

Um Algoritmo de Escalonamento para Redução do Consumo de Energia em Computação em Nuvem

Pedro Paulo Vezz  Campos pedrovc@ime.usp.br

Orientador: Prof. Dr. Daniel Mac do Batista batista@ime.usp.br

Departamento de Ci ncia da Computa  o — Instituto de Matem tica e Estat stica — Universidade de S o Paulo

Introdu  o

TI   respons vel por aproximadamente 2% das emiss es anuais de CO_2 , pr ximo do n vel gerado pela avia  o [Gar07]. Ao mesmo tempo, a **Lei de Moore**, que profetiza que o poder computacional de dispositivos dobra a cada 18 meses, est  chegando ao fim da sua vida [PH12]. Processadores modernos atingiram uma barreira de pot ncia mas no entanto n o eram eficientes no **consumo energ tico** [BH07]. Assim, novas tend ncias surgiram na ind stria: processadores mais simples, mais paralelos e mais eficientes.

Computa  o em nuvem surgiu como uma consequ ncia quase natural destas tend ncias. Ao consolidar poder de processamento, transfer ncia de dados e armazenamento   poss vel reduzir custos e desperd cios. Algumas estrat gias poss veis: **consolida  o** de m quinas virtuais, dimensionamento de tens o e frequ ncia (**DVFS**) e **algoritmos energeticamente eficientes**.

Aplica  es de processamento paralelo podem ser modeladas como **digrafos ac clicos** (DAGs). O problema de decidir qual o melhor **escalonamento** de uma tarefa (v rtice do DAG) em uma m quina de forma a otimizar o uso de algum recurso   um problema **NP-dif cil**. Assim, heur sticas s o necess rias para encontrar solu  es aproximadas.

Resultados

Este TCC apresenta um **novo algoritmo** de escalonamento de fluxos de trabalho em computa  o em nuvem voltado para a efici ncia energ tica. O desempenho foi comparado com o trabalho “*Energy-aware simulation with DVFS*” [GMDC+13] e com um algoritmo de escalonamento cl ssico mas sem um foco na efici ncia energ tica. Os estudos contaram com a contribui  o da aluna de mestrado **Elaine Naomi Watanabe** (elainew@ime.usp.br).

Como resultados secund rios, foram feitas **contribui  es** a projetos de **software livre** na forma de divulga  o de c digo fonte, incrementos e notifica  es de falhas nos simuladores de computa  o em nuvem utilizados pelo autor.

Algoritmo Proposto

ESCALONARPOWERHEFT(*tarefa*, *VM*)

- 1 F = filhos diretos da *tarefa* no DAG
- 2 Escalone *tarefa* em *VM*
- 3 Escalone F utilizando o algoritmo HEFT
- 4 // A modelagem energ tica utilizada est  descrita em [GMDC+13]
- 5 $energia = ESTIMARENERGIACONSUMIDA()$
- 6 Volte para o escalonamento do come o do la o
- 7 **retorne** *energia*

POWERHEFTLOOKAHEAD()

- 1 $V = \{VmMaisR pida\}$ // VMs usadas ao escalonar
- 2 O = os tipos de VMs que podem ser instanciadas
- 3 Ordene o conjunto de tarefas segundo o crit rio $rank_u$
- 4 **enquanto** h  tarefas n o escalonadas
- 5 t = a tarefa n o escalonada de maior $rank_u$
- 6 // Vamos tentar escalonar t em uma VM existente
- 7 **para** cada v em V :
- 8 ESCALONARPOWERHEFT(t, v)
- 9 // Vamos tentar escalonar t em uma nova VM
- 10 **para** cada o em O :
- 11 $V = V \cup \{o\}$
- 12 Atualize os valores de $rank_u$
- 13 t = a tarefa n o escalonada de maior $rank_u$
- 14 ESCALONARPOWERHEFT(t, o)
- 15 Escalone t na VM que minimiza a energia consumida
- 16 Atualize V e $rank_u$ caso necess rio

Escalonamento de fluxos de trabalho com computa  o em nuvem

O *Heterogeneous Earliest Finish Time* (**HEFT**) [THW02]   uma boa heur stica para o problema de escalonamento. Ele recebe como par metros um DAG a ser escalonado, um conjunto possivelmente heterog neo de m quinas que realizar o o processamento, os tempos de processamento de cada tarefa em cada m quina e o tempo de transmiss o entre duas tarefas. Ele   dividido em duas fases: **prioriza  o** e **sele  o**.

