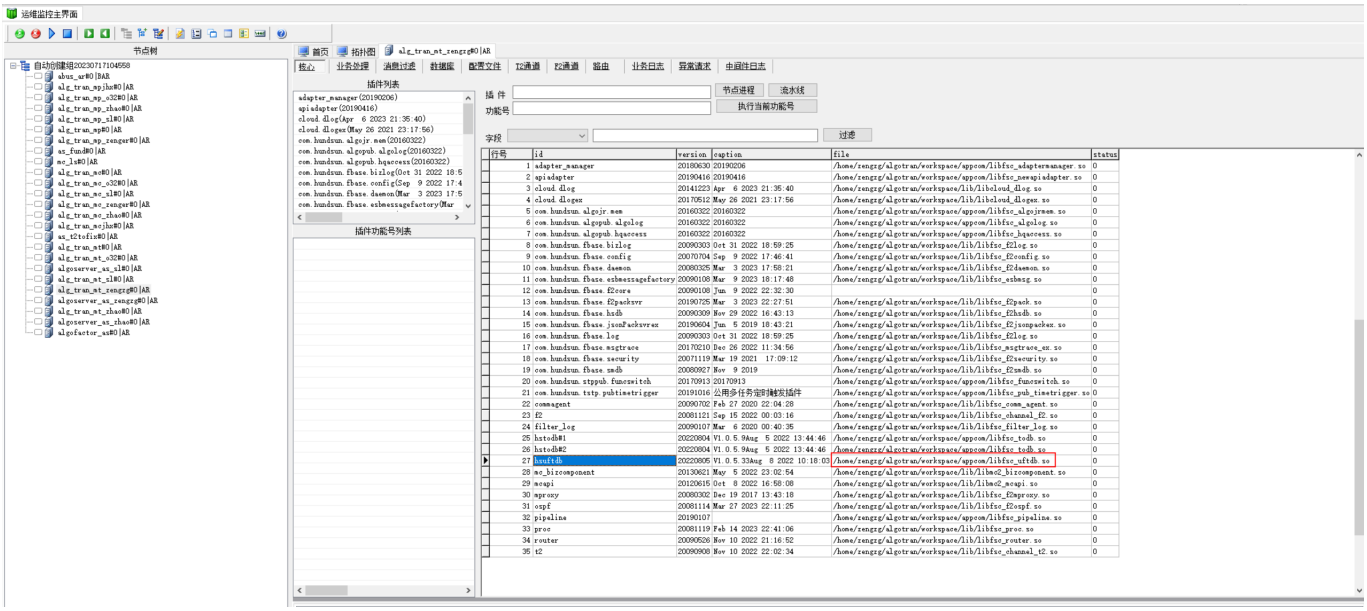
uftdb内存优化

需求：当前uftdb内存表每个表初始化的内存都比较大，需要升级uftdb

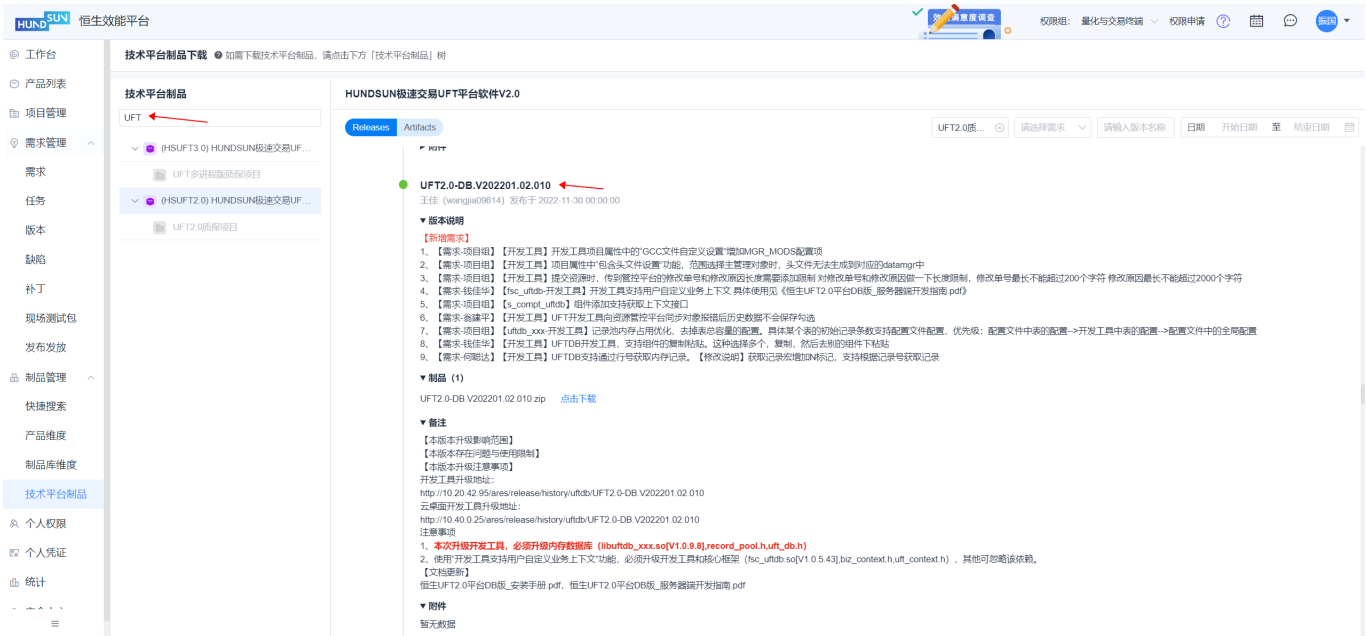
升级

1. 查看uft对应的so，在appcom目录下（如果在lib目录下，一般就是要升级中间件了）

