High-Performance Scientific Computing
Lecture 7: MPI Collectives, Intro to Performance

MATH-GA 2011 / CSCI-GA 2945 · October 17, 2012

Today

Tool of the day: Valgrind

MPI

Understanding performance through asymptotics

Closer to the machine

Bits and pieces

- HW3: reports out
- HW5: due today
- HW6: out tomorrow
- Project Pitches

Outline

Tool of the day: Valgrind

MPI

Understanding performance through asymptotics

Closer to the machine

Question

Problem: Debugging only deals with problems when they cause observable wrong behavior (e.g. a crash).

Doesn't find *latent* problems.

Suggested solution: *Monitor* program behavior (precisely) while it's executing. Possible?

What is Instrumentation?

What is Instrumentation?

A.k.a. how does Valgrind work?

x86(-64) Binary
↓
IR (SSA)
↓
Tool
↓
x86(-64) Binary

Tools:

- Memcheck (find pointer bugs)
- Massif (find memory allocations)
- Cachegrind/Callgrind (find cache misbehavior)
- Helgrind/DRD (find data races)

Valgrind demo time

Outline

Tool of the day: Valgrind

MPI

- Point-to-Point, Part II
- Collectives
- Leftovers

Understanding performance through asymptotics

Closer to the machine

Outline

Ordering demo recap

MPI: Ordering

MPI 3.0, Section 3.5:

> **Order** Messages are non-overtaking : If a sender sends two messages in succession to the same destination, and both match the same receive, then this operation cannot receive the second message if the first one is still pending.
>
> If a receiver posts two receives in succession, and both match the same message, then the second receive operation cannot be satisfied by this message, if the first one is still pending.

MPI: Ordering

MPI 3.0, Section 3.5:

> **Order** Messages are non-overtaking: If a sender sends two messages in succession to the same destination, and both match the same receive, then this operation cannot receive the second message if the first one is still pending.
>
> If a receiver posts two receives in succession, and both match the same message, then the second receive operation cannot be satisfied by this message, if the first one is still pending.

MPI: More on Ordering

Possible problem?

```
if (rank == 0)
{
  MPI_Bsend(buf1, count, MPI_DOUBLE, 1, tag1, comm)
  MPI_Ssend(buf2, count, MPI_DOUBLE, 1, tag2, comm)
}
else if (rank == 1) then
{
  MPI_Recv(buf1, count, MPI_DOUBLE, 0, tag2, comm, status)
  MPI_Recv(buf2, count, MPI_DOUBLE, 0, tag1, comm, status)
}
```

MPI: Progress

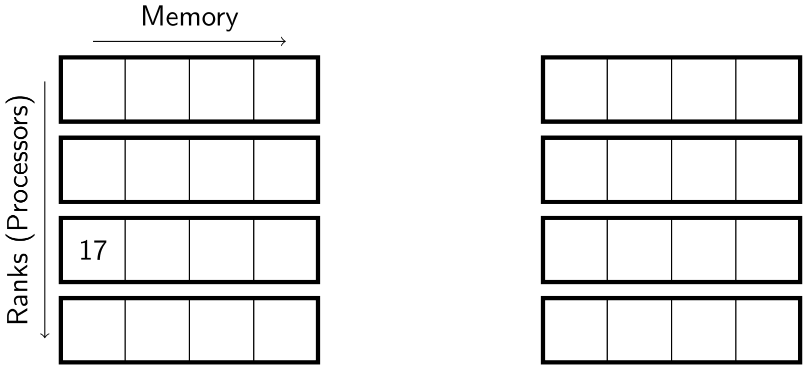
MPI 3.0, Section 3.5:

> **Progress** If a pair of matching send and receives have been initiated on two processes, then at least one of these two operations will complete, independently of other actions in the system:
>
> - the send operation will complete, unless the receive is satisfied by another message, and completes;
> - the receive operation will complete, unless the message sent is consumed by another matching receive that was posted at the same destination process.

Non-overtaking demo time

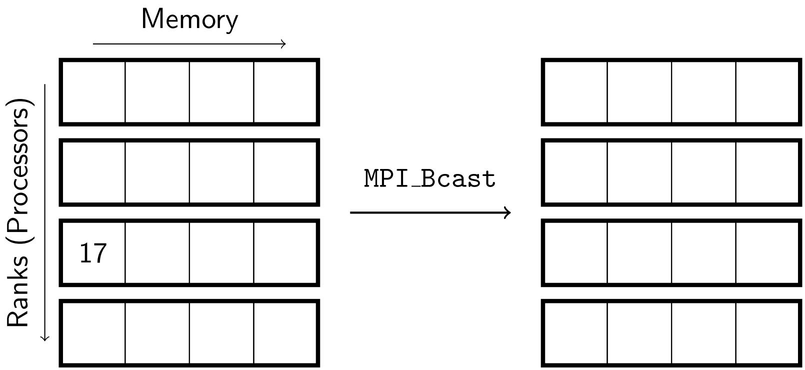
Outline

Broadcast



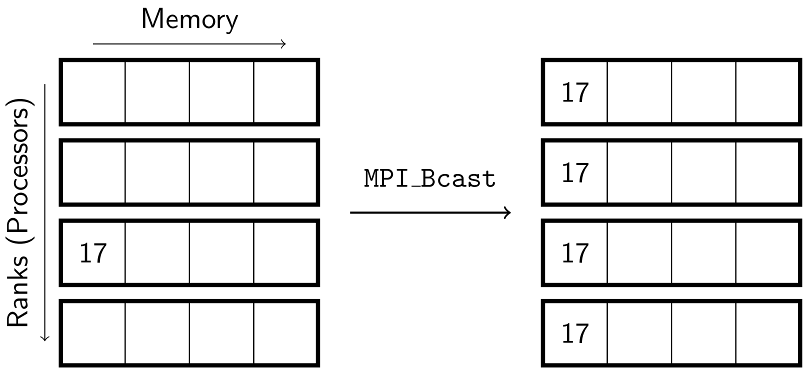
from Marsha Berger/David Bindel/Bill Gropp

Broadcast



from Marsha Berger/David Bindel/Bill Gropp

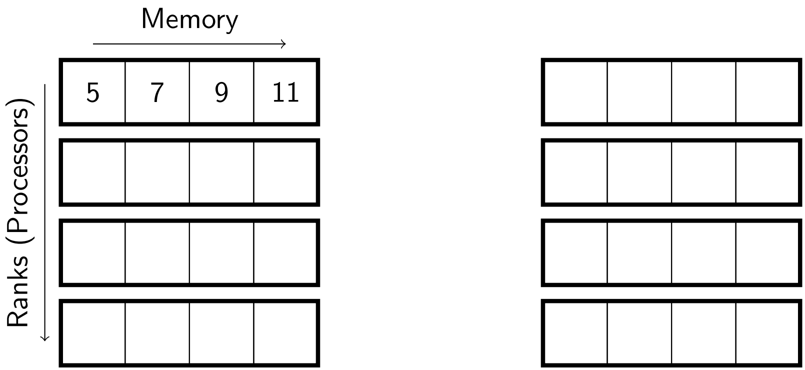
Broadcast



from Marsha Berger/David Bindel/Bill Gropp

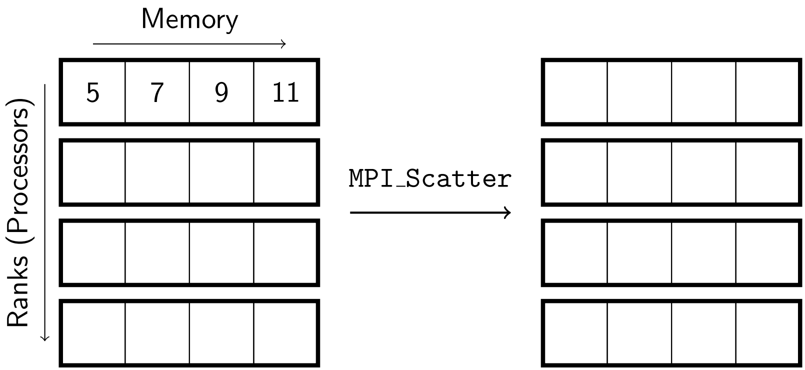
Collectives demo time

Scatter



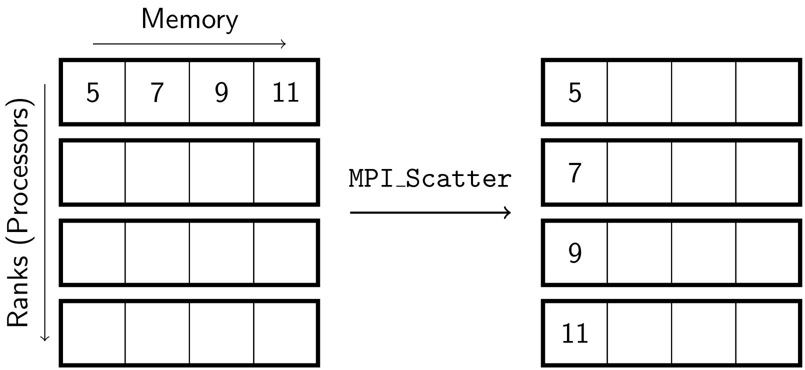
from Marsha Berger/David Bindel/Bill Gropp

Scatter



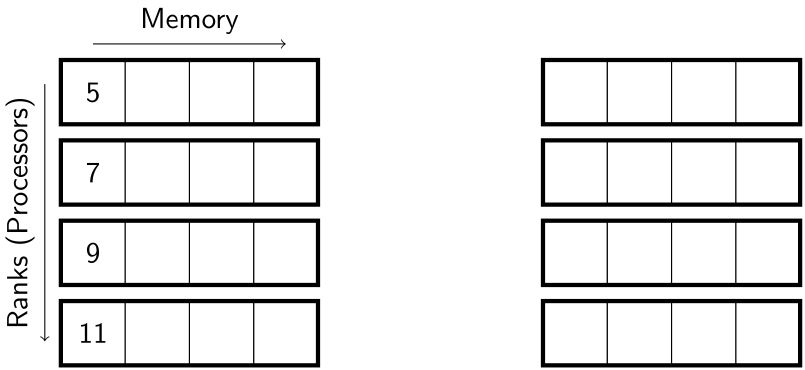
from Marsha Berger/David Bindel/Bill Gropp

Scatter



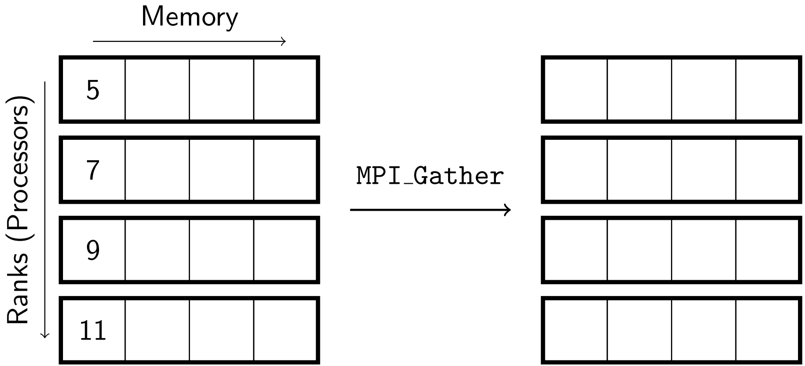
from Marsha Berger/David Bindel/Bill Gropp

Gather



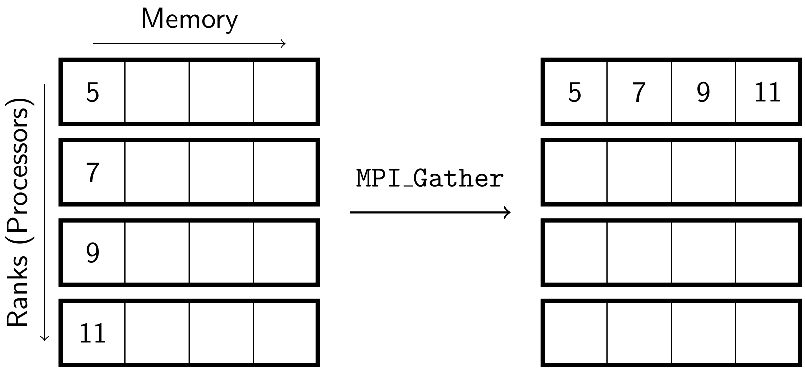
from Marsha Berger/David Bindel/Bill Gropp

Gather



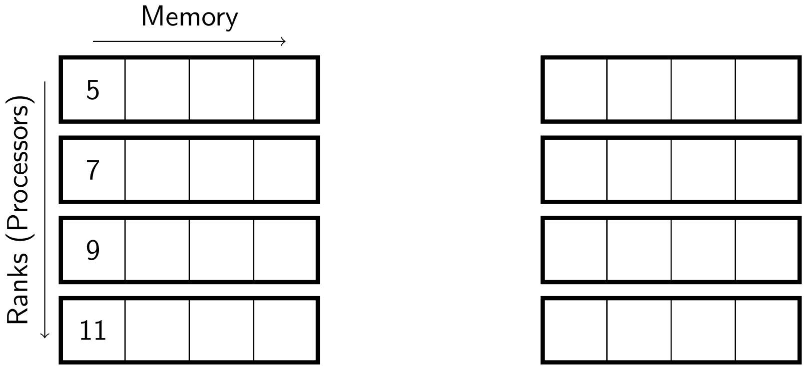
from Marsha Berger/David Bindel/Bill Gropp

Gather



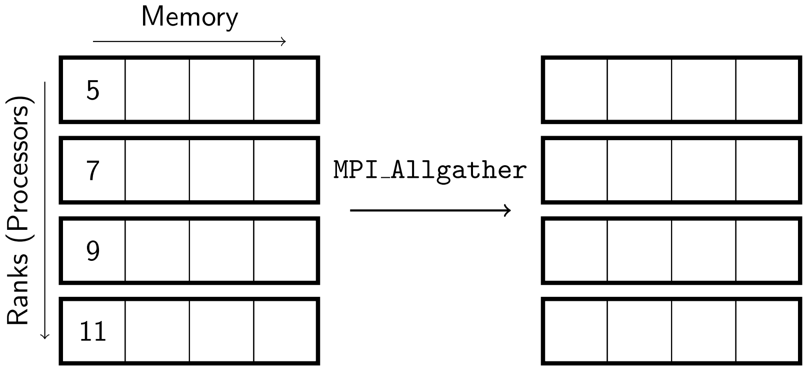
from Marsha Berger/David Bindel/Bill Gropp

