Scientific Computing III. High Performance Scientific Computing

(Phys 2109/Ast 3100H)

Lecture 1: Intro to Parallel Programming

SciNet HPC Consortium, University of Toronto

Winter 2013

SciNet compute • calcul CANADA

Why Parallel Programming?

SciNet
compute • calcul
CANADA

Why Parallel Programming?

1. **Faster**
   There's a limit to how fast 1 computer can compute.

SciNet compute • calcul CANADA

Why Parallel Programming?

1. **Faster**
   There's a limit to how fast 1 computer can compute.
2. **Bigger**
   There's a limit to how much memory, disk, etc, can be put on 1 computer.

SciNet
compute • calcul
CANADA

Why Parallel Programming?

1. **Faster**
   There's a limit to how fast 1 computer can compute.
2. **Bigger**
   There's a limit to how much memory, disk, etc, can be put on 1 computer.
3. **More**
   Want to do the same thing that was done on 1 computer, but *thousands of times*.

SciNet
compute • calcul
CANADA

Why Parallel Programming?

1. **Faster**
   There's a limit to how fast 1 computer can compute.
2. **Bigger**
   There's a limit to how much memory, disk, etc, can be put on 1 computer.
3. **More**
   Want to do the same thing that was done on 1 computer, but *thousands of times*.

   So use more computers!

SciNet compute • calcul CANADA

Why is it necessary?

- Big Data: Modern experiments and observations yield vastly more data to be processsed than in the past.
- Big Science: As more computing resources become available (SciNet), the bar for cutting edge simulations is raised.
- New Science: which before could not even be done, but now becomes reachable.

SciNet
compute • calcul
CANADA

Why is it necessary?

- Big Data: Modern experiments and observations yield vastly more data to be processsed than in the past.
- Big Science: As more computing resources become available (SciNet), the bar for cutting edge simulations is raised.
- New Science: which before could not even be done, but now becomes reachable.

However:

SciNet compute • calcul CANADA

Why is it necessary?

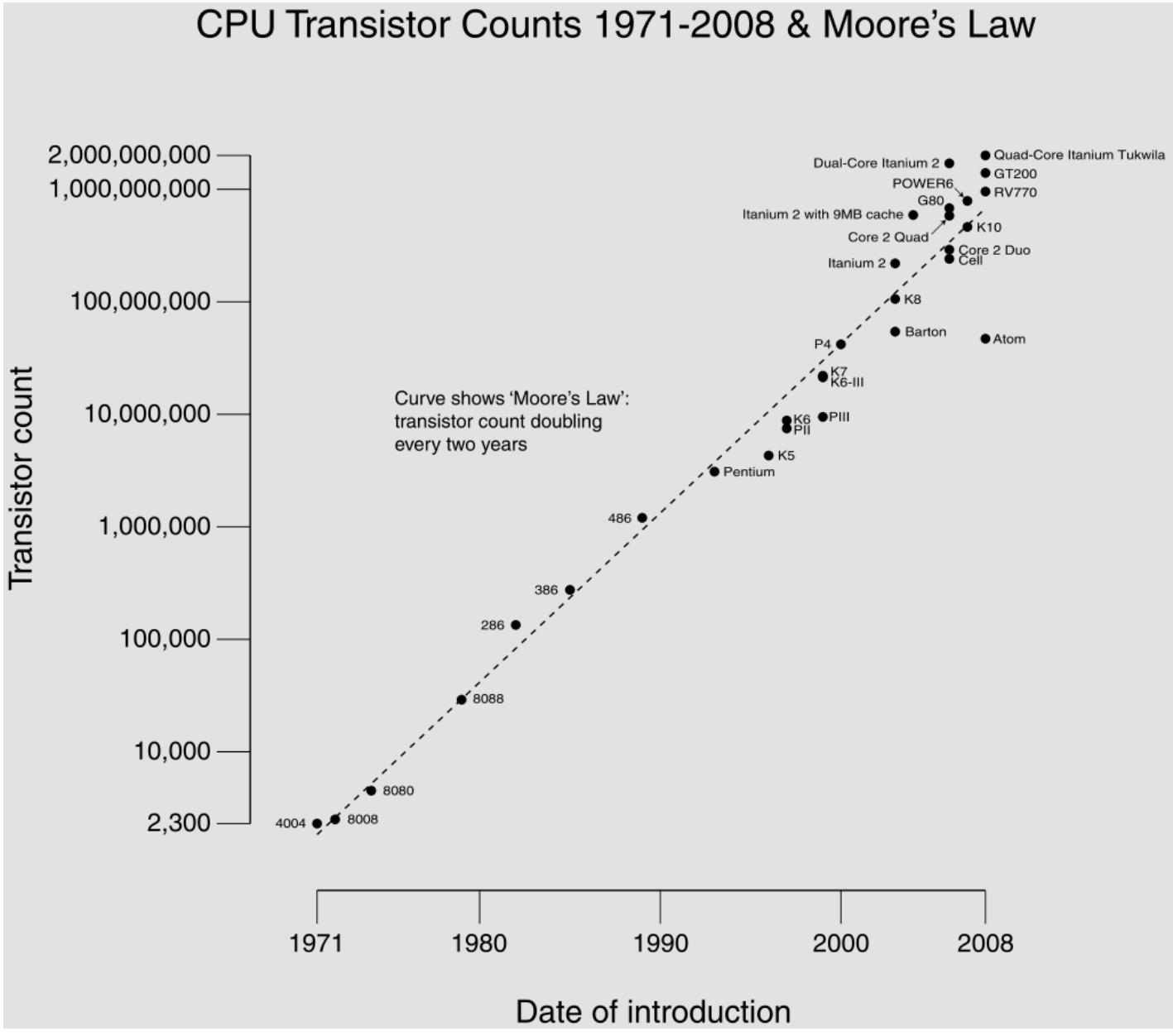
- Big Data: Modern experiments and observations yield vastly more data to be processsed than in the past.
- Big Science: As more computing resources become available (SciNet), the bar for cutting edge simulations is raised.
- New Science: which before could not even be done, but now becomes reachable.

However:

- Advances in clock speeds, bigger and faster memory and disks have been lagging as compared to e.g. 10 years ago.
  Can no longer "just wait a year" and get a better computer.
- So more computing resources here means: more cores running concurrently.
- *Even most laptops now have 2 or more cpus.*
- So parallel computing is necessary.

SciNet compute • calcul CANADA

Wait, what about Moore's Law?



(source: Transistor Count and Moore's Law - 2008.svg, by Wgsimon, wikipedia)

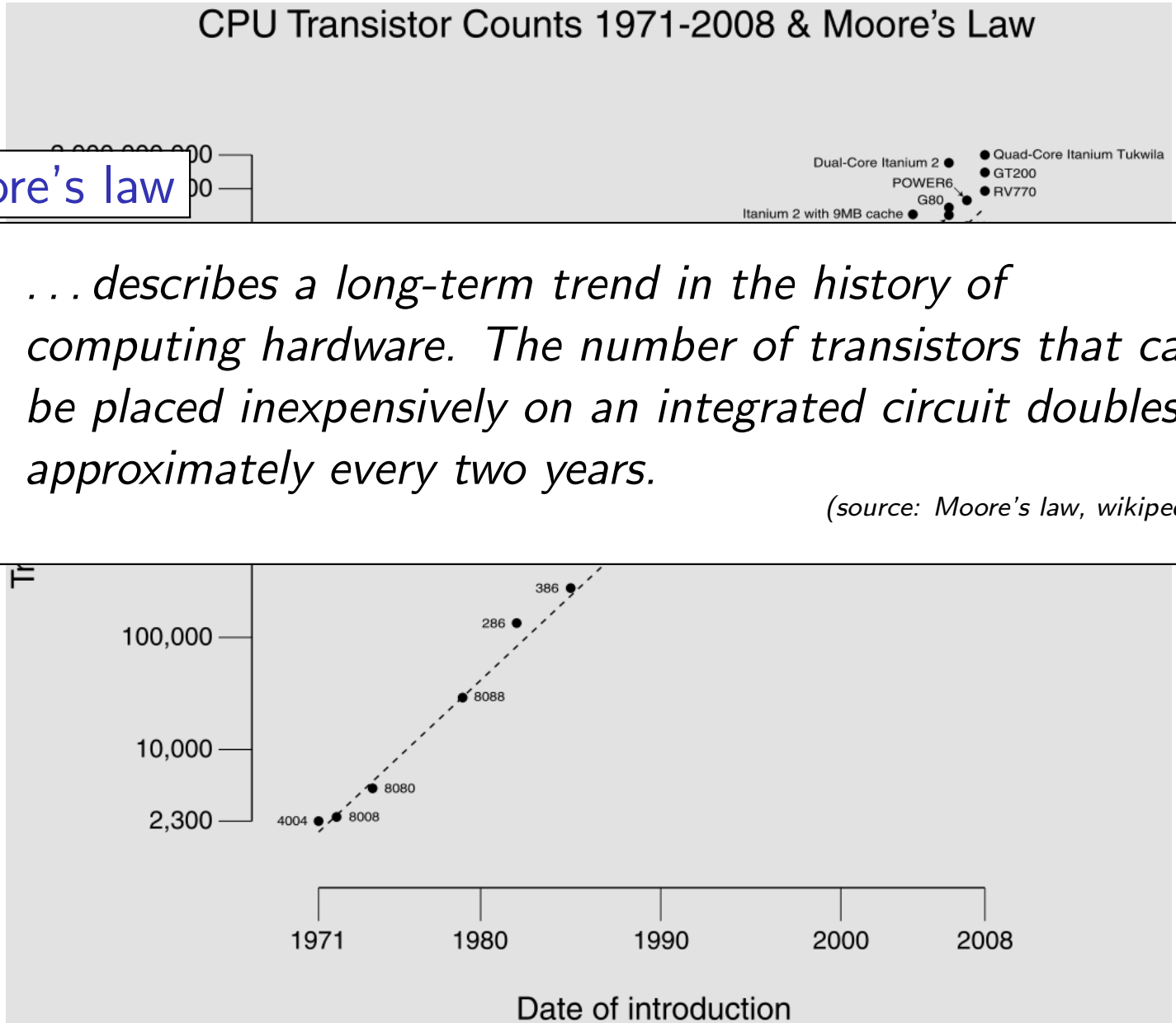
SciNet compute • calcul CANADA

Wait, what about Moore's Law?

Moore's law

. . . describes a long-term trend in the history of computing hardware. The number of transistors that can be placed inexpensively on an integrated circuit doubles approximately every two years.

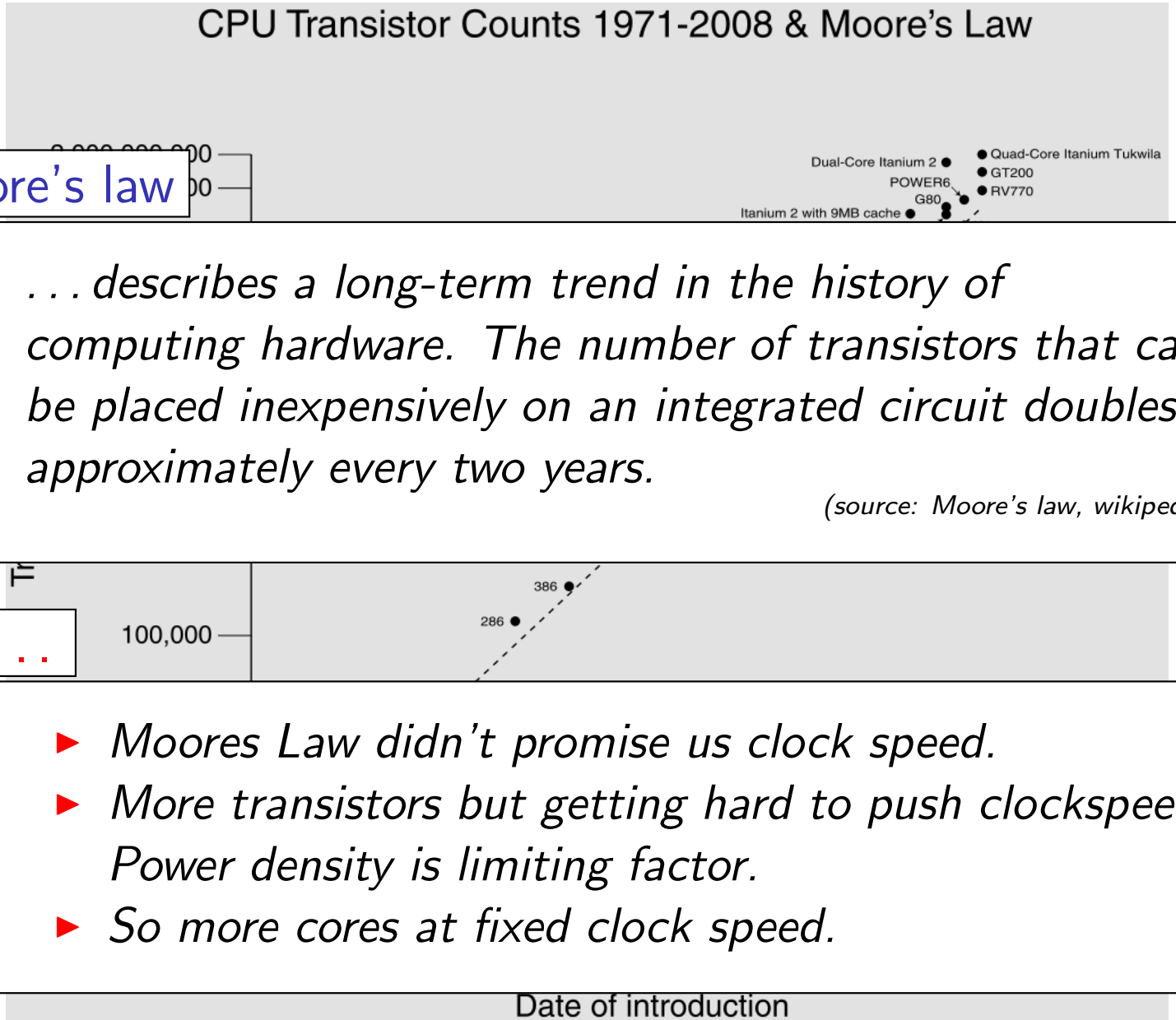
(source: Moore's law, wikipedia)



(source: Transistor Count and Moore's Law - 2008.svg, by Wgsimon, wikipedia)

SciNet compute • calcul CANADA

Wait, what about Moore's Law?



