V. SOLUCIÓN ADOPTADA

Tras el análisis de las distintas opciones se decidió adquirir una tarjeta y montarla en un equipo del Laboratorio.

Para ello había que decidir el tipo de tarjeta que fuera compatible con un equipo preexistente, teniendo en cuenta factores tan importantes como el precio y la disponibilidad.

1. CONFIGURACIÓN BASADA EN TESLA SUGERIDA

La opción de montar un equipo "a medida" es muy interesante, ya que ofrece una máxima flexibilidad y adaptación a las necesidades definidas para cada caso particular. Generalmente también deriva de ella un ahorro económico aunque siempre hay que valorar si compensa el tiempo invertido.

Se observa que estos equipos son relativamente sofisticados y se debe cuidar al detalle todos los aspectos de compatibilidad, además, salta a la vista que son muy exigentes.

Una solución analizada y que es meramente orientativa se basa en la tecnología más potente de GPUs NVIDIA para cálculo. Indica una solución posible para montar o adaptar un sistema ya existente para obtener un ordenador de cálculo basado en TESLA.

El sistema se basaría las siguientes especificaciones de partida:

- 3x Tesla C1060
- Quad-core CPU: 2.33 GHz (Intel o AMD)
- 12 GB de memoria del sistema (4GB para cada Tesla C1060)
- Linux 64-bit o Windows XP 64-bit
- Fuente de alimentación de 1200 W

A partir de ellos podemos proponer dos configuraciones ilustrativas.

Ejemplo de la configuración de un sistema Tesla C 1060 cuádruple

A título orientativo en la tabla 1, se ofrece un listado de componentes que pueden ser sustituidos por otros equivalentes en prestaciones y calidad.

Tabla 1: Configuración 4 Tesla C1060

Placa Base	Tyan S7025
PCI-e bandwidth	4x PCI-e x16 Gen2 slots
GPUs Tesla	4x Tesla C1060
Tarjeta Gráfica	On-board graphics (works with Linux, Windows requires NVIDIA GPU in one of the PCI-e slots
CPU	Dual-socket Intel Xeon Nehalem

Ejemplo de la configuración de un sistema Tesla C1060 Triple

A título orientativo en la tabla 2, se ofrece un listado de componentes que pueden ser sustituidos por otros equivalentes en prestaciones y calidad.

Tabla 2: Configuración 3 Tesla C1060 + 1 Quadro FX5800		
	Opción A	Opción B
PCI-e bandwidth	4 PCI-e x16 Gen2 Slots	4 PCI-e x8 Gen2 Slots
Placa Base	ASUS P6T7 WS SuperComputer	AsRock X58 SuperComputer
GPUs Tesla	3x Tesla C1060	3x Tesla C1060
Tarjeta Gráfica	1x Quadro FX o NVS	1x Quadro FX o NVS
CPU	Intel Xeon E55xx / Core i7	Intel Core i7
Min. Memoria	24 GB DDR3	24 GB DDR3
Fuente de alimentación	1200W	1200W

Debido a las exigencias de estos sistemas, veremos con más detalle cada uno de los elementos principales que lo forman, aparte de las GPUs.

Placas Base

El Tesla C1060 es una tarjeta de procesador PCI-e x16 Gen2 de doble ancho. Puede ser usada en una ranura CI-e x16 Gen 1, pero esto provocaría una disminución del ancho de banda entre la CPU y la GPU que dependiendo de la aplicación podría provocar una disminución del rendimiento. Por ello se recomienda el uso de placas base que dispongan de 3 o 4 ranuras PCI-e x16 que sean independientes por pares.

CPUs

La elección de la CPU depende de la placa base. Se recomienda al menos un sistema Quad-core CPU de 2.33 GHz como:

- Intel Xeon o Core i7 quad-core.
- AMD Phenom o Opteron quad-core.

Memoria del sistema

Se recomienda disponer de 4GB de memoria por cada Tesla C 1060, cómo mínimo.

Fuente de Alimentación

Se recomienda una fuente de alimentación que proporcione al menos 1200 W aunque es conveniente una de 1350 W. Por ejemplo la Coolmax CUQ-1350B 1350W.

Se debe tener presente que cada Tesla C 1060 necesita 2 conectores 6-pin para la alimentación o bien uno de 8.

Disco Duro y DVD

Se recomienda al menos un disco duro de 160 GB. Aunque queda a la libre elección del usuario.

