doi:10.3969/j.issn.1007-7162.2014.03.020

Storm 实现的应用模型研究

邓立龙[1],徐海水[1,2]
(广东工业大学 1. 计算机学院;2. 网络信息与现代教育技术中心,广东 广州 510006)

摘要: 通过对 Storm 的核心理念和编程模型进行探讨,分析了 Storm 的工作方式和应用方法,并对一个基于 Storm 实现的数据分析处理系统进行了性能测试和水平扩展测试. 实验结果表明,Storm 实现的数据分析处理系统在性能和可伸缩性上要优于传统的数据分析处理系统.

关键词: Storm;分布式并行计算;大数据
中图分类号: TP391 **文献标志码:** A **文章编号:** 1007-7162(2014)03-0114-05

Research on Applied Models Based on Storm

Deng Li-long[1], Xu Hai-shui[1,2]
(1. School of Computers; 2. Center of Internet Information and Modern Education, Guangdong University of Technology, Guangzhou 510006, China)

Abstract: It discussed the core ideas and programming model of Storm, and analyzed its working mode and application method. Finally, it implemented the performance and horizontal scaling test of a data processing system based on Storm. The experimental results show that the performance and scalability of Storm is superior to that of the traditional data processing system.

Key words: Storm; distributed real-time parallel computing; big data

在当前这个信息爆炸的时代,互联网上的数据正以几何级的速度增长. 截止 2012 年 1 月,新浪微博注册用户数已超过 3 亿,用户日平均在线时长 60 min,平均每天发布超过 1 亿条微博[1]. 在这种背景下,云计算(Cloud Computing)的概念被正式提出,立即引起了学术界和产业界的广泛关注和参与. Google 是云计算最早的倡导者,随后各类大型软件公司都争先在“云计算”领域进行了一系列的研究部署工作[2]. 目前最流行的莫过于 Apache 的开源项目 Hadoop 分布式计算平台,Hadoop 专注于大规模数据存储和处理. 这种模型对以往的许多情形虽已足够,如系统日志分析、网页索引建立(它们往往都是把过去一段时间的数据进行集中处理),但在实时大数据处理方面,Hadoop 的 MapReduce 却显得力不从心. 业务场景中需要低延迟的响应,希望在秒级或者毫秒级完成分析,得到响应,并希望能够随着数据量的增大而拓展[3]. 此时,Twitter 公司推出开源分布式、容错的实时流计算系统 Storm,它的出现使得大规模数据实时处理成为可能,填补了该领域的空白.

1 Storm 简介

Storm 是一个开源的分布式实时计算系统,可以简单可靠地处理大量数据流[4]. 其主要应用场景为实时分析、在线机器学习、持续计算、ETL、分布式 RPC 等[5]. 此外,Storm 支持水平扩展,具有高容错性,可以确保每个消息都被处理到,而且具有很高的处理速度,在一个小的 Storm 集群中,每个结可以达到每秒数以百万计消息的速度.

与其他大数据解决方案相比,Storm 有着不同的处理方式. 本质上来看 Hadoop[6-7] 是一个批处理系统,数据被引入 Hadoop HDFS,然后分发到各个节点进行处理,当处理任务完成之后,数据结果会再次返回到 HDFS 供始发者使用. Storm 通过创建拓扑结构来处理没有终点的数据流,与 Hadoop 作业不同的是,这些转换工作会一直进行,持续处理数据流中新

收稿日期: 2014-04-16
基金项目: 国家自然科学基金重大研究计划项目(90818008)
作者简介: 邓立龙(1988-),男,硕士研究生,主要研究方向为大数据和云计算.

到达的数据[8].

1.1 Storm 的工作机制

Storm 主要有两种类型的节点: 主节点(Master)和工作节点(Worker). 主节点通常会运行一个后台程序,称为 Nimbus. 它负责发送代码到集群,分配工作任务给每一个工作节点,并监控其运行状态,作用类似于 Hadoop 中的 Job Tracker. 工作节点会运行一个名为 Supervisor 的后台程序,Supervisor 负责监听从 Nimbus 分配给它执行的任务,据此启动或停止执行任务的工作进程[9]. 在集群系统中,一般一个节点上运行一个或多个工作进程,每一个工作进程都会执行一个 Topology 任务的子集. 一个 Topology 任务往往需要分布在不同工作节点上的多个工作进程来执行.

