Technical Validation of iLCSoft

André Sailer

CERN-PH-LCD, Humboldt-Universität zu Berlin

ILD Software Pre-Meeting
May 22, 2011

Contents

iLCSoft for CLIC CDR

- MC Mass Production for CLIC CDR
- Detector models based on SiD and ILD LoI models
- Reconstruction software from LoIs
- In addition some new developments since the LoI (independent of and connected to CDR):
  - MCRecoTruthLinker
  - PandoraPFANew
  - OverlayTiming
  - ILDCaloDigi
  - CLICTrackSelector
  - CLICPFOSelector
  - SiliconTrackingCLIC
  - CLICCDRMaterialDB
  - NewFTDDigiProcessor

Bugs, Crashes etc.

- Removed many bugs found by early analysis/testing
- Most crashes in recently added code or due to separating physics and $\gamma\gamma \rightarrow$ Hadrons background
- Running v01-11 without crashes

Some selected Issues

- Hard coded limit for track momenta (increased from 1 TeV to 2 TeV)
- Too many combinations from beam-pipe-scatter event, using 'adaptive selection of track seeds' in FTD track reconstruction
- Hard coded limits in vectors (much more MCParticles with background)
- `NULL` pointers show up in the darnedest places

Coverity

- Coverity is a static analyser, doing ‘full path analysis’ and more
- Some smaller bugs
- Next time run this first, would have pointed out some of the memory leaks

Examples

```
 369                }
use of "=" where "==" may have been intended
 370              if( accept_hit = false )
 371                {
 372                  streamlog_out(DEBUG) << "hit could not be smeared within ladder after
```

```
1763          int hits2a = int(hitVec2a.size());
Pointer "combinedTrack" returned by "this->CombineTracks(firstTrack, secondTrack)" is never used.
1764          TrackExtended * combinedTrack = CombineTracks(firstTrack,secondTrack);
1765
1766
```

Memory before Optimisation

- During a first test reconstruction with the use of `OverlayTiming` caused crashes in grid nodes due to massive memory leaks.
- Several memory leaks in `OverlayTiming` were removed by J. Marshall.
- Other processors also showed memory leaks
- Valgrind report (for 2 events w/ Overlay)
  - definitely lost: 29,344,064 bytes in 725,676 blocks
  - indirectly lost: 5,479,328 bytes in 51,380 blocks
  - possibly lost: 49,472,283 bytes in 518,578 blocks
  - still reachable: 12,484,801 bytes in 132,427 blocks

Memory after Optimisation

- Several independent memory leaks from un*deleted* arrays, objects, `LCCollection` not added to `event`, etc.
- Not only in recently added code, but made much worse with BG
- `clear` (processor member) containers at the end of the event (peaks at ≈500 MB)
  - Reduce the maximum memory use
- Afterwards:
  - definitely lost: 1,912 bytes in 27 blocks
  - indirectly lost: 768 bytes in 2 blocks
  - possibly lost: 6,564,315 bytes in 83,421 blocks
  - still reachable: 9,357,146 bytes in 37,744 blocks

