HPC Homework 2

Austin McDowell

Finding Memory Bugs

I start off by compiling `val_test01.cpp` with my `Makefile`. Then I run Valgrind using `valgrind --tool=memcheck --leak-check=full ./val_test01`. This returns and invalid write of size 4 and an invalid read of size 4 around lines 84 and 85. Invalid read and write occurs because I have attempted to access an array outside of the memory I have allocated for it. In this case I first try to write to `x[i]` for `n=10` and then try to read at the same point. In order to fix this I change the loop bounds from `i<=n` to `i<n`. Running valgrind again returns no leaks and `0 errors from 0 contexts`

Next I compile `val_test02` and run valgrind again. This time the test returns a message of a conditional jump or move that depends on uninitialized values. The error is traced to line 109 where the loop attempts to print out a value of `x[i]` that has not been initialized. To fix this I initialized the second half of `x` with zeros since this is what the uninitialized values printed. I recompile and run valgrind again and this time have no leaks and 0 errors.

Optimizing matrix-matrix multiplication

These matrix multiplications (MM) are ran on my laptop with processor:`Intel(R) Core(TM) i7-4980HQ CPU @ 2.80GHz`. I calculate the flop rate with the expression:

$$f = \frac{2N_{repeats}mnk}{1e9 \times t}$$

and the bandwidth with the expression:

$$m = \frac{N_{repeats}p^2}{1e9 \times t}\left(\frac{2p}{b} + 2\right) \times \texttt{sizeof(double)}$$

where p is the size of the matix and b is the `BLOCK_SIZE`.

Starting with a block size of 4 my MM timings are as follows:

Dimension	Time	Gflop/s	GB/s	Error
4	1.217870	1.642212	6.568847	0.000000e+00
52	1.117365	1.789933	3.855241	0.000000e+00
100	1.107231	1.808114	3.760878	0.000000e+00
148	1.101801	1.818321	3.734930	0.000000e+00
196	1.109743	1.804794	3.683253	0.000000e+00
244	1.115613	1.796946	3.652808	0.000000e+00
292	1.158474	1.762285	3.572851	0.000000e+00
340	1.162536	1.758060	3.557486	0.000000e+00
388	1.157775	1.816241	3.669931	0.000000e+00
436	1.201684	1.793256	3.619416	0.000000e+00
484	1.149857	1.774862	3.579061	0.000000e+00
532	1.216636	1.732616	3.491286	0.000000e+00
580	1.322606	1.770250	3.564918	0.000000e+00
628	1.376824	1.798873	3.620661	0.000000e+00
676	1.373300	1.799553	3.620403	0.000000e+00
724	1.298390	1.753727	3.526831	0.000000e+00
772	1.699325	1.624526	3.265887	0.000000e+00
820	1.345261	1.639438	3.294870	0.000000e+00
868	1.499411	1.744610	3.505299	0.000000e+00
916	1.768494	1.738372	3.491927	0.000000e+00
964	2.017390	1.776238	3.567217	0.000000e+00
1012	1.161950	1.783956	3.582015	0.000000e+00
1060	1.323281	1.800096	3.613777	0.000000e+00
1108	1.514271	1.796576	3.606123	0.000000e+00
1156	1.749663	1.765831	3.543882	0.000000e+00
1204	1.953679	1.786719	3.585310	0.000000e+00
1252	2.199422	1.784574	3.580550	0.000000e+00
1300	2.445413	1.796833	3.604724	0.000000e+00
1348	2.777107	1.764035	3.538538	0.000000e+00
1396	3.047439	1.785465	3.581161	0.000000e+00
1444	3.382675	1.780210	3.570284	0.000000e+00
1492	3.733666	1.779102	3.567744	0.000000e+00
1540	4.100612	1.781326	3.571906	0.000000e+00
1588	4.568992	1.752916	3.514663	0.000000e+00
1636	4.937752	1.773579	3.555831	0.000000e+00
1684	5.405570	1.766911	3.542216	0.000000e+00
1732	5.870476	1.770110	3.548397	0.000000e+00
1780	6.353449	1.775335	3.558650	0.000000e+00
1828	6.877075	1.776457	3.560689	0.000000e+00
1876	7.465387	1.768790	3.545122	0.000000e+00
1924	8.165621	1.744440	3.496133	0.000000e+00
1972	8.655159	1.772049	3.551286	0.000000e+00

