Asociación Argentina
de Mecánica Computacional

Mecánica Computacional Vol XXXVI, págs. 589-604 (artículo completo)
José G. Etse, Bibiana M. Luccioni, Martín A. Pucheta, Mario A. Storti (Eds.)
San Miguel de Tucumán, 6-9 Noviembre 2018

RESOLUCIÓN DE LAS ECUACIONES DE NAVIER-STOKES EN MULTI-GPGPU

NUMERICAL SOLUTION OF THE NAVIER-STOKES EQUATIONS ON MULTI-GPGPU

Fernando F. Benitez[a], Jorge D'Elia[a] y Mario A. Storti[a]

[a]*Centro de Investigación de Métodos Computacionales (CIMEC), CONICET, Predio CONICET Santa Fe, Colectora Ruta Nac 168, Km 472, Paraje El Pozo, Santa Fe, Argentina, http://www.cimec.org.ar*

Palabras clave: Multi-GPGPU, Ecuaciones de Navier-Stokes, Método de volúmenes finitos.

Resumen. En este trabajo se presenta la resolución de las ecuaciones de Navier-Stokes mediante un enfoque multi-GPGPU con GPUs ubicados en diferentes nodos del clúster que interactúan mediante el protocolo MPI. La idea es usar descomposición de dominio de modo que cada GPU resuelve una parte del dominio computacional y MPI es usado para intercambiar datos de borde. El problema a resolver es el conocido lid-driven cavity flow que ha sido muy usado como caso de validación para nuevos métodos numéricos y códigos en dinámica de fluidos computacional. Se utiliza para la discretización espacial un esquema de grilla colocada en el contexto del método de volúmenes finitos sobre mallas estructuradas cartesianas de paso constante. El acople de presión y velocidad se realiza mediante el método Semi-Implicit Method for Pressure Linked Equations (SIMPLE). La ecuación de corrección de la presión es resuelta por el método de Gradientes Conjugados (CG) ya que la matriz es simétrica. Mientras que las ecuaciones de momento son resueltas por Gradientes Biconjugado Estabilizado (Bi-CGSTAB), ya que la matrices no son simétricas debido a la discretización del término convectivo. Se presentan varias opciones de discretización del término convectivo como ser diferencias centradas, upwind y QUICK. Se presentan su performance computacional en multi-GPGPU.

Keywords: Multi-GPGPU, Navier-Stokes Equations, Finite Volume Method.

Abstract. In this work the resolution of the Navier-Stokes equations is presented by means of a multi-GPGPU approach with GPUs located in different nodes of the cluster interacting through the MPI protocol. The idea is to use domain decomposition so that each GPU solves a part of the computational domain and MPI is used to exchange boundary data. The problem to solve is the well-known lid-driven cavity flow that has been widely used as a validation case for novel numerical methods and codes in computational fluid dynamics. A grid scheme placed in the context of the finite volume method on structured cartesian meshes with a constant pitch is used for spatial discretization. The coupling between pressure and velocity is done through the Semi-Implicit Method for Pressure Linked Equations (SIMPLE). The pressure correction equation is solved by Conjugated Gradients (CG) method since the matrix is symmetric. While momentum equations are solved by Bi-Conjugate Gradients Stabilized (Bi-CGSTAB), since the matrix of the systems are not symmetric due to the discretization of the convective term. Several discretization options are presented for the convective term like central differences, upwind and QUICK. Their computational performance in multi-GPGPU is presented.

1. INTRODUCCIÓN

En los sistemas paralelos y distribuidos, las GPGPUs (General-Purpose Computing on Graphics Processing Units) se han convertido en un serio competidor en el clásico ambiente de las CPUs. El alto rendimiento que concierne a operaciones de punto flotante ofrece capacidades que las hacen muy atractivas de forma tal que los algoritmos numéricos pueden acelerarse sustancialmente siempre y cuando se mapeen bien a la característica específica del hardware. Además, la tecnología de transferencia de datos se sigue perfeccionando buscando reducir la brecha ya existente entre el ancho de banda y la velocidad de cálculo. El presente trabajo se centra en el problema driven cavity flow que ha sido utilizado como un clásico caso de análisis para validar técnicas y métodos como también códigos de CFD como puede verse en la Tabla 1. Un código de Volúmenes Finitos acelerados con GPUs por medio de descomposición de dominio es desarrollado y utilizado para producir resultados utilizando una resolución de grilla hasta las 512×512 celdas. Los resultados obtenidos se comparan con datos del caso estacionario disponible en la literatura y una simulación para el caso transiente es presentado.

Año	Referencia	Formulación	Esquema Discreto	Grilla	Número de Reynolds (Re)
1982	Ghia et al. (1982)	$\psi - \omega$	FD	257 x 257	100–10,000
1985	Kim y Moin (1985)	u, v, p	FS	97 x 97	1–5000
1998	Botella y Peyret (1998)	u, v, p	Cheb.	160	1000
2005	Erturk et al. (2005)	$\psi - \omega$	FDM	401x401 - 601x601	1,000–21,000
2005	Gupta y Kalita (2005)	$\psi - u, v$	FDM	161 x 161	100–10,000
2010	Kalita y Gupta (2010)	$\psi - \omega$	FD	161 x 161	1000–10,000
2013	Magalhães et al. (2013)	u, v, p	FV	6910	1000

Tabla 1: Literatura sobre driven cavity flow ordenada cronológicamente

2. ECUACIONES DE NAVIER-STOKES

Las ecuaciones de conservación de masa y de momento de forma integral para una solución por método de volúmenes finitos (FVM) son:

$$\int_S \rho \vec{v} \cdot \vec{n} dS = 0 \tag{1}$$

$$\int_V \frac{\partial(\rho u_i)}{\partial t} dV + \int_S \rho u_i \vec{v} \cdot \vec{n} dS = -\int_S p \vec{i_i} \cdot \vec{n} dS + \int_S \mu \nabla u_i \cdot \vec{n} dS \tag{2}$$

Estas ecuaciones son para un flujo transiente incompresible, fluido Newtoniano, con densidad ρ y viscosidad dinámica μ constantes. El versor $\vec{i_i}$ es $\vec{i}$, $\vec{j}$, $\vec{k}$ para $i = 1, 2, 3$ respectivamente, de la misma forma u_i representa cada componente de la velocidad $\vec{v}$.

Este sistema presenta 3 importantes cuestiones:

- La incompresibilidad muestra una falta de ecuación de evolución para la presión y requiere un tratamiento especial para el acoplamiento de la presión y velocidad.
- El término advectivo tiene que ser estabilizado.
- La no linealidad del término advectivo tiene que ser resuelto usando un sistema no lineal o mediante un sistema linealizado (esta última opción es la elegida en este trabajo).

3. DISCRETIZACIÓN

El dominio de solución es dividido en $N_i \times N_j$ volúmenes de control (VC) sobre una grilla cartesiana de paso constante (véase la Figura 1) donde la cantidad desconocida (presión y velocidad) se encuentra en el centro de la celda (grilla colocada).

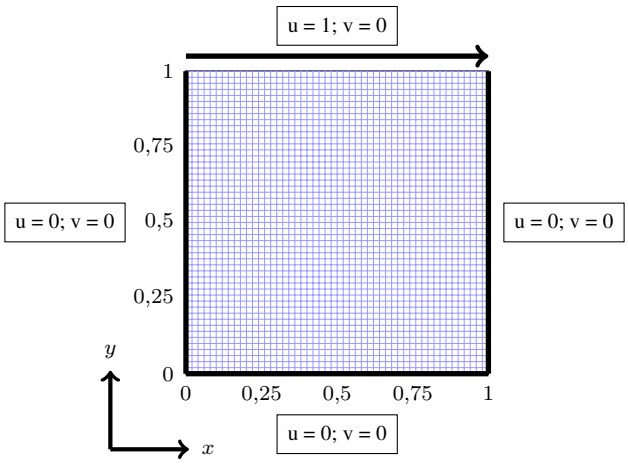

Figura 1: Esquema de grilla colocada del dominio computacional con condiciones de bordes en velocidad.

