PayPal高级工程总监：读完这100篇论文 就能成大数据高手

投递人 itwriter 发布于 2015-07-07 13:30 评论(16) 有1272人阅读 原文链接 [收藏] « »

英文原文：100 open source Big Data architecture papers for data professionals.

PayPal 高级工程总监 Anil Madan 写了篇大数据的文章，近日 CSDN 对此进行了翻译。一共有 100 篇数据的论文，涵盖大数据技术栈，全部读懂你将会是大数据的顶级高手。

开源（Open Source）用之于大数据技术，其作用有二：一方面，在大数据技术变革之路上，开源在众人之力和众人之智推动下，摧枯拉朽，吐故纳新，扮演着非常重要的推动作用。另一方面，开源也给大数据技术构建了一个异常复杂的生态系统。每一天，都有一大堆"新"框架、"新"类库或"新"工具，犹如雨后春笋般涌出，乱花渐欲"迷"人眼。为了掌控住这些"新玩意"，数据分析的达人们不得不"殚精竭虑"地"学而时习之"。

无论你是一个大数据的布道者，还是一个日臻成熟的技术派，亦或你还在大数据这条路上"小河才露尖尖角"，多花点时间，深入理解一下大数据系统的技术体系演进，对你都会有莫大益处。全方位地理解大数据体系结构中的各个组件，并掌握它们之间的微妙差别，可在处理自己身边的大数据案例时，助你张弛有度，"恢恢乎，其于游刃必有余地矣!"

在过去的几年里，我阅读了很多不错的大数据文献，这些文献陪我成长，助我成功，使我成为一个具备良好教育背景的大数据专业人士。在这里，撰写此文的目的，不限于仅仅和大家分享这些很不错的文献，更重要的是，借此机会，想和大家一起，集众人之智慧，破解大数据开源系统之迷宫。

需要提醒的是，下文提及到的 100 篇参考文献（这些文献中大多都是一些开创性的研究论文），将会为你提供结构性的深度剖析，绝非泛泛而谈。我相信，这可从根本上帮助你深度理解大数据体系组件间的细微差别。但如果你打算"走马观花"般地快速过一遍，了解大数据为何物，对不起，这里可能会让你失望。

那么，准备好了吗？让我们走起！

在介绍这 100 篇文献之前，首先让我们看一下大数据处理的关键架构层（如图 1 所示）：

关键架构层

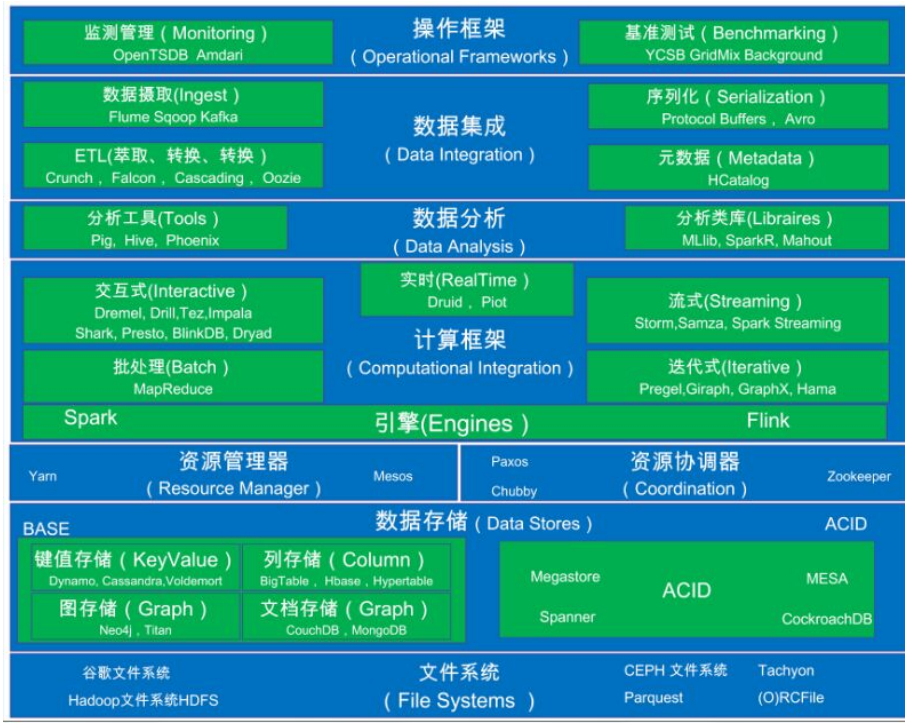

图1：大数据处理的关键架构层

- 文件系统层：在这一层里，分布式文件系统需具备存储管理、容错处理、高可扩展性、高可靠性和高可用性等特性。
- 数据存储层：由于目前采集到的数据，十之有七八为非结构化和半结构化数据，数据的表现形式各异，有文本的、图像的、音频的、视频的等，因此常见的数据存储也要对应有多种形式，有基于键值（Key-Value）的，有基于文档（Document），还有基于列（Column）和图表（Graph）的。如果采用单一的数据库引擎，“一刀切式”的满足所有类型的数据存储需求，通常会严重降低数据库管理的性能。因此，我们需要“兵来将挡，水来土掩”式的、多元的（Poly glot）【1】数据库解决方案（这就好比，如果“兵来了”和“水来了”，都要“将”去挡，遇到“兵”时，“将”可以“酣畅淋漓”，而遇到“水”时，还用“将”去挡，那这个“将”估计就要“舍生取义”了。文献【1】是一本有关 NoSQL 数据处理的图书）
- 资源管理层：这一层是为了提高资源的高利用率和吞吐量，以到达高效的资源管理与调度目的。
- 资源协调层： 在本层的系统，需要完成对资源的状态、分布式协调、一致性和资源锁实施管理。
- 计算框架层：在本层的计算框架非常庞杂，有很多高度专用的框架包含其内，有流式的，交互式的，实时的，批处理和迭代图的（Batch and Iterative Graph，BSP）等。为这些计算框架提供支撑的是运行时引擎，如 BDAS【2】(Spark) 和 Flink 等（注：这里的 BDAS 是指“Berkeley Data Analytics Stack”，即伯克利数据分析栈。文献【2】为 Spark 核心作者 Ion Stoica 的讲座幻灯片文档）。
- 数据分析层：在这一层里，主要包括数据分析(消费)工具和一些数据处理函数库。这些工具和函数库，可提供描述性的、预测性的或统计性的数据分析功能及机器学习模块。
- 数据集成层：在这一层里，不仅包括管理数据分析工作流中用到的各种适用工具，除此之外，还包括对元数据（Metadata）管理的工具。
- 操作框架层：这一层提供可扩展的性能监测管理和基准测试框架。