Figure 1: Timing results for MM with $b = 4$

With a block size of 16 my MM timings are as follows:

Dimension	Time	Gflop/s	GB/s	Error
16	0.847102	2.360994	2.360994	0.000000e+00
64	0.836304	2.391666	1.494791	0.000000e+00
112	0.841495	2.377457	1.358547	0.000000e+00
160	0.850575	2.359627	1.297795	0.000000e+00
208	0.852676	2.364034	1.272941	0.000000e+00
256	0.862418	2.334444	1.240173	0.000000e+00
304	0.857394	2.359244	1.241707	0.000000e+00
352	0.874176	2.295022	1.199670	0.000000e+00
400	0.898239	2.280016	1.185609	0.000000e+00
448	0.944343	2.285155	1.183384	0.000000e+00
496	0.930013	2.361722	1.218953	0.000000e+00
544	0.964055	2.337884	1.203323	0.000000e+00
592	0.910309	2.279168	1.170383	0.000000e+00
640	0.901180	2.327117	1.192647	0.000000e+00
688	1.153698	2.258203	1.155360	0.000000e+00
736	1.040147	2.299798	1.174897	0.000000e+00
784	1.257725	2.298866	1.172891	0.000000e+00
832	1.021761	2.254659	1.149009	0.000000e+00
880	1.198707	2.274024	1.157685	0.000000e+00
928	1.405405	2.274587	1.156902	0.000000e+00
976	1.634500	2.275227	1.156263	0.000000e+00
1024	0.957169	2.243578	1.139317	0.000000e+00
1072	1.091610	2.257079	1.145384	0.000000e+00
1120	1.236159	2.273055	1.152763	0.000000e+00
1168	1.423558	2.238635	1.134651	0.000000e+00
1216	1.586528	2.266643	1.148234	0.000000e+00
1264	1.873366	2.155999	1.091645	0.000000e+00
1312	1.996359	2.262522	1.145057	0.000000e+00
1360	2.227703	2.258341	1.142455	0.000000e+00
1408	2.439701	2.288239	1.157121	0.000000e+00
1456	2.744820	2.249056	1.136886	0.000000e+00
1504	3.131309	2.172939	1.098028	0.000000e+00
1552	3.411432	2.191636	1.107115	0.000000e+00
1600	3.744886	2.187516	1.104696	0.000000e+00
1648	4.210669	2.125937	1.073289	0.000000e+00
1696	4.811262	2.027909	1.023520	0.000000e+00
1744	4.732827	2.241552	1.131058	0.000000e+00
1792	5.025067	2.290352	1.155401	0.000000e+00
1840	5.819265	2.140994	1.079805	0.000000e+00
1888	6.480643	2.076911	1.047256	0.000000e+00
1936	7.015334	2.068701	1.042899	0.000000e+00
1984	7.001244	2.230899	1.124445	0.000000e+00

Figure 2: Timing results for MM with $b = 16$

Between these two block sizes we see an improvement in the time for all matrix sizes. Next I run the MM with block size of 32.

