Universidad de Talca
Facultad de Ingeniería
Ingeniería Civil en Computación

Monte Carlo Distribuido: Token Ring

Sistemas Distribuidos

Luis Pozo, Felipe A. Román Miranda
froman10@alumnos.utalca.cl
lpozo10@alumnos.utalca.cl

Curicó, 01 de julio de 2016

Introducción

En el presente trabajo se pretende dar a conocer como se creo un agente para jugar ajedrez. Para esto, las decisiones del agente serán tomadas a través de un análisis Montecarlo Distribuido Tree Search. Además se utilizara Token Ring entre los trabajadores para asegurar la exclusión mutua para asegurar la escritura en el árbol.

Conocimientos Básicos

Tree Search

El árbol de búsqueda para monte carlos es un árbol el cual en cada nodo se almacena las victorias y derrotas de una simulación, además dado un nodo padre y un nodo hijo el nodo hijo tendrá sus victorias y derrotas, en cambio el nodo padre tendrá sus victorias y derrotas a las cuales también se le suman las victorias y derrotas del nodo hijo.

Para la construcción del árbol de búsqueda se deben implementar los siguientes 4 pasos:

- Selección: En este paso se recorre el árbol y se decide seleccionar un nodo con el cual se va a trabajar, para esto ocupa una fórmula que es la UTC. Para efectos de este proyecto se usó otra forma ya que no era el objetivo del proyecto.

- Expansión: En el paso de expansión lo que se hace es dado el nodo seleccionado se procede a crear un nuevo nodo hijo, en el cual se realizará la simulación.

- Simulación: El objetivo de simulación es tomar el nodo expandido y sobre este poder realizar una simulación en el cual una vez terminada la simulación procederemos a colocar las victorias y derrotas obtenidas.

- Retropropagación: Lo que se hace es dado el nodo con la simulación hecha, se busca el nodo padre de este y se agregan las victorias y derrotas obtenidas, luego se busca al padre del padre para agregar las victorias y derrotas y así sucesivamente hasta poder llegar al nodo raíz.

Token Ring

Este algoritmo trata de realizar un círculo lógico entre los distintos host, así pues un host recibirá el testigo que permita entrar en los recursos compartidos y entregará el testigo al siguiente host , que siempre será el mismo, entre todos formarán un círculo.
Este sistema siempre entregará el testigo al siguiente host independientemente de si el host tiene que utilizar o no los recursos compartidos.Por lo tanto, utilizar este sistema consumirá bastante más ancho de banda y el tiempo de entrada en el recurso compartido se incrementará dependiendo del número de host que forman el anillo.

Trabajo Realizado

Agente Montecarlo Distribuido Básico para Ajedrez

Un agente representa a un participante de un juego de ajedrez. Dado un estado de tablero, el agente debe realizar la mejor mejor para asegurar su triunfo. Cuando el agente utiliza un análisis Montecarlo, simula aleatoriamente el resultado de cada jugada posible asignándole un valor. Luego de haber analizado cada una de las jugadas, las compara para finalmente tomar una decisión.

