Apache Cassandra 简介

Apache Cassandra 是一个开源的、分布式、无中心、弹性可扩展、高可用、容错、一致性可调、面向行的数据库，它基于 Amazon Dynamo 的分布式设计和 Google Bigtable 的数据模型，由 Facebook 创建，在一些最流行的网站中得到应用。

为什么会诞生 Apache Cassandra

2007 年 Facebook 为了解决消息收件箱搜索问题（ Inbox Search problem ）而开始设计 Cassandra 项目。 当时 Facebook 遇到了传统的方法难以解决的超大数据量存储可扩展性问题。具体来说，项目团队需要处理大量的消息副本、消息的反向索引等不同形式的数据，需要处理很多随机读和并发随机写操作。

该团队由 Jeff Hammerbacher 领导，核心工程师包括 Avinash Lakshman，Karthik Ranganathan 和搜索团队的 Prashant Malik 。2008年7月 Cassandra 的代码被作为开源项目发布到 Google Code。虽然代码在2008年作为 Google Code 的一个项目，但是这段时间基本上只有 Facebook 工程师来更新代码，基本上没有形成社区。所以在2009年3月，Cassandra 被转移到 Apache 孵化器项目，并在2010年2月17日，它被投票成为一个顶级项目。 在 Apache Cassandra Wiki 上，您可以找到 committers 列表，其中许多人自 2010/2011 年以来就一直参与该项目。committers 主要来自 Twitter，LinkedIn，Apple，其中还包括了许多独立开发者。

Cassandra 的设计很大程度受 Amazon Dynamo 的影响，具体可参见《Dynamo: Amazon's Highly Available Key-value Store》。2010年由 Facebook 的 Lakshman 和 Malik 在 ACM 首次发表了 Cassandra 的论文 《Cassandra: a decentralized structured storage system》，向全世界介绍了 Cassandra。

随着商界对 Cassandra 的兴趣增加，对 Cassandra 的生产支持变得越来越明显。2010年4月 Cassandra 的 Apache 项目主席 Jonathan Ellis 和其同事 Matt Pfeil 成立了一家名为 DataStax 公司（最初名为 Riptano ）。DataStax 雇佣了多名 Cassandra Committer，为 Cassandra 项目提供了相关支持，并引领其发展。

Cassandra 的名字由来
在希腊神话里，Cassandra 是特洛伊国王 Priam 和 Hecuba 王后的女儿。Cassandra 非常美丽，以至于阿波罗给了她预见未来的能力。但当她拒绝阿波罗的爱慕的时候，遭到他的诅咒。从此，她依然可以精确地预知未来，但是不会有任何人相信她。Cassandra
预知了她的特洛伊城终将覆灭，但却无力阻止这一悲剧。Cassandra 分布式数据库就据此命名。

Apache Cassandra 特性

分布式和去中心化（Distributed and Decentralized）

Cassandra 是分布式的，这意味着它可以运行在多台机器上，并呈现给用户一个一致的整体。事实上，在一个节点上运行 Cassandra 是没啥用的，虽然我们可以这么做，并且这可以帮助我们了解它的工作机制，但是你很快就会意识到，需要多个节点才能真正了解 Cassandra 的强大之处。它的很多设计和实现让系统不仅可以在多个节点上运行，更为多机架部署进行了优化，甚至一个 Cassandra 集群可以运行在分散于世界各地的数据中心上。你可以放心地将数据写到集群的任意一台机器上，Cassandra 都会收到数据。

对于很多存储系统（比如 MySQL, Bigtable），一旦你开始扩展它，就需要把某些节点设为主节点，其他则作为从节点。但 Cassandra 是无中心的，也就是说每个节点都是一样的。与主从结构相反，Cassandra 的协议是 P2P 的，并使用 gossip 来维护存活或死亡节点的列表。关于 gossip 可以参见《分布式原理：一文了解 Gossip 协议》。

去中心化这一事实意味着 Cassandra 不会存在单点失效。Cassandra 集群中的所有节点的功能都完全一样，所以不存在一个特殊的主机作为主节点来承担协调任务。有时这被叫做服务器对称（server symmetry）。

综上所述，Cassandra 是分布式、无中心的，它不会有单点失效，所以支持高可用性。

弹性可扩展（Elastic Scalability）

可扩展性是指系统架构可以让系统提供更多的服务而不降低使用性能的特性。仅仅通过给现有的机器增加硬件的容量、内存进行垂直扩展，是最简单的达到可扩展性的手段。而水平扩展则需要增加更多机器，每台机器提供全部或部分数据，这样所有主机都不必负担全部业务请求。但软件自己需要有内部机制来保证集群中节点间的数据同步。

