Dedispersion for LOFAR using GPUs

Wes Armour
Mike Giles
Oxford e-Research Centre

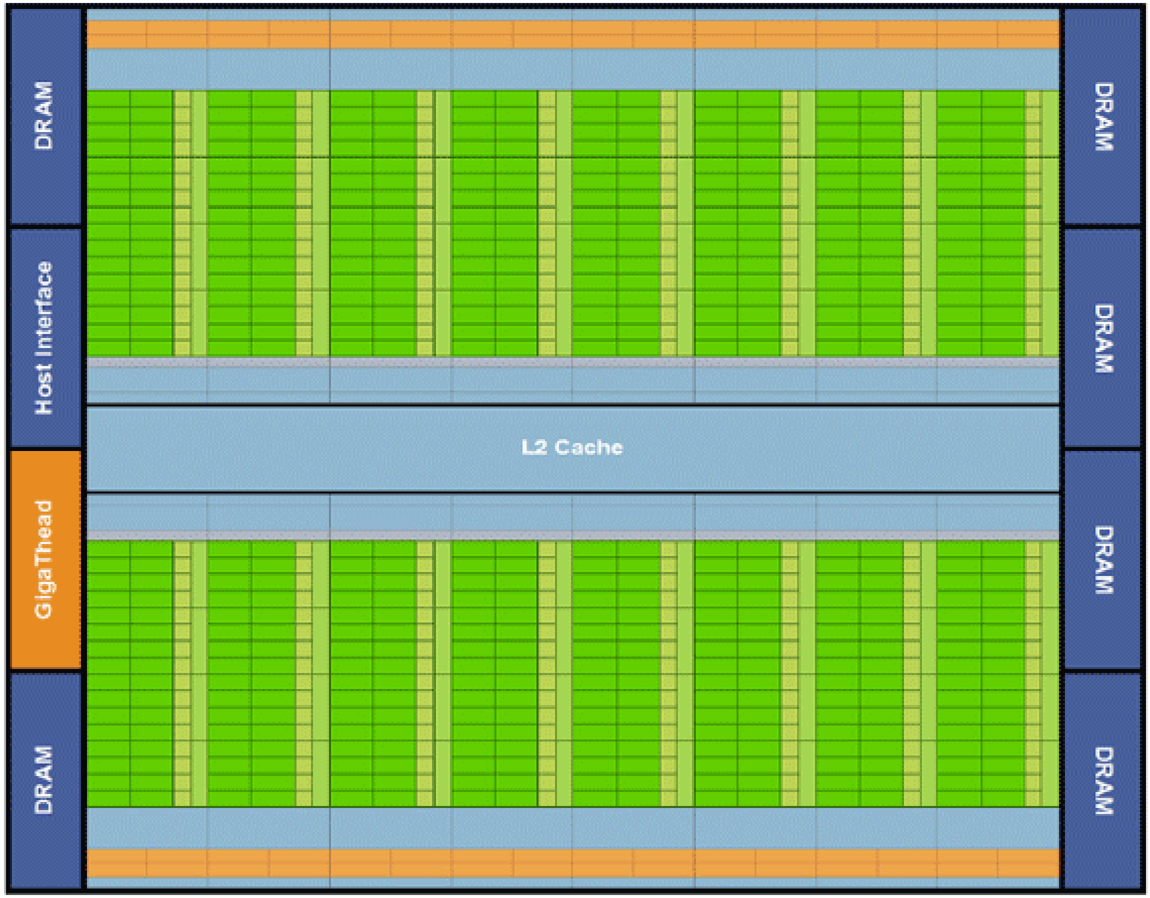
26th October 2011

Why use GPUs ?

Overview of current GPUs - Fermi

- Fermi released 2010
- 32/48 cores (stream processors or SP) per Streaming Multiprocessor (SM) (sm_20/sm_21)
- Configurable 16/48K or 48/16K L1 cache / shared memory (per SM)
- New L2 cache – 768K
- GigaThread – Concurrent kernel execution (16)
- ECC – slows things down, 10% - 25%
- Simple to use extensions - CUDA for C, C++ & FORTRAN

