3.6.2 Chamada ou disparo do kernel

CUDA estende a sintaxe de chamada de função do C com parâmetros de configuração de execução do kernel cercados por <<< e >>>. Esses parâmetros de configuração de execução só são usados durante uma chamada a uma função de kernel, ou um disparo do kernel. Discutimos sobre os parâmetros de configuração de execução que definem as dimensões da grade e as dimensões de cada bloco. O leitor deverá consultar o *CUDA Programming Guide* [NVIDIA 2007] para obter mais detalhes com relação às extensões de disparo do kernel, bem como outros tipos de parâmetros de configuração de execução.

3.6.3 Variáveis predefinidas

Os kernels CUDA podem acessar um conjunto de variáveis predefinidas que permitem que cada *thread* tenham uma distinção entre si e para determinar a área de dados em que cada *thread* deve atuar. Discutimos a variável `threadIdx` neste capítulo. No Capítulo 4, discutiremos melhor as variáveis `blockIdx`, `gridDim` e `blockDim`.[5]

3.6.4 API de runtime

CUDA contém um conjunto de funções de API para fornecer serviços aos programas CUDA. Os serviços que discutimos neste capítulo são as funções `cudaMalloc()` e `cudaMemcpy()`, que alocam memória de device e transferem dados entre o host e o device em favor do programa que está chamando. O leitor deverá consultar o *CUDA Programming Guide* [NVIDIA 2007] para procurar outras funções da API CUDA.

Nosso objetivo neste capítulo é apresentar os conceitos fundamentais do modelo de programação CUDA e as extensões CUDA à linguagem C essenciais para a escrita de um programa CUDA simples. O capítulo de forma alguma é um relato abrangente de todos os recursos do CUDA. Alguns desses recursos serão abordados no restante do livro; contudo, nossa ênfase será nos conceitos, não nos detalhes. Em geral, gostaríamos de encorajar o leitor a sempre consultar o *CUDA Programming Guide* para obter mais detalhes sobre os conceitos que abordamos.

Referências e leitura adicional

ATALLAH, M. J. (Ed.), (1998). *Algorithms and Theory of Computation Handbook.* Boca Raton, FL: CRC Press.

NVIDIA. (2009). *CUDA programming guide 2.3.* Santa Clara, CA: NVIDIA.

STRATTON, J. A., STONE, S. S. & HWU, W. W. (2008). MCUDA: An Efficient Implementation of CUDA Kernels for Multinúcleo CPUs. Em *Proceedings of the 21st International Workshop on Languages and Compilers for Parallel Computing (LCPC).* Canadá: Edmonton.

[5] Observe que as variáveis gridDim e blockDim são variáveis internas, pré-definidas, que são acessíveis nas funções de kernel. Elas não devem ser confundidas com as variáveis definidas pelo usuário dimGrid e dimBlock, que são usadas no código host com a finalidade de estabelecer os parâmetros de configuração. Os valores desses parâmetros de configuração por fim se tornarão os valores de gridDim e blockDim quando o kernel tiver sido disparado.

3.6.2 Chamada ou disparo do kernel

CUDA estende a sintaxe de chamada de função do C com parâmetros de configuração de execução do kernel cercados por <<< e >>>. Esses parâmetros de configuração de execução só são usados durante uma chamada a uma função de kernel, ou um disparo do kernel. Discutimos sobre os parâmetros de configuração de execução que definem as dimensões da grade e as dimensões de cada bloco. O leitor deverá consultar o *CUDA Programming Guide* [NVIDIA 2007] para obter mais detalhes com relação às extensões de disparo do kernel, bem como outros tipos de parâmetros de configuração de execução.

3.6.3 Variáveis predefinidas

Os kernels CUDA podem acessar um conjunto de variáveis predefinidas que permitem que cada *thread* tenham uma distinção entre si e para determinar a área de dados em que cada *thread* deve atuar. Discutimos a variável `threadIdx` neste capítulo. No Capítulo 4, discutiremos melhor as variáveis `blockIdx`, `gridDim` e `blockDim`.[5]

3.6.4 API de runtime

CUDA contém um conjunto de funções de API para fornecer serviços aos programas CUDA. Os serviços que discutimos neste capítulo são as funções `cudaMalloc()` e `cudaMemcpy()`, que alocam memória de device e transferem dados entre o host e o device em favor do programa que está chamando. O leitor deverá consultar o *CUDA Programming Guide* [NVIDIA 2007] para procurar outras funções da API CUDA.

Nosso objetivo neste capítulo é apresentar os conceitos fundamentais do modelo de programação CUDA e as extensões CUDA à linguagem C essenciais para a escrita de um programa CUDA simples. O capítulo de forma alguma é um relato abrangente de todos os recursos do CUDA. Alguns desses recursos serão abordados no restante do livro; contudo, nossa ênfase será nos conceitos, não nos detalhes. Em geral, gostaríamos de encorajar o leitor a sempre consultar o *CUDA Programming Guide* para obter mais detalhes sobre os conceitos que abordamos.

Referências e leitura adicional

ATALLAH, M. J. (Ed.), (1998). *Algorithms and Theory of Computation Handbook*. Boca Raton, FL: CRC Press.

NVIDIA. (2009). *CUDA programming guide 2.3*. Santa Clara, CA: NVIDIA.

STRATTON, J. A., STONE, S. S. & HWU, W. W. (2008). MCUDA: An Efficient Implementation of CUDA Kernels for Multinúcleo CPUs. Em *Proceedings of the 21st International Workshop on Languages and Compilers for Parallel Computing (LCPC)*. Canadá: Edmonton.

[5] Observe que as variáveis gridDim e blockDim são variáveis internas, pré-definidas, que são acessíveis nas funções de kernel. Elas não devem ser confundidas com as variáveis definidas pelo usuário dimGrid e dimBlock, que são usadas no código host com a finalidade de estabelecer os parâmetros de configuração. Os valores desses parâmetros de configuração por fim se tornarão os valores de gridDim e blockDim quando o kernel tiver sido disparado.