Energy-aware scheduling in heterogeneous computing systems

Santiago Iturriaga

TESIS DE MAESTRÍA
EN INFORMÁTICA

PEDECIBA INFORMÁTICA
Instituto de Computación, Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay

Outline

Introduction

Heterogeneous Computing (HC) systems

Distributed computing systems usually comprise a large number of heterogeneous computing resources which are able to work cooperatively, e.g. cluster or grid computing systems.

Scheduling problem in computing systems

- Consists in efficiently assigning tasks to computing resources.
- Several criteria: execution time, quality of service, energy consumption, reliability, etc.
- Key problem in order to fully take advantage of the available computing capabilities.

Introduction

Motivation

- Energy consumption in HC systems has increased considerably in the last decade.
- Large HC systems providers are concerned by energy consumption.
- CPU consume up to 50%-60% of the total energy consumption of a computational system.

Energy-aware Heterogeneous Computing Scheduling Problem (ME-HCSP)

- Goals: simultaneously minimize the **schedule length** (i.e. *makespan*) and the **energy consumption** of the system.
- The heterogeneity and the non-preemptive characteristics of the system increases the complexity of the scheduling problem.
- NP-hard problem, hence usually heuristics or metaheuristics are considered for solving it.

Introduction

Contributions

- First, this work presents the **ME-MLS** algorithm: a fast Multithreading Local Search (MLS) method to efficiently solve the ME-HCSP in reduced execution time.
- Instances comprised of up to 2048 tasks and 64 machines were tackled using ME-MLS.
- Ongoing work is being done for tackling larger problem instances.
- Regarding this line of ongoing work, it is presented the **gPALS** algorithm: a CPU/GPU hybrid algorithm for tackling the HCSP which considers makespan as it sole objective function.
- Instances with up to 32768 tasks and 1024 machines were tackled using gPALS.

Energy-aware Heterogeneous Computing Scheduling Problem (ME-HCSP)

Problem formulation

- An HC system is composed of a set of heterogeneous machines $P = \{m_1, \ldots, m_M\}$.
- A collection of heterogeneous tasks $T = \{t_1, \ldots, t_N\}$ to be executed.
- An *execution time function* $ET : T \times P \rightarrow \mathbf{R}^+$, where $ET(t_i, m_j)$ is the execution time of task t_i on machine m_j.
- An *energy consumption function* $EC : T \times P \rightarrow \mathbf{R}^+$, where $EC(t_i, m_j)$ is the energy required to execute task t_i on machine m_j.
- An *idle energy consumption function* $EC_{idle} : P \rightarrow \mathbf{R}^+$, being $EC_{idle}(m_j)$ the energy that machine m_j consumes per time unit when it is in idle state.
- A scheduling function $f : T \rightarrow P$, which states that task t_i is to be executed by machine m_j only if $f(t_i) = m_j$.

Energy-aware Heterogeneous Computing Scheduling Problem (ME-HCSP)

Problem goal

The ME-HCSP aims at finding the scheduling function f that simultaneously minimizes the *makespan* (C_{max}) and the *total energy consumption* (E).

$$C_{max} = \max_{m_j \in P} C_j \ , \ with \ C_j = \sum_{\substack{t_i \in T: \\ f(t_i)=m_j}} ET(t_i, m_j)$$

$$E = consumption \ in \ working \ state + consumption \ in \ idle \ state$$

$$= \sum_{t_i \in T} EC(t_i, f(t_i)) + \left\{ \sum_{m_j \in P} (C_{max} - C_j) \times EC_{idle}(m_j) \right\}$$

Energy-aware Heterogeneous Computing Scheduling Problem (ME-HCSP)

Problem characteristics

- *Offline*: assumes complete knowledge of the tasks to be scheduled.
- *Clairvoyant*: the complete job characteristics are available as inputs to the scheduling algorithm.
- *Independent tasks*: no dependency constraints between tasks.
- *Non-preemptive*: every task is atomic, and cannot be interrupted once it begins its execution.
- *Unrelated machines model*: no relationship between the execution time of a task and its executing machine.
- *Reduced scheduling time is mandatory*: a very important feature of every scheduler in practice.

Metaheuristic algorithms

What is a metaheuristic algorithm?

