NAS Parallel Benchmarks em Ambiente de Memoria Partilhada, Distribuida e Hibrida

Universidade do Minho

Adriana Meireles & Shahzod Yusupov

A82582 A82617

Email:a82582@alunos.uminho.pt Email:a82617@alunos.uminho.pt

***Abstract*— Ao analisar sistemas de computação de elevado desempenho são inúmeros os fatores que se devem ter em conta no ambiente de execução. Neste projeto são demonstradas as condicionantes no desempenho de aplicações que correm no sistema.Para tal, foi utilizado o *NAS Parallel Benchmarks* que possui versões para memória partilhada/ distribuída, sequencial e híbrida, afim de avaliar a performance relativa de programas. Ao longo do trabalho foram analisados diversos parâmetros tais como influência da arquitetura das máquinas, diferentes compiladores e respetivos níveis de otimização, topologia de comunicação e, ainda, diferentes dimensões de dados.**

I. INTRODUÇÃO

De modo a explorar os atuais sistemas de computação de alto desempenho e medir/comparar a sua performance com outros supercomputadores recorreu-se ao pacote *NAS PARALLEL BENCHMARKS* **(NPB)**, que consiste num conjunto de programas cujo objetivo é ajudar a avaliar a performance de supercomputadores paralelos.

Ao longo dos anos, estas benchmarks têm vindo a sofrer várias alterações/melhorias principalmente devido à constante evolução dos supercomputadores.Apesar de serem constituídos por inúmeros kernels e pseudo-aplicações, neste trabalho serão estudados apenas 3, com o intuito de avaliar e comparar os diferentes paradigmas : sequencial, memória partilhada, memória distribuída e também híbrido:

EP: Embarrassingly Parallel, tal como o nome indica é um kernel embaraçosamente paralelo o que indica que terá uma boa escalabilidade, tornando interessante a sua análise.

CG: recorre ao método do Gradiente Conjugado para calcular uma aproximação ao menor valor próprio de uma matriz esparsa simétrica de grandes dimensões. Este kernel testa computação e comunicação nao estruturada,o que em princípio indica má performance, sendo portanto interessante o seu estudo.

SP-MZ: pseudo-aplicação que resolve sistemas nao lineares baseado em algoritmos *scalar pentadiagonal*, sendo esta uma versão multizona útil para explorar os vários níveis de paralelismo e testar paradigmas de implementação híbridos, tornando interessante a sua comparação com os restantes.

II. CARACTERIZAÇÃO DO HARDWARE

Para a realização deste trabalho foram utilizados dois tipos de nós do SeARCH, os nós 431 e 662. Estes nós foram escolhidos para avaliar o comportamento dos kernels pois possuem arquitecturas distintas de processadores e suporte de ligação Myrinet e Gigabit Ethernet. De seguida é apresentada a caracterização do hardware dos nós escolhidos:

TABLE I

*

Especificações Cluster

	Processador	
Nodo	**662**	**431**
Nome código	Ivy Bridge	Nehalem
Modelo	Intel Xeon E52695v2	Intel Xeon X5650
# Cores	12	6
# Threads	24	12
Frequência base	2.4 GHz	2.66 GHz
Frequência turbo	3.2 GHZ	3.06 GHz

III. CARACTERIZAÇÃO DO SOFTWARE

Um dos fatores importantes a ter em conta neste estudo é o software utilizado na comparação da

	Memória
Nodo	**662**
Cache L1	12 x 32 Kib para Dados (8-way) 12 x 32 kib para Instruções (8-way)
Cache L2	12 x 256 Kib
Cache L3	30 Mib
Max Memory Bandwidth	59.7 GiB/s
Main Memory	64 GiB

	Memória
Nodo	**431**
Cache L1	8 x 32 Kib para Dados (8-way) 8 x 32 kib para Instruções (8-way)
Cache L2	8 x 256 Kib
Cache L3	12 Mib
Max Memory Bandwidth	32 GiB/s
Main Memory	48GB

performance dos diferentes kernels. Neste caso em específico, a nível de compiladores, serão usados o compilador da **GNU** e da **INTEL**, com dos diferentes níveis de otimização **O2** e **O3**.

Para além disso,para monitorizar o sistema durante a execução de testes será usada a ferramenta **dstat** que permite visualizar todos os recursos do sistema instantaneamente , desde a utilização da memória ao CPU, agregando a informação dos comandos *vmstat, netstat, iostat, ifstat e mpstat*.

IV. SELEÇÃO DAS CLASSES DE DADOS

De modo a obter resultados significativos é necessário avaliar o comportamento dos kernels para diferentes cargas computacionais. Cada implementação dos kernels contém diversas classses/tamanhos de problema que vão desde a classe **S** à classe **F**. As classes S e W por serem consideradas pequenas e para testes rápidos foram descartados deste estudo, enquanto que as classes D,E,F foram descartados pelo motivo oposto, não sendo viávelo seu estudo. Assim, foram comparadas as classes A, B e C no paradigma de versão sequencial, de modo a escolher os tamanhos de dados significativos para este estudo:

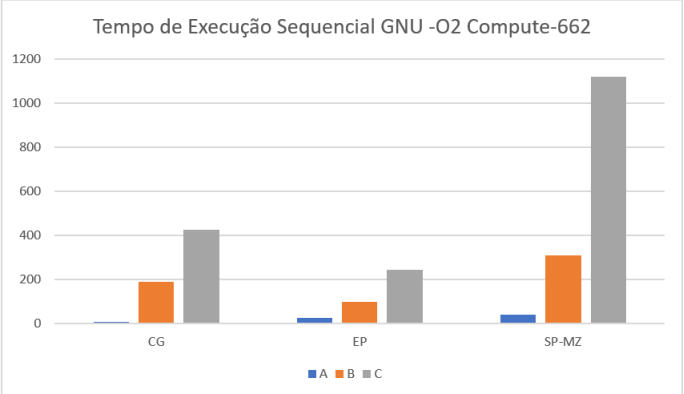

Fig. 1. Tempo de execução da versão sequencial dos kernels CG,EP, SP-MZ no nó 662 com nível de otimização O2 do compilador da GNU para as classes A, B e C.

