

Universitat de Lleida
Escola Politècnica Superior

Pràctica 1: Manfut
Sistemes concurrents i paral·lels

Francisco Javier Roig Gregorio 47433543E
Pere Antoni Rollon Baiges 39939768S

Professorat
Fernando Cores Prado
Marc Viladegut Abert

29 de novembre del 2020

Índex

Definició del disseny	1
Definició de les tasques concurrents	1
Definició de l'aplicació concurrent	1
Implementació concurrent	2
Implementació amb C	2
Implementació amb Java	2
Problemes i com s'han solventat	3
Versió de C	3
Versió de Java	3
Gràfics de temps entre les versions secuencial i concurrent	4
Versió de C	4
Versió de Java	5
Característiques del hardware	6
Conclusió	6

Índex de figures

1	Error de sobre-escriptura a pantalla	3
2	Gràfic de temps d'execució en C	4
3	Gràfic de temps d'execució en Java	5

Índex de taules

1	Speedup del codi amb C respecte el sequencial	4
2	Speedup del codi amb Java respecte el sequencial	5

Descripció de la aplicació concurrent

Definició de les tasques concurrents

En aquesta pràctica s'ha optat per la divisió entre tasques la funció `CalcularEquipOptim(param1,param2)`.

Per fer-ho s'ha optat per la descomposició de les dades d'entrada

En el disseny de la pràctica es poden trobar 2 tipus de granularitat:

- Granularitat fina: Calcular la puntuació d'un equip.
- Granularitat grossa: Calcular l'equip òptim per a un rang d'equips determinat.

Per el disseny s'ha utilitzat la granularitat grossa ja que correspon al rang principal dels equips, en aquest cas 2440H - 20000H depenent de cada fitxer de jugadors.

La càrrega de les tasques concurrents són homogènies i estancs ja que són subrangs del rang principal, això fa que les tasques no tinguin dependències entre si.

Definició de l'aplicació concurrent

Com s'ha esmentat anteriorment s'ha utilitzat la granularitat grossa i per cada tasca es destina un *thread* i un subrang del pricipal.

Cada tasca conté les dades privades següents:

- Buffer d'impressió de resultats
- L'equip inicial i l'equip final
- L'equip millor de cada tasca

Per l'aplicació s'ha utilitzat el patró *Single Proces Multiple Data (SPMD)* ja que les tasques son homogènies com s'ha mencionat anteriorment.

Per repartir les dades entre *threads* s'ha utilitzat un balanceig de càrrega estàtica i s'ha utilitzat un algoritme de modalitat d'assignació proporcional i d'aquesta forma s'obté un balanceig òptim sempre que el número de *threads* no superi a el numero de d'equips a analitzar.

Implementació concurrent

Implementació amb C

Per la versió concurrent amb C s'ha creat una estructura de dades *TeamInterval* on es guarden els parametres de la funció que executa cada thread. També s'ha fet servir un array bidimensional per guardar el rangs de cada equip per poder garantir un correcte balanceig de càrrega.

En el moment de la inicialització de cada *thread* es crea cada rang i també la estructura de dades *TeamInterval* de forma dinàmica on es guarden el paràmetres.

Per la distribució del valors dels *threads* per cada iteració es va dividint la càrrega del treball o rang general d'equips per el nombre de *threads* que es van generant de tal forma que els cada *TemaInterval* te una càrrega proporcional entre els processos.

Implementació amb Java

Per la versió concurrent de Java s'ha creat una classe estàtica *ManfutThread*, aniuada a la classe *Market*, estenent-la a la classe *thread* 'extends thread'.

