DEPARTAMENTO
DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico II

MPI

Sistemas Operativos
Primer Cuatrimestre de 2018

Integrante	LU	Correo electrónico
Gonzalo Ruarte	161/13	gonzalorpg@gmail.com
Fernando Nicolás Frassia	340/13	ferfrassia@gmail.com
Agustin Penas	668/14	agustinpenas@gmail.com

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria - (Pabellón I/Planta Baja)
Intendente Güiraldes 2160 - C1428EGA
Ciudad Autónoma de Buenos Aires - Rep. Argentina
Tel/Fax: (54 11) 4576-3359
http://www.fcen.uba.ar

Índice

1. Introducción

En este Trabajo Práctico desarrollaremos una implementación de sistemas distribuidos para simular una blockchain. Para esto, utilizaremos el protocolo MPI utilizando la biblioteca openMPI. En otras palabras, hablamos de sincronización entre procesos que no comparten los recursos y servicios de una misma computadora necesariamente (en nuestro caso sí).

2. Desarrollo

2.1. Análisis del protocolo:

El grueso del análisis de este trabajo está enfocado en la forma en la que sincronizamos los procesos, es decir, su protocolo. Esto es de vital importancia, ya que el mismo definirá las propiedades del sistema. Es decir: su performance, la posibilidad de network split, la facilidad para consensuar, la validez de los bloques minados. Dicho protocolo está descripto extensivamente en el enunciado del trabajo.

En primer lugar, podemos analizar la forma en que cada nodo avisa a los demás que acaba de minar un nuevo bloque. La forma en que lo hace es secuencial, es decir, él mismo avisa nodo por nodo comenzando por su siguiente respecto de los mpi ranks. De esta forma, garantizamos que el orden en que cada nodo broadcastea su nuevo bloque es único. Ahora bien, esto podría ser más performante, de la siguiente manera:

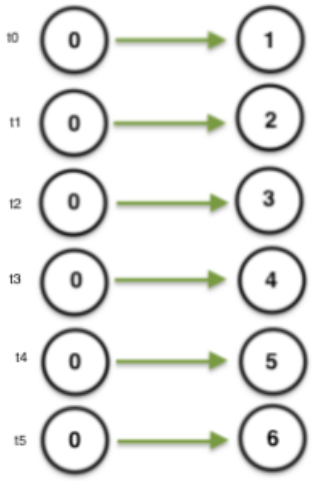

Figura 1: Forma en la que nosotros implementamos el broadcast de un nuevo bloque minado.

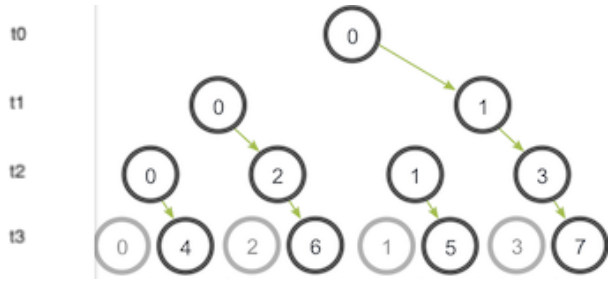

Figura 2: Forma más eficiente de broadcastear los mensajes de un nuevo bloque minado.

En la Figura 2 podemos ver que si logramos que los nodos que ya recibieron el mensaje lo forwardeen a los demás, podemos reducir el tiempo total de comunicación. Esto forma una estructura de AVL, que tiene altura mínima, es decir logarítmica con respecto a la cantidad de nodos. De esta forma el broadcasteo de un nuevo bloque tomaría O(log n) en lugar de O(n) - siendo n la cantidad de nodos en la red -.

En segundo lugar, para controlar que cuando un nodo broadcastea su nuevo bloque minado no escuche mensajes de nuevos bloques enviados por otros utilizamos un semáforo. Esto es importante porque sino se podrían dar condiciones de carrera con respecto al último bloque minado; es decir, mientras el nodo está broadcasteando podría cambiarse el puntero al último, con lo cual enviaría un bloque incorrecto.

En *validate_block_for_chain* y *proof_of_work* utilizamos un mutex para proteger el diccionario de bloques y el puntero al último bloque, ya que estos podrían cambiar mientras se los está utilizando (al validar un nuevo bloque recibido y/o al validar la cadena para migrar), produciendo así condiciones de carrera.

