Economia de Energia em Sistemas Embarcados com Multiprocessadores Homogêneos ou Heterogêneos

Introdução ao Escalonamento e Aplicações

Carlos Herrera Muñoz Victoriano Phocco Diaz

16 dezembro 2009

1. Introdução

Os atuais e futuros processadores embebidos assim como também os microprocessadores de alto desempenho, o consumo de energia é uma das características mais importantes no tempo de desenho. Os avances e melhoras na tecnologia de fabricação de processadores permitiram duplicar o numero de transistores integrados em cada geração de microprocessadores. As tecnologias vão implementando características em tamanhos cada vez menores e é esperado que o consumo de energia estático acrescente exponencialmente com os futuros avances tecnológicos. Nas tecnologias atuais o consumo de energia dinâmico domina ao consumo de energia estático, uma técnica eficaz para reduzir o consumo de energia estático é a variação de tensão elétrica (voltagem) também conhecida como "Dynamic Voltage Scaling", este é jeito mais comum para reduzir o consumo de energia em sistemas multiprocessadores, no qual um processo é escalonado para que rode em tantos processadores como for possível e depois aplicar uma diminuição de voltagem, esta técnica é chamada "Schedule-and-Stretch" (S&S), mais tem a desvantagens que não pensa no consumo de energia estática. Outras técnicas são apresentadas nos artigos "Leakage-Aware Multiprocessor Scheduling" e "CASPER: An Integrated Energy-Driven Approach for Task Graph Scheduling

on Distributed Embedded Systems" as quais são LAMPS e CASPER respectivamente. A técnica LAMPS tem por objetivo minimizar o consumo de energia utilizando uma heurística de escalonamento que determina o equilíbrio entre "Dynamic Voltage Scaling" y "Processor Shutdown" (desligar certos processadores) neste jeito o consumo de energia estática. A outra técnica apresentada é CASPER a qual muda o ponto de vista de como é tratado este problema pela maioria de técnicas, CASPER trata a atribuição, escalonamento e controle de energia de um jeito conjunto, não separadamente como outras técnicas abordam o problema, e utiliza um algoritmo genético para procurar a melhor relação entre eles e minimizar o consumo de energia. Nosso trabalho enfoca-se em descrever as técnicas LAMP y CASPER para depois compará-las e mostrar suas vantagens e desvantagens.

2. Definição do Problema

Atualmente os processadores como o IBM/Sony/Toshiba Cell e o ARM11 MPCore foram introduzidos ao mercado de alto desempenho. No tempo de desenho destes processadores é fundamental ter em conta o consumo de potencia gerado por duas partes uma a parte dinâmica que é quando o processador esta executando uma tarefa (comutação dos transistores) e a outra estática que é a perdida de potencia quando o processador está ligado, mas não executando uma tarefa (corrente de fuga).

Nos anos seguintes devido ao incremento de transistores nos processadores é esperado que o consumo de potencia estático supere ao consumo de potencia dinâmico.

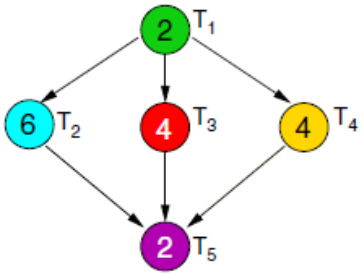
Para garantir o desempenho geralmente os sistemas acrescentam o numero de processadores mais do que eles realmente precisam, para depois escalonar as tarefas em todos eles com o menor "deadline" possível e por ultimo reduzir a voltagem e as freqüências de operação, esta técnica é chamada "Schedule-and-Stretch" (S&S).

No algoritmo anterior não foi tomando em conta a corrente de fuga, e por isto que neste trabalho é apresentado um primeiro algoritmo, LAMPS "Leakage Aware Multiprocessor Scheduling" o qual determina o balance ótimo entre o numero de processadores, reduzindo a corrente de fuga, e nível de voltagem/freqüência. Estes dois algoritmos S&S e LAMPS depois são modificados para suportar desligamento de processadores (PS).

3. Preliminares

Modelo se sistema e aplicação

Nós assumimos que temos um sistema que roda aplicações paralelamente onde o escalonamento das tarefas é determinado estaticamente, as aplicações (conjunto de tarefas) são representadas por DAG (Directed Acyclic Graphs) onde as setas são as dependências entre tarefas, os pesos dos nos são o tempo de processamento.



Modelo se de potência

O artigo utiliza o modelo de potência descrito por Jejuricar onde o consumo de potência de um processador é dado por:

$$P = P_{ac} + P_{dc} + P_{on}$$

Onde P_{ac} é o consumo de potencia dinâmico, P dc é o consumo de potencia estático e o P_{on} é a potencia para manter ligado o processador.

As equações 1 2 3 4 5 e a tabela 1 mostram o modelo de potencia para a tecnologia 70nm, a freqüência máxima de este processador é 3.1 GHz e requere um voltagem de 1V

$$P_{AC} = aC_{cff}V_{dd}^2 f \tag{1}$$

$$P_{DC} = V_{dd}I_{subn} + \mid V_{bs} \mid I_j \tag{2}$$

$$I_{subn} = K_3 e^{K_4 V_{dd}} e^{K_5 V_{bs}} \tag{3}$$

$$f = (V_{dd} - Vth)^{\alpha}/L_d K_6 \tag{4}$$

$$V_{th} = V_{th1} - K_1 V_{dd} - K_2 V_{bs} \tag{5}$$

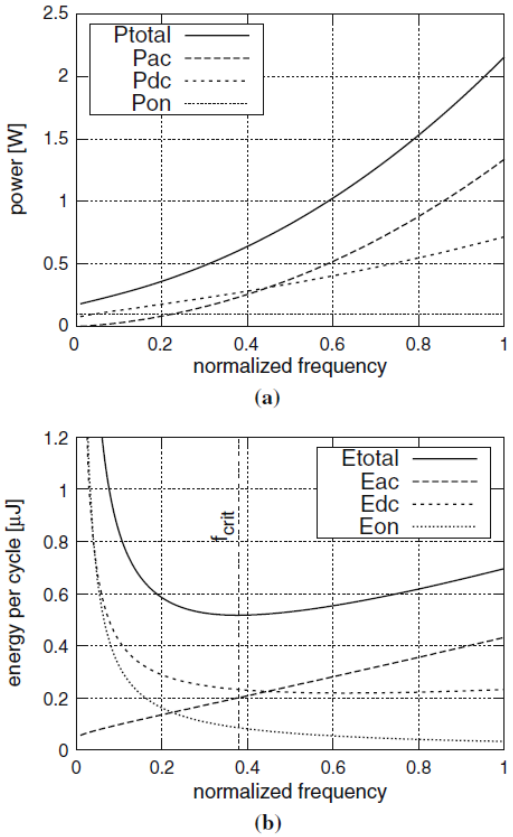


Figure 2 Power and energy consumption as a function of the normalized frequency. **(a)** Power consumption. **(b)** Energy consumption.

