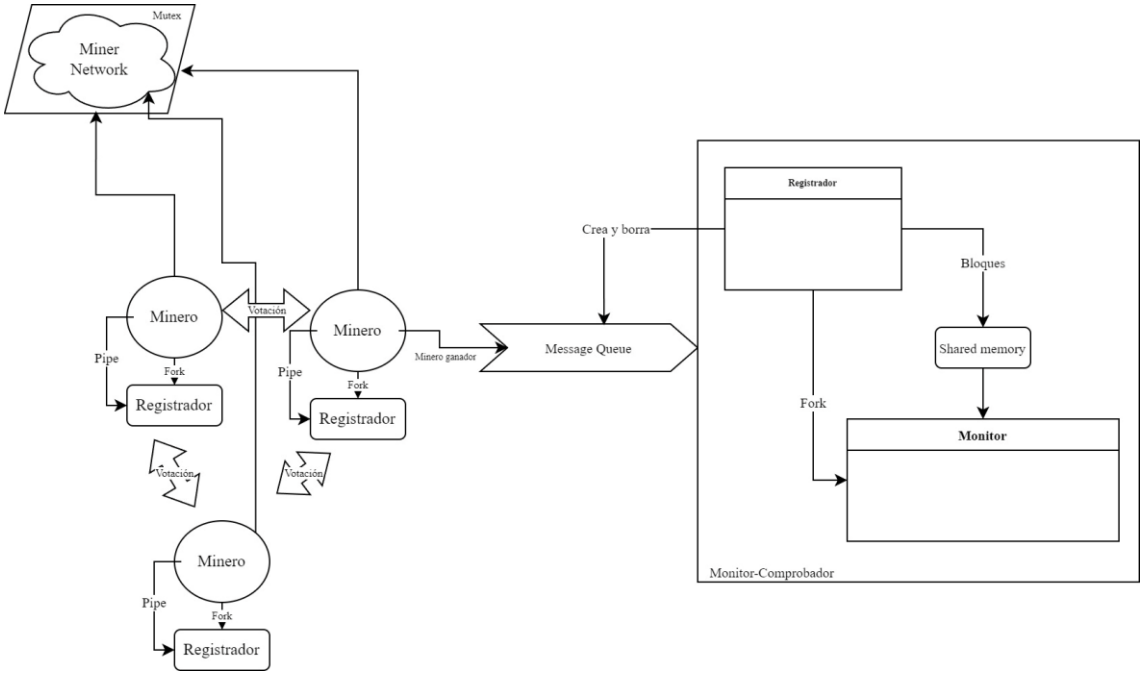
Memoria Miner Rush

Daniel Deneb Mecha Martín y Pedro Alemañ Quiles



1: Diagrama del sistema

El proyecto Miner Rush está conformado por varios elementos que detallaremos a continuación:

Miner network

Segmento de memoria compartida protegida por un mutex que almacena en todo momento la información del sistema. El primer minero en llegar la crea y el último minero en salir la destruye. La network es modificada por el proceso ganador para actualizar los bloques actuales y anteriores, y cada nuevo minero accede a ella para registrar su pid. Las principales dificultades que hemos encontrado a la hora de implementar la network son la detección del primer minero. Para ello utilizamos shm_open con la flag O_EXCL, de tal modo que si llega un minero y encuentra el segmento de memoria ya creado, sabrá que no es el primero. Esta operación es atómica y por tanto no hay problemas de condición de carrera.

La miner network almacena toda la información compartida entre mineros, como los pids de todos ellos, los votos de cada ronda, los monederos de cada minero (aunque una copia se almacena en el bloque), el último bloque y el actual.

Mineros

Los mineros entran en un proceso llamado Miner Rush. Durante este bucle, proceden al minado de un bloque mediante hilos y una vez hecho esto, a intentar proclamarse ganador. Esto es realizado mediante un semáforo llamado Ganador, que regula quién es el primer proceso en haber votado. Una vez hay un proceso ganador, el resto de procesos son también notificados mediante una señal. Después se entra en un proceso de votación por el cual los procesos emiten su voto sobre el bloque, y esperan a la señal de comienzo de ronda. Por otro lado, el proceso

ganador alterna esperas no activas con la comprobación de los votos, y una vez todos han votado (o ha pasado un número máximo de intentos) se envía el bloque a comprobador mediante la queue. Para garantizar la flexibilidad del sistema, el minero ganador detecta si existe dicha queue (porque la queue existe si y solo si existe un proceso comprobador). En caso contrario, no manda el bloque. El mayor problema en la implementación de los mineros ha sido la coordinación via señales de la miner rush con el resto de mineros, pero ha sido solucionado con un buen manejo de handlers y sigactions. Además, las señales se bloquean durante cada ronda para garantizar que las señales SIGINT y SIGALARM no interrumpan ningún proceso importante.

