Particle Physics in Real Time

Benjamin Nachman

Lawrence Berkeley National Laboratory

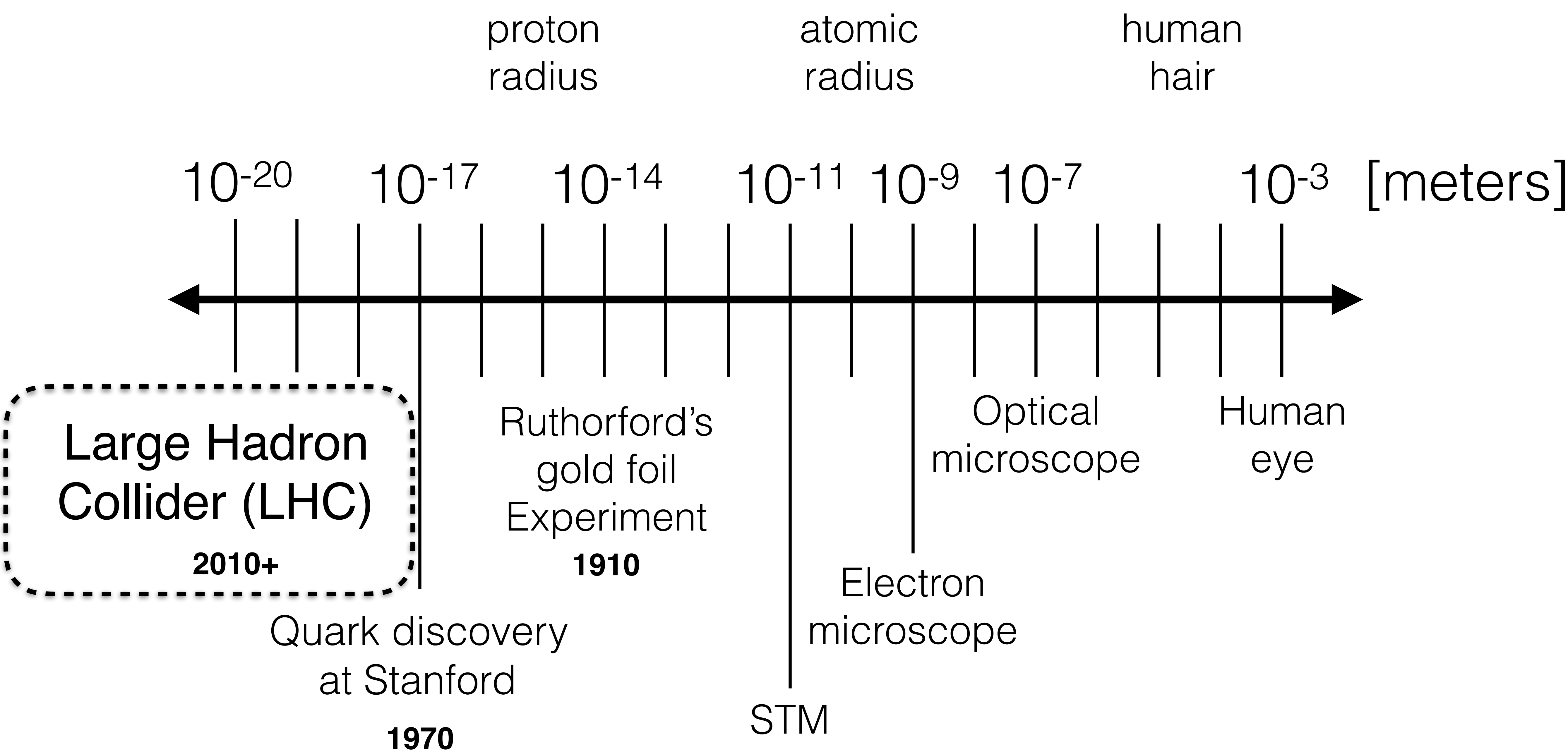
BERKELEY LAB

BERKELEY EXPERIMENTAL PARTICLE PHYSICS

Real-Time Decision Making Reunion, June 2, 2018

Science at the LHC

Goal: **We want to study the structure of the smallest building blocks of matter.** For this, we need the most powerful microscope ever built!



Data pipeline at the LHC

~few TB/s (99% thrown away in real time)

data / simulation

data	simulation	step
25 ns		O(100) *pp* collisions
10^{-19}-10^{-15} s	~ms	(sub-)nuclear physics
0.01-20 ns	~min	out-going particles interact with detector
1-100 ns	~100 ms	detector response (signal formation + digitization)
2.5 µs		hardware-based trigger decision
200 ms		software-based trigger decision
~100 ms		event reconstruction
		event processing (skim, thin, augment)
~months		final data analysis (uses millions of events)

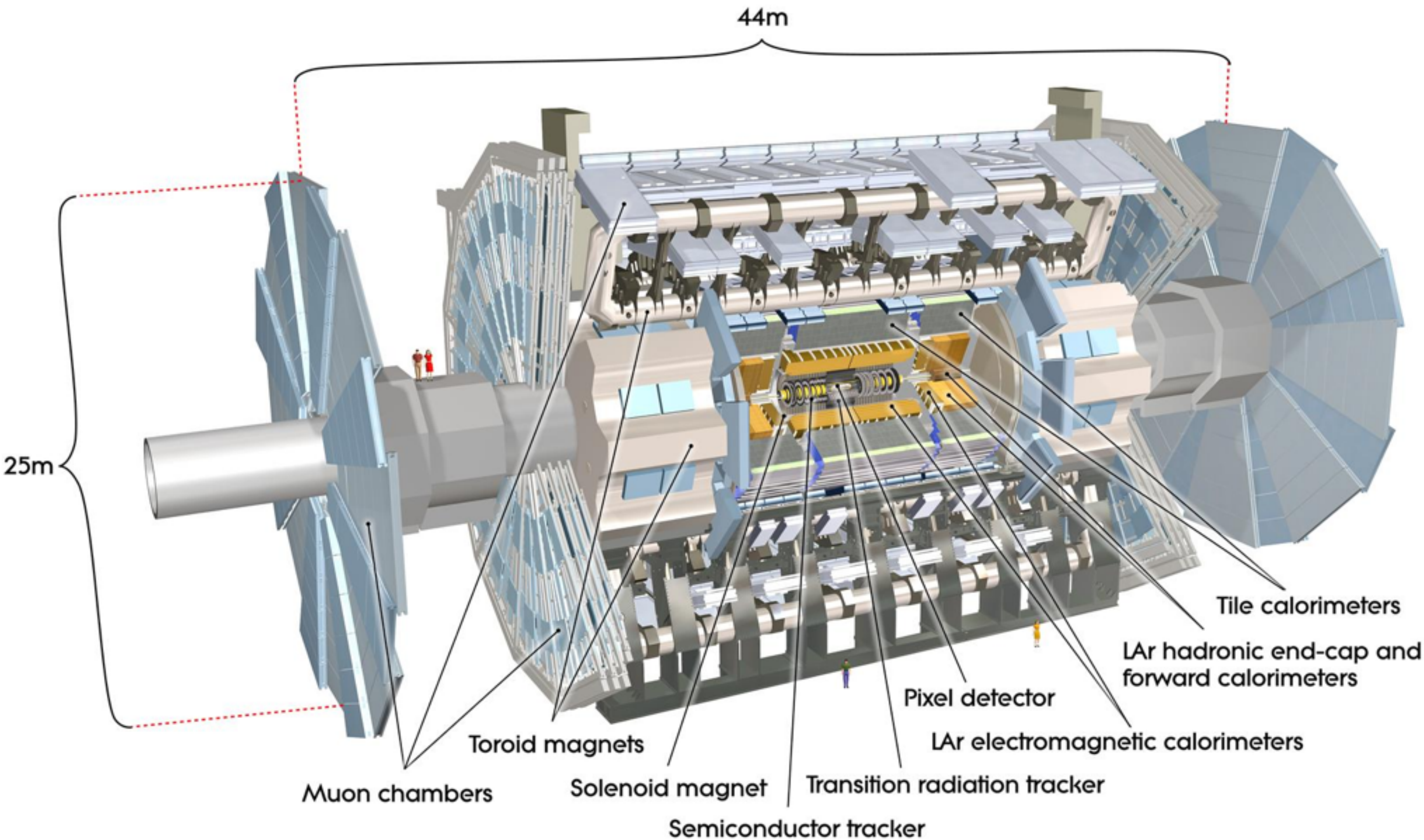
Data pipeline at the LHC

~few TB/s (99% thrown away in real time)

data / simulation

25 ns		O(100) *pp* collisions
10^{-19}-10^{-15} s	~ms	(sub-)nuclear physics
0.01-20 ns	~min	out-going particles interact with detector
1-100 ns	~100 ms	detector response (signal formation + digitization)
2.5 µs		hardware-based trigger decision
200 ms		software-based trigger decision
~100 ms		event reconstruction
		event processing (skim, thin, augment)
~months		final data analysis (uses millions of events)

Collider-based HEP detectors are like leeks



~100 million readout channels

Real time challenges

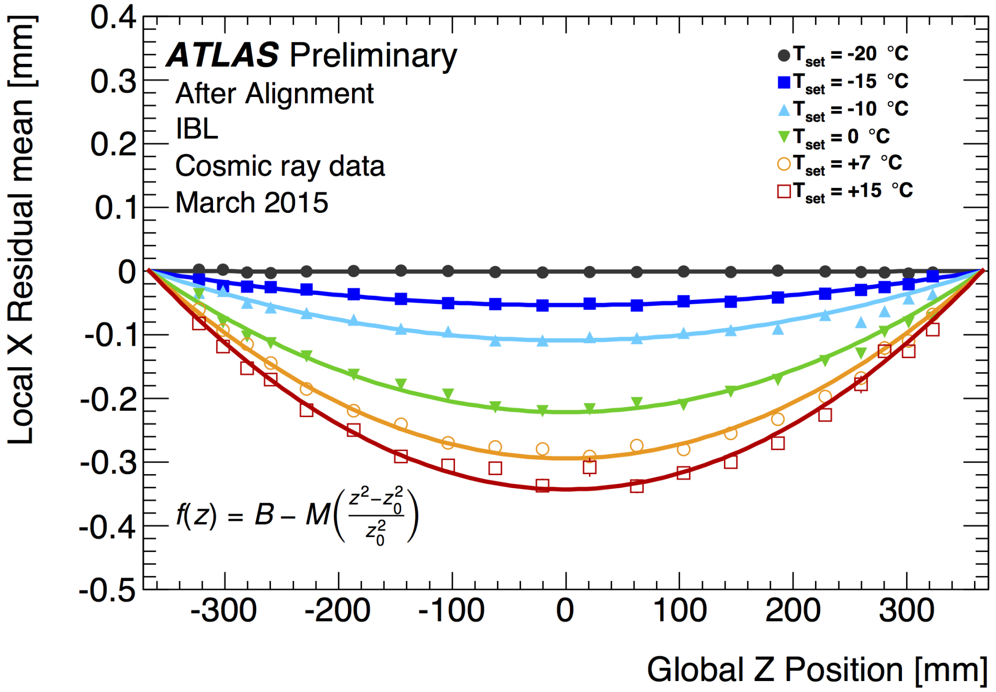
First decisions have to be on-detector and only with local information.

