Evaluation of HCM: a new model to predict the execution time of regular parallel applications on a heterogeneous cluster

Thiago Marques Soares, Rodrigo W. dos Santos and **Marcelo Lobosco**

Federal University of Juiz de Fora

10-13 September, 2017

Motivation

- Clusters are becoming heterogeneous
  - Some of them mix distinct processors, accelerators, and network connections
    - AMD, Intel, Fermi, Tesla, Ethernet, Infiniband in a single system
- To explore simultaneously all the resources available in such a heterogeneous platform, a data-parallel application must divide its data across multiple devices
  - Distinct processing power of devices and the distinct latencies of the networks
  - Which configuration leads to the best speedup?

Contribution

- Present HCM (Heterogeneous Cluster Model), a new parallel model that estimates the execution time of applications running on heterogeneous clusters
  - Extends some characteristics of our previous model
- The idea is to use the results of this estimation to predict the configuration that leads to the best speedup
  - Taking into account not only the processing power of each processor and accelerator, but also the communication costs.

Related works

- Lastovetsky *et alli*
  - Heterogeneous processors interconnected by an Ethernet-based network
    - Single network type
- HLoGP model
  - Takes into account the heterogeneity of both computation and communication resources
  - Large number of parameters is an issue
- This work proposes a simpler model that predicts the execution time of regular parallel applications on small clusters
  - Regardless of the computational environment used, homogeneous or heterogeneous one.

Heterogeneous Cluster Model

- Considers that execution is composed by two phases: computation and communication
  - All devices can be used, simultaneously, in the computation

Heterogeneous Cluster Model

- Steps to estimate the execution time of a regular application
  - Parameters and variables are used to describe mathematically the computation and communication phases of an application
  - Collect time spent in one of the computational platforms to execute a small number of sequential steps
  - Collect parameters from the heterogeneous environment

Estimating the computation time

Thiago Marques Soares, Rodrigo W. dos Santos and **Marcelo Lobosco**

- Parameter and variables used:
  - R_P, the relative computing power of a processing unit;
  - **size**, the size of the problem;
  - **I**, the total number of iterations.
- The value of R_P can be collected once, running a benchmark on the new processor/accelerator that is been included in the environment.

Estimating the computation time

Thiago Marques Soares, Rodrigo W. dos Santos and **Marcelo Lobosco**

$$T_{computation} = \frac{I}{I_s} \times \left(\frac{T_s}{Sum_{Rp} + F_r}\right) \quad (1)$$

- **I**, the total number of iterations;
- I_s, number of sequential iterations that will be used to predict the computation time of the application;
- T_s, time to execute Is;
- Sum_{Rp}, sum of R_p for all processors that will be used in the parallel execution
- F_r, a correction factor

Estimating the communication time

Thiago Marques Soares, Rodrigo W. dos Santos and **Marcelo Lobosco**

- Propose the use of a modified version of the LogP model
  - **P**, the number of processing units used;
  - $\mathbf{L}_d$ represents an upper bound on the communication latency of a device d;
  - $\mathbf{o}_d$ represents the overhead in device d
  - $\mathbf{g}_d$ represents the minimum time interval between consecutive message transmissions/receptions by a processor in a device d (gap)
  - N_{op} represents the number of communication operations per iteration, and
  - **M** represents the message size.

Estimating the communication time

- The communication time depends on the type of message sent (point-to-point or collective) and the message size.
- The cost of a single message is equal to

$$T_{Single(Send/SendReceive)} = N_{op} \times (L_d + \frac{M}{B_d} + o_d). \quad (2)$$

- The cost of all-to-all communication pattern is equal to

$$T_{AlltoAll} = N_{op} \times (P - 1) \times (L_d + \frac{M}{B_d} + o_d). \quad (3)$$

- The cost of all reduce communication pattern is equal to

$$T_{AllReduce} = N_{op} \times log_2 P \times (L_d + \frac{M}{B_d} + o_d). \quad (4)$$

Estimating the communication time

- How to measure the values of the latency ($\mathbf{L}_d$), gap ($\mathbf{g}_d$) and overhead ($\mathbf{o}_d$)?
  - Network benchmark is used for this purpose
  - Benchmark is executed for each type d of network that is available
    - Collects their values for distinct message sizes, ranging from 0 to 4MB

