MHPC
Master in High Performance Computing

Lecture 8: Introduction to benchmarking and tools

Stefano Cozzini

CNR/IOM and eXact-lab srl

SISSA
Scuola Internazionale Superiore di Studi Avanzati

ICTP The Abdus Salam International Centre for Theoretical Physics

MHPC
Master in High Performance Computing

Agenda/ Aims

- Give you the feeling how much is important to know how your system/ application/computational experiment is performing..
- Name a few standard benchmarks that can help you in making/taking a decision
- Discuss in some details some of them:
  - HPL/ HPCG/ STREAM
- Show you some tricks and tips how to make your own benchmarking procedure

SISSA ICTP

MHPC
Master in High Performance Computing

benchmark: a definition

a benchmark is the act of running a computer program, a set of programs, or other operations, in order to assess the relative performance of an object, normally by running a number of standard tests and trials against it

from wikipedia

SISSA ICTP

MHPC
Master in High Performance Computing

A few statements

- no single number can reflect overall performance
- the only benchmark that matters is the intended workload.
- The purpose of benchmarking is not to get the best results, but to get consistent repeatable accurate results that are also the best results.

SISSA ICTP

MHPC
Master in High Performance Computing

Some other more

- Benchmarking is absolutely essential
- Should be done by people who know the application, the hardware, and the operating system
- You needs several representative benchmarks
- Be careful with artificial benchmarks or marketing myths

SISSA ICTP

MHPC
Master in High Performance Computing

An important note

- Measuring and reporting performance is the basis for scientific advancement in HPC.
- Not always scientific papers/reports guarantee reproducibility
- A lack of standards/rule is actually present in benchmarking arena.

SISSA ICTP

MHPC
Master in High Performance Computing

challenges in benchmarking:

- Benchmarking is not easy and often involves several iterative rounds in order to arrive at predictable, useful conclusions.
- Interpretation of benchmarking data is also extraordinarily difficult:
  - Vendors tend to tune their products specifically for industry-standard benchmarks. Use extreme caution in interpreting their results.
  - Many benchmarks focus entirely on the speed of computational performance, neglecting other important features of a computer system.
  - Benchmarks seldom measure real world performance of mixed workloads — running multiple applications concurrently in a full, multi-department environment

SISSA ICTP

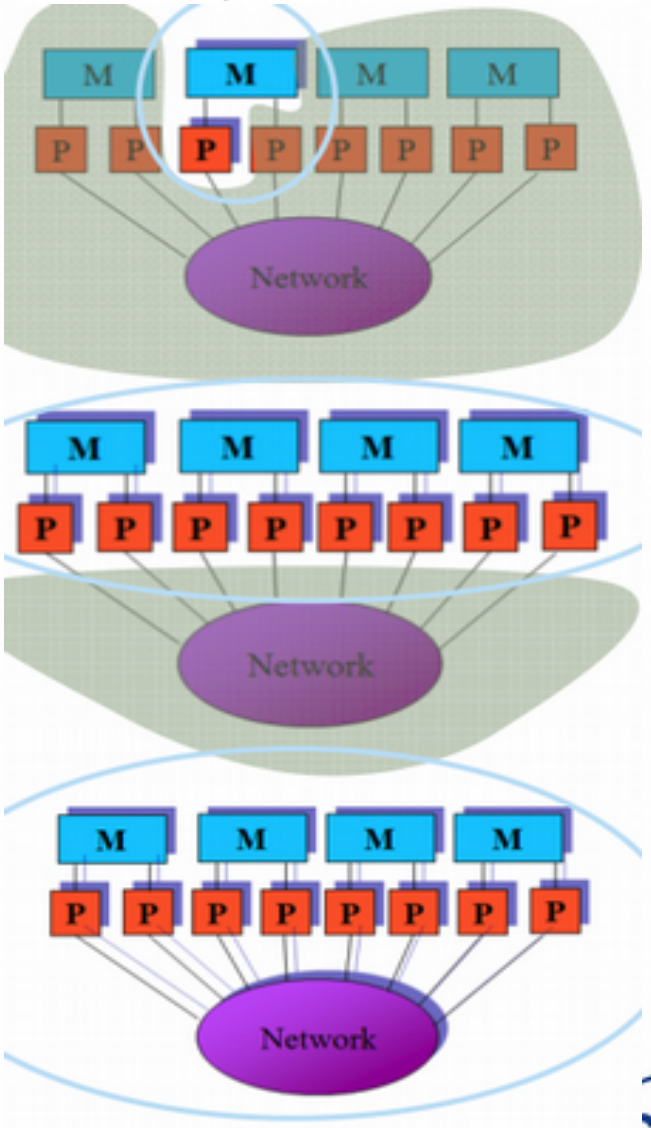
MHPC
Master in High Performance Computing

What we need to benchmark on a modern system

Local: only a single processor (core) is performing computations.

Embarrassingly Parallel each processor (core) in the entire system is performing computations but they do no communicate with each other explicitly.

Global -all processors in the system are performing computations and they explicitly communicate with each other.



SISSA
ICTP

MHPC
Master in High Performance Computing

Type of code for benchmark

- Synthetic codes
  - Basic hardware and system performance tests
  - Meant to determine expected future performance and serve as surrogate for workload not represented by application codes
  - useful for performance modeling
- Application codes
  - Actual application codes as determined by requirements and usage
  - Meant to indicate current performance
  - Each application code should have more than one real test case

SISSA ICTP

MHPC
Master in High Performance Computing

A good benchmark is

- Relevant and meaningful to the target application domain
- Applicable (portable) to a broad spectrum of hardware architecture
- Adopted both by users and by vendors to enable comparative evaluation

SISSA ICTP

MHPC
Master in High Performance Computing

Some freely available benchmark (1)

- General benchmark:
  - HPL Linpack (for Top500)
  - HPC Challenge Benchmark:
    - a collection of basic benchmark beyond HPL
  - NAS benchmark suite
    - math kernel implemented both in MPI and openMP
    - (http://www.nas.nasa.gov/publications/npb.html)
  - HPCG
    - A recently introduced benchmark to “fix” the HPL one
  - Stream
    - Memory benchmark
  - Graph500:
    - Data intensive application (used for Graph500)

