Máster Universitario en Nuevas Tecnologías de la Informática

TRABAJO FIN DE MÁSTER

TÉCNICAS HEURÍSTICAS PARALELAS EN ACOPLAMIENTO DE COMPUESTOS BIOACTIVOS

Autor:

Baldomero Imbernón Tudela

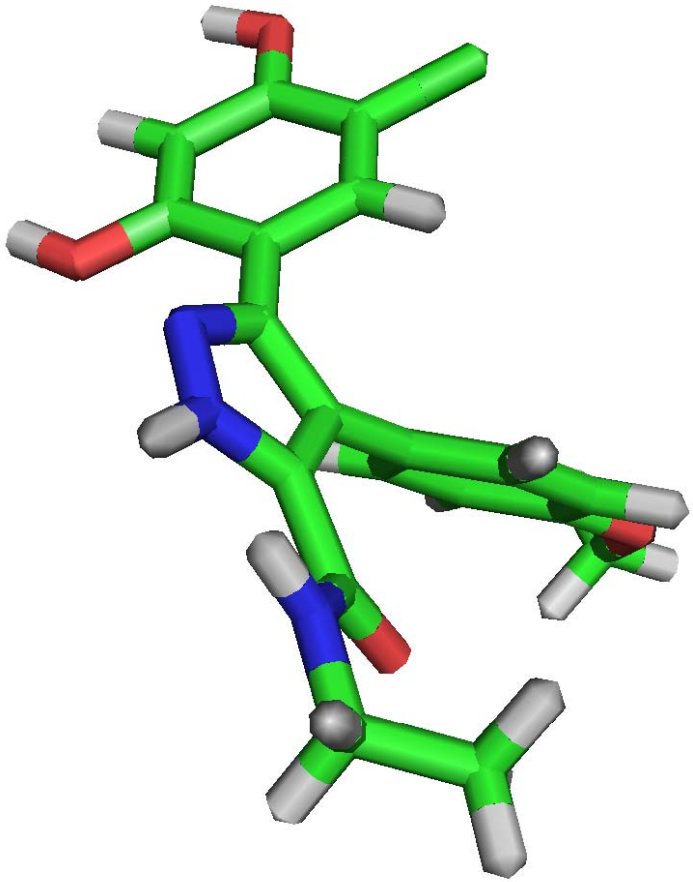
Dirigido por:

Domingo Giménez Cánovas

Murcia, Septiembre de 2015

Índice

- Introducción
- Problema
- Herramientas
  - Metaheurísticas
  - Paralelismo
- Objetivos
- Implementaciones
  - Secuencial
  - Multicore
  - GPU
  - multiGPU
- Conclusiones
- Trabajo futuro



Introducción

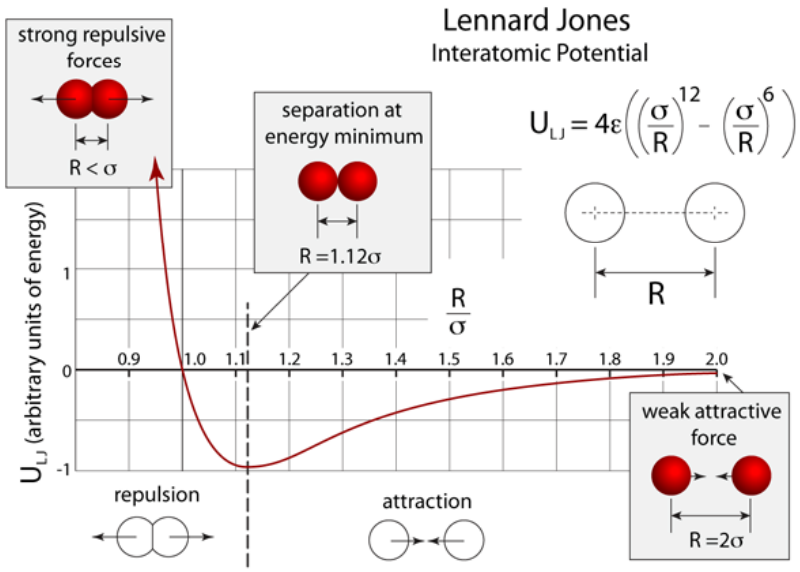
- El descubrimiento de compuestos candidatos a fármacos, es un problema donde cada vez más investigadores intentan contribuir desde sus respectivos campos de trabajo.