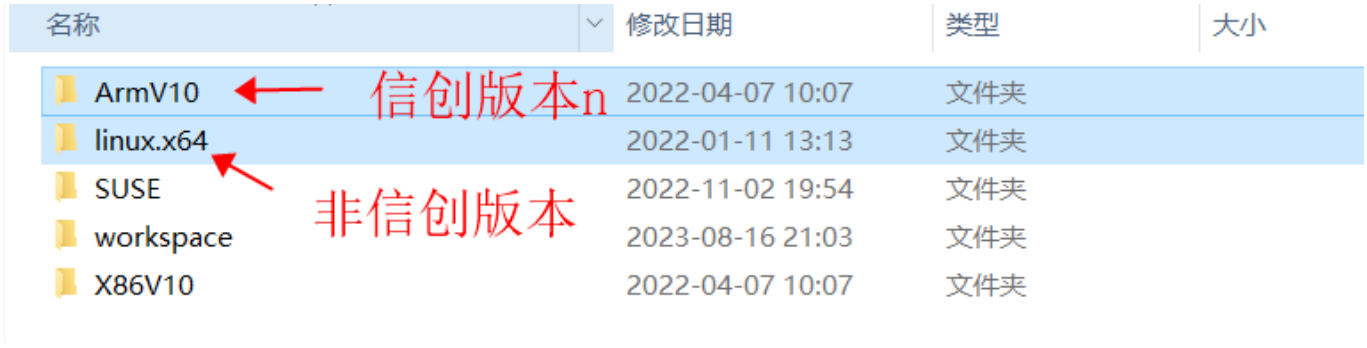
中间件升级

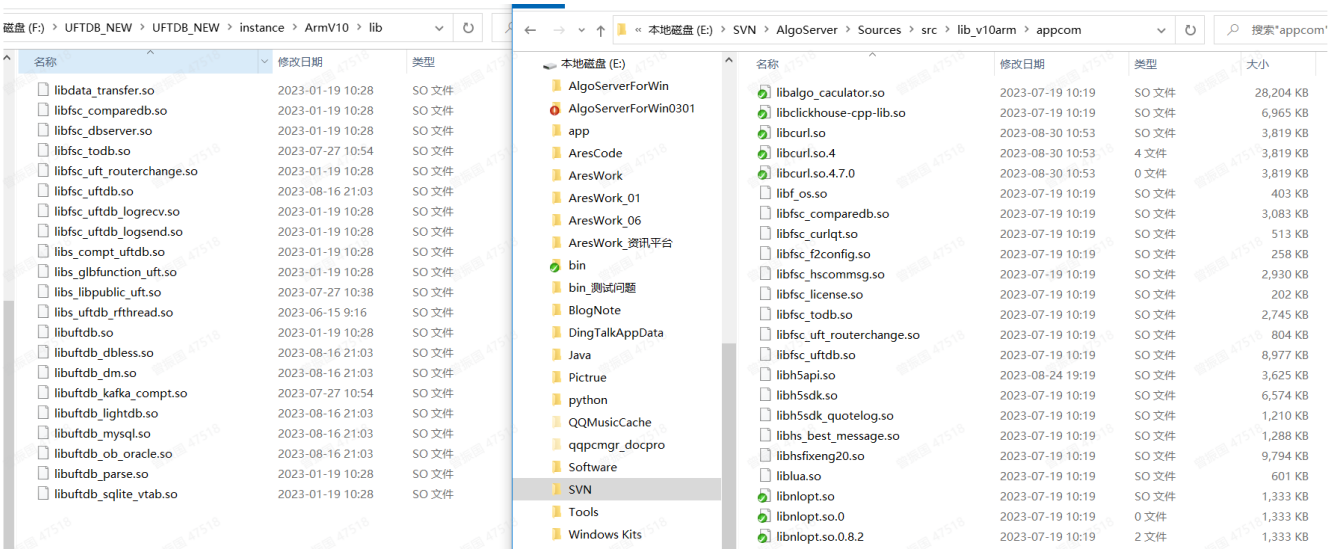


2. 下载制品



3. 覆盖`instance\linux.x64\lib`(非信创)、`instance\ArmV10\lib`(信创)下的so到`Sources\src\lib\appcom`和`Sources\src\lib_v10arm\appcom`