All-gather



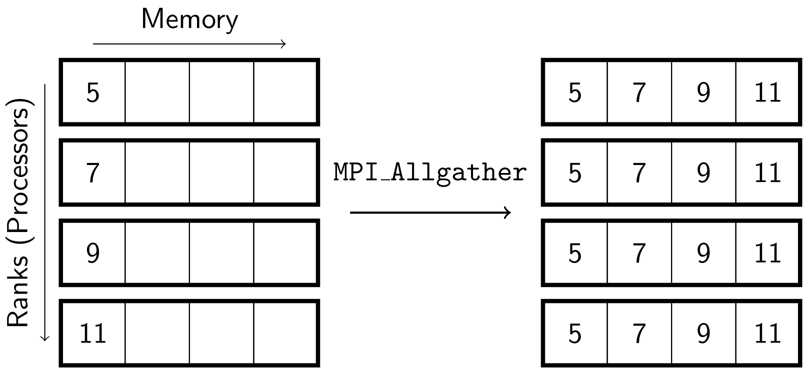
from Marsha Berger/David Bindel/Bill Gropp

All-gather



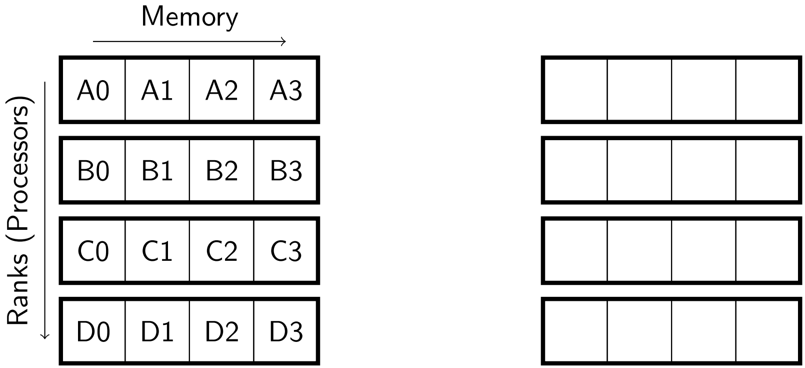
from Marsha Berger/David Bindel/Bill Gropp

All-gather



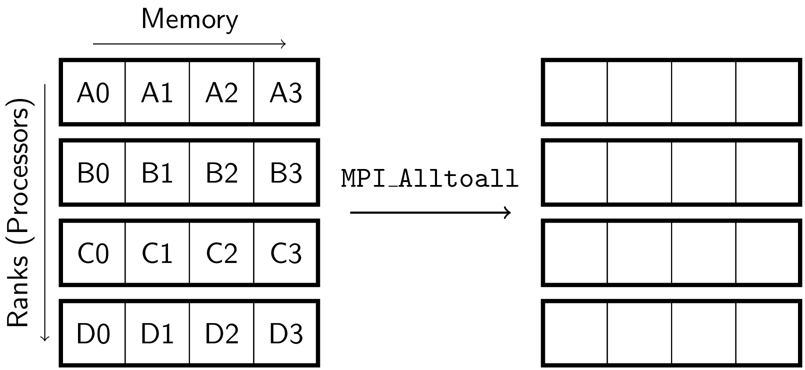
from Marsha Berger/David Bindel/Bill Gropp

All-to-all



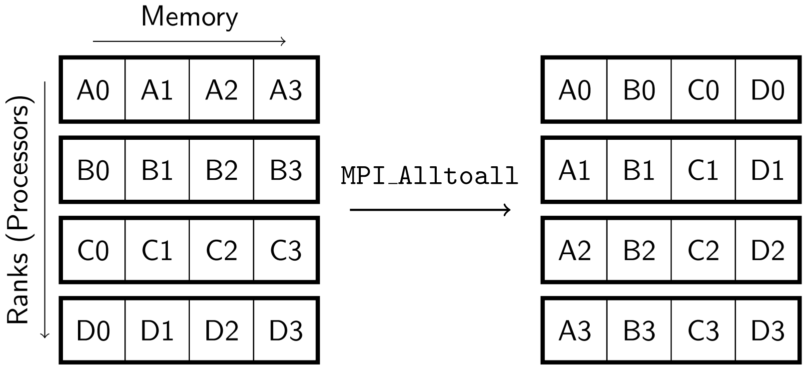
from Marsha Berger/David Bindel/Bill Gropp

All-to-all



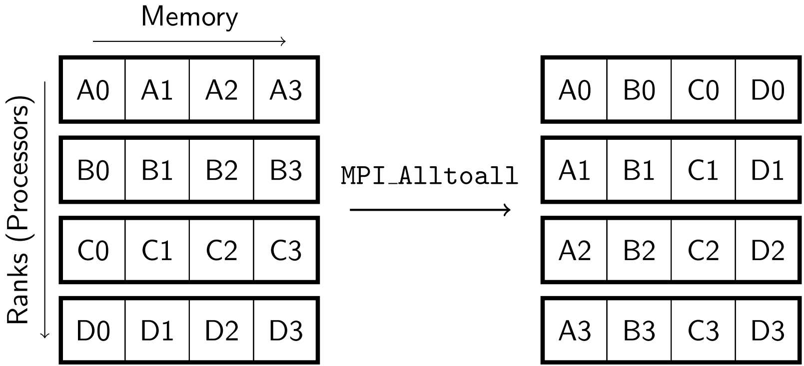
from Marsha Berger/David Bindel/Bill Gropp

All-to-all



from Marsha Berger/David Bindel/Bill Gropp

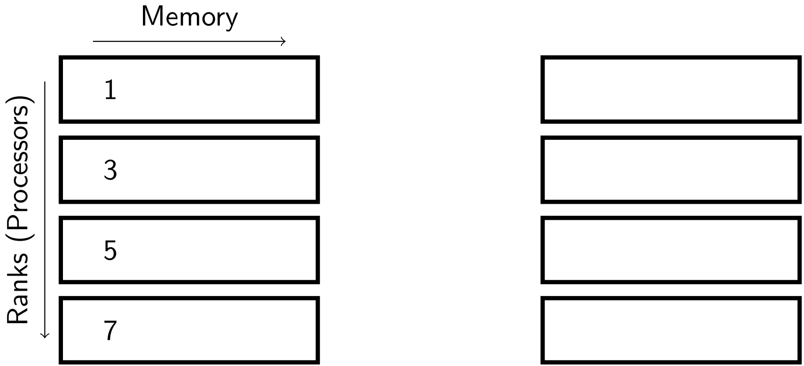
All-to-all



from Marsha Berger/David Bindel/Bill Gropp

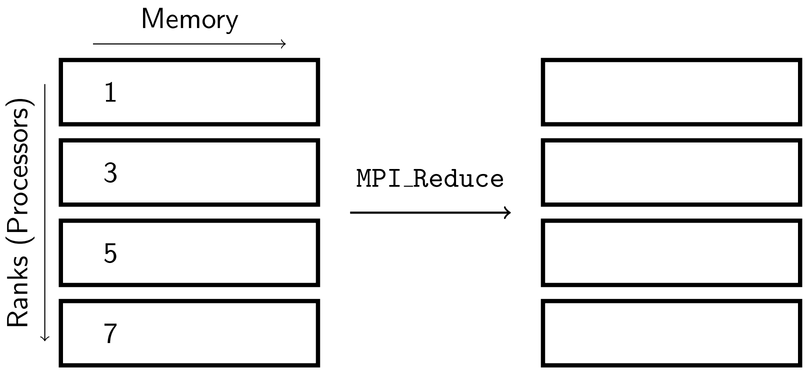
Also known as. . . ?

Reduce



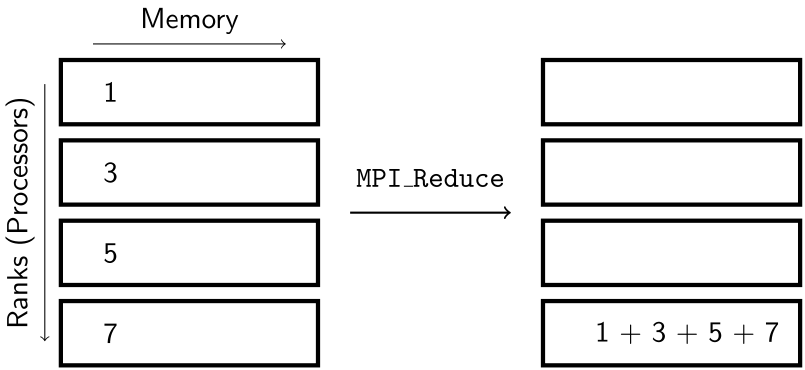
from Marsha Berger/David Bindel/Bill Gropp

Reduce



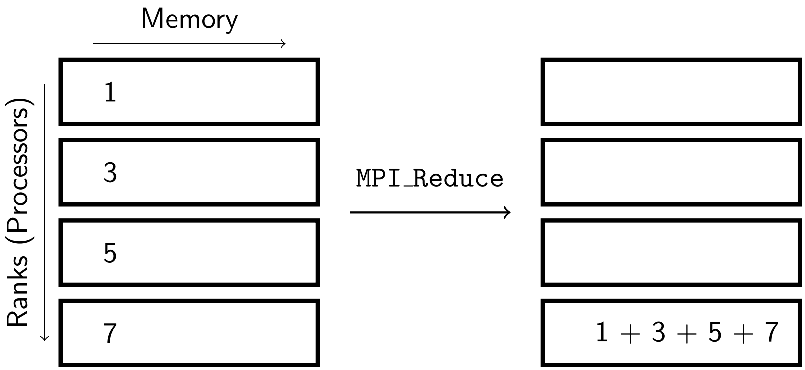
from Marsha Berger/David Bindel/Bill Gropp

Reduce



from Marsha Berger/David Bindel/Bill Gropp

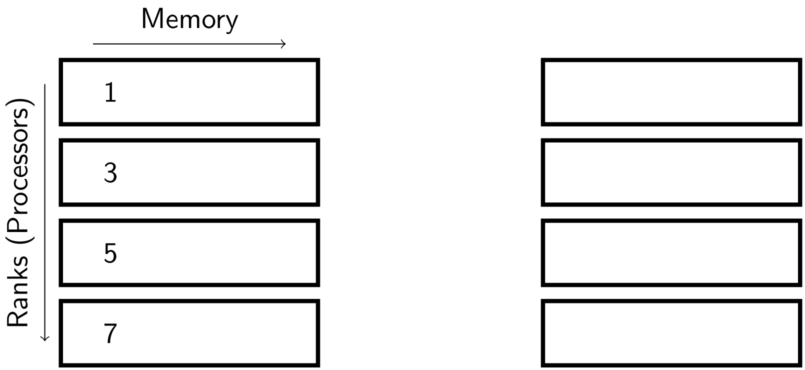
Reduce



Not just "+", also $\times$, `max`, `argmax`…

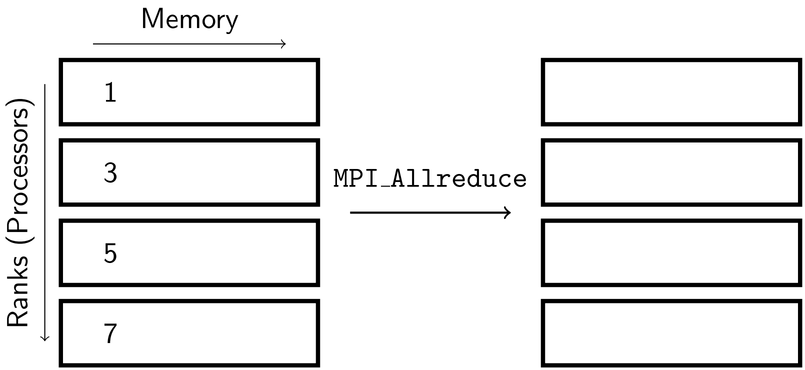
from Marsha Berger/David Bi

All-reduce



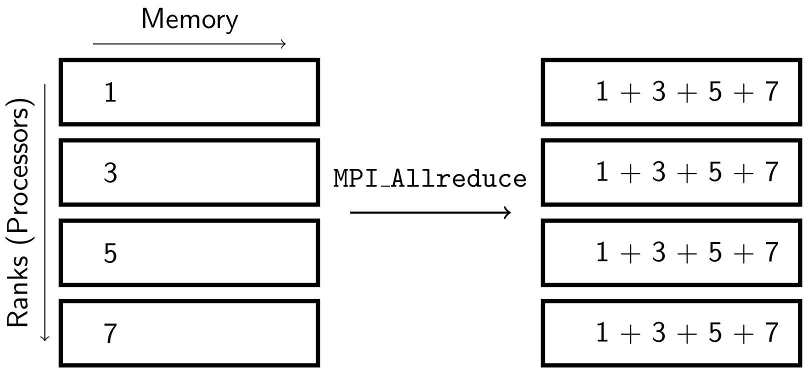
from Marsha Berger/David Bindel/Bill Gropp

All-reduce



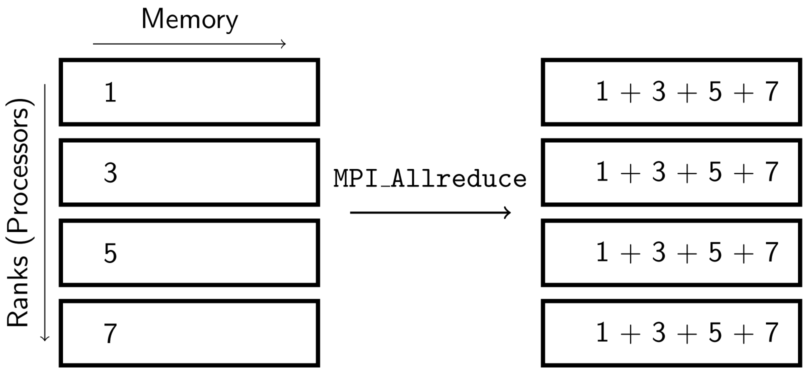
from Marsha Berger/David Bindel/Bill Gropp

All-reduce



from Marsha Berger/David Bindel/Bill Gropp

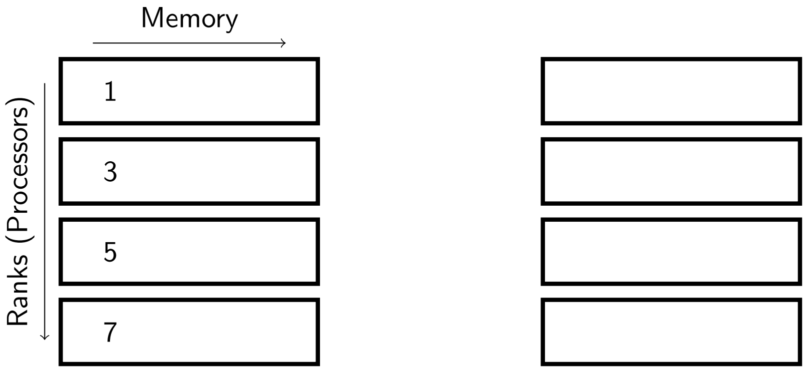
All-reduce



Often used for collective decision making.

