内存与片上渗透缓存之间数据迁移的理论分析

胡九川[1]，范东睿[2]，程建聪[1]，严龙[2]，叶笑春[2]，李灵枝[1]，万良易[1]，钟海斌[1]
（1. 北京交通大学计算机科学与技术学院，北京 100044；2. 中国科学院计算技术研究所，北京 100080）

摘 要：为提高处理器内核的访存效率和访存命中率，缩短访存时延，可以将具有局部关联关系的指令和数据以群组的方式从内存迁移到处理器片上渗透缓存。指令和数据之间存在的局部性关联关系以及在指令和数据被迁往片上缓存的过程中发生的变化必须从理论的高度予以分析研究。研究结果表明，将指令和数据渗透迁移到片上渗透缓存可以确保及时局部性得到有效保持；仿真实验表明，在完善数据从内存迁往片上渗透的过程中渗透缓存提高了处理器内核的访存命中率。研究成果可为营造片上渗透缓存内及时局部性环境以提高处理器性能提供新的方法。

关键词：片上数据迁移；及时局部组；渗透
中图分类号：TP302.1
文献标识码：A
DOI: 10.11959/j.issn.1000−436x.2021157

Theoretical analysis for the data immigration between memory and processor percolation cache

HU Jiuchuan[1], FAN Dongrui[2], CHENG Jiancong[1], YAN Long[2], YE Xiaochun[2],
LI Lingzhi[1], WAN Liangyi[1], ZHONG Haibin[1]
1. School of Computer and Information Technology, Beijing Jiaotong University, Beijing 100044, China
2. Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100080, China

Abstract: To improve the computer processor's memory access efficiency, increase hit rates, reduce data fetching latency, the data and instructors that had close relation with each other could build the just-in-time locality environment on processor's cache. To build such an environment, the relations between the data and instructors should be studied while them being immigrated into on-chip cache. The research results show that these relations could be well kept when the data and instructors are moved. The simulation also shows that the percolation cache can help the processor core to increase its hit rates. These research results give a new way to build the just-in-time locality environment on the percolation cache.

Keywords: data immigration on chip, just-in-time locality, percolation

1 引言

计算机存储程序原理决定了即将被执行的指令或被处理的数据必须分批次地调入处理器，在容量十分有限的处理器片上缓存内中转停留，而暂时不能调入的大量指令和数据停留在内存乃至硬盘中，等候调遣进入处理器参与运算实现程序运行；同样，被执行过的指令或处理过的数据也很可能在处理器片上缓存内中转停留，然后回到内存或硬盘。所以，将那些在时间和空间上具有紧密关联关系的指令与数据以群组为单位预先放置于紧邻处理器核的片上缓存的顶端，形成及时局部性环境[1-5]。这样的环境有利于缩短处理器内核的访存时延，提高内核的访存命中率。但是，对指令和数据在片上

收稿日期：2021−01−08；**修回日期**：2021−03−26
基金项目：国家自然科学基金资助项目（No.61732018）
Foundation Item: The National Natural Science Foundation of China (No.61732018)

缓存与内存之间往返迁移的特点和规律缺乏应有的研究和分析。本文将运用数学方法研究指令和数据在处理器片上的迁移，初步探索其特点和规律，以最大限度保持指令和数据之间的时间和空间联系，该方向的研究具有重要的理论和实践指导意义。

根据存储程序原理，指令和数据之间的时空关联关系来源于程序执行逻辑的内在规定。抽象地看，以指令执行顺序、数据安置顺序为基本构件形成的逻辑复合体构成了计算机程序，这个逻辑复合体隐含着指令之间、数据之间不可割舍的时空联系。在迁移进出处理器时，指令和数据之间的时空联系很可能由于它们在内存和片上缓存的存储位置变化而发生变化，导致在处理器原计划遵照程序内在的逻辑展开运算时，前来向处理器“报到”的指令或数据却打乱了这内在的程序执行逻辑。这样需求错位的矛盾局面产生的根源是内存容量和片上缓存容量存在巨大的落差，片上缓存中的指令和数据需要不断地更新才能将所有指令和数据从内存中推到处理器核的面前，方便处理器核访问。因此，需求错位的矛盾是结构性的。

为了缓解需求错位的矛盾，在处理器内核访问任意指令或数据时，应该将与之存在时空关系的其他指令、数据组合为一个整体迁移到处理器的片上缓存内，尽量保持指令或数据之间的时空联系，将指令之间、数据之间的时空关系转化为指令、数据在片上缓存体系中紧密分布的形态。由于处理器片上缓存是指令和数据进入处理器核的必经之路，此分布形态必然为提高处理器内核的访存命中率，减缓访存时延、控制指令和数据迁移以适应处理器运算需要，营造有利的片上及时局部性环境创造条件。

事实上，及时局部性的特质已经体现在处理器体系的内在结构之中。例如，存储在片上寄存器里的指令操作元分布在诸如指令解码器、指令流水线等功能单元可就近方便获取的地方[6-7]，各级流水线中众多状态寄存器的存在都是及时局部性存在的例证[8-9]。本文可以把片上通用寄存器理解为片上缓存，而片上缓存是通用寄存功能的具体延展。