如图 1 所示,当一个 Topology 定义好后被提交,首先会由 Storm 提供的方法把 jar 包上传到 Nimbus,它会对 Storm 本身和 Topology 进行校验,主要检查 Storm 的状态是否为 Active 以及 Topology 是否有同名的实例在运行. 接着,Nimbus 对每个 Topology 都会做出详细的预算,如工作量(多少个 Task),它会根据 Topology 中定义的 parallelism hint 参数,来给 Spout/Bolt 设定 Task 数目,并且分配与其对应的 Task-id,再把分配好 Task 的信息写入 Zookeeper[10] 上的/task 目录下. 然后 Nimbus 会给 Supervisor 分配工作,方法是把任务信息写在 Zookeeper 的/assignments. Supervisor 每隔一定时间都会查看/assignments 目录,检查 Nimbus 是否有新任务分配,当有新提交的任务时,它会先下载代码,然后根据任务信息安排 Worker 执行这些任务.

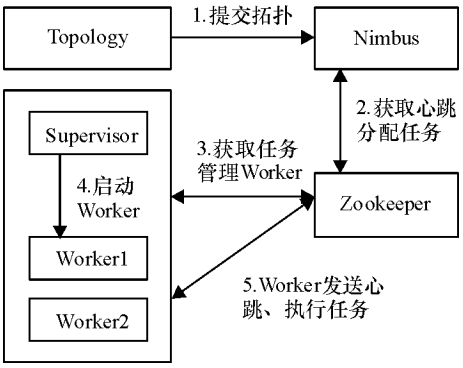

图 1　Topology 提交的流程图

Fig. 1　The flow chart of Topology

如图 2 所示,在 Storm 集群中 Nimbus 和 Supervisor 都是无状态的,并且两个模块之间没有直接的数据交互,所有的状态都是保存在 Zookeeper, Nimbus 通过写入 Zookeeper 来发布指令. 而 Supervisor 则通过读取 Zookeeper 节点信息来执行这些指令. 同时 Supervisor 和 Task 会定时发送心跳信息到 Zookeeper, 使得 Nimbus 可以监控整个 Storm 集群的状态. 当有 Task 节点挂掉时 Nimbus 能够快速使之重启. 这种工作方式使得整个 Storm 集群十分健壮,任何一台工作机器突然失效都不会影响到整个系统的正常运行,只需重启失效节点后再从 Zookeeper 上面重新获取状态信息即可.

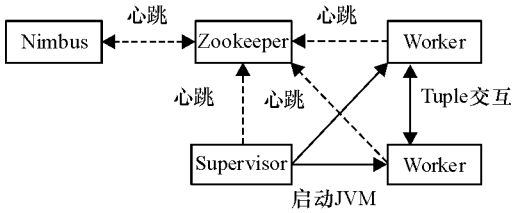

图 2　Storm 数据交互图

Fig. 2　The data interaction diagram of Storm

1.2 Storm 分布式并行计算编程模型

Storm 是 Twitter 开源的一个实时数据处理框架,Twitter 每天约 3.4 亿条的推文正是用 Storm 进行实时分析处理. Storm 实现了一种流式处理模型,流是一组有顺序并连续到达的数据序列[11]. 在 Storm 设计思想中,把流中的事件抽象为 Tuple 即元组,把源头处理抽象为 Spout,把流的处理抽象为 Bolt. 这种思想大大简化了分布式实时并行处理程序的开发难度.

在 Storm 计算模型中,主要有两种类计算过程,源头处理过程 Spout 和中间处理过程 Bolt. 因此需要用户去实现 ISpout 和 IBolt 这两种类型的接口. 作为 Storm 中的消息源,Spout 用于 Topology 生产消息,一般会不断地从外部数据源(如 Message Queue、NoSQL、RDBMS、Log File)读取然后发送消息给(Tuple 元组) Topology. 之后消息会以某种方式传给 Bolt,作为 Storm 的消息处理者,Bolt 可以执行过滤、聚合、数据库查询等操作或与外部实体通信,可以根据情况选择储存数据,或是把数据传给下一级 Bolt.

