GotoBLASと統計解析環境R

中野　純司　モデリング研究系　教授

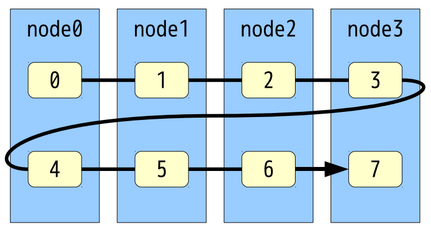
【はじめに】

BLAS(Basic Linear Algebra Subprograms)は線形計算のための基本的なライブラリAPIであり、統計解析環境Rなど、多くのソフトウェアで利用されている。 GotoBLASは後藤和茂氏によってTexas Advanced Computing Center (TACC) において開発された最適化・並列化された高性能なBLASの実装である。現在、TACCよりBSDライセンスで配布されているが、メンテナンスは終了しており、最近の計算機環境で容易に利用可能とは言い難い。われわれは動作確認が可能な環境、特にHigh performance computing(HPC)環境でGotoBLASのメンテナンスとRでの効率的な利用を試みた。本研究は中間栄治氏（株式会社COM-ONE）との共同研究である。

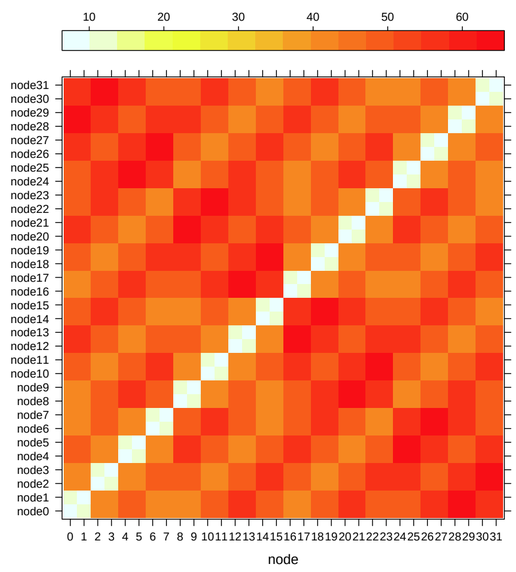
【SGI UVのGotoBLAS】

SGI UVはSilicon Graphics International (SGI) Corp.によって開発されたIntel Xeonを用いた共有メモリ型HPCアーキテクチャである。NUMA(Non Uniform Memory Access)アーキテクチャを特徴とする。GotoBLASは最大64コア、16NUMAノードを仮定しているが、これはNUMAノードが32、192コアであるようなSGI UVには適合しない。そこで簡単なパラメータの変更により拡張を試みたが、それでは2スレッド以上は動かないことが判明した。そのため、適切な改良を試みた。

GotoBLASではCPUキャッシュを有効に利用するためにノードごとに均等にスレッドをコアに割り当てる。



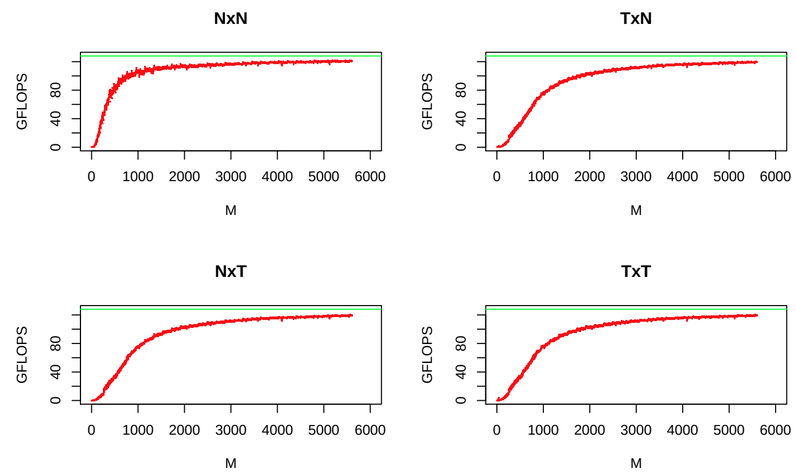
SGI UVにおいて2コア間のNUMA distanceを調べたところ以下の図のようになった。通常同じCPU内のコア間では10〜15であるが、SGI UVではQPI(Quick Path Interconnect)で結合されている2つのノード以外は40以上の大きな値になっている。