A metaheuristic algorithm is an iterative top-level stochastic search technique which guides a subordinate heuristic, exploring and exploiting the problem search space, in order to find near-optimal solutions to hard problems.

Local Search (LS) algorithm

Local search algorithms are metaheuristic methods which work by improving one or more candidate solutions by exploring a reduced neighborhood ($\mathcal{N}$) of *nearby* solutions.

Metaheuristic algorithms

Multi-objective optimization problems (MOP)

Optimization problems have two or more conflicting objectives that are to be optimized simultaneously.

$$\text{optimize } F(x) = (f_1(x), ..., f_n(x))$$
$$\text{with } x \in \mathcal{S}$$

In the most general case all objectives are equally important. The solution to a MOP is a set of trade-off solutions which define a *Pareto front* (PF).

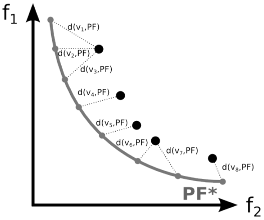
Goals when solving a MOP

- Compute accurate solutions for every objective function.
- Maintain diversity in the computed solutions.

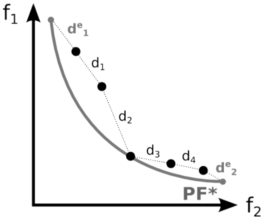
Metaheuristic algorithms

MOP performance metrics

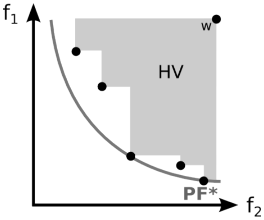
- Evaluating the results computed by a method for solving a MOP is not an easy task.
- Several performance metrics have been proposed.
- Metrics can be classified according to whether they are convergence-based metrics, diversity-based metrics, or hybrid metrics.



IGD metric
(convergence-based)



Spread metric
(diversity-based)



Hypervolume metric
(hybrid)

Previous approaches for solving the ME-HCSP

Single- vs multi-objective approaches

- The energy-aware HCSP is a multi-objective optimization problem.
- Most related work simplifies the problem solving a Single-objective Optimization Problem (SOP) (19 out of 23 of the related works).
- Solving a MOP demands significantly more computational effort than solving a SOP.

Dimension of the offline problem instances tackled

- Well-known classic offline HCSP instances scenarios with **512 tasks** and **16 machines**.
- Most of the related work tackles instances with less than **1000 tasks** and **32 machines** (even the ones using a SOP approach).
- The maximum instance dimension tackled in the related work is comprised of up 4096 tasks and 256 machines (Kołodziej et al., 2011).

ME-MLS: a MO metaheuristic for the ME-HCSP

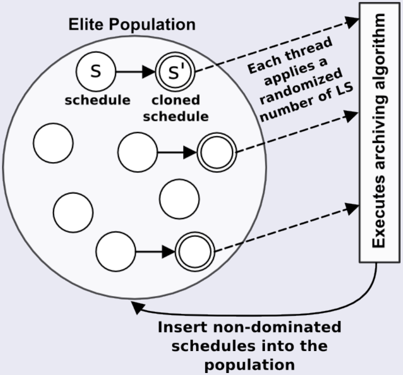
Characteristics

- Size-bounded population of candidate solutions.
- Fully multi-objective Pareto-based approach.
- Makes use of two subordinate heuristics:
  - A randomized MCT to initialize its population.
  - A randomized PALS as its local search method.
- Parallel algorithm design
  - Multiple concurrent LS are applied to schedules in the population in order to improve them.
  - There is no hierarchy, all concurrent LS are peers.
- Shared-memory multithreading implementation using C and POSIX threads.

ME-MLS: a MO metaheuristic for the ME-HCSP

Logic of each thread in the ME-MLS algorithm

1. Randomly select a schedule from the population.
2. The selected schedule is cloned.
3. A randomized number of LS are applied to the cloned schedule.
4. Try to lock the population.
    - If the population is successfully locked then executes the archiving algorithm.
    - Else return to step 3.
5. Return to first step (or end the algorithm).