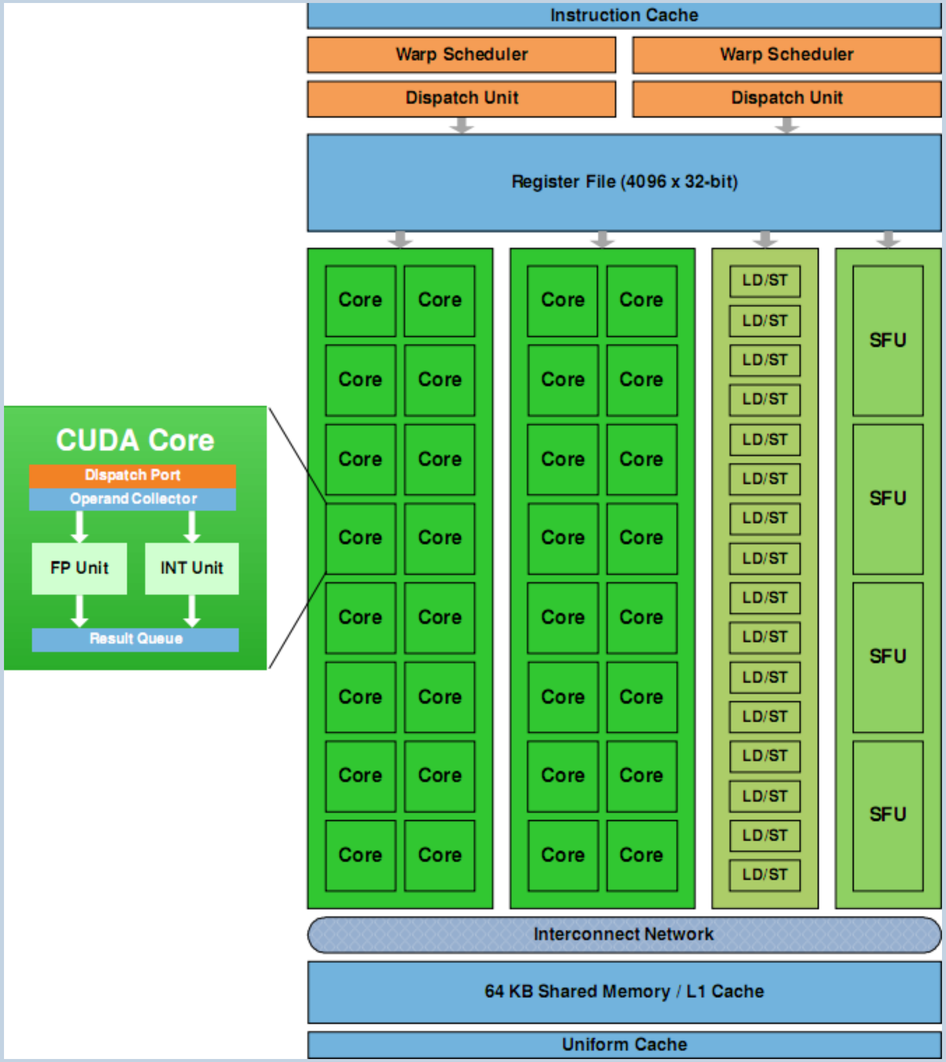
Fermi design



Fermi's 16 SM are positioned around a common L2 cache. Each SM is a vertical rectangular strip that contain an orange portion (scheduler and dispatch), a green portion (execution units), and light blue portions (register file and L1 cache).

OXFORD e-Research CENTRE

Fermi Streaming Multiprocessor (SM) design



OXFORD e-Research CENTRE

Parallel granularity and data sharing.

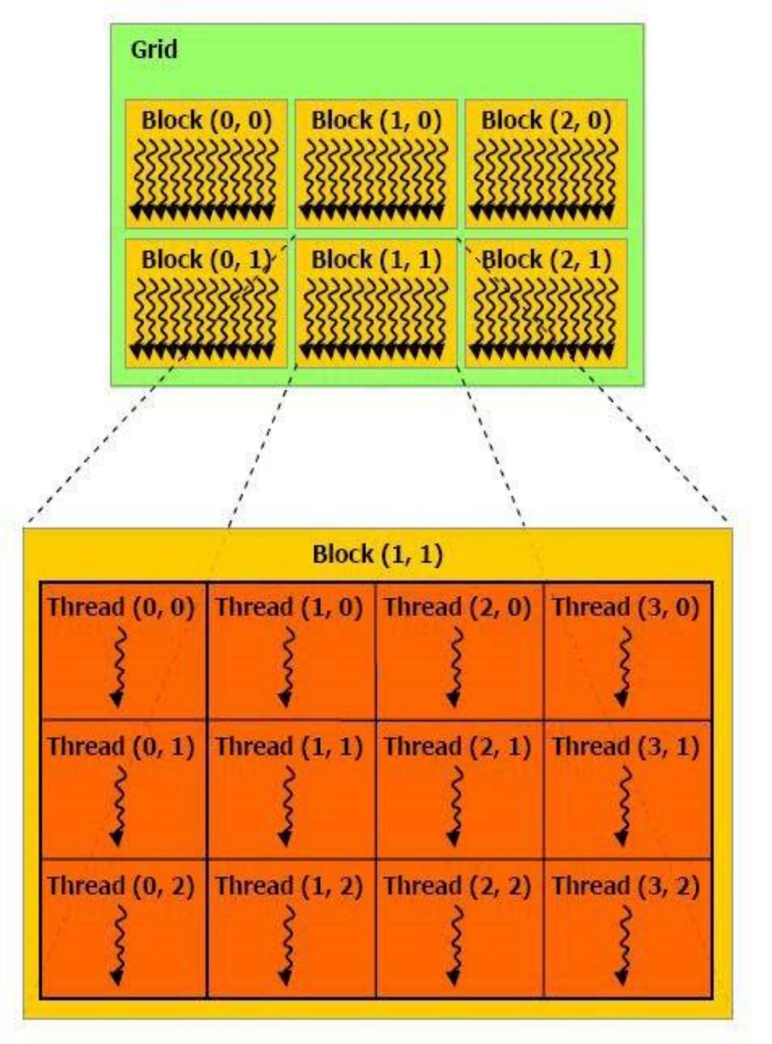
- Each cuda core (SP) executes a sequential thread, in SIMT (Single Instruction, Multiple Thread) fashion - all cores in the same group execute the same instruction at the same time (like SIMD).