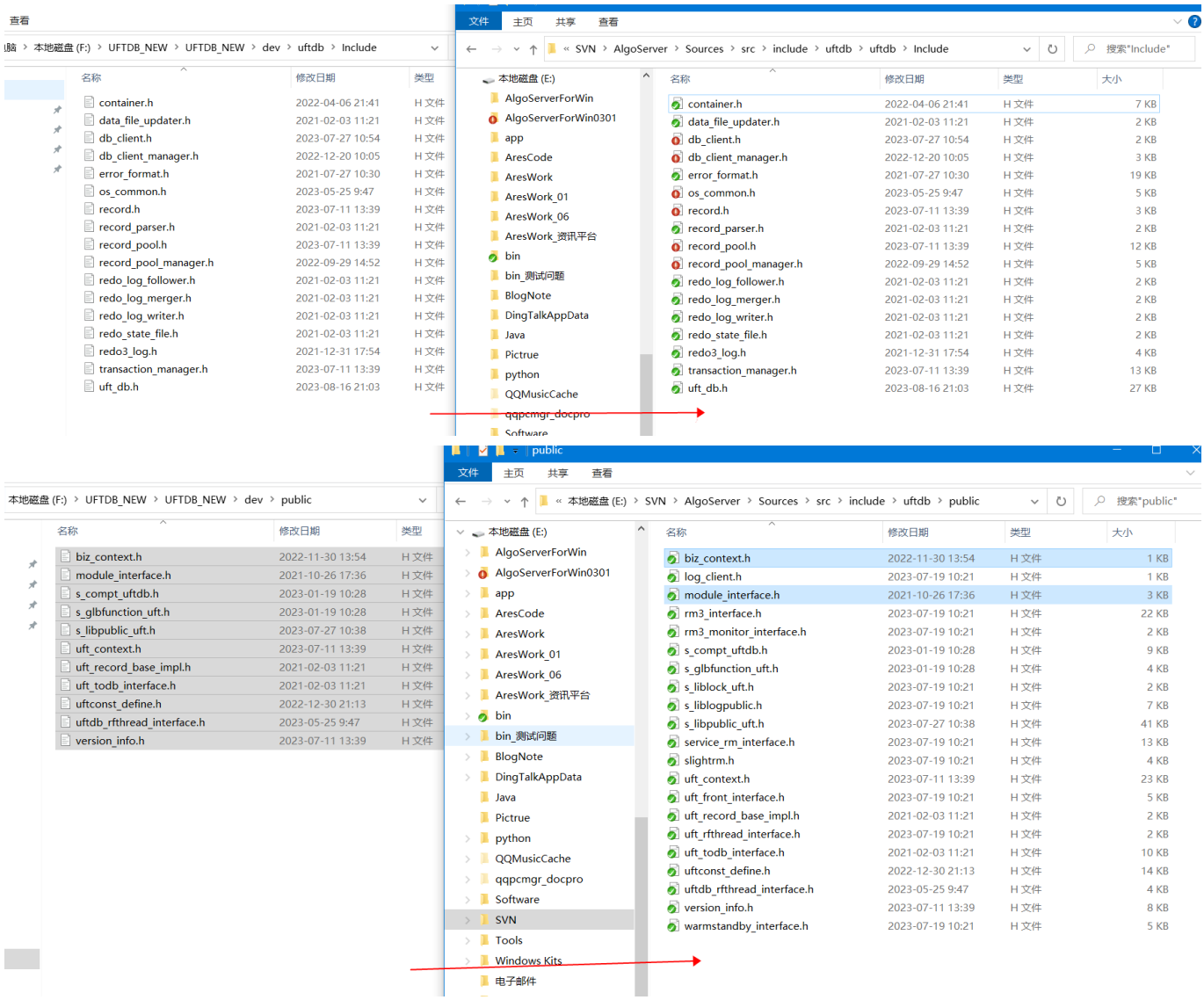




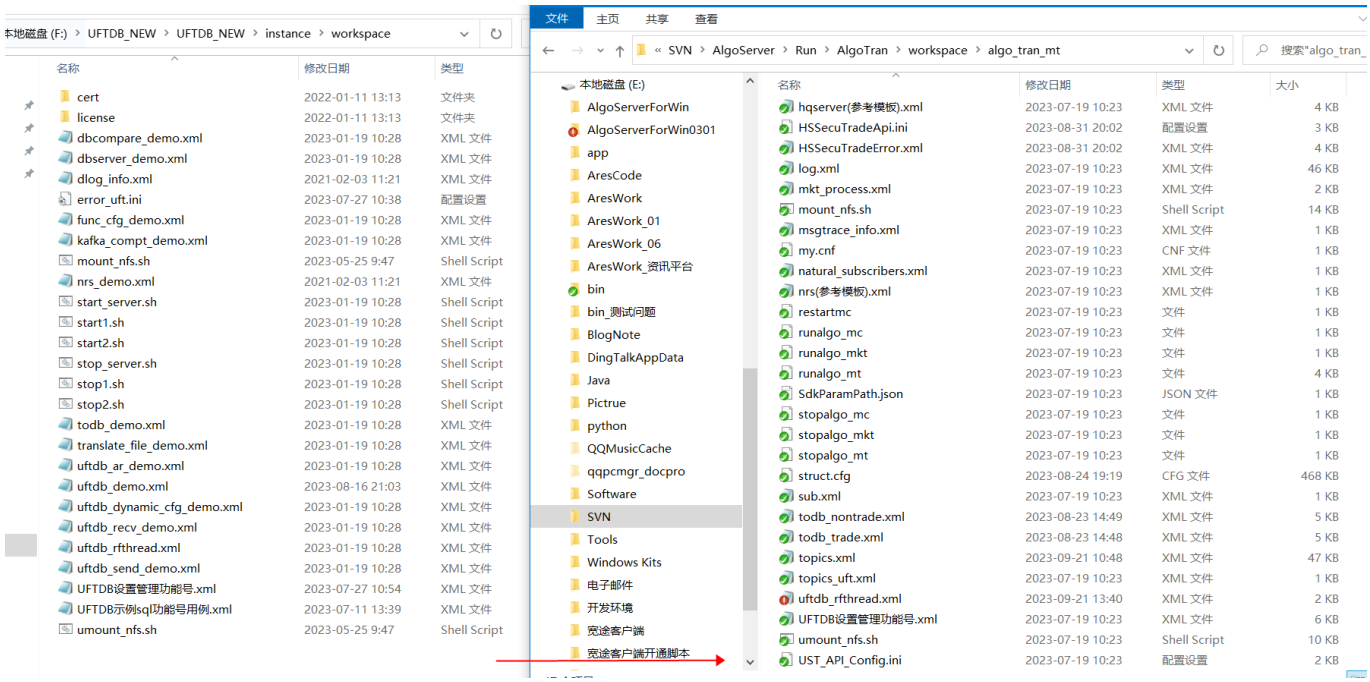
一定要注意：修改source/src下的makefile文件，将新增的so加入到STP_INSTALL中