SISSA ICTP

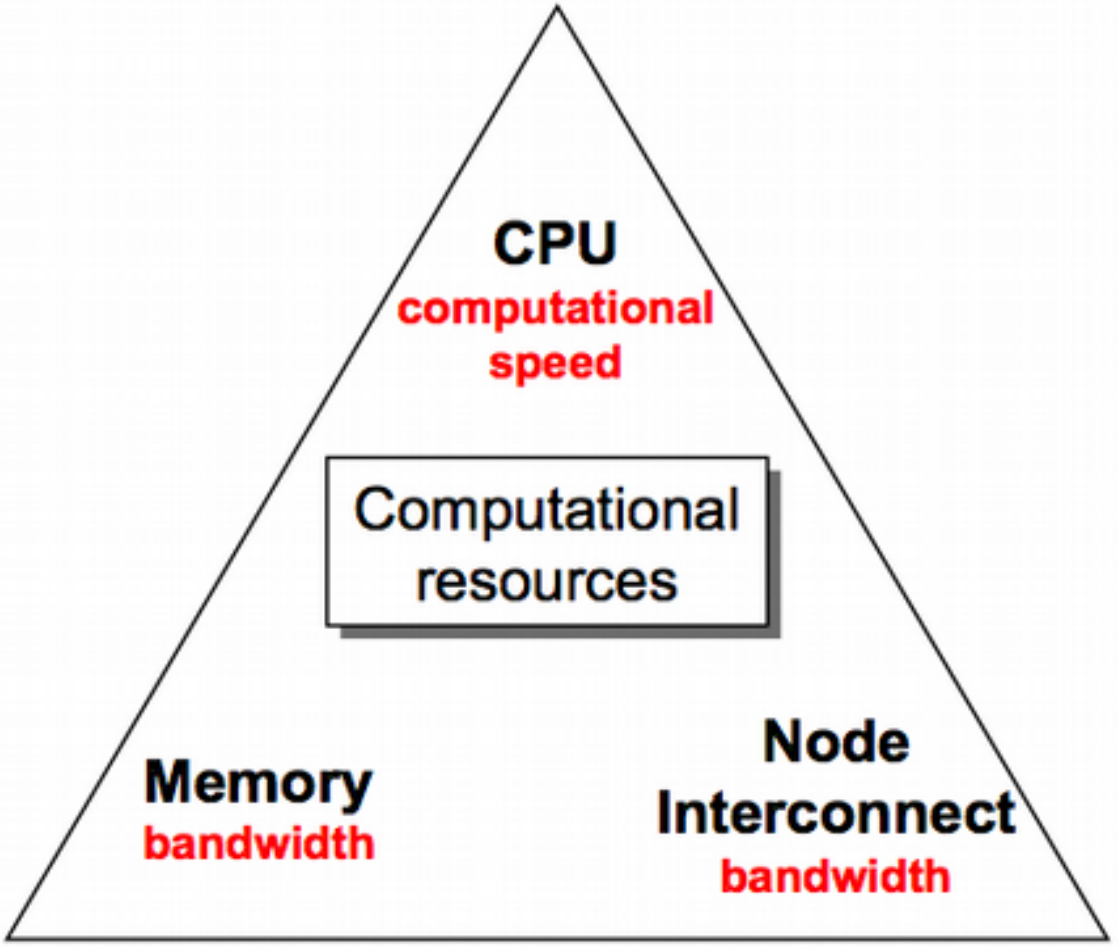
MHPC
Master in High Performance Computing

Some freely available benchmarks (2)

- Network benchmark:
  - Netpipe /Netperf (http://www.netperf.org/netperf/)
    - tcp/ip protocol and more
  - IMB-4.0 (now IMB2017) (INTEL MPI benchmark)
    - MPI protocol ()
    - https://software.intel.com/en-us/articles/intel-mpi-benchmarks
  - OSU benchmarks: http://mvapich.cse.ohio-state.edu/benchmarks/
- I/O benchmarks:
  - Iozone/b_eff_io/IOR/ Mdbench/