架构的演进

减少数据生产者和消费者之间的处理延迟，一直是现代计算构架不断演进的主要动力。由此，诞生了实时和低延迟处理的计算构架，如 Lambda 和 Kappa 等，这类混合架构取长补短，架起传统的批处理层和交互式层之间连接的桥梁。

- Lambda【3】-该架构是经典的大数据处理范式，是由南森马兹（Nathan Marz）提出的一个实时大数据处理框架。更多有关 Lamda 的信息，请读者访问 Lambda 官方网站。（注：文献【3】是由 James Kinley 在轻博客网站 Tumblr 发表的一篇博文：Lambda 架构：构架实时大数据系统的原则）。
- Kappa【4】-该计算构架可视为 Lambda 的一个强有力替代者，Kappa 将数据处理的上游移至流式层（注：文献【4】是一篇博客文章，作者是 Jay Kreps 是 Linkedln 的一名在线数据

Paypal：若不是跑得太快，还能走更远的路
大数据变现的4个新创意
PayPal“再出发”：要用支付连接一切
PayPal满足中国“海淘者”支付需求 推出新服务
PayPal从eBay拆分出来后将在纳斯达克上市

架构技术高管。Kreps 认为，虽然 Lambda 构架的理念很有价值，但终究还是一个临时解决方案。他设计了一个替代架构 Kappa，是基于他在 Linkedin 构建 Kafka 和 Samza 的经验设计而成）。

- SummingBird【5】-这是一个参考模型，用来桥接在线处理模式和传统处理模式。Summingbird 是由 Twitter（推特）公司用 Scala 语言开发的、并开源的大规模数据处理框架，支持开发者以批处理模式（基于 Hadoop）或流处理模式（基于 Storm），或混合模式（即前两种模式的组合）以统一的方式执行代码。（注：文献【5】是 Summingbird 的主要设计者 Oscar Boykin、Sam Ritchie 等人于 2014 年发表于知名期刊 PVLDB 中论文，其中论文的二作 Sam Ritchie 大有来头，他是计算机科学界的传奇人物、C语言和 Unix 的设计者 Dennis Ritchie 的侄子）。

在你尚未深入了解下面的各个具体的框架层次之前，建议你认真阅读一下下面的几篇非常有价值的文献，它们帮为你"恶补"一下诸如 NoSQL（非结构化）数据存储、数据仓库大规模计算及分布式系统等相关领域的背景知识：

- 计算中心即计算机【6】（Data center as a computer）-文献【6】是威斯康星大学-麦迪逊分校 Mark D. Hill 教授主编的一个论文集式的图书，在这本图书中，收集了很多有关数据仓库大规模计算的论文（注：将数据中心视为一台计算机，与传统的高性能计算机有很大不同。计算中心的实例将以虚拟机或者容器的形式存在，计算资源的配置对于用户而言是透明的，这样就大幅降低系统部署的复杂度、并提高资源使用的灵活性）。
- 非结构化（NOSQL）数据存储【7】- 文献是由 Rick Cattell 撰写的论文，论文讨论了可扩展的结构化数据的、非结构化的（包括基于键值对的、基于文档的和面向列的）数据存储方案（注：NOSQL 是支撑大数据应用的关键所在。事实上，将 NOSQL 翻译为"非结构化"不甚准确，因为 NOSQL 更为常见的解释是：Not Only SQL（不仅仅是结构化），换句话说，NOSQL 并不是站在结构化 SQL 的对立面，而是既可包括结构化数据，也可包括非结构化数据）。
- NoSQL 学位论文【8】-该文献是德国斯图加特传媒大学 Christof Strauch 撰写的学位论文，该论文对分布式系统和第一代非结构化系统提供了非常系统的背景知识介绍。
- 大规模数据管理【9】-文献是加拿大阿尔伯塔大学的研究人员撰写的一篇综述，讨论了大数据应用程序的大规模数据管理系统，传统的数据库供应商与新兴的互联网企业，它们对大数据管理需求是不同的。文章的讨论范围涵盖很广，数据模型、系统结构及一致性模型，皆有涉及。
- 最终一致性（Eventual Consistency）【10】：论文讨论了分布式系统中的各种不同的一致性模型。（注：原文给出的链接可能有误，因为根据所提供的链接下载而来的论文是关于"MapReduce 中日志处理的 Join 算法"的综述文章，与"最终一致性"的讨论议题无关。这里推荐 2 篇新的相关论文：（1）综述文章：数据库最终一致性：最新的进展【10】new1；（2）微软研究人员 2013 年发表于 SIGMOD 的文章："最终一致性的反思（Rethinking Eventual Consistency）【10】new2"。）
- CAP 理论【11】-文献以"CAP 理论十二年回顾："规则"已经变了"为题，探讨了 CAP 理论及其演化，是篇非常不错的介绍 CAP 理论的基础性论文（注：论文作者 Eric Brewer 是加州大学伯克利分校的知名计算机科学学者。该文首发于《Computer》杂志，随后又被 InfoQ 和 IEEE 再次发表。CAP 理论断言，任何基于网络的数据共享系统，最多只能满足数据一致性（Consistency，C）、可用性（Availability，A）、分区（Partition，P）容忍性这三要素中的两个要素。但通过显式处理分区，系统设计师可做到优化数据的一致性和可用性，进而取得三者之间的妥协与平衡）。

在过去，在大规模数据处理上，传统的并行数据库管理系统（DBMS）和基于 Map Reduce（映射-规约，以下简称 MR）的批处理范式之间，曾发生激烈辩论，各持己见。并行数据库管理系统的支持者【12】（注：由耶鲁大学、微软和麻省理工学院的研究人员于 2009 年发表在 SIGMOD 的一篇文章）和另外一篇文献【13】（注：2010 年发表于《美国计算机学会通讯》上的论文："MapReduce 和并行数据库管理系统，是朋友还是敌人？"），被 MR 的拥趸者【14】（注：发表于美国计算机学会通讯的论文：MapReduce:一个弹性的数据处理工具）狠狠地给批驳了一番。

