

基于预取的 Cache 替换策略

孙玉强,王文闻,巢碧霞,顾玉宛

(常州大学 信息科学与工程学院,江苏 常州 213164)

摘 要: 在页式虚拟存储管理系统中,合理的 Cache 替换策略可以减少 CPU 等待时间,提升 Cache 命中率和系统性能.许多经典 Cache 置换算法的依据都是页面的访问次数或访问时间,也有一些是两者兼顾.对此介绍了一种基于预取的 Cache 替换策略,主要做出了以下 2 点优化:(1)一次 Cache 置换淘汰多个页面;(2)Cache 首次调入页面由“拉取”模式改为“推送”模式.模拟实验结果显示,该方法有效提高了 Cache 命中率,提升了系统性能.

关键词: 预取;替换策略;Cache;命中率

中图分类号: TP316

文献标识码: A

文章编号: 1000-7180(2017)01-0085-05

Cache Replacement Policy Based on Prefetch

SUN Yu-qiang, WANG Wen-wen, CHAO Bi-xia, GU Yu-wan

(School of Information Science & Engineering, Changzhou University, Changzhou 213164, China)

Abstract: In page-virtual storage management system, So reasonable Cache replacement policy can reduce the waiting time of CPU and improve system performance. Many typical Cache replacement policies are based on the accessing frequency or time of the page and some policies do both. This article will describe a Cache replacement strategy which is based on Prefetch. We mainly make the following two optimization: (1) eliminate multiple pages in one Cache page replacement; (2) change the distribution model of the first batch of tasks from the pull model to the push model. Simulation experiment shows that this method can effectively improve the hit rate of Cache.

Key words: prefetch; replacement policy; Cache; hit rate

1 引言

Cache 是位于 CPU 与内存之间的一种高速缓冲存储器,它有效解决了 CPU 与内存存取速度不匹配的问题,其主要理论依据就是程序局部性原理.在页式虚拟存储管理系统中,为了提高 Cache 的性能,研究者们提出了许多经典的页面置换算法,主要有先进先出算法(FIFO)、最近最少使用算法(LRU)、简单时钟页面算法(Clock)等^[1].

但对于数据密集型应用,具有海量、分布式、异构、更新快等特点,其局部性不足,不能按照传统的数据管理方法来管理这些数据.在这种情况下,预取^[2]技术就被提了出来.关于预取,国内外已有大量研究,如 Next-N-Line 预取、Wrong-path 预取、

Markov 预取和 Tagged Prefetch 等.这些预取技术各有所长,也各有其缺点,至今还没有哪一种预取策略被证明为是最优的.本文将介绍一种基于预取的 Cache 替换策略,利用预取来提高 Cache 命中率.

2 Cache 页面置换执行框架

2.1 Cache 页面替换流程

为了清楚说明本文所提出的优化方法,我们有必要分析一下现代计算机存储系统中 Cache 页面置换流程,如图 1 所示.

从 CPU 读取一条指令到该指令执行完毕,大致要经过以下 8 个步骤^[3].

(1)分析指令并获得虚拟地址 VA.

(2)到 TLB 中查找相应的页表数据,TLB 的页

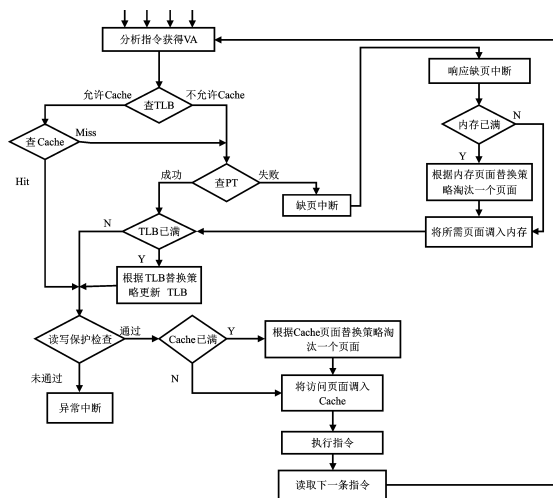


图1 Cache 页面置换流程图

表项中不仅保存着页面的物理基址,还包含了诸如读写访问权限、是否允许 Cache 等信息.若允许 Cache,则以 VA 为索引查找数据是否已经在 L1Cache、L2Cache 里,若 Cache 命中,则进行读写保护检查.

