*Estudantes:*

**Arthur Fernandes Minduca de Sousa –** [**fernandes.arthur@gmail.com**](mailto:fernandes.arthur@gmail.com)

**Carlos Eduardo Buarque Cruz Pimentel –** [**cebcp@cin.ufpe.br**](mailto:cebcp@cin.ufpe.br)

**Carlos Henrique Maciel Sobral Timóteo –** [**chmst@cin.ufpe.br**](mailto:chmst@cin.ufpe.br)

**Karina Rodrigues Pereira –** [**krp@cin.ufpe.br**](mailto:krp@cin.ufpe.br)

*Vínculo:*

**Mestrado Acadêmico**

*Disciplina:*

**Avaliação de Desempenho de Sistemas**

*Professor:*

**Paulo Maciel**

*Atividade:*

**Resolução da 3ª Lista de Exercícios**

# A planilha de dados Planilha1 apresenta uma amostra aleatória simples dos tempos de percurso entre Recife e Maceió dos ônibus da companhia A.

• Gere o histograma dos dados.

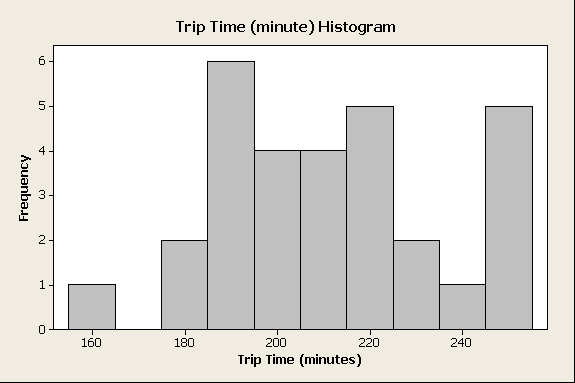


Figura – Histograma dos Dados da Planilha 1

• Gere o resumo estatístico dos dados, e forneça a média, mediana, moda (se houver), média das juntas, média do intervalo, desvio padrão, variância, quartís, intervalo inter-quartil, mínimo, máximo, assimetria e curtose.

As estatísticas descritivas geradas são apresentadas a seguir:

**Descriptive Statistics: Trip Time (minutes)**

***Trip Time*** Média DesvPadrao Variância Mínimo 1º Quartil Mediana

212,49 24,58 604,30 156,25 193,89 212,31

3º Quartil Máximo IQR Moda FreqModa Assim Curtose

229,99 254,96 36,10 219,17 2 0,06 -0,33

MedJuntas MedIntervalo

211,94 205,60

Média das Juntas = Q1+Q3/2

Média do Intervalo = Minimo+Maxino/2

• Faça um resumo textual que interprete os dados analisados e informe se há evidências que indiquem se os dados provêm de uma população com distribuição normal.

O histograma não apresenta uma forma gráfica que seja tolerável assumir como uma distribuição normal.

No entanto, ao analisar as estatísticas descritivas geradas observamos algumas características interessantes:

- Através da execução de um teste de aderência para funções de distribuições de probabilidades (FDP’s), como o Teste de Kolgomorov, a FDP Normal ocupou a 12ª posição no ranking. Em termos de confiança, foi obtido 90,566%, conforme mostra a Figura 2;

- A média e a mediana da amostra são bem próximas. Na FDP Normal, são os mesmos valores;

- A média das juntas é relativamente próxima da média da amostra. Na FDP Normal, são os mesmos valores;

- A assimetria da amostra é maior que zero, porém muito próxima de zero, o que indica a presença de uma cauda sutilmente alongada à direita. Na FDP Normal, a assimetria é zero;

- A curtose é menor que zero e relativamente próxima de zero, o que indica um leve achatamento. Na FDP Normal, a curtose é zero.

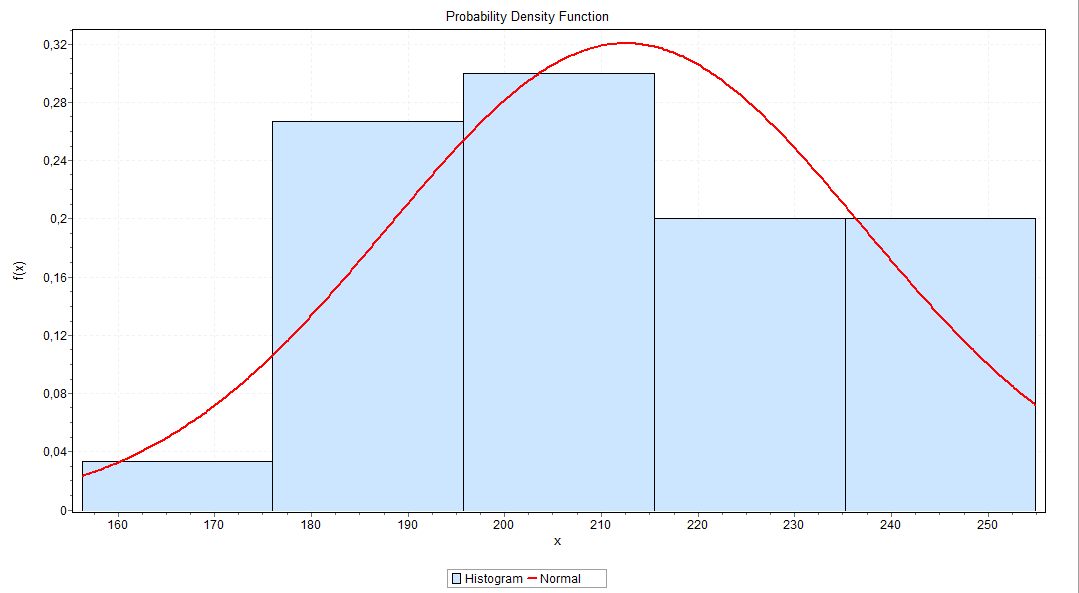




Figura – Teste de Kolgomorov para a amostra de dados da Planilha 1.

