·体系结构与软件技术·　　文章编号：1000-3428(2022)03-0197-07　　文献标志码：A　　中图分类号：TP393

基于异构Flink集群的节点优先级调度策略

汪文豪[1]，史雪荣[2]

（1.南京工业大学 计算机科学与技术学院，南京 211816；2.盐城师范学院 数学与统计学院，江苏 盐城 224002）

摘　要：Flink流处理系统默认的任务调度策略在一定程度上忽略了集群异构和节点可用资源，导致集群整体负载不均衡。研究分布式节点的实时性能和集群作业环境，根据实际作业环境的异构分布情况，设计结合异构Flink集群的节点优先级调整方法，以基于Ganglia可扩展分布式集群资源监控系统的集群信息为依据，动态调整适应当前作业环境的节点优先级指数。基于此提出Flink节点动态自适应调度策略，通过实时监测节点的异构状况，并在任务执行过程中根据实时作业环境更新节点优先级指数，为系统任务找到最佳的执行节点完成任务分配。实验结果表明，相比于Flink默认的任务调度策略，基于节点优先级调整方法的自适应调度策略在WorldCount基准测试中的运行时间约平均减少6%，可使异构Flink集群在保持集群低延迟的同时，节点资源利用率和任务执行效率更高。

关键词：Flink集群；异构集群；负载不均衡；节点优先级；自适应调度

开放科学（资源服务）标志码（OSID）：

中文引用格式：汪文豪，史雪荣.基于异构Flink集群的节点优先级调度策略[J].计算机工程，2022，48(3)：197-203.

英文引用格式：WANG W H，SHI X R.Node priority scheduling strategy based on heterogeneous Flink cluster[J].Computer Engineering，2022，48(3)：197-203.

Node Priority Scheduling Strategy Based on Heterogeneous Flink Cluster

WANG Wenhao[1]，SHI Xuerong[2]

（1.College of Computer Science and Technology，Nanjing Tech University，Nanjing 211816，China；
2.College of Mathematics and Statistics，Yancheng Teachers University，Yancheng，Jiangsu 224002，China）

【Abstract】The default task scheduling strategy of the Flink stream processing system ignores the cluster heterogeneity and available resources of nodes to a certain extent，resulting in an unbalanced overall cluster load. This study investigates the real-time performance of distributed nodes and the cluster operation environment and designs a node priority-adjustment method based on heterogeneous Flink clusters according to the heterogeneous distribution problem of the actual operation environment.The method dynamically adjusts the node priority index that adapts to the current operating environment based on the cluster information of the Ganglia scalable distributed cluster resource-monitoring system. Based on this，a dynamic adaptive scheduling strategy for link nodes is proposed. By monitoring the heterogeneous status of nodes in real time and updating the node priority index according to the real-time working environment during the task execution process，the best execution node for the system task to complete the task assignment can be found.The experimental results show that compared with Flink's default task scheduling strategy，the adaptive scheduling strategy based on the node priority adjustment method reduces the running time of the WorldCount benchmark by approximately 6% on average.This enables the heterogeneous Flink cluster to maintain low cluster latency while maintaining higher node resource utilization and task execution efficiency.

【Key words】Flink cluster；heterogeneous cluster；load unbalancing；node priority；adaptive scheduling

DOI：10. 19678/j. issn. 1000-3428. 0059944

0　概述

近年来，云计算、大数据、物联网、人工智能等新一代信息技术快速发展，信息产业的数据量急剧增长。截止到2020年底，我国的数据总量预计到达8 060 EB[1]。随着信息数据量剧增，数据处理技术也在发生着巨大变化，涌现出Apache Hadoop、Apache Spark、Apache Storm、Heron、Flink等[2]一大批数据计

基金项目：国家自然科学基金（11872327）；江苏省高等学校自然科学研究项目（20KJA190001）。

作者简介：汪文豪（1997—），男，硕士研究生，主研方向为分布式应用；史雪荣，教授、博士。

收稿日期：2020-11-09　　**修回日期：**2021-02-20　**E-mail：**18251919068@163.com

算系统。在数据处理的硬件方面,各种机构和设备的更新换代和新兴技术的引入,使得各个数据计算系统在实际生产环境中变得越来越异构[3-4]。这种异构集群特征直观表现在其节点的CPU、内存等各方面存在差异致使集群运行时的处理能力不尽相同,从而使得集群在调度节点执行任务时,集群效率明显下降[5]。

