SCO-GADL:一种用于科学计算的网格工作流描述语言

黄震春
(清华大学计算机系 北京 100084)

摘 要 应用开发的难度一直是制约网格技术成为科学计算基础设施的主要因素之一。虽然网格工作流等诸多技术的使用能够在一定程度上降低网格应用开发的难度,但是大多数网格应用所采用的基于流程的应用描述模型仍然是网格应用开发的一个主要障碍——尤其是对那些通常情况下不擅长编程的科学家们。为了降低网格应用开发的难度,提出了一种基于数据依赖关系的网格应用描述模型,力图使网格应用的描述更加符合科学工作者的思维习惯。在此基础上,设计和实现了一种被称作SCO-GADL的工作流描述语言及其执行引擎。该引擎采用核心-插件体系结构,能够使用在多种网格平台之中,为科学工作者提供一种方便、易用和快捷的网格应用开发工具,以便使网格中聚集的各种资源更加高效地进行科学研究。

关键词 科学计算,网格工作流,应用描述语言,数据依赖关系
中图法分类号 TP319 **文献标识码** A

SCO-GADL: A Grid Workflow Description Language for Scientific Computing

HUANG Zhen-chun
(Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China)

Abstract The difficulty of grid-enabled application development is one of the obstacles to employ grid as infrastructure of scientific computing. Although there are many projects, especially grid workflow projects, trying to make grid-enabled application development easier and quicker, the process-based application description model adopted by most of the grid-enabled applications still bothers scientists who are usually beginner in programming. In this paper, a data dependency based application description model was proposed, which describes a grid-enabled application via dependency among data items in the application, and much close to the thinking pattern of scientists. Based on this model, a scientific computing oriented grid application description language (SCO-GADL) was proposed, and an engine for supporting its execution was designed and implemented. The extensible plug-in architecture adopted by SCO-GADL and its engine makes them support different grid and non-grid platforms easily. At last, a test shows that grid-enabled scientific applications can be developed easily in SCO-GADL, and executed correctly on the application engine.

Keywords Scientific computing, Grid workflow, Description language, Data dependency

1 引言

在过去的十多年里,网格计算在理论、工具和应用等方面得到了长足的进步,连接了越来越多的科学仪器、超级计算设备、实验数据、分析算法和仿真模型等科学研究资源,并把这些资源整合为一个整体,为全世界的科学工作者和工程师提供了强有力的合作研究环境。虽然网格的支持者声称网格能够在计算能力、数据共享和协同工作等诸方面为科学工作者带来便利,但是网格被科学工作者所接受和使用还要依赖于在网格上运行的大量的为科学工作者们服务的科学应用。可惜的是,其开发技术门槛比较高,网格应用的开发者必须在开发应用的过程中解决如资源预约与分配、认证与安全、消息与通信、数据分布与传输等诸多网格应用所必须解决的问题,而这对于对网格知之不多、编程经验相对较少的科学工作者来讲是一件非常困难的事情。网格应用开发的高门槛严重影响了科学工作者们使用网格的热情,成为了网格在科学应用中推广的一个严重障碍。

当前已经有很多研究项目发布了不少网格应用开发工具或支持库,如著名的GrADS[1], Condor-G[2]和MPICH-G2[3],为科学家和工程师们提供了更加强大易用的开发工具或支持库,方便他们在自然科学、社会科学、工程技术诸方面建立数学模型并使用计算机对已有的和采集到的各种数据进行仿真与分析,从而解决了更多的科学与工程问题。网格工作流是这些应用开发工具中使用比较广泛也比较受到重视的一种。与WfMC对工作流的定义不同,网格中的工作流被定义为:"一种被用于解决实际问题或形成新的服务的自动化流程,依靠引用和组合各种网格服务、网格代理或其他网格中的资源实现"[4]。在这个定义中,网格工作流重视自动化、服务

