基于退火算法的动态负载均衡研究

孙峻文 周 良 丁秋林

（南京航空航天大学计算机科学与技术学院 南京 210016）

摘 要 分析现有服务器集群的动态负载平衡算法的特点和劣势，结合模拟退火算法和动态加权轮询算法的优势，提出一种动态负载平衡模型和解决方法。该模型使用模拟退火算法，自适应并且精确地确定性能权重向量，利用动态加权轮询算法，实时计算应用服务器负载，动态分配请求。通过该方法可以获得更合理的性能权重向量，有效地分配服务器负载，充分地利用服务器计算能力。实验结果表明，在负载逐渐提高的情况下，该算法保持了良好的平均响应时间和吞吐量，并且优于对比算法。

关键词 动态负载均衡，服务器集群，模拟退火，加权轮询

中图法分类号 TP393 **文献标识码** A

Research of Dynamic Load Balancing Based on Simulated Annealing Algorithm

SUN Jun-wen ZHOU Liang DING Qiu-lin

(College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China)

Abstract This paper analyzed characters and shortages of existing dynamic load balancing algorithms on server cluster, combined advantages of simulated annealing and dynamic weighted round-robin algorithm to propose a dynamic load balancing model and a solution. The solution uses simulated annealing algorithm to adaptively decide vector of performance weight and uses dynamic round-robin algorithm to balance load based on server load lively. It dynamically distributes requests based on calculated data. Our experiment shows that the dynamic load balancing algorithm effectively balances load and fully utilizes server resources in different load levels.

Keywords Dynamic load-balancing, Server cluster, Simulated annealing, Weighted round-robin

1 引言

随着计算机和 Internet 的高速发展，网络服务已经逐渐成为主要的商业手段，服务器承载的信息量和访问量成几何级数增长[1]，有效地利用服务器资源处理海量的访问请求是一个关键的问题，服务器集群成为一种能够提供高性能和高可靠性的解决方案。集群由多台互联的服务器组成，为 Internet 服务提供了数据冗余、服务能力冗余、内容分发等技术，多台服务器可以同时进行多个事务处理，提高了性能。集群中服务器数量众多，必然需要通过有效的负载均衡将用户的请求合理分配给这些服务器，其核心就是负载均衡算法，主要完成的工作是考虑每台服务器的负载水平的不同，合理、均衡地分配任务到各台服务器上，并实时地将负载重和出故障的服务器上的任务转移到负载轻的服务器上，从而平衡系统负载，提高系统的整体处理能力和服务质量。

2 现有负载均衡算法分析

现有的负载均衡算法可以分为静态负载均衡算法和动态负载均衡算法。静态负载均衡算法按照固定的方式分配请求，它并不考虑应用服务器的运行状态，只是按照特定的或者理想的方式进行分配，常见的有轮询(Round-Robin)和随机分配算法。静态负载均衡算法执行简单，但是由于条件理想化，因此静态负载均衡算法不能用于复杂的应用场景。动态负载均衡算法主要包括以实时监测性能的动态加权轮询(Weighted Round-Robin)[2]和组合优化(遗传算法和模拟退火算法)为基础的算法，以及相应的修改和改进算法。

文献[3]提出的动态反馈负载均衡算法，考虑了服务器的处理能力，同时实时计算服务器的负载，根据每台服务器当前负载比例值得出服务器动态的分配权值，按照分配权值加权轮询。该方法使用了负载向量和负载权值向量计算服务器负载，可以实现有效的动态负载平衡。但是该算法中的负载权值向量的基础权值是人为确定的，容易出现理想化的问题，由于各种服务器以及应用场景的不同，人工确定的权值难以保证正确合理。

文献[4]提出改进的动态遗传算法，由于服务器的性能不同，通过遗传算法，找到一组最优的分配方式，将负载分配至不同的服务器，其实验结果也证明，遗传算法得到的分配方案优于加权轮询算法。遗传算法具有很强的并行性，能够在较大的解空间内快速找到最优解，但其在进化后期收敛速度慢，易找到次优解，从而陷入局部最优。该文献中的动态遗传算

到稿日期：2012-07-30 返修日期：2012-10-26

孙峻文(1987－)，男，硕士生，主要研究方向为信息系统，E-mail：sunjw8888@gmail.com；周 良(1966－)，男，博士，副教授，硕士生导师，主要研究方向为信息系统、知识工程；丁秋林(1936－)，男，博士，教授，博士生导师，主要研究方向为信息系统、企业信息化。

