Computación distribuida y agentes móviles

Andrés Marín
Departamento de Ingeniería Telemática
Universidad Carlos III de Madrid
amarin@it.uc3m.es

Indice

- Protocolos previos a la computación distribuida (RPCs, HTTP, NFS)
- Computación distribuida
- Movilidad de datos y código: migración de procesos
- Agentes Móviles:
  - Introducción
  - IBM Aglets
  - MASIF

Introducción

- Las redes nos permiten distribuir cálculos, datos, compartir recursos, etc.
- Alternativas:
  - Modelo cliente/servidor: RPCs, HTTP
  - Movilidad de datos: NFS
  - Computación distribuida
  - Movilidad de código: Migración de procesos
  - Programación remota: agentes móviles

PROTOCOLOS PREVIOS A LA COMPUTACIÓN DISTRIBUIDA (RPCS, HTTP, NFS)

Remote Procedure Calls (RPCs)

- Sun Microsystems, Inc. (RFC 1057)
- Protocolo de mensajes para acceso a procedimientos remotos
- Uso de representaciones de datos independientes de la arquitectura (XDR)
- Utilización de stubs (rpcgen) para adecuar las llamadas y las respuestas.

Remote Procedure Calls (RPCs)

- Distintas versiones
- Servicio de portmaper (rpcinfo)
- Poco flexible o personalizable
- Trasiego elevado de datos
- Optimización de servidores

HTTP

- Servicio de información basado en el modelo cliente servidor (RFC 1945)
- Utilización de MIME para contenidos multimedia
- Protocolo sin estado (encima de TCP):

HTTP: personalización

- El servidor es genérico y el protocolo sin estado
  - La personalización requiere procesamiento adicional en el servidor
  - Utilización de cookies y campos ocultos para mantener un estado
  - Alternativa: HTTP/1.1(RFC 2616)
- Unión “transparente” de conexiones

NFS

- Sun Network File System (RFC 1094 y RFC 1813)
- Acceso remoto transparente a ficheros a través de redes
- Portable mediante el uso de Sun RPC
- Sistema de ficheros en red (granularidad)
- Requiere red de buenas prestaciones (LAN)
- Otras alternativas: NBD (Network Block Driver)

COMPUTACIÓN DISTRIBUIDA

Movilidad de código

- Computación distribuida:
  - DIPC (Distributed Inter-Process Communication)
  - PVM (Parallel Virtual Machine – procesador distribuido)
  - Clusters
  - Hurds (cluster trasparente)
- Migración de procesos:
  - Worms (Write Once Read Many)
  - MOSIX (Sistema gestion clusteres Linux)
  - Sprite (Experimental Unix-Like O.S – Clustering)

Computación Distribuida

- Agenda:
  - Caracteristicas y dificultades
  - Tipos de sistemas distribuidos
  - Concepto de transparencia
  - Ventajas de la distribución
  - Desventajas de la distribución
  - Sistemas de computación paralelos

Computación distribuida

- Caracteristicas y dificultades
  - Dificultad de distribuir el algoritmo
  - Ayudas del compilador y entorno
  - Equilibrio entre procesamiento en los nodos y comunicaciones entre nodos
  - Balance dinámico de carga

- Modelos: SIMD (**Single instruction, multiple data**), MISD, MIMD
- http://wallybox.cei.net/dipc
- http://www.epm.ornl.gov/pvm/intro.html

Tipos de sistemas distribuidos

- Sistemas débilmente acoplados: Single Instruction Single Data (SISD)
- Sistemas fuertemente acoplados: MIMD
- Procesadores sistólicos (en array): SIMD. Procesamiento de conjuntos regulares de datos a alta velocidad. Hypercube Machines, pipelining.

Transparencia

- Propiedad de un sistema para ser percibido como un todo y no como un conjunto de componentes (independientes).

- Niveles:
  - •Acceso
  - •Localización
  - •Concurrencia
  - •Replicación
  - •Fallos
  - •Migración
  - •Prestaciones
  - •Escala

Ventajas de la distribución

- Tiempo de ventaja predecible
- Extensibilidad
- Compartición de recursos
- Replicación (datos)
- Alta disponibilidad (tolerancia a fallos)
  - Minimización del TMF (Tiempo Medio entre fallos)

Desventajas de la distribución

- Pérdida de flexibilidad en la asignación de memoria y recursos
- Dependencia de la red (velocidad y fiabilidad)
- Debilidades de seguridad

Sistemas de computación paralelos

- P4:
  - librería de macros y subrutinas
    - Facilita la programación, abstrae
  - memoria compartida (monitores)
    - Gestiona los recursos
  - memoria distribuida (send, recv)
    - Permite replicar, copiar…
  - implementación de clusters
- Express:
  - soporte al desarrollo: seq-paralelo
    - Transformación de algo secuencial en algo en paralelo
  - sintonización mediante vtool