Las ecuaciones anteriores son aplicadas a cada VC donde las integrales son evaluadas usando una aproximación a la regla del punto medio, esto es, una aproximación a la integral de segundo orden que requiere solo el valor del integrando evaluado en el centro del dominio de integración.

$$\int_V \phi dV \approx \phi_C V \tag{3}$$

$$\int_{S_e} \phi \vec{i} \cdot \vec{n} dS \approx \phi_e S_e \tag{4}$$

$$\int_{S_e} \nabla \phi \cdot \vec{n} dS \approx (\nabla \phi)_e S_e \tag{5}$$

Para el término transiente usando con una integración temporal de primer orden (Forward Euler) quedaría de la siguiente forma

$$\int_V \frac{\partial(\rho u_i)}{\partial t} dV \approx \frac{(\rho u_i)_C^{n+1} - (\rho u_i)_C^n}{\Delta t} V \tag{6}$$

Los términos no lineales (flujos convectivos) son linealizados usando un esquema de iteración de Picard, por ejemplo, en la cara este del VC sería:

$$\int_{S_e} \rho \phi \vec{v} \cdot \vec{n} dS = \bar{\phi} \int_{S_e} \rho \vec{v} \cdot \vec{n} dS \approx \dot{m}_e \phi_e \tag{7}$$

Donde el flujo másico $\dot{m}$ es tomado desde una previa iteración y ϕ es reemplazado por algunas de las componentes de velocidad u_i. Esto significa que la no linealidad se trata por medio de un proceso iterativo en que los coeficientes son calculados al comienzo de cada iteración en base a valores obtenidos en la previa iteración. Este proceso puede producir grandes cambios

de $\vec{v}$ y puede afectar la tasa de convergencia a tal grado que una divergencia puede ocurrir. Para que los cambios no sean muy grandes, una técnica de relajación es usada cuando el paso de tiempo es largo como se mostrará más adelante.

Dado que las variables se encuentran en centros de celda (véase la Figura 2), los valores en las caras pueden aproximarse por distintos esquemas. En este trabajo se utilizan:

Upwind Differencing Scheme (UDS)

$$\phi_e = \phi_C \quad si \quad \dot{m}_e > 0 \tag{8}$$

$$\phi_e = \phi_E \quad si \quad \dot{m}_e < 0 \tag{9}$$

Central Differencing Scheme (CDS)

$$\phi_e = \phi_E f_e + \phi_C(1 - f_e) \qquad f_e = \frac{x_e - x_C}{x_E - x_C} \tag{10}$$

Quadratic Upwind Interpolation for Convective Kinematics (QUICK)

$$\phi_e = \frac{3}{8}\phi_E + \frac{3}{4}\phi_C - \frac{1}{8}\phi_W \quad si \quad \dot{m}_e > 0 \tag{11}$$

$$\phi_e = \frac{3}{8}\phi_C + \frac{3}{4}\phi_E - \frac{1}{8}\phi_{EE} \quad si \quad \dot{m}_e < 0 \tag{12}$$

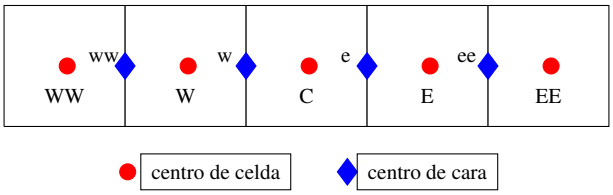

Figura 2: Esquema de valores centrados en celda y valores en caras de celda.

El UDS tiene un error a primer orden e introduce difusión numérica. El CDS es una aproximación con error a segundo orden y puede ser inapropiado para flujos convectivos más dominantes (con alto número de Peclet en celda) ya que conducen a oscilaciones no físicas en la solución cuando la grilla es gruesa dado que en estas los errores de discretización son mayores. El esquema QUICK fue propuesto por Leonard (1979) y consiste en una interpolación cuadrática entre los vecinos aguas arriba y aguas abajo cuando se evalúa ϕ en la cara de celda y su error es a tercer orden.

Dado que los coeficientes de la matriz provenientes de una discretización con CDS no es diagonal dominante, algunos solvers iterativos presentan dificultades en la resolución. Este problema pueden evitarse utilizando una aproximación por corrección diferida donde el valor en el centro de la cara es expresado como

$$\phi_e = \phi_e^{UDS} + (\phi_e^{HOS} - \phi_e^{UDS})^o \tag{13}$$

Donde el superíndice UDS denota la aproximación de primer orden upwind y HOS indica que el valor se obtiene por interpolación de mayor orden como CDS y QUICK, y o significa que los valores son obtenidos de una previa iteración. Entonces solo UDS contribuye a los coeficientes de la matriz mientras que los términos entre paréntesis (que pueden ser vistos como

una corrección de segundo orden a la aproximación UDS de primer orden) es incluido como término fuente. En un estado de convergencia de la solución los términos UDS se cancelan. Esta aproximación conduce a un método con dos ventajas, por un lado se preserva la propiedad de dominancia diagonal de los coeficientes de la matriz por el uso de UDS y por el otro la precisión del HOS. La desventaja es que la tasa de convergencia puede verse afectada.

El cálculo de los flujos difusivos requiere la derivada normal en los centros de las caras del volumen de control que son aproximados por diferencias centradas de segundo orden.

$$\left(\frac{\partial \phi}{\partial x}\right)_e \approx \frac{\phi_E - \phi_C}{x_E - x_C} \tag{14}$$

Las fuerzas de presión también son aproximadas en las caras de las celdas usando la regla del punto medio, donde la presión en el centro de las caras de la celda se obtiene interpolando linealmente entre los centros de celda que se encuentran a los lados de la cara.

Finalmente una ecuación algebraica se obtiene siguiendo los pasos anteriores para cada VC,

$$a_C u_C + \sum_{F \sim NB} a_F u_F = b_C \tag{15}$$

donde F recorre sobre los vecinos de la celda C que son : E, W, N, S.

La solución sobre el dominio completo puede verse como

$$Au = b \tag{16}$$

donde A es una matriz cuadrada de coeficientes y b es el vector RHS.

Esta ecuación es resuelta con Gradientes Bi-Conjugados estabilizado (BiCGSTAB) (mediante pocas iteraciones internas) en cada iteración del lazo SIMPLE (que puede verse como iteración externa) para cada componente de velocidad. El residuo usualmente cae uno o dos órdenes de magnitud después de cada lazo de iteraciones internas. Notar que ambos A y b son actualizados cada vez que se resuelve el sistema para cada iteración del lazo SIMPLE.

Luego de resolver las ecuaciones de momento un nuevo flujo másico que pasa a través de las caras del VC son calculados,

$$\dot{m}_e^* = \rho u_e^* \triangle y \tag{17}$$

donde el superíndice $*$ denota una solución temporaria nueva que debe ser corregida, y u_e es usado para denotar la velocidad en la dirección x. Los valores de velocidad en las caras de la celda, en este caso u_e, es calculado usando interpolación de Rhie y Chow (1983):

$$u_e^* = \overline{(u)_e^*} + \overline{\left(\frac{\triangle \Omega}{A_p}\right)_e} \left[\widetilde{\left(\frac{\partial p}{\partial x}\right)_e} - \left(\frac{\partial p}{\partial x}\right)_e \right] \tag{18}$$

donde $\triangle \Omega$ es el volumen del VC, la barra denota interpolación lineal, y la tilde denota promedio aritmético. Los corchetes encierran términos de corrección que es cero cuando la presión varia linealmente o de forma cuadrática. Esto está diseñado para detectar oscilaciones artificiales (que pueden desarrollarse cuando se utiliza mallas colocadas) las cuales son eliminadas. Las derivadas en las caras de la celda son calculadas siguiendo la ecuación (14).

Luego los flujos másicos calculados no satisfacen la ecuación de continuidad,

$$\sum_n \dot{m}_n^* = \triangle \dot{m} \quad n = e, w, n, s \tag{19}$$

por lo tanto estos flujos deben ser corregidos.

