NIST 大数据参考架构（草案）

NIST Big Data Public Working Group
Reference Architecture Subgroup

2015 年 1 月 15 日

National Institute of
Standards and Technology
U.S. Department of Commerce

鸣 谢

特别感谢以下单位对翻译 NIST 研究报告“DRAFT NIST Big Data Interoperability Framework: Volume 6, Reference Architecture”的大力支持（排名不分先后）：

同方股份有限公司

中标软件有限公司

清华大学

山东中创软件商用中间件股份有限公司

浪潮电子信息产业股份有限公司

北京市闪联信息产业协会

华东师范大学

华为技术有限公司

北京世纪互联宽带数据中心有限公司

亚信科技（中国）有限公司

浪潮软件集团有限公司

目录

1 引言

1.1 背景

大数据在激发创新、刺激商业、推动社会进步方面的巨大潜能已得到商业界、学术界和政府部门的共同认可。大数据是用来描述网络化、数字化、传感器化、信息化社会数据泛滥的常用术语。很多以前无法解答的问题，如今通过对大数据资源的有效利用，已经可能解答，这些问题主要包括：

- 如何及早发现流行性疾病并做好预防？
- 如何在高性能新材料合成以前，就通过数据分析来发掘它们？
- 如何扭转网络攻击优于防守的趋势，防范网络安全威胁？

但同时，各方对于大数据相对传统方法究竟有多大优势，也有清醒认识——数据本身的容量、速度和复杂性的增长率，已超出现有科技在数据分析、数据管理、数据传输以及用户领域的发展进度。

如上，对于大数据的内在潜力和目前局限，各方都达成越来越广泛的共识，但对一些重要的、根本的问题缺乏定论，也持续困扰着大数据的潜在用户并阻碍进步。这些问题主要包括以下：

- 大数据解决方案属于什么性质？
- 大数据与传统的数据环境和相关应用有何不同？
- 大数据环境的本质特征是什么？
- 大数据环境如何与既有架构集成？
- 哪些关键科技和标准化方面的挑战亟需解决，从而加速大数据解决方案的部署实施？

在此背景下，2012 年 3 月 29 日，白宫宣布启动大数据研究和开发，该计划的目标包括加快科学和工程领域探索，巩固国家安全，并且通过提高从大量繁琐数据资源中分离提取精华的能力，来改革教学模式。

6 个联邦部门和它们的分支机构承诺至少出资 2 亿美元扶持至少 80 个项目，力图通过显著改善设备和技术，从海量数据中获取、整理并得出结论。同时，该计划鼓励产业界、研究型大学和非盈利组织一起联手联邦政府，最大限度地利用大数据带来的机遇。

受政府和民众双重意愿的推动，美国国家标准与技术研究院（NIST）接受挑战，联合各行业专业人士，确保大数据计划安全有效实施。2013 年 1 月 15 至 17 日，NIST 召开了“云和大数据论坛”，受大会启发，NIST 决定创建一个公共工作组，开发大数据互操作性框架。论坛与会者指出，该框架应当定义并区分大数据技术需要满足的需求，包括互操作性、可移植性、可重用性、可扩展性、数据使用、分析及技术架构。通过这些工作，该框架将促成最为安全有效的大数据方法和技术。

2013 年 6 月 19 日，NIST 大数据公共工作组（NBD-PWG）成立，全国各地工业界、学术界和各级政府纷纷加入。这个公共工作组将致力于打造工业、学术和政府利益共同体，旨在对大数据的定义、分类、安全参考架构、安全隐私需求和技术路线图形成共识，最终形成一个中立于供应商并在技术和基础设施方面独立的框架。基于此框架，大数据利益相关者能够运用最好的分析工具，选择其最适合的计算平台和集群，处理问题或者解决可视化需求，大数据服务商也可从中挖掘增值机会。

《NIST 大数据互操作性框架草案》主要包括 7 卷内容：

- 第 1 卷：定义
- 第 2 卷：分类
- 第 3 卷：案例和总体需求
- 第 4 卷：安全和隐私
- 第 5 卷：架构白皮书调查
- 第 6 卷：参考架构
- 第 7 卷：标准路线图

1.2 参考架构工作小组的范围和目标

参考架构提供了“一个能够指导和约束多元架构、多重解决方案的特定专业领域的权威信息源”。同时，作为解决方案的基础，参考架构也可用于对比和校准。

NBD-PWG 参考架构小组旨在建立大数据技术和开放的参考架构，开放的参考架构则意图实现以下目标：

- 为各种利益相关者提供一种通用语言；
- 鼓励遵守通用标准、规范和模式；
- 提供一致的技术实施方案以解决类似的问题集；
- 在供应商和技术无关的大数据概念模型背景下，加强和提高对大数据成分、处理过程及系统的认识；
- 为美国政府部门、相关机构和其他用户在理解、讨论、分类和比较大数据解决方案的过程中提供技术参考；
- 促进对互操作性、可移植性、可重用性和可扩展性备选标准的分析。

该参考架构旨在促进对大数据业务复杂性的认识。它不代表某个特定大数据系统的系统架构，而是一个用基本参照框架系统来描述、讨论、开发其他系统架构的工具。它主要通过通用的高级概念模型来讨论大数据固有的需求、结构及操作。该模型不依赖于任何供应商提供的产品、服务或参考实例，也没有规定任何抑制创新的解决方案。

NIST 大数据参考架构（NBDRA）不解决以下问题：

- 任何机构的运营系统的详细规格；
- 信息交换或服务的详细规范；
- 基础设施产品集成的推荐规范或标准。

1.3 报告制作

工业界、学术界和政府部门已经探索和研制出各种大数据架构方案。NBDRA 方案的开发主要包括以下五个步骤：

1. 宣布 NBD-PWG 参考架构小组向公众开放，以吸引政府部门、工业界、学术界各方领域专家参与进来。
2. 公开收集大数据架构和资料，使之能够代表各种利益群体、不同数据类型及案例。大多案例出自“案例和需求”工作小组（参见 http://bigdatawg.nist.gov/usecases.php）。
3. 审查分析大数据资料，以便更好地理解大数据现有概念、用法、目标、目的、特征和关键原理，并运用 NIST 大数据分类模型（见《NIST 大数据互操作性框架：第 2 卷，分类法》）记录这些调查结果。
4. 基于大数据资料分析和其他 NBD-PWG 小组的结论，创建开放式参考构架。

5. 形成 NBD-PWG 参考架构小组的调查结果及最终成果文档，制作这份报告。

1.4 报告结构

本报告的组织结构大体遵循 NBD-PWG 开发 NBDRA 的过程，文档其余部分组织如下：

- 第 2 章包含与 NBDRA 设计相关的高层次需求并探讨这些需求的发展；
- 第 3 章呈现通用的、技术上独立的 NBDRA 系统；
- 第 4 章描述 NBDRA 的 5 个主要功能；
- 第 5 章描述系统及其生命周期管理注意事项；
- 第 6 章讲解安全和隐私；
- 第 7 章概括参考架构设计相关的高层次分类学；
- 第 8 章讨论未来的发展方向。
- 附录 A 总结了方案部署的注意事项；
- 附录 B 列出了术语和定义；
- 附录 C 定义了本文档中使用的缩略语；
- 附录 D 列出了本文档中使用的资源和参考文献。

1.5 本卷后续工作

开发 NIST 大数据参考架构（NBDRA）需要经历三个阶段，本卷（第 6 卷）将对三个阶段的进展以三个版本进行汇报，具体如下：（1）确定通用的关键参考架构部件；（2）定义 NBDRA 关键部件之间的通用接口；（3）通过通用接口建立大数据一般应用程序，验证 NBDRA。本文档（版本 1）以高层次描述和功能呈现 RA 全部关键部件。

版本 2 将集中在如何定义 RA 关键部件之间的通用接口，流程如下：

- 从 62 个已提交案例（51general+11SnP）中挑选几个，或者选择其它具有代表意义的案例；
- 与领域专家合作，鉴定系统协调者到其余 RA 部件之间的工作流和交互作用；
- 在规模小、易管理、界限清楚的密闭环境中，实现关键部件间的交互模拟；
- 汇总上述数据流及关键部件间的交互作用，并将其打包成通用接口。

版本 3 则将侧重于 RA 验证：界定的 RA 接口能否建立大数据应用程序？策略如下：

- 运用界定的通用接口操作版本 2 中同一组案例；
- 在版本 2 场景之外，鉴定并操作几个新案例；
- 从以上操作获取经验教训，增强通用接口。

注：版本 3 中开发的通用接口并非完美无缺，相反，这仅仅是一个初步成果，需要有关各方向更深层次细化完善。

2 高级参考架构的需求

大数据参考体系结构的发展涉及到对当前技术，问题，关注等全方位的理解。为此，美国国家标准及技术研究所大数据公共工作组（NIST Big Data Public Working Group， NBD-PWG）收集大数据的应用案例以了解大数据目前的应用现状。根据大数据的应用中的共性特征，开展了一次大数据参考架构的研究，提出了一种基于分类的分析方法，将收集的大量数据、现有大数据的相关技术、发展趋势和用户需求等进行分类，以便对大数据进行深刻理解。NBD-PWG 的这些成果为美国国家标准及技术研究所大数据参考模型（NIST Big Data Reference Architecture，NBDRA）的成型奠定了基础。

2.1 用户案例和需求

为了研究用户案例，在 9 大应用领域的大数据架构收集了大量公开的信息。NBD-PWG、需求组（Requirements Subgroup）、和其它感兴趣的第三方组织也提供了详细的用户案例。需求组使用模板来收集用户案例的细节，尝试着对响应进行标准化，并便于后续的分析和用户案例之间的对比。但是用户案例模板的不同部分，其数据有着不同的详细层次和不同的数量级别。《NIST 大数据互操作性框架草案第 3 卷：案例和总体需求》文档中提供有最原始的用户案例、可分析的汇编信息、和从这些案例中提取的用户需求。

下一步是从汇编的用户案例中提取需求，以形成一个参考架构。经过收集、处理、和对用户案例的评审，需求被整理为 7 个分类（在下表中的左列）。这些用户案例的具体需求被整理成高层次、通用的需求，且这些需求是技术中立的。

这些需求的分类是作为开发 NBDRA 的输入，直接映射到 NBDRA 的组件，如下表所示：

表 1: 用户案例分类特征到参考模型组件和架构的映射

用户案例分类特征		参考模型组件和架构
数据源	→	数据提供者
数据转换	→	大数据应用提供者
能力	→	大数据框架提供者
数据消费者	→	**数据消费者**
安全和隐私	→	**安全和隐私架构**
生命周期管理	→	管理架构
其它需求	→	所有组件和架构

一般性需求与 NBDRA 组件的映射关系，描述如下：

数据提供者的需求

- 可靠实时的、异步的、批处理的方式从集中，分布，和云数据源，传感器或设备获取数据
- 缓慢的、爆炸式的、或高吞吐量的不同数据源和计算集群之间的数据传输
- 多样的数据类型，从结构化数据到非结构化的文本、文档、图片、网站、空间信息、压缩数据、时间数据、空间数据、多媒体数据、模拟数据和仪器数据等

大数据应用提供者的需求

- 多样的计算密集类型、分析处理技术、机器学习技术
- 成批或实时的分析处理
- 处理不同的数据内容和模型
- 实时的数据处理，例如流处理、新内容的获取，数据追踪、可追溯、数据修改管理和数据边界

大数据框架提供者的需求

- 原有和先进的软件包
- 原有和先进的计算平台
- 原有和先进的分布式计算集群，协处理器、I/O 处理
- 先进的网络技术，例如软件定义网络和弹性数据传输，包括光纤、光缆、无线网络、LAN, WAN, MAN 和 WiFi
- 原有和先进的大量分布式数据存储
- 原有和先进的程序设计、应用、工具、软件库

数据消费者的需求

- 从处理的数据中快速搜索高相关性、结果准确和高召回的数据
- 多样的数据文件格式便于形式化、展示和报告
- 结果的可视化展示
- 丰富的接口供浏览器和可视化工具访问
- 高清晰、多维度的显示数据
- 流的形式给用户展示结果

安全和隐私需求

- 对敏感数据、隐私进行保护
- 确保多租户、多层策略驱动、沙盒、访问控制、身份认证在保护数据安全方面符合公认的管理、风险、承诺(governance, risk, compliance，GRC)，且保证机密性、完整性和可用性(confidentiality, integrity, availability， CIA)的最佳实践。

管理需求

- 数据质量加工包括预处理，数据分类，数据分类，噪声消除和格式转换
- 数据、用户梗概和连接的动态更新
- 数据生命周期和长期的维护策略，包含数据的出处
- 数据的有效性
- 数据的有效性的人工注释
- 数据丢失和损坏的预防
- 多站点文档，包含跨边界和地理上分散的文档
- 永久性的标示符和数据可追溯
- 不同数据源数据的标准化和整合

其它需求

- 移动平台有丰富的接口可访问数据的处理的结果
- 从移动平台可对分析处理过程进行性能监视
- 移动平台有丰富的可视化内容搜索和展示功能
- 对移动设备的数据获取和管理
- 跨移动设备和其它智能设备的安全

2.2 参考模型研究

由 NBD-PWG 的参考架构组撰写的 NIST Big Data Technology Roadmap Volume 5: Architectures 白皮书，是用来便于理解大数据中复杂的内容，还可以作为开发系统具体架构的一个参考框架。参考架构组调研了目前公开的大数据平台，且通过引导公司和个人支持大数据框架和分析的材料，揭示了一个完美的大数据体系架构。从这个文档中提取的通用主题可指导参考模型的整体研发。

2.3 分类

NBD-PWG 定义和分类子组（NBD-PWG Definitions and Taxonomy Subgroup）专注于定义大数据的概念，以及定义可以描述大数据模式和参考架构的术语。下面关于分类的部分提供了一个参考架构相关组件的层次结构。另外，分类的细节可参考 NIST Big Data Interoperability Framework: Volume 2, Taxonomy 的文档。

下图大致描述了由 NBD-PWG 定义和分类子组定义的七个角色的参与者和相关活动。每个白色方框包含了一个角色和隐含的参与者。每个角色可能的活动和各自的白色方框相连。

