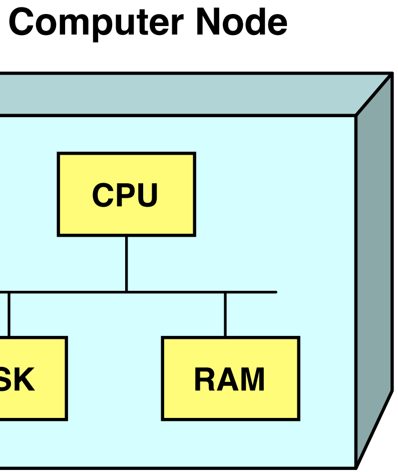
Parallel and Distributed Processing

Enterprise Architectures for Big Data

Single Node



Questions

- What is the difference between parallel computing and distributed computing?
  - Parallel Computing
    - more than one Core, CPU, GPU or Node is working on an algorithm at the same time
  - Distributed Computing
    - More than one Node is working on an algorithm at the same time

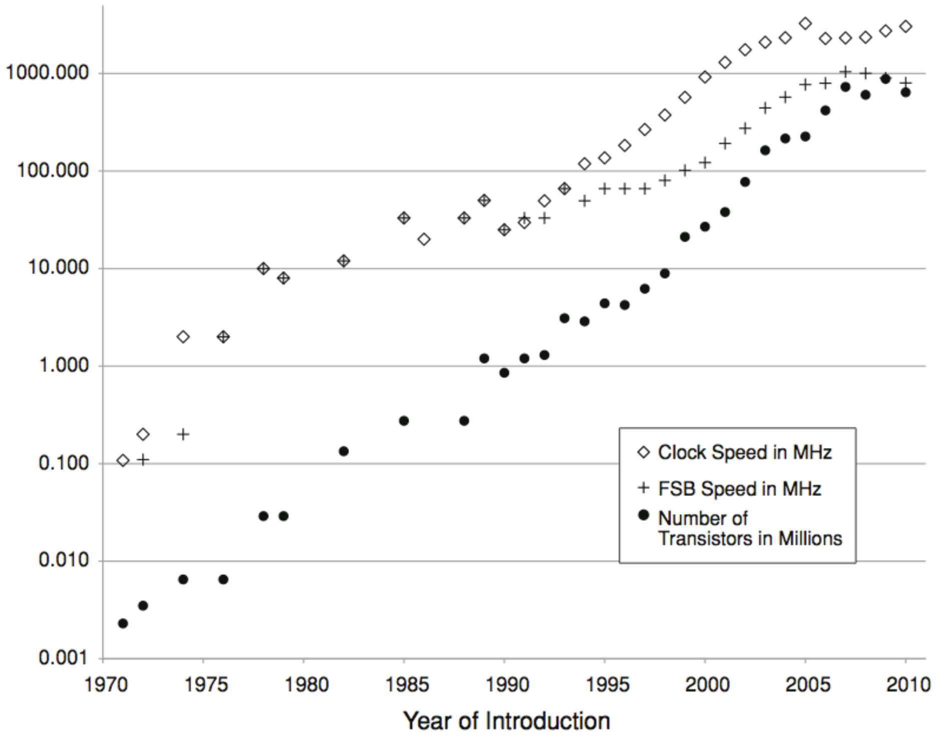
- What is the opposite of parallel computing?
  - Sequential Computing

