分布式数据流处理系统的动态负载平衡技术*)

邓华锋[1] 刘云生[1] 肖迎元[2]

(华中科技大学计算机学院 武汉 430074)[1] (天津理工大学计算机科学与工程系 天津 300191)[2]

摘 要 设计了一种新的大规模分布式数据流处理系统的体系结构。系统由一组异构的服务器集群组成,负载在每个服务器集群内部多台同构的服务器之间获得平衡,从而达到整个系统的负载平衡。集群设计的主要目标之一是以资源换性能,服务器集群中服务器的最大数目足够保证系统不再发生过载现象,不再需要会降低性能的卸载技术。而且投入运行的服务器的数目根据实际的系统负载来决定,负载较轻时,一部分服务器可以进入休眠状态来减少能源的消耗。根据系统动态增减服务器的特点,设计了全新的初始化算法、动态负载平衡算法。与以前的分布式数据流处理系统相比,由于单个集群的服务器的数目大大减少,算法复杂性降低、速度加快、优化的空间增大。

关键词 分布式数据处理流系统,动态负载平衡,卸载,节能

Dynamic Load Balancing Techniques for the Distributed Stream Processing Systems

DENG Hua-Feng[1] LIU Yun-Sheng[1] XIAO Ying-Yuan[2]

(School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074)[1]

(Department of Computer Science and Engineering, Tianjin University of Technology, Tianjin 300191)[2]

Abstract In the novel architecture for the large-scale distributed stream processing system, the whole system consists of a group of heterogeneous computer clusters. The whole system can achieve the global load balancing by balancing every cluster which consists of several homogeneous servers. The main goal of every cluster is exchanging the resources for the performance. In the cluster, enough servers are employed to get rid of the occurrence of overload phenomenon, so techniques for load shedding are not still necessary in the system. In the meanwhile, the number of active servers is decided by the practical load level and some servers can be put into the sleep mode for the sake of energy conservation when the load is rather low. The band-new initialization algorithm and dynamic load balancing algorithm are designed to accommodate the characteristic of increasing or decreasing the servers dynamically. Comparing to the traditional large-scale stream systems, these algorithms have better load balancing, lower complexity and faster response time because the number of servers in a single cluster is reduced sharply.

Keywords Distributed stream processing system, Dynamic load balancing, Load shedding, Energy conservation

1 引言

数据流处理系统广泛应用在众多领域,例如金融管理、网络监视、通信数据管理、Web 应用、传感器网络数据处理等。这些应用中有一个典型的特点:数据流数据量极大,具有相当高的突发性[1];当数据到达的速度超出系统的处理能力时,系统会出现过载并且性能下降。所以,无论是集中式数据流处理系统,还是分布式数据流处理系统,负载管理均成为研究的热点与重点[1,2]。

在网络和多媒体研究领域,对负载管理问题的研究,与数据流的负载管理有许多相似之处,但存在着本质上的区别[1,3]。对集中式数据流处理系统的负载管理提出的解决办法是局部算子或整个系统的卸载[3~5]。采用这种方法,检测到系统发生超载时,选择性地丢掉一些元组来保证系统的运行。这种方法的缺点是显而易见的,尽管在丢弃元组的时候要考虑系统的质量要求,但是显然系统的性能不可避免地会受到影响。由于流数据源及应用本身存在分布的特点,分布式流处理系统成为流处理研究的热点[1,8]。在分布式流处理系统中,研究集中在如何将负载均衡地分布到各个服务器并保持系统的负载动态平衡。文[7]研究了在进行分布式流处理系统卸载时,如何协调各个服务器结点,获取全局最优。文[9]提出一种分布式联邦流处理系统动态负载平衡算法,系统中各结点只在有协议的邻居之间平衡负载,并不寻求全局最优。文[2]中,流处理功能被封装成一项项服务,负载平衡考虑的是从能够提供服务的服务器中挑选合适的一组来处理每个动态提交的应用,主要用于多媒体流数据的分发与处理。文[6]对动态负载平衡进行了初步的探讨,相邻的算子尽量分布在同一服务器,以减少网络数据传输。以上的分布式流处理系统要求系统的各个结点是同构的,因为异构的系统之间迁移算子往往特别困难,甚至是不可能的。这些系统能消化的负载也相当有限,在应用中,往往清闲的时候所有服务器清闲,过载的时候所有服务器过载。