SISSA ICTP

resources to benchmark



MHPC
Master in High Performance Computing

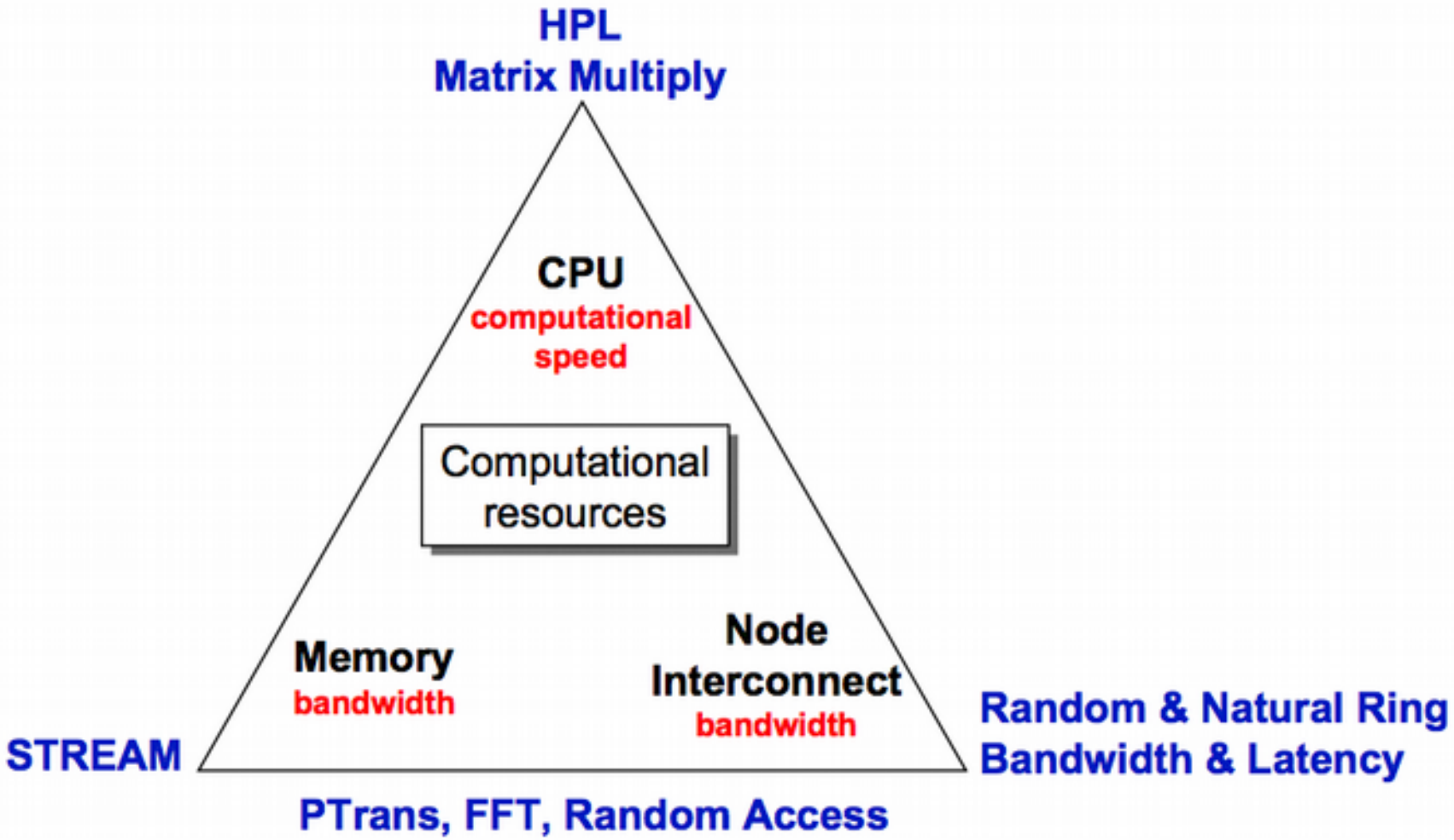
HPCC benchmark

7 tests:

1. HPL - the Linpack TPP benchmark which measures the floating point rate of execution for solving a linear system of equations.
2. DGEMM - measures the floating point rate of execution of double precision real matrix-matrix multiplication.
3. STREAM - a simple synthetic benchmark program that measures sustainable memory bandwidth (in GB/s) and the corresponding computation rate for simple vector kernel.
4. PTRANS (parallel matrix transpose) - exercises the communications where pairs of processors communicate with each other simultaneously. It is a useful test of the total communications capacity of the network.
5. RandomAccess - measures the rate of integer random updates of memory (GUPS).
6. FFT - measures the floating point rate of execution of double precision complex one-dimensional Discrete Fourier Transform (DFT).
7. Communication bandwidth and latency - a set of tests to measure latency and bandwidth of a number of simultaneous communication patterns; based on b_eff (effective bandwidth benchmark).

SISSA ICTP

HPCC components



http://icl.cs.utk.edu/hpcc/

MHPC
Master in High Performance Computing

Scientific application as benchmarks

- HPC benchmarks so far presented are not able to be too much representative of actual workload in specific scientific domain
- However running a complex application as standard benchmark could be difficult and sometime almost impossible.
- There are also the so called Mini-applications: smaller version of the real production scientific package
- They provide time to solution for several kernels and strong/weak scalability information

SISSA ICTP

MHPC
Master in High Performance Computing

Some examples of miniapplication

- CloverLeaf: CloverLeaf is a miniapp that solves the compressible Euler equations on a Cartesian grid, using an explicit, second-order accurate method.
- CoMD: A simple proxy for the computations in a typical molecular dynamics application. The reference implementation mimics that of SPaSM. In addition, we provide an OpenCL implementation which allows testing on multicore and GPU architectures, with both array-of-structures and structure-of-arrays data layouts. More information can be found at https://github.com/exmatex/CoMD.
- MiniFE: MiniFE is an proxy application for unstructured implicit finite element codes. It is similar to HPCCG and pHPCCG but provides a much more complete vertical covering of the steps in this class of applications. MiniFE also provides support for computation on multicore nodes, including pthreads and Intel Threading Building Blocks (TBB) for homogeneous multicore and CUDA for GPUs. Like HPCCG and pHPCCG, MiniFE is intended to be the "best approximation to an unstructured implicit finite element or finite volume application, but in 8000 lines or fewer."
- MiniMD: A simple proxy for the force computations in a typical molecular dynamics applications. The algorithms and implementation used closely mimics these same operations as performed in LAMMPS.
- MiniGhost: A Finite Difference proxy application which implements a difference stencil across a homogenous three dimensional domain.
- MiniXyce: A portable proxy of some of the key capabilities in the electrical modeling Xyce.

SISSA ICTP

MHPC
Master in High Performance Computing

Let us play with benchmarks !

- STREAM
- HPL
- HPCG

SISSA ICTP

MHPC
Master in High Performance Computing

STREAM benchmark

- Created in 1991
- Intended to be a oversimplified representation of low-computing intensity and long vector operations
- Widely used, more 1100 results in the database
- Hosted at www.cs.virginia.edu/stream

SISSA ICTP

MHPC
Master in High Performance Computing

STREAM benchmark

- Four kernels, separately timed:

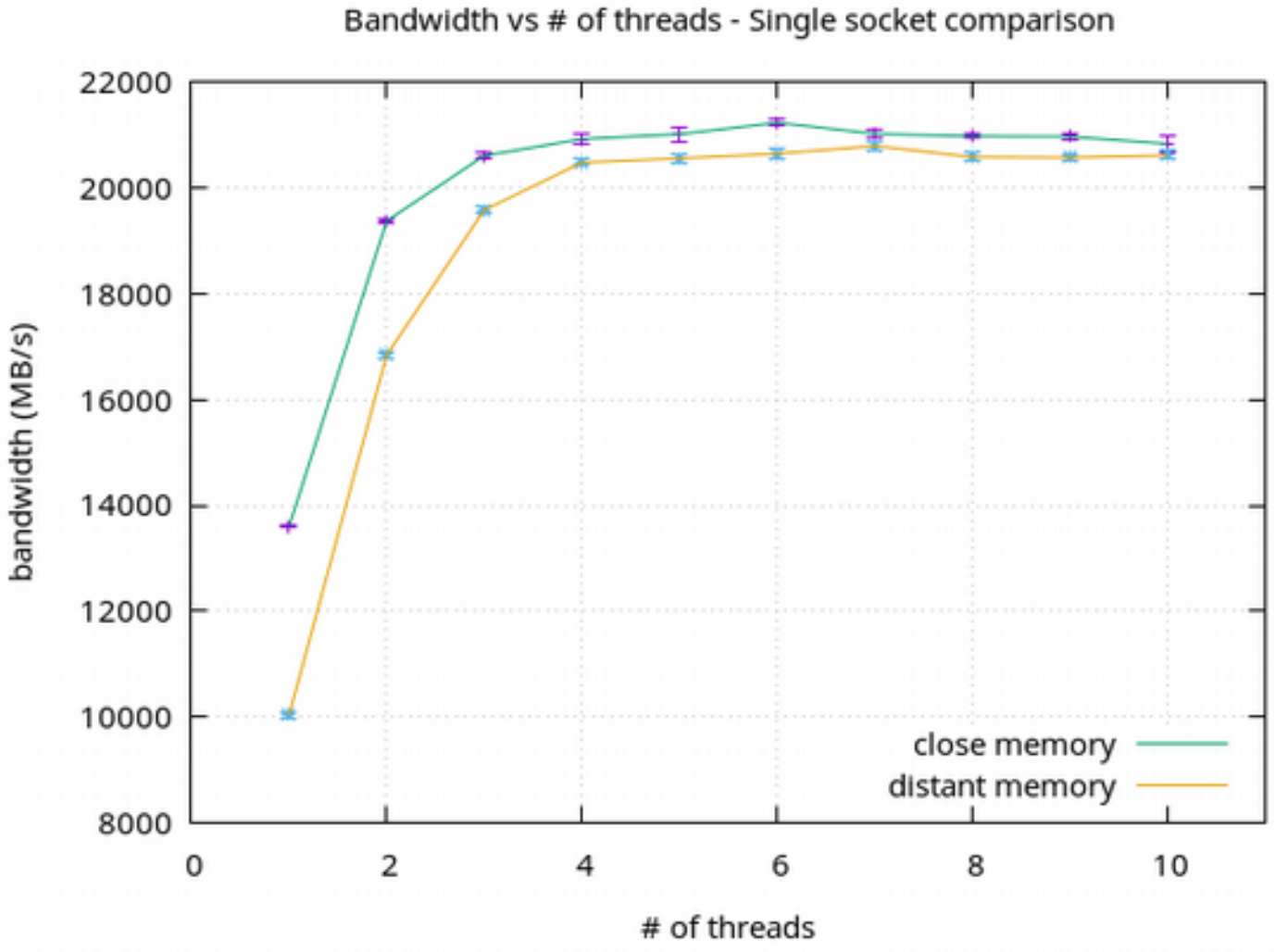
```
Copy:   C[i] = A[i];                    16 Bytes
Scale:  B[i] = scalar * C[i];           16 Bytes
Add:    C[i] = A[i] + B[i];             24 Bytes
Triad   A[i] = B[i] + scalar*C[i];      24 Bytes
```

