[**一共81个，开源大数据处理工具汇总（下），包括日志收集系统/集群管理/RPC等**](http://www.36dsj.com/archives/25042)

**作者：大数据女神-诺蓝（微信公号：dashujunvshen）。本文是36大数据专稿，转载必须标明来源36大数据。**

接上一部分：[一共81个，开源大数据处理工具汇总（上）](http://www.36dsj.com/archives/24852)，第二部分主要收集整理的内容主要有日志收集系统、消息系统、分布式服务、集群管理、RPC、基础设施、搜索引擎、Iaas和监控管理等大数据开源工具。

**日志收集系统**

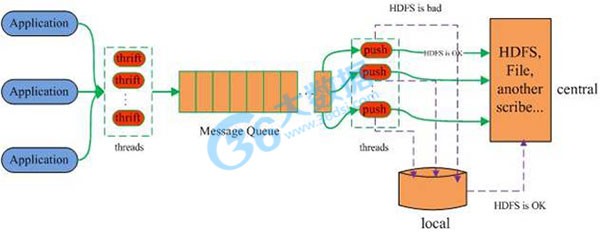
**一、Facebook Scribe**



**贡献者**：Facebook

**简介**：Scribe是Facebook开源的日志收集系统，在Facebook内部已经得到大量的应用。它能够从各种日志源上收集日志，存储到一个中央存储系统（可以是NFS，分布式文件系统等）上，以便于进行集中统计分析处理。它为日志的“分布式收集，统一处理”提供了一个可扩展的，高容错的方案。当中央存储系统的网络或者机器出现故障时，scribe会将日志转存到本地或者另一个位置，当中央存储系统恢复后，scribe会将转存的日志重新传输给中央存储系统。其通常与Hadoop结合使用，scribe用于向HDFS中push日志，而Hadoop通过MapReduce作业进行定期处理。

**Scribe的系统架构**



**代码托管**：<https://github.com/facebook/scribe>

**二、Cloudera Flume**

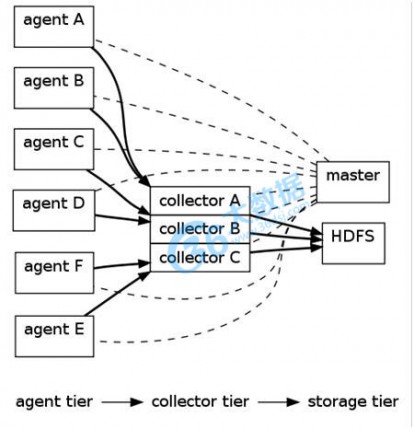
**贡献者**：[Cloudera](http://www.36dsj.com/archives/tag/cloudera)

**简介：**Flume是Cloudera提供的一个高可用的，高可靠的，分布式的海量日志采集、聚合和传输的系统，Flume支持在日志系统中定制各类数据发送方，用于收集数据；同时，Flume提供对数据进行简单处理，并写到各种数据接受方（可定制）的能力。

Flume提供了从console（控制台）、RPC（Thrift-RPC）、text（文件）、tail（UNIX tail）、syslog（syslog日志系统，支持TCP和UDP等2种模式），exec（命令执行）等数据源上收集数据的能力。

当前Flume有两个版本Flume 0.9X版本的统称Flume-og，Flume1.X版本的统称Flume-ng。由于Flume-ng经过重大重构，与Flume-og有很大不同，使用时请注意区分。

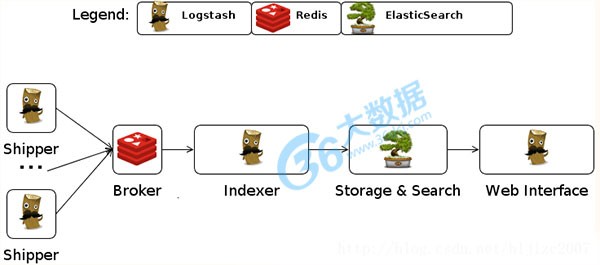
**Cloudera Flume构架**：



**官网**：<http://flume.apache.org/>

**三、logstash**

**简介**：logstash 是一个应用程序日志、事件的传输、处理、管理和搜索的平台。你可以用它来统一对应用程序日志进行收集管理，提供 Web 接口用于查询和统计。他可以对你的日志进行收集、分析，并将其存储供以后使用（如，搜索），您可以使用它。说到搜索，logstash带有一个web界面，搜索和展示所有日志。



**官网**：<http://www.logstash.net/>

**四、kibana**

**简介**：Kibana 是一个为 Logstash 和 ElasticSearch 提供的日志分析的 Web 接口。可使用它对日志进行高效的搜索、可视化、分析等各种操作。kibana 也是一个开源和免费的工具，他可以帮助您汇总、分析和搜索重要数据日志并提供友好的web界面。他可以为 Logstash 和 ElasticSearch 提供的日志分析的 Web 界面。

**主页**： <http://kibana.org/>

**代码托管**： <https://github.com/rashidkpc/Kibana/downloads>

**消息系统**

**一、StormMQ**

**简介**：MQMessageQueue消息队列产品 StormMQ，是一种服务程序。

**官网**：<http://stormmq.com/>

**二、ZeroMQ**

**简介**：这是个类似于Socket的一系列接口，他跟Socket的区别是：普通的socket是端到端的（1:1的关系），而ZMQ却是可以N：M 的关系，人们对BSD套接字的了解较多的是点对点的连接，点对点连接需要显式地建立连接、销毁连接、选择协议（TCP/UDP）和处理错误等，而ZMQ屏蔽了这些细节，让你的网络编程更为简单。ZMQ用于node与node间的通信，node可以是主机或者是进程。

