计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2017.20160837
54(2):267-283, 2017

遥感大数据的基础设施:集成、管理与按需服务

李国庆[1]　　黄震春[2]
[1](中国科学院遥感与数字地球研究所　北京　100094)
[2](清华大学计算机科学与技术系　北京　100084)
(ligq@radi.ac.cn)

Data Infrastructure for Remote Sensing Big Data: Integration, Management and On-Demand Service

Li Guoqing[1] and Huang Zhenchun[2]
[1](*Institute of Remote Sensing and Digital Earth, Chinese Academy of Sciences, Beijing* 100094)
[2](*Department of Computer Science and Technology, Tsinghua University, Beijing* 100084)

Abstract The increasing growth of remote sensing data and geoscience research pushes earth sciences strongly and poses great challenges to data infrastructures for remote sensing big data, including the collection, storage, management, analysis and delivery. The de-fact remote sensing data infrastructures become bottleneck of the workflows for remote sensing data analysis because of their capability, scalability and performance. In this paper, data infrastructures for remote sensing big data are catalogued into 6 classes based on the features such as basic service unit, distributivity, heterogeneous, space-time continuation and on-demand processing. Then, architectures are designed for all the 6 classes of data infrastructures, and some implementation technologies such as data collection and integration, data storage and management, data service interface, and on-demand data processing, are discussed. With the architecture designs and implementation technologies, data infrastructures for remote sensing big data will provide PaaS (platform-as-a-service) and SaaS (software-as-a-service) services for developing much more remote sensing data analysis applications. With continuously growing data, tools and libraries in the infrastructures, users can easily develop analysis models to process remote sensing big data, create new applications based on these models, and exchange their knowledge each other by sharing models.

Key words data infrastructure; remote sensing big data; on-demand processing; data integration; data management

摘　要 随着遥感技术的不断进步,遥感数据的数据量越来越大,种类越来越多,分布越来越分散,遥感应用的复杂程度和个性化程度也不断提高,遥感正在走向大数据时代.而目前遥感数据基础设施在容量、可扩展性、易用性和性能等方面都难以满足遥感应用的需求,成为了遥感科学与工程从获取到最终产品这个流程中的瓶颈.为此,首先从遥感数据的本质出发,讨论了遥感数据基础设施应当具备的分布、异构、时空连续和按需数据处理等特性,并依据遥感数据基础设施的基本服务单元、分布性、时空连续性

收稿日期:2016-11-15;**修回日期**:2016-12-27
基金项目:国家重点研发计划项目(2016YFB0501504);海南省重大科技计划项目(ZDKJ2016021)
This work was supported by the National Key Research and Development Program of China (2016YFB0501504), and the Grant of Hainan Provincial Department of Science and Technology (ZDKJ2016021).
通信作者:黄震春(huangzc@tsinghua.edu.cn)

和按需处理支持能力将遥感数据基础设施分成 6 类. 其次,针对这 6 类遥感数据基础设施展现出的特性,设计了实现这些基础设施可以采用的体系结构,并指出了其中实现的技术难点和解决思路. 最后,就遥感数据基础设施设计和实现过程中涉及到的数据收集与整合、数据组织与管理、数据服务接口、按需数据处理等方面的技术方案进行了深入的讨论. 在这些技术的支持下,遥感数据基础设施能够做到分布化、智能化和平台化,支持遥感科学的合作研究和工程上的协同应用.

关键词 数据基础设施;遥感大数据;按需处理;数据集成;数据管理

中图法分类号 TP315

随着遥感技术的不断发展进步,新的遥感卫星不断发射,新的传感器不断投入使用,遥感数据的时空分辨率越来越高,遥感数据的种类和数量也在不断增加. 例如,2015 年欧空局(Europe Space Agency, ESA)存档的遥感数据量就已经达到了 1.5PB,而且这个数字还在不断增加. 与此同时,计算能力的不断提升使得全球范围的复杂模拟已经成为可能. 这种数据和计算能力上的双重进步为遥感科学的发展提供了更好的条件,也为遥感科学基于"第四范式"进行基于大数据分析的研究提供了可能[1].

大数据,指无法在一定时间范围内用常规软件工具进行获取、管理和分析处理的数据集合,其主要特点可以用 5V 来描述,包括 Volume(数据量巨大)、Velocity(高速积累)、Variety(种类多样)、Value(价值密度低)、Veracity(数据质量难以保证)[2]. 这些特性大多数能够在遥感数据中得到体现[3]. 首先,在数据量巨大之外,各种历史和技术原因导致遥感数据还具有非常明显的异构特性:不同传感器、不同卫星平台、不同数据格式、不同存档结构等. 同时,遥感数据经常需要被高速处理,尤其是那些诸如灾害预警和响应、全球制图、高分辨率实时反演这样的实时或准实时遥感应用. 所以,遥感数据被认为是一种比较典型的大数据,遥感数据分析应用也被视为一种大数据应用,解决遥感数据的收集、整理、管理、存储、分析、处理、分发等过程中所遇到的一系列技术难题,为地学研究工作者提供遥感大数据的管理与按需处理服务,对于基于遥感数据的地学研究具有非常重要的意义.

随着遥感数据在种类和数量上的不断增加,以及遥感应用场景的不断复杂化和个性化,遥感数据基础设施在存储容量、访问时间、可扩展性、数据安全性、数据服务质量等方面面临着巨大的挑战. 一方面,在传统的政府主导的公益应用中,数据时空分辨率的不断提高带来了数据量和计算量的急剧增加,面向国家、区域和国际的环境变化、社会、经济等重大问题以及面向灾害和灾难等严重突发事件的各种遥感数据应用不断给遥感数据存储服务系统提出更高的要求. 卫星、无人机、地面传感器的观测能力也已经达到 PB 量级,基本具备了逐日的米级观测的能力. 在这种情况下,以集中式离线和准在线方式进行遥感数据存储与服务的传统数据基础设施已经远远无法满足当前与未来遥感数据存储、共享与应用的高速、海量、安全、稳定和合理成本等要求. 数据基础设施已经成为遥感数据应用这个木桶中的低木板.

另一方面,在地理信息系统和定位导航等技术逐渐商业化公众化之后,遥感数据观测与服务也正在从政府主导的公益应用向公众与商业机构主导的商业与公众应用扩展. 商业与公众应用逐渐成为遥感数据观测与服务的主要用户. 以 Google 为例,2004 年收购 Keyhole 公司并推出了谷歌地球(Google Earth),2008 年收购了 ImageAmerica 并开始启动专用卫星系列的发射,2014 年收购了微小卫星云厂商 SkyBox,形成了自己强大的全球亚米级高分辨观测能力. 除此之外,一些具有强烈个性化色彩的遥感数据观测应用也不断出现,例如,期货公司通过沙特储油罐连续图像的反演来估算沙特的石油产量进而估计欧佩克的产量,从而为期货贸易提供指导,通过卫星影像跟踪进出富士康的货车数量来估算 Apple 产品产量和半导体市场价格等.

以 Google 为代表的业界巨头们的一系列动作代表了公众和商业机构拓展遥感数据应用市场的决心,同时也预示着遥感数据基础设施将必须为适应这些商业与公众应用的需求而做出相应的变化. 除了和公益应用类似的高速、海量、安全、稳定和合理成本等要求之外,商业与公众应用的多样化特性也给遥感数据基础设施提出了具有强烈个性化色彩的多样化的遥感数据存储服务需求,传统的以收集数据为主要目标的、离线低速和手段单一的数据基础设施远远无法满足公众的遥感数据处理需求对数据存储和管理的更高更个性化的要求. 地球观测的

公众化不仅在工程层面对于地球观测的传感器、平台、通讯、地面单元和应用模式提出了巨大的挑战,也在科学层面对地球观测的基本理论和模型提出了新目标,进而在管理层面对于地球观测的政策、管理和可持续发展模式都提出了新的问题.遥感数据基础设施必须针对这些挑战从遥感数据的认识、组织、管理、存储、分发、处理等诸多方面予以回应.

针对这些挑战,学术界和业界已经在数据的组织、管理与服务等方面做出了一些努力,开放地理空间协会(Open Geospatial Consortium, OGC)在数据管理与服务方面的系列标准就是这些努力中影响比较大的一个. Web Coverage Service, Web Feature Service, Web Map Service, Catalogue Service, Web Map Tile Service, OpenSearch Geo 等组成了通过 HTTP 实现数据共享和服务的一组标准化共享协议,用以支持不同组织不同类型的空间观测数据的有序共享和发布[4].世界上最主要的对地观测政府间合作组织地球预测组织(Group on Earth Observations, GEO)在其 2005—2015 十年规划中所建立的 Global Earth Observation System of Systems (GEOSS)试图通过在多个机构组成的联邦内部制定或采用标准的协议进行观测数据与模型数据的共享服务与访问,并通过 Clearinghouse、Registries、处理服务、工作流等实现数据之间的有机协调,将联邦内部各机构所提供的数据组织到一起,形成全局数据视图,最终达到协同解决科学与现实问题的目标[5].目前,GEOSS 的数据基础设施已经支持了灾害、全球变化、水、农业、能源等 9 个领域的应用.

除此之外,在不断发展的信息科学技术的推动下,网格、云等各种现代分布式计算的研究成果也不断被应用到遥感数据基础设施之中.例如,Cossu 等人在 ESA 的网格项目支持下建立的用于高速遥感数据共享与处理的机遇网格的遥感数据服务环境 G-POD[6],GEO 的基于网格的数据共享的探索与实践[7],Li 等人在空间信息网格(SIG)的支持下建立的跨组织共享和分发空间数据的新型数据服务系统[8],Huang 在网格和数据密集型计算技术支持下基于按需数据服务和元数据适配器构建全局统一视图的新型遥感数据基础设施的探索[9].随着云计算技术的不断发展,云存储和 MapReduce 计算模型等的成熟,虽然云计算在性能和安全性等方面还存在一些不足[10],但其优秀的并行处理能力和可管理特性使得云计算技术在遥感数据管理方面得到越来越多的应用.其中最有代表性的事件当属 NASA 将其部分数据服务转移到亚马逊的 AWS 平台之上,并结合亚马逊提供的计算资源和工作流管理工具建立了 NASA NEX 平台[11],用来提供数据服务、高性能计算服务和科学工作流支持.