- N choosen to make each array >> cache size
- Repeated several times, first iteration ignored
- Min/Max/Avg reported and the best time used to compute Bandwidth

SISSA ICTP

MHPC
Master in High Performance Computing

STREAM results



SISSA ICTP

MHPC
Master in High Performance Computing

HPL

- From http://icl.cs.utk.edu/hpl/index.html:

  The code solves a uniformly random system of linear equations and reports time and floating-point execution rate using a standard formula for operation count.

  - Number_of_floating_point_operations = $2/3n^3 + 2n^2$ (n=size of the system)

```
================================================================================
T/V                N    NB     P     Q               Time                 Gflops
--------------------------------------------------------------------------------
WR03R2L2       86000  1024     2     1             191.06              2.219e+03
--------------------------------------------------------------------------------
||Ax-b||_oo/(eps*(||A||_oo*||x||_oo+||b||_oo)*N)=        0.0043644 ..... PASSED
```

SISSA ICTP

MHPC
Master in High Performance Computing

HPL has some problems

HPL performance of computer systems are no longer so strongly correlated to real application performance, especially for the broad set of HPC applications governed by partial differential equations.

Designing a system for good HPL performance can actually lead to design choices that are wrong for the real application mix, or add unnecessary components or complexity to the system.

SISSA ICTP

MHPC
Master in High Performance Computing

Concerns

- The gap between HPL predictions and real application performance will increase in the future.
- A computer system with the potential to run HPL at an Exaflop is a design that may be very unattractive for real applications.
- Future architectures targeted toward good HPL performance will not be a good match for most applications.
- This leads us to a think about a different metric

SISSA ICTP

MHPC
Master in High Performance Computing

HPCG benchmark

- High Performance Conjugate Gradient (HPCG).
  - Solves Ax=b, A large, sparse, b known, x computed.
  - An optimized implementation of PCG contains essential computational and communication patterns that are prevalent in a variety of methods for discretization and numerical solution of PDEs
- Patterns:
  - Dense and sparse computations.
  - Dense and sparse collective.
  - Data-driven parallelism (unstructured sparse triangular solves).
- Strong verification and validation properties (via spectral properties of CG)

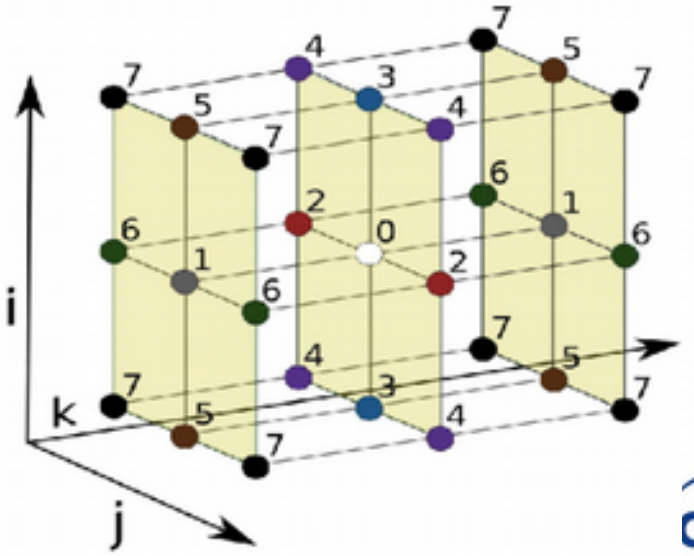
http://www.hpcg-benchmark.org/index.html

SISSA ICTP

MHPC
Master in High Performance Computing

Model problem description

- Synthetic discretized 3D PDE (FEM, FVM, FDM).
- Local domain: $n_x\, n_y\, n_z$ (see hpcg.dat)
- Process layout: $p_x * p_y * p_z$ ($p_x * p_y * p_z$=N of processor)
- Global domain: $(p_x * n_x)$ X $(p_y * n_y)$ X $(p_z * n_z)$
- Sparse matrix:
  - 27 nonzeros/row interior.
  - 8 – 18 on boundary.
- Symmetric positive definite



