| EI1024/MT1024 “Programaci´on Concurrente y Paralela” 2022–23  Nombre y apellidos (1): Belén Ariño Bolinches  Nombre y apellidos (2):Aleix Grau Bas  Tiempo empleado para tareas en casa en formato *h:mm* (obligatorio): 1:00 | Entregable  para  Laboratorio  la06 g |
| --- | --- |

Tema 07. *Thread Pools* e Interfaces Gráficas en Java

Se dispone de un programa secuencial que **calcula el municipio de una provincia que tiene la mayor diferencia entre la temperatura máxima y la mínima**.

En este cálculo, se considera la previsión realizada por la AEMET (Agencia Española de Meteorología) en un día determinado de la semana actual. El objetivo de esta práctica es desarrollar una versión paralela que mejore sus prestaciones.

Dado que no se conocen los códigos de municipio válidos de una provincia, se realiza una primera ejecución secuencial en las que se verifican todos los posibles códigos, guardando los códigos de pueblos válidos en el fichero “*codPueblos XX.txt*”, donde “*XX*” se corresponde con el código de la provincia. Posteriormente, se realiza una nueva ejecución secuencial en la que únicamente se procesan los municipios que aparecen en el fichero “codPueblos XX.txt”.

Para reducir el coste de ejecución de la aplicación, la primera ejecución sólo se realiza si el correspondiente “codPueblos XX.txt” no existe.

Para facilitar la implementación, el método **obtenMayorDiferenciaDeFichero,** incluye la implementación de todas las versiones que recibe como parámetro la versión que se desea ejecutar.

El código secuencial del que se dispone es el siguiente:

import j a v a . i o . ∗ ;

import j a v a . ne t . ∗ ;

import j a v a . u t i l . ∗ ;

import j a v a . u t i l . c o n c u r r e n t . ∗ ;

import j a v a . u t i l . c o n c u r r e n t . atomic . A t omic In te ge r ;

c l a s s E jemploTempera turaProvincia *{*

public s t a t i c void main ( S t ri n g [ ] a r g s ) *{*

int numHebras , c odP r o vinci a , desp ;

S t ri n g nombreFichero = ”” ;

long t1 , t2 , t t [ ] ;

double t s , tp ;

PuebloMaximaMinima MaxMin ;

*// Comprobacion y e x t r a c c i o n de l o s argumen tos de en t r a d a .*

i f ( a r g s . l e n g t h != 3 ) *{*

System . out . p r i n t l n ( ”ERROR: numero de argumentos i n c o r r e c t o . ” ) ;

System . out . p r i n t l n ( ”Uso : j a v a programa *<*numHebras*> <*p r o vi n ci a *> <*de spl az amien t o*>*” ) ; System . e x i t ( −1 ) ;

*}*

try *{*

numHebras = I n t e g e r . p a r s e I n t ( a r g s [ 0 ] ) ;

c o dP r o vi n ci a = I n t e g e r . p a r s e I n t ( a r g s [ 1 ] ) ;

desp = I n t e g e r . p a r s e I n t ( a r g s [ 2 ] ) ;

*}* catch ( NumberFormatException ex ) *{*

numHebras = −1;

c o dP r o vi n ci a = −1;

desp = −1;

System . out . p r i n t l n ( ”ERROR: Numero de en t r ad a i n c o r r e c t o . ” ) ;

System . e x i t ( −1 ) ;

*}*

i f ( numHebras *<*= 0 ) *{*

System . out . p r i n t l n ( ”ERROR: El numero de Hebras debe s e r un numero e n t e r o mayor que 0 . ” ) ; System . e x i t ( −1 ) ;

*}*

i f ( ( c o dP r o vi n ci a *<* 1 ) *| |* ( c o dP r o vi n ci a *>* 5 0 ) ) *{*

System . out . p r i n t l n ( ”ERROR: El c o di g o de l a p r o vi n ci a debe s e r un numero e n t e r o ” + ” comprendido e n t r e 1 y 5 0. ” ) ;

System . e x i t ( −1 ) ;

*}*

i f ( ( desp *<* 0 ) *| |* ( desp *>*= 7 ) ) *{*

System . out . p r i n t l n ( ”ERROR: El d e s pl a z ami e n t o debe s e r un numero e n t e r o comprendido ” + ” e n t r e 0 y 6 . ” ) ;

