2014 年 4 月 第 4 期
Vol. 35 No. 4 2014

面向流媒体的 Hadoop 调度策略研究

孙连生,王 雷,黄承真

(中国科学技术大学 自动化系网络传播系统与控制安徽省重点实验室,合肥 230027)
E-mail:sunls@ mail. ustc. edu. cn

摘 要: 流媒体数据的分布式处理的研究在流媒体处理领域具有重要的意义,电信网、广播电视网、计算机通信网的不断融合促进了媒体数据的广泛传播,随着数据量的不断增大数据处理的速度变慢,而传统的多处理器并行计算难以满足海量流媒体数据的并发处理需求. 使用当前流行的开源分布式处理系统 Hadoop 对流媒体数据进行分布式计算将大大提高流媒体数据的处理速度,虽然 Hadoop 常用于大数据企业类业务数据处理,但其对实时数据处理支持较差. 因此对 Hadoop 的调度和负载策略进行研究,通过改进 Hadoop 的实时调度和负载均衡策略,使其可以满足流媒体处理的实时性、并发性和处理速度的要求. 实验结果表明提出的算法和策略使得 Hadoop 能很好的适应实时流媒体的处理,减少了处理的时间.

关 键 词: 流媒体;Hadoop;实时调度;负载均衡
中图分类号: TP391 **文献标识码**:A **文 章 编 号**:1000-1220(2014)04-0728-06

Research on Scheduling Strategy of Hadoop for Streaming Media Oriented

SUN Lian-sheng, WANG Lei, HUANG Cheng-zhen
(Department of Automation, USTC Key Laboratory of Network Communication System and Control, Hefei 230027, China)

Abstract: Distributed processing of streaming media has been an important direction among research of streaming media area. The continuous emerging of telecommunication network, broadcast network and television network make streaming media data spread widely. With the data increasing, processing speed of the data becomes slower and slower. The traditional multiprocessor parallel computing is difficult to meet the demand for massive concurrent processing of streaming data. Distributed computing for streaming media data using Hadoop, one of the most popular distributed processing systems, will greatly improve the processing speed of the streaming data. Hadoop is used to process big business data of companies, but its framework is not applied to real-time media processing. In this paper, we focus on how to use Hadoop to process real-time streaming media data. Improving its real-time scheduling and load balancing strategy makes Hadoop meet the requirements of real-time streaming media processing, concurrency, and processing speed. This experiment result shows that our algorithm and strategy improve the performance of processing real-time streaming media apparently, reduce the processing time.

Key words: streaming media; Hadoop; real-time scheduling; load balancing

1 引 言

新媒体的发展推动了三网融合[1],电信网、广播电视网和互联网相互渗透、兼容,并逐步整合成为互联互通的信息通讯网络. 在三网融合的影响下,各种应用同时接踵而至,手机电视、IPTV(Internet Protocol Television)[2]、视频制作、VoIP(Voice over Internet Protocol)[3]等成为三网融合典型业务,这些应用都需要建立在大量视频数据的基础上. 另一份调查显示 2006 年我国在线影视正规运营商市场规模是 2 亿元,而 2011 年在线视频行业市场规模已达到 62 亿元左右,视频行业规模的增长也反映出了视频数据量的快速增长. 随着移动 3G 视频业务[4]迅猛发展,流媒体业务已占整个互联网业务的很大一部分,由于数据量变得越来越庞大,需要消耗大量的技术资源来满足当前的服务和业务需求,虽然对流媒体处理的编转码技术在不断改进、服务器和网络带宽等基础设施也在不断完善,但是对于当前的快速流媒体处理需求仍然难以满足,以至于服务器响应速度缓慢,照成网络拥塞不堪,最终影响到用户体验下降,因此如何对流媒体的快速处理将是市场的业务需求也是未来面临的重大挑战. 如何更快地处理、存储和发布这些海量的网络流媒体并且使服务器负载达到快速响应的稳定均衡状态,同时使得每个节点能够根据其性能得到合理的利用成为亟待解决的问题.

