Departamento de Ingeniería de
Sistemas y Computación

Arquitectura de Software
Desempeño

CSOF 5204

Universidad de los Andes

Agenda del día

- Introducción
- Preocupaciones
- Desempeño en diferentes puntos de vista
- ¿Cómo lograr desempeño?
- Tácticas arquitecturales
- Patrones arquitecturales

Introducción

- **Desempeño:** Rapidez con que el sistema realiza su actual carga de trabajo.

- **Escalabilidad:** Se enfoca en la previsibilidad del desempeño del sistema a medida que la carga de trabajo aumenta.

Agenda del día

- Introducción
- Preocupaciones
- Desempeño en diferentes puntos de vista
- ¿Cómo lograr desempeño?
- Tácticas arquitecturales
- Patrones arquitecturales

Grado de reacción

- La rapidez del sistema en responder a la carga de trabajo de rutina → Segundos

Ejemplo: Bajo una carga de 350 transacciones de actualización por minuto, el 98% de las transacciones debe retornar el control al usuario dentro de los siguientes 3 segundos siguientes al envío de la solicitud vía Web

Tiempo de entrega

- Tiempo tomado para completar tareas largas
  → Minutos u horas

Ejemplo: Debe ser posible sincronizar el sistema con todas las estaciones de monitoreo de la línea de producción y reiniciar la base de datos para reflejar el estado actual de dicha línea en no más de 5 minutos. Ninguna transacción de actualización será procesada durante dicho período de sincronización.

Rendimiento

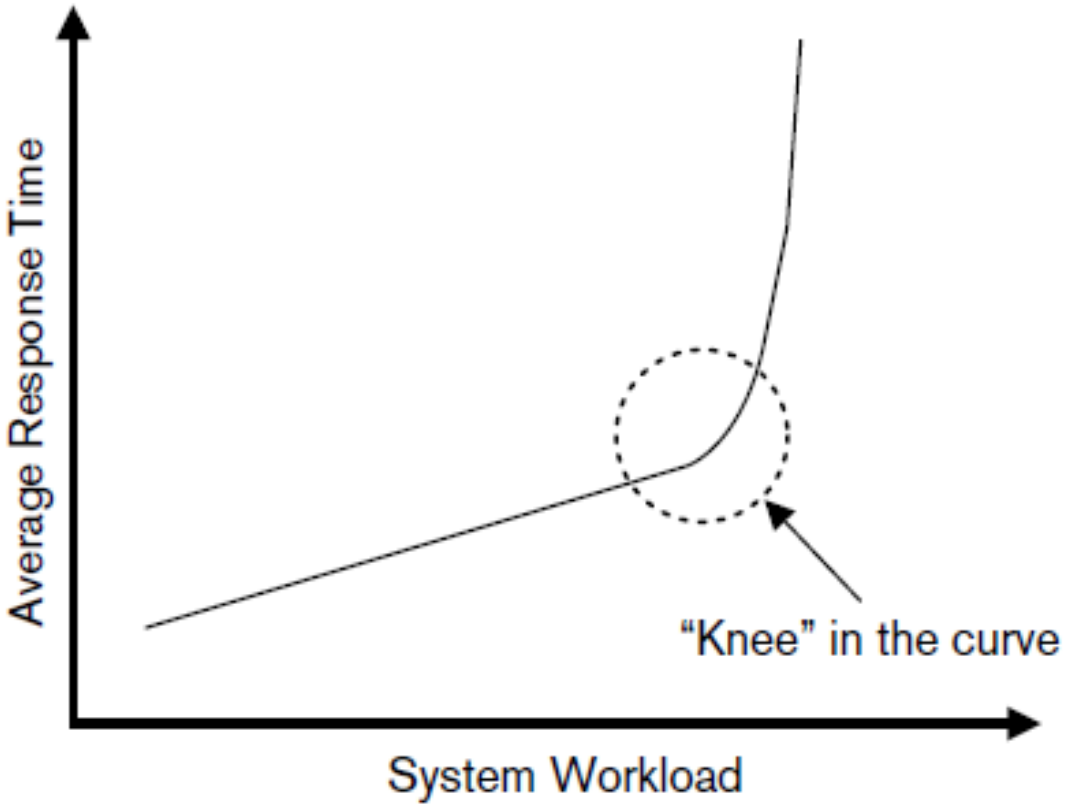
- Cantidad de carga de trabajo que el sistema es capaz de manejar en un periodo de tiempo
- Entre más corto sea el procesamiento de una transacción, más alto es el rendimiento del sistema
- A medida que la carga del sistema aumenta, el tiempo de respuesta de las transacciones individuales tiende a aumentar también

Previsibilidad

- Transacciones similares deben ser completadas en cantidades muy similares de tiempo, independientemente de cuando son ejecutadas.

Ejemplo: Un sistema de servicio al cliente utilizado por agentes comerciales para obtener información de clientes que efectúan solicitudes telefónicamente, necesita ofrecer un tiempo predecible por transacción de 1 a 5 segundos

Comportamiento bajo carga



Tomado de [3] pág. 404

Agenda del día

- Introducción
- Preocupaciones
- Desempeño en diferentes puntos de vista
- ¿Cómo lograr desempeño?
- Tácticas arquitecturales
- Patrones arquitecturales

Punto de vista funcional

- El análisis de desempeño revela la necesidad de cambios y compromisos para lograr una estructura funcional adecuada.

