一个C语言安全子集的可信编译器

王 蕾 石 刚 董 渊 白晓颖 王生原
(清华大学计算机科学与技术系 北京 100084)

摘 要 以安全关键领域的安全标准为依托、安全相关软件的语言编码和编译要求为指导,进行了以下几方面的研究和探索:首先对形式化验证可信编译技术进行分析研究,特别着重当前广受关注的经过验证的CompCert编译器。然后以我国安全领域C语言安全子集标准《航天型号软件C语言安全子集》为依据构造测试用例、创新测试方法,并以此对CompCert编译器进行测试评估。之后依据测试结果,为CompCert编译器增加未支持的C语言标准特性,裁剪不符合C语言安全子集要求的特性,构建符合C语言安全子集标准的可信编译器。最后的实测结果表明,所实现的编译器符合C语言安全子集标准的要求,且没有降低C代码的执行效率。

关键词 可信计算,CompCert,C安全子集,经过验证的编译器
中图法分类号 TP314 **文献标识码** A

Trusted Compiler for Safe Subset of C Language

WANG Lei SHI Gang DONG Yuan BAI Xiao-ying WANG Sheng-yuan
(Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China)

Abstract Using the safety standards of safety-critical areas as the basis, safety-related software coding and compiling requirements as the guideline, this paper introduced the following aspects of research and exploration: first analysed the formal verification technique for trusted compiler, especially emphasized the CompCert verified compiler which is widely concerned, then constructed test cases according to safe subset of C language for space armament software to test and assess the CompCert compiler with the new method. Based on the test results, the project added unsupported features of C language to CompCert and cuts off the features which do not meet the safe subset of C language requirements, built a verified compiler for the safe subset of C language at last. The experimental results show that the compiler we implemented complies with safe subset of C language, but it do not increase the execution time of C code.

Keywords Trustworthy computing, CompCert, Safe subset of C language, Verified compilers

1 引言

编译可信一直是核工业、航空航天等关键领域关注的焦点。当前主流C编译器则关注编译效率及代码运行的高效性,虽然已有许多改变,但距离编译高可信需求还有一定距离。依据相关领域标准要求,实现编译可信需要限制C语言使用、严格要求编译器使用、对比编译前后代码、需求测试、缺陷测试等[1,2]。然而,这些做法并不能从根本上解决问题。近年来,人们将目光聚焦到通过形式化验证的方法来解决问题。实际上,目前日渐成熟的编译器验证技术已基本能够达到编译高可信的要求,然而其发展现状还未能符合相应领域的应用安全标准。本文以此为出发点,利用较成熟的验证编译技术,与相应的C语言安全子集相结合,使其更符合相关领域的高可信要求。

2 高可信编译问题

安全可靠在核工业、航空航天等领域的软件工程项目设计实现过程中一直是备受关注的焦点。为此,该领域制定了非常严格的安全标准和项目实施流程,对项目规划、规范制定、计划流程、开发实现、验证测试、应用培训等各阶段都进行了严格定义和说明[1,2]。语言的使用规范及编译器的应用要求是其中非常重要的部分。标准规范中对于语言的使用进行了严格限制和定义,《航天型号软件C语言安全子集》就是以此为基础制定的语言规范。而标准规范中则严格要求了编译器的成熟度及稳定性,编译器必须忠实地反映源语言的代码结构和语义,以方便编译前后的代码审查比较,确保编译后代码的可信。以此为出发点,本文尝试以形式化验证技术为基础,将语言规范与经过验证的编译器相结合,以提高编译的可

到稿日期:2012-11-30 返修日期:2013-03-11 本文受国家自然科学基金(61170051,61272086,90818019),国家核高基项目(2012ZX01039-004-08-02),清华大学基础研究基金(2010Z05097),铁道部-清华大学科技研究基金(J2010Z064)资助。

王 蕾(1983—),男,硕士生,主要研究方向为编译器的形式化验证、微内核验证等,E-mail:Leopardguo25@gmail.com;石 刚(1972—),男,博士生,主要研究方向为形式化方法、系统设计和云计算;董 渊(1973—),男,副教授,主要研究方向为编译器设计及验证、操作系统设计及验证等;白晓颖(1973—),女,副教授,主要研究方向为软件工程、软件测试等;王生原(1964—),男,副教授,主要研究方向为编译器设计及验证、分布对象计算等。