Da análise do gráfico podemos observar que a classe **A** apresenta tempos de execução bastante reduzidos, o que torna inviável o seu estudo em paradigmas de versão partilhado/distribuído em que se recorre à paralelização. Já os kernels **B** e **C** possuem os requisitos necessários, apresentando tempos de execução razoáveis, sendo que vão ser estes os tamanhos de dados utilizados ao longo deste estudo.

V. BENCHMARKING NUM AMBIENTE DE TESTES SEQUENCIAL

O primeiro paradigma em questão é o Sequencial. Primeiramente, foi analisada a execução dos três kernels, nas diferentes máquinas.Deste modo realizaram-se testes para as configurações possíveis de hardware e software como se poderá observar nas secções seguintes.

A. Análise dos tempos de execução nos diferentes Processadores

Os vários kernels foram testados em duas máquinas com arquiteturas diferentes para se poder observar a sua influência no tempo de execução deles.

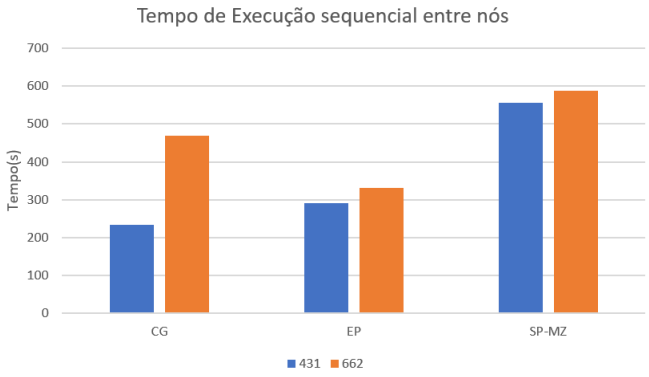

Fig. 2. Tempo de execução da sequencial para as diferentes máquinas utilizadas com o compilador GNU,classe C e otimização O2 .

Através da figura podemos constatar que o tempo de execução na máquina 431 é sempre inferior em relação à 662, para todos os kernels. Isto pode ocorrer devido ao facto da máquina 431 apresentar uma maior frequência de relógio.

B. Análise do Compilador: GNU vs Intel

Nesta secção será possível observar as diferenças entre o compilador da GNU e da Intel.

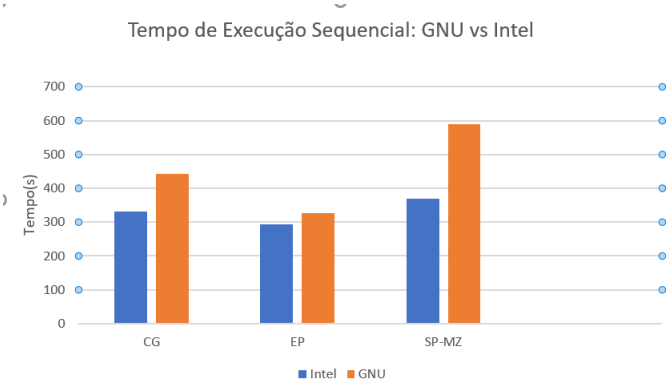

Fig. 3. Tempos de execução sequenciais para os diferentes Kernels,classe C,GNU vs Intel,nivel de otimização O2 e nó 662.

Com a análise dos resultados da figura 3, podemos observar que o compilador da Intel produz melhores resultados para todos os kernels num paradigma de testes sequencial. Estes resultados eram de certa forma espectáveis, visto que ambos os processadores usados (431 e 662) são processadores da família da intel, possuindo características ótimas para este tipo de compiladores.

C. Análise dos níveis de otimização: O2 vs O3

Neste secção pretende-se ver qual a influência dos níveis de otimização,-O2 e -O3, na performance dos diferentes kernels.É,no entanto, possível prever a obtenção de melhores resultados usando o nível de otimização O3.

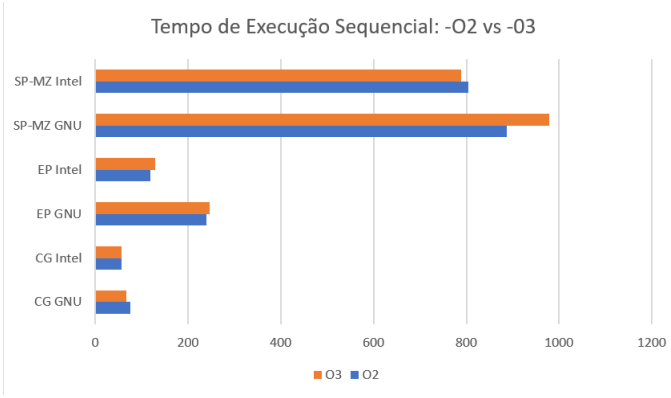

Fig. 4. Tempos de execução sequenciais para os diferentes Kernels,classe C,O2 vs 03, nó 662

O nível O3 tenta auto-vetorizar e/ou fazer loop unrolling o que diminui o número de instruções e conduzindo a uma melhoria da performance. No

entanto, como podemos observar na figura 4 para os kernels EP e SP-MP,por vezes, os custos de usar O3 sobrepõem-se aos ganhos, provocando uma diminuição na performance, devido ao overhead associado às melhorias que a versão O3 tenta aplicar. Nos outros casos, as melhorias não são significativas pelo que nos restantes testes será utilizado o nível de compilação O2.

