INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SÃO PAULO
Campus Catanduva

Arquiteturas Paralelas

Catanduva

19 de junho de 2024

ARQUITETURAS PARALELAS

Documento apresentado como requisito parcial para obtenção de aprovação na disciplina de Arquitetura de Computadores, do Curso de Análise e Desenvolvimento de Sistemas, no Instituto Federal de São Paulo, Campus Catanduva.

Prof.: Bruno L. P. de Azevedo

Grupo:
Eduardo Lemes
Guilherme Batista
Lucas Marcelino
Polayne Bastos

Catanduva

19 de junho de 2024

Resumo

A ideia central do documento é explicar e contextualizar sobre as Arquiteturas Paralelas, as quais são essenciais para a computação de grande desempenho, permitindo que múltiplas tarefas sejam racionadas e executadas ao mesmo tempo, aumentando a eficiência e a velocidade de processamento. O artigo busca apresentar uma síntese sobre arquiteturas paralelas, proporcionando também um detalhamento de alguns pontos específicos dessa forma de computação. Inicialmente, destaca-se os níveis de paralelismo, onde é evidenciado as diferentes formas de racionalização e execução de tarefas em um determinado processo. Posteriormente, a taxonomia de Flyn (categorização de arquitetura de computadores), arquiteturas superescalares (execução de instruções independentes) e superpipeline (execução de instruções simultâneas), VLIW(Very long instruction word, ou seja, execução de instruções longas e compostas) e arquiteturas vetoriais (operam sobre vários elementos). Além disso, são evidenciados os aspectos relacionados a multiprocessadores e multicomputadores, referente a utilização de mais de uma unidade de processamento e múltiplos computadores interligados em uma rede. Ao fim do documento, deve-se estar claro o que é Arquiteturas Paralelas e sua utilização, bem como muitas das funções e ramificações desta área.

Sumário

Lista de ilustrações

Lista de tabelas

1 Introdução

A partir do início da era da computação e dos grandes obstáculos que surgiram na área, principalmente o fato da velocidade da memória ram ser inferior a velocidade da memória do processador (gargalo de Von Neumann), surge a necessidade de formulação de algo capaz de reduzir essa discrepância e aumentar a capacidade de processamento e execução de diferentes tarefas computacionais. Assim, em estudos posteriores, Von Neumann, matemático nascido na Hungria, passa a desenvolver pesquisas introdutórias sobre arquiteturas paralelas, uma nova tecnologia interligada (Sistemas24horas,).

A computação em arquiteturas paralelas envolve a realização de múltiplos cálculos simultaneamente, baseada na premissa de que problemas complexos podem ser divididos em partes menores e resolvidos em paralelo. Este conceito tem sido aplicado há anos, especialmente em computação de alto desempenho. No entanto, o interesse recente na computação paralela tem sido impulsionado pelas limitações físicas que dificultam o aumento da frequência de processamento.

Com o crescente foco na eficiência energética dos sistemas computacionais, o ideal paralelo tornou-se central nas arquiteturas de processador, permitindo uma distribuição mais eficiente de tarefas e, consequentemente, um melhor aproveitamento dos recursos disponíveis (Wikipedia-contributors, 2024).

Dentre das inúmeras áreas e atuação das arquiteturas paralelas, pode-se destacar o desenvolvimento de novos modelos de computação, arquiteturas de sistemas computacionais (multiprocessadores) e a viabilização de outras aplicações, como previsão de tempo e de clima (Midorikawa, 2010).

2 Sistemas computacionais paralelos

2.1 Níveis de Paralelismo

Os níveis de paralelismo fazem referência a capacidade de executar, de diferentes maneiras, múltiplas tarefas computacionais, proporcionando melhor eficiência e desempenho. Existem diversos tipos de paralelismo capazes de serem explorados, os quais operam a partir de determinada técnica e escala específica.

2.1.1 Paralelismo em nível de bit

O paralelismo em nível de bit é a projeção dos processadores com unidades de processamento em paralelo, capazes de realizar tarefas simultaneamente. No processamento de dados em paralelo, os dados passam a ser divididos em palavras de tamanho fixo e são executados em paralelo. Exemplo: os dados são racionalizados em palavras de 32 bits, em um processador de 32 bits, e cada uma é processada individualmente.

Como vantagem da utilização dessa técnica, pode-se apontar a execução de operações e ciclos de clock reduzidos, o desligamento de partes não utilizadas para reduzir o consumo de energia e a manipulação de um grande volume de dados com eficiência (imagens e vídeos).

Entretanto, como limitação desse processo, é possível apontar a necessidade de um hardware específico com unidades de processamento paralelo e decodificadores de instruções, a demanda pela independência entre as instruções e os dados, e o requerimento de uma otimização do sistema para que esse processamento paralelo seja executado com eficácia (O-QUE-é-BIT-LEVEL-PARALLELISM, sem data).

2.1.2 Paralelismo em nível de instrução

O paralelismo espacial também é conhecido por paralelismo em nível de instrução (ILP - Instruction Level Parallelism).Essa técnica envolve a execução de várias instruções em paralelo, de modo que diferentes partes de várias instruções sejam executadas simultaneamente.No entanto,torna se eficaz o paralelismo ao ser complexa e requer hardware sofisticado, lidando com algoritmos inteligentes de agendamento de instruções.

Pipeline: O pipeline é uma técnica fundamental para melhorar o desempenho dos processadores, seu objetivo é organizar a execução das instruções de um programa em uma série de etapas sequenciais, onde cada etapa executa uma parte da instrução. Dentre suas funções as etapas incluem a busca de instrução, decodificação, execução, acesso à memória e escrita de resultados, portanto mediante sua eficácia o pipeline também pode introduzir latência (Tecnoblog Responde, sem data).

Superscalaridade: Superescalares incluem simultaneamente as características do pipeline , porém, podem ter várias instruções executando em um mesmo estágio de pipeline. Ou seja, em

suas funções permite que um processador execute múltiplas instruções em cada ciclo de clock.

Execução fora de ordem: Se trata de um método em que o processador define a ordem que irá executar a instrução de um programa, proporcionando a diminuição na ociosidade dos recursos do chip, assim, concluindo a tarefa em menos tempo. A técnica consiste ao executar instruções na ordem em que são encontradas no código fonte (Portal Insights,).

2.1.3 Paralelismo em Nível de Dados (Data-Level Parallelism - DLP)

O paralelismo em nível de dados consiste em uma técnica de racionalização de dados em grande volume para partes menores, as quais são operadas de forma paralela. Após executados, todos os dados são unidos novamente em um único conjunto.

Essa divisão de dados, proporciona mais velocidade de processamento e consequentemente a obtenção de um certo resultado em tempo reduzido.

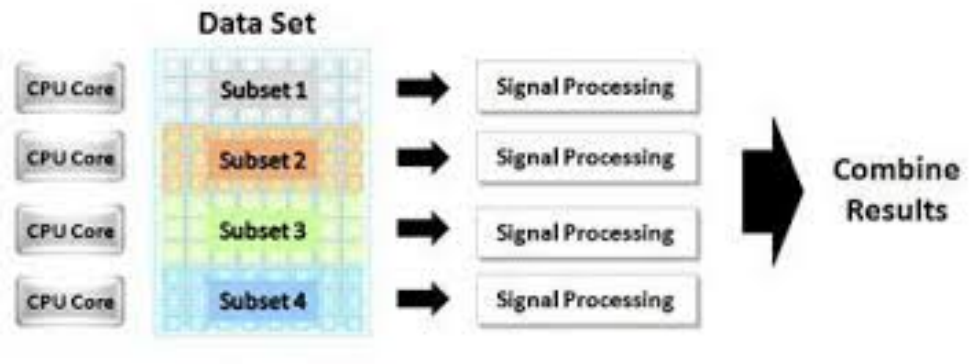

Figura 1 – Paralelismo em Nível de Dados

(SOBRAL,).

2.1.4 Paralelismo em Nível de Tarefas (Task-Level Parallelism)

O paralelismo em nível de tarefas possibilita a divisão de aplicações em tarefas exclusivas que são independentes entre elas, podendo ser executadas separadamente.

Um exemplo desse tipo de paralelismo seria a presença de dois loops em um programa. O primeiro loop realiza o processamento de sinais e o segundo executa atualizações da interface do usuário. Nesse sentido, são utilizados dois núcleos da cpu, já que é possível a execução de dois loops em threads (sequência de instruções) separadas.

No momento em que um código assíncrono (sem ordem de execução) é escrito, determinadas tarefas podem ser executadas sem provocar alguma interferência no fluxo de execução de outras tarefas. Logo, é possível iniciar a execução de várias tarefas, mesmo que outras ainda não estejam completas (SOBRAL,).