Bolt 既可实现传统 MapReduce 之类的功能,也可实现更复杂的操作,如过滤、聚合等. 如果对两个组件数据发送有特殊要求,例如在应用场景中需要把相同 key 值的元组发送到同一个 Bolt 下进行统计,可使用 Storm 提供的数据流分发(Stream Grouping)策略来解决这一问题. 为提高处理效率,可以在一个流上加入多个 Spout 和多个 Bolt. 典型的 Storm 拓扑结构会实现多个转换,因此需要多个具有独立元组流的 Spout. 如图 3 所示,Storm 集群是由许多 Bolt 组件组成的链式处理结构,每个 Bolt 对 Spout 发

射出的数据进行各种转换操作.

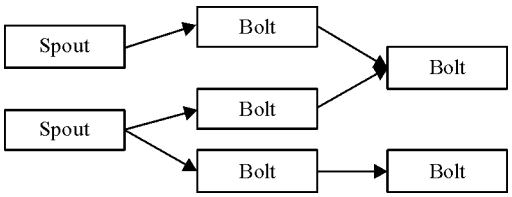

图3 Storm 数据流网络图

Fig. 3 The data flow network diagram of Storm

使用 Storm 可以轻松地实现 MapReduce 功能[12]. 如图4所示,Spout 生成文本数据流,Bolt 实现 Map 功能(令牌化一个流中的各个单词). 来自“map” Bolt 的流然后流入一个实现 Reduce 功能的 Bolt 中.

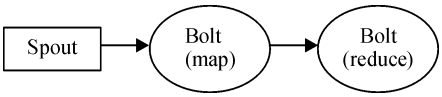

图4 Storm 实现 MapReduce 功能

Fig. 4 Implementation of the function of MapReduce with Storm

2 实时微博热词分析统计系统

作为一个新生的网络社交平台,微博成功吸引了数以万计的粉丝进行互动交流. Bernard J. Jansen, Mimi Zhang 等对 150 000 条微博进行分析,发现19%的微博会提及某个品牌,其中大约20%的微博会包含一些观点,因此快速并准确地掌握民众实时关注的热点信息对企业在未来的市场竞争中占领先机具有重要作用[13-14]. 以下为微博热词分析统计系统的设计,通过系统日志实时分析出当前热门词汇.

2.1 算法实现流程

用一个 Bolt 对流进行全局分组,并在内存中维护一个 Top N 的列表. 由于整个流都会发送到一个节点进行处理,所以这种方式对于大的数据量显然没有可伸缩性. 一种更好的方式是并行计算流的各部分 Top N,然后把局部的 Top N 合并在一起,再计算出一个全局的 Top N. 计算流程如图5所示,计算步骤如下:

(1) Spout 获取微博内容后把消息发送出;

(2) 运行若干个 Bolt 进行 count 统计,即每个 key 值出现的次数,然后把统计结果发射;

(3) 运行一组 Bolt 进行 Rank 计算,它们负责一部分数据的 Top N 计算,然后把结果数据发送到下一级 Bolt;

(4) 利用一个全局的 Bolt 来合并这些机器上计算出来的局部 Top N 结果,合并后得到最终全局的 Top N 结果.

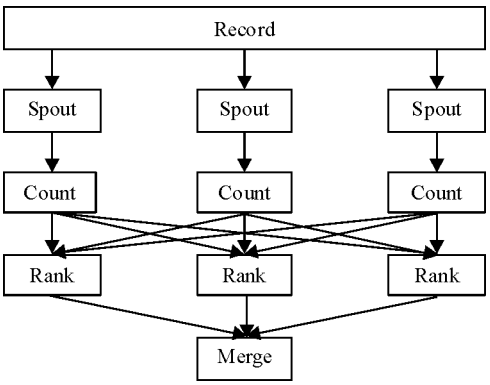

图5 热词分析流程图

Fig. 5 The flow chart of hot word analysis

2.2 Spout 实现

Spout() 是 topology 的数据源 Spout,在服务器上不间断的有微博产生,本实例从 Redis 读取记录,产生数据流“keyword”,输出 Fields 是“keyword”,核心代码如下:

```
public void nextTuple()
{
    ……
    String content = jedis.rpop("record"); // 从 Redis 读取记录
    if(content == null || "nil".equals(content)) {
    睡眠 300 微秒等待}
    try
    {
      JSONObject obj = JSONValue.parse(content);
      String keyword = obj.get("keyword");
      collector.emit(new Values(keyword)); // 提交 keyword
    }
    ……
    }
```

