异构集群上的宏基因组聚类优化

韦建文[1] 许志耿[1] 王丙强[2] Simon SEE[1,3] 林新华[1,3]
(上海交通大学高性能计算中心 上海 200240)[1] (国家超算深圳中心 深圳 518055)[2]
(NVIDIA公司 新加坡)[3]

摘 要 宏基因组基因聚类是筛选致病基因的新型方法,其依赖于海量的测序数据、有效的聚类算法以及高效的计算机来实现。相关系数矩阵的计算是进行聚类前必须完成的操作,占总计算量的比重较大。以某基因库为例,包含1300个样本、每样本百万基因的数据,单线程运行需要27年。充分发挥多核CPU的潜力,利用GPU加速卡强大的计算能力,将程序扩展到多节点集群上运行,是重要而迫切的工作。在仔细分析算法的基础上,首先针对单CPU节点和单GPU卡做了高效实现,获得了接近理想的加速比;然后利用缓存优化进一步提升性能;最后使用负载均衡方法在MPI线程间分发计算任务,实现了良好的扩展。相比未优化的单线程程序,16节点CPU获得了238.8倍的加速,6块GPU卡获得了263.8倍的加速。

关键词 基因聚类,异构计算,缓存优化,负载均衡
中图法分类号 TP391 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2017.03.005

Accelerating Gene Clustering on Heterogeneous Clusters

WEI Jian-wen[1] XU Zhi-geng[1] WANG Bing-qiang[2] Simon SEE[1,3] James LIN[1,3]
(Center for High Performance Computing, Shanghai Jiao Tong University, Shanghai 200240, China)[1]
(National Supercomputing Center in Shenzhen, Shenzhen 518055, China)[2] (NVIDIA Corporation, Singapore)[3]

Abstract Metagenome clustering is a novel approach to detect flaw genes which relies on massive gene data, effective clustering algorithms and efficient implementation. In clustering, calculating correlation matrix is essential, accounting most of computing time. To take a gene repo as an example, which has 1300 samples and million genes, it will take about 27 years to cluster them. Therefore, developing efficient implementations for calculating correlation matrix is most essential. After analyzing the algorithms, we proposed and took several optimization approaches. First, we implemented an efficient multithread one using OpenMP dynamic scheduling. Secondly, we further improved the performance by utilizing cache on CPU and shared memory on GPU efficiently. Thirdly, we implemented a loadbalance work distribution which works well on the MPI program on CPU. Compared to the unoptimized single-threaded CPU program, the two fasted one, MPI+OpenMP on 256 CPU cores and MPI+CUDA on 6 GPU cards, achieve 238.8 and 263.8 speedups.

Keywords Gene clustering, Heterogeneous computing, Cache optimization, Load balance

1 简介

基因测序技术及生物信息学的发展为病理解析提供了新的途径,而"聚类分析"由于具有不依赖外部标记、适应性好等特点,在该领域有着广泛的应用。算法的核心是计算两两距离或者相关关系,计算量会随着样本数以及每个样本的特征数呈平方增长。随着基因研究对象从人本身扩展到人体环境基因(如肠道微生物,统称元基因),以及高通量测序仪的普及,更多的数据必然带来更准确的分类结果,但也必然给计算能力提出更高要求。以已经收集到的包含上千个测序样本、数百万基因的数据集为例,在单核心处理器上完成聚类需要27年。

使用OpenMP和MPI并行的高性能计算平台一直是解决此类计算密集问题的有力武器[3]。云计算平台凭借低廉的价格和定制能力,在处理中小规模聚类问题中发挥了重要作用,但其受限于Drayd等高层编程范式开发者从而无法高效控制下层处理器[5]。GPU和MIC等加速卡的出现为高性能并行程序的实现提供了新的思路,并且已经在基因聚类应用中取得了一定效果[6]。

在前期工作[2]的基础上,本文在异构集群上高效实现了文献[1]所述的宏基因组聚类算法的关键步骤:基因相关性计算,分别在CPU和GPU平台上针对两种架构特性对单节点做了优化实现。其中,CPU多线程版本性能是单线程的

