Programación Distribuida y Tiempo Real

Movilidad de Código

Movilidad de Código

- ¿Datos o código?
- Hasta ahora: múltiples formas de transferir información/datos
- Transferir código ejecutable o procesos en ejecución
    - En principio: algo hay que transferir en un sist. distr.
- Se usan como sinónimos (no necesariamente lo son)
    - Movilidad de código
    - Migración de código/procesos

Movilidad de Código

- ¿Datos o código?
- Razones para migrar código
  - Eficiencia: repartir carga computacional y/o disminuir carga de la red de comunicaciones
  - Permitir la carga dinámica de código
    - Código no conocido a priori
    - Código por demanda
    - Mejorar la distribución/instalación del código
  - Últimamente, más relacionado con la eficiencia: mejorar la capacidad o velocidad de respuesta para un usuario

Movilidad de Código

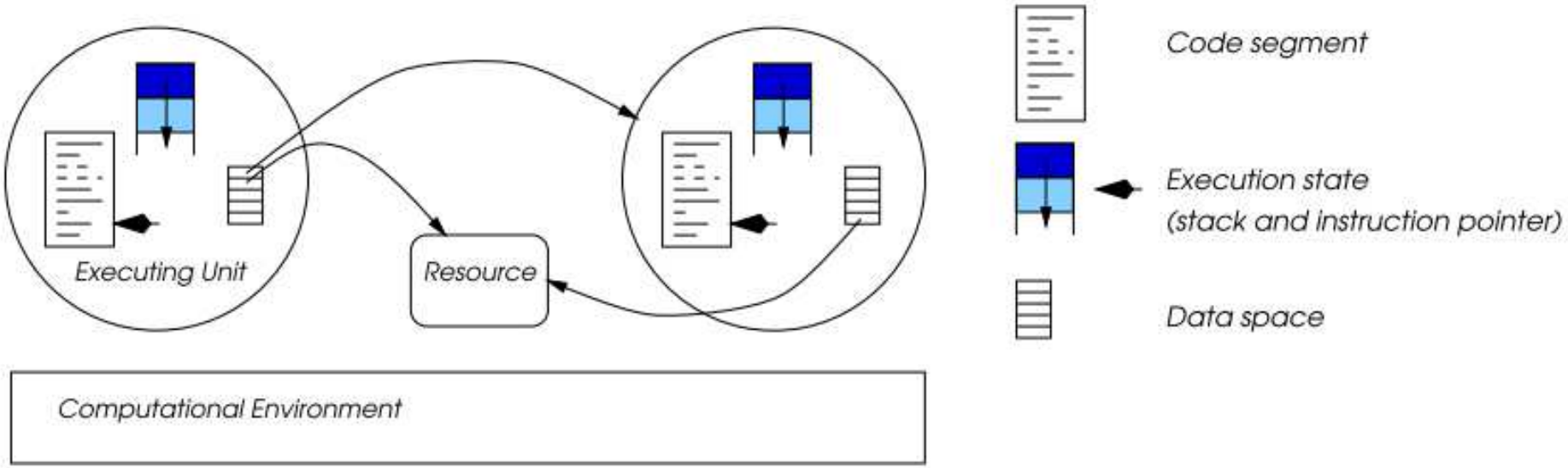
- "Migración" ==> Interpretación histórica
  - Proceso en ejecución
  - Balance de carga computacional en sistemas paralelos
    - Sobrecarga en algunas computadoras-CPUs
  - En algunos casos: checkpoint-restart
    - Tolerancia a fallas
  - Código + Datos + *Estado* de Ejecución
  - + … otros recursos

Migración: Interpretación Histórica

- Lo que tiene un proceso en ejecución (1998_fuggetta_picco_vigna_understanding.pdf)
  - Código: binario ejecutable de un proceso
  - Estado de avance de ejecución
    - Estado de ejecución: recursos no compartidos y propios de la ejecución, (pila, registros)
    - Espacio de datos: recursos a los que se accede/utiliza (en t de ejec.) como archivos o impresoras

Migración: Interpretación Histórica

- Lo que tiene un proceso en ejecución (1998_fuggetta_picco_vigna_understanding.pdf)



Modelos de Movilidad de Código

- Según lo que se transfiere (tipos de movilidad)
  - Débil: solamente el código de un proceso. No es un proceso en ejecución sino el código ejecutable
  - Fuerte: código y estado del proceso. Es un proceso en ejecución, migración *tradicional*
- Según dónde se inicia la movilidad
  - Proactiva: el origen del código inicia la transferencia
  - Reactiva: el destino del código inicia la transferencia
  - Relación entre “sincronismo” y tipos de movilidad

