Subject: Re: FairWriteoutBuffer::FillNewData and object ownership (memory leak)
Posted by Oliver Merle on Sat, 26 Jan 2013 16:39:18 GMT
View Forum Message <> Reply to Message

Thank you for this in-depth answer! But I think that the problem is not related to memory-pooling - it happens outside of the pool (TClonesArray):

The 'digits' are usually created on the heap and not in a pool. They are passed as pointers to FairWriteoutBuffer, which stores the pointers in maps. When the data has to be written out, the relating portion of digits is *copied* to a pool via AddNewDataToTClonesArray:

```
void PndSdsDigiStripWriteoutBuffer::AddNewDataToTClonesArray(FairTimeStamp* data)
{
  FairRootManager* ioman = FairRootManager::Instance();
  TClonesArray* myArray = ioman->GetTClonesArray(fBranchName);
  if (fVerbose > 1) std::cout << "Data Inserted: "  <<  *(PndSdsDigiStrip*)(data) << std::endl;
  new ((*myArray)[myArray->GetEntries()]) PndSdsDigiStrip(*(PndSdsDigiStrip*)(data));
}
```

At least I read this as the invocation of a copy-constructor. So the original object on the heap - which was initially passed to the buffer - is still alive and will at some point go out of scope -> memory leak.

For the creation of the objects, see for example PndSdsHybridHitProducer.cxx:370:

```
PndSdsDigiPixel *tempPixel = new PndSdsDigiPixel( fPixelList[iPix].GetMCIndex(),
fInBranchId, fPixelList[iPix].GetSensorID() ,fPixelList[iPix].GetFE(),
                                        fPixelList[iPix].GetCol(), fPixelList[iPix].GetRow(),
                                        smearedCharge, correctedTimeStamp);
//fChargeConverter->GetTimeStamp(point->GetTime(),
charge,fEventHeader->GetEventTime()) );

   if (fTimeOrderedDigi){
            tempPixel->ResetLinks();
            std::vector<int> indices = fPixelList[iPix].GetMCIndex();
            FairEventHeader* evtHeader =
(FairEventHeader*)FairRootManager::Instance()->GetObject("EventHeader.");
            for (int i = 0; i < indices.size(); i++){
              tempPixel->AddLink(FairLink(evtHeader->GetInputFileId(),
evtHeader->GetMCEntryNumber(),  fInBranchId, indices[i]));
            }
            tempPixel->AddLink(FairLink(-1, fEventNr, "EventHeader.", -1));
   }
   fDataBuffer->FillNewData(tempPixel,
fChargeConverter->ChargeToDigiValue(fPixelList[iPix].GetCharge())*6 + EventTime,
point->GetTime()+EventTime);
```

I'd say FairWriteoutBuffer should either delete the digits on the heap after copying them to the

TClonesArray (transfer of ownership by policy) or use one of the popular shared_ptr implementations.

As I said, the number of allocated FairTimestamp derived instances was increasing steadily during a digitization run while they were correctly freed in the simulation. This can be checked.

Kind regards,
Oliver