在处理器等硬件越来越便宜的今天,越来越多地通过增加服务器组成集群的方法出现在网络和多媒体领域。在分布式流处理领域,文[1]提出一种利用服务器集群处理负载过载问题的新方案。在该方案中,流数据的处理由一组由高速网

*)国家自然科学基金项目(60073045)资助;国家"十五"国防预研基金项目(413150403)资助。邓华锋 博士研究生,主要研究方向为现代数据库技术和流数据算法。

络互连的无共享的同构服务器来共同承担。由于系统的复杂性，负载的近似平衡由一组贪婪算法来保证，算法同时还尽可能地保持系统的负载的变化最小。但是在大规模流数据处理的情况下，由于系统结构和算法的复杂性，平衡能力依然有限，只能避免过载现象的发生，而不能完全消除它的发生。而且该方法过于理想，实现的算法并不能应用于实际的分布式系统。如系统假定所有服务器之间通过高速网络连接、服务器之间传输数据忽略不计、服务器是同构的等。这些条件，实际的系统很难满足。

在以上研究的基础上，我们提出了一种系统的新方案来解决以上提出的种种问题。在我们的分布式流处理系统中，一组异构服务器集群以对等的方式工作，每个服务器集群处理一个子领域的流数据，集群之间可以合作完成查询，但是流数据处理负载不在服务器集群之间迁移。每个服务器集群由多个同构的服务器组成，在将要出现过载的时候，集群可以激活备用服务器。只要服务器数目足够，系统就不会过载，不再需要会降低性能的卸载技术；而在系统负载很轻时，可以让一些服务器进入“睡眠”模式。这样，既保证了服务质量，又避免了资源的浪费。考虑到各个子领域的负载不一致，各个集群最大服务器数目也可以不一样。这样，分布式系统的负载平衡问题就转变为单个服务器集群服务器负载平衡问题。所以，本文的研究成果也可以应用到集中式流数据处理系统。在整个分布式流数据处理系统分成若干个子领域后，每个领域的服务器集群服务器数目大大减少，降低了负载平衡算法研究的复杂性，为研究最优算法或更好的近似算法提供了可能，从而进一步提高了系统性能。

2 问题的描述

2.1 系统模型

我们的分布式流处理系统由一组以 P2P 方式合作的服务器集群组成。每个集群中服务器数目不等，但是构成与工作原理相同。这样，我们研究的重心就集中在单个服务器集群上。单个服务器集群系统模型如图 1 所示。在采用服务器时，我们采用的是同构服务器。所有的同构服务器（CPU、内存、流处理引擎等一样）通过高速局域网互连，所以不再考虑网络的带宽的限制。在系统开始运行时，根据系统的负载，计算需要的服务器数目，从中任意挑选一台服务器作为调度服务器。各个服务器定期负责向调度服务器报告自己的负载情况。

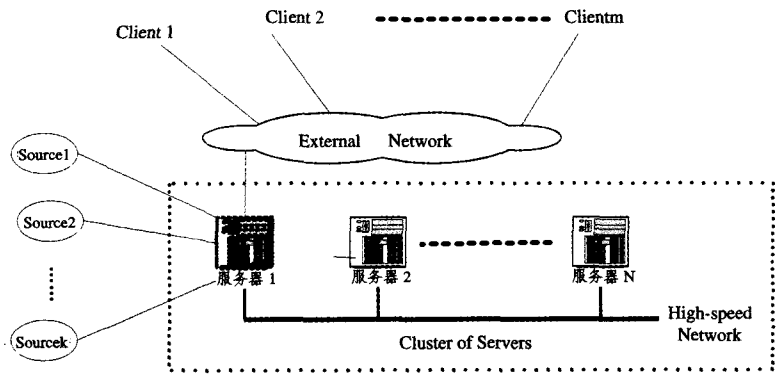

图 1 集群的体系结构

调度服务器在系统负载较轻的时候，负责全部流数据的处理与负载监控、负载预测；在负载超过一定阈值时，启动新的服务器，迁移一部分负载，并将工作中心转移到负载监控与数据流和查询的转发工作，减少其进行流数据处理的负载。当系统负载不均衡但可以在系统内平衡时，调用系统动态负载平衡算法，得出系统的最优调整方案，组织负载的迁移。当负载过轻时，调度服务器决定系统的最优负载调整方案，组织负载的迁移，减少服务器的数目。其他的工作服务器只负责流数据的处理与输出数据的转发。

