Sistemas Operativos II

Trata de tópicos avanzados en diseño e implementación de Sistgemas Operativos, tales como: estructura de sistemas operativos, multi-threading, sincronización, sistemas paralelos y sistemas distribuidos. Durante todo el curso de desarrollo, prácticas de laboratorio siguiendo los temas estudiados semanalmente.

***¡Con un laboratorio se entienden mejor!***

|  |  |  |
| --- | --- | --- |
| Summary chapter 9 | | |
| Existen tres tipos de programación: | | |
| **Programacion a largo-plazo = Long-term scheduling:**  Determina cuando un proceso es admitido al sistema. | **Programacion a medio-plazo = Medium-term scheduling:**  Es parte de funciones de cambio y determina cuando un programa es parcialmente llamado o completamente en la memoria principal. | **Programacion a corto-plazo = Short-term scheduling:**  Determina cual proceso esta listo para ser ejecutado en el procesador. |
| **Algoritmos:** Una variedad de algoritmos han sido desarrollado para realizar decisiones en programación a corto-plazo. | | |
| **FIRST-COME-FIRST-SERVE:**  Selecciona el proceso de servicio que tiene mayor espera. | **ROUND-ROBIN:**  Uso de partición para limitar cualquier proceso ejecución a una corta ráfaga de tiempo de procesamiento y rotación sobre todos los procesos listos. | **SHORTEST PROCESS NEXT:**  Selecciona el proceso con el menor tiempo de procesamiento y no se adelanta el proceso. |
| **SHORTEST REMINDING TIME:**  El proceso seleccionado con la menor expectativa de tiempo de procesamiento en espera. | **HIGHEST RESPONSE RADIO NEXT:**  Basado en la dirección agendada en un tiempo estimado de ciclos normalizado. | **FEEDBACK:**  Establecimiento de programación de colas y procesos ubicados para colas basadas en historial de ejecución y criterios. |
| **\*\* La elección de programación de algoritmos depende de una configuración esperada y una implementación compleja \*\*** | | |