目前为了解决这些问题,国内外有很多针对此类问题的研究. 其中使用 CDN(Content Delivery Network)[5]和 P2P(Peer to Peer)[6]等技术来为解决网络传输的问题,尽量使得视频服务靠近终端同时利用终端的资源来解决网络拥塞的问题. 而在实时流媒体处理方面,为了加快流媒体的处理速度,传统的方法是尽量提高单机的并行处理能力,包括 DSP(Dig-

收稿日期:2013-01-10 收修改稿日期:2013-03-01 基金项目:中央高校基本科研业务费专项资金项目(WK2100100012)资助. 作者简介:**孙连生**,男,1987 年生,硕士研究生,研究方向为网络多媒体以及复杂系统的仿真与优化;**王 雷**,男,1972 年生,博士,副教授,研究方向为网络多媒体以及复杂系统的仿真与优化;**黄承真**,男,1987 年生,硕士研究生,研究方向为网络多媒体以及复杂系统的仿真与优化.

ital Signal Processor)[7]、GPU(Graphic Process Unit)[8]技术,他们主要针对计算机的硬件乘法器、算术逻辑单元和显卡的性能进行提高,是在进程和线程的级别对流媒体的处理速度进行提高,虽然处理能力有所提高,但单台机器处理会达到瓶颈处理能力提高有限;另一种技术 DVC(Distributed Video Coding)[9]分布式视频编码主要适合编码终端设备能力比较弱而解码服务器能力较强的情况,但其对编码、转码的快速处理仍然无能为力;现在有很多研究是使用分布式的方式对流媒体进行处理,分布式处理的特点是利用了大量同构的处理能力一般的处理器对任务进行分解,多台机器并行处理任务,从而缩短任务的处理时间,目前的主要研究是使用 Google 提出的 Map Reduce 架构[10]和基于 Hadoop 平台的分布式视频转码方案[11]进行视频处理,这些研究的方向是如何进行任务的执行顺序安排和任务的大小划分,而为了适应实时流媒体的特点却未在分布式处理的调度和负载均衡策略方向上做出大的改进;微软提出了媒体云和云媒体的概念,它侧重于解决云如何为流媒体运算提供 QoS(Quality of Service)支持[12]的问题,Grio 使用 Amazon 的 S3(Simple Storage Service)服务器对视频文件进行存储,利用 EC2(Elastic Compute Cloud)对视频文件进行转码[11],这些方式都是使用云的计算和云的存储能力来完成流媒体的计算处理,其底层的实现细节也是使用了分布式处理的方式.

综上结合目前的流媒体分布式处理方法,本文拟采用基于 Map-Reduce[13]编程模式实现的 Hadoop 架构进行实时流媒体的编码和转码方案研究,在分析 Hadoop 特性的基础上,基于 Hadoop 架构设计实时流媒体处理系统,对其调度策略和负载均衡策略进行改进并进行仿真验证.

2　Hadoop 任务调度策略

Hadoop[14]是 Map-Reduce 的编程模式的一个实现,它是一个分布式系统基础架构,由 Apache 基金会开发. 用户可以在不了解分布式底层细节的情况下,基于 Hadoop 开发分布式程序. Hadoop 提供可靠、高效、可伸缩的处理方式,其可靠性在于维护多个数据副本,确保能够针对失败的节点进行重新分布处理;其高效性在于能够管理多个节点以并行的方式工作,以加快处理速度;其可伸缩性在于能够处理海量的数据(PB 级). 此外,由于 Hadoop 对服务器性能没有要求,可以使用配置低的服务器甚至普通计算机,因此应用系统的实现成本可以很低廉.

Hadoop 的实现包含了多种任务调度策略,包括 FIFO(First In First Out)调度、公平调度、容量调度等,适用于不同场景. FIFO 调度是 Hadoop 的默认调度器,作业按照开始时间进行排列,每次从队首取一个作业进行运行,该策略简单、易于实现,能够减轻主控节点的负担,缺点是所有作业都同等对待,没有考虑作业的迫切程度,另外对小作业的运行不利. 公平调度是 Facebook 开发的调度器,尽量将等量的资源份额分给不同种类的作业,当有新作业时该算法会释放一定的资源并分配给新作业,以保证所有的作业都能够获得公平的计算资源量,这样就使得短作业和长作业都能在合理的时间内完成. 该算法重点考虑公平性,但数据的本地化程度较差,导致调度数据往往来源于其他机器,从而造成带宽浪费. 容量调度是 Yahoo! 开发的调度器,与公平调度使用了类似的算法,将一个队列分为多个作业队列,每个队列共享集群资源的百分比,这样可以限制每个队列的最大资源,每个队列中使用 FIFO 调度,并支持优先级,每个队列分配给每个用户受限制的容量,它适用于具有多个客户端和不同类型、不同优先级的作业大型 Hadoop 集群中.

