**Intro to High Performance Computing CW1**

|  |  |  |
| --- | --- | --- |
| **Change** | **Time after change (s)** | **Input size** |
| None | 5.903894  6.084367  5.954831 | 1024x1024 |
| Added compiler flag -O0 | 5.894954  5.927843  5.871533 | 1024x1024 |
| Added compiler flag -O1 | 2.000707  2.101893  1.987344 | 1024x1024 |
| Added compiler flag -O2 | 2.007964  2.019441  1.991713 | 1024x1024 |
| Added compiler flag -O3 | 2.006684  2.009513  2.018796 | 1024x1024 |
| Added compiler flag -Os | 2.001239  1.998321  1.987236 | 1024x1024 |
| Added compiler flag -Ofast | 1.186113  1.189751  1.197843 | 1024x1024 |
| Added compiler flags -Ofast -mtune=native | 1.187224  1.190578  1.186724 | 1024x1024 |
| Added compiler flag -Ofast | 35.819981  35.674533  36.087991 | 4096x4096 |
| In void Stencil(...), changed all “.../ 5.0;” to “... \* 0.2;” | 1.186138  1.189921  1.181487 | 1024x1024 |
| In void Stencil(...), use to access/change cell “tmp\_image[j+i\*height] 5 times. Now only accesses once | 0.981292  0.968756  0.981153 | 1024x1024 |
| In void Stencil(...), changed all “...3.0 \* 0.2;” or “...0.5\*0.2;” to one value, removing one multiplication computation | 0.981774  0.982577  0.976993 | 1024x1024 |
| In void Stencil(...), switched the for loops from Row Major to Column Major | 0.296190  0.300486  0.291099 | 1024x1024 |
| Changes listed above | 6.094060  5.983122  6.105973 | 4096x4096 |
| In void Stencil(...), changed the order of the image array access additions are done | 0.283090  0.278631  0.281136 | 1024x1024 |
| Changed ++i and ++j to i++ and j++ | 0.283100  0.281879  0.290599 | 1024x1024 |
| Added compiler module in bcp4 languages/gcc/9.1.0 | 0.266931  0.262488  0.266115 | 1024x1024 |
| As above | 5.929233  5.953214  5.926451 | 4096x4096 |
| Calculating current position only once, and for (j+-1) values have replaced it with ‘currentPos+-1’ | 0.262777  0.259711  0.265439 | 1024x1024 |
| Adding all values that were \*0.1 together before multiplying, removing 2 multiplication operations. (Keep array access in order so is sequential) | 0.254630  0.251638  0.255876 | 1024x1024 |
| Using stored CurrentPos value to calculate all array access positions  Changed from “image[j + (i + 1) \* height]” to “image[currentPos + height]” | 0.250060  0.249384  0.252733 | 1024x1024 |
| Calculating CurrentPos differently so it can be incremented in the inner- most loop, instead of being recomputed every loop | 0.245467  0.251079  0.247886 | 1024x1024 |
| Changed to icc compiler. Loaded icc/2017.1.132-GCC-5.4.0-2.26 module | 0.245409  0.243877  0.243269 | 1024x1024 |
| Changed makefile to  “icc -std=c99 -Wall -fast $^ -o $@” | 0.173775  0.182969  0.175422 | 1024x102411.618329 |
| Chnaged Makefil to “icc -std=c99 -Wall -fast-transcendentals $^ -o $@” | 0.242298  0.238563  0.246007 | 1024x1024 |
| Changed to icc compiler icc/2016.3.210-GCC-5.4.0-2.26 | 0.181165  0.184983  0.179930 | 1024x1024 |
| Change all double types to float (except timer related values) | 0.179501  0.179292  0.179444 | 1024x1024 |
| Re-shuffled inner for loop of stencil from:  tmp\_image[currentPos] = ((image[currentPos - height] + image[currentPos - 1])\*0.1)  + (image[currentPos] \* 0.6)  + ((image[currentPos + 1] + image[currentPos + height]) \* 0.1);  to:  (image[currentPos] \* 0.6)  + ((image[currentPos - height] + image[currentPos - 1] + image[currentPos + 1] + image[currentPos + height]) \* 0.1); | 0.170361  0.166510  0.166101 | 1024x1024 |
| Re-shuffled inner for loop calculations from:  (image[currentPos] \* 0.6)  + ((image[currentPos - height] + image[currentPos - 1] + image[currentPos + 1] + image[currentPos + height]) \* 0.1);  to:  tmp\_image[currentPos] = ((image[currentPos - height] + image[currentPos - 1] + image[currentPos + 1] + image[currentPos + height]) \* 0.1)  + (image[currentPos] \* 0.6); | 0.155586  0.154831  0.158453 | 1024x1024 |
| All changes above | 3.243580  3.237303  3.247561 | 4096x4096 |
| All changes above | 11.715121  11.692594  11.618329 | 8000x8000 |
| Gained a compiler report by running icc with “–qopt-report=5“ | Code is being vectorized, howvever some “unaligned access” issues are occuring, which are less efficient  <https://software.intel.com/en-us/articles/fdiag15126>  https://software.intel.com/en-us/articles/fdiag15134 | 1024x1024 |

