耦合分布式系统多任务动态调度算法

刘金波,黄海于
(西南交通大学 信息科学与技术学院,四川 成都 611756)

摘 要:针对耦合分布式系统中一个计算模块独自占用某台计算资源,导致其他计算模块无法调度到该计算资源的情况,提出了一种动态任务分配的调度算法。该算法能够根据计算任务的调度要求和计算资源的运行状态,动态进行任务分配,使计算能力强的计算资源能够运行更多的计算模块,从而实现多计算任务调度,使多个符合调度要求的计算任务同时处于运行状态,提高计算资源的利用率,保证了计算任务调度的有效性和高效性。仿真结果表明,该多任务动态调度算法能够在不影响计算速度的情况下,使一台计算资源同时为多个计算任务提供计算能力,大大提高了计算资源的利用率,并且使原本因计算资源的限制而无法运行的计算任务能够提前开始运行,提高了计算任务调度的高效性和灵活性。

关键词:分布式系统;多任务;动态分配;利用率
中图分类号:TP302 **文献标识码**:A **文章编号**:1673-629X(2017)12-0016-04
doi:10.3969/j.issn.1673-629X.2017.12.004

A Dynamic Scheduling Algorithm with Multi-tasks for Distributed Coupled Systems

LIU Jin-bo, HUANG Hai-yu
(School of Information Science and Technology, Southwest Jiaotong University, Chengdu 611756, China)

Abstract: In a coupling-distributed system, one calculation task is distributed in a single computing node so that other tasks cannot be distributed in the same computing node, even with a strong computing power. A new dynamic task assignment algorithm is proposed. It can assign calculation task according to the scheduling requirements of computing task and running state of computing resources, which makes calculation resources with powerful computing capable of running more calculation modules for implementation of multi-tasks scheduling, and which enables computing tasks of meeting scheduling needs in the running state simultaneously, improving the utilization of computational resources and ensure the effectiveness and efficiency of scheduling. The simulation shows that the proposed algorithm can greatly improve the utilization of computing resources, not only ensuring the computing speed but also providing computing power for multiple computational tasks at the same time. The computational tasks that cannot be performed due to conditional constraints can run in time, thus improving the efficiency and flexibility of the scheduling system.

Key words: distributed system; multi-task; dynamic assignment; utilization

0 引 言

高速列车数字化仿真平台采用分布式体系结构[1-2],资源管理及任务调度子系统是平台下的一个子系统。在分布式体系结构中[3-6],任务调度是非常重要的一环。如何有效地利用分布式系统中的计算资源以及如何对现有的计算任务进行调度,均是任务调度子系统必须考虑的问题。分布式系统以及云计算中的调度问题一直都是研究人员研究的热点。为解决分布式系统中的任务调度问题,研究人员提出了很多调度策略,例如排队理论、图论、决策论以及启发式调度方法等[7-11]。

耦合分布式系统原有的调度算法[12]为每个计算模块分配一台计算资源,如果该计算模块只占用很少的计算资源,而其他的计算模块又不能调度到该计算资源上,不仅会造成计算资源的极大浪费,也会使得大量的计算任务处于阻塞状态,无法进行计算。文献[13]提出一种耦合分布式系统多线程任务管理算法,可以将多个计算模块调度到一台计算资源上。但是该算法存在三点不足:一是以多线程的方式管理各个计算模块,多个线程共用一个端口,采用互斥的方式与耦

收稿日期:2016-12-17 **修回日期**:2017-04-20 **网络出版时间**:2017-08-01
基金项目:"十一五"国家科技支撑计划(2009BAG12A01-A01)
作者简介:刘金波(1991-),男,硕士研究生,研究方向为分布式计算、云计算;黄海于,副教授,研究方向为软件无线电、分布式计算、游戏开发。