Constant	Value
K_1	0.063
K_2	0.153
K_3	$5.38 \cdot 10^{-7}$
K_4	1.83
K_5	4.19
K_6	$5.26 \cdot 10^{-12}$
K_7	−0.144
V_{dd0}	1.0
V_{bs}	−0.7
α	1.5
V_{th1}	0.244
I_j	$4.8 \cdot 10^{-10}$
C_{eff}	$0.43 \cdot 10^{-9}$
L_d	37.0
L_g	$4.0 \cdot 10^{6}$

Figura 1: Constantes

4. Técnicas baseadas em gerenciamento da potência

DVS

Desde que a energia é igual a poténcia pelo tempo, o consumo de energia com certeza vai começar acrescentar se a freqüência é reduzida por embaixo de certo ponto, esse ponto é chamado freqüência ótima ou critica.

Decrementando a freqüência por embaixo de esse ponto reduzir a potência consumida, mas não o total de energia consumida desde que o processador pode ser desligado para o tempo restante.

Desligamento de Processador

Consiste em por temporariamente os processadores desocupados em modo dormido (sleep) ou desligado, a vantagem desta técnica é que reduz todos os jeitos de consumo de energia, não só da parte dinâmica, mas quando um processador é desligado o conteúdo por exemplo as cachês e os preditores ramo são perdidos.

5. Técnicas baseadas em escalonamento

Nos escalonamentos produzidos por S&S e LAMPS, todos os processadores rodam na mesma freqüência e esta freqüência é constante ao longo de tudo o escalonamento, S&S e LAMPS utilizam listas escalonadas com "earliest deadline first " LS-EDF. Para saber se outros algoritmos de escalonamento têm melhor desempenho, assumimos um modelo onde os processadores desocupados não consumem energia.

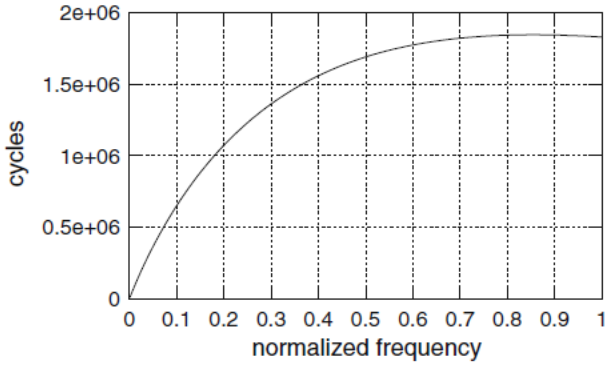
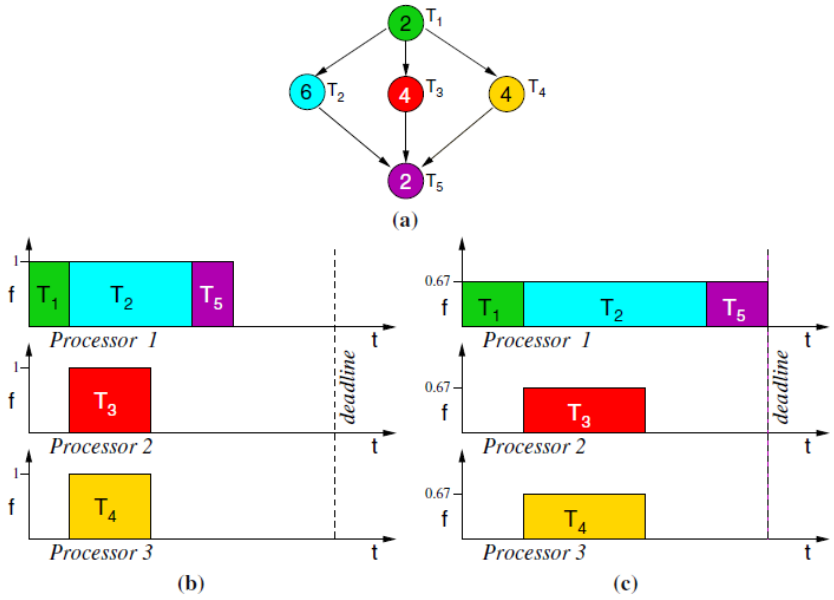


Figure 3 Minimum number of idle cycles required for PS to be beneficial, as a function of the normalized processor frequency.

i. Schedule & Strech

Primeiro o grafo é escalonado com EDF para reduzir o *makespan*, isto acrescenta o valor entre o deadline e o termino da ultima tarefa, este tempo é usado para decrementar o nível de voltagem/freqüência em todos os processadores.

Figure 4 Illustration of S&S. (**a**) Task graph. (**b**) Schedule produced by EDF. (**c**) Schedule produced by S&S.



ii. LAMPS

Este algoritmo encontra o balance entre o numero de processadores que deveriam ser utilizados é o nível de voltagem/freqüência aplicada. Os processadores restantes são desligados.

Primeiro determinamos o numero mínimo de processadores requeridos para finalizar a tarefa antes do deadline, primeiro damos um limite inferior no numero de processadores para completar a tarefa e depois damos um limite superior no numero de processadores que podem ser utilizados eficientemente, depois utilizamos uma pesquisa binária para determinar o numero mínimo de processadores requeridos para terminar o grafo de tarefas no tempo preciso.

