PRÀCTICA 3:

Arquitectura de Computadors

Miguel Lopes i Oriol Poblet
Curs 2023/2024

ÍNDEX

1. ESTUDI DE LES MÀQUINES

Primer de tot, volem fer un estudi sobre els servidors que utilitzarem per a la pràctica, per a que ens ajudin a contrastar la informació resultant de les nostres màquines.

CARACTERÍSTIQUES

SERVIDOR TEEN	*SERVIDOR ORCA*
2 processadors Intel Xeon E5-2690	1 processador AMD Ryzen Threadripper PRO 3995WX
2,9GHz de velocitat	2,7GHz de velocitat
8 cores (x2) amb 2 fils per core (=16+16)	64 cores amb 2 fils per core (=128)
32GB de memòria RAM (x2)	128GB de memòria RAM
135W TPD (x2)	280W TDP

Degut a les característiques observades, podem deduir que la presencia de dos processadors al servidor *teen* pot ajudar-nos a millorar el rendiment de forma eficient. En canvi, el servidor *orca*, ens pot ajudar a executar més conjunts de dades degut a la seva memòria RAM i a paral·lelitzar més fàcilment amb el seu nombre de nuclis (que a més a més compta amb la capacitat d'executar 2 fils per nucli).

En resum, tant el servidor *teen* com l'*orca* son servidors que poden gestionar tasques de càlcul complexes. No obstant això, l'*orca* ofereix més nuclis i memòria RAM mentres que *teen* ens ofereix una opció més econòmica per a tasques que no necessiten tants recursos. Aquestes característiques, converteixen al servidor *teen* en un servidor que ens permet executar tasques de càlcul en dos processadors que consumeixen menys mentre que el servidor *orca* ens ajudarà a executar més fàcilment grans conjunts de dades de forma paral·lela.

