CERN, LHCb
PH-LBC
Diego Berdeja Suárez
Supervisor: Daniel Cámpora

Summer Studentship Report

HPL Performance Tests

The first part of my project consisted on the customization of the High Performance Linpack benchmark tests for the LHCb online computer farm. The purpose of this was to measure the computing capability of the actual equipment.

The HPL is a benchmarking tool that measures a cluster's computational capability by tasking it with the resolution of a system of equations in matrix representation. It generates a random matrix with parameters provided by the user, and distributes it between the nodes in the cluster for computation. The benchmark assesses the computational capability by measuring the average FLOPS, and checks for correctness by comparing the residual norm against the computer's ε value. HPL uses a BLAS implementation, as well as a Message Parsing Interface.

After running some basic tests with the package, I created a guide that can be used for testing by further participants. I attach it at the end of the report, along with the results.

Retina Algorithm

My next task consisted on the design and implementation of a Retina Algorithm for the VErtex LOcator on the LHCb, considering a potential implementation of Graphics Processing Units into the computing system. I describe the general aspects of the algorithm in the following lines:

The Retina Algorithm attempts to emulate the human eye's ability to respond quickly to linear patterns. The general scheme consists on building a database of pre-constructed hit tracks and on assigning a weight to each track depending on the

proximity of actual input data hits. This weight is then considered in the parameter space of the database and searched for high-weight clusters. The advantage of this method is that it is very directly mappable onto a 3D grid of threads in a GPU, and can also be relatively easy to build in such a way that tiled memory access reduces latency and improves efficiency in thread synchronization.

Many considerations undertaken to prevent branching and reduce latency prolonged the design process, so the algorithm could only be planned, but not implemented. Thus, implementation will remain as a future task.

A graphic description of the algorithm is included in the next page, along with the HPL guide and the test results.

Installation

This guide assumes a secure and direct connection between server nodes (i.e., no need to generate ssh keys).

[edit]Required Packages

The following packages are necessary to build HPL from source:

- gcc
- gcc-c++
- atlas
- blas
- lapack
- mpich2
- make
- mpich2-devel
- atlas-devel

The packages must be installed onto all nodes participating in the benchmark. The following script can be used for such purpose, by modifying the pertinent lines:

```
#!/bin/bash for i in $(seq -w 1 20) do         ssh -t hlta10$i "sudo yum
install -y gcc gcc-c++ atlas blas lapack  mpich2 make mpich2-devel atlas-
devel" & done wait
```

[edit]Building from source

Download the .tar file from the Netlib website and unpack it. For example, in this case the following is used:

```
export http_proxy=http://<user>@netgw01:8080 export
ftp_proxy=http://<user>@netgw01:8080 wget
http://www.netlib.org/benchmark/hpl/hpl-2.1.tar.gz tar -zxvf hpl-2.1.tar.gz
```