到稿日期:2016-01-04 返修日期:2016-05-18 本文受国家高技术研究发展计划(863):高性能计算环境应用服务优化关键技术研究(2014AA01A302),日本学术振兴会(RONPAKU Fellowship)资助。
韦建文(1986-),男,硕士生,工程师,主要研究方向为高通量计算,E-mail:weijianwen@sjtu.edu.cn(通信作者);许志耿(1993-),男,硕士生,主要研究方向为异构计算;王丙强(1979-),男,博士生,研究员,主要研究方向为面向生物信息学异构计算;Simon SEE(1966-),男,博士生,研究员,主要研究方向为GPU异构计算;林新华(1979-),男,硕士生,副研究员,主要研究方向为性能计算、异构计算。

15.39 倍，单 GPU 卡性能是单线程的 63.86 倍。然后，使用 Cache 优化分别在 CPU 和 GPU 上又获得了 60%和 80%的性能提升。最后，在进行 MPI 扩展时进行作业负载均衡，在 16 节点 CPU 上实现了 10.56 倍加速，在 6 块 GPU 卡上实现了 1.92 倍加速。以 OpenMP 16 线程版本为基线，在 CPU 平台上最高实现了 15.52 倍加速，在 GPU 平台上最高实现了 17.14 倍加速。

2　宏基因组聚类问题

包含 n 个测序样本、m 个可识别基因组的测序数据可以表示为一个矩阵 $G_{n\times m}$，其中 w_{xj} 表示第 x 个样本、第 j 个基因的丰度。向量 g_j 是记录第 j 个基因在各个样本上的丰度，即 $g_j=(w_{1j},w_{2j},\cdots,w_{nj})$。计算第 j 个与第 k 个基因的相关系数可分为 3 步：

1)分别计算基因组 j 和 k 的检验值矩阵 $P^j_{n\times n}$ 和 $P^k_{n\times n}$；

2)由假设检验矩阵计算基因 j 和 k 的特征系数 v_j 和 v_k；

3)计算基因 j 和 k 的和谐数 c 以及不和谐对数 c'，然后与 v_j 和 v_k 计算得到相关系数 $T(j,k)$。

其中，假设检验矩阵第 x 行第 y 列元素 $p^j(y|x)$ 表示基因 j 在样本 x 与 y 之间的差异检验值，如式(1)所示。N_x 和 N_y 分别表示样本 x 和样本 y 的基因组丰度之和。

$$p^j(y|x)=(\frac{N_y}{N_x})^{w_{jy}}\frac{(w_{jx}+w_{jy})!}{w_{jx}!\ w_{jy}!\ (1+\frac{N_y}{N_x})^{w_{jx}+w_{jy}+1}} \quad (1)$$

基因 j 的特征值取为假设检矩阵中小于阈值 0.01 的元素的个数：

$$v_j=|\{p^j\,|\,p^j<0.01,p^j\in P^j\}|$$

对于基因向量 $g_j=(w_{1j},w_{2j},\cdots,w_{nj})$ 和 $g_k=(w_{1k},w_{2k},\cdots,w_{nk})$，若某两对丰度值$(w_{xj},w_{xk})$ 和(w_{yj},w_{yk})有这样的关系：$w_{xj}>w_{xk}$ 且 $w_{yj}>w_{yk}$，或者 $w_{xj}<w_{xk}$ 且 $w_{yj}<w_{yk}$，则认为这两个丰度对是和谐的；否则这两个丰度对是不和谐的。由两个基因 j 和 k 的和谐对数量 c、不和谐对数量 c'，以及基因特征数 v_j，v_k，可计算两个基因的 Kendall 相关系数 T(j，k)：

$$T(j,k)=2\times|\frac{c-c'}{v_j-v_k}|$$

其中，式(1)包含高次幂运算，直接计算时运算量较大，通常两边取对数后计算：

$$\log p^j(y|x)=w_{jy}\log(\frac{N_y}{N_x})+\log(w_{jx}+w_{jy})!-\log w_{jx}!-\log w_{jy}!-(w_{jx}+w_{jy}+1)\log(1+\frac{N_y}{N_x})$$

