Table of Contents

Compilers optimization for SMatrix usage

SMatrix is a highly optimized C++ library for "small" matrix manipulation. Originally developed for the Hera-B experiment is now part of Root-Math package. It is known to overperform other implementations such as TMatrix and CLHEP-Matrix.

Its is extensively used in CMS in particular in fitting code such as track Kalman filter and constrained kinematic fits.

SMatrix has been already the subject of performance investigations w.r.t. different compiler optimization strategies. See for instance Open Lab report and the Workshop with Intel.

Here we try to look in details to few performance-critical Matrix operations that shows up as hot-spots in CMS reconstruction.

Similarity

The Similarity operation between a generic matrix and a symmetric (actually positive-defined) matrix is a very typical operation when covariance matrices are involved. It can be decomposed in a transposition and two matrix multiplications. The result is a symmetric matrix as well. in SMatrix, symmetric matrices are stored using a lower-triangular representation which saves space and improve performances in several operations that act on each element sequencially. This representation seems on the other hand to prevent some compiler optimization in case of multiplications due to the non sequential access pattern to the elements.

We have compared the performances of four different implementations:

- (**root**) the original Root one: both input and output are symmetric matrices in lower-triangular representation
- (**sym**) having in input the symmetric matrix represented as a full-square matrix and in output a lower-triangular one
- (**std**) having in both input and input square matrices
- (**loop**) a brute force loop implementation `B(i,j) = U(i,k)*A(k,l)*U(j,l)`

code can be found in hte CMSSW cvs repository as well as in `/afs/cern.ch/user/i/innocent/w1/Similarity`, in the latter together with all the executable ready to run.

we have tested **gcc 4.6** and **icc 12** compilers with various options on three different architectures: **Intel Core 2 Duo @3.06 GHz** *(my MacBook)*, **Intel Core i7-2600K CPU @ 3.40GHz** *(my workstation)*, **Intel Xeon L5520 (Nehalem) @ 2.27GHz** *(standard CERN node)* All results are expressed in **ticks** obtained using `rdtsc` instruction. in case of gcc 4.6 `O2v` stands for `"-O2 -ftree-vectorize"`, `O2f` for `-O2 -fast math` and `ul` ofr `-funroll-loops`. in case of icc `nv` stands for `-no-vec`.

Intel Core 2 Duo @3.06 GHz																	
		3x3				5x5				5x15				15x15			
compiler	code size	root	sym	std	loop	root	sym	std	loop	root	sym	std	loop	root	sym	std	loop
gcc 4.6. -O2	46.9	358	156	164	95	557	431	542	241	6439	3174	3317	2220	22501	12692	13617	136743
gcc 4.6. -O2f	47.2	182	164	180	77	541	421	512	250	3224	2270	2416	1158	12455	9335	10218	135901
gcc 4.6. -Ofast	63.7	128	134	141	31	318	416	479	93	3182	2249	2356	931	11986	9441	10078	49532
gcc 4.6. -Ofast ul	100.6	119	134	136	37	329	428	446	54	3110	2174	2276	939	11629	8585	9518	49651

Intel Xeon L5520 (Nehalem) @ 2.27GHz																
		3x3				5x5				5x15				15x15			
compiler	code size	root	sym	std	loop	root	sym	std	loop	root	sym	std	loop	root	sym	std	loop
gcc 4.6. -O2	43.1	165	140	153	378	524	410	458	1781	4191	2583	2664	4097	15077	10379	10969	112606
gcc 4.6. -O2v	44.3	159	127	143	344	508	391	451	1763	4159	2546	2654	4175	14856	10027	10917	112752
gcc 4.6. -O2f	43.3	158	137	147	409	523	424	458	1859	2577	1793	1932	3880	9660	7200	8018	111207
gcc 4.6. -O2vf	43.8	154	127	145	368	510	416	464	1887	2591	1774	1907	3910	9514	6997	8199	111390
gcc 4.6. -O3	74.1	112	108	136	644	356	353	439	1761	4132	2514	2648	4883	14915	9977	10951	70756
gcc 4.6. -Ofast	70.0	127	123	139	940	363	359	456	1813	2530	1711	1898	3746	9465	6951	9053	37833
gcc 4.6. -Ofast ul	108.7	117	112	132	825	362	356	446	1813	2518	1706	1834	3656	9248	6704	7332	37089
icc 12 -O2 nv	237.0	174	172	157	711	487	480	448	1815	2413	2708	3547	6752	10298	10588	14700	39199
icc 12 -O2	181.4	203	197	179	911	506	504	515	2521	3023	2773	3588	6681	12091	11446	12639	68379