- Los modelos de esta vista proporcionan entradas para la construcción de modelos de desempeño.

Punto de vista de información

- Identificar recursos compartidos y los requerimientos transaccionales para cada uno de ellos.

- Sugerir elementos de esta vista que pueden ser replicados o distribuidos.

Punto de vista de concurrencia

- Identificar problemas como el uso excesivo de un recurso clave en un momento determinado.
- Resultar en que la concurrencia se convierta en el elemento de diseño más importante.
- Los elementos de esta vista pueden proporcionar métricas de calibración para los modelos de desempeño.

Punto de vista de despliegue

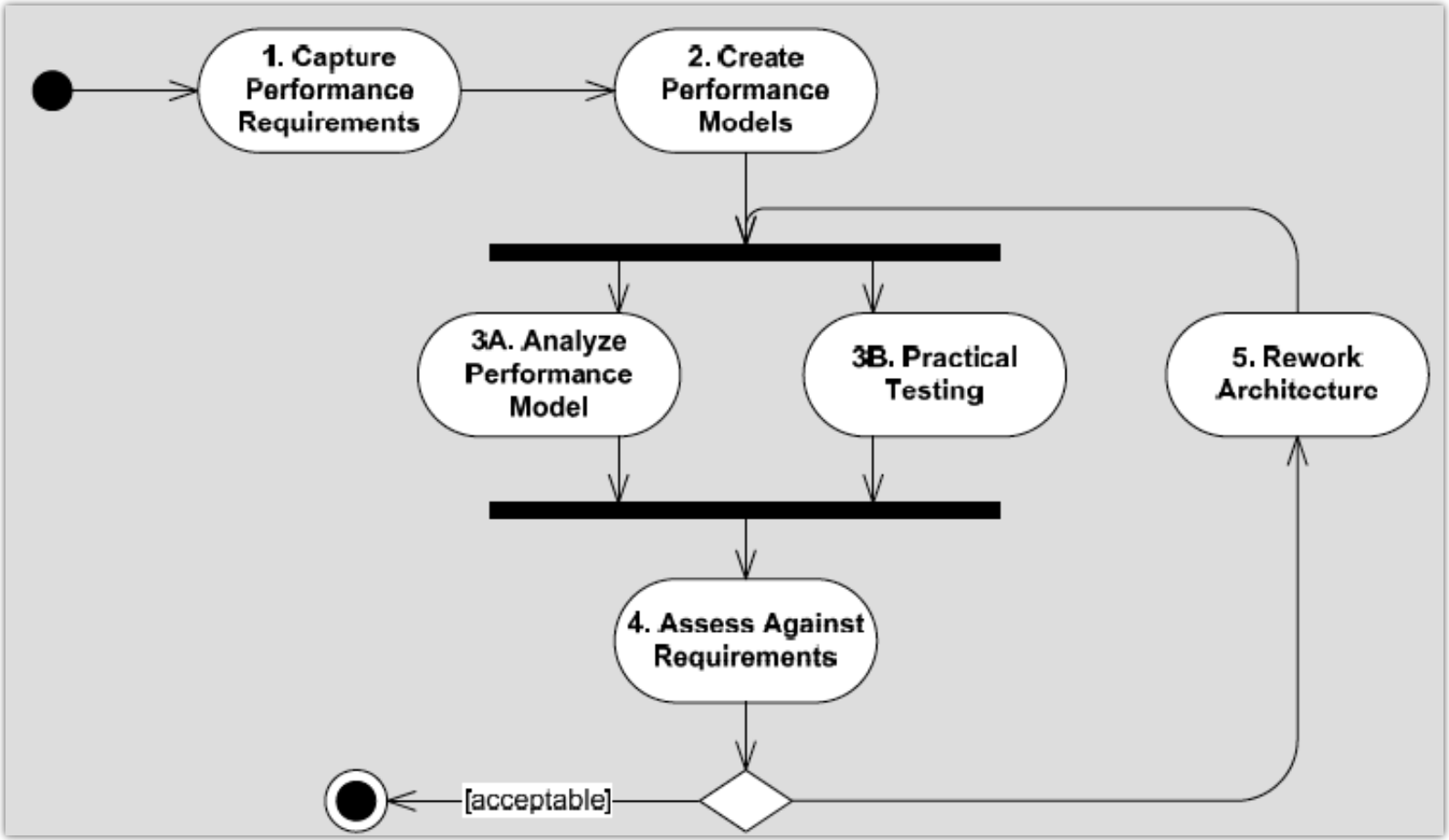
- El análisis de desempeño normalmente sugiere cambios y refinamientos en el ambiente de despliegue del sistema.

- A su vez, muchas partes de los modelos de desempeño son derivados de los contenidos de esta vista, ya que proporcionan un número de métricas críticas de calibración.

