**Daniel Moreira Cestari**

# *Execução de um conjunto de benchmarks em um sistema embarcado voltado a aplicações agrícolas*

São Carlos - SP, Brasil

2 de Dezembro de 2011

**Daniel Moreira Cestari**

# *Execução de um conjunto de benchmarks em um sistema embarcado voltado a aplicações agrícolas*

Monografia apresentada para o trabalho de conclusão do curso de Engenharia de Computação

Orientador:
Prof. Dr. Maximiliam Luppe

DEPARTAMENTO DE ENGENHARIA ELÉTRICA
ESCOLA DE ENGENHARIA DE SÃO CARLOS
UNIVERSIDADE DE SÃO PAULO

São Carlos - SP, Brasil

2 de Dezembro de 2011

Ficha catalográfica preparada pela Seção de Tratamento
da Informação do Serviço de Biblioteca – EESC/USP

C421e

Cestari, Daniel Moreira.
Execução de um conjunto de *benchmarks* em um sistema embarcado voltado a aplicações agrícolas. / Daniel Moreira Cestari ; orientador Maximiliam Luppe-- São Carlos, 2011.

Monografia (Graduação em Engenharia da Computação) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2011.

1. *Benchmark*. 2. Sistemas embarcados. 3. Agricultura de precisão. I. Titulo.

# FOLHA DE APROVAÇÃO

**Nome:** Daniel Moreira Cestari

**Título:** "Execução de um Conjunto de Benchmarks em um Sistema Embarcado Voltado a Aplicações Agrícolas"

**Trabalho de Conclusão de Curso defendido e aprovado em** 02 / 12 / 2011 ,

**com NOTA** 7,0 ( Sete , Zero ), **pela comissão julgadora:**

Marcelo Andrade da Costa Vieira

**Prof. Dr. Marcelo Andrade da Costa Vieira - SEL/EESC/USP**

Valdir Grassi Junior

**Prof. Dr. Valdir Grassi Júnior - SEL/EESC/USP**

*Prof. Associado Evandro Luís Linhari Rodrigues*
*Coordenador pela EESC/USP do*
*Curso de Engenharia de Computação*

# *Resumo*

Este trabalho visa a execução de um conjunto de *benchmarks* em sistema embarcado voltado a aplicações agrícolas. Esta é uma área de interesse devido à agricultura ser um setor estratégico da economia, e haver uma crescente demanda de soluções computacionais neste campo de atuação. Para tanto foram contextualizados estes dois temas, mostrando o uso conjunto de ambos. O desenvolvimento do trabalho foi baseado em uma revisão bibliográfica cobrindo agricultura de precisão, avaliação de desempenho e *benchmarks* bem difundidos. Foi realizado uma comparação dos *benchmarks* disponíveis, para selecionar aqueles que representavam melhor as características de interesse para tal aplicação. A partir desta seleção, um conjunto de *benchmarks* foram aplicados ao equipamento e os resultados listados ao final do trabalho.

**Palavras-chave:** *benchmark*, sistemas embarcados, agricultura de precisão.

# *Abstract*

This work intends to execute a set of benchmarks in a embedded system focused on precision agriculture. This is an area of interest due to agriculture be a strategical section in economy, and this section are having a growing demand for computing solutions. To achieve these, the two themes were contextualized and showed their combined use. The developed work was based on a literature review covering precision agriculture, performance evaluation and benchmarks. It was conducted a comparison of available benchmarks to select those that best represented the characteristics of interest for this application. From this selection, a set of benchmarks were applied to the equipament and the results listed at the end of this work.

**Keywords:** benchmark, embedded systems, precision agriculture.

# *Dedicatória*

Dedico este trabalho a todos os meu amigos da faculdade, pois estavam sempre comigo seja nas horas de diversão, seja na semana de provas. A meus pais e meus irmãos por sempre me apoiarem apesar de algumas divergências.

# *Agradecimentos*

Agradeço ao Prof. Dr. Maximiliam Luppe por me aceitar como orientando apesar da situação difícil em que encontrava. Agradeço também aos funcionários da USP que sempre estiveram lá quando eu precisei, Silvana, Denise e Shirley e ao Prof. Dr. Evandro Luís Linhari Rodrigues pelas conversas que tivemos. E acima de tudo à Deus, pois sem ele nada disso seria possível. Agradeço também à Jacto - Máquinas Agrícolas, por durante o período de estágio ter permitido a execução dos *benchmarks* em um de seus equipamentos.

# *Sumário*

# *Lista de abreviaturas e siglas*

AP Agricultura de precisão

HCI *Human-computer interaction*

ACM *Association for Computing Machinery*

GNSS *Global Navigation Satellite Systems*

SIG Sistema de informação geográfica

TRK *Real Time Kinematics*

MIPS *Millions of Instructions Per Second*

FLOPS *FLoating Point Operations Per Second*

OBD-II *On-Board Diagnostic* II

TPMS *Tire Pressure Monitoring System*

kWIPS *kilo-Whetstone Instructions Per Second*

MWIPS *Millions of Whetstone Instructions Per Second*

# 1 Introdução

Sistema embarcado é aquele no qual o *hardware*, micro processado, é dedicado ao sistema que ele controla. Diferentemente de computadores, que executam várias tarefas genéricas, sistemas embarcados realizam um conjunto de tarefas predefinidas, geralmente com requisitos específicos. Sendo então um sistema dedicado à tarefas específicas, pode-se aperfeiçoar o projeto reduzindo tamanho, recursos computacionais e custo do produto de acordo com a aplicação. Devido ao fato de conseguir reduzir recursos, estes sistemas estão cada vez mais presentes no cotidiano, controlando, monitorando e comunicando.

Em geral, sistemas embarcados têm componentes de *hardware*, *software* e componentes mecânicos adicionais, e executam funções dedicadas e compartilham propriedades semelhantes. Alguns exemplos de tais propriedades são recursos limitados, como memória, peso, espaço e energia. Tais sistemas estão ganhando mais espaço devido ao aumento da demanda por novas funções em equipamento, devido também às novas necessidades dos consumidores, aos requisitos de segurança e à concorrência do mercado. Estes sistemas também ganham destaque devido normas regulatórias que restringem emissões de poluentes, consumo de energia e água, e definem os requisitos de operação do equipamento.

Seguindo Butazzo [Butazzo 2006] e Berger [Berger 2002], a maioria dos sistemas embarcados divide propriedades importantes:

- **Recursos limitados:** muitos sistemas embarcados são desenvolvidos sobre restrições de espaço, peso e energia, impostos pela aplicação;

- **Sensíveis a custo:** frequentemente apresentam também restrições de custo devido à produção em massa e grande competição industrial. Consequentemente, aplicações embarcadas tipicamente operam em pequenas unidades de processamento com grande limitação de memória e potencial computacional, sendo que para obtenção de custos efetivos é mandatório o uso altamente eficiente dos recursos computacionais;

- **Limitações de tempo real:** a maioria dos dispositivos embarcados interage com o am-

biente e deve reagir a eventos externos e executar atividades computacionais dentro de restrições precisas de tempo, sendo necessária a previsibilidade e garantia off-line dos requisitos de desempenho;

- **Comportamento dinâmico:** consistem de dezenas ou centenas de tarefas concorrentes que interagem entre si para o uso de recursos compartilhados;

- **Diferentes processadores:** sistemas embarcados são suportados por uma grande quantidade de processadores e arquitetura de processadores.

Os requisitos listados acima podem ser mandatórios, desejáveis ou opcionais, e os diferentes tipos de sistemas embarcados apresentam diferentes necessidades para esses requisitos. Em geral, requisitos de qualidade e confiabilidade são maiores em sistemas embarcados se comparados à outros sistemas computacionais [Noergaard 2005].

Atualmente, é difícil encontrar na vida diária segmentos que não envolvam sistemas embarcados de alguma forma. Eles estão espalhados em diferentes áreas como indústria automotiva, eletrônica de consumo, aviônica, controle industrial, instrumentos médicos e dispositivos de rede (*hubs*, *gateways*, roteadores etc.). A tabela 1.1 mostra exemplos de aplicações de sistemas embarcados nos diferentes mercados.

Outra aplicação de sistemas embarcados está na agricultura, principalmente na agricultura de precisão (AP) . A agricultura, desde os primórdios da colonização, foi a base da economia brasileira. Além de base, a agricultura foi e talvez ainda seja um dos setores econômicos mais estratégico para a estabilização da economia iniciado com o plano Real[Baer 2003]. Apesar da palavra agrícola, pelo menos no Brasil, remeter à algo rudimentar, sem tecnologia, distante de computadores, este setor vem se utilizando de cada vez mais tecnologia a fim de melhorar a produtividade, pois com a globalização os produtores não tem apenas concorrentes locais, mas também concorrentes americanos, russos, argentinos dentre vários outros.

Muitas empresas por falta de recurso técnico/financeiro, ou mesmo para não divergir de seu foco de atuação, preferem comprar soluções computacionais, *hardware* principalmente, disponíveis no mercado a desenvolver uma solução específica para seu nicho de aplicação. Uma empresa que desenvolve tais soluções é a alemã Wachendorff. Em especial, ela desenvolve unidades para maquinário agrícola, como por exemplo o Opus A3s e Opus A1rvc. Estas unidades possuem todo um sistema computacional embarcado, processador, memória, sistema operacional, vários mecanismos de entrada e saída e até mesmo uma tela LCD sensível ao toque para facilitar a interação com o operador, tornando este dispositivo um verdadeiro sistema de interface homem máquina, em inglês *Humancomputer interaction* (HCI), perfeito para ser utilizado

**Tabela 1.1:** Exemplos de sistemas embarcados e seus mercados. Fonte: [Noergaard 2005]

| **Mercado** | **Dispositivo embarcado** |
|---|---|
| Automotivo | Sistemas de Ignição |
| | Controle de motor |
| | Freios (ABS) |
| Eletrônica de Consumo | Televisores |
| | DVD's, vídeos cassetes |
| | *Tablets* |
| | *Smartphones* |
| | Eletrodomésticos (Refrigeradores, Micro-ondas, Torradeiras) |
| | Brinquedos, Jogos |
| | Telefones, *Pagers* e Celulares |
| | Câmeras |
| | *Global Positioning Systems* (GPS) |
| Controle Industrial | Robótica e Controle de Sistemas (Manufatura) |
| Medicina | Bombas de Infusão |
| | Próteses |
| | Equipamentos de Diálise |
| | Monitores Cardíacos |
| Redes de Comunicação | Roteadores |
| | *Hubs* |
| | *Gateways* |
| Automação de Escritórios | Equipamentos de Fax |
| | Copiadoras |
| | Impressoras |
| | *Scanners* |

quase que diretamente em várias aplicações. A empresa, fundada em 1978, está intrinsicamente ligada à agricultura de primeiro mundo baseado no conceito de operação eletrônica, conceito desenvolvido em 1998.[Wachendorff 2010]

Outra empresa que desenvolve este tipo de equipamento é a Advantech Corporation, situada em Taiwan, que opera diretamente em 18 países. Ela produz PCs industriais, placas embarcadas, e controladores de automação. Segundo a própria empresa, atua em três frentes, plataformas eletrônicas embarcadas, automação eletrônica e serviços eletrônicos e computação aplicada. Pretende-se utilizar um produto desta empresa no trabalho, o Advantech Trek 550, que assim como a linha Opus da Wachendorff, é um sistema completo inclusive com HCI pronto para ser utilizado em vários tipos de operações.[Advantech 2010]

Segundo a *Association for Computing Machinery* (ACM)[Hewett et al. 2009], HCI é uma disciplina preocupada com o projeto, avaliação e implementação de sistemas computacionais

para uso humano. Este é um conceito importante nesta área pois como é um produto comercial, o alvo é o usuário final, e uma boa interface é sempre desejada.