Intel Core i7-2600K @ 3.40GHz running sse code																	
		3x3				5x5				5x15				15x15			
compiler	code size	root	sym	std	loop	root	sym	std	loop	root	sym	std	loop	root	sym	std	loop
gcc 4.6. -O2	43.1	131	129	132	266	425	346	386	1648	3258	1941	2011	3840	11393	7586	7764	93685
gcc 4.6. -O2v	44.3	142	113	137	724	384	331	394	1434	3212	1896	2011	2775	11396	7394	7809	94098
gcc 4.6. -O2f	43.3	125	120	128	857	402	352	391	1639	2075	1708	1808	3735	7725	6702	6995	93257
gcc 4.6. -O2vf	43.8	137	106	123	807	404	336	401	1508	1995	1708	1807	3718	7436	6567	7125	101895
gcc 4.6. -O3	74.0	96	99	113	130	287	272	376	1649	3232	1884	2038	3787	11350	7305	7825	70375
gcc 4.6. -Ofast	70.0	116	116	132	138	299	271	374	1491	2062	1667	1784	3747	7531	6552	7180	37713
gcc 4.6. -Ofast ul	108.7	117	108	136	627	271	267	348	1479	1949	1677	1754	3681	7276	6400	6889	37295
icc 12 -O2 nv	237.0	132	131	111	881	373	354	368	1839	2667	2663	3406	6313	8403	8735	10387	34587
icc 12 -O2	181.4	146	147	133	884	399	403	409	1952	2102	1925	2607	7285	8555	7897	9195	57647

Intel Core i7-2600K @ 3.40GHz running avx code																	
		3x3				5x5				5x15				15x15			
compiler	code size	root	sym	std	loop	root	sym	std	loop	root	sym	std	loop	root	sym	std	loop
gcc 4.6. -O2	43.1	134	122	128	864	392	318	376	1619	3473	2481	2565	5801	12563	9693	9939	93615
gcc 4.6. -O2v	46.6	138	119	137	827	369	299	374	1497	3400	2439	2558	5640	12150	9414	9960	93746
gcc 4.6. -O2f	43.3	119	115	119	248	413	348	384	1469	2049	1494	1761	1782	7482	5818	6503	101356
gcc 4.6. -O2vf	43.5	128	101	123	802	357	314	375	1529	1986	1458	1581	1783	7317	5542	6806	93257
gcc 4.6. -O3	74.0	115	103	123	144	250	244	379	1645	3470	2452	2569	6099	12258	9407	9994	68395

gcc 4.6. -Ofast	66.4	112	97	116	114	**259**	248	378	458	**1967**	1415	1548	2224	**7147**	5500	6102	32718
gcc 4.6. -Ofast ul	102.8	106	97	123	124	248	239	319	468	1948	1431	1648	2476	7081	5513	6128	35301
icc 12 -O2 nv	220.6	149	137	119	93	360	355	364	473	2186	2283	3036	6471	7583	8038	10183	35438
icc 12 -O2	178.4	136	139	151	105	443	424	445	517	2458	2131	2717	2327	9503	7663	9503	56308

-- VincenzoInnocente - 25-Mar-2011

This topic: LCG > VISMatrixBenchMarks
Topic revision: r4 - 2011-03-26 - VincenzoInnocente