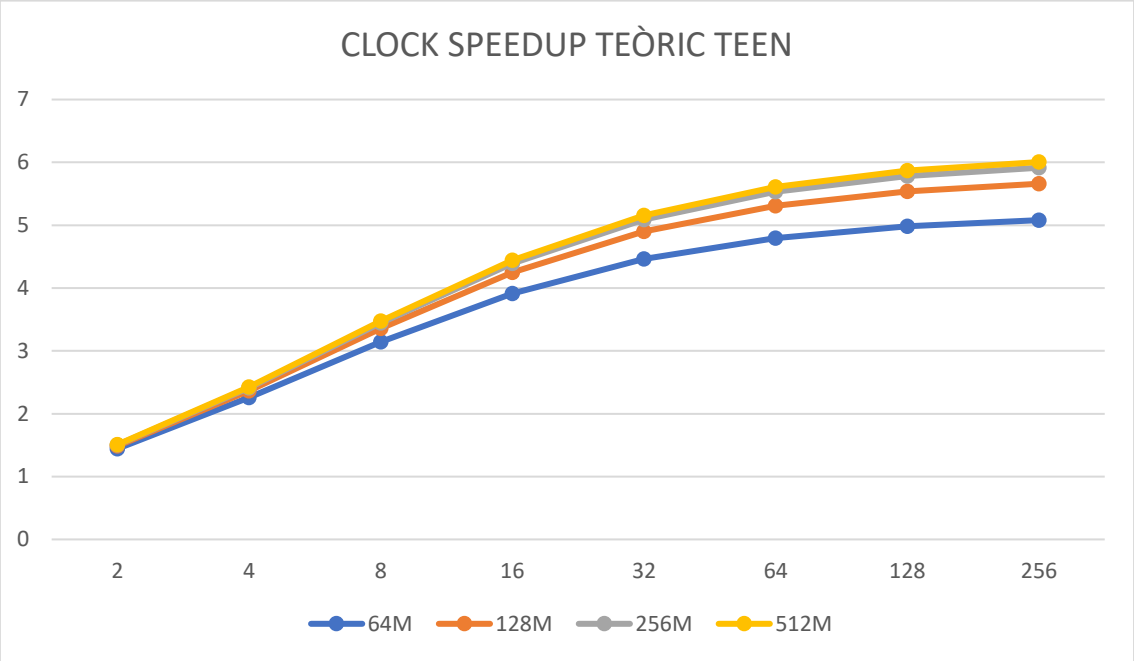
2. ANÀLISI SPEEDUP TEÒRIC

Per començar, estudiem l'*speedup* teòric del fitxer original que ens donen executat sobre els servidors *teen* i *orca* utilitzant la funció *clock()* per extreure els temps de *quicksort*, *merge* i *elapsed*. Posteriorment, usarem la llei d'*Amdahl* per a calcular l'*speedup* veient el temps que es tarda en executar les fraccions NO paral·lelitzables (*elapsed time – quicksort time – merge time*) i quan de temps tarden en executar-se les fraccions paral·lelitzables (*quicksort time + merge time*).

La formula utilitzada per calcular l'*speedup* teòric per tant serà:

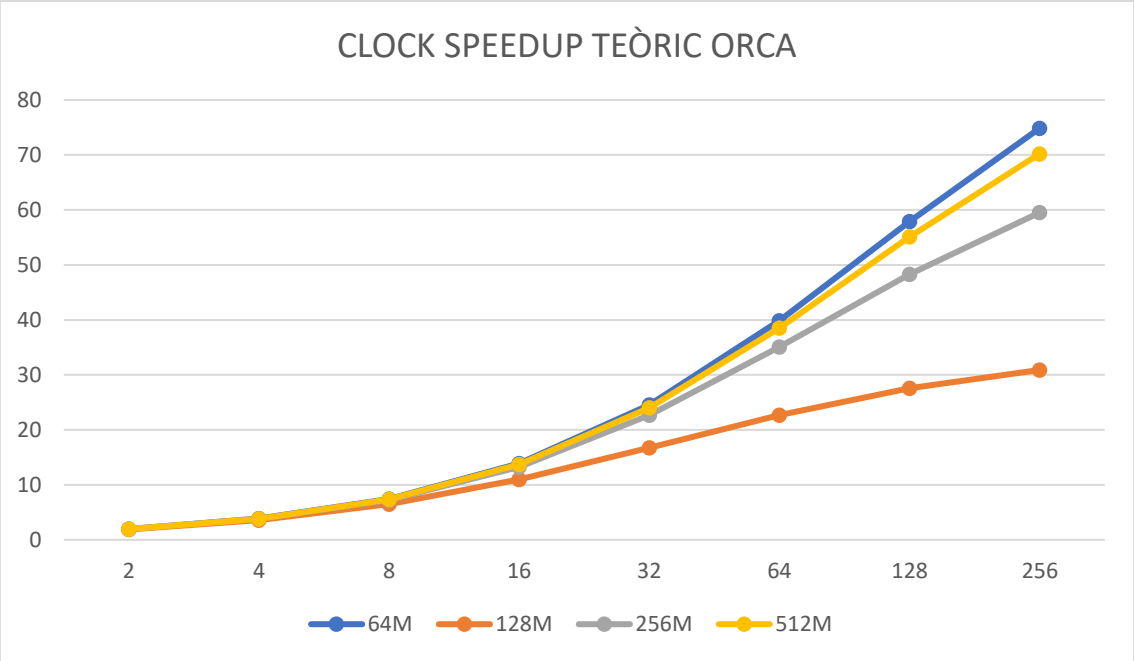
$$\frac{\text{Temps Total Paral·lelitzable}}{\text{Temps No Paral·lelitzable + (Temps Total Paral·lelitzable / nThreads)}}$$

Fent els càlculs segons 64, 128, 256 i 512 milions de dades en el servidor *teen*:



