SMPED：一种新型高性能服务器体系结构*)

任立勇　卢显良　韩　宏　侯孟书
（电子科技大学计算机学院　成都 610054）

摘　要　分析了几种服务器软件体系结构存在的优点与缺陷，设计了一种结合多进程并发与单进程事件驱动优点的对称式多进程事件驱动 SMPED 服务器体系结构。提出了一种新颖的连接调度算法，该算法根据服务器实际吞吐量进行连接分发和资源分配，试图实现服务器吞吐量最大化。仿真试验证明，SMPED 服务器性能在多种网络条件下明显优于现行服务器。

关键词　调度器，事件驱动，体系结构，服务器

SMPED: A New Architecture for High Performance Server

REN Li-Yong　LU Xian-Liang　HAN Hong　HOU Meng-Shu
(School of Computer Science & Engineering, UEST of China, Chengdu 610054)

Abstract　Soft architecture of severs have very important effects on their performance. In this paper, we introduce architectures of several prevalent servers and indicate their advantages & shortcomings. A new architecture for high performance server is proposed, which we call the symmetric multi-processes event-driven (SMPED) architecture. SMPED combines advantages of both multi-processes concurrent and single process event-driven. In SMPED, a novel schedule algorithm is introduced, which makes use of real throughput of each service process to distribute connection and resource. At last, we evaluate the performance of a Web server based on SMPED. The simulated results show that our SMPED is better than several servers on performance.

Keywords　Scheduler, Event-driven, Architecture, Server

1　引言

Internet 应用的爆炸式增长使得网络负载日益加重。尽管通信技术的发展大大提高了网络带宽，在一定程度上缓解了网络的拥挤状况，但指数级增长的网络访问请求却给网络服务器带来了前所未有的挑战。例如，AOL 的 Web 缓存每天服务超过 100 亿次点击。同时，HTTP 请求经常以爆发的方式到达 Web 服务器[1]。据统计，高峰时的 HTTP 请求率超过平均值的 8～10 倍，这时的负载一般超过 Web 服务器的负载能力，使其吞吐量下降，甚至停止服务。

提高服务器的性能主要有两种方法，一种是采用多 CPU 和高速 I/O 设备的高性能服务器，或者用多个服务器节点组成服务器群，由服务器群中的各个节点共同来提供同一种网络服务。另一种方法是改善服务器软件的服务模型，充分发挥服务器硬件尤其是 I/O 设备的并发处理能力，使服务器在高负载情况下仍能提供较高的吞吐量。本文主要专注于后者。

从服务器软件上提高服务器性能的方法目前分为三类：第一，修改操作系统内核或者系统调用实现[2~5]，由于操作系统种类繁多，因此这种方法不能通用。第二，在操作系统内核中实现 Web 服务器[6,7]，减少数据在内核与用户空间的拷贝次数，但这种方法不利于服务器软件的修改和维护。第三，设计新型通用服务器体系结构[8~12]，试图最大限度上以软件方式充分发挥 I/O 硬件的并发性。

本文首先分析了现有服务器体系结构存在的优缺点，设计了一种新型的服务器体系结构：对称式多进程事件驱动体系（Symmetric Multi-process Event-driven architecture, SMPED），SMPED 包括一个核心的调度进程和若干个服务进程。由调度进程接收连接请求，并根据各个服务进程的负载情况进行任务分派。仿真试验证明，采用该体系结构的 Web 服务器极大地提高了吞吐量，尤其是在过载情况下，仍能保持较高的吞吐量。

2　相关工作

为实现高性能服务器，最重要的是要采用特殊的技术解决大规模并发问题，即当大量用户同时访问服务器时，需要保证服务器在尽量不降低总吞吐量的情况下正常工作。

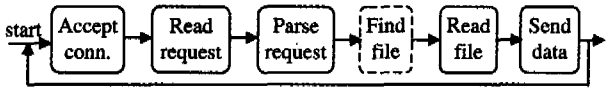

图 1　简化的 Web 服务器

如图 1 所示为简化的 Web 服务器流程图，按照顺序步骤执行的服务器几乎在每一步都有可能阻塞，从而无法并发地为多个客户提供网络服务。因此，为提高服务器吞吐量，需要将这些有可能阻塞的步骤进行复用。

改善上述服务模式最简单的方式是每个客户单独由一个进程（或线程）来提供服务，由于多进程（或多线程）能在操作系统中并发执行，并发执行的多个进程复用可能阻塞的 I/O 设备（如磁盘、网卡等），从而就实现了服务器能并发向多个客