引用官方的说法： “ZMQ(以下ZeroMQ简称ZMQ)是一个简单好用的传输层，像框架一样的一个socket library，他使得Socket编程更加简单、简洁和性能更高。是一个消息处理队列库，可在多个线程、内核和主机盒之间弹性伸缩。ZMQ的明确目标是“成为标准网络协议栈的一部分，之后进入Linux内核”。现在还未看到它们的成功。但是，它无疑是极具前景的、并且是人们更加需要的“传统”BSD套接字之上的一 层封装。ZMQ让编写高性能网络应用程序极为简单和有趣。”

**官网**：<http://zeromq.org/>

**三、RabbitMQ**

**简介：**RabbitMQ是一个受欢迎的消息代理，通常用于应用程序之间或者程序的不同组件之间通过消息来进行集成。本文简单介绍了如何使用 RabbitMQ，假定你已经配置好了rabbitmq服务器。



RabbitMQ是用Erlang，对于主要的编程语言都有驱动或者客户端。我们这里要用的是Java，所以先要获得Java客户端。

像RabbitMQ这样的消息代理可用来模拟不同的场景，例如点对点的消息分发或者订阅/推送。我们的程序足够简单，有两个基本的组件，一个生产者用于产生消息，还有一个消费者用来使用产生的消息。

**官网**：<https://www.rabbitmq.com/>

**四、Apache ActiveMQ**

**简介：**ActiveMQ 是Apache出品，最流行的，能力强劲的开源消息总线。ActiveMQ 是一个完全支持JMS1.1和J2EE 1.4规范的 JMS Provider实现，尽管JMS规范出台已经是很久的事情了，但是JMS在当今的J2EE应用中间仍然扮演着特殊的地位。

Apache ActiveMQ

**特性：**

⒈ 多种语言和协议编写客户端。语言: Java,C,C++,C#,Ruby,Perl,Python,PHP。应用协议： OpenWire,Stomp REST,WS Notification,XMPP,AMQP

⒉ 完全支持JMS1.1和J2EE 1.4规范 （持久化，XA消息，事务)

⒊ 对Spring的支持，ActiveMQ可以很容易内嵌到使用Spring的系统里面去，而且也支持Spring2.0的特性

⒋ 通过了常见J2EE服务器（如 Geronimo,JBoss 4,GlassFish,WebLogic)的测试，其中通过JCA 1.5 resource adaptors的配置，可以让ActiveMQ可以自动的部署到任何兼容J2EE 1.4 商业服务器上

⒌ 支持多种传送协议：in-VM,TCP,SSL,NIO,UDP,JGroups,JXTA

⒍ 支持通过JDBC和journal提供高速的消息持久化

⒎ 从设计上保证了高性能的集群，客户端-服务器，点对点

⒏ 支持Ajax

⒐ 支持与Axis的整合

⒑ 可以很容易得调用内嵌JMS provider，进行测试

**官网**：<http://activemq.apache.org/>

**五、Jafka**

**贡献者**：LinkedIn

**简介：**Jafka 是一个开源的、高性能的、跨语言分布式消息系统，使用GitHub托管。Jafka 最早是由Apache孵化的Kafka（由LinkedIn捐助给Apache）克隆而来。由于是一个开放式的数据传输协议，因此除了Java开发语言受到支持，Python、Ruby、C、C++等其他语言也能够很好的得到支持。

**特性：**

1、消息持久化非常快，服务端存储消息的开销为O(1)，并且基于文件系统，能够持久化TB级的消息而不损失性能。

2、吞吐量取决于网络带宽。

3、完全的分布式系统，broker、producer、consumer都原生自动支持分布式。自动实现复杂均衡。

4、内核非常小，整个系统（包括服务端和客户端）只有一个272KB的jar包，内部机制也不复杂，适合进行内嵌或者二次开发 。整个服务端加上依赖组件共3.5MB。

5、消息格式以及通信机制非常简单，适合进行跨语言开发。目前自带的Python3.x的客户端支持发送消息和接收消息。

**官网**：<http://kafka.apache.org/>

**六、Apache Kafka**

**贡献者**：LinkedIn

**简介**：Apache Kafka是由Apache软件基金会开发的一个开源消息系统项目，由Scala写成。Kafka最初是由LinkedIn开发，并于2011年初开源。2012年10月从Apache Incubator毕业。该项目的目标是为处理实时数据提供一个统一、高通量、低等待的平台。

Kafka是一个分布式的、分区的、多复本的日志提交服务。它通过一种独一无二的设计提供了一个消息系统的功能。

Kafka集群可以在一个指定的时间内保持所有发布上来的消息，不管这些消息有没有被消费。打个比方，如果这个时间设置为两天，那么在消息发布的两天以内，这条消息都是可以被消费的，但是在两天后，这条消息就会被系统丢弃以释放空间。Kafka的性能不会受数据量的大小影响，因此保持大量的数据不是一个问题。

**官网**：<http://kafka.apache.org/>

**分布式服务**

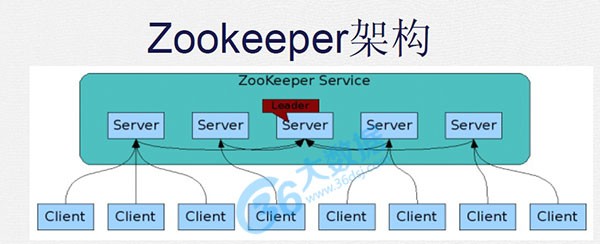
**一、ZooKeeper**

**贡献者：**Google

**简介：**ZooKeeper是一个分布式的，开放源码的分布式应用程序协调服务，是Google的Chubby一个开源的实现，是Hadoop和Hbase的重要组件。它是一个为分布式应用提供一致性服务的软件，提供的功能包括：配置维护、名字服务、分布式同步、组服务等。

