文章编号:1671-4598(2015)01-0046-04　　中图分类号:TP319　　文献标识码:A

基于故障矩阵的贝叶斯故障定位方法

张　会
(攀枝花学院 数学与计算机学院,四川 攀枝花　617000)

摘要:故障的准确诊断和定位是云计算系统提供持续服务的前提条件;为了提高系统故障诊断和定位的性能,文章提出了一种基于故障矩阵的贝叶斯故障定位方法;首先,对云计算系统的软件结构进行了抽象,对事物进行了定义,并描述了事务的执行路径;其次,将系统运行的多个执行路径表示为故障矩阵,并给出了组件健康状态的逻辑命题表达式;最后,应用贝叶斯概率分析了系统故障的概率;实验表明,文章提出的方法与其它相关方法相比,故障识别的准确性更高,所用的执行时间更短。

关键词:分布式系统;故障定位;推理;故障矩阵

Fault Matrix Based Bayesian Fault Localization Approach

Zhang Hui
(School of Mathematics and Computer Science, Panzhihua University, Panzhihua　617000, China)

Abstract: Accurate diagnosis and localization of fault is the premise for providing continuous services in Cloud systems. In order to improve the performance of fault diagnosis and localization, this paper proposed a fault matrix based Bayesian fault localization approach. Firstly, we abstracted the software architecture of Cloud computing system, defined the concept of transaction, and described the execution path of transaction. Secondly, we represented multiple execution paths as fault matrix, and defined the logic proposition of health for components. Finally, we applied the Bayesian probability to analyze the probabilities of system faults. The experiments show that, compared with other related works, the proposed approach has higher accuracy and less execution time for fault recognition.

Keywords: distributed system; fault localization; reasoning; fault matrix

0　引言

随着云计算系统规模的不断增大,其发生软件和硬件故障的概率也随之增大,对系统的故障进行自动诊断和修复的需求也越来越迫切[1]。自动计算将软件系统的正常运行建模成闭环的控制系统,对系统正常运行时的状态进行监测,并对监测结果和系统的预期结果(模型)进行对比[2]。当监测结果偏离预期的运行结果时,认定系统发生故障,并采取相应的修复措施[3-4]。分布式系统进行自动计算时采用的故障诊断方法通常包括3种。第一种故障诊断方法是软件修复[5-8]。第二种故障诊断方法是针对某一类系统或者某一类故障开发特定目的的诊断机制[9-11]。第三种故障诊断方法是对每个修复处理模块分配不同的诊断任务[12-14]。Rainbow 系统[13]中包含一系列修复措施,系统正常运行过程中某些系统结构常量发生变化时,触发相应的修复措施。

本文对云计算系统的软件结构进行了抽象,给出了事务以及事务执行路径的定义,并将事务的执行路径编码成组件的包含向量,最后基于贝叶斯概率对故障矩阵进行分析进行故障的识别和定位。

1　系统描述

本节首先对基于 Web 服务的云计算系统的结构进行了抽象,然后定义了典型的云计算事务,最后从服务执行路径的角度给出了系统状态的检测。

收稿日期:2014-08-07;　**修回日期**:2014-09-18。
基金项目:四川省教育厅基金资助项目(14ZB0407)。
作者简介:张　会(1980-),女,四川射洪人,硕士,讲师,主要从事计算机应用,计算机网络故障检测方向的研究。

1.1　系统结构

在 Web 服务中,客户端通常采用网页浏览器的方式访问云计算系统提供的服务。当浏览器向服务器发送服务请求时,采用的协议主要是 Http 协议。服务器在接收到客户端发来的服务请求时,进行本地处理或者对数据库服务器进行数据访问,并将处理结果返回给客户端。当服务器的规模和客户端的请求数量巨大时,通常在客户端和服务器集群之间加入若干分配器。分配器通过适当的路由规则将客户端的请求发送到不同的服务器,从而实现服务器端的负载均衡和客户端的及时响应等。本文对上述云计算系统的结构进行了抽象,其结构如图1所示。