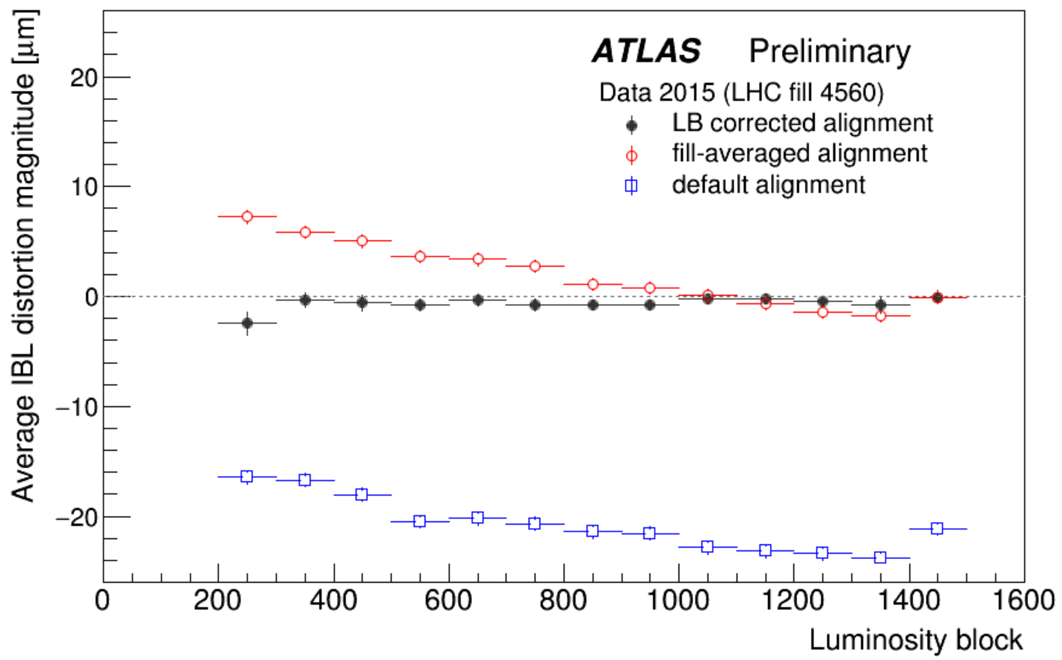
(also need to encode & buffer)

Detector is not constant with time - real time online calibration

Want to use offline-like algorithms as soon as possible

…but don't have time or resources to run all our deepest NN's, etc.





Real time tools

Application Specific Integrated Circuit

Fast decisions with incomplete information (ASICs)

On-detector, radiation hard and ultra fast - single purpose hardware

Field Programmable Gate Array

Fast decisions with full information (FPGAs)

Off-detector, re-programmable

(less) Fast decisions with full information (Software)

Far off-detector, offline-like algorithms

Real time tools

Application Specific Integrated Circuit

Fast decisions with incomplete information (ASICs)

On-detector, radiation hard and ultra fast - single purpose hardware

← I'll tell you a story about this for the rest of the talk

Field Programmable Gate Array

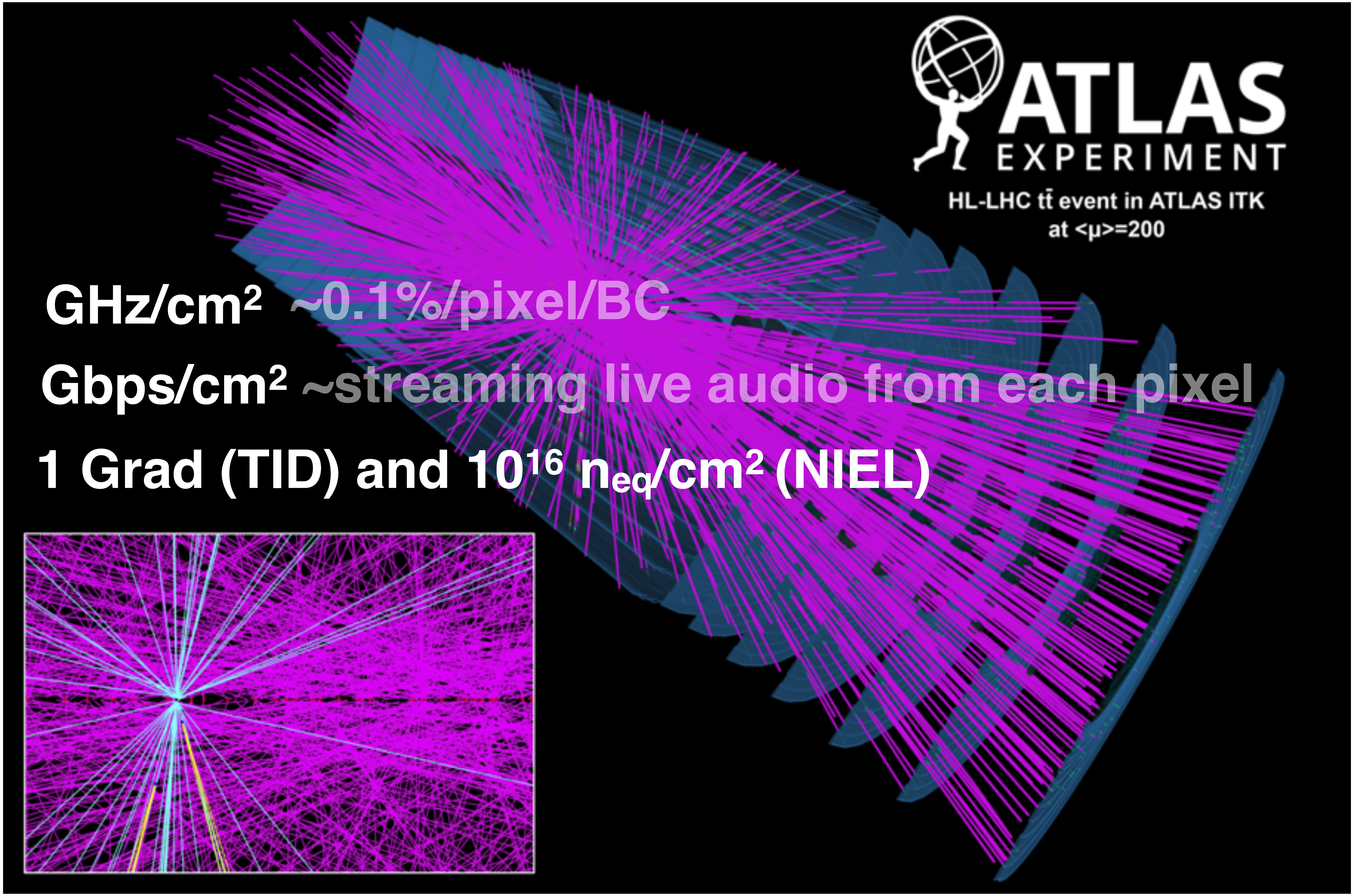
Fast decisions with full information (FPGAs)

Off-detector, re-programmable

(less) Fast decisions with full information (Software)

Far off-detector, offline-like algorithms

Innermost layer: high bandwidth, hit rate, rad. damage



GHz/cm² ~0.1%/pixel/BC

Gbps/cm² ~streaming live audio from each pixel

1 Grad (TID) and 10^{16} n_{eq}/cm^2 (NIEL)

Innermost layer: high bandwidth, hit rate, rad. damage

Generation	Run 1 (FEI3, PSI46)	Runs 2+3 (FEI4, PSI46DIG)	Runs 4+5
Chip Size	7.5 x 10.5 mm^2 8 x 10 mm^2	20 x 20 mm^2 8 x 10 mm^2	> 20 x 20 mm^2
Transistors	3.5 M 1.3 M	87 M	~1 G
Hit Rate	100 MHz/cm^2	400 MHz/cm^2	~2 GHz/cm^2
Hit Memory / Chip	0.1 Mb	1 Mb	~16 Mb
Trigger Rate	100 kHz	100 kHz	200 kHz - 1MHz
Trigger Latency	2.5 μs 3.2 μs	2.5 μs 3.2 μs	6 - 20 μs
Readout rate	40 Mb/s	320 Mb/s	1-4 Gb/s
Radiation	100 Mrad	200 Mrad	1 Grad
Technology	250 nm	130 nm 250 nm	65 nm
Power	~1/4 W/cm^2	~1/4 W/cm^2	1/2 - 1 W/cm^2

Innermost layer: high bandwidth, hit rate, rad. damage

Generation	Run 1 (FEI3, PSI46)	Runs 2+3 (FEI4, PSI46DIG)	Runs 4+5
Chip Size	7.5 x 10.5 mm^2 8 x 10 mm^2	20 x 20 mm^2 8 x 10 mm^2	> 20 x 20 mm^2
Transistors	3.5 M 1.3 M	87 M	~1 G
Hit Rate	100 MHz/cm^2	400 MHz/cm^2	~2 GHz/cm^2
Hit Memory / Chip	0.1 Mb	1 Mb	~16 Mb
Trigger Rate	100 kHz	100 kHz	200 kHz - 1MHz
Trigger Latency	2.5 μs 3.2 μs	2.5 μs 3.2 μs	6 - 20 μs
Readout rate	40 Mb/s	320 Mb/s	1-4 Gb/s
Radiation	100 Mrad	200 Mrad	1 Grad
Technology	250 nm	130 nm 250 nm	65 nm
Power	~1/4 W/cm^2	~1/4 W/cm^2	1/2 - 1 W/cm^2

e.g. the camera in your phone on steroids, next to a nuclear reactor

(unfortunately, Apple doesn't make one of these)

