# Benchmark Linpack

**Bruno Colla, Manuel Debarba, Tiago Gasparetto**

Universidade Regional Integrada do Alto Uruguai e das Missões
URI Campus de Erechim – RS – Brasil

Departamento de Engenharias e Ciências da Computação
bruno-colla@hotmail.com, manuelluis@yahoo.com.br,
thiago_gaspareto@yahoo.com.br

***Abstract.*** *This paper describes the LINPACK Benchmark and some of its variations commonly used to assess performance of computer systems. The latter is frequently used to obtained results for TOP500 submissions. Information is also given on how to interpret results of the LINPACK benchmark and how the results fit into performance evaluation process.*

***Resumo.*** *Este artigo descreve o Benchmark Linpack e algumas de suas variações geralmente utilizadas para avaliar o desempenho dos sistemas de computadores. Este é frequentemente utilizado para obter resultados apresentados no TOP500. É emitido também a informação em como interpretar os resultados do benchmark LINPACK e de como os resultados se encaixam em processo de avaliação de desempenho.*

## 1. Introdução

Um Benchmark é um programa de teste de desempenho que analisa as características de processamento e de movimentação de dados de um sistema de computação com o objetivo de medir ou prever seu desempenho e relevar os pontos fortes e fracos de sua arquitetura. Benchmarks podem ser classificados de acordo com a classe de aplicação para a qual são voltados como, por exemplo, computação científica, serviços de rede, aplicações multimídia, processamento de sinais, etc.

O Benchmark Linpack é um dos mais famosos benchmarks atualmente, utilizado inclusive nos testes das 500 máquinas mais rápidas existentes (Top500), por ser o que tem maior número de resultados reportados. Originalmente, o Linpack era um pacote de sub-rotinas que tinham por finalidade resolver sistemas de equações lineares algébricas. O Linpack encaixa-se na classificação de benchmark tipo algoritmo. Ele contém dois conjuntos de rotinas: um para decomposição de matrizes e outro para resolver o sistema de equações lineares resultantes da decomposição. Os resultados dos testes são reportados em MFLOPS (Millions of Floating Point Operations per Second), GFLOPS (Billions of Floating Point Operations per Second) ou TFLOPS (Trillions of Floating Point Operations per Second). Sua aplicação é visivelmente em máquinas que utilizam softwares para cálculos científicos e de engenharia, visto que as operações mais utilizadas nestes tipos de aplicações são em ponto-flutuante.

Existem versões diferentes para o Linpack cada versão diferencia-se no tamanho das matrizes (a mais utilizada é 100x100); na precisão, que pode ser dupla ou simples; e em relação aos tipos de laços (*unrolled/ rolled*). Os laços *unrolled* são incrementados de 4 em 4, e o corpo do laço contém sentenças para índices i, i+1, i+2 e i+3. Em algumas máquinas/compiladores o código executa mais rápido desta maneira. Nas máquinas

vetoriais mais modernas, o compilador já faz isto de maneira mais otimizada e, portanto, elas executam a versão "*rolled*" (laços incrementados de 1 em 1) mais rapidamente.

Com o tempo foram adicionados outros dados de desempenho, chegando a uma coleção de mais de 1300 diferentes sistemas de computadores, além da expansão do escopo do benchmark, podendo resolver problemas de matrizes de três níveis e também problemas de otimização. A partir destas evoluções, o benchmark LINPACK foi adotado por cientistas do mundo todo para avaliar o desempenho de computadores, especialmente para máquinas com arquiteturas avançadas.

## 2. O Pacote Linpack

O pacote LINPACK é um conjunto de sub-rotinas escrito inicialmente em Fortran para resolver sistemas de equações lineares, com algoritmo para resolver um sistema de álgebra linear. Ele foi escrito baseado em outro pacote de sub-rotinas chamado Basic Linear Álgebra Subprograms, que fazem operações comuns de vetor escalar, como multiplicação de um vetor por um escalar e operações entre vetores, por exemplo, produto vetorial, sempre trabalhando com ponto flutuante.

