基于总变分最小化模型的异步并行 GPU 加速算法

路万里, 蔡爱龙, 郑治中, 王林元, 李磊, 闫镔
解放军信息工程大学信息系统工程学院, 河南 郑州 450002

摘要 相比于传统同步并行计算策略,在异步并行计算框架下,针对最常用的总变分(TV)最小化重建模型,通过将其转化为不动点迭代问题,并利用异步交替方向法(ADM)进行求解,推导出基于 TV 最小化模型的异步 ADM 迭代重建算法,即异步交替方向总变分最小化算法(Async-ADTVM)。利用消息传递接口技术将该算法在图形处理器(GPU)集群上进行测试,进一步提高了原始基于 TV 最小化模型的迭代重建算法的计算效率。实验表明,该算法在计算求解精度上略优于 ADTVM 算法,同时在 GPU 性能存在差异的条件下相比传统多 GPU 加速策略可获得更高的加速比。

关键词 成像系统; 优化类重建算法; 异步并行迭代; 总变分最小化模型; 多图形处理器加速
中图分类号 TP391; O434.1 **文献标识码** A **doi**: 10.3788/AOS201838.0411004

Asynchronous Parallel GPU Acceleration Method Based on Total Variation Minimization Model

Lu Wanli, Cai Ailong, Zheng Zhizhong, Wang Linyuan, Li Lei, Yan Bin
Institute of Information System Engineering, Information Engineering University of Chinese People's Liberation Army, Zhengzhou, Henan 450002, *China*

Abstract Compared to the traditional synchronous parallel computing, an asynchronous parallel alternating direction method (ADM) for total variation (TV) minimization reconstruction, namely asynchronous alternating direction total variation minimization method (Async-ADTVM), is proposed in this paper. Under the asynchronous parallel computing framework, Async-ADTVM transforms TV minimization reconstruction model to the problem of fixed-point iteration, which is solved by asynchronous parallel ADM. It is implemented on the graphics processing unit (GPU) cluster based on message passing interface technology. Experimental results show that the proposed Async-ADTVM can provide a little higher calculation accuracy than ADTVM. Meanwhile, it can provide a higher speed-up ratio than the traditional multi-GPU acceleration strategy when the performance of each GPU is different.

Key words imaging systems; optimization-based reconstruction method; asynchronous parallel iteration; total variation minimization model; multi-graphics processing unit acceleration
OCIS codes 100.2000; 110.3010; 340.7440

1 引 言

计算机断层扫描(CT)成像技术可在非接触、不破坏的前提下快速有效地获取物体的内部结构信息,自上世纪 70 年代出现以来已经被广泛地应用于医疗诊断以及工业检测[1]。三维锥束 CT 由于其空间分辨率高、扫描速度快以及射线利用率高等方面的优势,得到了广泛的应用。其中,圆轨迹扫描因其扫描方式简单、易操作等优点成为最常用的一种扫描方式。在针对圆轨迹扫描的重建算法中,Feldkamp, Davis 和 Kress 提出的 FDK(Feldkamp-Davis-Kress)算法[2]是最著名的一种解析类重建算法,但是该算法对投影数量的需求较大,而大量的投影采集意味着更多的辐射剂量,会对人体产生较大

收稿日期: 2017-04-05; **收到修改稿日期**: 2017-05-13
基金项目: 国家自然科学基金(61372172,61601518)
作者简介: 路万里(1992—),男,硕士研究生,主要从事锥束计算机断层扫描图像重建算法方面的研究。E-mail: luwanli1992@163.com
导师简介: 闫镔(1976—),男,博士,教授,主要从事成像与智能信息处理方面的研究。E-mail: ybspace@hotmail.com(通信联系人)

的损害。对于医用 CT，辐射剂量同样是人们重点关注的问题之一。医用 CT 成像的一个主要目标就是在尽可能降低辐射剂量的同时获取高质量的成像结果。目前降低辐射剂量的方法一般可以分为两种[3-4]：一种方法是降低扫描电压电流，但是该方法会降低投影数据的信噪比，从而降低图像的重建质量；另一种方法是减小扫描角度，该方法不但可以有效降低患者接收的辐射剂量，还可以缩短扫描时间，因此通常采用减小扫描角度的策略来降低辐射剂量。常采用增大扫描间隔的方式减少扫描角度，但稀疏角度采样条件下的投影数据不满足 Tuy-Smith 数据完备性条件[5-6]。数学上，稀疏角度图像重建问题是一个病态求逆的不适定问题[7-8]，传统的解析重建算法以及迭代重建算法在处理稀疏角度问题时并不十分有效。

2006 年压缩感知(CS)理论[9]的提出为解决稀疏采样条件下的图像重建问题提供了新的思路。基于总变分(TV)最小化模型的迭代重建算法被提出，该模型利用一般图像的梯度幅度图像具有稀疏性的特点，使用稀疏优化的模型与求解算法，获得了精度较高的重建结果[10]。利用该性质，学者针对不同问题提出了多种重建算法[11-13]，但是该类算法在处理锥束 CT 大规模数据重建问题时仍然存在计算量大、耗时长等问题。其中，消耗的主要时间为正、反投影计算部分，而正、反投影因其具有较好的并行性，通常可采用加速策略对其进行改进，如图形处理器(GPU)加速等[14-15]。

为了进一步提升加速效果，将多 GPU 加速计算策略应用于稀疏角度迭代重建算法中[16-18]。多个 GPU 的引入可成倍地加速数值计算，获得更高的加速比。但这类加速算法均为同步并行加速算法，即每轮迭代各节点需等待所有节点结束后再进行下一轮迭代。对于同步并行算法，当多个加速部件间性能存在差异时，整体加速效果会受到最慢节点的影响，从而影响了其实际加速效果。2016 年 Peng 等[19]提出了一种异步并行计算框架，证明了其收敛性，并给出了算法收敛的条件及多种特例，为异步并行计算提供了理论支撑。本文在该异步并行计算框架下，利用异步交替方向法(ADM)推导得到基于 TV 最小化模型的异步并行重建算法，并利用 GPU 集群对其进行算法加速。相比传统 GPU 加速算法，该算法在理论上保证了异步并行的正确性，同时可将多 GPU 加速的优势发挥到最大，得到最理想的加速效果。

