hyPACK-2013

Four-Days Technology Workshop *on*

Hybrid Computing - Coprocessors & Accelerators -
Power-Aware Computing & Performance of Application Kernels

Jointly Organized by

Centre for Development of Advanced Computing (C-DAC), Pune

Centre for Modelling & Simulation (CMSD), HPC Facility, University of Hyderabad,

Venue : CMSD, University of Hyderabad, Hyderabad

Dates : October 15 (Tuesday) – October 18 (Friday)

hyPACK-2013 : A Suite of programs on CUDA enabled NVIDIA GPUs

The list of programs are developed based on CUDA enabled NVIDIA GPU – Fermi & Kepler technology and NVML library APIs. The information of these codes are publicly available as NVIDIA Books, NVIDIA technical documents, NVIDIA GPU Conferences, Workshops, Notes Material and especially recent webinars and these have been partially incorporated.

List of Modules – CUDA enabled NVIDIA GPUs

Module 1 : Getting Started: CUDA enabled NVIDIA GPUs
Module 2 : Getting Started: PGI OpenACC APIs on CUDA enabled NVIDIA GPU)
Module 3 : CUDA enabled NVIDIA GPU Programs on Numerical Computations (Dense Matrix Computations)
Module 4 : CUDA enabled NVIDIA GPU Programs using BLAS libraries for Matrix Computations
Module 5 : CUDA enabled NVIDIA GPU Programs - Application Kernels
Module 6 : CUDA enabled NVIDIA GPU Memory Optimization Programs - Tuning & Performance
Module 7 : CUDA enabled NVIDIA GPU Streams: Concurrent Asynchronous Execution
Module 8 : CUDA enabled NVIDIA Programs using **Kepler GPU** features

1. Module 1: Getting Started: CUDA enabled NVIDIA GPU Programs

1.1. Simple Vector-Vector Addition
1.2. Vector-Vector Addition: Thread Cooperation (Splitting Parallel Blocks)
1.3. Vector-Vector Addition: Dimension of Grid – Each Block

2. Module -2: Getting Started : PGI OpenACC APIs on CUDA enabled NVIDIA GPU)

2.1. Write a CUDA Program for Matrix Vector multiplication
2.2. Write a CUDA Program for Matrix into Matrix multiplication based on tiling Partitioning
2.3. Write a CUDA Program for implement solution of matrix system of linear equations Ax=b by Jacobi method.

3. Module 3 : CUDA enabled NVIDIA GPU Programs on Numerical Computations (Dense Matrix Computations)

3.1. Write a CUDA program to compute Vector - Vector addition
3.2. Write a CUDA program to compute Matrix - Matrix addition
3.3. Write a CUDA Program to compute vector - Vector multiplication
3.4. Write a CUDA Program to find prefix sum of a given array
3.5. Write a CUDA program to find transpose of a matrix
3.6. Write a CUDA Program to calculate value of PI using numerical integration method.
3.7. Write a CUDA Program to find infinity norm of a matrix
3.8. Write a CUDA Program for Matrix Vector multiplication
3.9. Write a CUDA Program for Matrix Vector multiplication
3.10. Write a CUDA Program for Matrix into Matrix multiplication based on tiling Partitioning
3.11. Write a CUDA Program for implement solution of matrix system of linear equations Ax=b by Jacobi method.
3.12. Write a CUDA program to implement the solution of Matrix system of Linear Equations Ax=b by Conjugate Gradient method (Iterative Method).
3.13. Write a CUDA Program to compute vector - vector multiplication on Multi-GPU
3.14. Write a CUDA Program for Matrix into Matrix multiplication on Multi-GPU
3.15. Write a CUDA Program for Vector into Vector multiplication using CUBLAS libraries on Multi-GPU
3.16. Write a CUDA Program for Matrix into Matrix multiplication using CUBLAS libraries on Multi-GPU

4. Module 4 : CUDA enabled NVIDIA GPU Programs using BLAS libraries for Matrix Computations

4.1. Simple test Programs using CUBLAS1, CUBLAS2, CBLAS3 library function calls.
4.2. Write a Program for vector into vector (dot product of two vectors) multiplication using CUBLAS1 library function calls.
4.3. Write a Program for multiplication of a scalar with a vector and add the resultant vector to a vector using CUBLAS1 library function calls.
4.4. Write a Program for Matrix Vector multiplication using CUBLAS2 library function calls.
4.5. Write a Program for Matrix by Matrix multiplication using CUBLAS3 library function calls.
4.6. Write a CUBLAS CUDA Program for implement solution of matrix system of linear equations Ax = b by Jacobi method.
4.7. Write a CUBLAS CUDA program to implement the solution of Matrix system of Linear Equations AX=b by Conjugate Gradient method (Iterative Method).

