UNIVERSIDAD FRANCISCO DE VITORIA
Computación de Alto Rendimiento

COMPUTACIÓN DE ALTO RENDIMIENTO

Tema4 . Sistemas de Tiempo Real- Introducción

Grado en Ingeniería Informática ***Curso 2022/2023***

Objetivos

- Sistemas de tiempo real
- Características de los Sistemas Operativos adecuados a esta programación
- Programación a gran escala

Introducción

- El alto rendimiento indica la cantidad de procesamiento que se realiza en un período de tiempo determinado, mientras que el tiempo real es **la capacidad de terminar con el procesamiento para producir una salida útil en el tiempo disponible**.

- *Ejemplo: Procesamiento de señales "real-time" sensores de un coche autónomo para generar una respuesta en la conducción*
- *Ejemplo: Buscador Google. Tiempo de respuesta (1 página) en tiempo umbral establecido, empleando técnicas de paralelismo (HPC)*

Clasificación de los Sistemas de Tiempo Real

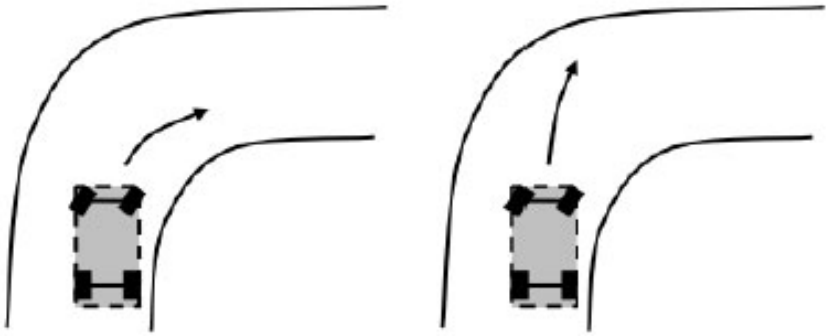
- Sistema en tiempo real son aquellos que deben producir respuestas correctas dentro de un intervalo de tiempo definido. **Si el tiempo de respuesta excede ese límite**, se produce una degradación del funcionamiento y/o un **funcionamiento erróneo**.
- Clasificación (según requisitos temporales)
  - Tiempo **real estricto** (hard real time): Cuando es **absolutamente necesario** que la respuesta se produzca dentro del límite especificado. Ej.: control de vuelo.
  - Tiempo **real no estricto** (soft real time): Cuando se permite la pérdida ocasional de especificaciones temporales, aunque debe cumplirse normalmente. Ej.: sistema de adquisición de datos
  - Tiempo **real firme** (firm real time): Cuando se permite la pérdida ocasional de especificaciones temporales, pero dicha perdida no implica beneficios ya que la respuesta retrasada es descartada. Ej.: sistema multimedia.