System . e x i t ( −1 ) ;

*}*

i f ( c o dP r o vi n ci a *<* 1 0 ) *{*

nombreFichero = ” c odPuebl o s 0 ” + c o dP r o vi n ci a + ” . t x t ” ;

*}* e l s e *{*

nombreFichero = ” c odPuebl o s ” + c o dP r o vi n ci a + ” . t x t ” ;

*}*

System . out . p r i n t l n ( ) ;

System . out . p r i n t l n ( ” Obtiene e l puebl o de una p r o vi n ci a con mayor d i f e r e n c i a ” + ” de tempe r a tu r a . ” ) ;

*// S e l e c c i o n d e l d i a e l e g i d o*

S t ri n g f e c h a ;

Calendar c = Calendar . g e t I n s t a n c e ( ) ;

I n t e g e r dia , mes , anyo ;

c . add ( Calendar .DAY OF MONTH, desp ) ;

di a = c . g e t ( Calendar .DATE) ;

mes = c . g e t ( Calendar .MONTH) + 1 ;

anyo = c . g e t ( Calendar .YEAR) ;

f e c h a = S t ri n g . forma t ( ” %02d” , anyo ) + ”−” + S t ri n g . forma t ( ” %02d” , mes ) + ”−” + S t ri n g . forma t ( ” %02d” , di a ) ;

System . out . p r i n t l n ( f e c h a ) ;

*//*

*// Implemen tac ion s e c u e n c i a l s i n t em p o r i z a r .*

*//*

MaxMin = new PuebloMaximaMinima ( ) ;

F i l e f = new F i l e ( nombreFichero ) ;

i f ( f . e x i s t s ( ) ) *{*

ob tenM a y o rDi fe renci aDeFiche r o ( nombreFichero , fech a , c odP r o vin ci a , MaxMin, 0 , numHebras ) ; *}* e l s e *{*

o b t e nM a y o rDi f e r e n ci aAFi c h e r o S e c u e n ci al ( nombreFichero , fech a , c odP r o vin ci a , MaxMin ) ; *}*

System . out . p r i n t l n ( ” Pueblo : ” + MaxMin . damePueblo ( ) + ” , Maxima = ” + MaxMin . dameTemperaturaMaxima ( ) + ” , Minima = ” +

MaxMin . dameTemperaturaMinima ( ) ) ;

*//*

*// Implemen tac ion s e c u e n c i a l .*

*//*

System . out . p r i n t l n ( ) ;

3

t 1 = System . nanoTime ( ) ;

MaxMin = new PuebloMaximaMinima ( ) ;

ob tenM a y o rDi fe renci aDeFiche r o ( nombreFichero , fech a , c odP r o vin ci a , MaxMin, 0 , numHebras ) ; t 2 = System . nanoTime ( ) ;

t s = ( ( double ) ( t 2 − t 1 ) ) / 1. 0 e9 ;

System . out . p r i n t ( ” Implementacion s e c u e n c i a l . ” ) ; System . out . p r i n t l n ( ” Tiempo ( s ) : ” + t s ) ;

System . out . p r i n t l n ( ” Pueblo : ” + MaxMin . damePueblo ( ) + ” , Maxima = ” + MaxMin . dameTemperaturaMaxima ( ) + ” , Minima = ” +

MaxMin . dameTemperaturaMinima ( ) ) ;

*//*

*// Implemen tac ion p a r a l e l a : Ge s t i on Prop ia .*

*//*

System . out . p r i n t l n ( ) ;

t 1 = System . nanoTime ( ) ;

MaxMin = new PuebloMaximaMinima ( ) ;

ob tenM a y o rDi fe renci aDeFiche r o ( nombreFichero , fech a , c odP r o vin ci a , MaxMin, 1 , numHebras ) ; t 2 = System . nanoTime ( ) ;

tp = ( ( double ) ( t 2 − t 1 ) ) / 1. 0 e9 ;

System . out . p r i n t ( ” Implementacion p a r a l e l a : Ge s ti on P ropia . ” ) ;

System . out . p r i n t l n ( ” Tiempo ( s ) : ” + tp + ” , Inc remen to : ” + . . . ) ;

System . out . p r i n t l n ( ” Pueblo : ” + . . . + ” , Maxima = ” + . . . + ” , Minima = ” + . . . ) ;

