CN 43-1258/TP
ISSN 1007-130X

计算机工程与科学
Computer Engineering & Science

第 44 卷第 5 期 2022 年 5 月
Vol. 44, No. 5, May. 2022

文章编号:1007-130X(2022)05-0779-09

面向云环境的 Flink 负载均衡策略*

徐浩桐[1,2,3],黄　山[1,2,3],孙国璋[1,2,3],贺菲莉[1,2,3],段晓东[1,2,3]

(1.大连民族大学计算机科学与工程学院,辽宁 大连 116600;
2.大数据应用技术国家民委重点实验室(大连民族大学),辽宁 大连 116600;
3.大连市民族文化数字技术重点实验室(大连民族大学),辽宁 大连 116600)

摘　要:作为新一代的大数据计算引擎,Flink 得到了广泛应用。Flink 在云环境下进行容器化部署时,其默认任务调度算法不能感知节点的资源信息,导致即时调整负载和自主均衡能力较差,而主流的容器编排工具虽然提供了管理容器的可能性,却也未能结合 Flink 特点解决平衡资源利用的同时降低容器组内的通信开销问题。针对以上问题开展研究,提出了一种面向云环境的 Flink 负载均衡策略 FLBS,综合考虑了 Flink 集群中算子的分布特点和容器间通信机制,以节点间通信开销和均衡负载作为评估标准。实验结果表明,与 Flink 默认调度策略相比,FLBS 能够有效提高计算效率,提升系统性能。

关键词:Flink;容器;通信开销;负载均衡;迁移

中图分类号:TP391　　**文献标志码**:A

doi:10.3969/j.issn.1007-130X.2022.05.003

A Flink load balancing strategy for cloud environment

XU Hao-tong[1,2,3], HUANG Shan[1,2,3], SUN Guo-zhang[1,2,3], HE Fei-li[1,2,3], DUAN Xiao-dong[1,2,3]

(1. College of Computer Science and Engineering, Dalian Minzu University, Dalian 116600;
2. State Ethnic Affairs Commission Key Laboratory of Big Data Applied Technology (Dalian Minzu University), Dalian 116600;
3. Dalian Key Laboratory of Digital Technology for National Culture (Dalian Minzu University), Dalian 116600, China)

Abstract: As a new generation of big data computing engine, Flink has been widely used. When containers of Flink are deployed in cloud environment, its default task scheduling algorithm cannot perceive node resources information and adjust the load in time, and the capacity for independent equilibrium is poorer. Although mainstream container layout tools provide the possibility of container management, they fails to combine Flink characteristics to solve the problem of balancing the resource utilization while reducing the communication overhead in the container group. Aiming at the above problem, this paper proposes a Flink load balancing strategy for cloud environment, which comprehensively considers the distribution characteristics of operators in Flink cluster and the communication mechanism between containers, and takes the communication cost between nodes and load balancing as evaluation criteria. Experimental results show that, compared with Flink default scheduling algorithm, this algorithm can effectively improve the computing efficiency and system performance.

Key words: Flink; container; communication overhead; load balancing; migration

* **收稿日期**:2021-11-09;**修回日期**:2022-01-13
基金项目:国家重点研发计划(2018YFB1004402)
通信作者:黄山(huangshan@dlnu.edu.cn)
通信地址:116600 辽宁省大连市大连民族大学计算机科学与工程学院
Address: College of Computer Science and Engineering, Dalian Minzu University, Dalian 116600, Liaoning, P. R. China

(C)1994-2022 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

1 引言

人类正从IT时代走向DT(Data Technology)时代,数据是新时代最有价值的资源。DT时代的大数据分析系统以云计算为平台,以数据为中心来组织存储模型、计算模型和应用[1]。Flink[2]作为新一代的大数据计算引擎,能够以数据并行和流水线的方式执行包括批处理和流处理在内的任意流数据程序,备受学术界与工业界青睐。同时,容器技术因其卓越的再生性、一致性、可追溯性与可移植性使得应用程序容器化已变为一种发展趋势。因此,在容器技术盛行的今天,大数据技术与容器技术相结合必然会成为一个热点研究问题。容器化Flink部署能够消除线上线下的环境差异,提高资源利用率,保证生命周期内的环境一致性与标准化,为开发者提供巨大便利,提升开发效率。

然而就传统的容器化Flink运行部署模式来看,资源利用不均衡和通信开销过大等问题依然存在。针对此问题,本文提出了一种面向云环境的Flink负载均衡策略FLBS(Flink Load Balancing Strategy)。FLBS通过计算节点间的容器开销差异和资源利用率,从通信代价和均衡负载2个方面,对Flink集群中的容器进行动态调整,实时迁移。在不同规模的不同Benchmark作业上的实验表明,与Flink默认调度策略相比,该策略在资源利用和计算时延方面均有优化。本文的主要工作如下所示:

(1)提出了一种面向云环境的Flink负载均衡策略FLBS:针对容器化Flink集群中负载较大的节点,对容器进行迁移,在减小节点间负载差异的同时,最小化容器间的跨节点通信开销,提升系统的计算效率。

(2)提出了Flink容器通信代价模型:通过计算同节点和跨节点的数据流大小,对待迁容器的目标节点进行评分,降低迁移后的跨节点通信开销。

(3)提出了负载均衡模型:迁移容器时,从资源利用率和资源均衡程度2个方面对计算节点进行考量,以均衡负载。

2 相关工作

现阶段国内外对于容器化大数据处理引擎任务调度优化的研究主要集中在以下2个方面:一是针对容器迁移策略的研究,二是优化Flink等大数据计算引擎的任务调度策略的研究。