2　异步交替方向总变分最小化(Async-ADTVM)迭代重建算法

2.1　异步并行计算框架

为保证内容的完整性，首先介绍异步并行计算框架的相关内容。对于大规模、高复杂度计算，通常采用并行加速技术来提升计算效率，在同步并行计算方式中，计算较快的节点需等待较慢节点的计算结束，最慢节点的计算性能将直接影响整体的加速效果。而采用异步并行计算技术时，计算效率较高的节点无需等待即可进行下一步计算，消除了计算节点间的空闲等待时间，提高了资源利用率以及计算效率。进行异步并行计算时，不同节点的数据传输时间点有所差异，这样可避免同步计算时数据传输所带来的通信与存储访问拥堵，提高了整体的通信效率。同步并行计算以及异步并行计算示意图如图 1 所示。图中每一行表示一个节点的计算-时间序列，灰色部分表示节点正在计算，白色部分表示节点处于空闲状态。图 1(a)为同步并行计算时各节点在不同时间序列时的计算状态，因计算较快的节点需等待最慢节点的计算，故节点间存在大量的空闲时间，导致计算资源的利用率下降。图 1(b)为异步并行计算时各节点在不同时间序列时的计算状态，异步并行计算消除了计算节点的等待时间，充分利用计算资源，提升了系统的加速效果。

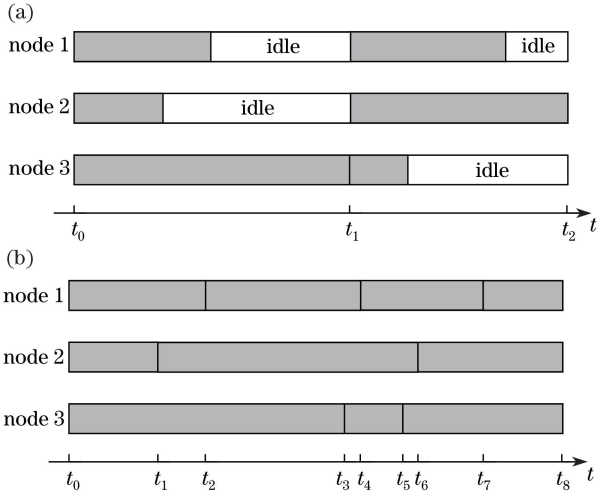

图 1　(a)同步并行计算；(b)异步并行计算

Fig. 1　(a) Synchronous parallel computing; (b) asynchronous parallel computing

对于同步并行计算，其算法收敛性与原始算法相同，而对于异步并行计算，每个节点每轮计算数据可能存在差异，导致算法收敛存在不确定性。对于该问题，2016 年 Peng 等[19]从不动点问题求解出

发，提出了异步并行计算框架并证明了其收敛性。该文献中异步并行计算框架的计算求解过程如下。

不动点问题可以表示为

$$\text{find}\quad \boldsymbol{x}^{*}\in\mathcal{H}\quad \text{s.t.}\quad \boldsymbol{x}^{*}=\boldsymbol{T}\boldsymbol{x}^{*}, \tag{1}$$

式中 $\mathcal{H}$ 表示希尔伯特空间的笛卡尔积，$\boldsymbol{x}^{*}$ 表示对于算子 $\boldsymbol{T}:\mathcal{H}\to\mathcal{H}$ 的不动点。当算子 $\boldsymbol{T}$ 非扩张，并且具有一个不动点时，该问题在利用异步计算框架时具有收敛性保证。设 $\boldsymbol{I}$ 为单位矩阵，令 $\boldsymbol{S}=\boldsymbol{I}-\boldsymbol{T}$，此时原问题可采用以下方式进行求解

$$\boldsymbol{x}^{k+1}=\boldsymbol{x}^{k}+\alpha(\boldsymbol{T}\boldsymbol{x}^{k}-\boldsymbol{x}^{k})\quad\text{or}\quad \boldsymbol{x}^{k+1}=\boldsymbol{x}^{k}-\alpha\boldsymbol{S}\boldsymbol{x}^{k}, \tag{2}$$

式中 $\boldsymbol{x}^{k}$ 表示在全局变量中更新前存储的 $\boldsymbol{x}$ 值；$\alpha\in(0,1)$，代表步长；k 代表迭代的轮数。在并行计算框架下，多个节点共同更新 $\boldsymbol{x}^{k}$ 中的元素以便通过更快速地迭代得到最终结果。对于任意一个节点 i，更新公式可表示为

$$\boldsymbol{x}^{k+1}=\boldsymbol{x}^{k}-\alpha\boldsymbol{S}_{i}^{k}\hat{\boldsymbol{x}}^{k}, \tag{3}$$

式中 $\hat{\boldsymbol{x}}$ 表示在节点内计算更新 $\boldsymbol{S}_{i}^{k}\boldsymbol{x}$ 前从全局存储中读取的 $\boldsymbol{x}$ 值，$\boldsymbol{S}_{i}^{k}\boldsymbol{x}$ 可表示为

$$\boldsymbol{S}_{i}^{k}\boldsymbol{x}:=[0,\cdots,0,(\boldsymbol{S}\boldsymbol{x})_{i}^{k},0,\cdots,0]。 \tag{4}$$