Questions

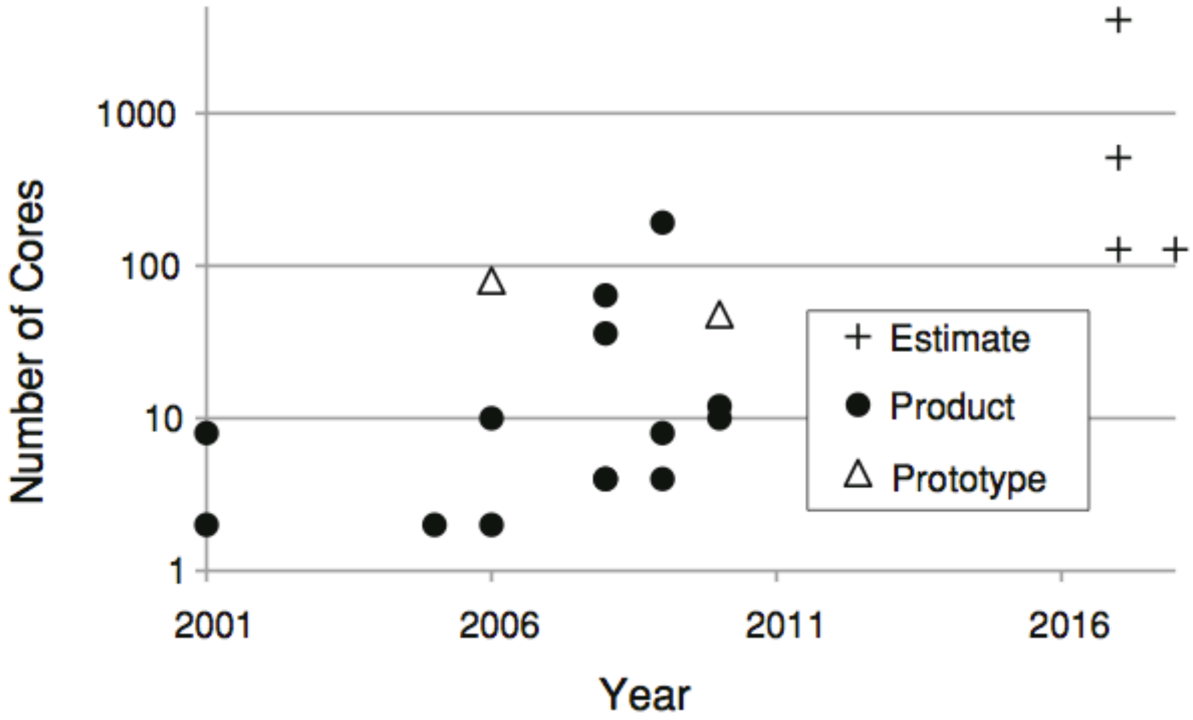
- Why distribute computation?
  - Speedup algorithms through parallel computing
  - Fault tolerance
  - Reduce latency through nearer nodes

- Why distribute data?
  - Storage limits of single node
  - Store data near computation
  - Fault tolerance through replication
  - Reduce latency through nearer nodes
  - Improves reading speed through parallel reading

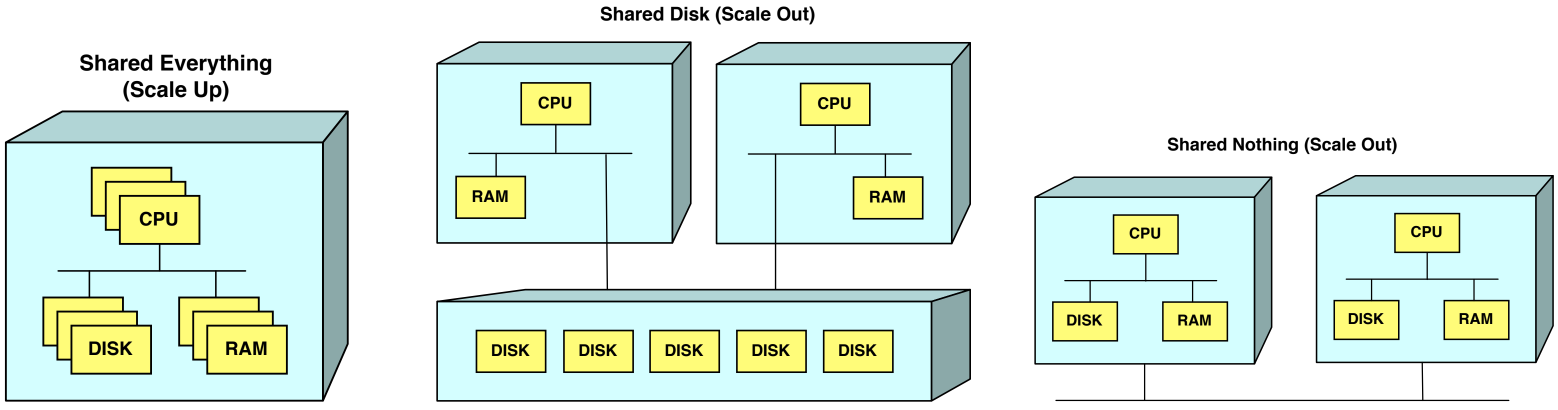
Clock speed, Front Side Bus (FSB) speed, and number of transistors