- Los procesos de *docking* o cribado virtual de moléculas, predicen lugares donde estos compuestos son capaces de acoplarse. Esta predicción se realiza mediante cálculos de potenciales energéticos.

- La función de *scoring* es encargada de aglutinar estos cálculos y está compuesta por un conjunto de estos potenciales. Nos centraremos en uno denominado *Potencial de Lennard-Jones.*

Problema

- El Potencial de Lennard-Jones resuelve las fuerzas de Van der Waals, que es uno de los fenómenos físicos que afectan a estos compuestos.



- **Problema de optimización**
  Generar un conjunto de conformaciones, en posiciones candidatas de acoplamiento. Posición donde el valor energético sea mínimo.

- **Alto coste computacional**
  Calcular el potencial entre cada átomo de cada una de las conformaciones o individuos generados, con cada átomo del otro compuesto receptor.

Herramientas

Metaheurísticas

- Nivel de abstracción superior al heurístico, que permite configurar estrategias de aplicación de heurísticas.
- Hay que mantener un equilibrio entre el rendimiento del sistema, la calidad de la solución y la optimización de los recursos disponibles. Las técnicas metaheurísticas ofrecen buenas aproximaciones a la solución óptima.

Esquema metaheurístico parametrizado.

```
Inicializar(S,ParamIni)
while no Fin(S) do
  Seleccionar(S,Ssel,ParamSel)
  Combinar(Ssel,Scom,ParamCom)
  Mejorar(Scom,ParamImp)
  Incluir(Scom,S,ParamInc)
end while
```

- ParamIni={NEIIni,NEFMIni,NEFPIni,PEMIni,IMEIni}
- ParamFin={NIRFin,NMIFin}
- ParamSel={NEMSel,NEPSel}
- ParamCom={NMMCom,NMPCom,NPPCom}
- ParamImp={PEMImp,IMEImp}
- ParamInc={NEMInc}

- El conjunto de parámetros metaheurísticos va a definir la metaheurística que vamos a aplicar
- También se permite con este esquema su combinación o realizar propuestas híbridas.

Herramientas

Paralelismo

- Un conjunto de parámetros paralelos se suman a los parámetros algorítmicos del esquema parametrizado.
- ***Multicore***. Se trabaja con uno o dos niveles de paralelismo en las diferentes funciones del esquema.
- ***Multicore + GPU***. Se utiliza en partes del código donde se han detectado que el coste del tratamiento de los datos entre el dispositivo y el *host,* así como su ejecución, es más ventajosa que usar exclusivamente *multicore*. Se trabaja con plataformas NVIDIA y utilizamos CUDA.
- ***Multicore + MultiGPU***. Grandes moléculas o computación intensiva, requieren un mayor uso de recursos. El utilizar varios dispositivos aumenta la complejidad, aunque también el rendimiento.

Objetivos

- Optimización del cálculo del potencial de Lennard-Jones, reduciendo el tiempo de ejecución y maximizando la eficiencia del cómputo.
- Estudiar las técnicas metaheurísticas aplicadas a la búsqueda de la mejor posición de acoplamiento entre los dos compuestos.
- Demostrar que uso de paralelismo reduce el coste computacional, observando el aumento del rendimiento desde el esquema secuencial, hasta los sistemas híbridos *multicore+GPU* y *multcore + multiGPU*.
- Obtener de forma automática un conjunto de valores de los parámetros de paralelismo lo más óptimo posible, para maximizar el rendimiento de los recursos asignados para cómputo.