然而,由于实时流媒体具有以下几方面特点,Hadoop 现有的任务调度策略并不能适应实时流媒体处理:

1)实时性要求数据处理必须在截止时间之前完成,否则会在终端产生波动延迟;

2)媒体数据块较小,作业任务短,且任务大小相似;

3)当数据块不在本地时,会产生节点间的迁移;

4)数据往往有类型与优先级区分,需要的处理时间也不同.

此外,Hadoop 只是一个开发框架,没有针对实时流媒体的编码、转码等处理过程的具体实现.

本文的工作在于改进 Hadoop 的调度算法和负载均衡策略,以使得任务能够尽量本地化执行,合理利用计算资源,加快实时流媒体的处理速度,提高整个集群系统的性能.

3　Hadoop 进行流媒体的实时处理

基于上述的特性分析,本文提出一种基于用户优先级的 DF(Deadline First)调度算法和使用基于堆动态阈值混合负载均衡算法[15]以快速地对实时流媒体进行处理.

3.1　实时流媒体的 Map-Reduce 处理方法

由于 Hadoop 对大数据块的处理能力较强,考虑将整个系统接收的各类流媒体数据(手机视频、IPTV、会议视频、因特网视频等)缓存后分块封装,并组装成合适大小的视频块,再使用 Hadoop streaming 的编程模式将其作为一个单独的任务启动,从而实现大数据的实时性处理. 整个分布式处理的过程对用户来说是透明的,被分块封装的数据将按负载均衡策略均衡地分布到系统中的每个节点进行处理.

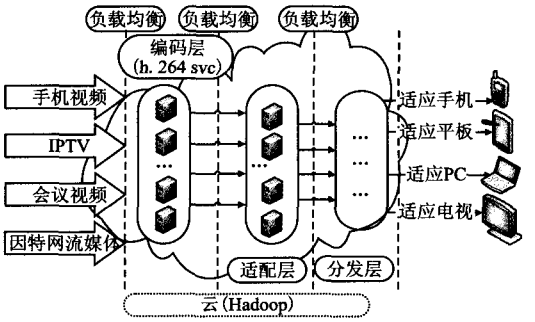

图 1　利用 Hadoop 进行实时流媒体处理架构图

Fig. 1　Architecture diagram of real-time streaming media processing with hadoop

完整的处理过程采用三层架构,如图 1 所示,包括编码层即 Map 阶段、适配层即 Reduce 阶段,以及分发层. 其中编码层将实时视频流编码生成 H. 264 SVC[16](scalable video coding,可伸缩视频编码)视频格式,H. 264 SVC 是 H. 264/AVC 视频编码标准的一个扩展部分,它的特点是可以增加增强层,

利用增强层提供时间层(帧率)、空间层(分辨率)以及质量层的可伸缩以适应不同客户端的需求. 适配层提取 H. 264 SVC 中的某些层转码成可以播放格式的视频以适配不同分辨率不同带宽不同处理能力的终端,如手机、个人电脑、平板等. 具体的编码和适配的具体处理方法将在下节进行介绍.

3.2 流媒体的编解码

SVC 的编转码工作可以使用 JSVM[17] ((Joint Scalable Video Model)处理. JSVM 是 H. 264/SVC 系列下的一款可扩展视频编码软件,主要用来实现可分级视频编码,以及进行转码和解码的操作.

在 Linux 环境下将 JSVM 编译成可执行程序后即可进行 SVC 编码,其输入是不同分辨率的视频文件,输出是 H. 264 SVC 文件. SVC 转码时使用 BitStreamExtractorStatic 提取 SVC 文件里的相应子流,使用 BitstreamExtractorStatic 命令生成高质量层或低质量层以适应不同终端的 SVC 文件,最后使用 H264AVCDecoderLibTestStatic 程序即可得到对应质量的 YUV 文件,供客户端使用播放.