Com toda essa estrutura de empresas interessadas em obter tais plataformas de processamento e empresas que as fornecem, é possível ver uma crescente utilização da computação para assistir tais atividades, um exemplo é o surgimento e utilização dos SIG por parte de produtores. Segundo Aronoff [Aronoff 1989], um SIG "é qualquer conjunto de procedimentos, manual ou automático, baseado em computador usado para armazenar e manipular dados geograficamente referenciados", isto é, localizados na superfície terrestre e representados numa projeção cartográfica.

Pelo o que foi dito acima, é possível ver que a utilização desses sistemas de HCI são uma tendência, contudo como há vários fabricantes de tais plataformas, é preciso um meio de compará-los para escolher o que melhor adequa-se às necessidades do comprador. Esta tarefa pode ser feita através de duas técnicas bastante difundidas, o *benchmark* e o monitoramento do sistema.

*Benchmark* é o ato de executar um programa, um conjunto de programas ou outras operações, a fim de avaliar o desempenho relativo de um objeto, normalmente executando uma séria de testes padrões. O termo "*benchmark*" é também comumente usado para os próprios programas desenvolvidos para executar o processo. Monitor de sistema é uma ferramenta que observa as atividades de um sistema coletando as características relevantes para a análise do sistema.

Existem outras técnicas de avaliação de desempenho sendo a escolha de uma delas dependente do objeto que se deseja avaliar. As técnicas de análise de desempenho obtêm informações que podem ser obtidas a partir do próprio sistema, através das técnicas de aferição ou através de um modelo representativo do sistema, que corresponde ao emprego de técnicas de modelagem[Santana e Santana 2009]. Os *benchmarks* são uma das técnicas de aferição que se adequa à análise de *hardware*, *software* e de sistemas computacionais como um todo. Essa última característica é sua principal vantagem sobre as outras técnicas de avaliação de desempenho existentes.

## 1.1 Motivação

Ter um único fornecedor normalmente é a melhor opção, pois quanto mais homogêneo o sistema, mais fácil será a integração entre os componentes. Mas esta situação nem sempre é verdadeira, tendo um único fornecedor deixa a empresa à mercê deste fornecedor, apesar de ter um poder de negociação melhor, a relação custo/benefício do produto pode ficar prejudicada

uma vez que o produto de tal fornecedor pode não ter as características para todas as aplicações. Por exemplo, um produto pode ter ótima capacidade computacional, mas possuir pouca conectividade, o que não é admissível em algumas aplicações, já outro, pode ter uma ótima interface mas pouco poder computacional. Devido a essas considerações, nessa área um único fornecedor, ou um único produto, pode não ser a melhor opção. Outro problema relacionado à um único fornecedor/produto reside na possibilidade do produto ser descontinuado, se isso acontecer gerará muitos problemas para a empresa que se utiliza dos mesmos, e não é possível ter muito controle desta variável, por isso é interessante as empresas sempre terem uma segunda opção.

## 1.2 Objetivo

No contexto apresentado, este trabalho visa executar um conjunto de *benchmarks* em um desses equipamentos, mais especificamente no TREK-550 da Advantech, por meio da aferição de características importantes para este tipo de sistema embarcado. Assim, no futuro será possível executar esse mesmo conjunto de *benchmarks* em outro sistema e assim conseguir compará-los de alguma forma, ou tendo algum tipo de especificação para uma aplicação saber se o equipamento é adequado. As características aferidas pelos *benchmarks* foram cálculo computacional com inteiro e ponto flutuante, operações em arquivos e em memória.

## 1.3 Estrutura da monografia

O presente capítulo serviu para uma introdução dos assuntos relacionados com o trabalho a fim de contextualizá-lo neste ambiente multidisciplinar. O capítulo 2 explica melhor a AP, a qual utiliza o tipo de equipamento analisado, bem como algumas áreas correlatas. No capítulo 3 é discutido o embasamento teórico por trás dos *benchmarks* e outras maneiras possíveis de avaliação de sistemas computacionais. O capítulo 4 fornece um detalhadamente do equipamento analisado na seção Material. Os *benchmarks* utilizados e como eles foram utilizados no trabalho também são descritos nesse capítulo, na seção Métodos. Os resultados estão expostos no capítulo 5 e a conclusão do estudo está contemplada no capítulo 6.

# 2 *Agricultura de Precisão*

Uma das primeiras imagens que vêm à cabeça das pessoas ligadas à tecnologia é a imagem de um computador, e isso tem fundamento, uma vez que este é um dispositivo altamente tecnológico, e a cada dia torna-se mais onipresente. Tão onipresente que até mesmo os produtores rurais estão começando a se utilizar dele para melhorar sua produtividade.

No contexto da agricultura, a AP compreende um conjunto de metodologias e técnicas para otimizar o cultivo e os insumos agrícolas usados. A AP permite portanto usar de modo racional os fertilizantes e agrotóxicos garantindo a redução dos impactos ambientais decorrentes da atividade agrícola.

Há relatos de que AP é usada desde o início do século XX. Porém, foram nos anos 80 na Europa que foi gerado o primeiro mapa de produtividade e nos EUA fez-se a primeira adubação em doses variadas. Mas o grande impulso na AP deu-se após o surgimento do GPS em torno de 1990. Sistemas Globais de Navegação por Satélite, em inglês *Global Navigation Satellite Systems* (GNSS) , trata-se de um termo genérico para se referir aos sistema de navegação por satélite. Neste momento existem dois sistemas a operar, o GPS (Norte-americano) e o GLONASS (Russo). Encontram-se, ainda outros dois em desenvolvimento, o Galileo (Europeu) e o Compass (Chinês). No Brasil, ainda que muito esparsas, atividades de AP datam de 1995 com a importação de equipamentos, especialmente colhedores com monitores de produtividade.[Filho et al. 2009]

Com isto pode-se perceber que o cenário da AP, principalmente no Brasil, é recente e exige equipamentos para o processamento das suas características necessárias à AP, como medidor de produtividade, GNSS, SIG além de uma interface amigável para interação do operador com o equipamento. Um Sistema de Informação Geográfica (SIG) é um sistema projetado para capturar, armazenar, manipular, analisar, gerenciar e apresentar tudo o tipo de dado georeferenciado. Em termos simples, SIG é a junção da cartografia, análise estatística e tecnologia de banco de dados.

Sem dúvida foram os sistemas de posicionamento (GNSS) que deram o principal impulso à

AP, que começou com o GPS e posteriormente GLONASS, Galileo e Compass[Filho et al. 2009]. Foi a partir do ano de 1995 que surgiram os primeiros usuário de GPS na agricultura brasileira, nesta época, era necessária a utilização de um mecanismo de correção das coordenadas para tornar o sistema prático e efetivo, devido ao erro do sistema.

Tal erro, intrínseco ao sistema, é causado por vários fatores atmosféricos, como o atraso da ionosfera e troposfera, acesso seletivo, multicaminhos, estabilidade do relógio, entre outros, portanto, é necessário que o *hardware* e *software* embarcados tenham capacidade para, se necessário, implementar alguma correção do posicionamento em tempo real, uma vez que um receptor com uma correção do tipo RTK (*Real Time Kinematics*), a qual melhora a qualidade do posicionamento, pode custar até trinta mil reais.[Spirent Communications plc]

A AP permite otimizar os insumos agrícolas, e sendo a agricultura um setor estratégico, o investimento em tecnologia para melhorar esta área é justificável. Um exemplo de sua importância, é a existência de um congresso apenas para tratar da AP, o ConBAP, no qual são expostos diversos trabalhos nas mais variadas áreas, desde *software* para gestão e apoio na tomada de decisões a métodos de interpolação e amostragem de solo além de muitos outros. Mesmo esta sendo uma área nova no Brasil, a necessidade está fazendo surgir várias soluções para supri-las, tanto que no ConBAP, foram apresentadas diversas empresas e estudos na área de AP.

**Figura 2.1: Vista de uma carreta adubadora**

Alguns exemplos de máquinas utilizadas na AP são carretas adubadoras, figura 2.1, pulverizadoras, figura 2.2, barras de luz, figura 2.3, e colhedoras de café, figura 2.4. Todas essas máquinas refletem a diversidade das aplicações da AP. O principal fator que faz estas máquinas estarem inseridas no contexto da AP é devido à algumas características, por exemplo: a adu-

**Figura 2.2: Vista externa de uma pulverizadora**

badora e a pulverizadora, conseguem operar a taxa variável, ou seja, utilizam-se de um mapa para guiar a dose de insumo aplicada a cada posição do mapa. Já a colhedora de café possui um medidor de produtividade, sendo possível gerar um mapa com a produção detalhada. E a barra de luz é usada para auxiliar na direção da máquina, uma barra com várias luzes é exibida e conforme a máquina sai da rota estipulada pelo mapa, as luzes da barra vão se acendendo permitindo uma direção mais precisa.


**Figura 2.3: Barra de luz**

Outro fato que tem ajudado na adoção da AP é o avanço na maneira de como a interação do operador com o sistema tem evoluído. Deve-se lembrar que no Brasil, os operadores de tais equipamentos raramente tem um alto nível técnico, por isso a facilidade de interação é primordial neste caso. Os sistemas de HCI atuais estão tão presentes no dia-a-dia, uma fornecedora de sistemas que possibilitam a implementação destas interfaces é a já citada empresa alemã Wachendorff Elektronik GmbH Co. KG, ela possui soluções robustas para aplicações agrícolas/automobilísticas [Wachendorff 2010], com *hard-buttons*, tela sensível ao toque, *encoder*, *LEDs*, *buzzers* e outras formas de interação. Todos estes acessórios ajudam o operador a

**Figura 2.4: Colhedora de Café**

interagir com o sistema.

