siStrip Unpacking On Demand

Restrict the number of unpacked regions of the siStrip tracker:

HLT oriented tracking

Jean-Roch Vlimant
siStrip Unpack On-Demand

- Silicon strip unpacking is known to take a long time

Regular reco

| Raw | All | digi | All | cluster | On the fly (ckf) | recHit |

Regional unpacking has been implemented (B. Mangano and all) in 131_HLT and 16X cycles
  - Full tracker
  - Regions around ecal super cluster

Regional unpack

| Raw | All in given region | digi | cluster | On the fly (ckf) | recHit |

- To go even further, we implemented the siStrip “unpacking on-demand”
  - Only unpack/clusters modules requested during pattern recognition

Unpack on-demand

| Raw | On the fly, upon request of pattern recognition | digi | cluster | On the fly (ckf) | recHit |
Two possible implementations

• The one presented on August, 22, 2007 (slides 4 to 10), slightly optimized
  ➔ Does absolutely everything on the fly
  ➔ Drawbacks and show-stoppers
    • The DetMap has to be duplicated in DetODMap
      (might be avoided with more in depth modification of xs)
    • SiStripClusters have to be copied over in a vector hold by the MeasurementTracker,
      to get begin and end iterator
    • TrackerRecHit2Ds have to be created with a pointer to the SiStripCluster
      pointer cannot be make persistent.

• A new one, presented here (slides 11 to ...)
  ➔ First a module interface the unpacking
    • Let the MeasurementTracker defines a region to unpack for all modules of the tracker
    • Produces a RefGetter with those regions: put into the event
  ➔ During pattern recognition
    • MeasurementTracker is updated: retrieve a handle to the RefGetter
    • MeasurementTracker::idToDet(id) is called: actually unpacking for the MeasurementDet (if not already done)
  ➔ Major drawback
    • A “region” has to be defined for ALL the module of the tracker: useless
First Implementation

• Pros
  ➔ Minimal number of region defined

• Cons
  ➔ Clusters are not in the event, and cannot be referenced
Concrete Implementation (1)

- The code I have written has the minimal effect on existing code
- Some more in depth modification may make it faster/smarter

- New class: MeasurementTrackerOnDemand
  - Inherits from MeasurementTracker: **minimal changes to existing code**
  - Overloaded function:
    - `update(const edm::Event&)`
      - Does `updatePixels(const edm::Event&)` for the pixel “a la” MeasurementTracker: nothing’s new here
    - Gets the `SiStripLazyGetter<SiStripCluster>` from the event
    - Reserve 10000 regions in a `SiStripRefGetter<SiStripCluster>` for further use
Concrete Implementation (2)

• New class: MeasurementTrackerOnDemand
  ➤ Inherits from MeasurementTracker: **minimal changes to existing code**
  ➤ Overloaded function:
    • idToDet(DetId & id)
      • Look up for the id “a la” MeasurementTracker: to check the id is valid (DetMap)
          ➤ If pixel: return the MeasurementDet*
          ➤ If strip det
            • Has it been updated yet ?
              ➤ If yes: return the MeasurementDet*
              ➤ If not:
                ➤ Unpack around the module
                ➤ Update the MeasurementDet*
                • Separate components if glued
                • Make an entry in the map (DetODMap) to avoid double unpacking
Overloading of \textit{idToDet(DetId & id)}

is id in DetODMap ?

- yes
  - is it a pixel det ?
    - yes
      - Has it been updated yet ?
        - yes
          - return the MeasurementDet*
        - no
          - retrieve center of module.
            - Calculate $\Delta \eta, \Delta \varphi$ from a corner
            - Unpack this region
              (bound to the interface of \textit{SiStripRefGetter})
    - no
      - Has it been updated yet ?
        - yes
          - return the MeasurementDet*
        - no
          - is it in a single det ?
            - yes
              - Copy corresponding clusters into a single vector
              - Get iterator to begin and end
              - Update the MeasurementDet with the iterators
                (bound to the interface of \textit{MeasurementDet})
              - Flag it updated in DetODMap
            - no
              - Loop over components
              - Flag them as updated in DetODMap
                (if component are later queried alone)

- no
  - exception
Validation/test (1)

- Nicholas Cripps is looking into fully validating the track reconstruction
- Regular track reconstruction cannot be broken ... (hopefully)
- “Regional” clustering has been run and still works: not extensively tested

- Extensive tests have been for the unpack-on-demand within L3Muon reconstruction
  - [http://indico.cern.ch/conferenceDisplay.py?confId=19406](http://indico.cern.ch/conferenceDisplay.py?confId=19406)
  - comparison between regular cluster reconstruction and on-demand only.
  - No loss of any track (not that I can noticed over thousands of events)