Portanto, podemos concluir que, estatisticamente, a amostra provém de uma população com FDP Normal, mesmo que não seja graficamente aparente, devido ao tamanho reduzido da amostra.

• Utilizando os dados da Planilha1, faça pelo menos 60 amostras aleatórias de tamanho 20, calcule a média destas amostras, gere o histograma das médias e o resumo estatístico.

As amostras foram geradas e suas médias são apresentadas a seguir:

**Médias:**

215,004/ 215,567/ 217,977/ 217,910/ 208,552/ 209,650/ 211,916/ 216,321/ 211,912/ 217,572/ 209,810/ 211,640/ 215,346/ 213,294/ 209,190/ 208,640/ 209,396/ 208,080/ 214,997/ 211,324/ 217,353/ 209,352/ 215,744/ 218,761/ 212,266/ 212,650/ 210,742/ 214,116/ 217,958/ 211,597/ 211,900/ 211,924/ 215,269/ 209,826/ 222,150/ 215,020/ 210,184/ 213,134/ 209,871/ 209,636/ 208,650/ 208,624/ 216,406/ 210,140/ 210,513/ 213,406/ 207,605/ 214,297/ 211,792/ 212,236/ 209,926/ 211,060/ 210,549/ 205,838/ 209,355/ 209,727/ 210,643/ 216,163/ 216,205/ 209,161.

**Histograma:**

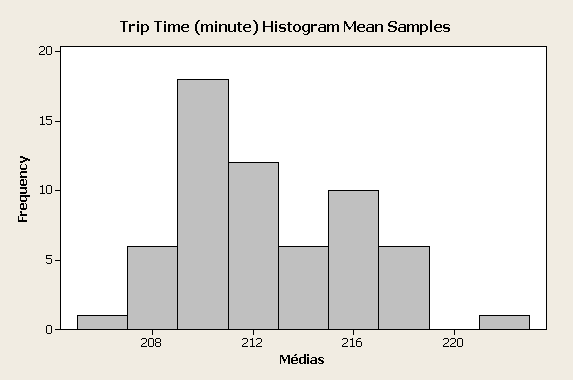


Figura – Histograma das médias de 60 amostras dos dados da Planilha 1.

**Descriptive Statistics: Trip Time (minutes)**

***Trip Time*** Média DesvPadrao Variância Mínimo 1º Quartil Mediana

212,43 3,38 11,42 205,84 209,75 211,85

3º Quartil Máximo IQR Moda FreqModa Assim Curtose

215,21 222,15 5,46 - 0 0,59 -0,20

MedJuntas MedIntervalo

212,48 213,99

Média das Juntas = Q1+Q3/2

Média do Intervalo = Minimo+Maxino/2

• Faça um resumo textual sobre a distribuição das médias (obtidos no passo d), comentando os resultados e descreva seu entendimento sobre o Teorema Central do Limite, dado o estudo realizado neste exercício.

O histograma não apresenta uma forma gráfica que seja tolerável assumir como uma distribuição normal.

No entanto, ao analisar as estatísticas descritivas geradas observamos algumas características interessantes:

- Através da execução de um teste de ajuste automático para funções de distribuições de probabilidades (FDP’s), como o Teste de Kolgomorov, a FDP Normal ocupou a 35ª posição no ranking. Em termos de confiança, foi obtido 87,372%, conforme mostra a Figura 4;

- A média, a mediana, a média das juntas e a média dos intervalos da amostra são próximas. Na FDP Normal, são os mesmos valores;

- A média das juntas é relativamente próxima da média da amostra. Na FDP Normal, são os mesmos valores;

- A assimetria da amostra é maior que zero, porém muito próxima de zero, o que indica a presença de uma cauda sutilmente alongada à direita. Na FDP Normal, a assimetria é zero;

- A curtose é menor que zero e relativamente próxima de zero, o que indica um leve achatamento. Na FDP Normal, a curtose é zero.

