doi:10.3969/j.issn.1000-436x.2012.z2.010

以实际算法为例评估 MapReduce 在石油勘探中的应用

赵长海[1]，晏海华[1]，刘晓朋[1]，熊登[2]，史晓华[1]
（1.北京航空航天大学 计算机学院，北京 100191；
2.中国石油东方地球物理公司 物探技术研究中心，河北 涿州 072751）

摘 要：石油勘探领域需要处理海量的地震数据，以获取地下构造用以发现和定位油藏。为评估云计算编程模型 MapReduce 对于石油勘探领域应用算法的适用性，设计并实现了基于 MapReduce 的三维 Fresnel 层析成像算法，实验发现 MapReduce 版本的性能比 MPI 版本慢 3 倍，而且对 MapReduce 作业调优的难度相当大。为了拓展 MapReduce 在石油勘探领域高性能计算领域的应用，需要在支持线程级并行、灵活性和提升 I/O 可扩展性 3 个方面进行改进，并提出了研究方法和技术路线。

关键词：石油勘探；MapReduce；Fresnel 层析成像；高性能计算；地震数据
中图分类号：TP311 **文献标识码**：A **文章编号**：1000-436X(2012)Z2-0081-09

Evaluating MapReduce for seismic data processing using a practical application

ZHAO Chang-hai[1], YAN Hai-hua[1], LIU Xiao-peng[1], XIONG Deng[2],SHI Xiao-hua[1]
(1. School of Computer Science and Engineering, BeiHang University, Beijing 100191, China;
2. GeoPhysical Technique Research Center, BGP, CNPC, Zhuozhou 072751, China)

Abstract: Huge amounts of seismic data undergo complex iterative processing in the oil industry to get knowledge of the earth's subsurface structure to detect where oil can be found and recovered. To evaluate the suitability of MapReduce for seismic processing algorithms, the algorithm design and implementation of Fresnel tomography on Hadoop MapReduce was described. Experiments demonstrate that MapReduce is approximately 3 times slower than MPI, and tuning the performance of MapReduce is really hard. To expand its applicability to high performance computing for oil industry, MapReduce should be improved in the flexibility and provide the opportunity to exploit fine-grained thread-level parallelism. Finally, research ideas to achieve these objectives were presented.

Key words: oil exploration; MapReduce; Fresnel tomography; high performance computing; seismic data

1 引言

在石油工业领域，地球勘探的主要目的是获取地下构造用以发现和定位油藏，为达到此目的，需要进行大量的地震勘测获取地下原始信息，这些信息经过复杂的迭代处理，从中抽取地质模型，然后由地球物理专家解释并推断油藏信息。地震勘探通过在地表激发地震波，同时利用高度敏感的检波器接收地下反射波，当前典型的地震勘探项目的数据量有 50~100TB，未来数年内将会达到 PB 级的规模[1,2]。由于原始数据信噪比非常低，而且数据量非常大，所以地震数据处理对计算能力要求非常高，

收稿日期：2012-06-29
基金项目：国家自然科学基金资助项目(61073010, 61272166)
Foundation Item: The National Natural Science Foundation of China(61073010, 61272166)

一直属于高性能计算的重要应用领域。

地震数据处理流程分为2个阶段。第一阶段是应用信号处理算法提高信噪比，目前有几百种算法可应用于这个阶段，地球物理专家根据地面和地下地质特点和经验选取合适的算法；第二阶段是地震偏移成像，将地震波能量归位到其空间的真实位置，获取地下的真实构造图像，成像算法的计算量非常大。由于速度场是未知的，上述阶段需要迭代执行，不断改进速度模型，以获取更加精确的地下图像。

高性能计算系统的并行规模和待处理的地震数据量持续增大，要求石油勘探领域的并行应用必须具备很高可扩展性，调度几百至上千计算节点，而用MPI编写这类高度可扩展的并行程序，难度非常高，目前MPI不支持容错，一旦出现节点故障，整个并行程序需要重启，实际应用中，需要在应用层实现检查点提高程序的可靠性，但是这不仅会增加编程难度，而且引入了巨大的性能开销，特别是存储I/O方面的开销[3]。

MapReduce[4]作为云计算最为重要的并行与分布式编程模型，具备易编程、自动并行、可容错与可扩展等优点。但是该模型的上述优势是以牺牲灵活性为代价的，限制了它的应用范围，此外MapReduce运行时系统中的诸多优化策略，例如数据局部性优化，都是建立在分布式文件系统基础上，而高性能计算系统一般采用计算与存储分离的体系架构，存储系统由多个I/O服务器组成，并利用Lustre[5]、GPFS[6]等并行文件系统将数据条带化成数据块存储在多个I/O节点，从而提高系统的聚合I/O带宽和并发度。已经有一些研究探讨将MapReduce应用于高性能计算[7~10]，这些研究都是选取若干个孤立的科学应用进行分析，没有考虑整个领域的背景和应用特征。