<table>
<thead>
<tr>
<th>Paths/modules</th>
<th>regular</th>
<th>on-demand</th>
</tr>
</thead>
<tbody>
<tr>
<td>SiStripRawToClustersFacility</td>
<td>N/A</td>
<td>5 ms</td>
</tr>
<tr>
<td>siPixelClusters</td>
<td>&lt; 1 ms</td>
<td>&lt; 1 ms</td>
</tr>
<tr>
<td>siPixelRecHits</td>
<td>&lt; 1 ms</td>
<td>&lt; 1 ms</td>
</tr>
<tr>
<td>siStripClusters</td>
<td>310 ms</td>
<td>N/A</td>
</tr>
<tr>
<td>Total:</td>
<td>310 ms</td>
<td>5 ms</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Paths/modules</th>
<th>cKfPXL</th>
<th>cKfRSpxl (new)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>From pixel pairs</td>
<td>bare seed on pxl surf.</td>
</tr>
<tr>
<td>Overhead</td>
<td>310 ms</td>
<td>5 ms</td>
</tr>
<tr>
<td>L3Muons</td>
<td>100 ms</td>
<td>115 ms</td>
</tr>
<tr>
<td>Total:</td>
<td>410 ms</td>
<td>120 ms</td>
</tr>
</tbody>
</table>

- ~300 ms overhead dropped
- Algorithm timing increases by 10-25%
Validation/test (2)

- Now comparing (more fair) the timing of
  - “Regional” unpacking (all raw to cluster)
  - Unpacking on demand (only requested raw to cluster)

<table>
<thead>
<tr>
<th>Paths/modules</th>
<th>modes</th>
</tr>
</thead>
<tbody>
<tr>
<td>SiStripRawToClustersFacility</td>
<td>“regional” 5 ms</td>
</tr>
<tr>
<td>SiStripRoi</td>
<td>10 ms</td>
</tr>
<tr>
<td>siPixelClusters</td>
<td>2 ms</td>
</tr>
<tr>
<td>siPixelRecHits</td>
<td>&lt; 1 ms</td>
</tr>
<tr>
<td>measurementTrackerUpdater</td>
<td>180 ms</td>
</tr>
<tr>
<td>L3 muon algorithm</td>
<td>X ms</td>
</tr>
</tbody>
</table>

On-demand will be faster for L3 muon path up to X ~ 1 s

Increase of 10-25% of the algorithm timing when on-demand
Additionnal tools

- **MeasurementTrackerUpdator:**
  - EDAnalyzer that calls `MeasurementTracker::update(const Event&)`
  - Avoid to count the update in timed module

- **MeasurementTrackerClusters:**
  - EDProducer that writes to the event the clusters unpacked/created by the MeasurementTracker
  - Keep a record of the unpacked/created clusters
Second Implementation

• Pros
  ➔ Cluster are in the event and can be Referenced

• Cons
  ➔ Way too many region defined: One per module. Large RefGetter
Concrete Implementation (1)

- The code I have written has the minimal effect on existing code

• New class: MeasurementTrackerOD
  - Inherits from MeasurementTracker: **minimal changes to existing code**
  - New function:
    - `define(const edm::Event&, Handle<SiStripLazyGetter<SiStripCluster>>, auto_ptr<SiStripRefGetter<SiStripCluster>>)`
      - Loop over available `MeasurementDet`
        - Does nothing for pixels
        - Update the getter with regions around strip modules (splitting the glued)

• Module MeasurementTrackerUpdator
  - Produces a `SiStripRefGetter<SiStripCluster>`
  - Retrieve the `LazyGetter` from the event
  - Retrieve the `MeasurementTrackerOD` and calls `define(event, LazyGetter, RefGetter)`
  - Put the `RefGetter` to the event
Definition of \textit{define}(...) 

1. \textbf{is id in DetMap ?}
   - \textbf{yes}:
     - \textbf{is it a pixel det ?}
       - \textbf{yes}:
         - (pixel are not on-demand)
       - \textbf{no}:
         - \textbf{no}:
           - \textbf{has it been defined yet ?}
             - \textbf{yes}:
               - exception
             - \textbf{no}:
               - retrieve center of module.
                 - Calculate $\Delta \eta, \Delta \varphi$ from a corner
                 - \textbf{Define} this region in the \textit{SiStripRefGetter}
                 - Get begin/end index of defined regions
                 - \textbf{is it in a single det ?}
                   - \textbf{yes}:
                     - Keep track of the begin/end index in DetODMap
                       - Flag the id defined in DetODMap
                   - \textbf{no}:
                     - Loop over components
                       - Flag them as defined in DetODMap
                       - (if component are later queried alone)
               - \textbf{next one}
         - \textbf{yes}:
           - \textbf{no}:
             - \textbf{next one}
   - \textbf{no}:
     - exception
Concrete Implementation (2)

• New class: \textit{MeasurementTrackerOD}
  ↳ Inherits from \textit{MeasurementTracker}:
  ↳ Overloaded function:
    • \textit{update(const edm::Event&)}
      ↳ Does \textit{updatePixels(const edm::Event&)} for the pixel “a la” \textit{MeasurementTracker}:
        nothing's new here

    • Gets the \textit{SiStripRefGetter<SiStripCluster>} from the event
Concrete Implementation (3)