但是,目前的遥感数据基础设施研究大多数还处于将已有的遥感数据通过标准的接口进行共享和分发的阶段,其解决的问题是如何将已有的已经按照传统的离线、低速的管理模式进行存档和管理的数据以标准化的方式进行共享,并没有涉及到提高数据存储能力、降低数据访问时间、提高数据服务质量等遥感数据基础设施迫切需要解决的技术问题,对于个性化、高维度、时空连续的数据服务需求所需要的对遥感数据在组织理论和管理方法上的变革更是没有涉及.

本文首先从遥感数据的本质出发,讨论了地学科学研究对遥感数据集成与服务的需求,并根据所提供服务能力的不同对遥感数据基础设施予以分类;然后,针对各种不同类型的遥感数据基础设施,本文给出了可行的体系结构,并针对服务实现过程中会遇到的一系列问题给出了解决方案;最后,本文还对遥感数据基础设施的未来发展提出了展望.

1 遥感数据与遥感数据基础设施

在传统的遥感数据基础设施(或者称为遥感数据服务)中,数据被认为是一些互相孤立的实体,这些实体使用元数据说明其空间位置、描述时间、光谱、波段、专题等特性,并以一定的格式(一般以文件的形式)存储、分发和处理,如图 1 所示.在使用时,应用通常通过某种接口对存储在空间机构的元数据进行查询,在已有的数据中找到合适的数据集并筛选出需要的数据实体,再经过一个数据准备阶段之后,将按照一定格式打包的空间数据下载到本地,供下一步的处理和可视化工作使用.

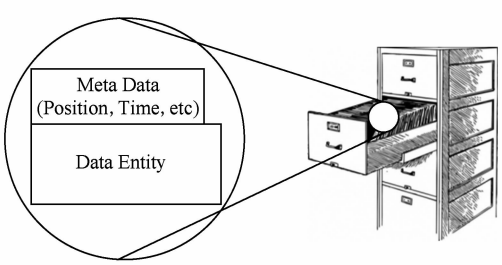

Fig. 1 Classical storage and management of remote sensing data

图 1 传统的遥感数据存储与管理方法

这种遥感数据的存储与管理体系简单、实用、易于实现,是当前遥感数据存储与管理最常见的方法.但是,这种存储与管理体系存在着很多弱点,比如不能反映遥感数据的本质、只能简单地将已有的数据予以共享而无法实现数据的按需服务、从不同数据中心获取数据可能需要使用不同的接口、数据的访问时延比较长等.

从本质上来讲,各种遥感数据都是客观世界的映像,反映了客观世界中某时空区间内的某种特性.因此,遥感数据应当具有用来表征空间(位置)、时间和被观测特性的3类维度.空间包括3个维度,用来标识数据反映的客观世界的位置,时间有1个维度,用来说明数据反映的客观世界的某个特定时刻(或时间区间),被观测特性可以包括传感器类型、波段类型、专题类型等多个可能的维度,这些特性可能具有一定的相关性,需要在分析其相关性的基础上最终归纳为几个确定的维度.

对于某个确定的时间、确定的空间位置和确定的被观测特性,客观世界一定有一个确定的值与之对应.例如,在2016年2月12日北京时间2:10:10(确定的时间),我的办公室窗外(确定的位置:40°00′08.0″N,116°20′01.4″E,60m.a.s.l)的气温(确定的被观测特性)是2.1℃(一个确定的值),窗前是一片落叶的树木(地表分类这个被观测特性为某个确定值).

在确定了空间、时间和被观测特性3类维度之后,客观世界就能够被映射到由这3类维度组成的一个连续的高维数据空间之中,这个高维数据空间中的每一个点都对应于客观世界某位置某时间某被观测特性的观测值.通过在这个高维连续数据空间中的各个维度定位,我们就可以寻找到需要的对客观世界的观测结果.这个过程可以由一个多元函数来表示:

$$f(latitude, longitude, altitude, time, characteristics) \rightarrow value.$$

例如,我们前面给出的例子就可以表示为:

$$f(40.0022, 116.3336, 60, 1455214210000, \text{“temperature”}) \rightarrow 2.1.$$

既然遥感数据的本质是这样一个多元函数,那么针对这个函数最常见的2个操作 *getValue* 和 *findArea* 就应该是遥感数据服务的基本操作.其中,*getValue* 用来根据指定的特定时间、位置和被观测特性取得一个确定的整数、浮点数或者字符串等数据类型的观测值.其函数原型可以描述为:

value *getValue*(*pos*,*time*,*characteristics*);

或者批量读取观测值以取得更好的服务性能:

valueArray *getValues*(*area*,*timeSpan*,*characteristicsList*);

反之,从指定的空间区域中查找符合某种条件(例如地表气温高于零度)的子区域也是非常常用的一个操作:

area *findArea*(*area*,*condition*).

不考虑服务性能,*getValue* 和 *findArea* 两个操作足以满足各种遥感应用对数据基础设施的需求.在这2个操作中,*getValue* 更加适用于栅格数据,而 *findArea* 更加适用于矢量数据.

总的来讲,遥感数据基础设施面临着如下3种主要挑战:

1) 分布与异构.遥感数据通常由遍布全球的各个空间机构所拥有和保存,这些数据在数据格式、管理方法、访问方法和数据政策等方面体现出明显的异构性.考虑到遥感数据通常具有非常大的数据量,通过广域网传输遥感数据将带来非常大额时间开销,所以将这些分布于全球不同空间机构的异构遥感数据组织在一起成为一个有机的整体并提供数据服务将是一件非常复杂的事情.

2) 数据缺失.理想情况下,遥感数据应当覆盖全球任何时间、任何地点和任何观测特性.但是对全球任何时间、任何地点和任何观测特性进行观测并将得到的遥感数据存储起来和提供服务是不现实的,也是没有必要的.从数据空间来看,从各种传感器和遥感数据分析模型得到的遥感数据实际上只是高维的遥感数据空间中的一些碎片(如图2所示),理想的遥感数据基础设施应当能够通过插值或反演等数据处理的手段以这些数据碎片为基础重建出

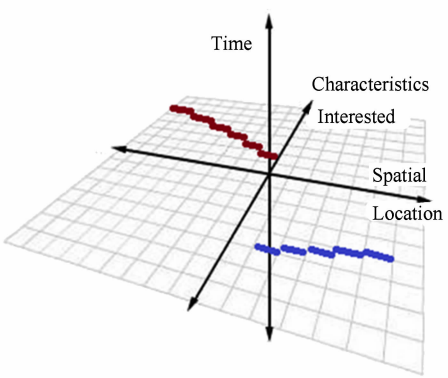

Fig. 2 Available data fragment in the high dimensional data space

图2 高维遥感数据空间中的已有数据碎片

整个数据空间中的所有数据,从而达到弥补缺失数据的目的.

但是,这种重建也受到很多限制.首先,由于插值和反演等遥感数据处理算法的缺乏,并不是所有的缺失数据都能够通过插值或反演等遥感数据算法得到重建.其次,由于插值和反演等遥感数据处理无论在处理模型上还是在处理精度上都存在不确定性,通过插值和反演等数据处理得到重建的缺失数据可能在精度和准确程度上有着一定的不足.最后,插值和反演算法需要的巨大计算量也对遥感数据基础设施提出了更高的计算能力需求.

3) 服务效率.由于遥感数据的数据量大,处理方法多样且复杂,遥感数据基础设施对于高性能、高吞吐率和低成本的要求非常迫切.同时,越来越多的全球范围的数据共享、覆盖全球范围的遥感应用以及跨越地域和组织的协同研究使得高带宽和低时延的遥感数据传输也显得非常重要.

在目前遥感数据的使用模式中,遥感数据的处理通常是在用户所在的遥感数据处理工作站上使用用户提供的处理程序完成的,这就需要将原始的遥感数据从数据提供者通过广域网传输到用户的遥感数据处理工作站上.由于遥感数据的数据量通常比较大,通过广域网络传输遥感数据所需要的时间也比较长,这就带来了一些额外的时间开销.因此,在遥感数据提供者一端提供按需处理的能力,允许用户将他们的数据处理模型提交到服务端运行,直接得到他们需要的处理结果,对于减少额外的数据传输开销,提高工作效率有着很大的意义.

除此之外,遥感数据基础设施还面临着一系列其他挑战,如服务性能、可管理性和数据质量控制等.其中,数据质量管理对遥感数据基础设施非常重要.遥感数据基础设施中的数据质量问题可以被简化地视为服务所提供的数据与被观测实体实际数据之间的差异,它可能来自于传感器的误差、数据处理过程中的人为失误或者计算模型带来的系统误差等.用户反馈与数据溯源是被用来解决数据质量问题的常用手段.根据用户对数据基础设施所提供数据的主观评价,数据基础设施可以根据对原始数据和数据处理模型进行溯源,并更进一步对于存储的数据和处理模型的可信性和可靠性进行进一步的评价.

作为一种通过虚拟化实现动态可伸缩的大规模计算资源、存储、平台、软件的管理并通过因特网为用户提供服务的技术手段[12],云计算可以被认为是一种解决服务性能和可管理性的可行方案.同时,指令级并行、线程级并行、集群计算、分布式计算等各种高性能计算方面的技术,以及高性能网络传输方面的研究成果都能够被用于提高遥感数据基础设施的性能.