信度，减少工作流程和工作量，达到行业规范要求。

2.1 编译形式化验证

编译器形式化验证的方法是利用严格的数学逻辑体系对编译过程语义及语言属性的保持进行证明，以便从根本上实现上述规范标准中对编译器的要求限制，确保语义的正确保持。相比其他形式化验证技术（如模型检查），定理证明更符合编译器验证的要求。这主要取决于语言编译过程有比较明确的语义保持要求及清晰的缺陷定义，这样能够更好地发挥定理证明的优势，从而实现以严谨的数学逻辑证明保证编译的高可信度。目前定理证明已经成为编译及操作系统内核验证的主要方法，成功的例子包括 CompCert[3]、seL4[4]等等。

定理证明是形式化验证的方式之一，即以数学公理为依据，按照一定的规则推理定理正确性的过程，其优势在于能够利用严谨的数学推理证明清晰的属性要求和明确的正确性标准。在使用定理证明方法时，需要高效的手段来准确定义属性、清晰描述定理及严谨推理证明，为此人们把这些推理证明及演绎过程转变为计算机符号和演算过程，从而变成自动定理证明系统。特别是以 Coq[5]、Isabelle[6]等为代表的自动定理证明系统，提高了形式化证明效率。

Coq 作为一款高效定理证明系统，为描述清晰的语言结构及构筑高效的证明环境提供了极大的支撑。该系统以函数式语言为基础，为使用者提供了一组形式化语言描述的数学定义、一套执行算法及自动证明环境。Coq 在以上功能结构的帮助下可以有效地定义各种数据结构、执行算法及严谨的自动证明逻辑。典型项目有 CompCert、四色定理的证明等等。

2.2 CompCert

在某些情况下，编译器的编译过程可能会成为从程序规范到可执行代码这个完整变换链条中最为薄弱的环节。一个有效的办法就是利用形式化的方法保证编译器在进行翻译变换的同时保持源代码的语义。CompCert 就是一款经过形式化验证的真正严格保持 C 代码语义的验证编译器。

目前，CompCert 编译器已经成为 Coq 系统的标志性项目[5]。该项目充分发挥了 Coq 系统的特性，将 C 语言的语义客观表达出来，并在保持语义的前提下生成汇编代码。该编译器生成的代码高效、紧凑，能够满足关键嵌入系统的需求。且 CompCert 是一个支持多遍编译、优良的寄存器分配算法和基本优化的编译器[7]。其基本编译流程依序为：gcc 预处理过程，C 语言前端处理，经过形式化验证的主框架，汇编代码生成。经过形式化验证的主框架是 CompCert 的核心，最具价值。源语言 Clight[8]（1.8 版本以后为 CompCert C[9]）是一种兼容于关键嵌入式软件推荐使用的 C 语言子集，输出为类汇编的形式化表达。

2.3 CompCert 测试评估

CompCert 项目是一个极具代表性的编译验证项目，其极高的复杂度使之相对于 C 语言标准《Programming languages—C》[10]（下文统称 C90）来说可能还存在一些不支持边界，而对于行业标准《航天型号软件 C 语言安全子集》[11]（下文统称 C 安全子集）来说会有更多不支持。

目前，普遍认同的 C90 标准已经被我国采纳为中华人民共和国国家标准《程序设计语言 C》[12]。CompCert 项目就是以 C90 为规范依据来设计 C 语言编译过程，且 C 语言的安全子集同样是以该标准为蓝本进行限定、规范，所以必须对其进行 C90 规范测试。

测试目标：CompCert-1.9；

主要测试部分：环境、语言、库；

环境、语言部分用例：使用标准中注释代码或 IBM 提供的标准 C 工程测试用例。

经过测试后所得的测试结果如下所示：

测试 C90 共 520 余个标准点；

测试不支持的特性点有 22 个；

CompCert 强烈建议不使用的特性有 3 个。

3 C 安全子集测试评估

3.1 C 安全子集测试

C 安全子集的范围以前述《航天型号软件 C 语言安全子集》为规范依据。该标准以汽车工业软件可靠性联会 MISRA 提出的 MISRA C[13]编程规范为基础，该规范定义的 C 语言被认为是易读、可靠、可移植、易于维护的。C 安全子集将 MISRA C 与航天型号软件的特点相结合，重新定义了一系列 C 语言软件的编程准则，为安全相关领域的 C 语言软件提供了相应的安全语言规范和编译要求。C 语言安全子集是在标准 C 即 C90 的范围框架内划分的一个相对规整、稳定、安全的可信子集。在该规范准则中定义了强制类和推荐类两种类型，其中强制类为必须执行的准则，推荐类为参照执行准则。

