

TrigFTKSim status

- TrigFTKSim on the grid:
 - TSP and split run in 128 parallel jobs (fast!)
 - Patt-from-const production: extremely fast
 - Alberto quickly made a huge “SameAngle” bank
 - But: we discovered a bug last week:-(
 - Efficiency curve job now runs on the grid
- PBS bugfix:
 - Job dependency string must be $< \sim 60$ job Ids
 - Failed to submit track_merger of 128 jobs
 - Committed a workaround \rightarrow unlimited now

Split architecture profiling

- 4L roadfinder stage is really slow at 3E34:
 - Could be fixed by splitting into subregions
 - But this would break grid running
- **Why is it so slow? - *8L road/track access***

8L stage (30 min):

58.99	798.90	798.90	3	266.30	446.06	FTK_AMBank::road_warrior()
39.80	1337.95	539.05	2132488718	0.00	0.00	FTK_AMBank::informationMatch(int, int)

4L stage (370 min):

53.09	1337.47	1337.47	5263978	0.00	0.00	FTKRoadStream::findRoad(int, int) const
43.34	2429.24	1091.77	4147507069	0.00	0.00	FTKRoadStream::getRoad(int) const

- **Clearly, there is a lot of room for optimization!**

Minimum-overlap rmap

- Script bootstraps region map from sectors
 - By definition, produces min-overlap regions
 - By design, does NOT affect efficiency
 - Don't duplicate hits in the modules that are not present in ANY sector in the region
- Studied with 1E34 and 3E34 –similar effect
- New rmap is not a plug-in replacement:
 - Regions 1,2,3,4,5,6 work out of the box
 - Some more coding needed to handle reg 0 & 7
 - Easy to do, but low priority

Minimum-overlap results

Format: # clusters (old rmap) # clusters (min-overlap rmap) Ratio

1E34:

7669	5357	1.43158
9927	6829	1.45365
12803	8931	1.43355
8437	5645	1.4946
13026	9260	1.4067
8566	5831	1.46904
6567	4546	1.44457
9796	6955	1.40848
10301	7182	1.43428

Avg reduction (100 evts):
1.43199

3E34:

29798	20606	1.44608
30103	20605	1.46096
29845	20918	1.42676
32223	22150	1.45476
30570	21357	1.43138
24793	17471	1.41909
22570	15849	1.42406
29953	20732	1.44477
27621	19904	1.38771

Avg reduction (100 evts):
1.43074

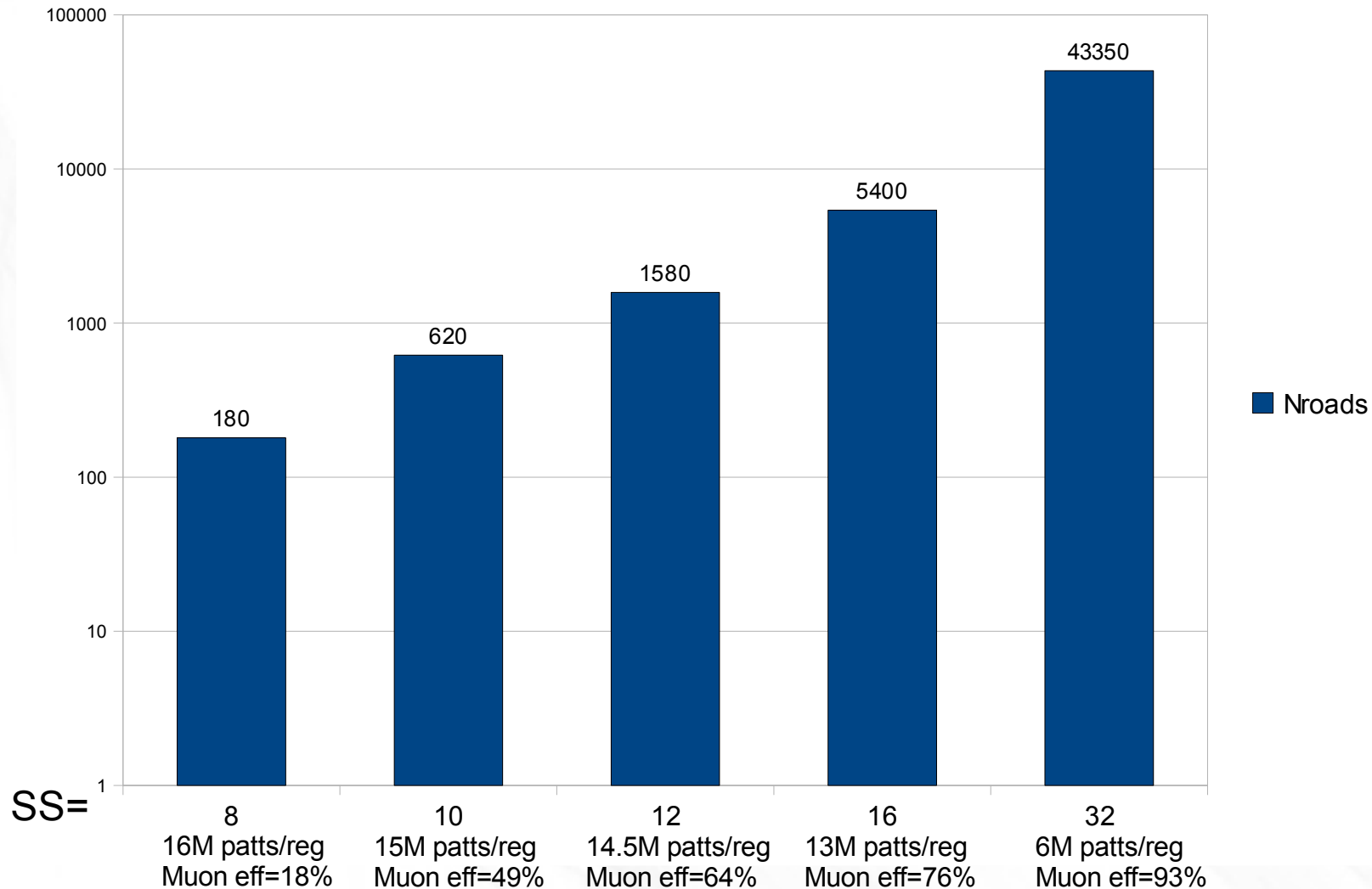
SUMMARY: we can route a **factor of 1.43 fewer hits** into each region

