高技术通讯 2022 年 第32 卷 第8 期:845-858

doi:10.3772/j.issn.1002-0470.2022.08.007

面向大电网在线实时仿真的通信敏感资源调度①

袁雨馨②[*,**] 唐宏伟[*] 赵晓芳③[*] 严剑峰[***] 周二专[***]

([*]中国科学院计算技术研究所 北京 100190)

([**]中国科学院大学 北京 100049)

([***]中国电力科学研究院 北京 100192)

摘 要 面向大电网的实时仿真计算被建模为协同运行的多进程并行计算任务的集合,任务调度与资源分配的优劣是其能否实现实时性目标的关键因素之一。本文针对大规模电网机电-电磁混合仿真计算的应用场景,根据任务进行了特性分析,总结了资源利用率规律,提出了一种通信敏感的组调度框架(CGS)。该框架提出了一种集中式两阶段调度架构,以在不中断在线流运行过程中对任务进行主动采样和调度,达到精准预测需求的目标;提出了基于通信图的图划分策略与基于调度模型的匹配策略相结合的CGS调度算法,实现了进程组调度,降低了任务跨节点的通信开销。实验从任务平均划分数、平均割边成本、有负载节点数、集群资源利用率和资源碎片率指标方面对5种基线算法与应用了CGS的算法进行对比测试,结果表明CGS与基本策略相比至少降低了37%的进程间通信开销,减少了19%的资源碎片,平均提高了34%的集群资源利用率。

关键词 进程组调度;在线实时电网仿真;通信敏感;多维资源分配

0 引 言

大规模电网仿真计算是进行电力系统稳定性分析的重要手段,通过对大规模电力系统的全数字建模与仿真,可以掌握电力网络及各种动态元件的稳态、暂态和动态特征,进而保证电网安全稳定运行[1]。传统大电网数字仿真技术为基于离线数据进行仿真计算,采用电网生产运行提供的历史数据进行潮流、短路电流与暂态稳定等仿真分析,计算时间长、计算量大,对失稳状况只能进行事后分析处理,无法对电网在线运行提供辅助决策。目前,随着电网实时数据采集精度和效率不断提升,利用在线运行数据进行实时仿真甚至超实时仿真成为了可能,在线实时仿真能够在短周期内快速评估其安全状况及变化趋势,并给出辅助决策控制策略,为事故预警提供更灵活的支持[2-4]。

针对具有泛在工业物联网属性的现代电网,国家电网提出了智能全景电网[4](intelligent panoramic grid, IPG)的概念,在线超实时机电-电磁混合仿真(hybrid electromechanical and electromagnetic transient simulation, TS-EMT)是其核心计算引擎之一。混合仿真不仅实现了在多核节点及集群上并行执行以提高效率[5],还接入了来自于智能电网调度技术支持系统(D5000)[6]的在线实时数据流,实现了同步调短周期并行计算。在集群环境下,仿真任务执行效率受到资源和通信的约束,任务在执行中一旦遇到资源分配不足或通信开销过大的情况,极有可能导致执行超时,从而影响结果判定。因此,将任务部署到合适的工作节点上并分配合理的资源,成为了大电网在线仿真能否实现实时性目标的关键之

① 国家重点研发计划(2018YFB0904503)资助项目。

② 女,1995 年生,博士生;研究方向:分布式计算,云计算;E-mail: yuanyuxin17z@ict.ac.cn。

③ 通信作者 E-mail: zhaoxf@ict.ac.cn。

(收稿日期:2021-03-05)

一。

本文的主要贡献如下:(1)对大规模电网的机电-电磁暂态混合仿真任务进行了特性分析,总结出了资源利用率规律,即仿真任务具有短时运行、资源需求较稳定、对通信敏感较高的特点;(2)提出了一种通信敏感的组调度框架(communication-aware gang scheduling framework, CGS),采用集中式两阶段调度架构,在不中断在线流运行过程中对任务进行采样和调度,最大程度保障在线实时任务运行的稳定性;(3)提出了一种 CGS 调度算法,实现了对任务多资源的主动采样以预测需求,并基于通信图对任务进行进程级的组调度[7](gang scheduling)。实验表明,CGS 降低了 37% 的进程间通信开销,减少了 19% 的资源碎片,平均提高了 34% 的集群资源利用率。

1 相关研究

目前,任务管理与资源调度计算平台[8-10]大多是基于资源请求的,而实际中资源需求与执行复杂度和集群硬件密切相关,使得提交的请求往往难以准确提出资源需求,导致集群资源碎片化严重、利用率较低。基于大规模电网实时仿真这一特定领域,CGS 提出了一种基于采样的资源调度方法,类似于 Sparrow[11]的批量抽样,根据实际运行数据不断修正调度策略。

现有的调度方法由面向约束机制组成。Google Borg[9]考虑到任务的优先级特征,设计了任务排序算法和低优先级任务替代机制。YARN[8]最近进行了扩展,支持多种资源类型、优先级、抢占和高级接纳控制。Apache Mesos[10]为异构框架实现了一种基于 offer 的方法,主要采用主导资源公平(dominant resource fairness, DRF)算法[12]实现了公平性。Sparrow[11]采用队列模型,提出了一种批量采样方法,其目的是考虑优先级、公平性、异质性和数据的位置性。Tetris[13]支持完成时间敏感任务的多资源分配。CGS 针对在线调度,提出了一种通信敏感的任务多资源分配策略,考虑了任务进程级别下的通信相关性,最大化满足进程间的亲和性约束。

2 电网在线实时仿真概述

2.1 电网在线实时仿真介绍

本研究针对的是大规模电网的机电-电磁暂态混合仿真(TS-EMT)任务的调度问题,TS-EMT 是描述电力系统物理特性的解决方案[14-15]。机电瞬态稳定性仿真(electromechanical transient simulation, TS)主要用于处理大型电力网络,仿真速度快,而电磁瞬态仿真(electromagnetic transient simulation, EMT)则在较小的范围内对电网进行高时间分辨率仿真。混合仿真是 TS 和 EMT 的统一,可以利用 TS 仿真的速度模拟非常大的网络,同时在关键部件上提供 EMT 仿真的精度[16]。具体而言,混合仿真任务包括 TS 和 EMT 两个过程,并结合二者优势进行综合分析。

本文讨论的每个仿真任务负责对电网拓扑结构中的不同故障进行 TS-EMT 混合仿真,任务内的进程间通信基于消息传递接口(message passing interface, MPI)协议。文献[17]总结并优化了大规模电力系统的并行仿真算法,指出电网可以被分割成多个子网,并以联络线相连接,每个子网由其计算进程(COMP)计算,而管理计算和计算联络线、汇总结果和 I/O 进度的过程被称为控制进程(CTRL)。

文献[18]阐述了电力系统实时仿真的关键是仿真执行时间小于等于求解模型方程的仿真时间步长。为了实现在线实时仿真,除了自身的计算执行时间外,由调度(如等待更多资源或通信)造成的延迟决定了任务处理的延迟。

仿真任务由应用和配置组成。应用指的是仿真计算逻辑,配置包括仿真规模、仿真步长、电网分割策略、故障类型等仿真参数。影响计算的因素有很多,例如不同的电网分割策略也会导致不同的计算量,不同的电网故障类型相应的计算量也会有所不同。一般来说,那些具有相同应用和配置的任务可以被视为相同的任务,在不同的潮流数据下,其执行时间和资源利用率都是相似的。

2.2 机电-电磁暂态混合仿真任务特性分析

任务执行环境如表 1 所示,以表 2 中列出的 3

个典型故障下的 TS-EMT 混合仿真任务为例。这些任务相互独立,代表了不同故障下的仿真,数据来自 2019 年国家电力调度通信中心,覆盖了 6 个地区(即华北、华东、华中、西北、西南、东北)的电网。3 个任务的仿真步长均为 10 s,每个任务中的每个过程按照执行时间采样约 100 次。