Depois de ter encontrado o numero mínimo de processadores requeridos, vamos calcular o numero de processadores que dissipam a menor quantidade de potência. Este passo é determinado calculando o consumo de o numero mínimo de processadores requeridos isto é feito baixando a freqüência do relógio e a voltagem subministrada de modo que o grafo de tarefas seja completado tão próximo como for possível ao deadline.

Figure 5 Pseudocode for the LAMPS heuristic.

```
Nmin ⇐ findMinimumProcs(N_lwb, N_upb)
numProcessors ⇐ N_min
M ⇐ ∞
E_min ⇐ ∞
while true do
  S ⇐ scheduleGraph(G, numProcessors)
  if M = makespan(S) then
    break                          /* exit if the makespan is not reduced */
  end if
  M ⇐ makespan(S)
  f_min ⇐ ⌈M · f_max/deadline⌉     /* rounded up to the next discrete voltage level */
  E ⇐ calculateEnergyConsumption(S, f_min)
  if E < E_min then
    E_min ⇐ E
  end if
  numProcessors ⇐ numProcessors + 1
end while
```

iii. S&S+PS e LAMP+PS

No S&S+PS o tempos restante depois de completar a ultima tarefa é usado para desligar um processador se o período é suficientemente grande para valer a pena consumir energia adicional devido à pedida do estado.

No LAMPS+PS determinamos o numero de processadores para encontrar o deadline e o numero de processadores que executam o grafo de tarefas eficientemente, para cada um dos valores desse intervalo, determinamos o balance entre DVS e OS variando a freqüência da máxima até a mínima requerida para encontrar ao deadline. Para cada freqüência usamos o tempo restante para desligar o processador do mesmo jeito do que S&S+PS

Figure 7 Illustration of LAMPS and S&S+PS. (**a**) Schedule produced by LAMPS. (**b**) Schedule produced by S&S+PS.

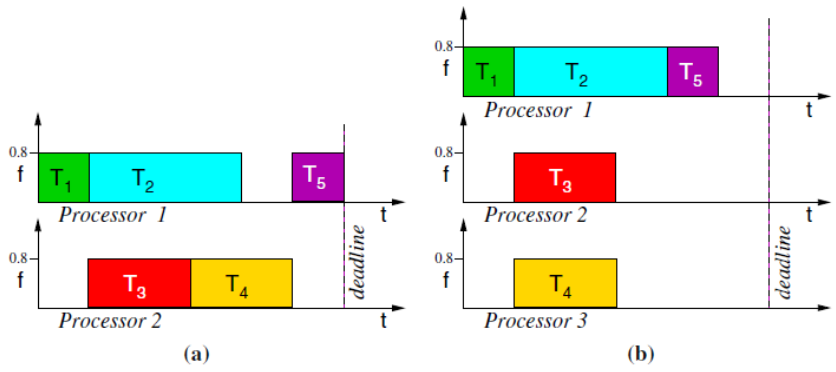


Figure 8 Pseudocode for the LAMPS+PS heuristic.

```
Nmin ⇐ findMinimumProcs(N_lwb, N_upb)
numProcessors ⇐ N_min
M ⇐ ∞
E_min ⇐ ∞
while true do
  S ⇐ scheduleGraph(G, numProcessors)
  if M = makespan(S) then
    break                          /* exit if the makespan is not reduced */
  end if
  M ⇐ makespan(S)
  f_min ⇐ ⌈M · f_max/deadline⌉     /* rounded up to the next discrete voltage level */
  for f = f_max to f_min do
    S' ⇐ processShutdownPeriods(S, f)
    E ⇐ calculateEnergyConsumption(S', f)
    if E < E_min then
      E_min ⇐ E
    end if
    numProcessors ⇐ numProcessors + 1
  end for
end while
```

iv. LIMIT-SF e LIMIT-MF

EDF não sempre da o escalonamento ótimo por isso nós damos dois limites inferiores (lower bounds) um para o caso com uma freqüência só, e outro para múltiplas freqüências.

No LIMIT-SF nos assumimos que os processadores desocupados não consumem energia, por tanto o numero de processadores é igual ao numero de tarefas, e que a freqüência é baixada até a freqüência ótima para encontrar o deadline ou tão perto for possível.

A diferencia entre LIMIT-SF e LIMIT-MF é que todas as tarefas em LIMIT-MF são escalonadas na freqüência critica, dado que os processadores desocupados não consumem energia, o LIMIT-MF é um limite inferior incluso se os processadores pudessem rodar freqüências diferentes e a freqüência pudesse cambiar todo o tempo.

v. Experimentos

Nesta seção nós apresentamos e comparamos os resultados dos diferentes algoritmos de escalonamento. Nós usamos o modelo de potência que já descrevemos numa entrega anterior, também aclaramos que o processador em estado dormido consume 50uW e que desligando o processador e ligando de novo consume 483uJ de energia.

Contexto dos Experimentos

Para as experiências usamos grafos de tarefas de *Standard Task Graph Set* , também utilizamos um grafo de tarefas de codificação MPEG-1, o qual consiste em codificar uma seqüência de 15 frames I, B e P, isto é apresentado na figura 9 Nós temos usado o tempo máximo de execução para a seqüência de *Tennis*, variado para encontrar a freqüência 3.1Ghz a qual é a máxima freqüência do relógio. O *deadline* foi fixado a 0.5 segundos para o GOP de 15 frames, para corresponder a um requisito de codificação de tempo real de 30 frames por segundo.

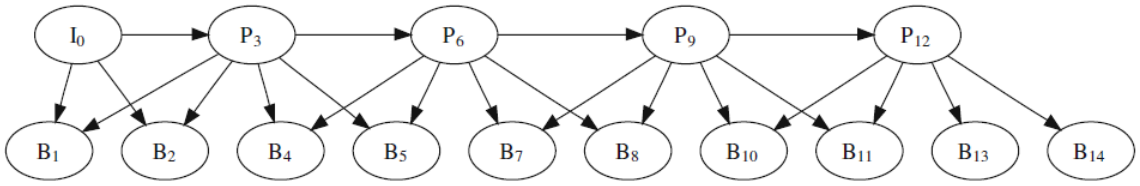


Figure 9 Dependence graph for processing 15 MPEG-1 frames, assuming execution times of 36700900, 178259300, and 73401800 cycles for I, B, and P frames respectively.

