Sistemas Operativos y Persistencia

Coloquios Abiertos

J. Baltasar García Perez-Schofield
http://www.ei.uvigo.es/~jgarcia/ca/

Departamento de Informática
Área de Lenguajes y Sistemas Informáticos
http://www.lsi.uvigo.es

Grupo IMO
http://www.lsi.uvigo.es/lsi/imo/

Introducción

- ¿Persistencia = Serialización?
  1. JAVA (.class)
  2. C++ (fwrite(&ptr ...)
2. La serialización parece ser la gran contribución de la persistencia.

Persistencia

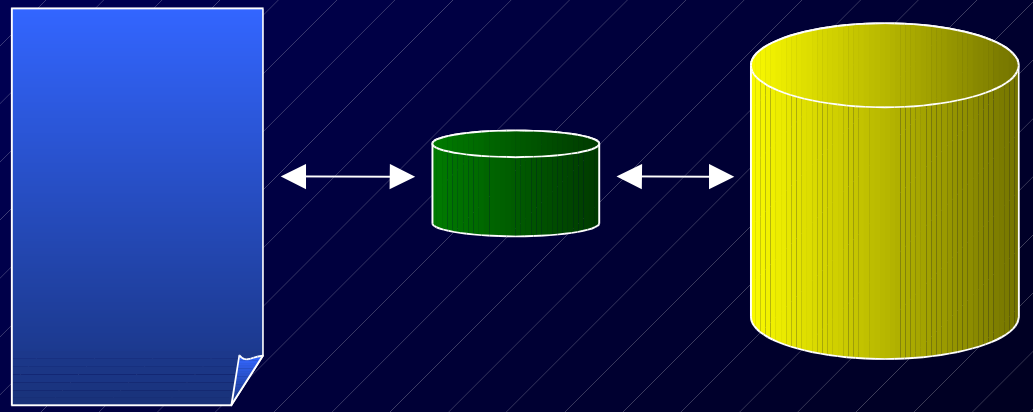
- Guardar y recuperar estructuras de datos de forma transparente y automática.
- Serialización se incluye en Persistencia, pero serialización no implica persistencia.
- Un proceso puede acceder a objetos que creó otro proceso.

Persistencia Ortogonal

- Totalmente transparente para el programador
- Tres principios:
  - Independencia del tipo de objetos persistentes y no persistentes.
  - Independencia de manejo de objetos persistentes y no persistentes.
  - Independencia de identificación de objetos persistentes.

- Introducción
  - Persistencia
  - Persistencia Ortogonal
  - Sistemas Operativos
- Modelo de persistencia
- Rendimiento
- Conclusiones

Persistencia Ortogonal



Programa Persistente

Object Caché

Almacena-miento persistente.

Persistencia Ortogonal

- Por ejemplo, Napier, Pjama, ...
- Puntos fuertes
  - La persistencia es totalmente automática y totalmente transparente al usuario.
- Puntos vulnerables
  - No existen sistemas de organización de la información en el almacenamiento persistente.
  - La transparencia se pierde fácilmente en cuanto se añaden transacciones, evolución del esquema ... etc.

Sistemas Operativos

- Por ejemplo, Windows, Solaris, GNU Linux
- Es el conjunto de programas que permiten un acceso (usabilidad) mínimo a las funciones del hardware.

Evolución de los Sistemas Operativos

Microsoft

- En 1985, sale al mercado MS Windows 1.0, una shell para MS-DOS
- Entre 1987 y 1990 salen Windows 2.0 y Windows 3.0, éste último incluyendo ya *multitarea*.
- En 1993, sale Windows 3.11, incluyendo la red de Windows para trabajo en grupo.

Microsoft: ahora en serio

- En 1993 también sale Windows NT 3.1, como un “UNIX”. Además, Windows NT Workstation soportando OPEN GL.
- En 1995 sale Windows 95. SO de 32 bits, con multitarea “real”.
- En 1996, Windows NT Workstation 4.0
- En 1998, Windows 98 y Windows 98 SE
- En 2000, Windows ME y Windows 2000
- En 2001, Windows XP (Professional, Workstation y Home Edition)
- En 2003, Windows .NET Server.