2.2 负载的计算

流处理系统的查询网络由一系列流算子组成无循环的流水线形式。由于集群中所有的同构服务器通过高速局域网互连，网络带宽不再作为受限制的资源，只考虑 CPU 的占用。我们把处理的时间分成定长的时间片段（T_0，T_1，T_2，……），算子的负载为其处理在每一个时间片段内到达的元组需要的 CPU 时间。我们并不把统计得到的算子的负载作为计算的依据，而是作为使用一些预测技术（灰色预测、指数平滑、移动平均等）来估计将来一段时间的负载的依据。根据若干时间片段的算子的实际负载，通过采用预测技术得到 K 个预测值，记为 L_1，L_2，…，L_K。

定义 1（算子计算负载） 算子 O_i 的计算负载等于以上 K 个预测值的最大值，记为 $L(O_i)$。

$$L(O_i)=\mathrm{MAX}\{L_1,L_2,\cdots,L_K \mid 1\leqslant i\leqslant K\} \tag{1}$$

取最大值的原因是因为负载经常波动。要保证在任何负载的情况下，都不允许系统过载。

定义 2（工作服务器的实际负载） 工作服务器 S_i 的实际负载等于服务器的空荷负载（LE）加上工作服务器上所有算子的负载之和，记为 $LR(S_i)$。服务器的空荷负载是指服务器启动就绪，准备处理流数据前所承担的负荷，也就是运行操作系统、流处理引擎等所要消耗的 CPU 时间。

$$LR(S_i)=LE(S_i)+\sum_{j=1}^{M}L(O_j) \tag{2}$$

定义 3（服务器最大工作能力） 服务器最大工作能力是指一段时间内服务器能够提供服务的时间长度，记为 $P(S_i)$。

定义 4（调度服务器的负载） 调度服务器的负载等于服务器的空荷负载、所有算子的负载、接收用户请求的负载（LA）、接收与转发流数据的负载（LD）、运行平衡算法的负载（LS）之和：

$$LR(S_s)=LS(S_s)+\sum_{j=1}^{M}L(O_j)+LA(S_s)+LD(S_s)+LS(S_s) \tag{3}$$

3 负载平衡策略

3.1 算法的目标

我们系统的出发点是以资源换取系统的性能，所以优化的目的与其它的系统有所不同。我们系统首先追求的是系统性能的保证，不允许出现过载现象。在保证不出现过载的前提下，保持系统负载的平衡分布，保证使用的服务器数目最小化。优化的目标影响着负载平衡算法的设计。

假设系统中有 N 个服务器，每个服务器的负载分别为 $LR(S_i)$，我们的算法的目标就是：

(1)$LR(S_i)\langle C_o * P(S_i)\ 1\leqslant i\leqslant N$；

(2)$LR(S_1)\approx LR(S_2)\approx\cdots\approx LR(S_N)$；

(3)minimize N_u。

其中，C_o 是服务器的负荷安全预警系数，取值在 0.7～0.9 之间。系统负载波动越大，C_o 的取值越小，表示系统预警和处理过载的时间越要提前。由于算子的迁移需要时间，不能因为算子的迁移过度降低系统的性能。

3.2 负载的监控

集群内工作服务器定期向调度服务器报告自己的负载情况，调度服务器先检查系统内的服务器是否过载或平衡。单服务器时只检查过载，其它情况需要检查负载是否平衡。

3.2.1 过载预警与处理

$$LR(S_i)\geqslant C_o * P(S_i),1\leqslant i\leqslant n \tag{4}$$

如果 $C_o=0.8$ 表示当服务器的实际负载超过服务器处理能力的 80%时，可以认为系统有过载危险，需要发出过载预警信号，系统进入过载处理流程。首先判断在目前负载水平下，负载是否能够平衡。不能平衡则增加服务器，然后调用动态负载平衡算法；能够平衡则直接调用动态负载平衡算法。

