GPUs and MeerKAT

Bruce Merry

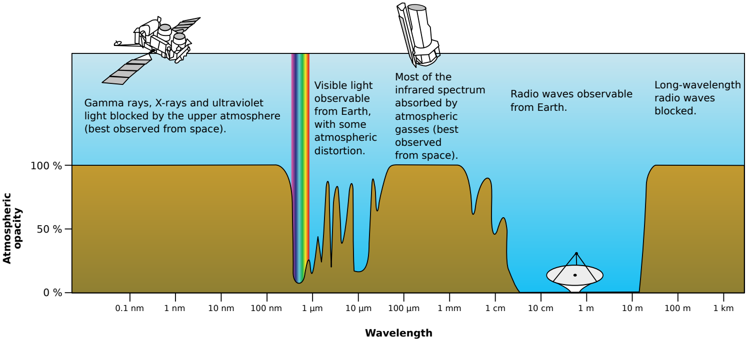
SKA South Africa

2015-04-24

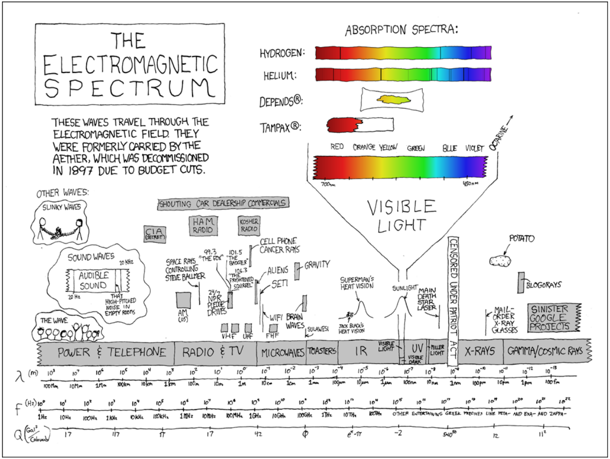
Outline

Outline

Why Radio?

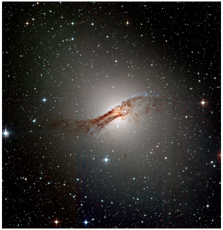

Gamma rays, X-rays and ultraviolet light blocked by the upper atmosphere (best observed from space).
Visible light observable from Earth, with some atmospheric distortion.
Most of the infrared spectrum absorbed by atmospheric gasses (best observed from space).
Radio waves observable from Earth.
Long-wavelength radio waves blocked.
Atmospheric opacity
100 %
50 %
0 %
0.1 nm
1 nm
10 nm
100 nm
1 µm
10 µm
100 µm
1 mm
1 cm
10 cm
1 m
10 m
100 m
1 km
Wavelength

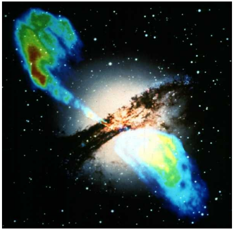
Another View



http://xkcd.com/273

The Full Picture

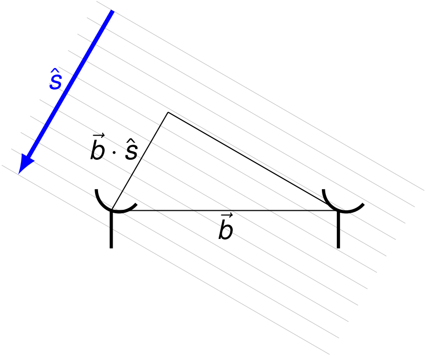


Shhhh...