El algoritmo SIMPLE está basado en la hipótesis de la corrección del campo de velocidad es proporcional al gradiente de la corrección de la presión, esto es, en la cara e :

$$u_e^{'} = -\overline{\left(\frac{\triangle\Omega}{A_P}\right)_e}\left(\frac{\partial p^{'}}{\partial x}\right)_e \tag{20}$$

En los bordes las velocidades son prescriptas y no son corregidas, esto equivale a especificar gradiente nulo en la corrección de la presión, por lo tanto, se debe proponer condiciones de tipo Neumann homogéneas para la ecuación de corrección de la presión.

Desde la necesidad de corregir los flujos másicos para satisfacer la ecuación de continuidad resulta una ecuación de corrección de la presión de la misma forma que la ecuación (15). Esto representa una forma discretizada de la ecuación de Poisson y es resuelto usando Gradiente Conjugado (CG) dado que la matriz resultante es simétrica. Pocas iteraciones son realizadas, si el residuo se reduce en 5 órdenes las iteraciones son detenidas.

La naturaleza no lineal de las ecuaciones requiere el uso de relajación. Aquí el cambio de las velocidades son relajadas por 0.8 y el cambio de presión por 0.2.

El algoritmo completo envuelve la solución de tres sistemas de ecuaciones, dos de momento y uno de corrección de presión. Los flujos másicos y presión son corregidos, y esos valores son usados para armar las matrices de coeficientes y el RHS para la siguiente iteración externa.

El proceso es repetido hasta que la norma del residuo (la suma de los valores absolutos sobre todos los VCs) para cada ecuación sea menor que una valor de tolerancia relativa prescripta teniendo en cuenta la norma del residuo inicial de las iteraciones externas. Para este trabajo se utilizo una caída del residuo de 4 órdenes de magnitud.

4. DIVISIÓN DEL DOMINIO Y ALGORITMOS DE RESOLUCIÓN

4.1. Descomposición de dominio

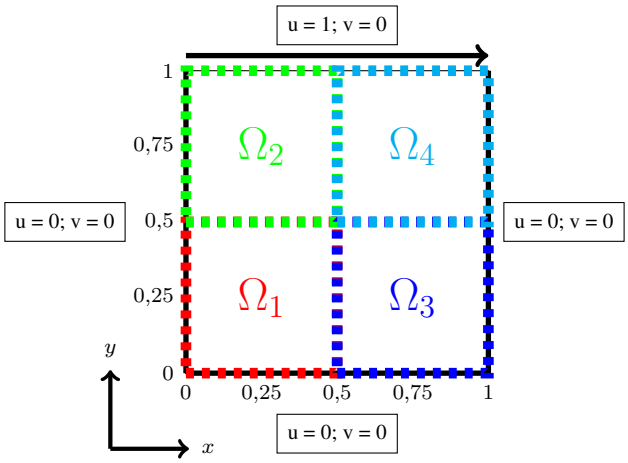

Figura 3: Esquema de partición del dominio computacional en subdominios.

El dominio computacional fue dividido en 4 subdominios (véase la Figura 3) mediante subdivisión en las dos direcciones x e y, sin embargo, el código desarrollado contempla también los casos en que se disponga de un mayor número de GPGPUs, basta con que la cantidad de GPGPUs disponibles pueda configurarse bajo la siguiente forma de descomposición $Nsub_x \times Nsub_y$.

4.2. Algoritmo de acople de presión y velocidad

Como se dijo anteriormente la solución del método esta basado en el algoritmo SIMPLE, uno de los algoritmos más usados para calcular flujos incompresibles. El atractivo de la solución del método es su versatilidad para extenderse a problemas de transferencia de calor, transferencia de masa y modelos de turbulencia. Pero por otro lado una de sus desventajas es el débil acople entre las variables, dado la linealización de las ecuaciones mediante un esquema de iteración de Picard. El esquema del algoritmo SIMPLE puede verse en la Figura 4 :

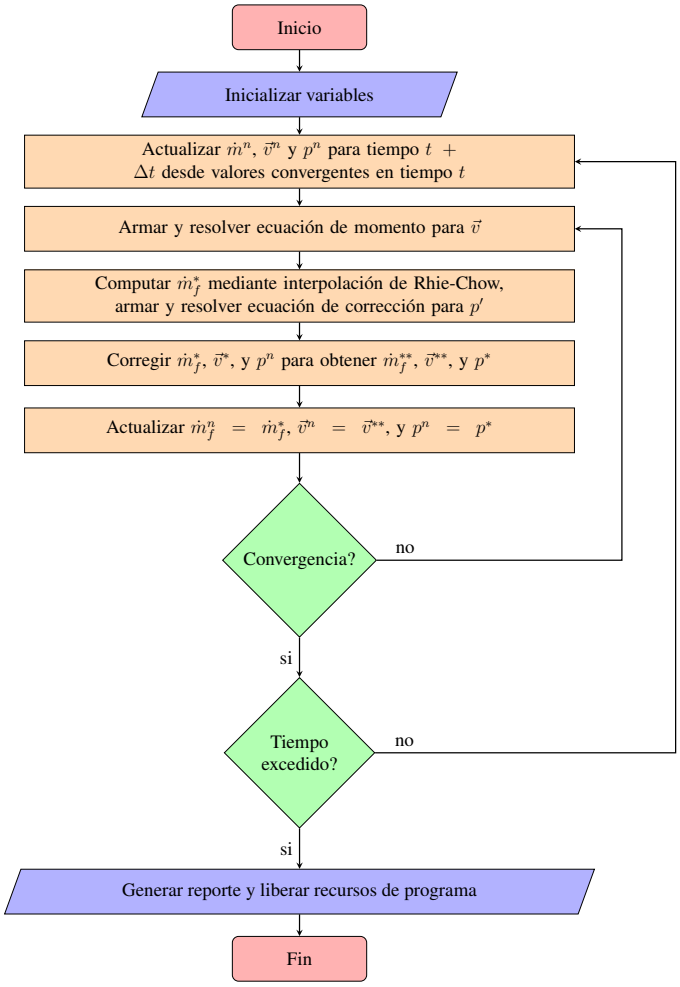

Figura 4: Diagrama de flujo del algoritmo SIMPLE.

4.3. Algoritmos de solución de sistemas lineales

En este trabajo se presenta los algoritmos CG y BiCGSTAB en sus versiones para sistemas de memoria distribuida. Estos métodos se pueden ver más detalladamente en Kelley (1995), Golub y Van Loan (1996). Pueden verse superíndices n en las matrices y vectores indicando a que $rank$ pertenece. También se puede ver las operaciones como el producto matriz-vector, AXPBY, AXPBYPCZ, DOT y las funciones de comunicación que sirven para intercambiar los valores de borde.

Algoritmo 1: Gradientes conjugados precondicionado para sistemas de memoria distribuida

$\mathbf{r}_0^n \leftarrow \mathbf{b}^n - \mathbf{A}^n\mathbf{x}_0^n$
$\rho \leftarrow 0$
$reduccion\_distribucion(\mathbf{r}_0^n\mathbf{r}_0^n, rr_{nuevo}, suma)$
$rr_0 \leftarrow rr_{nuevo}$
$i \leftarrow 0$
while $sqrt(rr_{nuevo}) < abtol + sqrt(rr_0)retol$ **and** $i < itermax$ **do**
  $intercambio\_halo(\mathbf{r}_i^n)$
  $\mathbf{z}_i^n \leftarrow \mathbf{M}^n\mathbf{r}_i^n$
  $\rho_{viejo} \leftarrow \rho$
  $reduccion\_distribucion(\mathbf{r}_i^n\mathbf{z}_i^n, \rho, suma)$
  if $i = 0$ **then**
    $\mathbf{p}_i^n \leftarrow \mathbf{z}_i^n$
  else
    $\beta \leftarrow \frac{\rho}{\rho_{viejo}}$
    $\mathbf{p}_i^n \leftarrow \mathbf{z}_i^n + \beta\mathbf{p}_{i-1}^n$
  end if
  $intercambio\_halo(\mathbf{p}_i^n)$
  $reduccion\_distribucion(\mathbf{p}_i^n\mathbf{A}^n\mathbf{p}_i^n, pAp, suma)$
  $\alpha \leftarrow \frac{\rho}{pAp}$
  $\mathbf{x}_i^n \leftarrow \mathbf{x}_{i-1}^n + \alpha\mathbf{p}_i^n$
  $\mathbf{r}_i^n \leftarrow \mathbf{r}_{i-1}^n - \alpha\mathbf{A}^n\mathbf{p}_i^n$
  $reduccion\_distribucion(\mathbf{r}_i^n\mathbf{r}_i^n, rr_{nuevo}, suma)$
  $i = i + 1$
end while

