计算机科学2002Vol. 29№. 7

LS MPP 图像数据分布技术的研究*)

Research on Image Data Distribution Technology of LS MPP

王 晖[1] 王 忠[2] 陈丹[1] 何华灿[1]

(西北工业大学计算机系 西安710072)[1] (西安微电子技术研究所 西安710054)[2]

Abstract This paper analyzes research of current image data distribution technology and presents a concept of ordered cover and ordered partition, by which image data access can be described as similar ordered cover access. Methods of data partition based on different data space are given. Based on analysis of access of data block, methods of continued access of covered block are proposed.

Keywords Parallel image processing, Data distribution, Cover, Partition

1 引言

在图像处理流程中,存在大量相互独立的重复计算,例如点操作和邻域操作中,像素之间的操作是互不相关的,这就使并行图像处理成为可能。很久以来,研究人员考虑设计并行图像处理算法在并行计算机上运行,以提高图像处理算法的执行效率,尤其在遥感图像处理、导弹武器系统的实时景像匹配制导等一些实时性要求较高的场合。在近二十年中,西方发达国家、俄罗斯等均在这些领域进行了大量研究并取得了具有实用价值的成果。如美国休斯公司生产的用于卫星上遥感图像处理的并行计算系统。美国、俄罗斯分别研制的巡航导弹景像匹配制导系统中的并行计算机。由西安微电子技术研究所研制的 LS MPP 就是一种面向实时图像处理应用的2D-Mesh嵌入式大规模并行处理计算机[1]。

图像数据分布是并行图像处理算法研究的一个重要内容。当前对图像数据分布的研究主要有以下两个特点:

(1)对图像数据分布的研究多侧重于对图像数据在数组空间划分实现的研究[2~4]。给定大小为 n×n 的图像,p 个处理机按照$\sqrt{p}\times\sqrt{p}$的2D-Mesh 排列,Lee 将图像数据的分布归结为6种类型的划分[2]:

(a)行划分(Row Partition)。图像数据的第 i 行分配到处理机 $P_{f_{row}(i)}$,$f_{row}(i)=\lfloor i/(n/p)\rfloor$,i=0,…,n-1。

(b)列划分(Column Partition)。图像数据的第 j 列分配到处理机 $P_{f_{column}(j)}$,$f_{column}(j)=\lfloor j/(n/p)\rfloor$,j=0,…,n-1。

(c)行-循环划分(Row-Cyclic Partition)。图像数据的第 i 行分配到处理机 $P_{f_{row-cyclic}(j)}$,$f_{row-cyclic}(i)=\lfloor i/b\rfloor \bmod p$,i=0,…,n-1,b 为连续打包到同一处理器上的行数。

(d)列-循环划分(Column-Cyclic Partition)。图像数据的第 j 列分配到处理机 $P_{f_{column-cyclic}(j)}$,$f_{column-cyclic}(j)=\lfloor j/b\rfloor \bmod p$,j=0,…,n-1,b 为连续打包到同一处理器上的行数。

(e)块划分(Block Partition)。像素(i,j)分配到处理机 $P_{f_{block}(i,j)}$,如果$\lfloor j/(n/\sqrt{p})\rfloor$是偶数,则 $f_{block}(i,j)=\lfloor j/(n/\sqrt{p}\rfloor\sqrt{p})+\lfloor i/(n/\sqrt{p})\rfloor$;如果是奇数,则 $f_{block}(i,j)=\lfloor j/(n/\sqrt{p})\rfloor\sqrt{p}+\lceil j/(n/\sqrt{p})\rceil-\lfloor i/(n/\sqrt{p})\rfloor$。

(f)块-循环划分(Block-Cyclic Partition)。像素(i,j)分配到处理机 $P_{f_{block-cyclic(i,j)}}$,如果 $f_{block-cyclic(i,j)}=(\lfloor i/b\rfloor+\lfloor j/b\rfloor) \bmod p$,b 是打包到同一处理器上的元素数。

(2)许多图像处理算法均假定待处理图像的尺寸与处理器阵列的规模具有相同的大小,或假定每一处理单元的局部存储器能够存储对应区域子图[5,6],从而使图像数据能够一次全部装入并行处理单元阵列。前者假定要求处理单元阵列足够大,如256×256或1024×1024甚至更高;后者要求有较大规模的局部存储器,如 IlliacIV 包含16KBytes 的局部存储器,可一次装入128×128×8位数字图像。

