CS344 Introduction to Parallel Programming

Lesson 7.1: Additional Parallel Computing Topics

L7.1-7.2-Overview

Parallel optimization patterns

Libraries

Programming power tools

Platforms

- Languages
- Cross-platform

Dynamic Parallelism - Stephen Jones, NVIDIA

L7.1-7.3-Parallel Optimization Patterns

Parallel Optimization Patterns

Stratton et al. In Par 2012
Computer 2012

→ survey optimized GPU codes

observe patterns that emerge

classify into 7 basic techniques

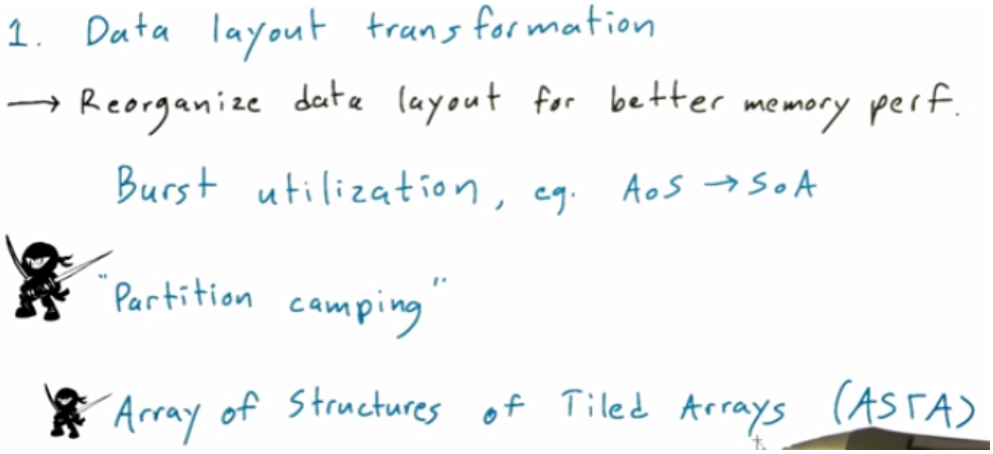
L7.1-7.4-Data Layout Transformation

1. Data layout transformation

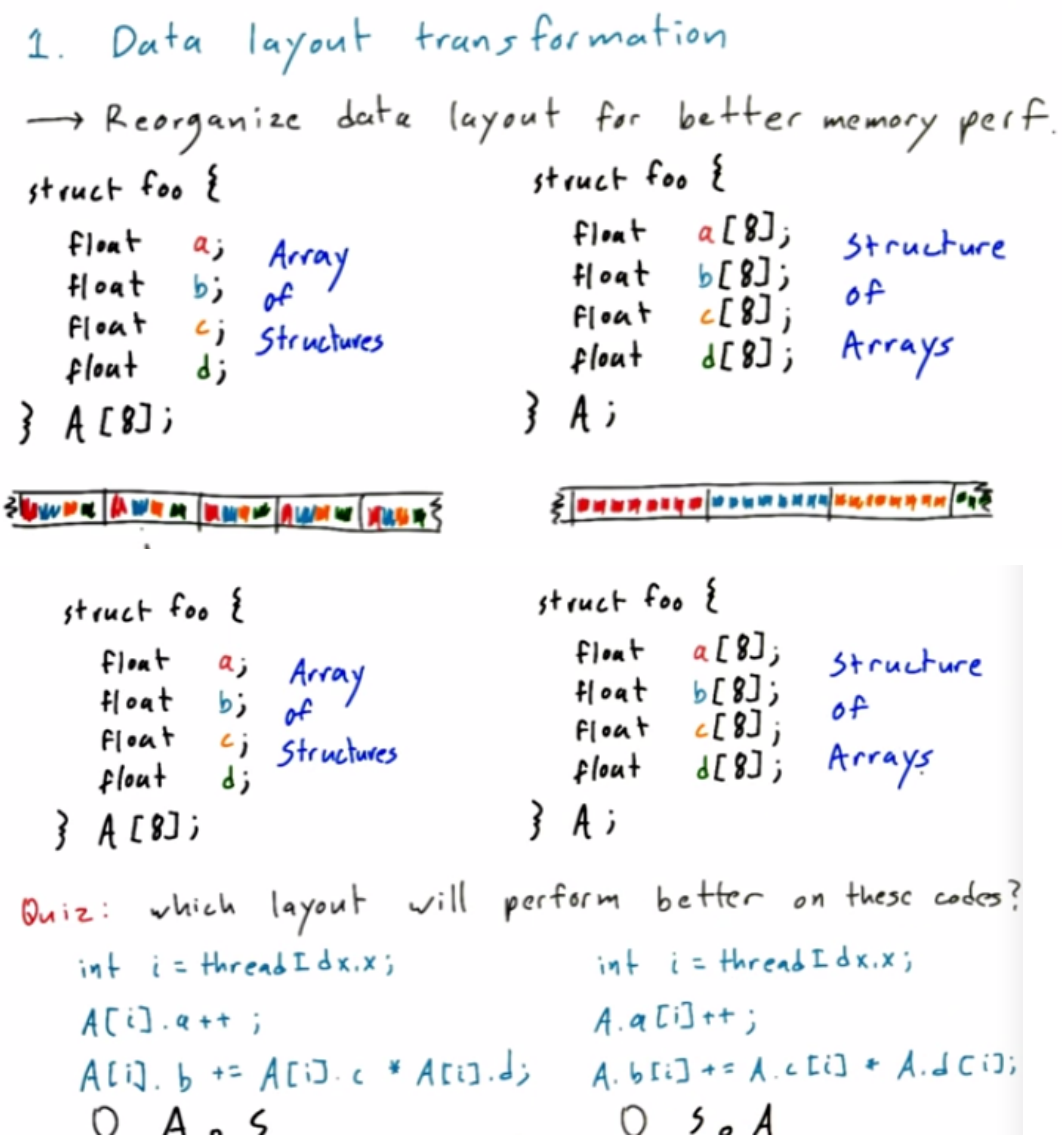
Quiz: Global memory coalescing is important because:

- ☐ DRAM systems transfer large chunks of data per transaction
- ☐ DRAM systems transfer data in many small chunks, one per thread request
- ☐ GPUs have no on-chip caches to prefetch data

L7.1-7.5-Additional Data Transformation Methods



L7.1-7.6-Burst Utilization



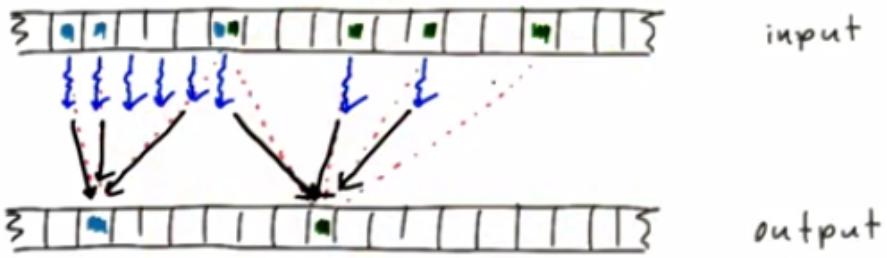
L7.1-7.7-Scatter To gather Transformation

Parallel Optimization Patterns

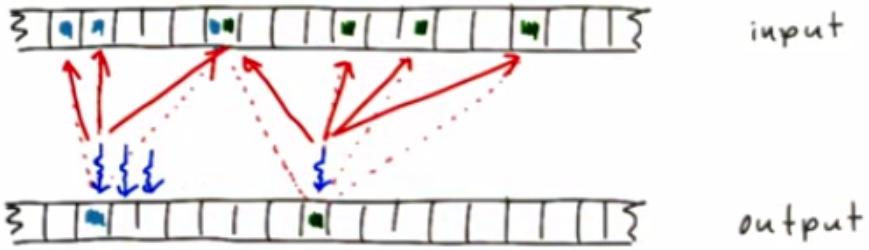
1. Data layout transformation
2. Scatter - to - gather transformation

---

2. Scatter - to - gather transformation



2. Scatter - to - gather transformation



2. Scatter - to - gather transformation

Quiz: which code snippet represents a gather? ____

which will run more efficiently? ____

①

```
float third = in[i] / 3.0f;
out[i-1] += third;
out[i]   += third;
out[i+1] += third;
```

②

```
out[i] = (in[i-1] +
          in[i] +
          in[i+1])
          / 3.0f;
```

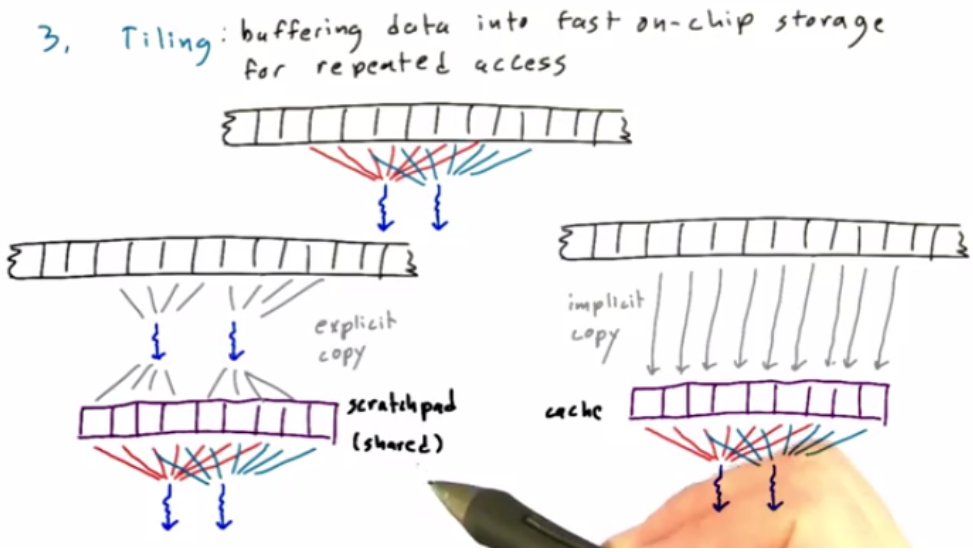
2. Scatter - to - gather transformation

Gather: many overlapping reads

Scatter: many potentially conflicting writes

→ may need to combine with #5 "binning"

L7.1-7.8-Tiling



3. Tiling: buffering data into fast on-chip storage for repeated access

Quiz: Which code snippets would benefit from tiling?