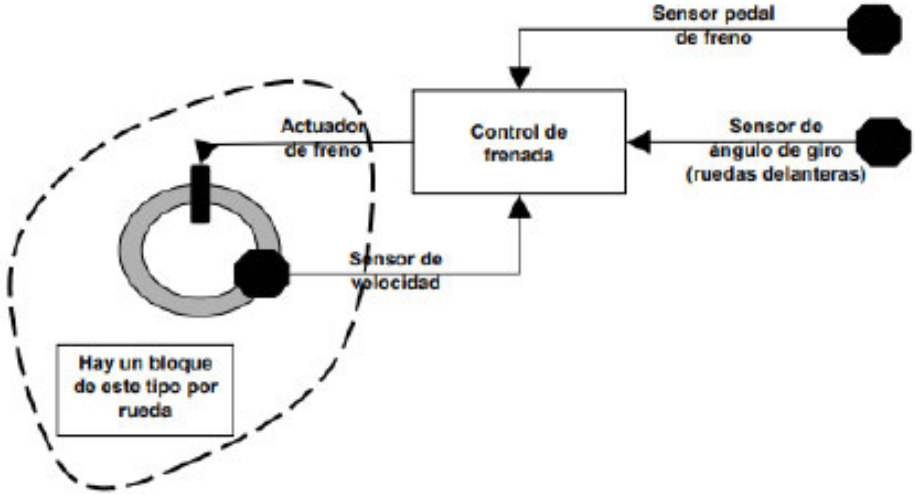
Ejemplos de Sistemas de Tiempo Real

- Sistemas empotrados se encuentran en multitud de productos y están integrados en la estructura física de la maquinaria que controlan:
  - Vehículos: Coches, autobuses, trenes, aviones, barcos, grúas,…
  - Maquinaria industrial: Robots, células de fabricación, maquinas/herramienta,…
  - Sistemas de comunicación: Teléfonos móviles, repetidores, rotures, radio,…
  - Electrodomésticos: Elec. línea blanca, televisores, videos, cámaras fotos, alarmas, …
  - Ocio: Consolas videojuego, juguetes, salas de cine, sistemas de sonido, iluminación,… – Armamento: Guía de misiles, balística,…

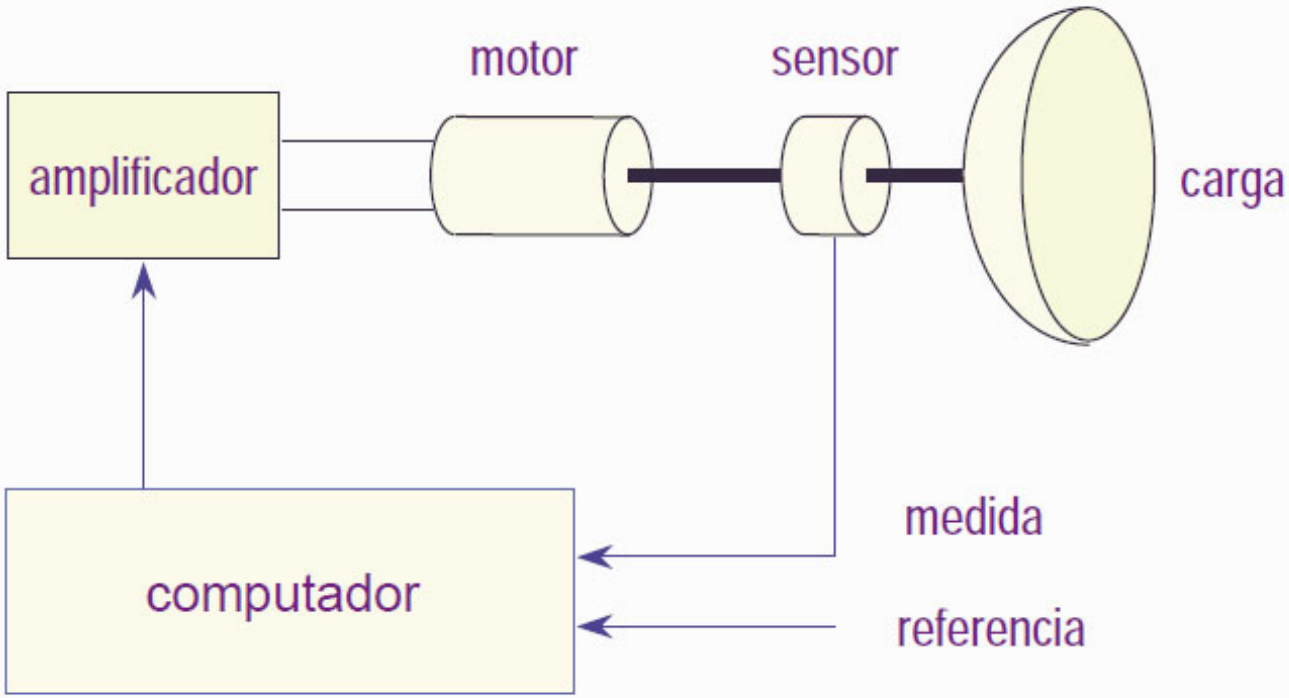
Ejemplos de Sistemas de Tiempo Real

- **ABS**: El sistema en función de la velocidad, el ángulo de giro y el deseo de conductor de frenar, tomará la decisión correcta y todo ello en un tiempo determinado.