以包含 n 个样本、m 个基因组的 $G_{n\times m}$ 为例，在这一步骤中计算所有基因向量两两之间的相关系数共需 C_n^2 次计算，而每一步运算需 C_m^2 次大小比较，因此相关系数的计算共需要完成$O(m^2n^2)$次基本操作。

3　并行优化

3.1　单 CPU 单 GPU 上的多线程优化

3.1.1　程序剖析

由 CPU Profile 结果估算的运算强度约为 0.69Flops/byte，浮点运算能力约为 0.138GFlops/sec，实测内存带宽0.2 GByte/s。由 Roofline Model 可知，该单线程版本位于性能非常低的一个区域，没有充分利用平台的内存带宽和浮点运算能力。因此，充分的并行化、Cache 优化、线程间负载均衡将是我们关注的重点。

3.1.2　CPU 上的 OpenMP 和循环展开优化

p 值计算两两独立，又因为计算 p 值不需要同步共享变量，因此可以取消 OpenMP 块末尾默认设置的同步屏障，使用 nowait 语句减小线程无谓的等待时间。使用 OpenMP 动态调度 schedule(dynamic)可改善原先静态分配时负载不均的现象。循环内嵌套的小循环使用 unroll 能利用 CPU 的 SIMD。

3.1.3　单 GPU 卡实现

GPU 卡可以支持数千并发线程进行并行计算，设计的关键是将数量庞大的平行计算任务映射到 GPU 线程上，并保证数据同步。让每一个线程计算如下值：$p(y_1|x)$，$p(y_2|x)$，$\cdots$，$p(y_m|x)$，这个粒度能避免由粒度太细而频繁分支带来的开销。GPU 计算线程间无需同步，具有很高的执行效率。

3.2　缓存优化

在 CPU 上使用 Cache Blocking 可以让 Cache Line 得到更充分的复用，若选取合理的值，则需从下一级较慢存储读取的数据量降为原先的 1/BLOCK_SIZE。

GPU 的 shared memory 与 CPU cache 有类似的功能。在 block 启动前将数据从 global memory 载入 shared memory，作业结束后通过 sync 保证该线程完全结束，然后再进行下一轮计算。由于 shared memory 是 block 私有的，且计算时仅当载入同一基因的丰度值才有复用意义，因此在 GPU 上启动的 block 数应与 BLOCK 大小保持一致，保证分块数据载入同一 block。

3.3　MPI 与负载均衡优化

在 CPU 集群上，使用 MPI+OpenMP 混合编程模型将程序扩展到多个节点上。每个节点启动 1 个 MPI 进程负责数据通信，计算任务由 OpenMP 线程执行。程序执行流程为：1)MPI 主进程向各 MPI 子进程分发计算数据；2)收到数据后，MPI 进程启动 OpenMP 线程计算子问题；3)MPI 主进程接收部分计算结果，归并后得到最终结果。这样的 MPI 计算过程要进行两轮，分别计算检验值矩阵和特征系数。

但是这样实现的版本扩展性并不理想：相比单节点，16 节点的加速比仅为 6.42。各个线程在计算、等待、网络通信上所消耗的时间如图 1 所示。在计算检验矩阵阶段，作业负载不均衡导致的超长等待时间是程序扩展性差的主要原因。

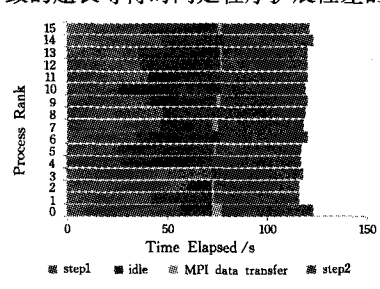

图 1　MPI 进程负载不均衡

MPI 主线程在初始阶段以静态分发任务，每个 MPI 子线

程收到的数据行数是一样的。但由式(1)可知,等行数的数据所需的计算量并不相等,w_{yj} 越大,计算 $p^j(y|x)$ 所需的时间越多。又因为每个 MPI 线程要计算与多个基因的检验系数,因此将每一行数据的总和,即样本 x 各基因丰度值的总和作为计算量的度量,在分发计算数据时考虑该线程的负载。