2.1.5 Paralelismo em Nível de Processos (Process-Level Parallelism)

O paralelismo em nível de processos é quando um sistema de computador executa várias tarefas ao mesmo tempo, isso já não é uma novidade, no entanto se torna alcançado por dividir

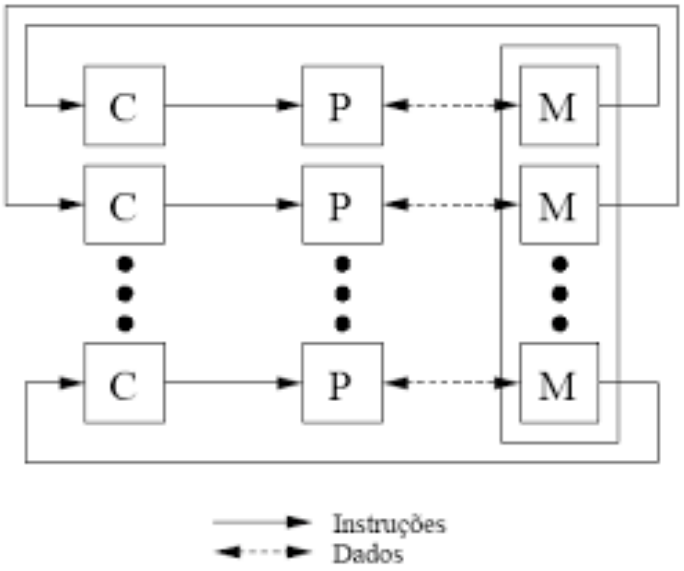

Figura 2 – Paralelismo em Nível de Processos

o tempo de processamento da CPU entre diferentes processos ou programas. Esses processos podem ser executados de maneira preemptiva, de forma que o sistema operacional decide quando parar um processo para iniciar outro. Mas também de maneira cooperativa, onde os processos decidem quando ceder o controle. Com está divisão de tarefas e processos , se torna mais eficiente ao realizadas (SOBRAL,).

2.1.6 Paralelismos em Nível de Cluster (Cluster-Level Parallelism)

O paralelismo em nível de cluster representa um sistema de processamento distribuído, em que diversos computadores executam tarefas em conjunto, assim como um recurso computacional unificado.

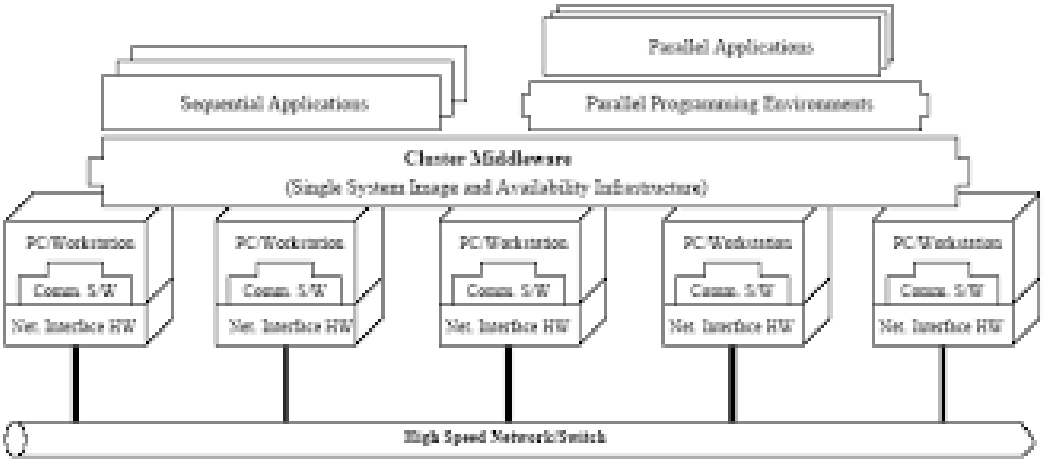

Figura 3 – Paralelismo Cluster

Nesse sistema, cada nó representa uma unidade computacional dotada de memória, dispositivos de entrada e saída, e sistema operacional próprio. Esses nós, operam de maneira integrada, visando solucionar problemas mais difíceis e desenvolver aplicações mais complexas (IBGE,).

2.2 Taxonomia de Flynn

A primeira descrição formal da Arquitetura Paralela foi a taxonomia de Flynn. Essa taxonomia foi desenvolvida em 1966 e ligeiramente expandida em 1972.

É um modelo de classificação de arquitetura de computadores baseado no fluxo de instruções e dados. Propõe uma abordagem para esclarecer os tipos de paralelismo suportados no hardware por um sistema de processamento ou disponíveis em uma aplicação. Sua classificação é baseada na visão da máquina ou do aplicativo pelo programador de linguagem de máquina (machine code). É dividido em quatro classes principais (DOCUMENTO,).

- SISD (Single Instruction, Single Data): Configuração mais simples e tradicional. Um único processador executa uma única instrução em um único conjunto de dados.
- SIMD (Single Instruction, Multiple Data): Frequentemente usada em computação de vetor, em que uma instrução é aplicada a vários elementos de um vetor ou matriz. Um únido processador executa múltiplas instruções em múltiplos conjuntos de dados.
- MISD (Multiple Instruction, Single Data): Configuração incomum, vários processadores executam instruções diferentes sobre os mesmos dados.
- MIMD (Multiple Instruction, Multiple Data): Tipo mais comum da arquitetura de computadores em sistemas multi-processador ou em clusters, em que múltiplos processsadores executam diferentes instruções em múltiplos conjuntos de dados.

2.3 Arquiteturas Superescalar e Superpipeline

Como visto, a taxonomia de Flynn veio como uma tese de funcionamento de uma nova tecnologia com base na quantidade dos fluxos e instruções simultâneas, onde por intermedio dela tivemos nossos primeiros computadores, Flynn apenas teorizou, mas muitos outros pesquisadores utilizaram sua tese para desenvolver suas maquinas, como John Eckert que juntamente com John Mauchly apresentaram ao mundo o ENIAC. Com o passar do tempo, a rede para se montar ou criar tais computadores ficou insustentavel, então houve-se a necessidade de buscar outros tipos de Arquitetura para aprimorar as maquinas.

A arquitetura superescalar foi desenvolvida no inicio da década de 1980, por Maurice Wilkes e David Patterson, com o intuito de permitir a execução de múltiplas intruções em paralelo, para melhorar o desempenho. Tal tecnologia foi impulsionada pelo avanço da tecnologia dos semicondutores, que permitiu a integração de multiplos transistores em um unico chip.

A computação com base na arquitetura superescalar trouxe muitas inovações para a época como agendamento de instruções, ou a execução paralela, principal foco da arquitetura, veremos mais a seguir (Wikipedia, sem data):

- Execução Paralela: Como o próprio nome da diz, ela utiliza-se da execução de multiplas tarefas em paralelo, ao mesmo tempo. Isso só é possivel através da utilização de multiplas

unidades de execução, como unidades de ponto flutuante, unidades de carga ou unidades de inteiro.

- Execução fora de ordem: Computadores antigos sofriam com a espera dos dados, ou execução de códigos desnecessarios, esta característica da computação superescalar veio para mudar isso, tirando a necessidade do código ser executado linha a linha, e dando a ele um foco, dum modo onde ele aproveita suas unidades de execução e evita atrasos causados dados ou instruções desnecessarias.

- Renomeação de Registradores: Tal característica vem de encontro a anteriormente citada, para que a execução fora de ordem aconteça pesquisadores desenvolveram um modo de atribuir registradores diferentes para as intruções, permitindo que elas sejam reordenadas sem conflitos de dependência de dados.

- Agendamento de Instruções: Esta é outra característica crucial para a computação superescalar, onde o processador decide qual será a instrução a executar em determinado ciclo de clock, maximizando a utilização das unidades de execução e minimizando os atrasos.

- Buffering de Instruções e Resultados: Para lidar com a execução paralela de múltiplas instruções, os processadores superescalares incluem buffers de instruções e resultados. Os buffers de instruções mantêm instruções que foram buscadas, mas ainda não executadas, enquanto os buffers de resultados mantêm os resultados de instruções que foram concluídas, mas ainda não estão prontos para serem gravadas na memória ou nos registradores.

Com tantas características e funções a arquitetura superescalar necessitava de uma separação, para dar uma enfâse maior a cada objetivo, e oferecer uma melhor perfomance ao cliente. Tendo em vista isso os cientistas buscaram aproveitar ao maximo cada característica apresentada por esta arquitetura, dando origem a varios tipos de computação superescalar (Docomomo Brasil, sem data).

