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 OpenCL enabled GPUs (AMD/NVIDIA) & CPUs (Intel)

The list of programs are developed based on OpenCL enabled AMD GPUs (Fire Stream /Fire Pro, AMD APUs, and NVIDIA GPU (Fermi & Kepler) technology. The information of these codes are publicly available as AMD technical documents, INTEL technical documents, and NVIDIA technical documents Books, GPU Conferences, Workshops, Notes Material and especially recent webinars and these have been partially incorporated.

List of Modules – OpenCL enabled NVIDIA GPUs /ADM GPUs

Module 1 : Getting Started : Basics - OpenCL
Module 2 : Getting Started: PGI OpenACC APIs on CUDA enabled NVIDIA GPU)
Module 3 : OpenCL Programs using BLAS libraries for Matrix Computations
Module 4 : OpenCL Programs - Application Kernels
Module 5 : OpenCL Memory Optimization Programs - Tuning & Performance

1. **Module 1: Getting Started: Basic OpenCL Program**
   - 1.1. Device Query
   - 1.2. Vector-Vector Addition: Simple (Code html Version)
   - 1.3. Vector-Vector Addition: Dimension of Workgroups – Work-items
   - 1.4. Simple Vector Addition Program
   - 1.5. Multiplication of Scalar with vector
   - 1.6. Open Source software: Matrix Computations OpenCL-MAGMA (cIMAGMA0.2)

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

2.1. Write a simple OpenCL Program for multiplication of scalar value and a vector using global memory (Using the keyword __global)
2.2. Write a simple OpenCL Program for multiplication of scalar value and a Matrix using global memory (Using the keyword __global)
2.3. Write a simple OpenCL Program for multiplication of scalar value and a Matrix using global memory (Using the keyword __global)
2.4. Write a OpenCL Program to compute vector vector addition using global memory
2.5. Write a OpenCL Program to compute matrix matrix addition using global memory
2.6. Write a OpencL program to find the total number of work-items in the x- and y-dimension of the NDRanges (Assume that OpenCL kernel is launched with a two-dimensional (2D) NDRange. Use API get_global_size(0),get_global_size(1))
2.7. Write a OpenCL program to device a query that returns the constant memory size supported by the device
2.8. Write a OpenCL program to get unique global index of each work item that calls the function get_global_id(0) of OpenCL API.
2.9. Write OpenCL program to compute the value of pie
2.10. Write a OpenCL Program to find prefix sum of an array using global memory
2.11. Write a OpenCL Program to perform vector - vector multiplication using global memory
2.12. Write a OpenCL Program to compute infinity norm of a real squre matrix using global
2.13. Memory
2.14. Write a OpenCL Program to compute tranpose of a square matrix using global memory & local memory
2.15. Write a OpenCL Program to compute matrix into vector multiplication using global & local memory
2.16. Write a OpenCL program to compute solve Ax=b Matrix System of linear equations based on Jacobi method on Multi-GPUs using OpenCL. (Assignment)
2.17. Write OpenCL program to implement the solution of Matrix system of Linear Equations [A]{x}={b} by Conjugate Gradient (Iterative) Method). (Assignment)
2.18. Write a OpenCL program to compute sparse Matrix into vector multiplication (Assignment)
2.19. Write a OpenCL program to compute sparse Matrix into vector multiplication (Assignment)

3. Module 3 : OpenCL Programs using BLAS libraries for Matrix Computations

3.1. Write OpenCL program for Matrix - Vector multiplication using BLAS Library (Assignment)

3.2. Write a OpenCL Program for Matrix -Matrix multiplication using AMD APP BLAS library function calls.

3.3. Write a OPENCL Program for implement solution of matrix system of linear equations Ax = b by Jacobi method using BLAS librarry (Assignment)

3.4. Write OpenCL program to implement the solution of Matrix system of Linear Equations [A]{x}={b} by Conjugate Gradient (Iterative) Method). (Assignment)

3.5. Demonstrate performance of LINPACK OpenCL open source Software on AMD-APP GPU (Assignment)

4. Module 4 : OpenCL Programs - Application Kernels.

4.1. Write a OpenCL Program for implementation of solution of Partial Differential Equations (PDEs) based on Finite Difference Method

4.2. Write a OpenCL Program for implementation of solution of Partial Differential Equations (PDEs) based on Finite Difference Method using AMD APP BLAS routines

4.3. Write a Program for implementation of Laplacian Edge Detection algorithm - Image Processing on multiple device-GPUs. (Assignment)

4.4. Write a OpenCL Program for implementation of solution of Partial Differential Equations (PDEs) based on Finite Element Method (Assignment)

4.5. Write a OpenCL Program for implementation of solution of Partial Differential Equations (PDEs) based on Finite Difference Method using multiple-GPUs (Assignment)