Valgrind: Reminder

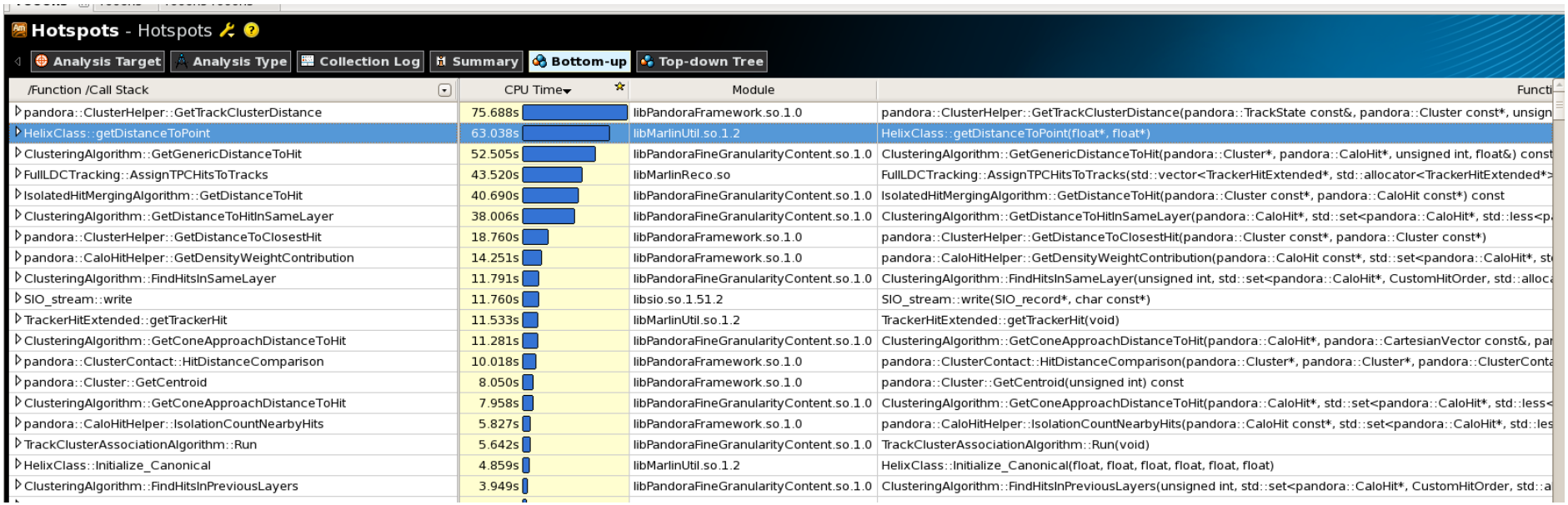
- If you are going to run this with Marlin:
  - `export MARLIN_DEBUG=1` (see Rel. Notes for ilcsoft v01-08)
  - `valgrind --leak-checks=full --suppressions=$ROOTSYS/etc/valgrind-root.supp Marlin steering.xml` (one line)
- Large increase in running time and memory consumption

Running Time

- Checked running time of Reconstruction with Profiler
- 2 events with full overlay
- Total time: 534 s*
- Result:
  - 64% of time spent in PandoraPFA (340 s)
  - 20% in Assign TPC Hits and related functions (100 s)
- Reconstruction time is very similar to simulation time

*This of course depends on the events and hardware, performance similar on CERN grid nodes

