**Avaliação de Cluster Raspberry Pi para Execução de Aplicações de Análises de Imagens Microscópicas Médicas**

Este trabalho avalia cluster formado com 16 placas Raspberry Pi, totalizando 64 núcleos e 16GB de memória RAM – memória secundária de 16GB (cartão SD), sistema operacional Raspbian, interface de rede de 100Mb/s e todas as placas interligadas por um switch ethernet de 100Mb/s com 24 portas, para análise de imagens microscópicas. Essa avaliação compara processadores de baixo custo com CPUs multicores considerando tempo de execução, gasto energético, custo dos equipamentos e performance do cluster. Cada raspberry (modelo 2B) possui: processador Quad Core ARM7 Cortex 900MHz, 1GB RAM, interface de rede 100Mbits/s. Para instalação do SO foram utilizados cartões micro SD 16GB classe 10 (até 80MB/s). O cluster foi construído em quatro torres contendo quatro placas em cada, cada fonte de energia (de 5V DC e 2A) alimentou duas placas. Foi utilizada linguagem C++ e a paralelização foi feita utilizando a biblioteca MPI, a execução foi no modelo mestre/escravo. Foram utilizadas como entrada para os testes 512 imagens de 1K x 1K pixels, totalizando 6GB de dados de entrada.

Em consumo de energia, a placa (em capacidade máxima de processamento) obteve consumo menor que os computadores (mesmo em estado ocioso: apenas as aplicações do sistema), chegando a 2x menos energia. O cluster custou mais caro que os PCs, porem em desempenho este foi 2x mais rápido que a máquina com processador I7 e 10x mais rápido que a máquina com processador Core2Duo, tornando o cluster mais eficiente. Nas avaliações das máquinas em comparação com uma placa Raspberry Pi, esta apresentou desempenho pior que das máquinas (I7 e Code2Duo). Devido as memórias (primárias e secundárias) da placa serem mais lentas que as dos PCs, operações que realizam muitos acessos as estas memórias fazem com que o desempenho da Raspberry caia se comparado ao dos PCs (por exemplo: operações que visitam muitos pixels vizinhos do ponto analisado dentro de uma imagem). O processamento das imagens se deu pelas seguintes etapas: normalização, segmentação, extração de características, sumarização das características e clusterização.

**Análise de Desempenho em Dispositivos Limitados e Emulados  
Estudo de caso: Raspberry Pi e Web Services RESTful.**

Este trabalho compara o desempenho de sistemas reais e sistemas emulados em dispositivos com recursos limitados, nesse caso o Raspberry Pi. A análise de desempenho foi realizada em serviços web configurados na placa Raspberry Pi (utilizando RESTful e o framework CXF) – foram considerados os tempos de processamento de requisições, de empacotamento e desempacotamento de mensagens. Para avaliar o desempenho dos serviços web, o seguinte cenário foi proposto: a aplicação cliente gera uma sequência aleatória de números, envia para o servidor, que então recebe a mensagem ordena estes números e então envia de volta ao cliente os mesmos, ordenados e após recebidos, a aplicação cliente finaliza a conexão com o servidor. Foi utilizado um Raspberry Pi modelo B com cartão SD de 16GB e SO Linux Raspbian. Para o sistema emulado, foi utilizado o software Qemu, que simulou um hardware equivalente ao do Raspberry e com o mesmo SO. Os experimentos foram realizados 50 vezes (IC de 95%) com mensagens de 100Kb e 500Kb tanto no sistema real quando no emulado. Uma solução foi implementada para obter tempo de serialização e deserialização e tempo total das aplicações tanto no cliente como no servidor.

Com base nos resultados pode-se perceber que o comportamento no sistema emulado e no sistema real são próximos tanto no cliente como no servidor. No sistema emulado há um leve aumento nos tempos, mas é justificado devido ao custo da emulação.

**Raspberry Pi e Experimentação Remota.**

Esse trabalho propõe um sistema para acesso remoto utilizando uma placa Raspberry Pi. Foi configurado um servidor web na placa que ira comunicar com uma pagina web (que deve estar na mesma rede). Através dessa pagina foi possível o gerenciamento de portas (envio de comandos, acionamento e/ou leitura de portas, etc) para o servidor web na placa. Também foi configurado uma webcan que monitora os experimentos, as imagens são envidas para a placa e esta a envia para a internet (streaming de vídeo). Foi configurada na placa: pacote lighttpd (um servidor web) para servidor de arquivos em C e também PHP, pacote Moiton para o servidor de câmeras (em que cada câmera recebe uma porta de endereço).

Na pagina web foram colocados 3 botões (acender, apagar e verificar o estado de um led), além de um quadro que exibe o streaming da webcan. Cada um desses botões enviar comandos de leitura de arquivos (.C) específicos localizados no servidor de arquivos. Em cada um desses arquivos contem as rotinas para cada função (acender, apagar e verificar). Concluiu-se que a aplicação proposta obteve ótimos resultados, inclusive da utilização do Raspberry como servidor, o que possibilita a criação de servidores web de baixo custo porem com bom desempenho.