本质上，片上及时局部性环境可以使处理器所需要的指令或数据及时、就近地在片上寄存器或缓存内找到[10-13]。因此，掌握指令和数据在片上寄存器、缓存和内存之间往返迁移的规律和特点，必将为人们探索在迁移过程中保持指令或数据之间的时空联系、营造高质量的及时局部性环境奠定基础。

以指令解码器为分水岭，可以把营造及时局部性环境的工作分为解码前期和解码后期2个阶段。在解码前期，及时局部性环境主要在片上缓存营造，指令和数据在缓存中的分布形态及其更新是人们关注的重点；在解码后期，及时局部性环境主要通过控制指令和数据在处理器片内各类寄存器中的分布来实现。

为了更好地缓解需求错位的矛盾，文献[1]给出了一个由2个传统缓存耦合而成的新型片上缓存，称之为渗透缓存，并在指令或数据的迁移中将具有时空关联关系的指令和数据以及时局部组的整体形式从内存迁往渗透缓存，同时控制指令和数据在渗透缓存内一方面似泉水涌出般地“涌”向渗透缓存的顶端，另一方面似泉水被吸入泉眼般地被“吸”回内存，使指令和数据在片上缓存内流动起来。

图1给出了一个渗透缓存结构及数据迁入规则示意，其中地址4是处理器核的访存焦点，相邻地址3与地址5、地址2与地址6、地址1与地址7构成内存中一个对称的及时局部组。当地址4上的指令或数据被迁往渗透缓存中的泉吸缓存的顶端时，及时局部组内的其他指令或数据被分别迁往渗透缓存中的第一级、第二级、第三级泉涌缓存。由于地址4的及时局部性程度最高，与之邻近的地址3和地址5的及时局部性程度也比较高，因此地址3和地址5上的指令或数据被迁移到泉涌缓存的第一级。如果处理器内核在随后的访存中其访存焦点仍然是地址4，那么该地址的及时局部性仍然保持最高；如果访存焦点不是地址4而是地址5，那么它的及时局部性程度变为最高，地址4的及时局部性降低。此刻，地址4中指令或数据可以停留在原地或被降级迁往渗透缓存中的第二级泉吸缓存，地址5中的指令或数据被升级迁往第一级泉吸缓存，地址6中的指令或数据被升级迁往第一级泉涌缓存，地址7中的指令或数据被升级迁往第二级泉涌缓存，如此，指令或数据在渗透缓存中变成流动的趋势。仿真实验表明[4]，将处理器片上缓存改进为渗透缓存提高了处理器内核访存

的效率和访存命中率，为缩短访存时延营造了及时局部性环境[2-3]。

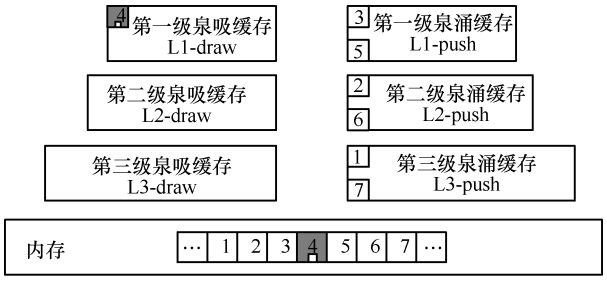


图 1　渗透缓存结构及数据迁入规则示意

显然，如果及时局部组内的指令或数据都能够满足处理器内核的访存需要，那将是一个理想状态，而渗透缓存是一种逼近这种理想状态的手段。本文将探讨这种理想状态产生的合理性。

及时局部组是一个以处理器访存焦点即当前被访问的指令或数据为中心、一定范围内的指令或数据组成的、具有时空关联关系的指令或数据族群。所以，应该以族群内的指令或数据之间的关联关系在迁移过程中发生的变化为入手点，去认识及时局部环境内在机制，发现理想状态产生的前提条件。为了从理论上深入认识及时局部环境的内在机制，实现理想的及时局部性状态，本文将运用抽象的数学方法研究分析指令或数据在处理器片上缓存和内存之间往返迁移的规律，寻找迁移过程中保持指令或数据及时局部性的前提条件。文献[1]具体介绍了渗透缓存中的数据迁移方式的有效性，本文将以数学抽象的方式证明渗透缓存结构的内在合理性。

2　内存、片上缓存的抽象描述

2.1　内存、地址和基本可寻址单位

地址是计算机体系结构中的核心概念，是建立存储体系的基本工具。地址的组成结构规定了计算机内存和片上缓存的关联法则[4,14]。下面，给出内存、地址和基本可寻址单位的抽象定义。由于存储空间是描述地址结构及其形式的基础，因此先给出存储空间的抽象定义。设 m 为自然数，集合 $D=\{0,1\}$，称集合 $D^m=\left\{(d_1,d_2,\cdots,d_m)\,|\,d_i\in D,i=1,2,3,\cdots,m\right\}$ 为存储空间，自然数 m 为存储空间维数。存储空间维数 m 决定存储空间的容量。由于 m 维存储空间中的向量是 m 位的二进制数，存储空间中的所有向量构成一个由 m 位二进制数组成、按数值大小关系排列顺序的全序集合[15]，这种序关系本质上就是地址的序关系。因此从存储空间演化出地址空间。

