基于共享内存的路径规划负载均衡算法

邱鲤鲤[1]，姚术林[2]，刘子立[2]，陈　云[2]

（1. 厦门大学 建筑与土木工程学院，福建 厦门 361005；2. 厦门精图信息技术股份有限公司，福建 厦门 361008）

摘　要：为满足前端对路径规划高并发服务需求，本文设计后台采用集群路径规划服务器来实现并行处理，以达到对前置 FastCGI 服务器请求进行分流的目的。由此，本文提出基于共享内存的路径规划负载均衡算法，可根据不同路径规划服务器的性能差异，设置相应的加权系数，并按照加权调度算法，将任务交由合适的路径规划服务器来处理，以均衡所有路径规划服务器之间的负载能力，减少服务延迟，从而提高了路径规划服务的稳定性、扩展性、高可用性和执行的效率。

关键词：负载均衡；路径规划；共享内存；分布式并行处理

中图分类号：P208　　**文献标识码**：A　　**文章编号**：1672－5867(2016)09－0064－04

Load Balancing Algorithm for Path Planning Based on Shared Memory

QIU Li－li[1], YAO Shu－lin[2], LIU Zi－li[2], CHEN Yun[2]

(1. School of Architecture and Civil Engineering, Xiamen University, Xiamen 361005, China; 2. Xiamen KINGTOP Information Technology Co. Ltd., Xiamen 361008, China)

Abstract: To meet the front end on the path planning for high concurrency service needs, this paper designs background using cluster path planning server to realize parallel processing, in order to achieve the diversion for the prepositive FastCGI server requests. Thus, this paper presents the load balancing algorithm for path planning based on shared memory, which can set the corresponding weighting coefficient based on the differences in the performance of different path planning server, and according to the weighted scheduling algorithm, the task to be processed by the appropriate path planning server. The result shows that this way can not only balance between all path planning server load capabilities, but reduce the server delay, so as to improve the stability of path planning service, expansion, high usability and execution efficiency.

Key words: load balancing; path planning; shared memory; distributed parallel processing

0　引　言

随着地理信息产业的建立和数字化信息产品在全世界的普及，地理信息系统已深入到各行各业甚至各家各户，成为人们生产、生活、学习和工作中不可缺少的工具和助手。路径规划问题是地理信息系统网络分析中最基本和关键的问题，但随着用户数目迅速扩容，并发访问人数不断增加，服务器负载越来越重，常常超负荷运行。

依靠硬件升级会造成大量的资源浪费，业务量的不断提升需要硬件设备不断升级，然而性能再卓越的设备也不一定能满足当前业务量的需求，于是多设备的集群系统由于其高性能、低价格、可扩展性强等特点，正被得到广泛地使用。如何在处理能力相当的多台网络设备之间进行合理的业务量分配，使得集群系统不至于出现一部分设备过忙而另一部分设备却未充分发挥处理能力的情况，就成为一个急需解决的问题，负载均衡机制应运而生[1]。

在现有的各种负载均衡策略中，他们在任务粒度、拓扑结构、局部性及优化的性能指标等方面做了不同的假设[2]。本文提出了基于共享内存的路径规划负载均衡算法，并通过 FastCGI 程序来实现。本算法根据不同路径规划服务器的性能差异设置相应的权值，并按照一定算法根据权值将任务交由负载较轻的服务器来处理，以平衡所有的路径服务器和应用之间的通信负载，提高了 FastCGI 服务器的稳定性、扩展性、高可用性和执行的效率。

1　运行环境概述

一台 FastCGI 服务器可以同时运行多个 FastCGI 实例进程，每个 FastCGI 实例进程可以响应多个前端请求队列，并且每个实例进程可以连接单个或多个路径规划服

收稿日期：2016－01－12

作者简介：邱鲤鲤(1983－)，女，福建惠安人，工程师，城市规划专业硕士研究生，主要从事 GIS 与空间行为分析的应用研究工作。

务器,而每个路径规划服务器又可以同时处理多个路径规划任务。由于路径规划耗时比较长,通常 500 ms 左右(依赖服务器的计算能力和数据规模),且 CPU 具有多核,可以同时执行多个路径规划任务,即可以同时有多个路径规划进程在处理。因此,单台普通路径规划服务器通常不能满足一台 FastCGI 服务器的需求。

由上述可知,一台 FastCGI 服务器可以配置多台路径规划服务器,并且实现分流。但是不同服务器的性能及运行效率各不相同,同时我们不希望因系统的故障而造成多层应用程序系统无法执行或执行错误。因此,系统的稳定性、扩展性和执行的效率是非常重要的,我们要求 FastCGI 服务器对多个路径规划服务器具有负载均衡的能力。

2　算法描述

由于 FastCGI 服务器调用的是无主控进程,各个进程之间是独立对等关系,而共享内存是操作系统多进程间的通信方法,通常用于一个程序的多进程间通信,所以本文采用共享内存来实现不同 FastCGI 实例进程之间的通信。

2.1　共享内存设计

共享内存数据结构见表 1。

表 1　共享内存数据结构表

Tab.1　Shared memory data structure table

序号	数据项	数据长度	备注
1	共享内存长度	2 字节	用于判断共享内存长度以及有多少台路径规划服务器。
2	共享内存格式版本号	1 字节	版本号从 1 – 254,0 和 255 保留。
3	配置文件 UTC 时间	4 字节	用于判断配置文件是否需要更新。如果需要更新,则重新加载配置文件,并更新共享内存结构。
4	上次选择的服务器	1 字节	初始化为 – 1
5	当前调度的权值	1 字节	初始化为 0
6	路径规划服务器个数	1 字节	1 个 FastCGI 最多支持 255 台路径规划服务器。如果路径规划服务器过多,总体性能无法提升,可能反而下降了。
7	服务器 1 名称标识长度(Len1)	1 字节	为了保证长度不会越界,以及基本够用的原则,定义服务器名称标识最大长度为 128 个字节。如果超过该长度,则只截取前面的 128 字节,因此 Len1 < = 128。剩下 128 个字节需要用于表示路径规划的其他含义,今后如需要,则可进一步扩展。
8	服务器 1 名称标识	Len1 字节	服务器 1 的名称标识。
9	服务器 1 权重	1 字节	设置路径规划服务器的权重信息。
10	服务器 2 名称标识长度(Len2)	1 字节	为了保证长度不会越界,以及基本够用的原则,定义服务器名称标识最大长度为 128 个字节。如果超过该长度,则只截取前面的 128 字节,因此 Len1 < = 128。剩下 128 个字节需要用于表示路径规划的其他含义,今后如需要,则可进一步扩展。
…	…	…	…

2.2　共享内存的创建和释放

因为 FastCGI 由 IIS 进行管理,IIS 将根据请求情况,自动启动或停止 FastCGI 进程,而共享内存是操作系统进行全局管理。因此,共享内存何时开辟、何时关闭,就成了一个问题。

1) 共享内存的创建

通常 FastCGI 启动时,我们检测共享内存是否已创建,如果未创建,则进行创建; 如果已创建,则判断已开辟的内存是否是上次 FastCGI 异常后遗留下来的,还是之前启动时创建的。

本文主要通过以下 3 点来进行判断: 共享内存大小是否一致; 共享内存格式版本是否一致; 共享内存与配置文件的时间戳是否一致。

如果检查的 3 点都一致,则可继续操作。否则,就表示当前已创建的内存是上次异常留下的,进程需要释放共享内存,并且重新创建共享内存。

2) 共享内存的释放

而当某一个 FastCGI 进程释放时,可能还有很多实例正在运行。只有当最后一个实例退出时,才能销毁共享内存,防止操作系统内存泄漏。但由于共享内存名称相同,在操作系统中也只有一份实例。因此,即使不销毁,也不会造成内存重复泄漏情况。同时由于命名规范性,该共享内存名称只属于指定的程序,不会造成与其他软件的共享内存命名冲突的问题。

当然,如果在一个 FastCGI 退出时,也可以马上销毁共享内存,但会丢失目前正在调度的信息,并且其他 FastCGI 进程又需要重新创建共享内存,因此不推荐。故而,可以采用引用计数器的方式,如果共享内存已创建,则引用计数器递增,而在 FastCGI 退出时,引用计数器递减,若减到 0 时,则可安全释放共享内存。

2.3　共享内存锁定操作

为了保证同一时刻,只有一个 FastCGI 进程操作共享内存,因此,需要添加互斥锁。在操作共享内存前进行加锁,在操作结束后进行解锁。

2.4　配置文件更新

当检测到配置文件更新时,则采用共享内存的创建方式进行重新创建。虽然会造成已有调度信息丢失,但影响不大。

2.5　算法流程

权值算法调度流程如下[3]: 假设一组路径规划服务器 $s = \{S_0, S_1, \cdots, S_{n-1}\}$,$W(S_i)$ 表示服务器 S_i 的权值,一个

指示变量 i 表示上一次选择的服务器，指示变量 cW 表示当前调度的权值，max(s) 表示集合 s 中所有服务器的最大权值，gcd(s) 表示集合 s 中所有服务器权值的最大公约数。变量 i 初始化为 -1，cW 初始化为 0。

```
while( true)
{
i = ( i + 1) modn;
if( i = =0)
{
cW = cW - gcd( s) ;
if( cW < =0)
{
cW = max( s) ;
if( cW = =0)
{
return NULL;
}
}
}
if ( W( S_i) > = cW )
{
returnS_i;
}
}
```

建议进行权重配置时，自动取所有路径规划服务器权值的最大公约数为 1，并且自动调整相应的权值系数。

2.6 路径规划服务器失效处理

如果路径规划服务器失效，则有如下几种可选方案，其一，进行重试 n 次(推荐 n 取 0 ~ 2) ; 其二，重试失败之后，可选择放弃该路径规划服务器，或者休眠一段时间(该段时间不参与调度) ，或者下次再参与调度; 其三，重试失败后，应把现请求移交给下一台路径规划服务器。本文采用采用第 3 种方案。

