

Universidad Nacional del Altiplano
Facultad de Ingeniería Estadística e Informática

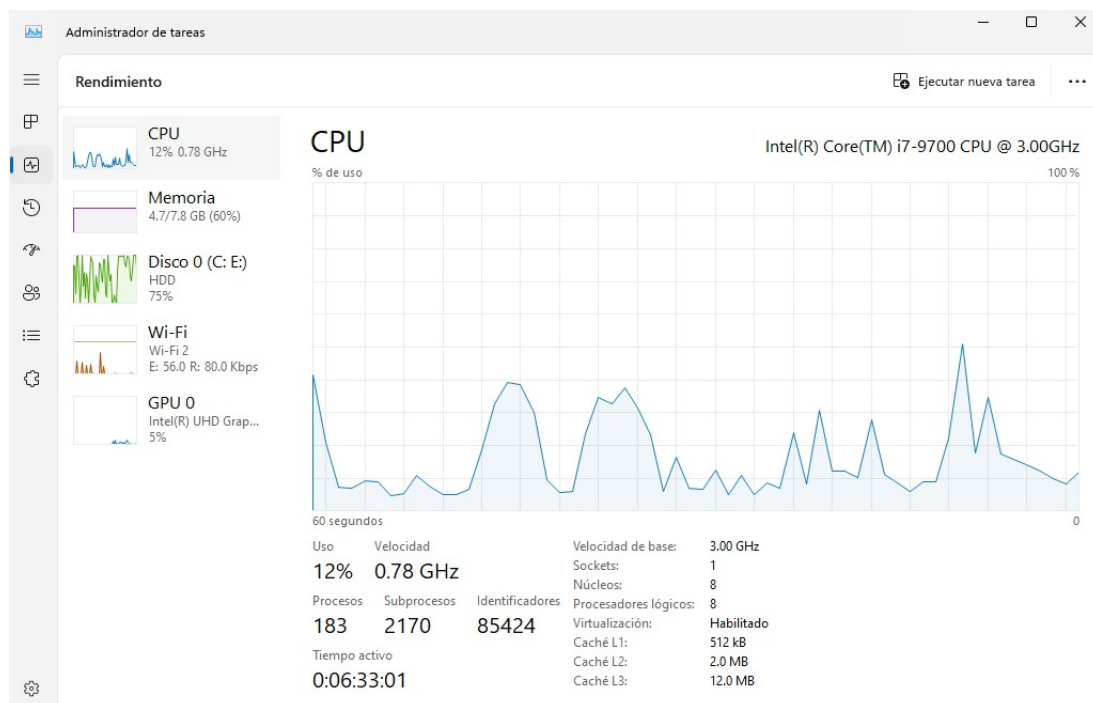
Docente: Fred Torres Cruz

Autor: Mary Luz Nina Palacios

Trabajo Encargado - N° 001

Realizar la impresión de los números pares entre 50000 y 100000, comparado entre un algoritmo paralelo y otro concurrente.

Características del equipo utilizado



Algoritmo Concurrente

a) Código en DEV C++

```
#include <iostream>

int main() {
    // Itera sobre los números entre 50000 y 100000
    for (int i = 50000; i <= 100000; i++) {
        // Verifica si el número es par
        if (i % 2 == 0) {
```

```

        // Imprime el número si es par
        std::cout << i << " ";
    }
}
return 0;
}

```

La ejecución secuencial en un solo hilo implica que las instrucciones del programa se ejecutan de manera lineal, una tras otra, en un único flujo de ejecución. Esto contrasta con el concepto de computación concurrente, que se refiere a la capacidad de un sistema para ejecutar múltiples tareas simultáneamente.