在容器迁移策略方面,有以下相关工作。文献[3]针对容器放置和重分配问题,提出了一种有效的通信感知最差拟合递减算法来将一组新的容器放置到数据中心,并通过在服务器之间迁移容器来优化容器的初始分配。文献[4]通过引入伪随机比规则,同时结合局部信息素蒸发和全局信息素更新,确定容器的迁移优先级,设计了一种基于改进蚁群系统MACS(Modified Ant Colony System)的迁移算法,利用负载均衡联合迁移成本LBJC(Load Balancing Joint Migration Cost)模型来进行容器迁移。文献[5]通过CRIU(Checkpoint/Restore In Userspace)[6]技术给出了一种支持Linux容器迁移的驱动程序的原型实现。文献[7]提出了一种考虑容器间距离、成本和可用带宽的局部动态迁移模型,对云环境下的容器进行热迁移。文献[8]针对边缘计算中基于容器的服务迁移问题,提出了一种基于移动感知的服务迁移机制,根据迁移成本和服务所在设备的移动方向,选择相应的目标节点进行迁移。文献[9]针对容器迁移过程中的资源开销和延迟会降低高性能计算机计算效率的问题,提出了一种多容器迁移策略,并设计了相应的迁移工具。文献[10]通过分析传统虚拟机和Docker容器的差异性,提出了一种面向Docker[11]容器的热迁移机制,实现了容器的运行状态迁移。

在优化Flink等大数据计算平台的任务调度策略方面,有以下相关工作。文献[12]提出了平滑加权轮询任务调度算法和基于蚁群算法的任务调度算法,解决了Flink集群运行过程中负载不均衡问题。文献[13]针对流处理系统Flink任务调度策略忽略了集群异构和节点可用资源,导致集群整体负载不均的问题,提出了一种基于异构Flink集群的节点优先级体系,动态调整适应当前作业环境的节点优先级指数,并按照节点优先级体系完成了任务的分配。文献[14]针对目前Spark大数据平台在任务调度时未考虑集群的异构性和节点资源利用的情况,提出了一种能够根据任务执行过程中的节点状态动态调整各个节点优先级的节点优先级调整算法。文献[15]针对大数据流式计算平台拓扑中因各关键节点上任务间不同类型的通信方式导致的通信开销较大问题,提出了一种Flink环境下的任务调度策略。通过各任务间数据流大小确定拓扑边权重,将有向无环图转化为拓扑关键路径模型,在保证关键路径上节点负载差异较小的同

(C)1994-2022 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

时,最小化关键任务的节点间通信开销。文献[16]通过建立负载预测模型,预测集群负载的变化趋势,提出了一种 Flink 环境下基于负载预测的弹性资源调度 LPERS-Flink(Load Prediction based Elastic Resource Scheduling strategy in Flink)策略。文献[17]建立了流网络模型并通过构建算法计算每条边的容量值;其次通过弹性资源调度算法确定集群性能瓶颈并制定动态资源调度计划;最后通过基于数据分簇和分桶管理的状态数据迁移算法,实施调度计划并完成节点间的数据迁移。

虽然上述研究都取得了不错的成果,但目前在容器化 Flink 集群上的调度优化工作仍有欠缺。现有工作一部分无法适应 Flink 资源调度,一部分没有结合容器化 Flink 特点考虑容器间通信开销对节点负载均衡的影响,有些针对 Flink 任务调度的优化工作只停留在任务分配阶段,而不能在运行时动态调整,不够灵活。针对以上问题,本文提出了一种面向云环境的 Flink 负载均衡策略 FLBS,既考虑了容器化 Flink 特点,又能够在运行时即时调整负载,有效提升了系统的计算效率。

3 研究技术背景

本节主要介绍相关的技术背景,主要包括 Flink 计算框架、容器技术和 CRIU 迁移技术。

3.1 Flink 计算框架—Apache Flink

Apache Flink 是由 Apache 软件基金会开发的开源流处理框架,其核心是用 Java 和 Scala 编写的分布式流数据处理引擎。Flink 以数据并行和流水线的方式执行包括批处理和流处理在内的任意流数据程序。此外,其运行时本身也支持迭代算法的执行。Flink 作为最新一代的大数据计算引擎,除具有低延迟、高吞吐量和高性能的优势外,还支持事件时间(Event Time)概念,支持有状态计算,支持高度灵活的窗口(Window)操作、基于轻量级分布式快照(CheckPoint)实现的容错和基于 JVM 实现的独立内存管理,支持保存点(Save Point)机制。因此,Flink 以其优越的性能备受学术界与工业界青睐。

3.2 容器技术

容器化因其在部署应用程序和服务时的便利性和良好性能而备受青睐。首先,容器通过名称空间技术提供了良好的隔离,消除了与其他容器的冲突。其次,容器将代码、运行时状态信息、系统工具和系统库等都放在一个包中,并且不需要任何外部依赖来运行进程[18],这使得容器具有高度的可移植性和分发速度。

3.3 CRIU 技术

CRIU 最早是由 Pavel Emelyanov 发布到 Linux 开发者社区的,依赖于从 3.11 版开始逐渐提供的 Linux 内核特性,该工具主要是在用户空间实现的,可以对一个正在运行的应用或该应用的一部分进行状态保存并设置检查点,将进程信息转存为一组文件,并通过这些文件从快照位置恢复应用运行,以实现容器的热迁移、快照和远程调试等其他功能。

