分布式存储系统架构

分布式文件系统

- GFS：闭源
- HDFS

> Quora Hadoop is suitable for **batch processing** and **data mining** tasks - more specifically **MapReduce** tasks. Thus, HDFS was not designed to support **real-time processing** in mind (though Hive has effectively addressed this issue, but fundamentally, HDFS does not support real-time processing). From my experience, HDFS is also more stable than Ceph in real-production environment. I haven't really worked with actual business projects, but I tested both systems on a 100-node cluster and Ceph was definitely way more unstable (required more tweaking and fixing) than HDFS.

- Ceph
    - 无中心元数据

> Quora Probably the most notable feature of Ceph is that it does **not rely on central metadata service** to locate data. It uses **CRUSH algorithm** to calculate location of data http://ceph.com/papers/weil-crus... .
>
> The drawback of this approach is that cluster topology changes lead to data movement to maintain CRUSH contracts.
>
> The data stored in Ceph cluster can be either replicated or erasure-coded to provide resiliency.
>
> Within single cluster it is possible to have distributed block storage, POSIX like file system and Swift/S3 object store.

    - 灵活性

> Quora On the other hand, if your data is **in the format of files**, for example when you need to organize your data into folders, then Ceph is better at handling a large-scale file system than HDFS. Notice that HDFS's design relies on a single central namenode, and this is also the single-point of failure in Hadoop. In fact, Ceph scales better than HDFS if you plan to have a large complex directory structure. Also, Ceph is **more flexible** (and thus more difficult) to configure in different cluster environments.

> 知乎 2018 说实话，这个基本没有可比性～虽然 Sage 在最初设计 Ceph 的时候是作为一个分布式存储系统，也就是说其实 CephFS 才是最初的构想和设计(题外音)， 但可以看到，后面 Ceph 俨然已经发展为一整套存储解决方案，上层能够提供对象存储(RGW)、块存储(RBD)和 CephFS，可以说是一套适合各种场景，非常灵活，非常有可发挥空间的存储解决方案～

- 曾经的问题：与Docker配合（2017）
  - 或者，现在是否可以尝试让 ceph 与wasm container/unikernel 配合？
  - 项目地址： https://github.com/ceph/ceph-container
  - 做一些移植

> Quora One fundamental problem is that Ceph can't be used directly as a docker solution; instead, you need something else to format your block store into a mountable storage volume (CephFS may be a future alternative, but it was not mature enough at the time we implemented Ceph).
>
> One of our team-members wrote a Ceph-mesos framework: **ceph-on-mesos** which we combined with the Docker volume plugin: **rbd-docker-plugin** . This enabled our Docker containers to mount and utilize Ceph volumes.
>
> Making Ceph work with Docker has been more trouble than it's worth due to a myriad of bugs / issues we have encountered (exclusive locks not being released, port conflicts, random outages, and multiple node failures with permanent data loss). We will probably replace Ceph with another Docker volume plugin, but this is yet to be determined. Infinit and Flocker are among the alternatives we are currently evaluating.

- 统一存储架构

> ARTICLE 2015 Ceph 最初设计的 RADOS 是为其实现一个高性能的文件系统服务的，并没有考虑对于块设备、对象存储的支持，也就没有什么 RBD、RADOS GateWay，跟别提 OpenStack 和 qemu 之类的了。但谁想无心插柳柳成荫，由于 RADOS 出色的设计和独立简洁的访问接口，再加上 Sage 敏锐的眼光，Ceph 社区果断推出了用于支持云计算的分布式块存储 RBD 和分布式对象存储 RADOS GateWay，并将开发中心全面转向云计算领域。

- CephFS

> ARTICLE 2015 **最先的CephFS文件系统已经不再是开发重点，甚至开发已经陷入了停滞状态。而与虚拟化相关的RBD、RGW则成了发展重点，成为发展最快的模块。**但是从代码中仍然能够看到各种遗迹，似乎在告诉后来人这段饶了一个大弯的历史。
>
> 正所谓成也萧何败萧何，**Ceph想把CephFS模块做到足够强大，甚至是最强大，但强大的同时也意味着不菲的代价。**元数据动态子树分区、目录分片、快照、权限控制、IOPS优化、故障恢复、分布式缓存、强弱一致性控制，这些高大上的名词背后都意味着复杂的工程性任务，更不要说将这些叠加在一起。很多时候，叠加不是想加，而是相乘的关系。**最终的结果就是整个MDS的工程难度已经超过了可以掌控的程度，无法做出足够成熟、稳定的系统。**