Prioriza  o

Qual tarefa escalonar primeiro? A f rmula abaixo   o crit rio de prioriza  o do HEFT. Al m de gerar uma **ordem topol gica**, d  prioridade a tarefas mais cr ticas do DAG.

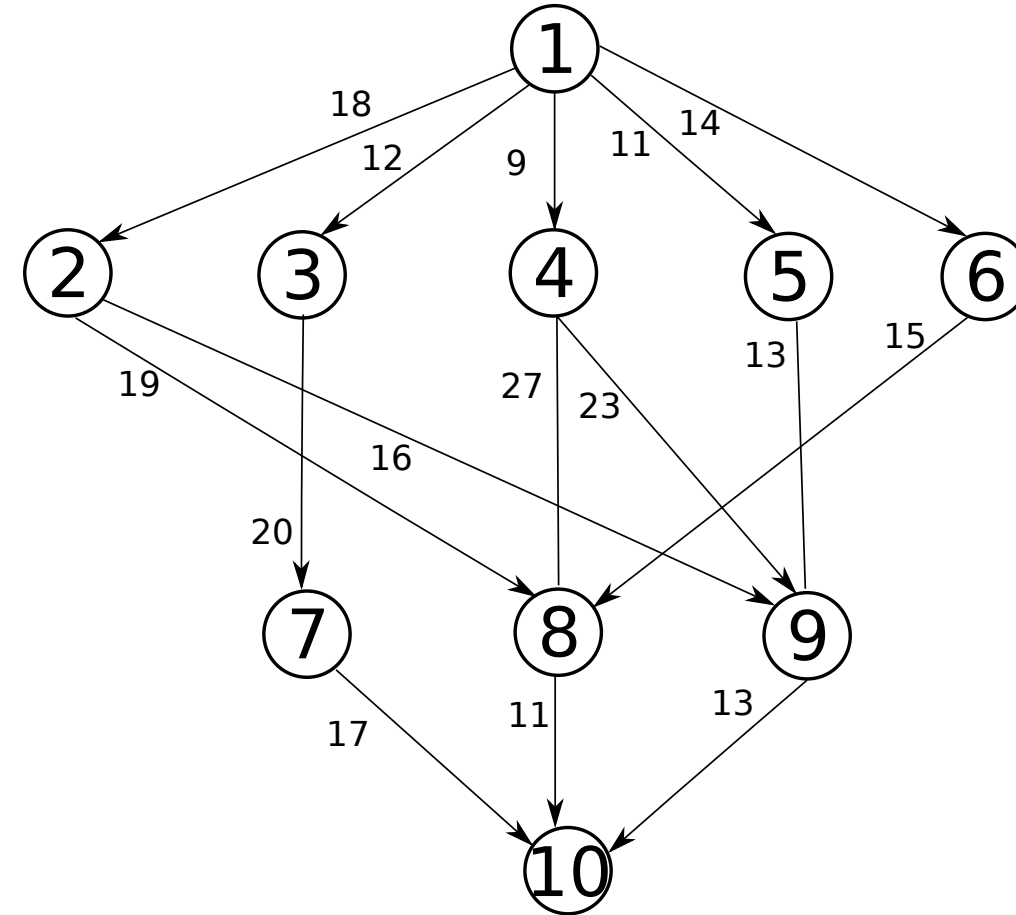
$$rank_u(n_i) = w_i + \max_{n_j \in succ(n_i)} (c_{i,j} + rank_u(n_j))$$