☐ Averaging values from 5 arrays?

```
__global__ void foo(float out[], float A[], float B[], float C[], float D[], float E[])
{
    int i = threadIdx.x;

    out[i] = (A[i] + B[i] + C[i] + D[i] + E[i]) / 5.0f;
}
```

☐ Averaging 5 values from 1 array?

```
__global__ void bar(float out[], float in[])
{
    int i = threadIdx.x;

    out[i] = (in[i-2] + in[i-1] + in[i] + in[i+1] + in[i+2]) / 5.0f;
}
```

L7.1-7.9-Tiled vs Non-Tilied Code

code available:
https://github.com/udacity/cs344/blob/master/Unit7%20Code%20Snippets/tiling/

L7.1-7.10-Tiling Program Quiz

3. Tiling: buffering data into fast on-chip storage for repeated access

Programming exercise: tile both foo() and bar() with 128-thread blocks and 128-element tiles

OK to ignore boundary effects at beginning and end of in[] and out[] arrays

% time of non-tiled version for foo() _____

bar() _____

L7.1-7.11-Privatization

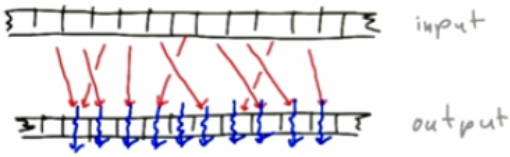
4. Privatization

threads sharing input: Good output: Bad

Histogram

L7.1-7.12-Binning

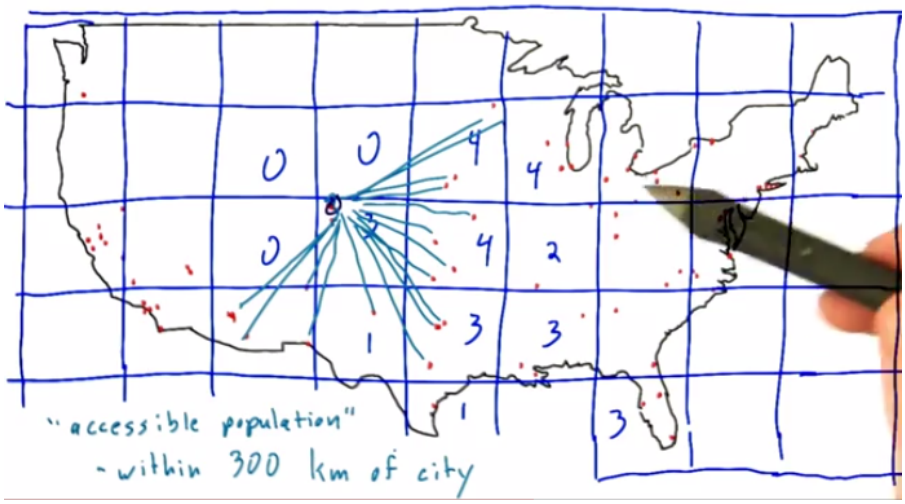
5. Binning / spatial data structures



binning: build data structure that maps output locations to the relevant input data

sort input into bins containing input elements

L7.1-7.13-Accessible Population

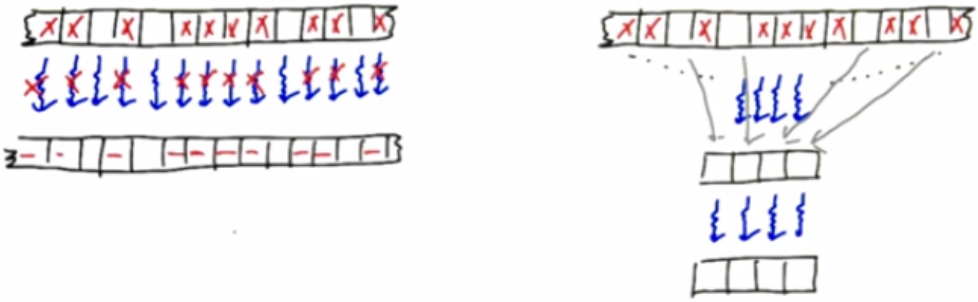


Quiz: which operations would benefit from this geographic binning?

- ☐ Compute a histogram of city distances from Denver?
- ☐ Given a list of roads (lists of line segments), compute population of cities within 20 km?
- ☐ Given a list of counties, and a list of cities each with an index to the containing county, compute the city-dwelling population of all counties

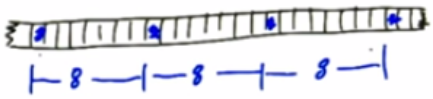
L7.1-7.14-Compaction

6. Compaction



Quiz: Suppose only every 8th element will be processed;

Assume computation-limited



About how much less time will computation take on the compacted array than the original input?

