Integração por Romberg de forma paralela em um MPSoC

Gabriel Boni Vicari, Nicolas Lodea

Prof. Dr. César Augusto Missio Marcon
Disciplina de Modelagem Computacional para Sistemas Embarcados

Método de Romberg

O Método de Romberg é um processo iterativo usado para estimar integrais definidas por meio da aplicação da Extrapolação de Richardson repetidamente sob a Regra dos Trapézios Composta.
Assim, o método utiliza a regra de trapézios para calcular os termos, começando no primeiro termo e cada termo subsequente depende de dois termos anteriores (Figura. 1). O último valor da diagonal é o valor mais preciso da integral.

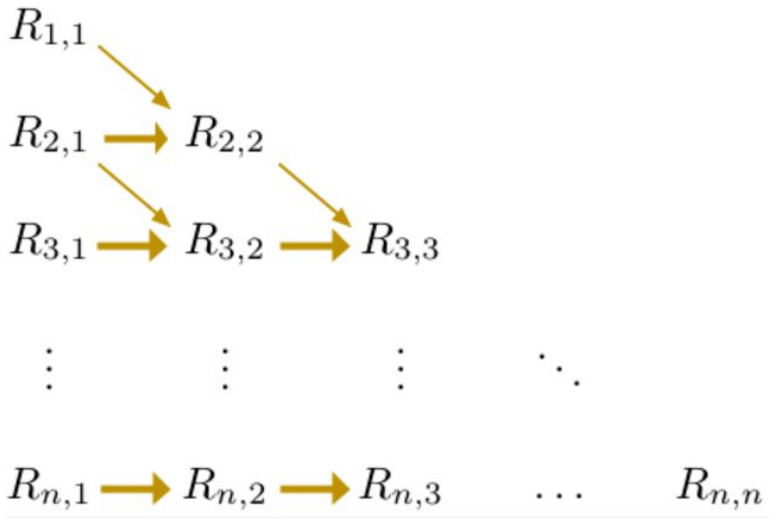

Figura 1. Computação dos valores no método de Romberg.

No projeto, a função escolhida para ser utilizada com o Método de Romberg foi a seguinte:

$$f(x) = \int_a^b \log_4(e^{x \times \ln x}) \times \sin(x)\ dx$$

Objetivos

Este trabalho tem como objetivo apresentar uma implementação sequencial e uma implementação paralela de um algoritmo de integração pelo método de Romberg. Após, os dados são extraídos da implementação paralela possibilitando que a aplicação seja modelada na framework Paloma. Nela são explorados dois balanceamentos: energia e carga. A aplicação balanceada, então, é verificada no framework Cafes utilizando os modelos computacionais CWM, CPCM e ACPM.

Implementação Sequencial

A implementação sequencial do método de Romberg foi obtida na internet, no site: https://physik.uni-graz.at/~pep/CompOriPhys/Python/romberg.py

Esta aplicação possui a função:

```
def romberg(f, a, b, eps, nmax)
```

A função recebe como parâmetros a função `f` a ser integrada, os limites `a` e `b` de integração, a acurácia `eps` desejada e a ordem máxima `nmax` do método, ou seja, o número de trapézios a serem calculados.
Então, ela inicializa uma matriz com o numpy e a preenche com zeros:

```
Q = np.zeros((nmax, nmax), float)
```

Em seguida, percorre-se a matriz para que a primeira coluna de todas as linhas da matriz seja calculado o número de trapézios.

```
N = 2**i
```

Na qual `i` é a linha atual e, então, é calculado o trapézio de `Q[i, 0]` por meio da função

```
def trapezoid(f, a, b, N);
```

que aplica a regra do trapézio composta à uma função `f` utilizando os limites de integração `a` e `b` e resolução `N`.
Essa função calcula o tamanho `h` de cada trapézio

```
h  = (b-a)/N
```

e o valor `s` do trapézio da função `fi` recebida:

```
s = (h/2)*(fi[0]+fi[N])+h*s
```

Após calculado o valor do trapézio, a função de Romberg, então, percorre a matriz calculando os outros valores pela extrapolação de Richardson:

```
Q[i, k+1] = 1.0/(4**(n-1)-1)*(4**(n-1)*Q[i, k]-Q[i-1, k])
```

Por fim, quando o número de iterações escolhido é atingido, é retornado o último valor de Q, ou seja, a melhor estimativa.