对于同步并行计算，不同节点中更新 $\boldsymbol{x}^{k}$ 值时均处于同一轮迭代，因此 $\hat{\boldsymbol{x}}^{k}\equiv\boldsymbol{x}^{k}$。而对于异步并行计算，因为某一节点在读取 $\boldsymbol{x}^{k}$ 的值后可能被其他节点更新，导致不同节点中更新 $\boldsymbol{x}^{k}$ 值时可能处于不同轮迭代，即 $\hat{\boldsymbol{x}}^{k}\neq\boldsymbol{x}^{k}$，这就是同步并行与异步并行之间的本质区别，因此将所求的问题转化为不动点迭代的形式，利用该框架进行求解便可得到异步并行算法。

2.2　CT 重建模型及 TV 最小化模型

对于 CT 系统成像，迭代型重建算法通常可采用线性方程组来描述，对成像模型进行离散化之后可得到如下模型：

$$\boldsymbol{W}\boldsymbol{x}=\boldsymbol{p}, \tag{5}$$

式中 $\boldsymbol{p}\in\mathbb{R}^{l}$，表示获取的投影数据；$\boldsymbol{x}\in\mathbb{R}^{s}$，表示重建的体数据；$\boldsymbol{W}\in\mathbb{R}^{l\times s}$，表示离散化的投影矩阵。采用射线穿过体素的长度来表示 $\boldsymbol{W}$ 中各元素的数值。当投影采样为稀疏采样时，该方程是不适定的，通常可采用稀疏优化的方法进行求解。对于(5)式问题的不适定性，最直接的思路就是利用稀疏性作为先验，构造一个正则化模型进行求解。CS 理论则表明在满足一定稀疏性的条件下，通过求解 L1 范数最小化问题可对(5)式进行精确求解。在 CT 成像中，通常认为图像的梯度幅度在平坦区域内为 0 而在边缘上不为 0，因此是非常稀疏的。TV 最小化目标的实质是图像的梯度幅度的 L1 范数最小化。因此对于稀疏角度 CT 图像重建，采用 TV 最小化模型可取得良好的重建结果。采用符号∇来表示图像的梯度算子，因此可得到如下 TV 最小化重建模型：

$$\begin{cases}\boldsymbol{x}^{*}=\arg\min\limits_{x}\|\nabla\boldsymbol{x}\|_{1}\\ \text{s.t.}\quad |\boldsymbol{p}-\boldsymbol{W}\boldsymbol{x}|\leqslant\varepsilon\end{cases}, \tag{6}$$

式中 ε 表示系统的噪声大小情况。令 $\boldsymbol{z}=\nabla\boldsymbol{x}$，则重建模型[(6)式]可表示为

$$\begin{cases}\boldsymbol{x}^{*}=\arg\min\limits_{x}\|\boldsymbol{z}\|_{1}+\dfrac{\lambda}{2}\|\boldsymbol{W}\boldsymbol{x}-\boldsymbol{p}\|_{2}^{2}\\ \text{s.t.}\quad \boldsymbol{z}=\nabla\boldsymbol{x}\end{cases}, \tag{7}$$

式中系数 $\lambda\in\mathbb{R}$，表示保真项参数，主要作用为控制保真项 $\|\boldsymbol{W}\boldsymbol{x}-\boldsymbol{p}\|_{2}^{2}$ 在目标函数中的比重；$\|\boldsymbol{z}\|_{1}$ 为差分项，这里采用各向异性的 TV 范数进行计算。

由于该问题中包含两个变量 $\boldsymbol{x}$ 和 $\boldsymbol{z}$，这与异步并行计算框架有所差异，因此可以考虑其对偶问题。首先得到其拉格朗日函数为

$$\mathcal{L}(\boldsymbol{z},\boldsymbol{x};\omega)=\|\boldsymbol{z}\|_{1}+\frac{\lambda}{2}\|\boldsymbol{W}\boldsymbol{x}-\boldsymbol{p}\|_{2}^{2}+\omega^{\mathrm{T}}(\boldsymbol{z}-\nabla\boldsymbol{x}), \tag{8}$$

进而可以得到其对偶问题：

$$\omega^{*}=\arg\min\ \boldsymbol{f}^{*}(\omega)+\boldsymbol{g}^{*}(\nabla^{\mathrm{T}}\omega), \tag{9}$$

式中 $\boldsymbol{f}(\boldsymbol{z})=\|\boldsymbol{z}\|_{1}$，$\boldsymbol{g}(\boldsymbol{x})=\dfrac{\lambda}{2}\|\boldsymbol{W}\boldsymbol{x}-\boldsymbol{p}\|_{2}^{2}$，其对偶函数

$$\begin{cases}\boldsymbol{f}^{*}(\omega)=\sup\limits_{z}(\langle\omega,\boldsymbol{z}\rangle-\|\boldsymbol{z}\|_{1})\\ \boldsymbol{g}^{*}(\omega)=\sup\limits_{x}\left(\langle\omega,\boldsymbol{x}\rangle-\dfrac{\lambda}{2}\|\boldsymbol{W}\boldsymbol{x}-\boldsymbol{p}\|_{2}^{2}\right)\end{cases}, \tag{10}$$

式中 $\boldsymbol{f}(\boldsymbol{z})$ 和 $\boldsymbol{g}(\boldsymbol{x})$ 均为闭合凸函数。对于 $\min\boldsymbol{f}(\boldsymbol{z})+\boldsymbol{g}(\boldsymbol{x})$ 该类问题的异步迭代求解，文献[9]中给出的不动点迭代公式为

$$\boldsymbol{z}^{k+1}=\boldsymbol{z}^{k}-\alpha\boldsymbol{U}_{i}\circ(\boldsymbol{I}-\mathbf{refl}_{\gamma f}\circ\mathbf{refl}_{\gamma g})\boldsymbol{z}^{k}, \tag{11}$$