本文选用实际生产中的三维Fresnel层析成像算法为例，介绍基于MapReduce的并行算法设计与实现，实验分析影响MapReduce作业性能与扩展性的因素，并与MPI实现进行对比分析；探讨MapReduce应用于石油勘探高性能计算面临的问题以及改进思路。

2 三维Fresnel层析成像原理与并行算法

图1描述了地震数据的采集过程。在地表激发地震波，在向地下传播时，遇有介质性质不同的岩层分界面，地震波发生反射与折射，在地表用检波器接收这种地震波。检波器接收到的一次激发所产生的地震波称为地震道(trace)，是地震数据处理的基本单位，从几KB到几十KB不等，地震道是一种结构化的数据，一般用SEG-Y[11]数据格式存储。检波器最先接收到的波称初至波，是三维Fresnel层析成像的输入数据。

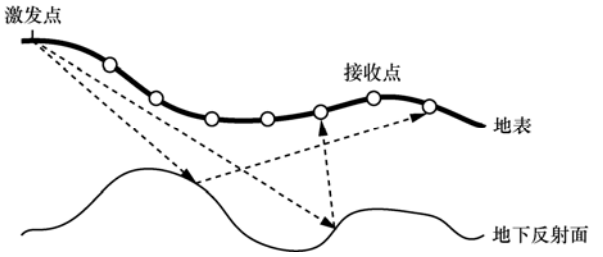


图1 地震数据采集

层析成像类似医学的CT成像技术，利用地震初至波旅行时，重建地下介质速度模型，从而揭示其地质构造、岩性分布等地质特征。Fresnel层析成像相比其他层析成像方法，更符合波传播规律，能够提高成像精度和分辨率。按照Fresnel带的定义，对于频率为f，周期为T的地震波，激发点S和接收点R之间的第一Fresnel带为满足下面条件的所有点F的集合[12]。

$$\left|t(F,A)+t(F,B)-t(A,B)\right| \leqslant T/2 \tag{1}$$

F点的集合在空间内形成一个三维Fresnel带，在均匀介质中为以S和R作为2个焦点的扁椭球体，射线处于三维Fresnel带的轴线。理论研究表明，影响地震波传播的主要区域集中于射线邻域的第一Fresnel带[13]。

为便于计算机模拟，首先将近地表区域用长方体网格剖分，离散化地下速度场，每个单元网格内的慢度（速度的倒数）是均匀的，那么当单元网格大小趋于零时，慢度值将客观反映地下的速度场信息。网点的疏密程度，就决定了射线追踪的精度、计算量和存储量。层析成像核心计算分为正演和反演2部分。

正演计算是确定激发点到接收点的射线路径。从激发点到接收点存在无数条路径，其中的最短路径，即旅行时最小的路径，按照Fermat原理，就可近似成射线路径。为得到射线路径，需要利用迎风格式有限差分法求解程函方程(2)，得到相邻网格单元间的旅行时，用堆排序算法选择最小旅行时点，

逐步得到激发点到每个网格单元的最小旅行时。

$$\left(\frac{\partial t}{\partial x}\right)^2+\left(\frac{\partial t}{\partial y}\right)^2+\left(\frac{\partial t}{\partial z}\right)^2=s_{x,y,z}^2 \tag{2}$$

其中，t 为地震波从激发点传播到空间任一点(x, y, z)的旅行时，$s_{x,y,z}$ 是空间任意点(x, y, z)的慢度。

反演是根据正演出来的射线路径，求解慢度增量，进而更新慢度场（或速度模型）。首先对网格单元统一编号，由式(1)确定一对激发点和接收点对应的三维 Fresnel 带，设第 i 条三维 Fresnel 带轴线在网格第 j 列单元中的长度为 l_{ij}，对应的 Fresnel 带包含 K 个单元，总体积为 V，其中第 k 个单元在 Fresnel 带内的体积为 v_k，通过求解式(3)的大型线程方程组[12]，得到慢度增量Δs_k。

$$\sum_{j=1}^{J}\sum_{k=1}^{K}\omega_k\frac{v_k}{V}l_{ij}\Delta_{s_k}=\Delta_{t_i} \tag{3}$$

其中，Δt_i 是第 i 对激发点和接收点的旅行时残差（初至波旅行时与当前模型计算旅行时之差）；Δs_k 是第 k 个单元的慢度增量；ω_k 是反映各网格单元能流密度大小的权系数，在 Fresnel 带轴线穿过的单元权系数最大，在 Fresnel 带边界上的单元权系数为零，且 $\sum_{k=1}^{K}\omega_k=1$。

最终的速度模型通过正演和反演多次迭代获得，如图 2 所示，迭代过程中不断修正速度模型，新的速度模型作为下一次迭代的输入，直到旅行时残差Δ_t达到一定精度要求为止。