Implementação Paralela

Após da compreensão da implementação sequencial do método de Romberg foi implementada a versão paralela. Tendo como base as funções do trapézio e extrapolação, da versão sequencial foi criada uma classe chamada Romberg. Todos os cálculos são realizados pelo método `romberg_thread`. Ele tem como argumentos um valor para retorno do cálculo, a coluna e a linha da matriz `Q` e os valores da linha superior e da coluna anterior.

```
def romberg_thread(self, Q, Q_act, Q_lst, i, k):
```

Dentro desta thread é verificado se o valor a ser calculado pertence a primeira coluna, assim o trapézio é calculado, ou extrapolação de Richardson para os outros valores da matriz.

O segundo método é o `run`, nele todas as threads são geradas, as threads executam o método `romberg_thread`. No método run a matriz é percorrida instanciando as threads, a cada linha da matriz uma thread a mais é gerada, calculando a próxima coluna da linha.

Modelagem da Aplicação no Framework Paloma

Descrição do Processador

A arquitetura selecionada para a aplicação no framework foi Xtensa de 32 bits. O Tamanho da NoC escolhido foi de 3x3, ou seja, 9 processadores com clock de 240MHz (Frequência utilizada nos microprocessadores Tensilica Xtensa LX6). Os outros parâmetros de processador foram os valores padrões do framework.

Descrição da Comunicação

A descrição da comunicação foi baseada nos valores encontrados na aplicação paralela. Assim, chegou-se em uma média de 452 phits, com mínimo e máximo de 32 e 1440 phits, respectivamente, e com desvio-padrão de 46 phits. Estes valores foram calculados na aplicação. A cada iteração o tamanho, em bits, dos argumentos enviados para cada thread são somados, estes são guardados em um vetor correspondente a cada thread.

Descrição das Tarefas

As estimativas de potência e uso de CPU foram baseadas no número de cálculos (iterações) que são realizadas, tomando como base que cada cálculo usa 0,5% do CPU e 0,1 uW de potência e o consumo quando o processador estiver ocioso em foi de 0,01 uW.
Além disso, foi estimado o tamanho do código e de dados, que foram 10 kB e 32 kB, respectivamente.

Assim, o particionamento da aplicação a partir do framework Paloma foi gerado com as restrições de tamanho de código de 32 kB e tamanho de dados de 10 kB. Os outros parâmetros foram deixados como padrão.

Exploração dos Modelos Computacionais no Framework Cafes

Balanceamento de Energia

O particionamento ótimo gerado pelo Paloma visando o balanceamento de energia utilizou cinco processadores, como mostra a Tabela 1.

Processador	Tarefa	Uso de CPU (%)
P0	T0, T5	37,49
P1	T3, T7	36,2
P3	T4, T2	36,86
P4	T1	17,05
P8	T8, T6	33,17

Tabela 1. Resultado do particionamento ótimo gerado por balanceamento de energia.

Utilizando o algoritmo de busca exaustiva no Cafes, foi gerado um mapeamento do uso dos processadores, como mostrado na Figura 2. Na qual P3 e P8 são os mais comunicantes, por isso foram centralizados na NoC, enquanto o P4, por ser o menos comunicante, ficou mais afastado, considerando o roteamento XY da NoC. Os outros três PEs não foram utilizados. O consumo total foi de 364,89 uJ.