D. Utilização do CPU

De seguida iremos apresentar a variação na utilização do CPU pelos kernels nos diferentes nodos.

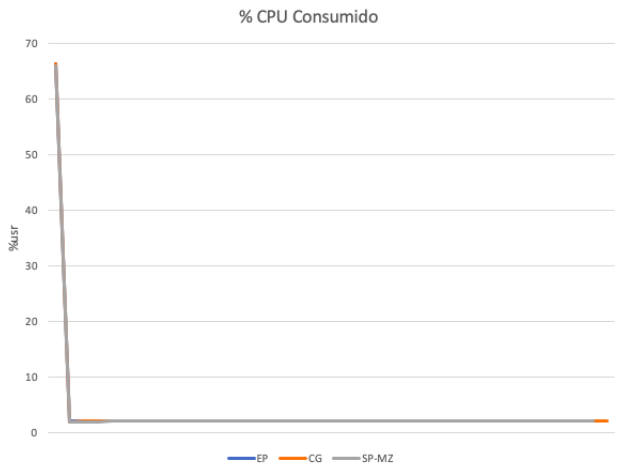

Fig. 5. Comparação dos diferentes kernels, nodo 662

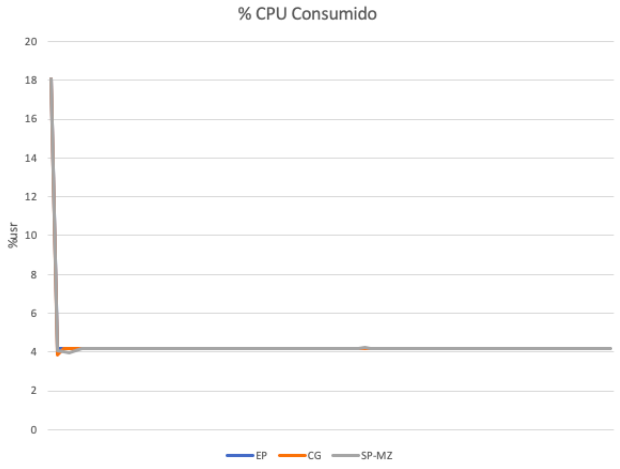

Fig. 6. Comparação dos diferentes kernels, nodo 431

Após a análise dos gráficos podemos concluir que a taxa de utilização do CPU para os diferentes kernels, independentemente do nodo, é relativamente baixa, rondando os 4%. Estes valores indicam que nao se está a tirar o máximo de proveito dos recursos disponíveis.

E. Consumo de Memória

Para além do estudo da utilização do CPU decidiu-se analisar o consumo de memória para os diferentes kernels de modo a corroborar os resultados anteriores.

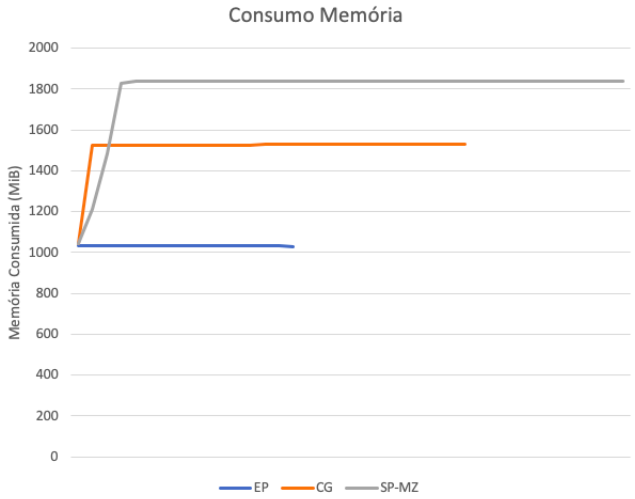

Fig. 7. Comparação dos diferentes kernels, para o compilador da Gnu, nodo 662

Através da observação da figura 7 podemos constatar que pode existir uma relação entre os tempos de execução e o consumo de memória, visto que os kernels com menor tempo de execução têm um menor consumo de memória e vice-versa.

VI. BENCHMARKING NUM AMBIENTE DE MEMÓRIA PARTILHADA: OpenMP

É indiscutível o facto da arquitetura dos sistemas de computação ter vindo a sofrer uma constante evolução ao longo do tempo, sendo uma das evidências a maior capacidade de paralelismo, devido ao aumento de número de cores por processador. Após feita a análise dos vários kerneles em ambiente sequencial iremos, de seguida, estudar o seu comportamento em ambiente de memória partilhada.

A. *Análise dos Tempos de Execução*

Nos seguintes 2 gráficos podemos visualizar a variação dos tempos de execução, nos vários kernels, com o aumento de número de cores, para a classe B, para ambas as máquinas 662 e 431.

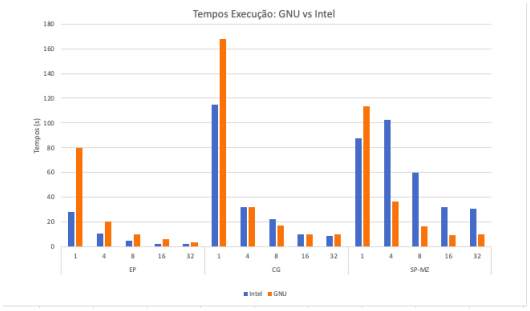

Fig. 8. Tempos de execução dos vários kernels, classe B e nível de otimização O2, nodo 662

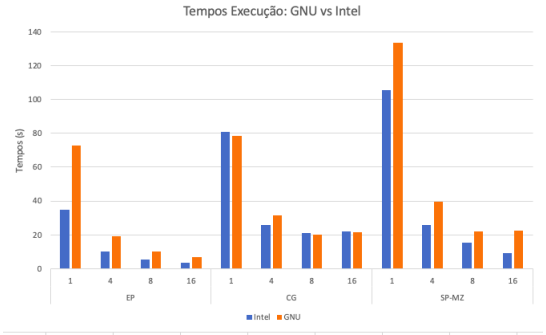

Fig. 9. Tempos de execução dos vários kernels, classe B e nível de otimização O2, nodo 431

Analisando os gráficos anteriores podemos verificar que independentemente do tipo de compilador usado, o tempo de execução diminui com o aumento de cores para ambas as máquinas, sendo que esta diminuição estagna, sensivelemente a partir dos 16 cores para a máquina 662 e 8 cores para a máquina 431, para os kernels Cg e Sp-mz. Esta estagnação pode ter origem no overhead causado na tentativa de paralelização de um tamanho de dados em que não se justifica tal tentativa.