*//*

*// Implemen tac ion p a r a l e l a : Thread Pool i sTe rm in a te d .*

*//*

*// . . .*

*//*

*// Implemen tac ion p a r a l e l a : Thread Pool con aw a i tTe rm in a t i on .*

*//*

*// . . .*

*//*

*// Implemen tac ion p a r a l e l a : Thread Pool con Fu ture .*

*//*

*// . . .*

*}*

*// −−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−* public s t a t i c void o b t e nM a y o rDi f e r e n ci aAFi c h e r o S e c u e n ci al ( S t ri n g nombreFichero , S t ri n g fech a , int c odP r o vin ci a , PuebloMaximaMinima MaxMin) *{*

Fil eW ri t e r f i c h e r o = nu l l ;

P ri n tW ri t e r pw = nu l l ;

*// V e r i f i c a t o d a s l o s c o d i g o s de p u e b l o s y e s c r i b e e l f i c h e r o*

try

*{*

*// Aper tura d e l f i c h e r o y c r e a c i o n de F il eW r i t e r para poder*

*// h ace r una l e c t u r a comoda ( d i s p o n e r d e l metodo re a dL ine ( ) ) .*

f i c h e r o = new Fil eW ri t e r ( nombreFichero ) ;

pw = new P ri n tW ri t e r ( f i c h e r o ) ;

for ( int i=c o dP r o vi n ci a ∗1 0 0 0; i *<*( c o dP r o vi n ci a +1)∗1000; i ++)*{*

i f ( P r oce s aPuebl o ( fech a , i , MaxMin, f a l s e ) == true ) *{*

pw. p r i n t l n ( i ) ;

*}*

*}*

4

*}* catch ( Excep ti on e ) *{*

e . p ri n t S t a c kT r a c e ( ) ;

*}* f i n a l l y *{*

try *{*

*// Nuevamente aprovechamos e l f i n a l l y para*

*// a s e g u r a r n o s que se c i e r r a e l f i c h e r o .*

i f ( nu l l != f i c h e r o )

f i c h e r o . c l o s e ( ) ;

*}* catch ( Excep ti on e2 ) *{*

e2 . p ri n t S t a c kT r a c e ( ) ;

*}*

*}*

*}*

*// −−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−*

public s t a t i c void ob tenM a y o rDi fe renci aDeFiche r o ( S t ri n g nombreFichero , S t ri n g fech a , int c odP r o vin ci a , PuebloMaximaMinima MaxMin, int opcion , int numHebras ) *{* F i l e f i c h e r o = nu l l ;

Fil eR e a d e r f r = nu l l ;

Bu f fe redRe ade r br = nu l l ;

*// Procesa e l f i c h e r o*

try

*{*

*// Aper tura d e l f i c h e r o y c r e a c i o n de B u f fe re dRe a de r para poder*

*// h ace r una l e c t u r a comoda ( d i s p o n e r d e l metodo re a dL ine ( ) ) .*

f i c h e r o = new F i l e ( nombreFichero ) ;

f r = new Fil eR e a d e r ( f i c h e r o ) ;

br = new Bu f fe redRe ade r ( f r ) ;

S t ri n g l i n e a ;

E x e c u t o r S e r vi c e e xec ;

switch ( opci on ) *{*

case 0 : *// Caso s e c u e n c i a l*

while ( ( l i n e a = br . r e adLi n e ( ) ) != nu l l ) *{*

int codPueblo = I n t e g e r . p a r s e I n t ( l i n e a ) ;

P r oce s aPuebl o ( fech a , codPueblo , MaxMin, f a l s e ) ;

*}*

break ;

case 1 : *// Ge s t i on Prop ia*

*// . . .*

break ;

case 2 : *// ThreadPools con i sTe rm in a te d*

*// . . .*

break ;

case 3 : *// ThreadPools con aw a i tTerm in a t i on*

*// . . .*

break ;

case 4 : *// ThreadPools + con Fu ture*

*// . . .*

break ;

de fau lt :

break ;

*}*

*}* catch ( Excep ti on e ) *{*

e . p ri n t S t a c kT r a c e ( ) ;

*}* f i n a l l y *{*

*// En e l f i n a l l y se c i e r r a e l f i c h e r o , para a s e g u r a r*