Note: this feeds into Naoki's simulation, but otherwise doesn't affect efficiencies/resolutions

Roads at 3E34

- We have a problem at 3E34 luminosity:
 - # roads is too large
- Alberto is studying the 11L TSP options
- I am taking a look at split architecture
- Today: 8L stage with $ss=32/16/12/10/8$
 - All banks are pattgen-only → low coverage
 - Moderate statistics: 100 3E34 Whbb events
 - Number of roads after RW (including x-sector)

of roads in 8L stage (AFTER RW)

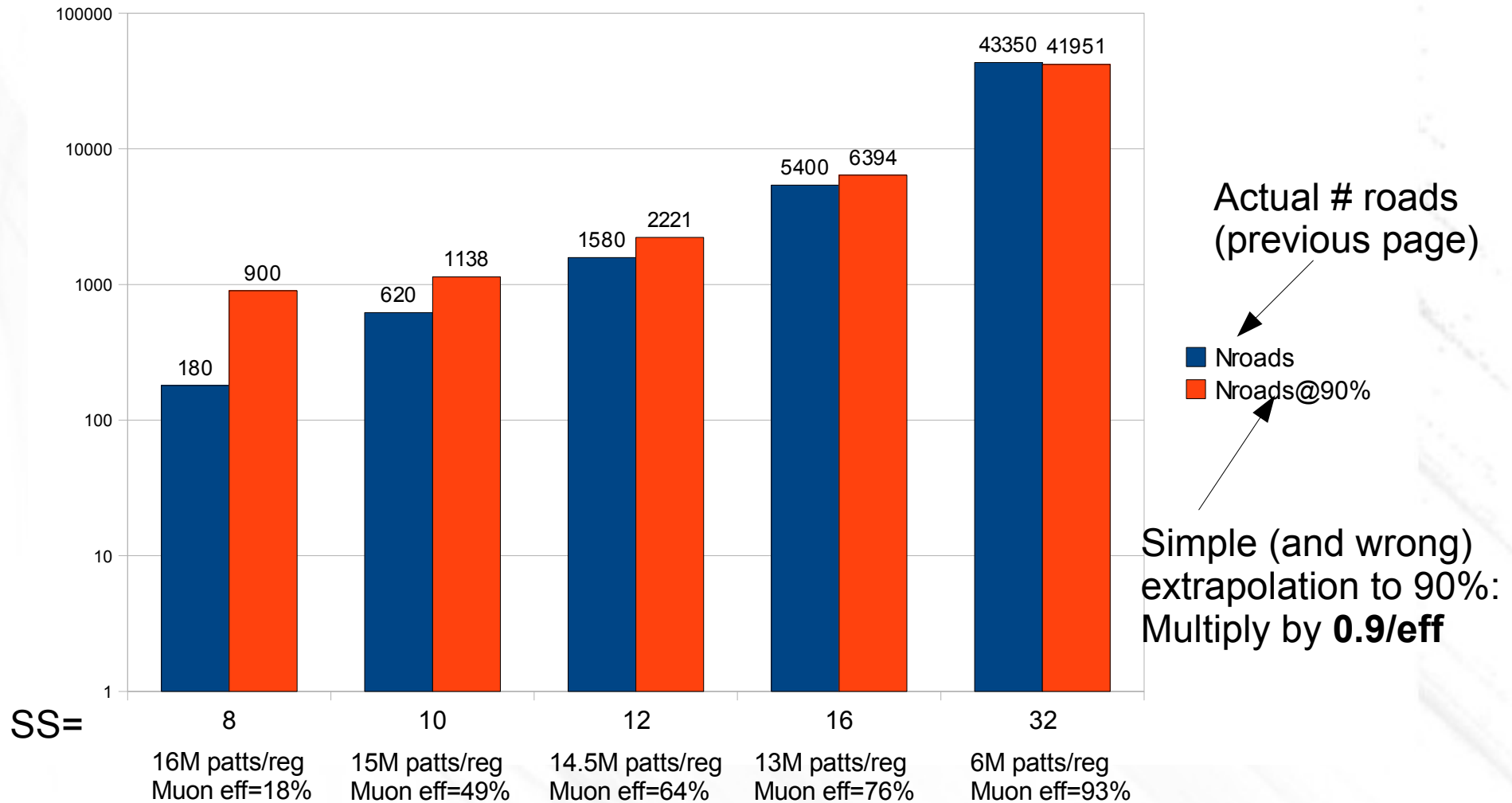


Due to time constraints, I couldn't make smaller-SS size with good efficiency.

Let's try to extrapolate #roads in small-SS banks to 90% efficiency.

For this simple exercise, **assume that #found roads is proportional to efficiency**

of roads in 8L stage (AFTER RW)



SCT superstrip sizes 8,10,12,16,32 (recall: module width = 768 strips)

Hardware limitations

- I need to clarify numbers with Paola/Alberto
- From Paola's email, it sounds like we want to have **1000-1500** roads per event (?)
 - possibly more if combined with TSP
- With the smallest SCT SS (ss=8), I optimistically extrapolated to **900 roads**
 - In reality we'll probably have more roads
- Plus, there is still 2nd stage: pix+SCTtrk
 - See next page!

4L stage: additional concerns

- I think our estimate of # 4L roads was off
 - Last time: Pixel SS=16x18 gives **10k roads** at 4L stage
- But some things may not be doable in the AM:
 - Sector lookup table inefficiency
 - Filtering double-misses: $\frac{7}{8}$ SCT + $\frac{3}{4}$ PIX
- Both of these types of roads were excluded from 10k count (i.e. filtered out).
- Next page for details

4L stage: sector lookup ineff


- In the original split arch, we have a 4L roads belonging to sector $sec4L$
- Each pseudo-layer hit represents an SCT track. These form a set of 8L sectors $\{sec8L1, sec8L2\}$
 - The hope is that we usually only have 1 SCT track per pseudo-layer superstrip
- The fits are done using 11L constants, so we have a lookup: $[sec4L, sec8L] \rightarrow sec11L$