O *Standard Task Graph Set* fornece três grafos os quais são gerados por as aplicações: fpppp, robot, e sparce. Este conjunto também tem 2700 grafos gerados de um jeito aleatório, agrupados por o numero de nós. Cada grupo em este conjunto tem 180 grafos diferentes. Como os resultados para diferentes grafos são comparáveis, nós só apresentamos resultados para 50, 100, 500, 2000, e 5000 nós. Para ambos, os grafos de uma aplicação real e os grupos de grafos aleatórios, o numero de nós e arestas, o caminho crítico e a suma de todos os pesos dos nós (trabalho total) são listados na tabela 2

Table 2 Employed benchmarks from STG [27] and their main characteristics.

Name	Number of nodes	Number of edges	Critical path	Total work
fpppp	334	1196	1062	7113
robot	88	130	545	2459
sparse	96	128	122	1920
50	50	66–926	24–447	204–644
100	100	138–1898	29–569	458–1347
300	300	412–8991	45–1164	1517–3568
500	500	698–24497	67–1941	2563–5530
1000	1000	1378–99164	50–3298	5179–11138
2000	2000	2797–396760	48–6770	10563–21615
5000	5000	7132–2491411	62–17386	27009–54010

Como *Standard Task Graph Set* não fornece *deadlines,* nós utilizamos *deadlines* de 1.5, 2, 4, 8 vezes o CLP (comprimento do caminho crítico) quando o processador roda na freqüência máxima de 3.1 GHz. O STG também não define pesos para os nós, é por isso que nós geramos os pesos como inteiros de 1 até 300. Por isso são considerados 2 casos. O primeiro corresponde a tarefas de grão medianamente grande, um peso de 1 em um grafo de tarefas implica um tempo de execução de 3.1x10^6 ciclos, o qual é um milissegundo quando o processador roda numa freqüência de 3.1GHz. O segundo cenário corresponde a tarefas de grão fino, o mesmo peso implica um tempo de execução de 3.1x10^4 ciclos, o qual na freqüência máxima toma 10 microssegundos.

Resultados para o STG

As figuras 10 e 11 descrevem o do consumo de energia relativo para tarefas de grão grande. Para cada cenário, nós mostramos o consumo de energia para *deadlines* de 1.5, 2, 4, 8 vezes o caminho crítico. Cada figura mostra os resultados para quatro diferentes abordagens explicados anteriormente como também os limites teóricos. Ao longo de esta seção, S&S é usado como linha base contra a qual comparamos as outras heurísticas.

Primeiro nós comparamos o consumo de energia dos escalonamentos produzidos por LAMPS com o consumo de energia gerado pelos escalonamentos de S&S. As figuras [ref 10] ref[11] mostram que LAMPS melhora a S&S principalmente para *deadlines* menos estrito. Isto pode ser esperado porque para deadlines ajustados (1.5x o CLP), LAMPS precisa o mesmo ou quase o mesmo numero

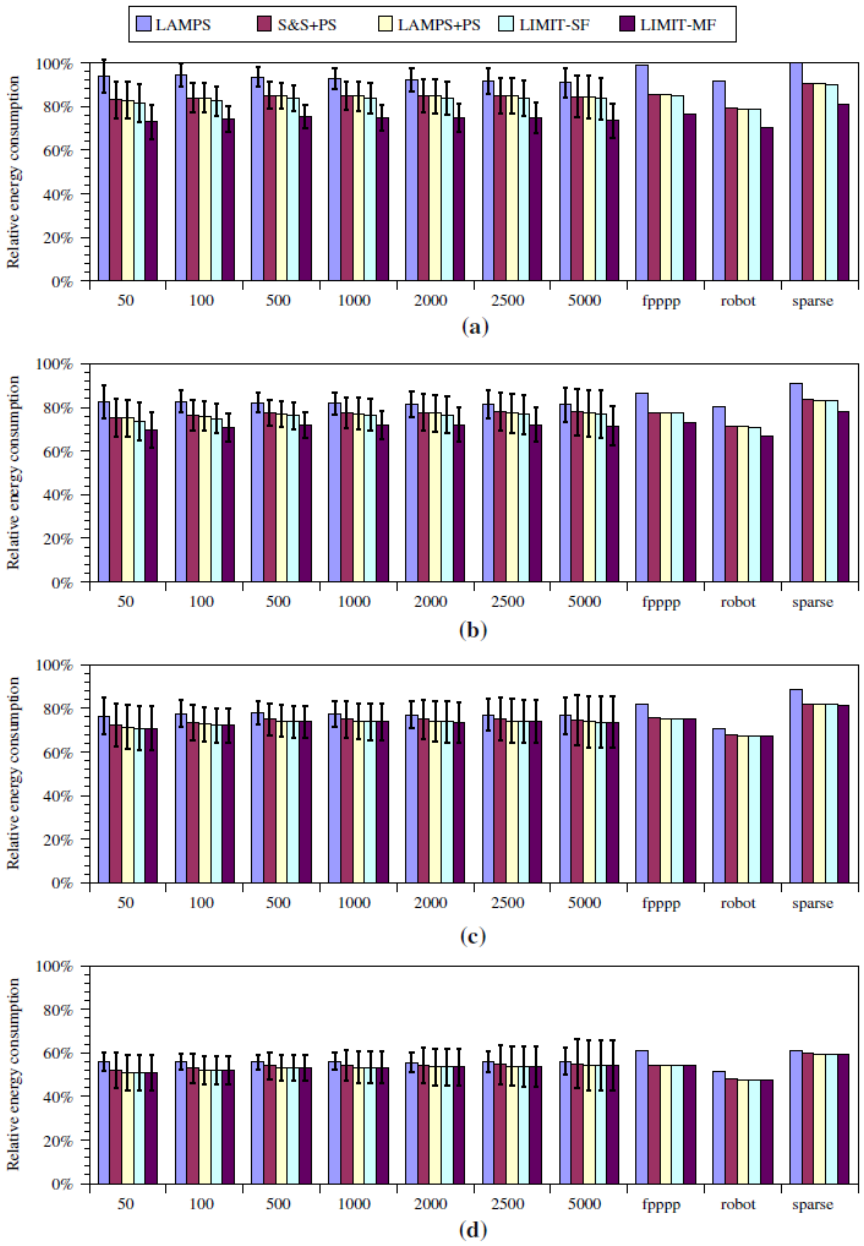


Figure 10 Energy consumption for coarse-grain tasks. (**a**) Deadline = 1.5 × CPL. (**b**) Deadline = 2 × CPL. (**c**) Deadline = 4 × CPL. (**d**) Deadline = 8 × CPL.