为了移植编转码程序使其能够在 Hadoop 上运行,必须对其源码进行修改,修改输入输出接口,以使其能够接收标准输入输出,然后使用 Hadoop streaming 运行相应的编解码程序. 我们将修改后的 H264AVCEncoderLibTestStatic 程序作为 Map 操作,将 BitsreamExtractorStatic 程序作为 Reduce 操作.

3.3 Hadoop 的 FIFO 调度和负载策略

Hadoop 默认使用 FIFO 调度策略,其任务调度队列通过 Listener 机制在每次心跳时形成的. 当有 Job 的添加、修改、删除等动作时,相应的函数将会被触发进行 Job 的增、删、改操作. 每次获得 Job 队列时,Job 队列都是按照 Job 的开始时间排好序的.

在 Hadoop 的节点中有 TaskTracker 和 JobTracker 两种角色,TaskTracker 负责完成作业,而 JobTracker 负责作业的调度工作. TaskTracker 通过心跳向 JobTracker 告知尚有空余的任务 Slot,此时 JobTracker 向 TaskTracker 分派任务,具体的分派步骤如下.

1)对 Job 队列中的每一个 Job 顺序遍历,选择第一个 RUNNING 状态的 Job,尝试从中分派任务. 如果没有任务可以分派,则再选择下一个 RUNNING 状态的 Job;

2)分派 Map 任务时,首先优先分派执行失败后等待重试的任务,对于这些需要重试的任务,并不考虑位置问题;其次从正常等待的 Map 任务中进行分派,根据位置信息优先选择输入 split 数据块离 TaskTracker 最近的 Map 任务.

从上面的描述可以看出,每次分配任务时会优先处理最先开始的 Job,如果其 Map 任务在 TaskTracker 上面,就直接分配给它,否则分配该 Job 中距离 TaskTracker 最近的 Map 任务. 由于 Hadoop 默认的 FIFO 调度策略调度时只考虑作业的开始时间,采用这种调度策略会经常导致应该尽早结束的任务却一直处于饥饿状态,不能得到运行的资源,显然该策略不适用于流媒体业务. 另一方面,由于流媒体任务具有实时性和数据块小的特点,其 Map 任务只有一个,当 TaskTracker 请求时,得到的 Map 任务不是本地化数据的概率很大,会明显增加网络的传输带宽,降低了流媒体的处理速度. 本文考虑使用基于用户优先级的 DF(Deadline First)调度策略,以适应实时流媒体的业务需求.

为提高本地性,当上传 Hadoop 数据块时,如果上传数据的节点是 Hadoop 集群中的某一节点时,直接将数据块放置在本节点上,否则随机放置. 如果设置了机架,Hadoop 会考虑将数据块分散到不同机架上. 由于 Hadoop 的每个节点处理能力不同,一味使用随机策略,并不能充分利用每个节点的性能,因此使用基于堆的动态阈值混合负载均衡算法. 该负载策略会根据节点的处理能力来实时自适应地分配任务,以减少服务器与节点的通信代价,比较适合实时流媒体的处理.

3.4 基于用户优先级的 DF 调度策略

本策略中,系统首先会将接收到的流化媒体的视频文件进行缓存,当缓存达到一定大小后则进行分块封装,如图 2 所示,系统会为每个接收到的数据块加上数据块等级和 deadline 时间戳属性,拥有较高等级和较小 deadline 时间戳属性的流媒体数据块将优先被处理.

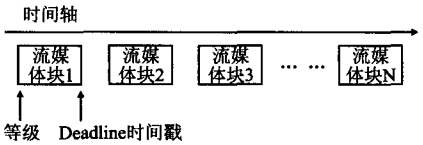

图 2 媒体流化的视频文件分块封装

Fig. 2 File block capsulation of streaming media

当数据缓存达到一定大小后,系统会从实时流媒体缓冲区中获取每个流媒体的部分,然后将其分块封装成数据块,并按照基于堆的动态阈值混合负载均衡算法(见 3.5)分配给相应的节点进行处理,如图 3 所示,每个缓冲区会缓存一个视频文件的一部分,然后将其分发到系统中. 负载方法将考虑到每个节点所剩余的 slot 数量,slot 的数量取决于节点剩余的资源能力情况. slot 数量多的将被分更多的任务,以充分利用节点的计算能力.

