APLICAÇÃO DE BENCHMARK PARA AVALIAÇÃO DE CLUSTER IMPLEMENTADOS COM CONTROLADORES DE BAIXO CUSTO

Gulliver Tadra Waldmann (PIBIC/FA/UEPG), Dierone Cesar Foltran Junior (Orientador), gullivertadra@gmail.com: foltran@uepg.br.

Universidade Estadual de Ponta Grossa/Departamento de Informática/Setor de Ciências Agrárias e de Tecnologias.

Palavras-chave: MPI, HPL, Beowulf Cluster, Sysbench.

Resumo

O *cluster* de baixo custo e consumo, montado e avaliado nesta pesquisa, utiliza-se de microprocessadores de baixo custo como os Raspberry Pi 3 Model B e um Raspberry Pi 2 Model B, em uma formação chamada *Cluster Beowulf* [2], que consiste em que um dos microprocessadores funcione como um *master*, que, no caso, é o Raspberry Pi 2, comandando e recebendo dos outros microprocessadores as aplicações executadas e são chamados de *slaves*. Assim podendo distribuir uma única aplicação em um ou mais *slaves*, e logo fica visível o paralelismo de um *cluster*. Além disso, tem-se uma o funcionamento e comparação do poder computacional de processamento dos microprocessadores fortemente acoplados e fracamente acoplados. Ao rodar os *benchmarks* escolhidos, em um único *slave*, com o *benchmark Sysbench* podemos ver o desempenho do seu processamento com o microprocessador fortemente acoplado, além disso vemos que seu poder computacional comparado com o do *cluster* no *benchmark* HPL (*High Perfomance Linpack*) [6] é maior devido as limitações ocorridas pelo *cluster*, mesmo com um desempenho mais baixo, pode-se visualizar que o c*luster* é um ótimo meio de mostrar como funciona o paralelismo de computadores.

Introdução

Após o lançamento do primeiro Raspberry Pi em 2012, várias pesquisas foram feitas utilizando-se desse novo microprocessador que parecia extremamente promissor pelo seu tamanho, e o seu baixo custo financeiro e energético, assim podendo explorar as várias possibilidades que poderiam ser feitas utilizando-o, e com o tempo vieram versões novas e melhoradas do mesmo. Em uma destas pesquisas realizadas, foi colocado dois ou mais Raspberrys para que executassem uma única tarefa em paralelo, no intuito de reduzir o tempo total de processamento da aplicação. Também conhecidos como *cluster*. Essa comunicação em paralelo é feita em forma de nós (*nodes*), que são comunicados através de uma rede local LAN. (LIMA; MORENO; DIAS, 2016)

Em específico nesta pesquisa, foi montado um *cluster* baseado na formação *Cluster* Beowulf, onde um dos microprocessadores funciona como nó *master* (mestre), sendo responsável pelo envio e coleta das aplicações enviadas para os outros nós chamados de *slaves* (escravos), que fazem todo o trabalho computacional das aplicações e enviam através de mensagens e sinalizações os resultados de volta para o nó mestre, assim um *master* pode ter vários *slaves*, porém um *slave* pode ter apenas um nó *master*. Neste conceito podemos visualizar o comportamento de fracamente acoplado onde são vários sistemas independentes conectados e comunicantes entre si, porém cada processador tem os seus núcleos em um único sistema operacional, revelando sua natureza de fortemente acoplado.

Material e Métodos

Para os testes, foi montado o *cluster*, seguindo o modelo Beowulf. Ao montar o *cluster*, foi utilizado um Raspbery Pi 2 como mestre, no entanto por mais que ele seja de um modelo mais antigo, nenhum processo será rodado nele, logo não interferindo nos resultados dos *benchmarks*. Os quatros *slaves* então foram utilizados como Raspberry Pi 3, conectados por uma rede LAN, utilizando-se de um *Switch* de 100 Mbps.