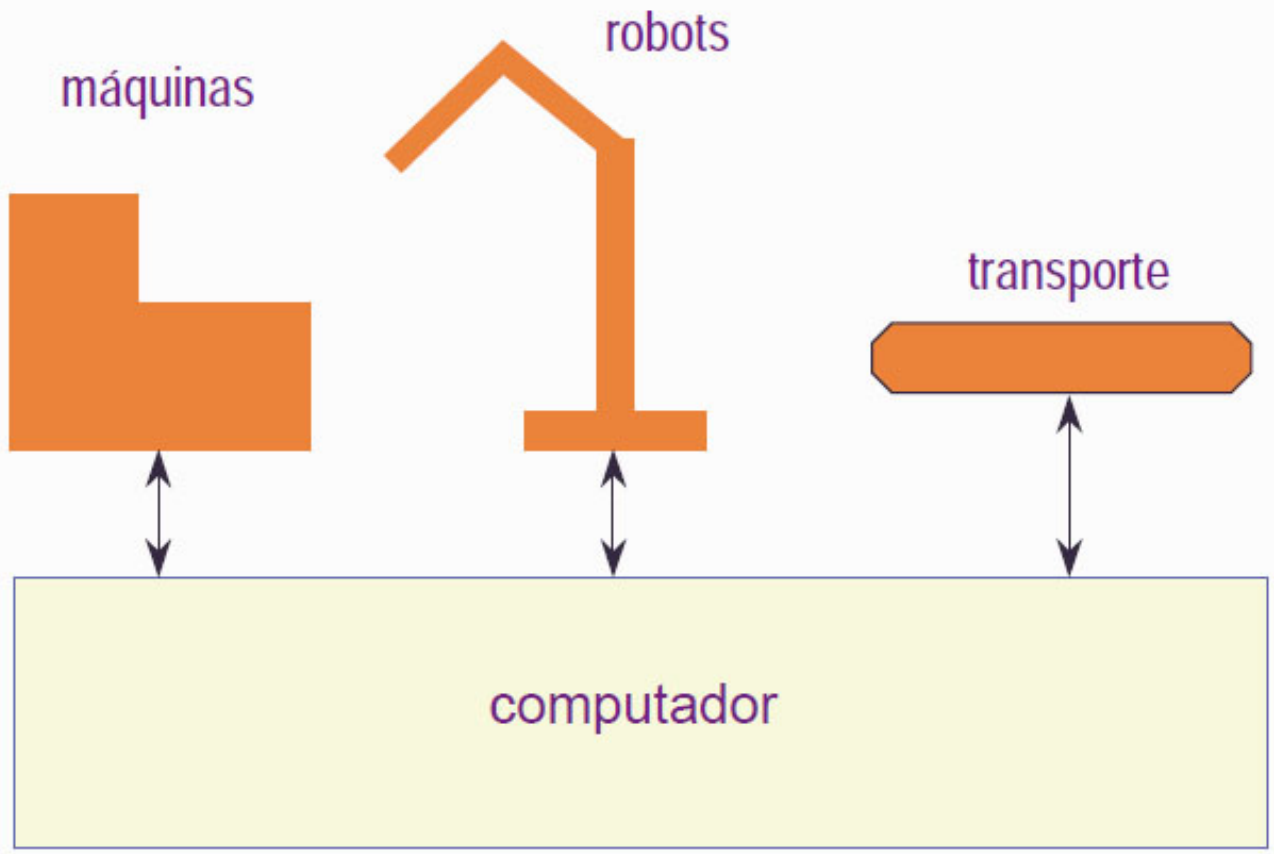




Ejemplo: control de posición



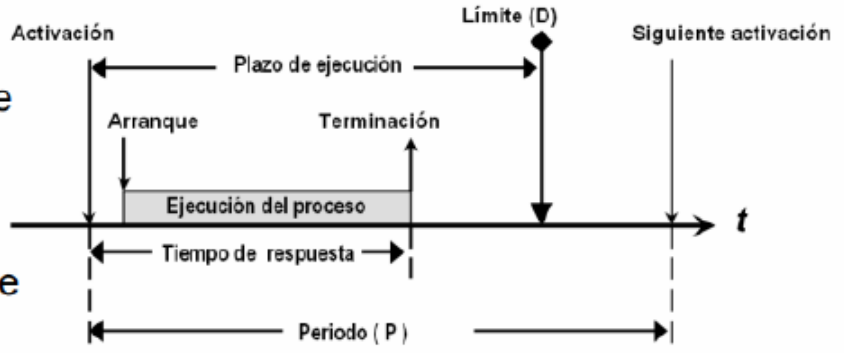
Ejemplo: control de fabricación



Clasificación de Tareas en Sistemas de Tiempo Real

- Las tareas se clasifican, atendiendo a su **semántica** en:
  - **Críticas**: El fallo de una de estas tareas puede ser catastrófico.
  - **Opcionales** (no críticas): Se pueden utilizar para refinar el resultado dado por una tarea crítica, o para monitorizar el estado del sistema, etc.
- En función de la **forma de ejecución**:
  - **tareas periódicas**. Se activan repetidamente a intervalos de tiempo fijo
  - **Tareas esporádicas**. Se activan en instantes aleatorios y tienen requisitos temporales críticos
  - **Tareas aperiódicas**. No poseen requisitos temporales rígidos

•*Período de activación (p)* o tiempo que transcurre entre dos activaciones consecutivas.
•*Plazo de ejecución (d)* o tiempo de respuesta máximo (máximo plazo de tiempo entre la activación y la terminación de forma correcta).
•*Tiempo de cómputo (c)* (tiempo de cómputo máximo en cada activación).