Sele  o

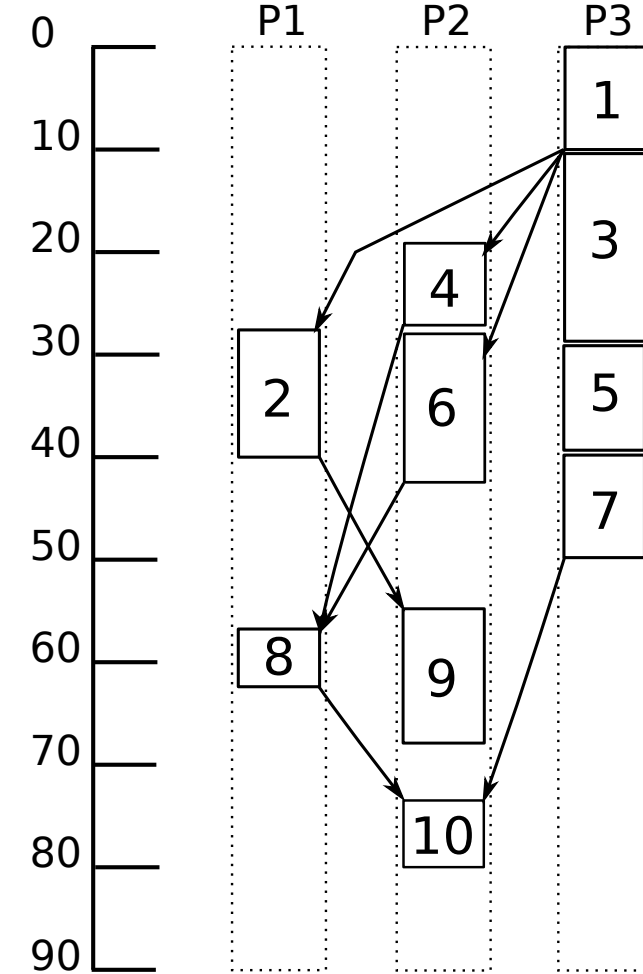
Em qual m quina escalonar uma tarefa? O HEFT tenta minimizar o **tempo mais cedo de conclus o** de cada tarefa na esperan a que isso minimize a conclus o do fluxo de trabalho como um todo.