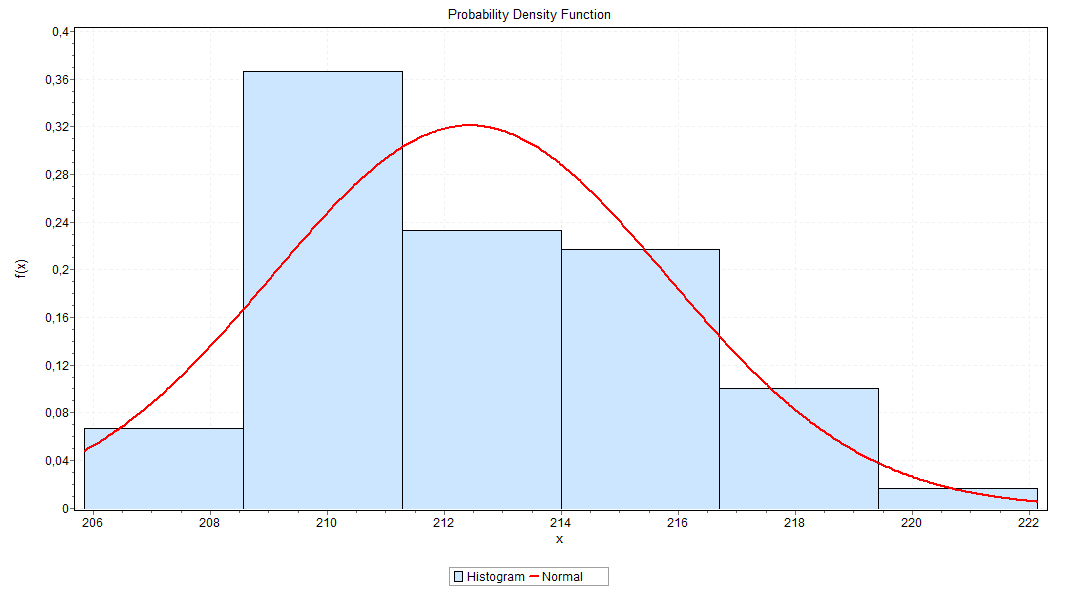




Figura – Teste de Kolgomorov para a amostra de dados das Médias.

Portanto, podemos concluir que, estatisticamente, a amostra provém de uma população com FDP Normal, mesmo que não seja graficamente aparente, devido ao tamanho da amostra.

# Suponha que os dados da Planilha2 provêm de uma amostra aleatória simples referente aos tempos de serviço dos caixas do Banco XYZ, localizado no endereço 123.

• Analise os dados através do histograma; do resumo estatístico dos dados que conste a média, mediana, moda (se houver), média das juntas, média do intervalo, desvio padrão, variância, quartís, intervalo inter-quartil, mínimo, máximo, assimetria e curtose.

**Histograma:**

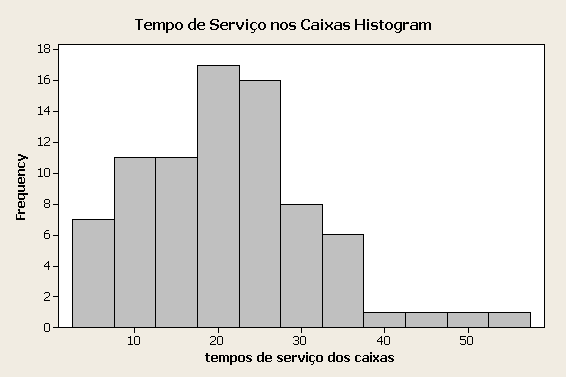


Figura – Histograma dos dados da Planilha 2.

As estatísticas descritivas geradas são apresentadas a seguir:

**Descriptive Statistics: Tempo de Serviço dos Caixas**

***Tempo Serviço*** Média DesvPadrao Variância Mínimo 1º Quartil Mediana

21,01 10,07 101,32 3,10 13,70 21,20

3º Quartil Máximo IQR Moda FreqModa Assim Curtose

26,20 52,80 12,50 13,7;21,6 3 0,60 0,78

MedJuntas MedIntervalo

19,95 27,95

Média das Juntas = Q1+Q3/2

Média do Intervalo = Minimo+Maxino/2

• Faça um resumo textual que interprete os dados analisados, indicando a presença de possíveis *outliers* e informe se há evidências que indiquem se os dados provêm de uma população com distribuição normal.

O histograma não apresenta uma forma gráfica que seja tolerável assumir como uma distribuição normal.

No entanto, ao analisar as estatísticas descritivas geradas observamos algumas características interessantes:

- Através da execução de um teste de ajuste automático para funções de distribuições de probabilidades (FDP’s), como o Teste de Kolgomorov, a FDP Normal ocupou a 9ª posição no ranking. Em termos de confiança, foi obtido 91,194%, conforme mostra a Figura 6;

- A média, a mediana e um dos valores de moda da amostra são bem próximas. Na FDP Normal, são os mesmos valores;

- Conforme ilustrado na Figura 7 há a indicação de *outliers* nessa amostra.

- A assimetria da amostra é maior que zero, porém próxima de zero, o que indica a presença de uma cauda alongada à direita. Na FDP Normal, a assimetria é zero;

- A curtose é maior que zero e relativamente próxima de zero, o que indica um leve alargamento. Na FDP Normal, a curtose é zero.