Sistemas de computación paralelos

- MPI (Message Passing Interface):
  - Sintáxis y semántica de conjunto de librerías de paso de mensajes
  - Portabilidad a MPPs y procesamiento paralelo
- Linda (Coordinación de comunicaciones):
  - Modelo de programación concurrente
    - in atomically reads and removes—consumes—a tuple from tuplespace, rd non-destructively reads a tuplespace, **out** produces a tuple, writing it into tuplespace, **eval** creates new processes to evaluate tuples, writing the result into tuplespace
  - Espacio de tuplas
  - Basado en memoria compartida (tuplas asociativas)

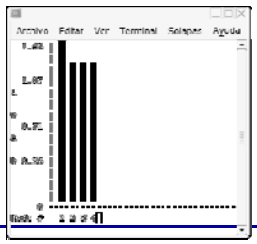
Sistemas de computación paralelos

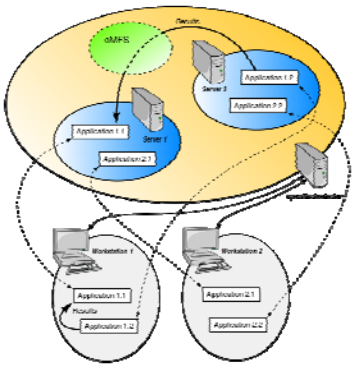
- PVM
  - Parallel Virtual Machine
  - Portabilidad
  - Distribución sencilla
  - No hay gran soporte al diseño (seq-paralelo)
  - http://www.netlib.org/pvm3/book

MOVILIDAD DE DATOS Y CÓDIGO: MIGRACIÓN DE PROCESOS

Migración de procesos

- Soportada por el kernel:
  - MOSIX – Open Mosix (***openmosix***.*sourceforge.net) Cerrado*
  - Sprite
  - Charlotte, Mach, Locus, V Kernel
- En espacio de usuario:
  - Emerald
  - Tui





Beneficios

La migración de procesos:

- permite el balance de carga
  - Varios procesos corriendo en diferentes procesadores
  - Equilibrio del % de uso
- mejora la tolerancia a fallos
  - Las piezas que lo componen se pueden reemplazar con coste bajo
- mejora el acceso a datos locales (disminuye el coste de comunicaciones)
  - Replicación

Worms

- Shoch & Hupp (Xerox PARC)
- Programas que se mueven en distintas máquinas y se replican en máquinas poco cargadas
- Un gusano está formado por múltiples segmentos cada uno corriendo en una máquina
- Búsqueda de máquinas libres a las que enviar los segmentos
- Cada segmento conoce la ubicación de los demás y se comunican via multicast

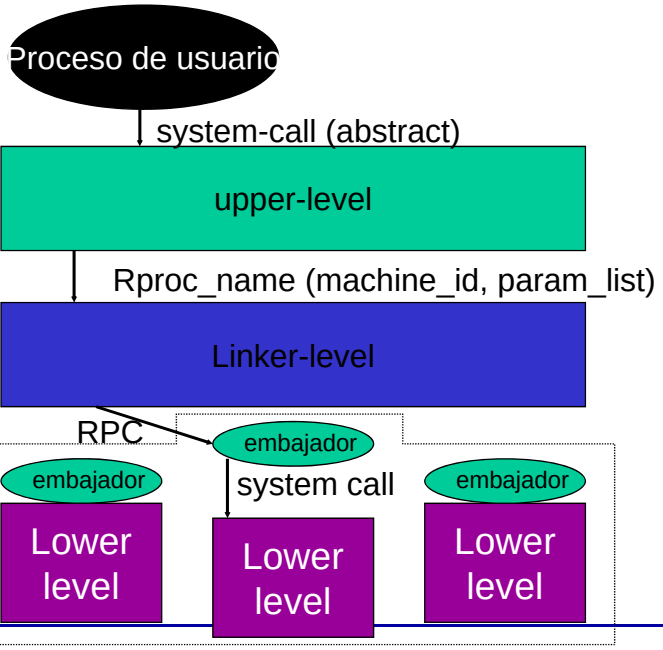
MOSIX

- Barak & Wheeler (Hebrew Univ Jerusalem)
- http://www.mosix.cs.huji.ca.il
- Todos los nodos de un cluster se presentan como una única imagen del sistema:

  Un proceso corriendo en un nodo accede a los otros de forma transparente
- Migración automática (cambios de contexto) entre nodos débilmente acoplados
- Kernels interactúan al nivel de llamada al sistema
- Máquinas con varios procesadores: master/slave

MOSIX: arquitectura

- Lower-kernel
  - rutinas de acceso a recursos locales (discos locales, etc.)
- Upper-kernel
  - nivel independiente de la máquina, referencias a todos los recursos
- Utilización de RPCs para la ejecución de las llamadas al sistema en la máquina concreta
- Linker-level
  - descompone las llamadas a RPCs en llamadas a procedimientos y lista de parámetros
- Hilo Embajador
  - invoca el procedimiento del lower-kernel



MOSIX: características

- Kernel replicado en cada nodo
- Transparencia de acceso a bus y red
- Control descentralizado (decisiones indep.)
- Autonomía de cada nodo
- Sistema de ficheros unificado
- Escalado (heurísticos de gestión y migración)
- Balance de carga adaptativo
- Configuración dinámica (añadir/quitar nodos)
- Compatibilidad con AT&T UNIX (Linux)

Sprite

- Douglis & Ousterhout (UCB)
- Evacuación y ejecución remota de procesos
- Localización de máquinas ociosas
- Gestión del estado:
  - •memoria virtual
  - •ficheros abiertos
  - •canales de mensajes
  - •estado de ejecución
  - •otros estados del kernel (pid, uid, cwd, etc.)

