Chris Rossbach and Calvin Lin

cs380p

Outline

Over the next few classes:

Background from many areas

Architecture

Vector processors

Hardware multi-threading

Graphics

Graphics pipeline

Graphics programming models

Algorithms

parallel architectures \rightarrow parallel algorithms

Programming GPUs

CUDA

Basics: getting something working

Advanced: making it perform

Outline

Over the next few classes:

Background from many areas

Architecture

Vector processors

Hardware multi-threading

Graphics

Graphics pipeline

Graphics programming models

⁻ This lecture

Programming GPUs

CUDA

Basics: getting something working

Advanced: making it perform





Each SM has multiple vector units (4) 32 lanes wide → warp size



Each SM has multiple vector units (4) 32 lanes wide → warp size Vector units use **hardware multi-threading**





							L1 Instru	ctio	n Cache							
		L0 I	nstruc	tion <u>C</u>	ache	_		1			L0 li	nstruc	tion <u>C</u>	ach <u>e</u>		
	Wa	rp Scl	nedule	r (32 t	hread	/clk)				Wa	p Sch	edule	r (32 t	hread	l/clk)	
	Di	spatc	h Unit	(32 th	read/c	:lk)				Di	spatcl	h Unit	(32 th	read/	clk)	
	Reç	gister	File ('	16,384	4 x 32	?-bit)				Reg	ister	File (1	16,384	4 x 3	2-bit)	
FP64	INT	INT	FP32	FP32	F		HHH.		FP64	INT	INT	FP32	FP32	H		
FP64	INT	INT	FP32	FP32					FP64	INT	INT	FP32	FP32	\square		
FP64	INT	INT	FP32	FP32					FP64	INT	INT	FP32	FP32	H		
FP64	INT	INT	FP32	FP32	TEN	SOR	TENSOR		FP64	INT	INT	FP32	FP32	TE	SOR	TENSO
FP64	INT	INT	FP32	FP32		RE	CORE		FP64	INT	INT	FP32	FP32		ORE	CORE
FP64	INT	INT	FP32	FP32					FP64	INT	INT	FP32	FP32	Ħ		
FP64	INT	INT	FP32	FP32	E.				FP64	INT	INT	FP32	FP32			
FP64	INT	INT	FP32						FP64	INT	INT	FP32	FP32			
LD/ LD ST ST	LD/ ST	LD/ ST	LD/ ST	LD/ ST	LD/ ST	LD/ ST	SFU		LD/ LD/ ST ST	LD/ ST	LD/ ST	LD/ ST	LD/ ST	LD/ ST	LD/ ST	SFU
		L0 I	nstruc	tion C	ache			ΪĒ			L0 li	nstruc	tion C	ache		
			nedule									edule				
	Di	spatc	h Unit	(32 th	read/c	:lk)				Di	spatcl	h Unit	(32 th	read/	clk)	
	Reç	gister	File ('	16,384	4 x 32	?-bit)				Reg	ister	File (1	16,38	4 x 3	2-bit)	
FP64	INT	INT	FP32	FP32					FP64	INT	INT	FP32	FP32			
FP64	INT	INT	FP32	FP32	E				FP64	INT	INT	FP32	FP32			
FP64	INT	INT	FP32	FP32					FP64	INT	INT	FP32	FP32			
FP64	INT	INT	FP32	FP32		SOR	TENSOR		FP64	INT	INT	FP32	FP32		ISOR	TENSO
FP64	INT	INT	FP32	FP32	co	DRE	CORE		FP64	INT	INT	FP32	FP32	C	ORE	CORE
FP64	INT	INT	FP32	FP32					FP64	INT	INT	FP32	FP32			
FP64	INT	INT	FP32	FP32					FP64	INT	INT	FP32	FP32	H		
	INT	INT	FP32	FP32					FP64	INT	INT	FP32	FP32			
FP64		LD/	LD/ ST	LD/ ST	LD/ ST	LD/ ST	SFU		LD/ LD/ ST ST	LD/ ST	LD/ ST	LD/ ST	LD/ ST	LD/ ST	LD/ ST	SFU
FP64 LD/ LD ST ST	LD/ ST	ST														
LD/ LD		ST		_		128KI	B L1 Data Ca	che	/ Shared M	emory						