Moore's law

. . . describes a long-term trend in the history of computing hardware. The number of transistors that can be placed inexpensively on an integrated circuit doubles approximately every two years.

(source: Moore's law, wikipedia)

But. . .

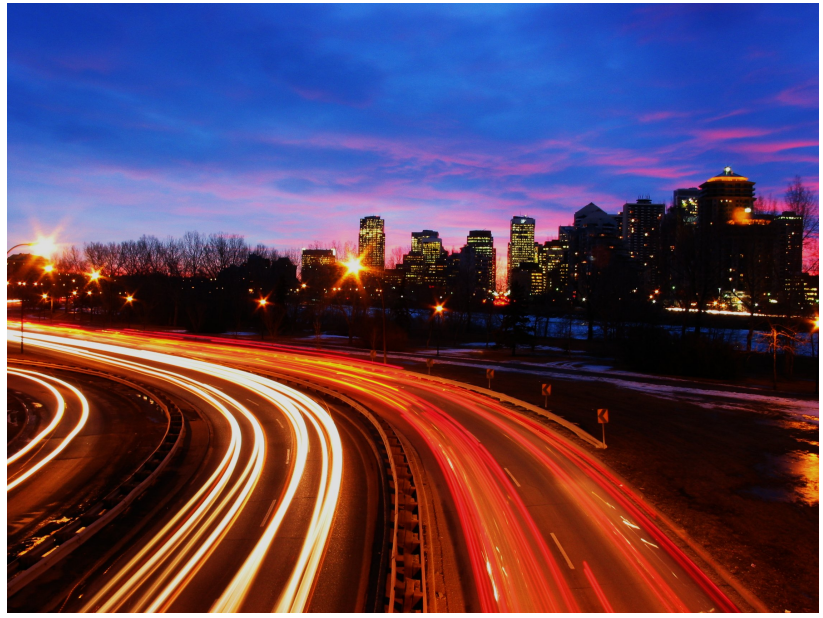
- *Moores Law didn't promise us clock speed.*
- *More transistors but getting hard to push clockspeed up. Power density is limiting factor.*
- *So more cores at fixed clock speed.*

(source: Transistor Count and Moore's Law - 2008.svg, by Wgsimon, wikipedia)

SciNet compute • calcul CANADA

Concurrency

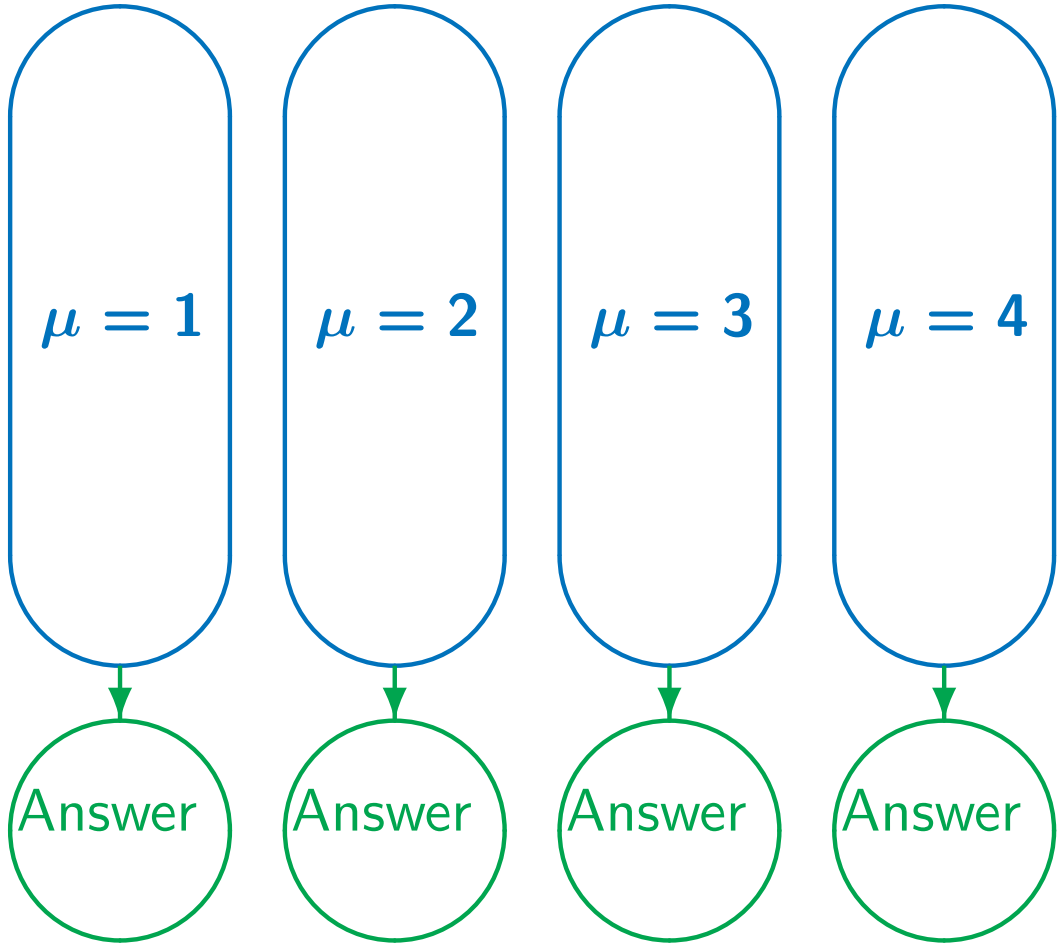
- Must have something to do for all these cores.
- Find parts of the program that can done independently, and therefore concurrently.
- There must be many such parts.
- There order of execution should not matter either.
- Data dependencies limit concurrency.



(source: http://flickr.com/photos/splorp)

Parameter study: best case scenario

- Aim is to get results from a model as a parameter varies.
- Can run the serial program on each processor at the same time.
- Get "more" done.

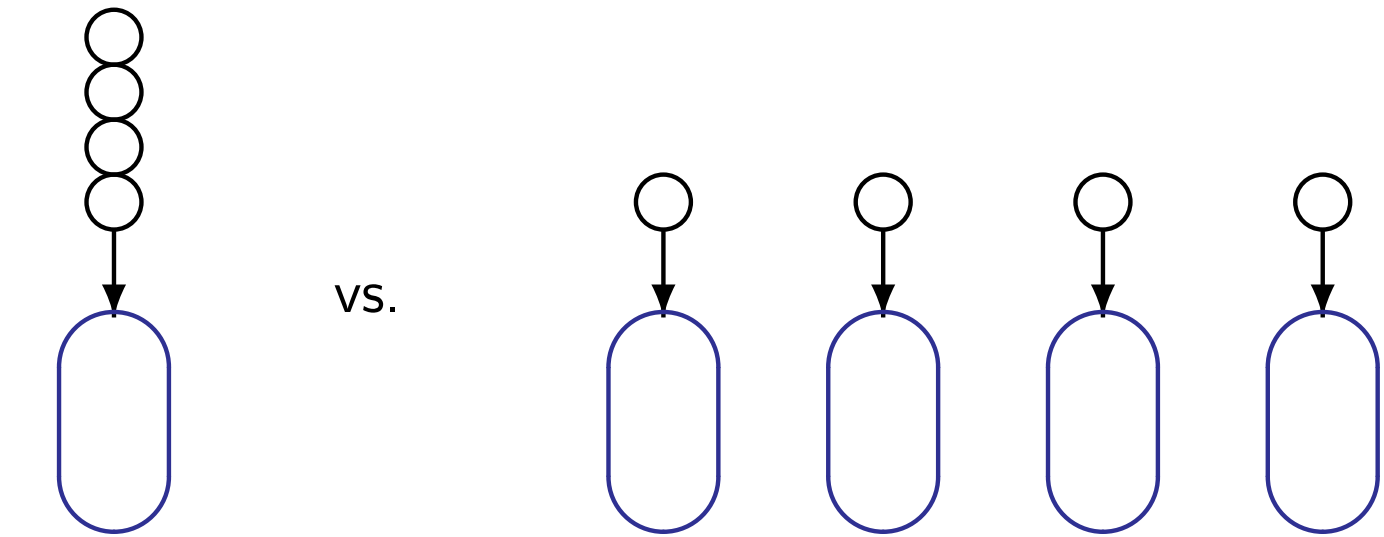


Throughput

- How many tasks can you do per time unit?

$$\text{throughput} = \mathbf{H} = \frac{\mathbf{N}}{\mathbf{T}}$$

- Maximizing **H** means that you can do as much as possible.
- Independent tasks: using **P** processors increases **H** by a factor **P**



$\mathbf{T = NT_1}$
$\mathbf{H = 1/T_1}$

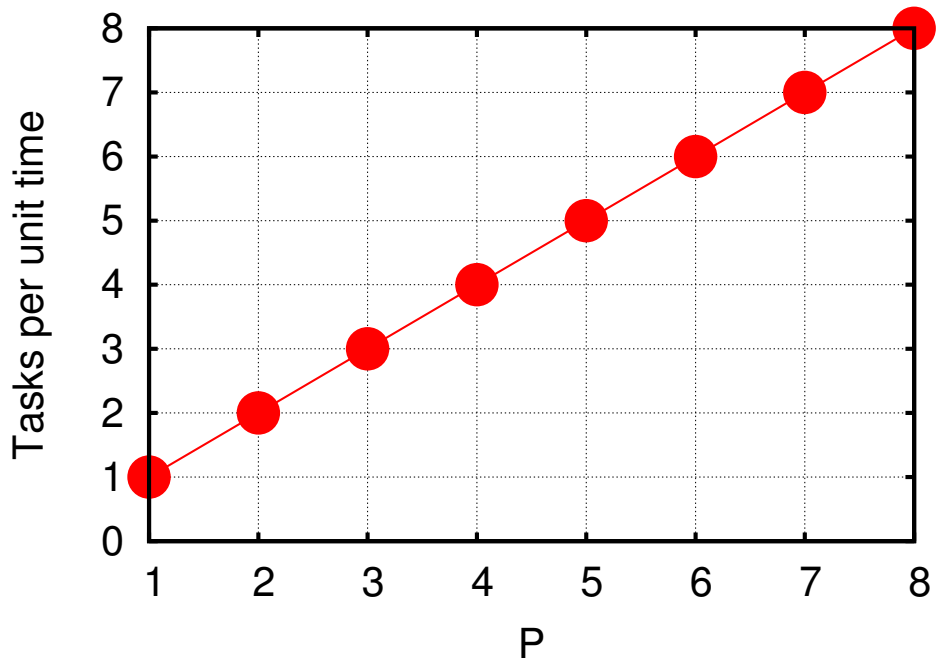
$\mathbf{T = NT_1/P}$
$\mathbf{H = P/T_1}$

Scaling — Throughput

- How a problem's throughput scales as processor number increases ("strong scaling").
- In this case, linear scaling:

$$\mathbf{H} \propto \mathbf{P}$$

- This is Perfect scaling.

