面向按需供给的资源需求滤波估算方法

黄 翔 [1, 2] 陈 伟 [3] 宋云奎 [3] 陈志刚 [1]

摘 要 随着按需供给资源使用模式的推广, 软件的资源需求已成为资源优化控制的重要属性. 监测和估算是目前常用的资源消耗获取方法, 但监测工具难以在运行时准确度量短任务的资源需求, 回归分析方法又因受到多元共线性和不确定性因素的影响, 导致其取值精度下降. 本文提出了一种基于 Kalman 滤波的资源需求估算方法. 该方法建立了可度量属性集与不可度量的资源需求间的关联, 并利用滤波过滤度量过程中的噪声, 达到降低估算误差的目的. 基准测试的结果表明, 通过合理的设置滤波参数, 本方法能够快速逼近真实值, 且平均误差小于 8 %.

关键词 按需供给, 资源需求, 滤波, 估算

引用格式 黄翔, 陈伟, 宋云奎, 陈志刚. 面向按需供给的资源需求滤波估算方法. 自动化学报, 2014, **40**(5): 942−951

DOI 10.3724/SP.J.1004.2014.00942

Filter Based Resource Demand Estimation for On-demand Provision

HUANG Xiang[1, 2] CHEN Wei[3] SONG Yun-Kui[3] CHEN Zhi-Gang[1]

Abstract As the development of demand resource provision, resource demands of software is becoming one of the most important attributes of resource management. Measurement and estimation are widely used in fetching the demands. However, it is hard to measure the short job′s resource demands by current measurement tools, and the regression methods suffer from the well-studied problem of multicollinearity. Therefore, the estimated results are not confident. In order to improve the estimation precision, we propose a Kalman filter based approach, which can predict the unobservable attribute by observable attributes, and filter the noise existing in the measurement. At last, we test our approach with a benchmark and compare the relative errors, which can demonstrate that with the reasonable parameters, our approach can get close to the real demands quickly, and get the estimated value with the mean error less than 8 %.

Key words On demand, resource demand, filter, estimation

Citation Huang Xiang, Chen Wei, Song Yun-Kui, Chen Zhi-Gang. Filter based resource demand estimation for on-demand provision. *Acta Automatica Sinica*, 2014, **40**(5): 942−951

按需供给是当前云计算领域的热点研究方向. 按需供给指在满足性能需求的前提下, 根据用户资源需求的变化动态的分配资源, 使资源利用率持续维持在一个较高的范围. 这种随需应变的供给方式, 对传统粗粒度的资源控制方法提出了严峻的挑战. 因为在新的需求下, 需要更细致地了解各个服务 (或者组件) 的资源需求和变化趋势, 以便准确地做出优化动作.

收稿日期 2013-05-22 录用日期 2013-08-22
Manuscript received May 22, 2013; accepted August 22, 2013
国家自然科学基金 (61272013), 广东省自然科学基金 (S2013040011941) 资助
Supported by National Natural Science Foundation of China (61272013) and Guangdong Natural Science Foundation of China (S2013040011941)
本文责任编委 夏元清
Recommended by Associate Editor XIA Yuan-Qing
1. 中国能源建设集团广东省电力设计研究院 广州 510663 2. 中山大学信息科学与技术学院 广州 510006 3. 中国科学院软件研究所 北京 100190
1. Guangdong Electric Power Design Institute, China Energy Engineering Group, Guangzhou 510663 2. School of Information Science and Technology, Sun Yat-Sen University, Guangzhou 510006 3. Institute of Software, Chinese Academy of Sciences, Beijing 100190

服务是云应用的基本组成单元. 服务会消耗各种计算资源, 但与内存、网络等其他资源相比, CPU 资源更加难以准确获得. 目前, 主流的 CPU 资源需求获取方法可分为直接度量和间接估算两类.

直接度量的方法通常基于探针技术, 通过插入监测代码收集资源需求. 但这类方法或者无法准确监测在线服务这类短任务的资源消耗[1−2], 或者会引入大量的性能开销[3−4] (30 % 以上), 对于在线系统而言这是完全不能忍受的. 特别是随着多 CPU、虚拟化等技术的引入, 直接监测的方法更难准确获得实际的资源消耗.

回归分析的方法常用来克服这种难以直接监测的问题[5]. 但是通常回归分析难以避免多元共线性 (Multicollinearity) 问题[6], 这会导致结果的严重失真, 而且如果资源需求是非确定的, 那么回归分析的方法的可信度又将大大降低.

针对上述问题, 本文针对服务的 CPU 资源需求的获取问题, 提出了一种基于 Kalman 滤波的估算方法. Kalman 滤波最大的贡献是通过整合尽可能多的观测值, 间接地估算出隐藏的参数[7−8]. 本

文利用这一特性, 建立起可观测值与资源需求的关系, 然后通过迭代逐步获得精确结果. 最后, 本文以 TPC-W 基准测试为例[9−10], 在模拟负载和软硬件环境变化的情况下, 验证了本文方法的准确性.

本文的组织结构如下: 第 1 节介绍了本文的理论基础; 第 2 节给出了问题的形式化描述; 第 3 节给出了 Kalman 滤波计算过程中所需参数的设置过程; 第 4 节通过实验手段验证了本方法有效性; 第 5 节则对相关工作进行比较; 最后, 第 6 节总结了本文的工作与主要贡献.

1 Kalman 滤波

Kalman 滤波提供了一个在离散时间点估算不可观测状态 X 的迭代计算方法[11−12]. 第 k 时刻状态 X_k 可以定义为一个线性随机差分方程:

$$X_k = AX_{k-1} + Bu_{k-1} + w_{k-1} \tag{1}$$

第 k 时刻测量值 Z_k 定义为

$$Z_k = HX_k + v_k \tag{2}$$

其中, A 是从 $k-1$ 时刻到 k 时刻状态转换矩阵, u_{k-1} 是可选的控制参数, B 是与控制相关的矩阵, w_{k-1} 为测量误差, 其协方差矩阵为 Q_{k-1}. H 是 X_k 到 Z_k 的转换矩阵, v_k 是测量误差, 其协方差矩阵为 R_k.