Exemplo

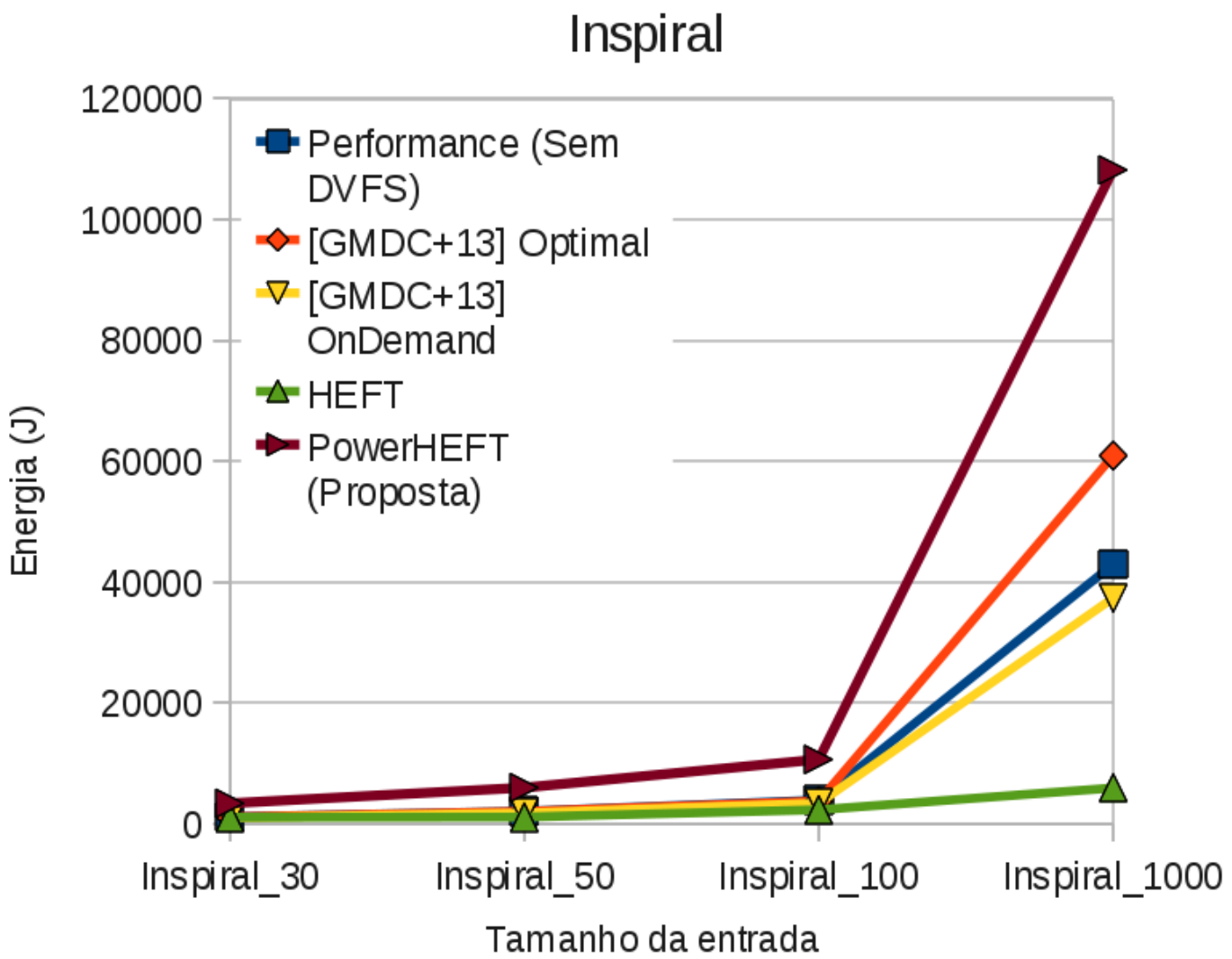
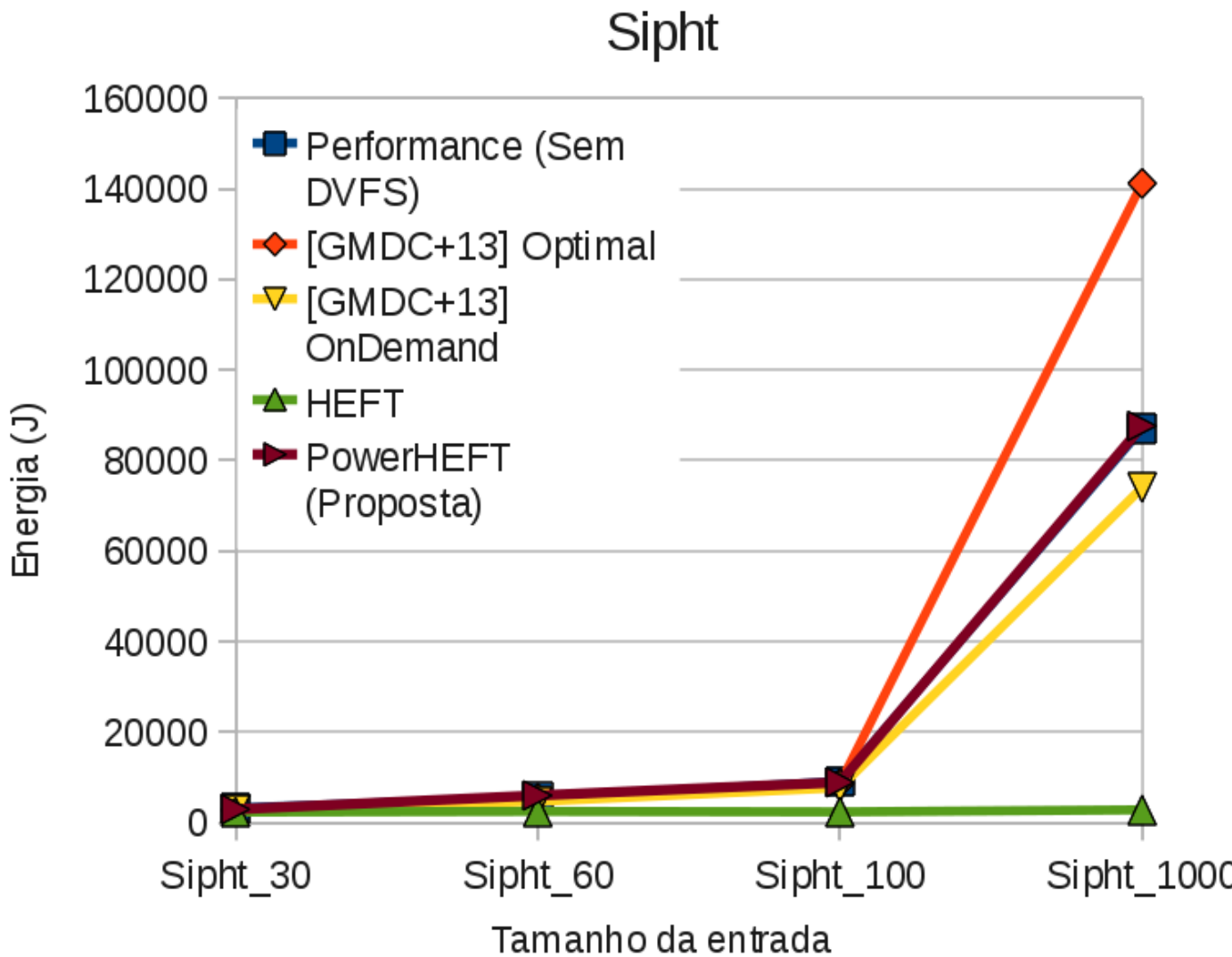