Nos últimos anos, com o surgimento de computadores de processamento paralelo e supercomputadores, as aplicações de software passaram a ser escaláveis, ou seja, capazes de tomar como vantagem a configuração destas supermáquinas e resolver grandes problemas com a mesma eficiência. Portanto, foi criada uma nova categoria de regras de testes e ambientes para o benchmark LINPACK, conhecido como Highly Parallel LINPACK (HPL- Linpack altamente paralelo) NxN Benchmark.

### 2.1. Seleção do algoritmo

No Benchmark Linpack, foi usado originalmente uma matriz de tamanho 100, pelo motivo do computador que estava em uso em 1980 possuir limitações de memória, a matriz tem $O(100^2)$ ponto flutuante de elementos e poderia ser acomodado na maioria dos ambientes da época.

Resolver um sistema de equações requer $O(n^3)$ ponto flutuante de operações, mais especificamente $2/3n^3+2n^2+O(n)$ ponto flutuante de adições e multiplicações. Assim, o tempo necessário para resolver tais problemas em uma determinada máquina pode ser aproximado por

$$time_n = \frac{time_{100} \cdot n^3}{100^3}$$

O algoritmo utilizado é baseado na decomposição parcial, o tipo de matriz é do tipo real com elementos distribuídos aleatoriamente, o gerador de números aleatório utilizado no benchmark não é sofisticado, mas seus principais atributos é a sua compactação.

## 3. Características do Linpack

As duas características 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 $O(n^3)$, este é o tempo gasto em DGEFA. Uma vez que

a matriz foi decomposta usando o DGESL, que trabalha com tempo de $O(n^2)$ operações de ponto flutuante.

Por sua vez DGEFA e DGESL chamam três rotinas que são: DAXPY, IDAMAX, DSCAL. Usando uma proporção de tempo a rotina DAXPY consome 90% do tempo de execução, sua função esta voltada na multiplicação de um escalar α * vetor X, e adicionar os resultados em outro vetor Y. Esta rotina é chamada aproximadamente $n^2/2$ vezes por DGEFA e 2n vezes por DGESL utilizando vetores de comprimento variável. A declaração yi ← yi+α.xi, que constitui um elemento da DAXPY é executado aproximadamente $n^3/3+n^2$ vezes o que dá origem a cerca de $2/3n^3$ operações de ponto flutuante. Assim a referência n=100 requer cerca de 2/3 milhões de operações de ponto flutuante.

A declaração yi ← yi+α.xi, além do ponto flutuante de adição e multiplicação, envolve algumas operações de armazenamento e referências. Enquanto o linpack envolve as rotinas DGEFA e DGESL que referenciam vetores bidimensionais. Uma vez que se diga que o Fortran possui vetores bidimensionais seria armazenado por colunas na memória, o acesso a elementos consecutivos de uma coluna conduz a cálculos de índice simples. As referências a elementos consecutivos são diferenciados por uma palavra, em vez de pela primeira indexação do vetor bidimensional.

### 3.1. Detalhes das Operações

Os resultados refletem em apenas um problema da área: resolver um sistema de Equações Lineares usando o linpack no qual em sua primeira versão era programado em ambiente Fortan. Dado que a maior parte do tempo é gasto em DAXPY, o ponto de referências é realmente medir o desempenho dos DAXPY. A média de duração para decomposição parcial do vetor utilizado é 2/3n. Assim no marco de n=10 o tempo de duração decomposição é de 66.

A fim de resolver este problema inicial é necessário executar quase 700.000 operações de ponto flutuante. O tempo necessário para resolver o problema é dividido pelo número de operações de ponto flutuante para determinar a taxa de Megaflops. As rotinas DGEFA chamadas de IDAMAX, DSCAL e DAXPY têm como função: Rotina IDAMAX que calcula o índice de um vetor com maior módulo, ex: chama-se 99 vezes essa rotina na execução de um vetor executando de 2 até 100.