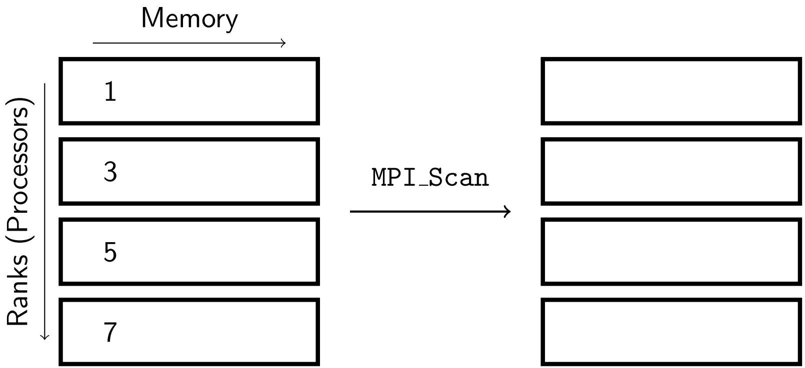
from Marsha Berger/Davi

Prefix sum



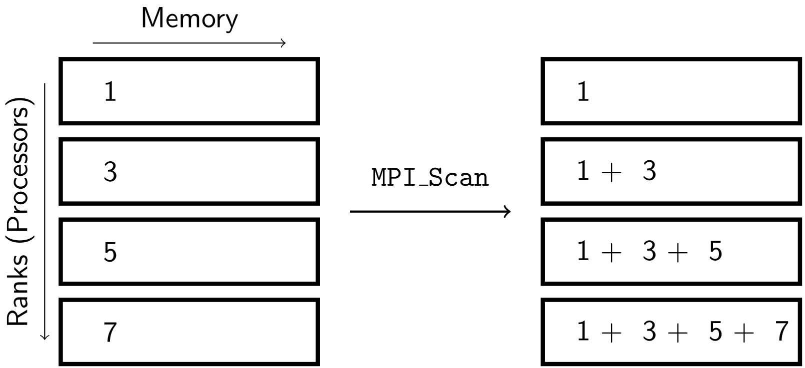
from Marsha Berger/David Bindel/Bill Gropp

Prefix sum



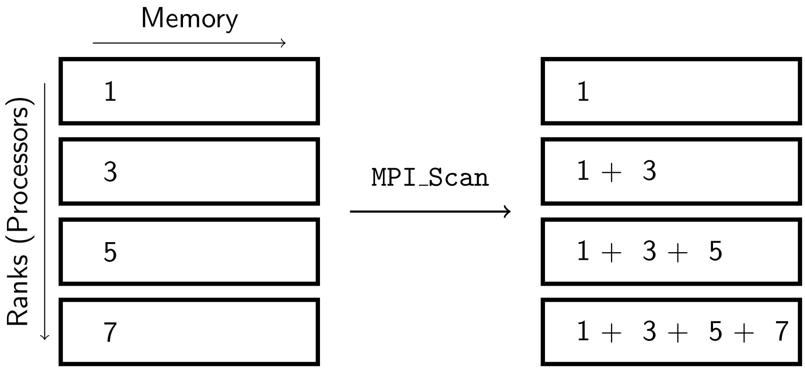
from Marsha Berger/David Bindel/Bill Gropp

Prefix sum



from Marsha Berger/David Bindel/Bill Gropp

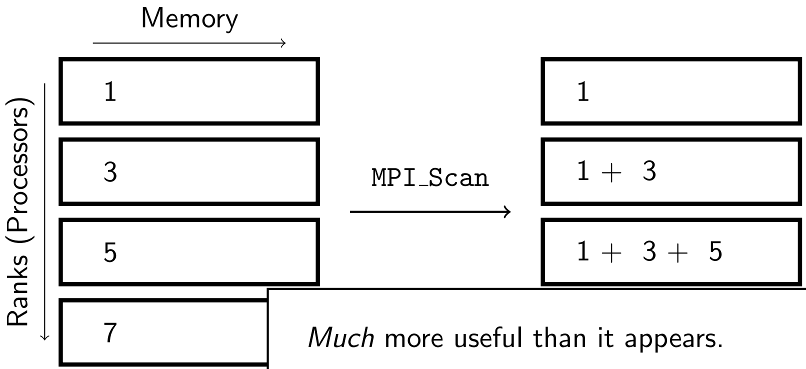
Prefix sum



Much more useful than it appears.

from Marsha Berger/David

Prefix sum



Much more useful than it appears.

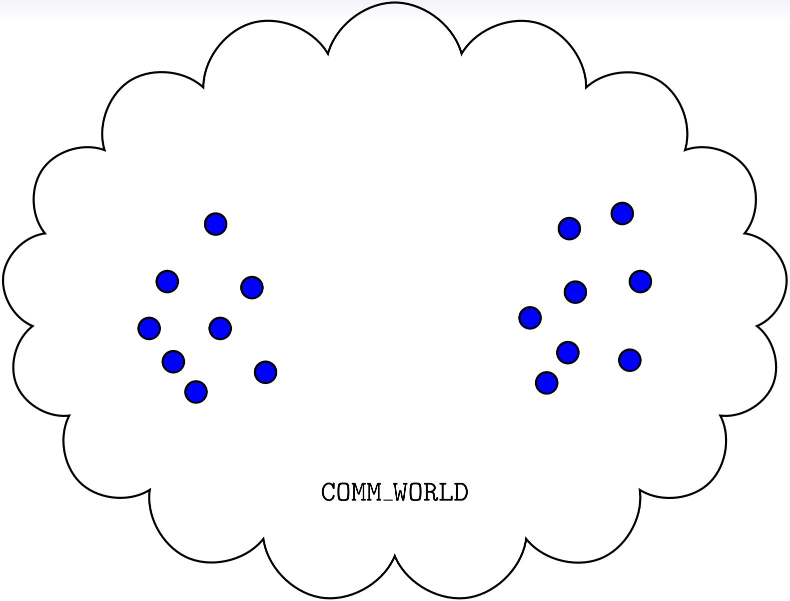
Q: How can I do collective ops on a subset of ranks?