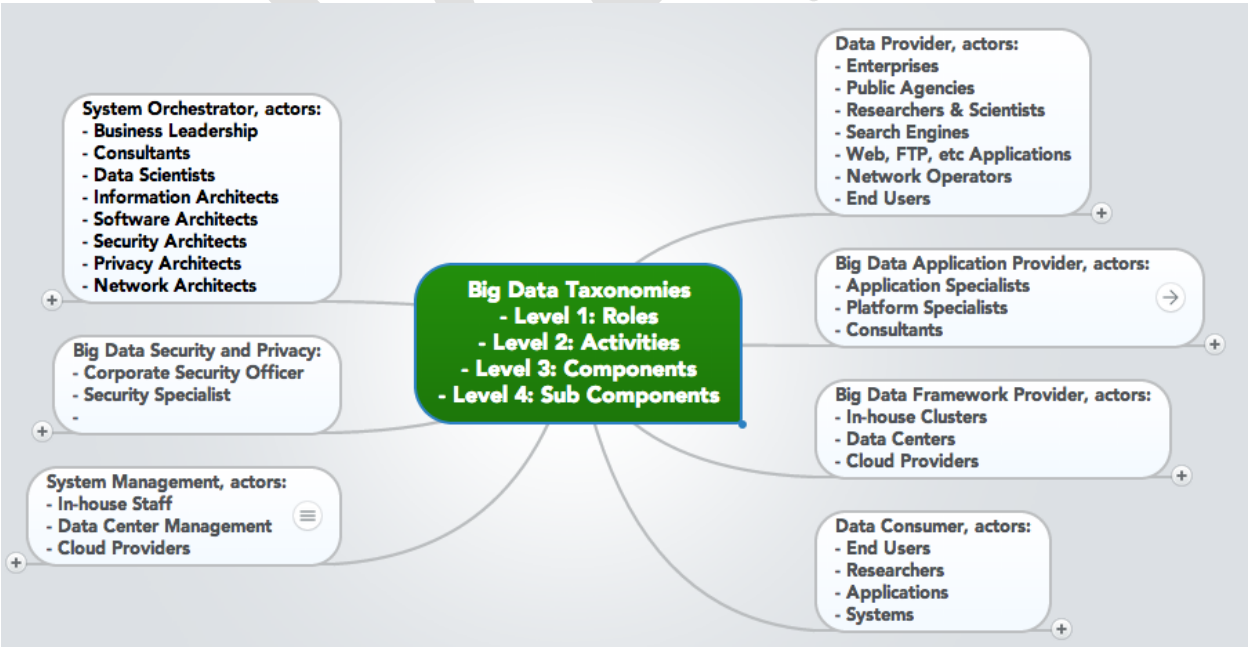

图 1: NBDRA 分类

2.3.1 系统协调者

系统协调者提供系统必须满足的总体需求，包含策略、架构、资源和商业需求，还包括监视和审计活动，来保证系统必须满足这些需求。该角色主要提供系统需求，高层设计，和数据系统的监视。其实该角色在大数据系统出现前就存在，因此该角色的一些设计活动需要根据大数据的模式做相应改变。

2.3.2 数据提供者

数据提供者需要保证数据对于他们自己和其他人是可用的。该角色的参与者可以是大数据系统的一部分，例如可以是组织内部其他系统的成员，或是组织外部的系统管理者。如果数据来自本地的系统，取回所需数据的需求将由大数据应用提供者发出并转给大数据框架提供者。但是，大数据提供者的概念并不新，更强大的数据收集和分析能力为提供更有价值数据提供了可能性。

2.3.3 大数据应用提供者

大数据应用提供者执行大数据生命周期中的相关操作，来满足由系统架构师制定的需求。这里是大数据框架中用来产生具体数据系统的关键所在。

2.3.4 大数据框架提供者

大数据框架提供者拥有可供大数据应用提供者在创建具体应用时使用的资源和服务。大数据框架提供者提供了很多子组件可供大数据应用提供者选择并利用这些资源和网络构建具体的系统。

2.3.5 数据消费者

数据消费者接收大数据系统输出的值。在许多方面，他们是同一个功能的接受者，该功能则是由数据提供者提供给大数据应用提供者。在系统给原始数据添加值后，大数据应用提供者则将给大数据消费者提供相同的功能。

2.3.6 安全和隐私结构

安全和隐私问题会影响 NBDRA 中的所有组件。安全和隐私组件与系统架构师为了策略、需求、审计、大数据应用提供者、大数据框架提供者的开发、部署和操作相互协作。无处不在的安全和隐私活动在 NIST Big Data Interoperability Framework: Volume 4, Security and Privacy Requirements 文档中有详细的描述。

2.3.7 管理结构

在 NBDRA 中，管理的作用是总体控制系统的执行，部署和维护。

3 NBDRA 概念模型

大数据参考架构是为系统工程师、数据科学家、软件开发者、数据架构师以及高级决策人员服务的。很多问题因为各种数据特性（数据量、种类、速度、精准度）紧密结合在同一大数据生态系统中，因而需要多样不同方法才能解决，而参考架构正是为这类问题提供了解决方案。大数据参考架构为支持包括高内聚型紧密交互的企业系统与松散耦合的垂直产业在内的各种商业环境提供了一个问题解决框架（参见附录 II.b2）。这是通过理解大数据是如何与现有的分析方法、商业智能、数据库以及系统体系结构互为补充，又各有不同的。

通过进行用例分析、参考架构调研和分类实施，NBD-PWG 参考架构小组开发了一个独立于供应商、技术方案和基础结构的大数据架构的概念模型。这个称作 NBDRA 的概念模型如图 2 所示，它展示了一个由相互关联的接口（亦称服务）连接的逻辑功能组件组成的技术独立的大数据系统。这些组件在大数据生态系统中扮演了不同的功能角色。

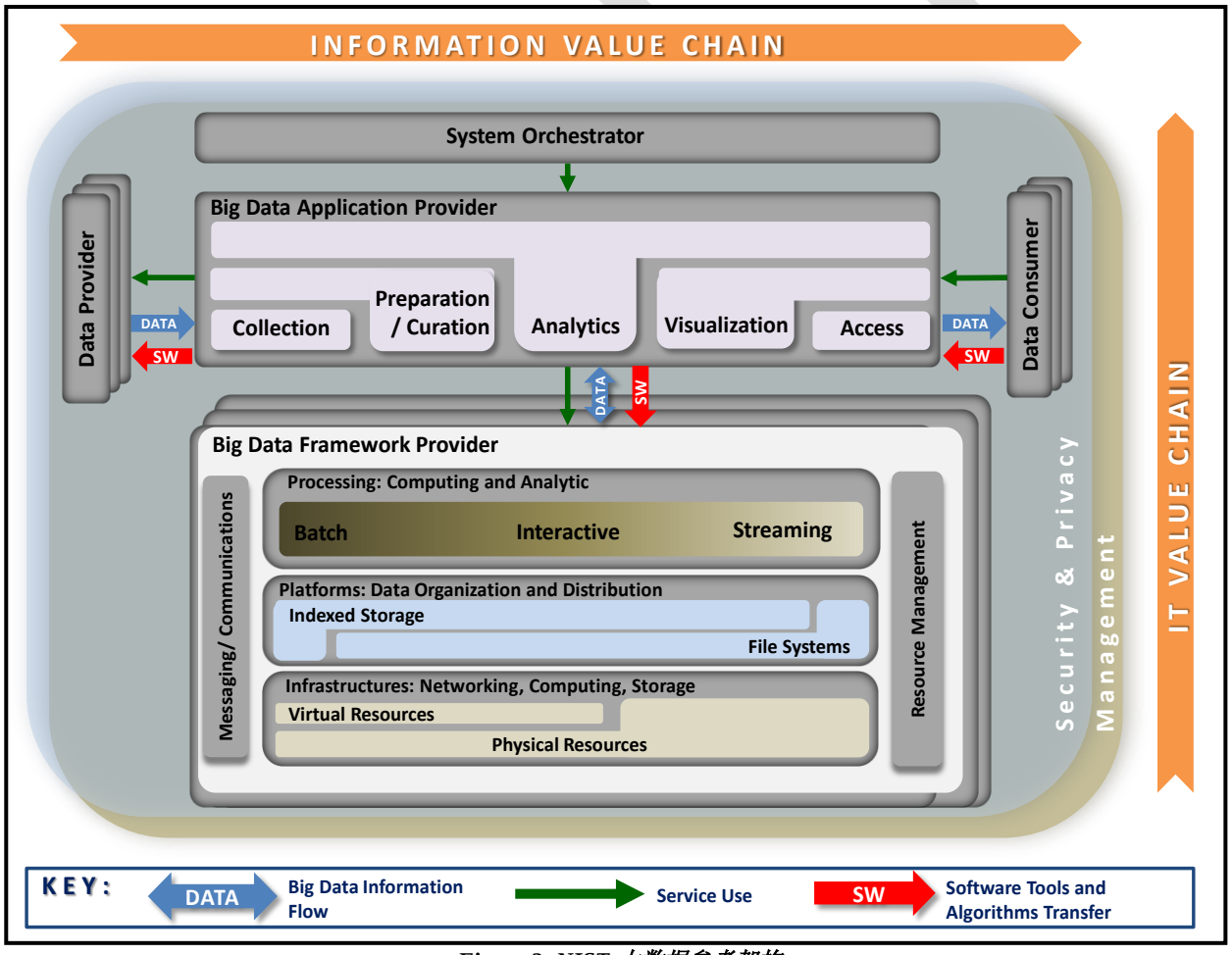

Figure 2: NIST 大数据参考架构.

NBDRA 围绕代表着两大数据价值链的两个轴线组织展开：信息流轴（水平轴）和 IT 集成轴（垂直轴）。信息流轴的核心价值由数据收集、数据集成、数据分析及其应用产生。IT 集成轴和

核心价值由提供网络、 基础设施、 平台、 应用工具和其他 IT 服务产生，这为大数据处理应用程序提供了托管和操作的支持。值得一提的是，大数据应用程序提供商组件位于两个轴线的交叉点，这意味着数据分析及其实现在这两个价值链中对大数据利益攸关者都具有特殊价值。

图 2 中所示和第 4 节中详细讨论了 NBDRA 的五个主要组件，这五个主要组件在每个大数据系统中扮演着不同技术角色。这些功能组件如下所示：

- 数据提供程序
- 大数据应用提供商
- 大数据架构提供程序
- 数据使用者
- 系统控制器

大数据应用提供商和大数据架构提供程序被统称为供应商，这表明在系统内由他们提供或实现特定的技术功能。

如图 2 所示，五个功能组件被涵盖在两种底层结构中，它们分别是：

- 安全和隐私
- 管理

这两种底层结构为系统的其他模块提供服务与功能，尤其是在大数据相关的领域，它们对帮助解决大数据相关的问题非常重要。

DATA 箭头显示了系统主要模块之间的数据流动。模块之间的数据流动要么是物理性的（例如通过数值传递）要么是通过传递数据的位置以及访问它的方法（即引用传递）。SW 箭头显示了大数据处理软件传输的原始位置。Service Use 箭头表示了软件的可编程接口。虽然 NBDRA 的主要目标是为了显示运行时环境，但实际上这三种类型的通信或事务也可以在配置阶段发生。人工协议 （比如服务协议 [SLA] ）和整个系统可能存在的人机互动不会在 NBDRA 中显示。

按照大数据分类法，一个参与者可以扮演多个角色，多个参与者也能扮演同一角色。NBDRA 并未划定参与商业业务的的利益相关者或执行者的边界，这样一个角色既可以只参与一个业务实体，也可以由不同的业务实体共同实现。因此，NBDRA 是适用于各种各样的商业环境，包括紧密结合的企业系统，以及依靠独立利益攸关者的松散耦合的垂直行业。

NBDRA 支持大数据系统堆叠或链接的表示形式。例如，一个系统的数据使用者可以通过堆叠或链接形式作为数据提供人把数据提供给下一个系统。

4 大数据参考架构功能组件

五个主要的架构模块代表在每个大数据系统中存在的不同技术角色：

- **系统协调者：**定义和集成所需的数据应用活动到垂直操作系统中来
- **数据提供者：**将数据和信息引入到大数据系统中
- **大数据应用提供者：**执行一个生命周期,以满足安全性和隐私需求，也包括系统协调者定义的需求
- **大数据框架提供者：**建立一个计算结构，在其中执行某些应用程序转换，同时保护隐私和数据的完整性
- **数据消费者：**包括最终用户或其他系统利用大数据应用提供者的结果

4.1 系统协调者

系统协调者的角色是定义和集成所需的数据应用活动到一个垂直操作系统中。通常,系统协调者代表一组更具体的一个或多个角色,负责管理和协调大数据系统的操作。这些参与者角色可能是人工组件,软件组件或两种组件的组合。系统协调者的功能作用是配置和管理大数据架构的其他组件执行一个或多个工作负载。分配/配置框架组件到单个物理或虚拟节点，提供一个图形用户界面,支持多个应用程序和组件连接在一起的工作流程规范。这种方式可能是低层次的。协调器也可以监视工作负载和系统，以确保服务要求的具体质量均达到每个工作负载，为满足变化引起的工作负载需求/激增的数据或用户数/交易，实际上可能弹性分配和提供额外的物理或虚拟资源。

大数据 RA 代表广泛的大数据系统:从紧耦合的企业解决方案(集成标准或专有接口)到松耦合垂直系统（由各种利益相关者、当局者或接口协议和标准来维护）。

在企业环境中,系统协调者通常起集中的作用,可以映射为系统协调者的传统角色，该系统协调者提供总体系统必须满足的需求和限制,包括策略、体系结构、资源、业务需求等。系统协调者与其他角色(如数据管理、数据安全、系统管理器)共同来实现需求和系统功能。

在松耦合垂直系统,系统协调者通常是起分散的作用。每个独立的利益相关者负责系统管理,安全,和集成。在这种情况下,每个利益相关者负责集成大数据分布式系统中其他利益相关者提供的接口。

4.2 数据提供者

数据提供者是大数据架构中的一种元素，可以代表任何东西,从一个传感器,人工手动输入数据,到另一个大数据系统。

数据提供者的角色是引入新的数据或信息反馈到大数据系统，供大数据系统发现、访问和转换。

注意,新的数据不同于已经在系统中使用并驻留在不同的系统库的数据，尽管用来访问数据的技术类似。

大数据系统的一个重要特征是能够导入和使用来自各种数据源的数据。数据源可以是内部和公共记录,在线或离线应用程序、磁带、图像、音频和视频,传感器数据、Web 日志,系统和审计日志,HTTP cookies 等数据源等，数据源可以由人工、机器、传感器、互联网技术等生成。

数据提供者创建一个抽象的数据源。数据提供者可以清理,纠正,存储数据为一个内部格式，该格式数据供大数据系统访问获取。

除非实现大数据应用提供者拥有的授权数据源，通常数据提供者角色和大数据应用提供者角色属于不同的授权系统。因此,来自不同数据源的数据可能有不同的安全与隐私方面的考虑。

数据提供者也可以提供一个抽象的数据，该抽象数据在转换前来自于另外一个系统，该系统可以是一个遗留系统或另一个大数据系统。在这种情况下,数据提供者则代表其他系统的一个数据消费者。例如,一个(大)流数据源可能由另一个系统操作(大)数据产生。

数据提供者活动包括：

- 收集数据
- 持久化数据
- 创建描述数据源的元数据,使用策略/访问权限,和其他相关属性
- 发布信息的可用性和访问它的方法
- 其他 RA 组件使用合适的可编程接口可以访问数据
- 提供推或拉访问机制
- 执行数据访问权限访问
- 建立正式或非正式的访问授权数据契约
- 提供转换函数实现隐式或显式的个人身份信息数据清洗

数据提供者将暴露接口的集合(或服务)用于发现和访问数据。这些服务通常包含一个注册表,以便应用程序可以定位数据提供者,它包含感兴趣的数据,允许了解什么类型的访问可以允许,了解支持什么类型的分析,数据源所在,如何访问数据,数据安全需求,隐私数据等。因此,该接口将包括注册数据源的方法,查询注册表和一组标准的注册表数据集。因为数据太大经常跨越网络，该接口还可以允许提交分析请求（软件代码实现一个特定的算法）与返回给请求者的结果。

受主题特征的数据特性(如数量、速度和多样性)和系统设计考虑,暴露和访问数据的接口都有不同的复杂性，并可以包括推和拉访问机制。这些机制包括订阅事件,监听数据,查询特定的数据属性或内容,并能提交代码的执行过程。