Implementaciones

Secuencial

Esquema metaheurístico parametrizado.

```
Inicializar(S,ParamIni)
while no Fin(S) do
   Seleccionar(S,Ssel,ParamSel)
   Combinar(Ssel,Scom,ParamCom)
   Mejorar(Scom,ParamImp)
   Incluir(Scom,S,ParamInc)
end while
```

- ***Inicializar***. Genera un conjunto inicial de posiciones y calcula su *fitness.* Según el valor de los parámetros metaheurísticos, el conjunto inicial se somete a una mejora.
- ***Fin***. Determina una condición de parada. (Tiempo o pasadas).

- ***Seleccionar***. Selecciona un conjunto de posiciones de cada uno de los lugares candidatos a acoplamiento.
- ***Combinar***. Calcula el punto medio entre pares de individuos generados a partir del mismo punto candidato, generando un elemento nuevo en dicha posición.
- ***Mejorar***. Busca un mejor *fitness* en cada uno de los individuos, evaluando su entorno más próximo.
- ***Incluir.*** Selecciona un porcentaje de los elementos combinados en cada punto candidato para volver a iniciar el ciclo.

Implementaciones

Secuencial

Resultados computacionales

COMBINACIONES	NEIIni	PEMIni	IMEIni	NEFMIni	NEFPIni	NEMSel	NEPSel	NMMCom	NPPCom	NMPCom	PEMImp	IMEImp	NEMInc	NIRFin	NMIFin
1	64	0	0	50	50	50	50	25	25	50	25	10	80	3	10
2	64	20	10	50	50	100	0	100	0	0	0	0	100	3	10
3	128	0	0	100	0	100	0	100	0	0	0	0	100	3	10
4	128	20	10	50	50	100	0	100	0	0	0	0	100	3	10

COMBINACIONES	fitness	Segundos
1	-101.41	2932.7
2	-108.80	15334.4
3	-117.01	8775.4
4	-123.47	35319.5

COMBINACIONES	Segundos
1	589,57
2	4142.36
3	4073.55
4	16272.27

COMBINACION 1		Segundos
Inicializar	Generar elementos	0.003
	Cálculo Fitnes	**34.582**
Combinar	Combinar Elementos	0.034
	Cálculo Fitnes	**535.767**

- Resultados de *fitness* lejanos al óptimo, que se sitúa en -249.38
- Los tiempos son muy elevados con respecto al resultado obtenido, dado que para realizar una pasada del esquema se invierte un tiempo excesivo. Hay que buscar paralelismo en las partes del cálculo más costosas.
- El cálculo del *fitness* aglutina más del 98% del cómputo total, por tanto hay que centrar los esfuerzos en este cálculo para aumentar el rendimiento de la aplicación.

Implementaciones

Multicore CPU

Esquema metaheurístico parametrizado paralelo.

```
Inicializar(S,ParamIni,Threads1Ini,Threads1Fit,Threads2Fit)
while no Fin(S) do
   Seleccionar(S, Ssel, ParamSel,Threads1Sel)
   Combinar(Ssel, Scom, ParamCom,Threads1Com,Threads2Com,Threads1Fit,
   Threads2Fit)
   Mejorar(Scom,ParamImp,Threads1Imp,Threads2Imp,Threads1Fit,Threads2Fit)
   Incluir(Scom,S,ParamInc,Threads1Inc)
end while
```

- Paralelismo hasta dos niveles en las funciones de combinar los elementos y el cálculo de *fitness*. Un total de 8 parámetros de paralelismo.
- Los procedimientos del esquema metaheurístico mantienen la misma función descrita en el esquema secuencial.

