Avaliação de Cluster Raspberry Pi para Execução de Aplicações de Análise de Imagens Microscópicas Médicas

Rafael M. Ramos[1], **Célia Ralha**[1], **George Teodoro**[1]

[1]Departamento de Ciência da Computação – Universidade de Brasília (UnB)
Campus Darcy Ribeiro, Edifício CIC/EST – Caixa Postal 4466
Asa Norte – CEP 70910-900 – Brasília – DF – Brasil

rafaelmr@gmail.com, ghedini@unb.br, teodoro@unb.br

Abstract. *The health care area, in particular, the clinical pathology with automated analysis of microscopic images is a field with growing demand for computing power. A barrier to the effective use of computing in this scenario is the unavailability of enough computational resources due to the high related costs. In this paper, we evaluate the use of a low-cost architecture named Raspberry Pi 2 to build computer clusters with 64 cores and 16GB RAM and their use in medical applications in the analysis of microscopic images. Our evaluation's goal the identification of the potential cost benefit of this platform compared to other processors, taking into account runtime, hardware cost and energy consumption. The experimental results showed that the use of a Raspberrys cluster is 10× and 2× faster than the execution of the application, respectively, on a Core2Duo and I7 machine. Even at full capacity, the cluster is more power efficient than other processors only in their idle mode. As such, the Raspberry Pi 2 has proved to be an excellent and promising platform for running our class of target applications.*

Resumo. *A área de saúde e, em especial, a patologia clinica com análise automatizada de imagens microscópicas é um campo com demanda crescente por poder computacional. Uma barreira para a utilização eficaz de computação nesse cenário é a indisponibilidade de recursos computacionais suficientes, devido aos altos custos relacionados. Dessa forma, nesse artigo, avaliamos a utilização de uma arquitetura de baixo custo e gasto energético formada por equipamentos Raspberry Pi 2 para construção de aglomerados de computadores com 64 cores e 16GB RAM para utilização em aplicações médicas em análise de imagens microscópicas. Nossa avaliação tem como objetivo identificar o potencial custo benefício dessa plataforma em comparação com outros processadores, levando em consideração tempo de execução, custo do hardware e gasto energético. Os resultados experimentais mostraram que a utilização de um cluster de Raspberrys é 10× e 2× mais rápido que a execução da aplicação, respectivamente, em máquinas Core2Duo e I7. Mesmo em sua capacidade máxima, o cluster foi energeticamente mais econômico do que os demais processadores. Dessa forma, o Raspberry Pi 2 mostrou-se uma excelente e promissora plataforma para execução de nossa classe de aplicações alvo.*

1. Introdução

A utilização de computação na resolução de problemas da área de saúde é um dos grandes desafios de pesquisa no país e no mundo. Notadamente, o desenvolvimento da área de

saúde vem cada vez mais contanto avanços tecnológicos advindos da aplicação das mais diversas técnicas computacionais. Além disso, com os recentes avanços em diversos tipos de censores, como microscópicos eletrônicos, têm aumentado rapidamente a quantidade de dados disponíveis para análise. Como consequência, a demanda computacional na área é cada vez maior e isso leva a necessidade de analisar a adequação das diversas alternativas computacionais disponíveis para essa classe de aplicações.

Assim, nesse trabalho, avaliamos o uso de clusters (aglomerados) de computação formados por 16 equipamentos Raspberry Pi 2 [FOUNDATION. 2015] com 64 cores e 16GB RAM em aplicações reais de análise de imagens microscópicas [Kong et al. 2013, Cooper et al. 2010, Teodoro et al. 2014, Kurc et al. 2015]. Essa avaliação inclui a comparação desses processadores de baixo custo com CPUs multicores, levando em conta aspectos tais como tempo de execução, custo do equipamento e gasto energético. O Raspberry tem atraído bastante atenção da comunidade científica e industrial pelo fato de apresentarem um baixo custo e gasto energético [Cox et al. 2013]. Essas características ganham ainda mais importância no atual e complicado cenário econômico nacional.

Entretanto, prover recursos computacionais poderosos com baixo custo parece ser uma contradição e o caminho para tal feito é complexo. Apesar de o Raspberry ser um componente básico promissor, ainda não é claro se o poder computacional desse processador é adequado para as altas demandas das aplicações alvo desse estudo. Além disso, o acréscimo de poder computacional nesse ambiente é obtido através da adição de nós em clusters de computadores, mas o impacto dessa estratégia na escalabilidade das aplicações também precisa ser avaliado. Nossas avaliações são feitas utilizando uma aplicação real de análise de imagens microscópicas de câncer cerebral.