注意,数据访问并不总是自动的,可能涉及例如一个人工角色登录系统并提供新的数据转移的方向(例如,通过 FTP)。

数据提供者和和应用提供者的接口通常会经历三个阶段开始,数据传输和终止。启动阶段可以由任何一方,通常包括一些级别的认证/授权。此阶段可能还包括查询元数据来源或消费者可用的来自发布/订阅模型中的主题列表和传输任何参数如对象计数/大小限制或目标存储位置。另外该阶段可能会非常简单,一边打开一个套接字连接到另一边的一个已知端口。数据传输阶段有可能是数据提供者推送过来的数据或者从应用提供者获取的数据。也可能是一个单一的传输或涉及多个重复传输。在重复传输情况下的数据可能是一个连续流的事务/记录/字节。在推送的情况下,应用提供者必须准备接受异步数据,但也可能被要求确认(或消极确认情况下)收到的单元数据。在拉模式的情况下,应用提供者将专门生成一个请求,定义了通过参数返回哪些数据。返回的数据本身可能是一个流或多个记录/单位的数据，还可能包含多个请求/发送交易。终止阶段可能非常简单,比如一边简单的删除连接或可能包括校验和、计数、哈希计算传输完成的信息。

4.3 大数据应用提供者

大数据应用提供者的角色是通过在数据生命周期中执行的一组特定操作，来满足由系统协调者规定的要求，以及安全性、隐私性要求。大数据应用提供者是体系结构组件,包含了业务逻辑和

由体系结构所执行的功能。这些过程包括从各种来源收集数据，以传统的或新兴技术实现的多重数据转换和各种各样的数据使用。 大数据应用提供者的活动通常基于应用而定，因此不适合进行标准化。尽管如此，特别是当应用示范于一个垂直行业时，元数据、所定义的策略和应用程序子块间的交互可以被标准化

虽然许多任务早已存在于数据处理系统中，大数据中的数据规模、数据速度和数据多样性从根本上改变了它们的实现。各种算法和机制需要重新编写和优化，来创建交互的、适应数据集扩展的应用程序。

在生态系统中传播的同时，数据也被以不同的方式处理和转换，进而从信息中提取价值。大数据应用提供者的每个活动可以由独立的利益相关者实施，并部署为独立的服务。因此，“应用提供者”可以是单独的，也可是一个提供者的集合，集合中的每一个个体在数据生命周期中实现不同的步骤。每个应用提供者的功能可以是由协调者、数据提供者、数据消费者调用的一般服务，如 Web 服务器或文件服务器，也可以是一个或多个应用程序的集合，也极有可能是以上两者的某种组合。每个功能或单个应用组件/程序可能有执行多种功能的多个不同的实例。需要引起注意的一个关键事情是每个组件能够与底层框架提供者、数据提供者和用户交互。另外，这些部件可以并行执行或以任意顺序执行，并通过处理框架中的消息/通信元件频繁的与彼此通信。此外，应用提供者的功能，尤其是数据收集和访问功能，将与安全性结构交互来进行认证/授权和记录/保持数据的来源。

每个步骤既可以在单独的“框架供应者”上运行，又可以全部共用同一个“框架供应者”。对这些不同系统方法背后的考量取决于（可能不同的）技术需求，业务和（或）部署限制，包括隐私和其它政策方面的考虑。基准的参考架构不给出底层的技术、业务考虑和拓扑约束，从而使它适用于任何种类的系统方法和部署。

例如，应用提供者“自己的”基础设施将被表示为“数据框架提供者”中的一个。如果应用提供者使用“外部的”/“外包的”基础设施，它将被表示为参考架构中的另一个数据框架提供者。参考架构中的多个框架提供者表明这些框架供应者可以被用作单独的应用提供者。

4.3.1 采集

应用提供者的采集功能一般是负责处理与数据提供者的接口。该组件可以是一个一般的服务，例如由系统协调者配置的文件服务器或 web 服务器，来接收特定的数据集合。或者，该元件也可以是由应用指定的服务，用来推送数据或接收来自数据提供者的推送数据。因为该功能接收的数据比较小，它必须存储/缓冲所接收的数据，直到它被框架提供者持久化。这种持久化不一定需要物理介质，但它可以是一个内存队列或由处理框架提供的其它服务。采集功能很可能发生在 ETL/ ELT 周期中的数据萃取部分。在采集的初始阶段，具有相同结构的数据集（例如，数据记录）会被收集起来（并合并在一起），这会导致在一致性安全方面的考虑，策略等。创建初始元数据（如标识带有键的科目）有助于后续的聚合或查找方法。

4.3.2 准备/综合处理

准备功能很可能发生在 ETL/ ELT 周期中的数据转换部分(虽然分析功能也同样可能发生在数据转换部分上)。该功能的任务可能包括数据验证（校验/哈希，格式检查等）、清洗（消除不良数据记录）、移除异常值、标准化、格式化或封装。 该功能也常发生在当数据源经常被持久化到框架提供者的存储中的场合，以及来源数据被校验的场合。这个过程可通过操作（删除重复数据等）和索引来优化数据，从而优化分析过程。该功能也可以利用元数据键来生成一个扩展和增强的数据集，从而聚合来自不同的数据提供者/数据集的数据。

4.3.3 分析

应用提供者的分析功能发生在对大数据系统的低端业务逻辑进行编码的场合（高端业务流程逻辑由系统协调者编码）。该功能基于数据科学家的需求或垂直应用的需求，确定处理数据的算法来产生新的分析，解决技术目标，从而实现从数据中提取知识的技术。分析功能将利用处理框架来实现相关的逻辑。这通常会涉及提供功能的软件在批处理和/或流处理元件上实现分析逻辑。处理框架的消息传递元件可以用于将数据或控制功能传递给运行在处理框架上的应用程序逻辑。分析逻辑可被分解成多个模块，这些模块运行在通过消息传递元件相互传递信息的体系结构上。

4.3.4 可视化

应用提供者的可视化功能提供给最终的数据消费者处理中的数据元素和呈现分析功能的输出。此功能的目的是格式化和展现数据，进而更好地交流数据意义和知识。这种准备可能涉及到产生文本报告或以图形化的方式渲染分析结果。所产生的输出可以是静态的可视化，并简单地通过框架提供者来存储以访问。然而更普遍的情况是，可视化功能和访问功能、分析功能以及框架提供者(处理和平台)进行交互，以基于最终数据消费者提供的访问功能参数为其提供数据的交互可视化。可视化功能可以利用一个或多个应用程序库，或者利用特殊的可视化处理架构，完全由应用程序实现。

4.3.5 访问

应用提供者中的访问功能聚焦于用户的数据通信及交互。和采集功能一样，该功能可以是一个通用服务，例如由协调者配置的 Web 服务器或应用服务器，来处理来自数据消费者的特定请求。该功能与可视化和分析功能交互，来响应应用程序请求；通过使用处理和平台框架来检索数据，并响应数据消费者请求。此外，该功能确保描述类型元数据、管理类型元数据、元数据的保护以及元数据结构被访问数据的数据消费者捕获和维护，同时和数据本身同时传送给数据消费者。与数据用户的接口本质上可以是同步或异步的，并且可以使用一个拉取或推送的模式来传输数据。

4.4 大数据框架提供者

数据提供者通常由一个或者多个组件的实例组成，这些组件根据 RA 图中的 IT 价值链分成不同的级别。同一个级别的所有实例不需要使用同样的技术。实际上，大多数大数据的实现是由多种不同的技术方案组合起来实现的，这样可以提供较好的灵活性，可以满足来自应用提供者框架的各种不同的需求。组成框架提供者的三个子组件包含：

- 基础设施
- 平台
- 处理框架

最近许多大数据问题空间的进步主要集中在对框架进行设计以使之可以满足不同的大数据需求（大数据量、多种类等等）的同时还能保持其线性特征。大数据空间的大多数技术成果的产生也主要来自于这些进步，因此有很多的可用信息和附加的内容可以反映这一情况。下面的几节将用从下到上的顺序依次对这些子组件进行描述。

4.4.1 基础设施

框架提供者的本要素可以为大数据系统中的所有其他要素提供必要的资源以控制/运行其他的要素。通常这些资源是由一些物理资源的组合构成，这些物理资源可以控制/支持相似的虚拟资源。这些资源通常分为下面几类：

- 网络：从一个资源向另一个资源传输数据的资源

- 计算：用于执行和保持其他组件的软件的实际处理器和存储器
- 存储：大数据系统中保存数据的资源
- 环境：在建立大数据实例的时候必须考虑的物理厂房资源（电力、制冷等）

大数据框架组件可能直接部署在物理资源上，也可能部署在虚拟资源上。在某种意义上，所有的资源都有其物理体现。经常用来部署多个组件的物理资源可能会在很多物理节点上被重复使用，从而提供我们所知的水平可扩展性。在分配物理资源的时候经常会使用虚拟化的方式来获取弹性和灵活性。在云计算社区虚拟化经常被称为基础设施即服务（IaaS）。在大数据架构中虚拟化通常以下面三种基础形式中的一种体现：

- 本生式 – 在这种形式中，一个超级管理程序本生的运行在物理的设备上并对多个包含操作系统和应用的虚拟机进行管理；
- 托管式 – 在这种形式中，操作系统本生的运行在物理的设备上，但是一个包含操作系统和应用的超级管理程序运行在其上。在大数据架构中这种形式并不常见，因为这种形式增加了多余的操作系统层的开销；
- 货柜式 – 在这种形式中，超级管理程序的功能内嵌在操作系统中，并运行在物理设备上，应用则在货柜内运行，从而可以限制其对操作系统的访问和物理设备资源的控制。这种方式广泛地应用于大数据架构中，因为它可以进一步的减少开销（因为大多数的操作系统功能是单一共享资源），但是这种方式不被认为是安全和稳定的，因为当货柜控制/限制失败的时候，一个应用可能导致共享这些物理资源的所有应用失败。

接下来的几节将分类描述构成大数据基础设施的物理资源和虚拟资源。

4.4.1.1 网络

大数据基础设施架构必须解决一个方面是连接性问题。一些大数据的实施是独自处理已经保存在数据中心的数据，所以不需要离开局域网范围。但是其他的实施必须在大数据系统设置时，规划和实施大数据进出数据中心的移动。尤其是在选择具有这种类型传输需求的大数据系统的位置的时候，可能需要不仅要基于外部网络的连接性的可用性（带宽），而且要考虑 TCP 的局限性，因为大数据的主要发送者或者接收者的需要低时延（通过测量包的往返时间（Round Trip Time）来获得）。为了解决 TCP 局限性的问题，大数据架构师需要考虑一些基于 TCP 以外的专门设计用于传输大型的利于视频或者图像文件的先进协议。

另外，在构建外部连接的时候必须考虑的与大数据的快速化相关的问题之一是外部链接的全部可用性。在操作正确的时候一个给定的链接可以很容易地处理数据的快速化。然而如果链路的服务质量下降或者链路完全中断，数据就可能丢失或者成为永远不能复原的备份。甚至在一些一直的用例中，网络中断的应急计划涉及将数据传输到物理的媒介上，然后再将其物理的送到目的地。但是即使是使用这种方法，由于时间上的限制，还是需要将数据转移到外部媒介上已进行转送。

大数据的大量化和快速化经常是执行这一架构组件的驱动因素。例如，如果执行需要经常地在簇节点之间进行大型的多个 G 比特的数据传输，就需要高速和低延迟的链路。根据可用性的需求，可能需要冗余和错误容忍链路。其他网络基础设施的方案包含防火墙和其他便捷访问控制能力提供域名解析（比如 DNS）和加密。最后这一层可能还包含自动部署/提供容量/代理，以及对基础设施的广泛监视代理，通过管理元素来对这些代理进行平衡以执行特定的模型。

通过使用软件定义网络（SDN）和网络功能虚拟化（NFV）这两个最近出现的概念可以支持网络扩展性和系统扩展性。

4.4.1.1.1 软件定义网络(SDN)

对于分布系统和框架的性能非常重要但是又经常被忽略的，尤其对大数据的执行至关重要的是对网络资源的有效和高效的管理。最近在网络资源管理方面的取得的非常重要的成果是“软件定义网络 SDN”，SDN 使用虚拟框架来对共享的 CPU/内存/磁盘池进行管理，使用软件定义或者虚拟网络而不是使用传统的专用物理数据网络的方式对物理的网络资源池进行管理。通过 SDN 进行管理、I/O 和控制，多种物理资源，包含链路和实际的交换机都被放到池中，根据特定的功能和特定的应用来进行分配。分配的内容包含原始的带宽、服务质量（QOS）优先级，甚至实际的数据路由。

4.4.1.1.2 网络功能虚拟化(NFV)

随着虚拟化的到来，现在虚拟应用可以合理地支持大量的国务由专用设备执行的网络功能。可以以这种方式执行的网络功能包括路由/路由器、边界防卫（比如防火墙）、远程访问鉴权和网络流量/负载监控。网络功能虚拟化的重要优点包含所有与应用/软件/系统虚拟化有关的内容，比如灵活性、错误容忍和资源管理等。例如，在用户或者数据连接大量出现的时候自动的部署/提供附加的防火墙，并且在大量连接结束后自动解除部署的能力在处理与大数据的数据量相关的问题时非常重要。

4.4.1.2 计算

簇/计算基础设施逻辑分布可能从一个机架上的物理机的密度，到云服务提供者中运行的一套虚拟机，到向全局提供的对空闲的计算资源的访问的松耦合机都是不同的。这一基础设施还经常包含下层的操作系统和相关的用于对簇资源进行交互的服务。

4.4.1.3 存储

存储基础设施可以任意进行组织，从各自独立的本地磁盘到存储区域网（SAN）或者附加网络的存储。

这一技术方面有两点直接影响着它们对大数据解决方案的适用性。第一是容量（也就是数据的量），本地磁盘/文件系统是受可用的媒介大小限制的。HW/SW RAID（阵列磁碟资料储存）方案（这种情况下从本地到处理节点）通过允许将多片媒介看做一个单独的设备来帮助进行容量调节。但是这种方式受到该节点能够接受的媒介的物理维度和设备数量的限制。SAN 和 NAS 的执行（我们常说的共享磁盘方案）通过将存储合并到一个特定的存储设备中解决了这一限制问题，然而我们又遇到了影响大数据方案的第二个因素：传输带宽。即使网络和 I/O 接口都变快了，而且多个执行可以支持多个传输信道，I/O 带宽仍旧是一个限制因素。另外尽管 RAID 可以提供冗余性，这些机箱需要的热备份、多电力供应和多控制器仍经常变成 I/O 的瓶颈或带来企业运行中的单点失败。许多大数据执行通过在框架的平台提供者区执行分布的文件系统来解决这些问题。

4.4.1.4 环境资源

例如电力和加热、通风以及空调等环境资源是大数据框架提供者的基础设施中的至关重要的部分。他们是全局框架必须进行管理的有限的资源。

对大数据的成功执行至关重要的是基础设施的规模必须能够足够支撑应用的需求。基础设施的架构需要覆盖从基本的电力和冷气供应到扩展的带宽可连接性。大数据驱动的一个关键的金华市对服务器密度的增加（一个机架单元中更多的 CPU/内存/磁盘）。然而，随着这种密度的增加，基础设施，尤其是电力和冷气等可能不能在数据中心中进行分配，从而为每个几家提供足够的电力和气流以带走多余的热量。另外，由于管理能量消耗的成本高昂，已经有一些数据中心内部技术被开发，从而在电力峰值使用时切断电力或者将不使用的资源停止使用，以节约能源，减少电力消耗。