(3)若不允许 Cache,则查找页表 PT.如果查找成功,即所需页面在内存中,然后再判断 TLB 是否已满.如果 PT 中找不到所需页面,即所需页面不在内存中,产生缺页中断信号.

(4)若 TLB 已满,则根据相应算法,淘汰 TLB 中一个表项,将所需页的信息填入 TLB 表.否则,直接将所需页的信息填入 TLB 表即可.

(5)产生缺页中断后,缺页中断处理程序响应异常,先检查内存是否已满.若内存已满,则根据给定算法,从内存淘汰一个页面,再将所需页面从外存调入内存,更新 TLB 表项.否则,直接将所需页面调入内存即可.接下来,再进行读写保护检查.

(6)若通过读写保护检查,再判断 Cache 是否已满.否则,产生异常中断.

(7)若 Cache 已满,则根据页面置换算法,从 Cache 淘汰出一个页面,并将所需页面调入 Cache.否则,直接将所需页面调入 Cache 即可.

(8)执行指令,读取下一条指令.

2.2 Cache 和内存的页面置换分析

如图 1 所示,每次访存或是产生缺页中断时,内存和 Cache 都只淘汰替换了一个页面,等到下一次访存或缺页中断发生时又要进行一次页面置换.这样的工作方式增加了缺页中断的次数,由此带来的流水线停顿也大大降低了处理器的效率.解决这个问题的办法就是:在一次页面置换中淘汰尽可能多

的页面.

无法预知处理器将要访问哪些页面是发生缺页中断的重要因素.因为如果可以预知页面访问的顺序,就可以在页面被访问之前将页面调入内存,这样就能降低缺页次数.虽然有些系统运用预取策略,如上文提到的 Next-N-Line 预取,在产生缺页中断时不仅将所访问的页面调入内存,同时根据程序局部性原理,将相邻的几个物理块数据也调入内存,但是这样很可能取回大量无用数据,造成严重的 Cache 污染.

另一个问题就是,在页面置换算算法判断将哪些页面淘汰出去的时候,不能确定这样做是否最合理.也就是说,替换出去的页面会不会在不久的将来再次被访问,这是页面置换算算法所难以确定的.

2.3 访问请求延时分析

如图 1 所示,指令在执行过程中总会出现以下两种情况.

(1)CPU 在执行第一条指令时,先查找 Cache,此时 Cache 为空,再到内存中查找,内存也为空,发生缺页中断.系统响应中断,将所需页面调入内存,同时也将此页面调入 Cache.

(2)CPU 执行某指令时,若 Cache 未命中,PT 命中,则将所需页面调入 Cache.

以上两种情况虽然发生在不同时机,但却有一个共同点:Cache 未命中.无论何种状况,此时 Cache 都要从内存“拉取”数据,这一过程不利于隐藏访问延迟.针对第一种情况,如果能在 CPU 访问首个页面之前就将此页面调入内存,并且在 Cache“拉取”数据之前主动调入 Cache,改“拉取”模式为“推送”模式,处理器效率将得到提高;对于第二种情况,Cache 未命中时就要从内存读取数据,而内存的访问速度远远慢于 Cache,如能进一步减少访存次数,将大大提升系统性能.

若想进一步减少缺页次数,解决方案就是:在 CPU 访问页面之前就将所要访问的页面调入内存和 Cache,这就是本文所要介绍的基于预取的 Cache 页面置换策略.这样做带来的另外一个好处就是,将大幅度减少访存次数.以上两个问题的优化方案在下一节预取策略的设计中给出.

3 基于预取的 Cache 替换策略

3.1 行为级设计

虽然严格来说页缺失并不算是一种错误,但由此带来的缺页中断却降低了 CPU 的效率.为了能

够让 CPU 一直高效运行,也提高存储器的性能,由此想到应用预取技术将接下来要访问的页面提前调入内存。

Cache 页面更新:查找最佳淘汰页面. 执行预取处理时,若页面已存在于 Cache 中,即此页面在不久的将来还要被访问,则标识此页面. 在进行 Cache 页面更新时,那些没有被标识的页面就是最佳淘汰页面,但是否全部置换,要视具体情况而定. 淘汰了那些最佳淘汰页面,将内存缓存队列中的多个页面调入 Cache 即完成了一次 Cache 页面更新.

算法基本思想如下所示.

情况 1: CPU 读取第一条指令,预取处理计算出接下来将要访问的页面,将这些页面调入内存和 Cache. 若 CPU 第一次访问 Cache 时未能找到相应页面, CPU 进入等待状态,并周期性询问 Cache. 直到页面调入 Cache, CPU 读取页面,并将这些页面从内存删除,以便内存中可存放更多预取页面,实现 Cache 首次装载页面由“拉取”模式转变为“推送”模式.