The package includes a variety of makefiles, and the user is encouraged to select one appropriate for the architecture and edit it to fit. Here, the makefile has already been created, and is provided below:

```
#   #  -- High Performance Computing Linpack Benchmark (HPL)
#     HPL - 2.1 - October 26, 2012                          #     Antoine P.
Petitet                                                    #     University of
Tennessee, Knoxville                                     #     Innovative
Computing Laboratory                                      #     (C) Copyright
2000-2008 All Rights Reserved                            #
```

```
#  -- Copyright notice and Licensing terms:                                  #
#  Redistribution  and  use in  source and binary forms, with or without #
modification, are  permitted provided  that the following  conditions #  are
met:                                                                       #
#  1. Redistributions  of  source  code  must retain the above copyright #
notice, this list of conditions and the following disclaimer.          #
#  2. Redistributions in binary form must reproduce  the above copyright #
notice, this list of conditions,  and the following disclaimer in the #
documentation and/or other materials provided with the distribution.  #
#  3. All  advertising  materials  mentioning  features  or  use of this #
software must display the following acknowledgement:                   #  This
product  includes  software  developed  at  the  University  of #  Tennessee,
Knoxville, Innovative Computing Laboratory.              #
#  4. The name of the  University,  the name of the  Laboratory,  or the #
names  of  its  contributors  may  not  be used to endorse or promote #
products  derived   from   this  software  without  specific  written #
permission.                                                            #
#  -- Disclaimer:                                                          #
#  THIS  SOFTWARE  IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS #
``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES,  INCLUDING,  BUT NOT #
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR #  A
PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE UNIVERSITY #  OR
CONTRIBUTORS  BE  LIABLE FOR ANY  DIRECT,  INDIRECT,  INCIDENTAL, #  SPECIAL,
EXEMPLARY,  OR  CONSEQUENTIAL DAMAGES  (INCLUDING,  BUT NOT #  LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, #  DATA OR PROFITS;
OR BUSINESS INTERRUPTION)  HOWEVER CAUSED AND ON ANY #  THEORY OF LIABILITY,
WHETHER IN CONTRACT,  STRICT LIABILITY,  OR TORT #  (INCLUDING NEGLIGENCE OR
OTHERWISE) ARISING IN ANY WAY OUT OF THE USE #  OF THIS SOFTWARE, EVEN IF
ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.  #
########################################################################## #   #
------------------------------------------------------------------------ # -
shell ----------------------------------------------------------------- # ------
---------------------------------------------------------------- # SHELL
= /bin/sh # CD           = cd CP           = cp LN_S         = ln -s MKDIR
= mkdir RM           = /bin/rm -f TOUCH        = touch # # ------------------
-------------------------------------------------- # - Platform identifier
---------------------------------------------- # -------------------------
------------------------------------------- # ARCH         = SLC6 # # ------
---------------------------------------------------------------- # - HPL
Directory Structure / HPL library ------------------------------ # ----------
-------------------------------------------------------- # TOPdir       =
```

```
$(HOME)/hpl INCdir       = $(TOPdir)/include BINdir       =
$(TOPdir)/bin/$(ARCH) LIBdir       = $(TOPdir)/lib/$(ARCH) # HPLlib       =
$(LIBdir)/libhpl.a  # # ----------------------------------------------------
----------------- # - Message Passing library (MPI) -----------------------
------------ # ------------------------------------------------------------
--------- # MPinc tells the  C  compiler where to find the Message Passing
library # header files,  MPlib  is defined  to be the name of  the library to
be # used. The variable MPdir is only used for defining MPinc and MPlib. #