from Marsha Berger/David

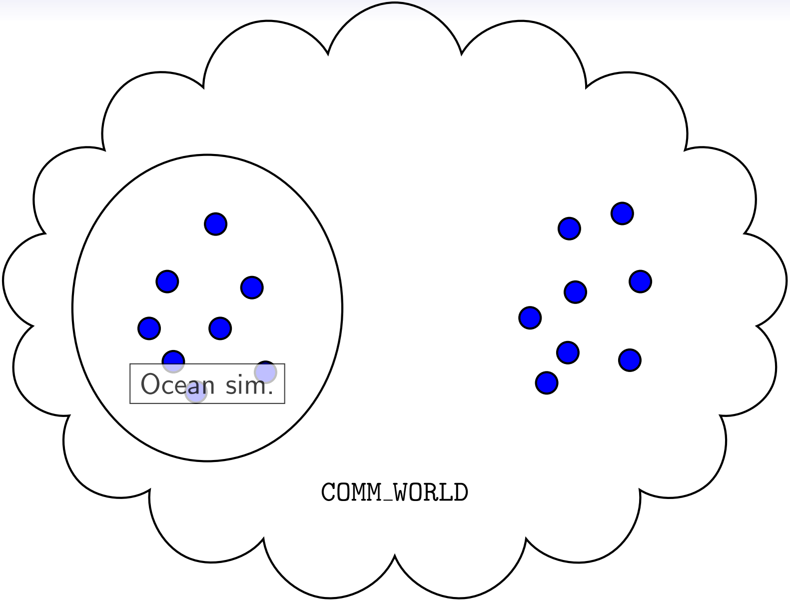
Outline

Communicators

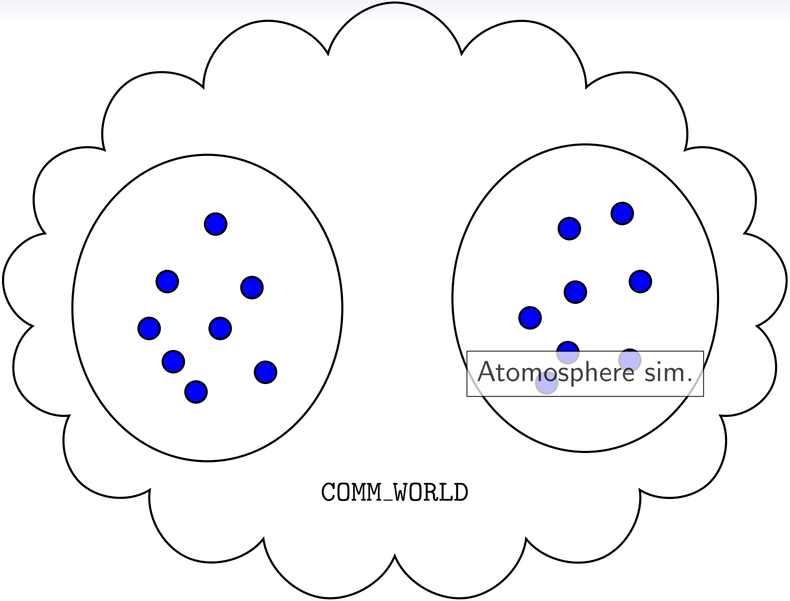
Communicators



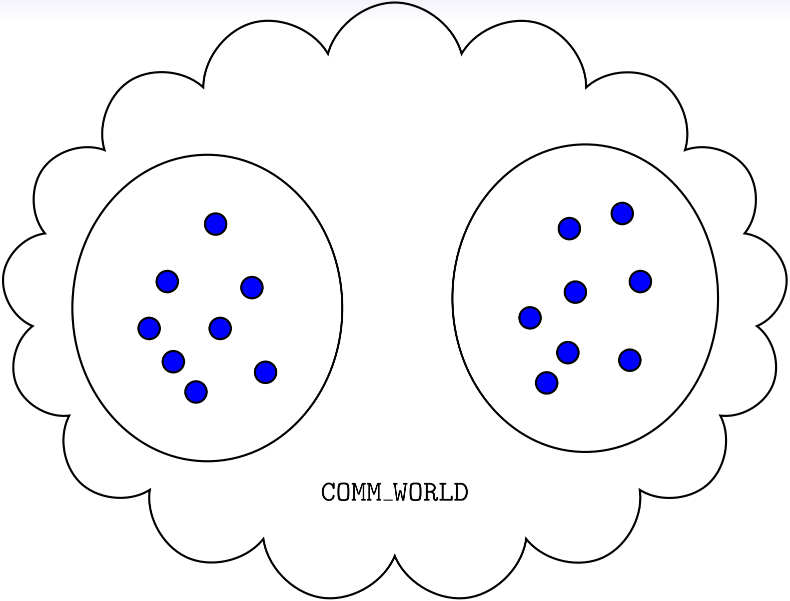
Communicators



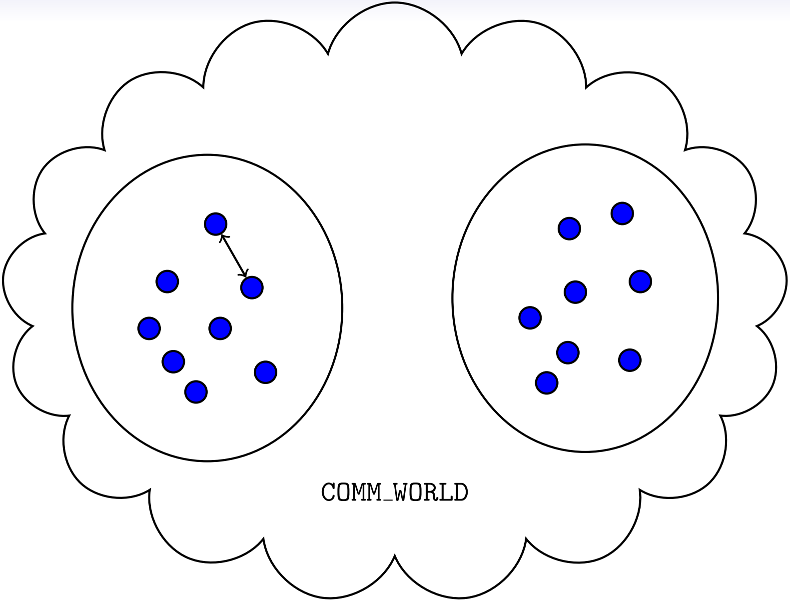
Communicators



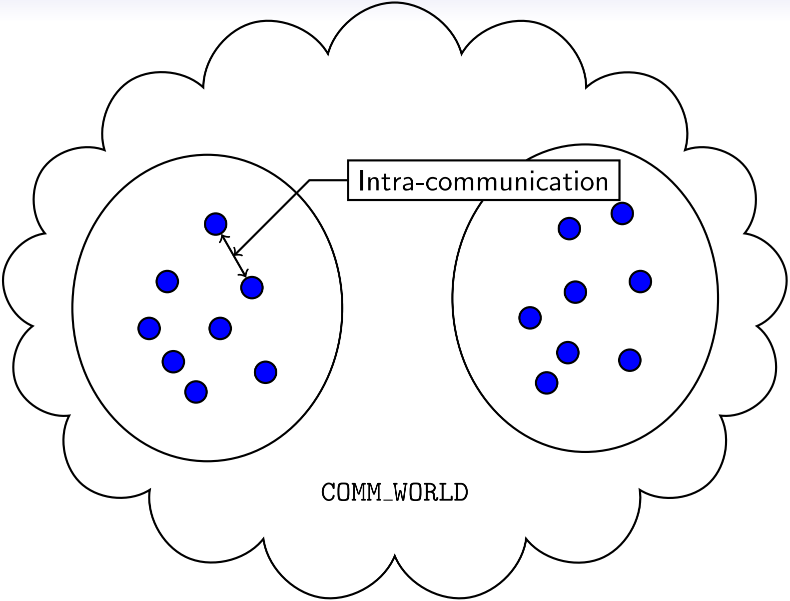
Communicators



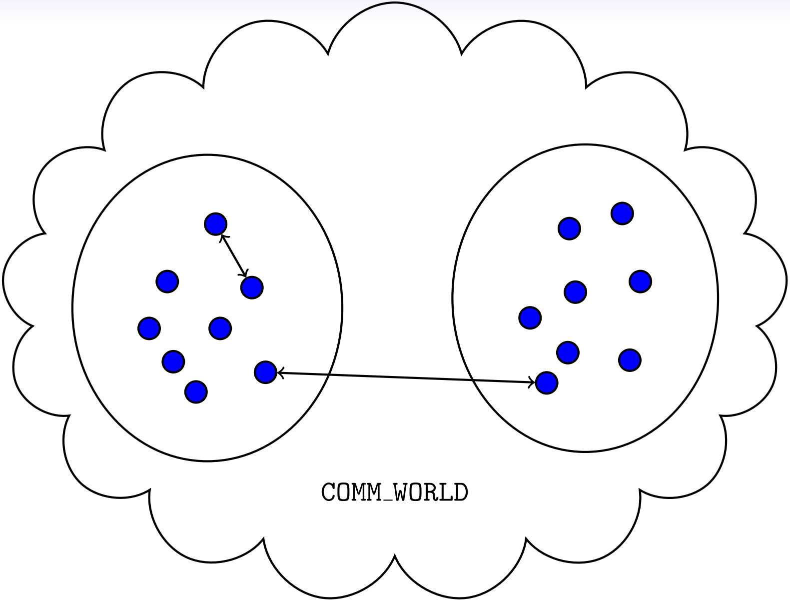
Communicators



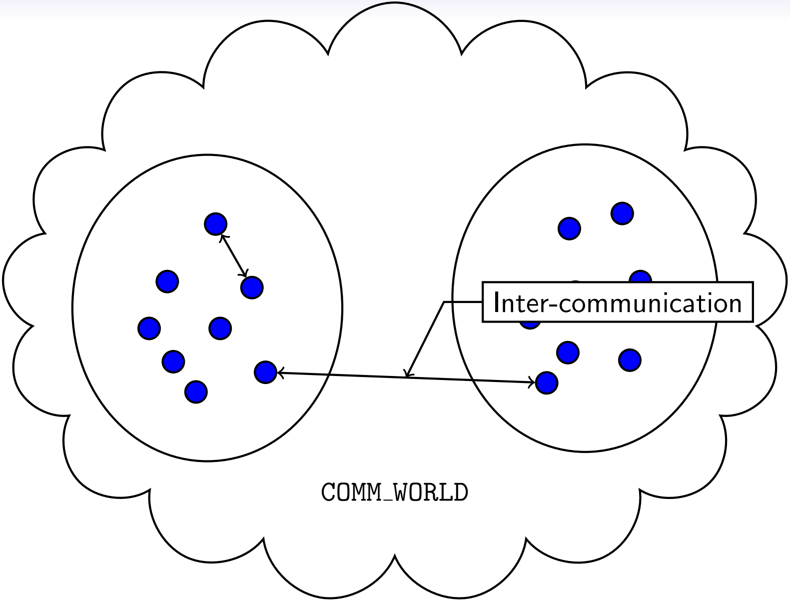
Communicators



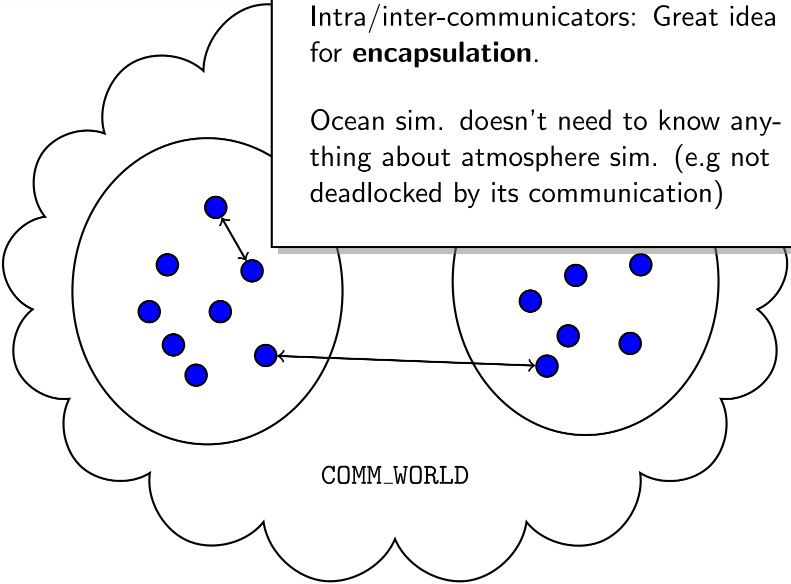
Communicators



Communicators



Communicators



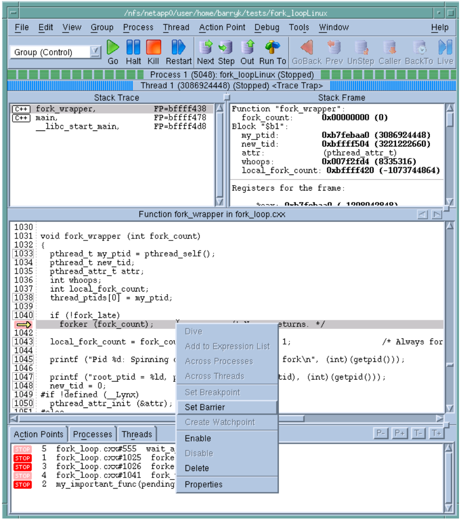
Intra/inter-communicators: Great idea for **encapsulation**.

Ocean sim. doesn't need to know anything about atmosphere sim. (e.g not deadlocked by its communication)

MPI: More shiny features

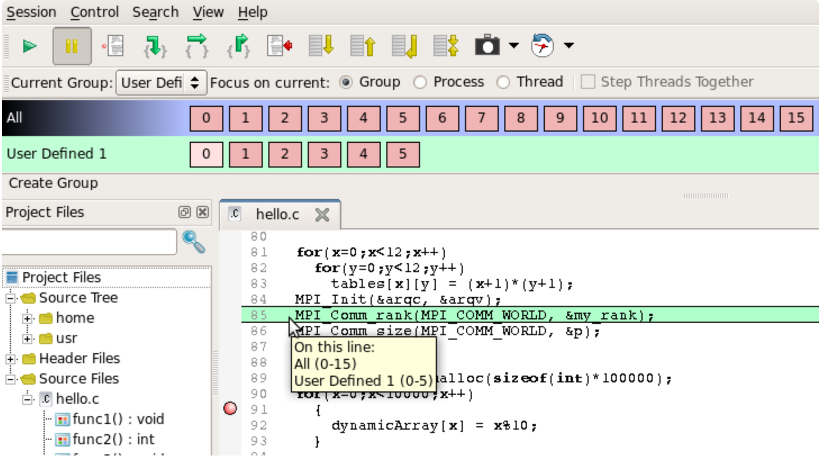
- One-sided communication
- Parallel I/O
- Create more ranks at run-time
- "Virtual topologies"
- A zoo of tools

MPI Debuggers: TotalView



TotalView (Proprietary)

MPI Debuggers: DDT



Allinea Distributed Debugging Tool (Proprietary)

MPE demo time

</ 1 >

Parallel Zoo

$$< 2 >$$

Understanding Computational Cost

Outline

Outline

Tool of the day: Valgrind

MPI

Understanding performance through asymptotics

- Work and Span
- Memory Cost
- Pebbles and I/O

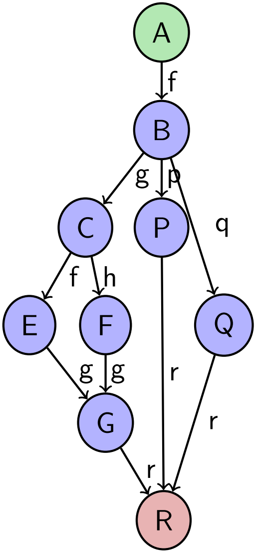
Closer to the machine

Key to parallelism: Dependencies

B = f(A)
C = g(B)
E = f(C)
F = h(C)
G = g(E,F)
P = p(B)
Q = q(B)
R = r(G,P,Q)

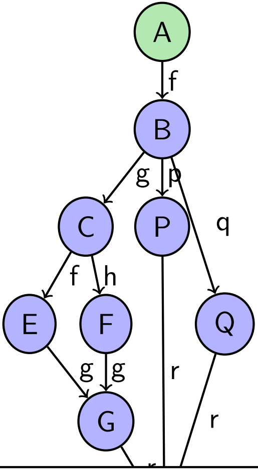
Key to parallelism: Dependencies

B = f(A)
C = g(B)
E = f(C)
F = h(C)
G = g(E,F)
P = p(B)
Q = q(B)
R = r(G,P,Q)



Key to parallelism: Dependencies

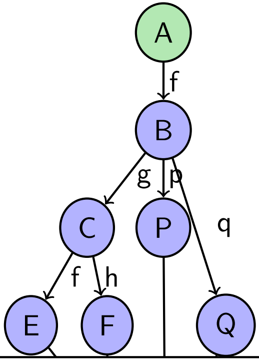
$B = f(A)$
$C = g(B)$
$E = f(C)$
$F = h(C)$
$G = g(E,F)$
$P = p(B)$
$Q = q(B)$
$R = r(G,P,Q)$



Does this look a bit like `make`?

Key to parallelism: Dependencies

B = f(A)
C = g(B)
E = f(C)
F = h(C)
G = g(E,F)
P = p(B)
Q = q(B)
R = r(G,P,Q)



Does this look a bit like `make`?

`make -j 5` runs in parallel (on 5 CPUs).

Thinking about Parallel Complexity

Let T_P be the time taken on P processors. Then;

- **Work / "Work Complexity"** T_1
  Total number of operations necessary
- **Span / "Step Complexity"** T_∞
  Minimum number of steps taken if an infinite number of processors are available
- **Parallelism** T_1/T_∞
  Average amount of work along span

Thinking about Parallel Complexity

Let T_P be the time taken on P processors. Then;

- **Work / "Work Complexity"** T_1
  Total number of operations necessary
- **Span / "Step Complexity"** T_∞
  Minimum number of steps taken if an infinite number of processors are available
- **Parallelism** T_1/T_∞
  Average amount of work along span

Does $P > T_1/T_\infty$ make sense?

Work/Span: Examples

Determine T_1 and T_∞ for:

- Adding two vectors of length n

Work/Span: Examples

Determine T_1 and T_∞ for:

- Adding two vectors of length n
- Matrix-vector multiplication ($n \times n$)

Work/Span: Examples

Determine T_1 and T_∞ for:

- Adding two vectors of length n
- Matrix-vector multiplication ($n \times n$)
- Summing a vector of length n

Work/Span: Examples

Determine T_1 and T_∞ for:

- Adding two vectors of length n
- Matrix-vector multiplication ($n \times n$)
- Summing a vector of length n
- Bubble sort

Work/Span: Examples

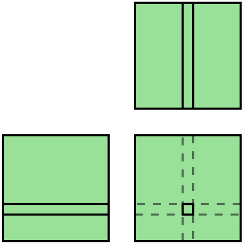
Determine T_1 and T_∞ for:

- Adding two vectors of length n
- Matrix-vector multiplication ($n \times n$)
- Summing a vector of length n
- ~~Bubble sort~~ Odd-even transposition sort

Outline

Memory Cost by Example: Matrix Multiplication

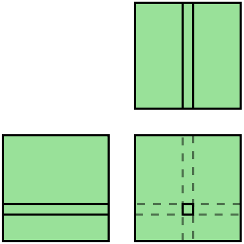
Floating Point operations: $2N^3$



Memory Cost by Example: Matrix Multiplication

Floating Point operations: $2N^3$

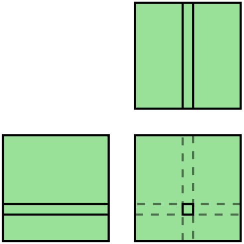
Inherent data motions:



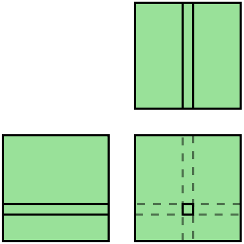
Memory Cost by Example: Matrix Multiplication

Floating Point operations: $2N^3$

Inherent data motions: $3N^2$



Memory Cost by Example: Matrix Multiplication



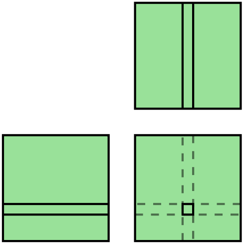
Floating Point operations: $2N^3$

Inherent data motions: $3N^2$

Inherent computational intensity:

$$\frac{\#\ \text{flops}}{\#\ \text{data motions}} = \frac{2N^3}{3N^2} \sim N$$

Memory Cost by Example: Matrix Multiplication



Floating Point operations: $2N^3$

Inherent data motions: $3N^2$

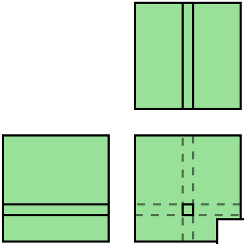
Inherent computational intensity:

$$\frac{\#\ \text{flops}}{\#\ \text{data motions}} = \frac{2N^3}{3N^2} \sim N$$

Achieved computational intensity (triple loops):

$$\frac{\#\ \text{flops}}{\#\ \text{data motions}} = \frac{2N^3}{2N^3 + O(N^2)} \sim 1$$

Memory Cost by Example: Matrix Multiplication



Floating Point operations: $2N^3$

Inherent data motions: $3N^2$

Inherent computational intensity:

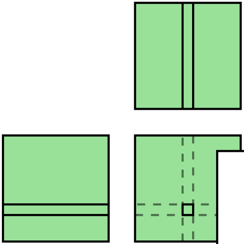
$$\frac{\#\text{ flops}}{\#\text{ data motions}} = \frac{2N^3}{3N^2} \sim N$$

Achieved computational intensity (triple

Motion: Implies a notion of "close" and "far away".