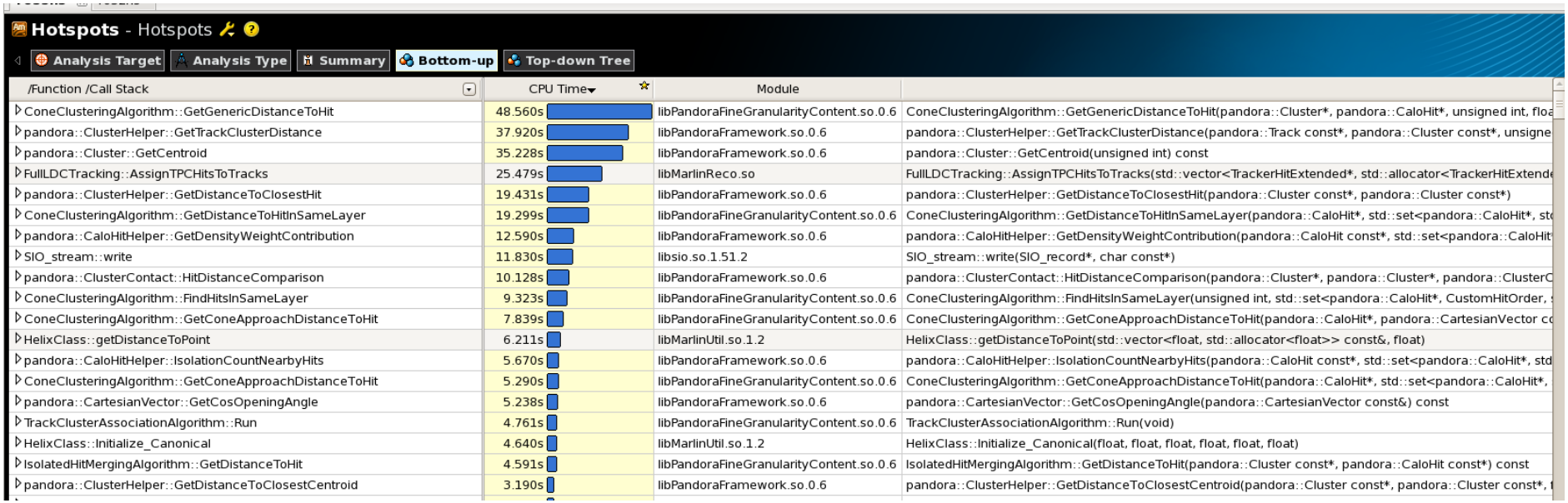
CPU Time Distribution



/Function /Call Stack	CPU Time	Module	Functi
pandora::ClusterHelper::GetTrackClusterDistance	75.688s	libPandoraFramework.so.1.0	pandora::ClusterHelper::GetTrackClusterDistance(pandora::TrackState const&, pandora::Cluster const*, unsign
HelixClass::getDistanceToPoint	63.038s	libMarlinUtil.so.1.2	HelixClass::getDistanceToPoint(float*, float*)
ClusteringAlgorithm::GetGenericDistanceToHit	52.505s	libPandoraFineGranularityContent.so.1.0	ClusteringAlgorithm::GetGenericDistanceToHit(pandora::Cluster*, pandora::CaloHit*, unsigned int, float&) const
FullLDCTracking::AssignTPCHitsToTracks	43.520s	libMarlinReco.so	FullLDCTracking::AssignTPCHitsToTracks(std::vector<TrackerHitExtended*, std::allocator<TrackerHitExtended*>
IsolatedHitMergingAlgorithm::GetDistanceToHit	40.690s	libPandoraFineGranularityContent.so.1.0	IsolatedHitMergingAlgorithm::GetDistanceToHit(pandora::Cluster const*, pandora::CaloHit const*) const
ClusteringAlgorithm::GetDistanceToHitInSameLayer	38.006s	libPandoraFineGranularityContent.so.1.0	ClusteringAlgorithm::GetDistanceToHitInSameLayer(pandora::CaloHit*, std::set<pandora::CaloHit*, std::less<pa
pandora::ClusterHelper::GetDistanceToClosestHit	18.760s	libPandoraFramework.so.1.0	pandora::ClusterHelper::GetDistanceToClosestHit(pandora::Cluster const*, pandora::Cluster const*)
pandora::CaloHitHelper::GetDensityWeightContribution	14.251s	libPandoraFramework.so.1.0	pandora::CaloHitHelper::GetDensityWeightContribution(pandora::CaloHit const*, std::set<pandora::CaloHit*, st
ClusteringAlgorithm::FindHitsInSameLayer	11.791s	libPandoraFineGranularityContent.so.1.0	ClusteringAlgorithm::FindHitsInSameLayer(unsigned int, std::set<pandora::CaloHit*, CustomHitOrder, std::alloc
SIO_stream::write	11.760s	libsio.so.1.51.2	SIO_stream::write(SIO_record*, char const*)
TrackerHitExtended::getTrackerHit	11.533s	libMarlinUtil.so.1.2	TrackerHitExtended::getTrackerHit(void)
ClusteringAlgorithm::GetConeApproachDistanceToHit	11.281s	libPandoraFineGranularityContent.so.1.0	ClusteringAlgorithm::GetConeApproachDistanceToHit(pandora::CaloHit*, pandora::CartesianVector const&, par
pandora::ClusterContact::HitDistanceComparison	10.018s	libPandoraFramework.so.1.0	pandora::ClusterContact::HitDistanceComparison(pandora::Cluster*, pandora::Cluster*, pandora::ClusterConta
pandora::Cluster::GetCentroid	8.050s	libPandoraFramework.so.1.0	pandora::Cluster::GetCentroid(unsigned int) const
ClusteringAlgorithm::GetConeApproachDistanceToHit	7.958s	libPandoraFineGranularityContent.so.1.0	ClusteringAlgorithm::GetConeApproachDistanceToHit(pandora::CaloHit*, std::set<pandora::CaloHit*, std::less<
pandora::CaloHitHelper::IsolationCountNearbyHits	5.827s	libPandoraFramework.so.1.0	pandora::CaloHitHelper::IsolationCountNearbyHits(pandora::CaloHit const*, std::set<pandora::CaloHit*, std::les
TrackClusterAssociationAlgorithm::Run	5.642s	libPandoraFineGranularityContent.so.1.0	TrackClusterAssociationAlgorithm::Run(void)
HelixClass::Initialize_Canonical	4.859s	libMarlinUtil.so.1.2	HelixClass::Initialize_Canonical(float, float, float, float, float, float)
ClusteringAlgorithm::FindHitsInPreviousLayers	3.949s	libPandoraFineGranularityContent.so.1.0	ClusteringAlgorithm::FindHitsInPreviousLayers(unsigned int, std::set<pandora::CaloHit*, CustomHitOrder, std::a

- 60 s in `HelixClass::getDistanceToPoint` called from `FullLDCTracking::AssignTPCHitsToTracks`
- `AssignTPCHitsToTracks` assigns remaining 250k TPC hits to 1000 Tracks (250 million combinations!)

