* Brenton Yiu
* byiu3 – 902601411
* Project 1: Experiments
* Taskgraph A
* For these experiments, I started out by finding out how many processors each taskgraph used. This was as easy as creating a printf statement in my source code that outputted the amount of processors at the end of the program. I also calculated the amount of bytes total that would be used for each taskgraph and outputted it so that I could see if I met the total budget.
* The first step I did was altered the amount of each context cache to meet the total budget of 54 kB. In order to do this I changed C which causes the most dramatic change in the total bytes.
* The second step I did was see which replacement policy, NMRU-FIFO or LRU, caused the smallest AAT. Third, I modulated the storage policy, subblocking or blocking, and saw which caused the smallest AAT.
* Fourth, I modulated the parameter S to see if increasing or decreasing it would cause a smaller AAT.
* Finally, I modulated the parameter B to see if increasing or decreasing it would cause a smaller AAT.
* The default parameter for taskgraph A was C=11, B=4, S=3, ST=Subblocking, and R=NMRU-FIFO. These parameters caused a total budget of 88704 bytes, which was greatly out of the 54 kB budget. I decreased C to be 10 and the new budge is 44928. Next, I tried LRU instead of NMRU-FIFO and found that the new AAT was 2.857586 compared to 2.857728. Third, I varied the S parameter between 2 and 4 and found that 3 is the best choice. Fourth, I varied the B parameter between 4 and 7 and found that 6 is the best choice. I then looked to see if the size of the cache met the budget and found that it was under-budget enough to increase the C from 10 to 11. This decreased the AAT again and I found that if I changed the B parameter to 7, I now get the lowest AAT possible.
* Choice: C = 11

B = 7

S = 3

ST = Blocking

R = LRU

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| # | C | B | S | ST | R | Cache Size | AAT | Comments |
| 1 | 11 | 4 | 3 | S | F | 88704 | ﻿2.855551 | Default |
| 2 | 10 | 4 | 3 | S | F | 44928 | ﻿2.85772 | Decrease the C parameter to meet the budget |
| 3 | 10 | 4 | 3 | S | L | ﻿48384 | ﻿2.857586 | Check the replacement policy |
| 4 | 10 | 4 | 2 | B | L | ﻿47232 | ﻿14.722831 | Check the storage policy |
| 5 | 10 | 4 | 3 | B | L | ﻿47808 | ﻿2.451593 | Check above the S parameter |
| 6 | 10 | 4 | 4 | B | L | ﻿48384 | ﻿4.467932 | Check above the S parameter again |
| 7 | 10 | 5 | 3 | B | L | ﻿28512 | ﻿2.249296 | Check above the B parameter |
| 8 | 10 | 6 | 3 | B | L | ﻿18864 | ﻿2.149734 | Check above the B parameter |
| 9 | 10 | 7 | 3 | B | L | ﻿14040 | ﻿4.349606 | Check Above the B parameter |
| 10 | 11 | 6 | 3 | B | L | ﻿37440 | ﻿2.149174 | Increase the C parameter to meet the budget |
| 11 | 11 | 7 | 3 | B | L | ﻿27936 | 2.102622 | Increase the B parameter for better AAT |
| 12 | 11 | 7 | 4 | B | L | ﻿28080 | ﻿4.105184 | Try increasing the B parameter again, worse AAT |

* Taskgraph B

Again, for this experiment, I changed the C parameter to get the total size below the budget. Then I checked to see which replacement policy and storage policy would give the smallest AAT. Then I modulated the S parameter and found the smallest AAT and found that this value was when S is 3 and B is 5. Any other values that I tried would cause the cache size to be so big that it fell out of budget.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| # | C | B | S | ST | R | Cache Size | AAT | Comments |
| 1 | 8 | 6 | 2 | S | L | 4256 | ﻿23.434765 | Default |
| 2 | 11 | 6 | 2 | S | L | ﻿33280 | ﻿23.095260 | Increase C parameter to meet budget |
| 3 | 11 | 6 | 2 | S | F | ﻿31744 | ﻿23.117628 | Check replacement policy |
| 4 | 11 | 6 | 2 | B | L | ﻿33024 | ﻿25.704069 | Check storage policy |
| 5 | 11 | 6 | 1 | S | L | ﻿2048 | ﻿23.109152 | Check below the S parameter |
| 6 | 11 | 6 | 2 | S | L | ﻿33280 | ﻿23.095260 | Default |
| 7 | 11 | 5 | 3 | S | L | ﻿50688 | ﻿22.973923 | Check above the S parameter, need to change B=5 to keep validity |
| 8 | 11 | 4 | 4 | S | L | ﻿86016 | ﻿24.990560 | Decrease B and increase S and check |
| 9 | 11 | 3 | 5 | S | L | ﻿157696 | ﻿28.792416 | Keep decreasing B and increasing S |
| 10 | 11 | 2 | 6 | S | L | ﻿303104 | ﻿37.145160 | Keep decreasing B and increasing S |
| 11 | 11 | 4 | 3 | S | L | ﻿153600 | ﻿21.912915 | Going off of row 7, Decrease B with S=3 to see if we get a better AAT. |

This table above shows the different parameters I tried to get the smallest AAT. We see that the smallest AAT that falls within budget is at row 7 when C=11, B=5, S=3, Subblocking, and LRU parameters are used.