Development of number of cores



Scaling Architectures: Scale up vs. Scale out



E.g.
SAN (Storage Area Network) or
NAS (Network Attached Storage)

Latency Comparison Numbers

L1 cache reference	0.5ns			
L2 cache reference	7	ns		14x L1 cache
Main memory reference	100	ns		14x L2 cache, 200x L1 cache
Send 1KB over 1 Gbps network	10,000	ns	0.01 ms	
SSD seek	100,000	ns	0.1 ms	1000x memory reference
Read 4KB randomly from SSD*	150,000	ns	0.15 ms	
Read 1 MB sequentially from memory	250,000	ns	0.25 ms	
Round trip within same datacenter	500,000	ns	0.5 ms	5x SSD seek, 5,000x memory reference
Read 1 MB sequentially from SSD*	1,000,000	ns	1 ms	4X memory sequential read
Disk seek	10,000,000	ns	10 ms	20x datacenter roundtrip, 100x SSD seek, 100,000x memory reference
Read 1 MB sequentially from disk	20,000,000	ns	20 ms	80x memory, 20X SSD
Send packet CA->Netherlands->CA	150,000,000	ns	150 ms	

Source: https://gist.github.com/jboner/2841832

Latency Comparison Numbers
Lets multiply all these durations by a billion (10^9)

### Seconds:	
L1 cache reference 0.5 s	One heart beat (0.5 s)
L2 cache reference 7 s	Long yawn
### Minutes:	
Main memory reference 100 s	Brushing your teeth
### Hours:	
Send 2K bytes over 1 Gbps network 5.5 hr	From lunch to end of work day
### Days	
SSD seek 1.7 days	A normal weekend
Read 1 MB sequentially from memory 2.9 days	A long weekend
Round trip within same datacenter 5.8 days	A medium vacation
Read 1 MB sequentially from SSD 11.6 days	Waiting for almost 2 weeks for a delivery
### Months	
Disk seek 16.5 weeks	A semester in university
Read 1 MB sequentially from disk 7.8 months	Almost producing a new human being
	The above 2 together = 1 year
### Years	
Send packet CA->Netherlands->CA 4.8 years	Average time it takes to complete a bachelor's degree