Algoritmo 2: Gradientes bi-conjugados estabilizado para sistemas de memoria distribuida

$\mathbf{r}_0^n \leftarrow \mathbf{b}^n - \mathbf{A}^n\mathbf{x}_0^n$
$\mathbf{p}_0^n \leftarrow \mathbf{r}_0^n$
$\hat{\mathbf{r}}_0^n \leftarrow \mathbf{r}_0^n$
$reduccion\_distribucion(\hat{\mathbf{r}}_0^n\mathbf{r}_0^n, \hat{r}r_{viejo}, suma)$
$reduccion\_distribucion(\mathbf{r}_0^n\mathbf{r}_0^n, rr_{nuevo}, suma)$
$rr_0 \leftarrow rr_{nuevo}$
$i \leftarrow 1$
while $sqrt(rr_{nuevo}) < abtol + sqrt(rr_0)retol$ **and** $i < itermax$ **do**
  $intercambio\_halo(\mathbf{p}_{i-1}^n)$
  $\mathbf{z}_i^n \leftarrow \mathbf{M}^n\mathbf{p}_{i-1}^n$
  $intercambio\_halo(\mathbf{z}_i^n)$
  $\mathbf{q}_i^n \leftarrow \mathbf{A}^n\mathbf{z}_i^n$
  $reduccion\_distribucion(\hat{\mathbf{r}}_0^n\mathbf{q}_i^n, \hat{r}q, suma)$
  $\alpha \leftarrow \frac{\hat{r}r_{viejo}}{\hat{r}q}$
  $\mathbf{s}_i^n \leftarrow \mathbf{r}_{i-1}^n - \alpha\mathbf{q}_i^n$
  $reduccion\_distribucion(\mathbf{s}_i^n\mathbf{s}_i^n, ss, suma)$
  if $sqrt(ss) < abtol + sqrt(rr_0)retol$ **then**
    $\mathbf{x}_i^n \leftarrow \mathbf{x}_{i-1}^n + \alpha\mathbf{z}_i^n$
    break
  end if
  $intercambio\_halo(\mathbf{s}_{i-1}^n)$
  $\mathbf{t}_i^n \leftarrow \mathbf{M}^n\mathbf{s}_{i-1}^n$
  $intercambio\_halo(\mathbf{t}_i^n)$
  $\mathbf{u}_i^n \leftarrow \mathbf{A}^n\mathbf{t}_i^n$
  $reduccion\_distribucion(\mathbf{u}_i^n\mathbf{s}_i^n, us, suma)$
  $reduccion\_distribucion(\mathbf{u}_i^n\mathbf{u}_i^n, uu, suma)$
  $\omega \leftarrow \frac{us}{uu}$
  $\mathbf{x}_i^n \leftarrow \mathbf{x}_{i-1}^n + \alpha\mathbf{z}_i^n + \omega\mathbf{t}$
  $\mathbf{r}_i^n \leftarrow \mathbf{s}_i^n - \omega\mathbf{u}_i^n$
  $reduccion\_distribucion(\mathbf{r}_i^n\mathbf{r}_i^n, \hat{r}r_{nuevo}, suma)$
  $\beta \leftarrow (\frac{\hat{r}r_{nuevo}}{\hat{r}r_{viejo}})(\frac{\alpha}{\omega})$
  $\hat{r}r_{viejo} \leftarrow \hat{r}r_{nuevo}$
  $\mathbf{p}_i^n \leftarrow \mathbf{r}_i^n + \beta\mathbf{p}_{i-1}^n - \beta\omega\mathbf{u}_i^n$
  $reduccion\_distribucion(\mathbf{r}_i^n\mathbf{r}_i^n, rr_{nuevo}, suma)$
  $i = i + 1$
end while

5. MULTI-GPU

5.1. Hardware y software

Clúster Pirayu - CIMEC

- 4 NVIDIA Tesla K40 (2880 cores), micro-arquitectura Kepler, Compute Capability 3.5.
- CUDA 8.0
- Thrust 8.0
- CUSP 0.5
- OpenMPI 2.0.1 con soporte CUDA-aware (GPUDirect2)

El tipo de dato utilizado para el cómputo es doble precisión. Además para las operaciones DOT, AXPBY, AXPBYCZ se utilizó la biblioteca Thrust y CUSP. La operación matriz por vector se implementó mediante el desarrollo del kernel producto matriz-vector ya que este producto es de estilo *matrix free*. Todas estas operaciones se efectúan en las GPU, solo en la etapa de generación de reporte los datos en GPU son transferidos al *host*. El código fue desarrollado de esta manera con el objetivo de minimizar la transferencia de datos entre la GPU y el *host*. La operación halo se implementó mediante funciones MPI con soporte GPUDirect2.

Si bien el uso de bibliotecas como Thrust y CUSP para realizar las operaciones algebraicas como AXPBYCZ y DOT reducen esfuerzos en implementar solvers en multi-GPU, el desarrollo de kernels para los operadores *matrix free* y la comunicación de halos no son triviales por que implican un profundo conocimiento del hardware como el uso y acceso eficiente de los distintos tipos de memoria y la implementación de iteradores.

El uso de las funciones de MPI con soporte GPUDirect2 reduce el esfuerzo y la cantidad de memoria a nivel de implementación en la comunicación de los halos ya que evita la duplicación de datos en el *host* para transmitir datos vía una implementación estándar de MPI. Por esta razón se obtiene una ganancia debido a que no se pierde tiempo en comunicar datos desde la memoria de la GPU a la memoria del *host* y además una reducción de líneas de código y recursos de memoria en tiempo de ejecución es obtenida para la comunicación. Estas ventajas serían más pronunciadas en 3D.

NVIDIA Tesla K40

Las aceleradoras NVIDIA Tesla K40 poseen 2.880 núcleos. Estos núcleos se organizan en 15 multiprocesadores (SM) que funcionan a una frecuencia 0,75 GHz. Cada SM posee 192 núcleos, todos los cuales están disponibles para cálculos de punto flotante de simple precisión, pero no todos están disponibles para doble precisión. Por lo tanto en cada SM solo 64 núcleos pueden realizar cálculos de doble precisión a una velocidad de 2 flops por ciclo.

Intel(R) Xeon(R) CPU E5-2650 v3

Los procesadores Intel(R) Xeon(R) CPU E5-2650 poseen 2 CPUs con 10 cores cada uno por lo que cada nodo cuenta con 20 cores con una frecuencia de reloj de 2.30 GHz, cada procesador cuenta con 25 MB de cache.

6. RESULTADOS DE CASO ESTACIONARIO

Se pueden observar en las Tablas 2, 3 y 4 el tiempo que demoró el problema en resolverse con las GPUs y cuanto más rápida fue la versión GPU con respecto a la CPU. De estos rendimientos se puede ver que el esquema UDS fue el más rápido (ya que no calcula el término de corrección

N=128	UDS	CDS	QUICK
Re 100	9 min. (1.2x)	10 min. (1.3x)	12 min. (1.3x)
Re 400	10 min. (1.3x)	11 min. (1.4x)	11 min. (1.4x)
Re 1000	11 min. (1.4x)	11 min. (1.6x)	12 min. (1.6x)

Tabla 2: Rendimiento del código en multi-GPU con respecto al código en multi-CPU

N=256	UDS	CDS	QUICK
Re 100	58 min. (1.3x)	66 min. (1.3x)	68 min. (1.4x)
Re 400	60 min. (1.4x)	62 min.(1.4x)	63 min. (1.5x)
Re 1000	61 min. (1.5x)	62 min. (1.5x)	64 min. (1.6x)

Tabla 3: Rendimiento del código en multi-GPU con respecto al código en multi-CPU

N=512	UDS	CDS	QUICK
Re 100	290 min. (2.1x)	313 min. (2.3x)	317 min. (2.5x)
Re 400	325 min. (2.2x)	329 min. (2.4x)	335 min. (2.5x)
Re 1000	337 min. (2.5x)	343 min. (2.6x)	349 min. (2.7x)

Tabla 4: Rendimiento del código en multi-GPU con respecto al código en multi-CPU

diferida) que los otros esquemas. También se puede observar que a medida que se aumenta el tamaño del problema el rendimiento va en aumento debido a que las operaciones vectoriales y matriciales se van incrementando en tamaño donde las GPUs sacan más provecho en su poder de cálculo. Además las comunicaciones son más eficientes en este escenario mediante el uso de la tecnología GPUDirect2 ya que no es necesario utilizar buffers auxiliares en el $host$ para comunicar los halos.