弹性可扩展是指水平扩展的特性，意即你的集群可以不间断的情况下，方便扩展或缩减服务的规模。这样，你就不需要重新启动进程，不必修改应用的查询，也无需自己手工重新均衡数据分布。在 Cassandra 里，你只要加入新的计算机，Cassandra 就会自动地发现它并让它开始工作。

高可用和容错（High Availability and Fault Tolerance）

从一般架构的角度来看，系统的可用性是由满足请求的能力来量度的。但计算机可能会有各种各样的故障，从硬件器件故障到网络中断都有可能。如何计算机都可能发生这些情况，所以它们一般都有硬件冗余，并在发生故障事件的情况下会自动响应并进行热切换。对一个需要高可用的系统，它必须由多台联网的计算机构成，并且运行于其上的软件也必须能够在集群条件下工作，有设备能够识别节点故障，并将发生故障的中端的功能在剩余系统上进行恢复。

Cassandra 就是高可用的。你可以在不中断系统的情况下替换故障节点，还可以把数据分布到多个数据中心里，从而提供更好的本地访问性能，并且在某一数据中心发生火灾、洪水等不可抗灾难的时候防止系统彻底瘫痪。

可调节的一致性（Tuneable Consistency）

2000年，加州大学伯克利分校的 Eric Brewer 在 ACM 分布式计算原理会议提出了著名的 CAP 定律。CAP 定律表明，对于任意给定的系统，只能在一致性（Consistency）、可用性（Availability）以及分区容错性（Partition Tolerance）之间选择两个。关于 CAP 定律的详细介绍可参见《分布式系统一致性问题、CAP定律以及 BASE 理论》以及《一篇文章搞清楚什么是分布式系统 CAP 定理》。所以 Cassandra 在设计的时候也不得不考虑这些问题，因为分区容错性这个是每个分布式系统必须考虑的，所以只能在一致性和可用性之间做选择，而 Cassandra 的应用场景更多的是为了满足可用性，所以我们只能牺牲一致性了。但是根据 BASE 理论，我们其实可以通过牺牲强一致性获得可用性。

Cassandra 提供了可调节的一致性，允许我们选定需要的一致性水平与可用性水平，在二者间找到平衡点。因为客户端可以控制在更新到达多少个副本之前，必须阻塞系统。这是通过设置副本因子（replication factor）来调节与之相对的一致性级别。

通过副本因子（replication factor），你可以决定准备牺牲多少性能来换取一致性。 副本因子是你要求更新在集群中传播到的节点数（注意，更新包括所有增加、删除和更新操作）。

客户端每次操作还必须设置一个一致性级别（consistency level）参数，这个参数决定了多少个副本写入成功才可以认定写操作是成功的，或者读取过程中读到多少个副本正确就可以认定是读成功的。这里 Cassandra 把决定一致性程度的权利留给了客户自己。

所以，如果需要的话，你可以设定一致性级别和副本因子相等，从而达到一个较高的一致性水平，不过这样就必须付出同步阻塞操作的代价，只有所有节点都被更新完成才能成功返回一次更新。而实际上，Cassandra 一般都不会这么来用，原因显而易见（这样就丧失了可用性目标，影响性能，而且这不是你选择 Cassandra 的初衷）。而如果一个客户端设置一致性级别低于副本因子的话，即使有节点宕机了，仍然可以写成功。

总体来说，Cassandra 更倾向于 CP，虽然它也可以通过调节一致性水平达到 AP；但是不推荐你这么设置。

面向行（Row-Oriented）

Cassandra 经常被看做是一种面向列（Column-Oriented）的数据库，这也并不算错。它的数据结构不是关系型的，而是一个多维稀疏哈希表。稀疏（Sparse）意味着任何一行都可能会有一列或者几列，但每行都不一定（像关系模型那样）和其他行有一样的列。每行都有一个唯一的键值，用于进行数据访问。所以，更确切地说，应该把 Cassandra 看做是一个有索引的、面向行的存储系统。

Cassandra 的数据存储结构基本可以看做是一个多维哈希表。这意味着你不必事先精确地决定你的具体数据结构或是你的记录应该包含哪些具体字段。这特别适合处于草创阶段，还在不断增加或修改服务特性的应用。而且也特别适合应用在敏捷开发项目中，不必进行长达数月的预先分析。对于使用 Cassandra 的应用，如果业务发生变化了，只需要在运行中增加或删除某些字段就行

了，不会造成服务中断。

当然， 这不是说你不需要考虑数据。相反，Cassandra 需要你换个角度看数据。在 RDBMS 里，你得首先设计一个完整的数据模型， 然后考虑查询方式， 而在 Cassandra 里，你可以首先思考如何查询数据，然后提供这些数据就可以了。

灵活的模式（Flexible Schema）

Cassandra 的早期版本支持无模式（schema-free）数据模型，可以动态定义新的列。无模式数据库（如 Bigtable 和 MongoDB）在访问大量数据时具有高度可扩展性和高性能的优势。无模式数据库的主要缺点是难以确定数据的含义和格式，这限制了执行复杂查询的能力。