合器进行数据交互,理论上这种方法比多进程的方式效率要低;二是这种算法只能将同一计算任务的多个计算模块分配给一台计算资源,实质上是一种静态任务调度算法;三是无法根据计算资源的实际使用情况进行任务分配,可能将多个模块分配给一台计算能力不佳的计算资源。考虑到计算模块是以进程的形式存在,一台计算机上可以运行多个进程,一台计算资源不应只服务于一个计算任务,在条件允许的情况下也要为其他计算任务提供计算能力。因此,在不影响计算效率的前提下,可以考虑将多个计算模块调度到一台计算资源上进行计算。

基于此,文中提出了一种动态的任务调度算法。该算法可以根据当前所有计算资源的实际使用情况以及用户提交的计算任务,为计算任务中的每个计算模块寻找最合适的计算资源。

1　系统的基本结构

耦合分布式系统结构如图 1 所示。

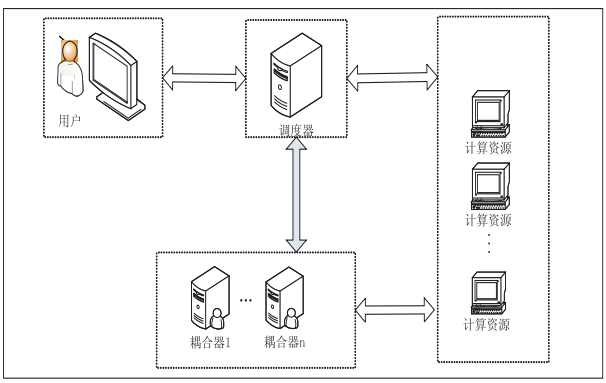

图 1　耦合分布式系统结构

该分布式系统主要包含客户机、作业调度器、耦合器以及计算资源。计算任务由多个计算模块组成,这些计算模块经过调度器进行调度后可以运行在多个计算资源上。

作业调度器负责接收用户提交的计算任务并将其存放在任务队列中,同时作业调度器也会维护一个计算资源链表,通过循环遍历计算任务队列以及计算资源队列来对计算任务进行调度,因此高效的调度算法能够合理地利用现有的计算资源,并且使尽可能多的计算任务同时处于运行状态。

耦合器是整个分布式计算平台的数据交互中心[14]。调度器在完成对某一个计算任务的调度后会指定一个耦合器作为该计算任务的数据交互中心和控制中心,该计算任务的所有数据都会经过耦合器转发到相应的目的模块。分布式计算平台中可以有多个耦合器,每个耦合器也可以同时管理多个工况的耦合计算,调度器根据每个耦合器的性能状态来进行任务的分配。

计算资源与调度器和耦合器处在同一个局域网下,为大规模的计算任务提供计算能力。计算资源上需要运行代理软件,通过代理软件接收调度器分配的计算任务,然后从数据库中下载相应的配置文件与可执行程序,可执行程序包含了计算任务的求解过程,代理启动该模块的可执行程序进行计算。

客户机是用户与分布式计算平台进行交互的接口。用户可以通过客户机进行计算任务的生成以及配置,同时可以将计算任务提交给调度器进行计算,此外用户也可以通过客户机对某个正在运行的计算任务进行过程监控。

2　动态任务调度算法

2.1　算法描述

计算任务是由一系列的子模块构成,调度的过程就是为计算任务的各个子模块寻找计算资源。动态任务分配算法的描述如下:

(1)解析一个未被调度的计算任务;

(2)判断其中固定 IP 的计算模块能否调度成功,不能则返回到(1),解析下一个未被调度的计算任务;

(3)调度独占计算资源的模块,调度失败则返回到(1),解析下一个未被调度的计算任务,如果调度成功,锁定该计算资源,在该计算任务完成计算之前,该计算资源不会参与到以后任务的调度;

