基于软件共享存储的 Co-Array Fortran 编译器实现

黄 春
(国防科学技术大学计算机学院 长沙 410073)

摘 要 Co-Array Fortran(CAF)已经成为 Fortran 语言标准的一部分,在科学计算领域逐渐被接受。基于软件共享存储实现了一个 CAF 编译器,其通过直接的数组赋值实现 Co-array 数据通信,利用数据垫塞技术提高数据局部性,减少伪共享,优化 CAF 程序性能。典型科学计算程序测试表明,CAF 能够获得和 MPI 相当的性能。

关键词 Co-Array Fortran,映像,协同数组,数据垫塞
中图法分类号 TP314 **文献标识码** A

Design and Implementation of Co-Array Fortran Compiler Based on Software Shared Memory

HUANG Chun
(School of Computer Science, National University of Defense Technology, Changsha 410073, China)

Abstract Co-Array Fortran (CAF) has become part of Fortran Programming Language Standard, and has been widely accepted in scientific computing community. The paper presented the design and implementation of a CAF Compiler based on software shared memory. Several techniques were presented to improve the performance of CAF programs, including Co-array data communication by direct array assignment, Data Padding for raising locality and reducing false share. The benchmark testing shows that the performance of CAF programs is similar to that of MPI programs, with our CAF compiler.

Keywords Co-Array Fortran, Image, Co-Array, Data padding

1 引言

处理器已经进入多(众)核时代,开发并行程序是发挥处理器性能的关键。目前,OpenMP 和 MPI 代表了多线程和消息传递这两种应用最广泛的编程模型。但是尽管 OpenMP 编程简单,却只适合于共享存储系统,可扩展性受限;MPI 可扩展性高,但编程复杂。

Co-Array Fortran(简称 CAF)[1]由 Numrich 和 Reid 在 1998 年提出,是基于 Fortran90/95 语言扩展的并行程序设计语言,支持分割全局地址空间(Partitioned Global Address SPACE,PGAS)的单程序多数据(SPMD)并行编程模型。和 OpenMP 相比,CAF 不仅可用于共享存储系统,还适合分布存储系统,此外显式的数据分割和通信、性能和可扩展性优于 OpenMP。和 MPI 相比,CAF 通过使用赋值语句对全局编址的存储空间直接进行存取实现数据通信,大大降低了编程复杂度。2005 年 Fortran 组织大会决定,接受 Co-Array Fortran 进入下一代的 Fortran 标准规范(Fortran2008)[2]。

Cray 公司在 T3E 系统上首次实现了 CAF 编译器[3],RICE 大学基于 Open64 开发了第一个开源的、可移植的 CAF 编译器。它采用源到源转换的方法,将 CAF 程序转换为 Fortran90 程序加上单边通信库 ARMCI 或者 GASNet 的调用[4],能够在 Linux 集群和 Cray XT 系统上运行。源到源转换工具 caf2omp 将 CAF 的一个子集转换为 OpenMP 程序[5]。GNU 的 Fortran 编译器 gfortran 从 4.5 以后的版本也部分实现了 CAF[6],将 CAF 转换为 MPI 程序。随着 CAF 成为 Fortran 语言标准的一部分,商用编译器也开始支持 CAF,例如 Intel 最新的 Fortran 编译器支持 Fortran2008 标准,包含对 CAF 的支持[7]。

分布存储并行系统是目前最具吸引力的主流并行计算平台,目前在该类型的系统上实现 CAF,均采用了将 CAF 转换为 Fortran 程序加上库调用的方法。本文提出了一种在分布存储系统上基于软件虚拟共享存储层实现 CAF 的方法。该方法通过直接的数组赋值实现 Co-Array 数据通信,利用数据垫塞技术提高数据局部性,减少伪共享,优化 CAF 程序性能。测试表明,CAF 能够获得和 MPI 相当的性能。随着 CAF 本身的不断完善,其可用性和性能还将进一步提高。

2 Co-Array Fortran 语言

CAF 程序采用 SPMD 执行模式,一个 Co-Array Fortran 程序就好像被复制好多次一样,并且每一份副本异步执行。每一份副本都有自己的数据集,称为"映像(image)"。Co-Array Fortran 在 Fortran95 数组语法的基础上,在原来数组的最后添加了在方括号内的额外下标,以便清晰、直接地表示跨映像的数据访问。

2.1 程序映像

CAF 的每一个映像异步执行,并且在程序的执行过程

到稿日期:2011-02-28 返修日期:2011-04-27 本文受"核高基"国产 CPU 编译系统及工具链项目(2009ZX01036-001-003)资助。
黄 春(1973—),女,博士,副教授,主要研究方向为并行程序设计环境、先进编译技术等,E-mail:chunhuang@nudt.edu.cn。