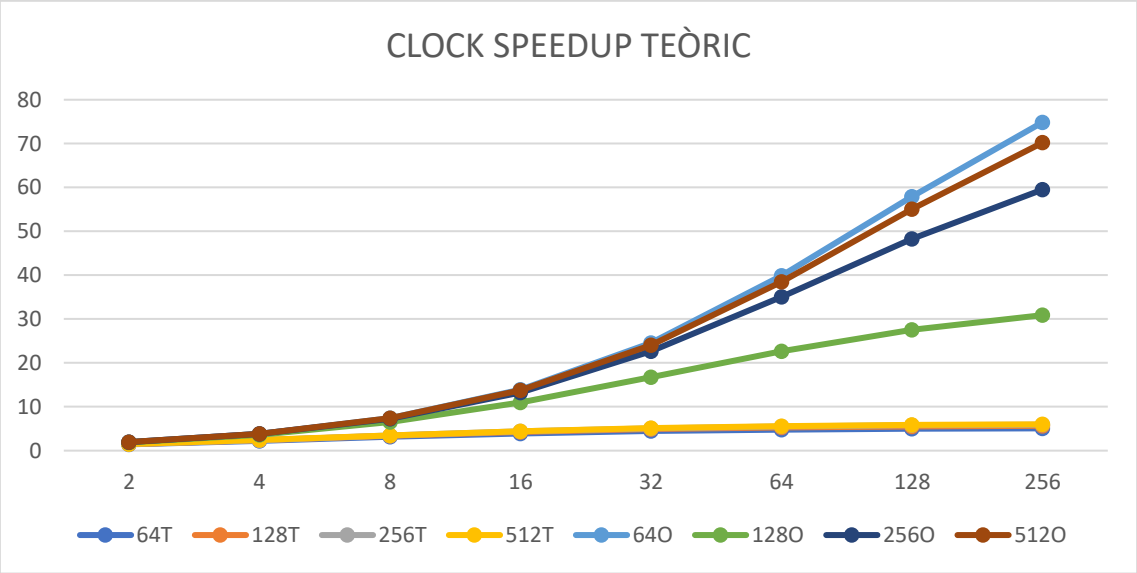
	64M	128M	256M	512M
2	1,44	1,49	1,5	1,51
4	2,26	2,37	2,41	2,42
8	3,15	3,36	3,45	3,48
16	3,92	4,25	4,39	4,44
32	4,46	4,9	5,09	5,16
64	4,8	5,31	5,53	5,61
128	4,98	5,54	5,78	5,87
256	5,08	5,66	5,91	6,01

I ara, fem el mateix però executant-lo en el servidor *orca*:



	64M	128M	256M	512M
2	1,96	1,89	1,95	1,96
4	3,85	3,59	3,8	3,84
8	7,44	6,52	7,25	7,39
16	13,9	10,99	13,26	13,73
32	24,57	16,75	22,65	24,05
64	39,88	22,68	35,06	38,52
128	57,92	27,56	48,29	55,1
256	74,86	30,89	59,52	70,21

Ajuntem els resultats dels dos servidors i observem:



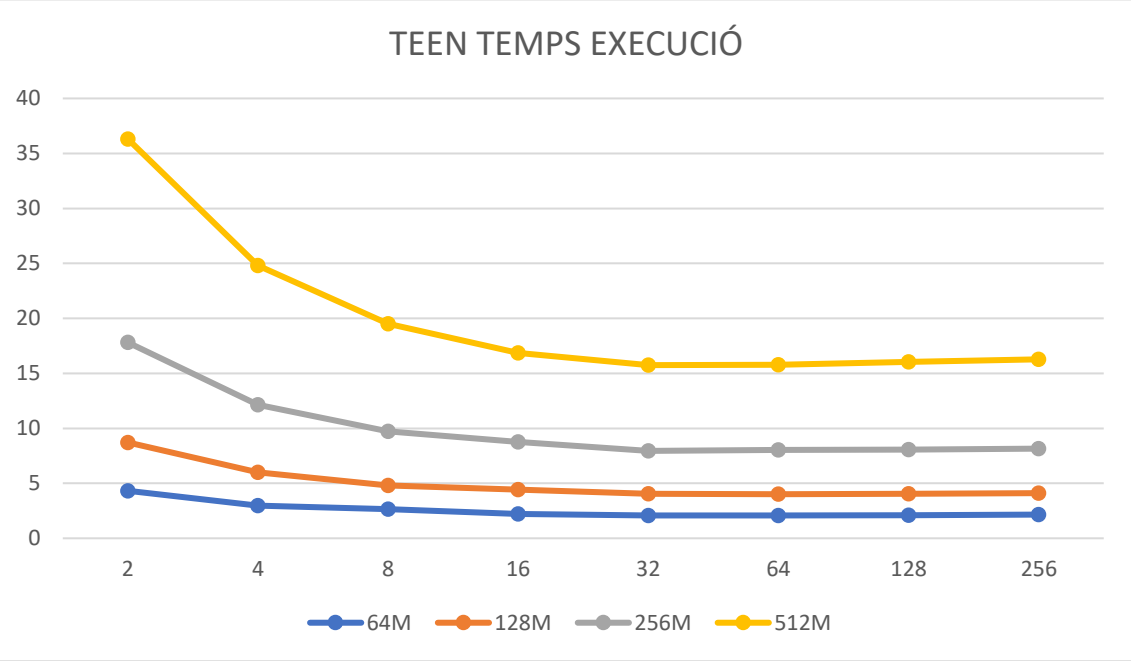
[1]

[1] “64T” “128T” “256T” “512T” “64O” “128O” “256O” “512O” indica els milions de dades segons el servidor, T = TEEN i O = Orca

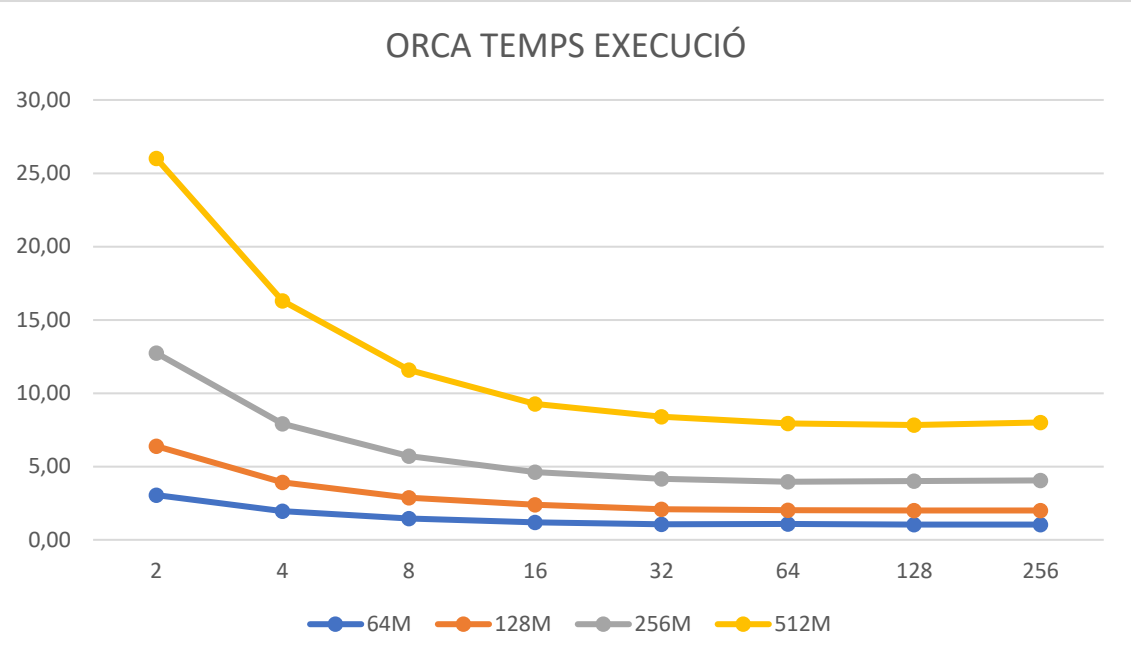
Com podem observar, el servidor *Orca* ens dona millors resultats alhora de executar dades degut a la seva eficiència al executar grans volums de dades. Això es degut a que la majoria de temps es passa fent parts paral·lelitzables en els seus nuclis i per conseqüència té més potencial de millora. En canvi, el servidor *teen* compta amb menys nuclis i una potencia inferior que el servidor *orca*, i com podem observar el seu *speedup* no millora gaire i obté uns valors bastant similars.

