基于 TD—SCDMA ARENA 终端开发平台的内存状态监测解决方案

A Solution of Memory Info Ispection on the Basis of TD – SCDMA ARENA Platform

葛向攀 （浙江大学 计算机学院 杭州 310027）
（华立通信集团 杭州 310012）
卜佳俊 （浙江大学 计算机学院 杭州 310027）

摘要：终端开发是一种小内存开发，内存资源非常宝贵。为提高终端系统的稳定性，本文针对大唐 ARENA TD – SCDMA 终端开发平台提出了内存全面跟踪解决方案。该解决方案在华立通信 TD 系列产品中得到很好实施、验证并达到预期目的。该解决方案对其它产品的内存监测同样具有参考意义。

关键词：TD – SCDMA 3G 内存监测 MMI 内存分配 钩子函数

1 引言

TD – SCDMA[1] 作为全球 3G 的三种标准制式之一，其 3G 业务的发展离不开终端的配合。为了解决 3G 业务发展终端瓶颈，满足全球各种 3G 无线网络制式不同的需求，大唐移动推出了 ARENA TD – SCDMA 终端开发平台。

ARENA TD – SCDMA 终端开发平台是大唐移动基于 TD – SCDMA 标准制式所推出的第一个终端开发平台。手机终端开发是小内存系统，基于 ARENA TD – SCDMA 终端开发平台的终端应用开发，在开发过程中尤其开发初期同样也无可避免的存在终端运行过程中的内存泄露和内存耗尽而造成的终端死机问题。所以很有必要设计一套内存状态的全面检测方案，提高终端性能的稳定性。

2 基本需求

要想全面监测终端运行时的内存使用状况，我们自然会有如下需求：

（1）终端运行时，测试人员可以查看内存使用状态。

（2）可以在终端运行状况记录信息中，根据测试用例的需要，在需要的测试步中输出内存使用状态信息。

（3）可以查看已经分配，尚未释放的所有内存块。

（4）可以查看在两个执行点之间分配的而且还没有释放的内存块。

（5）可以查看两个执行点之间分配的而且还没有释放的内存块的统计信息，包括内存块的数量，总计大小，最大内存块大小，是哪个内存块等。

（6）对于我们自己控制的源代码，可以输出每个内存的分配位置信息。

（7）对于他方代码中的内存分配，可以输出导致该分配的函数调用信息。

（8）可以在 PC 模拟开发环境中使用该功能，也可以在终端上使用该功能。

3 问题分析和目前普遍采用的方法分析

终端总内存量非常有限，要想有效监控内存状态，就必须先思考内存泄露问题。每一次申请的内存在使用完后，是否立即有效释放资源。正确处理了内存泄

露问题，对上面叙述的内存监控需求自然也就实现了。对这两个问题是合二为一的。

对于 MMI 程序来说，在两种情形下发生内存泄露

（1）MMI 程序调用 malloc 后，没有调用 free 释放该内存

（2）MMI 程序调用了他方提供的库函数，这些函数分配内存；此后应该调用另外一个函数来释放这些内存，但是 MMI 程序没有调用。

在 PC 模拟开发环境下，可以通过 VisualC++ debugger[2]和 CRT[2]库检查内存泄露。MMI 的内存泄露应该可以通过这个手段进行有效的检查。所以这里先探讨利用这个机制检查内存泄露的方法。

· 第一种情形的定位方法

使用_CrtMemCheckpoint[2]等函数，DUMP 信息中包含了调用 malloc 的文件名和行号，所以这种内存泄露非常容易定位。

但在实际使用中，发现_CrtMemCheckpoint 时常没有效果。CrtMemDumpAllObjectsSince 还是会打印所有已经分配的条目，包括_CrtMemCheckpoint 前面的内存分配条目。该方法对于 ARENA TD－SCDMA 终端开发平台并不完全适用，需做适当改进。