目前,国内外众多学者对资源弹性管理问题和分布式框架下的任务及节点调度问题展开研究。在资源弹性管理方面:文献[6]提出使用排队理论进行建模,通过部署主动弹性控制器和反应弹性控制器相结合的模型来估计集群节点的未来负载;文献[7]认为云计算存在资源倾斜的问题,提出动态实时云框架,使集群可以自动适应不同的工作负载,并根据需求重新分配资源;文献[8]基于云计算通用工作负载预测器,设计一种可以为不同工作负载提供高精度预测的长期记忆模型,解决了通用预测器精度不够的问题,进一步优化了集群资源弹性管理。在批处理框架方面:文献[9]认为Hadoop框架在默认调度时未考虑Map与Reduce之间存在差异性,据此提出一种基于Hadoop框架的截止时间约束的扩展MapReduce任务调度算法;文献[10]认为在Hadoop默认调度方式下缺乏对整体集群异构性的考虑,提出基于资源感知与资源异构的云计算环境任务调度算法;文献[11]基于Hadoop框架的节点计算能力,设计能够按计算能力分配数据块的数据局部性调度器;文献[12]认为在异构Hadoop集群中会出现节点随任务动态分配而产生性能差异的问题,提出基于动态工作负载调整的任务调度策略;文献[13]在YARN资源调度器的基础上,结合闭环反馈控制方法,使Hadoop集群在运行时可以动态对MapReduce作业数进行控制,避免出现主观性的问题;文献[14]基于异构Spark集群,提出通过监测节点资源的自适应动态节点任务调度策略,但是该策略依赖于人工设定的阈值,无法体现客观性;文献[15]在文献[14]的基础上,提出基于异构节点优先级的Spark动态自适应调度算法,该算法能够根据实时节点优先级完成调度,但是忽略了任务种类和集群整体工作环境,没有彻底解决主观性问题。在流处理框架方面:文献[16]提出在流处理框架Storm on YARN上构建一种双层调度模型,通过对流数据处理的实时监测,做出合理的资源分配预测,解决框架需要人工干预调整的问题;文献[17]针对流处理框架的数据拓扑中任务间通信开销较大的问题,提出Flink框架下的拓扑关键路径模型,该模型能够确保关键路径上节点负载差异较小的同时最小化任务的节点间通信开销。

但是现阶段国内外的相关研究主要存在两方面的问题。一方面,大部分的分布式调度研究主要关注于批处理,将Hadoop框架与Spark框架作为主要的实验环境,而对于流处理,尤其是以Flink框架作为实验环境的研究相对较少。另一方面,在实际作业环境中,很难避免整体集群出现异构的情况,针对结合实际作业环境的异构Flink环境的研究也相对较少。本文通过研究分布式节点的实时性能和集群的工作环境预测机制,充分考虑现实生产环境的异构分布问题,从而对Flink底层默认调度策略进行优化,以提高Flink系统的工作效率。

1 相关技术

Apache Flink是一个面向数据流的有状态计算框架,核心是一个流数据的处理引擎。在计算过程中,Flink将所有任务当作流来处理,批处理任务被看作具备有限边界的特殊数据流。相对于目前的大部分流处理框架,Flink框架在数据处理方面有低延迟、高吞吐的优势,在集群功能方面则提供了消息传递、状态管理、容错恢复等一系列服务[18-19]。

Flink集群主要由两个重要进程JobManager和TaskManager组成,如图1所示,其中虚线表示任务的流转。JobManager又称为Master,主要协调整体架构的数据流图的运算和执行,其中的调度器模块和Checkpoint协调器模块还负责调度任务、协调检查和故障恢复。TaskManager又称为Worker,主要执行由JobManager分配的任务。同时,每一个TaskManager都具有缓存数据和节点间通信的功能。在Flink中,一个TaskManager即为一个JVM进程,JVM进程允许并行地执行子任务,并能够指定若干slot。每一个slot可以执行若干子任务,即运行若干线程[18]。

Flink在调度任务分配slot时,遵循2个原则:1)同一Job中的同一分组的不同task可以共享同一slot;2)任务调度按照拓扑顺序从Source调度到Sink。以图2(a)的Flink拓扑模型为例,各顶点v表示Flink中的Operator算子,v_a表示Source算子组件,v_b、v_c和v_d表示并行度为3的Transformation算子组件,v_e和v_f表示并行度为2的Sink算子组件。首先由Source组件将读取的数据发送至Transformation组件;然后Transformation组件负责处理上游发送的数据;最后由Sink组件接收上游处理结果并持久化至数据库[20]。以有2个TaskManager的Flink分布式集群为例,每一个TaskManager配置有2个slot。Flink在默认调度下对于该任务拓扑的slot分配模型如图2(b)所示。首先调度默认从拓扑的Source任务开始,将v_a随机调度给任一slot,如图2(b)中v_a调度至slot11所示;然后将v_b、v_c和v_d调度至任意的slot,但v_b、v_c和v_d同属于一种Operator算子,不能共享同一slot,因此将v_b、v_c和v_d分别调度至slot11、slot12和slot21;最后将v_e和v_f也分别调度至slot11和slot12。