*)本课题得到国防预研基金（51406070201DZ0211）资助，得到电子科技大学青年科技基金（YF020803）资助。**任立勇**　博士，副教授，主要研究方向为分布式系统，高性能服务器。**卢显良**　教授，博士生导师，研究方向为分布式操作系统和网络应用技术。**韩　宏**　博士，研究方向为高性能网络技术。**侯孟书**　博士，研究方向为分布式计算。

户提供服务的能力，提高了服务吞吐量。如图 2 所示，由于众所周知的 Fork 开销和进程调度开销，当并发客户数超过一定数量时，大量的进程不仅会耗费完所有的内存资源，而且频繁的进程调度也会急剧降低系统吞吐量，增加服务延迟。

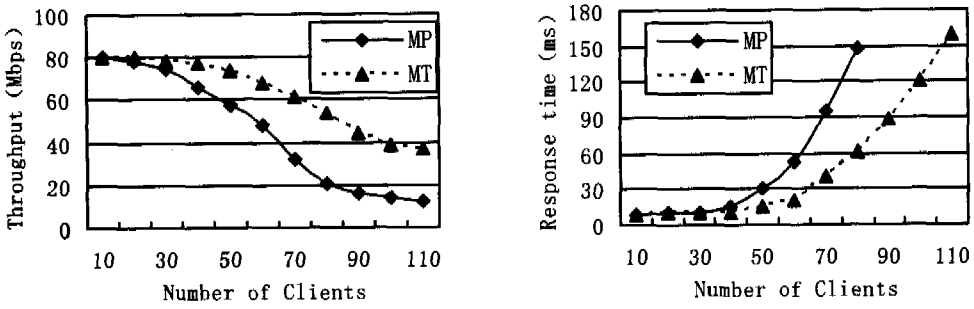

图 2　多进程(MP)服务器与多线程服务器(MT)吞吐量与时延比较

另一种提高服务器性能的方法是采用单进程事件驱动(single-process event-driven，SPED)[8]，SPED 不仅避免了服务器被单个网络 I/O 所阻塞，同时，事件驱动模式也保证了服务器能复用各种 I/O 设备，从而实现为多个客户服务的能力。但由于目前大多数操作系统并不支持非阻塞的磁盘 I/O 操作，因此只有当客户访问内容驻留在服务器主存中，SPED 的性能比较高，而一旦工作负载(workload)超过可用主存大小时，单进程的服务器将会被磁盘 I/O 阻塞，从而降低服务吞吐量。与此同时，由于系统调用 select 先天缺陷，从而导致 SPED 的可扩展性较差[3]。

为克服上述两种方法的缺点，Vivek S. Pai 等人提出了一种非对称的多进程事件驱动服务模型(asymmetric multi-process event-driven，AMPED)[9]。AMPED 由一个中心服务进程和若干个辅助进程(Helper)组成，中心服务进程以事件驱动方式工作，辅助进程主要负责完成磁盘 I/O，即读取磁盘文件。所有的网络 I/O 操作和解析请求都由中心服务器完成，一旦出现需要读取磁盘文件时，立即调用某个辅助进程读取文件并在读取完成后通知中心服务器。AMPED 既利用事件驱动复用了网络 I/O，也利用多进程方式复用了磁盘 I/O，因此其服务性能比传统的多进程(或多线程)服务器和 SPED 都有了显著的提高。但由于 AMPED 的中心服务器采用事件驱动模型，因此 SPED 存在的可扩展性差问题依然存在。

