阵列数据库系统 FASTDB 的研究与实现

邱能俊[1,2]，陈 梅[1,2]，李 晖[1,2+]，李宏源[1,2]，黄梦琳[3]，朱 明[3]

（1. 贵州大学 计算机科学与技术学院，贵州 贵阳 550025；2. 贵州大学 贵州省先进计算与医疗信息服务工程实验室，贵州 贵阳 550025；3. 中国科学院 国家天文台，北京 100012）

摘 要：为有效解决大规模科学数据的存储和分析问题，设计并实现一个分布式阵列数据库原型系统 FASTDB，优化大规模科学数据的存储和分析性能，单独分析用户自己上传的科学数据。为验证 FASTDB 的性能优势，设计一组真实的天文领域科学分析任务，将 FASTDB 系统与 SkyServer 系统进行实验比较，实验结果表明，FASTDB 系统在多数科学大数据分析场景下的性能远强于 SkyServer 系统。

关键词：科学数据；阵列数据库系统；科学分析；性能分析；大规模数据

中图法分类号：TP392 **文献标识号**：A **文章编号**：1000-7024（2016）04-1107-06

doi：10.16208/j.issn1000-7024.2016.04.050

Research and implementation of array database system FASTDB

QIU Neng-jun[1,2], CHEN Mei[1,2], LI Hui[1,2+], LI Hong-yuan[1,2], HUANG Meng-lin[3], ZHU Ming[3]

(1. College of Computer Science and Technology, Guizhou University, Guiyang 550025, China; 2. Guizhou Engineering Laboratory for Advanced Computing and Medical Information Services, Guizhou University, Guiyang 550025, China; 3. National Astronomical Observatories, Chinese Academy of Sciences, Beijing 100012, China)

Abstract: To solve the problem of storage and analysis of large scale scientific data efficiently, a distributed prototype database system called FASTDB was presented, which not only optimized the performance of massive scientific data, but also enabled users to easily upload their own scientific data into the array based scientific database engine for further distribution and parallel analysis. To verify the performance of FASTDB, some real scientific analysis tasks were presented. Results of comparison experiments of FASTDB and SkyServer show that FASTDB is far better than SkyServer in many typical analytical scenarios.

Key words: scientific data; FASTDB; scientific analysis; performance analysis; massive data.

0 引 言

科学领域每年都会产生海量的科学数据，大部分科学数据都是以数组形式存在，且科学分析过程通常比较复杂，常伴有矩阵计算（如矩阵相乘、求逆矩阵等）。传统基于关系型数据库的管理系统是以表作为数据模型，已经不能满足科学领域中大规模科学数据的存储和分析[1,2]。针对上述问题，本文设计并实现了一个基于阵列模型的数据库系统 FASTDB。

FASTDB 系统采用 Shared-nothing 无共享架构，以阵列作为一等公民以支持多维数组模型，同时利用 KVM 虚拟化技术提供云分析服务[3-7]。FASTDB 包含了几个关键模块：即基于阵列的存储分析引擎完成科学数据的存储和分析、分布式监控模块进行集群状态监控并收集系统的各种性能信息、数据处理引擎对科学数据进行复杂分析、数据上传和转换。FASTDB 的主要设计目标是让科学家们通过

收稿日期：2015-04-23；**修订日期**：2015-07-12

基金项目：国家自然科学基金项目（61462012）；贵州大学研究生创新基金项目（研理工 2014010）；贵州大学人才引进基金项目（700246003301）；贵州省科学技术基金项目（黔科合 J 字［2013］2099）；贵州省工业攻关基金项目（黔科合 GY 字［2014］3018）；贵州省应用基础研究重大项目子课题基金项目（黔科合 JZ 字［2014］2001-05）；贵州省发改委高技术产业发展专项基金项目（黔发高改计［2013］2069）

作者简介：邱能俊（1989-），男，福建龙岩人，硕士，CCF 会员，研究方向为分布式数据库、云计算；陈梅（1964-），女，贵州都匀人，教授，研究方向为数据库新技术；+通讯作者：李晖（1982-），男，湖南衡阳人，博士，研究方向为大规模数据管理与分析、高性能数据库；李宏源（1987-），男，山东泰安人，硕士，研究方向为数据库与科学工作流；黄梦琳（1987-），女，安徽蚌埠人，硕士，研究方向为天文数据管理与分析；朱明（1966-），男，北京人，研究员，研究方向为射电天文学。E-mail: huili_gm@gmail.com