- Arquitetura Superescalar baseada em Emissão de Instruções Estáticas e Dinâmicas: A respeito da Emissão de Intruções Estáticas, as intruções dadas são emitidas para as unidades de execução por uma ordem predefinida pelo compilador. Um exemplo perfeito são os processadores com execução muito paralela, a famosa VLIW (Very Long Instruction Word), onde o compilador agrupa varias instruções em uma unica instrução larga, simplificando a logic de controle de hardware. Ao contrario da Estática, a Emissão de Instruções Dinâmicas decide qual será a instrução emitida para a unidade de execução de acordo com a disponibilidade de recursos e resolução de dependencias.

- Arquitetura Superescalar baseada no Número de Instruções Emitidas: A cerca deste tipo temos duas ramificações o Dual-Issue e o Multi-Issue, onde como o próprio nome ja diz, o Dual-Issue permite apenas duas intruções de execução por ciclo de clock, e já o Multi-Issue permite mais de duas instruções de execução por clock. Atualmente as empresas se utilizam mais do Multi-Issue, como nos processadores da linha Intel Core I7 e AMD Ryzen.

- Arquitetura Superescalar baseada na Organização de Execução: Neste tipo de arquitetura, há duas diferencas principais nos processadores, aquele no qual as unidades de execução utilizadas são todas do mesmo tipo e podem se utilizar de qualquer instrução, a chamada Superescalar Simétrica, e por outro lado temos um processador com unidades de execução especificas, do qual ele contem unidades especializadas para cada area, aritmética, ponto flutuante, acesso à memoria, entre outros, a chamada Superescalar Assimétrica. A utilização dos processadores simétricos se deu com maior enfâse no começo da era computacional, pois simplificava o design, agora com um conhecimento maior no hardware e com mais recursos, as empresas preferem se utilizar do modelo assimétrico, onde vemos processadores com ALUs (Arithmetic Logic Units) ou FPUs (Ploating Point Units) por exemplo, os quais se utilizam melhor dos recursos de execução e dão maior eficiencia ao sistema.

A arquitetura superescalar varia amplamente em termos de design e implementação, dependendo das necessidades específicas de desempenho e eficiência. Outra arquitetura que tem um design à imagem da superescalar é a Arquitetura Pipeline, o que diferencia uma da outra é a questão do ciclo do clock, enquanto a arquitetura superescalar permite a multi execução de instruções no mesmo cliclo de clock, a arquitetura pipeline não permite tal funcionalidade, dando a multi execução de instruções em clicos diferentes de clock (DUCATTE, 2010).

Pipeline nada mais é do que a divisão das tarefas, ele decompoe uma instrução em varias etapas, desta forma ele possibilita que varias instruções sejam executadas em diferentes fases do pipeline, da divisão. O conceito de Pipeline pode ser comparado as teses antigas da Industria, como o fordismo, onde Henry Ford propos as insdustrias de montagem dividir o trabalho de forma que o produto final seja montado em etapas sucessivas, da mesma forma sera o processador que se utiliza do pipeline, a instrução ira passar por diferentes etapas desde a busca na memoria até a visualização do resultado (PANTUZA, sem data).

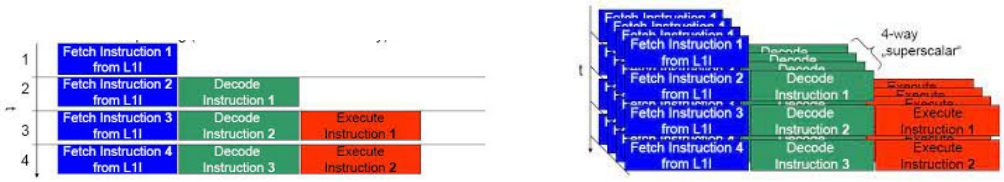

Figura 4 – Diferenca Pipeline x Superescalar

Desta forma, no pipeline, uma instrução deve ser dividida em varias etapas, e como esta divisão ocorre varia de processador a processador, mas em sua maioria ele retem as seguintes etapas (TECNOBLOG,):

- Busca de Instrução (Fetch): A instrução é buscada da memória.
- Decodificação de Instrução (Decode): A instrução buscada é decodificada para determinar a operação a ser realizada.
- Execução (Execute): A operação especificada pela instrução é realizada.

- Acesso à Memória (Memory Access): Qualquer acesso à memória necessário pela instrução é realizado.
- Escrita de Resultado (Write Back): O resultado da execução é escrito de volta ao registrador apropriado.

Ao dividir as tarefas de uma instrução, o pipeline permite que o processador inicie uma instruçao a cada ciclo de clock aumentando a taxa de instruções concluidas por unidade de tempo. Outro beneficio desta arquitetura é em relação ao hardware do sistema, o pipeline consegue utilizar de todas as partes do processador o tempo todo, evitando a ociosidade das peças.

Como podemos perceber o pipeline trouxe muitas mudanças e beneficios a computação, de tal forma que muitos computadores atuais utilizam tal tecnologia, tais como Intel Core I7 ou I9, ou até mesmo os Ryzen. Com tantas qualidades, o pipeline ainda enfrenta alguns desafios para sua total aprovação, como os problemas Hazards, que são problemas aonde uma instrução interfere em outra ocasionando outro problema as Stalls ou mais conhecidas Bolhas, aonde o pipeline é pausado para resolver as Hazards, tal processo pode reduzir seu desempenho. Em relação a sua implementação, existem dois desafios principais em relação ao custo de complexidade devido ao design da arquitetura e o gerenciamento de energia (ASTERA,)

2.4 VLIW (Very Long Instruction Word)

A arquitetura VLIW (Very Long Instruction Word), em português: Instrução de Palavra muito Grande, tenta alcançar maiores níveis de paralelismo de instrução pela execução de instruções longas compostas por múltiplas operações. Um processador VLIW permite que programas especifiquem explicitamente instruções a serem executadas em paralelo, enquanto as CPUs convencionais geralmente permitem que programas especifiquem instruções a serem executadas apenas em sequência.

Para isso, as instruções são divididas em subpassos para que elas sejam executadas quase que ao mesmo tempo (pipeline), despachando instruções individuais para serem executadas de forma independente, em diferentes partes do processador (arquiteturas superescalares) e até mesmo executar instruções em uma ordem diferente do programa (execução fora de ordem) (Wikipedia contributors, b).

Tal abordagem complica o hardware, visto que o processador fica reponsável por tomar todas as decisões internamente, porém, por depender de que os programas forneçam todas as decisões sobre quais instruções executar simultaneamente e como resolver conflitos, o software envolvido torna-se mais complexo, enquanto o hardware, por sua vez, é mais simples do que em outros tipos de paralelismo.

2.5 Arquiteturas Vetoriais

Arquiteturas vetoriais representa uma ferramenta de processamento de um grande volume de dados simultaneamente. Essa tecnologia possui os vetores como base, que funcionam como uma

lista de dados armazenado na memória. A partir da presença dos vetores, é possível promover aplicações simultâneas em todos os elementos presentes, em vez de aplicações individuais e sequenciais para cada item.

Os processadores vetoriais são classificados como SIMD (Single Instruction Multiple Data), isto é, uma única instrução é capaz de executar múltiplos dados. Ademais, outro ponto marcante é a presença de registradores vetoriais, o que proporciona certa vantagem, já que o programador pode pensar de forma sequencial enquanto opera sobre recursos paralelos, em decorrência dos vetores.

As instruções vetoriais são equivalentes a um "loop", o cálculo de cada operação independe do resultados anteriores. Além disso, é possível haver pipelines profundos sem a dependência de dados.

Os processadores vetoriais se dividem em 2 classificações: memória - memória e registrador - registrador. No primeiro caso, os operandos são da memória diretamente ao processador, e a medida que que as operações vetoriais são executadas, os resultados são gravados na memória. Já no segundo caso, os registradores vetoriais fazem a leitura dos operandos e o resultado, após a execução pelo processador, retorna aos registradores e posteriormente são gravados na memória.

