O AVX permite operações com vetores de 256 bits, processando até 8 floats ou 4 doubles simultaneamente, ideal para cálculos repetitivos (SIMD). Já o AVX2 expande essa capacidade para suportar operações inteiras e optimiza acessos à memória com operações como gather, podendo dobrar a velocidade em situações que envolvam inteiros. O AVX/AVX2 oferece ganhos em cálculos vetoriais (SIMD), processando múltiplos dados simultaneamente, enquanto o Hyper-Threading melhora a utilização de núcleos físicos ao permitir a execução de múltiplos threads, e o CUDA utiliza milhares de threads em paralelo para tarefas massivamente paralelas. O AVX/AVX2 é ideal para operações vetoriais densas, o Hyper-Threading melhora o throughput geral, e o CUDA maximiza o desempenho em GPUs para workloads altamente paralelos.

O Hyper-Threading permite que um único núcleo físico funcione como dois núcleos lógicos, aumentando o paralelismo e a utilização da CPU. As principais vantagens incluem melhor uso dos recursos disponíveis e aumento de desempenho em tarefas que podem ser divididas em threads leves, aproveitando melhor o tempo ocioso do núcleo.

O CUDA é utilizado pelo seu paralelismo massivo, permitindo a execução de milhares de threads em paralelo, ideal para problemas que exigem alto poder computacional. No projeto, foi aplicado para acelerar a busca por DETI coins, com threads CUDA a computar hashes MD5 em paralelo.

SIMD (Single Instruction, Multiple Data) permite processar múltiplos dados com uma única instrução, acelerando cálculos repetitivos. Já o OpenMP adiciona paralelismo em nível de threads, distribuindo o trabalho entre múltiplos núcleos da CPU. A combinação é vantajosa porque o SIMD optimiza operações dentro de cada thread, enquanto o OpenMP divide o trabalho entre threads, maximizando o desempenho.

Os endereços virtuais são transformados em endereços físicos para que o hardware aceda à memória de forma eficiente e segura. Isso traz benefícios como: isolamento de processos, garantindo que cada processo tenha o seu próprio espaço de memória virtual e não aceda a dados de outros; partilha de memória, permitindo que processos partilhem bibliotecas dinâmicas e reduzam redundâncias; e protecção de memória, com permissões como NX (No-Execute) e Read-Only para evitar execução ou alteração indevida. Além disso, o TLB (Translation Lookaside Buffer) acelera essa tradução, armazenando mapeamentos recentes e reduzindo a latência de acessos frequentes.

A organização da cache impacta directamente no desempenho, especialmente em tarefas como multiplicação de matrizes. Caches podem ser directamente mapeadas, associativas por conjunto ou totalmente associativas. Baixa associatividade pode causar "cache misses" frequentes, como no acesso vertical à matriz B (saltos de 8 kB).
Na multiplicação de matrizes $C=A\times B$, o acesso horizontal à matriz A é eficiente para a cache, mas o acesso vertical à matriz B gera problemas. Uma solução é processar blocos de 8 doubles (64 bytes) usando SIMD, o que melhora a localidade espacial e reutiliza os valores de A[i][k] várias vezes. Comparado ao código escalar, o código optimizado com

SIMD processa 8 colunas simultaneamente, reduz misses e aproveita melhor os registos vetoriais, resultando em maior desempenho e eficiência.

Prefetch é uma técnica em que o processador antecipa acessos a dados ou instruções, carregando-os na cache antes de serem necessários. Baseia-se em padrões de acesso anteriores para prever os próximos dados a serem usados. As principais vantagens incluem a redução da latência, evitando esperas para carregar dados, e a melhor utilização da cache, garantindo que os dados estejam disponíveis imediatamente para a CPU.

Uma arquitectura superescalar permite executar múltiplas instruções por ciclo de clock, utilizando várias unidades de execução no pipeline. Ela realiza prefetch de várias instruções, processa diferentes tipos de operações (aritméticas, lógicas, de memória) simultaneamente e utiliza execução fora de ordem para evitar bloqueios.
As vantagens incluem maior desempenho, com aumento do IPC (instruções por ciclo); melhor aproveitamento de recursos, reduzindo o tempo ocioso do pipeline; menor impacto de dependências entre instruções, graças à reordenação; e prefetch eficiente, diminuindo a espera por novas instruções.