总结起来,理想的遥感数据基础设施应当具有如下特征:1)以有语义特征(如波段、区域)的遥感数据对象而不是原始的遥感数据文件为基本服务对象;2)将分布异构的遥感数据集成到一起并提供统一的服务接口;3)支持通过插值或反演等方法生成缺失的遥感数据;4)实现按需的遥感数据处理以支持用户上载其数据处理模型在数据端按需处理遥感数据.根据遥感数据基础设施的基本服务单元、分布性、时空连续性、按需处理服务能力等特性,遥感数据基础设施可以被分为6类,如表1所示:

Table 1 Classes of Remote Sensing Data Infrastructures

表1 遥感数据基础设施的分类

Class	Basic Service Unit	Distributed	Missing Data Completion	On-Demand Processing
Ⅰ	Image File	No	No	No
Ⅱ	Image File	Yes	No	No
Ⅲ	Data Object	No	No	No
Ⅳ	Data Object	Yes	No	No
Ⅴ	Data Object	Yes	Yes	No
Ⅵ	Data Object	Yes	Yes	Yes

由于实现简单,第Ⅰ类遥感数据基础设施是目前最为常见的集中式遥感数据服务.这类遥感数据服务通常只能以景或条带为单位提供存储在一个单一的存储系统(文件系统或数据库)中的遥感数据,这些遥感数据被用户以其原始的影像文件的形式下载使用.而第Ⅱ类遥感数据基础设施目前存在少量可用的实例(如"综合定量遥感产品服务规范与运营系统"中的"分布式卫星数据服务系统").它们在第Ⅰ类的基础上增加了分布性和异构性,允许将多个地理上分布的异构遥感数据源通过一系列的中间件整合为一个整体提供服务,但其服务基本单元依旧是以景或条带为单位的原始的影像文件.

第Ⅲ~Ⅵ类遥感数据基础设施通过遥感数据的剖分与拼接,试图实现更加本质化的遥感数据服务接口,即以遥感数据对象(如波段、Bounding Box等)为基本服务单元,根据用户的需求将预先剖分好的数据拼接为用户需要的遥感数据对象,并返回给用户.目前有一些基于地理信息系统(geographic information system, GIS)的遥感数据服务实例可以

被视为第Ⅲ类遥感数据基础设施,OGC 的 WCS 和 WFS 是这些服务常见的服务接口.其余几类遥感数据基础设施由于其复杂性目前尚无广泛使用的实例.其中,第Ⅳ类遥感数据基础设施在第Ⅲ类的基础上增加了分布和异构特性,第Ⅴ类则以差值和反演等方法在第Ⅳ类的基础上增加了对遥感数据基础设施的时空连续性的支持,第Ⅵ类基础设施则通过对遥感数据数据分析处理模型的统一描述与调度在第Ⅴ类基础设施的基础上增加了对按需服务的支持,允许用户将其处理分析模型上传到服务平台对遥感数据进行处理并得到处理结果,是我们所期待的理想的遥感数据基础设施. OGC 的 WCPS(Web coverage process service)定义了一系列用于对 Coverage 进行按需处理服务的 WCS 扩展[13],为实现第Ⅴ类和第Ⅵ类基础设施进行了一些非常有益的探索.

2 体系结构设计

由于复杂程度的不同,不同类型的遥感数据基础设施需要不同的体系结构设计.对于第Ⅰ类遥感数据基础设施来讲,图 3 所示的体系结构就足以满足要求.在该体系结构中,用于说明遥感数据获取时间、覆盖范围、卫星与传感器等特性的元数据被存储在数据库中,用户可以通过一个图形用户界面或者基于 Web 的用户界面对这些元数据进行查询,根据诸如覆盖范围、获取时间、卫星与传感器名等条件找到合适的遥感影像,并通过 HTTP 或 FTP 等协议以文件的形式下载这些遥感影像用于进一步的分析处理.

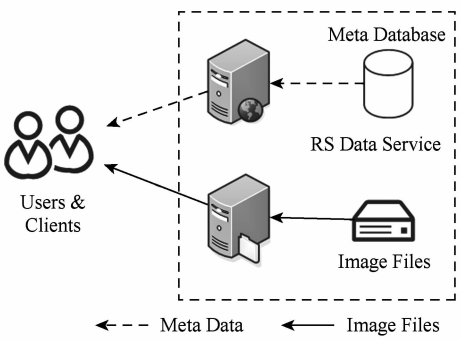

Fig. 3 Architecture for data infrastructures in class Ⅰ

图 3 第Ⅰ类遥感数据基础设施的体系结构

在这个过程中,第Ⅰ类遥感数据基础设施会为用户提供诸如 *query*(*area*, *timeSpan*, *characteristicsList*)和 *access*(*dataID*)的 API 用于遥感影像数据的查询与下载.其中,*query* 用于从元数据库中根据数据覆盖范围、获取时间、观测特性(卫星、传感器、观测主题)等条件查询元数据,并将查找到的元数据返回给用户以供选择.在用户得到这些返回结果之后,即可从中选取合适的数据,使用其数据 ID 作为参数调用 *access*,启动数据获取过程,请求数据源准备对应的影像数据并将其以 HTTP 或 FTP 服务的形式提供下载.

图 4 中所展示的 2 种体系结构都可以将图 3 所示的第Ⅰ类遥感数据基础设施增加分布和异构特性,从而形成第Ⅱ类遥感数据基础设施.由于遥感数据文件数据量巨大,数据传输的时间开销比较大,所以遥感数据文件通常被用户直接下载以避免多次数据传输带来额外的开销.

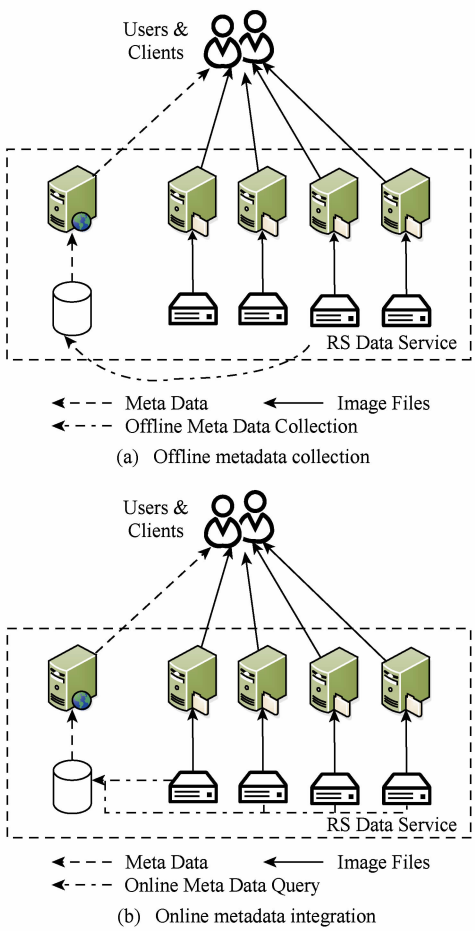

Fig. 4 Architecture for data infrastructures in class Ⅱ

图 4 第Ⅱ类遥感数据基础设施的体系结构

分布式的遥感数据基础设施主要解决的是元数据的收集、整理、组织和集成服务问题.根据元数据集成方式的不同,第Ⅱ类数据基础设施可以有“在线”和“离线”2 种实现方式.“离线”方式周期性地从数据提供者将元数据收集到一起,形成一个定期更

新的中央元数据库(这个中央元数据可能是分布可扩展的),并依靠此中央元数据库为用户提供元数据的查询服务(图 4(a)).“离线”方式服务性能较高,但元数据的定时收集使得数据基础设施在数据提供者的元数据修改时会出现部分的元数据不一致.例如,在元数据被收集之前,数据提供者最新获取的遥感影像数据无法被用户查询使用.

与之相反,“在线”方式并不保存和维护中央元数据库,而是通过“在线”翻译和转发用户对元数据的查询请求和数据源的查询结果的方式来形成一个虚拟的中央元数据库,从而达到将分布式异构数据组织到一起提供一站式服务的目的(图 4(b)).“在线”方式不会出现“离线”方式有可能出现的元数据不一致的现象,但是由于用户的每次查询请求都会被转发到所有数据源,查询的效率较低,对数据源的压力也比较大.

考虑到遥感数据一旦存档就不应当再被修改和删除的特性,“在线”方式和“离线”方式可以被结合起来使用,通过“离线”方式中周期性更新的中央元数据库来提供高性能的查询服务,并减少对数据源的查询请求;同时对于那些尚未得到收集的元数据以“在线”的方式进行实时的转发,以保证元数据查询结果的全局一致.

无论“离线”方式还是“在线”方式,第Ⅱ类遥感数据基础设施通常提供和第Ⅰ类遥感数据基础设施类似的 *query*()和 *access*()等 API 用于遥感影像数据的查询与下载.由于各遥感数据源通常使用不同的协议提供其元数据的查询服务,所以第Ⅱ类数据基础设施必须使用诸如元数据适配器等方式将这些遥感数据源之间的异构性予以消除,以达到一站式服务的目的.

在第Ⅰ类和第Ⅱ类数据基础设施中,遥感数据以原始的影像文件的形式存储和提供服务.在这种情况下,获取某特定区域或波段的遥感数据就必须要经过 2 步:1)从遥感数据基础设施中找到并下载包含指定区域的遥感影像文件;2)从下载的遥感影像文件中抽取出需要的区域或者波段的数据.这时,第 1 步下载到的数据中就存在无用的部分,也就是说出现了冗余的数据传输.例如,在运行全球干旱指数分析时,只有 Band2,Band6 和 BandState 三个波段被用于 NDVI 算法分析,这就意味着下载的 MOD09 数据的 13 个波段中超过 3/4 的数据是无用的.这种无用的数据传输带来了额外的网络传输开销,降低了遥感数据处理过程的效率.