さらに、NUMA distanceとメモリーバンド幅の関係は以下のようであった。NUMA distanceが40以上ではバンド幅が著しく不足している。

read.write	cpunode	memnode	distance	bandwidth(GB/s)
read/write	31	31	10	4.61
read/write	31	30	13	3.47
read/write	31	29	40	1.37
read/write	31	27	48	1.20
read/write	31	19	55	1.05
read/write	31	3	62	0.93

これらの事実より、われわれはNUMA distanceが40より小さいノードだけを選択して利用するための関数をGotoBLASに付加した。そして各ノードが6コアのCPUを1個持つような2ノードのSGI UV上で、改良GotoBLASの`dgemm`を実行してみた。理論上は127.968GFLOPS の性能となるが、次の図のようにそれに近い値が出ている。



【64bit WindowsのGotoBLASS】

64bit Windows上でMinGW4.4 (またはそれ以降)を用いてGotoBLASを利用できるようにした。そのために、複素数関係の関数を改良し、スレッド数を制御できるようにWindows用にAPI (`goto_set_num_threads`, `goto_get_num_procs`) を追加した。また、DLLが外部ルーチン `xerbla` を使うようにした。Rでは`xerbla` はR.dllの中にあり、BLASのDLLはRblas.dllという名前でなければならない。

```
make BINARY=64 CC=x86_64-w64-mingw32-gcc \\
    FC=x86_64-w64-mingw32-gfortran NUM_THREADS=24 \\
    NO_CBLAS=1 NO_LAPACK=1 REFBLAS_ANTILOGY=1 \\
    DYNAMIC_ARCH=1 \\
    EXTERNAL_XERBLA_LIBNAME=R.dll LIBDLLNAME=Rblas.dll
```

【PowerおよびSPARCアーキテクチャのGotoBLAS】

LinuxにおいてはAffinityフラグをセットすることで物理コアを占有的に利用でき、この機能は最適化されたBLASでは重要である。ただ、Powerアーキテクチャではこの機能はまだ若干不安定のようである。`symv`, `zdot`, `zgemm`, `zsymv` のバグを修正することによりPowerでもGotoBLASが利用できた。

また、Solaris10 (SPARC64vii)でもGotoBLASが利用できるように軽微な改善を行った。

【GotoBLASとReference BLASの違い】

GotoBLAS は `csymv`, `csyr`, `cspmv`, `cspr`, `zsymv`, `zsyr`, `zspmv`, `zspr`などを内部に実装している。また、Reference BLASより多くの複素数機能を実装する。例えば`cherk`,`cher2k`,`zherk`,`zher2k`は`TRANS='T|C'`をサポートする。

【RにおけるGotoBLASスレッド数の制御】

Rでもしばしば OpenMPが利用される。その場合、同時に使われたGotoBLASがAffinityをセットすることにより、OpenMPのスレッドを妨害することがある。それを防ぐためにGotoBLASのスレッド数を制御するためのRパッケージを作成した。

```
$ export GOTO_NUM_THREADS=1
$ R -q
> library(blasctl)
> A<-matrix(runif(3e3^2),3e3,3e3)
> blas_get_num_procs()
[1] 1
> system.time(A%*%A)
   user  system elapsed
  3.992   0.012   4.003
> blas_set_num_threads(4)
> system.time(A%*%A)
   user  system elapsed
  4.332   0.048   1.234
```

【おわりに】

統計解析システムRは広く利用されているが、他のシステムに較べてその実行速度は速いとはいえない。したがって特に大きな線形演算を行うとき、その計算速度を速くすることは重要である。RでGotoBLASを利用することは、そのためのひとつの使いやすい方法といえる。