Segundo Encuentro Nacional de Computación de Alto Rendimiento para Aplicaciones Científicas
WHPC13
7 al 10 de Mayo de 2013
FaMAF, Ciudad Universitaria, Universidad Nacional de Córdoba

Exámen para la casa

Alejandro Kolton, Nicolás Wolovick

Objetivos

- Manejar las bibliotecas Thrust y CUFFT que permiten una rápida productividad en ambientes heterogéneos de HPC.
- Comparar Thrust y PyCUDA en aplicaciones sencillas.
- Visualizar simulaciones utilizando Paraview.

Introducción

Los ejercicios a desarrollar son los que no se pudieron realizar el día viernes 10 de Mayo en la sesión de Laboratorio de la mañana. Estos ejercicios fueron desarrollados por Alejandro Kolton (CAB) y compilados por Nicolás Wolovick (FaMAF).
La mayoría de las referencias se realizan en base a la presentación que realizó Alejandro el jueves 9 de Mayo.

Se entrega el tarball `WHPC13TakeHome.tar.gz` que está preparado para correr en el cluster `mendieta`.

La estructura del tarball se divide en cuatro directorios:

- `common`: archivos comunes al resto de los directorios (temporizador y RNG).
- `cuerda`: PDE estocástica en función del tiempo para un campo escalar unidimensional en un potencial desordenado (slide 21). Versión simplificada del código utilizado en [arXiv:1211.7275]. Varios TODO para levantar.
- `multiplicacion_vectores`: esqueleto de `c=a*b` utilizando Thrust.
- `simplecufftthrust`: transforma de de Fourier unidimensional usando CUFFT, varios TODO para levantar.

Actividades

Ejercicio 1, multiplicacion_vectores, Slides 75-78

La idea de este ejercicio es que practique el uso básico de la biblioteca Thrust, y que compare las performances de la GPU y la CPU. Además que se realice una comparación con un lenguaje de scripting para HPC como PyCUDA.
Se presenta un esqueleto en C++ que hay que rellenar con Thrust para multiplicar dos

vectores punto a punto y almacenarlo en un tercero.
Tareas:

- Armar el programa usando Thrust.
- Correrlo en CPU para 1, 2, 4, 8, 16 procesadores. Correrlo en GPU.
- Armar una versión similar utilizando PyCUDA. En los slides de Nicolas Pinto se encuentran cosas similares usando gpuarray y elementwise.
- Obtener un gráfico comparativo como el que se muestra en el Slide 78. Discutir los datos mostrados.

Ejercicio 2, simplecufftthrust, Slides 49-57, 74

Los objetivos, practicar el manejo básico de la biblioteca CUFFT, practicar el manejo básico de la biblioteca Thrust, practicar la interoperabilidad CUFFT-Thrust.
La versión CPU es en base a FFTW y es una implementación completa multicore: `simple_fftw_thrust.cpp`.

Tareas:

- Levantar los TODO.
- Medir la performance utilizando simple y doble precisión, diferentes tamaños de datos. Realizarlo tanto en CPU (1, 2, 4, 8, 16 cores) como en GPU.
- Realizar un gráfico comparativo resumiendo todas las mediciones de tiempos. Discutir los datos mostrados.

Importante: ¿Cómo están ordenadas las frecuencias de la transformada?

Ejercicio 3, cuerda, Slides 21-47, 71-73

El programa `qew_minimal.cu` simula la dinámica de una cuerda elástica en un medio desordenado en la GPU usando la biblioteca Thrust y la Random123. Es una versión reducida de:

> *Phys. Rev. E 87, 032122 (2013)*
> *Nonsteady relaxation and critical exponents at the depinning transition*
> *http://pre.aps.org/abstract/PRE/v87/i3/e032122*
> *http://arxiv.org/abs/1211.7275*

La explicación resumida del código esta en el material suplementario de la revista:
http://pre.aps.org/supplemental/PRE/v87/i3/e032122

Sin resolver los TODO ya se puede compilar con `make`, y larga algunos *timmings*.

Genera dos ejecutables a la vez, uno de CUDA (corre en GPU) y otro de OpenMP (corre en CPU multicore).
Los objetivos son: practicar el manejo básico de la biblioteca Thrust y Random123; comparar performances CPU vs GPU; aprender a combinar herramientas para resolver una ecuación diferencial parcial estocástica con desorden congelado; visualizar los resultados

de la simulación con Paraview.

Tareas:

- Levantar los `TODO`.
- Medir performance para diferentes tamaños, tanto para CPU (1, 2, 4, 8, 16 cores) y GPU.
- Obtener un gráfico comparativo. Discutir los datos mostrados.
- Obtener una **película** de la simulación utilizando Paraview.

Tareas opcionales:

- ¿Cómo mejoraría la performance del código?
- Graficar la cuerda en realtime utilizando OpenGL.

Entrega

- El trabajo es individual.
- Entregar un tarball con las modificaciones al mail `nicolasw@famaf.unc.edu.ar`.
- Incluir un informe en formato PDF donde se explique brevemente lo realizado y se consignen los **tres gráficos**, junto con una mínima discusión sobre los datos que refleja cada uno, y un enlace a la **película** realizada con la simulación de la cuerda (youtube.com, mega.co.nz, etc.).
- Deadline: **26 de Julio**.