*// que e l c i e r r e se c om ple t a t a n t o s i t o d o va b i e n*

*// como s i a c t i v a una e x c e p c i o n .*

5

try*{*

i f ( nu l l != f r )*{*

f r . c l o s e ( ) ;

*}*

*}*catch ( Excep ti on e2 )*{*

e2 . p ri n t S t a c kT r a c e ( ) ;

*}*

*}*

*}*

*// −−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−*

public s t a t i c boolean P r oce s aPuebl o ( S t ri n g fech a , int codPueblo , PuebloMaximaMinima MaxMin, boolean imprime ) *{*

URL u r l ;

InputStream i s = nu l l ;

Bu f fe redRe ade r br ;

S t ri n g l i n e , p o bl a ci o n = new S t ri n g ( ) , p r o vi n ci a = new S t ri n g ( ) ; int s t a t e , num[]=new int [ 2 ] ;

boolean r e s = f a l s e ;

*// Procesam ien to de l a in f o rm ac i on XML a s o c i a d a a c o dPuebl o*

*// A c t u a l i z a c i o n de MaxMin de acuerdo a l o s v a l o r e s o b t e n i d o s*

try *{*

S t ri n g u r l S t r = ” h t t p s : / /www. aemet . e s /xml/ m u ni ci pi o s / l o c a l i d a d ” +

S t ri n g . forma t ( ” %05d” , codPueblo )+ ” . xml” ;

u r l = new URL( u r l S t r ) ;

i s = u r l . openStream ( ) ; *// t h r ow s an IOExcep t ion*

br = new Bu f fe redRe ade r (new InputStreamReader ( i s ) ) ;

i f ( imprime ) System . out . p r i n t l n ( u r l S t r ) ;

s t a t e = 0 ;

while ( ( ( l i n e = br . re a dLin e ( ) ) != nu l l ) && ( s t a t e *<* 6 ) ) *{*

*// System . ou t . p r i n t l n ( l i n e ) ;*

i f ( ( s t a t e == 0 ) && ( l i n e . c o n t ai n s ( ”nombre” ) ) ) *{*

p o bl a ci o n=l i n e . s p l i t ( ”*>*” ) [ 1 ] . s p l i t ( ”*<*” ) [ 0 ] . s p l i t ( ” / ” ) [ 0 ] ;

s t a t e ++;

*}* e l s e i f ( ( s t a t e == 1 ) && ( l i n e . c o n t ai n s ( ” p r o vi n ci a ” ) ) ) *{*

p r o vi n ci a=l i n e . s p l i t ( ”*>*” ) [ 1 ] . s p l i t ( ”*<*” ) [ 0 ] . s p l i t ( ” / ” ) [ 0 ] ;

s t a t e ++;

*}* e l s e i f ( ( s t a t e == 2 ) && ( l i n e . c o n t ai n s ( f e c h a ) ) ) *{*

s t a t e ++;

*}* e l s e i f ( ( s t a t e == 3 ) && ( l i n e . c o n t ai n s ( ” tempe r a tu r a ” ) ) ) *{*

s t a t e ++;

*}* e l s e i f ( ( s t a t e *>* 3 ) && ( ( l i n e . c o n t ai n s ( ”maxima” ) ) *| |* ( l i n e . c o n t ai n s ( ”minima” ) ) ) ) *{* num[ s t a t e −4] = I n t e g e r . p a r s e I n t ( l i n e . s p l i t ( ”*>*” ) [ 1 ] . s p l i t ( ”*<*” ) [ 0 ] ) ;

s t a t e ++;

*}*

*}*

*// System . ou t . p r i n t l n ( ” ( ” + c o dPuebl o + ”) ” + p o b l a c i o n + ”(” + p r o v i n c i a + ”) => ” + // ”(” + num [ 0 ] + ” , ” + num [ 1 ] + ” ) ” ) ;*

MaxMin . actualizaMaxMin ( p o bl a ci o n , codPueblo , num [ 0 ] , num [ 1 ] ) ;

r e s = true ;

*}* catch ( MalformedURLException mue ) *{*

mue . p ri n t S t a c kT r a c e ( ) ;

*}* catch ( IOException i o e ) *{*

*// i o e . p r i n t S t a c k T r a c e ( ) ;*

*}* f i n a l l y *{*

try *{*

i f ( i s != nu l l ) i s . c l o s e ( ) ;

*}* catch ( IOException i o e ) *{*

*// n o t h i n g t o s e e he re*

6

*}*

*}*

return r e s ;