定义 1　设 m 为自然数，D^m 为 m 维的存储空间，$\leqslant$ 为存储空间中的全序关系，称集合 $\{D^m,\leqslant\}$ 为地址空间，地址空间中的向量为地址。

定义 2　设 m,n,p,q,k 为自然数，$m=kn$，D^n 为地址空间，D^m 为存储空间，设

$$D^n\times D^m=\left\{(a_1,a_2,\cdots,a_n,d_1,d_2,\cdots,d_m)\right\}$$

其中

$$(a_1,a_2,\cdots,a_n)\in D^n,(d_1,d_2,\cdots,d_m)\in D^m$$

则称 $D^n\times D^m$ 为内存；D^m 中的向量为存储槽，内存空间中的向量为基本可寻址单位。

根据定义 2，基本可寻址单位由地址和存储槽构成，具有唯一的地址，泛指存储在内存中的指令或数据。2 个基本可寻址单元在内存中的距离可以定义为地址之间的距离。

2.2　片上缓存

在处理器运行过程中，被访问过或从内存取出等待访问的数据被置于一个具有层次结构的片上缓存内。片上缓存实质是内存层次化的自然延伸。片上缓存的结构及其上的数据分布策略影响处理器的性能。处理器内核应该尽可能在缓存层级的高层级内访问到原本需要到内存或低缓存层级中才能访问到的数据。所以，必须将加载到内存的数据迁移到容量相对有限的片上缓存高层级内。将一个大容量结构内的数据纳入一个容量相对小的结构内的根本方法是将大容量结构内的数据分成等价类，将等价类的代表放入容量相对小的结构之中。只有这样才能在逻辑意义上将一个大容量结构置于小容量结构之中。目前，片上缓存与内存之间的关联结构主要是通过等价分类实现的[16]。定义 3 描述了该等价类方法的本质。

定义 3　设 m,n,p,q 为自然数，$D^p\times D^q$ 和 $D^n\times D^m$ 为内存。如果映射

$$Q:D^n\times D^m\to D^p\times D^q$$

使对内存 $D^n\times D^m$ 中任意的基本可寻址单位

$$u=(a_1,a_2,\cdots,a_n,d_1,d_2,\cdots,d_m)$$

存在内存中的基本可寻址单位 $v=(i_1,i_2,\cdots,i_p,c_1,c_2,\cdots,c_q)$ 的地址满足

$$a_1 2^{n-1} + a_2 2^{n-2} + \ldots + a_n = g2^p + I$$

其中，g 为自然数，$2^p > I$，$I = i_1 2^{p-1} + i_2 2^{p-2} + \cdots + i_p$。则称内存 $D^p \times D^q$ 为内存 $D^n \times D^m$ 的常规缓存，I 为基本可寻址单位 u 在缓存 $D^p \times D^q$ 的索引号，映射 Q 为从 $D^n \times D^m$ 到 $D^p \times D^q$ 为常规相联。

图 2 是一个内存和缓存关联关系示意。内存里的数据地址按照定义 3 中规定的模运算进行换算，并以模运算的余数为索引号为该地址上的数据在缓存中确定新位置，即存储槽索引号。例如，图 2 中缓存容量值 c 为 7，以其为模数，对数值为 8 的地址进行模运算得到余数为 1，则位于内存位置 8 上的数据被迁移到缓存中后，其在缓存中的索引号为 1。图 2 中的平面坐标系内给出了从内存地址空间 U 到片上缓存地址空间 H（缓存槽索引号）之间的对应关系。

一般地，内存空间 $D^n \times D^m$ 中的基本可寻址单位 u 的地址与另一个内存空间 $D^p \times D^q$ 的容量 2^p 进行模运算，所得余数正好为基本可寻址单位 v 在内存空间 $D^p \times D^q$ 里的地址，即缓存槽索引号。根据拓扑学的知识[15]，模运算将内存空间中的所有基本可寻址单位分成 2^p 个等价类。

定义 3 说明了缓存是层次结构意义上的内存，只是离处理器核更近。根据定义 3，直接相联缓存和集合相联缓存是常规缓存。在大多数情况下，常规缓存用来建立一般情况下的片上缓存。但是，全相联缓存为非常规缓存，这是因为 $D^n \times D^m$ 内的可寻址单位可以在 $D^p \times D^q$ 内任意放置，不用模运算来定位。非常规缓存用于在特殊的条件下建立片上缓存。除非特别指出，本文的讨论围绕常规缓存展开。

2.3 访存焦点与及时局部性

处理器当前访存所指向的内存位置称为处理器的访存焦点。

定义 4 设 m，n 为自然数，r 为非负整数，$u = (a_1, a_2, \cdots, a_n, d_1, d_2, \cdots, d_m)$ 为内存 $D^n \times D^m$ 中任意一个基本寻址单位，$u' = (a'_1, a'_2, \cdots, a'_n, d'_1, d'_2, \cdots, d'_m) \in D^n \times D^m$，称集合 $N(u,r) = \{u' : e(u-u') < r\}$ 为以基本寻址单位 u 为中心的邻域。其中，$e(u-u')$ 为 2 个基本可寻址单位 u 和 u' 之间的距离。