$$1\,\text{Jy} = 1 \times 10^{-26}\,\text{W}\,\text{m}^{-2}\,\text{Hz}^{-1}$$

Sun	10^5 Jy (at 5 GHz)
Cellphone	10^8 Jy (1km away)
Source	1–10 µJy

Interferometry



$$V(u, v, w) = \int_{l,m} A(l, m, w) I(l, m) e^{-2\pi i [ul + vm]}\, dl\, dm$$

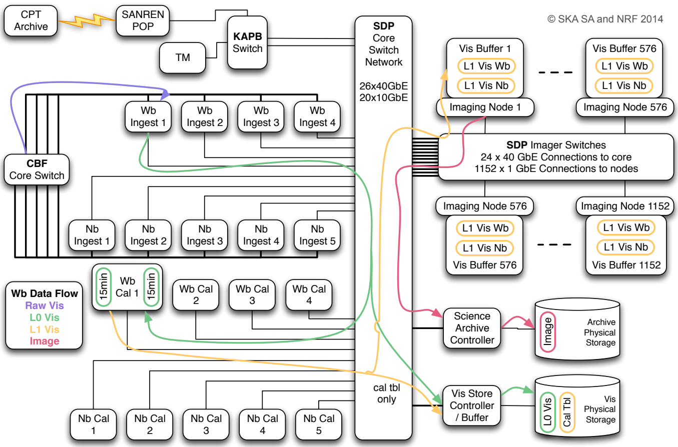
Outline

MeerKAT

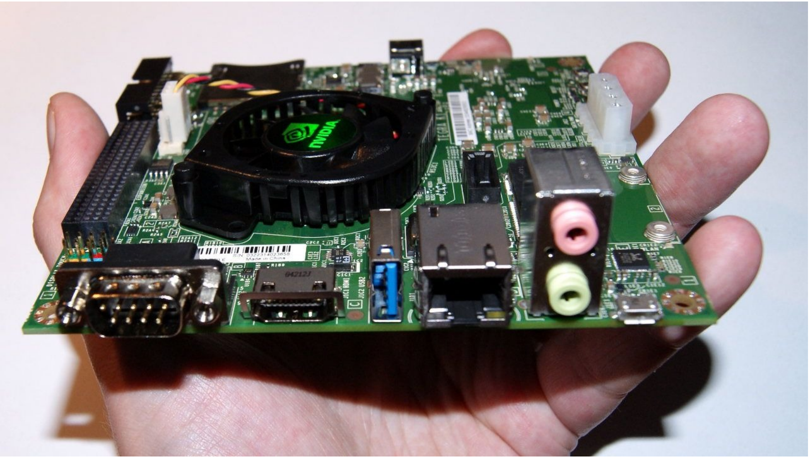
© SKA South Africa 2014

MeerKAT

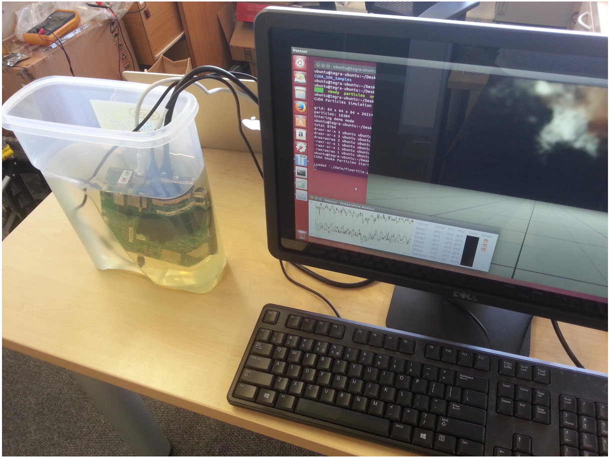
Science Data Processor



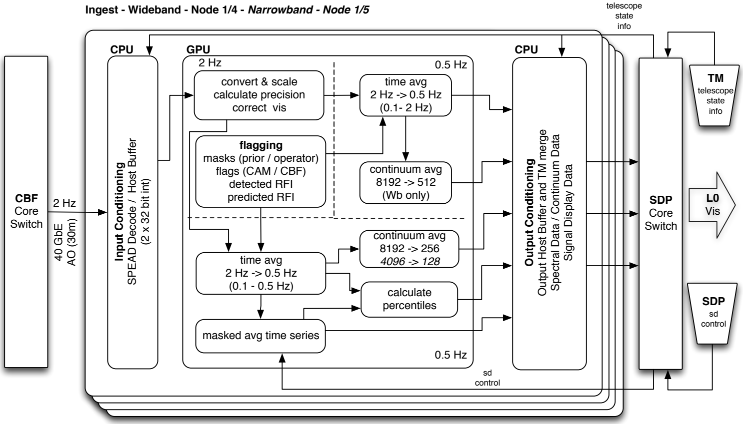
Prototype Imaging Node



This is Cool



Ingest

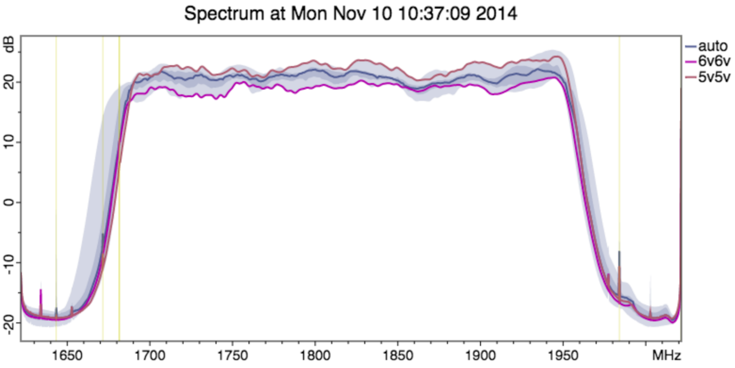

Ingest - Wideband - Node 1/4 - *Narrowband - Node 1/5*
telescope state info
CBF Core Switch
2 Hz
40 GbE AO (30m)
CPU
Input Conditioning
SPEAD Decode / Host Buffer
(2 x 32 bit int)
GPU
2 Hz
0.5 Hz
convert & scale
calculate precision
correct vis
flagging
masks (prior / operator)
flags (CAM / CBF)
detected RFI
predicted RFI
time avg
2 Hz -> 0.5 Hz
(0.1- 2 Hz)
continuum avg
8192 -> 512
(Wb only)
continuum avg
8192 -> 256
4096 -> 128
time avg
2 Hz -> 0.5 Hz
(0.1 - 0.5 Hz)
calculate percentiles
masked avg time series
0.5 Hz
sd control
CPU
Output Conditioning
Output Host Buffer and TM merge
Spectral Data / Continuum Data
Signal Display Data
SDP Core Switch
TM
telescope state info
L0 Vis
SDP
sd control

Ingest

Ingest is responsible for

- Converting from integer to float
- Correcting some instrumental effects
- Flagging RFI
- Reducing data in time
- Reducing data in frequency
- Computing signal statistics

Real-time Signal Displays



Lots of Data

2	frames per second
2080	baselines (64 antennas)
53248	frequency channels ($32k + 5 \times 4k$)
4	polarisations
8	bytes per visibility ($2 \times$ int32)

Lots of Data

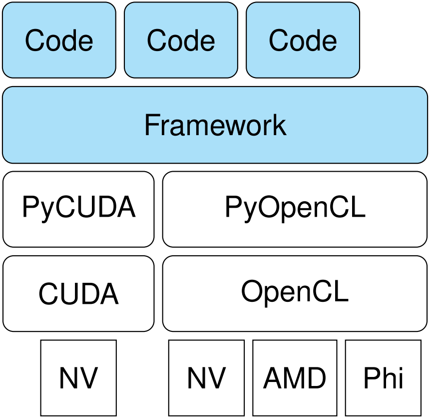
2	frames per second
2080	baselines (64 antennas)
53248	frequency channels ($32k + 5 \times 4k$)
4	polarisations
8	bytes per visibility ($2 \times$ int32)
6.6	GiB per second input