3.2.2 负载不平衡的检测与处理

当系统中有超过 1 台服务器在工作时，需要定期检测系统负载是否平衡。

$$|L(S_i)-L(S_j)|>\delta,1\leqslant i,j\leqslant N \tag{5}$$

其中，δ 是启动负载调整的预定的门槛值，只有当结点之间的负载差值大于 δ 时才启动动态负载平衡算法调整各服务器的负载。

3.2.3 负载过轻的检测与处理

$$\sum_{i=1}^{N}LR(S_i)<C_l * \sum_{i=1}^{N}P(S_i) \tag{6}$$

其中，C_l 为轻载常数，在 0.5 左右。当上式成立时，可以认为系统负载过轻。负载过轻时，计算需要的服务器数目，决定负载迁移方案。在负载迁移后，减少服务器数目，再启动动态负载平衡算法调整各服务器的负载。需要的服务器数目由下式计算：

$$N_r<C_{ls}\frac{\sum_{i=1}^{N}(P(S_i))}{\sum_{i=1}^{N}LR(S_i)} \tag{7}$$

其中，C_{ls} 为安全系数。$N_r>1$，系统中至少需要一台服务器。

4 负载平衡算法

4.1 动态平衡算法

系统在初始化阶段或在检测到系统内负载不平衡时（$N>2$），需要调用动态负载平衡算法，来决定系统的负载迁移方案。算法的基本思路是将 N 个服务器分成 $N/2$ 组，负载最小的服务器与负载最大的服务器在一组，负载次小的与负载次大的在一组，依次类推。在组内通过算子的迁移，将负载重的服务器上负载大的算子与负载轻的服务器上的负载较小的算子交换。每一次负载迁移都或多或少地减少服务器之间的负载不平衡。

算法 1 动态负载平衡算法

输入：服务器数目（$N>2$）、算子数目 M、算子的负载数组 $L[M]$ 以及算子在各服务器的分布数组（$SL[N][N/M+N]$）；
输出：平衡的服务器负载分布数组（$SL'[N][N/M+N]$）.

```
Algorithm Dynamic_Bal
Begin
  ①GetInitLoad(LR,SL,N,M);
  //得到各服务器初始的负载
  ②While (Changed)
  { //只要后续的服务器之间有算子的转移，服务器按照负载重新排
    序将有变化，则需要重新平衡服务器之间的负载
  ③Sort_Up(LR, N);
  // 将系统 N 个服务器按照负载轻重从小到大排列；
  ④DivideGroup(LR, N/2);
  //将 N 个服务器分成 N/2 组，负载最小的服务器与负载最大的服
    务器在一组，负载次小的与负载次大的在一组，依次类推。在组
    内通过算子的迁移，将负载重的服务器上负载大的算子与负载轻
    的服务器上的负载较轻的算子交换。
  ⑤For(i=0; i<N/2; i++)
    ⑥InPair(O(hi),O(li));
    //将负载较重的服务器上的算子与负载较轻服务器上的算子两
      两结成对
  ⑦While( continue)
  //有可以考察的候选算子对，则继续
  { ⑧SelectCand();
  // 根据下面的条件确定侯选算子对；
  // L(Ohi)>L(Oli),L(Ohi-L(Oli)<LR(Si)-LR(Si)
  ⑨If(NumofSelctCand==0)
    Exit(-1);
    //候选算子对的数目为 0，则算法结束；
  ⑩Else SelectBest();
  //根据以下标准确定最优算子对：计算每一对算子的 Score，Score
    最小的一组算子成为最优算子对；
  //Score(Ohi,Oli)=abs((L(Ohi-L(Oli))-LR(si)-LR(sj)/2)
  ⑪ExchangeBest();
  //最优算子对发生交换，更新算子的负载数组 L[M]以及算子在各
    服务器的分布数组 SL[N][N/M+N],Changed=true;
  ⑫DeleteSomeCan();
  //剔除候选算子对中，其中一个或两个算子已经被最优算子对中选
    择的算子对
  ⑬If(NoExsistCand())
    continue=false;
    //还有可以考察的候选算子对，则继续，否则，算法结束
    }
  }
End
```