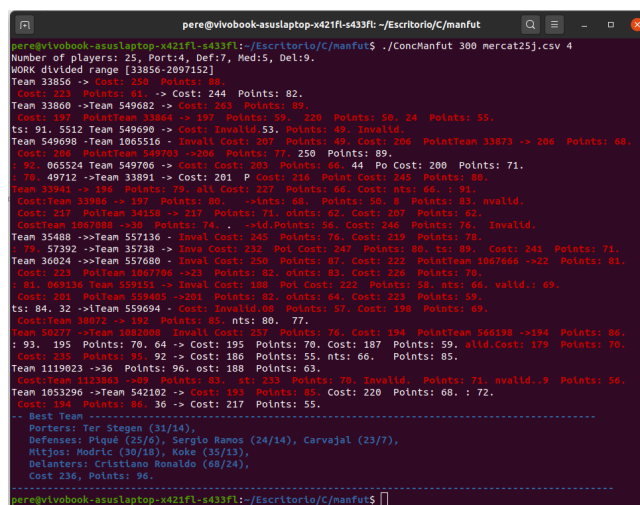
La implementació de la claase *ManfutThread* s'ha basat molt en la implemetació feta amb C i s'ha adaptat al llenguatge i per la distribució de càrrega també s'ha utilitzat la mateixa estratègia.

Problemes i com s'han solventat

Versió de C

Al finalitzar la pràctica de forma concurrent s'ha produït un error en mostrar la sortida de l'anàlisi dels equips. La sortida se sobreescrivia i interferia jutament amb els altres *threads* incloent que altres *threads* no poguessin mostrar-se a causa del fet que el buffer de sortida és compartit entre *threads*.

Per solucionar-ho s'ha afegit un buffer escriptura del buffer privada per cada *thread*. També s'han eliminat les escriptures que indicaven els equips erronis perquè solament mostres equips factibles amb el pressupost.



```
pere@vivobook-asuslaptop-x421fl-s433fl: ~/Escritorio/C/manfut
pere@vivobook-asuslaptop-x421fl-s433fl:~/Escritorio/C/manfut$ ./ConcManFut 300 mercat253.csv 4
Number of players: 25. Port:4, Def:7, Med:5, Del:9.
WDRK divided range [33856-2897152]
Team 33856 -> Cost: 196 Points: 81.
Cost: 223 Points: 61. -> Cost: 244 Points: 82.
Team 33860 ->Team 549682 -> Cost: 263 Points: 89.
Cost: 187 Points: 184 -> 197 Points: 79. 220 Points: 59. 24 Points: 55.
ts: 91. 5512 Team 549698 -> Cost: Invalid: 53. Points: 49. Invalid.
Team 549698 -Team 1065516 - Invalid Cost: 207 Points: 49. Cost: 208 PointTeam 33873 -> 208 Points: 68.
Cost: 188 Points: 9703 ->208 Points: 77. 250 Points: 89.
ts: 92. 865524 Team 549786 -> Cost: Cost: 203 Points: 66. 44 Po Cost: 280 Points: 71.
ts: 70. 49712 ->Team 33891 -> Cost: 201 P Cost: 216 PointCost: 245 Points: 80.
Team 33941 -> 196 Points: 79. all Cost: 227 Points: 66. Cost: nts: 66. : 91.
Cost:Team 33946 -> 197 Points: 80. ->nts: 68. Points: 50. 8 Points: 83. nvalid.
Cost: 117 PointTeam 34118 -> 117 Points: 71. nts: 62. Cost: 107 Points: 62.
Cost:Team 1067088 ->38 Points: 74. ->id:Points: 56. Cost: 246 Points: 76. Invalid.
Team 35488 ->Team 557136 - Invalid Cost: 245 Points: 76. Cost: 219 Points: 78.
ts: 79. 57392 ->Team 35738 -> Inva Cost: 232 Pol Cost: 247 Points: 80. ts: 89. Cost: 241 Points: 71.
Team 36024 ->Team 557680 - Invalid Cost: 250 Points: 67. Cost: 222 PointTeam 1807668 ->22 Points: 81.
Cost: 223 PointTeam 1807708 ->23 Points: 82. nts: 83. Cost: 226 Points: 78.
ts: 81. 869138 Team 559151 -> Invalid Cost: 188 Pol Cost: 222 Points: 58. nts: 66. valid.: 69.
Cost: 261 PointTeam 559205 ->261 Points: 82. nts: 64. Cost: 223 Points: 59.
ts: 84. 32 ->Team 559694 -> Cost: Invalid: 68 Points: 57. Cost: 198 Points: 69.
Cost:Team 36872 -> 192 Points: 85. nts: 80. 77.
Team 38277 ->Team 1082008 Invalid Cost: 257 Points: 76. Cost: 194 PointTeam 566198 ->194 Points: 86.
ts: 93. 195 Points: 70. 64 -> Cost: 195 Points: 70. Cost: 187 Points: 59. alid:Cost: 179 Points: 70.
Team 1119023 ->36 Points: 96. ost: 188 Points: 63.
Cost:Team 1123863 ->49 Points: 83. 80: 233 Points: 70. Invalid. Points: 71. nvalid.:9 Points: 56.
Team 1053290 ->Team 542102 -> Cost: 183 Points: 87 Cost: 220 Points: 68. : 72.
Cost: 186 Points: 96. 36 -> Cost: 217 Points: 55.
-----
-- Best Team -----
Porters: Ter Stegen (31/14),
Defenses: Piqué (25/6), Sergio Ramos (24/14), Carvajal (23/7),
Midfies: Rodri (30/13), Koke (35/13),
Delanters: Cristiano Ronaldo (68/24),
Cost 236, Points: 96.
-----
pere@vivobook-asuslaptop-x421fl-s433fl:~/Escritorio/C/manfut$
```