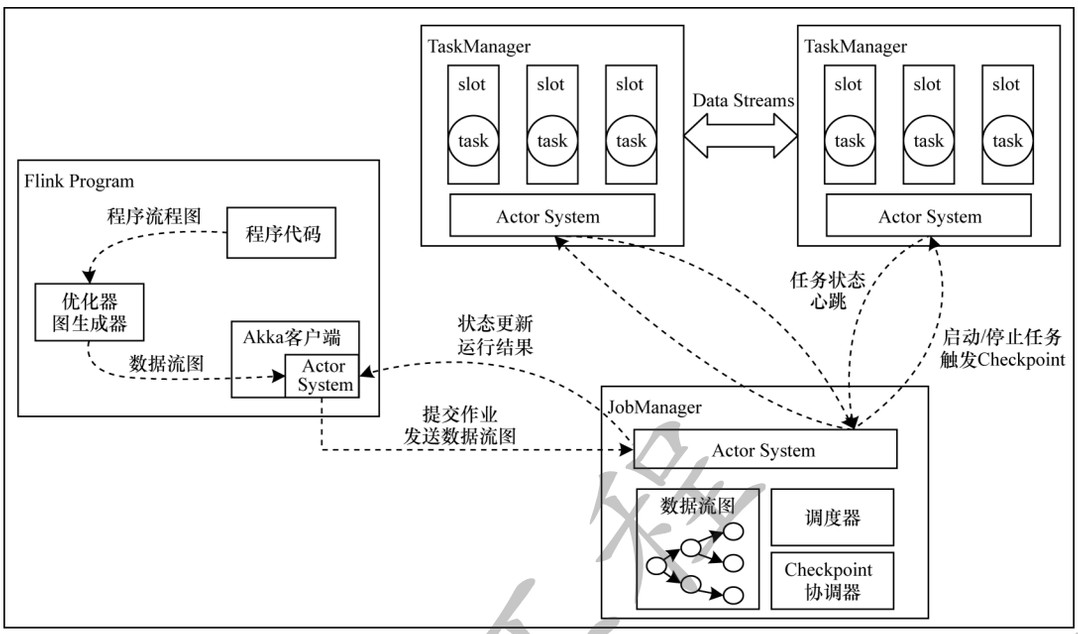
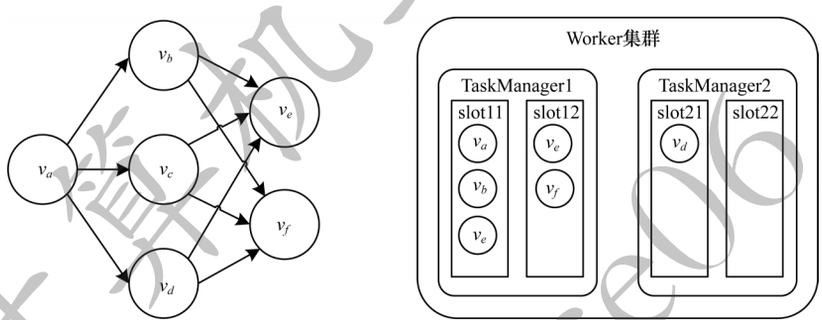


图1　Flink集群架构

Fig.1　Flink cluster architecture



(a)Flink任务拓扑模型　　(b)Flink默认任务分配模型

图2　任务拓扑分配模型

Fig.2　Task topology assignment model

2　基于节点优先级的Flink动态调度策略

2.1　节点优先级定义及调整方法描述

2.1.1　相关定义

定义1（节点优先级）　Flink集群的节点优先级集合为$P=\{P_1,P_2,\cdots\}$。P_i^x表示第i个节点的偏x相关性能优先级指数，当$x=c$时表示节点偏CPU相关性能优先级指数，当$x=m$时表示节点偏内存相关性能优先级指数。

定义2（节点性能指数计算权值）　在节点性能指数计算时，引入的各动静态性能因素权值表示为向量$\boldsymbol{M}_{\mathrm{MD}}^x=(M_{\mathrm{MD1}}^x,M_{\mathrm{MD2}}^x,M_{\mathrm{MD3}}^x)$与向量$\boldsymbol{M}_{\mathrm{MS}}^x=(M_{\mathrm{MS1}}^x,M_{\mathrm{MS2}}^x,M_{\mathrm{MS3}}^x,M_{\mathrm{MS4}}^x)$，当$x=c$时当前权值侧重计算CPU相关优先性能指数，当$x=m$时当前权值侧重计算内存相关优先性能指数。对于任意的$M_{\mathrm{MD}i}^x\in M_{\mathrm{MD}}^x$，有$\sum_{i=1}^{3}M_{\mathrm{MD}i}^x=1$；对于任意的$M_{\mathrm{MS}i}^x\in M_{\mathrm{MS}}^x$，有$\sum_{i=1}^{4}M_{\mathrm{MS}i}^x=1$。

定义3（节点资源因素指数[15]）　动态性能指数$D=\{D_1,D_2,\cdots\}$表示节点在执行任务时的实时性能变化。对于动态性能指数有3个资源约束，分别是CPU剩余率（CSR）、内存剩余率（MSR）和磁盘剩余率（DSR）。静态性能指数$S=\{S_1,S_2,\cdots\}$表示节点的不对称性能因素。对于静态性能指数有4个资源约束，分别为CPU速度（CS）、CPU核数（CQ）、内存容量（MC）和磁盘容量（DC）。