情况 2: 若某个页面已经在 Cache 中,则为此页面添加一个标志,表明此页面在下次 Cache 页面更新时不需被更新;若页面已经在内存中,则不需要再预取此页面.

情况 3: 若 Cache 不能再装载新的页面,则新的页面就暂时存放在内存中,实现时可用队列缓存这些页面,以便将来 Cache 页面更新时读取;若内存中不能再装载新的页面,则停止预取.

情况 4: 若处理器要访问的页面不在 Cache 中,需从内存中读取数据时,如果内存中有足够多预取页面,则 CPU 一次性从内存缓存队列中将除去被标志为不需更新的多个页面调入 Cache,即完成一次 Cache 页面更新;如果内存中没有足够多预取页面,可根据相关 LUR^[4] 算法淘汰部分页面,只更新内存中现有的页面,而那些暂时不能被替换出去的页面就留在 Cache 中,等待下一次 Cache 页面更新;如果内存中没有所需页面,即没有预取到页面,此时 CPU 进入等待状态. 如此,就在很大程度上减少了访存次数.

情况 5: 每次 TLB 总是一次性更新多个表项,这些更新的表项就是 Cache 所更新的页面信息.

优化后的基于预取的 Cache 页面置换流程图如图 2 所示. 从图 2 可以看出,在取指和译码阶段,由于预取处理部分和 CPU 处理部分执行了相似的流程,所以预取时得到的数据可以共享给 CPU,加快

CPU 处理速度.

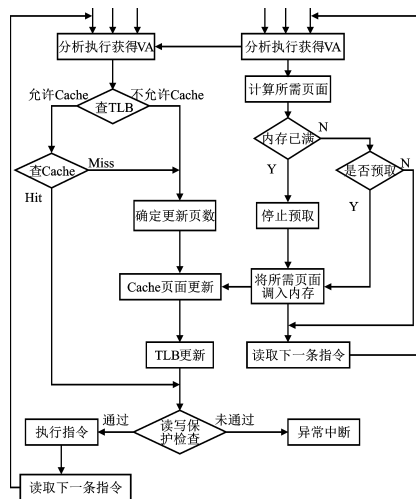


图 2 基于预取的 Cache 页面置换流程图

现在有两个问题需要解决:

(1) 若预取的页面在 Cache 中,如何确定此页面是否需要被标记,在下次 Cache 更新时不必被更新.

定义: $C = \{0, 1, 2, 3, \dots, n\}$, 表示 Cache 中页面集合;

$HP = \{0, 1, 2, 3, \dots, k\}$, 表示缓存预取页面集合.

如果预取的页面在 Cache 中就将此页面标记,则那些很久以后才会被访问的页面将长期贮存于 Cache 中,而即将被访问的页面却不能调入 Cache,这将增加 Cache 页面更新的次数,不利于提升系统性能. 为了避免这种情况,按如下方法:

① $K \leq n$, 为此页面添加一个标志,在下次 Cache 页面更新中不需被更新;

② $K > n$, 将此页面复制到内存预取页面缓存队列,不标记此页面.

(2) 如何判定预取的页面是否已在 Cache 或内存中.

定义: $M = \{0, 1, 2, 3, \dots, m\}$, 表示内存中页面集合. 由于 Cache 空间大小远小于内存,所以这里的 $n < m$.

设 P 为预取的页面,若按顺序查找来判定 P 是否在 C 或 M 中,那些既不在 C 中也不在 M 中的页面需要从页表头查到表尾,这将浪费大量的时间和系统资源. 为了加快查找速度,使用 Hash 查找^[5],即在 C 、 M 中匹配 $\text{Hash}(C, P)$ 和 $\text{Hash}(M, P)$ ^[6]. 查找步骤如下所示.

Step1: 在 C 中查找 P . 若 $\text{Hash}(C, P) \neq$

NULL,页面在 Cache 中,不执行该页面的预取;若 $Hash(C,P) = NULL$, 页面不在 Cache 中,转 Step2.

