

基于 SLA 约束的 IaaS 虚拟机的多维资源放置算法

吴运涛¹, 魏恒义², 骆登欣¹, 朱正东²

(1 西安交通大学 软件学院, 陕西 西安 710049; 2 西安交通大学 电子与信息工程院, 陕西 西安 710049)

摘 要: 针对 IaaS 层资源放置中的虚机放置问题, 提出了 IaaS 云资源放置 SLA 的服务质量指标模型和基于 QoS 的多维度差分进化放置算法。算法采用差分进化算法结合非支配排序思想, 通过对非支配排序在排序方式以及子代筛选上进行优化, 实现对 IaaS 的资源放置时兼顾多种类资源约束以及其他服务质量指标约束, 从而获得更合理的多类型资源放置结果。实验结果表明本文提出的多维资源虚机放置算法搜索速度快, 多维资源放置结果产生的违约率更低, 在 QoS 指标上具备更优的性能。

关键词: IaaS 层资源; SLA; 非支配排序; 虚机放置

中图分类号: TP391.4

文献标识码: A

文章编号: 1000-7180(2017)10-0005-06

Multidimensional Resource Placement Algorithm for IaaS Virtual Machine Based on SLA Constraint

WU Yun-tao¹, WEI Heng-yi², LUO Deng-xin¹, ZHU Zheng-dong²

(1 School of Software, Xi'an Jiaotong University, Xi'an 710049, China;

2 School of the Electronic and Information Engineering, Xi'an Jiaotong University, Xi'an 710049, China)

Abstract: Resource placement of IaaS service model is essential to the cloud infrastructure resource placement. This paper studies the virtual machine placement problem in the resource placement of the IaaS layer, presents quality of service model of the SLA in IaaS cloud resource placement, and then propose a multi-dimensional differential evolutionary algorithm base on QoS. The Algorithm combined with the non-dominate sorting and the differential evolution algorithm, and optimized speed sorting and filtering on progeny in the non-dominate, achieve taking into account the many types of resource constraints and other objective constraints when the resource placement of IaaS. Obtain more reasonable multiple types of resource placement result accordingly. Finally, the simulation results show that, to compared with other traditional heuristic algorithms, the proposed multi-dimensional resource VM placement algorithm is more faster in search, meanwhile the final result of the search get the minimum default rate, it has better performance on QoS metric, thus verifying the validity of algorithm.

Key words: IaaS resource; SLA; Non-dominated sorting; VM placement

1 引言

云计算中基础架构服务 IaaS 软件将客户所需的软硬件资源(CPU, 内存, 网络, 存储等)以虚拟机的形式提供给平台服务 PaaS 层和软件服务 SaaS 层。硬件资源到虚拟机的映射放置算法直接影响云计算的性能和服务质量。

IaaS 资源放置大体上分为静态放置与动态放置^[1], 云计算主要采用动态放置策略进行放置。R. Buyya 等人提出基于节省能耗的资源放置流程, 给出了一个基于资源是否闲置的能耗资源安置的优化方案^[2]。A. verma 等人建立了基于资源利用能耗模型和迁移成本模型, 提出一种启发式的能耗最小化和迁移成本最小化的安置算法^[3]。文献^[4]提出多目

标随机多组优化算法(RMRO),文献[5]提出基于NSGAIⅡ遗传算法的多目标优化算法,文献[6]中采用二阶段启发式算法对虚机的放置在能耗和网络性能上进行优化.

本文综合考虑硬件资源的利用率和云服务的服务质量 QoS,构建了基于服务等级协议 SLA(Service-Level Agreement)约束的 IaaS 虚拟机多维资源放置算法,该算法根据 SLA 协议对服务指标进行量化和制定,建立的参数模型作为资源放置的目标模型,对资源放置过程进行约束指引,促进资源的合理放置达到良好的服务交付水平.

2 基于 SLA 的 IaaS 资源放置模型

服务等级协议 SLA 从客户的角度出发,把云服务商承诺的服务品质(服务质量 QoS)进行量化,可以保证用户获得的服务质量. SLA 内容包括服务指标、违约处理以及交流规则,不同层次的资源放置具有不同的量化指标. 本文在云服务 IaaS 资源放置层次引入 SLA 量化模型,并通过相应的资源放置算法保证用户虚拟机的资源和服务质量承诺. 表 1 给出了 IaaS 资源放置中引入的 SLA 参数与服务质量的指标 QoS 的关系.