Kalman 滤波采用反馈的方法进行估算. 滤波首先会估算出参数的状态, 然后使用反馈控制过滤监测噪声. 因此, Kalman 滤波的公式可以分为两组: 一组是时间更新方程 (Time update); 一组是监测更新方程 (Measurement update). 时间更新方程利用当前状态和算误差的协方差矩阵 P 获得下一时间间隔的先验估计 (Priori estimate). 监测更新方程则负责反馈控制, 将新测试到的监测值加入先验估计中, 得到后验估算值.

时间更新方程可以看作一个预测方程 (Predictor equations), 而监测更新方程则可以看作修正方程 (Corrector equations). Kalman 滤波整个 “预测−修正” 过程如图 1 所示[11].

时间更新过程负责将 $k-1$ 时间片内参数的状态转化为 k 时间片内的状态. 该过程中需要使用到 A、B 和 Q 矩阵. 每次计算都需要给出这几个矩阵, 但根据系统类型的不同, 它们可以是固定值, 也可以是每一步更新都不相同. 计算开始时, 还需要给出待估计值的初始值和 P 矩阵.

监测更新过程首先会计算 Kalman 增益 (Kalman gain), 然后再利用监测数据 Z 和预测数据 HX 之间的误差更新待估算参数的状态, 得到一组后验数据. 最后再更新误差的协方差矩阵. 当 “预测−修正” 过程计算完成之后, 会不断重复上述计算过程, 将上一次的后验数据映射为下一次的先验数据.

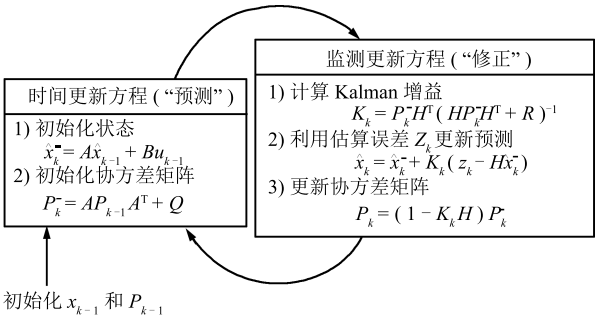


图 1　Kalman 滤波 “预测−修正” 过程

Fig. 1　Time update and measurement update process of Kalman filter

2 问题描述

本节主要介绍资源需求估算的形式化描述, 包括建立可观测值与不可观测值之间的联系, 以及如何计算资源需求的参考值.

2.1 参考值设置

资源需求虽然难以直接度量, 但是通过其他参数还是可以间接地了解其范围和变化情况. 本文引入资源需求签名的概念, 即资源需求的参考值, 用 s 表示.

首先, 考虑一个简单的队列系统, x 代表系统中某一组件的平均资源需求 (用资源占用时间表示), l 代表新任务到达时系统队列长度, d 代表组件在系统中的平均延迟时间 (组件自身的等待和执行时间, 不包括等待其他组件响应的时间). 由此可知, 延迟时间等于其自身的资源需求与队列等待时间之和:

$$d = x + x \times l \tag{3}$$

在闭合队列模型中, 如果存在 m 个客户, 那么队列长度 l 等于系统在 $m-1$ 个客户时的平均队列长度, 可近似表述为 $l \approx \bar{l} \times (m-1)/m$[13], 其中 $\bar{l}$ 是平均队列长度. m 趋于无穷时, $(m-1)/m$ 趋近于 1, 因此队列长度 l 可以近似为平均队列长度 $\bar{l}$, 即 $l \approx \bar{l}$. 由此得到等式:

$$d = x + x \times \bar{l} \tag{4}$$

设 t 代表系统平均吞吐量, 应用 Little′s 法则 $\bar{l} = t \times d$[13], 可得到等式:

$$d = x + x \times t \times d \tag{5}$$

进一步地, 根据效用法则 (Utilization law)[13], 可知资源利用率等于吞吐量乘以资源需求, 即 $U = t \times x$, 推导出等式:

$$\bar{l} = \frac{U}{1-U} \tag{6}$$

典型的计算机操作系统在处理多个任务时使用分时规则, 如果系统平均有 m 个并发任务, 则每个任务被分配到 $1/m$ 的平均资源需求, 因此延迟时间 d 可表示为

$$d = x \times m \tag{7}$$

由于 $m = \bar{l} + 1$, 根据式 (6) 和式 (7) 可得到某组件的资源需求签名 s 的等式:

$$s = d \times (1-U) \tag{8}$$

式 (8) 说明了资源需求、延迟时间以及资源利用率之间的关系. 由此, 可以得到资源需求的一个合理估计.

2.2 Kalman 滤波方程

参考值虽然可以估计服务时间的一个大致区间, 但是仍受到监测误差等因素的影响, 并不是准确的资源需求. 为此我们利用 Kalman 滤波对其进行处理. 本方法需周期性的收集监测数据, 相关参数如下:

1) W 表示时间窗口的长度, 设为定值;

2) n 表示服务器上的组件总数;

3) c_i 表示第 i 个组件在 W 时间内被调用的次数;

4) U 表示第时间 W 内的总 CPU 利用率;

5) x_i 表示服务器上第 i 个组件在 W 时间内的平均资源需求;

6) υ 表示系统自身消耗的资源, 如定时服务和系统调度等.

根据利用率的法则, 对每台服务器, 在每个时间间隔内, 可以得到式 (9), 即总 CPU 利用率等于各组件吞吐率与资源需求乘积的累加和:

$$U \times W = \sum_{i=1}^{n} c_i x_i + \upsilon \tag{9}$$

由于监测很难获得精确的资源需求 x_i 和系统开销 υ, 我们使用 $\hat{x}_i$ 表示对 x_i 的一个估计, 令 $\hat{\upsilon} = \upsilon/W$, 这样式 (9) 就能做如下映射:

$$\hat{U} = \sum_{i=1}^{n} \hat{x}_i \left(\frac{c_i}{W}\right) + \hat{\upsilon} \tag{10}$$

因此, 根据 Kalman 滤波的定义, 本文将式 (1) 和式 (2) 做如下映射, 其中 X_k 表示第 k 时间片内平均资源需求的集合:

$$\begin{aligned} X_k &= AX_{k-1} + w_{k-1} \\ z_k &= \sum_{i=1}^{n} t_i x_i^k + \upsilon_k \end{aligned} \tag{11}$$

其中, A 表示服务时间从上一时间片转换为下一时间片的控制矩阵 (下一小节中介绍), $X_k = [x_1^k, x_2^k, \cdots, x_n^k]^{\mathrm{T}}$ 表示 k 时刻各组件的资源需求, z_k 为总 CPU 利用率 U, $t_i = c_i/W$ 为各组件的吞吐率. 对于服务器而言, 系统自身消耗的资源 υ 通常很小, 可以忽略或者视为测量误差的一部分.

