www.bsc.es

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Hands-on: porting applications to ARM multicore

PRACE Spring School 2013
New and Emerging Technologies - Programming for Accelerators

Nikola Rajovic, Gabriele Carteni
Barcelona Supercomputing Center

Outline

- Simple job submissions
  - Intel IMB benchmarks
- Tuning
  - Synthetic FP micro-benchmarks
- Porting (SW stack exploration)
  - High-Performance LINPACK
- Hands-on happy hour (maybe, if there is enough time)
  - For those interested in porting of their own codes

Setup

- All examples can be found in

/gpfs/EXAMPLES/PSS2013

Prerequisites

- User account – mailed to you
- PARAVER (for visualization)
  - Installed on your local machine from
    - http://www.bsc.es/computer-sciences/performance-tools/downloads
- Good will and patience
  - Tibidabo cluster is an experimental cluster which never served 40+ users at once. ☺

Outline

- Simple job submissions
  - Intel IMB benchmarks
- Tuning
  - Synthetic FP micro-benchmarks
- Porting (SW stack exploration)
  - High-Performance LINPACK
- Hands-on happy hour (maybe)
  - For those interested in porting of their own codes

HANDS-ON #1: Excercise 1. Manage your job

- Get access to Tibidabo by using the information on the page that we have provided to you

- Copy everything is in:

  /gpfs/EXAMPLES/PSS2013/HANDSON-1/*

  to somewhere in your $HOME directory

- **Manage your job "myjob.job"**
  - Modify the content properly, submit it, check the queue, cancel it (if you want and if you are really fast) and check the output when it is COMPLETED.

HANDS-ON #1: Excercise 2. Modules

- The Intel MPI Benchmark is a suite of benchmarks to assess performance of the cluster network, MPI library implementations and compilers on communication

- Use the code **IMB-MPI1**, it is compiled with MPICH2

- Check the modules which are loaded, purge them and load only "mpich2”

HANDS-ON #1: Excercise 3. Single Transfer benchmarks

- 3.1) PingPong and PingPing within a node
  - use **total_tasks**, **cpus_per_task** and **tasks_per_node** properly
- 3.2) PingPong and PingPing between 2 cpus belonging to different nodes
- hint-1 (runs both at once):

  srun ~/IMB-MPI1 PingPong PingPing
- hint-2:

  use the command "paste" to compare the output

HANDS-ON #1: Excercise 4. Parallel Transfer benchmark

- 4) Run Parallel Transfer benchmark **Sendrecv**:
  - Try to use different combinations of **total_tasks**, **cpus_per_task** and **tasks_per_node** and check the output

HANDS-ON #1: Excercise 5. Collective benchmark

- 5) Run Collective benchmarks **Allgather** and **Alltoall**:
  - Try to use different combinations of **total_tasks**, **cpus_per_task** and **tasks_per_node** and check the output

Outline

- Simple job submissions
  - Intel IMB benchmarks
- Tuning
  - Synthetic FP micro-benchmarks
- Porting (SW stack exploration)
  - High-Performance LINPACK
- Hands-on happy hour (maybe)
  - For those interested in porting of their own/external codes

Synthetic benchmarks

- Microkernels
  - To test the FP performance of Cortex-A9 CPU
  - Developed to see if we can reach peak 1 GFLOPS
  - We will use it to test the importance of correct gcc flags

/gpfs/EXAMPLES/PSS2013/ex2_fp

FP addition

```
double *A;
double accum;
… … …
gettimeofday(&start, 0);

for (j=0; j<t; j++) {
        acum  = 0;

        for (i=n; i!=0; i--) {
            acum  += A[i];
}
}

gettimeofday(&end, 0);
```

- Sums all elements of an array
  - Double-precision FP
  - Repeats for a given number of times
- 1 GFLOPS
  - Expected when everything fits into L1 cache

FP multiply-add

```
double *A,*B;
double accum;
… … …
gettimeofday(&start, 0);

for (j=0; j<t; j++) {
    acum  = 0;
    for (i=0; i<n; i++) {
        acum  += A[i] * B[i];
}
}

gettimeofday(&end, 0);
```

- Vector dot product
  - Double-precision FP
  - Repeats for a given number of times
- 1 GFLOPS
  - Expected when everything fits into cache

Forgot about GCC flags?

