

# 线程池技术在网络服务器中的应用

吉利, 潘林云, 刘姚

(南京邮电大学 通信与信息工程学院, 江苏 南京 210003)

**摘要:**目前的网络服务器大多为 C/S 架构,其线程响应客户端要么短且频繁,要么长而连续,这两种服务请求均要求服务器根据系统负载提供稳定的响应时间,并在高并发情况下保证服务器系统的工作稳定性。传统的多线程技术虽能通过并发服务器发挥较好的性能,但线程频繁的创建与销毁会导致巨大的系统开销。为此,在研究线程池技术机理的基础上,针对常见的线程池容量估算的弊端,提出了经优化的池线程容量估算方法。该方法引入经优化的池线程,以解决因大量客户端服务请求所导致的服务器不稳定问题。线程池性能测试与验证结果表明,采用所提出的线程池容量估算方法和线程池策略,有效地降低了因频繁创建线程而导致的系统开销,既保证了高负载条件下服务器的稳定性,又能够为服务器提供稳定的吞吐量。

**关键词:**多线程;线程池;线程池容量;服务器

**中图分类号:**TP39

**文献标识码:**A

**文章编号:**1673-629X(2017)08-0149-03

doi:10.3969/j.issn.1673-629X.2017.08.031

## Application of Thread Pool Technique in Network Server

Ji Li, PAN Lin-yun, LIU Yao

(College of Communications and Systems Engineering, Nanjing University of Posts and Telecommunications, Nanjing 210003, China)

**Abstract:** Network servers often use C/S architecture as the main framework currently. The server needs to create a thread frequently in response to a client short or long and continuous service requests. Each type of server requests are required to be able to provide a stable system load according to the response time for the request and guarantee the stability of the server system under the high concurrency. Traditional multi-threading technology can provide excellent performance in the concurrent server, but can lead to huge system overhead because of its frequent thread creation and destruction. An optimization method for estimating the capacity has been proposed according to the disadvantages of common thread pool capacity estimation so as to improve the thread pool server performance, in which the introduction of the thread pool is employed to solve the instability because of a large number of client request thread creation and destruction causing borderless. Thread pool performance testing and verification results show that thread pool takes a more excellent concurrency control strategy and has provided the stable throughput for the server.

**Key words:** multi-threading; thread pool; thread pool size; server

## 0 引言

随着计算机技术的发展,计算机计算性能得到了质的飞跃,合理使用多线程技术较传统的单线程能够显著提升系统的吞吐量与响应速度,特别在高并发访问的情形下。电子商务<sup>[1]</sup>、消息中间件和其他基于网络的应用程序部分取决于网络服务器响应的及时和可靠性。多线程系统<sup>[2]</sup>,由于系统资源的有效利用和共享内存多处理器架构的普及已成为服务器实现的必然选择。然而在访问频率高且服务时间短的情况下,创

建和销毁线程将消耗巨大的系统资源<sup>[3]</sup>。

线程池中的线程可以重新使用,线程的创建和销毁只发生一次。通过线程池使得多个任务重用线程,大大降低了线程创建与销毁的系统开销<sup>[4]</sup>。且多线程的并发执行,能够充分使用服务器主机资源。

传统的线程池策略采用固定大小的核心线程,即使在低负载的情况下,线程切换也会引起巨大的系统开销。为此,提出了线程池容量分配方法,保证在系统低负载与高负载的情况下,均能得到较高的性能<sup>[5]</sup>。

收稿日期:2016-06-02

修回日期:2016-09-15

网络出版时间:2017-06-05

基金项目:国家自然科学基金资助项目(61271234)

作者简介:吉利(1990-),男,硕士,研究方向为卫星通信技术。

网络出版地址: <http://www.cnki.net/kcms/detail/61.1450.TP.20170605.1506.024.html>

## 1 线程池原理

线程池包含几个重要的参数,工作线程,线程容量与缓存客户端请求的请求队列。线程池的工作原理如图 1 所示。请求队列用于缓存客户端请求,线程用于执行客户端的每一次请求,工作线程从请求队列取出请求,用可重用线程处理请求的任务,并在任务完成后向客户端返回信息。

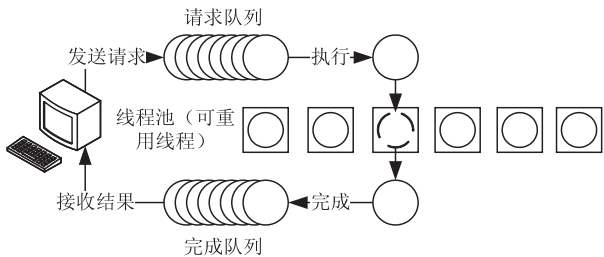


图 1 线程池的核心工作流程

当向线程池提交新的任务时,线程池执行以下规则来保证系统对客户端的响应:

- (1)工作线程用于轮询接受新的客户端请求,如果线程池有足够的空闲线程,则直接提交给线程。若线程池已满,则进入步骤(2)。
- (2)线程池判断请求队列是否已满,若未满载则工作线程将该请求添加到请求队列中;若已满则向客户端返回服务器忙信息<sup>[6]</sup>。
- (3)线程池判断是否有空闲线程以及请求队列是否有未完成的请求任务,循环从请求队列中取出请求任务交给空闲线程执行。
- (4)若线程池没有空闲,且线程池的请求缓存队列已经存在大量缓存请求,可以适当创建新的线程来服务请求<sup>[7]</sup>。

## 2 线程池容量

线程池的容量单方面决定服务器的吞吐量性能参数,如果容量过小且并发请求数量超过该容量,请求将会加入阻塞队列,若队列中已包含大量的已缓存请求,则对系统的响应时间影响巨大。同时若线程池容量过大,线程的上下文切换也会造成巨大的系统开销<sup>[8]</sup>。

图 2 展示了当线程池容量小于 CPU 的核心数目时,CPU 的利用率、系统的响应时间随请求数量增加的变化情况。

由于线程池容量较小,并没有充分利用 CPU 的全部资源,所以 CPU 使用率一直处于一个较低的水平,处理器的性能并没有被充分利用。在请求队列满时,服务器应用开始拒绝新的客户端请求,此时到达服务器的吞吐量。

图 3 展示了高容量线程池 CPU 的利用率、系统的响应时间随请求数量增加的变化情况。

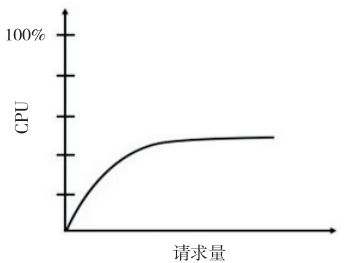


图 2 响应时间与请求数目的关系

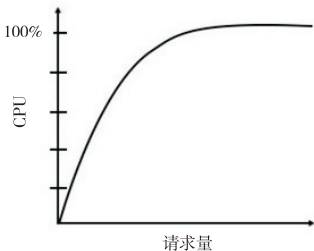


图 3 CPU 利用率与请求数目的关系

并发用户请求数量增加,服务器创建相同数量的线程来服务请求,此时 CPU 很容易达到满载状态,但由于操作系统中存在大量线程,系统资源大量消耗在线程的上下文切换中。此时服务器的响应时间也会增加。所以线程池容量要避免设置太小或太大,太小不能充分使用 CPU 的性能,太大会造成频繁的上下文切换,带来巨大的系统开销。所以线程池容量需要根据负载和部署条件动态改变。

最佳容量大小可以通过设置等比例间隔通过实际压力测试获取最优结果。在计算密集型系统执行中能够获得最佳的 CPU 利用率<sup>[9]</sup>。然而在 I/O 密集型系统,需要用等待概率比例来估算任务的执行时间。Little's Law 公式为:

$$W = \frac{L}{\lambda} \tag{1}$$

其中,  $W$  为请求的平均处理时间;  $L$  为长时间观

察得到的平均请求数量;  $\lambda$  为新请求的到达率。

例如每秒有 10 个请求到达,并且每个请求消耗 1 秒进行处理。这种情况下需要创建具有 10 个线程容量的线程池,从而保证正常的响应时间。

### 3 线程池容量估算

服务器性能以该服务器的响应时间、吞吐量与 CPU 利用率来衡量。系统可利用的资源包括 CPU 核心数目与内存等。处理器执行周期衡量了处理器的处理速度,同时也是线程池管理的主要资源<sup>[10]</sup>。

JDK 从 1.5 开始提供了 `ThreadPoolExecutor` 类管理线程池,该类提供了一系列接口配置与实时定制线程池容量,其可配置参数为 `CorePoolSize`(核心大小)、`MaximumPoolSize`(最大值)与 `KeepAliveTime`(线程存活时间)<sup>[11]</sup>。其中核心线程池尺寸可以通过设置这三个参数来达到线程池最优性能。而最大线程池尺寸一般不超过系统的资源限制。给出公式如下:

$$N_l = N_c * N_u * (1 + \frac{T}{C}) \quad (2)$$

其中,  $N_l$  为需要设置的核心线程数目;  $N_c$  为当前 CPU 的可用核心数;  $N_u$  为预期的 CPU 核心利用率,为任务的等待时间与计算时间的比值。