H 可以定义如下:

$$H_k = [t_1, t_2, \cdots, t_n] \tag{12}$$

本文所用的 Kalman 滤波迭代过程如下:

1) 设 $w_{k-1} = 0$ 估计 X 的状态:

$$\hat{X}_k^- = A_k \hat{X}_{k-1} \tag{13}$$

2) 估计协方差矩阵 P_k^-:

$$P_k^- = A_k P_{k-1} A_k^{\mathrm{T}} + Q_k \tag{14}$$

3) 计算 Kalman 增益 K:

$$K_k = P_k^- H_k^{\mathrm{T}} (H_k P_k^- H_k^{\mathrm{T}} + R_k)^{-1} \tag{15}$$

4) 更新 X 的状态:

$$\hat{X}_k = \hat{X}_k^- + K_k(z_k - H_k \hat{X}_k^-) \tag{16}$$

5) 更新协方差矩阵 P_k:

$$P_k = (I - K_k H_k) P_k^- \tag{17}$$

迭代过程中利用监测获得的新的状态信息, 可以不断地修正估测值, 使其能正确反映实际值的变化. 计算过程中, 因为观测值 z 定义为 CPU 总利用率, 而 H 和 K 是向量, 所以式 (15) 中求逆退化为求倒数. 因此, 整个迭代过程的复杂度由式 (16) 决定, 其复杂度为 $\mathrm{O}(n^3)$. 又因为一台服务器上组件的数量级很小, 所以可以快速计算出资源需求.

3 参数设置

滤波器的参数设置是影响估算精度的关键, 它们会对算法的逼近真实值的速度以及精确性构成不同程度影响. 需要设置的主要参数包括初始状态向量、误差协方差、Q 与 R 矩阵、监测窗口长度等.

3.1 初始值设置

一些研究工作认为初始状态向量及误差协方差的设置对评估结果影响较小[14]. 但是, 我们发现设置适当的初始状态向量, 并利用转换矩阵 A 对状态向量进行动态修正, 可以提高方法逼近真实值的速

度. 本文利用资源需求的参考值 s 作为资源需求的初始值 (详见第 2.1 节), 定义如下:

$$X_0 = \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_n \end{bmatrix} \tag{18}$$

另外, 因为各组件的服务时间是独立的, 所以 P_0 是一个对角矩阵. 如果 $P_0 = 0$, 滤波的更新过程会认为初始值可信, 不会更新初始值. 但 P_0 初始值的选取并不是十分关键, 可以选择任意 $P_0 \neq 0$ 的值, 滤波最后都会收敛[15−17]. 因此, 我们选取 P_0 初始值的形式如下:

$$P_0 = \begin{bmatrix} (x_1^0)^2 & 0 & \cdots & 0 \\ 0 & (x_2^0)^2 & \cdots & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & \cdots & (x_n^0)^2 \end{bmatrix} \tag{18}$$

考虑到 Kalman 滤波本身过滤噪声的特性, 本文将不精确的资源需求的参考值引入 Kalman 滤波的时间更新过程之中, 用于初始设置和转换矩阵. 本文将时间更新的转换矩阵 A 定义为如下形式:

$$A = \begin{bmatrix} a_1 & 0 & \cdots & 0 \\ 0 & a_2 & \cdots & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & \cdots & a_n \end{bmatrix} \tag{19}$$

其中, a_i 表示第 i 个组件的转换系数, 通过相邻时间片内的资源需求的参考值计算而来, 其定义如下:

$$a_i = \begin{cases} \dfrac{s_i^k}{s_i^{k-1}}, & k > 1 \bigwedge s_i^{k-1} > 0 \bigwedge s_i^k > 0 \\ 1, & \text{否则} \end{cases} \tag{20}$$

转换矩阵主要用于将上一次更新的后验服务时间转换为这一次更新的先验服务时间. 矩阵 A 的作用是为了反映服务时间的波动与时变性, 比如因为软件更新升级等因素引起的阶梯形变化. 但考虑初始情况以及某些组件在监测窗口中可能未被调用的情况, 本文对 a_i 引入了条件判断, 在条件不满足时不对服务时间进行修正.

3.2　参数限制

由于服务时间总是大于 0 的数, 因此必须对服务时间作出约束. 为适应服务时间为正的特性, 本文对 Kalman 滤波的计算过程的第 4 步中的 x 约束在如下范围:

$$0 \leq \tau^- < x < \tau^+ < +\infty \tag{21}$$

本文定义了变量 x 的单步最大变化的范围, 通过变化因子 δ 控制. 从第 k 时间片开始, 下一步的变化区间定义为

$$(x_{\text{lower}}, x_{\text{upper}}) = ((\delta\tau^- + (1-\delta)x^k), (\delta\tau^+ + (1-\delta)x^k)) \tag{22}$$

本文定义 $\delta = 0.8$, τ^- 和 τ^+ 的初始值为 $\tau_i^- = x_i/2$, $\tau_i^+ = 2x_i$. 因此, Kalman 滤波迭代计算的第 4 步 (式 (16)) 之后采用下式进行约束:

$$x_i = \min(x_{\text{upper}}, \max(x_{\text{lower}}, x_i)) \tag{23}$$

每次迭代之后, $\tau_i^- = \min(\hat{x}_i, \tau_i^-)$, $\tau_i^+ = \max(\hat{x}_i, \tau_i^+)$. 约束并不会对滤波造成太多影响, 因为它的反馈机制可以修正约束带来的误差.

3.3　Q 和 R 矩阵设置

Q 和 R 矩阵影响 Kalman 增益 K, 因此影响滤波器对新观测数据的敏感度. 本小节主要介绍滤波迭代过程中 Q 和 R 矩阵的设置方法.

实际中, Q 矩阵是不可知的. 但如果 Q 设置过大, 则会增大迭代的误差协方差矩阵 P, 进而增大 Kalman 增益 K, 导致滤波器对评估误差反应过大, 最终造成评估结果抖动; 如果 Q 设置过小, 则会降低 Kalman 增益 K, 最终降低滤波器对评估误差的响应性, 难以追踪服务时间变化.

