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算法这样具有随机性内存访问、对于访存敏感的程序，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][9][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。该算法实现采用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]中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他们设计的dFoF算法进行了实现，并且在一个小型集群上对算法进行了测试。他们的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等人提出了DiscFinder算法，并利用Hadoop实现了该算法[12]。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，最终对分组结果进行合并。

我们的工作与上述工作的不同之处在于我们使用UPC作为我们的实现工具，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。

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。初始时各个星系被当作独立的星系组，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]。若候选星系满足一定条件，它就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为一种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，因此对于本地与外部NUMA域的内存存取存在时延差异。

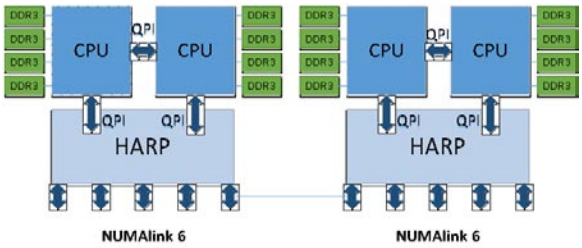

图2 SGI UV 2000体系结构

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，这是由于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。基于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460GB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以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。

4.2 OpenMP版在SGI UV 2000上的性能特征

我们的测试结果表明，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。原因主要有三点。

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。然而随着线程数量增加，这些线程会运行于更多的CPU，而它们位于不同的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，因此这种跨域访问对于程序性能十分不利。

其次，由于该算法的特性，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上的开销随之增大，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。

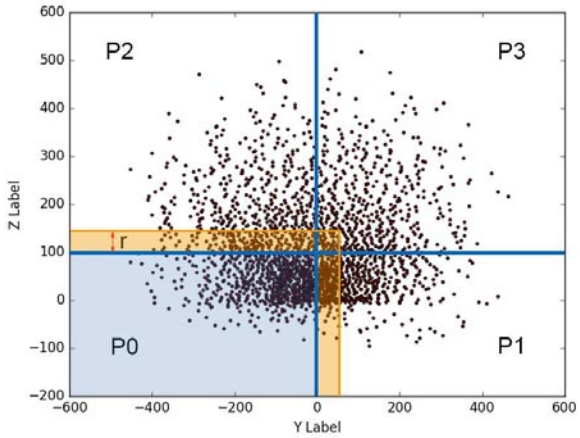

图3 星系域分解

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 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)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，对于原来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，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()函数从其他进程中进行大块内存读取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。我们使用GCC 5.2 作为C 编译器并使用Berkeley UPC和GUPC5.2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（单位：秒）

Nsample	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。我们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于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耗时对比 （单位：秒）

Nsample	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

表3 采用邻接星系列表与否耗时对比 （单位：秒）

Nsample	8	16	32	64
W/out AGL	12861.23	24254.31	47686.04	96131.80
With AGL	132.59	251.34	490.093	996.98

6.3 邻接星系列表设计有效性分析

表3对采用邻接星系列表设计方法与否的性能进行了对比，对于Nsample从8到64，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，每个进程则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。实验结果证明了我们设计的有效性。

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，对于单节点内存需求降低了67.5%，而使用8，16节点时，这一数值分别达到76.2% 与85.3%。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求解成为可能。

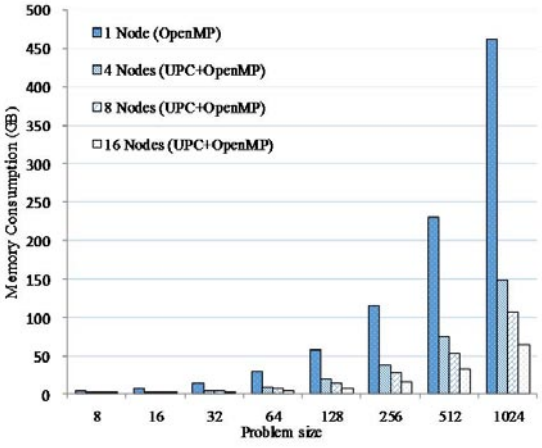

图4 不同节点数量下单点内存消耗

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进程数量增加，核心代码执行时间缩短。采用4，8，16进程时，核代码部分的平均加速比为2.25，2.78以及5.07。我们在进行区域划分时尽量使得各个节点内的星系数量均衡，但是由于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，造成加倍的节点数量无法成倍提升运