Lots of Data

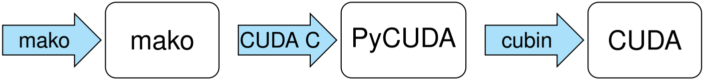
2	frames per second
2080	baselines (64 antennas)
53248	frequency channels ($32k + 5 \times 4k$)
4	polarisations
8	bytes per visibility ($2 \times$ int32)
6.6	GiB per second input
1.0	GiB per second per server

Another Layer of Abstraction

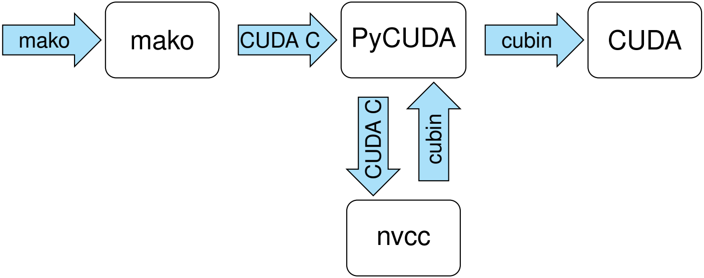
"All problems in computer science can be solved by another level of indirection" — *David Wheeler*



Kernel Compilation



Kernel Compilation



Example Template


```
KERNEL REQD_WORK_GROUP_SIZE(${block}, ${block}, 1)
void accum(
% for i in range(outputs):
    GLOBAL float2 * RESTRICT out_vis${i},
    GLOBAL float * RESTRICT out_weights${i},
    GLOBAL unsigned char * RESTRICT out_flags${i},
% endfor
    const GLOBAL float2 * RESTRICT in_vis,
    const GLOBAL float * RESTRICT in_weights,
    const GLOBAL unsigned char * RESTRICT in_flags,
    int out_stride,
    int in_full_stride,
    int in_kept_stride,
    int channel_start)
{
    ...
}
```

Example Template

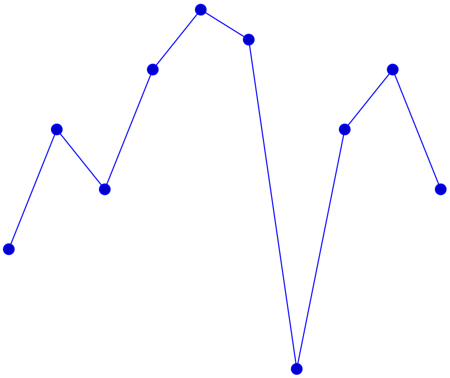
```
 __global__ __launch_bounds__((16)*(16)*(1))
void accum(
    float2 * __restrict out_vis0,
    float * __restrict out_weights0,
    unsigned char * __restrict out_flags0,
    const float2 * __restrict in_vis,
    const float * __restrict in_weights,
    const unsigned char * __restrict in_flags,
    int out_stride,
    int in_full_stride,
    int in_kept_stride,
    int channel_start)
{
    ...
}
```

Example Template

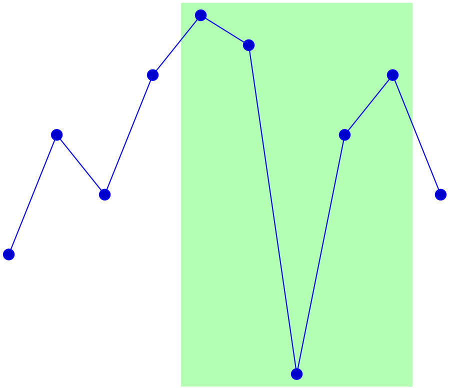

```
__kernel __attribute__((reqd_work_group_size(16,16,1)))
void accum(
    __global float2 * restrict out_vis0,
    __global float * restrict out_weights0,
    __global unsigned char * restrict out_flags0,
    const __global float2 * restrict in_vis,
    const __global float * restrict in_weights,
    const __global unsigned char * restrict in_flags,
    int out_stride,
    int in_full_stride,
    int in_kept_stride,
    int channel_start)
{
    ...
}
```