在本文中, 我们的应对策略是将 Q_k 设置为对角矩阵, 且对角线元素利用前三个时间片迭代产生的 x 值动态计算获得, 即:

$$Q_k = \begin{bmatrix} \xi_1^2 & 0 & \cdots & 0 \\ 0 & \xi_2^2 & \cdots & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & \cdots & \xi_n^2 \end{bmatrix} \tag{24}$$

$$\xi_i = \begin{cases} \dfrac{x_i^k + x_i^{k-1} + x_i^{k-2}}{3}, & K \geq 3 \\ x_i^k, & K < 3 \end{cases} \tag{25}$$

R 矩阵是总 CPU 利用率的测量误差的协方差矩阵, 因此可以事先通过采样的方法获得. 后文实验结果显示 Q 和 R 矩阵的设置对评估精确度存在较大的影响.

3.4　监测窗口长度

计算过程中, 本方法需要获得各组件的吞吐量、平均响应时间以及总 CPU 利用率等. 这些数据是

通过对监测数据求平均获得的. 因此, 较长的监测窗口会降低监测数据的抖动、提高数据精度, 但同时也会增加发现服务时间变化的难度. 相反, 较短的监测窗口可以加快追踪服务时间变化的速度, 但代价却是增大监测数据误差、影响估算的精度. 后文中, 我们将通过实验的方法分析监测窗口长度对估算精度的影响.

4 验证与评价

本节我们将通过实验的方法分析参数设置对 Kalman 滤波估算精度的影响.

4.1 实验设置

Bench4Q[10] 是中国科学院软件研究所软件工程技术中心开发的遵循 TPC-W 规范[9] 的企业级标准测试套件. 该套件为在线系统的测试提供了一个受控的环境.

根据 TPC-W 规范的定义, 测试床模拟了典型电子商务网站的构成, 分为 Servlet、EJB 和数据库三个层次. 其中, 客户机和 Servlet 容器之间, 采用了一个独立的 HTTP 负载均衡器, Servlet 容器和 EJB 容器之间则采用了嵌入式的负载均衡器[18], 即由 EJB 的客户端代理负责调度与均衡. 最后 EJB 容器会直接访问数据库服务器. 图 2 给出了本文所采用测试床的结构, 软硬件配置见表 1.

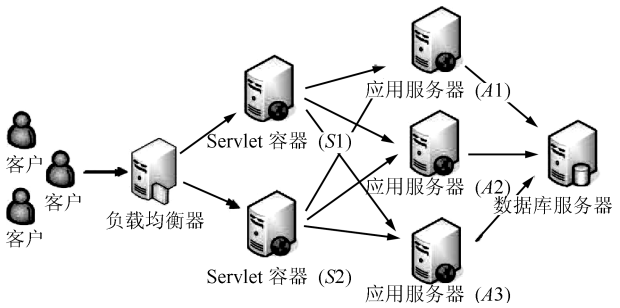


图 2 测试床

Fig. 2 Testing bed

一般而言, 对于一个电子商务网站, 用户完成一项任务需要经过若干次交互, 比如选择商品、提供物流信息、选择支付方式和确认购物等. 为反映实际交互过程, TPC-W 定义了 14 种不同类型的请求 (如表 2 所示), 分为浏览和订单两种.

此外, 为了更真实地反映用户与应用交互的过程, TPC-W 定义了 3 种典型的负载, 详见 TPC-W 规范.

1) 浏览混合模式 (Browsing mix): 95 % 的浏览, 5 % 的订单;

2) 购物混合模式 (Shopping mix): 80 % 的浏览, 20 % 的订单;

3) 订单混合模式 (Ordering mix): 50 % 的浏览, 50 % 的订单.

表 1 测试床配置

Table 1 Configuration of testing bed

Server	Processor	RAM	No.
Client (Emulated-browsers)	Pentium IV/1.8 GHz	2 GB	3
Load balance-ngnix	Pentium IV/3.9 GHz	3 GB	1
Servlet Server-tomcat 6.0	Pentium IV/2.8 GHz	3 GB	2
Application server-JBoss 7.0	Pentium IV/2.8 GHz	3 GB	3
Database server-DB2-V 9.5	Pentium IV/3.9 GHz	3 GB	1

表 2 TPC-W 中的 14 种事务

Table 2 Fourteen basic transactions in TPC-W

Browsing type	Ordering type
Home	Admin confirm
Best	Sellers admin request
New products	Buy confirm
Product detail	Buy request
Search request	Cart
Search results	Customer registration
	Order display
	Order inquiry

TPC-W 同时也定了不同混合模式下各个请求的混合比例和访问的顺序. 此外, 规范还给出了数据库的默认设置, 默认有 10 000 件商品和 1 440 000 个用户.

下面我们首先给出实验测试数据, 分析负载、应用和平台对性能造成的影响, 后续的验证将与这些数据进行比较.

4.2 Q 和 R 的设置与配置的验证

由于 Q 和 R 矩阵对估算精度有明显的影响, 这一部分我们分析如何合理地设置 Q 和 R 矩阵.

本实验以 1 分钟为一个时间片进行更新, 时间片大小对估算结果的影响在后续章节中介绍. 利用总体预测误差的平均值衡量 Q 和 R 矩阵对估算精度的影响, 即误差定义为如下形式:

$$e_z = \left(\frac{1}{k}\right)\sum_{k=1}^{k}\frac{|z_k - H(\hat{x}_k^-)|}{z_k} \tag{26}$$

每次迭代, 我们用比例因子 $Qfac$ 和 $Rfac$ 乘以矩阵 Q 和 R, 比例因子的取值范围是 $0.01 \sim 100$, 然后在上一小节介绍的负载环境下进行测试, 不同的

比例因子下的误差在图 3 中给出. 从图中可以看出误差在对角线处明显地分开, 在 $Qfac \geq Rfac/10$ 的区域内, 误差保持在 0.069. 说明在相同的配置下, Q 越大对测量误差的敏感性越高. 从图的左下角可以看出, 当 $Rfac$ 增长时, 预测误差明显增大, 当 $Rfac$ 增大到 100 时, 预测误差增大到了 0.756.