As nossas avaliações utilizam métricas tais como o gasto energético, custo e performance do cluster de Raspberrys proposto, comparado com uma plataforma Core2Duo e I7. Em relação a gasto energético cluster Raspberry foi muito melhor, em sua capacidade máxima, o consumo foi menor do que as máquinas Core2Duo e I7 em modo ocioso (executando apenas o sistema operacional). O custo do cluster foi um pouco mais caro que os demais, porém ele conseguiu ser 10x mais rápido que o Core2Duo e pouco mais que 2x mais rápido comparado com o I7. Dessa maneira, o cluster foi proporcionalmente melhor pela eficiência gerada. Também identificamos limitações na arquitetura Raspberry e consequentemente no cluster. Operações que possam usar demasiadamente disco ou que percorram intensivamente, por exemplo, visitando uma vizinhança muito grande para cada ponto analisado em uma imagem incorrem em desvantagens consideráveis comparada com as outras arquiteturas, devido à memória consideravelmente mais lenta do Raspberry.

Apesar de esse artigo ser focado em nossa análise de imagens microscópicas médicas, essa aplicação pertence a uma classe maior de aplicações científicas que analisam dados em baixa dimensionalidade (2D ou 3D) com objetivo de identificar e analisar objetos. Essa classe maior de aplicações inclui aquelas que processam dados de satélites, caracterizam reservatórios subterrâneos e analisam dados de telescópios em astronomia [Klie et al. 2006, Kurç et al. 2005, Parashar et al. 2005, Beynon et al. 2001, Shock et al. 1998, Chang et al. 1997]. Assim, acreditamos que os resultados obtidos das avaliações experimentais podem ser generalizados para outras classes de aplicações.

O restante desse artigo é organizado da seguinte forma. Na Seção 2 descrevemos

a aplicação motivadora em detalhes e as plataformas avaliadas. A implementação da aplicação é descrita na Seção 3 e os resultados experimentais são apresentados na Seção 4. Finalmente, na Seção 5 concluímos o artigo.

2. Aplicação Motivadora e Plataformas Avaliadas

Essa seção apresenta a aplicação real de processamento de imagens microscópicas de tecidos humanos utilizada nesse trabalho, assim como as plataformas nas quais avaliamos experimentalmente a aplicação motivadora.

2.1. Aplicação Motivadora

Alterações morfológicas nos tecidos nas escalas celulares e subcelulares fornecem informações valiosas sobre os tumores, complementando a informação genômica e clínica, podendo levar a uma melhor compreensão da biologia do tumor e resultado clínico [Cooper et al. 2010]. A aplicação motivadora tem sido utilizada no estudo de múltiplos tipos de tumores, que incluem o cerebral e pulmonar.

Os avanços nas tecnologias de captura e digitalização de imagens na última década foram muito significativos e simplificou o processo de capturas de imagens de tecido. Atualmente, é possível obter rapidamente uma imagem de amostra em alta resolução de tecido utilizando microscópico digital. Isso fortaleceu a área de estudos na busca de características morfológicas em escala celular e subcelular. Essa área relativamente nova em comparação as técnicas de ressonância e tomografia computadorizada é conhecida como micrologia digital.

Apesar da facilidade na obtenção das imagens em alta resolução, o processamento das mesmas não é trivial e é agravado pela grande quantidade de dados disponíveis. Uma única imagem pode ter da ordem de 100K $\times$ 100K pixels. Esse cenário torna extremamente complexo o processamento dessas imagens, sendo possível identificar milhões de células na mesma. O processamento de uma única imagem pode levar diversas horas ou dias em um processador sequencial. Um estudo em larga escala pode envolver milhares de imagens de alta resolução, que podem ser processadas múltiplas vezes para mensurar o impacto de estudos de parâmetros. Logo, um estudo em larga escala levaria décadas em um processador sequencial, o que inviabilizaria o processo. Desse modo, esses estudos são viáveis somente com a utilização de processamento paralelo, que estão disponíveis hoje em diversas plataformas.

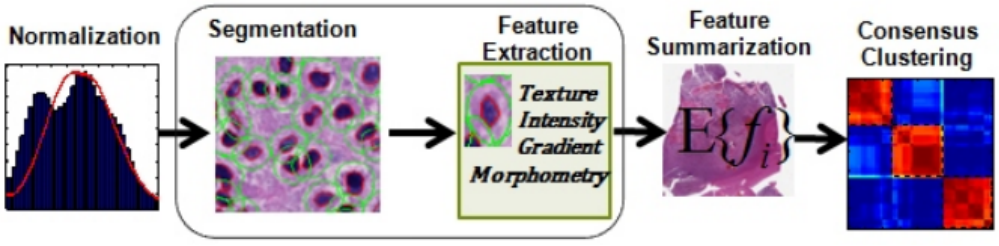

Figure 1. Fluxo de funcionamento da avaliação microscópica.