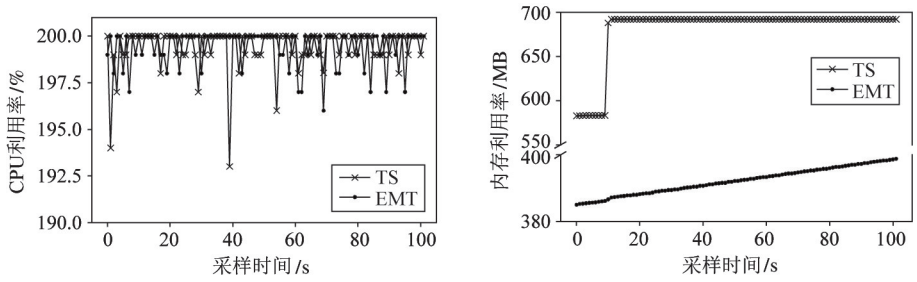
图 1 中对应任务的 3 张图描述了不同任务的整体资源使用情况。左边的子图展示了每个任务的中央处理器(central processing unit,CPU)使用率,右

表 1　分析环境

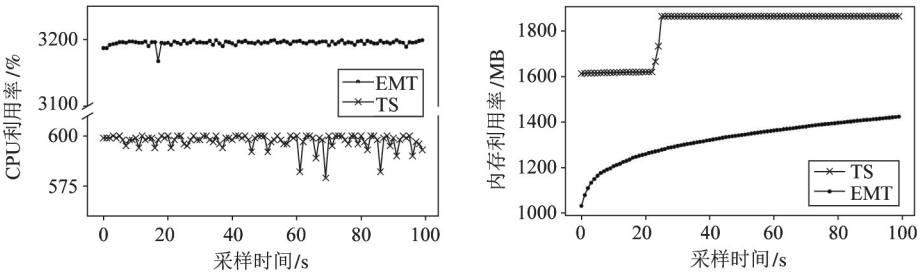
系统配置	规格参数
系统平台	sugon X785-G30 服务器
CPU	Dual 2.10 GHz Intel(R) Xeon(R) Gold 6230 20 cores/80 threads
内存大小	256 GB
网络	Intel I350 Gigabit
操作系统	CentOS 7.6.1810
MPI	Intel MPI Library 2019

表 2　TS-EMT 任务相关信息

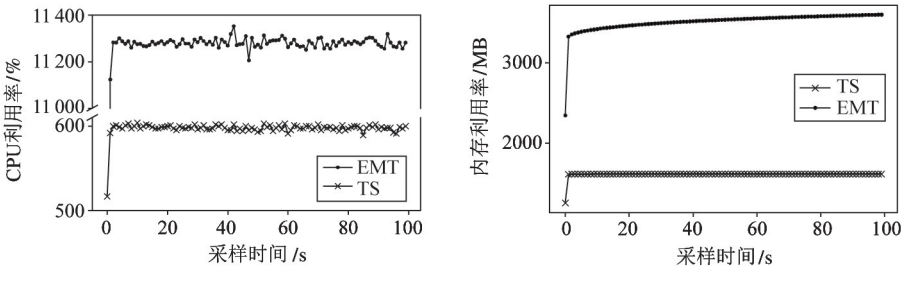
任务号	电网规模	故障类型	TS		EMT	
			子网数	进程数	子网数	进程数
1	10 000 + 节点	开路故障	2	2	1	2
2	40 000 + 节点	短路故障	6	6	31	32
3	40 000 + 节点	无	6	6	112	113



(a) 任务 1 资源利用率



(b) 任务 2 资源利用率



(c) 任务 3 资源利用率

图 1　任务资源利用率采样曲线

边的子图展示了这段时间内的内存使用率。在所有的剖析结果中,CPU 的利用率都很高,几乎达到了 $n \times 100\%$（n 为进程数),证实了这些任务是受 CPU 约束的。同时,内存使用率呈现上升趋势,说明对内存的需求在增加。另外,从 3 种情况的对比来看,任务规模越大,所需的计算资源越多。

为了深入分析内存相关的统计数据,表 3 和表 4 汇总了任务中进程的内存使用情况。在每个任务中,COMP 的变异系数(coefficient of variation, CV)较小,而 CTRL 的平均值比 COMP 大。原因是由于 COMP 需要对网格进行尽可能均匀地划分,所以 COMP之间的内存使用量是相似的。而CTRL需要做的其他工作包括聚合、通信、边界接触线的计算等。这个观察结果可以用于生成大量的模拟实验数据,使其更贴近实际生产环境。

表 3　TS 进程内存分析

任务号	COMP		
	进程数	平均数	变异系数
1	2	346.05	0.092
2	6	310.93	0.061
3	6	269.74	0.067

表 4　EMT 进程内存分析

任务号	CTRL		COMP		
	进程数	平均数	进程数	平均数	变异系数
1	1	383.83	2	346.05	0.092
2	1	171.48	6	310.93	0.061
3	1	658.12	6	269.74	0.067

由于仿真是以流式数据驱动运行的,因此需要观察多个截面数据在一段时间内的资源使用情况。任务 2 选取连续的电网实时潮流数据集进行多次执行,截取部分结果绘制在图 2 中。从图 2 中可以发现,不同的数据下多次执行的资源需求曲线差异很小,而且在资源充足的情况下,同一仿真任务中每个执行的完成时间都是相似的。因此,在线调度框架在流式过程中可以对资源使用情况进行采样,作为需求预测。

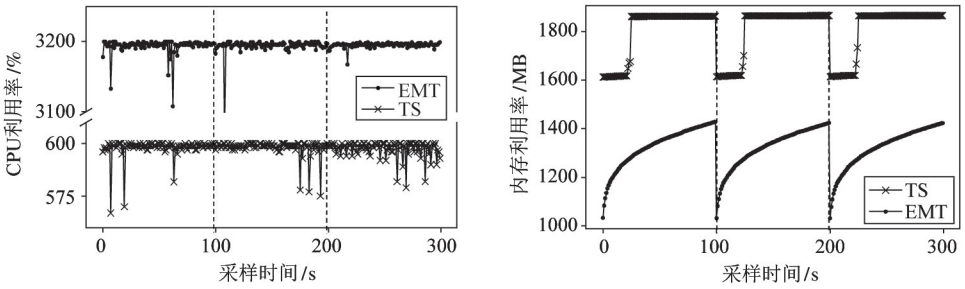


图 2　任务 2 资源利用率采样曲线

另一个特性分析是为了表现进程间的通信关系。文献[19]阐述了 EMT 网络分区的拓扑方案,该方案是由子网的相互连接形成的。图 3 通过分析任务 3 的通信关系得出进程间通信热力图,横纵坐标均为进程号,图中点代表横向到纵向进程的通信数据量,颜色越深代表进程传输的数据量越大。该图表明,通信强度是不平衡的,进程间通信的差异化使得进程聚集成组成为了可能。

