博睿数据海量小文件分布式存储系统

—BRFS

# 项目背景

随着博睿公司业务的迅速发展，推出了很多新的产品，如Bonree SDK、Bonree Browser、Bonree Server等），这些产品每天都会回收大量的如网页截图、网络抓包、代码堆栈快照等数据文件需要进行存储。由于此类文件体积小但数量多的特点，导致在存储上会遇到很多问题，如使用系统原生文件系统进行存储，则会遇到Linux索引节点（inode）耗尽导致磁盘存储空间无法有效利用的问题，同时也无法原生支持分布式及高可用的要求。但如果采用HDFS或HBASE方案存储，则一方面硬件资源开销较大，NameNode内存开销过大的问题，同时该方案读写效率也不足够高效，尤其当内部进行块文件合并时集群I/O压力过大的时候，读写效率更是难以保证。

为了解决以上这些问题，针对海量（10亿个文件以上）小文件（普遍大小在1KB-50KB）的分布式存储的场景，我们设计了BRFS（Bonree Distributed FileSystem）系统，以满足博睿公司目前各产品线对海量小文件数据存储需求。目前此项目已在Github上开源（https://github.com/zhangnianli/BRFS），欢迎Star、Fork和PR。

# BRFS是什么？

BRFS是一个主要针对海量小文件（快照、图片、语音等）存储而设计的高可用、高性能、易扩展的分布式文件系统。BRFS对存储的文件格式没有限制和要求，任何格式的文件都会以byte字节流的方式进行存储。此系统功能不仅包含了现有其他开源产品具备的权限控制、数据增删读、多副本备份、数据检验等基础功能，同时我们还设计了独特且高效的副本自动迁移和平衡策略、多应用数据隔离且独立配置，系统资源管理插件、集群状态可视化监控与报警等功能。

一般来说，一个BRFS系统包含如下三个部分：

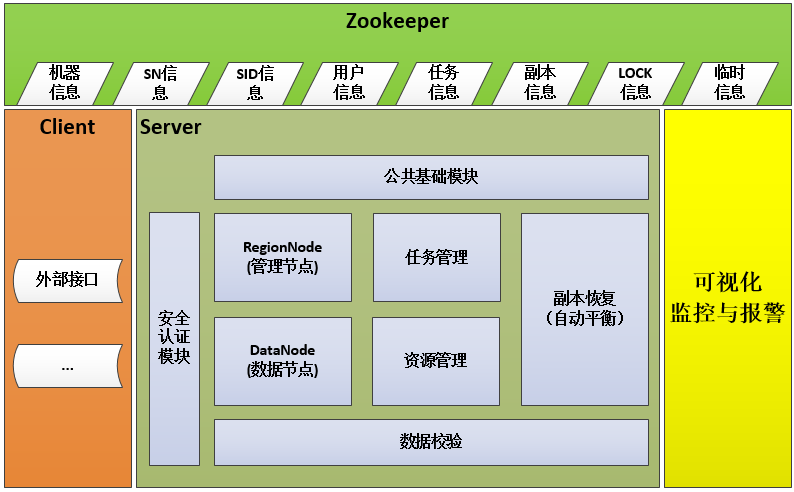
1. **FS\_Server.jar**：在集群每个节点上运行的核心服务模块。在模块运行时会启动若干ReginNode(管理节点)和DataNode（数据节点）进程来提供服务。其中ReginNode进程主要职责是管理存储域元信息、管理数据节点、把用户数据分配到不同的数据节点上进行处理；DataNode进程主要职责是用户数据文件的写入和读取、副本自动平衡恢复、执行定时任务（副本数校验、CRC校验、数据删除、数据归并）执行等。
2. **FS\_ResouceManager.jar**：系统资源管理模块，用于实时收集和监控集群各节点资源负载情况，以支持系统可根据节点负载情况分配资源，解决各节点资源利用和负载不均衡问题。BRFS系统内部默认提供了一组资源管理的策略，主要包含CPU、内存、I/O、磁盘容量等负载指标。目前此模块采用可热插拔的设计方式，但如果用户有特殊需求，可自定义此插件，自行实现集群资源的分配和管理。
3. **server.properties.example**：用于后台服务运行时所有的关键控制参数的默认值配置，如果想变更参数值，可以复制一个名为**server.properties的**文件，并把需要修改的属性和值添加到此文件中即可，程序运行时**server.properties**文件中的配置的参数值会覆盖**server.properties.example**文件中参数的默认值。