Agenda del día

- Introducción
- Preocupaciones
- Desempeño en diferentes puntos de vista
- ¿Cómo lograr desempeño?
- Tácticas arquitecturales
- Patrones arquitecturales

Proceso de análisis de desempeño



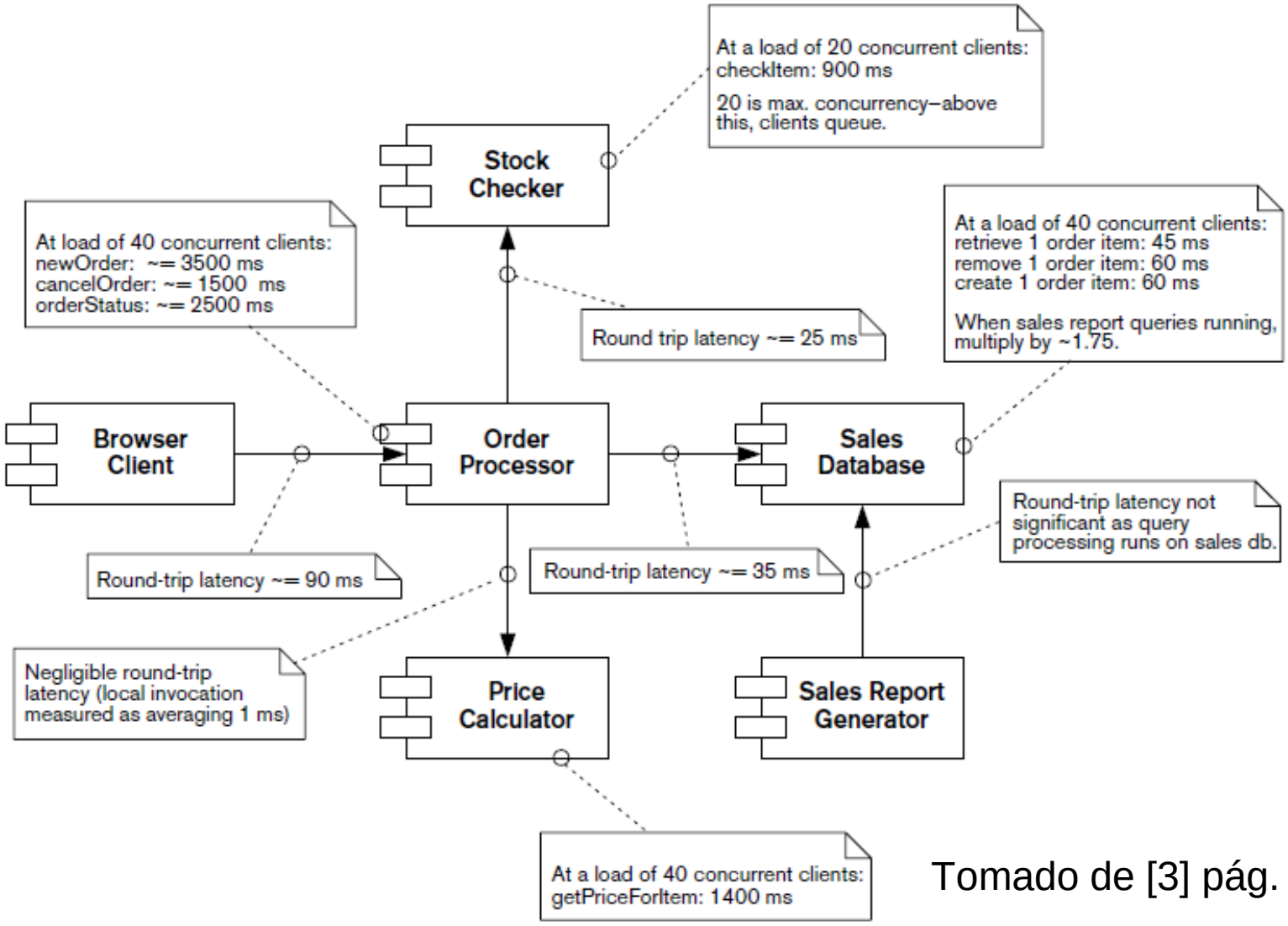
Tomado de [3] pág. 405

Proceso de análisis de desempeño (2)

1. Capturar los requerimientos de desempeño
   - Especificar tiempos de respuesta requeridos
   - Especificar requerimientos de rendimiento
   - Especificar requerimientos de escalabilidad

2. Crear **modelos de desempeño**
   - Identificar la estructura de desempeño crítica (procesos, nodos, enlaces, almacenamiento)
   - Identificar las métricas claves de desempeño
   - Estimar las métricas de desempeño

Proceso de análisis de desempeño (3)



