Acerca de las decisiones de diseño y análisis de rendimiento del programa

Diseño del sistema:

La idea de diseño que se me ocurrió naturalmente al leer la consigna fue, primero, tener una clase que cumpla la función de una PlantaDeProducción donde suceda todo el flujo del programa, donde encontramos tanto a los ProductoresDePaquetes como a los ConsumidoresDePaquetes y luego hacer en modulos distintos los procesos y paquetes.

En este aspecto, usé un patrón de diseño del estilo Productor – Consumidor, con hilos que producen los datos (Paquetes en nuestro caso) y otros que se encargan de "consumirlos", es decir, pasar a cada pedido por cada una de las etapas que se piden.

La idea es que de forma sincronizada se tomen pedidos de la ColaDePedidos y se los pase primero por el procesamiento de pago y recién luego al empaquetado y envío, los cuales se hacen concurrentemente. Para esto se usó la palabra reservada Synchronized a la hora de tomar Paquetes de la cola. Se añadió también que la lógica del procesamiento del pago tenga un 80% de éxito.

Para poder crear al "departamento" de Produccion y Consumo, los implementé como 2 pools de hilos gestionados por un Executor Service a donde puedo mandar las tareas de creación y procesamiento de Paquetes.

Para la implementación de priorización de pedidos urgentes, usé una PriorityBlockingQueue, la cual además de ordenar los pedidos siguiendo un criterio (pasado como parámetro por el usuario), permite que si un hilo accede a una Queue vacía en un momento del tiempo, que se quede esperando a que ingresen Paquetes para procesar.

Con respecto a la administración de los hilos, se me ocurrió poner como parámetros modificables los números de hilos consumidores y productores a efectos de poder modificarlos en 1 solo lugar cuando quiera ver cómo varía el rendimiento del sistema.

Otros parámetros que consideré útil implementar fueron la cantidad máxima de olas de paquetes/ pedidos con la que se desea tratar. Esto lo hice con el objetivo de que aparezcan olas nuevas de pedidos de prioridad variable (50% urgente, 50% no urgente) mientras se procesan los que ya estaban en la PBQ. La cantidad de olas de paquetes que se generan es modificable, junto con el rango de pedidos generados en cada ola y el tiempo de pausa entre ola y ola.

Agregué, no solo en la producción de olas sino en cada etapa del proceso del Paquete, un tiempo que duerme al hilo que ejecuta la tarea del momento para simular un tiempo de procesamiento. Dicho tiempo también es modificable desde la PlantaDeProduccion.

Clasifiqué el flujo del programa según velocidad de procesos y cantidad de pedidos por ola en los siguientes tipos, a efectos de ver si se desea o ver la capacidad del sistema para soportar varios pedidos rápidamente (rápido-muchos) o si se desea ver el correcto funcionamiento de cada etapa con cada paquete (lento-pocos):

- Rápido – Muchos:
  - Velocidades para procesos: 10 min, 20 max
  - Cantidad de paquetes por ola: 50 min, 301 max con 5 olas
- Rápido – Pocos
  - Velocidades para procesos: 10 min, 20 max
  - Cantidad de paquetes por ola: 1 min, 6 max
- Lento – Pocos
  - Velocidades para procesos: 5000 min, 7000 max
  - Cantidad de paquetes por ola: 1 min, 6 max

Para garantizar un cierre ordenado, se hizo uso del método shutdown de los ExecutorService, el cual permite que los hilos terminen de procesar los pedidos que ya estaban procesando, pero no permite que se acepten más Paquetes para procesar.

Análisis de rendimiento:

En la carpeta de RegistrosRendimiento, hice 6 recordings donde los parámetros que varíe fueron: Numero de threads de productores y consumidores, duración de tarea y cantidad de paquetes generados en cada ola. La cantidad de olas quedó fija como 5 en todos los registros. Frente a la imposibilidad de abrir los recordings en Java Mission Control, usé la métrica de calcular el tiempo que se demoraba en procesar todos los pedidos generados por todas las olas.

Los resultados fueron:

Registros	Th.Cons	Th.Prod	Tmin. Tarea (miliSeg)	Tmax. Tarea (miliSeg)	Cant.Min Paq	Cant.Max.Paq	Cant. Total. Paquetes	Tiempo (seg)
N°1	5	2	10	20	1	6	14	31
N°2	20	5	10	20	50	301	817	31
N°3	100	100	10	20	500	1201	4774	30
N°4	50	100	10	20	500	1201	4384	29
N°5	10	200	10	20	800	2000	6439	29
N°6	1000	200	10	20	3000	5001	19528	32

La razón por la que los tiempos en cada registro se asemejan se debe seguramente a que el rango de tiempo de las tareas (del orden de los milisegundos) es mucho menor al rango de tiempo de aparición de las olas (del orden de los segundos), haciendo que un incremento o decremento en la cantidad de threads productores o consumidores no impacte en la relación entre lo que demoran las tareas y la generación de la próxima ola