27-point stencil operator

CTP

November 2017 HPCG Results

November 2017 HPCG Results

Rank	Site	Computer	Cores	HPL Rmax (Pflop/s)	TOP500 Rank	HPCG (Pflop/s)	Fraction of Peak
1	RIKEN Advanced Institute for Computational Science Japan	**K computer** – , SPARC64 VIIIfx 2.0GHz, Tofu interconnect Fujitsu	705,024	10.510	10	0.603	5.3%
2	NSCC / Guangzhou China	**Tianhe-2 (MilkyWay-2)** – TH-IVB-FEP Cluster, Intel Xeon 12C 2.2GHz, TH Express 2, Intel Xeon Phi 31S1P 57-core NUDT	3,120,000	33.863	2	0.580	1.1%
3	DOE/NNSA/LANL/SNL USA	**Trinity** – Cray XC40, Intel Xeon E5-2698 v3 300160C 2.3GHz, Aries Cray	979,072	14.137	7	0.546	1.8%
4	Swiss National Supercomputing Centre (CSCS) Switzerland	**Piz Daint** – Cray XC50, Intel Xeon E5-2690v3 12C 2.6GHz, Cray Aries, NVIDIA Tesla P100 16GB Cray	361,760	19.590	3	0.486	1.9%
5	National Supercomputing Center in Wuxi China	**Sunway TaihuLight** – Sunway MPP, SW26010 260C 1.45GHz, Sunway NRCPC	10,649,600	93.015	1	0.481	0.4%
6	Joint Center for Advanced High Performance Computing Japan	**Oakforest-PACS** – PRIMERGY CX600 M1, Intel Xeon Phi Processor 7250 68C 1.4GHz, Intel Omni-Path Architecture Fujitsu	557,056	13.555	9	0.385	1.5%
7	DOE/SC/LBNL/NERSC USA	**Cori** – XC40, Intel Xeon Phi 7250 68C 1.4GHz, Cray Aries Cray	632,400	13.832	8	0.355	1.3%
8	DOE/NNSA/LLNL USA	**Sequoia** – IBM BlueGene/Q, PowerPC A2 1.6 GHz 16-core, 5D Torus IBM	1,572,864	17.173	6	0.330	1.6%

MHPC
Master in High Performance Computing

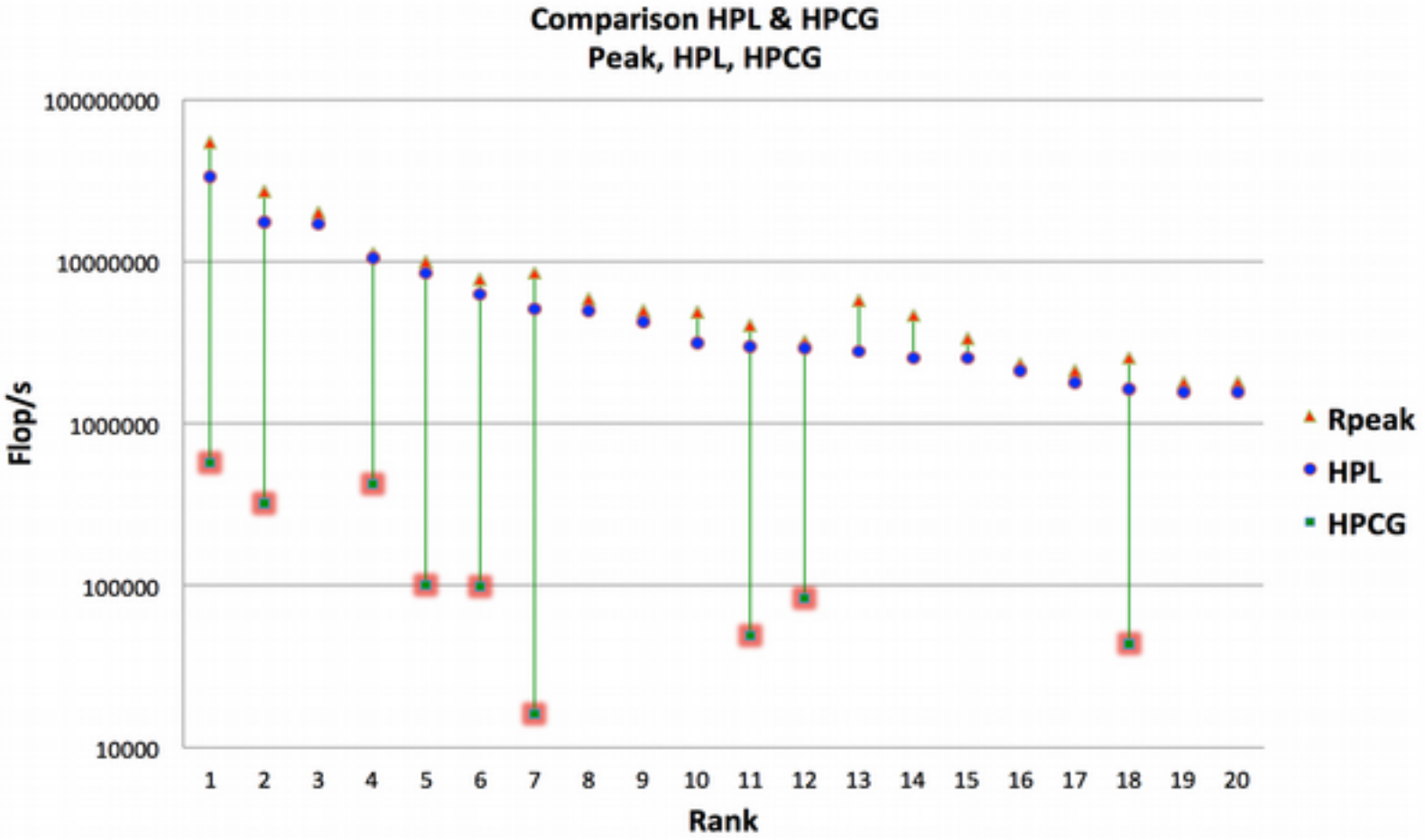
HPCG benchmark

- Balanced BW and compute:
- HPCG is memory BW bound in modern processors
  - 6 Byte/FLOP
  - HPCG can utilize at most x/6 of peak FLOP
- Scalable collectives: HPCG uses all-reduce
- Efficient parallelization of Gauss-Seidel: HPCG spends 2/3 of time in GS

SISSA ICTP

MHPC
Master in High Performance Computing

Comparison HPL vs HPCG



SISSA ICTP

MHPC
Master in High Performance Computing

Remember:

THERE IS NO BENCHMARK THAT SUBSTITUTES your own code on your dataset

Measurement should be done by you on your code !