然而在图像数据相关的情况下,简单的数据划分不能够实现计算上的数据对准,因此实现数据对准必须通过重复划分或处理单元间的数据通信完成,对于许多分布共享存储的松耦合计算系统,处理单元之间的数据通信周期远大于计算周期,因此考虑处理机间的大量数据通信是很困难的。

另一方面,受到工艺的限制或体积的要求,在一些应用领域,待处理图像尺寸远大于处理单元阵列或局部存储器的规模,图像数据不可能一次完全装入处理单元阵列,因此对这类图像进行处理只能分步分块进行,每一步仅针对图像的一块进行相应的处理。

2 LS MPP 共享存储空间中数据访问分析

在文[7]中我们已经对 LS MPP 的体系结构进行了说明,本文将不再进行重复。LS MPP 的存储器分为两部分:(1)*程序存储器*。程序存储器用于存放执行的指令序列、对共享存储器进行数据访问的地址参数及循环指令执行所需的循环计数。两部分具有不同的地址总线、数据总线和读写控制信号。在指令执行过程中,程序存储器中的信息是禁止进行写操作的。(2)*共享存储器*。用于存放计算操作的数据。典型的读共享存储器的过程分三步。

(1)设定数据访问的基地址。执行相应指令可以给控制器中的数据地址寄存器赋初值。指令操作数形式为:(AR$_i$,AR$_j$,disp$_1$)。

AR$_i$ 对应的 AR$_0$~AR$_2$,AR$_j$ 为指定的一个控制器寄存器,disp$_1$ 为一立即数,控制器把程序存储器中地址(AR$_j$)+disp$_1$处的内容装入地址寄存器 AR$_i$ 中,需要时 AR$_j$ 可以设为 AR$_3$,即值为0的寄存器。控制器将把程序存储器中地址为 disp$_1$ 处的内容装入地址寄存器中。

(2)将指定共享存储器的内容读到阵列缓冲器中。指令操

*)本文受到国防95预研项目(编号:45.7.1)的资助

作数形式为：

(AR_0,$disp_2$)，读单个数据
(AR_0,$disp_2$)，读入 N×1的列向量
(AR_0,$disp_2$)，读入 N×N 的矩阵

执行读数据操作时，指令中设定一个立即数 $disp_2$作为相对基地址的偏移地址，从 AR_0+disp_2指定共享存储器中的位置开始读入数据，指令执行完成后，地址寄存器 AR_0的值也随之改变为：

AR_0+disp_2(读单个数据)或
$AR_0+disp_2+(N-1)\Delta N_1$(N×1读列向量)或
$AR_0+disp_2+(N-1)\Delta N_2+N\times(N-1)\Delta N_1$(对应 N×N 指令)

其中 ΔN_1、ΔN_2分别对应 AR_1(列间距)和 AR_2(行间距)。

(3)将阵列缓冲器中的内容读到指定的寄存器平面上。

写共享存储器的操作过程与读操作类似，不同之处在于写操作首先将指定处理单元阵列中寄存器的内容写入阵列缓冲器，然后由阵列缓冲器写到指定的共享存储器中。

通过以上分析可以得出以下结论：

(1)如果数据访问前不指定数据访问基地址，则使用当前地址寄存器中的默认值作为起始地址。

(2)如果指定数据访问地址，则设置的基地址为((AR_i)+$disp_1$)。

(3)数据访问的起始地址为((AR_i)+$disp_1$)+$disp_2$。

(4)数据访问结束后，地址寄存器 AR_0的值为数据访问的最后一个数的地址。

3 有序覆盖与有序划分

针对并行图像处理数据分布的特点，本文提出了有序覆盖和有序划分的概念。

定义1 给定一个集合 F，$\pi=\{F_0, F_2, \cdots, F_c\}$为一集合族，满足：

(1)$(\pi,\leqslant)$是一良序集。即对于任意 i、j，$0\leqslant i\leqslant c$，$0\leqslant j\leqslant c$，$i\neq j$，如果 $i\leqslant j$，则 $F_i\leqslant F_j$，称为 F_i 先于 F_j

(2)对于任意 i，$0\leqslant i\leqslant c$，$F_i\subseteq F$

(3)$F=\bigcup_{i=0}^{c}F_i$

则称 π 为 F 的一个有序覆盖，F_i 称为覆盖 π 下的一个块，i 为 F_i 在 π 中的坐标。

定义2 给定一个集合 F，$\pi=\{F_0,F_2,\cdots,F_c\}$为一有序集合族，满足：

(1)π 是 F 的一个有序覆盖

(2)$F_i\cap F_j=\phi$，$0\leqslant i\leqslant c$，$0\leqslant j\leqslant c$，$i\neq j$