这个系统在节省昂贵成本的同时获得高性能的云分析服务。

1 FASTDB 设计与实现

FASTDB 系统能够支持科学数据存储、分析以及各种系统资源的监控。它利用科学数据库 SciDB 作为存储和分析引擎。用户可以通过 WEB 接口输入需要的查询分析语句执行得到分析结果。FASTDB 的资源监控模块能够监控系统执行分析过程中的各种性能信息，例如 CPU 状态、内存利用率和磁盘 I/O 等。

1.1 FASTDB 系统架构

FASTDB 是一个能够分析和管理大规模科学数据的分布式系统。它采用无共享架构，由数据处理子系统，分布式监控子系统和存储和分析子系统组成。数据处理子系统能够执行用户输入的查询语句得到分析结果；资源监控子系统能够提供实时的节点状态信息；存储和分析子系统支持科学数据到 SciDB 存储引擎的无缝对接，即提供科学数据的存储和分析功能。FASTDB 的架构如图 1 所示。FASTDB 有两种类型的节点：协调节点和工作节点，协调节点为单一节点，通过 PostgreSQL 管理工作节点的信息。工作节点负责处理从协调节点分发的子计划，同时存储数据块。通过把 FASTDB 部署在基于 KVM 的云环境中，能够充分利用基础设施资源，方便提供科学大数据的云分析服务。

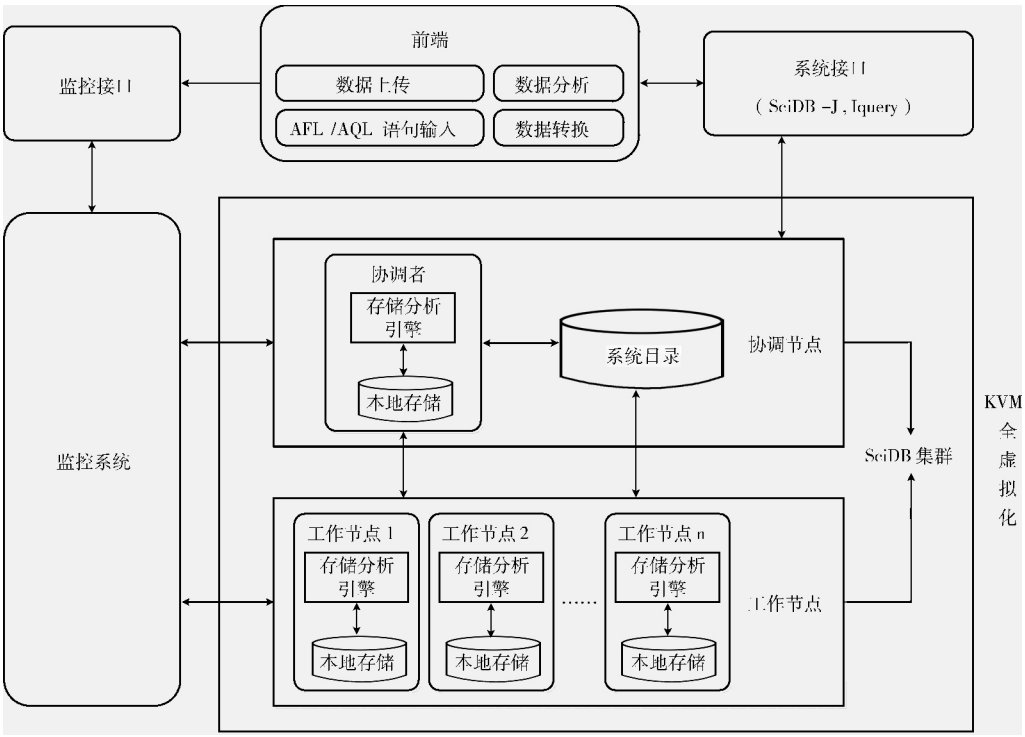

图 1 基于 KVM 全虚拟化的 FASTDB 基础架构

FASTDB 系统采用 SciDB 作为后端存储和分析引擎。SciDB-J 和 Iquery 作为系统接口运行用户提交的查询分析语句。用户通过前端上传自己的私有数据以后，FASTDB 将把这部分数据进行预处理并且转换为 CSV 的数据格式，最后存储到 SciDB 引擎中[8]。此后，数据处理子系统将从存储和分析子系统中取得数据进行并行分析，最终返回处理结果给用户。此外，系统所有的状态信息如 CPU、内存、磁盘 I/O 的都通过监控子系统收集以作性能测试和问题跟踪。

