星系分组算法的并行设计与优化:SGI系统与分布式集群对比

司雨濛[1] 韦建文[1] Simon SEE[1,2] 林新华[1]
(上海交通大学高性能计算中心 上海 200240)[1] (NVIDIA 新加坡 138522)[2]

摘 要 Halo-based Galaxy Group Finder (HGGF) 是一种有效的星系分组算法,它根据星系的空间位置、红移、质量等多种属性将星系分组,从而为星系组的形成与演化研究提供重要依据。但是,算法当前的 OpenMP 实现版本仅能利用单节点提供的资源,在大规模星系分组问题上的应用受到限制。一种优化思路是采用多机并行,使其可以利用更多资源来解决更大规模的星系分组问题,并缩短执行时间。因此,有必要对算法重新进行设计与实现。实现此目标的一大挑战是程序中存在大量半随机性远端内存访问,其在多机并行环境下会对性能造成重大影响。为克服这一难题,设计中提出了邻接星系链表思想,并采用 Unified Parallel C (UPC)进行程序实现。对于核代码部分,使用 4,8,16 节点时,可分别取得 2.25,2.78,5.07 倍的加速比;同时,对单个节点的内存需求也显著减少。OpenMP 版本在 SGI UV 2000 上的实验结果显示,受限于程序的访存特性与机器体系架构的特点,类似 HGGF 算法这种具有随机数据访问特征的程序,很难有效利用 NUMA 结构的共享内存系统中提供的大规模线程与内存资源来直接取得高加速比。在分布式内存集群上采用两级并行设计,以更好地利用局部性原理,可能是更好的解决方案。

关键词 高性能计算,星系分组,并行计算,UPC,OpenMP
中图法分类号 TP391 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2017.10.015

Parallel Design and Optimization of Galaxy Group Finding Algorithm on Comparation of SGI and Distributed-memory Cluster

SI Yu-meng[1] WEI Jian-wen[1] Simon SEE[1,2] James LIN[1]
(Center for High Performance Computing, Shanghai Jiao Tong University, Shanghai 200240, China)[1]
(NVIDIA, Singapore 138522, Singapore)[2]

Abstract Halo-based galaxy group finder (HGGF) is an effective algorithm that accomplishes the task of galaxy group finding based on galaxy coordinates, redshift and mass etc., and provides great help in the research of galaxy group formation and evolution. However, current pure OpenMP implementation of the algorithm is limited by the resource of the underlying single compute node when dealing with large-scale group finding problems. One of the possible solutions is using resources from multiple nodes to reduce execution time while solving large-size galaxy group finding problem. Therefore, it is essential to redesign and implement the algorithm. The major hurdle for such an attempt is remoting memory access due to semi-random galaxy access in the algorithm which damages the performance in multi-node environment. To tackle such a problem, we paralleled the algorithm with adjacent galaxy list design and used unified parallel C (UPC) to implement it. 2.25, 2.78 and 5.07 times speedup for the kernel were achieved with 4, 8 and 16 nodes respectively. Meanwhile, the memory requirement on each node was also reduced significantly. Experiments of OpenMP version of the algorithm on SGI UV 2000 show that due to the nature of the program and the features of NUMA architecture, programs with random memory access behavior like HGGF may not readily benefit from the large number of threads and shared memory provided by such machines. Two-level parallel design that takes advantage of locality principle on distributed memory clusters may be a better solution.

Keywords High performance computing, Galaxy group finding, Parallel computing, UPC, OpenMP

到稿日期:2016-11-21 返修日期:2017-01-18 本文受国家重点研发计划(2016YFB0201400,2016YFB0201800),日本学术振兴会 JSPS 的 RONPAKU 项目,上海交通大学 SMC一晨星青年学者奖励计划资助。

司雨濛(1992—),男,硕士,主要研究方向为高性能计算;**韦建文**(1986—),男,工程师,主要研究方向为高性能计算,E-mail:weijianwen@sjtu.edu.cn(通信作者);**Simon See** 男,研究员,主要研究方向为 GPU 计算;**林新华**(1979—),男,高级工程师,主要研究方向为性能建模与优化。