Cada chamada para IDAMAX da origem a dupla precisão de n, o valor absoluto da dupla computação é n-1. O número total de operações é 5364, os valores absolutos de dupla precisão é 4950. DSCAL é chamada 99 vezes na execução de um vetor de tamanho de 1 a 99. Cada chamada para DAXPY da origem a uma dupla comparação de precisão com zero. Isso leva a 4950 comparações contra 0, 328350 adiantamentos e 328350 multiplicações. Além disso, o próprio DGEFA faz 99 comparações de dupla precisão contra zero, e 99 dupla precisão recíprocas os valores da operação de contagem DGEFA é dada na tabela 1.

**Tabela 1. Contagens das operações de Duplas precisão para Linpack 100x100 da rotina DGEFA.**

| Tipo de Operação | Contador Operações |
|---|---|
| Soma | 328350 |
| Multiplicação | 333300 |
| Reciproca | 99 |
| Valor absoluto | 5364 |
| Menor/Igual | 4950 |
| Diferente de zero | 5247 |

A rotina DGESL, que é utilizada para solucionar um sistema de equações baseadas na fatorização de DGEFA, realiza uma quantia menor de operações de ponto flutuante. Em DGESL, DAXPY é posta em duas, uma vez um vetor com tamanho de 1 até 99, e outra com vetor de tamanho 0 até 99. Isso leva a um total de 9900 acréscimos de dupla precisão, o mesmo número de multiplicações de dupla precisão, e 199 contra zero. O DGESL faz 100 divisões e 100 negações. O valor total da operação de contagem DGESL é dado na tabela 2.

**Tabela 2. Contagens das operações de Duplas precisão para o Linpack 100x100 da rotina DGESL.**

| Tipo de Operação | Contador Operações |
|---|---|
| Soma | 9900 |
| Multiplicação | 9900 |
| Divisão | 100 |
| Negativo | 100 |
| Diferente de zero | 199 |

**Tabela 3. Contagens de operações de Dupla precisão para o Linpack 100x100.**

| Tipo Operações | Contador Operações |
|---|---|
| Soma | 338250 |
| Multiplicação | 343200 |
| Recíproca | 99 |
| Divisão | 100 |
| Negação | 100 |
| Valor Absoluto | 5364 |
| Menor/Igual | 4950 |
| Diferente de zero | 5446 |
| Total | 697509 |

Contam para o linpack referencias indicadas na tabela 3, que apontam um total de 697509 operações de ponto flutuante (o linpack utiliza aproximadamente $2/3n^3+2n^2$ operações que para n-100 tem o valor de 686667.). É instrutivo para olhar apenas a contribuição devida ao DAXPY. Destas Operações de ponto flutuante, as chamadas para DAXPY de DGEFA e DGESL contam para um total de 338160 acrescenta-se

338160 multiplicações e 5147 comparações com zero. Isso da um total de 681467 operações, ou mais de 97% de todas as operações de ponto flutuante que são executadas.

O tempo total é retomado com mais operações aritméticas. Em particular, há um lote de tempo gasto em carregar e armazenar os operando da operação de ponto flutuante. Podemos estimar o número de carregamento e armazenamento, assumindo que todos os operando devem ser armazenados em registradores, mas também assumindo que o compilador irá fazer um trabalho razoável de promover laço, de modo que eles não precisam ser carregados dentro do “circulo fechado” novamente.

### 3.2. Laço Unrolling

É observado frequentemente que a maior parte do tempo do processamento de um programa esta localizado em 3% ou menos do código fonte, muitas vezes o código a partir de uma perspectiva temporal é construído por um ou alguns laços, por exemplo, o produto escalar de dois vetores. Em computadores escalares, é importante usar técnicas simples para a otimização de tais loops. O Loop(laço) unrolling, é uma técnica aplicada para o tempo consumido do laço. Quando um ciclo é desenrolado, o seu conteúdo é replicado uma ou mais vezes, com as devidas adaptações para o índice do vetor e incrementos do laço.