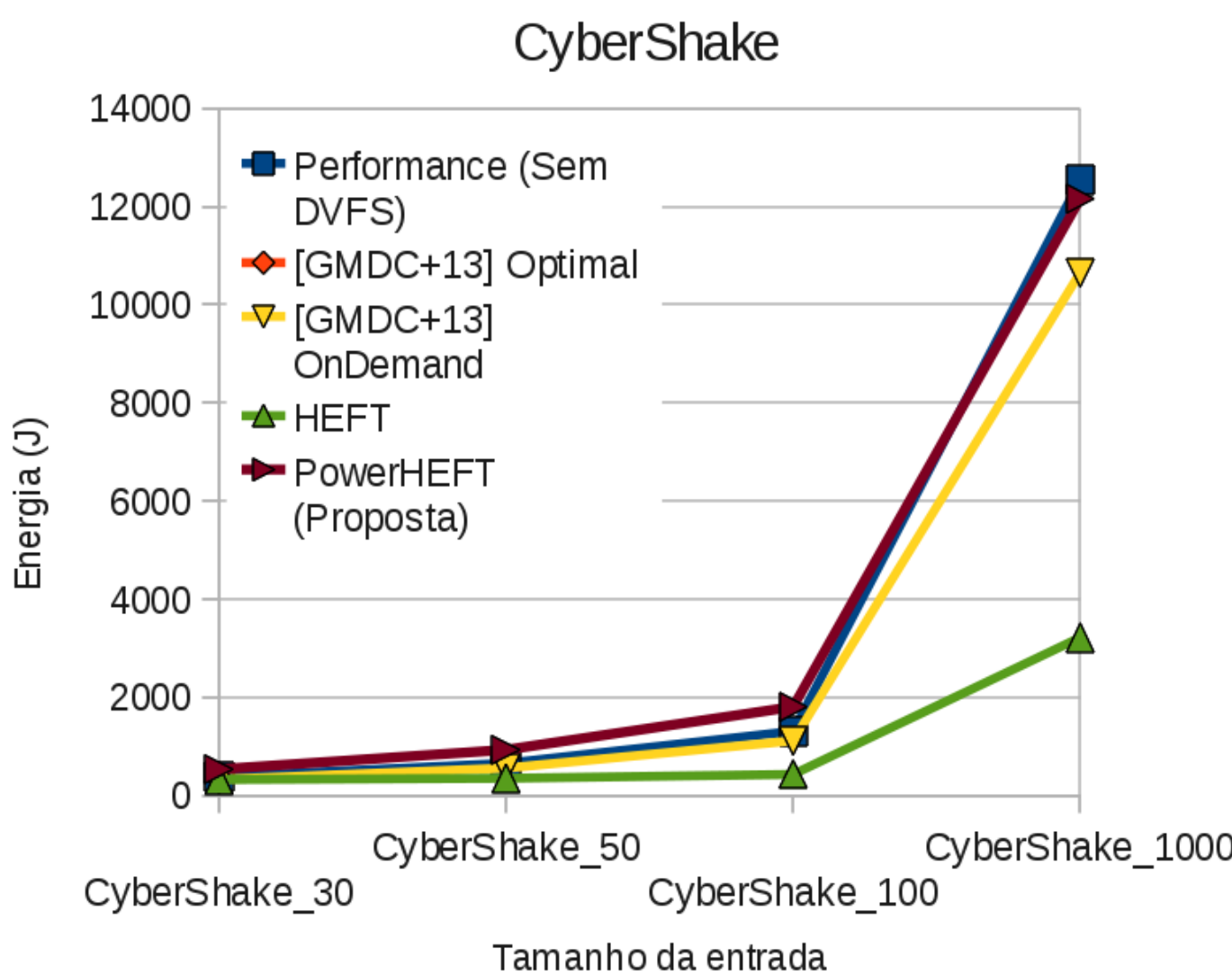