Dimension	Time	Gflop/s	GB/s	Error
32	0.953630	2.097278	1.048639	0.000000e+00
64	0.946207	2.113871	0.792702	0.000000e+00
96	0.948575	2.109767	0.703256	0.000000e+00
128	0.956901	2.090794	0.653373	0.000000e+00
160	0.965488	2.078782	0.623635	0.000000e+00
192	0.970672	2.070853	0.603999	0.000000e+00
224	0.973006	2.056120	0.587463	0.000000e+00
256	0.986116	2.041611	0.574203	0.000000e+00
288	1.012819	1.981184	0.550329	0.000000e+00
320	1.013523	2.004509	0.551240	0.000000e+00
352	1.014953	1.976696	0.539099	0.000000e+00
384	1.005084	2.028121	0.549283	0.000000e+00
416	1.048779	1.922003	0.517462	0.000000e+00
448	1.106680	1.949948	0.522308	0.000000e+00
480	1.151425	1.920960	0.512256	0.000000e+00
512	1.062792	2.020606	0.536723	0.000000e+00
544	1.177589	1.913951	0.506634	0.000000e+00
576	1.210046	1.895164	0.500113	0.000000e+00
608	1.178665	1.906866	0.501807	0.000000e+00
640	1.050058	1.997177	0.524259	0.000000e+00
672	1.273525	1.906297	0.499268	0.000000e+00
704	1.114211	1.878892	0.491074	0.000000e+00
736	1.274276	1.877246	0.489716	0.000000e+00
768	1.362312	1.995071	0.519550	0.000000e+00
800	1.097213	1.866547	0.485302	0.000000e+00
832	1.223456	1.882963	0.488846	0.000000e+00
864	1.386897	1.860189	0.482271	0.000000e+00
896	1.455424	1.976944	0.511887	0.000000e+00
928	1.717644	1.861105	0.481320	0.000000e+00
960	1.904845	1.857865	0.479948	0.000000e+00
992	2.097272	1.861831	0.480473	0.000000e+00
1024	1.106600	1.940615	0.500315	0.000000e+00
1056	1.274791	1.847493	0.475869	0.000000e+00
1088	1.367212	1.884000	0.484853	0.000000e+00
1120	1.534542	1.831071	0.470847	0.000000e+00
1152	1.561447	1.958215	0.503152	0.000000e+00
1184	1.846529	1.797749	0.461584	0.000000e+00
1216	1.940766	1.852923	0.475421	0.000000e+00
1248	2.086198	1.863452	0.477808	0.000000e+00
1280	2.101517	1.995846	0.511436	0.000000e+00
1312	2.387247	1.892057	0.484551	0.000000e+00
1344	2.587632	1.876399	0.480269	0.000000e+00
1376	2.751921	1.893431	0.484366	0.000000e+00
1408	2.772264	2.013740	0.514877	0.000000e+00
1440	3.180852	1.877474	0.479799	0.000000e+00
1472	3.444351	1.852022	0.473071	0.000000e+00
1504	3.696549	1.840675	0.469960	0.000000e+00
1536	3.678005	1.970568	0.502905	0.000000e+00
1568	4.252156	1.813256	0.462565	0.000000e+00
1600	4.428946	1.849650	0.471661	0.000000e+00
1632	4.709162	1.846064	0.470565	0.000000e+00
1664	4.647708	1.982673	0.505200	0.000000e+00
1696	5.278368	1.848451	0.470832	0.000000e+00
1728	5.846556	1.765067	0.449438	0.000000e+00
1760	6.096243	1.788569	0.455272	0.000000e+00
1792	8.534909	1.348482	0.343140	0.000000e+00
1824	9.842765	1.233069	0.313675	0.000000e+00
1856	8.790111	1.454687	0.369942	0.000000e+00
1888	9.377787	1.435277	0.364901	0.000000e+00
1920	7.102360	1.993109	0.506582	0.000000e+00
1952	8.072085	1.842823	0.468258	0.000000e+00
1984	8.448028	1.848841	0.469665	0.000000e+00

Figure 3: Timing results for MM with $b = 32$

In this case, increasing the block size does not result in a decrease in the MM timing. I tried increasing the block size once more to 64 and again saw a decrease in the time it takes to do the MM. It appears that a block size of $b = 16$ is the best for performance. Comparing similar matrix sizes to MMulti0 we see a significant speed up. For example, a matrix of size $\sim 580 \times 580$ ran with MMulti0 takes about 9 seconds whereas MMulti1 takes about 1 second for a block size of 16.