Pixel ASIC for the innermost layer of the LHC detectors

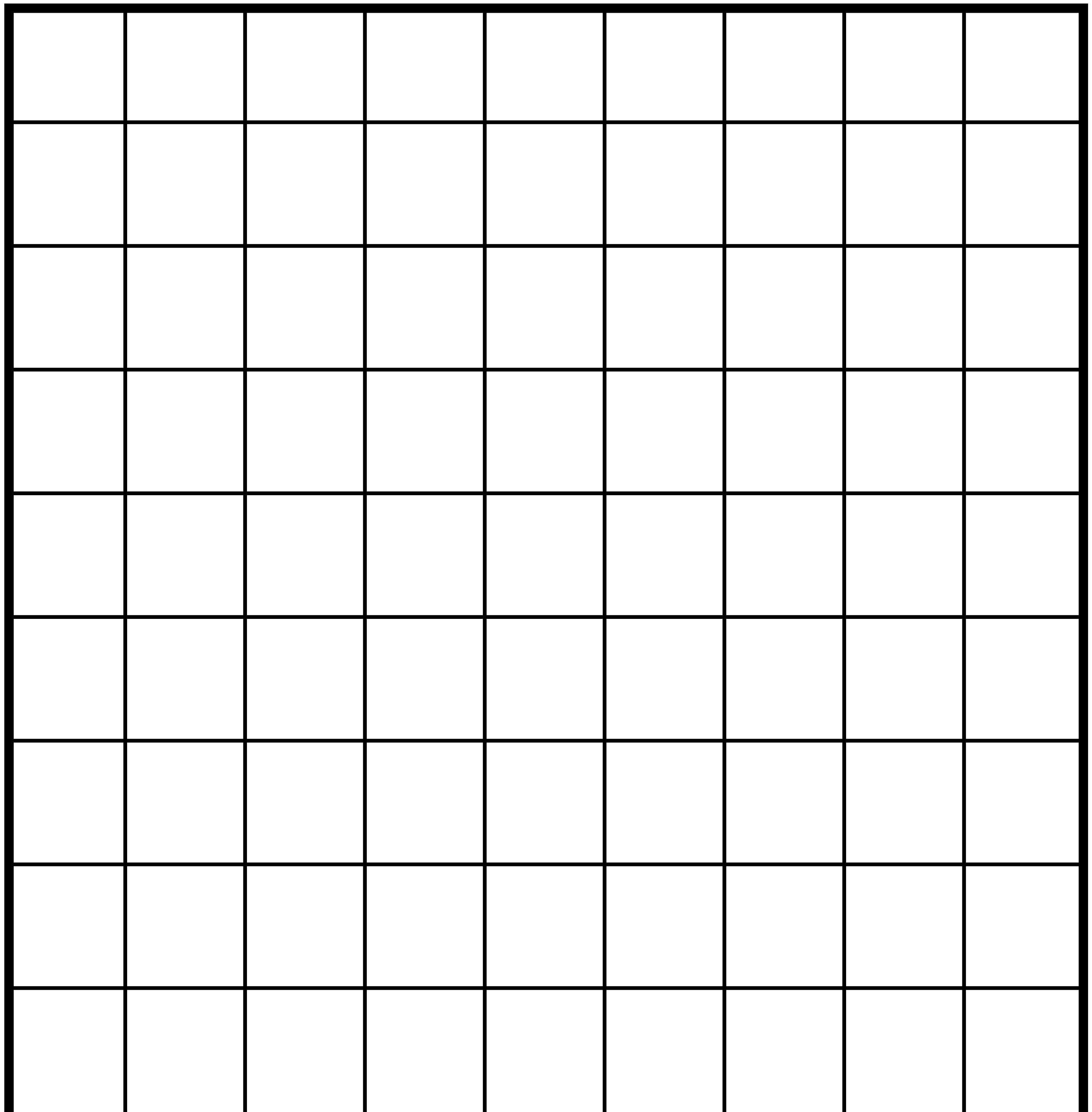
Specific RTDM Problem: Find hits, ignore noise

Top down view of pixel detector

iid Gaussian noise in each pixel

rate requirement on what we can read out

record only what exceeds a threshold



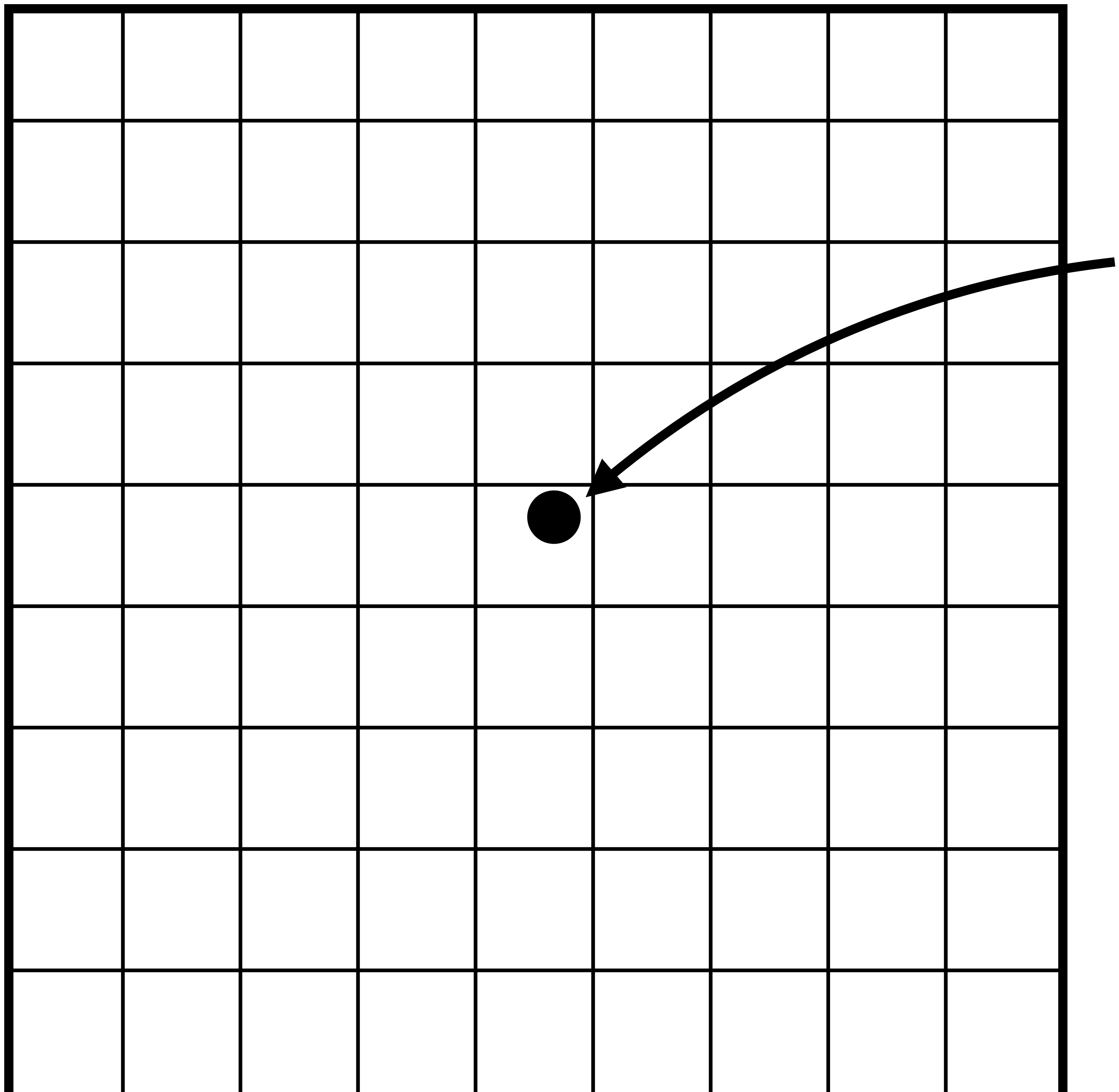
Specific RTDM Problem: Find hits, ignore noise