de processadores que S&S para cumprir o *deadline*, e, portanto, consume a mesma ou quase a mesma quantidade de energia que S&S. Em outras palavras, se o *deadline* é ajustado, há menos oportunidade de desligar processadores. Por outro lado, para amplos *deadlines* (8x o CLP), LAMPS consume significativamente menos energia que S&S, isto é óbvio porque o LAMPS utiliza menos processadores. Em este caso LAMPS reduz o consumo total de energia em 45% sobre a media comparado com S&S o qual tem um máximo de 67%. Para tarefas de grão fino, descritas na figura 11 as diferencias relativas entre S&S e LAMPS são as mesmas que no caso de tarefas de grão grande quando ambas heurísticas não desligam processadores.

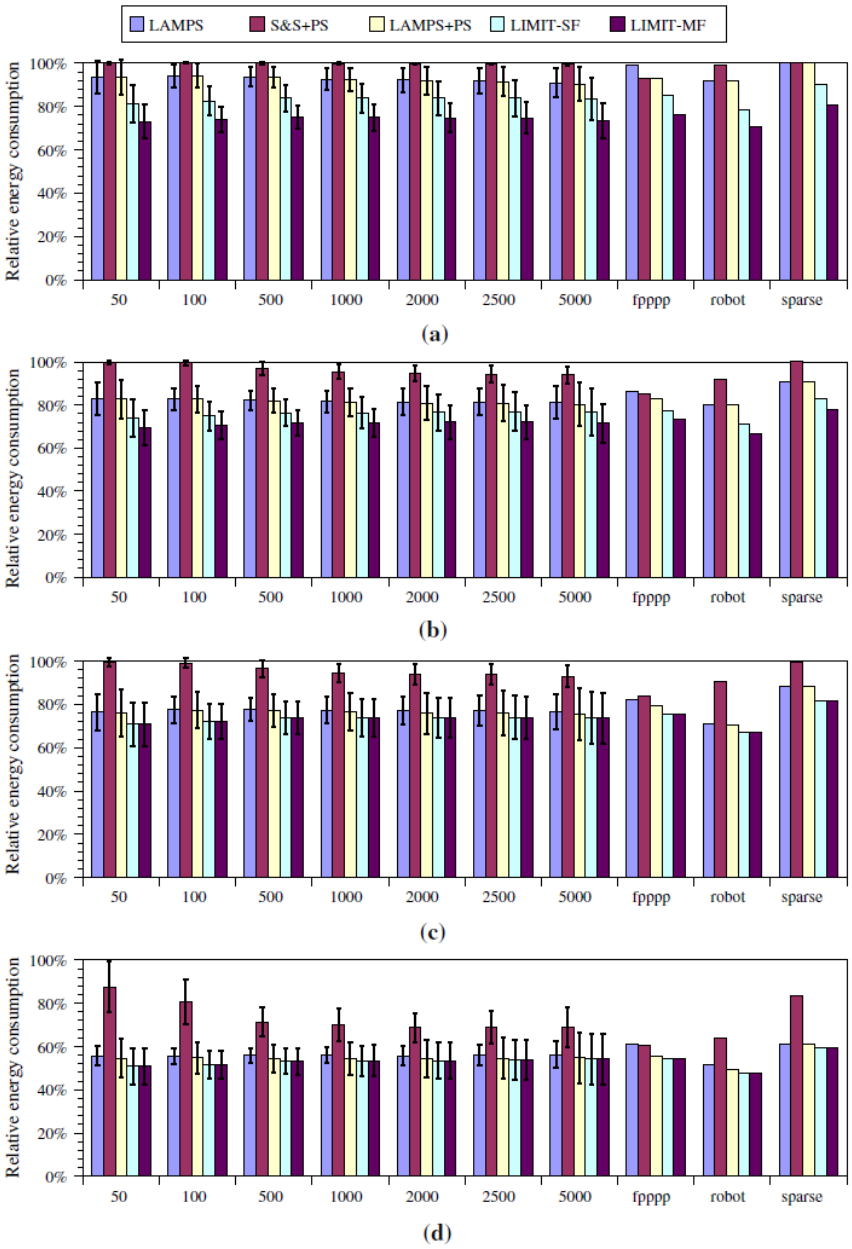


Figure 11 Energy consumption for fine-grain tasks. (a) Deadline = 1.5 × CPL. (b) Deadline = 2 × CPL. (c) Deadline = 4 × CPL. (d) Deadline = 8 × CPL.

Agora nós comparamos S&S+PS com S&S. Porque S&S utiliza um numero grande de processadores, consome uma quantidade grande de potência estática. Por isto S&S+PS melhora a S&S significativamente, por o só fato de desligar os processadores não utilizados temporariamente. Os ganhos, em este caso, são consideravelmente grandes para tarefas de grão grande (23% na media com um *deadline* de 2x do CLP*)* que para tarefas de grão fino (4% na media com um *deadline* de 2x do CLP), porque neste caso a folga usualmente não é suficientemente grande para fazer um desligamento benéfico.

LAMPS+PS melhora a LAMPS, mas para tarefas de grão grande. De novo, a razão principal é que para tarefas de grão fino a folga não é suficientemente grande, não obstante uma quantidade significativa de energia pode ser salvada desligando temporariamente os processadores. O melhoramento de LAMPS+PS sobre LAMPS é tipicamente menor que o melhoramento de S&S+PS sobre S&S. Isto é devido porque em LAMPS a dissipação estática já foi reduzida usando o menor numero de processadores comparado com S&S. Para tarefas de grão grande a melhora máxima de LAMPS+PS sobre a LAMPS foi de 12% e 18%, para *deadlines* de 1.5x e 8x o CLP respectivamente.