3 实验及性能分析

3.1 实验环境及参数

FastCGI 及路径规划服务器的配置信息见表 2，其中 FastCGI 服务器与路径规划服务器 1 是同一台机器，并且 LoadRunner12 安装部署在 FastCGI 服务器上。

表 2 服务器配置信息表

Tab. 2 Server configuration information table

序号	名称	机器类型	配置
1	FastCGI 服务器	台式机	CPU: i5 -4570 3.2GHz* 4 内存: 16G 硬盘: 128GB 闪存 硬盘: 2TB 硬盘: 500GB
2	路径规划服务器 1	台式机	CPU: i5 -4570 3.2GHz* 4 内存: 16G 硬盘: 128GB 闪存 硬盘: 2TB 硬盘: 500GB
3	路径规划服务器 2	台式机	CPU: Intel(R) Xeon(R) CPU E56202.4GHz* 2 内存: 24GB 硬盘: 160GB
4	路径规划服务器 3	台式机	CPU: i7 -2600 3.4GHz* 2 内存: 4GB 硬盘: 1TB
5	路径规划服务器 4	台式机	CPU: Intel(R) Xeon(R) CPU E55042GHz* 2 内存: 4GB 硬盘: 2TB
6	LoadRunner 测试服务器	台式机	CPU: i5 -4570 3.2GHz* 4 内存: 16G 硬盘: 128GB 闪存 硬盘: 2TB 硬盘: 500GB

3.2 路径规划结果

由于路径规划设定条件不同，如起讫点、必经点、避开点等取值不同，会造成同样配置的服务器及同样的路网数据(本文采用北京市路网数据) ，处理的时间也不尽相同，因此，测试采用相对简单的起讫点规划方式，但起讫点的取值是随机的。

3.3 结果性能对比

采用单台服务器进行动态随机查询，测试结果见表 3。

表 3　单台服务器测试结果

Tab. 3　Single server test results

迭代次数	并发用户数	FastCGI 线程数	最大请求数	平均请求响应时间	平均每秒字节数	平均每秒点击数	丢失数量	路径规划服务器
1	50	4	200	7.775	8 310	4	0	服务器 1
1	20	4	200	3.756	7 611	4.182	0	服务器 1
1	50	4	200	54.086	95	0.079	0	服务器 4
1	4	2	200	6.908	703	0.559	0	服务器 4
1	2	2	200	3.433	642	0.566	0	服务器 4
1	50	8	200	11.14	4 127	3.178	0	服务器 3
1	20	8	200	5.093	3 698	3.083	0	服务器 3
1	16	8	200	4.295	3 404	2.907	0	服务器 3
1	20	4	200	2.72	6 745	5.772	0	服务器 2
1	50	4	200	6.219	7 032	5.801	0	服务器 2

采用负载均衡方案后,测试结果见表 4。

表 4　负载均衡测试结果

Tab. 4　Load balancing test results

迭代次数	并发用户数	CGI 线程数	最大请求数	平均请求响应时间	平均每秒字节数	平均每秒点击数	丢失数量	服务器权重
1	50	14	200	4.435	9125	7.942	0	服务器 1:6 服务器 3:5 服务器 4:3
1	50	15	200	3.245	13.22	11.113	0	服务器 1:4 服务器 3:4 服务器 2:5 服务器 4:2
1	50	16	200	3.108	13.879	11.595	0	服务器 1:2 服务器 3:2 服务器 2:3 服务器 4:1
1	50	16	200	2.99	10.983	12	0	服务器 1:2 服务器 3:2 服务器 2:3 服务器 4:1
1	100	10	200	5.296	10.997	12.015	0	服务器 1:2 服务器 3:2 服务器 2:3 服务器 4:1
1	100	20	200	5.361	10.857	11.863	0	服务器 1:2 服务器 3:2 服务器 2:3 服务器 4:1
1	50	10	200	5.115	8.166	6.517	0	服务器 2:2 服务器 3:2 服务器 4:1

从表 3 和表 4 对比中可以发现,负载均衡算法实施后,能够有效地对 FastCGI 服务器请求进行合理的分流,符合预期的设计效果。由于 LoadRunner 测试软件部署在 FastCGI 服务器上,对测试的结果会有一定影响。如果独立安装部署,相信性能上会更理想。

4　结束语

该算法较好地解决了 FastCGI 服务器下路径规划高并发方面的问题,已编程实现并在地理信息系统开发平台中进行了性能测试,验证了方案的可靠性。但该算法也存在局限性: 只适合路径规划和查询方面的高并发,并不适合在线编辑,以及持续会话(session) 方面。由于篇幅限制,本文只给出了较好的可行性解决方案,关于自适应负载均衡及达到负载极限后,保持合理的响应服务,以及拒绝过载请求将另文讨论。

参考文献:

[1] 李坤,王百杰. 服务器集群负载均衡技术研究及算法比较[J]. 计算机与现代化,2009(8):7 - 10.

[2] 王荣生,杨际祥,王凡. 负载均衡策略研究综述[J]. 小型微型计算机系统,2010(8):1681 - 1686.

[3] J. B. Nagle. On packet switches with infinite storage[J]. IEEE Transactions on Communications, 1987, 35(4): 435 - 438.

[编辑: 刘莉鑫]