UNIVERSITAT ROVIRA i VIRGILI

ARQUITECTURA DE COMPUTADORS

PRÀCTICA 3

Processadors MultiThread-Multicore

Comportament multiprocessadors

Raúl Martin Morales

2024/25

Índex:

1. Introducció

En aquesta pràctica tenim com a objectiu fer un estudi del comportament d'una aplicació multithread quan s'executa en maquines amb capacitat de processar varis threads paral·lelament. L'objectiu principal es paral·lelitzar un algoritme sequencial utilitzant, com es el càlcul de numeros primers, utilitzant una llibreria POSIX pthreads, per observar com varia el rendiment al executar-se en dos servidors diferents amb diferents característiques. El primer servidor teen, basat en processadors Intel Xeon i l'altre anomenat orca, basat en AMD Ryzen Threadripper.

Gràcies a aquest estudi podrem obtenir un anàlisi de les capacitats de hardware i com la configuració de threads afecta al temps d'execució i el speedup relatiu al rendiment de la versió inicial que es seqüencial.

Per a dur a terme aquest estudi, implementarem un programa que cerca números primers fins a un cert valor N determinat per el usuari com a primer paràmetre, dividint aquests N números, entre n_threads que s'indicarà com a segon paràmetre, i així dividir el treball del càlcul entre els diferents threads.

Farem comprovacions del temps d'execució variant aquest N i aquest n_threads, i així observar com varia el resultat segons augmentem el número de threads.

2. Especificacions de la práctica

·Objectiu principal de la pràctica:

Paral·lelitzar un algoritme de càlcul de números primers mitjançant la llibreria POSIX pthreads, i així poder realitzar l'execució en múltiples threads.

·Especificacions dels servidors:

Teen (Intel Xeon):

Porcessadors: 2 x Intel Xeon ES-2690 a 2.9GHz.

Nuclis: 8 cores amb 2 threads per core, per cada processador.

Memoria RAM: 32 GB.

TDP: 135 W + 135 W.

Orca (AMD Ryzen Threadripper):

Processador: AMD Ryzen Threadripper PRO 3995WX a 2.6 GHz.

Nuclis: 64 cores, amb 2 threads per core.

Memoria RAM: 128 GB.

TDP: 280 W.

·Configuració del programa:

Numero de threads: Es variarà aquest valor amb els següents: 2, 4, 8, 16, 32, 64, 128 y 256.

Quantitat de números: Es variarà aquesta mida màxima de cerca de números primers entre: 70M, 80M, 90M i 100M.

3. Decisions de disseny

· Primerament he hagut de importar la llibreria <pthread.h> per poder fer la creació i gestió de threads i temporalment he utilitzat la llibreria <time.h> per fer diferents mesuraments de temps.

· Primer es defineix N com a valor màxim dels números a comprovar i INI que estableix el punt on es comença a utilitzar threads.

· He definit una estructura anomenada DadesThread per agrupar la informació per a cada thread, amb possibilitat d'implementar més informació. On he definit el numero que indica el inic i fi del thread, i un contador intern de cada thread.

· S'ha hagut de definir un mutex anomenat afegirPrimer per protegir la secció critica, on assegura que múltiples threads no accedeixin a la vegada a l'array 'p' on es van emmagatzemant els valors primers trobats en la posició 'pp'.

· He aprofitat la funció de cercar primers que ja venia implementada, on he afegit que comenci a cercar a partir d'un número senar, ja que tots els primers son senars a partir del 2.

· Primerament es fa el càlcul de primers fins a INI i es farà el càlcul paral·lelitzat a partir d'aquest primer.