•Se debe cumplir que $0 \leq c \leq d \leq p$

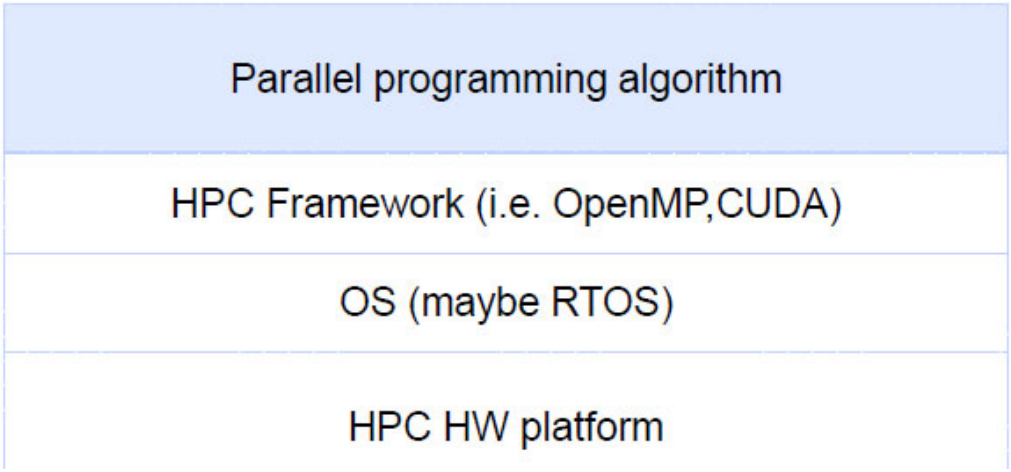


Sistemas Operativos de Tiempo Real

- **Definición**: aquel que ha sido desarrollado para aplicaciones de tiempo real. Como tal, se le exige corrección en sus respuestas bajo ciertas restricciones de tiempo. Si no las respeta, se dirá que el sistema ha fallado. Para garantizar el comportamiento correcto en el tiempo requerido se necesita que el sistema sea predecible. (QNX, LynxOS, VxWorks, Windows CE, ...)
- Los **objetivos que persigue la planificación** de tareas son:
  - Garantizar la correcta ejecución de todos los procesos críticas.
  - Ofrecer un buen tiempo ejecución de todos los procesos no periódicos.
  - Administrar el uso de recursos compartidos.
  - Posibilidad de recuperación ante fallos software y hardware.
  - Soportar cambios de modo, esto es, cambiar en tiempo de ejecución el conjunto de tareas. Por ejemplo: un cohete tiene que realizar acciones muy distintas durante el lanzamiento, estancia en orbita y regreso; en cada fase, el conjunto de tareas que se tengan que ejecutar ha de ser distinto

