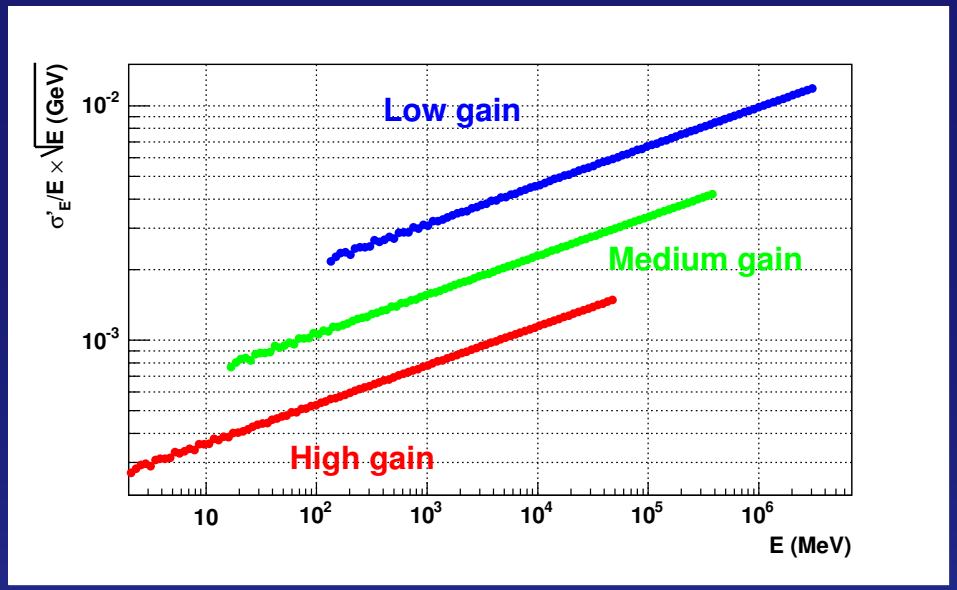
CaloCell Compactification

MPI HEC group meeting Sven Menke, MPI München 1. Oct 2004, MPI

- Compactification general remarks
  - how many bits per cell
  - how many bits per energy/time/quality/gain
  - what algorithms to use
- Actual Implementation
  - CaloEvent/CaloCompactCell
    CaloEvent/CaloCompactCellContainer
  - CaloTools/CaloCompactCellTool
  - CaloUtils/ICaloCompactCellTool
- Tests



Compactification ► General Remarks

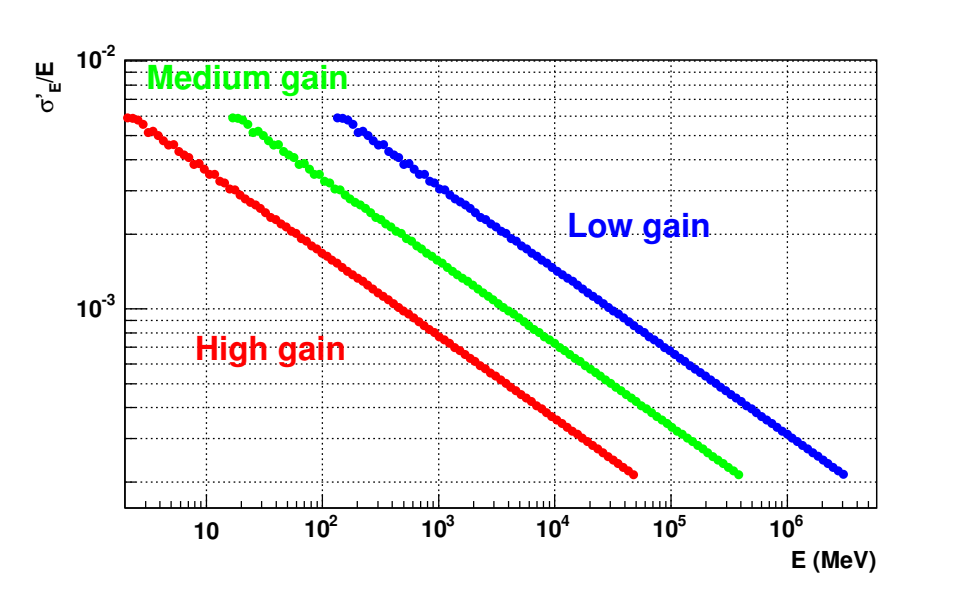
- in order to write all 187652 `CaloCell`s in less than 0.5 MB we can afford at most $0.5 \times 1024 \times 1024 \times 8/187652 = 22$ bit per `CaloCell`.
- keeping 16 bit boundaries this means we have to cope with 16 bit per `CaloCell`.
- at full length a `CaloCell` contains 256 bit:
  - 32 bit `Identifier`
  - 64 bit `double` for energy
  - 64 bit `double` for time
  - 64 bit `double` for quality
  - 32 bit `int` for gain
- what can be spared easily
  - 32 bit `Identifier` can be left out if all `CaloCell`s are stored ordered by `IdentifierHash`
  - 32 bit `int` gain can be stored in 2 bit (high, medium, low)
  - rest not so easy . . .