在 PC 模拟开发环境下，解决该问题的办法是，每次 DUMP 后，把最上面的内存块的分配序号记录下来（比如记录到纸上），相当于人工记录 Checkpoint；下次 DUMP 后，上次最大分配序号到本次最大序号之间的内存块就是尚未释放的内存块。有时，下次 DUMP 后，发现前次的最大序号对应的内存块已经不存在（已经释放），那么就以最接近的次大作为参考点。

实际环境终端运行主程序，目前只能输出内存使用的统计数据，无法提供内存块的详细信息。使用该办法不能有效解决。

· 第二种情形的定位方法

他方库调用 malloc 后泄露，DUMP 信息也会打印出来，但是没有文件名和行号，只提供内存地址和分配序号。那么如何定位哪个函数的调用导致该泄露？

一些文档中提供的一种办法是，假设程序的两次执行的内存分配的过程是完全相同的，那么只要在第一次运行时记下泄露内存的分配序号，在第二次执行时，根据该序号设置内存分配断点，然后在中断时，可以通过函数调用栈看到分配该内存的函数。

遗憾的是，模拟终端两次执行的内存分配过程有巨大的差异。即使在待机界面下不做任何人为操作，也会有随机的内存分配和释放。所以记录下来的泄露内存分配序号对于后一次执行没有任何意义。该方法对于 ARENA TD－SCDMA 终端开发平台终端程序不可行。

4 改进思路

那么该如何解决这一问题呢？采用基于跟踪功能的方案，可以解决这个问题，基本思路如下：

（1）在打印输出路径的同时，把路径标记记录到全局变量（可以是线程变量）中。

（2）在内存分配函数中（或内存分配钩子中），把内存分配序号或内存地址与执行路径标记值成对记录到一个内存块跟踪链表中。

（3）在释放内存时，把内存块跟踪链表中的匹配记录删除。

（4）在任何时候，打印内存块跟踪链表中的所有记录的内存分配序号或内存地址与执行路径标记值，就可以知道哪些内存尚未释放，而且可以知道分配该内存以前最近的执行路径标记。

（5）通过逐步逼近，必定可以找到分配了泄露内存的他方函数。

5 内存泄露解决方案

综合考虑上面的分析和遇到的现实情况，重新设计自己的内存使用状态检测功能是最彻底的方案，这里说明方案，阐述可行性。

5.1 是否能够监测到内存分配管理活动

那么自己重新设计方案，是否可以真正同时有效监测 PC 模拟运行开发环境和终端实际运行环境？

在 PC 模拟开发环境下，VC 提供了内存分配钩子函数。只要应用程序设置了该钩子函数，所有的内存分配操作都会导致执行该函数。所以自己的内存检测功能是可以检测到所有的内存分配活动的。（当然是在设置钩子函数以后的分配活动，或者使用特殊的方法）

实际环境终端运行主程序，是基于 ARENA TD－SCDMA 终端开发平台开发的终端主程序，其内存分配使用的是 tp_os_mem_malloc[3]，大唐提供该系列函数

的源代码,所以我们可以修改这些函数 ,使得自己的内存检测功能可以检测到所有的内存分配等管理活动。

5.2 内存分配检测功能

内存分配检测功能参考 VC 下的_CrtMemCheckpoint 系列接口[2]。

5.3 实现原理

由于 ARENA TD – SCDMA 终端开发平台提供的内存分配 tp_os_mem_malloc 接口函数无法传递文件 ID、行号等信息,所以需要通过线程私有变量来传递文件 ID 等数据 。

提供一个内存分配信息对象集合,记录所有已经分配尚未释放的内存块的信息,包括请求分配的线程 ID、请求分配的文件 ID 和行号(或最接近)。

另外提供一个线程内存分配统计信息,包括该线程发起的内存分配请求次数,释放次数,尚未释放的内存块个数,尚未释放的内存块的总的内存大小,最大块大小。

6 接口设计和实现

6.1 内存状态检查点

ftrace_mem_state

6.2 设置检查点

void ftrace_mem_checkpoint (ftrace_mem_state * p_mem_state)