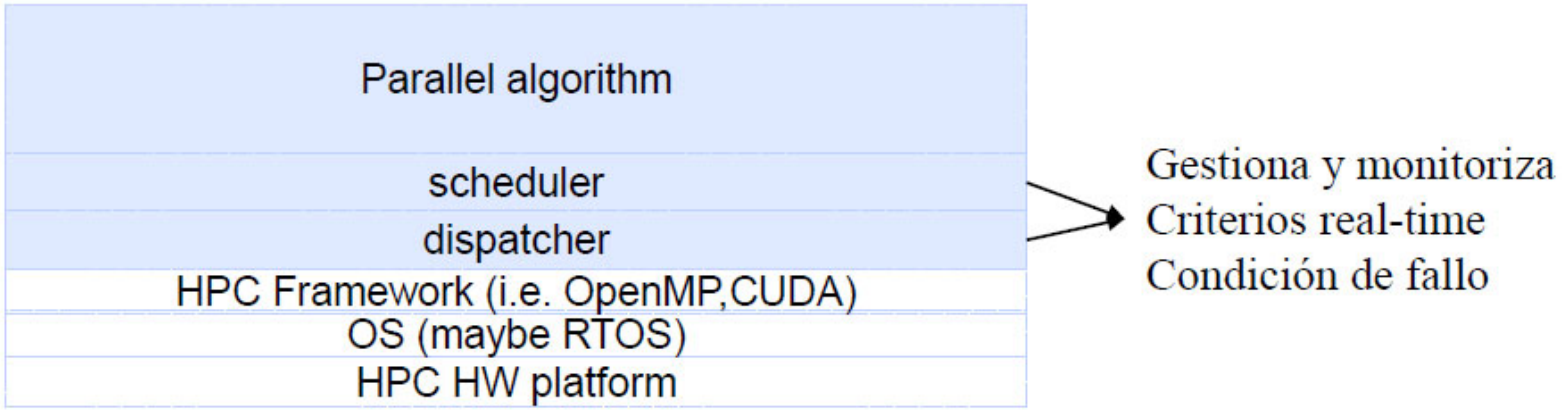
Consideraciones extras para el Diseño

- Nuestro diseño paralelo de la solución debería agregar consideraciones adicionales para que se logren garantizar los criterios de real-time:
  - Qué consideramos inicio de respuesta/retorno útil (toda la solución vs. una parte)
  - Precisar el plazo máximo para ejecución.
  - Precisar la frecuencia de iteración (si son periódicas)
  - Considerar sincronización de tareas
  - Nivel de criticidad (estricto, no estricto, firme)



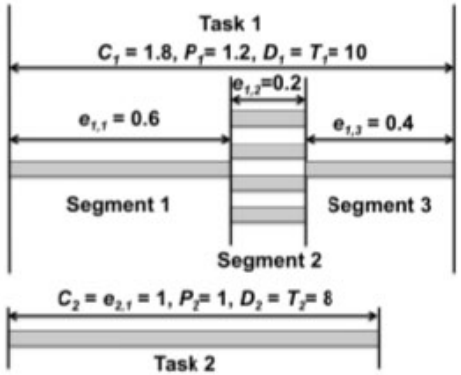
Consideraciones extras para el Diseño

- Especialmente **si las plataformas subyacentes** (OS, HPC framework) **no pueden garantizar el cumplimiento** de estos criterios (i.e: condiciones generales de funcionamiento, no posibilidad de especificar los criterios de fallo, etc ..., **puede resultar necesario la realización de un planificador (scheduler) y lanzador (dispacher)** parte de nuestro código a fin de regular dicho control.