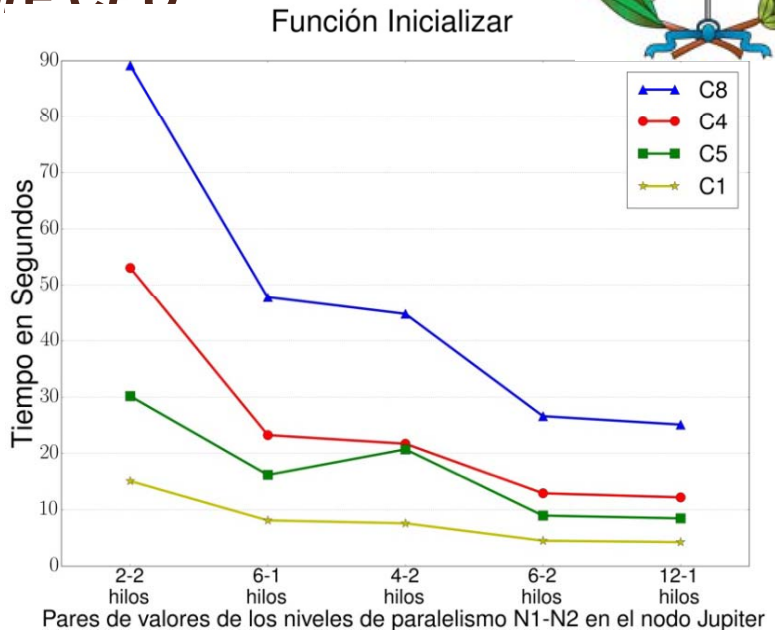
Parámetros Paralelismo
Threads1Ini
Threads1Fit
Threads2Fit
Threads1Sel
Threads1Com
Threads2Com
Threads1Imp
Threads1Inc

Implementaciones

Multicore CPU

Resultados computacionales

- Con el uso de paralelismo en los cálculos aumentando el rendimiento, podemos elevar el valor de los parámetros metaheurísticos para encontrar un mejor valor de *fitness*.



COMBINACIONES	NEIIni	PEMIni	IMEIni	NEFMIni	NEFPIni	NEMSel	NEPSel	NMMCom	NPPCom	NMPCom	PEMImp	IMEImp	NEMInc	NIRFin	NMIFin
1	96	0	0	100	0	100	0	50	0	0	25	10	80	5	20
2	128	20	10	100	0	50	50	25	25	50	25	10	80	5	20

COMBINACIONES	fitness	Segundos
1	-137.57	3570.12
2	-211.71	4979.27

COMBINACIONES	Secuencial	Multicore	Speed-Up
1	589,57	28.48	20.7
2	4142.36	188.25	22
3	4073.55	187.21	21.7
4	16272.27	755.46	21.5

- Se obtienen mejores valores de *fitness,* dado el aumento, tanto de pasadas, como de parámetros metaheurísticos.
- Conseguimos un *Speed-Up* considerable con respecto a la versión secuencial, demostrando la mejora de rendimiento.

Implementaciones

Multicore CPU

Auto-optimización

- Los valores de los parámetros paralelos óptimos para la ejecución se obtienen en un proceso de auto-optimización donde se realizan pequeñas ejecuciones con un número limitado de individuos para determinar su valor.
- Se realiza en la primera ejecución y los resultados se guardan para futuras ejecuciones. Consiste en la ejecución de un conjunto de pruebas con un reducido número de individuos para cada parámetro de paralelismo y para cada nivel que tenga la función.

COMBINACIONES	NEIIni	PEMIni	IMEIni	NEFMIni	NEFPIni	NEMSel	NEPSel	NMMCom	NPPCom	NMPCom	PEMImp	IMEImp	NEMInc	NIRFin	NMIFin
1	64	20	10	50	50	100	0	100	0	0	0	0	100	3	10
2	128	20	10	50	50	100	0	100	0	0	0	0	100	3	10

Parm\Conf	Manual	Óptima
Threads1Ini	6	11
Threads1Fit	6	12
Threads2Fit	2	1
Threads1Sel	6	8
Threads1Com	6	11
Threads2Com	2	1
Threads1Imp	6	9
Threads1Inc	6	8

COMBINACIONES	Manual	Óptima	Mejora
1	319.11	279.11	12.5%
2	1280.76	1099.87	14.12%

- Se parte del número de hilos más alto, y se va reduciendo almacenando el tiempo consumido. Al final el valor de ese parámetro paralelo será el que mejor rendimiento haya tenido.

