Journal of Southeast University (English Edition) Vol. 23, No. 3, pp. 443 – 446 Sept. 2007 ISSN 1003—7985

Emergency algorithm for enhanced survivability of key service

Zhao Guosheng[1,2] Wang Huiqiang[1] Wang Jian[1]

([1]College of Computer Science and Technology, Harbin Engineering University, Harbin 150001, China)

([2]Center of Network and Information, Harbin Normal University, Harbin 150001, China)

Abstract: To work out a solution for answering the question when it is suitable for emergency response strategies to be executed, an emergency algorithm for enhanced survivability of a key service is presented. First, based on the central limit theorem and the hypothesis testing theory, the confidence interval of each key service's history average service response time in the host server and the spare server can be figured out, respectively. This can also be updated dynamically by a proposed method using the method of time slide window. Then, according to the five kinds of distributed situations of the current service response time's confidence interval in the host server and the spare server, the proposed algorithm can dynamically choose the appropriate emergency policies such as resource reconfiguration, service degradation or service drifting, etc. in right time. Thus, the key service request can be finished within its expected deadline by users as far as possible. Furthermore, the whole process of dynamic configuration is transparent to users. Finally, simulation tests are performed to prove the feasibility.

Key words: emergency response; survivability; key service; confidence interval

At present, survivability has been a new research direction in network security technology. The definitions of survivability have been introduced by previous researchers[1-2], who define survivability as the capability of a system to fulfill its mission, in a timely manner, in the presence of attacks, failures, or accidents.

There are currently two primary angles in the realization technology of enhanced survivability. One is the redesign angle, whose research results are mainly manifested in software engineering methods[3-5]. Such designs lack the actual applications, and their costs are extremely high. So they only rely on a theory groping stage. Another is the structure optimized angle, whose research results are mainly manifested in the optimized structure of invasion tolerance systems[6-9]. By combining fault-tolerant, tolerant invasion and reliability technology to the existing system, the system reconfiguration ability or redundancy ability can be enhanced for the enhancement of system survivability. Such method costs are lower and the feasibility is good.

1 Algorithm Representation

1.1 Correlative definitions

Definition 1 The provided services are a set of 2-tuple $S=\{S_{\text{E-S}}, S_{\text{N-S}}\}$, where $S_{\text{E-S}}=\{S_1, S_2, \ldots, S_n\}$ is the set of key services, $S_{\text{N-S}}=\{S_{n+1}, S_{n+2}, \ldots, S_m\}$ is the set of non-key services.

The goal of enhancing key service survivability is to ensure that a service in $S_{\text{E-S}}$ can continually fulfill its mission, in a timely manner, in the presence of attacks, failures, or accidents.

Definition 2 For $\forall S_i \in S_{\text{E-S}}$, let $S_i=\{\langle q_i^1, t_i^1\rangle, \langle q_i^2, t_i^2\rangle, \ldots, \langle q_i^h, t_i^h\rangle\}$, where q_i^j denotes service grade, and $q_i^1 < q_i^2 < \ldots < q_i^h$, t_i^j represents the service response time, and $t_i^j=\{t_i^j(1), t_i^j(2), \ldots, t_i^j(k)\}$, in which $t_i^j(m) \in [\underline{t_i^j(m)}, \overline{t_i^j(m)}]$ denotes the confidence interval of response time for providing q_i^j grade's S_i service in the P_m server.

Definition 3 Suppose that the number of servers is k, let $\Omega=\{P_1, P_2, \ldots, P_k\}$ be the set of the servers, where $P_i=(\Delta_i, C_i)$, $\Delta_i=\{\Delta_i^1, \Delta_i^2, \ldots, \Delta_i^h\}$ denotes h key services provided by the P_i server, and $\sum_{i=1}^{k}\Delta_i = S_{\text{E-S}}$, namely, $\exists S_i \in S_{\text{E-S}}, P_i.\Delta_i = P_j.\Delta_j$; C_i denotes the load parameter of the P_i server.

Definition 4 The mapping table of key service requests is a set of θ. Since $\forall \theta_i \in \theta$, θ_i is a set of 7-tuple, namely, $\theta_i=(S_i, d, \bar{t}, p_a, t_a, p_b, t_b)$, where S_i, d and $\bar{t}$ denote the id number of service, service grade and requested service deadline, respectively; p_a and p_b denote id numbers of the host server and the spare server, respectively; t_a denotes the execution time of S_i

Received 2007-05-18.

Foundation items: Specialized Research Fund for the Doctoral Program of Higher Education (No. 20050217007), the Weaponry Equipment Foundation of PLA Equipment Ministry (No. 51416060104CB0101).