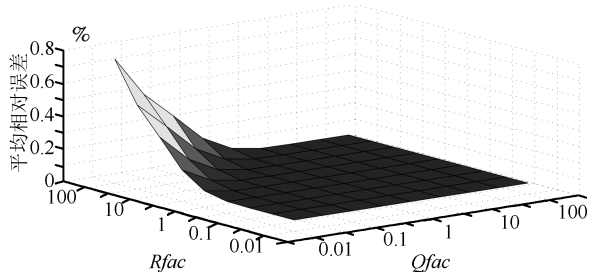


图 3　矩阵 Q 和 R 对预测误差的影响

Fig. 3　The impacts of Q and R

从上述数据可以看出, Q 和 R 对预测误差有着明显的影响. 对于实际系统, 如果不知道矩阵 Q 的值, 可以将其设置得相对较大, 而不是相对较小. 因为 Q 过小, 将会极大地降低逼近真实值的速度, 甚至于完全偏离实际值. 与 Q 矩阵相反, R 矩阵则应当设置得比较小, 否则会引起较大的估算误差.

但是 Q 的值也不宜过大, Q 越大估测值的变化就会越大. 下面我们针对 Q 矩阵对服务时间的影响做了一组实验. 该实验 $Rfac$ 设为 1, $Qfac$ 仍从 0.01 到 100 变化, 我们用服务时间的相对均方差分析 Q 对服务时间的影响.

$$e_x = \frac{\delta}{\mu} \tag{27}$$

其中, δ 和 μ 分别是服务时间误差 $e_{x,k}$ 的样本标准差和样本均值. 表 3 给出了标准差波动的变化情况. 从表中可以看出, Q 越大估算值的波动范围越大, 当 $Qfac$ 为 100 时, 相对方差会达到 0.57. 虽然 Q 越大, 预测误差会越小, 但是各个服务的抖动却会越大. 为了保证预测误差范围控制在较小范围, 而服务时间又不至于波动太大, 后文 $Qfac$ 和 $Rfac$ 均设置为 1.

表 3　矩阵 Q 对服务时间预测误差的影响

Table 3　The relative errors about the impacts of matrix Q on service time estimation

$Qfac$	0.01	0.03	0.1	0.3	1	3	10	30	100
e_x	0.0001	0.0009	0.0015	0.009	0.05	0.08	0.11	0.25	0.57

4.3　初始设置与调整的验证

一般而言, 初始值对滤波最终的收敛性并不会有决定性的影响, 可以设置任意可能的值. 但我们发现虽然初始值的设置不会影响算法的收敛性, 但是对逼近真实值的速度有着明显的影响. 本实验分析了状态向量的初始设置对估算结果的影响. 逼近真实值的速度主要通过迭代步数与误差进行衡量, 误差指服务时间的实测与预测之间的差距.

实测服务时间本文利用 JProbe[19] 收集. 考虑到 JProbe 会引入大量的额外开销, 本文引入了剖析比例因子 (Profiling ratio factor, PRF) 对收集到的数据进行处理. PRF 基于一个假设, 即额外开销会等概率的分布在整个测试之中, 因而可通过比较开启或关闭 JProbe 时的总 CPU 利用率进行折算.

由于 JProbe 引入了大量的开销. 本节通过两组对比实验, 在低负载情况下收集平均服务时间, 并换算出 PRF. 第一组实验启动 JProbe, 在并发量为 20 的情况下, 运行半小时, 收集各个组件的服务时间, 并记录总 CPU 利用率. 第二组实验则关闭 JProbe, 其他设置与前两组相同. 通过计算, Servlet 服务器和应用服务器的 PRF 分别为: 6.5 和 7.9. 值得注意的是, 由于服务时间并非固定值, 因此本文采用平均值作为服务时间的参考值, 目标是评估估算值是否能能接近平均值.

为了对比初始值对算法逼近真实值速度的影响, 本文分别考察了 $\hat{X}_0$ 为随机数和本文所述方法的差异, 设计了两组对比实验: 第一组实验 ($Ex1$) 采用随机数, 第二组实验 ($Ex2$) 采用服务时间签名做初始值.

本文利用余弦相似度 (Cosine similarity) 比较估算值 (X) 与直接度量值 (X') 的差异. 余弦相似度是一种用于比较两个 n 维向量相似度的度量, 相似度的值域为 0 到 1, 相似度接近 1 则表示两个向量越近似, 相似度为 0 则表示两个向量差距很大. 余弦相似度的定义如下:

$$\mathrm{Sim}(X, X') = \cos\theta = \frac{\sum_{k=1}^{n}(x_k x'_k)}{\sqrt{(\sum_{k=1}^{n} x_k^2)(\sum_{k=1}^{n} x'^2_k)}} \tag{28}$$

图 4 给出了两组实验的余弦相似度与迭代周期的关系. 从图中可以看出, 服务时间初始值与实际度量服务时间的余弦相似度大概为 0.81, 具有比较高的相似程度. 与之相比, 随机数与实际度量服务时间的余弦相似度则很低, 大概只有 0.51. 而且 $Ex1$ 逼近真实值的速度明显高于 $Ex2$ 的速度, $Ex1$ 大概经过 10 步的迭代可以达到相似度为 0.97 的程度, 而 $Ex2$ 则需要大概 250 步左右的迭代才能大致达到 $Ex1$ 初始值的相似度水平, 之后再经过约 100 步的迭代结果开始靠近真实值, 相似度水平与 $Ex1$ 基本相当.

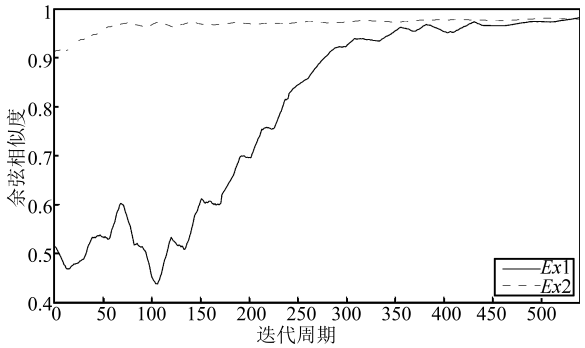


图 4 初始值设置与调整

Fig. 4 Setting and tuning of initial value

$Ex1$ 能快速逼近的一个主要原因是初始值设置以及由初始值派生出的参数设置更合理, 比如 Q 矩阵和 P 矩阵更加接近真实值, 因而经过不多步数的迭代就可以达到比较精确的程度. 而由随机数给定的初始值由于与实际状态相差甚远, 对实际服务时间不具备任何可解释性, 因而滤波初始时随机性过大, 经过若干次迭代之后才能逐渐发现服务时间实际的变化情况, 最后才能靠近实际值附近.