法收敛后，得到了一个分配方案，整个系统便退化为静态的负载均衡方法，不能及时地对系统负载的变化做出反应。

现有算法主要力图解决负载均衡中一个方面的问题，动态反馈负载均衡算法解决了动态负载和反馈的问题。它没有解决在不同服务器性能和不应用场景下如何自适应地调整初始情况的问题；动态遗传算法则着重解决初始情况，但不能处理随着系统运行而出现的变化，缺乏适应性。综合以上分析，我们需要确定更加有效的负载因素，用以正确地反映出集群中计算的负载变化，避免使用随机和复杂的算法，减少人为因素的干扰，自适应地选择服务器。最终的目的是降低服务器集群系统的总体响应时间、提高吞吐量。本文将探讨基于退火算法的动态负载均衡算法，介绍该算法使用的负载因素和计算方式，讨论退火算法的原理和确定负载向量的方法，并通过实验验证动态负载均衡算法可以有效地反映系统的负载变化，证明该方案可以合理分配负载，达到更好的负载平衡效果，更有效地利用系统资源。

3 基于退火算法的动态负载均衡

基于上述算法的分析，考虑不同方案的优点和缺点，本文将结合组合优化和动态加权轮询算法，提出一种动态负载均衡的模型和解决方法。该动态负载均衡模型如图 1 所示。

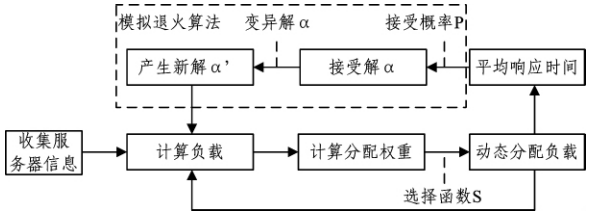

图 1 动态负载均衡模型

根据图 1 描述的模型，组合优化方法解决负载计算中权重向量 α 的确定问题，同时根据收集的服务器负载信息，基于模拟退火算法产生的权重向量 α，计算集群中服务器的分配权重，该权重会根据变化的服务器负载信息动态变化，选择函数 S 则能够根据历史信息和动态的分配权重，合理而均衡地分配负载。采用模拟退火算法解决组合优化问题，可避免人为确定权值的不准确问题，对于遗传算法可以解决的问题，模拟退火算法同样有能力解决，并且能保持较好的收敛速度。得到系统响应时间的反馈后，调整模拟退火算法的新解，并继续迭代，直至模拟退火算法达到的目标函数最优情况为止。该方法利用模拟退火算法在这种组合优化问题中的优势，结合动态加权轮询算法，使得即使在模拟退火算法收敛之后，仍可以保持整个负载平衡模型的动态性，真实地反映系统负载，更加合理地分配负载压力，发挥出集群系统的最大性能，避免文献[3,4]中单一算法带来的问题。后面几节，将仔细定义该动态负载均衡模型，并详细介绍模拟退火算法在模型中的应用以及动态分配负载的方法。

3.1 模型形式化定义

将图 1 的动态负载均衡模型描述为如下组合优化问题：有 N 个负载请求需要分配到 M 个应用服务器上进行处理，通过多个性能指标加权计算应用服务器的负载向量 L，目标是找到一个最优的权重向量 α，使负载向量 L 能够正确地反映集群系统的负载，应用选择函数选择服务器处理负载后，使整个任务处理的时间最短、吞吐量最大。由于不同的应用场景、不同的服务请求负载、异构的服务器系统，权重向量 α 各个值的选取对整个集群系统的性能发挥有较大的影响。例如对于 Web 应用服务器，CPU 使用率和内存使用率相对于磁盘读写量更加重要，而对于数据库服务器，磁盘读写量则是重要的性能指标。