Tomado de [3] pág. 409

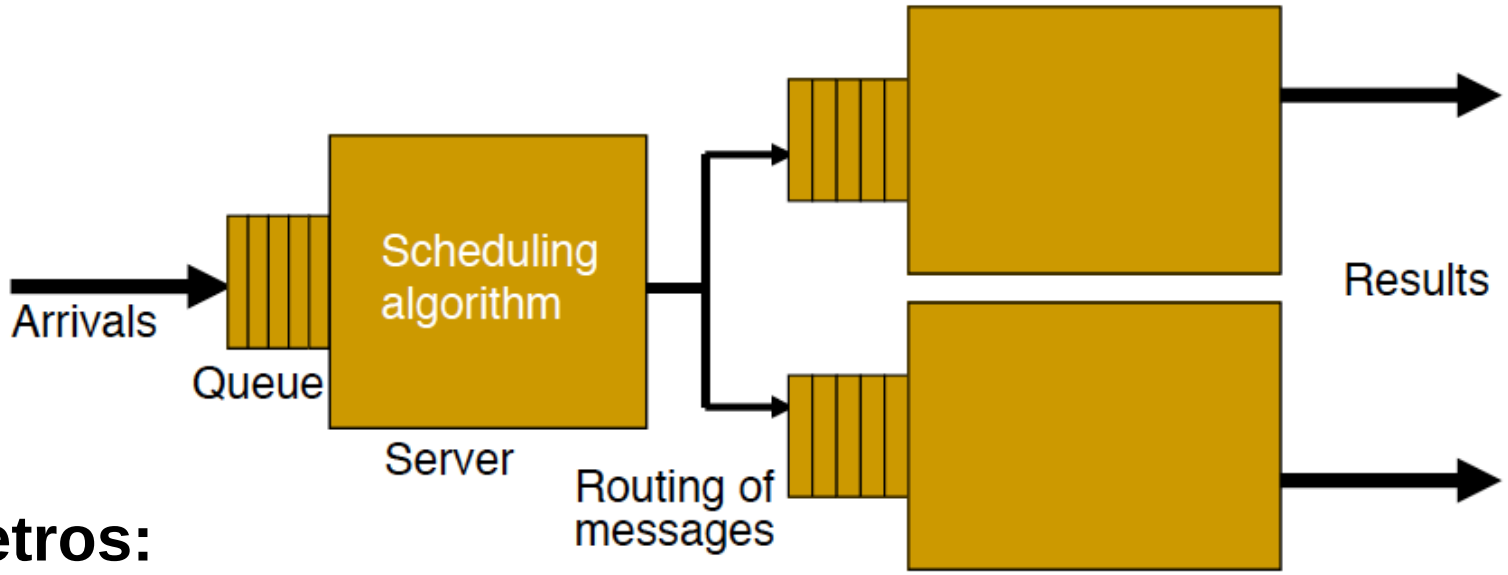
Proceso de análisis de desempeño (4)

3. Analizar los **modelos de desempeño**
   - Caracterizar la carga de trabajo: Priorizar y estimar el volumen de cada tipo de requerimiento que el sistema debe manejar
   - Estimar el desempeño
4. Ejecutar pruebas prácticas
   - Medir las métricas de desempeño
5. Evaluar contra los requerimientos
6. Iterar sobre la arquitectura

¿Qué se tiene en cuenta al analizar un modelo de desempeño?

- El objetivo es generar una respuesta a un **evento** que llega al sistema dentro de alguna **restricción** de tiempo, teniendo en cuenta:
  - Consumo de recursos
    - Unidad Central de Procesamiento (CPU), Unidades de almacenamiento de datos, Ancho de banda, Memoria
  - Tiempo de bloqueo de recursos
    - El recurso está en uso, no está disponible, o depende del resultado de otros cálculos que no están disponibles aún.

Modelo de desempeño



Parámetros:

- Arrival rate
- Scheduling algorithm
- Service time
- Topology
- Network bandwidth
- Routing algorithm

- **Latencia** (tiempo para ejecutar un evento) Puede *sólo* ser afectada por el cambio en uno de los parámetros

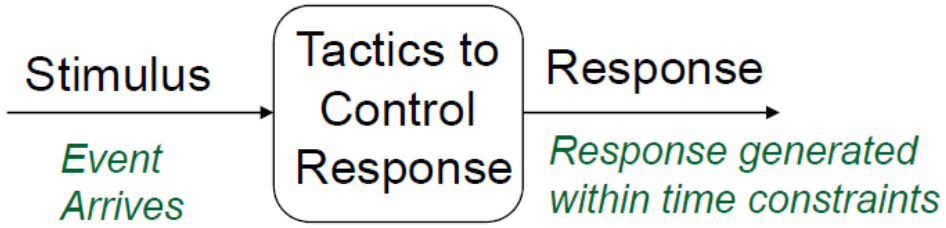
Parámetros a tener en cuenta en un modelo de desempeño

- **Arrival rate** → Restringir acceso a recursos
- **Service time**
  - Incrementar la eficiencia computacional (algoritmos)
  - Reducir la sobrecarga (reducir comunicación entre procesos, usar pools de threads, utilizar pool de conexiones a bases de datos, etc.)
  - Utilizar procesadores rápidos
- **Scheduling algorithm** → First-Come First Served (FCFS), prioridades dinámicas, round-robin, etc.
- **Topology** → Adicionar/eliminar procesadores
- **Network bandwidth** → Redes rápidas
- **Routing algorithm** → Balanceo de carga

Agenda del día

- Introducción
- Preocupaciones
- Desempeño en diferentes puntos de vista
- ¿Cómo lograr desempeño?
- Tácticas arquitecturales
- Patrones arquitecturales

Tácticas arquitecturales para favorecer desempeño



- Según el modelo de desempeño las tácticas se dividen en tres categorías:
  - Control de demanda de recursos (reducir o controlar)
  - Administración de recursos
  - Arbitrar recursos (control por programación)