到稿日期:2010-07-26 返修日期:2010-12-07 本文受863课题(2009AA12Z146),中欧科技合作龙计划二期项目(ID5258),中央级公益性科研院所基本科研业务费专项资金(IFRIT200905)资助。
黄震春(1975—),男,博士,副研究员,主要研究方向为分布式计算、并行计算、计算机体系结构,E-mail:huangzc@tsinghua.edu.cn。

和资源组合的特点被明确地表现出来。

由于灵活、表现力强、开发相对较容易等优点，网格工作流逐渐成为网格领域中最重要的应用开发工具之一，诸如 Triana[5]，Kepler[6]，Taverna[7] 这样的网格工作流工具包已经得到了比较广泛的应用。在网格工作流中，最有名的莫过于作为 OASIS 标准的“Web 服务业务流程语言”，即 WS-BEPL，或被称作 BEPL4WS 和 BPEL[8]。BPEL 与类似于 BPEL 的工作流描述语言已经在很多网格项目中得到使用[9]。但是，这些网格工作流对于科学与工程计算来讲依旧不是理想的应用描述手段。例如，作为一种业务流程描述语言，BPEL 采用了以流程/活动为基础的面向流程的描述体系，通过显式描述每个步骤中动作的细节和动作之间的先后次序来说明应用的业务流程，具有很强的描述能力，但是其开发门槛相对比较高，掌握起来比较困难，限制了其使用的范围。再如，在一组图形化用户界面的支持下，Kepler 的易用性和描述能力都非常优秀，但这种易用性和描述能力是以牺牲工作流描述的通用性而获得的：Kepler 被定义为一种面向生物信息学应用的描述语言，无法用于其他应用领域的网格应用开发。

为了提高网格应用开发的效率，降低网格应用的开发门槛，使科学工作者和工程师无需太多编程基础就可以开发网格应用，因此需要更符合科学工作者思维习惯的应用描述模型和应用描述语言。为此，本文从研究科学计算所使用的应用描述模型出发，以基于数据依赖关系的应用描述模型为基础，设计了一种面向科学计算应用的工作流描述语言，并实现了其执行引擎。一些实验测试表明这个被命名为 SCO-GADL（Scientific Computing Oriented Grid Application Description Language）的应用描述语言可以被用于描述各种不同要求的科学计算网格应用。

2 科学计算应用的描述模型

众所周知，传统的网格应用大都采用基于流程的描述模型。在这类描述模型中，应用被视为一个由一系列处理步骤（语句或动作）所组成的“流程”。流程各步骤之间的组织和衔接由一个与程序中的各种变量取值相关联的控制流所控制，应用必须明确定义控制流，以保证应用能够依照正确的次序激发正确的处理步骤，完成整个应用的处理过程。

虽然基于流程的应用描述模型在网格中最为常见，但是它并不是最适合描述科学计算应用的描述模型。对于科学工作者来讲，他们更习惯于把一个问题的解描述为“结果是 1 到 100 的和”，而不是像基于流程的应用描述那样描述为“首先给 sum 变量赋初值 0，然后令变量 i 从 1 循环到 100，并依次将 sum 赋值为 sum+i，最后将 sum 的值返回作为结果”。对于前一种描述，把它归纳为“基于数据依赖关系的应用描述模型”（Data Dependency based Application Description Model——D^2ADM）。

与基于流程的应用描述将应用描述为“怎样得到结果”不同，在基于数据依赖关系的应用描述模型中，应用被描述为“结果是什么”。D^2ADM 使用一个三元组 (T_D, D, dep) 来描述网格应用，其中 T_D 是应用所要得到的描述科学问题结论的结果数据集。假定 d_i 表示应用中出现的一个数据项，这个三元组可以被定义为：