A continuación podemos observar las gráficas de comparación con el trabajo de Ghia et al. (1982) para $N = 256$, las gráficas de función de corriente y vorticidad para cada esquema en función del número de Reynolds. A partir de las gráficas podemos inferir que el esquema UDS es menos preciso para capturar los puntos de los perfiles de velocidades en las líneas medias vertical y horizontal, mientras que los esquemas CDS y QUICK son más precisos. A su vez QUICK es más preciso que CDS como se puede ver también en las gráficas de función de corriente y vorticidad ya que QUICK captura mejor las curvas de nivel a medida que el número de Reynolds es más alto, mientras que CDS y UDS presentan un desvío siendo en UDS más notorio en comparación con las gráficas de función de corriente y vorticidad de Ghia et al. (1982).

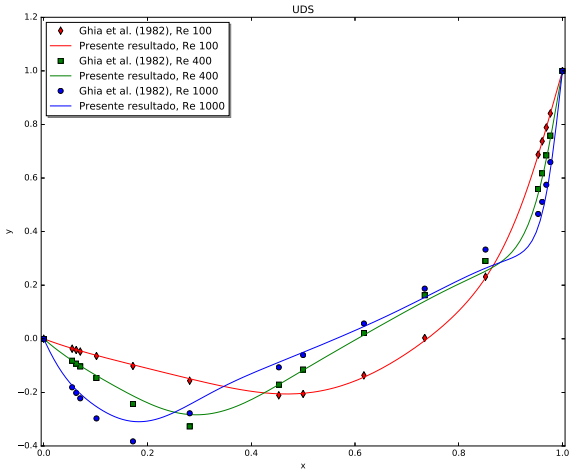

Figura 5: Perfil de u en la linea media vertical con UDS.

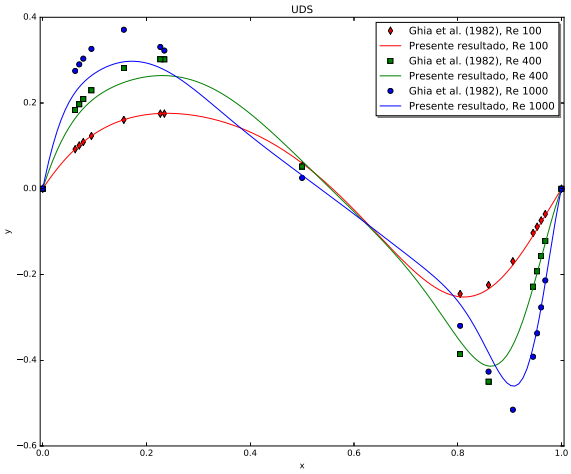

Figura 6: Perfil de v en la linea media horizontal con UDS.

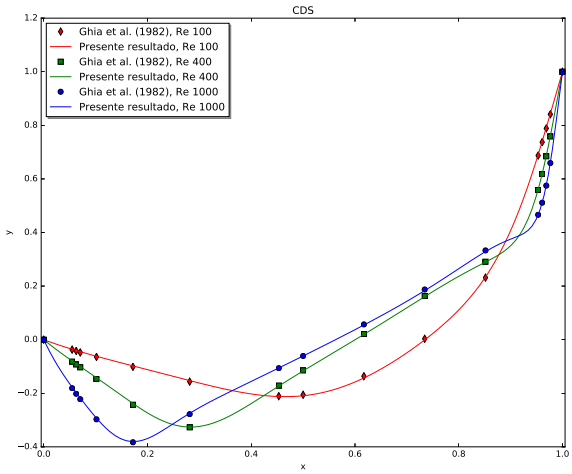

Figura 7: Perfil de u en la linea media vertical con CDS.

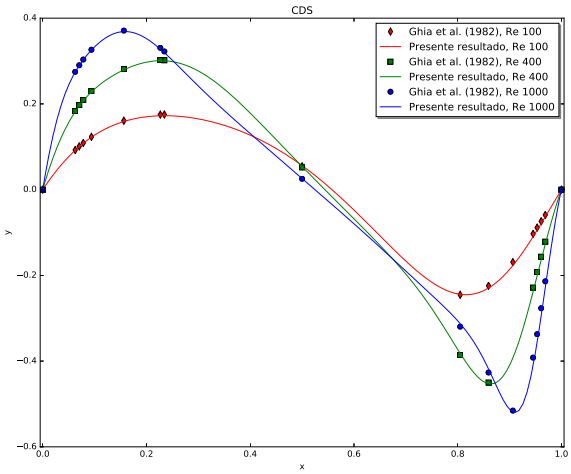

Figura 8: Perfil de v en la linea media horizontal con CDS.

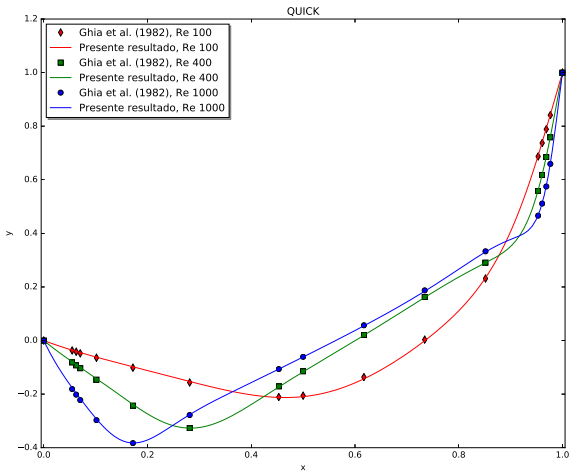

Figura 9: Perfil de u en la linea media vertical con QUICK.

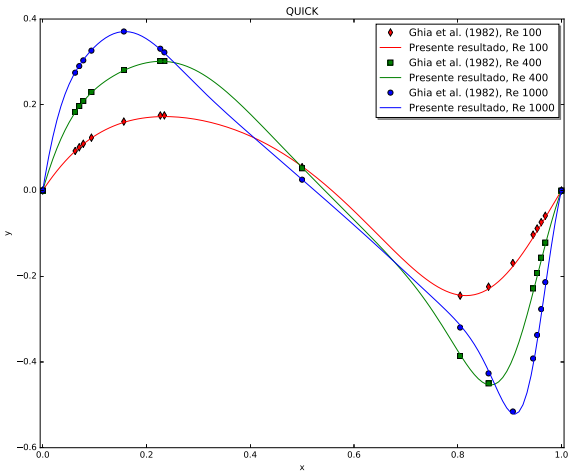

Figura 10: Perfil de v en la linea media horizontal con QUICK.