3. ANÀLISI TEMPS EXECUCIÓ

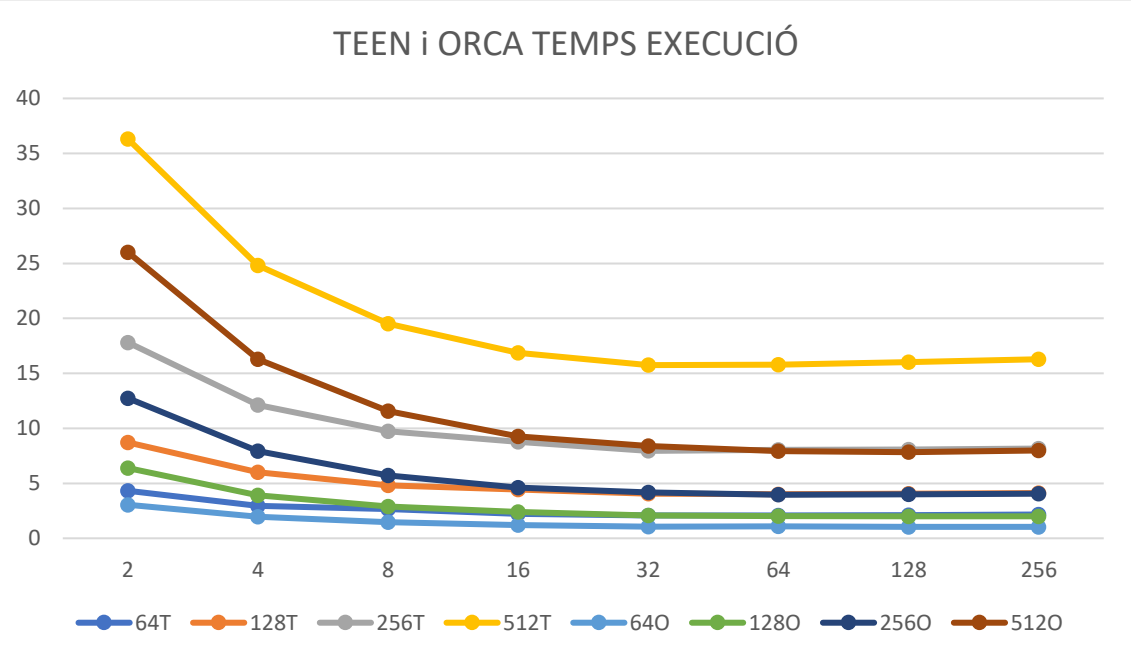
Posteriorment a calcular l'*speedup* teòric, modifiquem l'script i aconseguim obtindre resultats dels temps d'execució del codi en els servidors *teen* i *orca* enviant 64, 128, 256 o bé 512 milions de dades.



