UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CIENCIAS

Práctica 5: Spinlocks

ALUMNOS

Rivera Zavala Javier Alejandro - 311288876
López Diego Gabriela - 318243485
Diego Martínez Calzada - 318275457

PROFESOR

Gilde Valeria Rodríguez Jiménez

AYUDANTES

Rogelio Alcantar Arenas
Gibran Aguilar Zuñiga
Luis Angel Leyva Castillo

ASIGNATURA

Computación Concurrente

Fecha de entrega: 12 de Abril del 2024

Tablas y gráficas

Rivera Zavala Javier Alejandro

Especificaciones de la computadora HP Compaq dc5850 Small Form Factor.

1. CPU (Frecuencia, Núcleo, Hilos)
   - Frecuencia: 2.300GHz
   - Hilo(s) de procesamiento por núcleo: 2
   - Núcleo(s) por socket: 2
2. RAM (Cantidad, Módulos, Frecuencia)
   - Cantidad: 8 GB de RAM
   - Módulos: 4 módulos de 2 GB DDR2 cada uno
   - Frecuencia: 1600MHz (0.6ns)
3. MotherBoard: Hewlett-Packard - Modelo 3029h
4. Procesador: AMD Athlon Dual 4450B - CPU @ 2.300GHz

Hilos	TAS	TTAS	BackoffLock	CLH	MCS	Alock
2	7.056	5.086	73.07	44.625	45.001	42.502
3	6.989	5.182	64.851	103.652	119.543	105.222
7	12.969	13.886	91.722	333.932	401.676	410.341
15	22.355	13.496	133.664	1350.232	1410.432	1398.320
21	10.529	15.45	149.564	2476.652	2506.774	2492.506
30	14.866	17.009	169.091	5403.101	5495.325	5450.788
50	25.324	13.728	233.532	7001.344	7509.401	7411.239

Tabla 1: Comparación de candados-Alejandro

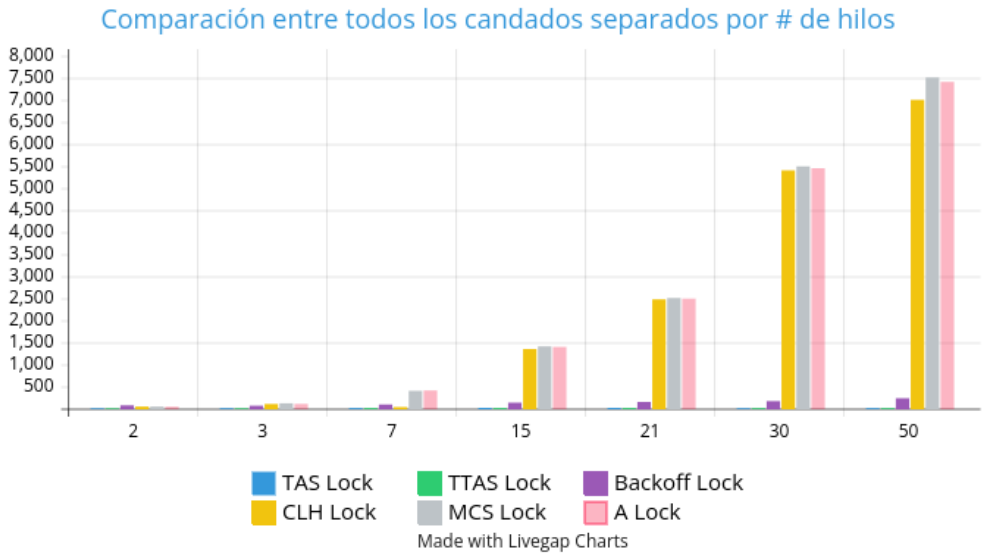

Figura 1: Gráfica comparación de candados-Alejandro

López Diego Gabriela

Especificaciones de la computadora Inspiron 5559 (06B2), Dell Inc.

1. CPU (Frecuencia, Nucleo, Hilos)
   - Frecuencia: 2.50GHz
   - Hilo(s) de procesamiento por núcleo: 2
   - Núcleo(s) por socket: 2
2. RAM (Cantidad, Modulos, Frecuencia)
   - Cantidad: 8 GB de RAM
   - Modulos:dos módulos de 4 GB cada uno
   - Frecuencia: 1600MHz (0.6ns)
3. MotherBoard: Inspiron 5559 (06B2)
4. Procesador: Intel(R) Core(TM) i7-6500U CPU @ 2.50GHz