**Avaliação de implementações do Algoritmo Genético Paralelo para soluções do Problema do Caixeiro Viajando usando OpenMP e Pthreads.**

Esse trabalho apresenta uma implementação do Problema do Caixeiro Viajante na estrutura de um Algoritmo Genético (AG) de forma paralela utilizando a biblioteca OpenMP e a biblioteca Pthreads. Um AG é inspirado na evolução da natureza, em que os melhores indivíduos sobrevivem e se reproduzem, fazendo com que cada geração seja melhor que a anterior. O paralelismo é utilizado para se ter melhor desempenho em relação a forma sequencial, e ainda aproveita os vários núcleos que os processadores atuais apresentam.

Nessa implementação do AG cada combinação de rota (passando por todas as cidades) foi definida como um indivíduo, e um conjunto desses indivíduos uma população; cada nó (ou cidade) foi definido como um gene. Assim quanto maior a população, maior a chance de encontrar uma boa rota porém maior será o processamento necessário para gerar a próxima geração. Foram gerados conjuntos de populações (ou ilhas) que sofrem a mutação um independente da outra (a função que realiza a mutação é executada de forma paralela, ou seja, as ilhas sofrem mutações ao mesmo tempo, porém independentes umas das outras). A implementação foi feita em C e cada indivíduo foi representado por um vetor com tamanho igual ao numero total de cidades (as coordenadas das cidades estavam em um vetor estático de ponto flutuante também com tamanho igual ao numero de cidades) - a população é um vetor de indivíduos. Na implementação usando a biblioteca pthreads, cada thread ficou com uma struct e nesta havia um vetor de indivíduos (ou população), assim foi possível ter as ilhas isoladas umas das outras (pois cada thread só acessa sua struct).

Os testes foram realizados em dois computadores ambos com a mesma distribuição Linux, um com quatro núcleos reais e outro com dois reais (em que cada núcleo real simula dois núcleos - SMP). Foram ajustados: taxas de erro, numero máximo de gerações, numero de indivíduos em cada população e numero de indivíduos na elite. Os resultados mostram que no computador com SMP a implementação com openMP obteve melhor resultado que a pthread, enquanto que no computador com 4 núcleos reais a pthread obteve melhor resultado. Nos dois casos a implementação paralela obteve resultado significativamente melhor em relação a sequencial. Apesar de que nos cenários propostos a implementação paralela foi melhor que a sequencial, deve-se tomar cuidado ao escolher entre as bibliotecas openMP e pthread, pois cada uma apresenta desempenho melhor em diferentes situações, tornando essencial a análise do problema antes da escolha.

**Estudo Comparativo Entre Clusters de Computadores Desktop e de Dispositivos ARM**

Este trabalho apresenta uma comparação de custo/benefício entre um cluster de computadores desktop tradicionais (computadores HP Compaq 6005 Pro Microtower) e de dispositivos de baixo consumo de energia, como Raspberry Pi Modelo B e Cubietruck. Para testá-los, foram implementados algoritmos (multiplicação de matrizes MxN e Caixeiro Viajante) para resolução de problemas das classes P e NP-Difícil além do software benchmark HP Linpack (HPL). A biblioteca MPI foi utilizada para troca de mensagens. Três clusters homogêneos (no formato mestre/escravo) foram montados: computadores desktop, Raspberry Pi e Cubietruck.

Foram realizadas análises para se obter dados de custo energético (foi desenvolvido um sistema com microcontrolador Tiva e um sensor de corrente), financeiro e velocidade de processamento (para esse parâmetro utilizou-se speedup e o benchmark HPL). Com os dados coletados de ambos os clusters foi possível estimar o custo e benefício gerado, bem como verificar sua aplicabilidade. Concluiu-se que os clusters de Raspberry Pi e de Cunietruck tem custo financeiro e consumo energético muito inferior ao de computadores desktops. Porém ao analisar os dados de speedup e do HPL pode-se perceber que a capacidade de processamento do cluster de computadores desktop é muito superior ao demais. Por fim, concluiu-se que cluster de dispositivos ARM são recomendados para aplicações que necessitam de baixo custo e baixo consumo energético, já cluster de computadores desktops são recomendados para aplicações em que o tempo seja crítico ao sistema e não preocupação com custo e consumo (inclusive de refrigeração).

**Avaliação do uso da computação paralela em redes de computadores desktop e dispositivos ARM**

Neste trabalho foi realizada a construção de dois clusters isolados um do outro: um com sete computadores HP Compaq 6005 Pro Microtower e outro com oito dispositivos ARM (Raspberry Pi) modelo B. Os computadores com distribuição Linux Ubuntu Server e Paspbian para os dispositivos ARM. A biblioteca MPI foi utilizada para a paralelização. Como parâmetros de análise foram considerados o benchmark HP Linpack (HPL), calculo de speedup, consumo de energia e custo monetário dos equipamentos. Os cálculos de speedup serão realizados a partir de algoritmos projetados para solucionar problemas, como: Quick-Sort, multiplicação de matriz e Caixeiro Viajante utilizando o método GRASP (Greedy Randomized Adaptive Search Procedure). Para calculo do consumo energético foram utilizados sensores de corrente e tensão.

Pode-se concluir que é possível utilizar técnicas de paralelização para equilibrar a inferioridade dos dispositivos embarcados em relação aos computadores convencionais. Sendo assim, é possível criar um sistema computacional distribuído de alto poder de processamento e baixo custo, tanto monetário, quanto energético. Contudo os dados obtidos após a avaliação mostram que o cluster de Raspberry Pi é inviável para grandes cargas de processamento e com poucos nós.

**Implantação e Análise de Desempenho de um Cluster com Processadores ARM e Plataforma Raspberry Pi**