Biographies: Zhao Guosheng(1977—), male, graduate; Wang Huiqiang (corresponding author), male, doctor, professor, wanghuiqiang@hrbeu.edu.cn.

service in the host server p_{a}, t_{b} denotes the running time of S_i service in the spare server p_{b}, the values of t_{a} and t_{b} are zero at starting, which can be modified every one unit of time.

Lemma 1 Any key service request θ_i can still continue to be run and finished on time even when it has failed in a server, iff the service requested by θ_i exists in different servers, and the sum of service response times is smaller than $\theta_i.\bar{t}$. The formalized description is $\forall \theta_i \in \theta, \theta_i.p_a \neq \theta_i.p_b$, and $\theta_i.t_a + \theta_i.t_b \leqslant \theta_i.\bar{t}$.

1.2 Interval estimation of service response time

The central limit theorem thinks that the distribution of the sample mean value approaches a normal distribution regardless of the distribution type of the statistical total, and the mean value of the normal distribution is equal to the mean value of the total distribution; the variance of normal distribution is equal to the variance of the total distribution dividing the size of sample. Therefore, the response time of one key service request can be estimated according to the mean response time $\overline{T}_{n+1}$ of the statistical total. The expression is

$$\overline{T}_{n+1} = \frac{T_{n+1}}{n+1} \tag{1}$$

where $T_{n+1} = T_n + t_{n+1}$, T_n denotes the accumulation sum of the sample mean response times in n time units ($T_0 = 0$), t_{n+1} is the sample mean response time in the $n+1$ time unit.

For $n+1$ data, the unbiased estimate between the standard difference of the sample and that of the statistical total is

$$S_{n+1} = \sqrt{\frac{1}{n}\left[\sum_{i=1}^{n+1}(t_i - \overline{T}_{n+1})^2\right]} = \sqrt{\frac{\sum_{i=1}^{n+1} t_i^2 - (n+1)\overline{T}_{n+1}^2}{n}} \tag{2}$$

The standard variance of the sample mean value whose value is $S_{\bar{X}_n} = S_n/\sqrt{n}$ can construct a confidence interval for the total mean value. Therefore, the confidence interval of the total sample mean value μ can be constructed as

$$\overline{T}_n - z\frac{S_n}{\sqrt{n}} \leqslant \mu \leqslant \overline{T}_n + z\frac{S_n}{\sqrt{n}} \tag{3}$$

where z is a quartile of the standard normal distribution. If the service response time of the key service accords with Eq. (3), it shows that the current service behavior is normal, or that it is abnormal.

1.3 Confidence interval's real-time update

For $\forall S_i \in S_{\text{E-S}}$, the network system maintains a history sliding window, whose duration is n time units. The sliding window model is a two-tuple (T_i, n_i), where n_i denotes the requested number of S_i services in the i-th time unit, T_i denotes the response time's sum of n_i service requests in the i-th time unit. We thus obtain the accumulation sum of the history window: $sT_{\mathrm{h}} = \sum_{i=1}^{n} T_i$, $sn_{\mathrm{h}} = \sum_{i=1}^{n} n_i$. The head index points to the head of the queue; the tail index points to the tail of the queue. After a new time unit passes, the accumulative renewal is shown as

$$\left.\begin{array}{l} sT_{\mathrm{h}} \leftarrow sT_{\mathrm{h}} - T_{\text{head}} + T_{\text{head}+n} \\ sn_{\mathrm{h}} \leftarrow sn_{\mathrm{h}} - n_{\text{head}} + n_{\text{head}+n} \end{array}\right\} \tag{4}$$

Simultaneously, head $\leftarrow$ head + 1, tail $\leftarrow$ tail + 1, thus the new mean value and the standard variance can be obtained as

$$s\overline{T}_{\mathrm{h}} = \frac{sT_{\mathrm{h}}}{sn_{\mathrm{h}}},\quad ss_{\mathrm{h}} = \sqrt{\frac{1}{sn_{\mathrm{h}} - 1}\left[\sum_{i=1}^{sn_{\mathrm{h}}}\left(\frac{T_i}{n_i} - s\overline{T}_{\mathrm{h}}\right)^2\right]} \tag{5}$$

It is necessary to explain that if the current response time of a service request is not in its confidence interval, the data in the history window will not be updated.

2 Realization of Emergency Algorithm

For $\forall \theta_i$, we assume that it has taken t_{a} time to execute the S_i service in p_{a}, but the S_i service has not yet been finished, and its service response time confidence interval is $[\alpha, \beta]$ and $[\alpha', \beta']$, respectively, in the host server p_{a} and the spare server p_b. According to the distribution of $[\alpha, \beta]$ and $[\bar{t} - \beta', \bar{t} - \alpha']$, we may divide two cases as follows.