Outline

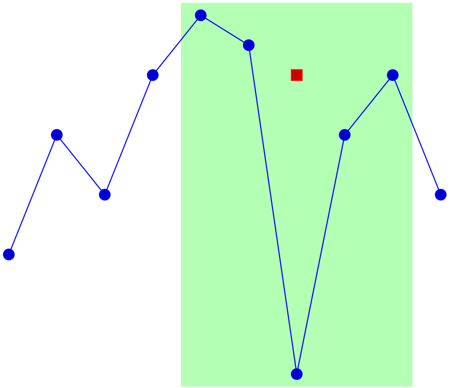
Definition



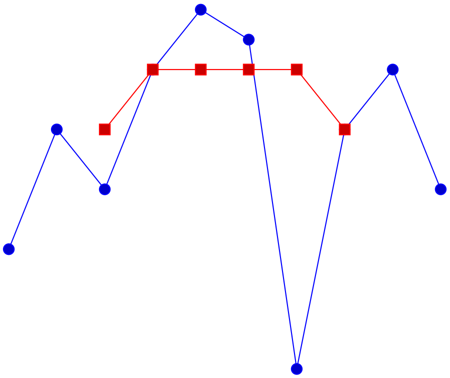
Definition



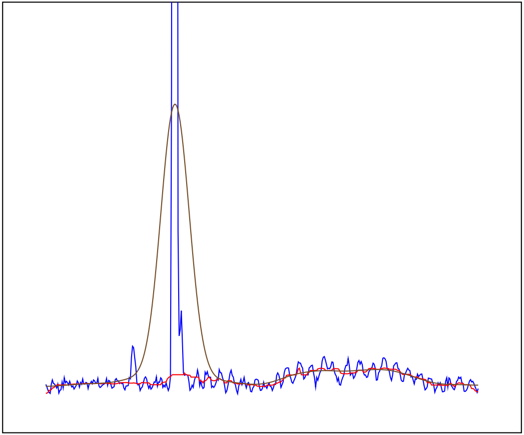
Definition



Definition



Example



Sliding Window Algorithm

10	0
17	2
12	1
22	3
34	4
14	

Sliding Window Algorithm

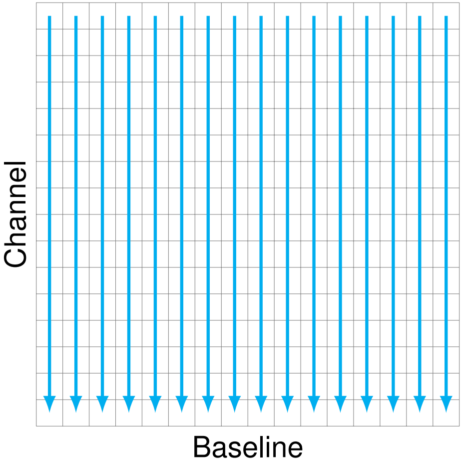
10	
17	1
12	0
22	2
34	3
14	

Sliding Window Algorithm

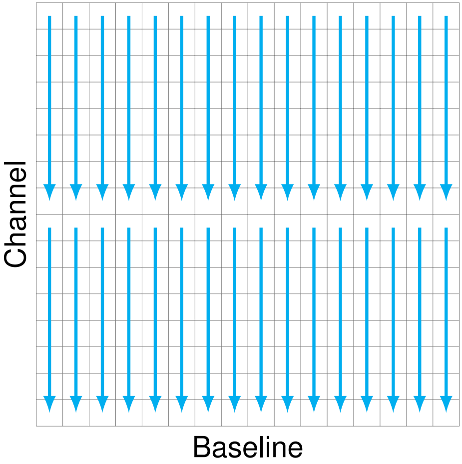
10	
17	2
12	0
22	3
34	4
14	1

Outline

Parallelisation



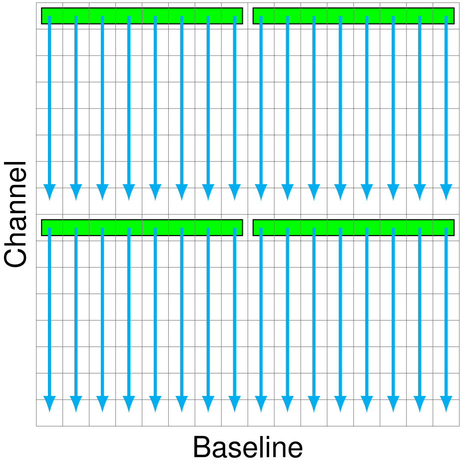
More Parallelism



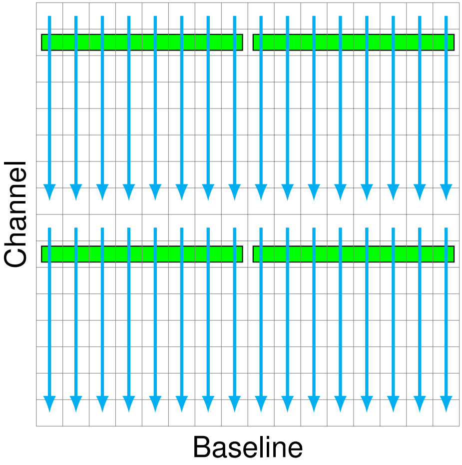
Sample code

```
__device__ void median_filter_slide(
    median_filter *self, float new_sample) {
{
  float old_sample = self->samples[0];
  int new_rank = WIDTH - 1;
#pragma unroll
  for (int j = 0; j < WIDTH - 1; j++) {
    self->samples[j] = self->samples[j + 1];
    int old_cmp = (old_sample <= self->samples[j]);
    int cmp = (new_sample < self->samples[j]);
    self->rank[j] = self->rank[j + 1] - old_cmp + cmp;
    new_rank -= cmp;
  }
  self->samples[WIDTH - 1] = new_sample;
  self->rank[WIDTH - 1] = new_rank;
}
```