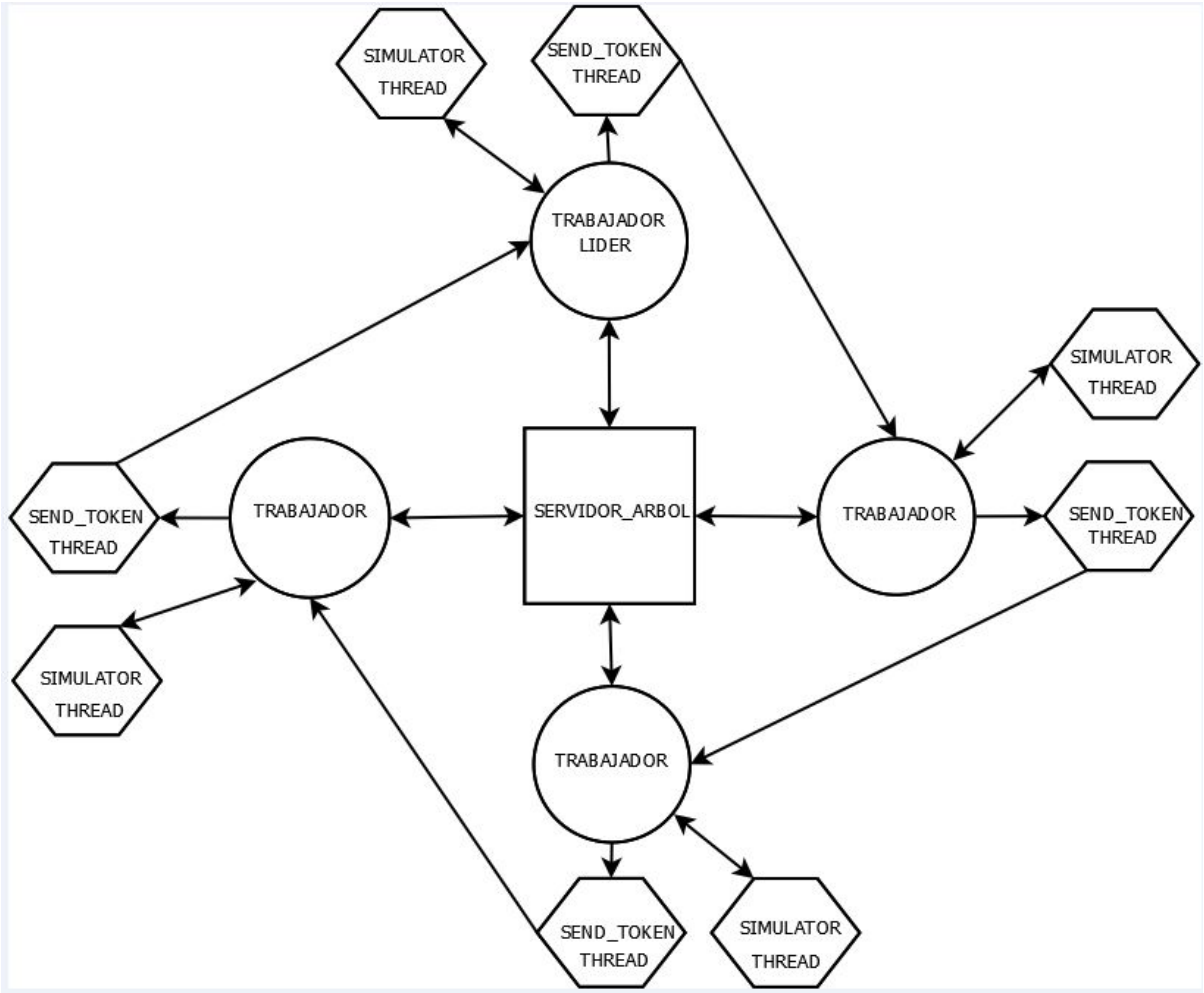
Agente Montecarlo Distribuido Tree Search Token Ring

En primera instancia, cada trabajador al ser iniciado, debe conocer a su vecino hacia la derecha además del servidor donde se encuentra el árbol. Luego se inicia el trabajador líder, quien se encarga de verificar el tiempo límite para tomar una decisión e inicia la entrega de token a sus vecinos.

Dado que el cliente pida un movimiento para un tablero en específico entregado por el agente, se iniciara un árbol de decisión de jugada para dicho tablero el cual tiene el objetivo de hacer muchas simulaciones y ir viendo las estadísticas de victorias y derrotas con las cuales saca un cociente el cual nos dirá que jugada tiene una probabilidad más alta en terminar en un juego ganado, entre más alto sea el cuociente mejor será la probabilidad. Para poder realizar esto se creará un nodo raíz el cual tendrá el tablero entregado y un arreglo de movimientos posibles, entonces el objetivo es que se implemente un nodo hijo por cada movimiento posible y al crear ese nodo se realicen múltiples simulaciones en las cuales se obtendrán las estadísticas de estas, una vez que el nodo raíz haya implementado todos sus hijos empezará a implementar los nodos hijos de los hijos los cuales ya tendrán una jugada más agregada y se harán más simulaciones en las cuales se guardaran las

estadísticas y se sumarán a las estadísticas del nodo padre y esta a su vez al nodo padre hasta llegar al nodo raíz el cual tiene un id igual a 1. Esta es la forma de ejecución del arbol el cual se seguirá implementando hasta que se cumpla el tiempo límite una vez que esto ocurra solo debemos enfocarnos en los nodos hijos del nodo raíz, los cuales los recorreremos y veremos el cuociente de cada uno, el que tenga un cociente mayor será el elegido y en base a ese nodo obtenemos el movimiento que él lo implementó el cual está guardado en la variable movimiento.

.



Escenario realizado en el proyecto.

Clase SistemaArbol

El árbol de búsqueda implementado se encontrará en un servidor, el cual se encargará de ir entregando los datos y luego recibir las simulaciones de cada trabajador y asi poder ir agregando los resultados al árbol.

