Performance of Scalar Add

The Program code was run on Stampede supercomputer and the following results were obtained:

	L=1	L=2	L=3	L=4	L=5	L=6
K=1	2.598279	2.584171	2.583644	2.583283	2.582152	2.58057
K=2	0	1.30411	0	1.30207	0	1.300478
K=3	0	0	0.885915	0	0	0.885639
K=4	0	0	0	0.886038	0	0
K=5	0	0	0	0	0.886641	0
K=6	0	0	0	0	0	0.887059

where K corresponds to the number of accumulator variables and

L corresponds to the unrolling factor.

Using the Architecture manuals we found that the latency for the addition takes 3 cycles and cycles per issue is 1.

Therefore theoretically we know that the optimal loop unroll factor should be

Ceil(latency/issues per cycle).

Hence it is 3.

We can also infer from the observed values that K=3 and L=3 gives a better performance.

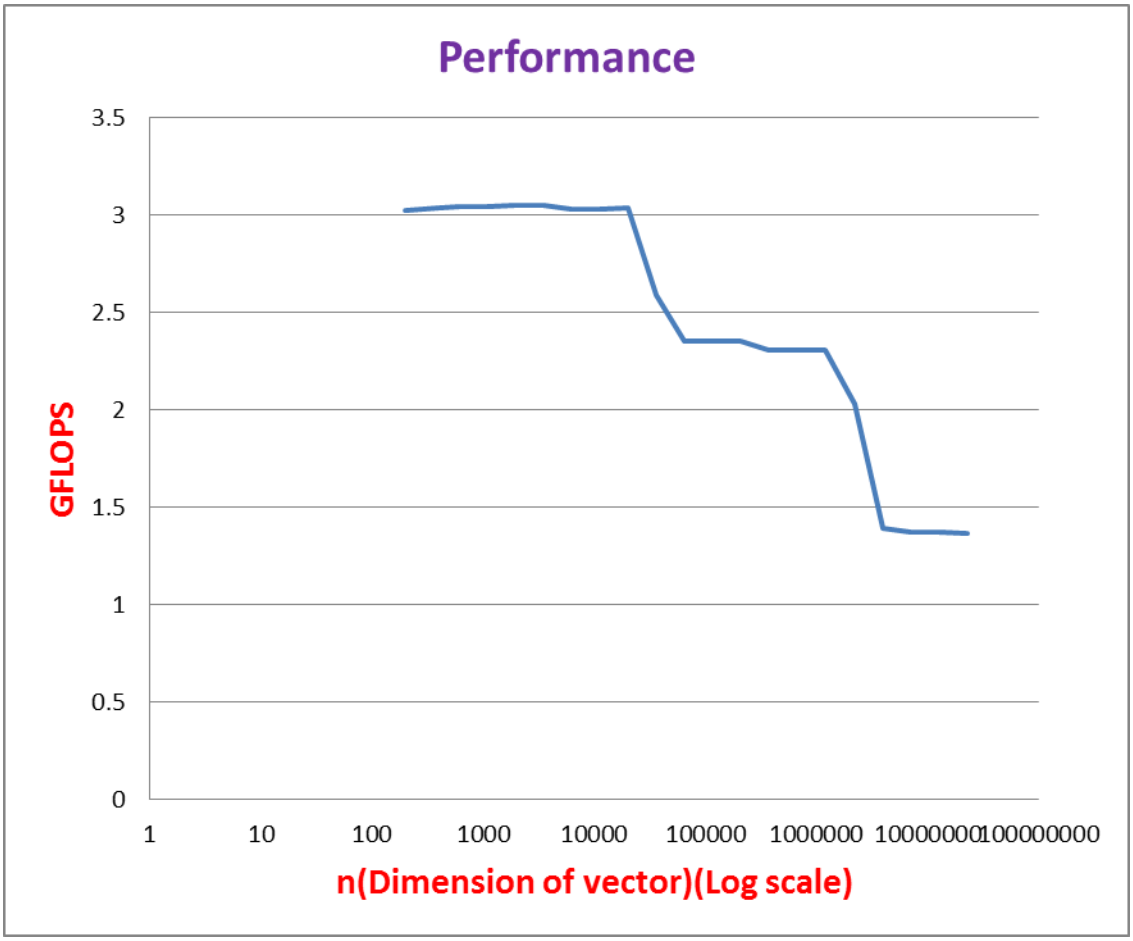
Part b

For the values K=3 and L=3 we ran the code for vectors of sizes n= 100.ceil(1.8^i)

The following results were obtained:

n	cycles/I	I/cycles	GFLOPS
200	0.893574	1.1191015	3.02157404
400	0.889108	1.12472276	3.03675144
600	0.887674	1.1265397	3.04165718
1100	0.886334	1.12824285	3.0462557
1900	0.885625	1.12914608	3.04869442
3500	0.885271	1.1295976	3.04991353
6200	0.890166	1.12338598	3.03314213
11100	0.890105	1.12346296	3.03335
19900	0.888254	1.12580411	3.03967109
35800	1.043652	0.9581738	2.58706925
64300	1.147102	0.87176206	2.35375756
115700	1.148406	0.87077218	2.3510849
208300	1.147954	0.87111504	2.35201062
374900	1.169881	0.85478779	2.30792705
674700	1.169379	0.85515474	2.30891781
1214400	1.169613	0.85498366	2.30845587
2186000	1.329387	0.7522264	2.03101129
3934700	1.936782	0.51632037	1.394065
7082400	1.973069	0.50682465	1.36842655
12748300	1.973448	0.50672731	1.36816374
22946900	1.974339	0.50649863	1.3675463

The plot with vector size in x-axis and GFLOPS in y-axis is



Performance counter stats for './scalaradd res' (for n=2000)

7,236,204,345 cycles # 0.000 GHz

9,003,952,491 instructions # 1.24 insns per cycle

4,921,954,045 L1-dcache-loads

137,934 L1-dcache-load-misses # 0.00% of all L1-dcache hits

The stampede supercomputer uses (in compute node)

Intel(R) Xeon(R) CPU E5-2680 0 @ 2.70GHz

It uses the Sandy bridge architecture.

It has 32KB L1 data cache and 32KB instruction cache

256 KB L2 cache

64 B cache line size.

The L1 cache latency is 3 cycles and L2 is 8 cycles.

Based on the graph we observe that there is distinct fall in the performance

When n=19900 to 35800 and then 1214400 to 2186000

And when n=3934700.

In the case of n=19900 the entire data fits in the cache but not in the case of n=35800 and hence the performance decreases.

Similarly with respect to the L2 cache he second discontinuity can be explained!!!!!!

When I initially ran I didn't have any printf statement because of which I think there was dead code elimination because of which the following results were obtained.

N	GFLOPS
200	0.506066
400	0.398959
600	0.363369
1100	0.329795
1900	0.313474
3500	0.302979
6200	0.297661
11100	0.294602
19900	0.292869
35800	0.291889
64300	0.291366
115700	0.291050
208300	0.290886
374900	0.290795

674700	0.290769
1214400	0.290708
2186000	0.290728
3934700	0.290685
7082400	0.290678
12748300	0.290694
22946900	0.290685