Estimating the computation and communication time

- Use of benchmarks to collect the communication costs, overheads, as well as the relative performance of the processors and accelerators, can be executed only once
  - Each time a new hardware or network is included in the system

Model Evaluation

Thiago Marques Soares, Rodrigo W. dos Santos and **Marcelo Lobosco**

- NAS benchmark were used in the initial validation of the model
  - Benchmarks were developed to execute in a CPU environment
- HIS (human immune system) simulator was chosen to evaluate the model on a hybrid environment
  - Uses GPUs and CPUs simultaneously

IS

Algorithm 1 Integer Sort

1: **for** i=1; i<=I; i++ **do**
2: generate sequence of rand numbers and subsequent keys on all processors ...
3: get the bucket size for the entire problem using MPI_Allreduce ...
4: determine the redistribution of keys ...
5: redistribute using MPI_AlltoAll ...
6: send the keys to the respective processors using MPI_Alltoallv ...
7: determine total # of keys on all other processors using MPI_Send_Receive ...
8: **end for**

$$T_{total} = T_{computation} + I \times (T_{AllReduce} + T_{AlltoAll} + T_{SendReceive}) \quad (5)$$

CG

Algorithm 2 Conjugate Gradient

```
1: for i=1; i<=I; i++ do
2:     calls the conjugate gradient routine:
   3:          obtain rho with a sum-reduce using MPI_Send ...
   4:          sum the partition submatrix-vec A.z's across rows using MPI_Send ...
   5:          exchange pieces of q using MPI_Send ...
6:     normalize z to obtain x ...
7: end for
```

$$T_{total} = T_{computation} + I \times T_{single} \tag{6}$$

HIS

Algorithm 3 HIS

```
main
  define the mesh slice to be computed by each GPU/CPU ...
  initialize submeshes according to their initial conditions ...
for t=1; t<=I; t++ do
    call the functions/kernels in order to compute the PDEs ...
    use MPI_Isend and MPI_Receive to exchange boundaries between machines ...
    synchronize all machines ...
end for
end-main
```

$$T_{total} = T_{computation} + I \times T_{single} \quad (7)$$

Experimental environment

Thiago Marques Soares, Rodrigo W. dos Santos and **Marcelo Lobosco**

- Sixteen machines
  - Two distinct CPUs
    - Intel $E5620$ dual quad-core processors
    - AMD 6272 dual sixteen-core processors
    - One process per machine
  - Three distinct GPUs
    - Tesla C1060
    - Tesla M2050
    - Tesla M2075
  - Two distinct networks
    - Gigabit ethernet
    - InfiniBand

Parameters

Table: Values of R_P for each processing unit available in the computational platform.

Processing unit	R_P
AMD	1
INTEL	1.78
C1060	131.22
M2050	299.34
M2075	333.73
M2090	364.41

Results

Table: Results for HIS using both GPUs and CPUs and Ethernet network. All times in seconds. Both absolute and percentage errors are presented. Configuration number 1: 2 CPUs (1 AMD and 1 Intel) and 2 GPUs (M2075 and C1060). Configuration number 2: 3 CPUs (1 AMDs and 2 Intels) and 3 GPUs (1 M2075 and 2 C1060). Configuration number 3: 7 CPUs (3 AMDs and 4 Intels) and 7 GPUs (3 M2075, 2 M2050 and 2 C1060).

Configuration #	**Measured**	**Estimated**	**Error**
1	47.2	51.2	4.0/8.6%
2	57.4	57.4	0.0/0.0%
3	107.8	95.1	12.7/11.8%

Results

Table: Results for the NAS benchmark using 8 AMD processors on two distinct network cards. All times are in seconds. Both absolute and percentage errors are presented. BT and SP require a square number of processors, and executed in 9 nodes.