Carcasas

La elección de la carcasa es importante ya que debe acomodar, para el caso cuádruple, 4 tarjetas C 1060 que necesitan 8 ranuras. Esto implica que la carcasa es mayor que el modelo ATX convencional. Algunas torres comercializadas indicadas:

- Lian-Li PC-P80
- ThermalTake ArmorPlus
- ABS Canyon 695

Sistema de refrigeración

Hay que disponer ventiladores suficientes para mantener la temperatura por debajo de los 45°C.

Sistema Operativo

Se recomiendan sistemas 64 bits. Se soportan tanto Linux como Windows.

Verificación del Sistema

Una vez montado el sistema e instalado el Sistema Operativo, se debe instalar el CUDA driver y CUDA toolkit. Seguidamente se ejecutarán los siguientes comandos desde la SDK de CUDA.

deviceQuery

- Informará del número de GPUs Tesla en el sistema.

bandwidthTest --memory=pinned --device=N

- Ejecutar para cada C1060, donde N=0, 1, 2, 3 para sistemas con 4 C1060.
- Devolverá el ancho de banda de PCI-E entre la CPU y cada una de las GPUs
- Se deben esperar máximos de 5 a 6 GBytes/s para PCI-E x16 Gen2. En el caso de PCI-E x16 Gen1 y PCI-E x8 Gen2 los valores deberían ser aproximadamente la mitad.

nbody --benchmark --n=131072 --device=N

- Ejecutar simultáneamente para tantas GPUs TESLA como hay instaladas en el sistema.
- Para 4 Tesla C1060s, ejecutar 4 veces con N=0,1,2,3 simultáneamente.
- Se ejecutará el programa nbody en todas las GPU Tesla.

Finalmente se recomienda usar el test DGEMM burn-in test para evaluar la estabilidad de la instalación.

Esta opción aunque válida tiene el inconveniente de no ser la mejor opción para una etapa en la que se explora la Tecnología y se estudia su idoneidad para las aplicaciones de nuestro caso particular.

De hecho se concluyó que las tarjetas Tesla eran más indicadas para una solución "definitiva" dedicada y no para un estudio exploratorio de la Tecnología. También se descartaron en parte por la misma razón Las tarjetas de la familia Quadro.

Finalmente la solución adoptada se basaría en la familia GeForce 2xx. Las capacidades de estas tarjetas, junto a su precio las hicieron idóneas para ser seleccionadas para cumplir los fines del proyecto.

Para ello también se debía adoptar una solución para el PC. Naturalmente había que seleccionar un equipo compatible dedicado que pudiera soportar las características de la nueva tarjeta y la nueva tecnología objeto de análisis.

Se barajaron las siguientes opciones:

2. SOLUCIÓN INICIAL: Dell Optiplex 745+GeForce 8800.

El primer problema consistía en que la nueva tarjeta necesitaría más energía, para ello se debía cambiar la fuente de alimentación. La alimentación del Dell es inferior a los 450 w aproximadamente que se recomiendan para la NVIDIA 2xx. De hecho este modelo dispone de una fuente de alimentación, según modelo, que varía entre 220W y 305W. En sí, esto no fue un inconveniente ya que se encontraron fuentes compatibles, con potencias del orden de 700 W y con precios razonables.

Sin embargo el equipo disponía de una ranura PCI 1x16 E. Esto limitaba el abanico de opciones para la tarjeta gráfica. En concreto descartaba todas las tarjetas modernas debido a que usan la ranura PCI E 2x16.

Se buscó información sobre la compatibilidad "hacia adelante" del Hardware; pero no se pudo confirmar. La conclusión era que los nuevos estándares de bus sí soportaban las antiguas tarjetas, pero no había garantía de que una tarjeta moderna funcionase correctamente en una ranura antigua.

Por ello y manteniendo la necesidad de cambiar de fuente de alimentación se consideró que la tarjeta idónea sería de la familia GeForce 8800 ultra o bien GTX. Sin embargo se tuvo que desechar esta opción ya que este modelo ya no estaba disponible en los distribuidores.

3. SOLUCIÓN TRANSITORIA: PC clónico+GeForce 9500 GT.

Para permitir un desarrollo más flexible del proyecto. Se decidió usar un Ordenador personal Dual Core 2.5 Ghz con una única tarjeta Gráfica GeForce 9500 GT. El bus de la tarjeta es PCI-E 1x16. Esta tarjeta está diseñada para juegos y vídeo, sin embargo soporta la tecnología CUDA y su fiabilidad es mayor que la opción de Emulador. Dispone de 4 Microprocesadores y soporta formato de precisión "single float".