为了避免第Ⅰ类和第Ⅱ类数据基础设施中这种额外开销,其他几类数据基础设施不再使用原始的影像文件作为基本的数据服务单元,而是直接给用户提供合适的遥感数据对象(如某个区域某个传感器的遥感数据).为了达到此目的,这些遥感数据基础设施首先必须针对不同的分辨率(如 10 km、1 km、100 m 等)定义一套网格分划体系,将整个地球划分为一系列的基本网格.原始的遥感数据将根据定义的分划体系被分割为一系列分片数据,并存储在数据库或文件系统中.当用户发起服务请求时,这些分片数据将被根据用户给定的需求拼接,形成遥感数据对象交给用户.

为了达到以遥感数据对象为基本服务单元的目的,第Ⅲ类和第Ⅳ类遥感数据基础设施应当采用图 5 所示的基本体系结构.原始的遥感数据应当按照基本网格分割为分片数据后再存储,这就使得数据分割和数据拼接成为第Ⅲ类及以上的遥感数据基础设施必须具备的功能.

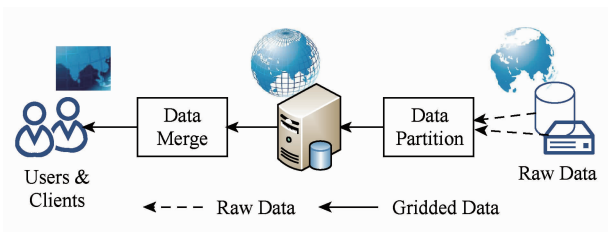

Fig. 5 Architecture for data infrastructures in class Ⅲ and Ⅳ

图 5 第Ⅲ类和第Ⅳ类遥感数据基础设施的体系结构

由于其数据地理上的分布性,第Ⅳ类遥感数据基础设施中数据分割和拼接的实现要比第Ⅲ类基础设施复杂.首先,数据的分割要在原始数据存储的节点“附近”进行,从而避免大量原始数据通过相对来讲带宽较低延迟较高的广域网络传输,提高数据分割的效率;其次,数据基础设施必须从各个分布于广域网上的不同存储节点收集分片数据,这种跨越广域网的数据收集会给数据基础设施带来额外的数据传输开销.因此,在第Ⅳ类遥感数据基础设施中,如何对分片数据进行合理的冗余复制和存储优化,以使得遥感数据基础设施在使用这些分片数据时能够在比较“近”的存储节点上获得,从而减少分片数据传输所需要的时间开销,将是对提高基础设施的数据服务性能非常有意义的一个问题.

另外,第Ⅲ类及以上的遥感数据基础设施应当在提供的 API 上也与第Ⅰ类和第Ⅱ类基础设施有所不同.由于第Ⅲ类及以上的遥感数据基础设施以

遥感数据对象为基本服务单元,那么这些服务提供第1节中所列出的 *getValue*()和 *findArea*()这些操作作为其编程接口就显得非常自然了.在这些编程接口的支持下,用户(或客户端)可以根据覆盖范围、数据采集时间、观测要素等条件直接获得需要的数据.和第Ⅰ、Ⅱ类服务所提供的查询-获取-下载的服务方式相比,这种服务方式更加简洁、方便,同时也避免了冗余数据传输带来的额外开销.

和前几类服务相比,第Ⅴ和第Ⅵ类基础设施更加依赖于高性能按需计算.在第Ⅴ类基础设施中,当用户发起的数据请求由于数据缺失而无法完成服务时,数据基础设施不再是简单地返回一个代表"数据不存在"的说明信息,而是试图找到合适的相关数据,通过预先部署好的插值或反演算法从这些存在的相关数据中将用户需要的数据重建出来,并返回给用户,如图6所示:

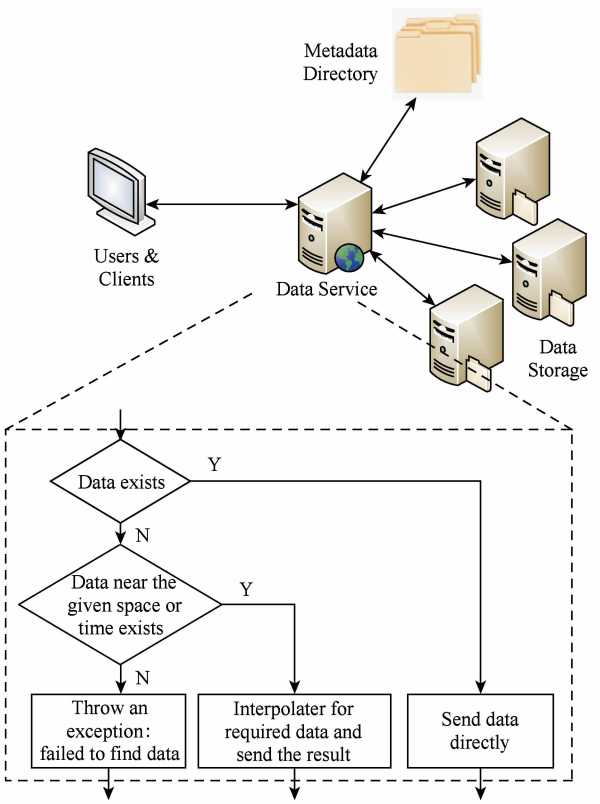

Fig. 6 Architecture for data infrastructures in class Ⅴ

图6 第Ⅴ类遥感数据基础设施的体系结构

由于将任何时间任何地点任何观测目标的数据一律予以存储是不可能的,也是没有必要的,因此,遥感数据基础设施中的数据缺失问题也是不可避免的,第Ⅴ类数据基础设施中对缺失数据的重建对于遥感数据基础设施的使用者来讲有着非常重要的意义.这种对缺失数据的重建则依赖于那些预先部署好的插值或反演算法,算法越丰富和精确,数据基础设施对确实数据的填补能力也越强,填补得到的数据也越准确.

在填补缺失数据的同时,第Ⅴ类数据基础设施的数据重建也会带来一系列其他问题.首先是数据的可信性.由于数据插值和反演算法不可避免地存在各种系统误差,这些算法得到的数据和客观世界的真实数据不可避免地存在差异.为了使用户能够更加准确地了解这些差异,方便用户对数据质量进行更加精细的控制,第Ⅴ类数据基础设施必须为用户提供详细的数据溯源信息,包括原始数据的来源和插值/反演算法的实现方法等.在这个过程中,数据溯源模型(data provenance models)可以被用于对数据的可信性和质量进行跟踪.

另外,插值和反演算法一般计算量都比较大,对于计算资源的要求也比较高,这就使得第Ⅴ类数据基础设施通常需要比较强大的计算资源支持.与之类似,对插值和反演算法的优化对于降低插值和反演带来的开销、提高数据服务效率也会有一定的帮助.另外,更好的调度算法可以寻找距离已有数据更"近"、性能更高的节点进行数据插值或反演处理,从而降低数据传输和处理的时间开销,达到提高数据基础设施整体性能的目的.

更进一步,当数据基础设施中配置了大量的通用计算资源,而且允许用户将遥感数据分析模型以源代码等形式上传到这些计算节点上时,用户就可以直接在这些计算节点上根据需要对遥感数据基础设施中存储的数据进行处理,并将计算结果展示给用户.在这种情况下,用户就无需再将原始数据下载到本地的工作站进行数据处理,从而避免了大量的原始数据在网络上传输所带来的时间开销.这就是提供按需数据处理能力的第Ⅵ类数据基础设施.配合一定的数据分析模型管理服务,用户可以使用第Ⅵ类数据基础设施提供的数据服务能力和数据处理能力,根据其需求灵活地处理各种数据,开发各种遥感应用,实现科学和工程目标.

3 实 现

遥感数据基础设施的实现依赖于一系列关键技术上的突破,例如数据组织与管理、数据收集与整合、按需数据处理等.

3.1 数据基础设施中的数据收集与整合

在数据基础设施中,为了提供一站式的遥感数据服务,来自不同空间机构的异构遥感数据,至少其

元数据必须被收集到一起并加以整合.由于不同空间机构提供遥感数据及其元数据的方式不同,收集这些数据与元数据的方法也随之不同,如使用FTP,HTTP或其他网络协议下载元数据文件,对元数据库进行复制,使用预定义的网络协议对元数据进行查询,通过网络爬虫从空间机构提供的Web站点爬取元数据等.

在这些方法中,使用FTP、HTTP或其他网络协议下载元数据文件通常需要对元数据文件进行进一步的解析和整理,而且获得这些元数据文件的难度也比较大;元数据库的复制通常受限于数据政策而无法实现;使用网络爬虫爬取元数据的技术难度较大,而且容易对空间机构的网站造成比较大的压力.所以,使用预定义的网络协议对元数据进行查询和获取是一种比较灵活、常用和有效的方法.

由于Web的普遍使用,基于HTTP协议的请求-响应模式在网络应用中非常流行.元数据查询与获取的协议也通常基于HTTP和请求-响应模式,以SOAP或RESTful Web服务的形式实现.这些协议通常包括查询和获取2类功能.其中,查询功能根据给定条件查询元数据库,并将查询结果以预先定义的格式予以返回.由于其可扩展性和灵活性,查询结果通常基于XML或JSON来定义.查询和获取通常使用异步访问模型,例如一个基于SOAP的元数据查询协议可能包括如下3个分别用来启动一个查询、获取查询结果和关闭一个查询的操作:

启动查询:String *query*(String *request*);

获取结果:String *getDescriptions*(String *queryID*, int *size*);

关闭查询:void *closeQuery*(String *queryID*).