	Ethernet			Infiniband		
	Measured	**Estimated**	**Error**	**Measured**	**Estimated**	**Error**
FT	73.8	68.7	5.1/6.9%	23.9	21.7	2.2/9.0%
IS	10.0	9.6	0.4/3.4%	3.4	3.3	0.1/5.4%
CG	150.3	169.2	18.9/12.6%	70.5	77.9	7.4/10.5%
MG	38.2	42.3	4.1/10.6%	23.3	25.1	1.8/7.4%
EP	71.3	74.0	2.7/3.8%	71.2	74.0	2.8/3.9%
LU	77.0	74.7	2.3/3.0%	62.0	57.2	4.8/7.7%
BT*	371.1	340.5	30.6/8.3%	294.7	264.5	30.2/10.2%
SP*	309.0	334.9	25.9/8.4%	238.7	266.5	27.8/12.7%

Results

Thiago Marques Soares, Rodrigo W. dos Santos and **Marcelo Lobosco**

Table: Results for the NAS benchmark using 16 processors (8 Intel and 8 AMD) and Ethernet. All times are in seconds. Both absolute and percentage errors are presented.

	Measured	**Estimated**	**Error**
FT	65.7	61.3	4.4/6.7%
IS	4.9	4.5	0.4/7.8%
CG	262.5	253.7	8.8/3.2%
MG	51.8	46.1	5.7/11.1%
EP	28.5	27.6	0.9/3.2%
LU	62.7	57.9	4.8/7.4%
BT	245.8	259.5	13.7/5.5%
SP	343.2	305.1	38.1/11.1%

Conclusion

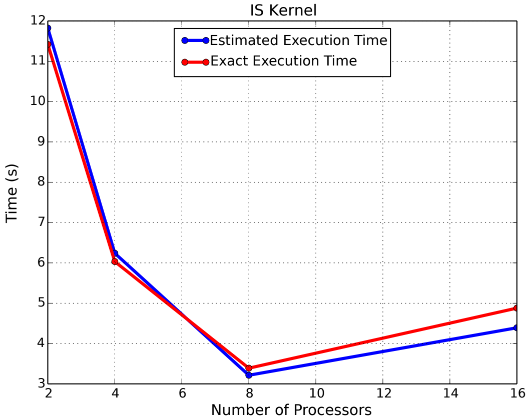
Thiago Marques Soares, Rodrigo W. dos Santos and **Marcelo Lobosco**

- HCM: a new model to predict the execution time of regular parallel applications on a small heterogeneous parallel environments.
- HCM can predict the total computation time of applications with distinct characteristics, running on distinct devices and interconnected by different network types
- Errors found during the estimation of the total execution time ranged from 0% to 12.7% in all experiments

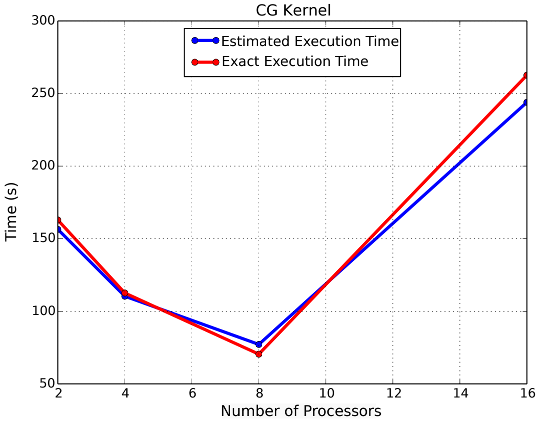
Future works

- Evaluate the model with more applications
- Use the model to choose the data partition and work assignment that minimizes the execution time of an application
  - Already Done!

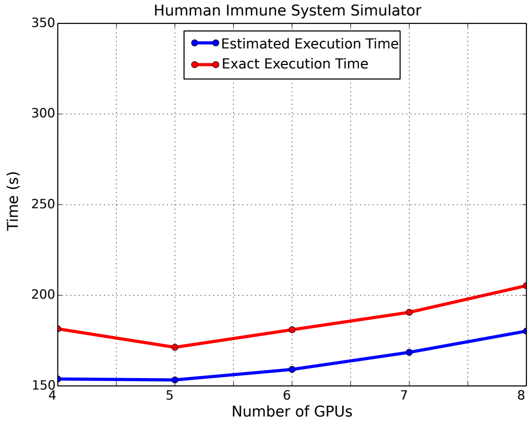
Future works



Future works



Future works



Thank you!

- The authors would like to thank UFJF, FAPEMIG, CAPES, and CNPq
- We have two open positions for visiting professors!