A principal fonte de melhoria em relação aos computadores “simples” é a redução da sobrecarga. Por exemplo, o laço unrolling poderia permitir mais de uma multiplicação a ser ativo simultaneamente em uma maquina segmentada.
Além disso, esse laço frequentemente aumenta a concorrência entre as unidades funcionais independentes em computadores tão equipados ou aqueles com múltiplas instruções adicionais. Um exemplo é o processador IBM Power, que possui unidades independentes e componentes multiplicadores, que pode obter concorrência entre adição e multiplicação de um elemento para o elemento seguinte, além de obter a segmentação ao mesmo tempo dentro de cada unidade.

No entanto, em maquinas com instruções vetoriais, a técnica unrolling tem efeito oposto, o compilador tenta detectar as operações vetoriais no lopp, mas o que dificulta o unrolling é que consequentemente o código do vetor poderá se tornar escalar, e consequentemente degradará o desempenho.

### 3.3. Desempenho

O desempenho do computador é uma questão complicada, possui uma quantidade de muitas funções inter-relacionadas. Estas quantidades incluem a aplicação, o algoritmo, o tamanho do problema, a linguagem de alto nível, a execução, os recursos humanos (nível de esforço utilizado para aperfeiçoar o programa), o compilador para aperfeiçoar a capacidade, a versão do compilador, o funcionamento do sistema, a arquitetura do computador, e as características de hardware.

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 frequentemente 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 ($10^{15}$flops/s).

O fabricante normalmente se refere ao seu desempenho maximo quando descreve um sistema. Este pico de desempenho chega através da contagem do numero de adições e multiplicações de ponto flutuante que pode ser completado em um período de tempo, geralmente o tempo de ciclo da maquina.

Por exemplo, um Intel Pentium III com um tempo de ciclo de 750 MHz tem dois pontos de unidades flutuante, durante cada ciclo, os resultados tanto do componente ou multiplicador pode ser concluído, e, assim, o pico de desenvolvimento é:

$$R_{peak} = \frac{1 \text{ operação}}{1 \text{ ciclo}} . 750 \text{ MHz} = 750 \text{ Mflop/s}$$

**Tabela 4. Mostra o desempenho máximo de alguns computadores utilizando o Linpack matriz de 100 por 100.**

| Maquina | Tempo ciclo [MHz] | Desempenho Pico [Mflop/s] | Desempenho LINPACK 100 por 100 [Mflop/s] |
|---|---|---|---|
| Intel Pentium III | 750 | 750 | 138 |
| Intel Pentium 4 | 2.530 | 5.060 | 1190 |
| Intel Itanium | 800 | 3.200 | 600 |
| AMD Athlon | 1.200 | 2.400 | 557 |

O desempenho de pico teórico é um limite informado pelo fabricante que garante que os programas não serão superiores a estes valores (informados na tabela acima). O Benchmark Linpack mostra que na prática pode haver uma diferença significativa entre o desempenho de pico teórico e o desempenho real.

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 $2 / 3n^3$, e o numero de dados referenciados, ambos loads e stores, é $2 / 3n^3$. 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 são 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-memoria-referência deve ser maior.

Em certo sentido é um problema simples fazer operações vetoriais de um vetor ou em maquina super-escalar. O gargalo esta na quantidade de dados em movimento limitando a taxa de execução. Podemos ver isso através da análise das taxas de transferências de dados e do desempenho máximo.

### 3.4. Reestruturações de Algoritmos

Hoje arquiteturas de computadores possuem vários estágios de memória hierárquica. Os algoritmos reestruturados para explorar essa organização hierárquica podem obter um alto desempenho. Para chegar a um desempenho máximo, é preciso

otimizar a utilização do nível mais baixo de memória (ou seja, manter os dados, o máximo possível, antes do próximo acesso à memória principal), obtendo reutilização tanto quanto possível.

### 3.5. Operações Matriz-Vetor

Estas operações têm a vantagem de reutilizar os dados e conseguir uma maior taxa de execução do que o vetor equivalente. De fato, o número de operações de ponto flutuante permanece o mesmo, apenas os dados de referência padrão são alterados. Esta mudança resulta em uma operação para memória, taxa de referência em vetor de computadores de grande efeito, 2 vetores de operações de ponto flutuante e 1 vetor de memória de referência.