b) Salida

```

C:\Users\PC\Desktop\CONCU X + -
98960 98962 98964 98966 98968 98970 98972 98974 98976 98978 98980 98982 98984 98986 98988 98990 98992 98994 98996 98998
99000 99002 99004 99006 99008 99010 99012 99014 99016 99018 99020 99022 99024 99026 99028 99030 99032 99034 99036 99038
99040 99042 99044 99046 99048 99050 99052 99054 99056 99058 99060 99062 99064 99066 99068 99070 99072 99074 99076 99078
99080 99082 99084 99086 99088 99090 99092 99094 99096 99098 99100 99102 99104 99106 99108 99110 99112 99114 99116 99118
99120 99122 99124 99126 99128 99130 99132 99134 99136 99138 99140 99142 99144 99146 99148 99150 99152 99154 99156 99158
99160 99162 99164 99166 99168 99170 99172 99174 99176 99178 99180 99182 99184 99186 99188 99190 99192 99194 99196 99198
99200 99202 99204 99206 99208 99210 99212 99214 99216 99218 99220 99222 99224 99226 99228 99230 99232 99234 99236 99238
99240 99242 99244 99246 99248 99250 99252 99254 99256 99258 99260 99262 99264 99266 99268 99270 99272 99274 99276 99278
99280 99282 99284 99286 99288 99290 99292 99294 99296 99298 99300 99302 99304 99306 99308 99310 99312 99314 99316 99318
99320 99322 99324 99326 99328 99330 99332 99334 99336 99338 99340 99342 99344 99346 99348 99350 99352 99354 99356 99358
99360 99362 99364 99366 99368 99370 99372 99374 99376 99378 99380 99382 99384 99386 99388 99390 99392 99394 99396 99398
99400 99402 99404 99406 99408 99410 99412 99414 99416 99418 99420 99422 99424 99426 99428 99430 99432 99434 99436 99438
99440 99442 99444 99446 99448 99450 99452 99454 99456 99458 99460 99462 99464 99466 99468 99470 99472 99474 99476 99478
99480 99482 99484 99486 99488 99490 99492 99494 99496 99498 99500 99502 99504 99506 99508 99510 99512 99514 99516 99518
99520 99522 99524 99526 99528 99530 99532 99534 99536 99538 99540 99542 99544 99546 99548 99550 99552 99554 99556 99558
99560 99562 99564 99566 99568 99570 99572 99574 99576 99578 99580 99582 99584 99586 99588 99590 99592 99594 99596 99598
99600 99602 99604 99606 99608 99610 99612 99614 99616 99618 99620 99622 99624 99626 99628 99630 99632 99634 99636 99638
99640 99642 99644 99646 99648 99650 99652 99654 99656 99658 99660 99662 99664 99666 99668 99670 99672 99674 99676 99678
99680 99682 99684 99686 99688 99690 99692 99694 99696 99698 99700 99702 99704 99706 99708 99710 99712 99714 99716 99718
99720 99722 99724 99726 99728 99730 99732 99734 99736 99738 99740 99742 99744 99746 99748 99750 99752 99754 99756 99758
99760 99762 99764 99766 99768 99770 99772 99774 99776 99778 99780 99782 99784 99786 99788 99790 99792 99794 99796 99798
99800 99802 99804 99806 99808 99810 99812 99814 99816 99818 99820 99822 99824 99826 99828 99830 99832 99834 99836 99838
99840 99842 99844 99846 99848 99850 99852 99854 99856 99858 99860 99862 99864 99866 99868 99870 99872 99874 99876 99878
99880 99882 99884 99886 99888 99890 99892 99894 99896 99898 99900 99902 99904 99906 99908 99910 99912 99914 99916 99918
99920 99922 99924 99926 99928 99930 99932 99934 99936 99938 99940 99942 99944 99946 99948 99950 99952 99954 99956 99958
99960 99962 99964 99966 99968 99970 99972 99974 99976 99978 99980 99982 99984 99986 99988 99990 99992 99994 99996 99998
100000
-----
Process exited after 1.654 seconds with return value 0
Presione una tecla para continuar . . .