4.4.2 平台

平台元素包含逻辑数据的组织和分布，并结合相关访问应用程序接口或方法。该数据组织涵括从简单的分隔平面文件（flat files）到完全分布式的关系或列式数据存储。相应地，访问方法也涵盖了文件访问应用程序接口，到像 SQL 这样的各类查询语言。一个典型的大数据框架的实施将支持以下两者：一个基本文件系统方式存储，和一个或多个索引存储方法。基于特定的大数据体系的考虑，这种逻辑组织可能分布，也可能不分布于同一个集群的计算资源内。

平台还可以包括数据注册、元数据服务项目，以及诸如形式本体或分类标准这样的语义数据描述。

从很多方面来看，大数据存储架构的数据逻辑分布/组织借鉴了大多数遗留系统中较为常见的方式。下图 XXX 以概览的形式展示了大数据的各类数据组织方式。

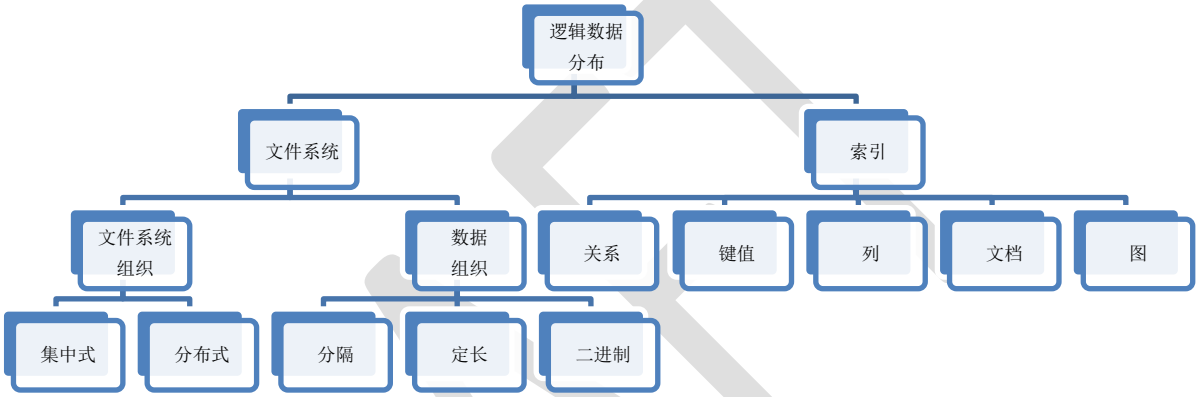

图 3: 数据组织方式

正如上面所提到的，许多大数据的逻辑存储组织是利用常见的文件系统概念（其中，数据簇被组编成一个目录分层命名空间）作为其基础，然后在单个文件中实施各种索引方法。这使得许多索引方法既可以运行于简单的本地存储文件系统中，用于测试目的；也可以大规模运行于完全分布式的文件系统中。

4.4.2.1 文件系统

许多大数据处理框架和应用程序直接从底层文件系统访问其数据。在几乎所有情况下，文件系统实施某种级别的 POSIX 标准以获取权限，进行相关的文件操作。这就使得在运行其他更高层次的索引或处理时，基层文件系统究竟是本地的还是分布式的，较为透明。基于文件的方法包含两个层面：文件系统组织，和文件内部的数据组织。

4.4.2.1.1 文件系统组织

文件系统通常为集中式或分布式。集中式文件系统基本上都是本地系统的实施；这些本地系统被放置在单个大型存储平台（SAN 或 NAS），通过某种网络性能进行访问。在一个虚拟环境中，多个物理集中式文件系统可以被合并、分割，或分配，从而创建多个逻辑文件系统。

分布式文件系统（也被称为集群文件系统）寻求采用以下方法解决这些问题：通过在数据块级别上多节点间镜像/复制数据，把每个节点上多个设备（主轴）的 I/O 吞吐量 和冗余、故障转移相结合。这是为大数据集群中使用异构商用硬件而专门设计的。万一单个驱动器或整个节点发生故障，由于数据是复制在其他节点上的，不会丢失任何数据；而且，因为数据处理可以转移到其他节点上，对吞吐量的影响也是最小的。 此外，复制使数据读取和原始写入两者之间得以实现高水准的并发。在创建被复制的数据块时产生的延迟会造成一致性问题（例如，某一个数据块被改

变了，而另一个节点读取的却是它被复制前的老的数据），所以对很多分布式文件系统而言，更新和交易方式的变化往往是个大问题。有些文件系统的实施也支持在多个层面上对数据进行压缩和加密。对此，要特别提请注意：基于分布式数据块的文件系统，其压缩/加密必须是可分割的，任何一个数据块都可以不按顺序，无须访问其他数据块的状态下被解压缩/解密。

分布式对象存储（有时也被称为全局对象存储）是分布式文件系统组织一个独特的例子。和这里描述的其他方法不同，分布式对象存储并未采用传统的文件系统分层命名空间法；相反，分布式对象存储推出的是一个扁平的命名空间，赋予任何给定的一簇数据一个全局 ID（GUID）。对存储的数据进行定位，一般是通过查询元数据目录，再由元数据目录回复相关的 GUID(s)。该软件的底层实施通常从 GUID 获知某个特定数据簇存储的位置。这些对象存储正在被开发和销售，用于大型数据对象的存储：从完整的数据集（如亚马逊的 S3）到大型个体对象（以几十 GB 计的高分辨率图像）。对于大数据而言，这些存储的最大局限往往在于网络吞吐量（又名：速度）。由于大多要求对对象进行总体访问，网络吞吐量可能成为一个限制因素。然而，引领未来发展趋势的理念为：是能够把数据发送给计算/应用，还是需要把数据导入应用。

从成熟度的角度来看，分布式文件系统有待改进的两个关键领域为：随机写入 I/O 性能和一致性；至少 IETF RFC 层面的标准的生成，诸如现用于 NFS 的标准。目前，无论是由一些商业服务机构提供并运作的分布式对象存储，还是作为像国家地理空间情报局（NGA）此类大型组织机构的规划蓝图的部分，分布式对象存储的实现基本上都是“火炉烟囱”式/专利所有型的。为了让它们在大数据生态系统里得以普及， 需要一定程度的互操作性（通过标准化的 API），需要基于标准的数据发现方法，或许最为重要的是，需要基于标准的方法，这些方法允许应用程序网格转移，本地运行数据，而不是把数据转移至应用程序。

4.4.2.1.2 基于文件的数据组织

大数据中基于文件的数据组织并没什么与众不同之处。它可以是文本或二进制数据，定长记录，或是某种定界结构（如逗号分隔值，XML）。

面向记录的存储（分隔或固定长度），除非单个记录超过数据块的大小，一般不会产生问题。一些分布式文件系统的实现提供卷或目录级别上的压缩功能，在逻辑块级别下面实施压缩（例如，当从文件系统读取一个数据块时，在数据块被返还之前，它会被解压缩/解密）。由于它们简单、熟识度高、可移植，在许多大数据的实施中，分隔文件经常是默认的存储格式。为此付出的代价是 IO 效率（又名：速度）。尽管分布式文件系统中的单个数据块可以并行访问，每个数据块仍需按顺序读取。就一个分隔文件而言，如果你感兴趣的只是某些记录的最后一个字段，而这些记录也许含几百个字段，这样就会造成大量的 IO 和处理带宽的浪费。

二进制格式往往是针对具体应用程序或实施的。由于数据相对较小（整数的每一个数位，在二进制里为 2 ~ 4 个字节，而在 ASCII 中则为 1 个字节），它们可以提供更为高效的访问，但它们能提供的不同实施中的可移植性非常有限。至少有一款流行的分布式文件系统提供了自己的标准二进制格式，该格式至少可以让数据在多个应用程序之间移植，无需额外的软件。即便如此，为了提升效率，下文讨论的索引数据组织方法大多采取了二进制格式。

4.4.2.2 索引储存结构

事实上，正是大数据本身的特性（主要是容量和速度）推动了对某种形式的索引结构的需求。大数据的容量要求无需扫描整个数据集，便可以迅速定位数据的具体要素。大数据的速度要求数据可以被快速定位，以用于数据匹配（例如，输入的数据是否和我现有数据集匹配），或用于获知写入/更新新数据的位置。

选择一个特定的索引方法或多个索引方法主要取决于你的数据类型，以及你想要实施的应用程序的性质。例如，图形数据（顶点，边，和属性）可以很容易地在平面文本文件中用顶点，边二元组，边，顶点，顶点三元组，或是顶点和边列表记录来表示。然而，高效处理这些数据 可能需要把整个数据集载入内存，或者需要能够把应用程序和数据集分布到多个节点；这样，每个节点的内存都存有图形的一部分。当图形各个部分含有和其他处理节点上的顶点相连接的顶点时，当然这就需要节点之间能相互沟通交流。这对那些像最短路径那样的图形应用程序是完全可接受的，特别是当图形是静态的时候。实际上，一些图形处理框架在运行时，使用的正是这个模式。然而，如果图形是动态的，或者你需要快速搜索或匹配图形的某一个部分，那么，对于那些需要专门图形存储架构的大型图形来说，这个方法就不可行了。

以下描述的各种索引方法，通常是按照它们在实施中提供的典型特性来予以分类，特别是它们可存储数据结构的复杂性，它们处理数据间联接的优劣性，以及如下图 4 所示，它们支持多路访问模式的难易度。因为这些特性均可以在自定义应用程序代码中实施，其描绘的值代表的是近似值规范。例如，键值存储适用于只需一键访问的数据，能够用单一平面结构来表示它们的值，多个记录间也无须相互关联。而文档存储，能够支持任意宽度的非常复杂的结构，往往是通过多个文档属性进行索引访问，但通常不支持记录间的关系。

务请注意：实际上，每种存储方法的具体实施差异极大，以至于被代表的特性的值实际为值域。例如，关系数据存储的实施越来越支持复杂数据结构，而在 bigtable 列式实施中本机添加更多弹性访问模式的工作也正在进行中。在大数据中，取决于要具体解决的问题，这些特性中的每一项的性能通常都会驱动该方法的可扩展性。例如，如果我的问题是要为某个独特的键查找一个单项数据，键值存储具有良好的扩展性。另一方面，如果我的问题需要在多个数据记录关系间进行通用导航，图形存储模型可能表现最为优越。

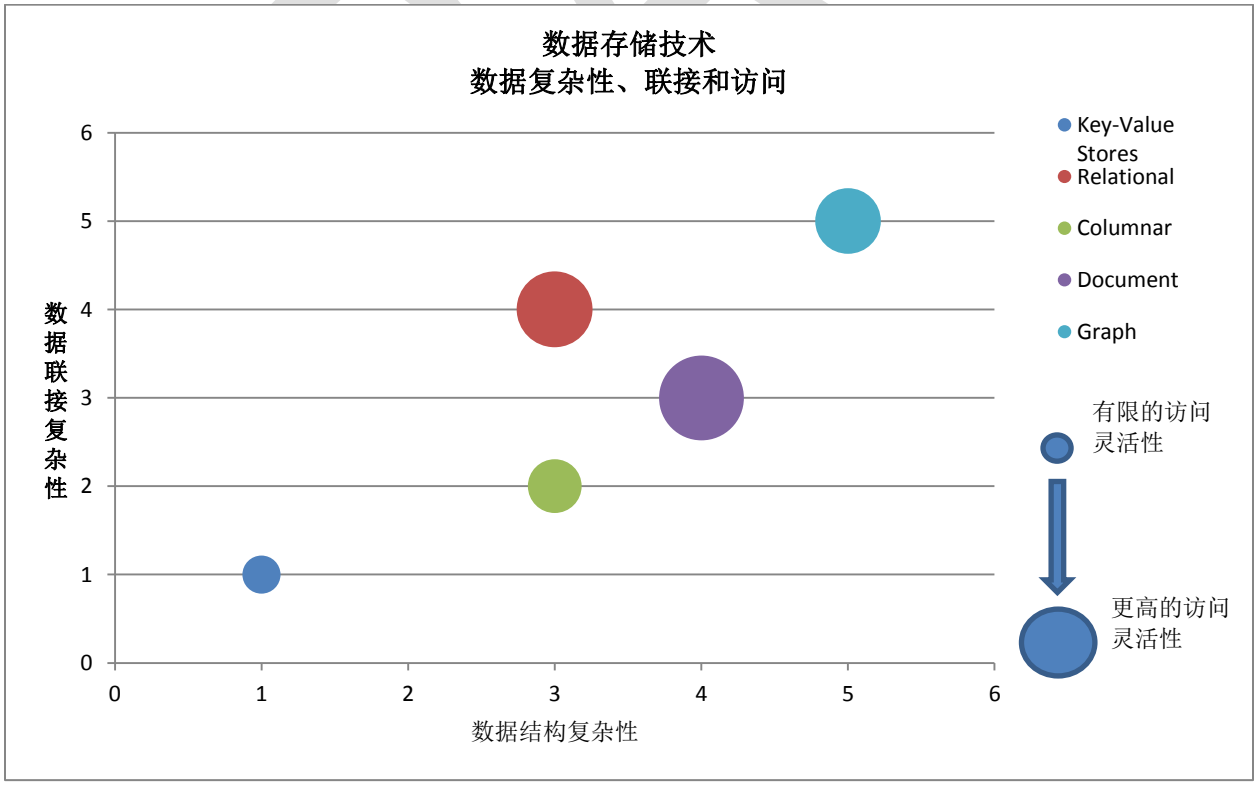

Figure 4: 数据存储技术

注：“Key-Value Stores”为“键值存储”，“Relational”为“关系”，“Columnar”为“列”，“Document”为“文档”，“Graph”为“图”。

以下段落将描述几种用于存储大数据的索引方法，每种方法各自常见的优势和问题。

4.4.3 处理框架

大数据处理框架提供了必要的基础设施软件以支持实现应用程序，来处理数据数量，速度和多样性。处理框架定义了如何将数据的计算/处理的组织结构。大数据应用依赖于各种平台和技术，以满足可扩展的数据分析和操作的挑战。

处理框架通常都专注于沿着批处理和流处理之间操作的数据交换或者数据操作。然而，根据具体的数据组织，平台，和实际处理要求，任何给定的框架实际可支持从高延时到近实时处理的处理范围。总的来说，多数大数据架构将包括多种框架以支持广泛的需求。

通常情况下处理框架是根据是否支持批处理或交互处理来进行分类。这通常是由用户或输出的角度来看（例如用户获得对请求的响应的速度）。然而，在现实中的大数据处理框架实际上应该解决三个不同处理阶段，一般拆分为数据摄取，数据分析和紧跟通过架构的数据流的数据发布。应用服务功能控制着具体框架的能力对这些处理单元的应用。例如，存在一个用例，其中数据以高速度进入系统，同时最终用户需要能够快速获取前一天数据汇总。在这种情况下，数据摄取必须是 NRT 并且跟得上数据流，分析部分可以是也可以不是增量的但可以是始于午夜或者某些组合的批处理，但是数据的获取（读取可视化）必须是可以互动的。根据具体的使用- 数据的变化可以发生在其在系统传输中的任何一点。例如，摄取阶段可能只是试图尽可能快地写下数据，或者它可能运行一些基本的分析跟踪的东西例如最小值，最大值，平均值等，因为这些可以进行增量计算。核心处理作业可能仅执行应用提供者所要求的分析元件和计算数据矩阵，或者可能实际上产生一些呈现就像一个热图以支持可视化组件来允许快速显示，数据发布部分激活肯定做了部分呈现，但多少要取决于数据本质和可视化。

