**Conceitos Básicos de Memória**

Para entender o conceito de Big e Little endian, você precisa entender a memória. Felizmente, só precisamos de uma abstração de nível muito alto para a memória. Não precisamos saber todos os pequenos detalhes de como funciona a memória.

Tudo que é necessário saber sobre a memória é que é uma grande matriz. Mas uma grande matriz contendo o quê? A matriz contém *bytes*. Na organização do computador, as pessoas não usam o termo "índice" para se referir aos locais da matriz. Em vez disso, usamos o termo "endereço". "Endereço" e "índice" significam o mesmo, basta pensar em "endereço" como "índice".

Cada endereço armazena um elemento da memória "matriz". Cada elemento é tipicamente um byte.

### Guardar palavras na memória

Definimos uma palavra para significar 32 bits. Este é o mesmo que 4 bytes. Inteiros, números de ponto flutuante de precisão simples e instruções MIPS são todos 32 bits longo. Como podemos armazenar esses valores na memória? Afinal, cada endereço de memória pode armazenar um único byte, não 4 bytes.

A resposta é simples. Dividimos a quantidade de 32 bits em 4 bytes. Por exemplo, suponha que temos uma quantidade de 32 bits, escrita como 90AB12CD16, que é hexadecimal. Como cada dígito hexadecimal é de 4 bits, precisamos de 8 dígitos hexadecimais para representar o valor de 32 bits.

Assim, os 4 bytes são: 90, AB, 12, CD onde cada byte requer 2 dígitos hexadecimais.

Acontece que existem duas maneiras de armazenar isso na memória.

#### **Big Endian**

No big endian, você armazena o bit mais significativo no menor endereço. Veja como seria:

|  |  |
| --- | --- |
| Endereço | Valor |
| 1000 | 90 |
| 1001 | AB |
| 1002 | 12 |
| 1003 | CD |

#### **Little endian**

Em little endian, você armazena o bit *menos* significativo no menor endereço. Veja como seria:

|  |  |
| --- | --- |
| Endereço | Valor |
| 1000 | CD |
| 1001 | 12 |
| 1002 | AB |
| 1003 | 90 |

Observe que isso é na ordem inversa comparado ao big endian. Para lembrar qual é qual, lembre se o bit menos significativo é armazenado primeiro (assim, little endian) ou o bit mais significativo é armazenado primeiro (assim, big endian).

**Direct Memory Access (DMA)**

O termo DMA (sigla da expressão inglesa Direct Memory Access) designa uma forma de acesso directo à memória utilizada por alguns aplicativos que permite que a informação seja transferida entre a memória e o disco rígido, sem passar pelo [processador](http://knoow.net/ciencinformtelec/informatica/processador/), ganhando em rapidez.

Enquanto a maioria dos dados, que são de entrada ou de saída do computador, são processados pelo [processador](http://knoow.net/ciencinformtelec/informatica/processador/), alguns dados não necessitam desse processamento, ou podem ser processados por um outro dispositivo. Nestas situações, o DMA pode economizar tempo de processamento e é uma maneira mais eficiente para mover os dados da memória do computador para outros dispositivos.

Os canais DMA são usados para comunicar dados entre o dispositivo periférico e a memória do sistema. Todos os quatro recursos do sistema dependem de certas linhas num barramento. Algumas linhas no barramento são usadas para IRQs, algumas para endereços (os endereços I / O e o endereço de memória) e algumas para canais DMA.

Um canal DMA permite que um dispositivo possa transferir dados sem expôr o [CPU](http://knoow.net/ciencinformtelec/informatica/processador/) a uma sobrecarga de trabalho. Sem os canais de DMA, o [CPU](http://knoow.net/ciencinformtelec/informatica/processador/) copia cada pedaço de dados através de um barramento de periféricos a partir do dispositivo I / O. Ao usar um barramento periférico, ele ocupa o [CPU](http://knoow.net/ciencinformtelec/informatica/processador/) durante o processo de leitura / escrita e não permite que outro trabalho possa ser executado até que a operação seja concluída.

Com o DMA, o [CPU](http://knoow.net/ciencinformtelec/informatica/processador/) pode processar outras tarefas enquanto a transferência de dados está a ser executada. A transferência de dados é primeiro iniciada pelo [CPU](http://knoow.net/ciencinformtelec/informatica/processador/). Durante a transferência de dados entre o canal de DMA e o dispositivo de I / O, o [CPU](http://knoow.net/ciencinformtelec/informatica/processador/) executa outras tarefas. Quando a transferência de dados está completa, o [processador](http://knoow.net/ciencinformtelec/informatica/processador/) recebe um pedido de interrupção do controlador de DMA.

Um dispositivo utiliza tecnologia DMA ao usar apenas um único canal. Para evitar um conflito, por vezes, a [BIOS](http://knoow.net/ciencinformtelec/informatica/bios/) (Basic Input/Output System ou Sistema Básico de Entrada / Saída) deve atribuir um canal diferente para um dispositivo. Um conflito pode acontecer quando mais do que um dispositivo tenta utilizar o mesmo canal.