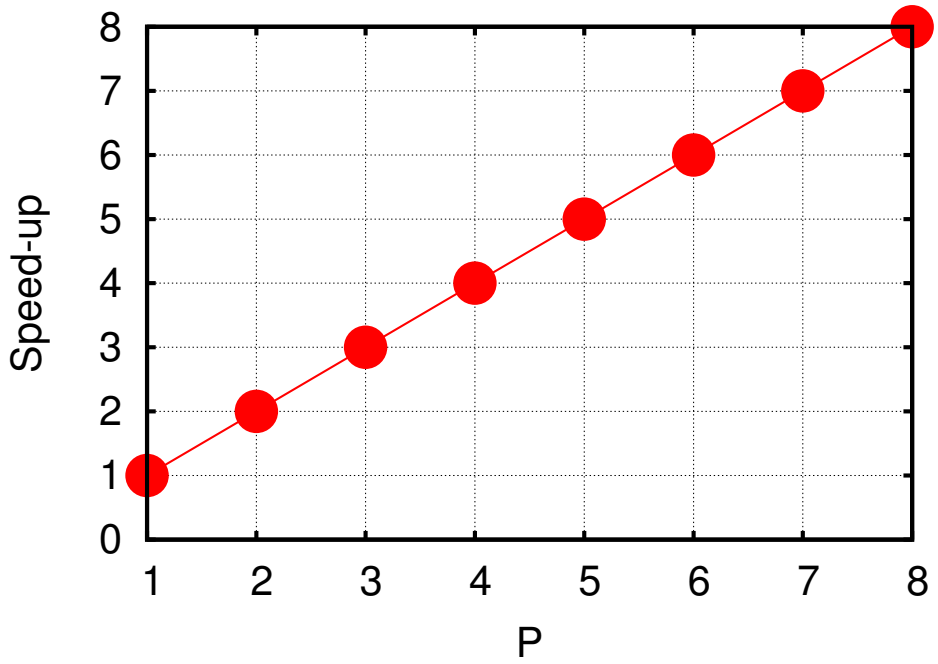


Scaling – Speedup

- How much faster the problem is solved as processor number increases.
- Measured by the serial time divided by the parallel time

$$\mathbf{S} = \frac{\mathbf{T_{serial}}}{\mathbf{T(P)}}$$

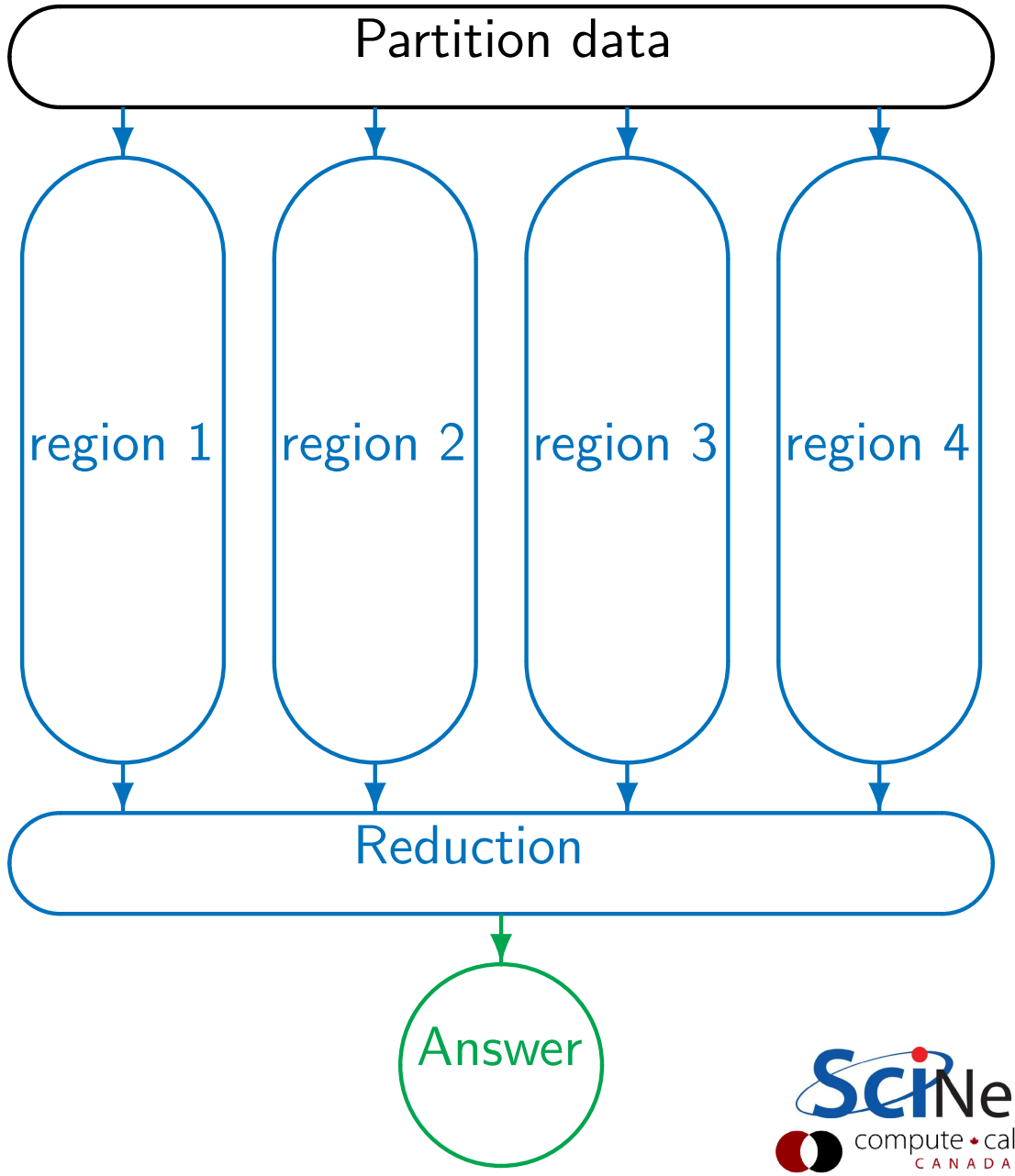
- For embarrasingly parallel applications, $\mathbf{S} \propto \mathbf{P}$: Linear speed up.



SciNet compute • calcul CANADA

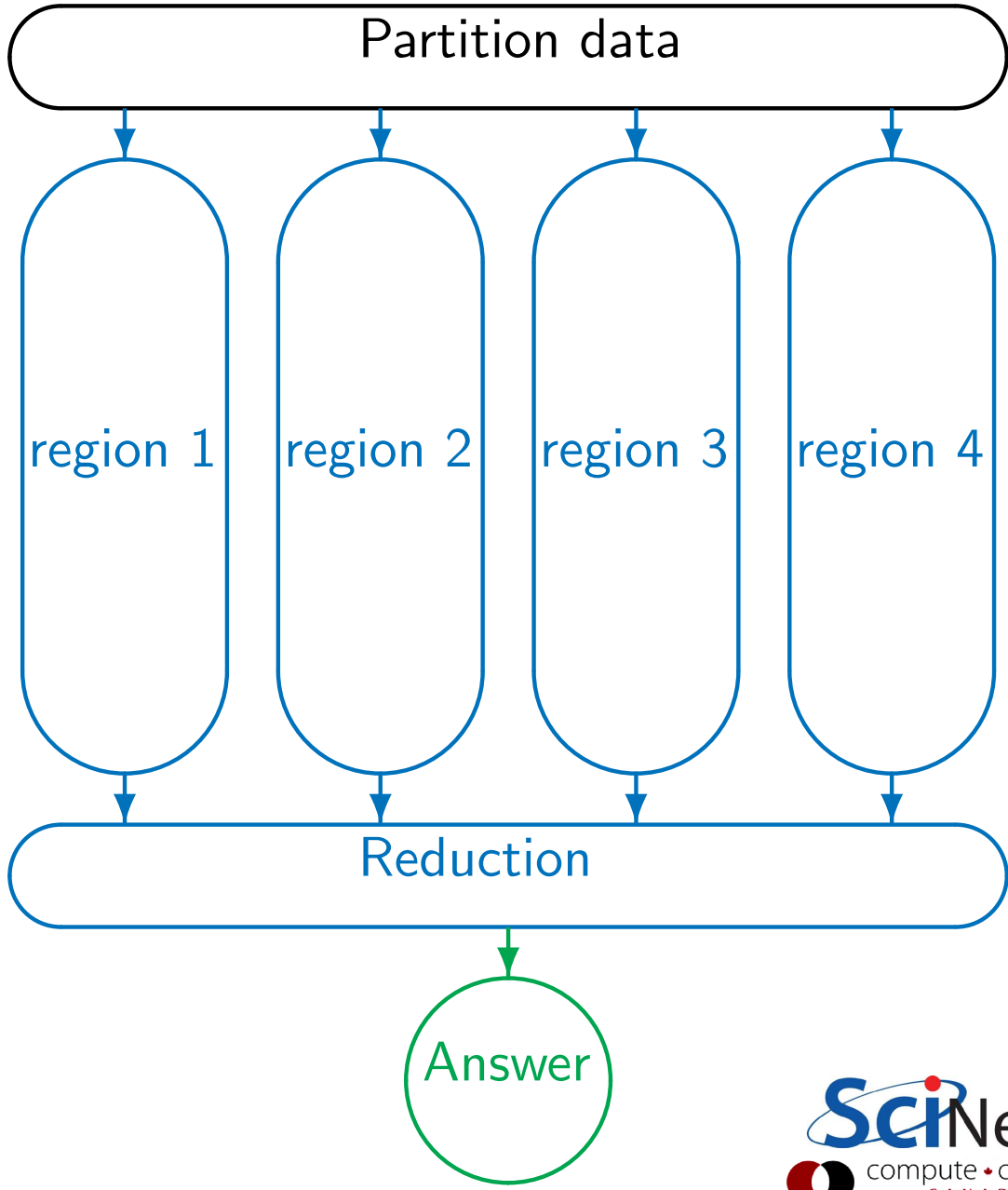
Non-ideal cases

- Say we want to integrate some tabulated experimental data.
- Integration can be split up, so different regions are summed by each processor.
- Non-ideal:
  - First need to get data to processor
  - And at the end bring together all the sums: "reduction"



Non-ideal cases

Parallel region $\Rightarrow$

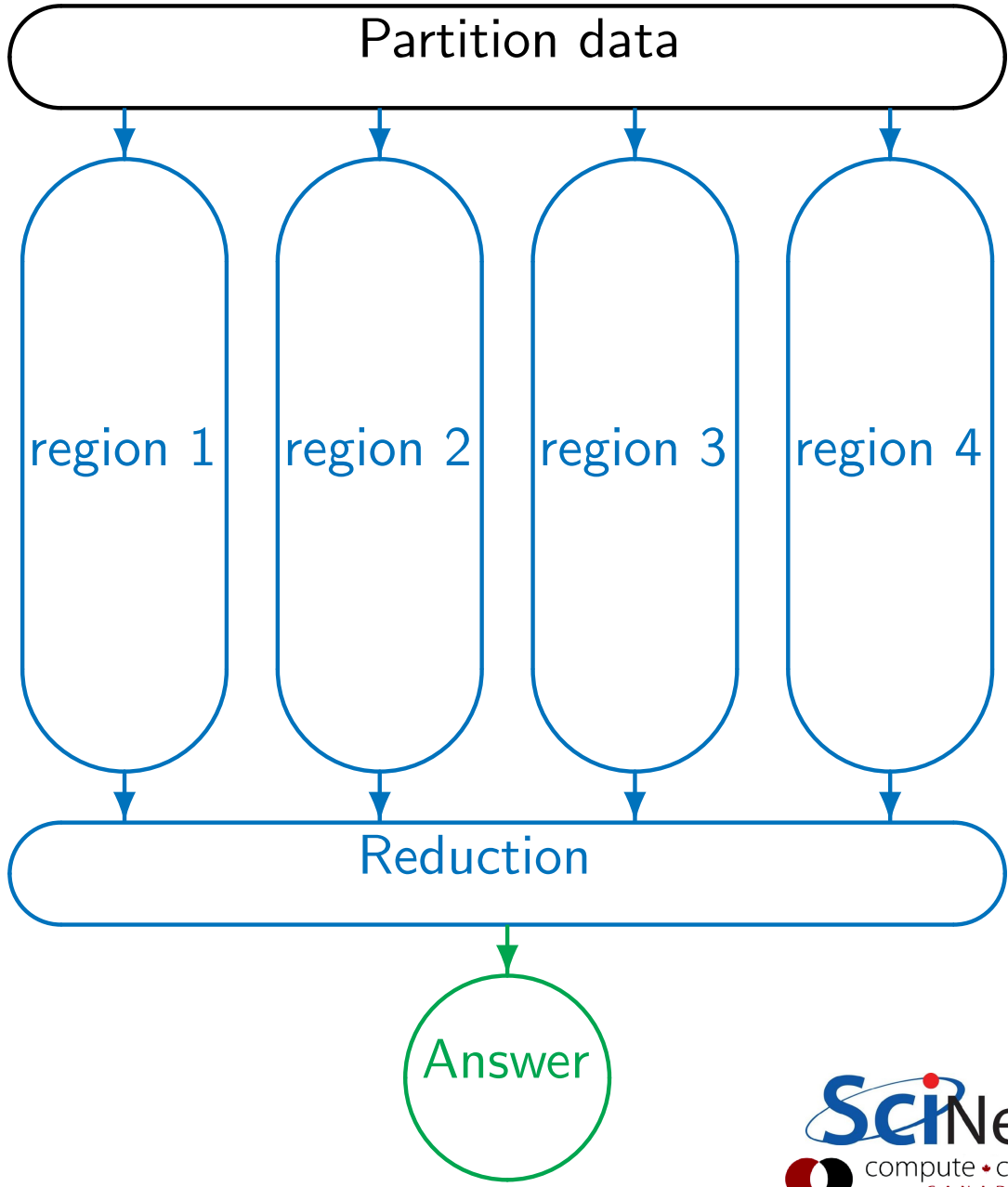


SciNet compute • calcul CANADA

Non-ideal cases

Parallel region $\Rightarrow$

Perfectly Parallel
(for large **N**)