在这些元数据查询操作里,*query*接收一个查询条件字符串*request*,解析这个查询条件,启动一个新的查询,并将查询的ID作为响应立即返回给用户.查询成功启动后,*getDescriptions*使用这个查询ID作为参数,以XML串的形式读取满足条件的"下*size*个元数据项",直到所有元数据都被读取或者使用*closeQuery*关闭查询为止.

*query*的参数*request*以XML格式组织描述,例如一个用于查询Landsat7卫星、ETM+传感器、覆盖北京地区(115.7°E~117.4°E,39.4°N~41.6°N)、采集于2010年最后10 d的查询串应当如图7所示.

如图7所示,参数*request*主要包括3类条件:第1类是以〈operator〉,〈attr〉和〈value〉三个元素组成的"简单比较"条件,用于对卫星名、传感器名、观测时间、影像云量等观测参数进行相等、不等、大于、小于、相似于等简单比较;第2类是以包含一组经纬度最大最小值的〈bbox〉元素的"覆盖范围边界"条件,用于对遥感数据的覆盖范围进行查询;第3类条件为"逻辑复合条件",它使用AND,OR和NOT等逻辑运算连接包括其他"逻辑复合条件"在内的多个条件组成复杂的查询条件,以满足用户多样化的查询需求.

```
〈?xml version="1.0" encoding="UTF-8"?〉
〈request〉
  〈condition relation="AND"〉
    〈condition〉
      〈operator〉EQ〈/operator〉
      〈attr〉SENSOR_ID〈/attr〉
      〈value〉ETM+〈/value〉
    〈/condition〉
    〈condition〉
      〈operator〉EQ〈/operator〉
      〈attr〉SPACECRAFT_ID〈/attr〉
      〈value〉Landsat7〈/value〉
    〈/condition〉
    〈condition〉
      〈bbox〉115.7,39.4,117.4,41.6〈/bbox〉
    〈/condition〉
    〈condition〉
      〈operator〉GT〈/operator〉
      〈attr〉CREATE_TIME〈/attr〉
      〈value〉2010-12-22 00:00:00〈/value〉
    〈/condition〉
    〈condition〉
      〈operator〉LT〈/operator〉
      〈attr〉CREATE_TIME〈/attr〉
      〈value〉2010-12-31 23:59:59〈/value〉
    〈/condition〉
  〈/condition〉
〈/request〉
```

Fig. 7 Example for XML description of query conditions

图7 查询条件的XML描述示例

*query*会启动一个新的查询并将这个查询的*queryID*返回,这个*queryID*可以被用于调用*getDescriptions*,以从前向后依次遍历的方式返回查找到的元数据(如图8所示),也可以用于调用*closeQuery*关闭查询以释放查询占用的系统资源.

在作为结果的XML串中,查找到的元数据被组织为一个〈result〉元素的列表,其中每个〈result〉元素中包括一组〈field〉子元素.这些子元素都有一个名为"name"的属性,这个属性的取值和元素的值共同组成了一个键值对,用来说明元数据的某个特性,如卫星名、传感器名、获取时间、产品级别、数据格式、四角坐标等.由于〈field〉子元素的数量不受限制,而且其"name"属性的属性值也不受限制,所以

这种基于 XML 的描述方法具有足够的扩展性,足以用于描述来自不同空间机构的不同卫星和传感器的遥感元数据.

```
<?xml version="1.0" encoding="UTF-8"?>
<result_list>
  <result>
    <field name="id">
      LSAT_LE71220322010365EDC00</field>
    <field name="CREATE_TIME">
      Fri Dec 31 00:00:00 CST 2010</field>
    <field name="SPACECRAFT_ID">
      LANDSAT7</field>
    <field name="SENSOR_ID">ETM+</field>
    <field name="ROW">32</field>
    <field name="PATH">122</field>
    <field name="UL_LON">117.40405</field>
    <field name="UL_LAT">41.28446</field>
    <field name="UR_LON">119.64354</field>
    <field name="UR_LAT">40.95797</field>
    <field name="LL_LON">116.91972</field>
    <field name="LL_LAT">39.682</field>
    <field name="LR_LON">119.10786</field>
    <field name="LR_LAT">39.36316</field>
    <field name="RESOLUTION">30</field>
    <field name="STATION">EDC</field>
    <field name="CLOULD_COVER">0.0221</field>
    <field name="LEVEL">l2</field>
    <field name="OFFSET">260202</field>
    <field name="FORMAT">Geotiff</field>
    <field name="DATASIZE">228085541</field>
  </result>
  <result>
    ⋮
  </result>
</result_list>
```

Fig. 8 Example for XML description of query results

图 8 查询结果的 XML 描述示例

除了查询之外,基于 SOAP 同样也可以定义一个由启动获取过程、读取获取结果和结束获取过程 3 个操作组成的基于 XML 的异步遥感数据获取协议:

启动获取过程:String *access*(String *dataID*);

读取结果:String *getResult*(String *accessID*);

结束获取:void *closeAccess*(String *accessID*);

这组异步协议用来请求数据提供者根据给定的数据 ID 准备数据并使该数据上线,允许用户下载.数据 ID 通常情况下来自于前述查找协议返回的元数据.当 *access* 被调用时,数据源将启动一个过程,找到用户需要的数据,在必要时将数据打包并复制到用户可以下载的区域.同时,*access* 为获取过程分配一个 *accessID*,这个 *accessID* 可以在调用 *getResult* 和 *closeAccess* 时作为参数.当 getResult 被调用时,根据数据准备情况的不同,会有不同的结果 XML 串被返回,如图 9 所示:

```
<?xml version="1.0" encoding="UTF-8"?>
<result>
  <state>RUNNING</state>
</result>
```

(a) Still running

```
<?xml version="1.0" encoding="UTF-8"?>
<result>
  <state>ERROR</state>
  <caused>Data not found.</caused>
</result>
```

(b) Access failed

```
<?xml version="1.0" encoding="UTF-8"?>
<result>
  <state>OK</state>
  <url usage="package">
    ftp://use:pwd@xx.xx.xx.xx/download/xxx.tar.gz</url>
  <url usage="quickview">
    ftp://use:pwd@xx.xx.xx.xx/download/xxx.jpg</url>
</result>
```

(c) Access accomplished successfully

Fig. 9 Examples for XML description of access results

图 9 *getResult* 的返回结果 XML 串示例

根据数据准备情况的不同,结果 XML 串会包含取值为"RUNNING"、"ERROR"或者"OK"的<state>元素.图 9(a)中所示的"RUNNING"状态代表数据尚未准备完成,图 9(b)所示的"ERROR"状态代表数据准备过程中出现了异常,以至于无法提供数据下载,异常原因在<caused>元素中描述.如果数据准备完成,图 9(c)所示的结果字符串将被返回,其中一组 url 元素用于指出不同用途的数据对应的下载地址.最后,当数据下载完成后,用户应当调用 *closeAccess* 关闭获取过程,允许数据源对该获取过程中使用到的一些资源进行释放,如释放数据库连接、删除临时文件等.

3.2 数据组织与管理

遥感数据通常由元数据和影像数据两部分组成.元数据是用来描述诸如覆盖范围、创建时间、数据类型、格式、用途等详细信息的文本,通常被存储在数据文件或数据库中.不幸的是,不同来源不同卫星的遥感数据其元数据的格式和内容相差很大.这些截然不同的元数据描述给元数据的统一组织和存储带来了不小的麻烦,定义一种灵活的、可扩展的元数据描述方法成为组织和存储元数据的必要前提.从灵活性和可扩展性出发,基于键值对的文档是一种

能够更加灵活方便地描述遥感数据的覆盖范围、采集时间和卫星/传感器等观测特性的组织方案.不同种类和来源的遥感数据可以具有不同名称的键值,键值的数据类型也可以不同(例如一个字符串、一个数、其他的文档,或者字符串、数和文档的列表等).XML 和 JSON 是比较常用的 2 种元数据传输和交换的格式,例如,MODIS 数据的元数据可以以 JSON 和 XML 格式分别描述如图 10 所示:

```
{ "id":"1456304112338__796__956875114",
  "CREATE_TIME":"2002-10-16 00:00:00",
  "COVERAGE":{ "TYPE":"Rectangle",
    "UL_LAT":30.05501,"UL_LON":115.0986,
    "UR_LAT":29.999105,"UR_LON":127.035,
    "LL_LAT":19.940222,"LL_LON":106.0635,
    "LR_LAT":19.886341,"LR_LON":117.0474}
  "SENSOR_ID":"MODIS","RESOLUTION":500,
  "SPACECRAFT_ID":"AQUA","LEVEL":"l2",
  "FORMAT":"HDF","DATASIZE":72675949
}
```

(a) JSON Description

```
<?xml version="1.0" encoding="UTF-8"?>
<dataItem id="146304112338__796__956875114">
  <CREATE_TIME>2002-10-16 00:00:00
  </CREATE_TIME>
  <COVERAGE type="Rectangle">
    <UL_LAT>30.05501</UL_LAT>
    <UL_LON>115.0986</UL_LON>
    <UR_LAT>29.999105</UR_LAT>
    <UR_LON>127.035</UR_LON>
    <LL_LAT>19.940222</LL_LAT>
    <LL_LON>106.0635</LL_LON>
    <LR_LAT>19.886341</LR_LAT>
    <LR_LON>117.0474</LR_LON>
  </COVERAGE>
  <SENSOR_ID>MODIS</SENSOR_ID>
  <SPACECRAFT_ID>AQUA</SPACECRAFT_ID>
  <RESOLUTION>500</RESOLUTION>
  <LEVEL>l2</LEVEL><FORMAT>HDF</FORMAT>
  <DATASIZE>72675949</DATASIZE>
</dataItem>
```

(b) XML Description

Fig. 10 Format examples for meta-data exchange

图 10 元数据传输与交换格式

在存储方面,由于元数据的结构过于复杂和灵活,使用对数据结构要求严格的传统关系型数据库存储这些元数据显得有些力不从心.因此,从灵活性和兼容性出发,元数据存储经常使用 NoSQL 数据库进行存储,例如类似 MongoDB 的文档数据库或者类似 Apache Cassandra 的键值对存储系统.而且,很多 NoSQL 数据库采用分布式的基本体系结构,其存储规模的水平可扩展性对于存储数据量巨大的元数据有着明显的优势.但是,NoSQL 数据库的数据查询能力通常较弱,尤其是对于复杂查询的支持能力不足.为此,我们通常要通过额外建立倒排表或者结构化索引等方式来提升元数据的查询能力,而这些都会给元数据存储的实现带来额外的难度和开销.