GNU Linux: la alternativa alternativa

- En 1984 Stallman había abandonado el MIT, y dedicado su vida a la hostelería y a desarrollar el software GNU.
- En 1991 Linus Torvalds escribe un kernel para el sistema Minix.
- Nace GNU/Linux, un sistema operativo con un kernel al margen del proyecto GNU y que va ganando adeptos debido a su gratuitidad.
- Distribuciones Red Hat, Mandrake, SuSE ...

Hitos en la evolución de los Sistemas Operativos

- UNIX (principios de los 70) es la gran referencia en Sistemas Operativos:
- Es el “padre” de Linux, Solaris, Minix ... Windows NT ...
- TODOS los sistemas operativos comerciales de la actualidad incorporan su arquitectura, su forma de trabajar: la metáfora de ficheros.

Hitos en la evolución de los Sistemas Operativos

- Smalltalk (principios de los 70), de los laboratorios Xerox en Palo Alto, es el que introduce la idea de un sistema de ventanas como interfaz de usuario.
- Las tareas se asimilan a una ventana sobre un escritorio, en la forma de papeles sobrepuestos.
- ... incluso hay una papelera ...

“Evolución” de los Sistemas Operativos

- Desde un punto de vista arquitectural, los sistemas operativos continúan utilizando una metáfora de ficheros como forma de trabajo.
- La evolución ha llegado desde la propia interfaz de usuario, que permite cierta Orientación a Objetos.

“Evolución” de los Sistemas Operativos

- Se utiliza algún “rasgo” del fichero como identificador del tipo de objeto/fichero.
- En sistemas Windows, la identificación llega desde la extensión del archivo.
- En sistemas tipo Unix, se trata de un número (MAGIC) en la cabecera del mismo.

¿Por qué no han evolucionado los Sistemas Operativos?

¿Por qué utilizamos una “metáfora” de objetos sobre una “metáfora” de archivos?

... en definitiva, si utilizamos objetos, ¿Por qué no un SOOO?

¿Es ***conservador*** el mundo de la informática?

El modelo de persistencia basado en contenedores

Modelo basado en Contenedores

- *Swizzling*: Conversión de punteros del formato en memoria principal al formato en memoria secundaria y viceversa.
- *Contenedor*: Objeto de gran grano u objeto contenedor de otros objetos de grano fino, a los que encapsula; exceptuando aquellos de *interface*, explícitamente públicos (pero de sólo lectura).

Clustering

- ¿Cómo agrupar objetos en el almacenamiento persistente?
  - Objetivo: optimización del proceso de recuperación de objetos.
  - ... ¿cuáles deberían ser grabados conjuntamente (en el mismo cluster) ?

Protección de memoria

- Es necesario asegurar que el almacenamiento persistente no se corrompe por código erróneo/malicioso.
- En sistemas persistentes ortogonales, sólo es posible asegurar ésto mediante lenguajes seguros respecto al tipo.
- ... ya que no es posible proteger los objetos en el almacenamiento persistente.

Contenedores

- Sólo ofrece ortogonalidad al tipo.
- Se trata de obtener un mejor rendimiento con este modelo.
- Se utiliza una abstracción de directorios que tratar de conseguir las mejores características de un sistema de ficheros y de un sistema de objetos.
- El programador utiliza esos directorios de una manera muy intuitiva.

