- está relacionado à programação em GPUs
- a eficiência de uma arquitetura depende do padrão de processamento

TENDÊNCIA:
- CPU convencionais + aceleradores (sistema heterogêneo)
- computadores / aceleradores de propósitos específico (ex: deep learning)
- o hardware está evoluindo muito mais rápido que a capacidade de produzir softwares que se adequem a essas arquiteturas

Modelos de programação importantes nessa área: MPI, OpenMP, OpenCL

Conclusão: o futuro será caracterizado pela diversidade de fabricantes e arquiteturas no mundo de supercomputadores!

OpenCL - Open Computing Language: padrão aberto para a programação paralela de sistemas heterogêneos
CENÁRIOS: CPUs cada vez mais paralelas e versáteis

CARACTERÍSTICAS:
- provê uma interface homogênea para a exploração da computação paralela heterogênea (abstração do hardware)
- padrão aberto (gerenciada pelo grupo Kronos)
- alto desempenho
- multi-plataforma
- código portável entre arquitetura e gerações
- paralelismo de dados (SIMD) e de tarefas (MIMD)
- baseado em C e C++
- define garantias para operações em ponto flutuante
- integração com outras tecnologias (OpenGL)

OpenCL 2.0
- memória virtual compartilhada
- paralelismo dinâmico
- objetos pipe
- diretivas atômicas

OBS: OpenCL e CUDA têm vários pontos de intersecção

OpenCL x MPI: tecnologias ortogonais, porém, podem ser combinadas

POSSÍVEL CENÁRIO FUTURO: convergência para às abordagens MPI (paralelismo distribuído) e OpenMP (paralelismo incremental / fácil)

Comentado [1]: altíssimo poder computacional, preço, energia, etc...

Comentado [2]: foi uma proposta da Apple e atualmente está na versão 2.2

Comentado [3]: explora todos os dispositivos computacionais do nó (ou do computador), enquanto o CUDA foca na GPU

Comentado [4]: paralelismo local, memória compartilhada

Comentado [5]: paralelismo distribuído, memória distribuída

GPUs: têm uma latência muito grande em acessar um dado, porém, a vazão dos dados é enorme (o que mascara a latência de acesso).

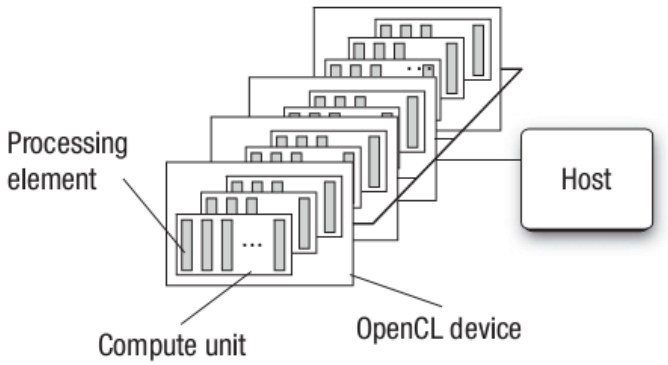
PERFIL ÓTIMO DE CARGA EM GPUs

- milhares de threads independentes
- minimiza desvio de fluxo (baixa ramificação)
- possui alta intensidade aritmética

INTRODUÇÃO EM OpenCL

Existem duas hierarquias de códigos no OpenCL:

- o kernel
- o hospedeiro

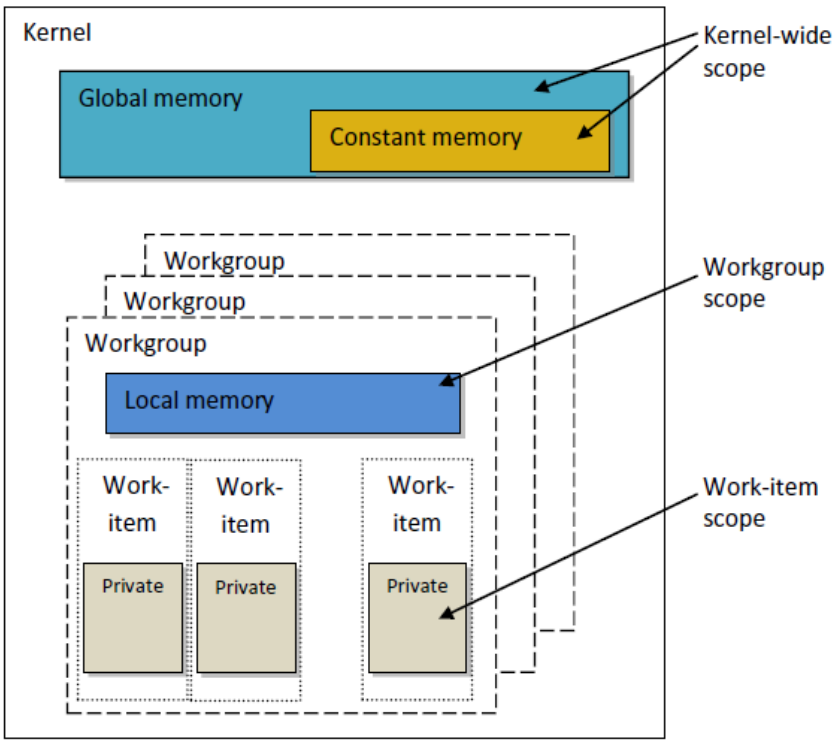


MODELO DE EXECUÇÃO:

- item de trabalho: uma instância kernel em execução
- grupo de trabalho: agrupamento dos itens de trabalho (comunicação ou sincronização)

obs: acesso coalescente à memória (muito importante em qualquer linguagem de programação)

Comentado [6]: ver mais em: https://www.intel.com.br/content/www/br/pt/support/articles/000007456/network-and-i-o/ethernet-products.html



PASSOS DE EXECUÇÃO:

Inicialização:

- descobrir e escolher as plataformas e dispositivos
- criar o contexto de execução
- criar a fila de comandos para um dispositivo
- carregar o programa, compilar e gerar o kernel

Preparação da memória

Execução