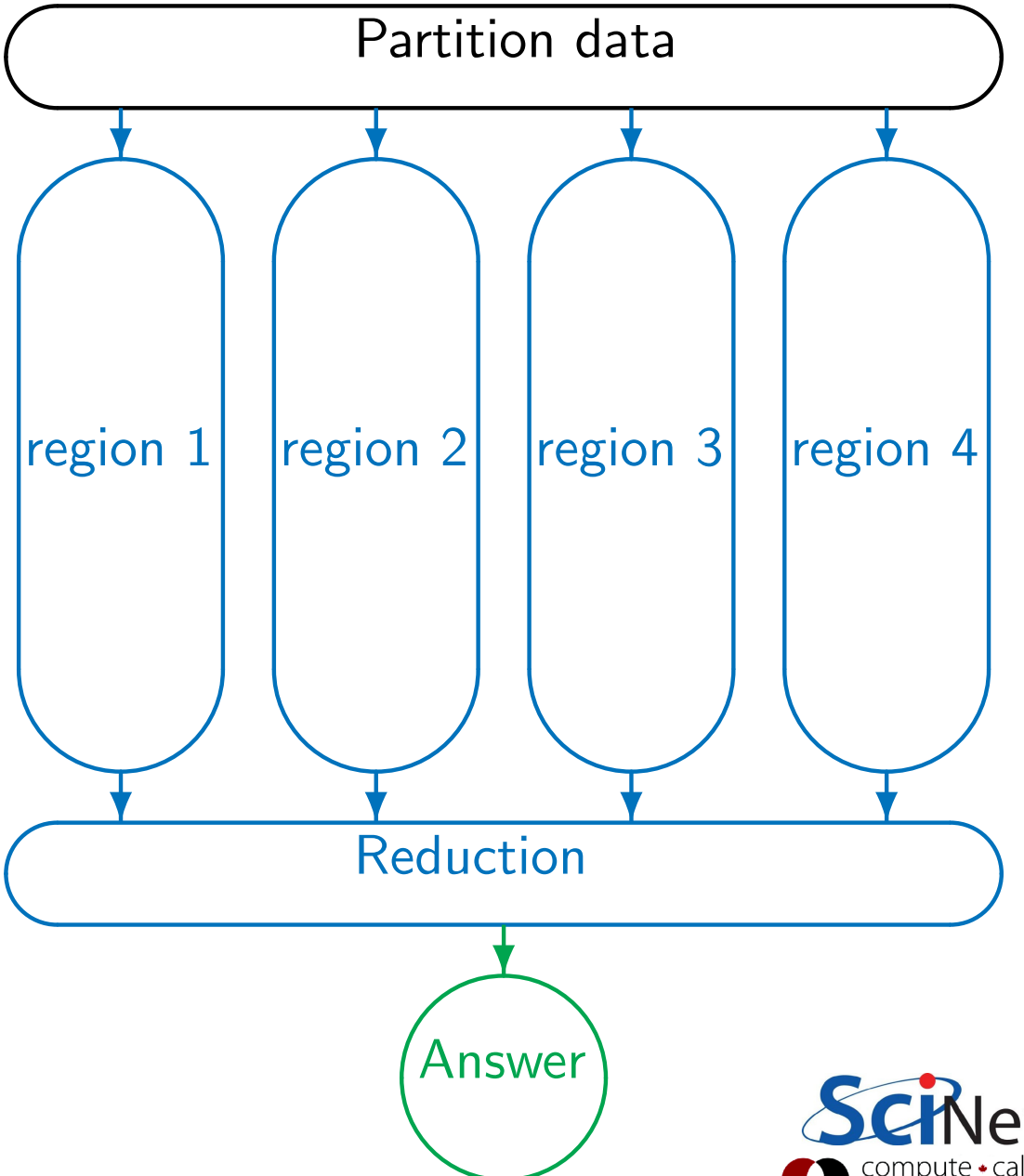
SciNet
compute • calcul
CANADA

Non-ideal cases

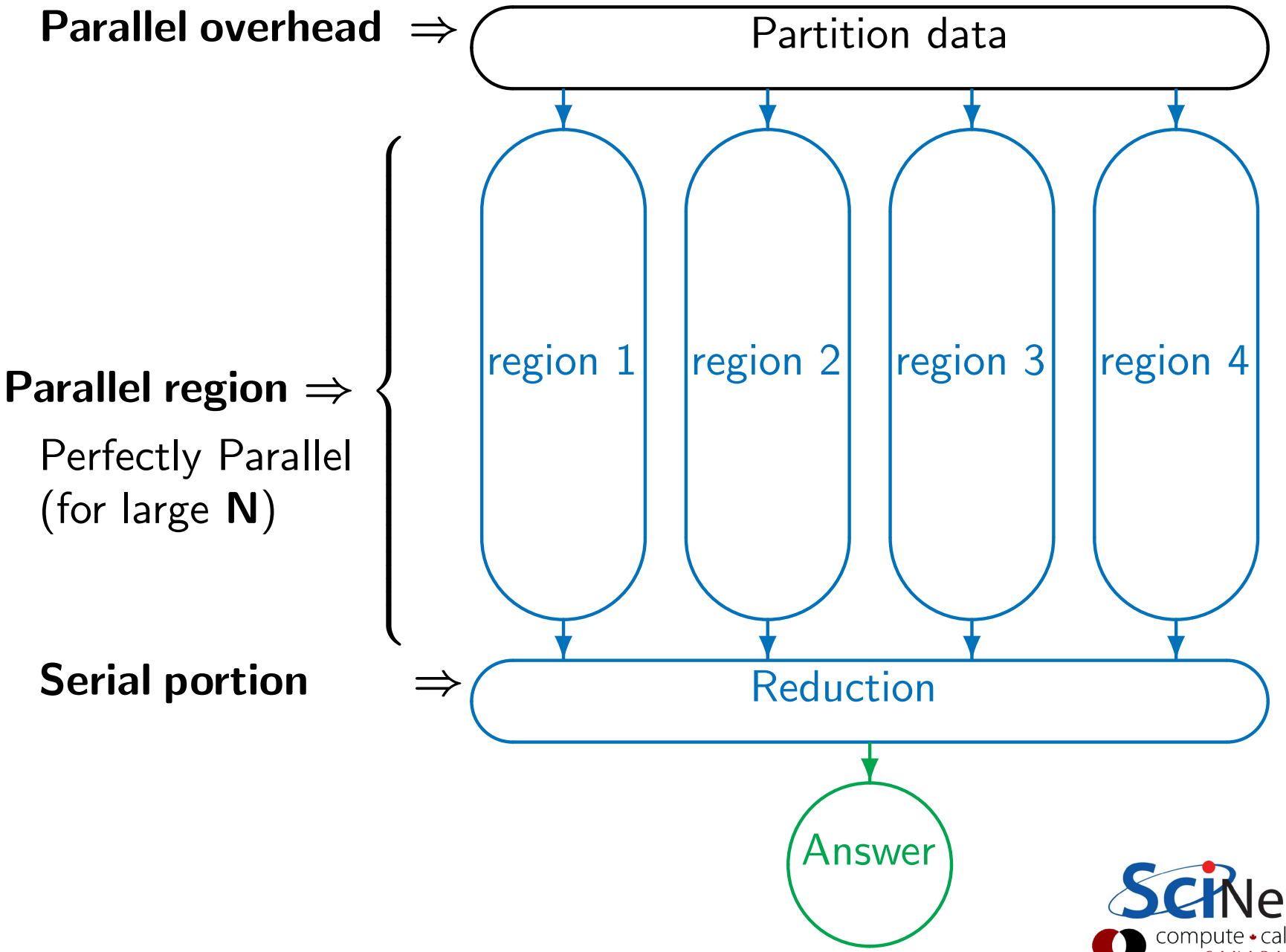
Parallel region $\Rightarrow$

Perfectly Parallel
(for large **N**)

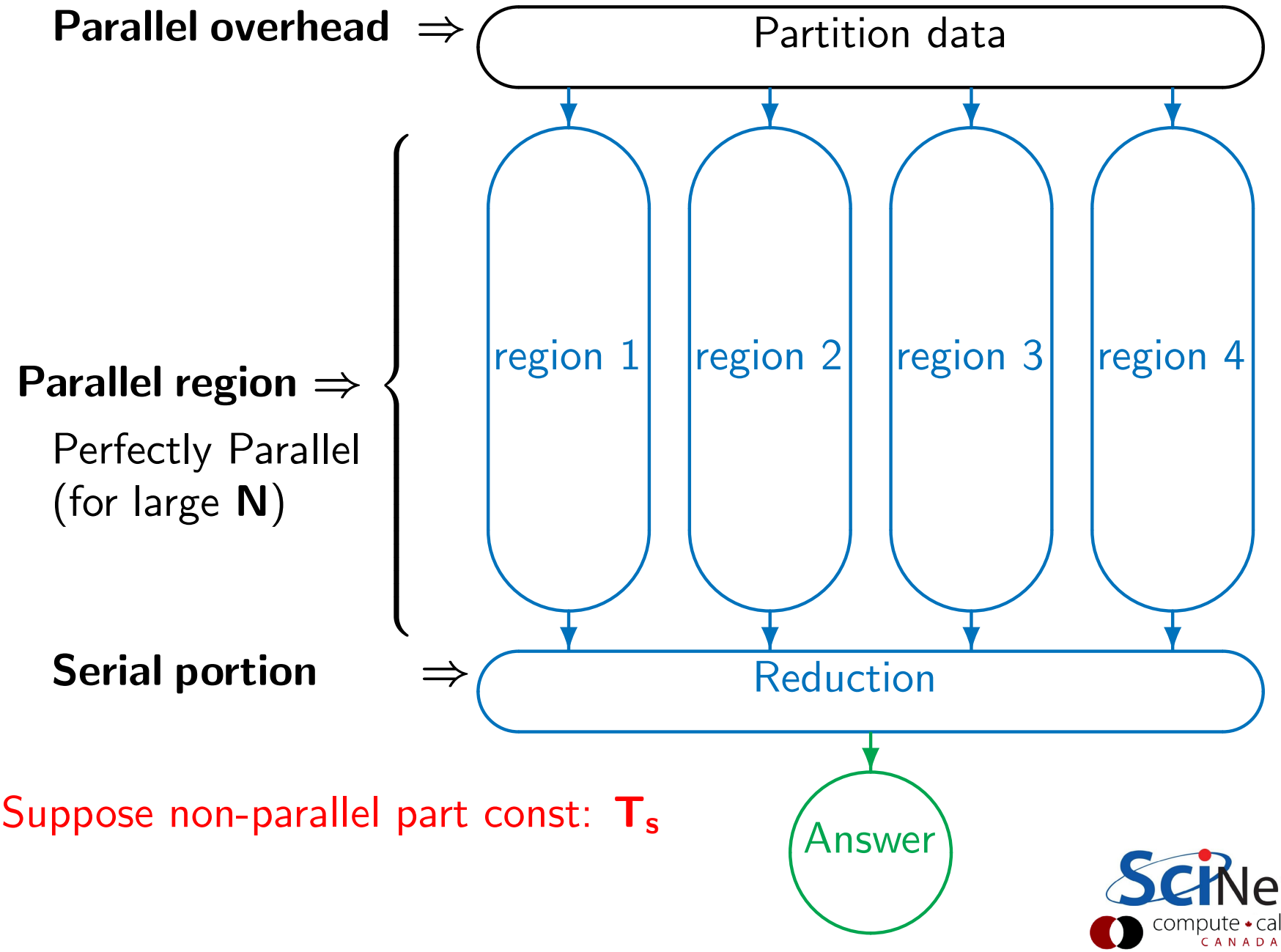
Serial portion $\Rightarrow$



Non-ideal cases



Non-ideal cases



Suppose non-parallel part const: $\mathbf{T_s}$

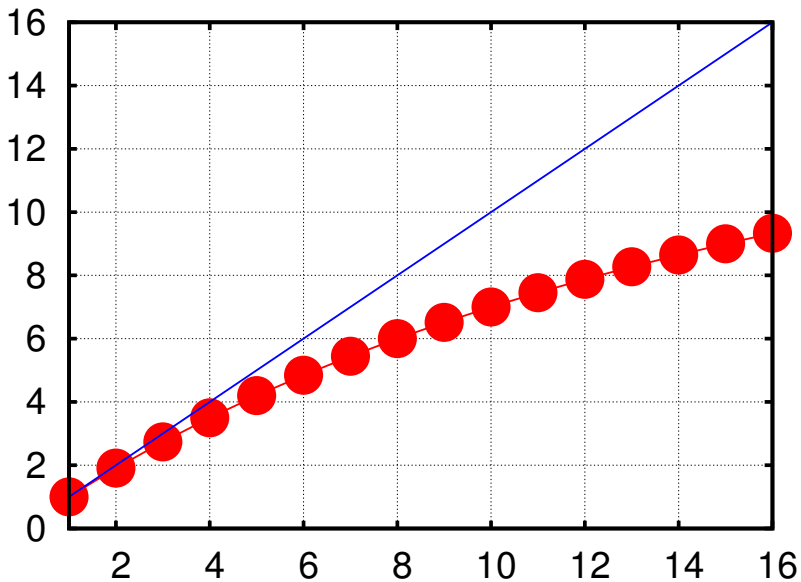
Amdahl's law

Speed-up (without parallel overhead):

$$S = \frac{T_{serial}}{T(P)} = \frac{NT_1 + T_s}{\frac{NT_1}{P} + T_s}$$

or, calling $f = T_s/(T_s + NT_1)$ the serial fraction,

$$S = \frac{1}{f + (1-f)/P}$$



(for $f = 5\%$)