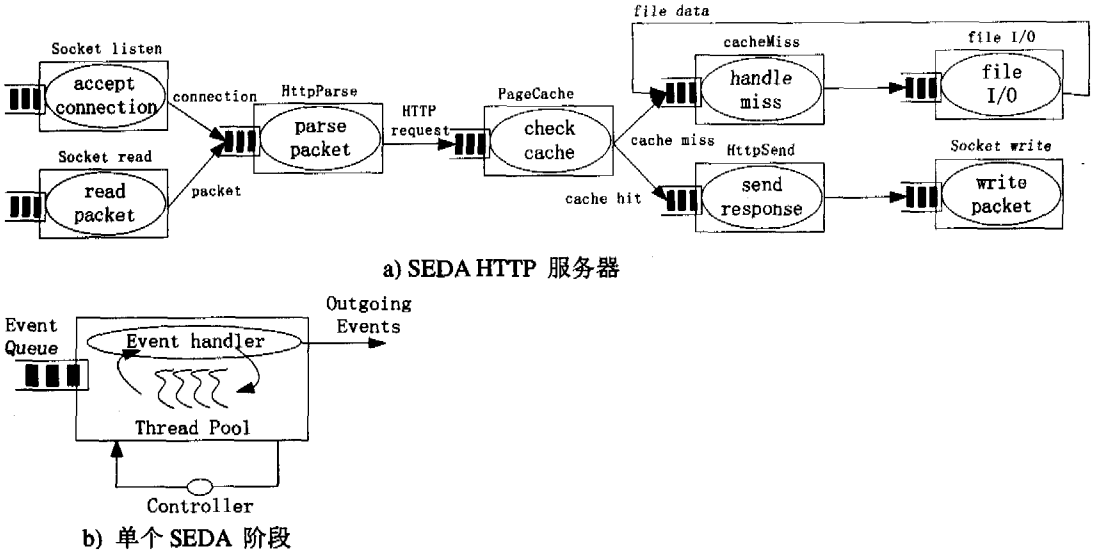

图 3　阶段式事件驱动体系结构(SEDA)

如图 3 所示，M. Welsh 等人提出了一种阶段式事件驱动体系结构(staged event-driven architecture，SEDA)[10]。SEDA 将 Web 服务器处理流程的每一步都用一个单独的阶段(Stage)来实现，阶段之间并发执行，并通过事件队列通信，从而实现复用各种 I/O 设备。每个阶段内部由一个输入事件队列、一个动态资源控制器和若干个线程组成，并通过事件分发器向各个线程分发需处理的各种事件，每个阶段处理后的事件传递给下一个阶段处理，资源控制器负责分配本阶段内资源及其调度。由于多个阶段并发处理 Web 服务中各个步骤，同时每个阶段内的多个线程并发执行本阶段需处理的 I/O 事件，因此，SEDA 最大限度地避免了 Web 服务过程中的 I/O 阻塞，其可扩展性极高。SEDA 流水线的工作模式要求阶段之间通过大量的 IPC 进行通信，尽管当服务器重载时，阶段式并发带来好处能忽略 IPC 带来的额外开销，但当服务器负载较轻时，IPC 通信的开销将会占整个服务开销的很大比重。因此，其服务性能反而不如 SPED 或 MT。另外，SEDA 复杂的编程模式也给 Web 服务器的实现带来很大的难度。

3　对称式多进程事件驱动体系结构：SMPED

如图 4 所示，本文提出了一种新型网络服务器软件体系结构：对称式多进程事件驱动体系 SMPED。SMPED 包括一个核心调度进程和一个由若干个服务进程组成的进程池。其中核心调度进程主要负责接收网络连接请求，并根据当前各服务进程的状态信息，将该客户连接分配给某个服务进程。同时，核心调度进程还负责根据当前服务器负载状况自适应调整服务进程数目。其目的是一方面避免过多的服务进程增

加系统开销，另一方面又要保证足够的服务进程满足并发服务需求。服务进程采用单进程事件驱动模型 SPED，主要完成向若干客户的数据请求做出处理并响应。

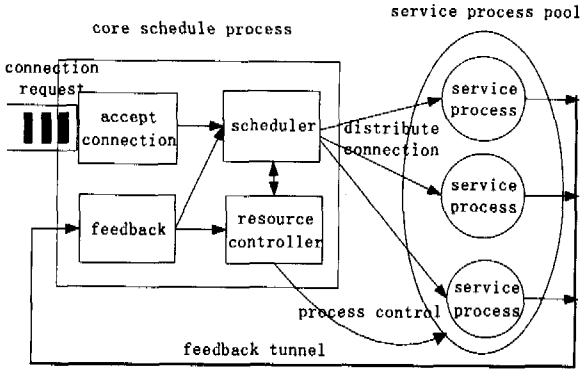

图 4　对称式多进程事件驱动体系，SMPED

3.1　核心调度进程

核心调度进程由四个模块组成，分别是：连接接收器、调度器、反馈处理器和资源控制器。四个模块可由一个进程来实现，也可以分别由四个不同的线程来实现。

连接接收器：负责接收客户的连接请求，并将建立的连接描述符传递给调度器。

反馈处理器：定期接收活动服务进程（即有连接分配的服务进程）的当前吞吐量，并计算其平均吞吐量。由于可能存在冷连接（cold connection）和长周期客户（long-lived），因此在计算服务进程平均吞吐量时，需要考虑其历史数据。我们采用指数加权滑动平均算法（EWMA）作为低通滤波器来计算平均吞吐量，如(1)式所示。

$$T_i \leftarrow \alpha T_i^{old} + (1-\alpha) T_i^{sample} \quad \alpha \in [0,1], i \in \{1,2,\cdots,n\} \quad (1)$$

