Cuestionario Teorías 5 a 7

Inciso 1

Defina y diferencie programa concurrente, programa distribuido y programa paralelo.

Respuesta

Programa distribuido: programa concurrente comunicado por mensajes. Supone la ejecución sobre una arquitectura de memoria distribuida, aunque puedan ejecutarse sobre una memoria compartida (o híbrida).
Programa concurrente: especifica dos o más "programas secuenciales" que pueden ejecutarse concurrentemente en el tiempo como tareas o procesos
Programa paralelo: Se **asocia con la ejecución concurrente en múltiples procesadores** con el objetivo principal de **reducir el tiempo de ejecución**

Característica	Programa Concurrente	Programa Paralelo	Programa Distribuido
Concepto Principal	Capacidad de ejecutar múltiples actividades simultáneamente o en paralelo.	Ejecución concurrente **en múltiples procesadores**.	Aplicación concurrente **en varias máquinas**.
Arquitectura	Concepto de software, no restringido a arquitectura. Puede ser monoprocesador.	Arquitectura de **hardware paralelo** (multiprocesadores).	Arquitectura de **sistemas distribuidos** (varias máquinas).
Objetivo	Estructura de aplicación más natural, **mejorar la respuesta** y uso de hardware.	**Reducir el tiempo de ejecución** y aumentar la *performance*	Compartir el procesamiento entre varias máquinas (e.g., C/S, $P2P$).
Comunicación	Memoria Compartida o Pasaje de Mensajes.	Memoria Compartida o Pasaje de Mensajes.	Típicamente, **Pasaje de Mensajes**.

Inciso 2

Marque al menos 2 similitudes y 2 diferencias entre los pasajes de mensajes sincrónicos y asincrónicos. Indicar cual es la principal ventaja de pasaje de mensajes sincrónicos respecto a pasaje de mensajes asincrónicos.

Respuesta

Similitudes	Diferencias
Ambos modelos utilizan un **canal** (lógico o físico) para la transferencia de datos entre procesos concurrentes.	En PMS la primitiva send es "bloqueante"
Ambos definen primitivas de **envío (`send`) y recepción (`receive`)** para orquestar la comunicación.	En PMS todos los canales son de tipo link (1 receptor, 1 emisor). En PMA pueden ser de varios tipos(mailbox,link, etc).

La principal ventaja es la **garantía de sincronización estricta y la integridad de los datos en el intercambio**.

En el modelo sincrónico, dado que la operación de envío (`send`) es bloqueante hasta que el receptor está listo, se asegura que el emisor solo continúa su ejecución **una vez que el mensaje ha sido entregado** o la recepción ha sido iniciada. Esto simplifica la lógica del programa, ya que se **elimina la incertidumbre** sobre el estado del mensaje y se evita el riesgo de que el proceso emisor altere el dato antes de que el receptor haya tenido la oportunidad de copiarlo (un riesgo de la comunicación asincrónica insegura).

Inciso 3

Analice qué tipo de mecanismos de pasaje de mensajes son más adecuados para resolver problemas de tipo Cliente/Servidor, Pares que interactúan, Filtros, y Productores y Consumidores. Justifique claramente su respuesta.

Respuesta

PMA	PMS
Productor/consumidor. La principal ventaja de usar PMA es que el Productor no tiene que esperar a que el Consumidor esté listo para recibir cada ítem. Esto permite al Productor **mantener un flujo continuo de producción** (no se bloquea), mejorando la **eficiencia** y usando una **cola de mensajes (búfer)** como punto de encuentro.	Pares que interactúan. Este patrón requiere un alto grado de **coordinación** y **sincronización estricta**. Esto garantiza que la información se transfiera de forma **segura e inmediata** y que ambos procesos estén en un **estado conocido** antes de continuar,
Cliente/Servidor. El **Cliente** envía una solicitud y debe poder **continuar con otras tareas** o enviar otras solicitudes sin quedarse bloqueado esperando la respuesta inmediata del Servidor. El PMA ofrece la **máxima disponibilidad** para el cliente (la primitiva `send` no bloquea) y	Filtros. Un filtro toma una entrada, la procesa y produce una salida. Al conectarse en **tuberías (*pipes*)**, la velocidad de procesamiento de los filtros debe estar **sincronizada**. Si un filtro produce datos más rápido de lo que el siguiente puede consumirlos, el mecanismo sincrónico **bloquea**

PMA	PMS
permite al Servidor manejar múltiples solicitudes de manera más flexible, usando colas. Y a su vez el servidor puede no esperar a que el cliente reciba el mensaje y seguir atendiendo solicitudes	**al productor** (el filtro anterior) hasta que el consumidor (el filtro siguiente) haya recibido el mensaje. Esto **evita el desbordamiento** y garantiza un flujo de datos controlado.

Inciso 4

Indique por qué puede considerarse que existe una dualidad entre los mecanismos de monitores y pasaje de mensajes. Ejemplifique

Respuesta

La dualidad se considera que existe porque, a pesar de usar mecanismos de comunicación opuestos, los **Monitores** y el **Pasaje de Mensajes (PM)** tienen el **mismo poder expresivo** (son equivalentes).