Implementaciones

Multicore + GPU

Esquema metaheurístico parametrizado paralelo híbrido.

```
Inicializar(S,Devices,ParamIni,Threads1Ini,ThreadsblockMoveIni,
ThreadsblockRot,ThreadsblockQuat,ThreadsblockRandom,
ThreadsblockMoveImp,ThreadsblockIncImp,ThreadsblockFit)
while no Fin(S) do
   Seleccionar(S,Ssel,ParamSel,Threads1Sel)
   Combinar(Ssel,Scom,Devices,ParamCom,Threads1Com,Threads2Com,
   ThreadsblockFit)
   Mejorar(Scom,Devices,ParamImp,ThreadsblockMoveImp,ThreadsblockRot,
   ThreadsblockIncImp,ThreadsblockFit)
   Incluir(Scom,S,ParamInc,Threads1Inc)
end while
```

Parámetros Paralelismo
Threads1Ini
Threads1Sel
ThreadsblockFit
ThreadsblockMoveIni
ThreadsblockRot
ThreadsblockQuat
ThreadsblockRandom
Threads1Com
Threads2Com
ThreadsblockMoveImp
ThreadsblockIncImp
Threads1Inc

- La implementación en GPU se ha realizado en CUDA, y la aplicación se ejecuta sobre tarjetas NVIDIA.
- Se modifica el conjunto de parámetros de paralelismo introduciendo la gestión de los hilos por bloque en GPU, y se mantiene el resto de parámetros paralelos que afectan a funciones donde no se va a usar la GPU.
- Los procedimientos del esquema metaheurístico, mantienen las mismas funciones que en los esquemas anteriores.

Implementaciones

Multicore + GPU

Resultados computacionales

- Aplicamos técnicas masivamente paralelas al cálculo de *fitness* que suponía casi la totalidad del tiempo de computación, logrando ganancias importantes en el rendimiento.

COMBINACIONES	NEIIni	PEMIni	IMEIni	NEFMIni	NEFPIni	NEMSel	NEPSel	NMMCom	NPPCom	NMPCom	PEMImp	IMEImp	NEMInc	NIRFin	NMIFin
1	96	0	0	100	0	100	0	50	0	0	25	10	80	5	20
2	128	20	10	100	0	50	50	25	25	50	25	10	80	5	20

- Conseguimos mejores resultados de *fitness* en menos tiempo que con el uso exclusivo de multicore CPU.

COMBINACIONES	fitness	Segundos
1	-180.78	28.12
2	-207.17	101.48

- Conseguimos un *Speed-Up* considerable con respecto a la versión multicore en CPU, aplicando técnicas masivamente paralelas a los procedimientos de mayor coste

COMBINACIONES	Multicore	GPU	Speed-Up
1	28.48	3.31	8.6
2	188.25	23.24	8.1
3	187.21	24.01	7.7
4	755.46	93.26	8.1