1.2 科学数据处理和分析

当前的大数据处理和分析有多种方案，如基于传统关系型数据库的 SkyServer 系统；基于高性能，分布式，可扩展，高可靠的 key/value 存储系统 Tair；重点面向行业大数据的 MPP 新型数据库集群等。FASTDB 能够快速响应用户提交的科学分析请求，它会把用户上传的科学数据分布式存储在集群的每个节点上。在用户通过 FASTDB 上传科学数据以后，SciDB 引擎会把载入的数据块 Chunk 采用数据块分布算法分布在集群的所有数据库实例上，每个数据块实例只存储部分数据片，这些数据片的大小以及位置的元数据信息会被保存在系统目录中，保证数据的可用性。FASTDB 包含了大量的数组相关算法，方便进行数组操作，如聚集，矩阵相乘、转置、求逆等。方便进行科学数据的分析操作。集群节点的每个数据库实例只对本地存储的数据进行分析。

图 2 展示了 FASTDB 存储和分析子系统的架构，从图中可知，FASTDB 的存储和分析引擎是基于科学数据库

SciDB 的，FASTDB 协调节点通过 JDBC 与 Web 前端通信，在获取到查询分析任务以后，将任务分解成多个子任务；接着根据系统目录中记录的集群信息找到各个工作节点，利用消息和数据传输通道把子任务分发到各个工作节点；FASTDB 工作节点上的分析引擎对子任务进行处理，并把相应的中间结果存储在本地存储器；当 FASTDB 协调节点发送搜集结果的消息时，各工作节点把本地的子结果返回，最终由协调节点对结果进行合并返回给 Web 前端，展示给用户。

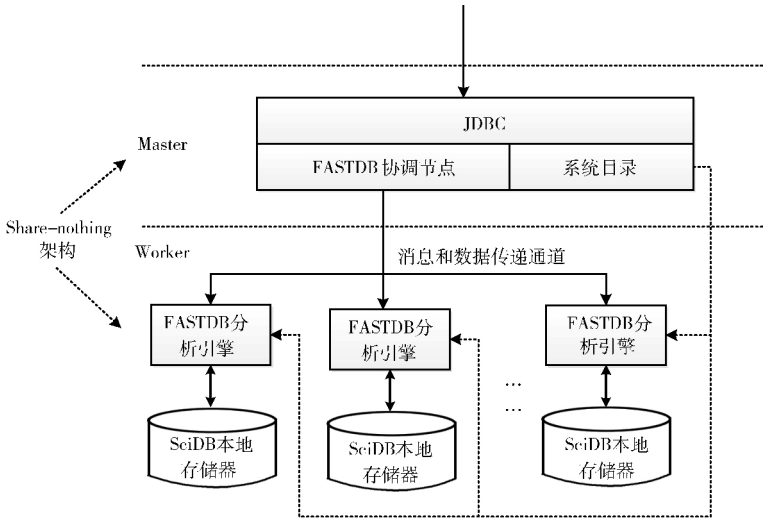

图 2　FASTDB 的存储和分析子系统

1.3　数据监控

分布式监控子系统能够改善 FASTDB 系统的服务质量。例如，当用户收到监控子系统收到警告信息时，能够根据监控子系统反馈的信息，及时调整分析任务使之达到性能最优化。

FASTDB 的监控子系统包含监控客户端和监控服务端两个组件。FASTDB 系统的每个节点都要安装和配置监控客户端。监控服务端可以按预先定义的规则对收集到的监控信息进行处理，同时，允许用户配置事件报警。所有的监控信息包括静态信息和配置信息都将被展示在基于 WEB 的面板上，通过监控面板，用户可清楚的了解系统当前状态情况。监控子系统的架构被设计成服务端—代理端—客户端架构，其中间层是个代理。代理层只会接受服务端的配置信息，再定时将数据传送给服务端，代理层本地只保存最近没有发送的数据，此外，所有配置将在服务端进行。

2　实验设计

为了验证 FASTDB 针对科学数据存储和分析的有效性和可行性，本小节将 FASTDB 系统和广泛使用的基于传统关系数据库的 SkyServer 系统进行对比实验。SkyServer 系统为斯隆数字巡天的数据仓库，它采用 SQLServer 数据库管理和分析科学数据。

