Serviço de Backup Distribuído

Este “read me” explica como correr o Serviço de Backup Distribuído.

Como correr

Em linux, abrir um terminal e, a partir do diretório da raiz do projeto, correr:

1) `sh createPeers.sh <versão> <nºpeers>`
   Por exemplo, `sh createPeers.sh 1.0  5`

Este é um script que limpa os chunks e ficheiros restaurados, compila, verifica se o serviço RMI está a correr e inicializa-o caso não esteja e cria a quantidade de peers passada como argumento com a versão do protocolo também indicada na chamada.

Em alternativa, pode-se fazer estes passos em separado:

```
sh clearFiles.sh // limpa os chunks e ficheiros restaurados
sh compile.sh // compila o projeto (também possível com o
comando javac)
sh rmi.sh // inicializa o serviço RMI em segundo plano
sh peer.sh <versão> <peer_id> // lançar um para cada peer que
se queira criar e idealmente em tabs separadas
```

As versões do protocolo podem ser verificadas mais abaixo.

2) A seguir, lançar a aplicação testApp no terminal inicial ou num novo no directório da raiz do projeto com:
`sh run.sh <peer_id> <serviço> <operando1> <operando2>`
em que o serviço pode ser BACKUP, RESTORE, RECLAIM, DELETE ou STATE. Operando1, no caso do serviço ser BACKUP, RESTORE ou DELETE é o

nome do ficheiro e no caso de ser RECLAIM indica o espaço de memória que se quer dedicada ao peer.

Por exemplo,

```
sh run.sh 1 BACKUP "files/lusiadas.txt" 3
sh run.sh 1 RESTORE "files/lusiadas.txt"
sh run.sh 1 DELETE "files/lusiadas.txt"
sh run.sh 1 STATE
sh run.sh 1 RECLAIM 63000
```

Versões do protocolo

Além do protocolo base da versão normal, implementamos mais quatro versões para os três melhoramentos:

Versão **1.0** : versão normal

Versão **1.1** : melhoramento do sub-protocolo de backup

Versão **1.2** : melhoramento do sub-protocolo de restore

Versão **1.3** : melhoramento do sub-protocolo de delete

Versão **2.0** : todos os anteriores melhoramentos em ação

Para compilar

Conforme já indicado, pode-se compilar o projeto com

```
javac -d bin -sourcepath src src/service/TestApp.java
src/service/Peer.java
```

ou com o script fornecido

```
sh compile.sh
```

Lançar o serviço RMI

Implementamos o serviço RMI que deve estar a correr quando se corre o Serviço Distribuído de Backup.

Para iniciar o serviço, deve-se correr

```
sh rmi.sh
```

Que chama o comando `rmiregistry &`

Criar peers

Pode-se criar cada peer através de

```
java -classpath bin service.Peer <versão> <peer_id> <peer_ap>
<mc:porto> <mdb:porto> <mdr:porto>
```

Em que, mc é o canal multicast, mdb é o canal de backup de dados e mdr é o canal de restore de dados.

Por exemplo, com os valores default para os canais e ponto de acesso:

```
java -classpath bin service.Peer 1.0 1 //localhost:1099/
224.0.0.0:8000 224.0.0.0:8001 224.0.0.0:8002
```

Correr a aplicação Test App

Pode-se lançar a Test App através de

```
java -classpath bin service.TestApp <PEER_AP> <SERVIÇO>
<OPERANDO1> <OPERANDO2>
```

em que o serviço pode ser BACKUP, RESTORE, RECLAIM, DELETE ou STATE. Operando1, no caso do serviço ser BACKUP, RESTORE ou DELETE é o nome do ficheiro e no caso de ser RECLAIM indica o espaço de memória que se quer dedicada ao peer.

Por exemplo,

```
java -classpath bin service.TestApp //localhost/1 BACKUP
"files/lusiadas.txt" 1
java -classpath bin service.TestApp //localhost/1 RESTORE
"files/lusiadas.txt"
java -classpath bin service.TestApp //localhost/1 DELETE
"files/lusiadas.txt"
```

```
java -classpath bin service.TestApp //localhost/1 STATE
java -classpath bin service.TestApp //localhost/1 RECLAIM
63000
```

Estrutura de diretórios

- src/ : directório com o código fonte
- bin/ : directório com os ficheiros das classes
- files/ : directório com os ficheiros a testar
- fileSystem/ : diretório onde será criado o sistema de ficheiros de cada peer com a estrutura exigida para o trabalho:
    - peer#
        - backup
            - 0x##
                - chk0
                - chk1
                - …
        - restored
            - Lusiadas.txt

Os nossos scripts

- compile.sh : script para compilar o projeto
- checkService.sh : script para verificar se o serviço rmi está a correr (se for necessário termina-lo usar `kill -9 pid`
- rmi.sh : script para lançar o serviço rmi em segundo plano
- clearFiles.sh : script para limpar sistema de ficheiros dos peers
- peer.sh : script para iniciar/criar cada peer
- run.sh : script para correr os sub-protocolos
- test.sh : script com uma bateria de testes (backup, state, restore, reclaim e delete) em que devemos confirmar o resultado