以一个处理器的当前访存焦点 u 为中心，以距离 r 为半径，可在内存中划定一个不超出此半径的对称区域，区域内汇集了一簇按照地址顺序排列的基本可寻址单位，访存焦点位于队列正中央，形成一个以访存焦点为中心的邻域 $N(u,r)$，构成一个对称及时局部组。控制邻域半径，即可

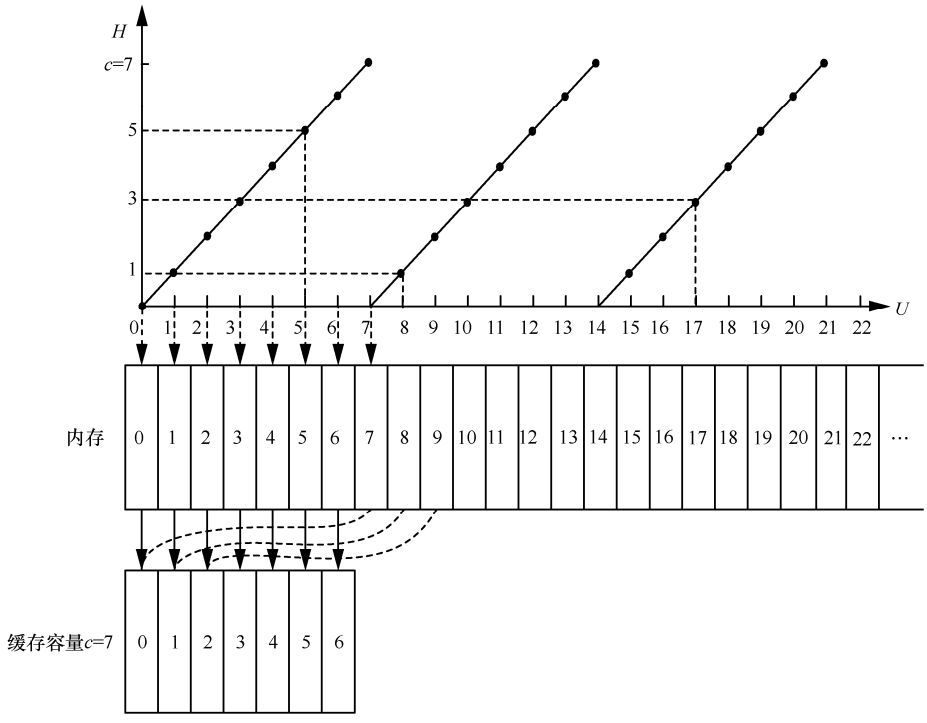


图 2 内存和缓存关联关系示意

控制邻域规模；控制邻域规模，即可控制迁移数据的规模。

在处理器运行中，访存焦点 u 可能被再次访问，多次成为处理器的执行焦点，这种执行焦点短暂保持在固定位置的处理器运行态势称为时间局部性。当访存焦点发生改变时，处理器要访问邻域 $N(u,r)$ 内其他的基本寻址单位，且执行焦点不超出邻域范围的微漂移执行态势。执行焦点在固定位置重复出现，或相对于固定位置产生微漂移的执行态势说明了及时局部组内数据之间具有相对突出的汇聚性，是及时局部性的动态形式。

因此，以基本寻址单位的邻域为媒介，可为处理器的运行营造快捷访问数据的及时局部性环境，进而在片上缓存内完善迁移数据的机制，将改进计算性能的视野从静止固定的缓存结构层面，提升到动态灵活利用数据关联特征的高度。

3　及时局部性的分拆与折叠

为了在片上缓存的高端层级营造理想的服务处理器核的及时局部性环境，作为整体迁移的及时局部组邻域 $N(u,r)$ 内各基本寻址单位之间相互靠拢的位置关系应该尽力保持稳定。然而，及时局部组内基本可寻址单位相互靠拢的态势可能会因内存和片上缓存的关联结构限制而在从内存迁移到缓存过程中发生形变，造成及时局部组邻域被分拆或折叠。

例 1　图 3 显示了一个常规缓存中缓存槽索引号和内存地址的对应关系。例如，索引号为 2 的缓存槽可以接纳来自内存地址为 2、10、18、26 等的数据。以任意一个内存地址为中心，取邻域半径为 2，可以构成一个以该地址为中心的及时局部组。从图 3 可以看出，那些以与缓存存储槽索引号为 2、3、4、5 对应的地址为中心的及时局部组可以不破坏地址相邻连续性关系地迁移到缓存内。例如，以地址 19 为中心的及时局部组占据一个连续的地址段 17、18、19、20、21。这些内存地址上数据之间的关联关系可以不被分拆地迁移到缓存内一组相邻索引号 1、2、3、4、5 所指定的缓存槽内。然而，以那些与索引号 0、1、6、7 相对应的地址为中心的及时局部组却只能被分拆后迁移到缓存内。例如，以地址 23 为中心的及时局部组占据一个连续地址段 21、22、23、24、25，该地址段上的数据被迁移到缓存内一组不相邻的索引号 5、6、7、0、1 所指定的片上缓存槽内。索引号为 5 的缓存槽与索引号为 1 的缓存槽之间的距离为 3，超过了及时局部组的邻域半径 2。