What's "good"? High CI? Low CI?

Memory Cost by Example: Matrix Multiplication



Floating Point operations: $2N^3$

Inherent data motions: $3N^2$

Inherent computational intensity:

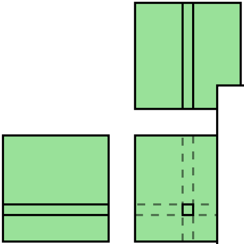
$$\frac{\#\text{ flops}}{\#\text{ data motions}} = \frac{2N^3}{3N^2} \sim N$$

Motion: Implies a notion of “close” and “far away”.

What’s “good”? High CI? Low CI?

“Moral CI”: Unachievable. Why?

Memory Cost by Example: Matrix Multiplication



Floating Point operations: $2N^3$

Inherent data motions: $3N^2$

Inherent computational intensity:

$$\frac{\#\ \text{flops}}{} = \frac{2N^3}{} \sim N$$

Motion: Implies a notion of “close” and “far away”.

What’s “good”? High CI? Low CI?

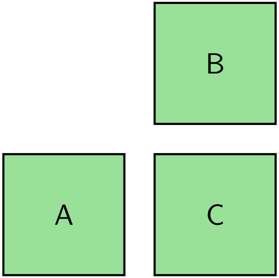
“Moral CI”: Unachievable. Why?

So, what to do?

Rearranging Matrix-Matrix Multiplication

Matrix Multiplication:

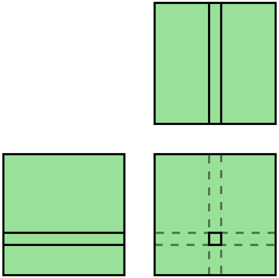
$$C_{ij} = \sum_k A_{ik} B_{kj}$$



Rearranging Matrix-Matrix Multiplication

Matrix Multiplication:

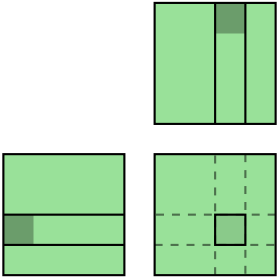
$$C_{ij} = \sum_k A_{ik} B_{kj}$$



Rearranging Matrix-Matrix Multiplication

Matrix Multiplication:

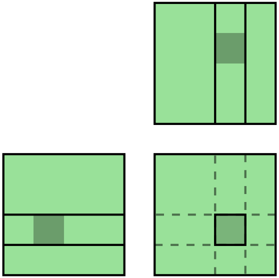
$$C_{ij} = \sum_k A_{ik} B_{kj}$$



Rearranging Matrix-Matrix Multiplication

Matrix Multiplication:

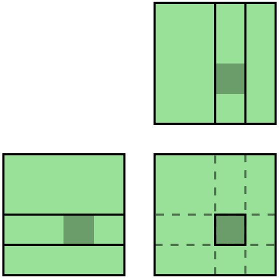
$$C_{ij} = \sum_k A_{ik} B_{kj}$$



Rearranging Matrix-Matrix Multiplication

Matrix Multiplication:

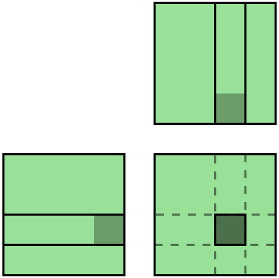
$$C_{ij} = \sum_k A_{ik} B_{kj}$$



Rearranging Matrix-Matrix Multiplication

Matrix Multiplication:

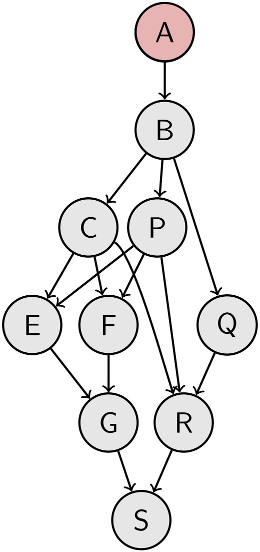
$$C_{ij} = \sum_k A_{ik} B_{kj}$$



Outline

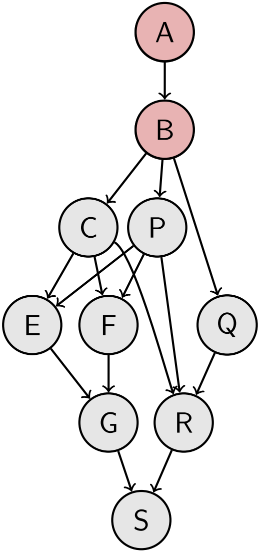
How much memory is needed?

How much memory do we need to evaluate this DAG?



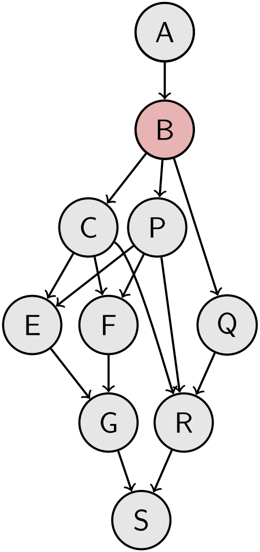
How much memory is needed?

How much memory do we need to evaluate this DAG?



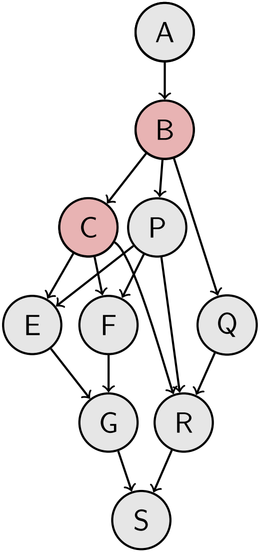
How much memory is needed?

How much memory do we need to evaluate this DAG?



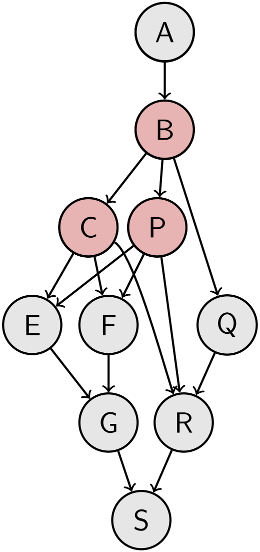
How much memory is needed?

How much memory do we need to evaluate this DAG?



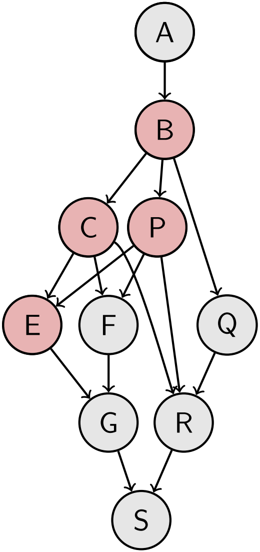
How much memory is needed?

How much memory do we need to evaluate this DAG?



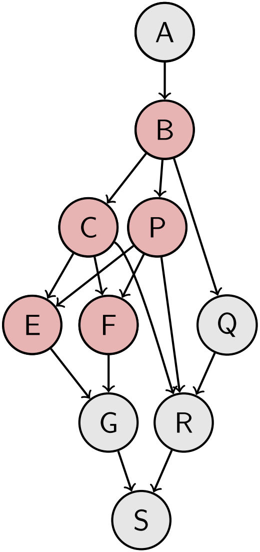
How much memory is needed?

How much memory do we need to evaluate this DAG?



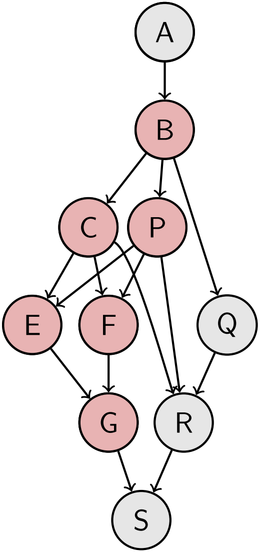
How much memory is needed?

How much memory do we need to evaluate this DAG?



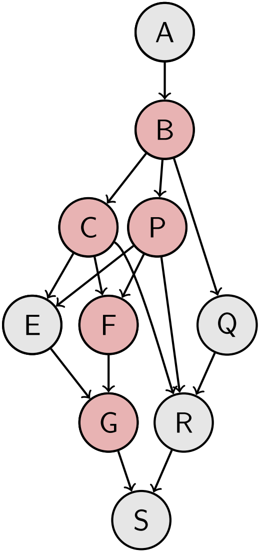
How much memory is needed?

How much memory do we need to evaluate this DAG?



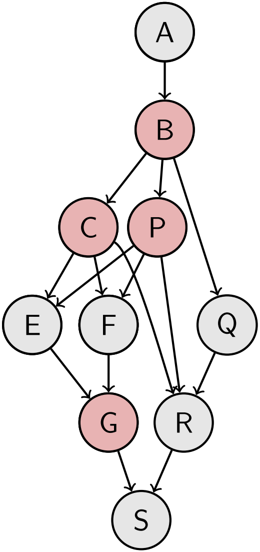
How much memory is needed?

How much memory do we need to evaluate this DAG?



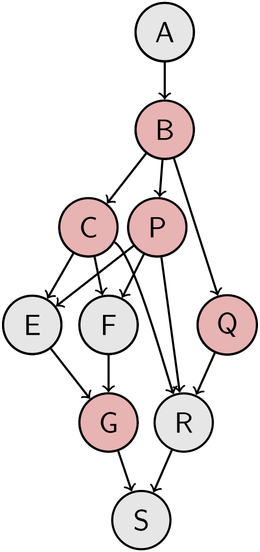
How much memory is needed?

How much memory do we need to evaluate this DAG?



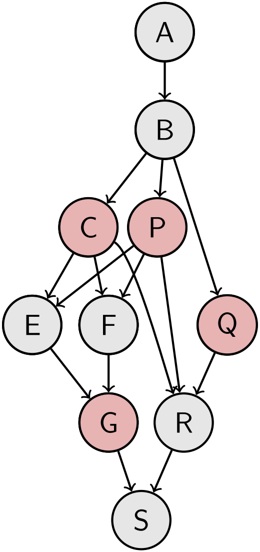
How much memory is needed?

How much memory do we need to evaluate this DAG?



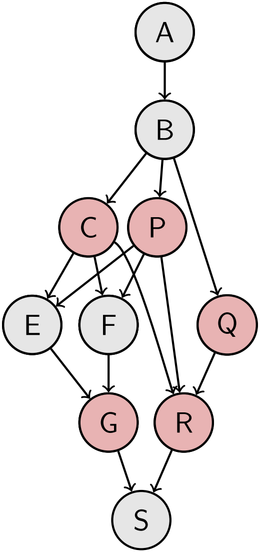
How much memory is needed?

How much memory do we need to evaluate this DAG?



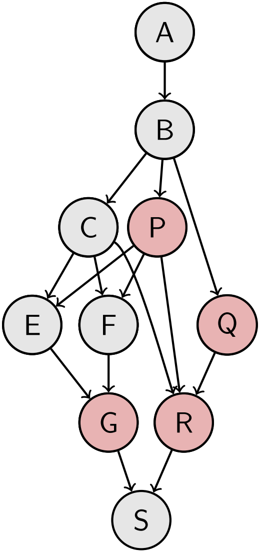
How much memory is needed?

How much memory do we need to evaluate this DAG?



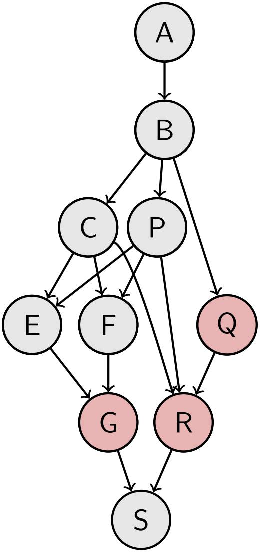
How much memory is needed?

How much memory do we need to evaluate this DAG?



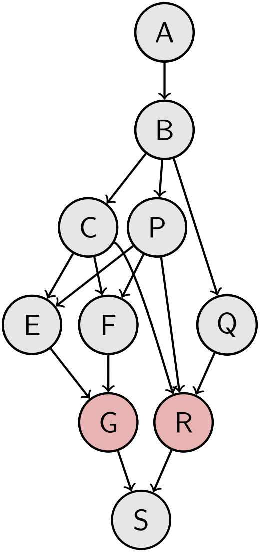
How much memory is needed?

How much memory do we need to evaluate this DAG?



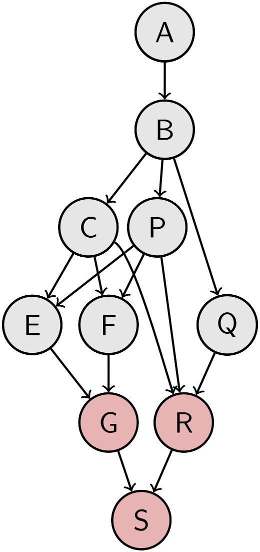
How much memory is needed?

How much memory do we need to evaluate this DAG?