然而，令人讽刺的是，从那时起，Hadoop 社区开始引入无共享的（Shared-Nothing）的 MPP（大规模并行处理）风格的大数据处理模式，文献"Hadoop 上的 SQL【15】"，便是例证。要知道，MPP 是并行数据库管理系统（DBMS）的灵魂，这样，Map Reduce 绕了一大圈，又似回到它当初离开的地方。

文件系统层

由于文件系统层关注的焦点，开始向"低延时处理"方向转移，所以传统基于磁盘存储的文件系

统，也开始向基于内存计算的文件系统转变 —— 这样做，会大大降低 I / O 操作和磁盘序列化带来的访问开销。Tachyon 和 Spark RDD【16】就是朝这个方向演化的范例（注：这里 RDD 指的是弹性分布式数据集（Resilient Distributed Datasets），它是一种高度受限的共享内存模型，文献【16】由伯克利大学加州分校的 Matei Zaharia 等撰写的，他们提出了一种面向内存集群运算的容错抽象模型）。

- Google 文件系统（GFS）【17】-该文献是分布式文件系统的奠基之作，著名的 Hadoop 分布式文件系统（HDFS），亦脱胎于 GFS，基本上可视为 GFS 的一个简化实现版（注：文献【17】提出了一个可扩展的分布式文件系统 GFS，可用于大型分布式数据密集型应用。文献认为，组件故障是常态而不是异常。其所提出的 GFS，着眼在几个重要的目标，比如性能、可伸缩性、可靠性和可用性。GFS 的新颖之处，并不在于它采用了多么令人惊艳的技术，而在于它能利用所提出的方案，采用廉价的商用机器，来构建高效的分布式文件系统。有用的创新，才是真的创新，GFS 做到了！）。
- Hadoop 文件系统【18】-该文献由雅虎公司的计算机科学家 Konstantin Shvachko 等人联合撰写的，论文给出了 HDFS 的进化历史背景及其架构的设计内涵，是了解 Hadoop 技术的经典之作。
- Ceph 文件系统【19】-Ceph 是 HDFS 有力的替代者【20】（注：Ceph 文件系统是加州大学圣克鲁兹分校（USSC）博士生 Sage Weil 博士期间的一项有关存储系统的研究项目。初出茅庐，略有小成。之后，在开源社区的推动下，Ceph 逐渐羽翼渐丰，风云叱咤，功成名就，逐渐发展成为一个 Linux 系统下 PB 级分布式文件系统。文献【19】是 Weil 本人在 2006 年顶级会议 OSDI 发表的有关 Ceph 的开山论文。文献【20】则是 Weil 率领他的一帮小伙伴们再次发文强调，Ceph 是 HDFS 强有力的替代者）。
- Tachyon【21】–是一个高容错的分布式内存文件系统，其设计的核心内涵是，要满足当下"低延迟"的数据处理要求（注：Tachyon 是在内存中处理缓存文件，允许文件以访问内存的速度在集群框架中进行可靠的共享，类似于 Spark。Tachyon 的吞吐量比 HDFS 高出 100 倍。Spark 框架虽然也提供了强大的内存计算能力，但其没有提供内存文件的存储管理能力，而 Tachyon 则弥补了 Spark 的不足之处。文献【21】是伯克利大学加州分校和麻省理工学院的研究者联合撰写的，发表在 2014 年的 SoCC 国际会议上，论文一作 UC Berkeley AMP 实验室博士生李浩源，他亦是 Spark 核心开发人员之一）。

文件系统的演化历程，其实也见证了文件格式和压缩技术的发展历程。下面的参考文献，可以让你了解到，"面向行"或"面向列"存储格式各自的优缺点，并且还可让你了然文件存储技术发展的新趋势——嵌套式的面向列的存储格式，这种存储格式可极大提高大数据的处理效率。

当前，在文件系统阶段，数据管理的最大挑战之一就是，如何处理大数据中的数据冗余。纠删码（Erasure code）是很有创意的冗余保护机制，它可以减少三倍的冗余副本，还不会影响数据的可恢复性与可用性。

- 面向列存储 vs. 面向列存储【22】—该文献是是 2008 年发表于 SIGMOD 的一篇论文，该文对数据的布局、压缩及物化（materialization）策略都做了很不错的综述。
- RCFile【23】-这是由 Facebook 数据基础设施小组和俄亥俄州立大学的华人学者共同提出的文件存储格式，他们走了一个"中庸之道"，充分吸取面向列和面向行存储模式的优点，扬长避短，提出了一种混合的数据存储结构 PAX（注：目前这种以行/列混合存储技术已成功应用于 Facebook 等国内外大型互联网企业的生产性运行体系）。
- Parquet【24】- 这是一种面向行的存储格式，其设计理念源于谷歌 Dremel 论文（注：Parquet 主要用于 Hadoop 的生态系统中。文献【24】是 Julien Dem 在 Github 发表的一篇博客文章）。
- ORCFile【25】–这是一种被 Hive（一种基于 Hadoop 的数据仓库工具）采用的、面向列存储的改进版存储格式（注：文献【25】是 2014 年发表于顶会 SIGMOD 的一篇学术论文）。
- 压缩技术【26】-这是是一篇阐述在 Hadoop 生态系统下的常见压缩算法的综述性文章，文章对常见的压缩算法和其适用场景以及它们的优缺点，做了非常不错的归纳总结。
- 纠删码技术（Erasure code）【27】-这是一篇是田纳西大学 EECS 系教授 James Plank 撰写的、有关存储系统纠删码技术的入门级的文献。有关纠删码改进技术的阐述，读者可参阅来自南加州大学和 Facebook 的 7 名作者共同完成的论文《XORing Elephants: 面向大数据的新型纠删码技术【28】》（注：文献【28】的作者开发了纠删码家族的新成员——基于 XOR 的本地副本存储 LRC，该技术是面向 Hadoop 生态系统的，可显著减少修复数据时的 I/O操作和存储开销）。

数据存储层

宽泛地讲，据对一致性（consistency）要求的强弱不同，分布式数据存储策略，可分为 ACID 和 BASE 两大阵营。ACID 是指数据库事务具有的四个特性：原子性（Atomicity）、一致性（Consistency）、隔离性（Isolation）、持久性（Durability）。ACID 中的一致性要求比较强，事务执行的结果必须是使数据库从一个一致性状态变到另一个一致性状态。而 BASE 对一致性要求较弱，它的三个特征分别是：基本可用（Basically Available）， 软状态/柔性事务（Soft-state，即状态可以有一段时间的不同步）， 最终一致性（Eventual consistency）。BASE 还进一步细分基于键值的，基于文档的和基于列和图形的 – 细分的依据取决于底层架构和所支持的数据结构（注：BASE 完全不同于 ACID 模型，它以牺牲强一致性，获得基本可用性和柔性可靠性，并要求达到最终一致性）。