$T_D=\{d_i \mid d_i$ 是结果数据$\}$

$dep=\{d_i \rightarrow d_j \mid d_j$ 依赖于 $d_i\}$

$D=Cl(T_D, dep)$

式中，dep 函数代表数据之间的依赖关系，$d_i \rightarrow d_j$ 代表数据项 d_j 的取值依赖于数据项 d_i，即 d_j 可以通过一个数据处理步骤以 d_i 为基础（输入）得到，这时 d_i 被称为 d_j 的“前驱数据”。C_l 是闭包运算，其结果是 T_D 的一个超集，其中的所有元素都可以从 T_D 中的元素通过有限次函数 dep 运算获得：

$D=\{d \mid \exists td \in T_D, \exists n \in N \rightarrow td=dep^n(d)\}$

因此，D 是用以计算结果数据集 T_D 所必需的所有数据的集合，这个集合被称为该应用的全集。如果把 D 中的数据项 d_i 看作顶点，将依赖关系 dep 看作边，则基于数据依赖的应用描述可以以一个有向图来描述，这个有向图被称作应用的数据依赖图，如图 1 所示。

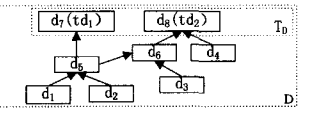


图 1 数据依赖图的例子

在数据依赖图中，如果有环的存在，则说明某个数据项直接或间接依赖于其自身，这种数据项是无法被求解的。因此，一个可以得到求解的应用的数据依赖图一定是一个有向无环图。在这个有向无环图中，有一些节点没有前驱节点（即不依赖于其他数据项，如图 1 中的灰色节点），这些节点被称作原始节点和原始数据，它们是应用求解的基础。如果数据依赖图中所有原始节点都已经得到了确定的值，那么应用数据项全集 D 中的所有数据项都可以通过依赖关系得到赋值，从而得到所有的结果数据，完成问题的解答。

D^2ADM 使用了一种比传统的基于流程的描述方法更加符合科学工作者思维方式的描述方法来描述科学应用，科学工作者们可以更加自然地使用这种方式来描述他们期望的科学计算应用，从而大大降低网格应用开发对于大多数科学工作者们的技术门槛。与此同时，D^2ADM 还比传统的基于流程的描述方法更有利于实现并行化。在传统的基于流程的应用描述方法中，用户必须根据其对应用结果的需求手工定义应用的控制流，对于那些未经过并行程序设计训练的非专业使用者（比如大部分科学工作者），编制一个具有良好并行特性的应用程序的难度非常大。而一个未经过良好编制和优化的控制流通常情况下会严重地损伤应用程序本应具有的并行性，从而使得一个本应能够得到高效并行求解的应用无法获得应有的性能。而在 D^2ADM 中，由于应用由数据之间的依赖关系描述，数据依赖关系中所隐含的各种并行性都不会被损害，这些并行性会被应用引擎所分析和利用，对问题进行并行求解。例如在图 1 所示的应用中，d_7 和 d_8 两个数据项的求解过程就可以并行进行，从而提高应用执行的效率。

3 面向科学计算的网格工作流 SCO-GADL

在当前的网格项目中，被用于描述网格应用的工作流大多数采用基于流程或基于 Perti 网的描述模型，这些工作流描述语言并不适合于描述基于数据依赖关系描述的网格应用。因此，本文基于 XML 定义了一种基于数据依赖关系应用描述模型的应用描述语言，用以支持各种科学计算应用的实现。这个基于 D^2ADM 的工作流描述语言被命名为“面向科学计

算的网格应用描述语言"(简称 SCO-GADL)。

SCO-GADL 使用一个 XML 文档来描述网格应用,该文档以一个名字为"application"的元素为根元素,根元素的"target"属性被用于描述应用目标数据集中包含的数据项名称。在 application 元素下,一组名字为"data"的子元素被用来描述应用数据项全集 D 中的所有数据项。Data 子元素至少要包含两个属性:name 和 type,分别用来描述对应的数据项的名称(唯一索引)和数据项的依赖类型。在 SCO-GADL 中,依赖类型被定义为该数据项应当采用何种方式求解,目前 SCO-GADL 定义了如表 1 所列的几种依赖类型。