则称 π 是 F 的一个有序划分。

定义3 给定集合 F、G，$\pi_1=\{F_0, F_2,\cdots,F_{c1}\}$、$\pi_2=\{G_0, G_2,\cdots,G_{c2}\}$分别对应 F、G 的一个有序覆盖。如果满足以下条件：

(1)存在一个映射 h：G→F。

(2)对于任意 $g, g\in G_i\Rightarrow h(g)\in F_i$。

则称 G、F 在有序覆盖 π_2、π_1下相似，记为 $\pi_2(G)\backsim\pi_1(F)$。

在不引起混淆的情况下，本文后面的覆盖与划分均指有序覆盖与有序划分。

假定图像数据是以一个 M 行 M 列的二维数组形式存储在共享存储空间中，这样的数组简称为图像数组，以区别其它应用的变量和常数。

给定一个图像数组 $F=\{f(x,y)\mid 0\leqslant x\leqslant M-1,\ 0\leqslant y\leqslant M-1\}$，可以找出它的多个覆盖或划分。当图像的尺寸大于处理器阵列规模时，根据需要对图像数组产生一个覆盖，覆盖中的块的大小与处理器阵列规模相适应，这个覆盖形成对图像数组的一个数据分布方案，针对整个图像的处理可以看作是针对覆盖中每一个块处理的总和。

为了便于问题讨论，本文做以下假定：

(1)假定待处理的图像为256级的灰度图像，每一像素取值为0～255，M 是 N(处理器规模 N×N)的整数倍，即 $M=N\times z$(z 为整数，$z>1$)。

(2)由于图像数组是按照字典序排列的集合，假定图像数据的覆盖或划分中的每一块内部也是按照字典序排列的子集，且块的内部行间距和列间距是一常数，块间距离也是一常数。

在块等秩划分的情况下，图像被划分的块数 $C=z\times z$，每块大小为 N×N。对于块间包含重复数据的覆盖，其块数 $C'>C$。

寻找图像数组覆盖与划分时应尽可能使相关的数据引用保持在同一块内。在过去的研究中图像数组的划分都是在数组空间上进行的。实际上，数据的覆盖和划分可以根据需要在不同的数据空间如迭代空间、共享存储空间[7]中进行。例如典型的五种块划分如图1所示。

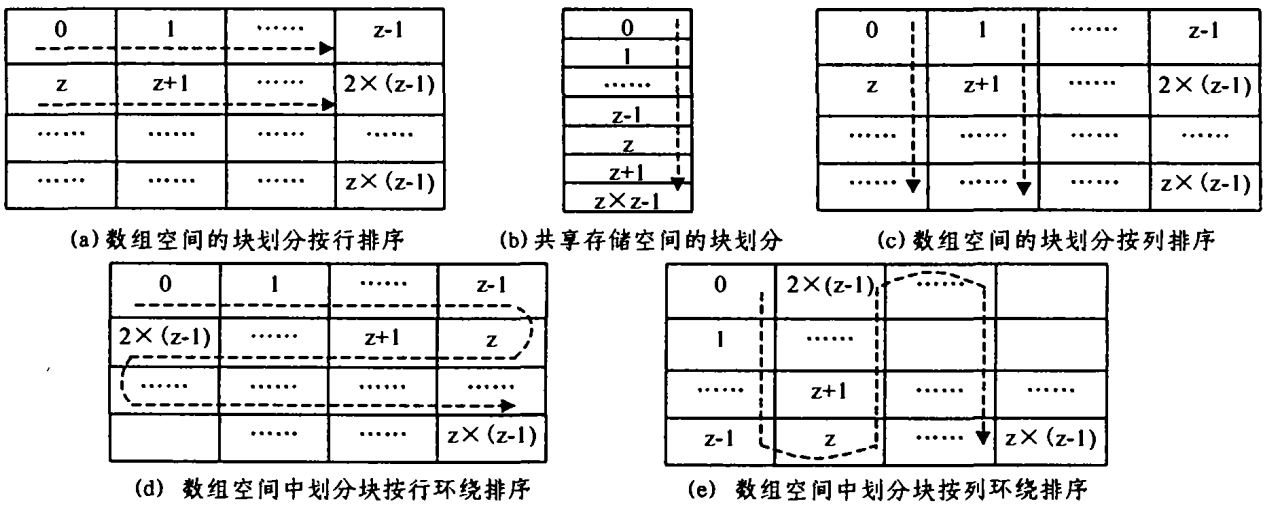

图1 数组空间中图像数据划分的表示(图中数字为块标识，箭头表明排序方向)