1　简介

当今的天文学研究在很大程度上依赖于天文观测与软件模拟收集到的数据。随着数据容量的不断增大，我们处理与分析数据的能力面临重大挑战，并逐渐成为天文学研究的一大瓶颈[1]。

星系组的形成与其结构研究是天文学科研领域一个至关重要的课题，它对于帮助人类了解宇宙大规模结构的演化具有重大意义。杨小虎教授提出的 Halo-based Galaxy Group Finder（HGGF）[2-3]是一种基于传统 FoF 方法做出若干改进与创新的星系分组算法，该算法将星系的空间位置、红移、质量等作为分组考虑的依据，比 FoF 方法更精准。然而，当前的算法实现无法满足我们解决大规模星系分组问题的需求，亟需对此做出改进。

我们认为有必要对 HGGF 算法进行全新的算法设计，以实现多机并行，从而利用更多的计算资源来达成加快算法的解题速度、处理更大规模星系数据的目标。使用 Unified Parallel C进行并行化工作，首先需要对问题空间进行划分，然后利用各个 UPC 进程对各子空间进行处理。利用邻接星系列表设计方案，消除了高昂的远程内存访问代价，避免了采用多节点时的大量远程内存访问开销，从而提高了程序性能。通过对不同的数据分布方法以及包括大块内存传输在内的优化手段进行研究发现，利用显式数据分布来安排数据在各个进程中的存储以利用空间局部性，以及使用大块内存传输避免细粒度远端访存，对于提升程序性能至关重要。各项实验结果表明，若想在 UPC 程序中取得良好的性能，手动调优是十分必要的。此外，在 SGI UV 2000 上的性能测试与分析指出，类似 HGGF 算法这种具有随机性内存访问且对访存敏感的程序，较难从提供大量线程与大容量共享内存的 NUMA 结构 SGI 机器中直接受益，使用多级多机并行设计，利用分布式内存集群解决此类问题，不失为一种更加行之有效的途径。

2　相关工作

许多研究人员针对各类星系分类算法[8-10]的并行化做出了许多工作。美国西北大学的 Liu Ying 等人对 HOP 算法进行了并行化设计与实现[11]，他们通过建立一棵并行 KD 树，将星系平均分布于多个处理器。该算法实现了 MPI 编程，并且被应用于 ENZO 宇宙模拟软件的输出。华盛顿大学的 YongChul Kwon 等人针对传统的 FoF 星系分组算法做出了改进与并行化，文献[1]主要介绍了如何利用 MapReduce 框架对星系分组问题进行处理。他们利用 DryadLINQ 在 Dryad 并行数据处理系统中对所设计的 dFoF 算法进行了实现，并在一个小型集群上对算法进行了测试，其并行版本与天文学研究中广泛使用的一个精细优化版本拥有相近的性能，并且在较不均衡的数据集上，相对于手动调优的版本有 2.7 倍左右的加速比。卡内基梅隆大学的 Fu Bin 等人[12]提出了Disc-Finder算法，并利用 Hadoop 实现了该算法。其主要思想是将星系数据按空间位置分块，然后在每个空间区域中利用线性分组算法进行分组，并最终对分组结果进行合并。

与上述工作的不同之处在于，本文使用 UPC 作为实现工具，针对一种不同的星系分组算法即 HGGF 算法进行了并行化，并对不同体系结构上不同算法实现的性能进行对比，探究更适合此类问题的解决方案。

3　背景

3.1　HGGF 算法

HGGF 算法根据星系的空间位置、红移、质量等属性对星系进行分组。图 1 为该算法的主要流程。

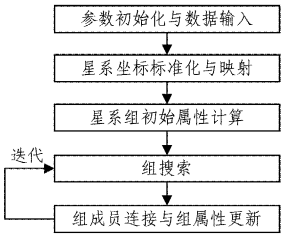

图 1　HGGF 算法的流程图

在星系坐标映射步骤中，星系坐标将进行伸缩变换并标准化。初始时，将各个星系当作独立的星系组，并计算出各个组的相关属性，然后开始组搜索。星系组的搜索是以每个中心星系为基础进行的。搜索过程中，每个周边星系的坐标、红移以及密度对比（density contrast）都与其可能的星系组的中心进行计算[3]。候选星系若满足一定条件，则会被当作该组的成员。紧接着，新的组属性以及组成员链接会被推导出来，并进行相应的更新操作。