- ultimately we need $\sigma_E/E < 0.1/\sqrt{E/\mathrm{GeV}}$
- 1 % decrease to $\sigma_E/E < 0.101/\sqrt{E/\mathrm{GeV}}$ tolerable
- precision loss $\sigma'_E/E < 0.014/\sqrt{E/\mathrm{GeV}}$
- compare different packing options for 3 gain ranges with gains 1, 8, and 64, respectively and $E^{\mathrm{high}}_{\mathrm{min}} = 8\,\mathrm{MeV}$, and $E^{\mathrm{low}}_{\mathrm{max}} = 3.2\,\mathrm{TeV}$
- logarithmic:
  - store $\ln|x|$ for $|x_0| < |x| < |x_1|$ with n bit
  - resolution: $\frac{\sigma_x}{x} = \frac{1}{\sqrt{12}}\left(1 - \left(\frac{|x_0|}{|x_1|}\right)^{2^{-n}}\right)$
  - needed bits: $n > -\frac{1}{\ln 2}\ln\left(\frac{-\ln\left(1 - \frac{0.014\cdot\sqrt{12}}{\sqrt{|x_1|/\mathrm{GeV}}}\right)}{\ln\left(\frac{|x_1|}{|x_0|}\right)}\right) = 13.315$

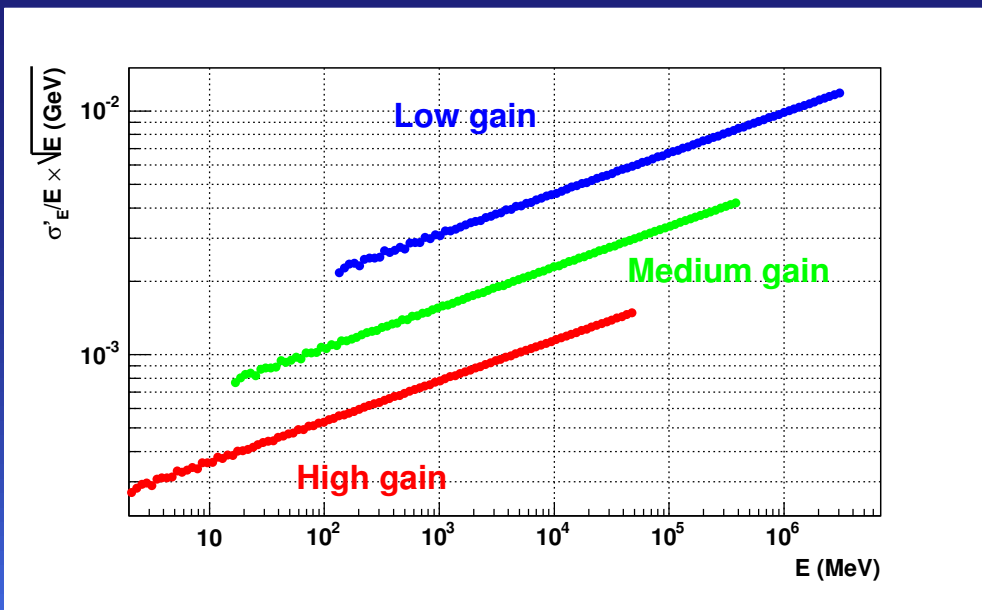
- square root:
  - store $\sqrt{|x|}$ for $|x| < |x_1|$ with n bit
  - resolution: $\frac{\sigma_x}{x} = \frac{2}{\sqrt{12}}\sqrt{\frac{|x_1|}{x}}2^{-n}$
  - needed bits: $n > -\frac{1}{\ln 2}\ln\left(\frac{\sqrt{12}}{2}\frac{0.014}{\sqrt{|x_1|/\mathrm{GeV}}}\right) = 11.188$
- cubic root:
  - store $|x|^{1/3}$ for $|x| < |x_1|$ with n bit
  - resolution: $\frac{\sigma_x}{x} = \frac{3}{\sqrt{12}}\left(\frac{|x_1|}{x}\right)^{1/3}2^{-n}$
  - needed bits: $n > -\frac{1}{\ln 2}\ln\left(\frac{\sqrt{12}}{3}\frac{0.014}{\sqrt{|x_1|/\mathrm{GeV}}}\right) = 11.773$
- best seems cubic root with $n = 12$ bit

- 12 bit cubic root packing of $|E|$ and 3 gain switches ($\times 1, \times 8, \times 64$) up to 3.2 TeV in the `Low` gain
- upper plot shows relative precision loss σ'_E/E
- lower plot shows degradation in the sampling term $\sigma'_E/E \times \sqrt{E\ (\mathrm{GeV})}$