式中 $\boldsymbol{U}_{i}:\boldsymbol{x}\mapsto(0,\cdots,0,\boldsymbol{x}_{i},\cdots,0)$ 表示选择第 i 个坐标进行更新。考虑到锥束 CT 图像重建规模较大时 $\boldsymbol{x}$ 中的元素个数较多，采用块划分的策略进行更新，此时变量 $\boldsymbol{U}_{i}$ 将变为 $\boldsymbol{U}_{i_t}$，其中 $\boldsymbol{U}_{i_t}:\boldsymbol{x}\mapsto(0,\cdots,0,\boldsymbol{x}_{i_1},\cdots,\boldsymbol{x}_{i_t},\cdots,0)$ 表示对含 t 个元素第 i 个块进行更新，算子 **refl** 可定义为

$$\mathbf{refl}_{\gamma f}:=2\mathbf{prox}_{\gamma f}-\boldsymbol{I}, \tag{12}$$

$$\mathbf{prox}_{\gamma f}(\boldsymbol{x}):=\arg\min_{y}\boldsymbol{f}(\boldsymbol{y})+\frac{1}{2\gamma}\|\boldsymbol{y}-\boldsymbol{x}\|^{2}。 \tag{13}$$

对于(11)式，可采用异步交替方向法进行迭代求解。将(9)式代入该求解公式最终可得基于 TV 最小化模型的 Async-ADTVM 的更新公式为

$$\boldsymbol{x}^{k+1}\in\arg\min_{\boldsymbol{x}}\frac{\lambda}{2}\|\boldsymbol{W}\boldsymbol{x}-\boldsymbol{p}\|_2^2+\langle\omega^k,\nabla\boldsymbol{x}\rangle+\frac{\gamma}{2}\|\nabla\boldsymbol{x}\|^2,\tag{14}$$

$$\omega_g^{k+1}=\omega^k+\gamma\nabla\boldsymbol{x}^{k+1},\tag{15}$$

$$\boldsymbol{z}^{k+1}\in\arg\min_{\boldsymbol{z}}\|\boldsymbol{z}\|_1-\langle 2\omega_g^{k+1}-\omega^k,\boldsymbol{z}\rangle+\frac{\gamma}{2}\|\boldsymbol{z}\|^2,\tag{16}$$

$$\omega_f^{k+1}=2\omega_g^{k+1}-\omega^k-\gamma\boldsymbol{z}^{k+1},\tag{17}$$

$$\omega^{k+1}=\omega^k+\alpha(\omega_f^{k+1}-\omega_g^{k+1})。\tag{18}$$

首先求解(14)式，该式为关于 $\boldsymbol{x}$ 的二次函数，其最小值点可利用其导数为 0 进行求解：

$$\lambda\boldsymbol{W}^{\mathrm{T}}(\boldsymbol{W}\boldsymbol{x}-\boldsymbol{p})+\nabla^{\mathrm{T}}\omega^k+\gamma\nabla^{\mathrm{T}}\nabla\boldsymbol{x}=0,\tag{19}$$

从而可以得到该子问题的解析解为

$$\boldsymbol{x}^{k+1}=(\lambda\boldsymbol{W}^{\mathrm{T}}\boldsymbol{W}+\gamma\nabla^{\mathrm{T}}\nabla)^{+}(\lambda\boldsymbol{W}^{\mathrm{T}}\boldsymbol{p}-\nabla^{\mathrm{T}}\omega^k),\tag{20}$$

式中 $\boldsymbol{M}^{+}$ 表示矩阵 $\boldsymbol{M}$ 的 Moore-Penrose 伪逆，但是(20)式中 $\lambda\boldsymbol{W}^{\mathrm{T}}\boldsymbol{W}+\gamma\nabla^{\mathrm{T}}\nabla$ 过大，导致其伪逆求解非常困难。采用线性化近似的方法进行求解，利用当前点 $\boldsymbol{x}^k$ 对(14)式中 $\|\boldsymbol{W}\boldsymbol{x}-\boldsymbol{p}\|_2^2$ 以及 $\|\nabla\boldsymbol{x}\|^2$ 进行近似化处理可以得到

$$\|\boldsymbol{W}\boldsymbol{x}-\boldsymbol{p}\|_2^2\approx\|\boldsymbol{W}\boldsymbol{x}^k-\boldsymbol{p}\|^2+2\boldsymbol{g}_{1,k}^{\mathrm{T}}(\boldsymbol{x}-\boldsymbol{x}^k)+\frac{1}{\tau_1}\|\boldsymbol{x}-\boldsymbol{x}^k\|^2,\tag{21}$$

$$\|\nabla\boldsymbol{x}\|^2\approx\|\nabla\boldsymbol{x}^k\|^2+2\boldsymbol{g}_{2,k}^{\mathrm{T}}(\boldsymbol{x}-\boldsymbol{x}^k)+\frac{1}{\tau_2}\|\boldsymbol{x}-\boldsymbol{x}^k\|^2,\tag{22}$$

式中 $\boldsymbol{g}_{1,k}=\boldsymbol{W}^{\mathrm{T}}(\boldsymbol{W}\boldsymbol{x}^k-\boldsymbol{p})$，$\boldsymbol{g}_{2,k}=\nabla^{\mathrm{T}}\nabla\boldsymbol{x}^k$，$\tau_1>0$，$\tau_2>0$。将其代入(14)式中，并计算线性化后的公式导数为 0，可得到该子问题的迭代求解公式为

$$\boldsymbol{x}^{k+1}=\boldsymbol{x}^k-(\lambda\boldsymbol{g}_{1,k}+\gamma\boldsymbol{g}_{2,k}+\nabla^{\mathrm{T}}\omega^k)\Big/\left(\frac{\lambda}{\tau_1}+\frac{\gamma}{\tau_2}\right)。\tag{23}$$

对于(16)式的求解，利用 shrinkage 算子可直接得到其解析解[17]为

$$\boldsymbol{z}^{k+1}=\max\left(\left|\frac{2\omega_g^{k+1}-\omega^k}{\gamma}\right|-\frac{1}{\gamma},0\right)\cdot\operatorname{sgn}\left(\frac{2\omega_g^{k+1}-\omega^k}{\gamma}\right)。\tag{24}$$