	64M	128M	256M	512M
2	4,32	8,72	17,81	36,31
4	2,96	6,01	12,13	24,8
8	2,66	4,81	9,74	19,5
16	2,22	4,44	8,76	16,86
32	2,08	4,06	7,95	15,75
64	2,08	4,01	8,03	15,78
128	2,11	4,06	8,06	16,03
256	2,15	4,11	8,15	16,27



	64M	128M	256M	512M
2	3,05	6,39	12,74	26,02
4	1,95	3,92	7,92	16,29
8	1,46	2,88	5,72	11,58
16	1,20	2,4	4,62	9,26
32	1,07	2,09	4,16	8,39
64	1,09	2,03	3,96	7,93
128	1,04	2	4	7,83
256	1,08	2	4,06	8

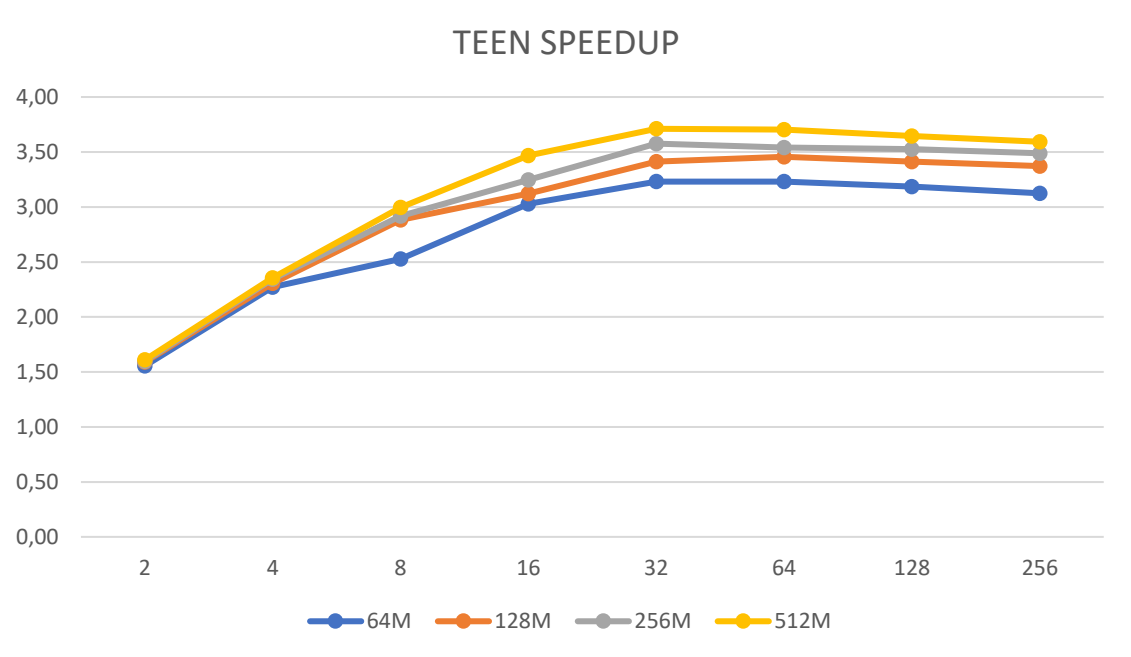


Com podem observar, al modificar l'script per a fer-lo més eficient i ràpid, obtenim que l'*orca* obté millors resultats degut a que té més potència individual i és capaç de calcular grans volums de dades de forma més ràpida i paral·lelitzar de forma més eficient que el servidor *teen*, que li costa més executar els grans volums de dades. Trobem la saturació esperada.

