nodes. | TBD | TBD | Capabilities |
| Resource Managers Framework | Because many Big Data storage and processing Frameworks are distributed and no single storage and processing solution may meet the needs of the end user, resource management solutions are required to manage and allocate resources across disparate frameworks. | TBD | TBD | Capabilities |
| Infrastructure Architecture | Big Data requires the ability to operate with sufficient network and infrastructure backbone. For Big Data to deploy, it is critical that the Infrastructure Architecture has been right-sized. | TBD | TBD | Capabilities |
| Information Architecture | Prior to any Big Data decision, the data itself needs to be reviewed for its informational value. | TBD | TBD | Data Services |
| Standards Integration Framework | Integration with appropriate standards (de jure, consortia, reference implementation, open source implementation, etc.) can assist both in cross-product integration and cross-product knowledge. Identifying which standards efforts address architectural requirements and which requirements are not currently being addressed provides input for future standards efforts. | TBD | TBD | Data Services |
| Applications Framework | The building blocks of applications are data. The Application Lifecycle Management needs to take into consideration how applications will interact with a Big Data solution. The presenting of data into information, intelligence, and, insight is typically the end value of Big Data. | TBD | TBD | Capabilities |
| Business Operations | Big Data is more than just technology, but also a cultural and organizational transformation. Business Operations need to be able to strategize, deploy, adopt, and, operate Big Data solutions. | TBD | TBD | Vertical Orchestrator |

The top features of the Roadmap can also be viewed visually. Two examples of displaying the NIST Big Data Roadmap and relevant decision making criteria are provided below. The goal of these diagrams is to enable decision-makers to see as best ‘around-the-corner’ as possible for their Big Data discussions. Alignment with the other NIST Big Data artifacts have taken place, providing the reader with the ability to toggle to other documents for reference and discussion drill-down. The below diagram groups Roadmap features into the four category ‘swimlanes’, with examples of the features spanning readiness. The pie charts are to provide either the level of value, security, or, privacy.



The second diagram displaying the Roadmap visually (below) is a scatter diagram with an X and Y axis. It provides an ‘X’ axis showing the life of the data. The ‘Y’ axis provides the level of readiness. The Roadmap features are shown by the color of the one of the four categories. They are placed in the diagram as to where they sit in terms of the longitudinal use of the data – transactional to analytical – and the level of their organizational and architectural readiness for adoption. The below diagram lands these features below as purely as examples.



In conclusion, the Roadmap outlines the future of Big Data with the best available for-sight as possible. It captures the relation of features to decision-making criteria, such as: readiness, value, security, and longitudinal use of data. Additional value-add is the incorporation of the Roadmap with other NIST Big Data artifacts.

## Feature 1: Storage Framework

We need to build off of the value statement described in the table above in the “Features Summary” section to develop a case for the feature. Why it is important in the Big Data Ecosystem? What readiness level do we need it to get to and/or what should this feature be like in 5 years to meet the collective needs of Government / Academia / Industry? What needs to happen and how do we get there?

Author: Dave

*Storage Framework defines how Big Data is logically organized, distributed, and stored. The volume and velocity of Big Data frequently means that traditional (file systems, RDBMS) solutions will not hold up to one or both of these attributes.*

Storage Frameworks have and continue to undergo dramatic changes as the need to capture, store, process, and archive ever-larger amounts of data. The nature of this increase covers both the total volume of data artifacts - 9100 Tweets per second (Twitter Statistics, 2013) to the extreme size of individual artifacts – 110 megapixel images from the Vigilant Stare Wide Area Persistent Surveillance (WAPS) Platform (Persistence On Patrol, 2013). In order to support the processing of data across this continuum unique and novel approaches are frequently required to support not just the storage but the indexing for access/processing but also the preservation/backup and transfer of the data.

Typically storage frameworks consist of two interdependent aspects, the physical organization of the data on the media, and the logical organization of the data within the physical layout. For the purposes of this roadmap we are specifically focusing on persistent storage approaches but it should be noted that most of these approaches could be implemented in non-persistent storage (e.g. RAM disk). In addition, we are not going to delve into the physical media itself. While today, the primary media are traditional spinning magnetic disks and Solid State Disks built on flash memory technology new storage media technologies such as holographic, quantum, and nano-bubble storage are under development and maturing rapidly. In addition the density of both current prevalent data storage technologies and emerging technologies continue to increase roughly along Moore’s Law. From a Big Data perspective this generally means that more data can be stored in a smaller footprint and that in general (especially where mechanical processes like disk head seeks are involved) access times will continue to decrease and throughput increase. As these new media types mature and their performance characteristics are understood they should provide drop in replacements for existing storage media.

### Physical Storage Frameworks

The physical organization of storage generally follows the continuum from local to distributed as shown in figure 6.4-1 below with their associated technology readiness indicated by the number in the circle.



There are two aspects of these technologies that directly influence their suitability for Big Data solutions. First there is capacity (dealing with the volume of the data). Local disks/filesystems are specifically limited by the size of the available media. HW/SW Raid solutions (in this case local to a processing node) help that scaling by allowing multiple pieces of media to be treated as a single device. That approach however is limited by the physical dimension of the media and the number of devices the node can accept. SAN and NAS implementations (often known as shared disk solutions) remove that limit by consolidating storage into a storage specific device. However, they start to run into the second influencing factor which is transfer bandwidth. While both network and I/O interfaces are getting faster and many implementations support multiple transfer channels I/O bandwidth can be a limiting factor. In addition, despite the redundancies provided by RAID, Hot Spares, multiple power supplies, and multiple controllers these boxes can often become I/O bottle necks or single points of failure in an enterprise. Distributed Filesystems (also known as cluster file systems) seek to overcome these issues by combining I/O throughput through multiple devices(spindles) on each node with redundancy and failover by mirroring/replicating data at the block level across multiple nodes. This specifically, is designed to allow the use of heterogeneous commodity hardware across the Big Data cluster. Thus, if a single drive or an entire node should fail no data is lost because it is replicated on other nodes and throughput is only minimally impacted as that processing can be moved to those other nodes. In addition, replication allows for high levels of concurrency for reading data and for initial writes. Updates and transaction style changes tend to be an issue for many distributed file systems because latency in creating replicated blocks will create consistency issues (e.g. a block is changed but another node reads the old data before it is replicated).

Unlike the other technologies described here which implement a traditional filesystem approach, Distributed Object Stores (sometimes called Global Object Stores) present a flat name space with a global id (GUID) for any given chunk of data. Data in the store is located generally through a query against a meta-data catalog that returns the associated GUID(s). The underlying implementation of the software generally knows from the GUID where that particular chunk of data is stored. These object stores are being developed and marketed for storage of very large data objects from complete data sets (which Amazon’s S3 exposes) to large individual objects (high resolution images in the 10s of GB size range). The biggest limitation of these stores for Big Datatends to be network throughput (aka: speed) that can be a limiting factor since many require the object to be accessed in total. Future trends however point to the concept of being able to send the computation/application to the data versus needing to bring the data to the application.

From a maturity perspective the two key areas where distributed file systems would be expected to improve are on random write I/O performance and consistency and the generation of at least IETF RFC level standards such as are available today for NFS. Distributed object stores while currently available and operational (Amazon S3) and part of the roadmap for large organizations such as the National Geopspatial Intelligence Agency (NGA) currently are essentially stove pipe/proprietary implementations. In order for them to become prevalent within Big Data ecosystems there will need to be some level of interoperability available (through standardized APIs), standards based approaches for data discovery, and probably most importantly standards based approaches that allow the application to be transferred over the grid and run local to the data versus the need for the data to be transferred to the application.

### Logical Data Distribution

In most aspects, the logical distribution/organization of data in Big Data storage frameworks mirrors what is common for most legacy systems. Figure 6.4.2 below shows a brief overview of data organizations approaches for Big Data.

As mentioned above many Big Data logical storage organizations leverage the common file system concept (where chunks of data are organized into a hierarchical namespace of directories) as their base and then implement various indexing methods within the individual files. This allows many of these approaches to be run both on simple local storage file systems for testing purposes or on fully distributed file systems for scale.

#### File Systems

Many Big Data processing frameworks and applications are content to access their data directly from an underlying file systems. In almost all cases, the file systems implement some level of the POSIX standards for permissions and the associated file operations. This allows other higher level frameworks for Indexing or Processing to operate relatively transparent to whether the underlying file system is local or fully distributed. Within, files systems there is really nothing new or novel in the storage of data. It can be text or binary data, fixed length records, or some sort of delimited structure (e.g. comma separated values, XML). Several of these file system implementations also support data compression and encryption at various levels. The one major caveat to this is that for distributed block based file systems the compression/encryption must be splittable and allow any given block to be decompressed/decrypted out of sequence and without access to the other blocks. For record oriented storage (delimited or fixed length) this generally is not a problem unless individual records can exceed a block size. Some distributed file system implementations provide compression at the volume or directory level and implement it below the logical block level (e.g. when a block is read from the file system it is decompressed/decrypted before being returned). Because of this simplicity, familiarity, and portability delimited files are frequently the default storage format in many Big Data implementations. The tradeoff for this is IO efficiency (aka: speed). While individual blocks in a distributed file system might be accessed in parallel, each block still needs to be read in sequence. In the case of a delimited file if you are interested in only the last field of certain records with maybe 100s of fields that is a lot of wasted IO and processing bandwidth.