在包含重复数据的图像覆盖中，块排序以数据块在数据空间中出现的第一个元素作为块的标识。覆盖块的排列顺序也可以分为行排序、列排序、按行环绕排序、按列环绕排序等多种类型。

假定图像数据保存在数组 F 中，f 为数组 F 的起始地址。

图1(a)中第 i 块($0\leqslant i<z\times z$)起始数据在数组空间中的坐标：

$$(x'_0,y'_0)=(\left\lfloor\frac{i}{z}\right\rfloor\times z,(i-\left\lfloor\frac{i}{z}\right\rfloor\times z)\times z)$$

可以得出对应共享存储空间的地址为[7,8]：

$$f+\left\lfloor\frac{i}{z}\right\rfloor\times z\times M+(i-\left\lfloor\frac{i}{z}\right\rfloor\times z)\times z$$

块中相邻元素行间距为1，列间距为 M－N＋1。

图1 (b)中第 i 块起始数据对应的共享存储空间地址为 f＋i×N×N，块中相邻元素行间距为1，列间距为1。可以求出对应数组空间的坐标[7,8]：

$$(x'_0,y'_0)=(\left\lfloor\frac{i\times N\times N}{M}\right\rfloor,i\times N\times N-\left\lfloor\frac{i\times N\times N}{M}\right\rfloor\times M)$$

同理可以求得图1(c)、(d)、(e)中数据块的访问地址。显然，如果可以计算出每一块的起始地址并赋值给相应数据访问的地址寄存器，则图1中的所有数据划分均可实现数据块通信。

4 图像数据访问的实现

给定图像数组的一个覆盖(包括图像数组的划分)，虽然可以计算出每一个数据块通信的参数，但计算出的块起始地址是块在覆盖中坐标 i 的一个函数而不是常数，而数据地址访问只能针对常数进行。另一方面，在图像处理过程中，往往包含多个图像数组的数据访问，使得数据访问的地址确定更为复杂。

根据前面对数据访问的分析，本文提出两种数据访问实现的方法：

1. 如果能够确定访问的图像数组在共享存储空间中的物理地址，就可以确定覆盖中每一个数据块访问地址。其步骤如下：

(1)编译程序将图像数组在共享存储空间中存储的物理地址固定，然后计算出每个块的起始物理地址(在这种情况下计算出的块起始地址为常数)。

(2)将这些地址常数值顺序写入到目标代码中的地址段。

(3)每次访问块数据时，通过设置寄存器 AR_i 的值，即每次 $AR_i+1\rightarrow AR_i$，使$((AR_i)+0)$指向计算出的程序存储器中常数地址，将此常数地址装入起始地址寄存器中，从而实现块通信。

这种方法的缺点是过早地确定了图像数组在共享存储空间中存储的物理地址，而物理地址通常是在多模块链接生成机器指令或指令装载时确定的。过早确定物理地址使多模块程序链接难以实现(因为不同的模块通常是分开编译的)；程序只能装载到固定物理地址运行；且当数据块较多时，代码的长度也会因数据地址数量的增加而增长。

2. 根据覆盖中数据块访问前后数据地址寄存器的变化和图像数组相邻数据块访问的地址差值确定数据块访问地址。

假设：(1)图像数组操作过程中所使用的非图像数组数据，应能通过编译优化调整到图像数据块装入前完成，使图像数组操作的过程中只包含图像数组对共享存储空间的访问。(2)如果操作过程中包含多个图像数组，它们均按照各自的覆盖进行数据访问，则这些图像数组应在当前程序模块中定义，且应通过编译优化使它们均与特定的图像数组在定义的覆盖下相似。即对于图像数组引用 $F^1\sim F^m$，存在 k，$1\leqslant k\leqslant m$，使得 $\pi_i(F^i)\backsim\pi_k(F^k)$。$\pi_i$ 为对应 F^i 的覆盖，$(1\leqslant j\leqslant m)$。