ZooKeeper是以Fast Paxos算法为基础的，paxos算法存在活锁的问题，即当有多个proposer交错提交时，有可能互相排斥导致没有一个proposer能提交成功，而Fast Paxos作了一些优化，通过选举产生一个leader，只有leader才能提交propose，具体算法可见Fast Paxos。因此，要想弄懂ZooKeeper首先得对Fast Paxos有所了解。

架构：



官网：<http://zookeeper.apache.org/>

**RPC**

（Remote Procedure Call Protocol）——远程过程调用协议

一、Apache Avro

简介：Apache Avro是Hadoop下的一个子项目。它本身既是一个序列化框架，同时也实现了RPC的功能。Avro官网描述Avro的特性和功能如下：

* 丰富的数据结构类型；
* 快速可压缩的二进制数据形式；
* 存储持久数据的文件容器；
* 提供远程过程调用RPC；
* 简单的动态语言结合功能。

相比于Apache Thrift 和Google的Protocol Buffers，Apache Avro具有以下特点：

* 支持动态模式。Avro不需要生成代码，这有利于搭建通用的数据处理系统，同时避免了代码入侵。
* 数据无须加标签。读取数据前，Avro能够获取模式定义，这使得Avro在数据编码时只需要保留更少的类型信息，有利于减少序列化后的数据大小。

官网：<http://avro.apache.org/>

二、Facebook Thrift

贡献者：Facebook

简介：Thrift源于大名鼎鼎的facebook之手，在2007年facebook提交Apache基金会将Thrift作为一个开源项目，对于当时的facebook来说创造thrift是为了解决facebook系统中各系统间大数据量的传输通信以及系统之间语言环境不同需要跨平台的特性。

thrift可以支持多种程序语言，例如: C++, C#, Cocoa, Erlang, Haskell, Java, Ocami, Perl, PHP, Python, Ruby, Smalltalk. 在多种不同的语言之间通信thrift可以作为二进制的高性能的通讯中间件，支持数据(对象)序列化和多种类型的RPC服务。

Thrift适用于程序对程 序静态的数据交换，需要先确定好他的数据结构，他是完全静态化的，当数据结构发生变化时，必须重新编辑IDL文件，代码生成，再编译载入的流程，跟其他IDL工具相比较可以视为是Thrift的弱项，Thrift适用于搭建大型数据交换及存储的通用工具，对于大型系统中的内部数据传输相对于JSON和xml无论在性能、传输大小上有明显的优势。

Thrift 主要由5个部分组成：

· 类型系统以及 IDL 编译器：负责由用户给定的 IDL 文件生成相应语言的接口代码

· TProtocol：实现 RPC 的协议层，可以选择多种不同的对象串行化方式，如 JSON, Binary。

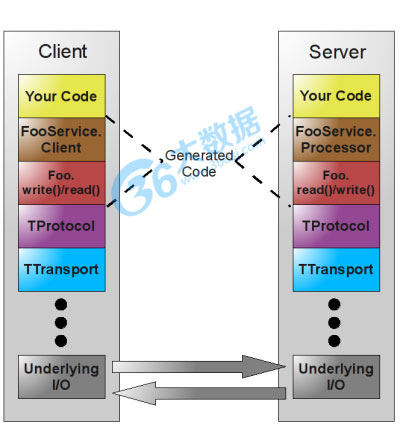
· TTransport：实现 RPC 的传输层，同样可以选择不同的传输层实现，如socket, 非阻塞的 socket, MemoryBuffer 等。

· TProcessor：作为协议层和用户提供的服务实现之间的纽带，负责调用服务实现的接口。

· TServer：聚合 TProtocol, TTransport 和 TProcessor 几个对象。

上述的这5个部件都是在 Thrift 的源代码中通过为不同语言提供库来实现的，这些库的代码在 Thrift 源码目录的 lib 目录下面，在使用 Thrift 之前需要先熟悉与自己的语言对应的库提供的接口。

Facebook Thrift构架：



官网：<http://thrift.apache.org/>

**集群管理**

**一、Nagios**

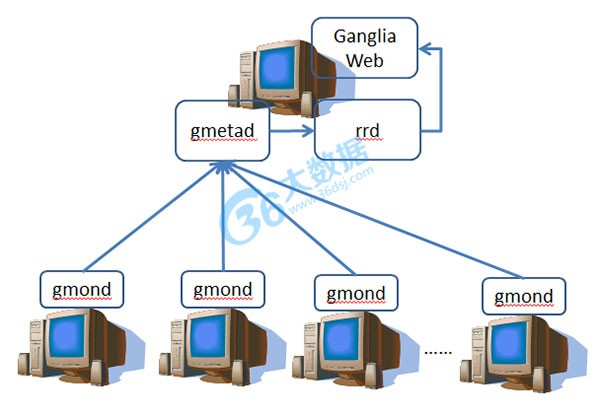
**简介：**Nagios是一款开源的免费网络监视工具，能有效监控Windows、Linux和Unix的主机状态，交换机路由器等网络设置，打印机等。在系统或服务状态异常时发出邮件或短信报警第一时间通知网站运维人员，在状态恢复后发出正常的邮件或短信通知。

Nagios可运行在Linux/Unix平台之上，同时提供一个可选的基于浏览器的WEB界面以方便系统管理人员查看网络状态，各种系统问题，以及日志等等。

**官网：**<http://www.nagios.org/>

**二、Ganglia**

**简介：**Ganglia是UC Berkeley发起的一个开源集群监视项目，设计用于测量数以千计的节点。Ganglia的核心包含gmond、gmetad以及一个Web前端。主要是用来监控系统性能，如：cpu 、mem、硬盘利用率， I/O负载、网络流量情况等，通过曲线很容易见到每个节点的工作状态，对合理调整、分配系统资源，提高系统整体性能起到重要作用。