CRIU 技术能够实现基于容器的迁移,因此本文使用该技术对 Flink 容器进行动态迁移,以提升系统计算效率,降低通信开销。

4 问题分析与系统架构

本节将从容器的编排工具和 Flink 容器化部署时默认的任务调度策略 2 个方面进行问题分析,并阐述整个均衡策略的系统流程。

为了确保服务的完整性,特定应用程序的一个函数可以实例化多个容器。例如,在 Flink 中,每个 TaskManager 或 JobManager 应该被部署为一个容器,这些容器部署在云或数据中心中,协同完成计算任务,并由 Kubernetes[19] 和 Mesos[20] 等编排工具管理。使用名称服务,这些编排工具可以快速定位不同服务器上的容器,因此可以很好地完成应用程序升级和故障恢复。由于容器易于构建、替换和删除,这样的体系结构使得维护基于多容器的应用程序变得很方便。

但是,基于多容器的体系结构也带来了一些副作用,即通信效率低。由于部署在同一组容器中的功能属于同一个服务,它们之间需要交换控制消息和传输数据。因此,同一应用程序的容器间的通信效率对整体任务性能影响很大。然而,简单地整合策略可能会导致多个资源的不平衡利用,因为同一组的容器通常集中于同一资源。上述容器编排工具虽然提供了利用容器的可能性,但是如何管理容器组以减少通信开销和平衡资源利用仍然是一个悬而未决的问题。

如图 1 所示,在 Flink 集群中,一个任务中不同容器中各算子之间需要频繁地通信,以协作完成计算任务,而各容器分布在 Flink 集群的不同物理机节点上,故容器间通信可分为同节点通信和跨节

(C)1994-2022 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

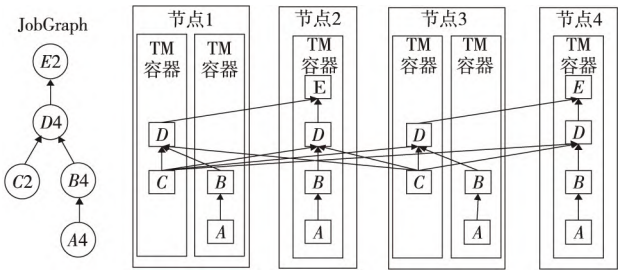

Figure 1 Flink default task allocation model

图 1 Flink 默认任务分配模型

点通信。由容器间的通信机制可知,跨节点的通信代价远高于同节点的通信代价,而 Flink 默认采用轮询的调度策略,在不同节点的各容器间随机分配算子,未考虑任务中算子分布特点和容器间通信开销,同时 Flink 默认的任务调度策略也不具备均衡负载的能力,即不能即时获取各节点的资源信息以进行实时调度,均衡负载。

综上,若能够在保证各节点负载均衡的同时尽可能地将容器间的跨节点通信转变为同节点通信,降低通信开销,便可以有效地提升系统计算效率。故本文提出了一种面向云环境的 Flink 负载均衡策略 FLBS,通过迁移 Flink 容器进行均衡负载的同时,结合 Flink 的算子分布特点,考虑了不同容器间的通信情况,从负载均衡和通信开销 2 方面对容器化 Flink 集群进行均衡优化。

如图 2 所示,本文提出的 FLBS 主要分为 3 个部分:(1)负载均衡探测对容器化 Flink 集群中的节点进行负载探测,并决定是否进行迁移;(2)基于负载均衡和通信代价的容器迁移模型;(3)面向 Flink 容器的动态迁移策略,包括迁移容器的选择和迁移目标节点的选择。下面将从以上 3 个方面介绍各部分的具体流程。

5 Flink 负载均衡策略

在 Flink 容器通信代价模型和节点负载均衡模型的基础上,FLBS 中的算法通过将部分容器从过载的节点迁移到跨节点通信开销较低的相对空闲的节点上以进行均衡负载。下面将详细介绍 FLBS 策略的执行流程与具体实现。

5.1 负载均衡探测

为了解决容器化 Flink 集群中节点负载不均的问题,首先需要获取集群中各节点的性能信息,并对整个集群进行负载均衡探测,确定超载节点,触发迁移算法以均衡负载。本文在进行容器迁移时所采用的负载均衡探测步骤如算法 1 所示。