4.2 系统初始化算法

系统初始化算法用来在集群启动时平衡各 N 个服务器之间的负载。集群在正式运行前，需要试运行来获得有关负载计算所需要的负载的统计值，根据式(7)计算需要的服务器数目。实现 N 个服务器的负载的最优平衡是个 NP 完全问题。所以，我们的系统初始化算法首先采用贪婪算法获取最初的分配。考虑到贪婪算法尽管计算速度快，但是我们在系统实际运行中往往平衡效果不好。所以，我们又在贪婪算法基础上调用上面的动态负载平衡算法，使负载进一步平衡，具体如下。

准备条件：

算法 2 系统初始化算法

输入：服务器数目（$N>2$），算子数目 M 以及算子的负载数组 $L[M]$；
输出：算子在各服务器的分布数组（$SL[N][N/M+N]$）.

```
Algorithm Greed_Bal(L[M], N, M)
Begin
①Sort_Down(L[M], M);
  //将算子按照负载从大到小排序
②For( i=0; i<M; i++)
      ③Sort_up(LR, N);
        // 将服务器按负载从小到大排序，LR 为服务器负载数组；
      ④Insert(SL[N][ N/M+ N]);
        //将未分配的负载最大的算子分配到负载最轻的服务器
      ⑤Update(LR);
        //更新服务器的负载
  EndFor
⑥Invoke Dynamic_Bal(SL, N, M);
  //调用动态负载平衡算法进一步平衡 。
End
```

5 实验

实验环境是一台具有 P Ⅳ 2.4GHz 处理器和 1GB 内存的 PC 工作站。运行 Microsoft Windows NT 系统，算法测试程序、流处理系统采用 C++ 编写。

实验 1 初始化算法的性能

负载采用为 1 到 1000 的随机数作为模拟负载数据。我们对服务器数目从 2 到 5、算子数目从 12 到 30 的各种组合进行了测试，每种组合重复 10 次取平均值。由于篇幅关系，我们仅提供部分数据。图 2 为贪婪法(Greed)、我们改进的算法(Greed_Bal)、最优算法(Opt)分别在 4 台、5 台服务器情况下，各自生成初始负载分配方案后系统负载分布的变异系数。由于最优算法的计算时间太长，我们仅提供了算子数目从 12 到 20 的最优算法的时间与负载分布的标准差。从图 2 中可以看出，在大多数情况下，Greed_Bal 的效果与最优化算法的时间比较接近，而纯粹的贪婪法效果比较差。随着算子数目的增多，Greed_Bal 效果明显优于纯粹的贪婪法，而纯粹的贪婪法得出的负载分配方案变异系数在 0.1 到 0.3 之间。也就是说，服务器负载相对负载平均值之间的变化幅度大概为平均值的 10 %到 30 %，所以该方法基本上不能用。

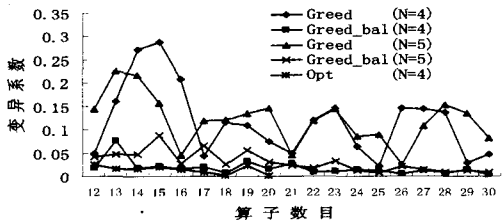

图 2 初始化算法的负载平衡变异系数

图 3 为贪婪法(Greed)与我们改进的算法(Greed_Bal)所消耗的时间的比较。由于最优化算法所消耗的时间远远大于以上两种算法所消耗的时间，最优化算法的时间单独在图 4 表示。从图 3 可以看出，贪婪法(Greed)与算法(Greed_Bal)所消耗的时间差别不大，Greey_Bal 大 Greed 1～2μs 时间。多用 1～2μs 获得与最优算法大致一样的效果是值得的。