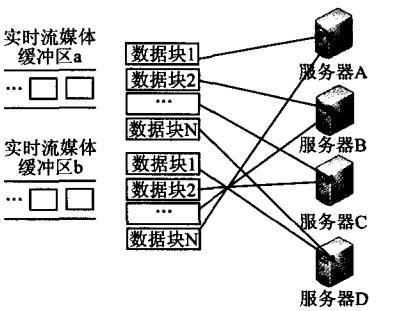

图 3 流媒体缓冲及分布式处理

Fig. 3 Buffer and distributed processing of streaming media

在每个节点上,使用基于用户优先级的 DF(Deadline First)调度算法,算法将考虑数据块优先级和数据块的时限性两方面特征. 在下页图 4 的示例中,包含三种类型的流媒体:TV(电视)流媒体、TEL.(电话)流媒体和 Internet(因特网)流媒体,从用户的角度希望 TV 流媒体的体验优于 TEL. 流媒体和 Internet 流媒体,因此设置其处理优先级高于其他两种. 同一流媒体中,数据块的用户优先级随时间逐渐降低,以便终端优先得到较早的数据块,否则将出现视频延迟和抖动的情况. 截止时间靠前的数据块也应该优先被处理,算法会为每个数

据块根据其实际情况设置 deadline 时间.

基于用户优先级的 DF 算法如下:

1)将来自不同流媒体源的实时流媒体数据按不同的用户优先级权重在 Hadoop 中生成不同类别的作业;

2)在提交形成的 Job 队列中,以 Job 的 deadline 属性进行排序,排在队列前面的 Job 是急需被处理的流媒体数据块,高优先级的作业系统会预留出一部分 slot 对其进行处理;

3)当检测到心跳时,对 Job 队列的前 N 个 Job 进行遍历,查找出本地的数据块任务,如果没有则依次将 Job 分给 TaskTracker.

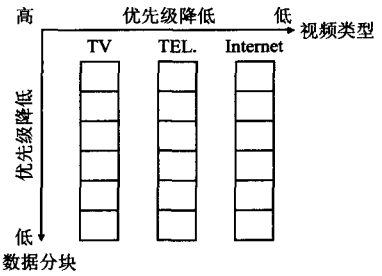

图 4　节点数据块处理示意图

Fig. 4　Data block processing of every node

算法考虑 Job 的优先程度的同时减少了数据块在各个节点的传送,减少由于数据传送占用的网络带宽,从而节省了任务的处理时间,同时也为了配合下节的负载均衡算法使其达到最理想的效果,最终使得分配的每个节点的任务都能尽力在本地执行,达到均衡的效果,并能充分利用不同性能节点使其达到最佳计算性能.

3.5　基于堆的动态阈值混合负载均衡算法

前文中提到的所设计的负载平衡算法称其为基于堆的动态阈值混合负载均衡算法,这种均衡算法适应于节点多并发量大的情况.

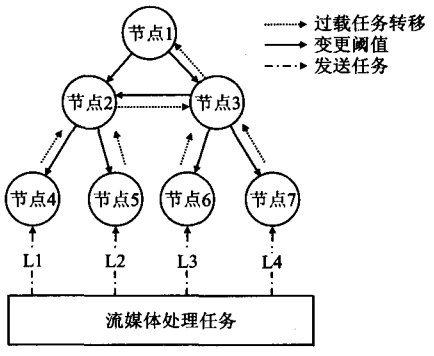

图 5　基于堆的动态阈值混合负载平衡算法示意图

Fig. 5　Heap-based dynamic threshold mixed load balance algorithm

此算法使用二叉大顶堆[18]的结构,如图 5 所示,每个堆的节点代表一台 TaskTracker 用来接收 JobTracker 发送的实时流媒体处理的任务,JobTracker 先将任务发送给 N/2(N 为 TaskTracker 数)个叶子节点,当叶子节点的负载超过其规定的负载阈值时,会将负载转移给其父节点,由其父节点接收传来的数据并执行相应的编转码操作,当父节点及其他中间节点(非根节点)出现负载超过指定阈值时,若过载的节点是左节点则将超过阈值的负载部分转移给其右兄弟节点,同时提高子节点的负载阈值,如果是右节点则将负载转移给父节点,同时过载的节点命令其子节点和左兄弟节点增加阈值. 每隔一定时间 JobTracker 会重新调整大顶堆,此算法的复杂度为 O(n).