(4)调度所有的一般模块,优先将一般模块调度给空闲的计算资源,如果没有空闲的计算资源,则将所有的可用计算资源按照负载评价方法进行排序,选择最合适的计算资源。

2.2　计算资源的负载评价方法

调度器在进行任务的调度时,需要对每台计算资源的计算能力进行评价,计算负载参数 W,以此来判断当前的计算资源能否用于任务调度。对一台计算资源的评价主要通过以下几个参数来进行衡量:CPU 利用率 M;网络流量 S;内存利用率 U;磁盘利用率 D。在判断过程中需要综合衡量各个参数对调度任务的影响,负载参数的计算公式如下:

$$W = X_1 * M + X_2 * S + X_3 * U + X_4 * D \qquad (1)$$

其中,x_i ($x_1 + x_2 + x_3 + x_4 = 1$ 且 $x_i > 0$)分别为 CPU 的利用率、网络流量、内存利用率、磁盘利用率对应的权值。

为这些参数设置不同的权值可以使某个参数影响到最终的调度结果。例如,如果希望将计算任务调度到通信状况良好的计算资源上,可以将 x_2 的权值调大,以此来影响任务的调度。负载值越大,说明该计算节点计算任务越繁重,不应该优先被分配计算任务。

如果某个参数过大,例如 CPU 的利用率已达到 80%,应该将该计算资源暂时剔除计算资源队列。

以上的参数信息由运行在计算资源上的代理软件获取,以固定的时间间隔以心跳信号的形式发送给调度器。调度器为每台计算资源保留最近五十步的心跳信号,每次进行任务调度时,取最近 20 步的心跳信号,计算 CPU 利用率、网络流量、内存利用率、磁盘利用率的均值,然后根据上述公式计算每台计算资源的负载值,选取负载最低的计算资源进行任务分配。为了保证评价的准确性,每当一个计算任务调度成功后,调度器端会清空该计算任务占用的计算模块所对应的心跳信号,以保证数据的准确性。

2.3 调度参数的设置

动态任务分配算法能够将不同计算任务的模块调度到同一台计算资源上,因此调度器需要尽可能多地了解计算任务中每个计算模块的计算要求。例如,某个模块的运行对内存要求很高或者对实时性要求很高,可以将其设置为独占计算资源;具体来说,需要设置的参数有:该计算模块是否独立运行;运行至少需要多大的内存;对操作系统的要求,例如运行于 Windows 或者 Linux 系统下;运行在哪个 IP 地址的计算资源上等。以上这些参数均作为某个模块的调度参数,会影响到调度结果。这些调度参数由用户在构建计算任务时为每个计算模块进行配置。

2.4 动态任务分配算法流程

调度器运行过程中会循环扫描任务队列和计算资源列表,在分配的过程中,如果该计算模块是独占计算资源模块,则需要将该模块占用的计算资源锁定,在该模块未完成计算之前,其他的计算模块无法调度到该计算资源。调度算法的流程如图 2 所示。