主要流程:

把 p_mem_state 指向内容清零

设置 p_mem_state 的类型项为 FMBI_CHECKPOINT (枚举类型)

加锁互斥

把 p_mem_state 添加到内存块分配信息链表的头部。

解锁互斥

6.3 输出从指定检查点到当前所有未释放的内存块信息

void ftrace_mem_dump_all_object_since (const ftrace_mem_state * p_mem_state)

主要流程:

检查 p_mem_state 的有效性

按需要的格式输出关于内存分配的统计数据,具体方法如下

遍历各个线程私有数据(系统中总共运行的线程),对其内存分配数据进行累计,得到总体数据,输出。

加锁互斥

对内存块分配信息链表进行正向遍历,直到到达 p_mem_state

对遍历中遇到的每个内存块分配信息作如下处理:

按需要的格式,输出线程号、文件 ID、行号、内存地址。

解锁互斥

6.4 输出当前所有未释放的内存块信息

int ftrace_dump_memory_leaks ( )

主要流程:

与 ftrace_mem_dump_all_object_since 类似,只是遍历全部。

6.5 比较两个检查点之间是否存在未释放的内存

int ftrace_mem_diff (const ftrace_mem_state * p_start,

const ftrace_mem_state * p_end) ;

主要流程:

如果 p_end 就是 p_start 的前一个节点,那么就不存在未释放的内存,返回 0(无差异);否则,返回 1(有差异);异常返回 -1;

6.6 输出两个检查点之间所有未释放的内存块信息

int ftrace_mem_dump_diff (const ftrace_mem_state * p_start,

const ftrace_mem_state * p_end)

主要流程:

与 ftrace_mem_dump_all_object_since 类似,只是从 p_end 开始遍历。注意,调用者应该保证顺序有效性。

6.7 内存状态监测模块初始化

调用初始化的位置需要选择,必须保证在第一次分配内存以前。

主要流程:

创建互斥量。

6.8 记录内存分配和释放

void ftrace_alloc_notify (int alloc_type, void * p_

data, size_t size, int block_use, long request_id);

alloc_type 分配操作类型:_HOOK_ALLOC, _HOOK_FREE

该函数被 tp_os_mem_alloc 和 tp_os_mem_free 调用,在内存分配成功,或释放成功后调用。

(1) _HOOK_ALLOC 的主要流程

从空闲内存块分配信息集合中提取一个空闲的对象 p_block_info。把 p_data,size 保存到 该对象中

获取当前线程私有数据指针,把其中的 thread_id 和其中 mem_info 中的 file_id,line_no,保存到该对象。

同时对 mem_info 中的 alloc_time, active_count, active_bytes, max_block_bytes 进行计算。

加锁

把 p_block_info 插入内存块已分配信息链表的头部

解锁

(2) _HOOK_FREE 的主要流程

加锁互斥

对内存块已分配信息链表进行正向遍历,直到找到地址相同的内存块分配信息对象。

把该对象拆离内存块已分配信息链表,添加到空闲链表。

根据其中保存的 thread_id,修改相应线程的内存分配统计数据 ,active_count,active_bytes。注意,分配的线程和释放的线程可能不同。

解锁互斥

7 结束语

本方案在华立通信集团基于大唐 ARENA TD-SCDMA 终端开发平台开发的 TD 系列产品中由本人实现,并由测试工程师和开发工程师实际使用验证了其可行性,发现并修改了大量内存使用问题,使我们的软件质量稳步提升,系统更加稳定。

参考文献

1 谢显中,TD-SCDMA 第三代移动通信系统技术与实现,电子工业出版社,2004.6.

2 http://msdn.microsoft.com/library/

3 大唐移动,ARENA 平台 API 参考手册