						L1 Instru	uction Cache							
			nstruc							nstruc				
					hread/clk					nedule				
					read/clk) 4 x 32-bit)				h Unit File (
				·				-		-				
FP64	INT	INT	FP32	FP32			FP64	INT	INT	FP32	FP32	Þ		
FP64	INT	INT	FP32	FP32			FP64	INT	INT	FP32	FP32	E		
FP64	INT	INT	FP32	FP32			FP64	INT	INT	FP32	FP32			
FP64	INT	INT	FP32	FP32	TENSO		FP64	INT	INT	FP32	FP32		ISOR	TENSO
FP64	INT	INT	FP32	FP32	CORE	CORE	FP64	INT	INT	FP32	FP32	со	ORE	CORE
FP64	INT	INT	FP32	FP32			FP64	INT	INT	FP32	FP32	F		
FP64	INT	INT	FP32	FP32			FP64	INT	INT	FP32	FP32	H		
FP64	INT	INT	FP32	FP32			FP64	INT	INT	FP32	FP32	H		
LD/ LD/ ST ST	LD/ ST	LD/ ST	LD/ ST	LD/ ST	LD/ LD ST S1	SFU	LD/ LD/ ST ST	LD/ ST	LD/ ST	LD/ ST	LD/ ST	LD/ ST	LD/ ST	SFU
		L0 I	nstruc	tion C	ache				L0 I	nstruc	tion C	ache		
		rp Scl	nedule	r (32 t	hread/clk				rp Scl	nedule	r (32 t	hread		
		rp Scl	nedule	r (32 t					rp Scl		r (32 t	hread		
	Di	rp Sch spatci	<mark>nedule</mark> h Unit	r (32 t (32 th	hread/clk)		Di	rp Scl ispatc	nedule	r (32 t (32 th	hread read/c	sik)	
FP64	Di	rp Sch spatci	nedule h Unit File ('	r (32 t (32 th	hread/clk read/clk)		FP64	Di	rp Scl ispatc	nedule h Unit File (r (32 t (32 th	hread read/c	sik)	
FP64 FP64	Di Reç	rp Sch spatc jister	hedule h Unit File (* FP32	r (32 t (32 th 16,384	hread/clk read/clk)		FP64 FP64	Di Reç	rp Scl ispatc gister	hedule h Unit File (FP32	r (32 t (32 th 16,38	hread read/c	sik)	
	Di Reg INT	rp Sch spatc jister INT	File (* FP32	r (32 t (32 th 16,38 FP32	hread/clk read/clk)			Di Reg INT	rp Scl ispatc gister INT	File (FP32	r (32 th (32 th 16,38 FP32	hread read/c	sik)	
FP64	Di Reç INT INT	rp Sch spatc jister INT INT	File (* File (* FP32 FP32 FP32	r (32 th (32 th 16,38 FP32 FP32	hread/clk read/clk)		FP64	Di Reç INT	rp Scl ispatc gister INT INT	File (File (FP32 FP32 FP32	r (32 th (32 th 16,38 FP32	thread/c aread/c 4 x 32	sik)	TENSO
FP64 FP64	Di Reç INT INT INT	rp Sch spatc jister INT INT INT	File (* File (* FP32 FP32 FP32 FP32 FP32	r (32 th (32 th 16,38 FP32 FP32 FP32	hread/clk read/clk) 4 x 32-bit		FP64 FP64	Di Reç INT INT INT	rp Scl ispatc gister INT INT INT	File (File (FP32 FP32 FP32 FP32 FP32	r (32 th (32 th 16,38 FP32 FP32 FP32	thread/contractions thread	2-bit)	TENSO