Implementaciones

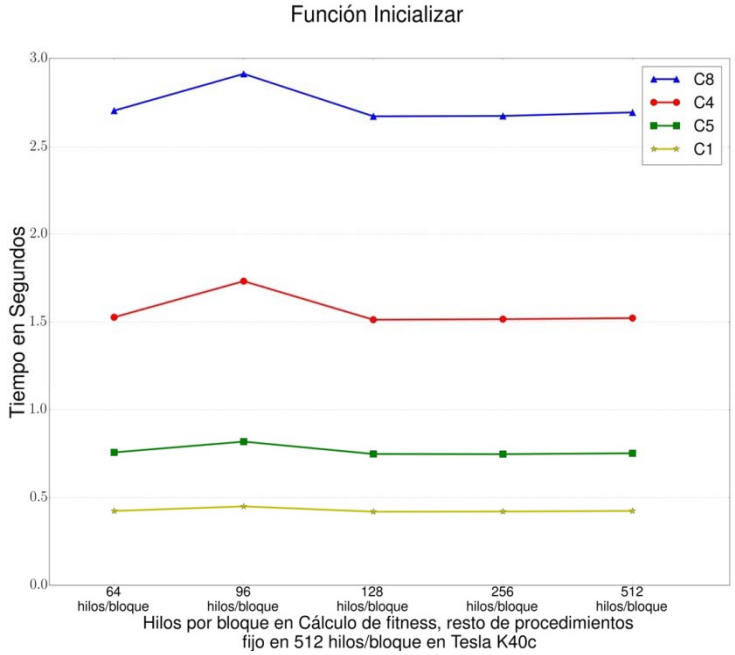
Multicore + GPU

Auto-optimización en GPU

- Los valores óptimos de los parámetros paralelos en GPU para realizar las ejecuciones en un dispositivo, se obtienen realizando unas ejecuciones con un conjunto de entrenamiento.
- Se realiza solo una vez en cada dispositivo CUDA para cada parámetro de paralelismo y para cada nivel que tenga la función .

Parm\Conf	Manual	Óptima
Threads1Ini	12	11
Threads1Sel	12	8
ThreadsblockFit	512	64
ThreadsblockMoveIni	256	448
ThreadsblockRot	128	512
ThreadsblockQuat	256	384
ThreadsblockRandom	128	128
Threads1Com	6	11
Threads2Com	2	1
ThreadsblockMoveImp	512	128
ThreadsblockIncImp	256	128
Threads1Inc	12	8

COMBINACIONES	Manual	Óptima	Mejora
1	45.41	39.75	12.4%
2	164.99	155.65	5,60%



- Se calcula para cada *kernel* entre los valores 64 – 512 hilos por bloque en múltiplos de 64.

Implementaciones

Multicore + multiGPU

Esquema metaheurístico parametrizado paralelo híbrido.

- Se trabaja con el mismo esquema metaheurístico parametrizado que en la implementación multicore + GPU, aunque en este caso, se añade un esquema internamente en multicore para el computo con varios dispositivos.

```
omp_set_num_threads(numero_GPUs)
#pragma omp parallel for
for i = 1 to numero_GPUs do
        Tratamiento a realizar por cada GPU
end for
```

- Los diferentes *kernels* se adaptan a la computabilidad de cada uno de los dispositivos NVIDIA, dado que se usan conjuntos de instrucciones no permitidos hasta cierta computabilidad, que favorecen el rendimiento.

Implementaciones

Multicore + multiGPU

Modos de cómputo y Resultados computacionales.

- En el ***modo homogéneo***, se trabaja con las GPUs del nodo de cómputo que sean iguales. La carga de trabajo se reparte de forma equitativa entre todos los dispositivos.
- En el ***modo heterogéneo***, se trabaja todas las GPUs de NVIDIA del nodo. La carga de trabajo se reparte en función de las características de los dispositivos.
- Con esta implementación obtenemos mejor valor de *fitness* en la misma cantidad de tiempo que en implementaciones anteriores dado que podemos realizar más pasadas al esquema.

COMBINACIONES	GPU	multiGPU Homogeneo	multiGPU Heterogeneo	Speed-Up
1	3.31	**2.53**	2.66	1.3
2	23.24	13.60	**11.85**	1.9
3	24.01	11.83	**10.88**	2.2
4	93.26	48.87	**42.54**	2.2

- Conforme aumentamos el cómputo, la ganancia va a ir creciendo de forma progresiva. La combinaciones 3 y 4 lo demuestran.

Implementaciones

Multicore + multiGPU

Auto-optimización en multiGPU

- Se trabaja en dos líneas de optimización, el ***reparto de cargas de trabajo en modo Heterogeneo*** entre los distintos dispositivos y el ***número de hilos por bloque*** en cada *kernel* de cada uno de los dispositivos.

- El reparto de carga se realiza ejecutando el cálculo de *fitness* a un conjunto de entrenamiento en cada dispositivo, dotando de un porcentaje de carga de trabajo en función del tiempo invertido en el cálculo.

