大数据应用之HBase数据插入性能优化实测教程

时间 2013-10-09 09:35:29  [CSDN博客](http://www.tuicool.com/sites/vQZRB3)原文  [http://blog.csdn.net/hadoopdevelop/article/details/12491827](http://blog.csdn.net/hadoopdevelop/article/details/12491827?utm_source=tuicool)

**引言：**

大家在使用HBase的过程中，总是面临性能优化的问题，本文从HBase客户端参数设置的角度，研究HBase客户端数据批量插入性能优化的问题。事实胜于雄辩，数据比理论更有说服力，基于此，作者设计了这么一个HBase数据插入性能优化实测实验，希望大家用自己的服务器跑出的结果，给自己一个值得信服的结论。

**一、客户单优化参数**

1.Put List Size   
HBase的Put支持单条插入，也支持批量插入。   
  
2. AutoFlush   
AutoFlush指的是在每次调用HBase的Put操作，是否提交到HBase Server。 默认是true,每次会提交。如果此时是单条插入，就会有更多的IO,从而降低性能   
  
3.Write Buffer Size   
Write Buffer Size在AutoFlush为false的时候起作用，默认是2MB,也就是当插入数据超过2MB,就会自动提交到Server   
  
4.WAL   
WAL是Write Ahead Log的缩写，指的是HBase在插入操作前是否写Log。默认是打开，关掉会提高性能，但是如果系统出现故障(负责插入的Region Server 挂掉)，数据可能会丢失。

|  |  |  |
| --- | --- | --- |
| 参数 | 默认值 | 说明 |
| JVM Heap Size |  | 平台不同值不同自行设置 |
| AutoFlush | True | 默认逐条提交 |
| Put List Size | 1 | 支持逐条和批量 |
| Write Buffer Size | 2M | 与autoflush配合使用 |
| Write Ahead Log | True | 默认开启，需要手动关闭 |
| … |  |  |
| … |  |  |

**二、源码程序**

import java.io.IOException;

import java.util.ArrayList;

import java.util.List;

import java.util.Random;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.hbase.HBaseConfiguration;

import org.apache.hadoop.hbase.HColumnDescriptor;

import org.apache.hadoop.hbase.HTableDescriptor;

import org.apache.hadoop.hbase.KeyValue;

import org.apache.hadoop.hbase.MasterNotRunningException;

import org.apache.hadoop.hbase.ZooKeeperConnectionException;

import org.apache.hadoop.hbase.client.Delete;

import org.apache.hadoop.hbase.client.Get;

import org.apache.hadoop.hbase.client.HBaseAdmin;

import org.apache.hadoop.hbase.client.HTable;

import org.apache.hadoop.hbase.client.Result;

import org.apache.hadoop.hbase.client.ResultScanner;

import org.apache.hadoop.hbase.client.Scan;

import org.apache.hadoop.hbase.client.Put;

import org.apache.hadoop.hbase.util.Bytes;

/\*

 \* -------优化案例说明------------

 \* 1.优化参数1：Autoflush                默认关闭，需要手动开启

 \* 2.优化参数2：put list size            支持单条与批量

 \* 3.优化参数3：JVM heap size             默认值是平台而不同，需要手动设置

 \* 4.优化参数4：Write Buffer Size        默认值2M

 \* 5.优化参数5：Write Ahead Log             默认开启，需要手动关闭

 \* \*/

public class TestInsert {

    static Configuration hbaseConfig = null;

    public static void main(String[] args) throws Exception {

        Configuration HBASE\_CONFIG = new Configuration();

        HBASE\_CONFIG.set("hbase.master", "192.168.230.133:60000");

        HBASE\_CONFIG.set("hbase.zookeeper.quorum", "192.168.230.133");

        HBASE\_CONFIG.set("hbase.zookeeper.property.clientPort", "2181");

        hbaseConfig = HBaseConfiguration.create(HBASE\_CONFIG);

        //关闭wal,autoflush,writebuffer = 24M

        insert(false,false,1024\*1024\*24);

        //开启AutoFlush，writebuffer = 0

        insert(false,true,0);

        //默认值，全部开启

        insert(true,true,0);

    }

    private static void insert(boolean wal,boolean autoFlush,long writeBuffer)

            throws IOException {

        String tableName="etltest";

        HBaseAdmin hAdmin = new HBaseAdmin(hbaseConfig);

        if (hAdmin.tableExists(tableName)) {

            hAdmin.disableTable(tableName);

            hAdmin.deleteTable(tableName);

        }

        HTableDescriptor t = new HTableDescriptor(tableName);

        t.addFamily(new HColumnDescriptor("f1"));

        t.addFamily(new HColumnDescriptor("f2"));

        t.addFamily(new HColumnDescriptor("f3"));

        t.addFamily(new HColumnDescriptor("f4"));

        hAdmin.createTable(t);

        System.out.println("table created");

        HTable table = new HTable(hbaseConfig, tableName);

        table.setAutoFlush(autoFlush);

        if(writeBuffer!=0){

            table.setWriteBufferSize(writeBuffer);

        }

        List<Put> lp = new ArrayList<Put>();

        long all = System.currentTimeMillis();

        System.out.println("start time = "+all);

        int count = 10000;

        byte[] buffer = new byte[128];

        Random r = new Random();

        for (int i = 1; i <= count; ++i) {

            Put p = new Put(String.format("row d",i).getBytes());

            r.nextBytes(buffer);

            p.add("f1".getBytes(), null, buffer);

            p.add("f2".getBytes(), null, buffer);

            p.add("f3".getBytes(), null, buffer);

            p.add("f4".getBytes(), null, buffer);

            p.setWriteToWAL(wal);

            lp.add(p);

            if(i%1000 == 0){

                table.put(lp);

                lp.clear();

            }

        }

        System.out.println("WAL="+wal+",autoFlush="+autoFlush+",buffer="+writeBuffer+",count="+count);

        long end = System.currentTimeMillis();

        System.out.println("total need time = "+ (end - all)\*1.0/1000+"s");

        System.out.println("insert complete"+",costs:"+(System.currentTimeMillis()-all)\*1.0/1000+"ms");

    }

}

**三、集群配置**

3.1 服务器硬件配置清单

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 节点名称 | CUP | 内存 | 硬盘 | 带宽 |
| 1 | HMaster |  |  |  |  |
| 2 | HregionServer1 |  |  |  |  |
| 3 | HregionServer2 |  |  |  |  |
| 4 | … |  |  |  |  |
| 5 |  |  |  |  |  |
| 6 |  |  |  |  |  |
| 7 |  |  |  |  |  |

3.2 客户端硬件配置清单

|  |  |  |
| --- | --- | --- |
| 设备 | 节点名称 |  |
| Cpu |  |  |
| 内存 |  |  |
| 硬盘 |  |  |
| 带宽 |  |  |

**四、测试报告**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 数据量 | JVM | AutoFlush | Put List Size | WriteBufferSize | WAL | 耗时 |
| 1000 | 512m | false | 1000 | 1024\*1024\*24 | false |  |
| 2000 |  |  |  |  |  |  |
| 5000 |  |  |  |  |  |  |
| 10000 |  |  |  |  |  |  |
| 20000 |  |  |  |  |  |  |
| 50000 |  |  |  |  |  |  |
| 100000 |  |  |  |  |  |  |
| 200000 |  |  |  |  |  |  |
| 500000 |  |  |  |  |  |  |
| 100000 |  |  |  |  |  |  |

**备注：该技术专题讨论正在群Hadoop高级交流群：293503507同步直播中，敬请关注。**