Choice: C = 11

B = 5

S = 3

ST = Subblocking

R = LRU

* Taskgraph C

For taskgraph C, I started out changing the C parameter to meet the required budget. I then checked to see which replacement policy and storage policy caused the smallest AAT. It was found to start with that C=10, blocking caused the smallest AAT. LRU and FIFO showed the same AAT when I tried to test them so I had to change the parameters, specifically S=2, to show that LRU is the better option. Next, I tried to vary the S parameter and found that S=2 is the best option. I then varied the B parameter and found that B=6 gives the best AAT, but the cache size became really small. I increased the C parameter to increase the cache size and found that C=13 gave the best value.

Choice: C = 13

B = 6

S = 2

ST = Blocking

R = LRU

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| # | C | B | S | ST | R | Cache Size | AAT | Comments |
| 1 | 6 | 2 | 1 | B | L | ﻿2144 | ﻿11.354116 | Default |
| 2 | 10 | 2 | 1 | B | F | ﻿32256 | ﻿1.830024 | Get the biggest size but still in the budget |
| 3 | 10 | 2 | 1 | B | L | ﻿35328 | ﻿1.830024 | Check the replacement policy |
| 4 | 10 | 2 | 2 | B | F | ﻿32768 | ﻿1.865255 | Since the AAT was the same when checking the replacement policy before, increase S and check the replacement policy. |
| 5 | 10 | 2 | 2 | B | L | ﻿35840 | ﻿1.811989 | Check the replacement policy |
| 6 | 10 | 2 | 3 | B | L | ﻿36352 | ﻿2.790413 | Check above the S parameter |
| 7 | 10 | 1 | 2 | B | L | ﻿69632 | ﻿2.134100 | Check below the B parameter |
| 8 | 10 | 3 | 2 | B | L | ﻿18944 | ﻿1.624123 | Check above the B parameter |
| 9 | 10 | 4 | 2 | B | L | ﻿10496 | ﻿1.531448 | Keep checking above the B parameter |
| 10 | 10 | 5 | 2 | B | L | ﻿6272 | ﻿1.772822 | Keep checking above the B parameter |
| 11 | 12 | 4 | 2 | B | L | ﻿40960 | ﻿1.237333 | B = 4 gives smallest AAT but has very small size so increase C. |
| 12 | 12 | 5 | 2 | B | L | ﻿24576 | ﻿1.189874 | Increase B again to check for better AAT |
| 13 | 12 | 6 | 2 | B | L | ﻿16384 | ﻿1.177252 | Keep checking above the B parameter |
| 14 | 12 | 7 | 2 | B | L | ﻿12288 | ﻿1.211547 | Keep checking above the B parameter |
| 15 | 13 | 6 | 2 | B | L | ﻿32512 | ﻿1.121050 | Increase the C parameter again to get biggest size below the budget |
| 16 | 13 | 7 | 2 | B | L | ﻿24448 | ﻿1.122510 | Check B = 7, doesn’t work. |

* Taskgraph D

Again, I decreased the C parameter until the total budget was satisfied. I then ran tests to figure out which replacement policy and storage policy caused smaller AAT values. I found that Blocking and LRU gave the smallest AAT. Next, I tested different values for the S parameter and found that the default S = 2 gave the best AAT. I then changed the B parameter and ran tests and found that B = 5 gives the best AAT. I did find that B = 4 gave a smaller AAT than B = 4 but the total size was out of budget. I tried decreasing the C parameter and running the test with B = 4 again and found that the AAT was much greater than the smallest AAT value I found.

Choice: C = 11

B = 5

S = 2

ST = Blocking

R = LRU

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| # | C | B | S | ST | R | Cache Size | AAT | Comments |
| 1 | 14 | 6 | 2 | B | L | ﻿258048 | ﻿1.015756 | Default Choices |
| 2 | 11 | 6 | 2 | B | L | ﻿33024 | ﻿1.962724 | Get within the size budget |
| 3 | 11 | 6 | 2 | B | F | ﻿31488 | ﻿2.047673 | Check if NMRU-FIFO gives a better AAT |
| 4 | 11 | 6 | 2 | SB | L | ﻿33280 | ﻿2.106137 | Check if Subblocking is better |
| 5 | 11 | 6 | 1 | B | L | ﻿32768 | ﻿2.489031 | Check below the S parameter |
| 6 | 11 | 6 | 3 | B | L | ﻿33280 | ﻿2.873982 | Check above the S parameter |
| 7 | 11 | 6 | 4 | B | L | ﻿33536 | ﻿4.853658 | Keep checking above the S parameter |
| 8 | 11 | 5 | 2 | B | L | ﻿49664 | ﻿1.741225 | Check below the B parameter |
| 9 | 11 | 6 | 2 | B | L | ﻿33024 | ﻿1.962724 | Default |
| 10 | 11 | 7 | 2 | B | L | ﻿24704 | ﻿2.622110 | Check above the B parameter |
| 11 | 11 | 4 | 2 | B | L | ﻿82944 | ﻿1.534330 | Keep checking below the B parameter |
| 12 | 11 | 5 | 3 | B | L | ﻿50176 | ﻿2.651807 | With B=5, check above the S parameter |
| 13 | 11 | 5 | 1 | B | L | ﻿49152 | ﻿2.196176 | With B=5, check below the S parameter |
| 14 | 10 | 4 | 2 | B | L | ﻿41984 | ﻿2.364243 | Check if decreasing the cache size gives a better AAT with B = 4 since it is small in row 11. |