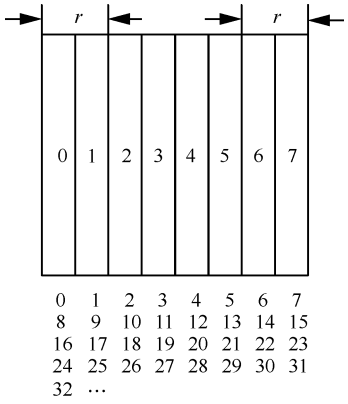


图 3　片上缓存中数据分配示意

及时局部组迁移后其邻域半径被拉长的情形称为及时局部性分拆。反过来，如果及时局部组半径过大，会出现一个缓存槽被来自内存中地址不同的 2 个数据同时占据的不合理情形。

例 2　在图 3 中，选择及时局部组邻域半径为 6，以地址 14 为中心的及时局部组占据一个连续地址段 8、…、13、14、15、…、20。这些地址上的数据被迁移到缓存内后，一些缓存槽同时被 2 个来自不同内存地址的数据所占据。例如，索引号为 3 的缓存槽被来自内存地址 11、19 的不同数据同时占据，这显然是不合理的。

同一缓存槽被来自不同地址的数据同时占据的情形称为及时局部性折叠。及时局部性折叠是数据过度靠拢的畸形形态，在基本可寻址单位从内存迁往缓存的过程中，应杜绝及时局部性折叠。定理 1 给出了消除及时局部性折叠的充分必要条件。

定理 1　设 m,n,p,q 为自然数，内存 $D^p \times D^q$ 为内存 $D^n \times D^m$ 的常规缓存，r 为对称及时局部组的半径，则发生及时局部性折叠的充分必要条件是 $r \geqslant c/2$。其中，$c = 2^p$ 为缓存 $D^p \times D^q$ 的容量。

证明　必要性。任取内存 $D^n \times D^m$ 里一个基本可寻址单位，设其在常规缓存 $D^p \times D^q$ 内的索引号为 h。如果出现及时局部组折叠，则只能出现图 4 和图 5 这 2 种情形中。在图 4 中，索引号为 h 的缓存槽到索引号为 c 的缓存槽的距离 $c-h$ 小于及时局部组的邻域半径 r，且

$$r-(c-h) \geqslant h-r$$

于是 $r-c+h \geqslant h-r$，则 $2r \geqslant c$，即

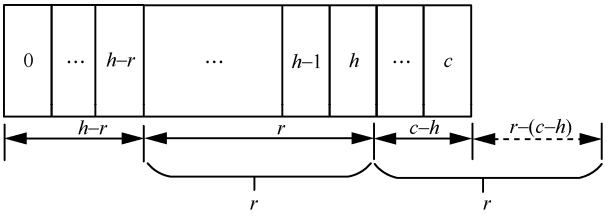


图 4　靠近缓存高地址端

$$r \geqslant c/2$$

同理，在图 5 中，索引号为 h 的缓存槽到索引号为 0 的缓存槽的距离 h 小于及时局部组的邻域半径 r 且

$$r-h \geqslant c-(h+r)$$

于是 $r-h \geqslant c-h-r$，则 $2r \geqslant c$，即

$$r \geqslant c/2$$

所以，如果发生及时局部性折叠，则必然有 $r \geqslant c/2$。

充分性。假设不发生及时局部性折叠，则 $r-(c-h)<h-r$ 而且 $r-h<c-(h+r)$，于是根据上述 2 个不等式，有

$$r<c/2$$

所以，如果 $r \geqslant c/2$，则一定发生及时局部性折叠。

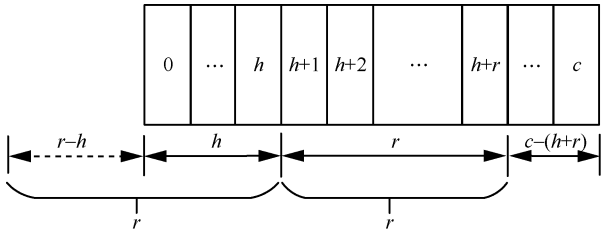


图 5　靠近缓存低地址端

定理 2　设 m, n, p, q 为自然数，内存 $D^p \times D^q$ 为内存 $D^n \times D^m$ 的常规缓存，r 为对称及时局部组的半径，则不发生及时局部性折叠的充要条件是

$$r<c/2$$

其中，$c=2^p$ 为缓存 $D^p \times D^q$ 的容量。

证明　根据定理 1，此结果是显然的。

定理 1 和定理 2 指出控制及时局部组的邻域半径在一定范围内能杜绝及时局部性发生折叠。然而，及时局部性发生撕裂却是一定的。只要控制内存地址的取值范围和及时局部组邻域半径规模，直接相连和组相连的内存与片上缓存的连接关系能够使及时局部组在从内存迁往片上缓存的过程中，及时局部组内各个数据之间的相对关系不被破坏，这意味着当前处理器中片上缓存和内存之间的相联机制能够实现保持数据及时局部性的迁移。

