新的分布式任务调度算法

New Distributed Task Scheduling Algorithm

周艳慧　张　凯　（国家广播电影电视总局机关服务局 北京 100866）

摘　要: 详细对比了传统 Min – Min 算法的高效特性和 Max – Min 算法的负载平衡特性,结合 Min – Min 和 Max – Min 算法的优点,提出新的具有动态特性的启发式算法(Heuristic task scheduling algorithm based on Min – Min and Max – Min, H – MM),H – MM 解决了 Min – Min 算法负载不平衡问题。实验表明,H – MM 在充分保留 Min – Min 算法执行任务高效基础上实现了算法的动态平衡负载执行特性,得到了更好的任务调度执行效果。

关键词: 分布式 任务调度 算法

1　引言

随着网络技术的发展,很多新的分布式计算模式正在被构建,并用于使用地理分布计算和通讯资源。例如,business – on – demand , Web Service, peer – to – peer 网络,还有网格计算[1]。在分布式计算中 ,任务管理、任务调度和资源管理是系统的三个基本功能。任务调度是决定系统性能优劣的重要组建之一,同时也是一个 NP 完全问题,一直以来成为分布式计算研究领域研究焦点。

传统的分布式任务调度算法有 Min – Min、Max – Min、Suffrage、XSufferage[2] 等。针对 Min – Min 算法具有简单,实用,高效但存在负载不平衡的特性,和 Max – Min 算法负载均衡的特性[3],本文在大量试验基础上,提出了一种新的,动态的基于 Min – Min 和 Max – Min 轮循、循环选择的任务调度策略,即基于 Min – Min, Max – Min 启发式任务调度算法(Heuristic task scheduling algorithm based on Min – Min and Max – Min, H – MM)。通过大量密集的试验可知,H – MM 在保持原有 Min – Min 执行效率高,负载平衡好等特性的同时,运用启发式选择临界值的方法动态解决了 Min – Min 执行的负载不平衡问题。

2　Min – Min、Max – Min 任务调度算法

从文献[2]可知 Min – Min 算法是一种易实现,执行速度快,实现较少时间跨度(Makespan)的算法。但是,在 Min – Min 算法中,每次都是选择小任务映射到执行快的资源上,这种映射会使得更多的任务映射到某一个或几个资源上,从而使得整个分布式系统中可用资源的负载不平衡。而 Max – Min 算法同 Min – Min 相反,它首先调度大的任务,实现一定程度上的平衡负载,减少执行任务的时间跨度。本文将通过算法描述和一个实例来阐述两种算法的执行特点。

数学模型中定义资源集合为 $R = \{r_1, r_2, \ldots, r_n\}$,等待队列任务集合为 $J = \{j_1, j_2, \ldots, j_n\}$。预期执行完成时间(Expected execution completion ,ETC)矩阵。资源 r_i 在没有负载的情况下执行任务 j_i 所需要的时间,记为 ETC_{ij}。所有 ETC_{ij} 值组成 ETC 矩阵。

定义 1 H – MM 判断数组,Min_Time。在 ETC 矩阵中分别取出任务 j_i 在资源 r_i 的最小 ETC_{ij},记为 Min_Time(i) = ETC_{ij},将所有所得值组成一维数组 Min_Time。

根据文献[3] 对 Min – Min 算法的描述,对资源分配过程重新建模,步骤如下:

(1) 判断任务集合 J 是否为空,不为空,执行步骤 2 ;否则跳到步骤 7 ;
(2) 对于 J 中任务 j_i,预测出 j_i 映射到所有可用资源 r_j 上的 ETC_{ij} 值,组成 ETC 矩阵 ;
(3) 得到 ETC 矩阵的一维数组 Min_Time ;
(4) 根据 3 的结果,在 Min_Time 中找出最小 ETC_{ij} 值,以及对应的任务 j_i 和资源 r_j ;
(5) 将任务 j_i 映射到机器 r_j 上,并将该任务从任务集合 J 中删除 ;
(6) 更新其它任务在机器 r_j 上的 ETC_{ij} 值,回到步骤 1 ;
(7) 此次映射事件结束,退出程序。