*}*

*}*

*// ============================================================================* c l a s s PuebloMaximaMinima *{*

*// ============================================================================* S t ri n g p o bl a ci o n ;

int c odi g o , max, min ;

*// −−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−* public PuebloMaximaMinima ( ) *{*

p o bl a ci o n = nu l l ;

c o di g o = −1;

max = −1;

min = −1;

*}*

*// −−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−* public void actualizaMaxMin ( S t ri n g p o bl a ci o n , int c odi g o , int max, int min ) *{* i f ( ( th is . p o bl a ci o n == nu l l ) *| |* ( ( th is . max−th is . min ) *<* (max−min ) ) *| |*

( ( ( th is . max−th is . min ) == (max−min ) ) && ( th is . min *>* min ) ) *| |*

( ( ( th is . max−th is . min ) == (max−min ) ) && ( th is . min == min ) && ( th is . c o di g o *>* c o di g o ) ) ) *{*

*// ( ( ( t h i s . max−t h i s . min ) == (max−min ) ) && ( t h i s . max < max ) ) ) {*

th is . p o bl a ci o n = p o bl a ci o n ;

th is . c o di g o = c o di g o ;

th is . max = max ;

th is . min = min ;

*}*

*}*

*// −−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−* public S t ri n g damePueblo ( ) *{*

return th is . p o bl a ci o n + ” ( ” + th is . c o di g o + ” ) ” ;

*}*

*// −−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−* public int dameCodigo ( ) *{*

return th is . c o di g o ;

*}*

*// −−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−* public int dameTemperaturaMaxima ( ) *{*

return th is . max ;

*}*

*// −−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−* public int dameTemperaturaMinima ( ) *{*

return th is . min ;

*}*

*}*

7

| 1 |
| --- |

En este apartado, debes realizar una **gestión propia de hebras**, cuyo número será igual al **parámetro recibido** en la línea de comando.

Tanto la lectura del fichero de texto en paralelo como el acceso a la AEMET puede tener un coste muy diverso, por lo que se va a aplicar el método del Productor-Consumidor.

El programa principal irá leyendo el fichero l´ınea a l´ınea, y tras cada lectura, se generará una nueva tarea que deberá ser procesada.

La generación de nuevas tareas se realizará en paralelo al procesamiento de los códigos, y en paralelo a la lectura de las siguientes líneas. Es por ello que se aconseja que la **creación y arranque de las hebras se realice antes de la lectura del fichero**.

En este esquema, la **hebra productora** (el programa principal) inserta las tareas en una cola bloqueante a la que acceden las hebras consumidoras para tomar las tareas. Cuando el fichero se ha leído completamente, la hebra productora inserta tareas envenenadas para avisar a las hebras consumidoras.

Por su parte, la hebra consumidora extrae tareas de la cola bloqueante hasta que encuentre una tarea envenenada. La hebra consumidora debe procesar cada una de las tareas no envenenadas que extraiga de la cola.

Se propone que las tareas sean objetos de una clase con dos variables de instancia:

**esVeneno** y **codPueblo**

En el caso de una tarea normal, la primera variable valdrá falso y la segunda contendrá´a el código de pueblo leído. En el caso de una tarea envenenada, la primera variable valdrá cierto y la segunda contendrá´a el código de pueblo *−*1. Una posible opción ser´ıa la siguiente:

*// ============================================================================*

c l a s s TareaEnColaGestionPropia *{*

*// ============================================================================* boolean esVeneno ;

int codPueblo ;

*// −−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−* public TareaEnColaGestionPropia ( . . . ) *{*

*// . . .*

*}*

*// . . .*

*}*

En el caso que necesites modificar la clase PuebloMaximaMinima para que sea *thread-safe*, crea una nueva clase e incluye los cambios en la copia.

Comprueba que el nuevo código paralelo funciona correctamente comparando sus resultados con los de la versión secuencial.

Escribe a continuaci ́on la parte de tu c´odigo que realiza tal tarea: la definici´on de nuevas clases, la modificaci´on de la rutina obtenMayorDiferenciaDeFichero, y el c´odigo a incluir en el programa principal.