2.1　实验环境

实验所使用的是第九版本的斯隆数字巡天数据集(SDSS DR9)。SDSS DR9 是斯隆数字巡天观测 25% 的天空，获取到的一百多万个天体的多色测光数据和光谱数据，数据总量约为 58 TB。

实验的所有集群节点采用的是 Intel(R) Xeon(R) E5-2620 2.00GHz 双 CPU，FASTDB 单协调节点的配置为 40 GB 的内存和 1 TB 的硬盘空间，15 个工作节点的配置为 8GB 的内存和 1TB 的硬盘空间。存储分析引擎分别为 SciDB 14.3 和 Microsoft SQL Server 2008 R2。为了结果的可重复性，SciDB 14.3 和 Microsoft SQL Server 2008 R2 均采用默认配置。

2.2　实验设计

为评估 SkyServer 和 FASTDB 的性能，设计了 3 种不同类型的天文分析任务。3 种类型的分析任务总共包含 8 个查询分析语句 (Q1-Q8)[8]。Q1 到 Q5 属于第一种类型的天文分析，该类语句可归为“SELECT * FROM * WHERE *”形式。Q6 属于第二种类型的天文分析任务，该类语句可以归为“SELECT * FROM * JOIN * ON * WHERE *”形式。第三种类型的天文分析任务可以归为“SELECT * FROM * AS * JOIN * ON * AS * JOIN * ON * WHERE * AND *”形式，Q7、Q8 就是属于这类分析。此外，我们通过 SDSS DR9 随机抽取了 5 种不同大小的数据集进行实验，具体数据大小和记录数见表 1。

表 1　不同大小数据集的记录数

数据大小	1 GB	10 GB	20 GB	50 GB	100 GB
记录条数	80 000	800 000	1 600 000	4 800 000	8 000 000

为了消除其它因素的影响，实验环境采用冷启动的方式，并且清除系统缓存[9]。

3 实验结果与性能分析

图 3（a）到 3（h）清楚的表明 SkyServer 和 FASTDB 在 5 种不同数据集上执行 8 个分析任务的性能对比结果。横坐标代表的是不同的数据大小，纵坐标表示完成各个分析任务所消耗的时间。FASTDB 运行 Q6、Q7 和 Q8 的响应时间超过了规定的有效时间阈值（7200 s），因此我们忽略超过的这部分时间区间。如图 3（f）到 3（h）所示。