4. 覆盖`instance\linux.x64\lib`(非信创)、`instance\ArmV10\lib`(信创)下的so到`Run\ASAR\bin`和`Run\ASAR_V10ARM\bin`下

5. 替换`dev\`下的头文件



6. 替换instance\workspace下的配置文件

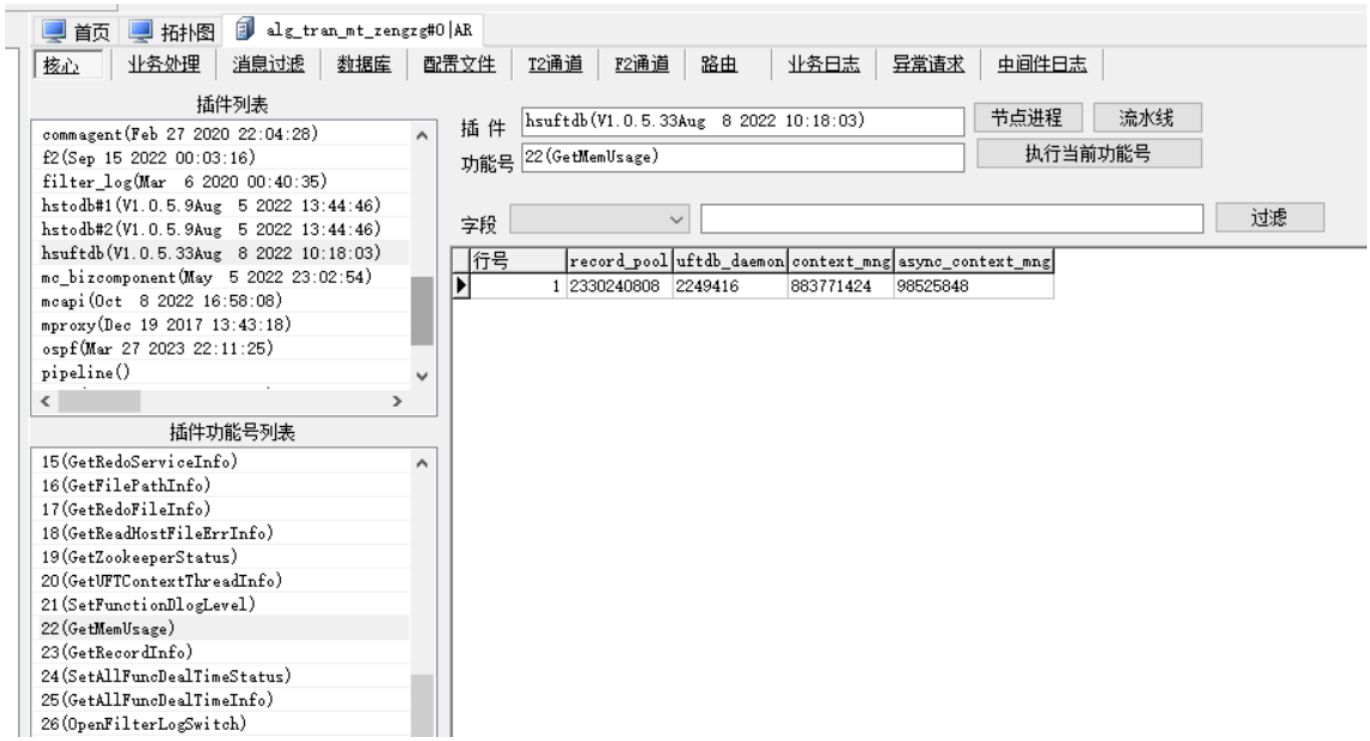


一般来说，如果algotran/workspace下没有的配置文件不需要更新，对已有的xml文件进行覆盖更新就行

7. 查看内存使用大小，这些函数可以在下载制品的pdf中查找

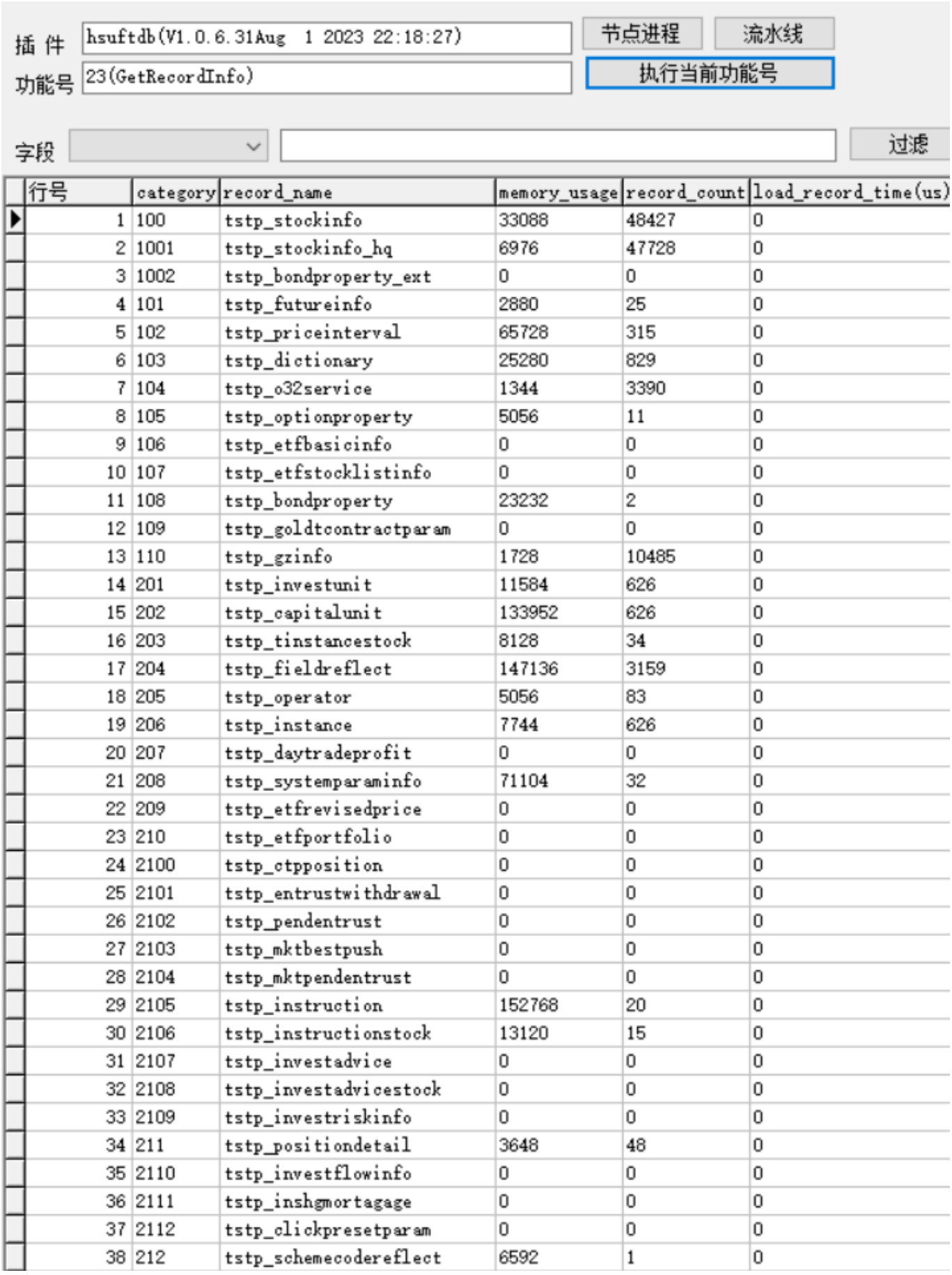
4.22 GetMemUsage 获取内存使用信息

功能名称	GetMemUsage	版本号	V1.0.0.0	更新日期	20171023
功能备注	获取内存使用信息				
功能说明	获取内存使用信息				
入参说明	字段名	说明			类型
	无				
出参说明	字段名	说明			类型
	record_pool	记录池使用内存(单位：字节)			S-字符串
	uftdb_daemon	Uftdb 守护进程使用内存(单位：字节)			S-字符串
	context_mng	上下文管理器使用内存(单位：字节)			S-字符串
	async_context_mng	异步上下文管理器使用内存（单位：字节）			S-字符串



4.23 GetRecordInfo 获取记录信息

功能名称	GetRecordInfo	版本号	V1.0.0.0	更新日期	20171023
功能备注	获取记录信息				
功能说明	获取记录信息				
入参说明	字段名	说明			类型
	无				