为了讨论的目的，大多数分析框架可以根据它们主要被采用的地方在信息流中进行描述，如下图 5 所示。

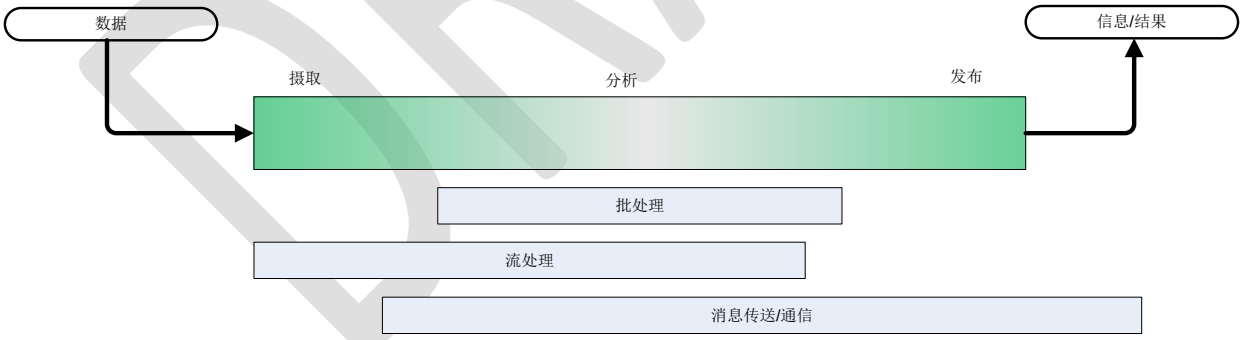

Figure 5: 信息流

上面的绿色阴影显示出该阶段处理对延迟的总灵敏度，它被定义为当一个请求或者一个数据片到达一个系统直到其被处理/交付完成的时间。对大数据而言，摄取阶段可能要求也可能不要求为跟上数据的实时性能，同时某些类型的分析（特别是那些分类为复杂事件处理）可以或可以不要求这种类型的处理。在最右边通常是数据消费者，根据用例批处理响应（例如夜间报告通过电子邮件发送）可能就足够了。在其他情况下，用户可能愿意为查询结果被返回等待数分钟，或者当关键信息到达系统时他们能够被立即告警。从另外一种方式来看待，批处理分析往往更好地支持长期的战略决策，在其中整体看法和方向不会受到最近改变的部分基础数据的影响。流处理分

析更适合新的数据需要立即采取行动的战术决策。一个主要用例是证券交易所的电子交易，在这里给定的片段数据的动作可以以微秒来测量。消息传送和通信提供了处理原件和缓冲之间的数据传输，来处理在数据率，处理时间和数据请求中的增量。

典型的大数据讨论重点围绕在提供分析的批处理和流处理框架，然而提供大数据交互式访问的获取框架也越来越普遍。当然这些分类的界限不是很固定或者很鲜明，因为一些框架提供了每个元件的各个方面。

4.4.3.1 批处理框架

来源于大型框架处理时期的批处理框架是一些大数据架构的最普遍和成熟的组件，很直接的因为数据的量通常需要很长时间来处理。批处理框架理想上并不依赖于特定的算法，甚至算法类型，而是提供多个算法类别可以执行的算法类型。并且，以大数据角度来讨论，这些处理模型经常分布在集群中的多个节点。它们根据在模型中的进程/活动之间的数据共享量来进行常规区分。

在 2004 年，工作于 DARPA 的高生产率计算系统（HPCS）计划的 Phillip Colella 开发了在物理科学中后来被称为“七个小矮人”（Colella，2004）的模拟算法列表。在最近，加州伯克利大学的 David Patterson 和 Katherine Yelick 根据“一个矮人是计算了计算和通信模式的计算方法”的定义，修改和扩展了列表条目至 13 个，如下表所示：

Table 2: 13 小矮人—物理科学的模拟算法

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	*原“7 个小矮人”算法之一 (被删除的是 Fast Fourier Transform, Particles, and Monte Carlo

最知名的两个批处理模型是 Map/Reduce（M/R）和批量同步并行（BSP），将在下面更详细地描述。

4.4.3.1.1 Map/Reduce

几大互联网搜索服务提供商推广了 Map/Reduce(M/R)，因为实现了他们的搜索能力。一般情况下，M/R 程序遵循如下五个基本阶段：

1. 输入准备和分配
2. 将几套密钥和值匹配成新的密钥和值：Map(k1,v1) -> list(k2,v2)
3. 将数据随机送入各个减速器中，各个减速器排序其输入- 每个减速器分配了几套密钥（K2）
4. 减速器运行在跟各个密钥关联的清单上（V2），并且产生输出：Reduce(k2, list(v2)) -> list(v3)
5. 最后输出的清单来自东减速器，由 K2 组合和排序。

需要注意的是，虽然只有一个单一输出，并没有在模型中禁止多输入数据集，并且是非常常见的为多个 M/R 作业的工作流而建立复杂分析。而这种编程模型是最适合于聚集（总和，平均值，基由）类型分析，各种各样的分析算法已在框架内实施。M/R 一般做不好需要直接更新基础数据的应用/算法,因为要更新单一密钥的值将需要对整个数据集进行读取，输出，则移动/复制到原始数据集。因为映射器和减速器在需要在部分数据或部分数据集的重复访问上进行迭代计算自然应用上是无状态的，往往不在 M/R 下衡量/表现良好。

大数据应用程序的 M/R 使用，由于其无共享的方式成为流行，以至于一些大型数据存储解决方案（主要是 NoSQL 类型）在其架构中进行实现。M/R 最重要的一个缺点就是对大多数实现的接口层次过低（用 Java 或者 Javascript 编写），然而，目前许多较为流行的实现，支持高层次的程序性和说明性语言界面，甚至可视化编程环境也开始出现。

4.4.3.1.2 批量同步并行（BSP）

BSP 程序模型最初是由哈佛大学的 Leslie Valiant 制定并在 1990 发表在 ACM 通信上。BSP 将并行处理和处理器的能力相结合，发送消息至其他处理器和显示同步步骤。BSP 算法是由包含三个不同元件的被称为超级步骤组成：

批量并行计算- 每个处理器执行本地块数据的计算/分析。

消息传递-由于每个处理器执行的计算可能会产生消息给其他处理器。这些消息被频繁更新成与其他处理器的本地数据相关联的值，但也有可能导致产生其他附加数据。

同步- 一旦一个处理器已经完成处理它的本地数据，它将暂停直到其他处理器也完成了它们的处理。

这个周期可以由所有处理器来终止，“投票停止”，它通常会发生在一个处理器不再产生任何消息给其他的处理器（如没有更新）。所有处理器轮流投票停止意味着没有任何对处理器数据的更新，计算已经完成。或者，这个周期可以在固定数量的步骤完成后终止（例如在一定数量的 monti-carlo 模拟的迭代之后）。

BSP 比之 Map/Reduce 的优点是处理可以实际上更新正在处理中的数据。正是这种区别让 BSP 在图形处理和模拟计算中流行，其在一个节点/元素上的计算直接影响了值或者和其他节点/元素的连接。BSP 的缺点是超级步骤之间的同步屏障的成本很高。如果处理器之间的数据分配和处理变得极度不平衡，则有些处理器可能会过载而其他在闲置状态。

众多的对基本 BSP 模型的扩展和增加以及制定并实施了多年，其中许多设计来解决同步问题的平衡和成本。

4.4.3.2 流处理框架

流处理框架用于处理要求处理速度跟其到达大数据系统的速度一致或更快的数据。流处理框架的最主要目标是减少数据到达系统与创建/存储/演示结果之间的延迟。流处理系统频繁涉及解决的其中一个问题领域被称为复杂事件处理或 CEP。CEP 使用来自一个或多个流/源近实时地推断或确认事件或模式。

几乎所有目前可用的大数据流处理框架实现了某种形式的处理流数据的基本工作流程。这些工作流程使用消息传送/通信框架在工作流程步骤之间传递数据对象（通常被称为事件）。这常常使用有向执行图的形式。流处理框架的显著特点通常是围绕着事件排序和处理保证，状态管理及分区/平行。以下分别介绍这些特点：

4.4.3.2.1 事件排序和处理保证

这个特性直接指向流处理部件是否保证事件/消息以他们被大数据接收的顺序来排序以及一个消息/事件可以或可以不被处理的频率。在一个非分布式的单流模式中这种类型的保证是无关紧要的。一旦我们引入了混合分布式和/或多流，问题就变得更加复杂。随着分布式处理，必须给每个分区的数据（见下面的分区和并行）强制执行保证。当一个分区的处理/任务/作业失败的时候，问题出现了。这些处理保证通常分为三类：

- 最多一次交付：这是最简单的保证形式，并当处理或通讯有事故或者到达发生乱序时，允许消息/事件被丢弃。这个类型的保证适用于在其中的新事件不依赖于先前事件产生的数据状态的数据。
- 最少一次交付：在这个分类，处理框架将跟踪是否每个消息/事件在一配置的时间框架内处理。未在要求时间处理完的消息/事件被重新引入流中。该模式需要由框架大规模的状态管理来跟踪哪些时间已经被哪个阶段的工作流程处理过。然而，在这个分类下，消息/事件可能被处理过不止一次，也可能到达是乱序。对于每一消息/事件必须被处理，但是顺序并不重要（例如对先前事件无依赖），并且应用程序要么不被事件重复处理影响或者本身有能力删除重复事件的系统，这类保证很好。
- 恰好一次交付：这类处理框架同最少一次交付一样要求顶层状态跟踪。但是框架内的嵌入机制检测并忽略重复事件。这个分类也通常保证事件顺序到达，并且需要应用于任何给定事件的处理依赖于先前事件的处理的应用程序。应该注意的是，这些保证只适用于框架内的数据处理。如果数据被传递至框架的处理拓扑外，则应用程序需要确定处理状态的元件由拓扑维持或者重复数据可以被转发至应用程序的非框架元件。

在后两种情况下，某种形式的唯一密钥必须与每个事件/消息关联，以支持重复数据删除和潜在的事件排序。

4.4.3.2.2 状态管理

流处理框架的一个关键特性是在框架内的一个进程或者节点故障的情况下，有恢复并且不丢失关键数据的能力。框架通常通过以某种形式存储的持久性数据提供该状态管理。此持久性可以是本地允许失败进程重新启动在同一节点上，到远程或允许进程重启在任何节点的分布式数据存储，或到被复制到其他的节点的本地存储器。它们之间的协定是由持久性引入的延迟。持续的状态数据量和确保数据持久性所需要的时间都贡献了延迟。在远程和分布式存储的情况下，所需要的等待时间通常取决于数据存储实现 ACID 或者 BASE 风格的一致性程度。在复制的最后方法，状态管理的可靠性完全依赖于复制恢复一个故障进程/节点的能力。有时，这种状态复制实际用用于与流处理器之间通信的同一个消息传送/通信框架来实现。实际上框架支持完整的事务语义，包括多阶段提交和事务回滚。协定是同样存在于任何事务系统，任何类型的 ACID 类保证将引入延迟。在数据流中的任何一点太多延迟可以创建瓶颈，根据顺序/进程，保证能到账死锁或者循环状态特别当某种程度的失败发生。

4.4.3.2.3 分区和并行

此框架的属性处理数据如何分布在节点和提供为处理数据流的量和速度横向可扩展性的工作任务。此分区方案必须与资源管理框架进行交互，以分配资源。任何分区方案的关键是数据均匀分布在整个分区，以使得相关的工作是均匀分配的。这直接涉及到可以均匀分配的密钥选择。最简单的形式是使用利用一个模式函数处理的任务/工作节点可用的一个号码。然而，如果有一个由相同工作节点处理的需要用一个公共密钥（如用户名，主机名等）的所有记录的数据，则确保在流周期内的均匀分配是非常困难的。一些框架通过支持动态分区来解决这个问题，即当一个工作节点过载并且备份的分区已经拆分，则要么分配给其他工作节点或者让新工作节点起转。为了使

之成功，特别是当有一个跟密钥相关的数据/状态，尤为关键的是，该框架有允许关联的状态数据被转移/传递给新的/不同的工作节点的状态管理功能。

4.4.4 信息交互/通信框架

长期以来，信息交互和通信框架在高性能计算（HPC）环境中具有很深的研究及科学基础，与此同时，该框架也在业内独领风骚。所有相关的框架体系既可适用于点对点传输模型，同样也被存储转发模型所接受。在点对点模型中，数据传输是从消息的发送者直接传送至接收者一方的，而大多数点对点模型架设并未提供任何形式的消息恢复机制以应对程序崩溃或发送者与接收者之间地通信中断。这些框架通常实现所有发送者和接收者程序层面上的逻辑，包括交付保证或消息转发能力。该模型的一个常见变化是当启用组播发送时，消息发送者会在一个“信道”内广播其消息，并且接收者会依次接收他们感兴趣的“信道”。当然，组播传输模式不会设置任何形式的消息回执。对于存储转发模型，发送者会事先定位单个或多个接收者，并将消息发送至第三方代理，以借此存储或转发消息至接收方。其中部分实例也支持某种形式的消息保存以应对系统或进程中断等带来的问题。同时，组播也可安装至此模型中，这通常被叫做发布/订阅模型。

4.4.5 资源管理框架

随着大数据系统技术不断成熟，架构逐步复杂，其商务模式也逐渐能更好的利用有限的计算能力与存储空间，借此以寻求更好的解决方案来支持层出不穷的应用及商务问题，因而，长期以来资源管理（框架）也取得了长足发展。资源管理和弹性计算等工具也不断的扩充增长、发展成熟进而满足云计算提供者及虚拟化技术的要求，伴随而来的也有大数据的特殊需求促使其不断优化。从某种角度上看来，大数据框架也更趋向于分布式计算模式，当然也将带来更多挑战。

如上文所述，在大数据发展的大背景下，卷存储与周转速率也不断提升其（现实）需求。弹性计算是解决数据存储空间及传输速度膨胀的一种最为常用的方法。在所有大数据解决方案中，需管理的最为关键的两个资源是 CPU 和内存。一旦此两者资源出现超负荷或不足便会对系统的运行带来极大的不便甚至导致崩溃。时至今日，大数据正在发生着日新月异的变化，同时也设计的更加复杂，其原因便是需要分配计算资源到不同的存储或处理框架中去，以促使数据结构和特定应用臻至完美。在如此资源管理框架内，通常的解决方案是部署使用数据局部性作为其中一项输入变量，以决定其对应的哪一项新的处理框架组件（主节点、处理节点、工作槽等）被实例化。然而，最为关键的也是最难以解决的即庞大的数据量（卷）是不允许被随意转至处理框架中的。此外，虽然大数据过程处理框架是可以运行在虚拟环境中，但是其中大多数的最初设计初衷是为了直接运行在实际硬件上以支撑数据卷的高效吞吐的。

为达成两种截然不同的解决方案，资源管理（模式）也一直伴随着大数据框架不断发展。首先是内部框架资源管理，在此框架中，其本身即可实现管理资源对于不同组件需求的分配，这种分配通常是框架的工作量，而且频繁“关闭”无用资源，以最小化系统上框架的整体需求，尽量降低能源消耗进而减少系统的营运成本。在这种解决方案中，应用可以设法调度并请求资源，就像其作为主框架操作系统是通过调度队列和作业类型实现管理一样。

