A Parallel Model for Heterogeneous Cluster

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

Federal University of Juiz de Fora

14-16 December, 2016

Motivation

- Clusters are becoming more heterogeneous
  - Distinct processors, accelerators, and network connections
- To explore 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 a parallel model that estimates the execution time of applications running on heterogeneous clusters
  - Extends some characteristics of the LogP model
  - Considers that processing units may have distinct computational power as well as they are interconnected by connections with distinct latencies
- The idea is to use the results of this estimation, in future works, to predict the best data division to be used in a heterogeneous cluster
  - Taking into account not only the processing power of each processor and accelerator, but also the communication and synchronization costs.

LogP Model

- Measures the effects of latency, occupancy and bandwidth on distributed memory multiprocessors
- Main parameters used in the LogP model
  - **L** represents an upper bound on the communication latency due to the use of point-to-point messages
  - **o** represents the overhead
  - **g** represents the minimum time interval between consecutive message transmissions/receptions by a processor (gap)
    - The reciprocal of the **g** parameter represents the communication bandwidth
  - **P** represents the number of processor/memory modules

Related works

- Lastovetsky *et alli*
  - Heterogeneous processors interconnected by an Ethernet-based network
    - Homogeneous network
- HLoGP model
  - Takes into account the heterogenity 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 parallel applications
  - Regardless of the computational environment used, homogeneous or heterogeneous one.

The new model

- Deal with modern heterogeneous environments, composed by distinct processors, accelerators and networks
  - $\mathbf{L}_d$ 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)
  - R_P represents the relative computing power of a processing unit
- Parameters and variables are used to describe mathematically the total execution time of an application

The new model

- How to measure the relative computing power (R_P)?
  - Running a benchmark on each processing unit to collect a metric, such as the processing units per time step
  - Using the average computation time that a processing unit takes to run some iterations of an application
- How to measure the values of the latency ($\mathbf{L}_d$) and the gap ($\mathbf{g}_d$)?
  - Network benchmark is used for this purpose
  - Benchmark is executed for each type d of network that is available
    - Collects the values of $\mathbf{L}_d$ and $\mathbf{g}_d$ for distinct message sizes, ranging from 0 to 4MB

The new model

- How to measure the overhead ($\mathbf{o}_d$)?
  - Also measured with a specific benchmark
  - It considers that the overhead varies with the message size
- 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

- Two kernels (EP and FT) and one application (SP) from the 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

EP

- Embarrassingly Parallel kernel solves a typical problem of many Monte Carlo based applications
  - Generate pairs of Gaussian pseudorandom deviates
- Communication occurs only at the end of the computation
  - Collective MPI routine is used to combine the sums generate from all processors
- Class C used in the evaluation

EP

Algorithm 1 EP

1: **main**
2: ... generate the seed for each process ...
3: ... calculate counts and sums in each process ...
4: ... Use MPI_Allreduce to send parameter to all processes ...
5: **end-main**

FT

- 3-D Fast-Fourier transform kernel
  - Used to numerically solve partial differential equation (PDE)
- All-to-all communication used to exchange the transpose results
- Class B used in the evaluation

FT

Algorithm 2 FT

```
1: main
2:     for t from 1 to number of iterations do
3:         ... evolve u0 to u1 (t time steps) in fourier space ...
4:         ... calls the fft subroutine ...
5:         ... transpose operations in each process ...
6:         ... use MPI_Alltoall to exchange the transpose results ...
7:         ... call checksum ...
8:     end-for
9: end-main
```

SP

- Scalar Penta-Diagonal solver
  - Solves multiple, independent systems of nondiagonally-dominant, scalar pentadiagonal equations
- Coarse grained communication
- Class B used in the evaluation

SP

Algorithm 3 SP

```
1: main
2:     for t from 1 to number of iterations do
3:       ... performs the block-diagonal matrix vector multiplicator ...
4:       ... use MPI_Isend to send the buffer ...
5:       ... use MPI_Ireceive to receive the buffer ...
6:       ... performs aproximate factorization in the x-plane ...
7:       ... use MPI_Isend to send the buffer ...
8:       ... use MPI_Ireceive to receive the buffer ...
9:       ... performs aproximate factorization in the y-plane ...
10:      ... use MPI_Isend to send the buffer ...
11:      ... use MPI_Ireceive to receive the buffer ...
12:      ... performs aproximate factorization in the z-plane ...
13:      ... use MPI_Isend to send the buffer ...
14:      ... use MPI_Ireceive to receive the buffer ...
15:      ... add the u vector ...
16:    end-for
```