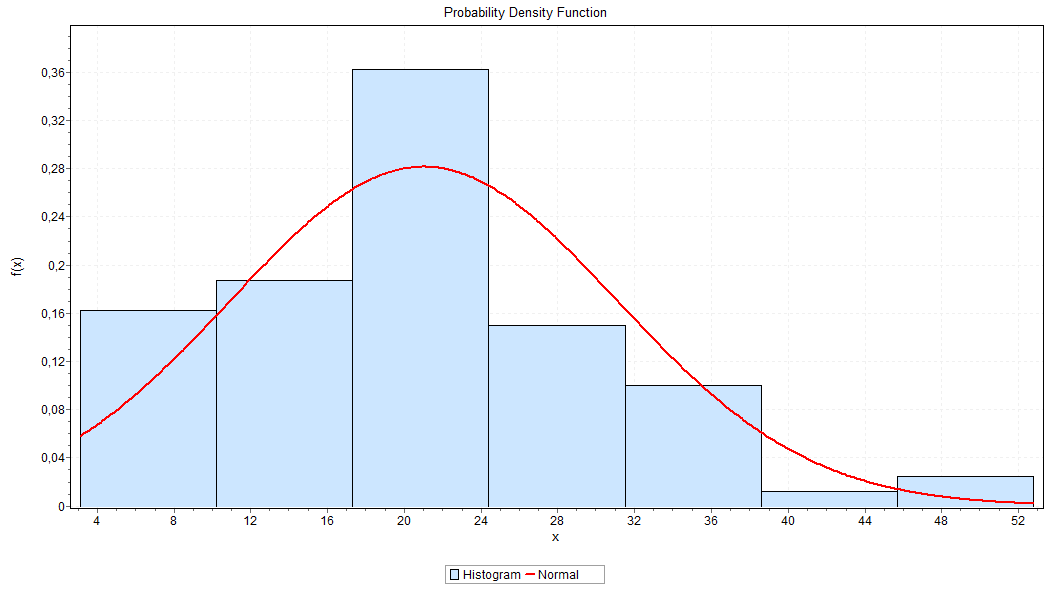




Figura - Teste de Kolgomorov para a amostra de dados da Planilha 2.

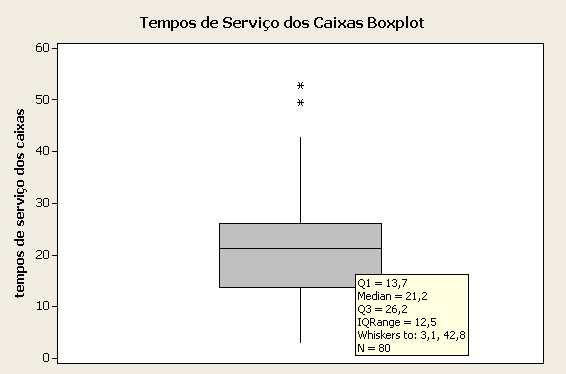
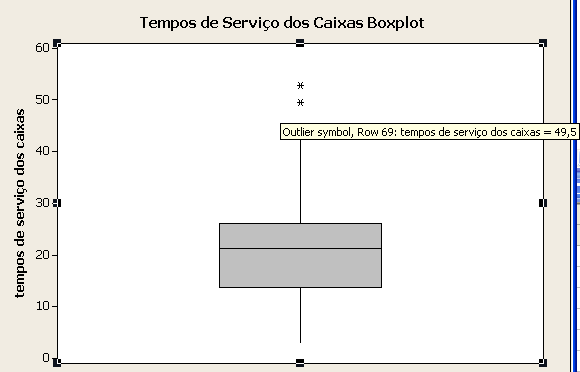


Figura – *Boxplot* indicando a presença de *outliers*.

Portanto, podemos concluir que, estatisticamente, a amostra provém de uma população com FDP Normal, mesmo que não seja graficamente aparente.

• Caso detecte *outliers*, retire-os dos dados e gere novamente o histograma e o resumo estatístico solicitado no passo a.

Os dois *outliers* indicados na seção anterior foram identificados retirados da amostra.



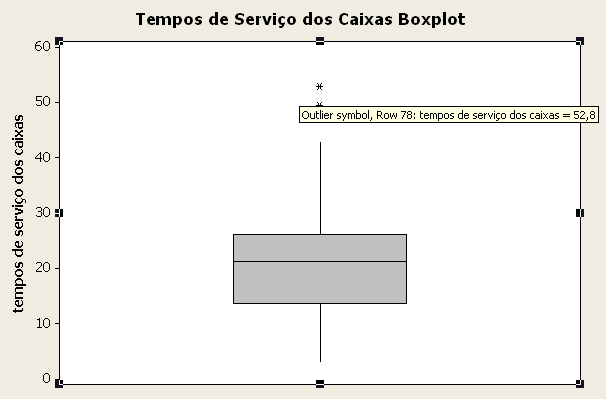


Figura – Identificação dos *outiliers.*

**Histograma sem *outliers*:**

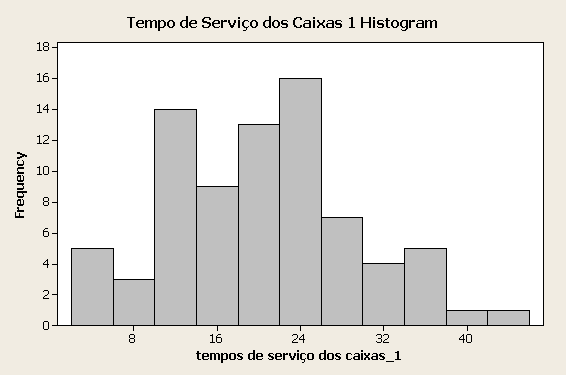


Figura – Histograma dos dados das amostras sem *outliers*.

As estatísticas descritivas geradas são apresentadas a seguir:

**Descriptive Statistics: tempos de serviço dos caixas\_1**

***Tempo Serviço*** Média DesvPadrao Variância Mínimo 1º Quartil Mediana

20,23 8,93 79,67 3,10 13,67 21,10

3º Quartil Máximo IQR Moda FreqModa Assim Curtose

25,83 42,80 12,15 13,7;21,6 3 0,12 -0,37

MedJuntas MedIntervalo

19,75 22,95

Média das Juntas = Q1+Q3/2

Média do Intervalo = Minimo+Maxino/2

• Faça um resumo textual que interprete os dados analisados e informe se há evidências que indiquem se os dados provêm de uma população com distribuição normal.