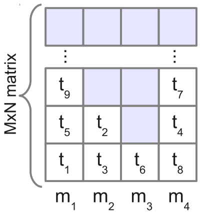
ME-MLS: a MO metaheuristic for the ME-HCSP

Population initialization

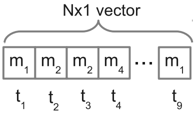
- Sensitive issue in the design of a metaheuristic algorithm.
- A randomized version of the well-known MCT heuristic is proposed.
- The randomized MCT (rMCT) has a complexity order of $O(n^2)$.

In-memory schedule encoding

- Two well-known structures are proposed in the related literature.
- ME-MLS uses a multi-structure comprising both encodings.



Machine-oriented encoding



Task-oriented encoding

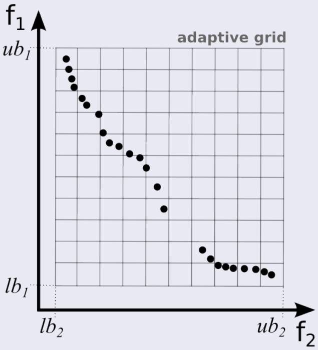
ME-MLS: a MO metaheuristic for the ME-HCSP

Fast Greedy Ad-hoc Archiving (FGAA) algorithm

- Simple method with strong emphasis on computational efficiency.
- Not especially conceived to maintain high diversity in the population.

Adaptive Grid Archiving (AGA) by Knowles and Corne (2000)

- Divides the objective space into hypercubes (multi-dimensional grid).
- Solutions are discarded according to how crowded their hypercube is.
- Guarantees three desirable properties:
    - Maintains solutions at the extremes
    - Maintains solutions in all of the Pareto occupied regions
    - Distributes the remaining solutions evenly among the Pareto regions.



ME-MLS: a MO metaheuristic for the ME-HCSP

Local search algorithm

- PALS is an efficient local search proposed by Alba and Luque (2007).
  - Efficient δ-function representing a relative improvement estimation.
- Nesmachnow et al. (2012b) proposed a randomized PALS (rPALS) for the HCSP.
  - Random-sized neighborhood structures.
  - Multiple search neighborhood structures (*move* and *swap*).
- A variation of rPALS for the ME-HCSP (ME-rPALS) is proposed.
  - Considers multiple target metrics in order to tackle both objective functions (*makespan* and *energy consumption*).
- ME-rPALS is able to efficiently find local optima when searching large solution spaces.

ME-MLS: a MO metaheuristic for the ME-HCSP

Logic of the ME-rPALS algorithm

1. A target metric is randomly selected (*makespan* or *energy consumption*).
2. Randomly selects a machine *m* to perform the search.
   - With high probability, the machine with the worst local metric is selected.
3. Then, a random number of tasks currently assigned to be executed by the machine *m* are selected.
4. Some changes are evaluated on the selected tasks. These changes include *moving* some tasks to other machines, and *swapping* some tasks with other tasks.
5. Finally, the algorithm applies the *move* or *swap* which improves the most the current schedule.

Experimental analysis of ME-MLS

Experimental platform

- All the experiments were performed in a 24-core server with AMD Opteron 6172 processors at 2.1GHz and 24 GB of RAM.
- Hosted as part of the HPC facility, Cluster FING, of the Facultad de Ingeniería, Universidad de la República.

Test instances

- A total of **792** instances where generated for the ME-HCSP.
- Dimensions (#tasks×#machines): 512×16, 1024×32, and 2048×64.
- The task workloads were generated following the ETC model by Ali et al. (2000), and the machine scenarios using the methodology proposed by Nesmachnow et al. (2012a).

Experimental analysis of ME-MLS

Lower bounds computed using a Linear Programming (LP) relaxation

- Lower Bounds (LB) from a preemptive ME-HCSP relaxation.
- Two LP relaxations: makespan and energy consumption.
- The relative integrality gap (*rgap*) is reported.

MinMin-based list-scheduling heuristics

- MinMin is a well-known accurate heuristic for the HCSP.
- Works in two-phases.
- Four versions are generated by alternating the minimization objective
  - **MinMin**: makespan on both phases.
  - **MINMIN**: energy on both phases.
  - **MinMIN**: makespan on the first phase and energy on the second.
  - **MINMin**: energy on the first phase and makespan on the second.

Experimental analysis of ME-MLS

Parameter configuration analysis

- Performed on a subset of four of the small instances.
- ME-MLS best configuration: 24 threads, population of 34 individuals.
- Stopping criterion of 10 s. *Significantly shorter* than the execution time of the metaheuristics in the related work (40-90 s.).