表 1 SLA 参数与 QoS 性能关系表

SLA 输入参数	性能指标	影响 QoS 指标
内存资源	内存大小	资源利用率、违约率
计算资源	CPU 主频	能耗、资源利用率、违约率
网络资源	网络带宽	响应时间、资源利用率、违约率
	延迟	响应时间
其他服务性能阈值	高低阈值	违约率、负载均衡、收益
	权重	负载均衡

假设用户请求的模型为 R : $R = \{N, \text{input}, \text{result}\}$, 其中 N 表示用户的数量集; input 代表用户输入参数,其中包含请求的物理资源量和 SLA 协议参数, result 代表返回的结果. $\text{input} = \{VM_1, VM_2, VM_3, \dots\}$, VM_i 为一次请求的资源模型.

系统以虚拟机 VM 为单位进行资源放置,其中除了要满足计算资源、存储资源和网络资源三个基本资源量外,还必须关注 SLA 参数承诺的违约率 VioRate、负载均衡 VLB 和服务平均响应时间 ART 约束. 资源放置中首要目标是先满足用户的基础资源请求,即减少资源放置 SLA 约束下的违约率,其次再考虑负载均衡和响应时间等其他指标.

(1) 服务平均响应时间 ART(单位:ms):

$$\begin{cases} \text{ART} = \text{RST} + \text{NTD} \\ \text{RST} = \frac{\sum_{i=1}^n \text{RST}_i}{n} \end{cases}$$

式中, RST 是平均资源放置时间; NTD 是网络传输时延;

RST_i 是虚拟机 i 的资源放置时间; n 是待分配虚拟机数. 资源调度时间 RST 体现了资源放置的效率,在资源放置中对响应时间的影响最大.

(2) 违约率 VioRate(单位: %):

$$\text{ViolRate} = N_{\text{vio}} / N_{\text{req_total}}$$

式中, N_{vio} 是违约请求数量; $N_{\text{req_total}}$ 是总请求数量.

(3) 负载均衡 VLB:

$$\begin{aligned} \text{VLB}_{i,r} &= \sqrt{\frac{1}{n-1} \sum_{i=1}^n (L_{i,r} - \bar{L}_{i,r})^2} \\ \text{VLB}_i &= P_{i,\text{cpu}} * \text{VLB}_{i,\text{cpu}} + P_{i,\text{mem}} * \text{VLB}_{i,\text{mem}} + \\ &\quad P_{i,\text{net}} * \text{VLB}_{i,\text{net}} \end{aligned}$$

式中, $P_{i,r}$ 是物理节点 i 的 r 类型资源的相应权重; $\text{VLB}_{i,r}$ 是物理节点 i 的 r 类型资源的负载均衡度; $L_{i,r}$ 表示物理节点 i 的 r 类型资源的利用率.

3 基于 SLA 的 IaaS 资源放置算法设计

设计有效的 IaaS 资源放置算法是基于 SLA 的 IaaS 资源放置模型求解的核心. 资源动态放置算法有模拟退火算法、蚁群算法和遗传算法等启发式算法. 差分进化算法(DEA)是基于种群启发式的随机搜索优化算法,具有受控参数少、较好的收敛性以及鲁棒性等优点,但也具有收敛过早且易陷入局部最优等问题,这也是本文给出算法要解决的问题.

3.1 算法描述

基本差分进化算的初始化、变异、交叉以及选择等操作算子影响着算法的稳定性与效率,为使算法尽量避免早熟收敛,并获得理想的资源放置效果,本文需结合非支配排序分层的思想对算子进行优化,使计算结果更优.

(1) 改进后的差分进化算法步骤

第 1 步: 初始化种群 P_g , 对种群每个个体随机生成.

第 2 步: 判断是否达到最大进化代数,若是则跳转 9, 否则继续执行.

第 3 步: 对种群进行变异、交叉, 形成临时种群 R_g .

第 4 步: 种群 P_g 与种群 R_g 合并形成数量为 $2N$

的种群 Q ,对 Q 初始化非支配解集 F_i ,记 i 为 0.

第 5 步:根据新子代种群 P_{g+1} 个体数量是否达到 N ,跳转到 7 步.