由上文分析可知,仿真任务具有短时运行、资源需求较稳定、对通信敏感较高的特点。根据这些特点,面向大电网的在线实时仿真调度必须做到的就

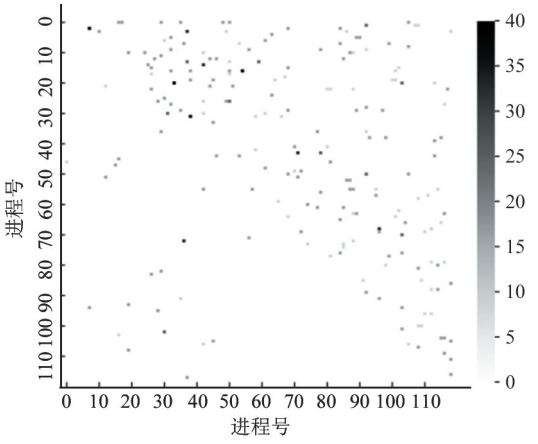


图 3　任务 3 的通信热力图

是保障在线实时运行效率,并解决资源分配的准确性,提高集群的资源利用率,减少碎片化。

3 在线调度问题描述

为了解决多资源上的在线调度问题,本文提出了通信敏感组调度框架(CGS)。本节介绍 CGS 中的调度模型,作为在线调度的形式化表示。

3.1 定 义

假设一个集群有 N 个资源容量为 $\{\vec{\boldsymbol{R}}_1,\vec{\boldsymbol{R}}_2,\cdots,\vec{\boldsymbol{R}}_N\}$ 的异构节点,节点 h 的容量可以由一个 d 维向量 $\vec{\boldsymbol{R}}_h=(R_h^1,R_h^2,\cdots,R_h^d)$ 表示,如果节点没有某些类型的资源,比如图形处理器(graphic processing unit, GPU)或特定的硬件,向量中的元素可以用 0 来填充。

一个仿真任务由 n 个 MPI 进程组成,资源需求被写作 $\{\vec{\boldsymbol{D}}_1,\vec{\boldsymbol{D}}_2,\cdots,\vec{\boldsymbol{D}}_n\}$,任务中进程 p 的资源需求可以由一个 d 维向量 $\vec{\boldsymbol{D}}_p=(R_p^1,R_p^2,\cdots,R_p^d)$ 表示。一般来说,每个进程都会被绑定在一个 CPU 核心上,内存则会按需分配。任务的需求则是所有进程资源需求的总和,即为 $\boldsymbol{D}=\sum_{p=1}^{n}\vec{\boldsymbol{D}}_p$。

根据式(1)可知资源向量可比性,从而可判断进程 p 的资源向量是否可以被节点 h 满足。具体而言,当所有维度的进程需求 $\boldsymbol{D}_p^i$ 小于节点可用容量 $\boldsymbol{R}_h^i$ 时,可以判断节点能够满足该进程需求。换言之,如果有一个维度的资源不能满足任务的需求,资源匹配就会失败。

$$\forall i(0<i\leqslant d)R_1^i<R_2^i\leftrightarrow\vec{\boldsymbol{R}}_1<\vec{\boldsymbol{R}}_2 \tag{1}$$

当进程数 $n>1$ 时,进程间的通信基于 MPI 协议。计算进程 p_j 负责仿真分网拓扑下的子网 j,并向邻居集 $set(p_{j'})$、$p_j\neq p_{j'}$ 传输数据。任务进程间通信集的拓扑结构描述为一个有向无环图 $G=(V,E)$,其中 V 是 n 个顶点的有限集,对应 n 个进程,$E\subset V\times V$ 是一个有定向边的有限集,代表顶点之间的通信关系。图 G 满足如果 $\{j,j'\}\in E$,则 $j\in E\wedge j'\in E$。

假定任务可将 n 个进程划分为 $b_1\cup b_2\cup\cdots\cup b_k$ 这 k 个划分,并满足约束 $b_l\cap b_{l'}=\varnothing$,$l\neq l'$ 和 $\cup_{l=1}^{k}b_l=V$,那么每个划分 b 就可以部署在一个节点上,不同划分就可以部署在不同节点上,从而实现任务的跨节点调度。每个划分均为多个进程的集合,称为进程组(gang)。

当定义了任务划分后,边 $\{j,j'\}$ 的通信开销 $\omega(\{j,j'\})$ 如式(2)所示,与进程间的通信长度 $L_{j,j'}(bits)$、带宽 B 和在不同节点下的性能损耗 $\varepsilon\in(0,1)$ 有关。

$$\omega(\{j,j'\})=\begin{cases}0 & \text{若 } j\in b\wedge j'\in b\\ \dfrac{L_{j,j'}}{B\times(1-\varepsilon)} & \text{其他情况}\end{cases} \tag{2}$$

通信图 G 刻画了进程间的通信关系。为简单描述,对图的所有边和顶点引入二元决策变量 $e_{j,j'}$,对于每条边 $\{j,j'\}\in E$,变量取值为 $e_{j,j'}\in\{0,1\}$,当 $\{j,j'\}$ 为割边时 $e_{j,j'}=1$,否则 $e_{j,j'}=0$。由于位于同一节点上的连通性边缘的通信开销可以忽略,因此总开销 $COMM$ 是任务划分的割边成本。

$$COMM=\sum_{\{j,j'\}\in E}e_{j,j'}\omega(\{j,j'\}) \tag{3}$$

由于任务进程采用 MPI 通信协议,其中有较多同步操作,降低进程组的割边成本有助于减小由于通信造成的同步等待时间,从而提高任务执行的效率。

3.2 调度模型

调度负责将在线提交的任务与最多个节点进行匹配,最小的调度单元是进程,它只能分配给一个节点。调度的主要目标是尽可能减少有负载的节点数量,更好地降低能耗,实现绿色节能计算,并且空闲的节点可以作为备用节点,以实现高可用性或为其他计算提供服务。除了这个目标之外,还要考虑到多节点的通信开销。

调度模型可以形式化为多维装箱问题[20](multi-dimensional bin packing problem, MD-BPP),它是经典的一维装箱问题的推广,是 NP 难问题。在多资源调度的基础上,在多节点上调度一个任务的多个进程是复杂的,这也是 TS-EMT 混合仿真任务调度的难点。为了解决这些问题,调度模型采用整数线性模型将 MD-BPP 解与图划分策略相结合,其决策变量如下。

(1) x_{ij}: 若进程 i 被分配到节点 j 上,则 $x_{ij}=1$,否则 $x_{ij}=0$。

(2) y_j: 若节点 j 有任务进程在运行中,则 $y_j=1$,否则 $y_j=0$。

(3) z_{ib}: 若进程 i 在子任务 b 中时 $z_{ib}=1$,否则 $z_{ib}=0$。

调度模型的线性整数规划公式可以写为

$$\min \quad Z_1=\sum_{j=1}^{N} y_j \tag{4a}$$

$$Z_2=\sum_{\{i,\,i'\}\in E} e_{j,j'}\omega(\{j,j'\}) \tag{4b}$$

$$\text{s.t.} \quad \sum_{j=1}^{N} x_{ij}=1 \qquad \forall i\in\{1,\cdots,n\} \tag{4c}$$

$$\sum_{i\in b}\vec{\boldsymbol{D}}_i x_{ij}\leqslant\vec{\boldsymbol{R}}_j y_j \quad \forall b,\forall j\in\{i,\cdots,N\} \tag{4d}$$

$$e_{i,\,i'}\geqslant z_{ib}-z_{i'b} \quad \forall\{i,\,i'\}\in E,\forall b \tag{4e}$$

$$e_{i,i'}\geqslant z_{i'b}-z_{ib} \quad \forall\{i,i'\}\in E,\forall b \tag{4f}$$