中，映像的数目是固定的(num_images()个映像)。映像的编号从1开始，程序员可以通过内部函数 this_image 来获得调用此函数的映像的编号。程序员通过显式使用 Fortran95 的控制结构和 CAF 内部同步过程，例如 sync_memory、sync_team 等来控制在映像间的执行序列。

2.2 数据对象

每个映像都有自己的数据对象集，可以通过正常的 Fortran 方式来访问它们。CAF 定义了新的全局可见的数据类型——协同数组(Co-Array)。Co-Array 的维度(co-dimension)使用方括号声明，紧跟在圆括号(正常的 Fortran 数组声明)后或者变量名之后。例如：

```
real,dimension(20)[8,*]::a
real::c[*],d[*]
```

除非数组是可分配数组，否则方括号中的最后一维必须是"*"。所有分散在映像上的数据对象组成一个数组，称为 Co-Array。方括号中的每个下标对应着 co-dimension 的一维，可以通过圆括号下标后跟方括号下标(或只有方括号下标)对 Co-Array 进行寻址，访问 Co-Array 数组元素，例如：

```
d[3]=c
a(:)[2,3]=c[1]
```

方括号中的下标映射到映像上的方式与 Fortran 语言中数组下标映射到本地存储器中的方式相同。例如，a(5)[3,7]引用的是映像31上的 a(5)。

2.3 数据对象的访问

无论数据对象是否为 Co-Array，每一个数据对象都存在在每个映像上。Co-Array 的引用有两种方式：本地引用和跨映像的引用。在表达式中，没有方括号的引用总是引用执行该表达式的映像上的本地对象。例如：

```
REAL,DIMENSION(100)[*]::A,B,C
A(I)=B(I)+C(I) ！本地引用 A,B,C
```

如果指定了方括号，则引用方扩号中的下标指定的映像上的对象。例如：

```
A(I)[IP]=B(I)+C(I)  ！引用映像"IP"上的 A，本地引用 B,C
B(:)=C(:)[IP2]  ！引用映像"IP2"上的 C，本地引用 B
```

2.4 执行控制

多数情况下，每个映像独立执行，并不关心其他映像的执行情况。程序员需要保证当某个映像更改了 Co-Array 数据后，其他映像不会再需要访问旧值。程序员通过调用内部同步函数来实现这些要求。内部同步过程包括数据同步 sync_memory，执行同步 sync_all/sync_team、文件同步 sync_file 以及临界区同步 start_critical/end_critical 等。

3 Co-Array Fortran 实现

分布存储并行系统在可扩展性和性能代价比上显示了强大的优势，是目前最具吸引力的主流并行计算平台。在这样的系统上，每个结点运行一个操作系统，可以利用硬件机制实现软件共享存储层，为实现 OpenMP 等共享存储程序设计语言提供支持。利用软件虚拟共享存储层，可以避免像 RICE 大学的实现方法一样，将 Co-Array 访问转换为库调用，影响编译优化，而是采用直接数组赋值的形式表示 Co-Array 之间的通信。

CAF 编译器实现需要解决的问题主要是：(1)映像管理和控制；(2)Co-Array 数据表示和实现；(3)Co-Arrays 数据的高效访问；(4)实现 CAF 定义的内部函数。

3.1 映像管理和控制

映像用进程实现。CAF 采用 SPMD 执行模式，和 MPI 一样，程序加载运行时，派生出 $n-1$ 个映像，直到程序终止时，派生的 $n-1$ 个映像才终止。使用进程实现映像，一般变量(非 Co-Array 变量)都成为映像的私有变量，编译器通过显式的软件虚拟共享层提供的数据分配函数将 Co-Array 数据放在共享区，实现 Co-Array 的全局可见。

3.2 Co-Array 表示和实现

```
real,dimension(n)[*]::x,y
real z
```

声明了 Co-Array 数组 x 和 y，一般变量 z。即每个映像上都有一个大小为 n 的一维单精度浮点数组 x 和 y，都有浮点变量 z。但是每个映像可以访问其他映像上的 x 和 y，不能访问一般变量 z(称 z 为私有数据)，如图1所示。

