

**Mestrado em Engenharia Informática**

*Engenharia dos Sistemas de Computação - 2014/2015*

NAS Parallel Benchmarks (NPB)

24 de Março de 2015

**Resumo**

Este trabalho demonstra o resultado do Benchmark no nó compute-431-6 do SeARCH no Departamento de Informática da Universidade do Minho. Desta forma será possível analisar os recursos disponíveis com os testes intensivos que o NPB providencia.

**Fábio Gomes** pg27752

# **Índice**

[**Índice** 2](#_Toc414834399)

[Introdução 3](#_Toc414834400)

[NAS Parallel Benchmarks (NPB) 4](#_Toc414834401)

[Caracterização do Sistema 5](#_Toc414834402)

[Compilação, Versões e Parâmetros 6](#_Toc414834403)

[gcc / mpicc / gfortran 6](#_Toc414834404)

[Icc / ifort 6](#_Toc414834405)

[Serial 6](#_Toc414834406)

[OpenMP 6](#_Toc414834407)

[MPI 6](#_Toc414834408)

[Benchmark EP 7](#_Toc414834409)

[*Análise da Duração* 7](#_Toc414834410)

[*Análise da Memória* 7](#_Toc414834411)

[*Análise da Rede* 8](#_Toc414834412)

[*Análise de Acessos ao Disco* 8](#_Toc414834413)

[*Conclusão* 8](#_Toc414834414)

[Análise Final de Resultados 9](#_Toc414834415)

[Conclusão 10](#_Toc414834416)

# Introdução

O problema que nos foi apresentado consiste em analisar a performance de um sistema de forma a poder saber as capacidades e limites operacionais para as aplicações que irão ser desenvolvidas para aquela infraestrutura.

Foi proposto que os testes incidissem em diferentes classes com diferentes compiladores, como o *icc* e *gcc* (versão 4.4.6 e 4.9.0), número de processos (*MPI*) e threads (*OpenMP*) e ainda para as versões SERIAL, OpenMP e MPI para cada Benchmark.

Os dados que o NPB devolve não são suficientes e para tal foi necessário recorrer a ferramentas externas para obter informações extras ao longo do tempo de execução. Especificamente o derivado *dstat* para a alocação de memória, acessos ao disco e dados enviados/recebidos na rede; o *mpstat* para a utilização dos diferentes processadores.

Com estes dados recolhidos foi possível gerar gráficos para comparar as diferentes combinações de compiladores e número de processos e threads.

# NAS Parallel Benchmarks (NPB)

São um pequeno conjunto de programas destinados a ajudar a avaliar o desempenho dos supercomputadores paralelos. Os *Benchmarks* são derivadas de aplicações da dinâmica de fluido computacional (CFD) e consistem em cinco *kernels* e três *pseudo-aplicações* (NPB 1). O pacote de Benchmark foi estendido para incluir novos pontos de referência para malha adaptativa não-estruturada, I/O, aplicações multi-zona, e grelhas computacionais paralelas. Os tamanhos dos problemas no NPB são predefinidos e indicados com diferentes classes (A,B,C,D,E,F ou S,W). Implementações de referência de NPB estão disponíveis em modelos de programação mais usadas como MPI e OpenMP (NPB 2 e NPB 3), sendo estas as versões que mais nos interessam.

Há diferentes especificações de Benchmark:

* 5 kernels
  + IS - Integer Sort, random memory access
  + EP - Embarrassingly Parallel
  + CG - Conjugate Gradient, irregular memory access and communication
  + MG - Multi-Grid on a sequence of meshes, long- and short-distance communication, memory intensive
  + FT - discrete 3D fast Fourier Transform, all-to-all communication
* 3 pseudo-aplicações
  + BT - Block Tri-diagonal solver
  + SP - Scalar Penta-diagonal solver
  + LU - Lower-Upper Gauss-Seidel solver

E classes de tamanho:

* S: small for quick test purposes
* W: workstation size (a 90's workstation; now likely too small)
* A, B, C: standard test problems; ~4X size increase going from one class to the next
* D, E, F: large test problems; ~16X size increase from each of the previous classes

Optei pela *EP*, *FT*, *IS* e *SP* como foco para os testes com classes A e B em cada.

# Caracterização do Sistema

Para utilização deste projeto tive que recorrer ao nó 431-6 do SeARCH, recorrendo a jobs específicos para ele, com a seguinte especificação:

|  |  |
| --- | --- |
|  | **Cluster Node** |
| **Manufacturer** | intel |
| **Model** | X5650 |
| **Clock speed** | 2.67 GHz |
| **Architecture** | x86-64 |
| **#Cores** | 12 |
| **#Threads per Core** | 2 |
| **Total Threads** | 24 |

# 

# Compilação, Versões e Parâmetros

Para este trabalho tive que utilizar diferentes versões de compiladores e ferramentas.