AGENTES MÓVILES: INTRODUCCIÓN IBM AGLETS, MASIF

Agentes móviles

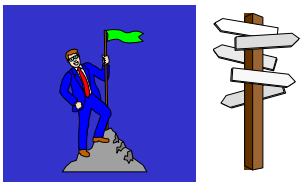
- Grandes esperanzas depositadas en grandes plataformas
- Metodologías, APIs, librerías
- Los lenguajes interpretados son más sencillos de incorporar y controlar
  - agentes controlados por la VM (o intérprete)
  - VM controlada por la plataforma


Ventajas de agentes móviles

- Disminución del volumen de comunicaciones
- Autonomía, toma de decisiones
- Representación del usuario: perfil, interés, referencias a UI/GUI
- Extensibilidad: cooperación entre agentes
- Capacidad de adaptación


Places

Una red de ordenadores engloba un conjunto de sitios que ofrecen servicios a los agentes móviles que entren:



Personal communicator

Sitio: proveedor de servicios
Plataforma de agentes móviles

Movimiento

- Los agentes se mueven de una plataforma a otra de forma autónoma
- Plan de viajes
- Serialización de los agentes (estado)
- Envío de código necesario
- Creación, arranque, interrupción, parada y rearranques (eventos?)

Comunicaciones

- Encuentros:
  - Varios agentes se reunen en una misma plataforma
- Conexiones:
  - Intercambio asíncrono de información entre agentes en distintas plataformas
  - Mensajes, pizarras, etc.

Seguridad

- La autoridad de un agente es el usuario o entidad física (o jurídica) a que representa.
- Una región es un conjunto de redes operados por una misma autoridad
- Una plataforma destino comprueba la autoridad de un agente que quiere entrar en ella
- La plataforma comprueba certificados y administra las listas de acceso a recursos

Telescript

- General Magic: http://www.genmagic.com
- Lenguaje: completo, OO, dinámico, persistente, portable, seguro, centrado en comunicaciones
- Secciones críticas (resources)
- Permits (permisos basados en autoridad)
- go <teleadress>
- Eventos de señalización

Ejemplo

```
Shopper: class (Agent, EventProcess) =
( public
      see initialize
      see live
      see meeting
      see getReport
  private
      see goShopping
      see goHome
  property
      client: Telename;
      desiredProduct: String;
      desiredPrice, actualPrice: Integer;
      exception: Exception|Nil;
);
```

Ejemplo: live

```
live: sponsored op (cause Exception|Nil)=
{
      homeName:= here.name;
      homeAddres:= here.address;
      permit:= Permit(
            (if *.permit.age==nil{nil}
            else{(*.permit.age*90).quotient(100)})
                  ,
            (if *.permit.ch==nil{nil}
            else{(*.permit.ch*90).quotient(100)})
            );
      {
        try{
            *.goShopping(Warehouse.name)
      }catch e: Exception{exception=e}}
       catch e: PermitViolated{exception=e};
      {
            try{*.goHome(homeName, homeAddress)}
             catch Exception {}
       );
```

Ejemplo: goShopping

```
goShopping: op (warehouse: ClassName)
throws ProductUnavailable=
{
*.go(Tiket(nil,nil,warehouse);
*.enableEvents(PriceReduction(*.name));
*.signalEvent(PriceReduction(),'responder);
*.enableEvents(PriceReduction(here.name));

actualPrice= desiredPrice+1;
while (actualPrice>desiredPrice)
{
 *.getEvent(nil,PriceReduction());
 try{actualPrice=here@Warehouse.
              getCatalog()[desiredProduct].price;
 }
catch KeyInvalid {throw ProductUnavailable}
}
};
```

Ejemplo: goHome

```
goHome: op (homeName:Telename,
            homeAddress: TeleAddress)=
{
 *.disableEvents();
 *.clearEvents();
 *.go(Ticket(homeName, homeAddress);
 *.enableEvents(PartEvent(client));
 here@MeetingPlace.meet(Petition(client));
 *.getEvent(nil,PartEvent(client))
};
```

Bibliografía

- Mobility: processes, computers, and agents ed. D. Milojicic, F. Douglis, and R. Wheeler ACM Press, 1999
- Mobile Agents, by W. R. Cockayne and M. Zyda, Manning Publications, 1998
- Mobile Agents, ed. by K. Rothermel and R. Popescu-Zeletin, LNCS, April 1997