Hilos	TAS	TTAS	BackoffLock	CLH	MCS	Alock
2	4.3	3.7	3.8	36.825	40.362	42.323
3	6.8	4.9	3.55	61.518	63.702	84.542
7	10.2	10.09	3.94	221.097	223.642	363.546
15	15.32	10.5	6.02	554.969	525.146	563.025
21	14.3	11.1	7.78	758.459	712.637	763.263
30	6.5	7.9	11.69	1,106.554	1,046.134	681.396
50	8.4	7.6	19.7	1,724.324	1,545.277	1,624.817

Tabla 2: Comparación de candados-Gabriela

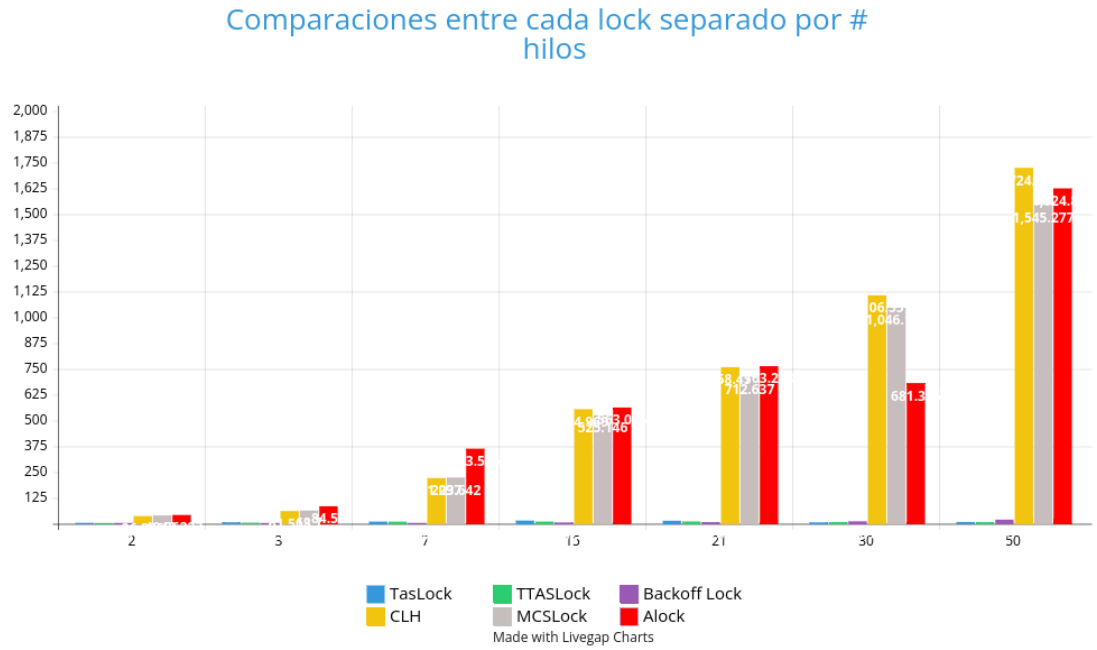

Figura 2: Gráfica comparación de candados-Gabriela

Martínez Calzada Diego

Especificaciones de la computadora HP Laptop 15-da1xxx:

1. CPU (Frecuencia, Nucleo, Hilos)
   - Frecuencia: 2.30GHz
   - Hilos: 4
   - Núcleos: 2
2. RAM (Cantidad, Modulos, Frecuencia)
   - Cantidad: 8 GB de RAM
   - Modulos: 1 de 8 GB
   - Frecuencia: 2133MHz (0.6ns)
3. MotherBoard: HP 8532
4. Procesador: Intel(R) Pentium(R) CPU 5405U @ 2.30GHz 2.30 GHz

Hilos	TAS	TTAS	Backoff	CLH	MCS	ALock
2	6.872	4.205	6.982	13.6	10.363	9.306
3	9.523	8.355	7.589	12.964	12.689	10.848
7	11.587	8.332	8.874	106.388	107.098	106.145
15	13.984	11.653	9.562	206.995	206.538	205.546
21	14.669	13.569	9.492	282.646	283.083	280.905
30	14.87	10.587	11.144	399.74	402.639	400.529
50	11.756	11.398	13.7	756.014	651.15	650.148

