论文背景综述报告

每个数据中心花费数十亿美元来构建，填充，并运作。批处理工作负载可以有效地收集服务剩余的备用周期和存储空间。然而，该服务必须被屏蔽从任何不平凡的性能干扰，它由批处理工作负载或其存储访问产生。之前的工作[1, 2, 3]已经解决了这个问题。同时，要确保服务的资源要求和管理不会不必要地降级批量工作负载的性能或妥协批处理工作负载数据的可用性和持久性。张等人[3]为批处理工作负载数据建立了一个资源收集型分布式文件系统，它通过在服务器上巧妙地放置文件块副本来实现这些特征。但是，他们没有解决如何在整个数据中心（例如，50k个服务器）范围收集扩展文件系统。

大部分分布式文件系统还没有被设计成扩展到（透明的或根本）到如此大的规模。例如，当超过几千台服务器时，HDFS [4]，Cosmos Store[5]和GFS [6]通常不能很好地扩展，因为他们依靠一个中心化的元数据管理器（带备用副本）。为了扩展超过这个规模，管理员必须创建单独的子集群（无论最大规模是多少都可以高效地进行处理），为一个命名空间的独立部分每个子集群运行一个独立的元数据管理器。

糟糕的是，这种扩展方法有好几个缺点。首先，给用户呈现一个命名空间的分区视图，并且用户经常完全控制使用哪个子集群来放置他们的文件夹/文件。第二，利用这个控制，用户可能会无意中填满一个子集群或者用过高的访问需求使一个子集群超过负载。第三，为了缓解这些情况，管理员必须手动管理文件夹/文件放置（通过文件夹/文件迁移和/或强制用户更轻度使用子集群）。第四，管理员了解每个子集群中的同位置服务的特征足够好，从而做出适当的文件夹/文件放置决定，是很困难的，（对用户来说是不可能的）特别是因为服务是复杂的和众多的。

另一种扩展的方法是实现多个，高度一致的活跃元数据管理器，例如 [7，8]。Google ‘s Colossus（GFS的后续）实现了这样的元数据管理器。然而，这种方法也有两个关键的缺点。第一，系统变得更加复杂，而这种复杂性只是大型装置所需（更简单的系统对于更流行的小型系统也工作的很好是一个更好的选择）。第二，当子集群没有提供隔离时，任何软件错误，故障或操作员犯错会产生更大的影响，如[9]中所强调的。

HDFS有几个优点，第一，它是一个在大型互联网公司中使用的流行的开源系统，例如微软，Twitter和雅虎。第二，当目标工作负载大多数是大数据量的分析工作，HDFS提供了足够的功能并表现良好。第三，许多数据分析框架，像Spark和Hive，可以在HDFS上运行。

收集中的一个挑战是保护文件块的可用性和耐用性。第一，如果我们存储所有块的副本在主力租户，同时主力租户负荷飙升的情况下，该块可能变得不可用。第二，如果开发者或管理系统在一个短的时间跨度内重新映像包含全部块的副本的磁盘，该块可能丢失。张的放置算法[3]放置副本在一个集群（即几千台服务器）内，同时最大限度地多样化，它不允许放置一个块的多个副本在任何逻辑（例如，同一主租户的服务器）或物理（例如，机架）服务器分组上，它引起资源使用情况相关性，磁盘重新映像或故障。

对于大的装置几个分布式文件系统（例如[7,10,8]）已被提出。尽管存在潜在的可扩展性，他们涉及元数据管理的复杂性和开销，以及在大规模生产中难以管理和维护。而且，它们通常是针对一般的工作负载进行优化，而不是那些数据中心应用程序（例如，一次写入，仅追加）。