O histograma não apresenta uma forma gráfica que seja tolerável assumir como uma distribuição normal.

No entanto, ao analisar as estatísticas descritivas geradas observamos algumas características interessantes:

- Através da execução de um teste de ajuste automático para funções de distribuições de probabilidades (FDP’s), como o Teste de Kolgomorov, a FDP Normal ocupou a 3ª posição no ranking. Em termos de confiança, foi obtido 90,566%, conforme mostra a Figura 10;

- A moda e a mediana da amostra são bem próximas. Na FDP Normal, são os mesmos valores;

- Observando as estatísticas, comprova-se que a mediana é imune à *outliers*, o seu valor permaneceu inalterado;

- A assimetria da amostra é maior que zero, porém muito próxima de zero, o que indica a presença de uma cauda sutilmente alongada à direita. Na FDP Normal, a assimetria é zero;

- A curtose é menor que zero e relativamente próxima de zero, o que indica um leve achatamento. Na FDP Normal, a curtose é zero.

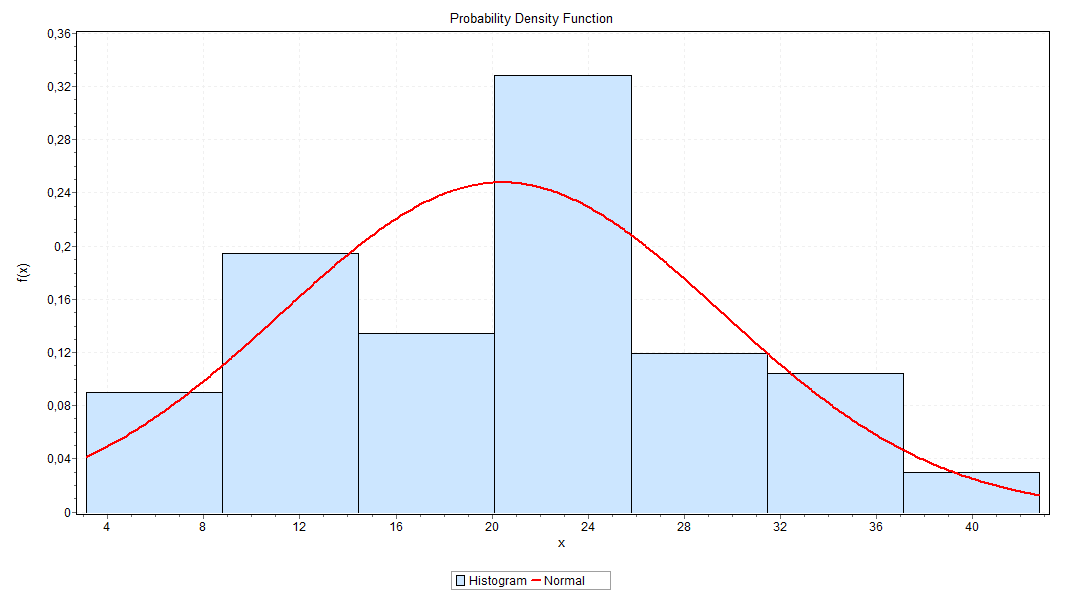




Figura – Teste de Kolgomorov para a amostra de dados da Planilha 2 sem *outliers*.

Portanto, podemos concluir que, estatisticamente, a amostra provém de uma população com FDP Normal, mesmo que não seja graficamente aparente, devido ao tamanho da amostra.

# Descreva detalhadamente os benchmarks Dhrystone, Whetstone e Linpack.

O que é Benchmark?

Podemos dizer que **benchmark** é o ato de executar um programa de computador, um conjunto de programas ou outras operações, a fim de avaliar a performance relativa de um objeto, normalmente executando uma série de testes padrões e ensaios. Também é comumente usado para os próprios programas (de benchmarking) desenvolvidos para executar o processo. Normalmente, benchmarking é associado com avaliação de características de performance de um hardware de computador como, por exemplo, a performance da operação de ponto flutuante de uma CPU, mas há circunstâncias em que a técnica também é aplicável a software. Benchmarks de software são feitos, por exemplo, em compiladores ou sistemas de gerenciamento de banco de dados. Benchmarks provêm um método de comparação da performance de vários subsistemas dentre as diferentes arquiteturas de chips e sistemas. Benchmarking é útil para o entendimento de como o gerenciador de banco de dados responde sob a variação de condições.

Esses programas de teste de desempenho podem ser classificados de acordo com a classe de aplicação para a qual são destinados: computação científica, sistemas distribuídos, aplicações multimídia, processamento de sinais. Benchmarks são desenvolvidos para imitar um determinado tipo de comportamento em um componente ou sistema. Benchmarks "sintéticos" fazem isso através de programas especialmente criados para impor o tipo de comportamento desejado no componente. Em oposição, benchmarks "de aplicação" executam programas do "mundo real" no sistema. Embora normalmente os benchmarks de aplicação ofereçam uma melhor medida para a performance no "mundo real" para um dado sistema, benchmarks sintéticos ainda são usados no teste de componentes individuais, como um disco rígido ou um dispositivo de rede.