System.*out*.println("Implementación paralela");

System.*out*.println();

t1 = System.*nanoTime*();

MaxMin = new PuebloMaximaMinima();

*obtenMayorDiferenciaDeFichero* (nombreFichero, fecha, codProvincia, MaxMin, 1, numHebras);

t2 = System.*nanoTime*();

tp = ( ( double ) ( t2 - t1 ) ) / 1.0e9;

System.*out*.print( "Implementacion paralela: Gestion Propia. " );

System.*out*.println( " Tiempo(s): " + tp + " , Incremento: " +ts/tp );

System.*out*.println( " Pueblo: " + MaxMin.damePueblo() + " , Maxima = " +MaxMin.dameTemperaturaMaxima() + " , Minima = " +MaxMin.dameTemperaturaMinima() );

case 1: // Gestion Propia

// cola

ArrayBlockingQueue<TareaEnColaGestionPropia> queue= new ArrayBlockingQueue<>(numHebras);

//creo hebras y start

Vector<MiHebraParalela1> vHebras=new Vector<>(numHebras);

for(int i=0;i<numHebras ;i++) {

MiHebraParalela1 hebra = new MiHebraParalela1(queue, fecha,MaxMin);

vHebras.add(hebra);

}

for(int i=0;i<numHebras;i++){

vHebras.get(i).start();

}

// leo fich y añado taras a la cola

while( ( linea = br.readLine() ) != null ) {

int codPueblo = Integer.*parseInt*(linea);

TareaEnColaGestionPropia tarea=new TareaEnColaGestionPropia(codPueblo,false);

queue.put(tarea);

}

for(int i=0;i<numHebras;i++) {

queue.put(new TareaEnColaGestionPropia(-1, true));

}

break;

| ATENCION: Los ejercicios anteriores deben realizarse en casa. Los siguientes, en el aula. ´ |
| --- |

9

| 2 |
| --- |

Realiza una implementaci´on paralela con un *Thread Pool* del tipo newFixedThreadPool

*ExecutorService fixedPool = Executors.newFixedThreadPool(2);*

, en la que el programa principal incorpore los códigos de municipio leídos a un *ThreadPool*. En esta implementación, el programa principal debe esperar a que las

hebras terminen con una **espera activa**, es decir, con el m´etodo **isTerminated**.

Ten cuidado con la clase **PuebloMaximaMinima**, recuerda que el *Thread Pool* emplea internamente varias hebras. Si estas acceden a algún objeto compartido, éste debe ser *thread-safe* por lo que debes pensar que versión de este método debes utilizar.

Comprueba que el nuevo código paralelo funciona correctamente comparando sus resultados con los de la versión secuencial inicial.

Escribe a continuación la parte de tu código que realiza tal tarea:

* la definición de nuevas clases

public class TareaFixedThreadPool implements Runnable{

int codPueblo;

PuebloMaximaMinima maxMin;

String fecha;

public TareaFixedThreadPool(int codPueblo,PuebloMaximaMinima maxMin,String fecha) {

this.codPueblo=codPueblo;

this.maxMin=maxMin;

this.fecha=fecha;

}

@Override

public void run() {

EjemploTemperaturaProvincia.*ProcesaPueblo*(fecha, codPueblo, maxMin, false);

}

}

* la modificación de la rutina obtenMayorDiferenciaDeFichero

case 2: // ThreadPools con isTerminated

exec = Executors.*newFixedThreadPool* ( numHebras ) ;

//ThreadPoolExecutor pool=(ThreadPoolExecutor) exec;

//Encolamos las taras

while ((linea =br.readLine())!=null){

int codPueblo = Integer.*parseInt*(linea);

exec.submit(new TareaFixedThreadPool(codPueblo,MaxMin,fecha));

}

exec.shutdown();

while (!exec.isTerminated()){

}

break;

* el código a incluir en el programa principal.

System.*out*.println("Implementación paralela Thread Poll isTerminated");

System.*out*.println();

t1 = System.*nanoTime*();

MaxMin = new PuebloMaximaMinima();

*obtenMayorDiferenciaDeFichero* (nombreFichero, fecha, codProvincia, MaxMin, 2, numHebras);

t2 = System.*nanoTime*();

tp = ( ( double ) ( t2 - t1 ) ) / 1.0e9;

System.*out*.print( "Implementacion paralela: Thread Poll. " );

System.*out*.println( " Tiempo(s): " + tp + " , Incremento: " +ts/tp );

System.*out*.println( " Pueblo: " + MaxMin.damePueblo() + " , Maxima = " +MaxMin.dameTemperaturaMaxima() + " , Minima = " +MaxMin.dameTemperaturaMinima() );

| 3 |
| --- |

Realiza una implementación paralela con un *Thread Pool* del tipo newFixedThreadPool. Esta implementación es similar al ejercicio anterior, aunque en este caso, el programa principal debe esperar a que las hebras terminan sin una espera activa, es decir, con el método **awaitTermination.**

Comprueba que el nuevo código paralelo funciona correctamente comparando sus resultados con los de la versión secuencial inicial.