Os canais DMA são mais lentos do que os métodos de transferência de dados mais recentes, e, portanto, não são tão comuns. Uma interface mais recente é o Ultra DMA, que tem uma taxa de transferência de dados de até 33 MB por segundo. Cada transferência DMA usa aproximadamente 2 MB de dados por segundo.

# **O QUE É CLUSTER**

Um Clusteré formado por um conjunto de computadores interligados através de uma rede, as máquinas membros deste cluster são denominadas nó ou node. É importante utilizar uma infraestrutura de rede que facilite a inclusão, alteração e exclusão de máquinas.   
  
Na maioria das vezes o cluster é formado por computadores convencionais e se apresenta de forma transparente ao usuário, como sendo um único computador de grande porte. É válido frisar que é possível a utilização de máquinas mais robustas para construção de clusters.   
  
Não é necessário que as máquinas sejam idênticas, mas sim o Sistema Operacional, para que os softwares que gerenciam as trocas de mensagens e sincronismo dos dados funcionem de forma correta.   
  
De acordo com Zem (2005), existem hoje alguns tipos de cluster, mas alguns se destacam pela aplicação e custo benefício:

* **Cluster de Alto Desempenho**: denominado, também, de Alta Performance (High Performance Computing - HPC), sua característica é o grande volume de processamento de dados em computadores convencionais, que garante baixo custo na construção, e com processamento na ordem de gigaflops. Os servidores deste cluster trabalham com a tecnologia de paralelismo, dividindo o processamento com as outras máquinas, buscando a otimização e desempenho de um supercomputador.
* **Cluster de Alta Disponibilidade** (High Availability - HA): são caracterizados por se manterem em pleno funcionamento por um longo período de tempo, utilizando redundância para manter um serviço ativo e se proteger de falhas, geralmente são computadores convencionais que disponibilizam o mesmo recurso em todas as máquinas da rede, configuradas com prioridades diferentes, onde existe um servidor ativo e os outros ociosos.
* **Cluster de Balanceamento de Carga** (Horizontal Scaling - HS): são caracterizados por dividirem, de forma equilibrada, as tarefas entre os membros do cluster, onde cada nó atenda a uma requisição e não, necessariamente, que divida uma tarefa com outras máquinas.

É importante salientar que é possível a combinação de mais de uma metodologia de construção de clusters, onde uma implementação de Alta Disponibilidade para garantir acesso aos serviços de vendas online, possa ser incrementada com a utilização de um cluster de Balanceamento de Carga para atender o aumento nos acessos ao serviço.   
  
As características marcantes dos clusters, são a facilidade de gerenciamento dos nós, onde podemos adicionar, dar manutenção e remover um nó do cluster sem que afete seu funcionamento, recuperação de falhas de forma otimizada.   
  
Podemos obter resultados tão satisfatórios no uso de um cluster, quanto em servidores sofisticados com um custo muito menor. A implementação pode ser utilizada para aplicações sofisticadas e, também, para aplicações domésticas.

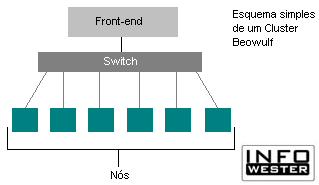
**Cluster Beowulf**

O **Beowulf** não é, necessariamente, um middleware, como muitas pensam. Na verdade, este nome faz referência a um padrão de clustering disponibilizado pela NASA (National Aeronautics and Space) em 1994 e amplamente adotado desde então.

Originalmente, Beowulf é o nome de um poema extenso e bastante antigo, cujo manuscrito foi encontrado no século XI. A obra descreve os atos de um herói de mesmo nome que se destaca por sua força descomunal e que, portanto, enfrenta um perigoso monstro para salvar um reino. A história serviu de inspiração para que os pesquisadores Thomas Sterling e Donald Becker, da NASA, batizassem o projeto de cluster no qual trabalhavam de Beowulf.

Um cluster Beowulf se define, basicamente, pela ênfase nas seguintes características:

* Entre os nós, deve haver pelo menos um que atue como mestre para exercer o controle dos demais. As máquinas mestres são chamadas de front-end*;* as demais, de back-end*.* Há a possibilidade de existir mais de um nó no front-end para que cada um realize tarefas específicas, como monitoramento, por exemplo;



A comunicação entre os nós pode ser feita por redes do tipo Ethernet, mais comuns e mais baratas, como você já sabe;

* Não é necessário o uso de hardware exigente, nem específico. A ideia é a de se aproveitar componentes que possam ser encontrados facilmente. Até mesmo PCs considerados obsoletos podem ser utilizá-los;
* O sistema operacional deve ser de código aberto, razão pela qual o Linux e outras variações do Unix são bastante utilizados em cluster Beowulf.
* Os nós devem se dedicar exclusivamente ao cluster;
* Deve-se fazer uso de uma biblioteca de comunicação apropriada, como a PVM (Parallel Virtual Machine) ou a MPI(Message Passing Interface). Ambas são direcionadas à troca de mensagens entre os nós, mas o MPI pode ser considerado mais avançado que o PVM, uma vez que consegue trabalhar com comunicação para todos os computadores ou para apenas um determinado grupo.