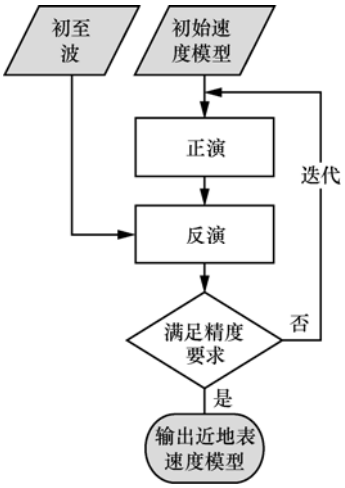


图 2　Fresnel 层析成像串行执行流程

由于从激发点到所有网格单元的旅行时计算是相互独立的，使得在反演阶段不同激发点和接收点对应的 Fresnel 带的计算也是相互独立的，根据旅行时残差计算网格单元的慢度增量，也是独立的。因此，每次迭代，可以并行计算每一对激发点和接收点对应的正演和反演，最后归约所有网格单元的慢度增量。该并行算法的形式化描述如图 3 所示，可以看出，该并行算法适合 MapReduce 表达。

Input : first-arrival waves and the initial velocity model
Output : near-surface velocity field
NS = total number of sources
NR = total number of receivers
ε = precision required
While $RMS_M > \varepsilon$ do
  For $i = 1$ to NS do in parallel
    For $j = 1$ to NR do in parallel
      Compute ray path between i and j in forward stage
      Compute $\Delta_{t(i,j)}$ in forward stage
      Compute Fresnel zone of i and j in inverse stage
      Compute $\Delta_{s(i,j)}$ in inverse stage
    End parallel
  End parallel
  //reduce Δ_t and Δ_s
  $RMS_{\Delta_t}=\sqrt{\sum_{i=1}^{NS}\sum_{j=1}^{NR}\frac{\Delta_{t(i,j)}^2}{NS\times NR}}$
  $\Delta_s=\sum_{i=1}^{NS}\sum_{j=1}^{NR}\frac{\Delta_{s(i,j)}}{NS\times NR}$
  Update velocity field using Δ_s
End while

图 3　Fresnel 层析成像并行算法

3　基于 MapReduce 的实现

Fresnel 层析成像算法需要多次迭代来逼近正确解，每一次迭代的并行算法使用 MapReduce 来表达，通过串联多个 MapReduce 作业的方式实现多次迭代，每次迭代输出的经修正的速度模型作为下一次迭代的初始速度模型。

3.1　Map 与 Reduce 实现

根据图 2 的并行算法，一对激发点和接收点是并行任务分解的最小单位。实际生产中，勘探工区的激发点与接收点对数可达数千万，每一对激发点和接收点的正反演计算在 100ms 量级左右，若以激发点为单位分配 Map 任务，实际数据中的一个激发点对应的接收点通常有数千个，一个 Map 任务处理数千个 Fresnel 带，可以保证较大的并行粒度，从而降低 Map 任务启动和初始化开销所占的比例。

Map 的实现逻辑如式(4)所示，其中 b 为激发点编号，W_b 为该激发点对应的初至波；k 为三维网格的网格单元编号，$\Delta s_{(b)}$是网格单元 k 对应的慢度增量。Map 任务以激发点 b 对应的初至波作为输入，对该激发点与所有接收点组成的菲涅尔带进行正反演计算，输出每一个网格单元的慢度增量。reduce

任务的实现逻辑如式(5)所示，将每一个网格单元的慢度增量累加，得到每一个网格单元的平均慢度增量。

$$\text{map}(b,W_b)\rightarrow list(k,\Delta_{s(b)}) \tag{4}$$

$$\text{reduce}(k,list(\Delta_{s(b)}))\rightarrow(k,\Delta_s) \tag{5}$$

Map 中得到慢度增量之后，根据图 2，还要计算旅行时残差的均方根误差，为了与慢度增量相区分，将输出 *key* 设置为特定的负数，实现逻辑如式(6)所示，其中$\Delta_{t(b)}$是激发点 *b* 对应的旅行时残差平方和，reduce 中累加所有的旅行时残差，如式(7)所示，然后计算均方根误差，作为是否进行下一次迭代的依据。

$$\text{map}(b,W_b)\rightarrow(-1,\Delta_{t(b)}) \tag{6}$$

$$\text{reduce}(-1,\Delta_{t(b)})\rightarrow RMS(\Delta_t) \tag{7}$$

3.2 数据分块及作业参数

以激发点为单位，对初至波数据进行分块，每一个分块内的激发点数目，由式（8）得到。

$$N_{\text{split}}=floor\left(\frac{N_{\text{source}}}{N_{\text{map}}}\right) \tag{8}$$

其中，N_{source}为初至文件记录的总激发点数目，N_{Map}为 Map 任务总数，采取向下取整平均分配，剩余部分平摊到所有的分块内，保证每个分块数据内的激发点数目尽量均匀。

Map 构造阶段，需要大量的参数，其中网格大小、旅行时残差精度、三维平滑半径等用基本数据类型表示的参数，可以采用 MapReduce 常规参数传递方式。但是作为参数的初始速度模型，大小一般在数百兆左右，较大的工区可以达到数吉比特，由于要被所有的 Map 读入内存，因此传递该参数要消耗巨大的 I/O 带宽，每一次迭代结束，该速度模型被更新，若采用 Hadoop 的 DistributedCache[14]优化机制，每次迭代，需要将该速度模型拷贝至所有作业节点，作业节点数越多拷贝时间越长，影响作业的可扩展性。考虑到高性能集群环境配备有并行文件系统，相比分布式文件系统，具有更高的并发吞吐率，速度模型存放至并行文件系统，供所有的 Map 并发读取。