σ'_E/E



$\sigma'_E/E \times \sqrt{E(\mathrm{GeV})}$

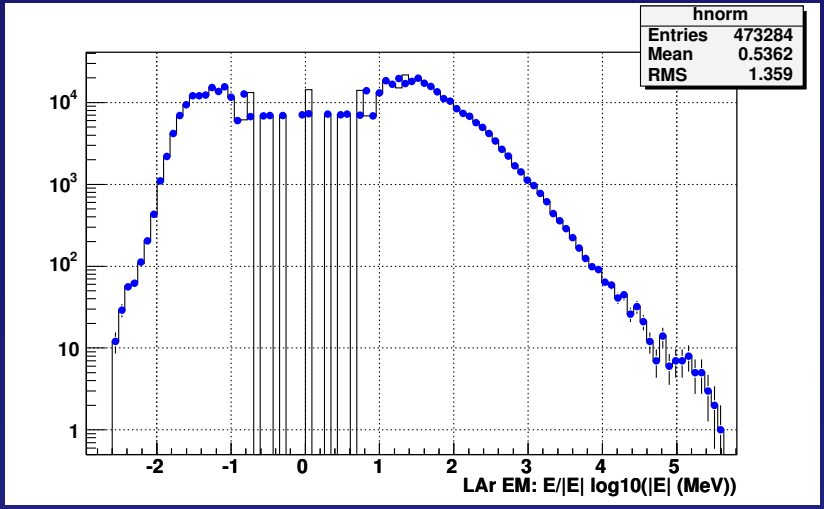


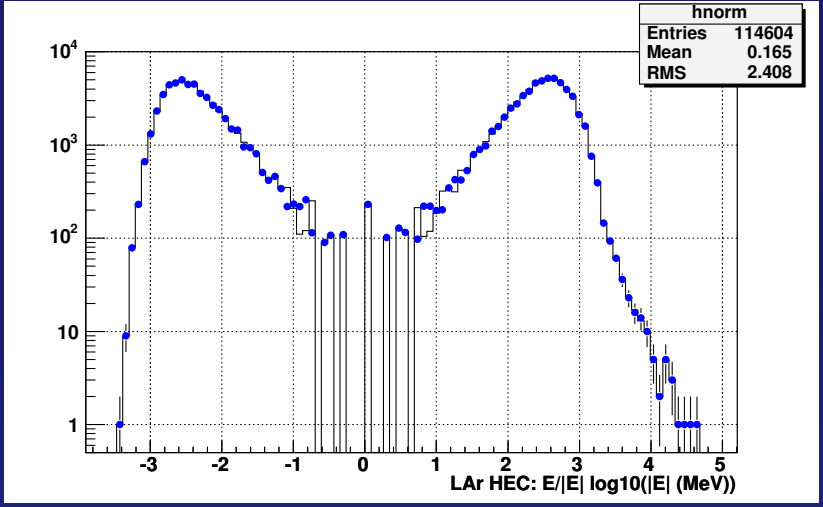
- with 12 bit for $|E|^{1/3}$ and 1 bit for the sign of E we are left with:
  - 2 bit for gain (`Low`,`Medium`,`High`)
  - 1 bit for quality (`Bad`, `Good`), originaly intended for goodness of fit (probability or χ^2) currently distinguishes hits with/without time information
- reduced set of gain values for the Tile
  - `TILELOWLOW`, `TILEONELOW`, `LARLOWGAIN` are `Low`
  - `TILELOWHIGH`, `TILEHIGHLOW`, `LARMEDIUMGAIN` are `Medium`
  - `TILEHIGHHIGH`, `TILEONEHIGH`, `LARHIGHGAIN` are `High`
- Only 10 % of all cells have time information
  - store 15 bit for $\log|t|$ with $0.01\,\text{ns} < |t| < 125\,\text{ns}$ and 1 bit for the sign of t in a second 16 bit word
  - precision loss for time: $\sigma_t'/t = 8.31 \cdot 10^{-5}$

