Análisis y diseño de software

dit
UPM

ejercicio 6

José A. Mañas

20.4.2016

ETSIT UPM

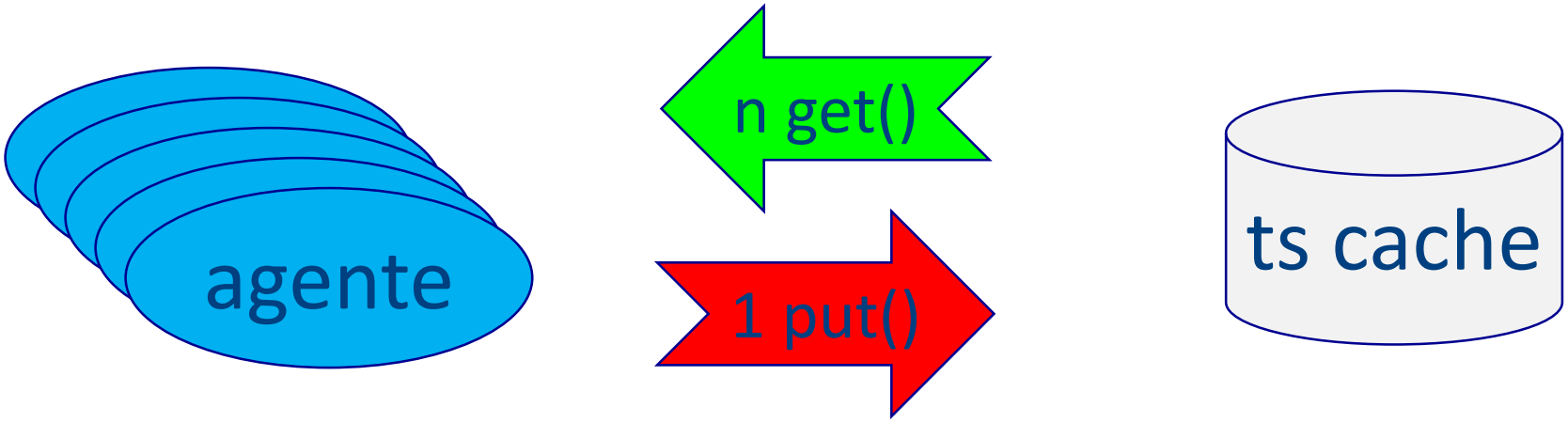
temas

- concurrencia
  - zonas de exclusión mutua
  - espera condicional: wait()
  - notificaciones: notify() & notifyAll()
- programar y probar

tareas

- programar una cache thread-safe
  - una caché es un diccionario
    - ej. url → página web
  - las tareas miran si está en la cache
    - si está usan el valor almacenado
    - si no, descargan la página web y la almacenen
  - varias tareas recorren la web concurrentemente
    - hay que evitar que la cache se corrompa
    - hay que permitir que varias tareas lean a la vez
    - hay que evitar que varias tareas escriban a la vez

tareas



clases java

javadoc

- class TsCache
- class TsList
- class RW_Monitor

- class TsCacheSmokeTest
- class TestAgent implements Runnable

- class CV
- class My
- class Nap
- class LogViewer

agente

```
@Override
  public void run() {
     while (true) {
        try {
           String key = String.valueOf(random.nextInt(1000));
           if (cache.get(key) == null) {
              String val = "{" + key + "}";
              Nap.random(10, 20);
              cache.put(key, val);
           }
           Nap.sleep(10);
        } catch (Exception e) {
           e.printStackTrace();
        }
     }
  }
```

implementación

- TsCache es una tabla hash con listas de desbordamiento (TsList)
  - la identificación de la ranura puede hacerse concurrentemente, sin zonas de exclusión
  - cada ranura tiene su TsList
- TsList es una lista clásica
  - las operaciones get(), put(), remove() y clear() deben cuidar que no se corrompa la lista
  - puede haber múltiples operaciones get() al tiempo
  - solo puede haber una operación de modificación en cada momento
  - problema readers-writers