$$\sum_{b} z_{ib}=1 \qquad \forall i\in V \tag{4g}$$

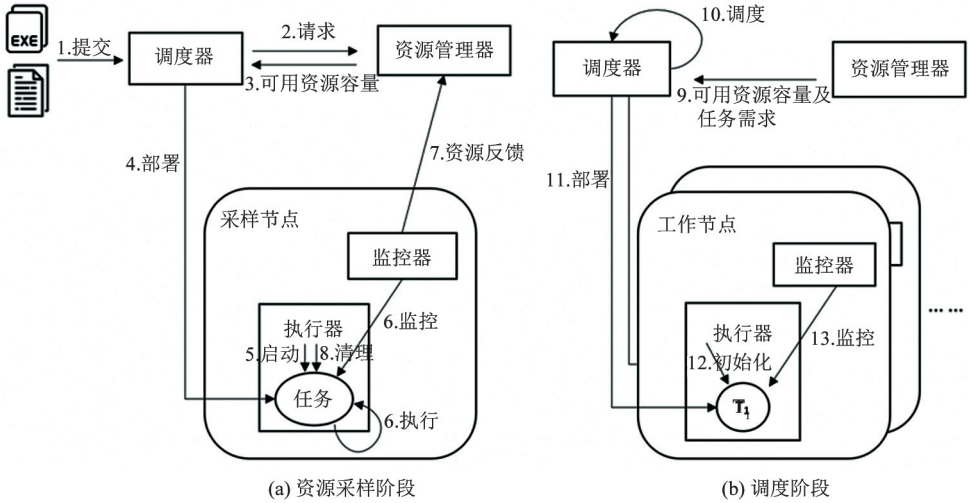
目标是最小化有负载节点数量和任务划分的割边成本,用式(4a)和式(4b)表示。约束条件式(4c)说明一个任务的任何进程都应且必须部署在一个节点上,而式(4d)说明节点上的进程需求之和不应该超过节点的可用容量,确保资源不会被过度分配。式(4e)和(4f)保证了一个有效的划分,式(4g)保证了每个进程正好分配到一个分区。最后,可以得到一个集合 $(x_{ij},\ y_j)$,将任务的所有进程与集群中的节点进行匹配。

4 通信敏感的组调度 CGS

本节首先介绍 CGS 的体系结构,再具体说明 CGS 算法与调度模型,最后介绍了 CGS 框架的实现方式。

4.1 CGS 架构

CGS 架构如图 5 所示,可归为集中式调度框架,由控制节点上的调度器、资源管理器以及工作节点上的执行器和监控器组成。调度器通过非侵入式采集器收集资源需求,负责调度任务。资源管理器负责检测集群的状态及最新的可用容量。执行器负责任务的全生命周期活动管理,监控器负责记录任务和节点的资源使用情况。架构上将资源管理服务与调度服务解耦,可部署多种不同优先级的调度策略,方便更换策略,增强了灵活性。



(a) 资源采样阶段　　(b) 调度阶段

图 5　CGS 两阶段架构工作流

架构采用了先采样后调度两阶段架构。采样是调度的基础,是为了收集和预测任务资源需求,包括将在线任务部署在采样节点上,以非侵入式的方式持续收集使用情况并估算资源需求。采样工作流程如图 5(a)所示。任务由应用程序和一些配置文件组成,将在流式过程中随时提交。提交后,调度器会向资源管理器申请最新的可用容量,再将任务部署在采样节点上。之后执行器开始运行任务,同时监

控器来记录任务的执行情况和资源使用情况。在多次采样获得稳定需求后,监控器会将任务资源使用情况报告给资源管理器,然后执行器会清理执行垃圾,为下一次任务采样做准备。

图5(b)展示了调度阶段。详细来说,调度器使用了预定义的调度策略,根据任务需求和集群可用容量运行CGS算法(4.2中有更详细介绍),将任务部署在节点上,为所有进程分配一定的资源并进行资源隔离。之后执行器初始化任务,当任务准备就绪,且输入数据接收完毕后,任务会在监控器的监视下执行。一旦任务抛出资源使用异常,监控器就会扩大使用阈值,直到节点上没有资源可用,然后执行器就会清除该任务并予以警告,以免影响其他任务,清除后的任务将会被允许重新提交。此外在任务执行过程中,执行器会与控制节点保持联系。

4.2 CGS算法

CGS的一个重要组成部分是组调度的设计,它结合了贪心调度和图划分策略。该算法引用了分治法[21]思想。算法可分为以下3个部分:

(1) 分解。将任务划分为子任务。

(2) 解决。递归搜索最优节点,直到匹配成功。

(3) 合并。组合子任务节点集得到最终的解决方案。

分治算法判断子问题能否解决的方法是过滤,即找到部署任务的可行候选节点集。如果集合为空,说明目前任务需求不能被任何一个节点满足,那么任务将被递归地划分成子任务,即子进程组。只要候选集不为空,就会结束递归分支。

任务划分意味着必然出现跨节点调度,进程间的通信开销将成为效率的瓶颈。为了降低开销,CGS利用了图划分策略,如k-way多级划分算法[22]或其他启发式算法[23]。图划分策略平衡了处理器之间的工作负载,使得通信开销最小化。

CGS算法利用扁平化匹配策略搜索最优节点以求解MD-BPP,该策略同时考虑了多维资源,以获得一个适应值,表示为*SCORE*。该值量化了任务与节点的匹配程度,最常见的方法是在Grandl等人[13]提出的多资源启发式策略,包括余弦相似度、点积、L^2距离等。这些方法都考虑了集群中的多维资源,并将任务需求与最优节点进行匹配,CGS框架并没有限制*SCORE*的计算方式,可以根据实际测试选择最优的资源匹配计算方案。

CGS算法递归实现的流程如算法1所示。

算法1 通信敏感混合粒度调度算法CGS

输入: 集群所有节点H可用资源容量$\vec{R}$
任务所有进程资源需求$\vec{D}$
任务进程通信图G

输出: 任务是否被调度(true or false)

1: $Candidates \leftarrow \varnothing$
2: for $h \in H$ do
3: if $\vec{D} \leqslant \vec{R_h}$ then
4: $Candidates \leftarrow Candidates \cup h$
5: 计算$SCORE(\vec{D},\vec{R_h})$
6: end if
7: end for
8: if $Candidates \neq ?$ then
9: 通过*SCORE*选择最优*Candidates*
10: 更新$\vec{R_h}$
11: else if G不能再被分割了 then
12: returnfalse
13: else
14: 将任务划分为子任务P'并得到子任务需求D'和图G':
$P', D', G' \leftarrow Partition(\vec{D},G)$
15: for all $p \in P'$ do
16: $CGS(\vec{R},\vec{D'_p}, G'_p)$
17: end for
18: end if
19: return true

在CGS算法的形式描述中,前7行用于筛选出有足够资源的节点(*Candidates*),并计算出与任务的匹配度。第8行到第10行表明,如果有一些*Candidates*,将其部署在最佳匹配节点上并更新资源。第11和12行表明,如果任何节点的可用容量不足,任务将不会被调度。第13行到第18行利用图划分将任务分割成子集,并递归地重新调度。

该框架将图划分策略与基于调度模型的匹配策略相结合。CGS算法中,任务的图划分策略采用k-way多级算法,将通信图中包含m条边的n个进程任务划分为k个子任务,划分图的时间复杂度为$O((n+m)\times\log(k))$,递归实现的总时间复杂度为

$O((n+m)\times\log(k)\times\log n)$。

4.3 CGS 实现

CGS 包括资源采样和调度 2 个阶段。在采样阶段,调度器接收并解析任务信息,包括进程数和一些执行配置,同时会使用性能分析工具如 Intel Vtune Profiler[24] 收集 MPI 进程间通信开销。采样过程被设计成多次重复,直到得到一个比较稳定的值(根据经验通常是 3 ~ 5 次),为了减少由于资源限制导致的任务失败或超时,可以将进程的内存需求放宽到最大内存的 δ 倍。

在资源调度方面,调度器会筛选出满足约束条件的节点,并使用优化的资源匹配策略来匹配任务。此外,资源管理器在任务提交或固定时间段内,会触发监控器收集任务需求和可用容量。此外,调度器、资源管理器、监控器和执行器会作为框架的常驻服务运行。

