# Capitulo 8

**Introducción**

Performance: capacidad del software de cumplir con los requisitos de tiempo, capacidad del sistema de responder a eventos o interrupciones que se ocacionen

En un sistema web, se traduce en el numero de transacciones que puede procesar en un minuto

Durante gran parte del tiempo en la ingenieria de software el rendimiento ha sido un factor determinante en la arquitectura, sin embargo los precios y el costo de desarrollo han hecho que otras carateristicas compitan con el rendimiento.

Sin embargo todos los sistemas tienen requisitos de rendimiento, incluso aunque no se expresen. Por ejemplo un procesador de texto no tiene un requisito de rendimiento explicito, pero sin duda todo el mundo estaria de acuerdo en qe esperar una hora (minuto o segundo) a que un caracter o imagen aparezca en pantalla seria inaceptable. Con lo cual el rendimiento sigue siendo un atributo de calidad fundamental para todo software

El rendimiento suele estar relacionado con la escalabilidad, el aumento del sistema debe estar acompañado de el aumento de la capacidad de trabajo y desempeño. Tecnicamentela escalabilidad es hacer que un sistema sea facil de cambiar y agrandarse y por lo tanto es una especie de modificabilidad.

### 8.1 Panorama general de performance

Un escenario de rendimiento comienza con un evento, para responder correctamente al evento se requieren recursos (incluyendo tiempo). Mientras esto sucede el sistema deberia poder simultaneamente atender otros eventos

Concurrencia

La concurrencia es uno de los conceptos mas importantes que un arquitecto debe entender, y uno de los que menos se enseña.

Concurrencia se refiere a las operaciones que ocurren en paralelo

La concurrencia se produce cada vez que su sistema crea un nuevo hilo, los hilos por definicion son secuencias independientes de control

Se debe tener en cuenta la concurrencia en un sistema multitarea, en un sistema que ejecuta en mas de un procesador, algoritmos paralelos, infraestructuras paralelas, algoritmos de programacion concurrentes.

En definitiva la concurrencia es una herramienta disponible para nosotros de muchas maneras

Permitir operaciones en paralelo mejora el rendimiento ya que los retrasos introducirdos en un hilo permiten al procesador progresar en otro hilo; sin embargo la concurrencia debe ser cuidadosamente manejada por el arquitecto

La gestion de la concurrencia con frecuencia se reduce a gestionar estados, una tecnica es el uso de bloqueos para forzar un acceso secuencial.

Las “race conditions” (no entendi que son) son bugs dificiles de manejar, que aparecen esporádicamente

Los eventos pueden llegar en patrones previsibles o con una distribución matemática o pueden ser imprevisibles. Los patrones de llegada pueden ser periódicos, estocásticos o esporádicos.

- Los eventos periodicos previsibles llegan en intervalos. Un evento puede llegar cada 10 segundos. En los sistemas en tiempo real la frecuencia es mas corta

- Eventos esporadicos: llegan de acuerdo a un patron que no es periodico ni estocástico

- Las llegadas estocásticas (estadísticas) significan que los eventos llegan de acuerdo a una distribución probabilistica

La respuesta del sistema se puede medir por

- la latencia: tiempo entre la llegada del estimulo y la respuesta del sistema

- deadlines: cuando un estado depende de otro anterior (queda trancado esperando por otro estado)

- rendimiento: numero de transacciones que el sistema puede procesar en una unidad de tiempo

- jitter (no se como traducirlo): variacion de latencia permitido

- numero de eventos que el sistema no pudo atender

Partes del escenario

Fuente de estimulo: estimulos que llegan por fuentes externas o internas

Estimulo: llegadas de eventos, el patron de llegada puede ser periodico, estocastico o esporadico

Artefacto: el sistema o uno o mas de sus componentes

Medio ambiente: Estado operacional del sistema, puede ser modos del sistema normal, de emergencia, carga pico o sobre carga

Respuesta: al procesar el sistema un evento, este puede causar un cambio de entorno por ejemplo, pasar de normal a sobrecarga

Medida de respuesta: tiempo que necesita el sistema para procesar los eventos que llegan (latencia), la variacion en este tiempo (jitter), numero de eventos que se pueden procesar dentro de un intervalo de tiempo (rendimiento), o una caracterizacion de los acontecimientos que no pudieron ser procesados (tasa de fallos)

**8.2 Tácticas de performance**

Las tácticas de rendimiento controlan el tiempo dentro del cual se genera una respuesta

En cualquier momento durante el período posterior a un evento, pero antes de que llegue la respuesta del sistema,el sistema está trabajando para responder a ese evento o el procesamiento se bloquea por alguna razón.   
Esto lleva a los dos componentes básicos del tiempo de respuesta: el **tiempo de procesamiento** (cuando el sistema está trabajando para responder) y el **tiempo de bloqueo** (cuando el sistema es incapaz de responder).