表 1 SCO-GADL 中的数据依赖类型

数据依赖类型	求解方法
const	常量数据
bsh	使用一段 beanshell 脚本求解
java	使用一个具有特定接口的 java 类求解
shell	通过一个本地 shell 脚本求解
soap	通过一个 SOAP 调用的 Web Service 求解
sigre	通过一个 SIGRE 任务求解

SCO-GADL 中不同依赖类型的数据项求解方法是不一样的。例如,const 数据项具有一个字符串常量,该常量直接由 data 元素的文本所确定;soap 数据项元素应当使用一个 endpoint 子元素来指定其所引用的 Web service 的访问点 URL,并使用一组 refMappings 子元素来指出该数据项的所有前驱数据项。很显然,const 数据项不应该具有前驱数据项。表 2 给出了所有依赖类型的数据项应当定义的子元素。

表 2 不同依赖类型的数据所应定义的子元素

数据依赖类型	应定义的子元素
Const	(无)
bsh	〈refMapping name="name"〉数据项名〈/refMapping〉 〈script〉beanshell 脚本〈/script〉
java	〈refMapping name="name"〉数据项名〈/refMapping〉 〈class〉实现了应用引擎定义的 org. thgrid. sco_gadl. IData 接口的一个 java 类〈/class〉 refMapping 所指定的数据项将存放在一个 java. util. Map 中并作为参数传递给 IData. process(Map parameter). 方法
shell	〈refMapping〉数据项名〈/refMapping〉 〈cmd〉本地 shell 命令〈/cmd〉 refMapping 所指定的数据项将依照其出现的次序通过命令行参数传递给 shell 命令
soap	〈refMapping name="name"〉数据项名〈/refMapping〉 〈endpoint〉WSDL 描述的访问点〈/endpoint〉 refMapping 指定的数据项用作 WSDL 输入消息的参数
sigre	〈refMapping name="name"〉数据项名〈/refMapping〉 〈endpoint〉SIGRE 任务管理服务的访问点〈/endpoint〉 〈job〉SIGRE 任务引用描述〈/job〉 refMapping 指定数据项作为 SIGRE 任务的输入参数使用 SIGRE 是一种为空间信息网格(SIG)构建的运行时环境[10],提供一种异步的"SIGRE 任务"用以实现长时间的处理服务

图 2 是一个使用 SCO-GADL 描述的应用例子,该应用由一个 const 数据项、两个 java 数据项和一个 soap 数据项组成,其中的 Show 数据项就是应用的最终结果,如图 3 所示。

```
〈application target="Show"〉
    〈data name="Show" type="soap"〉
        〈refMapping〉Inversion〈/refMapping〉
        〈endpoint〉http://localhost:8080/test/services/show? wsdl
〈/endpoint〉
    〈/data〉
    〈data name="Inversion" type="java"〉
        〈refMapping〉Query〈/refMapping〉
        〈endpoint〉org. thgrid. test. NDVIInversion〈/endpoint〉
    〈/data〉
    〈data name="Query" type="java"〉
        〈refMapping〉Source〈/refMapping〉
        〈class〉org. thgrid. test. DataQuery〈/class〉
    〈/data〉
    〈data name="Source" type="const"〉
        Sensor=MODIS
    〈/data〉
〈/application〉
```