SISSA ICTP

MHPC
Master in High Performance Computing

Tips to benchmark

- use /usr/bin/time and take note of all times

  wall time/ user time /sys time

- repeat the same run at least a few time to estimate the fluctuations of the numbers (this should be generally within a few percent)
- be sure to be alone on the system you are using and with no major perturbation on your cluster
- execution runs should be at least in the order of tens of minutes
- always check the correctness of your scientific output

SISSA ICTP

MHPC
Master in High Performance Computing

Performance Evaluation process

- Monitoring your System:
  - Use monitoring tools to better understand your machine's limits and usage
    - is the system limit well suited to run my application ?
  - Observe both overall system performance and single-program execution characteristics. Monitoring your own code
    - Is the system doing well ? Is my program running in a pathological situation ?
- Monitoring your own code:
  - Timing the code:
    - timing a whole program (time command :/usr/bin/time)
    - timing a portion (all portions) of the program
  - Profiling the program (already seen)

SISSA ICTP

MHPC
Master in High Performance Computing

Tools to monitor your system

- atop - Advanced System & Process Monitor
- iotop - simple top-like I/O monitor
- iftop - display bandwidth usage on an interface by host
- dstat - versatile tool for generating system resource statistics
- vmstat – to check memory

SISSA ICTP

MHPC
Master in High Performance Computing

Common best practice

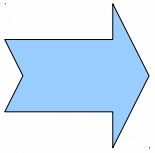
- Use speed-up with care !
- When publishing parallel speed-up, report in the base case is a single parallel process or best serial execution, as well the as the absolute execution performance of the base case

SISSA ICTP

MHPC
Master in High Performance Computing

Optimization Techniques

- There are basically three different categories:
  - Improve memory performance (The most important)
  - Improve CPU performance
  - Use already highly optimized libraries/subroutines



MHPC
Master in High Performance Computing

What are performance libraries ?

- Routines for common (math) functions such as vector and matrix operations, fast Fourier transform etc. written in a specific way to take advantage of capabilities of the CPU.
- Each CPU type normally has its own version of the library specifically written or compiled to maximally exploit that architecture

SISSA ICTP

MHPC
Master in High Performance Computing

Why use performance libraries ?

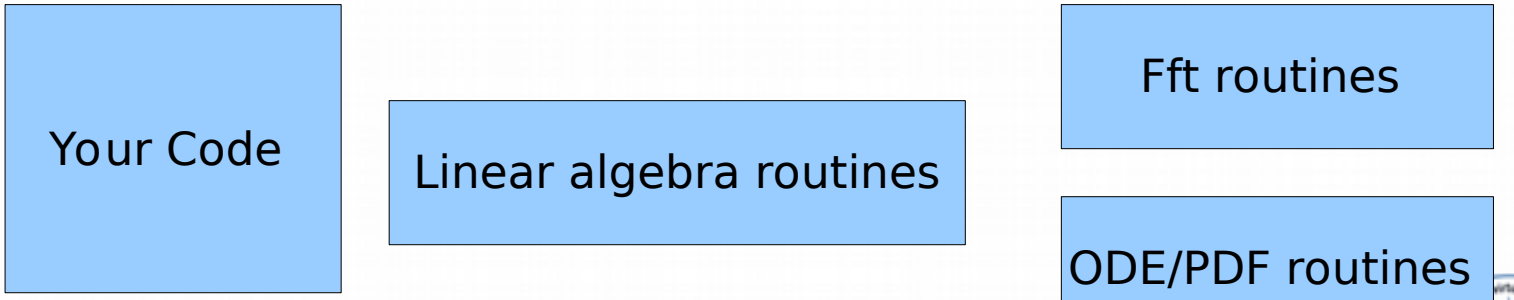
- Compilers can optimize code only to a certain point. Effective programming needs deep knowledge of the platform
- Performance libraries are designed to use the CPU in the most efficient way, which is not necessarily the most straightforward way.
- It is normally best to use the libraries supplied by or recommended by the CPU vendor
- On modern hardware they are hugely important, as they most efficiently exploit caches, special instructions and parallelism

SISSA ICTP

MHPC
Master in High Performance Computing

Any other reason apart from optimization ?

- Usage of libraries
  - Make coding easier. Complicated math operations can be used from existing routines
  - Increase portability of code as standard (and well optimized) libraries exist for ALL computing platforms.
- Lego approach: build your own code using available bricks..



SISSA ICTP

MHPC
Master in High Performance Computing

What is available ?

- Linear Algebra: BLAS/LAPACK/SCALAPACK
- FFT:
  - FFTW
- ODE/PDE
  - PETSC
- And many others…
- Good Starting point: www.netlib.org

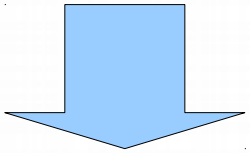
SISSA ICTP

MHPC
Master in High Performance Computing

Should I write my own algorithm for L. A. ?

99.99% of time NO

- Tons of libraries out there
- Well tested
- Extremely efficient in 99.99% of the case
- With some "de facto" standard implemented



PORTABILITY IS COMING (almost) FOR FREE

SISSA ICTP

MHPC
Master in High Performance Computing

Standardization (BLAS example)

- Subroutines have a standardized layout
- BLAS is documented in the source code
- Man pages exist
- Vendor supplied docs
- Different BLAS implementations have the same calling sequence

```
      SUBROUTINE DGEMM ( TRANSA, TRANSB, M, N, K, ALPHA, A, LDA, B, LDB,
     $                   BETA, C, LDC )
*     .. SCALAR ARGUMENTS ..
      CHARACTER*1        TRANSA, TRANSB
      INTEGER            M, N, K, LDA, LDB, LDC
      DOUBLE PRECISION   ALPHA, BETA
*     .. ARRAY ARGUMENTS ..
      DOUBLE PRECISION   A( LDA, * ), B( LDB, * ), C( LDC, * )
*     ..
*
*  PURPOSE
*  =======
*
*  DGEMM  PERFORMS ONE OF THE MATRIX-MATRIX OPERATIONS
*
*     C := ALPHA*OP( A )*OP( B ) + BETA*C,
*
*  WHERE  OP( X ) IS ONE OF
*
*     OP( X ) = X   OR   OP( X ) = X',
*
*  ALPHA AND BETA ARE SCALARS, AND A, B AND C ARE MATRICES, WITH OP( A )
*  AN M BY K MATRIX,  OP( B )  A  K BY N MATRIX AND  C AN M BY N MATRIX.
*
*  PARAMETERS
*  ==========
*
*  TRANSA - CHARACTER*1.
*           ON ENTRY, TRANSA SPECIFIES THE FORM OF OP( A ) TO BE USED IN
*           THE MATRIX MULTIPLICATION AS FOLLOWS:
*
*              TRANSA = 'N' OR 'N',  OP( A ) = A.
*
*              TRANSA = 'T' OR 'T',  OP( A ) = A'.
*
*              TRANSA = 'C' OR 'C',  OP( A ) = A'.
*
*           UNCHANGED ON EXIT.
*
*  TRANSB - CHARACTER*1.
*           ON ENTRY, TRANSB SPECIFIES THE FORM OF OP( B ) TO BE USED IN
*           THE MATRIX MULTIPLICATION AS FOLLOWS:
*
*              TRANSB = 'N' OR 'N',  OP( B ) = B.
*
*              TRANSB = 'T' OR 'T',  OP( B ) = B'.
```