Modelo do processador vetorial:

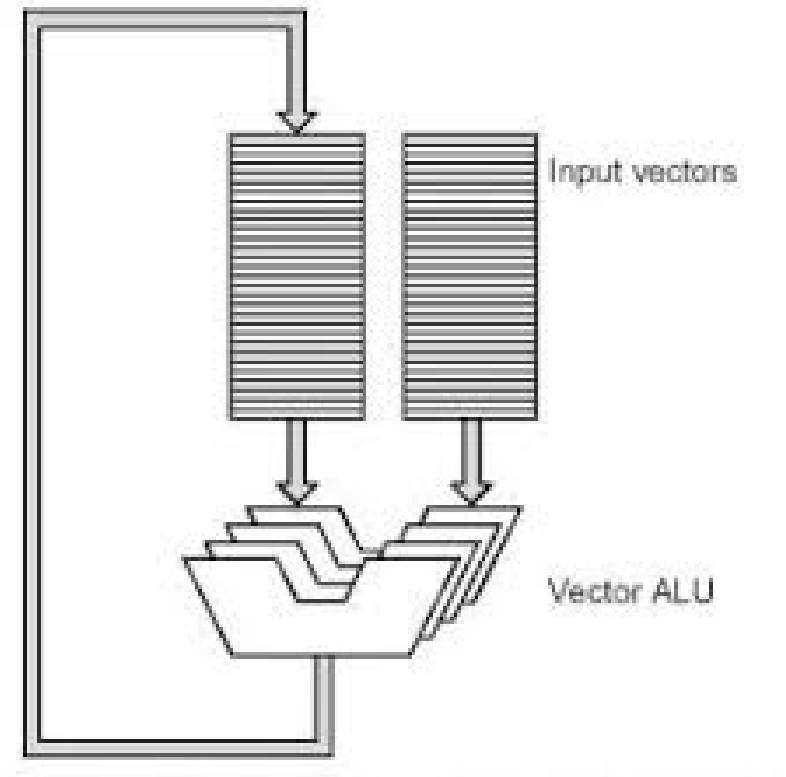

Figura 5 – Arquitetura Vetorial

(SANTANA, a)

Como vantagem dessa tecnologia, é possível apontar uma alta taxa de clock, com possibilidade de ganhos por replicação em hardware, o decaimento da quantidade de buscas e instruções, ausência da verificação de atritos entre dados e ausência de sobrecarga, já que as

partículas são atômicas. Entretanto, como desvantagem, pode-se apresentar o alto custo de manutenção, operação e desenvolvimento (ARTIGO, 2015).

2.6 Multiprocessadores e Multicomputadores

2.6.1 O que são Multiprocessadores?

Um multiprocessador trata-se de um processador com vários núcleos de processamento, isso quer dizer que dentro de si contem mais de uma unidade de processamento principal, as famosas CPUs. A principal característica dos multiprocessadores e o que diferencia eles de outros processadores comuns, é a capacidade de executar multiplas tarefas ao mesmo tempo, divindo as tarefas entre seus nucleos. (Wikipedia contributors, a)

Com o avanço da tecnologia se desenvolveu alguns tipos de multiprocessadores, como os dual-core, quad-core ou octa-core, que são amplamente utilizados pelos processadores da maioria das maquinas. Além destes, existem também arquiteturas mais avançadas como o SMP, o processamento simétrico de múltiplos processadores, aonde vários processadores físicos trabalham em conjunto, partilhando o mesmo sistema de memória e interconexão. (Tecnoblog,)

2.6.2 O que são Multicomputadores?

Multicomputadores são sistemas computacionais que consistem em vários computadores interligados, cada qual com suas caracteristicas tanto de hardware, como de software, mas ambos trabalhando juntos na solução de problemas computacionais. (SANTANA, b)

A interligação destes computadores ocorre de diferentes formas, a mais comum é atraves de nós, e esses nós podem se dar de varias maneiras, o que chamamos de topologia de nós, a primeira topologia se refere ao barramento, aonde todos os nós são ligados em unico cabo ou barramento; a segunda topologia se da por meio de uma conexão em um unico aparelho central, seja ele um hub ou um switch, o que chamamos de Topologia Estrela; e a terceira topologia se da por meio da conexão de nós, cada qual se ajunta com mais dois nós, formando um anel, dai o nome Topologia em Anel; A topologia Malha, aonde cada nó esta conectado a todos os outros nós; Utilizando-se da topologia estrela, mas acrescentando uma estrutura hierárquica, vemos a topologia arvore; Existe uma topologia que se assemelha muito a uma forma geométrica, aonde cada nó é representado por vértices num hiper-cubo n-dimensional; E por fim, a topologia que abrnage todos os nós, Rde Completa, aonde todos os nós estão diretamente conectados aos outros nós.(CRUZ,)

Além da divisão pela interligação dos computadores, existe uma tipificação de cada multicomputador, segundo o tamanho da rede e seu uso.

- Cluster de Computadores: Conjunto independente de computadores conectados em uma rede local.
- Grid Computing: Combina recursos/resultados de varios computadores distribuídos geograficamente para resolver um problema em comum

Abaixo teremos um quadro comparativo para exibir e expressar melhor a diferença de cada um dos multiprocessadores

Tabela 1 – Tipo de Multicomputadores

Tipo de Computadores	Descrição	Vantagens	Desvantagens	Exemplos
Cluster de Computadores	Conjunto de Computadores conectados por uma rede local	Fácil escalabilidade e Redundância	Desempenho reduzido com muitos nós e requer gerenciamento	Beowulf Clusters
Grid Computing	Computadores distribuídos geograficamente conectados por uma rede ampla	Flexibilidade e Aproveitamento de recursos ociosos	Latência de Rede e Complexidade de Gerenciamento	Folding@home
Computação em Nuvem	Recursos computacionais fornecidos pela internet	Escalabilidade elástica, custo efetividade, alta disponibilidade	Dependência da conectividade de internet e custos recorrentes	Amazon Web Services (AWS), Google Cloud, Microsoft Azure
Supercomputadores	Sistemas com milhares de processadores trabalhando em paralelo	Ato desempenho e capacidade de resolver problemas	Alto custo e complexidade de configuração e manutenção	Summit, supermaquina IBM
Computação Distribuida	Múltiplos computadores colaborando para realizar uma tarefa	Descentralização, escalabilidade e tolerância a falhas	Latência de rede e Complexidade de Gerenciamento	Apache Hadoop
Arquitetura de Computação Hibrida	Combinação de clusters locais e serviços em nuvem	Flexibilidade e Otimização dos Custos	Complexidade de Integração e Dependência da conectividade de internet	Integracão de clusters locais com serviços em nuvem

2.7 Computação em nuvem e serviços

2.7.1 Computação em nuvem

A computação em nuvem é desenvolvida a partir de arquiteturas paralelas, uma vez que, possibilita a execução de múltiplas tarefas de maneira simultânea e distribuída.

A infraestrutura dessa tecnologia é composta por um grande número de máquinas físicas de baixo custo, interligados por uma rede. Cada dispositivo possui o mesmo tipo de software,

rntrtanto pode haver variação nas capacidades de processamento e armazenamento. (Wikipedia, b).

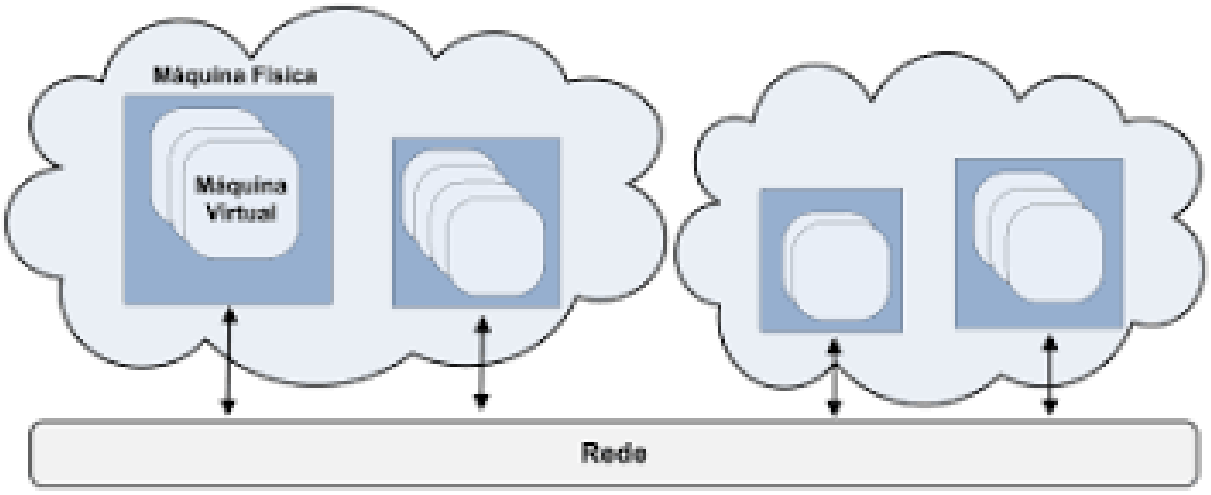

Figura 6 – Computação em Nuvem

A computação em nuvem também detém de um importante tipo de paraelismo denominado cluster, como citado anteriormente, que representa uma rede composta por vários nós que não dependem de proximidade física.