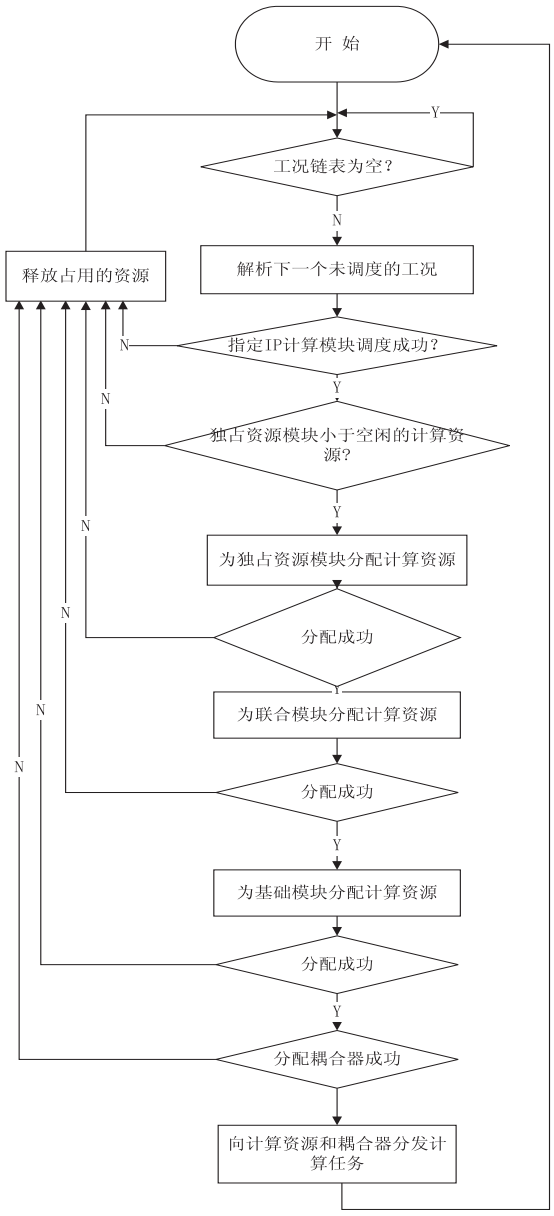

图 2 多任务调度算法流程

每次的调度过程总是优先将计算模块调度给空闲的计算资源,当没有空闲的计算资源时,再尝试将模块分配给已经有计算任务的计算资源。计算资源匹配算法流程如图 3 所示。

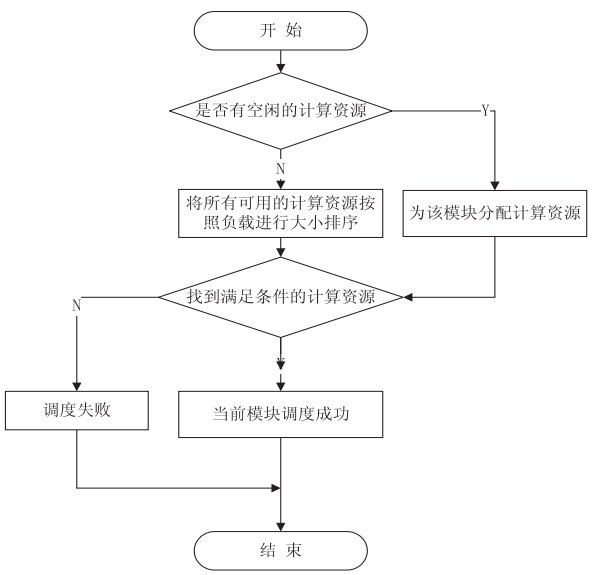

图 3 资源匹配算法

资源匹配算法主要针对非固定 IP 和非独占计算资源的一般模块,通过采用式(1)的资源负载评价方法筛选出合适的资源。算法描述如下:

(1)遍历每台计算资源的心跳信号链表,找到心跳信号步数大于 50 的计算资源,心跳信号小于 50 步表示该计算资源刚刚分配计算任务,不参与此次调度;

(2)取最近的 20 步心跳信号,按照式(1)计算每台计算资源的负载值。在当前的耦合分布式计算平台中,优先考虑内存利用率以及 CPU 利用率的影响。其中内存利用率与 CPU 利用率分别占 0.3 的权值,网络流量与磁盘使用率分别占 0.2 的权值,以此来估算每台计算资源的负载;

(3)找到负载值最小的计算资源,将计算模块分配给该计算资源。

3　实　验

3.1　实验环境

为了验证耦合分布式系统多任务动态调度算法的有效性,构建了如下的实验环境:仿真计算采用八台计算资源,其中一台运行调度器,一台充当耦合器,剩下的六台作为计算资源。所有的计算机全运行在百兆局域网内,同时提交一个分布式人脸识别和车线弓计算任务。其中在人脸识别的计算任务中,三个人脸识别模块是独占并且绑定固定 IP 的计算模块,车线弓的计算任务需要三台计算资源。按照原有的算法,运行这两个计算任务至少需要九台空闲的计算资源才能进行计算,而按照新的算法,理论上只需要六台空闲的计算资源。

