Project 1 Schedule Simulation

CSCE 4600 Operating Systems

Faris Hawamdeh

Dustin Fennessy

Eric Guzman

d

D

# Introduction

In this project, different scheduling strategies had to be developed and implemented to simulate a scheduler that assigns processes (or jobs) to a set of available processors or processing nodes. The different scheduling strategies had to attempt to provide the best schedule combination for each of the four hardware scenarios detailed in the project requirement. The experiment generated 5 input graphs using a modified process generator from homework 2 that were scheduled using a scheduling strategy developed for each of the 4 scenarios.

# Scenarios

In each of the 4 scenarios, it is assumed that the computer that it is simulating has 5 processors available to use by the scheduler. The scheduler is to schedule 100 processes with specific requirements. The processes have burst times (processing times) between 10 million cycles (10 \* 106 cycles) to 50 trillion cycles (50 \* 10^12) cycles with memory requirements between 0.25 MB and 8 GB. In scenario 1 to 3, it is assumed that the processes will arrive in a batch and are scheduled all at once.

## Scenario 1

Scenario 1 is the simplest of the 4 scenarios. It is assumed that all 5 processors are identical in speed and memory. The goal of scenario 1 is to implement an algorithm that assigns the set of 100 processes to the 5 processors with the minimal total turnaround time for the system. For this scenario, the scheduling algorithm implemented was Shortest Job First (SJF). The reason that an unmodified SJF was chosen was that because the 5 processors are identical so SJF will yield the best average waiting times for processes assigned to the 5 processors. To ensure the total turnaround time for the system, the scheduler finds the processor with the least load and assigns the next process to that processor. For this simplistic case, the scheduler choses the processor with the least load by finding the processor with the fewest cycles that have already been assigned to the processor.

## Scenario 2

Scenario 2 is a variation of scenario 1 where the 5 processors do not have identical memory availability. The processors are identical in speed but have the memory availability that have been outlined in the project requirement: PA= PB = 2 GB, PC= PD = 4 GB and PE= 8GB. In order for a specific processor to execute a process, the processor must have sufficient memory available to load the process into memory. For this scenario, the scheduling algorithm implemented was a modified SJF that ensures that process that is compatible with the processor’s memory limitation. The scheduler first picks the processor with least load at the time, then picks a compatible process using SJF that has a memory requirement that is within that processor’s memory limitation. The goal of this implementation is to maintain the minimal average wait time on processors as well as manage the processor limitations to create a schedule with the best total turnaround time.

## Scenario 3

Scenario 3 is a variation of scenario 3 where the processors do not have identical clock speeds. The processors are identical in memory availability but have the clock speeds that have been outlined in the project requirement: PA= PB = 2 GHz, PC= PD = 3 GHz and PE= 4 GHz. The goal of the implementation in this scenario is to make the best use of the faster processors to minimize the total turnaround time. For this scenario, the scheduling algorithm implemented was a modified SJF that ensures that takes advantage of the fast 4 GHz processor that system has. For processors 1-4, the scheduler schedules processes the same way as in scenario 1, but the scheduler takes advantage of the faster processor 5 by scheduling the longest jobs first when the processor is picked by the scheduler. By assigning the longest processes to the fastest processor first, the total turnaround time is reduced because the longest processes are completed before the slower processors that are working through the queue from the other side.

## Scenario 4

Scenario 4 is identical to scenario 1 except that the processes are assumed to arrive sequentially and must schedule them one by one in the order they arrive. The speed of the processors are assumed to be 2 GHz since it was not specified in the project requirements. The scheduler cannot inspect the set of processes. The implemented scheduling algorithm that fits the limitations set out by the scenario to pick processes is essentially First in First Out (FIFO). The scheduler picks the process at the top of the set that has yet to be assigned to a processor and finds the processor with the least load. Because the processes must be scheduled one by one in the order they arrive, the only action the scheduler can take is to spread the process load over the 5 processors such that none of the processors are overloaded with burst times. The total turnaround time will be the same as Scenario 1, but the average wait time will be influenced by the order in which the longest burst time processes are randomly ordered in the set.