对于并行计算系统，还需要对更新公式中的变量进行划分，利用多个节点来异步并行更新变量中的元素，即对于单个子节点仅需计算相关变量中的部分元素，然后利用计算后的元素更新主节点中变量相应的元素即可。迭代公式中涉及到的更新变量有 $\boldsymbol{x}$、$\boldsymbol{z}$、ω_g、ω_f 和 ω，以及不变的 $\boldsymbol{W}$ 和 $\boldsymbol{p}$。在更新 $\boldsymbol{x}$ 时最为耗时，而其余 4 个变量的更新均为简单的矩阵运算，所需时间相比 $\boldsymbol{x}$ 的更新可以忽略。为减少整体的数据传输次数，仅在 $\boldsymbol{x}$ 更新后对主节点内的存储数据进行更新，而其余 4 个变量在全部计算完毕后再更新主节点内的数据。每个节点更新变量时，任意选取其中一个子块进行迭代更新。

Async-ADTVM 算法在子节点对第 i 个子块中变量的更新流程为：1)读取主节点所存变量 $\boldsymbol{x}$、$\boldsymbol{z}$、ω_g、ω_f 和 ω；2)更新变量 $\boldsymbol{x}$ 的第 i 块，$\boldsymbol{x}_i^{k+1}=\boldsymbol{x}_i^k-\{\lambda[\boldsymbol{W}_i^{\mathrm{T}}(\boldsymbol{W}\boldsymbol{x}^k-\boldsymbol{p})]+\gamma\nabla_i^{\mathrm{T}}(\nabla\boldsymbol{x}^k)+\nabla_i^{\mathrm{T}}\omega^k\}/(\lambda/\tau_1+\gamma/\tau_2)$；3)利用计算后的 $\boldsymbol{x}_i^{k+1}$ 更新主节点中的变量 $\boldsymbol{x}$；4)更新变量 ω_g 的第 i 块，$\omega_{g,i}^{k+1}=\omega_i^k+\gamma\nabla_i\boldsymbol{x}^{k+1}$；5)更新 $\omega_{g,i}^{k+1}=\omega_i^k+\gamma\nabla_i\boldsymbol{x}^{k+1}$，$\boldsymbol{z}_i^{k+1}=\max\left(\left|\frac{2\omega_{g,i}^{k+1}-\omega_i^k}{\gamma}\right|-\frac{1}{\gamma},0\right)\times\operatorname{sgn}\left(\frac{2\omega_{g,i}^{k+1}-\omega_i^k}{\gamma}\right)$；6)更新变量 ω_f 的第 i 块，$\omega_{f,i}^{k+1}=2\omega_{g,i}^{k+1}-\omega_i^k-\gamma\boldsymbol{z}_i^{k+1}$；7)更新变量 ω 的第 i 块，$\omega_i^{k+1}=\omega_i^k+\alpha(\omega_{f,i}^{k+1}-\omega_{g,i}^{k+1})$；8)利用计算后的变量 $\omega_{g,i}^{k+1}$、$\boldsymbol{z}_i^{k+1}$、$\omega_{f,i}^{k+1}$、ω_i^{k+1} 更新主节点中相应的变量；9)返回步骤 1)进行下一轮迭代。

2.3　基于消息传递接口(MPI)的 GPU 机群优化实现

由于上述迭代公式中涉及 $\boldsymbol{W}$ 的相关计算，虽然已经对其进行了划分，但划分后的矩阵仍然非常巨大，耗时较长，影响整体的计算速度。锥形束 CT 系统的投影矩阵过于庞大，导致已无法对其进行存储并计算。因为其计算相当于正、反投影运算，所以可以采用计算代替存储解决该问题，同时利用 GPU 对其进行加速计算，以提高计算效率。对于该异步并行重建算法的优化实现，考虑到加速平台的可扩展性，采用 GPU 机群对其进行测试。

采用主控机节点、计算节点、高速交换机、高速互联网络等主要基础设施组成 GPU 机群系统，如图 2 所示。主控机节点(简称主节点)主要用于存储整体数据，并控制数据的传输与程序的停止。而计算节点(简称子节点)均配置有专用计算级的 GPU 显卡，主要用于变量的迭代更新计算。所有节点均通过高速交换机以高速互联网络相互连接进行数据的传输。

对于 GPU 机群系统，其中所需要考虑的主要问题在于节点内的计算过程，以及节点间的数据传输过程。GPU 机群系统中通常涉及到节点与节点之间的数据传输，以及节点内中央处理器(CPU)与 GPU 之间的数据传输。对于节点之间的数据通信，采用常用的 MPI 进行编程实现；对于节点内部 CPU 与 GPU 之间的数据通信，采用统一设备构架(CUDA)技术进行编程实现。

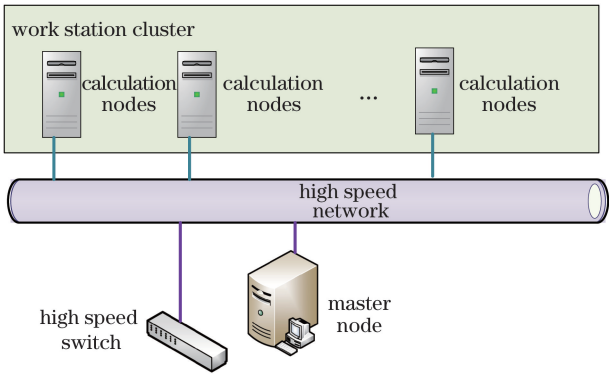

图 2　GPU 机群的组成结构

Fig. 2　Structures of GPU cluster

通用的并行程序设计主要分为两大类：任务并行和数据并行。传统迭代重建算法为明确的串行算法，即轮与轮之间依次计算，轮中子问题的求解也依次进行。所提出的异步并行迭代重建算法在理论上进行了并行设计，能够很好地与并行计算系统相契合。节点之间的并行计算为任务并行计算，而节点内的 GPU 加速计算为数据并行计算。对于一个机群系统，数据的传输同样是一个重要的考虑内容，这里从节点间的数据传输和节点内的加速计算两个方面进行讨论。