FP64 FP64 FP64	Di Reg INT INT INT INT	rp Sch spatc jister INT INT INT INT	File (FP32 FP32 FP32 FP32 FP32 FP32	r (32 th (32 th 16,38 FP32 FP32 FP32 FP32	hread/clk read/clk) 4 x 32-bit	R TENSOR	FP64 FP64 FP64	Di Reg INT INT INT	rp Scl ispatc gister INT INT INT INT	File (File (FP32 FP32 FP32 FP32 FP32	r (32 th (32 th 16,38 FP32 FP32 FP32	thread/contractions thread	2-bit)	
FP64 FP64 FP64 FP64	Di Reg INT INT INT INT	rp Sch spatc jister INT INT INT INT	File (* FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32	r (32 th (32 th 16,38 FP32 FP32 FP32 FP32 FP32	hread/clk read/clk) 4 x 32-bit	R TENSOR	FP64 FP64 FP64 FP64	Di Reg INT INT INT INT INT	rp Scl ispatc gister INT INT INT INT	File (FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32	r (32 th (32 th 16,38 FP32 FP32 FP32 FP32	thread/contractions thread	2-bit)	
FP64 FP64 FP64 FP64 FP64	Di Reg INT INT INT INT INT	rp Sch spatc jister INT INT INT INT INT	File (* FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32	r (32 tt (32 th 16,38- FP32 FP32 FP32 FP32 FP32 FP32 FP32	hread/clk read/clk) 4 x 32-bit	R TENSOR	FP64 FP64 FP64 FP64 FP64	Di Reg INT INT INT INT INT	rp Scl ispatc jister iNT INT INT INT	File (FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32	r (32 tt (32 tth 16,38- FP32 FP32 FP32 FP32 FP32 FP32 FP32	thread/contractions thread	2-bit)	
FP64 FP64 FP64 FP64 FP64 FP64	Di Reg INT INT INT INT INT INT	rp Scl spatc jister INT INT INT INT INT INT	File (File (FP32 FP32 FP32 FP32 FP32 FP32 FP32	r (32 tt (32 th 16,38- FP32 FP32 FP32 FP32 FP32 FP32 FP32	hread/clk read/clk) 4 x 32-bit	R TENSOR CORE	FP64 FP64 FP64 FP64 FP64 FP64	Di Reg INT INT INT INT INT INT	rp Scl ispatc jister int int int int int int	File (FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32	r (32 tt (32 tth 16,38- FP32 FP32 FP32 FP32 FP32 FP32 FP32	thread/contractions thread	2-bit)	
FP64 FP64 FP64 FP64 FP64 FP64 LD/LD/LD/	Di Reg INT INT INT INT INT INT INT INT	rp Sch spatc spatc ister int int int int int int int int int	File (' FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32	r (32 t (32 th 16,38 FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32	hread/clk) 4 x 32-bit TENSO CORE LD/ LC S1	TENSOR CORE	FP64 FP64 FP64 FP64 FP64 FP64 FP64 LD/LD/LD/	Di Reç INT INT INT INT INT INT INT INT INT	rp Scl ispatc jister INT INT INT INT INT INT INT INT INT INT	File (FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32	r (32 t (32 th 16,38 FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32	thread/ read/c 4 x 32 TEN CO	2-bit) SOR PRE	CORE