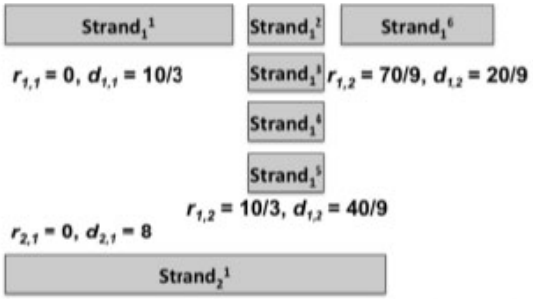


Consideraciones extras para el Diseño

- Objetivos de scheduler:
  - descompone una tarea paralela en un conjunto de hebras secuenciales, cada una con su propio tiempo de liberación y fecha límite
  - asignación de prioridad y partición algoritmo que establece prioridades para cada secuencia hebra



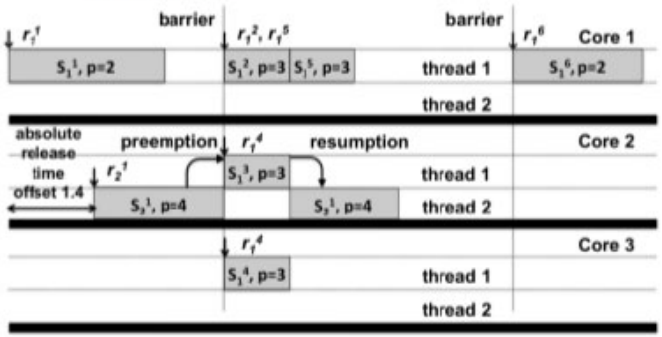
(a) Task set consists of two tasks.



(b) The decomposed tasks on 3 processors. Each strand has its own release time and deadline.

strand	deadline	priority	core
s_1^1	10/3	2	1
s_1^2	40/9	3	1
s_1^3	40/9	3	2
s_1^4	40/9	3	3
s_1^5	40/9	3	1
s_1^6	20/9	1	1
s_2^1	8	4	2

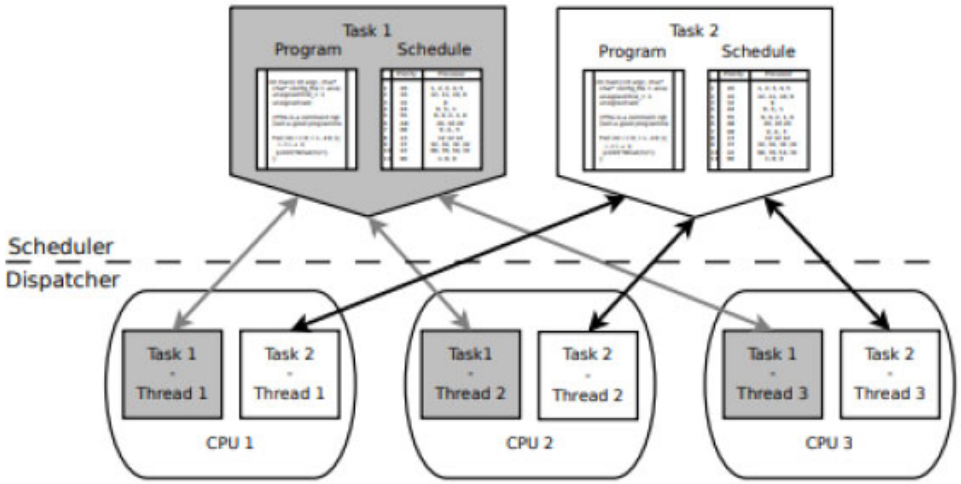
(c) Priority Assignment and Partitioning



(d) Execution trace of the strands execution

Consideraciones extras para el Diseño

- Objetivos de dispatcher:
  - es responsable de hacer cumplir los horarios generados previamente y proporcionar sincronización al final de cada segmento durante el tiempo de ejecución.
  - Esto requiere que el vigilar las prioridades de programación, la preferencia en el tiempo de ejecución y la sincronización utilizando los servicios de capas inferiores



Ajustes del Sistema y Estudio de Umbrales

- Poniendo medios para el control de condiciones de fallo (no se respetan los timelines), se pueden establecer pruebas de carga estadística y obtener umbrales de asignación con los que realizar el ajuste y administración de carga del sistema.