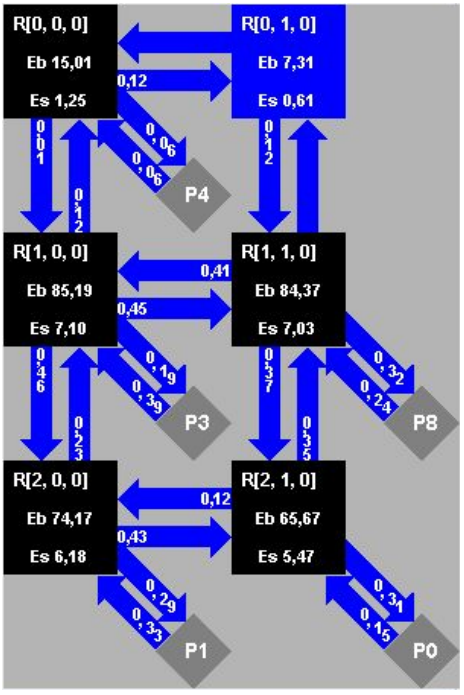

Figura 2. PEs utilizados no ótimo particionamento gerado por balanceamento de energia.

Em seguida, baseado no CWM do balanceamento de energia, foi modelado o CDCM na ferramenta Cafes e chegou-se no resultado mostrado na Figura 3, na qual o consumo de energia total foi de 70,12 uJ e o consumo em modo ocioso foi de 5,78 uJ.

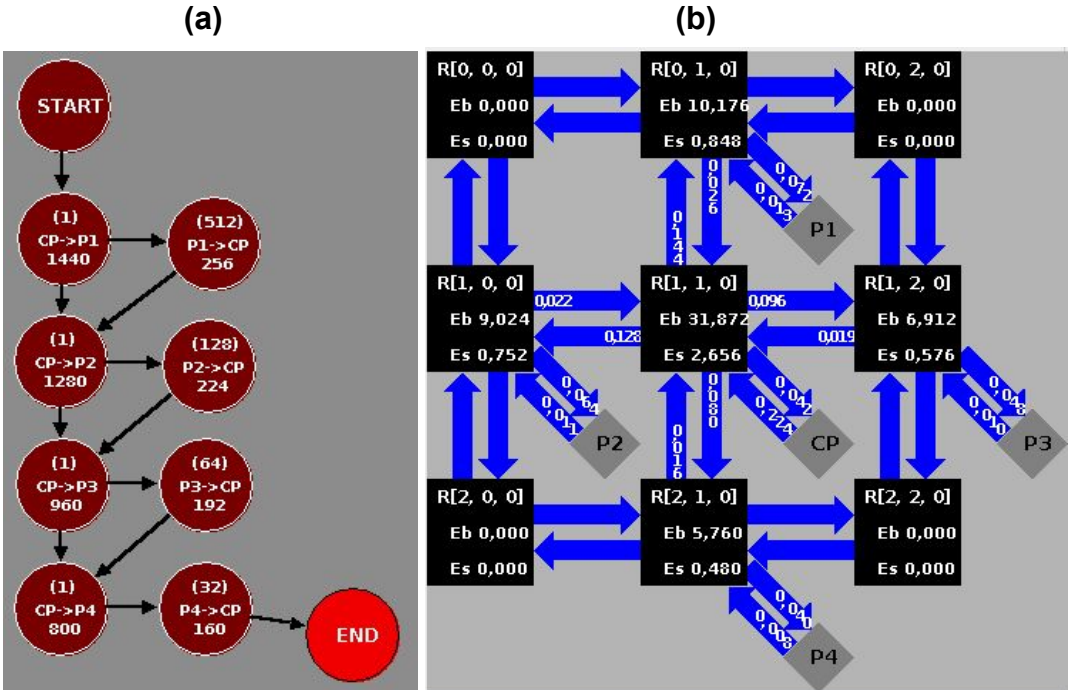

Figura 3a. CDCM modelado na ferramenta Cafes.
Figura 3b. Resultado do CDCM modelado utilizando busca exaustiva.