En este ordenador, se desarrollaron, depuraron y ejecutaron todos los códigos iniciales. Se hizo usando un sistema operativo Ubuntu 9.04 con la versión cuda toolkit 2.3. La compatibilidad y la instalación del software no ofrecieron mayores dificultades que las previstas. La mayor parte del desarrollo se realizó en este equipo y se recogieron gran número de experiencias que exponemos en los siguientes puntos:

INSTALACIÓN DE SOFTWARE CUDA SOBRE UBUNTU:

Los pasos realizados en este apartado se basan en los que aparecen en la guía oficial CUDA, sin embargo al ser más particulares los completan.

Primero se deben instalar las bibliotecas esenciales de programación. Se puede hacer mediante la orden en la línea de comandos:

sudo apt-get install build-essential libglut3-dev

Además se deben instalar las siguientes bibliotecas:

libxmu-dev
libxmu-headers
libxi-dev

Esto se puede hacer mediante "apt-get" o mediante el gestor de paquetes "Synaptic".

Seguidamente se instala el driver Nvidia mediante las siguientes órdenes:

Ctrl + Alt + F1: para usar la consola.

sudo /etc/init.d/gdm stop. Para apagar el modo gráfico.

sudo ./cudadriver_*.run . Para ejecutar la instalación.

Previamente se debe haber permitido la ejecución de dicho software mediante el cambio de permisos en el modo gráfico o bien por consola mediante la orden:

chmod +x ./cudadriver_*.run

Se siguen las instrucciones que aparecen en pantalla y una vez finalizado el proceso, se reactiva el modo gráfico mediante:

sudo /etc/init.d/gdm start

Para la instalación del toolkit hay dos opciones:

Modo gráfico:

Se cambian las opciones para permitir la ejecución del archivo.

Con el botón derecho del ratón se ejecutn eligiendo **run** o bien **run in terminal**.

Modo Terminal:

En un terminal previamente abierto se ejecutan las siguientes instrucciones:

chmod +x ./cudatoolkit_2.3_linux_64_ubuntu9.04.run

sudo ./cudatoolkit_2.3_linux_64_ubuntu9.04.run

Desarrollo del proyecto:

También se disponía en el mismo equipo de MATLAB 2009 instalado para realizar pruebas con la versión Matlab del problema. Así como para generar el mallado y modificarlo.

Incluso se realizaron pruebas en el marco de la supervisión y seguimiento del desarrollo del proyecto. Los resultados fueron interesantes y arrojaban luz sobre la técnica de desarrollo, sin embargo se debe observar que el PC no era de uso exclusivo para el proyecto y las condiciones de las pruebas no eras las de un laboratorio.

Las pruebas definitivas y en un marco exhaustivo de ejecución y en condiciones controladas se reservaron para el equipo final del Laboratorio.

Compilación de código CUDA:

En caso de que el archivo principal tenga extensión .cu y haga uso de las bibliotecas cublas que se "linkan". En un terminal se introduce el comando:

nvcc archivo.cu -lcublas -o ejecutable

En caso de que además se llame a un código no CUDA en este caso C (mmio.c por ejemplo).

nvcc archivoCUDA.cu archivoC.c -lcublas -o ejecutable

Ejecución de programa CUDA:

En un terminal se introduce el comando: **./ejecutable**

Naturalmente existe la posibilidad de salida por pantalla o de redirigirla a un archivo.

4. SOLUCIÓN ADOPTADA: Dell studio xps+GeForce 260 GTX

El laboratorio adquirió para el proyecto un equipo nuevo y exclusivo.

El equipo es un Quadcore de 2.5 Ghz con tarjeta Gráfica GeForce 260 GTX con 24 microprocesadores. Esta tarjeta tiene la particularidad de soportar el formato "double" de precisión. El equipo dispone de 2 discos duros que pueden ser usados en formato RAID o bien independientemente.

Se decidió usar los discos duros de manera independiente y se procedió a instalar en un disco duro el sistema operativo openSuse 11.2 que ofreció mejores capacidades de instalación que Ubuntu 9.10 y Ubuntu 10.04. Se instaló la versión Cuda toolkit 3.1, los pilotos actualizados de la tarjeta y se compiló la SDK con éxito. Se ejecutaron los programas de prueba bandwidthTest y deviceQuery obteniendo sendas salidas correctas.

También se instaló el Matlab 2009 con el mismo objetivo que en el caso anterior.

INSTALACIÓN DE SOFTWARE CUDA SOBRE OpenSuse:

Al igual que en el caso anterior, se detallan los pasos particulares correspondientes a la instalación de herramientas CUDA sobre OpenSuse 11.2 64 bits.