第二种解决方案即框架间资源管理，此方案设计初衷即为满足大数据系统以支撑不同的存储与处理框架，进而有能力更好的处理更多应用程序。在该解决方案中，资源管理框架实际上作为一项服务运行，被用以支持和管理来自框架（系统）的请求，监控框架（系统）资源使用（情况），甚至可以管理应用队列。总体看来，该解决方案就像是通常云计算或虚拟化环境中的资源管理层，同样，使用者也就有可能搭建一套复合（混合）资源管理框架以同时处理物理与虚拟资源。

甚至进一步思考并整合这些概念，围绕着被称为"软件定义数据中心"的新兴技术。此类模型在云计算及弹性计算方面都远远领先于原先基于物理资源虚拟化出的固定的物理资源池式的管理。例如，全自动部署工具通过与虚拟化环境接口或框架 API 可以自动搭建起整个集群或额外添加物理资源进入物理或虚拟集群中去。

此类基础结构子组件的核心元素就是计算、存储及实现两者互联互通的网络连接。

4.5 数据消费者

本节中所涉及角色基本等同于数据提供者，它可以是一个实际的终端用户或是一个操作系统。通常情况下，该角色是一个具备完整大数据框架数据提供者的镜像，因此就该角色而言，它就是一个数据提供者。数据“消费者”通过调用大数据应用提供商提供的接口按需访问信息。上述服务包括数据汇总、数据检索、数据呈现。

该角色通常调用应用提供商提供的访问功能与其产生可视的、事后可查的交互。此类交互是基于指令集的操作模式，即用户发起交互或启动指令，应用提供商回复结果。因此，在此模式中，这种可视化交互形成报告，实现由应用框架提供商交付的商务智能的数据挖掘。抑或，此类交互是基于流或推送的操作模式，即用户仅订阅或接收一个或多个应用自动输出。在几乎所有情况下，大数据架构下的安全性与私密性，用来支撑用户与整个架构之间的认证与授权（无所谓哪一方做认证方或授权方）。这种模型可以看作是介于大数据架构与数据提供者，以及其与数据“消费者”之间的接口，经过三个典型的步骤即启动、数据传输和终止。

5 NBDRA 的管理构成

大数据的数量，速度及多样性的特性需要有一个多功能管理平台，目的是存储，处理和管理复杂的数据。大数据管理涉及系统，数据，安全和隐私的考虑，以确保达到高层次的数据质量和安全访问的目标。

如以上所讨论的，NBDRA 代表广泛的大数据的系统，从紧密耦合的企业解决方案（通过标准的或专有的接口集成），到松散耦合的垂直构成包含各种利益相关者或机构，通过协议和标准或准则性事实上的接口来捆绑。因此不同的考虑和技术解决方案将适用于不同的情况。

所述 NBDRA 的管理构成包括系统管理和数据生命周期管理，如下所述。

5.1 系统管理

针对于大数据的复杂性和容量面向传统管理平台提出了挑战。为了能有效地捕捉，存储，处理，分析和分发到具有高速度交换性的复杂及海量数据集，需要一种弹性的系统管理。

正如在传统的系统中，对于包含配置，监控和大数据基础设施维护的大数据架构系统管理，包括计算节点，存储节点和网络设备。换言之，可以考虑系统管理具有两个维度：一方面是配置，监视和维护的方面;另外一方面，这些要适用于计算，存储和网络资源。然而，由于如第四章节所描述的大数据基础设施的分布式和复杂性，大数据的系统管理是有挑战的，特别是用于控制，调度和管理处理流程的能力，以确保大数据应用提供者所需的可扩展性，强大和安全的分析。

事实上，基础设施可能包含 SAN 或 NAS 存储设备，云存储空间，NoSQL 数据库，Map-Reduce 集群，数据分析功能，搜索和索引引擎，以及短信平台。

此外，计算基础设施的范围可以从传统的数据中心，云服务，和网格的分散计算节点。系统管理结构将依赖于以下方面：

- 标准协议，如 SNMP
- 部署代理或管理连接器

这两个项目将有助于监控各类计算资源的运行状况和应对性能及故障事件，同时保证了大数据应用提供者所需的 QoS 级别。

管理连接器所需情景是云服务供应商通过 API 公开管理能力。可以想象的是，该基础结构元件包含自动化，自动调整，和自动修复功能，因此，降低了系统管理的集中式模型。由于与多个成千上万的计算和存储节点的潜在“大”的基础设施，工具和应用程序配置应该尽可能地自动化。

软件安装，应用程序配置，以及定期补丁维护，应推出并以自动方式跨节点复制，这可以基于基础设施的拓扑知识来完成。

随着虚拟化技术的出现，虚拟影像的运用可以加快恢复进程，并提供有效的修补，可以最大限度地减少定期维护的停机时间。

在企业环境中，管理平台通常会提供大数据分布式组件的企业范围监控和管理。网络管理，故障管理，配置管理，记帐系统，性能和安全性管理也非常重要。

在一个松散耦合的垂直体系里，每个独立的利益相关者负责自己的系统管理，安全性和集成。在这种情况下，每个利益相关者通用利用其他利益相关者提供的接口负责整合大数据分布式系统。

5.2 数据生命周期管理

对比传统的数据生命周期管理（DLM）不需要太多的数据传输，处理和存储，大数据生命周期管理（BDLM）具有很大的挑战。然而，BDLM 依然继承了 DLM 阶段的数据采集，流通，使用，迁移，维护和处置方面，而且面对更大的数据加工规模。

从 NBDRA 应用程序提供商来看，为了利用分析结果，它可能需要以利用分析结果多少来计算处理的汇集，准备/策展，分析，可视化和访问。

换句话说，BDLM 的作用是确保该数据是否被正确处理，由其他 NBDRA 组件参与，从它们被摄取到系统那一刻直到对数据进行处理或从系统中去除的完整数据生命周期全过程。

对于大数据，BDLM 是至关重要的，由于以下原因：

- 数据量可能非常大，可能会占用全部存储容量，或使存储传入的数据过于昂贵。
- 数据速度与可以被捕获并摄取到系统的数据随时在给定时间溢出的储存空间。即便是具有弹性的存储服务，云计算可以提供用于处理强劲增长存储需求，但不受控制的数据管理也可能是由于特定应用需求成为不必要的消耗。
- 不同的大数据应用将可能针对于数据生命周期有不同的要求。这对必须经常刷新数据以便有效处理结果，以及最新分析趋势和模式方面有用而产生影响。数据更新意味着旧的数据必须待处理，而不是送入分析或研究计划。与此同时，新的数据将被摄取并通过计算考虑。例如，实时应用程序将需要非常短的数据寿命。但消费者所感兴趣的产品研究可能需要收集较长一段时间数据便于挖掘数据。

由于大数据的计算环境，BDLM 的任务分布在不同的组织和/或个人之间，在遵守政策，法规和安全要求方面 NBDRA 组件之间协调数据处理存在更大的困难。

鉴于以上的背景下，BDLM 可能需要包括以下组件：

- **策略管理器** 要捕获的数据生命周期需求，使旧的数据待处理，新的数据由大数据的应用需要考虑。另一项功能即策略管理是维持指定的机制，数据转换/迁移和配置策略：转换数据，传输旧数据至较低的存储层备案，删除数据，或只在原地将它们标记。
- **元数据管理：**在 BDLM 中，由于元数据用于存储 管辖系统内数据的管理信息。元数据还包含重要信息，如数据的持久性识别，固定性/质量，接入权利。定义的元数据元素是至关重要的。此外，面临的挑战是找到最小元素的集合以有效的方式来执行所要求的 BDLM 策略。
- **辅助管理**
    - 用于安全隐私的数据屏蔽：隐私信息可能需要先于数据分析过程而匿名。例如，人口统计数据可以聚合和分析以揭示数据的趋势，但具体带有姓名和社会保障号码的个人身份信息（PII）应该屏蔽。屏蔽取决于应用程序的使用类型以及由安全和隐私指定授权的使用。
    - 数据的可访问性随时间而改变。例如，普查数据在 75 年后提供给公众。在这种情况下，根据策略和法律要求，BDLM 是负责触发可访问的更新数据或数据集。通常情况下，可访问性信息的数据被存储在元数据中。
- **数据恢复：**BDLM 还应该恢复数据，包括由于灾难或系统/存储故障导致的数据丢失。传统上，数据恢复可以通过使用常规的备份和恢复机制来实现。不过，由于大体积大数据，传统的备份可能是不可行的。相反，复制可能要在大数据生态系统中设计。根据数据丢失的公差 - 每个应用都有自己的公差水平 即寻找分析丢失的数据，复制策略必须被设计。

复制策略包括复制窗口时间，选择性要复制的数据，距离视差的要求。此外，为了应对大量的数据，数据备份和恢复应该考虑由大数据架构提供商提供的最新开发技术。

- **保存管理：**基本来说，系统需要以确保数据的完整性，以使分析过程的准确性和速度得到满足。由于极大海量数据，保存管理负责包含在系统配置过时的数据。根据保留策略，这些旧数据可以删除或迁移到归档存储。另一方面，在该情况下，数据需要保留多年，数十年来，甚至几百年，其中，一个保存策略将需要这样的数据，如果需要进行访问，可以由提供者组件提供，这将调用所谓的长期数字保存，可以通过使用大数据框架资源，由大数据应用提供商提供。

在大数据的背景下，BDLM 必须处理大数量，速度和多样化三'V'的数据特点。正因为如此，BDLM 及其组件与 NBDRA 其他组件交互。下面是这样的功能的例子：

- **数据提供商：**从数据源（S）到大数据系统 管理采集数据和元数据。这可能需要将数据提供登录元数据中的“项”事件。
- **大数据应用提供商：** 执行数据封装和为数据准备或展示目的而进行的格式转换。
- **大数据框架提供商** ：执行基本位级保护，以及数据备份和依据修复策略的数据修复。
- **数据消费者：** 确保相关数据和分析结果依据 BDLM 的政策让消费者及软件代理商采用恰当的访问控制去使用。
- **系统 Orchestrator：**使数据科学家开始处理来自辅助管理，数据备份/恢复和保护管理的任意组合。该过程可能涉及 NBDRA 的其它部件，诸如大数据应用程序提供商和大数据框架提供程序。例如，数据科学家可能要为了数据收集和策展与应用程序提供商进行互动，并调用大数据框架提供商来执行某些分析，并授予某些用户访问权限，从数据消费者那访问分析结果。
- **安全与隐私：**根据新的安全政策，法规确保 BDLM 是最新的。

在另外一个方向上，安全性和保密性也使用信息，该信息来自于 BDLM 相对于数据的可访问性。假设安全和隐私控制访问的功能和数据使用由大数据系统产生，该数据的访问控制可以通过管理和更新由 BDLM 的元数据被告知。

6 NBDRA 的安全性和隐私

对于安全性和隐私的考虑是大数据的一个基本方面，这会对 NBDRA 的所有组件产生影响。这种全面的影响在图 2 中标记为“安全与隐私”的灰色矩形框中给出了展示，矩形周围是所有的参考架构组件。一个大数据的参考架构至少应该提供并确保遵守 GRC、保密性、完整性、可用性（CIA）规则、标准以及最佳实践，并且可以进行验证。关于安全性参考架构的细节可参考 NIST 大数据交互性框架：卷 4，安全性和隐私需求。

除了大数据应用和框架提供商组件之外，数据提供商和数据消费者也是安全架构应该考虑的一个方面，至少数据提供者和数据消费者应该在适当的地方对安全协议和机制达成一致。

NIST 大数据交互性框架：卷 4,安全性和隐私，对与个人隐私和公司安全中的关键敏感区域相关的附加指导和最佳实践进行了讨论。

附录 A: 部署注意事项

NIST 大数据参考架构适用于各种各样的业务环境和技术。因此，可能的部署模型并不是这份文档主体所讨论的核心概念的一部分。但是，正如章节 4.4.1 中所描述的那样，大数据框架提供商功能性组件松耦合和分布式的性质可以使其使用多种基础设施元素来部署。最常用的配置方式通常有两种：一种是直接在物理资源上配置，一种是在 IaaS 云计算框架上进行配置。具体选择哪一种配置方式取决于效率/性能以及可伸缩性方面的需求。通常会使用物理基础设施以获得可预测的性能并高效地利用 cpu 和 I/O 带宽，因为它消除了大部分 IaaS 实现在虚拟环境中的额外开销和多余的抽象层。当有伸缩性需求以便于支持工作负载需求变化时通常会使用基于 IaaS 云的部署。迅速实例化额外的处理节点或者框架组件的能力使得部署能够适应工作负载增加或者减少的变化。因为部署足迹可以根据工作负载的需要而增长或收缩，所以当使用公有云或者共享云服务时，这种部署模型能够节省开支；当使用私有云时，能获得更高的使用效率和更高效的能源消耗。最近有一种称为云爆发的混合部署模型十分流行。在云爆发的物理部署中通过公有的或者私有的 IaaS 云服务来增强。当增加的工作负载需要额外的处理流程支持时，IaaS 基础设施会创建额外的框架组件实例，当不再需要这些实例时则会将其删除。

除了提供 IaaS 支持之外，云计算供应商现在还提供了 PaaS 模型下的大数据框架。在这种模型下，系统实现者可以从很多典型的大数据框架组件所需的复杂配置、部署的构建和管理中解放出来。实现者只需要简单的指定所需集群的大小，而所有框架组件的供应，配置以及部署由云提供商来管理。市场上甚至出现了一些专业化的 SaaS 大数据应用，它们在云环境中实现了大数据应用提供商的功能。

图 a – 1 阐述了大数据参考架构中的元素如何与 NIST 云计算参考架构对齐。以下部分描述了大数据架构元素与云服务提供商元素之间所需的一些高级别的交互。

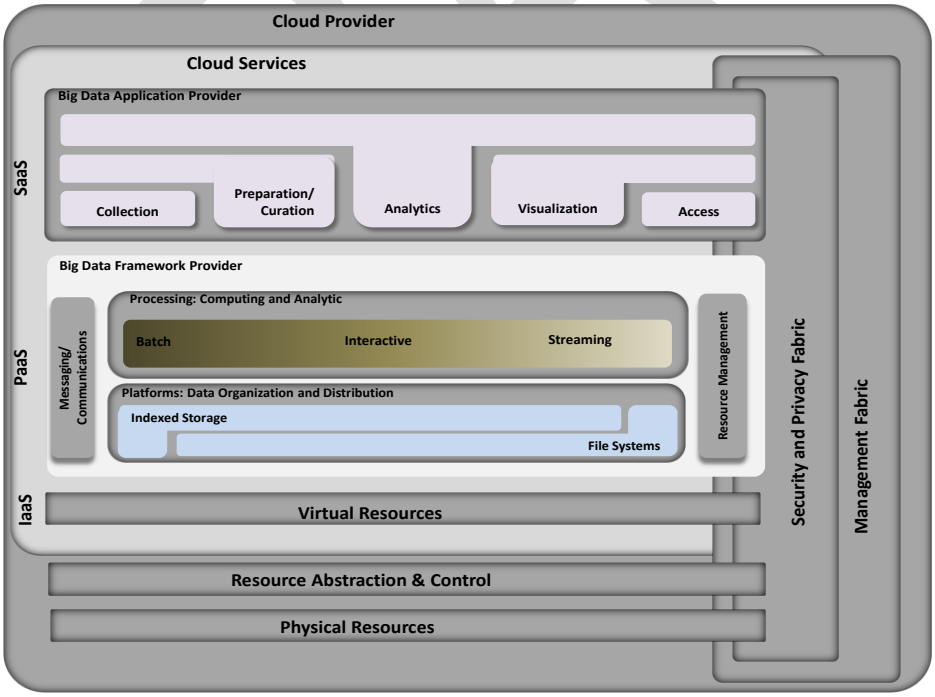

Figure A-1: 大数据框架部署选项

云服务提供商

最近，数据分析解决方案开始使用能够利用并受益于云计算系统框架的算法。云计算的基本特征包括：快速弹性、可伸缩性、多租户、按需自主服务以及资源池，这些特征组合在一起可以显著降低大数据实现的障碍。