3.2　Unified Parallel C

Unified Parallel C 是一种 Partitioned Global Address Space（PGAS）语言，它对标准 C 语言进行了扩展，为程序员提供了一个抽象的全局内存地址空间；开发人员可以使用这一地址空间而无需关心其底层的内存分布[4]。UPC 程序采用 Single Program Multiple Data（SPMD）执行方式，每一个进程根据其唯一标识符可以拥有各自不同的执行路径。我们使用的 UPC 实现为 Berkeley UPC，它是一种高度可移植、高性能的 UPC 实现。

3.3　SGI UV 2000 概览

SGI UV 2000 是 SGI 公司生产的第六代一致性共享内存（Coherent Shared Memory，CSM）计算机，它配备 Intel® Xeon® 处理器并提供最高可达 64TB的内存[6]。NUMAlink® 技术为该机器提供高性能、低延迟的系统互联，并支持上千个 CPU 核心。图 2 为其体系结构图。其共享内存基于 NUMA 结构，因此对本地与外部 NUMA 域的内存存取存在时延差异。

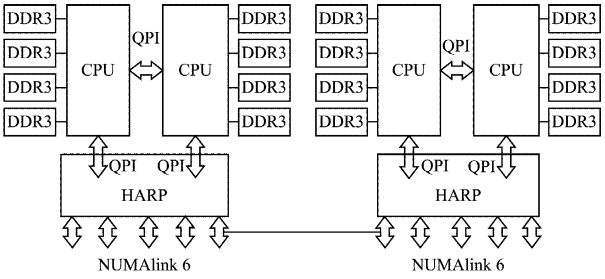

图 2　SGI UV 2000 体系结构

4　程序特性

4.1　性能瓶颈

Intel VTune[5]性能分析结果显示，HGGF 算法拥有非规

则内存访问的特性。因此,本算法主要受限于内存访问开销,这是由半随机性星系数据访问引起的。

在 HGGF 算法中,参数 Nsample 影响着星系分组问题的规模,而程序的内存需求随着该参数线性增加。根据杨小虎教授等人的研究[3],$Nsample=1024$ 可以提供较为良好的星系分组研究数据,而在此情况下需要 460 GB 左右的内存。由于单节点资源的限制,需要提出新的解决方案来更快速地解决大规模星系分组问题。一种方案是采用原始程序,直接利用 CSM 机器提供的大量内存与线程资源试图解决此类问题;另一种方案是对算法进行重新设计,利用多机多级并行策略进行解决。

4.2 OpenMP 版在 SGI UV 2000 上的性能特征

通过测试表明,上述第一种方案中,当前 OpenMP 实现[7]无法有效利用提供了大规模线程与内存资源的 CSM 机器,原因主要有 3 点。

首先,NUMA 结构中不同的 CPU 在访问不同的内存时存在访存速度差异,这限制了大量线程的有效性。当程序使用 16 线程时,每个线程的执行时间相对均衡,因为这些线程都运行在位于同一 NUMA 域中的 CPU 上,它们的内存访问均为本地存取。然而随着线程数量的增加,这些线程会运行于更多的 CPU 上,但它们位于不同的 NUMA 域,此时各线程读取星系数据时就要跨越 NUMA 域,造成更长的内存访问时间。由于此算法受限于内存访问速度,因此这种跨域访问会严重影响程序的性能。

其次,该算法的特性导致各个线程对于星系数据的需求无法预判,即每个线程不知道未来需要哪些星系数据,因此也就无法利用 first-touch 原理由每个线程将其需要的星系数据放置于该线程所在的 NUMA 域内存中。

最后,随着 OpenMP 线程数量的增加,在线程 fork/join 上的开销增大,这一开销在一定程度上也降低了程序的性能。

因此,为了进一步加速算法,解决更大规模的问题,有必要设计并实现 UPC+OpenMP 的两级并行算法。

5 并行设计与实现

5.1 问题空间分解

将三维问题空间在 Y-Z 方向上进行划分,使得每个子空间中星系数量尽可能接近。每个进程所负责的区域被称为其管理区域(managed area)。每个管理区域向外延伸一个搜索半径 r,这一区域被称作邻接区域(adjacent area)。使用 4 个进程时的划分如图 3 所示。