Thread block scheduler warp (thread) scheduler



							L1 Instruc	ction							
		L0 li	nstruc	tion C	ache										
					hread/c										
	Di	spatc	h Unit	(32 th	read/cll	k)								-	
	Reg	gister	File ('	16,38	4 x 32-l	bit)			Reg	ister	File ('	16,384	4 x 32	2-bit)	
FP64	INT	INT	FP32	FP32				FP64	INT	INT	FP32	FP32			
FP64	INT	INT	FP32	FP32		+		FP64	INT	INT	FP32	FP32	\vdash		
FP64	INT	INT	FP32	FP32		+		FP64	INT	INT	FP32	FP32	+		
FP64	INT	INT	FP32	FP32	TENS	OR	TENSOR	FP64	INT	INT	FP32	FP32	TEN	ISOR	TENSOR
FP64	INT	INT	FP32	FP32	COF	٩E	CORE	FP64	INT	INT	FP32	FP32	cc	DRE	CORE
FP64	INT	INT	FP32	FP32				FP64	INT	INT	FP32	FP32	H		
FP64	INT	INT	FP32	FP32				FP64	INT	INT	FP32	FP32			
FP64	INT	INT	FP32	FP32				FP64	INT	INT	FP32	FP32			
LD/ LD ST ST	ST		LD/ ST	LD/ ST	ST	LD/ ST	SFU	LD/ LD/ ST ST	LD/ ST		LD/ ST		_	LD/ ST	SFU
	ST Wa Di	ST LO II rp Sch spatc	ST Instruc Inedule h Unit	LD/ ST tion C r (32 th	ST ache hread/c read/cll	ST :lk) k)	SFU		ST War Di	ST L0 II rp Sch spatcl	ST Instruc Iedule h Unit	ST tion C r (32 tl (32 th	ST ache hread read/o	ST I/clk) clk)	SFU
	ST Wa Di	ST LO II rp Sch spatc	ST Instruc Inedule h Unit	LD/ ST tion C r (32 th	ST ache hread/c	ST :lk) k)	SFU		ST War Di	ST L0 II rp Sch spatcl	ST nstruc ledule	ST tion C r (32 tl (32 th	ST ache hread read/o	ST I/clk) clk)	SFU
	ST Wa Di	ST LO II rp Sch spatc	ST Instruc Inedule h Unit	LD/ ST tion C r (32 th (32 th	ST ache hread/c read/cll	ST :lk) k)	SFU		ST War Di	ST L0 II rp Sch spatcl	ST Instruc Iedule h Unit	sт tion C r (32 th (32 th 16,384	ST ache hread read/o	ST I/clk) clk)	SFU
ST ST	ST Wa Di Reç	ST LO II spatc gister	ST nstruc nedule h Unit File (* FP32	LD/ ST tion C r (32 th (32 th	ST ache hread/c read/cll	ST :lk) k)	SFU	ST ST	ST Wa Di Reg	ST LO II rp Sch spatcl jister	ST nstruc nedule h Unit File (*	ST tion C r (32 th (32 th 16,38 ⁴ FP32	ST ache hread read/o	ST I/clk) clk)	SFU
ST ST	ST Wa Di Reç	ST LO II rp Sch spatc gister INT	ST nedule h Unit File (* FP32 FP32	LD/ ST tion C (32 th (32 th 16,38-	ST ache hread/c read/cll	ST :lk) k)	SFU	ST ST	ST Wai Di Reg	ST LO II rp Sch spatcl jister INT	ST nstruc nedule h Unit File (* FP32 FP32	ST tion C r (32 th (32 th 16,38 ⁴ FP32	ST ache hread read/o	ST I/clk) clk)	SFU
ST S1	ST Wa Di Reç INT	ST LO II rp Sch spatc gister INT INT	ST nedule h Unit File (* FP32 FP32	LD/ ST tion C (32 th (32 th 16,38 FP32 FP32	st ache hread/c 4 x 32-l TENS	st :Ik) k) bit)	TENSOR	ST ST	ST War Di Reg	ST LO II FP Sch spatc jister INT INT	ST nstruc nedule h Unit File (* FP32 FP32	st tion C (32 th (32 th 16,384 FP32 FP32 FP32	st ache hread/ 4 x 32	ST //clk) clk) 2-bit)	TENSOR
ST S1	ST Wa Di Reç INT INT INT	ST LO II rp Sch jspatc jster INT INT	st nedule h Unit File (* FP32 FP32 FP32	LD/ ST tion C (32 th (32 th 16,384 FP32 FP32 FP32 FP32	st ache hread/cll 4 x 32-l	st :Ik) k) bit)		FP64 FP64	ST Wan Di Reg INT INT	ST LO II rp Sch spatc jister INT INT	st nstruc h Unit File (* FP32 FP32 FP32	ST tion C r (32 th (32 th 16,384 FP32 FP32 FP32 FP32	st ache hread/ 4 x 32	ST //clk) clk) 2-bit)	