Como já citado, a Advantech também possui um dispositivo deste, o Trek 550, um sistema embarcado utilizado em veículos e que pode ser utilizado em maquinário agrícola, além de vir com uma tela sensível ao toque, possui diversas interfaces de entrada e saída, rs-232, rs-845, vga, usb, DI/DO, *Ethernet*, *CAN*, *Video in*, GPS, Wi-Fi, *Smart Display*. Todas estas interfaces possibilitam a concentração de várias tarefas num mesmo dispositivo, facilitando a manutenção e a gestão da máquina, eles também possibilitam a inserção de novas maneiras de interação do usuário com o equipamento.

Outra ferramenta muito utilizada junto com a AP são os SIG. Com a expansão da utilização dos sistemas de posicionamento globais (GNSS), o uso dos SIGs tem seguido a mesma tendência. Este sistema não tem sua utilização apenas restrita às atividades agrícolas, suas aplicações estão espalhadas por uma gama de áreas. Inicialmente para aplicações agrícolas eram utilizados SIGs genéricos, mas com a crescente demanda, já existem diversas soluções específicas para a agricultura. Exemplos da utilização de SIG são: sistemas que geram e visualizam mapas com a produtividade, também geram de mapas de taxa variável. No mapa de taxa variável, cada ponto marcado terá uma taxa do produto associado, e o equipamento aplica a dose naquele

ponto. Tudo isso é possível pois as informações contidas no mapa são georeferenciadas, tendo portanto coordenadas relacionadas à cada elemento do mapa.

Neste ponto há outro problema, como estes sistemas SIGs têm que trabalhar com um sistema de posicionamento, deve-se considerar o erro apresentado por este sistema. Segundo [Filho et al. 2009], foram os GNSS que impulsionaram a utilização da AP e SIGs, devendo portanto serem considerados no escopo deste trabalho. E sua participação no trabalho deve-se ao fato de que como tais sistemas de posicionamento com a correção de erro são muito caros, o *benchmark* tem de garantir que, quando for necessária uma implantação desta correção em *software*, o sistema seja capaz de responder sem atrasos significativos. Pois dependendo dos recursos para determinado projeto, a aquisição de um sistema com a correção já pronto pode não ser viável.

# 3 *Avaliação de Desempenho*

Uma medida de desempenho de um sistema computacional corresponde ao resultado de uma avaliação da quantidade de serviços prestados ao longo de um determinado período de tempo. O desempenho de um sistema computacional é, antes de tudo, observado por seus usuários a partir dos tempos de resposta observados. Embora não constitua um parâmetro confiável, a qualidade de um sistema é, de certa forma, avaliada de acordo com o grau de contentamento de seus usuários. Assim, quando há descontentamento entre um grupo de usuários de um determinado sistema, torna-se necessária uma avaliação minuciosa a fim de se determinar possíveis gargalos. A avaliação de desempenho em sistemas computacionais, caso o sistema não exista, deve ser capaz de fazer previsões sobre o comportamento do novo sistema.

Apesar da avaliação de desempenho ser de grande importância, muitas vezes ela é negligenciada, o que acarreta consequências graves para as atividades do sistema em questão, esse descaso, na maioria das vezes é atribuído a dois fatores: à falta de conhecimento de ferramentas para efetuar a avaliação, ou à dificuldade de realizá-la - apesar da utilização dessas ferramentas. De modo geral, dependendo do sistema em foco e do tipo desejado de avaliação, é possível utilizar-se de técnicas de aferição (*benchmarks*, prototipação, coleta de dados), ou modelagem (com solução analítica ou por simulação).

A avaliação de sistemas pode ser efetuada através de técnicas de aferição e as técnicas de modelagem. As técnicas de modelagem são adotadas quando os sistemas computacionais são complexos e as técnicas de aferição são difíceis de serem aplicadas. Já as técnicas de aferição são utilizadas quando se tem o sistema pronto e em uso ou existe um protótipo disponível. Os dois tipos de técnicas possuem vantagens e desvantagens. Uma das desvantagens das técnicas de aferição é quando o sistema a ser testado ainda não existe e é necessária a realização de testes prognósticos sobre o sistema em desenvolvimento. As técnicas de modelagem oferecem grande flexibilidade e baixo custo, podendo ser aplicadas tanto quanto o sistema ainda não está concluído, como quando está, e também quando o uso das técnicas de aferição apresenta um custo elevado.

O planejamento de experimentos designa uma área de estudos da estatística que desenvolve técnicas de planejamento e análise de experimentos. Existe um grande número de técnicas, as quais possuem vários níveis de sofisticação, bem como uma grande quantidade de ferramentas que oferecem suporte as condições necessárias para o planejamento de experimentos. Essas técnicas cobrem todas as possibilidades dos possíveis experimentos; seus diversos fatores, cada qual com seus diferentes níveis; tratamento de replicações; etc. Assim, é de suma importância dentro de avaliação de desempenho, pois possibilitam, por sua vez, uma melhor utilização de suas técnicas e suas ferramentas, assim como a análise dos resultados[Santana e Santana 2009].

Na maioria das vezes, os experimentos são executados sem o devido planejamento prévio, o que pode levar à incerteza sobre os resultados obtidos. Isto ocorre devido ao fato de não haver uma visão satisfatória do sistema a ponto de se ter conhecimento sobre o todo que forma o sistema e as interações entre cada um de seus componentes. Assim, um conhecimento do sistema em observação é um fator básico na execução de uma avaliação de nível melhor. O planejamento de experimentos pode ser usado em casos onde avaliação de desempenho é necessária e apresenta técnicas que levam a melhores resultados com um mínimo de aferições obtidas através de experimentos[Júnior 2010].

Os objetivos para o planejamento de experimentos são a maximização da informação com um número mínimo de experimentos e a identificação dos efeitos dos vários fatores no resultado observado, determinando o quão significante é o efeito de um no resultado observado. Todos estes objetivos convergem para uma melhor qualidade dos resultados dos testes e para a obtenção de um projeto com desempenho superior em termos de suas características funcionais e de sua robustez [Santana e Santana 2009].

Um experimento pode ser definido como um teste, ou uma série de testes, no quais mudanças propositais são feitas nas variáveis de entrada do modelo de modo a se observar e identificar razões para mudanças nas variáveis de saída resultantes [Montgomeryr 2000]. Na área de planejamento e análise de experimentos existem métodos específicos para que conclusões válidas possam ser obtidas. Considerando um sistema a ser avaliado, as seguintes observações podem ser feitas:
-Quais são as variáveis de entrada?
-As variáveis escolhidas são as melhores?
-Quais são os parâmetros de saída?
-Qual é a influência das variáveis de entrada sobre os parâmetros de saída?

O objetivo deste trabalho não é descobrir a relação entre os parâmetros aferidos, mas sim

apenas aferi-los. Portanto este trabalho não será focado no planejamento formal dos experimentos, mas sim na aferição de alguns parâmetros pré-determinados.

Técnicas de avaliação de desempenho são métodos pelos quais informações associadas aos parâmetros significativos à análise de desempenho são obtidas. Essas informações podem ser obtidas a partir do próprio sistema (técnicas de aferição) ou através de um modelo representativo do sistema (modelo analítico ou modelo de simulação). Análise de desempenho pode ser aplicada a três grandes áreas:

- Projeto de componentes de *hardware* e/ou *software* de novos sistemas, como ferramentas de avaliação progressiva durante o decorrer do projeto;
- Seleção de sistemas, como instrumento de decisão na aquisição ou substituição de componentes de *hardware* e/ou *software*;
- Avaliação de sistemas já implementados, para otimização, previsão e planejamento de futuras alterações.

As técnicas de avaliação de desempenho existentes podem mostrar diferentes graus de adequação a cada uma das 3 áreas definidas. As técnicas mais importantes são mostradas na Tabela 3.1, obtida em [Macedo 1979], juntamente com o nível de aplicabilidade a cada um dos casos.

Não é interessante para as indústrias, centros de pesquisas, ou qualquer outra instituição, ter que construir um sistema para só então verificar seu desempenho. A verificação e o teste de um sistema, antes mesmo de este existir, representa uma redução enorme de gastos para o desenvolvedor. Para esse tipo de situação, em que se deseja testar um sistema sem que ele exista, ou que sua experimentação seja bastante complexa, é que se propõe a modelagem. Um modelo nada mais é que a representação de um determinado sistema, com o intuito de evidenciar suas características mais importantes.

Os modelos sofrem alterações com o tempo, pois o estado também se altera dessa forma. Assim, os modelos podem ser classificados em relação ao tempo como: discretos ou contínuos. Uma vez que a alteração de estado em computadores ocorre a intervalos discretos de tempo, para a modelagem desse tipo de sistema são usados também modelos discretos. Existem várias técnicas para modelar um sistema computacional. Três técnicas têm sido amplamente utilizadas e possuem vantagens e desvantagens conforme o domínio de aplicação considerado: redes de fila, redes de petri e *statecharts*. Como este trabalho se baseia nas técnicas de aferição um detalhamento das técnicas de modelagem não será fornecido.

Para a avaliação de desempenho de um sistema, o avaliador deve coletar informações referentes aos parâmetros significativos à avaliação. Técnicas de avaliação são, justamente, métodos para a obtenção dessas informações. Isso pode ser feito através do próprio sistema ou a partir de modelagem do sistema. Os sistemas computacionais atuais atingiram uma grande complexidade tornando a utilização de técnicas de aferição em sua avaliação também bastante complexas. Dessa forma, as técnicas de modelagem têm sido amplamente utilizadas na avaliação de sistemas em geral. Entretanto, no caso específico deste trabalho, como o sistema já está implementado, é mais interessante a aferição de características do mesmo.

Quando o sistema a ser avaliado já existe ou está em fase final de desenvolvimento, o estudo de seu desempenho pode ser feito através da experimentação direta. Pode-se utilizar o sistema e coletar dados a seu respeito diretamente. Em relação à modelagem, as técnicas de aferição apresentam vantagens - observadas através de resultados mais precisos- e desvantagens - a necessidade de se ter um sistema disponível. Nos dois casos, é necessário garantir que a própria técnica não interfira no comportamento do sistema e comprometa os resultados obtidos. Algumas das principais técnicas de aferição são [Orlandi 1995]: os protótipos, os *benchmarks* e a coleta de dados. A seguir, essas técnicas serão discutidas.

**Protótipos**

A construção de protótipos representa uma simplificação de um sistema computacional, mantendo, contudo, a mesma funcionalidade. Os protótipos consideram algumas características em detrimento de outras e podem ser considerados um meio termo entre o sistema final e as expectativas que se tem dele, quando ele ainda não existe. Essa técnica possui um custo menor do que a construção do sistema real, apesar de ainda relativamente elevado em relação às demais técnicas de aferição, *benchmarks*, coleta de dados e modelagem. Há outra dificuldade ao se construir um protótipo: determinar quais características são essenciais ao sistema.

