Coprocessor 简介

Authors: Trend Micro Hadoop Group: Mingjie Lai, Eugene Koontz, Andrew Purtell

HBase非常有效的整合了分布式计算的MapReduce模型和存储在其表上的数据。但是在许多情况下（例如，简单的加法或累积操作诸如求和，计数，等等），将计算过程提交到服务器，让服务器直接操作数据，可以避免通信开销，进而高性能的HBase带来更大的性能提升。

在0.92之前，想要向HBase添加自定义功能，而不继承HBase中的相关基类是不可能的。由于Java没有多重继承的支持，这需要实现一个单独的类来实现扩展功能，同时还需要提供所有基础代码的实现。当面对多个扩展功能时，这样的实现会变得很脆弱。谁继承谁？协处理器(coprocessor)提供了更加灵活的扩展模型。

在这篇文章中，我将介绍HBase的新特性协处理器Coprocessor。协处理器是灵活的通用扩展框架，在HBase的服务器进程直接处理分布式计算。以下会讨论它是什么，它是如何工作的，以及如何开发协处理器扩展。

HBase协处理器的灵感来源于Google BigTable协处理器。Jeff Dean 在LADIS ’09上发表的报告(<http://www.scribd.com/doc/21631448/Dean-Keynote-Ladis2009> ,page 66-67)，其中提到了一些协处理器的基本概念，谷歌用这一思想实现了具有并行处理能力的BigTable。它们具有以下特点：

* 每个子表服务器中的每个子表都可以运行任意代码
  + 为客户端提供高层调用接口
  + 调用需要指定行或行的区间，然后协处理器客户端库解析他们到对应的子表服务器
* 跨多行的调用自动分解为多个并行的RPC
* 提供了构建分布式服务的非常灵活的模型
* 自动分片，负载均衡，应用路由

回到HBase中来，我们非常需要比Hadoop MapReduce更高效的计算并行性。并且，可以提供构建在其上的令人兴奋的新功能，例如二级索引，复杂的过滤（倒推谓词，push down predicate），访问控制。HBase协处理器受到BigTable协处理器的启发，但有不同的实现细节。我们已经建立了一个包含库和运行时环境的框架，利用这个框架可以在HBase的Region服务器和master服务进程中执行用户代码。而谷歌协处理器，在子表服务器的进程地址空间之外运行协处理器。（HBase也考虑了部署协处理器代码到服务器进程空间之外的选项。见<https://issues.apache.org/jira/browse/HBASE-4047>。）

协处理器可以装载到全局region server承载的所有的表(table)以及region上，这通常被称为系统协处理器。管理员也可以指定哪个协处理器需要加载到一个表(table)的所有region上，这被称为表协处理器。

为了给潜在的协处理器行为提供足够的灵活性，扩展框架提供了两类不同的方面。一个是观察者（Observer），类似于传统数据库中的触发器，另一种是端点（Endpoint），动态RPC端点，类似于存储过程。

观察者

观察者的思想是通过覆盖协处理器框架提供的唤醒方法(upcall method)，将用户代码插入到HBase中去。 当某些事件发生时，HBase会执行回调函数。协处理器的框架处理所有回调函数的调用细节。协处理器只需要插入额外的或替换的功能。

在当前的HBase 0.92中，提供了三个观察者接口：

RegionObserver：提供了数据操作(Get, Put, Delete, Scan等)事件的钩子点。有一个RegionObserver协处理器的每一个表的区域和它们可以使被约束到该区域的观测范围的一个实例。

WALObserver：提供了预写日志（WAL）相关操作的钩子点。WALObserver提供了监视或拦截WAL写入和重建事件的方法。WALObserver运行在WAL处理的上下文。每个region server有一个这样的环境。