- Threads are executed in groups of 32 – a *warp*.

- To hide high memory latency, warps are executed in a time-multiplexed fashion - When one warp stalls on a memory operation, the multiprocessor selects another ready warp and switches to that one.

OXFORD e-Research CENTRE

Parallel granularity and data sharing.



- Kernel launches a grid of (3,2) thread blocks…

Kernel<<< (3,2),(4,3) >>>(params)

- Each thread block consists of (4,3) unique threads.

Parallel granularity and data sharing.

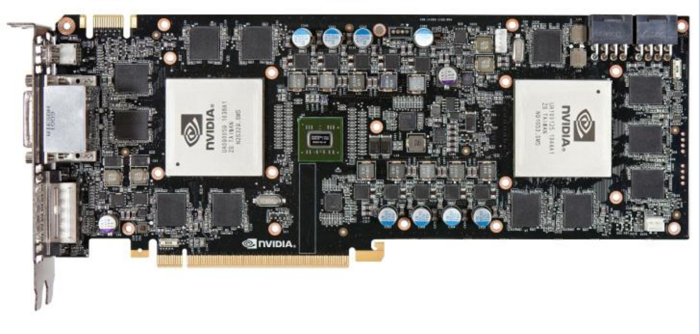
- Single thread has its own (cannot be shared by other threads) per-thread very fast local memory (registers).

- A block of threads has its own per-block shared memory allowing for communication and data sharing between threads in a thread block.

- A grid of thread blocks can communicate via global memory.

Latest Fermi based cards

- GeForce GTX 590 – 2x GF110 GPUs
- 6.8 GFLOPS / watt
- Price point ~ £600



- M2090 – 1x GF110 GPU, 16 SMs, 512 cores.
- 6.8 GFLOPS / watt
- Price point ~ £2.5K

Influence and Take-up

TOP 500 – 3 out of top 5 utilise Fermi/Tesla

- Tianhe-1A 2.5 petaflops
- Based on 14336 Xeon and 7168 M2050
- To achieve same performance using only CPUs 50000 CPUs, 2x floor space and 3x power (Estimates made by NVIDIA)
- Uses Lustre :-s

OXFORD e-Research CENTRE

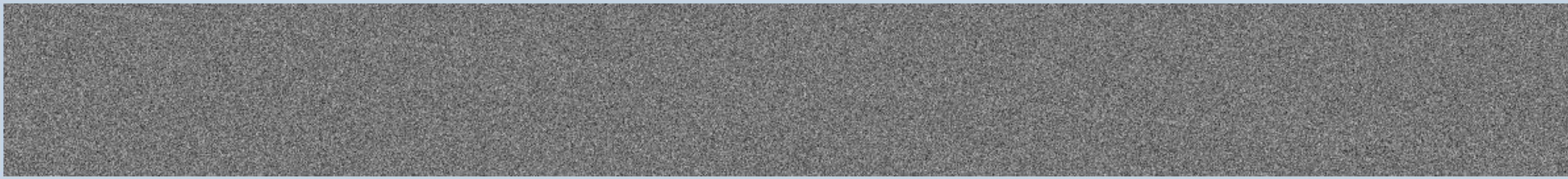
Aims of the project…

- Process a 3.2 Gb/s data stream on a single GPU.
- Allowing for a vast reduction in the cost (capital/maintenance) of compute.
- Identify appropriate areas of parallelism on both CPU and GPU.

Dedispersion

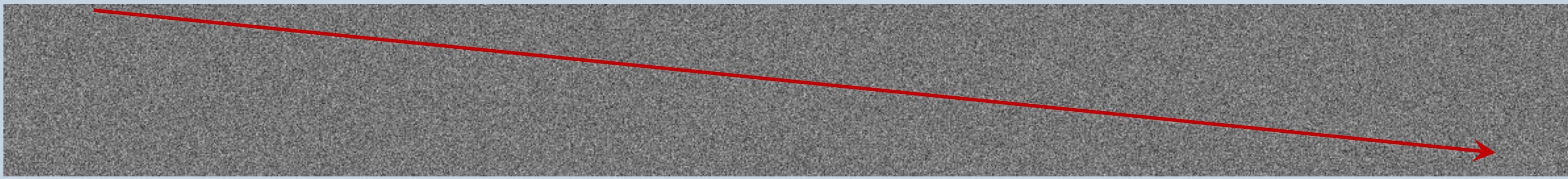
Experimental data

Most of the measured signals live in the noise of the apparatus.