As fases da aplicação são apresentadas na Figura 1. As imagens são tipicamente dadas como entrada para um estágio de normalização, que remove possíveis ruídos advindo de diferenças na captura da imagem. Em seguida, as células e os núcleos

são detectados e delineados durante a fase de segmentação. Nesta etapa aplica-se uma cascata de operações que incluem limites de valor de pixel e reconstrução morfológica para identificar objetos candidatos, preenchimento de buracos dentro de objetos e definição das áreas onde os objetos não são de interesse. Além disso, objetos são separados utilizando operações como distance transform e watershed. A fase de extração de características calcula um conjunto de propriedades espaciais e textura por objeto, incluem pixels e estatísticas de gradiente, borda e as características morfométricas [Teodoro et al. 2014, Teodoro et al. 2013a, Teodoro et al. 2015]. As etapas de sumarização de características e clusterização são tipicamente mais baratas, pois trabalham em uma representação mais curta das imagens com um vetor de características por imagem.

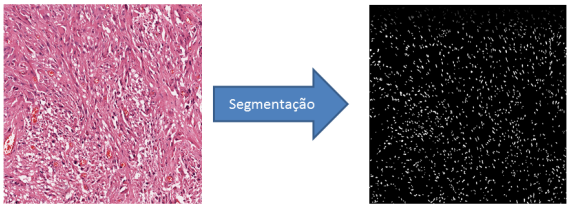

Figure 2. Aplicação da segmentação em uma amostra de tecido cerebral.

As avaliações feitas nesse trabalho utilizam o estágio de segmentação de células da aplicação, pois esta é a etapa mais cara de todo o processo. Na Figura 2 podemos observar um exemplo de entrada (amostra de tecido cerebral) e o resultado da segmentação para a mesma. A descrição do funcionamento dessa fase e as técnicas de programação aplicadas para paralelizar essa fase serão abordadas na sessão de implementação.

2.2. Plataformas Avaliadas

Tivemos grandes evoluções nos últimos anos na computação científica com o crescimento do número de arquiteturas alternativas para execução de aplicações com alta demanda computacional. Como exemplo, podemos citar os coprocessadores como GPUs e o Intel Xeon Phi, além de CPUs multicores com um número cada vez maior de núcleos de processamento. Embora esses processadores tenham conseguido aumentar significativamente a capacidade de processamento de máquinas modernas, eles são encontrados em suas versões mais modernas em valores financeiros elevados. Assim, apresentamos nessa seção, uma arquitetura com um cluster de Raspberry como uma alternativa mais barata em relação a esses processadores, assim como descrevemos duas CPU que são utilizadas para comparação.

Figure 3. Raspberry Pi 2 Modelo B [FOUNDATION. 2015].

A plataforma foi construída utilizando Raspberry Pi 2 B [FOUNDATION. 2015] apresentado na Figura 3, com a configuração apresentada na Tabela 1. Na proposta de um cluster devemos observar com mais atenção processadores, memória RAM e as interconexões de comunicação entre os nós do mesmo. O Raspberry é equipado com um processador Quad Core ARM7 Cortex 900MHz, 1 GB de Memória RAM e interface de rede de 100Mbits/s. O equipamento suporta a utilização de cartão micro SD para o uso de armazenamento de dados, sendo que em nossa solução usamos uma memória micro SD classe 10 Ultra de velocidade até 80MB/s [Corporation 2016b]. Alguns destaques desse equipamento são seu baixo gasto energético (3W) e ausência de sistema de resfriamento, como coolers e dissipadores de calor.

Table 1. Especificações do Raspberry Pi 2 [FOUNDATION. 2015] utilizado.

Ethernet	10/100 BaseT Ethernet socket
USB	4 x USB 2.0 Connector
CPU	Quad-core ARM Cortex-A7 900MHz
card slot	Micro SDIO
Chip	Broadcom BCM2836 SoC
Memory	1GB LPDDR2
Video out	HDMI v1.4
Power	Micro USB socket 5V, 2A

O cluster que construímos é composto por 16 Raspberry Pi2 B com cartão de memória para armazenamento secundário de 16GB classe 10 Ultra. Os processadores são interligados através de um Switch Ethernet de 100MB/s com 24 portas. O sistema operacional utilizado é o Raspian, uma variação do Linux Debian compilado para arquitetura Arm7 [Raspian 2016]. A comunicação entre processos e gerenciamento dos mesmos na nossa solução utiliza a plataforma MPI (Message Passing Interface) [mpi].

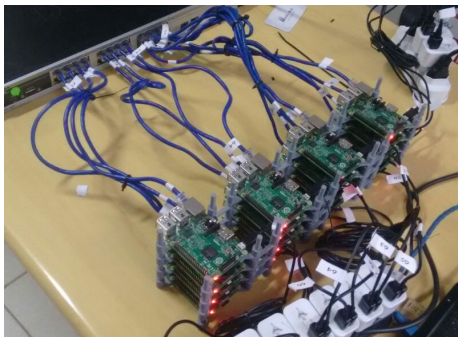

Figure 4. Cluster de baixo custo montando utilizado processadores Raspberry.

Como podemos observar na Figura 4, o cluster foi construído com quatro torres com quatro Raspberrys cada. Temos dois Raspberrys conectados em cada fonte de energia compatível com carregadores de celular 2A. As torres dos processadores foram impressas utilizando uma impressora 3D. Para comparar a eficiência de nosso cluster, decidimos comparar nossa capacidade de processamento com os processadores CPU Core2Duo e I7 apresentados na Tabela 2.