____ every 8th element?

____ every 32nd element?

____ every 128th element?

L7.1-7.15-Regularization

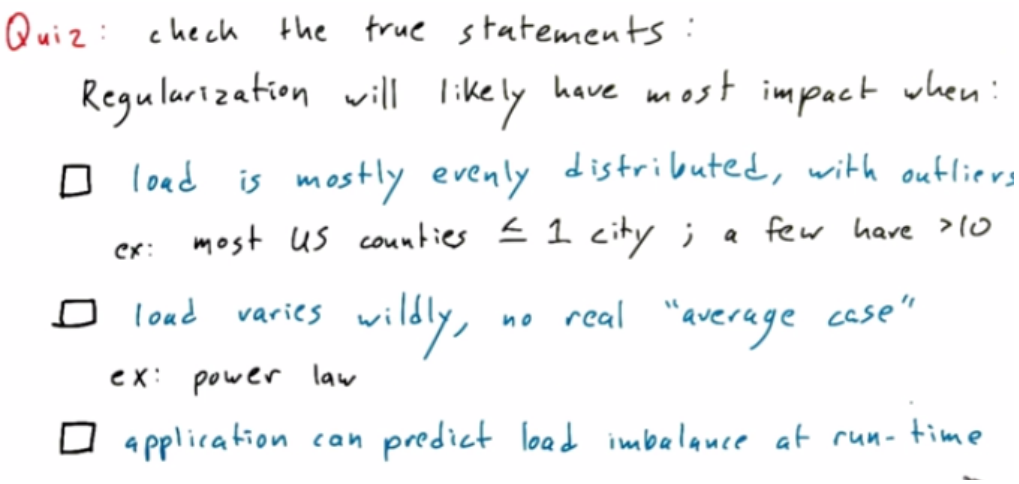
7. Regularization — load balancing

→ reorganizing input data to reduce load imbalance

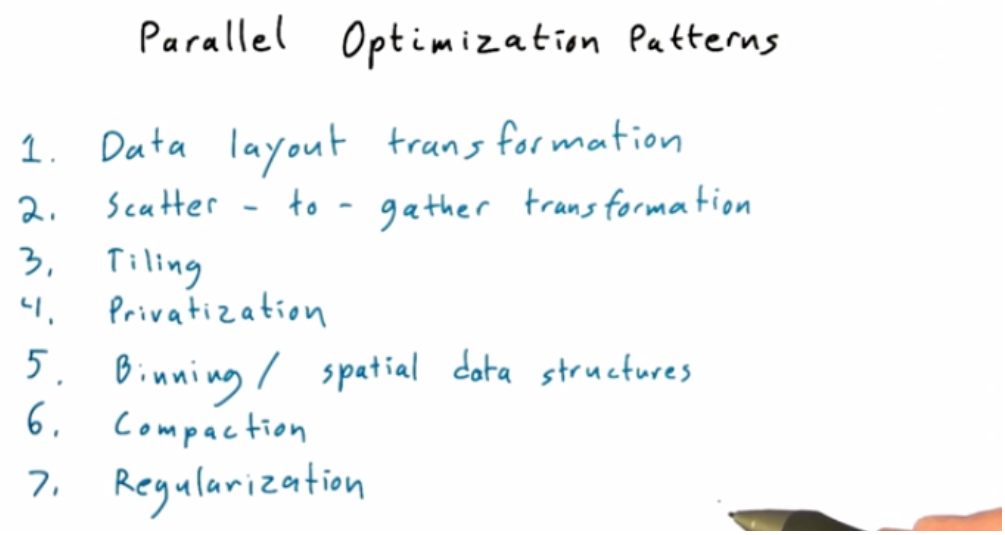
irregular parallelism → regular parallelism

ex: provision 5 cities / bin, handle overflow specially
- eg on CPU
- different kernel, different algorithm

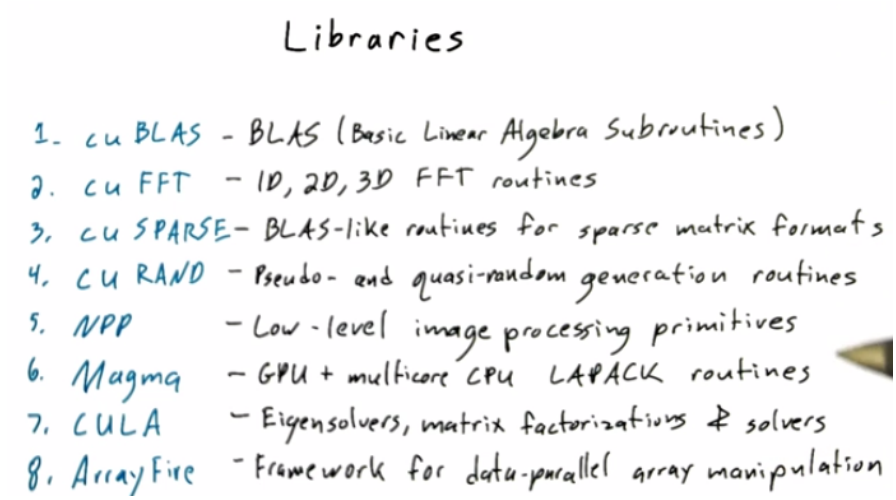
L7.1-7.16-Regularization has the most impact when