**Tiempo de procesamiento.** El procesamiento consume recursos, lo que lleva tiempo. Los eventos son manejados por la ejecución de uno o más componentes, cuyo tiempo gastado es un recurso. Los recursos de hardware incluyen CPU, almacenes de datos, ancho de banda de la comunicación en red, y la memoria. Los recursos de software incluyen entidades definidas por el sistema en fase de diseño .Por ejemplo, supongamos que un mensaje es generado por un componente . Podría ser colocado en la red , después de lo cual se llega a otro componente . Se coloca a continuación en la red ; transformado de alguna manera ; se procesa de acuerdo con algún algoritmo ; y se transforma para la salida ; se coloca en un búfer de salida , y se envía hacia adelante a otro componente , otro sistema , o algún actor . Cada uno de estos pasos consume recursos y tiempo y contribuye a la latencia global del procesado de ese evento . Diferentes recursos se comportan de manera diferente ya que su utilización se aproxima a su capacidad , es decir, a medida que se saturan . Por ejemplo , como una CPU vuelve se hace más pesada por la carga , por lo general el rendimiento se degrada de manera bastante constante . Por otro lado , cuando se empieza a quedar sin memoria , en algún momento la página de intercambio se convierte en abrumadora y el rendimiento se bloquea repentinamente

Tiempo de bloqueo . Un cálculo puede ser bloqueado a causa de no acceder a algún recurso necesario, porque el recurso no está disponible , o debido a que el cálculo depende del resultado de otros cálculos que aún no están disponibles

* Bloqueo de recursos . Muchos recursos sólo pueden ser utilizados por un solo cliente a la vez . Esto significa que otros clientes tienen que esperar a que el acceso a esos recursos. Estos eventos pueden estar en una sola corriente o en múltiples flujos . Corrientes múltiples que compiten por el mismo recurso o diferentes eventos en la misma corriente que compiten por el mismo recurso contribuyen a la latencia .
* La disponibilidad de recursos . Incluso en ausencia de la discordia , el cálculo no puede proceder si un recurso no está disponible. Falta de disponibilidad puede ser causada por el recurso que se está fuera de línea o por fallo del componente o por alguna otra razón . En cualquier caso, identificar los lugares donde no hay disponibilidad de recursos podría causar una importante contribución a la latencia global . Algunas de nuestras tácticas están destinadas a hacer frente a esta situación.
* La dependencia de otro tipo de cálculo . Un cálculo puede tener que esperar porque debe sincronizar con los resultados de otro cálculo o porque está esperando los resultados de un cálculo que se inició. Si un componente llama a otro componente y debe esperar a que el componente responda , el tiempo puede ser significativo si el componente es llamado en el otro extremo de una red (en lugar de co - situado en el mismo procesador )

**TACTICAS**

**Controlar la demanda de recursos**. Esta táctica funciona en el lado de la demanda para producir menor demanda en los recursos que tendrán que atender los eventos.

Una forma de aumentar el rendimiento es administrar cuidadosamente la demanda de recursos . Esto se puede hacer mediante la reducción del número de eventos procesados ​por la aplicación, o mediante la limitación de la velocidad a la que el sistema responde a los eventos . Además , hay una serie de técnicas para garantizar que los recursos que usted tiene se aplican juiciosamente :

Gestionar velocidad de muestreo- Si es posible reducir la frecuencia de muestreo a la cual se captura una secuencia de datos ambientales , se puede reducir la demanda , típicamente con alguna pérdida de fidelidad . Esto es común en los sistemas de procesamiento de señales , donde , por ejemplo , diferentes codecs pueden ser elegidos con diferentes frecuencias de muestreo y formatos de datos . Esta opción de diseño se hace para mantener los niveles predecibles de latencia , usted debe decidir si tener una fidelidad más baja pero constante flujo de información es preferible a la pérdida de paquetes de datos .

Limite respuesta evento- Cuando eventos discretos llegan al sistema ( o elemento ) demasiado rápido como para ser procesados ​​, los eventos deben estar en cola hasta que puedan ser procesados ​​. Debido a que estos eventos son discretos , que por lo general no es deseable disminuir la resolución de ellos . En tal caso, puede optar por procesar eventos sólo hasta un porcentaje máximo establecido, garantizando así un proceso más predecible cuando los hechos sean transformados . Esta táctica podría ser provocada por un tamaño de cola o la utilización del procesador en una medida superior a un cierto nivel de alerta . Si se adopta esta táctica y es inaceptable perder ningún evento , entonces usted debe asegurarse de que las colas son lo suficientemente grandes como para manejar el peor de los casos . Si , por el contrario , decide abandonar eventos, entonces usted tiene que elegir una política para el manejo de esta situación : Log de registro que los eventos quitados , o simplemente los ignoran ,Notifica a otros sistemas , a los usuarios o los administradores

Priorizar eventos- Si todos los eventos no tienen la misma importancia se puede imponer un esquema de priorización que rankea los eventos de acuerdo a que tan importante es responder a ellos. Si no se tiene suficientes recursos para procesar todas las solicitudes cuando llegan, las solicitudes con menor prioridad serán ignoradas. Ignorar eventos consume recursos minimos y da una mejora significativa en comparación a sistemas que responden a todas las solicitudes en todo momento.