```

Algoritmo Paralelo

a) Código en DEV C++

```

#include <iostream>
#include <omp.h>

int main() {

    #pragma omp parallel for

```

```

    for (int i = 50000; i <= 100000; i++) {
        // Verifica si el número es par
        if (i % 2 == 0) {
            // Imprime el número si es par
            std::cout << i << " ";
        }
    }

    return 0;
}

```

El algoritmo paralelo aprovecha la capacidad de procesamiento concurrente del hardware mediante la biblioteca OpenMP (Open Multi-Processing). La directiva `#pragma omp parallel for` indica al compilador que debe distribuir las iteraciones del bucle `for` entre múltiples hilos de ejecución simultáneos.

Cada hilo procesa un subconjunto de las iteraciones del bucle, permitiendo que el trabajo se realice en paralelo en los diferentes núcleos de la CPU. De esta manera, se logra un rendimiento mejorado para tareas que pueden paralelizarse eficientemente, gracias al aprovechamiento del paralelismo a nivel de hilos.

b) Salida

```

C:\Users\PC\Desktop\PARALE x + -
67710 67712 67714 67716 67718 67720 67722 67724 67726 67728 67730 67732 67734 67736 67738 67740 67742 67744 67746 67748
67750 67752 67754 67756 67758 67760 67762 67764 67766 67768 67770 67772 67774 67776 67778 67780 67782 67784 67786 67788
67790 67792 67794 67796 67798 67800 67802 67804 67806 67808 67810 67812 67814 67816 67818 67820 67822 67824 67826 67828
67830 67832 67834 67836 67838 67840 67842 67844 67846 67848 67850 67852 67854 67856 67858 67860 67862 67864 67866 67868
67870 67872 67874 67876 67878 67880 67882 67884 67886 67888 67890 67892 67894 67896 67898 67900 67902 67904 67906 67908
67910 67912 67914 67916 67918 67920 67922 67924 67926 67928 67930 67932 67934 67936 67938 67940 67942 67944 67946 67948
67950 67952 67954 67956 67958 67960 67962 67964 67966 67968 67970 67972 67974 67976 67978 67980 67982 67984 67986 67988
67990 67992 67994 67996 67998 68000 68002 68004 68006 68008 68010 68012 68014 68016 68018 68020 68022 68024 68026 68028
68030 68032 68034 68036 68038 68040 68042 68044 68046 68048 68050 68052 68054 68056 68058 68060 68062 68064 68066 68068
68070 68072 68074 68076 68078 68080 68082 68084 68086 68088 68090 68092 68094 68096 68098 68100 68102 68104 68106 68108
68110 68112 68114 68116 68118 68120 68122 68124 68126 68128 68130 68132 68134 68136 68138 68140 68142 68144 68146 68148
68150 68152 68154 68156 68158 68160 68162 68164 68166 68168 68170 68172 68174 68176 68178 68180 68182 68184 68186 68188
68190 68192 68194 68196 68198 68200 68202 68204 68206 68208 68210 68212 68214 68216 68218 68220 68222 68224 68226 68228
68230 68232 68234 68236 68238 68240 68242 68244 68246 68248 68250 68252 68254 68256 68258 68260 68262 68264 68266 68268
68270 68272 68274 68276 68278 68280 68282 68284 68286 68288 68290 68292 68294 68296 68298 68300 68302 68304 68306 68308
68310 68312 68314 68316 68318 68320 68322 68324 68326 68328 68330 68332 68334 68336 68338 68340 68342 68344 68346 68348
68350 68352 68354 68356 68358 68360 68362 68364 68366 68368 68370 68372 68374 68376 68378 68380 68382 68384 68386 68388
68390 68392 68394 68396 68398 68400 68402 68404 68406 68408 68410 68412 68414 68416 68418 68420 68422 68424 68426 68428
68430 68432 68434 68436 68438 68440 68442 68444 68446 68448 68450 68452 68454 68456 68458 68460 68462 68464 68466 68468
68470 68472 68474 68476 68478 68480 68482 68484 68486 68488 68490 68492 68494 68496 68498 68500 68502 68504 68506 68508
68510 68512 68514 68516 68518 68520 68522 68524 68526 68528 68530 68532 68534 68536 68538 68540 68542 68544 68546 68548
68550 68552 68554 68556 68558 68560 68562 68564 68566 68568 68570 68572 68574 68576 68578 68580 68582 68584 68586 68588
68590 68592 68594 68596 68598 68600 68602 68604 68606 68608 68610 68612 68614 68616 68618 68620 68622 68624 68626 68628
68630 68632 68634 68636 68638 68640 68642 68644 68646 68648 68650 68652 68654 68656 68658 68660 68662 68664 68666 68668
68670 68672 68674 68676 68678 68680 68682 68684 68686 68688 68690 68692 68694 68696 68698 68700 68702 68704 68706 68708
68710 68712 68714 68716 68718 68720 68722 68724 68726 68728 68730 68732 68734 68736 68738 68740 68742 68744 68746 68748
68750
-----
Process exited after 1.905 seconds with return value 0
Presione una tecla para continuar . . .