3.3 作业的串联

每一次迭代结束，提取 MapReduce 作业结果文件中的网格编号和对应的慢度增量，修正初始速度模型。提取结果文件中的旅行时残差，与用户要求精度对比，若满足精度要求，则输出修正后的速度模型作为最终结果，否则，将该速度模型作为初始速度模型，启动下一轮次的 MapReduce 作业。这一串联步骤的计算量和 I/O 量较小，由独立的辅助程序完成。

4 实验与分析

MapReduce 版本层析成像程序输出的速度模型，通过与生产用的 MPI 版本结果反复对比，证明了其结果的正确性，本节实验不再讨论正确性，从如下几个方面进行讨论：

1) 每节点并发 Map 任务数目对性能的影响，并从应用特点分析影响并发 Map 数的因素；

2) Reduce 任务数目对程序性能的影响；

3) Map 任务粒度对性能和负载均衡的影响；

4) 与 MPI 对比，分析 MapReduce 的性能与可扩展性。

4.1 实验环境

实验在 256 节点的集群上进行评估，软硬件配置如表 1 所示，其中的本地磁盘读性能，使用命令：cat 大文件>/dev/null 度量得到。Hadoop 版本为 0.21.0，集群 2 个节点分别用作 NameNode 和 JobTracker，其他节点同时用作 DataNode 和 TaskTracker。层析程序的 Map 和 reduce 代码使用基于 Hadoop Pipe 的 C++接口实现。

表 1 集群硬件配置

系统	配置
nodes	256
CPUs in each node	2, Intel® Xeon® Processor X5650 2.67GHz
Threads in each CPU	6(Cores) × 2(Threads)
Memory Per Node	24GB
Network	Gigabit
Disks	1, SCSI disk, Read Speed: 171MB/sec
I/O servers	8
Software	Hadoop 0.21.0, Sun JDK 6, MPICH2 1.2
OS	Red Hat Enterprise Linux Server release 5.4 -64 bit

4.2 实验数据和参数

为保证实验结果的合理性，采用某工区约 100km^2 真实地震数据，总数据量是 6.7GB，激发点 29 368 个，接收点 183 521 个，共 163 249 930 个地

震道。离散模型正演反演网格数为 60 760 278 个。实际生产中，通常需要迭代 10 次以上才能得到满足精度要求的速度模型，每次迭代的时间基本相同，本文实验只取一次迭代的时间。

4.3　并发 Map 任务数的影响

本次实验使用 124 个节点，一个 Map 任务处理 4 个激发点，共 7 342 个 Map 任务，设定 64 个 reduce 任务。通过不断增加每个节点的最大 Map 数，获得最佳的性能。

作业执行时间由 3 个阶段组成：Map 阶段、Reduce 阶段、利用辅助程序更新速度模型，其中 Map 和 Reduce 阶段是重叠执行的。图 4 中分别是 Map 和作业执行时间与每个节点并发 Map 数的关系，从图中可以看出，Map 阶段占作业执行时间的比例最大，更新速度模型的时间基本恒定。

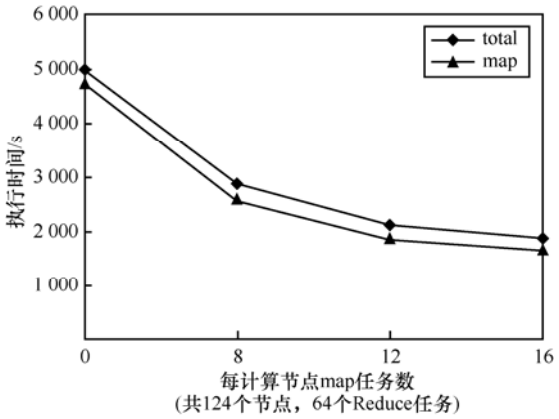


图 4　每节点最大 Map 任务数对性能的影响

作为计算密集型计算，Fresnel 层析作业随着节点内并发 Map 数的增加，执行时间不断下降，但是节点内并发 Map 数超过 16 以后，并发 Map 任务的内存开销超过了节点物理内存，执行时间急剧增加，不得不终止后续的实验。一个 Map 任务内存开销多达 1.4GB，主要用于存储速度模型、慢度增量、能流密度等与网格单元有关的数据。由于 Map 任务都是独立的动态进程，节点内无法利用共享内存降低内存使用量，导致部分处理器资源闲置，实验作业的 CPU 利用率最高只能达到 60%左右。

4.4　Reduce 任务数对性能的影响

reduce 任务规约的最重要的数据是每一个网格的慢度增量，网格编号和慢度增量分别使用 4byte 的整型和单精浮点型表示，由总的网格数，得出 Map 输出的<key, value>对最多可达到 60 760 278 个，共 464GB 的中间数据。为降低中间数据量，在实现中，Map 只输出 Fresnel 带内网格的慢度增量，本次实验，所有 Map 输出的中间数据量有 100GB。