***TERMINOS CLAVE = KEY TERMS***

|  |  |  |
| --- | --- | --- |
| **KEY TERMS** | | |
| Arrival Rate – Tiempo de llegada  Dispatcher - Despachador  Exponential averaging – Promedio exponencial  Fair-share scheduling – Programacion equilibrada  Fairness - Justicia  First-Come-First-Served – Primero en llegar primero en salir  First-in-first-out – Primero dentro primero en salir  Long-Term Scheduler – Programacion de plazo extendido | Medium-Term Scheduler – Programacion de medio-plazo  Multilevel Feedback – Retroalimentacion de multinivel  Predictability - Predictividad  Residence Time – Tiempo de Residencia  Response Time – Tiempo de Respuesta  Round Robin – Sistema Todos contra todos  Scheduling Priority – Prioridad de programacion  Service Time – Tiempo de servicio | Short-Term Scheduler – Programacion de corto-plazo  Througput - Rendimiento  Time Slicing – Tiempo de partición  Turnaround Time (TAT)  Utilization - Utilitidad  Waiting Time – Tiempo de espera |
|  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Chapter 10 | | | |
| **Multiprocessor and multicore scheduling = Planificacion en multiprocesos y multinucleo** | | | |
| Clasificación de sistemas de multiprocesamiento: | | | |
| * **Levemente acoplado o multiprocesador distribuido o cluster:**   Consiste en un conjunto de sistemas autónomos relativos, cada procesador tiene su propia memoria principal y canales I/O. | * **Procesadores especializados funcionales:**   Hay un maestro de procesador de propósito genera; proceso especializado son controlados por el procesador maestro y le provee servicios. | | * **Procesador muy bien acoplado:**   Conjunto de procesadores que son compartidos en la misma memoria principal y están bajo el control integrado de los sistemas operativos. |
| **GRANULARIDAD:**  La mejor manera de caracterización de multiprocesadores y ubicarlos en contexto a otras arquitecturas es considerar la sincronización de granularidad o frecuencia de sincronización entre procesos y el sistema.  **Se distinguen 5 categorías.** | | | |
| **INDEPENDIENT PARALELLISM = PARALELISMO INDEPENDIENTE**   * **Procesos múltiples no relacionados**   Características:   1. No tiene una sincronización explicita. 2. Cada una representa una separación, aplicación o trabajo independiente. 3. Mas de un procesador es disponible, tiempo promedio de respuesta hacia el usuario debe ser corto.   Ejemplo de uso: Time-sharing system = Systema de tiempo compartido. | **COURSE AND VERY COURSE GRAINED PARALLELISM = PARALELISMO GRANULADO GRUESO Y MUY GRUESO**   * **Grueso: Procesos concurrentes de multiprocesamiento en ambiente de multiprogramación** * **Muy Grueso: Procesos distribuidos en nodos de red en un único medio computacional.**   Características:   1. Sincronización de procesos a un nivel grueso. 2. Conjunto de ejecución de procesos concurrentes en un único procesador programado. 3. Cualquier agrupación de procesos concurrentes necesarios para comunicar o sincronizar se puede beneficiar por el uso de arquitectura de multiprocesador. | | **MEDIUM-GRAINED PARALLELISM = PARALELISMO GRANULADO MEDIO**   * **Procesamiento paralelo o multitarea, dentro de una aplicación.**   Características:   1. Se debe tener incluso a grado elevado de coordinación e interaccion en los subprocesos de una aplicación, guiando a un nivel de sincronización de granularidad mediana. 2. El programador debe ser explícitamente especifico en el potencial del paralelismo en la aplicación. |
| **FINE-GRAINED PARALLELISM, = PARALELISM GRANULADO FINO**  **Inherente paralelismo en una única trama de instrucción.**  Características:   1. Representa el mejor uso complejo de paralelismo que se encuentra en el uso de sub-procesos. | | | |
| **Desing Issues = Problemas de diseño** | | | |
| Programación en un multiprocesador envuelve tres problemas interrelacionados   1. **Asignación de procesador4es a procesos.** 2. **Uso de multiprogramación en procesos individuales.** 3. **Envió actual de procesos.**   Se debe considerar que la toma de decisiones puede depender, en general, en el grado de granularidad de aplicaciones y un numero de procesos disponibles. | | | |
| **Asignación de procesos a procesadores**  Se asume una arquitectura uniforme.  El programa trata al procesador como recursos agrupados y demanda de asignación de procesos a procesadores.   * Estático: Si es asignado a un único procesador para activación, aunque sea completado, por lo que se dedica una cola short-term es mantenido para procesadores. * Dinámico: Balanceo de carga, los sub-procesos son ubicadas en la cola por un procesador hacia otro procesador. | **Uso de multiprogramación en procesadores individuales**  En el multiprocesador tradicional, que es tratado con granularidad gruesa o granularidad sincrónica independiente. Está claro que cada procesador individual debería estar disponible para cambiar entre un conjunto de procesos enumerados para archivar alta utilización y mejorar configuración. | | **Envió de procesos**  Se aprecia en uni-procesadores de multiprogramación de uso de prioriedades o de algoritmos de programación sofisticada basados en el pasado uso puede mejorar el desempeño en una estrategia fácil First-Come-First-Serve. |
| **Proces Scheduling = Programacion de procesos** | | | |