如图 5 所示,节点 4、5、6、7 为叶子节点,负责接收 JobTracker 分配的流媒体处理任务 L1、L2、L3、L4,其中任务分配的数量按照节点的负载的反比进行分配,出现过载时将任务转移,即图中虚线所示,当节点出现负载超过阈值的情况则增大子节点或左兄弟节点阈值,即图中实线所示,具体的负载算法如下.

1)JobTracker 每隔 t′时间周期收集 TaskTracker 节点负载信息,并更新堆结构,使得其具有大顶堆性质;

2)JobTracker 负责将实时流媒体处理任务分发给所有叶子节点;

3)当叶子节点出现过载时,则通知 JobTracker,JobTracker 会将待要发送的任务发送给叶子节点的父亲节点;

4)当堆中除叶子节点和根节点外的左节点出现过载时,则通知 JobTracker 将任务发送给右兄弟节点,并通知子节点增加阈值,当右节点出现过载则通知 JobTracker 将任务发送给其父节点,并通知子节点和左兄弟节点增加阈值;

5)当根节点出现过载时,通知子节点增加阈值,其不将过载信息转移给其他 TaskTracker.

上面算法中的每个 TaskTracker 负载数量一般使用公式(1)表示.

$$L = \sum_{i=1}^{n} w_i l_i \tag{1}$$

其中 w 为负载权重,l 为 CPU、内存等占用率,Hadoop 中的负载可以使用 slot 反比来表示,slot 是 Hadoop 用来表示节点剩余的资源数量,所以在此,我们使用 slot 作为每个节点的负载衡量值.

4　实验及性能分析

为验证算法性能,本文进行了仿真实验. 操作系统使用 RedHatEnterpriseLinuxServerrelease5.4,gcc编译器为4.7.1,Hadoop版本为0.20.2,JSVM为9.13. 为了更直观地展示Map任务的节点分配及执行调度过程,系统首先使用四个节点,配置见下页表 1 前四行,其中一台作为 JobTracker,其他三台 A、B、C 作为 TaskTracker. 将相同的任务分别放在原 Hadoop 和改进的 Hadoop 上执行后,程序运行结果分别如图 6

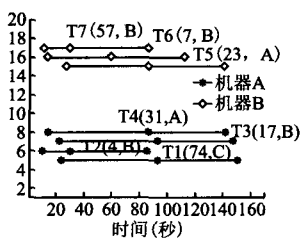

图 6　原 Hadoop 的任务执行顺序及执行时长

Fig. 6　Sequence and duration of original hadoop task

和图 7 所示,其中任务括号内的数字指的是任务的 Deadline,即任务应该在这个时间段内结束,括号内的字母指此任务原始 split 数据所在的节点号. 从图中看到每一条直线上面有三个点,第一个点为任务开始时间,第二个点为 Map 任务的开始时间,即 Job 的实际开始时间,最后一个点为 Map 任务的结束时间.

图 6 以任务的开始时间作为调度依据,其中所有的任务未设置优先级,并以 FIFO 方式调度执行,7 个任务被随机分配到 Hadoop 集群的各个节点上,由于任务分配时心跳消息尚未到达,TaskTracker(C)并没有被分配任务,而节点 A 和 B 被分配了几乎相同数目的任务,此时任务的执行节点和数据所在的节点大多并不相同,因而增加了节点间的数据传送,造成了很大的带宽占用.

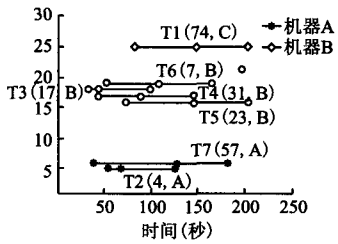

图 7 改进 Hadoop 的任务执行顺序及执行时长

Fig. 7 Sequence and duration of improved hadoop task

图 7 以任务的截止时间作为调度依据,并根据节点能力分配任务,由于节点 B 的配置较高,任务被分配到节点 B 的数量较 A 和 C 多,此时看到任务执行的所在节点和原始数据所在的节点一致,减少了占用网络带宽.

为进一步验证性能,接下来使用 7 个节点进行实验,并考虑节点在机架上的分配,节点配置见表 1.