- 性能问题

> ARTICLE 2015 **最先的CephFS文件系统已经不再是开发重点，甚至开发已经陷入了停滞状态。而与虚拟化相关的RBD、RGW则成了发展重点，成为发展最快的模块。** Ceph的性能总的来说还是不错的，基本上能发挥出物理硬件的性能，但是存在以下几个隐患：

> 1）**数据双倍写入。**Ceph本地存储接口（FileStore）为了支持事务，引入了日志（Journal）机制。所有的写入操作都需要先写入日志（XFS模式下），然后再写入本地文件系统。简单来说就是**一份数据需要写两遍，日志+本地文件系统。**这就造成了在大规模连续IO的情况下，实际上磁盘输出的吞吐量只有其物理性能的一半。
>
> 2）**IO路径过长。**这个问题在Ceph的客户端和服务器端都存在。以osd为例，一个IO需要经过message、OSD、FileJournal、FileStore多个模块才能完成，每个模块之间都涉及到队列和线程切换，部分模块在对IO进行处理时还要进行内存拷贝，导致整体性能不高。
>
> 3）**对高性能硬件的支持有待改进。****Ceph最开始是为HDD设计的，没有充分考虑全SSD，甚至更先进的PCIe SSD和NVRAM的情况NVRAM。**导致这些硬件的物理性能在Ceph中无法充分发挥出来，特别是延迟和IOPS，受比较大的影响。

  - Bluestore: 直接操作硬件，改进性能

> **ARTICLE 2017** BlueStore is a clean implementation of our internal ObjectStore interface from first principles, motivated specifically by the workloads we expect. BlueStore is built on top of a raw underlying block device (or block devices). It embeds the RocksDB key/value database for managing its internal metadata. A small internal adapter component called BlueFS implements a file-system-like interface that provides just enough functionality to allow RocksDB to store its “files” and share the same raw device(s) with BlueStore.

- GlusterFS

> **CSDN 2017** Ceph和Gluster是Red Hat旗下的成熟的开源存储产品，Ceph与Gluster在原理上有着本质上的不同。Ceph基于一个名为RADOS的对象存储系统，使用一系列API将数据以块(block)、文件(file)和对象(object)的形式展现。Ceph存储系统的拓扑结构围绕着副本与信息分布，这使得该系统能够有效保障数据的完整性。
>
> Gluster描述为Scale-out NAS和对象存储系统。它使用一个Hash算法来计算数据在存储池中的存放位置，这点跟Ceph很类似。在Gluster中，所有的存储服务器使用Hash算法完成对特定数据实体的定位。于是数据可以很容易的复制，并且没有中心元数据分布式存储无单点故障且不易造成访问瓶颈，这种单点在早期Hadoop上出现，对性能和可靠性造成较大影响。
>
> 然而，实践表明Ceph访问存储的不同方法使其成为更流行的技术。更多的公司正在考虑Ceph技术而不是GlusterFS，而且GlusterFS仍然与Red Hat密切相关。例如，SUSE还没有GlusterFS的商业实施，而Ceph已经被开源社区广泛采用，市场上有各种不同的产品。在某种意义上来说，Ceph确实已经胜过GlusterFS。

> **Reddit 2019** Ceph is object first. For most of Ceph's history, it was object layered on top of a native file system (xfs usually) and ran very slowly relative to the raw IOPs/throughput of the underlying hardware. Ceph has recently released "bluestore" which attempts to let Ceph handle writing data straight to disk without the intermediate FS. EVERYTHING Ceph does is around object. it's entirely based around RADOS, their own internal object layer. Ceph-FS is layered on top of object. Ceph block is layered on top of object, Ceph Object? Unless your application speaks native RADOS, which most don't, you're using a translation layer to go from swift/S3 to RADOS. Ceph, based on the documentation, is a swiss-army chainsaw, complete with add-on toothpick and umbrella. Setup is therefore not necessarily easy. They have made some strides with this, but it's not simple.