为了解决这些问题，Cassandra 引入了 Cassandra Query Language（CQL），它提供了一种通过类似于结构化查询语言（SQL）的语法来定义模式。最初，CQL 是作为 Cassandra 的另一个接口，并且基于 Apache Thrift 项目提供无模式的接口。在这个过渡阶段，术语“模式可选”（Schema-optional）用于描述数据模型，我们可以使用 CQL 的模式来定义。并且可以通过 Thrift API 实现动态扩展以此添加新的列。在此期间，基础数据存储模型是基于 Bigtable 的。

从 3.0 版本开始，不推荐使用基于 Thrift API 的动态列创建的 API，并且 Cassandra 底层存储已经重新实现了，以更紧密地与 CQL 保持一致。 Cassandra 并没有完全限制动态扩展架构的能力，但它的工作方式却截然不同。 CQL 集合（比如 list、set、尤其是 map）提供了在无结构化的格式里面添加内容的能力，从而能扩展现有的模式。CQL 还提供了改变列的类型的能力，以支持 JSON 格式的文本的存储。

因此，描述 Cassandra 当前状态的最佳方式可能是它支持灵活的模式。

高性能（High Performance）

Cassandra 在设计之初就特别考虑了要充分利用多处理器和多核计算机的性能，并考虑在分布于多个数据中心的大量这类服务器上运行。它可以一致而且无缝地扩展到数百台机器，存储数 TB 的数据。Cassandra 已经显示出了高负载下的良好表现，在一个非常普通的工作站上，Cassandra 也可以提供非常高的写吞吐量。而如果你增加更多的服务器，你还可以继续保持 Cassandra 所有的特性而无需牺牲性能。

Cassandra 的应用场景

我们已经介绍了 Cassandra 的主要特点，对 Cassandra 的长处有了一定的理解。尽管 Cassandra 设计精巧，功能出色，但也不能胜任所有的工作。所以我们来介绍一下 Cassandra 最适合的场景。

大规模部署

你可能不会开着一辆轻型的小卡车去取干洗的衣服，小卡车显然不适合这种工作。Cassandra 的很多精巧设计都专注于高可用、可调一致性、P2P 协议、无缝扩展等，这些都是 Cassandra 的卖点。这些特性在单节点工作时都是没有意义的，更无法实现它的全部能力。

但是，单节点关系数据库在很多情况下可能正是我们需要的。所以你需要做一些评估。考虑你的期望的流量、吞吐需求以及 SAL 等。关于评估没有什么硬性的指标和要求。但如果你认为有几种关系型数据库可以很好地应付你的流量，提供不错的性能，那可能选关系型数据库更好。简单地说，这是因为 RDBMS 更易于在单机上运行，对你来说也更熟悉。

但是，如果你认为需要至少几个节点才能支撑你的业务，那 Cassandra 就是个不错的选择。如果你的应用可能需要数十个节点，那 Cassandra 可能就是个很棒的选择了。

写密集、统计和分析型工作

考虑一下你的应用的读写比例，Cassandra 是为优异的写吞吐量而特别优化的。

许多早期使用 Cassandra 的产品都用于存储用户状态更新、社交网络、建议/评价以及应用统计等。这些都是 Cassandra 很好的应用场景，因为这些应用大都是写多于读的，并且更新可能随时发生并伴有突发的峰值。事实上，支撑应用负载需要很高的多客户线程并发写性能，这正是 Cassandra 的主要特性。

根据项目的 wiki，Cassandra 已经被用于开发了多种不同的应用，包括窗口化的时间序列数据库，用于文档搜索的反向索引，以及分布式任务优先级队列。

地区分布

Cassandra 直接支持多地分布的数据存储，Cassandra 可以很容易配置成将数据分布到多个数据中心的存储方式。如果你有一个全球部署的应用，那么让数据贴近用户会获得不错的性能收益，Cassandra 正适合这种应用场合。

变化的应用

如果你正在“初创阶段”，业务会不断改进，Cassandra 这种灵活的模式的数据模型可能更适合你。这让你的数据库能更快地跟上业务改进的步伐。

谁在使用 Cassandra

Cassandra 在全世界有多达 1500 家公司使用：

- 苹果的 Cassandra 集群达到 75,000 节点，存储了 10PB 的数据；
- Netflix 的 Cassandra 集群达到 2,500 个节点，存储了多达 420TB 的数据；
- 宜搜的 Cassandra 集群达到 270 个节点，存储多达 300TB 的数据；
- eBay 的 Cassandra 集群达到 100 个节点，存储多达 250TB 的数据；
- 360 的 Cassandra 集群达到 1500 个节点；

- 饿了么的 Cassandra 集群达到 100 个节点。