以上是针对 Min – Min 算法的描述,对于 Max – Min 算法,只需在第 4 步中找出最大 ETC_{ij} 值,以及对应的任务 j_i 和资源 r_j 即可。表 1 是一个 ETC 矩阵实例,图 1 为两算法得到的 makespan 分布。从图 1 分析,Max – Min 算法在执行过程中将三个任务均匀分配到三个资源上,负载均衡,最终用了较少的执行时间。而 Min – Min 算法将所有任务分配到一个资源上,负载不均衡,导致执行时间远远高于 Max – Min 算法的执行结果。

表 1　ETC matrix

	r_1	r_2	r_3
j_1	2	4	8
j_2	4	8	16
j_3	8	16	32

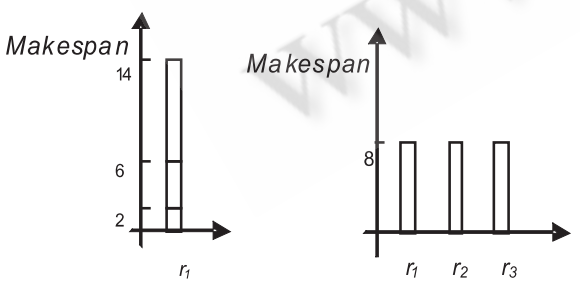

图 1　Max – min　　Min – min

3　H – MM 算法

通过大量经验可知在调度任务大小非均一的计算系统中,Max – Min 算法的调度性能优于 Min – Min 算法,而在任务大小较均一的计算系统,Min – Min 算法的调度性能优于 Max – Min 算法。本文利用计算 ETC 矩阵中 Min_Time 数组的相对标准偏差值判断任务的均一性。

定义 2 相对标准偏差(Relative Standard Deviation,RSD)。$RSD = \frac{s}{\bar{x}}$,其中 $s = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}}$。s 是样本标准偏差,表示样本参数的离散程度,是样本均值,相对标准偏差能较好地说明一组数据的分散程度。

根据以往工作[3]阐述,传统 Min – Min 算法在调度任务大小非均一时的负载均衡、调度效率远不如 Max – Min 算法。在一些分布式场景中,任务为长期任务,符合分布式中任务复杂,大小非均一的特征。H – MM 算法的基本思想是 :

根据系统的性能预测得到 ETC 矩阵,提取 ETC 矩阵中元素得到 Min_Time 数组,计算出 Min_Time 数组的相对标准偏差值,记为 ξ。H – MM 算法利用相对标准偏差值说明数据分散程度的特性,判断调度任务大小的均一性。如果 ξ 值小于文中通过每次循环确定的相对标准偏差临界值 ξ′,说明 Min_Time 数组中数值大小的波动比较小,即调度任务大小较均一,调度系统选择使用 Min – Min 算法 ;反之,则说明 Min_Time 数组中数值大小的波动比较大,调度任务大小非均一,调度系统选择使用 Max – Min 算法。最后,通过循环、轮循选择,可以得到较 Min – Min, Max – Min 更优的时间跨度。其中,ξ通过每次在循环选择时,在[0.1,1]之间,每隔 0.05 取值,比较取不同值时的执行效果,获得执行效果最优的 ξ即为相对标准偏差临界值,H – MM 算法步骤 :

(1) 判断任务集合 J 是否为空,不为空,执行步骤 2 ;否则跳到步骤 8 ;
(2) 对于 J 中任务 j_i,预测得到 j_i 映射到所有可用资源 r_j 上的 ETC_{ij},组成 ETC 矩阵 ;
(3) 得到 ETC 矩阵的一维数组 Min_Time ;
(4) 根据步骤 3 得到的结果,计算 ξ = Min_Time 的相对标准偏差(RSD) ;
(5) 选择 ξ 的临界值方法 :
ξ 在[0.1,1] 之间,每隔 0.05 取值,比较取不同值时的执行效果,获得执行效果最优的 ξ即为相对标准偏差临界值 ;
(6) 当 ξ 值大于 ξ 时,使用 Max – Min 算法 ;当 ξ 值小于 ξ时,使用 Min – Min 算法 ;
(7) 回到步骤 1 ;
(8) 此次映射事件结束,退出程序。