**Benchmark**

*Benchmarks* são programas usados para o teste de *software*, *hardware* ou sistemas computacionais completos [Collin 1993]. Na medição do desempenho de sistemas computacionais, os *benchmarks* podem utilizar tanto tarefas mais gerais (operações de entrada e saída), quanto tarefas específicas (como representação de polígonos ou operações sobre matrizes). Em suma, qualquer aspecto de desempenho de um sistema computacional que importe aos usuários pode ser objeto de medição por meio de *benchmarks*. Entretanto, alguns cuidados devem ser observados em relação à utilização dos *benchmarks*. Primeiro, por se tratar de uma técnica de aferição, deve-se verificar se o próprio *benchmark* não influenciará no comportamento do sistema. Outra dificuldade reside na escolha da unidade usada como referência para a realização

da comparação. A utilização, por exemplo, de unidades como MIPS (*Millions of Instructions Per Second*) ou FLOPS ( *FLoating Point Operations Per Second*) gera bastante controvérsia, pois elas fornecem valores perigosamente absolutos, mesmo diante de fatores distintos (como arquiteturas RISC e CISC), que podem influenciar de sobremaneira os resultados.

**Coleta de dados**

Como mencionado, as técnicas de aferição são métodos que oferecem os resultados mais precisos. Dentre essas técnicas, a coleta de dados é a mais precisa. Uma utilização bastante interessante pode ser dada à coleta de dados: os resultados obtidos por meio dessa técnica podem ser usados para comparações com os resultados obtidos a partir de modelos do mesmo sistema. Esse procedimento pode ser empregado como parte da validação de um modelo. A coleta de dados pode ser realizada através de dois recursos: os monitores de *hardware* e os monitores de *software*. Monitores de *hardware* são *hardwares* específicos empregados para coletar e analisar alguns dados pertinentes ao objeto em estudo [Orlandi 1995]. Os monitores de *hardware* devem ser também não intrusivos, isto é, devem limitar-se a obter os sinais sem alterá-los, mantendo fidelidade aos valores originais. Monitores de *software* são usados nos casos em que se deseja observar características específicas de *software* como, por exemplo, a verificação da existência ou não de uma fila de espera, associada a algum recurso do sistema.

Para o caso particular em que o sistema já foi desenvolvido, as técnicas de aferição (*benchmarks*, protótipos e coleta de dados) podem ser empregadas diretamente. Embora seja uma técnica de aferição, os protótipos se aplicam melhor no caso de sistemas inexistentes ou que se apresentam em fase de construção. Outra possibilidade para esse caso é a utilização de coleta de dados. Como já visto, essa técnica é a que fornece os resultados mais precisos. Entretanto, ela também apresenta problemas. Um ponto importante a se destacar é que, como os dados são coletados durante a execução do sistema, cada coleta apresenta um resultado diferente. Isso porque a utilização real de um sistema, vista por um período de tempo, nunca se repete exatamente da mesma forma. Com isso, a impossibilidade de se reproduzir as medições pode se tornar um problema. Por fim, a utilização de *benchmarks* representa mais uma maneira de se avaliar sistemas existentes, e como as outras, apresenta suas vantagens e desvantagens. Os *benchmarks* podem alcançar resultados bastante precisos e, por utilizar cargas de trabalho definidas por meio de parâmetros fixos, a reprodução de seus resultados não é um problema. Mais um ponto a se destacar é que, para coleta de dados, ainda é necessária a utilização de monitores, sejam de *hardware* ou *software*. Dependendo dos dados que se pretende coletar, obter esses monitores pode não ser trivial. Já os *benchmarks* se apresentam em diversas ferramentas, algumas delas disponíveis publicamente e de fácil aquisição, e assim como mostra a tabela 3.1, os *benchmarks*, são mais adequados quando o sistema já existe.

**Tabela 3.1: Nível de aplicabilidade de cada técnica FONTE:[Macedo 1979]**

| | **Área de Aplicação** | | | | | |
|---|---|---|---|---|---|---|
| **Técnicas** | **Projeto** | | **Seleção** | | **Avaliação** | |
| | ***Hardware*** | ***Software*** | ***Hardware*** | ***Software*** | ***Hardware*** | ***Software*** |
| Protótipo | 3 | 3 | 2 | 2 | 0 | 0 |
| *Benchmark* | 0 | 1 | 3 | 3 | 2 | 2 |
| Modelos Analíticos | 2 | 1 | 2 | 1 | 2 | 1 |
| Simulação | 3 | 3 | 2 | 2 | 2 | 2 |
| Monitoramento | 2 | 2 | 2 | 2 | 3 | 3 |

0- Técnica não aplicável na área.
1- Técnica aplicável, porém inadequada.
2- Técnica aplicável, porém insuficiente (utilizada em conjunto com outras áreas).
3- Técnica satisfatória.

Geralmente, os *benchmarks* de baixo nível medem o desempenho de partes específicas do sistema, como comunicação, *cache*, entrada e saída, memória, sincronização, etc. Mas os *benchmarks* de alto nível, fazer uma análise envolvendo vários componentes do sistema. Estes são usados para verificar o comportamento da máquina quanto a problemas de áreas específicas. Como os *benchmarks* de baixo nível analisam partes específicas, são mais fáceis de serem compreendidos que os de alto nível, visto que os de alto nível tratam de problemas de áreas específicas que às vezes não fazem parte da área de conhecimento do *benchmark*. Portanto, os *benchmarks* de alto nível precisam de uma melhor organização para que funcionem adequadamente ou para que seus resultados sejam mais confiáveis. [Couto 1999]

# 4 *Materiais e Métodos*

## 4.1 Material

O Advantech TREK-550, figura 5.1, é um computador dedicado para frotas de veículos industriais, caminhões, ônibus e taxi como pode ser visto na figura 5.2. De acordo com a Advantech, ele inclui *software* para gerenciamento de energia e proteção de força(*ignition on/off, delay on/off, low-battery monitor*) o que protege o sistema de dano elétrico causados por ruídos e picos. O sistema não possui ventoinha e opera no intervalo de -30$^{o}$C à 70$^{o}$C, devido a seu processador específico para uso veicular, Atom Z510PT (1.1GHz) ou Z520PT (1.3GHz).

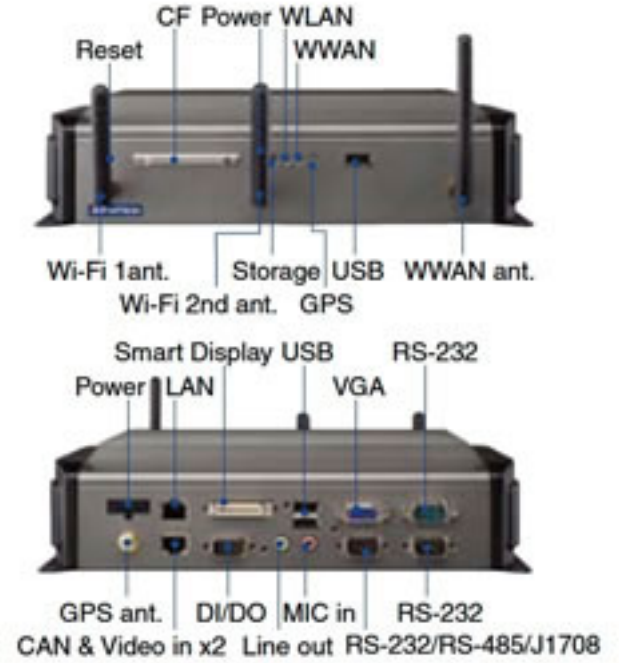


**Figura 4.1: Vista frontal e traseira do TREK-550**

Os vários conectores que o TREK-550 possui ajudam a trabalhar com outros sistemas veiculares como *On-Board Diagnostic* II (OBD-II) e *Tire Pressure Monitoring System* (TPMS). O dispositivo inclui um porto CAN 2.0 A/B, via um conector RJ45, mais um conector DB9

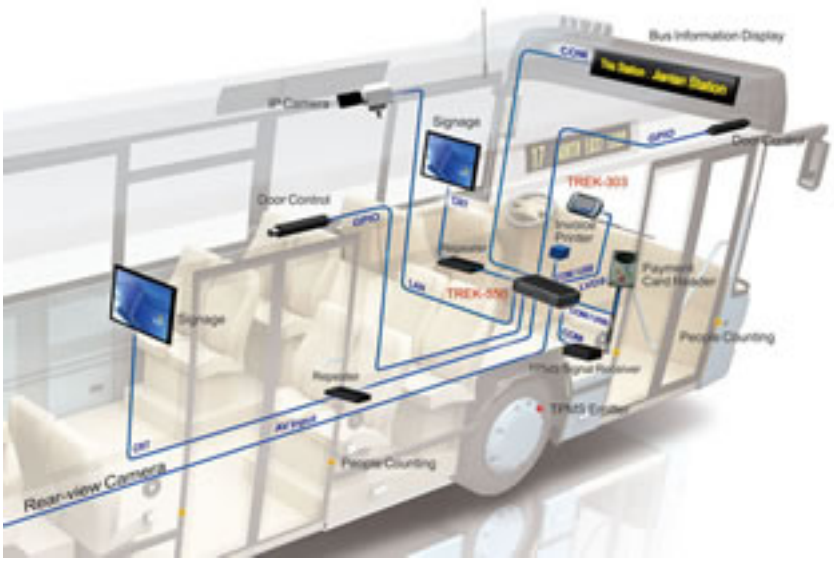


**Figura 4.2: Possível aplicação do TREK-550 em ônibus**

que provê quatro entradas e quatro saídas digitais. Também via o conector DB9 há dois portos RS232, mais um terceiro porto serial que pode ser configurado ou como RS232, RS485 ou J170B.

Outras particularidades presentes no dispositovo são conectores high density "*Smart Display*", figura 5.3, que oferece uma saída 18-bit LVDS para um visor de sete polegadas com uma resolução de 800x480 pixels. No mesmo conector também está presente uma saída de áudio mono, um USB host, 12 VDC Power, e dois portos RS232 adicionais.

Há também uma saída VGA, conector jack para entrada de microfone e saída de som. O conector RJ45 suporta além do barramento CAN duas entradas de video compostas, ambas compatíveis com NTSC, PAL, ou SECAM.