Após, baseado no CWM do balanceamento de energia, foi modelado o ACPM na ferramenta Cafes e chegou-se no resultado mostrado na Figura 4, na qual o consumo de energia total foi de 13,94 uJ.

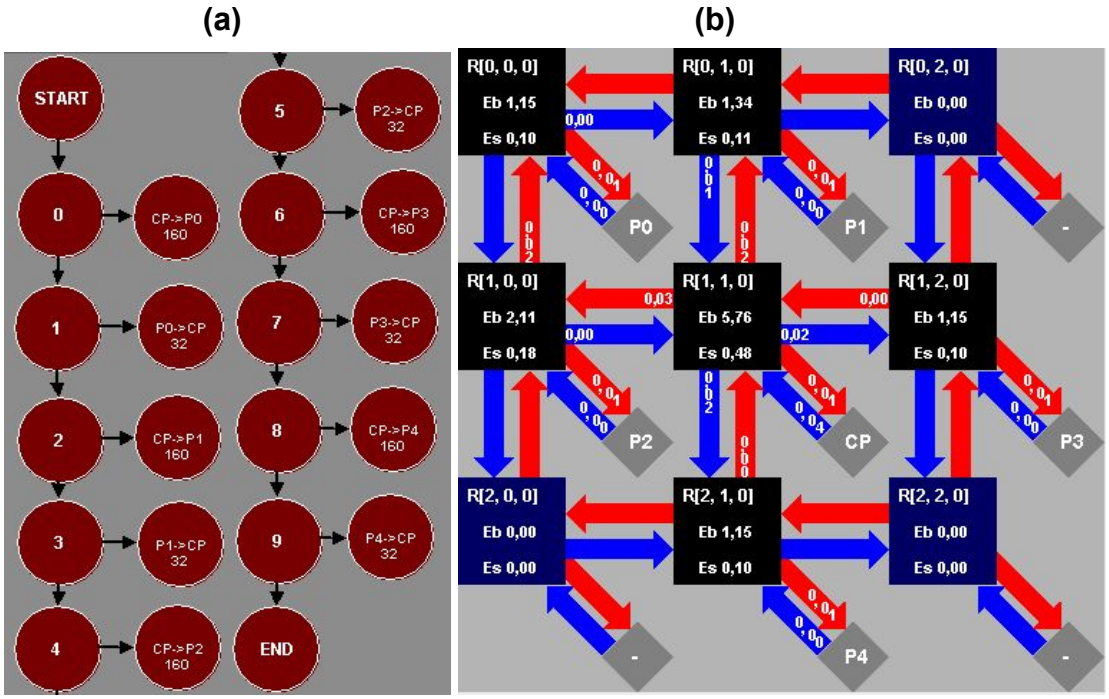

Figura 4a. ACDM modelado na ferramenta Cafes.
Figura 4b. Resultado do ACDM modelado utilizando busca exaustiva.

Balanceamento de Carga

O particionamento ótimo gerado pelo Paloma visando o balanceamento de carga utilizou todos os processadores, como mostra a Tabela 2. E teve um consumo total de 510,64 uJ.

Processador	Tarefa	Uso de CPU (%)
P0	T2	21,02
P1	T4	15,84
P2	T7	16,67
P3	T3	19,53
P4	T8	17,9
P5	T5	20,97
P6	T1	17,05
P7	T6	15,27
P8	T0	15,52

Tabela 2. Resultado do particionamento ótimo gerado por balanceamento de carga.