同时，用户如需调用BRFS服务，则需要在工程中引入**FS\_Client.jar，**并在代码中调用相关的接口对BRFS系统进行操作；

BRFS分布式文件系统接收的数据形式可以是快照、图片或者任何以byte数组方式进行存储的数据文件。

# BRFS的整体架构

BRFS整体架构如下图所示：



【**BRFS整体架构图**】

此系统主要由Zookeeper、Client、Server以及可视化监控UI等四部分模块组成。

BRFS使用Zookeeper来管理集群服务，同步节点状态，确保服务高可用。Zookeeper上保存具体信息包括：机器节点信息、Storage信息、SID信息、任务信息、副本信息、lock锁信息、用户信息、临时信息等元数据信息。

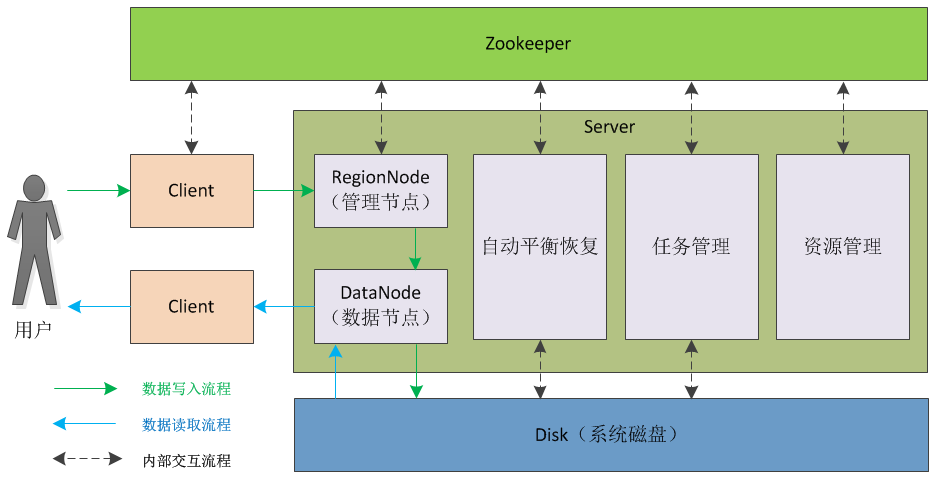
Client即用户客户端，它是以Jar的形式被用户在用户工程代码中引用，并通过调用其相应的接口对BRFS进行数据添加、修改、读取等操作。

Server即后台集群服务，包括RegionNode和DataNode两组进程。主要功能包含了安全认证、副本管理、磁盘管理、任务(副本数校验、crc校验、删除、归并等)管理、节点资源管理和副本自动平衡与恢复、可视化监控与报警等功能模块。后台服务运行的相关进程是通过zookeeper进行管理的。

可视化监控与报警,它是把集群节点上存储的文件情况、后台任务执行情况、资源负载情况、服务运行状态等都通过可视化监控直观的观察到，当某些状态达到阀值后可以自动触发报警。