Tendo em vista que o hardware dos Raspberry Pi 3 Model B, responsável pelo processamento das aplicações consistem em um CPU Quad-core 1.2GHz Broadcom BCM2837 64bit, 1 GB de RAM, placas de rede wifi e Bluetooth, comunicação Ethernet, 4 portas USB, entrada HDMI e entrada micro USB para alimentação aguentando até 2.5 A. Sendo o chip BCM2837 uma arquitetura ARM Cortex A53 (ARMv8), ele é 50% mais rápido que os últimos modelos do seu antecessor Raspberry Pi 2.

No nó mestre foi instalado uma imagem do Raspbian [8], com uma interface gráfica, e nos nós restantes foi utilizado a imagem Raspbian Lite, sem interface, assim diminuindo o número de processos que rodam na máquina, e melhorando o desempenho do *cluster*.

Para que os Raspberry pudessem se comunicar paralelamente, foi habilitado o SSH, e configurando o IP único a cada máquina, foi instalado a biblioteca auxiliar para a comunicação em paralelo MPI, possibilitando então a comunicação entres os *slaves* e *master*.

Os *benchmarks* selecionados para o teste, foi o Sysbench e HPL (*High Perfomance Linpack*), cada um no intuito de medir e avaliar o desempenho da CPU do Raspberry e do *cluster*.

Resultados e Discussão

Ao rodar o *benchmark* Sysbench para teste e avaliação da CPU do Raspberry ou de um único nó, o algoritmo é *multi-thread*, ou seja, ele pode ser dividido em sub-problemas e distribuídos nos diferentes núcleos da única pastilha da CPU, então melhorando a perfomance total do programa. Então realizando os testes elevando o número de *threads* e a medida dos valores N, levamos em consideração a resolução de números de 5000 e 20.000 variáveis.

Tabela 1 – Intervalos de Confiança de 95% para N = 5000

Nº de Threads	Tempo Médio de Execução (em segudos)	Intervalo de Confiança
1	52,3625	(52,3469 - 52,3781)
2	26,2245	(26,1866 - 26,2623)
3	17,5535	(17,5371 - 17,5698)
4	13,1655	(13,1612 - 13,1697)

Nos primeiros exemplos com N igual a 5000, vemos que a diferença de desempenho conforme o número de núcleos em execução, o tempo de processamento diminui em 74,8%, ficando evidente como execuções de *multi-thread* elevam o potencial computacional de uma CPU.

ISSN 1676-0093

Tabela 2 – Intervalos de Confiança de 95% para N = 20000

Nº de Threads	Tempo Médio de Execução (em segudos)	Intervalo de Confiança
1	369,2003	(369,1898 - 369,2108)
2	184,6146	(184,6001 - 184,6292)
3	123,5319	(123,5074 - 123,5564)
4	97,4636	(95,0073 - 99,9198)

Para N igual a 20000, obteve-se uma melhora de 74,4% então demonstrando como o processador utilizando-se dos seus quatro núcleos, pode melhorar o desempenho de um programa em aproximadamente 74%. Vale notar que, na execução com quatro *threads* teve um intervalo de confiança maior, justamente que ao exigir mais do processador e sua temperatura aumentar, logo por questão de arquitetura, o Raspberry limitou a capacidade de processamento e logo atrasando o tempo de execução do processo, fazendo assim com que varia a cada execução.

Rodando o *benchmark* HPL, que exibe o processamento em Gflops/s e o tempo gasto para rodar a tarefa, podemos observar como se comporta o desempenho do *cluster* conforme aumentamos o número N que é o número de variáveis do sistema linear, juntamente quando aumentamos o número de nós.

