LS2-Scalaradd:

(a)

- When the loop is unrolled, there still exist sequential dependencies (because the result of every addition is being put in the same variable!).
- To overcome these dependencies, we increase the number of accumulators. And, as expected the performance is better.
- This is because; the more number of independent instructions available, the pipeline of the corresponding execution unit can be filled (in this case, the pipeline of the floating point unit!).
- The following results are based on running the code on a compute node on Trestles.
- Each node has 4 sockets; each socket has 8 core 2.4GHz AMD – 6136 [K10] Magny-Cours processors.
- I checked the assembly file generated by the compiler (gcc) and found that the instructions being used for the floating point add was ADDSD.

Latency = 4cycles; cycles/issue = 1.

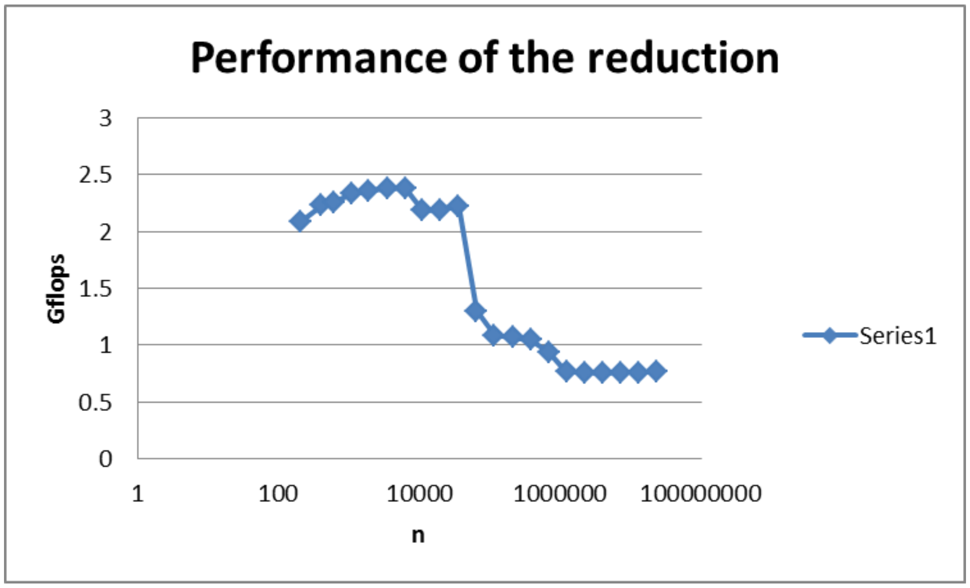
[Source: http://www.agner.org/optimize/instruction_tables.pdf]

Accumulators/Unrolling factor	1	2	3	4	5	6
1	4.017137	4.020056	4.017092	4.020903	4.021047	4.017824
2	0	2.016405	0	2.01686	0	2.016865
3	0	0	1.349423	0	0	1.351381
4	0	0	0	1.016129	0	0
5	0	0	0	0	1.017868	0
6	0	0	0	0	0	1.019907

Maximal Performance for Accumulators (K) = 4, Unrolling (L) = 4

K = 4 is enough to fill the pipeline of the execution unit.

(b)



- The plateaus in the above graph correspond to different performance levels of the processor.
- As the size of the vector (n) increases, it can be observed that the performance reduces.
- Hence, this reduction in performance can be attributed to the performance of the memory hierarchy of the processor.
- The **first plateau**, where the performance is almost equal to the theoretical peak

  [i.e. 1flop/cycle * 2.4GHz = **2.4 Gflops**], for i = 1 through 7 which means n = 200 to 6200 corresponds to the case where all the elements of the vector can fit in the **L1 cache** !!
- The **second plateau** after the 'small' drop corresponds to the case when the vector size spills the data to **the L2 cache.** n = 11100, 19900, 35800.
- After the big drop, the **third plateau** corresponds to the case when the data spills to the **L3 cache**. n = 64300 to 674700
- The **fourth plateau** corresponds to the case when there are **DRAM accesses**.

  n = 674700 to 22946900.
- The big drop after the plateau corresponding to L2 cache can be attributed to **greater number of misses in the TLB** with increase in size of n. [virtual address translation]

- **Note:** I could have given figures of number of L1, L2, L3 cache misses, but there is no ‘perf’ on Trestles and I was taking time to get started with ‘papi’ !!