SISSA

MHPC
Master in High Performance Computing

comment on BLAS

Basic Linear Algebra Subroutines

Name	Description	Examples
Level-1 BLAS	Vector Operations	$C = \sum X_i Y_i$
Level-2 BLAS	Matrix-Vector Operations	$B_i = \sum_k A_{ik} X_k$
Level-3 BLAS	Matrix-Matrix Operations	$C_{ij} = \sum_k A_{ik} B_{kj}$

Efficiency: q parameter

Table 2: Basic Linear Algebra Subroutines (BLAS)

Operation	Definition	Floating point operations	Memory references	q
saxpy	$y_i = \alpha x_i + y_i,\ i = 1, \ldots, n$	$2n$	$3n+1$	$2/3$
Matrix–vector mult	$y_i = \sum_{j=1}^{n} A_{ij} x_j + y_i$	$2n^2$	$n^2 + 3n$	2
Matrix–matrix mult	$C_{ij} = \sum_{k=1}^{n} A_{ik} B_{kj} + C_{ij}$	$2n^3$	$4n^2$	$n/2$

The parameter q is the ratio of flops to memory references. Generally:

1. Larger values of q maximize useful work to time spent moving data.
2. The higher the level of the BLAS, the larger q.

It follows..

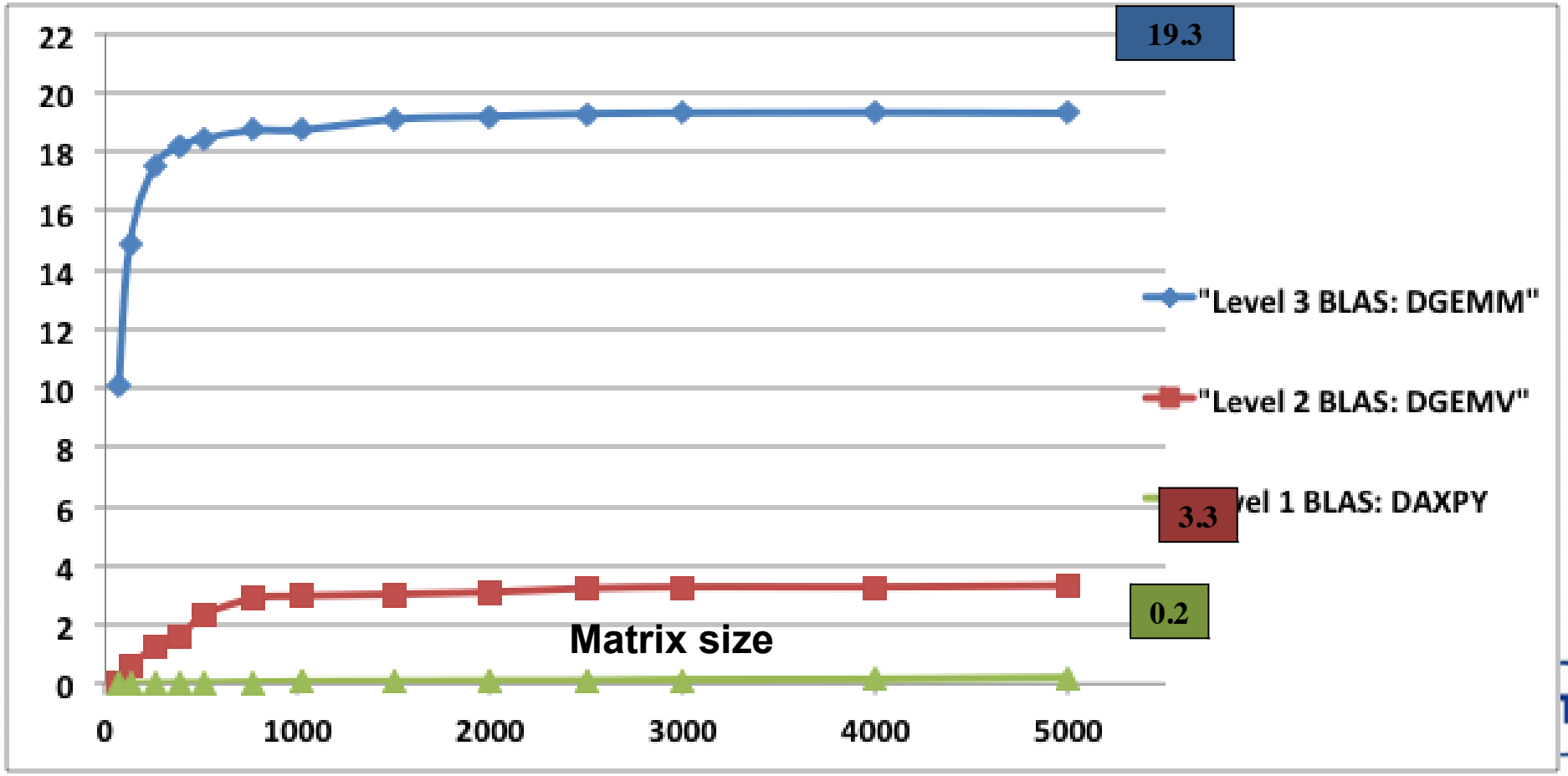
- BLAS1 are memory bounded ! (for each computation a memory transfer is required)
- BLAS2 are not so memory bounded (can have good performance on super-scalar architecture)
  - BLAS3 can be very efficient on super-scalar computers because not memory bounded

It follows…

-

BLAS performance

1 core Intel Xeon E5-2670 (Sandy Bridge), 2.6 GHz.
24 MB shared L3 cache, and each core has a private 256 KB L2 and 64 KB L1.
The theoretical peak for this architecture in double precision is 20.8 Gflop/s per core.
Compiled with gcc 4.4.6 and using MKL_composer_xe_2013.3.163



MHPC
Master in High Performance Computing

Vendor/Optimized BLAS libraries

- ACML
  - The AMD Core Math Library, supporting the AMD processors
- ATLAS
  - Automatically Tuned Linear Algebra Software, an open source implementation of BLAS APIs for C and Fortran 77
- Intel MKL
  - The Intel Math Kernel Library, supporting x86 32-bits and 64-bits. Includes optimizations for Intel Pentium, Core and Intel Xeon CPUs and Intel Xeon Phi; support for Linux, Windows and Mac OS X
- cuBLAS
  - Optimized BLAS for NVIDIA based GPU cards
- clBLAS
  - An OpenCL implementation of BLAS
- ESSL
  - IBM's Engineering and Scientific Subroutine Library, supporting the PowerPC architecture under AIX and Linux
- GotoBLAS
  - Kazushige Goto's BSD-licensed implementation of BLAS, tuned in particular for Intel Nehalem/Atom, VIA Nanoprocessor, AMD Opteron
- BLIS
  - BLAS-like Library Instantiation Software framework for rapid instantiation
- OpenBLAS
  - Optimized BLAS based on Goto BLAS hosted at GitHub, supporting Intel Sandy Bridge and MIPS_architecture Loongson processorss

SISSA ICTP

MHPC
Master in High Performance Computing

How to link optimized libraries?