算法 1 负载均衡探测

输入:当前周期 t 的资源利用率 r_t,所有节点集合 N。

输出:超载节点 n_{hot}。

1:N←set of all nodes;//Flink 集群中计算节点集合

2: N_{hot} ← set of hot nodes;//超载节点集合

3:**While** $node \in N \neq \varnothing$ **do**

4: **If** $r_t > k$ **then** //判断资源利用率是否超过阈值

5: Add $node$ to N_{hot};//将当前节点添加到超载集合

6: **end If**

7:**end While**

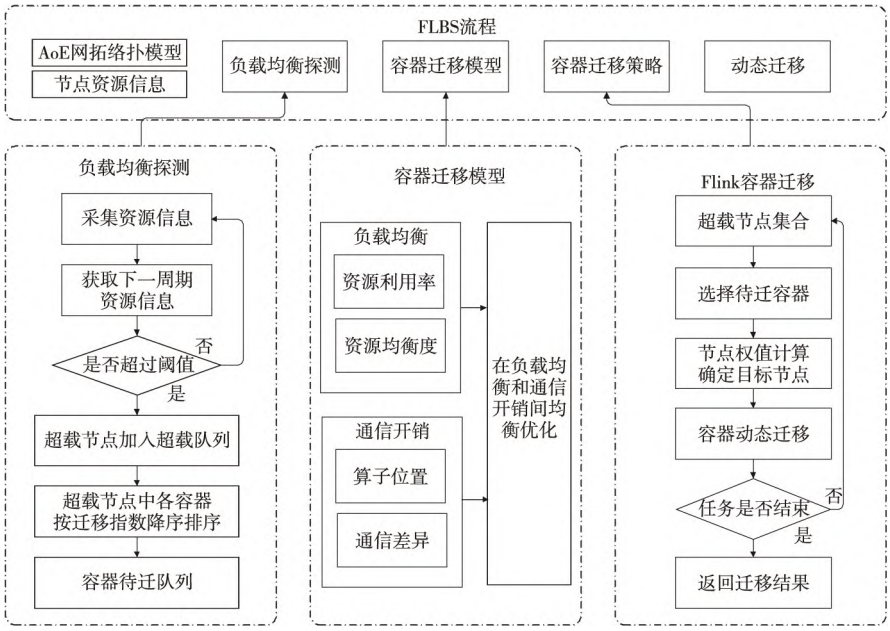

Figure 2 Structure of FLBS

图 2 FLBS 结构

(C)1994-2022 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

8:Sort N_{hot} by R_U; //将超载节点按负载排序

9:Pick the $n_{hot} \in N_{hot}$ with the largest R_U value;/*挑选负载最大的节点*/

10:**Return** n_{hot}

首先,周期性采集 Flink 集群中每个节点的资源利用率,监控每个节点的资源利用情况,判断集群中是否存在超载节点,当超载节点出现时,将其加入超载集合。其中,当前周期 t 的节点资源利用率 r_t 都是通过使用自回归模型(Autoregressive Model)由最近 n 个周期的资源使用情况预测得到的,如式(1)所示:

$$r_t = \varphi_0 + \varphi_1 r_{t-1} + \varphi_2 r_{t-2} + \cdots + \varphi_n r_{t-n} + \omega \tag{1}$$

其中,$r_{t-n}, r_{t-n-1}, \cdots, r_{t-1}$ 表示节点中某一资源在 n 个周期内的利用率,φ_0 为常数项,$\varphi_1, \cdots, \varphi_n$ 为自相关系数,ω 是均值为 0、方差为 σ 的随机误差值。若节点在当前周期的资源利用率超过了给定的阈值,则该节点被标记为超载节点。

下一步计算针对不同资源利用的综合指数。一个 Flink 任务通常需要多种不同类型的资源来提供服务,如 CPU、内存和网络带宽等,在负载均衡探测过程中,不能以单一性能信息作为衡量标准,故本文定义了一个资源综合利用指数来统计各节点中不同资源的综合利用率,如式(2)所示:

$$R_U = \frac{1}{1-mem} * \frac{1}{1-cpu} * \frac{1}{1-net} \tag{2}$$

其中,mem 表示节点中的内存利用率,cpu 表示 CPU 利用率,net 表示节点网络带宽的使用率。R_U 值为 Flink 集群中各节点的资源利用率的综合指数,R_U 值越大,节点中的资源利用率越高,负载越大,将计算节点按 R_U 值从大到小进行降序排列,则排列越靠前的,执行迁移程序的优先级越高。

5.2 模型构建

本节主要结合通信代价与均衡负载对问题进行定义和建模。

5.2.1 AoE 网络拓扑模型

为确定 Flink 集群中各算子间数据流大小,量化容器间跨节点通信和同节点通信的开销差异,根据 Flink 的任务拓扑模型,定义有向无环图 $G=(V(G),E(G))$,它由顶点和作业边组成,其中,$V(G)$ 表示拓扑中的算子集合,$E(G)=\{e\langle v_s,v_t\rangle \mid v_s,v_t \in V(G)\}$ 表示算子间的数据流集合。由流式计算的任务拓扑结构可知,若将顶点 v_s 流向顶点 v_t 的数据流大小作为弧 $v_s \rightarrow v_t$ 的权重,那么可以将流式计算的拓扑图转为带权值的 AoE 网络。AoE 网络拓扑模型如图 3 所示。

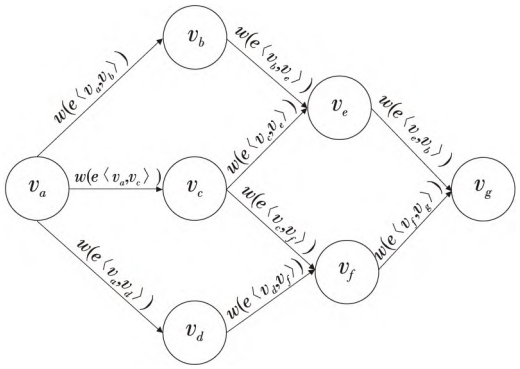

Figure 3 Topology model of AoE network

图 3 AoE 网络拓扑模型

5.2.2 Flink 容器通信代价模型

如第 3 节所述,为最小化容器间跨节点通信开销,提升计算效率,需要分别计算当前容器的跨节点通信开销和同节点通信开销,通过 AoE 网络拓扑模型,便可以量化计算各节点与当前容器的开销差异。现将 2 个容器间的通信代价定义如式(3)所示:

$$f(H(c_i),H(c_j)) = \sum_{v_s,v_t \in V(c_i),V(c_j)} w(e\langle v_s,v_t\rangle) \tag{3}$$