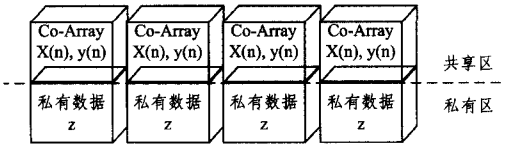

图1 CAF 程序数据

每个 Co-Array 对应有一个描述符，该描述符通过自定义的导出类型定义，编译器用导出类型的数组实现 Co-Array，数组大小为 num_images()，数组的形状和 co-dimension 定义的一致。例如 Co-Array y 数据 x 对应的描述符是：

```
type co$_x
  real,dimension(n)::x
  real,dimension(m)::padding
end type co$_x
type(co$_x)::caf_$x(num_images)
```

导出类型 co_$x 包含两个域：一个对应原数据，域的类型和大小与原数据相同(不带 co-dimension)；另一个域是数据垫塞域 padding。由于操作系统按页(假设页大小为 PAGESIZE)分布数据，因此在描述符中加入 padding 域，使每个 Co-Array 数据的大小是页大小的倍数，以保证每个映像的 Co-Array 数据都在自己所在的结点上，同时避免伪共享，提高数据局部性。对于分布存储系统而言，远程访问延迟远远大于本地延迟，如果不采用数据垫塞，会造成大量的远程访问。

padding 的数据类型和 Co-Array 数据 x 相同，大小(字节)等于：

$$\left\lfloor \frac{\text{size}(x)+\text{PAGESIZE}-1}{\text{PAGESIZE}} \right\rfloor \times \text{PAGESIZE} - \text{size}(x)$$

如图2所示，p 表示 padding 域。如果 Co-Array 数据 x 的大小正好是页大小的倍数，那么 padding 大小为零。

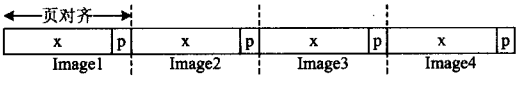

图2 Co-Array 数据分布

为了保证 Co-Array 数据能够被所有映像访问，CAF 编译器通过数据分配函数 lv_malloc 将 Co-Array 数据分配到软件共享层管理的共享区：

lv_malloc (size(caf_$x)×num_images, m×size(caf_$x),0)

分配一个大小为 size(caf_$x)×num_images 的共享区域，分配从 0 号节点开始，每个节点分配的大小为 m×size(caf_$x)。通常情况下，一个逻辑 CPU 上运行一个映像，m 为一个节点包含的逻辑 CPU 数。

3.3 Co-Array 通信

Co-Array 数据的访问有两种方式：本地 Co-Array 数据的访问和远程 Co-Array 数据的访问。

1)本地 Co-Array 访问

在表达式中，不带方括号的引用访问执行该表达式的映像上的本地对象。例如，$a=y(1)$，访问本地 Co-Array 数组 y 的元素 $y(1)$。预编译器将其转换为：a=caf_$x(my_image)%y(1)

$y(2)=b$ 写本地 Co-Array 数组 y。预编译器将其转换为：caf_$x(my_image)%y(2)=b

其中，my_image 通过固有函数 this_image 获得，在 module caf_module 中定义。

2)远程 Co-Array 访问

Co-Array 数据存放在进程可以共享的区域，因此，可以直接通过存/取指令访问远程 Co-Array 数据。例如，$a=x(1)[2]$，访问映像 2 的 Co-Array 数据 y。转换为：a=caf_$x(2)%x(1)

用赋值语句而不是运行库调用实现 Co-Array 通信，可以为本地串行编译器优化提供更多的机会，提高程序性能。

3.4 内部函数的实现

Co-Array Fortran 定义了一组内部函数，查询 Co-Array 数据分布信息，同步指定映像。CAF 编译器通过 Fortran90 语言中的 Interface 机制实现内部函数的参数多态性，通过预编译器转换内部函数，插入新的参数或替换内部函数的参数。

为了实现查询内部函数，每个 Co-Array 数据除了对应一个 3.2 节的描述符外，还包含一个结构描述 Co-Array 数据的 co-dimension。例如，声明：

```
real,dimension(n)::a[0:2,*]
```

其 co_dimension 结构是：

```
type co$_dim                    type co$_array
integer lower                   integer num_codim
integer upper                   type (co$_dim), dimension(MAX_DIMENSION)::co_dims
end type co$_dim                end type co$_array
                                type (co$_array) codim_a
```

CAF 编译器在程序的一开始，根据 Co-Array 数据的声明，初始化每个数据的 co-dimension 结构。

```
codim_a%num_codim=2
codim_a%co_dims(1)%lower=0
codim_a%co_dims(1)%upper=2
codim_a%co_dims(1)%lower=1
```

对于查询函数 this_image(a)，转换器插入描述 a 的 co-dimesion 的变量 *codim_a* 替换原有参数 a，并增加映像的数目 N 作为第 2 参数。

this_image(a)→this_image (*codim_a*,N)

