Apache HTrace

Apache HTrace是Cloudera开源的分布式系统跟踪框架，类似于Dapper，为Hadoop提供请求跟踪和性能分析，其使用HDFS及HBase作为底层存储系统。

# HTrace的使用

在Hadoop中，可以通过hadoop trace命令来使用trace功能：

*hadoop trace*

*-add [-class classname] [-Ckey=value] [-Ckey2=value2] ...*

*//通过配置类型来增加Span Receiver*

*-help:*

*-host [hostname:port} //配置检查的主机和端口*

*-list: //查看系统中当前的Span Recievers*

*-remove [id] //删除系统中的Reciver*

*-principal:*

命令的使用如下：

1. 添加Span Receiver

*hadoop trace -add -class org.apache.htrace.core.LocalFileSpanReceiver -Chadoop.htrace.local.file.span.receiver.path=/tmp/htrace.out -host fys1.cmss.com:8020*

*Added trace span receiver 1 with configuration hadoop.htrace.local.file.span.receiver.path = /tmp/htrace.out*

1. 查看Span Receiver

*$ hadoop trace -list -host fys1.cmss.com:8020*

*ID CLASS*

*1 org.apache.htrace.core.LocalFileSpanReceiver*

1. 删除Span Reciver

*$ hadoop trace -remove 1 -host fys1.cmss.com:8020*

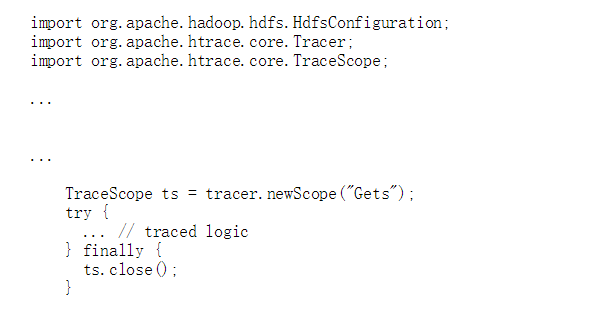
*Removed trace span receiver 1*

*$ hadoop trace -list -host fys1.cmss.com:8020*

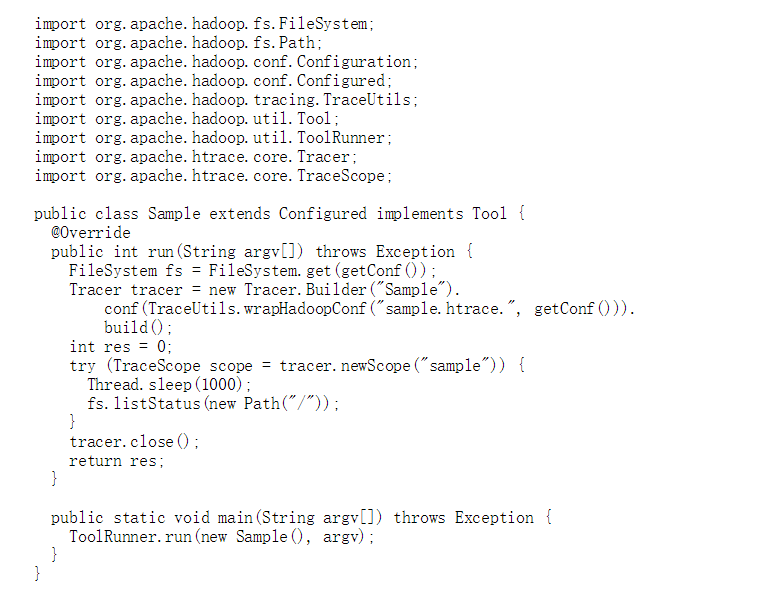
*[no span receivers found]*

1. 使用示例

为了实现Trace，将执行代码封装在Tracing Span中，当执行时Trace消息会发送给Rpc请求中，Trace Scop的设置如下：



示例程序如下：



编译及执行如下：

*$ javac -cp `hadoop classpath` Sample.java*

*$ java -cp .:`hadoop classpath` Sample \*

*-Dsample.htrace.span.receiver.classes=LocalFileSpanReceiver \*

*-Dsample.htrace.sampler.classes=AlwaysSampler*

结果输出

*8/09/21 11:55:17 DEBUG core.Sampler$Builder: Creating new instance of org.apache.htrace.core.AlwaysSampler...*