Para tarefas de grão grande, a melhora total de LAMPS+PS sobre S&S é 16% na media, com um Maximo de 46% para *deadlines* de 1.5x o CLP e um Maximo de 73% para *deadlines* de 8x o CLP. Para tarefas de grão fino, LAMPS+PS melhorou sobre S&S em 8% na media, com um Maximo de 40% para *deadlines* de 1.5x o CLP e um Maximo de 71% para *deadlines* de 8x o CLP.

LIMIT-SF nas figuras 10 e 11 dão um limite superior na poupança de energia, usando o nosso modelo atual de freqüência única. Usando S&S como linha base e o LIMIT-SF como o Maximo, se mostra que LAMPS+PS alcança mais do 94% da possível redução de energia com tarefas de grão grande, para todas as combinações do *benchmarks* e *deadlines.* Para tarefas de grão fino e *deadlines* estritos (1.5x o CLP) , LAMPS+PS logra mas de 50% da poupança de potência em 54% do *benchmark*. Com *deadlines* menos estritos, LAMPS+PS alcança mais de 88% da possível poupança em todos os *benchmarks.*

Nas figuras 10 e 11**,** Limit-MF é um indicador para as possíveis melhoras que podem ser logradas permitindo que o processador rode a freqüências diferentes, e permitindo que essas freqüências mudem ao longo do tempo. Os resultados indicam que existe uma pequena janela para melhorar quando o *deadline* é relativamente amplo. Para *deadlines* estritos algumas poupanças podem ser logradas, mas mais principalmente para tarefas de grão fino. Neste caso de tarefas de grão fino com *deadlines* estritos os períodos de inatividade são quase sempre muito pequenos como para fazer um desligamento que valha a pena. Neste caso permitir que as freqüências mudem poderia ganhar algumas poupanças extras. Não obstante quando o *deadline* é menos estrito e o grafo de tarefas tem muitas tarefas de grão grande, desligar processadores vale a pena. Neste caso escalonar tarefas a diferentes freqüências não vai melhorar significativamente.

Para explicar melhor porque LAMPS e LAMPS+PS logram poupanças significativas para certos grafos de tarefas, as figuras 12 e 13 descrevem o total de energia dividida por o total de trabalho como função da media quantidade

de paralelismo. A figura 12 mostra esses resultados para tarefas de grão grande, enquanto a figura 13 mostra os resultados para tarefas de grão fino. Em ambos os casos, é usado um *deadline* de 2x o CLP. A energia total tinha sido dividida por o total do trabalho porque tem quase uma relação linear entre os dois. A quantidade media de paralelismo é definida como o total de trabalho dividida pelo CPL. Cada ponto representa um grafo de tarefa, em ambas as figuras mostram os resultados para grafos gerados de um jeito aleatório com 1000, 2000, 2500 e 3000 nós.

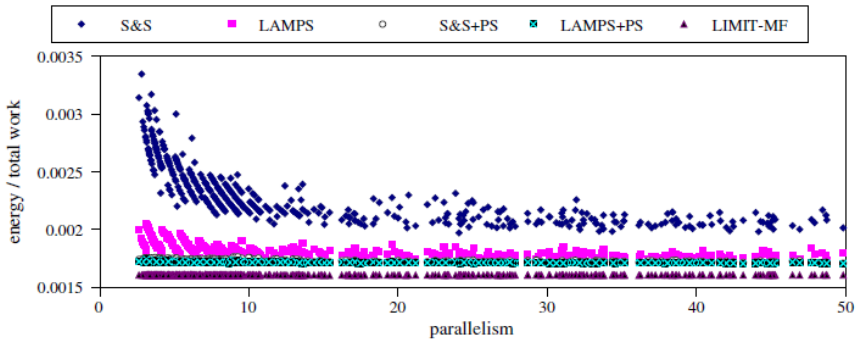


Figure 12 Energy/total work as a function of the average amount of parallelism for coarse-grain tasks. Each dot represents one task graph.

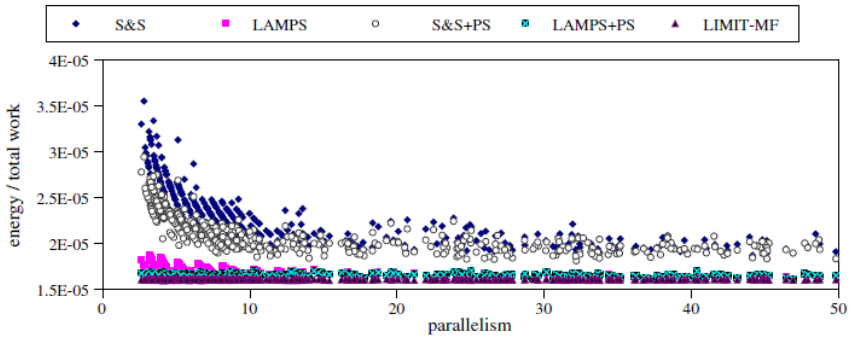


Figure 13 Energy/total work as a function of the average amount of parallelism for fine-grain tasks. Each dot represents one task graph.

Das figuras pode dar se conta que o consumo de energia por unidade de trabalho para S&S incrementa significativamente quando a quantidade media de paralelismo é pequena. O mesmo é visível para S&S+PS com tarefas de grão fino. Isto mostra que especialmente quando a quantidade media de paralelismo é pequena, o consumo de energia vai incrementar quando o desligamento de processadores não esteja disponível ou não possa ser usada eficientemente. A razão principal é que S&S tentara usar tantos processadores como for possível, mas quando o paralelismo é pequeno os processadores

estarão inativos, mas continuarão consumido energia. Para tarefas de grão fino, os períodos de inatividade não são sempre suficientemente longos para salvar energia desligando processadores temporariamente, é por isso que S&S+PS com tarefas de grão grande consumem mais energia que LAMPS e LAMPS+PS, Para ambos as LAMPS e LAMPS+PS uma pequena quantidade de paralelismo não afeta significativamente o consumo de energia por unidade de trabalho, como ambos as abordagem podem decidir usar menos processadores.

