# Introduction

This document discusses ideas for speeding up processing of readout data for the Gd-GEM detector by using multiple cores and/or EFUs.

# Speedup by partitioning

A detector panel consist of 1280 channels (strips) in both the x- and y- directions. It will be segmented into four ’modules’ but will share the gas volume so particle tracks can span one to four modules.

One speedup method would be to partition the detector in ‘square’ regions, and to assign all readout data from each region to a dedicated CPU or EFU. This would, potentially, speed processing up by a factor corresponding to the number of regions. Thus for a 2x2 segmentation a maximum speedup of four is achieved, etc.

## Module partition – no overlap

The simplest reasonable case for segmenting the detector panel is to follow the layout of the modules (2x2) and ignoring the borders between the modules. This will speedup processing by a factor of four but it will give a lower count rate in the overlapping regions.



Figure 1: Detector panel with the four module segments.

## Module partition - Including overlap

Figure 2 shows the four modules of a panel and the border regions belonging to module B. Readout data from overlap regions in modules A, C and D must to be duplicated and sent to B. Likewise for the other regions.



Figure 2: Detector overlap regions

Assuming for the calculations that the detector area receives a constant rate R0 of neutrons per unit area, module B will receive a higher rate corresponding to the ‘new’ larger area which includes the overlaps. The total detector rate will now be

(where *N*=640), and depends only on *n*. For *n*=20 the total rate is 1.063R0. However this rate will be processed by four cores thus the theoretical maximum speedup is 3.75.

For larger number of partitions (here assuming a square layout) the formula is slightly more complicated due to the fact that we have three different overlap regions: corner-, edge- and internal regions which all have different areas. The rate is

Note that *N = 1280/M*, which only produces equal and square partitions for those values of *M* that divide 1280. As the overlap regions increase with *n* so does the rate as shown in Figure 3.

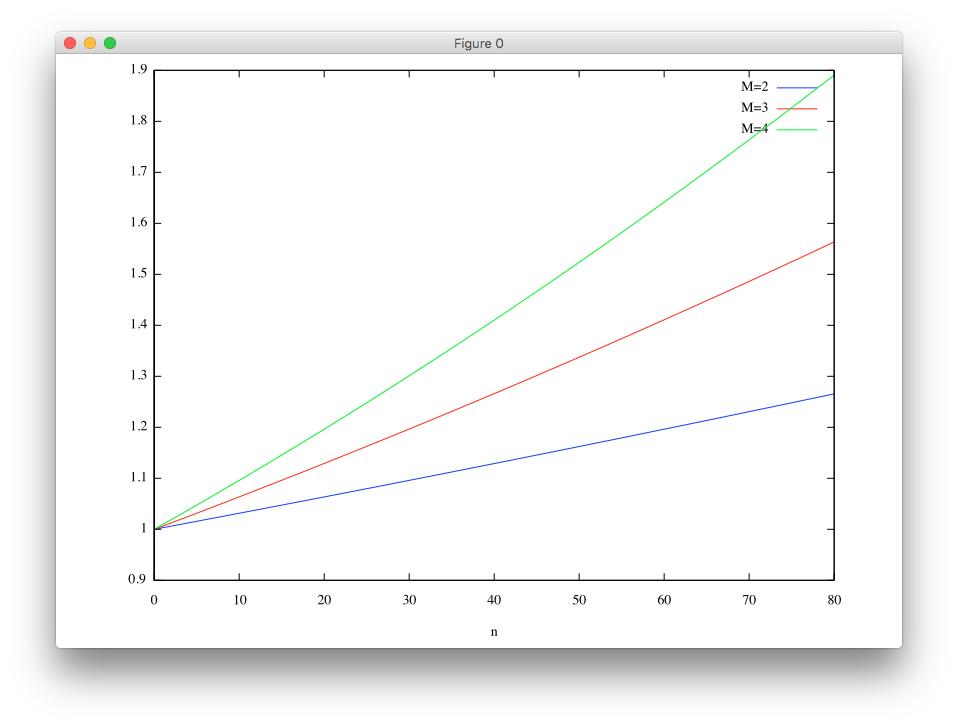


Figure 3: Data rate as function of overlap size (n) for partition sizes 2x2, 3x3 and 4x4.

