Proyecto Final de Curso

Aplicación Distribuida y Paralela

El proyecto final de curso tiene como objetivo integrar los conocimientos adquiridos en la asignatura y concluir la misma construyendo una aplicación completa que incluya el diseño, implementación y evaluación de:

a) Un servicio y/o un protocolo que surja del estudio de las técnicas de programación de red y sistemas distribuidos (Cliente/Servidor, Cloud, P2P, GRID, MPI, etc), el cual permita distribuir la carga de trabajo a lo largo de los nodos participantes, y por otro;

b) Una solución de procesamiento de cálculo intensivo con algoritmos paralelos en cada uno de los nodos implicados (CUDA, OpenMP) que explote las características del Hardware disponible.

Se recomienda que se incluyan ambos paradigmas en el TF.

1. Aspectos a considerar:

- El trabajo puede ser desarrollado de forma individual o grupal (hasta 4 participantes).

- Los temas posibles se debatirán con el equipo docente a los efectos de poder dimensionar el problema (alcance y requisitos) y adecuarlo a la cantidad de participantes y al tiempo cuatrimestral de la asignatura.

- Se deberá relevar la literatura asociada al tema elegido para conocer en profundidad los antecedentes, el estado del arte, los problemas asociados, entre otros aspectos relevantes para así determinar y utilizar las técnicas de resolución adecuadas al problema.

- Una vez definido el tema, se deberá escribir una propuesta básica (no más de 5 hojas) que incluya la idea, una revisión bibliográfica mínima, una gráfica de los servicios implicados y la/s funcionalidad/es que se implementarán.

- Para aprobar la asignatura se deberá presentar un informe final basado en formato de reporte técnico[1]. A través de este, se debe explicar la propuesta completa, el estado del arte, el modelo definido, la experimentación realizada, los problemas encontrados, las soluciones implementadas junto con una evaluación del software y los resultados. Las técnicas usadas deberán estar sustentadas por la bibliografía. Este reporte será corregido por el equipo docente y – de ser necesario – será devuelto para hacer modificaciones antes de su aprobación.

[1] William Rapaport. "How To Prepare Technical Reports". http://www.cse.buffalo.edu/~rapaport/howtowrite.html

- Al final el curso, en las fechas previstas, se deberá presentar el trabajo, incluyendo una demostración del funcionamiento del software. El reporte debe entregarse al menos una semana antes de la fecha de presentación.

2. Aspectos Técnicos:

Como el objetivo es diseñar e implementar una aplicación distribuida y paralela, deberán tenerse en cuenta aspectos de Calidad de Servicio (QoS), disponibilidad y flexibilidad. Para ello, principalmente corresponderá implementar lo siguiente:

- Manejo de nodos: Se debe asegurar la conectividad y comunicación entre un conjunto de ordenadores que podrán entrar y salir del sistema. Para ello se utilizarán mensajes preestablecidos, reglas de propagación, búsqueda de nuevos usuarios, asignación de contenido, tareas, procesos o acciones según corresponda por la entrada o salida de nodos, entre otros.
- Manejo de múltiples clientes en simultáneos: Multithreading (Hilos)
- Explotación de capacidades de hardware: Diseñar las tareas a realizar de la manera más independiente posible entre ellas, con el objetivo de explotar al máximo el paralelismo y buscar aprovechar los recursos disponibles en cada nodo.
- Manejo de Sincronismo: Sincronización de procesos (definición de secciones críticas, progreso, controles, bloqueos, etc.) y de acceso a recursos compartidos (Base de datos, archivo de log, etc.)
- Distribución y procesamiento de tareas: Se debe tener en cuenta la algoritmia necesaria para la distribución y, si fuera necesario, la fragmentación y fusión del contenido. A su vez, tener en cuenta la administración de los nodos que cumplen la función de trabajadores.
- Manejo de transparencia: El usuario final no debería enterarse de la complejidad del sistema distribuido que subyace detrás de su implementación. Los errores deben enmascararse para que no lleguen directamente al usuario final. Los usuarios solicitarán servicios al sistema y recibirán resultados como si fuera un sistema centralizado.
- Contemplar aspectos de eficiencia: Incluir mecanismos de balanceo de carga (dinámicos, probabilísticos, estáticos, etc.), fragmentación de recursos entre nodos disponibles, distribución de tareas en base al estado de la red, entre otros.
- Contemplar alta disponibilidad: Incluir mecanismos de tolerancia a fallos (replicación, rollbacks), detección rápido de errores (Sensado), consistencia y coherencia de datos en datos replicados (actualización).
- Contemplar escalabilidad del sistema: Flexibilidad ante el crecimiento o decrecimiento del sistema (Autoorganización, distribución de recursos compartidos, directorios, gestión de tareas individuales, roles, asociación de nodos dependiendo de tareas, ajuste de variables de trabajo en base a la carga de trabajo y nodos disponibles, etc.).
- Registro de actividades del sistema: Deberán manejarse los eventos relevantes del sistema a través de registros en archivos de Logs (Conexiones aceptadas, información del cliente, resultado de las operaciones, errores, etc.). Se debe utilizar estructura bien definida y documentada, que permita obtener información estadística a través de

consultas y filtros (gráficas, tablas y recursos digitales)

- Manejo de aspectos de seguridad: Más allá que el prototipo no entre en producción en Internet, se debe tener en cuenta algunas características mínimas de seguridad (Cifrado en el paso de mensajes, HTTPS, protección de recursos contra accesos no autorizados, modelo de roles/permisos/usuarios, etc.)

- Métricas de Rendimiento: Finalmente, con el objetivo de evaluar las prestaciones y confiabilidad del sistema distribuido, deberán tomarse métricas como: Tiempo de respuesta, latencia, overhead del protocolo, Throughput (Número de trabajos por hora), utilización del sistema, fallas y errores, otros.