Modelo basado en contenedores

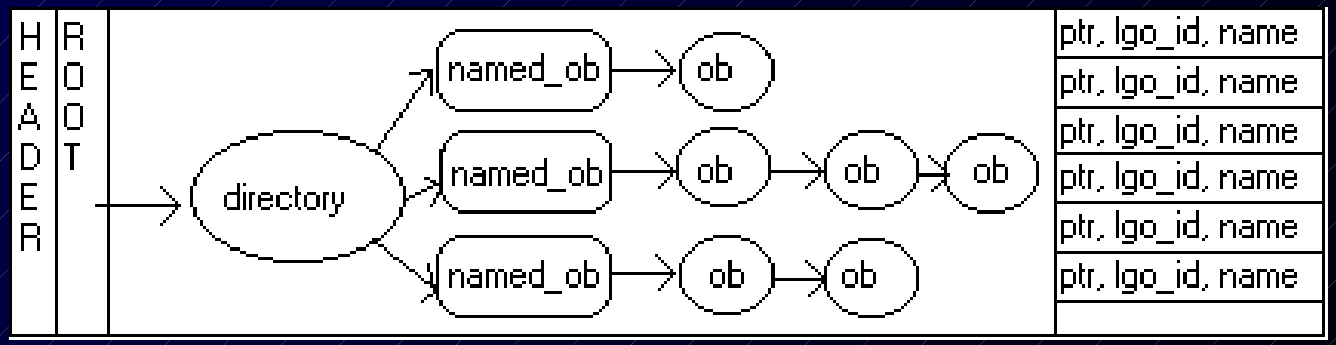
- Implementado en el prototipo *Barbados*:
  - La organización del almacenamiento persistente consiste en el uso de los *containers*.
  - Compilador de C++ a código nativo **intel**.
  - Swizzling no basado en OID's.
- Transparencia:
  - La organización basada en *containers* se esconde bajo una metáfora de directorios.
  - La asociación container=directorio permitirá otras características como asignación de permisos ... etc.

Modelo basado en contenedores

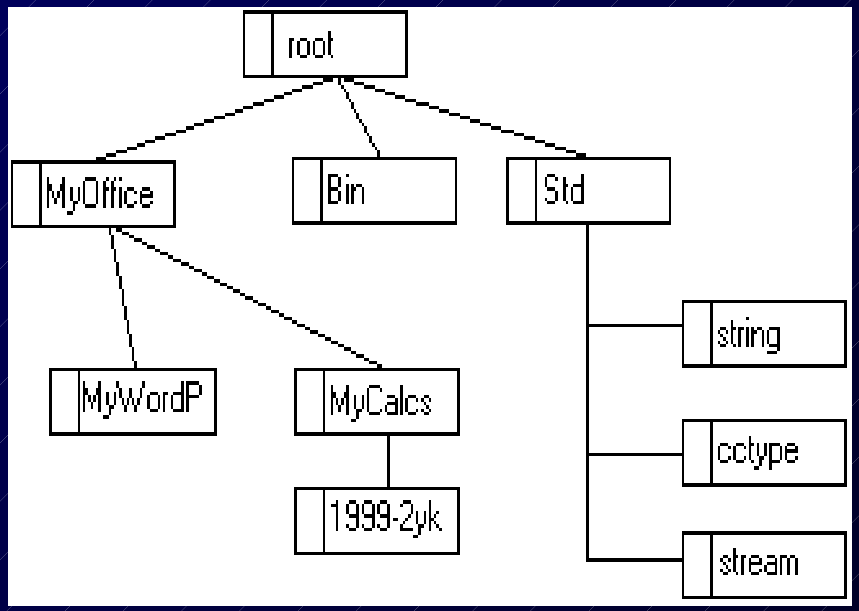
- Los *contenedores* favorecen la compartición del almacenamiento persistente.
- Así, se evita que un error o un programa malicioso corrompa el almacenamiento persistente.
- Sin embargo, es necesario permitir una comunicación, aunque restringida, entre *containers*.

Contenedores

- Almacenamiento persistente totalmente particionado.
- ... con “comunicación” restringida (compartición de datos)