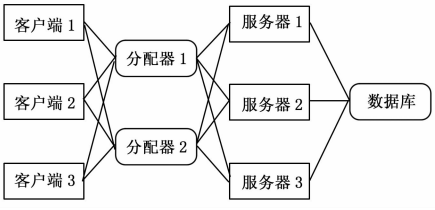

图1　云计算系统结构图

1.2　事务定义

当用户通过浏览器向服务器发送 Http 服务请求时,系统中一个成功返回结果的 Http 响应可以看作一个事务。事务所涉及的系统组件有客户端、分配器、服务器和数据库,这些组件之间的消息传输序列如图2所示。在云计算系统中,数据通常存储在专门的数据库中,该数据库往往采用副本或纠删码等措施从而保证数据的可用性。所以,本文假设系统的数据库是可靠的,即当服务器通过 SQL 查询语句向数据库进行数据请求时总能在合理的时间内得到查询结果。

一个成功的事务包含图2所示的6个消息传递过程,并且

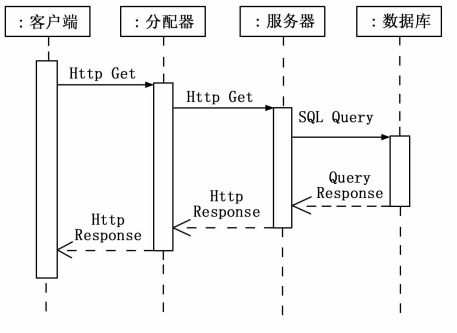

图 2　Http 请求事务的消息序列

每一步消息传递都在合理的时间范围内。由于客户端设备不能被云服务系统所控制，因此客户端的故障不在本文的考虑范围之内。在数据库可靠性的假设前提下，云服务系统在处理客户端的 Http 请求时，系统的故障发生在分配器和服务器上。

1.3　执行路径检测

事务定义的是云服务系统在概念层次上对客户端的 Http 请求处理过程。在系统实际运行过程中，往往通过事务的一个运行实例进行监控来监测系统的运行状态。事务通过消息的序列来描述，那么事务的运行实例可以通过消息的具体传输路径来监测。图 3 为云计算系统中事务执行路径的检测结构图。

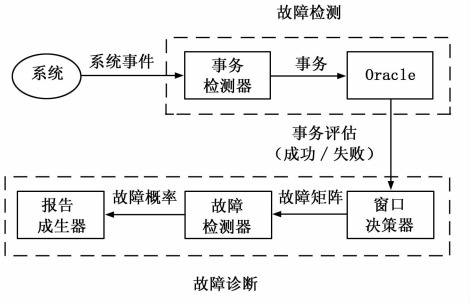

图 3　LDA 的图形表示

在云计算系统中，其运行过程由一系列系统事件或者事务组成，系统检测的目的是通过事务的检测来判断系统是否发生故障，并且进行故障的定位和修复。在故障检测中，通过事务检测器对系统中的事务进行监测，并将每一个事务实例进行编码后存储在数据库（如 Oracle）中。由于事务中的消息往往在网络中传输，本文假设信息超时导致事务失败。在故障诊断中，首先根据窗口决策器将一定时间片段内的事务数据提取出来，并转化为故障矩阵，然后应用故障检测器对故障矩阵进行分析，并给出系统发生每个故障的概率，最后根据故障概率生成诊断报告供诊断专家决策或进行故障的自动修复。

2　故障定位方法

在图 3 中，系统的每一个事务的运行实例都存储在数据库中，其中存储的是事务的执行路径。如果对系统中的每个组件进行编码，那么事务的执行路径可以表示为一个向量。通过窗口决策器可以对一定时间片段内的执行路径进行筛选，从而构成了一个故障矩阵。通过对该故障矩阵进行分析，可以进行系统的故障诊断和定位。

2.1　故障矩阵

假设系统中包含 M 个组件，每个组件用 c_i 来表示，$1 \leqslant i \leqslant M$。系统中可以同时产生 $C(0 \leqslant C \leqslant M)$ 个故障。诊断报告 $D = < \cdots, d_k, \cdots >$ 是系统中诊断出的可能发生的故障的有序列表，其中每一项 d_k 按照其发生故障的概率进行排序。

事务的运行实例为一个 $M+1$ 维向量，其中第 i 个（$1 \leqslant i \leqslant M$）元素表明该运行实例是否包含组件 c_i，第 $M+1$ 个元素表明该事务实例是否成功。系统中的 N 个运行实例构成一个 $N \times (M+1)$ 的状态矩阵，通过对该矩阵的分析可以进行系统的故障识别和定位，故该矩阵成为故障矩阵。在本文的后续讨论中，将故障矩阵记为 (A, e)，其中 A 为 $N \times M$ 维的组件矩阵，e 为 $N \times 1$ 维的结果向量。

