[**Apache 2.0中prefork.c模块和worker.c模块的比较**](http://www.blogjava.net/hjh/archive/2007/11/13/160139.html)

空闲子进程：是指没有正在处理请求的子进程。

**1、prefork.c模块(一个非线程型的、预派生的MPM)**    prefork MPM 使用多个子进程，每个子进程只有一个线程。每个进程在某个确定的时间只能维持一个连接。在大多数平台上，Prefork MPM在效率上要比Worker MPM要高，但是内存使用大得多。prefork的无线程设计在某些情况下将比worker更有优势：它可以使用那些没有处理好线程安全的第三方模块，并且对于那些线程调试困难的平台而言，它也更容易调试一些。

<IfModule prefork.c>  
ServerLimit  20000  
StartServers  5  
MinSpareServers  5  
MaxSpareServers  10  
MaxClients  1000  
MaxRequestsPerChild 0  
</IfModule>

ServerLimit     2000  
//默认的MaxClient最大是256个线程,如果想设置更大的值，就的加上ServerLimit这个参数。20000是ServerLimit这个参数的最大值。如果需要更大，则必须编译apache,此前都是不需要重新编译Apache。  
生效前提：必须放在其他指令的前面

StartServers  5  
//指定服务器启动时建立的子进程数量，prefork默认为5。

MinSpareServers  5  
//指定空闲子进程的最小数量，默认为5。如果当前空闲子进程数少于MinSpareServers ，那么Apache将以最大每秒一个的速度产生新的子进程。此参数不要设的太大。

MaxSpareServers  10  
//设置空闲子进程的最大数量，默认为10。如果当前有超过MaxSpareServers数量的空闲子进程，那么父进程将杀死多余的子进程。此参数不要设的太大。如果你将该指令的值设置为比MinSpareServers小，Apache将会自动将其修改成"MinSpareServers+1"。

MaxClients  256  
//限定同一时间客户端最大接入请求的数量(单个进程并发线程数)，默认为256。任何超过MaxClients限制的请求都将进入等候队列,一旦一个链接被释放，队列中的请求将得到服务。要增大这个值，你必须同时增大ServerLimit 。

MaxRequestsPerChild 10000  
//每个子进程在其生存期内允许伺服的最大请求数量，默认为10000.到达MaxRequestsPerChild的限制后，子进程将会结束。如果MaxRequestsPerChild为"0"，子进程将永远不会结束。

将MaxRequestsPerChild设置成非零值有两个好处：  
1.可以防止(偶然的)内存泄漏无限进行，从而耗尽内存。   
2.给进程一个有限寿命，从而有助于当服务器负载减轻的时候减少活动进程的数量。

工作方式：  
一个单独的控制进程(父进程)负责产生子进程，这些子进程用于监听请求并作出应答。Apache总是试图保持一些备用的(spare)或者是空闲的子进程用于迎接即将到来的请求。这样客户端就不需要在得到服务前等候子进程的产生。在Unix系统中，父进程通常以root身份运行以便邦定80端口，而Apache产生的子进程通常以一个低特权的用户运行。User和Group指令用于设置子进程的低特权用户。运行子进程的用户必须要对它所服务的内容有读取的权限，但是对服务内容之外的其他资源必须拥有尽可能少的权限。

**2、worker.c模块(支持混合的多线程多进程的多路处理模块)**  
    worker MPM 使用多个子进程，每个子进程有多个线程。每个线程在某个确定的时间只能维持一个连接。通常来说，在一个高流量的HTTP服务器上，Worker MPM是个比较好的选择，因为Worker MPM的内存使用比Prefork MPM要低得多。但worker MPM也由不完善的地方，如果一个线程崩溃，整个进程就会连同其所有线程一起"死掉".由于线程共享内存空间，所以一个程序在运行时必须被系统识别为"每个线程都是安全的"。

<IfModule worker.c>  
ServerLimit  50  
ThreadLimit  200  
StartServers  5  
MaxClients  5000  
MinSpareThreads  25  
MaxSpareThreads  500  
ThreadsPerChild  100  
MaxRequestsPerChild 0  
</IfModule>

ServerLimit 16  
//服务器允许配置的进程数上限。这个指令和ThreadLimit结合使用设置了MaxClients最大允许配置的数值。任何在重启期间对这个指令的改变都将被忽略，但对MaxClients的修改却会生效。

ThreadLimit 64  
//每个子进程可配置的线程数上限。这个指令设置了每个子进程可配置的线程数ThreadsPerChild上限。任何在重启期间对这个指令的改变都将被忽略，但对ThreadsPerChild的修改却会生效。默认值是"64".

StartServers 3  
//服务器启动时建立的子进程数，默认值是"3"。

MinSpareThreads 75  
//最小空闲线程数,默认值是"75"。这个MPM将基于整个服务器监视空闲线程数。如果服务器中总的空闲线程数太少，子进程将产生新的空闲线程。

MaxSpareThreads 250  
//设置最大空闲线程数。默认值是"250"。这个MPM将基于整个服务器监视空闲线程数。如果服务器中总的空闲线程数太多，子进程将杀死多余的空闲线程。MaxSpareThreads的取值范围是有限制的。Apache将按照如下限制自动修正你设置的值：worker要求其大于等于MinSpareThreads加上ThreadsPerChild的和

MaxClients 400  
//允许同时伺服的最大接入请求数量(最大线程数量)。任何超过MaxClients限制的请求都将进入等候队列。默认值是"400",16(ServerLimit)乘以25(ThreadsPerChild)的结果。因此要增加MaxClients的时候，你必须同时增加ServerLimit的值。

ThreadsPerChild 25  
//每个子进程建立的常驻的执行线程数。默认值是25。子进程在启动时建立这些线程后就不再建立新的线程了。

MaxRequestsPerChild  0  
//设置每个子进程在其生存期内允许伺服的最大请求数量。到达MaxRequestsPerChild的限制后，子进程将会结束。如果MaxRequestsPerChild为"0"，子进程将永远不会结束。

将MaxRequestsPerChild设置成非零值有两个好处：  
1.可以防止(偶然的)内存泄漏无限进行，从而耗尽内存。   
2.给进程一个有限寿命，从而有助于当服务器负载减轻的时候减少活动进程的数量。   
注意  
对于KeepAlive链接，只有第一个请求会被计数。事实上，它改变了每个子进程限制最大链接数量的行为。

工作方式：  
每个进程可以拥有的线程数量是固定的。服务器会根据负载情况增加或减少进程数量。一个单独的控制进程(父进程)负责子进程的建立。每个子进程可以建立ThreadsPerChild数量的服务线程和一个监听线程，该监听线程监听接入请求并将其传递给服务线程处理和应答。Apache总是试图维持一个备用(spare)或是空闲的服务线程池。这样，客户端无须等待新线程或新进程的建立即可得到处理。在Unix中，为了能够绑定80端口，父进程一般都是以root身份启动，随后，Apache以较低权限的用户建立子进程和线程。User和Group指令用于设置Apache子进程的权限。虽然子进程必须对其提供的内容拥有读权限，但应该尽可能给予它较少的特权。另外，除非使用了suexec ，否则，这些指令设置的权限将被CGI脚本所继承。

公式：  
ThreadLimit >= ThreadsPerChild  
MaxClients  <= ServerLimit \* ThreadsPerChild  必须是ThreadsPerChild的倍数  
MaxSpareThreads >= MinSpareThreads+ThreadsPerChild

硬限制：

ServerLimi和ThreadLimit这两个指令决定了活动子进程数量和每个子进程中线程数量的硬限制。要想改变这个硬限制必须完全停止服务器然后再启动服务器(直接重启是不行的)。

Apache在编译ServerLimit时内部有一个硬性的限制，你不能超越这个限制。  
prefork MPM最大为"ServerLimit 200000"  
其它MPM(包括work MPM)最大为"ServerLimit 20000

Apache在编译ThreadLimit时内部有一个硬性的限制，你不能超越这个限制。  
mpm\_winnt是"ThreadLimit 15000"  
其它MPM(包括work prefork)为"ThreadLimit 20000

注意  
使用ServerLimit和ThreadLimit时要特别当心。如果将ServerLimit和ThreadLimit设置成一个高出实际需要许多的值，将会有过多的共享内存被分配。当设置成超过系统的处理能力，Apache可能无法启动，或者系统将变得不稳定。