在数据存储层，还有很多类似的系统和某些系统的变种，这里，我仅仅列出较为出名的几个。如漏掉某些重要系统，还请谅解。

BASE

键值存储（**Key Value Stores**）

Dynamo【29】– 这是由亚马逊工程师们设计的基于键值的高可用的分布式存储系统（注：Dynamo 放弃了数据建模的能力，所有的数据对象采用最简单的 Key-value 模型存储，可简单地将 Dynamo 理解为一个巨大的 Map。Dynamo 是牺牲了部分一致性，来换取整个系统的高可用性）。

Cassandra【30】 – 这是由 Facebook 工程师设计的一个离散的分布式结构化存储系统，受亚马逊的 Dynamo 启发，Cassandra 采用的是面向多维的键值或面向列的数据存储格式（注：Cassandra 可用来管理分布在大量廉价服务器上的巨量结构化数据，并同时提供没有单点故障的高可用服务）。

Voldemort【31】 –这又是一个受亚马逊的 Dynamo 启发的分布式存储作品，由全球最大的职业社交网站 LinkedIn 的工程师们开发而成（注：Voldemort，这个在《哈利·波特》中常被译作“伏地魔”的开源数据库，支撑起了 LinkedIn 的多种数据分析平台）。

面向列的存储（**Column Oriented Stores**）

BigTable【32】 –这是一篇非常经典的学术论文，阐述了面向列的分布式的数据存储方案，由谷歌荣誉出品。（注：Bigtable 是一个基于 Google 文件系统的分布式数据存储系统，是为谷歌打拼天下的“三驾马车”之一，另外两驾马车分别是分布式锁服务系统 Chubby 和下文将提到的 MapReduce）。

HBase【33】 –目前还没有有关 Hbase 的定义性论文，这里的文献提供了一个有关 HBase 技术的概述性文档（注：Hbase 是一个分布式的、面向列的开源数据库。其设计理念源自谷歌的 BigTable，用 Java 语言编写而成。文献【33】是一个有关 Hbase 的幻灯片文档）。

Hypertable【34】-文献是一个有关“Hypertable”的技术白皮书，对该数据存储结构做了较为详细的介绍（注：Hypertable 也是一个开源、高性能、可伸缩的数据库，它采用与 Google 的 Bigtable 类似的模型）。

面向文档的存储（**Document Oriented Stores**）

CouchDB【35】– 这是一款面向文档的、开源数据存储管理系统（注：文献【35】是一本 Apache CouchDB 的 400 多页的官方文档）。

MongoDB【36】 –是目前非常流行的一种非关系型(NoSQL)数据库（注：文献【36】是一个有关 MongoDB 的白皮书，对 MongoDB 结构做了很不错的介绍）。

面向图（**Graph**）的存储

Neo4j【37】 –文献是 Ian Robinson 等撰写的图书《Graph Databases（图数据库）》（注：Neo4j 是一款目前最为流行的高性能 NoSQL 图数据库，它使用图来描述数据模型，把数据保存为图中的节点以及节点之间的关系。这是最流行的图数据库）。

Titan【38】 –文献是有关 Titan 的在线文档（Titan 是一款 Apache 许可证框架下的分布式的开源图数据库，特别为存储和处理大规模图而做了大量优化）。

ACID

我注意到，现在很多开源社区正在悄悄发生变化，它们开始“亦步亦趋”地跟随谷歌的脚步。这也难怪，谷歌太牛，跟牛人混，近牛者牛 —— 下面 4 篇文献，有 3 篇来自于谷歌的“神来之笔”，他们

解决了全球分布一致的数据存储问题。

Megastore【39】–这是一个构建于 BigTable 之上的、高可用的分布式存储系统，文献为有关 Megastore 的技术白皮书（注：Megastore 在被谷歌使用了数年之后，相关技术信息才在 2001 年公布。CSDN 网站亦有文献【39】的中文解读：Google Megastore 分布式存储技术全揭秘）。

Spanner【40】–这是由谷歌研发的、可扩展的、全球分布式的、同步复制数据库，支持 SQL 查询访问。（注：Spanner 的"老爹"是 Big Table，可以说，没有"大表"这个爹，就不可能有这个强有力的"扳手"儿子。它是第一个把数据分布在全球范围内的系统，并且支持外部一致性的分布式事务）。

MESA【41】–亦是由谷歌研发的、跨地域复制(geo-replicated)、高可用的、可容错的、可扩展的近实时数据仓库系统（注：在 2014 年的 VLDB 大会上，谷歌公布了他们的分析型数据仓库系统 MESA，该系统主要用于存储 Google 互联网广告业务相关的关键衡量数据。文献【41】是 VLDB 的会议论文）。

CockroachDB【42】–该系统是由 Google 前工程师 Spencer Kimball 领导开发的 Spanner 的开源版本（注：这个项目的绰号是"蟑螂（Cockroach）"，其寓意是"活得长久"，因为蟑螂是地球上生命力最强的生物之一，即使被砍下头颅，依然还能存活好几天！文献【42】是代码托管网站 GitHub 上对 Cockroach 的说明性文档）。

资源管理器层（**Resource Managers**）

第一代 Hadoop 的生态系统，其资源管理是以整体单一的调度器起家的，其代表作品为 YARN。而当前的调度器则是朝着分层调度的方向演进（Mesos 则是这个方向的代表作），这种分层的调度方式，可以管理不同类型的计算工作负载，从而可获取更高的资源利用率和调度效率。

YARN【43】– 这是新一代的 MapReduce 计算框架，简称 MRv2，它是在第一代 MapReduce 的基础上演变而来的（注：MRv2 的设计初衷是，为了解决第一代 Hadoop 系统扩展性差、不支持多计算框架等问题。对国内用户而言，原文献下载链接可能会产生 404 错误，这里提供一个新文献：由 2011 年剥离自雅虎的 Hadoop 初创公司 Hortonworks 给出的官方文献【43】new，阅读该文献也可对 YARN 有较为深入的理解。CSDN 亦有对 YARN 详细解读的文章：更快、更强——解析 Hadoop 新一代 MapReduce 框架 Yarn）。