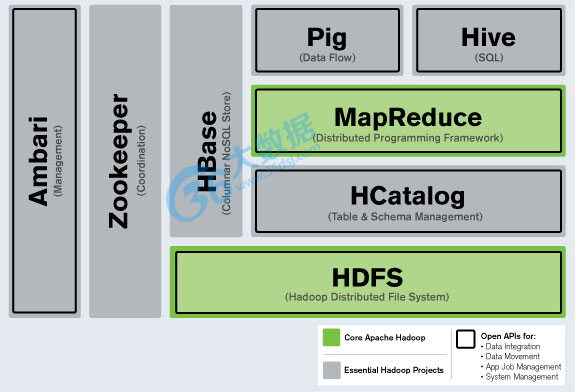


**官网：**<http://ganglia.sourceforge.net/>

**三、Apache Ambari**

**简介：**Apache Ambari是一种基于Web的工具，支持Apache Hadoop集群的供应、管理和监控。Ambari目前已支持大多数Hadoop组件，包括HDFS、MapReduce、Hive、Pig、 Hbase、Zookeper、Sqoop和Hcatalog等。

Apache Ambari 支持HDFS、MapReduce、Hive、Pig、Hbase、Zookeper、Sqoop和Hcatalog等的集中管理。也是5个顶级hadoop管理工具之一。



**Ambari主要取得了以下成绩：**

* 通过一步一步的安装向导简化了集群供应。
* 预先配置好关键的运维指标（metrics），可以直接查看Hadoop Core（HDFS和MapReduce）及相关项目（如HBase、Hive和HCatalog）是否健康。
* 支持作业与任务执行的可视化与分析，能够更好地查看依赖和性能。
* 通过一个完整的RESTful API把监控信息暴露出来，集成了现有的运维工具。
* 用户界面非常直观，用户可以轻松有效地查看信息并控制集群。

Ambari使用Ganglia收集度量指标，用Nagios支持系统报警，当需要引起管理员的关注时（比如，节点停机或磁盘剩余空间不足等问题），系统将向其发送邮件。

此外，Ambari能够安装安全的（基于Kerberos）Hadoop集群，以此实现了对Hadoop 安全的支持，提供了基于角色的用户认证、授权和审计功能，并为用户管理集成了LDAP和Active Directory。

**官网**：<http://ambari.apache.org/>

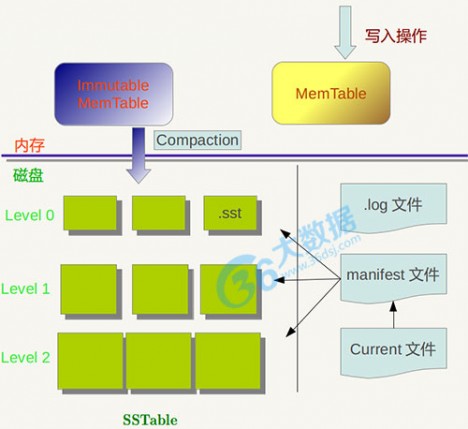
**基础设施**

**一、LevelDB**

**贡献者**：Jeff Dean和Sanjay Ghemawat

**简介：**Leveldb是一个google实现的非常高效的kv数据库，目前的版本1.2能够支持billion级别的数据量了。 在这个数量级别下还有着非常高的性能，主要归功于它的良好的设计。特别是LMS算法。LevelDB 是单进程的服务，性能非常之高，在一台4核Q6600的CPU机器上，每秒钟写数据超过40w，而随机读的性能每秒钟超过10w。

**Leveldb框架**：



**官网**：<http://code.google.com/p/leveldb/>

**二、SSTable**

**简介**：如果说Protocol Buffer是谷歌独立数据记录的通用语言 ，那么有序字符串表（SSTable，Sorted String Table）则是用于存储，处理和数据集交换的最流行​​的数据输出格式。正如它的名字本身，SSTable是有效存储大量键-值对的简单抽象，对高吞吐量顺序读/写进行了优化。

SSTable是Bigtable中至关重要的一块，对于LevelDB来说也是如此。

**三、RecordIO**

**贡献者**：Google

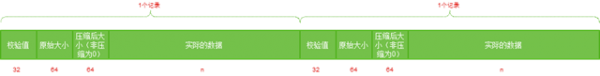
**简介：**我们大家都在用文件来存储数据。文件是存储在磁盘上的。如果在一些不稳定的介质上，文件很容损坏。即时文件某个位置出现一点小小的问题，整个文件就废了。

下面我来介绍Google的一个做法，可以比较好的解决这个问题。那就是recordio文件格式。recoidio的存储单元是一个一个record。这个record可以根据业务的需要自行定义。但Google有一种建议的处理方式就是使用protobuf。

reocordio底层的格式其实很简单。一个record由四部分组成：

* MagicNumber (32 bits)
* Uncompressed data payload size (64 bits)
* Compressed data payload size (64 bits), or 0 if the data is not compressed
* Payload, possibly compressed.

**详细格式如下图所示：**



到这里，大家可能已经知道，recordio之所以能对付坏数据，其实就是在这个MagicNumber（校验值）。

**四、Flat Buffers**

**贡献者**：Google

**简介**：谷歌开源高效、跨平台的序列化库FlatBuffers。

该库的构建是专门为游戏开发人员的性能需求提供支持，它将序列化数据存储在缓存中，这些数据既可以存储在文件中，又可以通过网络原样传输，而不需要任何解析开销。

**FlatBuffers有如下一些关键特性—**—

* 访问序列化数据不需要打包/拆包
* 节省内存而且访问速度快——缓存只占用访问数据所需要的内存；不需要任何额外的内存。
* 灵活性——通过可选字段向前向后兼容
* 代码规模小
* 强类型——错误在编译时捕获，而不是在运行时
* 便利性——生成的C++头文件代码简洁。如果需要，有一项可选功能可以用来在运行时高效解析Schema和JSON-like格式的文本。
* 跨平台——使用C++编写，不依赖STL之外的库，因此可以用于任何有C++编辑器的平台。当前，该项目包含构建方法和在Android、Linux、Windows和OSX等操作系统上使用该库的示例。