另一种可行的解决方案就是混合使用关系型数据库和 NoSQL 数据库.由于元数据中通常只有少量比较"通用"的域被用于数据查询,比如卫星名、传感器名、数据覆盖范围、采集时间等,我们可以将这些数据从元数据中提取出来并存储在关系型数据库中用于查询,而元数据的主体依旧保存在 NoSQL 数据库中以保证其灵活性.这时,关系型数据库作为一个"索引数据库"来使用.

在图 11 所示的解决方案中,数据 ID、可查询的域以及一个到 NoSQL 数据库的引用指针被保存在分布式数据库中,同时建立完善的数据索引以提高数据查询的效率.数据基础设施可以首先从关系数据库中根据条件查询到一组数据 ID 和到 NoSQL 数据库的引用指针,再根据这些引用指针(通常是 NoSQL 数据库中的数据 ID)从 NoSQL 数据库中迅速读出元数据的全部内容,从而达到快速查询的目的.

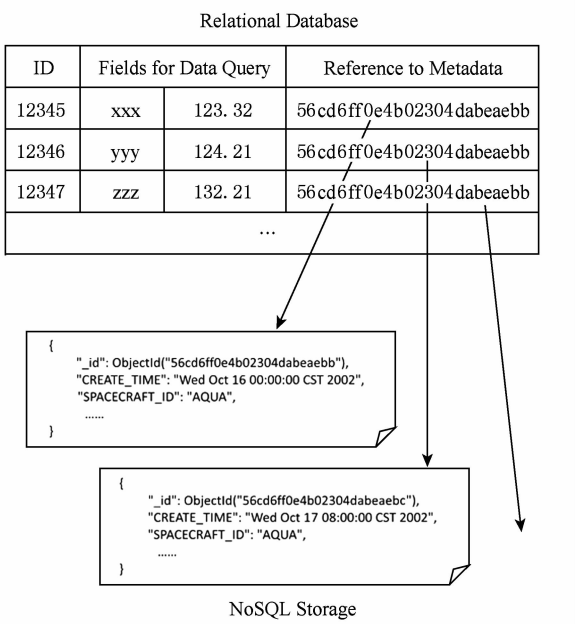

Fig. 11 Hybrid meta-data storage

图 11 结合关系数据库与 NoSQL 数据库的元数据存储方案

另一方面,作为遥感数据主体的影像数据有矢量和栅格 2 种不同的数据模型.栅格数据是按网格单元的行与列排列、具有不同取值的阵列数据,它使用

大小相等、分布均匀、紧密相连的像元(网格单元)阵列来表示空间地物或现象分布的数据组织. 栅格数据易于实现,算法简单,易于扩充、修改,直观性强,是遥感影像数据常用的表示模型. 但是,由于栅格数据需要针对每个网格单元给出观测值,其数据量比较大,通常以文件的形式存储在文件系统之中. 另外,为了提高栅格数据的访问效率,一种面向栅格数据的数据管理系统 Array DBMS 近年来也受到了一定程度的重视.

与栅格数据相反,矢量数据使用点、线、矩形、多边形、圆和弧线等图形来描述一个地理实体的空间分布. 矢量数据通常结构紧凑、数据量比较小,除了以文件的形式存储在文件系统之外,还经常以编码的字符串或 BLOB(二进制大对象)的形式存储在关系数据库或 NoSQL 数据库中. 无论矢量数据还是栅格数据,无论这些数据以文件的形式被存储在文件系统中还是以数据对象的形式存储在数据库中,遥感影像数据通常被赋予一个唯一 ID,这个 ID 被用来将影像数据与元数据关联在一起.

从数据的安全性和高效访问出发,遥感数据基础设施可以将元数据和影像数据复制多份并将其存储在地理上分布的多个不同的存储节点中,从而避免存储节点的故障带来的数据丢失. 另一方面,多个不同拷贝也使得遥感数据可以就近访问,减少传输遥感数据所需要的时间;同时多个不同拷贝也可以有效地分散遥感数据查询与获取给存储节点带来的压力,提高数据访问的效率. 与此同时,遥感数据基础设施还可以建设在一些具有良好的可扩展性和伸缩性的分布式存储系统之上,诸如分布式文件系统、云存储系统、分布式数据库等都可以为遥感数据基础设施提供性能更好、扩展性更强、开销更低的存储基础.

3.3 数据服务接口

从本质上来说,遥感数据基础设施的最基本任务是根据用户给出的时间、空间、观测特性等条件,找到符合其需求的遥感数据并提供给用户. 所以,数据服务最基本的接口应当是用来根据给定的时间、空间位置和观测特性返回确切观测值(整数、浮点数、字符串或者空值等)的 *getValue*,猝发返回一个时间范围和空间区域(多数情况下是一个矩形)内大量数据以提高服务性能的 *getValues*,以及在指定时间、指定区域内寻找某项或某几项观测值符合指定条件的子区域的 *findArea*. 这些服务功能接口的原型可以表示如下:

value *getValue*(*position*,*time*,*characteristics*);

valueArray *getValues*(*area*, *timeSpan*, *characteristicsList*);

area *findArea*(*area*,*conditions*).

在这些功能接口中,*position* 应当是一个用经纬度等方式描述的确定地点;*time* 是一个确定的时间,*timeSpan* 是由开始时间和结束时间所指定的一段时间间隔;*characteristics* 用来指明观测特性(如卫星、传感器、波段等),*characteristicsList* 则是观测特性的一个列表;*area* 是感兴趣的区域,通常用多边形甚至矩形来描述;*conditions* 是一个复杂的条件选项,其结构和功能与 3.1 节中 *query* 的 *request* 参数相仿.

很明显,前述几种服务功能是建立在以遥感数据对象为基本服务单元这个前提下的,而以影像文件为基本服务单元的第Ⅰ类和第Ⅱ类数据基础设施就无法提供这些服务功能接口. 对于第Ⅰ类和第Ⅱ类数据基础设施,其基本服务功能应当围绕影像数据文件的查找和获取来提供. 因此,用来查询可用遥感数据的 *query* 和用来获取遥感数据文件以供下载的 *access* 应当是第Ⅰ类和第Ⅱ类数据基础设施最基本的服务功能. 这些服务功能接口的原型可以表示为:

MetaDataList *query*(*conditions*);

AccessResult *access*(*dataID*).

其中,*conditions* 为条件选项,MetaDataList 是以元数据列表形式描述的查询结果;*dataID* 是从元数据中分离出来的数据唯一 ID,AccessResult 则用来描述数据获取得到的结果,包括数据能否下载、哪些用途的数据能够从哪个 URL 下载等. 除此之外,遥感数据基础设施还有可能提供诸如根据给定条件统计数据的条目数和数据量等更多的附加功能以方便用户的使用.

在 Web 的基础上,基于 XML,SOAP,WSDL,UDDI 等一系列标准协议,Web 服务由于其出色的平台无关性、自描述性、可扩展性和灵活性而被广泛使用于基于 Internet 的软件服务,成为目前面向服务的体系结构(service oriented architecture, SOA)最常见的实现方式. 使用 Web 服务技术来提供服务也是遥感数据基础设施一种比较理想的实现方案. 但是,比较复杂的协议,XML 解析与验证等过程也给基于 SOAP 的 Web 服务带来一系列的额外开销,在一定程度上影响了服务的效率. 为此,轻量级的 RESTful 服务也经常被用来提供高性能的遥感数据服务.

REST 即 REpresentational State Transfer,是一种互联网服务的架构原则. RESTful 服务将 Web 服务视为可以由其 URL 唯一标识的资源,使用不同的 HTTP 方法(如 GET,POST,PUT 和 DELETE)来表示对资源的不同操作(如读取、新建、修改和删除). 和 Web 服务明确使用 XML 作为其编码基础不同,RESTful 服务可以使用任何表示层编码协议,如 XML,JSON 或者普通文本,这就使得 RESTful 服务的实现更加灵活. RESTful 服务的编程模型简单,诸如 JSON 等轻量级编码方案的使用也减少了序列化和反序列化过程的复杂性,提高了服务效率.

一个 RESTful 的遥感数据服务可能基于 HTTP GET 方法提供其 *getValue* 操作接口,读取用户给出的时间、位置和观测特性信息,并返回一个诸如 17 或者 226.0 这样的观测值. 例如,一个读取 TERRA 卫星、MODIS 传感器、2 号波段指定地理位置和时间的观测数值的 HTTP GET 命令可能会是:

GET /rsdata/39.9,116.3,2016-02-12+02:10:10,
SC%3DTERRA,SR%3DMODIS,BAND%3D2.