Modelos de Movilidad de Código

- En el caso de movilidad débil: dónde (contexto) se ejecuta el código móvil
  - En el proceso receptor, se transfiere una porción de código, ej: JavaScript
  - En un proceso separado, ej: Applet Según lo que se transfiere (tipos de movilidad)
- En el caso de movilidad fuerte: qué se hace con el proceso original
  - Migrar: el proceso literalmente se “mueve” y deja de existir en el sistema inicial/original
  - Clonar: se crea una copia exactamente igual en otro sistema y ambos coexisten

Modelos de Movilidad de Código

- El gráfico (quizás clásico) de movilidad de código (Tanenbaum, 2Ed., migration vs. mobility):

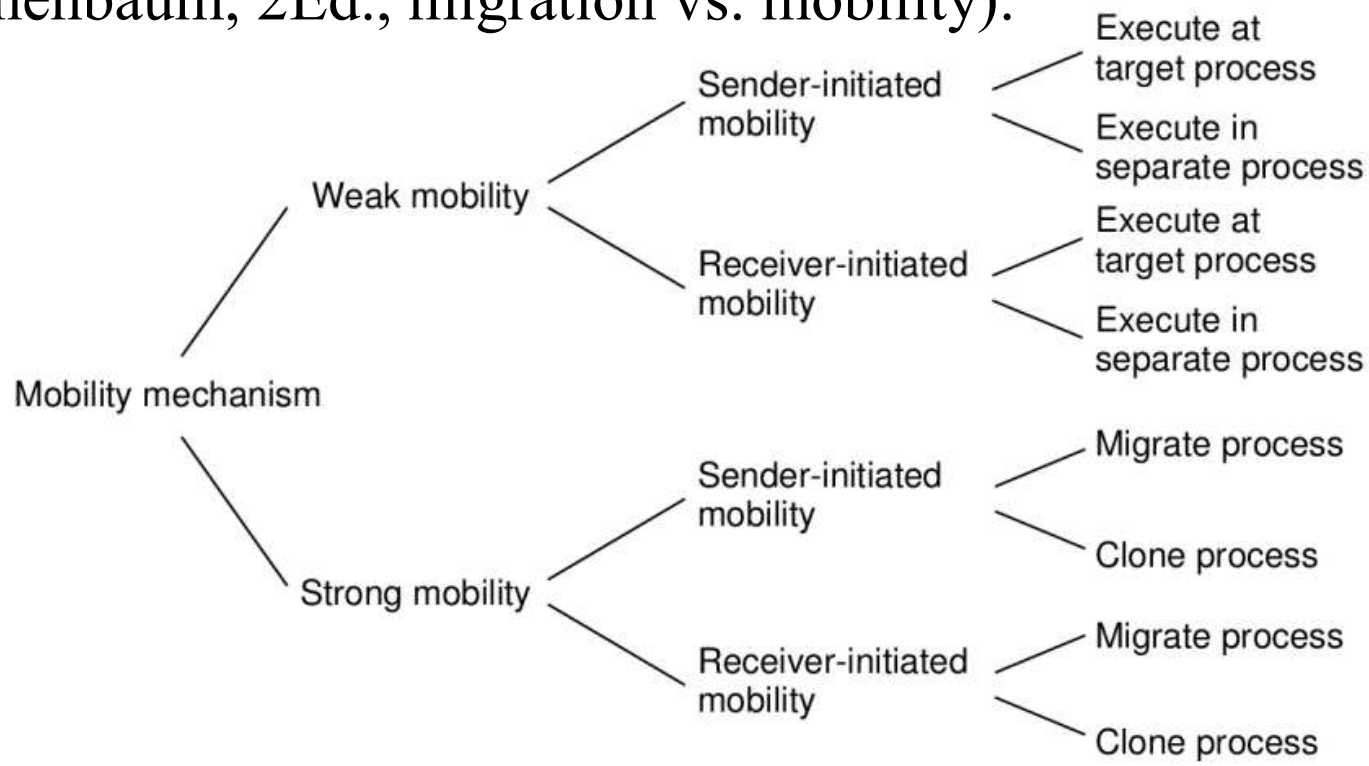


Figure 3.18: Alternatives for code migration.

