

# Economizing in Geant4

Eric Church, LArSoft mtg, 23-Feb-2011.

# Problem: huge memory/CPU resources needed in Geant4

- These jobs in LArG4 module can get enormous.
- Jobs that run on EXPTgpvm01 begin to fall over en masse on condor local nodes.
- Seems to me we have 2GB/node, as Dennis has dialed it on local nodes. Can people please verify with top, e.g., that your job exceeds this and report back to me?

# LArVoxelReadOut.cxx

- Huge amount of cpu/wallclock time spent here for small voxels.
- There is a loop here over each tiny little step for each particle's energy deposition, in which the (x,y,z) is queried and a new LArVoxel object is created or (copied? and) contributed to.
- Potential quick solution for now: Increase your voxelSize

services.user.LArVoxelCalculator.VoxelSizeX: 0.3 # cm x10-30 larger than default.

services.user.LArVoxelCalculator.VoxelSizeY: 0.3 # cm

services.user.LArVoxelCalculator.VoxelSizeZ: 0.3 # cm

# Track-Stacking

- User has to implement three methods.
- **G4ClassificationOfNewTrack ClassifyNewTrack(const G4Track\*)**
  - Invoked every time a new track is pushed to G4StackManager.
  - Classification
    - **fUrgent** - pushed into Urgent stack
    - **fWaiting** - pushed into Waiting stack
    - **fPostpone** - pushed into PostponeToNextEvent stack
    - **fKill** - killed
- **void NewStage()**
  - Invoked when Urgent stack becomes empty and all tracks in Waiting stack are transferred to Urgent stack.
  - All tracks which have been transferred from Waiting stack to Urgent stack can be reclassified by invoking **stackManager->ReClassify()**
- **void PrepareNewEvent()**
  - Invoked at the beginning of each event for resetting the classification scheme.

# Track Stacking

This is now implemented in LArG4/LArG4.cxx using from flag at LArG4/largeantmodules.fcl.

```
lbne_largeant:
{
  module_type:      "LArG4"
  GenModuleLabel:   "generator"
  GeantCommandFile: "LArG4/LArG4.mac"
  DumpParticleList: false
  DumpLArVoxelList: false
  DebugVoxelAccumulation: 0
  VisualizeEvents:  false
  PMTSensitiveVolumeName: ""
  SmartStacking: true
}
```

# Track Stacking

LAr20MuonStackingAction.cxx : public G4UserStackingAction

```
LAr20MuonStackingAction::ClassifyNewTrack(const G4Track * aTrack)
```

```
{ //snippet
```

```
switch(stage)
```

```
{
```

```
case 0: // Stage 0 : Primary cosmic muons only
```

```
if(aTrack->GetParentID()==0)
```

```
{ classification = fUrgent; }
```

```
...
```

```
case N:
```

```
TString volName(InsideTPC(aTrack));
```

```
if(volName.Contains(geom->GetLArTPCVolumeName()) )
```

```
{
```

```
classification = fUrgent;
```

```
break;
```

```
}
```

```
else if (volName.Contains("unknown") )
```

```
{
```

```
classification = fKill;
```

```
break;
```

```
}
```

Track the primary cosmic

Everything in TPC is Urgent

Tracks in outer space killed.

# Track-Stacking

Kill the rock ionization electrons

```
if (! volName.Contains(geom->GetLArTPCVolumeName()) && aTrack->GetDefinition()-  
>GetPDGEncoding()==11 && ((TString)aTrack->GetCreatorProcess()->GetProcessName  
()).Contains("muIoni") )
```

```
{  
  classification = fKill;  
  break;  
}  
  classification = fUrgent;
```

```
//end snippet
```

declare all else as urgent

# Results

- 10000 cosmic events in LBNE/LAr20 used to take 5-10 days, if they had survived that long.
- These 2 things -- voxelSize increase and trackStacking - made them run in 20 minutes.
- Half, not all, of my condor jobs die.

# Future G4 Economizing

- Explore G4's Scoring for rare processes
- Look further into LArVoxelReadout. Can it be streamlined? Is there some horribly expensive operation currently that can be worked around so that voxelSize can again be reduced?