计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2018,27(6):129−133 [doi: 10.15888/j.cnki.csa.006379]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于 CPU 因子影响的二次调度算法①

王志佳[1,2], 李伦波[1], 王嘉春[2]

[1](南京理工大学 计算机科学与工程学院, 南京 210094)
[2](北京空间飞行器总体设计部 信息中心, 北京 100094)
通讯作者: 王志佳, E-mail: wzj_newlife@163.com

摘　要: 在航天器型号设计阶段需要利用高性能计算系统开展大量的仿真分析工作, 昂贵的许可证资源使用极其紧张, 作业计算效率低. 针对高性能计算系统中现有作业派发机制未动态考虑高性能运算主机空闲状态的缺陷和不足开展研究, 基于资源调度软件 Platform LSF, 结合航天器仿真分析特点, 提出一种新的思路, 设计并实现一种新的基于 CPU 因子 (CPU Factor) 影响的二次调度算法, CPU 因子用于区分不同机器的相对运行速度, 仿真结果表明算法能够有效提升作业计算效率, 缩短许可证资源占用时间. 实际案例说明算法具备推广应用的可能, 一定程度的提高了许可证资源利用率, 满足了航天器仿真分析过程中对于成本控制和资源精益化利用的实际需求.

关键词: 高性能计算; 调度算法; 资源利用率; 仿真分析

引用格式: 王志佳,李伦波,王嘉春.基于 CPU 因子影响的二次调度算法.计算机系统应用,2018,27(6):129–133. http://www.c-s-a.org.cn/1003-3254/6379.html

Rescheduling Optimal Algorithm Based on CPU Factor

WANG Zhi-Jia[1,2], LI Lun-Bo[1], WANG Jia-Chun[2]

[1](College of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China)
[2](Information Center, Beijing Institute of Spacecraft System Engineering, Beijing 100094, China)

Abstract: In the stage of spacecraft model design, the use of expensive licensed resources is extremely tight, the calculation efficiency is low. This study analyzed the shortcomings of the existing job dispatching mechanism in the high performance computing system that without considering the idle state of the high performance host. Based on the platform LSF, combined with the characteristics of spacecraft simulation analysis, a new resolution, i.e., rescheduling algorithm based on high CPU factor, which is used to distinguish the relative running speed of different machines, priority has been designed and implemented. The simulation results show that the algorithm can effectively improve the efficiency of the operation and shorten the occupancy time of licensed resources. The practical case shows that the algorithm has the possibility of popularization and application, which can improve the utilization of licensed resources and meet the needs of cost control for spacecraft simulation analysis and the efficacial utilization of resources.

Key words: HPC; scheduling algorithm; resource utilization; simulation analysis

航天器型号研制是一个涉及众多新学科、新技术, 具有耗资大、风险高、周期长、技术复杂的系统工程. 型号设计阶段需要考虑的影响因素较多, 在卫星总体、分系统和单机产品之间存在着大量的协调和互动关系, 在输入与输出之间、参数与系统功能之间也涉及大量的迭代关系, 通过 CAE 软件的仿真分析, 极大提高了产品设计周期, 增强了产品可靠性, 涵盖了有效载荷、结构、热控、能源、姿态和轨道控制等多个层

① 基金项目: “十二五”国防基础科研项目 (C0320110002)
收稿时间: 2017-10-02; 修改时间: 2017-10-24; 采用时间: 2017-10-31; csa 在线出版时间: 2018-05-28

次和多个环节[1].

随着计算技术与网络技术的高速发展和广泛应用. 基于高性能计算系统的仿真计算技术, 在提高产品的设计质量, 降低研究开发成本, 缩短开发周期方面都发挥了重要作用, 成为了实现产品创新的支撑. 然而, 仿真计算资源成本的投入逐渐成为企业成本投入的重中之重, 其仅次于人力资源成本投入. 虽然购置了高性能的计算设备, 但却无法保障设备的充分利用, 尤其是一些国外工程软件,如 ANSYS Fluent、MSC Nastran、LS-DYNA 等, 仅一个并行求解器许可就需要近百万的花费, 使得企业竞争能力明显不足. 本文以某航天器制造企业为项目背景, 基于高性能计算系统的资源调度软件 Platform LSF(Load Sharing Facility), 研究面向航天器仿真分析的资源调度策略优化问题, 从而提高计算资源利用率以及企业仿真能力.