1）数据的划分与节点间的数据传输

对于所提出的 Async-ADTVM 算法，需要将整体数据进行划分，并在子节点中对划分后的数据分别进行迭代更新。理论上数据划分的越多，算法的收敛性越好，但过多的数据划分会影响到整体的计算效率。因此，对于拥有 m 个计算节点的并行计算系统，采用对数据进行 m 等分的策略，最大化计算节点利用率的同时，尽可能提高系统的计算效率。同时，由于 $\boldsymbol{p}$ 不变，可在计算前将其分别存储在各子节点内，从而避免计算过程中数据的传输。

通过分析迭代公式可以看出，对于子节点中局部数据 $\boldsymbol{z}_i$、$\boldsymbol{\omega}_{g,i}$、$\boldsymbol{\omega}_{f,i}$、$\boldsymbol{\omega}_i$ 的更新，仅需要对相关的局部数据进行计算即可。而对于变量 $\boldsymbol{x}_i$ 的更新，涉及到正、反投影过程，并且其中正投影过程为对整体数据 $\boldsymbol{x}$ 的计算，而反投影过程为对部分数据的计算。因此，采用对三维重建体数据 $\boldsymbol{x}$ 沿垂直方向 m 等分的策略，从而简化反投影过程中的计算。

同时可以看出，对于任意子节点，在每一轮迭代更新过程中涉及的数据传输仅有三部分：第一部分为迭代开始时从主节点内获取此时刻所有节点更新后的整体数据；第二部分为子节点更新 $\boldsymbol{x}_i$ 后向主节点传输数据以更新整体数据 $\boldsymbol{x}$；第三部分为其余4 个变量 $\boldsymbol{z}_i$、$\boldsymbol{\omega}_{g,i}$、$\boldsymbol{\omega}_{f,i}$、$\boldsymbol{\omega}_i$ 更新结束后向主节点进行数据的传输以更新整体数据。三部分的数据传输均采用 MPI 技术中点对点通信方式进行。通过分析可以发现，在更新子节点内数据 $\boldsymbol{x}_i$ 时需要用到主节点中存储的整体数据 $\boldsymbol{x}$ 和 ω，而在更新其他 4 个变量时仅需本地存储的局部数据即可。对于第一部分的数据传输，仅需在数据 $\boldsymbol{x}$ 和 ω 传输完毕后即可进行对 $\boldsymbol{x}_i$ 的更新计算。因此，对 $\boldsymbol{x}$ 和 ω 采用 MPI 技术中的同步通信方式，而其余变量采用异步通信方式，进而节约部分数据传输时间。对于第二部分的数据传输，后续变量的更新计算（算法 1 中的第 4～7 步）与 $\boldsymbol{x}_i$ 无数据冲突，因此对 $\boldsymbol{x}_i$ 采用异步通信方式，以提高通信的效率。

2）节点内的加速计算

针对节点内部的加速计算，主要涉及到变量 $\boldsymbol{x}_i$ 更新中公式 $\boldsymbol{W}_i^{\mathrm{T}}(\boldsymbol{W}\boldsymbol{x}^k-\boldsymbol{p})$ 的计算，该部分需要利用 GPU 进行加速。因此，可将 $\boldsymbol{x}_i$ 更新公式进行如下转换

$$\boldsymbol{x}_i^{k+1}=\left\{\boldsymbol{x}_i^k-\left[\gamma\nabla_i^{\mathrm{T}}(\nabla\boldsymbol{x}^k)+\nabla_i^{\mathrm{T}}\boldsymbol{\omega}^k\right]\Big/\left(\frac{\lambda}{\tau_1}+\frac{\gamma}{\tau_2}\right)\right\}-\left\{\lambda\left[\boldsymbol{W}_i^{\mathrm{T}}(\boldsymbol{W}\boldsymbol{x}^k-\boldsymbol{p})\right]\Big/\left(\frac{\lambda}{\tau_1}+\frac{\gamma}{\tau_2}\right)\right\},\tag{25}$$

式中等号右侧的左半部分均为简单计算操作，仅需在 CPU 端进行；而右半部分为涉及 $\boldsymbol{W}$ 的复杂计算操作，需要利用 GPU 进行加速。通常 GPU 加速计算的时间远大于其余 CPU 计算的时间。随着 GPU 技术的发展，现有 GPU 可实现对不相关的数据进行 CPU 与 GPU 异步并行计算技术。因此，利用该技术对变量 $\boldsymbol{x}_i$ 更新中两部分并行计算，可节约其 CPU 计算耗时，从而提高计算效率。

而对于涉及 $\boldsymbol{W}$ 或其子块 $\boldsymbol{W}_i$ 的计算，如 $\boldsymbol{W}\boldsymbol{x}$ 与 $\boldsymbol{W}^{\mathrm{T}}\boldsymbol{x}$ 的计算，可利用其每个元素计算的可并行性，采用 GPU 进行加速计算。在正投影过程中，投影中各元素的值为该角度下经过该点射线穿过体素的长度与体素值对应乘积的和，采用此方式计算每个投影点的值不会产生数据读写冲突。而对于反投影计算，采用体素驱动的方式，即每个体素的数值为经过该点的所有射线对应的投影值与相应射线穿过该体素长度的乘积的和，采用此方式计算反投影时每个体素的值不会产生数据读写冲突。因此，针对正、反投影的计算，分别采用如上计算方式作为 GPU 核函数进行加速计算。