TsCache & TsList

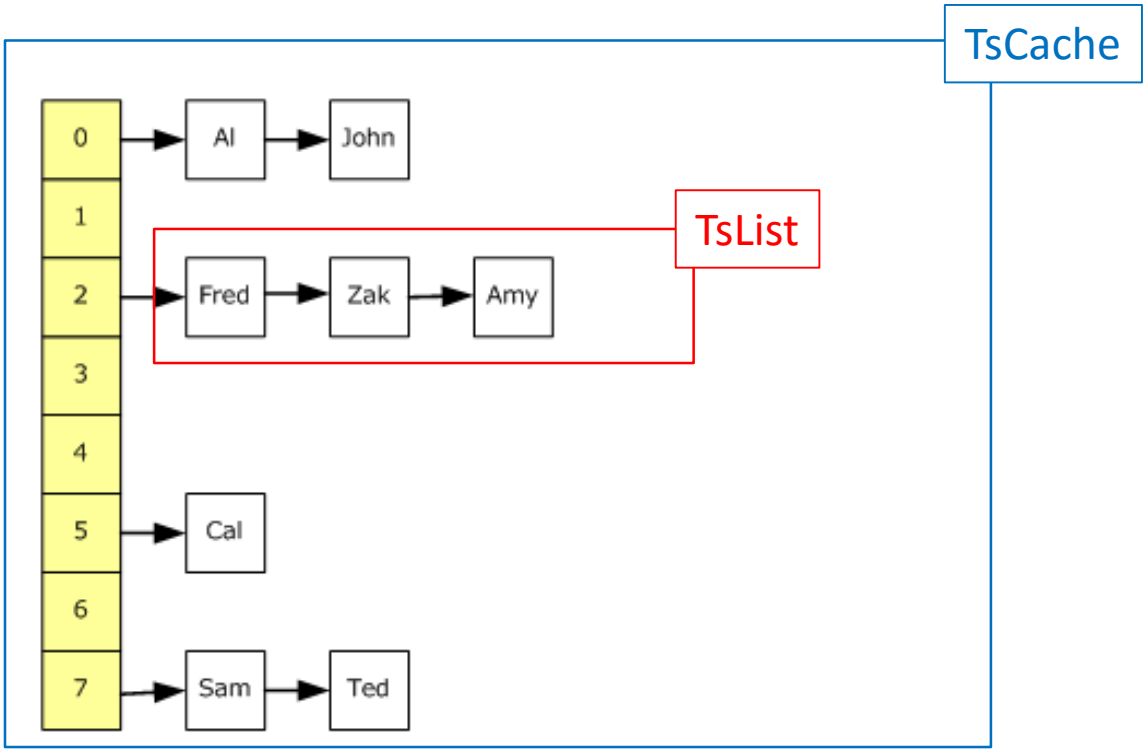


Figure 11. Chain created by a series of items hashed to the same bucket

https://msdn.microsoft.com/en-us/library/ms379571(v=vs.80).aspx

prueba - TsCacheSmokeTest

1. se crea una TsCache con 10 ranuras
2. se lanzan 50 TestAgent

- si las zonas de exclusión están bien
  - no deben saltar excepciones
  - no deben fallar las aserciones
  - deben verse las colas funcionando concurrentemente, con varios lectores simultáneos o 1 escritor aislado

pruebas unitarias: TsCacheTest

- se pueden hacer pruebas unitarias, JUnit,
  - sin concurrencia,
  - para validar que las operaciones get() y put() funcionan adecuadamente
  - reuse las del ejercicio 3

pruebas

- corrección (correctness)
  - junit:
    salen o son desechados según los criterios apuntados
- seguridad (safety)
  - el estado no se corrompe;
    un estado corrupto genera excepciones
  - se satisfacen las condiciones del problema
- vivacidad (liveness)
  - se guardan y se recuperan datos;
    el sistema no se queda congelado
- equidad (fairness)
  - todos los agentes progresan
  - use LogViewer

NOTA: pasar todas las pruebas no implica que no haya errores;
pero unas buenas pruebas reducen la probabilidad de que queden.

aserciones

- cuando conseguimos un permiso de lectura, verificaremos que
  - monitor.getNWritersIn() == 0
- cuando conseguimos un permiso de escritura, verificaremos que
  - monitor.getNWritersIn() == 1
  - monitor.getNReadersIn() == 0
- puede usar la clase auxiliar My

LogViewer

- Si cada operación pinta cuando empieza y termina, podemos observar
  - si todos están vivos
  - si se reparten el trabajo equitativamente

dump

```
public void dump(java.lang.Object id,
         int nReaders,
         int nWriters)
```

Pintor.

Parameters:

`id` - el objeto que lo llama.

`nReaders` - numero de lectores en este momento.

`nWriters` - numero de escritores en este momento.

LogViewer

- ejemplo de uso

```
public class TsList {
   private LogViewer viewer = LogViewer.getInstance();


   public String remove(String clave) {
      ...
      try {
         ....
      } finally {
         viewer.dump(this, monitor.getNReadersIn(), monitor.getNWritersIn());
      }
   }
```

entrega

- package es.upm.dit.adsw.ej6
- todas las clases java que haya hecho
  - con los nombres que se han indicado