Memory access pattern

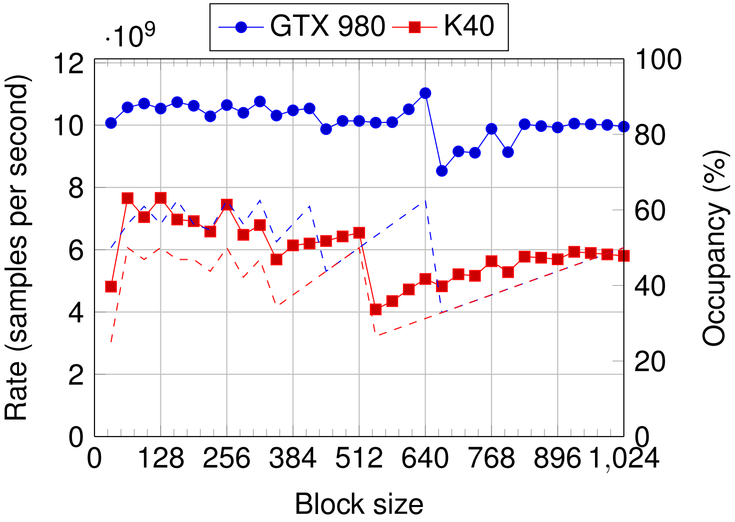


Memory access pattern

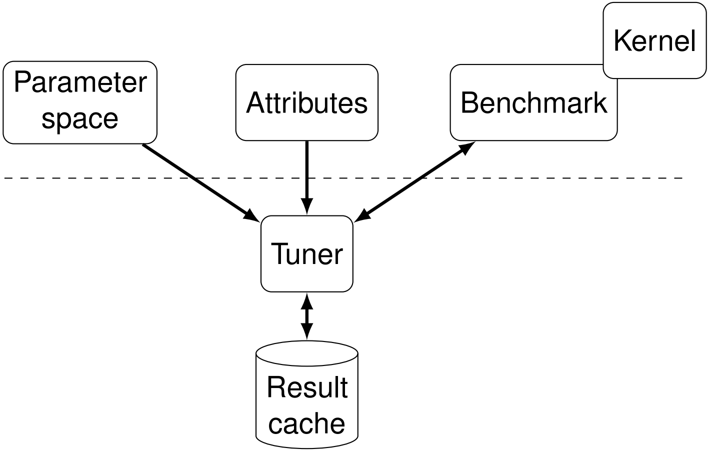


Outline

Block size


GTX 980
K40
$\cdot 10^9$
12
10
8
6
4
2
0
Rate (samples per second)
100
80
60
40
20
0
Occupancy (%)
0
128
256
384
512
640
768
896
1,024
Block size

Autotuning


Kernel
Parameter space
Attributes
Benchmark
Tuner
Result cache

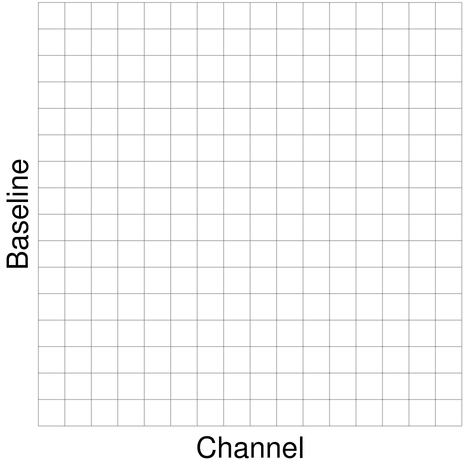
Autotuning for a fill kernel

```
@classmethod
@tune.autotuner(test={'wgs': 128})
def autotune(cls, context, dtype, ctype):
    queue = context.create_tuning_command_queue()
    shape = (1048576,)
    data = accel.DeviceArray(context, shape, dtype)
    def generate(**tuning):
        fn = cls(
            context, dtype, ctype,
            **tuning).instantiate(queue, shape)
        fn.bind(data=data)
        return tune.make_measure(queue, fn)
    return tune.autotune(
        generate, wgs=[64, 128, 256, 512])
```

Outline

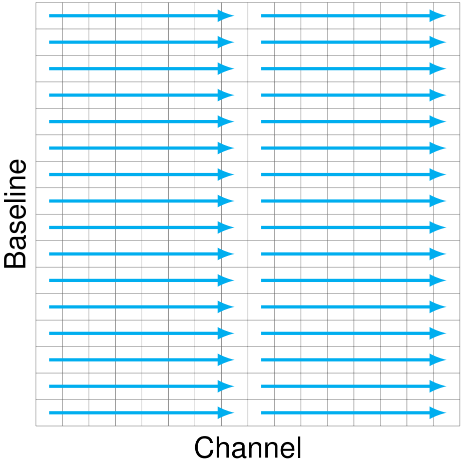
Motivation

What if the incoming data looks like this?



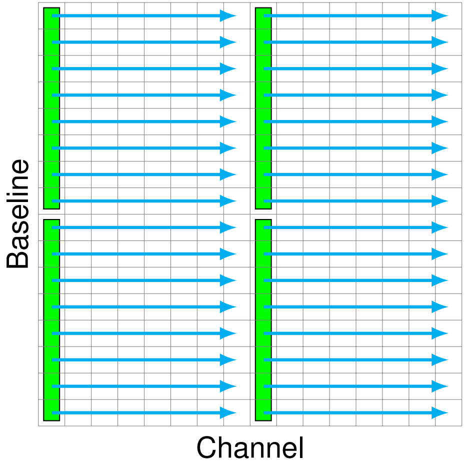
Motivation

What if the incoming data looks like this?

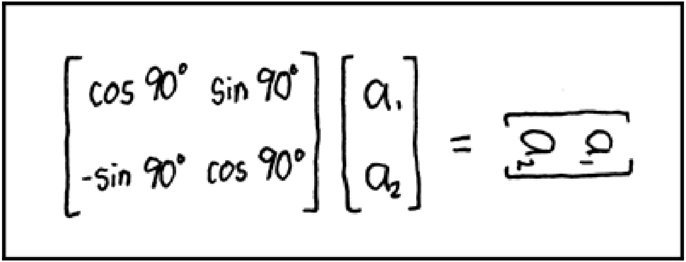


Motivation

What if the incoming data looks like this?



Transpose It



http://xkcd.com/184/

Naive Kernel

```
__global__ void transpose(
    float *out, const float *in, int R, int C)
{
    int x = blockIdx.x * blockDim.x + threadIdx.x;
    int y = blockIdx.y * blockDim.y + threadIdx.y;
    float value = in[y * C + x];
    out[x * R + y] = value;
}
```

Naive Kernel Results

L2 Cache

Reads	8388679	30.937 GB/s	
Writes	67108870	247.496 GB/s	
Total	75497549	278.433 GB/s	Idle Low Medium

Unified Cache

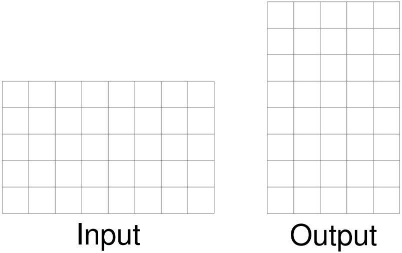
Local Loads	0	0 B/s	
Local Stores	0	0 B/s	
Global Loads	16777216	61.874 GB/s	
Global Stores	67108864	247.496 GB/s	
Texture Reads	8388608	30.937 GB/s	
Unified Total	92274688	340.307 GB/s	Idle Low Medium