De seguida iremos fazer a mesma análise para a classe C, tentanto encotrar algum padrão de semelhança.

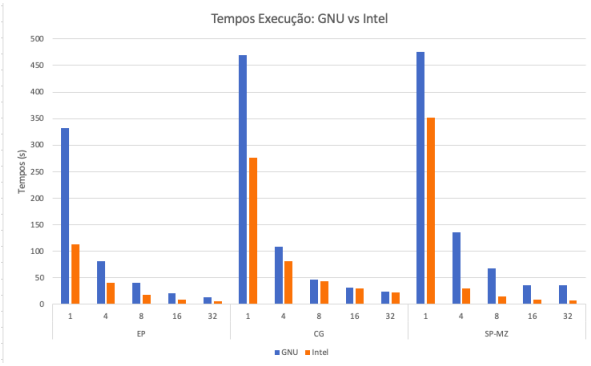

Fig. 10. Tempos de execução dos vários kernels, classe C e nível de otimização O2, nodo 662

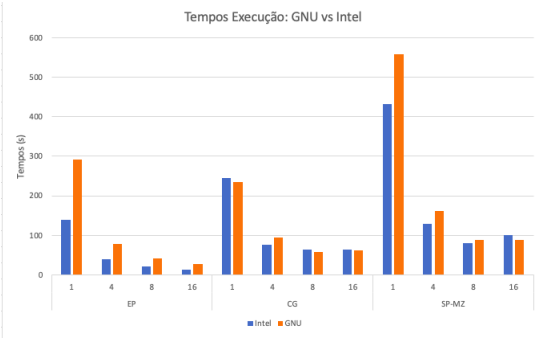

Fig. 11. Tempos de execução dos vários kernels, classe C e nível de otimização O2, nodo 431

Como era esperado, para a classe C o comportamento é semelhante sendo que para a máquina 431 e kernel Sp-mz o tempo de execução com 32 cores é ligeiramente superior do que com 16. Já para o kernel Ep, este apresenta um comportamento favorável ao paralelismo, aparesentando uma diminuição contínua do tempo de execução com o aumento de número de cores, comportamento esse que se pode dizer espectável visto que se trata de um kernel embraçosamente paralelo.

B. *Utilização do CPU*

Tal como foi feito para a versão sequencial, iremos analisar a taxa de utilização do CPU de maneira a poder comparar com os resultados obtidos anteriormente e tentar chegar a algumas conclusões.

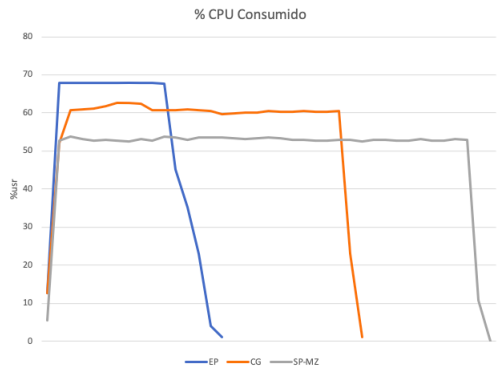

Fig. 12. Utilização de Cpu pelos kernels, com 32 cores, no nodo 662

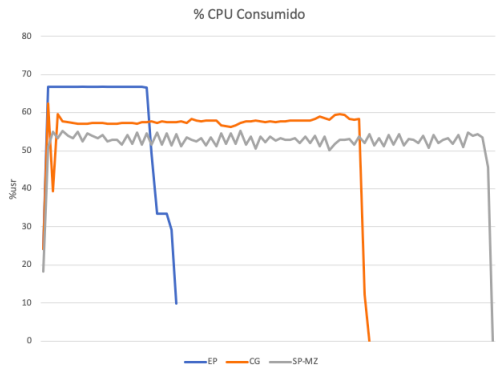

Fig. 13. Utilização de Cpu pelos kernels, com 16 cores, no nodo 431

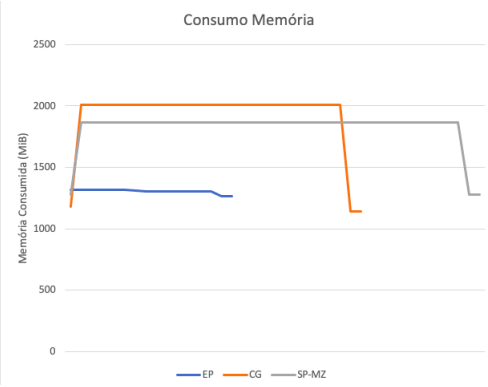

Fig. 14. Utilização de memória pelos kernels, com 32 cores, no nodo 662, classe C

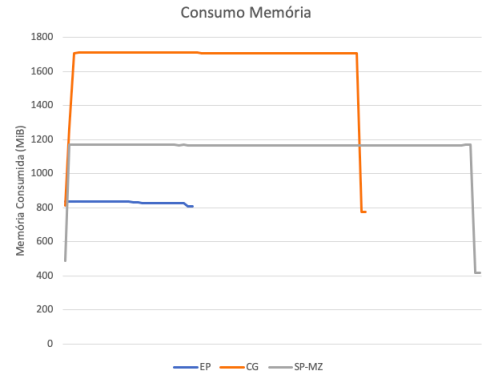

Fig. 15. Utilização de memória pelos kernels, com 16 cores, nodo 431, classe C

Da observação dos gráficos podemos constatar que independentemente da máquina usada existe um padrão na taxa utilização de CPU. É possivel observar claramente, utlizando os dados anteriores, que os kernels que apresentam menor tempo de execução são aqueles que têm uma maior taxa de utilização do CPU e vice-versa. Para além disso é possível concluir que as taxas de utilização de CPU sao as mesmas para as 2 máquinas.