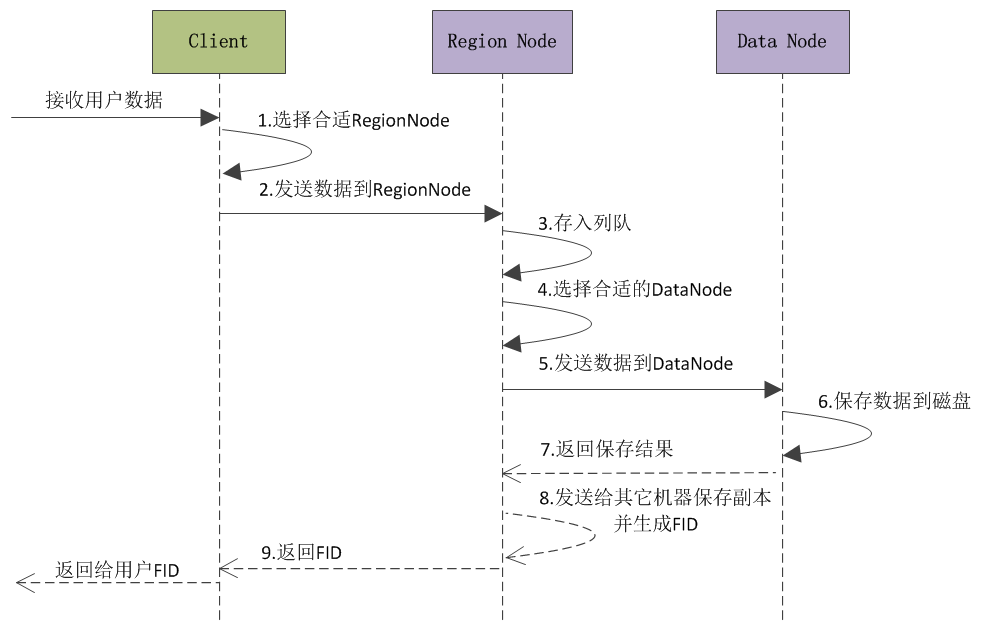
# BRFS的核心运行机制

BRFS分布式文件系统整体运行机机制如下图所示：



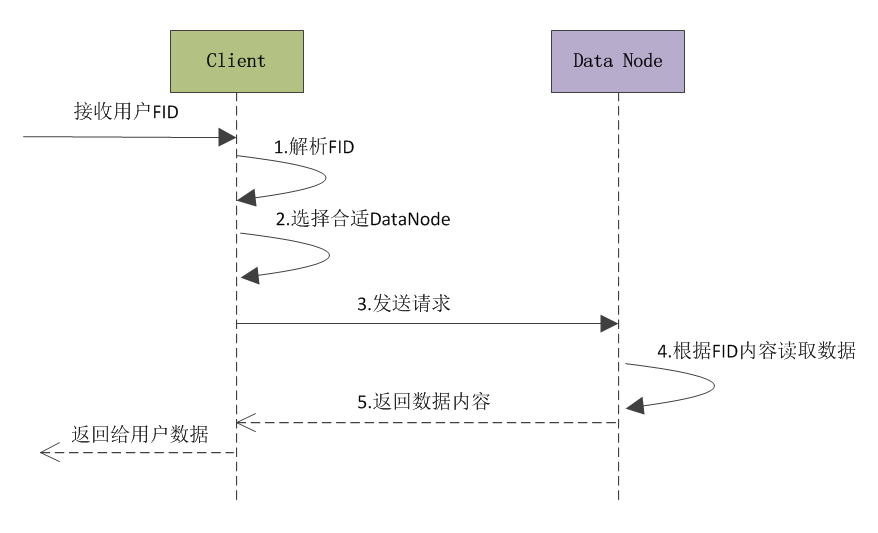
【**BRFS整体运行机制图**】

## 写数据机制



注：选择RegionNode和选择DataNode的策略支持随机选择和根据集群机器负载（CPU、内存、IO、磁盘容量）选择两种策略。

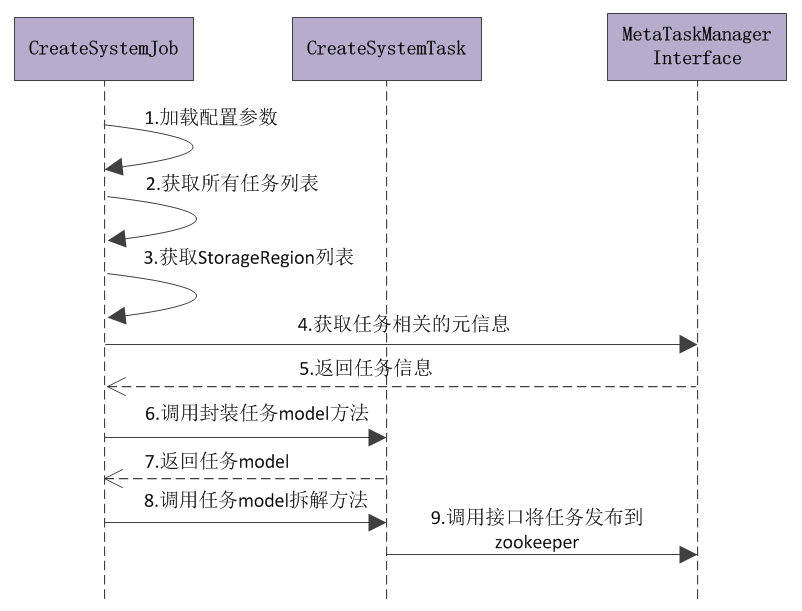
## 读取数据机制



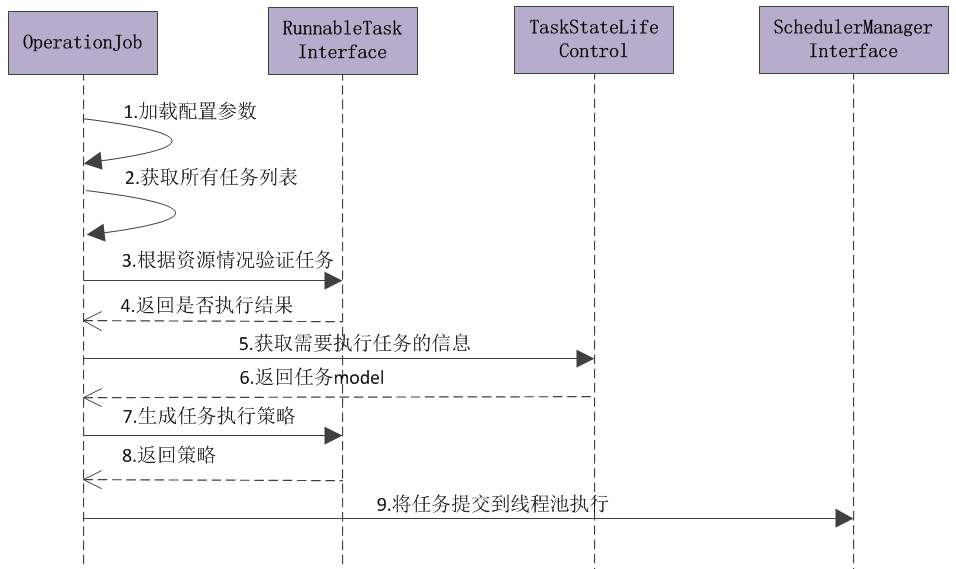
注：选择DataNode是根据解析FID后得到的相关信息及配合路由规则来获取的。