4.8. Write a CUBLAS CUDA program on sparse matrix multiplication of size n x n and vector of size n.(Assignment)
4.9. Write a CUDA program for matrix into matrix multiplication using vendor supplied BLAS libraries (DGEMM) on host-CPU & Device-GPU to perform computations on host-CPU & device-GPU and extract performance in Gigaflops
4.10. Write a CUDA program for matrix into matrix multiplication using vendor supplied CUDA BLAS libraries (DGEMM) and extract the performance in terms of Gflops.
4.11. Demonstrate performance of matrix and vector computations on NVIDIA GPUs using MAGMA BLAS library functions.
4.12. Demonstrate performance of data-parallel algorithm primitives such as parallel prefix-sum ("*scan*"), parallel sort and parallel reduction using CUDPP Library.
4.13. Demonstrate performance of sparse matrices computation using NVIDIA CUDA CUSPARSE library.

5. Module 5: CUDA enabled NVIDIA GPU Programs - Application Kernels

5.1. Write a CUDA Program for implementation of solution of Partial Differential Equations (PDEs) based on Finite Difference Method
5.2. Write a CUDA Program for implementation of solution of Partial Differential Equations (PDEs) based on Finite Difference Method using CUBLAS routines
5.3. Write a Program for implementation of Laplacian Edge Detection algorithm - Image Processing on multiple device-GPUs. (Assignment)
5.4. Write a CUDA Program for implementation of solution of Partial Differential Equations (PDEs) based on Finite Element Method (Assignment)
5.5. Write a CUDA Program for implementation of solution of Partial Differential Equations (PDEs) based on Finite Difference Method using multiple-GPUs (Assignment)
5.6. Write a CUDA Program for implementation of String Search (Boyer-Moore) algorithm (Assignment)

6. Module 6 : CUDA enabled NVIDIA GPU Memory Optimization Programs - Tuning & Performance

6.1. Write CUDA program to find out the number of CUDA enabled devices and the device information
6.2. Write CUDA Program to calculate achieved bandwidth for different access patterns of global memory
6.3. Write CUDA Program to use of shared memory to get coalesced data access pattern and analyze the bandwidth results.
6.4. Write CUDA Program to calculate achievable shared memory bandwidth for typical read operation by all threads.
6.5. Write CUDA Program to calculate achievable shared memory bandwidth while accessing arrays of different inbuilt data types.
6.6. Write CUDA Program to demonstrate bank conflicts that can occur while accessing the shared memory
6.7. Write CUDA Program to demonstrate advantage of Structure of arrays in terms of the bandwidth of the global memory that is achievable
6.8. Write CUDA Program to demonstrate the global memory bandwidth differences for

varying block sizes

6.9. Write CUDA Program to Bandwidth improved when thread handles more than one element by making use of GPU as a 32-way SIMD processor

6.10. Write CUDA Program to demonstrate the difference in bandwidth achieved when blocks access global memory with and without partition camping.

6.11. Write CUDA Program to demonstrate the difference in bandwidth achieved when threads within a warp follow different execution paths. Write CUDA Program to demonstrate the sustainable memory bandwidth (Stream Benchmark: Global memory of the NIVIDA GPU device).

6.12. Write a program to calculate the bandwidth of GPUs for pageable/pinned memory from Host-to-Device and Device-to-Host, Device-to-Device.

6.13. Write a program to demonstrate a strategy to hide bandwidth latency using CUDA stream APIs and concurrent execution of kernel through one stream, while memory copy of data set is also going on for the purpose of execution through other kernel and analyze the advantages in terms of the execution time taken.

7. Module 7 : CUDA enabled NVIDIA GPU Streams: Concurrent Asynchronous Execution

7.1. Multiple Kernels - Single GPU-Matrix-Matrix Addition

7.2. Multiple Kernels - Single GPU-Matrix-Vector Multiplication

7.3. Multiple Kernels - Multi-GPUs - Matrix-Matrix Addition

8. Module 8 : CUDA enabled NVIDIA Programs using Kepler GPU features

8.1. Write a program for block matrix computation a CPU and GPU using dynamic parallelism (Dynamic Parallelism is a new feature introduced with Kepler GK110 that allows the GPU to generate new work for itself, synchronize on results, and control the scheduling of that work via dedicated, accelerated hardware paths, all without involving the CPU)

8.2. Write a program to implement the solution of Matrix system of Linear Equations AX=b by Conjugate Gradient method using dynamic parallelism

8.3. Write a program to implement the solution of Matrix system of Linear Equations AX=b by Jacobi method using Hyper-Q (Hyper Q enables multiple CPU cores to launch work on a single GPU simultaneously)

8.4. Write a program to measure performance per watt for Matrix into Matrix Multiplication using your own code and CUBLAS Library calls using NVIDIA NVML libraries.

8.5. Write NLA Code in which one kernel can launch another kernel, and can create the necessary streams

8.6. Write a CUDA Program for implementation of solution of Partial Differential Equations (PDEs) based on Finite Difference Method using CUBLAS routines