基于tcmalloc对引擎内存占用进行分析

* 进程实际占用的物理内存为62G，tcmalloc监控到的物理内存语实际使用不一致性，但是本实验在进行前后内存的对比，因此不影响分析
* 引擎配置为纯中文模式，采用所有音频测试集进行验证

1. 对比heap.480 和 heap.500

|  |  |  |  |
| --- | --- | --- | --- |
|  | heap.480 | Heap.500 |  |
| currently in use | 51358.2 | 51638.9 | 280.7 |
| ArrMemBlock::ArrMemBlock | 238.7 | 292.9 | 54.2 |
| DeccoderInst::get\_nbest\_from\_lattice | 228.9 | 251.8 | 22.9 |
| DecContext::wPropagateTokenToNodeDNN | 83.2 | 152.6 | 69.4 |
| ArrMemBank::Malloc | 21.2 | 72.5 | 51.3 |
| FixElementBlock::FixElementBlock | 365.8 | 383.9 | 18.1 |
|  |  |  |  |

2.对比heap.500 和 heap.1500

|  |  |  |  |
| --- | --- | --- | --- |
|  | heap.500 | Heap.1500 |  |
| currently in use | 51638.9 | 52024.9 | 386 |
| ArrMemBlock::ArrMemBlock | 292.9 | 255.5 | -37.4 |
| DeccoderInst::get\_nbest\_from\_lattice | 251.8 | 206.1 | -45.7 |
| DecContext::wPropagateTokenToNodeDNN | 152.6 | 124.9 | -27.7 |
| ArrMemBank::Malloc | 72.5 | 43.2 | -29.3 |
| FixElementBlock::FixElementBlock | 383.9 | 664.4 | 280.5 |
|  |  |  |  |

3.对比heap.1000 和 heap.1500

|  |  |  |  |
| --- | --- | --- | --- |
|  | heap.1000 | Heap.1500 |  |
| currently in use | 51929.9 | 52024.9 | 95 |
| ArrMemBlock::ArrMemBlock | 284.8 | 255.5 | -29.3 |
| DeccoderInst::get\_nbest\_from\_lattice | 297.7 | 206.1 | -91.6 |
| DecContext::wPropagateTokenToNodeDNN | 152.6 | 124.9 | -27.7 |
| ArrMemBank::Malloc | 68.1 | 43.2 | -24.9 |
| FixElementBlock::FixElementBlock | 510.2 | 664.4 | 154.2 |
|  |  |  |  |

4.对比heap.1000 和 heap.1990

|  |  |  |  |
| --- | --- | --- | --- |
|  | heap.1000 | Heap.1990 |  |
| currently in use | 51929.9 | 52405.1 | 475.2 |
| ArrMemBlock::ArrMemBlock | 284.8 | 328.8 | 44 |
| DeccoderInst::get\_nbest\_from\_lattice | 297.7 | 137.5 | -160.2 |
| DecContext::wPropagateTokenToNodeDNN | 152.6 | 138.8 | -13.8 |
| ArrMemBank::Malloc | 68.1 | 101.8 | 33.7 |
| FixElementBlock::FixElementBlock | 510.2 | 742.5 | 232.3 |
| GPCreate | 161.0 | 520 | 359 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 内存大小（M） | heap.480 | Heap.500 | heap.1000 | Heap.1500 | heap.1800 | Heap.1900 | heap.2000 | Heap.2100 |
| currently in use | 51358.2 | 51638.9 | 51929.9 | 52024.9 | 52100.9 | 52288.0 | 52368.5 | 52194.1 |
| ArrMemBlock::ArrMemBlock | 238.7 | 292.9 | 284.8 | 255.5 | 275.3 | 341.2 | 320.7 | 272.4 |
| DeccoderInst::get\_nbest\_from\_lattice | 228.9 | 251.8 | 297.7 | 206.1 | 160.3 | 114.5 | 137.5 | 68.7 |
| DecContext::wPropagateTokenToNodeDNN | 83.2 | 152.6 | 152.6 | 124.9 | 138.8 | 97.1 | 111.0 | 69.4 |
| ArrMemBank::Malloc | 21.2 | 72.5 | 68.1 | 43.2 | 53.5 | 108.4 | 86.4 | 50.5 |
| FixElementBlock::FixElementBlock | 365.8 | 383.9 | 510.2 | 664.4 | 704.9 | 735.0 | 742.4 | 751.8 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | heap.1800 | Heap.1900 | heap.2000 | Heap.2100 |
| currently in use | 52100.9 | 52288.0 | 52368.5 | 52194.1 |
| ArrMemBlock::ArrMemBlock | 275.3 | 341.2 | 320.7 | 272.4 |
| DeccoderInst::get\_nbest\_from\_lattice | 160.3 | 114.5 | 137.5 | 68.7 |
| DecContext::wPropagateTokenToNodeDNN | 138.8 | 97.1 | 111.0 | 69.4 |
| ArrMemBank::Malloc | 53.5 | 108.4 | 86.4 | 50.5 |
| FixElementBlock::FixElementBlock | 704.9 | 735.0 | 742.4 | 751.8 |
|  |  |  |  |  |

    FixElementMemPool(size\_t elementNumPerBlock, size\_t onceAllocElementNum, bool bMemSetZero = false)

        : initElementNumPerBlock\_(elementNumPerBlock)

        , perBlockElementNum\_(elementNumPerBlock)

        , onceAllocElementNum\_(onceAllocElementNum)

    {

        bMemSetZero\_    = bMemSetZero;

        pCurBlock\_      = new FixElementBlock<T>(perBlockElementNum\_, onceAllocElementNum\_,bMemSetZero\_);

        vecBlock\_.reserve(64);

        vecBlock\_.push\_back(pCurBlock\_);

        curBlockPos\_ = 0;

    }

    FixElementBlock(size\_t maxElementNum, size\_t onceAllocElementNum, bool bMemSetZero)

    {

        maxElementNum\_          = (maxElementNum / onceAllocElementNum)\*onceAllocElementNum;

        onceAllocElementNum\_    = onceAllocElementNum;

        data\_                   = new T[maxElementNum\_];

        if (bMemSetZero)

        {

            memset(data\_, 0, sizeof(T) \* maxElementNum\_);

        }

        reset();

#if USE\_MEM\_TRACE

        G\_ArrMemMgr.add\_block<T>(sizeof(T)\*maxElementNum\_);

#endif

    }

FixElement内存池可以分为三层结构

* Pool、Block、Elements
* 内存申请时，可以先从freeList中尝试获取，然后在从已有的block中获取，如果失败，最后重新new一个新的block
* 使用完后

FixElementBlock机构具备的功能

* 一次性按照“maxElementNum”分配内存，称为一个block
* 每次对外提供内存的粒度是“onceAllocElementNum”，分配内存的粒度可以进行设置
* 线程不安全