Case 1 As shown in Fig. 1, if $t_{\mathrm{a}} < \beta$, then the S_i service continues running to β; if $\beta \leqslant t_{\mathrm{a}} < \bar{t} - \alpha'$, then the S_i service will deprive the resources owned by other non-key services, and then continues running to $\bar{t} - \alpha'$; if $t_{\mathrm{a}} \geqslant \bar{t} - \alpha'$, then all the processes correlated to the S_i service in p_{a} will be stopped, and the resources owned by S_i will be released to the system; if the S_i service is finished in $[\bar{t} - \beta', \bar{t} - \alpha']$, then the corresponding service in the spare server p_{b} will be stopped at once, and the owned resources will also be released to the net-

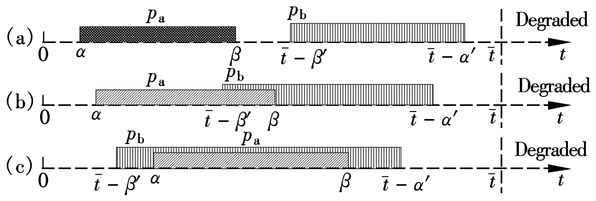

Fig. 1 Interval distribution of case 1. (a) $\beta \leqslant \bar{t} - \beta'$; (b) $\alpha \leqslant \bar{t} - \beta' \leqslant \beta$; (c) $\bar{t} - \beta' \leqslant \alpha$ && $\beta \leqslant \bar{t} - \alpha'$

work system.

Case 2 As shown in Fig. 2, if $t_a < \beta$, then the S_i service continues running to β; if $\beta \leqslant t_a < \bar{t}$, then the S_i service will deprive the resources owned by other non-key services, and then continues running to $\bar{t}$; if the S_i service is finished in $[\bar{t} - \beta', \bar{t} - \alpha']$, then the corresponding service in the spare server p_b will be stopped at once, and the owned resources will also be released to the system.

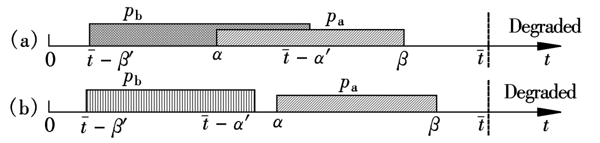

Fig. 2 Interval distribution of case 2. (a) $\bar{t}-\beta' < \alpha \leqslant \bar{t}-\alpha'$; (b) $\bar{t}-\alpha' \leqslant \alpha$

Steps of the algorithm are as follows:

```
Input: $\forall \theta_i \in \theta$, the service response time confidential interval of $\theta_i.S_i$ is $[\alpha,\beta]$ and $[\alpha',\beta']$ in the host server $p_a$ and the spare server $p_b$, respectively.
Output: Success or fail.
Step 1 While ($\theta_i.S_i$ has not been finished) {
    if ($S_i$ has not been finished in $\bar{t}-\beta'$) then
        $S_i$ in $p_b$ can be active to continue to run from a recent checkpoint;
    if ($(\bar{t}-\beta' \geqslant \beta) \| (\alpha \leqslant \bar{t}-\beta' \leqslant \beta) \| (\bar{t}-\beta' \leqslant \alpha \ \&\&\beta < \bar{t}-\alpha')$) then
        apply the strategies of case 1 to configure;
    if ($(\bar{t}-\beta' < \alpha \leqslant \bar{t}-\alpha') \| (\bar{t}-\alpha' < \alpha)$) then
        apply the strategies of case 2 to configure;
    if ($S_i$ has not been finished in $p_a$ and $p_b$ in $\bar{t}$) then
        if ($\theta_i.d \geqslant 0$) then
            {$\theta_i.d = \theta_i.d - 1$; //$S_i$ need to be degraded.
            according to the current server load parameters $C_i$, choose a new $p_a$ and $p_b$ again;
            let $\theta_i.t_a$ and $\theta_i.t_b$ be zero, the value of $\theta_i.\bar{t}$ can be obtained through consulting with user;
            jumps to step 1; }
    else
            return fail; }
Step 2 Delete the $\theta_i$ item in the mapping table;
Step 3 Return success.
```

3 Simulation

By virtue of C language simulation, we used the gprof procedure in Linux to analyze the finished rate of the HTTP key service request. We compared the finished rates of the proposed algorithm (EA) with that of the polling algorithm (PA).