Perceba que, com estas características, pode-se construir um cluster "poderoso" e, ao mesmo tempo, poupar gastos com equipamentos, licenças de software e manutenção. O cluster montado por Thomas Sterling e Donald Becker para a NASA, por exemplo, era composto por 16 PCs com processador Intel 486 DX4 e [sistema operacional Linux](https://www.infowester.com/historia_linux.php) conectados por uma rede Ethernet de 10 Mb/s. Como se vê, esta foi uma solução consideravelmente mais barata e, possivelmente, menos complexa que um supercomputador.

**Message Passing Interface (MPI)**

Quando se trata em computação paralela, múltiplos computadores (ou múltiplo processadores dentro de um mesmo computador) são chamados de nós, e cada um desse nó desempenha uma função no sistema computacional completo. Um desafio dessa organização de sistemas funcionando em paralelo é sincronizar as ações de cada nó, trocar dados entre eles e controlar o todo. A MPI (Message Passing Interface) define um padrão de funções que coordenam tais tarefas.

Usando MPI, o programador é responsável por distribuir as tarefas entre os nós. Então, os resultados são enviados à uma máquina receptora que irá agrupar e fornecer os dados coletados.

Os conceitos básicos, de forma resumida, da MPI são:

* Processo: cada parte da aplicação que foi dividida pelo programador. Pode ser executado por diversas máquinas ou em apenas uma.
* Rank: identificação de cada processo, onde varia de 0 a N (número total de processos).
* Mensagem: conteúdo de uma comunicação.
* Comunicador: objeto que representa o domínio de um comunicação.
* Grupo: Conjunto de processos que se comunicam entre si. Está sempre associado a um comunicador.

Alguns exemplos de funções utilizados no MPI:

* MPI\_Init MPI\_Init – inicialização para o ambiente MPI inicialização para o ambiente MPI
* MPI\_Comm\_size MPI\_Comm\_size – retorna o número de processadores retorna o número de processadores
* MPI\_Comm\_rank MPI\_Comm\_rank – retorna o " retorna o "rank" (índice, identificador) do processador " (índice, identificador) do processador
* MPI\_Send MPI\_Send – envia uma mensagem envia uma mensagem
* MPI\_Recv MPI\_Recv – recebe uma mensagem recebe uma mensagem
* MPI\_Finalize MPI\_Finalize – sai do ambiente MPI

**Referência**

* Material de aula (relatórios e apostila);
* <https://www.google.com.br/imgres?imgurl=https%3A%2F%2Fi.stack.imgur.com%2FWgv7D.png&imgrefurl=https%3A%2F%2Fsuperuser.com%2Fquestions%2F707161%2Fcaculating-least-significant-bit-in-little-endian-and-big-endian&docid=EKKmwQW9wqRweM&tbnid=ESmvwAQF_WtGhM%3A&vet=10ahUKEwjCveW7sqzTAhVKI5AKHabADdsQMwhwKEMwQw..i&w=157&h=201&bih=613&biw=1366&q=memory%20endianness%20o%20que%20%C3%A9&ved=0ahUKEwjCveW7sqzTAhVKI5AKHabADdsQMwhwKEMwQw&iact=mrc&uact=8> (Imagem do big e little endian)
* <http://www.intel.com.br/content/www/br/pt/support/network-and-i-o/ethernet-products/000007456.html>
* <http://www.ebah.com.br/content/ABAAAfgp4AF/estrategias-io-dma>
* <https://www.google.com.br/url?sa=t&rct=j&q=&esrc=s&source=web&cd=9&cad=rja&uact=8&ved=0ahUKEwjO0LG_s6zTAhUGk5AKHbMEDkYQFghaMAg&url=ftp%3A%2F%2Fftpaluno.umc.br%2FAluno%2FLuciano_Goncalves%2FProcessamento%2520Distribu%25EDdo%2520e%2520Paraleliza%25E7%25E3o%2FTrabalhos_Cluster%2FCluster%2520Beowulf%2520-%2520Apresenta%25E7%25E3o.pps&usg=AFQjCNEql8zH6jJ4NCbQrLPAuBCWU5Xorg>
* <http://si-masters.blogspot.com.br/2011/08/cluster-2.html>
* <http://searchenterprisedesktop.techtarget.com/definition/message-passing-interface-MPI>
* http://www.ufjf.br/ciro\_barbosa/files/2012/03/MPI.pdf