Abaixo está um sumário das especificações técnicas do Advantech TREK-550:

**Processador:** Intel Atom Z510PT (1.1GHz) ou Z520PT (1.3GHz)

**Chipset:** SCH US15WPT

**Memória:** Até 2GB de DDR2 RAM via slot single SODIMM

**Armazenamento:** CompactFlash slot (cartucho com 2GB), ou 40GB/80GB SATA hard disk drives (tamanho não especificado)

**Rede:**

LAN – Gigabit Ethernet

WLAN – 802.11a/b/g (opcional)

WWAN – GSM/GPRS, CDMA, HSDPA (opcional)

PAN – Bluetooth 2.1 (opcional)

**Outras E/S:**

Receptor GPS

interface ”Smart Display”:

saída 18-bit LVDS

2 x RS232

saída de áudio mono

1 x USB host

saída 12VDC

1 x CAN 2.0 A/B via conector RJ45

3 x USB 2.0 host portos

áudio: mic in, line out

3 x serial (2 x RS232, 1 x RS232/RS485/J170B)

4 x portos E/S digital via conector DB9

2 x Entradas de vídeo composto conector via RJ45

1 x Saída VGA

**Requisitos de energia:** 6 to 36VDC

**Temperatura de operação:** -30$^{o}$C to 70$^{o}$C

**Dimensões:** 266 x 149 x 68.2mm

Peso: 2kg

**Sistemas Operacionais suportados:** Linux, Windows CE 6.0, Windows XP Embedded, Windows XP

## 4.2 Métodos

Esta seção fornecerá um detalhamento dos *benchmarks* utilizados.


**Figura 4.3: *Smart Display* TREK-303L**

### 4.2.1 DBENCH

Este *benchmark* apenas produz uma carga no sistema de arquivos. Ele faz as mesmas chamadas ao sistema de arquivos que seriam produzidas pelo NetBench, mas sem efetuar nenhuma chamada de rede. NetBench é um *benchmark* usado para determinar o desempenho de um servidor de arquivos em uma rede local. O NetBench utiliza vários clientes para gerar a carga no servidor de arquivos.

O dbench lê e carrega um arquivo de descrição chamado client.txt que foi derivado de um farejador de rede, programa que captura todo o tráfego circulando em determinada rede, de uma execução real do NetBench. O arquivo client.txt utilizado tem cerca de 25MB e descreve as milhares de operações que um cliente NetBench faz, o arquivo é processado e usado para produzir a mesma carga sem precisar fazer nenhuma chamada de rede, e portando sem precisar de um grande laboratório. Para maiores informações sobre a carga do *benchmark* fornecida através do arquivo client.txt visitar os links:

http://dbench.samba.org/web/iscsi-loadfiles.html

http://dbench.samba.org/web/nfs-loadfiles.html

http://dbench.samba.org/web/smb-loadfiles.html

Uma questão que algumas pessoas podem fazer é se a abordagem acima representa uma carga real. Ela não representa, quase 90% das operações que o NetBench faz é leitura/escrita

enquanto que na carga ”normal” de um escritório, a leitura é dominante. A carga também é muito maior que a de um escritório normal, não há muitas situações em que muitos computadores fazem a escrita de centenas de megabytes em um servidor em poucos minutos.

### 4.2.2 MBW

O mbw (*memory bandwidth benchmark*) determina a largura de banda para cópia de memória de programas no espaço de usuários (*user space*). Sua abordagem simplista modela programas reais. Ele não está configurado para extremos e nem para uma arquitetura específica de *hardware*. Utiliza 3 métodos de cópia, `memcpy`, `dumb` e `mcblock`.

O método `memcpy` utiliza a função `memcpy` da `libc`, biblioteca padrão da linguagem C, para fazer a cópia do vetor de variáveis do tipo *long* alocado pelo programa, todo o vetor é copiado de uma vez. O método mcblock também utiliza a função memcpy para a transferência, mas ao invés de transferir todo o vetor de uma vez, esta opção faz a transferência em blocos do tamanho do número fornecido na entrada do *benchmark*. No caso a entrada foi 250, que diz ao programa para transferir 250 MB entre vetores, este método então fará a transferência em blocos de 250 *bytes* entre os vetores. O terceiro método utilizado faz a transferência utilizando o indexador de vetor da linguagem C, o colchetes, para fazer a transferência. Este método faz um loop sobre o vetor e usa o indexador para transferir. Para cada método é computado o tempo levado para transferir os vetores, sendo denotado pela cadeia de caracteres `Elapsed` na saída do *benchmark*, este tempo está computado em segundos, esse tempo é usado no cálculo da taxa de transferência, última coluna da saída.

Ele alocará dois vetores em memória e copiará um para o outro. A ”banda” reportada é a quantidade de dado copiado pelo tempo gosto na operação de cópia. Obviamente mbw precisa de duas vezes o tamanho dos vetores em megabytes de memória física.

### 4.2.3 FBENCH

O fbench é um algoritmo completo de *ray tracing*, técnica de gerar uma imagem traçando o caminho da luz através dos *pixels* de uma imagem plana e assim simular o efeito de seu encontro com objetos, sem interface do usuário e reformulado em código C portável. Não apenas determina a velocidade de execução de uma aplicação que usa extensivamente ponto flutuante (incluindo funções trigonométricas), como também checa a precisão de um algoritmo muito sensível à erros. O desempenho deste programa é tipicamente muito sensível a mudanças na eficiência da biblioteca de rotinas trigonométricas que um programa comum que usa ponto

flutuante.

Relatórios imprecisos deste *benchmark* devem ser levados à sério. Este programa demonstrou ser invariante a mudanças do formato de ponto flutuante, desde que esse formato seja reconhecido como um formato de precisão dupla. Tais imprecisões normalmente referem à problemas na implementação das bibliotecas matemáticas como a das funções trigonométricas. Este *benchmark* foi originalmente criado para comparar o desempenho de diferentes computadores e implementações da linguagem C. Com o passar dos anos, entretanto, o *benchmark* foi portado para uma variedade de linguagens.

### 4.2.4 Dhrystone

O *Dhrystone* se tornou um *benchmark* popular para o desempenho de processador/compilador, em particular na área de minicomputadores, *workstations*, *PC* e microprocessadores. Isto aparentemente satisfaz a necessidade para facilidade de uso em *benchmarks* de inteiros. Também fornece um primeiro indicador de desempenho, o qual é muito mais significativo que *MIPS*, no seu significado literal, não deve ser usado para diferentes conjuntos de instruções como *RISC* ou *CISC*.

*Dhrystone* inicialmente foi publicado em Ada, versão em Pascal e C também eram distribuídas via *floopy disk* por Reinhold Weicker. Mas a versão mais utilizada foi a tradução feita por Rick Richardson. Uma segunda versão do *benchmark* foi necessária para ter apenas uma versão do *benchmark* garantindo assim consistência das medidas. Outro problema resolvido com a segunda versão foi a prevenção da supressão partes do código que ocorre em alguns compiladores durante a geração do código, remoção de código ”morto” e eliminação de variáveis ”mortas”, os quais, segundo o compilador não eram utilizados pelo *benchmark*. Isto levava ao perigo dos resultados tornarem se sem significado.

Normalmente é difícil ver a diferença exata entre código otimizado ou não. Alguns compiladores fazem certas ações por padrão enquanto que outros apenas se explicitamente solicitado. Também não é possível evitar que as pessoas tentem alcançar melhores resultados no processo de *benchmark*. Portanto, otimizações feitas pelo compilador não são proibidas quando a execução do *Dhrystone* é medida. *Dhrystone* não tem a intenção de ser não otimizado, mas ter uma otimização similar aos programas normais. Por exemplo, há muitos lugares em que o desempenho do *Dhrystone* é beneficiada de otimizações como eliminação de sub expressões comuns e propagação de valores, mas programas normais geralmente também se beneficiam destas otimizações. Entretanto, relatórios de medidas devem indicar o nível de otimização do compilador, e níveis de otimização diferentes para o mesmo *hardware* são desencorajados.

O *dhrystone* compara o desempenho do processador ao de uma máquina de referência, o que é considerado por muitos como uma vantagem em relação à utilização direta da métrica *MIPS*. A justificativa para tanto é que usar máquina de referência efetivamente compensa as diferenças na complexidade das instruções, onde comparar os números de *MIPS* de uma arquitetura *RISC* com os de uma *CISC* não é considerado válido por muitos pesquisadores [Ide 2008].

O *benchmark* não computa nada significativo, mas é sintática e semanticamente correto. Todas as variáveis têm um valor atribuído antes de serem usadas. Não foi levado em conta efeitos de cache, ou balanceamento entre o uso de variáveis *long* e *short*.

Alguns pontos específicos do *Dhrystone*:

- *Dhrystone* é uma medida da eficiência do processador mais compilador executando programas ”típicos”. O programa ”típico” foi projetado para medir estatísticas de um grande número de programas ”reais”. O programa ”típico” foi projetado por Reinhold P. Weicker usando estas estatísticas. O programa é balanceado de acordo com instruções e dados.

- *Dhrystone* não usa ponto flutuante.

- *Dhrystone* não faz E/S. Programas típicos fazem, mas esta modelagem conjunta ficaria muito complexa.

- *Dhrystone* não contêm muito código que pode ser otimizado por processadores vetoriais.

- *Dhrystone* não é perfeito, mas é muito melhor que outros *benchmark* como *Sieve of Eratosthenes*, o qual gera números primos no seu cálculo.

- *Dhrystone* dá um resultado em *dhrystones*/segundo, que quanto maior melhor. Como medida de comparação o *IBM PC* de referência tem cerca de 300-400 *dhrystones*/segundo com um bom compilador.

### 4.2.5 Whetstone

O whetstone é um *benchmark* sintético para avaliar o desempenho de computadores. Foi originalmente escrito em Algol 60 em 1972 no *National Physical Laboratory* no Reino Unido. Foi desenhado para seu modelo não ser afetado pelas otimizações do compilador, a versão em FORTRAN, a qual se tornou o primeiro *benchmark* de propósito geral que criou um padrão na indústria para o desempenho computacional.

O whetstone originalmente media o poder computacional em unidades de kilo instruções whetstone por segundo (*kilo-Whetstone Instructions Per Second*, kWIPS). Isto foi posteriormente mudado para milhões de instruções whetstone por segundo (*Millions of Whetstone Instructions Per Second*, MWIPS). Este *benchmark* primariamente mede o desempenho de aritmética de ponto flutuante.

# 5 *Resultados*

Uma questão que foi observada é que com relação à execução do teste, tomou-se o cuidado de sempre executar nas mesmas condições para todos os *benchmarks*, senão os resultados poderiam apresentar algum vício. Como mesmas condições, é entendida a execução dos testes da mesma maneira e com o estado do sistema o mais próximo possível entre os teste. Todo teste era executado após o sistema ser ligado e esperava-se 10 minutos antes do teste para evitar que o sistema estivesse fazendo alguma operação como atualizações e checagens durante o *benchmark*, o equipamento também estava desconectado de qualquer rede de comunicação.