> Gluster-- Gluster is basically the opposite of Ceph architecturally. Gluster is a file store first, last, and most of the middle. A drunken monkey can set up Gluster on anything that has a folder and can have the code compiled for it, including containers, vms, cloud machines, whatever. Want to create a Gluster volume? Here are the 2 steps (assuming you've already installed the package on 2/3 nodes and you want replica 3):
>
> 1. gluster peer probe othernode(x however many othernodes)
> 2. gluster volume create replica 3 node1:/path node2:/path node3:/path
>
> That's it, you're done. you have a Gluster replica 3 volume.Warning! If you do this, you're going to have a shitty time. Gluster should be run with correct FS options, LVM thinpools, snapshots, etc. RH recommends XFS, not ZFS, because of certain bugs. EXT is definitely not recommended, and idk about btrfs or other weird ones.

- OpenStack Swift
- Lustre
- GridFS
- mogileFS
- TFS
- FastDFS

分布式键值对式 NoSQL 数据库

- Memcached
- Redis

分布式列存储 NoSQL 数据库

- Bigtable

分布式文档存储 NoSQL 数据库

- MongoDB

NewSQL 数据库

- MegaStore
- OceanBase

分布式计算系统架构

分布式批处理计算(MapReduce-base)

- Hadoop
  - 局限性

> **知乎 2015**
>
> - 抽象层次低，需要手工编写代码来完成，使用上难以上手。
> - 只提供两个操作，Map和Reduce，表达力欠缺。
> - 一个Job只有Map和Reduce两个阶段（Phase），复杂的计算需要大量的Job完成，Job之间的依赖关系是由开发者自己管理的。
> - 处理逻辑隐藏在代码细节中，没有整体逻辑
> - 中间结果也放在HDFS文件系统中
> - ReduceTask需要等待所有MapTask都完成后才可以开始
> - 时延高，只适用Batch数据处理，对于交互式数据处理，实时数据处理的支持不够
> - 对于迭代式数据处理性能比较差

- Spark
  - 与Hadoop相比的优势为：

> **知乎 2015**
>
> - 抽象层次低，需要手工编写代码来完成，使用上难以上手。=> 基于RDD的抽象，实数据处理逻辑的代码非常简短。。
>
> - 只提供两个操作，Map和Reduce，表达力欠缺。=>提供很多转换和动作，很多基本操作如Join，GroupBy已经在RDD转换和动作中实现。
> - 一个Job只有Map和Reduce两个阶段（Phase），复杂的计算需要大量的Job完成，Job之间的依赖关系是由开发者自己管理的。=>一个Job可以包含RDD的多个转换操作，在调度时可以生成多个阶段（Stage），而且如果多个map操作的RDD的分区不变，是可以放在同一个Task中进行。
> - 处理逻辑隐藏在代码细节中，没有整体逻辑=>在Scala中，通过匿名函数和高阶函数，RDD的转换支持流式API，可以提供处理逻辑的整体视图。代码不包含具体操作的实现细节，逻辑更清晰。
> - 中间结果也放在HDFS文件系统中=>中间结果放在内存中，内存放不下了会写入本地磁盘，而不是HDFS。ReduceTask需要等待所有MapTask都完成后才可以开始=> 分区相同的转换构成流水线放在一个Task中运行，分区不同的转换需要Shuffle，被划分到不同的Stage中，需要等待前面的Stage完成后才可以开始。
> - 时延高，只适用Batch数据处理，对于交互式数据处理，实时数据处理的支持不够=>通过将流拆成小的batch提供Discretized Stream处理流数据。
> - 对于迭代式数据处理性能比较差=>通过在内存中缓存数据，提高迭代式计算的性能。

- ....