3.2　实验结果与分析

首先使用原有的调度算法。由于计算资源数量的限制,对两个计算任务分两次提交,单独测试计算时间,然后再采用动态任务分配算法,同时提交车线弓与分布式人脸识别的计算任务,测试两种计算任务在两种调度算法下的运行时间。表 1 列出了采用两种调度算法完成车线弓计算任务的计算时间。表 2 列出了分布式人脸识别计算任务运行所用的时间。

表 1　运行车线弓用时对比

运行步数	原有算法/s	文中算法/s
1 000	30	30
5 000	157	162
10 000	319	323
15 000	472	481
20 000	632	657

表 2　运行分布式人脸识别用时对比

运行步数	原有算法/s	文中算法/s
500	286	294
1 000	702	712
2 000	1 410	1 424
5 000	2 884	2 909

由表 1 和表 2 可以看出,两种计算任务的运行时间相差不大,但是动态任务调度算法相比原调度算法可以节省三台计算资源,同时能够保证更多的计算任务都处于运行状态。

4　结束语

详细介绍了耦合分布式系统动态任务分配算法的实现。仿真结果显示,该动态任务分配算法能够根据每台计算资源的实际负载情况进行动态的任务调度,能够实现计算资源的合理利用,同时保证计算任务的计算速度不会受到太大影响,从而提高了计算资源的使用效率,同时也使得尽可能多的计算任务同时进行,提高了任务调度的效率。

参考文献:

[1] 张卫华. 高速列车耦合大系统动力学理论与实践[M]. 北京:科学出版社,2013:144-151.

[2] 万春阳,黄海于. 分布式仿真系统的数据监控软件的实现[J]. 计算机技术与发展,2013,23(9):14-17.

[3] D′ Angelo G,Marzolla M. New trends in parallel and distributed simulation:from many-cores to cloud computing[J]. Simulation Modelling Practice and Theory,2014,49:320-335.

[4] 郭奇胜,徐享忠. 计算机仿真[M]. 北京:国防工业出版社,2011:208-243.

[5] Fujimoto R M. Parallel and distributed simulation[C]//Proceedings of the 1999 winter simulation conference. [s. l.]:ACM,1999:122-131.

[6] Fujimoto R M. Distributed simulation system[C]//Proceedings of the 2003 winter simulation conference. [s. l.]:IEEE,2003:124-134.

[7] 杨岩岩. 分布式耦合仿真系统故障的分析与研究[D]. 成都:西南交通大学,2013.

[8] 王　涛,刘大昕. 一种启发式与/或优先约束任务调度算法[J]. 小型微型计算机系统,2007,28(3):504-509.

[9] 王占杰,刘晶晶. 基于多 Agent 的分布式多目标任务调度机制研究[J]. 大连理工大学学报,2011,51(5):755-760.

[10] Hagras T,Janecek J. A high performance,low complexity algorithm for compile-time task scheduling in heterogeneous system[C]//Proceedings of the 18th international parallel and distributed processing symposium. Santa Fe:IEEE,2004:107-115.

[11] Abdllash S,Lesser V. Modeling task allocation using a decision theoretic model[C]//Proceedings of international conference on autonomous agents and multiagent systems. New York:ACM,2005:719-726.

[12] 向　鹏,黄海于. 耦合分布式仿真中任务调度的研究与实现[J]. 计算机技术与发展,2013,23(12):78-81.

[13] 张　宇,黄海于. 耦合分布式系统多线程任务管理算法[J]. 成都大学学报:自然科学版,2012,31(3):251-253.

[14] 杜志强,黄海于. 分布式仿真系统通信故障检测和恢复研究[J]. 计算机技术与发展,2015,25(11):172-176.