## 定时任务机制

* **任务创建机制**



* **任务执行机制**

****

注：任务执行时会受机器资源负载情况影响，负载高时相应的任务会停止工作，避免机器压力过大而造成其它负面的影响。

## 副本自动平衡恢复机制及算法原理

ServerID即服务标识，简称SID，每个服务在启动的时候，必须拥有一组服务标识。ServerID分为first level ID、secondlevel ID和Virtual ID。first level ID是1级服务标识，标识特定机器的某个服务，该serverID会随着服务第一次启动而生成，之后不会再变；secondlevel ID是2级服务标识，在创建SN（storageName）的时候，会为每个服务进行分配，可以理解为某个SN在某个服务上的编号，数据迁移时，需要做相应的变化，若被迁移的服务重新启动，则该ID会过期，并重新生成；Virtual ID是针对当副本数多于可用Server时，来进行虚拟存储时所用的编号，可以理解为该标识为未来的某个2级服务标识的映射。

**算法原理**

我们可以从下面这个故事中梳理清楚副本自动平衡恢复的原理：

假如某公司目前有A,B,C,D四个员工，这四个员工分别持有各种不同的消息。某一天，A员工要离职，那么公司会让A员工把自己所知的消息告诉其他人。因为A持有的消息可能比较多，所以只告诉其中的一个人，那么他的工作压力会变大。因此，公司会通过消息的特征，将A知道的消息尽可能的均匀的告诉其他的三个人。公司只需要记录接手A消息的人，即[A->B,C,D]。此时，若来了两个新员工E,F。过了不久，C也离职了。那么C同样会把消息尽可能的均匀的告诉其他人，公司同样需要记录接手C消息的人。即，[C->B,D,E,F]。

若某一天，客户来找A询问一个消息，但是A已经离职了，那么公司会通过[A->B,C,D]以及消息特征来寻找接手的人。若该消息命中B，因B还在，所以B可以为客户服务。若命中为C，因C也已离职，则继续通过[C->B,D,E,F]以及消息特征来寻找接手的人。若此时命中的是E，因E存在，则由E来为客户进行服务。

在上面的故事中，我们知道接手记录是非常重要的。因为每一次交接，都会记录一次交接记录，方便根据不同的消息特征，来分配不同的人来接手。后面在找由那一个人接手的具体任务时，也有据可循。

遵循上述的原理，SID即为员工，我们将新增或离开的机器节点的SID记录存储在zookeeper中，形成一套类似路由规则，当觖发了副本平衡或恢复机制时，程序会按照路由规则对相应的副本数据进行迁移操作。

## 集群节点扩展机制

BRFS所有集群节点的相关信息都需要注册到zookeeper上，由zookeeper统一管理。在新加的机器上启动BRFS的服务后，该服务会把自身的相关信息注册到zookeeper中保存。当客户端再有新数据写入时，服务节点选择逻辑（根据机器的负载情况选择合适存储数据的节点）会把新加的机器添加进来，并优先在新机器上保存数据，同时会把副本相对均衡的保存到其它集群节点上。

为了更好的磁盘利用率，扩展机器需要在集群节点上所有机器的磁盘占用在80%以内增加新节点。否则会出现新加的机器上磁盘空间还有很多，但其它集群节点上已经没有磁盘空间保存副本了，所以会导致副本保存失败。

## 数据存储结构

BRFS分布式文件系统的数据编码结构是参考了protobuf编码结构后制定的，对于数字类型采用了Varint这种紧凑数字的方法来表示的；对于字符串采用了Key-Pair这样的结构来表示的。系统中需要编码的包含FID和文件内容两部分：