C. Consumo de Memória

Verificou-se, anteriormente, que em ambiente sequencial, em termos de consumo de memória, os kernels que apresentam menores tempos de execução são aqueles que têm menor consumo e vice-versa. Assim, para testar esta hipótese também em memória partilhada, após vários testes obtiveram-se os resultados apresentados nos seguintes gráficos.

A principal diferença em relação aos resultados obtidos num ambiente sequencial é a alteraçao no consumo de memória por parte dos kernels Cg e Sp-mz, sendo que neste caso, independentemente da máquina, o kernel CG tem um maior consumo de memória em relação ao Sp-mz, o que nao acontecia num ambiente sequencial. Isto pode ocorrer talvez pela maneira como cada kernel está implementado e aos dados necessários buscar à memória em cada instante de tempo.

D. Escalabilidade

Para além dos fatores analisados anteriormente e , sendo este um ambiente de memória partillhada, é imprescindível analisar a escalabilidade dos kernels com o aumento do número de cores. Observamos anteriormente que os tempos de execução diminuiam consoante o aumento do número de cores até um certo ponto e esta diminuição diferia de kernel para kernel. De seguida apresentamos os

gráficos com os ganhos das implementações em relação à versão sequencial, para a classe B.

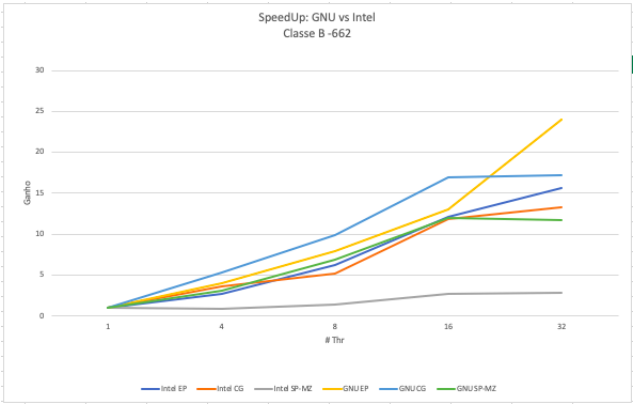

Fig. 16. SpeedUp obtido pelos kernels no nodo 662, para a classe B

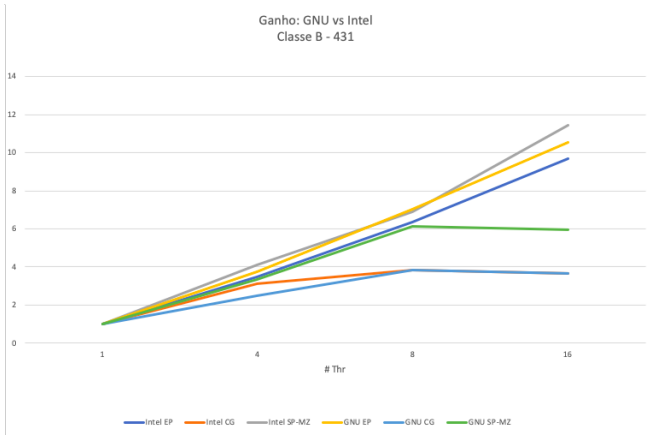

Fig. 17. SpeedUp obtido pelos kernels no nodo 431, para a classe B

Da análise destes 2 gráficos, existem algumas características que podemos de imediato realçar. A primeira e a mais óbvia é o facto do kernel **Ep** ter a melhor escalabilidade em relação aos outros, com o compilador da Gnu, apesar de apresentar também boa escalabilidade com o compilador da Intel. Para além disso outra característica interessante é o facto do kernel **Sp-mz**, usando o compilador da Intel, ter a pior escalabilidade na máquina 662 mas, por outro lado, na máquina 431 ter a melhor escalabilidade, apresentando ganhos sucessivamente maiores em relação aos outros kernels.

Em relação ao kernel **CG** podemos observar que para a máquina 662 apresentar uma boa escalabildiade até os 16 cores, sendo que a partir disso estagna e para a máquina 431 o mesmo acontece até aos 8 cores, isto com o compilador da Gnu. O mesmo kernel, com o compilador da Intel apresenta uma escalabilidade irregular, apresentando um ganho máximo para 32 cores, na máquina 662 e para 8 cores, na máquina 431.

De seguida iremos apresentar os resultados obtidos para a classe C.