Step2: 在 M 中查找 P . 若 $Hash(M,P) \neq NULL$,页面在内存中,不执行该页面的预取;若 $Hash(M,P) = NULL$,页面不在内存中,执行该页面的预取. 综上所述,则需要执行预取的页面可表示为:

$$P = \{p \mid Hash(C,P) = NULL \& \& Hash(M,P) = NULL\}$$

由于从外存读取数据的速度较之内存和 Cache 非常慢以及程序局部性原理,所以这样做的好处是不会预取回大量已经存在于 Cache 或内存中的页面,提高了预取的有效性.

对比图 1 可以看出,图 2 对 Cache 页面置换流程主要进行了两方面的优化:(1)在 Cache 失效时,不是采用单个页面置换的策略,而是在一次页面置换中淘汰多个页面;(2)首次预取到页面直接调入内存和 Cache,避免 CPU 首次访问 Cache 失效,再到内存读取页面,改 Cache 从内存“拉取”模式为从内存向 Cache “推送”模式.

但是由于条件分支指令的存在,有时无法精确预测指令的执行顺序,一种简单又不会加剧访问延迟的方法是,利用减少条件分支转移算法^[7]删减条件分支,执行时等待条件分支判定完毕,确定要执行的分支后,再按上述方案进行页面预取.

3.2 结构级设计

CPU 获取到第一条指令,预取处理部分把将要访问的页面从外存调入内存和 Cache,CPU 再访问相关页面. 如果遇到分支指令,根据判定结果,编译指令,预取页面,最后执行指令. 整个处理结构如图 3 所示,图中只包含了与预取相关的内容.

之所以发生缺页异常,是因为无法提前预知处理器将要访问哪些页面. 那么是否有办法能成功解决此问题呢? 我们不妨将视线转移到 CPU 上. CPU 每一次页面访问都是准确无误的,如果预取处理能够像 CPU 一样,得到 CPU 执行完一系列指令所访问的页面集合,那么就可以做到快速而精确的页面预取. 为此,预取处理部分须具备和 CPU 类似的结构,但并不需要做实际的数值运算,而只需获取数据所在地址即可. 例如,有指令 ADD AUGEND ADDEND,其功能是 $(AUGEND) = (AUGEND) + (ADDEND)$,那么预取处理在分析这条指令时只需得到 AUGEND 与 ADDEND 的地址即可,而并不

做具体的加法运算. 再加之程序的局部性原理,降低了页面预取速度不匹配 CPU 处理速度而造成流水线停顿的概率. 由以上分析可以看出,其实 CPU 从 Cache 读取数据与预取是并行的:预取处理为 CPU 提供大量待访问页面,而 CPU 只负责不间断的处理数据,简化了 CPU 的工作,其效率也因此得到极大提高.

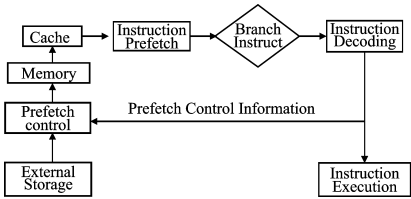


图 3 预取处理结构框图

3.3 状态机设计

状态机使用了 6 个状态来描述基于预取的 Cache 页面置换动态,分别是开始、预取、命中、访存、等待和终止. 这些状态的描述如表 1 所示,状态转换图如图 4 所示.

表 1 预取状态描述表

状态	状态描述
开始	用户程序被新建
预取	访问的数据不在 Cache 和主存中,需要从外存预取
命中	访问的数据在 Cache 中
访存	Cache 页面更新,从主存向 Cache 调入页面
等待	指令的缺失,等待接受指令
终止	用户程序运行完毕

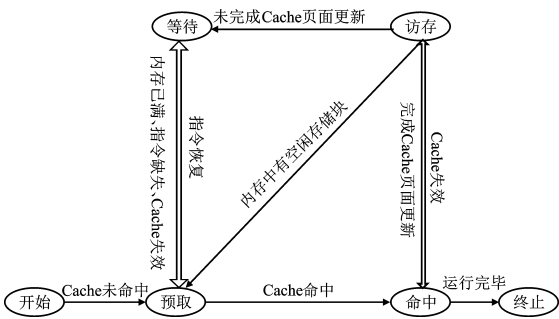


图 4 预取状态转换图

在程序启动时,由于新的数据需要从外存读取,故发生了从开始到预取的状态转换. 此时,从外存预取页面进入内存,以备 CPU 读取并调入 Cache;当 CPU 执行第一条指令时,首先在 Cache 中查找,若页面已经由内存“推送”至 Cache,则 Cache 命中,发生了从预取到命中的状态转换. 若页面尚未缓存进 Cache,则 CPU 进入等待状态并周期性询问页面是