Registrador

El proceso registrador es el encargado de registrar en un fichero de datos los bloques minados por cada minero. Este proceso se comunica con su proceso padre, minero, con un pipe por el cual se transfieren los bloques, y después imprime en cada ronda el estado de la votación en el log. Hemos tenido problemas trabajando con las pipes, especialmente porque Registrador ha sido el último módulo en ser implementado, pero ha sido solucionado con una gestión correcta de los ficheros pipe. Adicionalmente, Registrador no tiene acceso a pow.h ni debería, ya que no es su misión comprobar si un bloque es correcto o incorrecto. Por ello, comprueba el número de votos del bloque y en función de eso imprime si es correcto o incorrecto. Por tanto, aunque en la práctica no suceda, es teóricamente posible que registrador anuncie un resultado diferente al validado por comprobador.

Comprobador y monitor

Los procesos comprobador y monitor son equivalentes a los de la práctica anterior, y ha costado muy poco adaptarlos. Las únicas modificaciones han sido crear monitor mediante un fork (con la consecuente espera necesaria), modificar la estructura del bloque impreso y cambiar el tamaño de la cola circular. Lo más importante de la nueva pareja comprobador/monitor es la gestión de la cola de mensajes, ya que es la interfaz con la red de mineros. El modo en que lo hemos implementado es que Comprobador está relacionado de forma inequívoca con la cola de mensajes, esto es, la cola de mensajes existe sólo cuando existe Comprobador, y viceversa. De este modo podemos comprobar si existe un comprobador tan sólo intentando acceder a la cola de mensajes y comprobar si existe, sin necesidad de semáforos o señales.

Todos los módulos anteriores han sido aislados y modularizados según su categoría en diferentes ficheros .c y .h, siguiendo un esquema de jerarquías, para garantizar la portabilidad a otros proyectos.

Limitaciones del sistema y pruebas realizadas

Una posible limitación del sistema es que mientras minero esté bloqueado en el envío de un mensaje, se cierre la cola de mensajes (por ejemplo porque comprobador termine por señal). Para este caso, hemos implementado un timed send, para que después de n tiempo minero siempre avance. Como la tasa de generación de bloques de minero es de un segundo, igual a la tasa de recepción de comprobador y menor al tiempo del timed send, esta función solo fallará en caso de que el caso extremo anteriormente nombrado suceda.

Otra limitación del sistema es la reducción del rendimiento que ocurre cuando un nuevo minero entra en la red: esto pasa porque el minero se registra en la red, y después espera el comienzo de una nueva ronda sin llegar a participar en ella. Durante este tiempo, el minero ganador ve no todos los votos han sido emitidos (el nuevo minero está en la red pero no vota) y por tanto tienen que esperar al número de intentos permitidos para poder avanzar. Una implementación alternativa sería que los nuevos mineros esperaran ante un semáforo para poder registrarse, que sería válida en Linux, pero no en otros sistemas como MacOs que utilizan semáforos débiles: en estos casos se podría causar inanición de tal modo que nuevos mineros no pudieran entrar en la red. Por tanto nos hemos decidido por esta primera implementación, mas exportable.

Otra decisión de diseño del proyecto es cuando en ocasiones aparece Voto 0/0. El minero ganador no tiene necesidad de votar en su propio bloque, considerándolo siempre correcto, y por tanto sólo los mineros perdedores entran en las votaciones. Esto quiere decir que el número de votantes siempre será igual a uno menos el número de mineros en el sistema.

Para probar el sistema durante las primeras fases de la práctica, hemos contrastado monitor contra los resultados impresos por cada minero (en ese momento minero todavía imprimía sus propios resultados), y después con los loggeados por registrador. Una parte crítica del proceso ha sido asegurarse de que el último minero siempre liberaba los recursos, pero lo hemos resuelto comprobando las zonas de memoria compartida después de la ejecución para garantizar que eran eliminadas correctamente.

Por otro lado, para comprobar el correcto funcionamiento de comprobador/monitor al principio de la práctica, hemos utilizado el minero de la práctica 3 para transmitir bloques y comprobar su funcionamiento.

Otras pruebas realizadas incluyen la comprobación de liberación de recursos mediante valgrind, la comprobación de coordinación entre mineros ejecutados simultáneamente y entre mineros ejecutados en momentos distintos. También hemos comprobado que los mineros con más tiempo de ejecución tienden a minar más monedas, que los mineros con más hilos tienden a minar más monedas, que el monitor puede comenzar y terminar su ejecución en cualquier momento, incluyendo durante el minado y en varias sucesiones, y que las señales interrumpen los procesos monitor y minero sin que ello cause problemas de coordinación o de liberación de recursos.