NUEVA ESTRATEGIA DE PARALELIZACIÓN EXPLÍCITA PARA EL METODO DE RED DE VÓRTICES INESTACIONARIO Y NO-LINEAL

Adrián Barone[a], Luis Ceballos[a,b], Ariel Flores[a] y Sergio Preidikman[a,b,c]

[a] *Facultad de Ingeniería. Universidad Nacional de Río Cuarto. Ruta Nacional 36 Km. 601, 5800 Río Cuarto. Argentina, abarone@ing.unrc.edu.ar, http://www.ing.unrc.edu.ar*

[b] *CONICET – Consejo Nacional de Investigaciones Científicas y Técnicas, Av. Rivadavia 1917, Buenos Aires, Argentina.*

[c] Facultad de Ciencias Exactas Físicas y Naturales, Universidad Nacional de Córdoba, Casilla de Correo 916, 5000 Córdoba, Argentina, *spreidikman@efn.uncor.edu, http://www.efn.uncor.edu*

Palabras Clave: Aerodinámica, Método de red de vórtices inestacionario y no-lineal, Computación en paralelo, Memoria compartida, OpenMP.

Resumen. En este trabajo se presenta una nueva estrategia de paralelización explícita del método de red de vórtices no-lineal e inestacionario (NUVLM). Este método permite modelar correctamente las no-linealidades aerodinámicas asociadas al movimiento inestacionario de cuerpos inmersos en el seno de un fluido "levemente" viscoso.

Para comprender la estrategia de paralelización desarrollada, en la primera parte de este trabajo se describen los aspectos más importantes del NUVLM y se identifican aquellos que son inherentemente paralelizables. La estrategia de paralelización está orientada a mejorar el rendimiento (o *speedup*) de la herramienta numérica mediante la reducción de los tiempos de ejecución de aquellas partes del código computacionalmente más intensivas. Esto se logra distribuyendo la carga de trabajo entre los procesadores/núcleos disponibles. La tarea que actúa como "cuello de botella" ocurre en la generación de las estelas, conocida como etapa de convección; es la más crítica pues consume un altísimo porcentaje del tiempo de ejecución, hecho determinado en un trabajo anterior.

La estrategia de paralelización explicita desarrollada en este esfuerzo consiste en dividir las estelas discretizadas, también llamadas sábanas vorticosas libres, en diferentes zonas y se implementó sobre una arquitectura de memoria compartida utilizando directivas de compilación OpenMP. Se realizaron pruebas para una geometría que se mantuvo sin cambios en su forma y posición durante toda la simulación numérica.

En la sección de resultados se muestran mediciones de tiempo comparando la ejecución de un código computacional utilizando diferente número de procesadores. Estos resultados muestran importantes mejoras en la velocidad de ejecución. Además, se analizan la eficiencia y el *speedup* que presenta la estrategia desarrollada.

1 INTRODUCCIÓN

Este trabajo forma parte de un esfuerzo mayor orientado a desarrollar herramientas numéricas de alta fidelidad para estudiar problemas aeroservoelásticos inestacionarios y fuertemente no-lineales (Ceballos et al., 2008a, b, 2009; 2010, Roccia et al., 2008, 2009, 2010, Verstraete et al., 2009). Para que las herramientas en desarrollo produzcan resultados confiables deben necesariamente emplear un modelo aeroservoelástico, y para ello deben incorporar un modelo aerodinámico, un modelo estructural y un sistema de control trabajando juntos como un único sistema dinámico. En un esfuerzo orientado a desarrollar herramientas computacionales del tipo mencionado se están siguiendo los lineamientos de Preidikman (1998). Integrantes del grupo al que pertenecen los autores de este trabajo han desarrollado previamente, algunas herramientas computacionales con un modelo aerodinámico que implementa el método de red de vórtices no lineal e inestacionario (*non-linear unsteady vortex lattice method* o NUVLM).

En lo que respecta a los requerimientos computacionales, la implementación del NUVLM posee varias ventajas respecto de otros métodos numéricos basados en técnicas propias de la Mecánica de Fluidos Computacional (por ejemplo: elementos finitos, volúmenes finitos, etc.). No obstante, implementaciones "seriales" del NUVLM (comportamiento aerodinámico de vehículos aéreos con configuraciones de alas fijas, con alas que mutan, o con alas batientes) demostraron que el NUVLM requiere recursos computacionales altos. El uso de técnicas tales como recortar las estelas o aprovechar la simetría del problema permiten que las implementaciones computacionales del NUVLM puedan ejecutarse más rápido, a costo de restringir el tipo de problemas que pueden ser atacados (Katz y Plotkin, 2001, Ravetta, 2005). Por esta razón, el camino natural para lograr mayores velocidades de ejecución es desarrollar códigos computacionales que implementen el NUVLM sobre arquitecturas de sistemas distribuidos.

Fritz y Long (2004) presentaron herramientas computacionales que implementan el NUVLM usando técnicas de computación en paralelo sólo sobre el método utilizado para resolver sistemas de ecuaciones algebraicas lineales. Recientemente, los autores de este trabajo desarrollaron e implementaron computacionalmente una estrategia de paralelización explícita que consiste en realizar una descomposición del dominio centrada en los datos de entrada sobre un modelo de arquitectura de memoria compartida (Ceballos et al. 2010). En ese trabajo se analizaron que porciones del algoritmo NUVLM consumen más tiempo de ejecución y que partes son "naturalmente" paralelizables, concluyendo que la etapa más "costosa" es aquella en la que se realiza la convección de partículas de fluido. La idea básica de la estrategia allí presentada consiste en hacer, simultáneamente, la convección de varias partículas de fluido.

El presente trabajo continúa con los lineamientos de Ceballos et al. (2010) y presenta una estrategia de paralelización que propone una nueva manera distribuir las tareas entre diferentes unidades de cómputo. Esta estrategia de paralelización permite atacar las porciones más conflictivas o "cuellos de botella". La estrategia se desarrolla de manera general y está pensada para ser aplicada a casos en los que existen uno o varios cuerpos sumergidos en el seno de un fluido. La implementación computacional de la estrategia se realiza utilizando una arquitectura de memoria compartida, OpenMP (http://www.openmp.org) y Fortran. En este trabajo se muestran resultados obtenidos con un código computacional capaz de simular el comportamiento aerodinámico inestacionario y no lineal de configuraciones simples de alas fijas, y se realiza un análisis de la *performance* del código.

2 EL MÉTODO DE RED DE VÓRTICES

2.1 Modelo aerodinámico

El modelo aerodinámico basado en el NUVLM permite modelar correctamente no-linealidades aerodinámicas asociadas con grandes ángulos de ataque, deformaciones estáticas, y flujos dominados por vorticidad en los que el fenómeno conocido como *vortex bursting* no ocurre. El modelo predice correctamente la emisión de vorticidad desde un cuerpo (o varios), inmerso en el seno de un fluido, hacia el campo del flujo. Esta vorticidad es transportada por el flujo de aire desde el cuerpo hacia el fluido y forma así las estelas. La distribución de la vorticidad en las estelas y la forma de las mismas son, también, parte de la solución del problema. El NUVLM es un método confiable y muy buen predictor de las cargas aerodinámicas inestacionarias y no-lineales.

En flujos sobre superficies sólidas donde el número de Reynolds es alto, se genera vorticidad por efectos viscosos en capas muy delgadas, llamadas capas límites, que están pegadas a las superficies del cuerpo. Los efectos viscosos son responsables de la existencia de las capas límites. Parte de esta vorticidad es emitida desde los bordes filosos de los sólidos y transportada por el fluido, formando las estelas. El campo de velocidades asociado con toda esta vorticidad interactúa con la llamada corriente libre: mientras las condiciones de borde de no-penetración y no-deslizamiento son satisfechas sobre las superficies sólidas generadoras de vorticidad, la vorticidad en las estelas se mueve libremente en el fluido de forma tal que no se produzcan saltos de presión a través de las estelas.

El método de red de vórtices inestacionario esta basado en la idea de representar las capas límites y las estelas mediante sábanas vorticosas. Nos referiremos a estos dos tipos de sábanas vorticosas como ‘sábanas adheridas’ (*bound-vortex sheets*) y ‘sábanas libres’ (*free-vortex sheets*).

El flujo asociado con la vorticidad en la estela cercana al sólido afecta el flujo alrededor del mismo sólido y por lo tanto las cargas actuantes sobre ella. Debido a que la vorticidad presente en las estelas en un instante dado fue generada y convectada desde el ala en un tiempo anterior, las cargas aerodinámicas dependen de la historia del movimiento; las estelas contienen la “historia”. El campo de velocidades, asociado con la vorticidad existente en un punto del espacio, decae con el cuadrado de la distancia a dicho punto; en consecuencia, a medida que la vorticidad en la estela va siendo transportada flujo abajo, su influencia decrece y por lo tanto se dice que “el historiador” va perdiendo memoria.

En las secciones siguientes se presentan algunos aspectos importantes de la implementación del modelo empleado en las herramientas computacionales usadas en este trabajo. Para más detalles acerca del NUVLM pueden consultarse los trabajos de Konstadinopoulos et al. (1981) o Preidikman (1998).

2.2 Implementación computacional

En el NUVLM se reemplaza la sábana vorticosa adherida a la superficie del sólido y la sábana vorticosa libre por una red de segmentos vorticosos rectos y de longitud finita. La red de segmentos vorticosos de la sábana adherida divide la superficie del sólido en elementos de área que tienen forma de rectángulo, de paralelogramo, o en el caso más general de un cuadrilátero con todos sus lados diferentes.