C 语言安全子集是一个规定严苛的 C 语言规范，需要依据该子集对 CompCert 编译器进行严格的测试评估。以下为 C 安全子集测试结果。

测试目标：CompCert-1.9；

主要测试部分：C 安全子集各规范类；

各部分用例：标准中注释代码。

经过测试所得的测试结果如下：

测试 C 安全子集声明定义类、版面书写类、分支控制类等 15 个分类共 124 个标准点；

强制类中有 13 个支持点、77 个不支持点、2 个异常点；

推荐类中有 2 个支持点、30 个不支持点。

由以上所示的测试结果可明确清晰地发现 CompCert 编译器并没有支撑 C 语言安全子集，存在大部分的不支持点。

3.2 C 安全子集评估

从以上所示的测试结果不难发现，CompCert 编译器距离行业应用还有一些差距。CompCert 编译器的覆盖范围要小于 C90 标准的范围，且并没有完全覆盖 C 安全子集。C 标准、CompCert C 及 C 安全子集（Sub C）之间的范围示例如图 1 所示。

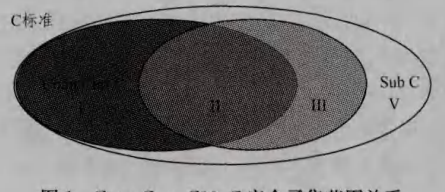

图 1 CompCert、C90、C 安全子集范围关系

如图所示，C90 标准作为最大范围包括后两者，CompCert C 与 C 安全子集之间存在较大交集但互不覆盖。这对以后的工作具有非常重要的指导性意义。需要做的主要工作包括为 CompCert 裁减部分 I 并增加部分 III，最后需要实现

的部分为 II 与 III，也就是说与 C 安全子集相符合(Sub C)。图中所示各部分包括以下主要内容。

部分 I：C 安全子集限定不支持的声明定义、版面书写、分支控制、指针使用、跳转控制、运算处理等部分所包括的 C 语言特性。

部分 II：CompCert 支持的 C 语言子集与 C 安全子集的交集，包括 C 语言基本类型、语法、语句、库函数等。

部分 III：为 C90 标准中明确定义的超长数据类型、三联符序列、环境特殊限定值、结构体特殊类型、浮点类型一般表示、前导文卷特殊定义、特殊算术库函数等 C 语言特性。

部分 V：该区域是两子集皆未覆盖的点，多重标号声明、空 switch 语言使用、参数指针使用、指针嵌套等包括在其中。

4 C 安全子集可信编译

针对三者之间的特殊关系，C 安全子集可信编译器的实现方法如图 2 所示。

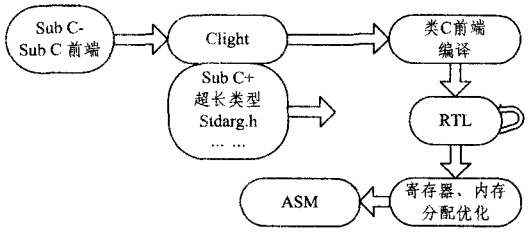

图 2 C 安全子集可信编译的实现

CompCert 经过验证的编译过程自 Clight 开始经过类 C 前端编译、RTL 中间优化转换、后端寄存器、内存分配优化及 ASM 生成等过程。为实现 C 安全子集可信编译，需要在其原有编译过程中增加超长类型、变长参数等 C 语言特性，并在其前端增加 C 语言安全子集来分析前端修剪不需要的 C 语言特性。C 语言安全子集实际并不支持可变长参数的编码书写规范。但是在实际的使用过程中，该方法会带来一些便利，在某些特定场合下很难被替代，因此这里加入了对变长参数库 stdarg. h 的支持，并在翻译过程中保证其正确性，设置相应的编译提示符，以在编译时开启该特性的编译。