Top down view of pixel detector

iid Gaussian noise in each pixel

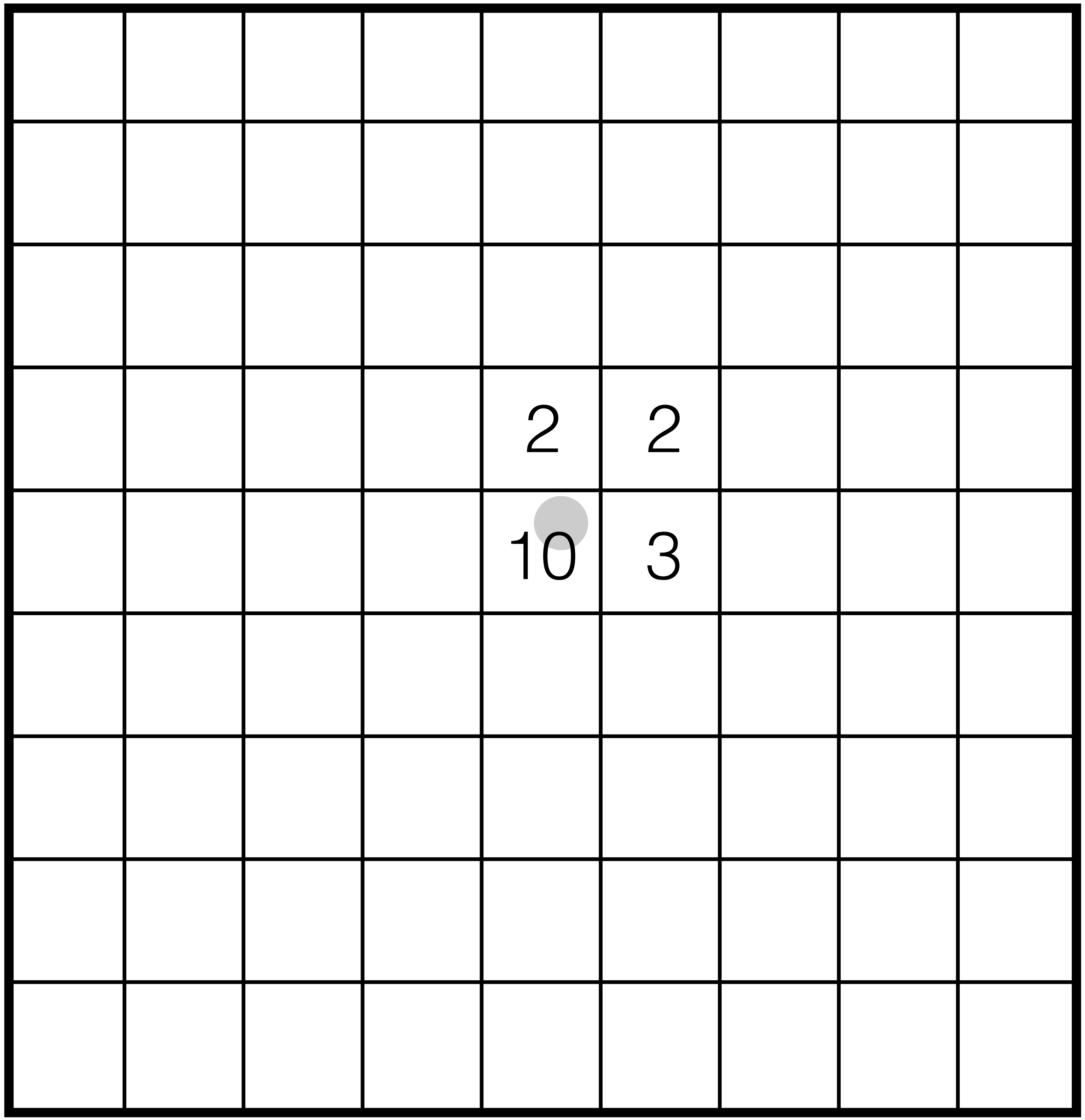
rate requirement on what we can read out

record only what exceeds a threshold



real particle goes through the detector

Specific RTDM Problem: Find hits, ignore noise



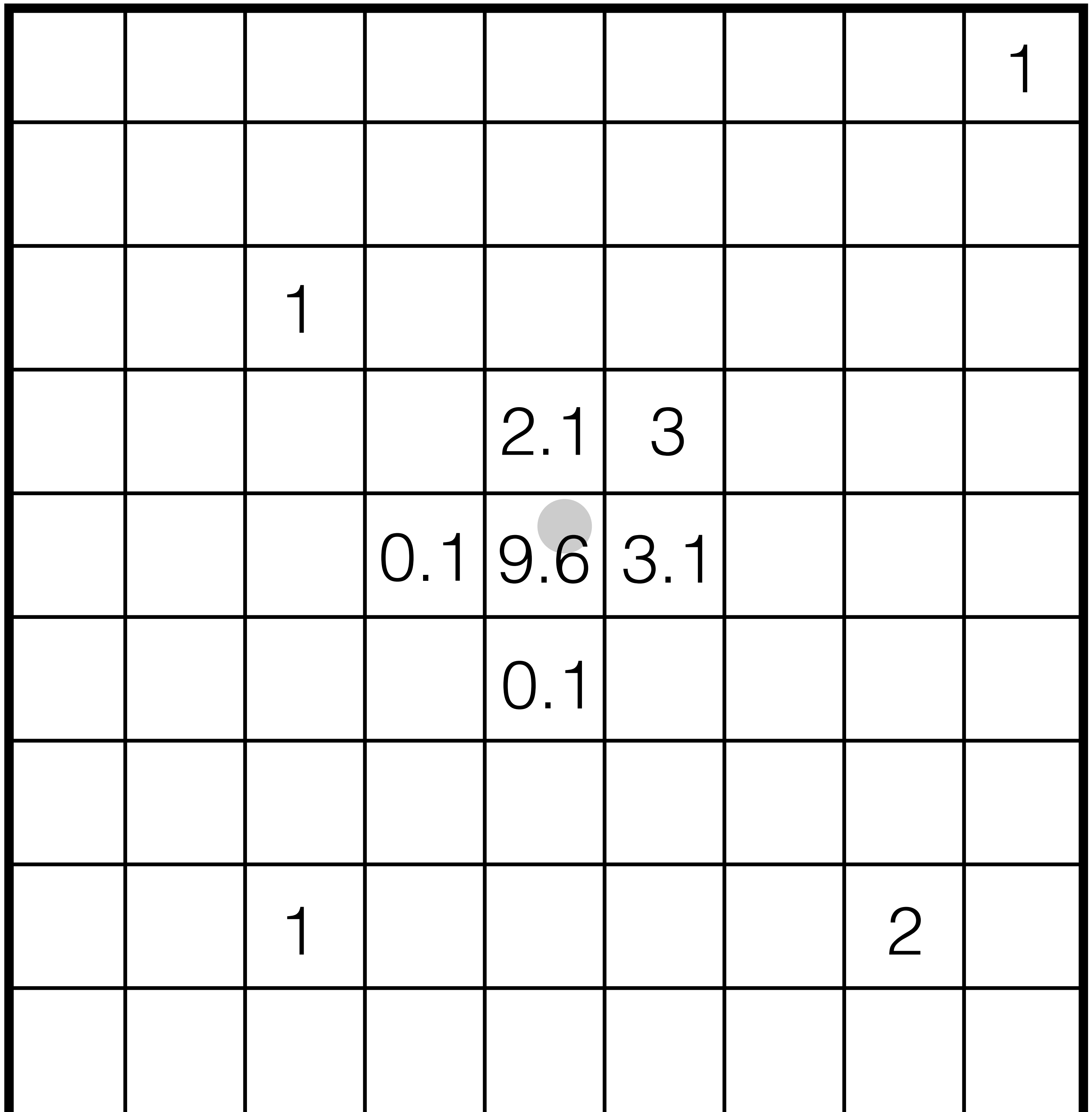
charge **deposited** by the particle

Specific RTDM Problem: Find hits, ignore noise

								1
		1						
				2	3			
				10	3			
		1					2	

apparent charge after adding **noise**

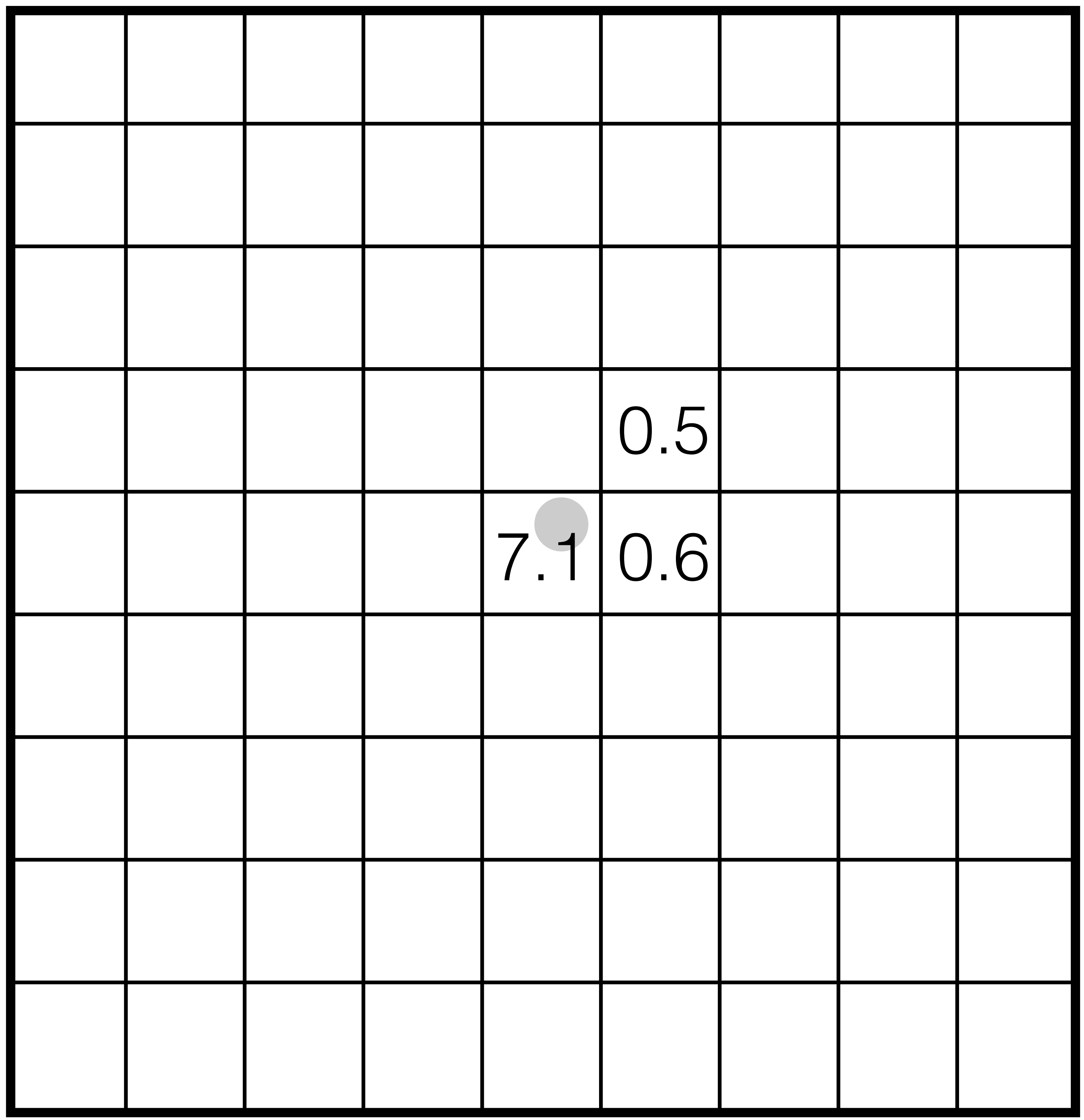
Specific RTDM Problem: Find hits, ignore noise



apparent charge after **charge sharing**

(diffusion + electronics are capacitively coupled)