Mesos【44】–这是一个开源的计算框架，可对多集群中的资源做弹性管理（注：Mesos 诞生于 UC Berkeley 的一个研究项目，现为 Apache 旗下的一个开源项目，它是一个全局资源调度器。目前 Twitter、 Apple 等国外大公司正在使用 Mesos 管理集群资源，国内用户有豆瓣等。文献【44】是加州大学伯克利分校的研究人员发表于著名会议 NSDI 上的学术论文）。

这些计算框架和调度器之间是松散耦合的，调度器的主要功能就是基于一定的调度策略和调度配置，完成作业调度，以达到工作负载均衡，使有限的资源有较高的利用率。

调度器（**Schedulers**）

作业调度器，通常以插件的方式加载于计算框架之上，常见的作业调度器有 4 种：

计算能力调度器【45】（Capacity Scheduler）-该文献是一个关于计算能力调度器的指南式文档，介绍了计算能力调度器的不同特性。

公平调度器【46】（FairShare Scheduler） -该文献是 Hadoop 的公平调度器设计文档，介绍了公平调度的各项特征（注：公平调度是一种赋予作业资源的方法，它提供了一个基于任务数的负载均衡机制，其目的是让所有的作业随着时间的推移，都能平均的获取等同的共享资源）。

延迟调度【47】（Delayed Scheduling） –该文献是加州大学伯克利分校的一份技术报告，报告介绍了公平调度器的延迟调度策略。

公平与能力调度器【48】（Fair & Capacity schedulers ）–该文献是一篇关于云环境下的 Hadoop 调度器的综述性论文。

协调器（**Coordination**）

在分布式数据系统中，协调器主要用于协调服务和进行状态管理。

Paxos【49】 –文献【49】是经典论文"The Part-Time Parliament（兼职的议会）【50】" 的简化版。

注：两篇文献的作者均是莱斯利·兰伯特（Leslie Lamport），此君是个传奇人物，科技论文写作常用编辑器 LaTex，其中"La"就是来自其姓"Lamport"的前两个字母。Lamport 目前是微软研究院

首席研究员，2013 年，因其在分布式计算理论领域做出的杰出贡献，荣获计算机领域最高奖——图灵奖。

牛人的故事特别多，Lamport 亦是这样。就这两篇文献而言，Lamport 的奇闻轶事都值得说道说道。光看其经典论文题目"The Part-Time Parliament（兼职的议会）【50】"，或许就让读者"一头雾水"，这是一篇计算机科学领域的论文吗？和读者一样感觉的可能还有期刊编辑。其实，早在 1990 年时，Lamport 就提出 Paxos 算法，他虚构了一个希腊城邦 Paxos 及其议会，以此来形象比喻说明该算法的流程。论文投出后，期刊编辑建议 Lamport，将论文用更加严谨的数学语言重新进行描述一下。可 Lamport 则认为，我的幽默，你不懂！拒绝修改。时隔八年之后的 1998 年，Paxos 算法才被伯乐期刊《ACM Transactions on Computer Systems》发表。由于 Paxos 算法本身过于复杂，且同行不理解自己的"幽默"， 于是，2001 年 Lamport 就用简易语言撰写这篇文章，重新发表了该论文的简化版【49】，即"Paxos made simple（Paxos 变得简单）"。简化版的摘要更简单，就一句话："Paxos 算法，用简易英语说明之，很简单"，如果去掉中间的那个无故紧要的定语从句，就是"Paxos 算法，很简单"。弄得你都来不及做深思状，摘要就完了。这…，这…，完全颠覆了我们常用的"三段论式（提问题、解问题、给结论）"的论文摘要写法啊。

后来，随着分布式系统的不断发展壮大，Paxos 算法开始大显神威。Google 的 Chubby 和 Apache 的 Zookeeper，都是用 Paxos 作为其理论基础实现的。就这样， Paxos 终于登上大雅之堂，它也为 Lamport 在 2013 年获得图灵奖，立下汗马功劳。从 Lamport 发表 Paxos 算法的小案例，我们可以看出：彪悍的人生，不需要解释。牛逼的论文，就可以任性！

Chubby【51】– 该文献的作者是谷歌工程师 Mike Burrows。Chubby 系统本质上就是前文提到的 Paxos 的一个实现版本，主要用于谷歌分布式锁服务。（注：原文链接会出现 404 错误，CSDN 网站有 Chubby 论文的下载链接）。

Zookeeper【52】 –这是 Apache Hadoop 框架下的 Chubby 开源版本。它不仅仅提供简单地上锁服务，而事实上，它还是一个通用的分布式协调器，其设计灵感来自谷歌的 Chubby（注：众所周知，分布式协调服务开发困难很大，分布式系统中的多进程间很容易发生条件竞争和死锁。ZooKeeper 的开发动力就是减轻分布式应用开发的困难，使用户不必从零开始构建协调服务）。

计算框架（**Computational Frameworks**）

运行时计算框架，可为不同种类的计算，提供运行时（runtime）环境。最常用的是运行时计算框架是 Spark 和 Flink。

Spark【53】 –因 Spark 日益普及，加之其具备良好的多计算环境的适用性，它已对传统的 Hadoop 生态环境，形成了严峻的挑战（注：Spark 是一个基于内存计算的开源的集群计算系统，其目的在于，让数据分析更加快速。Spark 是由加州大学伯克利分校的 AMP 实验室采用 Scala 语言开发而成。Spark 的内存计算框架，适合各种迭代算法和交互式数据分析，能够提升大数据处理的实时性和准确性，现已逐渐获得很多企业的支持，如阿里巴巴、百度、网易、英特尔等公司均是其用户）。

Flink【54】 –这是一个非常类似于 Spark 的计算框架，但在迭代式数据处理上，比 Spark 更给力（注：目前大数据分析引擎 Flink，已升级成为 Apache 顶级项目）。

Spark 和 Flink 都属于基础性的大数据处理引擎。具体的计算框架，大体上，可根据采用的模型及延迟的处理不同，来进行分门别类。

批处理（**Batch**）

MapReduce【55】– 这是谷歌有关 MapReduce 的最早的学术论文（注：对于国内用户，点击原文献链接可能会产生 404 错误，CSDN 网站有 MapReduce 论文的下载链接）。