- Reference implementation: order matters !
  - LAPACK uses BLAS

    => -L/usr/local/lib -llapack -lblas
- OpenBLAS:
  - Automatically includes lapack reference implementation so no need to specify anything else. Please check !
- ATLAS is written C with f77 wrappers:
  - -L/opt/atlas/lib -lf77blas -latlas
- MKL:
  - Generally complex and highly dependent on version and/or HW/ SW implementation
  - https://software.intel.com/en-us/articles/intel-mkl-link-line-advisor

SISSA ICTP

Running HPL on C3HPC (precompiled version)

- Connect to the machine
  - `ssh youraccount@hpc.c3e.cosint.it`
- Submit interactive job:
  - `qsub -I -l nodes=$NODE:ppn=24 -l walltime=6:0:0 -q mhpc`
- Identify the right executable in /lustre/mhpc/eas/hpl/bin
  - `xhpl.plasma-gnu`
  - `xhpl.openblas`
  - `xhpl.mkl-gnu`
  - `xhpl.atlas`
  - `hpl.plasma-mkl`
- Execute it:
  - Load appropriate modules
  - Run it:
  - `mpirun -np XX /lustre/mhpc/eas/hpl/bin/xhpl.mkl-gnu`

MHPC
Master in High Performance Computing

A few notes:

- Standard input file should be present
- Beware of threads
  - How to control them ?
- Help is provided by courtesy of M.Baricevic
  - Check out wrap.sh and run.sh

SISSA ICTP

MHPC
Master in High Performance Computing

What about N ?

- N should be large enough to take ~75% of RAM..
  - N = sqrt (0.75 * Number of Nodes * Minimum memory of any node / 8)
- You can compute it via:
  - http://www.advancedclustering.com/act-kb/tune-hpl-dat-file/

SISSA ICTP

MHPC
Master in High Performance Computing

HPL benchmark input file HPL.dat

```
HPLinpack benchmark input file
Innovative Computing Laboratory, University of Tennessee
HPL.out      output file name (if any)
6            device out (6=stdout,7=stderr,file)
1            # of problems sizes (N)
50000 Ns
1             # of NBs
768         NBs
0            PMAP process mapping (0=Row-,1=Column-major)
1            # of process grids (P x Q)
4 1 2 1        Ps
4 2 2 4        Qs
16.0         threshold
1            # of panel fact
0 1 2        PFACTs (0=left, 1=Crout, 2=Right)
1            # of recursive stopping criterium
2 8          NBMINs (>= 1)
1            # of panels in recursion
2            NDIVs
1            # of recursive panel fact.
0 1 2        RFACTs (0=left, 1=Crout, 2=Right)
1            # of broadcast
0 2          BCASTs (0=1rg,1=1rM,2=2rg,3=2rM,4=Lng,5=LnM)
1            # of lookahead depth
1 0          DEPTHs (>=0)
1            SWAP (0=bin-exch,1=long,2=mix)
192          swapping threshold
1            L1 in (0=transposed,1=no-transposed) form
1            U  in (0=transposed,1=no-transposed) form
1            Equilibration (0=no,1=yes)
8            memory alignment in double (> 0)
```

SISSA ICTP

MHPC
Master in High Performance Computing

Parameters for HPL.dat input file

N	Problem size	**Pmap**	Process mapping
NB	Blocking factor	**threshold**	for matrix validity test
P	Rows in process grid	**Ndiv**	Panels in recursion
Q	Columns in process grid	**Nbmin**	Recursion stopping criteria
Depth	Lookahead depth	**Swap**	Swap algorithm
Bcasts	Panel broadcasting method	**L1, U**	to store triangle of panel
Pfacts	Panel factorization method	**Align**	Memory alignment
Rfacts	Recursive factorization method	**Equilibration**	

SISSA ICTP

MHPC
Master in High Performance Computing

Tips to get performance..

- Figure out a good block size (NB) for the matrix multiply routine. The best method is to try a few out. If you happen to know the block size used by the matrix-matrix multiply routine, a small multiple of that block size will do fine. This particular topic is discussed in the FAQs section.
- The process mapping should not matter if the nodes of your platform are single processor computers. If these nodes are multi-processors, a row-major mapping is recommended.
- HPL likes "square" or slightly flat process grids. Unless you are using a very small process grid, stay away from the 1-by-Q and P-by-1 process grids.

SISSA ICTP

MHPC
Master in High Performance Computing

What your are supposed to do:

- Test a few N and Nb to identify the most performing one..
- Tip:
  - Write a small script and submit it before lunch
- Compute the ratio between theoretical peak performance and sustained hpl performance
  - If the result is less than 75% leave the master

SISSA ICTP

MHPC
Master in High Performance Computing

Run the hpcg benchmark

- The hpcg.dat file

```
#/lustre/mhpc/eas/hpcg/bin/xhpcg.mpi
#time=300 size=64
64 64 64
300
```

- Time is in second / size is local to the node (weak scaling approach)
-

SISSA ICTP

MHPC
Master in High Performance Computing

Some interesting documents/links

- http://crd-legacy.lbl.gov/~dhbailey/dhbpapers/twelve-ways.pdf

  Twelve Ways to Fool the Masses When Giving Performance Results on Parallel Computers

  David H. Bailey June 11, 1991

- http://www.hpcwire.com/2011/12/13/ten_ways_to_fool_the_masses_when_giving_performance_results_on_gpus/
- http://htor.inf.ethz.ch/publications/img/hoefler-scientific-benchmarking.pdf

  Scientific Benchmarking of Parallel Computing Systems)Twelve ways to tell the masses when reporting performance results)

- http://www.sandia.gov/~maherou/docs/HPCG-Benchmark.pdf

SISSA ICTP