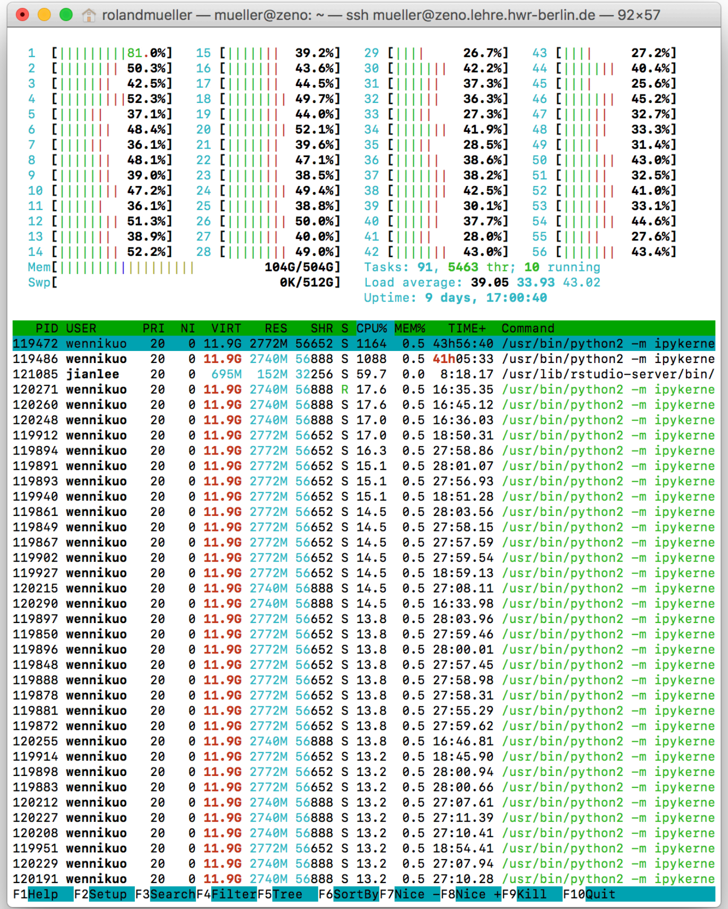
The Problem of Shared State

```
x = 10
x = function1(x)
x = function2(x)
x = x + 10
```

- Value of x is time-dependent
- Data dependency
- Single process (sequential computing) ➔ No problem
- Multiple parallel processes
  - Locking of value
  - Processes have to wait for each other