- `-march=armv7a -mcpu=cortex-a9 -mtune=cortex-a9`
  - Specifies the target CPU
    - gcc chooses the correct instructions to emit
    - Activates CPU-specific optimizations

- `-mfloat-abi=softfp`
  - Generates HW floating point instructions
  - Soft-FP calling conventions (affects function calls)

- `-mfp=vfpv3-d16`
  - Specifies floating point hardware that is available in the CPU

The importance of correct flags

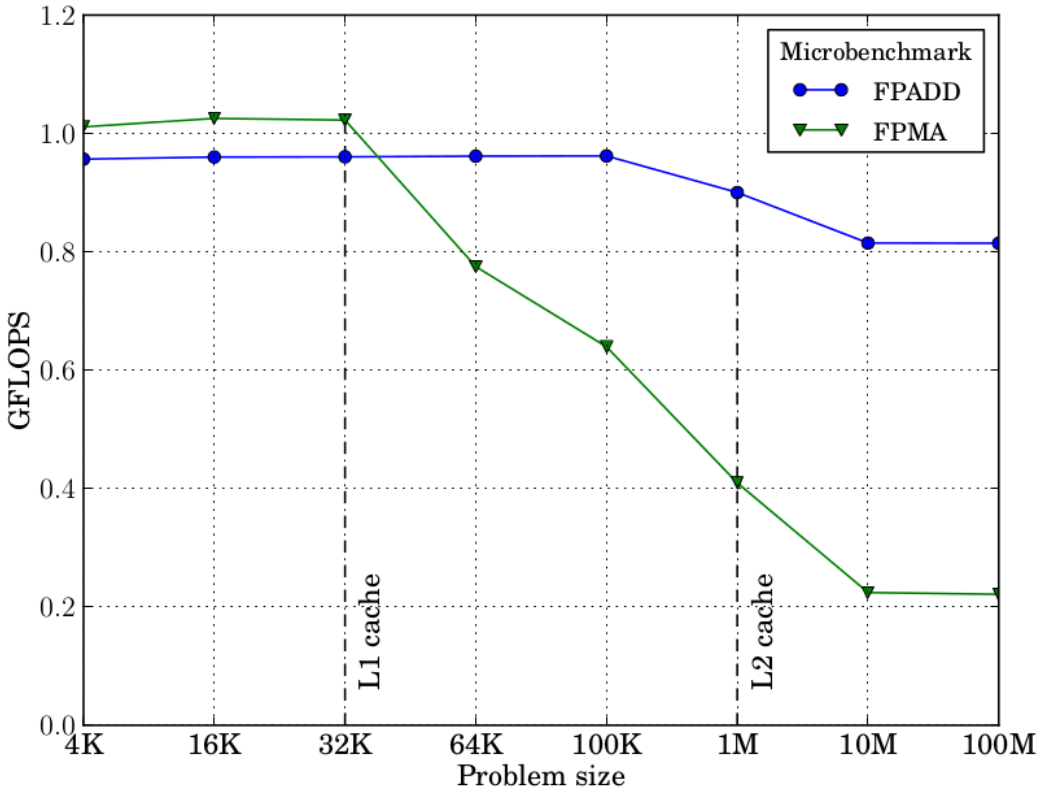
- Execute synthetic benchmarks

```
make
mnsubmit job.slurm
```

- Observe the difference in reported MFLOPS for different versions
  - **~12x**
- Still not 1 GFLOPS
  - Is the FP pipeline capable of delivering this performance?

Can we achieve 1 GFLOPS?

- Yes, but we need to find a way to feed floating point unit properly with the data…if data reuse is there, obviously we can do it
  - This is what is possible to achieve, any idea?



Outline

- Simple job submissions
  - Intel IMB benchmarks
- Tuning
  - Synthetic FP micro-benchmarks
- Porting (SW stack usage)
  - High-Performance LINPACK

- Hands-on happy hour (maybe)
  - For those interested in porting of their own codes

High Performance LINPACK

- Official Top500 list benchmark
  - Rank HPC Machines by the rate of solving the dense systems of linear equations in double precision arithmetic
  - Lets see how good is Tibidabo

- Assignment: port and execute the benchmark on Tibidabo
  - Source code is provided (hpl-2.1.tar.gz)
  - Compile it for OpenMPI
  - Input file is provided (HPL.dat)
  - Use ATLAS library as CBLAS backend (/gpfs/LIBS/BIN/ATLAS)

/gpfs/EXAMPLES/PSS2013/ex3_hpl

High Performance LINPACK

- Execute Linpack on one node
  - Set the block size Nb=160
  - Set the problem size as N=X*Nb so that it fits in ~750 MB of memory
  - Set the process grid map to P=1 Q=1
  - Save the results for the later comparison

High Performance LINPACK

- Make two runs of LINPACK with following parameters:
  - N to fit in 4*750MB
  - Nb=160 and Nb=1600
  - P=2 Q=4
  - What can you notice?