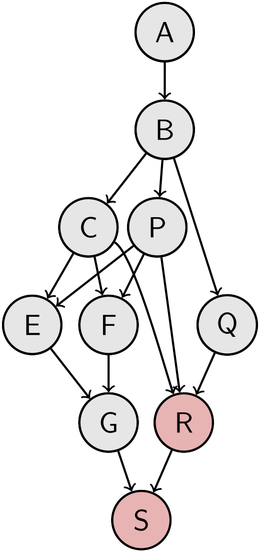
How much memory is needed?

How much memory do we need to evaluate this DAG?



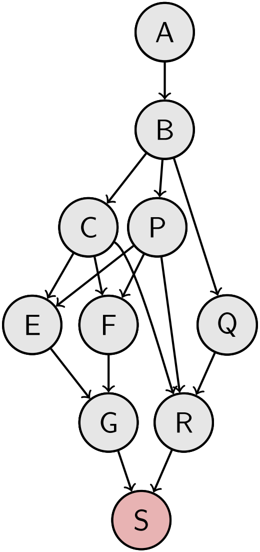
How much memory is needed?

How much memory do we need to evaluate this DAG?



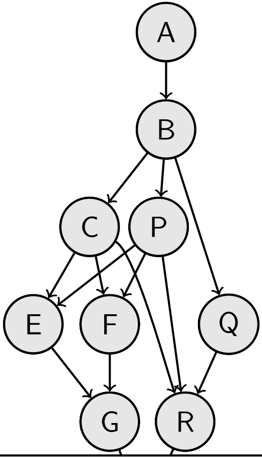
How much memory is needed?

How much memory do we need to evaluate this DAG?



How much memory is needed?

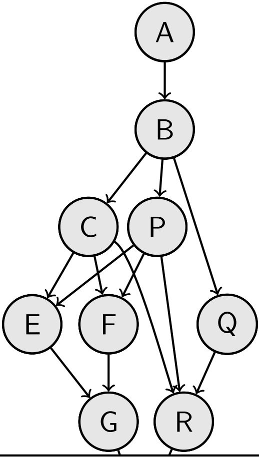
How much memory do we need to evaluate this DAG?



How many ‘memory cells’ needed?

How much memory is needed?

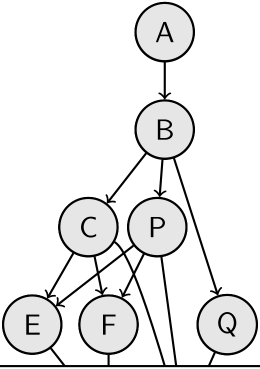
How much memory do we need to evaluate this DAG?



How many 'memory cells' needed? **6**

How much memory is needed?

How much memory do we need to evaluate this DAG?

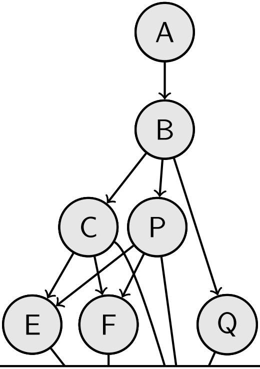


How many 'memory cells' needed? **6**

What if nodes were repeatable?

How much memory is needed?

How much memory do we need to evaluate this DAG?

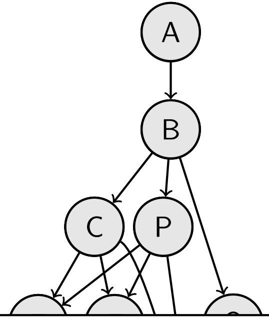


How many 'memory cells' needed? **6**

~~What if nodes were repeatable?~~
(Interesting, but not now.)

How much memory is needed?

How much memory do we need to evaluate this DAG?



How many 'memory cells' needed? **6**

~~What if nodes were repeatable?~~
(Interesting, but not now.)

What if we only had 4 cells *near* the processor?

Modeling local/*close* memory

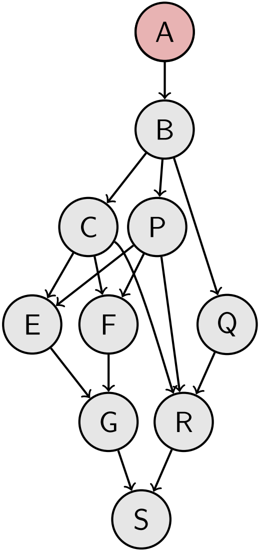
Rules, each with unit cost:

Compute	If all inputs of a pebble are red, color the pebble red.
Delete	Remove a pebble from the board.
Evict	Turn a red pebble into a blue pebble.
Bring close	Turn a blue pebble into a red pebble.

Hong & Kung '81

Turning Non-local Storage into Cost

How long does it take to evaluate this DAG with only 4 'red pebbles' ('close' memory cells)?



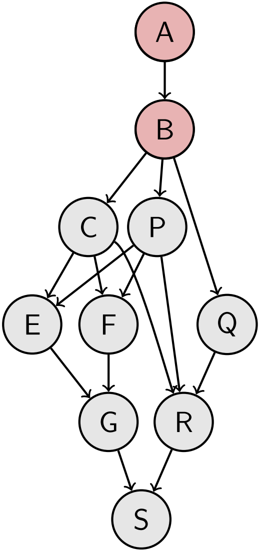
1

Hong & Kung '81

Turning Non-local Storage into Cost

How long does it take to evaluate this DAG with only 4 'red pebbles' ('close' memory cells)?

2

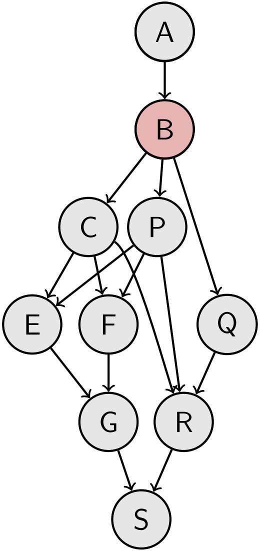


Hong & Kung '81

Turning Non-local Storage into Cost

How long does it take to evaluate this DAG with only 4 'red pebbles' ('close' memory cells)?

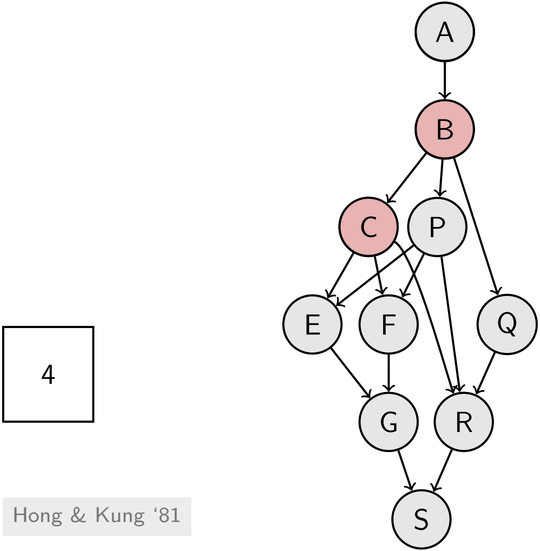
3



Hong & Kung '81

Turning Non-local Storage into Cost

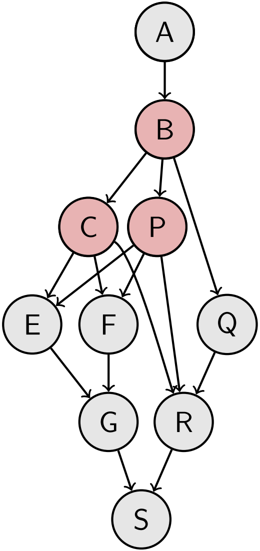
How long does it take to evaluate this DAG with only 4 'red pebbles' ('close' memory cells)?



Hong & Kung '81

Turning Non-local Storage into Cost

How long does it take to evaluate this DAG with only 4 'red pebbles' ('close' memory cells)?



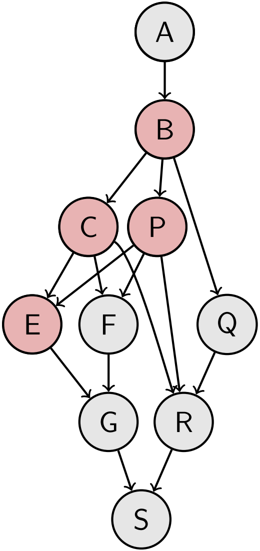
5

Hong & Kung '81

Turning Non-local Storage into Cost

How long does it take to evaluate this DAG with only 4 'red pebbles' ('close' memory cells)?

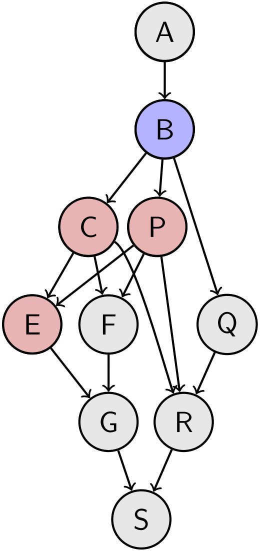
6



Hong & Kung '81

Turning Non-local Storage into Cost

How long does it take to evaluate this DAG with only 4 'red pebbles' ('close' memory cells)?

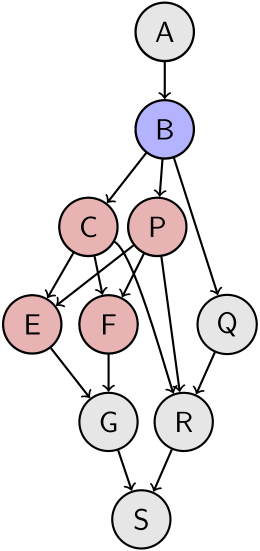


7

Hong & Kung '81

Turning Non-local Storage into Cost

How long does it take to evaluate this DAG with only 4 ‘red pebbles’ (‘close’ memory cells)?



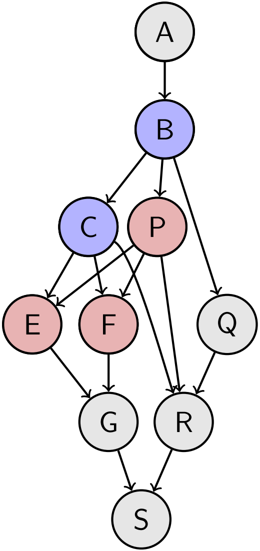
8

Hong & Kung ‘81

Turning Non-local Storage into Cost

How long does it take to evaluate this DAG with only 4 ‘red pebbles’ (‘close’ memory cells)?

9

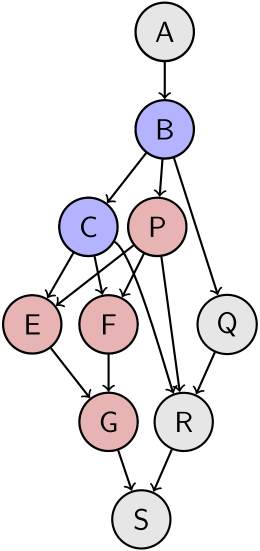


Hong & Kung ‘81

Turning Non-local Storage into Cost

How long does it take to evaluate this DAG with only 4 'red pebbles' ('close' memory cells)?

10

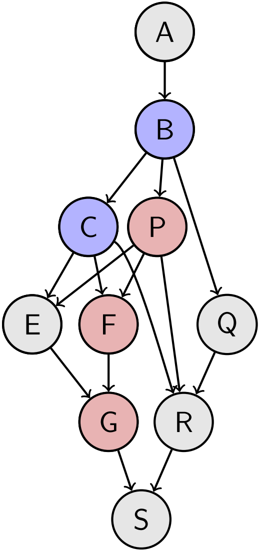


Hong & Kung '81

Turning Non-local Storage into Cost

How long does it take to evaluate this DAG with only 4 ‘red pebbles’ (‘close’ memory cells)?

11

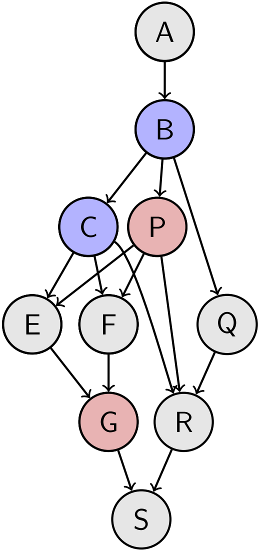


Hong & Kung ‘81

Turning Non-local Storage into Cost

How long does it take to evaluate this DAG with only 4 'red pebbles' ('close' memory cells)?

12

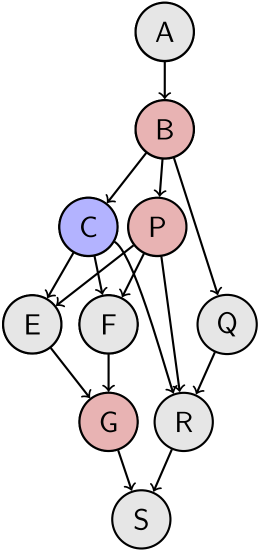


Hong & Kung '81

Turning Non-local Storage into Cost

How long does it take to evaluate this DAG with only 4 'red pebbles' ('close' memory cells)?

13

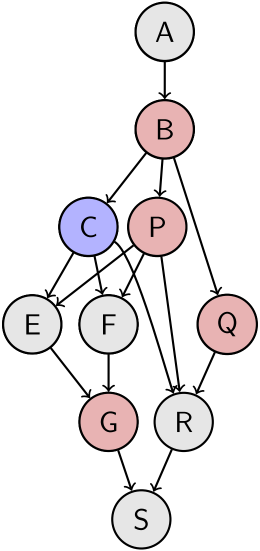


Hong & Kung '81

Turning Non-local Storage into Cost

How long does it take to evaluate this DAG with only 4 ‘red pebbles’ (‘close’ memory cells)?

14

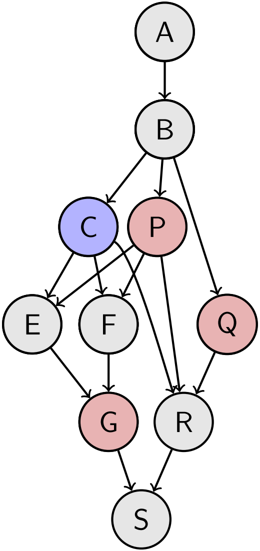


Hong & Kung ‘81

Turning Non-local Storage into Cost

How long does it take to evaluate this DAG with only 4 'red pebbles' ('close' memory cells)?