Outro item de nossa avaliação é o consumo de energia de cada um dos equipamentos, que são apresentados na Tabela 3. Como visto, o consumo de energia dos sistemas

Table 2. Configuração dos equipamentos comparados.

Equipamento	Proc.(MHz)	Núcleos	RAM	RAM(MHz)	Processador
Core2Duo	2300	2	4GB	667	Intel(R) Core(TM)2 Duo CPU E6550
I7	3000	8	8GB	1600	Intel(R) Core(TM) i7-4765T
Raspberry Pi 2 B	1000-700	4	1GB	500-400	Armv7

com CPU são maiores até que os do cluster de Raspberrys, mesmo quando comparamos os gastos das CPUs ociosas com os valores máximos dos Raspberrys. Alguns dos fatores que contribuem para o baixo consumo dos Raspberrys é a baixa frequência dos processadores e arquitetura mais simples, que inclusive fazem com que o mesmo não precise de dissipadores ou coolers para o resfriamento. Quando avaliamos o custo financeiro, nota-se que o valor unitário do Raspberry é significativamente menor que dos outros equipamentos, enquanto um cluster com 16 processadores Raspberry incluindo todos os equipamentos para sua montagem é um pouco mais caro que o valor em reais do I7.

Table 3. Consumo de energia e sistema de ventilação. O consumo dos processadores ociosos e trabalhando com carga máxima dos processadores foi extraídos de [Page 2009, Butler 2012]. Os preços dos equipamentos CPU foram extraídos de [DELL 2016], visitado em fevereiro de 2016.

Equipamento	Potência (W)		Preço(R$)	Observações
	Ocioso	Max.		
Core2Duo	114	163	1.200,00	HD Sata 500GB, 4G RAM
I7	100	538	3.000,00	HD Sata 1TB, 8G RAM
Raspberry Pi2 B	-	3	250,00	1 fonte de 2A, 1 cartão micro SD classe 10 Ultra 16GB
Cluster RB	-	48	3.440,00	8 fontes de 2A, 16 cartão micro SD classe 10 Ultra 16GB, switch 24 portas 10/100 Dlink

3. Implementação

Nosso estudo se baseou na fase de segmentação da aplicação motivadora, cujas operações fundamentais são apresentadas na Tabela 4 . Estas operações são categorizadas de acordo com o padrão de acesso a dados e a intensidade de computação. Sempre que possível, usamos implementações sequenciais das operações disponíveis com OpenCV [Bradski 2000] ou de outros grupos de pesquisa, tal como apresentado na coluna Fonte CPU da Tabela 4.

As operações na fase de segmentação realizam computações em elementos do domínio de dados de entrada, pixels em nosso caso. As operações realizadas podem ser categorizadas quanto ao acesso aos dados: regulares e irregulares. Os regulares acessam os dados em regiões contínuas e os irregulares acessam dados distribuídos de maneira irregular. Em alguns casos alguns dados que necessitam serem processados só são definidos em tempo de execução como resultado de alguma computação e por isso teremos um acesso irregular (pseudo-randômico) [Teodoro et al. 2013b, Teodoro et al. 2012]. Ainda em relação ao acesso de dados, temos a classificação em relação à computação de um dado

Table 4. Tabela de operações fundamentais da fase de segmentação

Operação	Descrição	Padrão de Acesso a Dados	Computação	CPU Source
Convert RGB to grayscale	Converter a imagem RGB em escala em cinza	Regular, multicanal local	Moderada	OpenCV
Morphological Open	Abrir e remover pequenos objetos, e preencher buracos no primeiro plano	Regular, vizinhança (disco 13x13)	Baixa	OpenCV
Morphological Reconstruction	Ajustar a imagem a uma máscara.	Irregular, vizinhança (4-/8-connected)	Baixa	Vincent [Vincent 1993]
Area Threshold	Remover objetos fora da área de interesse	Mista, vizinhança	Baixa	Desenvolvido
FillHolles	Preencher buracos em objetos usando pontos selecionados	Irregular, neighborhood (4-/8-connected)	Baixa	Vincent [Vincent 1993]
Distance Transform	Computar a distância do ponto mais próximo do fundo para cada pixel do primeiro plano.	Irregular, vizinhança (8-connected)	Moderada	Desenvolvido
Connected Components Labeling	Rotular com os mesmos pixels de valor em objetos de imagens de uma máscara	Irregular, global	Baixa	Desenvolvido

e pode ser local, operação depende apenas do seu valor, multicanal local, uma variação do local com a possibilidade do acesso aos dados em outras camadas do mesmo índice, vizinhança, operações sobre vizinhos espaciais ou temporais, global, operação dentro da área delimitada. As operações de segmentação variam de baixa a moderada no grau de computação. O fluxo de trabalho detalhado da aplicação é apresentado na Figura 5.