Out-of-Order Execution (OOE) permite que instruções independentes sejam executadas fora de ordem, optimizando o uso do pipeline e reduzindo atrasos causados por dependências. Instruções rápidas podem ser processadas enquanto outras aguardam a resolução de dependências ou acesso à memória.
Execução Especulativa complementa o OOE ao prever o caminho mais provável de execução (branch prediction), preenchendo o pipeline com instruções antecipadamente. Se a previsão falhar, o trabalho especulativo é descartado. Vantagens: melhor utilização do pipeline, minimizando atrasos; maior desempenho, com mais instruções processadas por ciclo (IPC); e o paralelismo implícito, extraindo eficiência de código sequencial sem modificações.

SIMD (Single Instruction, Multiple Data) é uma técnica de paralelismo que aplica uma única instrução a múltiplos dados simultaneamente, ideal para operações vetoriais e matriciais. Com paralelismo em nível de dados, uma única instrução controla várias unidades de processamento, permitindo operações como somar dois vetores de uma só vez. Processadores modernos utilizam registos vetoriais, como SSE e AVX (Intel/AMD) ou NEON (ARM), para armazenar múltiplos valores em uma única estrutura. As principais aplicações incluem gráficos 3D, processamento de áudio e vídeo, machine learning, inteligência artificial e simulações científicas, acelerando operações repetitivas em grandes volumes de dados.

SMT (Hyper-Threading) permite que um único núcleo físico execute múltiplos threads simultaneamente, melhorando o uso dos recursos do núcleo e o desempenho em tarefas paralelas. As threads partilham caches, o que pode causar contenção e reduzir a eficiência, mas aproveitam unidades de execução ociosas, aumentando a utilização. O pipeline é partilhado, mas com registos duplicados, permitindo alternância rápida entre threads sem custos adicionais significativos.

Many cores são processadores com vários núcleos físicos que executam tarefas em paralelo. O OpenMP é uma API que facilita a programação paralela nesses sistemas, com directivas como #pragma omp parallel para criar threads, #pragma omp critical para garantir exclusão mútua em secções críticas, e #pragma omp reduce para operações acumulativas paralelas, como somas e produtos.

CUDA utiliza warps, grupos de 32 threads que executam simultaneamente seguindo o modelo SIMD, aplicando a mesma instrução a dados diferentes. Em acessos à memória, threads consecutivas acedendo a endereços alinhados proporcionam acessos coalescidos e maior eficiência. Já acessos desalinhados, com endereços não consecutivos, reduzem a eficiência por falta de coalescência. Além disso, a divergência de fluxos ocorre quando threads em um warp seguem caminhos diferentes, como em branches if/else, obrigando o warp a processar cada caminho separadamente e causando perda de desempenho. Para mitigar, é importante minimizar branches divergentes e estruturar o código de forma uniforme.

Implementação de um Barrel Shifter

O barrel shifter é um circuito usado para realizar deslocamentos lógicos de múltiplos bits em um único ciclo. Ele funciona em níveis, onde cada nível realiza um deslocamento específico (1, 2, 4 bits, etc.), controlado por sinais de selecção (sel). Se sel = 0, o valor permanece inalterado; se sel = 1, o deslocamento é aplicado. A sua principal vantagem é permitir deslocamentos rápidos e flexíveis para diferentes tamanhos de dados. No trabalho, foi usado no acumulador para deslocar o incremento (inc) pelo valor de shift, melhorando a eficiência do circuito.

As arquiteturas modernas de processamento incluem diversos elementos-chave projetados para maximizar a eficiência e o desempenho. A cache é uma memória rápida de pequena capacidade que armazena dados frequentemente acessados, reduzindo a latência ao evitar acessos frequentes à memória principal. A técnica SIMD (Single Instruction, Multiple Data) permite que uma única instrução seja aplicada simultaneamente a múltiplos dados, aumentando o paralelismo e otimizando operações matemáticas e vetoriais. O multithreading possibilita que múltiplas threads sejam executadas simultaneamente em um mesmo núcleo, melhorando a utilização dos recursos do processador. Além disso, o uso de many cores incorpora múltiplos núcleos em um único processador, permitindo que tarefas sejam divididas e processadas em paralelo, acelerando significativamente as operações.

Já as GPUs se destacam como processadores especializados em cálculos paralelos massivos. Projetadas inicialmente para processamento gráfico, as GPUs evoluíram para desempenhar papéis cruciais em aplicações de alto desempenho, como simulações científicas, inteligência artificial e mineração de criptomoedas. Nesse contexto, surge o CUDA (Compute Unified Device Architecture), uma plataforma de computação paralela desenvolvida pela NVIDIA. O CUDA permite programar GPUs para executar milhares de threads simultaneamente, oferecendo um desempenho excepcional em tarefas que exigem paralelismo massivo. A sua eficiência é ampliada pelo uso de técnicas como coalescência de memória, que organiza acessos consecutivos para maximizar a largura de banda, e pela capacidade de gerenciar divergências entre threads em grupos (warps), garantindo o máximo aproveitamento da arquitetura.