上述观察表明, 可以看出初始值对迭代速度有着明显的影响. 因而, 通过设置适当的初始状态向量可以提高方法的计算速度.

4.4 监测窗口的设置与调整的验证

本方法使用的可度量的监测数据是按时间窗口取平均获得的, 因此采集样本的集合与窗口长度有关, 而样本集的选取又很可能影响到估算精度. 本小节以不同时间窗口为单位, 验证监测窗口设置与估算精度之间的关系. 时间窗口设置分别为 10 秒、30 秒、1 分钟、5 分钟和 10 分钟, 负载规模设置为 200、500、1 000、2 000、3 000、4 000 和 5 000.

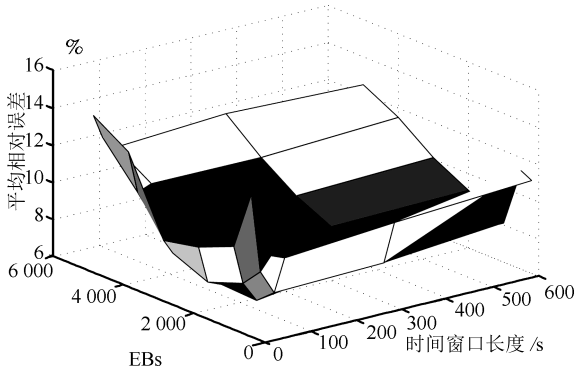
我们用平均相对误差作为衡量服务时间估算精度的标准, 即利用实际测量的服务时间作为基准, 验证估算服务时间与实际测量值之间的的差异. 相对误差定义如下:

$$\overline{e} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{1}{m}\sum_{k=1}^{m}\frac{|x_i^k - \hat{x}_i^k|}{x_i^k}\right) \tag{29}$$

其中, i 表示组件的序号, k 表示迭代周期, x_i^k 表示实际测量服务时间, $\hat{x}_i^k$ 表示估算服务时间, 系统中总共有 n 个组件, 记录了 m 个时间片的样本. 图 5 给出了不同层次服务器上服务时间估算在不同时间窗口和不同负载下的平均相对估算误差.

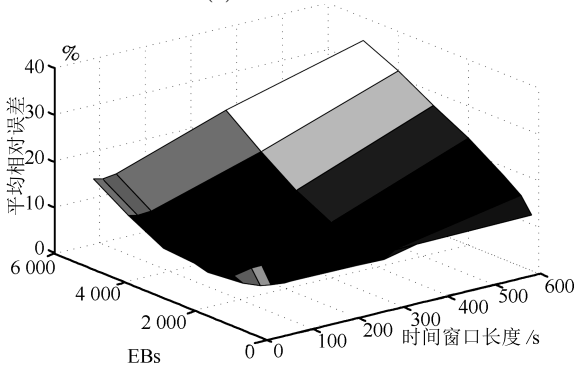
从性能数据中我们可以得到如下结论:

1) Servelt 服务器上, 当时间窗口设置小于 1 分钟时, 估算误差较高, 时间窗口为 1 分钟时估算误差最低, 随着时间窗口的增长, 估算误差会缓慢地有所增加. 而当负载小于 1 000 时, 估算误差也会明显增高, 而当负载大于 3 000 时估算误差会有略微增高. 在时间窗口为 60 秒, 负载为 1 000 和 2 000 时估算误差最低, 分别为 7.1 % 和 7.2 %, 当负载为 200 且时间窗口为 10 秒时估算误差最大, 误差为 15 %, 当负载为 5 000 且时间窗口为 600 时误差为 12.5 %.



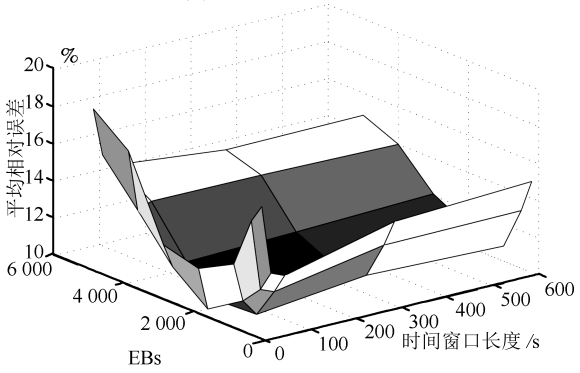
(a) Servlet 服务器

(a) Servlet server



(b) 应用服务器

(b) Application server



(c) DB 服务器

(c) DB server

图 5 不同时间窗口下的平均相对误差

Fig. 5 Mean relative errors under different windows

2) 应用服务器上, 当负载和时间窗口增大时误差明显增高, 当负载为 5 000 且时间窗口为 10 分钟

时, 平均相对误差为 35 %, 当负载为 200 且时间窗口为 10 秒时, 平均相对误差为 15 %. 与 Servlet 服务器相比, 应用服务器上的误差在负载增加时增加非常显著. 在时间窗口为 1 分钟且负载为 1 000 和 2 000 时, 平均最小误差分别为 7.3 % 和 7.4 %.

3) 数据库服务器上的平均估算误差整体高于前两类服务器, 最小误差也是在时间窗口为 1 分钟且负载为 1 000 和 2 000 时, 平均最小误差分别为 10 % 和 10.3 %. 数据库服务器上平均相对误差的变化情况与 Servlet 服务器相似, 当负载为 200 且时间窗口为 10 秒时估算误差最大, 误差为 18.6 %.

服务时间估算的基本原理是利用监测数据过滤估算误差, 估算过程中需要用到 CPU 总利用率、组件吞吐率、组件的响应时间等测量数据. 而这些数据的精度与测量窗口的长度又有着密切的联系. 其中最为显著的是响应时间. 一方面, 当监测窗口过小时误差较大, 而当监测窗口增长, 误差会逐步降低. 另一方面, 在中等负载即 CPU 利用率分布在 30 % ∼ 60 % 时误差相对较小.

因而可以看出, 当负载过高或过低时测量误差也会增大. 此外, 受监测技术的影响, 组件吞吐率在时间窗口较短时, 组件执行跨监测窗口的问题就会比较显著, 因而误差相对较大. 从图 5 (b) 中可以明显看出, 由于应用服务器在高负载时 CPU 利用率接近饱和, 导致监测误差明显增大, 因而估算误差也随之增大. 而三种不同的服务器在低负载和短时间窗口时估算误差也都相对较大.