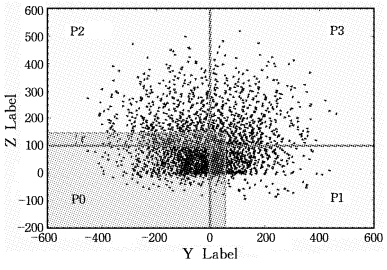

图 3 星系域分解

5.2 邻接星系列表

在搜索过程中,绝大多数星系中心都能在其管理区域内找到所需的可能组成员,然而位于管理区域边缘的星系组中心星系所需的组成员星系可能位于管理区域之外。通过对管理区域外的星系的全局编号进行映射,可以将其转换为本地邻接星系列表中的一个下标,然后在本地的邻接星系列表中找到对应星系数据,从而避免远程内存访问。

全局星系编号到本地邻接星系列表下标的映射是通过每个进程中的一张映射表完成的,此表反映了本进程邻接区域中的邻接星系。该映射表的每一项可能是一个局部下标,标志着具有某个全局编号的星系在本地邻接星系列表数组中的位置;也可能是-1,标志着该星系不存在于本地邻接星系列表。

5.3 UPC 的实现与优化

在 UPC 程序中可以采用两种数据分布方法:隐式分布与显式分布。采用隐式分布时,星系数据被存储于一个大的逻辑数组中,进而在所有进程之间以一种循环(round-robin)的方式进行存储。该方法的优点是,我们无需关注数据分布的细节,实现相对简单,对于原代码的修改量相对较小;但其缺点是性能会由于局部性的缺失而受到较大影响。在显式分布中,每个进程在共享内存空间中进行动态内存分配,并将属于本进程的星系数据存储其中。由于某个进程所需的大部分星系(管理区域内的星系)都存储在本进程中,因此程序可以发挥局部性优势,进而在一定程度上削减远程内存的访问需求,其代价是需要我们对数据分布进行掌控。

在星系坐标映射与组成员星系链接的过程中,存在细粒度的远程内存访问。为了对此进行优化,主进程可以利用 upc_memget()函数从其他进程中读取大块内存,并将其放置于本地缓存中,然后在本地进行进一步操作。通过这种方式,可以将频繁的细粒度远端访存转换为粗粒度数据传输结合本地内存访问,从而达到提升性能的目的。

6 性能评估

实验环境为上海交通大学的 Pi 超级计算机以及一台 SGI UV 2000。Pi 超级计算机的每个计算节点配置 2 个 Intel E5-2670 CPU,256 GB DDR3 1600MHz 内存,并采用 56Gbps FDR Infiniband。SGI UV 2000 拥有 32 个 Intel E5-4620 v2 CPU,4 TB 内存,采用 NUMAlink 6 互联技术。使用 GCC 5.2作为 C 编译器并使用 Berkeley UPC 和 GUPC 5.2 作为 UPC 编译器。每个 UPC 进程运行于一个节点上,由于存在两级并行,因此每个 UPC 进程包含 16 个 OpenMP 线程。

6.1 隐式与显式数据分布

表 1 所列为分别采用隐式与显式数据分布时,4 个 UPC 进程在核代码中的耗时。前者的耗时相较于后者高出了两个数量级,这是由于在此模式中星系数据根据星系的全局编号以一种循环的方式分布于各个节点,因此在组搜索过程中无法很好地利用空间局部性,从而导致大量的远端内存访问,造成了性能下降。事实上,采用隐式数据分布,使用 4 个 UPC 进程时,每个进程仅有大约 25%的星系存储于本地。因此,尽管隐式数据分布方法在编程上较为简单,但其数据分布模式对于 HGGF 算法并不理想。相比之下,显式数据分布使得各个进程将属于自己管理区域空间范围内的星系数据存储于自己在全局共享内存空间中动态分配的内存中,这样星系分组过程中大部分所需星系数据都可以在本地取得,大大提升

了执行效率，再结合邻接星系列表设计，还可进一步减少远端内存访问的需求。

表 1　隐式与显式数据分布下核心代码耗时对比/s

	8	16	32	64
Implicit	23966.10	47901.93	95370.72	196321.20
Explicit	132.59	251.34	490.093	996.98