O AVX (vetores de 256 bits) processa até 8 floats ou 4 doubles simultaneamente, otimizando cálculos vetoriais. O AVX2 expande para operações inteiras e melhora acessos à memória (ex.: gather), dobrando o desempenho em tarefas com inteiros. Ambos são ideais para cálculos densos e repetitivos no modelo SIMD.

O Hyper-Threading permite que um núcleo físico funcione como dois lógicos, aumentando o paralelismo e a utilização da CPU. Melhora o desempenho em tarefas multithreaded, aproveitando o tempo ocioso do núcleo.

O CUDA habilita paralelismo massivo, com milhares de threads em paralelo, ideal para tarefas altamente computacionais. Foi aplicado no projeto para acelerar buscas por DETI coins, usando threads para calcular hashes MD5.

SIMD acelera cálculos repetitivos processando múltiplos dados com uma única instrução, enquanto o OpenMP distribui trabalho entre múltiplos núcleos, combinando paralelismo em nível de dados e threads para maximizar desempenho.

O uso de endereços virtuais garante isolamento, partilha de memória e proteção de dados. O TLB (Translation Lookaside Buffer) acelera a conversão de endereços virtuais para físicos, reduzindo latências e otimizando acessos frequentes à memória.

A cache é uma memória hierárquica organizada em diferentes níveis (L1, L2, L3), que oferece tamanhos e velocidades variados. A organização em linhas permite o carregamento eficiente de dados em blocos de 64 bytes. Por exemplo, a cache L1 é extremamente rápida, mas limitada em tamanho (~32 KB), enquanto a L3 é maior (~12 MB), mas mais lenta. A organização da cache utiliza estratégias de associatividade, como totalmente associativo ou associativo por conjunto, para armazenar e localizar dados. Quando os dados necessários não estão na cache, ocorrem os chamados "cache misses", que impactam negativamente o desempenho. Um exemplo prático de otimização é a multiplicação de matrizes $C=A\times B$, onde técnicas como o blocking dividem as matrizes em blocos que cabem na cache, reduzindo os acessos à memória principal.

O prefetch é outra técnica que antecipa acessos à memória, carregando dados antes de serem necessários. Isso é particularmente útil em operações que percorrem grandes arrays de forma sequencial, reduzindo a latência associada aos acessos à memória principal.

A execução especulativa e a execução fora de ordem (out-of-order) são técnicas avançadas que aumentam a utilização do pipeline do processador. Na execução especulativa, instruções são executadas com base em previsões, sendo descartadas caso estejam erradas. Já a execução fora de ordem permite que instruções sejam reorganizadas para evitar tempos de espera e aproveitar melhor os recursos disponíveis.

O SIMD (Single Instruction, Multiple Data) é uma técnica que permite a execução de uma única instrução em vários dados simultaneamente, sendo muito utilizada em cálculos paralelos envolvendo vetores ou matrizes. Extensões como AVX, que operam em vetores de 256 bits, podem processar oito valores float ou quatro valores double simultaneamente, melhorando significativamente o desempenho em operações matemáticas e científicas.

O multithreading é uma abordagem que divide uma aplicação em múltiplas threads, permitindo que sejam executadas ao mesmo tempo. Com o SMT (Simultaneous Multithreading), conhecido como Hyper-Threading nos processadores Intel, um núcleo pode executar múltiplas threads simultaneamente, otimizando a eficiência. Um exemplo prático seria comparar o desempenho de um processador com e sem Hyper-Threading ativado.

Os processadores com muitos núcleos (many cores) possibilitam a execução paralela de diferentes partes de uma tarefa. A ferramenta OpenMP simplifica a programação paralela, utilizando diretivas que permitem dividir loops e acelerar cálculos em CPUs com muitos núcleos.

O CUDA, uma plataforma de computação paralela desenvolvida pela NVIDIA, facilita a programação de GPUs, organizando threads em warps de 32 threads para processamento eficiente. Técnicas como a coalescência de memória, que garante acessos consecutivos por threads de um mesmo warp, e a minimização de branches divergentes, onde threads

seguem diferentes caminhos, otimizam o uso da memória e do paralelismo. As GPUs são ideais para tarefas com paralelismo massivo, como simulações ou processamento gráfico.

Finalmente, o barrel shifter é um hardware que possibilita deslocar bits rapidamente, sendo útil em operações binárias, como multiplicações ou divisões por potências de 2. Em algumas arquiteturas, essa funcionalidade foi implementada para otimização e foi testada em cenários práticos envolvendo operações de deslocamento.