出参说明	字段名	说明			类型
	category	记录号			I-整形
	record_name	表名			S-字符串
	memory_usage	使用内存(单位：KB)			S-字符串
	record_count	记录条数			I-整形
	load_record_time	加载记录耗时			I-整形



行号	category	record_name	memory_usage	record_count	load_record_time(us)
1	100	tstp_stockinfo	33088	48427	0
2	1001	tstp_stockinfo_hq	6976	47728	0
3	1002	tstp_bondproperty_ext	0	0	0
4	101	tstp_futureinfo	2880	25	0
5	102	tstp_priceinterval	65728	315	0
6	103	tstp_dictionary	25280	829	0
7	104	tstp_o32service	1344	3390	0
8	105	tstp_optionproperty	5056	11	0
9	106	tstp_etfbasicinfo	0	0	0
10	107	tstp_etfstocklistinfo	0	0	0
11	108	tstp_bondproperty	23232	2	0
12	109	tstp_goldtcontractparam	0	0	0
13	110	tstp_gzinfo	1728	10485	0
14	201	tstp_investunit	11584	626	0
15	202	tstp_capitalunit	133952	626	0
16	203	tstp_tinstancestock	8128	34	0
17	204	tstp_fieldreflect	147136	3159	0
18	205	tstp_operator	5056	83	0
19	206	tstp_instance	7744	626	0
20	207	tstp_daytradeprofit	0	0	0
21	208	tstp_systemparaminfo	71104	32	0
22	209	tstp_etfrevisedprice	0	0	0
23	210	tstp_etfportfolio	0	0	0
24	2100	tstp_ctpposition	0	0	0
25	2101	tstp_entrustwithdrawal	0	0	0
26	2102	tstp_pendentrust	0	0	0
27	2103	tstp_mktbestpush	0	0	0
28	2104	tstp_mktpendentrust	0	0	0
29	2105	tstp_instruction	152768	20	0
30	2106	tstp_instructionstock	13120	15	0
31	2107	tstp_investadvice	0	0	0
32	2108	tstp_investadvicestock	0	0	0
33	2109	tstp_investriskinfo	0	0	0
34	211	tstp_positiondetail	3648	48	0
35	2110	tstp_investflowinfo	0	0	0
36	2111	tstp_inshgmortagage	0	0	0
37	2112	tstp_clickpresetparam	0	0	0
38	212	tstp_schemecodereflect	6592	1	0

