GPU Donanım Notları

Cpu ve gpu arasındaki farkı anlamak için cpu'da çekirdek ve thread(iş parçacıkları)'na bakalım

CPU Tarafı

CPU'da işlem şu şekilde işler:

- **İş Parçacıkları (Threads)**: CPU'da programlar iş parçacıkları (threads) aracılığıyla çalıştırılır. Her iş parçacığı bağımsız bir hesaplama birimidir.
- **Çekirdekler (Cores)**: CPU, sınırlı sayıda çekirdeğe sahiptir (örneğin 4, 8 veya 16 çekirdek). Her çekirdek aynı anda yalnızca bir iş parçacığı çalıştırabilir.
- **İş Parçacığı Zamanlayıcıları**: Çok iş parçacıklı uygulamalarda, iş parçacıkları çekirdeklere gönderilir ve çekirdekler bu iş parçacıklarını sırayla işler. Zamanlayıcılar, çekirdeklerin iş parçacıkları arasında geçiş yapmasını sağlar, çünkü çekirdek sayısı iş parçacığı sayısından genellikle daha azdır.
- CPU iş yükü bu nedenle iş parçacıkları arasında zamanlanır ve sınırlı paralellik sunar.

Aslında temel işleyiş thread'leri core'lara yolluyoruz ve thread görevleri yapılıyor. Tek çekirdekli bir işlemcide core schedule(thread zamanlayıcısı gibi) denen zımbırtı sanki paralel iş yapıyormuş gibi thread'lerdeki görevler arasında geçişler yapar. Örneğin görev1 başlar, 10ms geçer ve kod kesilir(interrupt handler çağrılır) görev2'ye geçer 10ms sonra görev1'e döner. Bu esnada görev1 verileri "context switch memory"'de depolanır ve geçiş yapıldığında buradan datalar çekilir. Bu sayede tek cpu'da "yanıltmacalı(bu terimi ben salladım) paralel processing" uygulanmış olur.

GPU ve CUDA Tarafı: Warplar, Çekirdekler ve Paralel Hesaplama

GPU'lar çok daha geniş bir paralel hesaplama yapısı sunar ve bu yapı CUDA ile kontrol edilir. İş parçacığı yönetimi CPU'ya göre farklıdır:

Streaming Multiprocessor (SM) ve CUDA Çekirdekleri

- **SM (Streaming Multiprocessor)**: GPU'daki temel işlem birimidir. Her SM, kendi içinde birçok CUDA çekirdeği(Stream Processor) barındırır. CUDA çekirdekleri, CPU'daki çekirdekler gibi düşünülebilir ancak çok daha basit ve hafiftirler(düşük saat hızı ve küçük önbellek).
- **CUDA Çekirdekleri**: GPU'nun SM'leri içindeki işlem birimleridir. Her CUDA çekirdeği, bir iş parçacığını işler, ancak bir SM'de yüzlerce CUDA çekirdeği olabilir.

Warp ve İş Parçacıkları (Threads)

- **Warp**: CUDA iş parçacıklarını (thread) organize eden temel birimdir. Bir warp, **32 iş parçacığından** oluşur. Bu iş parçacıkları GPU'da aynı anda çalıştırılır.
  - GPU, **warp düzeyinde** komutları işler. Yani bir SM, 32 iş parçacığını (1 warp) aynı anda çalıştırır.
  - **SIMD (Single Instruction, Multiple Data)**: Bir warp'taki tüm iş parçacıkları aynı komutu yürütür, ancak farklı veriler üzerinde çalışabilir. Bu, SIMD (tek komut, çoklu veri) işleme modelidir.

CUDA ve Warp Yönetimi

- **Bloklar ve Grid'ler**: CUDA programlamasında iş parçacıkları "blok" adı verilen gruplar halinde organize edilir. Bir blok, bir veya birden fazla warp'tan oluşur. Bloklar ise "grid" adı verilen daha büyük yapılar oluşturur. Yani:
  - **Grid**: Bloklardan oluşan daha büyük bir yapı.
  - **Blok**: Warplardan oluşan yapı.
  - **Warp**: 32 iş parçacığından oluşan temel birim.
- **SM Yönetimi**: Bir SM, aynı anda birden fazla warp'ı işleyebilir. Bir SM, bir grid içindeki bir veya daha fazla bloğa atanır ve bu blokların içerdiği warp'ları işler. Her warp'taki iş parçacıkları paralel çalışır.

Warp ve Divergence (Ayrışma)

- Warplar, aynı anda komut yürütse de, iş parçacıkları farklı dallara ayrılırsa ("if" yapılarında olduğu gibi), warp divergence adı verilen bir problem ortaya çıkar.
    - Eğer bir warp'taki bazı iş parçacıkları farklı bir yolda ilerlerse (örneğin bir "if-else" yapısında), bu iş parçacıkları sırayla yürütülür ve paralel çalışmanın verimliliği düşer.

CPU ve CUDA Arasındaki Farklar

- **İş Parçacığı Yönetimi**: CPU'da her çekirdek bağımsız bir iş parçacığı çalıştırır ve çekirdek sayısı sınırlıdır. GPU'da ise çok sayıda iş parçacığı bir SM içinde paralel olarak çalıştırılır.
- **Paralellik Düzeyi**: CPU genellikle daha sınırlı sayıda iş parçacığını paralel çalıştırabilirken, GPU yüzbinlerce iş parçacığını aynı anda çalıştırabilir.
- **Warp Kavramı**: CPU'da warp gibi bir kavram yoktur. GPU'da ise bir warp, 32 iş parçacığının birlikte çalıştığı en küçük paralel çalışma birimidir.