与Protocol Buffers或JSON Parsing这样的可选方案相比，FlatBuffers的优势在于开销更小，这主要是由于它没有解析过程。

**代码托管**：<https://github.com/google/flatbuffers>

**五、Protocol Buffers**

**贡献者：**Google

**简介**：Protocol Buffers是Google公司开发的一种数据描述语言，类似于XML能够将结构化数据序列化，可用于数据存储、通信协议等方面。它不依赖于语言和平台并且可扩展性极强。现阶段官方支持C++、JAVA、Python等三种编程语言，但可以找到大量的几乎涵盖所有语言的第三方拓展包。

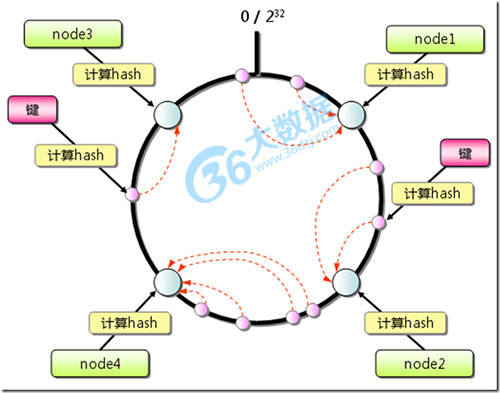
通过它，你可以定义你的数据的结构，并生成基于各种语言的代码。这些你定义的数据流可以轻松地在传递并不破坏你已有的程序。并且你也可以更新这些数据而现有的程序也不会受到任何的影响。

Protocol Buffers经常被简称为protobuf。

**官网**：<http://code.google.com/p/protobuf/>

**六、Consistent Hashing（哈希算法）**

**简介：**一致性哈希算法在1997年由麻省理工学院提出的一种分布式哈希（DHT）实现算法，设计目标是为了解决因特网中的热点(Hot spot)问题，初衷和CARP十分类似。一致性哈希修正了CARP使用的简 单哈希算法带来的问题，使得分布式哈希（DHT）可以在P2P环境中真正得到应用。



一致性hash算法提出了在动态变化的Cache环境中，**判定哈希算法好坏的四个定义：**

1、平衡性(Balance)：平衡性是指哈希的结果能够尽可能分布到所有的缓冲中去，这样可以使得所有的缓冲空间都得到利用。很多哈希算法都能够满足这一条件。

2、单调性(Monotonicity)：单调性是指如果已经有一些内容通过哈希分派到了相应的缓冲中，又有新的缓冲加入到系统中。哈希的结果应能够保证原有已分配的内容可以被映射到原有的或者新的缓冲中去，而不会被映射到旧的缓冲集合中的其他缓冲区。

3、分散性(Spread)：在分布式环境中，终端有可能看不到所有的缓冲，而是只能看到其中的一部分。当终端希望通过哈希过程将内容映射到缓冲上时，由于不同终端所见的缓冲范围有可能不同，从而导致哈希的结果不一致，最终的结果是相同的内容被不同的终端映射到不同的缓冲区中。这种情况显然是应该避免的，因为它导致相同内容被存储到不同缓冲中去，降低了系统存储的效率。分散性的定义就是上述情况发生的严重程度。好的哈希算法应能够尽量避免不一致的情况发生，也就是尽量降低分散性。

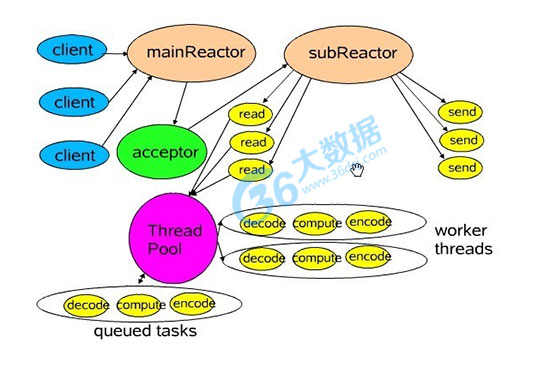
4、负载(Load)：负载问题实际上是从另一个角度看待分散性问题。既然不同的终端可能将相同的内容映射到不同的缓冲区中，那么对于一个特定的缓冲区而言，也可能被不同的用户映射为不同 的内容。与分散性一样，这种情况也是应当避免的，因此好的哈希算法应能够尽量降低缓冲的负荷。

在分布式集群中，对机器的添加删除，或者机器故障后自动脱离集群这些操作是分布式集群管理最基本的功能。如果采用常用的hash(object)%N算法，那么在有机器添加或者删除后，很多原有的数据就无法找到了，这样严重的违反了单调性原则。

**七、Netty**

**贡献者**：JBOSS

**简介：**Netty是由JBOSS提供的一个java开源框架。Netty提供异步的、事件驱动的网络应用程序框架和工具，用以快速开发高性能、高可靠性的网络服务器和客户端程序。



也就是说，Netty 是一个基于NIO的客户，服务器端编程框架，使用Netty 可以确保你快速和简单的开发出一个网络应用，例如实现了某种协议的客户，服务端应用。Netty相当简化和流线化了网络应用的编程开发过程，例如，TCP和UDP的socket服务开发。

“快速”和“简单”并不意味着会让你的最终应用产生维护性或性能上的问题。Netty 是一个吸收了多种协议的实现经验，这些协议包括FTP,SMTP,HTTP，各种二进制，文本协议，并经过相当精心设计的项目，最终，Netty 成功的找到了一种方式，在保证易于开发的同时还保证了其应用的性能，稳定性和伸缩性。