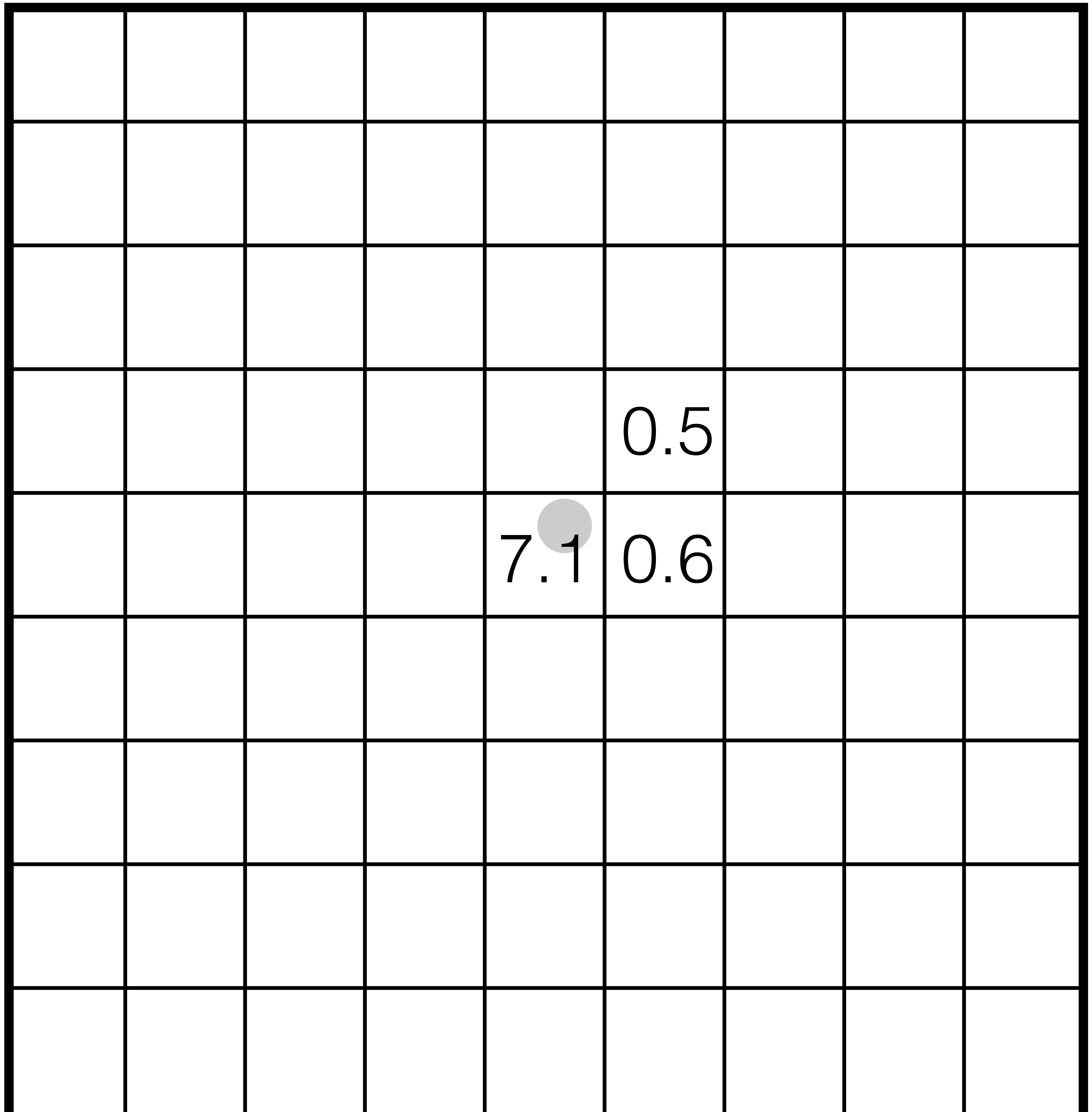
Specific RTDM Problem: Find hits, ignore noise



charge over threshold that is **observed**

(threshold = 2.5)

Specific RTDM Problem: Find hits, ignore noise



charge over threshold that is **observed** (threshold = 2)

Question: can we do better?

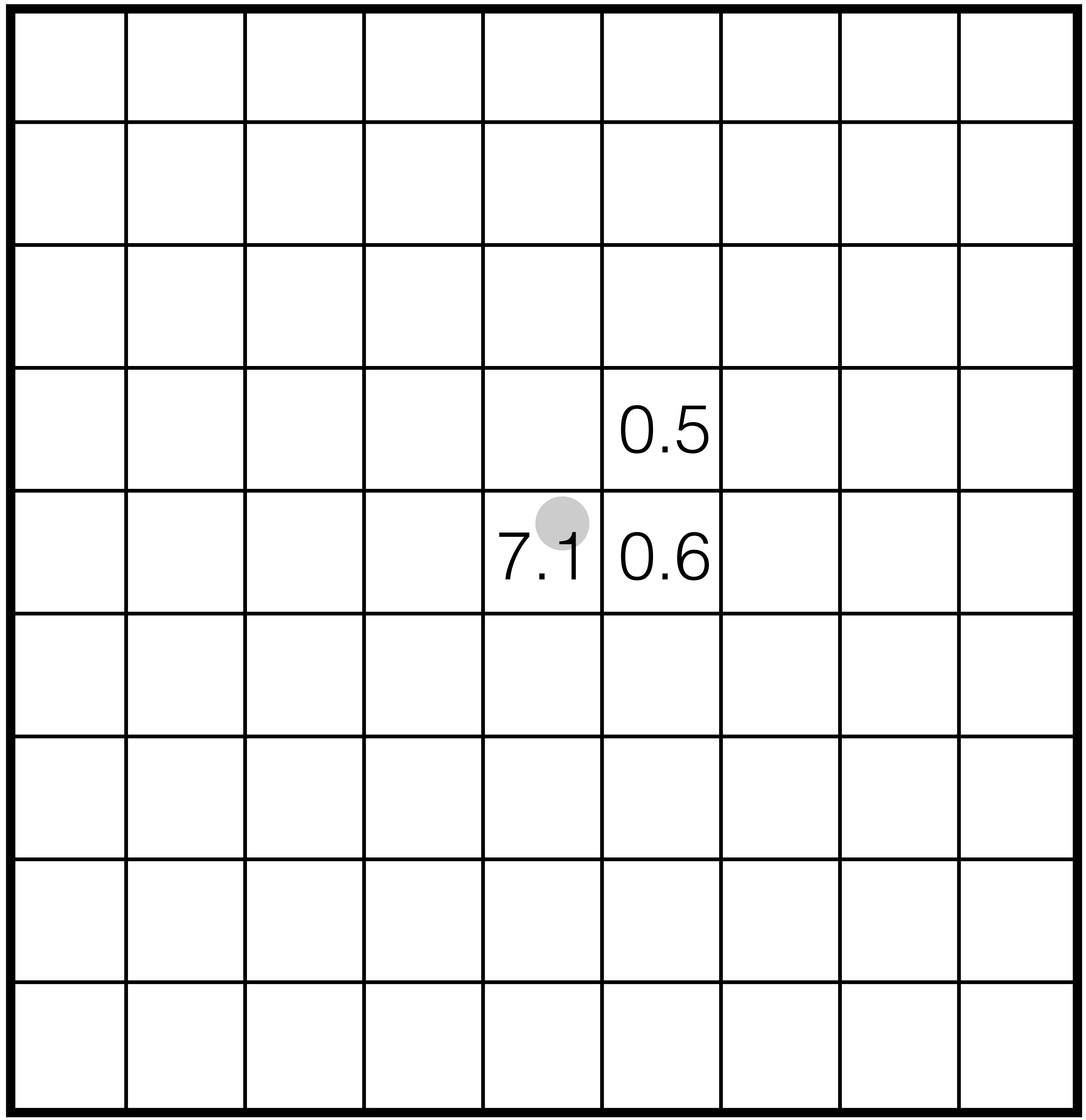
Specific RTDM Problem: Find hits, ignore noise

Facts

Prob(hit from real particle) << 1

Prob(hit | next to pixel from real particle) ~ 1

Dynamic thresholds?



charge over threshold that is **observed** (threshold = 2)

Two charge sharing schemes

Option 1:

One parameter: f_{share}

As a result of capacitative coupling, a charge Q on one pixel adds fQ on neighbors. f depends on length of shared edge and is ~few %.

Usually want this to be small, but maybe can gain by artificially increasing it?

$$Q_{primary} + fQ_{neighbor} > \text{threshold}$$

$$\updownarrow$$

$$Q_{primary} > \text{threshold} - fQ_{neighbor}$$

(effectively lowers threshold)

N.B. Pr(hit) << 1 but Pr(hit | neighbor) ~ 1

Two charge sharing schemes

Option 2:

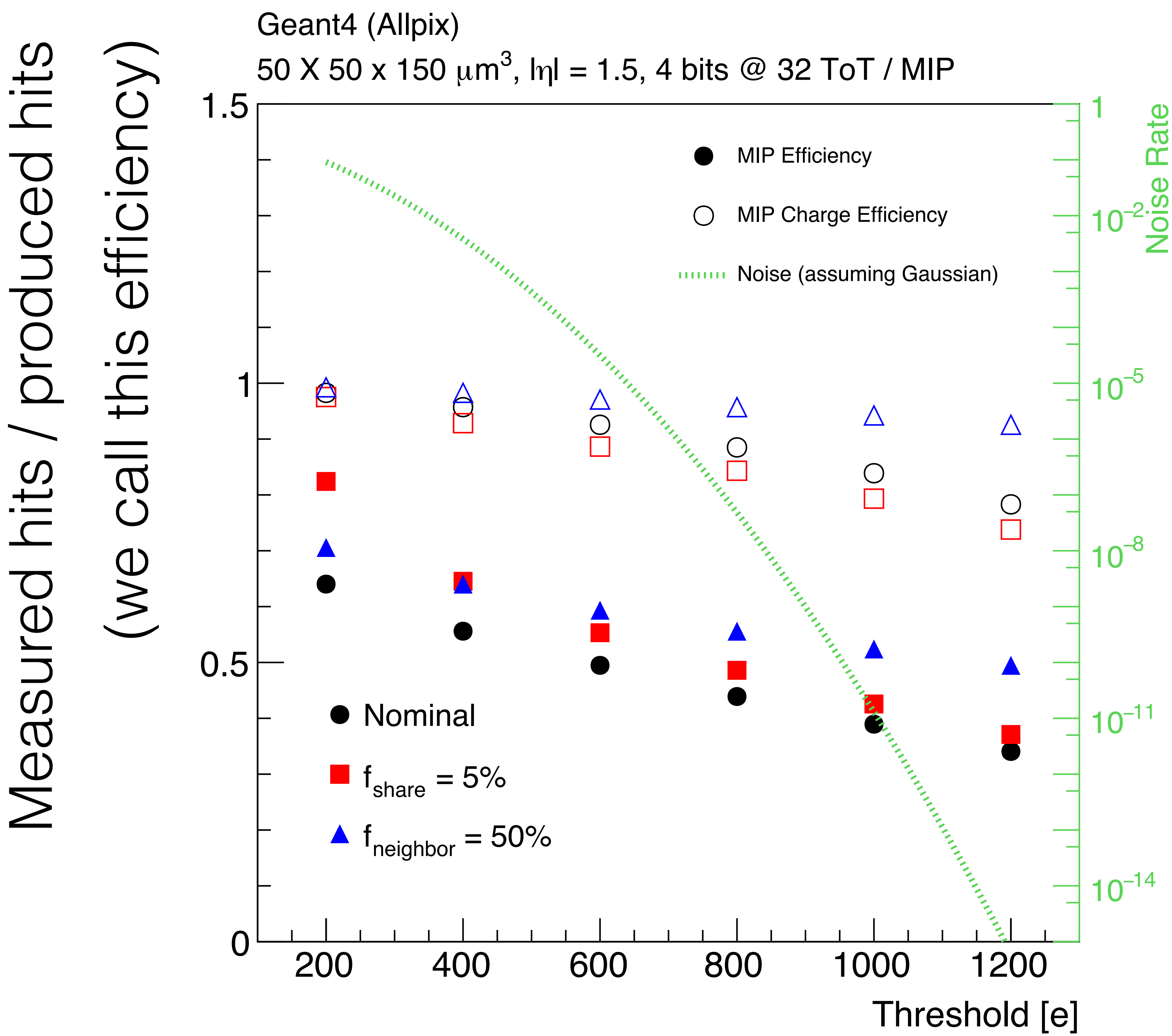
Whenever a pixel is above threshold, lower the threshold of the neighbors.

One parameter: $f_{neighbor}$

This is hard(er) to implement in practice because it requires more active logic (which means more power &/or more memory)

N.B. these are quite simple, but I'll show that they work well. Can probably do even better by using less local information.

Charge sharing as a dynamic threshold



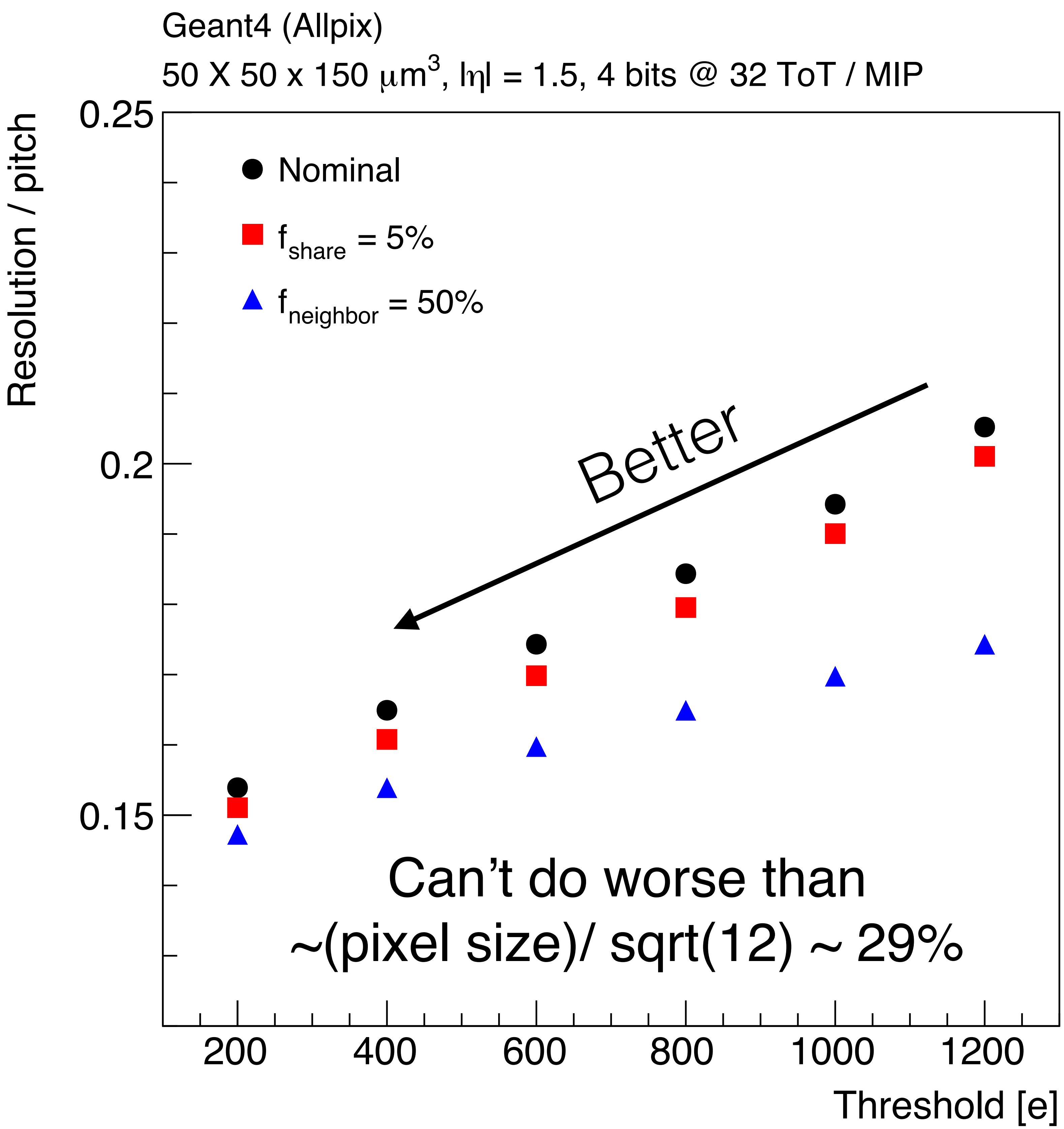
Filled: #

Open: charge-weighted

Scheme 1: Give a fraction f_{share} of your charge to your neighbor

Scheme 2: Set the threshold of your neighbor to $f_{neighbor}$ of your threshold.

Charge sharing as a dynamic threshold



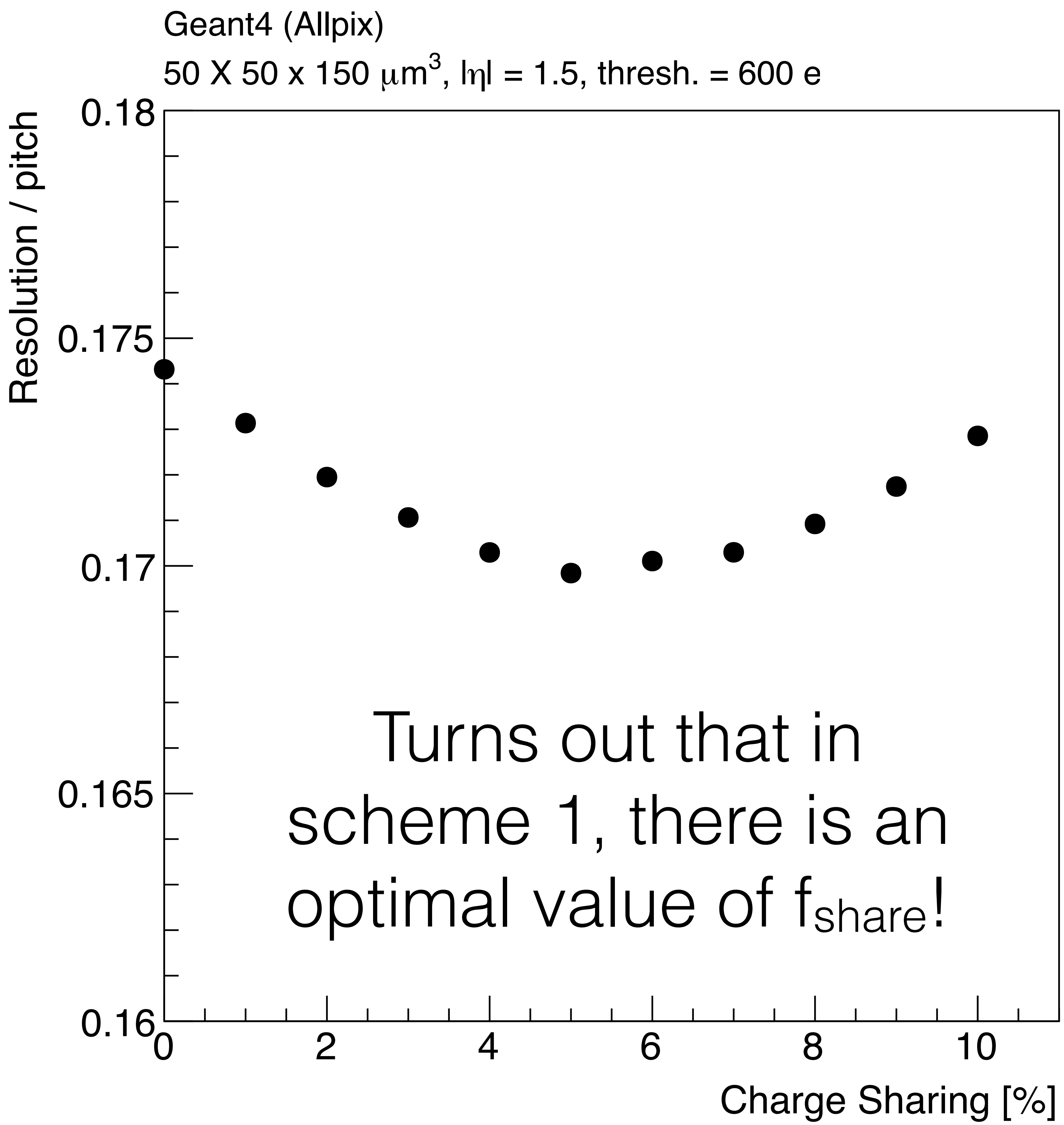
Position resolution improves when more information is kept

estimated position = weighted average over hit pixels

Scheme 1: Give a fraction f_{share} of your charge to your neighbor

Scheme 2: Set the threshold of your neighbor to $f_{neighbor}$ of your threshold.