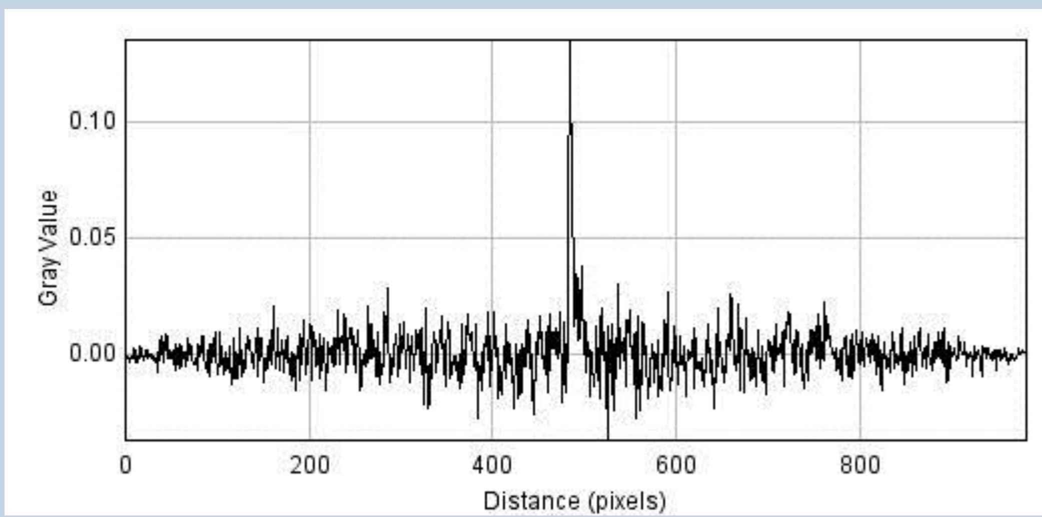


Experimental data

Most of the measured signals live in the noise of the apparatus.



Hence frequency channels have to be “folded”



OXFORD e-Research CENTRE

MDSM – Alessio, Brute force algorithm.

Algorithm 3.1 Brute-Force Dedispersion CUDA implementation

Require: nsamp, nchans, startdm, dmstep, tsamp, dm_shifts

$s \leftarrow threadIdx.x + blockIdx.x * blockDim.x$

$shift\_temp = {}^{startdm+blockIdx.y*dmstep}/_{tsamp}$

for $samp = 0$ **to** $nsamp$ **do**

  $samp \leftarrow samp + blockDim.x * gridDim.x$

  $soffset \leftarrow s + samp$

  for $c = 0$ **to** $nchans$ **do**

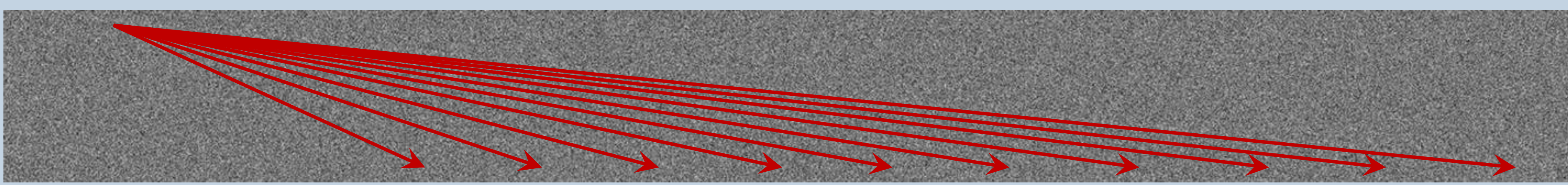
    $index \leftarrow (soffset + dm\_shifts[c] * shift\_temp) * nchans * c$

    $output[blockIdx.y * nsamp + soffset] += input[index]$

  end for

end for

Every DM is calculated to see if a signal is present.



1st Stage – Memory access optimisation

- Began by profiling the code using the **Cuda Profiler**.
- Identified bottle necks in memory access patterns.
- By moving the variable that holds the running total from shared memory to a register kernel execution was **2.6x** faster.

```
__device__ float localvalue[4096];
__global__ void opt_dedisperse_loop(…) {
```

→

```
__global__ void opt_dedisperse_loop(…) {
    float localvalue;
```

7: Performed Brute-Force Dedispersion 1: 1743.604004
7: Performed Brute-Force Dedispersion 0: 1747.496582

→

7: Performed Brute-Force Dedispersion 0: 675.611206
7: Performed Brute-Force Dedispersion 1: 677.087769

2.6x

Produces a **5x** speed increase in the less accurate “sub-band” method.