**官网**：<http://netty.io/>

**八、BloomFilter**

**简介：**Bloom filter 是由 Howard Bloom 在 1970 年提出的二进制向量数据结构，它具有很好的空间和时间效率，被用来检测一个元素是不是集合中的一个成员。如果检测结果为是，该元素不一定在集合中；但如果检测结果为否，该元素一定不在集合中。因此Bloom filter具有100%的召回率。这样每个检测请求返回有“在集合内（可能错误）”和“不在集合内（绝对不在集合内）”两种情况，可见 Bloom filter 是牺牲了正确率和时间以节省空间。

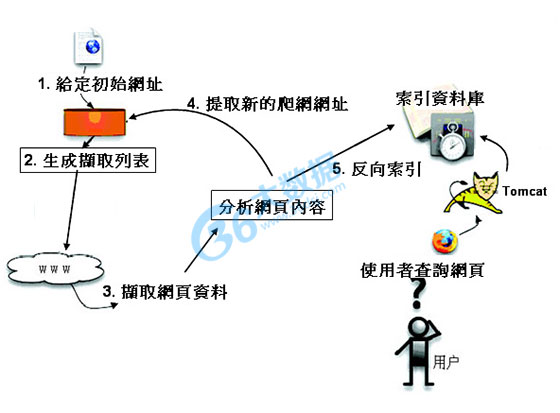
Bloom filter 优点就是它的插入和查询时间都是常数，另外它查询元素却不保存元素本身，具有良好的安全性。

搜索引擎

**一、Nutch**

**简介：**Nutch 是一个开源Java 实现的搜索引擎。它提供了我们运行自己的搜索引擎所需的全部工具。包括全文搜索和Web爬虫。

尽管Web搜索是漫游Internet的基本要求, 但是现有web搜索引擎的数目却在下降. 并且这很有可能进一步演变成为一个公司垄断了几乎所有的web搜索为其谋取商业利益.这显然 不利于广大Internet用户.



Nutch为我们提供了这样一个不同的选择. 相对于那些商用的搜索引擎, Nutch作为开放源代码 搜索引擎将会更加透明, 从而更值得大家信赖. 现在所有主要的搜索引擎都采用私有的排序算法, 而不会解释为什么一个网页会排在一个特定的位置. 除此之外, 有的搜索引擎依照网站所付的 费用, 而不是根据它们本身的价值进行排序. 与它们不同, Nucth没有什么需要隐瞒, 也没有 动机去扭曲搜索的结果. Nutch将尽自己最大的努力为用户提供最好的搜索结果.

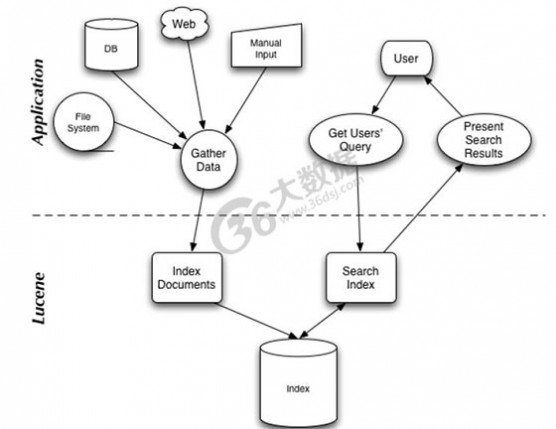
Nutch目前最新的版本为version v2.2.1。

官网：<https://nutch.apache.org/>

**二、Lucene**

**开发者**：Doug Cutting（Hadoop之父，你懂的）

**简介：**Lucene是apache软件基金会4 jakarta项目组的一个子项目，是一个开放源代码的全文检索引擎工具包，即它不是一个完整的全文检索引擎，而是一个全文检索引擎的架构，提供了完整的查询引擎和索引引擎，部分文本分析引擎（英文与德文两种西方语言）。Lucene的目的是为软件开发人员提供一个简单易用的工具包，以方便的在目标系统中实现全文检索的功能，或者是以此为基础建立起完整的全文检索引擎。

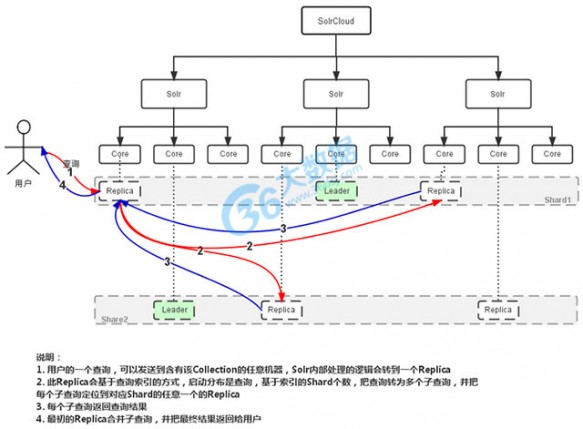


官网：<http://lucene.apache.org/>

**三、SolrCloud**

**简介**：SolrCloud是Solr4.0版本以后基于Solr和Zookeeper的分布式搜索方案。SolrCloud是Solr的基于Zookeeper一种部署方式。Solr可以以多种方式部署，例如单机方式，多机Master-Slaver方式。

**原理图**：



**SolrCloud有几个特色功能：**

集中式的配置信息使用ZK进行集中配置。启动时可以指定把Solr的相关配置文件上传

Zookeeper，多机器共用。这些ZK中的配置不会再拿到本地缓存，Solr直接读取ZK中的配置信息。配置文件的变动，所有机器都可以感知到。另外，Solr的一些任务也是通过ZK作为媒介发布的。目的是为了容错。接收到任务，但在执行任务时崩溃的机器，在重启后，或者集群选出候选者时，可以再次执行这个未完成的任务。

自动容错SolrCloud对索引分片，并对每个分片创建多个Replication。每个Replication都可以对外提供服务。一个Replication挂掉不会影响索引服务。更强大的是，它还能自动的在其它机器上帮你把失败机器上的索引Replication重建并投入使用。