Control de demanda de recursos

- Controlar el número de recursos requeridos
  - *Aumentar eficiencia computacional*
  - *Reducir la carga de trabajo computacional*
- Controlar el número de eventos a procesar…
  - *Administrar la llegada de los eventos*
  - *Controlar la frecuencia de muestreo (colas de eventos)*

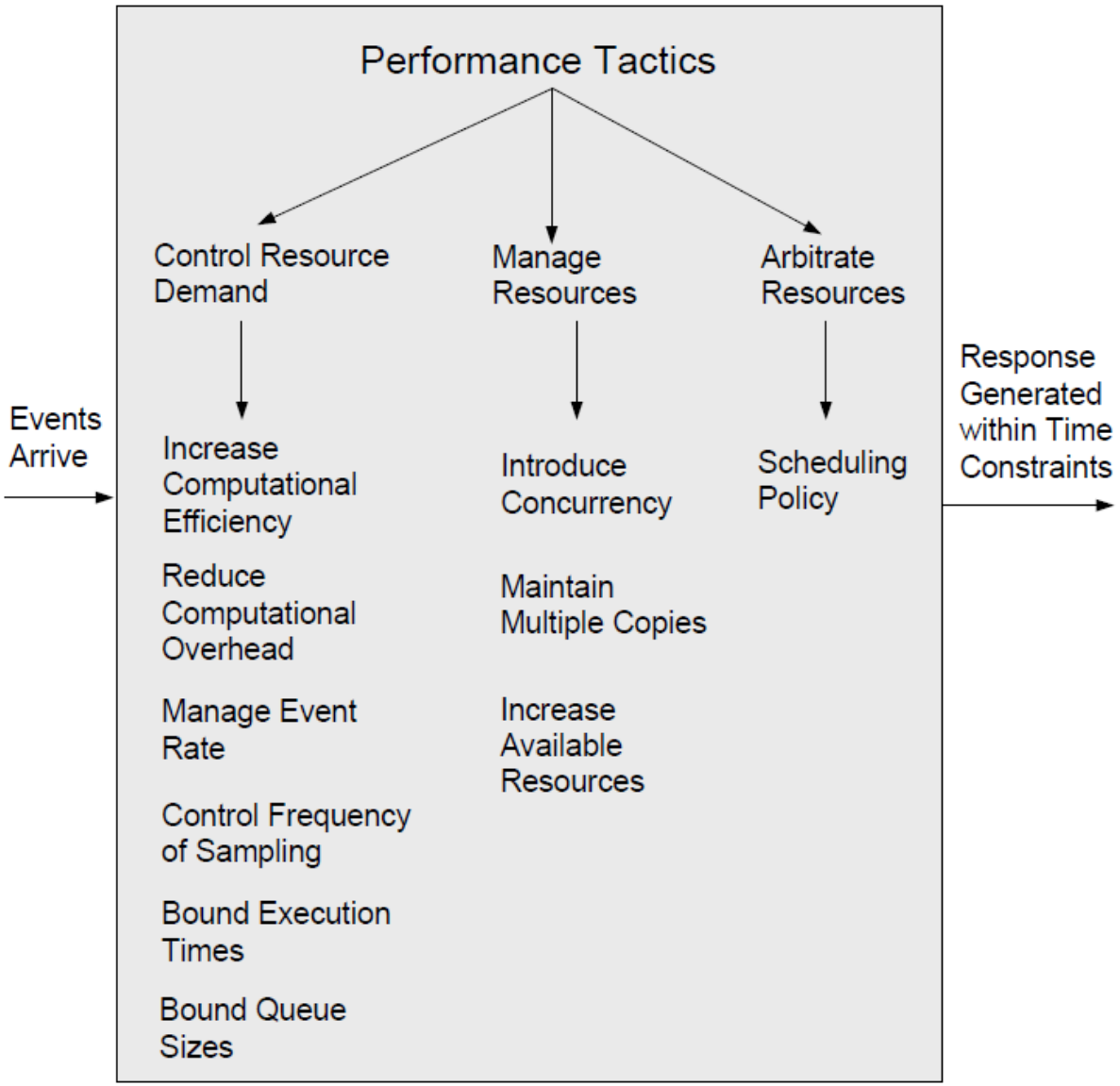
Administración de recursos

- Introducir concurrencia
  - *Procesar peticiones en paralelo reduce el tiempo de bloqueo de recursos*
- Mantener múltiples copias
  - *Reduce el tiempo de bloqueo de la información*
- Incrementar los recursos disponibles

Arbitrar recursos

- Políticas de asignación
  - ***FIFO:*** *Todas las peticiones son iguales*
  - ***Prioridades asignadas:*** *Ciertas peticiones tienen prelación sobre otras*
  - ***Prioridades dinámicas:*** *Reorganización de acuerdo con los deadlines más cortos*
  - ***Asignación estática:*** *Los criterios de priorización se asignan offline (tiempo de compilación)*

Resumen


Performance Tactics
Control Resource Demand
Manage Resources
Arbitrate Resources
Events Arrive
Increase Computational Efficiency
Reduce Computational Overhead
Manage Event Rate
Control Frequency of Sampling
Bound Execution Times
Bound Queue Sizes
Introduce Concurrency
Maintain Multiple Copies
Increase Available Resources
Scheduling Policy
Response Generated within Time Constraints