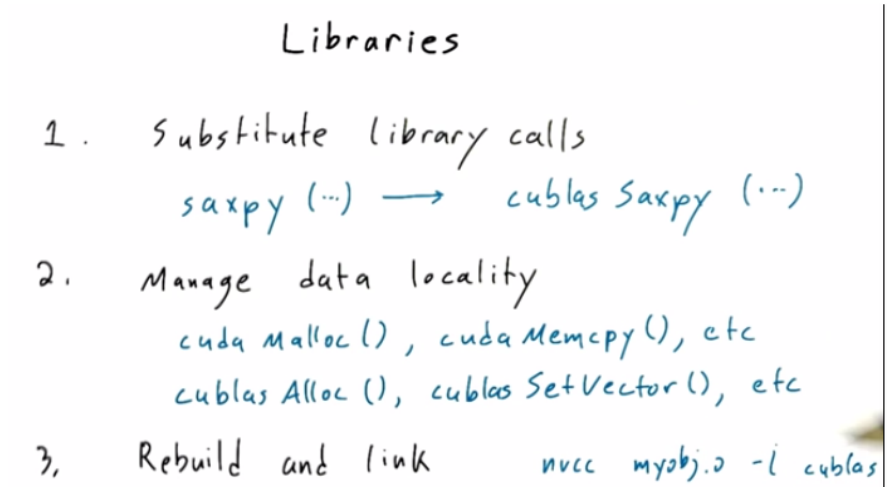
L7.1-7.17-Parallel Optimziation Patterns Recap



L7.1-7.18-Libraries



L7.1-7.19-How to Use Libraries



L7.1-7.20-cuBLAS

```
1  // Accelerated using CUBLAS
2  int N = 1 << 20;
3  cublasInit();
4  cublasAlloc(N, sizeof(float), (void**)&d_x);
5  cublasAlloc(N, sizeof(float), (void*)&d_y);
6
7  cublasSetVector(N, sizeof(x[0]), x, 1, d_x, 1);
8  cublasSetVector(N, sizeof(y[0]), y, 1, d_y, 1);
9
10 // Perform SAXPY on 1M elements: y[]=a*x[]+y[]
11 cublasSaxpy(N, 2.0, d_x, 1, d_y, 1);
12
13 cublasGetVector(N, sizeof(y[0]), d_y, 1, y, 1);
14
15 cublasFree(d_x);
16 cublasFree(d_y);
17 cublasShutdown();
18
```

L7.1-7.21-Timing Sorting in Thrust

code available:
https://github.com/udacity/cs344/tree/master/Unit7%20Code%20Snippets/thrust

1. Thrust

Analogous to C++ Standard Template Library (STL)

containers iterators ("Boost")

Host code using thrust::device_vector

- Sort, scan, reduce, reduce-by-key

```
thrust::device_vector<float> X(3);
...
float result = thrust::reduce(X.begin(), X.end());
```

1. Thrust

Analogous to C++ Standard Template Library (STL)

containers iterators ("Boost")

Host code using thrust::device_vector

- Sort, scan, reduce, reduce-by-key

```
thrust::device_vector<float> X(3);
float init = 0.0f;
float result = thrust::reduce(X.begin(), X.end(),
                              init, thrust::maximum<float>());
```

1. Thrust

Analogous to C++ Standard Template Library (STL)

containers iterators ("Boost")

Host code using thrust::device_vector

- Sort, scan, reduce, reduce-by-key
- Transform input vector(s)
  - Eg. vector addition thrust::plus
  - Apply user-defined transformation-functor

1. Thrust

Analogous to C++ Standard Template Library (STL)

containers iterators ("Boost")

Host code using thrust::device_vector

- Sort, scan, reduce, reduce-by-key
- Transform input vector(s)
- Interoperate with CUDA code

thrust::device_vector ⟷ CUDA

1. Thrust

Analogous to C++ Standard Template Library (STL)

containers iterators ("Boost")

Host code using thrust::device_vector

- Sort, scan, reduce, reduce-by-key
- Transform input vector(s)
- Interoperate with CUDA code

