Subject: Out of memory problem in EmcPoint - FairLink !
Posted by StefanoSpataro on Tue, 14 Sep 2010 11:05:45 GMT
View Forum Message <> Reply to Message

Finally I have found who is making the analysis crash.
I would like to repeat, the problem occurs trying to run many dpm events, i.e. 10000.
You can find two kinds of crashes:

terminate called after throwing an instance of 'std::bad_alloc'
  what(): std::bad_alloc

or

Error: Symbol #include is not defined in current scope run_sim_tpccombi_dpm.C:141:
Error: Symbol exception is not defined in current scope run_sim_tpccombi_dpm.C:141:
Syntax Error: #include <exception> run_sim_tpccombi_dpm.C:141:
Error: Symbol G__exception is not defined in current scope run_sim_tpccombi_dpm.C:141:
Error: type G__exception not defined
FILE:/d/panda02/spataro/pandaroot/macro/pid/64/2/./run_sim_tpccombi_dpm.C LINE:141
*** Interpreter error recovered ***
terminate called after throwing an instance of 'std::bad_alloc'
  what(): std::bad_alloc

After a lot of tests, disabling and enabling detectors, putting cout and using valgrind, I have find that the main reason of these leaks is explained by valgrind by the following message:

==18644== 10,028,992 bytes in 156,703 blocks are definitely lost in loss record 358 of 361
==18644==   at 0x4A20929: operator new(unsigned long) (vg_replace_malloc.c:230)
==18644==   by 0x12800BAB: __gnu_cxx::new_allocator<std::_Rb_tree_node<FairLink>
>::allocate(unsigned long, void const*) (new_allocator.h:92)
==18644==   by 0x12800BCF: std::_Rb_tree<FairLink, FairLink, std::_Identity<FairLink>,
std::less<FairLink>, std::allocator<FairLink> >::_M_get_node() (stl_
tree.h:357)
==18644==   by 0x12800C35: std::_Rb_tree<FairLink, FairLink, std::_Identity<FairLink>,
std::less<FairLink>, std::allocator<FairLink> >::_M_create_node(Fair
Link const&) (stl_tree.h:366)
==18644==   by 0x12800D4E: std::_Rb_tree<FairLink, FairLink, std::_Identity<FairLink>,
std::less<FairLink>, std::allocator<FairLink> >::_M_insert_(std::_Rb
_tree_node_base const*, std::_Rb_tree_node_base const*, FairLink const&) (stl_tree.h:852)
==18644==   by 0x12800EB3: std::_Rb_tree<FairLink, FairLink, std::_Identity<FairLink>,
std::less<FairLink>, std::allocator<FairLink> >::_M_insert_unique(Fa
irLink const&) (stl_tree.h:1148)
==18644==   by 0x12800FDC: std::set<FairLink, std::less<FairLink>, std::allocator<FairLink>
>::insert(FairLink const&) (stl_set.h:381)
==18644==   by 0x12801F34: FairMultiLinkedData::InsertLink(FairLink)
(FairMultiLinkedData.cxx:137)
==18644==   by 0x128044D9: FairMultiLinkedData::AddLink(FairLink, bool, float)
(FairMultiLinkedData.cxx:123)
==18644==   by 0x1280347A: FairMultiLinkedData::SetLink(FairLink, bool, float)
(FairMultiLinkedData.cxx:58)
==18644==   by 0x1529D9B1: PndEmcPoint::SetTrackID(int) (PndEmcPoint.h:65)
==18644==   by 0x141BF945: PndStack::UpdateTrackIndex(TRefArray*) (PndStack.cxx:357)

==18644==

I have commented out the three calls to "SetLink" in emc classes, and after I am able to run 10k dpm events without any problems, instead of the old crashes starting from event #200.

For the moment, before the fix of all the problems with the STL code inside FairLink, I will commit these changes in EMC code, just commenting the FairLink lines. At least, the problem seems to appear only for EMC, maybe because the larger size of the data with respect to other detectors.

Of course, if we want to use links for EMC, we need that somebody tries to understand what is wrong with the Links.
