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

Pedro Paulo Vezzá Campos **pedrovc@ime.usp.br**
Orientador: Prof. Dr. Daniel Macêdo Batista **batista@ime.usp.br**
Departamento de Ciência da Computação — Instituto de Matemática e Estatística — Universidade de São Paulo

Introdução

TI é responsável por aproximadamente 2% das emissões anuais de CO_2, próximo do nível gerado pela aviação [Gar07]. Ao mesmo tempo, a **Lei de Moore**, que profetiza que o poder computacional de dispositivos dobra a cada 18 meses, está chegando ao fim da sua vida [PH12]. Processadores modernos atingiram uma barreira de potência mas no entanto não eram eficientes no **consumo energético** [BH07]. Assim, novas tendências surgiram na indústria: processadores mais simples, mais paralelos e mais eficientes.

Computação em nuvem surgiu como uma consequência quase natural destas tendências. Ao consolidar poder de processamento, transferência de dados e armazenamento é possível reduzir custos e desperdícios. Algumas estratégias possíveis: **consolidação** de máquinas virtuais, dimensionamento de tensão e frequência (**DVFS**) e **algoritmos energeticamente eficientes**.

Aplicações de processamento paralelo podem ser modeladas como **digrafos acíclicos** (DAGs). O problema de decidir qual o melhor **escalonamento** de uma tarefa (vértice do DAG) em uma máquina de forma a otimizar o uso de algum recurso é um problema **NP-difícil**. Assim, heurísticas são necessárias para encontrar soluções aproximadas.

Resultados

Este TCC apresenta um **novo algoritmo** de escalonamento de fluxos de trabalho em computação em nuvem voltado para a eficiência energética. O desempenho foi comparado com o trabalho "*Energy-aware simulation with DVFS*" [GMDC+13] e com um algoritmo de escalonamento clássico mas sem um foco na eficiência energética. Os estudos contaram com a contribuição da aluna de mestrado **Elaine Naomi Watanabe** (elainew@ime.usp.br).

Como resultados secundários, foram feitas **contribuições** a projetos de **software livre** na forma de divulgação de código fonte, incrementos e notificações de falhas nos simuladores de computação em nuvem utilizados pelo autor.

Escalonamento de fluxos de trabalho com computação em nuvem

O *Heterogeneous Earliest Finish Time* (**HEFT**) [THW02] é uma boa heurística para o problema de escalonamento. Ele recebe como parâmetros um DAG a ser escalonado, um conjunto possivelmente heterogêneo de máquinas que realizarão o processamento, os tempos de processamento de cada tarefa em cada máquina e o tempo de transmissão entre duas tarefas. Ele é dividido em duas fases: **priorização** e **seleção**.

Priorização

Qual tarefa escalonar primeiro? A fórmula abaixo é o critério de priorização do HEFT. Além de gerar uma **ordem topológica**, dá prioridade a tarefas mais críticas do DAG.

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

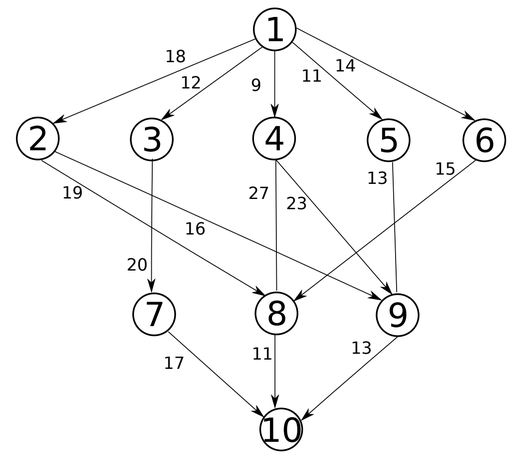
Seleção

Em qual máquina escalonar uma tarefa? O HEFT tenta minimizar o **tempo mais cedo de conclusão** de cada tarefa na esperança que isso minimize a conclusão do fluxo de trabalho como um todo.

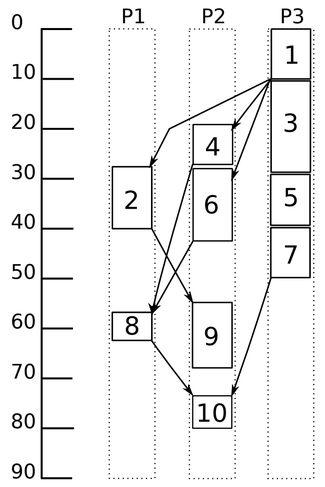
HETEROGENEOUS-EARLIEST-FINISH-TIME()
1 Defina os custos computacionais das tarefas e os custos de de comunicação das arestas com valores médios
2 Calcule $rank_u$ para todas as tarefas varrendo o grafo de "baixo para cima", iniciando pela tarefa final.
3 Ordene as tarefas em uma lista de escalonamento utilizando uma ordem não crescente de valores de $rank_u$.
4 **enquanto** há tarefas não escalonadas na lista
5 Selecione a primeira tarefa, n_i da lista de escalonamento.
6 **para** cada processador p_k no conjunto de processadores
7 Calcule o tempo mais cedo de conclusão da tarefa n_i considerando que ela execute em p_k
8 Defina a tarefa n_i para executar no processador p_j que minimiza o tempo mais cedo de conclusão da tarefa n_i.