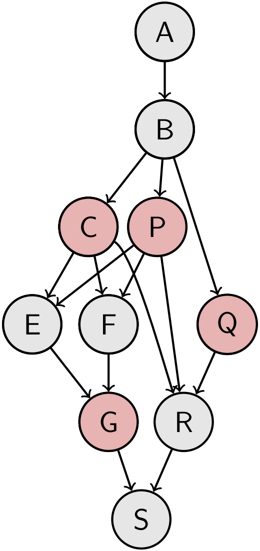
15



Hong & Kung '81

Turning Non-local Storage into Cost

How long does it take to evaluate this DAG with only 4 'red pebbles' ('close' memory cells)?

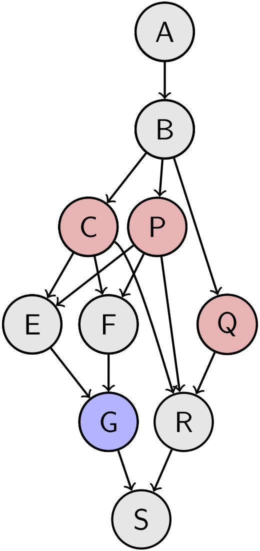


16

Hong & Kung '81

Turning Non-local Storage into Cost

How long does it take to evaluate this DAG with only 4 'red pebbles' ('close' memory cells)?



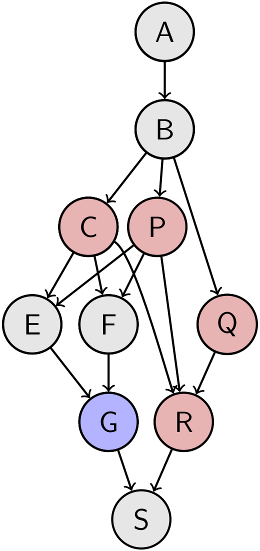
17

Hong & Kung '81

Turning Non-local Storage into Cost

How long does it take to evaluate this DAG with only 4 ‘red pebbles’ (‘close’ memory cells)?

18

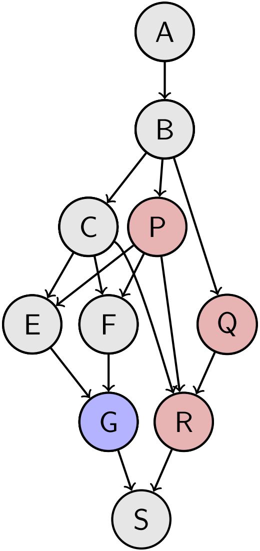


Hong & Kung ‘81

Turning Non-local Storage into Cost

How long does it take to evaluate this DAG with only 4 'red pebbles' ('close' memory cells)?

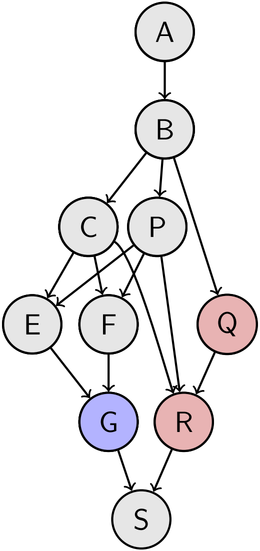
19



Hong & Kung '81

Turning Non-local Storage into Cost

How long does it take to evaluate this DAG with only 4 ‘red pebbles’ (‘close’ memory cells)?



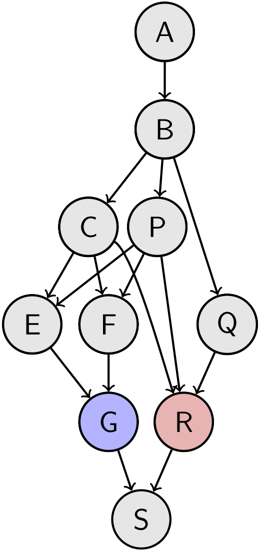
20

Hong & Kung ‘81

Turning Non-local Storage into Cost

How long does it take to evaluate this DAG with only 4 'red pebbles' ('close' memory cells)?

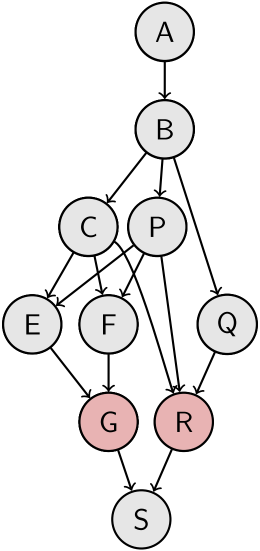
21



Hong & Kung '81

Turning Non-local Storage into Cost

How long does it take to evaluate this DAG with only 4 'red pebbles' ('close' memory cells)?

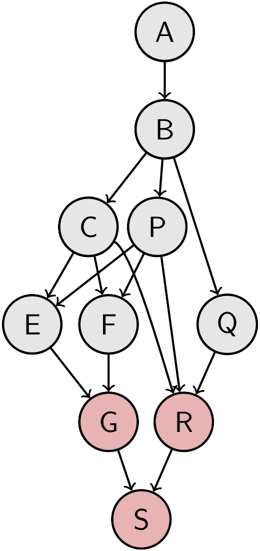


22

Hong & Kung '81

Turning Non-local Storage into Cost

How long does it take to evaluate this DAG with only 4 'red pebbles' ('close' memory cells)?



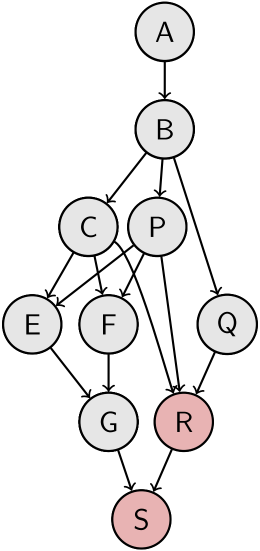
23

Hong & Kung '81

Turning Non-local Storage into Cost

How long does it take to evaluate this DAG with only 4 ‘red pebbles’ (‘close’ memory cells)?

24

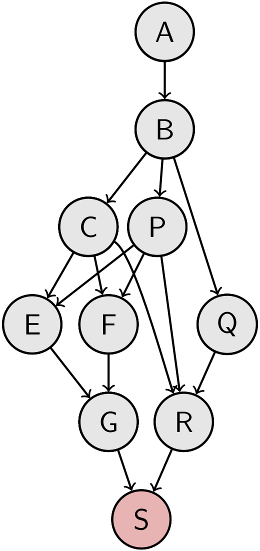


Hong & Kung ‘81

Turning Non-local Storage into Cost

How long does it take to evaluate this DAG with only 4 'red pebbles' ('close' memory cells)?

25



Hong & Kung '81

Red/Blue Pebbles: Theory

Examples of theoretical results from [Hong, Kung '81]:

Matrix-vector Multiplication:

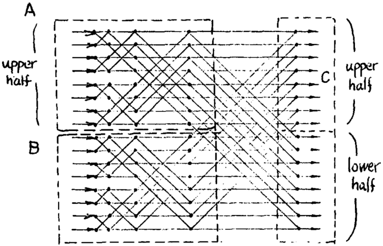
$$\text{Min I/O time} \sim \frac{n^2}{\#\text{ close cells}}$$

Matrix-matrix Multiplication:

$$\text{Min I/O time} \sim \frac{n^3}{\sqrt{\#\text{ close cells}}}$$

Red/Blue Pebbles: Theory

Fast Fourier Transform:



$$\text{Min I/O time} \sim \frac{n \log n}{\log \# \text{ close cells}}$$

Outline

Taking a step back

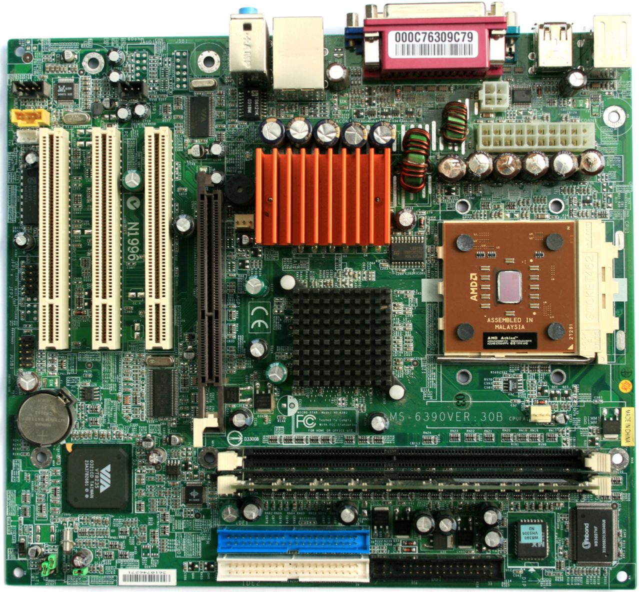
Want to answer:

How fast does a computer execute my code?

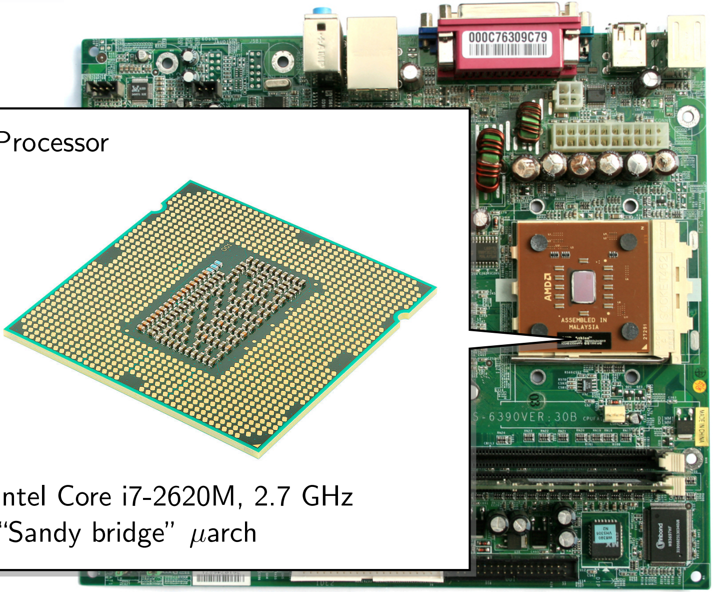
Need to answer first:

How does a computer execute my code?

What's in a computer?



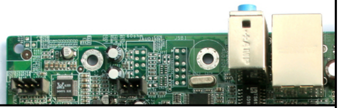
What's in a computer?



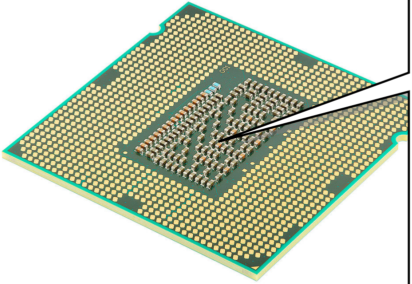
Processor

Intel Core i7-2620M, 2.7 GHz
"Sandy bridge" μarch

What's in a

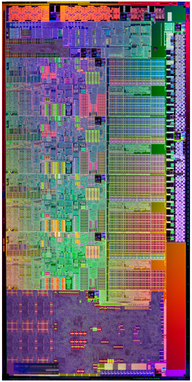


Processor



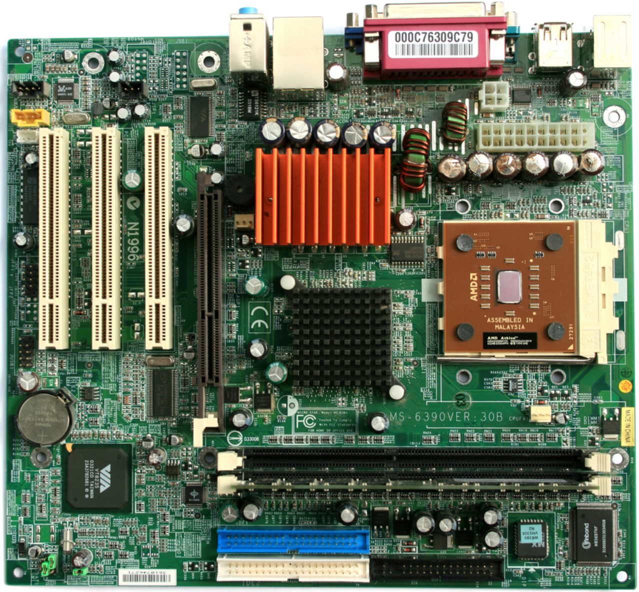
Intel Core i7-2620M, 2.7 GHz
"Sandy bridge" μarch

Die

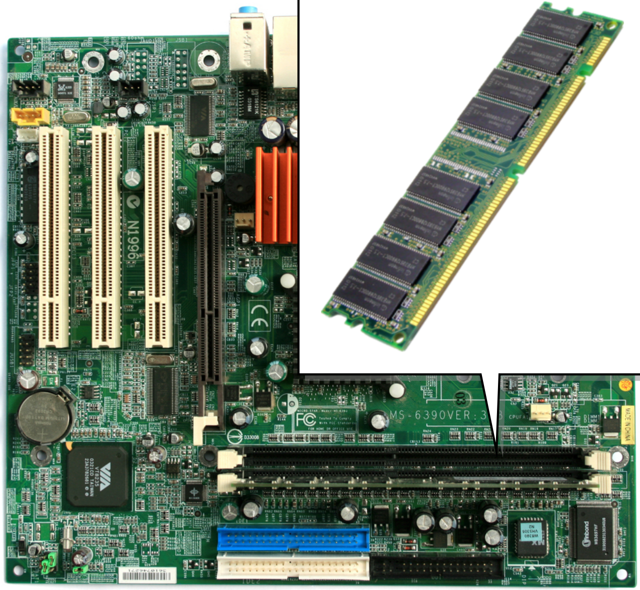


149 mm^2, 2 real cores
624,000,000 transistors
$\sim$ 35W

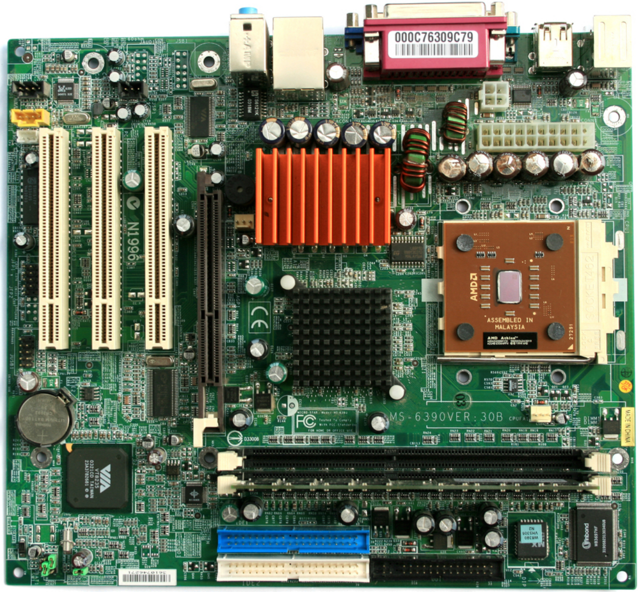
What's in a computer?