4 实验结果与分析

在上海交通大学高性能计算集群 π 上完成了实验。实验使用了两种类型的节点——CPU 节点和 GPU 节点,CPU 和 GPU 的主要特性如表 1 所列。每个计算节点配置 2 颗 Intel E5-2670 CPU、64GB 内存、56Gbps FDR Infiniband 网卡,GPU 节点还配备 2 块 Tesla K40 加速卡。输入数据包含 1300 个样本,每个样本测量 50000 个基因组的丰度,数据大小约 260MB。

表 1 CPU 和 GPU 测试平台主要性能参数

	核心数	主频	双精度浮点峰值/TFLOPS	内存带宽/GB/s
E5-2670×2	16	2.6GHz	0.33	51.2
Tesla K40×2	2880	745MHz	1.43	288

4.1 单 CPU 节点和单 GPU 卡优化

OpenMP CPU 单节点扩展性如图 2 所示。在 CPU 节点上单线程版本需要 16693s 完成计算,静态负载版本以 16 核运行需要 1134s,动态负载版本以 16 核运行需要 1085s,分别实现了 14.72 倍和 15.39 倍的加速。并且动态负载版本比静态负载版本有约 5%的性能提升。在单块 K40 加速卡上运行的版本耗时为 261s,相比单线程 CPU 版本,加速比为 63.86。

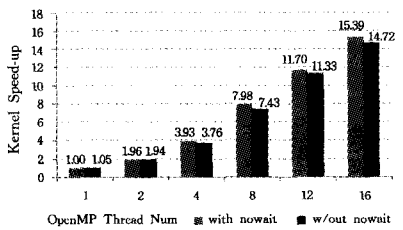

图 2 OpenMP CPU 单节点扩展性

4.2 缓存优化

测试了取不同 BLOCK 大小时的性能。CPU 版本块大小取 256 时性能最佳,计算耗时为 679s,性能是动态负载 OpenMP 版本的 1.6 倍。GPU 版本分块大小取 32 时性能最佳,计算耗时为 115s,性能是未优化版本的 1.8 倍。

4.3 缓存优化 MPI 与负载均衡测试

在 CPU 版本使用负载均衡任务划分后,16 节点 CPU 运行时,线程等待所占的比例变小。在 1—16 节点的测试结果如图 3 所示,未使用负载均衡与使用负载均衡的版本间差距逐渐扩大。在使用 16 节点时,负载均衡版本加速比达到 10.56,远高于非均衡版本的 6.42。

但是,负载均衡在 MPI+GPU 中效果不明显。1—6 块 GPU 卡的可扩展性测试如图 4 所示。即使增加到 6 块卡,两种方法的差距也在 5%以内。6 块卡的计算耗时为 66s,相比于单块卡只实现了 1.92 倍的加速比,说明之前建立的以 w_{yj} 为标准的计算量度量并不适合 GPU。