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

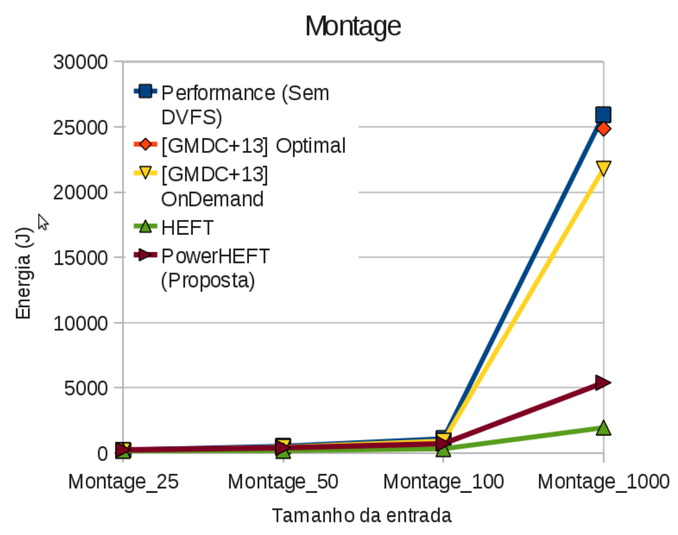


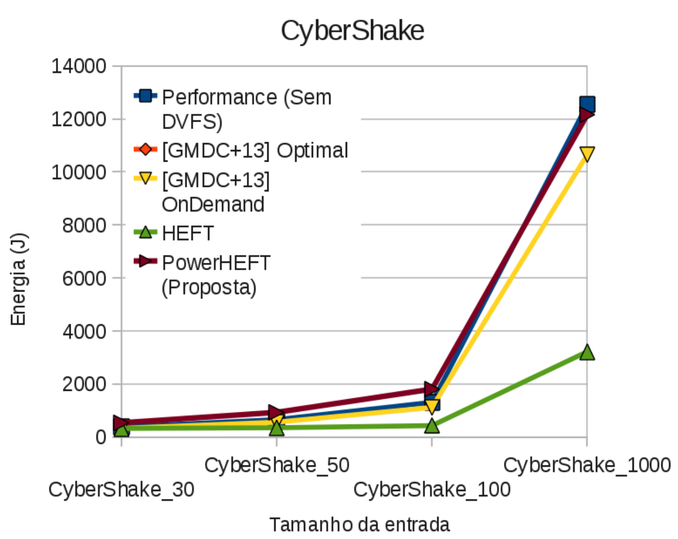
Algoritmo Proposto

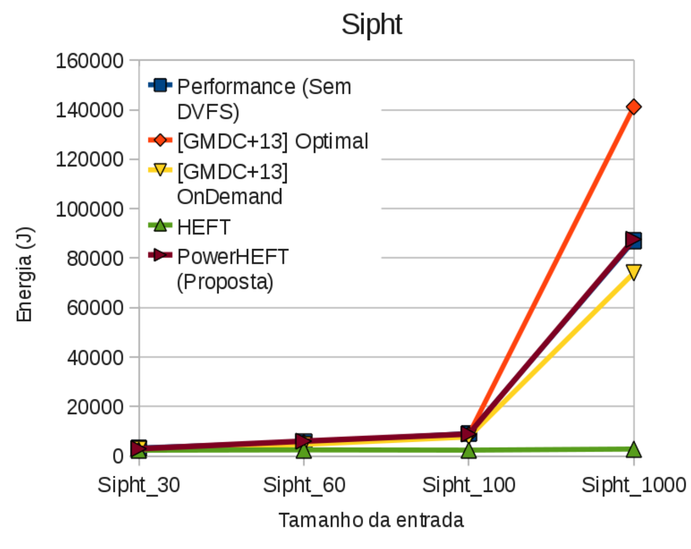
ESCALONARPOWERHEFT($tarefa, VM$)
1 F = filhos diretos da $tarefa$ no DAG
2 Escalone $tarefa$ em VM
3 Escalone F utilizando o algoritmo HEFT
4 // A modelagem energética utilizada está descrita em [GMDC+13]
5 $energia$ = ESTIMARENERGIACONSUMIDA()
6 Volte para o escalonamento do começo do laço
7 **retorne** $energia$

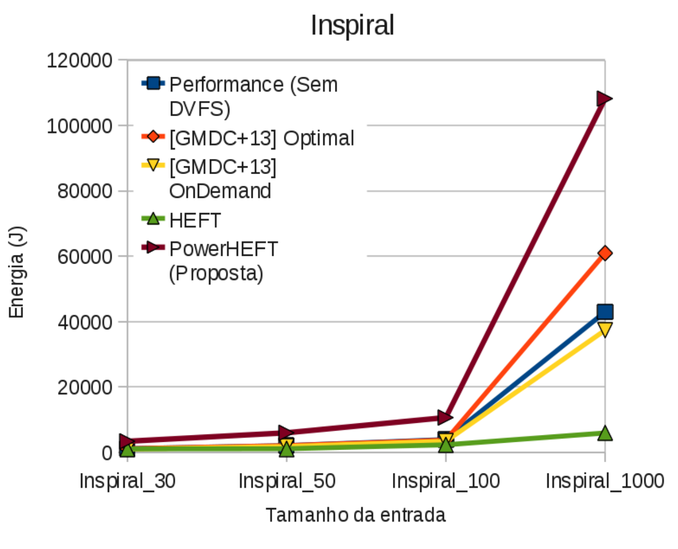
POWERHEFTLOOKAHEAD()
1 $V = \{VmMaisRápida\}$ // VMs usadas ao escalonar
2 O = os tipos de VMs que podem ser instanciadas
3 Ordene o conjunto de tarefas segundo o critério $rank_u$
4 **enquanto** há tarefas não escalonadas
5 t = a tarefa não escalonada de maior $rank_u$
6 // Vamos tentar escalonar t 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 não 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 necessário

Resultados









Referências

[BH07] L.A. Barroso e U. Hölzle. 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 Guérout, 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.