5 实验与分析

为了更全面地评估 CGS 的调度性能,而不仅仅局限于 2.2 节所述的任务,本文进行了模拟实验。本节首先讨论了实验环境,再评估性能结果。

5.1 实验指标

实验设置了一个大规模节点和一批任务的环境,分别从任务平均划分数、平均割边成本、有负载节点数、集群资源利用率和资源碎片率 5 个指标对算法进行比较。为了显示出 CGS 调度框架的优势,实验采用了 5 种常用的调度策略作为基线算法,包括自适应先到先得[25](adaptive first-come-first-served, AFCFS)、最大组优先调度[25](largest gang first served, LGFS)、最佳适应算法[25](best fit decreasing, BFD)、主导资源公平算法[12](dominant resource fairness, DRF)与 Tetris[13]。实验比较了基线算法与应用了 CGS 的算法之间的差异。调度算法均以进程为最小单元进行调度。详细的性能指标介绍如下。

(1) 任务平均划分数。任务划分的数量反映了策略中对任务进程之间亲和力的考虑。划分数量越多,进程被调度的越分散,任务执行效率越低。该指标较低有助于保证任务执行的效率和稳定性。

(2) 平均割边成本。该值表示跨节点的通信开销,如式(2)所示。该开销由边缘切割权重决定,成本越低表明跨节点通信开销越小。

(3) 有负载节点数。该指标衡量的是集群中的负载聚集程度,降低碎片的同时更好地降低能耗,实现绿色节能计算。

(4) 集群资源利用率。该指标会对每个节点的不同维度资源进行加权。式(5)中 u_k^i 用于表示节点 k 在维度 i 上的资源需求与可用容量之比,式(6)中 u_k 衡量节点 k 在所有维度中的平均资源利用率。值越大代表资源利用率越高、闲置越少。

$$u_k^i = \frac{\sum_{j\in k} D_j^i}{R_k^i} \tag{5}$$

$$u_k = \frac{1}{d}\sum_{i=1}^{d} u_k^i \tag{6}$$

(5) 集群资源碎片率。资源碎片率反映了一个节点中资源耗尽而其他资源剩余的硬约束,这种剩余的碎片化导致了资源的浪费,无法再用于任何其他任务。式(7)显示的是节点 k 的碎片率,式(8)中 w 衡量的是集群中所有节点的资源碎片程度。

$$w_k = \begin{cases} \dfrac{1}{d}\sum_{i=1}^{d} \dfrac{R_k^i - \sum_{j\in k} D_j^i}{R_k^i} & 若(\exists i)u_k^i = 1(0 < i \leqslant d) \\ 0 & 其他情况 \end{cases} \tag{7}$$

$$w = \frac{1}{n}\sum_{k=1}^{n} w_k \tag{8}$$

5.2 实验环境

虽然 CGS 支持任何维度的资源,但在实时模拟中,其性能主要取决于 CPU 和内存资源,记为 $\vec{R}_h = (C_h, M_h)$。本文电网实时仿真环境中,CPU 和内存资源被认为是同等重要的。实验设置了一个由 1000 节点组成的集群,其中 CPU 和内存容量来自一个截断的正态分布。实验 1 探究了不同 CPU 和内存下算法的有效性,实验设置了集群中每个节点的 CPU 平均值为 10 ~ 50 核,步长为 10,内存范围为 300 ~ 1500 MB,步长为 300,变异系数设置为 1.5。实验 2 探究不同变异系数下的算法在评价指标体系

下的表现,因此本实验重点分析了在 CPU 和内存均值分别为 30 核和 1000 MB 环境下方法的有效性。由于在实际场景中,集群可用容量几乎不可能是完全同质的,因此变异系数从 0.5 到 3.0,以 0.5 为增量,产生 6 种不同的集群环境。随着变异系数的增加,资源的异质性逐渐增强。

实验构建了 100 个在线任务,模拟的需求是基于 2.2 节中的分析结果。实验设置 TS 的数量小于 EMT 的数量,并随机生成 2 ~ 200 的进程数量,其中 3% 的进程被设置为 CTRL,其余都是 COMP。在所有进程中,CPU 资源需求被设置为 1,内存则存在差异。在 TS 中,内存是由截断的正态分布产生的($Mean = 300$, $CV = 0.1$)。在 EMT 中,CTRL 中内存需求的平均值为 400,COMP 中截断正态分布设置为 $Mean = 30$, $CV = 0.3$。此外,通信图是根据相应的任务进程数生成的,边权重设置为 0 ~ 10。如果一对连接的节点被分割成两组不同的节点,权重将被加到整体的割边成本中。

所有的模拟任务在表 1 所述的平台环境中被调度,实验首先对 5 种基线算法进行单独实验(称为 X),再分别利用这 5 种算法计算 *SORE* 的方式置入 CGS 算法中(称为 X + CGS),以评估在 CGS 图划分策略与匹配策略相结合的算法下的优势,共 8 种策略。实验参数采用 $k = 2$,$\delta = 1.5$。

5.3 实验结果

实验 1 探究了不同 CPU 和内存下算法的表现情况,为了方便展示,实验分别选取了 CPU 为瓶颈和内存资源为瓶颈时的两种场景展示。首先当固定 CPU 而内存以一定步长增长时,各方法在不同指标下的结果如图 6 所示,其中未分配数指标表示了由于集群节点资源短缺导致集群无法满足的任务个数,即任务调度失败的个数。图 6 中虚线表示基线算法,实线表示应用了 CGS 的算法,随着内存的不断增大,CPU 愈发成为了瓶颈资源,使得集群利用率有所下降而碎片率上升。图 6 结果表明,CGS 方法在不同内存场景下保持了负载聚集从而降低能耗,使得更多任务得以调度执行,并大幅降低了跨节点通信开销,提高了资源利用率并降低了碎片化程度。

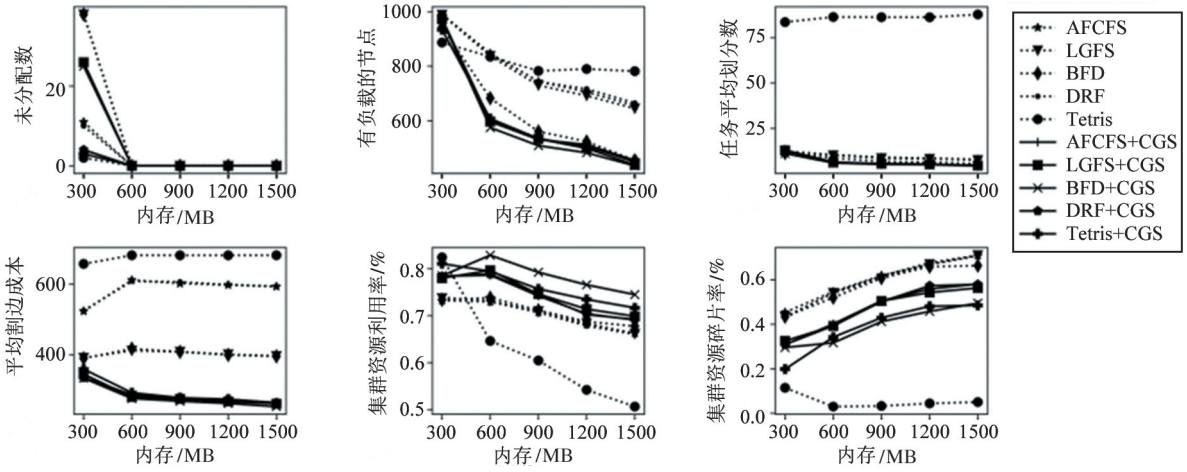


图 6 CPU 为 10 时不同指标下各方法随内存增长的结果

其次,当固定内存而 CPU 以一定步长增长时,各方法在不同指标下的结果如图 7 所示。从图中发现 CGS 方法在 CPU 资源不同时依旧比基线算法有性能优势,尤其是在降低通信开销上。