O dispositivo utilizado nos testes foi o já descrito Advantech TREK-550 rodando o sistema Linux Ubuntu 10.04. Cada experimento foi executado 20 vezes para poder ter algumas métricas estatística dos testes.

Na sequência são exibidos os resultados para cada *benchmark*.

## 5.1 DBENCH

O repositório do dbench é http://dbench.samba.org/web/download.html. A versão utilizada foi a 4.0 de 2001. A compilação foi simples, sendo preciso apenas executar o comando:

```
sudo apt-get install dbench
```

O processo de instalação sugerido pelo repositório consiste na execução dos seguintes comandos na pasta do código fonte:

```
./autogen.sh
./configure
make
make install
```

Mas esse processo não foi concluído com sucesso devido a erros no código fonte do repositório. Na execução do comando *make* foram exibidos vários erros de referência de variáveis. O concerto de tais erros se mostrou um pouco complexo por envolver constantes de outras bibliotecas como 'POPT_AUTOHELP' e erros de sintaxe o que dificultou esse processo. Portanto optou-se pelo aplicativo apt-get.

Este *benchmark* recebe como entrada apenas um número que informa o número de clientes a simular. A ideia de se utilizar este *benchmark* foi testar operações no sistema de arquivos, por isso este *benchmark* foi executado simulando apenas um cliente. Outra razão para não utilizar mais clientes é que em sistemas embarcados principalmente os dedicados à AP, não há muita concorrência de clientes para acessar o sistema de arquivos do processador central do sistema. Normalmente os componentes periféricos do sistema são apenas sensores ou outras unidades de processamentos que enviam dados pré-processados à unidade central de processamento.

Na saída deste *benchmark* é exibido um sumário das operações no sistema de arquivos executadas bem como as medidas resumo finais que são o *Throughput* e a latência máxima observada. As operações descritas são operações executadas pelo samba, o qual simula um servidor Windows, em que na primeira coluna é exibida o nome da operação, NTCreateX, Close, Rename ..., na segunda o número de vezes que tal operação foi executada e por fim a latência média e máxima para cada operação.

Abaixo nas figuras 5.1 e 5.2 são exibidas as respostas dos testes executados pelo dbench bem como algumas medidas resumo dos testes, na tabela 5.1.

**Tabela 5.1:** Medidas-resumo da execução do dbench

| | **Throughput (MB/sec)** | **Latência máxima (ms)** |
|---|---:|---:|
| Mínimo | 30,55 | 2055,56 |
| Média | 31,11 | 3132,78 |
| Desvio padrão | 0,41 | 631,38 |
| Máximo | 32,36 | 3931,51 |

## 5.2 MBW

Neste trabalho, foi utilizada a versão 1.1 de Abril de 2006 obtida em http://ahorvath.web.cern.ch/ahorvath/mbw/. A compilação foi simples, sendo preciso apenas executar o comando:

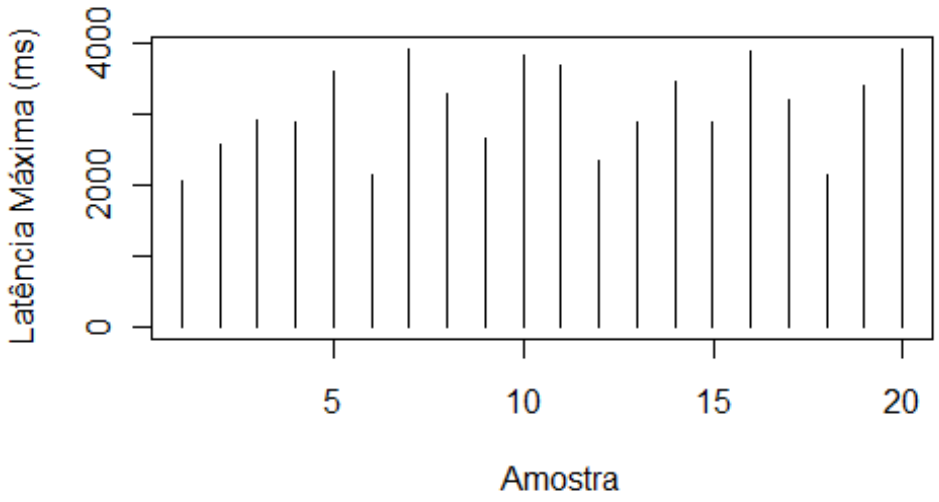


**Figura 5.1: Latência máxima na execução do dbench**

```
make mbw
```

na pasta do código fonte.

Na execução deste *benchmark* utilizou-se como parâmetro de entrada 250, que diz ao programa para fazer transferências de 250 MB. Porque acima deste valor a resposta do *benchmark* demora muito.

Na saída do *benchmark*, é possível ver a divisão em 3 partes, uma para cada método utilizado pelo *benchmark* para efetuar a transferência de memória.

Abaixo nas figuras 5.3, 5.4 e 5.5 são exibidas as respostas dos testes executados pelo mbw bem como algumas medidas resumo dos testes, na tabela 5.2.

**Tabela 5.2:** Medidas-resumo da execução do mbw

| | **MEMCPY** | **DUMB** | **MCBLOCK** |
|---|---|---|---|
| | **Transferência (MiB/s)** | **Transferência (MiB/s)** | **Transferência (MiB/s)** |
| Mínimo | 478,90 | 1000,25 | 1421,32 |
| Média | 483,30 | 1003,54 | 1564,59 |
| Desvio padrão | 2,86 | 4,26 | 102,54 |
| Máximo | 488,83 | 1011,57 | 1771,59 |