Dentre as vantagens da utilização do paralelismo de cluster na computação em nuvem, destaca-se a escalabilidade, ou seja, existe a possibilidade de aumento do poder de processamento e área de armazenamento sob demanda. Outra vantagem evidente, é eficiência temporal, já que é possível alcançar um tempo de processamento baixo utilizando um grande volume de dados (Controle.Net,).

2.7.2 AWS Lambda

A AWS Lambda é um serviço de computação sem servidor oferecido pela Amazon Web Services, seu objetivo é oferecer um serviço de computação sem servidor que permite executar código em resposta a eventos sem precisar provisionar ou gerenciar servidores, além disso, só paga pelo tempo de computação consumido pelo seu código.

Nesse serviço oferecido o usuário carrega o código da função e em seguida define o simpósio que acionam a execução dessa função. Quando tal contingência ocorre, a AWS Lambda executa o código associado e conduz automaticamente os recursos necessários em sua execução (Amazon Web Services,).

Devido sua dimensão, automaticamente sua infraestrutura de computação é de acordo com a demanda, ou seja, que se você tiver picos de tráfego, a AWS Lambda pode escalar instantaneamente para lidar com a carga adicional, algo que é fornecido graças a computação em nuvem.

Em sua parte de segurança, obtém recursos de robustos, isso incluí IAM (Identity and Access Management), que controla o acesso aos recursos da AWS, além de integração com VPC (Virtual Private Cloud) para isolar as funções Lambda em uma rede privada.

No seu programa obstrui linguagens de programação, permite criar tempos de execução personalizados, o que permite aos desenvolvedores executarem código em linguagens que não são suportadas nativamente. Além disso, fornece recursos integrados de monitoramento e logging, permitindo rastrear o desempenho e o comportamento das suas funções, bem como diagnosticar problemas rapidamente (Wikipedia, a).

2.7.3 Azure Functions

Azure é uma nuvem complexa, dotada de ramificações para atender todas as demanadas. O Azure Functions representa uma dessas ramificações, responsável por prestar serviços de computação sem servidor.

Essa tecnologia permite a execução de códigos a paartir de eventos específicos. Essas execuções são feitas sob demanda e em rapidez, já que a nuvem permite uma rápida disponibilização de recursos para a execução dessas ações (Eximia,).

Dentre os benefícios desse serviço, é possível evidenciar os seguintes aspectos:

- Escalabilidade automática: as respostas do serviço são devolvidas a partir de uma certa demanda, o que garante a cobrança daquilo que é realmente utilizado.
- Integração profunda: o azure functions s integra com outros serviços da própria plataforma, como azure storage e azure service bus.
- Linguagens e frameworks diversificados: é possível a escrita de funções em diversas linguagens e estipular um tempo personalizado para a execução dessas ações.
- Monitoramento: é possível rastrear e depurar, de maneira avançada, todas as funções estabelecidas.

Em resumo, a Azure Functions utiliza computação em nuvem como uma ferramenta, visando promover flexível, escalável e segura para a execução de determinadas funções sob demanda (Coodesh,).

2.7.4 Google Cloud

A tecnologia google cloud é um serviço que utiliza conjuntos de recursos físicos e recursos virtuais, os quais estão localizados em data centers por todo o mundo. Essa distribuição é promove diversas vantagens, uma vez que promove latência reduzida em locais próximos aos clientes e a possibilidade de utilização de recursos simultânemanete.

Essa plataforma é responsável por proporcionar uma capacidade de armazenamento de dados com eficiência na nuvem, ou seja, essas informações podem ser acessadas de qualquer lugar, o que faz relação com o conceito de arquiteturas paralelas.

Além disso, o google cloud também oferecem uma infraestrutura escalável, já que os clientes podem gerenciar aplicações que usam vários núcleos de cpu em paralelo, dividindo assim as cargas de processamento.

Ademais, é valido apontar a existência de uma ramificação do google cloud, chamada google cloud dataflow, a qual viabiliza a manipulação de um grande volume de dados (Google Cloud,).

2.7.5 Comparativo entre os serviços

A partir dos serviços apresentados, é possível comparar cada plataforma e apresentar as particularidades de cada uma.

Tabela 2 – Comparativo Computação em Nuvem

Tabela Comparativa			
Características	AWS	Azure	Google Cloud
Computação	EC2 e Lambda	Virtual Machines e Azure Functions	Compute Engine e Cloud Functions
Armazenamento	S3 e Glacier	Blob Store e Azure Files	Cloud Storage e Persistent Disks
Banco de Dados	RDS e DynamoDB	SQL Database e Cosmos DB	Cloud SQL e Firestore
Rede	VPC e Direct Connect	Virtual Network e ExpressRoute	Virtual Private Cloud e Cloud Interconnect
Inteligencia Artificial	SageMaker e Rekognition	Cognitive Services e Azure Machine Learning	AI Platform e Vision AI
DevOps	CodeBuild e CodeDeploy	Azure DevOps e Azure Pipelines	Cloud Build e Cloud Deploy
Ferramentas de Migração	Database Migration Service	Azure Migrate	Database Migration Service

2.8 Inteligencia Artificial e Machine Learning Paralelo

A Inteligência Artifial (IA) está cada vez mais presente no mundo contemporâneo e, junto a ela, há o treinamento envolvido dessas IAs: o Machine Learning (ML). Com isso, em busca de aprimorar o ML, o paralelismo pôde ser usado para aperfeiçoar essa área.

Modelos de Machine Learning grandes e complexos encontram alguns desafios, como longo tempo de treinamento, alto consumo de memória e escalabilidade limitada. Uma maneira de superar esses problemas é usar paralelismo de modelo, uma técnica que divide um modelo em partes menores e as distribui por vários dispositivos ou nós.

2.8.1 Paralelismo de Modelo

O paralelismo de modelo é mais uma das formas de computação paralela, que divide um modelo de Machine Learning em submodelos e os atribui a diferentes dispositivos ou nós para

execução simultânea. Dessa maneira, cada dispositivo ou nó só precisa armazenar e processar uma fração do modelo, reduzindo os requisitos de memória e computacionais. O paralelismo de modelo também pode melhorar a precisão dos modelos de Machine Learning, permitindo arquiteturas maiores e mais expressivas que podem capturar padrões e relacionamentos mais complexos nos dados.

Um modelo pode ser dividido em dois métodos principais: camadas e tensores.

- Camadas: divide o modelo por camadas, de forma que cada dispositivo ou nó maipule uma ou mais camadas do modelo.
- Tensores: divide o modelo por tensores, de maneira que cada dispositivo ou nó manipule apenas uma parte de uma camada ou tensor.

Exemplificadamente, é possível dividir uma camada convolucional por canais, filtros ou até dimensões espaciais. Ferramentas como MPI (Message Passing Interface), utilizada para estabelecer uma comunicação entre os processos independetes e NCCL (NVIDIA Collective Communications Library), utilizada para facilitar a comunicação de múltiplas GPUs dentro de um nó, podem ser usadas para estabelecer comunicação entre submodelos por meio de protocolos de transmissão de mensagens, ou então utilizando de memória compartilhada ou até sistemas de arquivos distribuídos (Wikipedia, c), (NVIDIA,).

Dentre os benefícios do paralelismo de modelo na área do Machine Learning, é possível destacar a redução do espaço ocupado pela memória do modelo, permitindo que modelos maiores e mais complexos sejam treinados; a aceleração do processo de treinamento, por meio da distribuição da carga de trabalho entre os dispositivos ou nós, diminuindo a latência e aumentando a taxa de transferência e, por fim, aumentando a capacidade do modelo por meio da adição de mais dispositivos ou nós sem alterar a arquitetura ou os hiperparâmetros do modelo (LinkedIn,).

2.9 Banco de dados paralelos

Com o desenvolvimento da tecnologia, ao nivel que vemos hoje em dia, principalmente a respeito do desenvolvimento do machine learning, vemos que os bancos de dados tem tido uma defasagem enorme. Tal problema se da pela sociedade informacional, uma sociedade que vive e se desenvolve atraves da informação, e deve ser alimentada por ela, de tal maneira que se torna vital para sobrevivencia tecnologica da sociedade.

Tendo isso em vista, desenvolve-se a necessidade de criar um novo banco de dados, uma nova forma de guardar e reter tais informações de uma forma mais rapida e facil. Então por volta dos anos 70 e 80, começam a surgir pesquisas a respeito dos Bancos de Dados Paralelos, os quais vieram com o mesmo intuito de guardar os dados, fornecendo uma resposta rapida as requests, pretendendo aprimorar a velocidade das mesmas e lidar com grandes quantidades de dados.