Pseudo-Random Number Generator (PRNG) analysis

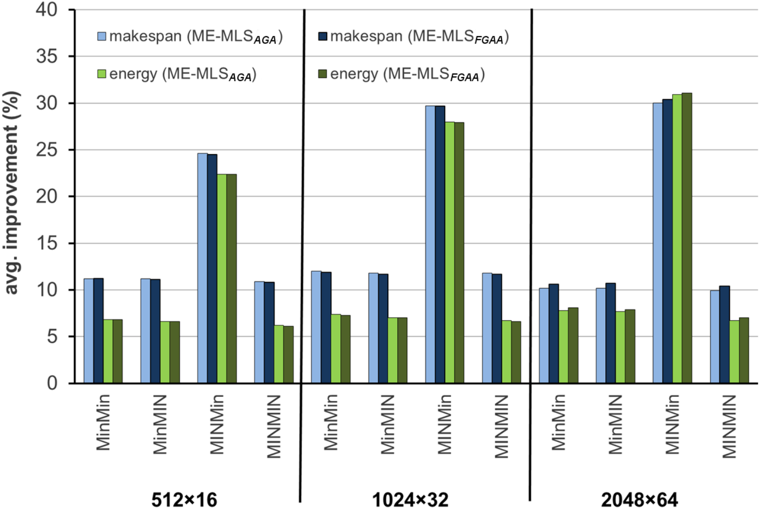
- Significant time-consuming function in metaheuristic methods.
- Three PRNG were evaluated for the ME-MLS: `rand_r`, `drand48_r`, Mersenne Twister (MT).
- ME-MLS using MT was able to execute **3.5**$\times$ faster than `drand48_r`, and **9.5**$\times$ faster than `rand_r`.
- Henceforth, the MT function is used as PRNG.

Experimental analysis of ME-MLS

Average improvement over the best MinMin-based heuristic after 30 independent executions:

dim.	cons.	makespan				energy			
		average		*rgap*		*average*		*rgap*	
		AGA	*FGAA*	*AGA*	*FGAA*	*AGA*	*FGAA*	*AGA*	*FGAA*
512×16	cons.	7.2%	7.1%	4.1%	4.2%	4.8%	4.7%	4.6%	4.6%
	semi.	10.8%	10.7%	4.4%	4.6%	6.1%	6.0%	4.1%	4.1%
	incons.	12.9%	12.9%	4.0%	4.1%	6.8%	6.7%	3.7%	3.7%
	avg.	**10.3%**	**10.2%**	**4.2%**	**4.3%**	**5.9%**	**5.8%**	**4.1%**	**4.2%**
1024×32	cons.	6.8%	6.8%	4.9%	5.0%	6.6%	6.7%	5.3%	5.2%
	semi.	10.9%	10.8%	7.7%	7.8%	4.6%	4.3%	6.2%	6.5%
	incons.	15.9%	15.8%	6.6%	6.7%	8.0%	7.9%	5.9%	6.0%
	avg.	**11.2%**	**11.1%**	**6.4%**	**6.5%**	**6.4%**	**6.3%**	**5.8%**	**5.9%**
2048×64	cons.	4.1%	5.2%	7.7%	6.4%	6.9%	8.0%	7.4%	6.1%
	semi.	7.9%	8.3%	12.0%	11.4%	4.8%	4.7%	8.4%	8.5%
	incons.	16.8%	16.7%	10.9%	11.1%	8.0%	7.8%	8.5%	8.7%
	avg.	**9.6%**	**10.1%**	**10.2%**	**10.1%**	**6.6%**	**6.8%**	**8.1%**	**7.8%**

Average improvement over each of the MinMin-based heuristic:



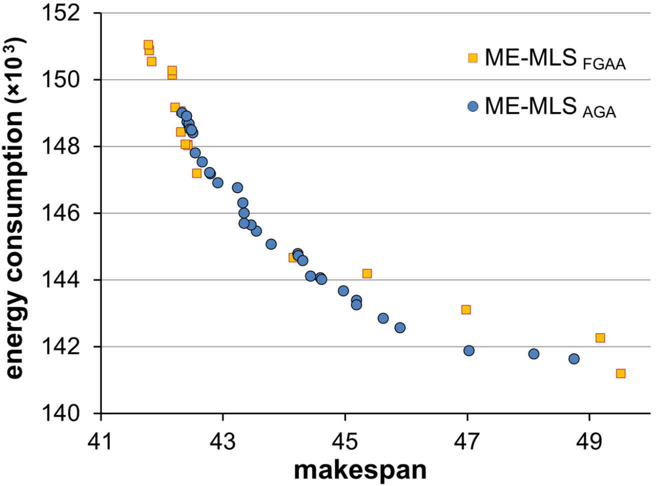
Experimental analysis of ME-MLS

Multi-objective optimization metrics for the Pareto front approximations computed by each algorithm (for the 264 instances each dimension):

dimension	**ND**			**IGD** (normalized)		
	AGA	*FGAA*	best$_{AGA/FGAA}$	*AGA*	*FGAA*	best$_{AGA/FGAA}$
512×16	**4.70±2.23**	3.10±0.87	**110**/16	**1.00±0.08**	1.20±0.13	**67**/24
1024×32	**6.70±3.20**	3.57±0.92	**173**/5	**1.00±0.26**	1.62±0.21	**150**/18
2048×64	**7.35±2.90**	3.95±0.99	**193**/9	**1.00±0.27**	1.71±0.27	**173**/19

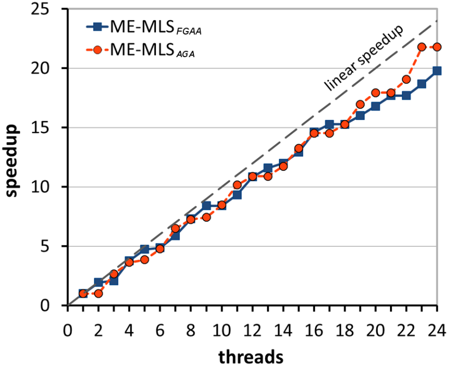
dimension	**Spread** (normalized)			**RHV**		
	AGA	*FGAA*	best$_{AGA/FGAA}$	*AGA*	*FGAA*	best$_{AGA/FGAA}$
512×16	**1.00±0.09**	1.04±0.06	**65**/30	**0.83±0.05**	0.81±0.05	**55**/21
1024×32	**1.00±0.10**	1.13±0.09	**124**/35	**0.82±0.06**	0.76±0.05	**130**/29
2048×64	**1.00±0.09**	1.25±0.16	**148**/30	**0.81±0.05**	0.75±0.06	**146**/36

Sample Pareto front computed after 30 executions when solving an instance with dimension 1024x32 with a stopping criterion of 10 seconds:



Speedup analysis

- Performed using 1024×32 instances with a stopping criterion of 6 million iterations.
- The analysis shows a speedup of 22.4 for ME-MLS$_{AGA}$ and 20.2 for ME-MLS$_{FGAA}$, both using 24 threads.



Scheduling very large HCSP scenarios with gPALS

Motivation

- Heterogeneous computing systems have not stopped growing in size.
- Profit from the computing power of the massively-parallel GPU device.
- Profit from the proven efficacy and efficiency of the PALS-based LS.

Goal

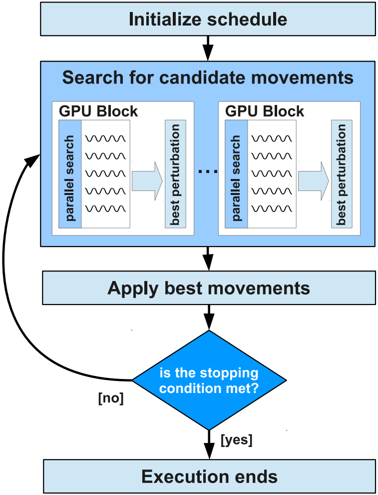
- Tackle very large instances in a few seconds.
- Design an efficient PALS-based local search for the GPU.
- Solve the HCSP, considering the schedule length as objective function.

gPALS design

- gPALS is a hybrid CPU/GPU rPALS-based local search.
- The search is guided by a high-level schema executed on the CPU.
- The neighborhood evaluation is massively performed on the GPU.

gPALS high-level schema

1: $s \leftarrow$ GenerateInitialSolution()
2: **while** stopping condition not met **do**
3: $M \leftarrow$ Massively parallel neighborhood evaluation applied to s in the GPU
4: $s \leftarrow$ Apply the best movement from M
5: $s \leftarrow$ Apply the rest of the movements in M in random order
6: **end while**
7: **return** s



Scheduling very large HCSP scenarios with gPALS

In-memory schedule encoding

- The *task-oriented encoding* is used.
- The *machine-oriented encoding* is discarded because memory footprint concerns.