| **En los sistemas tradicionales de multiprocesadores, no están dedicados al procesador.**  Existe una cola únicamente para todos lo sprocesadores o si algún tipo de esquema,as de prioridad es usado, existen multiples colas basadas en prioridad, llenándose en una pila o almacenamiento en común de procesadores. Se puede ver como un sistema asi como una arquitectura en colas de multiservicio.  **Caso>** Sistema de procesamiento dual en el cual cada proceso del sistema tiene una mitad de eventos de procesamiento de un procesador, en el único sistema de procesamiento.  En conclusión general, tiene la especifica disciplina de programación menos importancia con dos procesadores que con uno. Deberia evidenciar que se vuelve mucho más fuerte la conclusión con un incremento en números de procesadores. Por lo que una uninca disciplina FCFS con un esquema estatico de prioriedad, puede bastar para un sistema de multiprocesamiento. | | | |
| **Thread Scheduling = Programacion de subprocesos** | | | |
| **El concepto de ejecución es separado por el resto de la definición de un proceso.** Una aplicación puede ser implementada como un conjunto de subprocesos que componente ejecutan frecuentemente la misma dirección de espacio.  En un único procesador, los subprocesso pueden ser usados como un apoyo estructura miento de programa y para procesamiento de entradas de ciclos. Por lo que la misma falta haciendo comparaciones de cambios en subproceso a un cambio de proceso, por lo que beneficio son realizados a un pequeño costo.  Si varios suprocesos de una aplicación se ejecutan de manera simultánea en un proceso separado, dramáticamente gana en configuración. Como sea, puede ser mostrado por aplicaciones que requieren una interacción significativa en el conjunto de subprocesos (paralelismo de granularidad media), pequeñas differencias en la administración de subprocesos y programación pueden tener un impacto significante de configuración.  Respecto a la cantidad de propuestas de programacion de subprocesos en multiprocesadores y asignación de procesos, se tiene **cuatro** enfoques: | | | |
| **Load Sharing = Carga compartida:**  Son asignados a procesadores particulares  Una cola global de subprocesos listos para ejecutar se mantiene, en cada procesador, cuando está disponible, se selecciona un proceso de la cola.  El termino **carga compartida** es usado para disticion de la estructura en la estrategia de balanceo de carga en el cual se ubica en mas de una base permanente.  Ventajas:   1. Es distribuida uniformemente en los procesadores, asegurando que no hallan procesadores disponibles, mientras haya trabajo que hacer. 2. Es requerida la programación no centralizada: cuando un procesador está disponible, la agenda de ejecución del sistema operativo, se ejecuta en el procesador seleccionado en el próximo subproceso. 3. La cola global puede ser organizada y accedida usando esquemas incluyendo base de prioridad es12uemas de ejecución de historiales o demandas de procesos anticipadas.   Tres diferentes versiones de balanceo de carga:   1. **FIRSDT-COME-FIRST-SERVE(FCFS)**: 2. **Smallest Number of threads first**: 3. **Preemptive smallest numbre of threads first**: | | **Gang Scheduling = Programacion de conjuntos:**  Como un conjunto de subprocesos se ejecutan para correr en un conjunto de procesos al mismo tiempo, en base de uno a uno.  Ventajas:   1. Si los procesadores en el grupo son relacionados o coordinados en la misma línea, el bloqueo de sincronización puede ser reducido menor procesos de cambios son necesarios y la configuración puede incrementar. 2. Una única desciucon de programación afecta a numerois de procesadores y procesos en un tiempo reduciendo la programación de recursos. | |
| **Dedicated processors assigned = Asignacion de procesos dedicados**  Es opuesta al planeamiento balanceo de carga y porovee una programación impolicita a la asignación de subprocesos a procesadores. | | **Dynamic Scheduling = Programacion dinámica**  El número de subprocesos pueden ser alterado durante el curso de ejecucion. | |
| Summary chapter 10 | | | |
| **Tigly coupled mulktiprocessor = procesador levemente acoplado:**  Multiples procesadores tienen acceso a la misma memoria principal | **A real-time process or task = Procesamiento de tiempo real o conjunto de tareas:**  Se ejecuta en conexión a algunos procesops o funcino o eventos externos para el sistema de computadora y con conocimiento de varios plazos para una correcta, efectiva interacción con el medio externo. | | **Real-Time Operating Systems = Sistemas Operativos de tiempo-real:**  Es capaz de manejar procesos en tiempo real |
| **En conclusion: El factor clave es conocer los plazos.**  **Se confía Algoritmos de mucho peso en la planificación y reacción de plazos relativos que son apropiados en el contexto.** | | | |
| **TERNINOS CLAVE = KEY TERMS** | | | |
| Key Terms | | | |
| Aperiodic task  Deadline scheduling  Deterministic  Deterministic operating system  Fail-soft operation  Gang scheduling  Granularity  Hard affinity  Hard real-time task | Load sharing  Periodic task  Pirority ceiling  Priority inheritance  Priority inversión  Processor affinity  Pull mechanism  Push mecahnism  Rate monotomic scheduling | | Real-time operating system  Real-time scheduling  Responsiveness  Soft affinity  Soft real-time task  Thread scheduling  Unbounded priority inversion |