4 Co-Array Fortran 编译器实现和评测

4.1 编译实现

目前，CAF 并行编译器完整地实现了美国 Fortran 标准委员会(J3)2006 年制定的 Co-Array Fortran 规范草案(J3/05-183)，由预编译器(见图 3 虚框内部分)与运行支持库组成，其结构如图 3 所示。

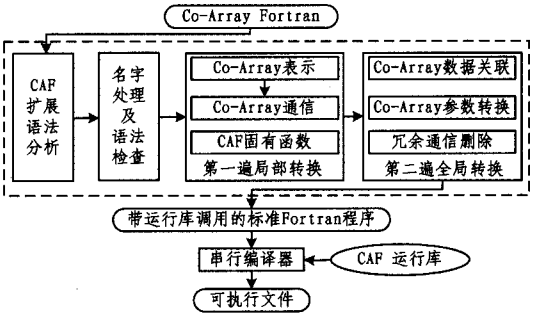

图 3 CAF 编译器结构

预编译器扫描和分析 Co-Array Fortran 程序，处理和转换 Co-Array Fortran 语法，实现 Co-Array 变量之间的通信。预编译器由语法分析器、语法检查器、转换器组成。在编译过程中，转换器执行两遍，第一遍局限于函数内，处理 Co-Array 数据表示与通信，转换内部函数。第二遍转换处理函数之间的 Co-Array，包括 Co-Array 数据关联，Co-Array 参数转换、删除无用的 Co-Array 数据通信。

运行库实现映像管理、Co-Array 变量的分配以及 Co-Array 固有函数。

4.2 实验对比与分析

CAF 是基于 Fortran90/95 语言扩展的并行程序设计语言，适合于科学计算领域。为了对其进行性能测试，选取了 4 个典型的科学计算程序进行测试，如表 1 所列。

表 1 典型科学计算程序

程序	功能	程序	功能
yg	二维拉格朗日和欧拉结合法	Md	分子动力学方程
La	三维电子运动模拟	Mf	二维欧拉方程

测试平台采用了国防科大自行研制的并行系统，该系统共有 128 个结点，每个节点包含一个 Itanium2 处理器(2 核，1.6GHz)，4GB 内存，操作系统为 Linux，串行编译器为 Intel ifort 11.1。节点之间通过 Infiniband 互联。MPI 采用了支持 Infiniband 的 mvapich2-1.2p1。

表 2 和表 3 分别给出了数据规模一时 CAF 和 MPI 的程序性能。表 4 和表 5 分别给出了数据规模二时 CAF 和 MPI 的程序性能。

表 2 CAF 程序性能(规模一：秒)

映像数	yg	la	md	mf
1	819.2	1496.2	719.6	520.92
2	349.1	758.5	365.9	192.38
4	185.0	387.2	192.0	90.42
8	97.2	192.3	98.8	49.06
16	54.9	92.7	46.5	26.26

表 3 MPI 程序性能(规模一：秒)

任务数	yg	La	md	mf
1	718.2	1250.5	874.831	348.77
2	353.5	773.2	443.890	156.25
4	183.8	385.6	232.226	88.20
8	91.3	184.9	115.4	46.04
16	50.5	86.6	57.5	29.35

(下转第 304 页)

虚拟化优化方法以很小的性能开销实现了I/O资源高效共享和安全隔离，并在我们设计的众核处理器芯片的应用中，取得了很好的效果。

参考文献

[1] Tang Hong, Gulbeden, Zhou Jing-yu, et al. The Panasas ActiveScale Storage Cluster-Delivering Scalable High Bandwidth Storage[C]//Proc of Supercomputing'04. 2004:53-53

[2] Adams K, Agesen O. A comparison of software and hardware techniques for x86 virtualization PPT[EB/OL]. http://www.cs.wisc.edu/areas/os/schedules/archive/vmware_osseminar.ppt

[3] Pratt I, Fraser K, et al. Xen 3.0 and the Art of Virtualization [C]//Proceedings of the Ottawa Linux Symposium. 2005

[4] Intel Corp. Intel Virtualization Technology Specification for the Intel Itanium Architecture[EB/OL]. www.intel.com/technology/vt/

[5] Intel Corp. Intel Virtualization Technology for Directed I/O Architecture Specfication[S]. 2006

[6] AMD Inc. Live Migration with AMD-V Extended Migration Technology[S]. 2007

[7] AMD Inc. Putting Server Virtulization to Work [R]. AMD White Paper, No32915B

[8] Ben-Yehuda M, Mason J, et al. Utilizing IOMMUs for Virtualization in Linux and Xen[C]//Proceedings of the Ottawa Linux Symposium. 2006