2.3 Count 实现

“word”流入 Count 这个 Bolt 中进行 keyword count 计算,为了保证同一个 keyword 的数据被发送到同一个 Bolt 中进行处理,按照“keyword”字段进行 field grouping;在 Count 中会计算各个 keyword 的出现次数,然后产生“count”流,输出“keywordcount”和“count”两个 Field,核心代码如下:

```
public void execute (Tuple tuple) {
    /* 如果 keyword 不在哈希表中,则加入哈希表,如果在其中,则把计数加 1* /
```

```
if(! counters.containsKey(str))
{
  counters.put(str, 1);
} else
{
  Integer c = counters.get(str) + 1;
  counters.put(str, c);
}
collector.emit(new Values(obj, totalObjects(obj)));
//确认此元组处理完
collector.ack(input);
}
```

2.4 Rank 实现

Rank 这个 Bolt 按照“count”流的“keyword-count”字段进行 field grouping; 在 Bolt 内维护 Top *N* 个有序的链接,如果超过 Top *N* 个链接,则将排在最后的链接数据移除,同时每隔一定时间(2 s)产生“rank”流,输出“list”字段,计算出的局部 Top *N* 结果会到下一级数据流“merge”流,核心代码如下:

```
public void execute(Tuple tuple, BasicOutputCollector collector)
{
Object tag = tuple.getValue(0);
Integer existingIndex = _find(tag);
//如果元素不存在,再加入集合中,如果存在就更新值
if(null! = existingIndex)
{
    rankings.set(existingIndex, tuple.getValues());
} else
{
    rankings.add(tuple.getValues());
}
对 rankings 列表进行排序
if(rankings.size() > _count) {
    移除末尾的元素
}
long currentTime = System.currentTimeMillis();
if(时间过去 2 s)
    发射列表,更新当前时间
}
```

2.5 Merge 实现

Merge 这个 Bolt 会按照“rank”流的进行全局的 grouping,即所有上一级 Bolt 产生的局部 Top *N* 结果“rank”流都流到这个“merge”流进行处理; Merge 的计算逻辑和 Rank 类似,只是将各个 Merge 的 Bolt 合并后计算得到最终全局的 Top *N* 结果,代码不再赘述.

3 实验结果与分析

3.1 实验环境

本文使用了 5 台计算机搭建实验平台,建立了一套 Storm 集群系统进行分布式并行计算,各个计算节点所用的操作系统为 Linux CentOS6.3,内核版本为 2.6.18,使用的 Storm 版本为 0.9,JDK 版本为 jdk1.7.0_40. 其中一台计算机用来运行 Nimbus 守护进程和 Supervisor 守护进程,其余 4 台只运行 Supervisor,每个 Supervisor 节点最多启动 6 个 Worker 进程.

3.2 性能测试

为方便比较,本实验使用 Java 语言开发了一个普通分布式版本的处理系统[15],运行相同的线程数,在相同机器配置上进行微博热词统计测试. 实验结果如图 6 所示.

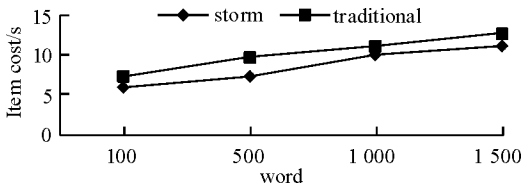

图 6　Strom 并行数与处理时间对照表

Fig. 6　The time table of storm parallel processing

从图 6 中可以看出,Storm 版本的处理性能优于普通分布式版本,处理相同的一批任务,用 Storm 实现的版本需要花费的时间更短.