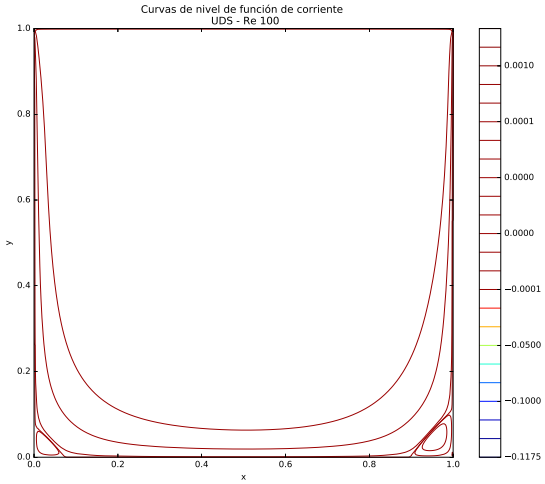

Figura 11: Curvas de nivel de función de corriente con UDS para Re = 100

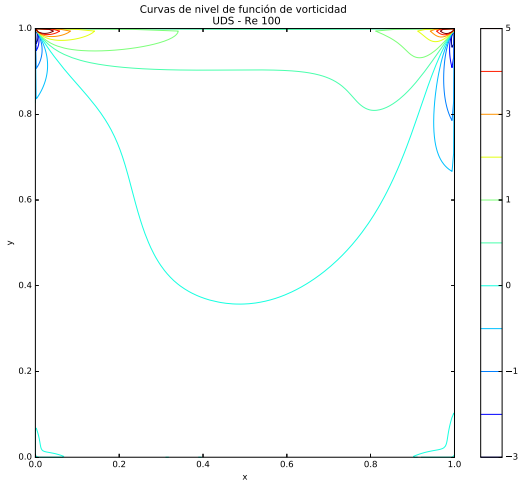

Figura 12: Curvas de nivel de función de vorticidad con UDS para Re = 100

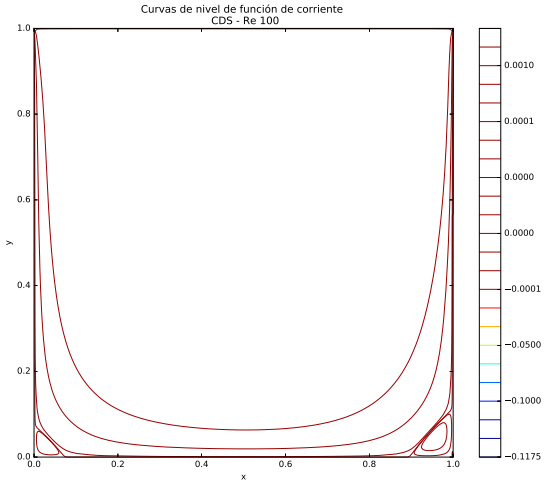

Figura 13: Curvas de nivel de función de corriente con CDS para Re = 100

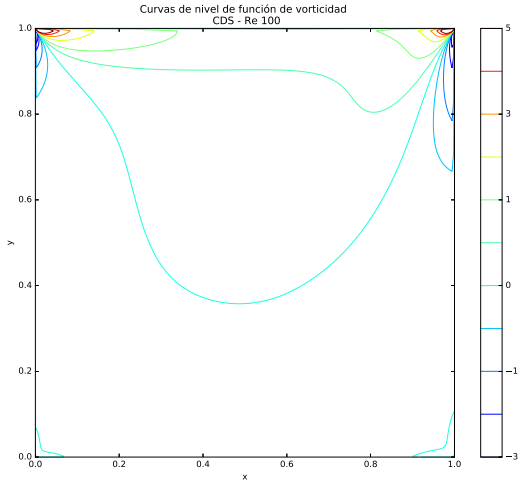

Figura 14: Curvas de nivel de función de vorticidad con CDS para Re = 100

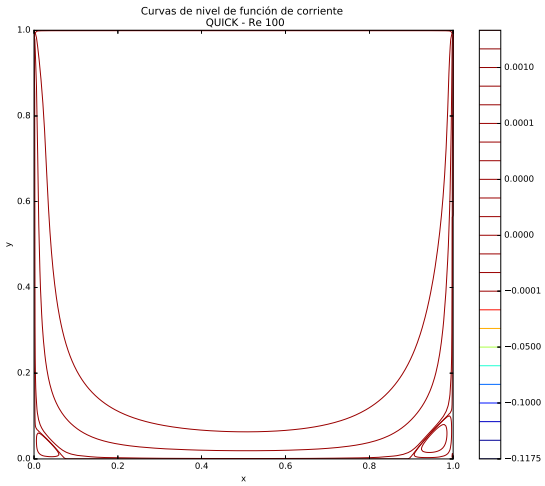

Figura 15: Curvas de nivel de función de corriente con QUICK para Re = 100

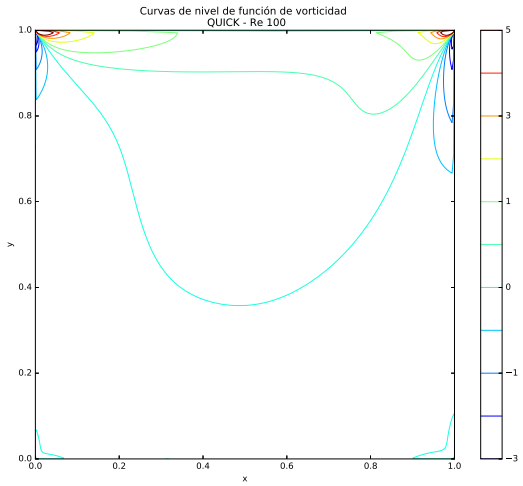

Figura 16: Curvas de nivel de función de vorticidad con QUICK para Re = 100

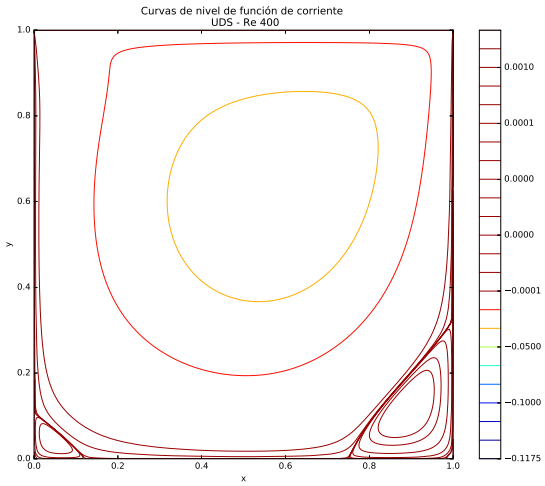

Figura 17: Curvas de nivel de función de corriente con UDS para Re = 400

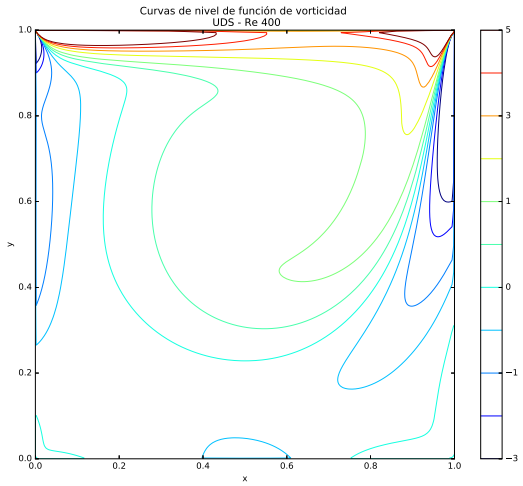

Figura 18: Curvas de nivel de función de vorticidad con UDS para Re = 400

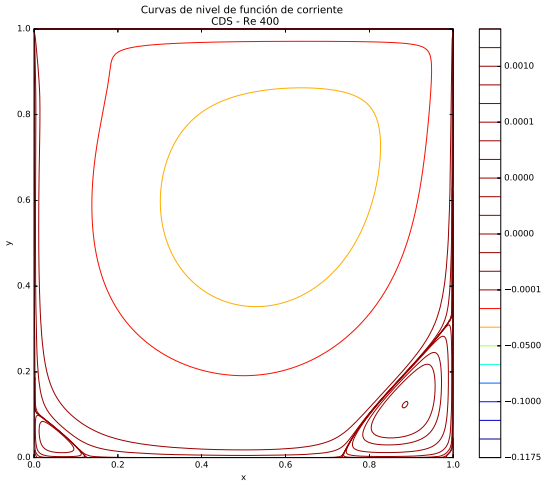

Figura 19: Curvas de nivel de función de corriente con CDS para Re = 400

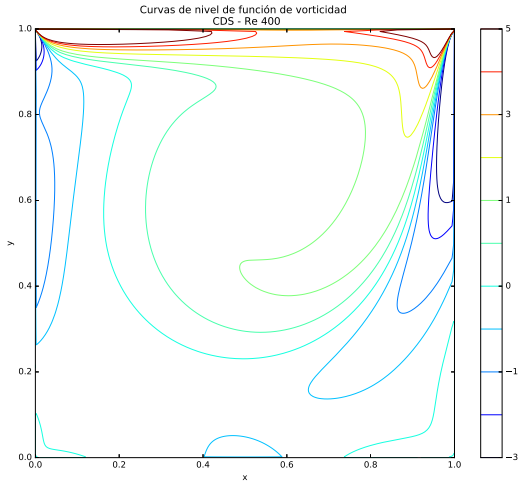

Figura 20: Curvas de nivel de función de vorticidad con CDS para Re = 400

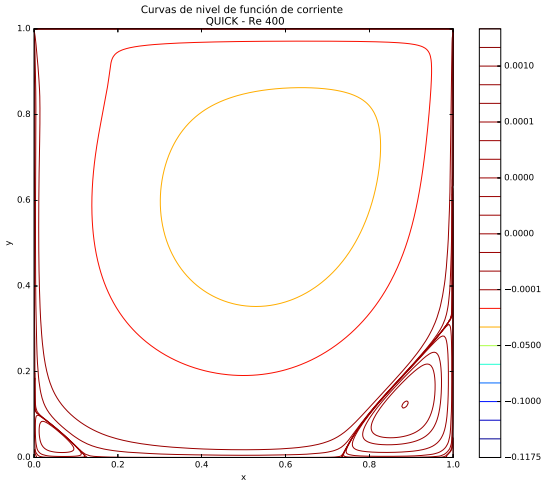

Figura 21: Curvas de nivel de función de corriente con QUICK para Re = 400

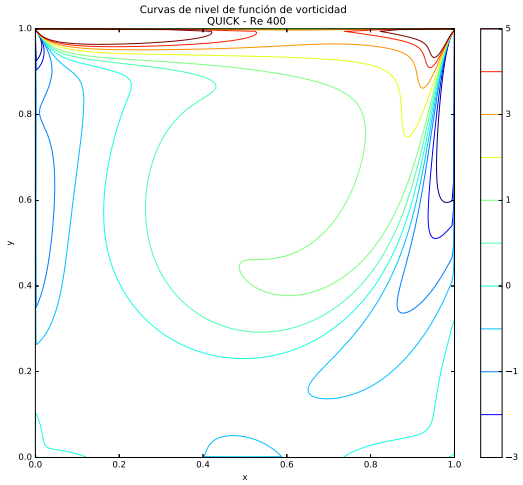

Figura 22: Curvas de nivel de función de vorticidad con QUICK para Re = 400

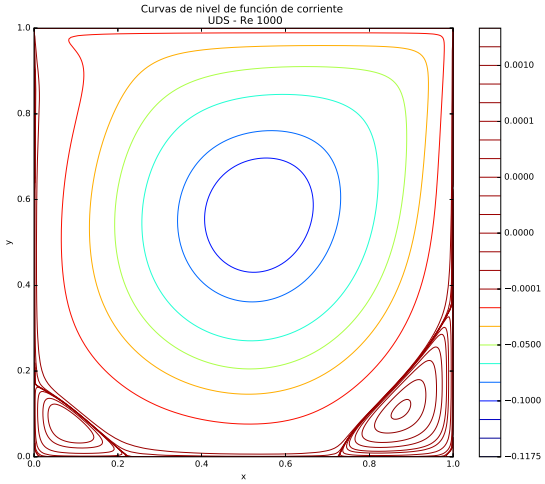