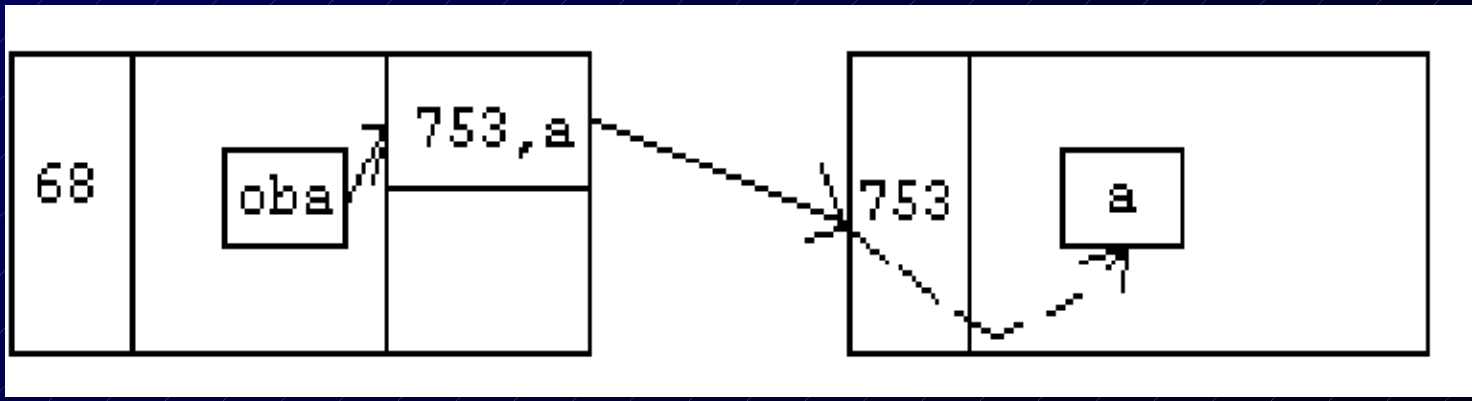
Contenedores y Directorios



- Los directorios permiten organizar objetos de una manera muy intuitiva.
- Un contenedor *es un* directorio.
- Para referenciar objetos, se utiliza el estilo *path*: /std/cctype/toupper();

- Introducción
- Modelo de persistencia
  - Contenedores
  - Contenedores/ Directorios
  - C-N Swizzling
- Rendimiento
- Conclusiones

C-N Swizzling



¿Cómo funciona?

C++ con ficheros

```
int main(void) {
    int x;
    FILE *f  = fopen(“datos.dat”);
    if (f!=NULL) {
        do {
            fread(&x, sizeof(int), 1, f);
            cout << x << endl;
        } while(!feof(f));
        fclose(f);
    } else cerr << “Error de E/S” << endl;
}
```

Pjama

```
public class ejemplo {
  public static void main(void) {
     PJavaStore pjs = PJavaStore.getStore();
     int [] vectnum = (int []) pjs.getPRoot(“vect_ejemplo”);
     for (int j = 0; j < vectnum.size; ++j)
          System.out.println(vectnum[j]);
  }
}
```

Barbados

```
void ponVectorEjemplo(void) {
    int *vectnum = /ejemplos/ca/vect_ejemplo;
    while (listanum != NULL)
        cout << *(listanum++) << endl;
}
```

Rendimiento

Introducción

- Evaluación del rendimiento del prototipo.
- ¿Cuán “rápido” es?

Objetivo

- ¿Puede ser este sistema persistente tan eficiente como un sistema tradicional?

Más información:

Cooper, T. "*Barbados: An Integrated Persistent Programming Environment*". PhD Thesis. Basser Department of Computer Science, Univ. of Sydney, Sydney, Australia.