ST S1 FP64 FP64 FP64 FP64 FP64 FP64	ST Wa Di Reg INT INT INT INT INT	ST LO II rp Sch spatc jister INT INT INT INT INT	st nstruct nedule h Unit FP32 FP32 FP32 FP32 FP32 FP32	LD/ ST tion C (32 th (32 th (32 th (32 th (32 th (32 th (32 th (32 th (32 th))))))))))))))))))))))))))))))))))))	st ache hread/c 4 x 32-l TENS	st :Ik) k) bit)	TENSOR	ST ST FP64 FP64 FP64 FP64	ST Wal Di Reg INT INT INT	ST LO II rp Sch spatcl ister INT INT INT	st netruc hedule h Unit FP32 FP32 FP32 FP32 FP32 FP32	ST tion C (32 th (32 th FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32	st ache hread/ 4 x 32	ST //clk) clk) 2-bit)	TENSOR
ST ST FP64 FP64 FP64 FP64 FP64 FP64	ST Wall Di Reg INT INT INT INT INT	ST LOIN rp Sch ispatc jister INT INT INT INT INT	st nstruct nedule h Unit FP32 FP32 FP32 FP32 FP32 FP32	LD/ ST tion C (32 th (32 th 16,38 FP32 FP32 FP32 FP32 FP32	st ache hread/c 4 x 32-l TENS	st :Ik) k) bit)	TENSOR	ST ST FP64 FP64 FP64 FP64 FP64	ST Wain Di Regg INT INT INT INT	ST LO II rp Sch spatcl jister INT INT INT INT	ST nstruc nedule h Unit File (* FP32 FP32 FP32 FP32	ST tion C (32 th (32 th FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32	st ache hread/ 4 x 32	ST //clk) clk) 2-bit)	TENSOR
ST ST FP64 FP64 FP64 FP64 FP64 FP64 FP64 FP64 FP64 FP64 FP64 FP64	ST Wall Wall Wall Wall Wall Wall Wall Wal	ST FD Sch Spatc Spatc INT INT INT INT INT INT INT INT	st matrice and the structure and the structure	LD/ ST tion C (32 th (32 th 16,38- FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32	st ache hread/cl 4 x 32-1 TENS COF	st k) bit)	TENSOR	57 57 FP64 FP64 FP64 FP64 FP64 FP64 FP64 FP64	ST Wall Di Di Regg INT INT INT INT INT INT INT	ST LO III rp Sch spatcl ister INT INT INT INT INT INT INT	st astruc redule h Unit FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32	st tion C r (32 th (32 th 16,384 FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32	st ache hread/ 4 x 32	st /clk) 2-bit) ISOR	TENSOR
ST ST FP64 FP64 FP64 FP64 FP64 FP64 FP64 FP64 FP64 FP64	ST Wall Wall Wall Wall Wall Wall Wall Wal	ST LOI rp Sch ispatc jister INT INT INT INT INT INT	st mstruc nedule h Unit FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32	LD/ ST tion C (32 th (32 th) (32 th) (3	st ache hread/cl 4 x 32-l TENS COF	st :Ik) k) bit)	TENSOR	ST ST FP64 FP64 FP64 FP64 FP64 FP64 FP64 FP64 FP64 FP64	ST Wan Di Regg INT INT INT INT INT INT	ST LO III rp Sch spatcl spatcl ister INT INT INT INT INT INT	ST Instruction and the second second second second second second second second second	ST tion C (32 tl (32 tl (32 th 16,384 FP32 FP32 FP32 FP32 FP32 FP32 FP32	st ache hread/ 4 x 32	ST //clk) clk) 2-bit)	TENSOR
ST ST FP64 FP64 FP64 FP64 FP64 FP64 FP64 FP64	ST Wald Di Di Reg INT INT INT INT INT INT INT INT INT	ST FD Sch Spatc Ster Ster Ster ST Ster S	st medule h Unit File (* FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32	LD/ ST tion C (32 th (32 th 16,38- FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32	st ache hread/cl 4 x 32-l TENS COF	st st st st st st st st st st st st st s	TENSOR	FP84 FP84 FP64 FP64 FP64 FP64 FP64 FP64 FP64 FP6	ST Wait Di Di Regg INT INT INT INT INT INT INT INT INT INT	ST LO II P Sch Spatcl INT INT INT INT INT INT INT INT	st nstruc redule h Unit File (° FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32	st tion C (32 th (32 th 16,384 FP32 FP32 FP32 FP32 FP32 FP32 FP32 LD/	st ache hread/k 4 x 32	st //clk) clk) //clk)//clk) //clk)/clk)	TENSOR