Charge sharing as a dynamic threshold



Turns out that in scheme 1, there is an optimal value of f_{share}!

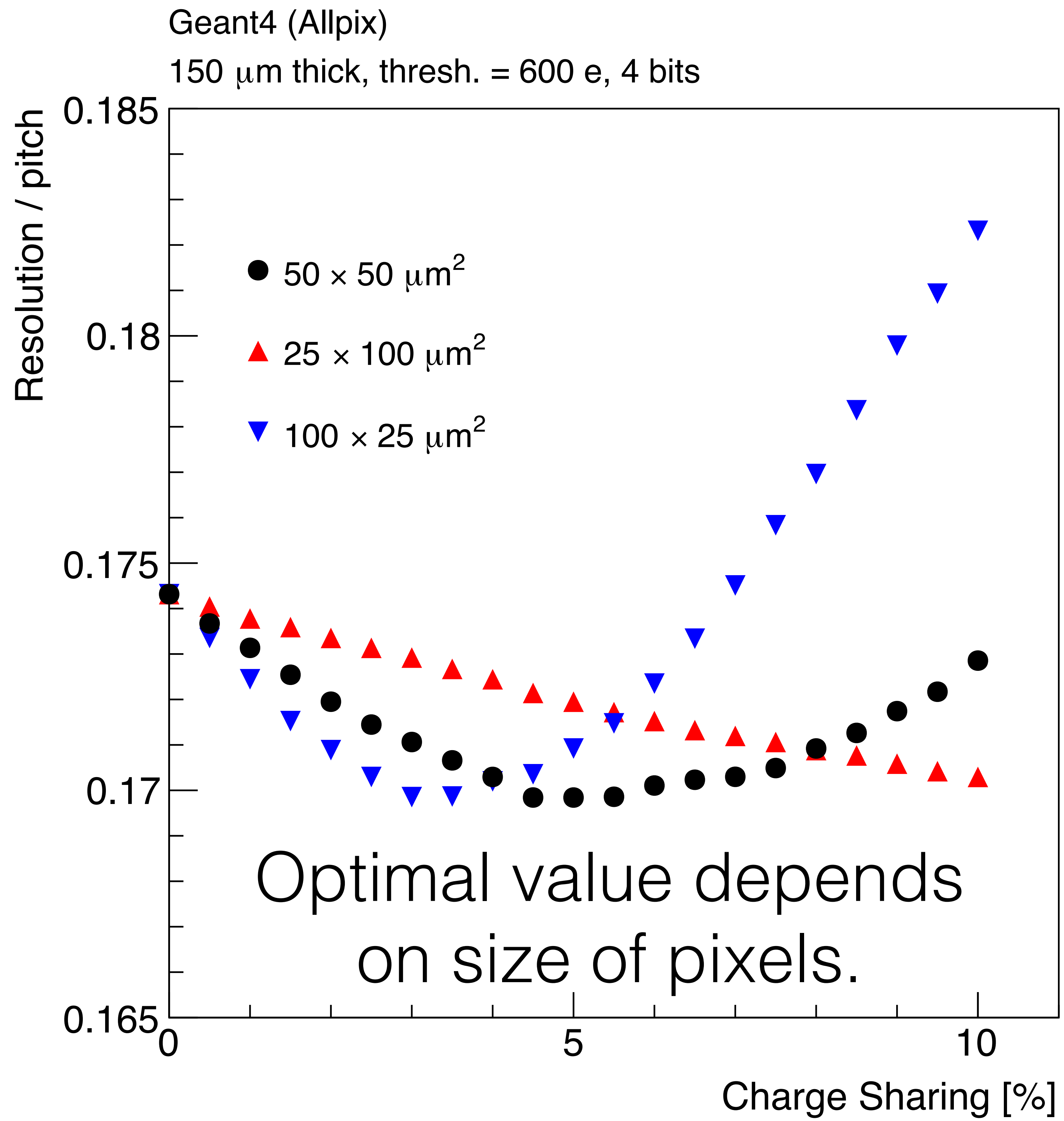
Position resolution improves when more information is kept

estimated position = weighted average over hit pixels

Scheme 1: Give a fraction f_{share} of your charge to your neighbor

Scheme 2: Set the threshold of your neighbor to $f_{neighbor}$ of your threshold.

Charge sharing as a dynamic threshold



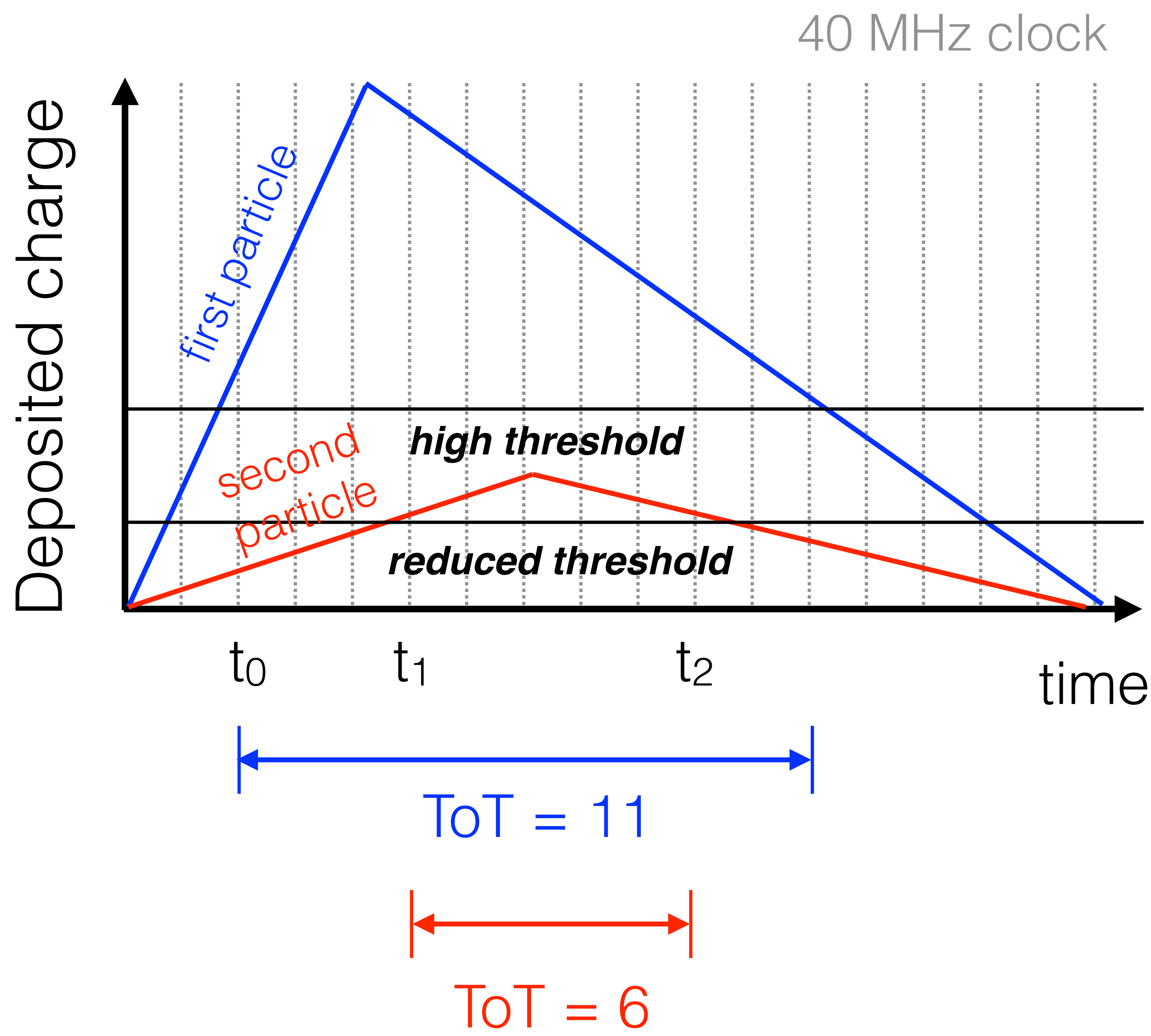
Position resolution improves when more information is kept

estimated position = weighted average over hit pixels

Scheme 1: Give a fraction f_{share} of your charge to your neighbor

Scheme 2: Set the threshold of your neighbor to $f_{neighbor}$ of your threshold.

Implementation



ToT = time over threshold

Scheme 1: Give a fraction f_{share} of your charge to your neighbor

…information transferred ~instantly to neighbors

Scheme 2: Set the threshold of your neighbor to $f_{neighbor}$ of your threshold.

…need time to tell neighbor to lower threshold.