1 算法优化的需求分析

在航天器研制领域中, 仿真学科主要涉及力学分析、热分析、电磁分析、碰撞分析、电路分析、流体力学分析等, 在用的 CAE 软件多为国外产品, 多为面向制造行业进行销售, 版权许可控制极为严格, 价格制定非常昂贵, 且按照许可证数量来限制终端用户的使用权限, 如仅购买了一个许可, 那么某个时间仅能一个作业进行计算, 直到该作业计算完毕, 释放许可证资源后, 才能计算其他作业.

许可资源已经成为制造企业成本投入的重点. 以 ANSYS FLUENT 为例, 一个支持 16 核并行的 Fluent 求解器, 价格高达 320 万人民币, 针对大型制造企业, 以支持 50 位型号设计师同时开展流体计算来进行估计, 仅流体计算软件就需要上亿的资金来购买软件许可, 使得企业的生产成本过于沉重. 同时, 随着高性能计算技术的不断发展, GPU 技术、众核技术[2]等新兴技术不断推出, 大大提高了软件计算性能, 为支持新的技术, 需要单独购买软件的扩展许可, 对现有许可进行升级, 通过市场调研, 后期软件及许可证资源的升级维护, 费用占据产品 list 价格的 15%~30%, 又将投入大量资金. 因此, 对于资源调度策略的进一步优化和定制, 是提高许可资源利用率, 使得昂贵的许可证资源能够充分发挥效益的必要条件, 也是企业追求精益化生产的必经之路.

航天器制造行业高性能计算既要满足多人多任务小作业的计算需求, 也要适应更高密度网格的大作业计算需求, 尤其是随着后续对模型精度要求的提高、对网格划分粒度的加大, 同时开展作业人数的增多, 昂贵的 CAE 许可证资源竞争激烈, 对于如何优化作业计算效率成为难点问题, 而现有资源调度算法多数以用户公平或用户权益角度出发, 未针对计算效率进行优化, 导致昂贵的许可证资源使用极其紧张, 许可证及计算资源长时间被占用, 作业计算效率较低, 需要针对以上的行业特点来设计和定制一个满足需求的调度算法.

作业调度算法是计算集群的大脑, 最为常用的调度算法就是先来先服务算法, 即计算顺序严格按照作业提交顺序进行, 不考虑作业的紧急程度、作业大小、优先级等一系列因素. 随着应用的不断深入和集群技术的不断发展, 作业调度算法被不断改进, 一些高级的调度算法不断被提出. 例如, 国内外较为流行的 LSF、PBS 等作业调度管理系统中常用的抢占式调度、公平式调度、独占式调度、预约调度、回填调度、节能调度[3]等. 同时, 不断有人基于现有调度算法进行优化, 进一步提升算法性能. 任小西等人[4]优化了抢占式调度算法, 实现了基于动态抢占阈值的 LSF 调度算法, 减小了系统开销, 进一步提高了资源利用率; 李荣胜等人[5]基于回填调度提出一种基于价值密度的调度算法, 提高了资源利用率. 为进一步优化资源占用时间, 沈萍萍等人[6]提出对已运行计算作业重新调度的思想,但未考虑作业规模的大小, 对于航天器仿真分析这种需要同时计算较多大作业和短作业的情况, 在主机性能差异较大时, 显然让高性能主机计算大型作业能够节省更多计算时间和资源占用时间.

综上, 根据应用的实际, 当前并无完全满足应用需求的资源调度算法. 基于该情况, 本文提出一种新的思路, 设计并实现一种新的基于 CPU Factor 影响的二次调度算法, 完善了现有作业调度算法未动态考虑高性能运算主机空闲状态、未考虑作业规模大小的缺陷和不足, 进一步提升了资源利用率.