Device Memory

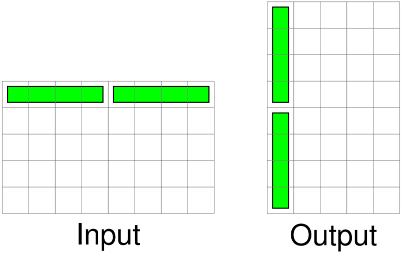
Reads	8388632	30.937 GB/s	
Writes	8390508	30.944 GB/s	
Total	16779140	61.881 GB/s	Idle Low Medium

Only 62 of 224 GB/s (GTX 980)

Naive Kernel Access Pattern

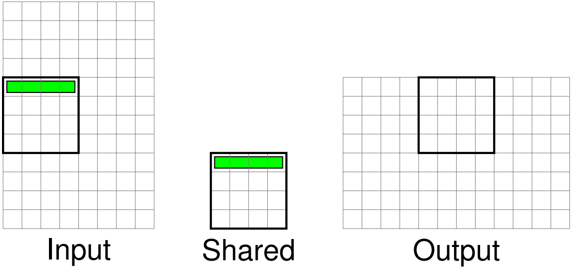


Naive Kernel Access Pattern

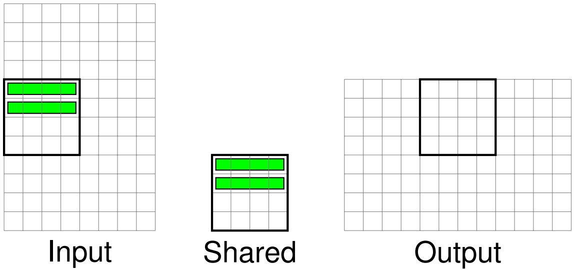


Outline

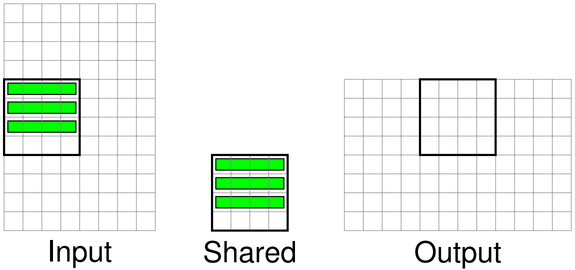
Using Shared Memory



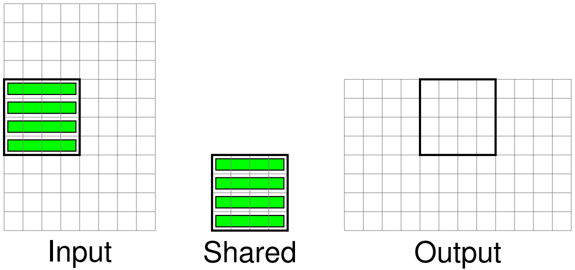
Using Shared Memory



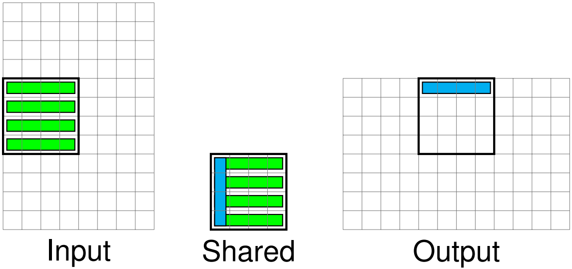
Using Shared Memory



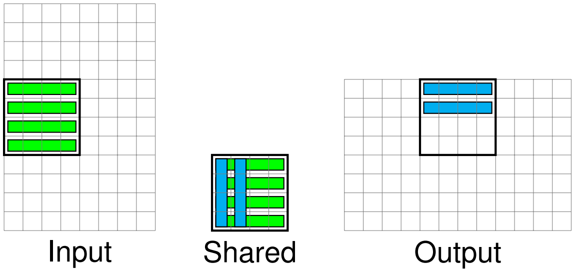
Using Shared Memory



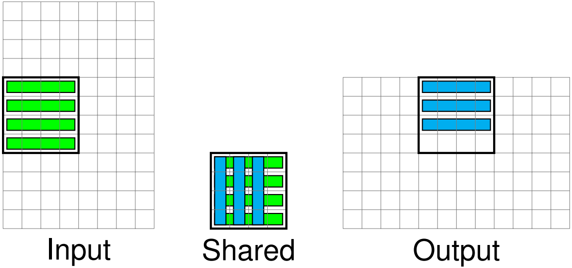
Using Shared Memory



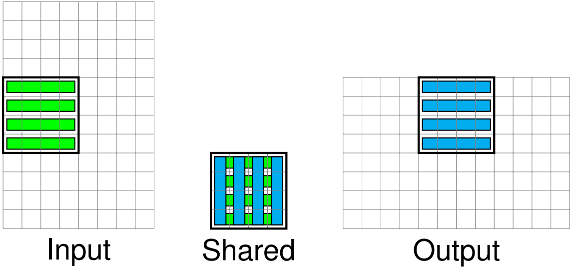
Using Shared Memory



Using Shared Memory



Using Shared Memory



Shared Memory Version 1

```
__global__ void transpose(
    float *out, const float *in, int R, int C)
{
    __shared__ float tile[TILE][TILE];
    int x = blockIdx.x * blockDim.x + threadIdx.x;
    int y = blockIdx.y * blockDim.y + threadIdx.y;
    tile[threadIdx.y][threadIdx.x] = in[y * C + x];
    __syncthreads();
    int x2 = blockIdx.x * blockDim.x + threadIdx.y;
    int y2 = blockIdx.y * blockDim.y + threadIdx.x;
    out[x2 * R + y2] = tile[threadIdx.x][threadIdx.y];
}
```