La implementación numérica de este modelo requiere que la geometría del cuerpo se defina de una manera particular: las distintas partes componentes del cuerpo son representadas mediante un conjunto de superficies. Estas superficies se definen mediante mallas formadas por paneles y nudos. En la Figura 1 se presenta una malla que define una geometría muy simple referida a una placa plana y rectangular, mientras que en la Figura 2 se muestra una geometría más compleja que corresponde a un de vehículo aéreo no tripulado (UAV) con una configuración de alas unidas.

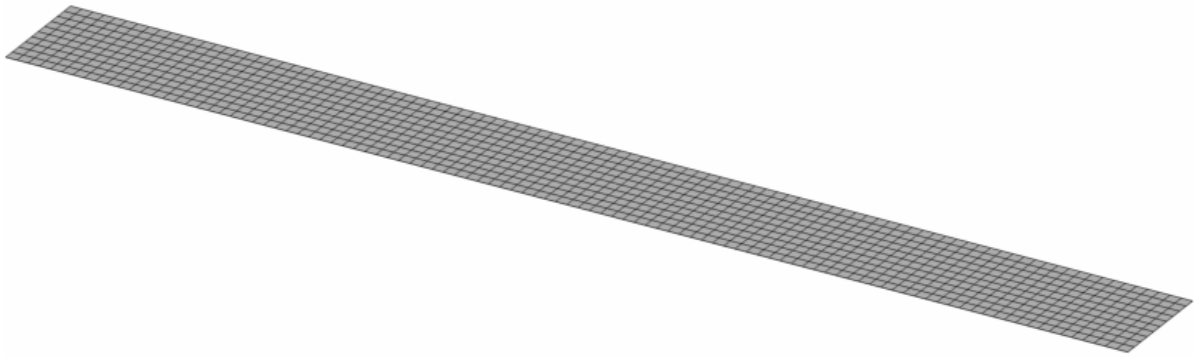

Figura 1: Malla de paneles que represente una geometría simple: placa rectangular.

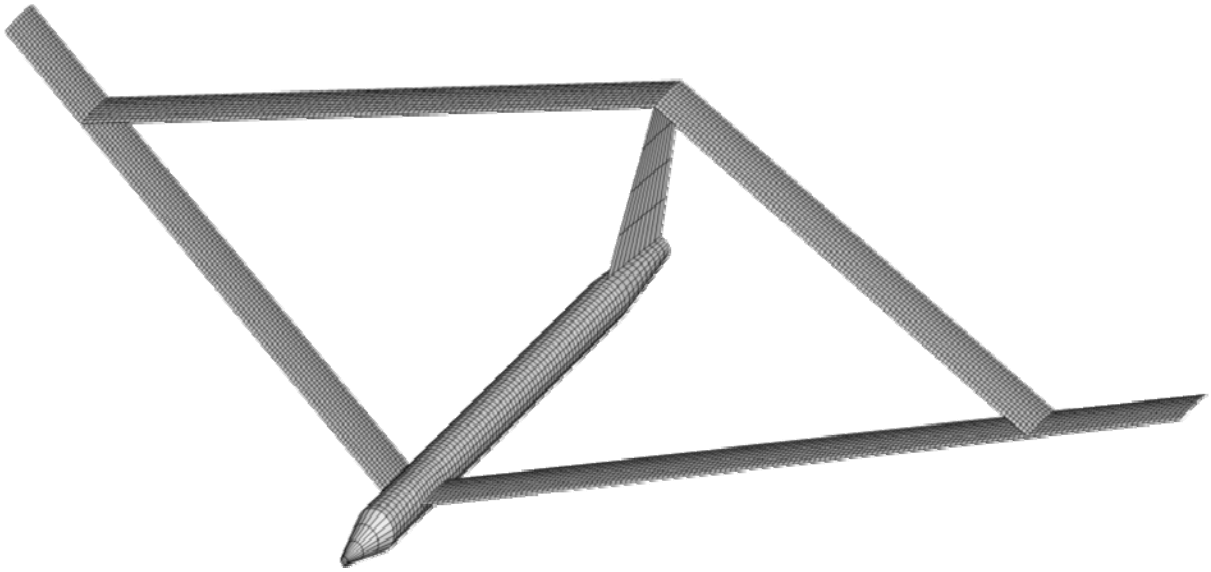

Figura 2: Malla de paneles generada para una configuración de UAV con alas unidas (Ceballos et al., 2008a).

En cada uno de los paneles se define un punto de control que ese encuentra en el centroide de área del elemento. En el punto de control se impone la condición de no penetración en su versión discretizada. Esta imposición resulta luego en un sistema de ecuaciones algebraico lineal. La resolución de este sistema de ecuaciones permite obtener el valor de circulación que esta asociado a cada uno de los segmentos vorticosos utilizados para describir aerodinámicamente la superficie del UAV.

Con posterioridad, se efectúa el cálculo de las cargas aerodinámicas sobre la superficie sustentadora. Para cada elemento, se debe hallar primero la presión en el punto de control y luego multiplicarla por el área del elemento y por el vector unitario normal. La versión inestacionaria de la ecuación de Bernoulli se usa para calcular la distribución de la presión sobre la superficie de las alas. Finalmente, se suman las fuerzas y los momentos de dichas fuerzas actuantes en todos los elementos.

2.3 Generación de las estelas

En esta subsección se muestra un ejemplo de generación de estelas para el caso de un ala que comienza a moverse impulsivamente. Para generar las estelas, hay que realizar una convección de las partículas de fluido que se encuentran sobre los bordes filosos del ala: borde de fuga y punteras. La convección se hace calculando el desplazamiento de las partículas que se produce en un intervalo de tiempo Δt.

En la Figura 3 se muestra un esquema de la evolución de las estelas para los primeros dos pasos de tiempo de una simulación. El ejemplo mostrado en la figura, corresponde al de un ala rectangular discretizada con 40 paneles (4 paneles a lo largo de la cuerda por 10 paneles a lo largo de la envergadura).

En la Figura 3a se muestra el instante inicial $t = 0$, en el cual no existe la estela. En esa figura se destacan con puntos negros las partículas que están sobre los bordes filosos que emiten estela. A estas partículas se les realiza la convección durante el primer paso de tiempo, y como resultado se obtiene una fila de paneles que conforma la estela en el instante $t = \Delta t$ (ver Figura 4b). Durante el segundo paso de tiempo, se realiza el convección sobre las partículas que pertenecen a la estela y sobre las partículas que están sobre el borde de fuga y las punteras (partículas destacadas con puntos negros en la Figura 3b). El resultado al final del segundo paso de tiempo ($t = 2\Delta t$) está esquematizado en la Figura 3c, en la cual se muestran las dos filas de las estela obtenidas por 'convección'. En los pasos siguientes, se procede de manera similar, esto es, realizando el proceso de convección sobre todas las partículas que pertenecen a la estela, y sobre las partículas que están sobre los bordes filosos que emiten estela.