Em Máquinas vetoriais para processamento, um dos objetivos destas implantações é a de manter a duração do vetor tanto tempo quanto possível, e na maioria dos algoritmos, os resultados são computados num vetor (linha ou coluna) de uma só vez. Além disso, em registro vetorial o desempenho de maquinas é aumentado pela reutilização dos resultados de registro de um vetor.

Infelizmente, esta abordagem de construção do software muitas vezes não se adéqua a computadores com uma hierarquia de memória (como memória global, memória cache ou local, e registradores vetoriais) e computadores de processamento paralelo. Para essas arquiteturas, muitas vezes, é preferível particionar a matriz ou matrizes em blocos para realizar a computação pela matriz-matriz sobre esses blocos de operações.

Organizar o cálculo dessa forma fornece a máxima reutilização de dados, enquanto o bloco é realizado na cache ou na memória local. Esta abordagem evita a deslocação excessiva dos dados da memória e atribui efeitos para a relação das operações de circulação de dados. Além disso, arquiteturas que fornecem processamento paralelo, o paralelismo pode ser explorado de duas formas: primeiro operações em blocos distintos podem ser realizadas em paralelo, e segundo no âmbito das operações em cada bloco, operações escalares ou vetoriais podem ser realizadas em paralelo.

### 3.6. Operações Matriz-Matriz

Existem vários algoritmos para bloquear tais problemas em matrizes, muitos dos primeiros algoritmos utilizavam uma pequena memória principal, com fita ou disco como armazenamento secundário, técnicas semelhantes foram posteriormente utilizadas para a exploração comum de algoritmos page-swapping em máquinas com memória virtual.

Essas técnicas são aplicáveis sempre que existe uma hierarquia de armazenamento de dados (em termos de velocidade de acesso). Além disso, blocos completos (e consequentemente, a multiplicação de matrizes completas) podem aparecer como um subproblema ao manusear grandes sistemas de equações. Recentemente, vários pesquisadores têm demonstrado a efetivação de algoritmos em computadores modernos com arquiteturas de processamento vetoriais ou capacidades de processamento paralelo, sobre os quais, potencialmente de alto desempenho podem ser facilmente degradados pela excessiva transferência de dados entre diferentes níveis de memória (registradores vetoriais, cache, memória local, memória principal, ou estado de discos sólidos.

## 4. Benchmark Linpack Paralelo

Nos últimos anos, o aparecimento de Memórias Distribuídas (MD) em computadores, com seu grande potencial para solução de grandes problemas numéricos levou a pesquisas na análise comparativa. Exemplos de computadores usando "memórias simples" IBM Scalable Power Parallel SP-2, Intel Paragon e o Cray T3E, redes e grupos de trabalho (NOWs e COWs). A principal característica de terem atingido um alto desempenho é que estes computadores incluem um conjunto de Unidades de Processamento (PUs), onde cada unidade é constituída por um processador, as memórias locais são organizadas de maneira hierárquica, e outros dispositivos de apoio. Estes PUs são interligados por uma rede ponto a ponto (direta). Sem modificação da máquina básica arquitetural, este sistema de memória distribuída é capaz de proporcionar aumento de desempenho com o número de PUs, sua capacidade de memória e a largura da banda da rede são aumentadas. Computadores com MD são bons, mas são feitas algumas melhorias.

Estas melhorias consistem na construção a partir de um pequeno número de nós, onde cada nó é um computador com uma pequena MD(memória distribuída) virtual. Esses nodos são interligados por uma simples interconexão de rede chamada Crossbar (rede não bloqueante). A programação nestas máquinas bem como a sua produção é facilitada pela relativa simplicidade nas interligações de rede. Além disso, o aumento das capacidades computacionais das PUs parece ser uma tarefa mais fácil do que aumentar o desempenho da rede.