由于这些原因，Microsoft [4, 5], Facebook [4，11], Twitter [4]和其他数据中心运营商的文件系统简单得多。他们依靠一个集中的元数据管理器（例如HDFS中的“Namenode”），它持有所有元数据，处理所有的元数据访问，并跟踪存储节点。为了扩展，管理员手动分区整个文件集变成独立的文件系统，每个在一个子集群中。一些系统（例如，ViewFS [12]，Cosmos store[5]）通过利用“装载表”，它翻译文件夹名称到子集群，允许用户透明地访问多个子集群。但是，客户端本地安装表是独立的，不保存相干。相比之下，谷歌的Colossus据传是实现多个活动元数据管理器。这种方法是更复杂的，不受益于由子集群提供的故障和错误隔离[9]。有些系统会在没有子集群的元数据管理器中重新平衡元数据，例如 [7，8]。

子集群面临的挑战是它们在空间使用和/或访问加载方面可能变得失衡。 在资源收集数据中心，不平衡在主要租户的资源使用和管理行为方面也可能存在。例如主租户磁盘重新映像模式的变化可能开始损害一个子群集的耐久性。

已经有几个考虑在单个子集群再平衡的工作。例如，[13]和[14]提出了平衡访问负载。考虑到空间和访问负载，Singh等人 [15]在进行贪婪的重新平衡决定中解释了多个资源（例如，交换机，磁盘）。内置的HDFS重新平衡器可以用来手动重新平衡数据，并填充新添加的服务器。HDFS有一个选项来分割命名空间（和块管理）显式跨越独立的元数据管理器，同时存储数据在任何服务器中[16]。这种方法不涉及子集群，但公开了用户必须手动管理的多个名称空间[12]，并由于所有服务器仍然发送心跳信息给所有的元数据管理器而限制扩展。

处理大量数据的应用程序（如搜索引擎，网格计算应用程序，数据挖掘应用程序等）需要一个后端存储数据的基础设施。分布式文件系统是存储数据基础设施的核心组件。有很多重点关于网络计算的项目，它们已经设计并实现了各种架构和功能的分布式文件系统。分布式文件系统（DFS）用于构建一个多个文件服务器的分层视图，并且可以在在网络上共享。用户将仅仅需要记住一个名字，而不必为每组文件考虑一个具体的机器名称。这对于在网络上的多个服务器中找到一个共享列表将是关键。DFS是一个支持共享的文件系统，文件以持久存储的形式存在于一组网络连接节点[17]。很多DFS发展了很多年，以及近二十年的研究还没有成功地产生一个全面特征的DFS [18，19，20]。多个用户共享使用一个通用文件系统，它们的物理位置分散在一个自治计算机网络中。一个查看这样一个系统有用的方法就是将其视为一个分时文件系统抽象的分布式实现。该挑战在于以一种有效安全可靠的方式来实现这种抽象。另外还有文件位置和可用性的问题也很重要。增加DFS内文件可用性的一个方法是通过使用文件副本。大部分的复制技术可以分为两大类如乐观的复制和悲观复制[21]。DFS性能的另一个主要瓶颈是处理器速度的显着改进。为了克服这个限制，DFS在不同的位置使用缓存[22]，在文件服务器或客户端可以定位这些缓存[23]。为了提供一个DFS中所有客户端都能看到的一致性数据视图并在故障情况下的可靠性，写入操作只有在数据提交稳定存储后才能完成。所以在文件服务器上占主导地位的负载是写入操作。从而允许从客户端写回可以减少这种服务器上的写入负载[22,24]。

现在在DFS中能够轻松使用商品设备并且扩展规模比较省钱是非常重要的，因为大规模分布式应用的需求。这包括增量扩展，这是以增量方式增加更多设备来扩大系统的能力。

[1] LO, D., CHENG, L., GOVINDARAJU, R., RANGANATHAN, P., AND KOZYRAKIS, C. Heracles: Improving Resource Efficiency at Scale. In ISCA(2015).

[2] MARS, J., TANG, L., HUNDT, R., SKADRON, K., AND SOFFA, M. L. Bubble-up: Increasing Utilization in Modern Warehouse Scale Computers Via Sensible Co-Locations. In MICRO (2011).

[3] ZHANG, Y., PREKAS, G., FUMAROLA, G. M., FONTOURA, M., GOIRI, I., AND BIANCHINI, R. History-Based Harvesting of Spare Cycles and Storage in Large-Scale Datacenters. In OSDI(2016).