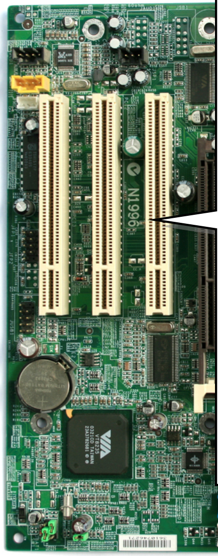
What's in a computer?



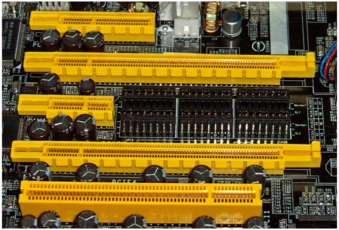
What's in a computer?



What's in a computer?



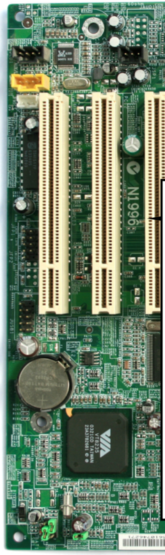
Expansion Slots



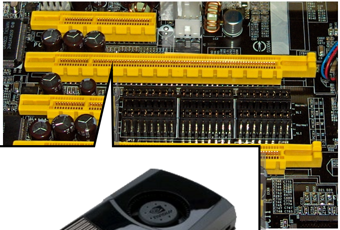
PCI-Express (x4, x16, x1, x16) and regular PCI

PCIe V2, x16 Bandwidth: $\sim$ 6 GB/s

What's in a computer?



Expansion Slots

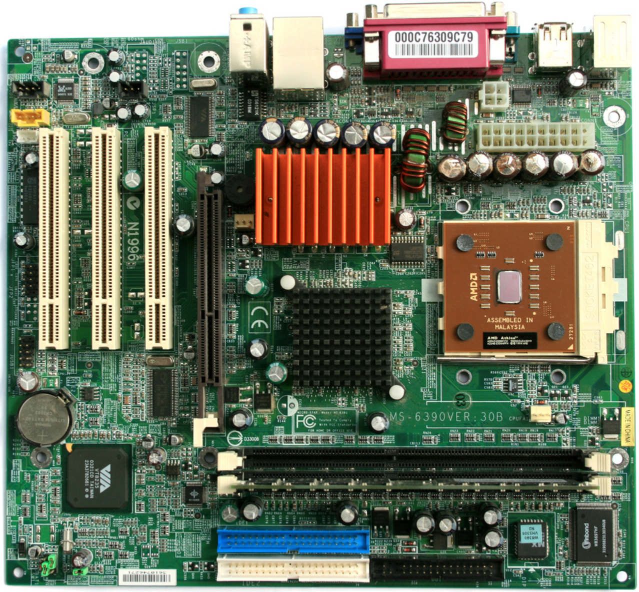


, x1, x16)

idth:

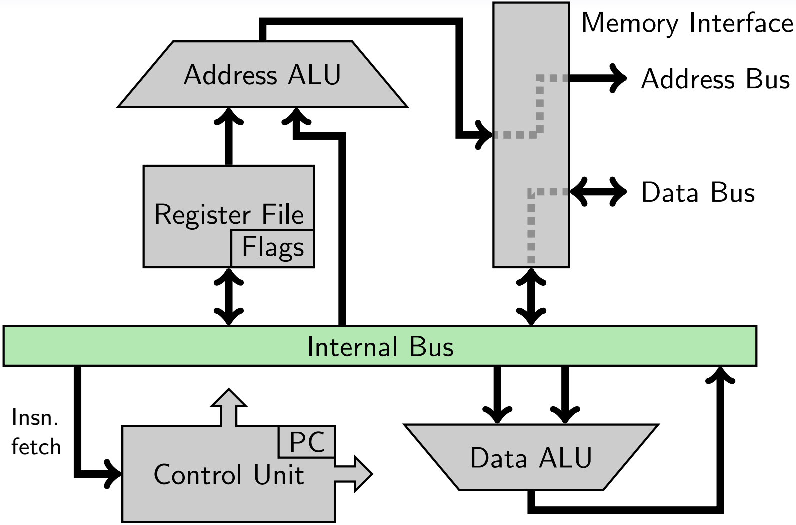
GPU goes here

What's in a computer?



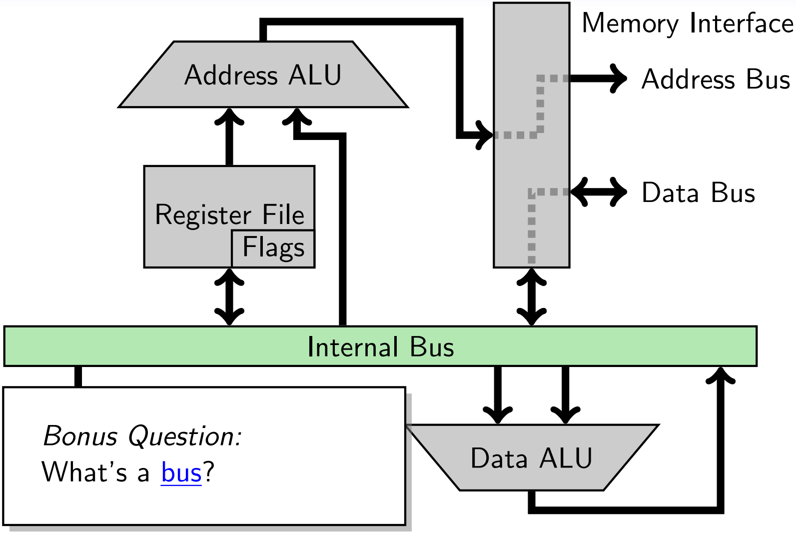
Outline

A Basic Processor



(loosely based on Intel 8086)

A Basic Processor



Bonus Question:
What's a bus?

(loosely based on Intel 8086)

How all of this fits together

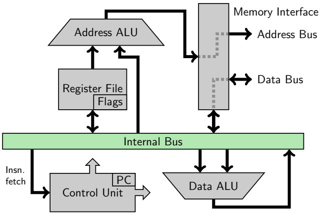
Everything synchronizes to the *Clock*.

Control Unit ("CU"): The brains of the operation. Everything connects to it.

Bus entries/exits are *gated* and (potentially) *buffered*.

CU controls gates, tells other units about 'what' and 'how':

- What operation?
- Which register?
- Which addressing mode?



What is... an ALU?

Arithmetic **L**ogic **U**nit

One or two operands A, B

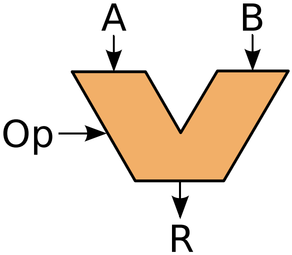
Operation selector (Op):

- (Integer) Addition, Subtraction
- (Logical) And, Or, Not
- (Bitwise) Shifts (equivalent to multiplication by power of two)
- (Integer) Multiplication, Division

Specialized ALUs:

- Floating Point Unit (FPU)
- Address ALU

Operates on binary representations of numbers. Negative numbers represented by two's complement.



What is. . . a Register File?

Registers are *On-Chip Memory*

- Directly usable as operands in Machine Language
- Often "general-purpose"
- Sometimes special-purpose: Floating point, Indexing, Accumulator
- Small: x86_64: 16×64 bit GPRs
- Very fast (near-zero latency)

%r0
%r1
%r2
%r3
%r4
%r5
%r6
%r7

What is. . . a Register File?

Registers are *On-Chip Memory*

- Directly usable as operands in Machine Language
- Often "general-purpose"
- Sometimes special-purpose: Floating point, Indexing, Accumulator
- Small: x86_64: 16×64 bit GPRs
- Very fast (near-zero latency)

%r0
%r1
%r2
%r3
%r4
%r5
%r6
%r7

First red/blue pebble game, played by compiler.

Outline

A Very Simple Program

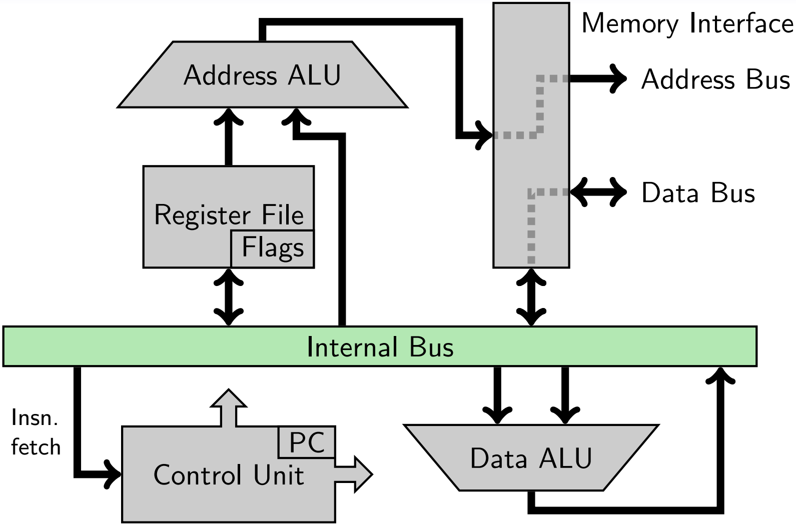
```
int a = 5;
int b = 17;
int z = a * b;
```

4:	c7 45 f4 05 00 00 00	movl	$0x5,−0xc(%rbp)
b:	c7 45 f8 11 00 00 00	movl	$0x11,−0x8(%rbp)
12:	8b 45 f4	mov	−0xc(%rbp),%eax
15:	0f af 45 f8	imul	−0x8(%rbp),%eax
19:	89 45 fc	mov	%eax,−0x4(%rbp)
1c:	8b 45 fc	mov	−0x4(%rbp),%eax

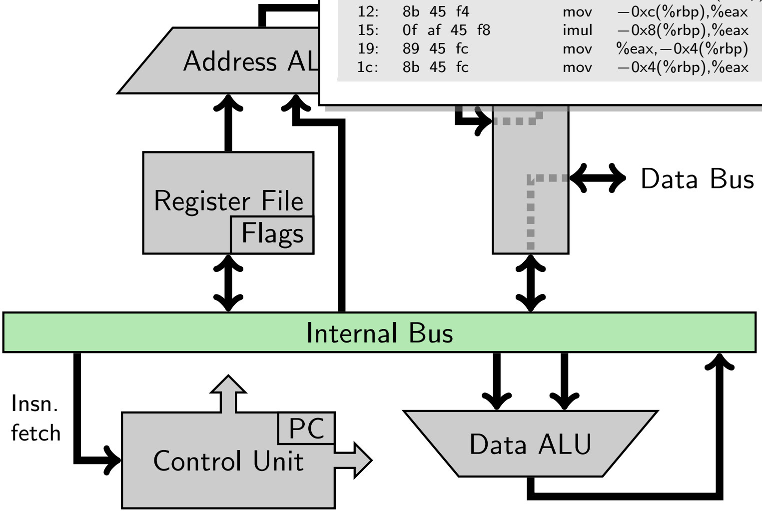
Things to know:

- Addressing modes (Immediate, Register, Base plus Offset)
- 0xHexadecimal
- "AT&T Form": (we'll use this)
  `<opcode><size> <source>, <dest>`

Another Look



```
 4:  c7 45 f4 05 00 00 00  movl  $0x5,-0xc(%rbp)
 b:  c7 45 f8 11 00 00 00  movl  $0x11,-0x8(%rbp)
12:  8b 45 f4              mov   -0xc(%rbp),%eax
15:  0f af 45 f8           imul  -0x8(%rbp),%eax
19:  89 45 fc              mov   %eax,-0x4(%rbp)
1c:  8b 45 fc              mov   -0x4(%rbp),%eax
```



A Very Simple Program: Intel Form

4:	c7 45 f4 05 00 00 00	mov	DWORD PTR [rbp−0xc],0x5
b:	c7 45 f8 11 00 00 00	mov	DWORD PTR [rbp−0x8],0x11
12:	8b 45 f4	mov	eax, DWORD PTR [rbp−0xc]
15:	0f af 45 f8	imul	eax, DWORD PTR [rbp−0x8]
19:	89 45 fc	mov	DWORD PTR [rbp−0x4],eax
1c:	8b 45 fc	mov	eax, DWORD PTR [rbp−0x4]

- "Intel Form": (you might see this on the net)
  `<opcode> <sized dest>, <sized source>`
- Goal: Reading comprehension.
- Don't understand an opcode?
  Google "`<opcode> intel instruction`".

Questions?

?

Image Credits

- Valgrind logo: Julian Seward
- Mainboard: Wikimedia Commons ©
- PCI Express slots: Wikimedia Commons ©
- DIMM: sxc.hu/gobran11
- Sandy bridge bottom: Wikimedia Commons ©
- Sandy bridge die: Intel Corp.
- GTX 480: Nvidia Corp. ©