对于每个动态指数$D_i\in D$，对应资源约束向量$\boldsymbol{d}_i=(C_{\mathrm{CSR}i},M_{\mathrm{MSR}i},D_{\mathrm{DSR}i})$；对于每个静态指数$S_i\in S$，对应资源约束向量$\boldsymbol{s}_i=(C_{\mathrm{CS}i},C_{\mathrm{CQ}i},M_{\mathrm{MC}i},D_{\mathrm{DC}i})$。引入定义2中权值的计算后，得到$D_i^x$表示第$i$个节点的动态偏$x$性能指数，$S_i^x$表示第$i$个节点的静态偏$x$性能指数。

2.1.2　调整方法描述

本文所设计的节点优先级调整方法是即时反馈方法，节点优先级P_i^x计算公式如下：

$$P_i^x=\alpha\times D_i^x+\beta\times S_i^x \tag{1}$$

其中：α、β为节点整体动静态因素指数权值，$\alpha+\beta=1$；D_i^x和S_i^x表示节点动静态偏x性能指数。D_i^x和S_i^x计算公式如下：

$$D_i^x=\sum_{\varphi=1}^{e}\left[M_{\mathrm{MD}(\varphi)}^x\times d_{i(\varphi)}\right] \tag{2}$$

$$S_i^x=\sum_{\omega=1}^{f}\left[M_{\mathrm{MS}(\omega)}^x\times s_{i(\omega)}\right] \tag{3}$$

其中：$M_{\mathrm{MD}i}^x$、$M_{\mathrm{MS}i}^x$满足$\sum_{\varphi=1}^{e}M_{\mathrm{MD}(\varphi)}^x=1$与$\sum_{\omega=1}^{f}M_{\mathrm{MS}(\omega)}^x=1$，$i$表示当前计算的是第$i$个节点的因素指数；$d_i$、$s_i$分别表示第$i$个节点的各项动态和静态性能因素指数参数；$e$、$f$分别表示每个节点包含的动态与静态性能因素数量。

以4个节点组成的Flink异构集群为例，给出节点优先级调整架构如图3所示。

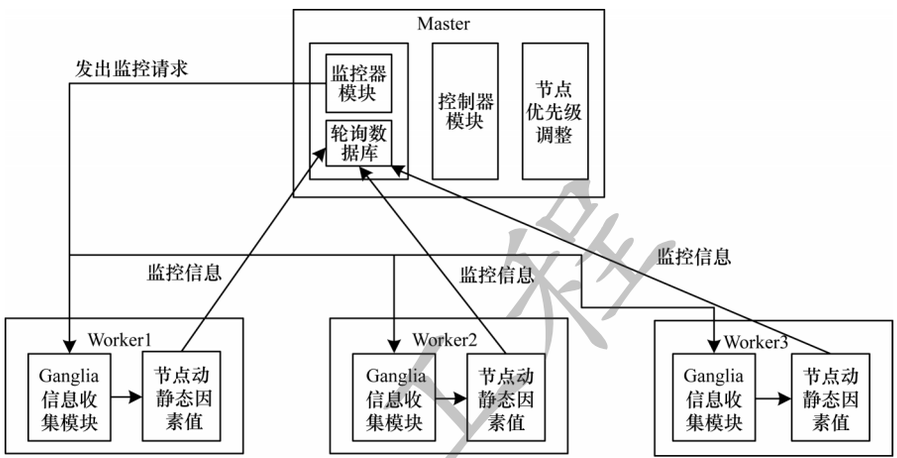


图3 节点优先级调整架构

Fig.3 Node priority adjustment architecture

监控器运行在集群Master节点中，负责周期性收集节点的资源使用情况和任务队列长度。获取的资源信息将传入到控制器中，由控制器完成具体节点优先级计算。监控器采用Ganglia[21]进行实现，Ganglia是一个可扩展分布式集群资源监控系统，能够实现对集群信息的监控。

控制器负责根据监控器监测到的信息计算出各节点的优先级。本文节点优先级调整方法引入作业环境预测机制，在一个周期内判定当前集群作业环境的参数x由式(4)确定。如果集群节点的I/O操作平均时间T(I/O)超过了整体操作时间T(all)的65%，则判定当前集群作业环境为I/O密集型，否则判定为CPU密集型。

$$x=\begin{cases}m, T(\mathrm{I/O})\geqslant T(\mathrm{all})\times 0.65\\ c, T(\mathrm{I/O})< T(\mathrm{all})\times 0.65\end{cases} \tag{4}$$

当监控器判定为I/O密集型环境时，控制器将节点动静态因素指数计算的权值重置为M_{MD}^m和M_{MS}^m，节点的偏内存性能优先级指数的计算公式如下：

$$D_i^m=M_{\mathrm{MD1}}^m\times C_{\mathrm{CSR}i}+M_{\mathrm{MD2}}^m\times M_{\mathrm{MSR}i}+M_{\mathrm{MD3}}^m\times D_{\mathrm{DSR}i} \tag{5}$$

$$S_i^m=M_{\mathrm{MS1}}^m\times C_{\mathrm{CS}i}+M_{\mathrm{MS2}}^m\times C_{\mathrm{CQ}i}+M_{\mathrm{MS3}}^m\times M_{\mathrm{MC}i}+M_{\mathrm{MD4}}^m\times D_{\mathrm{DC}i} \tag{6}$$