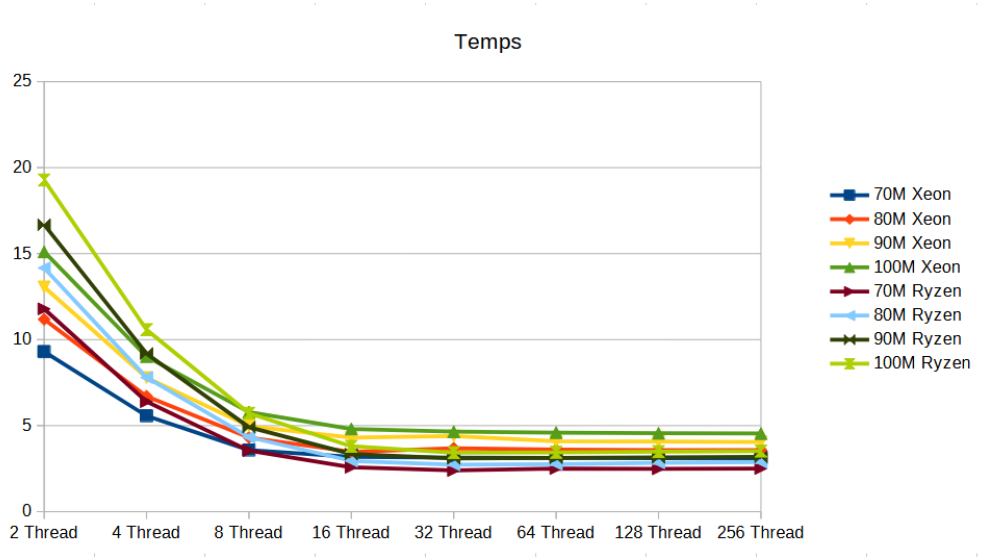
· Es fa una divisió del treball entre els threads que s'hagin escollit per paràmetre, es calcula la mida de números que haurà de gestionar cada thread i també els nombres sobrants que s'afegiran a cada thread de manera repartida.

· Finalment es fa la creació dels threads, on se li atribueix el rang de números a gestionar, on cada thread fa la seva cerca de nombres primers, i amb l'ajuda de pthread_join s'assegura que el programa principal no continuï fins que tots els threads acabin.

· Per últim es mostra el nombre de primers totals trobats i el darrer primer trobat.

4. Anàlisi dels Resultats

Temps d'execució (elapsed)



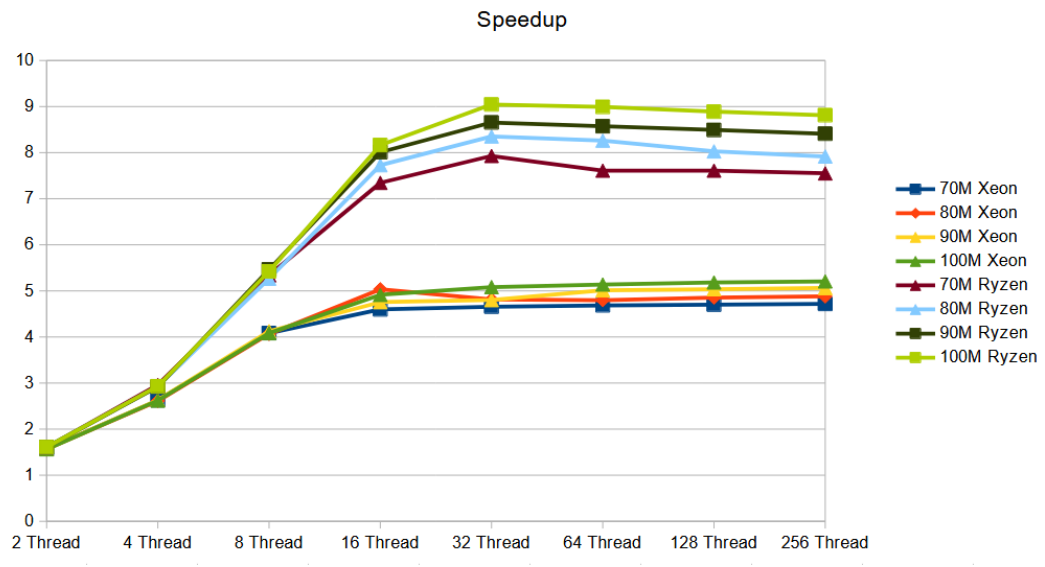
La gràfica mostra com varia el temps d'execució en funció del nombre de threads utilitzats i la mida de números totals, en els dos servidors.