2 算法思想

算法基于“能者多劳”的思维进行设计, 实现将更多的大作业交由高性能的主机进行处理, 提升整体的作业处理能力, 缩短许可证占用时间, 避免因主机性能差异, 造成高性能主机空闲, 导致资源浪费. 同时, 考虑

到作业调度的消耗, 算法仅针对大作业开展二次调度, 避免因频繁调度而导致的调度效率的降低.

资源调度系统在派发作业时, 总是将作业调度到最佳的运算主机, 以便作业的运算时间最小. 资源调度系统在每次尝试调度作业时, 总需要检测主机的被派发资格, 需要参考一系列因素, 其中主机负载水平 (CPU Load Average)[7]和主机性能因子 (CPU Factor) 最为关键[8].

在面向航天器仿真分析的环境中, CAE 计算作业多为计算密集型任务[9], 对于 CPU 的性能要求较高, 需要频繁进行计算. 考虑到计算效率, 已在集群配置文件 lsb.hosts 中限制了各主机运行的计算作业数量不能超过 CPU 核数, 即主机负载水平小于等于 1, 因此在 LSF 进行作业调度时, CPU 因子成为主要的评估指标. LSF 会将作业派发至当前情况下空闲且性能最好的主机上 (作业数量小于 CPU 个数且当前 CPU 因子最高的主机), 当这台主机的作业数量达到 CPU 个数时, 才考虑另外一台空闲且性能最好主机.

在航天器仿真制造过程中, 越来越多新研型号需要通过大型模拟仿真实验进行验证, 逐步取消以往用于实验验证的航天器, 取而代之的是通过大型结构、热仿真分析实验等来支撑. 在这种应用情境下, LSF 总是将作业派发至当前作业数量小于 CPU 个数且 CPU 因子最高的主机, 并且除了出错或某些意外的情况, 作业不会在被调度到其他主机[10,11]. 但每个作业的计算时间几乎均不相同, 随着时间的推演, 运算主机间的空闲情况将不断变化, 此时, 即使有空闲且性能更高的主机, 作业也无法迁移, 必然造成计算资源的浪费. 尤其是对于大型计算任务, 高性能运算主机与低性能运算主机间的差别体现更大. 例如, 对近 1400 万自由度的航天器部件模型 (500 Hz 内 2500 阶模态), 采用全模型标准的模态频响应分析 SOL111 进行求解, 低 CPU 因子主机求解时间长达 26 小时. 如果采用基于高 CPU 因子主机优先的二次调度算法, 将大作业 (根据工程实践, 将运行时间超过 4 小时的计算作业定义为大作业) 重新迁移至高 CPU 因子的主机上运行, 求解时间可以缩减至 10 小时, 将大幅加快作业运行效率, 减少对于昂贵的许可证资源的占用, 提高系统资源的利用率[12,13].

基于上述思想, 考虑对调度算法进行优化, 实现一种基于 CPU 因子影响的二次调度算法.

3 基于 CPU 因子影响的二次调度算法

算法的执行过程包括 5 个步骤, 流程如图 1 所示.

