引擎采用jemalloc内存管理工具下的效率验证分析

1. 分别采用jemalloc和glic自带ptmalloc库对引擎在35路和50路并发的效率进行验证

* 中文测试集（循环模式100次）

|  |  |  |  |
| --- | --- | --- | --- |
| 并发路数 | 内存工具 | PGS首字响应时间(ms) | 子句响应时间(ms) |
| 35 | ptmalloc | 962.9 | 579.9 |
| jemalloc | 1020.5 | 612.3 |
| 50 | ptmalloc | 1013.6 | 594.1 |
| jemalloc | 1090.2 | 636.9 |

1. Jemalloc内存管理效率优于ptmalloc，为何效率降低？

* 验证ptmalloc组件集成是否存在问题

采用ldd查看引擎组件的依赖库，组件依赖libjemalloc.so，功能正常

1. 采用LD\_PRELOAD方式验证效率是否存在差异性？

* 中文测试集（循环模式100次），**和编译链接方式的效率保持一致**

|  |  |  |  |
| --- | --- | --- | --- |
| 并发路数 | 内存工具 | PGS首字响应时间(ms) | 子句响应时间(ms) |
| 35 | Jemalloc(preLoad) | 1035.4 | 615.4 |

1. 更换jemalloc组件（上面测试结论使用的debug版本的jemalloc组件）

* 采用jemalloc对引擎在35路和50路并发效率重新进行验证
* 中文测试集（循环模式100次）
* ptmalloc和jemalloc

|  |  |  |  |
| --- | --- | --- | --- |
| 并发路数 | 内存工具 | PGS首字响应时间(ms) | 子句响应时间(ms) |
| 35 | ptmalloc | 962.9 | 579.9 |
| jemalloc | 973.9 | 580.7 |
| 50 | ptmalloc | 1013.6 | 594.1 |
| jemalloc | 1014.6 | 592.2 |

|  |  |  |  |
| --- | --- | --- | --- |
| 并发路数 | 内存工具 | PGS首字响应时间(ms) | 子句响应时间(ms) |
| 35 | ptmalloc | 962.9 | 579.9 |
| jemalloc | 973.9 | 580.7 |

1. 验证jemalloc和ptmalloc在不同内存分配大小下的效率差异性

* 编写简单的代码，实现对malloc和free函数的调用，验证两者的性能
* 按照不同的固定内存大小进行分配
* 循环进行分配释放100,000,000次（malloc和free）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 执行时间(s) | | | | |
| 内存大小(Bytes) | ptmalloc | Jemalloc | 相对ptmalloc提升 | Jemalloc(preLoad) | 相对ptmalloc提升 |
| 100 | 2.11 | 1.59 | **-24.64%** | 1.61 | **-23.70%** |
| 100 | 2.11 | 1.59 | **-24.64%** | 1.62 | **-23.22%** |
| 400 | 3.26 | 1.59 | **-51.23%** | 1.61 | **-50.61%** |
| 400 | 3.26 | 1.59 | **-51.23%** | 1.64 | **-49.69%** |
| 1000 | 3.19 | 1.59 | **-50.16%** | 1.61 | **-49.53%** |
| 1000 | 3.19 | 1.59 | **-50.16%** | 1.62 | **-49.22%** |
| 4000 | 3.18 | 1.59 | **-50.00%** | 1.66 | **-47.80%** |
| 4000 | 3.18 | 1.59 | **-50.00%** | 1.62 | **-49.06%** |
| 8000 | 3.19 | 1.88 | **-41.07%** | 1.94 | **-39.18%** |
| 8000 | 3.18 | 1.86 | **-41.51%** | 1.94 | **-38.99%** |
| 10000 | 3.19 | 1.88 | **-41.07%** | 2.11 | **-33.86%** |
| 10000 | 3.19 | 1.89 | **-40.75%** | 2.01 | **-36.99%** |
| 12000 | 3.28 | 1.88 | **-42.68%** | 1.94 | **-40.85%** |
| 12000 | 3.28 | 1.92 | **-41.46%** | 1.94 | **-40.85%** |
| 20000 | 3.28 | 2.53 | **-22.87%** | 2.51 | **-23.48%** |
| 20000 | 3.29 | 2.52 | **-23.40%** | 2.51 | **-23.71%** |
| 24000 | 3.28 | 2.53 | **-22.87%** | 2.53 | **-22.87%** |
| 24000 | 3.28 | 2.53 | **-22.87%** | 2.53 | **-22.87%** |
| 30000 | 3.28 | 2.54 | **-22.56%** | 2.52 | **-23.17%** |
| 30000 | 3.28 | 2.53 | **-22.87%** | 2.52 | **-23.17%** |
| 36000 | 3.28 | 25.3 | **671.34%** | 24.51 | **647.26%** |
| 36000 | 3.29 | 24.5 | **644.68%** | 24.63 | **648.63%** |
| 40000 | 3.29 | 24.4 | **641.64%** | 25.26 | **667.78%** |
| 40000 | 3.29 | 23.97 | **628.57%** | 25.25 | **667.48%** |

* 循环100,000,000次下，jemalloc和ptmalloc方案差异性
* jemalloc内存分配性能存在一个拐点，具体拐点的数值未确定，在30000~36000范围内
* 编译链接与preLoad的方式在性能上保持一致

1. 验证trim\_malloc功能在jemalloc组件下是否生效？

* 功能验证代码

|  |
| --- |
| #include <malloc.h>  #include <unistd.h>  #include <stdio.h>  int main()  {  int ret = malloc\_trim(0);  printf("ret=%d\n", ret);    void \*array0 = (void \*)malloc(100);  free(array0);  ret = malloc\_trim(0);  printf("ret=%d\n", ret);  }  /\* Release all but \_\_pad bytes of freed top-most memory back to the     system. Return 1 if successful, else 0. \*/  extern int malloc\_trim (size\_t \_\_pad) \_\_THROW; |

* 在ptmalloc下，调用malloc和free函数分配释放内存，然后调用malloc\_trim(0)函数，该函数返回1，说明内存回收功能正常
* 在jemalloc下，调用malloc和free函数分配释放内存，然后调用malloc\_trim(0)函数，该函数返回0，说明内存回收功能失败，即trim\_malloc无法接入 jemalloc分配的内存资源，即jemalloc下，trim\_malloc功能不生效