week-4

JX-Ma

2024/3/23

1 本周工作

1. 测试了 bchw 和 bhwc 的 gflops
2. 对上周内存进行了内存分析
3. 使用 perf 工具测试了 cache-miss
4. 获取了计算节点 gpu 的信息

2 实验部分

本次实验是在申请的服务器上跑的。

2.1 实验环境

- 系统: CentOS7
- gcc version : 13.2.0
- 优化选项: -O3 -fopenmp -avx2 -fmadd
- cpu: Intel(R) Xeon(R) Gold 6330 CPU @ 2.00GHz

2.2 实验结果

从实验结果可以看出，在服务器上的结果时 bhwc 的除了 conv8 其他都比 bchw 的块，所以接下来在 im2win 的代码以 bhwc 的布局为主。

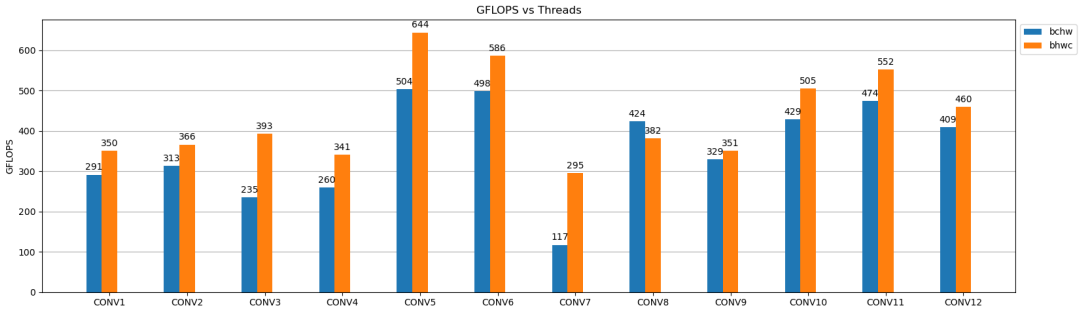

图 1: bchw vs bhwc-gflops

3 perf 工具使用

使用 perf 工具测试了各级 cache 的引用和 cache-miss 次数，从 CONV5 分析看到 im2win 的 cache-miss 数明显比 direct 高，但是实际上 im2win 的性能要比 direct 好，我感觉可能是自己的电脑不稳定造成的，因为我在测试 gflops 时，因为是连续跑很多次，有的时候测出的第一次明显低于平均水平。而我在测试 cache 命中次数的时候只测一次这样偶然性会很高，但是这周在服务器上测量 gflops 时输出的结果很稳定，不会出现个别数据偏低的情况，最大最小值之间相差不会超过 10，下周打算先看 gpu 的相关知识，代码都有，后面有时间就整理到服务器上在服务器上跑，对于 perf 工具，我这周尝试过可以在服务器上测试 cache-miss 和 cache-reference 这些参数，但是后面详细的各级缓存就需要 sudo 权限了。

4 内存分析

DIMENTION	param1	param2	param3	param4	param5	param6	gflops	layout
Batch	706	309900	0	87763	34	135	389	direct
	841	438664	3	119947	40	1212	153	im2win
Channel	717	310152	0	87847	39	174	386	direct
	722	438536	4	120029	45	126	338	im2win
Height	1062	310024	1	467675	180	916	120	direct
	1208	438920	0	491857	695	3150	122	im2win
Batch+Channel	716	309900	0	87825	53	167	390	direct
	481	438668	10	119955	80	116	160	im2win
channel+height	732	310408	0	87947	53	104	350	direct
	768	439304	3	120127	68	457	329	im2win

表 1: CONV1-memory-analyze

从总体 gflops 上看，直接卷积的性能较好，影响性能的主要因素是次要缺页的次数，次要缺页次数 im2win 上要比 direct 多 3w 次左右。他们在 Height 上的表现都只有最高性能的一半，这里可能得原因是 CONV1 的输出张量的 Height 维度只有 55, 而我们使用的线程数是 16，这里可能发生较多的进程切换，而且 height 维度相比 batch 和 channel 较小，无法发挥 Openmp 的全部性能。从程序自己中断的次数来说，在 height 上的相比其他维度都比较高。再来看看二者在 batch 维度上面性能表现，首先次要缺页次数是影响元素之一这不用说，但是在 batch 上面真正限制 IM2win 的性能的是程序非自愿中断次数，中断次数是直接卷积的 10 倍，性能也不到直接卷积的 50%。然后在 Batch+Channel 维度上面 im2win 性能同样没有直接卷积的 50%, 这里影响的元素应该是 cpu 利用率不充分，im2win 利用率比直接卷积少 30%。