A fim de explorar plenamente o crescente poder computacional dos computadores MD, a aplicação software deve ser escaláveis, ou seja, capazes de tirar "maior proveito" da máquina para resolver maiores problemas com a mesma eficiência. O Linpack referência endereços escaláveis por uma introdução de uma nova categoria, essa nova categoria é a que se refere ao Linpack altamente paralelo (HPL) benchmark NxN. Exige uma solução de sistemas de equações lineares por algum método, o problema é o tamanho permitido para variar, e a melhor taxa de execução de ponto flutuante deve ser satisfatória. Em computação a taxa de execução, a um número de operações deve ser $2n^3 / 3 + 2n^2$ independente do método utilizado. As seguintes quantidades de referência são relatadas na lista TOP500:

$R_{max}$ = Desempenho para o maior problema executado em uma máquina(em Gflop/s);

$N_{max}$= Tamanho do maior problema executado em uma máquina;

$N_{1/2}$ = Tamanho quando metade de $R_{max}$ é executado;

$R_{peak}$= O pico de desempenho teórico para a máquina (em Gflop/s).

Para o desempenho do benchmark HPL NxN contribui na eficiência do código executado em uma única CPU, bem como o algoritmo paralelo, o que torna todos os CPUs cooperar. O antigo pode ser utilizado com o uso de técnicas de otimização mencionadas anteriormente.

## 6. O Código HPL

HPL é um pacote de software que resolve um denso sistema de equações lineares em computadores com memória distribuída. O pacote usa 64-bits aritmética de ponto flutuante e rotinas portáteis para operações de álgebra linear e passagem de mensagens. Além disso, ele dá a possibilidade de escolher um de vários algoritmos de fatorização e fornece o tempo e uma estimativa de precisão da solução.

Por consequência, não pode ser considerado como uma aplicação portátil da HPL de referência NxN. Ele exige implementações do MPI e BLAS ou a Biblioteca de processamento de imagem (VSIPL). Principais etapas executadas pelo pacote HPL para obter a HPL NxN :

- Gerar e partição de dados entre matriz MPI computação nodos.
- Todos os nós começam ao mesmo tempo. MPI_Barrier (...);
- Iniciar temporizador. HPL_ptimer (...);
- HPL_pdgesv (...); Resolva sistema de equações.
- Parar relógio temporizador. HPL_ptimer (...);
- Obter o máximo do tempo relógio. MPI_Reduce (...);
- Recolher estatísticas sobre a taxa de desempenho (base na máxima do relógio) e precisão da solução.

### 6.1. O Algoritmo

A Tabela 5 mostra em 2-D um bloco de distribuição dos dados utilizados pelo HPL. Os dados são distribuídos em uma grade bidimensional (de dimensões P por Q) dos processos de acordo com o regime de bloco-cíclico para assegurar um bom equilíbrio de carga, bem como a escalabilidade do algoritmo. O coeficiente n por n + 1 da matriz é logicamente particionado em blocos (cada um de dimensão NB por NB), que são tratadas ciclicamente para o P pelo processo Q da grade. Isto é feito em ambas as dimensões da matriz.

**Tabela 5. Distribuição utilizada pela HPL.**

| | | | |
|---|---|---|---|
| P0 | P1 | P0 | P1 |
| P2 | P3 | P2 | P3 |
| P0 | P1 | P0 | P1 |
| P2 | P3 | P2 | P3 |

O número de processadores é de 4 (denominado P0, P1, P2 e P3), elas estão organizadas em 2 por 2 na grade (P = Q = 2). O número de sub-blocos é de 4 em ambas as dimensões ($N / N_B = 4$). Numa dada interação do ciclo principal, e por causa da propriedade cartesiana da distribuição deste esquema, cada painel de fatorização ocorre em uma coluna de processos. Para esta operação, ao usuário é forçado uma escolha de três recursos variantes matriz-matriz com base em multiplicação.

