Summary for three papers

Bigtable: A Distributed Storage System for Structured Data

Bigtable是用于管理结构化数据的分布式存储系统，该系统旨在扩展到非常大的规模：数千个商用服务器中的PB级数据。 Google的许多项目都将数据存储在Bigtable中，包括Web索引，Google Earth和Google Financial。这些应用程序在数据大小（从URL到网页到卫星图像）和延迟要求（从后端批量处理到实时数据服务）方面对Bigtable提出了截然不同的要求。尽管需求千差万别，Bigtable已成功为所有这些Google产品提供了一种灵活的高性能解决方案。在本文中，我们描述了Bigtable提供的简单数据模型，该模型为客户提供了对数据布局和格式的动态控制，并描述了Bigtable的设计和实现。
我们已经介绍了Bigtable，这是一个用于在Google存储结构化数据的分布式系统。自2005年4月以来，Bigtable集群已投入生产使用，在此日期之前，我们在设计和实施上花费了大约7个人年的时间。截至2006年8月，超过60个项目正在使用Bigtable。我们的用户喜欢Bigtable实现所提供的性能和高可用性，他们可以通过随资源需求的变化而向系统中添加更多机器来扩展集群的容量。
鉴于Bigtable具有非同寻常的界面，一个有趣的问题是我们的用户适应使用它有多困难。新用户有时不确定如何最好地使用Bigtable接口，特别是如果他们习惯于使用支持通用事务的关系数据库时。但是，许多Google产品成功使用Bigtable的事实表明我们的设计在实践中效果很好。
我们正在实现几个其他的Bigtable功能，例如对二级索引的支持以及用于构建具有多个主副本的跨数据中心复制Bigtable的基础结构。我们也已经开始将Bigtable作为服务部署到产品组，这样单个组就不需要维护自己的集群。随着服务集群规模的扩大，我们将需要在Bigtable自身内部处理更多的资源共享问题。
最后，我们发现在Google建立自己的存储解决方案有很多优势。通过为Bigtable设计我们自己的数据模型，我们获得了很大的灵活性。此外，我们对Bigtable实施的控制以及Bigtable依赖的其他Google基础架构意味着我们可以消除瓶颈和效率低下的情况。

MapReduce: Simplified Data Processing on Large Clusters

MapReduce是用于处理和生成大型数据集的编程模型和相关的实现。用户指定一个处理键/值对以生成一组中间键/值对的映射函数，以及一个reduce函数，该函数合并与同一中间键关联的所有中间值。如该论文所示，许多现实世界的任务在这种模型中都是可以表达的。
以这种功能风格编写的程序会自动并行化，并在大型商品计算机集群上执行。运行时系统负责划分输入数据，安排程序在一组机器上的执行，处理机器故障以及管理所需的机器间通信的细节。这使没有并行和分布式系统经验的程序员可以轻松利用大型分布式系统的资源。
我们对MapReduce的实现可在大型商用机器集群上运行，并且具有高度可扩展性：典型的MapReduce计算可在数千台机器上处理许多TB的数据。程序员发现该系统易于使用：每天实施数百个MapReduce程序，每天在Google集群上执行多达一千个MapReduce作业。
MapReduce编程模型已在Google上成功用于许多不同目的。我们将此成功归因于几个原因。首先，该模型易于使用，即使对于没有并行和分布式系统经验的程序员也是如此，因为该模型隐藏了并行化，容错，局部性优化和负载平衡的细节。其次，各种各样的问题很容易表达为MapReduce计算。例如，MapReduce用于为Google的生产网络搜索服务生成数据，进行分类，进行数据挖掘，用于机器学习以及许多其他系统。第三，我们开发了MapReduce的实现，该实现可扩展到包含数千台机器的大型机器集群。该实现有效利用了这些机器资源，因此适合用于Google遇到的许多大型计算问题。
我们从这项工作中学到了几件事。首先，限制编程模型可以简化计算的并行化和分布，并使此类计算具有容错性。其次，网络带宽是一种稀缺资源。因此，我们系统中的许多优化旨在减少通过网络发送的数据量：位置优化使我们能够从本地磁盘读取数据，并将中间数据的单个副本写入本地磁盘可节省网络带宽。第三，可以使用冗余执行来减少慢速计算机的影响，并处理机器故障和数据丢失。

The Google File System

我们已经设计并实现了Google File System，这是一个可扩展的分布式文件系统，用于大型分布式数据密集型应用程序。它可以在廉价的常用硬件上运行时提供容错功能，并且可以为大量客户端提供较高的聚合性能。
在实现与以前的分布式文件系统相同的许多目标的同时，我们的设计是由对我们当前和预期的应用程序工作负载和技术环境的观察推动的，这反映出与某些早期文件系统假设的明显偏离。这使我们重新审视了传统选择，并探索了根本不同的设计要点。
文件系统已成功满足我们的存储需求。它已在Google内广泛部署，作为存储平台，用于生成和处理供我们服务使用的数据以及需要大量数据集的研发工作。迄今为止，最大的群集可在一千台计算机上的数千个磁盘上提供数百TB的存储量，并且数百个客户端可以同时访问它。
在本文中，我们介绍了旨在支持分布式应用程序的文件系统接口扩展，讨论了我们设计的许多方面，并报告了来自微基准测试和实际使用情况的测量结果。
Google文件系统展示了在商用硬件上支持大规模数据处理工作负载所必需的质量。尽管某些设计决策是特定于我们独特的设置的，但许多决策可能适用于规模和成本意识相似的数据处理任务。
我们首先根据当前和预期的应用程序工作负载和技术环境，重新检查传统的文件系统假设。我们的观察结果导致了设计领域的根本不同。我们将组件故障视为正常现象，而不是例外情况，针对通常附加到（也许同时）然后读取（通常是顺序）的大型文件进行优化，并扩展和放松标准文件系统接口，以改善整个系统。
我们的系统通过不断监控，复制关键数据以及快速自动恢复功能来提供容错能力。块复制使我们能够容忍chunkserver故障。这些故障的发生率激发了一种新颖的在线维修机制，该机制定期透明地修复损坏并尽快补偿丢失的复制品。此外，我们使用校验和来检测磁盘或IDE子系统级别的数据损坏，鉴于系统中的磁盘数量，这种情况变得非常普遍。
我们的设计为执行各种任务的许多并发读写器提供了高的总吞吐量。我们通过将通过主机的文件系统控制与直接在块服务器和客户端之间传递的数据传输分开来实现此目的。大型块的大小和块的租用将主操作对常见操作的影响降到最低，这将权限授权给数据突变中的主副本。这样就可以成为一个简单的，集中的主服务器，而不会成为瓶颈。我们认为，网络堆栈的改进将解除当前对单个客户端看到的写入吞吐量的限制。
GFS已成功满足了我们的存储需求，并已在Google内部广泛用作研究，开发以及生产数据处理的存储平台。它是使我们能够继续创新并解决整个Web规模问题的重要工具。