在实验 2 中,每个指标的结果用分别代表了 6 种不同的异构环境下的子图表示,每个子图都以柱状图的形式显示。横轴显示的是 5 种基线算法,而纵轴则表示性能指标。线型阴影条形图表示基本策略 X,点型阴影条形图表示 X + CGS 中应用的策略。

图 8 说明了所有任务的平均划分数,显示了任务的进程间聚集程度,值越大意味着任务越分散。所有基本策略中任务都相对切分得较为分散,尤其是 Tetris 方法。通过 CGS 框架的应用,任务按递归方式进行划分,一旦能放置成功就结束划分,最大程

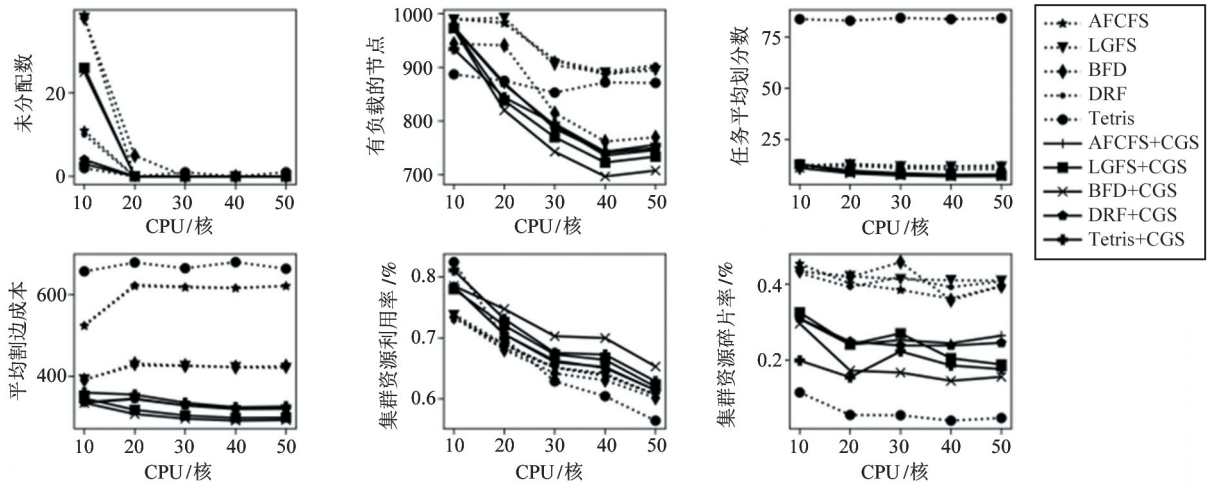


图 7　内存为 300 时不同指标下各方法随 CPU 增长的结果

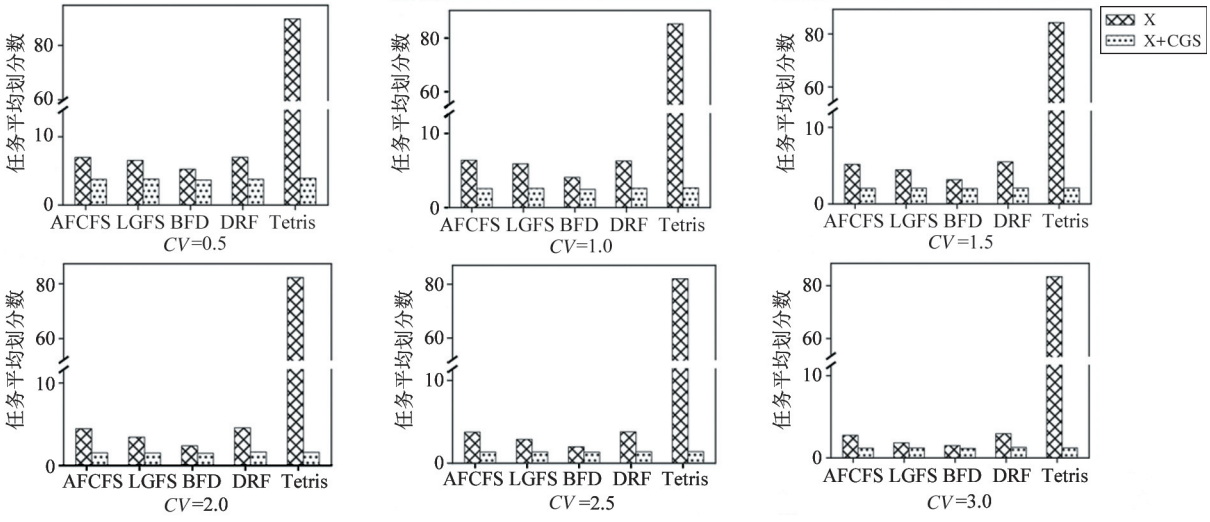


图 8　任务平均划分数

度减少了任务的划分数量,保证了任务进程间的亲和力。

图 9 中,每个子图中的纵轴展示了任务划分的割边成本,代表了通信开销。显然,无论集群环境的异质性程度如何,CGS 均显著降低了跨节点通信开销,至少降低了 37% 的通信开销($CV=0.5$ 环境下的 LGFS 方法中),最大达到了 89% ($CV=3.0$ 环境下的 Tetris 方法中)。

图 10 所示的有负载的节点数证明 CGS 尽可能将任务调度在较少的节点上,使得节点负载更加紧凑。但在 BFD 的调度策略中,任务会被放置在资源最合适的节点上,大幅提升了节点资源的紧凑度。而 CGS 框架为了保证任务的亲和性,会尽量少切分大需求任务,这导致了负载节点数相比基线 BFD 方法近乎相同,但 CGS 是基于基线算法进行的优化,并不会导致其负载节点数大为增加。

图 11 以带误差条的直方图形式,比较了集群中所有节点的的平均值和标准差。结果表明,CGS 不仅提高了利用率(平均提高了 34%),还缩小了标准差,间接证明了集群负载更加均衡。此外采用 CGS 方法可以更好地利用节点上的闲置资源,使得调度结果更加稳定。

图 12 显示了集群中碎片率的平均值。CGS 框架降低了 AFCFS、LGFS、BFD、DRF 方法的集群的平均碎片化程度,至少降低了 19% 的碎片率($CV=2.5$ 环境下的AFCFS方法中),最大达到了81% ($CV=0.5$