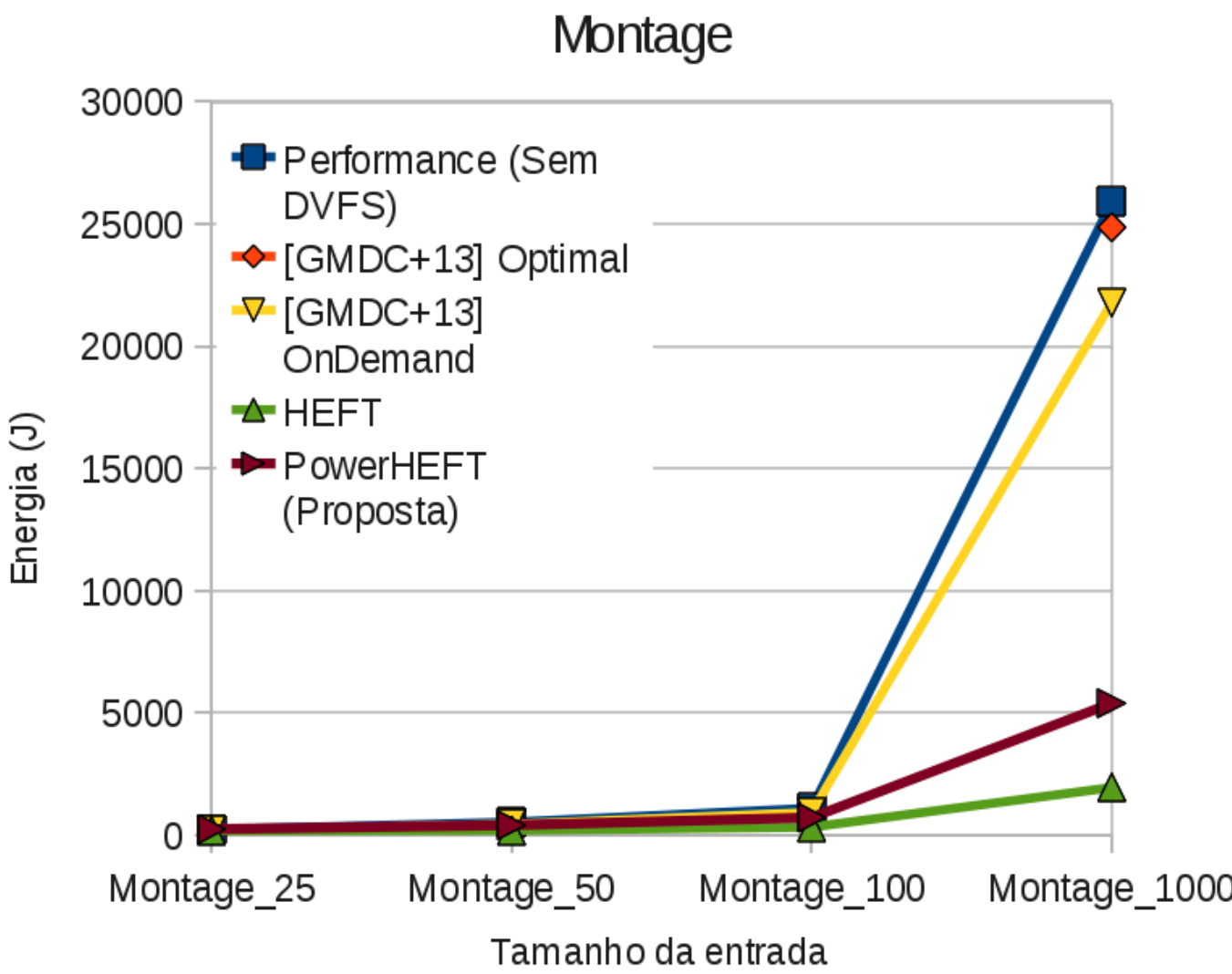
Os dados abaixo foram adaptados de [THW02].