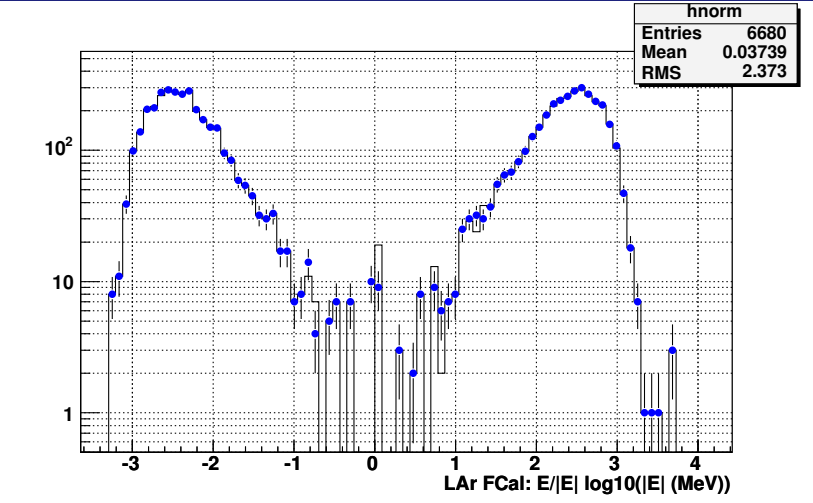
Actual Implementation

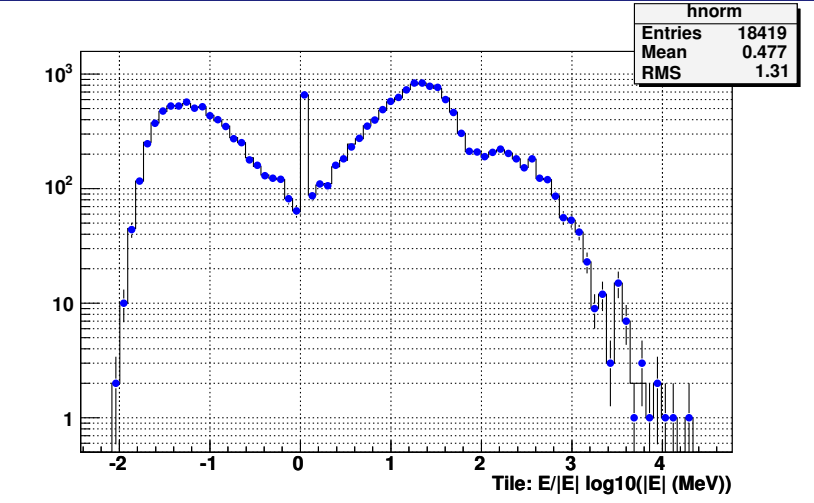
- `CaloUtils/ICaloCompactCellTool`
  - is the interface to use for conversions from and to `CaloCellContainer`
  - `virtual StatusCode getPersistent(const CaloCellContainer &theCellContainer, CaloCompactCellContainer *theCompactContainer, const int theVersion = VERSION_210)=0;`
    fills the persistent object
  - `virtual StatusCode getTransient(const CaloCompactCellContainer &theCompactContainer, CaloCellContainer *theCellContainer)=0;`
    fills the transient `CaloCellContainer`
- `CaloTools/CaloCompactCellTool`
  - is the actual tool used by the interface
  - defines the bit masks and conversion routines
  - can be extended easily and will be backward compatible since version information is kept in a header of the persistent object