Tabla 3: Comparación de candados - Diego

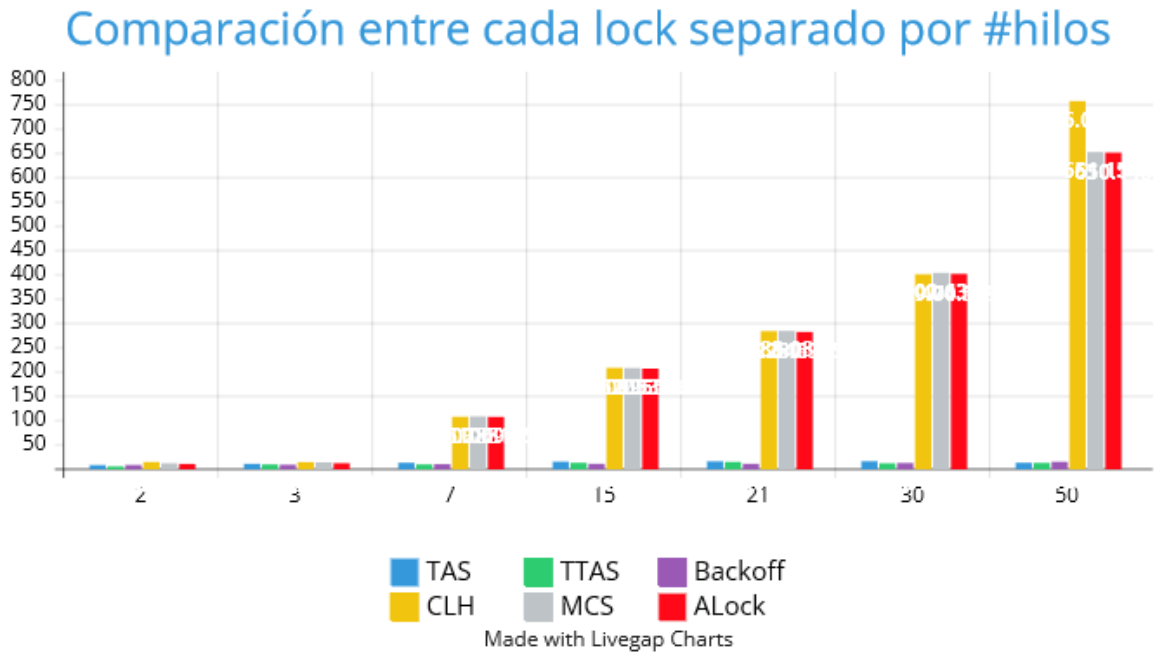

Figura 3: Tabla de comparación - Diego

Cuestionario

1. ¿Para qué sirve el método Yield? (yield())
   Se usa para que el calendarizador sepa que el hilo que uso ese método está dispuesto a ceder su tiempo o uso del procesador. Se usa con la intención de mejorar la progresión de un programa que usa subprocesos, evitando que se sobreutilicen recursos.

2. ¿Que es un atributo atómico? (En la biblioteca atomic de Java)
   Un atributo atómico en Java es una variable de instancia que permite realizar operaciones de forma atómica sobre ella. Que se puedan realizar dicho tipo de operaciones con tal atributo, quiere decir que se ejecutan como una sola unidad indivisible, como si ocurrieran de forma instantánea, de modo que cualquier otro hilo que no sea el que aplica una operación atómica, verá el valor del atributo antes o después de que se opere con él, pero nunca verá su valor intermedio.

3. Ventajas de usar atributos atómicos
   Los atributos atómicos se pueden leer y escribir atómicamente, además los objetos `Atomic` tienen otras operaciones atómicas como `compareAndSet`. Todo esto nos ayuda a la construcción de estructuras de datos concurrentes, y por lo mismo de contar con métodos atómicos se reduce la posibilidad tener errores relacionados con la concurrencia.

4. Desventajas de usar atributos atómicos
   Al usar variables atómicas y en situación de contención de hilos, podríamos generar costo adicional en el rendimiento total, es decir, en la complejidad de tiempo. En nuestra implementación debido a la sincronización que se lleva a cabo, el tiempo cada vez mayor cuando la cantidad de hilos va en aumento ya que cada hilo debe esperar su turno para poder acceder o modificar el recurso compartido y este no debe ser interrumpido.

5. ¿Qué locks cumplen con la propiedad de Justicia?

   - TasLock: No cumple con la propiedad de justicia.
     En la operacion getAndSet(true) dentro del while se establece el candado como disponible. Si varios hilos intentan adquirir el candado al mismo tiempo, por alguna extraña razón un hilo que llego después podría tomar el candado antes que otro que haya llegado antes. No tenemos alguna condición que impida que lo anterior no ocurra.
   - TTAsLock: Ocurre lo mismo que para el hilo TasLock, por ende, tampoco garantiza la propiedad de justicia.
   - BackOffLock: Mejora el rendimiento contra la contención de hilos, pero los hilos siguen accediendo de forma no justa
   - CLH: Sí, cumple con la propiedad de justicia. Utiliza una estructura de cola (con QNode y el candado locked) para organizar el acceso de los hilos conforme a su modo de llegada, a la sección crítica.
   - MCS: Ocurre lo mismo que en CLH (QNode y el candado locked).
   - Alock: Sí, cumple con Justicia. Cada hilo se le asigna un turno (slot) de acuerdo al orden que solicitarón el candado. Luego, esperan su turno correspondiente (sin intervenir antes) para poder acceder a la sección crítica. Después de acceder a la SC, libera el candado para que los demás hilos restantes puedan ingresar.