Avoid boilerplate, exploit high performance code

1. Thrust

Programming exercise:

sort 10^6 floats using Thrust

time sorting 10^5 floats ____ bytes ____

(in ms) 10^6 floats ____ bytes ____

10^7 floats ____ bytes ____

L7.1-7.23-CUB

CUDA C++ Programming Power Tools

1. Thrust - host-side interface, no kernels
2. CUB

Problem: software reuse in CUDA kernels

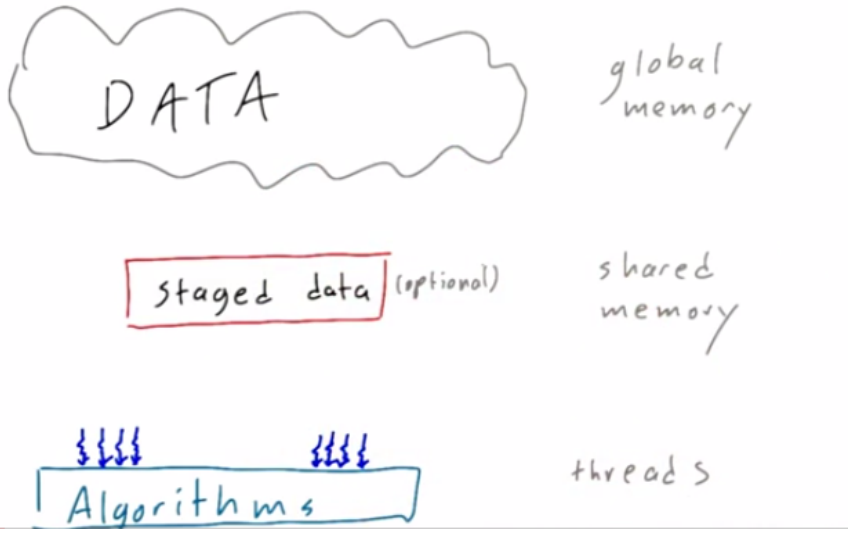
radix sort library
- how many threads?
- how much shared memory?
- which algorithm?

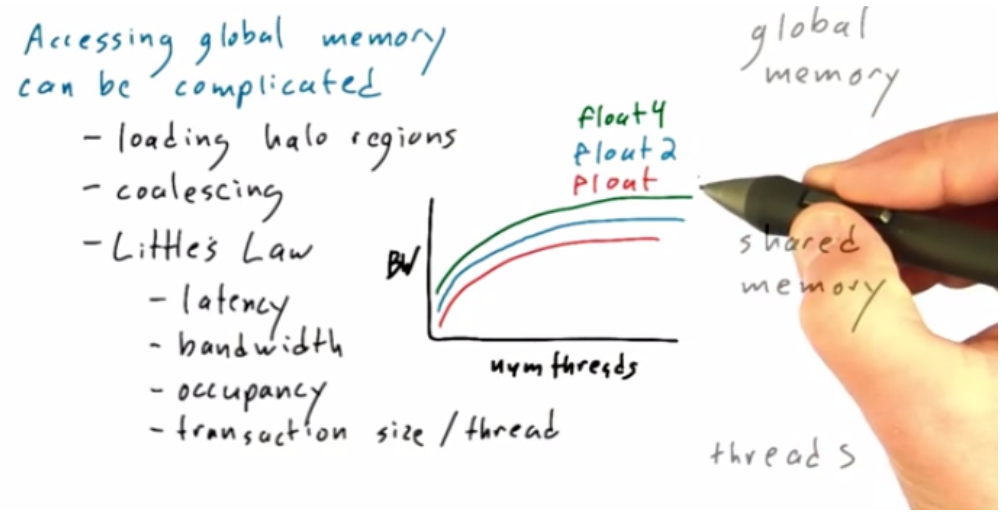
CUDA C++ Programming Power Tools

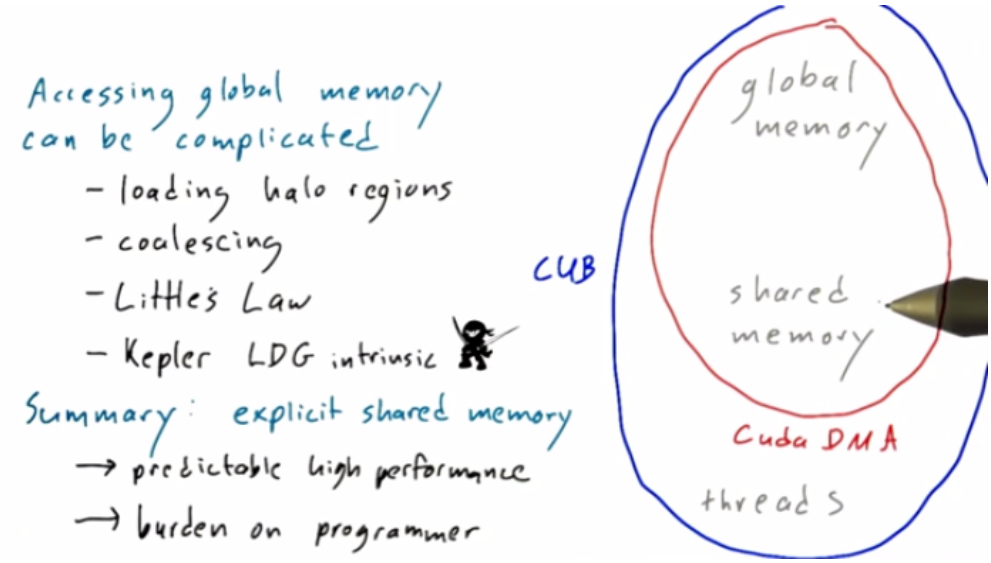
1. Thrust - host-side interface, no kernels
2. CUB - CUDA Unbound

software reuse in CUDA kernels
compile-time binding with templates, typing
high performance
enables optimization, auto tuning