COMBINACIONES	NEIIni	PEMIni	IMEIni	NEFMIni	NEFPIni	NEMSel	NEPSel	NMMCom	NPPCom	NMPCom	PEMImp	IMEImp	NEMInc	NIRFin	NMIFin
1	64	0	0	100	0	100	0	50	0	0	0	0	100	3	10
2	64	0	0	50	50	50	50	25	25	50	25	10	80	3	10
3	64	20	10	100	0	50	50	50	25	25	25	10	80	3	10

COMBINACIONES	multiGPU Heterogeneo CON Reparto	multiGPU Heterogeneo SIN Reparto	Mejora
1	888.42	946.66	8
2	734.21	721.13	2
3	1019.07	928.03	9

- Conforme el cómputo es mayor, el porcentaje de mejora va en aumento, como se puede observar en la tabla.

- El número de hilos por bloque de cada dispositivo, se realiza de igual forma que la auto-optimización para una sola GPU.

Conclusiones

- Se ha optimizado el cálculo del potencial de Lennard-Jones, evolucionando desde una versión *secuencial*, hasta la versión paralela *multicore* + *multiGPU*, que obtiene el mejor rendimiento en los compuestos evaluados.

COMBINACIONES	NEIIni	PEMIni	IMEIni	NEFMIni	NEFPIni	NEMSel	NEPSel	NMMCom	NPPCom	NMPCom	PEMImp	IMEImp	NEMInc	NIRFin	NMIFin
1	64	0	0	100	0	100	0	50	0	0	0	0	100	3	10
2	64	0	0	50	50	50	50	25	25	50	25	10	80	3	10
3	128	20	10	50	50	100	0	100	0	0	0	0	100	3	10

COMBINACIONES	Versión Secuencial	Versión Multicore	Esquema híbrido Multicore + GPU	Multicore + multiGPU Heterogénea	SPEED-UP
1	1035.01	47.86	5.54	3.46	**299.13**
2	589.57	28.48	3.31	2.66	**221.64**
3	16272.27	755.46	93.26	42.54	**382.51**

Tiempos secuencial, multicore y multicore+multiGPU en nodo Jupiter.

Tiempos multicore + GPU en Hertz con Tesla K40c.

- El uso de un esquema metaheurístico parametrizado paralelo ha sido una solución acertada para aumentar el rendimiento de la aplicación.

Conclusiones

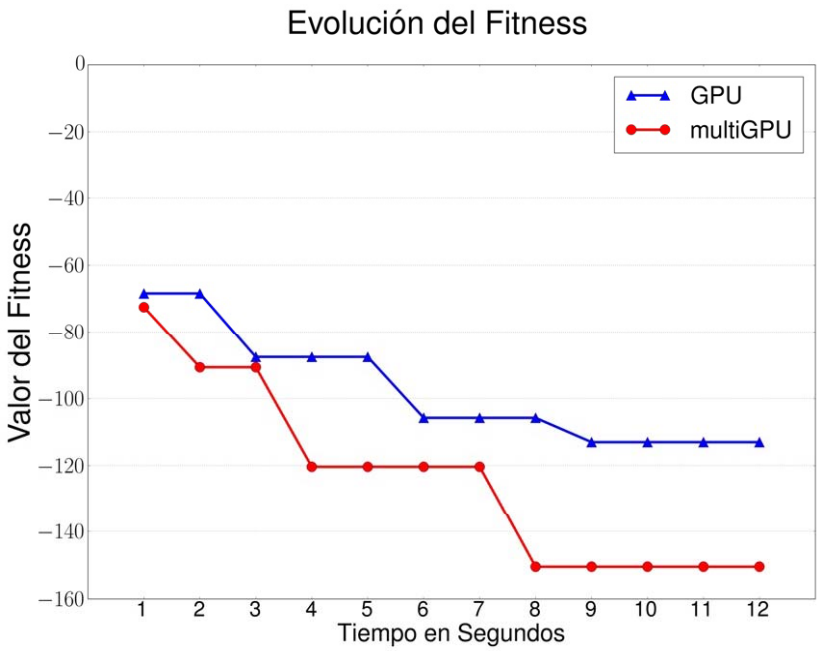
- Con este esquema metaheurístico parametrizado obtenemos buenos resultados de *fitness* en pocos segundos, usando técnicas paralelas y masivamente paralelas.