在某些时候,遥感数据查询或获取的操作可能需要比较长的时间,容易引起客户端程序的阻塞. 为此,遥感数据基础设施也可以将其接口操作以异步化的形式提供. 比如,以轮询方式异步化的遥感影像文件获取操作可能会将操作 *access* 分为 3 个异步的操作:启动一个获取过程的 *startAccess*,读取获取结果的 *getAccessResult* 和关闭获取过程的 *closeAccess*. 这 3 个操作分别使用 HTTP 方法 POST,GET 和 DELETE 来调用. 基于 SOAP 的 Web 服务也可以提供类似的异步化操作接口,以方便用户的异步服务调用.

3.4 按需数据处理

当前的遥感数据基础设施普遍停留在简单地将已有的存档数据按照原样提供出来的初级阶段,用户必须将服务所提供的数据通过网络下载到本地以供继续分析处理之用. 由于这个下载过程通常要通过相对低带宽、高时延的广域网,遥感数据的下载通常需要比较长的时间(如图 12 所示).

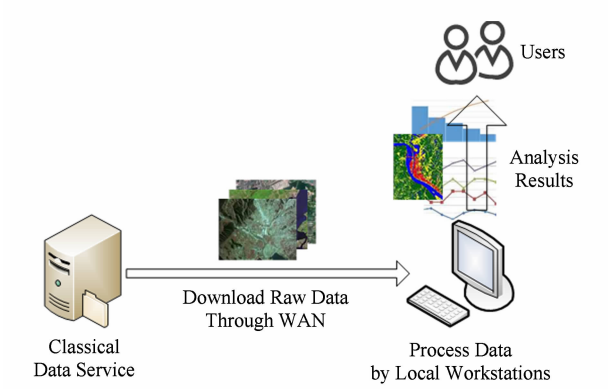

Fig. 12 Classical workflow for remote sensing data analysis

图 12 传统遥感数据分析过程

在大多数情况下,遥感数据处理模型的输出结果在数据量上要远远小于这些处理模型的输入数据,这就使得"就近计算"成为一种提高遥感数据分析处理效率的有效手段. 所谓"就近计算",就是在遥感数据基础设施中部署具有比较强大计算能力的计算资源,并且使这些计算资源和遥感数据之间具有高带宽、低时延的高性能网络连接(大多数情况下是计算资源和数据资源位于同一个高速局域网之中). 这样,用户就可以将其处理遥感数据所使用的遥感数据分析模型推送到数据基础设施所提供的计算节点上按需地部署运行,再将数据量相对比较小的输出结果通过广域网返回给用户(如图 13 所示).

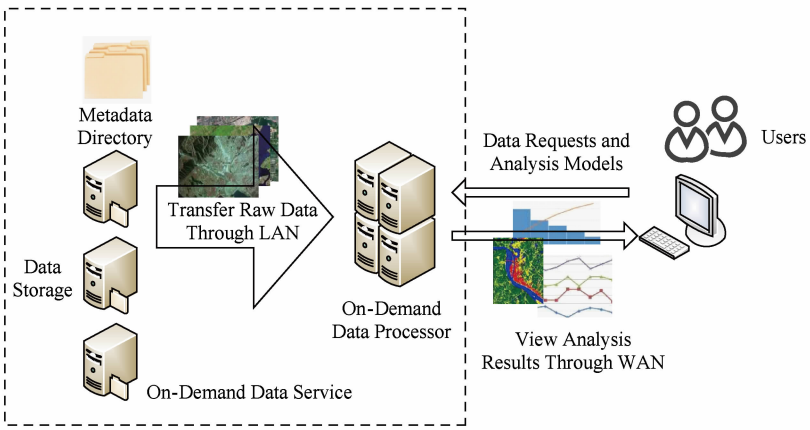

Fig. 13 Workflow for on-demand remote sensing data analysis

图 13 按需数据处理支持下的遥感数据分析过程

遥感数据的按需数据处理将对数据的处理计算从用户的本地工作站转移到了遥感数据基础设施所提供的计算节点之上,一方面可以有效地利用遥感数据基础设施提供的高性能计算资源对数据进行高速处理,另一方面可以避免海量的原始遥感数据在低带宽、高时延的广域网上的反复传输,对于提高遥感数据的处理速度有着非常明显的效果.同时,按需数据处理也能够给遥感数据基础设施提供集成的数据处理能力,对于通过数据的插值和反演等操作实现时空无缝化的数据服务非常有利.但是,由于遥感数据分析模型的部署和运行可能依赖于不同的软硬件环境,如硬件平台、操作系统、编译器、支持库等,这就给遥感数据分析模型的按需部署和运行带来了一些困难.

随着虚拟化技术的不断发展,虚拟机的运行效率不断提高,其部署和管理也越来越方便,这就给使用虚拟机支持遥感数据分析模型的按需部署和运行提供了一种工程上可行的解决方案.数据基础设施中的按需数据处理支持模块可以为用户提供一系列安装了基本环境的虚拟机镜像,用户可以在这些虚拟机镜像的基础上按照其要求进行定制,使之满足用户的遥感数据分析模型的部署与运行需求.由于用户可以在系统提供的基本环境虚拟机镜像基础之上进行深度定制,此方案对于遥感数据分析模型的匹配程度非常高,理论上可以支持任何遥感数据分析模型的部署和运行.

庞大的虚拟机镜像无论在存储还是在传输上对于按需数据处理的实现都是一个不小的负担.虽然可以通过增量传输和存储以及容器技术等方法来减少传输和存储过程中的冗余,提高系统镜像传输和存储的效率,但是这种为了灵活性而付出的代价依旧不可小视.另外,基于虚拟机的遥感数据分析模型的种类繁多,运行方法多样,对这些数据分析模型进行描述和进一步的流程组合将会非常困难.相比起来,基于模型描述的按需数据处理技术就能够在一定程度上解决这些问题.

限制遥感数据分析模型部署和运行的主要难点在于如何为遥感数据分析模型提供合适的运行时环境.如果能够有一套标准规范能够将遥感数据分析模型所使用的运行时环境加以规范描述,从而使得这些运行时环境能够准确地被复现,对运行时环境的描述就可以被遥感数据分析模型携带着上传到按需数据处理系统中,指导按需数据处理系统选择和构建合适的运行时环境部署和运行遥感数据分析模型.更进一步,如果遥感数据分析模型能够被以一套规范的形式予以描述,那么遵照此规范构建的高性能并行运行时环境就可以支持这些遥感数据分析模型的高效运行.而且,规范化的描述也将有利于遥感数据分析模型的迁移、共享和流程化运行.

基于 XML 和 JSON 语言的强大描述能力,运行时环境的描述可以基于 XML 或 JSON 来规范.例如,一个"安装有 Windows 10 64bits 操作系统、Visual Studio 2010 编程环境的单机"的环境需求可以用图 14 所示 JSON 或 XML 文档描述.

```
{"type":"simple",
 "OS":{
   "name":"Windows",
   "Version":"10",
   "Distribution":"Windows 10 64bits"
 },
 "Tools":[
   { "name":"Visual Studio",
     "Version":"2010"
   }
 ]
}
```

(a) JSON Description

```
<?xml version="1.0" encoding="UTF-8"?>
<environment type="simple">
<OS name="Windows">
<version>10</version>
<distribution>Windows 10 64bits
</distribution>
</OS>
<tool name="Visual Studio">
  <version>2010</version>
</tool>
</environment>
```

(b) XML Description

Fig. 14 Examples for runtime environment description

图 14 运行时环境描述的示例

由于按需数据处理带来的一系列好处,目前已经出现了一些相关的研究.例如 OGC 的 WCPS 就试图定义一种协议无关的处理语言以实现对多维覆盖的按需抽取、处理和分析[13].另外,论文[14]也试图将遥感数据处理所需要的工具从用户所在的工作站迁移到数据提供者一端,以提高数据处理流程的速度.但是这些研究基本上还是处在解决单个问题的阶段,并没有针对遥感数据分析模型的按需部署与应用问题做全景式的分析与探索.

为了验证按需数据处理为遥感数据基础设施带来的性能增益,我们建立了一个简单的原型系统,并在该原型系统上以基于归一化水指数 NDWI[15] 的

干旱检测为典型应用进行了性能测试.该原型系统运行在2台以千兆位以太网连接的桌面型服务器上,每台服务器配有一个i5-4570@3.20 GHz CPU,16 GB存储器和4 TB硬盘.这2台桌面型服务器中的一台用来模拟提供按需数据处理功能的数据基础设施服务端,另一台用来模拟用户端的工作站.

在该原型系统中,数据处理模型被以Java字节码的形式描述,并被按需地部署到一个具有JDK和HDF运行库的64位Windows环境之中,按需数据处理的运行时环境描述采用XML格式.其描述如下:

```
〈?xml version="1.0" encoding="UTF-8"?〉
〈environment type="simple"〉
  〈OS name="Windows"〉
    〈version〉10〈/version〉
    〈distribution〉
      Windows 10 64bits
    〈/distribution〉
  〈/OS〉
  〈tool name="JDK"〉
    〈version〉1.8〈/version〉
  〈/tool〉
  〈tool name="JHDF"〉
    〈version〉3.2.1〈/version〉
  〈/tool〉
〈/environment〉
```

我们使用从2000—2010年这11年间的4 d中覆盖h27v05和h27v06的MOD09数据共88个HDF格式的遥感数据作为测试数据,测试数据总量为约6.43 GB,作为处理结果的AWI文件数据总量约483 MB.

根据我们在多个典型网络环境下进行遥感影像数据下载的测试,通过Internet进行遥感影像下载的数据传输率通常在1～10 MB/s之间,因此我们使用软件将数据传输带宽分别限制在1 MB/s,4 MB/s,10 MB/s,以及不做限制(此时的数据传输带宽上限应当是硬件限制的1 Gbps,即125 MB/s)进行性能测试.在不同数据传输带宽限制下的性能测试结果如图15所示,其中Classical表示传统的数据处理流程,ODP表示按需数据处理.在图15中,柱状图为不同处理方法各步骤所消耗的时间(左上右下斜线部分为数据传输与模型部署时间,右上左下斜线部分为模型运行时间),折线图为不同处理方法的运行效率(定义为模型运行时间占总时间的百分比).