Binary formats tend to be application or implementation specific. While they can offer much more efficient access both due to smaller data sizes (integers are 2-4 bytes in binary while they are 1byte per digit in ASCII) they offer limited portability between different implementations. At least one popular distributed file system provides its own standard binary format which at least allows data to be portable between multiple applications without additional software. That said the bulk of the indexed data organization approaches discussed below leverage binary formats for efficiency.

#### Indexed Storage Organization

The very nature of big data (volume and velocity primarily) practically drive requirements to some form of indexing structure. The volume requires that specific elements of data can be locate quickly without scanning across the entire dataset. The velocity, also requires that data can be located quickly either for matching (e.g. does any incoming data match something in my existing data set) or to know where to write/update new data.

The choice of a particular indexing method or methods depends mostly on your data and the nature of the application you are trying to implement. For example, graph data (vertexes, edges, and properties) can be easily represented in flat text files as vertex,edge pairs, edge, vertex,vertex triples, or vertex, and edge list records. However, processing this data efficiently would require potentially loading the entire data set into memory or being able to distribute the application and data set across multiple nodes so a portion of the graph is in memory on each node. That then of course would require the nodes to communicate when graph sections have vertexes that connect with vertexes on other processing nodes. This is perfectly acceptable for some graph applications such as shortest path especially when the graph is static. And in fact some graph processing frameworks operate using this exact model. However, if the graph is dynamic or you need to quickly search or match to a portion of the graph then this approach becomes infeasible for large scale graphs requiring a specialized graph storage framework.

The indexing approaches described below tend to be classified by the features provided typically in the implementation specifically - the complexity of the data structures that can be stored, how well they can process links between data, and how easily they support multiple access patterns as shown in Figure 6.2.2.2-1 below. Since any of these features can be implemented in custom application code the values portrayed represent approximant norms. Key-Value stores for example work really real for data that is only accessed through a single key, whose values can be expressed in a single flat structure and where multiple records do not need to be related. Document stores, can support very complex structures of arbitrary width and tend to be indexed for access via multiple document properties but do not tend to support inter-record relationships well.

Please Note: In reality, the specific implementations for each storage approach vary significantly enough that all of the values for the features represented are really ranges. For example, relational data storage implementations are supporting increasingly complex data structures and there is work going on to add more flexible access patterns natively in bigtable columnar implementations. Within big data the performance of each of these features tends to drive the scalability of that approach depending on the problem being solved. For example, if my problem is to locate a single piece of data for a unique key then Key-value stores will scale really well. If on the other hand my problem requires general navigation of the relationships between multiple data records then a graph storage model will likely provide the best performance.

The following paragraphs will describe several indexing approaches for storing big data and the advantages and issues commonly found for each one.

##### Relational Storage Models

This model is perhaps the most familiar to folks as the basic concept has existed since the 1950s and the Structured Query Language (SQL) is a mature standard for manipulating (search, insert, update, delete) relational data. In the relational model, data is stored as rows with each field representing a column organized into Table based on the logical data organization. The problem with relational storage models and big data is the join between one or more tables. While the size of 2 or more tables of data individually might be small the join (or relational matches) between those tables will generate exponentially more records. The appeal of this model for organizations just adopting big data is its familiarity. The pit falls are some of the limitations and more importantly the tendency to adopt standard RDBMS practices (high normalization, detailed and specific indexes) and performance expectations

Big data implementations of relational storage models are relatively mature and have been adopted by a number of organizations. They are also maturing very rapidly with new implementations focusing on improved response time. Many big data implementations take a brute force approach to scaling relational queries. Essentially, queries are broken into stages but more importantly processing of the input tables is distributed across multiple nodes (often as a map reduce job). The actual storage of the data can be flat files (delimited or fixed length) where each record/line in the file represents a row in a table. Increasingly however these implementations are adopting binary storage formats optimized for distributed file systems. These formats will often use block level indexes and column oriented organization of the data to allow individual fields to be accessed in records without needing to read the entire record. Despite this, most Big Data Relational storage models are still “batch oriented” systems designed for very complex queries which generate very large intermediate cross-product matrices from joins so even the simplest query can required 10s of seconds to complete. There is significant work going on and emerging implementations that are seeking to provide a more interactive response and interface.

Early implementations only provided limited data types and little or no support for indexes. However, most current implementations have support for complex data structures and basic indexes. However, while the query planners/optimizers for most modern RDBMS systems are very mature and implement cost-based optimization through statistics on the data the query planners/optimizers in many big data implementations remain fairly simple and rule-based in nature. While for batch oriented systems this generally acceptable (since the scale of processing the big data in general can be orders of magnitude more an impact) any attempt to provide interactive response will need very advanced optimizations so that (at least for queries) only the most likely data to be returned is actually searched. This of course leads to the single most serious draw back with many of these implementations. Since distributed processing and storage are essential for achieving scalability these implementations are directly limited by the CAP (Consistency, Availability, and Partition Tolerance) theorem. Many in fact provide what is generally referred to as a t-eventual consistency which means that barring any updates to a piece of data all nodes in the distributed system will eventually return the most recent value. This level of consistency is typically fine for Data Warehousing applications where data is infrequently updated and updates are generally done in bulk. However, transaction oriented databases typically require some level of ACID (Atomicity, Consistency, Isolation, Durability) compliance to insure that all transactions are handled reliably and conflicts are resolved in a consistent manner. There are a number of both industry and open source initiatives looking to bring this type of capability to Big Data relational storage frameworks. One approach is to essentially layer a traditional RDBMS on top of an existing distributed file system implementation. While vendors claim that this approach means that the overall technology is mature a great deal of research and implementation experience is needed before the complete performance characteristics of these implementations are known.

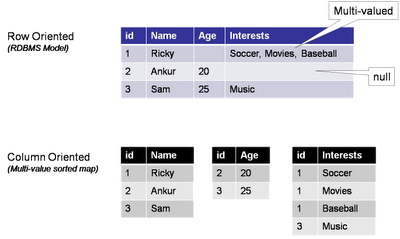
##### Key-Value Storage Models

Key-value stores are one of the oldest and mature data indexing models. In fact, the principles of key value stores underpin all the other storage and indexing models. From a Big Data perspective, these stores effectively represent random access memory models. While the data stored in the values can be arbitrarily complex in structure all the handling of that complexity must be provided by the application with the storage implementation often providing back just a pointer to a block of data. Key-Value stores also tend to work best for 1-1 relationships (e.g. each key relates to a single value) but can also be effective for keys mapping to lists of homogeneous values. When keys map multiple values of heterogeneous types/structures or when values from one key need to be joined against values for a different or the same key then custom application logic is required. It is the requirement for this custom logic that often prevents Key-Value stores from scaling effectively for certain problems. However, depending on the problem certain processing architectures can make effective use of distributed key-value stores. Key-value stores generally deal well with updates when the mapping is one to one and the size/length of the value data does not change. The ability of key-value stores to handle inserts is generally dependent on the underlying implementation. Key-value stores also generally require significant effort (either manual or computational) to deal with changes to the underlying data structure of the values.

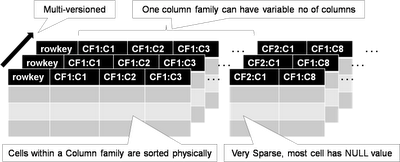
Distributed key-value stores are the most frequent implementation utilized in big data applications. One problem that must always be addressed (but is not unique to key-value implementations) is the distribution of keys across over the space of possible key values. Specifically, keys must be chosen carefully to avoid skew in the distribution of the data across the cluster. When data is heavily skewed to a small range it can result in computation hot spots across the cluster if the implementation is attempting to optimize data locality. If the data is dynamic (new keys being added) for such an implementation then it is likely that at some point the data will require rebalancing across the cluster. Non-locality optimizing implementations employ various sorts of hashing, random, or round-robin approaches to data distribution and don’t tend to suffer from skew and hot spots. However, they perform especially poorly on problems requiring aggregation across the data set.

##### Columnar Storage Models

Much of the hype associated with Big Data came with the publication of the Big Table paper by Google in 2006 (Chang, et al., 2006) but column oriented storage models like Bigtable are not new to even big data and have been stalwarts of the data warehousing domain for many years. Unlike traditional relational data that store data by rows of related values, columnar stores organize data in groups of like values. The difference here is subtle but in relational databases an entire group of columns are tied to some primary-key (frequently one or more of the columns) to create a record. In columnar, the value of every column is a key and like column values point to the associated rows. The simplest instance of a columnar store is little more than a key-value store with the key and value roles reversed. In many ways, columnar data stores look very similar to indexes in relational databases. Figure 6.4.2.2.2-1 below shows the basic differences between row oriented and column oriented stores.



In addition, implementations of columnar stores that follow the Google Bigtable model introduce an additional level of segmentation beyond the table, row, column model of the relational model. That is called the column family. In those implementations rows have a fixed set of column families but within a column family each row can have a variable set of columns. This is illustrated in figure 6.4.2.2.2-2 below.



