Presentación TP2

Sistemas Operativos
DC - UBA - FCEN

14 de mayo de 2019

Problema a Resolver

- Implementar un cómputo de forma distribuida, que será realizado por un conjunto de $n \geq 1$ procesos, que se podrán ejecutar en $m \geq 1$ máquinas. A estos procesos los llamaremos **nodos**.

Problema a Resolver

- Implementar un cómputo de forma distribuida, que será realizado por un conjunto de $n \geq 1$ procesos, que se podrán ejecutar en $m \geq 1$ máquinas. A estos procesos los llamaremos **nodos**.
- **Distribuido**:

Problema a Resolver

- Implementar un cómputo de forma distribuida, que será realizado por un conjunto de $n \geq 1$ procesos, que se podrán ejecutar en $m \geq 1$ máquinas. A estos procesos los llamaremos **nodos**.
- **Distribuido**:
  - Varios nodos, varios hilos, con envío de mensajes.

Problema a Resolver

- Implementar un cómputo de forma distribuida, que será realizado por un conjunto de $n \geq 1$ procesos, que se podrán ejecutar en $m \geq 1$ máquinas. A estos procesos los llamaremos **nodos**.
- **Distribuido**:
  - Varios nodos, varios hilos, con envío de mensajes.
  - MPI como herramienta para esta tarea.

Problema a Resolver

- Implementar un cómputo de forma distribuida, que será realizado por un conjunto de $n \geq 1$ procesos, que se podrán ejecutar en $m \geq 1$ máquinas. A estos procesos los llamaremos **nodos**.
- **Distribuido**:
  - Varios nodos, varios hilos, con envío de mensajes.
  - MPI como herramienta para esta tarea.
  - Excusa para este trabajo: **Blockchains**.

Contexto: blockchains

Se requiere entender qué es una **Blockchain**

Contexto: blockchains

Se requiere entender qué es una **Blockchain**

- Un gigantesco libro de cuentas en los que los registros (los bloques) están enlazados y cifrados para proteger la seguridad y privacidad de las transacciones.

Contexto: blockchains

Se requiere entender qué es una **Blockchain**

- Un gigantesco libro de cuentas en los que los registros (los bloques) están enlazados y cifrados para proteger la seguridad y privacidad de las transacciones.
- Una base de datos distribuida y segura (gracias al cifrado) que se puede aplicar a todo tipo de transacciones que no tienen por qué ser necesariamente económicas.

Contexto: blockchains

Se requiere entender qué es una **Blockchain**

- Un gigantesco libro de cuentas en los que los registros (los bloques) están enlazados y cifrados para proteger la seguridad y privacidad de las transacciones.
- Una base de datos distribuida y segura (gracias al cifrado) que se puede aplicar a todo tipo de transacciones que no tienen por qué ser necesariamente económicas.
- Una cadena de bloques, distribuida y segura, para hacer transacciones de diversa índole.

Estructura de un bloque

- unsigned int index: Índice del bloque en la cadena **(empieza en 1)**.
- unsigned int node_owner_number: Número del nodo que creó el bloque.
- unsigned int difficulty : Dificultad pedida para el bloque (definido más adelante)
- unsigned long int created_at: Fecha de creación en formato POSIX [1].
- char nonce[NONCE_SIZE]: String para resolver el **proof-of-work** (definido más adelante).
- char previous_block_hash [HASH_SIZE]: Hash del bloque anterior (en formato SHA256).
- char block_hash[HASH_SIZE]: Hash del bloque (en formato SHA256).

[1] https://es.wikipedia.org/wiki/Problema_del_ano_2038

Dinámica de la gestión de blockchains

- Objetivo de cada nodo: tener la mayor cantidad posibles de bloques que hayan sido creados por el mismo.
- Para agregar un nodo deberán **minarlo**, es decir, superar una prueba que requiere un cierto costo de cómputo (**POW: Proof-Of-Work**).
- Cada nodo mantendrá un diccionario con los bloques minados por él o comunicados a él por los otros nodos donde la clave de cada bloque será su propio hash.
- **Consenso:** Se aceptan nuevos bloques de otros nodos o no?
- No solo son importantes los bloques propios, sino que interesa tener los que la mayoría apoya. Consenso de **Nakamoto**.

Ejercicio

- Completar una función que comunique a todos los demás nodos cada nuevo bloque creado.
- Modificar un método pre-escrito para que cree un nuevo thread que mine bloques mediante una función proof_of_work, utilizando conocimientos de sincronización (evitar condiciones de carrera).
- Completar una función para que se respete las reglas de consenso.
- Analizar el protocolo descrito.

Por último: algunas pautas de entrega

- La entrega se realizará a través del **campus virtual**.
- Completar los datos de todos los integrantes del grupo y subir un archivo comprimido que deberá contener únicamente:
  1. El documento del informe (en PDF).
  2. El código fuente (NO incluir código compilado).
  3. Incluir tests mostrando la correcta implementación.
  4. Makefile para correr los test agregados.

Por último: algunas pautas de entrega

- Fecha límite: 01/06/2019 (OJO! es Sábado)
- Implementación libre de condiciones de carrera.
- Informe conciso.

¿Preguntas?