After some Restructuring (see Backup Slides)



Hotspots - Hotspots

Analysis Target | Analysis Type | Summary | **Bottom-up** | Top-down Tree

/Function /Call Stack	CPU Time	Module	
ConeClusteringAlgorithm::GetGenericDistanceToHit	48.560s	libPandoraFineGranularityContent.so.0.6	ConeClusteringAlgorithm::GetGenericDistanceToHit(pandora::Cluster*, pandora::CaloHit*, unsigned int, floa
pandora::ClusterHelper::GetTrackClusterDistance	37.920s	libPandoraFramework.so.0.6	pandora::ClusterHelper::GetTrackClusterDistance(pandora::Track const*, pandora::Cluster const*, unsigne
pandora::Cluster::GetCentroid	35.228s	libPandoraFramework.so.0.6	pandora::Cluster::GetCentroid(unsigned int) const
FullLDCTracking::AssignTPCHitsToTracks	25.479s	libMarlinReco.so	FullLDCTracking::AssignTPCHitsToTracks(std::vector<TrackerHitExtended*, std::allocator<TrackerHitExtende
pandora::ClusterHelper::GetDistanceToClosestHit	19.431s	libPandoraFramework.so.0.6	pandora::ClusterHelper::GetDistanceToClosestHit(pandora::Cluster const*, pandora::Cluster const*)
ConeClusteringAlgorithm::GetDistanceToHitInSameLayer	19.299s	libPandoraFineGranularityContent.so.0.6	ConeClusteringAlgorithm::GetDistanceToHitInSameLayer(pandora::CaloHit*, std::set<pandora::CaloHit*, st
pandora::CaloHitHelper::GetDensityWeightContribution	12.590s	libPandoraFramework.so.0.6	pandora::CaloHitHelper::GetDensityWeightContribution(pandora::CaloHit const*, std::set<pandora::CaloHit
SIO_stream::write	11.830s	libsio.so.1.51.2	SIO_stream::write(SIO_record*, char const*)
pandora::ClusterContact::HitDistanceComparison	10.128s	libPandoraFramework.so.0.6	pandora::ClusterContact::HitDistanceComparison(pandora::Cluster*, pandora::Cluster*, pandora::ClusterC
ConeClusteringAlgorithm::FindHitsInSameLayer	9.323s	libPandoraFineGranularityContent.so.0.6	ConeClusteringAlgorithm::FindHitsInSameLayer(unsigned int, std::set<pandora::CaloHit*, CustomHitOrder,
ConeClusteringAlgorithm::GetConeApproachDistanceToHit	7.839s	libPandoraFineGranularityContent.so.0.6	ConeClusteringAlgorithm::GetConeApproachDistanceToHit(pandora::CaloHit*, pandora::CartesianVector c
HelixClass::getDistanceToPoint	6.211s	libMarlinUtil.so.1.2	HelixClass::getDistanceToPoint(std::vector<float, std::allocator<float>> const&, float)
pandora::CaloHitHelper::IsolationCountNearbyHits	5.670s	libPandoraFramework.so.0.6	pandora::CaloHitHelper::IsolationCountNearbyHits(pandora::CaloHit const*, std::set<pandora::CaloHit*, std
ConeClusteringAlgorithm::GetConeApproachDistanceToHit	5.290s	libPandoraFineGranularityContent.so.0.6	ConeClusteringAlgorithm::GetConeApproachDistanceToHit(pandora::CaloHit*, std::set<pandora::CaloHit*,
pandora::CartesianVector::GetCosOpeningAngle	5.238s	libPandoraFramework.so.0.6	pandora::CartesianVector::GetCosOpeningAngle(pandora::CartesianVector const&) const
TrackClusterAssociationAlgorithm::Run	4.761s	libPandoraFineGranularityContent.so.0.6	TrackClusterAssociationAlgorithm::Run(void)
HelixClass::Initialize_Canonical	4.640s	libMarlinUtil.so.1.2	HelixClass::Initialize_Canonical(float, float, float, float, float, float)
IsolatedHitMergingAlgorithm::GetDistanceToHit	4.591s	libPandoraFineGranularityContent.so.0.6	IsolatedHitMergingAlgorithm::GetDistanceToHit(pandora::Cluster const*, pandora::CaloHit const*) const
pandora::ClusterHelper::GetDistanceToClosestCentroid	3.190s	libPandoraFramework.so.0.6	pandora::ClusterHelper::GetDistanceToClosestCentroid(pandora::Cluster const*, pandora::Cluster const*,