4.1 增加 C 特性

这一过程主要以 C90 标准测试结果为依据。其难点包括为 CompCert 编译过程增加不支持的 C90 特性，其中包括：long long/long double 长类型编译支持、变长参数函数实现等等。

Long long/long double 长类型编译是实现 C 标准编译过程的难点，这里以 long long 类型为例进行说明。因为 CompCert 经过验证的编译过程抽象语法树 AST 中只定义了两种基本类型：int 及 float，该类型贯穿整个编译过程并对应其后的寄存器类型，其他类型都是这两种基本类型的衍生。

以 32 位 x86 架构为例，其代表 32 位整型和 32 位浮点型。而整型是一个非常特殊的类型，C 语言中大部分其它整型都可以转换为该基本整型。而 long long 类型是一个特例，其特殊长度要求使其不能被基本类型代替，且在 CompCert 的编译后端很难符合基本类型所对应的整型寄存器的使用要求。因此需要在基础框架中增加全新的 Tint64 长类型，为其分配合适的存储空间及寄存器。CompCert 类型转换如图 3 所示，CompCert 主要类型如图 4 所示。

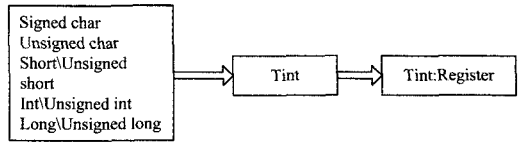

图 3 CompCert 类型转换

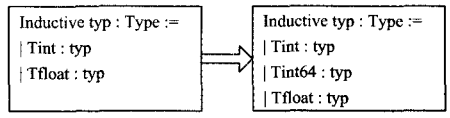

图 4 CompCert 主要类型

在增加了新的类型后，需要在各编译过程为其增加相应的支持，并加以证明。特别在存储分配过程中要注意类型长度为 64 位。而在寄存器分配过程中可采用两种方案：一种方案为超长类型对应可用的超长类型寄存器，另一种为超长类型对应的多个短类型寄存器。

方案一实现时要求相应的计算机体系结构具备相应的寄存器，以供分配使用。当方案一不能实现时，就要采取方案二为超长类型分配多寄存器。在方案二实现过程中需要特别注意寄存器分配冲突，因此在 Coloring 算法实现寄存器分配之前要构建一个基于超长类型与相应普通类型冲突的冲突图。超长类型寄存器分配如图 5 所示。寄存器分配类型冲突说明如图 6 所示。

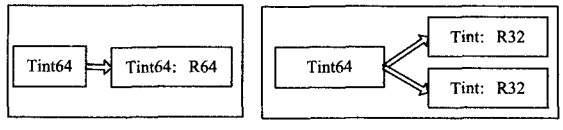

图 5 超长类型寄存器分配

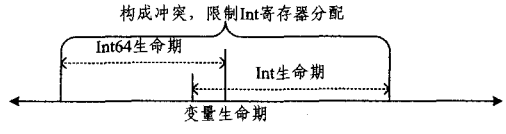

图 6 寄存器分配类型冲突说明

通过以上方法构建的冲突图能够保证两种类型通过上文所述方法二进行寄存器分配时不会发生冲突，并通过 Coloring 的 validation(确认)验证。在实现了超长类型寄存器、存储分配之后，需要为超长类型定义相应的计算，最终实现超长类型编译。

变长参数是另一个难点，主要由 CompCert 的形式化验证带来的问题在前端及中端处理过程中指针传递不便造成。以下需要明确变长参数的库函数表述：

```
#include "stdarg. h"
va_list arg_ptr;
va_start(arg_ptr,start);
va_arg(arg_ptr,type);
va_end(arg_ptr);
```

由于 CompCert 的预编译过程采用 gcc 提供的预编译功能，而变长参数库函数经过 gcc 预处理之后进行了初步加载，因此为了适合其预处理之后的结果，需要将变长参数函数进行指针化处理，处理过程如下：

```
_INTSIZEOF(n)=>((sizeof(n)+sizeof(int)-1)&~(sizeof(int)-1))
va_start(arg_ptr,start)=>(arg_ptr=(va_list)&start+_INTSIZEOF(start))
```

va_arg(arg_ptr,t)=>(*(t*)((arg_ptr+=_INTSIZEOF(t))-_INTSIZEOF(t)))

va_end(arg_ptr)=>(arg_ptr=(va_list)0)