* **FID**：一条数据或小文件的唯一标识，由一系列编码过的数字和字符串组成，用户需要提供相应的FID才可以读取数据。

FID组成 ：

Header：FID的头信息

StorageName：存储空间名称(也可有来区分不同的业务)

uuid：文件唯一标识

time：时间串(精确到分钟)

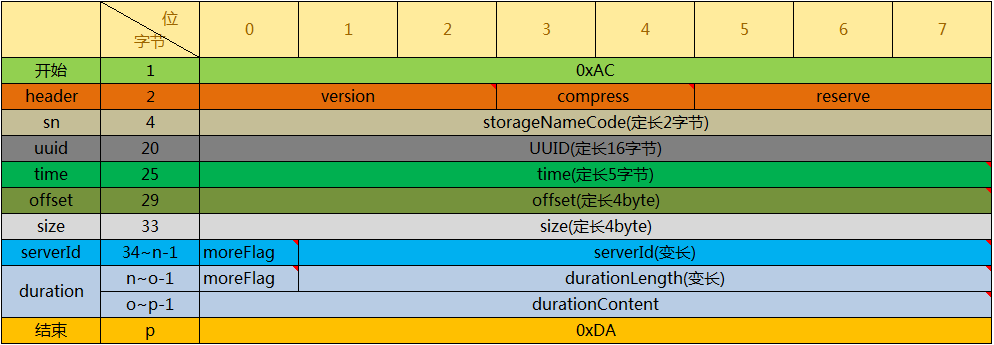
ServerId：服务标识

Offset：当前文件所在大文件里的偏移量

Size：当前文件的长度

Duration：存储数据的时间目录的粒度

FID编码格式如下图：



* **文件内容**：用来保存小文件的存储结构，配合FID可以对文件进行读写操作。

文件内容组成：

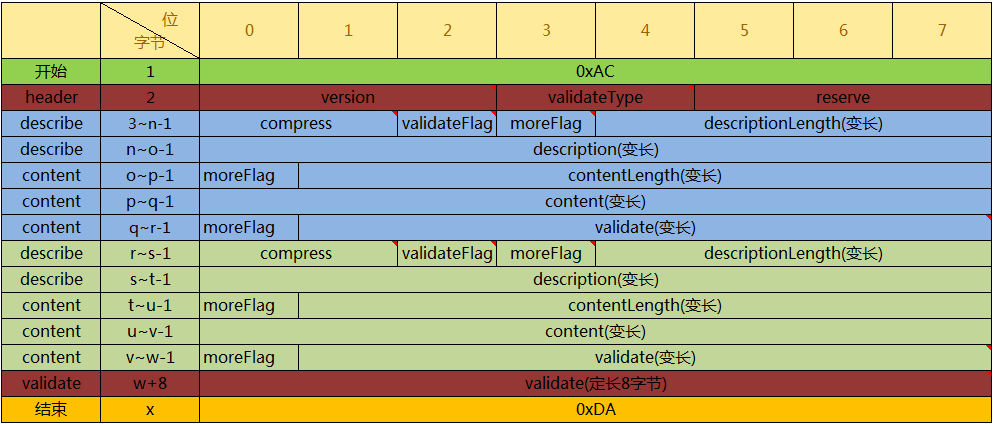
Header：整个大文件的头信息；

Describe：一个小文件的描述信息，由用户自定义；

Content：一条文件的内容，单个小文件也可以有检验码；

Validate： 整个大文件的检验码；

文件内容编码格式如下图：



上图编码中描述的是一个合并后的大文件包含了两个小文件。

# 为什么要用BRFS？

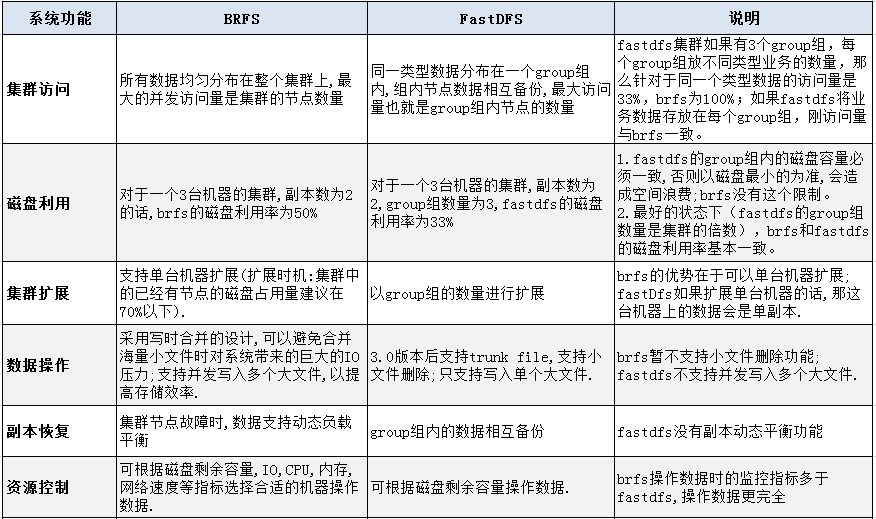
BRFS与FastDFS比较，部署更简单，性能更高效，尤其在数据写入方面更是有数倍的性能提升。