The key distinction in the implementation of columnar store over relational stores is that data is high de-normalized for column stores and that while for relational stores every record contains some value (perhaps NULL) for each column, in columnar store the column is only present if there is data for one or more rows. This is why many column oriented stores are referred to as sparse storage models. Data for each column family is physically stored together on disk sorted by rowed, column name and timestamp. The last (timestamp) is there because the Bigtable model also includes the concept of versioning. Every, RowKey, Column Family, Column triple is stored with either a system generated or user provided Timestamp. This allows users to quickly retrieve the most recent value for a column (the default), the specific value for a column by timestamp, or all values for a column. The last is most useful because it permits very rapid temporal analysis on data in a column.

Because data for a given column is stored together two key benefits are achieved. First aggregation of the data in that column requires only the values for that column to be read. Conversely in a relational system the entire row (at least up to the column) needs to be read (which if the row is long and the column at the end it could be lots of data). Secondly, updates to a single column do not require the data for the rest of the row to be read/written. Also, because all the data in a column is uniform, data can be compressed much more efficiently. Often only a single copy of the value for a column is stored followed by the row keys where that value exists. And while deletes of an entire column is very efficient, deletes of an entire record are extremely expensive. This is why historically column oriented stores have been applied to OLAP style applications while relational stores were applied to OLTP requirements.

Recently, security has been a major focus of existing column implementations primarily due to the release by the National Security Agency (NSA) of it’s Bigtable implementation to the open source community. A key advantage of the NSA implementation and other recently announced implementations is the availability of security controls at the individual cell level. With these implementations a given user might have access to only certain cells in group based potentially based on the value of those or other cells.

There are several very mature distributed column oriented implementations available today from both open source groups and commercial foundations. These have been implemented and operational across a wide range of businesses and government organizations. Emerging are hybrid capabilities that implement relational access methods (e.g. SQL) on top of BigTable/Columnar storage models. Also, relational implementations are adopting columnar oriented physical storage models to provide more efficient access for big data OLAP like aggregations and analytics.

##### Document

Document storage approaches have been around for some time and popularized by the need to quickly search large amounts of unstructured data. Modern, document stores have evolved to include extensive search and indexing capabilities for structured data and metadata and why they are often referred to as semi-structured data stores. Within a document-oriented data store each “document” encapsulates and encodes the metadata, fields, and any other representations of that record. While somewhat analogous to a row in a relational table one-reason document stores evolved and have gained in popularity is that most implementations do not enforce a fixed or constant schema. While best practices hold that groups of documents should be logically related and contain similar data there is no requirement that they be alike or that any two documents even contain the same fields. That is one reason that document stores are frequently popular for data sets which have sparsely populated fields since there is far less overhead normally than traditional RDBMS systems where null value columns in records are actually stored. Groups of documents within these types of stores are generally referred to as collections and like key-value stores some sort of unique key references each document.

In modern implementations documents can be built of arbitrarily nested structures and can include variable length arrays and in some cases executable scripts/code (which has significant security and privacy implications). Most document-store implementations also support additional indexes on other fields or properties within each document with many implementing specialized index types for sparse data, geospatial data, and text.

When modeling data into document-stores the preferred approach is to de-normalize the data as much as possible and embed all one-to-one and most one-to-many relationships within a singled document. This allows for updates to documents to be atomic operations which keeps referential integrity between the documents. The most common case where references between documents should be use is when there are data elements that occur frequently across sets of documents and whose relationship to those documents is static. As an example, the publisher of a given book edition does not change and there are far fewer publishers than there are books. It would not make sense to embed all the publisher information into each book document. Rather the book document would contain a reference to the unique key for the publisher. Since for that edition of the book the reference will never change and so there is no danger of loss of referential integrity. Thus information about the publisher (address for example) can be updated in a single atomic operation the same as the book. Where this information embedded it would need to be updated in every book document with that publisher.

In the big data realm document stores scale horizontally through the use of partitioning or sharding to distribute portions of the collection across multiple nodes. This partitioning can be round-robin based insuring an even distribution of data or content/key based so that data locality is maintained for similar data. Depending on the application required the choice of partitioning key like with any data base can have significant impacts on performance especially where aggregation functions are concerned.

There are no standard query languages for document store implementations with most using a language derived from their internal document representation (e.g. JSON, XML).

##### Graph

While social networking sites like Facebook and LinkedIn have certainly driven the visibility of and evolution of graph stores (and processing as discussed below), graph stores have been a critical part of many problem domains from military intelligence and counter terrorism to route planning/navigation and the semantic web for years. Graph stores represent data as a series of nodes, edges, and properties on those. Analytics against graph stores include very basic shortest path and page ranking to entity disambiguation and graph matching.

Graph databases typically store two types of objects nodes and relationships as show in figure 6.4.2.2.4-1 below. Nodes represents objects in the problem domain that are being analyzed be they people, places, organizations, accounts, etc. Relationships describe those objects in the domain relate to each other. Relationships can be non-directional/bi-directional but are typically expressed as uni-directional in order to provide more richness and expressiveness to the relationships. Hence, between two people nodes where they are father and son there would be two relationships. One “is father of” going from the father node to the son node, and the other from the son to the father of “is son of”. In addition, nodes and relationships can have properties or attributes. This is typically descriptive data about the element. For people it might be name, birthdate, etc. For locations it might be an address or geospatial coordinate. For a relationship like a phone call it could be the date, time of the call, the duration of the call. Within graphs relationships are not always equal or have the same strength. Thus relationship often have one or more weight, cost, or confidence attributes. A strong relationship between people might have a high weight because they have known each other for years and communicate every day. A relationship where two people just met would have a low weight. The distance between nodes (be it a physical distance or a difficulty) is often expressed as a cost attribute on a relation in order to allow computation of true shortest paths across a graph. In military intelligence applications, relationships between nodes in a terrorist or command and control network might only be suspected or have not been completely verified so those relationships would have confidence attributes. Also, properties on nodes may also have confidence factors associated with them although in those cases the property can be decomposed into it’s own node and tied with a relationship. Graph storage approaches can actually be viewed as a specialized implementation of a document storage scheme with two types of documents (nodes and relationships). In addition, one of the most critical elements in analyzing graph data is locating the node or edge in the graph where you want to begin the analysis. To accomplish this most graph databases implement indexes on the node or edge properties. Unlike, relational and other data storage approaches most graph databases tend to use artificial/pseudo keys or guides to uniquely identify nodes and edges. This allows attributes/properties to be easily changed due to both actual changes in the data (someone changed their name) or as more information is found out (e.g. we get a better location for some item or event) without needing to change the pointers two/from relationships.



The problem with graphs in the Big Data realm is that they grow to be too big to fit into memory on a single node and by their typically chaotic nature (few real world graphs follow well defined patterns) makes their partitioning for a distributed implementation problematic. While distance between or closeness of nodes would seem like a straight forward partitioning approach there are multiple issues which must be addressed. First, would be balancing of data. Graphs often tend to have large clusters of data very dense in a given area thus leading to essentially imbalances and hot spots in processing. Second, no matter how you distribute the graph there are connections (edges) that will cross the boundaries. That typically requires that nodes know about or how to access the data on other nodes and requires inter-node data transfer or communication. This makes the choice of processing architectures for graph data especially critical. Architectures that do not have inter-node communication/messaging tend not to work well for most graph problems. Typically, distributed architectures for processing graphs assign chunks of the graph to nodes then the nodes use messaging approaches to communicate changes in the graph or the value of certain calculations along a path.

Even small graphs quickly elevate into the realm of big data when one is looking for patterns or distances across more than one or two degrees of separation between nodes. Depending on the density of the graph, this can quickly cause a combinatorial explosion in the number of conditions/patterns that need to be tested.

A specialized implementation of a graph store known as the Resource Description Framework (RDF) is part of a family of specifications from the World Wide Web Consortium (W3C) that is often directly associated with Semantic Web and associated concepts. RDF triples as they are known consist of a Subject (Mr. X), a predicate (lives at), and an object (Mockingbird Lane). Thus a collection of RDF triples represents and directed labeled graph. The contents of RDF stores are frequently described using formal ontology languages like OWL or the RDF Schema (RDFS) language, which establish the semantic meanings and models of the underlying data. To support better horizontal integration (Smith, Malyuta, Mandirck, Fu, Parent, & Patel, 2012) of heterogeneous data sets extensions to the RDF concept such as the Data Description Framework (DDF) (Yoakum-Stover & Malyuta, 2008) have been proposed which add additional types to better support semantic interoperability and analysis.

Graph data stores currently lack any form of standardized APIs or query languages. However, the W3C has developed the SPARQL query language for RDF which is currently in a recommendation status and there are several frameworks such as Sesame which are gaining popularity for working with RDF and other graph oriented data stores.

## Feature 2: Processing Framework

We need to build off of the value statement described in the table above in the “Features Summary” section to develop a case for the feature. Why it is important in the Big Data Ecosystem? What readiness level do we need it to get to and/or what should this feature be like in 5 years to meet the collective needs of Government / Academia / Industry? What needs to happen and how do we get there?