用一个三元组表示该动态负载均衡模型：

$$resp=(L,R,S) \tag{1}$$

式中，$L=[L_1,L_2,L_3,\cdots,L_m]$ 表示 M 台服务器的动态负载向量，$R=[R_1,R_2,R_3,\cdots,R_n]$ 表示 N 个独立的任务请求，S 是一个选择函数，定义了 R 到 M 台服务器的映射关系。$resp$ 表示在 L、R 和 S 确定的情况下，集群系统的响应时间。为了计算 L，定义服务器性能向量 U_i 和性能的权重向量 α：

$$U_i=[U_i^{v1},U_i^{v2},U_i^{v3},\cdots,U_i^{vn}] \tag{2}$$

$$\alpha=[\alpha_1,\alpha_2,\alpha_3,\cdots,\alpha_n] \tag{3}$$

向量方向 U_i^v 表示收集到的一种硬件或者软件资源的使用量，且有 $U_i^v\in[0,1]$。对权重向量 α 有 $\alpha_i\in(0,1)\cap\sum\alpha=1$。由性能向量 U_i 和权重向量 α 可以计算出服务器负载向量 L，计算公式为：

$$L_i=U_i\cdot\alpha=\sum_{j=1}^{n}U_i^{vj}\cdot\alpha_j \tag{4}$$

计算服务器的负载后，通过函数 $w(L,i)$ 计算服务器分配权重向量 $W=[W_1,W_2,W_3,\cdots,W_n]$，其中 W_i 的计算式定义为：

$$W_i=w(L,i)=\frac{\min(L)}{L_i} \tag{5}$$

通过选择函数 $\lambda=S(r,h,W)$ 确定由服务器 $SERV_\lambda$ 处理负载，函数 S 根据当前的请求 r、负载分配历史向量 h 和服务器分配权重向量 W，选择一台合适的服务器 λ 来处理当前请求。最后，通过集群对负载的分配和处理，可以得到集群每次处理负载的响应时间。我们将组合优化问题的目标函数定义为集群系统在一次迭代实验中的平均响应时间：

$$obj(r)=\overline{\sum resp} \tag{6}$$

由以上定义，该组合优化问题的目标可以描述为：当服务器处理完成的负载与其能力最大限度匹配时，整个系统处于最佳运行状态，由此可以保证整个集群达到最佳性能。根据给出的目标函数，目标函数越小则集群系统的总体性能最好，也就是说，对于模拟退火算法 $\alpha'=SA(\alpha,resp,obj)$ 收敛时，目标函数得到最小值：

$$\begin{cases}\alpha_{best}=\lim SA(\alpha,resp,obj)\\ resp(L,R,S)=resp(U,\alpha_{best},R,S)=\min(obj)\end{cases} \tag{7}$$

本文将目标函数的优化作为寻优的目的，服务器的权重向量 α 作为寻优的目标，采用模拟退火算法对权重向量 α 进行寻优，待求的最优解是那些使得目标函数最小，也就是集群平均响应时间最小的权重向量。

3.2 权重向量的寻优

在上述动态负载均衡模型中，权重向量 α 的改变将直接影响最终计算出的分配权重向量 W，从而导致动态分配负载结果的变化，并且 α 向量有多个方向，要求和为 1，人为确定的值可能导致最终解和最优解的严重偏离。在模型中采用模拟退火算法确定 α 向量，退火算法是一种基于 Monte Carlo 方法的启发式随机搜索过程，用于求解复杂的组合优化问题的

极值。它所模拟的是热力学中经典粒子系统的降温过程：将固体加温至充分高，其内部粒子随温度上升变为无序状态，内能增大；之后再让固体徐徐冷却，其粒子渐趋有序，在每个温度都达到平衡态，最后在常温时达到基态，内能减为最小。模拟退火算法用于解决组合优化问题，它的最终收敛结果与初始状态无关，并且渐近收敛于全局最优解[5]。根据这一原理，该模型依照以下步骤执行模拟退火算法的迭代过程：

(1)产生新解：由一个产生函数从当前解 α 产生一个位于解空间的新解 α'，实现的方式是由当前解经过变换产生新解，对现有解的全部或部分元素进行置换、互换。产生新解的变换方法决定了解的邻域结构，也保证了模拟退火算法不会陷入局部最优解空间。

(2)计算新解的目标函数值：将新解 α' 作为计算负载向量 L 的参数，代入模型中执行动态负载均衡，收集集群系统的响应时间，得到目标函数在 α' 下的结果 $obj(r_m)$。