BRFS在设计上有以下几个特点：

1. 数据写入采用写时合并的策略，可以避免文件合并时IO瓶颈问题，并且支持同时打开多个文件并发写入，所以在数据写入性能上有很大的提升（在“性能对比”节点做详细说明）；
2. 增加了数据自动平衡恢复的容错机制，当集群有节点挂掉的情况下，该节点上的数据文件的副本将会被自动备份到其它机器节点上，以保证数据不会丢失，这样可在不需要人工干预的情况下很好保证数据安全。
3. 在集群机器节点的资源负载方面，增加了对CPU、内存、IO、磁盘容量等指标的状态监控，可以在有读写数据、后台任务、副本平衡等这些操作时很好的避开高负载的机器，从而实现了机器资源相对均衡的使用，不会出现因为某台机器的负载过高而造成服务不可用的现象，同时这些指标还可以用来做报警的阀值，让使用者更加清晰准确的了解集群节点的相关资源状态，更好的达到防患于未然的目的，比如集群节点服务出现异常、集群节点的机器负载是否过高、机器的磁盘空间不足等。
4. 引入应用分区的概念，即用户可对不同业务数据进行不同的权限控制、存储隔离、过期等策略设置等，灵活满足业务需求。
5. 扩容方式极为灵活，用户可横向扩展任意个节点，而无需成倍扩容；

## 与FASTDFS的功能对比

以开源产品FastDFS为参考，我们在二者功能设计上做了如下对比：



从上面对比可知，FastDFS引入了一个group组的概念，可以简化数据副本的备份策略，但在并发访问、写入、磁盘利用及集群扩展方面就带来一定局限。另外，在资源控制和副本动态恢复和平衡等方面BRFS的在设计上也是优于FastDFS的。

## 压力测试性能对比

以下为压力测试的具体环境：

集群规模：由2台物理机器组成集群，数据保存一副本；

单台配置：CPU：4核、内存：18G、 磁盘：STAT盘 4T 7.2K；

网络：千兆；

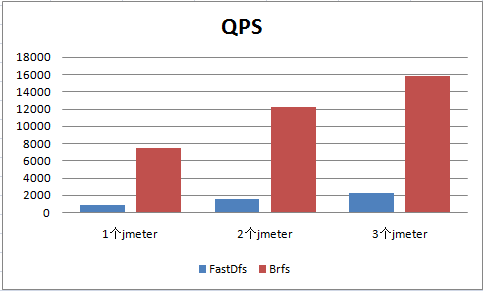
压测结果指标：QPS、CPU、MEMORY、IO。

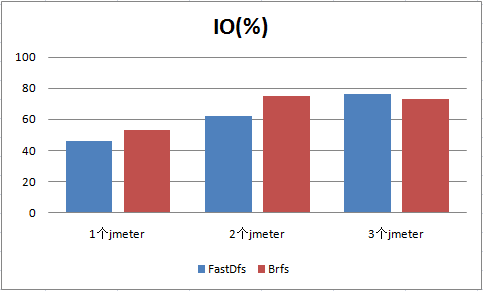
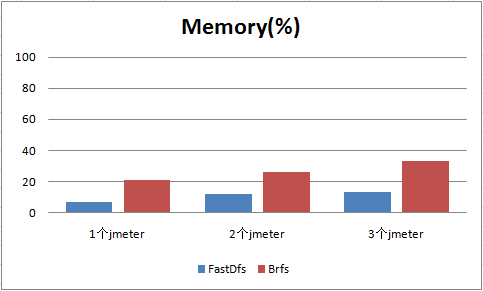
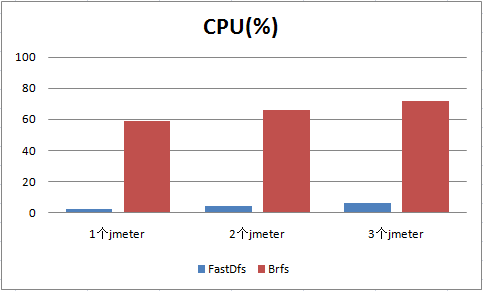
压测方式：分别使用1个/2个/3个/5个/8个/10个JMeter客户端对BRFS和FastDFS进行压测，每个JMeter客户端开启50个并发线程；