In Fig. 3, when the network load is smaller ($\leqslant 0.6$), the key service's finished rate of the PA algorithm (PA-KS) nearly approaches 1, but when the network load is higher ($\geqslant 0.7$) and the resources which are necessary for the completion of key service are insufficient, the key service's finished rate of the PA algorithm proportionally drops. However, in the same conditions, the key service's finished rate of the EA algorithm (EA-KS) is relatively better even when the network load is higher causing it to drop a little, which effectively realizes the guarantee of key service survivability, but the introduced mechanisms also increase the failure rate of non-key services. When the network load is between 0.55 and 0.75, the resource deprivation frequently occurs, which causes the non-key service's failure rate of the EA algorithm (EA-NKS) to be relatively higher than that of the PA algorithm (PA-NKS); but when the network load is between 0.75 and 0.90, the incremental non-key service's failure rate of the EA algorithm is lower than that of the PA, which is related to the way of the pre-choosing server based on the current network load.

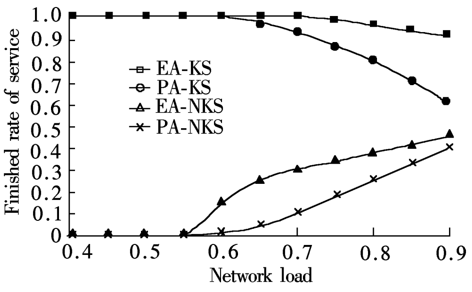

Fig. 3 The finished rate of key services

4 Conclusion

The proposed algorithm works out a solution for answering the question when it is suitable for emergency response strategies to be executed. This has not been fully considered by the existing literatures. Consequently, it provides an important priori-foundation for the next step of research in emergency response technology based on survivability strategy. But the presupposition of the proposed algorithm is a bit rigorous, and the introduction of deprivation mechanisms should also consider a certain limitation, otherwise, it may cause additional expense and network instability.

References

[1] Westmark V R. A definition for information system survivability [C]//*Proc of the 37th International Conference on System Sciences*. Washington, DC: IEEE Computer Society Press, 2004: 2086 – 2096.

[2] Ellison R J, Fisher D A, Linger R C, et al. Survivable network systems: an emerging discipline[R]. Software Engineering Institute of Carnegie Mellon University, 1997.

[3] Linger R C, Mead N R, Lipson H F. Requirements definition for survivable network systems [C]//*Proc of the Third International Conference on Requirements Engineering*. Los Alamitos, CA, USA, 1998: 14 – 23.

[4] Richard C L, Howard F L, John M, et al. Life-cycle models for survivable systems, CMU/SEI-2002-ESC-TR-026 [R]. Boston: Carnegie Mellon University, 2002.

[5] Zinky J A, Bakken D E, Schantz R. Architectural support for quality of service for CORBA objects [J]. *Theory and Practice of Object Systems*, 1997, **3**(1): 55 - 73.

[6] Liu Peng. Architectures for intrusion tolerant database systems [C]//*Proc of the Foundations of Intrusion Tolerant Systems* (*OASIS'* 03). Las Vegas, Nevada, 2003: 3 - 13.

[7] Wang F, Gong F, Jou F, et al. SITAR: a scalable intrusion tolerance architecture for distributed services[C]//*Proc of the* 2001 *IEEE Workshop on Information Assurance and Security*. New York: IEEE Press, 2001: 38 - 45.

[8] Knight J C, Heimbigner D, Wolf A, et al. The willow architecture: comprehensive survivability for large-scale distributed applications [C]//*Proc of the International Conference on Dependable Systems and Networks*. New York: IEEE Press, 2002: 54 - 74.

[9] Tally G, Whitmore B, Sames D. Intrusion tolerant distributed object systems: project summary[C]//*Proc of the DARPA Information Survivability Conference and Exposition*. Washington, DC: IEEE Computer Society Press, 2003: 149 - 151.

一种增强关键服务可生存性的应急算法

赵国生[1,2] 王慧强[1] 王 健[1]

([1] 哈尔滨工程大学计算机科学与技术学院,哈尔滨 150001)

([2] 哈尔滨师范大学网络中心,哈尔滨 150001)

摘要:为了解决“何时对应急服务进行响应是合适的?”这一问题,提出了一种增强关键服务可生存性的应急算法. 首先,基于中心极限定理和假设检验理论,分别计算求得各关键服务在主、备服务器上的历史平均服务响应时间的置信区间,并且提出了一种基于时间滑动窗口的置信区间动态更新的方法;然后,根据主、备服务器上服务响应时间置信区间的 5 种分布情况,提出的算法可以为应急服务在不同的时间响应区选择合适的应急策略,如资源重配、服务降级或者服务迁移等,以此保证关键服务请求能够在用户期望的服务截止期内完成,且整个配置过程对用户透明. 最后,仿真试验表明了该算法的可行性.

关键词:应急响应;生存性;关键服务;置信区间

中图分类号:TP393.08