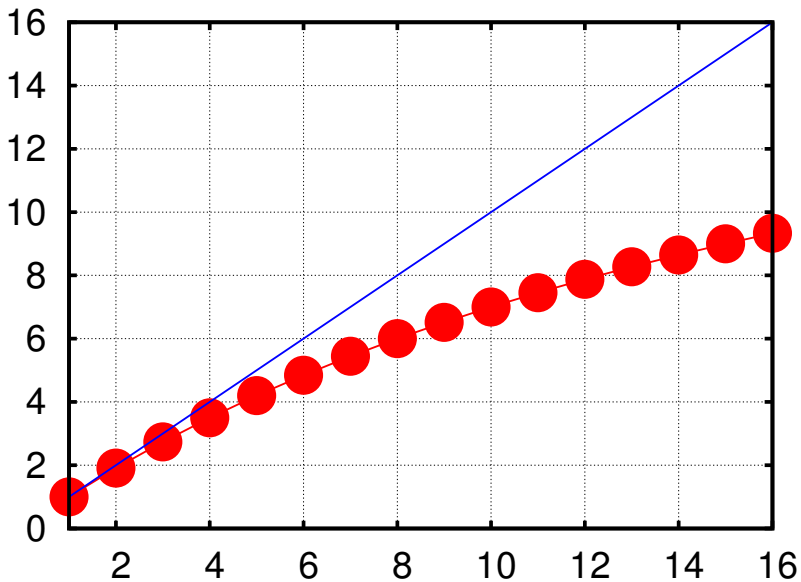
Amdahl's law

Speed-up (without parallel overhead):

$$S = \frac{T_{serial}}{T(P)} = \frac{NT_1 + T_s}{\frac{NT_1}{P} + T_s}$$

or, calling $f = T_s/(T_s + NT_1)$ the serial fraction,

$$S = \frac{1}{f + (1-f)/P} \xrightarrow{P\to\infty} \frac{1}{f}$$



(for $f = 5\%$)

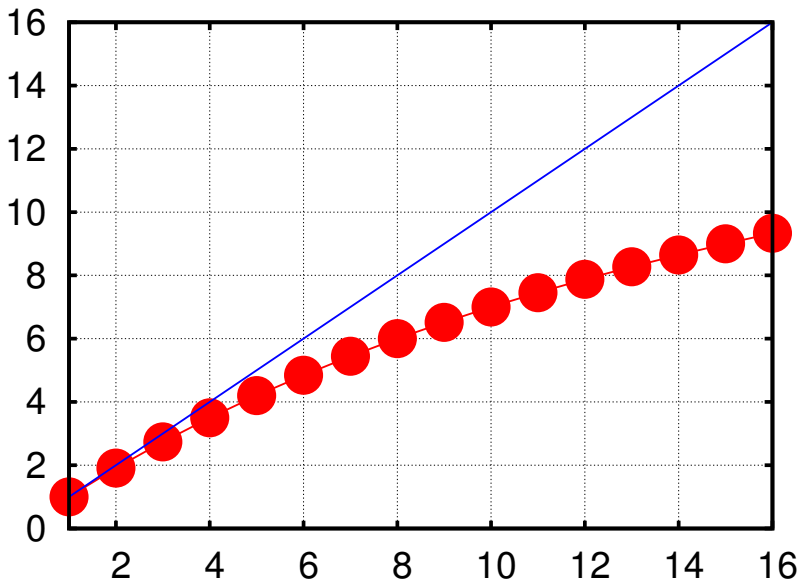
Amdahl's law

Speed-up (without parallel overhead):

$$S = \frac{T_{serial}}{T(P)} = \frac{NT_1 + T_s}{\frac{NT_1}{P} + T_s}$$

or, calling $f = T_s/(T_s + NT_1)$ the serial fraction,

$$S = \frac{1}{f + (1-f)/P} \xrightarrow{P\to\infty} \frac{1}{f}$$



Serial part dominates asymptotically.

Speed-up limited, no matter size of P.

(for $f = 5\%$)

Scaling efficiency

Speed-up compared to ideal factor **P**:

$$\textbf{Efficiency} = \frac{\textbf{S}}{\textbf{P}}$$

This will invariably fall off for larger P except for embarrasing parallel problems.

$$\textbf{Efficiency} \sim \frac{\textbf{1}}{\textbf{fP}} \overset{\textbf{P}\to\infty}{\longrightarrow} \textbf{0}$$

You cannot get 100% efficiency in any non-trivial problem.
All you can aim for here is to make the efficiency as least low as possible.

Less ideal case of Amdahl's law

We assumed reduction is constant.
But it will in fact increase with P,
from sum of results of all processors

$$\mathbf{T_s \approx P T_1}$$

Serial fraction now a function of **P**:

$$\mathbf{f(P) = \frac{P}{N}}$$

Amdahl:

$$\mathbf{S(P) = \frac{1}{f(P) + [1 - f(P)]/P}}$$

Less ideal case of Amdahl's law

We assumed reduction is constant.
But it will in fact increase with P,
from sum of results of all processors

$$\mathbf{T_s \approx PT_1}$$

Serial fraction now a function of **P**:

$$\mathbf{f(P) = \frac{P}{N}}$$

Amdahl:

$$\mathbf{S(P) = \frac{1}{f(P) + [1 - f(P)]/P}}$$

Example: $\mathbf{N = 100}$, $\mathbf{T_1 = 1s}$...

Less ideal case of Amdahl's law

We assumed reduction is constant.
But it will in fact increase with P,
from sum of results of all processors

$$T_s \approx PT_1$$

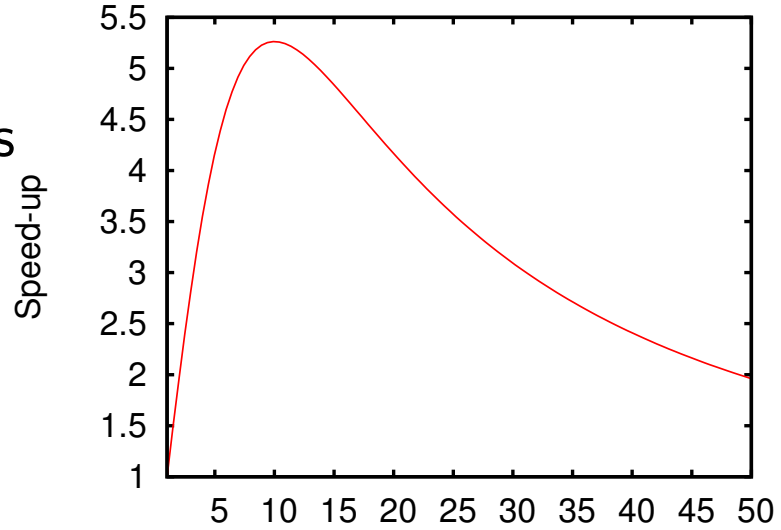
Serial fraction now a function of **P**:

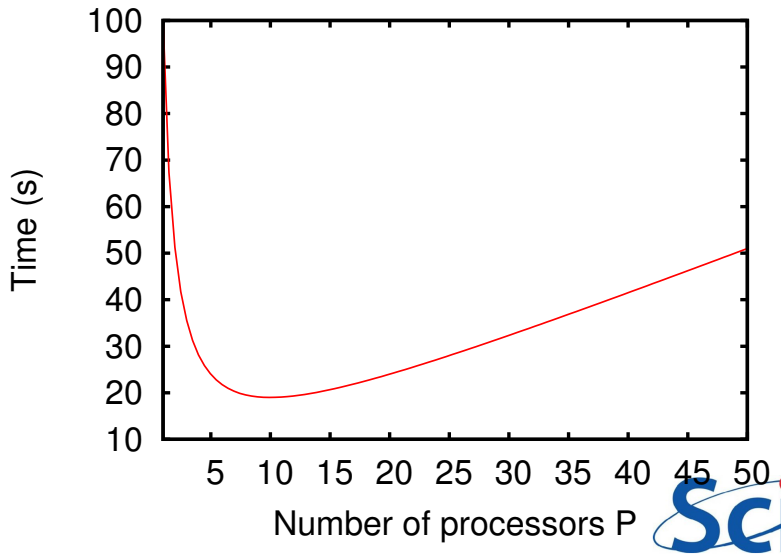
$$f(P) = \frac{P}{N}$$

Amdahl:

$$S(P) = \frac{1}{f(P) + [1 - f(P)]/P}$$

Example: $N = 100$, $T_1 = 1s$...





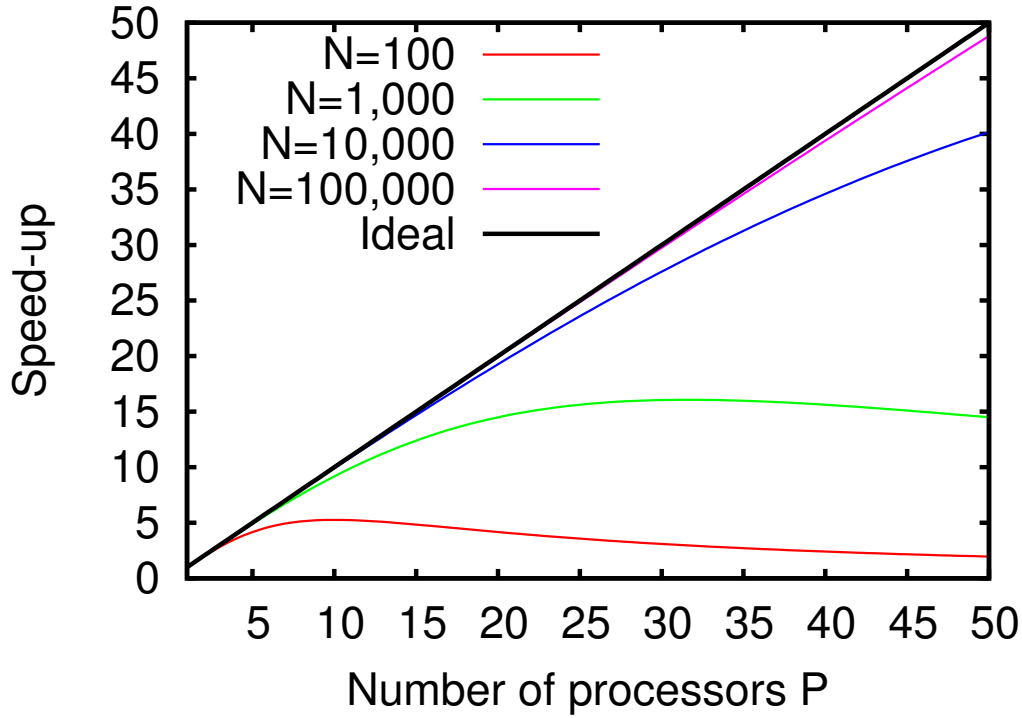
SciNet
compute • calcul
CANADA

Trying to beat Amdahl's law #1

Scale up!

The larger **N**, the smaller the serial fraction:

$$\mathbf{f(P) = \frac{P}{N}}$$



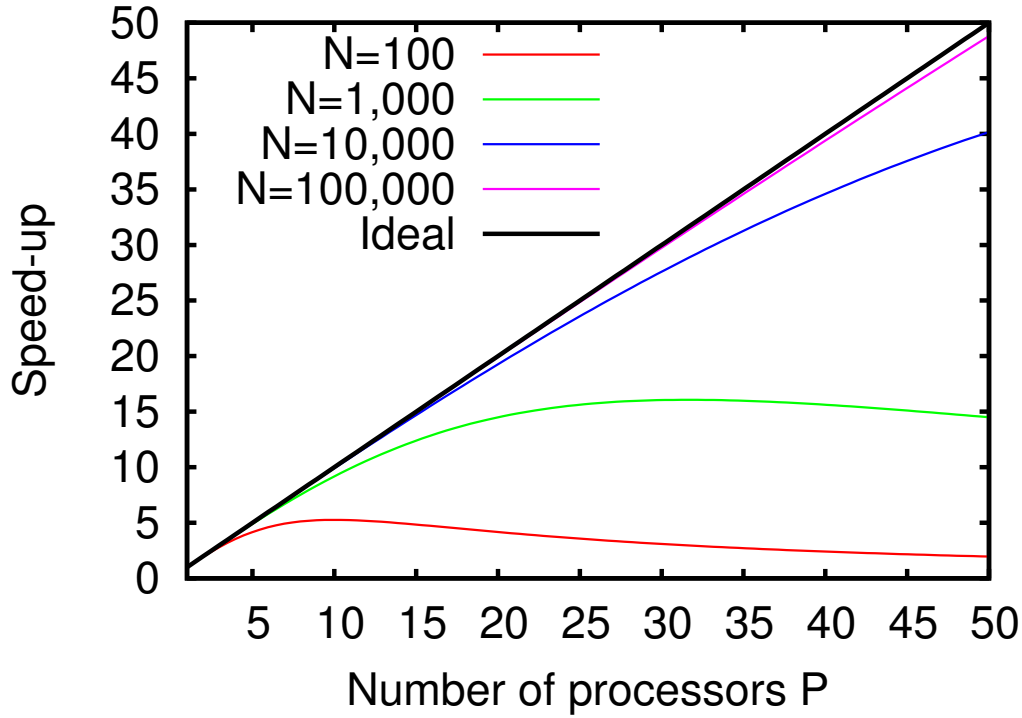
SciNet compute • calcul CANADA

Trying to beat Amdahl's law #1

Scale up!

