Apache IoTDB C#客户端介绍

大家好，我是袁秀龙，目前硕士就读于清华软件学院。去年4月份，我的导师刘璘老师建议我、卢展和曾正同学为Apache IoTDB研发C#客户端以支持C#技术栈的企业和个人能够方便的使用IoTDB。在经过数周的调研和开发后，在去年五月份，我们对该客户端进行了开源和在NuGet上的发布。自发布后，C#客户端得到了用户的数千次下载，并收到了许多来自用户的反馈和改进建议：如提供更加友好的接口，提供并行请求发送处理、支持Session的异常重连（来自卢展同学的支持）等。同时C#客户端也及时更新支持最新的Apache IoTDB的特性，如对齐序列插入、SchemaTemplate操纵接口的支持、支持插入空值的Tablet结构等（来自卢展同学的支持）。我们甚至还有一位来自立陶宛的小哥参与进来给我们重构了代码结构。在这篇博客中我们对C#客户端的关键特性以及安装使用方式进行基本介绍。

客户端安装

为了方便安装，我们为C#用户准备了NuGet包，用户可直接通过.NET CLI进行客户端安装，NuGet包链接如下，在你的命令行中运行如下命令即可完成最新版本客户端的安装。同时为了支持更好的跨平台特性，我们的客户端基于.Net SDK进行开发与构建（来自曾正同学的支持）。

Bash

```
dotnet add package Apache.IoTDB
```

关键特性

SessionPool

与Apache IoTDB的Java客户端类似，C#客户端同样提供支持并行请求发送处理的SessionPool的实现，在网络请求耗时较长或者不稳定的场景，SessionPool能够提供比单客户端更优异的性能。关于SessionPool，我们在文档中提供了详细的实现细节说明以及性能测试，具体可见文档链接（来自马文煊同学的支持）。

ByteBuffer

客户端和服务器进行数据通信时需要大量的按照Apache IoTDB序列化协议进行序列化和反序列化操作，如果直接使用C#的序列化则会带来大量频繁的内存申请和释放开销。为此，我们在C#客户端中提供了对大Tablet的内存占用估计以及使用倍增法进行内存扩容来提升序列化和反序列化的性能，使用实现的ByteBuffer进行序列化和反序列化要比C#原生序列化带来30倍的性能提升，具体细节可参见文档。

使用示例

本节我们对C#客户端的使用进行代码示例，C#客户端暴露的所有接口均为异步接口。使用C#客户端从首先建立一个SessionPool开始，建立SessionPool时需要指定服务器的`IP`、`Port`以及SessionPool的大小，SessionPool的大小代表本地与服务器建立的连接的数目。

C#

```
var session_pool = new SessionPool(host, port, pool_size);
await session_pool.Open(false);
```

建立好SessionPool后我们便可以借助它创建一个时间序列：

Visual Basic

```
status = await session_pool.CreateTimeSeries(
                "root.97209_TEST_CSHARP_CLIENT_GROUP.TEST_CSHARP_CLIENT_DEVICE.TEST_CSHARP_CLIENT_TS1", TSDataType.TEXT,
                TSEncoding.PLAIN, Compressor.UNCOMPRESSED);
```

我们可以使用SessionPool进行逐行Record的插入，由于SessionPool可以支持并行发送，所以当发送大量Record时，SessionPool会提供比单个客户端连接更优异的性能。

Dart

```
var tasks = new List<Task<int>>();

for (var timestamp = 1; timestamp <= fetch_size * processed_size; timestamp++)
{
    var rowRecord = new RowRecord(timestamp, values, measures);
    var task = session_pool.InsertRecordAsync(
        "root.97209_TEST_CSHARP_CLIENT_GROUP.TEST_CSHARP_CLIENT_DEVICE", rowRecord);
    tasks.Add(task);
}

Task.WaitAll(tasks.ToArray());
```

我们还可以使用SessionPool去执行SQL语句，并对结果进行显示。

TypeScript

```
var res = await session_pool.ExecuteQueryStatementAsync(
                "select * from " + string.Format("{0}.{1}", test_group_name, test_devices[1]) + " where time<15");
res.ShowTableNames();
while (res.HasNext()) Console.WriteLine(res.Next());
```

以上我们对C#的客户端使用的几个接口调用进行简单介绍，我们在仓库中提供了相关的API文档以及大量的测试代码，用户可以阅读这些文档和代码来了解C#客户端支持的接口的使用方式（来自卢展、曾正、马文煊同学的支持）。

总结

以上我们对Apache IoTDB的C#客户端进行了总体介绍，非常欢迎C#技术栈的Apache IoTDB用户使用我们的客户端并向我们提出改进建议！