Agenda del día

- Introducción
- Preocupaciones
- Desempeño en diferentes puntos de vista
- ¿Cómo lograr desempeño?
- Tácticas arquitecturales
- Patrones arquitecturales

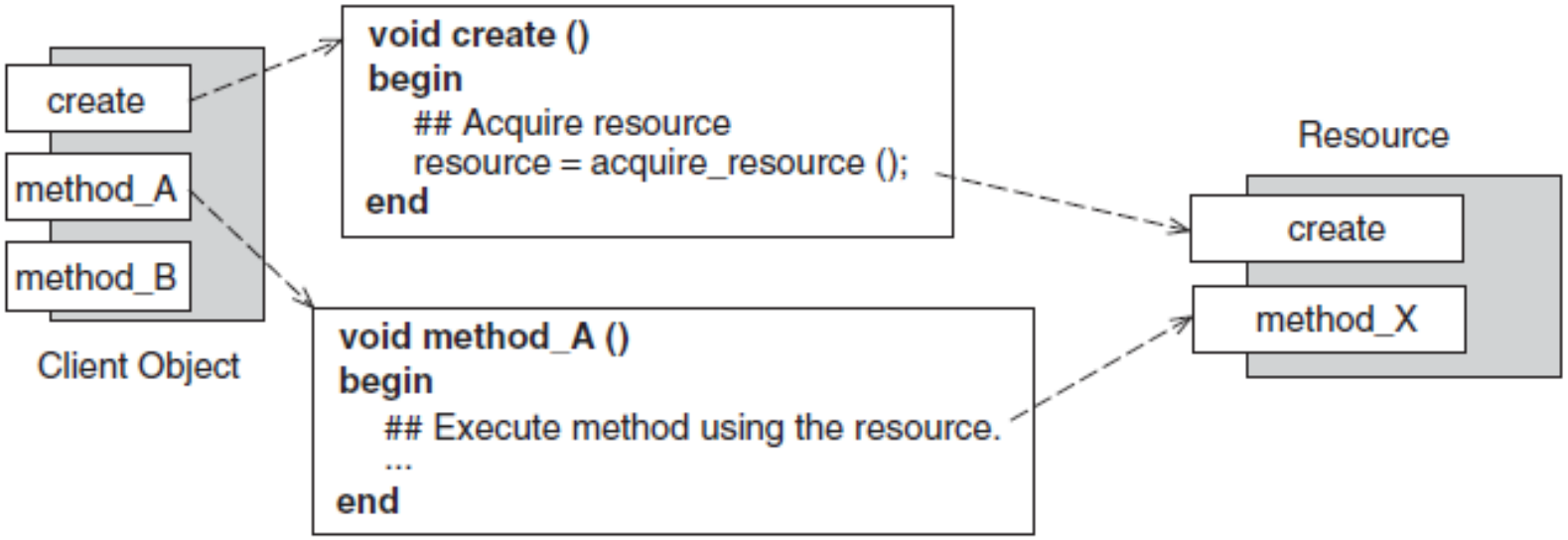
Patrones arquitecturales

- Acceso a recursos
  - Temprana
  - Tardía
  - Parcial
  - Caché
- Manejo de eventos
  - Respuesta reactiva a eventos
  - Respuesta proactiva a eventos
- Concurrencia y Sincronización
  - Coordinador

Adquisición anticipada (Eager Acquisition)

- Problema
  - Disponibilidad de los recursos. La **adquisición** (creación, localización, configuración) de un recurso es muy costosa.
- Solución
  - Adquirir **anticipadamente** los recursos antes de su uso.
  - **Separar responsabilidades:** Existe un proveedor de recursos y un proxy que se comunica con él.
  - Los recursos son guardados en una estructura de datos **eficiente**.

Adquisición anticipada (Eager Acquisition)



Adquisición tardía (Lazy Acquisition)

- Problema
  - Disponibilidad de los recursos. Cuando la adquisición anticipada conlleva a malgastar recursos (i.e. Los recursos se utilizan con poca frecuencia).
- Solución
  - Adquirir los recursos en momento en el que se necesiten