4　渗透缓存内命中率分析

数据迁移到片上缓存的中心地带可能会产生及时局部性分拆（由于通常的及时局部组邻域半径远小于片上缓存容量的一半，本文在此假设及时局部性折叠的情况可以忽略）。例如，考虑数据从图 6 中的内存 C_s 迁移到缓存 C_t 的情形。如果及时局部组邻域某个基本可寻址单位的地址值为缓存 C_t 容量值的整数倍，那么如图 6 中的阴影区域所示，这些阴影区域内的数据迁入缓存后将被分拆为两部分，阴影左半部分内的数据被安置在缓存 C_t 的右端，阴影右半部分的数据被安置在缓存 C_t 的左端。否则，及时局部组邻域内的数据迁入缓存 C_t 内后不发生分拆，直接安置在缓存 C_t 内连续的存储槽号所指定的缓存槽里。

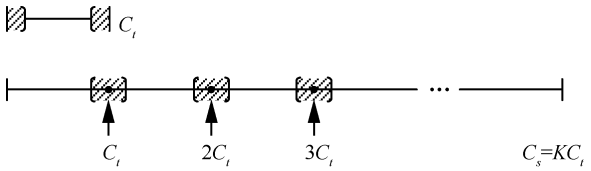


图 6　数据从内存 C_s 迁移到缓存 C_t

如果在迁移中保持局部性不被分拆，那么数据可以在片上缓存内接收处理器内核的访问，免除了处理器内核“远涉”内存去获得数据；但是，当处理器访存焦点移出该及时局部组时，后续到来的数据及时局部组很可能全面覆盖之前到来的及时局部组，此刻，若处理器内核的访存焦点移回之前的及时局部组，那么处理器内核将不能在片上缓存内访问到之前及时局部组内的数据，只能“远涉”内存再次迁移数据。因此，处理器内核访问片上缓存的命中率会在这种条件下降低，访存时延会相应增加。

如果迁移过程中发生及时局部性分拆，那么及时局部组内的数据就会分拆开来，分布在片上缓存的两端。此时，该及时局部组被后续到来的及时局部组全面覆盖的可能性存在降低的可能。如果及时局部组在迁移到片上缓存的过程中，及时局部组内的数据分布在如图 1 所示的三级缓存内，那么，这

些局部组内的数据被随后到来的新的局部组全部覆盖的可能性就更低了。这就是本文在第 1 节中介绍的渗透缓存能够提高处理器访存命中率、缩短访存时延的原因。下面的仿真实验证实了本文的理论分析。

5　仿真实验

本文利用 modelsim 10.1a 仿真平台对向传统缓存和渗透缓存实施数据迁移的过程进行了模拟，核心工作是模拟处理器 load/store 指令的数据传输过程。为此，本文构建了处理器模块、缓存模块、内存模块、渗透次数收集模块。其中处理器模块用来解析测试集地址以及产生访存请求；缓存模块用来处理数据的替换、渗透以及向渗透次数收集模块反馈单次渗透完成信号；内存模块用来处理数据的替换、渗透迁移以及存储数据；渗透次数收集模块用来统计单轮渗透次数信号，当收集到一轮完整的渗透信号之后，向处理器汇报，使处理器开始下一轮工作，各模块通过数据传输协调工作，如图 7 所示。实验中传统缓存和渗透缓存的配置规格如表 1 所示。

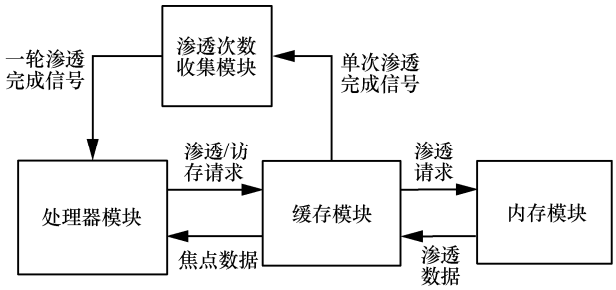


图 7　仿真模块协调工作示意

表 1　　传统缓存和渗透缓存参数设置

缓存形式	缓存	容量大小/KB	块大小/B	组数	每组块数
传统缓存	L1	32	32	8	128
	L2	64	32	16	128
	L3	128	32	32	128
渗透缓存	L1-draw	16	32	4	128
	L1-push	16	32	4	128
	L2-draw	32	32	8	128
	L2-push	32	32	8	128
	L3-draw	64	32	16	128
	L3-push	64	32	16	128