8. 最后，在algo的Run\ASAR\bin\CRES2.0历史发布需求清单.xlsx文件中新增记录

uft内存优化方式

每次清流重启mt，都会将内存表的数据备份到./workspace/uftdata/data/目录下，然后重新将表数据加载到内存表。

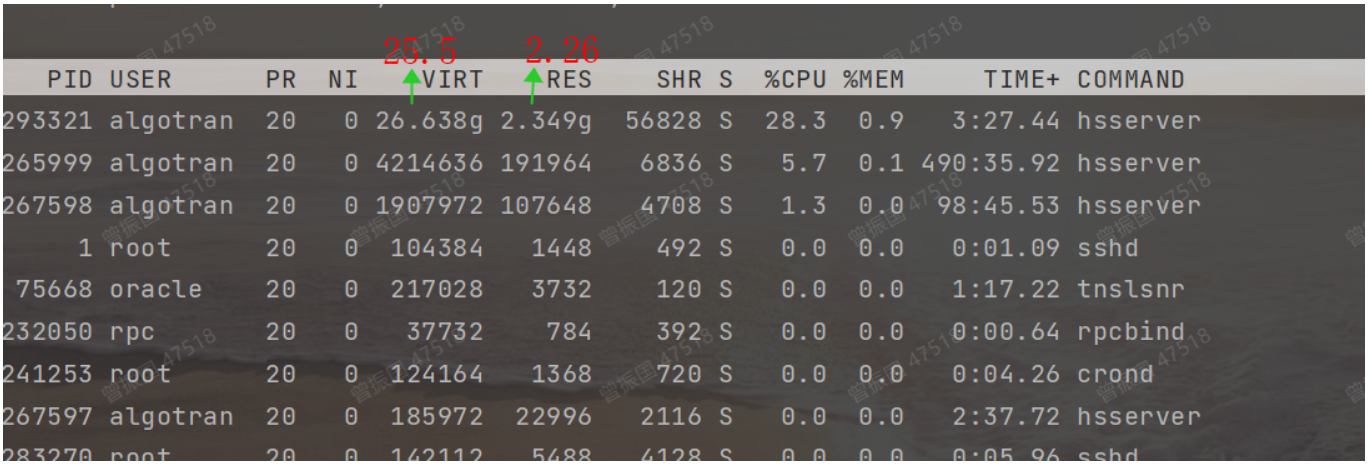
tip:也会将.dat文件压缩备份到workspace下（*.tar.gz）参考链接

低版本的uft每加载一张非空的内存表，都会分配300M的内存，而其中有很多小表，根本用不了300M，因此造成表空间内存浪费非常大，如./workspace/uftdata/data/下的表数据只有85M,但却分配了1.5G的内存

优化后的uft每次加载内存表的数据时，若发现已有内存不足，只会分配一个2M的小块，总的分配的内存为200M左右

查看内存

使用top命令查看mt启动后的占用内存信息：



可以看到：虚拟内存降低了1.1G

注意：因为内存表的表数据一直是85M，虚拟内存映射到物理内存的页数量并未变化，因此mt占用的物理内存不会有明显变化

待优化点

- mt启动后，占用的虚拟内存过大，达到了26G，这会导致mt一旦崩溃，产生的core也会特别大;
- 其次，虚拟内存分配太大，如果超过物理内存，会将物理内存中暂未使用的内存交换到磁盘（在虚拟机中会产生很大一块交换区，在公共服务器上内存很大，交换区内存为0），降低mt性能。