通过以上处理过程，将变长参数问题转换为函数参数指针操作过程。va_start 读取变长参数起始参数地址，调用 va_arg 来读取下一个参数地址，va_end 停止变长参数调用。这里就涉及到函数调用后的栈指针操作。而 CompCert 编译器的函数调用栈处理与传统的基址压栈操作模式并不相同，采用了直接栈帧操作的方式。在调用函数的栈帧中由高到低分别存储的是自身变量、被调用函数的参数、下一条指令的返回地址。在被调用函数指针操作中没有了传统的基址寄存器的辅助，给变长参数实现过程中参数指针的读取操作带来不便；且由于 CompCert 验证式方法的限制，在其前端验证操作时对参数指针操作进行了预读取的特殊处理，该处理需要进行修改。

为了解决上述问题，需要将 CompCert 读取参数的指针操作进行简化，该操作进行了预读取操作；同时在后端需要记录变长参数的起点相对位置，以适应 CompCert 特殊的函数栈帧结构。

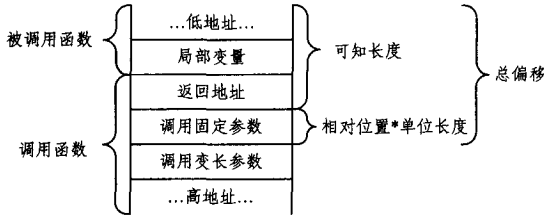

图 7　CompCert 变长参数栈结构

除了以上所述的难点，还增加了 auto 属性、register 存储属性等，修改了指针变量作用域冲突、全局变量作用域冲突等问题。

4.2　C 安全子集编译

在经过验证的编译过程基础上，覆盖 C 安全子集是项目重点，需要对该编译过程进行特性裁剪以符合 C 安全子集的标准范围。

实现 C 安全子集的过程以 C 安全子集的测试为主要依据，针对测试结果所示 109 个不支持特性逐个进行功能修剪。对标准中的强制类做 Error 处理，而推荐类做 Warning 处理。在对验证编译过程的前端进行属性的微调后，又构建了一个与 C 安全子集相匹配的翻译前端，如图 8 所示。

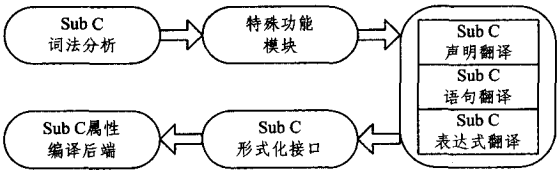

图 8　与 C 安全子集匹配的前端结构

图 8 结构中处理了 Sub C 定义的大部分语法、静态语义。该结构的核心模块是 C 安全子集翻译模块，该模块除了包括 C 安全子集声明、语句、表达式等基本解析翻译功能子模块外，还包括函数定义处理及静态语义分析等等。

为实现 C 语言安全子集的可信编译，本项目的主要工作量包括：

超长类型实现代码 1800 行；

变长参数实现代码 800 行；

C 安全子集翻译前端实现代码 3200 余行。

5　实验测试

CompCert 编译器的形式化方法虽然得到广泛认同，但是其与关键行业的应用要求还存在一些距离。经过形式化验证的编译方法与 C 语言安全子集相结合后最大程度地符合了行业应用标准。为了更好地说明该问题，以 C 安全子集为主要依据对 C 安全子集可信编译器进行了测试，所得结果如表 1 所列。

表 1　C 安全子集可信编译器测试结果

类别	强制类			推荐类		合计
	正确	错误	异常	正确	错误	
声明定义类	21			8		29
版面书写	7					7
分支控制类	8					8
指针使用类	3			2		5
跳转控制类	3			1		4
运算处理类	18			4		22
过程调用类	9			3		12
语句使用类	3			8		11
调用返回类	5					5
程序注释类		1		1	1	3
循环类	2			3		5
类型转换类	1			2		3
初始化	3		1			4
比较判断类	3			1		4
名称使用类	6			2		8
合计	92	1	1	35	1	130
	94			36		