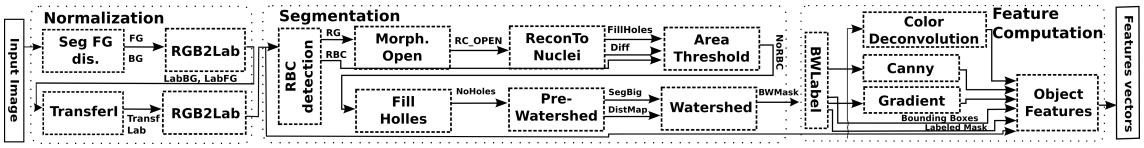

Figure 5. Fluxo de trabalho detalhado da aplicação.

A implementação da fase de segmentação foi feita utilizando C++ e sua paralelização utiliza MPI [mpi] para instanciar processos e gerenciar a comunicação entre eles. Nessa implementação, a execução é feita utilizando o modelo Mestre/Escravo, onde o mestre armazena a informação sobre as imagens que precisam ser processadas e assinala as mesmas para execução nos escravos sob demanda. Cada escravo então lê a imagem a ser processada e invoca o estágio de segmentação que é executado sequencialmente. Múltiplos núcleos de processamento em uma máquina ou em um ambiente distribuído são utilizados através criação de várias cópias do processo escravo.

4. Resultados

A avaliação experimental foi realizada utilizando a aplicação motivadora, implementada em C++ com MPI, utilizando 512 imagens de 1K$\times$1K pixels como entrada. As imagens somam um total de 6GB de dados.

4.1. Avaliação do desempenho dos processadores para diferentes configurações

Uma preocupação inicial com a execução no Raspberry era a velocidade de leitura de arquivos, já que a primeira etapa da aplicação lê a imagem antes de repassar a mesma para computação na segmentação. Ao contrário dos outros equipamentos que usam HD SATA, o Raspberry utiliza uma memória micro SD que possuí uma velocidade inferior. Logo, avaliamos variações do micro SD para analisar o impacto do mesmo para o desempenho da aplicação. As versões analisadas foram o micro SD Classe 10 Ultra que chega a velocidade de 80MB/s [Corporation 2016b] e micro SD Classe 10 Extreme que chega a 95MB/s [Corporation 2016a]. O Raspberry possui um processador Arm7 que tem velocidade variável de 700MHZ a 1000MHZ. Também avaliamos a velocidade mínima e máxima do processador para verificar seu impacto nos resultados.

Table 5. Proc. 512 imagens com apenas um único processo trabalhador (1 core apenas é utilizado para processamento.

Equipamento	Vel.(MHz)	Total(min.)	Img Read(seg.)
Core 2 Duo	2300	100,07	79,84
RB SD Classe 10 Extreme	1000	186,43	338,23
RB SD Classe 10 Ultra	1000	187,25	345,93
RB SD Classe 10 Extreme	700	227,95	339,78
RB SD Classe 10 Ultra	700	227,45	345,36
I7	3000	49,17	49,55

A Tabela 5 apresenta o tempo total de execução e tempo de leitura de arquivo de imagem na sua execução sequencial da aplicação em cada processador para as diferentes configurações. A diferença de velocidade pelo uso do micro SD Classe 10 Ultra ou Extreme não alterou significativamente os resultados. Como cada arquivo é lido uma única vez no início da execução e os resultados gravados ao fim da mesma, o ganho com o micro SD mais rápido é pequeno, pois as operações de E/S representam uma parte pequena do tempo de execução. Entretanto, a comparação dos tempos de leitura do RB com as outras plataformas mostra uma grande diferença, que pode ser impactante para outra aplicações mais intensivas em E/S. A comparação do tempo de execução total da aplicação mostra que cada core do I7 é cerca de 3,8$\times$ mais rápido que um core do RB, enquanto essa diferença diminuí para aproximadamente 1.8$\times$ para o Core 2.

4.2. Desempenho por tipo operação

Nessa seção apresentamos os dados referentes ao custo de execução de cada uma das operações principais do estágio de segmentação nas plataformas alvo da nossa análise. Os valores apresentados na Tabela 6 referem-se ao custo médio de cada etapa na execução das 512 imagens utilizadas como entrada.

Table 6. Tempo médio de cada etapa da fase de segmentação em segundos

Equipamento	RBC	Morph.Open	ReconToNuclei	AreaThreshold	FillHolles	Pré-Watershed	Watershed
Core2Duo	247	418	2282	241	1238	1966	519
I7	92	11	1196	116	633	1047	301
Raspberry Pi2B	1533	1891	1735	515	1155	5178	978

Conforme pode ser visto, existe uma grande variabilidade em relação ao desempenho relativo entre os processadores para as diversas operações. Em especial, o Raspberry apresentou um desempenho significativamente inferior as outras plataformas na execução do Morph.Open, que é uma convolação que utiliza um disco de raio relativamente grande. Assim, para cada ponto da imagem, um grande número de vizinhos precisa ser acessado. Entretanto, como a memória do Raspberry é mais lenta e seu cache menor, logo, o desempenho do processador para essa operação é ruim. Inicialmente essa operação apresentava um tempo de execução de 18 segundos para cada imagem. Como essa operação se tornou um gargalo para nossa solução, realizamos uma modificação em sua implementação. Pesquisamos alternativas a essa função e simulamos algumas configurações. Descobrimos que poderíamos gerar o mesmo resultado diminuindo o tamanho do disco e alterando o tamanho da vizinhança que deveria ser acessada. O resultado então pode ser melhorado para aproximadamente 3 segundos para cada imagem.