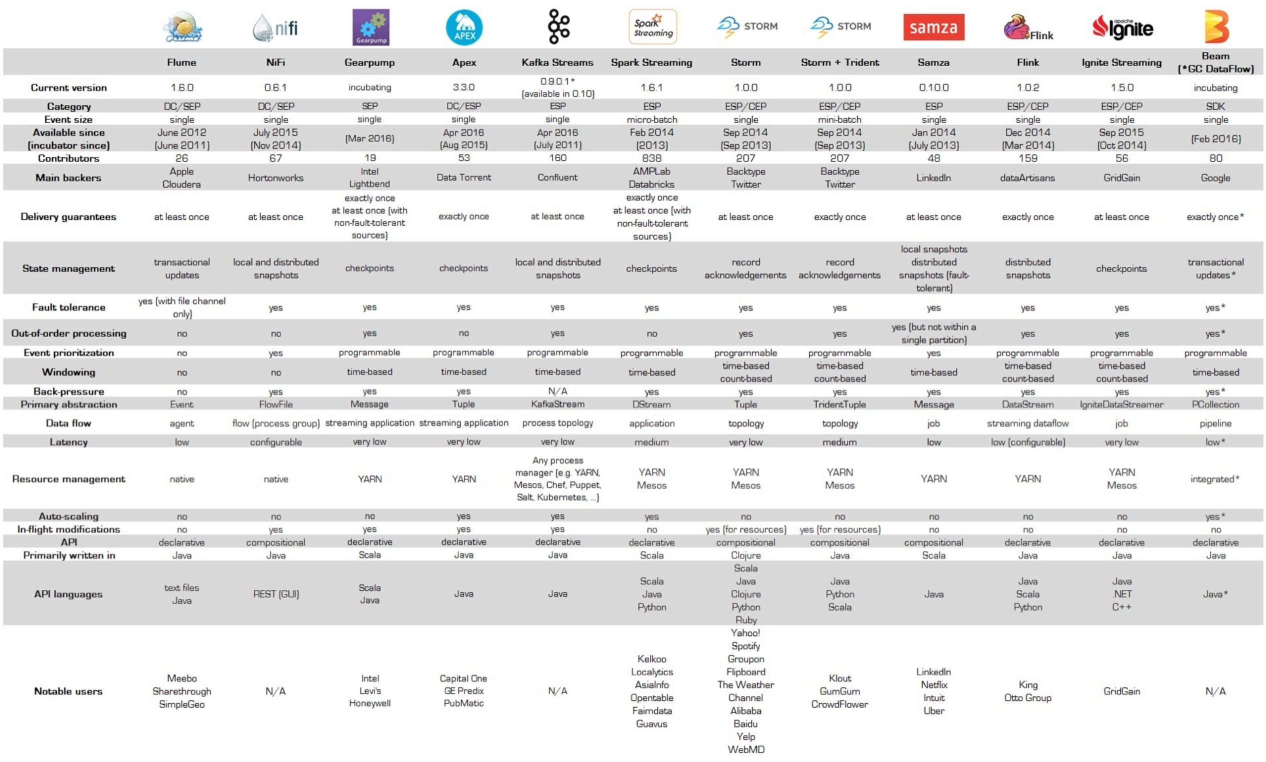
分布式流处理计算

- Storm
- Spark Streaming
- Flink

知乎 2015

TUTORIAL 2018

TUTORIAL 2018



	Flume	NiFi	Gearpump	Apex	Kafka Streams	Spark Streaming	Storm	Storm + Trident	Samza	Flink	Ignite Streaming	Beam (*GC DataFlow)
Current version	1.6.0	0.6.1	incubating	3.3.0	0.9.0.1* (available in 0.10)	1.6.1	1.0.0	1.0.0	0.10.0	1.0.2	1.5.0	incubating
Category	DC/SEP	DC/SEP	SEP	DC/ESP	ESP	ESP	ESP/CEP	ESP/CEP	ESP	ESP/CEP	ESP/CEP	SDK
Event size	single	single	single	single	single	micro-batch	single	mini-batch	single	single	single	single
Available since **(incubator since)**	June 2012 (June 2011)	July 2015 (Nov 2014)	(Mar 2016)	Apr 2016 (Aug 2015)	Apr 2016 (July 2011)	Feb 2014 (2013)	Sep 2014 (Sep 2013)	Sep 2014 (Sep 2013)	Jan 2014 (July 2013)	Dec 2014 (Mar 2014)	Sep 2015 (Oct 2014)	(Feb 2016)
Contributors	26	67	19	53	160	838	207	207	48	159	56	80
Main backers	Apple Cloudera	Hortonworks	Intel Lightbend	Data Torrent	Confluent	AMPLab Databricks	Backtype Twitter	Backtype Twitter	LinkedIn	dataArtisans	GridGain	Google
Delivery guarantees	at least once	at least once	exactly once at least once (with non-fault-tolerant sources)	exactly once	at least once	exactly once at least once (with non-fault-tolerant sources)	at least once	exactly once	at least once	exactly once	at least once	exactly once*
State management	transactional updates	local and distributed snapshots	checkpoints	checkpoints	local and distributed snapshots	checkpoints	record acknowledgements	record acknowledgements	local snapshots distributed snapshots (fault-tolerant)	distributed snapshots	checkpoints	transactional updates*
Fault tolerance	yes (with file channel only)	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes*
Out-of-order processing	no	no	yes	no	yes	no	yes	yes	yes (but not within a single partition)	yes	yes	yes*
Event prioritization	no	yes	programmable	programmable	programmable	programmable	programmable	programmable	yes	programmable	programmable	programmable
Windowing	no	no	time-based	time-based	time-based	time-based	time-based count-based	time-based count-based	time-based	time-based count-based	time-based count-based	time-based
Back-pressure	no	yes	yes	yes	N/A	yes	yes	yes	yes	yes	yes	yes*
Primary abstraction	Event	FlowFile	Message	Tuple	KafkaStream	DStream	Tuple	TridentTuple	Message	DataStream	IgniteDataStreamer	PCollection