- 7 s in a new `HelixClass::getDistanceToPoint`
- 25 s in `FullLDCTracking::AssignTPCHitsToTracks` (instead of 43 s)
- All other non-Pandora parts combined take 55 s
- (Picture also includes improvements in Pandora by John)

Time per Function

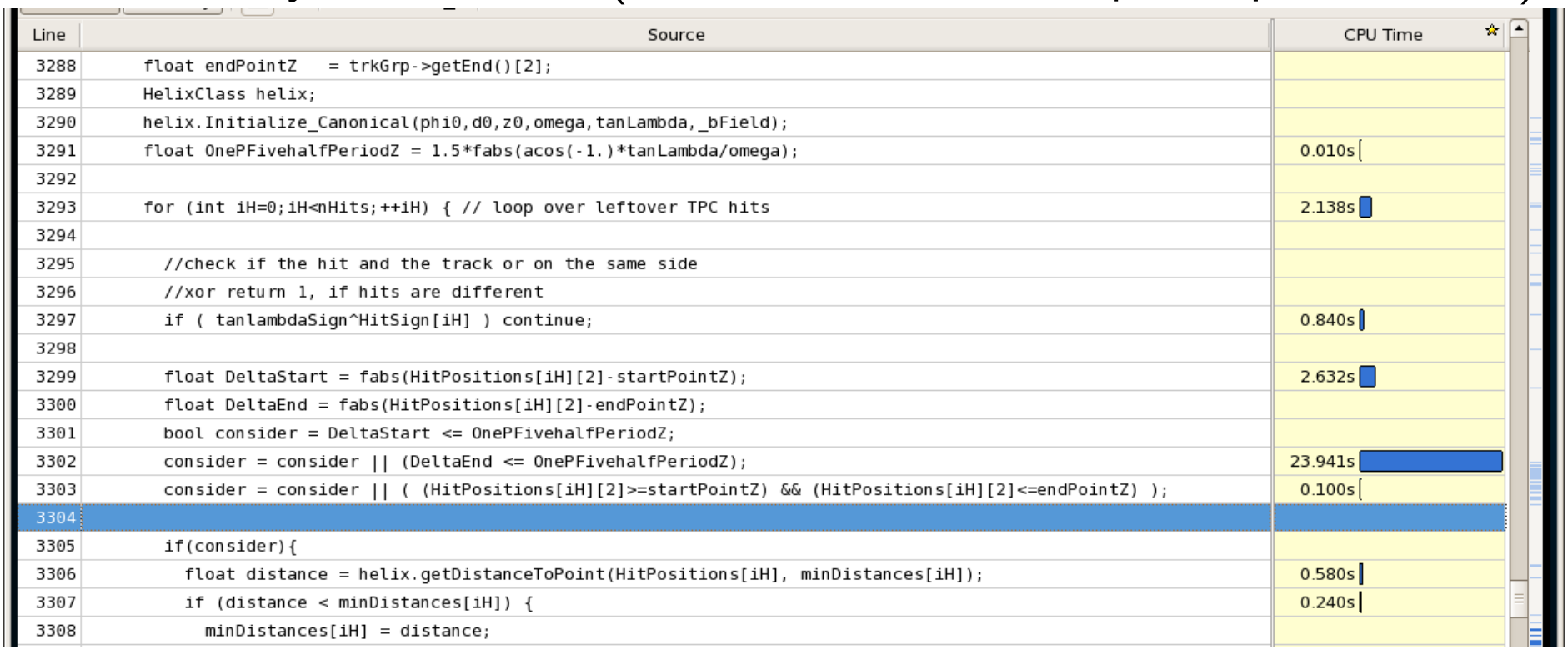
Function	Time [s]
FullLDCTracking::AssignTPCHitsToTracks	23.120
HelixClass::getDistanceToPoint	7.229
HelixClass::Initialize_Canonical	4.481
trseek	2.680
trhlx2	2.660
OverlayTiming::merge_collections	2.420
TPCDigiProcessor::processEvent	2.298
HelixClass::getDistanceToPoint	2.130
SIO_stream::read	1.752
vzero_	1.540
282 more pieces	25.638