CUDA Donanımı Özet

1. **SM (Streaming Multiprocessor)**: GPU'daki temel işleme birimi, birden çok warp çalıştırabilir.
2. **Warp**: 32 iş parçacığından oluşan ve aynı anda çalıştırılan temel hesaplama birimi.
3. **CUDA Çekirdekleri**: SM'ler içindeki iş parçacıklarını çalıştıran basit işlem birimleri.

SM ve warp'ların donanım düzeyinde çalışma şekli

(Dikkat! Bu konu biraz beyin yakar, Türkçem yettiğince bildiklerimi yazıya döküyorum...)

(İkinci dikkat! Bu bölümde anlatacağım donanımsal bilgiye örnek verirken, kendi kullandığım kart olan GeForce RTX 2060 (12 GB)'ı baz alalım.)

Şimdi warp'lar SM'lerde işleniyor demiştik. Kartımızda 34 adet SM bulunuyor(rtx serisinin wiki sayfasından donanım özelliklerini bulabiliyoruz). Kartımız Turing mimarisi olduğundan her SM'de 64 adet cuda core bulunuyor. O halde 34x64 = 2176 adet cuda core mevcut. Bunu wikiden de doğrulayabiliriz. 1 warp 32 thread'den oluşur. Turing mimarisinde bir SM, aynı anda 2,048 thread'i **çalıştırabilir**. (Çalıştırma kelimesine döneceğiz)

Şimdi kart ile ilgili teknik bilgileri aldığımıza göre süreç nasıl işliyor ona bakalım. CPU'da olduğu gibi 1 core aynı anda 1 thread işleyebilir. GPU'da thread'ler warp'lar halinde işlenir demiştik. SM içerisinde 64 core bulunduğuna göre, 1 SM aynı anda 2 warp işleyebilir. O halde 34 SM 64 warp işler ve 64x32(1warp'taki thread) = 2176. Görüldüğü gibi core sayımızı veriyor.

Şimdi warp bazında işlemeye geçelim. 32 core'dan oluşan bir grup (SM içinde), **aynı anda** bir warp'taki 32 thread'i işler ve bu thread'ler **aynı komutu** uygular (Buna Single Instruction, Multiple Threads - SIMT denir.) Aynı komutu uygulamasının nedeni latency hiding yaşanmaması, yani 1 thread'in komutu 1 sn sürüyorsa ve diğeri 2 sn sürüyorsa 1sn süren core diğerinin bitmesini bekler. Tabii donanımsal ve yazılımsal kolaylıklarda söz konusu. O kadar detay istiyorsal Nvidia'da çalışmayı düşünebiliriz...
Örneğin 32 adet değişkenimiz var ve bunların her birine +2 eklemek istiyoruz. SM bu +2 ekleme talimatını aynı anda 32 değişkene(1 warp)'de uygular.
Eğer 28 değişkenimiz varsa ve aynı işlemi yapmak istersek warp sayısı sabit 32 thread olduğundan 4 thread boşta kalacaktır. Bunun performansa zararı yoktur ancak fiziksel olarak o 4 thread diğer 28 thread'i bekler (aynı hayat gibi :-}).
Eğer 36 değişkenimiz varsa 2. warp'a geçilir ve 4 thread 2. warp'ta çalıştırılır.
Peki 32 değişkenimiz var ancak değişkenlerin yarısı tek yarısı çift ve çiftlere +2, teklere +1 uygulamak istiyoruz diyelim. Böyle bir durumda **aynı komutu** uygulayamayız çünkü araya bir if koşulu giriyor. Değişkenlerimizin yarısına +2 eklenirken yarısına +1 eklenmesi lazım. Bu durumda önce bir yarısı sonra diğer yarısı işlenerek 2 döngüde işlemi bitiririz. Buna **Divergence** denir ve istenmeyen bir durumdur. Cuda'da döngülerden ve koşullu ifadelerden kaçınmamız gerekir.

Şimdi **çalıştırma** ve **işleme** kelimelerine bakalım:

Çalıştırmak ve İşlemek Arasındaki Fark:

1. **Çalıştırmak (Scheduling)**: Bir SM, belirli bir anda birçok thread'i "çalıştırabilir", yani bu thread'ler aktif olabilir ve belleğe, register'lara erişebilir. Turing mimarisinde bir SM, aynı anda 2048 thread'i **çalıştırabilir** (veya "barındırabilir"). 34 SM olduğuna göre 34x2048 = 69,632 adet thread aynı anda aktif olabilir. Ancak aynı anda işlenmez, belleğe erişebilir, işlenmek için sıraya konulur vs.
2. **İşlemek (Execution)**: İşlemek ise fiziksel CUDA core'larda **aynı anda işlem gören** thread'ler anlamına gelir. Bu noktada, SM'deki **fiziksel core** sayısı devreye girer. Bir warp'taki 32 thread, aynı anda 32 CUDA core'da işlenir. Yani tek saat döngüsünde karttaki maksimum çekirdekten fazla thread **işlenemez**.

Bir dipnot: Cuda mimarisinde **Scheduling** işlemi çok optimedir. Eğer bellekten veri bekleyen bir warp varsa hazırda olan warpı araya sokup işleme kabiliyeti bile vardır. Ama CPU'da olan **dallanma tahmini(Branch prediction)** GPU'da mevcut değildir.

Olay bundan ibaret. Sonuç olarak adamlar deli gibi donanım geliştiriyorlar. Helal olsun...