The theoretical speedup for a MxM partition is

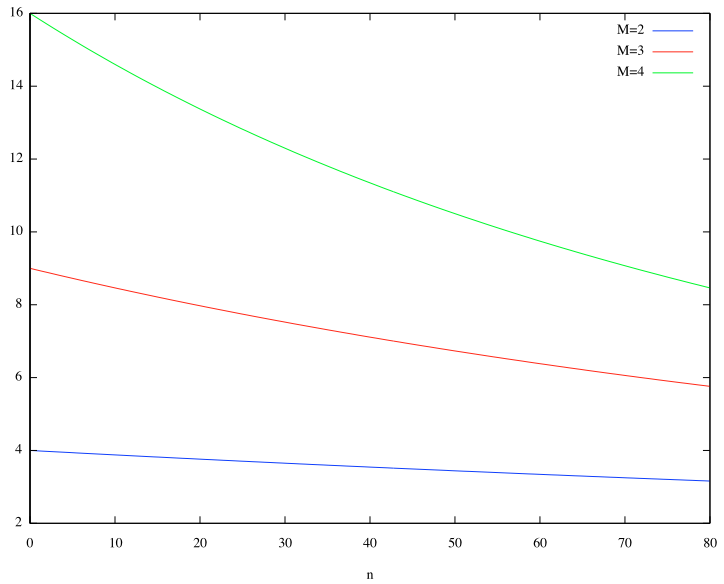


Figure 4: Theoreticsl speedup by MxM partitioning including overlap.

From Figure 4 it would seem that a speed up of around 11.3 and 13.4 for a 4x4 partitioning is achievable utilizing 16 cores, assuming n is between 20 and 40. However, the more partitions we have the more load we put on the subsystem responsible for duplicating and distributing the data.

## Readout data distribution

Due to the overlaps each readout value, for a 2x2 partition, may need to be sent to up to four segments. So the system must support some kind of data duplication and distribution.

This means that somewhere in the system there must be a mapping between a ‘hit’ and one or more ‘destinations’.

[destinations] = f(panel, fec, asic, channel)

for destid in destinations:

ip = g(destid)

port = h(destid)

This is ideally done in the readout system as the decision has to be made individually for each ‘hit’. If instead hits are bundled into a packet and then transmitted there is great risk of including unwanted data, thereby increasing the data rate.

## Events in the overlap region

We should evaluate the impact of not doing the overlap data duplication and distribution as it is not obvious whether this will actually be a problem, or at least not known quantitatively how big the problem is.

Firstly, let us consider hits near the edges assuming we’re interested in calculating events for region B. We assume that *n* is the maximum number of strips involved in a readout belonging to an event. (see later description of idealized tracks). The following description refers to Figure 5. On the border between two partitions there is exactly 50% chance of the whole track is within B. As we move the neutron entry point upwards into B probability increases until it becomes 100% at distance *n*. The fraction of discarded tracks as function of distance from the border is

where *h* is an integer strip count. The probability that the neutron position is calculated correctly in this region (full track is available) is 69%.



Figure 5: Losses in the overlap regions. Dots denote point of entry for the neutron. Lines are idealised tracks. Left: neutron entry on the border between regions. Center: neutron entry in the overlap region. Right: neutron entry outside the overlap region.

Further events may be correctly reconstructed, even if it does not have the full track length, depending upon the shape and quality of the track fragment. On the other hand it may also be the case that the missing fragment would have had additional information leading to invalidation of the track as an event.

In Figure 6 we consider tracks overlapping the two regions. In one case (left column) we do not use an overlap region, in the other (right column) we include one.

In both cases we essentially create new, shorter tracks, by ‘clamping’ to the border region.



Figure 6: Effect of the overlap region on event processing. Top row: identical tracks near the border between B and D. Bottom left: tracks when ignoring the border. Bottom right: tracks after including the overlap.

The effect is that without overlap region some events belonging to region D are falsly identified as occuring on the border, whereas some event belonging to region B may be discarded because their tracks are truncated. This side effect is diminished (eliminated?) when using data in the overlap region.

## Ideas for validation

While it is certainly doable to partition the detector it is necesarry to validate that it produces the correct results and that it does not produce artificial events which are a by-product of the partition in combination with the algorithms and does not signify any physics. One approach is to create an idealised reference dataset.

This dataset is then subjected to the NMX clustering and event processing algorithms. The calculated and simulated event positions should agree. Then the same processing is done on the subset of the dataset that belongs to, say, region B and its overlap regions. Finally the events in region B should match the reference set.



Figure 7: Generating random straight particle tracks. Circles mark the entry point.

# Further speedup

It might be possible to further speed up the processing by distributing x- and y- events to different cores on the same CPU. The calculation of x- and y- entry points can be done independently. However this requires a later step to combine and correlate these results.