Final Result

- Reduced the time per (fully overlaid) event almost by 50%
- Backup slides with more details regarding changes in the code
- The results are unchanged
- `AssignTPCHitsToTracks` is still most time consuming part of reconstruction
- The rest of the time is spent more evenly distributed
  - It is more complicated and less rewarding to find other places to save time
- John will maybe mention something about the timing improvements in Pandora?

Very detailed Profiling

CPU Time by line of code (not exact due to compiler optimisations)



Line	Source	CPU Time
3288	float endPointZ = trkGrp->getEnd()[2];	
3289	HelixClass helix;	
3290	helix.Initialize_Canonical(phi0,d0,z0,omega,tanLambda,_bField);	
3291	float OnePFivehalfPeriodZ = 1.5*fabs(acos(-1.)*tanLambda/omega);	0.010s
3292		
3293	for (int iH=0;iH<nHits;++iH) { // loop over leftover TPC hits	2.138s
3294		
3295	//check if the hit and the track or on the same side	
3296	//xor return 1, if hits are different	
3297	if (tanlambdaSign^HitSign[iH]) continue;	0.840s
3298		
3299	float DeltaStart = fabs(HitPositions[iH][2]-startPointZ);	2.632s
3300	float DeltaEnd = fabs(HitPositions[iH][2]-endPointZ);	
3301	bool consider = DeltaStart <= OnePFivehalfPeriodZ;	
3302	consider = consider || (DeltaEnd <= OnePFivehalfPeriodZ);	23.941s
3303	consider = consider || ((HitPositions[iH][2]>=startPointZ) && (HitPositions[iH][2]<=endPointZ));	0.100s
3304		
3305	if(consider){	
3306	float distance = helix.getDistanceToPoint(HitPositions[iH], minDistances[iH]);	0.580s
3307	if (distance < minDistances[iH]) {	0.240s
3308	minDistances[iH] = distance;	

Summary and Conclusions

- iLCSoft v01-11 is running stably for CDR Mass production with overlay
- No crashes due to bugs or memory leaks
- If I had to do a validation again:
  - Run Coverity first
  - Fix memory leaks and possible obvious bugs
  - Check memory consumption with Valgrind
  - Running time check with a profiler
- Many thanks to everyone who helped!

Backup Slides

Profiler: Intel VTune Amplifier

- At CERN provided by IT (CERN Openlab)
  `/afs/cern.ch/sw/IntelSoftware/linux/`
  (with setup shell script)
- See: `http://indico.cern.ch/conferenceDisplay.py?confId=130853`

Running time improvements: Comments

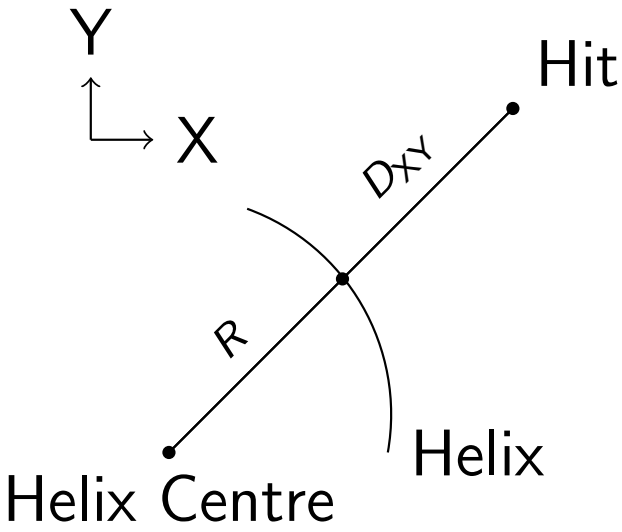
- Using c++ and some pseudo code for explanation
- Removed/simplified some lines for clarity