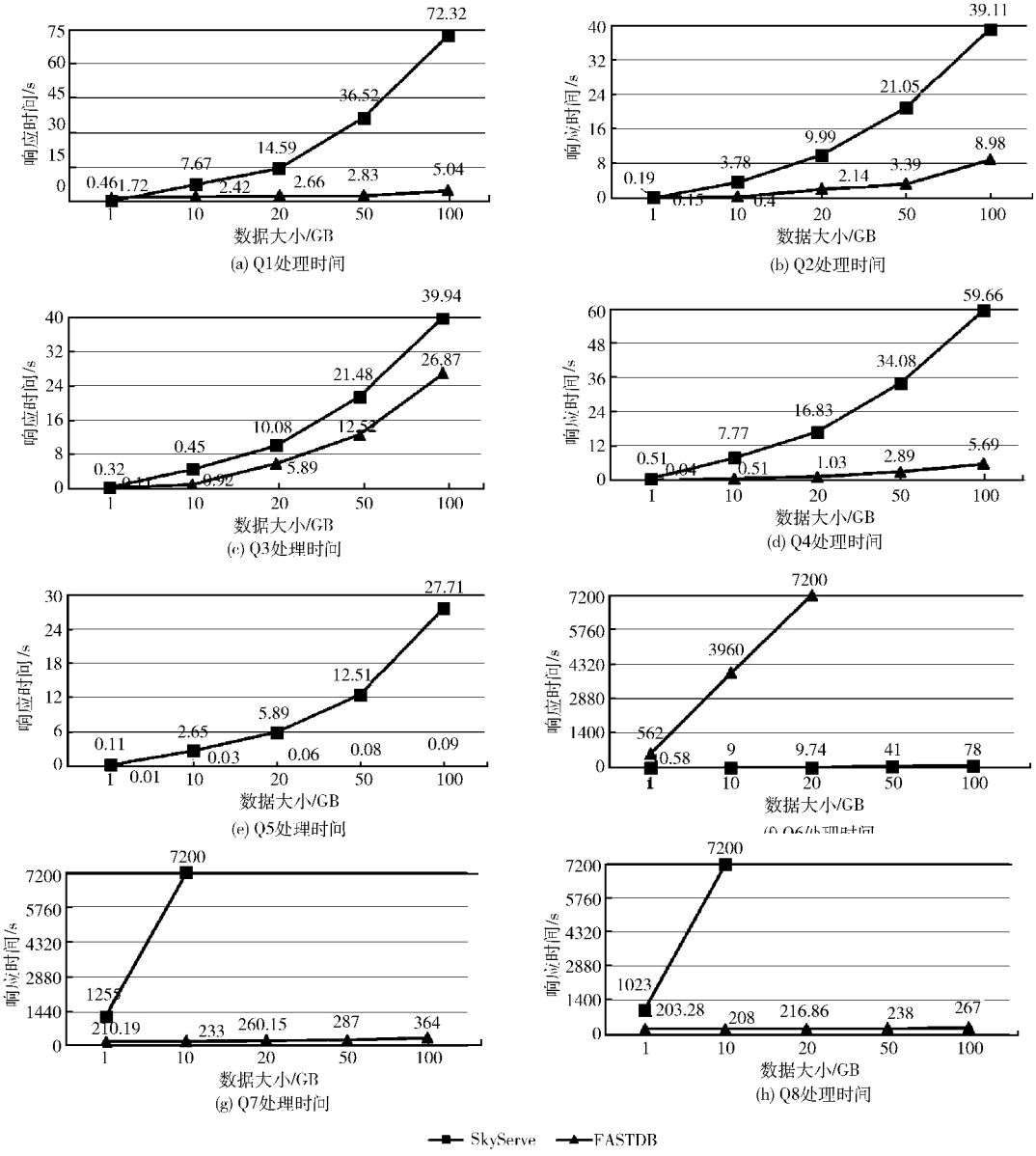

图 3 FASTDB 和 SkyServer 在不同大小数据集上的总体性能表现

从实验结果可以看出，FASTDB 执行 Q1-Q5 这 5 个查询分析时，FASTDB 总的性能表现比 SkyServer 高两个数量级，且随着数据的增大，FASTDB 的性能表现越来越好。FASTDB 出色的性能表现归结于两个原因，首先，FASTDB 使用数组作为一等公民，它所支持的多维数组模型能够满足大部分科学领域中的数据存储模型。实际上，大部分的科学分析任务都会用到数组操作，比如矩阵求逆等。因此 FASTDB 的数据模型特性很好的支持了科学数据的分析。其次，FASTDB 对存储的数据进行了压缩并且分割存储到每个工作节点中。这种机制加快了查询分析的速度，让数据以更小的粒度进行网络传输，也减少了网络开销。

Q6、Q7、Q8 分析任务都有一个共同的特征，即都包含带 Join 的分析语句。目前，分布式的科学数据库并不能很好的支持 Join 操作，因为 Join 操作需要进行频繁的网络数据包传输，在带宽有限的情况下，大量的数据 Chunk 传输容易形成网络拥堵。对很多分布式数据库来说，如果发生网络的堵塞，查询分析任务往往不能被有效的执行[10]。此外，FASTDB 的数据压缩机制也给 Join 查询带来一部分

的解压开销，数据 Chunk 的传递过程中是被压缩的，在进行分析时需要对压缩的 Chunk 进行解压缩，这个过程会增加额外的 CPU 成本，导致执行 Join 查询语句时性能的下降。更进一步分析，Join 操作需要在协调节点进行大量的计算，随着数据集的增大这种情形往往增加了内存负担。

并行机制也是 FASTDB 性能表现优异的一个原因。FASTDB 把载入的科学数据分布到每个节点并行的执行分析语句。相反，SkyServer 不能进行扩展，也无法并行的执行分析，虽说其有很好的索引机制，能够加快查询语句的执行速度（FASTDB 并没有完整的索引），但是科学领域更加注重数据的分析而并非简单的检索功能，因此针对科学领域，FASTDB 的性能表现远远优于 SkyServer。为了更详细地对比 FASTDB 和 SkyServer 的性能表现，选择 Q5 在不同系统中的响应时间换算成不同的单位得到结果如图 4 所示。