MapReduce 综述【56】 –这是一篇过时、但依然值得一读的、有关 MapReduce 计算框架的综述性文章。

迭代式（**BSP**）

Pregel【57】–这又是一篇谷歌出品的大手笔论文，主要描述了大规模图处理方法（注：Pregel 是一种面向图算法的分布式编程框架，其采用的是迭代式的计算模型。它被称之为 Google 后 Hadoop 时代的新"三驾马车"之一。另外两驾马车分别是："交互式"大数据分析系统 Dremel 和网络搜索引擎 Caffeine）。

Giraph【58】 – 该系统建模于谷歌的 Pregel，可视为 Pregel 的开源版本，它是一个基于 Hadoop 架构的、可扩展的分布式迭代图处理系统。

GraphX【59】–这是一个同时采用图并行计算和数据并行的计算框架（注：GraphX 最先是加州大学伯克利分校 AMPLab 实验室的一个分布式图计算框架项目，后来整合到 Spark 中，成为其中的一个核心组件。GraphX 最大的贡献在于，在 Spark 之上提供一栈式数据解决方案，可方便高效地完成图计算的一整套流水作业）。

Hama【60】– 是一个构建 Hadoop 之上的基于 BSP 模型的分布式计算引擎（注：

Hama 的运行环境需要关联 Zookeeper、HBase、HDFS 组件。Hama 中最关键的技术，就是采用了 BSP 模型(Bulk Synchronous Parallel，即整体同步并行计算模型，又名大同步模型)。BSP 模型是哈佛大学的计算机科学家 Viliant 和牛津大学的 BillMcColl 在 1990 年联合提出的，他们希望能像冯·诺伊曼体系结构那样，架起计算机程序语言和体系结构间的桥梁，故又称作桥模型(Bridge Model)。

开源图处理系统【61】（Open source graph processing ）-这是滑铁卢大学的研究人员撰写的综述性文献，文献【61】对类 Pregel（Pregel-like）的、基于 BSP 模型的图处理系统进行了实验性的比较。

流式（**Streaming**）

流式处理【62】（Stream Processing）- 这是一篇非常棒的、有关面向大数据实时处理系统的综述性文章。

Storm【63】– 这是一个大数据实时处理系统（注：Storm 有时也被人们称为实时处理领域的 Hadoop，它大大简化了面向庞大规模数据流的处理机制，从而在实时处理领域扮演着重要角色。文献【63】是 Twitter 工程师们在 2014 年发表于 SIGMOD 上的学术论文）。

Samza【64】-这是一款由 Linkedin 公司开发的分布式的流式数据处理框架（注：所谓流式数据，是指要在处理单位内得到的数据，这种方式更注重于实时性，流式数据有时也称为快数据）。

Spark 流【65】（Spark Streaming） -该文献是加州大学伯克利分校的研究人员于 2013 年在著名操作系统会议 SOSP 上发表的学术论文，论文题目是《离散流：容错大规模流式计算》（注：这里的离散流是指一种微批处理构架，其桥接了传统的批处理和交互式处理。Spark Streaming 是 Spark 核心 API 的一个扩展，它并不会像 Storm 那样逐个处理数据流，而是在处理前，按时间间隔预先将其切分为很多小段的批处理作业）。

交互式（**Interactive**）

Dremel【66】–这又是一篇由谷歌出品的经典论文，论文描述了如何处理"交互式"大数据的工作负载。该论文是多个基于 Hadoop 的开源 SQL 系统的理论基础（注：文献【66】写于 2006 年，"捂"藏 4 年之后，于 2010 年公布于众。文章针对 MR 交互式查询能力不足，提出了 Dremel，阐述了 Dremel 的设计原理，并提供了部分测试报告）。

Impala【67】–这是一个大规模并行处理（MPP）式 SQL 大数据分析引擎（注：

Impala 像 Dremel 一样，其借鉴了 MPP（Massively Parallel Processing，大规模并行处理）并行数据库的思想，抛弃了 MapReduce 这个不太适合做 SQL 查询的范式，从而让 Hadoop 支持处理交互式的工作负载。本文作者阿尼尔马丹在 LinkedIn 上的博客原文，在此处的"MPI"系"MPP"笔误，读者可参阅文献【67】发现此问题）。

Drill【68】–这是谷歌 Dremel 的开源版本（注：Drill 是一个低延迟的、能对海量数据（包括结构化、半结构化及嵌套数据）实施交互式查询的分布式数据引擎）。

Shark【69】–该文献是 2012 年发表于 SIGMOD 的一篇学术论文，论文对 Spark 生态系统上的数据分析能力，给出了很深入的介绍（注：Shark 是由加州伯克利大学 AMPLab 开发的大数据分析系统。Shark 即"Hive on Spark"的含义，本质上是通过 Hive 的 HQL 解析，把 HQL 翻译成 Spark 上的 RDD 操作。然后通过 Hive 的元数据获，取数据库里的表信息。HDFS 上的数据和文件，最后会由 Shark 获取，并放到 Spark 上运算。Shark 基于 Scala 语言的算子推导，可实现良好的容错机制，对执行失败的长/短任务，均能从上一个"快照点（Snapshot）"进行快速恢复）。

Shark【70】–这是另外一篇很棒的于 2013 年发表在 SIGMOD 的学术论文，其深度解读在 Apache Hive 之上 SQL 访问机制（注：这篇文献描述了如何构建在 Spark 上构建 SQL 引擎——Shark。更重要的是，文章还讨论了之前在 Hadoop/MapReduce 上实施 SQL 查询如此之慢的原因）。

Dryad【71】– 文献讨论了使用有向无环图(Directed Acycline Graph，DAG)来配置和执行并行数据流水线的方法（注：Dryad 是一个通用的粗颗粒度的分布式计算和资源调度引擎，其核心特性

之一，就是允许用户自己构建 DAG 调度拓扑图。文献【71】是微软于 2007 年在 EuroSys 国际会议上发布的学术论文）。

Tez【72】–其核心思想来源于 Dryad，可视为利用 Yarn (即 MRv2) 对 Dryad 的开源实现（注：Apache Tez 是基于 Hadoop Yarn 之上的 DAG 计算框架。由 Hadoop 的二东家 Hortonworks 开发并提供主要技术支持。文献【72】是一个关于 Tez 的简要介绍文档）。

BlinkDB【73】–可在抽样数据上实现交互式查询，其呈现出的查询结果，附带有误差标识。

（注：BlinkDB 是一个用于在海量数据上运行交互式 SQL 查询的大规模并行查询引擎。BlinkDB 允许用户通过适当降低数据精度，对数据进行先采样后计算，其通过其独特的优化技术，实现了比 Hive 快百倍的交互式查询速度，而查询进度误差仅降低2~10%。