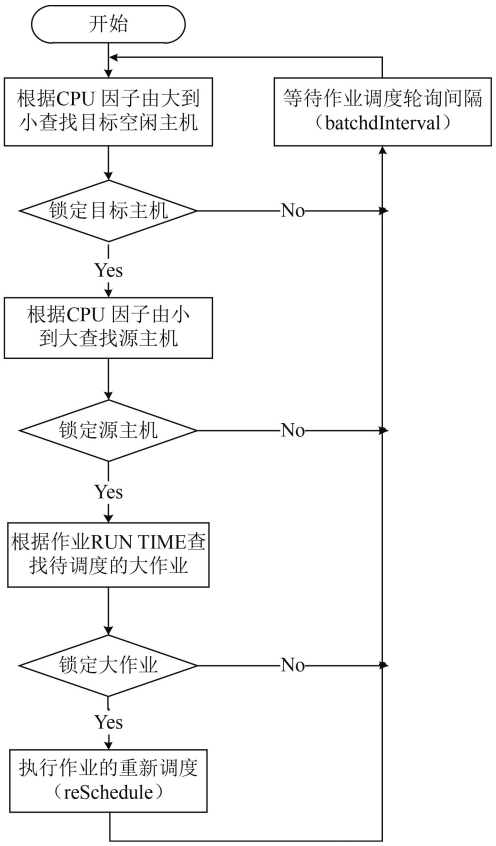

图 1 基于 CPU 因子影响的二次调度算法调度流程

(1) 根据 CPU 因子值由大到小依次对比各主机当前运行作业数与允许运行的最大作业数, 如果前者等于后者, 说明主机已满载, 不能作为目标主机, 当定位到第一个前者小于后者的主机, 可以确定为空闲目标主机; 如全部主机的运行作业数等于最大作业数, 则表示当前主机均已满载, 无法进行二次优化调度.

(2) 反向根据 CPU 因子的值由小到大依次对比各主机当前运行作业数与允许运行的最大作业数, 当第一次定位到有主机当前运行作业数不为零时, 可以确定为源主机, 如果所有主机的运行作业数均为零, 则表示当前没有作业需要进行二次调度.

(3) 遍历源主机中的所有作业, 对比作业运行时间, 当第一次定位到源主机上存在作业运行时间超过 240 分钟时 (根据工程实践, 作业运行时间超过 240 分钟被定义为大型仿真作业), 可以确定为二次调度的目标作业, 锁定作业 ID.

(4) 通过变量传递待转移作业的 ID, 设置候选主机

数量为 1, 传递候选主机名称, 最终通过调用 LSF API[14,15]函数 *lsb_mig*(*&mig*, *&badHostIdx*), 实现将目标作业迁移至目标主机, 即实现作业的二次调度.

(5) 经过一个作业调度轮询间隔后, 重复上述过程, 持续寻找需要调度的作业.

4 算法的仿真实验与结果分析

在实际计算环境中, 选取一台双路 6 核的高 CPU 因子主机 Host_high 和一台双路 6 核的低 CPU 因子主机 Host_low, 通过提交相同数量的同一作业, 测试二次优化调度前后的运算时间差别. 以下时间单位均为分钟.

4.1 测试单个作业的基础运行时间

单个测试作业在两台主机上的运行时间参数如表 1 所示. 从表中可以看出, 对于短作业 Job1, 在高配置主机和低配置主机上的运行时间差不多; 对于大作业 Job2 和 Job3, 在低配主机和高配主机上的运行时间差别较大.

表 1 单个测试作业运行时间差异 (单位: 分钟)

作业名称	Host_high 主机运行时间	Host_low 主机运行时间
Job1	51	58
Job2	226	372
Job3	541	814

4.2 测试算法开启前后的作业总运行时间差异

测试一: 验证对于运行时间小于 240 分钟的短作业算法是否生效. 将短作业 Job1 复制 8 份, 同时提交给 high 主机和 low 主机, 对比开启二次调度与否对作业 Job1 运行时间的影响, 以及作业总计算时间的差异, 结果如表 2 所示. 从表中可以看出, 二次调度开启前后作业的总运算时间差异很小, 说明为减小二次调度时间上的损耗, 对于短作业二次调度算法不生效.

