Installation de HPL Linpack

HPL est un benchmark destiné à estimer les capacités d'un ou plusieurs ordinateurs à résoudre un système de n équations linéaires à n inconnues, le résultat obtenu est exprimé en Gflops.

Nous présentons deux méthodes pour l'installation de ce dernier.

Approche 1 = Installation de hpl-2.0

Approche 2 = Installation de hpl-2.1

Aproche 1

Prérequis

Installer les paquets nécessaires :

```
# apt-get install gcc gfortran make openmpi-bin libopenmpi-dev
```

Un client ssh ou rsh au choix :

```
# apt-get install openssh-client
# apt-get install rsh-client
```

Télécharger et décompresser les sources :

```
$ cd ~ && wget http://www.netlib.org/benchmark/hpl/hpl-2.0.tar.gz
$ tar -xvzf hpl-2.0.tar.gz
```

BLAS (Basic Linear Algebra Subprograms) :

HPL a besoin d'une bibliothèque de fonctions BLAS. Il est possible d'installer les paquets Debian en fournissant une mais elle ne permettra pas d'exploiter au maximum les capacités de votre processeur. Pour cela elle doit être compilée de manière optimisée pour votre architecture. Plusieurs sont disponibles, une des plus performante est GotoBLAS. Celle-ci est désormais disponible en Open Source. Vous pouvez la télécharger ici (https://www.tacc.utexas.edu/tacc-projects/gotoblas2/downloads/)

```
$ tar -xvzf GotoBLAS2-1.13_bsd.tar.gz && cd GotoBLAS2

// OS 32 bit :
$ ./quickbuild.32bit

// OS 64 bit :
$ ./quickbuild.64bit
```

Copier le lien vers la bibliothèque partagée dans le répertoire approprié (root) :

```
# cp libgoto2.so /usr/local/lib
```

Compilation

Le fichier Make :

Ceux fournis dans le répertoire *hpl-2.0/setup* ne sont pas directement utilisables. Vous pouvez éditer le votre en utilisant comme base le fichier *hpl/setup/Make.UNKNOW.in* ou bien utiliser un des suivants :

- Make.Debian_GotoBlaS (monothread) pour processeur(s) "mono coeur" (ex : Pentium IV, AMD Athlon XP...).

- Make.Debian_GotoBlaS_p (multithread) pour processeur(s) "multi coeur" (ex : Intel Core 2 Duo/Quad, AMD Athlon II x2/x4...).

Placer le fichier Make choisi dans le répertoire *hpl-2.0*. Admettons pour la suite que nous utilisons le fichier *Make.Debian_GotoBLAS_p*.

```
$ cd ~/hpl-2.0
$ make arch=Debian_GotoBLAS_p
```

Si la compilation s'est bien déroulée deux fichiers ont été créé dans le répertoire bin/Debian_GotoBLAS_p. Pour vérifier :

```
$ ls bin/Debian_GotoBLAS_p
HPL.dat xhpl
```

Lorsqu'une compilation échoue il est nécessaire de nettoyer avant de recommencer :

```
$ make clean_arch_all arch=[arch]
```

Exécution

Le fichier HPL.dat :

Il sert à configurer les tests, la documentation officielle est disponible ici. Cette page permet d'en générer un automatiquement en fournissant quelques renseignements.

Pour démarrer rapidement, remplacer celui généré par défaut dans le répertoire *bin/Debian_GotoBLAS_p* par celui-ci HPL.dat.

Pour parvenir aux meilleures performances il faudra adapter la valeur de Ns (ligne 6) en fonction de la mémoire vive disponible sur votre système. Voici quelques exemples :

- 1024 Mo : ≈ 10000 Ns
- 2048 Mo : ≈ 14000 Ns
- 4096 Mo : ≈ 20000 Ns
- 8192 Mo : ≈ 29000 Ns

La formule pour calculer approximativement la valeur maximum de Ns est la suivante :

```
√ ( 0.8 x ram_en_octets / 8 )
```

Exemple pour 1024 Mo :

```
1024 Mo = 1073741824 octets (1024 x 1024 x 1024)
√ ( 0.8 x 1073741824 / 8 ) ≈ 10362
```

Lancer les tests :

```
$ cd bin/Debian_GotoBLAS_p
$ mpirun -np 1 xhpl
```

Lecture des résultats :

Ils sont exprimés sous la forme d'une puissance de 10, exemples :

$1.000e\text{-}01 = 1 \times 10^{-1} = 0.1$
$1.000e+00 = 1 \times 10^{0} = 1$
$1.000e+01 = 1 \times 10^{1} = 10$
$1.000e+02 = 1 \times 10^{2} = 100$
$1.000e+03 = 1 \times 10^{3} = 1000$