6. ¿Qué locks cumplen con la propiedad libre de Hambruna (starvation-free)?

   - TASLock: No, no se garantiza Starvation free. Como no tenemos algo que impida que un hilo o varios hilos consigan el candado repetidamente, podemos ocasionar que otros hilos que deseen entrar a la SC se encuentren en un modo de espera indeterminado.
   - TTAsLock: Tampoco se cumple Starvación free. Misma razón que TasLock
   - BackOffLock: No cumple con Starvation free. La estrategia de espera con retroceso trabaja mejor con la contención, sin embargo, nada impide que un hilo o varios se queden esperando de forma indefinida por acceder a la SC

- CLH: Sí cumple con starvation free. Al cumplir con Justicia, garantiza que habrá un orden de acceso para todos y cada uno de los hilos que deseen ingresar a la CS ya que a todos se les asigna un QNode y locked.
- MCS: Sí, ocurre los mismo que en CLH
- Alock: Sí, cumple con starvation free. Como cumple con Justicia, todos los hilos ingresan de forma ordenada a la SC y por ende, ninguno quedará esperando de forma indeterminada por la obtención del candado.

7. ¿Cuál es la implementación más eficiente? ¿Porqué crees que es así?
   En realidad no hay una implementación que sea inherentemente la más eficiente, según el contexto y la arquitectura del equipo, sobre la que corre el programa en el que se hace uso de tal o cuál spinlock, uno puede resultar mejor que otro. Por ejemplo, el candado TA Lock es eficiente en sistemas con alta carga y acceso frecuente a la sección crítica, mientras que candados como CLH Y MCS lo son más en sistemas multiprocesador debido a su estructura de cola que minimiza la contención entre los hilos. A pesar de lo antes dicho, cabe mencionar que la evidencia experimental sobre los 3 equipos de los miembros de este equipo, nos deja ver que para este caso en concreto, todo indica que TTAS es el más eficiente.

8. Por último, describe un problema en el que se pueda utilizar un candado visto en esta práctica Podría ser si se quiere implementar un sistema de compra de boletos en línea para viajes en autobus y se solucionaría con un CLHLock ya que se usa una cola, de esta manera se atendería a los usuarios como van accediendo, de manera justa. Además como, por lo menos para nosotros, el rendimiento del candado es algo pesado para varios hilos, al ser un autobus no corremos riesgo de recibir tantos clientes a la vez, pues no hay tantos asientos como en un avión por ejemplo.

9. Escribe lo aprendido en esta práctica así como diferencias respecto a las anteriores.
   En esta práctica, hemos visto y comparado diferentes tipos de spinlocks en el contexto de concurrencia en Java, hemos profundizado en el concepto mismo de que es un spinlock y sobre los distintos mecanismos de sincronizaicón que Java nos ofrece. Estudiamos varios tipos de spinlocks, como TAS, TTAS, Backoff, CLH, MCS y Alock, cada uno con sus características distintivas y las ventajas ó desventajas que tiene cada uno en términos de concurrencia, basados en las propeidades que ya habíamos estudiado con anterioridad, starvation free, justicia, etc., más aspectos nuevos que en prácticas anteriores no habíamos tomado en cuenta. Analizamos propiedades del rendimiento de los candados y como impacta en los mismos la arquitectura del sistema sobre el que corren. Por lo anterior, diríamos que la gran lección de esta práctica es precisamente la importancia de elegir el spinlock adecuado dependiendo del contexto de uso, considerando factores como la carga del sistema, el número de hilos y las propiedades de concurrencia requeridas. A diferencia de prácticas anteriores que involucraban candados como el de Peterson, esta práctica nos ha permitido explorar una gama más amplia de técnicas de sincronización y comprender mejor cómo afectan al rendimiento y la concurrencia en diferentes escenarios.

Referencias

- Oracle. (s.f.). Atomic Variables. Recuperado de https://docs.oracle.com/javase/tutorial/essential/concurrency/atomicvars.html
- Oracle. (s.f.). Class Thread. Recuperado de https://docs.oracle.com/javase/8/docs/api/java/lang/Thread.html#yield--
- Geeks for geeks. (10 de julio de 2021). Atomic Variables in Java with Examples. Recuperado el 11 de abril de 2024. https://www.geeksforgeeks.org/atomic-variables-in-java-with-examples/