Next I used OpenMP to parallelize the for-loop. Below are the timings for MM with $b = 16$ and OpenMP used. In this case the first matrix of size 16×16 takes longer with OMP, however, there is a significant speed up for higher values.

Dimension	Time	Gflop/s	GB/s	Error
16	5.946140	0.336353	0.336353	0.000000e+00
64	0.296076	6.755558	4.222224	0.000000e+00
112	0.263381	7.595907	4.340518	0.000000e+00
160	0.253538	7.916131	4.353872	0.000000e+00
208	0.266889	7.552789	4.066886	0.000000e+00
256	0.210694	9.555402	5.076307	0.000000e+00
304	0.212490	9.519513	5.010270	0.000000e+00
352	0.215037	9.329806	4.876944	0.000000e+00
400	0.228101	8.978479	4.668809	0.000000e+00
448	0.216511	9.967020	5.161492	0.000000e+00
496	0.238289	9.217508	4.757424	0.000000e+00
544	0.302715	7.445447	3.832216	0.000000e+00
592	0.256967	8.073982	4.146099	0.000000e+00
640	0.210296	9.972382	5.110846	0.000000e+00
688	0.278149	9.366510	4.792168	0.000000e+00
736	0.276865	8.640058	4.413942	0.000000e+00
784	0.349027	8.284006	4.226534	0.000000e+00
832	0.225757	10.204430	5.200334	0.000000e+00
880	0.273585	9.963587	5.072372	0.000000e+00
928	0.322504	9.912172	5.041536	0.000000e+00
976	0.412990	9.004714	4.576166	0.000000e+00
1024	0.211058	10.174851	5.166916	0.000000e+00
1072	0.246808	9.982863	5.065931	0.000000e+00
1120	0.282912	9.931908	5.036896	0.000000e+00
1168	0.324297	9.826879	4.980747	0.000000e+00
1216	0.361895	9.936836	5.033792	0.000000e+00
1264	0.409934	9.852746	4.988732	0.000000e+00
1312	0.465110	9.711265	4.914848	0.000000e+00
1360	0.528830	9.513288	4.812604	0.000000e+00
1408	0.572604	9.749528	4.930159	0.000000e+00
1456	0.649749	9.500982	4.802694	0.000000e+00
1504	0.734988	9.257490	4.677987	0.000000e+00
1552	0.816597	9.155823	4.625106	0.000000e+00
1600	0.904355	9.058390	4.574487	0.000000e+00
1648	0.991383	9.029426	4.558545	0.000000e+00
1696	1.093549	8.922145	4.503158	0.000000e+00
1744	1.229391	8.629376	4.354272	0.000000e+00
1792	1.291404	8.912138	4.495855	0.000000e+00
1840	1.399077	8.905162	4.491299	0.000000e+00
1888	1.581467	8.510907	4.291517	0.000000e+00
1936	1.698398	8.544892	4.307756	0.000000e+00
1984	1.780758	8.771020	4.420877	0.000000e+00

Figure 4: Timing results for MM with $b = 16$ and OpenMP

Finally I run MM with a block size of $b = 32$ and OpenMP. The inital matrix of size 32×32 is much faster however at higher values there is not a significant speed up.