如图 5 所示，随着 reduce 任务数的增加，Map 阶段时间保持正常的波动，没有被 Map 阶段重叠的 Reduce 阶段时间逐渐减少。仅设定 4 个 reduce 任务时，reduce 阶段成为作业的性能瓶颈，reduce 任务数达到 64 时，作业性能最优，继续增加 reduce 任务数，虽然 reduce 时间在降低，但输出的文件数量在增加，导致更新速度模型的辅助程序 I/O 开销增大。

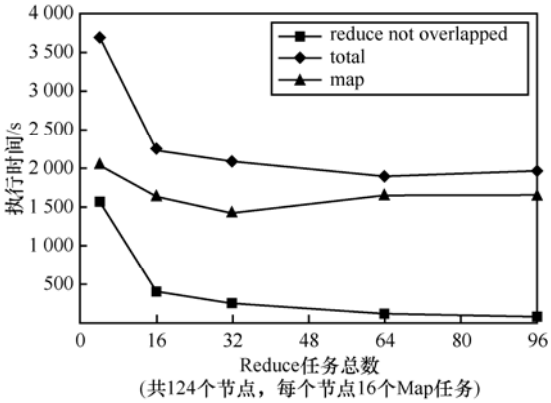


图 5　reduce 任务数对性能的影响

4.5　任务粒度对性能的影响

本次实验使用 124 个节点，每节点并发 16 个 Map 任务，共有 64 个 reduce 任务。Hadoop MapReduce 采用动态进程机制，每处理一个新的数据分块，都要创建一个 Map 进程，处理完毕 Map 进程消亡，这个过程引入了一定的开销；Fresnel 层析应用本身在 Map 的构造函数中，需要读入 234MB 的初始速度模型以及其他必要的参数，这个过程会带来更大的性能开销。适当增加 Map 任务粒度，可以降低上述开销所占的比重。

如图 6 所示，随着一个 Map 任务处理的激发点数目增加，作业执行并没有按照预期加快，反而越来越慢。经过分析发现，由于速度模型等参数存放在并行文件系统之上，并行文件系统的客户端缓存，提升了数据读取的速度，Map 的初始化在 1s 左右即可完成，同时，每一个激发点的处理时间在 94s 左右，从图 5 看出，Map 初始化所占比例已经很低，如果此时增加任务的粒度，总的 Map 任务数目减少，会导致负载的不均衡，进而影响性能。

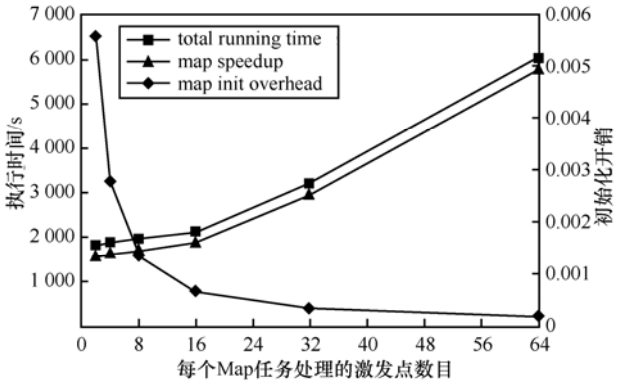


图 6 任务粒度对性能的影响

4.6 与 MPI 对比

如图 7 所示，MPI 版本的层析程序性能比 MapReduce 版本高出接近 2 倍，最重要的性能差异原因是 MPI 版本利用多线程技术，单个节点内共享速度模型等数据结构，降低内存开销，创建 24 个计算线程，CPU 利用率接近 100%，而 MapReduce 版本则由于内存限制，最多只能启动 16 个并发 Map 任务。此外，MPI 采用静态进程模型，没有反复创建和初始化的开销。Hadoop MapReduce 的核心代码基于 Java 语言，运行时开销比 MPI 也大得多。

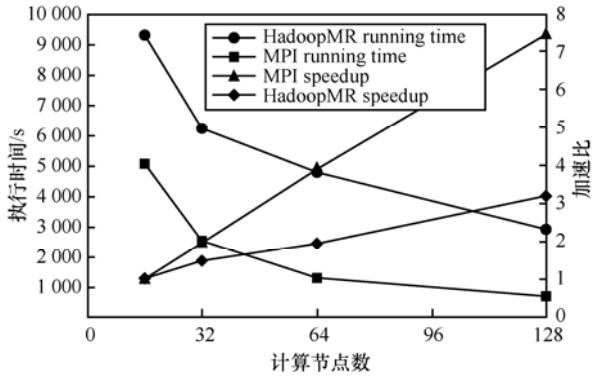


图 7 与 MPI 对比（加速比以 16 节点的执行时间为基准计算）

从图 7 的加速比曲线可以看出，2 个版本的加速比接近线性，表现出了良好的可扩展性。虽然图 6 展示的 MPI 版本性能与可扩展性优于 MapReduce，但是由于 MPI 不能自动容错，实验过程中，集群节点的软硬件故障，导致 MPI 程序重新执行，以及排除故障节点耽搁了很多时间，MPI 版本总的实验花费时间与 MapReduce 版本相当。