MPdir        = /usr/lib64/mpich2 MPinc        = -I$(MPdir)/include MPlib
= $(MPdir)/lib/libmpich.a # # -------------------------------------------
----------------------- # - Linear Algebra library (BLAS or VSIPL) ----------
------------------- # ------------------------------------------------------
--------------- # LAinc tells the  C  compiler where to find the Linear
Algebra  library # header files,  LAlib  is defined  to be the name of  the
library to be # used. The variable LAdir is only used for defining LAinc and
LAlib. # LAdir        = /usr/lib64/atlas LAinc        = LAlib        =
$(LAdir)/libcblas.a $(LAdir)/libatlas.a # # ------------------------------
----------------------------------- # - F77 / C interface -----------------
-------------------------------- # -------------------------------------
---------------------------- # You can skip this section  if and only if
you are not planning to use # a  BLAS  library featuring a Fortran 77
interface.  Otherwise,  it  is # necessary  to  fill out the  F2CDEFS
variable  with  the  appropriate # options.  **One and only one**  option
should be chosen in **each** of # the 3 following categories: # # 1) name
space (How C calls a Fortran 77 routine) # # -DAdd_              : all lower
case and a suffixed underscore  (Suns, #                        Intel, ...),
[default] # -DNoChange          : all lower case (IBM RS6000), # -
DUpCase            : all upper case (Cray), # -DAdd__             : the
FORTRAN compiler in use is f2c. # # 2) C and Fortran 77 integer mapping # # -
DF77_INTEGER=int   : Fortran 77 INTEGER is a C int,         [default] # -
DF77_INTEGER=long  : Fortran 77 INTEGER is a C long, # -DF77_INTEGER=short :
Fortran 77 INTEGER is a C short. # # 3) Fortran 77 string handling # # -
DStringSunStyle    : The string address is passed at the string loca- #
tion on the stack, and the string length is then #
passed as  an  F77_INTEGER  after  all  explicit #
stack arguments,                       [default] # -DStringStructPtr   : The
address  of  a  structure  is  passed  by  a #                       Fortran
77  string,  and the structure is of the #                       form: struct
{char *cp; F77_INTEGER len;}, # -DStringStructVal   : A structure is passed
by value for each  Fortran #                        77 string,  and  the
structure is  of the form: #                        struct {char *cp;
```

```
F77_INTEGER len;}, # -DStringCrayStyle   : Special option for  Cray
machines,  which  uses #                      Cray  fcd  (fortran  character
descriptor)  for #                       interoperation. # F2CDEFS      = # #
---------------------------------------------------------------------- # -
HPL includes / libraries / specifics ------------------------------ # ------
---------------------------------------------------------------- #
HPL_INCLUDES = -I$(INCdir) -I$(INCdir)/$(ARCH) $(LAinc) $(MPinc) HPL_LIBS
= $(HPLlib) $(LAlib) $(MPlib) # # - Compile time options --------------------
-------------------------- # # -DHPL_COPY_L           force the copy of the
panel L before bcast; # -DHPL_CALL_CBLAS       call the cblas interface; # -
DHPL_CALL_VSIPL       call the vsip  library; # -DHPL_DETAILED_TIMING  enable
detailed timers; # # By default HPL will: #    *) not copy L before
broadcast, #    *) call the BLAS Fortran 77 interface, #    *) not display
detailed timing information. # HPL_OPTS     = -DHPL_CALL_CBLAS # # ----------
------------------------------------------------------------ # HPL_DEFS     =
$(F2CDEFS) $(HPL_OPTS) $(HPL_INCLUDES) # # ---------------------------------
---------------------------------- # - Compilers / linkers - Optimization
flags --------------------------- # ---------------------------------------
---------------------------- # CC           = /usr/bin/mpicc CCNOOPT      =
$(HPL_DEFS) CCFLAGS      = $(HPL_DEFS) -fomit-frame-pointer -O3 -funroll-
loops # # On some platforms,  it is necessary  to use the Fortran linker to
find # the Fortran internals used in the BLAS library. # LINKER       =
/usr/bin/mpicc LINKFLAGS    = $(CCFLAGS) # ARCHIVER     = ar ARFLAGS      = r
RANLIB       = echo # # -----------------------------------------------------
-----------------
```