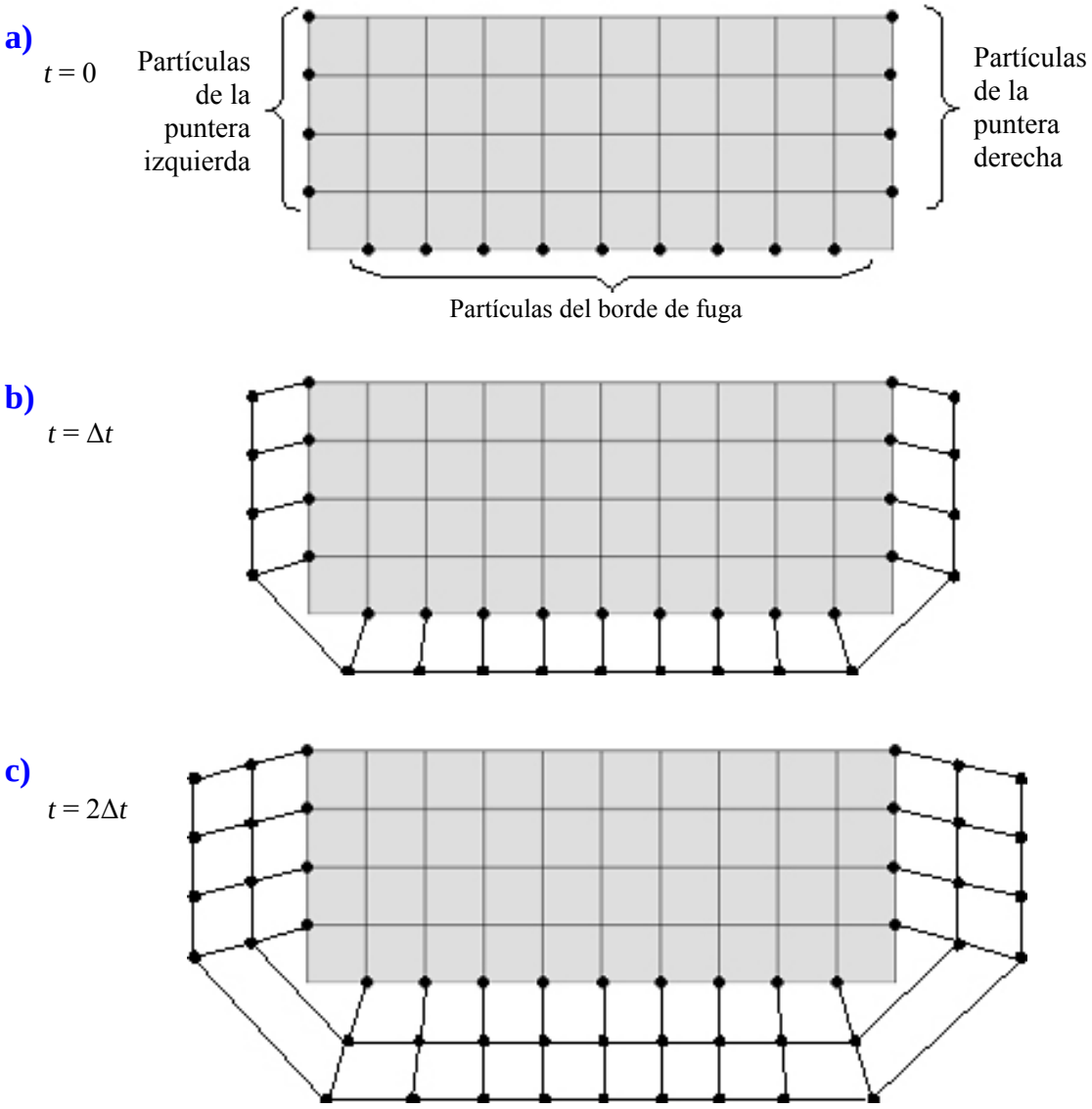

Figura 3: Esquema de la evolución de las estelas en los primeros pasos: a) $t = 0$, b) $t = \Delta t$ y c) $t = 2\Delta t$.

En la Figura 4 se muestran ejemplos de estelas obtenidas con códigos desarrollados con anterioridad por el grupo de investigación al que pertenecen los autores de este trabajo (Roccia et al., 2009; Ceballos et al., 2008b). En la Figura 4a se muestra la estela generada luego de 100 pasos de simulación para el caso una placa plana con un ángulo de ataque geométrico de 5 grados y un alargamiento con valor igual a 10. En la Figura 4b se muestra la estela generada luego de 100 pasos de simulación, para el caso de una configuración de UAV con alas unidas con un ángulo de ataque geométrico de 5 grados. En la Figura 4c se graficó la estela generada por un par de alas batientes durante la simulación del 75 % del primer ciclo de batimiento. En este último caso, se ha coloreado en azul a los puntos de las estelas desprendidos desde los bordes de fuga y en rojo los puntos desprendidos desde las punteras de las alas batientes.

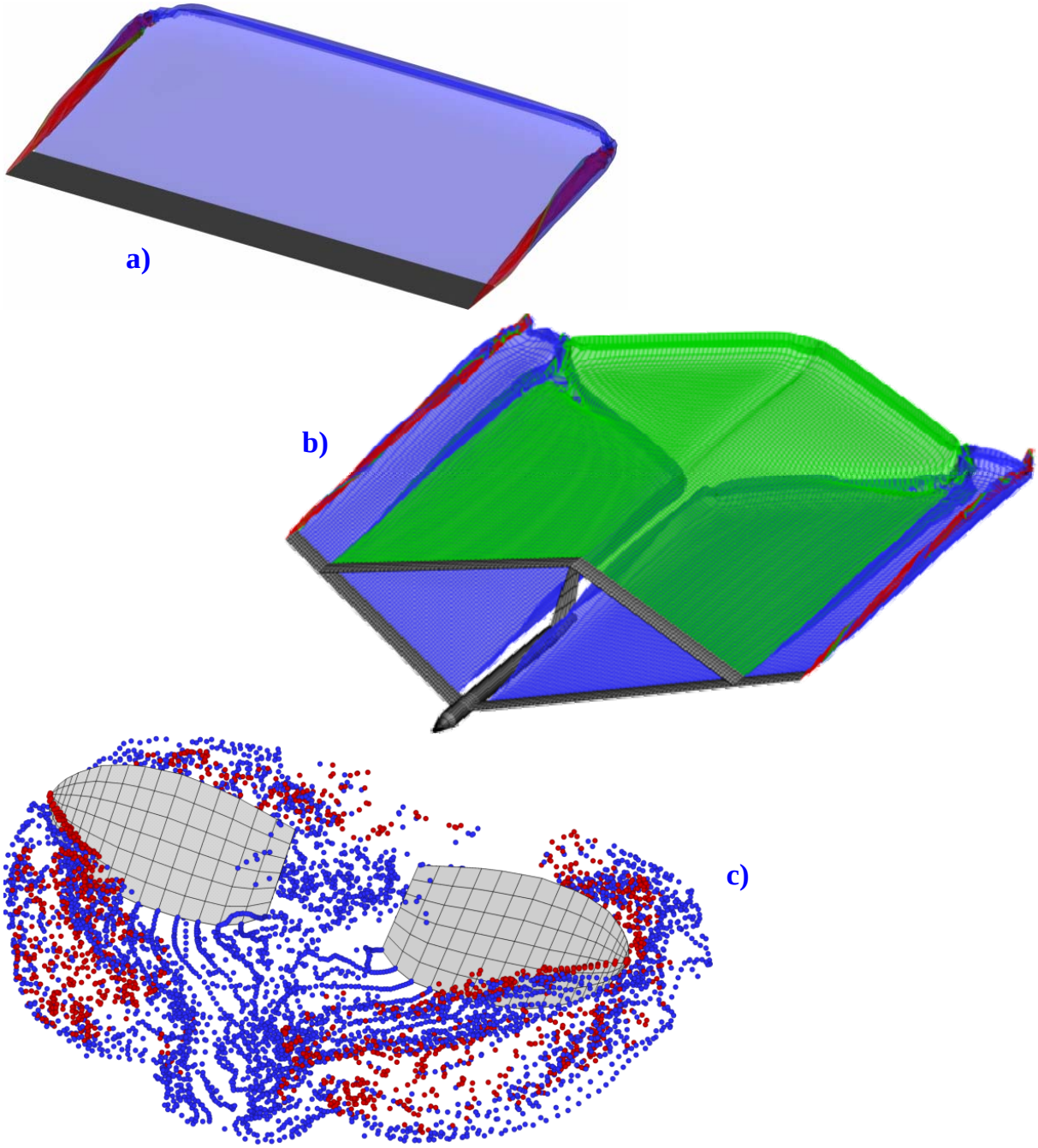

Figura 4: Ejemplos de estelas generadas para a) una geometría simple, b) una configuración de UAV de alas unidas y c) una configuración de alas batientes.

2.4 Algoritmo del NUVLM

A continuación, se lista un algoritmo del NUVLM que sido tomado como base para las distintas implementaciones computacionales que se han desarrollado en este esfuerzo (entre paréntesis se indican los nombres de las subrutinas más significativas del código):

1. *Lectura y ordenamiento de datos.*
2. *Cómputo de la matriz de coeficientes de influencia aerodinámicos.*
3. *Descomposición LU de la matriz de coeficientes de influencia aerodinámicos.*
4. *Cómputo del lado derecho del sistema de ecuaciones algebraicas lineales.*
5. *Solución del sistema de ecuaciones algebraicas lineales: cálculo de la circulación.*
6. *Para cada paso de tiempo se realizan los siguientes sub-pasos de cálculo:*
   - 6.1. *Convección de las partículas de fluido* (**`convect`**).
     - 6.1.1. *Cálculo de las influencias de la sábana adherida discretizada sobre las partículas de fluido presentes en la sábana libre discretizada* (**`vbnd`**).
     - 6.1.2. *Cálculo de las influencias de la sábana libre discretizada sobre las partículas de fluido presentes en la sábana libre discretizada* (**`vwakels`**).
   - 6.2. *Actualización de la geometría y/o cinemática.*
   - 6.3. *Cómputo de la matriz de coeficientes de influencia aerodinámicos.*
   - 6.4. *Descomposición LU de la matriz de coeficientes de influencia aerodinámicos.*
   - 6.5. *Cómputo del nuevo lado derecho del sistema de ecuaciones algebraicas lineales* (**`rhs`**).
     - 6.5.1. *Cálculo de las influencias de la sábana libre discretizada sobre los puntos de control de los paneles de la sábana adherida discretizada* (**`vwakels`**).
   - 6.6. *Solución del nuevo sistema de ecuaciones algebraicas lineales.*
   - 6.7. *Cómputo de los coeficientes de presión y cargas* (**`loadsls`**).
   - 6.8. *Almacenamiento de los resultados obtenidos en el paso de tiempo actual.*

Los puntos 6.2 y 6.3 de este algoritmo no se realizan cuando no hay cambios en la geometría y/o en la posición del cuerpo, durante el transcurso del tiempo.

3 ESTRATEGIAS DE PARALELIZACIÓN

En esta sección se presenta una estrategia que tiene como fin incrementar la velocidad de ejecución de un código computacional que implementa el NUVLM. La estrategia consiste en una paralelización explícita del código computacional en la zona que consume más tiempo de ejecución, la etapa limitante de la velocidad de ejecución o "cuello de botella".

3.1 Identificación de los "cuellos de botella"

Las experiencias de medición de tiempos de ejecución insumidos por algunos códigos computacionales que implementan el NUVLM, muestran que el mayor consumo de tiempo esta asociado al proceso de convección realizado para generar la estela que se desprende desde los bordes filosos de un cuerpo sumergido en el seno de un fluido.

