O que se ganha com AVX e AVX2?
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 otimiza 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, Hyper-Threading melhora o throughput geral, e 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 computando hashes MD5 em paralelo. Comparado à CPU, apresenta desempenho superior, como no exemplo da GTX 1660 Ti, que processa cerca de 5.6 x 10^13 hashes por hora.

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 otimiza 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 acesse a memória de forma eficiente e segura. Isso traz benefícios como: isolamento de processos, garantindo que cada processo tenha seu próprio espaço de memória virtual e não acesse dados de outros; partilha de memória, permitindo que processos compartilhem bibliotecas dinâmicas e reduzam redundâncias; e proteção 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 diretamente no desempenho, especialmente em tarefas como multiplicação de matrizes. Caches podem ser diretamente 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×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 otimizado com SIMD processa 8 colunas simultaneamente, reduz misses e aproveita melhor os registros 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 arquitetura 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, otimizando 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 registros 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 compartilham caches, o que pode causar contenção e reduzir a

eficiência, mas aproveitam unidades de execução ociosas, aumentando a utilização. O pipeline é compartilhado, 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 diretivas como #pragma omp parallel para criar threads, #pragma omp critical para garantir exclusão mútua em seções 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 acessando 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 seleção (sel). Se sel = 0, o valor permanece inalterado; se sel = 1, o deslocamento é aplicado. 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.