Um banco de dados paralelo é um tipo de banco de dados projetado para acelerar o processamento e análise de grandes volumes de informações, dividindo as tarefas entre vários computadores ou processadores que trabalham simultaneamente. Em vez de um único processador

fazer todo o trabalho, várias unidades processam partes diferentes dos dados ao mesmo tempo, resultando em respostas mais rápidas para consultas complexas e maior capacidade de lidar com grandes quantidades de dados (OLGUIN, sem data)

Diferentemente dos bancos de dados relacionais comumente utilizados nos dias atuais, os quais se utilizam de duas ou mais entidades as relacionando entre si, salvando os dados em uma ou na outra e os separando conforme desejado, apresentando a informação atraves de um select bem desenvolvido, os bancos paralelos salvam uma informação de uma entidade em diferentes bancos de dados, particionando e separando a informação, de modo que ao darmos um select, ao pesquisarmos determinada informação, sua busca fique mais rapida, pois ela estara, mesmo que dividida, em varios bancos de dados.

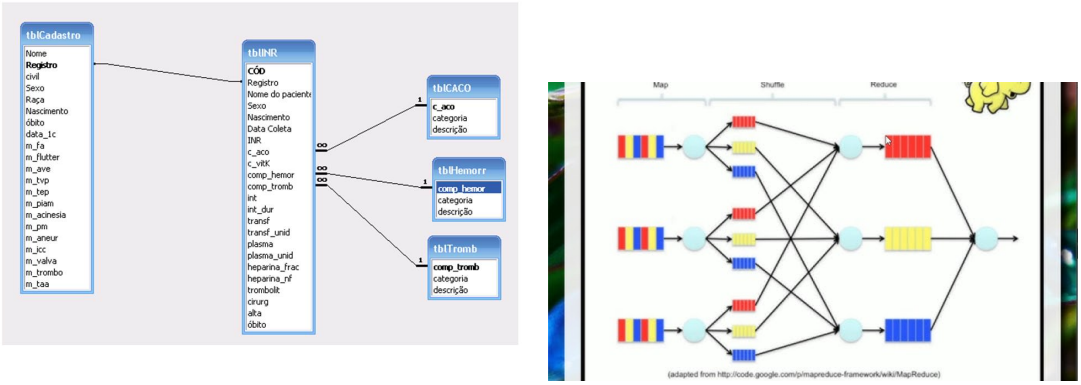

Figura 7 – Diferenca BD Relacional X BD Paralelo

Como visto este banco de dados traz consigo muitas características das arquiteturas aqui ja citadas, como a utilização do processamento paralelo e o particionamento de dados, mas também traz consigo caracteísticas novas como sua flexibilidade, sua escalabilidade, sua tolerância a falhas evitando a redundancia e desenvolvendo a recuperação rapida de seus dados. No demais os bancos de dados paralelos tem razidos muitas inovações como a integração a novas tecnologias como o BigData ou a Computação em Nuvem, trazendo maior rapidez no processamento de dados, aumentando sua capacidade de armazenamento e diminuindo os custos (TESE,).

Numa visão geral os bancos de dados paralelos representam um avanço significativo no mundo tecnologico, principalmente pela forma com que ele gerencia grandes quantidades de dados, oferecendo soluções para as crescentes demandas de desempenho, escalabilidade e disponibilidade. Dentre alguns anos sua utilização sera vital em organizações que lidam com grandes volumes de dados, pois ao distribuir as tarefas de processamento, realizando o particionamento da informação, os bancos paralelos conseguem acelerar a execução de consultas mais complexas, algo fundamental para muitas aplicações modernas (Centro de Informática - UFPE,).

2.10 Desafios e limitações das arquiteturas paralelas

Embora as arquiteturas paralelas estejam atreladas ao ideal vantajoso, é de suma importância transparecer as adversidades agregadas a essa tecnologia.

2.10.1 Complexidade de desenvolvimento

Há uma necessidade de garantir que os diferentes processadores existentes acessem recursos de maneira segura. Além disso, em razão desse tipo de arquitetura ser mais complexa, existe uma dificuldade na correção de erros presentes (MEDIDAS-DE-DESENVOLVIMENTO,).

2.10.2 Sobrecarga de processamento

A presença de muitos processadores focados em um número limitado de tarefas pendentes, pode influenciar como um todo, provocando certa latência e um desempenho abaixo do esperado.

2.10.3 Custo alto para desenvolvimento

Apesar da ideia de muitos processadores trabalhando ao mesmo tempo para solucionar algo seja ótimo, é imprescindível evidenciar que a medida que o número de processadores paralelos cresce, o custo de desenvolvimento e manutenção também se assemelha (Sistemas 24 Horas,).

2.10.4 Limite de desempenho

Segundo a lei de Amdahl existe uma limitação na velocidade de um sistema baseado em arquiteturas paralelas. De acordo com a teoria, existe uma pequena parcela do sistema que não pode ser paralelizada, o que limitará a velocidade real que o paralelismo poderia oferecer.

Na figura a seguir, é apresentado na cor azul o aumento possível, referente a aceleração do sistema, em um caso ideal, enquanto na cor rosa, é apresentado o aumento, também referente a aceleração do sistema, em um caso real. Na mesma linha de raciocínio, uma curva amarela representa o tempo ideal do sistema e uma curva vermelha faz referência ao tempo de execução real.

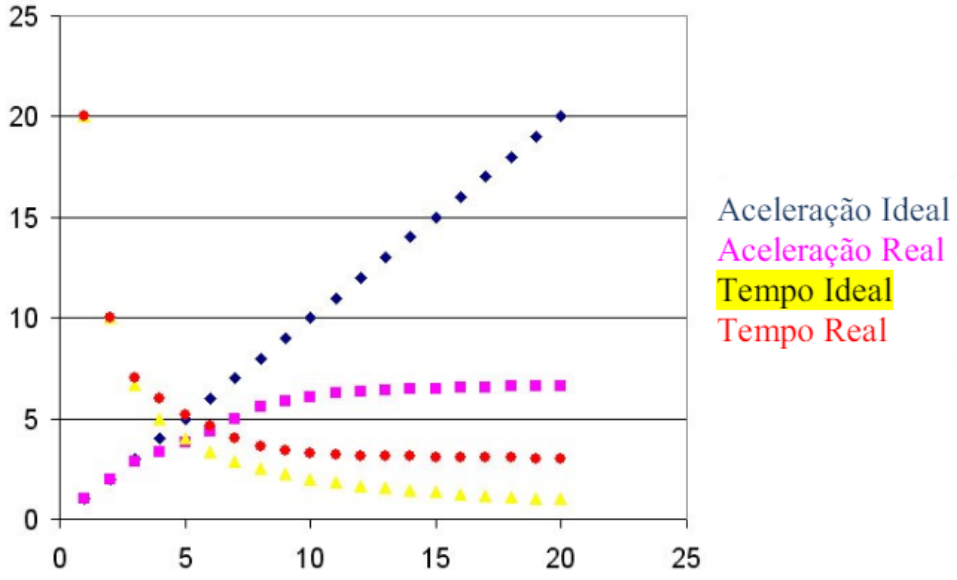

Figura 8 – Gráfico - lei de Amdahl

(LIMITE-DESEMPENHO-PARALELISMO,)

3 Conclusão

A computação em estruturas paralelas representa um avanço considerável na capacidade de processamento computacional ao possibilitar a realização de múltiplas ações e tarefas simultâneamente. Desde os princípios básicos de paralelismo em nível de bit até estruturas superescalares e vetoriais, diversas estratégias foram criadas para melhorar o desempenho dos sistemas computacionais. Essas estruturas não só diminuem gargalos convencionais, como a restrição da velocidade da memória em comparação com o processador, mas também permitem o processamento de grandes quantidades de dados de maneira mais eficiente.

Com a necessidade cada vez maior de análises de dados elaborados e simulações detalhadas em setores como inteligência artificial e física computacional, o paralelismo passou a estar sob forte demanda. Ademais, auxiliam no avanço em áreas importantes para a prestação de serviços para a sociedade, como previsão do clima e computação em nuvem, promovendo a inovação tecnológica e a criação de aplicações mais complexas.

Dessa forma, as estruturas paralelas surgem como uma alternativa para contornar as limitações dos sistemas computacionais convencionais, apresentando-se como uma base para o desenvolvimento da computação de alta performance e escalabilidade. Isso contribui para impulsionar importantes progressos na ciência e tecnologia, influenciando diretamente o cenário digital que se desenvolve cada vez mais.