OXFORD e-Research CENTRE

2nd Stage – Occupancy optimisation

- Optimize GPU block size (number of threads) for each kernel to get the maximum occupancy of threads on the GPU.
- Investigated the optimal block size for GPU hardware versions sm_13 (Tesla) and sm_20 (Fermi).
- Found a block size of 192 on sm_13 and 256 on sm_20 produced best results.
- This produced a **1.25x** speed increase.

Produces a **1.3x** speed increase in the less accurate “sub-band” method.

OXFORD e-Research CENTRE

Initial results after optimisation…

Timings for 1.8 seconds of telescope data, 496 channels, 800 dm's, max dm 79.9

Original Kernel → **Optimised Kernel**

1156 ms → **375 ms**

371 ms reduces to 67 ms in the less accurate "sub-band" method.

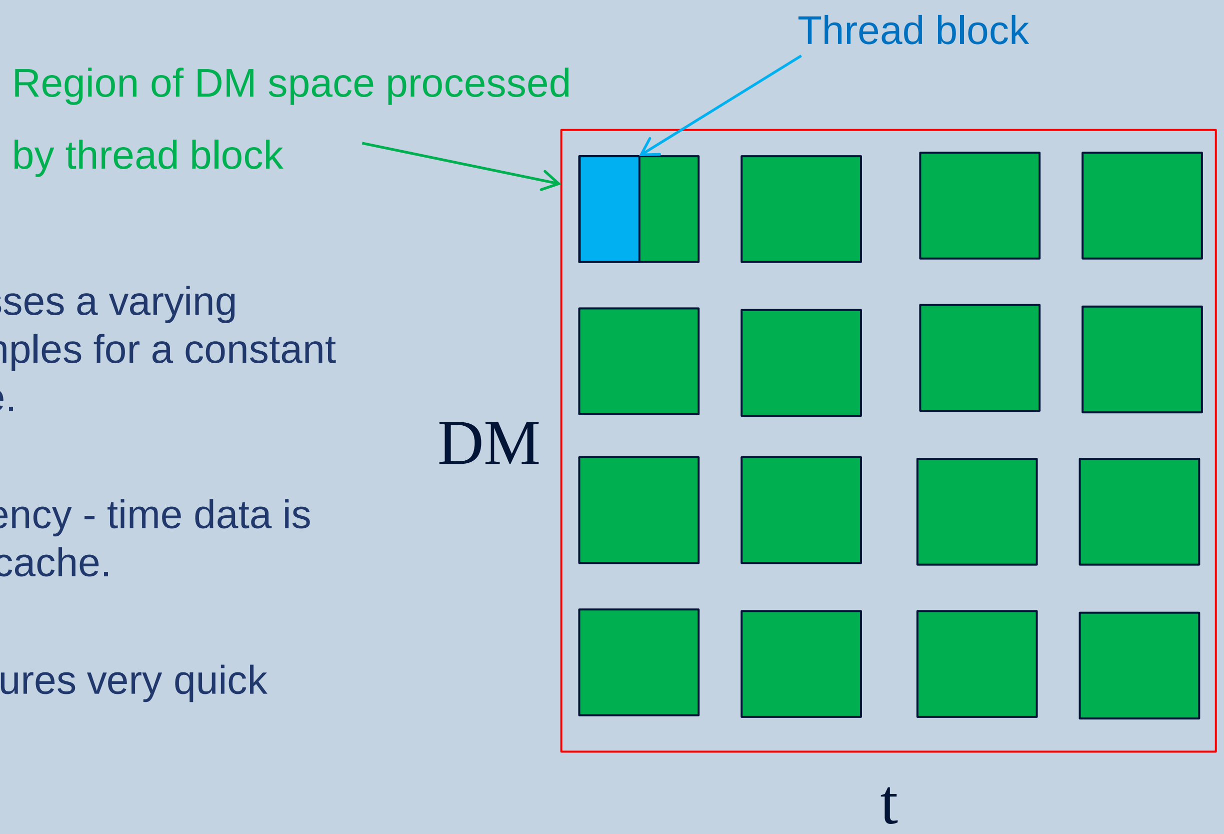
A new brute force algorithm

Three key features…

- Each thread processes a variable number of dispersion measures in local registers.
- Exploit the L1 cache present on the Fermi architecture.
- Optimise the region of dispersion space being processed (thread blocksize).

Processing several DM's per thread

New Algorithm works in the DM - t space rather than frequency – time space.