6.2　大块内存传输优化对性能的影响

在组内星系链接过程中存在大量细粒度远端内存访问，这会造成性能恶化，可以采用大块内存传输（bulk memory transfer）方式对此进行优化。表 2 展现了大块内存传输对组内星系链接的性能影响。可以看出，采用该优化手段后，星系链接过程的执行时间有了较大幅度的削减。

表 2　星系链接过程采用大块内存传输优化与否的耗时对比/s

	8	16	32	64	128	256	512
Non-bulk	100.40	207.53	511.96	947.99	1652.66	5274.92	11063.74
Bulk	18.87	44.85	101.36	196.84	422.23	963.90	1455.75

6.3　邻接星系列表设计的有效性分析

表 3 对是否采用邻接星系列表设计方法的性能进行了对比，*Nsample* 从 8 到 64 变化，采用 4 个进程执行程序，两种情况下核代码执行时间相差两个数量级。若不使用邻接星系列表，对于位于进程管理区域边缘的星系组，搜索管理区域以外的组成员星系时必须从远端获取星系数据，这一操作需要付出高昂的代价并造成较长的执行时间。采用该设计后，每个进程可以在本地进行一次映射，然后使用映射得到的下标在邻接星系列表中得到所需的星系数据，从而大大加速了程序的执行速度。实验结果证明了文中设计的有效性。

表 3　采用邻接星系列表与否的耗时对比/s

	8	16	32	64
Wiout AGL	12861.23	24254.31	47686.04	96131.80
With AGL	132.59	251.34	490.093	996.98

6.4　内存消耗与耗时

图 4 为采用不同数量计算节点时的单点内存需求。使用 4 个节点、4 个 UPC 进程时，对单节点内存需求降低到 32.5%，而使用 8，16 节点时，这一数值分别达到 23.8%与 14.7%。这使得利用多个具有较小内存容量的集群节点对大规模星系分组问题的求解成为可能。

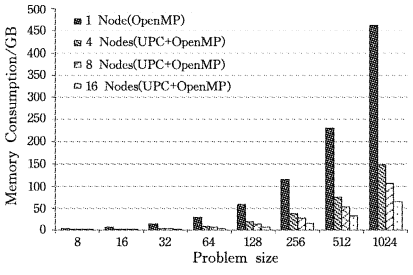

图 4　不同节点数量下的单点内存消耗

图 5 为采用不同数量 UPC 进程（4，8，16）时，UPC 版本与单节点 OpenMP 版本的核代码耗时对比。随着 UPC 进程数量的增加，核心代码的执行时间缩短。采用 4，8，16 进程时，核代码部分的平均加速比为 2.25，2.78，5.07。在进行区域划分时尽量使得各个节点内的星系数量均衡，但是同一子空间内不同位置的星系密度差异以及随之而来的计算强度的不同，会造成加倍的节点数量无法成倍提升运算速度。

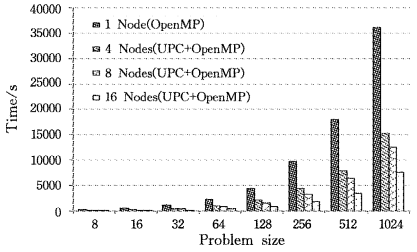

图 5　核心代码在集群上的耗时

图 6 为 SGI UV 2000 上采用不同线程数量时的核心代码的耗时。从中可以看出，同样使用 16 个 OpenMP 线程时 SGI 上的性能较优，这是由于其 CPU 相较于 Pi 超级计算机计算节点的 CPU 更先进。然而，如 4.2 节讨论过的原因，在使用更大规模的线程时，其性能反而有所下降。

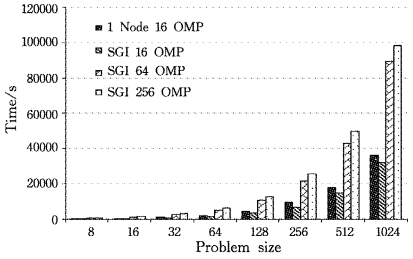

图 6　核心代码在 SGI UV 2000 上的耗时

可以看出，在采用相同数量的 CPU 核的情况下，相较于纯 OpenMP 版本直接在 SGI UV 2000 上的运行结果，UPC＋OpenMP 两层并行实现方案的性能更优，这主要归功于更好的局部性以及更低的 OpenMP 开销。