# Experiment

In order to test the effectiveness of the scheduling algorithms implemented as part of the project. The modified process generator was used to generate 5 different sets of 100 processes in separate files that were run by all 4 scheduling algorithms. The results of the experiments are listed in tables below. Samples of the results printed by the schedulers and the input files are attached in the appendix.

## Scenario 1

Table below shows the results of running the 5 input sets for the algorithm implemented for scenario 1.

|  |  |  |
| --- | --- | --- |
|  | Total Turnaround of System (Millions of Instructions) | Average Wait of Processes (Millions of Instructions) |
| Input 1 | 534953088 | 199201919 |
| Input 2 | 531582433 | 194750558 |
| Input 3 | 507937696 | 179166716 |
| Input 4 | 486824811 | 170219768 |
| Input 5 | 537693206 | 188304229 |

By using shortest job first as the scheduling algorithm with simple load balancing, the best case schedule can be achieved when all the processor are identical. The turnaround of the system is minimal when burst times are distributed as evenly as possible and the average wait time is minimal when shortest job first is used.

## Scenario 2

Table below shows the results of running the 5 input sets for the algorithm implemented for scenario 2.

|  |  |  |
| --- | --- | --- |
|  | Total Turnaround of System (Millions of Instructions) | Average Wait of Processes (Millions of Instructions) |
| Input 1 | 826252973 | 228712001 |
| Input 2 | 624724977 | 201038425 |
| Input 3 | 561642618 | 182454072 |
| Input 4 | 607711421 | 178928207 |
| Input 5 | 770180472 | 205179430 |

As the results of the preceding table show, the limitation of memory availability has a great effect on the performance of the system. The actual difference in performance varies greatly based on the distribution of the processes with large memory requirements. If there are many processes with very large memory requirements, the few processors with large memory availability become overloaded with processes while the smaller processors sit idle. This causes the system performance to suffer depending on the distribution within any given set of processes.

## Scenario 3

Table below shows results of running the 5 input sets for the algorithm implemented for scenario 3 using longest job first only on the fastest processor, processor #5.

|  |  |  |
| --- | --- | --- |
|  | Total Turnaround Time of System (Millions of Seconds) | Average Wait Time of Processes (Millions of Seconds) |
| Input 1 | 190200 | 77726 |
| Input 2 | 187908 | 75820 |
| Input 3 | 178038 | 70461 |
| Input 4 | 176987 | 66165 |
| Input 5 | 195255 | 73030 |

Table below shows results of running the 5 input sets for running a purely shortest job first algorithm using the same constraints.

|  |  |  |
| --- | --- | --- |
|  | Total Turnaround Time of System (Millions of Seconds) | Average Wait Time of Processes (Millions of Seconds) |
| Input 1 | 253499 | 75778 |
| Input 2 | 251983 | 74081 |
| Input 3 | 238439 | 68007 |
| Input 4 | 231165 | 64708 |
| Input 5 | 254908 | 71590 |

As the results in the 2 preceding tables show, the strategy of using the fastest processor on the system to tackle the longest jobs first gives a boost to performance by reducing the total turnaround time of the system by 20% over just using a regular shortest job first algorithm. The only drawback is the longer wait time on the processor that is using longest job first than if it using shortest job first. However from the perspective of the entire system, the modification of using the fastest processor to tackle the biggest jobs is beneficial for performance. There is no chance for the slowest processors to execute the longest processes when using the modified scheduling algorithm which greatly improves system turnaround.