|  |
| --- |
| chapter 14: Máquinas Virtuales |
| Virtula Machine Concepts = Conceptos de maquinas virtuales |
|  |
| Hypervisor = Hipervisor |
| Container Virtualization = Virtualizacion de contenedores |
| Processor Issues = Problemas de procesamiento |
| Memory management = Administracion de memoria |
| I/O Management = Administracion I/O |
| VM ESXi |
| Microsoft Hyper-V and Xen Variants |
| Java VM |
| Linux Vserver Virtual Machine Architecture |
|  |

|  |  |  |
| --- | --- | --- |
| Summary chapter 14 | | |
| **La tecnologia de virtualización da la posibilidad a un único computador o servidor para que de manera mimultanea ejecute multiples sistemas operativos o multiples sesiones en un único SO. En escencia, el sistema operativo cliente puede soportar un numero de maquinas virtuales (VM), cada una tiene las características de un SO en particular y multiples versiones de virtualización, características de una plataforma hardware particular.**  **LA estructura normal de una maquina virtual hace uso de monitores de maquinas virtuales o hypervisor, dado es a un nivel bajo que las MV y soportes MVs. Existen dos tipos de hypervisores, distinguido por el medio, hay otro SO entre el hipervisor y el cliente.**   * **Hipervisor Tipo 1: directamente ejecutado en el hardware de la maquina.** * **Hipervisor Tipo 2: opera en la cima del SO cliente.**   **Java VM es una forma muy diferente de implementar un sistema MV. La meta de una MV java es proveer un espacio de ejecutando un conjunto de comandos java esa ejecutar etapa de SO en cualquier plataforma hardware. Sin la nevesidad de generar cambios de codigo para definir la diferencia de sistemas operáticos o hardware diferente.** | | |
|  | | |
| **KEY TERMS** | | |
| **Container**  **Container virtualization**  **Consolidtion ratio**  **Docker**  **Guest OS**  **4Hardware virtualization**  **Hardware assisted virtualization**  **Hist OS** | **Hypervisor**  **Java virtual machune**  **(JVM)**  **Kernel control group**  **Memi9ry balloning**  **Memory overcommit**  **Microservice**  **Page sharing**  **paravitualization** | **Type 1 hyoercusor**  **Type 2 hypercisor**  **Vurtual appliance**  **Virtualization container**  **Virtualization**  **Virtual machine (VM)**  **Virtual machine monitor (VMM)** |

|  |
| --- |
| **CONCEPTOS** |
| **UNIPROCESSOR SCHEDULING = PROGRAMACION DE UNIPROCESOS** |
| **TYPES OF PROCESSORS SCHEDULING = TIPOS DE PROGRAMACION DE PROCESADORES** |
| **El objetivo principal de un programador de procesamiento es la asignación de procesos para ejecutar por el procesador o tiempo de procesamiento.** |

|  |  |  |  |
| --- | --- | --- | --- |
| **HOJA DE COMANDOS = HOJA DE PROGRAMAS = shell** | | | |
| **COMANDO = PROGRAMA** | **DESCRIPCION** | **COMANDO = PROGRAMA** | **DESCRIPCION** |
| **cd** | **Control Directorio** | **ls** | **Listado, archivos, carpeta** |
| **cal <ano>** | **Calendario** | **alias <nombre> = PROGRAMA <VALOR>** | **Crea alias para ejecución** |
| **Bourne shell** |  | **C shell** |  |
| **Bourne Again Shell** |  | **tcsh** |  |
| **Korn shell** |  | **zsh** |  |
| **Vi o vim** | **Editor de archivos** | **emacs** | **Editor de archivos** |
| **pico** |  | **nano** |  |
| **dpkg** |  | **apt-get** |  |
| **apt-cache** |  | **rpm** |  |
| **yum** |  | **echo <>** | **Mostrar salida en el terminal** |
| **env <>** | **Variable de entorno** | **export <>** | **Puede utilizarse para crear var entorno** |
| **unset** | **Elimina variables exportadas** | **Which<>** | **ubicación programas de variable PATH** |
| **whoami** | **Visualiza el nombre de usuario** | **type<>** | **Identifican alias de otros comandos** |
| **uname** | **Visualiza el nombre del kernel** |  |  |
| **man<>** |  |  |  |

|  |  |  |
| --- | --- | --- |
| **Capitulo 9 = Chapter 9** | | |
| **Términos clave = Key terms** | | |
| **Arrival rate**  **Dispatcher**  **Exponential averaging**  **Fair-share scheduling**  **Fairness**  **First-come-first-served**  **First-in-first-out**  **Long-term scheduler** | **Medium-term scheduler**  **Multilevel feedback**  **Predictability**  **Residence time**  **Response time**  **Round robin**  **Scheduling priority**  **Service time** | **Short-term scheduler**  **Throughput**  **Time slicing**  **Turnaround time (TAT)**  **Utilization**  **Waiting time** |