其中,C 代表 Flink 集群中所有 Flink 容器集合,对于每一个容器 $c \in C$,$H(c)$ 表示容器 c 部署在节点 H 上,若 $H(c_i)=H(c_j)$,则代表容器 c_i 和容器 c_j 部署在同一个节点上;$f(H(c_i),H(c_j))$ 表示在相同或不同节点上容器 c_i 和容器 c_j 之间的通信代价,它是由互相通信的 2 容器间各算子的数据流大小的累加得到;$V(c_i)$ 表示容器 c_i 上所有算子集合。$w(\cdot)$ 为 2 个算子间的数据流大小。

同时,在 Flink 集群中,一个节点可能部署有多个 Flink 容器,每个容器中均可能有上下游算子,故在计算一个容器的跨节点通信代价和同节点通信代价时,通常是一个容器对多个容器的,故定义一个容器与多个容器间的通信代价如式(4)所示:

$$\sum_{\forall c_i,c_j \in C,c_i,\neq c_j} f(H(c_i),H(c_j)) \tag{4}$$

最终待迁容器与待选节点的通信指数如式(5)所示:

$$C_{score} = \sum_{\forall c_i,c_j \in C,H(c_i)\neq H(c_j)} f(H(c_i),H(c_j)) - \sum_{\forall c_i,c_j \in C,H(c_i)=H(c_j),c_i,\neq c_j} f(H(c_i),H(c_j)) \tag{5}$$

其中,C_{score} 表示待选节点的通信评分,它是由待迁容器的跨节点通信代价与同节点通信代价相减得

(C)1994-2022 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

到的。若 $C_{score}>0$,则表示跨节点通信代价大于同节点的通信代价,由 Flink 拓扑模型可知,当提交拓扑给节点后,拓扑实例便不会发生改变,其包含的任务总数和数据流总量也不会改变[15],故迁移后能够有效降低跨节点通信代价,缩短通信时间,增加通信效率,且增加的通信效率将大于迁移后因同节点通信变为跨节点通信而降低的通信效率;其差值越大,则迁移后提高的通信效率越多,因迁移而降低的通信效率越少,待选节点的通信评分越高。

5.2.3 节点负载均衡模型

在迁移过程中对目标节点进行选择不仅要考虑容器间的通信代价,负载均衡也是一个重要指标,故本文提出资源需求评分 R_{score} 和资源均衡评分 B_{score} 来衡量节点的负载均衡程度,以实现迁移容器时目标节点的优选。如式(6)所示,资源需求评分 R_{score} 是由节点空闲资源与节点资源总容量的比值计算而来,即由 CPU 或内存资源的总容量减去节点上已有容器和当前要迁移容器的需求总量,得到的结果再除以总容量。其中,$capacity$ 为资源总容量,$requested$ 为容器资源需求量,CPU 和内存具有相同的权重,资源空闲比例越高的节点得分就越高。

$$R_{score}=\mathrm{CPU}\left(\frac{capacity-sum(requested)}{capacity}\right)+\mathrm{MEM}\left(\frac{capacity-sum(requested)}{capacity}\right) \tag{6}$$

而资源均衡评分 B_{score} 是以 CPU 利用率 cpu 和内存资源利用率 mem 的相近程度作为评估标准,如式(7)所示,二者越接近的节点权重越高。资源均衡评分与资源需求评分相结合用于平衡优化节点资源的使用情况,以选择那些在迁移当前容器后系统资源更为均衡的节点。

$$B_{score}=\frac{1}{|cpu-mem|} \tag{7}$$

综上所述,本文共提出了 2 个模型:Flink 容器通信代价模型和节点负载均衡模型,3 个指标:待选节点的通信指数、资源需求评分和资源均衡评分。由上述可知,使用任意单一指标均不能很好地完成容器化 Flink 集群的负载均衡优化,对多个优化目标进行优化的常用方法是将多个目标转换为单个目标。本文就采用这种方法,将最终待选节点评分定义为所有上述定义的评分的加权和,如式(8)所示,从通信代价、资源利用和资源均衡 3 个方面对待选节点进行评估,以选出最优目标节点进行迁移,提升系统计算效率。

$$finalScoreNode=\omega_C*C_{score}+\omega_R*R_{score}+\omega_B*B_{score} \tag{8}$$

其中,ω_C 为通信评分权重,ω_R 为资源需求评分权重,ω_B 为资源均衡评分权重,$\omega_C+\omega_R+\omega_B=1$。

5.3 Flink 容器迁移

容器迁移要解决的主要问题是被迁移容器的选择和容器的重映射。其中,选择待迁容器时主要考虑 2 个方面:一是所在节点的负载情况;二是容器迁移时的开销大小,在进行迁移时,负载越大、内存占用越小的容器迁移优先级应越高。在进行容器重映射时,根据 5.2 节中提出的评分模型,通过评分对集群中的待选节点进行择优,将由负载均衡探测算法选出的待迁容器迁往负载较为均衡的、跨节点通信代价相对较低的目标节点。Flink 容器重映射的具体过程如算法 2 所示。