## Scenario 4

Table below shows the results of running the 5 input sets for the algorithm implemented for scenario 4.

|  |  |  |
| --- | --- | --- |
|  | Total Turnaround Time of System (Millions of Seconds) | Average Wait Time of Processes (Millions of Seconds) |
| Input 1 | 263367 | 119217 |
| Input 2 | 259984 | 122050 |
| Input 3 | 255772 | 117391 |
| Input 4 | 239227 | 114083 |
| Input 5 | 270523 | 122175 |

As the results in the preceding table shows, the limitation of not being able to look at the entire set of processes when making a decision on which process to schedule next has a drastic impact on both the turnaround time of the system as well as the average wait time of the processes.

# Appendix

## Sample Input File

The sample input file shown below has been formatted so that it fits in one page. The actual input file has only 2 columns: Left is number of cycles in millions ( x 106 ), Right is memory in kilobytes.

|  |  |
| --- | --- |
| 40038604 3625318  22787628 4393626  21028167 3719510  37501393 3273561  24470043 2829451  18990151 458247  11319358 3062959  36319629 4850974  21251394 4059890  44836847 1234019  7433530 4683279  24680736 870871  30785255 2370525  31134352 2203591  22603713 4979308  15706973 4950238  21190190 4657694  38498093 3728093  31615548 2516638  30391360 4206965  33450285 4488736  17978496 2538508  25595159 2493053  25211186 4060425  17756768 391625  26447782 2311015  19708917 2342080  21666962 4611624  25932262 4215563  32446392 4070100  24845308 4106719  13549816 4232944  31521626 1612244  30270634 3415405  36603105 1070356  12645146 2383348  22614643 4544331  48150226 754628  15998324 46140  4470754 2570929  29562556 3367834  17152144 763668  15524458 1812617  12009947 341480  24447309 3791785  21821772 4424682  14568583 3380650  22723409 4717349  20711239 2846674  23246620 4381406 | 40116827 4461236  25831919 3148398  24605341 2587003  18513160 926647  19269896 3185735  21160364 3476583  32643858 4206876  32203111 263468  23374903 3364643  16670588 3232640  26437582 4822619  24134404 1129470  28016320 4163069  24659840 1949184  10766253 1977609  39714406 488396  25790278 4663786  24727014 1575760  37256942 3116069  14613203 3267322  29479666 4953279  22255389 2607590  36274849 4112522  28281291 3347273  33207999 2157714  14098123 1275411  28985817 510392  31169895 3303554  30616405 291241  30095713 622953  33708641 2964347  22797996 4370400  28418426 3583023  26673523 1444331  25740414 4256961  29254006 2447723  24357181 1083968  14856870 1791219  34477211 2821004  25545188 3984733  35641276 2223560  23091209 2596954  30629889 3265112  48152373 1454690  23029488 4970785  21118799 1847532  35259547 2632832  34338884 4677010  23696918 3358576  32414481 825063 |