总的来说, 监测时间窗口和负载的选取对估算误差有着显著的影响, 本文通过实验发现, 时间窗口选 1 分钟到 5 分钟比较合适, 而负载应当选取 CPU 平均利用率分布在 30 % ∼ 60 % 为宜.

5 相关工作

资源需求是一个与运行平台相关的参数, 也是性能建模过程中最难获取的一个参数. 这类方法假设性能模型的其他参数已经获得, 比如组件的执行序列、用户的行为等, 唯一缺少的只有资源需求. 而在运行时由于很难直接度量资源消耗的实际值, 通常都是通过间接的手段统计或者估算资源需求[20].

回归分析 (Regression analysis) 是定量分析两种或两种以上变数间相互依赖关系的一种统计分析方法. 回归分析应用领域非常广泛, 主要用于分析现象之间的具体相关形式和因果关系等.

Bard 和 Shatzoff 首先研究操作系统函数的资源消耗的问题[21]. 他们所研究的系统不具备直接度量每个函数资源消耗的能力. 系统运行时, 他们将执行时各个函数使用的比率以及资源总消耗量不断收集起来. 然后, 他们使用回归的方法估算每个函数所消耗的资源. 其后, 基于回归分析估算函数资源消耗的方法被广泛使用[5, 22−23]. 对于这些方法, 只需要给定一组功能负载的混合情况, 以及总的资源利用率, 各个功能模块的资源消耗就可以计算出来. 系统总的资源消耗需求就可以通过负载的需求与各个功能模块资源消耗的估算值相乘获得.

Rolia 等指出基于回归方法估算的精度会受到函数资源消耗不确定性 (Deterministic) 的影响[24−25]. 确定性指一个函数的资源消耗是否是一个确定值, 是否会因使用的情况和时机不同而发生变化. 对于计算机系统而言, 函数的资源消耗并非确定性的, 而是与使用方式密切相关, 因而往往导致回归分析的结果不够理想.

Sun 试图通过考虑资源消耗估算值的敏感度, 解决资源消耗的不确定性问题[26]. 他们使用最小二乘法和随机系数方法 (Random coefficients method, RCM) 进行回归分析函数的资源消耗, 并利用仿真的方法验证其方法的正确性. 他们使用确定性的、服从正态分布的和服从指数分布的方法生成资源消耗, RCM 用于克服最小二乘法存在的不确定性问题. 但是, 在超过 5 个待估算函数, 且资源消耗逐渐不确定时, 这两个方法都会失效[6].

最近有研究基于性能模型估算资源消耗[27−29]. 这类方法的特点是利用性能模型的预测结果与可以直接度量的性能指标进行比较, 然后逐步修正实际的资源消耗. 使用的前提是性能模型已经正确的建模, 然后通过分析模型中各个分量对性能的影响调整参数. 但是, 这些方法将性能模型限定到某种特定的排队论模型, 而这些模型针对的是面向服务器的建模, 并不适用于描述基于组件的性能模型. Zheng 和 Woodside 等的方法[30−31] 虽然可以适用于不同类型的性能模型, 但是由于该方法需要分析每个分量的扰动对性能模型的影响, 所以一次迭代的计算复杂度等于组件个数乘以性能模型求解时间. 通常, 对于细粒度组件化的性能模型, 求解的复杂度会很高. 且随着系统的复杂, 求解时间也会增长, 导致其总计算开销非常庞大, 不适用于细粒度模型的快速计算. 而且由于性能模型的估算误差通常较大, 其计算精度也会受到影响.

6 总结

用户请求系统的方式不同, 导致响应用户请求的服务发生变化, 而服务的变化又将导致资源需求的变化. 因而需要细粒度地分析用户的请求和服务的资源需求. 但服务的资源需求由于周期短, 难以直接度量. 传统基于回归的方法, 又难以克服多元共线性和不确定性问题. 本文提出了一种基于 Kalman 滤波的服务时间估算方法, 主要贡献包括: 1) 在

Kalman 滤波的基础上建立了资源需求与可观测值之间的函数关系; 2) 给出了 Kalman 滤波迭代过程中参数设置的方法. 实验结果表明, 估算精度比回归分析的方法具有很大的提高.

References

1 JVM Tool Interface (JVMTI) [Online], available: http://~java.sun.com/j2se/1.5.0/docs/guide/jvmti/, September 3, 2013

2 Jordan M, Czajkowski G, Kouklinski K, Skinner G. Extending a J2EE server with dynamic and flexible resource management. In: Proceedings of the 2004 International Conference on Middleware. Toronto, Canada: ACM, 2004. 439−458

3 Binder W, Hulaas J. A portable CPU-management framework for Java. *IEEE Internet Computing*, 2004, **8**(5): 74−83

4 Hulaas J, Kalas D. Monitoring of resource consumption in Java-based application servers. In: Proceedings of OpenView University Association 10th Workshop. Geneva, Switzerland: University of Geneva, 2003. 1−6

5 Islam S, Keung J, Lee K, Liu A. Empirical prediction models for adaptive resource provisioning in the cloud. *Future Generation Computer Systems*, 2012, **28**(1): 155−162

6 Kalbasi A, Krishnamurthy D, Rolia J, Dawson S. DEC: service demand estimation with confidence. *IEEE Transactions on Software Engineering*, 2012, **38**(3): 561−578

7 Teunissen P J G, Khodabandeh A. BLUE, BLUP and the Kalman filter: some new results. *Journal of Geodesy*, 2013, **87**(5): 461−473

8 You Ke-You, Xie Li-Hua. Survey of recent progress in networked control systems. *Acta Automatica Sinica*, 2013, **39**(2): 101−118
(游科友, 谢立华. 网络控制系统的最新研究综述. 自动化学报, 2013, **39**(2): 101−118)

9 A Transactional Web e-Commerce Benchmark [Online], available: http://www.tpc.org/tpcw/default.asp, September 3, 2013

10 QoS Oriented B2C Benchmark for Internet Middleware [Online], available: http://forge.ow2.org/projects/jaspte/, September 3, 2013