FullLDCTracking::AssignTPCHitsToTracks

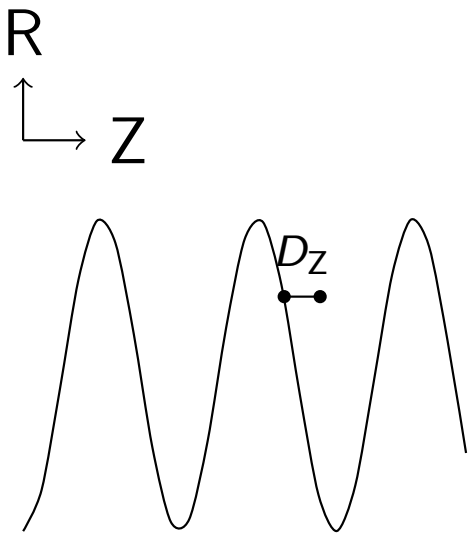
```
forall Tracks helix {
  forall TPCHits Hit {
    helix.getDistanceToPoint(Hit,Distance);

    if (Distance[2]<dcut &&
        Distance[2]<minDist[Hit]) {
      minDist[Hit] = Distance[2];
      tracksToAttach[Hit] = foundTrack;
    }
  }
}
```

`helix.getDistance` fills the array `Distance` with rPhi, Z and 3D distance between the point of the hit and the helix of the track. Its exact value is only used, if the 3D distance is smaller than `dcut`.

What is the distance between Helix and Point?



Distance in XY/RPhi:



Distance in Z:

$$D_{3D} = \sqrt{D_{XY}^2 + D_Z^2}$$

HelixClass::getDistanceToPoint

```
void getDistanceToPoint(float Hit[3], float Distance[3]) {
  float xOnHelix, yOnHelix, zOnHelix;
  float phi = atan2(Hit[1]-_yCentre,Hit[0]-_xCentre);
  xOnHelix = _xCentre + _radius*cos(phi);
  yOnHelix = _yCentre + _radius*sin(phi);
  float phi0 = atan2(_referencePoint[1]-_yCentre,
                     _referencePoint[0]-_xCentre);
  float DistXY = (_xCentre-Hit[0])*(_xCentre-Hit[0])
    + (_yCentre-Hit[1])*(_yCentre-Hit[1]);
  DistXY = fabs(sqrt(DistXY) - _radius);

  [...]//Calculate nCircles
  float DPhi = _const_2pi*((float)nCircles) + phi - phi0;
  zOnHelix = _referencePoint[2] - _charge*_radius*_tanLambda*DPhi;
  float DistZ = fabs(zOnHelix - Hit[2]);
  [...]
  Distance[0] = DistXY;
  Distance[1] = DistZ;
  Distance[2] = sqrt(DistXY*DistXY+DistZ*DistZ);
}
```

HelixClass::getDistanceToPoint

```
void getDistanceToPoint(float Hit[3], float Distance[3]) {
  float zOnHelix;
  float phi = atan2(Hit[1]-_yCentre,Hit[0]-_xCentre);


  float phi0 = atan2(_referencePoint[1]-_yCentre,
                     _referencePoint[0]-_xCentre);
  float DistXY = (_xCentre-Hit[0])*(_xCentre-Hit[0])
    + (_yCentre-Hit[1])*(_yCentre-Hit[1]);
  DistXY = fabs(sqrt(DistXY) - _radius);

  [...]//Calculate nCircles
  float DPhi = _const_2pi*((float)nCircles) + phi - phi0;
  zOnHelix = _referencePoint[2] - _charge*_radius*_tanLambda*DPhi;
  float DistZ = fabs(zOnHelix - Hit[2]);
  [...]
  Distance[0] = DistXY;
  Distance[1] = DistZ;
  Distance[2] = sqrt(DistXY*DistXY+DistZ*DistZ);
}
```

HelixClass::getDistanceToPoint

```
void getDistanceToPoint(float Hit[3], float Distance[3]) {
  float zOnHelix;
  float DistXY = (_xCentre-Hit[0])*(_xCentre-Hit[0])
    + (_yCentre-Hit[1])*(_yCentre-Hit[1]);
  DistXY = fabs(sqrt(DistXY) - _radius);



  float phi = atan2(Hit[1]-_yCentre,Hit[0]-_xCentre);
  float phi0 = atan2(_referencePoint[1]-_yCentre,
                     _referencePoint[0]-_xCentre);
  [...]//Calculate nCircles
  float DPhi = _const_2pi*((float)nCircles) + phi - phi0;
  zOnHelix = _referencePoint[2] - _charge*_radius*_tanLambda*DPhi;
  float DistZ = fabs(zOnHelix - Hit[2]);
  [...]
  Distance[0] = DistXY;
  Distance[1] = DistZ;
  Distance[2] = sqrt(DistXY*DistXY+DistZ*DistZ);
}
```