*18/09/21 11:55:17 DEBUG core.Tracer: sampler.classes = AlwaysSampler; loaded AlwaysSampler*

*18/09/21 11:55:17 DEBUG core.SpanReceiver$Builder: Creating new instance of org.apache.htrace.core.LocalFileSpanReceiver...*

*18/09/21 11:55:17 DEBUG core.LocalFileSpanReceiver: Created new LocalFileSpanReceiver with path = /tmp/17208, capacity = 5000*

*18/09/21 11:55:17 DEBUG core.Tracer: span.receiver.classes = LocalFileSpanReceiver; loaded LocalFileSpanReceiver*

*18/09/21 11:55:19 DEBUG ipc.Client: The ping interval is 60000 ms.*

*18/09/21 11:55:19 DEBUG ipc.Client: Connecting to fys1.cmss.com/10.139.4.98:8020*

*......*

打开文件/tmp/17208，可以看到Trace的输出结果



1. 在Hadoop中可以通过配置，对FileSystem Shell进行Trace，使用命令如下：

*$ hdfs dfs -Dfs.shell.htrace.span.receiver.classes=LocalFileSpanReceiver \*

*-Dfs.shell.htrace.sampler.classes=AlwaysSampler \*

*-ls /*

输出结果与上述的结果类似。也可以通过在hadoop conf中添加配置来实现Shell的Trace

*<property>*

*<name>hadoop.htrace.span.receiver.classes</name>*

*<value>LocalFileSpanReceiver</value>*

*</property>*

*<property>trac*

*<name>fs.client.htrace.sampler.classes</name>*

*<value>ProbabilitySampler</value>*

*</property>*

*<property>*

*<name>fs.client.htrace.sampler.fraction</name>*

*<value>0.01</value>*

*</property>*

# HTrace源码分析

在上述示例中当客户端对某个操作进行追踪时，在该操作之前添加TraceScope，listPaths的添加示例如下：

*public DirectoryListing listPaths(String src, byte[] startAfter,*

*boolean needLocation) throws IOException {*

*try (TraceScope ignored = newPathTraceScope("listPaths", src)) {*

*return namenode.getListing(src, startAfter, needLocation);*

*}*

*}*

在newPathTraceScope中，TraceScope的初始化如下：

*TraceScope newPathTraceScope(String description, String path) {*

*TraceScope scope = tracer.newScope(description);*

*if (path != null) {*

*scope.addKVAnnotation("path", path);*

*}*

*return scope;*

*}*

TraceScope对应分布式跟踪链中的一个节点

## 相关概念

Apache HTrace借鉴了Dapper的思想，下面是几个典型概念及实现

1. Span，分布式跟踪的基本工作单元，其对应链路调用（RPC、DB连接等），类图如下：



各字段含义如下表所示：

|  |  |
| --- | --- |
| 字段 | 描述 |
| begin | 服务调用的开始时间 |
| end | 服务调用的结束时间 |
| description | 描述 |
| parents[] | 父服务调用 |
| spanId | 调用的ID号 |
| traceInfo:Map<String,String> | 追踪信息 |
| tracerId | 追踪ID |
| timeline:List<TimelineAnnotation> | Timeline信息 |

Span的示例如下：

{

*"a": "2270ee3edb072c1e472b0e214521808a",*

*"b": 1537503015754,*

*"e": 1537503016596,*

*"d": "FileSystem#createFileSystem",*

*"r": "FSClient/10.139.4.98",*

*"p":* [

*"2269ee3edb072c1e7dbd872f9bf1073f"*

]*,*

*"n":* {

*"scheme": "hdfs"*

}

}

1. TraceScope

TraceScope对象进行Span的生命周期管理，其与Span对象一一相关，当Scope关闭时将Span传递给SpanReceiver进行处理。HTrace通过Tracer使用Thread-Local Data来感知TraceScope的状态，其构建如下：