Author: Dave

The processing frameworks for big data provide the necessary infrastructure software to support the implementation of applications which can deal with the volume, velocity, and variety of data. Typically processing frameworks are categorized based on whether they support batch or interactive processing. This is generally viewed from the user or output perspective (e.g. how fast does a user get a response to a request). However, in reality Big Data processing frameworks actually have three distinct processing phases that should be addressed. These generally brake down to data ingestion, data analysis, and data dissemination which closely follow the flow of data through the architecture. For example, there may be a use case where the data comes into the system at high velocity and the end user needs to be able to quickly retrieve a summary of the prior day’s data. In this case, the ingestion of the data into the system needs to be NRT and keep up with the data stream, the analysis portion may or may not be incremental (e.g. as the data is ingested) but could be a batch process that kicks off at midnight or some combination, but the retrieval (read visualization) of the data needs to be interactive. Depending on the specifics of the use - the transformation of the data may take place at any point during its transit through the system. For example, the ingestion phase may just seek to write down the data as quickly as possible, or it may run some foundational analysis to track things like min, max, avg., etc. since those can be incrementally computed. The core processing job may simply compute a matrix of data or may actually generate some rendering like a heatmap to permit rapid display, the dissemination portion almost certainly does some portion of the rendering but how much depends on the nature of the data and the visualization.

For the purposes of this discussion, most analytic frameworks can be described based on where they are primarily employed within the information flow as illustrated in figure 6.5-1 below.



The green shading above illustrates the general sensitivity of that phase of the processing to latency which is defined as the time from when a request or piece of data arrives at a system until its processing/delivery is complete. For Big Data the ingestion may or may not require near real time performance to keep up with the data flow, and some types of analytics (specifically those categorized as Complex Event Processing) may or may not require that type of processing. At the far right generally sits the data consumer depending upon the use case and application batch responses (e.g. a nightly report is emailed) may be sufficient. In other, cases the user may be willing to wait minutes for the results of a query to be returned, or they may need immediate alerting when critical information arrives at the system. Another way to look at this is that batch analytics tend to better support long term strategic decision making where the overall view or direction is not going to be affected by a recent change to some portion of the underlying data. Streaming analytics are better suited for tactical decision making where new data needs to be acted upon immediately. A primary use case for this would be electronic trading on stock exchanges where the window to act of a given piece of data can be measured in microseconds.

Typically Big Data discussions focus around batch and streaming frameworks for analytics, however retrieval frameworks which provide interactive access to big data are becoming a more prevalent. Of course the lines between these categories are not solid or distinct with some frameworks providing aspects of each element.

### Batch Frameworks

Batch frameworks whose roots stem from mainframe processing days are some of the most prevalent and mature components of a big data architecture simply because the volume of data typically simply required a long time to process. Batch frameworks ideally are not tied to a particular algorithm or even algorithm type but rather provide a programming model where multiple classes of algorithms can be implemented. Also, when discussed in terms of Big Data these processing models are frequently distributed across multiple nodes of a cluster. They are routinely differentiated by the amount of data sharing between processes/activities with-in the model.

Two of the best-known batch processing models for are Map/Reduce (M/R) and Bulk Synchronous Parallel (BSP) will be described in more detail below.

In 2004, Phillip Colella working on DARPA’s High Productivity Computing Systems (HPCS) program developed a list of algorithms for simulation in the physical sciences that became known as the “Seven Dwarfs” (Colella, 2004). More recently David Patterson and Katherine Yelick of the University of California – Berkley modified and extended this list to 13 shown in the table below based on the definition where “A dwarf is an algorithmic method that computes a pattern of computation and communication” (Patterson & Yelick).

|  |  |
| --- | --- |
| Dense Linear Algebra\* | Combinational Logic |
| Sparse Linear Algebra\* | Graph Traversal |
| Spectral methods | Dynamic Programming |
| N-Body Methods | Backtrack and Branch-and-Bound |
| Structured Grids\* | Graphical Models |
| Unstructured Grids\* | Finite Stat Machines |
| MapReduce | \*One of the original 7 dwarfs (removed were Fast Fourier Transform, Particles, and Monte Carlo |

#### Map/Reduce

Yahoo and Google popularized the Map/Reduce model as they worked to implement their search capabilities. In general, M/R programs follow five basic stages:

1. Input preparation and assignment to mappers
2. Map some set of keys and values to new keys and values: Map(k1,v1) -> list(k2,v2)
3. Shuffle data to each reducer and each reducer sorts it’s input – each reducer is assigned some set of keys (k2).
4. The reduce runs on some list(v2) associated with each key and produces output: Reduce(k2, list(v2)) -> list(v3)
5. Final output the lists(v3) are from east reducer are combined and sorted by k2.

Of note that while there is a single output, nothing in the model prohibits multiple input data sets and that it is extremely common for complex analytics to be built as workflows of multiple M/R jobs. While this programming model is best suited to aggregation (sum, average, group-by) type analytics a analytics wide variety of analytic algorithms have been implemented with-in the framework. M/R does not generally do well with applications/algorithms that need to directly update the underlying data since to update say the values for a single key would require the entire data set be read, output, then moved/copied over the original data set. Because the mappers and reducers are stateless in nature applications that require iterative computation on parts of the data or repeated access to parts of the data set do not tend to scale/perform well under M/R.

The usability of M/R for big data applications due to it’s shared nothing approach has made it popular enough that a number of large data storage solutions (mostly those of the NOSQL variety) provide implementations with-in their architecture. One major criticism of M/R early on was that the interfaces to most implementations were too low a level (written in Java or Javascript), however many of the more popular implementations now support high level procedural and declarative language interfaces and even visual programming environments are beginning to appear.

#### Bulk Synchronous Parallel(BSP)

The BSP programming model was originally developed by Leslie Valiant of Harvard and published in the Communications of the ACM in 1990. BSP combines parallel processing with the ability of processors to send messages to other processor and explicit synchronization of the steps. A BSP algorithm is composed of what are termed supersteps which are comprised of three distinct elements:

1. Bulk Parallel Computation – Each processor performs the calculation/analysis on its local chunk of data.
2. Message Passing – As each processor performs its calculations it may generate messages to other processors. These messages are frequently updates to values associated with the local data of other processors but may also result in the creation of additional data.
3. Syncronization – Once a processor has completed processing it’s local data it pauses until all other processors have also completed their processing.

This cycle can be terminated by all the processors “voting to stop” which will generally happen when a processor has generated no messages to other processors (e.g. no updates). All processors voting to stop in turn means that there are no new updates to any processors data and the computation is complete. Alternatively, the cycle may be terminated after a fixed number of supersteps have been completed (for example after a certain number of iterations of a monti-carlo simulation).

The advantage of BSP over Map/Reduce is that processing can actually create updates to the data being processed. It is this distinction that has made BSP popular for graph processing and simulations where computations on one node/element of data directly affects values or connections with other nodes/elements. The disadvantage of BSP is the high cost of the synchronization barrier between supersteps. Should the distribution of data or processing between processors become high unbalanced then some processors may become overloaded while others remain idle.

Numerous extension and enhancements to the basic BSP model have been developed and implemented over the years andmany are designed to address the balancing and cost of synchronization problems.

## Feature 3: Resource Manager Framework

We need to build off of the value statement described in the table above in the “Features Summary” section to develop a case for the feature. Why it is important in the Big Data Ecosystem? What readiness level do we need it to get to and/or what should this feature be like in 5 years to meet the collective needs of Government / Academia / Industry? What needs to happen and how do we get there?

Author: Dave

As Big Data systems have evolved and become more complex and as businesses work to best leverage limited computation and storage resources to address a broader range of applications and business problems the requirement to effectively manage those resources has grown significantly. While tools for resource management and “elastic computing” have expanded and matured in response to the needs of Cloud Providers and virtualization technologies, Big Data brings some unique requirements to the mix. The NIST Cloud Computing reference architecture and frameworks have solid discussions of resource management within the cloud computing realm. Big Data frameworks on the other hand tend to fall more into a distributed computing paradigm which presents additional challenges.

In discussing resource management in the context of Big Data the Volume and Velocity attributes drive the requirements. Elastic computing approaches (e.g. spawning another instance of some service) is the most common response to dealing with expansion in volume or velocity of data coming into the system. Where, Big Data becomes different and more complex is in the allocation of computing resources to different storage or processing frameworks optimized for specific applications and data structures. Critical, is the fact that because the data is Big (Volume) it is not generally feasible to move data to the processing frameworks. In addition, while pretty much all Big Data processing frameworks can be run in virtualized environments most are designed to run on bare metal commodity hardware to provide efficient I/O for the volume of the data. The two resources which tend to be most critical to manage in Big Data situations are CPU and memory. While shortages or over allocation of either will have significant impacts on system performance, improper or inefficient memory management is frequently catastrophic. The primary function of these managers tend to be allocation of framework components (master nodes, processing nodes, job slots, etc.) to physical machines with many seeking to optimize data locality. Because, data locality is frequently extremely important in the performance of Big Data systems unlike cloud/virtualization resource managers it is often not feasible to relocate framework components to different machines without also moving the associated data.