2.2　候选故障生成

在每一个事务运行实例中，包含的每个组件 c_i 可能处于正常状态，也可能处于故障状态。组件 c_i 的状态逻辑命题公式如下：

$$h_i \Rightarrow (ok_{in_i} \Rightarrow ok_{out_i}) \tag{1}$$

其中：h_i 表示组件 c_i 的运行状态（正常或者故障），ok_{in_i} 和 ok_{out_i} 分别表示组件 c_i 的输入和输出变量的正确性（true 或者 false）。

在公式（1）中，$h_i = true$ 表明组件 c_i 处于正常运行状态，此时正确的输入变量 $ok_{in_i} = true$ 将生成正确的输出变量 $ok_{out_i} = true$。当组件 c_i 发生故障（$h_i = false$）时或者输入数据错误（$ok_{in_i} = false$）时，系统仍可能产生正常的输出。因此，$e_j = 0$ 并不能表明事务实例 A_j 成功运行并返回结果。

表 1　简单的故障矩阵

c_1	c_2	c_3	e
1	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0

表 1 为一个简单的故障矩阵，根据该故障矩阵，可以得到如下的组件组件健康状态命题：

$$\begin{aligned} &\neg h_1 \vee \neg h_2 \\ &\neg h_2 \vee \neg h_3 \\ &\neg h_1 \end{aligned} \tag{2}$$

其中：$\neg h_1 \vee \neg h_2$ 表示组件 c_1 和 c_2 中的一个或两个发生故障，即 $d_1 = \{1,2\}$；$\neg h_2 \vee \neg h_3$ 表示组件 c_2 和 c_3 中的一个或两个发生故障，即 $d_2 = \{1,3\}$；$\neg h_1$ 表示组件 c_1 发生故障，即 $d_3 = \{1\}$。

2.3　故障排序与定位

本文采用贝叶斯概率方法对上述生成的候选故障进行排名。对于每一个候选故障 d_k，其发生故障的概率为其对故障矩阵中所有的事务实例进行解释的概率。在给定事务运行实例 obs_i 的前提下，基于贝叶斯规则的 d_k 的概率计算公式如公式（3）所示。

$$\Pr(d_k \mid obs_i) = \frac{\Pr(obs_i \mid d_k)}{\Pr(obs_i)} \cdot \Pr(d_k \mid obs_{i-1}) \tag{3}$$

在公式（3）中，分母 $\Pr(obs_i)$ 是一个规范化项，对于不同的 d_k 都有相同的取值，因此可以省略。$\Pr(d_k \mid obs_{i-1})$ 为 d_k 的先验概率。当无先验知识时，$\Pr(d_k \mid obs_{i-1})$ 可转化为 $\Pr(d_k) = p^{|d_k|} \cdot (1-p)^{M-|d_k|}$，其中 p_j 为组件 c_j 发生故障的先验概率，本文令 $p_j = p$。$\Pr(obs_i \mid d_k)$ 的定义如下：

$$\Pr(obs_i \mid d_k)=\begin{cases}0, & \text{如果 } obs_i \text{ 和 } d_k \text{ 不一致;}\\ 1, & \text{如果 } obs_i \text{ 由 } d_k \text{ 唯一确定;}\\ \varepsilon, & \text{其它情况。}\end{cases} \quad (4)$$

当事务实例 $obs_i=(A_{i\cdot},e)$ 时，其正常运行的概率为所有组件都健康的概率，其发生故障的概率为事务实例中一个或者多个组件发生故障的概率。于是，ε 可由如下公式计算出：

$$\varepsilon=\begin{cases}\prod\limits_{j\in d_k \wedge a_{ij}=1} h_j & e_i=0;\\ 1-\prod\limits_{j\in d_k \wedge a_{ij}=1} h_j & e_i=1.\end{cases} \quad (5)$$