```

Reportes de tiempos de ejecución

Cuadro 1: Tabla de tiempos de ejecución en segundos

N° DE PRUEBA	ALGORITMO CONCURRENTE	ALGORITMO PARALELO
1	1,683	1,942
2	1,555	1,853
3	1,61	1,973
4	1,497	1,82
5	1,787	1,886

Prueba estadística - t Student

Código en RStudio

```
# Datos de los tiempos de ejecución
tiempos_concurrente <- c(1.683, 1.555, 1.61, 1.497, 1.787)
tiempos_paralelo <- c(1.942, 1.853, 1.973, 1.82, 1.886)

# Crear un data frame con los datos
datos <- data.frame(concurrente = tiempos_concurrente, paralelo
= tiempos_paralelo)

# Imprimir los datos
print(datos)

# Realizar la prueba t de Student para muestras pareadas
resultado <- t.test(datos$concurrente, datos$paralelo, paired = TRUE)

# Imprimir el resultado
print(resultado)

# Interpretación del resultado
if (resultado$p.value < 0.05) {
  cat("Hay una diferencia significativa entre los tiempos medios de
ejecución de los dos algoritmos.\n")
}

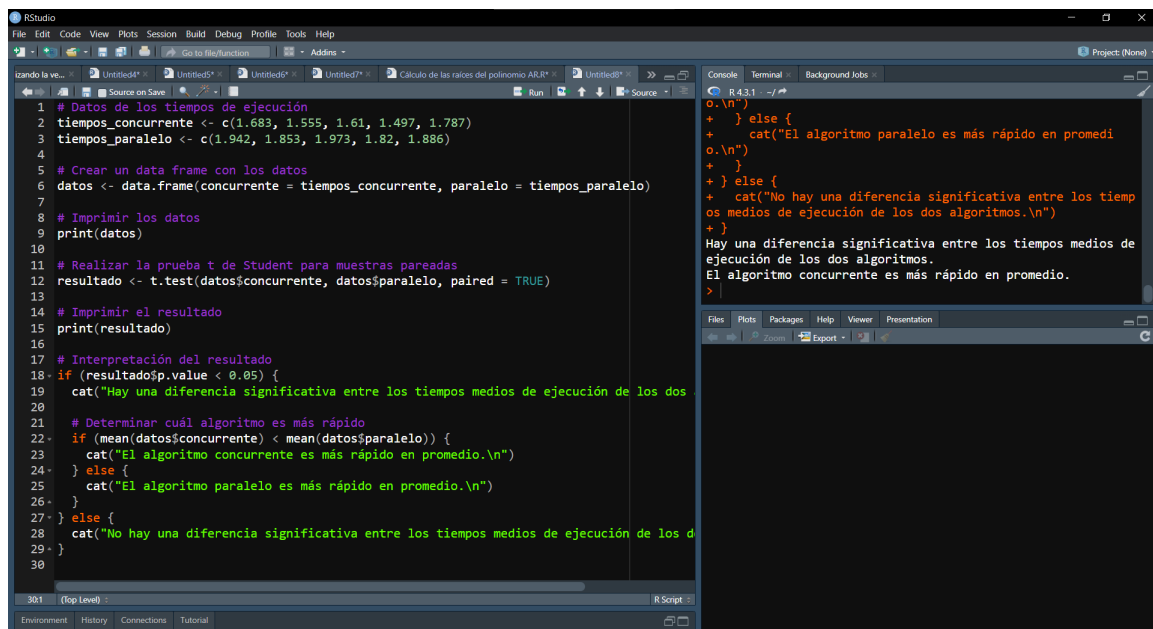
# Determinar cuál algoritmo es más rápido
if (mean(datos$concurrente) < mean(datos$paralelo)) {
  cat("El algoritmo concurrente es más rápido en promedio.\n")
} else {
  cat("El algoritmo paralelo es más rápido en promedio.\n")
}
} else {
```

```
cat("No hay una diferencia significativa entre los tiempos medios de
ejecución de los dos algoritmos.\n")
}
```

```
Paired t-test

data:  datos$concurrente and datos$paralelo
t = -5.8857, df = 4, p-value = 0.004166
alternative hypothesis: true mean difference is not equal to 0
95 percent confidence interval:
 -0.395012 -0.141788
sample estimates:
mean difference
 -0.2684
```