In response to this need two distinct approaches to resource management have been evolving in respect to Big Data frameworks. The first is Intra-framework resource management. Under this approach the framework itself manages allocation of resources between it’s various components. This allocation is typically driven by the framework’s workload and often seek to “turn off” unneeded resources to either minimize the overall demands of the framework on the system or in to minimize the operating cost of the system by reducing energy use. Under this approach, applications can seek to schedule and request resources that much like main frame operating systems of the past are managed through scheduling queues and job classes.

The second approach is Inter-framework resource management which is designed to address the needs of many Big Data systems to support multiple storage and processing frameworks that can address and be optimized for a wide range of applications. Under this approach, the resource management framework actually runs as a service supporting and managing resource requests from frameworks, monitoring framework resource usage, and in some cases managing application queues. In many ways this approach is like the resource management layers common in cloud/virtualization environments and there are efforts under way to create hybrid resource management frameworks that handle both physical and virtual resources.

Frequently ignored but critical to the performance of distributed systems and frameworks and especially critical to Big Data implementations is the efficient and effective management of networking resources. Recently there have been significant contributions in network resource management through what is being termed as “Software Defined Networking”. Much like virtualization frameworks manage shared pools CPU/Memory/Disk, software defined or virtual networks manage pools of physical network resources. Rather than the traditional approaches of dedicated physical network links for data, management, I/O, and control with software defined networks the multiple physical resources to include links and actual switching fabric are pooled and allocated as required to specific functions and sometimes to specific application. This allocation, can consist of raw bandwidth, quality of service (QOS) priority, and even actual routes of data.

Taking these concepts even further and combining them is the emerging technologies built around what is being termed “Software Defined Data Centers”. This expansion, on elastic and cloud computing goes beyond the management of fixed pools of physical resources as virtual resources to include the automated deployment and provisioning of features and capabilities onto physical resources. For example, automated deployment tools that interface with virtualization or other framework APIs can be used to automatically stand up entire clusters or to add additional physical resources to physical or virtual clusters.

## Feature 4: Infrastructure Architecture

We need to build off of the value statement described in the table above in the “Features Summary” section to develop a case for the feature. Why it is important in the Big Data Ecosystem? What readiness level do we need it to get to and/or what should this feature be like in 5 years to meet the collective needs of Government / Academia / Industry? What needs to happen and how do we get there?

Author: Dave

Critical to the success of Big Data implementations is that the infrastructure is sized to adequately support the application requirements. Infrastructure Architecture needs range from basic power and cooling to external bandwidth connectivity. A key evolution that has been driven by Big Data is the increase in server density (more CPU/Memory/Disk per rack unit). However, with this increased density infrastructure specifically power and cooling may not be distributed within the data center to allow for sufficient power to each rack or adequate air flow to remove excess heat.

The other aspect of infrastructure architecture that must be addressed for Big Data is connectivity. While, some Big Data implementations may solely deal with data that is already resident in the data center and not need to leave the confines of the local network others need to plan and account for the movement of Big Data either into or out of the data center where the Big Data system is located. Specifically, location of Big Data systems with these types of transfer requirements may need to be chosen based on availability of external network connectivity (bandwidth) but also given the limitations of TCP where there is low latency (as measured by packet Round Trip Time) with the primary senders or receivers of Big Data. To address the limitations of TCP architects for Big Data systems may need to consider some of the advanced non-TCP based communications protocols available that are specifically designed to transfer large files such as video and imagery.

Another, aspect that relates to the velocity of Big Data that must be consider in architecting external connectivity is the overall availability of the external links. A given connectivity link may be able to easily handle the velocity of data while operating correctly. However, should the quality of service on the link degrade or the link fail completely data may be lost or simply back up to the point that it can never recover. There are even known use cases where the contingency planning for network outages involves transferring data to physical media and transporting it physically to the destination. But even this approach is limited by the time it may require to transfer the data to external media for transport.

## Feature 5: Information Framework

We need to build off of the value statement described in the table above in the “Features Summary” section to develop a case for the feature. Why it is important in the Big Data Ecosystem? What needs to happen and how do we get there?

What readiness level do we need it to get to and/or what should this feature be like in 5 years to meet the collective needs of:

Government /

Academia /

Industry?

Author: Pw

We are currently going thorough an organic spurt of technical growth based upon two strategies for the mining of information data as well as gold. This spurt is gradually gaining momentum and coming together as; Big Data and Cloud Eco-systems and eventually evolving into a Big Data/Cloud Eco-system.

This will not be a neat linear growth curve, ever upward. Rather, there will be financial adjustments, marketing hype resulting in market over-valuations followed by market corrections as well as both fraud and opportunistic failures. Yet these two technical strategies for handling data will continue to grow, and grow organically inspite of the associated hype and/or overhype.

Within the next five years there be a coming together within this expansion of technological innovation, supporting mechanisms including standards and regulatory organizations, for each of the constructs surrounding integrity, availability, confidentiality, governance, risk and compliance, each of which will finally influence the building out of a truly valued technological explosion of innovation based upon Big Data.

Such a growth in innovative applications will be driven by the implementation of Big Data across all industries, broadening the footprint while at the same time reducing the cost of entry into these new economies.

These new scalability factors will transform industries and sectors across all economies including; communications, energy, life sciences, information, data, science, finance, manufacturing, engineering, health, politics, innovation and knowledge. [Content Goes Here]

## Feature 6: Standards Integration Framework

Author: Keith

Integration with appropriate standards (de jure, consortia, reference implementation, open source implementation, etc.) can assist both in cross-product integration and cross-product knowledge. Identifying which standards efforts address architectural requirements and which requirements are not currently being addressed provides input for future standards efforts.

Standards efforts support aspects of the following characteristics:

* Interoperability – the ability for analytical tools from one source (open source project or commercial vendor) to access a data provider implemented using database tools from another source (open source project or commercial vendor).
* Portability – this attribute depends on ones point of view.
  + Application portability – the ability to move analytical tools from one environment to another without changes – will be necessary to avoid moving large amounts of data
  + Data portability – the ability to transparently migrate stored data from one data provider implemented using one set of database tools to another set of database tools – is unlikely to happen. Migrating data using some set of unload and load tools or a standard interface is needed, but will time and resource requirements are proportionate to the amount of data.
* Reusability – need some more words
* Extensibility – need some more words

The benefits of standardization are mostly found in the interfaces between technologies. In the Reference Architecture, the Data Service Abstraction layer can be expanded to include:

* Data Provider Registry and Location Services
  + Allow data providers to register their existence
  + Allow data consumers to identify and locate useful data providers
* Data Provider Interfaces – A common interface is needed to allow data consumers to communicate with data providers
* Data Stores – Common language needed to interface with data stores

Standardization of Capabilities Service Abstraction layer elements benefits the creation, management, and deployment of very large data providers. This area includes:

* Support for easily creating and deploying physical and virtual machines
* Support for easily creating and deploying very large data stores

Requirements to support security and privacy cross all aspects of the Reference Architecture. Integrating existing security and privacy standards will require a great deal of coordination and cooperation.

There are currently a number of standards efforts that address pieces of the elements in the reference architecture. The existing elements and their gaps are documented in Section 7 Big Data Related Multi-stakeholder Collaborative Initiatives.

## Feature 7: Application Framework

We need to build off of the value statement described in the table above in the “Features Summary” section to develop a case for the feature. Why it is important in the Big Data Ecosystem? What readiness level do we need it to get to and/or what should this feature be like in 5 years to meet the collective needs of Government / Academia / Industry? What needs to happen and how do we get there?

Author: Carl

## Feature 8: Business Operations

Author: Bruno

Big Data is more than just technology, but also a cultural and organizational transformation. Business Operations need to be able to strategize, deploy, adopt, and, operate Big Data solutions. As mentioned in the Readiness section earlier, Organizational Readiness is critical if Big Data is to transcend all aspect of the business. Strategy, governance, portfolio management, and, the organization per se require coordination if Big Data is to be operationalized. Once a business [unit] has decided on pursuing a Big Data initiative, they will have access their organizational readiness and level of intended adoption. Operationalizing this for mid and long term sustainment will require

Putting Big Data into action – operationalizing – is more than just a technology upgrade. The NIST Big Data Reference Architecture shows a vertical ‘Orchestrator’ that ensures the

Author’s Note: Provide Methodology, here : Validate – Prioritize – Plan – Prototype - Pivot

Decide the Why: “What are the reasons for us to pursue Big Data?”

* Optimiztion
* Agility
* Innovate
* Compliance
* Green

Authors Note: Mention how to Prioritize for Operation’s sake: Cost versus Return -- Decide

Decide the Return: “How do we justify this investment?”

* CAPEX/OPEX savings
* Ability to Respond
* New Businesses [Revenue]
* Improved Compliance Metrics
* Carbon footprint]

Decide the Blockers: “What has – or will – stop us?”

* Time
* Budget
* Resources
* Culture
* Skills Gap
* Bureaucracy

Assess Internal Readiness: “Where are we now?” “How do we qualify ourselves?”

* Levels 1-6

Confirm Use Cases: “Who is this for – do they want it – will they use it?”