(3)接受新解：计算新解 α' 所对应的目标函数 $obj(r_m)$ 值与前一个解 α 的目标 $obj(r_{m-1})$ 函数值的差，判断新解是否被接受。依据 Metropolis 准则，该准则定义的新解的接受概率 P 为：

$$P=\begin{cases}1, & obj(r_m)\leqslant obj(r_{m-1})\\ \exp[\dfrac{obj(r_m)-obj(r_{m-1})}{t}], & obj(r_m)>obj(r_{m-1})\end{cases} \tag{8}$$

式中，$obj(r)$ 为模型的目标函数，t 表示当前温度。若 $obj(r_m)\leqslant obj(r_{m-1})$，也就是说在 α' 下，集群系统的平均响应时间更小，则接受 α'；反之，α' 使得系统平均响应时间变大，则根据公式，t 越高，接受概率 P 越大，越容易接受 α'。随着迭代的进行，t 逐渐降低，解将会逐渐收敛。

(4)降温过程：每轮迭代计算后，对 t 进行降温，以保证算法能够正常收敛。对于该动态负载均衡模型，我们定义第 m 轮迭代时，t 由以下公式确定，式中，t_0 是初始温度，取值为 10000：

$$t=\xi^{m-1}t_0(\xi\in[0.85,0.95]) \tag{9}$$

(5) 停止条件：为了让退火算法能够在合理的时间内停止，考虑到该模型的收敛速度，定义退火算法的停止条件是 $t<0.1$，也就是说，执行大约 225 次迭代后停止，输出当前的最佳权重向量 α。该停止条件可以使退火算法找到最佳的解，也能够保证其在合理的时间内停止。

上文中已经定义了目标函数 $obj(r_m)=\overline{resp_m}$，若新 α 向量带入模型后，在多轮负载分配测试中，使得服务器集群的平均响应时间降低，即 $obj(r_m)\leqslant obj(r_{m-1})$，则接受该 α；若新 α 向量增加了平均响应时间，则根据接受概率 P 决定是否接受该 α。最终经过多轮迭代，系统将收敛于一个最佳的 α 向量，使得该负载均衡系统能够准确地计算系统当前负载，合理地分配新到达的负载。

3.3 动态分配负载

当模拟退火算法产生新解并代入模型后，可以计算出服务器负载向量 L，进而计算出分配权重向量 W，需要一种方法来根据 W 选择一台服务器执行负载，同时应当考虑到之前的分配，做出合理且平稳的分配方案，为模拟退火算法后面的迭代过程提供信息。在模拟退火算法收敛后，将继续通过最优的权重向量 α 计算负载向量 L 和分配权重向量 W。文献[6]中提出使用随机概率空间的方法来选择服务器，我们认为通过随机函数会使得精心计算的负载均衡模型失去其准确度，本文采用更为稳定的饥饿度向量[7]作为选择函数的基础，该函数根据已经计算出来的数据，在众多服务器中选择一台来处理负载请求。我们希望请求的分配能够按照分配权重向量 W 进行，充分发挥处理能力较强的服务器的性能，那么对于应用服务器应当有：

$$\frac{h_{serv}}{totalreq}\approx\frac{W_{serv}}{\sum W} \tag{10}$$

式中，$totalreq$ 表示当前积累的总负载请求，h_{serv} 表示单台服务器的负载分配历史，也就是负载分配历史向量 h 中的一个方向。将式(10)作为负载平衡的目标，计算服务器的饥饿度向量 $hungry=[hungry_1,hungry_2,hungry_3,\cdots,hungry_n]$，其中，单一服务器的 $hungry_{serv}$ 定义如下：

$$hungry_{serv}=totalreq\times W_{serv}-h_{server}\times\sum W \tag{11}$$

基于饥饿度向量，将选择函数 $\lambda=S(r,h,W)$ 描述为以下形式：

$$\{\lambda=S(r,h,W)\mid hungry_\lambda=\max(hungry)\} \tag{12}$$

通过该选择函数，可以有效地按照计算出的分配权重向量向服务器分配负载，例如有两台服务器 A、B 组成的集群，它们的分配权重向量 $W=[70,30]$，那么根据该选择算法，最终的分配序列将是：ABAAABAABA，其中 70% 的负载交给服务器 A，30% 的负载交给服务器 B，从结果序列可以看出，该方法是一种改进的轮询方法，能够按照已经计算好的分配权值来分配负载，并且具有稳定的结果序列，参照分配历史向量，该选择函数可以动态而平滑地调整各个服务器之间的负载。