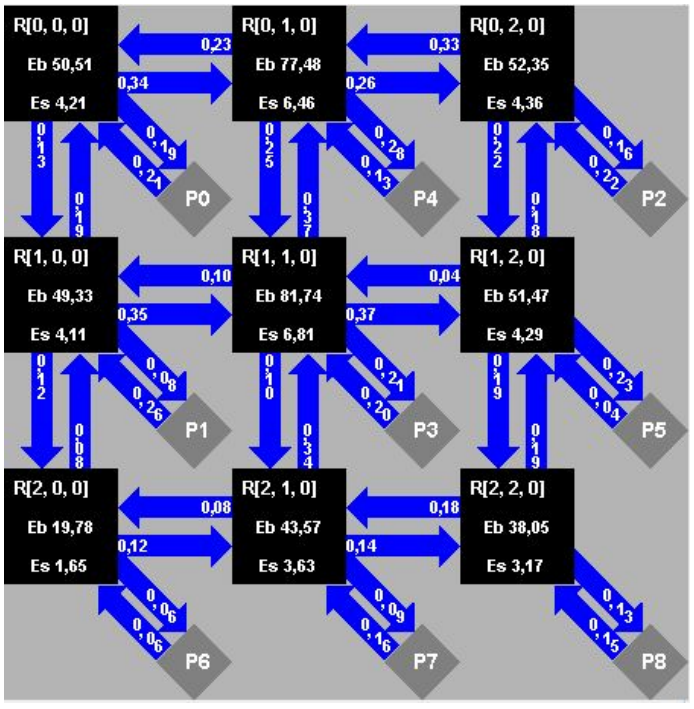

Figura 5. PEs utilizados no ótimo particionamento gerado por balanceamento de carga.

Utilizando o algoritmo de busca exaustiva no Cafes, foi gerado um mapeamento do uso dos processadores, como mostrado na Figura 5. Na qual P3 se tornou o CP, ou seja, o processador central que distribui as tarefas para os outros processadores.

Em seguida, baseado no CWM do balanceamento de carga, foi modelado o CDCM na ferramenta Cafes e chegou-se no resultado mostrado na Figura 6, na qual o consumo de energia total foi de 108,13 uJ e o consumo em modo ocioso foi de 7,72 uJ.

