·实践与应用·

基于线程池的多任务并行处理模型

高 翔 张金登

(中国电子科技集团公司第二十八研究所南京 210007)

摘 要:通过对大型系统中多线程模式在实际应用中存在的优缺点分析,提出基于线程池的多任务并行处理模型,并在此基础上详细描述了该模型的3个主要功能模块,为解决多线程环境下如何提高任务并行处理效率提供了一种实现方法。

关键词:多任务并行处理;多线程技术;线程池

中图分类号:TP312 **文献标识码**:A **文章编号**:1674-909X(2012)04-0054-03

Multi-task Parallel Processing Model Based on Thread Pool

Gao Xiang Zhang Jindeng

(The 28th Research Institute of China Electronics Technology Group Corporation, Nanjing 210007, China)

Abstract: By analyzing advantages and disadvantages of multi-thread mode in large systems, a multi-task parallel processing model based on the thread pool is established. On this basis, the three major functional blocks in the model are presented. The model provides a method for improving the efficiency of task parallel processing in the multi-threaded environment.

Key words: multi-task parallel processing; multi-thread technology; thread pool

0 引 言

多线程技术是现代操作系统所提供的重要能力之一,并广泛应用于各类软件的设计与实现,它能够在主进程之外并发多通道地提供数据处理支持,从而使得程序处理效率得到提升。多线程技术的使用使得软件交互界面更加友好,任务响应时间更加迅速,并凭借其特有的优势为多任务高效处理提供了有效的解决方案[1]。

然而,在现有的大型系统服务中,系统所要处理的任务数以万计,多线程模式纵然可以解决多任务并行处理的需求,提高多任务处理能力,但大量任务处理请求导致频繁的线程创建和销毁消耗了大量的系统额外时间,使多线程技术并没有发挥出预期的性能。如何对现有系统中多线程应用进行改造,使其能够尽可能发挥出多线程技术的效能,是本文主要讨论的问题。

1 线程池工作原理

在面向对象编程中,创建和销毁对象很费时,因为创建一个对象要获取内存资源或者其他更多资源;在Java中更是如此,虚拟机试图跟踪每一个对象,以便能够在对象销毁后进行垃圾回收。因此,提高程序效率的一个手段就是尽可能减少创建和销毁对象的次数,特别是一些很耗资源的对象创建和销毁[2]。为了节省对象创建和销毁所产生的系统时间需要准备一个容器用来保存一批这样的对象。当需要用这种对象时,不再需要每次去创建,直接从容器中取出现成的直接使用。由于节省了创建和销毁对象所需的开销,程序性能得以提升,存放和管理这些线程对象的容器就是线程池[3]。

1.1 线程池组成

如图1所示[4],一般简单线程池至少由以下3部分组成:

收稿日期:2012-04-29

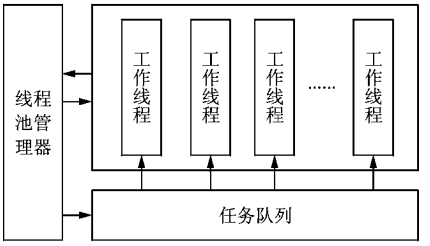

图1　线程池组成结构

1) 线程池管理器：用于创建和管理线程池中的工作线程，定时对任务队列扫描、清理过期未提交工作线程执行的积压数据；

2) 工作线程：用于任务处理的独立单元，并根据系统初始设定创建其数量；

3) 任务队列：提供一种缓冲机制用于存放等待处理的任务，工作线程根据相应的原则从任务队列中获取数据处理。

1.2　运作规律

线程池采用预创建的技术，在应用程序启动之后，将立即创建一定数量的线程(N_1)，放入空闲队列中。这些线程都处于阻塞状态，不消耗CPU，但占用较小的内存空间。当任务到来后将其存入任务队列，线程池管理器选择一个空闲线程，把任务传入此线程中运行。当N_1个线程都在处理任务后，线程池管理器自动创建一定数量的新线程，用于处理更多的任务。在任务执行完毕后线程也不退出，而是继续保持在池中等待下一次的任务[5]。