表 1 节点配置表

Table 1 Node configuration

类型	机架	配置情况
JobTracker	Rack1	PU:8 * 2.0 GHz Mem:4G Ethernet:100M
TaskTracker (A)	Rack2	PU:8 * 2.0 GHz Mem: 2G Ethernet:100M
TaskTracker (B)	Rack3	PU:8 * 2.0 GHz Mem: 4G Ethernet:100M
TaskTracker (C)	Rack4	PU:8 * 2.0 GHz Mem: 2G Ethernet:100M
TaskTracker (D)	Rack2	PU:8 * 2.0 GHz Mem: 4G Ethernet:100M
TaskTracker(E)	Rack3	PU:8 * 2.0 GHz Mem: 2G Ethernet:100M
TaskTracker(F)	Rack4	PU:8 * 2.0 GHz Mem: 2G Ethernet:100M
交换机	-	Ethernet:1000M

实验中对任务设置了优先级,为更好地描述系统性能,本文提出任务平均处理时间 T、任务的处理满意度 S 两个指标. T 是所有任务处理时长的平均值; S 是视频处理结束的期望时长与视频实际处理时长的比值再乘以权重的平均值,其中权重表示任务的优先级,其计算公式如(2)和(3)所示.

$$T=\frac{\sum_{i=1}^{n}(F_i-S_i)}{n} \tag{2}$$

$$S=\frac{\sum_{i=1}^{n}W_i\frac{D_i-S_i}{F_i-S_i}}{n} \tag{3}$$

其中公式(2)中 F 表示任务结束时间,S 表示任务开始时间,n 表示任务数量;公式(3)中 D 表示任务的截止时间,S 表示任务开始时间,F 表示任务结束时间,n 表示任务数量,W 表示优先级权重.

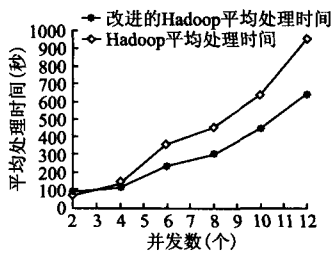

图 8 平均处理时间比较图

Fig. 8 Average processing time

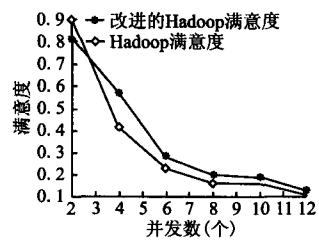

图 9 满意度比较图

Fig. 9 Satisfaction

实验结果如图 8 和图 9 所示,其中图 8 显示任务的平均处理时间 T 随任务并发数目而变化的曲线图,T 总会随着并发数目的增加而变长,说明由于任务的增加,每个节点的排队时长变长,平均处理能力在降低. 然而改进的后 Hadoop 的平均处理时间明显优于原策略,只是在并发数很小时由于当前所有节点的处理能力都有空余,两种策略性能相差无几. 可见改进后的算法和负载策略效率更高.

图 9 是满意度 S 随并发数而变化的示意图. 从图中可以看到,满意度随着并发数量的增多而逐渐变小,这是由于当任务数变多时,节点的处理能力也会随之下降,所以处理时长变长,最终导致满意度下降. 然而随着并发数量的逐渐增多,改进 Hadoop 的满意度下降的趋势明显较原 Hadoop 更加缓和一些,并高于原 Hadoop 的满意度,从而证明使用的策略对满意度有较为有利的影响.

5 总 结

本文面向流媒体的实时的处理,在研究 Hadoop 的原有调度和负载均衡策略的基础上,提出了基于用户优先级的 DF 调度策略和基于堆的动态阈值混合负载均衡算法,通过减少节点间的数据传输,以及充分利用高性能节点,从而得到较低的任务平均处理时间和更好的用户体验,实验结果验证了算法的有效性,证明其适用于流媒体的实时处理. 本文的下一步工作将在真实的网络环境中对系统性能以及用户体验进行研究分析.

References:

[1] Wei Le-ping. Thinking on three network convergence[J]. Telecommunications Science,2010,26(3):1-6.