目标文件：随机生成1KB和5KB数据文件；

### 1KB数据文件

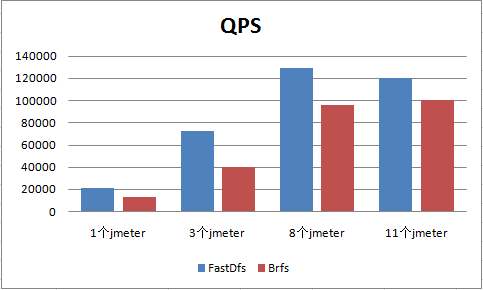
#### 数据写入

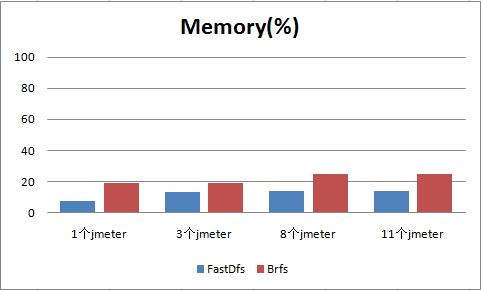
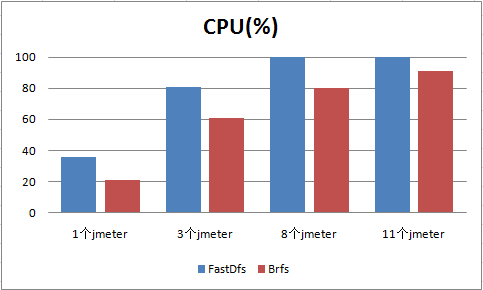


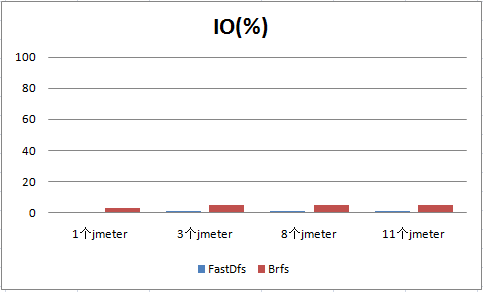


##### **此轮压测结论**：通过上面压测对比来看，在处理海量1KB的数据文件并发写入时，BRFS的写入性能要明显优于FASTDFS，平均效率是FastDFS的5倍以上。FASTDFS在写入数据时，硬件资源明显没有被充分利用。

#### 数据读取







##### 此轮压测结论：

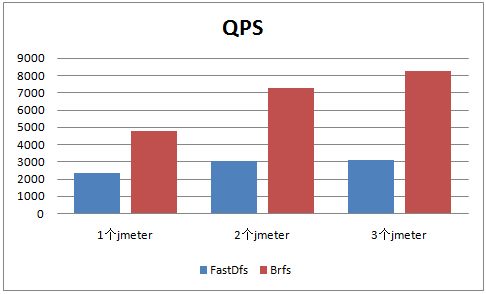
**FastDFS**：8个Jmeter客户端压测时IO占用100%，带宽也打满，到了机器的极限，QPS达到了12.9万，但当增加到11个Jmeter客户端时QPS反而降到了12万，说明已经到了极限；

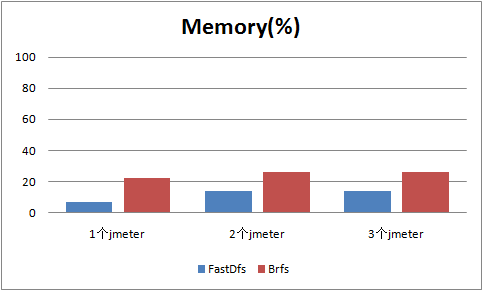
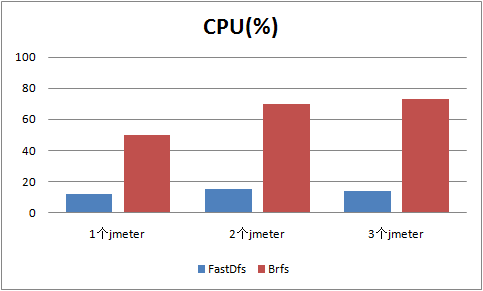
**BRFS**：在11 jemter压测时，CPU占用91%，带宽占用53MB，QPS达到10万；由于压测机器数量不足，BRFS机器资源还没有测到极限，QPS应该还有上升的空间；

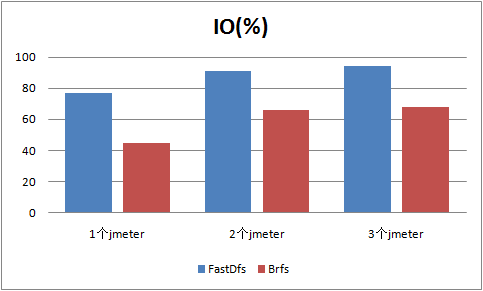
通过此轮对比来看，在处理海量1KB大小的数据文件并发读取时，BRFS的读取性能基本与FastDFS持平。