表 2 Job1 作业运行时间差异 (单位: 分钟)

不使用二次调度			使用二次调度		
作业名称	运行主机	运行时间	作业名称	运行主机	运行时间
Job1_1	Host_high	51	Job1_1	Host_high	51
Job1_2	Host_high	53	Job1_2	Host_high	52
Job1_3	Host_high	51	Job1_3	Host_high	52
Job1_4	Host_high	50	Job1_4	Host_high	50
Job1_5	Host_low	57	Job1_5	Host_low	57
Job1_6	Host_low	56	Job1_6	Host_low	56
Job1_7	Host_low	59	Job1_7	Host_low	58
Job1_8	Host_low	56	Job1_8	Host_low	56
总计		433			432

测试二: 验证对于运行时间大于 240 分钟的大作业算法是否生效. 将 Job2 复制 8 份, 同时提交给 high 主机和 low 主机, 对比开启二次调度与否对作业 Job2 运行时间的影响, 以及作业总计算时间的差异, 结果如表 3 所示. 从表 3 中可以看出, 对于低配主机上的运行时间超过 240 分钟的大作业被重新调度到了高配主机后, 总运算时间节省了 300 分钟.

表 3 Job2 作业运行时间差异 (单位: 分钟)

不使用二次调度			使用二次调度		
作业名称	运行主机	运行时间	作业名称	运行主机	运行时间
Job2_1	Host_high	219	Job2_1	Host_high	220
Job2_2	Host_high	225	Job2_2	Host_high	230
Job2_3	Host_high	226	Job2_3	Host_high	226
Job2_4	Host_high	229	Job2_4	Host_high	227
Job2_5	Host_low	368	Job2_5	Host_low+ Host_high	294
Job2_6	Host_low	375	Job2_6	Host_low+ Host_high	301
Job2_7	Host_low	370	Job2_7	Host_low+ Host_high	291
Job2_8	Host_low	372	Job2_8	Host_low+ Host_high	295
总计		2384			2084

测试三: 验证运算时间的缩短程度与作业规模的关系. 将 Job3 复制 8 份, 同时提交给 high 主机和 low 主机, 对比开启二次调度与否对作业 Job3 运行时间的影响, 以及作业总计算时间的差异, 结果如表 4 所示. 从表中可以看出, 对于低配主机上的运行时间远超过 240 分钟的大作业被重新调度到了高配主机后, 总运算时间节省了 827 分钟.

表 4 Job3 作业运行时间差异 (单位: 分钟)

不使用二次调度			使用二次调度		
作业名称	运行主机	运行时间	作业名称	运行主机	运行时间
Job3_1	Host_high	535	Job3_1	Host_high	536
Job3_2	Host_high	540	Job3_2	Host_high	539
Job3_3	Host_high	541	Job3_3	Host_high	545
Job3_4	Host_high	543	Job3_4	Host_high	541
Job3_5	Host_low	818	Job3_5	Host_low+ Host_high	603
Job3_6	Host_low	814	Job3_6	Host_low+ Host_high	611
Job3_7	Host_low	805	Job3_7	Host_low+ Host_high	605
Job3_8	Host_low	819	Job3_8	Host_low+ Host_high	608
总计		5415			4588

从三次测试数据得出如下结论:

(1) 为保证调度效率, 减少因频繁调度导致的调度

效率损耗, 对于作业运行时间远小于 4 小时的短计算作业, 二次调度不发挥作用.

(2) 对于作业运行时间为 4 小时左右的计算作业, 通过大作业二次优化调度可以降低总运行时间约 12.6%.

(3) 对于作业运行时间为 12 小时左右的计算作业, 通过大作业二次优化调度可以降低总运行时间约 15.27%.

如图 2 所示, 从图中可以看出随着作业规模增大, 作业总运算时间降低越多.

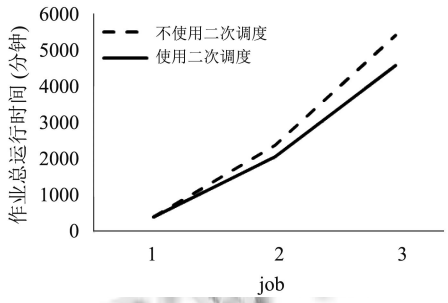

图 2 作业总运算时间与二次调度算法的关系

由此可以推断, 基于 CPU 因子影响的二次调度算法适用于 CAE 仿真等计算密集型任务, 一定程度的缩短了大作业的运行时间, 且缩短的程度与作业的规模呈正比.