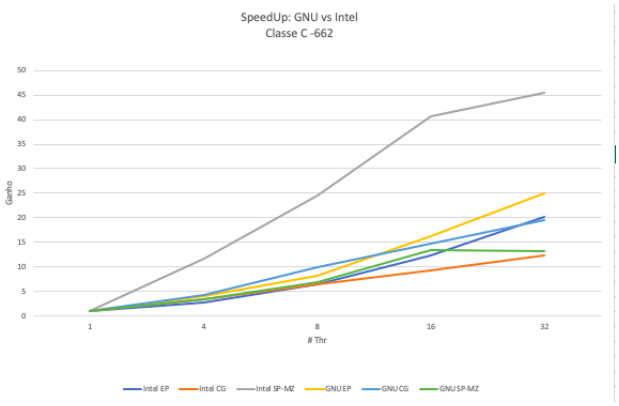

Fig. 18. SpeedUp obtido pelos kernels no nodo 662, para a classe C

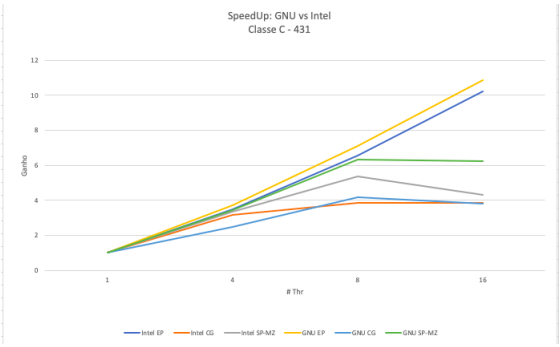

Fig. 19. SpeedUp obtido pelos kernels no nodo 431, para a classe C

Podemos observar que existem inúmeras semelhanças com a classe B, sendo que a que se mais destaca é a escalabilidade do kernel Sp-mz na máquina 662, com o compilador da Intel. Todas as observações feitas na classe B são também aplicáveis para a classe C.

Assim, podemos concluir que independentemente do tamanho de dados e da máquina onde os testes são realizados, o kernel **Ep** é o que, maioritariamente, apresenta melhor escalabilidade, apesar desta nao ser proporcional ao número de cores.

Em relação ao kernel Sp-mz podemos constatar que existe uma irregularidade na sua escalabilidade quando comparado nas 2 máquinas e nas 2 classes de dados. Para um tamanho de dados maior, este apresenta uma melhor escalabilidade na máquina 662, enquanto que para dados mais pequenos, a máquina 431 é aquela apresenta melhores resultados.

Já para o kernel **Cg**, podemos concluir que é um kernel que apresenta escalabilidade bastante fraca, apresentando os melhores resultados para a classe B, na máquina 662.

VII. BENCHMARKING NUM AMBIENTE DE MEMÓRIA DISTRIBUÍDA: MPI

Após a análise do comportamento nos paradigmas sequencial e partilhado nesta secção irá ser estudado o paradigma de memória distribuída. Tem várias vantagens em relação ao paradigma anterior destacando-se a exclusão de *race conditions*. O principal obstáculo será distribuir os dados e tarefas pelos processadores. Iremos analisar os tempos de execução assim como a utilização do CPU, da memória e da rede de modo a perceber a influência da distribuição da carga na computação.Para a realização do estudo, foram utilizados quatro cores das máquinas 662, que possuem as duas tecnologias de rede distintas - ethernet e myrinet. O mapeamento dos processos pelos processadores é realizado por nodo, especificado pela opção -map-by-node,fornecida pelo mpirun.

A. Análise dos Tempos de Execução: Gigabit Ethernet vs. Myrinet

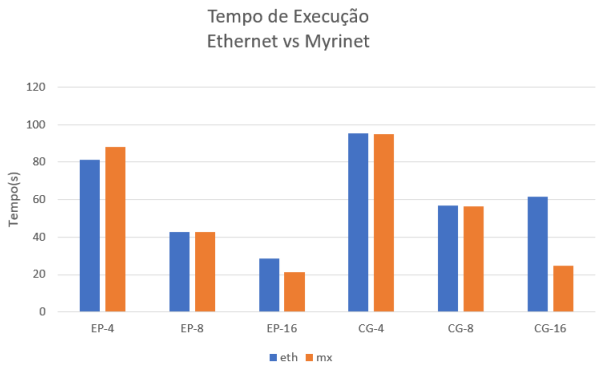

Fig. 20. Tempos de execução dos vários kernels,classe C e nível de otimização O2, utilizando Myrinet e Gigabit ethernet na máquina 431

Como foi dito anteriormente, os nós 431 estão conectados por Gigabit Ethernet(1Gps) e Myrinet(10Gbps,baixa latência).Deste modo é de prever que os testes que utilizam Myrinet obtenham melhores resultados pois a sua largura de banda é dez vezes superior à da Ethernet. É,então, possível estabelecer como limite teórico ganhos na performance dos kernels de 10 vezes. Como se pode observar nos resultados da figura, a tecnologia de comunicação myrinet apresenta,em geral, melhores resultados, sendo esta diferença mais vísivel no kernel cg, ainda que pequena.

B. Análise dos Tempos de Execução: Sequencial vs. OpenMP vs. MPI

Relativamente a um programa com memória partilhada, num ambiente de execução em memória distribuída a performance de um programa é afetada não só pela distribuição de carga pelos processadores, mas também pela quantidade de comunicação necessária entre os processos.Portanto, é expectável obter piores resultados em relação às implementações com memória partilhada.

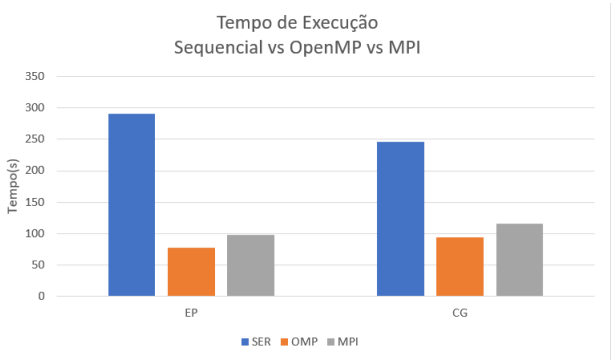

Fig. 21. Tempos de Execução: Sequencial vs. OpenMP vs. MPI na máquina 431

Na figura 21 podemos observar os resultados obtidos para as soluções em MPI. Tal como era de prever obteve-se piores resultados do que as suas implementações em OpenMP.Isto acontece devido aos custos de comunicação entre processos, afetando a performance do programa. Quanto maior é a troca de informação entre processos para os kernels, maior será a discrepância em relação às implementações com memória partilhada.