结束语　HGGF 算法对于天文学领域的星系组形成与演变的研究具有重要意义，然而由于其 OpenMP 实现受限于单节点资源，因此该算法的利用价值无法被充分发掘。为了更快地对更大规模的分组问题进行处理，需要对其实施进一步的优化与改进。实验结果表明，具备大规模线程与内存的 CSM 机器由于其 NUMA 结构特性的限制，对于类似 HGGF 算法这种受限于随机内存访问的程序很难提供较高的加速比，因此我们希望采用新的算法设计，利用多机并行来有效地解决此类问题。实现这一目标的一大障碍是程序中存在半随机性远程星系数据访问。为了解决这一难题，我们探索了隐式与显式数据分布，并通过实验发现后者更适合此类应用，同时我们提出了邻接星系列表设计来进一步减少远端内存访问的需求。实验结果表明，文中的 UPC＋OpenMP 两层并行设计可以有效加速算法并减少对单节点的内存需求。虽然 UPC 为程序开发者提供了一个简单易用的共享内存编程模型，但实验结果表明有必要对 UPC 程序进行手动调优。对此，采取包括大块内存传输在内的优化手段，以取得理想的性能。

下一步我们希望针对空间划分做出更深入的研究与优化，例如通过评估数据密度使得节点之间的负载更加均衡，从而进一步提升执行效率。

致谢　感谢杨小虎教授在算法优化方面提供的指导以及 SGI UV 2000 计算机硬件支持。

参 考 文 献

[1] KWON Y C,DYLAN N,JEFFREY G,et al. Scalable clustering algorithm for N-body simulations in a shared-nothing cluster [M]// Scientific and Statistical Database Management. Springer Berlin Heidelberg,2010:132-150.

[2] YANG X H,MO H J,VAN DEN B,et al. A halo-based galaxy group finder: calibration and application to the 2dFGRS[J]. Monthly Notices of the Royal Astronomical Society,2005,356(4):1293-1307.

[3] YANG X H,MO H J,VAN DEN B,et al. Galaxy groups in the SDSS DR4. I. The catalog and basic properties[J]. The Astrophysical Journal,2007,671(1):153.

[4] WILLIAM C,et al. UPC Language Specifications Version 1.3 [OL]. https://upc-lang.org/assets/Uploads/spec/upc-lang-spec-1.3.pdf.

[5] Intel® VTune® Amplifier XE[OL]. https://software.intel.com/en-us/intel-vtune-amplifier-xe.

[6] Technical Advances in the SGI® UV® Architecture[OL]. https://www.sgi.com/pdfs/4192.pdf.

[7] HAO H,SI Y M,WEI J W,et al. Optimizing Irregular Memory Access in Astrophysical Clustering Studies[J]. Journal of Frontiers of Computer Science and Technology,2017,11(1):80-90. (in Chinese)
郝赫,司雨蒙,韦建文,等. 天体物理成团研究中的非规则访存优化[J]. 计算机科学与探索,2017,11(1):80-90.

[8] MARK D,GEORGE E,CAROS F,et al. The evolution of large-scale structure in a universe dominated by cold dark matter[J]. The Astrophysical Journal,1985,292(2):371-394.

[9] NEAL K,LARS H,DAVID W. Galaxies and gas in a cold dark matter universe[J]. The Astrophysical Journal,1992,399(2):L109-L112.

[10] EISENSTEIN D J,HUT P. Hop:A new group-finding algorithm for n-body simulations[J]. The Astrophysical Journal,1998,498(1):137.

[11] LIU Y,LIAO W K,CHOUDHARY A. Design and evaluation of a parallel HOP clustering algorithm for cosmological simulation [C]// International Parallel and Distributed Processing Symposium,2003. IEEE,2003.

[12] FU B,REN K,LÓPEZ J,et al. DiscFinder:A data-intensive scalable cluster finder for astrophysics[C]// Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing. 2010:348-351.