4 实验及结果分析

实验中，一共使用 6 台计算机来组成一个小型实验服务器集群，其中 1 台作为负载均衡服务器，1 台作为数据库服务器，另外 4 台作为应用服务器。负载均衡服务器运行一个基于 Apache HTTP Server 的负载均衡服务程序，分别实现了本文提出的基于退火算法的动态均衡方法以及文献[3]的动态反馈负载均衡算法和文献[4]的改进的动态遗传算法。数据库服务器进行了专门的设置，可以支持并发 1024 个连接，使其不成为测试中的瓶颈。4 台应用服务器运行 Apache Tomcat 服务，并且均部署了统一的 J2EE 测试应用，使用局域网络共享 Session。测试应用包含了 Internet 应用服务常见的页面生成、数据库访问和写入操作，代码来自于实际运行的系统。

实验中使用测试计算机进行 50、100、200、500 和 1000 个并发线程，向负载均衡服务器发送 HTTP POST 请求，一共发送 5000 次。在实验中，选择 CPU 使用率、内存使用率、磁盘 I/O 速率、网络 I/O 速率、页错误(Page fault)次数和 J2EE 测试应用运行正确率作为性能向量 U 的方向，即：

$$U=[U^{cpu},U^{mem},U^{dio},U^{nio},U^{pf},U^{cor}] \tag{13}$$

这几个方向可以充分体现服务器的性能和运行状态[8]。对于文献[3]的动态反馈负载均衡算法，根据其理论，选取倾向于 CPU 使用率和网络 I/O 速率的权重向量，即 $\alpha=[0.3,0.1,0.1,0.3,0.1,0.1]$。对于本文算法，权重向量 α 由模拟退火算法确定，最终收敛的权重向量如表 1 所列。

表 1　权重向量 α

性能向量方向	动态反馈算法	退火算法收敛
CPU 使用率	0.3	0.288
内存使用率	0.1	0.103
磁盘 I/O 速率	0.1	0.080
网络 I/O 速率	0.3	0.221
页错误	0.1	0.161
运行正确率	0.1	0.147

表 1 的数据显示，模拟退火算法也收敛于偏向 CPU 使用率和网络 I/O 速率的组合，但是相对于人为确定的值，页错误和测试系统运行正确率的权重更高，磁盘 I/O 速率的权重较低。这是因为系统缓存命中率低，导致了较多的页错误，测试系统有大量的连接线程在运行，导致系统运行不平稳。测试服务器不运行数据库，大多数据均在内存中运算，所以磁盘 I/O 速率对于整体系统的影响较小，模拟退火算法得到的结果中磁盘 I/O 速率权重比人工确定值更小，也体现了其实际的情况。模拟退火算法得出的权重向量 α 比较精确，为动态负载均衡的执行算法提供了更加有效的信息，后面的实验数据也验证了权重向量 α 的合理有效性。

实验集群系统的平均响应时间如图 2 所示。

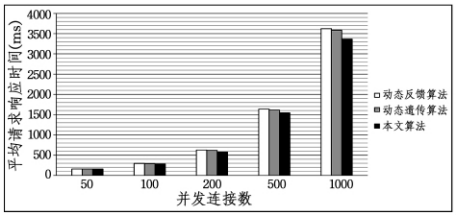

图 2　并发平均响应时间

图 2 中，横坐标表示测试时采用的并发连接数，分别是 50、100、200、500 和 1000；纵坐标表示集群的总体响应时间，单位为毫秒(ms)。

响应时间图像显示，在 50 和 100 个线程的较低并发连接情况下，应用服务器负载压力不大，3 种负载均衡算法下均可以得到很短的响应时间。随着并发连接数的提高，并发线程达到了 200、500 和 1000 时，应用服务器负载出现变化。动态反馈算法在总体负载较高的情况下，其算法对于高负载下的服务器性能区分不够精确，不能合理地将负载分配到实际负载较轻的应用服务器上，导致高负载的应用服务器更加不堪重负，集群整体性能下降。动态遗传算法在低负载下，通过有效的权重分配方案可以达到较好的分配效果。在高负载时，由于静态的分配方案不能够自适应调整，也会导致较多的请求堆积在较慢的服务器上，总体响应时间提高，不能充分发挥集群系统的性能。

