week4实验记录

zxp

March 23, 2024

1 environment

cpu:Intel(R) Xeon(R) Gold 6330 CPU @ 2.00GHz (使用时申请了16个核心)
System:CentOS7
Compiler:13.2.0 和icpx

2 code

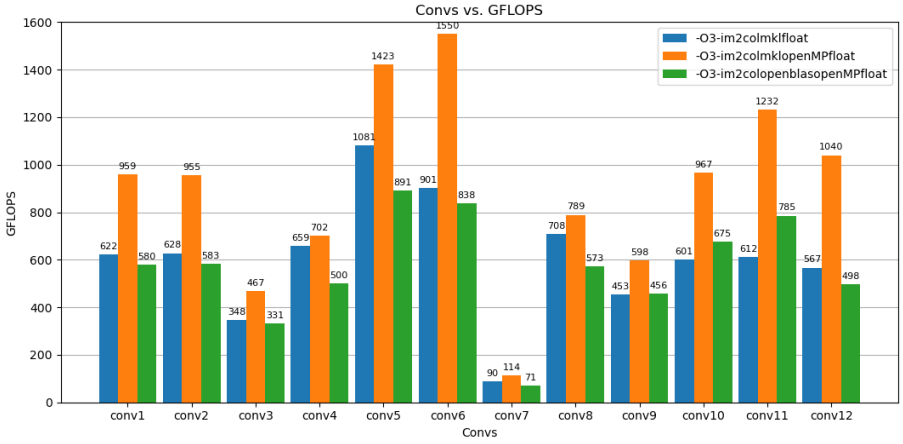

Figure 1: O3

3 Experiment

将上周测试的im2col在服务器上面测试了一遍，申请了16个核心，测试MKL是使用英特尔的编译器icpx，测试openblas使用的是gcc13，批次设置为128.
英特尔编译器的使用：英特尔在2021年的时候推出了oneapi，把他老的编译器（c编译器icc，c++编译器ipcc）放弃了，在oneapi里面弄了一套新的编译器（c编译器icx，c++编译器icpx）。按英特尔教程，使用的时候运行oneapi里面的一个脚本就会设置环境变量，然后编译的时候指定cmake使用ipcx编译就可以成功使用英特尔的编译器。
英特尔的编译器编译时并不需要手动链接MKL，在编译选项中添加-qmkl=sequential会调用串行的MKL库，-qmkl=parallel会调用并行的MKL库。用英特尔的编译器编译libtorch的时候编译第三方库（fmt）的时候会报很多警告并且停止，在git上找到一个问题是有一样的情况，按照那个帖子这是英特尔编译器某些优化选项带来的问题，关掉这些选项就行，但还没尝试。
在服务器上使用openblas还是上周的问题，并行版本的openblas不对，尚未解决。

3.1 Analysis

mkl特别快，快的异常，conv5和conv6甚至超过了roofline（马吉祥测出来的是1200），我觉得有问题又测了一遍，结果一样，然后考虑到启用openmp有可能是没设置并行的线程的影响，设置环境变量export OMP_NUM_THREADS=16，但结果还是一样，但我又在服务器上用英特尔的编译器测试了正确性，正确性也没问题。但这是在服务器上的结果，在我自己电脑里就没出现这个问题，在我自己电脑里用英特尔的编译器跑出来的结果和用gcc是一样的，并且都比roofline低（roofline900，mkl600）。完全不知道为什么。

4 Experiment2

测试了im2col的内存，使用的是openblas库，在批次纬度的循环加上openmp。

- param1:Percent of CPU this job got
- param2:Maximum resident set size (kbytes)
- param3:Major (requiring I/O) page faults
- param4:Minor (reclaiming a frame) page faults
- param5:Voluntary context switches
- param6:Involuntary context switches

conv	param1	param2	param3	param4	param5	param6	gflops	layout
conv1	607	823804	0	8406	159	88	580	im2col
	781	430596	0	2664	23	135	350	im2win
conv2	561	853460	0	11613	238	85	583	im2col
	586	445976	0	2464	46	108	366	im2win
conv3	613	1461360	0	8099	179	107	331	im2col
	575	738116	0	2306	46	118	393	im2win
conv4	507	20700948	0	30258	258	1122	500	im2col
	739	7458572	0	7213	185	1904	341	im2win
conv5	898	605864	0	5263	227	82	891	im2col
	1043	199256	0	3964	48	115	644	im2win
conv6	774	205804	0	4860	219	50	838	im2col
	1027	104004	0	3637	94	101	584	im2win
conv7	603	2550836	0	4994	232	214	71	im2col
	397	1877944	0	2681	47	115	295	im2win
conv8	566	4795636	0	12893	314	384	573	im2col
	724	2361032	0	3402	93	538	382	im2win
conv9	470	1080732	0	6773	203	76	456	im2col
	565	485940	0	2209	45	94	351	im2win
conv10	521	522640	0	4543	221	53	675	im2col
	931	235620	0	2688	241	133	505	im2win
conv11	640	253312	0	4747	178	51	785	im2col
	772	113420	0	3891	47	59	552	im2win
conv12	728	130992	0	3848	206	49	498	im2col
	807	65436	0	2578	51	63	460	im2win

Table 1: memory-analyze

4.1 Analysis

gflops没法比，openblas肯定是使用了avx512的，不知道怎么指定不使用avx512，从内存上看im2win是远比im2col小的，从cpu的使用率来看im2win的也都是更高，im2win除了被迫中断次数都是更好的。