DYNAMIC PARALLELISM IN CUDA

Dynamic Parallelism in CUDA is supported via an extension to the CUDA programming model that enables a CUDA kernel to create and synchronize new nested work. Basically, a child CUDA Kernel can be called from within a parent CUDA kernel and then optionally synchronize on the completion of that child CUDA Kernel. The parent CUDA kernel can consume the output produced from the child CUDA Kernel, all without CPU involvement.

Example:

```
__global__ ChildKernel(void* data){
    //Operate on data
}

__global__ ParentKernel(void *data){
    ChildKernel<<<16, 1>>>(data);
}

// In Host Code
ParentKernel<<<256, 64>>(data);
```

Recursion is also supported, and a kernel may call itself:

```
__global__ RecursiveKernel(void* data){
    if(continueRecursion == true)
        RecursiveKernel<<<64, 16>>>(data);
}
```

The language interface and Device Runtime API available in CUDA C/C++ is a subset of the CUDA Runtime API available on the Host. The syntax and semantics of the CUDA Runtime API have been retained on the device in order to facilitate ease of code reuse for routines that may run in either the Host or Dynamic Parallelism environments.

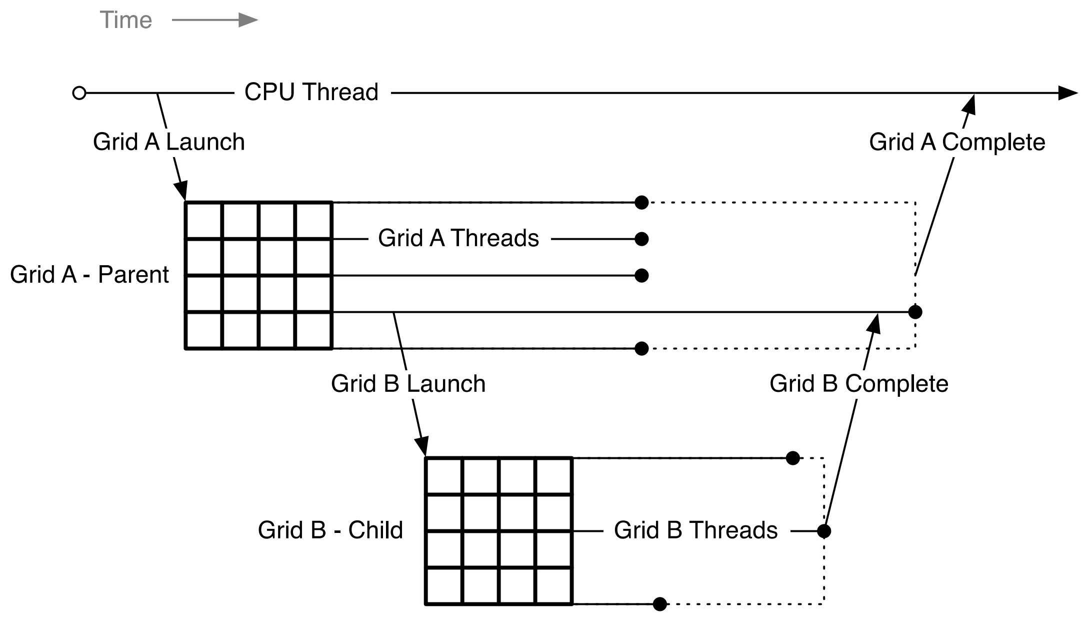
Important benefits when new work is invoked within an executing GPU program include removing the burden on the programmer to marshal and transfer the data on which to operate. Additional parallelism can be exposed to the GPU's hardware schedulers and load balancers dynamically, adapting in response to data-driven decisions or workloads. Algorithms and programming patterns that had previously required modifications to eliminate recursion, irregular loop structure, or other constructs that do not fit a flat, single-level of parallelism can be more transparently expressed.

The CUDA execution model is based on primitives of threads, thread blocks, and grids, with kernel functions defining the operation of individual threads within a thread block and grid. When a Kernel Function is invoked, the grid's properties are described by an execution configuration, which has a special syntax in CUDA C. Dynamic parallelism support in CUDA extends the ability to configure and launch grids, as well as wait for the completion of grids, to threads that are themselves already running within a grid.

Parent and Child Grids

A thread that is part of an executing grid and which configures and launches a new grid belongs to the Parent Grid and the grid created by the invocation is the Child Grid.

Parent-Child Launch Nesting



The invocation and completion of Child Grids is properly nested, meaning that the Parent Grid is not considered complete until all Child Grids created by its threads have completed. Even if the invoking threads do not explicitly synchronize on the Child Grids launched, the runtime guarantees an implicit synchronization between the Parent and Child.