本文基于退火算法的动态负载均衡算法的在实验中结合了前两种算法的优势，模拟退火算法得到的分配权重向量，合理且精确。同时，动态负载分配算法利用了动态反馈算法的特点，在运行中微调分配方案，由于有准确的权重向量 α，使得动态分配算法运行于一组合理的数据上，因此比较清晰地展现了不同服务器的运行负载，有效地将新的负载分配到压力较低的应用服务器上。在 500 和 1000 个线程的高负载情况下，本文算法仍保持了相对较低的系统响应时间，更好地利用了集群系统的性能。

实验中服务器集群的平均吞吐量结果如图 3 所示。

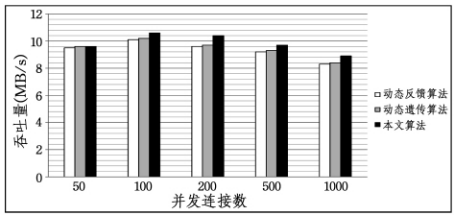

图 3　并发平均吞吐量

图 3 中，横坐标表示测试时采用的并发连接数，分别是 50、100、200、500 和 1000；纵坐标表示集群的吞吐量，单位为 MB/s。

吞吐量数据图像显示在并发数量较低的情况下，由于实验用服务器性能较强，3 种算法难以拉开差距。当并发连接数量逐渐增大，本文算法体现出一定的优势，当达到 200 个并发线程时，已经达到测试服务器集群的承载能力的上限附近，对比的两种算法已经达到其处理能力的极限，开始出现性能下降。在 500 和 1000 线程并发的高负载情况下，3 种测试算法的吞吐量均出现了不同程度的下降。动态反馈算法的权重向量不精确，所以高负载下计算出的分配方案离最优的方案差距较大。动态遗传算法的分配方案一开始比较有效，但负载增大后，由于不能动态调整导致了性能的下降。本文算法则保持了高于对比算法的吞吐量，吞吐量测试的结果也证明了本文基于模拟退火的动态负载均衡算法结合了合理的权重向量 α 和高效的动态分配算法，反映了集群系统中应用服务器的负载情况，有效地平衡了负载，其实际表现优于对比的算法。

结束语　本文分析了现有的几种动态负载均衡算法及它们的优缺点，提出一种基于退火算法的动态负载均衡算法。退火算法解决了权重向量的确定问题，而动态的负载计算则能够较准确地反映当前的系统负载。该方法结合了多种方法的优势，能合理地分配请求，平衡负载，测试也表明其准确性和有效性，对于服务器集群系统的负载均衡，有一定的应用价值。但本文的算法还有不足之处，如模拟退火算法在负载均衡的应用中，收敛特性还需要进一步研究，在今后的研究中需要不断对该算法进行改进和完善。

参 考 文 献

[1] Labovitz C, Iekel-Johnson S, McPherson D, et al. Internet inter-domain traffic[C]// ACM SIGCOMM Conference. 2010

[2] Choi D J, Chung K S, Shon J G. An Improvement on the Weighted Least-Connection Scheduling Algorithm for Load Balancing in Web Cluster Systems[J]. Grid and Distributed Computing, Control and Automation, 2010, 121: 127-134

[3] Tong R X, Zhu X F. A Load Balancing Strategy Based on the Combination of Static and Dynamic[C]// Database Technology and Applications. 2010 2nd International Workshop. Nov. 2010

[4] 杨锦，李肯立，吴帆. 异构分布式系统的负载均衡调度算法[J]. 计算机工程，2012，38(2)：166-168

[5] Bandyopadhyay S, Saha S, Maulik U, et al. A Simulated Annealing-Based Multiobjective Optimization Algorithm: AMOSA [J]. Evolutionary Computation, IEEE Transactions, 2008, 12 (3): 269-283

[6] 刘松，钟珞. 一种基于服务类型的 Web 集群负载均衡算法[J]. 武汉理工大学学报，2009，19：134-136

[7] Sun J Y, Li H F, Han Y B. An Adaptive Scheduler for Enhancing the Efficiency of Multi-engine BPM Systems[C]// Parallel and Distributed Processing with Applications. IEEE International Symposium 2009. 2009: 606-610

[8] Fujii T, Dohi T. Statistical failure analysis of a Web server system[C]// Availability, Reliability and Security. 2009