### 4 线程池性能测试

构建简单的网络服务器,用 Java 代码实现服务器主线程代码片段(服务器绑定端口为 8080),如下:

```
ThreadPoolExecutor exec = prepareThreadPool();
ServerSocket listener = new ServerSocket(8080);
while(true) {
    Socket sock = listener.accept();
    exec.execute(new Task(sock));
}
```

线程池线程的任务是服务用户的响应,并模拟出 1 ms 的运算延时<sup>[1]</sup>。

```
public class Task implements Runnable {
    private Socket socket;
    public Task(Socket socket) {
        this.socket = socket;
    }
    public void run() {
        for(long i=0;i<10;i++);
        try {
            outPutData(socket);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

性能测试从 CPU 的利用率与响应时间两个参数

进行测试<sup>[12]</sup>。服务器为每个请求执行 0.1 ms 的服务时间。并且预期的 CPU 利用率为 60%,请求的平均等待时间与计算时间之比为 500(预期等待时间为 0.2 s),根据容量估算公式得到相应的估算容量为 600<sup>[13]</sup>。

在测量 CPU 利用率上,用户并发数采样点分别为 [1,2,4,8,16,32,64,128,256,512,1 024]。监控 CPU 使用率,并通过 Matlab 绘制曲线<sup>[14-15]</sup>。

其 CPU 利用率如图 4 所示。在核心并发请求数不超过 600 时,CPU 利用率线性增加,由于线程池并发线程数的影响,CPU 利用率逐渐上升并稳定。

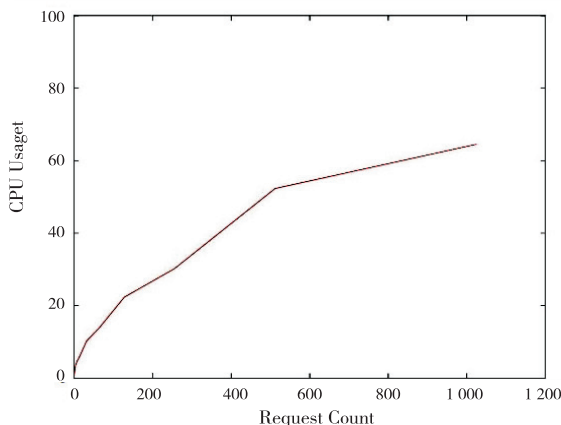


图 4 CPU 利用率与请求数目的关系

测试响应时间时,并发请求数从 0 至 1 200,相邻采样间隔为 50。测量每组给定的并发请求数目下的系统平均响应时间,记录并绘图。

其响应时间随并发连接数的关系如图 5 所示。在并发连接 600 以内可以保证稳定的系统响应时间(ms)。由于资源的竞争与大量请求的入队,系统的响应时间相应增加,当达到系统的最高负载后,拒绝新的请求。

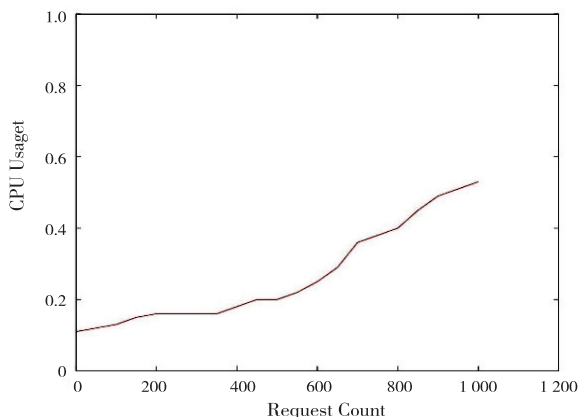


图 5 CPU 利用率与请求数目的关系

### 5 结束语

为了降低因频繁创建线程而导致的巨大系统开销,在研究线程池工作原理的基础上,提出了线程池容

(下转第 155 页)

同义词库的容量和准确性,通过系统界面信息或系统后台人员与供给方的互动沟通等方式引导供给方增强商品或服务信息的准确性。

3 结束语

为解决自然语言实现供需信息的检索和自动匹配,满足供给方对自身商品或服务的特有属性扩展进行描述的需求,提出了一种基于可扩展多语种供求商品或服务信息库和协同机器翻译自然语言的供需信息跨语言信息检索算法。测试结果表明,该算法一定程度上满足了供需信息检索与自动匹配的需求,弥补了传统供需检索匹配方式在自然语言和特性描述支持上的不足,可方便地进行多语种的扩展,使得供需双方的语言障碍在一定程度上得以克服,并获得了较好的用户体验效果。

参考文献:

[1] 蒋忠中,樊治平,汪定伟,等. 具模糊信息的多数量多属性电子交易匹配问题[J]. 管理科学学报,2014,17(5):52-65.

[2] Alpar F Z. Matchmaking framework for B2B e-marketplaces[J]. Informatica Economica Journal,2010,14(4):164-170.

[3] 陈向,刘义,柴跃廷. 基于图论的电子易货商品自动匹配系统[J]. 计算机工程,2009,35(17):283-284.

[4] 陈冬林,聂规划,刘平峰. 基于本体的 B2B 电子商务 MAS 模型及商品匹配算法[J]. 计算机工程与应用,2007,43(10):199-201.

(上接第 151 页)

量的估算方法。以服务器的 CPU 使用率与系统响应时间作为衡量标准,对线程池容量估算方法进行了有效性测试。结果表明,采用该方法的服务器在各种负载条件下均能满足有效的 CPU 利用率与稳定的系统响应时间,既保证了整体稳定性,又能提供稳定的吞吐量。

参考文献:

[1] 张垠波. 线程池技术在并发服务器中的应用[J]. 计算机与数字工程,2012,40(7):153-156.

[2] Ling Yibei, Mullen T, Lin Xiaola. Analysis of optimal thread pool size[J]. ACM SIGOPS Operating Systems Review,2000,34(2):43-45.

[3] 李昊,刘志镜. 线程池技术的研究[J]. 现代电子技术,2004,27(3):77-80.

[4] 詹新林,王公亭,徐晓钟. 基于线程池数据分析系统的设计与实现[J]. 微计算机信息,2008,24(33):266-268.

[5] 张尧学,宋虹,张高. 计算机操作系统教程[M]. 第 4 版. 北京:清华大学出版社,2013.

[5] 梁海明,姜艳萍. 一种考虑中介交易态度的买卖双边匹配决策方法[J]. 运筹与管理,2013,22(5):128-133.

[6] 耿骞,赖茂生. 自然语言检索的实现及其关键问题[J]. 情报科学,2007,25(5):733-741.

[7] Anssi Y J, Andras K, Jacques S. Finite-state methods and models in natural language processing[J]. Natural Language Engineering,2011,17(2):141-144.

[8] 谢文亮,王石榴. 基于语义 Web 的科技期刊网络信息检索及其应用[J]. 科技管理研究,2015,35(2):196-200.

[9] 司莉,庄晓喆,贾欢. 近 10 年来国外多语言信息组织与检索研究进展与启示[J]. 中国图书馆学报,2015,41(4):112-126.

[10] 庞观松,张黎莎,蒋盛益. 个性化跨语言学术搜索技术研究[J]. 情报学报,2011,30(8):870-874.

[11] 张玥杰,郭依昆,连理,等. 基于英汉机译实现跨语言信息检索[J]. 小型微型计算机系统,2004,25(7):1135-1140.

[12] 吴晨,缪建明,张全. 跨语种信息检索中的文本比较及结果生成算法[J]. 计算机工程与应用,2005,41(29):11-15.

[13] Vulic I, Smet W, Moens M F. Cross-language information retrieval models based on latent topic models trained with document-aligned comparable corpora[J]. Information Retrieval, 2013,16(3):331-368.

[14] Al-Nazer A, Albukhitan S, Helmy T. Cross-domain semantic web model for understanding multilingual natural language queries; english/arabic health/food domain use case[J]. Procedia Computer Science,2016,83:607-614.

[6] 封玮,周世平. Java 中的线程池及实现[J]. 计算机系统应用,2004(8):16-18.

[7] 王俊峰. 一种实时集群系统负载均衡通用模型的研究及应用[D]. 长沙:湖南大学,2008.

[8] 欧昌华,李炳法. 线程池在网络服务器程序中的应用[J]. 信息技术,2002(5):11-14.

[9] 宋立昊. 基于线程池的 WEB 服务器实现和监测[D]. 长春:吉林大学,2011.

[10] 王华,马亮,顾明. 线程池技术研究与实现[J]. 计算机应用研究,2005,22(11):141-142.

[11] Pyarali I, Spivak M, Cytron R, et al. Evaluating and optimizing thread pool strategies for real-time CORBA[J]. ACM SIGPLAN Notices,2001,36(8):214-222.

[13] 刘雪梅. 服务器端软件性能分析和诊断[M]. 北京:北京邮电大学出版社,2011.

[14] Belkin R. Mechanism for obtaining a thread from, and returning a thread to, a thread pool without attaching and detaching; U. S. , 6 766 349[P]. 2004-07-20.

[15] 许俊奎,徐凤刚,潘清. Web 服务器性能测试工具的设计与实现[J]. 计算机测量与控制,2005,13(11):1204-1206.