- New class: MeasurementTrackerOnDemand
  - Inherits from MeasurementTracker:
  - Overloaded function:
    - idToDet(DetId & id)
      - Look up for the id "a la" MeasurementTracker: to check the id is valid
        - If pixel: return the MeasurementDet*
        - If strip det
          - Has it been updated yet?
            - If yes: return the MeasurementDet*
            - If not:
              - Retrieve cluster reference from the RefGetter
                - Update the MeasurementDet*
                  - Separate components if glued
                  - Make an entry in the map (DetODMap) to avoid double unpacking
Overloading of \textit{idToDet(DetId & id)}

- \textbf{is id in DetODMap ?}
  - yes: \textbf{is it a pixel det ?}
    - yes: \textbf{return the MeasurementDet*}
    - no: \textbf{Has it been updated yet ?}
      - yes: \textbf{return the MeasurementDet*}
      - no: \textbf{Look for corresponding clusters between begin/end index of region in \textit{SiStripRefGetter}}
        - \textbf{is it in a single det ?}
          - yes: \textbf{Get iterator to begin and end (RefGetter::record_pair)}
            - Update the MeasurementDet with the iterators, and handle to the \textit{RefGetter}
              - (bound to the interface of \textit{MeasurementDet})
              - Flag it updated in DetODMap
          - no: \textbf{Loop over components}
            - Flag them as updated in DetODMap
              - (if component are later queried alone)
  - no: \textbf{exception}
Second Implementation (bis)

• Pros
  ➔ Less region defined

• Cons
  ➔ Still more regions defined than really needed
Concrete Implementation (1)

- The code I have written has the minimal effect on existing code

• New class: MeasurementTrackerOD
  - Inherits from MeasurementTracker: **minimal changes to existing code**
  - New function:
    • `constructor(...)`
      - Loop over available MeasurementDets
      - Does nothing for pixels
      - Find out the ElementIndex for this strip modules
    • `define(const edm::Event&,
        Handle<SiStripLazyGetter<SiStripCluster>>,
        auto_ptr<SiStripRefGetter<SiStripCluster>>)`
      - Update the getter with the list of elementIndex
      - Register region index in a map

• Module MeasurementTrackerUpdator
  - Produces a SiStripRefGetter<SiStripCluster>
  - Retrieve the LazyGetter from the event
  - Retrieve the MeasurementTrackerOD and calls `define(event,LazyGetter,RefGetter)`
  - Put the RefGetter to the event
Overloading of \texttt{constructor(...)}

- Loop all defined \texttt{MeasurementDet} in \texttt{DetMap}

  - is it a pixel det? (pixel are not on-demand)
    - no
      - Retrieve center of module. Calculate ElementIndex for this module
    - yes
      - Next one

  - Keep track of the ElementIndex in \texttt{region\_mapping} and in \texttt{region\_reverse\_mapping}
Definition of \textit{define}(\ldots)

- Called by an external module that put the \textit{RefGetter} in the event

- Loop all defined \textit{ElementIndex} in \textit{region\_reverse\_mapping}

- Define this region in the \textit{SiStripRefGetter}
  Get begin/end index of defined regions

- Loop over \textit{MeasurementDet}
  associated with the \textit{ElementIndex}
  Keep track of the region begin/end index in \textit{DetODMap}
  Flag the id defined in \textit{DetODMap}
Concrete Implementation (2)

- New class: `MeasurementTrackerOD`
  - Inherits from `MeasurementTracker`:
  - Overloaded function:
    - `update(const edm::Event&)`
      - Does `updatePixels(const edm::Event&)` for the pixel “a la” `MeasurementTracker`:
        - nothing's new here
    - Gets the `SiStripRefGetter<SiStripCluster>` from the event
Concrete Implementation (3)

- New class: MeasurementTrackerOnDemand
  - Inherits from MeasurementTracker:
  - Overloaded function:
    - idToDet(DetId & id)
      - Look up for the id “a la” MeasurementTracker: to check the id is valid
        - If pixel: return the MeasurementDet*
        - If strip det
          - Has it been updated yet?
            - If yes: return the MeasurementDet*
            - If not:
              - Retrieve cluster reference from the RefGetter
                - Update the MeasurementDet*
                  - Separate components if glued
                  - Make an entry in the map (DetODMap) to avoid double operation
Overloading of \textit{idToDet}(\textit{DetId} & \textit{id})

- \textit{is id in DetODMap?} 
  \begin{itemize}
  \item \textbf{yes}: \textit{is it a pixel det?} (pixel are not on-demand)
  \item \textbf{no}: exception
  \end{itemize}

- \textit{Has it been updated yet?}
  \begin{itemize}
  \item \textbf{yes}: return the MeasurementDet*
  \item \textbf{no}: \textit{Look for corresponding clusters between begin/end index of region in \textit{SiStripRefGetter}}
  \end{itemize}

- \textit{is it in a single det?}
  \begin{itemize}
  \item \textbf{yes}: Get iterator to begin and end (\textit{RefGetter}\texttt{::record_pair})
  Update the MeasurementDet with the iterators, and handle to the \textit{RefGetter} (bound to the interface of \textit{MeasurementDet})
  Flag it updated in DetODMap
  \item \textbf{no}: Loop over components
  Flag them as updated in DetODMap (if component are later queried alone)
  \end{itemize}