Dimension	Time	Gflop/s	GB/s	Error
32	1.384263	1.444832	0.722416	0.000000e+00
64	0.433639	4.612497	1.729686	0.000000e+00
96	0.285591	7.007479	2.335826	0.000000e+00
128	0.209427	9.553128	2.985353	0.000000e+00
160	0.304156	6.598719	1.979616	0.000000e+00
192	0.250807	8.014610	2.337594	0.000000e+00
224	0.212462	9.416354	2.690387	0.000000e+00
256	0.187011	10.765495	3.027795	0.000000e+00
288	0.238189	8.424324	2.340090	0.000000e+00
320	0.218054	9.317032	2.562184	0.000000e+00
352	0.198199	10.122420	2.760660	0.000000e+00
384	0.212455	9.594652	2.598552	0.000000e+00
416	0.222996	9.039428	2.433692	0.000000e+00
448	0.222487	9.699306	2.598028	0.000000e+00
480	0.216006	10.239716	2.730591	0.000000e+00
512	0.197198	10.889987	2.892653	0.000000e+00
544	0.239923	9.394050	2.486660	0.000000e+00
576	0.232952	9.844241	2.597786	0.000000e+00
608	0.215846	10.412781	2.740206	0.000000e+00
640	0.192523	10.892995	2.859411	0.000000e+00
672	0.254707	9.531405	2.496320	0.000000e+00
704	0.217669	9.617731	2.513725	0.000000e+00
736	0.235051	10.177066	2.654887	0.000000e+00
768	0.266447	10.200561	2.656396	0.000000e+00
800	0.215530	9.502157	2.470561	0.000000e+00
832	0.229975	10.017269	2.600637	0.000000e+00
864	0.247728	10.414205	2.699979	0.000000e+00
896	0.266629	10.791371	2.794194	0.000000e+00
928	0.327794	9.752207	2.522123	0.000000e+00
960	0.351418	10.070469	2.601538	0.000000e+00
992	0.378431	10.318304	2.662788	0.000000e+00
1024	0.203465	10.554560	2.721098	0.000000e+00
1056	0.241832	9.738857	2.508493	0.000000e+00
1088	0.256387	10.046636	2.585531	0.000000e+00
1120	0.274395	10.240187	2.633191	0.000000e+00
1152	0.286552	10.670481	2.741721	0.000000e+00
1184	0.347704	9.547187	2.451305	0.000000e+00
1216	0.366138	9.821683	2.520037	0.000000e+00
1248	0.392243	9.911025	2.541288	0.000000e+00
1280	0.403223	10.401946	2.665499	0.000000e+00
1312	0.485520	9.303029	2.382483	0.000000e+00
1344	0.508742	9.543995	2.442808	0.000000e+00
1376	0.542282	9.608600	2.458014	0.000000e+00
1408	0.561823	9.936615	2.540612	0.000000e+00
1440	0.658612	9.067506	2.317251	0.000000e+00
1472	0.698113	9.137507	2.334037	0.000000e+00
1504	0.729356	9.328975	2.381866	0.000000e+00
1536	0.747150	9.700538	2.475658	0.000000e+00
1568	0.862135	8.943199	2.281428	0.000000e+00
1600	0.921537	8.889497	2.266822	0.000000e+00
1632	0.956815	9.085786	2.315985	0.000000e+00
1664	0.990136	9.306687	2.371415	0.000000e+00
1696	1.111965	8.774380	2.234983	0.000000e+00
1728	1.155609	8.929976	2.273836	0.000000e+00
1760	1.193913	9.132619	2.324667	0.000000e+00
1792	1.250959	9.200278	2.341142	0.000000e+00
1824	1.379215	8.799794	2.238544	0.000000e+00
1856	1.446098	8.842319	2.248693	0.000000e+00
1888	1.490948	9.027624	2.295159	0.000000e+00
1920	1.514737	9.345369	2.375281	0.000000e+00
1952	1.676868	8.870959	2.254096	0.000000e+00
1984	1.727006	9.044012	2.297471	0.000000e+00

Figure 5: Timing results for MM with $b = 32$ and OpenMP

Special Functions, Taylor Series, and Approximations

Timings for sine vectorization

```
Reference time: 4.1983
Taylor time:    1.4400      Error: 6.927903e-12
Intrin time:    1.4201      Error: 6.927903e-12
Vector time:    1.5428      Error: 6.927903e-12
```

Figure 6: Timing results for vectorized sine approximation