## gcc / mpicc / gfortran

4.4.6 e 4.9.0

## Icc / ifort

13.0.1

Para cada teste de Benchmark foi necessário usar estas 3 versões:

## Serial

Versão base em que tipicamente é utilizado um processador.

## OpenMP

Utiliza um número especificado de threads, neste caso 2, 8, 16 e 48.

## MPI

Com o número providenciado de processos, 2, 4, 8 e 16 (no caso do SP 4, 9 e 16).

# Benchmark EP

Há 2 classes, A e B, que estão representadas. A verde são marcados os tempos melhores entre o mesmo teste comparando com as 3 versões de compiladores e a vermelho o pior.

## *Análise da Duração*

A seguinte tabela inclui os tempos de execução e os *Mop/s* (Milhões de Operações por segundo) para cada combinação de Teste.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **ep** | | | | | | | | | | | |
|  |  | SER | | MPI | | | | | | | | OMP | |
|  |  | 2 Procs | | 4 Procs | | 8 Procs | | 16 Procs | |
|  |  | s | Mop/s | s | Mop/s | s | Mop/s | s | Mop/s | s | Mop/s | s | Mop/s |
| **gcc 4.4.6** | A | 19,4 | 27,7 | 8,59 | 62,5 | 5,36 | 100 | 2,27 | 237 | 1,15 | 467 |  |  |
| **gcc 4.9.0** | 20,3 | 26,5 | 8,88 | 60,4 | 4,47 | 120 | 2,44 | 220 | 1,21 | 442 |  |  |
| **icc 13.0.1** | 9,36 | 57,4 | 4,14 | 130 | 2,56 | 210 | 1,19 | 453 | 0,64 | 835 |  |  |
| **gcc 4.4.6** | B | 77,2 | 27,8 | 34,3 | 62,6 | 17,3 | 124 | 9,29 | 231 | 4,59 | 468 |  |  |
| **gcc 4.9.0** | 79,3 | 27,1 | 34,9 | 61,5 | 22,5 | 95,3 | 9,8 | 219 | 5,01 | 428 |  |  |
| **icc 13.0.1** | 37,4 | 57,4 | 15,9 | 135 | 10,2 | 210 | 5,23 | 411 | 2,2 | 975 |  |  |

É possível distinguir que o icc foi o melhor compilador porque obteve 100 % dos tempos melhores. Sendo assim foi utilizado o icc como base dos testes seguintes.

Este gráfico comparativo em que temos os 5 testes para o EP com a Classe A, não é muito interessante pois fica apenas claro quais são as combinações mais rápidas. Assim utilizando a soma acumulada, ou apenas individual, das percentagens de utilização por processador dá uma melhor perceção dos resultados. Visto o mpstat ter monitorizado durante os jobs 48 processadores o máximo de performance será 48\*100% = 4800%. Os gráficos seguintes são alguns exemplos.

Como a versão Serial só utiliza um processador a utilização individual será de 100 % (como a acumulada).

Com 2 processos, a utilização acumulada é de 200 % como previsto.

Com 8 processos a utilização bate nos 800 %.

## *Análise da Memória*

Os gráficos seguintes demonstram a alocação de Memória ao longo do tempo de execução.

Classe A:

Classe B:

Como expectável a memória necessária para o funcionamento dos programas aumenta com o número de paralelismo. Cabendo a nós, programadores, decidir qual o melhor balanço entre duração e memória disponível.

## *Análise da Rede*

Os gráficos são muito diferentes e não é fácil fazer uma comparação entre eles pois também os valores ora são na ordem das centenas de milhar como passam para as centenas como estão a zero. De notar que o gráfico para o Serial da Classe A está com eixo dos Bytes com escala logarítmica de base 10.

## *Análise de Acessos ao Disco*

Os acessos ao Disco são muito pouco frequentes, acontecendo casos pontuais não dignos de registo. Os valores foram todos registados mas não projetados em gráfico, não tendo uma opinião concreta sobre eles.

## *Conclusão*

O icc neste teste foi o que obteve melhores resultados.

# Análise Final de Resultados

Com tudo concluído, é tempo de fazer as comparações das diferentes métricas e analisar os resultados. O icc é o melhor !

# Conclusão

Para além de ser um caso prático e cativar por si só foi também um trabalho muito bom na medida em que permitiu usar novos conhecimentos adquiridos nesta Unidade Curricular que não foram abordados anteriormente. Um trabalho que apesar de parecer simples engloba em si vários aspectos que tiveram de ser devidamente considerados para que o algoritmo funcionasse corretamente.