实验结果表明，在内存和片上缓存之间实施渗透数据迁移之后，相比传统的数据迁移做法，处理器内核访问片上缓存的命中率总体上得到了有效提高，自然相应地缩短了处理器的访存时延。由于篇幅的限制，本文在此给出部分实验数据，如表 2~表 4 所示。其中，程序 Wordcount、Cholesky 以及 LU 是通常测试处理器性能的测试程序，具有典型的代表性，Wordcount 的指令和数据的地址存放序列具有突出的时间局部性，LU 具有突出的空间局部性，而 Cholesky 兼具时间与空间局部性。以 Wordcount 测试集作为处理器访问内存的请求地址，将访存命中率从传统缓存的 88.036%提升至渗透缓存的 92.336%；以 Cholesky 测试集作为处理器访存的请求地址，将访存命中率从传统缓存的 90.118%提升至 99.920%；以 LU 测试集作为处理器访存的请求地址，将访存命中率从传统缓存的 0.076%提升至 92.160%。由于渗透缓存由泉吸缓存与泉涌缓存构成，而传统缓存在结构上相当于泉涌

表 2　　传统缓存和渗透缓存上的命中率（Wordcount）

缓存形式	渗透缓存层级	访问次数	命中次数	命中率	总命中率
渗透缓存	L1-draw	1 000 000	645 076	64.508%	92.336%
	L1-push	35 4924	46 151	13.003%	
	L2-draw	308 773	136 667	44.261%	
	L2-push	172 106	9 654	5.609%	
	L3-draw	162 452	82 403	50.725%	
	L3-push	80 049	3 405	4.254%	
传统缓存	L1	1 000 000	722 311	72.231%	88.036%
	L2	277 689	96 518	34.757%	
	L3	181 171	61 534	33.965%	

表 3　　**传统缓存和渗透缓存上的命中率（LU）**

缓存形式	渗透缓存层级	访问次数	命中次数	命中率	总命中率
渗透缓存	L1-draw	877 441	3 200	0.365%	92.160%
	L1-push	874 241	110	0.013%	
	L2-draw	874 131	8 091	0.926%	
	L2-push	866 040	769 933	88.903%	
	L3-draw	96 107	27 309	28.415%	
	L3-push	68 798	3	0.004%	
传统缓存	L1	877 441	7 105	0.008%	0.076%
	L2	870 336	15 500	0.017%	
	L3	854 836	44 442	0.052%	

表 4　　**传统缓存和渗透缓存上的命中率（Cholesky）**

缓存形式	渗透缓存层级	访问次数	命中次数	命中率	总命中率
渗透缓存	L1-draw	1 000 000	898 061	89.806%	99.920%
	L1-push	101 939	97 863	96.002%	
	L2-draw	4 076	2 330	57.164%	
	L2-push	1 746	251	14.376%	
	L3-draw	1 495	573	38.328%	
	L3-push	922	118	12.798%	
传统缓存	L1	1 000 000	899 589	89.9589%	90.118%
	L2	100 411	1 356	0.013%	
	L3	99 055	237	0.002%	

缓存容量为零的渗透缓存，且渗透缓存比传统缓存更加注重空间局部性，对于 LU 这种具有极端的空间局部访存特性的测试集，渗透缓存对访存命中率的提升幅度更大。从实验结果分析，在 LU 测试集中，大部分处理器内核所需要的数据在被迁移渗透访问的过程中驻留在渗透缓存里，这可以从表 4 绝大部分成功访问发生在缓存 L2-push 中的结果得到验证。从整体上看，访存总的命中率得到提升，在渗透缓存各层级的命中率较传统缓存中对应层级的命中率也得到提升。

在片上缓存容量固定不变的条件下，被迁移的缓存块大小变化与处理器内核访问缓存的命中率之间的关系比较复杂。确定缓存块的规模需要以形成片上及时局部性环境为目标，在动态调控缓存块的规模和迁移过程中尽量使包含在程序中的计算逻辑或业务逻辑在片上及时局部性环境得到保持。为此，本文下一步将进行动态调控缓存块大小规模方面的探索研究。

6　结束语

从本文的讨论中可以发现，数据迁移到处理器片上缓存中之后，自然不存在处理器内核到内存里访问数据的时间消耗，缩减了处理器内核的访存时延；如果处理器内核在片上缓存内的访存命中率得到提高，那么缩减访存时延的积累效应就更加理想。

所以，构成及时局部组的数据整体迁移到处理器片上后，处理器访问内存数据的时间消耗将进一步减少；如果以及时局部组的形式整体到片上缓存内的数据能够化整为零地分布在缓存内的各个层级，那么这些数据就可能在片上缓存内停留更长的时间而不被后续到来的及时局部组所覆盖，进而为处理器内核的访存焦点再次转移到这些化整为零的数据上创造了更好的命中率，如此迭代积累，必然在片上缓存内建立起理想的及时局部性环境，为改进处理器计算性能挖掘出新的潜力。本文给出的理论分析证明了以及时局部组形态迁移数据从内

存迁往片上缓存不仅是可行的，而且营造处理器片上缓存内及时局部性环境是合理的。在内存和片上缓存之间的数据迁移方式蕴含了在微观层面实施数据通信的潜在基础。

参考文献：

[1] 胡九川, 范东睿, 李丹萍, 等. 一种支持数据渗透迁移的片上缓存模型研究[J]. 北京交通大学学报, 2017, 41(5): 1-9.
HU J C, FAN D R, LI D P, et al. An on-chip cache model research on supporting data permeation and migration[J]. Journal of Beijing Jiaotong University, 2017, 41(5): 1-9.