DIMENTION	param1	param2	param3	param4	param5	param6	gflops	layout
Batch	737	317832	0	90468	42	225	370	direct
	912	451592	0	123822	45	383	149	im2win
Channel	739	318472	1	90594	36	530	359	direct
	743	451720	1	123945	39	339	344	im2win
Height	975	317960	4	470230	40	395	123	direct
	1243	451336	0	504814	442	3376	122	im2win
Batch+Channel	716	317960	0	90535	96	667	370	direct
	900	451724	0	123867	62	400	154	im2win
channel+height	774	318604	0	90684	56	468	346	direct
	721	452104	1	124069	54	555	339	im2win

表 2: CONV2-memory-analyze

在 CONV2 上面直接卷积略高于 Im2win, 总体来说 CONV2 的情况和 CON1 的情况很像，但是 CONV2 在 Batch+Channel 上面影响最大的应该是次要缺页次数，CONV1 上面是 CPU 的利用率。在 batch 维度上 CONV1 上面是程序非自愿中断次数。CONV1 和 CONV2 情况一样的原因，从 benchmark 维度来说首先他们卷积核大小相同，且步长也一样。并且二者在实现卷积方法时，对输出张量分块大小接近。因此从 CONV1 和 CONV2 可以总结 gflops 规律，首先单个来看，直接卷积在 Batch,Channel,Batch+channel 这三种加 openmp 的方式都能取得较好的 gflops，而 channel+height 要差一点，在 height 上是最差的，性能都不到最高性能的一半。而 im2win 只有在 channel 和 channel+height 性能较好。总的来看，IM2win 涉及到 batch(batch | batch+channel) 上添加 openmp 性能都不到直接卷积的一半。

DIMENTION	param1	param2	param3	param4	param5	param6	gflops	layout
Batch	497	957320	0	249313	53	497	305	direct
	1292	17051044	0	4562748	44	5455	126	im2win
Channel	509	956808	0	249370	40	250	299	direct
	1168	17050828	10	4563471	136	3285	277	im2win
Height	886	956680	0	1024213	76	453	73	direct
	1403	17051144	0	16976644	505	9759	73	im2win
Batch+Channel	505	957320	0	249371	58	359	308	direct
	1279	17051660	0	4562855	64	6122	133	im2win
channel+height	526	957320	0	249474	56	248	290	direct
	1147	17051528	0	4563486	124	4167	271	im2win

表 3: CONV3-memory-analyze

CONV3 的数据存在问题是错误的数据，因为在跑 CONV3 的时候，IM2WIN 的输入张量的 batch*16，忘了改回来，所以导致 IM2win 的数据都很大。

DIMENTION	param1	param2	param3	param4	param5	param6	gflops	layout
Batch	1153	2464776	11	1008548	65	3992	170	direct
	1471	6328584	0	1974460	73	11273	73	im2win
Channel	958	2464520	0	1008545	52	1980	373	direct
	1390	6328456	0	1974460	238	2763	174	im2win
Height	1058	2464776	0	1316390	144	6220	299	direct
	1478	6328584	0	2260323	466	8914	86	im2win
Batch+Channel	1166	2465288	0	1008608	69	3247	171	direct
	1453	6328712	0	1974521	80	10355	73	im2win
channel+height	965	2465292	0	1008652	58	2317	371	direct
	1385	6328840	0	1974572	354	4729	173	im2win

表 4: CONV4-memory-analyze

CONV4 的 im2win 的内存消耗比直接卷积多那么多的原因是因为我在 CONV4 上面对输出张量做了填充，也就是相当于生成了一个中间输出变量，所以内存开销相比其他的较大。gflops 直接卷积要大好多，中断次数非自愿中断次数和缺页次数 im2win 都比较多。

DIMENTION	param1	param2	param3	param4	param5	param6	gflops	layout
Batch	839	168332	1	40453	50	382	396	direct
	797	256648	0	62337	49	301	372	im2win
Channel	917	168584	0	49236	35	164	381	direct
	853	256140	0	72693	121	177	434	im2win
Height	1102	168840	0	119062	77	687	212	direct
	1209	256264	4	144832	611	1576	233	im2win
Batch+Channel	877	168584	0	40511	54	334	395	direct
	790	256652	0	62393	95	1327	366	im2win
channel+height	946	169352	0	49832	75	282	373	direct
	863	256776	8	73086	165	711	424	im2win

表 5: CONV5-memory-analyze

从 CONV5 开始 Im2win 的 gflops 都会较高于直接卷积，但是从内存参数上来看，直接卷积缺页次数比 im2win 少，其他方面如中断次数也少，但是性能确比 Im2win 差，猜测可能是因为现在的 cpu 有多级缓存，im2win 数据连续性比较好，所以它就算是发生缺页中断也有可能是从比较快的缓存中发生了错误，所以还需要更精确的内存分析器去分析缺页失败次数。