基于这种预创建技术，线程池将线程创建和销毁的时间开销分摊到各个具体任务上，执行次数越多，每个任务所分担的线程开销则越小。因此，线程池技术适用于大量频繁的任务执行，并且执行任务时任务之间联系较少的系统需求。如果任务之间联系过多，则需要考虑线程之间同步带来的开销。因此，在线程池使用时，应尽量减少该类任务执行，确保处理性能。

2　基于线程池的多任务处理建模

通过对上述线程池工作原理分析，同时结合系统业务处理需求，建立基于线程池的多任务处理模型，如图2所示。

图2中，基于线程池的多任务处理模型主要由以下3部分组成：

1) 任务调度模块

任务调度模块负责接收任务处理请求，并将接收到的处理请求加入任务等待队列，同时，对任务等待队列实时扫描，发现处理请求后按照先进先出的原则选择任务，并依据任务调度策略为该任务分配工作线程[6]。主要包括以下2个工作单元：

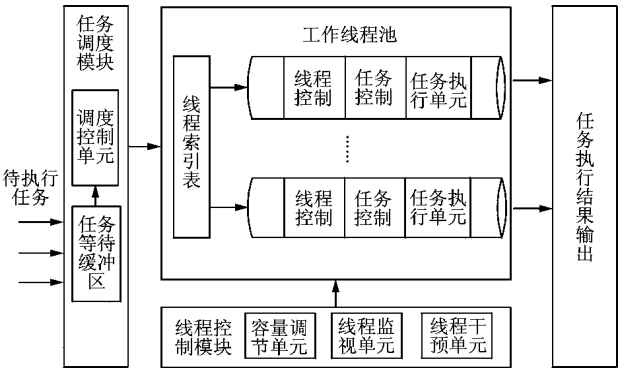

图2　基于线程池的多任务处理模型

(1) 任务等待缓冲区

任务等待缓冲区用于存放接收到的任务处理请求，缓冲区大小设置可根据当前缓冲区使用情况自适应调节。

(2) 调度控制单元

实时扫描任务等待缓冲区中的任务处理请求，获取到任务处理请求后，通过对线程索引表遍历确定空闲线程。如存在空闲线程，则将任务处理请求直接交由该工作线程的任务等待队列等待执行；如未遍历到空闲线程，则根据先空闲先执行的原则通过线程监视单元记录的工作日志，查询最近一次空闲的线程标识，将任务处理请求交由该工作线程的任务等待队列等待执行。

2) 工作线程池

工作线程池主要负责创建线程池和线程索引表，根据预设的线程池容量创建相应数量的工作线程，同时为每个工作线程开辟与之对应的任务等待队列，线程运行过程中实时扫描该等待队列，获取相关任务参数完成数据计算以及数据显示等操作。主要包括以下2个工作单元：

(1) 线程索引表

线程索引表以链表形式将线程池中各工作线程标识和与之相对应的任务等待队列进行关联保存。调度控制单元根据当前任务分配情况实时更新线程索引表。

(2) 工作线程

工作线程是实际处理任务的计算单元，主要包括线程控制、任务控制和任务执行单元3部分。线程控制单元负责线程的创建、运行和销毁，任务控制单元通过对各自任务等待队列的扫描获取任务执行参数，并提交任务执行单元进行运算将计算结果输出。

3) 线程控制模块

线程控制模块主要负责在系统运行过程中实时

监视工作线程池中的多线程，并通过计算当前系统负载，利用容量调节单元对线程池中线程数量进行相应的增减，使计算资源达到最优的负载平衡，同时对工作异常的线程进行回收。该模块主要包括以下 3 个工作单元：