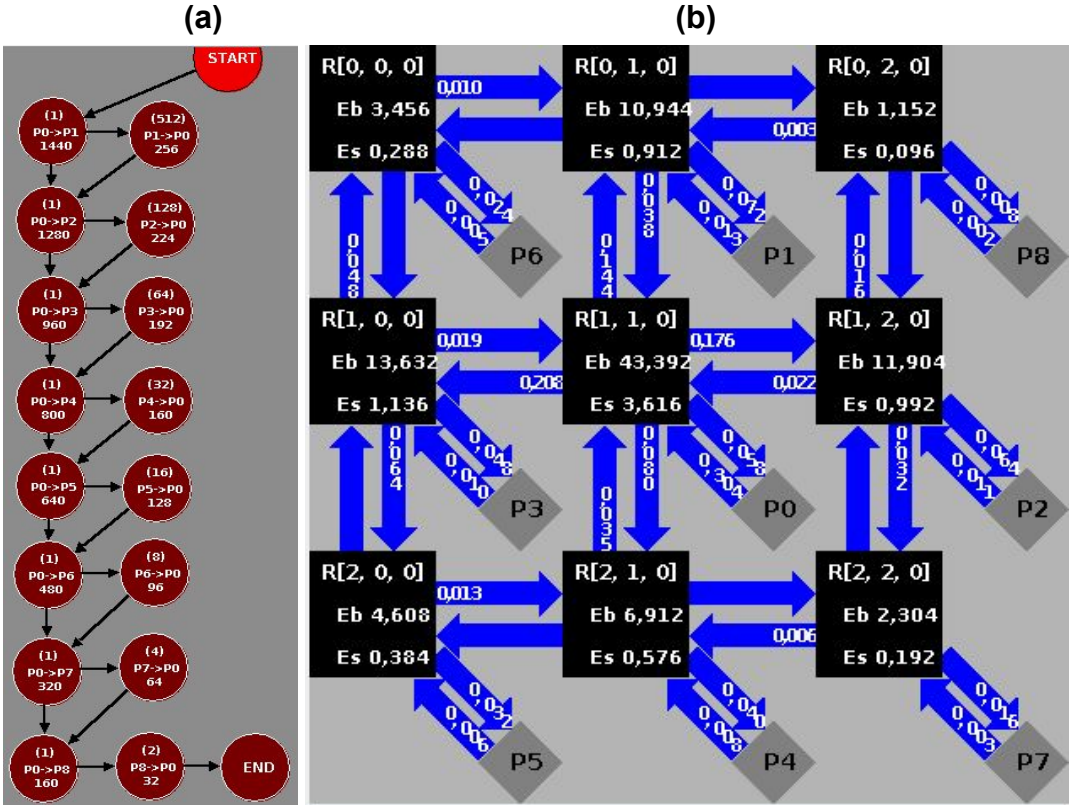

Figura 6a. CDCM modelado na ferramenta Cafes.
Figura 6b. Resultado do CDCM modelado utilizando busca exaustiva.

Após, baseado no CWM do balanceamento de carga, foi modelado o ACPM na ferramenta Cafes e chegou-se no resultado mostrado na Figura 4, na qual o consumo de energia total foi de 25,34 uJ.

(a)

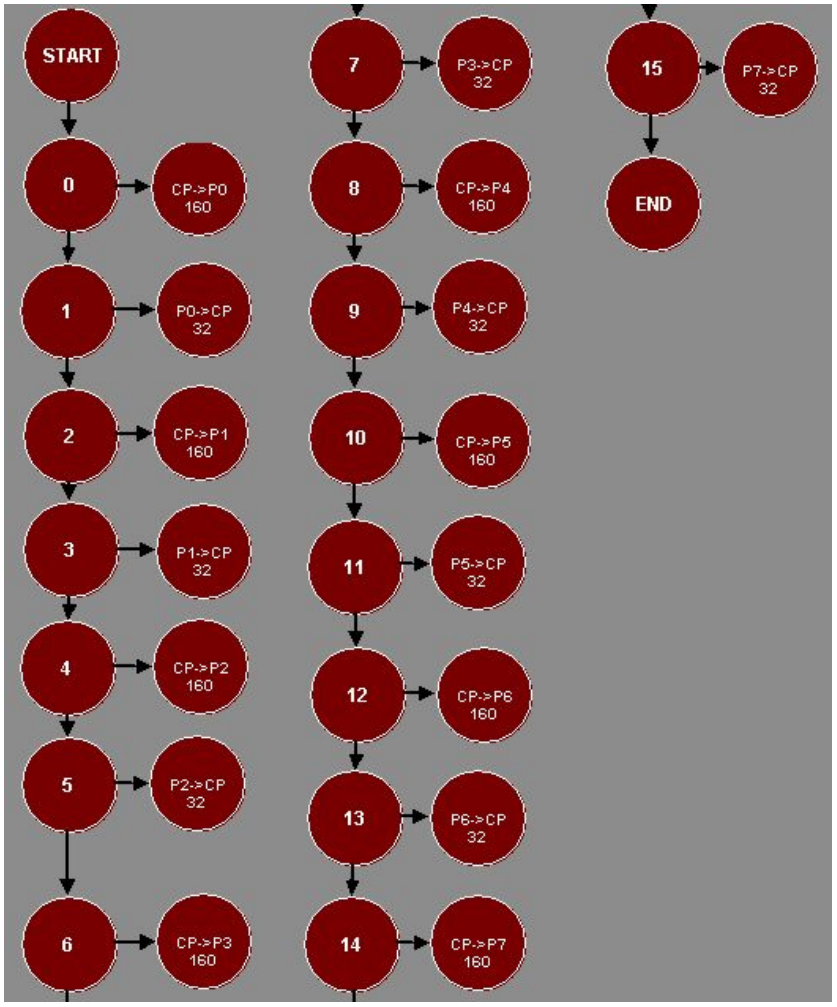

Figura 7a. ACDM modelado na ferramenta Cafes.