3.3 横向扩展测试

为测试 Storm 横向扩展给系统性能带来的提升,笔者进行了分组实验. 实验分为 3 组,每组分配给 Topology 不同的 Worker 数目,实验中 Executors 和 Tasks 的个数不变. 每组分配的任务数不同,统计每组任务平均消耗的时间,实验结果取 3 次的平均值.

由图 7 可知,随着处理节点数的增加,集群的处理能力也会随着线性增长. 遇到性能瓶颈时,可以通过提高瓶颈节点的并行任务数量来提高性能. Storm 适用于解决高度并行性的海量数据型实时计算问

题,并能将计算结果实时反馈给用户. 使用 Storm 处理并行计算问题时,用户只需把任务分解成单个节点的计算单元,利用传统的串行算法写出计算函数和接收函数.

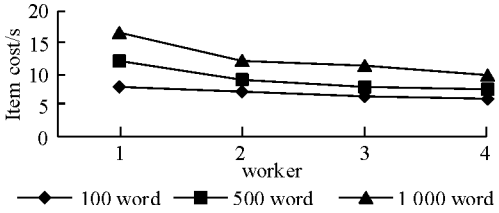

图7 水平扩展性能测试

Fig. 7 The performance test for extending

4 结束语

本文分析了基于 Storm 应用的工作机制和编程模型,并基于 Storm 成功实现了实时微博热词分析统计系统,实现了在低端机组成的集群上进行的分布式算.

参考文献:

[1] 新浪微博数据中心. 2012 年新浪微博用户发展报告 [EB/OL]. [2014 - 06 - 15]. http: // data. weibo. com/report/report? copy _ ref = AEdhAAT9K9K7& _ key = INEZOM.

[2] 陈康,郑纬民. 云计算: 系统实例与研究现状 [J]. 软件学报,2009(5): 1337-1348.
Chen K, Zheng W M. Cloud computing: system instances and current research [J]. Journal of Software, 2009,05: 1337-1348.

[3] Leibiusky J, Eisbruch G, Simonassi D. Getting Started With Storm [M]. US: O'Reilly Media, 2012.

[4] The Apaehe Foundation. Storm official website [EB/OL]. [2014 - 04 - 08]. http: // storm-project. net/.

[5] Github Inc. Storm Wiki [EB/OL]. [2013 - 12 - 07]. https: // github. com/nathanmarz/storm/wiki.

[6] The Apaehe Foundation. Apache Hadoop [EB/OL]. [2014 - 03 - 03]. http: // hadoop. apache. org/. White T. Hadoop: The definitive guide [M]. US: O'Reilly Media, 2012.

[7] White T. Hadoop: The definitive guide [M]. US: O'Reilly Media, 2012.

[8] 金晓军. Trident Storm 与流计算经验 [J]. 程序员, 2012(10): 99-103.
Jin X J. Trident Storm and flow calculation experience [J]. Journal of Programmers, 2012(10): 99-103.

[9] Petko V. Integrating parallel application development with performance analysis in periscope [J]. IPDPS Workshops, 2010: 1-8.

[10] The Apaehe Foundation. Apache ZooKeeper [EB/OL]. [2014 - 03 - 18]. http: // zookeeper. apache. org/.

[11] 顾伟. 分布式流数据实时计算框架的研究和开发 [D]. 杭州: 浙江理工大学信息电子学院,2013.

[12] Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters [J]. Communications of the ACM, 2008, 51(1): 107-113.

[13] 白晓晴. 微博互动营销的优势及策略分析 [J]. 企业技术开发, 2010, 29(005): 20-21.
Bai X Q. Research on the advantages and strategies of micro-blog interactive marketing [J]. Technological Development of Enterprise, 2010, 29(005): 20-21.

[14] 孙擎. 浅析国内微博营销面临的挑战 [J]. 中国商贸, 2011(3): 28-29.
Sun Q. Research on the challenges of domestic micro-blog marketing [J]. China Business & Trade, 2011(3): 28-29.

[15] 林昊. 分布式 Java 应用基础与实践 [M]. 北京: 电子工业出版社,2010.