当监控器判定为CPU密集型环境时，控制器将节点动静态因素指数计算的权值重置为M_{MD}^c和M_{MS}^c，节点的偏CPU性能优先级指数的计算公式如下：

$$D_i^c=M_{\mathrm{MD1}}^c\times C_{\mathrm{CSR}i}+M_{\mathrm{MD2}}^c\times M_{\mathrm{MSR}i}+M_{\mathrm{MD3}}^c\times D_{\mathrm{DSR}i} \tag{7}$$

$$S_i^c=M_{\mathrm{MS1}}^c\times C_{\mathrm{CS}i}+M_{\mathrm{MS2}}^c\times C_{\mathrm{CQ}i}+M_{\mathrm{MS3}}^c\times M_{\mathrm{MC}i}+M_{\mathrm{MS4}}^c\times D_{\mathrm{DC}i} \tag{8}$$

2.2 节点优先级调整算法

算法1 节点优先级调整算法EP-NPAA

输入 节点集(NodeSets，h)，h表示节点个数

输出 集群节点优先级指数$P=\{P_1,P_2,\cdots\}$

1.if Condition is true then/*判断当前集群是否出现变化*/

2.s_i←getStatic(i)；/*获取每个节点的静态因素向量*/

3.Condition←false；

4.end if；

5.for i from 1 to h/*获取当前集群的动态因素资源信息*/

6.d_i←getDynamic(i)；/*获取每个节点的动态因素向量*/

7.if T(I/O)<T(all)×0.65 then /*根据当前作业环境计算节点优先级指数*/

8.for i from 1 to h

9.D_i←calculate.C(d_i)，S_i←calculate.C(s_i)；/*利用式(7)、式(8)计算动态因素指数*/

10.end for；

11.else if

12.for i from 1 to h

13.D_i←calculate.M(d_i)，S_i←calculate.M(s_i) /*利用式(5)、式(6)计算动态因素指数*/

14.end for；

15.end if；

16.P_i←calculate.Priority(D_i,S_i)；/*利用式(1)计算各节点优先级指数*/

17.return P；

2.3 基于节点优先级的Flink节点动态自适应调度策略

在实际运行环境中，异构分布式集群的节点往往会出现资源不平衡和负载不均衡的情况。为保证任务的高效完成，需要准确衡量各节点的性能以及

整体集群的负载程度，选择合适的节点分配任务[22]。本文提出一种基于节点优先级的Flink节点动态自适应调度算法。

2.3.1　基于节点优先级的调度方法描述

基于节点优先级的Flink节点动态自适应调度方法建立在集群异构的情况下，该方法的主要构成如下：1）Master节点监控器定期监控每个Worker节点自身资源情况及负载变化情况，并反馈控制器定期动态计算各Worker节点的优先级指数；2）在Master节点进行任务调度时，读取各节点优先级指数，选择指数较大的节点分配任务。Flink节点动态自适应调度架构如图4所示。

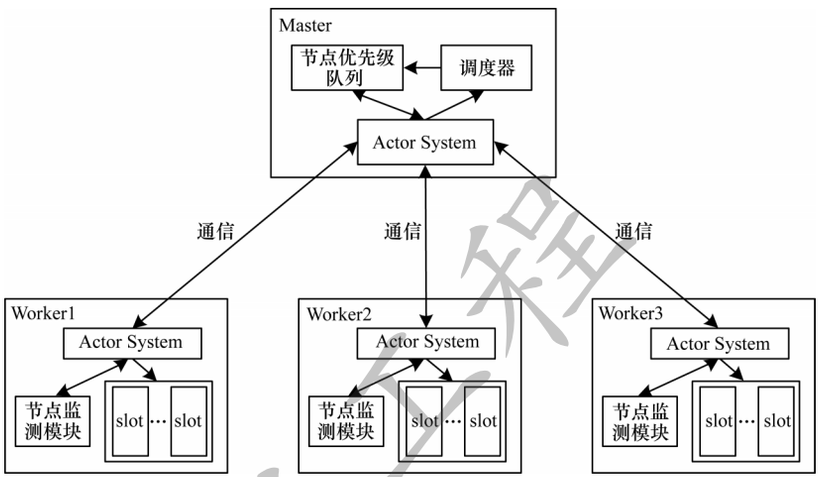


图4　Flink节点动态自适应调度架构

Fig.4　Dynamic adaptive Flink node scheduling architecture

2.3.2　Flink节点动态自适应调度算法

Flink节点动态自适应调度算法运行在Master节点上，具体步骤为：1）启动集群，Master节点检测节点是否出现变化，根据变化状态调用EP-NPAA算法重新计算节点的静态因素指数S_i；2）Master节点内控制器调用EP-NPAA算法，依次计算每个Worker节点的优先级得到$P=\{P_1,P_2,\cdots\}$，并对节点集合NodeSets进行排序；3）控制器从节点集合NodeSets中按优先级指数高低依次调用每个节点，检测当前调用节点是否有当前task可用的slot，若有可用的slot，则分配任务给该节点；若无可用slot，则继续调用下一节点。