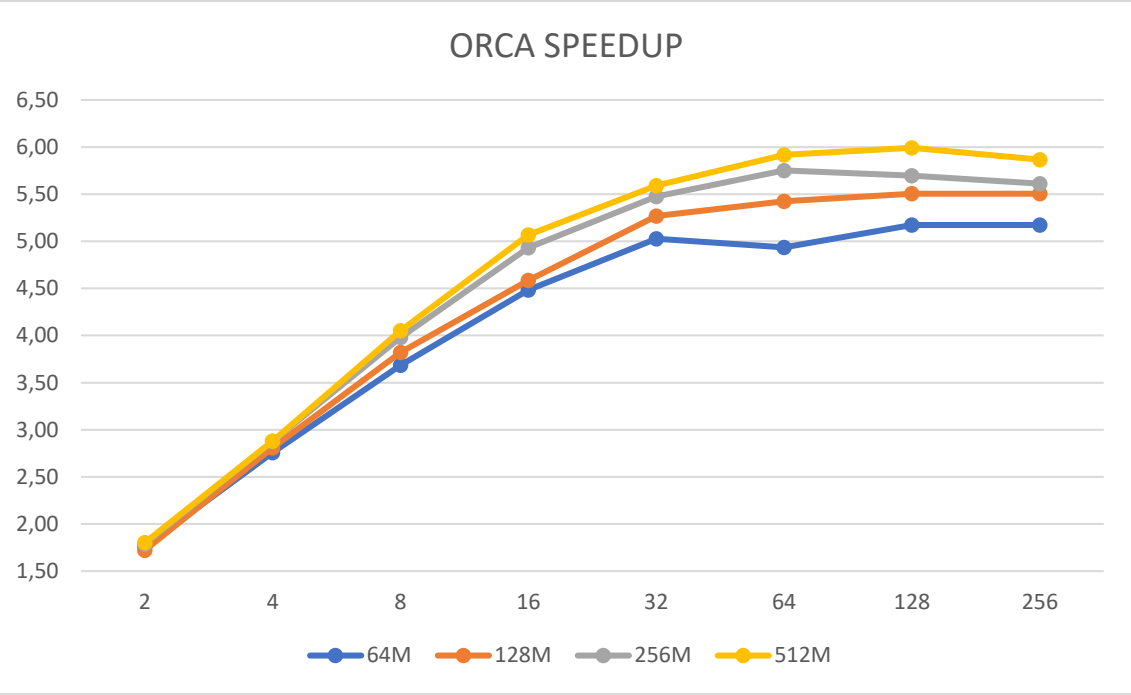
4. ANÀLISI SPEEDUP RELATIU

Finalment, després de la modificació del script i de obtenir els valors dels temps d'execució podem calcular l'*speedup* relatiu mitjançant la següent formula:

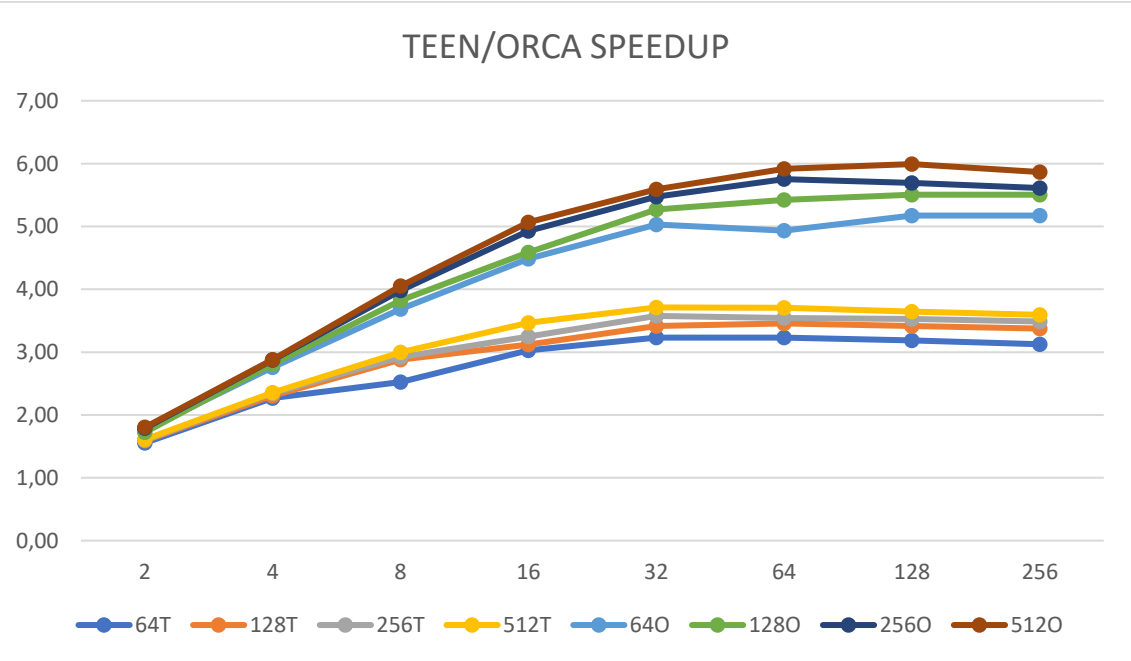
$$Speedup(E) = \frac{TEj\ sin\ M}{TEj\ con\ M} = \frac{Performance\ con\ M}{Performance\ sin\ M}$$