El temps d'execució disminueix de manera significativa quan augmentem el nombre de threads, on més es pot veure aquesta disminució entre els 2-8 primers threads. A partir dels 16 threads, la reducció del temps d'execució és menor, on podem observar que el rendiment està arribant al seu punt de saturació.

El servidor de Intel Xeon mostra un gran rendiment fins als 16 thread, on el temps disminueix en major quantitat. A partir dels 32 threads, la millora es molt menor a causa de la saturació dels seus recursos ja que te 2 CPUs amb 8 cores i 2 threads per core, es a dir amb un màxim de 32 threads concurrents.

El servidor AMD Ryzen mostra una major escalabilitat en comparació al Xeon, gràcies a tenir un major nombre de cores, 64 cores i 2 threads per core que dona un màxim de 128 threads concurrents. Aleshores podem observar una millora del temps d'execució fins als 64 threads, segueix millorant en menor quantitat fins al 128 threads, on arriba al seu punt de saturació. Per fer el càlcul per una mida màxima major, Ryzen manté millors temps d'execució que Xeon, segons anem incrementant el nombre de threads, podem concloure que te millor capacitat de paral·lelització.

Speedup Relatiu



La gràfica mostra de la mateix manera que en l'anterior, com va variant el speedup segons augmentem el numero de threads i variant la mida màxima de números a calcular.

El speedup augmenta de manera significativa entre els 2 i 16 threads, amb una certa millora en els dos servidors, a partir dels 16 threads, el speedup arriba al punt de saturació i s'estabilitza en els valors on han arribat als 16 threads. Podem observar com clarament el speedup de la maquina AMD Ryzen es major a comparació de la Intel Xeon indiferentment dels valors maxims a calcular.

A la màquina de Intel Xeon s'aconsegueix un maxim de 5,2 aproximadament. La millora s'atura als 16-32 threads, amb una escalabilitat bastant limitada. Això ens pot indicar que la màquina de Intel té una arquitectura que aprofita menys la paral·lelització i per això arriba ràpid al seu punt de saturació.

Per altra banda, el speedup màxim de la màquina de AMD Ryzen es molt superior al de Intel, arriben a un màxim de 9 aproximadament. La millora que podem observar es bastant visual fins als 32 threads on varia el speedup entre els 8 i els 9, i a partir d'aquí es comença a mantenir constant. Amb això podem concloure que la màquina de AMD Ryzen té una major capacitat de paral·lelisme, gràcies a que té un major nombre de cores.

Speedup Teòric

El speedup teòric el calculem amb la Llei d'Amdahl, on s'assumeix que la maquina utilitza els recurs de manera perfecta, no es te en compte els factors com els que hi ha als speedup pràctica, com les càrregues de comunicació, latències de memòria o falta de eficiència amb programes que implementen paral·lelisme.

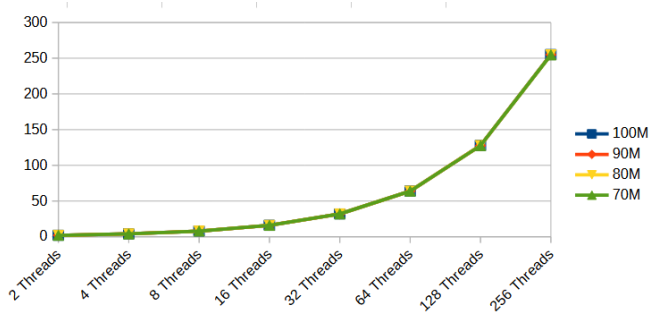
$$\text{Speedup}(N) = \frac{1}{(1-P)+\frac{P}{N}}$$

Serial part of job = 1 (100%) - Parallel part

Parallel part is divided up by N workers

On P es la fracció del treball que es fa amb paral·lelització, es a dir la part que no es seqüencial, i N representaria el nombre de threads que s'utilitzen.