Tarefa	P1 (s)	P2 (s)	P3 (s)	$rank_u(n_i)$
1	14	16	9	108.000
2	13	19	18	77.000
3	11	13	19	80.000
4	13	8	17	80.000
5	12	13	10	69.000
6	13	16	9	63.333
7	7	15	11	42.667
8	5	11	14	35.667
9	18	12	20	44.333
10	21	7	16	14.667



Resultados



Refer ncias

[BH07] L.A. Barroso e U. H lzle. The case for energy-proportional computing. *Computer*, 40(12):33–37, 2007.

[Gar07] Gartner. Gartner Estimates ICT Industry Accounts for 2 Percent of Global CO2 Emissions, 2007. <http://www.gartner.com/newsroom/id/503867> [Online; acessado em 7 de dezembro de 2013].

[GMDC+13] Tom Gu rout, Thierry Monteil, Georges Da Costa, Rodrigo Neves Calheiros, Rajkumar Buyya e Mihai Alexandru. Energy-aware simulation with dvfs. *Simulation Modelling Practice and Theory*, v.39, i.1, p.76–91, 2013.

[PH12] D.A. Patterson e J.L. Hennessy. *Computer Organization and Design: The Hardware/software Interface*. Morgan Kaufmann Series in Computer Graphics. Morgan Kaufmann, 2012.

[THW02] H. Topcuoglu, S. Hariri e Min-You Wu. Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE Transactions on Parallel and Distributed Systems*, 13(3):260–274, 2002.