Escribe a continuaci ́on la parte de tu código que realiza tal tarea:

la definición de nuevas clases,

la modificación de la rutina obtenMayorDiferenciaDeFichero

case 3: // ThreadPools con awaitTermination

exec = Executors.*newFixedThreadPool* ( numHebras ) ;

//ThreadPoolExecutor pool=(ThreadPoolExecutor) exec;

//Encolamos las taras

while ((linea =br.readLine())!=null){

int codPueblo = Integer.*parseInt*(linea);

exec.submit(new TareaFixedThreadPool(codPueblo,MaxMin,fecha));

}

exec.shutdown();

try{

while (!exec.awaitTermination(2L,TimeUnit.*MILLISECONDS*)){

}

}catch (InterruptedException ex){

ex.printStackTrace();

}

break;

el código a incluir en el programa principal.

System.*out*.println("Implementación paralela Thread Poll awaitTermination");

System.*out*.println();

t1 = System.*nanoTime*();

MaxMin = new PuebloMaximaMinima();

*obtenMayorDiferenciaDeFichero* (nombreFichero, fecha, codProvincia, MaxMin, 3, numHebras);

t2 = System.*nanoTime*();

tp = ( ( double ) ( t2 - t1 ) ) / 1.0e9;

System.*out*.print( "Implementacion paralela: Thread Poll awaitTermination." );

System.*out*.println( " Tiempo(s): " + tp + " , Incremento: " +ts/tp );

System.*out*.println( " Pueblo: " + MaxMin.damePueblo() + " , Maxima = " +MaxMin.dameTemperaturaMaxima() + " , Minima = " +MaxMin.dameTemperaturaMinima() );

11

| 4 |
| --- |

Realiza una implementación paralela con un ***Thread Pool***en el que se maneje la interfaz **Callable**. En esta implementación, el programa principal incorpora los códigos de municipio leídos a un *ThreadPool*, y posteriormente, debe procesar el resultado obtenido en la ejecución de cada tarea del *Thread Pool*, que debe ser un objeto **PuebloMaximaMinima**.

Comprueba que el nuevo c´odigo paralelo funciona correctamente comparando sus resultados con los de la versi´on secuencial inicial.

Escribe a continuaci ́on la parte de tu código que realiza tal tarea:

* la definición de nuevas clases

public class ThreadPoolCallable implements Callable {

int codPueblo;

PuebloMaximaMinima maxMin;

String fecha;

public ThreadPoolCallable(PuebloMaximaMinima maxMin,int codPueblo,String fecha){

this.codPueblo=codPueblo;

this.maxMin=maxMin;

this.fecha=fecha;

}

@Override

public Object call() {

PuebloMaximaMinima nuevoMaxMin=new PuebloMaximaMinima();

EjemploTemperaturaProvincia.*ProcesaPueblo*(fecha, codPueblo, nuevoMaxMin, false);

return nuevoMaxMin;

}

}

* la modificación de la rutina obtenMayorDiferenciaDeFichero

case 4: // ThreadPools + con Future

Future < PuebloMaximaMinima > f;

exec = Executors.*newFixedThreadPool* ( numHebras ) ;

ArrayList < Future < PuebloMaximaMinima >> alf = new ArrayList <Future<PuebloMaximaMinima>>( );

while ((linea =br.readLine())!=null){

int codPueblo = Integer.*parseInt*(linea);

f = exec . submit ( new ThreadPoolCallable(MaxMin,codPueblo,fecha) ) ;

alf.add(f);

}

for ( int i = 0; i < numHebras ; i ++ ) {

try {

f = alf . get ( i ) ;

// Espera a que el resultado est é disponible .