5 MapReduce 在石油勘探中的应用分析

MapReduce 编程模型最初提出主要是面向大数据集的处理，由于其简单而强大的数据处理接口和对大规模并行执行、容错及负载均衡等实现细节的隐藏，在机器学习、数据挖掘、数据分析等领域得到了广泛应用[15]。石油勘探领域同样面临大规模数据处理的问题，特别是 MapReduce 具备卓越的扩展能力，可以调度上千台计算机加速数据的处理，对石油勘探领域的众多应用算法具有很强的吸引力。在实际应用中，发现 MapReduce 存在一些问题，限制了 MapReduce 在石油勘探领域的广泛应用。下面本文基于 MapReduce 的开源实现 Hadoop 为主，分析其在应用中面临的问题，并提出改进方向。

5.1 应用问题分析

对内存密集型应用的支持。石油勘探应用算法的一个突出的特点是对物理内存的大小有较高的需求，物理内存不足时将会引发大量磁盘 I/O，降低系统性能。商用集群一个 CPU 核一般配备 1~2GB 内存，常用节约内存开销的优化手段是利用 MPI 与多线程混合编程方式，每台机器只分配一个 MPI 进程，节点内利用多线程并行，共享只读数据，例如本文 MPI 版本的层析成像算法，内存占用量最大的速度模型是只读的，整个网格的慢度增量也可以共享，利用互斥锁保护对同一网格单元的并发写，内存开销降低之后，CPU 的计算能力就可以得到充分发挥。而 MapReduce 的 Map 都是独立的动态进程，无法利用共享内存解决内存占用，导致并发 Map 数降低，影响作业性能。

Map 任务的构建开销。石油勘探应用算法开始真正计算前，需要读入大量的参数，创建和初始化一系列数据结构，Map 的任务粒度足够大，才能降低算法的初始化开销所占的比例，但是任务粒度与 Map 任务总数之间存在矛盾，如本文实验所呈现的状况，任务粒度增大之后，Map 任务总数减少，出现负载不均衡，作业运行时间变长。

作业运行时参数的优化配置难题。石油勘探中的多数并行应用对高性能计算系统的使用方式属于“能力计算(capability computing)[16]”类，地震数据处理由一系列有先后次序的流程组成，要求每一流程内的并行作业充分利用所有计算节点，缩短执行时间。MapReduce 作业的每节点并发 Map 数、Map 任务粒度和 reduce 数目等参数对程序性能有重要的影响，然而这些参数互相之间有影响，同时它们又与 CPU 核数、作业节点数目、问题规模等相关，动态确定一组较优参数值是非常困难的，进而导致作业的计算效率非常低。

对迭代算法的支持。迭代求解是科学计算中常见的计算方式，MapReduce 并不能很好地支持这类迭代算法，需要借助辅助程序，串联多个独立的 MapReduce 作业，作业结束和再次启动过程中，多数计算节点处于空闲状态，造成性能损失。

5.2　改进方向

现有研究对 MapReduce 编程模型的改进主要分为 2 类，一类是扩展 MapReduce 的语义表达能力，MapReduceMerge[17]框架在 MapReduce 阶段之后加入 Merge 函数，做 2 个数据集的合并操作。Twister[18]是一个支持迭代计算过程的 MapReduce 模型实现，根据文献[7, 19]实验对比，在编写迭代算法方面，Twister 的易编程性和性能都要显著优于 Hadoop，Twister 的主要问题是不能自动容错。另外一类的改进是在性能方面，研究如何通过配置参数在资源消耗与作业性能方面取得平衡[20]，根据运行环境和应用特点设计更有效的调度算法[21, 22]，一些研究认为 Map 任务输出的中间数据写本地盘的方式带来较大的通信开销，利用内存缓存中间数据可以提高性能[18, 23]，而笔者认为对于内存密集型而且中间数据量较大的并行应用，利用本地磁盘存储中间数据，能够降低进程间的耦合性，利于计算与 I/O 的重叠，基于 MPI 的应用程序常采用类似策略优化性能。

为适用石油勘探高性能计算，MapReduce 可以在如下 3 个方向上改进。

1) 多核与众核的支持。当前基于集群体系结构的 MapReduce 实现，任务的调度执行都是采用进程级的粗粒度并行模式，而多核或者众核计算节点内更加适合中细粒度的线程级并行。可以考虑改进 MapReduce，使其支持进程与线程混合并行的调度机制与编程接口，一个节点仅有一个 Map 进程，Map 进程内调度线程并行处理<key, value>对，利用共享内存的特点，减少内存占用和拷贝，提高 cache 命中率，充分发挥多核和众核的性能。基于共享内存的 MapReduce 实现[24~27]等对该方向改进有重要的借鉴意义。

2) 灵活性方面的提升。石油勘探领域很多问题难以抽象成 Map 操作和 Reduce 操作，但是其中的子问题却可以用 MapReduce 表达，例如叠前 Kirchhoff 深度偏移算法[28]包括旅行时计算与偏移成像两部分，其中只有旅行时计算的并行模式适合 MapReduce 表达。如果适当降低 MapReduce 的抽象层次，将它作为通用并行编程模型之上的库或组件，与自身、其他并行组件、或底层的进程间通信原语随意组合，灵活性就可以得到显著提升，对迭代算法也能够提供更好的支持。

3) 分布式文件系统和并行文件系统的融合。本文选取的算法仅处理初至波数据，数据量小，计算强度大，而石油勘探中偏移成像算法则需要处理数 TB 级的陆地勘探数据，海洋勘探数据已经接近 PB 级，是很典型的数据密集型计算，并行文件系统的 I/O 存储带宽严重制约这类算法的性能和并行规模。为支撑 MapReduce 等编程模型的大规模并行，可以利用 I/O 中间件，融合分布式和并行文件系统，通过聚合计算节点的本地 I/O 带宽，提高整个系统的 I/O 可扩展性。I/O 中间层与编程模型协同负责数据的一致性和中间数据的容错，并根据应用特点、I/O 访问模式和负载情况，优化数据的放置与调度。

6　结束语

石油勘探过程中需要处理海量的地震数据，应用算法的主要是目标是充分利用片内、节点内和全系统多个层次的并行计算资源，实现超大规模的并行数据处理。MapReduce 编程模型简化大规模并行算法实现的能力，在很多应用中得到了体现。

本文的主要目的是探讨 MapReduce 对于石油勘探领域应用算法的适用性，利用 Hadoop MapReduce 实现了 Fresnel 层析成像算法，并与 MPI 实现对比，发现目前的 MapReduce 在性能和灵活性方面存在局限性，提出了 3 个改进方向：一是增加节点内的中细粒度的线程级并行支持，地球物理的算法多数属于内存密集型，通过共享内存减少内存占用量，也有助于提升性能，这个改进是非常迫切的，Intel 即将发布的 MIC 架构的众核处理器[29, 30]，每核分配到的内存大小非常有限，并不适合粗粒度数据并行；二是降低 MapReduce 的抽象层次，作为通用编程模型之上的并行模式库，提升其在表达能力和性能调优方面的灵活性；三是利用 I/O 中间件融合并行文件系统和分布式文件系统，提高高性能计算系统的 I/O 可扩展性。

参考文献：

[1] 赵改善. 我们需要多大和多快的计算机[J]. 勘探地球物理进展, 2004, 27(1): 22-28.

ZHAO G S. How big and fast computers can meet out needs[J].

Progress in Exploration Geophysics, 2004, 27(1): 22-28.

[2] 赵改善. 高性能计算在石油物探中的应用现状与前景[J]. 高性能计算发展与应用. 2009(29): 19-23.
ZHAO G S. Current status and outlook of high performance computing for seismic exploration[J]. Development and Application of High Performance Computing, 2009(29):19-23.

[3] BAUTISTA-GOMEZ L, TSUBOI S, KOMATITSCH D, *et al*. FTI: high performance fault tolerance interface for hybrid systems[A]. SC '11[C].Seattle, WA,USA, 2011. 31-32.

[4] DEAN J, GHEMAWAT S. MapReduce: simplified data processing on large clusters[J]. Communications of the ACM, 2008, 51(1): 107-113.

[5] Oracle Inc. Lustre: a scalable high-performance file system[EB/OL].http://www.lustre.org,2012

[6] SCHMUCK F, HASKIN R. GPFS: a shared-disk file system for large computing clusters[A]. FAST'02[C]. Monterey, CA, USA, 2002. 231-244.

[7] SRIRAMA S N, JAKOVITS P, VAINIKKO E. Adapting scientific computing problems to clouds using MapReduce[J]. Future Generation Computer Systems. 2012, 28(1): 184-192.

[8] FADIKA Z, DEDE E, GOVINDARAJU M, *et al*. MARIANE: MApReduce implementation adapted for HPC environments[A]. GRID '11[C].Lyon, France, 2011. 82-89.

[9] GUNARATHNE T, WU T, QIU J, *et al*. MapReduce in the clouds for science[A]. CLOUDCOM '10[C]. USA, 2010. 565-572.

[10] MACKEY G, SEHRISH S, BENT J, *et al*. Introducing Map-reduce to high end computing[A]. Petascale Data Storage Workshop, 2008. PDSW '08[C]. Austin, USA, 2008. 1-6.

[11] Committee of SEG Technical Standards. SEG Y rev1 Data Exchange format[S]. Society of Exploration of Geophysicist, 2002.

[12] 张建中，杨国辉，林文等. Fresnel 层析成像并行算法研究[J]. 计算机研究与发展. 2007, 44(10): 1661-1666.
ZHANG J Z, YANG G H, LIN W, *et al*. A parallel algorithm for fresnel tomography[J]. Journal of Computer Research and Development. 2007, 44(10):1661-1666.

[13] 刘玉柱，董良国，李培明等. 初至波菲涅尔体地震层析成像[J]. 地球物理学报. 2009, 52(9): 2310-2320.
LIU Y Z, DONG L G, LI P M, *et al*. Fresnel volume tomography based on the first arrival of the seismic wave[J]. Chinese Journal of Geophysics, 2009, 52(9):2310-2320.

[14] The apache software foundation. MapReduce tutorial[EB/OL]. http://hadoop.apache.org/common/docs/r0.20.203.0/Mapred_tutorial.html,2012.

[15] 王珊，王会举，覃雄派等. 架构大数据:挑战、现状与展望[J]. 计算机学报, 2011, 34(10): 1741-1752.
WANG S, WANG H J, TAN X P, *et al*. Architecting big data: challenges, studies and forecasts[J. Chinese Journal of Computers, 2011, 34(10):1741-1752

[16] RAICU I, FOSTER I T, BECKMAN P. Making a case for distributed file systems at Exascale[A]. LSAP '11[C]. San Jose, USA, 2011. 11-18.

[17] YANG H, DASDAN A, HSIAO R, *et al*. Map-reduce-merge: simplified relational data processing on large clusters[A]. SIGMOD '07[C]. Beijing, China, 2007. 1029-1040.

[18] EKANAYAKE J, LI H, ZHANG B, *et al*. Twister: a runtime for iterative MapReduce[A]. HPDC '10[C]. Chicago, USA, 2010. 810-818.

[19] ZHANG B, RUAN Y, WU T, *et al*. Applying twister to scientific applications[A]. CLOUDCOM '10[C]. Indianapdis, USA, 2010. 25-32.

[20] KAMBATLA K, PATHAK A, PUCHA H. Towards optimizing hadoop provisioning in the cloud[A]. HotCloud'09[C]. San Diego, CA, USA, 2009.

[21] SANDHOLM T, LAI K. MapReduce optimization using regulated dynamic prioritization[A]. SIGMETRICS '09[C]. 2009. 299-310.

[22] ZAHARIA M, Konwinski A, Joseph A D, *et al*. Improving MapReduce performance in heterogeneous environments[A]. OSDI'08[C]. San Diego, CA, USA, 2008. 29-42.

[23] EKANAYAKE J, PALLICKARA S, FOX G. Mapreduce for data intensive scientific analyses[A]. IEEE Fourth International Conference on eScience[C]. Indianapdis, USA, 2008. 277-284.

[24] TALBOT J, YOO R M, KOZYRAKIS C. Phoenix++: modular MapReduce for shared-memory systems[A]. MapReduce '11[C]. San Jose, USA. 2011. 9-16.

[25] RANGER C, RAGHURAMAN R, PENMETSA A, *et al*. Evaluating Mapreduce for multi-core and multiprocessor systems[A]. IEEE 13th International Symposium on High Performance Computer Architecture(HPCA)[C]. Phoenix, USA, 2007. 13-24.

[26] JIANG W, RAVI V T, AGRAWAL G. A Map-Reduce system with an alternate API for multi-core environments[A]. CCGRID '10[C]. Melbourne, Australia, 2010. 84-93.

[27] HE B, FANG W, LUO Q, *et al*. Mars: a MapReduce framework on graphics processors[A]. PACT '08[C]. Toronto, Canada, 2008. 260-269.

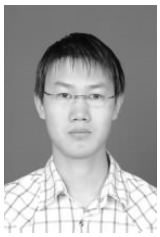
[28] CHANG H, VANDYKE J P, SOLANO M, *et al*. 3-D prestack kirchhoff depth migration: from prototype to production in a massively parallel processor environment[J]. Geophysics, 1998, 63(2): 546-556.

[29] SAULE E, CATALYUREK U V. An early evaluation of the

scalability of graph algorithms on the Intel MIC architecture[A]. 26th International Symposium on Parallel and Distributed Processing, Workshops and PhD Forum (IPDPSW), Workshop on Multithreaded Architectures and Applications (MTAAP)[C]. Shanghai, China, 2012.

[30] Intel many integrated core architecture[EB/OL]. http://en.wikipedia.org/wiki/Intel_MIC, 2012.

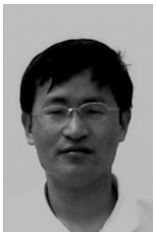
作者简介：



赵长海（1979），男，河南驻马店人，博士，北京航空航天大学讲师，主要研究方向为高性能计算。

晏海华（1964-），男，湖南益阳人，硕士，北京航空航天大学副教授，主要研究方向为软件测试和高性能计算。

刘晓朋（1989-），男，河北邯郸人，北京航空航天大学硕士生，主要研究方向为高性能计算。



熊登（1977-），男，河北涿州人，博士，中国石油东方地球物理公司物探技术研究中心工程师，主要研究方向为地震数据规则化和波场重建。

史晓华（1973-），男，贵州贵阳人，博士，北京航空航天大学副教授，主要研究方向为编译技术，并行计算和微处理器体系结构。