在计算 $\Pr(d_k)$ 之前，必须根据故障矩阵 (A,e) 对组件 h_j 的健康状态进行估计。对于每一个组件 h_j，其估计值根据公式（5）中的 ε 策略进行计算，其核心思想是对 d_k 中的故障组件最大化概率 $\Pr(obs \mid d_k)$，其中 obs 为故障矩阵中的所有事物运行实例。于是，在满足公式（5）所示的约束的基础上，求解 $\underset{\langle h_j \mid j\in d_k\rangle}{\operatorname{argmax}}\Pr(obs \mid d_k)$。

以表 1 为例，假设事物运行实例之间是相互独立的，为了阐述如何对候选故障进行排名，根据公式（4）和（5），$\Pr(obs \mid d_k)$ 的计算公式为：

$$\Pr(obs \mid d_1)=(1-h_1\cdot h_2)\cdot(1-h_2)\cdot(1-h_1)\cdot h_1 \quad (6)$$

在给定故障矩阵 (A,e) 后，可以计算出每个候选故障 d_k 的后验概率 $\Pr(obs \mid d_k)$。通过对这些候选故障的后验概率进行排名，可以得到系统的故障诊断报告 $D=<\cdots,d_k,\cdots>$。对于排名靠前的每一个故障 d_k，可以通过组件的包含情况以及包含组件的健康情况进行故障的定位。

3 实验设计与分析

物联网基本结构图如图 4 所示。

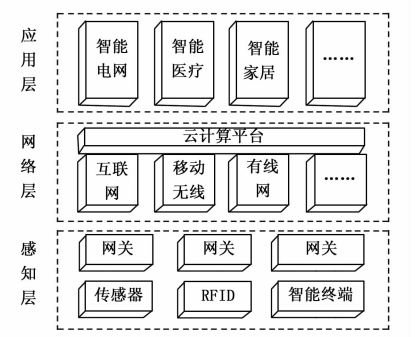

图 4 物联网基本结构图

为了对本文提出的故障诊断方法进行验证，本实验搭建了一个包含 5 个节点的服务器集群。其中 1 个节点为分配节点，并启动两个分配器进程；3 个节点为服务器节点；1 个节点作为数据库服务器。

实验中，将本文提出的方法记为 FaultMatrix，并将其与 FuzzyClustering[15]、FuzzyLogic[16] 和 NeutralNet[17] 3 种算法进行对比。分别对比了 4 种算法的故障预测准确率和算法的执行效率。在实验分析中，为了加速系统的性能测试，采用两种方法对系统中注入故障，并且故障事务在所有事务中的比例为 10%。第一种方法分别在分配器和服务器节点上加入时间延迟故障，即算法在一定的时间阀值内部返回查询结果。第二种方法通过杀掉分配进程和服务进程的方法注入故障。

首先，实验对比了 4 种算法的故障预测准确率。采用的评价指标为算法识别的故障中正确故障的比例，实验结果如图 5 所示。从图中可以看出，在时间延迟和杀掉进程两种类型的故障中，本文提出的 FaultMatrix 方法都能有效地对故障进行正确的判断，其故障识别的准确率明显高于其它 3 中方法。

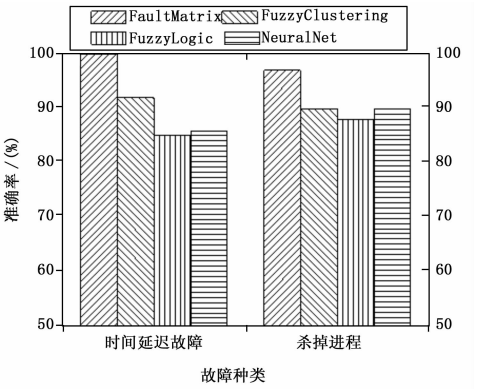

图 5 算法的故障诊断准确率对比

然后，实验对比了 4 种方法在进行故障识别时所用的平均时间，实验结果如图 5 所示。在杀掉进程进行故障注入时，其它 3 种方法都等待进程超时才触发相应的诊断策略，因此其故障诊断所用的时间都高于基于时间延迟的故障注入。在 FaultMatrix 方法中，当杀掉进程时，事务运行路径中所有的后续组件都处于故障状态，因而可以很容易地进行故障的识别。此外，在 4 种方法的对比中，本文提出的 FaultMatrix 方法的故障诊断时间明显低于其它 3 种方法。