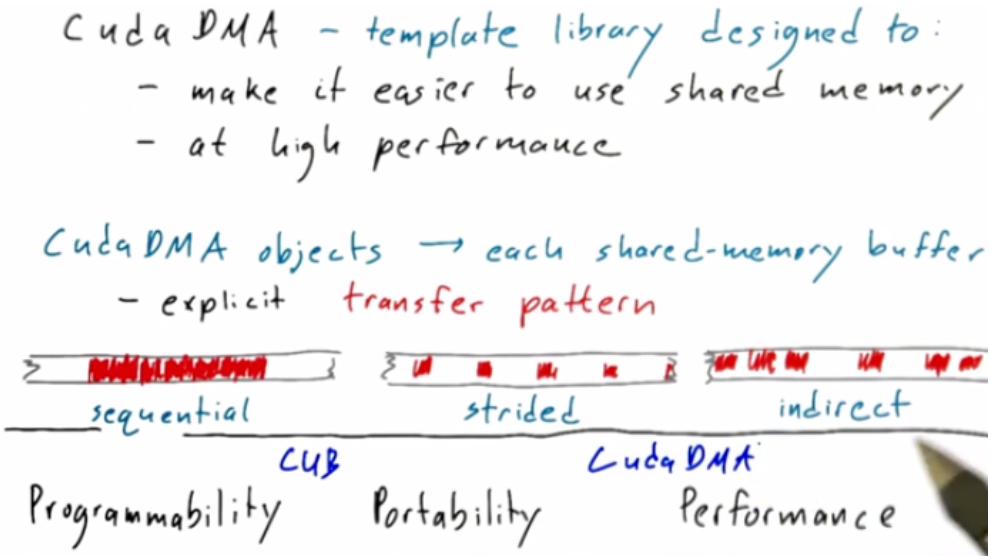
L7.1-7.24-Why Use CUB







L7.1-7.25-CUDA DMA



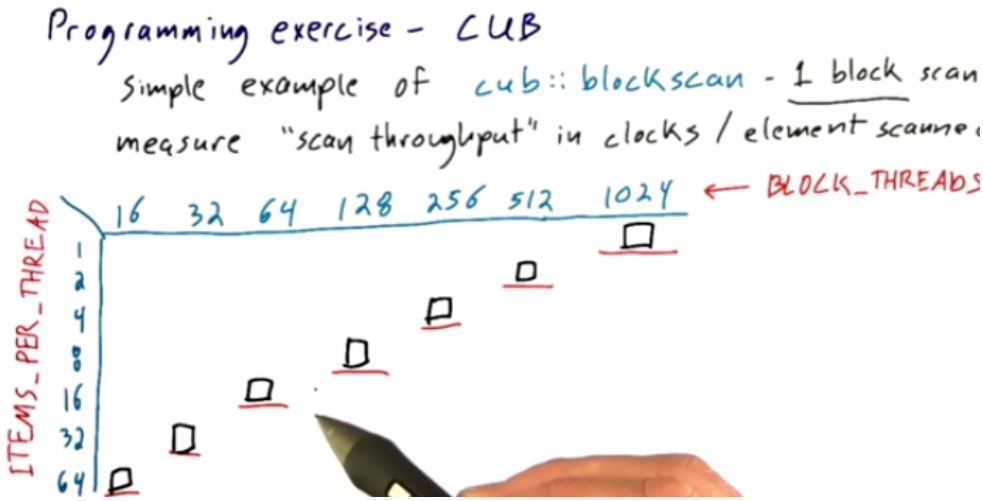
CUDA C++ Programming Power Tools

1. Thrust - host-side interface, no kernels
2. CUB - CUDA Unbound
3. Cuda DMA - global ⟷ shared memory

L7.1-7.26-CUB Exercise

code available:
https://github.com/udacity/cs344/blob/master/Unit7%20Code%20Snippets/cub/

L7.1-7.27-CUB Exercise Continued



L7.1-7.28-Other Parallel Computing Platforms

Other Platforms

"CUDA" ≠ CUDA C/C++

Other languages targeting CUDA

Cross-platform solutions

Other Languages

Wrap CUDA C++ → PyCUDA

Target CUDA directly

Copperhead — data-parallel subset of Python

- map, reduce, filter, scan, sort
- generates Thrust code
- interoperate with eg. numpy, matplotlib

Other Languages

Wrap CUDA C++ → PyCUDA

Target CUDA directly

Copperhead — data-parallel subset of Python

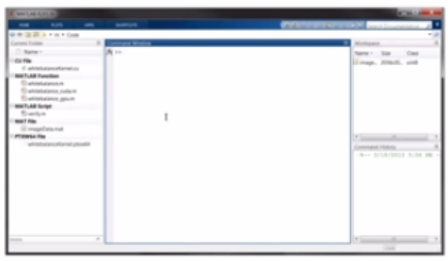
CUDA Fortran – Fortran with CUDA constructs