Clase InterfazSistemaArbol

Esta clase entrega los métodos que serán utilizados para generar la comunicación con el SistemaArbol. Define cuatro método.

- recibirTablero: permite definir ese tablero como parte del nodo raíz.
- obtenerJugada: permite obtener el nodo sobre el cual se harán la simulaciones .
- guardarJugada: permite guardar un nodo simulado en el árbol.
- obtenerPrimerNivel: obtiene todos los nodos pertenecientes al primer nivel.

Clase ServidorSistemaArbol

Esta clase es utilizada para para mantener una conexión constante con el SistemaArbol.

Clase MonteCarloAgentInterface

Esta clase entrega los métodos que serán utilizados para generar la comunicación con el trabajador. Se define el método recibirToken, mediante el cual podrá saber que tiene el token para realizar las acciones según corresponda.

Clase MonteCarloAgent

Esta clase implementa toda la lógica definida por su interfaz. Cada vez que recibe el token, pregunta si el tiempo limite ya paso, en caso de ser asi y que a la vez sea el trabajador lider, elegirá la mejor jugada. De no ser asi pasará el token al siguiente trabajador.

En caso de que no se haya cumplido el tiempo limite, evalúa si hay una simulación, si no la hay solicita un nodo al SistemaArbol, y comienza la simulación en paralelo mientras pasa el token.

Clase MonteCarloAgentServer

Esta clase es utilizada para para mantener una conexión constante con cada trabajador.

Clase SendTokenThread

Esta clase permite enviar el token desde un trabajador a su vecino de forma paralela al hilo principal.

Clase SimulatorThread

Esta clase permite realizar las n simulaciones en paralelo al hilo principal del agente, para un nodo específico. Cada vez que se reciba el token en cada agente, se preguntará aca si las simulaciones ya fueron terminadas, si es asi se pide el nodo con las estadísticas.

Clase Arbol

Esta es la clase principal que implementara el nodo central que guarda el árbol, las variables que almacena dicho árbol son las siguientes.

- Cantidad de nodos: Se mantiene una cuenta de cuántos nodos tiene el árbol.
- Tablero: Se guarda el tablero inicial del árbol con el cual se empiezan a hacer las simulaciones.
- Nodos: Arreglo donde están contenido todos los nodos del árbol.

Clase Nodo

La clase nodo representara cada nodo del árbol el objetivo de esta clase es poder almacenar las variables que nos permitirán poder armar el árbol. Las variables necesarias son las siguientes:

- Id: Guardó el identificador del nodo, con el cual nos permitirá encontrar el nodo.
- Id padre: Guardó el identificador del nodo padre el cual se usará para reconstruir el árbol colocando las nuevas victorias y derrotas simuladas en el nodo.

- Victorias: Guardo todas las victorias obtenidas por el nodo además se le irán sumando las victorias obtenidas por los nodos hijos que tenga.

- Derrotas: Guardo todas las derrotas obtenidas por el nodo además se le irán sumando las derrotas obtenidas por los nodos hijos que tenga.

- Tablero: Guardo el tablero con el cual el nodo deberá hacer las simulaciones, también se usa para pasarle el tablero a los nodos hijos más una posible jugada para que empiecen a hacer las simulaciones.

- Hijos: Guardo la cantidad de hijos que tengo eso lo usó para comparar con las posibles jugadas si el número de hijos es menor seguiré haciendo hijos con posibles jugadas faltantes.

- Movimiento: Guardo el movimiento que ejecutara este nodo, se guarda para retornarlo después si es que fue la mejor simulación por tanto el mejor movimiento.

- Movimientos: Guardo un arreglo con todas las posibles jugadas a partir del tablero entregado, cada movimiento será usado en un nodo distinto de simulación.

- Cantidad de movimientos: Guardo la cantidad de movimientos posibles para un tablero dado.

Conclusión

Dada esta implementación completamos el funcionamiento de un token ring el cual maneja el token para poder acceder al árbol de búsqueda. Se aplicaron los conocimientos vistos en clases y las herramientas entregadas en los laboratorios las cuales fueron el uso de rmi y docker para crear contenedores y simular como tuvieramos muchos equipos conectados. Gracias a rmi podemos conectar estos equipos a través de invocaciones las cuales las logramos pasando la ip de cada contenedor, las cuales se obtuvieron con el comando en linux ifconfig. También cabe señalar que este tipo de implementación tiene debilidades como las vistas en las clases como si alguno de los nodos se cae el sistema no es persistente ya que los equipos quedan esperando un token el cual nunca les llegará porque se ha perdido la via de comunicación.