Approche 2

Télécharge hpl-2.1

```
wget http://www.netlib.org/benchmark/hpl/hpl-2.1.tar.gz
```

Décompréser

```
tar xvf hpl-2.1.tar.gz
cd hpl-2.1
```

Copier, dans le repertoire setup, le fichier de configuration qui convient à votre système. Dans notre cas nous utiliserons Make.Linux_PII_CBAS.

```
cp  setup/Make.Linux_PII_CBLAS  .
```

Installer le paquet mpich2

```
apt-get install mpich2
```

Installer le paquet libatlas

```
apt-get install libatlas-base-dev
```

Modifier le fichier de configuration. Make.Linux_PII_CLAS remplacer g77 par mpif77

S'assurer que les variables Ladir et Mpdir pointe vers le bon repertoire des librairies.
Et pour finir compiler votre code source.

```
Make arch=Linux_PII_CBLAS
```

Execution:

Exécuter avec la commande:

```
mpirun -np 1 ./xhpl
```

Voici un exemple d'output après exécution du bench.

```
================================================================================
HPLinpack 2.1  --  High-Performance Linpack benchmark  --   October 26, 2012
Written by A. Petitet and R. Clint Whaley,  Innovative Computing Laboratory, UTK
Modified by Piotr Luszczek, Innovative Computing Laboratory, UTK
Modified by Julien Langou, University of Colorado Denver
================================================================================

An explanation of the input/output parameters follows:
T/V    : Wall time / encoded variant.
N      : The order of the coefficient matrix A.
NB     : The partitioning blocking factor.
P      : The number of process rows.
Q      : The number of process columns.
Time   : Time in seconds to solve the linear system.
Gflops : Rate of execution for solving the linear system.

The following parameter values will be used:

N      :    5000
NB     :     128
PMAP   : Row-major process mapping
P      :       1
Q      :       1
PFACT  :    Left    Crout    Right
NBMIN  :       4
NDIV   :       2
RFACT  :    Left    Crout    Right
BCAST  :  1ringM
DEPTH  :       1
SWAP   : Mix (threshold = 64)
L1     : transposed form
U      : transposed form
EQUIL  : yes
ALIGN  : 8 double precision words

================================================================================
T/V                N    NB     P     Q               Time                 Gflops
--------------------------------------------------------------------------------
WR11L2L4        5000   128     1     1               3.58              2.328e+01
HPL_pdgesv() start time Fri Aug  1 17:06:19 2014

HPL_pdgesv() end time   Fri Aug  1 17:06:22 2014

WR11L2C4        5000   128     1     1               3.56              2.342e+01
HPL_pdgesv() start time Fri Aug  1 17:06:23 2014

HPL_pdgesv() end time   Fri Aug  1 17:06:27 2014

WR11L2R4        5000   128     1     1               3.48              2.396e+01
HPL_pdgesv() start time Fri Aug  1 17:06:28 2014

HPL_pdgesv() end time   Fri Aug  1 17:06:32 2014
```

```
WR11C2L4        5000   128     1     1               3.44             2.425e+01
HPL_pdgesv() start time Fri Aug  1 17:06:33 2014

HPL_pdgesv() end time   Fri Aug  1 17:06:37 2014

WR11C2C4        5000   128     1     1               3.30             2.530e+01
HPL_pdgesv() start time Fri Aug  1 17:06:38 2014

HPL_pdgesv() end time   Fri Aug  1 17:06:41 2014

WR11C2R4        5000   128     1     1               3.28             2.545e+01
HPL_pdgesv() start time Fri Aug  1 17:06:43 2014

HPL_pdgesv() end time   Fri Aug  1 17:06:46 2014

WR11R2L4        5000   128     1     1               3.29             2.534e+01
HPL_pdgesv() start time Fri Aug  1 17:06:47 2014

HPL_pdgesv() end time   Fri Aug  1 17:06:51 2014

WR11R2C4        5000   128     1     1               3.29             2.534e+01
HPL_pdgesv() start time Fri Aug  1 17:06:52 2014

HPL_pdgesv() end time   Fri Aug  1 17:06:55 2014

WR11R2R4        5000   128     1     1               3.42             2.439e+01
HPL_pdgesv() start time Fri Aug  1 17:06:57 2014

HPL_pdgesv() end time   Fri Aug  1 17:07:00 2014

================================================================================

Finished      9 tests with the following results:
              9 tests completed without checking,
              0 tests skipped because of illegal input values.
--------------------------------------------------------------------------------

End of Tests.
================================================================================
```