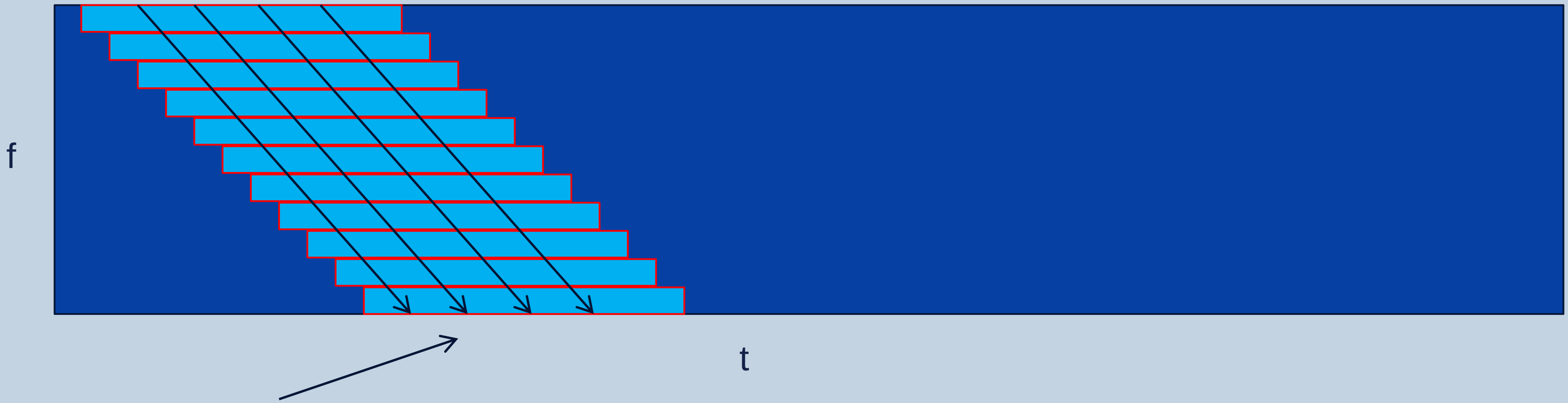


- Each thread processes a varying number of time samples for a constant dispersion measure.
- This ensures frequency - time data is loaded into fast L1 cache.
- Using registers ensures very quick memory access.

OXFORD e-Research CENTRE

Exploiting the L1 cache…

Each dispersion measure for a given frequency channel needs a shifted time value.



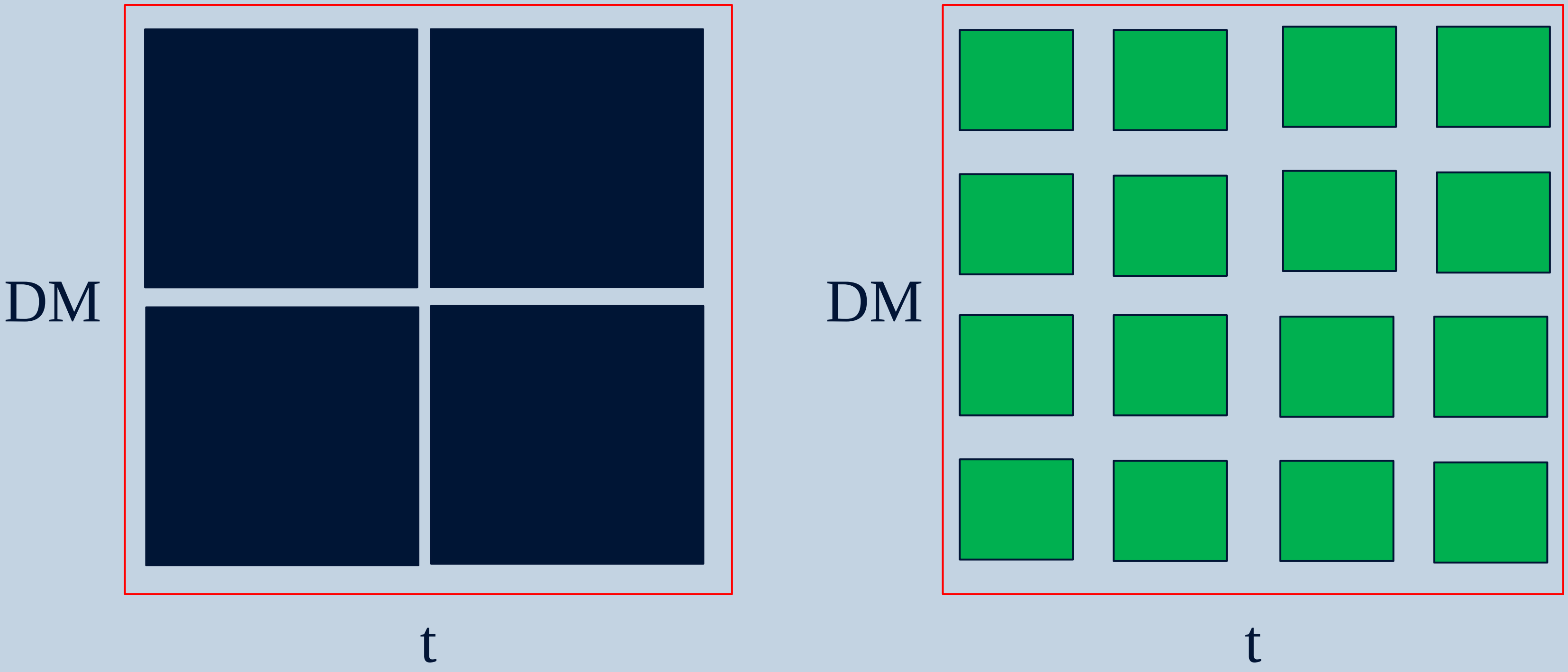
Constant DM's with varying time.