Esto significa que cualquier problema de sincronización y exclusión mutua que se pueda resolver con uno, se puede resolver con el otro.

La dualidad se expresa modelando la abstracción de alto nivel de un **Monitor** (orientado a memoria compartida) como un proceso **Servidor** que usa Pasaje de Mensajes para gestionar el acceso a un recurso.

Programas con monitores	Programas basados en PM
Variables permanentes	Variables locales del servidor
Identificadores de procedures	Canal request y tipos de operación
Llamado a procedure	send request(): recieve respuesta
Entry del monitor	recieve request()
Retorno del procedure	send respuesta()
Sentencia wait	Salvar pedido pendiente
Sentencia signal	Recuperar/ procesar pedido pendiente
Cuerpos del procedure	Sentencias del "case" de acuerdo a la clase de operación

Inciso 5

¿En qué consiste la comunicación guardada (introducida por CSP) y cuál es su utilidad? Describa cómo es la ejecución de sentencias de alternativa e iteración que contienen comunicaciones guardadas.

Respuesta

Consiste en poder "disminuir" o evitar las limitaciones de ? y ! ya que son bloqueantes. Hay problema si un proceso quiere comunicarse con otros (quizás por != ports) sin conocer el orden en que los otros quieren hacerlo con él.

Existen 2 alternativas, el if y el do guardado, ambos tienen esta ejecución:

- Primero, se evalúan las guardas.
  - Si todas las guardas fallan, el if termina sin efecto.
  - Si al menos una guarda tiene éxito, se elige una de ellas (no determinísticamente).
  - Si algunas guardas se bloquean, se espera hasta que alguna de ellas tenga éxito.
- Segundo, luego de elegir una guarda exitosa, se ejecuta la sentencia de comunicación de la guarda elegida.
- Tercero, se ejecuta la sentencia S_i.
  En el caso del if se ejecuta 1 sola vez, en el do se ejecuta hasta que todas las guardas fallen.

Inciso 8

Marque similitudes y diferencias entre los mecanismos RPC y Rendezvous. Ejemplifique para la resolución de un problema a su elección.

Respuesta

Similitudes:

- Ambos utilizan técnicas de comunicación y sincronizacion a través de un canal bidireccional
- Combinan una interfaz tipo monitor con operacciones a través de llamadas y mensajes sincronicos.
- El **emisor (cliente)** queda bloqueado esperando la respuesta del **receptor (servidor)**.
  Diferencias:

- Rpc cada operación es declarar un proceso y crear uno nuevo para manejar ese llamado
- En rendezvous ya se hace con un proceso existente. A través de una sentencia de entrada que espera una invocación

Inciso 9

Describa sintéticamente las características de sincronización y comunicación de ADA. Indicar que diferencia hay entre la comunicación guardada de Rendezvous (general) con la provista por ADA.

Respuesta

Sincronización

- **Sincronización:** Es **sincrónica** y **bloqueante** por diseño (similar al Pasaje de Mensajes Sincrónico).
  - La tarea que hace la llamada (`nombreTarea.nombreEntry(...)`) se **bloquea** hasta que la tarea receptora (`Task`) está lista para la ejecución y ha completado el cuerpo del `accept`.
  - La tarea receptora que ejecuta el `accept nombreEntry(...)` se **bloquea** hasta que una tarea cliente realiza la llamada al *entry*.
- **Exclusión Mutua:** Se logra de forma implícita, ya que **solo una tarea** puede estar ejecutando el cuerpo del `accept` asociado a un *entry* en un momento dado.

Comunicación

- **Mecanismo:** Se realiza mediante **puntos de entrada (`entry`)** declarados en la interfaz de una tarea.
- **Parámetros:** Los parámetros definidos en el `entry` (`in`, `out`, `in out`) permiten la transferencia de datos entre las tareas durante la fase de encuentro.
- **Rendezvous:** El encuentro es el período de tiempo en el que la tarea llamadora y la tarea receptora están acopladas y se ejecuta el cuerpo del `accept`.

Diferencia comunicación guardada

Característica	Rendezvous con Guardas Generales (Conceptual)	Rendezvous Provisto por Ada
Control de Aceptación	La comunicación guardada general solo establece una **condición lógica** (`guarda`)	La aceptación de un `entry` se realiza a través de la sentencia `select`, la cual permite que una tarea **espere**

Característica	Rendezvous con Guardas Generales (Conceptual)	Rendezvous Provisto por Ada
	para que la operación (ej., `send` o `receive`) se ejecute.	**concurrentemente** por múltiples `entry`s y es la única que ejecuta el cuerpo del `accept`.
Exclusividad del `accept`	El Rendezvous general se centra solo en la sincronización de las operaciones `send` y `receive` mediante guardas.	La sentencia `accept` en Ada **debe estar dentro de la tarea que declara el `entry`**. Una tarea no puede hacer un `accept` del *entry* de otra tarea.