Shared Memory Version 1


```
__global__ void transpose(
    float *out, const float *in, int R, int C)
{
    __shared__ float tile[TILE][TILE];
    int x = blockIdx.x * blockDim.x + threadIdx.x;
    int y = blockIdx.y * blockDim.y + threadIdx.y;
    tile[threadIdx.y][threadIdx.x] = in[y * C + x];
    __syncthreads();
    int x2 = blockIdx.x * blockDim.x + threadIdx.y;
    int y2 = blockIdx.y * blockDim.y + threadIdx.x;
    out[x2 * R + y2] = tile[threadIdx.x][threadIdx.y];
}
```

102 GB/s (GTX 980)

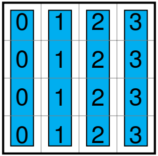
Bank Conflicts

GPUs and MeerKAT
Bruce Merry
Background
Radio Astronomy
MeerKAT
Median Filter
Serial
Parallel
Tuning
Transposition
Introduction
Shared Memory
Summary

0	1	2	3
0	1	2	3
0	1	2	3
0	1	2	3

Shared

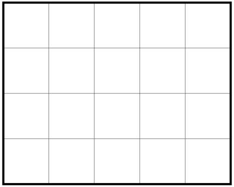
Bank Conflicts



GPUs and MeerKAT

Bruce Merry

Padding



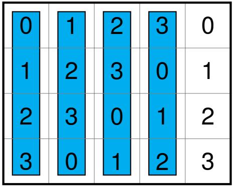
Shared

Padding

0	1	2	3	0
1	2	3	0	1
2	3	0	1	2
3	0	1	2	3

Shared

Padding



Shared

Shared Memory Version 2


```
]
__global__ void transpose(
    float *out, const float *in, int R, int C)
{
    __shared__ float tile[TILE][TILE + 1];
    int x = blockIdx.x * blockDim.x + threadIdx.x;
    int y = blockIdx.y * blockDim.y + threadIdx.y;
    tile[threadIdx.y][threadIdx.x] = in[y * C + x];
    __syncthreads();
    int x2 = blockIdx.x * blockDim.x + threadIdx.y;
    int y2 = blockIdx.y * blockDim.y + threadIdx.x;
    out[x2 * R + y2] = tile[threadIdx.x][threadIdx.y];
}
```

Shared Memory Version 2

```
]
__global__ void transpose(
    float *out, const float *in, int R, int C)
{
    __shared__ float tile[TILE][TILE + 1];
    int x = blockIdx.x * blockDim.x + threadIdx.x;
    int y = blockIdx.y * blockDim.y + threadIdx.y;
    tile[threadIdx.y][threadIdx.x] = in[y * C + x];
    __syncthreads();
    int x2 = blockIdx.x * blockDim.x + threadIdx.y;
    int y2 = blockIdx.y * blockDim.y + threadIdx.x;
    out[x2 * R + y2] = tile[threadIdx.x][threadIdx.y];
}
```

144 GB/s (GTX 980)

Thread Coarsening

Instruction breakdown

1. Load from Global
1. Store to Shared
1. Load from Shared
1. Store to Global
1. Barrier (`__syncthreads`)

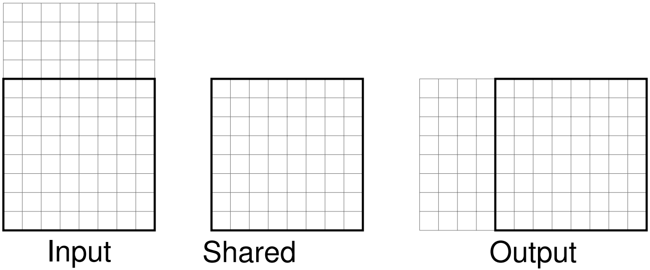
Thread Coarsening

Instruction breakdown

1	Load from Global
1	Store to Shared
1	Load from Shared
1	Store to Global
1	Barrier (`__syncthreads`)
32	Other

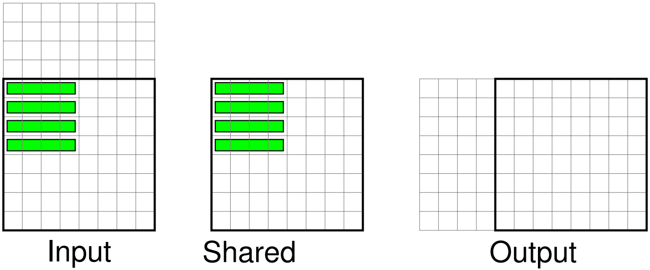
Thread Coarsening

What if we apply *less* parallelism?



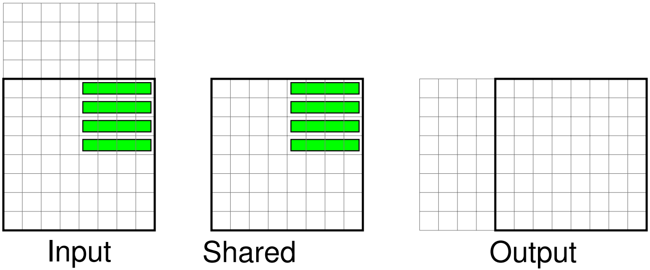
Thread Coarsening

What if we apply *less* parallelism?



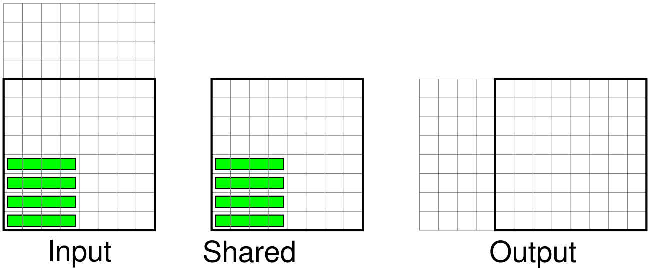
Thread Coarsening

What if we apply *less* parallelism?