* Choose from at least 44 Use Case examples

Assess Intended Adoption: “What is the consumption [rate] for our Big Data initiative – per Use Case?”

* (Levels 1-6)

Assess Costs: “What’s our Bottom’s Up Costs?”

* People
* Hardware
* Software
* Licenses
* Data Center(s)
* Backup/DR
* Storage
* Network
* Vendors
* Data Cleansing
* Support
* Deploy
* User Training
* UAT
* User Rollout

Develop Internal Bill-Of-Material (BoM): “What is the break-down if we had to hand over a BoM”

* Bill-of-IT
* Other Cost Centers
* Business Units – Marketing/Finance/Sales/HR)

Which Data?: “How do we decide on which Data to evaluate?”

* CAP Theorem-per Distributed Systems
* Emerging versus Existing data types
* Data Cleansing
* Transactional-Real Time-Analytical

Per the RA: “Transpose the RA against our operational needs.” Or “Trace the Use Case requirement though the RA – how does this affect our operations?”

* Data Services versus Operations
* System Services versus the Operations
* Capabilities versus the Operations
* Usage Services versus the Operations

Define Risks “Which Methodology do/have we use?”

* CMMI
* ITIL
* CoBIT
* Lean
* Agile
* PMI
* DevOps
* Blue Ocean Strategy
* Six Sigma

Assess Operational Sustainment: “If we decide to go forward, how do we sustain our success?”

* Services Catalogue
* SLA’s
* Change Control
* Chargeback & Metering Model
* Financial & Portfolio Management
* Licensing
* Support
* Backup
* Recovery
* Fault Tolerance
* Risk Management
* Governance
* Vendor Account Management
* IT training
* End User training

# Big Data Related Multi-stakeholder Collaborative Initiatives

Author: Keith

Big Data has generated interest in a wide variety of organizations, including the de jure standards process, industry consortiums, and open source organizations. Each of these organizations operates differently and focuses on different aspects, but with a common thread that they are multi-stakeholder collaborative initiatives.

Integration with appropriate collaborative initiatives can assist both in cross-product integration and cross-product knowledge. Identifying which collaborative initiative efforts address architectural requirements and which requirements are not currently being addressed provides input for future multi-stakeholder collaborative initiative efforts.

“Collaborative initiatives” include:

* Subcommittees and working groups of Accredited Standards Development Organizations (the de jure standards process)
* Industry Consortia
* Reference implementations
* Open Source implementations

Open source implementations (such as those from the Apache Software Foundation) are providing useful new technology that is being used either directly or as the basis for commercially supported products. These open source implementations are not just individual products. One needs to integrate an eco-system or products to accomplish ones goals. Because of the ecosystem complexity, and because of the difficulty of fairly and exhaustively reviewing open source implementations, such implementations are not included in this section.

The following sections describe work currently completed, in planning and in progress in the organizations:

* INCITS and ISO – de jure standards process
* IEEE – de jure standards process
* W3C – Industry consortium
* OGC – Industry consortium

Organizations working in this area that are not included in this section are omitted through oversight. It is anticipated that as this document is more widely distributed, standards efforts addressing additional segments of the Big Data mosaic will be identified.

This work is mapped onto the Big Data Reference Architecture abstraction layers:

* Data Service Abstraction
* Capability Service Abstraction
* Usage Service Abstraction
* Usage Service Abstraction

In all cases, the standards identified cover aspects of the Big Data requirements identified in this document. They contain starting points for additional work but are not complete answers today. The value of identifying these incomplete efforts in this document is to avoid the complete recreation of work that has already been accomplished.

### Characteristics supported by standards

Standards efforts support aspects of the following characteristics:

* Interoperability
* Portability
* Reusability
* Extensibility

Evaluating the effectiveness of particular standards efforts with respect to these characteristics is complicated because of the complexity of the Big Data Architecture and the current patchwork nature of relevant standards.

### Information and Communications Technologies (IT) Standards Life Cycle

Different collaborative initiatives have different processes and different end goals, so the life cycle varies. The following is a broad generalization of the steps in a multi-stakeholder collaborative initiative life cycle:

* No standard
* Under development
* Approved
* Reference implementation
* Testing and certification
* Products/services
* Market acceptance
* Sunset

## Data Service Abstraction

The data service abstraction layer needs to support the ability to:

* Identify and locate data providers with relevant information
* Access the data source through some sort of query and/or analysis tool.

The following sections describe the standards related to:

* Data Provider Registry and Location Services
* Data Provider Interfaces
* Data Stores

### Data Provider Registry and Location service

A Data Provider Registry and Location service allows a data provider to:

* Create metadata describing the data source(s), usage policies/access rights, and other relevant attributes
* Publish the availability of the information and the means to access it
* Make the data accessible by other RA components using suitable programmable interface.

While ISO/IEC JTC1 SC32 WG2 has a variety of standards in the areas of registering metadata, the pieces do not completely specify the support needed to create a registry of the content and location of data providers. The related standards are:

| **Data Provider Interface** | **Standards Group** | **Related Standards** |
| --- | --- | --- |
| Metadata | INCITS DM32.8 & ISO/IEC JTC1 SC32 WG2 | The ISO/IEC 11179 series of standards provides specifications for the structure of a metadata registry and the procedures for the operation of such a registry. These standards address the semantics of data (both terminological and computational), the representation of data, and the registration of the descriptions of that data. It is through these descriptions that an accurate understanding of the semantics and a useful depiction of the data are found. These standards promote:   * Standard description of data * Common understanding of data across organizational elements and between organizations * Re-use and standardization of data over time, space, and applications * Harmonization and standardization of data within an organization and across organizations * Management of the components of data * Re-use of the components of data |
|  | INCITS DM32.8 & ISO/IEC JTC1 SC32 WG2 | The ISO/IEC 19763 series of standards provides specifications for a metamodel framework for interoperability. In this context interoperability should be interpreted in its broadest sense: the capability to communicate, execute programs, or transfer data among various functional units in a manner that requires the user to have little or no knowledge of the unique characteristics of those units (ISO/IEC 2382-1:1993). ISO/IEC 19763 will eventually cover:   * A core model to provide common facilities * A basic mapping model to allow for the common semantics of two models to be registered * A metamodel for the registration of ontologies * A metamodel for the registration of information models * A metamodel for the registration of process models * A metamodel for the registration of models of services, principally web services * A metamodel for the registration of roles and goals associated with processes and services * A metamodel for the registration of form designs |

The missing pieces are:

* A data model for the registry
* Mechanisms (such as a call interface or query language) to register a data source
* Mechanisms (such as a call interface or query language) to retrieve information about a data source
* Mechanisms to broker the connection to the data source
* Integration of security and privacy requirements

### Data Provider Interfaces

Once a data consumer has identified a data source with the require characteristics, a communication link must be established. This communications link needs to be able to transfer:

* Queries and Requests to the data provider where the request could include some sort of analytical function.
* Result sets to the requester where the result set could be as simple as a series of documents or as complex as the output training from a machine learning engine.

The following data source interfaces are frequently used:

| **Data Source Interface** | **Standards Group** | **Related Standards** |
| --- | --- | --- |
| SQL/CLI | INCITS DM32.2 & ISO/IEC JTC1 SC32 WG3 | ISO/IEC 9075-9:2008 Information technology – Database languages – SQL – Part 9: Management of External Data (SQL/MED) supports mapping external files underneath an SQL interface. |
| JDBC™ | Java Community ProcessSM | JDBC™ 4.0 API Specification |

These interfaces are primarily designed for row-style data. They need to be enhanced, or new interface languages constructed, to support more complex datatypes such as images, video, trained machine learning engines, etc.

### Data Sources

The Data Service Abstraction layer needs to support a variety of data retrieval mechanisms including (but not limited to):

* Flat Files with known structure
* Flat files with free text
* XML documents
* SQL Databases
* Audio, Picture, Multimedia, and Hypermedia
* Spatial Data
* Sensor network data
* Streaming data – Video
* Steaming data – Textual
* NoSQL Data Stores