图 4（a）是 SkyServer 执行 Q5 的响应时间随数据量变化图，图 4（b）是 FASTSDB 执行 Q5 的响应时间随数据量变化图。随着数据集的增大，FASTDB 和 SkyServer 执行 Q5 所需的时间开销也随之增大，然而，数据集从 20 GB 增加到 50 GB 时，两者都有一个明显的性能转折，FASTDB 在此区间的性能变化相对 SkyServer 系统较小是因为其数据被分布到所有节点中进行并行的分析处理，而且返回的结果的数据量达不到网络瓶颈，因而查询速度快，同时 FASTDB 在把数据分发给每个 Worker 节点时，也对数据做了压缩处理。因此其性能比 SkyServer 提升了 10 到 30 倍左右。

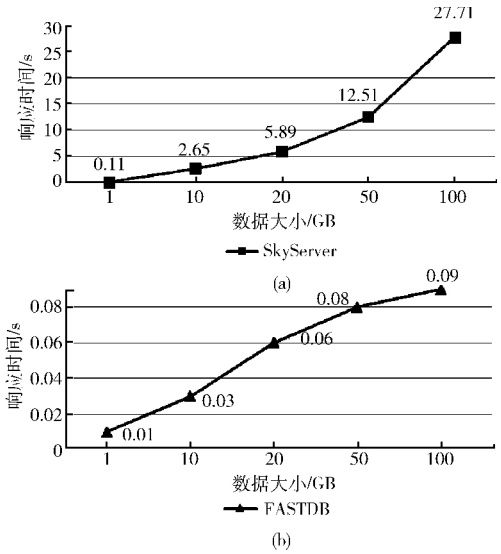

图 4　FASTDB 和 SkyServer 分别在 Q5 上的表现性能

FASTDB 的性能表现如此好并不是全得益于并行机制。因此，我们对基于单 SciDB 节点的 FASTDB single 系统和 SkyServer 系统进行了另外一组实验来对比分析其性能，见表 2。结果显示 FASTDB single 在 Q1-Q5 的查询中比 SkyServer 快至少两倍以上，尤其随着数据集的增大，FASTDB single 系统的性能表现越来越好。因为基于 SciDB 的 FASTDB single 系统是采用基于数组的垂直存储模型，可以读取更少字节获得 Chunk 数据，同时数据的紧密堆积也方便快速的获取数据。另一方面，在数据集达到 100 GB 的时候，FASTDB single 在 Q2，Q3，Q4 上的性能表现并不让人满意，这是为了减少磁盘开销，FASTDB single 采用了 Chunk 压缩机制，数据在被载入的存储引擎时已经被压缩了。当它们需要被访问时从磁盘读取入内存中进行解压缩处理，压缩和解压缩操作会造成一定的 CPU 开销，随着数据集的增大，这部分开销占的比重也随之增大造成性能总体下降。在科学领域，科学分析的复杂性决定了很多分析操作都要进行大量的计算，因此在越复杂的科学分析中压缩和解压缩所造成的时间开销并不是主要的性能影响因素了。表 2 中 timeout 代表查询处理时间超过了 7200 s。FASTDB single 在小数据量进行 Join 分析的时候用了 1900 s，相反，SkyServer 在 1 s 以内就完成了查询。而对于多 Join 的语句以及随着数据量的增大，FASTDB single 的性能表现为 timeout。造成 FASTDB single 性能如此之差的原因在于 Q6-Q8 都是 CPU 密集型的操作，同时 SciDB 并没有对 Join 操作有很好的支持。

表 2　FASTDB single 和 SkyServer 的性能对比（timeout 代表查询时间超过 7200 s）

系统	数据集	查询响应时间/s							
		Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8
SkyServer	1 GB	0.46	0.19	0.32	0.51	0.11	0.58	210	203.28
	10 GB	7.67	3.78	4.55	7.77	2.65	9.01	233	208
	20 GB	14.59	9.99	10.08	16.83	5.89	9.74	260	216.86
	50 GB	36.52	21.05	21.48	34.08	12.51	41	287	238
	100 GB	72.32	39.11	39.94	59.66	27.71	78	364	267
FASTDB single	1GB	0.76	0.72	0.65	0.99	0.89	1900	timeout	timeout
	10 GB	1.02	1.94	2.89	3.96	2.26	timeout	timeout	timeout
	20 GB	1.16	4.23	7.34	5.69	2.71	timeout	timeout	timeout
	50 GB	1.36	8.02	15.56	10.21	3.85	timeout	timeout	timeout
	100 GB	3.11	24.45	31.11	19.59	5.12	timeout	timeout	timeout