Os resultados agrupados nestas figuras, especialmente visível quando o paralelismo é baixo, são escalonamentos que melhoram o mesmo numero de processadores, Para soluções com o mesmo número de processadores, a diminuição do consumo de energia como a media de paralelismo, se aproxima mais ao número de processadores utilizados. De novo este efeito é muito mais visível para S&S e S&S+PS, os quais usam tantos processadores como efetivamente podem ser usados para explorar o paralelismo.

Resultados para MPEG-1

A Figura 9 mostra o grafo de tarefas para o benchmark MPEG-1. Os resultados para os experimentos com este benchmark são apresentados na tabela 3. Igualmente aos experimentos, estes números foram obtidos escalonando o grafo de tarefas usando as heurísticas apresentadas anteriormente.

Table 3 Energy consumption for the MPEG-1 benchmark using various approaches.

	S&S	LAMPS	S&S+PS	LAMPS+PS	LIMIT-SF	LIMIT-MF
Energy	18.116	13.290	10.949	10.947	10.940	10.940
# of processors	7	3	7	6	N/A	N/A

Quando S&S é usado para escalonar o grafo, usa tantos processadores como for possível para reduzir o *makespan* do grafo, o qual neste caso é 7 processadores. LAMPS, por outra parte usa 3 processadores é mais eficiente, e é possível reduzir o consumo de enrgia mas de 26% comparado com S&S, S&S+PS também usa o maior numero de processadores, mas tendo a habilidade de desligar os processadores temporariamente, isto reduz o consumo de energia em 40% comparado com S&S, LAMPS+PS reduz o consumo de energia por quase a mesma quantidade que S&S+PS, embora com menos 1 processador. De isto podemos concluir que os períodos de folga no escalonamento são o suficiente para compensar o custo de usar um processador mais. Alem disso, os resultados para S&S+PS e LAMPS+PS são extremamente próximos aos limites inferiores LIMIT-SF e LIMIT-MF. De isto nós podemos concluir que não será possível reduzir ainda mais o consumo de

energia usando um algoritmo de escalonamento diferente ou permitindo que os processadores rodem o mudem diferentes freqüências.

vi. CASPER

Multiprocessadores Homogêneos

La meta dos experimentos é medir a efetividade do framework contra os enfoques que tem separado o mapeamento, ordenação e gerenciamento de energia, e também comparar o framework com enfoques integrados, mas que utilizam um laço externo e outro interno(para a otimização). São utilizados dois conjuntos:

Conjunto de Grafos referenciados (RG) que também foram usados em diferentes pesquisas de similares, consiste de 10 grafos de tarefas RG1-RG10: RG1 e RG2 são utilizados em [1] , RG3 é a filtro de quadratura, RG4 está baseado na eliminação gaussiana para resolver equações , RG5 e RG6 são implementações da transformada rápida de Fourier(FFT) RG7 é uma adaptação do algoritmo PDG, RG8 é uma implantação da transformada de Laplace, RG9 é outra implementação do FFT e finalmente RG10 está baseado na técnica de Analise de valor principal[8]. Os valores dos deadlines dos grafos de tarefas foram gerados usando um método semelhante usado em [5] que esta baseado no CPL (critical path lenght).

O segundo conjunto de grafos de tarefa consiste de 5 grandes e aleatórios grafos de tarefas (aproximadamente 50-100 nodos) que foram gerados usando TGFF [13].

Compara-se HGLS(Heterogeneus/Homogeneus Genetic List Scheduling) cujo enfoque usa também um Algoritmo genético de lista de escalonamentos mas a diferença é que usa um laço externo para minimização do makespan e outro interno para otimização de energia.

Os resultados experimentais do CASPER são apresentados na tabela 1. Aqui é observado que HGLS obtém um melhor makespan do que CASPER, contudo o CASPER atingi consistentemente uma redução do consumo de energia. Em média o HGLS poupa 53.8% de energia contra os 57.8% de CASPER, isto é uma melhora de 7.8% sobre o HGLS.

Task Graph	$\|V\|/\|E\|$	τ_d	HGLS + PDP-SPM		Proposed (CASPER)		
			μ	% saving	μ	% saving	%improv.
RG1	16/24	65	44	57.4	45	60.7	7.8
RG2	17/28	50	37	49.1	38	54.3	10.2
RG3	14/15	130	102	41.0	102	44.0	5.1
RG4	20/39	2120	1596	50.4	1597	52.3	3.7
RG5	28/32	225	150	57.4	151	61.5	9.5
RG6	28/32	460	265	64.1	265	65.5	4.1
RG7	41/69	925	585	58.5	610	62.2	9
RG8	18/29	665	390	62.0	420	65.6	9.3
RG9	95/158	151	118	47.1	122	50.5	6.4
RG10	361/684	17154	11933	58.8	12818	62.2	8.1
TG1	43/74	1400	1014	47.1	1025	50.5	6.5
TG2	68/119	2000	1345	57.1	1353	59.3	5.3
TG3	93/170	3300	2462	49.3	2472	53.5	8.4
TG4	93/170	3300	2132	59.5	2172	67.3	19.3
TG5	113/216	5400	4325	47.8	4422	50.0	4.2
Average Energy Saving				**53.8**	-	**57.3**	**7.8**

Multiprocessadores Heterogêneos

O conjunto consiste de 25 grafos de tarefas(TG) gerados usando também TGFF[13] que inclui grafos de 8 a 100 nodos(tarefas) que são mapeadas a arquiteturas heterogêneas com gestão de potencia DVS-PE habilitado e não habilitado.

Novamente aqui é utilizado HGLS, mas para a etapa de otimização de energia é usada a técnica de PV-DVS , seguidamente o CASPER também é comparado com o algoritmo GMA+EE+GLSA.

Os resultados experimentais do CASPER são apresentados na tabela 2. Aqui é observado que o CASPER atingiu uma maior redução de energia do que HGLS+PV-DVS em um 8.2%, e atingi um 23.3% redução de energia do que GMA+EE+GLSA(também usa dois laços de otimização). Podemos dizer então que uma focalização integrada da alocação, ordenação e escalonamento de tarefas com a otimização de energia oferece melhores resultados se são feitos num só laço.

