Listado de Mini-Proyectos

18 de junio - 27 de julio de 2012

1.1 Memcached

- Introducción:

Memcached es un sistema de caché en memoria distribuida entre varios nodos utilizado frecuentemente para acelerar servidores web dinámicos. Memcached permite almacenar resultados de consultas a bases de datos, páginas web dinámicas o cualquier otra cosa susceptible de ser cacheada mediante una interfaz simple de clave/valor ofreciendo de forma automática balanceo de carga entre varios nodos y política LRU.

- Objetivos:
  - Instalación de Memcached en una máquina real y preparación de algún benchmark ya disponible (e.g.: https://github.com/antirez/mc-benchmark).
  - Creación de un checkpoint con ese mismo benchmark para GEM5.
- Requisitos:
  - Entender el inglés escrito.
  - Conocimientos de arquitecturas web o diseño de aplicaciones cliente/servidor.
- Dificultad: Media

1.2 Instalación y preparación de benchmarks para GEM5

- Introducción:

GEM5 es un simulador recientemente liberado que se basa en el simulador de sistema completo M5 y en partes del simulador GEMS. GEM5 permite simular varias arquitecturas (Alpha, ARM, x86…) pero actualmente solo tenemos disponibles unos pocos benchmarks para Alpha.

- Objetivos:
  - Instalación de GEM5 en el cluster de GACOP.
  - Preparación de varios benchmarks para las arquitecturas x86, ARM y/o SPARC.
    - PARSEC
    - Otros (SPEC OMP, SPLASH, …)
- Requisitos:
  - Entender el inglés escrito.
  - Conocimiento de C++ (y otros lenguajes).
  - Conocimiento de las herramientas usadas en UNIX habitualmente para construir programas (guiones shell, make, gcc, …).
- Dificultad: Media

1.3 Evaluación del simulador Graphite

- Introducción:

Graphite es un simulador de arquitecturas multinúcleo paralelo y distribuido. Gracias a que es paralelo y distribuido podría ser útil para simular arquitecturas con cientos de núcleos.

- Objetivos:
  - Instalación Graphite en el cluster de GACOP, documentando minuciosamente el proceso.
  - Preparar algunos benchmarks para su ejecución:
    - PARSEC (instrucciones ya disponibles)
    - Otros (SPEC OMP, SPLASH, …)
  - Evaluar el rendimiento del simulador, comparándolo con GEMS / GEM5
- Requisitos:

        - Entender el inglés escrito.
        - Conocimiento de C++ (y otros lenguajes).
        - Conocimiento de las herramientas usadas en UNIX habitualmente para construir programas (guiones shell, make, gcc, …).
- Dificultad: Media

1.4 Sistema de nodos de computación autónomos alimentados por placas solares

- Requisitos:
    - Conocimientos de montaje de dispositivos y componentes.
    - Programación (Java, C, C++, lo que el alumno prefiera)
    - Conocimiento del SDK de Android (si se decide utilizar Android)
    - Gestión y administración de sistemas operativos.
- Objetivos:
    - Versión 1 (Raspberry Pi, PandaBoard, BeagleBoard u otra placa ARM):
        - Adquisición de un sistema integrado tipo Raspberry Pi
        - Adquisición de un sistema de paneles solares con potencia suficiente para alimentar el dispositivo.
        - Opcional: Adquisición de un sistema de suministro ininterrumpido intermedio (batería).
        - Montaje del sistema.
        - Configuración del sistema operativo del sistema integrado mediante scripts y/o procesos para que se de de alta como nodo autónomo en un servidor externo para proveer algún tipo de servicio (servidor web, nodo de cómputo, etc).
    - Versión 2 (Tablet con procesador ARM v9):
        - Adquisición de un tablet Android (a ser posible 4.0 con 1GB de RAM) de bajo coste (70–80€).
        - Adquisición de un sistema de paneles solares con potencia suficiente para alimentar el dispositivo.
        - Montaje del sistema.
        - Creación de una aplicación Android para el tablet para que se dé de alta como nodo autónomo en un servidor externo, con el fin de proveer algún tipo de servicio (servidor web, nodo de cómputo, etc).
- Dificultad: Media

1.5 Actualización del cluster de computación científica Castillo

- Introducción:

El grupo de investigación GACOP dispone de un cluster de computación científica con 268 cores y 268 GB de RAM que se utiliza principalmente para realizar simulaciones para estimar el rendimiento de arquitecturas propuestas. Las versiones del software instalado en este cluster están ya obsoletas, lo cual crea dificultades a la hora de utilizar nuevos simuladores. Además, la configuración de algunas de las herramientas que se utilizan se puede mejorar significativamente.