The makefile simply sets up the necessary environmental variables for the BLAS and MPI libraries. Copy it into the HPL parent directory with the name *Make.SLC6*, and run:

```
make arch=SLC6
```

[edit]Setting Up The Benchmark

[edit]Hosts

To verbalize the nodes that will be participating in the benchmark, move into (HPL parent directory)-> bin-> SLC6->. Once there, create a file titled *hosts*, and write a list of the nodes to be used. For example, in this run we use the hlta10** nodes:

```
hlta1001 hlta1002 hlta1003 hlta1004 hlta1005 hlta1006 hlta1007 hlta1008
hlta1009 hlta1010 hlta1011 hlta1012 hlta1013 hlta1014 hlta1015 hlta1016
```

```
hlta1017 hlta1018 hlta1019 hlta1020
```

[edit]**The HPL.dat file**

In the same directory, there exists a file called HPL.dat. This file is the one used to configure the benchmark run, and is pretty much self explainatory. In this section, the particular choices for this run will be made explicit and explained:

- Lines 1 & 2: **Titles**.
- Line 3: **Results.out**. Output file name.
- Line 4: **1**. This line is used to specify the output device. 1 directs it to the output file.
- Line 5: **1**. Number of problem sizes. Only one was used, because previous tests had helped determine the most appropriate problem size.
- Line 6: **131840**. Problem size. This determines the amount of memory that will be used during the benchmark. It is recommended to leave 10% free for the OS and kernel. A good rule of thumb is: Memory=sqrt((Memory in GB per node)(1024)(1024)/8)*.9.
- Line 7: **1** Number of block sizes (NBs).
- Line 8: **256**. Block size. For large clusters, it has been empirically concluded that best block sizes run form 32 to 256, and with 8+ nodes 256 is the best size. Adjust the problem sizes to be multiples of the block size.
- Line 9: **0**. This runs the process in a row-mapping method, which is convenient for clusters communicating through a node.
- Line 10: **1**. This is the number of process grids to be used. Each process grid is a virtual mapping of the matrix to be solved, and it is logically best for it to be as square as possible. The multiplication of the P's and Q's has to match the number of cores to be used.
- Line 11: **16**. P's. Should be smaller than Q, but as square as possible.
- Line 12: **40**. Q's.
- Line 13: **16**. Residual threshold. If the ratio between matrix residue and the ε value exceeds this value, the run will be considered a failure. This value should be of the order of 10. Set it to negative to consider every run a success.
- Lines 14-23: These lines specify the algorithms used for LU factorisation. Copy the standard Top500 values from the file below.
- Lines 24 & 25: **1;1**. These lines specify the number of look-ahead depths and the depths themselves. Basically, these parameters determine the level of communication between the nodes. A depth of one is enough, any larger depth would slash the performance.
- Lines 26 & 27: **1;60**. These lines specify the SWAP procedures to be followed in case of a memory overload. Leave as is.
- Lines 28 & 29: **0;0**. These lines specify whether the run should store matrix steps in a transposed form. 0 stands for yes. It is the humble opinion of the author that this option is not very useful, since storage in an untransposed way is not efficient for any kind of system or network topology.
- Line 30: **1**. Equilibration phase. Irrelevant. Leave as is.
- Line 31: **8**. Alignment memory. 8 bytes should be enough for large clusters. Use 4 for smaller runs.

The HPL.dat file used in this example is as follows:

```
HPLinpack benchmark input file LHCb Online farm benchmarking, CERN.
LHCb_Online_Results.out 1 1 131840 1 256 0 1 16 40 16.0 1 1 2 4 8 1 2 1 2 2 1
3 1 1 1 60 0 0 1 8
```

[edit]Running The Benchmark

Again in the (HPL parent directory)-> bin-> SLC6-> directory, run:

```
mpiexec.hydra -f hosts -n x ./xhpl
```

Replace 'x' with the number of cores in the cluster.

[edit]Results

HPL benchmark results are to be posted here following the format of the first one.