Com isso diminuímos o acesso intensivo a memória minimizando o gargalo de memória do Raspberry.

O desempenho relativo do Raspberry em relação aos outros processadores é melhor para operações irregulares, pois nenhuma arquitetura é otimizada para esses tipos de acesso a dados e, inevitavelmente, nesse caso o cache vai ter menos eficiência e o custo de acesso a memória é o limitador.

4.3. Avaliação de escalabilidade e melhor desempenho em cada plataforma

Nessa seção apresentamos os resultados onde os equipamentos são avaliados instanciando um processo MPI trabalhador por núcleo de processamento, sendo assim possível processar múltiplas imagens em paralelo. A Figura 6 apresenta o speedup obtido na execução da aplicação no cluster Raspberry. Como visto, a aplicação atingiu um speedup de aproximadamente 47× em relação a sua versão sequencial rodando no mesmo processador, o que leva a uma eficiência de aproximadamente 75%. Adicionalmente, a eficiência na execução com 4 núcleos em um único Raspberry é de 80%. Logo, as perdas de eficiência não são relacionadas a utilização de múltiplas maquinas, mas em relação ao uso dos núcleos de processamento em um Raspberry. Isso se deve ao fato de que essa aplicação é sabidamente intensiva em acesso a dados e algumas de suas operações fazem acesso irregulares à memória, o que leva a criação de um gargalo nos subsistemas de memória e limitação da escalabilidade de um processador.

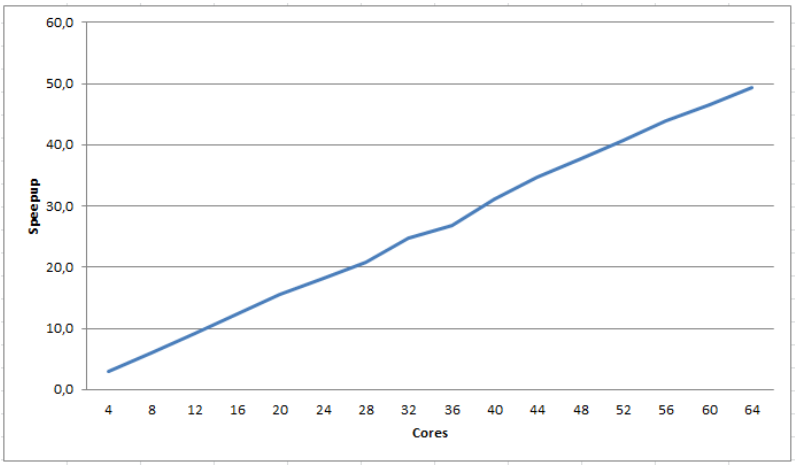

Figure 6. Speedup pela quantidades de trabalhadores (núcleos de processamento)

A Tabela 7 apresenta resultados comparativos das plataformas avaliadas. Como visto, o I7 atingiu a pior eficiência entre os processadores analisados. Esse resultado é consequência da maior quantidade de núcleos de processamentos desse processador, que são acoplados ao mesmo subsistema de memória que se torna um gargalo rapidamente devido ao fato das operações serem intensivas em dados. Esse efeito é menor no cluster de Raspberry, pois cada uma das memórias funciona independentemente e criam um ambiente mais adequado para aplicação. Adicionalmente, como visto, o cluster apresentou o melhor tempo de execução dentre as plataformas avaliadas. Ele é 10,62× mais rápido que o Core2Duo e 2,32× mais rápido que o I7.

Discussão: Em relação ao consumo de energia, apesar de não termos medido exatamente o que os equipamentos consomem, nos baseamos na documentação encontrada

Table 7. Resultados de tempo na capacidade máxima dos equipamentos.

Equipamento	Núcleos	Tempo(min)	speedup	Eficiência
Core 2 Duo	2	51,33	1,95	97%
I7	8	11,22	4,38	55%
Cluster Raspberry	64	4,83	47,16	78%

sobre as plataformas. Cada Raspberry Pi2 B na pior hipótese irá consumir cerca de 3W, ou seja, e baseado nisso o cluster com os 16 Raspberrys irão consumir na pior hipótese 48W. Buscando as informações de consumo do Core2Duo e do I7, apenas os processadores, desconsiderando o resto dos equipamentos necessários para montar a CPU, em modo ocioso esses processadores irão consumir 114W e 100W respectivamente. Logo o cluster irá consumir $2\times$ menos energia que qualquer outro equipamento no pior caso e o mesmo tempo de execução. Entretanto, essa diferença é ainda maior na prática porque as CPU não estarão ociosas e elas demoram mais tempo para executar a aplicação. Ainda no aspecto físico, o Raspberry não necessita de dispositivos de ventilação o que diminui custos para criação de racks ou salas cofres com sistema especial de resfriamento.