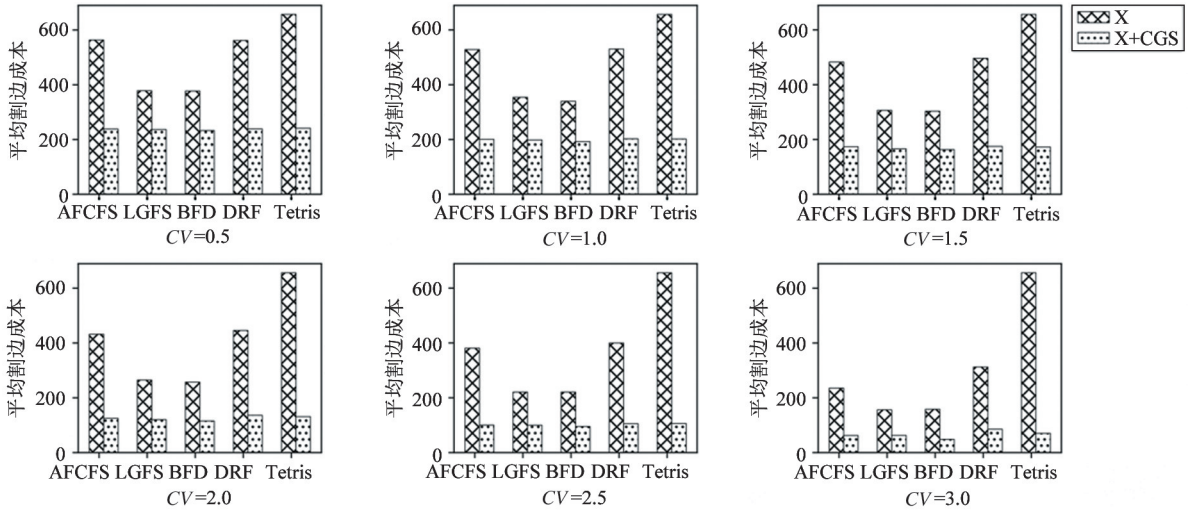


图 9　平均割边成本

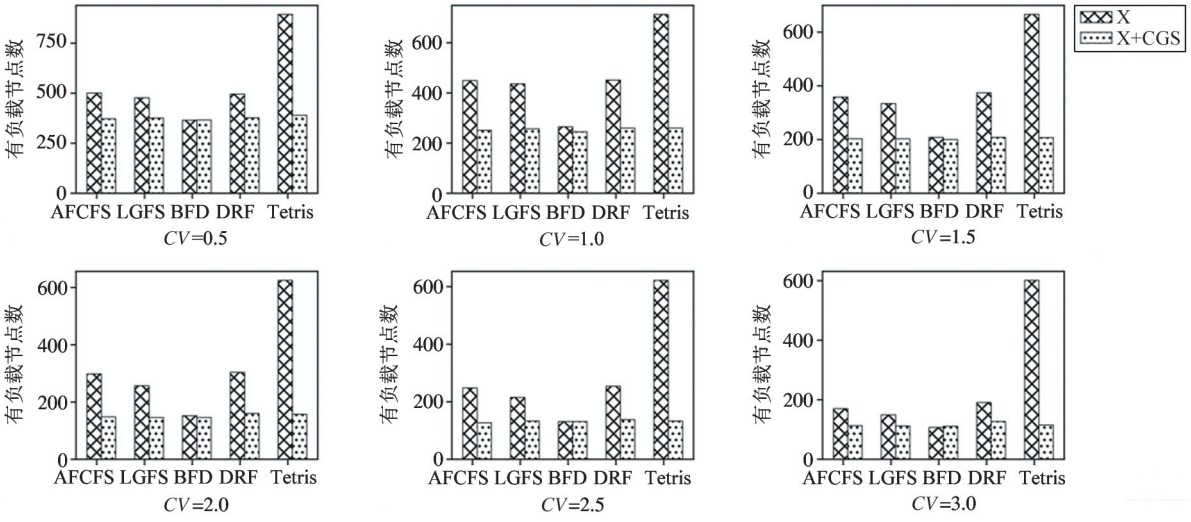


图 10　有负载的节点数

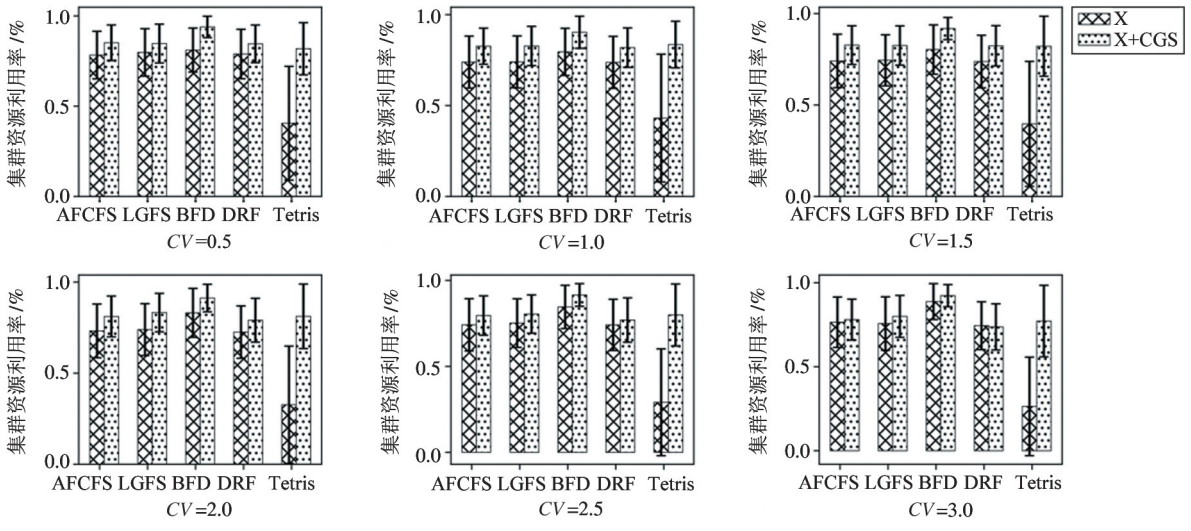


图 11　节点利用率平均值和方差

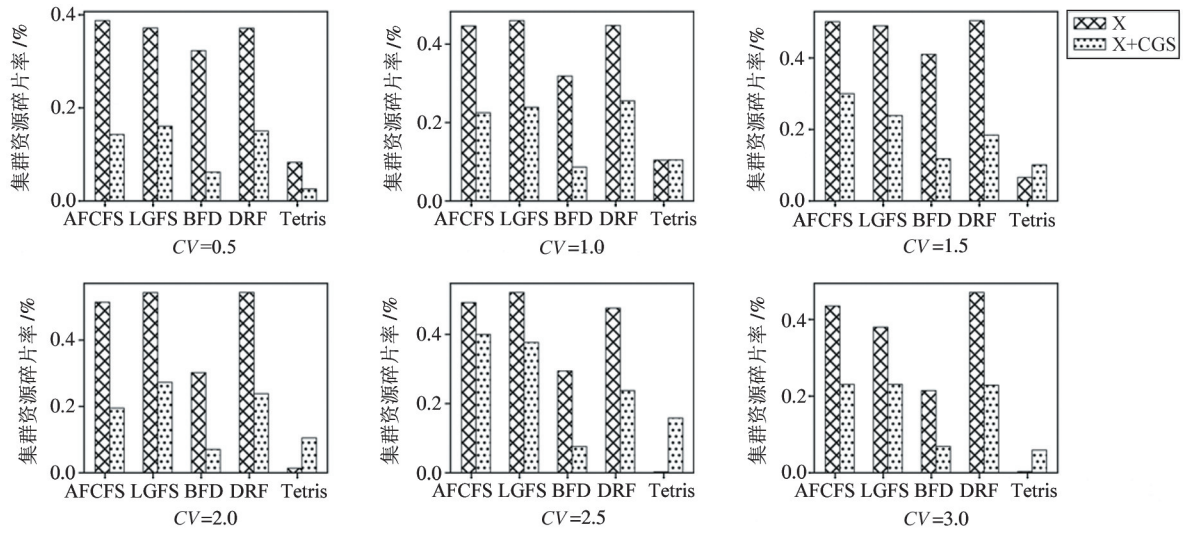


图 12 集群平均碎片率

环境下的 BFD 方法中),这证实了使用 CGS 可以降低集群碎片率,并可提高资源多维度间的平衡性。但在 Tetris 方法中,由于余弦相似度的多维资源匹配策略使进程能被分配在多维资源最合适的节点上,最大化避免了多维资源碎片化。而 CGS 框架优先考虑任务间进程的亲和性,尽可能将任务划分为大组进行分配,会导致一定程度上的资源碎片化。

综合考虑所有指标发现,CGS 方法能够在各种集群资源情况下合理有效调度任务并分配资源。CGS 方法在满足任务资源需求的同时,综合考虑了多维度资源,优化了任务和节点之间的匹配策略,尽可能减少有负载的节点数量,更好地降低了能耗,减少碎片化的同时提高了集群的资源利用率。

