2014年6月13日　統計数理研究所　オープンハウス

Rの並列計算用パッケージRhpcの改良

中野 純司　　モデリング研究系　教授

はじめに

今日ではCPU単体の性能の向上はほぼ望めず、マルチコアによる並列処理をいかに効率よく行うかが大規模データ処理において重要になってきている。そのため統計解析システムRでも既にいくつかの並列計算のためのパッケージが存在するが、それらの多くはスーパーコンピュータにおける良い並列環境とは言えない。例えば、よく利用されているパッケージsnowは多くがRで書かれており、大規模な環境での利用には遅延が大きくなる。またmulticoreはシングルノードのマルチコア環境で最良の速度を得ることができるが、分散メモリ型のクラスタシステムでは利用できない。

そこでわれわれはRのパッケージRhpcとしてlapplyを並列計算環境MPIを効率的に利用して実装したRhpc_worker_call(計算ノードに対する処理)、Rhpc_lapply(分散処理単位の処理を行わせる)を提供し、また、2GB以上のデータの処理に対応した。今回、当初の実装を改良し、さらに性能を上げることができ、また、プログラミング環境の整備を行った。

serialize機能の高速化

Rのbaseパッケージには、R内部で扱うオブジェクトをファイルで保存したりネットワークで送受信することが出来るように変換するserialize機能が用意されており、システムの中核機能の一つとなっている。われわれは当初Cの外部プログラムからRのインタプリタを経由してserializeを行っていたが、MPIを利用するときに呼び出し回数の多いこの操作は速度的に不利である。なぜならRの公開されている内部関数にはR_Serialize、R_Unserializeのシンボルがあるが具体的な呼び出し方法は面倒であったので、そのように実装していた。

今回、serializeのソースコードを元にして、Rのインタプリタを経由せずにその機能を呼び出すことにして、おおよそ5倍程度のserialize処理の高速化に成功した。

また、このアイデアは、Rcppパッケージなどの作者であるDirk Eddelbuettel氏の興味を引き、RApiSerializeパッケージとしてより一般化され、CRANに登録された。この処理手法は、データーベースや通信処理など、外部でRオブジェクトを扱うプロジェクトにおいて、多くの遅延を削減するのに利用できる。

MPIを利用した外部プログラムを容易にRhpcで利用するための手段

一般にMPIを用いたプログラム(C及びFortran等)ではMaster(rank0)とWorker(rank1以上)をコミュニケータを用いて通信させ、SPMDスタイルのプログラミングを行う。したがってRhpcのようにMPIを利用しているプログラムで利用するためには、MPIのコミュニケータの受け渡しが必要になる。そこで、Rhpcと既存のMPI外部プログラムが比較的容易に共存可能な環境を提供した。これにより、Rhpc_lapplyを使いつつMPIの外部プログラムを呼び出す事も可能となる。

RからMPI外部プログラムを呼び出す方法

Rhpcではmpi関数の初期化等によって広域変数_options_(options関数を参照) に以下の変数を設定する。

- Rhpc.mpi.f.comm　Fortranのためのコミュニケータ(R型:整数型)
- Rhpc.mpi.c.comm　C等のためのコミュニケータ(R型:外部ポインタ型)
- Rhpc.mpi.procs　MPIのコミュニケーションサイズ
- Rhpc.mpi.rank　MPI上のランク

右のRプログラムはFortran等のサブルーチンを呼び出す例である。Rhpc_lapply 等のRhpcの関数群はMaster上ではWorkerが行うような演算は行わないが、Rhpc_worker_noback(呼び出すだけで何もしない)により通常のSPMDのプログラムを呼び出すことによりMaster上でもWorker関数を呼び出し、実行することができる。これによって、#if等のプリプロセッサのマクロ等によって比較的少ない労力でRと移植性のある外部プログラムの開発が可能になる。

```
mpipif<-function(n)
{
  ## Exported functions get values by getOption()
  ## when they run on workers
  out<-.Fortran("mpipif",
                comm=getOption("Rhpc.mpi.f.comm"),
                n=as.integer(n),
                outpi=as.double(0))
  out$outpi
}

library(Rhpc)
Rhpc_initialize()
cl<-Rhpc_getHandle(4)

n<-10

## Load shared library
Rhpc_worker_noback(cl,dyn.load,"pi.so")
dyn.load("pi.so")

## Rhpc_worker_noback calls a function, but does not
## get any result.
## Workers should be started faster than a master.
Rhpc_worker_noback(cl,mpipif,n); mpipif(n)

Rhpc_finalize()
```

MPI外部サブルーチンの変更方法

呼び出すMPI Fortranプログラムには以下のような若干の変更が必要である。

- プログラムのサブルーチン化
- コミュニケータの変更
- MPI_InitおよびMPI_Finailizeの除去

以下にそのような変更をした例を示す。

```
--- pif.f       2014-05-19 17:23:36.000000000 +0900
+++ Rhpc_pif.f  2014-05-19 17:24:13.000000000 +0900
@@ -1,42 +1,33 @@
-      program main
+      subroutine mpipif(mpi_comm,n,outpi)
       include "mpif.h"
       double precision  mypi, sumpi, h, sum, x, f, a
       double precision  pi
       parameter (pi=3.14159265358979323846)
       integer n, rank, procs, i, ierr
-      character*16 argv
-      integer argc
+      integer mpi_comm
       f(a) = 4.d0 / (1.d0 + a*a)

-      argc = COMMAND_ARGUMENT_COUNT()
-      n=0
-      if (argc .ge. 1) then
-       call getarg(1, argv)
-        read(argv,*) n
-      endif
-
-      call MPI_INIT(ierr)
-      call MPI_COMM_RANK(MPI_COMM_WORLD, rank, ierr)
-      call MPI_COMM_SIZE(MPI_COMM_WORLD, procs, ierr)
+      call MPI_COMM_RANK(mpi_comm, rank, ierr)
+      call MPI_COMM_SIZE(mpi_comm, procs, ierr)
```

```
       call MPI_BCAST(n,1,MPI_INTEGER,0,
-     &               MPI_COMM_WORLD,ierr)
+     &               mpi_comm,ierr)
       if ( n .le. 0 ) goto 30
       h = 1.0d0/n
       sum  = 0.0d0
       do 20 i = rank+1, n, procs
          x = h * (dble(i) - 0.5d0)
          sum = sum + f(x)
  20   continue
       mypi = h * sum
       call MPI_REDUCE(mypi,sumpi,1,
   MPI_DOUBLE_PRECISION,
     &                 MPI_SUM,0,
-     &                 MPI_COMM_WORLD,ierr)
+     &                 mpi_comm,ierr)

       if (rank .eq. 0) then
          print *, 'pi = ', sumpi, ' diff = ', abs(
   sumpi - pi)
       endif
- 30   call MPI_FINALIZE(ierr)
-      stop
+ 30   continue
+      return
       end
```

おわりに

最新のRhpcパッケージではこの他にタスクのスケジューリング等などのチューニングを行っている。ただし、速度的にはserializeとMPIの通信回数の削減をどこまで最適化可能かが重要であるので、この点のさらなる改良を行っているところである。なお、本研究は中間栄治氏（株式会社COM-ONE）との共同研究である。