Initialization heuristics

- Two heuristics were evaluated: MCT and pMinMin/DD.
- pMinMin/DD by Canabé and Nesmachnow (2012) is a parallel MinMin heuristic with domain decomposition.
- It improves the efficiency of the MinMin heuristic sacrificing efficacy.
- MinMin has a $O(n^3)$ execution order, while pMinMin/DD using p threads has an execution order of $O\left(\frac{n^3}{p^2}\right)$.

Experimental analysis of gPALS

Experimental platform

- Bull B505 server with a 12-core Intel Xeon CPU L5640 at 2.27GHz, 24 GB RAM, and a NVIDIA Tesla M2090 GPU device.
- Hosted in the HPC facility, Gaia, of the University of Luxembourg.
- Mersenne Twister for Graphic Processors (MTGP) library is used.

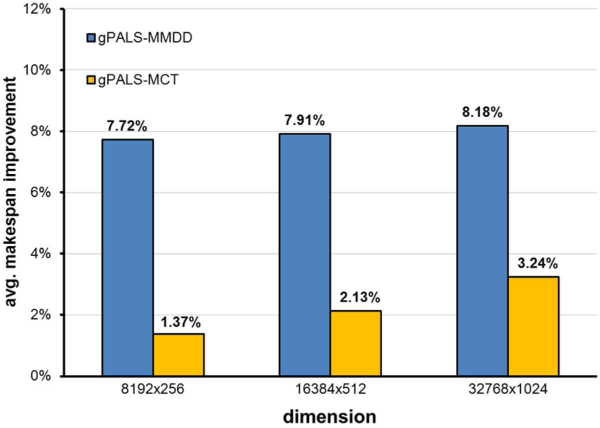
Test instances

- Generated using the ETC model by Ali et al. (2000).
- High heterogeneity in both task and machines.
- With dimensions: 8192×256, 16384×512, and 32768×1024.
- 60 different problem instances (20 for each dimension).

Experimental analysis of gPALS

Numerical efficiency analysis

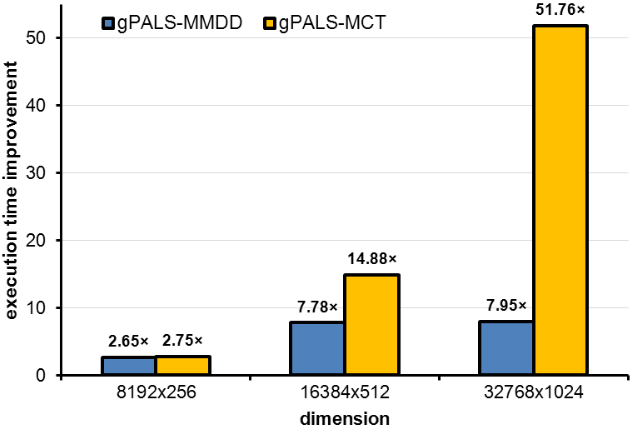
- Stopping criterion: 30 seconds PALS execution time.
- Method for baseline comparison: MinMin.



Experimental analysis of gPALS

Parallel performance analysis

- Stopping criterion: fixed solution quality.
- Method for baseline comparison: MinMin.