| **Data Source** | **Standards Group** | **Related Standards** |
| --- | --- | --- |
| Flat Files with known structure | INCITS DM32.2 & ISO/IEC JTC1 SC32 WG3 | ISO/IEC 9075-9:2008 Information technology — Database languages — SQL — Part 9: Management of External Data (SQL/MED) supports mapping external files underneath an SQL interface. |
| Text | INCITS DM32.2 & ISO/IEC JTC1 SC32 WG4 | ISO/IEC 13249-2 SQL/MM Part 2: Full Text provides full information retrieval capabilities and complement SQL and SQL/XML. SQL/XML provides facilities to manage XML structured data while MM Part 2 provides contents based retrieval. |
| XML documents | W3C XQuery Working Group | XQuery 3.0: An XML Query Language — uses the structure of XML to express queries across all these kinds of data, whether physically stored in XML or viewed as XML via middleware. |
| INCITS DM32.2 & ISO/IEC JTC1 SC32 WG3 | ISO/IEC 9075-14:2011 Information technology — Database languages — SQL — Part 14: XML-Related Specifications (SQL/XML) supports the storage and retrieval of XML documents in SQL databases |
| SQL Databases | INCITS DM32.2 & ISO/IEC JTC1 SC32 WG3 | The SQL Database Language is defined by the ISO/IEC 9075 family of standards:   * ISO/IEC 9075-1:2011 Information technology — Database languages — SQL — Part 1: Framework (SQL/Framework) * ISO/IEC 9075-2:2011 Information technology — Database languages — SQL — Part 2: Foundation (SQL/Foundation) * ISO/IEC 9075-3:2008 Information technology — Database languages — SQL — Part 3: Call-Level Interface (SQL/CLI) * ISO/IEC 9075-4:2011 Information technology — Database languages — SQL — Part 4: Persistent Stored Modules (SQL/PSM) * ISO/IEC 9075-9:2008 Information technology — Database languages — SQL — Part 9: Management of External Data (SQL/MED) * ISO/IEC 9075-10:2008 Information technology — Database languages — SQL — Part 10: Object Language Bindings (SQL/OLB) * ISO/IEC 9075-11:2011 Information technology — Database languages — SQL — Part 11: Information and Definition Schemas (SQL/Schemata) * ISO/IEC 9075-13:2008 Information technology — Database languages — SQL — Part 13: SQL Routines and Types Using the Java TM Programming Language (SQL/JRT) * ISO/IEC 9075-14:2011 Information technology — Database languages — SQL — Part 14: XML-Related Specifications (SQL/XML) |
| Audio, Picture, Multimedia, and Hypermedia | INCITS L3 & ISO/IEC JTC 1 SC29 | ISO/IEC 9281:1990 Information technology — Picture coding methods  ISO/IEC 10918:1994 Information technology — Digital compression and coding of continuous-tone still images  ISO/IEC 11172:1993 Information technology — Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s  ISO/IEC 13818:2013 Information technology — Generic coding of moving pictures and associated audio information  ISO/IEC 14496:2010 Information technology — Coding of audio-visual objects  ISO/IEC 15444:2011 Information technology — JPEG 2000 image coding system  ISO/IEC 21000:2003 Information technology — Multimedia framework (MPEG-21) |
| INCITS DM32.2 & ISO/IEC JTC1 SC32 WG4 | ISO/IEC 13249-5 Part 5: Still Image provides basic functionalities for Image data management within SQL databases. |
| Spatial Data | INCITS L1 - Geographical Information Systems & ISO/TC 211 – Geographic information/Geomatics | ISO 6709:2008 Standard representation of geographic point location by coordinates  The ISO 191nn suite of geospatial standards |
| Open Geospatial Consortium | The OGC suite of geospatial standards and Abstract Specifications |
| INCITS DM32.2 & ISO/IEC JTC1 SC32 WG4 | ISO/IEC 13249-3 Part 3: Spatial provides support for the functionalities required by geospatial applications. This work is carefully coordinated with ISO TC 211 and the Open Geospatial Consortium |
| Sensor network data | IEEE | ISO IEEE 21451 series of sensor standards and standards projects e.g. ISO IEEE 21451-2 Information technology — Smart transducer interface for sensors and actuators — Part 2: Transducer to microprocessor communication protocols and Transducer Electronic Data Sheet (TEDS) formats  ISO IEEE 21451-7 Standard for Information Technology - Smart Transducer Interface for Sensors and Actuators - Transducers to Radio Frequency Identification (RFID) Systems Communication Protocols and Transducer Electronic Data Sheet Formats |
| Streaming data – Video | IEEE | IEEE 2200-2012 Standard Protocol for Stream Management in Media Client Devices |
| Streaming Data — Textual | INCITS DM32.2 & ISO/IEC JTC1 SC32 WG4 | ISO/IEC 9075-2:201x Information technology —Database languages — SQL — Part 2: Foundation (SQL/Foundation) supports queries using regular expressions across series of rows, but does not (yet) support operating on data streams |
| NoSQL Data Stores | Various | A large number of “open source” products exist but currently have no common interface language |

Note that while standards such as ISO/IEC 15444:2011 “Information technology — JPEG 2000 image coding system”, provide standard encodings for images, additional information is needed so that the images can be appropriately transformed for analysis. The transformation needed for images of fingerprints is different from those of photos of text, people, aerial photos, astronomical photos, etc. Many of these transforms already exist in various environments. Additional work is needed to identify the transforms (as well as related standards) and describe how the appropriate transforms can be represented in the Data Provider Registry and Location Service.

## Usage Service Abstraction

It is possible that a data consumer in the Usage Service Abstraction could make some transformation available as a data provider in the Data Service Abstraction. Therefore, the standards discussed in the Data Service Abstraction section are potentially relevant here.

## Capability Service Abstraction

Need some introductory words…

### Security and Privacy Management

Security and Privacy topics have been the subject of multiple standards efforts as offered by the following table.

| **Requirement** | **Standards Group** | **Related Standards** |
| --- | --- | --- |
| Infrastructure Security | INCITS CS1 & ISO/IEC JTC 1/SC 27 | ISO/IEC 15408-2009 Information technology — Security techniques — Evaluation criteria for IT security  ISO/IEC 27010:2012 Information technology — Security techniques — Information security management for inter-sector and inter-organizational communications  ISO/IEC 27033-1:2009 Information technology — Security techniques — Network security  ISO/IEC TR 14516:2002 Information technology — Security techniques — Guidelines for the use and management of Trusted Third Party services |
| Data Privacy |  | ISO/IEC 29100:2011 Information technology — Security techniques — Privacy framework |
| Data Management, Securing data stores, Key management, and ownership of data |  | ISO/IEC 9798:2010 Information technology — Security techniques — Entity authentication  ISO/IEC 11770:2010 Information technology — Security techniques — Key management |
| Integrity and Reactive Security |  | ISO/IEC 27035:2011 Information technology — Security techniques — Information security incident management  ISO/IEC 27037:2012 Information technology — Security techniques — Guidelines for identification, collection, acquisition and preservation of digital evidence |

Table Name Here:------>

Each of these standard efforts by various organizations must coordinate and calibrate their efforts into a complementary series of standards that do not muddy the compliance waters resulting in an intuitive sequence of guidelines that are not open to multiple interpretations. Each organization needs to have their standards reviewed and mechanisms identified to apply these complimentary capabilities to the data service abstraction layer.

How to transform these standards into a beneficial tool (one among many) in the areas of security and privacy is demonstrated by the following Audit/Digital Forensics RACI Matrix:

Big Data Supporting: Security & Privacy (Audit/Digital Forensics) Baseline Management Reference Checklist for RACI (Responsible, Accountable, Consulted & Informed)

Big Data Security & Privacy (Audit/Digital Forensics) RACI Checklist (Responsible, Accountable, Consulted & Informed)

| **No.** | **Requirement** | **RACI Roles (Responsible, Accountable, Consulted & Informed) - Job Title** | **Governance, Risk & Compliance and/or Confidentiality, Integrity & Availability, (GRC/CIA) Requirements Solution:**  **Addressed/Resolved (Yes/No or N/A)** | **Why/Why Not/(N/A)** |
| --- | --- | --- | --- | --- |
| **What Is The Business Purpose For Moving From A Current Legacy Eco-system To A New Cloud/Big Data Eco-system?** | | | | |
|  | Is there a business reason for moving from physical to virtual and/or to Cloud/Big Data Eco-system? |  |  |  |
|  | Does the virtual IT System impact a Regulatory GRC/CIA compliance requirement? |  |  |  |
|  | Are business goals met by utilizing Cloud/Big Data/Virtualization Eco-systems? |  |  |  |
|  | Conduct and evaluate a cost benefit’s analysis, via CAPEX and/or OPEX pre-implementation vs post-implementation expectations. |  |  |  |
|  | Conduct and evaluate a six (6) month follow along cost benefit’s analysis, via CAPEX, OPEX post-implementation. |  |  |  |
| **Cloud/Big Data Eco-system Enterprise Wide - Risk Analysis / Risk Assessment** | | | | |
|  | Is there sufficient expertise to support the new environment? |  |  |  |
|  | Has there been sufficient training of the responsible teams for working and maintaining the new eco-system? |  |  |  |
|  | Are enterprise operational procedures regularly updated? |  |  |  |
|  | Are there any SPFs (Single Points of Failure). If so, have they been identified and corrected? |  |  |  |
|  | Are the security zones separated and/or combined? |  |  |  |
|  | How are the IT resources separated and aggregated throughout the entire eco-system? |  |  |  |
|  | How is the enterprise eco-system security managed? |  |  |  |
|  | Is there administrative access to the host machine? |  |  |  |
|  | Does the management console have tight access controls locked down to specific users, specific partitions or machines and is each identified and up-to-date? |  |  |  |
| **Architecture & Infrastructure Controls** | | | | |
|  | Are the partitions on different OS’s (Operating Systems)? |  |  |  |
|  | Are the partitions on a single server and/or across multiple servers? |  |  |  |
|  | Is there a network map indicating what partitions exist, for each environment and on which machines? |  |  |  |
|  | Are there controls over each partition, similar to those expected for a server? |  |  |  |
|  | Are there controls for specific users that limit access and/or read/write capabilities? |  |  |  |
|  | Does a standard naming convention exist for; servers, partitions, libraries and/or folder and file names? |  |  |  |
|  | What controls are in-place for deploying multiple copies of software? |  |  |  |
| **Network Maps (Current Legacy And/Or Future Environments)** | | | | |
|  | Where are the following types of systems located?   * System Development * Systems Testing * Production Systems * Business Unit Servers   Are the environments separated by data sensitivity? |  |  |  |
| **Evaluation of Policies, Procedures and Documentation** | | | | |
|  | Evaluate the standards prepared by the organization for system and security administration of the virtual IT systems. |  |  |  |
|  | Evaluate the procedure of creating, deploying, managing and making changes to the Cloud/Big Data Eco-system. |  |  |  |
|  | Evaluate the lock-down and hardening policies. |  |  |  |
|  | Evaluate the completeness, currency and accuracy of the Cloud/Big Data Eco-system documentation. |  |  |  |
| **Management Evaluation Controls** | | | | |
|  | Does the documentation for Change Control (CC) and Change Management (CM) refer to the correct partition(s) on the correct server(s)? |  |  |  |
|  | Has the testing and evaluation of system(s) backup capabilities been conducted and verified? |  |  |  |
|  | Has enterprise wide Disaster Recovery (DR) procedures been tested, evaluated and verified? |  |  |  |
|  | Are all software licenses up to date, catalogued and tracked? |  |  |  |
|  | Have contracts and/or vendor licenses been evaluated? |  |  |  |
|  | How frequently are formal security audits conducted, as well as informal security audits and security audit walkthroughs? |  |  |  |
|  | Evaluate (via documented testing and/or walkthroughs) of all 3rd party solutions being used to enhance the security of the virtual environment and their compatibility with the virtual IT systems. |  |  |  |
|  | Evaluate the control standards prepared by the organization for the virtual IT system(s). |  |  |  |
|  | Are there appropriate and verifiable resource usage and cost allocations among all applications, across a shared infrastructure? |  |  |  |
|  | Is there any indication of; image sprawl/virtual sprawl due to system mismanagement? |  |  |  |
|  | Does the system have orphaned images? |  |  |  |
|  | Is there a provision for hypervisor security contained within the SLA (Service Level Agreement)? |  |  |  |
|  | Evaluate, test and document the business continuity and capacity management strategies for the virtual IT systems. |  |  |  |
|  | Evaluate, test and document the existing configuration management infrastructure to determine the scalability and efficiency of the virtual IT system. |  |  |  |
|  | Evaluate, test and document the patch management procedures of all service providers. |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