Las características fundamentales de la plataforma son:

- kernel Linux 2.6.31.14-0.1-desktop
- Equipo: Dell Studio XPS 8100
- Sistema Quadcore Intel(R) Cre(TM) i3 CPU 530 @ 2.93 Ghz.
- Tarjeta Gráfica: NVIDIA 260GTX

Instalación del Driver NVIDIA

Permitir la ejecución de paquete NVIDIA Driver como aplicación.

Ctrl+Alt+Backspace

Ctrl+Alt+F1 consola

log in como root

Ejecutar comando **./NVIDIA (driver)** y seguir instrucciones de pantalla.

Instalación del NVIDIA toolkit

Con el botón derecho del ratón se ejecuta eligiendo **run** o bien **run in terminal.**

Realizar los cambios de ruta mediante:

export PATH=/usr/local/cuda/bin:$PATH

export LD_LIBRARY_PATH=/usr/local/cuda/lib64:$LD_LIBRARY_PATH

Para hacer estos cambios permanentes, editar ***~/.bash_profile.***

Instalación de la SDK

Permitir ejecución de paquete sdk como aplicación.

cuda@linux-p07k:~/Download> **ls**

cudatoolkit_3.1_linux_64_suse11.2.run gpucomputingsdk_3.1_linux.run

devdriver_3.1_linux_64_256.40.run
NVIDIA_CUDA_C_ProgrammingGuide_3.1.pdf

GettingStartedLinux.pdf NVIDIA-Linux-x86_64-256.53.run

gpucomputingsdk_3.0_linux.run

cuda@linux-p07k:~/Download> **./gpu*3.1*run**

Link de bibliotecas necesarias

ln -s /usr/lib64/libglut.so.3 /usr/lib64/libglut.so

Compilación de los ejemplos

cuda@linux-p07k:~> **ls**

bin Desktop Documents Download Music NVIDIA_GPU_Computing_SDK
Pictures Public public_html Templates Videos

cuda@linux-p07k:~> **cd NV*/C**

cuda@linux-p07k:~/NVIDIA_GPU_Computing_SDK/C> **make**

Comprobación de la compilación:

```
cuda@linux-p07k:~/NVIDIA_GPU_Computing_SDK/C> cd bin/linux/release

cuda@linux-p07k:~/NVIDIA_GPU_Computing_SDK/C/bin/linux/release> ls

alignedTypes          convolutionTexture  FunctionPointers
MonteCarlo            scalarProd                simpleTexture
threadMigration

asyncAPI              cppIntegration      histogram
MonteCarloMultiGPU    scan                      simpleTexture3D
transpose

bandwidthTest         dct8x8              imageDenoising       nbody
simpleAtomicIntrinsics    simpleTextureDrv      transposeNew

bicubicTexture        deviceQuery         lineOfSight
oceanFFT              simpleCUBLAS              simpleVoteIntrinsics
vectorAdd

binomialOptions       deviceQueryDrv      Mandelbrot
particles             simpleCUFFT               simpleZeroCopy
vectorAddDrv

BlackScholes          dwtHaar1D           marchingCubes
postProcessGL         simpleGL                  smokeParticles
volumeRender

boxFilter             dxtc                matrixMul            ptxjit
simpleMultiCopy           SobelFilter

clock                 eigenvalues         matrixMulDrv
quasirandomGenerator  simpleMultiGPU            SobolQRNG

concurrentKernels     fastWalshTransform  matrixMulDynlinkJIT
radixSort             simplePitchLinearTexture  sortingNetworks

convolutionFFT2D      FDTD3d              mergeSort
recursiveGaussian     simpleStreams             template

convolutionSeparable  fluidsGL            MersenneTwister
reduction             simpleTemplates           threadFenceReduction
```

5. ANEXOS:

ANEXO I: **Salida del programa deviceQuery** en el equipo del Laboratorio. Informa de las GPU instalada y de sus características.

```
./deviceQuery Starting...
 CUDA Device Query (Runtime API) version (CUDART static linking)
There is 1 device supporting CUDA
Device 0: "GeForce GTX 260"
  CUDA Driver Version:                           3.10
  CUDA Runtime Version:                          3.10
  CUDA Capability Major revision number:         1
  CUDA Capability Minor revision number:         3
  Total amount of global memory:                 1878720512 bytes
  Number of multiprocessors:                     24
  Number of cores:                               192
  Total amount of constant memory:               65536 bytes
  Total amount of shared memory per block:       16384 bytes
  Total number of registers available per block: 16384
  Warp size:                                     32
  Maximum number of threads per block:           512
  Maximum sizes of each dimension of a block:    512 x 512 x 64
  Maximum sizes of each dimension of a grid:     65535 x 65535 x 1
  Maximum memory pitch:                          2147483647 bytes
  Texture alignment:                             256 bytes
  Clock rate:                                    1.08 GHz
  Concurrent copy and execution:                 Yes
  Run time limit on kernels:                     Yes
  Integrated:                                    No
  Support host page-locked memory mapping:       Yes
  Compute mode:                                  Default (multiple
host threads can use this device simultaneously)
  Concurrent kernel execution:                   No
  Device has ECC support enabled:                No
deviceQuery, CUDA Driver = CUDART, CUDA Driver Version = 3.10, CUDA
Runtime Version = 3.10, NumDevs = 1, Device = GeForce GTX 260
PASSED
Press <Enter> to Quit...
```