第 6 步:将非支配解集 F_i 填入 P_{g+1} 中,为了有效利用种群空间,对 F_i 有概率性地剔除部分过于靠近的个体,采用最小个体距离来筛选出局个体,剩下个体将放入 P_{g+1} . 重新在种群 $Q - F_i$ 的剩余个体中根据多个约束函数计算并求解新的非支配解集 F_{i+1} ,回到第 5 步.

第 7 步:对最后一批超过种群 P_{g+1} 剩余个体数的非支配解 F_i 计算相应的适应值最小个体距离并剔除适当数量的个体,并将剩余部分的个体填入种群 P_{g+1} ,使得 $|P_{g+1}| = N$.

第 8 步:采用精英策略对种群 P_{g+1} 与 P_g 个体依次选出较优的个体,将在较优的个体纳入 P_{g+1} 形成新父代种群,跳转到第 2 步.

第 9 步:计算该最优解的适应度值,结束流程.

算法编码及实现

(2) 算法编码及实现

在 IaaS 的云平台中,云系统拥有 m 个物理服务器 $\{PM_1, PM_1, PM_1, \dots, PM_m\}$, $PM_i = \{C_i^{cpu}, C_i^{mem}, C_i^{net}\}$, 每个物理服务器具有各自的资源容量,这里主要考察三种资源类型指标:cpu、内存 mem 和网络带宽 net.

为了能将差分进化算法应用到 $PM = \{PM_i | 0 \leq i \leq m\}$ 和 $VM = \{VM_i | 0 \leq i \leq n\}$,需对虚拟机列表和物理机列表在算法中进行映射. 由于 $\{VM_i | 0 \leq i \leq n\}$ 的目的是为了能够放置到合适的 PM 上,首先将从系统中检测到的可用 PM 列表放置到数组中,这样每个 PM 就可对应到数组中的一个具体的下标索引.

通过将每组所有 VM 对应到 PM 下标值组成的数组作为一个解来建立模型. 可以设定:

(1) 所有 VM 对应到 PM 下标值组成的数组作为一个可能解,并看作是一个个体.

(2) 每个个体具体的“基因”是可能解中相应下标的值,该值对应 $\{PM\}$ 中的具体 PM 位置.

(3) 多个由数组表示的个体组成一个种群,该种群可用二维数组来进行表示,每行代表一个个体,每个具体单元值代表该行(某具体行)个体中该列(某具体列)的“基因”.

算法实现的伪代码如下:

input: population: 种群
maxGap: 最大进化代数

list_VM: 请求虚拟机列表

list_PM: 物理机列表

output: bestFitness: 最佳适应度

```

1  gap = 0 //年代
2  While(gap < maxGap)
3      For each indual in population do
4          calculate_fitness(indual)
5      Endfor
6      If (check_Density(population) < Density_indicator)
7          then
8              mutation_Random(population) //个体变异
9          Endif
10         tempPop ← {} //临时种群
11         For each indual in poplation do //变异
12             tempPop. put(mutation_Random(indual))
13         Endfor
14         For each indual in population do //交叉
15             For each indualTemp in tempPop do
16                 indualTemp ← cross_Random(indual, indualTemp)
17             Endfor
18         Endfor
19         newpop ← {}
20         mergePop ← merge(pop, tempPop)
21         While(newpop. size() < N)
22             nextDomain ← getNondomain(mergePop) //获取非支配解
23             If (nextDomain. size() < N - newpop. size())
24                 then
25                     newpop. add(nextDomain)
26                 Endif
27             EndWhile
28             newpop. add(getLargeDistance(nextDomain)) //距离大的个体进入种群
29             population ← elitist(population, newpop) //选择较优个体组成新种群
30             gap++
31         EndWhile
32         allocation(Transboundary_VM) //根据结果放置虚拟机
33         //重新计算最后适应值
34         bestFitness += calculate_fitness(Transboundary_VM, new_PM)
35     return bestFitness
36 End

```

3.2 算法算子优化设计

(1) 变异算子

改进的差分进化算法的变异算子采用 DE/

best/2/bin 策略,即在每次变异中都有最优值个体的参与,这样能够弥补差分进化算法在进化初期的收敛速度慢的情况. DE/best/2/bin 策略因子如下公式:

$$v_{ij}^{g+1} = u_{best}^g + F \cdot (u_{ij}^g - u_{cj}^g)$$

式中: v_{ij}^{g+1} 是第 $g+1$ 代变异个体 i 的第 j 个基因; F 是变异因子; u_{best}^g 是第 g 代个体 a 的第 j 个基因; u_{ij}^g 是第 g 代个体 b 的第 j 个基因; u_{cj}^g 是第 g 代个体 c 的第 j 个基因.