- If $\{sec8L1, sec8L2, \dots\}$ all fail the lookup, the 4L road is rejected in simulation
 - This cannot be done in AM, but maybe before

Filtering double-misses: $7/8$ SCT + $3/4$ PIX

- If all 8L tracks in the pseudo-layer superstrip are majority ($7/8$), AND one of the pixel layers is missing in Pixels ($3/4$), we are rejecting this 4L road in the current simulation
- This procedure could possibly be done in the AM:
 - If each 8L track has an extra bit that says “isMajority”, and we take logical AND of all 8L tracks in the pseudo-layer SS

4L stage: # roads at 1E34

Our software currently filters out these roads
(i.e. they don't proceed to the fitting stage)



Event #	# AM roads	Sector lookup	7/8+3/4 filter	Quoted #roads
1	1594	921	400	273
2	8517	3475	2647	2395
3	3033	1545	400	1088
4	11382	5504	4005	1873
5	2209	912	528	769

If the number we need to look at is in the 1st column (# AM roads), we are in trouble.
Already at 1E34, it's in the thousands!

10k roads/event in 3E34 case
correspond to this column!

8L patt-from-const

- Patterns-from-const production at 8L makes a lot of junk → currently unusable
- Don't understand exactly why, but some ideas:
 - For each set of truth pars, sector lookup matches 30-50 different sectors
 - For each sector, we compute $x[8]$ and check that each one satisfies $0 < x < 768$. Sometimes a wrong sector will produce a pattern that by chance satisfies the criteria and is accepted
 - In the 11L case, it had to check $x[14]$. It's harder to have 14 numbers satisfy the criteria, than it is for 8 numbers

Proposed solution: bootstrapping from 11L patts

- We can get 8L patts from 11L patts:
 - Throw the dice ($\phi, z_0, \dots, \text{constraints}$) in 11L
 - Find matching 11L **sectors**
 - Make $x[14]$. Make sure all satisfy $0 < x < \text{MAX}$
 - Take SCT part of 11L sector. Use 8L sector lookup to find corresponding 8L secID
 - Save resulting 8L pattern and its secID
- Slight modification for 4L generation:
 - Use $x[14] * \{8\text{L constants}\}$ to compute FTK parameters that define pseudo-layer SS