图 2 SCO-GADL 描述的应用示例

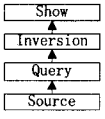


图 3 示例应用的数据依赖图

4 面向科学计算的网格工作流 SCO-GADL

为了支持 SCO-GADL 描述的网格应用运行,本文还用 java 语言实现了一个 SCO-GADL 的应用引擎,并将其部署在 Apache Tomcat 5.5.28 之上。这个应用引擎可以解释和运行 SCO-GADL 描述的各种网格应用,控制这些应用的运行,并为这些应用提供基于 Web 的用户交互支持。为了支持 SCO-GADL 定义的各种不同的数据依赖关系,应用引擎采用核心-插件结构,由核心控制整个引擎的运行,调用不同的插件来完成不同依赖类型数据项的求解工作。应用引擎的插件可以使用本地资源实现,也可以引用其他的远程资源。例如,一个数据依赖类型为 java 的数据项可以通过在本地运行一段 java 代码而得到求解;而一个数据依赖类型为 sigre 的数据项则要通过 SIGRE 运行时环境远程启动一个 SIGRE 任务来求得其结果。不同种类的插件在核心的调度下协同工作,依照应用中各数据项的依赖关系通过引用本地和远程的各种资源求解每个数据项,从而完成整个应用的运行。应用引擎的体系结构如图 4 所示。

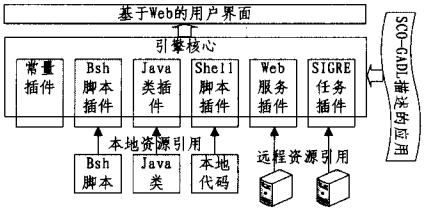


图 4 应用引擎的体系结构

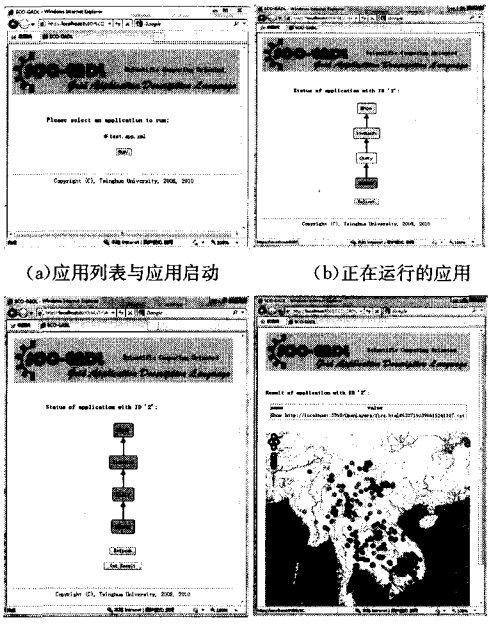
在应用引擎中,各种插件都要实现一个 java 接口 org. thgrid. sco_gadl. IData,该接口定义了一个方法 Object solve(java. util. Map parameter),引擎核心会调用这个方法来完成数据项的求解工作。只要将一个新的实现了 org. thgrid. sco_gadl. IData 接口的 java 类加入 SCO-GADL 引擎的配置文件,SCO-GADL 引擎就能够支持新种类的数据项的求解,从而扩展应用引擎的功能。在这些插件的支持下,应用引擎核心可以读取以. app. xml 为扩展名的应用描述并对这些应用描述进行解释,支持和控制其中数据项的求解工作,完成应用的执行。

(下转第 34 页)

Manufacturing Systems[J]. IEEE Trans. on Automatic Control, 1997,42(3):420-425
[9] 林闯,单志广,任丰原. 计算机网络的服务质量(QoS)[M]. 北京:清华大学出版社,2004
[10] 郑大钟,赵千川. 离散事件动态系统[M]. 北京:清华大学出版社,2001
[11] 李响,孙华志. 网格资源选择性配置研究[J]. 计算机科学,2010,37(4):114
[12] 官荷卿,张文博,魏峻,等. 一种应用敏感的 Web 服务请求调度策略[J]. 计算机学报,2006,29(7):1189-1198

(上接第 30 页)

为了测试 SCO-GADL 及其应用引擎,图 2 所示的应用也被部署到了 SCO-GADL 应用引擎中,命名为 test. app. xml。这个应用的运行过程如图 5 所示。



