Universidad Nacional de Córdoba

FaMAF | GPGPU Computing Group

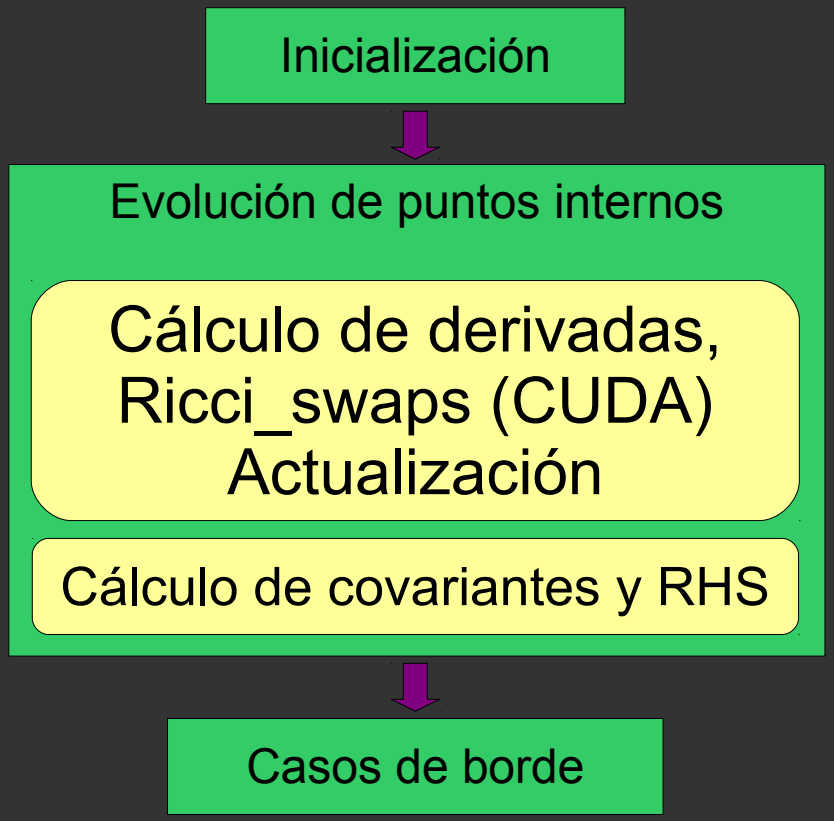
Clase 06 (Magneto-Hydro Dynamics)

Dionisio E Alonso

Magnetohydrodynamics

- Nombre completo: evolutionary Partial Differential Equations in magnetohydrodynamics and geometry.
- Se utilizan diferencias finitas para resolver ecuaciones de diferencias parciales.
- Nos concentramos en Magneto-hidro-dinámicas de fluídos y flujos de Ricci.

Estructura del programa



- A futuro se planea escribir en CUDA la sección de Covariantes y Right-Hand Side
- Los casos de borde se calculan usando MPI

Problemas de integración

- La estructura interna del programa está pensada para correr en nodos MPI
- Actualmente contamos con una sóla máquina con capacidades CUDA
- Se necesita secuencializar los cálculos de CUDA

Ricci_flow.c

```
   /* Define swap fields for box_g of Phi */

#ifdef CUDA
   turn=0; /* This is absolutely no safe, but is an approach */
   MPI_Comm_rank(MPI_COMM_WORLD, &rank);
   MPI_Comm_size(MPI_COMM_WORLD, &size);

   if (rank > 0)
      MPI_Recv(&turn, 1, MPI_INT, rank-1, rank-1, MPI_COMM_WORLD, &status);

   ricci_swaps(fields_ptr, &dfields, &auxfields,
n_gridpts_1*n_gridpts_2*n_gridpts_3);

   turn++;
   //~ printf("TURN: t:%d - r:%d - s:%d\n", turn, rank, size);
   if (rank < size-1)
      MPI_Send(&turn, 1, MPI_INT, rank+1, rank, MPI_COMM_WORLD);
#else
   {
      register int grid_ind1, grid_ind2, grid_ind3;
      for (grid_ind1 = ni_1; grid_ind1 < nf_1; ++grid_ind1) {
```

Problemas de integración (Cont.)

- Compilación con flags que no conoce nvcc
  - Hay 2 posibles soluciones
- Linkeo de las bibliotecas, una vez lograda la compilación

Makefile

```
LINK := g++ -fPIC
export LD_LIBRARY_PATH := $(CUDA_INSTALL_PATH)/lib
CFLAGS += -O3 -DCUDA
CCFLAGS += -ffast-math -finline-functions
NVCCFLAGS += -ccbin /usr/bin/gcc-4.3 -arch sm_13
LDFLAGS = -L$(CUDA_INSTALL_PATH)/lib -lcudart -L. -lbbhutil -lm -lmpi

CC = mpic++

%.o: %.c
   $(CC) -c $< $(CFLAGS) $(CCFLAGS) -o $@
   $(CC) -M $< $(CFLAGS) $(CCFLAGS) > $@.dep

%.o: %.cu
   $(NVCC) -c $< $(CFLAGS) $(NVCCFLAGS) -o $@
   $(NVCC) -M $< $(CFLAGS) $(NVCCFLAGS) > $@.dep

%.ptx: %.cu
   $(NVCC) --ptx $< $(CFLAGS) $(NVCCFLAGS) -o $@
```

Otra forma hubiera sido

```
LINK := g++ -fPIC
export LD_LIBRARY_PATH := $(CUDA_INSTALL_PATH)/lib
CFLAGS += -O3 -DCUDA
CCFLAGS +=
NVCCFLAGS += -ccbin /usr/bin/gcc-4.3 -arch sm_13 --compiler-options "-ffast-math -finline-functions"
LDFLAGS = -L$(CUDA_INSTALL_PATH)/lib -lcudart -L. -lbbhutil -lm -lmpi

CC = mpic++
```

- Y compilar todo con nvcc

Otros conflictos de paralalización

- El manejo de memoria con matrices n-dimensionales
- Diferentes acercamientos a una solución
  - Greedy: alojar en el dispositivo el puntero ** (NO)
    - Ejemplo: float **a;
  - Plan B: alojar los vectores que cuelgan del ** en el dispositivo. Anda, pero es complicado de visualizar
  - El mejor, serializar matrices con matemática (perfecto)
    - Falencias: tamaño máximo del vector

Resultados

Por el momento pendientes... :-(