**Dhrystone**

O Dhrystone foi publicado pela primeira vez em 1984 por Reinhold Weicker da Siemens em C, Pascal e ADA, na tentativa de medir e comparar o desempenho dos computadores. O objetivo seria que com o Dhrystone, fosse criado um benchmark pequeno e representativo para programação de sistemas (no caso de operações aritméticas). O código do Dhrystone é em grande parte representado por aritmética simples, operações com strings, decisões lógicas e acessos de leitura e escrita de memória com intenção de refletir as atividades da CPU nas aplicações de computação de propósito mais geral. O resultado é determinado através do cálculo do tempo médio que um processador leva para executar as todas as iterações de um laço que, por sua vez, contém uma seqüência fixa de instruções que o compõem.

O programa 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 uma 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. Como os processadores atuais são capazes de executar várias instruções por ciclo e dependem muito da velocidade e quantidade de memória *cache*, o resultado serve apenas como uma referência de desempenho bruto, que não indica necessariamente o desempenho do processador em aplicativos reais.

Inicialmente a indústria adotou o VAX 11/780 como uma máquina de referência de 1 MIPS. O VAX 11/780 alcança 1757 D/S (Dhrystone por segundo). O resultado do benchmark é calculado mensurando o número de D/S para o sistema, e dividindo-se este número por 1757 (da máquina de referência).

Podemos considerar que algumas das vantagens aparentes do Dhrystone também são fraquezas significativas. Os números do Dhrystone refletem na verdade o desempenho do compilador da linguagem C e suas bibliotecas, provavelmente mais do que o desempenho do próprio processador e seu projeto foi baseado na análise de vários outros programas escritos em diferentes linguagens e por diferentes autores, porém voltados à programação de sistemas operacionais e compiladores. Esta é uma característica bastante relevante, pois diferentes classes de aplicações enfatizam diferentes tipos de operações. De acordo com a freqüência das operações nos diferentes programas analisados, foram construídas algumas tabelas que deram origem ao Dhrystone. Em seu código original, em Ada, ele contém 100 sentenças entre o início e o fim do contador de tempo, balanceadas em relação a tipos de sentenças, tipos de dados (operandos) e sua localidade (global, local,parâmetro, constante). O programa contém uma distribuição de 51% de atribuições, 32% de sentenças de controle e 17% de chamadas de funções e procedimentos. O corpo do programa contém 12 procedimentos e, durante um laço (isto é, 1 Dhrystone), as 100 sentenças que compõem o laço são executadas.

O Dhrystone oferece como saída duas métricas: tempo em microsegundos e Dhrystone por segundo. As métricas são inversamente proporcionais, isto é, tempo em microsegundos – quanto maior o valor pior o desempenho; Dhrystone por segundo – quanto maior o valor melhor o desempenho. Seu objetivo é substituir o benchmark denominado Whetstone, mais antigo e menos confiável. O Dhrystone, como a maioria dos benchmarks, consiste em trechos de código padronizados, que são revistos periodicamente para minimizar vantagens que possam favorecer injustamente a determinadas combinações de hardware, compilador e ambiente. O Dhrystone se concentra no tratamento de strings e não utiliza operações com ponto flutuante. A exemplo da maioria dos testes de benchmark, ele é fortemente influenciado pelo projeto do hardware e do linker, a otimização do código, o uso de cache de memória, os estados de espera e os tipos de dados inteiros.

Para o Dhrystone existem duas versões do sistema, a 1.1 e a 2.1.

Versão 1.1 - Nesse caso de tratamento do relógio do PC, o mesmo é atualizado pela função clock() que executa a uma taxa de 18 vezes por segundo, com uma resolução de 0,05. As unidades utilizadas pelo benchmark na versão 1.1 são: Micro segundos, Dhrystones\_Por\_Segundo e Vax\_Mips. O calculo para cada uma das unidades é dado a seguir:

Microseconds = benchtime \* 1000000 / (double) Loops;

Dhrystones\_Per\_Second = (double) Loops / benchtime;

Vax\_Mips = Dhrystones\_Per\_Second / 1757.0;

A função de contagem de inteiros é definida como:

Loops = Loops \* 2;

count = count - 1;

Array2Glob[8][7] = 10;

start\_time();

for (i = 0; i < Loops; ++i)

{

Proc5();

Proc4();

IntLoc1 = 2;

IntLoc2 = 3;

strcpy(String2Loc, "DHRYSTONE PROGRAM, 2'ND STRING");

EnumLoc = Ident2;

BoolGlob = ! Func2(String1Loc, String2Loc);

while (IntLoc1 < IntLoc2)

{

IntLoc3 = 5 \* IntLoc1 - IntLoc2;

Proc7(IntLoc1, IntLoc2, &IntLoc3);

++IntLoc1;

}

Proc8(Array1Glob, Array2Glob, IntLoc1, IntLoc3);

Proc1(PtrGlb);

for (CharIndex = 'A'; CharIndex <= Char2Glob; ++CharIndex)

if (EnumLoc == Func1(CharIndex, 'C'))

Proc6(Ident1, &EnumLoc);

IntLoc3 = IntLoc2 \* IntLoc1;

IntLoc2 = IntLoc3 / IntLoc1;

IntLoc2 = 7 \* (IntLoc3 - IntLoc2) - IntLoc1;

Proc2(&IntLoc1);

}

end\_time();

benchtime = secs;