算法2　Flink节点动态自适应调度算法F-DASA

输入　节点集（NodeSets，h），h表示节点个数，任务集（TaskSets，m），m表示任务个数

输出　第i个任务分配在第j个节点上（$node_i$，$task_j$）

1.sort P in descending order; /*根据算法1获得优先级指数P对节点进行排序*/

2.for i from 1 to h

3.for j from 1 to m

4.if j on i is true and $flag_j$ == 0 then /*判断该节点是否有可用的slot*/

5.launch j on i;

6.$flag_j \leftarrow 1$; /*flag用于标识该任务是否被分配*/

7.end if;

8.end for;

9.end for;

10.return($node_i$，$task_j$);

3　实验与结果分析

3.1　实验环境配置

为验证本文自适应调度策略对Flink异构集群的节点调度性能更佳，构建Flink异构集群数据实验平台。该平台由5台服务器组成，其中，1台为主服务器Master，4台为从服务器Worker。集群节点硬件与软件配置如表1、表2所示。

表1　集群节点硬件配置

Table 1　Hardware configuration of cluster nodes

配置项	配置详情
Master主节点	四核，4 GB内存，40 GB硬盘
Worker1节点	四核，4 GB内存，40 GB硬盘
Worker2节点	四核，2 GB内存，40 GB硬盘
Worker3节点	四核，4 GB内存，20 GB硬盘
Worker4节点	双核，4 GB内存，40 GB硬盘

表2　集群节点软件配置

Table 2　Software configuration of cluster nodes

配置项	配置详情
OS	CentOS 6.10
CPU	Intel Xeon Platinum 8269CY
Flink	Flink 1.6.0
Parallelism	4 Task Manger，8 Task slot

在实验中进行节点优先级计算时，规定动静态因素权值α和β分别取值为0.65和0.35；在内存密集型作业环境下，各动态因素权值M_{MD}^{m}取值为（0.3，0.6，0.1），各静态因素权值M_{MS}^{m}取值为（0.05，0.30，0.55，0.10）。在CPU密集型作业环境下，各动态因素权值M_{MD}^{c}取值为（0.5，0.4，0.1），各静态因素权值M_{MS}^{c}

取值为(0.10,0.50,0.35,0.05)。

3.2 实验数据设置

为有效验证F-DASA算法对Flink异构集群的影响,实验将本文策略与Flink默认调度策略以及TSS-Flink策略[17]进行大数据基准测试WorldCount和TeraSort对比。WorldCount是词频统计基准测试,特点是CPU资源占用率较高、内存占用率较低,数据集采用9个表生成模拟5种事务处理的TPC-C数据集[23]。TeraSort是分布式排序基准测试,在执行过程中会大量占用内存资源,数据集为待排序的数值型数据集。

3.3 结果分析

3.3.1 运行时间对比

实验通过对比使用Flink默认调度策略、TSS-Flink策略和本文策略后的作业运行时间,验证F-DASA算法对Flink异构集群的影响。在实验过程中,基准测试WorldCount的数据集规模分别为2 GB、4 GB和6 GB,且设置不同的作业并行度,运行时间如图5所示。

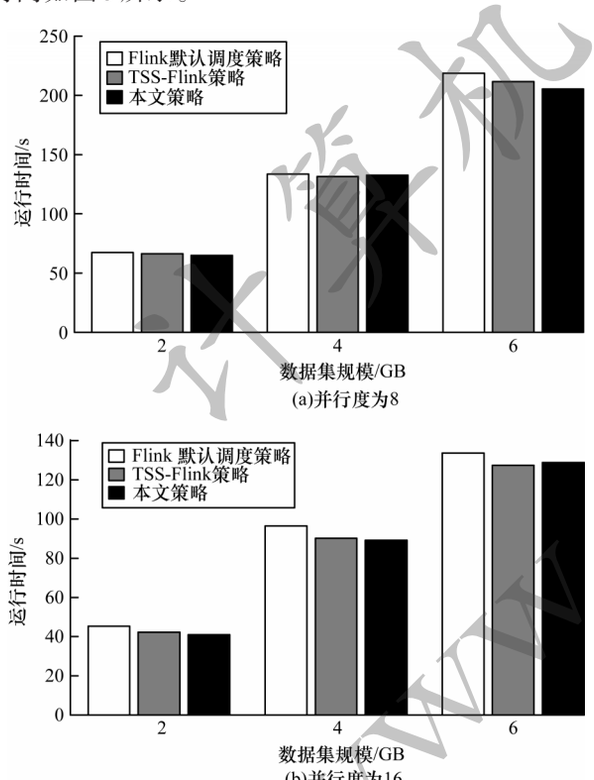


图5 作业运行时间对比

Fig.5 Comparison of job runtime

从图5的实验结果可以看出,在并行度为8和16的情况下,使用TSS-Flink策略和本文策略后,基准测试WorldCount的运行时间都有所减少。在使用本文策略之后,比使用Flink默认调度策略约平均减少了6%。这是因为在默认调度策略下,调度器使异构集群中资源较少的节点完成和其他节点同等的任务,拖慢了整体作业运行时间。在本文策略下,调度器使任务得到均匀分配,资源多的节点可以完成尽可能多的任务,缩短了整体运行时间。相较于TSS-Flink策略,本文策略的运行时间更少,这是因为TSS-Flink策略缺少对于异构环境下节点资源不均衡问题的考虑,从而导致性能有所差异。