MasterObserver提供了DDL类型操作的钩子点，例如创建，删除，修改表等。MasterObserver在HBase master的上下文中运行。

可以加载多个观察者到同一个位置（region，master或WAL）来扩展更多的功能。这些观察者按优先级链接到一起顺序执行。有没有办法来防止协处理器实现者从他或她的背景下安装观察家之间的内部沟通，提供全面覆盖的HBase功能。

一个较高的优先级的协处理器可能会在低优先级的抢占行动由那些通过抛出一个IOException（或子类）。我们将利用这一先发制人的功能，下面的访问控制协处理器的一个例子。

正如我们上面提到的各种事件会导致，观察者方法被调用加载观察员。 HBase的API，开始与HBase的0.92版本的事件和方法签名的。请知道的API可以改变未来，由于HBase的客户端API的变化，或其他可能的原因。在0.92版本之前，我们一直在努力，稳定的API，但也不能保证）。

RegionObserver接口提供的回调：

网上主盘前，postOpen：之前和之后调用该地区的报道。

前置液，postFlush的调用之前和之后的memstore被刷新到一个新的存储文件。

preGet，postGet：之前和之后调用的客户端发出GET请求。

preExists，postExists：使用GET存在的客户端测试之前和之后调用。

prePut和postPut：调用之前和之后的客户端存储一个值。

preDelete和postDelete：调用之前和之后的客户端删除的值。

等等

请参考HBase的0.92的javadoc中得到整个列表的声明的方法。

我们提供了一个方便的的抽象类BaseRegionObserver，实现了的所有RegionObserver方法的默认行为，这样你就可以专注于你有兴趣，什么样的事件，而无需关注的过程中向上调用所有的人都。

下面是一个序列图，示出如何RegionObservers工作与其他的HBase组件：

Endpoint

正如前面提到的，观察者就像是传统数据库中的触发器。而Endpoint类似于存储过程，是更强大的。在任何时候，Endpoint都可以由客户端调用。Endpoint的实现会在对应的远程region 服务器上执行，处理的结果会返回给客户端。

Endpoint是一个动态RPC的扩展。Endpoint的实现安装在服务端，由HBase RPC调用。客户端库提供了方便的方法来调用这些动态接口。

如上所述，有没有任何观察者执行停止执行的端点进行通信。这些扩展表面结合，您可以添加全新的功能，而无需修改或重新编译的HBase本身HBase的。这可以是非常强大的。

为了建立和使用自己的Endpoint，你需要：

定义一个扩展自CoprocessorProtocol的新的协议接口。

实现Endpoint接口。实现会被加载到region上下文然后执行。

扩展抽象类BaseEndpointCoprocessor。这个类隐藏了某些实现者无需考虑的内部细节，例如协处理器框架的类加载。

在客户端，Endpoint可以通过两个新的HBase的客户端API调用：

针对单一region执行：HTableInterface.coprocessorProxy(Class<T> protocol, byte[] row)

跨region执行：

HTableInterface.coprocessorExec(Class<T> protocol, byte[] startKey, byte[] endKey, Batch.Call<T,R> callable)

下面是一个用以说明Endpoint如何工作的例子。

在这个例子中，端点将扫描区域在一个给定列和聚合值，预计可序列化的Long值，然后将结果返回给客户端。客户端收集返回的远程端点调用在个别地区，部分聚集和总结的结果，在整个表中得到最终的答案。需要注意的是HBase的客户端有责任为并行的端点调用到目标区域调度，用于收集返回的结果呈现给应用程序的代码。这是一个轻量级的MapReduce工作：“地图”是端点在该地区的服务器上每一个目标区域，和“减少”是最后聚集在客户端执行。同时，在服务器端和客户端库中的协处理器架构是一样的MapReduce框架，移动繁琐的分布式系统编程的细节背后一个干净的API，使程序员可以专注于应用程序。

还要注意的是目前的HBase，和所有的Hadoop，需要Java 6中，其中有详细的语法为匿名类。 HBase的和Hadoop的发展与引进的Java语言功能，我们预计可以大幅减少客户端的端点代码的详细程度。