Os testes envolvidos neste benchmark são: testes aritméticos, testes de overhead. As assinaturas a seguir resumem os procedimentos de testes:

void Proc0();

void Proc1(RecordPtr PtrParIn);

void Proc2(OneToFifty \*IntParIO);

void Proc3(RecordPtr \*PtrParOut);

void Proc4();

void Proc5();

void Proc6(Enumeration EnumParIn, Enumeration \*EnumParOut);

void Proc7(OneToFifty IntParI1, OneToFifty IntParI2, OneToFifty \*IntParOut);

void Proc8(Array1Dim Array1Par, Array2Dim Array2Par, OneToFifty IntParI1, OneToFifty IntParI2);

Versão 2.1 - A diferença está basicamente no tipo de teste que é feito em cima de leitura/escrita de arquivos, entre outros e também a forma que o autor chama de otimização de sistema. As funções assim como escritas na versão 1.1 são resumidas como a seguir:

void Proc\_1 (REG Rec\_Pointer Ptr\_Val\_Par);

void Proc\_2 (One\_Fifty \*Int\_Par\_Ref);

void Proc\_3 (Rec\_Pointer \*Ptr\_Ref\_Par);

void Proc\_4 ();

void Proc\_5 ();

void Proc\_6 (Enumeration Enum\_Val\_Par, Enumeration \*Enum\_Ref\_Par);

void Proc\_7 (One\_Fifty Int\_1\_Par\_Val, One\_Fifty Int\_2\_Par\_Val, One\_Fifty \*Int\_Par\_Ref);

void Proc\_8 (Arr\_1\_Dim Arr\_1\_Par\_Ref, Arr\_2\_Dim Arr\_2\_Par\_Ref, int Int\_1\_Par\_Val, int Int\_2\_Par\_Val);

Boolean Func\_2 (Str\_30 Str\_1\_Par\_Ref, Str\_30 Str\_2\_Par\_Ref);

Também tem mudanças na medida de início de fim da estrutura de loop do benchmark que ao invés do start\_time() e end\_time() como na versão 1.1, a versão 2.1 definem como tipos: Begin\_Time, End\_Time, User\_Time. Cada um desses tipos busca a função da macro timer.h da linguagem C.

Exemplo:

Begin\_Time = dtime();

Mais um exemplo de que varia o exemplo de acordo com o compilador e a linguagem que vai ser utilizada. O benchmark originalmente foi escrito na linguagem ADA que não precisa de várias macros para sua execução. Na transição para a linguagem C/C++ esta influencia diretamente no resultados.

Ao final destes testes é apresentado um resumo dos resultados obtidos, onde temos uma variável Array2Glob, que possui um valor fixo pré-definido afim de atribuir um peso diferente para cada um dos testes executados além de apresentar o número de repetições executados durante os testes. Os resultados gerais são mostrados de forma mais simplificada, tornando possível a comparação entre diferentes máquinas com maior facilidade.

Exemplo:

Compiler Watcom C/ C++ 10.5 Win386

Optimisation -otexan -zp8 -5r

10000 runs 0.00 seconds

100000 runs 0.38 seconds

200000 runs 0.66 seconds

400000 runs 1.43 seconds

800000 runs 2.80 seconds

1600000 runs 5.55 seconds

Array2Glob8/7: O.K. 1600010

Microseconds for one run through Dhrystone: 3.47

Dhrystones per Second: 288288

VAX MIPS rating = 164.08

Dhrystone Benchmark, Version 2.1 (Language: C)

Compiler Watcom C/ C++ 10.5 Win386

Optimisation -otexan -zp8 -fp5 -5r

Register option not selected

10000 runs 0.05 seconds

100000 runs 0.44 seconds

200000 runs 0.94 seconds

400000 runs 1.86 seconds

800000 runs 3.68 seconds

1600000 runs 7.36 seconds

Final values (\* implementation-dependent):

Int\_Glob: O.K. 5 Bool\_Glob: O.K. 1

Ch\_1\_Glob: O.K. A Ch\_2\_Glob: O.K. B

Arr\_1\_Glob[8]: O.K. 7 Arr\_2\_Glob8/7: O.K. 1600010

Ptr\_Glob-> Ptr\_Comp: \* -5959980

Discr: O.K. 0 Enum\_Comp: O.K. 2

Int\_Comp: O.K. 17 Str\_Comp: O.K. DHRYSTONE PROGRAM, SOME STRING

Next\_Ptr\_Glob-> Ptr\_Comp: \* -5959980 same as above

Discr: O.K. 0 Enum\_Comp: O.K. 1

Int\_Comp: O.K. 18 Str\_Comp: O.K. DHRYSTONE PROGRAM, SOME STRING

Int\_1\_Loc: O.K. 5 Int\_2\_Loc: O.K. 13

Int\_3\_Loc: O.K. 7 Enum\_Loc: O.K. 1

Str\_1\_Loc: O.K. DHRYSTONE PROGRAM, 1'ST STRING

Str\_2\_Loc: O.K. DHRYSTONE PROGRAM, 2'ND STRING

Microseconds for one run through Dhrystone: 4.60

Dhrystones per Second: 217391

VAX MIPS rating = 123.73

The above results (less calibration details) are appended to file Dhry.txt,

the version 2 results format being slightly different.

**Linpack**

Este benchmark foi produzido por Jack Dongarra, Jim Bunch, Cleve Moler e Gilbert Stewart, do pacote “LINPACK” das rotinas de álgebra linear. Tornou-se o benchmark básico para aplicações científicas desde a metade de 1980 com uma inclinação para o desempenho do supercomputador.