[2] Wei Le-ping. Networks convergence and development & challenges

of IPTV[J]. Telecommunications Science,2006,22(7):1-5.
[3] Zhu Hai-yi,Zhou Chun-nan. A brief description of VoIP basic principles[J]. Information Technology,2003,27(5):83-84.
[4] Li Ying,Wei Ying-qi,Tan Hua,et al. Discussion on integrating 2G voice communication and 3G video service[J]. Telecommunications Science,2010,26(3):31-34.
[5] Cranor C D,Green M,Kalmanek C,et al. Enhanced streaming services in a content distribution network[J]. IEEE Internet Computing,2001,5(4):66-75.
[6] Xiang Zhe, Zhang Qian, Zhu Wen-wu, et al. Peer-to-peer based multimedia distribution service[J]. IEEE Transactions on Multimedia,2004,6(2):343-355.
[7] Liu Dang-hui,Shen Lan-sun. DSP chip and its aapplication in image [J]. Measurement & Control Technology,2001,20(5):16-23.
[8] Owens J D,Luebke D,Govindaraju N,et al. A survey of general-purpose computation on graphics hardware[J]. Compute Graph Forum,2007,26(1):80-113.
[9] Fang Sheng,Zhong Yu-zhuo. Advances in distributed video coding [J]. Computer Engineering and Applications,2005(21):45-48.
[10] Geng Chen-yao,Yao Dan-ya,Zhang Ying-ying,et al. Distributed video processing platform based on map reduce[J]. Computer Engineering,2012,38(10):280-283.
[11] Yang Fan,Shen Qi-wei. Distributed video transcoding on hadoop [J]. Computer Systems & Applications,2011,20(11):80-85.
[12] Zhu Wen-wu,Luo Chong,Wang Jian-feng,et al. Multimedia cloud computing[J]. Signal Processing Magazine,IEEE,2011,28(3):59-69.
[13] Jeffrey D, Sanjay G. MapReduce: simplified data processing on large clusters[J]. Communications of the ACM,2008,51(1):107-113.
[14] Nicolae B, Moise D, Antoniu G, et al. BlobSeer: bringing high throughput under heavy concurrency to hadoop map-reduce applications[C]. Parallel & Distributed Processing (IPDPS), 2010:1-11.
[15] Sun Lian-sheng,Wang Lei,Ding Cai-wei. LandRoom:a distributed storage system that improves LandHouse's throughput under high concurrency[C]. The 31st Chinese Control Conference Hefei China,2012:5481-5486.
[16] Schwarz H,Marpe D,Thomas W. Overview of the scalable video coding extension of the H.264/AVC standard[J]. IEEE Transactions on CSVT,2007,17(9):1103-1120.
[17] Khalifa, Issa, Olanrewaju, et al. Development of a scalable video compression algorithm[C]. Communication,Networking & Broadcasting;Components, Circuits, Devices & Systems;Computing & Processing (Hardware/Software);Engineered Materials, Dielectrics & Plasmas;Fields,Waves & Electromagnetics,2012:755-759.
[18] Thomas H Cormen,Charles E Leiserson, Ronald L,et al. Introduction to algorithms[M]. Beijing:China Machine Press,2006.

附中文参考文献:

[1] 韦乐平. 三网融合的思考[J]. 电信科学,2010,26(3):1-6.
[2] 韦乐平. 三网融合与 IPTV 的发展和挑战[J]. 电信科学,2006,22(7):1-5.
[3] 朱海毅,周春楠. VoIP 基本原理[J]. 信息技术,2003,27(5):83-84.
[4] 李　颖,魏颖琪,谭　华,等. 2G 语音通信和 3G 视频业务融合探讨[J]. 电信科学,2010,26(3):31-34.
[7] 刘党辉,沈兰荪. DSP 芯片及其在图像技术中的应用[J]. 测控技术,2001,20(5):16-23.
[9] 房　胜,钟玉琢. 分布式视频编解码技术的研究进展[J]. 计算机工程与应用,2005,41(21):45-48.
[10] 耿晨曜,姚丹亚,张盈盈,等. 基于 Map Reduce 的分布式视频处理平台[J]. 计算机工程,2012,38(10):280-283.
[11] 杨　帆,沈奇威. 分布式系统 Hadoop 平台的视频转码[J]. 计算机工程与应用,2011,20(11):80-85.
[15] 孙连生,王　雷,丁才伟. LandRoom:一种改进 LandHouse 支持高并发高吞吐量的分布式存储系统[C]. 第三十一届中国控制会议论文集 C 卷,2012:5481-5486.