5. Conclusão e Trabalhos Futuros

Este trabalho avaliou alternativas computacionais para executar aplicações de processamento de imagens médicas microscópicas de tecidos, capazes de oferecer um bom custo benefício e os requisitos computacionais necessários para executar eficientemente essas aplicações. Para tanto, focamos no estudo de plataforma de baixo custo e comparamos custos, consumo energético e performance com outros processadores. Dentre essas plataformas, se destaca o Raspberry, que tem todos os atributos anteriores, mas com poder computacional limitado. Assim, para atingir a capacidade de computação desejada construímos um cluster com 16 máquinas Raspberrys, que foi comparada com processadores Core2Duo e I7.

O estudo realizado utiliza a fase de segmentação de nossa aplicação motivadora por ser a que demanda mais recurso. Verificamos que o cluster em seu consumo máximo de energia era de 48W enquanto o consumo de energia do Core2Duo e I7 respectivamente 114 e 110 em modo ocioso. Levantamos os custos financeiros de cada plataforma e constatamos que o cluster era um pouco mais custoso que o I7 e cerca de quase 2x mais caro que o Core2Duo. Entretanto, a avaliação de desempenho comparativa das plataformas mostrou que o cluster com Raspberry é 2x mais rápido que o I7 e 10x mais rápido que o Core2Duo. Assim, o custo benefício cluster em função do desempenho alcançado é muito melhor que das outras plataformas.

Embora o desempenho geral da aplicação tenha sido melhor no cluster com Raspberry, o comportamento de desempenho relativo as plataformas em função das operações internas da aplicação pode variar conforme a forma de acesso a dados e intensidade de computação. Se tivermos um algoritmo que explora muito a memória RAM pode ser que o Raspberry seja inferior por ser equipado com memória RAM mais lenta. Outro potencial gargalo dessa máquina pode ser a leitura e escrita de arquivos em uma aplicação que realize muitas operações E/S.

Durante nossos testes, nos deparamos com alguns resultados ruins na execução do cluster Raspberry. A etapa de Morphological Open apresentava um resultado de tempo

muito ruim. Avaliando mais a fundo, descobrimos que essa operação precisa percorrer de maneira intensa a memória RAM. Como o Raspberry Pi2 B possui uma memória RAM bem mais lentas que os demais equipamentos, foi necessário realizar alterações. Verificamos que o tempo estava relacionado diretamente com o tamanho do disco de entrada para realização da função Morphological. Conseguimos ajustar esse disco e chegar a um resultado muito melhor de desempenho sem alterar o resultado da aplicação. Com essa situação, ficou claro que devemos pesquisar mais a fundo o funcionamento das etapas de segmentação para melhorarmos sua eficiência com foco na plataforma de nosso cluster. Outra linha de trabalho que devemos continuar a desenvolver são as demais fases de nossa aplicação, normalização e extração de características.

Agradecimentos. Gostaríamos de agradecer ao CNPq e CAPES pelo apoio ao trabalho e a empresa CleverSystems (http://www.cleversystems.com.br), que nos forneceu os equipamentos Raspberrys Pi2 B e a estrutura para realização dos testes.

References

The Message Passing Interface (MPI).

Beynon, M. D., Kurc, T., Catalyurek, U., Chang, C., Sussman, A., and Saltz, J. (2001). Distributed processing of very large datasets with DataCutter. *Parallel Comput.*, 27(11):1457–1478.

Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

Butler, H. (2012). Intel core i7-3770k power consumption. `http://www.bit-tech.net/hardware/2012/04/23/intel-core-i7-3770k-review/8`.

Chang, C., Moon, B., Acharya, A., Shock, C., Sussman, A., and Saltz, J. H. (1997). Titan: A High-Performance Remote Sensing Database. In *Proceedings of the Thirteenth International Conference on Data Engineering*, ICDE '97, pages 375–384, Washington, DC, USA. IEEE Computer Society.

Cooper, L. A. D., Kong, J., Gutman, D. A., Wang, F., Cholleti, S. R., Pan, T. C., Widener, P. M., Sharma, A., Mikkelsen, T., Flanders, A. E., Rubin, D. L., Meir, E. G. V., Kurc, T. M., Moreno, C. S., Brat, D. J., and Saltz, J. H. (2010). An integrative approach for in silico glioma research. *IEEE Trans Biomed Eng.*, 57(10):2617–2621.