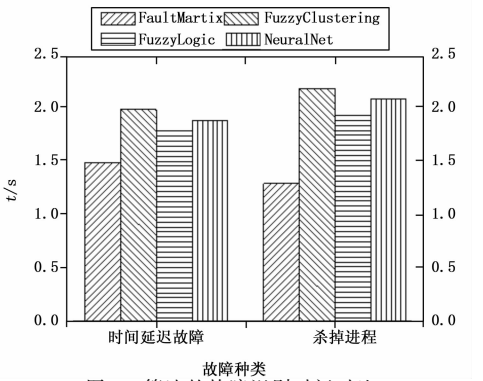

图 6 算法的故障识别时间对比

最后，实验对 FaultMatrix 算法进行了分析。表 2 为该算法对诊断出的故障进行定位的统计分布。其中的每一项表示某分配器到某服务器的消息传递，超时说明消息传递失败，否则说明消息传递成功。从统计结果可以看出，失败的消息传递次数所占的比例为 75/（676+75）=9.99%，与故障的注入比例 10%十分接近。此外，实验观察了 FaultMatrix 算法随着时间的变化所识别出的故障比例的变化图。从图 7 可以看出，随着时间的变化，FaultMatrix 算法识别出的故障所占的比例上下波动，但是波动逐渐变小并且收敛于 10%。

表 2 故障统计分布表(成功/失败)

	服务器 1	服务器 2	服务器 3	总计
分配器 1	85/31	128/4	93/18	306/53
分配器 2	116/7	122/3	132/12	370/22
总计	201/38	250/7	225/30	676/75

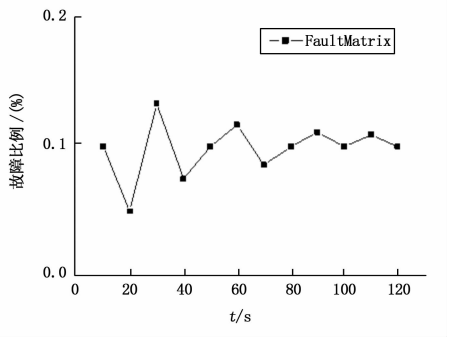

图 7　FaultMatrix 的故障比例随着时间的变化

4　结束语

自动计算是云计算提供持续服务的基础，而故障的识别与定位是自动计算的前提条件。本文对云计算系统的软件结构进行了抽象，给出了事务以及事务执行路径的定义，并将事务的执行路径编码成组件的包含向量，最后基于贝叶斯概率对故障矩阵进行分析进行故障的识别和定位。实验表明，本文提出的方法与其它相关方法相比，故障识别的准确性更高，所用的执行时间更短。

参考文献：

[1] Breiter G, Naik V K, A Framework for Controlling and Managing Hybrid Cloud Service Integration [A]. Cloud Engineering (IC2E), 2013 IEEE International Conference on [C]. 2013, p. 217 - 224.

[2] Gomez I, Marojevic V, Gelonch A, Automatic computing resource awareness in resource managers for cognitive radios [A]. Cognitive Information Processing (CIP), 2010 2nd International Workshop on [C]. 2010, p. 122 - 127.

[3] Yadav S K, Kalra P K, Automatic Fault Diagnosis of Internal Combustion Engine Based on Spectrogram and Artificial Neural Network [A]. in ROCOM'10 [C]. Stevens Point, Wisconsin, USA, 2010, p. 101 - 107.

[4] Gardazi S U, Shahid A A, Survey of software architecture description and usage in software industry of Pakistan [A]. Emerging Technologies, 2009. ICET 2009. International Conference on [C]. 2009, p. 395 - 402.

[5] Vidacs L, Beszedes A, Tengeri D, et al. Test suite reduction for fault detection and localization: A combined approach [A]. Software Maintenance, Reengineering and Reverse Engineering (CSMR-WCRE), 2014 Software Evolution Week-IEEE Conference on [C]. 2014, 204 - 213.

[6] Avritzer A, Cole R G, Weyuker E J. Methods and Opportunities for Rejuvenation in Aging Distributed Software Systems [J]. J. Syst. Softw., 2010, 83, 1568 - 1578.

[7] Alonso J, Matias R, Vicente E, et al. A Comparative Evaluation of Software Rejuvenation Strategies [A]. in WOSAR'11 [C]. Washington, DC, USA, 2011, 26 - 31.