	100M	90M	80M	70M
2 Threads	1,9999700004	1,9999660006	1,9999620007	1,9999580009
4 Threads	3,9998200081	3,9997960104	3,999772013	3,9997480159
8 Threads	7,9991600882	7,9990481133	7,9989361415	7,9988241728
16 Threads	15,99640081	15,99592104	15,995441299	15,994961587
32 Threads	31,985126916	31,983144883	31,981163095	31,979181553
64 Threads	63,9395771	63,931529332	63,92348359	63,915439873
128 Threads	127,75662363	127,72424336	127,69187949	127,65953203
256 Threads	255,02453117	254,89503004	254,76566038	254,63642196



El speedup com tal mesura quant més ràpid s'executa un programa quan s'utilitzen múltiples threads en comparació amb l'execució seqüencial

Com en aquesta pràctica utilitzem quantitats de N molt grans(70M-100M) i la part seqüencial es molt petita(1500), es quasi insignificant per al programa en relació al paral·lelitzat.

Quan el nombre de threads augmenta, el speedup també creix, i s'apropa a un valor molt semblant al nombre de threads, degut aquesta petita part seqüencial. Com a resultat es pot concloure que el programa pot aprofitar el màxim dels threads que es fan servir, per fet de que la part seqüencial es molt petita.

5. Manual de la pràctica

A continuació es mostraran els passos per realitzar les proves:

1) Accedí al servidor front-end 'zoo':
   milax@dxxx:~$ ssh loginURV@zoo.urv.cat

2) Copiem les dades del programa en C desde milax a 'zoo'
   milax@dxxx:~$ scp pim2ac.c loginURV@zoo.urv.cat:

3) Compilem el codi per a procesadors compatibles X86
   loginURV@zoo:~$ cc -O prim2ac.c -o prim2ac -lm –lpthread

4) Executar la maquina 'teen' de Intel Xeon amb la funció time per observar el temps d'execució com a 'elapsed'
   loginURV@zoo:~$ srun -p teen -c 32 time ./prim2ac

5) Executar la maquina 'orca' de AMD Ryzen amb la funció time per observar el temps d'execució com a 'elapsed'
   loginURV@zoo:~$ srun -p orca -c 128 time ./prim2ac

6) Exemple d'execució:
   Srun –p <nom_particio> -c <n_cores> time ./prim2ac <N> <n_threads>

```
          @zoo:~$ srun -p orca -c 128 time ./prim2ac 100000000 8
Tots els primers fins a 100000000 amb 8 threads
Hi ha 5761454 primers
Darrer primer trobat 99999989
34.34user 2.01system 0:05.86elapsed 619%CPU (0avgtext+0avgdata 26220maxresident)
k
0inputs+0outputs (0major+602minor)pagefaults 0swaps
```

7) Altres comdandes:

   loginURV@zoo:~$ sinfo

   Per comprovar l'estat de les particions disponibles

   loginURV@zoo:~$ squeue

   Per veure l'estat dels treballs en execució o en cua

   loginURV@zoo:~$ scancel 'JOBID'

   Per cancel·lar un treball específic, amb el seu JOBID

6. Conclusions

Amb la realització d'aquesta pràctica hem pogut analitzar com el disseny de l'arquitectura del hardware afecten al rendiment de l'execució paral·lela de threads d'un programa. Amb els resultats obtinguts podem observar que la implementació multithread, quan s'assigna a servidors amb arquitectures de processadors més avançats, podem generar grans millores en el temps d'execució i la seva escalabilitat.

En el cas del primer servidor, amb els dos processadors de Intel Xeon(teen), vaig observar que el rendiment augmenta bastant fins a un màxim de 16-32 threads. Aquesta limitació es pot dir que es per el seu disseny físic, ja que el seu numero de cores determina un punt de saturació en les proves, aquest disseny redueix la capacitat d'escalabilitat a partir d'aquests threads. A partir d'això podem determinar la que la seva arquitectura esta menys optimitzada per a la paral·lelització que la de AMD Ryzen.

Perquè per altra banda, tenim el servidor de AMD Ryzen(orca) on destaca la seva capacitat d'escalabilitat a comparació de la de Intel, gràcies a que te un major número de corees, té 128, i això li permet processar una quantitat major de dades simultàniament, i mantindre un major rendiment fins als 128 threads.

Finalment, s'ha vist que la paral·lelització no garanteix un augment lineal del rendiment, ja que existeixen factors de saturació i límits segons l'arquitectura.