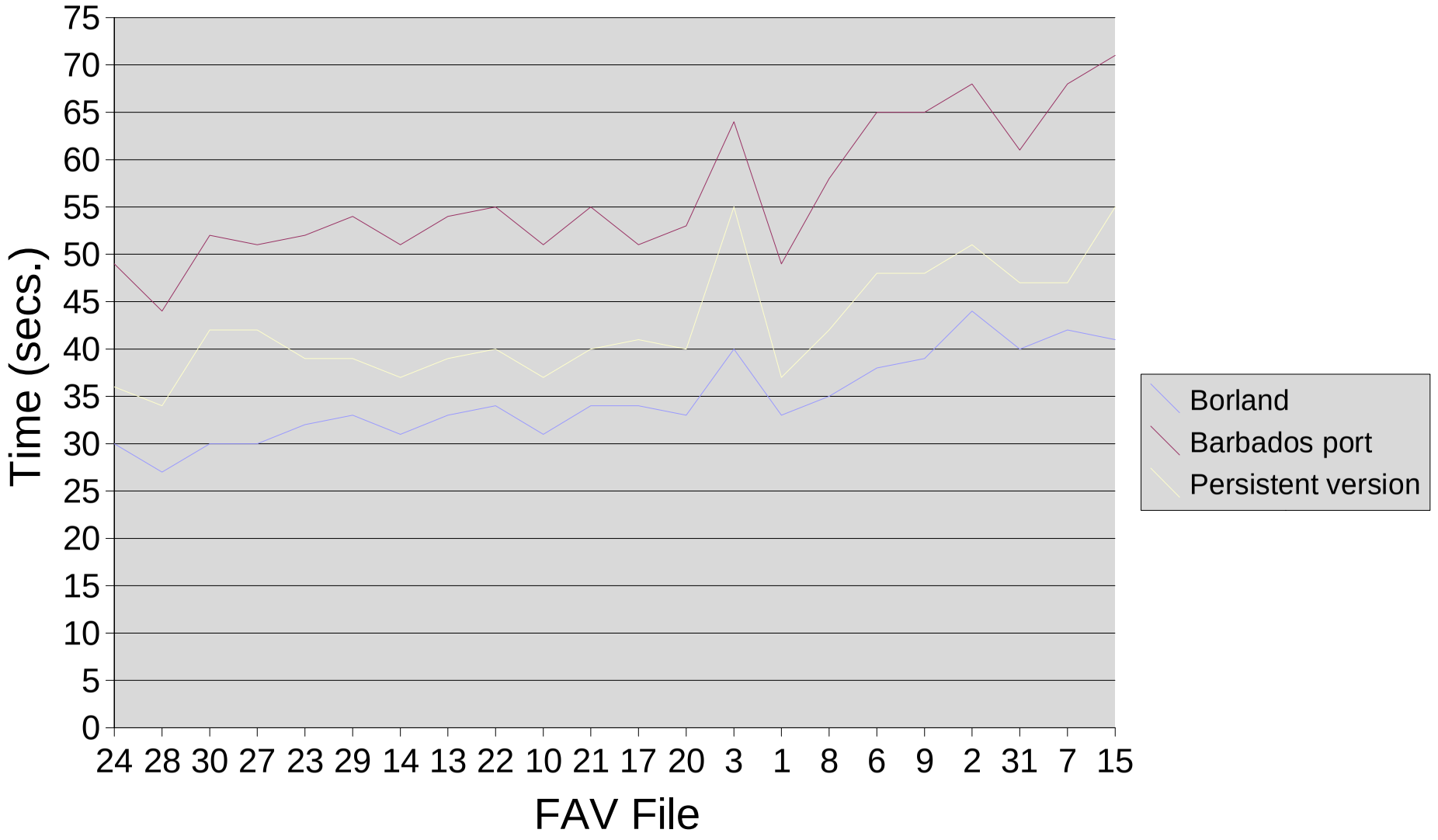
Objetivo

- Tomar un módulo de una aplicación de Smarts, Pty, e implementarla en Borland C++.
- Adaptar esa aplicación a Barbados.
- Comparar el rendimiento de ambas aplicaciones.
- Transformar la aplicación en una aplicación persistente.
- Comparar de nuevo su rendimiento.

Resultados

- La versión Barbados no persistente es en media un 62% más lenta que la versión convencional.
- La versión Barbados persistente es sólo un 22% más lenta que el compilador.
- En otra versión de Barbados, con código interpretado, el rendimiento era un 2000% peor.

Performance results ordered by number of messages



Conclusiones

Conclusiones

- El campo de investigación sobre persistencia no ha conseguido demostrar, sin lugar a dudas, su lugar dentro de los sistemas operativos actuales .
- El modelo de contenedores ofrece varias ventajas, concretamente en relación al rendimiento.
- La evolución natural apunta a un entorno orientado a objetos, probablemente mezclado con algunas características de los sistemas de ficheros.

Sistemas Operativos y Persistencia

Coloquios Abiertos

J. Baltasar García Perez-Schofield
http://www.ei.uvigo.es/~jgarcia/

Departamento de Informática
Área de Lenguajes y Sistemas Informáticos
http://www.lsi.uvigo.es

Grupo IMO
http://www.lsi.uvigo.es/lsi/imo/