[edit]27th of June, 2014 (hlta10**)

```
================================================================================
=== 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      :  131840
NB     :     256  PMAP   : Row-major process mapping P      :      16
Q      :      40  PFACT  :   Crout  NBMIN  :       4        8  NDIV   :
2  RFACT  :   Right  BCAST  :  1ringM   2ringM  DEPTH  :       1  SWAP   :
Spread-roll (long) L1     : transposed form U      : transposed form EQUIL  :
yes ALIGN  : 8 double precision words  ------------------------------------
----------------------------------------  - The matrix A is randomly
generated for each test. - The following scaled residual check will be
computed:       ||Ax-b||_oo / ( eps * ( || x ||_oo * || A ||_oo + || b ||_oo
) * N ) - The relative machine precision (eps) is taken to be
1.110223e-16 - Computational tests pass if scaled residuals are less than
16.0
```

```
================================================================================
=== T/V                N    NB     P     Q               Time
Gflops ------------------------------------------------------------------------
---------- WR11R2C4      131840   256    16    40           31960.21
4.780e+01 HPL_pdgesv() start time Fri Jun 27 10:45:04 2014  HPL_pdgesv() end
time   Fri Jun 27 19:37:44 2014  ----------------------------------------------
---------------------------------- ||Ax-
b||_oo/(eps*(||A||_oo*||x||_oo+||b||_oo)*N)=        0.0018327 ...... PASSED
================================================================================
=== T/V                N    NB     P     Q               Time
Gflops ------------------------------------------------------------------------
---------- WR11R2C8      131840   256    16    40           37288.97
4.097e+01 HPL_pdgesv() start time Fri Jun 27 19:56:15 2014  HPL_pdgesv() end
time   Sat Jun 28 06:17:44 2014  ----------------------------------------------
---------------------------------- ||Ax-
b||_oo/(eps*(||A||_oo*||x||_oo+||b||_oo)*N)=        0.0018250 ...... PASSED
================================================================================
=== T/V                N    NB     P     Q               Time
Gflops ------------------------------------------------------------------------
---------- WR13R2C4      131840   256    16    40           30761.89
4.966e+01 HPL_pdgesv() start time Sat Jun 28 06:49:47 2014  HPL_pdgesv() end
time   Sat Jun 28 15:22:29 2014  ----------------------------------------------
---------------------------------- ||Ax-
b||_oo/(eps*(||A||_oo*||x||_oo+||b||_oo)*N)=        0.0016907 ...... PASSED
================================================================================
=== T/V                N    NB     P     Q               Time
Gflops ------------------------------------------------------------------------
---------- WR13R2C8      131840   256    16    40           37551.80
4.068e+01 HPL_pdgesv() start time Sat Jun 28 15:50:52 2014  HPL_pdgesv() end
time   Sun Jun 29 02:16:44 2014  ----------------------------------------------
---------------------------------- ||Ax-
b||_oo/(eps*(||A||_oo*||x||_oo+||b||_oo)*N)=        0.0016494 ...... PASSED
================================================================================
===  Finished      4 tests with the following results:              4 tests
completed and passed residual checks,              0 tests completed and
failed residual checks,              0 tests skipped because of illegal
input values. ----------------------------------------------------------------
-----------------  End of Tests.
================================================================================
===
```

Observations:

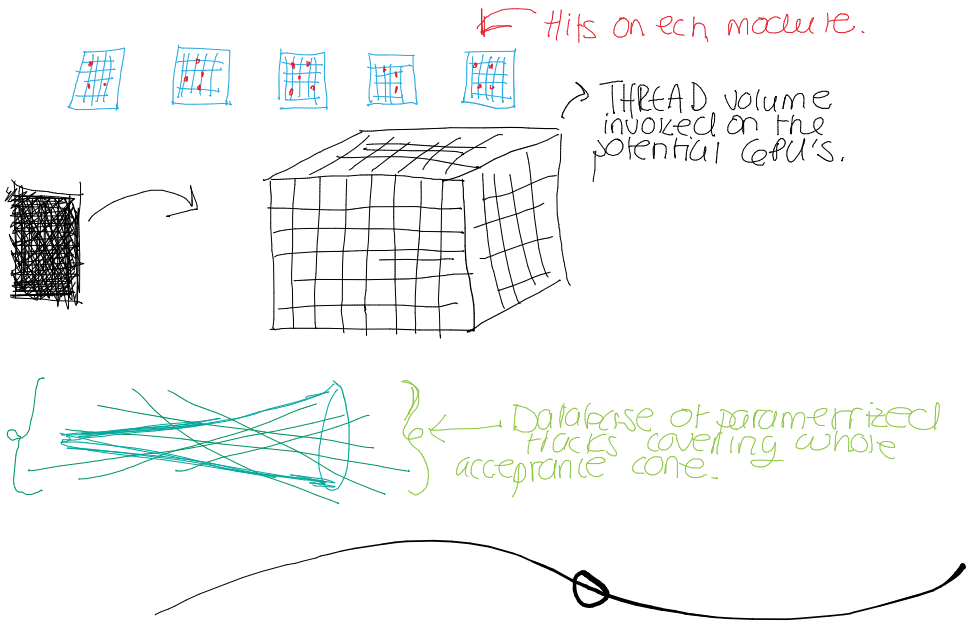
- Litecoin mining processes were running in the background. Although at lowest priority, they did occupy a significant amount of resources during the test.