NEW DATA IN PROPER ORDER

Icc -std=c99 -Wall -fast -qopt-report=5 -qopt-report-file=stderr $^ -o $@

|  |  |  |
| --- | --- | --- |
| **Change** | **Time after change (s)** | **Input size** |
| None | 5.805884  5.904141  5.907237 | 1024x1024 |
| Loaded languages/gcc/9.1.0  REPORT:  *Unit growth for small function inlining: 0->0 (0%)*  *Inlined 0 calls, eliminated 0 functions* | 5.875045  5.874897  5.873470 | 1024x1024 |
| Changed Makefile to icc  Loaded languages/intel/2016-u3-cuda-8.0  REPORT:  *Stencil() Not vectorised due to assumed dependence between tmp\_image line 65 and tmp\_image line 69*  *Most other for loops in code were vectorised* | 2.004066  2.003769  2.003888 | 1024x1024 |
| Used language/intel/2017.01  REPORT:  *Changed order of the for loops inside Stencil(…)*  *Stencil for loops Vectorisation was enabled and performed with minimal unalignment access*  *Able to peel loop*  *Vector length 2*  *Unmasked aligned load 1*  *Unmasked aligned stores 5*  *Unmasked unaligned load 5* | 1.795804  1.795908  1.796646 | 1024x1024 |
| Used language/intel/2018-u3  REPORT:  *Changed order of the for loops inside Stencil(…)*  *Stencil for loops Vectorisation was enabled and performed with minimal unalignment access*  *Able to peel loop*  *Vector length 2*  *Unmasked aligned load 1*  *Unmasked aligned stores 5*  *Unmasked unaligned load 5* | 1.796016  1.795903  1.796347 | 1024x1024 |
| Used Compiler flag -O0 | 6.045124  6.045124  6.045124 | 1024x1024 |
| Used Compiler flag -O1 | 2.002685  2.002447  2.002114 | 1024x1024 |
| Used Compiler flag -O2 | 1.795450  1.795518  1.795528 | 1024x1024 |
| Used Compiler flag -O3 | 1.796668  1.796386  1.796326 | 1024x1024 |
| Used Compiler flag -Os | 2.003531  2.003419  2.003054 | 1024x1024 |
| Used Compiler flag -Ofast  REPORT:  *Vector length 2*  *Unmasked aligned load 1*  *Unmasked aligned stores 5*  *Unmasked unaligned load 5*  *Changed order of the for loops inside Stencil(…)*  *Stencil for loops Vectorisation was enabled and performed with minimal unalignment access*  *Able to peel loop* | 0.250762  0.243643  0.244600 | 1024x1024 |
| Used Compiler flag -Ofast-mtune=native  REPORT:  *Vector length 2*  *Unmasked aligned load 1*  *Unmasked aligned stores 5*  *Unmasked unaligned load 5*  *Changed order of the for loops inside Stencil(…)*  *Stencil for loops Vectorisation was enabled and performed with minimal unalignment access*  *Able to peel loop* | 1.795566  1.795755  1.795737 | 1024x1024 |
| Used Compiler flag -fast  REPORT:  *Vector length 4*  *Unmasked unaligned load 5*  *Unmasked unaligned store 5*  *No aligned access* | 0.195560  0.185741  0.186437 | 1024x1024 |
| Switching from Column to Row major can be done by switching the for loops around in Stencil(), however the compiler is doing this for us already so no speed improvement would be seen.  No change as compiler does it for us | 0.193184  0.185437  0.186299 | 1024x1024 |
| Changed all  *Tmp\_image[j+i\*height] =*  *Image[j+i\*height]\*(3.0/5.0)* ***OR*** *(0.5/5.0)*  To  *Tmp\_image[j+i\*height] =*  *Image[j+i\*height]\*0.6* ***OR*** *0.1* | 0.190995  0.185497  0.186443 | 1024x1024 |
| Removed 4 accesses of *tmp\_image[j+i\*height]* by changing all *+=* to *+* | 0.186736  0.177447  0.177127 | 1024x1024 |
| Changing memory access by calculating index *currentPos = j+i\*height*  and adding *+1/-1* and *height* where appropriate | 0.184885  0.178962  0.177978 | 1024x1024 |
| Add together all terms *\*0.1* before multiplying by the constant to remove multiplication operations | 0.189350  0.178125  0.178673 | 1024x1024 |
| Changed order of accesses so *image[currentpos+/-…]* accesses is done sequentially, with image[currentPos]\*0.6 being added last | 0.189200  0.179186  0.178939 | 1024x1024 |
| Changed *++I*  and *++j* to *i++* and *j++* | 0.188878  0.178646  0.178729 | 1024x1024 |
| Replaced all *double* types with *float* | 0.157259  0.154261  0.154729 | 1024x1024 |
| Testing 4096x4096 | 3.224267  3.217690  3.221639 | 4096x4096 |
| Testing 8000x8000 | 11.635089  11.641640  11.681603 | 8000x8000 |
|  |  |  |