[2] TAN G M, SUN N H, GAO G R. Improving performance of dynamic programming via parallelism and locality on multicore architectures[J]. IEEE Transactions on Parallel and Distributed Systems, 2009, 20(2): 261-274.

[3] GARCIA E, OROZCO D, KHAN R, et al. A dynamic schema to increase performance in many-core architectures through percolation operations[C]//20th Annual International Conference on High Performance Computing. Piscataway: IEEE Press, 2013: 276-285.

[4] 李丹萍. 单核处理器片上渗透数据调配方法研究[D]. 北京: 北京交通大学, 2016.
LI D P. Research on allocation method of penetration data on single-core processor[D]. Beijing: Beijing Jiaotong University, 2016.

[5] FAN D R, YUAN N, ZHANG J C, et al. Godson-T: an efficient many-core architecture for parallel program executions[J]. Journal of Computer Science and Technology, 2009, 24(6): 1061-1073.

[6] ZHENG T H, ZHU H S, EREZ M. SIPT: speculatively indexed, physically tagged caches[C]//2018 IEEE International Symposium on High Performance Computer Architecture. Piscataway: IEEE Press, 2018: 118-130.

[7] CRUZ E H M, DIENER M, ALVES M A Z, et al. LAPT: a locality-aware page table for thread and data mapping[J]. Parallel Computing, 2016, 54: 59-71.

[8] BHATTI M K, OZ I, AMIN S, et al. Locality-aware task scheduling for homogeneous parallel computing systems[J]. Computing, 2018, 100(6): 557-595.

[9] 王子聪, 陈小文, 郭阳. 片上多核处理器 Cache 访问均衡性研究[J]. 计算机学报, 2019, 42(11): 2403-2416.
WANG Z C, CHEN X W, GUO Y. Research on cache access equalization in chip multi-processor[J]. Chinese Journal of Computers, 2019, 42(11): 2403-2416.

[10] GAUR J, CHAUDHURI M, RAMACHANDRAN P, et al. Near-optimal access partitioning for memory hierarchies with multiple heterogeneous bandwidth sources[C]//2017 IEEE International Symposium on High Performance Computer Architecture. Piscataway: IEEE Press, 2017: 13-24.

[11] EL-SAYED N, MUKKARA A, TSAI P A, et al. KPart: a hybrid cache partitioning-sharing technique for commodity multicores[C]//2018 IEEE International Symposium on High Performance Computer Architecture. Piscataway: IEEE Press, 2018: 104-117.

[12] NORI A V, GAUR J, RAI S, et al. Criticality aware tiered cache hierarchy: a fundamental relook at multi-level cache hierarchies[C]//2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture. Piscataway: IEEE Press, 2018: 96-109.

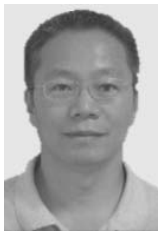
[13] YI L, SHAN G B, LIU S, et al. High-performance processor design based on 3D on-chip cache[J]. Microprocessors and Microsystems, 2016, 47: 486-490.

[14] 胡伟武, 陈云霁, 肖俊华. 计算机体系结构[M]. 北京: 清华大学出版社, 2011.
HU W W, CHEN Y J, XIAO J H. Computer architecture[M]. Beijing: Tsinghua University Press, 2011.

[15] 熊金城. 点集拓扑讲义[M]. 第 4 版, 北京: 高等教育出版社, 1993.
XIONG J C. Point set topology lecture[M]. 4th edition, Beijing: Higher Education Press, 1993.

[16] PATTERSON A, HENNESSY L. Computer organization and design: the hordware/software interface[M]. 5th edition, Morgan Kaufmann: Morgan Kaufmann Publisher, 2014.

[作者简介]



胡九川（1965− ），男，重庆人，博士，北京交通大学副教授、硕士生导师，主要研究方向为计算机体系结构、软件工程等。

范东睿（1979− ），男，黑龙江鹤岗人，博士，中国科学院计算技术研究所研究员、博士生导师，主要研究方向为众核处理器设计、高通量处理器设计、数据流处理器设计等。

程建聪（1997− ），男，山西运城人，北京交通大学硕士生，主要研究方向为计算机体系结构。

严龙（1988− ），男，北京人，中国科学院计算技术研究所工程师，主要研究方向为高通量计算机体系结构。

叶笑春（1981− ），男，江西万载人，博士，中国科学院计算技术研究所副研究员、硕士生导师，主要研究方向为众核处理器体系结构、高性能计算、高通量计算、软件模拟技术等。

李灵枝（1995− ），女，山西吕梁人，北京交通大学硕士生，主要研究方向为计算机体系结构。

万良易（1995− ），男，江西九江人，北京交通大学硕士生，主要研究方向为计算机体系结构。

钟海斌（1996− ），男，安徽黄山人，北京交通大学硕士生，主要研究方向为计算机体系结构。

《通信学报》第十届编辑委员会