Halide – image processing DSL

Matlab

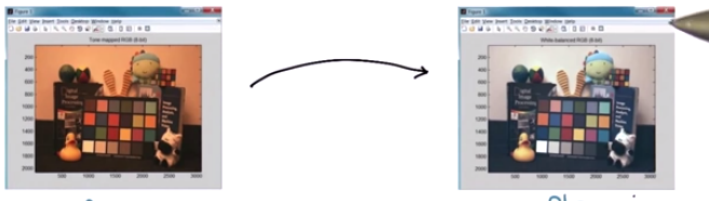
L7.1-7.29-Matlab

MATLAB



- Algorithm development
- Data analysis & viz
- Math modeling
- GPU computing

white balance



before after

L7.1-7.30-Matlab Demo

whitebalance.m

```
function adjustedImage = whitebalance(imageData)
% WHITEBALANCE forces the average image color to be gray

% Find the average values for each channel
pageSize = size(imageData,1) * size(imageData,2);
avg_rgb = mean( reshape(imageData, [pageSize,3]) );

% Find the average gray value and compute the scaling array
avg_all = mean(avg_rgb);
scaleArray = max(avg_all, 128)./avg_rgb;
scaleArray = reshape(scaleArray,1,1,3);

% Adjust the image to the new gray value
adjustedImage = uint8(bsxfun(@times,double(imageData),scaleArray));
end
```

Editor - H:\Code\whitebalance_gpu.m

whitebalance.m | whitebalance_gpu.m

```
function adjustedImage = whitebalance_gpu(imageData)
% WHITEBALANCE forces the average image color to be gray.

% Find the average values for each channel
pageSize = size(imageData,1) * size(imageData,2);
avg_rgb = mean( reshape(imageData, [pageSize,3]) );

% Find the average gray value and compute the scaling factor
avg_all = mean(avg_rgb);
factor = max(avg_all, 128)./avg_rgb;
factor = reshape(factor,1,1,3);

% Adjust the image to the new gray value
imageData = gpuArray(imageData);
adjustedImage = uint8(bsxfun(@times,double(imageData),factor));
end
```

Editor - H:\Code\whitebalance_cuda.m

whitebalance.m | whitebalance_gpu.m | whitebalanceKernel.cu | whitebalance_cuda.m

```
function adjustedImage = whitebalance_cuda(imageData)
% WHITEBALANCE forces the average image color to be gray

% Find the average values for each channel
pageSize = size(imageData,1) * size(imageData,2);
avg_rgb = mean( reshape(imageData, [pageSize,3]) );

% Find the average gray value and compute the scaling factor
avg_all = mean(avg_rgb);
factor = max(avg_all, 128)./avg_rgb;
factor = reshape(factor,1,1,3);

% Load the kernel and set up threads
kernel = parallel.gpu.CUDAKernel('whitebalanceKernel.ptxw64',...
                                 'whitebalanceKernel.cu' );
[nRows, nCols,~] = size(imageData);
blockSize = 256;
kernel.ThreadBlockSize = [blockSize, 1, 3];
kernel.GridSize = [ceil(nRows/blockSize), nCols];

```

```
>> tic,wh
Elapsed t
>>
>> tic,wh
Elapsed t
>>
```

```
6 -  avg_rgb = mean( reshape(imageData, [pageSize,3]) );
7
8    % Find the average gray value and compute the scaling factor
9 -  avg_all = mean(avg_rgb);
10 - factor = max(avg_all, 128)./avg_rgb;
11 - factor = reshape(factor,1,1,3);
12
13   % Load the kernel and set up threads
14 - kernel = parallel.gpu.CUDAKernel('whitebalanceKernel.ptxw64',...
15                                     'whitebalanceKernel.cu' );
16 - [nRows, nCols,~] = size(imageData);
17 - blockSize = 256;
18 - kernel.ThreadBlockSize = [blockSize, 1, 3];
19 - kernel.GridSize = [ceil(nRows/blockSize), nCols];
20
21   % Copy image data to the GPU and allocate and initialize return data
22 - imageDataGPU = gpuArray(imageData);
23 - adjustedImage = gpuArray.zeros( size(imageData), 'uint8');
24
25   % Adjust the image to the new gray value
26 - adjustedImage = feval( kernel, adjustedImage, imageDataGPU,...
27                          factor, nRows, nCols );
28 - end
```

Course Codes

MATLAB®

- High-level programming language
- More than 1,000,000 people use MATLAB
- Easily design/test algorithms for GPUs
  - **gpuArray** command
  - Interface to CUDA kernels

L7.1-7.31-Cross Platform Solutions

Cross - Platform solutions

GPUs, multicore CPUs

NVIDIA, AMD, Intel, Arm ...

Open CL ——— Similar to CUDA

Open GL Compute - Integrated w/ Open GL graphics

Open ACC — Directives - Great for legacy code
- evolution of Open MP
- C, C++, Fortran

L7.1-7.32-Summary