BlinkDB 采用的策略，与大数据布道师，维克托·迈尔-舍恩伯格在其著作《大数据时代》中提到的观点，“要全体，不要抽样”，恰恰相反。

基于常识，我们知道：多了，你就快不了。好了，你就省不了。对大数据处理而言，也是这样。英特尔中国研究院院长吴甘沙认为，大体量、精确性和速度快，三者不可兼得，顶多取其二。如果要实现在大体量数据上的“快”，就得想办法减少数据，而减少数据，势必要适度地降低分析精确性。

事实上，大数据并不见得越“大”越好，有时候一味的追求“大”是没有必要的。例如，在医疗健康领域，如果来监控某个病人的体温，可穿戴设备可以一秒钟采集一次数据，也可以一分钟采集一次数据，前者采集的数据总量比后者“大”60 倍，但就监控病人身体状况而言，意义并不是太大。虽然后者的数据忽略了人体在一分钟内的变化，监控的精度有所下降，但对于完成监控病人健康状态这一目的而言，是可以接受的。）

实时系统（**RealTime**）

Druid【74】–这是一个开源的分布式实时数据分析和存储系统，旨在快速处理大规模的数据，并能做到快速查询和分析（注：文献【74】是 2014 年 Druid 创始人 Eric Tschetter 和中国工程师杨仿今等人在 SIGMOD 上发表的一篇论文）。

Pinot【75】–这是由 LinkedIn 公司出品的一个开源的、实时分布式的 OLAP 数据分析存储系统，非常类似于前面提到的 Druid，LinkedIn 使用它实现低延迟可伸缩的实时分析。（注：文献【75】是在 GitHub 上的有关 Pinot 的说明性文档）。

数据分析层（Data Analysis）

数据分析层中的工具，涵盖范围很广，从诸如 SQL 的声明式编程语言，到诸如 Pig 的过程化编程语言，均有涉及。另一方面，数据分析层中的库也很丰富，可支持常见的数据挖掘和机器学习算法，这些类库可拿来即用，甚是方便。

工具（**Tools**）

Pig【76】–这是一篇有关 Pig Latin 非常不错的综述文章（注：Pig Latin 原是一种儿童黑话，属于是一种英语语言游戏，形式是在英语上加上一点规则使发音改变，让大人们听不懂，从而完成孩子们独懂的交流。文献【76】是雅虎的工程师们于 2008 年发表在 SIGMOD 的一篇论文，论文的题目是“Pig Latin：并不是太老外的一种数据语言”，言外之意，他们发明了一种数据处理的“黑话”——Pig Latin，一开始你可能不懂，等你熟悉了，就会发现这种数据查询语言的乐趣所在）。

Pig【77】– 这是另外一篇由雅虎工程师们撰写的有关使用 Pig 经验的论文，文章介绍了如果利用 Pig 在 Map-Reduce 上构建一个高水准的数据流分析系统。

Hive【78】–该文献是 Facebook 数据基础设施研究小组撰写的一篇学术论文，介绍了 Hive 的来龙去脉（注：Hive 是一个建立于 Hadoop 上的数据仓库基础构架。它用来进行数据的提取、转化和加载（即 Extract-Transform-Load ， ETL），它是一种可以存储、查询和分析存储在 Hadoop 中的大规模数据的机制）。

Hive【79】–该文献是另外一篇有关 Hive 的值得一读的好论文。论文作者来自 Facebook 数据基础设施研究小组，在这篇论文里，可以帮助读者理解 Hive 的设计理念。

Phoenix【80】–它是 HBase 的 SQL 驱动（注：Phoenix 可将 SQL 查询转成 HBase 的扫描及相应的动作。文献【80】是关于在 Hbase 上部署 SQL 的幻灯片文档）。

Map Reduce 上的连接（join）算法【81】–该文献介绍了在 Hadoop 环境下的各种并行连接算法，并对它们的性能作出系统性评测。

Map Reduce 上的连接算法【82】–这是威斯康星大学和 IBM 研究团队撰写的综述性文章，文

章对在 Map Reduce 模型下的各种连接算法进行了综合比较。

库（**Libraires**）

MLlib【83】–这是在 Spark 计算框架中对常用的机器学习算法的实现库，该库还包括相关的测试和数据生成器（注：文献【83】是 MLlib 的一个幻灯片说明文档）。

SparkR【84】–这是 AMPLab 发布的一个R开发包，为 Apache Spark 提供轻量级的前端（注：R是一种广泛应用于统计分析、绘图的语言及操作环境。文献【84】是有关 SparkR 的幻灯片文档）。

Mahout【85】–这是一个功能强大的数据挖掘工具，是一个基于传统 Map Reduce 的分布式机器学习框架（注：Mahout 的中文含义就是"驭象之人"，而 Hadoop 的 Logo 正是一头小黄象。很明显，这个库是帮助用户用好 Hadoop 这头难用的大象。文献【85】是有关 Mahout 的图书）。

数据集成层（**Data Integration**）

数据集成框架提供了良好的机制，以协助高效地摄取和输出大数据系统之间的数据。从业务流程线到元数据框架，数据集成层皆有涵盖，从而提供全方位的数据在整个生命周期的管理和治理。

摄入/消息传递（**Ingest/Messaging**）

Flume【86】–这是 Apache 旗下的一个分布式的、高可靠的、高可用的服务框架，可协助从分散式或集中式数据源采集、聚合和传输海量日志（注：文献【86】是 Apache 网站上有关 Flume 的一篇博客文章）。

Sqoop【87】–该系统主要用来在 Hadoop 和关系数据库中传递数据（注：Sqoop 目前已成为 Apache 的顶级项目之一。通过 Sqoop，可以方便地将数据从关系数据库导入到 HDFS，或反之亦可。文献【87】是有关 Sqoop 的幻灯片说明文档）。

Kafka【88】–这是由 LinkedIn 开发的一个分布式消息系统（注：由 Scala 编写而成的 Kafka，由于可水平扩展、吞吐率高等特性，得到广泛应用。文献【88】是 LindedIn 的工程师们在 2011 年发表于 NetDB 的会议论文）。

ETL/工作流

ETL 是数据抽取（Extract）、清洗（Cleaning）、转换（Transform）、装载（Load）的过程，是构建数据仓库的重要一环。

Crunch【89】–这是 Apache 旗下的一套 Java API 函数库，它能够大大简化编写、测试、运行 MapReduce 处理工作流的程序（注：文献【89】是有关 Crunch 的幻灯片解释文档）。