Figura 1: Error de sobre-escriptura a pantalla

Versió de Java

Per a la versió de java s'ha utilitzat la mateixa estratègia que a la versió de C tot i que no tenien un buffer compartit. Això s'ha fet perquè els *threads* utilitzen com a destinació d'escriptura la pantalla provocant la sobreescritura d'abans mencionada. I en aquest cas s'ha fet la escriptura amb `BufferedWriter`.

Gràfics de temps entre les versions secuencial i concurrent

Versió de C

Temps d'execució per thread en C

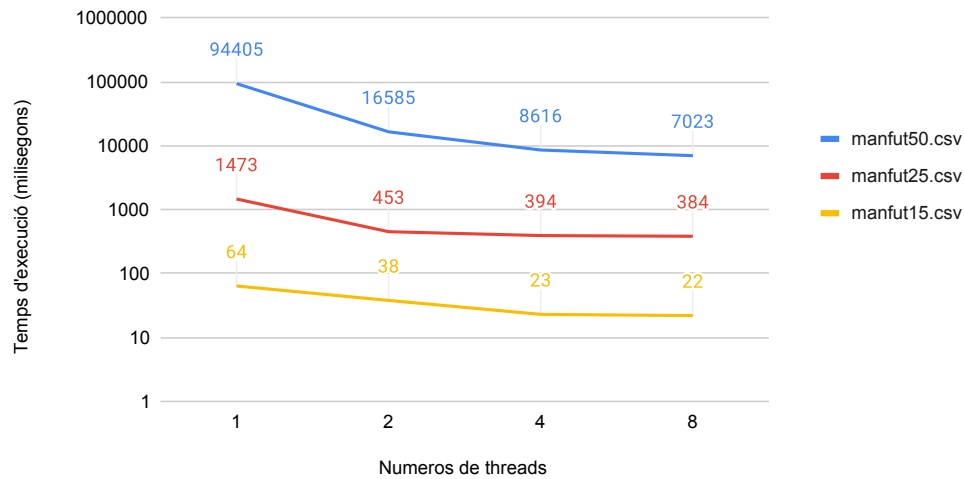


Figura 2: Gràfic de temps d'execució en C

	Threads		
	2	4	8
manfut50j.csv	82,43 %	90.87 %	92.56 %
manfut25j.csv	69.24 %	73.25 %	73.93 %
manfut15j.csv	40.62 %	64.06 %	65.62 %

Taula 1: Speedup del codi amb C respecte el secuencial

Com es pot veure a la figura2 i la taula 1 ha augmentat considerablement a partir de la seva execució amb 2 threads ja que respecte els demés threads no hi ha un canvi significatiu. Si considerem el speedup respecte la execució secuencial la millora es notable a partir de 8 threads ja que a partir d'aquí el temps d'execució s'estabilitzen fins a ser constants.

