Universidade: presente!

UFRGS PROPESQ

XXXI SIC

Salão UFRGS 2019 – Conhecimento, Formação, Inovação

21 . 25 . OUTUBRO . CAMPUS DO VALE

Exploração de espaço de projeto para otimização de aplicações paralelas

Letícia dos Santos
lsantos@inf.ufrgs.br
Orientada por Prof. Antônio Carlos S. Beck

Universidade Federal do Rio Grande do Sul
Instituto de Informática

MOTIVAÇÃO

- Computadores com vários núcleos de processamento.
- Sistemas formados por vários computadores.
- *High-Performance Computing:* necessidade de eficiência energética combinada com alto desempenho.

PROGRAMAÇÃO PARALELA

- Para aumentar desempenho, é possível utilizar programação paralela.
- Ao invés da aplicação ser executada sequencialmente por um único núcleo, o trabalho é dividido em várias tarefas que serão executadas simultaneamente nos múltiplos núcleos do sistema.
- Para executar o seu trabalho, as tarefas precisam se comunicar durante a execução.

MODELOS DE COMUNICAÇÃO E INTERFACES DE PROGRAMAÇÃO PARALELA

- Memória compartilhada (ex.: OpenMP) - requer sistema com memória compartilhada (apenas 1 nó computacional).
- Troca de mensagens (ex.: MPI) - permite mais nós computacionais, mas é menos eficiente em sistemas com apenas 1 nó.
- Híbrido (MPI+OpenMP): Comunicação em memória compartilhada dentro do nó e comunicação por mensagens entre os nós.

OBJETIVO

Analisar desempenho e consumo de energia de diferentes abordagens para programação paralela (OpenMP, MPI e Híbrido).

METODOLOGIA

Ambiente: sistema com 2 nós computacionais, onde cada nó é composto por 2 processadores *octa-core* Intel Xeon E5-2630 (Sandy Bridge) com *multithreading*, totalizando 32 threads por nó.

Aplicações com diferentes comportamentos:

- Cálculo de PI – intensa carga de trabalho para o processador.
- Multiplicação de matrizes - muitas consultas e escritas na memória.

Versões implementadas: Sequencial (sem paralelismo), OpenMP, MPI e Híbrido (MPI+OpenMP).

RESULTADOS

- As Figuras 1 e 2 apresentam o aumento de desempenho e economia de energia em relação a versão sequencial de cada algoritmo.
- Onde, 1 nó permite 32 tarefas e 2 nós permitem 64 tarefas.
- Considerando apenas um nó computacional, OpenMP (OMP) é mais eficiente que MPI, devido a seu modelo de comunicação por memória compartilhada.
- Ao utilizar dois nós não existe memória compartilhada, por isso não é possível utilizar OpenMP puro. Nesse caso, a versão híbrida é utilizada.
- MPI puro é melhor que a versão híbrida.
- A versão MPI com 2 nós usa o dobro de recursos da versão OpenMP (OMP em 1 nó), mas não apresenta ganho proporcional a este aumento.

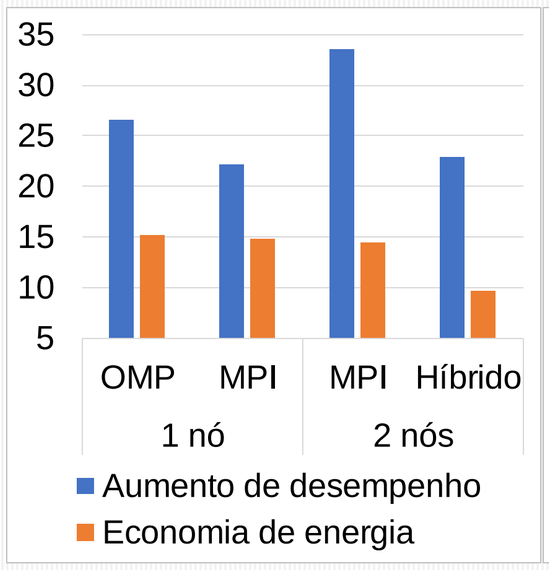

Figura 1 – Cálculo de PI

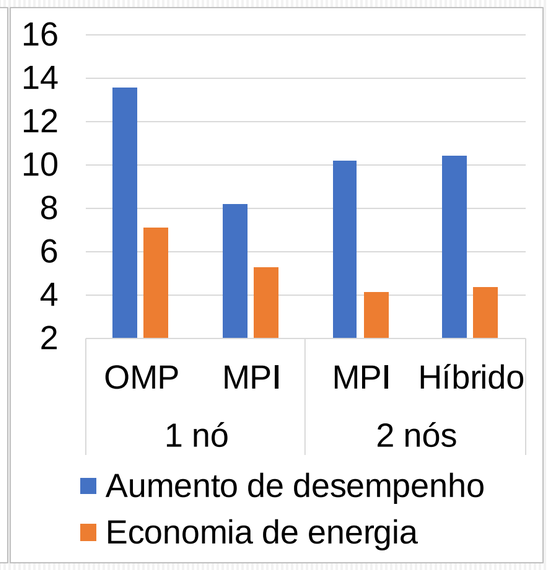

Figura 2 – Multiplicação de matrizes

CONCLUSÃO

- O desempenho da interface depende do tipo de aplicação e do sistema.
- Uso de mais nós computacionais não significa necessariamente melhor desempenho e economia de energia. Isso é devido ao custo de comunicação entre as múltiplas tarefas.
- Para as aplicações executadas, a versão híbrida não apresentou benefícios em relação a versão MPI, muito embora o uso de OpenMP em um único nó seja mais eficiente que MPI.

Referências: RABENSEIFNER, Rolf; HAGER, Georg; JOST, Gabriele. *Hybrid MPI/OpenMP parallel programming on clusters of multi-core SMP nodes.* In: **2009 17th Euromicro international conference on parallel, distributed and network-based processing**. IEEE, 2009. p. 427-436.