### System Management

| **Requirement** | **Standards Group** | **Related Standards** |
| --- | --- | --- |
| System Management and Scalable Infrastructure | Distributed Management Task Force | ISO/IEC 13187: 2011 “Information Technology – Server Management Command Line Protocol (SM CLP) Specification”  ISO/IEC 17203 2011 “Open Virtualization Format” |

## Standards Summary

While there are many standards efforts that touch pieces of the elements identified in the Big Data Reference Architecture, significant gaps exist. This section identifies many standards efforts, the pieces they do support, and areas needed to fully support the Big Data Reference Architecture described in this document.

# Big Data Strategies

## Strategy of Adoption

Author: Keith

### Identify and include stakeholders

Who are the stakeholders in this project?

* Marketing?
* Operations?
* Information Technology?
* Others?

It is important to include the critical stakeholders.

### Identify potential road blocks

In any project, there are a variety of potential people and situations that can derail progress.

**People**

For a Big Data project the people can be:

* Systems/network/security people who are too overwhelmed with day-to-day requirements to spend time working on a new project
* People who are wedded to the status quo
* ?

Where ever possible, co-op these people into the project.

**Data**

An additional potential issue is incomplete and/or poor quality data. In general, data that is write-only (Write Once, Read Never – WORN) will have data quality issues. These data risks must be identified as early as possible.

As data issues are identified, their potential impact must be reviewed to assess the impact. The possible impacts are:

* Issue is annoying but will have only minor impact on the results
* Issue will affect the validity of the results
* Issue is a complete show stopper

Whenever possible, these types of impediments should be documented and bypassed.

### Define Achievable Goals

Much of the current hype around Big Data takes the form of “By accumulating all of this data, we will be able to understand the universe, end hunger, and achieve world peace.” Most Big Data projects will achieve much more limited results, so it is critical to set realistic expectations up front. These expectations might look like:

* We expect to identify types of items that customers frequently purchase together so that we can organize the shelves in a way that increases year-over-year per-store sales.
* We are looking for patterns and correlations that will help identify potential maintenance issues before the problems actually occur, thereby reducing production down time.
* We are looking for places where we can slow down storm runoff into the sewage system in order to keep peak volumes within the sewage processing plant capacity and prevent untreated sewage discharges into the river.
* We don’t know yet what we can do with the data, but if we don’t accumulate it now, we will not have the opportunity to investigate.

It usually makes sense to include both a realistic estimate of the possibility of success as well as the potential benefit if the project does actually achieve something useful. This could take the form of:

* From what we know today, there is about a 10% chance that we will achieve something useful, but if we can identify changes that improve efficiency by 1%, it could save the company ten million dollars a year.
* ???

### Define “Finished” and “Success” at the beginning of the project

Two decades ago, I worked with a project manager who claimed there are two critical criteria that should be defined at the start of a project:

* How do you know when the project is done?
* How do you know if the project is successful?

This can also be categorized as correctly setting expectations at the beginning of the project but it also provides dispassionate criteria for evaluating a project’s status.

## Strategy of Implementation

Author: Bruno

The NIST Roadmap Working Group cannot provide the answer for every organization on how to decide on which Big Data solution to use. The Working Group can provide general direction for stakeholders on how to incorporate Roadmap artifacts into decision making. A Big Data Framework has been designed to enable stakeholders to make decisions based on an agnostic approach.

To assist in deciding on how to decide on Big Data for an organization, templates, business conversations, and, dependencies have been outlined by the NIST Working Group.

A set of templates have been created to provide both a technology and methodology agnostic approach in making a decision on Big Data. The following describes the upcoming templates to be provided:

1. Internal workshops: This is a daily agenda format providing an outline for a team to collaborate on Big Data strategizing.
2. Readiness Self-Assessment: This template provides an approach to defining if the organization and its technology is prepared for Big Data.
3. Questionnaire: This template provides example Big Data questions a team should ask themselves.
4. Vendor Management: This template explains how a team can use its findings and incorporate them into an RFI, RFQ, or, RFP.

For the above templates, business conversations will typically drive the Big Data course-of-action. Five types of business conversations are:

1. Optimize: This conversation revolves around how Big Data will improve the efficiency of the business, to include processes, CAPEX and OPEX.
2. Agility: This conversation revolves around Big Data assisting in the ability to pivot to the demands of the market, customers, and, any other dependencies.
3. Innovate: This conversation revolves around Big Data assisting the business to create new ways to operate.
4. Compliance: This conversation revolves around Big Data supporting audit capabilities for industry and government standards as: SOX, HIPAA, SEC, etc.
5. Green: This conversation revolves around Big Data supporting Green initiatives for the business.

The templates and business conversations have dependencies. These dependencies have two groups: 1) Business, and, 2) Technology.

The following are the Business dependencies which feed into the templates and business conversations:

1. Culture
2. Organizational (Structure, Silos, Processes)
3. Governance
4. Fiscal Planning
5. Mergers & Acquisitions

The following are the Technology dependencies which feed into the templates and business conversations:

1. As-Is Architecture
2. IT Roadmap
3. IT Team (Skills and Aptitude)
4. IT Services Catalogue
5. Bill-of-IT
6. Vendor Strategy

Below is an output example of how a template can assist stakeholders in articulating their Big Data needs:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Business Conversation | Value | Big Data  Feature | Readiness Level | Use Cases | Actors |
| Agility:  IT needs to provide Marketing the ability respond real-time to acquiring on-line customers | **Value Statement:**  Lower cost of acquiring new customers by ‘X’ percent by October 1st. | **Roadmap Feature**:  Business Intelligence  (Real-Time BI)  [Reference Architecture capabilities can also be outlined here as well.] | **Technology**:  Reference Implementation  -One or more reference implementations are available  -Reference implementations are usable at scale  **Organization**:  Ad Hoc  -Awareness of Big Data exists  -Some groups are building solutions  -No Big Data plan is being followed | **Use Cases:**  #1, 3,6 | **Management**:  -On-line Marketing Officer  -Revenue Officer  **Analyst**:  -Online Marketing Leads (5)  **Technical**:  Network SME  Datacenter SME  Infrastructure SME  CRM Data SME  Storage SME (TBD)  **End Consumer:**  -Online Customers  -Retail Stores (TBD) |
|  |  |  |  |  |  |

## Resourcing

What are the types of skills, how many types of people, where should an organization start with building their Big Data team? If we can provide the types of skills, and an estimate of how many (obviously dependent on organization size), we could give organizations a starting point for building their Big Data team. Alignment with readiness indicators.

Author: Dan

[Content Goes Here]

# Concerns and Assumptions Statement

Include our master file we are developing to integrate across subgroups.

Author: Carl

[Content Goes Here]