发件人: "陈明" <chenming@sinovoice.com.cn>
发送时间: 2014-04-02 11:04:17 (星期三)
收件人: "'Dong Wang'" <wangdong99@mails.tsinghua.edu.cn>, 'zhangzy' <zhangzy@cslt.riit.tsinghua.edu.cn>, "郑晓明" <zhengxiaoming@sinovoice.com.cn>, "'王晓曦'" <349543593@qq.com>, "'刘超'" <cbmixx@gmail.com>
抄送:

主题: 识别过程所花时间的测试

		125/model2		170/model1		170/model2	
huawei3	feat_compute	23939.502	1.05%	12119.347	1.32%	12174.751	1.34%
	nnet_forward	1605914.693	70.54%	545436.990	59.19%	544983.595	60.18%
	decode	646804.812	28.41%	363946.986	39.49%	348364.420	38.47%
	total	2276659.007		921503.323		905522.766	
fuce	feat_compute	6181.848	1.00%	3127.806	1.15%	3125.324	1.26%
	nnet_forward	410500.537	66.18%	138432.910	50.79%	137869.653	55.77%
	decode	203627.007	32.83%	130983.423	48.06%	106215.543	42.97%
	total	620309.392		272544.139		247210.520	
hubei	feat_compute	8719.832	0.97%	4451.181	1.08%	4397.785	1.22%
	nnet_forward	574203.973	64.08%	194679.447	47.22%	192111.043	53.09%
	decode	313138.244	34.95%	213144.175	51.70%	165319.170	45.69%
	total	896062.049		412274.803		361827.998	
bjyd	feat_compute	9719.538	0.97%	4987.841	1.10%	4968.370	1.23%
	nnet_forward	653455.148	65.20%	225924.919	50.01%	220670.096	54.78%
	decode	339080.592	33.83%	220855.606	48.89%	177189.582	43.99%
	total	1002255.278		451768.366		402828.048	

以上是识别过程中不同步骤所花时间的测试结果。

测试的 8K 数据, beam=9.0/max_active=5000/acoustic_scale=12

共四个测试集：huawei3, fuce, hubei, bjyd

在两个不同的机器上 120(E5606 @ 2.13GHz) 170(E5-2690 @2.90Hz), 前一台机器不支持 AVX，后一台机器支持 AVX，MKL 库应该分到不同的处理库上了。

170 上还测试了两个不同的模型。

基本上可以看到,在使用了 AVX 加速情况下,基本上 nnet_forward 步骤和 decode 步骤时间差不大，但也是 forward 时间略高，

如果在没有 AVX 加速情况下，则 nnet_forward 花的时间就更大了。华为的测试环境，E5620　@ 2.40GHz，也是不支持 AVX 的。

如果考虑到流式情况下，要冗余计算一些帧，可能花的时间会更大。

昨天会议听您说感觉 decoder 花的时间可能更大,这个和目前我的测试结果不太一致？

另外，我前几天还测试了一下多线程情况下的识别速度问题。

线程数	1	2	4	8	10	12	14	16	20	24	28	32
no-affinity	0.128	0.063	0.034	0.02	0.017	0.017	0.017	0.018	0.02	0.021	0.021	0.022
set-affinity	0.128	0.064	0.034	0.019	0.017	0.017	0.017	0.016	0.017	0.017	0.017	0.015

这是在 170 上测的，就是一个测试集数据，开不同的线程数目并行处理，看最后总共完成的时间，也是用这个时间除以了总的声音长度。

170 机器是 2*8Core，因此总共是 16 个物理核，超线程超成了 32 核。

可以看到，在没有设置线程亲缘性的情况下，到 16 线后，时间反而增长了。

如果将 32 线程只绑定到 16 个物理核上，反而要快得多。

这个大概可以解释为使用超线程情况下，反而因为抢占，CPU cache 不能很好使用。

但我奇怪的是,为啥即使在设置亲缘性后,也是到了 8 线程之后,就不太下降了，这个我还没想太明白，难道线程切换的代价已经很大了？

因为华为的指标，是按照多少个 CPU 核，一小时可以批量处理多少小时语音数据作为标准的，因此多线程下的行为也需要考虑下。