实验表明：贪婪法(Greed)与算法(Greed_Bal)所消耗的时间差别不大；Greed_Bal 效果明显优于纯粹的贪婪法。

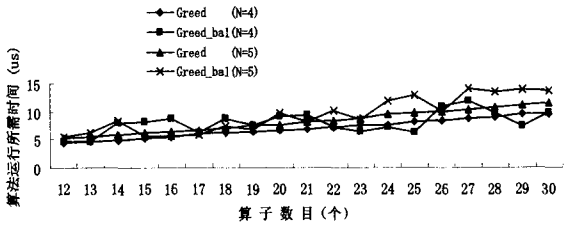

图 3 贪婪算法的运行时间

实验 2 占用的机器时间

实验中采用 5 台与实验 1 相同的服务器组成一个集群。传统的分布式流处理系统为 5 台一起用。在我们的系统中，只有在需要的时候才启用服务器。数据集是 3 个小时的广域 TCP 网络交通轨迹(Internet traffic archive)，可以根据文[10]下载，实验中流算子数目为 50。图 5 为系统中服务器的实际使用情况。测试的时间为 3h。实际占用的总机时大约为 480min，大概为 5 台同时使用情况下所占用机时的一半(0.53)。

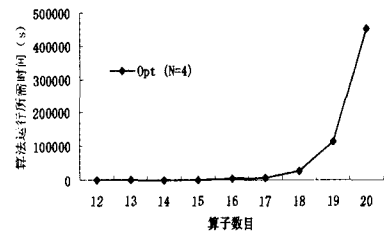

图 4 最优算法的运行时间

结论 我们设计了一种新的大规模分布式流处理系统的体系结构，系统由一组异构的服务器集群组成，每个服务器集群内部负载在各个同构的服务器之间获得平衡。实际运行服务器的数目根据实际的系统负载来决定，负载较轻时，一部分服务器可以进入休眠状态来减少能源的消耗。服务器集群中服务器的最大数目足够保证系统不再发生过载现象，不再需要会降低性能的集中式流处理系统中的卸载技术，为流处理系统的负载管理提供了一条新的思路。系统不再要求所有服务器同构，只需要同一集群内服务器同构，对系统的假设比较符合实际应用的特点。根据系统动态增减服务器的特点，我们设计了全新的初始化算法，动态负载平衡算法。与以前的分布式流处理系统相比，由于采用新的体系结构，单个集群的服务器的数目大大减少，算法的速度加快，复杂性降低，优化的空间增大。我们下一步将对以上算法进行优化。

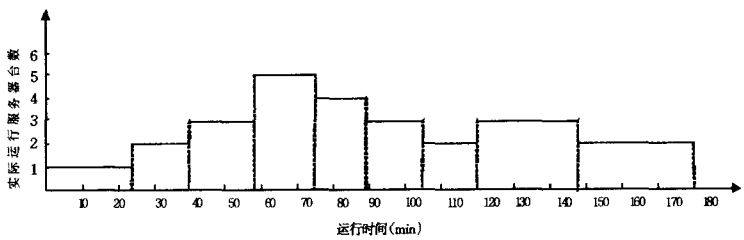

图 5 实际运行中机器占用情况

参 考 文 献

1 Xing Y, Zdonik S, Hwang J-H. Dynamic Load Distribution in the Borealis Stream Processor. In: ICDE, 2005

2 Cardellini V, Colajanni M, Yu P S. Load Balancing Techniques for Distributed Stream Processing Applications in Overlay Environments. In: Ninth IEEE International Symposium on Object and Component-oriented Real-time Distributed Computing, 2006

3 韩东红，王国仁. 数据流系统中卸载技术研究综述. 计算机科学，2005，32(8)：102～105

4 Tatbul N, Cetintemel U , et al. Load Shedding in a Data Stream Manager. In: VLDB, 2003

5 Babcock B, Datar M , Motwani R. Load Shedding for Aggregation Queries over Data Streams. In: ICDE, 2004

6 Xing Y. Load Distribution for Distributed Stream Processing. In: Proc. of the ICDE Ph D Workshop, 2004

7 Tatbul N, Zdonik S. Dealing with Overload in Distributed Stream Processing Systems. In: ICDEW'06, 2006

8 Zhou Y L, Chin B, Tan K L. Dynamic Load Management for Distributed Continuous Query Systems. In: ICDE, 2005

9 Balazinska M, Balakrishnan H, Stonebraker M. Contract-based load management in Federated Distributed Systems. In: USENIX Symposium on Net-worked Systems Design and Implementation (NSDI), March 2004

10 Internet Traffic Archive , trace LBL-TCP-3. http:/ / www. acm. org/ sigcomm/ ITA/, 2006