Depois que o painel fatorização foi realizado, o painel de colunas fatorado é difundido para a outra coluna de processo. São muitos os possíveis algoritmos, a difusão de software atualmente oferece as seguintes variantes:

- Aumentar o anel,
- Modificar o aumento do anel,
- Aumento de dois anéis,
- Modificar o aumento de dois anéis,
- Redutores,

- Modificar banda de redução,

As modificações variam para aliviar o próximo processador (o que teria uma participação na fatorização do painel após o atual) a partir da carga de envio de mensagens (outro que tem a receber bem como enviar matriz atualizadas dos dados). O anel variante propaga a atualização de dados em uma única direção, enquanto que as duas variantes de anel propagam dados em duas direções concomitantemente. A banda de redução de variantes divide uma mensagem a ser enviada para um número de peças e envia-as através de uma única linha da grade de processadores, para que mais mensagens sejam trocadas, mas o volume total da comunicação é independente do número de processadores. Isto se torna particularmente importante quando a computação nodos é relativamente muito mais rápido do que a interconexão. Uma vez que o atual painel foi transmitido (ou durante a operação de difundir) a sub-matriz tem de ser atualizada.

## 5. A Lista TOP500

Estatísticas sobre computadores de alto desempenho são de grande interesse para os fabricantes, e os potenciais usuários. Eles querem saber não só o número de sistemas instalados, mas também a localização dos diversos supercomputadores de alto desempenho e as aplicações para as quais um sistema de computador está sendo usado. Essas estatísticas podem facilitar a colaboração de sistemas, o intercâmbio de dados e software, e fornecer uma melhor compreensão do computador de alto desempenho no mercado.

Estatísticas baseadas apenas no nome do fabricante não são mais úteis, no entanto novas estatísticas são exigidas para refletir a diversificação de supercomputadores. Para fornecer esta nova base, em 1993 foi criada a lista do TOP500 onde é mantida uma lista dos 500 mais poderosos sistemas de computadores.

A lista é atualizada duas vezes por ano, com a ajuda de especialistas em computadores de alto desempenho, cientistas computacionais, fabricantes, bem como a comunidade da Internet em geral. O ranking TOP500 consiste em uma listagem ordenada dos quinhentos supercomputadores de maior desempenho do mundo, sendo tal medida feita atualmente com o LINPACK Benchmark.

O primeiro ranking foi criado em Junho de 1993, e desde então anualmente são publicadas duas listagens, uma em Junho e outra em Novembro. A idéia inicial deste ranking foi a de prover uma nova base estatística para os supercomputadores atuais, mas atualmente o intuito principal dessa lista é a de fornecer informação sobre a evolução dos supercomputadores.

Para se candidatar ao ranking as empresas e usuários fornecem à equipe do top500 os resultados de desempenho do seu equipamento no benchmark LINPACK. Porém, para evitar possíveis fraudes e corrigir possíveis erros, a equipe do TOP500 tenta testar independentemente os resultados. A listagem é fornecida gratuitamente no site do TPO500, sendo que grandes empresas como a DELL, IBM, HP, entre outras grandes fabricantes de computadores patrocinam esse projeto.

### 6.2. Resultados de desempenho

A fim de adquirir o melhor desempenho de um determinado sistema, o maior problema era o tamanho de memória a ser usado. A quantidade de memória utilizada pelo HPL é basicamente o tamanho do coeficiente da matriz HPL, o mesmo usa $N_B$

tamanho do bloco para a distribuição dos dados, bem como para a granularidade computacional. Distribuição dos dados a partir de um ponto de vista, o menor $N_B$, melhor o equilíbrio de carga, assim deve-se evitar valores muito grandes da $N_B$. A partir de um local computacional baseando-se em valores pequeno de $N_B$ podem limitar o desempenho computacional por um grande fator porque quase não há dados reutilizados, isso irá ocorrer mais rápido no nível de hierarquia de memória.

O número de mensagens também irá aumentar a eficiência da matriz multiplicar rotinas, são frequentemente bloqueados internamente. Pequeno múltiplo que bloqueia este fator é provável que sejam tamanhos de bons blocos para HPL. As tabelas 6 e 7 descrevem um cluster baseado no processador Pentium III.