(b)

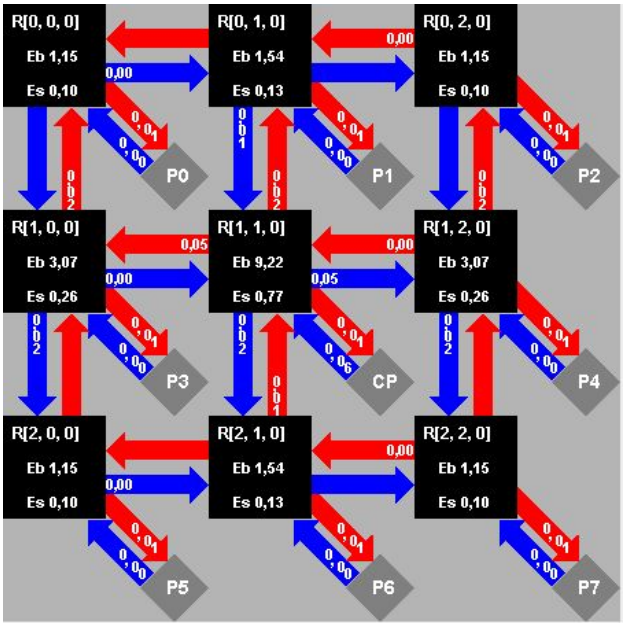

Figura 7b. Resultado do ACDM modelado utilizando busca exaustiva.

Soluções

O gráfico (Figura 8) mostra o consumo de energia dos dois balanceamentos realizados. Como esperado o balanceamento por energia resultou em um consumo 28,52% menor de energia se comparado ao do balanceamento por carga.

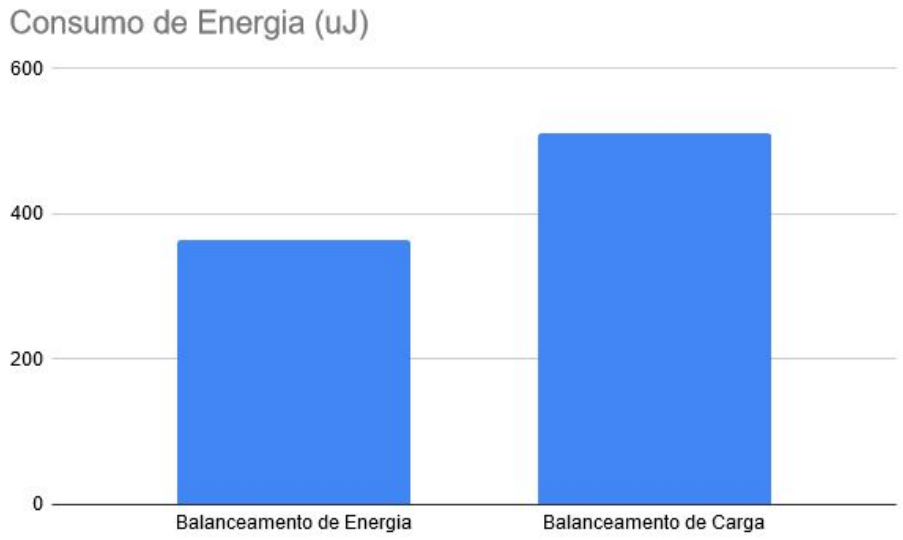

Figura 8. Comparação do consumo de energia entre os balanceamentos.

Além disso, no balanceamento por carga, chegou-se em um tempo de comunicação maior do que o balanceamento por energia. Entretanto, espera-se que o tempo total de execução por balanceamento de energia seja menor.
Porém, é necessário de mais informações da arquitetura, principalmente em questões de consumo de energia, para ter mais conclusões.