H – MM 算法能在两种传统算法的基础上,最大限度地发挥 Min – Min 算法的快速执行和 Max – Min 算法的负载均衡特性。而文献[4] 中算法在没有 QoS 需求时,其性能是与 Min – Min 算法等价的,而 H – MM 算法在任何场景下,只要选择好合适的临界值,就会得到比 Min – Min 算法较优的结果。

4　仿真实验

本文首先用实例阐述 H – MM 如何实现动态获得 ξ′。设定 15 个任务和 5 个服务资源的场景,在这一个场景下 ξ′从[0.1,1] 间每隔 0.05 选一个值,获得不同的执行结果,如图 2。从图 2 中可以知道当 ξ取 0.7 时,H – MM 获得的少于 Min – Min 和 Max – Min 时间跨度值比在取其它值时要大,所以我们就取 0.7 为这个场景下的相对标准偏差临界值。H – MM 算法在每次循环调度中用这种方法获得 ξ′。

表 2　三个算法的 Makespan 性能比较

场景	H – MM	Min – Min		Max – Min	
	Makespan	Makespan	Improvement	Makespan	Improvement
1	130.8079	171.2853	23.63%	160.0749	18.28%
2	67.1319	90.7532	26.03%	90.7532	26.03%
3	287.6115	460.7481	37.58%	327.462	12.17%
4	325.07	411.2768	20.96%	473.661	31.37%

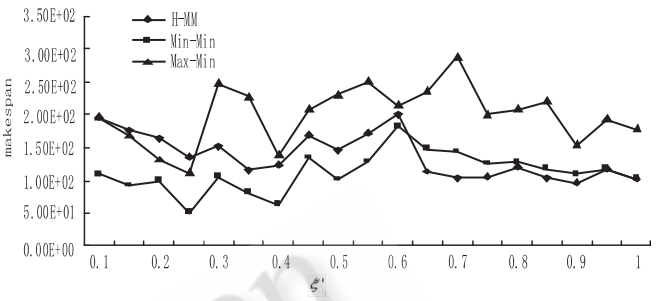

图 2　不同 J/R 比值的分布

在阐明获得最优 ξ′方法后,我们要比较 H – MM, Min – Min 和 Max – Min 算法在 4 中场景下的执行性能。场景 1,5 个任务和 5 个资源 ;场景 2,15 个任务和 5 个资源 ;场景 3,20 个任务和 20 个资源 ;场景 4,20 个任务和 50 个资源。

由表格 2 可知,场景(1) 中 H – MM 执行速度比 Min – Min 和 Max – Min 分别提高了 23.6% 和 18.28% ;场景(2) 中分别提高了 26.03% 和 26.03% ;场景(3) 中分别提高了 37.58% 和 12.17% ;场景(4) 中分别提高了 20.96% 和 31.37% 。对于 4 个场景,H – MM 的执行效果要比 Min – Min 和 Max – Min 好很多。通过充分结合 Min – Min 的高效执行性和 Max – Min 的负载平衡特性,H – MM 可以获得比这两种算法更好的执行特性。所以,只要选择到合适的 ξ′,H – MM 算法可以在任何情况下获得比 Min – Min 和 Max – Min 都好的执行效果。

5　结束语

Min – Min 算法以其高效的执行能力,简单的执行步骤使其在很多调度系统中得到了应用,然而某些情况下 Min – Min 算法不能够负载平衡,本文提出了一个新的动态的启发式调度算法 H – MM 来解决 Min – Min 负载不平衡问题。H – MM 不但充分实现了 Min – Min 的高效执行特性,还结合了 Max – Min 的负载平衡特性,动态的循环选择使用两种算法得优点。H – MM 任务调度策略还存在一些局限性。例如调度策略还没有考虑多维 QoS 约束的情况,同时在调度失败后的数据迁移问题还需要深入的研究。下一步工作将从以上问题着手,完善 H – MM 算法。(下转第 80 页)