由此我们可以得出：

性质1 给定图像数组引用 $F^1\sim F^m$，则可以根据数据访问过程中数据地址寄存器变化确定数据访问地址，若满足：

(1)访问的一个数据块后地址寄存器存储的数据地址与该图像数组下一个相邻数据块的起始地址之差为一个可计算的常数；

(2)在同一模块中定义的两个图像数组在共享存储空间中存储地址之差为一个可计算的常数。

实现图像数据访问的算法如下：

输入：并行图像处理算法程序段。

输出：针对该程序段的图像数组引用链表和预装入数据链表及图像数组访问的初始状态。

(1)初始化。初始化图像数组引用链表、预装入数据链表、链表状态变量 R＝0、当前地址位置变量 Position＝0、访问状态 AccessState＝0，数据访问初始状态(包括块访问起始地址 InitAddr＝″、列间距 $\Delta N_1=0$、行间距 $\Delta N_2=0$)。

图像数组引用链表中每一节点是如下的一个五元组：

(ID、ImageArrayName，Offset，BlockTempName，Line，Flag)

五元组中各项分别为引用的图像数组标识、图像数组名、图像数组访问的偏移地址，转换为中间代码后引用子块对应的临时变量名、临时变量名按照出现顺序依次标记为 temp0、temp1……，第一次需要进行数据读入的语句标号或最后一次需要输出的语句标号、读入或输出的状态标志。

预装入数据链表用于描述需要在图像处理操作前装入的带有初值的变量。

(2)从程序段入口读入一个语句。

(3)对语句中的变量引用依次进行分析。

(4)如果该变量是图像数组变量，则转到6。

(5)查找程序段中该语句执行前的语句，如果不包含该变量且该变量不在该语句中赋值，则将该变量插入预装入数据链表。转到13。

(6)查找程序段中该语句执行前的语句序列，如果该图像数组元素未被引用或在该语句中赋值，则执行下列操作，否则转到13。

(7)如果该图像数组元素是在该语句中赋值，且查找程序段中该语句执行后的语句包括该图像数组元素的赋值，则转到13。

(8)根据引用下标计算数据块访问的初始位置。

(9)如果 R＝0则在此之前没有图像数组引用，令 InitAddr＝当前图像数组名，根据数组引用下标计算块的列间距 ΔN_1 和行间距 ΔN_2，R＝1。

(10)如果当前地址位置变量 Position 为空，则设置 Position 为此图像数组 ID，Offset＝0，访问状态 AccessState 为0(0表示指针指向块的起始位置)。转到12。

(11)如果 AccessState＝0，Offset＝0，否则 $Offset=-((N-1)\Delta N_2+N\times(N-1)\Delta N_1)=1-N\times N$；再判断如果 Posizion 不等于该数组 ID，则 Offset＝Offset＋(该数组的起始地址－Position对应的数组起始地址)。设置当前地址位置

(下转第107页)

先严格确定的接口函数与外界进行交互作用，能根据目标主动规范化自己的行为，使用户界面达到"人性化"；如需要集成的问题：通过给旧系统上包装一层 Agent 外壳，其它系统可以调用旧系统的功能。

运用移动 Agent 技术开发应用系统一般有以下的步骤：

·分析系统的特点，选择合适的实现技术：决定哪些采用移动代理实现，哪些采用其它方法实现。

·移动 Agent 的功能设计：确定系统中决定用 Agent 技术实现部分的数据和功能，移动 Agent 间明确分工后，应当根据各自功能确定内部数据，注意，移动 Agent 的内部数据应尽可能少，减少由于移动带来的网络负担。

·Agent 的接口设计：Agent 的接口设计非常关键，它往往是系统性能好坏的关键。这时既要考虑系统中的 Agent 间交互方式又要考虑 Agent 与非 Agent 部分的交互方式。

·Agent 的详细设计：应当了解目前移动 Agent 平台各自的优缺点，选择合适的平台。

·Agent 的运行与维护：目前由于移动 Agent 可靠性还不高，维护工作特别重要。

总结和展望 移动 Agent 不同于基于过程的 RPC（如 OSF/DCE 中的），也不同于面向对象的对象引用（如 OMG/CORBA，OLE/DCOM 和 Java/RMI 中的），其独特的对象传递思想和卓越的特性给分布式计算乃至开放系统带来了巨大的革新。但 Agent 平台间的互操作还不是非常理想，容错性、可靠性、安全性还应当加强，面向 Agent 的软件工程也应该规范化。而且要正确理解面向 Agent 的方法，而不能有偏差。须知 Agent 方法不是万能的，Agent 的不成熟限制了它的应用领域，如整个系统不可预测，不可确定，哪个 Agent 将与其它哪个 Agent 以什么方式交互实现什么目标，这是不能事先确定的；而且由于 Agent 可以自由作出决策，我们不能保证 Agent 之间的依赖关系能得到有效管理，由此可能导致死锁；又如整个系统的性质和行为在设计阶段不能确定，整体行为只有在运行时才能体现。其二、Agent 受理论和硬件的限制。Agent 由于自身带有大量的精神状态，在某些情况下，性能反而不高。其三、在应用 Agent 方法进行具体设计时必须考虑异质性问题、不确定、不完备和矛盾信息问题、实时操作问题以及适应性问题。但我们相信，移动 Agent 技术将会日趋成熟，Internet 与移动 Agent 技术的结合将给远程教学、电子商务带来无限发展机遇，并在移动计算、信息服务等新兴应用领域里找到用武之地。