**Tabela 6. Descrição do cluster usando processador Pentium.**

| | |
|---|---|
| CPU | Intel Pentium III 550 Mhz |
| Memória | 512 MB |
| Interconexão | Myrinet |
| SO | RedHat Linux 6.1 (kernel 2.2.15) |
| Compilador C | gcc ver. 2.91.66 (egcs-1.1.2 release) |
| C ags | fomit-frame-pointer -O3 -funroll-loops |
| MPI | MPI GM ver. 1.2.3 |
| BLAS | ATLAS ver. 3.0 beta |

**Tabela 7: Desempenho (em Gflops/s) do HPL em um cluster com 16 nodos Pentium.**

| Dimensões do Processo | Dimensão Matriz | | | | | |
|---|---|---|---|---|---|---|
| | 2000 | 5000 | 8000 | 10000 | 15000 | 20000 |
| 2 ate 4 | 1.76 | 2.32 | 2.51 | 2.58 | 2.72 | 2.73 |
| 4 ate 4 | 2.27 | 3.94 | 4.46 | 4.68 | 5.00 | 5.16 |

## 6.3 Testes Realizados

Este capítulo tem por objetivo apresentar os resultados de uma experiência prática de aplicação do benchmark Linpack utilizando uma matriz de 2000x2000. O teste consistiu em uma comparação de desempenho feita entre vários computadores de linhas e famílias totalmente distintas, fabricados por indústrias concorrentes e amplamente utilizados no mercado atual de computadores pessoais.

Este teste não tinha como objetivo medir o desempenho entre as máquinas, pelo motivo de as mesmas pertencerem a linhas e famílias distintas. Os resultados mostrados a seguir são fruto da execução de computadores usados no dia a dia, não sofrendo assim nenhuma alteração (software e hardware).

**Tabela 8: Desempenho (em Mflops/s) dos testes realizados no Linpack em uma matriz 2000x2000 .**

| Processador | Frequência | Tempo Execução (segundos) | Desempenho (Mflops/s) |
|---|---|---|---|
| Intel Core 2 Quad | 2.40 GHz | 1.04 | 448.074 |
| Intel Core 2 Duo | 2.20 GHz | 1.78 | 375.445 |
| Intel Core 2 Duo | 1.66 GHz | 3.56 | 187.772 |

## 7. Conclusão

No decorrer deste trabalho nota-se que nos últimos anos, o Benchmark Linpack tem evoluído a partir de uma simples listagem de um problema para uma matriz expandida que descreve o desempenho em três níveis de especificações em centenas de computadores. O referencial é hoje utilizado por cientistas no mundo inteiro para avaliar o desempenho de computadores, especialmente para máquinas de arquitetura inovadoras e avançadas.

No entanto, uma nota de cautela é necessária. Os Benchmarkings não devem ser utilizados indiscriminadamente para julgar o desempenho global de um sistema de computador. O desempenho é um tema complexo, dependente de uma variedade de diferentes quantidades incluindo o algoritmo, tamanho do problema, bem como a implementação. O Benchmark Linpack prevê três parâmetros distintos que podem ser utilizados para avaliar o desempenho em um computador com um sistema denso de equações lineares: o primeiro para a matriz de 100 por 100, o segundo de uma matriz de 1000 por 1000. A terceira referência, em particular, está dependente do algoritmo escolhido pelo fabricante e a quantidade de memória disponível no computador que está a ser aferido.

## 8. Referências


DE ROSE, Cesar A. F.; NAVAUX, Philippe O. A.. **Arquiteturas paralelas.** Porto Alegre: Sagra Luzzatto, 2003.

**Frequently Asked Questions**, LINPACK Benchmark. Disponível em: nettlib.bell-labs.com/netlib/benchmark/faq-linpack.html. Acessado em maio de 2009.

**Top500**, Supercomputing Sites. Disponível na internet via: www. URL:http://www.top500.org. Acessado em  de maio de 2009.

Jack J. Dongarra, Piotr Luszczek, and Antoine Petitet.. **The LINPACK Benchmark: Past, Present, and Future**. Disponível em: http://www.netlib.org/utk/people/JackDongarra/PAPERS/hplpaper.pdf.

**Versão Java do Benchmark Linpack.** Disponível na internet via: http://www.netlib.org/benchmark/linpackjava/. Acessado em maio de 2009.

Wikipedia. **Benchmark (Computação).** Disponível na internet via www. URL: http://pt.wikipedia.org/Benchmark_(computação). Acessado em maio de 2009.