La anterior afirmación se ve corroborada realizando un *profiling* o análisis dinámico del código. Este tipo de análisis se lleva a cabo con un programa de computación, conocido como *profiler*. Este tipo de programa permite recolectar información acerca del comportamiento de algún código en particular mientras el mismo se está ejecutando. En el presente trabajo se utilizó un *framework* denominado Valgrind (http://valgrind.org/), el cual ofrece distintas herramientas entre las cuales hay un *profiler* y un programa que permiten visualizar gráficamente la información recolectada durante la ejecución del código que se está analizando. Valgrind es un

software libre y de código abierto disponible para ser usado bajo la GNU General Public License, versión 2, que puede utilizarse en sistemas operativos basados en GNU Linux.

Luego de ejecutar el código que implementa el NUVLM sobre el *profiler*, se postprocesa la información del comportamiento del código con la herramienta gráfica *Kcachegrind* (http://kcachegrind.sourceforge.net). Con Kcachegrind es posible obtener un diagrama como el que se muestra en la Figura 5. Ese diagrama indica las dependencias de las rutinas más significativas del programa que implementa el NUVLM (en su versión secuencial), como así también, el número de llamadas a las distintas rutinas y la cantidad de instrucciones ejecutadas en porcentajes.

En la Figura 5 puede observarse que la rutina **`main`** y todas las rutinas que dependen de ella, ejecutan el 100 % de las instrucciones del programa. Directamente dependiendo de **`main`** se encuentran las rutinas **`loadls`**, **`convect`** y **`rhs`**, correspondiéndoles respectivamente un 12, un 65 y un 21 % de las instrucciones ejecutadas. El porcentaje restante para completar el 100 %, corresponde a instrucciones que son propias de la rutina **`main`** que son llamadas a otras rutinas que se ejecutan. Esas instrucciones y rutinas no son mostradas en el esquema de la Figura 5, por simplicidad y debido a que el porcentaje de su participación es poco significativo. A un lado de cada una de las flechas, se pueden observar dos números que representan: el porcentaje, antes mencionado, y el número de veces que la rutina apuntada es llamada.

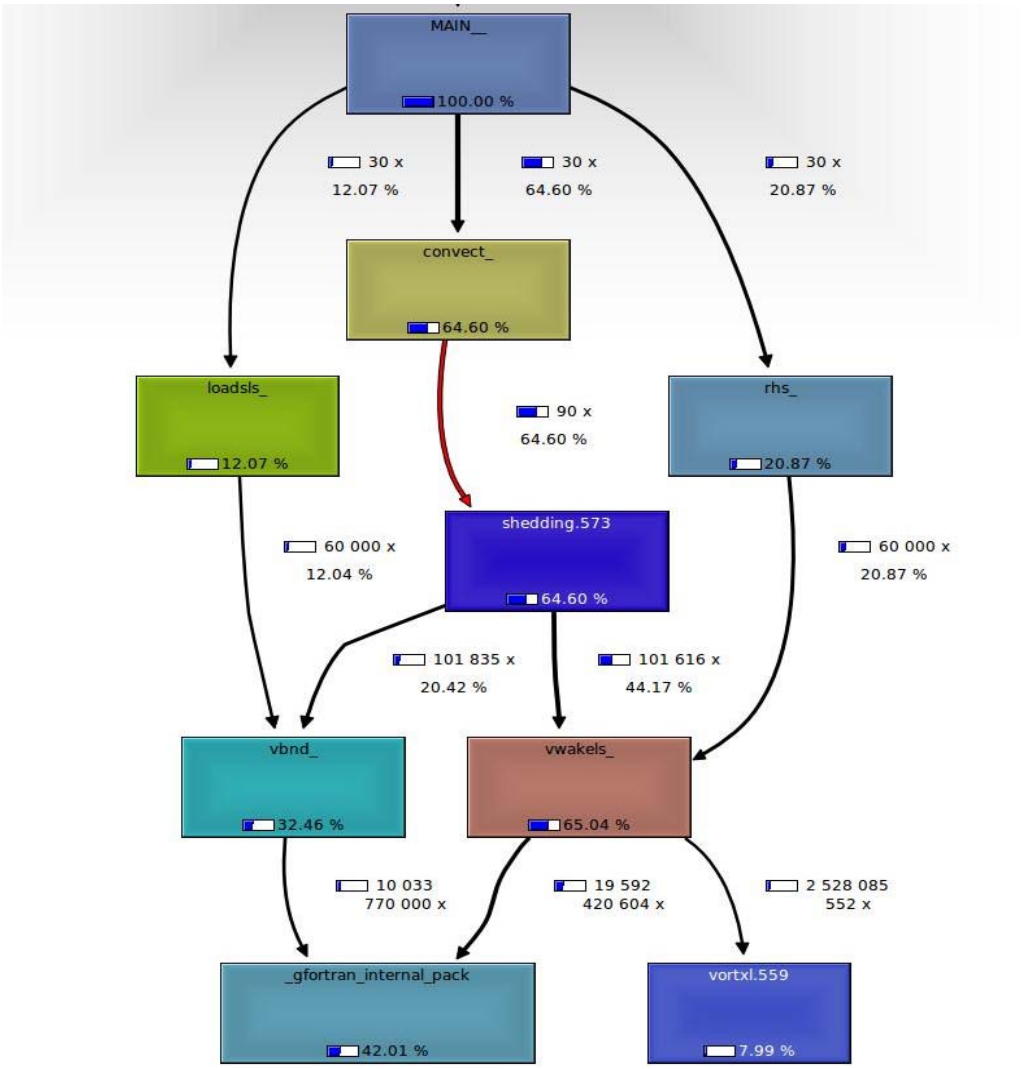

Figura 5: Esquema de las dependencias de rutinas de la versión secuencial del programa que implementa el NUVLM

La flecha que se destaca en color rojo en la Figura 5, representa las interacciones de las rutinas **convect** y **shedding**. En la valores que aparecen junto a esta flecha puede observarse que la rutina **shedding** es llamada 90 veces por la rutina **convect** (**convect** es llamada por **main** 30 veces y **convect** realiza 3 llamadas a **shedding** cada vez que se ejecuta).

La información que acompaña a las flechas que relacionan los bloques que se encuentran en la parte inferior del esquema, no presentan porcentajes como el resto de las flechas del esquema. Esto se debe a que la cantidad de llamadas a las rutinas son demasiado grandes y el valor de la cantidad está escrito en doble línea, esto significa que, por ejemplo, **vortxl** es llamado por **vwakels** 2.528 millones de veces.

Observando la cantidad de llamadas realizadas a las distintas rutinas, vemos que la rutina que interviene en la generación de la estela, **vwakels**, toma el 65 % de la ejecución del programa, siendo llamada por **shedding** unas 100 mil veces y por **rhs** unas 60 mil veces. A su vez, **vwakels** realiza una cantidad de llamadas a las rutinas internas de Fortran del orden de 19 mil millones de veces. Esto demuestra que la rutina **vwakels** es el "cuello de botella" del código computacional.

En la Figura 6 se presentan con mayor detalle las rutinas que invocan a **vwake**, las rutinas llamadas por ella, y el número de veces que lo hace.

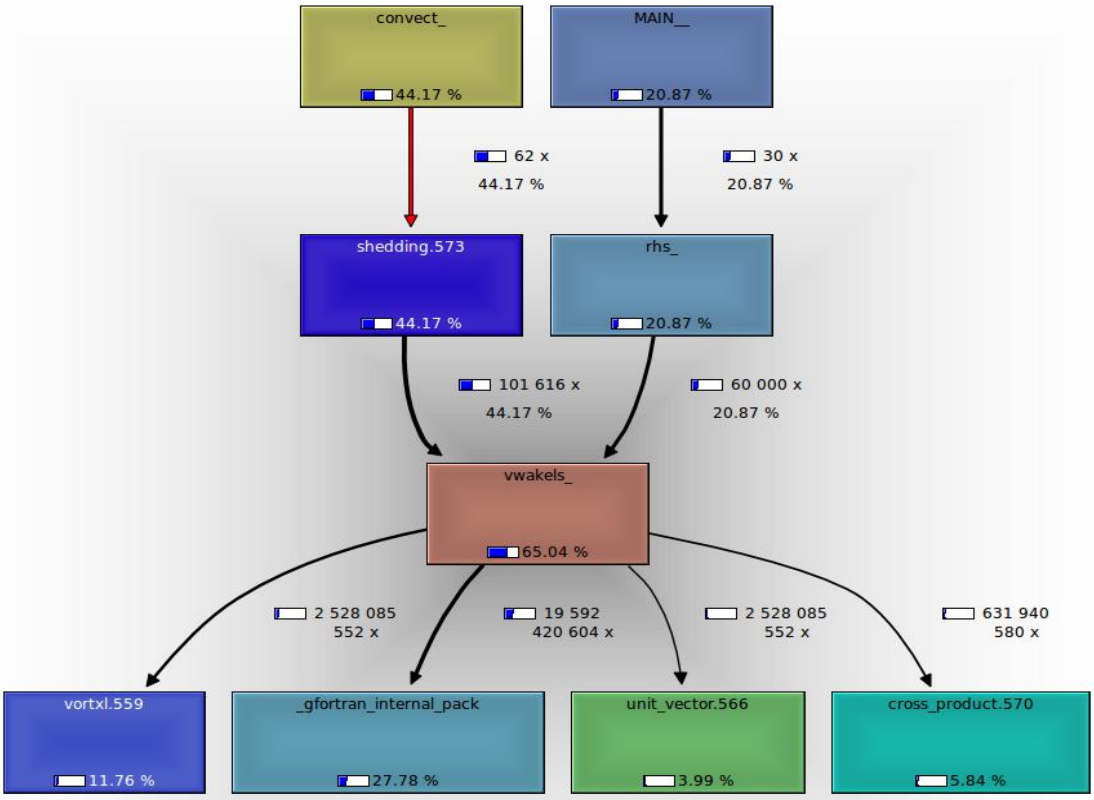

Figura 6: Módulo **vwakels** y las rutinas que de ella dependen

El código fuente secuencial de la rutina **vwake** contiene originalmente tres estructuras iterativas que permiten calcular la evolución espacial de la estela en las punteras y el borde de fuga del ala con una configuración simple. El número de iteraciones en cada caso esta dado por la cantidad de paneles de la estela. Se presentan dos iteraciones adicionales para el cálculo de los paneles de las esquinas. En la implementación de la estrategia de paralelización presentada en este trabajo, se realizaron cambios mínimos sobre el código fuente original, minimizándose así las horas de programación.