的依赖注入特性。而 service 层的 StudentServiceImpl 类依赖于 DAO 层的 StudentManage,类,同理,这种关系也是 spring 容器来处理(通过配置文件设置)。而泛型 HibernateEntityDAO 即处理数据库操作的各个细节。

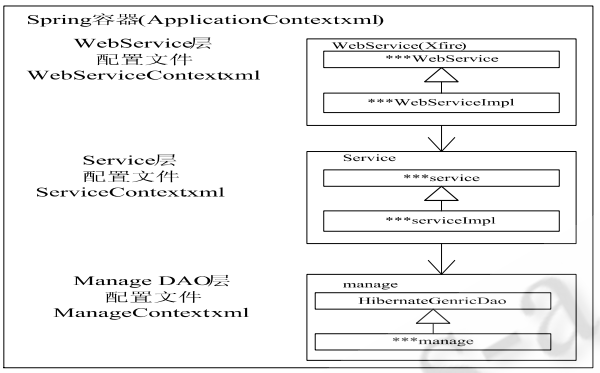

图 4　分层实现技术详细说明

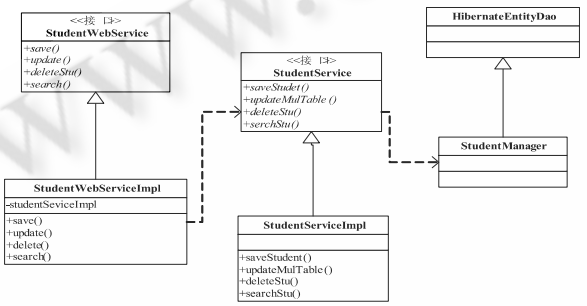

图 5　分层实现的类图说明

以下是客户发送请求得到服务端数据的系统顺序图。

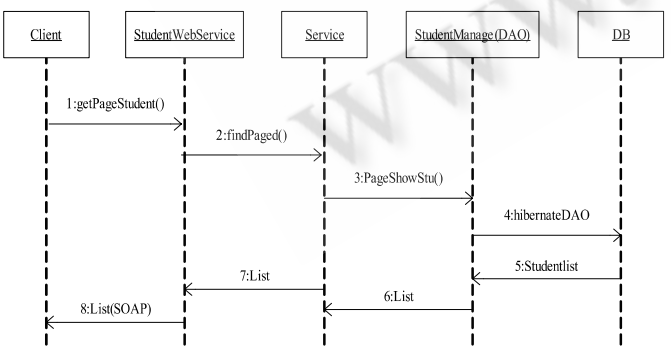

图 6　客户发送请求得到服务端数据的系统顺序

由于系统的这种分层设计获得了各层的独立变化和很好的复用。

参考文献

1　戴侃,杨小虎. 基于 J2EE 和 FLEX 技术构建 RIA 系统的探索与实现. 微电子学与计算机,2006,23(5):22 – 24.

2　吕小红,杨小虎. 基于 Flex/Struts + EJB 技术的报表发布框架. 计算机应用与软件,2007,24(9):81 – 82.

3　Adobe Systems Inc. Flex 2 Developer' s Guide. www. adobe. com,2007.

4　Adobe Systems Inc. Flex 2 Developer' s Guide . www. adobe. com,2007.

5　Harrop R,Machacek J. Spring 专业开发指南. 北京:电子工业出版社,2006. 30 – 79.

(上接第 42 页)

参考文献

1　Foster I, Kesselman C, M. Nick J, et al. The physiology of the grid: An Open Grid Services Architecture for Distributed Systems Intergration. http://www. globus. org/alliance/publications/papers/ogsa. pdf. 2002 – 6 – 12.

2　Braun T D, Siegel H J, Beck N. A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems. Parallel and Distributed Computing. 2001,61(6): 810 – 837.

3　Hou Y, Yu J and Turgun, NDA – MM: A New Adaptive Task Scheduling Algorithm Based on the Non – dedicated Constraint Grid, Sixth International Conference on Grid and Cooperative Computing, 2007,8: 275 – 281.

4　He XiaoShan, Sun XianHe, Von Laszewski Gregor. QoS Guided Min – Min Heuristic for Grid Task Scheduling. Journal of Computer Science and Technology, 2003,18(4):442 – 451.