(2) 交叉算子

交叉算子是影响算法种群多样性的重要因素之一,基本差分进化算法的交叉因子一般是静态的,即一旦初始值定义完成后,后续的进化过程中不会随进化代数的变化而变化. 为了让交叉算子能够在进化过程中进行适应性地交叉概率的调整,将该交叉因子 CR 设计为:

$$CR = CR_{max} - (G - x)^2 \cdot \frac{(CR_{max} - CR_{min})}{G^2}$$

式中: CR_{max} 是交叉概率最大值; CR_{min} 是交叉概率最小值; G 是最大进化代数; x 是当前进化代数.

交叉因子改为动态性后,变异基因进入种群机会会随着进化代数增大而增大, CR 采用以进化代数的平方函数形式表示,越到后期种群中变异个体的概率会迅速增大,将使进化后期的种群依然保持一定程度的多样性.

(3) 选择算子

非支配解集作用在选择算子上,非支配排序选择是实现多目标优化的关键步骤,多目标优化算法和其他启发式算法一样都面临快速收敛问题与全局搜索问题.

为了提高非支配排序速度,本文将采用种群分块的方式对非支配解进行快速排序,由于最终结果只需要原种群个体数 N ,而待排种群 Q 则有 $2N$ 个个体,这时可对种群 Q 分为两部分筛选解集. 可以同时求解这两部分各自的非支配解集且互不影响,一次筛选过后将有两部分的非支配解集 Z' 、 Z'' ,然后对这两部分合并再求一次非支配解集,最终的非支配解集则为整个种群 Q 的第一批非支配解集 Z_0 . 通过这样的方式减少了非支配排序的比较次数同时可以很快地求出非支配解集. 非支配快速排序示意图如图 1 所示.

在对子种群求非支配解集 Z' 时可以采用擂台法进行快速筛选非支配解,擂台法利用个体两两对比,满足非支配条件的个体则放入非支配解集 Z' ,

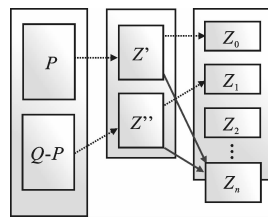


图 1 非支配快速排序示意图

并将 Z' 里的个体继续与其他未被比较个体进行比较,只要有一次不满足非支配约束则剔除出非支配解集 Z' ,通过擂台法实现单次非支配排序快速筛选的比较次数相比将会明显减少.

为防止早熟收敛,需要合理地在非支配层的解集上筛选并去掉相应的个体,使当前最优非支配解集 Z_0 能够均匀分布,以最大效率利用有限的种群空间,同时扩大全局搜索能力. 本文采用区别于适应值密度的方法,通过比较最小个体距离来对非支配解集个体进行筛选. 图 2 是以仅涉及到在二维约束下的种群分布以及对非支配层解集的筛选示例.

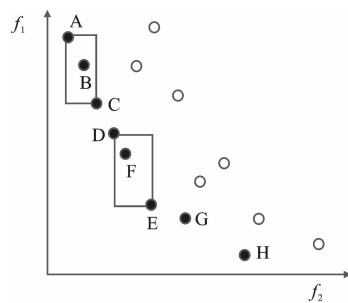


图 2 二维约束种群分布示意图

图 2 中黑点表示当前最优非支配解集 Z_0 的个体,可以看出,若采用密度来筛选,有可能更合适的点 B 被除掉而点 F 会留下,而本文算法在筛选过密的节点上首先考虑个体间的最近距离的非支配个体而非个体的综合密度,因此可以采用个体间在有序目标函数值上的前后相邻两个个体的最小函数差值作为衡量个体的密度.

在非支配解集中,首先根据这些目标函数对非支配解中个体的单个目标函数值进行排序,其次对有序的前后个体间进行两两比较,并根据前述的个体单目标函数的密度求法求出非支配解集个体在该目标函数上的密度,然后将这些个体在所有目标函数上的密度值进行叠加,作为个体最后的密度. 最后对非支配解集 Z_0 个体根据最后的密度再进行一次排序,在密度排序后的解集中淘汰相应数量的最小密度个体, Z_0 中剩下的将是分布较疏的个体.