否已在 Cache 中;当执行到某条指令,Cache 失效时,便发生了从命中到访存的状态转换.需要说明的是,由于从内存读取数据速度较慢,故访存态并不是 CPU 从内存读取页面,而是将页面调入 Cache,完成一次 Cache 页面更新,再从 Cache 读取数据,避免从内存直接读取.页面被调入 Cache 后,CPU 再访问页面,故发生了从访存到命中的状态转换.但此时若内存中无预取页面,无法完成 Cache 页面更新,故发生了从访存到等待的状态转换;如果主存中有空闲的存储块,则继续预取更多的页面,于是便有了从访存到预取的状态转换;如果主存中没有空闲物理存储块或指令缺失(指令缺失可能是由于条件分支指令未完成判定),就发生了预取到等待的状态转换;等待接受下一个指令,流水线恢复正常,发生从等待到预取的状态转换^[8].如此循环,直到程序运行完毕.

4 实验与结果分析

本实验使用由 Wisconsin-Madison 发布的一个用于模拟处理器程序的系统软件框架-SimpleScalar,它提供了五种功能的模拟器^[9].这是一个开源的软件,具有很强的可移植性和可扩展性.实验中基本配置如下:I-Cache 的容量为 32 kB,2 路组相联结构,行大小为 32 字节;D-Cache 的容量为 16 kB,2 路组相联结构,行大小为 16 字节.选择 SPEC2000 中的程序作为测试程序.为了考察基于预取的 Cache 替换策略(记为“P”)性能,选取了上文提到的 Markov 预取(记为“MP”)和无预取处理(记为“NP”)同时进行模拟实验,主要从一下几个方面进行评估:(1)Cache 命中率,考察经过预取处理后 Cache 的命中率是否得到有效提高;(2)无效预取率,考察预取处理的准确性;(3)测试程序的 SPEC Ratio 值,考察预取处理能否提升系统处理速度,因为如果一个方法尽管提高了 Cache 命中率,却牺牲了时间,增加了系统等待,这样的方法无疑是不能接受的.

从三种策略在 Cache 的命中率可以看出,使用预取策略可以有效提高 Cache 命中率,Markov 预取相对于没有使用预取策略命中率提高约 12.5%,而基于预取的 Cache 替换策略相对于 Markov 预取命中率提高约 5.5%.

从实验结果可以看出,在无效预取率方面,两者并无太大差距,Markov 预取的无效预取率稍高 2%.但在 SPEC Ratio 方面,可以看到在 6 个测试程

序中只有 Vpr、Vortex、Jpeg、Gzip 这 4 个程序的处理速度快于 MP 和 NP,而 mcf、twolf 的处理速度却慢于 MP.事实上,mcf 是由 C 语言编写的组合优化/单点轮换调度程序,teolf 是由 C 语言编写的定位和路由模拟程序,两者都是基于属于数字运算类.由于数字运算类程序与 Cache、内存的交互较少,这就会导致在运行这两个程序时,可能使系统进行了多个周期的无效预取.在 Cache 命中率比较方面,与其他测试程序相比,在 Mcf 和 Twolf 两个测试程序中,三种策略的命中率并无太大差距.所以,从以上实验结果来看,改进后的预取策略性能有所提高.

5 结束语

本文提出了一种将预取应用到 Cache 页面替换的策略,在提高 Cache 命中率的同时提升了系统性能.但是,这种预取策略也有其自身的缺点,主要表现在以下两个方面:(1)额外增加的预取处理部分;(2)若预取处理不及时,不能为处理器提供待访问的页面时,同样会产生缺页中断.所以,关于 Cache 预取的问题仍需更深入研究与完善.

参考文献:

- [1] 谌卫军,王浩娟.操作系统[M].北京:清华大学出版社,2012.
- [2] 方娟,张红波.一种改进的多核处理器硬件预取技术[J].计算机科学,2012,39(增刊2):48-50.
- [3] 唐朔飞.计算机组成原理[M].北京:高等教育出版社,2008.
- [4] Barriga L, Ayani R. Lazy Update: An Efficient Implementation of LRU Stacks[J]. Information Processing Letters, 1995, 54(2):81-84.
- [5] Ding J, Yang B Y. Multivariate polynomials for hashing[J]. Lecture Notes in Computer Science, 2007, 4990:358-371.
- [6] 涂小琴,商伟,周帆帆.一种结合预调度方式的页面置换方法[J].计算机科学,2014,41(11A):410-411.
- [7] Bodik R, Gupta R, Soffa M. Interprocedural conditional branch elimination[C]//Proceedings of the SIGPLAN'97 Conference on Programming Language Design and Implementation. ACM,1997:146-158.
- [8] 李伟立,于立新.应用预取策略的行缓冲指令 Cache 设计[J].微电子学与计算机,2011,28(1):38-41.
- [9] 金立忠,窦勇.微处理器体系结构模拟器 SimpleScalar 分析与优化[J].计算机应用研究,2006, 23(8): 197-198.