6 结 论

本文首先对大规模电网的机电-电磁暂态混合仿真(TS-EMT)任务进行了特性分析,总结了资源利用率规律:短时运行、资源需求较稳定、对通信敏感较高。其次,针对大规模电网机电-电磁混合仿真计算的应用场景,提出了一种用于实时电网仿真的通信敏感组调度框架(CGS),采用了集中式两阶段调度架构,在不中断在线流运行过程中对任务进行主动采样和调度,最大程度保障在线实时任务运行的稳定性。最后,本文提出了一种 CGS 调度算法,基于通信图的图划分策略与基于调度模型的匹配策略相结合,实现了进程组调度,降低了任务跨节点的通信开销。

通过模拟实验证实,CGS 算法会最大程度保证任务进程的聚合性,与基本策略相比至少降低了 37% 的进程间的通信开销,保证了任务执行效率。同时 CGS 将任务调度在较少的节点上,一方面在保证集群不过载的前提下,提高了节点资源利用率,降低了资源碎片率;另一方面降低了集群能耗,符合当今绿色节能计算的主题。此外,未来可以研究更细粒度的调度优化,如使用非均匀内存访问(non uniform memory access, NUMA)架构优化节点内的通信开销;调度算法方面可以尝试群体智能方式,在在线调度允许的范围内进行全局搜索得到较优解。

参考文献

[1] 田芳, 黄彦浩, 史东宇,等. 电力系统仿真分析技术的发展趋势[J]. 中国电机工程学报, 2014, 34(13): 2151-2163

[2] LI F, WANG Y, WU F, et al. Review of real-time simulation of power electronics[J]. *Journal of Modern Power Systems and Clean Energy*, 2020, 8(4):796-808

[3] GUILLAUD X, FARUQUE M O, TENINGE A, et al. Applications of real-time simulation technologies in power and energy systems[J]. *IEEE Power and Energy Technology Systems Journal*, 2015, 2(2):103-115

[4] 张晓华, 刘道伟, 李柏青,等. 智能全景系统概念及其

在现代电网中的应用体系[J]. 中国电机工程学报, 2019, 39(10):2885-2894

[5] 张星, 徐得超, 李亚楼,等. 基于超算的大电网数字并行仿真系统构建及应用[J]. 电网技术, 2019, 43(4):1144-1150

[6] 单茂华, 姚建国, 杨胜春,等. 新一代智能电网调度技术支持系统架构研究[J]. 南方电网技术, 2016, 10(6):1-7

[7] Wikipedia. Gang scheduling[EB/OL]. https://en.wikipedia.org/wiki/Gang_scheduling:Wikipedia,[2021-03-05]

[8] KULKARNI A P, KHANDEWAL M. Survey on hadoop and introduction to YARN[J]. *International Journal of Emerging Technology and Advanced Engineering*, 2014,5(4):82-87

[9] VERMA A, PEDROSA L, KORUPOLU M, et al. Large-scale cluster management at Google with Borg[C]//Proceedings of the 10th European Conference on Computer Systems, Bordeaux, France, 2015: 1-17

[10] HINDMAN B, KONWINSKI A, ZAHARIA M, et al. Mesos: a platform for fine-grained resource sharing in the data center[C]//Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation, Berkeley, USA, 2011: 295-308

[11] OUSTERHOUT K, WENDELL P, ZAHARIA M, et al. Sparrow: distributed, low latency scheduling[C]//Proceedings of the 24th ACM Symposium on Operating Systems Principles, New York, USA, 2013: 69-84

[12] GHODSI A, ZAHARIA M, HINDMAN B, et al. Dominant resource fairness: fair allocation of multiple resource types[C]// Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation, Berkeley, USA, 2011: 323-336

[13] GRANDL R, ANANTHANARAYANAN G, KANDULA S, et al. Multi-resource packing for cluster schedulers[J]. *ACM SIGCOMM Computer Communication Review*, 2014, 44(4): 455-466

[14] 刘洪波, 边娣, 孙黎,等. 交直流混联系统机电—电磁暂态混合仿真研究[J]. 电力系统保护与控制, 2019, 47(17): 39-47

[15] JIAJUE L, CHENGSHUANG C, BAOZHU S, et al. Research on electromechanical-electromagnetic hybrid simulation of UHV AC / DC transmission system[C]// 2019 2nd International Conference on Information Systems and Computer Aided Education, Dalian, China, 2020: 60-63

[16] HAN X W, ZHANG H B. Power system electromagnetic transient and electromechanical transient hybrid simulation based on PSCAD[C]//The 5th International Conference on Electric Utility Deregulation and Restructuring and Power Technologies, Changsha, China, 2016: 210-215

[17] 田芳, 周孝信. 交直流电力系统分割并行电磁暂态数字仿真方法[J]. 中国电机工程学报, 2011, 31(22): 1-7

[18] FARUQUE M D O, STRASSER T, LAUSS G, et al. Real-time simulation technologies for power systems design, testing, and analysis[J]. *IEEE Power and Energy Technology Systems Journal*, 2015, 2(2): 63-73

[19] 徐得超, 张星, 何飞,等. 电力系统机电-电磁混合仿真边界解耦算法研究[J]. 电网技术, 2019, 43(4): 1130-1137

[20] CHRISTENSEN H I, KHAN A, POKUTTA S, et al. Approximation and online algorithms for multidimensional bin packing: a survey[J]. *Computer Science Review*, 2017, 24: 63-79

[21] Wikipedia. Divide-and-conqueralgorithm[EB/OL]. https://en.wikipedia.org/wiki/Divide-and-conquer_algorithm:Wikipedia,[2021-03-05]

[22] KARYPIS G, KUMAR V. Multilevelk-way partitioning scheme for irregular graphs[J]. *Journal of Parallel and Distributed Computing*, 1998, 48(1): 96-129

[23] BULUÇ A, MEYERHENKE H, SAFRO I, et al. Recent advances in graph partitioning[J]. *Algorithm Engineering*, 2016(9220): 117-158

[24] Intel. Intel vtune profiler[EB/OL]. https://software.intel.com/content/www/us/en/develop/tools/oneapi/components/vtune-profiler.html:IntelCorporation,[2021-03-05]

[25] STAVRINIDES G L, KARATZA H D. Scheduling data-intensive workloads in large-scale distributed systems: trends and challenges[J]. *Modeling and Simulation in HPC and Cloud Systems*, 2018(36): 19-43

Communication-aware gang scheduling for online real-time power grid simulations

YUAN Yuxin[*,**], TANG Hongwei[*], ZHAO Xiaofang[*], YAN Jianfeng[***], ZHOU Erzhuan[***]

([*] Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190)

([**] University of Chinese Academy of Sciences, Beijing 100049)

([***] China Electric Power Research Institute, Beijing 100192)

Abstract

Real-time simulations of power grid are modeled as a collection of multi-process parallel tasks running in cooperation. The result of task scheduling and resource allocation is one of the key factors deciding whether a task can achieve its real-time goal. In this paper, a communication-aware gang scheduling framework (CGS) is proposed for the application scenario of large-scale grid electromechanical-electromagnetic hybrid simulation. CGS is based on task profiling and the resource utilization summary. CGS builds a centralized two-stage scheduling architecture to proactively sample and schedule tasks without interrupting online streaming process to predict accurate requirements; it proposes a CGS scheduling algorithm that combines a communication-based graph partitioning strategy with a model-based matching strategy to achieve gang scheduling and reduce the communication overhead of tasks on multi-nodes. Experiments are conducted to test five baseline algorithms against the algorithm with CGS applied in terms of five metrics: the average number of task partition, the average cost of edge-cut, the number of workload bearing nodes, the cluster resource utilization and the resource fragmentation rate. The experimental results conclusively demonstrate that CGS reduces the inter-process communication overhead by at least 37% and reduces the resource fragmentation by 19%, and also improves the cluster resource utilization by 34% on average compared to the baseline algorithms.

Key words: process gang scheduling, online real-time power grid simulation, communication-aware, multi-resource allocation