Penultimate slide: timeline

Now

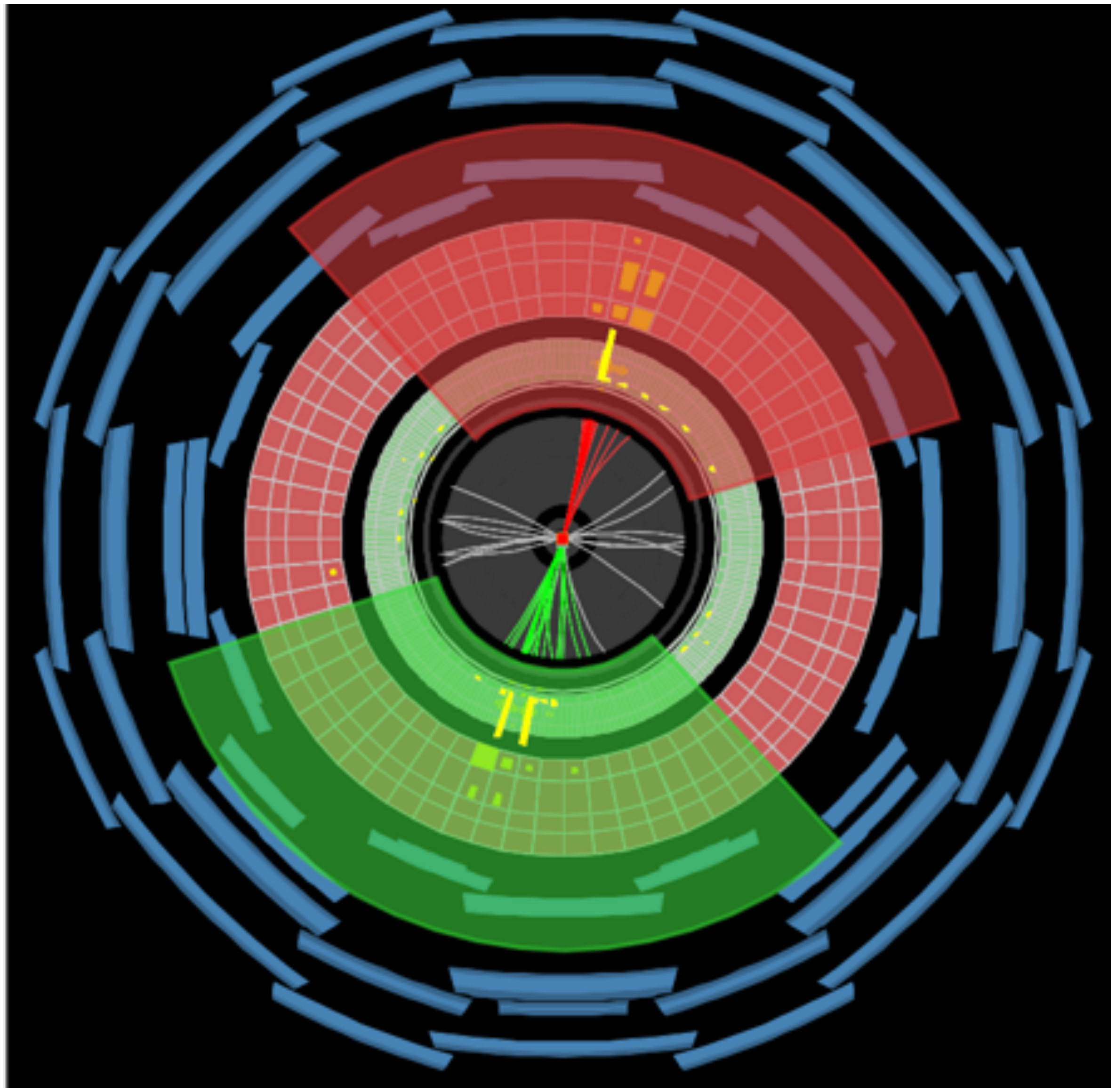
Firmware + software updates

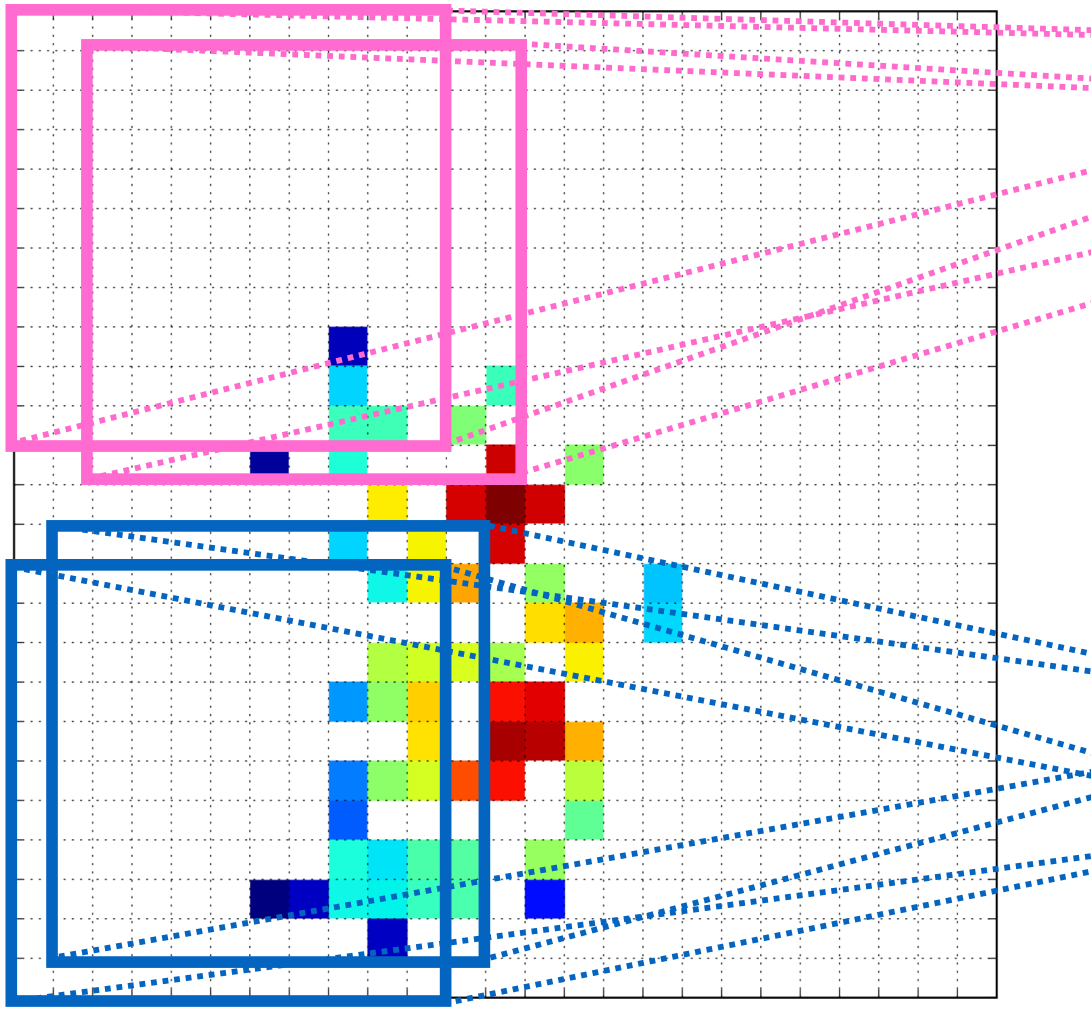
Hardware updates

Software updates ~only!

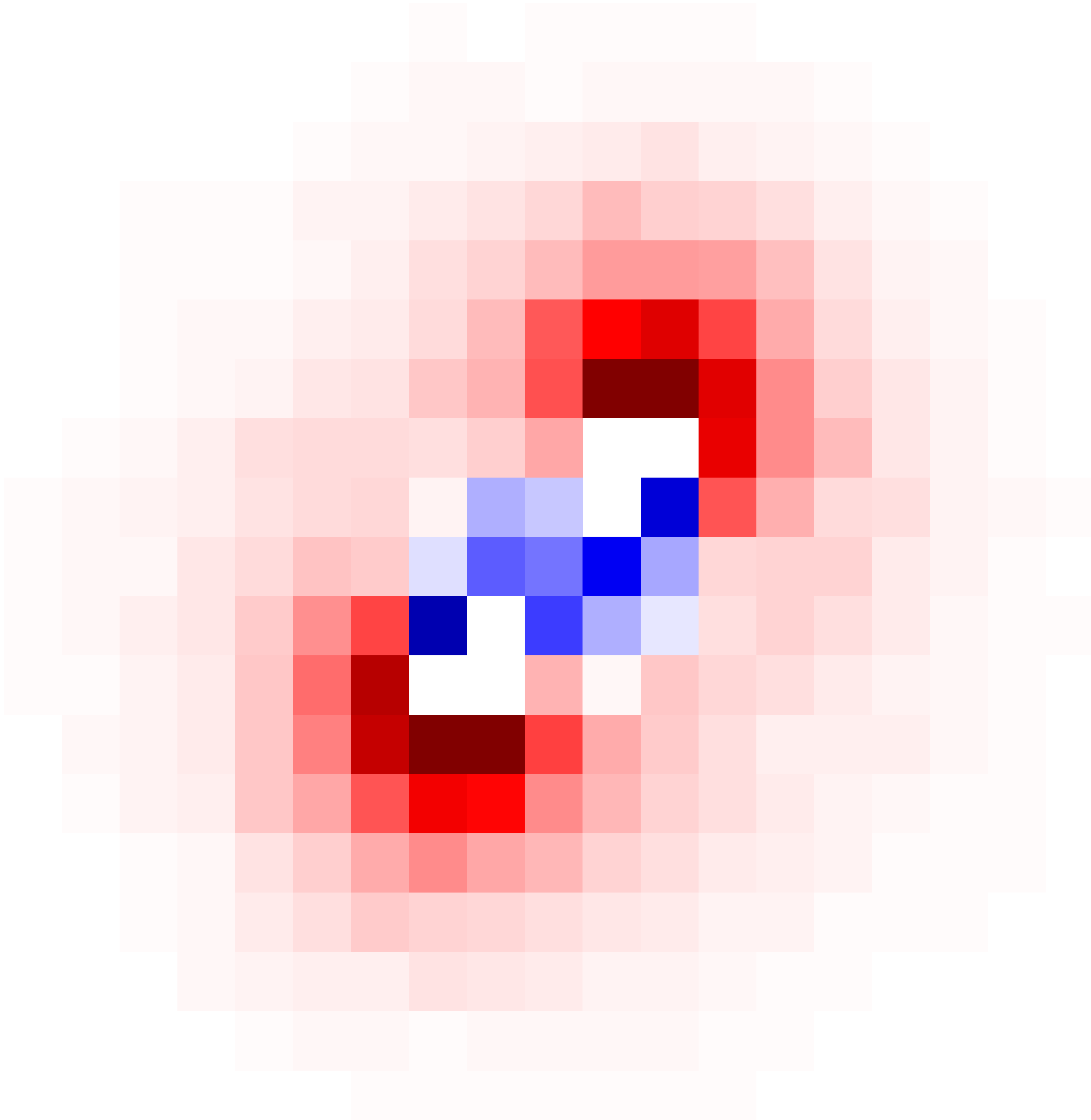
Conclusions and outlook

The LHC is a unique science tool with extreme challenges related to the data rate: real time / ultra fast algorithms are required.





There are many exciting opportunities and ideas for fully exploiting our data
we must make sure no stone is left unturned !



Fin.