3　实验结果与分析

为验证本章所提的 Async-ADTVM 算法在 GPU

机群上的计算性能，首先搭建了一个小型 GPU 机群。GPU 机群包含 4 个 GPU 计算服务器、1 个管理服务器、千兆交换机以及千兆以太网。所采用的软件配置为 Visual Studio 2012、MPICH2-1.41 以及 CUDA7.0，其中最主要的 GPU 计算服务器的硬件配置如表 1 所示。

表 1　重建加速的实验平台参数

Table 1　Parameters of reconstruction acceleration experimental platform

Item	Computing server model	
	GPU computing server Ⅰ	GPU computing server Ⅱ
CPU	Intel IvyBridgeE5-2630v2 (2.6 GHz)	Intel IvyBridgeE5-2630v2 (2.6 GHz)
GPU	Tesla K20 (5 G)×1	Tesla K20 (5 G)×1 Tesla K40 (12 G)×1
RAM	32 G	32 G
Operating system	Windows 7, 64 bit	Windows 7, 64 bit
Number	2	2

为测试算法的正确性并对比计算效率，共设计 4 组实验方案：1)利用一台 GPU 计算服务器Ⅰ进行算法求解计算，但仅利用 CPU 进行计算；2)利用一台 GPU 计算服务器Ⅰ进行算法单 GPU 加速求解；3)利用 4 台 GPU 计算服务器分别进行同步并行重建与异步并行重建，并且所有计算服务器均只利用 K20 显卡进行加速计算；4)利用 4 台 GPU 计算服务器分别进行同步并行重建与异步并行重建，同时 2 台 GPU 计算服务器Ⅰ采用 K20 显卡，2 台 GPU 计算服务器Ⅱ采用 K40 显卡。所有实验采用相同的系统扫描参数，具体参数如表 2 所示。体模数据采用标准三维 Shepp-Logan 体模进行测试，计算中所有数据格式均采用单精度浮点类型。

表 2　锥束 CT 系统扫描参数

Table 2　Scanning parameters of cone-beam CT system

Item	Parameter
Scanning angle range /(°)	0-360
Number of probes	1024×1024
Distance from source to rotation center /mm	600
Distance from source to detector /mm	1200
Projection number	60
Total of projection data	1024×1024×60
Reconstruction scale	512×512×512
Pixel size /mm×mm	0.25×0.25

利用重建图像质量与原始体模之间的均方根误差(RMSE)来评价重建结果的质量，并利用加速比来评价 GPU 机群系统对算法的加速效果。以第一组 CPU 计算以及第二组单 GPU 加速的实验结果分别作为评价的基准，通过计算其他组实验的加速比来判断其计算效率。加速比定义为

$$R_{\text{speed}}=\frac{t_{\text{s}}}{t_{\text{p}}},\tag{26}$$

式中 t_{s} 表示并行的计算时间，t_{p} 表示作为基准的计算时间。

4 组实验在第 2000 轮迭代时中间层的重建结果展示如图 3 所示，其对应的 RMSE 以及平均每轮迭代时间如表 3 所示。实验 3、4 中的异步并行计算重建结果是指计算最快节点到达第 2000 轮时整体的重建结果。从结果可以看出 4 组实验结果均可取得良好的重建质量。对于实验 3、4 中的同步并行计算策略，可取得同 CPU 计算和单 GPU 加速计算相同的重建结果。对于实验 3、4 中的异步并行计算策略，可取得更优的重建结果。其主要原因是异步加速计算中计算性能较优的节点计算的速度较快，可更快迭代出较优的结果，而该结果会被其余较慢节点使用，从而优化计算较慢节点的重建结果。

相比于传统 CPU 计算，算法采用 GPU 加速策略可取得 50 倍以上的加速比。对于多 GPU 机群系统的加速效果，实验 3、4 中同步并行计算的加速效果并无显著差异。因为同步并行计算仅受性能最差节点计算速度影响，而 2 组实验中性能最差的 GPU 均采用 K20 显卡，因此并不存在较大的整体计算性能差异。实验 3、4 中异步并行计算的加速比因各 GPU 性能的不同而存在一定的差异。对比实验 3 中同步并行计算和异步并行计算的计算时间，由于所有节点均采用相同的 GPU 进行加速计算，异步并行计算的效果比同步并行计算略有提高。通过对比实验 4 中两种并行计算策略的计算时间，发现异步并行计算在 GPU 机群系统中最差节点的计算性能相同且各节点存在性能差异时可取得更优的加速效果。

表 3　4 组实验在第 2000 轮迭代时结果的 RMSE 以及平均每轮迭代时间

Table 3　RMSE of results at iteration of 2000 times and average iteration time of each iteration for four experiments

Result	Experiment 1	Experiment 2	Experiment 3		Experiment 4	
			Synchronous parallel	Asynchronous parallel	Synchronous parallel	Asynchronous parallel
RMSE	6.78×10^{-4}	6.78×10^{-4}	6.78×10^{-4}	5.69×10^{-4}	6.78×10^{-4}	5.34×10^{-4}
Average single time /s	6842.00	130.50	53.93	51.49	53.26	45.67
R_{speed} (compared with CPU)	—	52.4	126.9	132.9	128.5	149.8
R_{speed} (compared with single GPU)	—	—	2.42	2.53	2.45	2.86