由表 1 可知，C 安全子集可信编译器已经基本实现了对于 C 语言安全子集的支持，对大部分问题进行了改进并保持正确处理。强制类型中出现的 1 个不支持特性及推荐类中出现的 1 个不支持特性都属于程序注释类，分别为禁止使用嵌套注释、避免使用不加分析的注释，这两点通过编译器判断实现，严格情况下代码注释处理应在工程审查校验阶段实现。出现的 1 个异常情况是禁止使用未经赋值的变量，该情况的判断比较复杂，特别是变量在判断型语句中赋值不会进行任何处理。将 C 语言安全子集可信编译器针对 MISRA C 测试集进行测试评估后同样得到了比较好的效果，即共 124 个测试特性、12 个未支持特性，这主要是两标准间的不同点及代码特殊预处理要求带来的问题。

从编译效率方面考虑，因为整个工作并没有修改 CompCert 主干框架及寄存器、栈分配基础，所以生成的新代码在运行效率方面并不会发生太大变化。以 CompCert 项目提供的测试源码为例，将其代码进行 C 安全子集修改后的测试结果如图 9 所示。

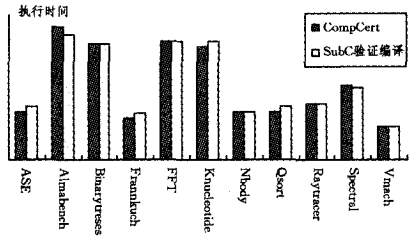

图 9　C 代码执行时间对比

从图 9 给出的结论可以明显看出，C 安全子集可信编译器所生成的代码并没有明显降低 C 代码的执行效率。这样既保证了 C 代码编译的可信度，又没有降低 C 代码的执行效率。

结束语 为了将 CompCert 的验证式方法引入具体实践以符合安全领域的应用要求，这里尝试以行业标准为依据对其经过形式化验证的编译方法进行改造、升级。首先，以 C90 及 C 安全子集为依据对 CompCert 进行测试，充分了解经过验证的编译器特性。然后，以测试结论为指导，对该编译器经过验证的编译过程进行 C 标准改造及实现 C 安全子集编译，最后形成 C 安全子集可信编译器。通过测试应用发现，该编译器已经基本符合 C 安全子集的编译要求，且没有降低 C 代码的执行效率。

参 考 文 献

[1] CEI/IEC 60880:2006, Nuclear power plants. Instrumenation and control systems important to safety. Software aspects for computer-based systems performing category A functions [S]. Geneva:CEI/IEC,2006

[2] RTCA/DO-178B, Software Considerations in Air-borne Systems and Equipment Certification[S]. Washington DC:RTCA, 1992

[3] INRIA. CompCert development team. CompCert [EB/OL]. http://compcert.inria.fr/index.html,2012-07

[4] Klein G, Elphinstone K, Heiser G, et al. seL4: Formal verification of an OS kernel[C]// Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles. NY, USA: ACM New York,2009:207-220

[5] Coq development team. The Coq proof assistant [EB/OL]. http://coq.inria.fr/,2012-07

[6] Nipkow T, Paulson LC, Wenzel M. Isabelle/HOL:a proof assistant for higher-order logic[M]. London:Springer-Verlag,2002

[7] Xavier L. Formal verification of a realistic compiler[J]. Commun ACM,2009,52(7):107-115

[8] Sandrine B, Xavier L. Mechanized semantics for the Clight subset of the C language[J]. Journal of Automated Reasoning, 2009,43(3):263-288

[9] Xavier L. The CompCert C verified compiler Documentation and user's manual, version 1.11[R]. Paris:INRIA Paris-Rocquencourt,2012

[10] ISO/IEC 9899-1990. Programming languages-C[S]. Geneva: ISO/IEC,1990

[11] GJB 5369-2005. 航天型号软件 C 语言安全子集[S]. 北京：国防科学技术工业委员会,2005

[12] GB/T 15272-94. 程序设计语言 C[S]. 北京：国家质量监督检验检疫总局,1994

[13] MISRA. The MISRA C team. MISRA C [EB/OL]. http://www.misra-c.com/Activities/MISRAC/tabid/160/Default.aspx,2012-07