Incrementing all of the registers at every frequency step ensures a high data reuse of the stored frequency time data in the L1 cache.

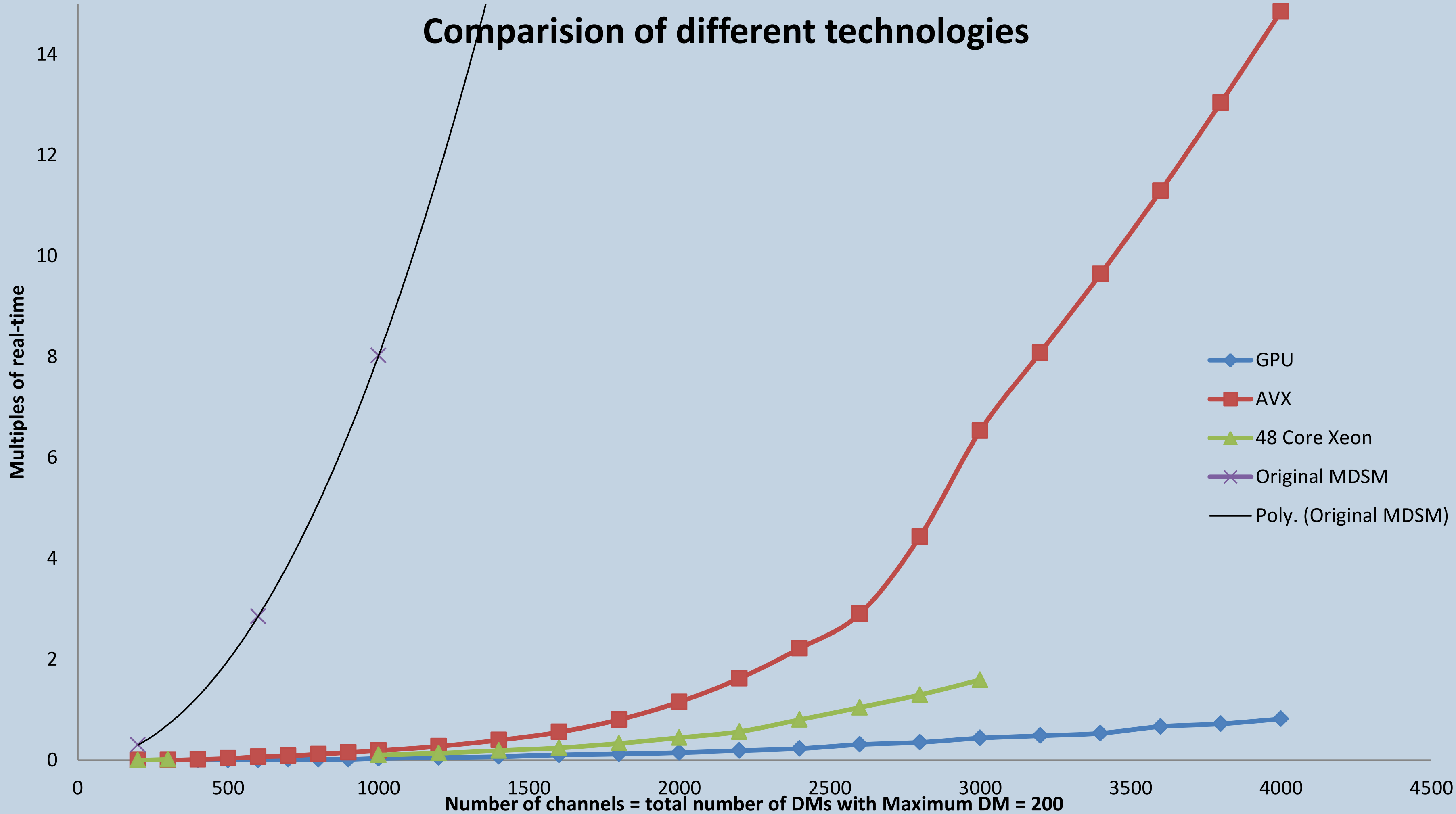
OXFORD e-Research CENTRE

Optimising the parameterisation.