了 0.3 m 的准确度和精度,其得到一个位置点消耗时间为 1 s,达到了实时性要求.

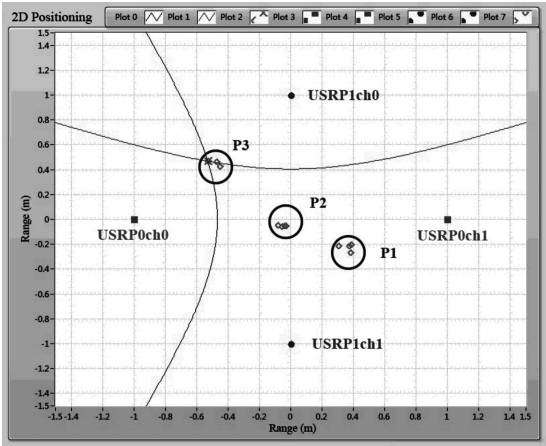


图 7 二维平面移动测试的结果

5 结束语

本文设计了一套基于 Wi-Fi 技术和 TDOA 定位原理的室内定位系统方案,完成了 Wi-Fi 协议的接收,最后在软件无线电平台 NI USRP 2952R 上搭建起了室内定位系统原型机,并实现了室内环境下二维平面移动定位,达到 0.3 m 的准确度和精度,以及每个位置 1 s 的刷新频率. 相比基于 RSSI 的方案^[8],准确度和精度得到大大提高,具有良好的应用前景.

参考文献:

[1] 陈丽娜. WLAN 位置指纹室内定位关键技术研究[D]. 上海:华东师范大学,2014.

[2] Want R. An introduction to RFID technology[J]. IEEE Pervasive Computing, 2006,5(1):25-33.

[3] Selmi A Vervisch-Picois , Samama N . Design of new codes for reducing interference between -Like signals

transmitted indoors[C]// Indoor Positioning and Indoor Navigation (IPIN), 2013 International Conference on. Montbeliard-Belfort, 2013:1-9.

[4] Soba J, Munir A, Suksmono A B . Barker code radar simulation for target range detection using software defined radio[C]// Information Technology and Electrical Engineering (ICITEE), 2013 International Conference on. Yogyakarta, 2013: 271-276.

[5] IEEE. IEEE Std 802. 11-2012 IEEE Standard for Information technology-Telecommunications and information exchange between systems Local and metropolitan area networks-Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification[S]. IEEE, 2012.

[6] Haring L , Bieder S , Czylik A ,et al. Estimation algorithms of multiple channels and carrier frequency offsets in application to multiuser OFDM systems[J]. IEEE Transactions on Wireless Communications, 2010,9(3):865-870.

[7] 龚国辉,李思昆. 提高 DSSS 信号 PN 码相位测量精度的三点二次插值法[J]. 通信学报, 2007, 28(2): 130-133.

[8] Youssef M, Agrawala A. The Horus WLAN location determination system[C]//Proceedings of the 3rd international conference on Mobile systems, applications, and services. ACM, 2005: 205-218.

作者简介:

李振宁 男,(1990-),硕士. 研究方向为基于 Wi-Fi 的室内定位技术. E-mail: lizn13@mails. tsinghua. edu. cn.

张 春 男,(1972-),副教授. 研究方向为射频识别、数字信号处理,数模混合信号集成电路设计.

李永明 男,(1945-),教授. 研究方向为数/模集成电路设计.

谢 拓 男,(1990-),博士. 研究方向为射频定位技术.

(上接第 89 页)

作者简介:

孙玉强 男,(1956-),博士,教授. 研究方向为信息科学、软件工程、并行算法.

王文闻(通讯作者) 男,(1991-),硕士研究生. 研究方向为软件工程、并行算法. E-mail:1552726162@qq. com.

巢碧霞 女,(1991-),硕士研究生. 研究方向为并行算法.

顾玉宛 女,(1982-),博士. 研究方向为并行算法.