Modelos de Movilidad de Código

- El gráfico (quizás clásico) de movilidad de código (Tanenbaum, 2Ed., migration vs. mobility):

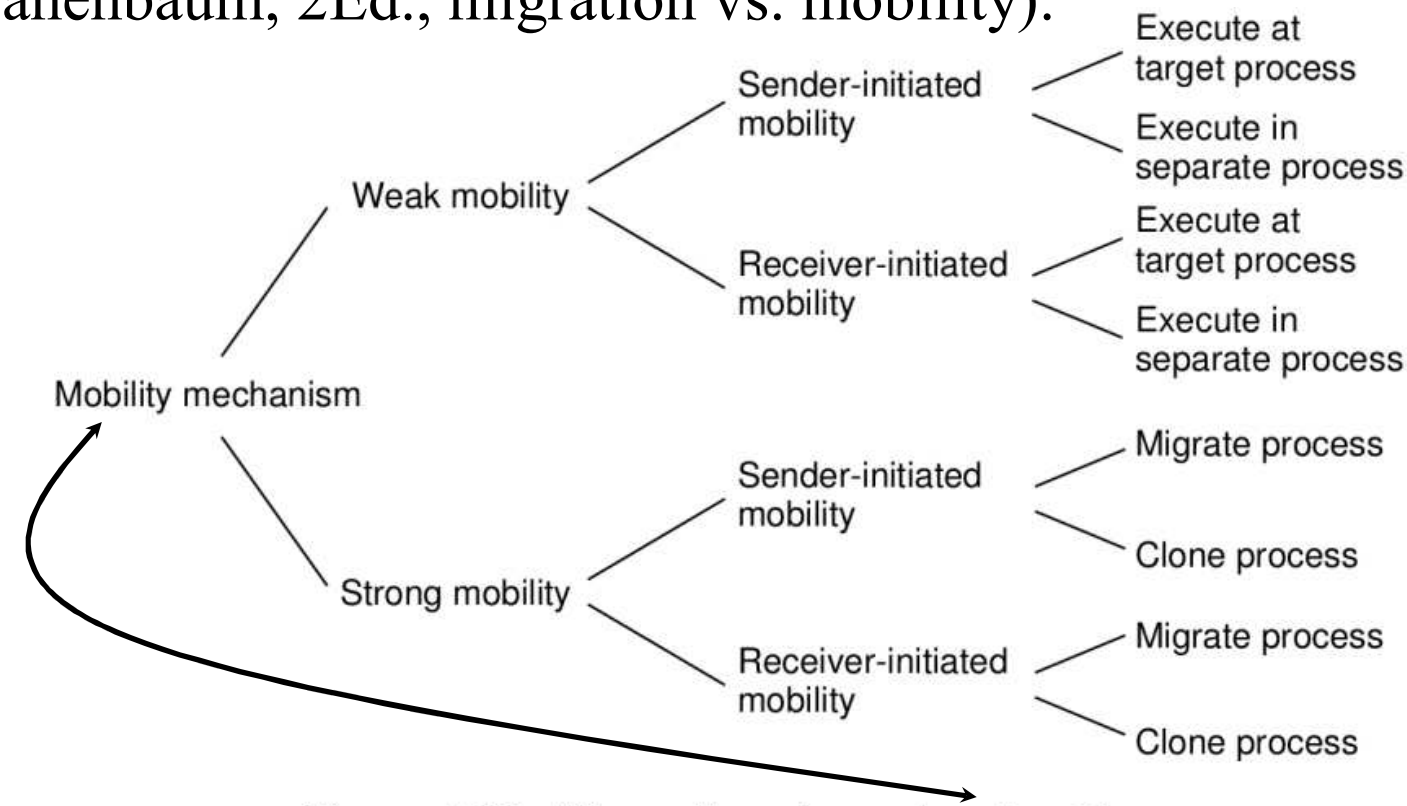


Figure 3.18: Alternatives for code migration.

Modelos de Movilidad de Código

- Movilidad fuerte
  - Casi descartada en general
    - Complejidad
    - Utilidad de movilidad débil
  - El enorme problema de los recursos

Modelos de Movilidad de Código

- Movilidad fuerte
  - Casi descartada en general
    - Complejidad
    - Utilidad de movilidad débil
  - El enorme problema de los recursos

Resource-to-machine binding

Process-to-resource binding	Unattached	Fastened	Fixed
By identifier	MV (or GR)	GR (or MV)	GR
By value	CP (or MV,GR)	GR (or CP)	GR
By type	RB (or MV,CP)	RB (or GR,CP)	RB (or GR)

GR	Establish a global systemwide reference
MV	Move the resource
CP	Copy the value of the resource
RB	Rebind process to locally-available resource

Dudas/Consultas

- Plataforma Ideas