Corporation, S. (2016a). Cartao de memoria sandisk extreme microsdhc/microsdxc uhs-i. `https://www.sandisk.com.br/home/memory-cards/microsd-cards/extreme-microsd`.

Corporation, S. (2016b). Cartao de memoria sandisk ultra microsdhc/microsdxc uhs-i para cameras. `https://www.sandisk.com.br/home/memory-cards/microsd-cards/ultra-microsd-for-cameras`.

Cox, S. J., Cox, J. T., Boardman, R. P., Johnston, S. J., Scott, M., and O'Brien, N. S. (2013). Iridis-pi: a low-cost, compact demonstration cluster. *Cluster Computing*, 17(2):349–358.

DELL (2016). Ecommerce dell. `http://www.dell.com.br`.

FOUNDATION., R. P. (2015). Raspberry pi 2 model b. `https://www.raspberrypi.org/products/raspberry-pi-2-model-b/`.

Klie, H., Bangerth, W., Gai, X., Wheeler, M. F., Stoffa, P. L., Sen, M. K., Parashar, M., Çatalyürek, Ü. V., Saltz, J. H., and Kurç, T. M. (2006). Models, methods and middleware for grid-enabled multiphysics oil reservoir management. *Eng. Comput. (Lond.)*, 22(3-4):349–370.

Kong, J., Cooper, L. A. D., Wang, F., Gao, J., Teodoro, G., Mikkelsen, T., Schniederjan, M. J., Moreno, C. S., Saltz, J. H., and Brat, D. J. (2013). Machine-based morphologic analysis of glioblastoma using whole-slide pathology images uncovers clinically relevant molecular correlates. *PLoS ONE*.

Kurc, T., Qi, X., Wang, D., Wang, F., Teodoro, G., Cooper, L., Nalisnik, M., Yang, L., Saltz, J., and Foran, D. J. (2015). Scalable analysis of big pathology image data cohorts using efficient methods and high-performance computing strategies. *BMC Bioinformatics*, 16(1):1.

Kurç, T. M., Çatalyürek, Ü. V., Zhang, X., Saltz, J. H., Martino, R., Wheeler, M. F., Peszynska, M., Sussman, A., Hansen, C., Sen, M. K., Seifoullaev, R., Stoffa, P. L., Torres-Verdín, C., and Parashar, M. (2005). A simulation and data analysis system for large-scale, data-driven oil reservoir simulation studies. *Concurrency - Practice and Experience*, 17(11):1441–1467.

Page, M. (2009). Cpu power consumption tests. `http://www.pcstats.com/articleview.cfm?articleid=2395&page=2`.

Parashar, M., Matossian, V., Bangerth, W., Klie, H., Rutt, B., Kurç, T. M., Çatalyürek, Ü. V., Saltz, J. H., and Wheeler, M. F. (2005). Towards Dynamic Data-Driven Optimization of Oil Well Placement. In *International Conference on Computational Science (2)*, pages 656–663.

Raspian (2016). Raspian. `http://www.raspian.org/`.

Shock, C. T., Chang, C., Moon, B., Acharya, A., Davis, L., Saltz, J., and Sussman, A. (1998). The design and evaluation of a high-performance earth science database. *Parallel Computing*, 24(1):65 – 89.

Teodoro, G., Kurc, T., Andrade, G., Kong, J., Ferreira, R., and Saltz, J. (2015). Application performance analysis and efficient execution on systems with multi-core CPUs, GPUs and MICs: a case study with microscopy image analysis. *International Journal of High Performance Computing Applications*.

Teodoro, G., Kurç, T. M., Kong, J., Cooper, L. A. D., and Saltz, J. H. (2014). Comparative Performance Analysis of Intel (R) Xeon Phi (TM), GPU, and CPU: A Case Study from Microscopy Image Analysis. In *2014 IEEE 28th International Parallel and Distributed Processing Symposium, May 19-23, 2014*, pages 1063–1072.

Teodoro, G., Pan, T., Kurc, T. M., Cooper, L., Kong, J., and Saltz, J. H. (2012). A fast parallel implementation of queue-based morphological reconstruction using GPUs. *Emory University, Center for Comprehensive Informatics, CCI-TR-2012-2*.

Teodoro, G., Pan, T., Kurc, T. M., Kong, J., Cooper, L. A., Podhorszki, N., Klasky, S., and Saltz, J. H. (2013a). High-throughput Analysis of Large Microscopy Image Datasets on CPU-GPU Cluster Platforms. In *IPDPS '13: Proceedings of the 2013 IEEE International Symposium on Parallel and Distributed Processing*.

Teodoro, G., Pan, T., Kurc, T. M., Kong, J., Cooper, L. A., and Saltz, J. H. (2013b). Efficient Irregular Wavefront Propagation Algorithms on Hybrid CPU-GPU machines. *Parallel Computing*, 39(4-5):189–211.

Vincent, L. (1993). Morphological grayscale reconstruction in image analysis: Applications and efficient algorithms. *IEEE Transactions on Image Processing*, 2:176–201.