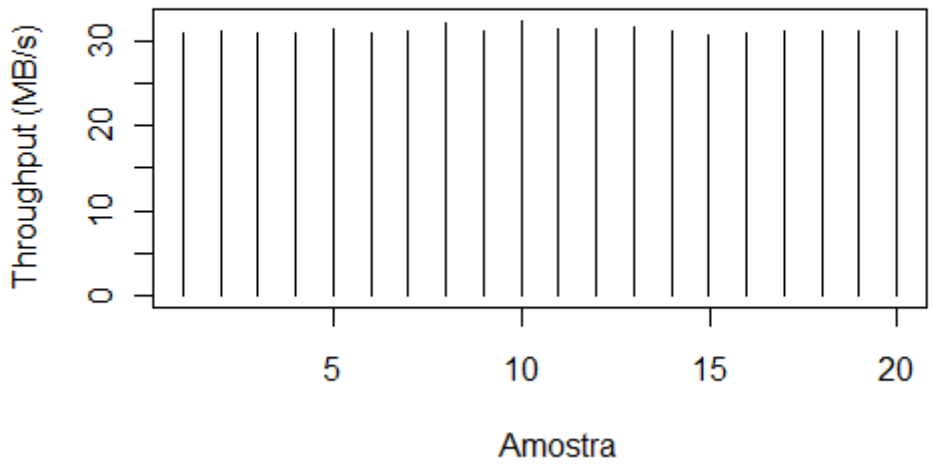


**Figura 5.2: *Throughput* na execução do dbench**

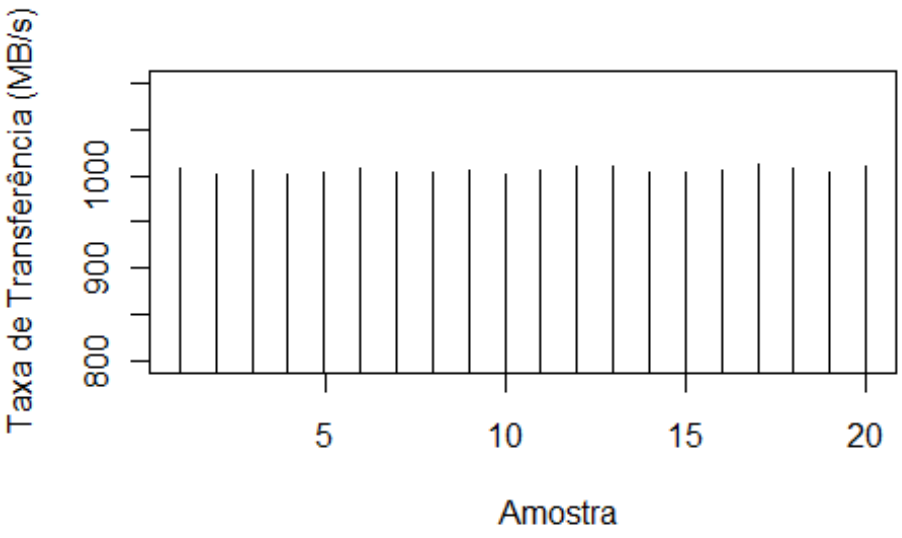


**Figura 5.3: Taxa de transferência do método dumb do *benchmark* mbw**

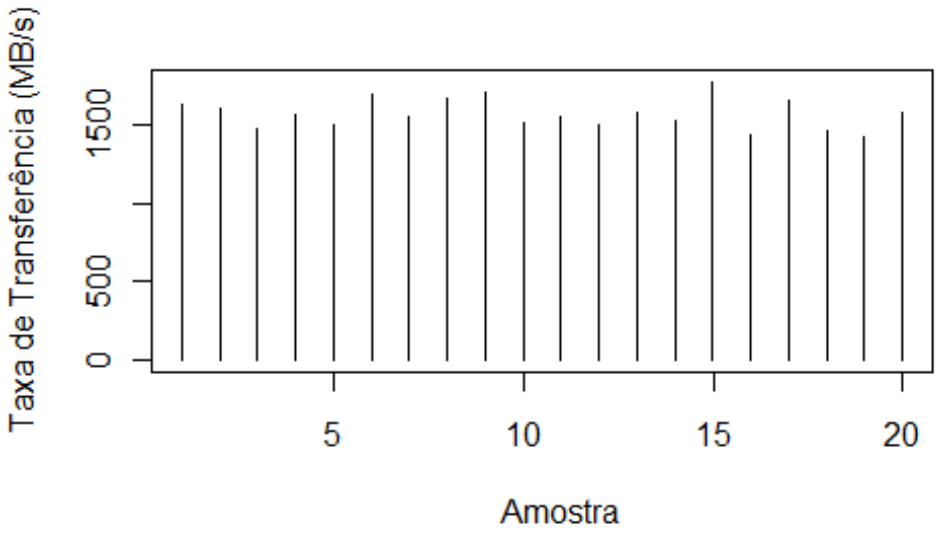


**Figura 5.4: Taxa de transferência do método mcblock do *benchmark* mbw**

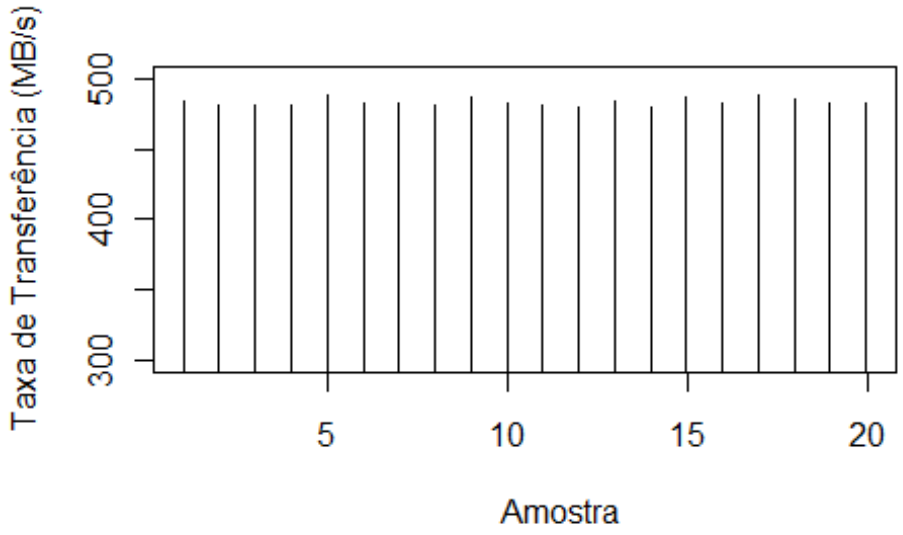


**Figura 5.5: Taxa de transferência do método memcpy do *benchmark* mbw**

## 5.3 FBENCH

O repositório do *benchmark* é http://www.fourmilab.ch/fbench/ e é mantido pelo criador do fbench, John Walker. A compilação foi simples, foi preciso apenas executar o comando:

```
make fbench
```

na pasta do código fonte.

Na execução deste *benchmark*, o parâmetro de entrada, o número de interações, deve ser ajustado de tal forma que o *benchmark* execute por cerca de 5 minutos. O valor requerido para este tempo de execução foi 32000000.

Como ele não retorna o tempo de execução do *benchmark*, utilizou-se a função time do linux para medir o tempo gasto. Esta função retorna o tempo de execução do programa, este tempo é dividido entre o tempo real gasto (real), o tempo gasto pelo sistema (sys), e o tempo gasto pelo usuário (user). Esta divisão é feita, pois como há um SO rodando por trás existe o chaveamento de contexto entre o espaço de usuário, tempo do usuário, e o espaço de kernel, tempo do sistema. Um exemplo do retorno de tal função pode ser observado abaixo:

```
real 0m0.996s
user 0m0.964s
sys 0m0.004s
```

Abaixo nas figuras 5.6, 5.7 e 5.8 são exibidas as respostas dos testes executados pelo mbw bem como algumas medidas resumo dos testes, na tabela 5.3.

**Tabela 5.3:** Medidas-resumo da execução do fbench

| | **Tempo real (s)** | **Tempo usuário (s)** | **Tempo sistema (s)** |
|---|---|---|---|
| Mínimo | 3,08 | 3,13 | 0,00005 |
| Média | 3,16 | 3,15 | 0,004 |
| Desvio padrão | 0,05 | 0,005 | 0,003 |
| Máximo | 3,24 | 3,15 | 0,009 |

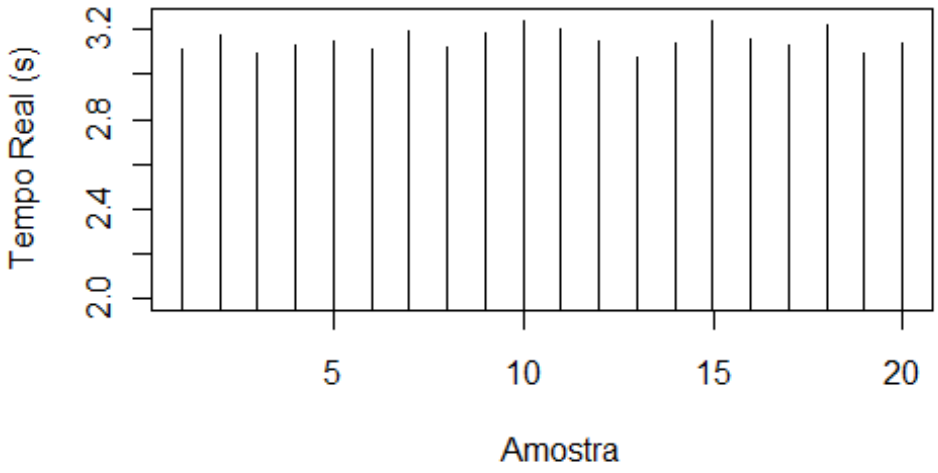


**Figura 5.6: Tempo real de execução do fbench**

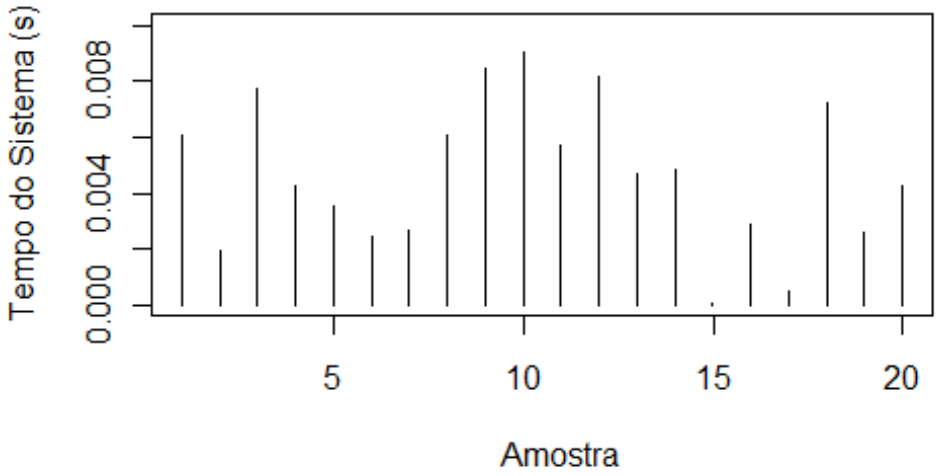


**Figura 5.7: Tempo do sistema da execução do fbench**

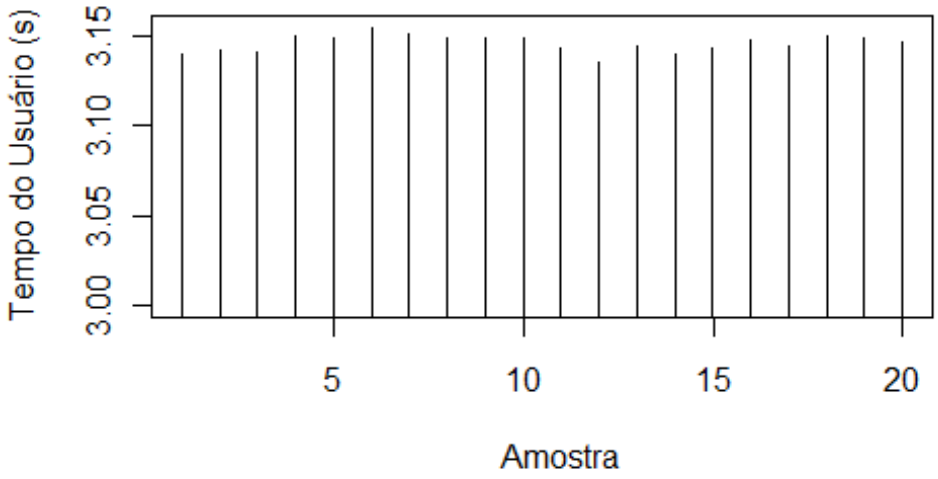


**Figura 5.8: Tempo de usuário da execução do fbench**

## 5.4 Dhrystone

A versão utilizada neste trabalho foi obtida no portal openpkg.org, um repositório de *software UNIX* ( http://download.openpkg.org/components/cache/dhrystone/dhry2.1.tar.Z ). A compilação foi simples, foi preciso apenas executar o comando:

```
make dry2
```

na pasta do código fonte. Um problema encontrado nesse processo foi o conflito da função times, declarada no arquivo de cabeçalho sys/times.h. Para resolver este problema foi necessário apenas comentar a linha 48

```
extern  int     times ();
```

do arquivo dhry_1.c.

Na entrada deste *benchmark* foi fornecido como o número de execuções o número 1.000.000.000, uma vez que a partir de 10.000.000 execuções, o valor de saída do *benchmark* não variou muito e para valores acima de 10.000.000.000 a execução demora muito.

Abaixo na figura 5.9 são exibidas as respostas dos testes executados pelo mbw bem como algumas medidas resumo dos testes, na tabela 5.4.

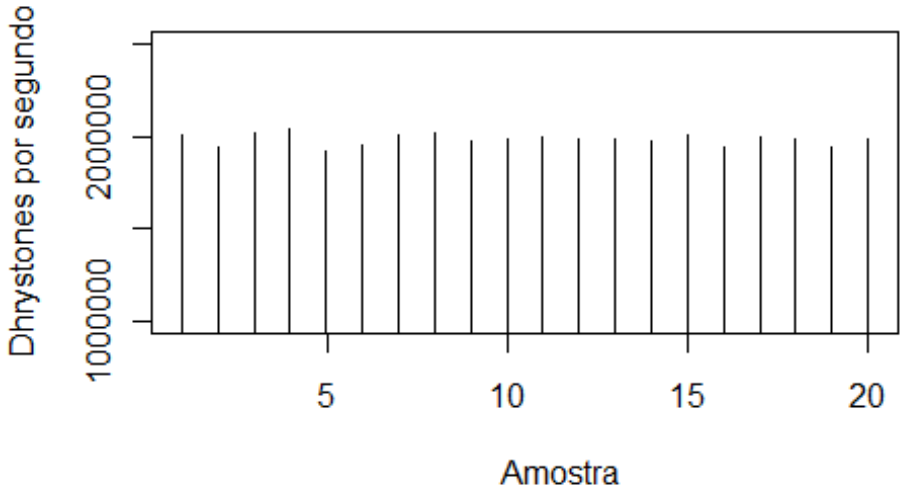


**Figura 5.9: Resultado do *benchmark* Dhrystone**

**Tabela 5.4:** Medidas-resumo da execução do dhrystone

| | **Dhrystone/s** |
|---|---|
| Mínimo | 1921466 |
| Média | 1978052,07 |
| Desvio padrão | 38321,09 |
| Máximo | 2033297 |

## 5.5 Whetstone

A versão utilizada do whetstone foi obtida em: http://www.fourmilab.ch/fbench/. A compilação foi simples, foi preciso apenas executar o comando:

```
gcc whetstone.c -o whetstone
```

na pasta do código fonte.