[edit]**30th of June, 2014 (hlta10**, hlta08**, hlta09**)**

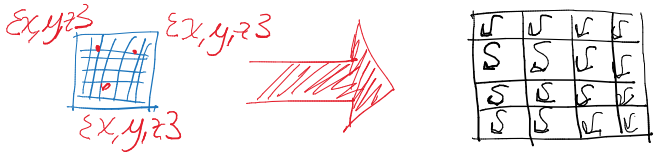
```
================================================================================
=== HPLinpack 2.1  --  High-Performance Linpack benchmark  --   October 26,
2012 Written by A. Petitet and R. Clint Whaley,  Innovative Computing
Laboratory, UTK Modified by Piotr Luszczek, Innovative Computing Laboratory,
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      :  228352
NB     :     256  PMAP   : Row-major process mapping P      :      16
Q      :      30  PFACT  :   Crout  NBMIN  :       4        8  NDIV   :
2  RFACT  :   Right  BCAST  :  1ringM   2ringM  DEPTH  :       1  SWAP   :
Spread-roll (long) L1     : transposed form U      : transposed form EQUIL  :
yes ALIGN  : 8 double precision words  ----------------------------------
----------------------------------------  - The matrix A is randomly
generated for each test. - The following scaled residual check will be
computed:       ||Ax-b||_oo / ( eps * ( || x ||_oo * || A ||_oo + || b ||_oo
) * N ) - The relative machine precision (eps) is taken to be
1.110223e-16 - Computational tests pass if scaled residuals are less than
16.0
================================================================================
=== T/V                N    NB     P     Q               Time
Gflops --------------------------------------------------------------------
---------- WR11R2C4      228352   256    16    30          167682.00
4.734e+01 HPL_pdgesv() start time Mon Jun 30 14:54:10 2014  HPL_pdgesv() end
time   Wed Jul  2 13:28:52 2014
```

Observations:

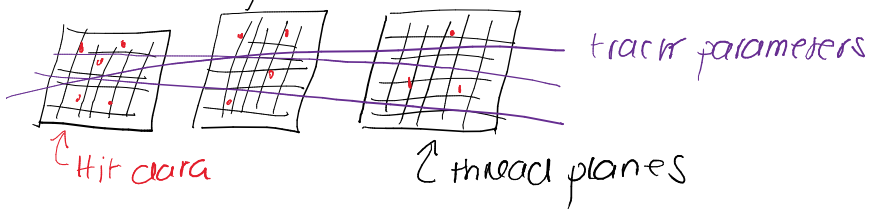
- Test run on three subfarms for the first time.
- Litecoin mining killed for this test.



1. Load hit data onto device memory, assigning a thread plane for each module



2. Load database onto device memory, pairing the parameters to the module thread planes



3. Each thread then performs two tasks:
   i. Find the distance between tracks and hits, and decide upon the nearest track
   ii. Weight the distance to that track:

$$w = \exp\left(\frac{\|(\vec{x} - \vec{y})\|}{\xi^2}\right)$$

($\vec{x}$: hit, $\vec{y}$: track, ξ: grid step (NEXO resolution))

4. Launch another kernel to add track weights for each module.

5. Keep tracks with highest weight, limited in number by the number of hits in module (lowest).