The larger **N**, the smaller the serial fraction:

$$\mathbf{f(P) = \frac{P}{N}}$$



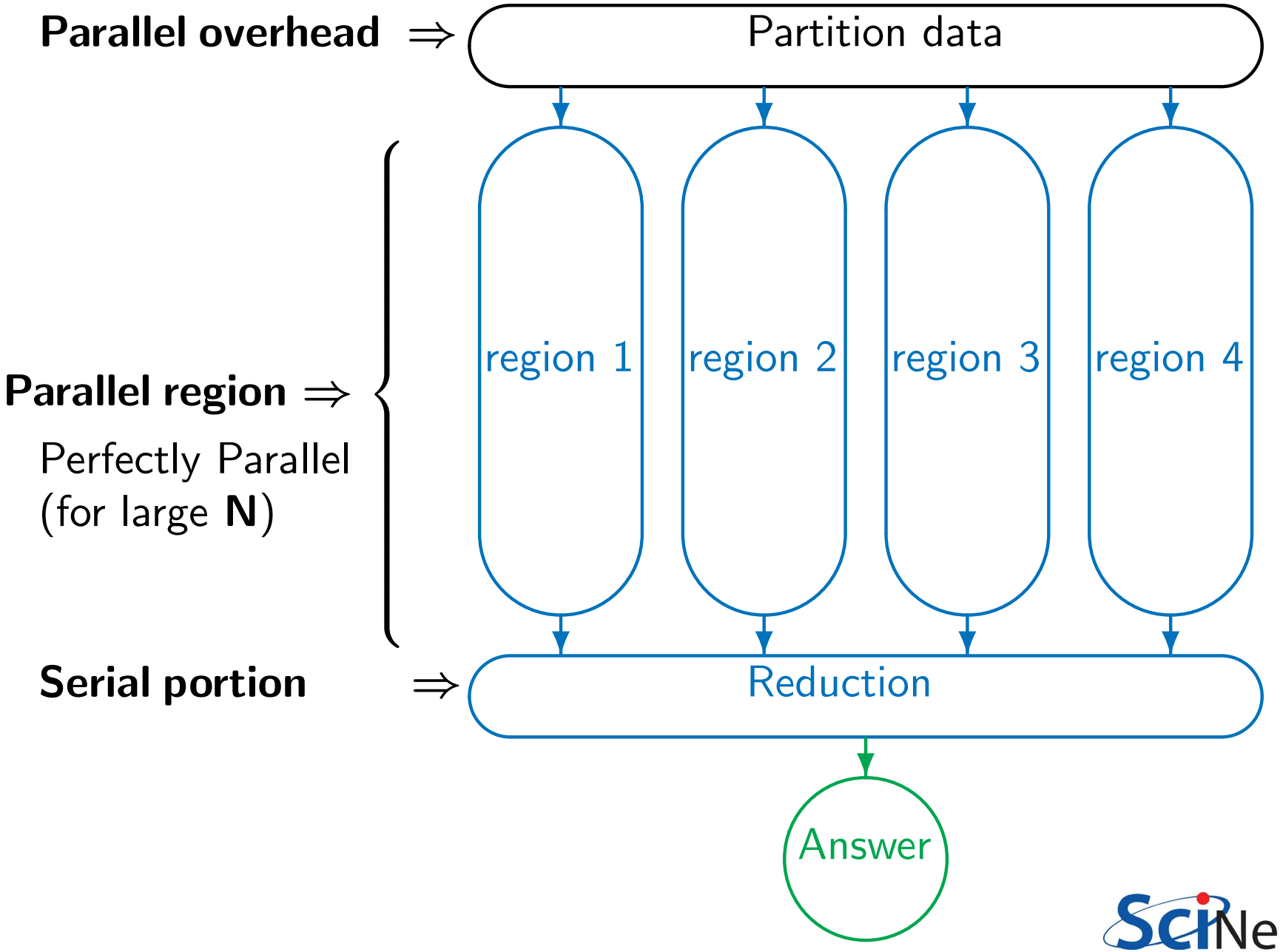
Weak scaling: Increase problem size while increasing **P**

$$\mathbf{Time_{weak}(P) = Time(N = n \times P, P)}$$

Good weak scaling means this time approaches a constant for large **P**.

SciNet compute • calcul CANADA

Trying to beat Amdahl's law #2



Trying to beat Amdahl's law #2

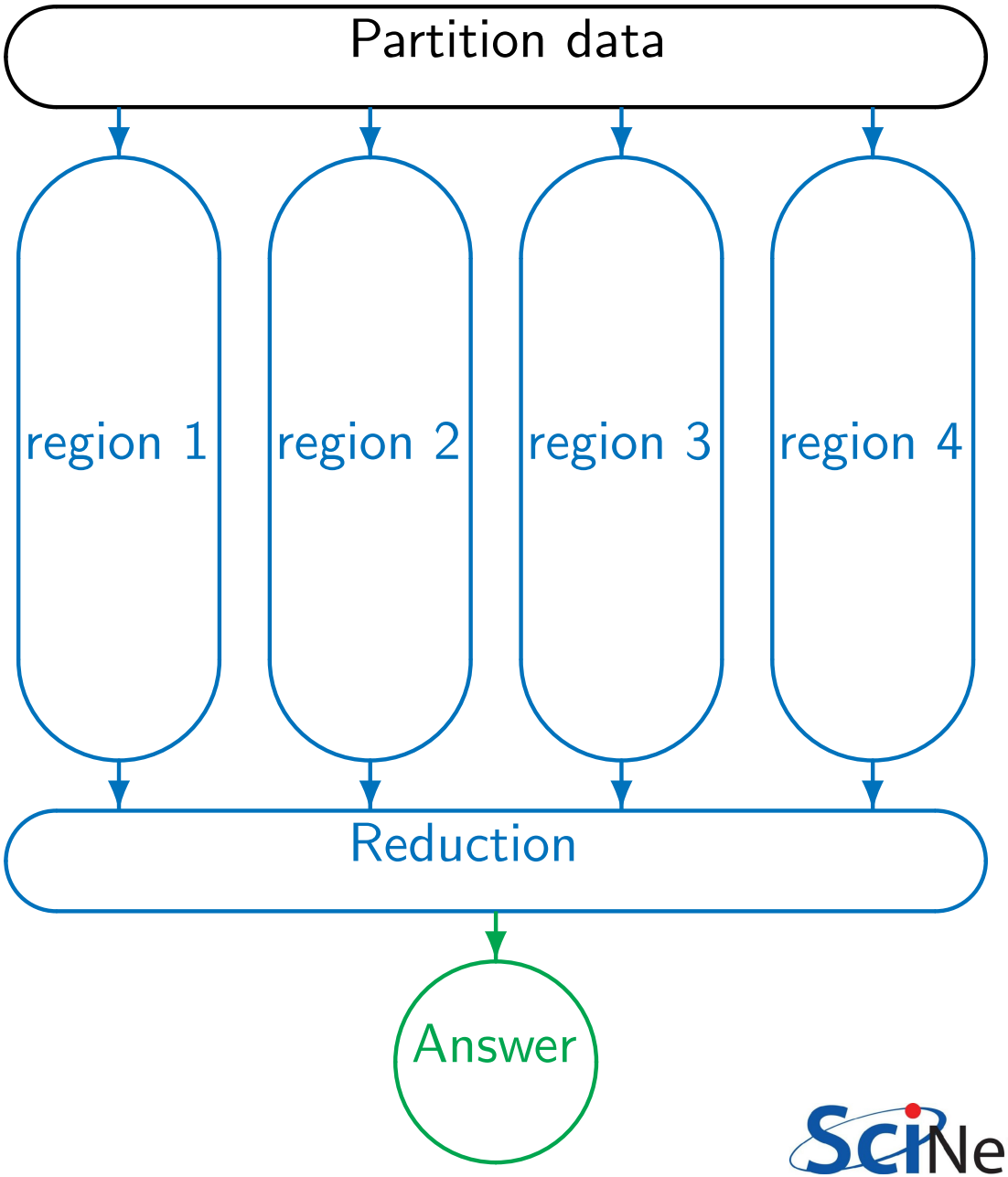
Parallel overhead ⇒

Parallel region ⇒

Perfectly Parallel
(for large **N**)

Serial portion ⇒

Rewrite



Trying to beat Amdahl's law #2

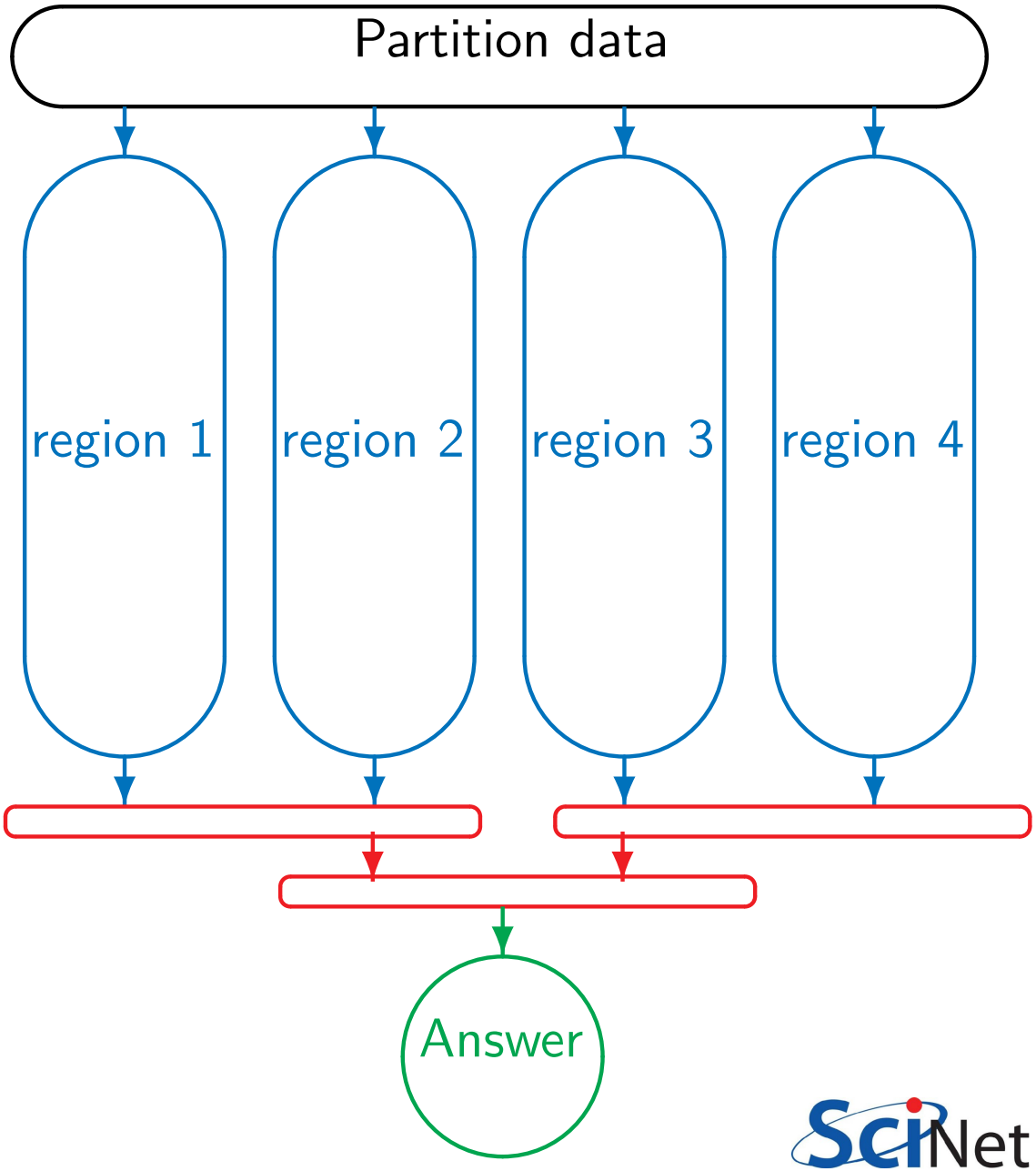
Parallel overhead ⇒

Parallel region ⇒

Perfectly Parallel
(for large **N**)