Data flow	agent	flow (process group)	streaming application	streaming application	process topology	application	topology	topology	job	streaming dataflow	job	pipeline
Latency	low	configurable	very low	very low	very low	medium	very low	medium	low	low (configurable)	very low	low*
Resource management	native	native	YARN	YARN	Any process manager (e.g. YARN, Mesos, Chef, Puppet, Salt, Kubernetes, ...)	YARN Mesos	YARN Mesos	YARN Mesos	YARN	YARN	YARN Mesos	integrated*
Auto-scaling	no	no	no	yes	yes	yes	no	no	no	no	no	yes*
In-flight modifications	no	yes	yes	yes	yes	no	yes (for resources)	yes (for resources)	no	no	no	no
API	declarative	compositional	declarative	declarative	declarative	declarative	compositional	compositional	compositional	declarative	declarative	declarative
Primarily written in	Java	Java	Scala	Java	Java	Scala	Clojure	Java	Scala	Java	Java	Java
API languages	text files Java	REST (GUI)	Scala Java	Java	Java	Scala Java Python	Scala Java Clojure Python Ruby	Java Python Scala	Java	Java Scala Python	Java .NET C++	Java*
Notable users	Meebo Sharethrough SimpleGeo	N/A	Intel Levi's Honeywell	Capital One GE Predix PubMatic	N/A	Kelkoo Localytics AsiaInfo Opentable Faimdata Guavus	Yahoo! Spotify Groupon Flipboard The Weather Channel Alibaba Baidu Yelp WebMD	Klout GumGum CrowdFlower	LinkedIn Netflix Intuit Uber	King Otto Group	GridGain	N/A

虚拟化与云计算

容器

- Docker
- Kubernetes
- containerd

Unikernel

Webassembly

开发工具

- Emscripten： compiler - mainly for the web
- WASI： Standard Interface
- Wasmtime: Mozilla Runtime
- WAPM：相对比较成熟的包管理器，容器等需要可以办法与之兼容
- WebAssembly.sh：一个可以在浏览器中运行的shell， 运行， 不过估计环境是js，程序启动的效率不高。

Wasm 无服务器计算

- serverless-wasm ： 长期没有更新（Latest commit 1fab6e2 on 4 Jun 2018）
    - 缺少一个抽象文件系统
- Faasm：
    - 功能即服务FaaS，
    - 用这个东西是不是已经实现到Kubernetes上的移植呢？
- kubeless ：这个似乎可以支持各种无服务器计算的runtime。

Wasm 容器

- IceCore
    - Archived 估计被作者废弃

> WEBSITE
>
> - Get WebAssembly code running √
> - Provide native interfaces with permission control √
> - TCP networking √
> - Blocking file I/O √
> - Asynchronous file I/O
> - UDP networking
> - Built-in high-level abstraction for HTTP services
> - Profiling & statistics
> - Manager API & management script
> - Runtime application migration across machines

- containerd-wasm： https://github.com/dmcgowan/containerd-wasm
    - 确实缺乏文档，作者本人搞 containerd 去了，没空管这个闲项目

Wasm Unikernel

- nanovm ：基于Emscripten，生成一个html，运用wasmer运行。可以考虑用其结合wapm打造可分发的程序
- cervus 、 nebulet： 有待进一步调查

Wasm 应用移植

似乎 emscripten 还算能实现比较好的移植。

图计算的移植 √