从图15中可以看出,无论网络带宽高达1 Gbps还是低至1 MB/s,按需数据处理都能表现出相对于传统数据处理方法更明显的性能增益,数据处理总时间明显缩短,处理效率明显提高.和传统的数据处理方式相比,按需数据处理所需要的总处理时间

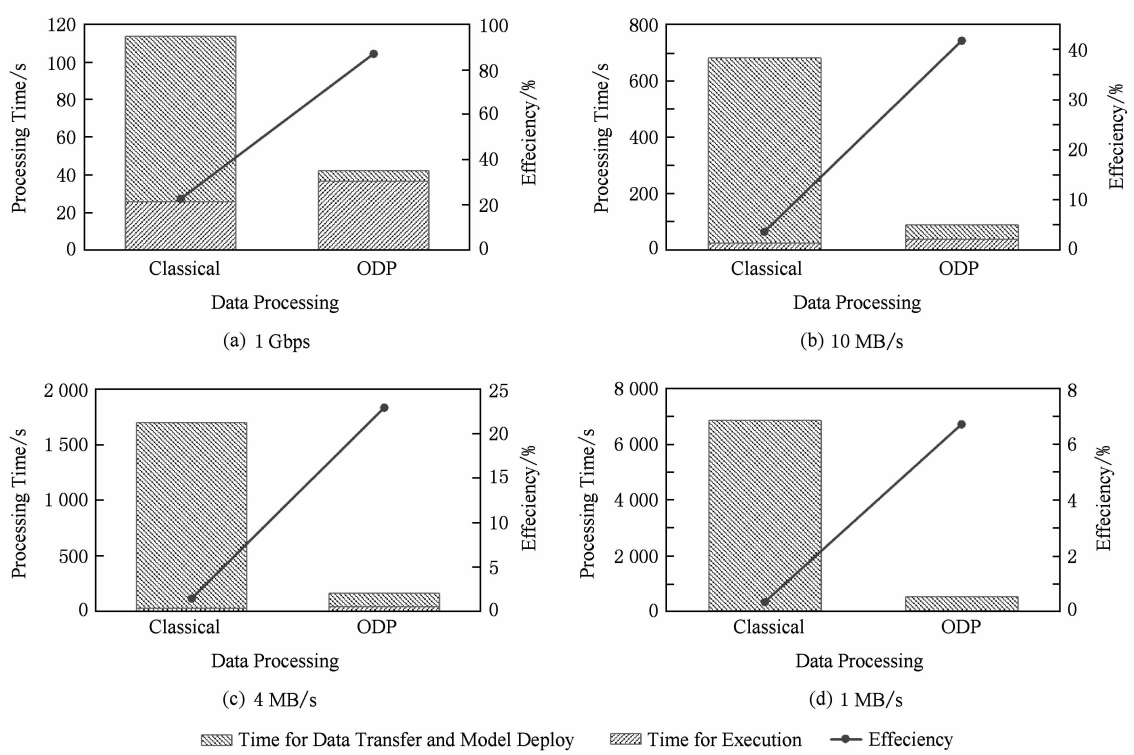

Fig. 15 Performance test results of on-demand processing

图15 按需数据处理的性能测试

缩短了 60%～90%,而且这种性能增益在低带宽的网络环境下显得更加明显.

按需数据处理对于遥感数据基础设施来讲具有非常重要的意义.首先,正如本文实验所证明,按需数据处理将数据的处理过程从用户端移动到了数据服务端,避免大量原始影像数据通过广域网络传输带来的网络开销,提高遥感处理的整体效率.其次,分布式的数据基础设施可以通过对遥感数据分析模型的有效调度,使这些分析模型在尽可能接近数据的高性能计算资源上运行,这种"就近计算"的调度能够有效地提高遥感数据分析模型的运行效率.最后,更加重要的是,在用户将遥感数据分析模型推送到遥感数据基础设施中之后,用户可以依照其意愿将这些分析模型与其他用户共享,将来自不同机构的分析模型组织在一起,更好地支持遥感数据的分析和研究.在这种情况下,遥感数据基础设施就能够更好地为遥感科学研究和工程应用提供 PaaS (platform-as-a-service)和 SaaS(software-as-a-service)服务.在这个基础设施提供的各种基本环境、数据、工具、库等的支持下,用户可以方便地开发和共享遥感数据分析模型,实现合作研究和协同应用.

4 总 结

随着遥感数据量的不断增加、遥感应用在规模上的不断扩大和实现上的不断复杂化多样化、以及跨地域跨组织合作研究需求的不断增加,遥感科学与工程应用对于大数据时代的遥感数据基础设施在地域分布性、可扩展性、可用性、易用性和性能表现等方面的要求越来越高.传统的遥感数据基础设施大多数以影像文件为基本服务单元,只能将单一机构的存档数据通过 Web 界面等方式提供给用户查询和下载,在服务效率和易用性等方面远远不能满足不断发展的遥感科学与工程应用的需求.

从本质上来讲,遥感数据是对客观世界的数值化反映,是一个由时间、空间和观测特性所组成的高维数据空间.对于确定的时间、空间和观测特性,遥感数据应当有确定的取值.相应地,遥感数据基础设施应当以形式化、规范化的访问接口,为遥感应用提供以 *getValues* 和 *findArea* 为代表的数据服务操作,允许用户根据遥感数据的高维空间各维度的数值直接得到相应的观测值,或者根据给定的条件在遥感数据的高维空间中确定符合条件的子空间.这样的遥感数据基础设施会更加本质和易用.

根据遥感数据基础设施的基本服务单元、分布性、数据的时空连续性和按需处理支持,本文将遥感数据基础设施分成了 6 类,并从系统构造的角度,讨论了构建这 6 类遥感数据基础设施所应采用的体系结构,指出了各类型遥感数据基础设施实现的关键问题.在此基础上,本文还就各类遥感数据基础设施在构建过程中需要考虑的数据收集与整合、数据组织与管理、数据服务接口、按需数据处理等方面的实现方案进行了深入的讨论.在这些技术的支持下,遥感数据基础设施能够做到分布化、智能化和平台化,达到数据与处理"存算一体"的目标,并在此基础上实现遥感数据分析模型的共享和流程化,支持基于大数据的遥感科学合作研究和工程上的协同工作.

参 考 文 献

[1] Hey T, Tansley S, Tolle K. The Fourth Paradigm: Data-Intensive Scientific Discovery [M]. Redmond, WA: Microsoft Corporation. 2009: 252

[2] Demchenko Y, Grosso P, De Laat C, et al. Addressing big data issues in scientific data infrastructure [C] //Proc of 2013 Int Conf on Collaboration Technologies and Systems (CTS). Piscataway, NJ: IEEE, 2013: 48-55

[3] Ma Yan, Wu H, Wang L, et al. Remote sensing big data computing: Challenges and opportunities [J]. Future Generation Computer Systems, 2015, 51(2015): 47-60

[4] Open Geospatial Consortium (OGC). OGC® Standards and Supporting Documents [OL]. [2016-10-02]. http://www.opengeospatial.org/standards

[5] GEO. GEOSS Core Architecture Implementation Report [OL]. [2016-10-02]. http://portal.opengeospatial.org/files/?artifact_id=24315

[6] Cossu R, Bally P, Colin O, et al. ESA grid processing on demand for fast access to earth observation data and rapid mapping of flood events [G]. Munich, Germany: European Geosciences Union General Assembly. 2008

[7] Sekiguchi S, Tanaka Y, Kojima I, et al. Design principles and IT overviews of the GEO grid [J]. IEEE Systems Journal, 2008, 2(3): 374-389

[8] Li G, Liu D, Huang Z, et al. Spatial data service models in grid environment [C] //Proc of the Int Symp on Parallel and Distributed Processing and Applications. Berlin: Springer, 2006: 598-602

[9] Huang Z C. On-demand data service for the next generation spatial data infrastructure [C] //Proc of the 5th Int Conf on Semantics, Knowledge and Grid. Piscataway, NJ: IEEE, 2009: 286-289

[10] Iosup A, Ostermann S, Yigitbasi M N, et al. Performance analysis of cloud computing services for many-tasks scientific computing [J]. IEEE Trans on Parallel and Distributed Systems, 2011, 22(6): 931-45

[11] NASA. NASA NEX [OL]. [2015-02-28]. http://aws.amazon.com/cn/nasa/nex/ [2015-02-28]

[12] Foster I, Zhao Y, Raicu I, et al. Cloud computing and grid computing 360-degree compared [C] //Proc of 2008 Grid Computing Environments Workshop. Piscataway, NJ: IEEE, 2008: 1-10

[13] Baumann P. The OGC Web coverage processing service (WCPS) standard [J]. Geoinformatica, 2010, 14(4): 447-479

[14] Davis B N, Werpy J, Friesz A, et al. Interactive access to LP DAAC satellite data archives through a combination of open-source and custom middleware Web services [J]. IEEE Geoscience and Remote Sensing Magazine, 2015, 3(4): 8-20

[15] Gao B C. NDWI—A normalized difference water index for remote sensing of vegetation liquid water from space [J]. Remote sensing of environment, 1996, 58(3): 257-266

Li Guoqing, born in 1968. PhD, professor and PhD supervisor. Senior member of CCF. His main research interests include high performance geocomputation, spatial data infrastructure, and digital earth.

Huang Zhenchun, born in 1975. PhD, associate professor. His main research interests include remote sensing data processing, high performance computing, and distributed computing.