Figura 23: Curvas de nivel de función de corriente con UDS para Re = 1000

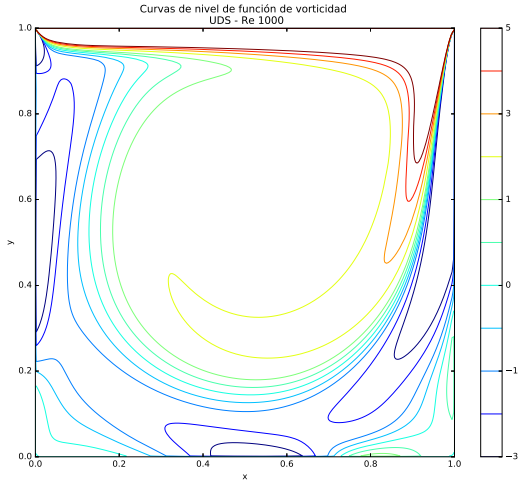

Figura 24: Curvas de nivel de función de vorticidad con UDS para Re = 1000

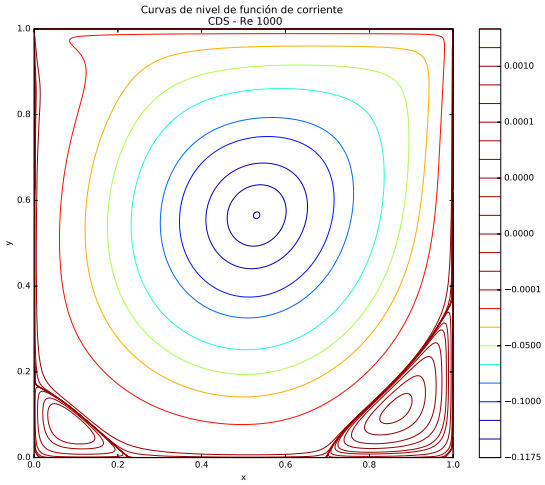

Figura 25: Curvas de nivel de función de corriente con CDS para Re = 1000

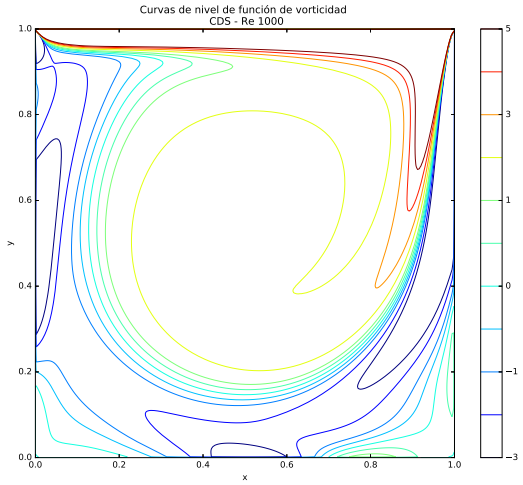

Figura 26: Curvas de nivel de función de vorticidad con CDS para Re = 1000

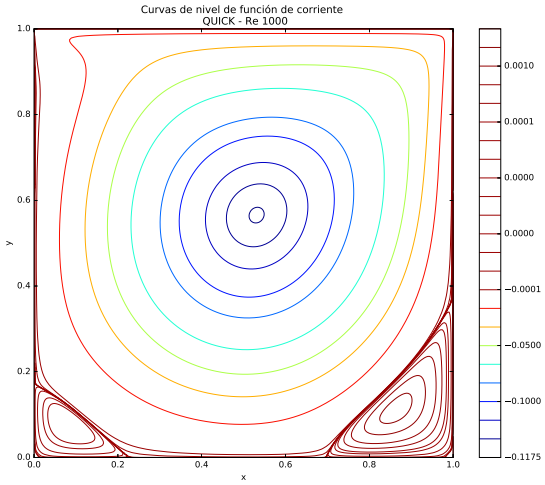

Figura 27: Curvas de nivel de función de corriente con QUICK para Re = 1000

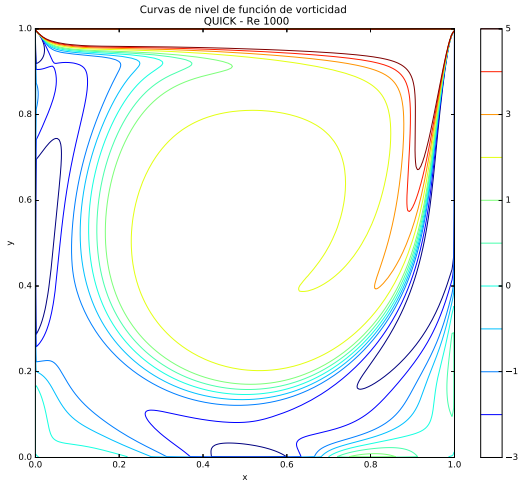

Figura 28: Curvas de nivel de función de vorticidad con QUICK para Re = 1000

7. SIMULACIÓN DE CASO TRANSIENTE

Una simulación del caso transiente es presentado para $Re = 1000$ y $N = 256$. El esquema utilizado fue QUICK debido a las mejoras observadas para el caso estacionario. El tiempo de simulación física fue de 16 segundos. El tiempo de simulación numérica fue de 54 minutos. La cantidad de pasos de tiempo fue 1600, por lo que en promedio cada paso de tiempo llevó 2,025 segundos. Si bien el tiempo no está cerca del tiempo necesario para llegar a *real time* (lo que se aspira obtener, que el tiempo de simulación esté cerca del tiempo del fenómeno físico que se desea estudiar) aún queda mejoras por hacer al código como mejorar la convergencia de los solvers lineales mediante el uso de precondicionadores o bien acelerar la convergencia del método SIMPLE, por ejemplo con estrategias multigrilla.