*TraceScope computationScope = tracer.newScope("CalculateFoo");*

*try {*

*calculateFoo();*

*} finally {*

*computationScope.close();*

*}*

在calculateFoo中创建Trace Span，将CalculateFoo作为parent Span。因为使用Thread-Local数据，因此不需要显示的设置Span之间的父子关系。

1. Tracers

Tracer对应一个调用链，其使用newScope方法来创建Trace Scope对象。分布式跟踪系统很难追踪每个操作，每个追踪产生的数据量比较大，因此可以使用Sampler来减少数据量。在创建Scope方法时，如果Tracer有active Span，则会创建一个子Span，从而最终创建一个调用链。

在分布式系统中会有很多的Tracer，每个Tracker都有自己的配置，其类似于Log4j对象，例如为NN和DN分别创建Tracer，用户可以分别控制其调用链。在HTrace中包含许多的Wrapper，例如TraceRunnable、TraceCallable和TraceExecutorService，辅助Tracer来创建Trace Span。相关类图如下：



4）SpanReceivers

HTracer是可插拔的框架，在运行时可配置SpanReciever，SpanReciever用户将Span持久化存储，类图如下所示：



## **跟踪树**

使用FsShell查询文件的跟踪树结构如下：

ls

*+--- FileSystem#createFileSystem*

*+--- Globber#glob*

*| +---- GetFileInfo*

*| +---- ClientNamenodeProtocol#GetFileInfo*

*| +---- ClientProtocol#GetFileInfo*

*+--- listPaths*

*+---- ClientNamenodeProtocol#getListing*

*+---- ClientProtocol#getListing*

如下图所示：

****

DFSClient与NameNode之间通过RPC进行交互，Trace SpanId定义在Rpc库中，如下所示：

*message RPCTraceInfoProto {*

*optional int64 traceId = 1; // parentIdHigh*

*optional int64 parentId = 2; // parentIdLow*

*}*

1. 客户端生成RpcQuestHeader的代码如下：

*public static RpcRequestHeaderProto makeRpcRequestHeader(RPC.RpcKind rpcKind,*

*RpcRequestHeaderProto.OperationProto operation, int callId,*

*int retryCount, byte[] uuid) {*

*RpcRequestHeaderProto.Builder result = RpcRequestHeaderProto.newBuilder();*

*......*

*Span span = Tracer.getCurrentSpan();*

*if (span != null) {*

*result.setTraceInfo(RPCTraceInfoProto.newBuilder()*

*.setTraceId(span.getSpanId().getHigh())*

*.setParentId(span.getSpanId().getLow())*

*.build());*

*}*

*return result.build();*

*}*

当启用Trace时，通过获取当前Trace信息，并将SpanId传递给Server。

2）NameNode的Rpc Server接收客户端传递的RpcRequestHeaderProto后进行处理，如下所示：

*Server#processRpcRequest(RpcRequestHeaderProto)*

*TraceScope traceScope = null;*

*if (header.hasTraceInfo()) {*

*if (tracer != null) {*

*// If the incoming RPC included tracing info, always continue the*

*// trace*

*SpanId parentSpanId = new SpanId(*

*header.getTraceInfo().getTraceId(),*

*header.getTraceInfo().getParentId());*

*traceScope = tracer.newScope(*

*RpcClientUtil.toTraceName(rpcRequest.toString()),*

*parentSpanId);*

*traceScope.detach();*

*}*

*}*

其对应的Rpc Request的Span信息如下：

*{*

*"a": "e1b3096de6bb5a905124103f9d9a5a63",*

*"b": 1537614613378,*

*"e": 1537614613381,*

*"d": "ClientProtocol#getListing",*

*"r": "NameNode/10.139.4.98",*

*"p": [*

*"e1b3096de6bb5a909fac082ab7be63d9"*

*],*

*"t": [*

*{*

*"t": 1537614613379,*

*"m": "called"*

*}*

*]*

*}*

https://github.com/apache/incubator-htrace

http://bigbully.github.io/Dapper-translation/

https://issues.apache.org/jira/browse/HDFS-5274

http://htrace.incubator.apache.org/

https://hadoop.apache.org/docs/r3.1.1/hadoop-project-dist/hadoop-common/Tracing.html