最后,以 16 线程动态负载的 CPU 版本作为性能基线,各种优化方法所取得的加速比如图 5 所示。

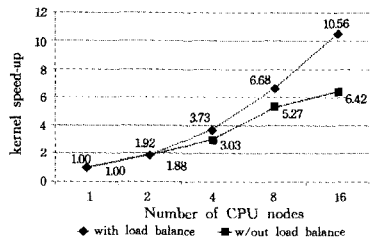

图 3 CPU 节点上的可扩展性

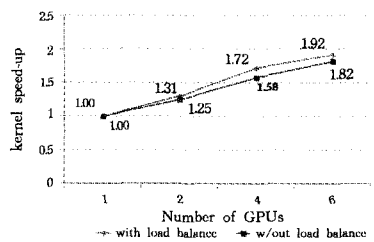

图 4 GPU 卡的扩展性

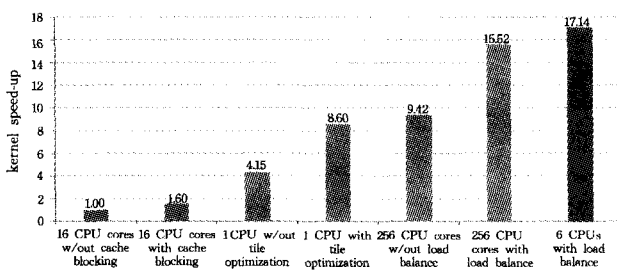

图 5 相关系数计算的整体加速比

结束语 计算能力是制约宏基因组聚类应用发展的重要因素,对聚类分析中耗时占 99%以上的相关系数计算做了优化。首先使用 OpenMP 动态调度在 16 核平台上实现了 15.39倍的加速。其次,在 CPU 和 GPU 上充分利用高速缓存,性能分别得到 60%和 80%的提升。最后,在进行 MPI 扩展时进行作业负载均衡,在 16 节点 CPU 上实现了 10.56 加速,在 6 块 GPU 卡上实现了 1.92 倍加速。以 OpenMP 16 线程版本为基线,CPU 平台上最高实现了 15.52 倍加速,GPU 平台最高实现了 17.14 倍加速。后续计划在 GPU 负载均衡、CPU 向量化、CPU+GPU 协同工作上继续完善工作。

致谢 感谢上海交通大学高性能计算中心提供 π 上的 CPU 和 GPU 实验环境;感谢上海交通大学高性能计算中心秦强同学前期的优化工作。

参 考 文 献

[1] QIN J, LI Y, CAI Z, et al. A metagenome-wide association study of gut microbiota in type 2 diabetes [J]. Nature, 2012, 490 (7418): 55-60.

[2] GUO G X, LU X J, QIU S, et al. GPU-accelerated Gene Clustering Method for Metagenome [C]// Proceedings of HPC China 2014. Guangzhou, 2014: 324-328. (in Chinese)
郭贵鑫,陆旭佳,邱爽,等. 基于 GPU 加速的宏基因组聚类方法 [C]// HPC China 2014 会议. 广州,2014:324-328.

(下转第 47 页)

应时间。图 10 示出了在不同的任务规模下 CTRQ 并行化算法相对上述串行算法的加速比。当线程数为 2 时,最大取得 1.6 倍的加速比;线程数为 4 时,最大取得 1.8 倍的加速比;随着线程数量的增加,加速比随之增大,且在线程数量为 8 时达到峰值,最大取得 1.9 倍的加速比。此后,随着线程的有效计算时间以及线程切换开销占整个运行时间的比例增大,加速比逐渐减小。

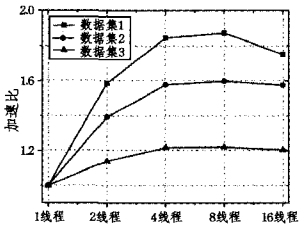

图 10　CTRQ 并行算法加速比

结束语　本文深入分析了 MARS 平台架构,针对该系统执行大范围区域查询时效率不高的问题,设计了一种基于数学补集思想的面向大规模数据的区域查询方法。实验表明,在数据集一定的情况下,区域查询范围与系统响应时间成反相关。另外,本文给出了大规模数据下补集转换区域查询结合 OpenMP 编程模型的并行化算法实现,进一步降低了系统检索的响应时间,提高了数据的查询效率,为基于高性能文件存储系统的 MARS 数据归档与检索奠定了基础,同时,对应用其他领域的超立方体技术优化方法也具有参考价值。

参 考 文 献

[1] RAOULT B. Architecture of the new MARS server[EB/OL]. [2015-06-01]. http://old. ecmwf. int/archive/publications/manuals/mars/server. pdf.

[2] SARAWAGI S, AGRAWAL R, MEGIDDO N. Discovery-driven Exploration of OLAP Data Cubes[J]. Lecture Notes in Computer Science, 1998, 1377: 168-182.

[3] HAN J, KAMBER M. Data Mining: Concepts and Techniques. Second Edition[J]. San Francisco, 2006(1): 1-25.

[4] GRAY J, CHAUDHURI S, BOSWORTH A, et al. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals[J]. Data Mining and Knowledge Discovery, 1997, 1(1): 29-53.