The GPU block size of the new algorithm can take on any size that is integer multiples of the size of a “data chunk”…

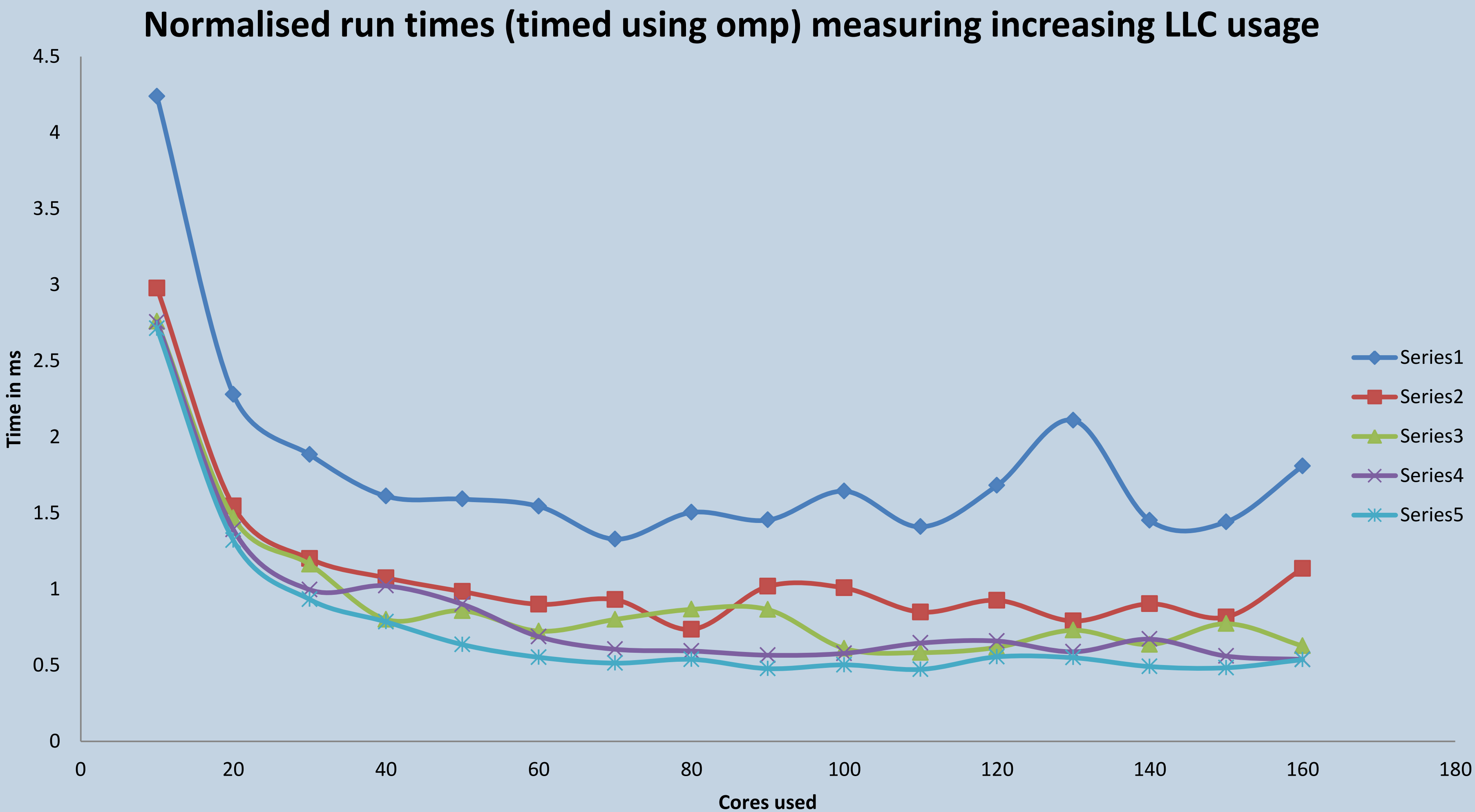


Results…



OXFORD e-Research CENTRE

Results…



OXFORD e-Research CENTRE

Conclusions (So far…)

- GPU wins hands-down. **At the moment!**

- AVX puts up a good fight.

- Watch out for Intels MIC (Many Integrated Core) chip – 32 in-order cores, 4 threads per core 512 bit SIMD units running a 1024 bit ring bus.

OXFORD e-Research CENTRE

On-going and future work

- Shared memory algorithm ensures more predictable data reuse and is about 15% quicker for some maximum DM and number of channels combinations – results to come.

- Man made radio frequency interference can lead to false positives. Have tried to incorporate an RFI Clipper into the dedeispersion code. Significantly slows things down at the moment.

- Working with Ben and Mike on AVX vectorizaton of the Polyphase filter.

- Looking into adding Markov detection processes to eliminate false positives.

Acknowledgments and Collaborators

University of Malta

Alessio Magro – MDSM

University of Oxford

Mike Giles (Maths) – Cuda, GPU algorithms.

Aris Karastergiou (Physics) – ARTEMIS, Astrophysics, Experimental Work.

Kimon Zagkouris (Physics) – Astrophysics, Experimental Work.

Chris Williams (OeRC) – RFI Clipper, Data pipeline.

Ben Mort (OeRC) – Data Pipeline, pelican.

Fred Dulwich (OeRC) – Data Pipeline, pelican.

Stef Salvini (OeRC) – Data Pipeline, pelican.

Chris Williams (OeRC) – RFI Clipper, Data pipeline.

Steve Roberts (Engineering) – Signal processing/detection algorithms.