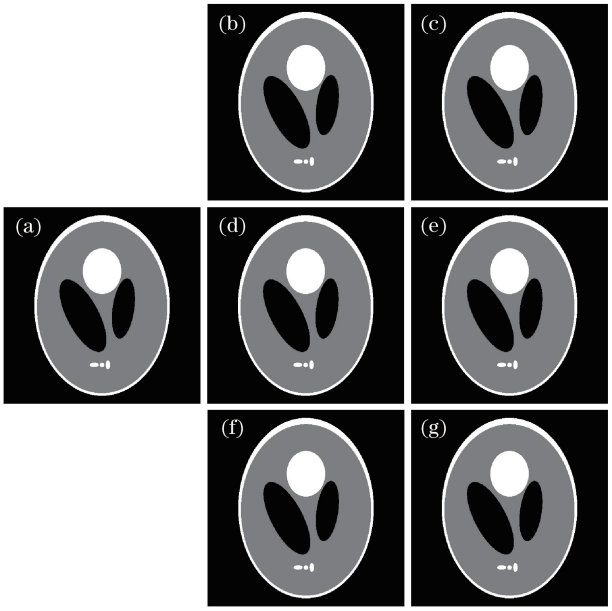

图 3　原始图像及 4 组实验在第 2000 轮迭代时中间层的重建结果。(a)原始图像;(b)实验 1 的结果;(c)实验 2 的结果;(d)实验 3 中同步并行结果;(e)实验 3 中非同步并行结果;(f)实验 4 中同步并行结果;(g)实验 4 中非同步并行结果

Fig. 3　Original image and reconstruction results of middle slices at iteration of 2000 times. (a) Original image; (b) result of experiment 1; (c) result of experiment 2; (d) result of sync-parallel computing in experiment 3; (e) result of async-parallel computing in experiment 3; (f) result of sync-parallel computing in experiment 4; (g) result of async-parallel computing in experiment 4

4　结　　论

图像重建速度是限制迭代类重建算法在实际中应用的一个重要因素。针对该问题,提出了 Async-ADTVM 算法。该算法在异步并行计算框架的基础上,针对最常用的 TV 最小化重建模型,通过将其转化为不动点迭代问题,并利用异步 ADM 进行求解。相比于传统的同步并行计算方法,该算法在保证收敛性的同时,减少了并行计算系统中各节点性能差异所带来的影响。同时,利用 GPU 机群系统对该异步并行算法进行优化。分析迭代过程中的数据通信以及数据并行性,并分别采用 MPI 和 CUDA 技术进行实现。实验结果表明:该算法可取得同常规 TV 最小化迭代重建算法相同的重建精度,同时可大幅度提高系统的计算效率;相比于多 GPU 的同步并行计算模式,该算法在 GPU 机群上具有更快的计算速度以及收敛速度。

参　考　文　献

[1] Buzug T M. Computed tomography: From photon statistics to modern cone-beam CT[M]. Berlin Heidelberg: Springer-Verlag, 2008.

[2] Feldkamp L A, Davis L C, Kress J W. Practical cone-beam algorithm[J]. Journal of the Optical Society of America A, 1984, 1(6): 612-619.

[3] Liu Y, Liang Z R, Ma J H, *et al*. Total variation-stokes strategy for sparse-view X-ray CT image reconstruction[J]. IEEE Transactions on Medical Imaging, 2014, 33(3): 749-763.

[4] Xu Q, Yu H Y, Mou X Q, *et al*. Low-dose X-ray CT reconstruction via dictionary learning[J]. IEEE Transactions on Medical Imaging, 2012, 31(9): 1682-1697.

[5] Tuy H K. An inversion formula for cone-beam reconstruction[J]. SIAM Journal on Applied Mathematics, 1983, 43(3): 546-552.

[6] Smith B D. Image reconstruction from cone-beam projections: Necessary and sufficient conditions and reconstruction methods[J]. IEEE Transactions on Medical Imaging, 1985, 4(1): 14-25.

[7] Natterer F, Wubbeling F. Mathematical methods in image reconstruction[M]. Philadelphia: Society for Industrial and Applied Mathematics, 2001.

[8] Gordon R, Bender R, Herman G T. Algebraic reconstruction techniques (ART) for three-dimensional electron microscopy and X-ray photography[J]. Journal

of Theoretical Biology, 1970, 29(3): 471-481.

[9] Donoho D L. Compressed sensing[J]. IEEE Transactions on Information Theory, 2006, 52(4): 1289-1306.

[10] Romberg J, Tao T. Exact signal reconstruction from highly incomplete frequency information[J]. IEEE Transactions on Information Theory, 2006, 52(2): 489-509.

[11] Sidky E Y, Kao C M, Pan X C. Accurate image reconstruction from few-views and limited-angle data in divergent-beam CT[J]. Journal of X-ray Science and Technology, 2006, 14(2): 119-139.

[12] Sidky E Y, Pan X C. Image reconstruction in circular cone-beam computed tomography by constrained, total-variation minimization[J]. Physics in Medicine & Biology, 2008, 53(17): 4777-4807.

[13] Ma J M, Zhang J Q, Song G Z, *et al*. Total variation constrained iterative filtered backprojection CT reconstruction method[J]. Acta Optica Sinica, 2015, 32(2): 0234002.
马继明, 张建奇, 宋顾周, 等. 全变分约束迭代滤波反投影 CT 重建[J]. 光学学报, 2015, 35(2): 0234002.

[14] Jia X, Lou Y F, Li R J, *et al*. GPU-based fast cone beam CT reconstruction from undersampled and noisy projection data via total variation[J]. Medical Physics, 2010, 37(4): 1757-1760.

[15] Liu R, Luo Y, Yu H Y. GPU-based acceleration for interior tomography[J]. IEEE Access, 2014, 2: 757-770.

[16] Li B, Tian Z, Zhou L X, *et al*. Employing a novel consensus optimization strategy to achieve iterative cone beam CT reconstruction on a multi-GPU platform[J]. Medical Physics, 2016, 40(6): 3344-3345.

[17] Wang X Y, Yan H, Cervino L, *et al*. Iterative cone beam CT reconstruction on a multi-GPU platform[J]. Medical Physics, 2013, 40(6): 543.

[18] Fehringer A, Lasser T, Zanette I, *et al*. A versatile tomographic forward-and back-projection approach on multi-GPUs[C]. SPIE, 2014, 9034: 90344F.

[19] Peng Z M, Xu Y Y, Yan M, *et al*. ARock: An algorithmic framework for asynchronous parallel coordinate updates[J]. SIAM Journal on Scientific Computing, 2016, 38(5): A2851-A2879.