3.3.2 系统延迟对比

实验通过对比Flink默认调度策略与本文策略之间的延迟关系,验证F-DASA算法对Flink异构集群的影响。在实验过程中,在设置作业并行度为8的情况下,对比基准测试WorldCount和TeraSort下不同数据吞吐量的系统延迟,实验结果如图6所示。

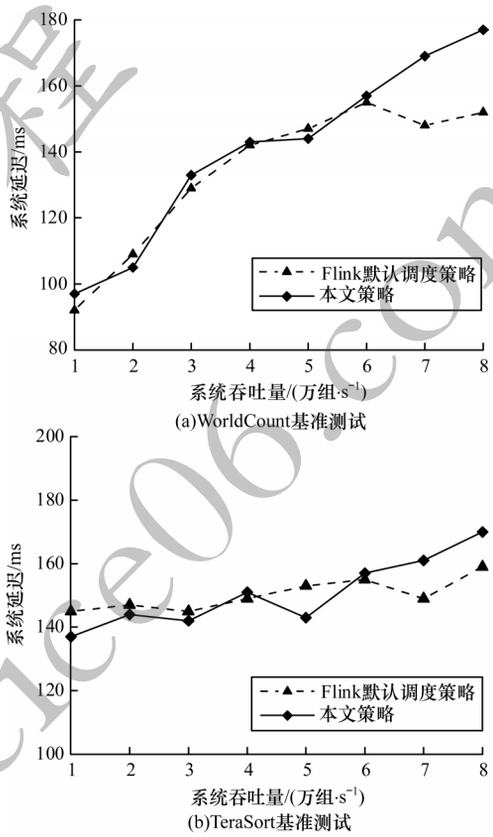


图6 2种策略的系统延迟对比

Fig.6 Comparison of system delay of two strategies

由图6(a)可以看出,基准测试WorldCount在使用默认调度策略时,随着吞吐量的增加,系统延迟在100 ms至150 ms内缓慢上升。使用本文策略后,在吞吐量为1~6万组/s时,系统延迟在100 ms至150 ms内缓慢上升,与Flink默认调度策略的系统延迟相近;在吞吐量达到6万组/s以上时,系统延迟上升幅度略微增大,属于可接受范围。由图6(b)可以看出,基准测试TeraSort在使用默认调度策略时,随着吞吐量的增加,系统延迟始终在150 ms上下浮动。在使用本文策略后,系统延迟在吞吐量为1~6万组/s时,在145 ms上下浮动,略微优于Flink默认调度策略;在吞吐量达到6万组/s以上时,系统延迟上升幅度略微增大,属于可接受范围。综上所述,在异构Flink集群中使用本文策略后,依然能够保持较为稳定的延迟率,达到集群原有的响应速度。

4　结束语

针对异构Flink集群默认策略在节点调度过程中存在部分节点负载不均衡的问题，本文提出一种基于节点优先级的Flink节点动态自适应调度策略。该策略能够监控集群中任务与节点的各项数据，并在任务执行过程中根据实时的作业环境更新各个节点的优先级指数，为系统任务找到最佳的执行节点。实验结果表明，该策略可在保持集群低延迟的基础上，提高异构Flink集群对于节点资源的利用率。下一步将针对节点的CPU、内存和带宽性能设置合理的阈值，确保集群不会出现满负载状态，同时设计集群任务选择算法，并将其与F-DASA算法相结合进一步提升异构Flink集群整体性能。

参考文献

[1] 孙大为，张广艳，郑纬民. 大数据流式计算：关键技术及系统实例[J]. 软件学报，2014，25(4)：839-862.
SUN D W，ZHANG G Y，ZHENG W M. Big data flow computing：key technologies and system examples[J]. Journal of Software，2014，25(4)：839-862. (in Chinese)

[2] SHAHVERDI E，AWAD A，SAKR S. Big stream processing systems：an experimental evaluation[C]// Proceedings of the 35th International Conference on Data Engineering Workshops. Washington D. C.，USA：IEEE Press，2019：53-60.

[3] QIN X，JIANG H. A dynamic and reliability-driven scheduling algorithm for parallel real-time jobs executing on heterogeneous clusters[J]. Journal of Parallel and Distributed Computing，2005，65(8)：885-900.

[4] ZHU X M，HE C，LI K L，et al. Adaptive energy-efficient scheduling for real-time tasks on DVS-enabled heterogeneous clusters[J]. Journal of Parallel and Distributed Computing，2012，72(6)：751-763.

[5] WU Y，WANG Z R，RUAN Q，et al. Node scheduling：a blockchain-based node selection approach on sapiens chain[C]//Proceedings of 2019 IEEE International Conference on Big Data and Smart Computing. Washington D. C.，USA：IEEE Press，2019：1-7.