3.2 Influencia de la cantidad de pasos de simulación

A continuación se presentan resultados que muestran como influye la cantidad de pasos de tiempo de las simulaciones sobre la distribución porcentual del consumo de tiempo. Los resultados mostrados corresponden a ejecuciones del programa secuencial. En todas las ejecuciones se observó que el proceso de convección de la estela insume mucho más tiempo que todo el conjunto de las otras tareas.

En la Tabla 1 se muestran resultados de mediciones de tiempo efectuadas sobre ejecuciones del programa secuencial, para simulaciones de 25, 50, 75 y 100 pasos de tiempo. La geometría utilizada en estas ejecuciones es siempre la misma y posee las características de la malla de 10x200 paneles. Puede observarse que la distribución de los tiempos consumidos por las tareas cambia según la cantidad de pasos de simulación realizados. La tarea de convección, con solo 25 pasos de simulación ya insume el 58 % del total. Esto crece de manera importante a medida que crece la cantidad de pasos, alcanzado un 87 % para 100 pasos. Este incremento del porcentaje de tiempo insumido por la tarea de convección, trae consigo un decremento en la participación porcentual en el tiempo total por parte de las otras tareas. La segunda tarea en importancia en cuanto a consumo de tiempo presenta una caída desde un 22 a un 11 % al incrementar de 25 a 100, la cantidad de pasos de simulación.

Cantidad de pasos de simulación →		25	50	75	100
Tiempo total de ejecución en horas →		0.28	1.35	3.85	9.37
Generar la matriz de coef. de influencia	%	2.04	0.40	0.14	0.06
Generar los lados derechos		22.04	17.01	13.23	10.73
Resolver el sistema de ecuaciones		3.10	0.63	0.23	0.11
Realizar la convección de las estelas		58.72	76.39	83.50	87.31
Calcular las cargas		14.01	5.52	2.88	1.76

Tabla 1: Tiempo insumido por el programa usando distintas cantidades de pasos de tiempo.

En la Figura 7 se amplía la información brindada por la Tabla 1. Allí se muestra de manera gráfica como influye la cantidad de pasos de simulación en la distribución de tiempos consumidos por las diferentes rutinas o partes del código computacional. La cantidad de pasos de simulación se ha variado desde uno a cien. En la figura, puede observarse claramente que para pasos de simulación mayores a 10, la etapa de convección comienza a ser la etapa predominante en el consumo de tiempo.

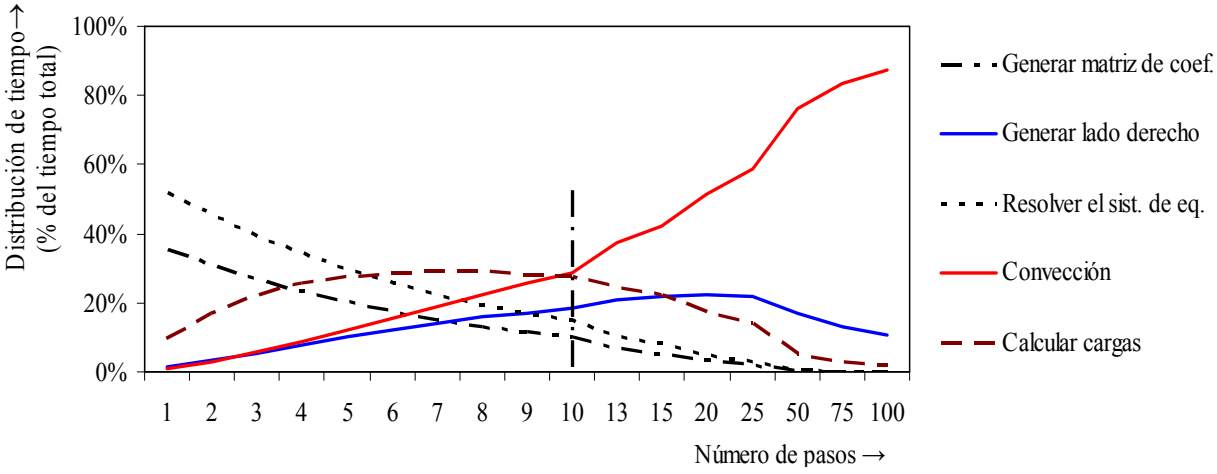

Figura 7: Influencia de la cantidad de pasos de simulación sobre la distribución de tiempos.

3.3 Estrategia de paralelización

La estrategia de paralelización explicita desarrollada en este esfuerzo consiste en: *i*) dividir las estelas o sábanas vorticosas libres discretizadas en diferentes zonas; *ii*) determinar, por separado y en simultáneo, cual es el aporte de cada una de esas zonas a la convección (velocidades asociadas) de una partícula de fluido; y por último *iii*) sumar los aportes de todas las zonas para obtener la resultante de la velocidad asociada para llevar a cabo la convección de cada partícula de fluido.

En la Figura 8 se esquematizan ejemplos de la aplicación de esta estrategia de paralelización sobre la geometría de un ala simple para: *a*) 2 procesadores, *b*) 3 procesadores y *c)* 4 procesadores. Las iteraciones a paralelizar están dadas por la cantidad de paneles de la sabana libre. La cantidad total de iteraciones se divide en grupos aproximadamente iguales, asignándole a cada procesador uno de estos grupos. En la Figura 8 se representaron estos grupos con distintos colores asociándolos a un hilo de procesamiento. En los tres casos presentados en la figura, se muestra la evolución de la estela al finalizar 4 pasos de simulación.

En cada una de las estelas mostradas en las Figuras 8a, 8b y 8c puede observarse que la estela se desprende desde distintas partes de la geometría del ala: las esquinas del ala, las punteras y el borde de fuga. Las estelas generadas a partir de las esquinas de las alas son casos de simples "fajas" de paneles uno a continuación de otro, y la cantidad de esos paneles es pequeña en casos generales. En los casos de las estelas desprendidas desde las punteras y el borde de fuga del ala, la cantidad de paneles contenidos es importante, luego, en cada una de esas partes de la estela se aplica la división de en zonas propuesta por la estrategia presentada en este trabajo.

En el ejemplo esquematizado, las sábanas vorticosas fueron divididas en distintas zonas dependiendo de la cantidad de procesadores: cada una de esas zonas, diferenciadas por el número de panel, es entregada a un procesador. Los aportes a la convección de cada zona son computados al mismo tiempo por diferentes hilos de ejecución.

Esta estrategia posee generalidad, podrían considerarse *n* zonas a procesar con *n* hilos de ejecución, y permite efectuar un adecuado balanceo en la carga de los hilos de ejecución si las *n* zonas se eligen de manera tal que todas posean un tamaño similar, esto es, que tengan en lo posible la misma cantidad de paneles.

4 RESULTADOS

En esta sección se presentan resultados de mediciones de tiempos hechas sobre un código computacional actualmente en uso, que implementan el NUVLM y que ha sido desarrollado por integrantes del grupo de trabajo al que pertenecen los autores. Este programa es una versión que aplica el método de red de vértices sobre una geometría elemental.

El programa ha sido escrito con Fortran 95 y compilado con GNU Fortran 4.4.1 para ser ejecutados en un sistema operativo Linux. El programa posee una paralelización explícita y fue desarrollado utilizando las bibliotecas de OpenMP que están embebidas en el compilador de Fortran utilizado.

Las ejecuciones se realizaron en una computadora de escritorio con una memoria RAM DDR2 de 2 Gb con un bus de 800 MHz y un procesador con una velocidad de reloj de 2.4 GHz, con tecnología de 4 núcleos, un bus del sistema de 1066 MHz y memoria cache L2 de 8Mb. El sistema operativo, Linux, fue instalado mediante una distribución Gentoo sin entorno grafico para evitar cualquier retardo en la ejecución.