[8] Cotroneo D, Natella R, Pietrantuono R, et al. A Survey of Software Aging and Rejuvenation Studies [J]. J. Emerg. Technol. Comput. Syst., 2014, 10 (8): 1 - 8.

[9] Candea G, Kawamoto S, Fujiki Y, et al. Microreboot — A Technique for Cheap Recovery [A]. in OSDI'04 [C]. Berkeley, CA, USA, 2004, 3 - 3.

[10] Ghemawat S, Gobioff H, Leung S, The Google File System [Z]. SIGOPS Oper. Syst. Rev., 2003, 37, 29 - 43.

[11] Hazelhurst S. PH2: An Hadoop-based Framework for Mining Structural Properties from the PDB Database [A]. in SAICSIT'10 [C]. New York, NY, USA, 2010, 104 - 112.

[12] Cheng S W, Garlan D, Schmerl B. Architecture-based self-adaptation in the presence of multiple objectives [A]. InProc. of SEAMS' 06 [C]. 2006, 21 - 22.

[13] Garlan D, Cheng S, Huang A, et al. Rainbow: Architecture-Based Self-Adaptation with Reusable Infrastructure [J]. Computer, 2004, 37, 46 - 54.

[14] Kramer J, Magee J. A rigorous architectural approach to adaptive software engineering [J]. J. Comput. Sci. Technol., 2009, 24: 183 - 188.

[15] Eslamloueyan R, Designing a Hierarchical Neural Network Based on Fuzzy Clustering for Fault Diagnosis of the Tennessee-Eastman Process [J]. Appl. Soft Comput., 2011, 11, 1407 - 1415.

[16] de Miguel L J, Bl a zquez L F, Fuzzy Logic-based Decision-making for Fault Diagnosis in a DC Motor [J]. Eng. Appl. Artif. Intell., 2005, 18, 423 - 450.

[17] 张　宁, 等. 非均匀采样数据系统故障检测 [J]. 计算机测量与控制, 2008, 16 (6): 774 - 776.

(上接第 45 页)

参考文献：

[1] 杨晓光. 火灾自动报警系统的发展和前景 [J]. 广东公安科技, 2007, 17 (4): 53 - 55.

[2] 刘伟鹏. 无线火灾自动报警系统的设计与应用研讨 [J]. 山西建筑, 2010, 36 (4): 197 - 199.

[3] 陈树学. 刘萱. LabVIEW 宝典 [M]. 北京: 电子工业出版社, 2011.

[4] 阮奇桢. 我和 LabVIEW--一个 NI 工程师的十年编程经验 [M]. 北京: 北京航空航天大学出版社, 2009.

[5] 李晓林. 单片机原理与接口技术 [M]. 北京: 电子工业出版社, 2009.

[6] 尹康泽. 基于单片机温度测量控制系统 [D]. 福建: 仰恩大学, 2010.

[7] 程　军. 传感器与实用检测技术 [M]. 西安: 西安电子科技大学出版社, 2008.

[8] 王新玲, 孙运强, 姚爱琴. 列车车轴温度无线监测系统设计 [J]. 计算机测量与控制, 2012, 20 (1): 119 - 121.

[9] 王灵芝, 苏文松, 翁文陶. 基于 Simplici Ti 的无线火灾报警系统设计 [J]. 微型机与应用, 2011, 30 (14): 24 - 27.

[10] 赵　娜. 无线火灾报警控制器的研制 [D]. 哈尔滨: 哈尔滨工业大学, 2006.

[11] 熊志金. 基于无线传感器网络的列车货物安全监测系统 [J]. 计算机测量与控制, 2012, 20 (8): 2102 - 2104.

[12] 张　宁, 韩　海. 用于森林火灾监测和救灾的无线传感器网络 [J]. 微计算机信息, 2008, 24 (10): 169 - 171.

[13] 曾素琼. 嵌入式低压电力线通信风机控制系统设计 [J]. 计算机测量与控制, 2012, 20 (1): 105 - 108.

[14] 顾亚雄, 朱翠英, 许方华. 基于 LabVIEW 的单片机多路数据采集系统的设计 [J]. 自动化技术与应用, 2009, 28 (10): 46 - 48.

[15] 姚运萍, 陈继开, 高艳雯. 基于 LabVIEW 和单片机的多功能病房监护系统 [J]. 微计算机应用, 2007, 28 (10): 1081 - 1084.