Serial portion ⇒

Rewrite



Trying to beat Amdahl's law #2

Parallel overhead $\Rightarrow$

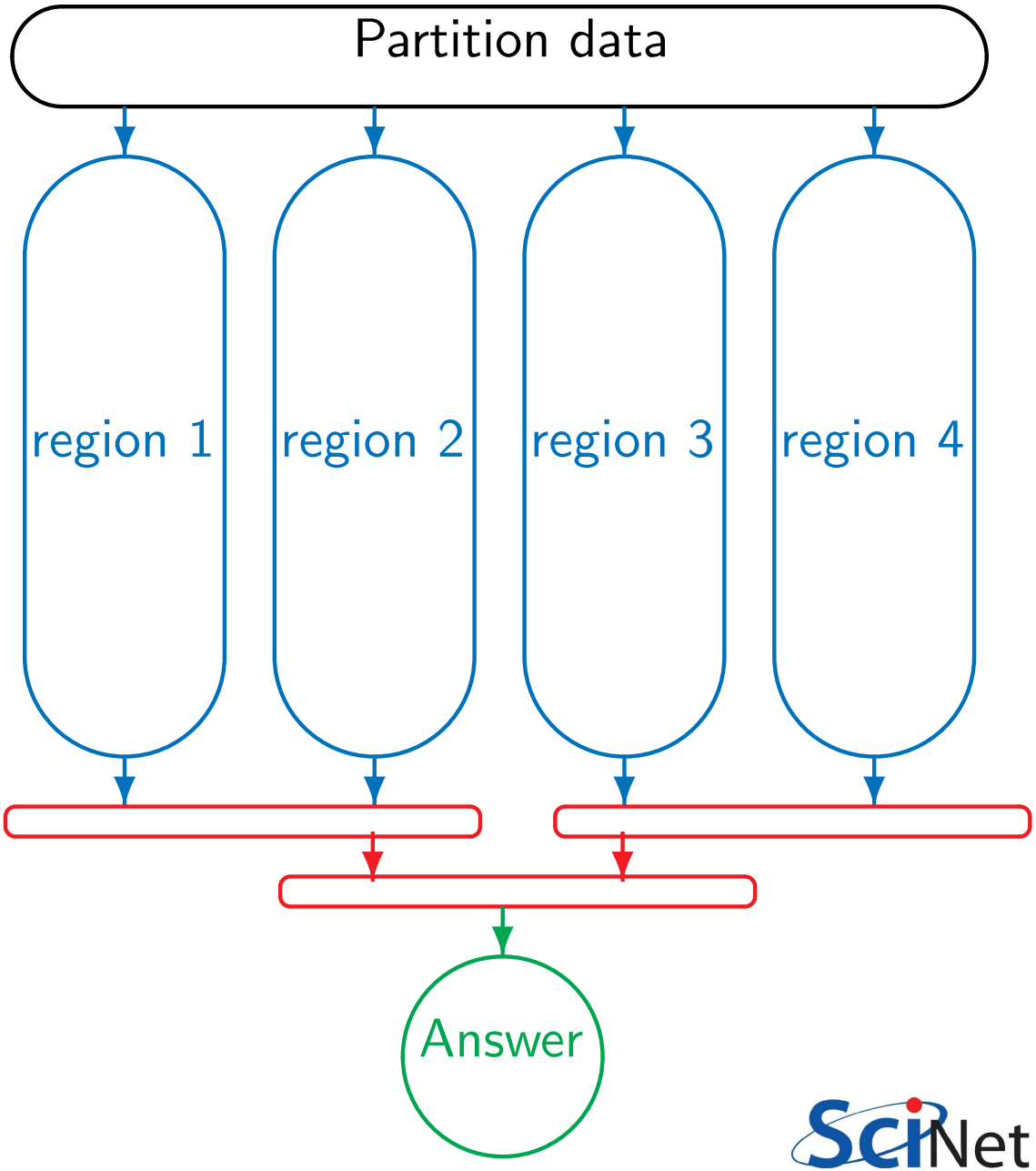
Parallel region $\Rightarrow$

Perfectly Parallel
(for large **N**)

Serial portion $\Rightarrow$

Rewrite

$\propto$ **2 log P**



SciNet
compute • calcul
CANADA

Trying to beat Amdahl's law #2

'Serial' fraction now different function of **P**:

$$\mathbf{f(P) = \frac{^2 \log P}{N}}$$

Amdahl:

$$\mathbf{S(P) = \frac{1}{f(P) + [1 - f(P)]/P}}$$

Trying to beat Amdahl's law #2

'Serial' fraction now different function of $\mathbf{P}$:

$$\mathbf{f(P) = \frac{^2 \log P}{N}}$$

Amdahl:

$$\mathbf{S(P) = \frac{1}{f(P) + [1 - f(P)]/P}}$$

Example: $\mathbf{N = 100}$, $\mathbf{T_1 = 1s}$...

Trying to beat Amdahl's law #2

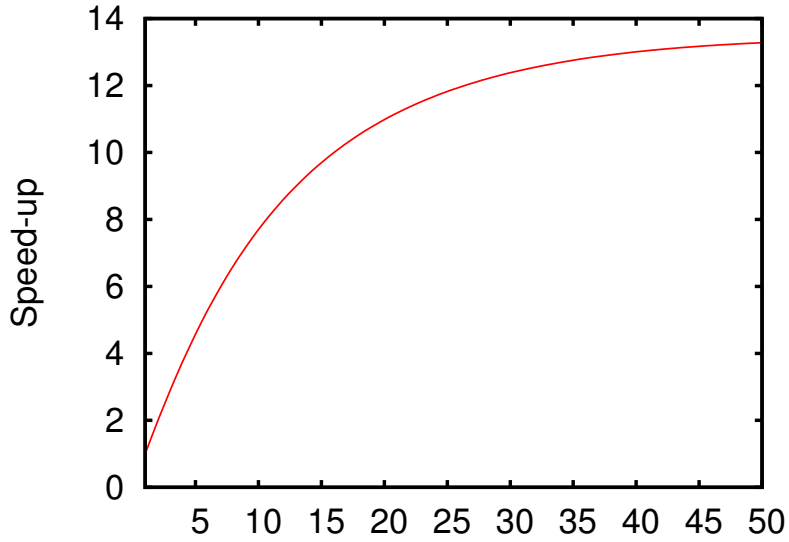
'Serial' fraction now different function of **P**:

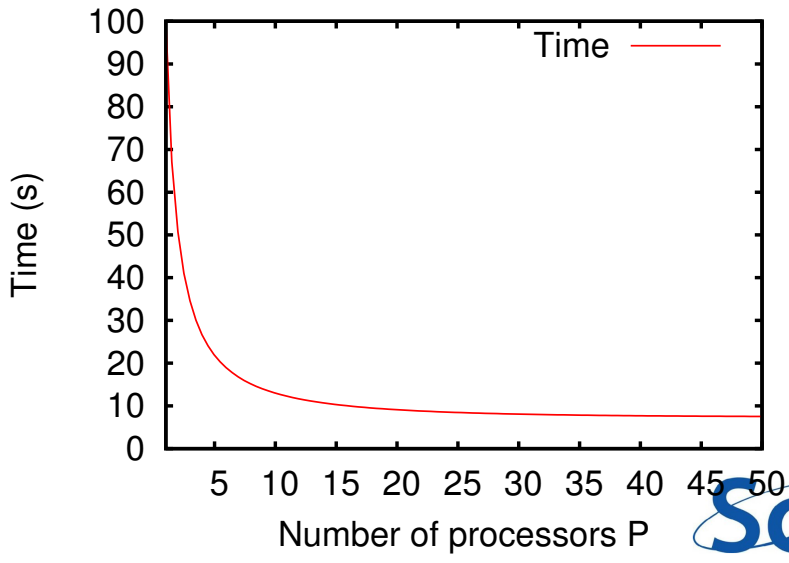
$$\mathbf{f(P) = \frac{^2 \log P}{N}}$$

Amdahl:

$$\mathbf{S(P) = \frac{1}{f(P) + [1 - f(P)]/P}}$$

Example: $\mathbf{N = 100}$, $\mathbf{T_1 = 1s}$...





Trying to beat Amdahl's law #2

Weak scaling

$$\mathbf{Time_{weak}(P) = Time(N = n \times P, P)}$$

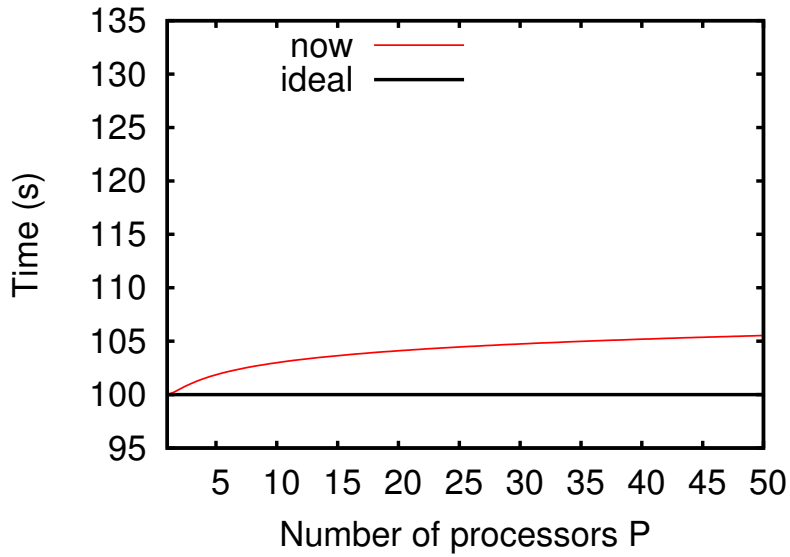
Should approach constant for large **P**.
Let's see...

Trying to beat Amdahl's law #2

Weak scaling

$$\mathbf{Time_{weak}(P) = Time(N = n \times P, P)}$$

Should approach constant for large **P**.
Let's see...
Not quite!



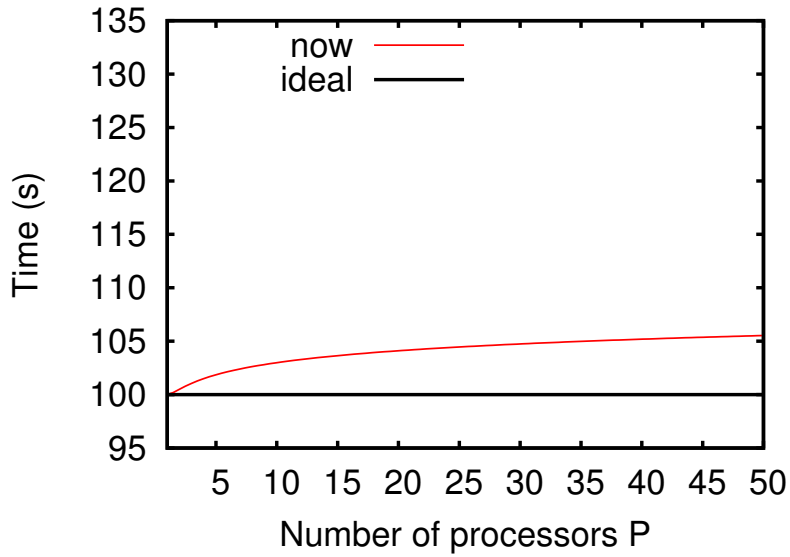
SciNet
compute • calcul
CANADA

Trying to beat Amdahl's law #2

Weak scaling

$$\mathbf{Time_{weak}(P) = Time(N = n \times P, P)}$$

Should approach constant for large **P**.
Let's see...
Not quite!
But much better than before.



SciNet compute • calcul CANADA

Trying to beat Amdahl's law #2

Weak scaling

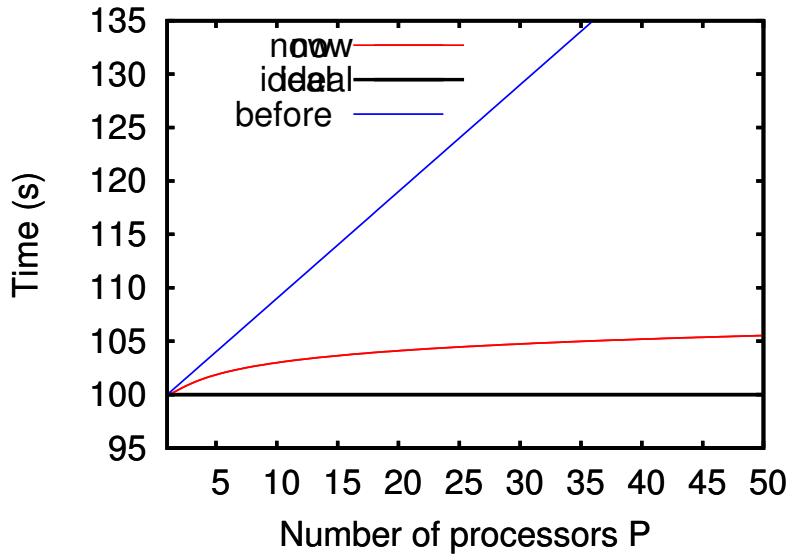
$$\mathbf{Time_{weak}(P) = Time(N = n \times P, P)}$$

Should approach constant for large **P**.