式中 n 为当前活动服务进程个数，T_i 为第 i 个服务进程的平均吞吐量，初始化为 0，T_i^{old} 为上一个周期的平均吞吐量，T_i^{sample} 为本次周期第 i 个服务进程的吞吐量。平滑系数 α 的大小反映服务进程平均吞吐量受短时间吞吐量的影响程度。当 α 取值较大时，平均吞吐量不能较好反映服务进程当前的工作负载，反之，则不能反映其历史吞吐量。因此，本文选取 α 为 0.2，一方面能较准确地反映出当前工作负载，另一方面也能对处于冷连接状态的客户进行处罚。

另外，当某个服务进程服务的客户连接数降为 0 时，恢复 T_i 为初始值 0。

资源控制器：根据服务器的当前负载状况，动态调整进程池中服务进程数目。进程池中初始服务进程为 10 个，用户可配置其最大和最小进程个数。当资源控制器从反馈处理器收到某个服务进程连接数降为 0 消息后，若延迟一段时间后分配连接数仍为 0 则中止该服务进程运行，并释放其所占用的所有系统资源。当从调度器收到被调度服务进程连接数超过阈值时，则创建一个新的服务进程，并将该服务进程返回给调度器供调度器调度。

调度器：是核心调度器中最重要的模块。比较简单和通用的调度器通常根据各服务进程所承载的客户连接数进行负载均衡。由于实际网络环境中存在大量的冷连接和长周期客户，因此连接个数多少并不能反映服务进程真正负载状况。为此，本文采用各服务进程的吞吐量结合连接数作为调度依据。假设服务进程集合为 $\{P_1, P_2, \cdots, P_N\}$，各服务进程吞吐量向量为 $\{T_1, T_2, \cdots, T_N\}$，各服务进程承载连接数向量 $\{C_1, C_2, \cdots, C_N\}$。形式化的调度算法如下：

1、从连接接收器处接收新到达的网络连接。

2、随机选择一个连接数为 0 的服务进程，并把该新连接分配给这个进程，退出调度。

3、根据式(2)选择负载最小的服务进程。

$$P_i \leftarrow \left\{ P_k \mid \min\left(\frac{T_i}{\sum T_j} \cdot \frac{C_i}{\sum C_j}\right), 1 \leqslant k \leqslant N \right\} \quad (2)$$

4、如果选定的服务进程连接个数小于预先设定的阈值 C_{max}，则将该连接分配给此服务进程，退出调度。

5、如果选定的服务进程连接个数大于或等于预先设定的阈值 C_{max}，则判断当前进程池中服务进程个数是否达到预先设定的进程阈值 P_{max}，如果没有达到，则通知资源控制器创建新的服务进程，并将该连接分配给此服务进程，退出调度。

6、如果进程池中进程个数已达到阈值 P_{max}，则回到第 3 步，重新计算(2)式，选择下一个负载最小的进程。

经过上述调度算法分配连接后，能保证进程池中的进程个数能维持在设定范围之内，从而能有效地节约系统资源和降低进程调度的开销。与此同时，由于事件驱动的核心函数 select 固有的缺陷，当其处理的描述符增加到一定数目时，执行效率会急剧下降，为此，上述调度算法设定的连接阈值也能保证每个服务进程处理的连接数不会太大。

3.2　服务进程

服务进程池中包含着若干个专门向客户提供网络服务的服务进程，服务进程数目由核心调度进程中的资源控制器根据服务器的负载状况动态调整。为避免频繁创建和关闭服务进程造成系统开销增加，从而降低服务器吞吐量，我们为服务进程池中进程个数设定了上限 P_{max} 和下限 P_{min}。

服务进程采用单进程事件驱动 SPED 方式，完成向多个客户提供绝大多数网络服务工作（例如在 Web 服务器中，完成接收请求、解析请求、读文件或查询数据库、发送数据等）。除此以外，每个服务进程需要定期采集本进程在一个周期内的吞吐量，并通过 IPC 机制传递给反馈处理器。同时，当关闭某个连接时，也需要通知反馈定时器。