算法 2 Flink 容器重映射

输入:超载节点 n_{hot}、有向无环图 $G=(V(G),E(G))$、有向边权值集合 W、资源利用率 r_t。

输出:迁移结果。

1: $C_n \leftarrow$ the set of containers in node n_{hot};
2: $N_{target} \leftarrow$target node;//目标节点
3:Sort containers in C_n by RSV;//容器排序
4:**While** $r_t > k$ **do**
5: **If** $C_n \neq \varnothing$ **then**
6: Pick the $c \in C_n$ with the largest RSV value;
7: $N_{target} \leftarrow \mathrm{Max}(finalScoreNode(G,W,r_t))$;
8: Migrate c to N_{target};
9: **End If**
10:**End While**

在算法 2 中,集群中的超载节点已由负载均衡探测算法给出,但由于每个节点中各容器的体量不同,迁移代价也不同,故在超载节点队列中拥有最大 R_U 值的节点中,将所有 Flink 容器按 RSV 值降序排序,以选出迁移效率最高的待迁容器;然后为待迁容器进行节点评分,将分数最高的节点作为目标节点。其中,RSV 值为资源利用综合率指数 R_U 与容器内存占用大小 $size$ 的比值,如式(9)所示,资源使用率越高,容器内存越小,则 RSV 值越大,容器的迁移优先级越高。

$$RSV=\frac{R_U}{size} \tag{9}$$

算法按 RSV 值的大小依次执行容器的迁移程序,直到超载队列中所有节点的各资源均低于相关阈值,Flink 容器动态迁移结束。

容器迁移的具体设计架构图如图 4 所示,集群

(C)1994-2022 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

中每个 Worker 节点上都有一个资源监测模块,周期性地对当前节点的资源使用情况和负载均衡情况进行监测,并将监测结果实时反馈给 Master 节点的调度器;调度器根据资源信息选择超载节点和待迁容器,根据待迁容器的通信情况和各节点的负载信息,对节点进行评分,已选择目标节点进行迁移;直到集群中不再出现超载节点,迁移结束。

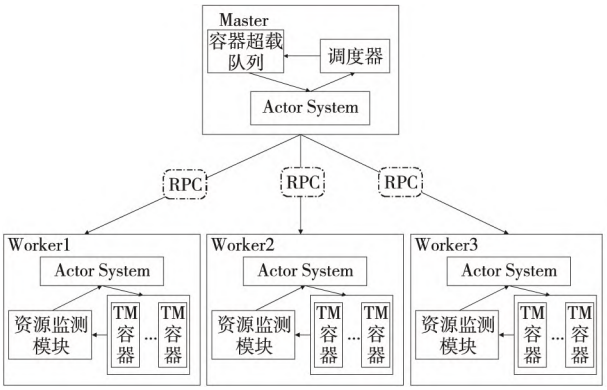

Figure 4 Architecture of container migration design

图 4 容器迁移设计架构

6 实验与结果分析

本节主要将 Flink 默认调度策略、Kubernetes 容器调度策略和本文提出的 FLBS 策略进行实验和对比分析,使用不同规模的数据集在不同类型的计算任务下从任务运行时间、资源利用率和节点间数据流大小 3 个方面验证 FLBS 的有效性。

6.1 实验环境与参数设置

实验使用的 Flink 集群包括 1 个 Master 主节点和 4 个 Worker 从节点共 5 台物理机,每台机器软件、硬件配置如表 1 和表 2 所示。

Table 1 Software configuration

表 1 软件配置

软件	版本
CentOS	7.5
JDK	1.8
Docker	18.09.2
Kubernetes	1.9
Prometheus	2.14.0
Flink	1.8.0
Hadoop	2.7.5
Hibench	7.0

本文使用大数据基准测试工具 Hibench 中的标准测试数据集,在不同类型的计算任务下进行实验,为保证实验结果的可靠性同时减小误差,通过多次重复实验的平均值作为最终结果,实验中涉及的参数配置如表 3 所示。

Table 2 Hardware configuration

表 2 硬件配置

配置项	配置详情
Master 主节点	四核 CPU,16 GB 内存,200 GB 硬盘
Worker1 节点	四核 CPU,8 GB 内存,200 GB 硬盘
Worker2 节点	四核 CPU,4 GB 内存,200 GB 硬盘
Worker3 节点	双核 CPU,4 GB 内存,200 GB 硬盘
Worker4 节点	双核 CPU,2 GB 内存,200 GB 硬盘

Table 3 Default parameters configuration

表 3 默认参数配置

参数	描述	值
K	阈值	0.75
ω_C	通信评分权重	1/3
ω_R	资源需求评分权重	1/3
ω_B	资源均衡评分权重	1/3
Taskmanager.numberOfTaskSlot	每个 TaskManager 中 slot 个数	2
Parallelism.Default	默认并行度	8

6.2 实验结果分析

图 5 展示了分别使用 2 GB,4 GB 和 6 GB 数据进行 WordCount 计算任务时,3 种调度策略运行时间的对比。从图 5 中可以看出,相较于其他 2 种策略,本文提出的 FLBS 在 3 种规模的数据下 Flink 任务的运行时间均有明显减少,且随着数据量的增加,优化效果逐渐明显。