Interpretación



```
1 # Datos de los tiempos de ejecución
2 tiempos_concurrente <- c(1.683, 1.555, 1.61, 1.497, 1.787)
3 tiempos_paralelo <- c(1.942, 1.853, 1.973, 1.82, 1.886)
4
5 # Crear un data frame con los datos
6 datos <- data.frame(concurrente = tiempos_concurrente, paralelo = tiempos_paralelo)
7
8 # Imprimir los datos
9 print(datos)
10
11 # Realizar la prueba t de Student para muestras pareadas
12 resultado <- t.test(datos$concurrente, datos$paralelo, paired = TRUE)
13
14 # Imprimir el resultado
15 print(resultado)
16
17 # Interpretación del resultado
18 if (resultado$p.value < 0.05) {
19   cat("Hay una diferencia significativa entre los tiempos medios de ejecución de los dos
20   # Determinar cuál algoritmo es más rápido
21   if (mean(datos$concurrente) < mean(datos$paralelo)) {
22     cat("El algoritmo concurrente es más rápido en promedio.\n")
23   } else {
24     cat("El algoritmo paralelo es más rápido en promedio.\n")
25   }
26 } else {
27   cat("No hay una diferencia significativa entre los tiempos medios de ejecución de los d
28 }
29 }
30
```

```
o.\n")
+ } else {
+   cat("El algoritmo paralelo es más rápido en promedio.\n")
+ }
+ } else {
+   cat("No hay una diferencia significativa entre los tiempos medios de ejecución de los dos algoritmos.\n")
+ }
+ }
+ Hay una diferencia significativa entre los tiempos medios de ejecución de los dos algoritmos.
+ El algoritmo concurrente es más rápido en promedio.
+ >
```

En resumen, aunque el algoritmo paralelo generalmente ofrece ventajas de rendimiento, en ciertas circunstancias, el overhead introducido por la paralelización, la granularidad del problema, la contención de recursos, la escalabilidad limitada y una implementación ineficiente pueden anular estas ventajas, permitiendo que el algoritmo concurrente, más simple y sin sobrecarga de paralelización, sea más rápido, como se evidencia en los resultados de la prueba estadística.

GITHUB - LUZ052002