(a)应用列表与应用启动　(b)正在运行的应用

(c)应用运行结束　(d)应用的结果

图 5　示例应用

图 5 显示了一个由 SCO-GADL 描述的森林火灾监测应用,该应用首先根据 source 数据项中描述的条件搜索合适的 MODIS 数据,然后对该 MODIS 数据进行反演处理以获得疑似火点的经纬度,最后再通过一个 Web 服务将这些疑似火点显示在地图上以供查找确认。在 SCO-GADL 应用引擎的支持下,该应用得到成功运行并得到了正确的结果,这表明 SCO-GADL 具有足够的能力来表达各种科学计算应用。

结束语　虽然已经有很多项目试图提供包括被广泛使用的网格工作流在内的各种工具和库来降低网格应用开发的难度,但是网格应用开发的门槛依旧是阻止网格成为科学计算基础设施的重要因素。本文首先研究了当前网格应用开发所使用的基于流程或基于 Petri 网的应用描述模型,并提出一种与之不同的基于数据依赖关系的应用描述方法,称为"基于数据依赖的应用描述模型"。在该模型中,应用可以以被称作"数据依赖图"的有向图表示,而且可解应用的数据依赖图一定是一个有向无环图。在这个模型基础上,本文提出了一种更加接近于科学工作者们思维方式的网格应用描述语言 SCO-GADL,并给出其 XML 描述。在 SCO-GADL 中,一个应用可以由一个根元素为 application 的 XML 文档所描述,application 元素有一系列的 data 子元素,用来描述应用数据集中的所有数据项。最后,本文设计和实现了一种采用核心-插件结构的应用引擎,用来支持 SCO-GADL 描述的应用的运行。在应用引擎上运行示例程序的实验表明,SCO-GADL 在保证描述科学计算应用的能力的同时,提供了更符合科学工作者思维方式的网格应用描述方法,大大降低了网格应用开发的难度,使得网格成为科学工作者进行科学研究的一种有力工具。

参 考 文 献

[1] Berman F, Chien A, Cooper K, et al. The GrADS Project: Software Support for High-Level Grid Application Development [J]. International Journal of High Performance Computing Applications, 2001, 15(4): 327-344
[2] Frey J, Tannenbaum T, Livny M, et al. Condor-G: A Computation Management Agent for Multi-Institutional Grids [C]// 10th IEEE International Symposium on High Performance Distributed Computing (HPDC-10 '01). 2001: 55-66
[3] Karonis N T, Toonen B, Foster I. MPICH-G2: A Grid-enabled implementation of the Message Passing Interface [J]. Journal of Parallel and Distributed Computing, 2003, 63(5): 551-563
[4] Fox G C, Gannon D. Workflow in Grid Systems [J]. Concurrency and Computation: Practice and Experience, 2006, 18(10): 1009-1019
[5] Majithia S, Shields M, Taylor I, et al. Triana: A Graphical Web Service Composition and Execution Toolkit [C]// IEEE International Conference on Web Services (ICWS'04). 2004: 514
[6] Altintas I, Berkley C, Jaeger E, et al. Kepler: An Extensible System for Design and Execution of Scientific Workflows [C]// 16th International Conference on Scientific and Statistical Database Management (SSDBM'04). 2004: 423
[7] Oinn T, Addis M, Ferris J. Taverna: a tool for the composition and enactment of bioinformatics workflows [J]. Bioinformatics, 2004, 20(17): 3045-3054
[8] OASIS Standard, Web Services Business Process Execution Language Version 2.0 [OL]. (11 April 2007). http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html, 2010-7-26
[9] Guan Z, Hernandez F, et al. Grid-Flow: a Grid-enabled scientific workflow system with a Petri-net-based interface [J]. Concurrency and Computation: Practice and Experience, 2005, 18(10): 1115-1140
[10] Huang Z C, Li G Q, Du B, et al. SIGRE-An Autonomic Spatial Information Grid Runtime Environment for Geo-computation [C]// The 7th International Symposium on Advanced Parallel Proceeding (APPT 2007). 2007: 319-326