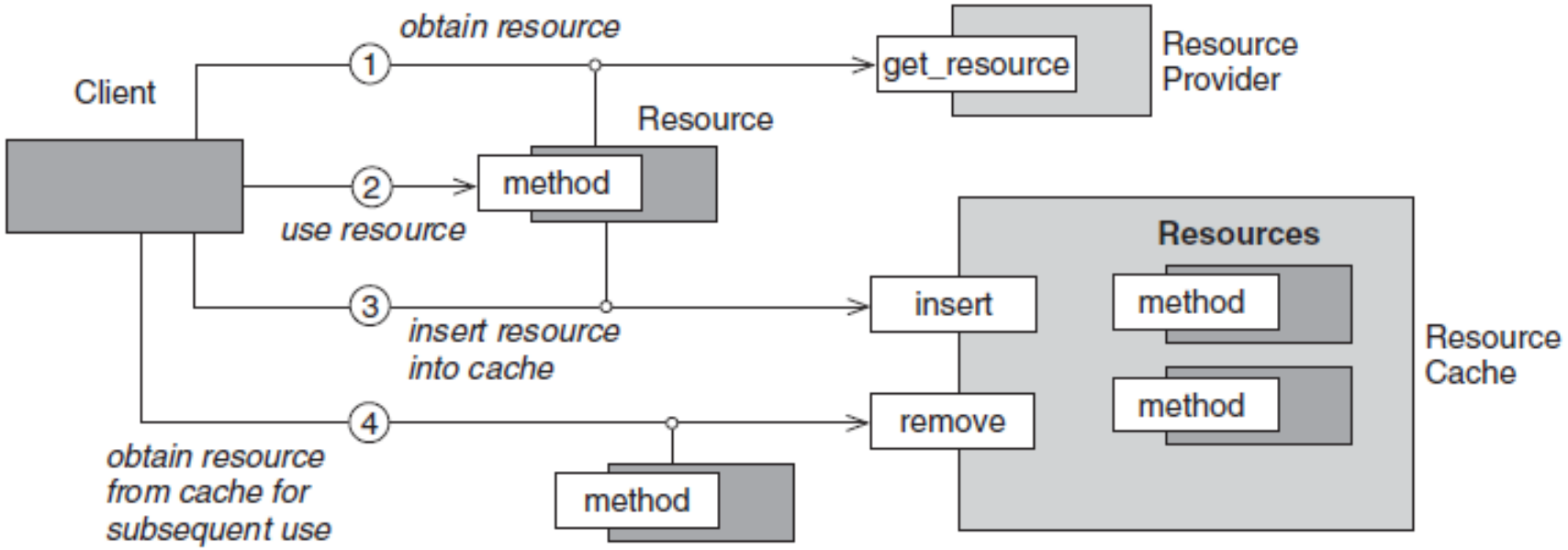
Adquisición por partes (Partial Acquisition)

- Problema
  - Disponibilidad de los recursos. Cuando la adquisición anticipada conlleva a malgastar recursos (i.e. los recursos se utilizan con poca frecuencia) y la adquisición tardía es muy costosa.
- Solución
  - Adquirir los recursos por partes

Manejo de caché (Cache Management)

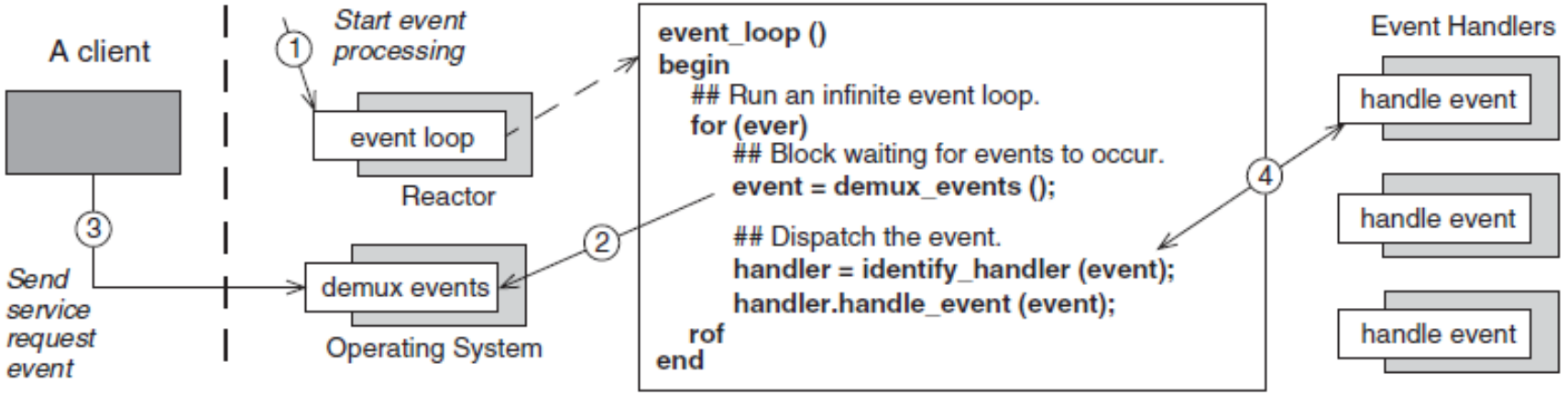
- Problema
  - Disponibilidad de los recursos. La re-adquisición de los recursos puede llegar a ser muy costosa.
- Solución
  - Antes de eliminar un recurso, guardarlo temporalmente en una estructura de datos de rápido acceso
  - Establecer una política efectiva de eliminación de elementos en el caché

Manejo de caché
(Cache Management)