Task Graph	$\|V\|/\|E\|$	CASPER %Saving	%improv vs. GMA	%improv vs. HGLS
tgff1	8/9	82.84	41.64	18.53
tgff2	26/43	71.43	46.02	-0.24
tgff3	40/77	73.27	19.33	-32.71
tgff4	20/33	88.40	32.25	85.30
tgff5	40/77	87.46	72.74	31.44
tgff6	20/26	92.68	59.00	-44.01
tgff7	20/27	34.65	8.28	3.74
tgff8	18/26	74.46	7.32	69.33
tgff9	16/15	43.42	-5.33	-18.40
tgff10	16/21	38.48	19.49	-78.52
tgff11	30/29	17.85	-10.70	-4.22
tgff12	36/50	88.37	40.52	86.05
tgff13	37/36	51.75	-24.43	34.26
tgff14	24/33	22.99	7.12	2.48
tgff15	40/63	84.84	80.34	71.20
tgff16	31/56	22.29	-9.40	-0.59
tgff17	29/56	94.63	90.17	3.37
tgff18	12/15	8.04	-31.42	-28.51
tgff19	14/19	17.05	-56.93	-31.22
tgff20	19/25	83.40	29.59	-28.35
tgff21	70/99	85.80	78.67	55.62
tgff22	100/135	36.27	-21.34	-22.64
tgff23	84/151	86.72	65.08	6.08
tgff24	80/112	75.43	12.01	10.72
tgff25	49/92	51.07	33.49	15.85
Avg. Energy Saving		60.54	23.34	8.2

6. Conclusões

Como os tamanhos dos dispositivos continuam diminuindo. Dependendo da quantidade de tempo restante (slack) que continua antes do deadline, a quantidade de paralelismo, granularidade da aplicação, variação da voltagem e desligamento de processadores pode ser usado para reduzir significativamente a energia consumida. No mesmo tempo, é importante não utilizar muitos processadores.

LAMPS-PS diminui o consumo de energia para uma implementação do MPEG-1 por não menos de 40% comparado com o S&S.

Para tarefas de grão grosso e freqüência constante o LAMPS+PS alcançou mais do 94% da redução de energia possível.

Os resultados indicam que existe uma pequena janela para melhorar quando o deadline é relativamente grande.

Para deadlines escritos algumas poupanças poderiam ser conseguidas, principalmente para tarefas de grão fino onde os períodos de inatividade são muito pequenos para desligar o processador, neste caso variar a freqüência poderia acrescentar algumas poupanças de energia, entretanto quando o deadline é menos estrito e/ou o grafo de tarefas é razoavelmente de grão grosso, vale a pena desligar os processadores, neste caso escalonar as tarefas a freqüências diferentes não vai melhorar significativamente.

Referencias

[1] A. Al-Maasarani,"Priority-Based scheduling and evaluation of prece- dence graphs with communication times," M.Sc. Thesis, King Fand University of Petroleum and Minerals, Saudi Arabia, 1993.

[2] M. A. Al-Mouhamed, "Lower bound on the number of processors and time for scheduling precedence graphs with communication costs," *IEEE Trans. Software Engineering*, Vol. 16, no. 12, pp. 1390-1401, 1990.

[3] S. Hua and G. Qu, "Power Minimization Techniques on Distrib- uted Real-Time Systems by Global and Local Slack Management," IEEE/ACM Asia South Pacific Design Automation Conference, Jan- uary 2005.

[4] R. C. Correa, A. Ferreira and P. Rebreyend, "Scheduling multiproces- sor tasks with genetic algorithms," *IEEE Tran. on Parallel and Dis- tributed Systems*, Vol. 0, pp. 825-837, 1999.

[5] R. Dick, D. Rhodes, and W. Wolf, "TGFF: Task graphs for free," *Proc. Int. Workshop Hardware/Software Codesign*, pp. 97-101, March 1998.

[6] F. Gruian and K. Kuchcinski, "LEneS: Task scheduling for low- energy systems using variable supply voltage processors," *Proc. of Asia and South Pacific Design Automation Conference*, pp. 449-455, Jan. 2001.

[7] N. K. Jha, "Low power system scheduling and synthesis," *Proc. of Int. Conf. on Computer Aided Design*, pp. 259-263, 2001.

[8] Y. Kwok and I. Ahmad, "Benchmarking and comparison of the task graph scheduling algorithms," *Journal of Parallel and Distributed Computing*, Vol. 59, no. 3, pp. 381-422, Dec. 1999.

[9] J. Luo and N. K. Jha, "Power-profile driven variable voltage scal- ing for heterogeneous distributed real-time embedded systems," *Int. Conf. on VLSI Design*, Jan. 2003.

[10] C. L. McCreary, A. A. Khan, J. J. Thompson, and M. E. McArdle, "A comparison of heuristics for scheduling DAGS on multiprocessors," *Proc. of the Int. Parallel Processing Symp.*, p 446-451, 1994.

[11] R. Mishra, N. Rastogi, D. Zhu, D. Mossé, and R. Melhem, "Energy aware scheduling for distributed real-time systems," *Int. Parallel and Distributed Processing Symp.*, pp. 243-248, April 2003.

[12] M. Schmitz, B. Al-Hashimi, and P. Eles, "Energy-efficient mapping and scheduling for DVS enabled distributed embedded systems," *De- sign, Automation and Test in Europe Conference*, March 2002.

[13] M. Schmitz, B. Al-Hashimi, and P. Eles, "Iterative Schedule Opti- misation for Voltage scalable Distributed Embedded Systems," *ACM Trans. on Embedded Computing Systems*, vol. 3, pp. 182-217, 2004.

[14] D. Sylvester and H. Kaul, "Power-driven challenges in nanometer design," *IEEE Design and Test of Computers*, pp. 12-21, Nov. 2001.

[15] M. -Y. Wu and D. D. Gajski, "Hypertool: A programming aid for message-passing systems," *IEEE Trans. on Parallel and Distributed Systems*, 1(7), pp. 330-343, July 1990.

[16] V.Kianzad, Shuvra S., Gang Qu. "CASPER: An Integrated Energy-Driven Approach for Task graph Scheduling on Distributed Embedded Systems" ECEDepartment and Institute for Advanced Computer Studies Universityof Maryland, CollegePark,MD 20742.