[9] Xen IOMMU trees[EB/OL]. http://xenbits.xensource.com/ext/xen-iommu.hg, 2007

[10] Sun Microsystems, Inc. The T1 Hypervisor and the sun4v Architecture/API[S]. 2008:123-128

[11] Chisnall D. The Definitive Guide to the Xen Hypervisor[M]. Prentice Hall, 2007

[12] Russel R. lguest: Implementing the little Linux hypervisor[C]//Proceedings of the 2007 Ottawa Linux Symposium. 2007

[13] PCI-SIG. PCI Express Base Specification Revision2.0[S]. 2009

[14] IOzone Filesystem Benchmark[EB/OL]. http://www.iozone.org

[15] Blum R. Network Performance Open Source Toolkit[M]. Wiley Publishing, Inc

(上接第289页)

表4 CAF程序性能(规模二:秒)

映像数	yg	la	md	mf
16	987.3	1731.2	763.22	954.61
32	418.98	864.4	393.21	388.01
64	209.12	443.1	203.01	128.05
128	149.63	218.9	109.84	67.98
256	75.31	104	55.59	42.06

表5 MPI程序性能(规模二:秒)

任务数	yg	la	md	mf
16	858.7	1510.8	919.3	892.07
32	426.5	747.9	467.8	382.51
64	205.2	385.4	237	104.3
128	99.2	185.5	119.4	53.93
256	54.3	89.3	58.6	31.86

对于规模一，CAF和MPI程序性能相当。尤其是md程序，CAF性能明显好于MPI。这是因为采用数据垫塞，使md程序的Cache利用率得到提高。对于yg和mf，当映像为1时，性能低于MPI。

对于规模二，当映像/进程数在16、32和64时，CAF和MPI的程序性能基本相当，当任务数达到128时，对于yg程序，CAF程序性能明显低于MPI。其原始是因为CAF规范中目前没有聚合操作，例如广播、规约等。广播操作通过所有映像读取同一个Co-Array数据实现，当规模扩大时，造成存储访问竞争，严重影响性能。归约操作通过临界区实现，规模扩大时，性能远低于采用树方式实现的MPI归约。yg程序中，一次迭代执行4次广播、6次归约，因此，当存在256个映像时，CAF性能低于MPI。CAF目前正在讨论的规范，已经将聚合操作加入[8]。

结束语 CAF编程简单，可扩展性好，极大地简化了SPMD模式的并行程序设计，是PGAS语言的典型代表，在科学计算领域已经逐步被人们接受。尤其是CAF已成为Fortran语言的一部分，极大地促进了CAF的发展和应用。

基于软件虚拟共享存储，在分布式系统上实现了一个CAF编译器。典型科学计算程序测试表明，其能够获得和MPI相当的性能。在下一步的工作中，将实现包含聚合操作的CAF的新标准，通过数据流分析，删除冗余的Co-Array的通信，提高CAF程序的性能。

参考文献

[1] Numrich R W, Reid J. Co-array Fortran for Parallel Programming[J]. ACM Fortran Forum, 1998, 17(2):1-31

[2] Fortran J3 Committee. Fortran 2008 Working Draft, J3/09-007r1[EB/OL]. Oct. 8, 2011. ttp://www.j3-fortran.org/doc/standing/links/007.pdf

[3] Cray Fortran Reference Manual. Volume 3[EB/OL]. http://docs.cray.com/books/S-3694-54/S-3694-54.pdf

[4] Adhianto L, Jin G, Krentel M, et al. Coarray Fortran 2.0 Pre-alpha Release Notes[EB/OL]. Oct. 8, 2011. http://caf.rice.edu

[5] Subset Co-Array Fortran translator[EB/OL]. Oct. 8, 2011. http://neptune.ce.ncsu.edu/~sarat/sc07/PGAS-SC2007/CAF/caf2omp.htm

[6] Coarrays via GNU Fortran's Coarray Communication Library [EB/OL]. http://gcc.gnu.org/wiki/CoarrayLib, 2011-10-08

[7] Distributed Memory Coarray Fortran with the Intel Fortran Compiler for Linux: Essential Guide[EB/OL]. Oct. 8, 2011. http://software.intel.com/en-us/articles/distributed-memory-coarray-fortran-with-the-intel-fortran-compiler-for-linux-essential-guide/

[8] Sottile M J, Rasmussen C E, Graham R L. Co-Array Collectives: Refined Semantics for Co-Array Fortran [C] // International Conference on Computational Science (2). 2006:945-952