(1) 容量调节单元

实时对系统性能进行监测，并根据监测到的系统性能情况自适应增加或减少线程池中工作线程数量，以保证系统运行的整体性能稳定。

(2) 线程监视单元

用于对工作线程池中工作线程进行监视，定时记录工作线程的工作状态，并以工作日志形式进行存档。

(3) 线程干预单元

线程干预单元定时调阅监视单元所记录的运行日志，并对记录中存在异常的线程及时释放和重新分配。

4) 任务执行结果输出

任务执行结果输出部分为任务计算结果提供输出接口，系统通过该接口获取任务执行数据并提交前端应用进行展现。

3 试验结果

测试主要考察在固定任务数下随着线程数量的增加，采用线程池技术和未采用线程池技术对系统性能的影响。

测试中对1 000 个随机数进行快速排序，每个工作线程相互独立完成5 000 次排序，计算不考虑线程间同步的耗时，在任务执行开始和结束时分别记录当前的系统时间，以便于获取任务处理的耗时。

测试机器为普通 PC 机(戴尔 8600)，CPU 为奔腾单核3 .0 GHz，内存2 GB，试验结果如表1 所示(尚未在多核 CPU 中测试验证)。

由表 1 中可以看出，采用传统多线程模式时，随着线程数量的增加系统运行耗时相对稳定，但频繁的线程创建和销毁对系统的消耗导致运行性能过低；采用线程池模式时，由于节省了不必要的系统开销，对于处理大量任务的效率明显提高。

对测试结果进一步分析发现：合理配置线程池容量对系统性能影响至关重要。一旦尺寸选择不合理(过大或过小)，就会降低系统运行性能。线程池容量过小，将出现任务数据排队，导致任务出现等待不能及时处理的情况；线程池容量过大，则会出现多线程调度导致系统开销太大及计算资源占用过多，以至于计算机整体性能下降等问题。如何合理的设置线程池容量是一个复杂工作，需要考虑各方面因素平衡和估计一个合理的线程池容量。

表1 测试数据及对应结果

线程数量/个	完成数量/次	没有应用线程池所用时间/ms	应用线程池所用时间/ms
1	5 000	3 896	1 130
2	5 000	3 455	951
4	5 000	3 425	720
8	5 000	3 475	260
16	5 000	3 505	211
32	5 000	3 455	251
64	5 000	3 595	301
128	5 000	3 515	381
256	5 000	3 495	504
512	5 000	3 425	788

4 结束语

通过对线程池技术工作原理的分析，建立了基于线程池的多任务处理模型，并通过试验对比分析证明了采用线程池技术的任务处理效率比普通多线程的任务处理有明显的提高。同时，由于采用了线程控制模块等辅助技术，使得多线程的运行稳定性也得到了大幅提高。

综上所述，利用基于线程池的多任务处理技术有效解决了多线程模式下如何降低系统额外开销、提高系统稳定性等关键问题，为实现复杂系统中多任务高效和可靠的并行处理提供了一种可行的设计思路。

参考文献(References)：

[1] 阿克特. 多核程序设计技术[M]. 北京：电子工业出版社，2007.

[2] 迟利华，刘杰. 并行处理基本原理[M]. 北京：清华大学出版社，2004.

[3] 郝文化. Windows 多线程编程技术与实例[M]. 北京：中国水利水电出版社，2005.

[4] 刘永红. Widows 多任务管理机制分析[J]. 成都大学学报：自然科学学报，2000，19(4)：47-48.
Liu Yonghong. Analysis of multitasking management principles in Windows[J]. Journal of Chengdu University: Natural Science, 2000, 19(4): 47-48. (in Chinese)

[5] 张林波，迟学斌，莫则尧，等. 并行计算导论[M]. 北京：清华大学出版社，2006.

[6] 吴志军. 模块化多任务实时航管雷达数据处理计算机系统[J]. 中国民航学院学报，1999，17(5)：30-31.
Wu Zhijun. Modular multimission real-time ATC radar data processing computer system[J]. Journal of Civil Aviation University of China, 1999, 17(5): 30-31. (in Chinese)

作者简介：

高翔，男(1981－)，工程师，研究方向为信息系统开发。

张金登，男(1979－)，工程师，研究方向为信息系统开发。