[5] SHAPIRO M A, THORPE A J. THORPEX International Science Plan1[J]. Boletín De La Organización Meteorológica Mundial, 2004, 6(11): 238-242.

[6] SHEN W H, ZHAO F, GAO H Y, et al. The construction of national meteorological archival and retrieval system [J]. Journal of Applied Meteorological Science, 2004, 15(6): 727-736. (in Chinese)
沈文海,赵芳,高华云,等. 国家级气象资料存储检索系统的建立[J]. 应用气象学报,2004,15(6):727-736.

[7] HO C T, AGRAWAL R, MEGIDDO N, et al. Range queries in OLAP data cubes[J]. ACM Sigmod Record, 1970, 26(2): 73-88.

[8] HONG S, SONG B, LEE S. Efficient Execution of Range-Aggregate Queries in Data Warehouse Environments[M]// International Symposium on Requirements for Poultry Virus Vaccines. S. Karger, 1974: 299-310.

[9] AGARWAL S, AGRAWAL R, DESHPANDE P M, et al. On the computation of multidimension alaggre gates[C]// VLDB. 1996: 506-521.

[10] ZHAO Y, DESHPANDE P M, NAUGHTON J F. An array-based algorithm for simultaneous multidimensional aggregates [C]// Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data. 1997: 159-170.

[11] XUE Y S, HUANG Z H, DUAN J J, et al. An Efficient Method for Parallel Multi-Dimensional Join and Aggregation[J]. Journal of Computer Research and Development, 2004, 41(10): 1661-1669. (in Chinese)
薛永生,黄震华,段江娇,等. 一种并行处理多维连接和聚集操作的有效方法[J]. 计算机研究与发展,2004,41(10):1661-1669.

[12] BOOCH G. Object-oriented development[J]. IEEE Transactions on Software Engineering, 1986, 12(2): 211-221.

[13] WU P, XU H P, CHEN H G. Application of Object-oriented for Metadata Research[J]. Journal of Tongji University (Natural Science), 2010, 38(11): 145-151. (in Chinese)
吴萍,许惠平,陈华根. 面向对象方法在元数据研究中的应用[J]. 同济大学学报(自然科学版),2010,38(11):145-151.

[14] SHENG L I, WANG S. Star Cube——An Approach to Implementing Data Cube Efficiently[J]. Journal of Computer Research & Development, 2004, 41(4): 587-593.

[15] GUTTMAN A. R-trees: A dynamic index structure for spatial searching[C]// Proc. of the ACM SIGMOD International Conference on Management of Data. 1984: 47-57.

[16] LI J, ROTEM D, SRIVASTAVA J. Aggregation Algorithms for Very Large Compressed Data Warehouses[C]// Proceeding of the 25th VLDB Conference. 1999: 651-662.

[17] SONG S L, SONG J Q, REN K J. Design of a parallel algorithm for data cube of MARS[J]. Computer Engineering & Science, 2014, 36(12): 2410-2417. (in Chinese)
宋石磊,宋君强,任开军. 气象数据归档与查询系统超立方体结构并行算法设计[J]. 计算机工程与科学,2014,36(12):2410-2417.

[18] SATO M. OpenMP: parallel programming API for shared memory multiprocessors and on-chip multiprocessors[C]// International Symposium on System Synthesis. IEEE, 2002: 109-111.

(上接第 22 页)

[3] WANG M, ZHANG W, DING W, et al. Parallel Clustering Algorithm for Large-Scale Biological Data Sets[J]. PLoS ONE, 2014, 9(4): 13-15.

[4] W W M, KIRK D B. Programming Massively Parallel Processors [M]. New York: Morgan Kaufmann, 2012: 120-128.

[5] EKANAYAKE V, GUNARATHNE T, QIU J. Cloud Technologies for Bioinformatics Applications[J]. IEEE Transactions on Parallel and Distributed Systems, 2011, 22(6): 998-1011.

[6] BUSTAMAM A, BURRAGE K, HAMILTON N A. Fast Parallel Markov Clustering in Bioinformatics Using Massively Parallel Computing on GPU with CUDA and ELLPACK-R Sparse Format[J]. IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB), 2012, 9(3): 234-240.