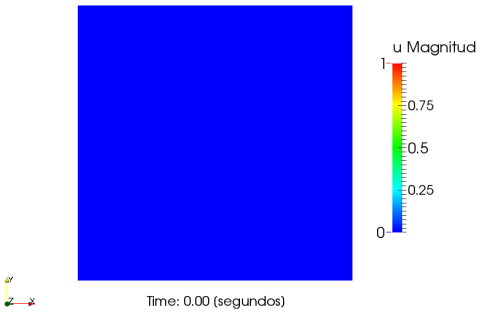

Figura 29: Magnitud de velocidad en t=0.0 segundos.

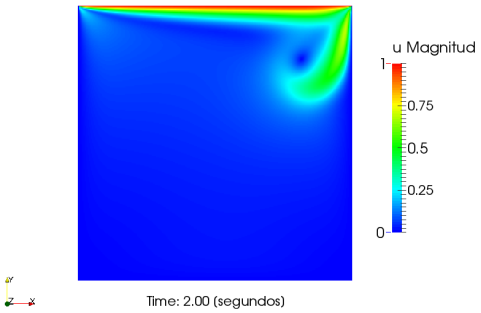

Figura 30: Magnitud de velocidad en t=2.0 segundos.

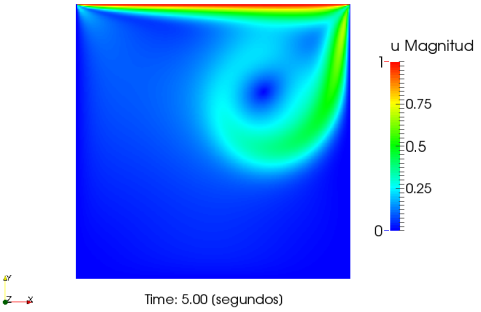

Figura 31: Magnitud de velocidad en t=5.0 segundos.

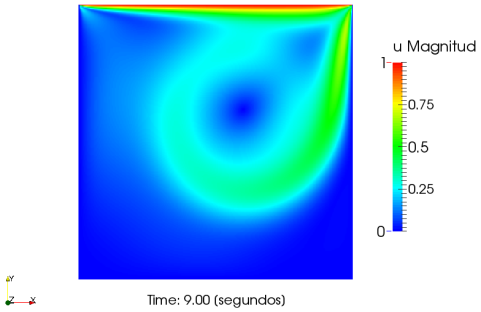

Figura 32: Magnitud de velocidad en t=9.0 segundos.

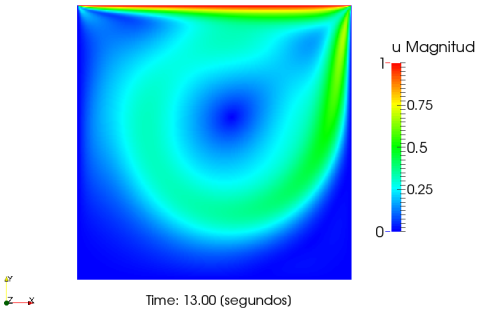

Figura 33: Magnitud de velocidad en t=13.0 segundos.

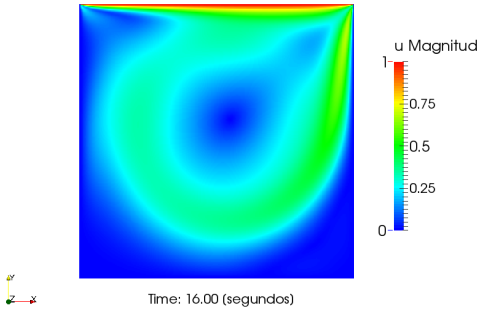

Figura 34: Magnitud de velocidad en t=16.0 segundos.

8. CONCLUSIONES

Se implementó algoritmos para resolver el problema driven cavity flow para arquitecturas multi-GPU aprovechando las capacidades de las GPUs en el cálculo de las operaciones más

costosas en tiempo como es el producto matriz por vector y se obtuvieron soluciones acordes a resultados de la literatura. Los resultados de las pruebas han mostrado que el método que mejor compromiso tiene entre rapidez y precisión ha sido el esquema QUICK. Con respecto a la comunicación de datos, las transferencias de halos se han logrado acelerar mediante el uso de la tecnología GPUDirect2 debido a que mediante esta tecnología MPI puede enviar datos directamente desde arreglos alojados en la memoria de la GPU mejorando así la eficiencia en esta operación ya que no es necesario mantener un arreglo de datos flotantes de doble precisión en el $host$ para realizar las comunicaciones. Como trabajo futuro se extenderá el código a 3D y se implementará mejoras para reducir las iteraciones del método SIMPLE utilizando por ejemplo estrategias basadas en multigrilla.

AGRADECIMIENTOS

Este trabajo ha recibido aporte financiero del Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET, Argentina, grant PIP 11220150100588CO), Agencia Nacional de Promoción Científica y Tecnológica (ANPCyT, Argentina, grants PICT 2660–14, PICT-2015-2904), y Red CYTED 2015 CAD-ING 516RT0512.

Los autores hacen un amplio uso de *Software Libre* como GNU/Linux OS, compiladores GCC/G++, Octave, y *Open Source* software como VTK, ParaView, LaTeX entre muchos otros.

REFERENCIAS

Botella O. y Peyret R. Benchmark spectral results on the lid-driven cavity flow. *Computers & Fluids*, 27(4):421–433, 1998.

Erturk E., Corke T.C., y Gökçöl C. Numerical solutions of 2-d steady incompressible driven cavity flow at high reynolds numbers. *International journal for Numerical Methods in fluids*, 48(7):747–774, 2005.

Ghia U., Ghia K.N., y Shin C. High-re solutions for incompressible flow using the navier-stokes equations and a multigrid method. *Journal of computational physics*, 48(3):387–411, 1982.

Golub G.H. y Van Loan C.F. *Matrix Computations (3rd Ed.)*. Johns Hopkins University Press, Baltimore, MD, USA, 1996. ISBN 0-8018-5414-8.

Gupta M.M. y Kalita J.C. A new paradigm for solving navier–stokes equations: streamfunction–velocity formulation. *Journal of Computational Physics*, 207(1):52–68, 2005.

Kalita J.C. y Gupta M.M. A streamfunction–velocity approach for 2d transient incompressible viscous flows. *International journal for numerical methods in fluids*, 62(3):237–266, 2010.

Kelley C. *Iterative Methods for Linear and Nonlinear Equations*. Frontiers in Applied Mathematics. Society for Industrial and Applied Mathematics, 1995. ISBN 9780898713527.

Kim J. y Moin P. Application of a fractional-step method to incompressible navier-stokes equations. *Journal of computational physics*, 59(2):308–323, 1985.

Leonard B.P. A stable and accurate convective modelling procedure based on quadratic upstream interpolation. *Computer methods in applied mechanics and engineering*, 19(1):59–98, 1979.

Magalhães J.P., Albuquerque D.M., Pereira J.M., y Pereira J.C. Adaptive mesh finite-volume calculation of 2d lid-cavity corner vortices. *Journal of computational physics*, 243:365–381, 2013.

Rhie C. y Chow W.L. Numerical study of the turbulent flow past an airfoil with trailing edge separation. *AIAA journal*, 21(11):1525–1532, 1983.