4 结束语

FASTDB 系统的设计目标是实现对大规模科学数据的存储和分析。它包含了几个关键模块：即基于 SciDB 的存储分析引擎、分布式监控模块和数据处理引擎。为了进一步验证 FASTDB 的性能，将 FASTDB 与基于 SQLServer 的 SkyServer 系统进行对比，实验选取了 8 个具有代表性的真实的天文分析任务。分析实验结果可知，FASTDB 系统在传统的科学数据分析中有很好的性能表现，它能获得比基于传统关系数据库的 SkyServer 系统 10 倍以上的性能。

在下一步计划中，我们将对 FASTDB 进行以下改进：①实现额外的科学分析工具以供科学家们更方便的处理数据，比如科学工作流插件，可以帮助科学家们免去繁琐的以编程实现功能的方式。②把科学数据分析功能以云服务的形式提供给大众。③优化 FASTDB 系统的分布式 Join 性能，使之能够支持更为复杂分析任务，比如多 Join 查询等。④改进查询优化器，提升查询分析的性能。

参考文献：

[1] Stonebraker M, Becla J, Dewitt D, et al. Requirements for science data bases and SciDB [C] //Conference on Innovative Data Systems Research Perspectives, 2009: 7-16.

[2] Soroush E, Balazinska M, Wang D. ArrayStore: A storage manager for complex parallel array processing [C] //ACM SIGMOD International Conference on Management of data, 2011: 253-264.

[3] Seering A, Cudre-Mauroux P, Madden S, et al. Efficient versioning for scientific array databases [C] //International Conference on Data Engineering, 2012: 1013-1024.

[4] Hey T, Tansley S, Tolle K. The fourth paradigm: Data-intensive scientific discoveries [M]. Microsoft Research, 2009: 153-164.

[5] Cornacchia R, Héman S, Zukowski M, et al. Flexible and efficient IR using array databases [J]. VLDB Journal, Specialissue on IR&-DB Integration, 2008, 17 (1): 151-168.

[6] Kersten M, Zhang Y, Ivanova M, et al. Sciql, a querylanguage for science applications [C] //Proceedings of the EDBT/ICDT Workshop on Array Databases, 2011: 1-12.

[7] Seo S, Edward J, Yoon, et al. HAMA: An efficient matrix computation with the MapReduce framework [C] //IEEE Cloud Com, 2010: 721-726.

[8] Sample astronomy query [EB/OL]: http: //cas. sdss. org/dr7/en/help/docs/realquery. asp.

[9] Stonebraker M, Brown P, Poliakov A, et al. The architecture of SciDB [C] //Scientific and Statistical Database Management, 2011: 1-16.

[10] Cudre-Mauroux P, Kimura H, Cudre-Mauroux P, et al. A demonstration of SciDB: A science-oriented DBMS [C] //Very Large Data Bases, 2009: 1534-1537.

（上接第 1070 页）

[10] YANG Hai, WANG Hongguo, HOU Lunan, et al. Application of chaos ant colony optimization in the intelligent transportation system and its algorithm [J]. Journal of Chengdu University: Natural and Science, 2008, 26 (4): 309-312 (in Chinese). [杨海，王洪国，侯鲁男，等. 混沌蚁群算法及其在智能交通中的应用 [J]. 成都大学学报：自然科学版，2008，26 (4)：309-312.]

[11] LIU Xiaoying, CAI Zixing, YU Lingli, et al. An orthogonal-cluster chaos ant colony algorithm based on swarm-robots system mission planning application research [J]. Journal of Chinese Computer Systems, 2010, 31 (1): 164-168 (in Chinese). [刘晓莹，蔡自兴，余伶俐，等. 一种正交混沌蚁群算法在群机器人任务规划中的应用研究 [J]. 小型微型计算机系统，2010，31 (1)：164-168.]

[12] ZHOU Yongquan, HUANG Zhengxin. Artificial glowworm swarm optimization algorithm for TSP [J]. Control and Decision, 2012, 27 (12): 1816-1821 (in Chinese). [周永权，黄正新. 求解 TSP 的人工萤火虫群优化算法 [J]. 控制与决策，2012，27 (12)：1816-1821.]