(上接第 58 页)

[14] YE X C,LIN W,FAN D R,et al. Parallel optimization of protein sequence alignment algorithm in public nucleus structure[J]. Journal of Software,2010,21(12):3094-3105. (in Chinese)
叶笑春,林伟,范东睿,等. 蛋白质序列比对算法在众核结构上的并行优化[J]. 软件学报,2010,21(12):3094-3105.

[15] CAO Z Y,LANG X Y,LIU X,et al. Parallel optimization of super-large-scale sequence alignment calculation[J]. Journal of Computer Applications,2011,31(2):32-35. (in Chinese)
曹宗雁,郎显宇,刘昕,等. 超大规模序列比对计算的并行优化[J]. 计算机应用,2011,31(2):32-35.

[16] JIN X,LUO Z G,JIANG X Z,et al. Parallel Design and Implementation of Multi-Sequence Alignment Software T_Coffee[J]. Computer Applications and Software,2008,25(4):221-223. (in Chinese)
靳新,骆志刚,蒋晓舟,等. 多序列比对软件 T_Coffee 的并行化设计与实现[J]. 计算机应用与软件,2008,25(4):221-223.

[17] WEI S F,LIU Y,JIANG C Y. Parallel Research on Multiple Sequence Alignment of Genetic Annealing Based on GPU[J]. Computer Engineering and Design,2014,35(4):1247-1252. (in Chinese)
韦树烽,刘羽,蒋财运. 基于 GPU 的遗传退火多序列比对并行研究[J]. 计算机工程与设计,2014,35(4):1247-1252.

[18] WANG W D,TANG W,DUAN B,et al. Study on Parallel Acceleration Technique of High-throughput Gene Sequences based on HASH index[J]. Journal of Computer Research and Development,2013,50(11):2463-2471. (in Chinese)
王文迪,汤文,段勃,等. 基于 Hash 索引的高通量基因序列比对并行加速技术研究[J]. 计算机研究与发展,2013,50(11):2463-2471.

[19] ZHANG L,CHAI H,WO L K,et al. Research on Biological Sequence Alignment Algorithm and Graphics Hardware Acceleration[J]. Chinese Journal of Biomedical Engineering,2011,30(6):853-858. (in Chinese)
张林,柴惠,沃立科,等. 生物序列比对算法与图形硬件加速研究[J]. 中国生物医学工程学报,2011,30(6):853-858.

[20] ZHU X Y,LI R F,LI K L,et al. Research on Parallel Processing of Biological Sequence Alignment Based on Heterogeneous System[J]. Computer Science,2015,42(11):390-399. (in Chinese)
朱香元,李仁发,李肯立,等. 基于异构系统的生物序列比对并行处理研究进展[J]. 计算机科学,2015,42(11):390-399.

[21] ABUÍN J M,PENA T F,PICHEL J C. PASTASpark:multiple sequence alignment meets Big Data[J]. Bioinformatics,2017.

[22] ZAMBRANO-VEGA C,NEBRO A J,GARCÍA-NIETO J,et al. M2Align:parallel multiple sequence alignment with a multi-objective metaheuristic[J]. Bioinformatics,2017.

[23] ZOU Q,HU Q,GUO M,et al. HAlign:Fast multiple similar DNA/RNA sequence alignment based on the centre star strategy[J]. Bioinformatics,2015,31(15):2475-2481.

[24] LI D P,JU Y,ZOU Q. Application in tumor research of multiple sequence star alignment method based on MapReduce[J]. Cancer Progress,2016,14(6):510-513. (in Chinese)
李大鹏,鞠颖,邹权. 基于 MapReduce 的多序列星比对方法在肿瘤研究中的应用[J]. 癌症进展,2016,14(6):510-513.

[25] ZOU Q,GUO M,WANG X,et al. An Algorithm for DNA Multiple Sequence Alignment Based on Center Star Method and Keyword Tree[J]. Acta Electronica Sinica,2009(37):1746-1750.

[26] CHEN X,WANG C,TANG S,et al. CMSA:a heterogeneous CPU/GPU computing system for multiple similar RNA/DNA sequence alignment[J]. Bmc Bioinformatics,2017,18(1):315.

[27] ZOU Q,SHAN X,JIANG Y. A Novel Center Star Multiple Sequence Alignment Algorithm Based on Affine Gap Penalty and K-Band[J]. Physics Procedia,2012,33:322-327.