C. Utilização de Rede

Para além de analisar a influência nas comunicações entre processos, é essencial estudar a necessidade de comunicação para cada kernel pois a mesma vem com elevados custos associados. Deste modo, o tráfego de rede foi analisado durante a execução de todos os kernels. Em baixo, são apresentados os resultados obtidos.

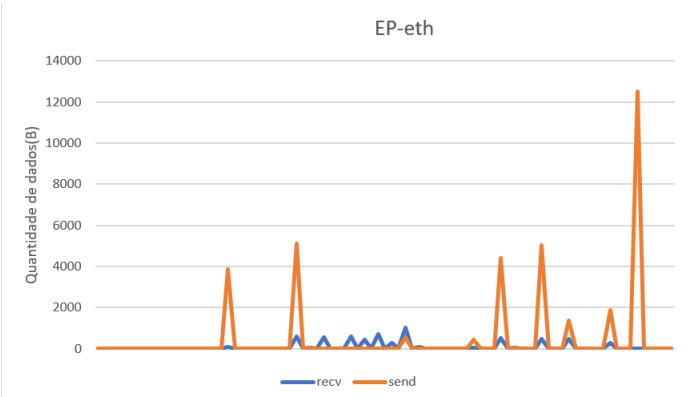

Fig. 22. Utilização de rede pelo kernel EP ,classe C,otimização O2 no nó 431

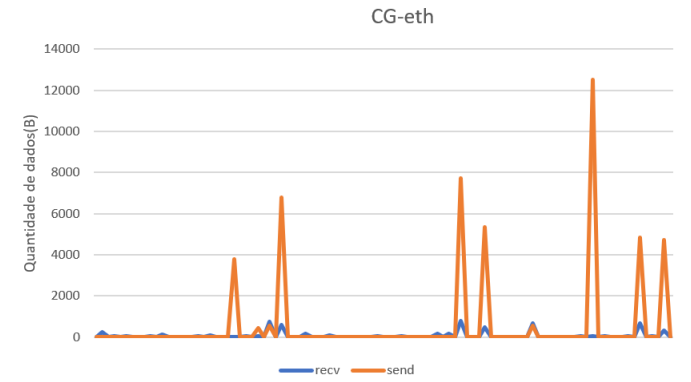

Fig. 23. Utilização de rede pelo kernel CG ,classe C,otimização O2 no nó 431

Observando as várias figuras apresentadas anteriormente, podemos constatar que o kernel CG é o que utiliza mais a rede apesar de as suas diferenças com o kernel EP serem pouco vísiveis, sendo,que este último utiliza menos rede devido ao facto de a comunicação ser feita no início da execução e,portanto,as diferenças no desempenho são ainda menos vísiveis.

D. Utilização do CPU

Em termos de desempenho,as implementações MPI não trouxeram vantagens,portanto, decidimos analisar a utilização do CPU do sistema no geral relativamente às versões OpenMP. Para todos os kernels os resultados obtidos foram semelhantes, fazendo as implementações em OpenMP uma maior utilização do CPU em oposição às versões em MPI que é mais relaxada.

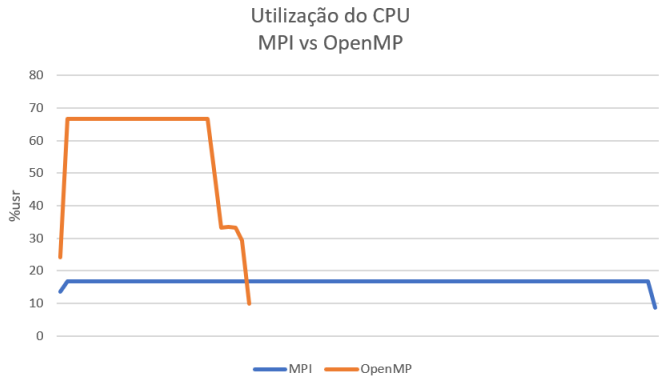

Fig. 24. Utilização do CPU no kernel EP,classe C,otimização O2 versão OMP e mpi no nó 431

Através da figura anterior podemos concluir que apesar deste tipo de implementações com memória distribuída apresentar piores resultado, a nível do desempenho de aplicações as implementações MPI sobrecarregam menos o sistema. No entanto, isto também poderá significar que os recursos não estão a ser aproveitados ao máximo, sendo portanto interessante analisar o desempenho de uma implementação híbrida.

E. Consumo de Memória

Relativamente ao consumo de memória do sistema, é expectável que as versões MPI utilizem-na menos uma vez que a informação é distribuída pelos processadores, onde cada processo faz um uso mais eficiente do seu espaço de memória privado.

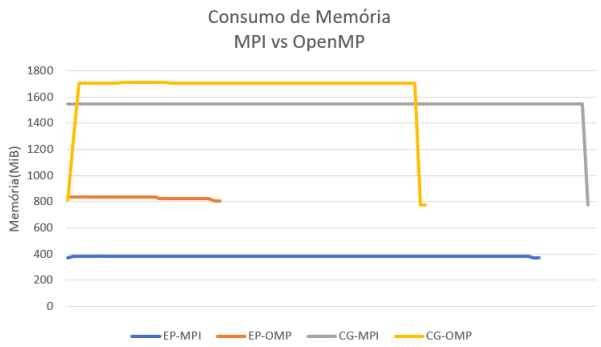

Fig. 25. Consumo de Memória nos 3 kernels ,classe C,otimização O2 versão OMP e mpi no nó 431

Através da análise da figura 25 pode constatar-se que, para cada kernel, os consumos de memória das implementações em OpenMP são superiores relativamente às respetivas versões MPI, para um mesmo nodo computacional. Tal facto deve-se ao modelo de memória partilhada utilizar o mesmo espaço de endereçamento, para os fios de execução, enquanto que nos modelos MPI, a memória é distribuída pelos processadores.Adicionalmente, em memória distribuída, também é permitido utilizar vários nodos computacionais.No entanto, os resultados são para um nodo, combinado com a utilização de memória nos quatro cores que permite a obtenção de melhores resultados.