Versió de Java

Temps d'execució per thread en Java

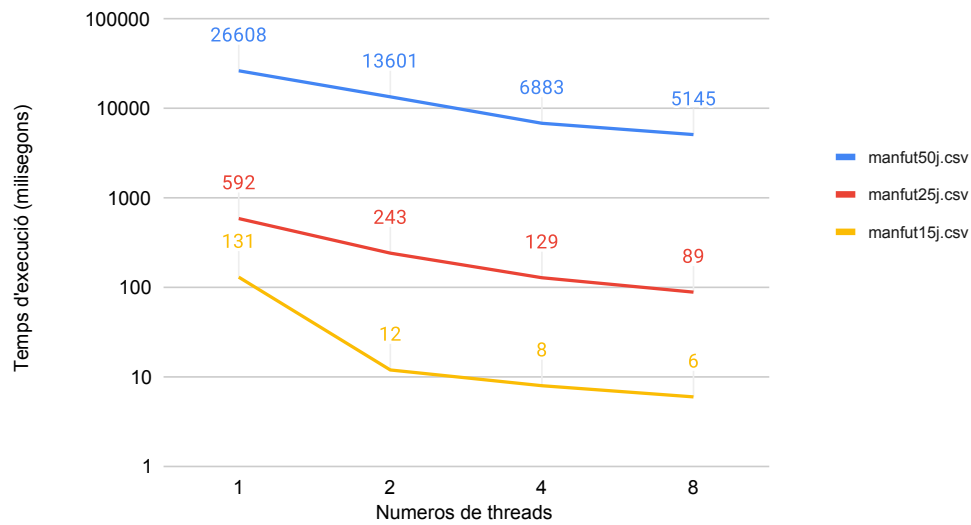


Figura 3: Gràfic de temps d'execució en Java

	Threads		
	2	4	8
manfut50j.csv	48.88 %	74.13 %	80.66 %
manfut25j.csv	58.95 %	78.20 %	84.96 %
manfut15j.csv	90.83 %	93.89 %	95.41 %

Taula 2: Speedup del codi amb Java respecte el sequencial

Igual que la versió de C el programa també mostra una millora de del temps d'execució. No obstant com que C es un llenguatge més pròxim al *hardware* la seva millora ha estat mes significativa. Encara que amb l'us de threads s'ha aconseguit una millora significativa.

Característiques del hardware

Arquitectura:	x86_64
modo(s) de operaci3n de las CPUs:	32-bit, 64-bit
Orden de los bytes:	Little Endian
Address sizes:	39 bits physical, 48 bits virtual
CPU(s):	8
Lista de la(s) CPU(s) en lnea:	0-7
Hilo(s) de procesamiento por ncleo:	2
ID de fabricante:	GenuineIntel
Familia de CPU:	6
Modelo:	142
Nombre del modelo:	Intel(R) Core(TM) i7-10510U CPU @ 1.80GHz

Conclusi3

Hem comprovat que aquesta aplicaci3 millora el temps d'execuci3 seqüencial, fins a un màxim nombre de threads. Aix3 és així perquè aquesta millora de rendiment està limitada per la fracci3 de codi seqüencial (Llei d'Amdahl).

Obtenim un speedup òptim utilitzant 2 threads per cada nucli del processador, ja que generalment per cada nucli hi ha dos fils.