理论上，一个 SPED 服务进程通过非阻塞和事件驱动方式能并发处理多个客户的网络请求和处理。但由于目前绝大多数操作系统均不支持非阻塞的磁盘 I/O，因此可能会导致进程在处理某个客户连接的磁盘 I/O 时阻塞对其他客户的请求。实际上，由于我们采用了多个服务进程组成的进程池，它们可以复用各种 I/O 设备和 CPU，因此对整个服务器的吞吐量并没有太大的影响。

为进一步提高服务性能，当需要发送文件时，服务进程直接调用系统调用 sendfile，从而避免了两次数据拷贝工作。

4　性能测试

为测试 SMPED 体系的性能，我们实现了一个基于 SMPED 的 Web 服务器，并与流行的 Apache Web 服务器[13]和 Rice 大学实现的 Flash Web 服务器[9]进行了比较。Apache 服务器由 150 个进程组成固定大小的进程池，每个进程采用迭代的方式，在同一时间处理一个网络连接，即以阻塞 I/O 方式按顺序读取网络请求、读取磁盘文件、发送文件。Flash 服务器采用单进程事件驱动方式处理多个网络连接。同时由多个辅助进程执行可能发生阻塞的 I/O 操作，例如磁盘 I/O、路径解析等。而 SMPED 采用的是一种结合多进程和事件驱动的处理模型，并采用专用的调度算法在各服务进程间分配网络连接，试图最大化服务吞吐量。

我们的测试服务器运行在 Linux Redhat 7.2 下，硬件平台是 Pentium III 500，内存 256M，网卡是 100M/10M 自适应

网卡，通过百兆交换机与客户机组成一个 100M 网络。15 台配置各异的客户机器运行着我们自己编写的负载产生器。为更好模拟广域网环境，每台客户机由大量并发的线程模拟网络客户，这些网络客户按照一定比例分别模拟热连接和冷连接。其中热连接客户每隔 0～20 毫秒随机发起一次网络请求，然后读取应答，5 次 HTTP 请求后，关闭当前连接，然后重新建立连接。冷连接客户每隔 5 秒发送一次网络请求，然后读取应答，20 次 HTTP 请求后，关闭当前连接，然后重新建立连接。

首先，测试了三种服务器在过载时吞吐量的变化状况（本试验测试文件集为 32MB，保证所有文件均能在服务器系统缓冲区命中，无冷连接客户）。如图 5 所示，随着并发连接数的增加，三种服务器的吞吐量都有不同程度的下降。其中 Apache 服务器的吞吐量在大约 150 个客户连接时达到饱和，这主要是因为 Apache 采用 150 个进程的进程池作为服务进程，随后虽有所下降，但基本保持平稳。吞吐量下降的原因是因为进程调度开销增加和连接达到导致的中断增加，从而影响服务器吞吐量。由于采用了单进程事件驱动模型，Flash 服务器在客户连接较少时（小于 450 个），吞吐量是三个服务器中最大的，但随着客户连接继续增加，其吞吐量急剧下降，甚至在并发连接超过 800 时低于 Apache 的吞吐量，这是由于 Flash 使用的 Select 系统调用可扩展性较差造成的[2]（当我们将测试文件集增加到 256M 时，会有大量的磁盘 I/O，由于 Flash 中 helper 的辅助磁盘 I/O，上述情况会有所缓解，但 Flash 可扩展性的问题依然存在）。我们的 SMPED 服务器在客户连接小于 450 时，略低于 Flash，但远超过 Apache。当网络连接继续增加时，SMPED 服务器的吞吐量平稳下降，并没有出现 Flash 的情况。这主要因为 SMPED 服务器采用了合理的调度算法，利用事件驱动并发使用网络 I/O 设备，利用多进程模型并发使用 CPU 和磁盘设备。