	64M	128M	256M	512M
2	1,56	1,59	1,60	1,61
4	2,27	2,31	2,34	2,36
8	2,53	2,88	2,92	3,00
16	3,03	3,12	3,25	3,47
32	3,23	3,41	3,58	3,71
64	3,23	3,46	3,54	3,70
128	3,18	3,41	3,53	3,65
256	3,13	3,37	3,49	3,59



	64M	128M	256M	512M
2	1,76	1,72	1,79	1,80
4	2,76	2,81	2,88	2,88
8	3,68	3,82	3,98	4,05
16	4,48	4,59	4,93	5,07
32	5,03	5,27	5,48	5,59
64	4,94	5,42	5,75	5,92
128	5,17	5,51	5,70	5,99
256	5,17	5,51	5,61	5,87



I, com podem observar, el servidor *orca* ens acaba oferint un speedup més alt que el servidor *teen* per els fets que hem explicat posteriorment:

- Paral·lelització més eficaç
- Més RAM
- Més nuclis
- Execució més ràpida de volums de dades més grans

5. CONCLUSIONS

- La saturació arriba tal i com s'espera al rendiment quan executem amb 128/256 threads, però actua diferent en els dos tipus de servidors. Com bé em indicat al estudi dels servidors, el servidor teen és capaç d'executar 32 threads concurrentment, mentre que el servidor orca és capaç d'executar-ne fins a 128. Per això obtenim que els resultats es saturen a partir d'un cert nombre de threads que causen que els resultats deixin de mostrar-se amb més diferència que l'anterior resultat a mesura que els augmentem. Per exemple, al servidor orca, tant els 128 threads com els 256 podem veure que majoritàriament obtenim resultats inferiors que a 64 threads, quan, utilitzant la lògica no hauria de ser així, però sabent la capacitat dels nostres servidors podem deduir que es degut a aquests factors explicats anteriorment.
- Quantes més parts paral·lelitzables o més temps en el que un treball es pot paral·lelitzar, més gran serà el *speedup* teòric i així s'ha traslladat a la millora de rendiment en les gràfiques, que ens mostren que aquesta afirmació darrera és certa.
- El overhead de threads causa que aquest *speedup* teòric no es vegi complementant reflectit en els quantitats altes (nombre de threads) i per tant sigui contraproduent utilitzar més threads.
- Després de totes les execucions fetes podem concloure amb seguretat que el servidor *orca* és més efectiu que el servidor *teen* cosa que confirma les nostres hipòtesis que teníem alhora d'estudiar els servidors. Tot això és degut al simple fet de que té més nuclis i més RAM.
- Finalment, aquesta pràctica ens ha ensenyat a com gestionar els threads de la forma més eficient possible, a aprofundir els nostres coneixements també sobre el funcionament de l'execució de threads en un processador i a explorar les característiques de processadors.