Respecto a la adición de un nuevo bloque a la cadena lo importante es que el thread principal escuche mensajes de nuevos nodos, reciba el bloque y lo agregue siguiendo las pautas del enunciado (y teniendo en cuenta las condiciones de carrera mencionadas anteriormente). A su vez, el caso más interesante es cuando hay que migrar la cadena ya que se requiere un intercambio de mensajes específico entre dos nodos particulares (el que desea migrar su cadena y el que debe proveerla) teniendo en cuenta que la cantidad de bloques enviados y pedidos no es fija, depende de VALIDATION_BLOCKS y/o de la cantidad de bloques que hay en la cadena del proveedor.

2.1.1. Convergencia de las cadenas

Con respecto a ésta propiedad es importante notar su relación con los siguientes elementos: DEFAULT_DIFFICULTY, VALIDATION_BLOCKS, la cantidad de nodos lanzados, y el tiempo que se ponga a correr el sistema, ya que de ellos dependerá la probabilidad de convergencia. En otras palabras, mientras más fácil sea minar un bloque, más probable es que un nodo se adelante a otro en más de VALIDATION_BLOCKS, porque el poder de cómputo de cada nodo influye más. Lo mismo sucede con disminuir o eliminar VALIDATION_BLOCKS (ya que si alguno se mantiene rezagado por una cantidad de bloques mayor a VALIDATION_BLOCKS no podrá migrar).

Por otro lado, si hay dos cadenas y cada grupo mina su nuevo bloque al "mismo"tiempo entonces las cadenas seguirían separadas. Nuevamente, la probabilidad de que cada grupo mine su nuevo bloque al "mismo"tiempo dependerá de DEFAULT_DIFFICULTY. Cabe aclarar que si esto es lo suficientemente difícil, es probable que eventualmente una cadena migre a otra, porque es probable que en algún momento un grupo mine su nuevo bloque, y lo broadcastee al otro grupo antes que ese grupo mine su nuevo bloque, con lo cual migraría su cadena.

Además, VALIDATION_BLOCKS cumple el rol de permitir una pequeña cantidad de pérdida de paquetes en la red de manera tal que no haya un network split. Es decir, si un nodo mina *n* bloques, pero los primeros *n-1* se pierden en la comunicación, cuando un nodo (cuya cadena es más corta) recibe el *n-ésimo* bloque chequea que la diferencia esté dentro del rango de VALIDATION_BLOCKS y migra su cadena en lugar de descartarla. Entonces el tamaño de VALIDATION_BLOCKS definirá la flexibilidad ante la pérdida de paquetes y la convergencia o divergencia de la cadena.

Por todo esto, creemos que sería una buena idea hacer que DEFAULT_DIFFICULTY sea proporcional al poder de cómputo presente en la red, de manera que el poder de cómputo de un nodo en particular (o de un grupo de nodos) tenga la menor influencia posible. Y, dado que el poder de cómputo global aumenta año a año, DEFAULT_DIFFICULTY debería ajustarse año a año también (de hecho, esto ocurre con *Bitcoin*).

Finalmente, cabe aclarar que no podemos dar garantías de convergencia, sólo probabilidades.

3. Conclusión

La sincronización entre procesos que corren en sistemas distribuidos añade más variables a tener en cuenta, con lo cual la tarea se vuelve más compleja. Darse cuenta de que hay una condición de carrera no es tan fácil a simple vista, e incluso en momentos pensábamos que ya no teníamos condiciones de carrera cuando en realidad todavía existían. Esto nos da a entender lo difícil que puede ser trabajar con sistemas distribuidos en el mundo real, a pesar de su gran ventaja de performance contra los sitemas no distribuidosl.

Al utilizar MPI el proceso de debugging se vuelve más difícil, porque puede introducir bloqueos cuando los nodos esperan un mensaje que puede haberse perdido, y es difícil saber si es el caso.

Respecto a la implementación, ésta nos sirvió como un pantallazo general sobre parte del funcionamiento de bitcoin y otras criptomonedas, lo cual nos parece relevante para nuestra formación como futuros profesionales.