云服务提供商(Cloud Service Provider, CSP)实现并交付云服务。服务调用的处理是通过服务实现的一个实例来完成的，这可能会涉及到其他服务的组成与调用，具体决定于服务实现的设计和配置。

云服务组件

云服务组件包括云服务提供商提供的云服务实现。它包含并控制着实现服务的软件组件（但不包括底层监控程序、主机操作系统、设备驱动程序等）。

云服务可以使用服务类别的形式来描述。

云服务也可以分为不同的类别，每个类别中的服务具有共同的本质特征。NIST 云计算参考模型定义了以下云服务类别：

- 基础设施即服务
- 平台即服务
- 软件即服务

资源抽象和控制组件

云服务提供商通过软件抽象使用资源抽象和控制组件来提供对物理计算资源的访问。资源抽象需要确保能够高效、安全、可靠地使用底层的物理资源。组件的控制特性允许对资源抽象特性进行管理。

资源抽象和控制组件使得云服务提供商可以提供诸如快速弹性、资源池、按需自助服务和可扩展这样的特性。资源抽象和控制组件可以包括监控程序，虚拟机，虚拟数据存储以及分时器等软件元素。

资源抽象和控制组件具备控制功能。例如，可能会存在一个集中的算法控制、关联和连接物理资源中的各种处理、存储和网络单元，它们在一起交付的环境可以提供一个 IaaS，PaaS 或者 SaaS 等云服务类别。控制器需要决定哪些 CPUs 或者机架包含哪些虚拟机，执行给定云负载的哪部分，处理单元之间如何进连接，以及当条件发生变化时如何动态透明地将部分负载分配给新单元。

安全性、隐私性和管理功能

几乎所有的情况下，云提供商都必须提供安全性，隐私性和管理功能的元素。通常提供商会支持高级别的安全/隐私功能以控制对大数据应用和框架的访问，同时框架本身必须控制对其底层数据和应用服务的访问。很多时候，大数据在安全性和隐私性方面的特定功能需要依赖且必须与云服务提供商所提供的功能交互。同样地，管理功能通常也分为大数据实现和云提供商实现。在这里，云提供商需要处理 IaaS 基础架构中大数据架构元素的部署和配置。云提供商可能会提供高层次的监控功能以便于让大数据实现可以跟踪组件的性能和资源利用情况。在许多情况下，大数据框架的资源管理元素需要与云服务提供商的管理框架对接以获取额外的资源。

物理资源部署

如上文所述，当强调性能时，经常需要在物理资源上进行部署。用来满足大数据需求的底层物理资源实现多年来获得了显著地发展。通过定制方式在节点间共享资源（内存，cpu，存储）的专业化、高性能的超级计算机已经让位于通过商业服务器来构建的无共享集群。为了发挥共享资源的优势，定制的超级计算架构几乎总是需要定制开发和定制组件。使用商业服务器不仅减少了硬件投资，同时还使得大数据架构可以为集群内的资源共享和管理提供更高级别的抽象。最近的发展趋势是优化服务器的密度、功耗，散热以寻求对可利用资源的最大化使用，同时最小化体积、功耗和冷却需求。这种方法保留了无共享方式抽象性和可移植性的优势，同时效率更高。

附录 B: 术语和定义

NBDRA 组件

- **大数据引擎：**当数据集的特性需要新的架构来进行高效存储、操作和分析时，利用独立资源构建可扩展数据系统的一种先进技术。
- **数据提供商：**引入信息并馈入大数据系统以供大数据系统进行发现、访问和转换的组织或者实体。
- **大数据应用供应商：**执行通用垂直系统数据生命周期的实体或组织，包括：（a）从各种来源收集数据，（b）利用传统技术和新技术实现多种数据的转换，（c）各种数据使用，（d）数据归档
- **大数据框架提供商：**一些组织或实体，他们提供执行某些大数据应用且满足安全性和隐私性要求的计算架构（如系统硬件，网络，存储，虚拟化和计算平台）。
- **数据消费者：**数据应用结果的最终使用者或者其他系统。
- **系统协调者：**定义并且将所需的数据转换组件集成到可运行的垂直系统中的组织或实体。

运行特征

- 可扩展性：
- 互操作性：在特定的条件下多种功能单元之间进行交流、执行程序或者数据转换的能力。
- 可移植性：不需要重建或者重新输入数据描述，也不需要对正在移植的应用进行大幅修改就可以将数据从一个系统转移到其他系统的能力。
- 隐私性：在整个生命周期中对与个人信息和个人身份信息相关的数据进行确定，合适以及一致的收集、处理、交流、使用和处置。
- 安全性：保护数据、信息和系统免受未经授权的访问、使用、披露、破坏、修改或销毁，目的是提供：
  - 完整性：防止数据被不恰当的修改或破坏，确保数据的不可抵赖性和真实性
  - 保密性：对于访问和披露保留授权限制，提供方法保护个人隐私和私有数据
  - 可用性：保证数据使用和访问的实时性和可靠性
- 弹性：随着实时响应工作负载的需求动态向上和向下扩展的能力。弹性取决于大数据系统，但是添加或者删除“软件线程”、“虚拟/物理服务器”是两种最常用的扩展技术。很多类型的工作负载需要驱动弹性响应，包括基于 Web 的用户、软件代理和周期性的批处理作业。

供应模型

- 基础设施即服务：为消费者提供处理、储存、网络以及其他基础运算资源，让消费者能够部署并运行任意软件（可以包括操作系统和应用）。消费者不能管理或控制底层的云基础设施，但是可以控制操作系统、存储和已部署的应用，对选择的网络组件有有限的控制权（例如主机防火墙）。
- 平台即服务： 消费者可以在云基础设施上部署自己创建或者获取的应用，但是这些应用需要使用供应商支持的编程语言和工具创建。用户不需要管理或者控制底层的云基础设施，包含网络、服务器、操作系统或存储，但需要控制部署的应用程序以及可能的应用主机环境配置。

- 软件即服务：为消费者提供的将应用运行在云基础设施上的能力。消费者不需要管理或控制底层的云基础设施，包括网络、服务器、操作系统、存储、甚至某个应用功能，有限的用户特定的应用配置可能是个例外。

附录 C: 大数据索引方法案例

本附录概述了一些常见的大数据索引方法。读者应该记住，新的创新性的方法会定期出现，同时其中的某些方法有可能是结合了几种索引技术特性的混合模型（例如关系型和列式、或者是关系型和图）。

关系型存储模型

该模型可能是人们最熟悉的，因为相关概念从 20 世纪 50 年代以后就存在了，同时结构化查询语言（SQL）也是一种成熟的操作（搜索、插入、更新和删除）关系型数据的语言。在关系型模型中，数据按照行的方式存储，每一行有多个列，每一列表示一个数据域，这些列和数据行基于逻辑数据组织成表。关系型存储模型和大数据的问题在于一个或者更多表之间的连接。虽然被连接的那两个或者多个表本身的数据量可能比较小，但是这些表的连接（或者关系匹配）操作所产生的数据记录可能会呈指数增长。对于组织而言，在大数据中采用该模型的吸引力可能在于对它很熟悉。但是问题是它有一些限制，同时更重要的是采用标准 RDBMS 实践的趋势（高度标准化、详细具体的索引）与性能期望。

关系型存储模型的大数据实现相对而言比较成熟，并且已经被一部分组织采用。同时这些方案在以提升响应时间为焦点的新实现上也成熟地非常快。很多大数据实现采用了非常野蛮的方式扩展关系型查询。基本上，查询会被分成几个阶段，但是更重要的是输入表的处理会被分散到多个节点上（通常是作为一个 MapReduce 任务）。数据实际上的存储介质可以是普通文件（分隔的或者固定长度），文件中的每一条记录/行表示表中的一行。但是，这些实现正越来越多地采用针对分布式文件系统而优化的二进制存储格式。这种格式通常会使用块级别的索引和面向列的数据组织方式，从而能够允许在不读取整条记录的情况下访问记录中的某一列。尽管是这样，大部分大数据关系型存储模型依然是为非常复杂的查询而设计的“面向批处理的”系统，连接会产生非常大的中间叉积矩阵，因此即使最简单的查询也需要 10 多秒才能完成。这种机制依然有重要的工作要做，而新兴的实现则正试图提供跟更好的交互式响应和接口。

早期的实现仅提供了有限的数据类型，对索引的支持也很少甚至完全不支持；而当前大部分的实现都对复杂的数据结构和基础索引提供了支持。虽然大部分现代 RDBMS 系统的查询计划器/优化器都非常成熟，它们通过对数据的分析实现了基于成本的优化，但是很多大数据实现中的查询计划器/优化器依然非常简单，本质上是基于规则的。虽然对于面向批处理的系统而言这一般是可以接受的（因为处理大数据的规模通常会产生数量级的影响），但是所有试图提供交互式响应的系统都需要非常先进的优化器以便于仅返回与查询内容最相似的数据。对于很多这样的实现，这显然会导致一个非常严重的倒退。因为分布式处理和存储基本上是为了实现可扩展性，这些实现直接受限于 CAP 理论（一致性、可用性和分区容错性）。实际上，它们中的许多提供了 T 最终一致性，也就是说禁止对数据做任何更新，分布式系统中的所有节点最终会返回最近的数据。对于数据仓库应用程序而言，这种级别的一致性通常还是比较好的，因为仓库中的数据很少更新，即使更新也是批量实现。但是，面向事务的数据库通常需要遵从一定程度的 ACID（原子性、一致性、隔离性和持久性）从而确保所有的事务都能得到可靠的处理，所有的冲突都能以一致的方式解决。很多工业和开源的组织都在寻找将这种类型的能力带入大数据关系型存储框架中的方法。一种方式是在已有的分布式文件系统实现之上实现一个传统的 RDBMS。虽然供应商声称使用这种方式意味着所有的技术都是成熟的，但是在这些实现的完整性能特性面世之前需要大量的研究和实现尝试。

键值存储模型

键值存储是最老且成熟的数据索引模型之一。实际上，键值存储的原理支持其他所有的存储和索引模型。从大数据的视角出发，这些存储有效地代表了随机访问的内存模型。虽然存储在值中的数据在结构上可以任意复杂，但是该复杂性的所有处理工作几乎都是由应用程序来负责，存储实现通常只是提供给程序一个指向某数据块的指针。键值存储能够非常好的适用于 1 对 1 的关系（例如，每一个键关联一个单独的值），同时也能够有效地处理键到同类型值列表的映射。如果键映射到不同类型/结构的多个值，或者如果某个键的值需要与不同的或者相同的键的值进行连接，那么需要编写定制的应用程序逻辑。这种需要通常会让键值存储无法针对某些问题进行有效地扩展。但是，依赖于这些问题，某些处理架构可以非常高效地使用分布式键值存储。当映射是 1 对 1 或者值的大小/长度不会发生变化的时候，键值存储通常能够非常好地处理更新。键值存储对插入的处理能力通常依赖于底层的实现。一般来说，键值存储还需要付出巨大的努力（人工的或者计算机的）从而能够处理对底层值的数据结构的改变。

在大数据应用程序中分布式键值存储基本上是使用最频繁的。一个始终必须要处理的问题（但是键值实现并不是唯一的）是键在可能的键值空间内的分布。尤其是，键的选择必须要慎重，避免数据在集群上的分布出现交叉。如果数据在小范围内交叉非常严重，那么当实现试图优化数据位置的时候就会导致集群上的计算热点。如果对于这样的实现而言数据是动态的（正在添加新键），那么很有可能某些点上的数据需要重新在集群上均衡分配。非位置的优化实现会采用各种各样的哈希、随机或者循环方法处理数据分布，这样不会受交叉和热点之痛。但是，在处理需要对数据集进行聚合的问题时，这些方法表现得都很差。

列式存储模型

很多有关于大数据的宣传都会提到 2006 发表的 Big Table 论文，但是像 Bigtable 这样的面向列的存储模型甚至对大数据而言都已经不是新事物了，在数据仓库领域它们已经应用了很多年。与传统的关系型数据存储系统通过相关值的行存储数据不同的是，列式存储通过相似值的组组织数据。这两者之间的差别非常微秒，在关系型数据库中一个完整的列组与一些主键（通常是一个或者多个列）绑定在一起形成一条记录。在列式数据库中，每一个列的值是一个键，相似的列值指向相关的行。一个列式存储最简单的实例与一个使用键值角色反转的键值存储相差无几。在很多情况下，列式数据存储看起来非常像关系型数据库中的索引。下面的图 5 展示了面向行的和面向列的存储之间的不同。

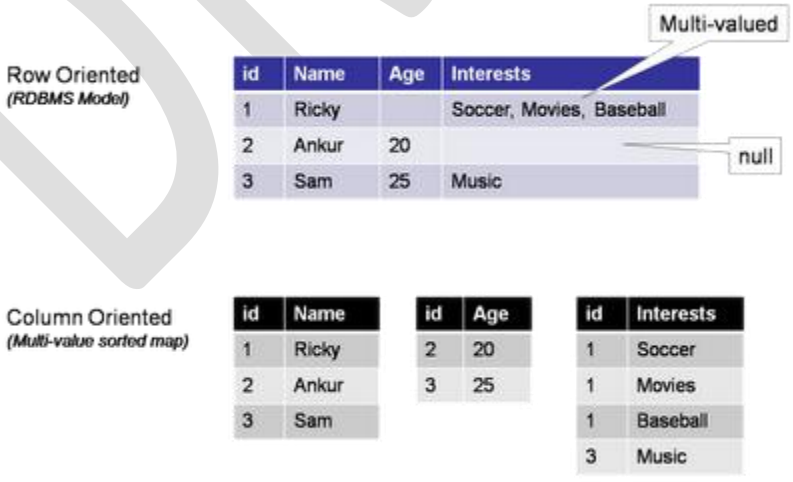

Figure B-1: 面向行的和面向列的存储之间的不同

另外，遵循 Bigtable 模型的列式存储实现在关系型模型的表、行、列模型之外引入了一个额外的分割层，称为列簇。在这些实现中，行拥有一个固定的列簇集合，但是在一个列簇中每一行都可以有一个可变的列组。下面的图 6 将会对此进行介绍。