VIII. Benchmarking num Ambiente de Memória Híbrido: OpenMP + MPI

Atualmente, os clusters apresentam uma grande quantidade de nós, alguns com vários cores.

O modelo híbrido pode permitir a redução dos custos de comunicação entre os processos e ao mesmo tempo tirar partido dos vários cores disponíveis nos nós.

Ao longo desta secção são apresentados os resultados obtidos na comparação das versões do kernel SP-MZ em paradigma de memória partilhada e em paradigma híbrido.

A. Tempo de Execução

De maneira a comparar os tempos de execução foi elaborado o seguinte gráfico com os dados repetivos a cada implementação

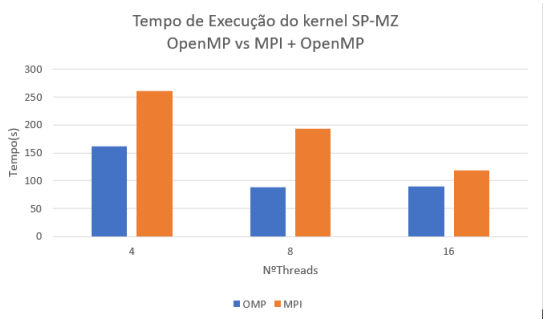

Fig. 26. Tempos de execução dos kernels SP-MZ compilador GNU, classe C,nível de otimização O2 versão OMP e versão híbrida na máquina 431

É possível verificar que apesar da dimuição do tempo de execução com aumento do número de threads, a implementação híbrida tem sempre piores tempos de execução relativamente à implementação em memória partilhada. De maneira a perceber o motivo desta degradação é ,de seguida, medido as taxas de utilização de CPU e consumo de memória.

B. Utilização do CPU

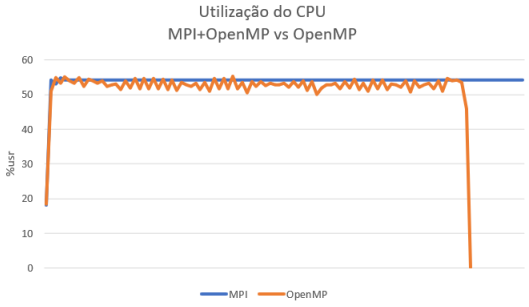

Fig. 27. Consumo de Memória dos kernels SP-MZ,compilador GNU, classe C,nível de otimização O2 versão OMP e híbrida no nó 431

Através da observação do gráfico podemos constatar que a taxa de utiliação de CPU pelo kernel Sp-mz nos diferentes paradigmas é praticamente o mesmo, cerca de 55%, facto que não permite responder à diferença nos tempos de execução.

C. Consumo de Memória

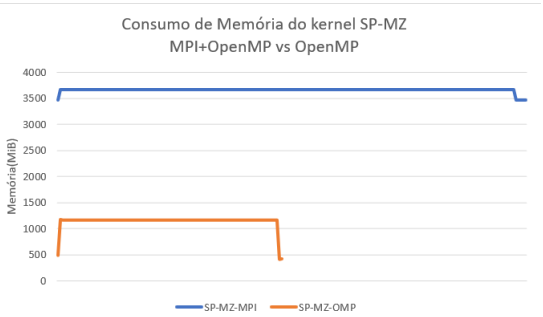

Fig. 28. Consumo de Memória dos kernels SP-MZ,compilador GNU, classe C,nível de otimização O2 versão OMP e híbrida no nó 431

Em relação ao consumo de memória podemos observar que a implementação hībrida utiliza, sensivelmente, cerca de 3x mais memória que a implementação em memória partilhada. Por um lado pode indicar um bom aproveitamento dos recursos disponíveis, mas por outro pode ser o motivo pela degradação na performance visto que os acessos à RAM têm um custo associado.

IX. CONCLUSÕES

Ao longo do desenvolvimento deste documento reparamos que devemos ter uma visão global do sistema sobre o qual uma aplicação irá ser executada tal como as características do hardware das diversas máquinas disponíveis bem como o ambiente geral de execução.

Este trabalho permitiu-nos ter uma melhor perceção do melhor compilador a utilizar, dos melhores níveis de otimização, de que maneira diferentes tecnologias de rede afetam o tempo de execução de um dado programa.Com isto, é possível concluir que devem ser testadas todas as alternativas referidas anteriormente de modo a encontrar as mais eficientes.

Tendo em conta a quantidade de testes e submissões de trabalho para retirar todos os dados necessários, a utilização de Linguagens como *Shell Script* permitiu agilizar o processo na obtenção desses mesmos dados.

Por fim, o uso exaustivo do SeARCH permitiu um melhor conhecimento da sua estrutura e organização que poderá ser útil em trabalhos futuros.

REFERENCES

[1] Cpu-world.com. 2020. Intel Xeon E5-2695 V2 - CM8063501288706 / BX80635E52695V2. [online] Available at: <http://www.cpu-world.com/CPUs/Xeon/Intel-Xeon%20E5-2695%20v2.html > [Accessed 15 April 2020].

[2] Search6.di.uminho.pt. 2020. Search | Services And Advanced Research Computing With HTC/HPC Clusters. [online] Available at: <http://search6.di.uminho.pt/wordpress/?page_id=55 > [Accessed 17 April 2020].