Falcon【90】– 这是 Apache 旗下的 Falcon 大数据管理框架，可以帮助用户自动迁移和处理大数据集合（注：文献【90】是一份关于 Falcon 技术预览报告）。

Cascading【91】–这是一个架构在 Hadoop 上的 API 函数库，用来创建复杂的可容错的数据处理工作流（注：文献【91】是关于 Hadoop 上的 Cascading 的概论和技术随笔）。

Oozie【92】–是一个工作流引擎，用来协助 Hadoop 作业管理（注：Oozie 字面含义是驯象之人，其寓意和 Mahout 一样，帮助用户更好地搞定 Hadoop 这头大象。文献【92】是 Apache 网站上有关 Oozie 的官方文档）。

元数据（**Metadata**）

HCatalog【93】– 它提供了面向 Apache Hadoop 的数据表和存储管理服务（注：Apache HCatalog 提供一个共享的模式和数据类型的机制，它抽象出表，使用户不必关心数据怎么存储，并提供了可操作的跨数据处理工具。文献【93】是 Apache 网站有关 Hcatalog 的官方说明文档）。

序列化（**Serialization**）

Protocol Buffers【94】–由 Google 推广的一种与语言无关的、对结构化数据进行序列化和反序列化的机制（注：Protocol Buffers 可用于通讯协议、数据存储等领域的语言及平台无关、可扩展的序列化结构数据格式。文献【94】是有关 Protocol Buffers 幻灯片文档）。

Avro【95】–这是一个建模于 Protocol Buffers 之上的、Hadoop 生态系统中的子项目（注：Avro 本身既是一个序列化框架，同时也实现了 RPC 的功能）。

操作框架（**Operational Frameworks**）

最后，我们还需要一个操作性框架，来构建一套衡量标准和测试基准，从而来评价各种计算框架

的性能优劣。在这个操作性框架中，还需要包括性能优化工具，借助它来平衡工作负载。

监测管理框架（**Monitoring Frameworks**）

OpenTSDB【96】 –这是构建于 HBase 之上的实时性能评测系统（注：文献【96】提供了 OpenTSDB 的简要概述，介绍了 OpenTSDB 的工作机理）。

Ambari【97】– 这是一款基于 Web 的系统，支持 Apache Hadoop 集群的供应、管理和监控（注：文献【97】阐述了 Ambari 架构的设计准则）。

基准测试（**Benchmarking**）

YCSB【98】 –该文献是一篇使用 YCSB 对 NoSQL 系统进行性能评估的期刊论文（注：YCSB 是雅虎云服务基准测试（Yahoo! Cloud Serving Benchmark）的简写。见名知意，它是由雅虎出品的一款通用云服务性能测试工具）。

GridMix【99】 –该系统通过运行大量合成的作业，对 Hadoop 系统进行基准测试，从而获得性能评价指标（注：文献是 Apache 网站有关 GridMix 的官方说明文档）。

最后一篇文献是有关大数据基准测试的综述文章【100】，文章讨论了基准测试的最新技术进展以及所面临的几个主要挑战。

译者寄语：

在你迈步于大数据的旅途中，真心希望这些文献能助你一臂之力。但要知道，有关大数据的文献，何止千万，由于个人精力、能力有限，有些领域也不甚熟稔，故难免会挂一漏万。如有疏忽，漏掉你的大作，还请你海涵。最后，希望这些文献能给你带来“学而时习之，不亦乐乎”的快感！

译者介绍：张玉宏，博士。2012 年毕业于电子科技大学，现执教于河南工业大学。中国计算机协会（CCF）会员，ACM/IEEE 会员。主要研究方向为高性能计算、生物信息学，主编有《Java 从入门到精通》一书。

来自: CSDN

标签： PayPal 大数据

« 上一篇：Mozilla考虑从Firefox剥离XUL和XBL(2015-07-07 13:09)

» 下一篇：小米汽车的多项专利被曝光：确实有做汽车的准备(2015-07-07 13:34)

已经有 **16** 位园友对此新闻发表了看法。

第**1**楼 Lazyzoon 发表于 2015-07-07 13:45

mark

支持(0) 反对(0) 回复 引用

第**2**楼 小灬菜 发表于 2015-07-07 14:01

mark

支持(0) 反对(0) 回复 引用

第**3**楼 国立秀才 发表于 2015-07-07 14:09

mark

支持(0) 反对(0) 回复 引用

第**4**楼 高导 发表于 2015-07-07 14:24

这个得实践吧。

支持(0) 反对(0) 回复 引用

第**5**楼 胡萝卜星星 发表于 2015-07-07 14:43

为了读懂这100篇论文，你得先读懂起码10本专业书籍

支持(0) 反对(0) 回复 引用

第**6**楼 木叶之夏 发表于 2015-07-07 14:56

mark、

支持(0) 反对(0) 回复 引用

第**7**楼 K战神 发表于 2015-07-07 15:01

没什么卵用~也就是理论高手~实际，很困难

支持(1) 反对(0) 回复 引用

第**8**楼 newbreeze 发表于 2015-07-07 15:11

mark

支持(0) 反对(0) 回复 引用

第**9**楼 重构者 发表于 2015-07-07 15:24

mark

支持(0) 反对(0) 回复 引用

第**10**楼 楠小楠 发表于 2015-07-07 15:33

@胡萝卜星星
论文可不是看一遍二遍就懂的，首先，你英语要好，其次，你底子要好。

支持(0) 反对(0) 回复 引用

第**11**楼 wanglifeng 发表于 2015-07-07 15:36

mark

支持(0) 反对(0) 回复 引用

第**12**楼 浙入围城 发表于 2015-07-07 15:56

mark

支持(0) 反对(0) 回复 引用

第**13**楼 水之乡 发表于 2015-07-07 16:01

mark

支持(0) 反对(0) 回复 引用

第**14**楼 树大 发表于 2015-07-07 17:17

mark
有什么用，最后还不是不去看

支持(0) 反对(0) 回复 引用

第**15**楼 向振文 发表于 2015-07-07 18:33

mark

支持(0) 反对(0) 回复 引用

第**16**楼 顾晓北 发表于 2015-07-08 09:19

...

支持(0) 反对(0) 回复 引用

刷新评论

提交评论 Ctrl + Enter键快速提交