(上接第 29 页)

结束语 领域应用需求的飞速增长和多样化使得数据中心的规模越来越大、越来越复杂。领域科学数据云为此提供一种解决思路。领域科学数据云的核心在于实现数据共享，而共享的前提是数据的统一标准和一致管理。领域科学数据云的实现基础是中间连接代理系统。由连接代理系统利用数据云代理模型、异构源共享模型以及资源交换消息模型构建统一的数据中心管理环境，用以支持异构存储和服务器，并提供一体化的数据传输与资源服务，实现数据中心的积木式连接、去中心化的资源融合以及存储容量扩充。通过实验验证，将领域科学数据云资源聚合模型应用在油气井科研数据共享服务是行之有效的。领域科学数据云为实现数据中心资源聚合以及从传统数据中心架构过渡到云架构提供了有价值的参考。下一步的工作是要进一步完善领域科学数据云的资源交付、自动部署以及运维管理等，以期能更好地服务于具体的领域应用。

参 考 文 献

[1] 王意洁，孙伟东，周松，等. 云计算环境下的分布存储关键技术[J]. 软件学报,2012,23(4):962-986

[2] Gray J. Rethinking data center network connectivity for new architectures [EB/OL]. http://www.searchnetworking.techtarget.com/news/2240036484/Rethinking-data-center-network-connectivity-for-new-architectures,2012-08-06

[3] 从孤岛到融合：数据中心网络架构的革命[EB/OL]. http://net.chinaunix.net/a2011/0920/1248/000001248682_2.shtml, 2012-08-06

[4] 新蓝图：云计算和数据中心融合[EB/OL]. http://www.ciotimes.com/infrastructure/sjk/61535.html,2012-10-06

[5] Meng X, et al. Improving the scalability of data center networks with traffic-aware virtual machine placement[C]// Proc IN-FOCOM. 2010: 1154-1162

[6] Greenberg A, Hamilton J R, Jain N, et al. VL2: A scalable and flexible data center network[C]// Proc. of the SIGCOMM 2009. 2009:51-62

[7] 数据中心：融合基础架构[EB/OL]. http://www.dlnet.com/datacenter/tech/78307.html,2012-08-06

[8] Li B, et al. EnaCloud: an energy-saving application live placement approach for cloud compu-ting environments[C]// Proc of International Conf on Cloud Computing. 2009:17-24

[9] Valancius V, Laoutaris N, et al. Greening the Internet with nano data centers[C]// Proc of CoNext. 2009:37-48

[10] 张伟，宋莹，阮利，等. 面向 Internet 数据中心的资源管理[J]. 软件学报,2012,23(2):179-198

[11] Chang F, Dean J, Ghemawat S, et al. Bigtable: A distributed storage system for structured data [C]// Proc. of the 7th USENIX Symp. on Operating. Systems Design and Implementation. Berkeley:USENIX Association,2006:205-218

[12] Chaganti P. Cloud computing with Amazon Web Services. Part 5:Dataset processing in the cloud with SimpleDB[OL]. http://www.ibm.com/developerworks/library/ar-cloudaws5/,2009

[13] A brief overview of the Cassandra storage engine [EB/OL]. http://cassandra.apache.org/,2012-10-05

[14] Waldspurger C A. Memory resource management in VMware ESX server[C]// Proceedings of the 5th Symposium on Operating Systems Design and Implementation(OSDI2002), ACM Operating Systems Review, Winter 2002 Special Issue. Boston, MA, USA, Dec. 2002:181-194

[15] Velte A, Velte T. Microsoft Virtualization with Hyper-V[M]. McGraw-Hill Inc. New York, NY, 2009

[16] 钱琼芬，李春林，张小庆，等. 云数据中心虚拟资源管理研究综述[J]. 计算机应用研究,2012,29(07):2411-2415

[17] 张成峰，谢长生，罗益辉，等. 网络存储的统一与虚拟化[J]. 计算机科学,2006,33(06):11-14

[18] 初佃辉，王显志，王忠杰，等. 面向个性化需求的虚拟服务资源整合方法[J]. 计算机学报,2011,34(12):2370-2380