Reducir la sobrecarga . El uso de intermediarios aumenta los recursos consumidos en el procesamiento de un flujo de eventos , por lo que la eliminación de ellos mejora la latencia. Este es un clásico equilibrio modificabilidad / rendimiento. La separación de preocupaciones , otro eje de la modificabilidad , también puede aumentar la sobrecarga de procesamiento necesario para dar servicio a un evento si conduce a un evento siendo administrados por una cadena de componentes en lugar de un solo componente . El contexto de cambio y los costos de comunicación entre componentes se suman, sobre todo cuando los componentes se encuentran en diferentes nodos de una red . Una estrategia para reducir la sobrecarga de cálculo es co - localizar recursos. Co - ubicación de alojamiento puede significar componentes cooperantes en el mismo procesador para evitar el retardo de tiempo de la comunicación de red , ya que puede significar poner los recursos en el mismo componente de software de tiempo de ejecución para evitar incluso el gasto de una llamada de subrutina . Un caso especial de la reducción de sobrecarga de cálculo es para llevar a cabo una limpieza periódica de los recursos que se han vuelto ineficaces . Por ejemplo , las tablas hash y mapas de memoria virtual pueden requerir recálculo y reinicialización. Otra estrategia común es ejecutar servidores de un único subproceso (por simplicidad y evitar la contención ) y dividir la carga de trabajo a través de ellos .

Tiempos de ejecución- Ponga un límite a la cantidad de tiempo de ejecución que se utiliza para responder a un evento. Para algoritmos iterativos que dependen de datos, limitar el número de iteraciones es un método para delimitar tiempos de ejecución . El costo es por lo general obtener un cálculo menos preciso. Si se adopta esta táctica, que tendrá que evaluar su efecto en la precisión y ver si el resultado es " suficientemente bueno". Esta táctica de gestión de recursos se combina frecuentemente con la gestión táctica de frecuencia de muestreo .

Aumentar la eficiencia de los recursos- Mejorar los algoritmos utilizados en áreas críticas disminuirá la latencia.

Administrar los recursos.

Esta táctica funciona en el lado de la respuesta para que los recursos disponibles funcionen con mayor eficacia en el manejo de las demandas formuladas.

Aumentar recursos- Mejor procesador, mas procesadores, mas memoria, velocidad de red más rápida, todas estas cosas reducen la latencia. El costo siempre es un factor importante a la hora de elegir el recurso a aumentar. En muchos casos suele ser la forma más barata e inmediata para mejorar la latencia.

Introducir concurrencia- Si las solicitudes pueden ser procesadas en paralelo, el tiempo bloqueado puede ser reducido. La concurrencia se puede introducir procesando conjunto de eventos en distintos hilos o creando hilos para ciertos conjuntos de actividades. Una vez que se tiene concurrencia agregar políticas de scheduling puede utilizarse para alcanzar las metas que se consideren deseables.

Mantener multiples copias de computos (computations)- Varios servidores en una arquitectura cliente servidor son replicas de computo. El propósito de tener replicas de los servidores es reducir la lucha por los recursos que se daría si todos los request llegaran a un único servidor. Mediante un balanceador de carga lse reparten las solicitudes entrantes entre los distintos servidores de acuerdo a algún criterio dado, por ejemplo round robin.

Mantener multiples copias de datos- Cachear es una tactica que implica mantener una copia de datos almacenada en un recurso que tenga una velocidad de acceso diferente a la original. Dado que la la información cacheada normalmente es una copia de datos existentes, mantener los datos sincronizados y consistentes pasa a ser una responsabilidad que el sistema debe asumir. Otra responsabilidad es seleccionar la información que será cacheada, algunos sistemas almacenan toda la información relativa a las ultimas solicitudes mientras que otros predicen cual será la información que será solicitada en el futuro para ya tenerla disponible.

Limitar el tamaño de la cola- Esto controla el numero máximo de solicitudes que pueden ser encoladas y como consecuencia los recursos utilizados para procesar las solicitudes entrantes. Si se adopta esta táctica se tiene que establecer una política de que sucede si se alcanza el limite de la cola, y se debe evaluar si es aceptable no responder algunas solicitudes. Esta táctica normalmente se utiliza junton con Limitar respuesta a eventos

Schedule recursos- Siempre que se da una “pelea” por un recurso el recurso debe ser programado. Los procesadores son scheduleados, los buffers son scheduled y la red es scheduled. La meta de esta táctica es entender las características del uso de cada uno de los recursos para establecer una estrategia que sea compatible con esto.

**8.3 check list para performance**

**8.4 Resumen**

El rendimiento es sobre el manejo de los recursos del sistema frente a determinados tipos de demanda para lograr el comportamiento de tiempo aceptable. La performance puede medirse en términos de rendimiento y latencia para los sistemas en tiempo real, aunque el rendimiento es generalmente más importante en los sistemas interactivos, y la latencia es más importante en los sistemas embebidos. El rendimiento puede mejorarse mediante la reducción de la demanda o mediante la gestión de recursos de manera más apropiada. La reducción de la demanda tendrá el efecto secundario de reducir la fidelidad o negarse a atender algunas peticiones. La gestión de los recursos más adecuadamente se puede hacer a través de la programación, la replicación, o simplemente aumentar los recursos disponibles.