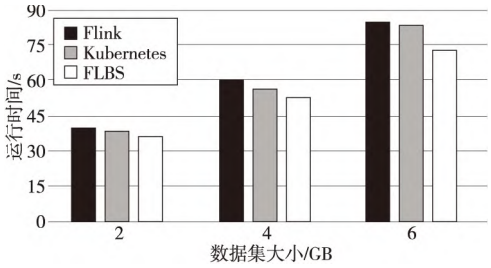

Figure 5 Comparison of task execution time on different scale datasets

图 5 不同规模数据集上任务运行时间对比

本文分别使用 WordCount、PageRank 和 TeraSort 3 种不同类型的计算任务来验证本文 FLBS 的普遍有效性,每种 Flink 任务均采用 4 GB 数据集,实验结果如图 6 所示。从实验结果可以看出,与 Flink 默认策略和 Kubernetes 调度策略相比,本文提出的 FLBS 在运行时间上均有一定的优化效果,其中,在 PageRank 任务上的优化效果最为

(C)1994-2022 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

明显。这是由于 PageRank 属于计算密集型作业，而 WordCount 属于数据密集型作业，TeraSort 属于 I/O 密集型作业，后 2 类作业中消耗较多的 I/O 资源，而本文提出的 FLBS 在资源均衡方面主要考虑了 CPU 和内存的利用率，故在计算密集型作业上的优势更为明显。

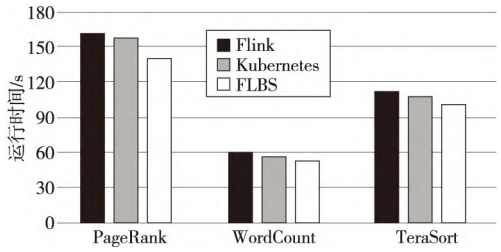

Figure 6 Comparison of execution time of different types of computing tasks

图 6 不同类型计算任务运行时间对比

为验证策略对容器化 Flink 集群中各节点均衡负载的有效性，本文使用 Prometheus 对各计算节点的 CPU 和内存的使用情况进行监测，实验结果如图 7 所示。从实验结果可以看出，Flink 默认调度策略在各节点上 CPU 和内存的负载差异较大。这是由于默认调度策略使用轮询的方式在节点间随机分配任务，没有考虑节点间的资源均衡，其中，节点 2 和节点 4 的 CPU 负载最大，节点 3 和节点 4 的内存负载最大，超出了预设阈值 0.75。运行 FLBS，执行容器迁移后，各节点的 CPU 和内存负载差异明显缩小且低于阈值 0.75。

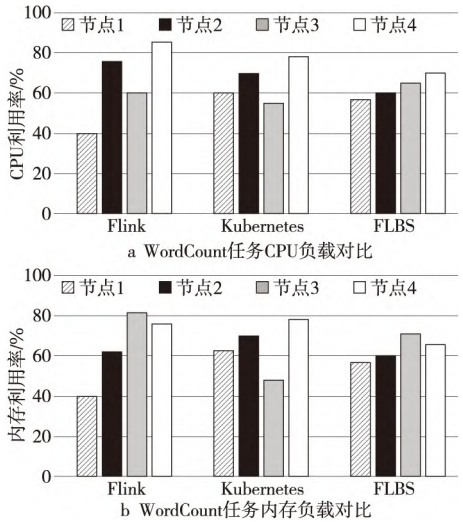

Figure 7 Load comparison of cluster nodes

图 7 节点负载对比

FLBS 策略在不同容器数量下对运行时间的影响如图 8 所示。从实验结果可以看出，Flink 任务的运行时间随容器数的增加逐渐减少，但当容器数增加到一定数量时，运行时间的下降速度开始逐渐减小。这是由于 Flink 集群中的容器数量增加时，容器间通信增加，算法的时间开销增加，总运行时间上升，但 FLBS 的平均运行时间仍小于 Flink 默认调度策略的。

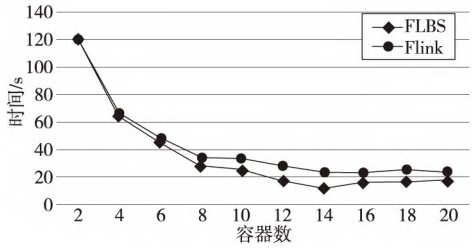

Figure 8 Execution time vs. the number of containers

图 8 运行时间随容器数变化

图 9 为在 Flink 默认调度策略下和本文提出的 FLBS 策略下各节点间数据流大小随时间变化曲线。从实验结果可以看出，2 种策略的节点间数据流均从 0 开始快速上升最后趋于稳定，而 FLBS 稳定时期的节点间数据流量明显小于 Flink 默认调度策略的，即容器间跨节点通信的数据流大小小于默认策略，可见，FLBS 在降低跨节点开销方面有明显效果且符合 Flink 容器通信代价模型，验证了策略的有效性。

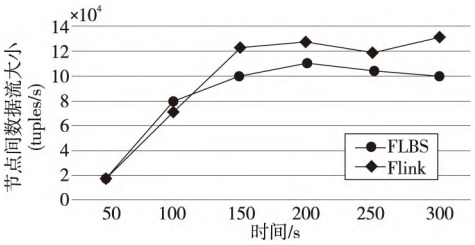

Figure 9 Data flow between nodes over time

图 9 节点间数据流大小随时间变化

7 结束语

通过归纳梳理现阶段国内外对于容器化大数据处理引擎任务调度优化的相关工作，发现现有的成果大多存在调度不灵活，与 Flink 平台不匹配，或未考虑容器间通信开销对负载均衡的影响等问题，因此本文提出了一种面向云环境的 Flink 负载均衡策略(FLBS)，以通信代价和负载均衡为衡量标准对容器进行迁移，在集群出现负载不均的情况时，能够有效减少作业运行时间，提高系统计算效率。

下一步的研究工作计划使用更大规模的数据集和更多类型的计算任务进行实验，对算法进行进

(C)1994-2022 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

一步优化,以减小迁移的时间和空间开销。

参考文献:

[1] Du Xiao-yong, Chen Hong. Big data management and analytics ecosystem: Independent and integrated development[J]. Frontier Science, 2019(2): 84-87. (in Chinese)

[2] Carbone P, Katsifodimos A, Ewen S, et al. Apache Flink: Stream and batch processing in a single engine[J]. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, 2015, 36(4): 28-38.