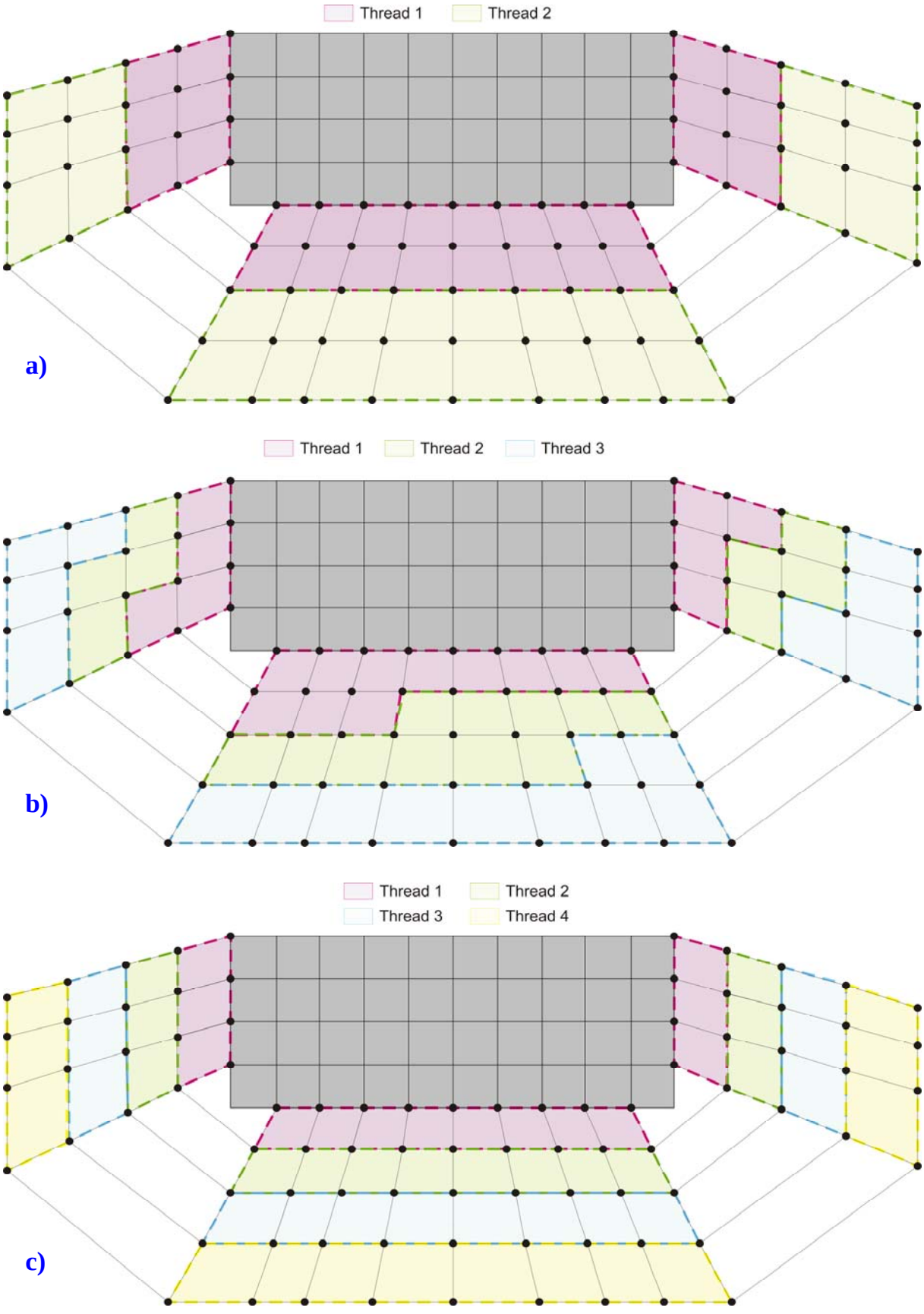

Figura 8: Esquema que ilustra la estrategia de paralelización sobre una placa para: a) 2 procesadores, b) 3 procesadores y c) 4 procesadores.

4.1 Implementación de la estrategia

En esta sección se presentan resultados obtenidas al ejecutar los programas. Se realizaron ejecuciones utilizando 1, 2, 3 y 4 hilos de ejecución. Los tiempos totales se expresan en horas.

En la Tabla 2 se muestran resultados obtenidos al medir el tiempo sobre distintas ejecuciones del programa. La geometría utilizada en estas ejecuciones es una malla de 10x200 paneles. En la última fila de resultados, se muestran variaciones porcentuales de los tiempos consumidos por las ejecuciones, referidas al tiempo consumido por la versión ejecutada utilizando un solo procesador.

La utilización de versiones paralelizadas con 2, 3 y 4 hilos de ejecución muestra una importante disminución en los tiempos totales de ejecución (entre un 43 y un 65 %).

Número de hilos de ejecución →	1	2	3	4
Tiempo total de ejecución en horas →	9,37	5,35	3,98	3,32
Variación respecto de la versión con un procesador →		4 3 %	57 %	65 %

Tabla 2: Tiempo insumido con el programa usando distintas cantidades de hilos de ejecución.

En la Figura 9 y Figura 10 se muestran resultados obtenidos al usar el *profiler* con la versión paralelizada del código que implementa el NUVLM. En la Figura 9 se muestra que el porcentaje de instrucciones ejecutadas por **`vwakels`** descendió a poco menos del 33 %. En la figura, está representada una nueva rutina, **`vwakels.omp`**, que representa la porción de código paralelizada de la rutina **`vwakels`**. De esta forma, el *profiler* diferencia entre las instrucciones ejecutadas secuencialmente y las que fueron paralelizadas mediante instrucciones de OpenMP.

Al paralelizar, se quitan instrucciones del hilo principal y se las reparte entre los otros hilos de ejecución, disminuyendo el tiempo de procesamiento de esa sección del programa. Es por ello que al analizar los resultados mostrados en la Figura 9, solo podemos compararlo con la ejecución de su versión secuencial equivalente, pero dicho diagrama no muestra donde "han ido" las instrucciones no ejecutadas por el hilo principal.

Por esto último, en la Figura 10 se destaca el bloque **`vwakels.omp`**, donde se observa que esta rutina es llamada no solo por el hilo maestro, representado por las llamadas hechas por **`vwakels`**, sino también por los otros tres hilos de ejecución, representados por la rutina **`start_thread`** que es la encargada de iniciar la ejecución de los hilos paralelos.

Así mismo, también podemos ver que del total de instrucciones paralelizadas, un 25 % de las instrucciones llegan desde el hilo maestro, y el 75 % restante por los otros 3 hilos, dando una idea de un buen balance de carga entre los distintos hilos de ejecución. Esto es deseable ya que minimiza las demoras por espera entre los procesadores.

Por otra parte, en la Figura 9 podemos observar que el mayor porcentaje de instrucciones esta depositado sobre la rutina **`vbnd`**, que es la que calcula las influencias de la estela adherida sobre las partículas de fluido presentes en la estela, transformándose en el nuevo cuello de botella del programa.

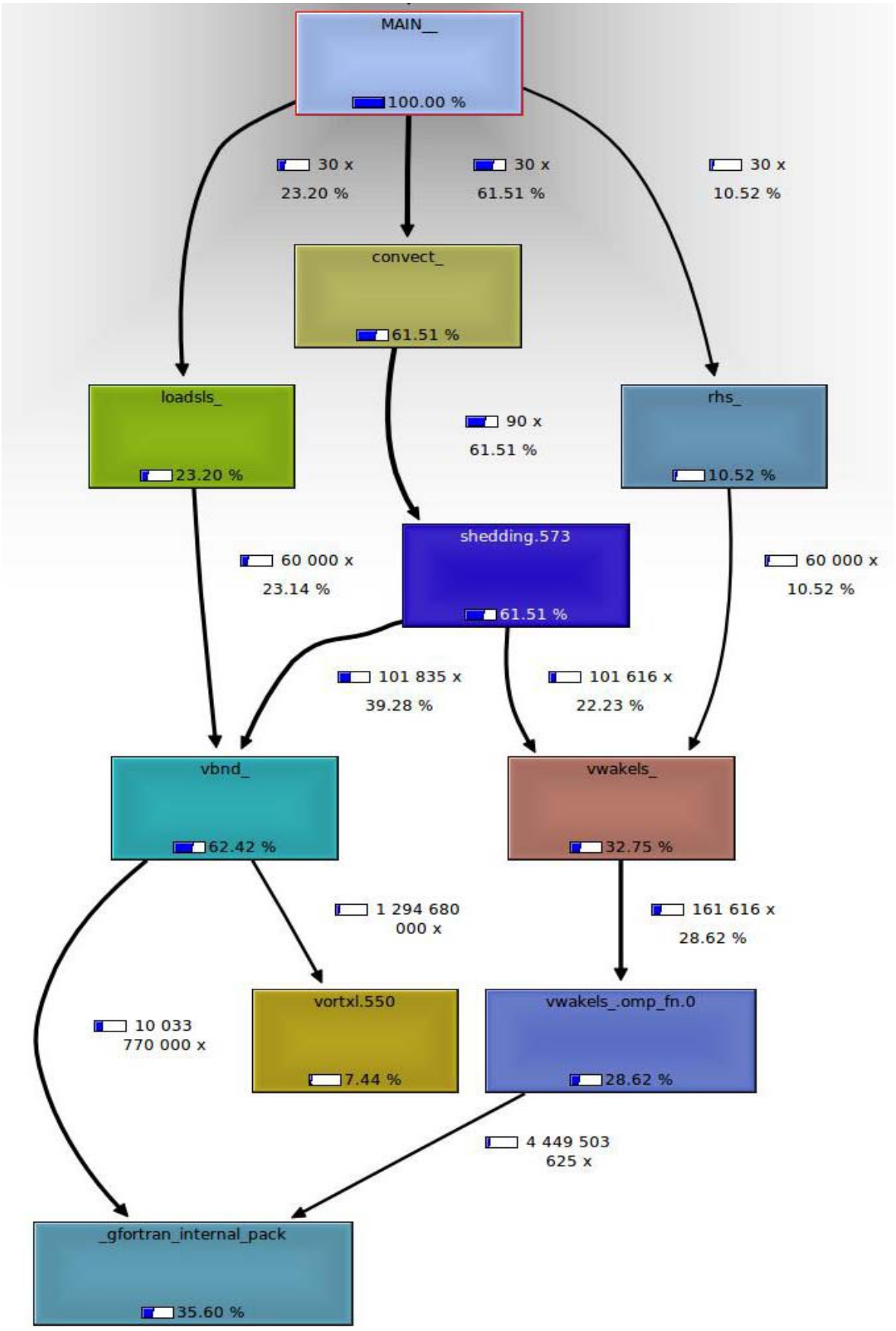

Figura 9: Esquema de la versión paralelizada del programa que implementa el NUVLM.