参 考 文 献

1 Joshi N, Ramesh V C. On Mobile Agent Architectures. avialable at: http://vcr.ece.iit.edu/papers/mobileAgents/mobile Agents.html

2 Cockayne W R, Zyda M. Mobile Agents. Manning Publications Co., http://flower.ce.cnu.ac.kr/~dkkang/mobile_Agent_books/HTML/Front.htm, 1998

3 张云勇. 面向 Agent 的软件工程，电子科技大学计算机学院博士生研究报告，2000年8月

4 刘锦德、张云勇. 一个实用的移动 Agent 系统(Aglet)的综述. 计算机应用

5 胡健. 开放式分布协作信息技术. 电子科技大学计算机学院博士论文，2000年10月

6 马俊涛，刘积仁. 移动 Agent 体系结构及关键技术探讨. 小型微型机系统，1998，19(2)

7 刘建勋、李仁发、张申生，移动 Agent 及其安全性问题，计算机工程与应用，2000年第7期

8 王济勇、温涛、李春光等，一个基于 Java 的移动 Agent 安全体系，计算机工程与应用，2000，(10)：141～144

9 董红斌、石纯一，移动 Agent 系统的安全问题，计算机科学，2000，27(10)

10 张云勇、刘锦德，移动 Agent 互操作性研究，计算机科学，2001，28(9)

11 张云勇、刘锦德，一种基于移动 Agent 的主动网的框架方案，计算机应用，2001，21(7)

12 印鉴、刘斌、邹胜等，用演化 Agent 方法处理整数线性规划问题，小型微型机系统，2000，(6)：608～610

13 王柏、王红漫、邹华，分布计算环境，北京邮电大学出版社，2000.8

14 Danny B. Lange, Mobile Objects and Mobile Agents: The Future of Distributed Computing available via http://www.generalmagic.com/asa/danny/Ecoop98.pdf

（上接第126页）

变量 Position 为该数组 ID，访问状态 AccessState 为1（指针指向块的最后位置）。

(12)将该图像数组信息写入图像数组引用链表。

(13)如果有待分析的语句，读入下一语句，转到3。

(14)算法结束。

结论 图像处理过程中包含大量独立的重复运算，因此作为提高图像处理速度的重要途径—并行图像处理一直是图像处理研究领域中的热点问题。本文在对当前图像数据分布研究的基础上，提出了图像数据有序覆盖和有序划分的概念，将图像处理过程中图像数据的访问描述为一组相似有序覆盖访问的过程。给出了基于不同数据空间的图像数据划分方法。在对 LS MPP 数据块访问地址的计算方法进行了深入分析的基础上，建立了实现图像覆盖块连续访问的两种模式，并详细讨论了其中一种模式的图像数据访问地址计算及数据访问实现算法。这些方法在面向图像处理的 LS MPP C 编译器中得到了实现。

参 考 文 献

1 沈绪榜. MPP 嵌入式计算机设计. 清华大学出版社，1999

2 Lee Cheolwhan. Global Optimization for Mapping Parallel Image Processing Tasks on Distributed Memory Machines: [Technical Report]. University of Califonia. 1997

3 Copty N. A Data Parallel Algorithm for Solving the Region Growing Problem on the Connection Machine: [Technical Report]. Syracuse University, 1994

4 Bevilacqua A. Parallel image restoration on parallel and distributed computers. Parallel Computing, 2000, 26(4): 495～500

5 Potter J L. Image Processing on the Massively Parallel Processor. IEEE Computer, 16(1), 1983

6 Ibrahim H A. Low-Level Image Analysis Tasks on Fine Grained Tree-Structured SIMD Machines. Journal of Parallel and Distributed Computing, 4, 1987

7 王晖，等. LS SIMD C 编译器的数据通信优化算法. 计算机科学，2001.9.

8 王晖. LS MPP C 编译器的研究与实现. 西北工业大学博士学位论文，2001.4