### 5KB数据文件

#### 数据写入







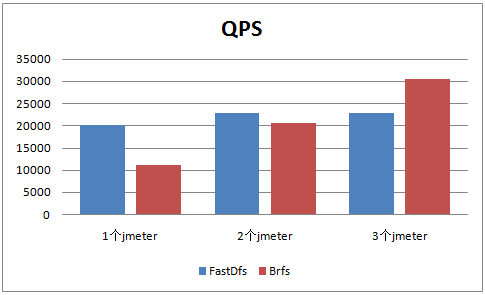
##### 此轮压测结论：

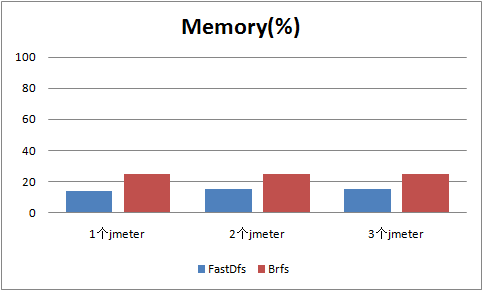
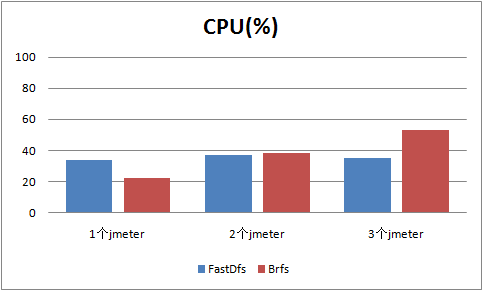
FastDFS：1个Jmeter压测时I/O占用接近80%，增加到3个时I/O基本满了；

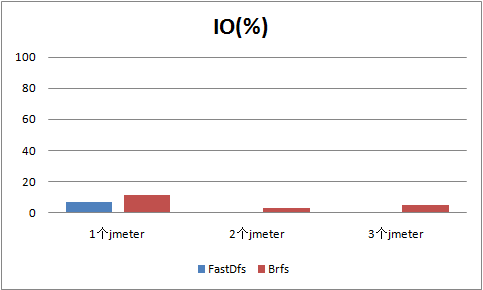
BRFS：在1个、2个、3个jemter压测时，QPS均要明显优于FastDFS；同时BRFS的QPS、TPS、I/O、CPU等指标是随着Jmeter压力增加而线性提高的；

通过上述对比来看，在处理海量5KB文件并发写入时，BRFS的写入性能要明显优于FASTDFS，性能提升约2-3倍。FASTDFS在写入数据时，明显硬件资源没有被合理的利用起来。

#### 读数据







##### 此轮压测结论：

**FastDFS**：1个jmeter压测时的QPS要优于BRFS，带宽也基本达到瓶颈；增加到3个时QPS没有明显提高；

**BRFS**：QPS、I/O、CPU等指标随着Jmeter个数增加而线性提高的，在3个时的QPS要优于FastDFS；在3个Jmeter时未到带宽瓶颈，再增加jmeter时QPS应还会有提高的空间；

通过上述对比来看，在处理海量5KB文件的并发读取时，BRFS的读取性能会随着并发数的提高性能优势逐渐体现出来，BRFS在读取性能方面是占有优势的。

# 怎样使用BRFS？

BRFS系统依赖第三方组件极少，除依赖JDK等基础组件外，其他组件只依赖Zookeeper服务进行集群状态同步，且核心服务只有两个Jar文件，因此安装部署极为简单，以下是部署安装的具体步骤：

1. 安装基础组件，主要包括Zookeeper、JDK等；
2. 根据业务需要配置server.properties文件；
3. 启动各节点**FS\_Server.jar**服务；
4. 通过测试客户端，测试读写功能是否正常。

BRFS分布式文件系统目前推出仅是第一个版本，并已在博睿公司所有产品线上实际运行，大大降低了博睿公司对于海量小文件的存储成本。同时，BRFS下个版本还将进行两方面优化，一是对大文件存储的支持和优化，不再区分大文件或是小文件，而定位为海量非结构化数据分布式存储系统；二是解决目前所有类似服务都存在的需要用户保存文件FID的问题，对于用户来说，存储海量文件的FID同样是个很大的开销，BRFS将开发一款全新的基于磁盘的key-value系统，以解决海量FID和文件元数据关联存储的问题，请持续关注。