A versão original foi produzida em Fortran, porém uma versão em “C” apareceu posteriormente. A versão padrão “C” opera em matrizes 100x100, com precisão dupla ou simples e os tipos de loops podem ser rolled/unrolled. A versão pré-compilada é de precisão dupla, rolled, otimizada ou não otimizada. Também existem as versões para DOS e OS/2.

O Linpack faz uso das bibliotecas BLAS (Basic Linear Algebra Subprograms) para realização de vetores básicos e operações de matrizes. É uma medida da potência de computação de ponto flutuante de um sistema. Ele mede o quão rápido um computador resolve um denso N por N sistema de equações lineares Ax = b, o qual é uma tarefa comum em engenharia. A solução é obtida por eliminação Gaussiana com giro parcial, com operações de ponto flutuante 2/3·N3 + 2·N2. O resultado é reportado em milhões de operações de pontos flutuantes por segundo (MFLOP/s, as vezes simplesmente chamado FLOPS).

Para sistemas de memória distribuída em larga escala, Linpack de alto desempenho, uma implementação portátil do benchmarck Linpack de alto desempenho, é utilizado como uma medida de desempenho para qualificar supercomputadores no ranking da lista dos computadores mais rápidos do mundo, TOP500. Agora também existe o ranking Green500 que lista as máquinas baseadas em eficiência energética, em FLOPS por Watt. O benchmark HCP roda para matrizes de diferentes tamanhos N procurando pelo Nmax no qual o máximo desempenho Rmax é obtido. O benchmark também reporta o problema de tamanho N1/2 onde metade do desempenho (Rmax/2) é alcançado.

Uma das rotinas que mais consomem tempo no Linpack é a DAXPY. Arquitetos de computadores projetam sistemas para aperfeiçoar DAXPY com o intuito de obter uma alta avaliação no Linpack. Existem reclamações não verificadas de que o Linpack não seja um bom benchmark para supercondutores porque ele não estressa a interconexão entre nós, mas foca em unidades aritméticas de pontos flutuantes e memória cachê.

Característica interessante do Limpack são a referenciação de duas rotinas : DGEFA e DGESL (estas são rotina que trabalha com ponto flutuante de 64 bits; já as SGEFA e SGESL trabalham normalmente com expressões de ponto flutuante de 32 bits). DGEFA realiza a decodificação parcial do vetor, e DGESL usa esse tipo de decodificação para resolver um determinado sistema de equações lineares. A maior parte das execuções de ponto flutuante gira em torno de O(n³), este é o tempo gasto em DGEFA. Uma vez que a matriz foi decomposta usando o DGESL, que trabalha com tempo de O(n²) operações de ponto flutuante.

Os resultados dos Benchmarks não devem ser utilizados como medidas de desempenho total do sistema (a não ser que a análise foi realizada suficiente para a carga de trabalho de um determinado interesse), mas sim como pontos de referências para novas avaliações.

O desempenho é freqüentemente medido em termos de Megaflops, Gigaflops, ou Teraflops. Incluem geralmente tanto adições e multiplicações na contagem das operações de ponto flutuante por segundo, e os valores dos operandos são assumidos de 64 bits de ponto flutuante. Atualmente, pode se medir o desempenho de um supercomputador chegando a Petaflops (1015flops/s).

Analisando o algoritmo Linpack e observando a forma de como os dados são referenciados, vemos que cada passo do processo de fatorização existe operações vetoriais que modificam completamente uma sub-matriz de dados. Esta atualização faz um bloco de dados ser lido, atualizando e escrevendo de volta para a memória principal.

O número de operações de ponto flutuante é 2 / 3n3, e o numero de dados referenciados, ambos loads e stores, é 2 / 3n3. Assim, para cada par soma/multiplicação temos de efetuar um load(carregar) e store(armazenar) dos elementos, infelizmente não obtendo a reutilização dos dados. Mesmo que as operações sejam completamente vetoriais, há uma movimentação significativa dos dados, resultando em um baixo desempenho. Em computadores vetoriais a um desempenho bem inferior ao pico estimado. Em computadores super-escalares isso resulta em uma grande quantidade de dados movimentados e atualizados. Para alcançar taxas de alto desempenho esta operação-memória-referência deve ser maior.

**Whetstone**

O Whetstone foi o primeiro benchmark sintético na literatura, com fins específicos de medida de desempenho. Apesar de ter sido utilizado na prática em FORTRAN, sua primeira versão foi publicada na linguagem ALGOL 60, em 1976, por H. J. Curnow e B. A. Wichmann. O Whetstone é conciso e modularizado. Cada módulo tem um tipo diferente, explora diferentes recursos da linguagem de programação e é executado inúmeras vezes através de um laço.

As vantagens são o tamanho e a simplicidade do código-fonte, além de explorar as operações em ponto-flutuante. Portanto, servindo de referência para pequenas aplicações científicas em computadores de pequeno e médio porte.

Algumas desvantagens são: só há uma versão oficial em Pascal; o desempenho medido pelo programa depende principalmente da velocidade das funções matemáticas (seno, cosseno, raiz quadrada).

Os resultados exibidos pelo benckmark são bem simples: *loops* que representa a quantidade de repetições do laço, *iterations* que representa o número de execuções do programa, *duration* que representa o tempo de duração do teste e o número de milhões de instruções por segundos executadas (MIPS).