PuebloMaximaMinima resultado = f.get() ;

MaxMin.actualizaMaxMin(resultado.damePueblo(),resultado.dameCodigo(), resultado.dameTemperaturaMaxima(), resultado.dameTemperaturaMinima());

} catch ( ExecutionException ex ) {

ex . printStackTrace () ;

} catch ( InterruptedException ex ) {

ex . printStackTrace () ;

}

}

break;

* y el código a incluir en el programa principal.

System.*out*.println("Implementación paralela Thread Poll awaitTermination");

System.*out*.println();

t1 = System.*nanoTime*();

MaxMin = new PuebloMaximaMinima();

*obtenMayorDiferenciaDeFichero* (nombreFichero, fecha, codProvincia, MaxMin, 3, numHebras);

t2 = System.*nanoTime*();

tp = ( ( double ) ( t2 - t1 ) ) / 1.0e9;

System.*out*.print( "Implementacion paralela: Thread Poll awaitTermination." );

System.*out*.println( " Tiempo(s): " + tp + " , Incremento: " +ts/tp );

System.*out*.println( " Pueblo: " + MaxMin.damePueblo() + " , Maxima = " +MaxMin.dameTemperaturaMaxima() + " , Minima = " +MaxMin.dameTemperaturaMinima() );

| 5 |
| --- |

Completa la siguiente tabla, seleccionando el código de provincia de Castellón (**12**) y eligiendo el desplazamiento para que se analice la previsión del día en el que realizais la pr ́ actica (0).

Municipio Temp. Máxima Temp. Ḿınima

|  |  |  |
| --- | --- | --- |

Obtén los resultados en tu ordenador local, tanto con 4 como con 8 hebras. Redondea los tiempos, dejando **tres decimales**, y los incrementos, dejando dos decimales.

4 hebras 8 hebras

| Implementaci´on | Tiempo | Incremento | Tiempo | Incremento |
| --- | --- | --- | --- | --- |
| Secuencial | 1.814 | — | 2.017 | — |
| Paralela con gestión propia de hebras | 0.476 | 3.804 | 0.305 | 6.602 |
| Paralela con newFixedThreadPool y espera activa con isTerminated | 0.451 | 4.017 | 0.279 | 7.212 |
| Paralela con newFixedThreadPool y espera con awaitTermination | 0.661 | 2.743 | 0.280 | 7.184 |
| Paralela con newFixedThreadPool e interfaz Callable | 0.456 | 9.977 | 0.265 | 7.603 |

Justifica los resultados obtenidos de prestaciones.

¿Qué versión de todas las paralelas ha sido la más fácil de escribir?

La más fácil de escribir en mi opinión ha sido la **ThreadPools con awaitTermination**

¿Estos cálculos están limitados por la CPU, la memoria central o la E/S? ¿Por qué?

Los cálculos estarán limitados claramente por la memoria, ya que dentro de cada hebra no tendremos que hacer una gran cantidad de cálculos pero tendremos que esperar para poder actualizar los argumentos de la clase PuebloMaxMin (Realentiza)

| 6 |
| --- |

Repite los c´alculos en patan, tanto con 16 como con 32 hebras, seleccionando el c´odigo de provincia de Castell´on (12) y eligiendo el desplazamiento para que se analice la previsi´on del d´ıa en el que realiz´ais la pr´actica (0). Para acortar el tiempo de ejecuci´on, copia tambi´en en patan el fichero “codPueblos XX.txt’.

Municipio Temp. M´axima Temp. M´ınima

|  |  |  |
| --- | --- | --- |

Redondea los tiempos dejando s´olo tres decimales y los incrementos dejando dos decimales.

16 hebras 32 hebras

| Implementaci´on | Tiempo | Incremento | Tiempo | Incremento |
| --- | --- | --- | --- | --- |
| Secuencial |  | — |  | — |
| Paralela con gesti´on propia de hebras |  |  |  |  |
| Paralela con newFixedThreadPool y espera activa con isTerminated |  |  |  |  |
| Paralela con newFixedThreadPool y espera con awaitTermination |  |  |  |  |
| Paralela con newFixedThreadPool e interfaz Callable |  |  |  |  |

Justifica los resultados obtenidos de prestaciones.

¿Qu´e versi´on de todas las paralelas ha sido la m´as f´acil de escribir?

¿Estos c´alculos est´an limitados por la CPU, la memoria central o la E/S? ¿Por qu´e?

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .