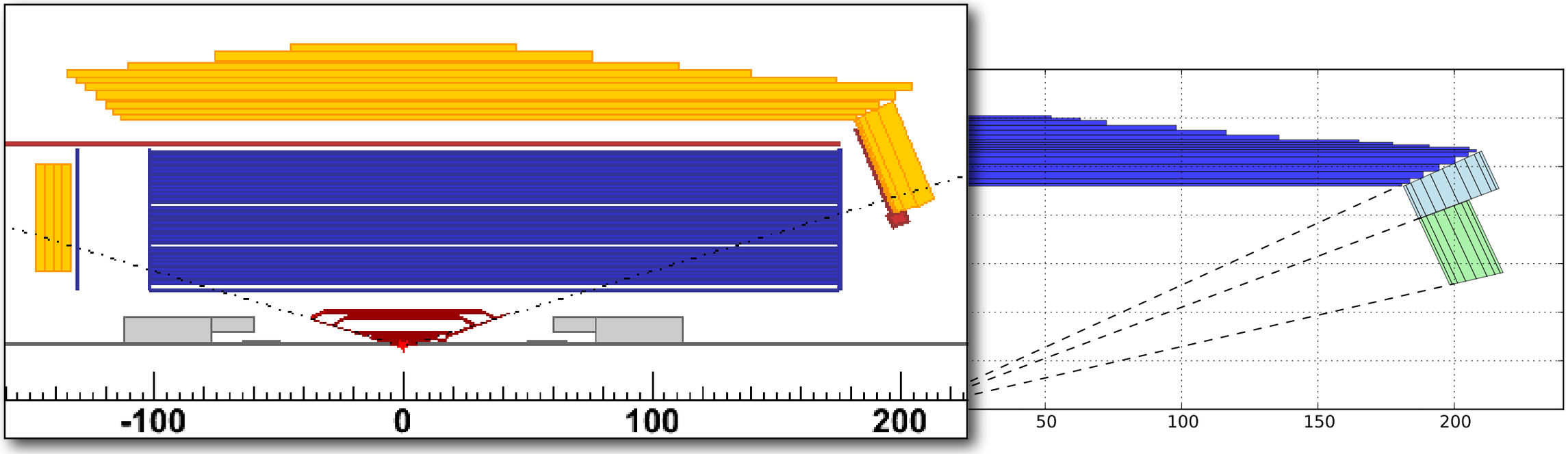
EMC in FastSim

Chih-hsiang Cheng
Caltech

SuperB Collaboration Meeting
La Biodola (Isola d'Elba), Italy, 2012/05/31–06/04

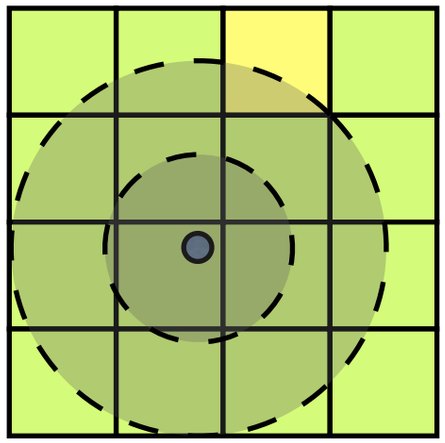
Overview (1)

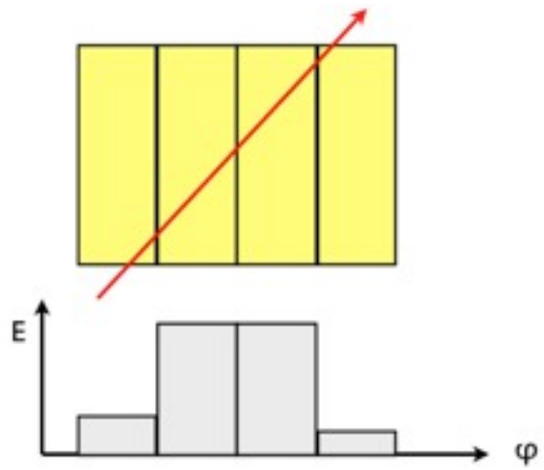
- EMC volumes are modeled with layers of 2D planes.
- It is divided into four regions; any one can be null, thus degenerate to less than four.
- It was hard coded with 3 regions and was hard to change; now it's more flexible and can be changed to five or more if needed with not much effort.

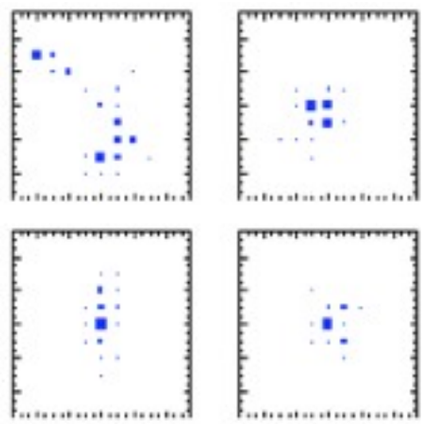


Overview (2)

- Probability of showering and energy deposition of a particle intercepting a plane is calculated based on particle type, radiation/interaction length, shower profile (Moliere radius).
- For EM shower, each energy point is distributed to a grid of crystals (θ,ϕ) based on the integral of a profile $f(r;R_M)$ over the crystal area projected onto a plane perpendicular to the entrance direction.
- Minimal ionizing particle by a straight line.
- Hadron shower is modeled with a mixture of EM shower and random walk.

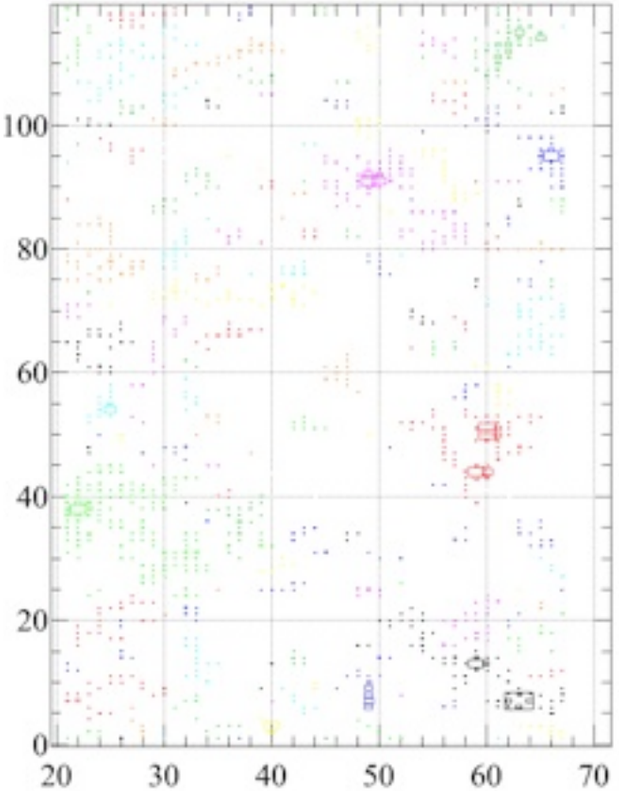






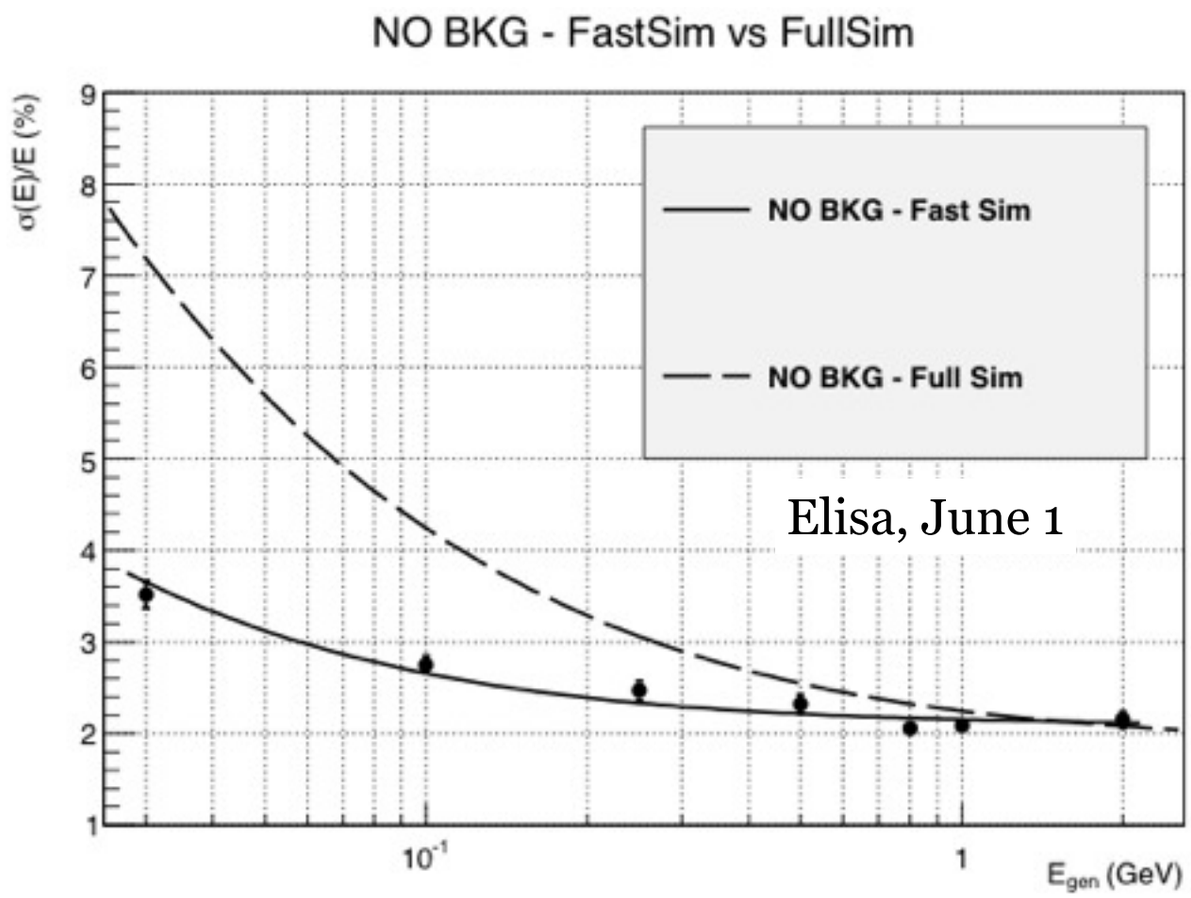
Overview (3)

- Contribution from all layers are then merged to become a single cluster.
- Energy is smeared based on a resolution function and then scaled up/down based on a calibration function.
- Crystal energy is fluctuated to randomize cluster shape; noise added around the cluster.
- All clusters in an event fill their energy to a map; select seeds and re-build clusters; and then split them up if there is more than one bump in a cluster. Particle candidates are single-bump clusters.



Issues(1): Energy resolution

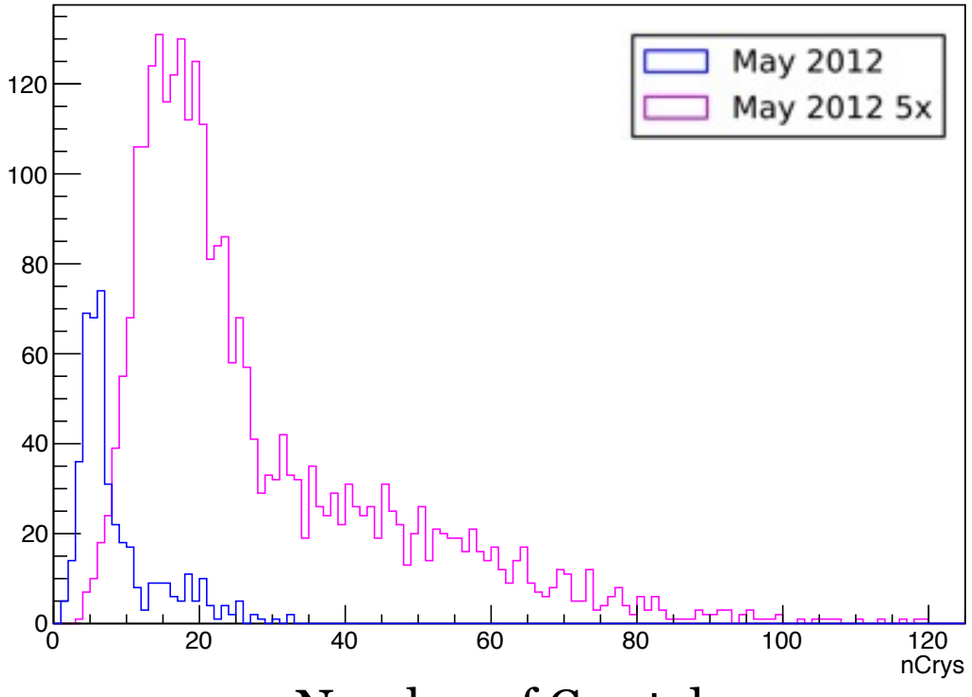
- Current energy resolution underestimate, compared to G4 simulation.
- Use 2 sets of parameters: for Gaussian smearing & exponential tail; hard to compare the one set of parameters describing the resolution.
- Plan to change to Crystal Ball function smearing.



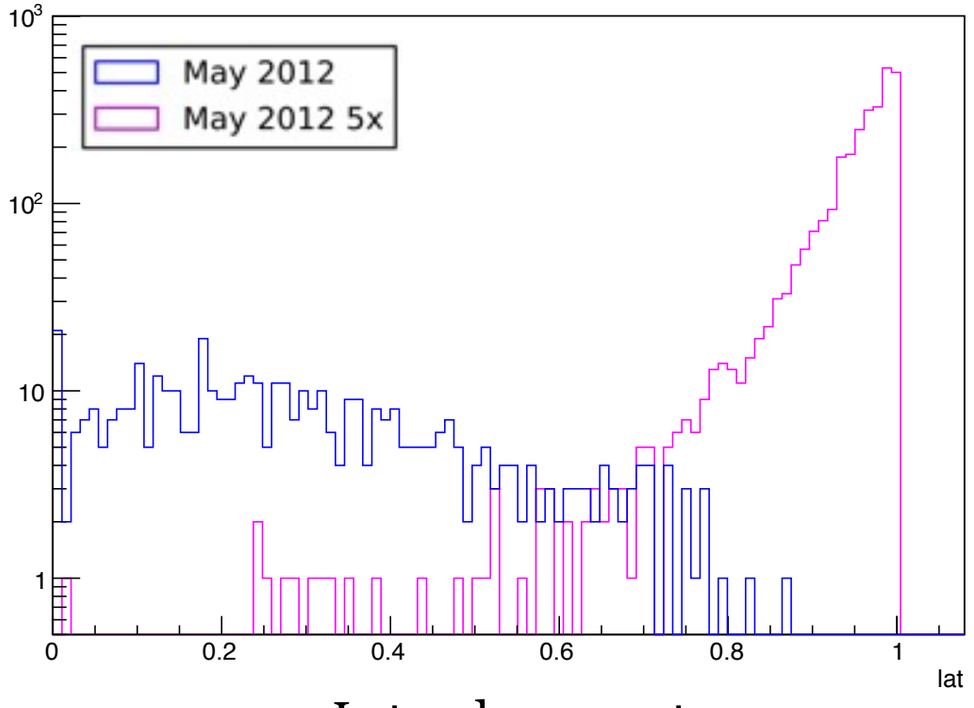
- Complication: even without the smearing, the resolution is not a delta function, mainly due to gaps between crystals, and crystal space projection approximation.
  - ✦ Solution: model this intrinsic resolution and subtract from the intended resolution then do the smearing (ToDo).

Issues(2): Clustering algorithm

- Current algorithm: select seed crystals above 20 MeV, from high to low; connect adjacent crystals; if a crystal is below 5 MeV and none of its neighbors is above 10 MeV then stop (this crystal is not used).
- At low background, it's not a problem. But at high background, the cluster can get pretty large.
- Limiting cluster size may help resolution and background multiplicity. Idea: limit cluster diameter using a function of seed energy and Moliere radius.



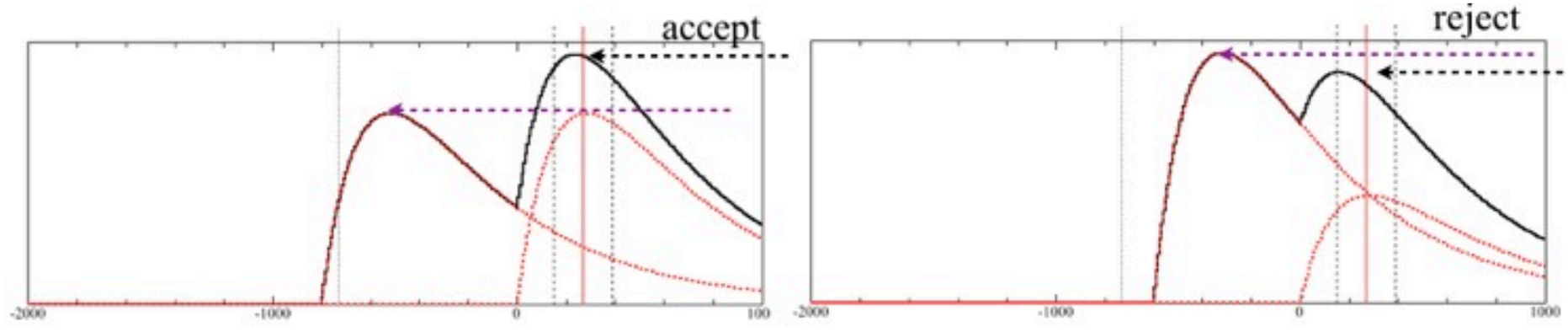
Number of Crystals

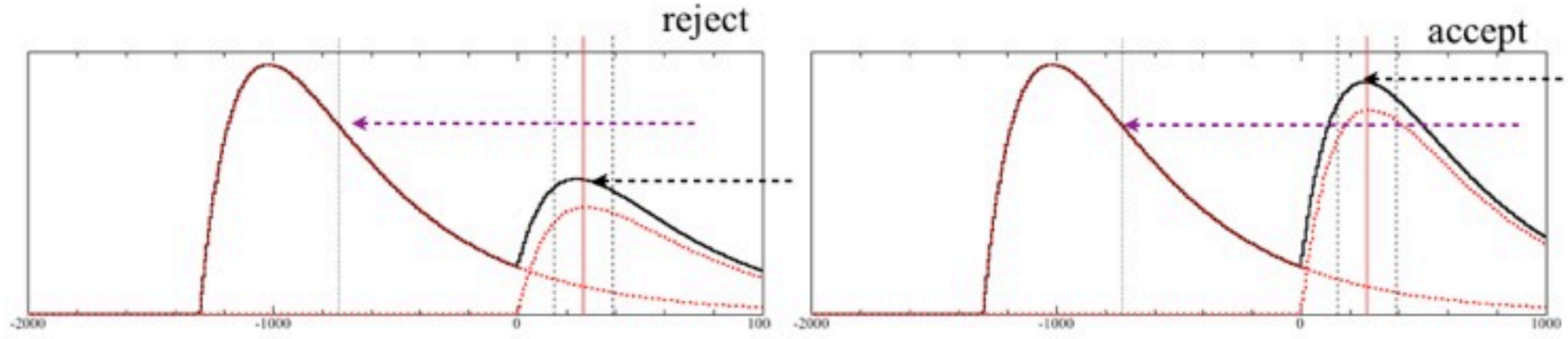


Lateral moment

Issues(3): Timing model

- Pulse model is calculated by an electronic simulator.
- Somewhat non-trivial way of treating pile-up from background:





- Is this the right/best way? Is what Stefano G. is doing in FullSim study a better model (ask him for detail)? Need some work and coordination.
- Need consistency between FastSim and FullSim in both timing model and cluster algorithm for meaningful comparison.

Issues(4): Background importing

- During background frame productions, particles are stored in a `TTree` as a `TClonesArray` of `TParticles`.
- FastSim reads a certain number of TParticles in succession, stores them in another `TClonesArray`, then converts them to `GTracks` (then proceed as usual).
- Current setup with 6 μs window (~1400 bunch crossings will include ~25k particles. If applying 5x safety factor, that's more than 120k particles.
- Each additional background particle increases the memory usage by ~15 kB, resulting >2GB memory usage under 5x background.
  - ✦ Jobs would fail if many cores sharing the same memory run the jobs concurrently.
  - ✦ There apparently is also a memory leak or some sort of memory overwriting that crashes the jobs, as we experienced sometimes at 5x background even without reaching the memory limit.

Potential fixes

1. Fix memory leaks (duh!). But it may not be trivial to identify the buggy code.
2. Eliminate duplicated information (`TParticles` v. `GTracks`).
3. Produce bgframe files with bare minimal information (x, y, z, p_x, p_y, p_z, type) in a flat format (text?); FastSim code then reads them in and stores (int/floats) in standard library containers, rather than using `TClonesArray` of `TParticles`.
4. Produce a crystal hit map for each bunch crossing by running a special FastSim job on bgframe or directly from FullSim production. FastSim then takes these maps as the bgframes; then directly mixes them with the crystal map produced by physics event; then proceeds with clustering.

Comments on 4.

- It bypasses the shower creation of background particles in FastSim; less CPU consumption.
- Memory usage may be more transparent and easy to control.
- Each bkg map is time integrated energy deposition for a bunch crossing, introducing correlation between fast and slow particles.
- Each map still needs to undergo timing model (issue(3)), not simply overlapping several maps.
- Each map is fixed after creation. One can fluctuate each crystal at time of mixing. Or, since there are many bunch crossings in each event, one may get enough random combinations anyway.
- For higher energy background, it's rare, so the statistics may be an issue (but not worse than current situation).
- If a cluster is created by a track, this correlation is lost unless the tracking devices use the bkg hits from the same bunch crossing.

Closing comments

- EMC in FastSim has worked reasonably well and has little development for a while.
- For shorter term, resolution issue and cluster size control can be improved with relatively small effort.
- Background issue limits the performance of FastSim production. We have several ideas that can potentially improve the performance (greatly), but it takes non-trivial human power and time.