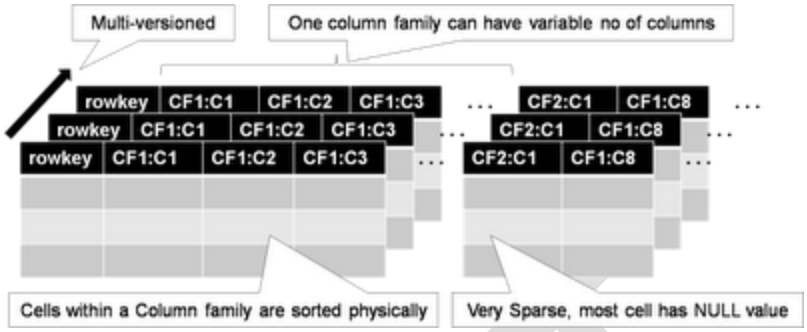

Figure B-2: 列式存储模型的列族分割

列式存储和关系型存储实现之间的关键区别是，列式存储中的数据是高度非规范化的，关系型存储中每条记录的所有列都会包含某个值（可能是 NULL），但是列式存储仅当一个或者多个行有数据的时候列才会存在。这也是为什么很多列式存储被称作稀疏存储模型的原因。每一个列簇的数据在磁盘上物理存储在一起，通过行、列名和时间戳来分类。最后一个时间戳也在那里，因为 Bigtable 模型还包含版本的概念。每一个行键、列簇和列三元组在存储时或者使用系统生成的时间戳，或者使用用户提供的时间戳。这样用户能够非常快地检索到一个列最新的值（默认），能够通过时间戳检索特定值，或者直接检索该列的所有值。最后一个是最有用的，因为它允许用户非常快速地对某个列的数据进行时域分析。

对于某个给定的列数据是存储在一起的，这样做有两个关键的好处。首先，对该列中的数据进行聚合运算时仅需要读取这一列的数据。而在关系型数据库系统中则需要读取整行（至少读取到这一列，如果行非常长同时列在最后面那么数据量可能很大）的数据。其次，对某一列进行更新时不需要读取/写入行其他部分的数据。同时，因为一个列中的所有数据都是统一的，我们可以对其进行更有效地压缩。通常，一个列在行键之后存储该值的地方仅会有一个值副本。虽然这样做在删除整个列时非常高效，但是如果是删除一条完整的记录则会非常昂贵。这也是为什么过去在 OLAP 类型的应用程序中使用列式存储，而在需要 OLTP 的系统中使用关系型存储的原因。

最近，安全已经成为已有列式存储实现的一个主要关注点，这主要是由于美国国家安全局（NSA）将自己的 Bigtable 实现发布到了开源社区。NSA 实现以及最近发布的一些其他实现有一个关键的优势，那就是它们实现了单元格层次的安全控制。通过这些实现，一个指定的用户可以根据这些或者其他单元格的值仅访问组中的某些单元格。

现在已经有一些非常成熟的分布式列式存储实现，既有开源的，也有商业的，并且在企业和政府组织中得到了广泛的应用。而新出现的系统则具有混合能力，它们在 BigTable/列式存储模型之上实现了关系型访问方法（例如 SQL）。同时，关系型实现也正在采用面向列的物理存储模型以便于为聚合分析等大数据 OLAP 提供更有效地访问。

文档

文档存储方式已经存在一段时间了，而流行起来则是因为用户需要对大量非结构化的数据进行快速搜索。现在，文档存储已经获得了长足的发展，为结构化数据和元数据提供了大量的搜索和索引能力，这也是为什么它们通常被称为半结构化数据存储的原因。在一个面向文档的数据存储中，每一个文档封装并编码该记录的元数据、域以及所有其他的表示。虽然与关系表中的一行有点相似，但是文档存储演进并获得流行的一个原因是：大部分实现都不强制使用一个固定的或

者恒定的模式。虽然保留那组文档的最佳实践应该是逻辑相关的，并且包含相似的数据，但是实际上它们不需要相似，不需要任意两个文档包含同样的域。这也是为什么对于域稀少的数据集文档存储非常流行的一个原因，因为通常其开销要比需要存储记录中空值列的传统 RDBMS 系统小的多。在这种类型的存储中文档组一般会被当作集合，同时类似于键值存储，它们通过某种唯一键引用每一个文档。

在现代的实现中，文档可以由任意嵌套的结构组成，可以包含可变程度的数组，在某些情况下可以是可执行的脚本/代码（有重大的安全和隐私影响）。大部分文档存储实现还支持对每个文档中的其他域或者属性建立额外的索引，很多还针对稀疏数据、地理空间数据以及文本数据实现了特殊的索引类型。

对文档存储中的数据进行建模时，最好的方式是尽可能地非标准数据，将所有一对一和大部分一对多的关系潜入到一个文档中。这样能够让文档的更新成为原子操作，保持文档之间的参照完整性。文档之间需要使用引用最常见的情况是：有些数据元素在文档集中频繁出现，同时这些文档之间的关系是静态的。例如，一个给定图书版本的出版商并不会发生变化，但是出版商的数量远比图书的数量要少得多，将所有出版商的信息潜入到每一本图书文档中并不合适，而是应该让每一本图书文档包含一个出版商唯一键的引用。因为引用对于图书的某个版本永远不会发生变化，所以这样并没有丢失参照完整性的危险。这样出版商的信息就能够和图书一样通过一个单独的原子操作进行更新。否则所有嵌入该出版商信息的图书文档都需要进行更新。

在大数据领域，文档存储可以通过分区或分片将集合的一部分分布到多个节点上从而实现水平扩展。分区可以是以循环的方式实现，确保数据平均分布；也可以基于内容/键实现，确保某个数据位置维护相似的数据。依赖于应用程序的需要，分区键的选择对性能有非常重大的影响，特别是在考虑聚合函数的时候。

文档存储实现没有标准的查询语言，大部分都使用衍生自其内部文档展现的语言（例如 JSON 和 XML）。

图

虽然 Facebook 和 LinkedIn 这样的社交网站有驱动图存储可见性和演进的动力，但是实际上多年来图存储已经成为很多问题领域关键的一部分，包括军事情报和反恐以及路线规划/导航和语义网络。图存储将数据表示为一系列的节点、边以及与之相关的属性。对图存储的分析包含非常基本的最短路径、实体消歧的页面排名以及图匹配。

图数据库通常存储两种类型的对象，节点和关系，正如下面图 7 所展示的。节点表示问题域中正在被分析的对象，可能是人、位置、组织、帐号等。关系描述这些对象在问题域中的相互关系。关系可以是无向的/双向的，但是一般会被表示为单向的，这样可以为关系提供更丰富、更富有表现力的表示。因此，对于父亲与儿子两个人的节点应该有两个关系。一个“是孩子的父亲”从父亲节点到儿子节点，另一个“是父亲的儿子”从儿子节点到父亲节点。另外，节点和关系可以有属性或者特征，它们通常是元素的描述数据。对于人来说，这可能是姓名、出生日期等。对于位置，这可能是一个地址或者地理坐标。对于像电话呼叫这样的一个关系，这可能是打电话的日期、时间以及持续时间。在图中，关系并不是始终相等或者有相同的权重。因此，关系一般会有一个或者多个权重、成本或者信任属性。人与人之间的强关系可能会有很高的权重，因为他们可能已经认识好几年了，同时可能每天都会交流。而两个仅见过面的人之间的关系则有较低的权重。节点之间的距离（物理距离或者难度）通常表示为关系上的一个成本属性，从而能够计算一幅图中真正的最短路径。在军事情报应用程序中，一个恐怖或指挥控制网络中的节点之间的关系可能仅仅是怀疑或者并没有完全查实，因此这些关系可能有信任属性。同时，节点上的属性还可以有与之关联的信任因素，尽管在这些情况下属性可以被分解到它自己的节点上并与关系绑在一

起。实际上，图存储方法可以看作是拥有两种类型文档（节点和关系）的文档存储模式的一种特殊实现。另外，在分析图数据的时候最重要的一个问题是定位你想要开始分析的节点或者边在图中的位置。为了实现该功能，大部分图数据库在节点或者边的属性上实现了索引。与关系型或者其他数据存储方法不同的是，大部分图数据库趋向于使用伪造的/虚假的键，或者指向唯一确定的节点和边。这样特征/属性能够很容易地随着数据的实际变化（有人改变了它们的名字）而变化，当有更多信息被发现的时候（例如，得到了某些条目或者事件的更好的位置）不需要改变关系的指向。

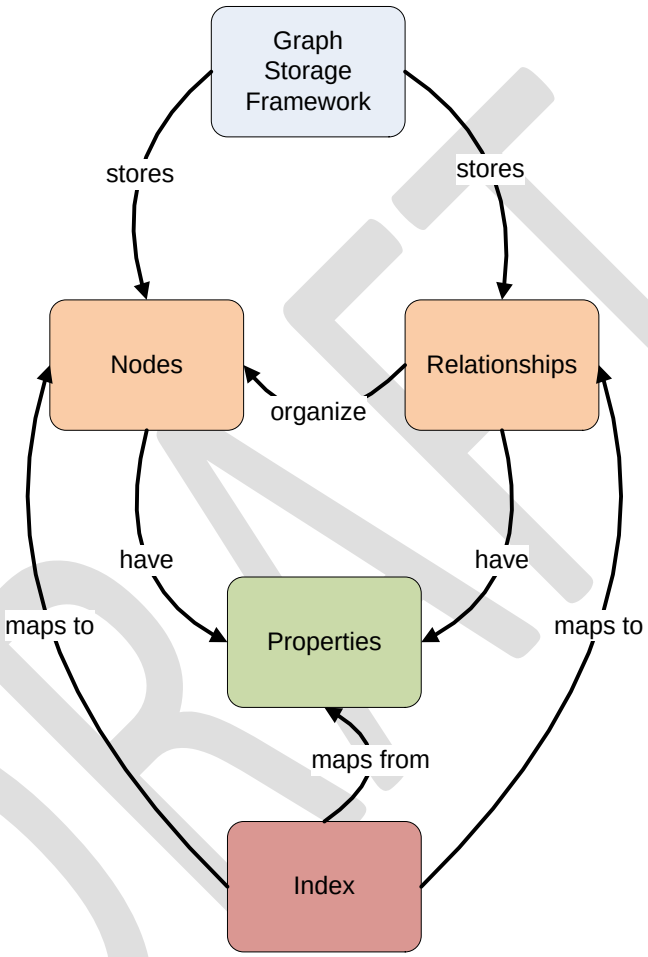

Figure B-3: 对象节点和数据库的关系图

图存储在大数据领域中的问题是，它们会变得很大难以放到单个节点的内存中，同时由于其自身一般具有无序（现实中有很少图遵循良好定义的模式）的特性针对分布式实现对其进行分区也存在问题。虽然节点之间的距离或者亲疏程序看起来好像是一个直接的分区方法，但是这样做也有很多问题需要处理。首先是数据的平衡问题。图通常在一个给定的区域会存在大量非常密集的数据簇，致使其从本质上就分布不平衡，同时还有热点处理问题。其次，无论图如何分布，跨边界的连接（边）始终存在。这通常需要节点清楚如何访问其他节点上的数据，需要节点间的数据传输和通信。这使得图数据处理架构的选择非常关键。不支持节点间通信/消息的架构基本上无法很好地应对大部分图问题。典型的分布式图处理架构将大块的图数据分配给节点，然后节点使用消息机制处理图中的变化或者一个路径上的某些计算值。

当我们在一个或者更多节点之间的分离度上寻找模式或者距离的时候，即使很小的图也会很快变为大数据领域内的问题。依赖于图的密度，这很快就会引发对需要测试的条件/模式数量的组合探索。

图存储的一个专门的实现是著名的资源描述框架（RDF），它是万维网联盟（W3C）发布的众多规范中的一员，通常与语义网络以及相关的概念有直接的联系。RDF 三元组由一个主题（Mr.X）、一个谓语（生活在）以及一个对象（Mockingbird Lane）组成。因此，一个 RDF 三元组的集合表示并定向标注了图。RDF 存储的内容常用正式的本体语言描述，例如 OWL 或者 RDF 模式（RDFS）语言，它们建立了底层数据的语义关系和模型。为了与异构数据集更好地水平集成 (Smith, Malyuta, Mandirck, Fu, Parent, & Patel, 2012)，人们提出了数据描述框架（DDF） (Yoakum-Stover & Malyuta, 2008)等理论对 RDF 概念进行了扩展，它们添加了额外的类型以便于更好地支持语义互操作和分析。

图数据存储现在缺少任何形式的标准化 API 或者查询语言。但是，W3C 已经为 RDF 开发了 SPARQL 查询语言，该语言目前在推荐状态，同时在使用 RDF 以及其他的图数据存储时 Sesame 等框架正变得流行起来。

附录 D: 缩略词

APIs	application programming interface
CIA	confidentiality, integrity, and availability
CRUD	create/read/update/delete
CSP	Cloud Service Provider
DNS	Domain Name Server
GRC	governance, risk, and compliance
HTTP	HyperText Transfer Protocol
I/O processing	Input/Output Processing
IaaS	Infrastructure as a Services
IT	Information Technology
LAN	Local Area Network
MAN	Metropolitan Area Network
NaaS	Network as a Service
NBD-PWG	NIST Big Data Public Working Group
NBDRA	NIST Big Data Reference Architecture
NIST	National Institute of Standards and Technology
PaaS	Platform as a Service
PII	Personally Identifiable Information
SaaS	Software as a Service
SANs	Storage Area Networks
SLAS	Service-level Agreements
VM	Virtual Machine
WAN	Wide Area Network
WiFi	

附录 E: 参考资料

GENERAL RESOURCES

The following resources provide additional information related to Big Data architecture.

[1] Office of the White House Press Secretary, “Obama Administration Unveils “Big Data” Initiative”, *White House Press Release* (29 March 2012)
http://www.whitehouse.gov/sites/default/files/microsites/ostp/big_data_press_release_final_2.pdf

[2] White House, “Big Data Across The Federal Government”, 29 March 2012,
http://www.whitehouse.gov/sites/default/files/microsites/ostp/big_data_fact_sheet_final_1.pdf

[3] National Institute of Standards and Technology [NIST], Big Data Workshop, 13 June 2012,
http://www.nist.gov/itl/ssd/is/big-data.cfm

[4] NIST, Big Data Public Working Group, 26 June 2013, http://bigdatawg.nist.gov

[5] National Science Foundation, “Big Data R&D Initiative”, June 2012,
http://www.nist.gov/itl/ssd/is/upload/NIST-BD-Platforms-05-Big-Data-Wactlar-slides.pdf

[6] Gartner, “3D Data Management: Controlling Data Volume, Velocity, and Variety”,
http://blogs.gartner.com/doug-laney/files/2012/01/ad949-3D-Data-Management-Controlling-Data-Volume-Velocity-and-Variety.pdf

[7] Gartner, “The Importance of 'Big Data': A Definition”,
http://www.gartner.com/DisplayDocument?id=2057415&ref=clientFriendlyUrl

[8] Hilbert, Martin and Lopez, Priscilla, “The World’s Technological Capacity to Store, Communicate, and Compute Information”, Science, 01 April 2011

[9] U.S. Department of Defense, “Reference Architecture Description”, June 2010,
http://dodcio.defense.gov/Portals/0/Documents/DIEA/Ref_Archi_Description_Final_v1_18Jun10.pdf

[10] Rechtin, Eberhardt, “The Art of Systems Architecting”, CRC Press; 3rd edition, 06 January 2009

[11] ISO/IEC/IEEE 42010 Systems and software engineering — Architecture description, 24 November 2011, http://www.iso.org/iso/catalogue_detail.htm?csnumber=50508

DOCUMENT REFERENCES