4 实验结果与分析

4.1 实验环境

实验环境采用 CloudSim 工具^[7]进行仿真模拟,CloudSim 是一个通用、支持无缝模拟和建模的云计算仿真软件,它是由澳大利亚墨尔本大学开发的一个开源仿真器。

4.2 放置算法性能验证实验

为验证改进算法的性能,将改进后的差分进化算法与原差分进化算法以及具有多目标寻优的 NSGA 算法进行对比,主要从算法的收敛速度、收敛结果和违约结果上进行对比。为确保公平性,输入数据分别在多个算法中进行多次实验,在所有测试完成后进行汇总取平均值作为结果。

(1)实验一,单目标函数下算法在收敛速度与收敛结果的综合对比。

实验采用进化代数 400 代,变异算子 F 为 0.5,交叉算子交叉因子为 CR_{max} 为 0.7, CR_{min} 为 0.3。

图 3 给出单目标实验对比曲线说明,改进后的算法的曲线转折点较早,说明改进后的算法具有较好的收敛性,即收敛速度快。另外还具有较低的指标值,表明比其他算法具有更好的全局搜索能力。而随着进化代数的增加,三种算法均能收敛到最佳的函数值。

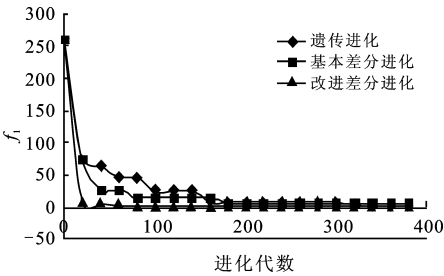


图 3 单目标实验对比

(2)实验二,不同资源规模情况下算法的收敛速度对比。

图 4 给出不同资源规模情况下首次收敛代数对比柱状说明,改进的算法在首次收敛代数上优于 NSGA 算法,而差分进化算法收敛速度最快,主要因为差分进化只考虑单目标的优化,因此计算复杂度与其他算法相比最低,而多目标寻优比单目标寻优要更耗时一些。

(3)实验三,不同资源规模情况下算法的收敛结果的对比。

图 5 给出不同资源规模情况下适应值 f_1 收敛

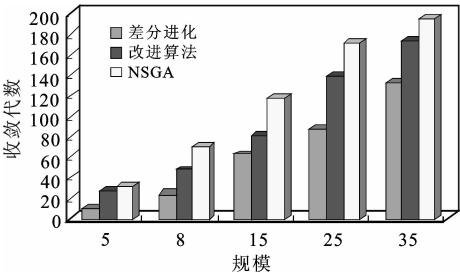


图 4 首次收敛代数对比

对比曲线说明,多目标环境下差分进化算法并没有得到很好的收敛结果,优化效果较差。改进算法与 NSGA 多目标算法在适应值收敛上相比,从图中可看出略优于 NSGA 算法,曲线增长趋势也与 NSGA 相近,这得益于在进化中改进算法比 NSGA 具有更合理的淘汰策略,通过增长次一级的非支配解增大了收敛时的搜索面积。

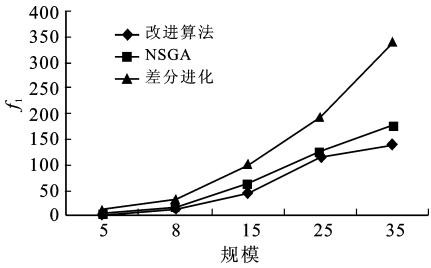


图 5 适应值 f_1 收敛对比

(4)实验四,多维约束情况下算法的违约结果分布对比。

图 6 给出违约数占比分布对比说明,改进后的算法在违约数量分布方面在 0-2 区间上占比较大,而 NSGA 算法则多分布在 1-2 区间上,传统差分进化算法的结果分布更为靠后。由此看出违约数量方面改进后的算法出现的违约情况比其他两个算法更小。

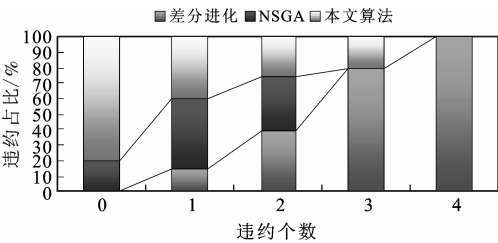


图 6 违约数占比分布对比图

4 个实验验证了改进算法的主要性能,通过在收敛速度、收敛结果和违约分布结果上的对比显示,改进算法具有更好的目标寻优结果。尤其是在多目标优化中,改进算法比原 NSGA 算法具备更广的全

局搜索和更快的收敛速度.