近实时搜索立即推送式的replication（也支持慢推送）。可以在秒内检索到新加入索引。

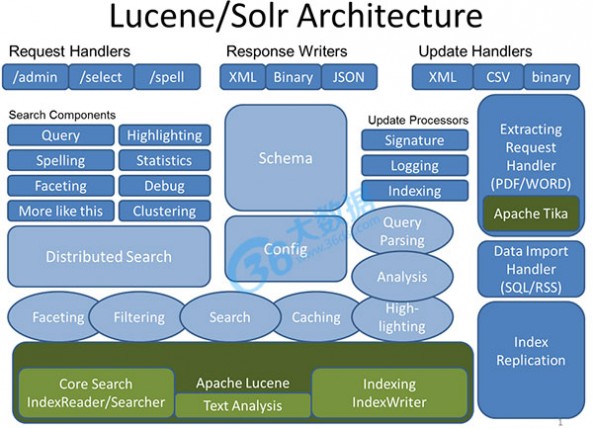
查询时自动负载均衡SolrCloud索引的多个Replication可以分布在多台机器上，均衡查询压力。如果查询压力大，可以通过扩展机器，增加Replication来减缓。

自动分发的索引和索引分片发送文档到任何节点，它都会转发到正确节点。

事务日志事务日志确保更新无丢失，即使文档没有索引到磁盘。

**四、Solr**

**简介：**Solr是一个独立的企业级搜索应用服务器，它对外提供类似于Web-service的API接口。用户可以通过http请求，向搜索引擎服务器提交一定格式的XML文件，生成索引；也可以通过Http Get操作提出查找请求，并得到XML格式的返回结果。



Solr是一个高性能，采用Java5开发，基于Lucene的全文搜索服务器。同时对其进行了扩展，提供了比Lucene更为丰富的查询语言，同时实现了可配置、可扩展并对查询性能进行了优化，并且提供了一个完善的功能管理界面，是一款非常优秀的全文搜索引擎。

**官网：**<https://lucene.apache.org/solr/>

**五、ElasticSearch**

**简介**：ElasticSearch是一个基于Lucene的搜索服务器。它提供了一个分布式多用户能力的全文搜索引擎，基于RESTful web接口。Elasticsearch是用Java开发的，并作为Apache许可条款下的开放源码发布，是第二最流行的企业搜索引擎。设计用于云计算中，能够达到实时搜索，稳定，可靠，快速，安装使用方便。

**官网：**[http://www.elasticsearch.org/](http://www.elasticsearch.org/http:/www.elasticsearch.cn/)

**六、Sphinx**

**简介**：Sphinx是一个基于SQL的全文检索引擎，可以结合MySQL,PostgreSQL做全文搜索，它可以提供比数据库本身更专业的搜索功能，使得应用程序更容易实现专业化的全文检索。Sphinx特别为一些脚本语言设计搜索API接口，如PHP,Python,Perl,Ruby等，同时为MySQL也设计了一个存储引擎插件。

Sphinx单一索引最大可包含1亿条记录，在1千万条记录情况下的查询速度为0.x秒（毫秒级）。Sphinx创建索引的速度为：创建100万条记录的索引只需 3～4分钟，创建1000万条记录的索引可以在50分钟内完成，而只包含最新10万条记录的增量索引，重建一次只需几十秒。

**官网**：[http://sphinxsearch.com](http://sphinxsearch.com/)

**七、SenseiDB**

**贡献者**：linkedin

**简介**：SenseiDB是一个NoSQL数据库，它专注于高更新率以及复杂半结构化搜索查询。熟悉Lucene和Solor的用户会发现，SenseiDB背后有许多似曾相识的概念。SenseiDB部署在多节点集群中，其中每个节点可以包括N块数据片。Apache Zookeeper用于管理节点，它能够保持现有配置，并可以将任意改动（如拓扑修改）传输到整个节点群中。SenseiDB集群还需要一种模式用于定义将要使用的数据模型。

从SenseiDB集群中获取数据的唯一方法是通过Gateways（它 没有“INSERT”方法）。每个集群都连接到一个单一gateway。你需要了解很重要的一点是，由于SenseiDB本身没法处理原子性 （Atomicity）和隔离性（Isolation），因此只能通过外部在gateway层进行限制。另外，gateway必须确保数据流按照预期的方 式运作。内置的gateway有以下几种形式：

* 来自文件
* 来自JMS队列
* 通过JDBC
* 来自Apache Kafka

**官网**：[http://senseidb.com](http://senseidb.com/)

**数据挖掘**

**一、Mahout**

[](http://www.36dsj.com/wp-content/uploads/2015/03/2012.jpg)

**简介**：Apache Mahout 是 Apache Software Foundation (ASF) 开发的一个全新的开源项目，其主要目标是创建一些可伸缩的机器学习算法，供开发人员在 Apache 在许可下免费使用。该项目已经发展到了它的最二个年头，目前只有一个公共发行版。Mahout 包含许多实现，包括集群、分类、CP 和进化程序。此外，通过使用 Apache Hadoop 库，Mahout 可以有效地扩展到云中。

虽然在开源领域中相对较为年轻，但 Mahout 已经提供了大量功能，特别是在集群和 CF 方面。**Mahout 的主要特性包括：**

1. Taste CF。Taste 是 Sean Owen 在 SourceForge 上发起的一个针对 CF 的开源项目，并在 2008 年被赠予 Mahout。
2. 一些支持 Map-Reduce 的集群实现包括 k-Means、模糊 k-Means、Canopy、Dirichlet 和 Mean-Shift。
3. Distributed Naive Bayes 和 Complementary Naive Bayes 分类实现。
4. 针对进化编程的分布式适用性功能。
5. Matrix 和矢量库。
6. 上述算法的示例。