- `CaloEvent/CaloCompactCellContainer`
  - can be persisted since its data is a simple `std::vector<int>`
  - uses 0.39 MB for a full `CaloCellContainer`
- `CaloEvent/CaloCompactCell`
  - is used by the tool and the compact container as storage for one compactified `CaloCell`

- I've tested the code with `atlrel_0` (21-Sep-2004)
- plot $E/|E|\log_{10}|E|$ for `CaloCell`s in `CaloCluster`s with/without compactification



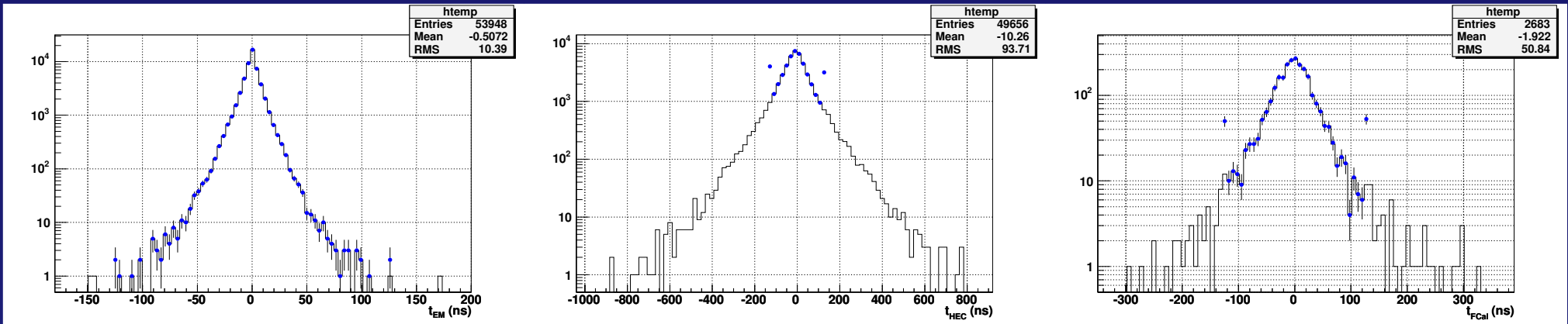






Tests ► Time/Gain

- test with `atlrel_0` (21-Sep-2004), continued
- plot *t* for `CaloCell`s in `CaloCluster`s with/without compactification



- and the gain
  - 6 out of 600000 LAr gain values shifted from `HIGH` to `MEDIUM`
  - single PMT Tile cells could restore correct gain from geometrical information