[6] ALI-ELDIN A，TORDSSON J，ELMROTH E. An adaptive hybrid elasticity controller for cloud infrastructures[C]// Proceedings of 2012 IEEE Network Operations and Management Symposium. Washington D. C.，USA：IEEE Press，2012：204-212.

[7] TRAN G P C，CHEN Y，KANG D I，et al. Automated demand-based vertical elasticity for heterogeneous real-time workloads[C]//Proceedings of the 9th International Conference on Cloud Computing. Washington D. C.，USA：IEEE Press，2016：831-834.

[8] JAYAKUMAR V K，LEE J，KIM I K，et al. A self-optimized generic workload prediction framework for cloud computing[C]//Proceedings of 2020 IEEE International Parallel and Distributed Processing Symposium. Washington D. C.，USA：IEEE Press，2020：779-788.

[9] MICHELLE Y，GAREGRAT N，MOHAN S. Towards a resource aware scheduler in Hadoop[C]//Proceedings of the 7th IEEE International Conference on Web Services. Washington D. C.，USA：IEEE Press，2009：102-109.

[10] CARUANA G，LI M Z，QI M，et al. gSched：a resource aware Hadoop scheduler for heterogeneous cloud computing environments[J]. Concurrency and Computation：Practice & Experience，2017，29(20)：1-7.

[11] NAIK N S，NEGI A，BAPU T B R，et al. A data locality based scheduler to enhance MapReduce performance in heterogeneous environments[J]. Future Generation Computer Systems，2019，90：423-434.

[12] XU X L，CAO L L，WANG X H. Adaptive task scheduling strategy based on dynamic workload adjustment for heterogeneous Hadoop clusters[J]. IEEE Systems Journal，2017，10(2)：471-482.

[13] 廉华，刘瑜. 基于YARN资源调度器的MapReduce作业数调节方法[J]. 计算机系统应用，2020，29(3)：218-222.
LIAN H，LIU Y. Number adjustment method of MapReduce jobs based on YARN resource scheduler[J]. Computer Systems & Applications，2020，29(3)：218-222. (in Chinese)

[14] 杨志伟，郑烇，王嵩，等. 异构Spark集群下自适应任务调度策略[J]. 计算机工程，2016，42(1)：31-35，40.
YANG Z W，ZHENG Q，WANG S，et al. Adaptive task scheduling strategy for heterogeneous spark cluster[J]. Computer Engineering，2016，42(1)：31-35，40. (in Chinese)

[15] 胡亚红，盛夏，毛家发. 资源不均衡Spark环境任务调度优化算法研究[J]. 计算机工程与科学，2020，42(2)：203-209.
HU Y H，SHENG X，MAO J F. Task scheduling optimization in Spark environment with unbalanced resources[J]. Computer Engineering & Science，2020，42(2)：203-209. (in Chinese)

[16] 郝志峰，黄泽林，蔡瑞初，等. 基于YARN的分布式资源动态调度和协同分配系统[J]. 计算机工程，2021，47(2)：226-232.
HAO Z F，HUANG Z L，CAI R C，et al. Dynamic resource scheduling and collaborative allocation system based on YARN[J]. Computer Engineering，2021，47(2)：226-232. (in Chinese)

[17] 何贞贞，于炯，李梓杨，等. 基于Flink的任务调度策略[J]. 计算机工程与设计，2020，41(5)：1280-1287.
HE Z Z，YU J，LI Z Y，et al. Task scheduling strategy based on Flink environment[J]. Computer Engineering and Design，2020，41(5)：1280-1287. (in Chinese)

[18] 庆骁. 面向FLINK流处理框架的容错策略优化研究[D]. 哈尔滨：哈尔滨工业大学，2019.
QING X. Research on fault-tolerant strategy optimization for FLINK stream processing framework[D]. Harbin：Harbin Institute of Technology，2019. (in Chinese)

[19] 赵娟，程国钟. 基于Hadoop、Storm、Samza、Spark及Flink大数据处理框架的比较研究[J]. 信息系统工程，2017(6)：117，119.
ZHAO J，CHEN G Z. Comparison of big data processing frameworks based on Hadoop，Storm，Samza，Spark and Flink[J]. China CIO News，2017(6)：117，119. (in Chinese)

[20] TOLIOPOULOS T，GOUNARIS A. Adaptive distributed partitioning in Apache Flink[C]//Proceedings of the 36th International Conference on Data Engineering Workshops. Washington D. C.，USA：IEEE Press，2020：127-132.

[21] Ganglia[EB/OL]. [2020-10-19]. http://ganglia. info/.

[22] AJILA T，MAJUMDAR S. Data driven priority scheduling on a spark streaming system[C]//Proceedings of the 19th IEEE/ACM International Symposium on Cluster，Cloud and Grid Computing. Washington D. C.，USA：IEEE Press，2019：561-568.

[23] KAMSKY A. Adapting TPC-C benchmark to measure performance of multi-document transactions in MongoDB[J]. Proceedings of the VLDB Endowment，2019，12(12)：2254-2262.

编辑　陆燕菲