Comparison against the cEA by Pinel et al. (2013)

Comparison of the average makespan improvement over MinMin:

dimension	**avg. makespan improvement** (over MinMin)		
	gPALS$_{MCT}$	gPALS$_{MMDD}$	cellular EA
8192×256	1.37%±0.83%	**7.72%±0.73%**	6.75%±0.73%
16384×512	2.13%±0.53%	**7.91%±0.49%**	6.05%±0.57%
32768×1024	3.24%±0.30%	**8.18%±0.32%**	5.19%±0.34%

Average execution time comparison:

dimension	**avg. execution time** (s)		
	gPALS$_{MCT}$	gPALS$_{MMDD}$	cellular EA
8192×256	**38.7±0.6**	39.3±0.6	1630.3±5.6
16384×512	**40.7±0.6**	47.4±0.6	4382.3±16.4
32768×1024	**49.6±0.7**	136.3±7.1	8088.3±58.3

Conclusions and future work

Summary and contributions

- The scheduling problem in HC environments was studied.
  - The schedule length time and energy consumption were considered.
  - Efficient schedulers were designed.
- Two LP relaxation formulation and four Min-Min based heuristics were presented, considering both ME-HCSP objectives.
- Proposed ME-MLS: a multithreading local search method to efficiently solve the ME-HCSP in reduced execution time.
  - Two different archiving algorithms were compared.
- Proposed gPALS: a massively parallel CPU/GPU hybrid local search for tackling very large HCSP instances.
  - Two different initialization algorithms were compared.

Conclusions and future work

Experimental analysis conclusions

- Both presented algorithms are able to compute accurate schedules in reduced execution times.
- Regarding ME-MLS experimental analysis:
  - Average improvements of **10%** for the makespan and **6%** for the energy consumption in **10 s.** execution time for the ME-HCSP.
  - ME-MLS$_{AGA}$ offers a better overall performance than ME-MLS$_{FGAA}$.
  - Instances of up to 2048×64 where tackled.
- Regarding gPALS experimental analysis:
  - Achieved average makespan improvements of up to **8%** over the MinMin heuristic in **30 s.** execution time.
  - Average acceleration of up to **51**× when compared to MinMin.
  - Faster and more accurate than cEA by Pinel et al. (2013).
  - Instances of up to 32768×1024 where tackled.

Conclusions and future work

Future work

- Improve the efficacy and efficiency of the proposed algorithms.
- Regarding the ME-MLS:
  - Improve the Pareto-front analysis comparing the results with the ones computed by well-known MOEA.
  - Integrate the proposed algorithm into well-known MOEA.
- Regarding the gPALS:
  - Parallelization based on domain decomposition in GPUs.
  - Design ME-gMLS: a CPU/GPU local search algorithm for the ME-HCSP.
- Tackle an online version of the ME-HCSP considering multi-core computing resources.

Thanks for your attention

Brief summary of the publications issued from this thesis' work:

- S. Iturriaga, S. Nesmachnow, and B. Dorronsoro. A Multithreading Local Search For Multiobjective Energy-Aware Scheduling In Heterogeneous Computing Systems. In *Proceedings of the 26th European Conference on Modelling and Simulation (ECMS)*, pages 497–503, Koblenz, Germany, 2012a. ISBN 978-0-9564944-4-3
- S. Iturriaga, S. Nesmachnow, F. Luna, and E. Alba. A parallel online GPU scheduler for large heterogeneous computing systems. In *Proceedings of the 5th High-Performance Computing Latin America Symposium (HPCLatAm)*, JAIIO '12, Buenos Aires, Argentina, 2012b
- S. Iturriaga, S. Nesmachnow, and C. Tutté. Metaheuristics for multiobjective energy-aware scheduling in heterogeneous computing systems. In *EU/Metaheuristics Meeting Workshop (EU/ME)*, Copenhaguen, Denmark, 2012c
- S. Iturriaga, S. Nesmachnow, B. Dorronsoro, and P. Bouvry. Energy efficient scheduling in heterogeneous systems with a parallel multiobjective local search. *Computing and Informatics Journal (CAI)*, 2013a. Accepted on November 2012, to appear
- S. Iturriaga, S. Nesmachnow, F. Luna, and E. Alba. A parallel local search in CPU/GPU for scheduling independent tasks on large heterogeneous computing systems. *Journal of Parallel and Distributed Computing (JPDC)*, 2013b. Submitted on January 2013, pending acceptance
- S. Iturriaga, P. Ruiz, S. Nesmachnow, B. Dorronsoro, and P. Bouvry. A Parallel Multi-objective Local Search for AEDB Protocol Tuning. In *Proceedings of the 16th International Workshop on Nature Inspired Distributed Computing*, in the 27th IEEE/ACM International Parallel & Distributed Processing Symposium, Boston, Massachusetts, USA, 2013c. Accepted on February 2013, to appear