算速度。

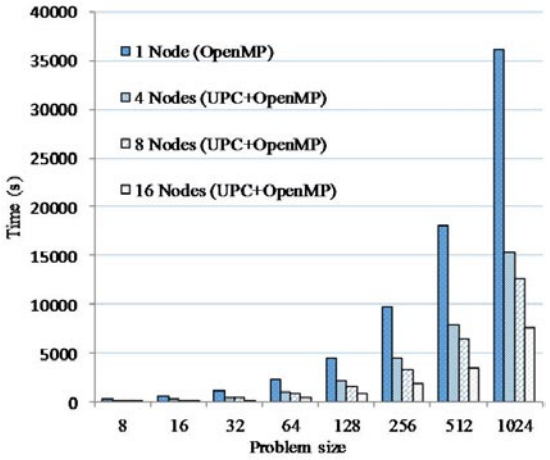

图5 核心代码在集群上的耗时

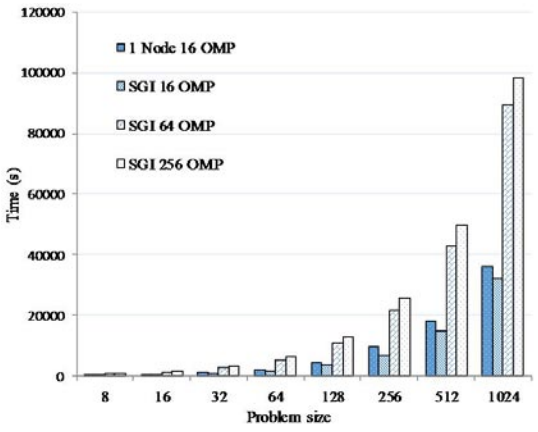

图6 核心代码在SGI UV 2000上的耗时

图6为SGI UV 2000上采用不同线程数量时的核代码耗时。如图所示，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更加先进。然而，由于在4.2节讨论过的原因，在使用更大规模线程时，其性能反而有所下降。

可以看出，在采用相同数量的CPU核的情况下，我们的UPC + OpenMP 两层并行实现方案相较于纯OpenMP版本直接在SGI UV 2000上的运行结果，性能更加优越。这是由于更好的局部性以及更低的OpenMP 开销造成的。

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，然而当前HGGF算法的OpenMP实现受限于单节点资源，无法充分发掘该算法的利用价值。为了更快地对更大规模的分组问题进行处理，我们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算法这样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，利用多机并行以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进一步减少了远端内存访问的需求。实验结果表明我们的UPC + OpenMP 两层并行设计可以有效加速算法并减少对于单节点的内存需求。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，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，进一步提升执行效率。

致谢

本文受国家重点研发计划（2016YFB0201400，2016YFB0201800）资助。林新华特别致谢日本学术振兴会JSPS的RONPAKU项目资助。同时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] YongChul Kwon, Nunley Dylan, Gardner Jeffrey, et al. Scalable clustering algorithm for N-body simulations in a shared-nothing cluster[C] //International Conference on Scientific and Statistical Database Management. Springer Berlin Heidelberg, 2010: 132-150.

[2] Yang Xiao-hu, Mo H J, Bosch Van Den, 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 Xiao-hu, Mo H J, Bosch Van Den, et al. Galaxy groups in the SDSS DR4. I. The catalog and basic properties[J]. The Astrophysical Journal: 2007, 671(1): 153.

[4] Carlson William 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:(https://software.intel.com/en - us/intel - vtune - amplifier - xe)
[6] Technical Advances in the SGI® UV™ Architecture(https://www.sgi.com/pdfs/4192.pdf)
[7] Hao He, Si Yu - meng, Wei Jian - wen et al. Optimizing Irregular Memory Access in Astrophysical Clustering Studies[J]. Journal of Frontiers of Computer Science and Technology, 2016. (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], 2016.
[8] Davis Mark, Efstathiou George, Frenk Caros, et al. The evolution of large - scale structure in a universe dominated by cold dark matter[J]. The Astrophysical Journal: 1985, 292: 371 - 394.
[9] Katz Neal, Hernquist Lars, Weinberg David. Galaxies and gas in a cold dark matter universe[J]. The Astrophysical Journal: 1992, 399: L109 - L112.
[10] Eisenstein Daniel J, Hut Piet. Hop: A new group - finding algorithm for n - bodysimulations[J]. The Astrophysical Journal: 1998, 498(1): 137.
[11] Liu Ying, Liao Wei - keng, Choudhary A. Design and evaluation of a parallel HOP clustering algorithm for cosmological simulation[C]//Parallel and Distributed Processing Symposium, 2003. Proceedings. International. IEEE, 2003: 8 pp.
[12] Fu Bin, Ren Kai, L ó pez Julio, et al. DiscFinder: A data - intensive scalable cluster finder for astrophysics[C] //InProceedings of the 19th ACM International Symposium on High Performance Distributed Computing, 2010: 348 - 351.

要闻集锦

IBM利用超级计算机和云计算普及数字技术

据www.zdnet.com网站2017年2月10日消息报道，近日，美国IBM公司与联合国开发计划署（UNDP）合作，发布了一项规模为7000万美元、持续5年的“数字非洲”计划，在全球那些面临大规模技能短缺以及失业问题的地区开展大规模数字培训计划。

“数字非洲”基于IBM的Bluemix云平台，将包含一系列面向不同受众的工具。最基本的是提供关于计算机和网络使用的课程，其中将包括关于网络安全和隐私的建议。对于那些希望拥有数字技能和创业能力的人来说，“数字非洲”将提供编程课程、推广应用程序的营销资料。

Watson超级计算机将用于根据学生所取得的进展和知识来制定个性化的学习计划，对接下来学习哪些课程给出建议。它还将收集“数字非洲”计划参与者的行为数据，以找到那些缺乏未来发展所需技能和知识的关键领域。

（李 苏）