HIS

- Three dimensional simulator of the Human Immune System
  - Set of eight Partial Differential Equations (PDEs) used to describe how some cells and molecules involved in the innate immune response react to a pathogen.
- $200 \times 200 \times 200$
- Border exchange occurs at the end of each time iteration

HIS

Algorithm 4 HIS

1: **main**
2: ... define the mesh slice to be computed by each GPU/CPU ...
3: ... initialize submeshes according to their initial conditions ...
4: **for** t **from** 0 **to** final time **do**
5: ... call the functions/*kernels* in order to compute the PDEs ...
6: ... use MPI_Isend and MPI_Receive to exchange boundaries between distinct machines ...
7: ... synchronize all machines ...
8: **end-for**
9: **end-main**

EP

- The EP benchmark is modeled using the following equation:

$$T_{total} = \frac{size}{R_p} + I \times N_{op} \times \log_2 P \times (L_d + \frac{M}{B_d} + o_d), \quad (1)$$

- $size$ is the size of the problem
- R_p is the relative computing power
- I is the number of iterations
- N_{op} is the number of communication operations per iteration
- L_d is the latency
- o_d is the overhead
- B_d represents the bandwidth
- P is the number of processors used in the experiments and
- M is the message size

FT and SP

- The FT benchmark is modeled using the following equation:

$$T_{total} = I \times (R_p + N_{op} \times (P-1) \times (L_d + \frac{M}{B_d} + o_d)) \quad (2)$$

- The SP benchmark is modeled using the following equation:

$$T_{total} = I \times (R_p + N_{op} \times (L_d + \frac{M}{B_d} + o_d)) \quad (3)$$

HIS

- The HIS benchmark is modeled using the following equation:

$$T_{total} = I \times (R_p + T_{ij}), \tag{4}$$

where

$$T_{ij} = (L_d + \frac{M}{B_d} + o_d) \tag{5}$$

Experimental environment

- 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

Results

Table: Results for the EP, FT and HIS using 2 AMD processors. All times are in seconds.

	Ethernet			Infiniband		
	Real	**Estimated**	**Error**	**Real**	**Estimated**	**Error**
EP	295.6	295.8	0.1%	297.2	295.8	0.5%
FT	95.0	96.3	1.5%	66.1	69.4	5.0%
HIS	213.4	219.1	2.7%	102.7	109	6.1%

- SP code requires a square number of processors

Results

Table: Results for the EP, FT and SP kernel using both Intel and AMD processors (half of each), Ethernet network. All times in seconds.

	4 Nodes			8(9) Nodes*			16 Nodes		
	Real	**Estimated**	**Error**	**Real**	**Estimated**	**Error**	**Real**	**Estimated**	**Error**
EP	118.0	110.5	6.4%	52.0	55.2	6.3%	28.6	28.6	0.0%
FT	71.4	72.0	0.9%	67.0	68.1	1.8%	65.8	64.1	2.7%
SP	442.3	445.7	0.6%	265.9	267.7	1.0%	343.7	345.4	0.5%

- *For SP, we used 9 nodes (4 AMDs and 5 Intels) since the code requires a square number of processors

Results

Table: Results for HIS using both GPUs and CPUs. All times in seconds.

	Real	Estimated	Error
1	47.2	42.5	10.0%
2	59.8	54.2	9.2%
3	107.8	95.0	12.0%

- Configuration number 1: 2 CPUs (1 AMD and 1 Intel) and 2 GPUs (M2075 and C1060)
- Configuration number 2: 4 CPUs (2 AMDs and 2 Intels) and 4 GPUs (2 M2075 and 2 C1060)
- Configuration number 3: 7 CPUs (5 AMDs and 2 Intels) and 7 GPUs (3 M2075, 2 M2050 and 2 C1060)

Conclusion

- New model that generalizes the LogP model in order to deal with heterogeneous parallel environments
- Model 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 were below 6.4% in all experiments
  - Except for the HIS simulator, where the error was about 12% when distinct CPUs and GPUs were used in the simulation

Future works

- Better understand the causes of the higher error found in HIS
- 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

Thank you!

The authors would like to thank UFJF, FAPEMIG, CAPES, and CNPq