[4] APACHE FOUNDATION. HDFS Architecture Guide, 2008. http://hadoop.apache.org/

docs/current/hdfs\_design.html.

[5] CHAIKEN, R., JENKINS, B., LARSON, P. °A., RAMSEY, B., SHAKIB, D., WEAVER, S., AND ZHOU, J. SCOPE: Easy and Efficient Parallel Processing of Massive Data Sets. In VLDB (2008).

[6] GHEMAWAT, S., GOBIOFF, H., AND LEUNG, S. T. The Google File System. In SOSP (2003).

[7] ADYA, A., BOLOSKY, W. J., CASTRO, M., CERMAK, G., CHAIKEN, R., DOUCEUR, J. R., HOWELL, J., LORCH, J. R., THEIMER, M., AND WATTENHOFER, R. P. FARSITE: Federated, Available, and Reliable Storage for an Incompletely Trusted Environment. In OSDI (2002).

[8] WEIL, S. A., BRANDT, S. A., MILLER, E. L., LONG, D. D., AND MALTZAHN, C. Ceph: A

Scalable, High-Performance Distributed File System. In OSDI (2006).

[9] VERMA, A., PEDROSA, L., KORUPOLU, M., OPPENHEIMER, D., TUNE, E., AND WILKES, J. Large-Scale Cluster Management at Google with Borg. In EuroSys (2015).

[10] MUTHITACHAROEN, A., MORRIS, R., GIL, T. M., AND CHEN, B. Ivy: A Read/Write Peerto-Peer File System. In OSDI (2002).

[11] BEAVER, D., KUMAR, S., LI, H. C., SOBEL, J., VAJGEL, P., ET AL. Finding a Needle in Haystack: Facebook’s Photo Storage. In OSDI (2010).

[12] APACHE FOUNDATION. ViewFs Guide, 2016. http://hadoop.apache.org/

docs/current/hadoop-project-dist/ hadoop-hdfs/ViewFs.html.

[13] HILDRUM, K., DOUGLIS, F., WOLF, J. L., YU, P. S., FLEISCHER, L., AND KATTA, A. Storage Optimization for Large-Scale Distributed Stream-Processing Systems. Transactions on Storage(TOS) 3, 4 (2008), 5.

[14] HSIAO, H. C., CHUNG, H. Y., SHEN, H., AND CHAO, Y. C. Load Rebalancing for Distributed File Systems in Clouds. TPDS 24, 5 (2013), 951–962.

[15] SINGH, A., KORUPOLU, M., AND MOHAPATRA, D. Server-Storage Virtualization: Integration an Load Balancing in Data Centers. In Supercomputing(2008)

[16] FOUNDATION, A. HDFS Federation, 2016. https://hadoop.apache.org/docs/

r2.7.2/hadoop-project-dist/hadoop-hdfs/Federation.html.

[17] Eliezer levy and Abraham silberschatz, "Distributed File Systems: Concepts and Examples", ACM Computing Surveys, Vol. 22, No. 4, December 1990..

[18] Chandramohan A. Thekkath, et al, "Frangipani: A scalable Distributed File System", System Research Center, Digital Equipment Corporation, Palo Alto, CA, 1997.

[19] Barbara Liskov, et al, "Replication in the Harp File System", Laboratory of Computer Science, MIT, Cambridge, CA, 1991.

[20] John Douceur and Roger Wattenhofer, "Optimizing file availability in a server-less distributed file system" In Proceedings of the 20th Symposium on Reliable Distributed Systems, 2001.

[21]Yasushi Saito and Marc Shapiro, "Optimistic Replication", ACM Computing Surveys, Vol. 37, No. 1, March 2005, pp. 42-81.

[22] Howard, J.H., et al, "Scale and Performance in a Distributed File System," ACM Transactions on Computer Systems, Vol. 6, Issue 1, February 1988.

[23] Satyanarayanan, M., "A Survey of Distributed File Systems," Technical Report CMU-CS-89- 116, Department of Computer Science, Camegie Mellon University, 1989

[24] Nelson, M.N., et al. "Caching in the Sprite Network File System," ACM Transactions on Computer Systems, February, 1988