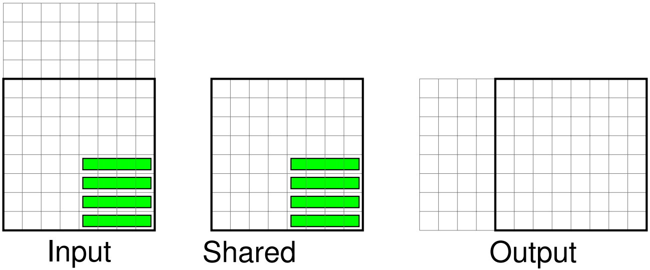
Thread Coarsening

What if we apply *less* parallelism?



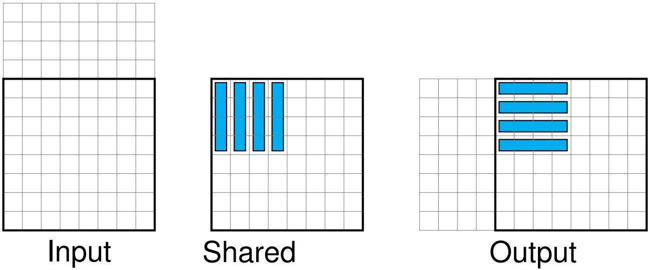
Thread Coarsening

What if we apply *less* parallelism?



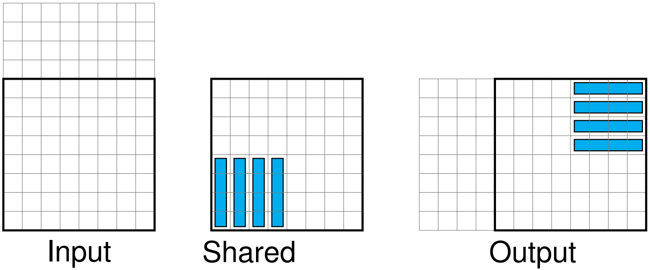
Thread Coarsening

What if we apply *less* parallelism?



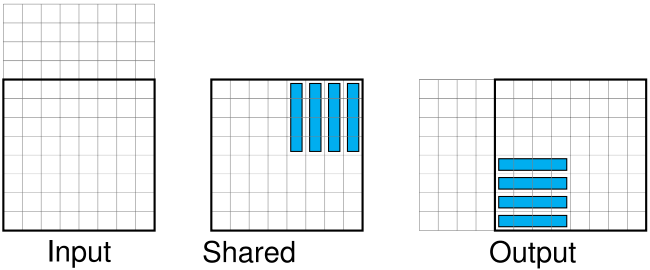
Thread Coarsening

What if we apply *less* parallelism?



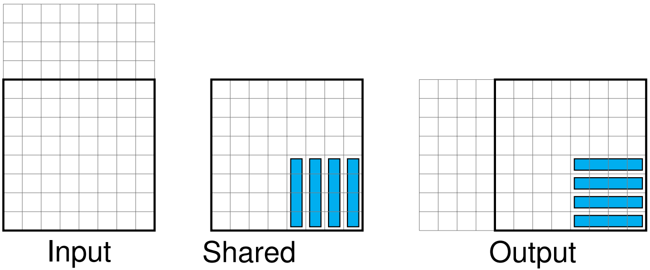
Thread Coarsening

What if we apply *less* parallelism?



Thread Coarsening

What if we apply *less* parallelism?



Thread Coarsening Code


```
__shared__ float tile[SCALE * TILE * (TILE + 1)];
int x0 = blockIdx.x * (SCALEX * TILE);
int y0 = blockIdx.y * (SCALEY * TILE);
int addr0 = (y0+threadIdx.y) * C + x0+threadIdx.x;
int tile0 = threadIdx.y * (TILE + 1) + threadIdx.x;
for (int i = 0; i < SCALE; i++)
  tile[tile0+ofs.in_tile[i]] = in[addr0+ofs.in[i]];
__syncthreads();
addr0 = (x0 + threadIdx.y) * R + (y0 + threadIdx.x);
tile0 = threadIdx.x * (TILE + 1) + threadIdx.y;
for (int i = 0; i < SCALE; i++)
  out[addr0+ofs.out[i]] = tile[tile0+ofs.out_tile[i]];
```

Thread Coarsening

New instruction breakdown

4	Load from Global
4	Store to Shared
4	Load from Shared
4	Store to Global
1	Barrier (`__syncthreads`)
79	Other

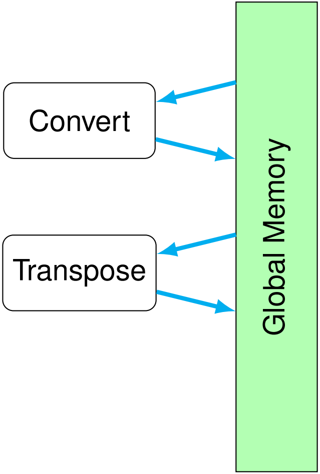
Thread Coarsening

New instruction breakdown

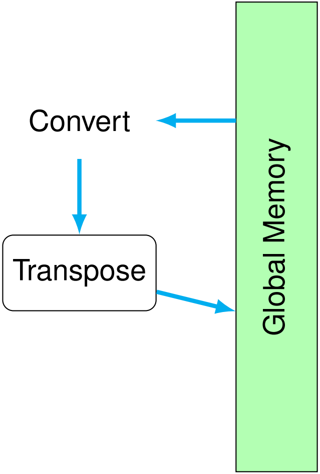
4	Load from Global
4	Store to Shared
4	Load from Shared
4	Store to Global
1	Barrier (`__syncthreads`)
79	Other

154 of 224 GB/s (GTX 980)

Kernel Fusion



Kernel Fusion



Summary

Summary

Think about the memory system.