**官网**：<http://mahout.apache.org/>

**Iaas**

IaaS（Infrastructure as a Service），即基础设施即服务。

**一、OpenStack**

**简介：**OpenStack是一个由NASA（美国国家航空航天局）和Rackspace合作研发并发起的，以Apache许可证授权的自由软件和开放源代码项目。

OpenStack是一个开源的云计算管理平台项目，由几个主要的组件组合起来完成具体工作。OpenStack支持几乎所有类型的云环境，项目目标是提供实施简单、可大规模扩展、丰富、标准统一的云计算管理平台。OpenStack通过各种互补的服务提供了基础设施即服务（IaaS）的解决方案，每个服务提供API以进行集成。



6个核心项目：Nova（计算，Compute），Swift（对象存储，Object），Glance（镜像，Image），Keystone（身份，Identity），Horizon（自助门户，Dashboard），Quantum & Melange（网络&地址管理），另外还有若干社区项目，如Rackspace（负载均衡）、Rackspace（关系型数据库）。

**相关阅读：**

[什么是OpenStack？](http://www.36dsj.com/archives/19596)

[成功部署OpenStack的十大要点](http://www.36dsj.com/archives/22179)

**官网**：<https://www.openstack.org/>

**二、Docker**

**贡献者**：dotCloud



**简介**：Docker 是一个开源的应用容器引擎，让开发者可以打包他们的应用以及依赖包到一个可移植的容器中，然后发布到任何流行的 Linux 机器上，也可以实现虚拟化。容器是完全使用沙箱机制，相互之间不会有任何接口（类似 iPhone 的 app）。几乎没有性能开销,可以很容易地在机器和数据中心中运行。最重要的是,他们不依赖于任何语言、框架或包括系统。

**官网**：<http://www.docker.io/>

**三、Kubernetes**

**贡献者**：Google

**简介**：Kubernetes是Google开源的容器集群管理系统。它构建Ddocker技术之上，为容器化的应用提供资源调度、部署运行、服务发现、扩容缩容等整一套功能，本质上可看作是基于容器技术的mini-PaaS平台。

Kubernetes从另一个角度对资源进行抽象，它让开发人员和管理人员共同着眼于服务的行为和性能的提升，而不是仅仅关注对单一的组件或者是基础资源。

那么Kubernetes集群到底提供了哪些单一容器所没有功能?它主要关注的是对服务级别的控制而并非仅仅是对容器级别的控制，Kubernetes提供了一种“机智”的管理方式，它将服务看成一个整体。在Kubernete的解决方案中，一个服务甚至可以自我扩展，自我诊断，并且容易升级。例如，在Google中，我们使用机器学习技术来保证每个运行的服务的当前状态都是最高效的。

代码托管：<https://github.com/GoogleCloudPlatform/kubernetes/>

**四、Imctfy**

**贡献者**：Google

**简介**：Google开源了自己所用Linux容器系统的开源版本lmctfy，读音为lem-kut-fee。包括一个C++库（使用了C++11，文档可以参考头文件）和命令行界面。目前的版本是0.1，只提供了CPU与内存隔离。项目还在密集开发中。

mctfy本身是针对某些特定使用场景设计和实现的，目前拥有一台机器上所有容器时运行情况最好，不推荐与LXC和其他容器系统一起使用（虽然也可行）。已在Ubuntu 12.04+和Ubuntu 3.3与3.8内核上测试。

**代码托管**：<https://github.com/google/Imctfy/>

**监控管理**

**一、Dapper**

**贡献者：**Google

**简介：**Dapper是一个轻量的ORM(对象关系映射（英语：Object Relational Mapping，简称ORM，或O/RM，或O/R mapping）。并不单纯的是一个DBHelper.因为在Dapper中数据其实就是一个对象。Dapper扩展与IDbConnection上，所以事实上它的倾入性很低。我用了StructureMap。如果不喜欢可以自己更换，或者自己实现下。

代码就一个SqlMapper.cs文件,主要是IDbConnection的扩展方法，编译后就40K的一个很小的dll。

**特性：**

1. Dapper很快。Dapper的速度接近与IDataReader。
2. Dapper支持主流数据库 Mysql,SqlLite,Mssql2000,Mssql2005,Oracle等一系列的数据库
3. 支持多表并联的对象。支持一对多 多对多的关系，并且没侵入性。
4. 原理通过Emit反射IDataReader的序列队列，来快速的得到和产生对象
5. Dapper语法十分简单。并且无须迁就数据库的设计

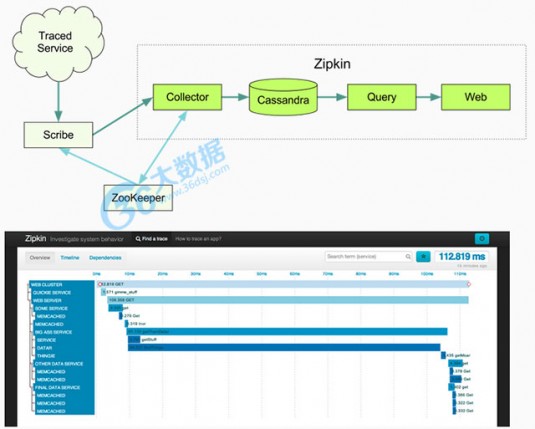
**官方站点** <http://code.google.com/p/dapper-dot-net/>

**代码托管**：<http://bigbully.github.io/Dapper-translation/>

**二、Zipkin**

**贡献者：**Twitter

**简介：**Zipkin （分布式跟踪系统）是 Twitter 的一个开源项目，允许开发者收集 Twitter 各个服务上的监控数据，并提供查询接口。该系统让开发者可通过一个 Web 前端轻松的收集和分析数据，例如用户每次请求服务的处理时间等，可方便的监测系统中存在的瓶颈。



**官方网站**：<http://twitter.github.io/zipkin/>

**代码托管**：<https://github.com/twitter/zipkin/>

End.