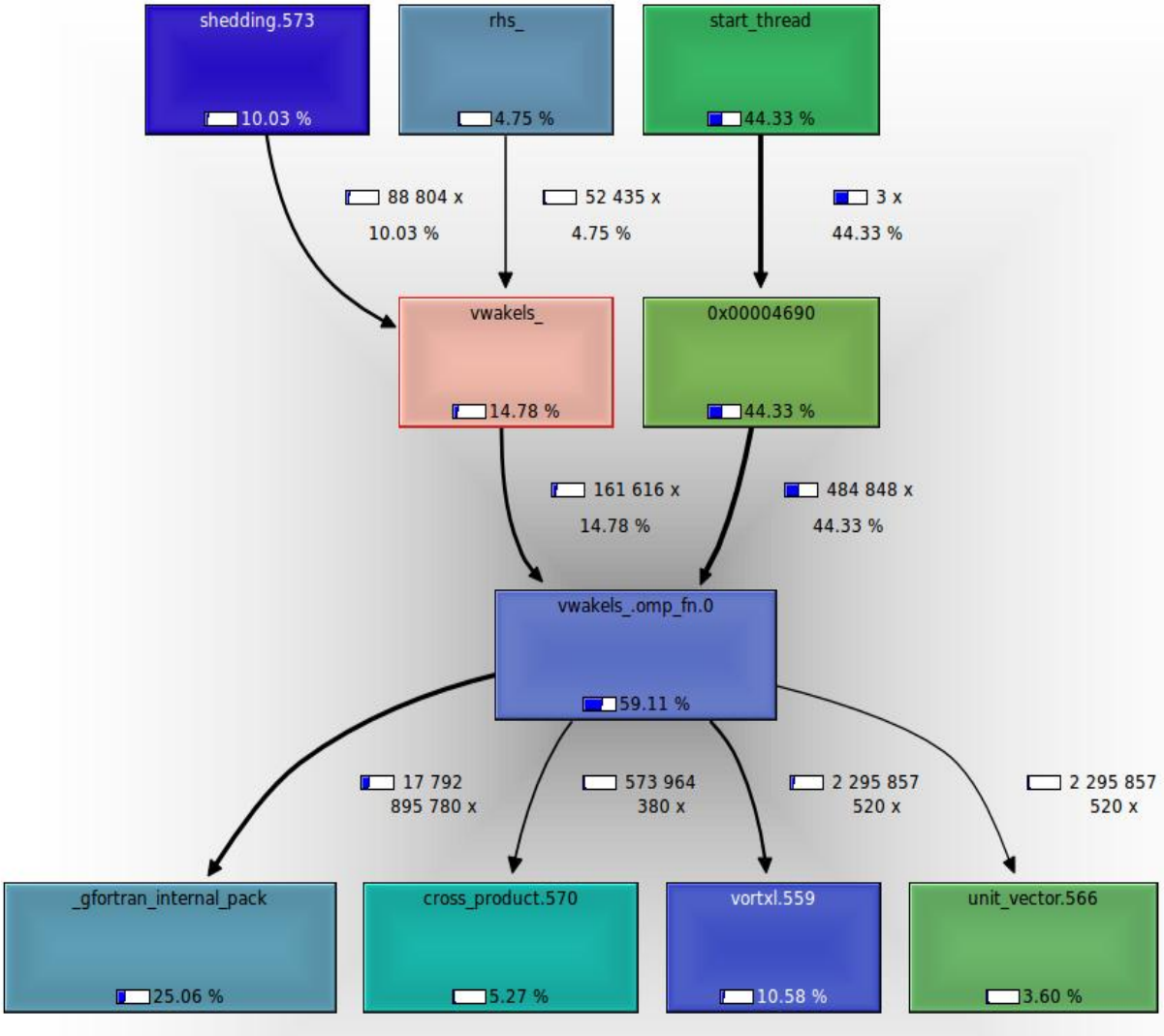

Figura 10: Rutina **`vwakels.omp`**, mostrando el detalle de los 4 threads que la ejecutan y la llamada a las demás rutinas paralelizadas.

4.2 Análisis de *performance*

En esta sección se presenta un análisis empírico de la *performance* del código computacional que implementa la estrategia de paralelización desarrollada en este trabajo. Los modelos de *performance* utilizados en este trabajo se basan en la ley de Amdahl. (Chapman et al., 2008, Almeida et al., 2008). En las subsecciones siguientes se presentan valores de *speedup* y eficiencia del código paralelizado en nuestro programa.

Speedup

El *speedup* se calcula de la siguiente manera:

$$S(n,p) = \frac{t(n)}{t(n,p)} \tag{1}$$

donde n es el tamaño del problema, p es el número de hilos de ejecución o threads, $t(n)$ es el tiempo de ejecución del programa secuencial y $t(n,p)$ es el tiempo de ejecución del programa paralelizado, utilizando p hilos de ejecución o *threads*.

En las Figura 11 se muestran de dos maneras los resultados de *speedup* obtenidos experimentalmente. En la Figura 11a se muestran curvas, para diferentes pasos de simulación, del *speedup* vs. el número de *threads* empleados. En la Figura 11b se muestran curvas del *speedup* vs. el número de pasos de simulación, para diferentes números de *threads*,

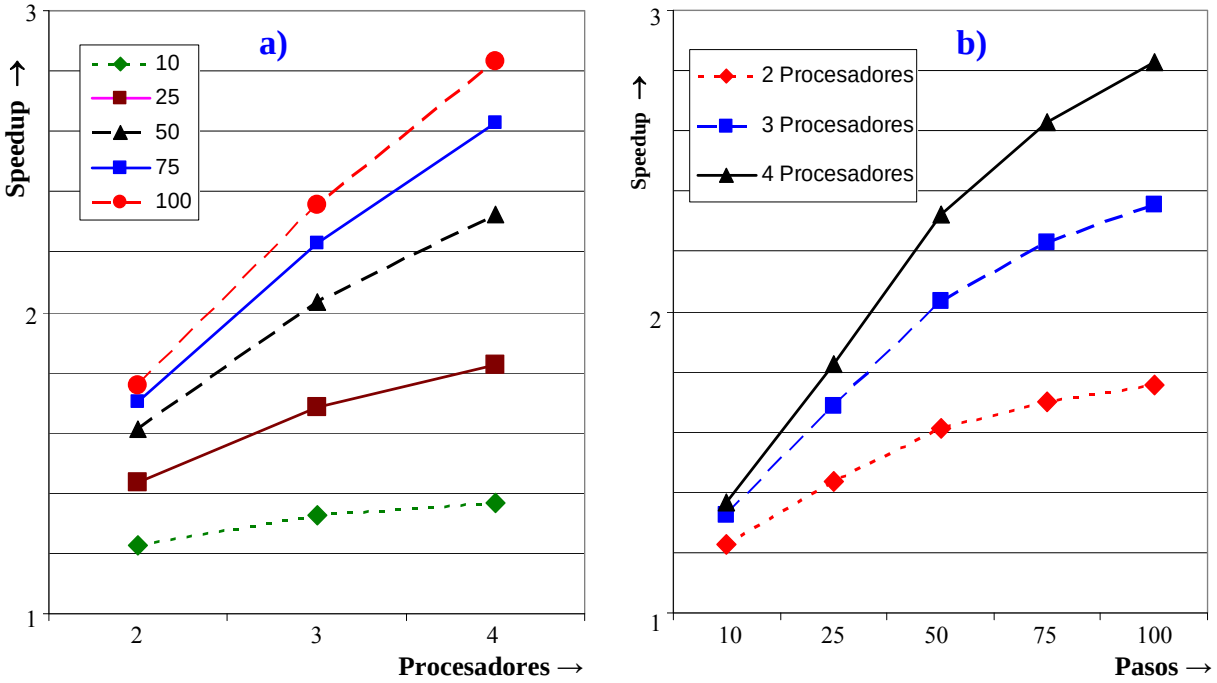

Figura 11: *Speedup* de la versión paralelizada del código: a) en función del número de *threads* y b) en función de la cantidad de pasos de simulación.

Las graficas de la Figura 11 muestran la ganancia de velocidad del código paralelizado respecto del código secuencial. Puede observarse que a medida que aumenta el número de hilos de ejecución utilizados, el *speedup* tiende a alejarse de su valor óptimo, p. Por ejemplo, en el caso de 100 pasos de ejecución se puede observar que con 2, 3 y 4 *threads*, se obtiene aproximadamente el 90, 80 y 75 % del valor óptimo de *speedup*, respectivamente.