Tabela 3 – Intervalos de Confiança de 95% no HPL para N = 1280

N° de nó	Tempo (em segundos)	Intervalo de Confiança
1	0,662	(0,643 - 0,681)
2	1,978	(1,915 - 2,040)
3	3,496	(3,464 - 3,528)
4	2,094	(2,041 - 2,146)

HPL - High Perfomance Linpack

De acordo com os valores obtidos com um tamanho de N = 1280, podemos observar que o *cluster* houve uma redução no desempenho e tempo de execução do *benchmark*, sendo que uma estimativa de redução no tempo de execução era esperada, no entanto, isso pode-se ser explicado por questões de limitação do *cluster* e sua comunicação em paralelo.

Tabela 4 – Intervalos de Confiança de 95% no HPL para N = 2560

N° de nó	Tempo (em segundos)	Intervalo de Confiança
1	4,088	(4,061 - 4,116)
2	6,93	(6,800 - 7,059)
3	11,1	(10,999 - 11,200)
4	6,352	(6,278 - 6,452)

HPL - High Perfomance Linpack

Com um tamanho de N maior, podemos visualizar um comportamento melhor do *cluster*, tendo uma melhora significativa de 163% de quantidade de operações realizadas por segundo em 4 nós rodando ao mesmo tempo, isso podemos relacionar diretamente que com este tamanho de N, a CPU é capaz de trabalhar

muito mais em cima da aplicação, do que na comunicação em paralelo, e isso podemos observar também no tempo de execução do mesmo.

Conclusão/ões

Em virtude dos dados mencionados no *benchmark* Sysbench, pode-se constatar como o desempenho de um processador fortemente acoplado é diretamente influenciado pela quantidade de carga colocada no processador, uma vez que trabalhando com *threads*, os núcleos no processador são colocados para rodar. Seu desempenho computacional no tempo de execução melhora significativamente com o aumento de *threads*, consequentemente seu consumo de energia também aumenta.

Já trabalhando com o HPL, o desempenho de um único nó continua melhor que o do *cluster,* uma vez que seu desempenho é fortemente acoplado, quando analisamos o desempenho do *cluster* com dois ou mais nós, temos que levar em consideração, como a biblioteca MPI e hardware externo, com cabos de rede e switches, limitam ainda mais o desempenho computacional do *cluster*.

Conclui-se que o desempenho computacional do *cluster* poderia ser melhorado com hardwares mais adequados que não o limitassem tanto, ou até mesmo que aplicações ou *benchmarks* rodados em paralelo utilizassem de *threads* na execução de suas rotinas.

Além do fato, o *cluster* montado pode ser utilizado para ensino de distribuição de cargas, podendo funcionar como um balanceamento móvel de aplicações pesadas e repetitivas.

Agradecimentos

Devo agradecer, primeiramente, à Fundação Araucária pela bolsa concedida durante a realização da pesquisa, depois meu professor orientador por ter me ensinado muito e orientado durante toda a pesquisa e meus amigos e familiares pelo suporte.

Referências

[1] LIMA, F. A.; MORENO, E. D.; DIAS, W. R. A. Performance Analysis of a Low Cost Cluster with Parallel Applications and ARM Processors. **IEEE Latin America Transactions**, v. 14, n. 11, p. 4591-4596, 2016.

[2] COX, S. J.; COX, J. T.; BOARDMAN, R. P.; JOHNSTON, S. J.; SCOTT, M.; O' BRIEN, N. S. Iridis-pi: a low-cost, compact demonstration cluster. **Cluster Computing**, v. 17, i. 2, p. 349-358, 2014.

[3] MPI. Disponível em: <http://www.open-mpi.org/>. Acessado em 23 de Novembro de 2018.

[4] Raspberry Pi Foundation. Disponível em: <https://www.raspberrypi.org>. Acessado em 28 de Agosto de 2018.

[5] Raspbian. Disponível em: <https://www.raspbian.org>. Acessado em 28 de Agosto de 2018.

[6] HPL. Disponível em: <https://www.netlib.org/benchmark/hpl/>. Acessado em 15 de Janeiro de 2019.