Let's see...

Not quite!

But much better than before.



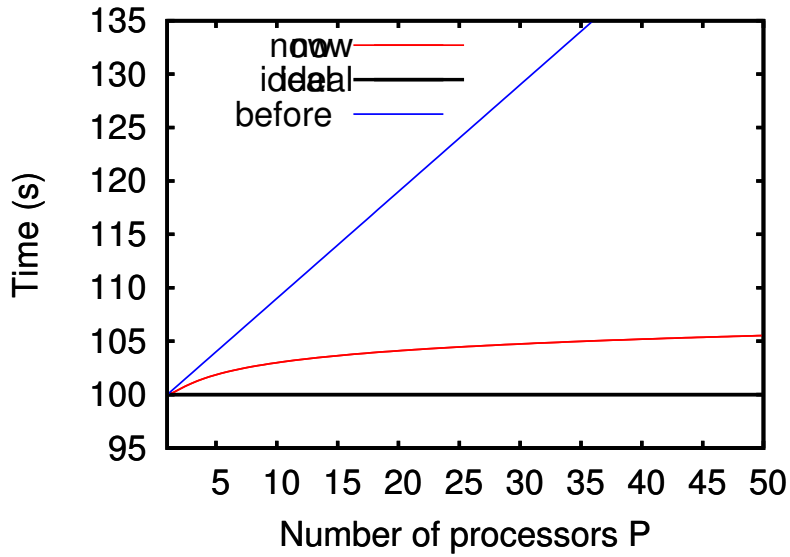
SciNet compute • calcul CANADA

Trying to beat Amdahl's law #2

Weak scaling

$$\mathbf{Time_{weak}(P) = Time(N = n \times P, P)}$$

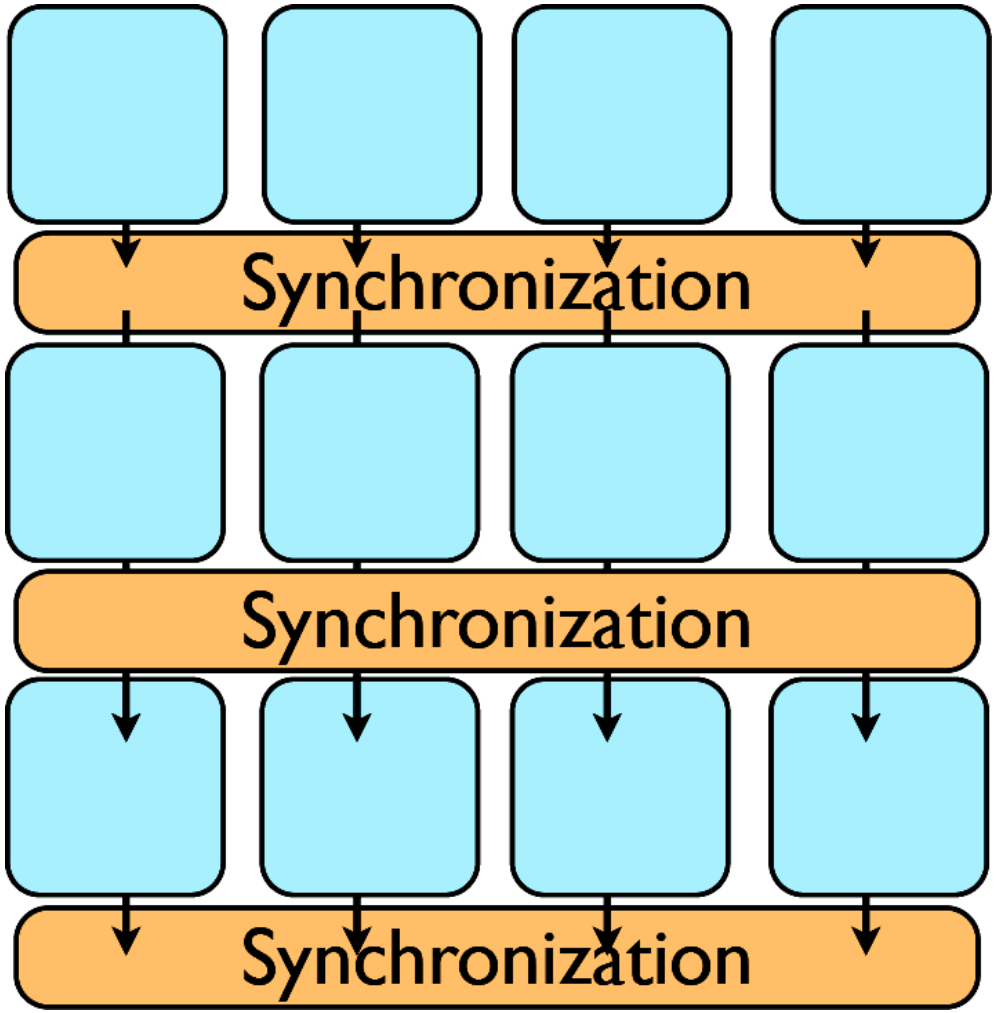
Should approach constant for large **P**.
Let's see...
Not quite!
But much better than before.



Really not that bad.
& other algorithms can do better.

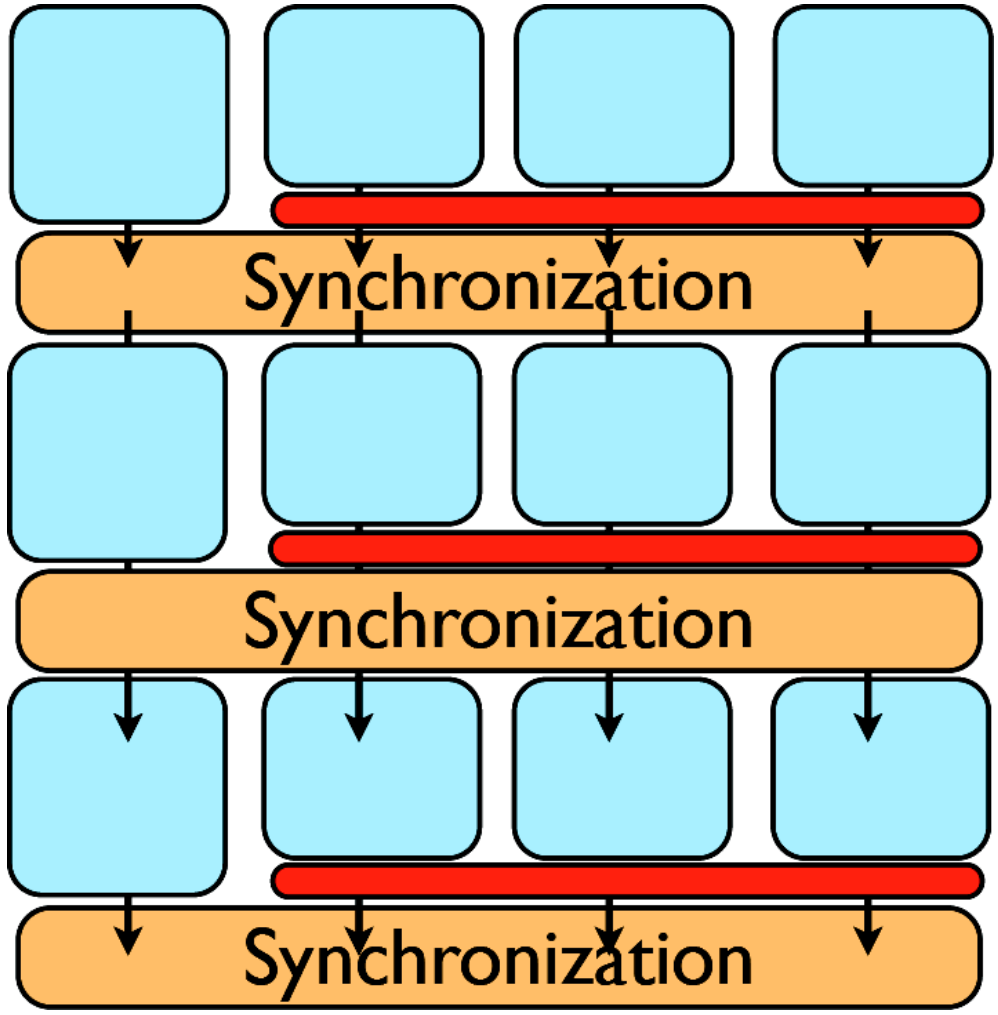
Synchronization

- Most problems are not purely concurrent.
- Some level of synchronization or exchange of information is needed between tasks.
- While synchronizing, nothing else happens: increases Amdahl's **f**.
- And synchronizations are themselves costly.



Load balancing

- The division of calculations among the processors may not be equal.
- Some processors would already be done, while others are still going.
- Effectively using less than **P** processors: This reduces the efficiency.
- Aim for load balanced algorithms.



Locality

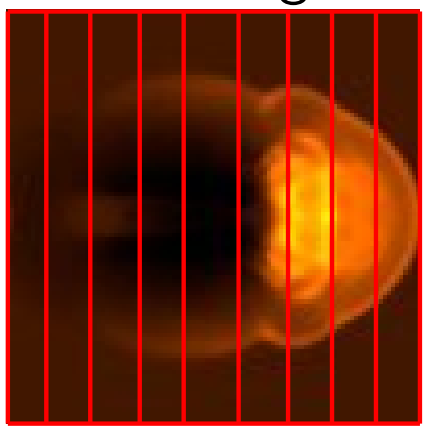
- So far we neglected communication costs.
- But communication costs are more expensive than computation!
- To minimize communication to computation ratio:
  * Keep the data where it is needed.
  * Make sure as little data as possible is to be communicated.
  * Make shared data as local to the right processors as possible.
- Local data means less need for syncs, or smaller-scale syncs.
- Local syncs can alleviate load balancing issues.

Locality

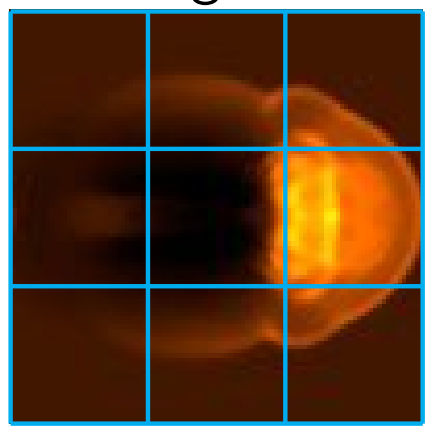
- So far we neglected communication costs.
- But communication costs are more expensive than computation!
- To minimize communication to computation ratio:
  * Keep the data where it is needed.
  * Make sure as little data as possible is to be communicated.
  * Make shared data as local to the right processors as possible.
- Local data means less need for syncs, or smaller-scale syncs.
- Local syncs can alleviate load balancing issues.

Example (PDE Domain decomposition)

wrong



right



Big Lesson

Parallel algorithm design is about finding as much concurrency as possible, and arranging it in a way that maximizes locality.

Parallel Computers

Top500.org:

List of the worlds 500 largest supercomputers. Updated every 6 months,

Info on architecture, etc.

Home ▸ Lists ▸ November 2010

TOP500 List - November 2010 (1-100)

R_{max} and R_{peak} values are in TFlops. For more details about other fields, check the TOP500 description.

Power data in KW for entire system

next

Rank	Site	Computer/Year Vendor	Cores	R_{max}	R_{peak}	Power
1	National Supercomputing Center in Tianjin China	Tianhe-1A - NUDT TH MPP, X5670 2.93Ghz 6C, NVIDIA GPU, FT-1000 8C / 2010 NUDT	186368	2566.00	4701.00	4040.00
2	DOE/SC/Oak Ridge National Laboratory United States	Jaguar - Cray XT5-HE Opteron 6-core 2.6 GHz / 2009 Cray Inc.	224162	1759.00	2331.00	6950.60
3	National Supercomputing Centre in Shenzhen (NSCS) China	Nebulae - Dawning TC3600 Blade, Intel X5650, NVidia Tesla C2050 GPU / 2010 Dawning	120640	1271.00	2984.30	2580.00
4	GSIC Center, Tokyo Institute of Technology Japan	TSUBAME 2.0 - HP ProLiant SL390s G7 Xeon 6C X5670, Nvidia GPU, Linux/Windows / 2010 NEC/HP	73278	1192.00	2287.63	1398.61
5	DOE/SC/LBNL/NERSC United States	Hopper - Cray XE6 12-core 2.1 GHz / 2010 Cray Inc.	153408	1054.00	1288.63	2910.00
6	Commissariat a l'Energie Atomique (CEA) France	Tera-100 - Bull bullx super-node S6010/S6030 / 2010 Bull SA	138368	1050.00	1254.55	4590.00