Eficiencia

La eficiencia se calcula de la siguiente manera:

$$E(n,p) = \frac{S(n,p)}{p} \tag{2}$$

donde $S(n,p)$ es el valor de *speedup* obtenido con p hilos de ejecución.

En la Figura 12 se muestra la eficiencia obtenida experimentalmente con el código paralelizado. En la Figura 12a se muestran curvas, para diferentes pasos de simulación, de la eficiencia vs. el número de *threads* empleados. En la Figura 12b se muestran curvas, para diferente número de *threads*, de la eficiencia vs. el número de pasos de simulación.

Los resultados presentados en la Figura 12 confirman las observaciones hechas en las curvas de *speedup* de la subsección anterior y dan una idea de la porción de tiempo que los procesadores dedican a trabajo útil. Puede observarse que a medida que el número de hilos de ejecución crece la eficiencia se ve mejorada, para un número fijo de hilos de ejecución. Por

otro lado, también puede observarse una pérdida en eficiencia a medida que se aumenta el número de *threads*. El mejor valor de eficiencia observado que se aproxima al valor óptimo (el cual es uno), se obtiene en el caso 100 pasos de simulación utilizando dos *threads*.

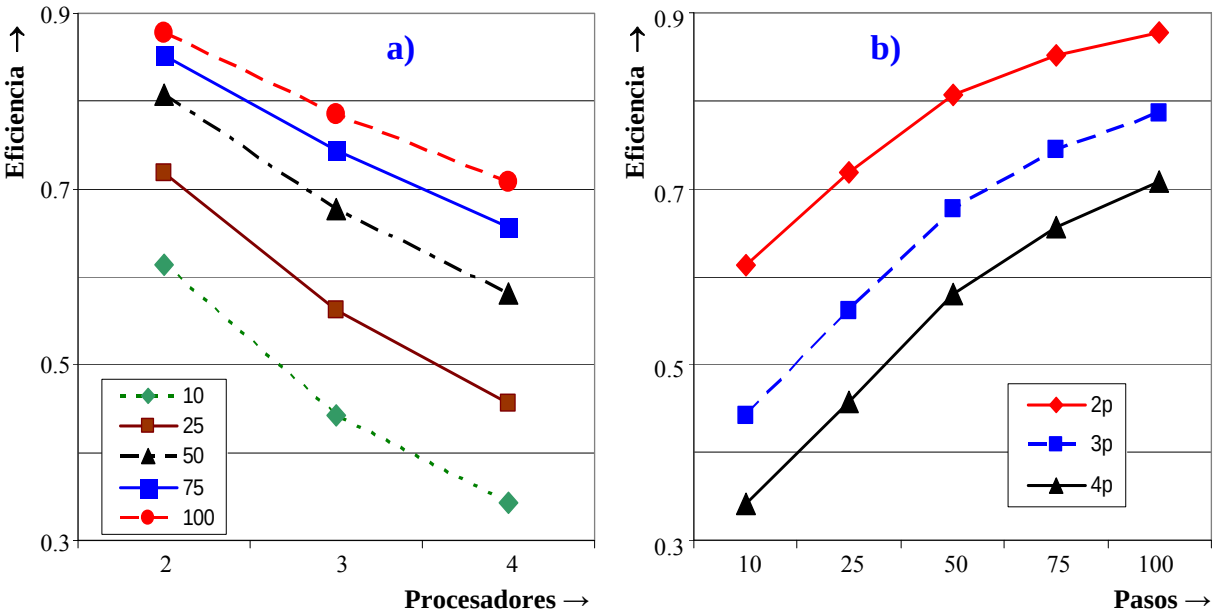

Figura 12: Eficiencia de la versión paralelizada del código vs. número de: a) *threads* , b) pasos de simulación.

5 CONCLUSIONES Y TRABAJOS FUTUROS

Se presentó una estrategia para paralelizar explícitamente códigos computacionales que implementan el método general inestacionario de red de vórtices (NUVLM) que incrementan la velocidad de ejecución de la zona del código computacional que consume más tiempo.

Se identificaron los posibles "cuellos de botella" utilizando un programa de análisis de performance sobre un código que implementa el método de red de vértices inestacionario y no lineal sobre una geometría sencilla. En las ejecuciones realizadas se observó que la forma en que se distribuyen porcentualmente los consumos de tiempo, entre las distintas etapas del cálculo del NUVLM, no se ve influenciada por el número de elementos de las mallas aerodinámicas elegidas para discretizar la geometría de un problema. Se comprobó que la forma en que se paraleliza la etapa de convección es crucial en cuanto al tiempo consumido por el NUVLM.

Las ejecuciones realizadas con los códigos paralelizados en los casos presentados en este trabajo mostraron incrementos notables en la velocidad de ejecución utilizando 2, 3 y 4 hilos de ejecución lo que se traduce en importantes ahorros en el tiempo total de ejecución que oscilan entre un 60 y un 70 %. Los análisis de speedup y eficiencia permitieron determinar los alcances de la mejora introducida en la implementación propuesta del NUVLM.

Esta estrategia se adapta perfectamente a cualquier sistema, utilizando la cantidad de hilos que tenga el equipo, sin tener que hacer modificación de ningún tipo para su utilización. Las modificaciones propuestas resultaron ventajosas, más cuando fueron implementadas en equipos de computación de escritorio, sin grandes gastos.

Como trabajo futuro esta previsto implementar dicha estrategia en programas de aplicación práctica del método de red de vértices, profundizando la estrategia en los nuevos cuellos de botella de las sucesivas modificaciones También esta previsto combinar esta estrategia con otras estrategias de paralelización, a fin de analizar las posibles mejoras de eficiencia del código.

REFERENCIAS

Almeida, F; Gimenez, D.; Mantas, J.M. y Vidal, A.M., *Introducción a la programación paralela*. Paraninfo, 2008

Ceballos L., Barone A., Flores A. y Preidkman S., Desarrollo de una estrategia de paralelización explícita para el método de red de vórtices inestacionario y no lineal. *II Congreso Argentino de Ingeniería Mecánica*. Noviembre, San Juan, Argentina, 2010.

Ceballos L., Preidikman S., Gebhardt C. y Massa J., Comportamiento aeroelástico inestacionario y no-lineal de vehículos aéreos no tripulados de alas unidas: Herramienta para relacionar el modelo aerodinámico con el estructural, *V Congreso Argentino de Tecnología Espacial*, Mayo, Mar del Plata, Argentina, 2009.

Ceballos L., Preidikman S., y Massa J., Generador paramétrico de geometrías de UAVs de alas unidas orientado al método no-lineal e inestacionario de red de vórtices. *Mecánica Computacional*, 27: 2983-3007, 2008a.

Ceballos L., Preidikman S. y Massa J., Herramienta computacional para simular el comportamiento aerodinámico de vehículos aéreos no tripulados con una configuración de alas unidas. *Mecánica Computacional*, 27: 3169-3188, 2008b.

Chapman, B., Jost, G. y Van der Pas, R. *Using OpenMP: Portable shared memory parallel programing*, MIT Press, 2008

Fritz T.E. y Long L.N., A Parallel, Object-Oriented Unsteady Vortex Lattice Method for Flapping Flight. AIAA-2004-39. *42nd AIAA Aerospace Sciences Meeting and Exhibit*, January 5-8, Reno, Nevada, 2004.

Katz J. y Plotkin A., *Low-Speed Aerodynamics.* 2nd Edition, Cambridge Aerospace Series, Cambridge, UK, 2001.

Konstadinopoulos P., Mook D.T. and Nayfeh A.H., A numerical method for general unsteady aerodynamics. AIAA-81-1877. *AIAA Atmospheric Flight Mechanics Conference*, August 19-21, Albuquerque, New Mexico, 1981.

Preidikman S., *Numerical simulations of interactions among aerodynamics, structural dynamics, and control systems*, Ph.D. Dissertation, Department of Engineering Science and Mechanics. Virginia Polytechnic Institute and State University, Blacksburg, VA, 1998.

Ravetta P.A., Desarrollo de simulaciones numéricas para el estudio aeroelástico del control de actitud de generadores eólicos medianos, Tesis de Magíster, Facultad de Ingeniería. Universidad Nacional de Río Cuarto, Argentina, 2005.

Roccia B., Preidikman S., Ceballos L. y Massa J., Implementación del método de red de vórtices no-lineal e inestacionario para estudiar la aerodinámica de micro-vehículos aéreos de alas batientes inspirados en la biología. *V Congreso Argentino de Tecnología Espacial*, Mayo, Mar del Plata, Argentina, 2009.

Roccia B., Preidikman S., y Massa J., De la biología a los insectos robots: Desarrollo de un código computacional interactivo para estudiar la cinemática de alas batientes. *Mecánica Computacional*, 27:3041-3058, 2008.

Roccia B., Preidikman S., y Massa J., Implementación de un modelo no-lineal e inestacionario para estudiar la aerodinámica de un micro-vehículo aéreo en "hover". *II Congreso Argentino de Ingeniería Mecánica*. Noviembre, San Juan, Argentina, 2010.

Verstraete, M., Ceballos, L., Preidikman, S., Aviones No-Tripulados Inspirados en el Vuelo Natural con Alas que Mutan: Aspectos Aerodinámicos. *Mecánica Computacional*, 28: 2975-2993, 2009.