- Objetivos:
    - Adquisición de un nuevo servidor con conexión de fibra.
    - Instalación del sistema operativo, gestor de colas, firewall, etc.
    - Conexión a la NAS y asignación de espacio en la misma.
    - Configuración de los simuladores utilizados por los investigadores de GACOP.
        - Extraer del cluster 1 nodo de cada tipo.
        - Instalar el mismo sistema operativo que el servidor.
        - Configurar y comprobar que todo funciona.
        - Migrar el resto de nodos paulatinamente al nuevo servidor.
- Requisitos:
    - Gestión y administración de sistemas operativos Linux.
    - Conocimiento de NFS, NIS y sistemas de gestión de colas (SGE, Condor o similar).
- Dificultad: Alta (además, mucha gente depende de este sistema y no podemos arriesgarnos a romperlo o

hacerlo mal).

1.6 Sistema de backups para el cluster de computación científica Castillo

- Introducción:

El sistema de backups utilizado en Castillo está actualmente fuera de servicio debido a un fallo hardware, por lo que no se realizan copias de seguridad de forma regular (salvo las que haga cada usuario por su cuenta de sus datos). Debido a limitaciones del sistema usado hasta ahora, no vale la pena arreglarlo y es más práctico diseñar uno nuevo.

- Objetivos:
    - Diseño e implementación de una estrategia efectiva y lo más automática posible de backups utilizando:
        - Un servidor de backups el cual deberá poder soportar varios discos duros (ya sean internos o externos).
        - Varios discos duros de entre 1 TB y 3 TB.
        - Un software de backup ya existente (preferiblemente Bacula, que es el que ya usamos) o un conjunto de scripts fiable.
    - El sistema debe realizar copias de seguridad incrementales, de forma que podamos tener varias versiones para recuperar (por ejemplo, conservar los datos durante 15 días haciendo copias de seguridad cada día o cada dos días).
- Requisitos:
    - Gestión y administración de sistemas operativos Linux.
- Dificultad: Baja
- Notas: no es necesario ni conveniente que se utilice ningún tipo de RAID para los backups. Ya tenemos RAID en la NAS, que es donde sirve de algo.

1.7 Hacer que las máquinas del clúster se apaguen y enciendan cuando haga falta

- Introducción: El cluster de computación científica Castillo está compuesto por muchas máquinas que están normalmente siempre encendidas. La mayoría del tiempo estas máquinas se están utilizando a pleno rendimiento porque la cola de trabajos suele estar llena, pero a veces la cola se vacía (por ejemplo, durante un puente largo o cuando casualmente nadie está haciendo experimentos). Cuando ocurre esto último, convendría apagar las máquinas para ahorrar energía.
- Objetivos:
    - Implementar un mecanismo para que los nodos del cluster se enciendan automáticamente (utilizando Wake-On-Lan) cuando haya trabajos pendientes y se apaguen cuando no.
- Requisitos:
    - Gestión y administración de sistemas operativos Linux.
- Dificultad: Media

1.8 Actualización de la aplicación CUDA de los momentos de Zernike a la versión 4.2 de CUDA

- Requisitos:
    - Conocimientos de Programación paralela.
    - Programación (CUDA, C, C++)
    - Conocimiento del SDK de CUDA
- Objetivos:
    - Ejecución de un código CUDA dado que implementa los momentos de zernike.
        - Adaptación del código CUDA a la nueva versión 4.2
    - Dificultad: Media

1.9 Predicción de la estructura de proteínas mediante GPUs

- Requisitos:
    - Linux, C++, Python, Bash
- Objetivos:
    - Aplicar metodologías bioinformáticas desarrolladas en este grupo y que explotan la potencia de cálculo de las GPUs a la predicción de la estructura de proteínas.
    - Dificultad: Baja

1.10 Depuración y mejora de la interfaz de usuario del programa BINDSURF

- Requisitos:
    - Linux, C++, Bash
- Objetivos:
    - Depurar y mejorar el sistema de ficheros de entrada/salida del programa BINDSURF (http://f1000.com/posters/browse/summary/1089353), que explota la potencia de cálculo de las GPUs para el descubrimiento de fármacos para su ejecución óptima en un clúster de GPUs.
    - Dificultad: Baja

1.11 Depuración de un programa utilizado para la visualización de superficies moleculares

- Requisitos:
    - Linux, C++, Bash
    - Conocimientos medios de geometría computacional y de gráficos por ordenador
- Objetivos:
    - Depurar la integración de un método de computación de alto rendimiento en GPUs (MURCIA; Molecular Unburied Rapid Calculation of Individual Areas, http://cdn.f1000.com/posters/docs/81919452) para el cálculo de superficies moleculares con uno de los software estándar usado por miles de científicos en todo el mundo para la visualización de biomoléculas (Visual Molecular Dynamics, http://www.ks.uiuc.edu/Research/vmd/).
    - Dificultad: Media