Programming Model

"kernels" == "shader programs"
1000s of HW-scheduled threads per kernel
Threads grouped into independent blocks.
Threads in a block can synchronize (barrier)
This is the *only* synchronization
"Grid" == "launch" == "invocation" of a kernel

a group of blocks (or warps)

Programming Model

"kernels" == "shader programs"
1000s of HW-scheduled threads per kernel
Threads grouped into independent blocks.
Threads in a block can synchronize (barrier)
This is the *only* synchronization
"Grid" == "launch" == "invocation" of a kernel

a group of blocks (or warps)



Sequential algorithms often do not permit easy parallelization

- Does not mean there work has no parallelism
- A different approach can yield parallelism
- but often changes the algorithm
- Parallelizing != just adding locks to a sequential algorithm

Sequential algorithms often do not permit easy parallelization

Does not mean there work has no parallelism

A different approach can yield parallelism

but often changes the algorithm

Parallelizing != just adding locks to a sequential algorithm

If you can express your algorithm using these patterns, an apparently fundamentally sequential algorithm can be made parallel

Key idea:

Key idea:

Express sequential algorithms as combinations of parallel patterns

Key idea:

Express sequential algorithms as combinations of parallel patterns

Examples:

Key idea:

Express sequential algorithms as combinations of parallel patterns

Examples: Map

Key idea:

Express sequential algorithms as combinations of parallel patterns

Examples: Map



Key idea:

Express sequential algorithms as combinations of parallel patterns

Examples: Map

Key idea:

Express sequential algorithms as combinations of parallel patterns

Examples: Map Reductions



Key idea:

Express sequential algorithms as combinations of parallel patterns

Examples: Map Reductions

Scans



Key idea:

Express sequential algorithms as combinations of parallel patterns

Examples: Map Reductions Scans Re-orderings (scatter/gather/sort)





Inputs Array A Function f(x)map(A, f) \rightarrow apply f(x) on all elements in A Parallelism trivially exposed f(x) can be applied in parallel to all elements, in principle



Inputs

Array A

Function f(x)

 $map(A, f) \rightarrow apply f(x) on all elements in A$

Parallelism trivially exposed

f(x) can be applied in parallel to all elements, in principle



Gather:

Read multiple items to single /packed location

Gather:

Read multiple items to single /packed location

Scatter:

Write single/packed data item to multiple locations

Gather:

Read multiple items to single /packed location

Scatter:

Write single/packed data item to multiple locations Inputs: x, y, indices, N

Gather:

Read multiple items to single /packed location Scatter:

Write single/packed data item to multiple locations Inputs: x, y, indices, N

Gather:

Read multiple items to single /packed location Scatter:

Write single/packed data item to multiple locations Inputs: x, y, indices, N



Scatter





Gather

Input Associative operator op Ordered set s = [a, b, c, ... z] Reduce(op, s) returns a op b op c ... op z

Input Associative operator op Ordered set s = [a, b, c, ... z] Reduce(op, s) returns a op b op c ... op z

for(i=0; i<N; ++i) {
 accum += (point[i]*point[i])
}</pre>



Input Associative operator op Ordered set s = [a, b, c, ... z] Reduce(op, s) returns a op b op c ... op z

for(i=0; i<N; ++i) {
 accum += (point[i]*point[i])
}</pre>

accum = reduce(*, point)

Why must op be associative?