Anexo II: **Salida del programa bandwidthTest** en el equipo de Laboratorio. Informa del ancho de banda disponible para los intercambios de información donde intervienen la CPU y la GPU instalada.

```
./bandwidthTest Starting...

Running on...

 Device 0: GeForce GTX 260
 Quick Mode

 Host to Device Bandwidth, 1 Device(s), Paged memory
   Transfer Size (Bytes)      Bandwidth(MB/s)
   33554432             3110.4

 Device to Host Bandwidth, 1 Device(s), Paged memory
   Transfer Size (Bytes)      Bandwidth(MB/s)
   33554432             2777.7

 Device to Device Bandwidth, 1 Device(s)
   Transfer Size (Bytes)      Bandwidth(MB/s)
   33554432             82700.7


[bandwidthTest] - Test results:
PASSED



Press <Enter> to Quit...
```

ANEXO III: **Salida del programa deviceQuery** en el equipo de desarrollo. Informa de las GPU instalada y de sus características.

```
ouadie@ouadie-desktop:~/NVIDIA_GPU_Computing_SDK/C/bin/linux/release$
./deviceQuery
./deviceQuery Starting...
 CUDA Device Query (Runtime API) version (CUDART static linking)
There is 1 device supporting CUDA
Device 0: "GeForce 9500 GT"
  CUDA Driver Version:                           3.0
  CUDA Runtime Version:                          3.0
  CUDA Capability Major revision number:         1
  CUDA Capability Minor revision number:         1
  Total amount of global memory:                 1073020928 bytes
  Number of multiprocessors:                     4
  Number of cores:                               32
  Total amount of constant memory:               65536 bytes
  Total amount of shared memory per block:       16384 bytes
  Total number of registers available per block: 8192
  Warp size:                                     32
  Maximum number of threads per block:           512
  Maximum sizes of each dimension of a block:    512 x 512 x 64
  Maximum sizes of each dimension of a grid:     65535 x 65535 x 1
  Maximum memory pitch:                          2147483647 bytes
  Texture alignment:                             256 bytes
  Clock rate:                                    1.38 GHz
  Concurrent copy and execution:                 Yes
  Run time limit on kernels:                     Yes
  Integrated:                                    No
  Support host page-locked memory mapping:       No
  Compute mode:                                  Default (multiple
host threads can use this device simultaneously)
deviceQuery, CUDA Driver = CUDART, CUDA Driver Version = 4249427, CUDA
Runtime Version = 3.0, NumDevs = 1, Device = GeForce 9500 GT
PASSED
Press <Enter> to Quit...
```

ANEXO IV: **Salida del programa bandwidthTest** en el equipo de desarrollo. Informa del ancho de banda disponible para los intercambios de información donde intervienen la CPU y la GPU instalada.

```
./bandwidthTest Starting...

Running on...

 Device 0: GeForce 9500 GT
 Quick Mode

 Host to Device Bandwidth, 1 Device(s), Paged memory
   Transfer Size (Bytes)      Bandwidth(MB/s)
   33554432             164.1

 Device to Host Bandwidth, 1 Device(s), Paged memory
   Transfer Size (Bytes)      Bandwidth(MB/s)
   33554432             203.7

 Device to Device Bandwidth, 1 Device(s)
   Transfer Size (Bytes)      Bandwidth(MB/s)
   33554432             7658.0


[bandwidthTest] - Test results:
PASSED


Press <Enter> to Quit...
```

ANEXO V: Configuración del Equipo del Laboratorio.

Dell Studio XPSTM 8100

Base: Studio XPS 8100 - procesador Intel Core i3 530 (2,93 GHz y 4 MB)

Sistema operativo Microsoft: Windows® 7 Home Premium original 64bit - 409486

Disco duro: Disco duro doble de 1,2 TB Raid 0 (2 x 640 GB - 7.200 rpm)

Memoria: SDRAM DDR3 de 4.096 MB de doble canal a 1.333 MHz [2 x 2.048]

Tarjeta de vídeo: Single 1.8GB Nvidia GeForce GTX 260