O whetstone pede como parâmetro de entrada apenas o número de *loops* que o algoritmo irá rodar. O valor fornecido à entrada, 1000000, foi ajustado para não ter uma execução deste *benchmark* tão demorada. Como resultado o whetstone retorna a duração e o número de whetstone MIPS (*Million Instructions Per Second*) da execução.

Abaixo na figura 5.10 são exibidas as respostas dos testes executados pelo mbw bem como algumas medidas resumo dos testes, na tabela 5.5.

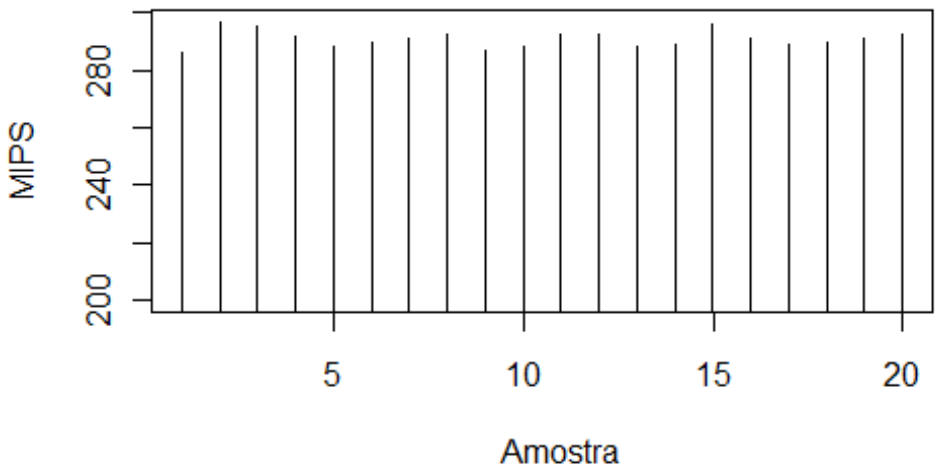


**Figura 5.10: Resultado do *benchmark* Whetstone**

## 5.6 Discussão

No geral, conseguiu-se pouca variação nas execuções dos vários testes para cada *benchmark*, o que mostrou consistência dos resultados. Tal consistência foi obtida devido ao cuidado

**Tabela 5.5:** Medidas-resumo da execução do whetstone

| | **MIPS** |
|---|---|
| Mínimo | 285,93 |
| Média | 288,78 |
| Desvio padrão | 4,12 |
| Máximo | 296,80 |

na execução dos testes, em que esperou-se o sistema estabilizar antes de executar os testes. A única exceção foi à latência máxima do dbench, que teve um desvio padrão da ordem de 20% da média. Isto pode se dever ao estado do sistema de arquivos, que pode ocasionar mais acessos a disco, atividade demorada, ao invés de trabalhar com os *buffers* do disco.

Na execução do mbw, os métodos *DUMB* e *MCBLOCK*, obtiveram os melhores resultados. *MCBLOCK* teve o melhor resultado geral se considerado o valor absoluto de transferência de dados, a taxa do pior teste é consideravelmente maior que a taxa do melhor teste do método *DUMB*. Mas teve um desvio padrão de 6% da taxa média, enquanto que o método *DUMB* obteve um desvio padrão de apenas 3% da taxa média. Como o método *DUMB* obteve menor variação, é mais adequado quando é preciso ter uma certeza maior sobre o tempo de execução, como é o caso dos sistemas de tempo real. O método *DUMB* é, portanto, mais determinístico e mais adequado a este último caso.

A execução do fbench, não detectou nenhum erro de cálculo, mostrando que as bibliotecas matemáticas estão bem implementadas. Este *benchmark* obteve uma resposta geral com pouca variação. O tempo do sistema por outro lado teve uma variação alta. Mas como esse tempo computa o tempo gasto no espaço de kernel e o chaveamento de contexto, não é estranho esta grande variação. Apesar desta variação, o maior tempo gasto pelo sistema equivale a apenas 0,3% do menor do tempo real gasto, sendo, portanto desprezível.

O Dhrystone e o Whetstone obtiveram uma baixa variação, com desvio padrão representando 1,9% e 1,4%, respectivamente. Estes *benchmarks* modelam operações matemáticas com inteiros e pontos flutuantes, e esta pequena variação mostra que estas operações, no sistema analisado, são adequadas à aplicações de tempo real.

A proposta do trabalho é ter estes valores para quando necessário conseguir compara dois equipamentos utilizados para o mesmo fim, mas com este resultados, também é possível escolher e verificar, algumas técnicas e situações em que o sistema analisado é adequado. Como pode-se perceber pelo resultados do método *MCLOCK* e *DUMB* em que pode-se ver que um é

mais adequado para sistemas de tempo real que o outro, a latência máxima do dbench também indica que o uso intenso de operações no sistema de arquivos pode não ser adequadas à sistemas de tempo real.

# 6 *Conclusões e trabalhos futuros*

## 6.1 Conclusões

A proposta deste trabalho era a execução de alguns *benchmarks* num sistema embarcado presente nos veículos empregados na AP. Conseguiu-se executar certos *benchmarks* para aferir algumas características interessantes deste tipo de sistema. Este trabalho mostra que é possível portanto utilizar-se dessas ferramentas para ter uma medida do sistema em questão e poder decidir qual o melhor equipamento para determinada aplicação.

Este trabalho tem portanto apenas a alusão de mostrar o poder computacional do Advantech TREK-550 em alguns dos seus mais variados aspectos. Estes valores podem ser utilizados futuramente para checar a adequação do equipamento à determinada aplicação, principalmente se souber de antemão quais características esta aplicação exige do *hardware*. Por exemplo, tendo-se os resultados dos *benchmarks*, se a aplicação a ser desenvolvida exigir operações em memória os resultados do mbw serão úteis, se intenso cálculo de inteiro, o dhrystone pode dar uma idéia se o equipamento é adequado. Deve-se lembrar que o melhor *benchmark* sempre é a própria aplicação que se deseja implementar, sendo esta a melhor forma de se averiguar se o *hardware*/*software* estão adequados à aplicação.

## 6.2 Dificuldade Encontradas

O manuseio de alguns *benchmark* se mostraram mais complexo do que o esperado, ou mesmo não existiam *benchmarks* para testar tal característica e com isso não foi possível a execução de alguns *benchmarks* que testariam alguns parâmetros importantes como *Qt*, *sqlite* e *CAN*. Outros fatores que limitaram os testes foram a dificuldade de encontrar *benchmarks* livres que rodassem nos sistemas alvos, e a dificuldade para encontrar um *benchmark* pronto e de licensa livre que avaliasse a rede *CAN*.

## 6.3 Trabalhos Futuros

Os trabalhos que podem ser desenvolvidos futuramente são:

- análise entre *benchmarks* de baixo e alto nível a fim de deixar apenas os *benchmarks* mais representativos;

- acrescentar mais características a serem testadas, *CAN* e *sqlite* por exemplo, e se necessário desenvolver um *benchmark* para isto;

- automatizar a execução dos *benchmarks* para não ser necessária nenhuma intervenção durante a execução. Como alguns exigem uma configuração inicial a cada execução, é preciso interagir com o programa sempre, e como cada iteração pode demorar cerca de uma hora, caso do *lmbench*, isto deixa a automatização prejudicada;

- adequar os *benchmarks*, quando necessário, para o padrão ANSI C, de modo a ser compilado na maioria dos sistemas operacionais embarcados, quando de um compilador disponível.

# *Referências Bibliográficas*

[Advantech 2010]ADVANTECH. *Advantech Corporation - industrial computer, embedded computer, industrial automation, industrial motherboard, network security appliance, digital video surveillance, panel PC, industrial IO*. 2010. Disponível em: <http://www.advantech.com/>.

[Aronoff 1989]ARONOFF, S. *Geographical information system: a management perspective*. [S.l.]: Ottawa: WDL, 1989.

[Baer 2003]BAER, W. *A Economia Brasileira*. [S.l.]: Nobel, 2003. ISBN 9788521311973.

[Berger 2002]BERGER, A. S. *: An Introduction to Processes, Tools and Techniques*. [S.l.]: CMP Books, 2002.

[Butazzo 2006]BUTAZZO, G. Research trends in real-time computing for embedded systems. *ACM SIGBED Review*, Julho 2006.

[Collin 1993]COLLIN, S. M. H. *Michaelis: Dicionário Prático de Informática*. [S.l.]: Melhoramentos, 1993.

[Couto 1999]COUTO, C. L. S. *Um Estudo Comparativo de Benchmarks Paralelos*. Dissertação (Mestrado) — Universidade de São Paulo, Novembro 1999.

[Filho et al. 2009]FILHO, J. C. R. et al. *Agrilcultura de Precisão Boletim Técnico*. [S.l.], 2009.

[Hewett et al. 2009]HEWETT, T. T. et al. *Curricula for Human-Computer Interaction*. [S.l.], 2009. Disponível em: <http://old.sigchi.org/cdg/index.html>.

[Ide 2008]IDE, F. H. *Avaliação de métricas para determinar o grau de heterogeneidade de sistemas computacionais*. Dissertação (Mestrado) — Universidade de São Paulo, Março 2008.

[Júnior 2010]JÚNIOR, L. A. P. *Planejamento de experimentos com várias replicações em paralelo em grades computacionais*. Dissertação (Mestrado) — Universidade de São Paulo, Fevereiro 2010.

[Macedo 1979]MACEDO, L. T. de. *Análise do desempenho de computadores: Avaliação, controle e otimização*. [S.l.]: Edgard Blucher, 1979.

[Montgomeryr 2000]MONTGOMERYR, D. C. *Design and Analysis of Experiments*. [S.l.]: John Wiley and Sons, Inc., 2000. ISBN 0471316490.

[Noergaard 2005]NOERGAARD, T. *Embedded System Architecture - A comprehensive Guide for Engineers and Programmers*. [S.l.]: Elsevier, 2005.

[Orlandi 1995]ORLANDI, R. C. G. de S. *Ferramentas para Análise de Desempenho de Sistemas Computacionais*. Dissertação (Mestrado) — Universidade de São Paulo, 1995.

[Santana e Santana 2009]SANTANA, M. J.; SANTANA, R. H. C. *Avaliação de desempenho - planejamento de experimentos. Notas de aula.* 2009. Disponível em: <http://lasdpc.icmc.usp.br>.

[Spirent Communications plc]SPIRENT COMMUNICATIONS PLC. *Testing GNSS System Errors*. Application note dan002 issue 1-01. Aspen Way, Paignton, Devon TQ4 7QR.

[Wachendorff 2010]WACHENDORFF. *Wachendorff - Elektronik Gmbh Co. KG - ABOUT US*. 2010. Disponível em: <http://www.wachendorff-elektronik.de/eng/index.php>.