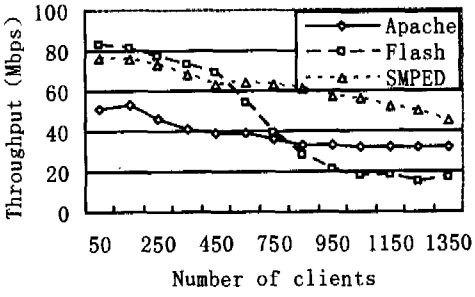

图 5　三种服务器吞吐量比较（无冷连接）

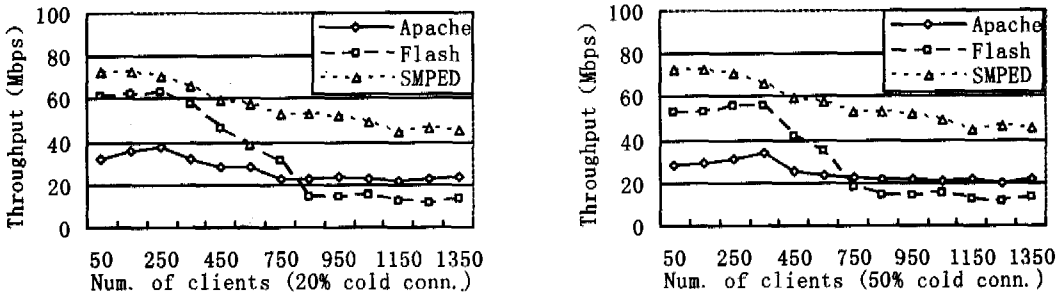

图 6　三种服务器吞吐量比较（有冷连接）

我们的下一个试验是测试当有冷连接客户存在时，三种服务器的吞吐量比较。如图 6 所示，由于有冷连接的存在，因此三种服务器的吞吐量均有不同程度的下降，而且达到饱和值时的客户连接数也比无冷连接时要大。非常有趣的是，尽管在客户连接总数较少时，Flash 服务器的吞吐量也比我们的 SMPED 要低，尤其是冷连接的比例增加时更为明显。与此同时，SMPED 服务器采用各服务进程的吞吐量作为主要的调度参数，因此对冷连接并不敏感。

结论　互联网的爆炸式增长给处在风暴中心的服务器带来了巨大的压力，而软件体系结构是决定服务器性能好坏的一个非常关键的因素。本文首先简单介绍了目前较为流行的几种服务器软件体系结构，分析其优点与存在的缺陷。然后提出了一种结合事件驱动与多进程模型的对称式多进程事件驱动体系结构 SMPED，重点研究了一种核心的连接调度算法，该算法能力求服务器吞吐量的最大化。最后，文章给出了一种基于 SMPED 的 Web 服务器的性能测试结果。

参 考 文 献

1　Mogul JC. Network behavior of a busy Web server and its clients. Technical Report, WRL 95/5, Palo Alto: DEC Western Research Laboratory, 1995

2　Banga G, Mogul J C. Scalable kernel performance for Internet servers under realistic loads. In: Proc. of the 1998 USENIX Annual Technical Conf. New Orleans, LA, 1998

3　Banga G, Mogul J C, Druschel P. A scalable and explicit event delivery mechanism for UNIX. In: Proc. of the 1999 USENIX Annual Technical Conference, Monterey, CA, June1999

4　Provos N, Lever C. Scalable network I/O in Linux. In: Proc. of the USENIX Annual Technical Conf. FREENIX Track, June 2000

5　Provos N, Lever C, Tweedie S. Analyzing the overload behavior of a simple Web server. In: Proc. of the Fourth Annual Linux Showcase and Conference, Oct. 2000

6　Lever C, Eriksen M, Molloy S. An analysis of the TUX Web server: [Technical report]. University of Michigan, CITI Technical Report 00-8, Nov. 2000

7　Brecht T, Ostrowski M. Exploring the performance of select-based Internet servers: [Technical Report]. HPL-2001-314, HP Labs, Nov. 2001

8　Zeus Technology Limited. Zeus Web Server. http://www.zeus.co.uk.

9　Pai V S, Druschel P, Zwaenepoel W. Flash: An efficient and portable Web server. In: Proc. of the USENIX Annual Technical Conf. Monterey, June 1999. 106～119

10　Welsh M, Culler D, Brewer E. SEDA: An Architecture for Well-Conditioned, Scalable Internet Services. In: Proc. of the Eighteenth Symposium on Operating Systems Principles (SOSP-18), Banff, Canada, Oct. 2001

11　姚念民，郑名扬，鞠九滨. 基于流水线的高性能 Web 服务器. 软件学报，2003，14(6)：1127～1133

12　Gribble S, Welsh M, von Behren R. The Ninja architecture for robust Internet-scale systems and services. Journal of Computer Networks, March 2001, 35(4)

13　Apache Software Foundation. The Apache Web server. http://www.apache.org