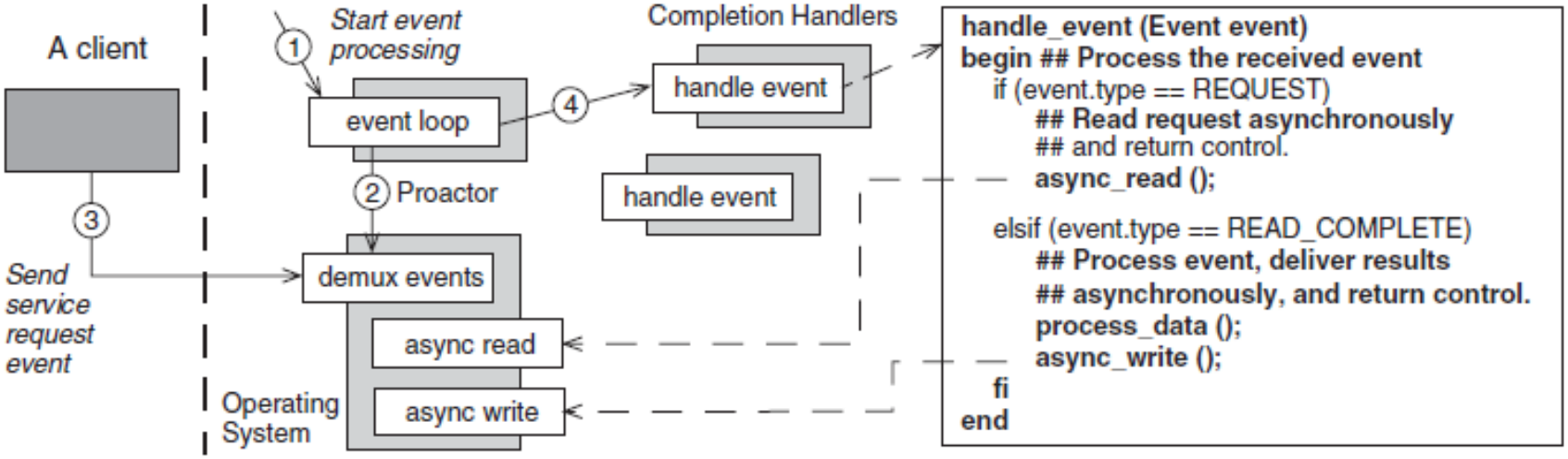
Respuesta reactiva a eventos

- Problema
  - En sistemas distribuidos basados en eventos, ¿Cómo responder a varios servicios simultáneamente?
- Solución
  - Usar una infraestructura que delegue la responsabilidad de manejar el evento a elementos que se ejecuten en diferentes hilos de ejecución.



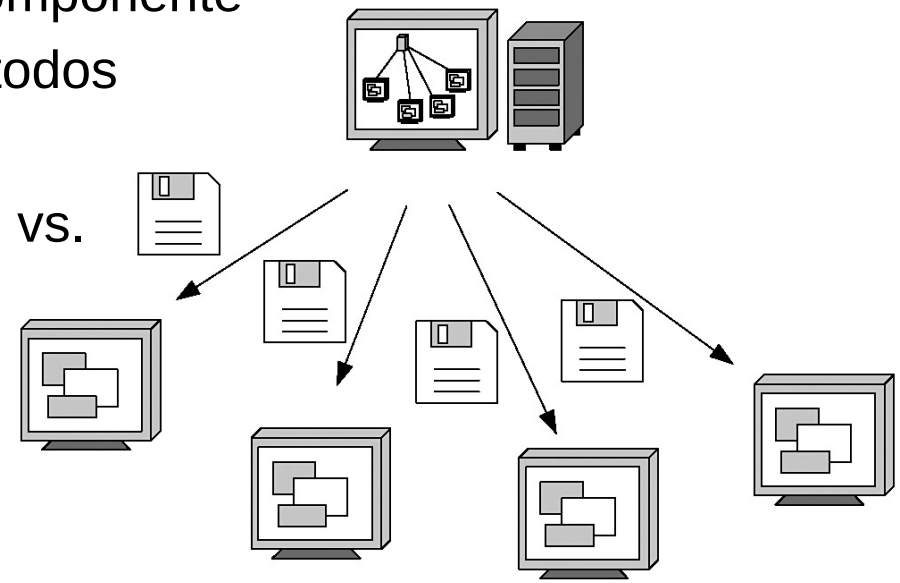
Respuesta proactiva a eventos

- Problema
  - En sistemas distribuidos basados en eventos, ¿Cómo responder a varios servicios simultáneamente?
- Solución
  - Usar una infraestructura que delegue la responsabilidad de manejar el evento a operaciones asíncronas.



Coordinador

- Problema
  - Sincronización entre tareas concurrentes. Cuando se usa concurrencia para dividir el control en subtareas que se ejecutan en paralelo ¿Cómo detectar una falla? ¿Cómo asegurar la consistencia de la operación?
- Solución
  - Centralizar el control en un componente responsable de la ejecución de todos los componentes
  - Tener en cuenta coordinación vs. orquestación.



Universidad de los Andes

Material preparado por…

- Rafael Meneses

¿Preguntas?

Referencias

1. Matthew Bass, **Certificate Program: Software Architecture Design**, iCarnegie, Agosto 4 2009, Bogotá, Colombia
2. Len Bass, Paul Clements, Rick Kazman. **Software Architecture in Practice**, Second Edition.
3. Rozanski Nick, Woods Eoin. **Software Systems Architecture.** Addison Wesley. 2005
4. Len Bass, Paul Clements, Rick Kazman. **Aspectos Avanzados en Arquitectura de Software.** Universidad de los Andes, Curso de Verano 2010, Bogotá, Colombia