## Sample Results (Scenario 3 Input 3)

|  |  |
| --- | --- |
| Current processor #: 0  Clock Speed: 2000  Processor Total Cycle Count: 346972274  Processor Turn Around Time: 173486 (Million Seconds)  Process List  ID Cycles Memory Wait Time  0 749861 3672051 0  88 8402212 4124499 749861  93 12457937 2632843 9152073  43 14094687 4305720 21610010  59 15219807 2533172 35704697  22 18507634 1678378 50924504  37 19089793 4512612 69432138  58 19710276 4851620 88521931  85 21207226 4861544 108232207  47 22839381 537280 129439433  8 23969993 2656386 152278814  97 25449092 1033327 176248807  10 26286533 1809379 201697899  26 27974249 4494802 227984432  77 28624142 1917102 255958681  31 30788131 3549941 284582823  87 31601320 576555 315370954  Processor Average Wait Time: 59107 (Million Seconds)  Current processor #: 1  Clock Speed: 2000  Processor Total Cycle Count: 356076543  Processor Turn Around Time: 178038 (Million Seconds)  Process List  ID Cycles Memory Wait Time  16 3142972 474615 0  68 9326837 4794798 3142972  55 13000073 2508295 12469809  67 14389796 4760063 25469882  40 16289522 1876879 39859678  45 18616719 3397948 56149200  80 19408009 4647441 74765919  60 19742362 3459721 94173928  33 21981477 1958735 113916290  42 23304501 1477350 135897767  12 24452103 1657397 159202268  51 25689875 3570187 183654371  64 26629467 1039760 209344246  82 28099027 2271661 235973713  57 28930965 1786126 264072740  41 30889070 3442245 293003705  79 32183768 2407212 323892775  Processor Average Wait Time: 61804 (Million Seconds)  Current processor #: 2  Clock Speed: 3000  Processor Total Cycle Count: 517423825  Processor Turn Around Time: 172474 (Million Seconds)  Process List  ID Cycles Memory Wait Time  91 6195903 4809390 0  1 11925242 4610763 6195903  81 12733272 3834275 18121145  71 13895916 3742906 30854417  83 14398854 3554828 44750333  5 15534028 3946275 59149187  19 17567712 775923 74683215  50 18690020 3104420 92250927  27 19184468 1460682 110940947  6 19624717 3345116 130125415  73 20669852 386711 149750132  66 21851879 841612 170419984  72 22266536 4668693 192271863  70 23388485 1823454 214538399  56 24085117 2932878 237926884  39 25376238 2489432 262012001  3 25787162 4526518 287388239  14 26317755 1871791 313175401  62 27629429 1304051 339493156  74 28135453 4696334 367122585  49 28638099 2752600 395258038  89 30408558 3842913 423896137  35 31041657 2219819 454304695  96 32077473 617515 485346352  Processor Average Wait Time: 64799 (Million Seconds) | Current processor #: 3  Clock Speed: 3000  Processor Total Cycle Count: 530463293  Processor Turn Around Time: 176821 (Million Seconds)  Process List  ID Cycles Memory Wait Time  4 8313552 4850329 0  52 12057252 3898760 8313552  84 13552509 3589142 20370804  98 14281795 2030861 33923313  18 15165193 2857331 48205108  90 16549166 4370292 63370301  28 18550331 3516365 79919467  53 18894260 690653 98469798  9 19555903 3453979 117364058  23 19738957 3976901 136919961  63 20991138 2989880 156658918  94 22177605 3212710 177650056  69 22923139 193864 199827661  25 23423694 2125507 222750800  54 25296418 2692763 246174494  92 25684329 4752210 271470912  86 26149315 4054144 297155241  95 26939946 1401853 323304556  20 28060450 4705930 350244502  21 28339264 3230507 378304952  44 28973102 911917 406644216  24 30847769 1177677 435617318  17 31438119 3877717 466465087  11 32560087 2197445 497903206  Processor Average Wait Time: 67159 (Million Seconds)  Current processor #: 4  Clock Speed: 4000  Processor Total Cycle Count: 692685854  Processor Turn Around Time: 173171 (Million Seconds)  Process List  ID Cycles Memory Wait Time  29 48026439 1450335 0  2 47332843 1767230 48026439  7 47121835 758000 95359282  99 41602280 4551283 142481117  65 39495235 571274 184083397  75 39480746 1726446 223578632  15 38986607 1649977 263059378  13 37402874 4347576 302045985  30 37309680 3789840 339448859  32 36980356 3402148 376758539  78 36666975 4447049 413738895  46 35818797 3659827 450405870  61 35603642 1450002 486224667  38 35181677 3282945 521828309  76 35133492 4819310 557009986  36 34852463 3185632 592143478  34 32870386 4574534 626995941  48 32819527 2695326 659866327  Processor Average Wait Time: 82671 (Million Seconds)  Total Cycles: 20532937264 (Million Seconds)  Total Turn Around Time: 873990 (Million Seconds)  Total Wait Time: 7046156 (Million Seconds)  Average Wait Time: 70461 (Million Seconds) |