11 Kalman R E. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 1960, **82**(1): 35−45

12 Tao Gui-Li, Deng Zi-Li. Self-tuning fusion Kalman filter with unknown parameters and its convergence. *Acta Automatica Sinica*, 2012, **38**(1): 109−119
(陶贵丽, 邓自立. 含未知参数的自校正融合 Kalman 滤波器及其收敛性. 自动化学报, 2012, **38**(1): 109−119)

13 Lazowska E D, Zahorjan J, Graham G S, Sevcik K C. *Quantitative System Performance: Computer System Analysis Using Queueing Network Models*. Upper Saddle River, NJ, USA: Prentice-Hall, 1984

14 Hou Zhen-Wei, Fang Hai-Tao. *L*2-stability of discrete-time Kalman filter with random coefficients under incorrect covariance. *Acta Automatica Sinica*, 2013, **39**(1): 43−52
(周振威, 方海涛. 在不准确方差下带随机系数矩阵的卡尔曼滤波稳定性. 自动化学报, 2013, **39**(1): 43−52)

15 Rao J, Wei Y D, Gong J Y, Xu C Z. QoS guarantees and service differentiation for dynamic cloud applications. *IEEE Transactions on Network and Service Management*, 2013, **10**(1): 43−55

16 Malek O, Venetsanopoulos A, Alamgir L, Alirezaie J, Krishnan S. A discrete-time convergence model for proliferationable stem cell and its estimation using Kalman filter. *Journal of Bioengineer and Biomedical Sciences*, 2013, **3**(1): 1−10

17 You K Y, Xie L H. Kalman filtering with scheduled measurements. *IEEE Transactions on Signal Processing*, 2013, **61**(6): 1520−1530

18 Load Balancing for EJBs and RMI Objects [Online], available: http://~docs.oracle.com/cd/E13222_01/wls/docs90/cluster/~load_balancing.html, September 3, 2013

19 Product Overview — Software and IT Management Products [Online], available: http://~www.quest.com/jprobe/, September 3, 2013

20 Simon Spinner. Evaluating Approaches to Resource Demand Estimation [Master dissertation], University of the State of Baden-Wuerttemberg and National Laboratory of the Helmholtz Association, Germany, 2011

21 Bard Y, Shatzoff M. Statistical methods in computer performance analysis. *Current Trends in Programming Methodology*. New Jersey: Prentice-Hall, 1978, **3**: 1−51

22 Han R, Guo L, Ghanem M M, Guo Y K. Lightweight resource scaling for cloud applications. In: Proceedings of the 2012 Cluster, Cloud and Grid Computing. Ottawa, CA: IEEE, 2012, 644−651

23 Pacifici G, Segmuller W, Spreitzer M, Tantawi A. CPU demand for web serving: measurement analysis and dynamic estimation. *Performance Evaluation*, 2008, **65**(6−7): 531−553

24 Rolia J, Vetland V. Parameter estimation for performance models of distributed application systems. In: Proceedings of the 1995 Centre for Advanced Studies on Collaborative Research Conference. Toronto, CA: ACM, 1995. 54−63

25 Rolia J, Kalbasi A, Krishnamurthy D, Dawson S. Resource demand modeling for multi-tier services. In: Proceedings of the 1st joint WOSP/SIPEW International Conference on Performance Engineering. New York, USA: ACM, 2010. 207−216

26 Sun X. Estimating Resource Demands for Application Services [Master dissertation], Carleton University, Canada, 1999

27 Lu Y, Abdelzaher T, Lu C Y, Sha L, Liu X. Feedback control with queueing-theoretic prediction for relative delay guarantees in Web servers. In: Proceedings of the 2003 Real Time and Embedded Technology and Applications Symposium. Toronto, Canada: IEEE, 2003. 208−217

28 Kraft S, Pacheco-Sanchez S, Casale G, Dawson S. Estimating service resource consumption from response time measurements. In: Proceedings of the 2009 Performance Evaluation Methodologies and Tools. Coleraine, UK: ACM, 2009, 1−10

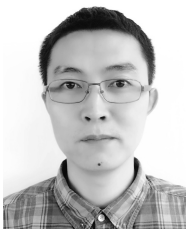
29 Menasce D. Computing missing service demand parameters for performance models. In: Proceedings of the 2008 Computer Measurement Group. Las Vegas, USA: CiteSeer, 2008. 241−248

30 Zheng T, Woodside M. Performance model estimation and tracking using optimal filters. *IEEE Transactions on Software Engineering*, 2008, **34**(3): 391−406

31 Woodside M, Zhen T, Litoiu M. Service system resource management based on a tracked layered performance model. In: Proceedings of the 2006 International Conference on Autonomic Computing. Washington, USA: IEEE, 2006. 175−184

黄 翔 中国能源建设集团广东省电力设计研究院博士后. 2012 年获得中国科学院软件研究所博士学位. 主要研究方向为云计算和大数据. 本文通信作者.
E-mail: huangxiang@gedi.com.cn
(**HUANG Xiang** Postdoctor at Guangdong Electric Power Design Institute, China Energy Engineering Group. He received his Ph. D. degree from the Institute of Software, Chinese Academy of Sciences in 2012. His research interest covers cloud computing and big data. Corresponding author of this paper.)

陈 伟 中国科学院软件研究所助理研究员. 2013 年获得中国科学院博士学位. 主要研究方向为网络分布式计算, 软件工程.
E-mail: wchen@otcaix.iscas.ac.cn
(**CHEN Wei** Assistant professor at the Institute of Software, Chinese Academy of Sciences. He received his Ph. D. degree from Chinese Academy of Sciences in 2013. His research interest covers distributed network computing and software engineering.)



宋云奎 中国科学院软件研究所软件工程技术研究中心助理研究员. 2009 年获得中国科学院硕士学位. 主要研究方向为网络分布式计算, 软件工程.
E-mail: songyk@otcaix.iscas.ac.cn
(**SONG Yun-Kui** Assistant professor at the Institute of Software, Chinese Academy of Sciences. He received his master degree from Chinese Academy of Sciences in 2009. His research interest covers distributed network computing and software engineering.)

陈志刚 中国能源建设集团广东省电力设计研究院副总工程师. 1992 年获得华中理工大学硕士. 主要研究方向为系统规划.
E-mail: chenzhigang@gedi.com.cn
(**CHEN Zhi-Gang** Deputy chief engineer at Guangdong Electric Power Design Institute, China Energy Engineering Group. He received his master degree from Huazhong University of Science and Technology in 1992. His main research interest is system planning.)