[3] Lü L, Zhang Y, Li Y, et al. Communication-aware container placement and reassignment in large-scale internet data centers[J]. IEEE Journal on Selected Areas in Communications, 2019, 37(3): 540-555.

[4] Ma Z, Shao S, Guo S, et al. Container migration mechanism for load balancing in edge network under power Internet of Things[J]. IEEE Access, 2020, 8: 118405-118416.

[5] Pickartz S, Eiling N, Lankes S, et al. Migrating Linux containers using CRIU[C]//Proc of International Conference on High Performance Computing, 2016: 674-684.

[6] Clark C, Fraser K, Hand S, et al. Live migration of virtual machines[C]//Proc of the 2nd Symposium on Networked Systems Design & Implementation, 2005: 273-286.

[7] Fan W B, Han Z J, Li P, et al. A live migration algorithm for containers based on resource locality[J]. Journal of Signal Processing Systems, 2019, 91: 1077-1089.

[8] Yin L X, Li P, Luo J. Smart contract service migration mechanism based on container in edge computing[J]. Journal of Parallel and Distributed Computing, 2021, 152: 157-166.

[9] Di Z Y, Shao E, Tan G M. High-performance migration tool for live container in a workflow[J]. International Journal of Parallel Programming, 2021, 49: 658-670.

[10] Jü Rui. Research on live migration of Docker container[D]. Wuhan: Wuhan University, 2017. (in Chinese)

[11] Anderson C. Docker [software Engineering] [J]. IEEE Software, 2015, 32(3): 102-104.

[12] Wang Zhi-feng, Zhao Yu-hai, Wang Guo-ren. Research and implementation of load balancing algorithm in heterogeneous Flink cluster[J]. Journal of Nanjing University Natural Science, 2021, 57(1): 110-120. (in Chinese)

[13] Wang Wen-hao, Shi Xue-rong. Node priority scheduling strategy based on heterogeneous Flink cluster[J]. Computer Engineering, 2022, 48(3): 197-203. (in Chinese)

[14] Hu Ya-hong, Sheng Xia, Mao Jia-fa. Task scheduling optimization in Spark environment with unbalanced resources [J]. Computer Engineering & Science, 2020, 42(2): 203-209. (in Chinese)

[15] He Zhen-zhen, Yu Jiong, Li Zi-yang, et al. Task scheduling strategy based on Flink environment[J]. Computer Engineering and Design, 2020, 41(5): 1280-1287. (in Chinese)

[16] Li Zi-yang, Yu Jiong, Wang Yue-fei, et al. Load prediction based elastic resource scheduling strategy in Flink[J]. Journal on Communications, 2020, 41(10): 92-108. (in Chinese)

[17] Li Zi-yang, Yu Jiong, Bian Chen, et al. Flow-network based auto rescale strategy for Flink[J]. Journal on Communications, 2019, 40(8): 85-101. (in Chinese)

[18] Felter W, Ferrira A, Rajamony R, et al. An updated performance comparison of virtual machines and Linux containers[C]//Proc of 2015 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), 2015: 171-172.

[19] Bernstein D. Containers and cloud: From LXC to Docker to Kubernetes[J]. IEEE Cloud Computing, 2014, 1(3): 81-84.

[20] Hindman B, Konwinski A, Zaharia M, et al. Mesos: A platform for fine-grained resource sharing in the data center [C]//Proc of the 8th USENIX Conference on Networked Systems Design and Implementation, 2011: 295-308.

附中文参考文献:

[1] 杜小勇,陈红. 大数据管理和分析系统生态:独立与融合发展并存[J]. 前沿科学,2019(2):84-87.

[10] 鞠瑞. Docker 容器热迁移方案研究[D]. 武汉:武汉大学,2017.

[12] 汪志峰,赵宇海,王国仁. 异构 Flink 集群中负载均衡算法研究与实现[J]. 南京大学学报(自然科学),2021,57(1):110-120.

[13] 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.

[14] 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.

[15] 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.

[16] 李梓杨,于炯,王跃飞,等. Flink 环境下基于负载预测的弹性资源调度策略[J]. 通信学报,2020,41(10):92-108.

[17] 李梓杨,于炯,卞琛,等. 基于流网络的 Flink 平台弹性资源调度策略[J]. 通信学报,2019,40(8):85-101.

作者简介:

徐浩桐(1996 -),男,内蒙古赤峰人,硕士生,研究方向为大数据和流计算。**E-mail**:xu_haotong@163.com

XU Hao-tong, born in 1996, MS candidate, his research interests include big data, and stream computing.

黄山(1986 -),男,辽宁锦州人,博士,讲师,研究方向为大数据和云计算。**E-mail**:huangshan@dlnu.edu.cn

HUANG Shan, born in 1986, PhD, lecturer, his research interests include big data, and cloud computing.

(C)1994-2022 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net