HelixClass::getDistanceToPoint

```
void getDistanceToPoint(float Hit[3], float Distance[3], float dcut){
  float zOnHelix;
  float DistXY = (_xCentre-Hit[0])*(_xCentre-Hit[0])
    + (_yCentre-Hit[1])*(_yCentre-Hit[1]);
  DistXY = fabs(sqrt(DistXY) - _radius);
  if( DistXY > dcut ) {
    Distance[0] = Distance[1] = Distance[2] = DistXY;
    return;  }
  float phi = atan2(Hit[1]-_yCentre,Hit[0]-_xCentre);
  float phi0 = atan2(_referencePoint[1]-_yCentre,
                     _referencePoint[0]-_xCentre);
  [...]//Calculate nCircles
  float DPhi = _const_2pi*((float)nCircles) + phi - phi0;
  zOnHelix = _referencePoint[2] - _charge*_radius*_tanLambda*DPhi;
  float DistZ = fabs(zOnHelix - Hit[2]);
  [...]
  Distance[0] = DistXY;
  Distance[1] = DistZ;
  Distance[2] = sqrt(DistXY*DistXY+DistZ*DistZ);
}
```

FullLDCTracking::AssignTPCHitsToTracks

```
forall Tracks helix {
  forall TPCHits Hit {
    helix.getDistanceToPoint(Hit,Distance);

    if (Distance[2]<dcut &&
        Distance[2]<minDist[Hit]) {
      minDist[Hit] = Distance[2];
      tracksToAttach[Hit] = foundTrack;
    }
  }
}
```

FullLDCTracking::AssignTPCHitsToTracks

```
forall TPCHits Hit {
  forall Tracks helix {
    float distCut = Min(dcut,minDist[Hit]);
    helix.getDistanceToPoint(Hit,Distance,distCut)

    if (Distance[2]<distCut]) {
      minDist[Hit] = Distance[2];
      tracksToAttach[Hit] = foundTrack;
    }
  }
}
```

Further changes

- After understanding the code better I found some more improvement possibilities
- Move Hit retrieval outside of the nested loops, removes 99.9% of all calls to `GetHitPosition`.
- Start distanceCut with minimal accepted distance, halves the number of comparisons.
- Also removed every unnecessary statement in `GetDistanceToPoint`, calculate once and store
- https://svnsrv.desy.de/viewvc/marlinreco/MarlinUtil/trunk/source/src/HelixClass.cc?r1=2163&r2=2162
- https://svnsrv.desy. . . /FullLDCTracking/src/FullLDCTracking.cc?r1=2162&r2=2161

AssignTPCHitsToTracks Before

```
forall Tracks helix {
  forall TPCHits Hit {
    HitPosition = hit.getPosition()
    helix.getDistanceToPoint(HitPosition,Distance)
    if (Distance[2]<dcut &&
        Distance[2]<minDist[Hit]) {
      minDist[Hit] = Distance[2];
      tracksToAttach[Hit] = foundTrack;
    }
  }
}
```

AssignTPCHitsToTracks After

```
forall TPCHits Hit {
  pos[Hit] = hit.getPosition()
  minDist[Hit]=dcut
}
forall Tracks helix {
  forall TPCHits Hit {
    distance = helix.getDistanceToPoint(pos[Hit]);
    if (distance<minDist[Hit]) {
      minDist[Hit] = distance;
      tracksToAttach[Hit] = foundTrack;
    }
  }
}
```

Bug Example: Vector out of bounds

- `theMCPs.reserve(1000);`
  - And later: `theMCPs[entries]=it->first;`
  - Now: `theMCPs.push_back(it->first);`
- Corruption of memory

Bug Example: Missing `MCParticle`

- Back-scattering `MCParticles` not saved in `Collection`, even though they caused a hit in a tracker (but time $>$ 10 ns)
- `MCParticle *mcp = simHit->getMCParticle()`
- `mcp->getEnergy()` $\leftarrow$ Causes crash
- Check if `mcp` exists