4.6. Write a OpenCL Program for implementation of String Search (Boyer-Moore) algorithm (Assignment)

5. Module 5: OpenCL Memory Optimization Programs - Tuning & Performance

5.1. Write a OpenCL program to measure the time taken for different data sizes that copy (write) data from the pinned host memory to the device memory using clEnqueueWriteBuffer()

5.2. Measure time for OpenCL (blocking or non-blocking calls) and kernel executions using either CPU or GPU timers (OpenCL GPU timers or OpenCL events)

5.3. Code to measure the effective bandwidth for a 1024x1024 matrix (Single /Double Precision) using Fast Memory Path and Complete Path and measure Ratio to Peak Bandwidth

5.4. Write a program to calculate memory throughput using OpenCL visual profiler

5.5. Analyze the differences in calculation of efficient memory bandwidth with memory throughput using OpenCL visual profiler

5.6. Write a code to analyze the performance of highly data parallel computation such as matrix-matrix computations on GPUs in which each multiprocessor contains either 8,192 or 16,384 32-bit registers, these are partitioned among concurrent threads.

5.7. OpenCL Program to measure highest bandwidth between host and device based on page-locked

5.8. OpenCL Program to measure highest bandwidth between host and device based on

page-locked using blocking (synchronous transfer) *clEnqueueReadBuffer()* / *clEnqueueWriteBuffer()* call

5.9. OpenCL Program to measure highest bandwidth between host and device based on page-locked memory using non-blocking write (Asynchronous transfer) clEnqueueWriteBuffer() call with parameter CL_FALSE and blocking read from device to host using *clEnqueueReadBuffer()* call with parameter CL_TRUE

5.10. OpenCL Program to measure highest bandwidth between host and device based on pinned memory using non-blocking write (Asynchronous transfer) clEnqueueWriteBuffer() call with parameter CL_FALSE and blocking read from device to host using *clEnqueueReadBuffer()* call with parameter CL_TRUE

5.11. OpenCL : OpenCL program on Overlapping Transfers and Device Computation *oclCopyComputeOverlap* SDK. The SDK sample "*oclCopyComputeOverlap* " is devoted exclusively to exposition of the techniques required to achieve concurrent copy and device computation and demonstrates the advantages of this relative to purely synchronous operations.)

5.12. Write test suites focusing on performance and scalability analysis of different size of the data on different memory spaces i.e. 16 KB per thread limit on local memory, a (OpenCL __private memory), 64 KB of constant memory (*OpenCL_constant memory*) , 16 KB of share memory (OpenCL_local memory), and either 8,192 or 16,384 32-bit registers per multi-processor.

5.13. Write a code on how to use default work-group size at compile time, size of the work-group, role of compiler to allocate the number of registers for work-item, & enough number of wave fronts)

5.14. OpenL program for Matrix into Matrix Computation for different partition of matrices for coalescing global memory accesses (a) A Simple Access Pattern - the kth thread accesses the kth word in a segment; the exception is that not all threads need to participate; (b). A Sequential but Misaligned Access Pattern (sequential threads in a half warp access memory but not aligned with the segments); (c) Effects of misaligned accesses (d) Stride Access

5.15. Simple OpenCL program that computes matrix into vector multiply choosing the best optimized NDRange in which optimized number of work-items is launched. Setting up right *worksize* in *clEnqueueNDRangeKernel()* and number of work-items, a multiple of the warp size (i.e. 32), can be explored.

5.16. Simple OpenCL program that computes the product w of a width x height matrix Mby a vector V in which global memory access are coalesced and the kernel must be rewritten to have each work-group, as opposed to each work-item, compute elements of W. (Each work-item is now responsible for calculating part of the dot product of V and a row of M and storing it to OpenCL local memory)

5.17. OpenCL Parallel reduction (a) with shared memory Effects of Misaligned Accesses bank conflicts, (b) without shared memory Effects of Misaligned Accesses bank conflicts, (c) warp based parallelism

5.18. OpenCL Code for matrix-matrix multiply C = AAT based on (a) strided accesses to global memory, (b) shared memory bank conflicts, (an optimized version using coalesced reads from global memory)

5.19. AMD-APP (using CAL) OpenCL : Write a simple *HelloCAL* application program using CAL of AMD Accelerated Parallel Processing Technology calculating part of

the dot product of V and a row of M and storing it to OpenCL local memory)

5.20. AMD-APP: Write a Direct memory access (DMA) code to move data between the system memory and GPU local memory using *CALMemCopy* of AMD APP

5.21. AMD-APP : Write a code to use AMD-APP Asynchronous operations using CAL API of an application that must perform CPU computations in the application thread and also run another kernel on the AMD-APP stream processor

5.22. AMD-APP : Multiple device Write a matrix into vector multiplication based on self-scheduling algorithm using AMD-APP CAL CAL Application using Multiple Stream Processors Using *calDeviceGetCount* AMD-APP : CAL API of ADM-APP and ensure that application-created threads that are created on the CPU and are used to manage the communication with individual stream processors.)