5 结束语

本文针对 IaaS 层虚拟机放置问题,给出了基于 SLA 的 QoS 相应指标,并在此基础上提出了基于 QoS 多维度改进的差分进化算法为资源放置算法.算法以差分进化算法为原型,结合 NSGA 的非支配排序思想,通过对非支配排序的改进,优化收敛速度和全局搜索,实现多维约束资源的放置.实验通过将 QoS 指标作为算法的相应约束,验证了本文算法在资源放置过程中可以提高虚拟机的放置效率与放置效果.

参考文献:

[1] 吴海双,张亮,李杰辉,等. IaaS 环境中一种保证 SLA 的资源调度策略[J]. 计算机工程,2013,38(7):51-54.
[2] Buyya R, Beloglazov A, Abawaiy J H. Energy-efficient management of data center resources for cloud computing: a vision, architectural elements, and open challenges[J]. Eprint Arxiv, 2010, 12(4):6-17.
[3] Verma A, Ahuja P, Neogi A. pMapper: Power and migration cost aware application placement in virtual-

ized systems[C]//International Conference on Middleware. New York, NY, USA,2008.
[4] 王新杰,雷印杰,严华,等. 云计算中面向能耗降低的虚拟机多资源放置算法[J]. 计算机测量与控制,2015,22(12):4133-4137. DOI:10.1109/ICACCS.2013.6938749.
[5] 李强,郝沁汾,肖利民,等. 云计算中虚拟机放置的自适应管理与多目标优化[J]. 计算机学报,2011, 34(12):2253-2256.
[6] 董健康,王洪波,李阳阳,等. IaaS 环境下改进能源效率和网络性能的虚拟机放置方法[J]. 通信学报,2014,34(1):72-80.
[7] Rodrigo N Calheiros, Rajiv Ranjan. Cloudsim:a novel framework for modeling and simulation of cloud computing infrastructures and services [R]. Australia, The University of Melbourne, 2010.

作者简介:

吴运涛 男,(1991-),硕士. 研究方向为移动云计算. E-mail:sr71tao@sina.com.
魏恒义 男,(1958-),高级工程师,硕士生导师. 研究方向为网络及分布式计算.
骆登欣 男,(1994-),硕士. 研究方向为移动云计算.
朱正东 男,(1963-),高级工程师. 研究方向为高性能计算.

(上接第 4 页)

[4] Moorandi D, Sicari S, Pellegrini F D, et al. Internet of things: vision, applications and research challenges [J]. Ad Hoc Networks, 2012, 10(9): 1497-1516.
[5] Gubbi I, Buyya R, Marusic S, et al. Internet of Things (IoT): a vision, architectural elements, and future directions [J]. Future Generation Computer Systems, 2013, 29(9):1645-1660.
[6] Piguet C, Renaudin M, Omnes J T, et al. Special session on low-power systems on chips (SoCs) [C] // Design, Automation and Test in Europe. Conference and Exhibition. Germany, Munich, 2001: 488-494.
[7] Keating M, Flynn D, Aitken R, et al. Low power methodology manual for system-on-chip design [M]. New York, USA: Springer, 2007: 33-40.
[8] Singhal V K, Menezes V, Chakravarthy S, et al. A

10.5 uA/MHz at 16 MHz single-cycle non-volatile memory access microcontroller with full state retention at 108nm in a 90nm process [C] // International of Solid-State Circuit Conference. San Francisco, CA, USA, 2015:1-3.

作者简介:

黄泽林 男,(1991-),硕士研究生. 研究方向为低功耗 SoC 设计.
乔树山(通讯作者) 男,(1981-),博士,副研究员. 研究方向为低功耗数字集成电路、电力线通信等. E-mail:qiaoshushan@ime.ac.cn.
袁 甲 男,(1985-),博士. 研究方向为极低功耗数字集成电路的设计方法.