Referências

Amazon Web Services. *AWS Lambda*. <https://aws.amazon.com/pt/lambda/>. Acessado em 18 de junho de 2024. Citado na página 17.

ARTIGO, A. do. Arquiteturas-vetoriais. *International Journal of Critical Accounting and Economics*, v. 4, n. 1, p. sem páginas, 2015. Acessado em 18 de junho de 2024. Disponível em: <https://www2.sbc.org.br/ceacpad/ijcae/v4_n1_dec_2015/IJCAE_v4_n1_dez_2015_paper_3_vf.pdf>. Citado na página 15.

ASTERA. *Papeline*. <https://www.astera.com/pt/type/blog/data-pipeline-architecture/>. Acessado em 18 de junho de 2024. Citado na página 13.

Centro de Informática - UFPE. *Resumo*. <https://www.cin.ufpe.br/~if694/aulas_pdf/12>. Acessado em 18 de junho de 2024. Citado na página 21.

Controle.Net. *O-que-é-Cluster?* <https://www.controle.net/faq/o-que-e-cluster>. Acessado em 18 de junho de 2024. Citado na página 17.

Coodesh. *Azure-vantagens*. <https://coodesh.com/blog/dicionario/o-que-e-azure-functions/>. Acessado em 18 de junho de 2024. Citado na página 18.

CRUZ, E. *Multiprocessadores-Multicomputadores*. <https://www.gta.ufrj.br/~cruz/courses/eel770/slides/17_multiproc_multicomp.pdf>. Acessado em 18 de junho de 2024. Citado na página 15.

Docomomo Brasil. *O-que-é-Arquitetura-Superescalar*. sem data. <https://docomomo.org.br/glossario/o-que-e-arquitetura-superescalar/>. Acessado em 18 de junho de 2024. Citado na página 11.

DOCUMENTO, A. do. *Taxonomia*. <https://www.maxwell.vrac.puc-rio.br/16578/16578_4.PDF>. Acessado em 18 de junho de 2024. Citado 2 vezes nas páginas 10 e 27.

DUCATTE, M. *Arquiteturas-Superescalares*. 2010. <https://www.ic.unicamp.br/~ducatte/mo401/1s2010/T2/100602-t2.pdf>. Acessado em 18 de junho de 2024. Citado na página 12.

Eximia. *Azure*. <https://eximia.co/o-que-e-e-por-que-usar-azure-functions-relacao-com-serverless-architectures-e-pontos-de-atencao/>. Acessado em 18 de junho de 2024. Citado na página 18.

Google Cloud. *Google-Cloud*. <https://cloud.google.com/docs/overview?hl=pt-br>. Acessado em 18 de junho de 2024. Citado na página 19.

IBGE. *Paralelismo-cluster*. <https://www.ibge.gov.br/confest_e_confege/pesquisa_trabalhos/CD/palestras/368-1.pdf>. Acessado em 18 de junho de 2024. Citado na página 9.

LIMITE-DESEMPENHO-PARALELISMO. <https://www.maxwell.vrac.puc-rio.br/38359/38359_4.PDF>. Acessado em 18 de junho de 2024. Citado na página 22.

LinkedIn. *Utilizando-paralelismo-modelo*. <https://pt.linkedin.com/advice/1/how-can-you-use-model-parallelism-improve-machine-z7tsf?lang=pt>. Acessado em 18 de junho de 2024. Citado na página 20.

MEDIDAS-DE-DESENVOLVIMENTO. <https://www.inf.pucrs.br/~emoreno/undergraduate/SI/orgarq/class_files/Aula17.pdf>. Acessado em 18 de junho de 2024. Citado na página 22.

Midorikawa. *ERAD-sp*. 2010. Documento PDF. Disponível em: <https://eradsp2010.wordpress.com/wp-content/uploads/2010/10/curso1_arquiteturas_midorikawa.pdf>. Citado na página 6.

NVIDIA. *NVIDIA NCCL*. <https://developer.nvidia.com/nccl>. Acessado em 18 de junho de 2024. Citado na página 20.

O-QUE-é-BIT-LEVEL-PARALLELISM. sem data. Acessado em 18 de junho de 2024. Disponível em: <https://napoleon.com.br/glossario/o-que-e-bit-level-parallelism/>. Citado na página 7.

OLGUIN, R. *Banco-de-Dados-Paralelo*. sem data. <https://www.inf.unioeste.br/~olguin/4458-semin/G6-apresentacao.pdf>. Acessado em 18 de junho de 2024. Citado na página 21.

PANTUZA, B. *Introdução-Pipeline*. sem data. <https://blog.pantuza.com/artigos/organizacao-e-arquitetura-de-computadores-pipeline-em-processadores>. Acessado em 18 de junho de 2024. Citado na página 12.

Portal Insights. *O-que-é-execução-fora-de-ordem?* Acessado em 18 de junho de 2024. Disponível em: <https://www.portalinsights.com.br/perguntas-frequentes/o-que-e-execucao-fora-de-ordem>. Citado na página 8.

SANTANA, M. *Processador-vetorial*. <https://www.docentes.univasf.edu.br/max.santana/material/aoc-ii/ProcessadoresVetoriais.pdf>. Acessado em 18 de junho de 2024. Citado na página 14.

SANTANA, M. *Sistemas-Multicomputadores*. <https://www.docentes.univasf.edu.br/max.santana/material/OAC-II/sistemas_multicomputadores.pdf>. Acessado em 18 de junho de 2024. Citado na página 15.

Sistemas 24 Horas. *Sobrecarga-custo*. <https://www.sistemas24horas.com.br/aulas/files_semi2018/processamento-paralelo-distribuido-001.pdf>. Acessado em 18 de junho de 2024. Citado na página 22.

Sistemas24horas. <https://www.sistemas24horas.com.br/aulas/files_semi2018/processamento-paralelo-distribuido-001.pdf>. Accessed: 2024-5-10. Citado na página 6.

SOBRAL, B. *Paralelismo-de-Dados-Paralelismo-de-Tarefas-Paralelismo-Pipelining*. <https://www.inf.ufsc.br/~bosco.sobral/ensino/ine5645/Paralelismo_de_Dados_Paralelismo_de_Tarefas_Paralelismo_Pipelining.pdf>. Acessado em 18 de junho de 2024. Citado 2 vezes nas páginas 8 e 9.

Tecnoblog. *Tipos-multiprocessador*. <https://tecnoblog.net/responde/o-que-e-nucleo-core-do-processador/>. Acessado em 18 de junho de 2024. Citado na página 15.

TECNOBLOG. *Tipos-Pipeline*. <https://tecnoblog.net/responde/o-que-e-pipeline-processador/>. Acessado em 18 de junho de 2024. Citado na página 12.

Tecnoblog Responde. *O-que-é-pipeline-de-processador?* sem data. Acessado em 18 de junho de 2024. Disponível em: <https://tecnoblog.net/responde/o-que-e-pipeline-processador/>. Citado na página 7.

TESE, A. da. *Conceitos-BD-Paralelo*. [S.l.]: PUC-Rio. <https://www2.dbd.puc-rio.br/pergamum/tesesabertas/0410861_06_cap_02.pdf>. Acessado em 18 de junho de 2024. Citado na página 21.

Wikipedia. *AWS-Lambda*. <https://pt.wikipedia.org/w/index.php?title=AWS_Lambda&oldid=67698394>. Acessado em 18 de junho de 2024. Citado na página 18.

Wikipedia. *Computacao-em-Nuvem*. Acessado em 18 de junho de 2024. Disponível em: <https://pt.wikipedia.org/wiki/Computa%C3%A7%C3%A3o_em_nuvem>. Citado na página 17.

Wikipedia. *Message-Passing-Interface*. <https://en.wikipedia.org/w/index.php?title=Message_Passing_Interface&oldid=1225276468>. Acessado em 18 de junho de 2024. Citado na página 20.

Wikipedia. *Superescalar*. sem data. <https://pt.wikipedia.org/w/index.php?title=Superescalar&oldid=54513103>. Acessado em 18 de junho de 2024. Citado na página 10.

Wikipedia contributors. *Multiprocessamento*. <https://pt.wikipedia.org/wiki/Multiprocessamento>. Acessado em 18 de junho de 2024. Citado na página 15.

Wikipedia contributors. *VLIW*. <https://en.wikipedia.org/wiki/Very_long_instruction_word>. Acessado em 18 de junho de 2024. Citado na página 13.