COMBINACIONES	NEIIni	PEMIni	IMEIni	NEFMIni	NEFPIni	NEMSel	NEPSel	NMMCom	NPPCom	NMPCom	PEMImp	IMEImp	NEMInc	NIRFin	NMIFin
1	96	20	10	50	50	100	0	100	0	0	10	5	100	5	20
2	128	0	0	100	0	100	0	50	0	0	25	10	100	5	20
3	256	0	0	100	0	50	50	50	25	25	25	10	80	5	20

COMBINACIONES	fitness	Segundos
1	-198.65	110.46
2	-205.84	87.41
3	-207.17	101.48

Tiempos multicore + GPU en Hertz con Tesla K40c.

- Con el uso de paralelismo obtenemos mejor *fitness* en un espacio de tiempo más pequeño.



Conclusiones

- Obtener los valores óptimos de la plataforma en que se vaya a ejecutar la aplicación, va a suponer una ganancia considerable en el rendimiento.

COMBINACIONES	NEIIni	PEMIni	IMEIni	NEFMIni	NEFPIni	NEMSel	NEPSel	NMMCom	NPPCom	NMPCom	PEMImp	IMEImp	NEMInc	NIRFin	NMIFin
1	64	20	10	50	50	100	0	100	0	0	0	0	100	3	10
2	128	20	10	50	50	100	0	100	0	0	0	0	100	3	10

Configuraciones Multicore

Parm\Conf	Manual	Óptima
Threads1Ini	6	11
Threads1Fit	6	12
Threads2Fit	2	1
Threads1Sel	6	8
Threads1Com	6	11
Threads2Com	2	1
Threads1Imp	6	9
Threads1Inc	6	8

COMBINACIONES	Manual	Óptima	Mejora
1	319.11	279.11	12.5%
2	1280.76	1099.87	14.12%

Configuraciones Multicore + GPU

Parm\Conf	Manual	Óptima
Threads1Ini	12	11
Threads1Sel	12	8
ThreadsblockFit	512	64
ThreadsblockMoveIni	256	448
ThreadsblockRot	128	512
ThreadsblockQuat	256	384
ThreadsblockRandom	128	128
Threads1Com	6	11
Threads2Com	2	1
ThreadsblockMoveImp	512	128
ThreadsblockIncImp	256	128
Threads1Inc	12	8

COMBINACIONES	Manual	Óptima	Mejora
1	45.41	39.75	12.4%
2	164.99	155.65	5,60%

Trabajo futuro

- Diseñar hiperheurísticas para determinar la metaheurística más adecuada a nuestro problema, obteniendo la mejor combinación paramétrica posible.
- Extensión a un sistema de memoria compartida, basado en paso de mensajes para aprovechar todos los recursos que nos puede ofrecer un cluster.
- Ampliación a un sistema multiobjetivo, donde se utilicen otros potenciales de interacción para dotar de mayor confianza al mejor lugar de acoplamiento que la aplicación determine.
- Perfeccionar los mecanismos de autooptimización, realizando un conjunto de *benchmark* a dos niveles, tanto en la propia instalación de la aplicación, como en las primeras ejecuciones, ampliando además a nivel de cluster si la aplicación lo permite.
- Obtención de recursos de cómputo en la nube a través de sistemas como *Amazon EC2* para abaratar costes y obtener una amplia gama de recursos si es necesario en momentos puntuales.

TÉCNICAS HEURÍSTICAS PARALELAS EN ACOPLAMIENTO DE COMPUESTOS BIOACTIVOS

Gracias

PREGUNTAS….