5 结束语

目前该算法已应用于航天器型号研制仿真分析实践, 例如通信、遥感、导航等领域, 不同程度的缩短了型号仿真分析周期. 以导航领域某型号卫星整星力学分析为例: 模型划分为 500 万网格, 应用当前计算环境, 单个作业计算时间约需要 12 小时, 同时需开展 10 个方案进行对比论证, 还需针对整星开展模态分析, 采用现有先来先服务算法, 一颗卫星的整体力学分析至少需要 5 天时间. 采用基于 CPU 因子影响的二次调度算法后, 调度系统充分调度高性能主机运算能力, 动态的将作业二次调度值高性能主机, 使得仿真计算时间由 5 天缩减为 3.5 天, 有效的减少了许可证资源总占用时间, 提高了资源利用率.

本文提出一种基于 CPU 因子影响的二次调度算法, 实现将正在低性能运算主机上运行的大作业迁移至高性能主机上继续运行, 从而加快大作业计算效率, 缩短作业运行时间, 减少对许可证资源的占用时间, 提升了许可证资源的利用率, 间接的减少了企业成本投入, 具备一定的推广意义.

参考文献

1 陈月根. 航天器数字化设计基础. 北京: 中国科学技术出版社, 2010. 2–3.
2 周小宇, 雒江涛, 罗林, 等. 众核与 Spark 结合的高速流量监测系统. 计算机系统应用, 2017, 26(6): 112–117.
3 Van Den Dooren D, Sys T, Toffolo TAM, *et al*. Multi-machine energy-aware scheduling. EURO Journal on Computational Optimization, 2017, 5(1–2): 285–307. [doi: 10.1007/s13675-016-0072-0]
4 任小西, 赵公怡. 基于动态抢占阈值的 LSF 调度算法. 计算机工程, 2012, 38(4): 275–277, 280.
5 李荣胜, 赵文峰, 徐惠民. 价值密度—截止期—回填的网格作业调度算法. 计算机应用, 2010, 30(10): 2771–2773.
6 沈萍萍, 陈珂, 张燕, 等. 分布仿真系统二次调度负载均衡策略研究. 计算机仿真, 2011, 28(9): 223–225, 290.
7 Teodoro S, Do Carmo AB, Adornes DC, *et al*. A comparative study of energy-aware scheduling algorithms for computational grids. Journal of Systems and Software, 2016, 117: 153–165. [doi: 10.1016/j.jss.2016.02.017]
8 Platform Corporation. Administering and configuring IBM platform LSF. https://www.ibm.com/support/knowledgecenter/en/SSETD4_9.1.3/lsf_kc_managing.html. [2017-09-25].
9 郝永生, 卢俊文, 刘冠峰, 等. 计算密集型与数据密集型混合网格作业调度算法. 计算机工程与科学, 2014, 36(8): 1423–1429.
10 Goswami S, Das A. Optimization of workload scheduling in computational grid. Proceedings of the 5th International Conference on Frontiers in Intelligent Computing: Theory and Applications. Singapore. 2017.
11 王小宁, 肖海力, 曹荣强. 面向高性能计算环境的作业优化调度模型的设计与实现. 计算机工程与科学, 2017, 39(4): 619–626.
12 巩子杰, 张亚平, 张铭栋. 分布式计算中基于资源分级的自适应 Min-Min 算法. 计算机应用研究, 2016, 33(3): 716–719, 725.
13 廖大强, 邹杜, 印鉴. 一种基于优先级的网格调度算法. 计算机工程, 2014, 40(10): 11–16. [doi: 10.3778/j.issn.1002-8331.1308-0418]
14 Platform Corporation. LSF API Reference. Platform Computing, 2016: 77–78.
15 盛乐标, 周庆林. 利用 LSF API 实现高性能计算机集群的机时统计. 实验科学与技术, 2014, 12(2): 50–52, 117.