Apache电子商务解决方案之一 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/242/2021_2022_Apache_E7_94_B5_E5_c40_242493.htm 摘要 这篇文章论述安全web服务器的部署，使用它们作为后端系统的代理、SSL负载均衡、及如何解决其他大规模系统的性能和可靠性问题。我们调查了在多服务器环境下安全事务处理的影响,并且研究了负载共享技术创新的途径。 这里包括分布式会话缓存、交叉计算机的CPU增强操作(针对专有硬件)，而且对合并的加密硬件的加速问题和密钥管理系统进行了优化。 背景 这文章的目标是考察关于你的 Web服务器执行安全事务的效果；特别地它怎样影响你的系统的性能和布局。我们按照大量站点使用的机器布局和配置，在标准的http解决方案下，看看它们是否可以扩大到足以处理安全事务（https）。虽然有一些解决方案处于专有的硬设备空间中，但是我们要考察用什么方法使你能使用Apache服务器做相同的事情。 最近公共社区正在关注软件中维护私人密钥的问题，所以如果有其他制造你自己的安全系统的方法，我们要看看谁将会受到这个影响。我们试图了解是否基于硬设备的密码系统装置是一个代价有效的解决方案，并且给你提供基本的知识，帮助你搞清那些厂商宣称的系统的真实情况。 因为大多数浏览器和服务器支持SSL和TLS安全协议，我们将特别集中在它们的讨论上。既然这是一个关于Apache服务器的讨论，虽然很多其他的服务器有这样的概括性的材料到（也包括安全相关的那些服务）,但我们不去考虑其他的服务器，或深入研究太多供应厂商的产品具体细节，。 Apache服务器中的SSL 所以在安全连接和非安全连接之

间的差异是什么？当你使用"https"前缀访问一个站点时，你在试图建立一个使用SSL或TLS协议的连接。一旦地那个连接建立，在浏览器和服务器之间的请求和回应被加密编码成为端到端(1)的连接。这个连接的二个阶段是；握手协议（认证和密钥协定）和隧道协议（请求和回应通过加密编码往返传输(2)）。 SSL密码系统 一旦浏览器和服务器已经完成握手协议,它们可以使用一个诸如 DES或RC4的标准加密算法将其余的通讯译成密码进行数据交换。这个算法被称为加密，并且这个技术称作对称密码系统。对称密码系统使用一个公用密钥用于双方加密并且解密数据，使得这个密码系统通常运行的飞快。但是在浏览器和服务器之前可能之前没有交互，它们需要一些方法用来建立公用密钥。这个过程在握手协议过程中由密钥交换技术完成，该技术基于非对称加密算法（公共/私有密钥密码系统）。这是通常基于RSA算法(3)，典型地使用一对1024位组的 RSA密钥。如同以下表格说明的那样（显示了不同的平台下1024位组RSA签名的速度试验），公钥密码是一种CPU密集型的任务。 在表格1中的图形是在使用了一闲置的机器上所有的可用的CPU资源，并且使用优化组合的RSA算法的版本获得的结果。 因此，为了估计一个提供SSL支持的服务器所需要进行的工作，我们必须检验加密操作产生有效的系统开销，并判定它们多久需要进行一次。除了这些RSA签名操作以外，和一旦初始的SSL事务（握手协议）已经完成、是否那写加密请求的时间和响应数据是有效的呢？原来对于大多数机器而言，这些对称加密算法有着极低的系统开销，并且那些额外的传送未加密数据上的开销相对可以忽略不计。表 2 说明了在不同的闲置机器上大量的数据每秒

可以同时使用二个公共的SSL密码进行加密。 你的密钥在哪里 ？ 当Web服务器在处理SSL事务时，为了完成SSL私有密钥的操作，需要有一份服务器的私有密钥保存在它的内存里。在 1999初，ncipher发现了一个很有效的方法大通过扫描大量数据来寻找私有密钥。 在某些配置下，一些操作系统允许应用程序以同一个用户的身份运行，该程序可以读取其他进程的内存区域。由于CGI程序通常运行于这种方式，在同一个Web服务器上任何一个能够访问的CGI程序，潜在地都可以扫描该Web服务器的内存，以发现该服务器上所有安全站点的私有密钥。 ncipher示范了一个CGI程序，扫描内存并返回运行在服务器上的所有安全虚拟主机的私有密钥。 他们提出了这个问题的解决方案：安全地将你的密钥写入一个外部硬件装置，比如说写入到一个硬件加速器中。然后，加速器不但执行密钥的操作，而是它也有唯一的密钥拷贝。Web服务器不需要向加速器提供密钥，实际上通常将在该装置内部产生密钥并且不会输出它们。然后Web服务器必须手动切断该硬件加速器上的全部密钥操作，因此每个Web服务器都需要具备自己附属的硬件加速器。 nCipher示范的攻击仅仅工作在几个操作系统上，这些操作系统允许其他的进程读取每个存储空间，甚至在那些系统上这些攻击很简单就击败了这些系统。Apache仅需在开始时作为root用户，并允许同非root用户混合以便提供网页服务（事实上这是缺省值配置）。即使Apache作为非root用户启动，你仅仅被影响，是否允许人们编写他们自己的用于安全事务的CGI程序并将之运行在一个Web服务器上。令人遗憾的是这并不完全是被最近覆盖该问题的催促发行的画面。 SSL 会话高速缓冲存储器 为了加速

该请求的处理,当打开一新的连接时,SSL协议特别地允许一客户要求该服务器预先地给出协议会话的摘要。因此一旦该客户机和服务器已经执行了公众密钥操作并协商一对对话密钥（i.e. 4 版本或keep - alive功能被配置为低的超时以保持打开的连接的数目并且进程在控制下完成该握手协议），然后一个将来产生的新连接理论上不需要执行CPU密集的公开密钥操作。每个会话必须有一超时期同它关联（出于安全理由），因此该服务器管理员可以强制一个新的会话，例如，至少每天或每小时被建立一次。 这个意味着如果想要利用这个特色，该服务器必须有一个方法的存储该会话的信息；一个会话高速缓冲存储器。用Apache1.3服务器，我们情况复杂化了：持有大量独立的派生子进程，子进程运行在那个没有公共存储的服务器上。每个进入Apache服务器的请求可能通过不同的子进程来终结。在SSL中，当客户程序标志出它所希望恢复的服务器会话时，会话恢复过程被初始化。即使该客户程序已经就每个子进程进行了会话协商，并且正确地猜测它已经连接的子进程，它也仅能恢复一个SSL会话。对负载很重的已经有上千个子进程的站点来说，这不是有效的方法，而且一个会话高速缓冲存储器会在全部需要该缓存的Apache子进程之间共享。 尽管这是一个十分简单概念，也要需一段时间来了解清楚安全Apache服务器的解决方案。Apache服务器-SSL(5)服务作为一个外部进程，同会话高速缓冲存储器一起开始运行。每个子进程将为那些需要内存高速缓存的会话进程建立一个独立的连接。 但是这种解决方案对于服务器管理员来说实际上是十分难操纵的，因为该Web服务器不得不将它的外部进程重新启动。同样地，由于该Web服务器将继续

运行并且试图对那些始终失败的请求进行服务，所以如果这些进程已经被杀死或者崩溃，这个问题还是依然存在，无法解决。 mod_ssl (6)有它自己的解决方案。它和Apache服务器的SSL扩展服务以相同的方式启动，然后转移到一个会话高速缓冲存储器，这个缓冲存储器通过一个轻量级数据库而不是外部进程来实现。在最近的几个月，mod_ssl开始支持类似堡垒（Stronghold）的方法--使用共享内存高速缓冲存储器。 随着新的会话高速缓冲存储器的到来，有责任正确地配置它。如果系统需要越多的并发操作会话和更长的会话终止时间，就需要更大的内存块来做会话高速缓冲存储器。检验会话高速缓冲存储器的关键速度结果将同样重要。在该会话高速缓冲存储器内全部的读取和写入操作必须被同步以避免数据损坏，因此如果该高速缓冲存储器很大并且仅有一个Apache服务器的子进程能够同时与之通信，它本身可能变成性能的限制因素而不是解决的办法。例如，一个负载很重地服务器，正在运行大量的密集型的SSL密钥操作，将相应地引起所有的会话高速缓冲存储器操作，并增加了那些为了访问它的子进程排队等候的可能性，从而导致系统速度大大减慢。 由于我们将密钥操作看作同样的情况，在同一个Web服务器系统上运行SSL会话高速缓存服务是有缺点的。如果该高速缓冲存储器太大的，同步化法访问可能变成该瓶颈，否则它可能仅仅耗费太多的共享资源。如果高速缓冲存储器不是足够的大，它就会在即将期满之前很快被输入项目填满。当该会话高速缓冲存储器被写满时，将选择遵循已经设定的会话超时值（而在这样情况下新的SSL会话根本不会被高速缓存，直到在高速缓存内的输入项目超时），否则将导致过早地终止旧的会

话。令人遗憾地大多数SSL Apache服务器服务器按照前者的方式运行，但是两者也都不是特别理想。 Table 3- 使用通用SSL解决方案的高速缓存技术 可能的话将SSL会话高速缓存运作为一个专用服务运行在另一个系统上，于是它能够独立于那个Web服务器进行资源扩展和缩放管理。 1、https请求总是经由服务器验证并且选择性地被客户端验证。 2、任何时候，在SSL/TLS协议许可中任何一个的连接结束时，它要强制一个重新协商的过程。大多数服务器和浏览器不允许这么做（不会改变正在讨论的重点），所以我们将不会更深入的考虑这个问题。 3、 RSA现有的备选方案有DSA--基于证书和密钥的系统，但是这些技术不是如此广泛地受到支持。因此在最差的情况下，每个到服务器的连接都需要一个新的签名操作，这将大大地限制我们的Sparc机器的速度每秒处理小于27个连接。甚至假定那些机器在执行RSA签名操作以外，极少处理其他的浏览器的 https请求的应答工作。实际上HTTP1.1协议能够保持一个打开的连接，并使得后继的多重请求都在这个连接上进行通信（相继地而不是并发地进行）。不幸的是这个"keep alive"功能并不总是在服务器端允被许。 4、另外、我们发现浏览器为了下载WEB页面的内嵌图像或是这类文件，经常通过打开大量的到Web服务器的同时连接，以避免相继地执行每个请求而因此带来延迟。 5、Ben Laurie的Apache-SSL服务 , http://www.apache-ssl.org/ 6、Ralf Engelschall的mod_ssl, http://www.modssl.org/ 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com