Wikipedia-contributors. *Computação Paralela — Wikipédia, a enciclopédia livre*. 2024. [Online; accessed 18-Jun-2024]. Disponível em: <https://pt.wikipedia.org/wiki/Computa%C3%A7%C3%A3o_paralela>. Citado na página 6.

A Apêndice

A.1 Taxonomia de Flynn

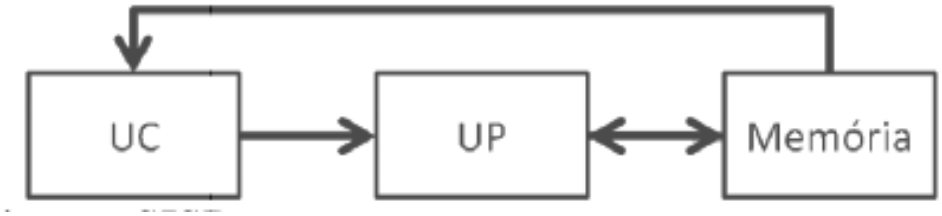

Figura 9 – Modelo arquitetura SISD

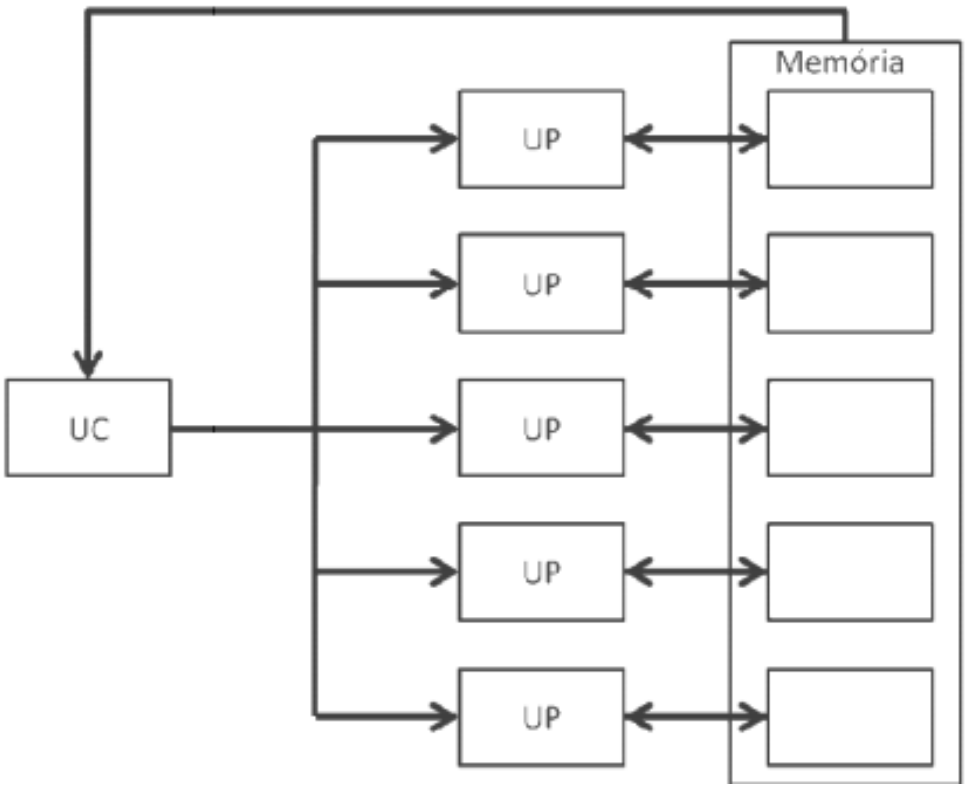

Figura 10 – Modelo arquitetura SIMD

(DOCUMENTO,)

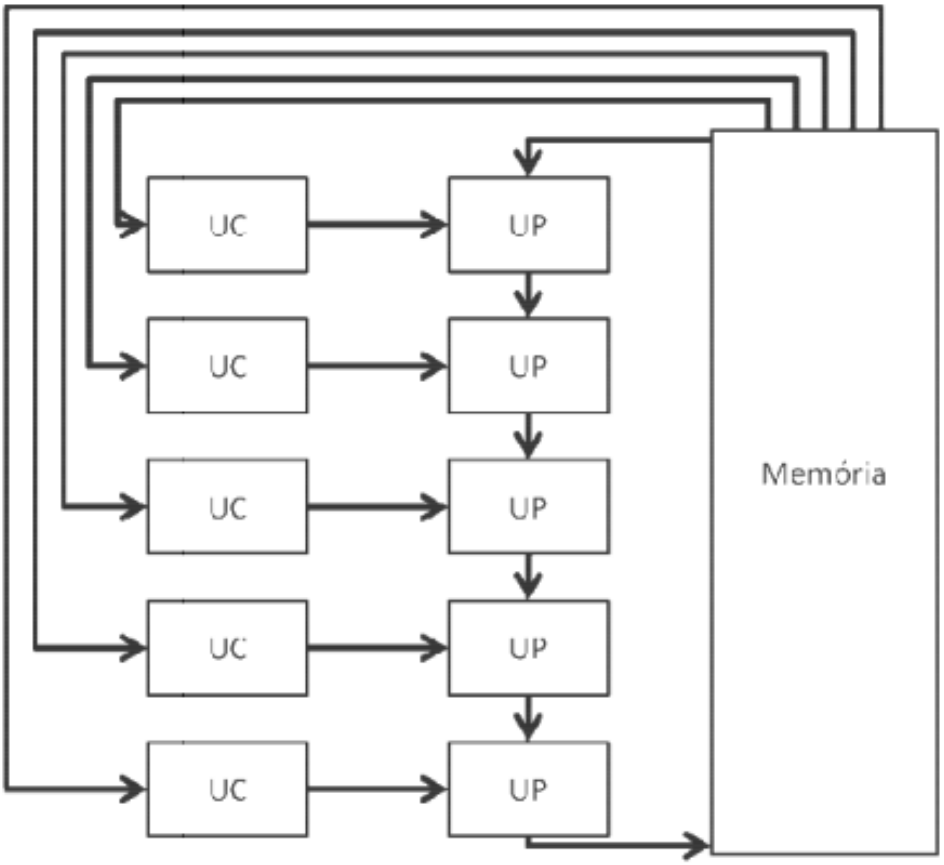

Figura 11 – Modelo arquitetura MISD

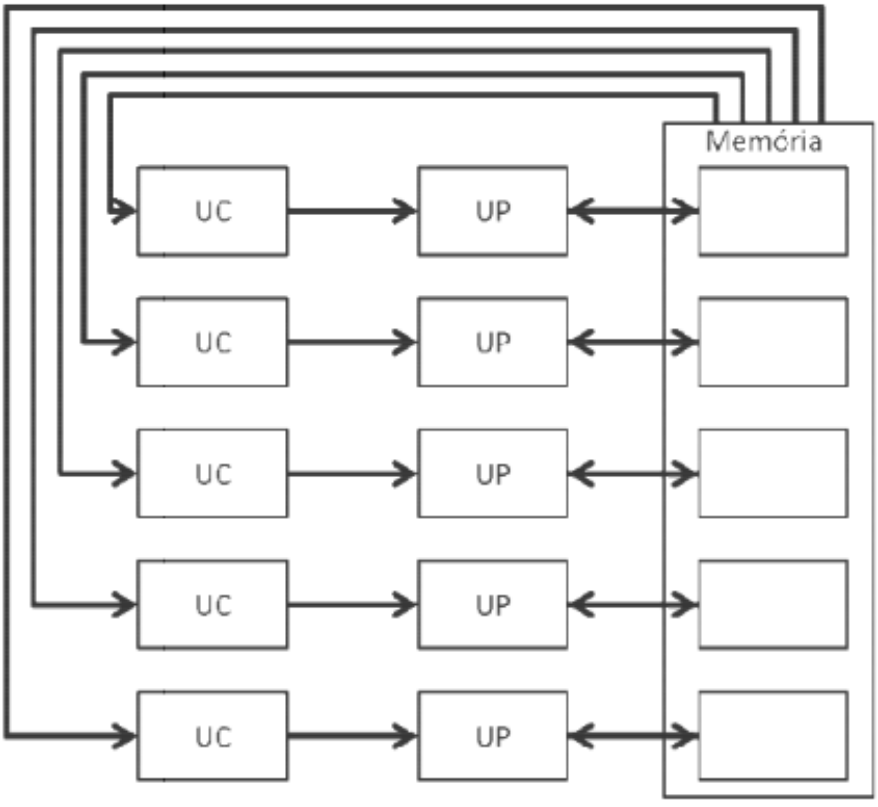

Figura 12 – Modelo arquitetura MIMD