Input Associative operator op Ordered set s = [a, b, c, ... z] Reduce(op, s) returns a op b op c ... op z



for(i=0; i<N; ++i) {
 accum += (point[i]*point[i])
}</pre>

Why must op be associative?
Scan (Prefix Sum)

Input

Associative operator **op** Ordered set s = [a, b, c, ... z] Identity I

scan(op, s) = [I, a, (a op b), (a op b op c) ...]

Scan is the workhorse of parallel algorithms:

Sort, histograms, sparse matrix, string compare, ...



Group a collection by key

Lambda function maps elements \rightarrow key

Group a collection by key

Lambda function maps elements \rightarrow key

var res = ints.GroupBy(x => x);

Group a collection by key

Lambda function maps elements \rightarrow key

var res = ints.GroupBy(x => x);



Group a collection by key

Lambda function maps elements \rightarrow key

var res = ints.GroupBy(x => x);



Group a collection by key

Lambda function maps elements \rightarrow key

```
var res = ints.GroupBy(x => x);
```



foreach(T elem in ints)
{
 key = KeyLambda(elem);
 group = GetGroup(key);
 group.Add(elem);
}

Group a collection by key

Lambda function maps elements \rightarrow key

```
var res = ints.GroupBy(x => x);
```







Process each input element in parallel



Process each input element in parallel



Process each input element in parallel



Process each input element in parallel



Process each input element in parallel



Process each input element in parallel



Process each input element in parallel



Process each input element in parallel



Process each input element in parallel



Process each input element in parallel



Process each input element in parallel



Process each input element in parallel

grouping ~ shuffling input item → output offset such that groups are contiguous output offset = group offset + item number ... but how to get the group offset, item number?



Process each input element in parallel

grouping ~ shuffling input item → output offset such that groups are contiguous output offset = group offset + item number ... but how to get the group offset, item number?

ints 30 20 10 20 10 30 10 10 10 30 30 20 10 20 res Start index of each group in the output sequence

Process each input element in parallel

grouping ~ shuffling input item \rightarrow output offset such that groups are contiguous output offset = group offset + item number ... but how to get the group offset, item number?

ints 30 20 10 20 30 10 10 20 10 10 30 20 10 30 res Start index of each Number of group in the elements in each output sequence group

Process each input element in parallel

grouping ~ shuffling input item \rightarrow output offset such that groups are contiguous output offset = group offset + item number but how to get the group offset, item number?

... but how to get the group offset, item number?



Process each input element in parallel

grouping ~ shuffling input item \rightarrow output offset such that groups are contiguous output offset = group offset + item number

... but how to get the group offset, item number?



























Thrust:

Large set of algorithms ~75 functions ~125 variations

Flexible User-defined types User-defined operators

Algorithm	Description
reduce	Sum of a sequence
find	First position of a value in a sequence
mismatch	First position where two sequences differ
inner_product	Dot product of two sequences
equal	Whether two sequences are equal
min_element	Position of the smallest value
count	Number of instances of a value
is_sorted	Whether sequence is in sorted order
transform_reduce	Sum of transformed sequence

Dwarf Popularity (Red Hot \rightarrow Blue Cool)



TBB is a collection of components for parallel programming:

- Basic algorithms: parallel_for , parallel_reduce , parallel_scan
- Advanced algorithms: parallel_while , parallel_do , parallel_pipeline , parallel_sort
- Containers: concurrent_queue, concurrent_priority_queue, concurrent_vector, concurrent_hash_map
- Memory allocation: scalable_malloc, scalable_free, scalable_realloc, scalable_calloc, scalable_allocator, cache_aligned_allocator
- Mutual exclusion: mutex , spin_mutex , queuing_mutex , spin_rw_mutex , queuing_rw_mutex , recursive_mutex
- Atomic operations: fetch_and_add , fetch_and_increment , fetch_and_decrement , compare_and_swap , fetch_and_store
- Timing: portable fine grained global time stamp
- Task scheduler: direct access to control the creation and activation of tasks



Re-expressing apparently sequential algorithms as combinations of parallel patterns is a common technique when targeting GPUs

Examples Reductions Scans Re-orderings (scatter/gather) Sort Map

What is the *right* set of parallel patterns to support?