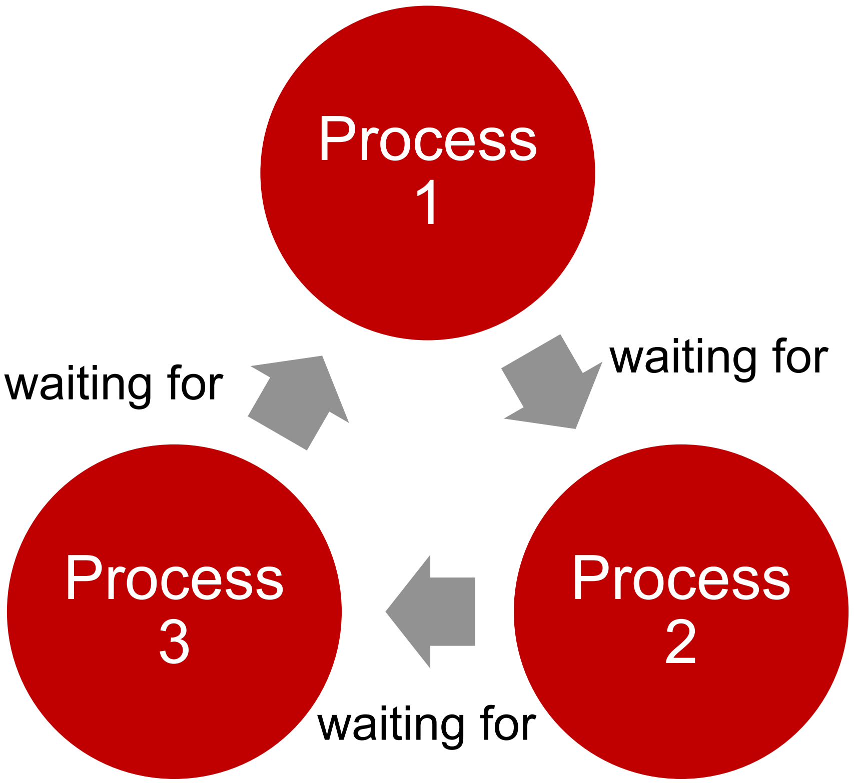
Multi-Core

htop



rolandmueller — mueller@zeno: ~ — ssh mueller@zeno.lehre.hwr-berlin.de — 92×57
1 [||||||||81.0%] 15 [|||||| 39.2%] 29 [|||| 26.7%] 43 [||| 27.2%]
2 [|||||||| 50.3%] 16 [|||||| 43.6%] 30 [||||||| 42.2%] 44 [||||||| 40.4%]
3 [|||||| 42.5%] 17 [|||||| 44.5%] 31 [||||| 37.3%] 45 [||| 25.6%]
4 [||||||||52.3%] 18 [|||||||| 49.7%] 32 [||||| 36.3%] 46 [|||||| 45.2%]
5 [||||| 37.1%] 19 [|||||| 44.0%] 33 [|||| 27.3%] 47 [||||| 32.7%]
6 [|||||| 48.4%] 20 [|||||||| 52.1%] 34 [|||||| 41.9%] 48 [||||| 33.3%]
7 [||||| 36.1%] 21 [|||||| 39.6%] 35 [|||| 28.5%] 49 [|||| 31.4%]
8 [|||||| 48.1%] 22 [|||||| 47.1%] 36 [||||| 38.6%] 50 [|||||| 43.0%]
9 [|||||| 39.0%] 23 [|||||| 38.5%] 37 [|||||| 38.2%] 51 [||||| 32.5%]
10 [||||||| 47.2%] 24 [||||||| 49.4%] 38 [|||||| 42.5%] 52 [|||||| 41.0%]
11 [||||| 36.1%] 25 [|||||| 38.8%] 39 [||||| 30.1%] 53 [||||| 33.1%]
12 [||||||| 51.3%] 26 [||||||| 50.0%] 40 [||||| 37.7%] 54 [|||||| 44.6%]
13 [|||||| 38.9%] 27 [|||||| 40.0%] 41 [|||| 28.0%] 55 [|||| 27.6%]
14 [||||||| 52.2%] 28 [||||||| 49.0%] 42 [|||||| 43.0%] 56 [|||||| 43.4%]
Mem[||||||||||||||||| 104G/504G] Tasks: 91, 5463 thr; 10 running
Swp[0K/512G] Load average: 39.05 33.93 43.02
Uptime: 9 days, 17:00:40
PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command
119472 wennikuo 20 0 11.9G 2772M 56652 S 1164 0.5 43h56:40 /usr/bin/python2 -m ipykerne
119486 wennikuo 20 0 11.9G 2740M 56888 S 1088 0.5 41h05:33 /usr/bin/python2 -m ipykerne
121085 jianlee 20 0 695M 152M 32256 S 59.7 0.0 8:18.17 /usr/lib/rstudio-server/bin/
120271 wennikuo 20 0 11.9G 2740M 56888 R 17.6 0.5 16:35.35 /usr/bin/python2 -m ipykerne
120260 wennikuo 20 0 11.9G 2740M 56888 S 17.6 0.5 16:45.12 /usr/bin/python2 -m ipykerne
120248 wennikuo 20 0 11.9G 2740M 56888 S 17.0 0.5 16:36.03 /usr/bin/python2 -m ipykerne
119912 wennikuo 20 0 11.9G 2772M 56652 S 17.0 0.5 18:50.31 /usr/bin/python2 -m ipykerne
119894 wennikuo 20 0 11.9G 2772M 56652 S 16.3 0.5 27:58.86 /usr/bin/python2 -m ipykerne
119891 wennikuo 20 0 11.9G 2772M 56652 S 15.1 0.5 28:01.07 /usr/bin/python2 -m ipykerne
119893 wennikuo 20 0 11.9G 2772M 56652 S 15.1 0.5 27:56.93 /usr/bin/python2 -m ipykerne
119940 wennikuo 20 0 11.9G 2772M 56652 S 15.1 0.5 18:51.28 /usr/bin/python2 -m ipykerne
119861 wennikuo 20 0 11.9G 2772M 56652 S 14.5 0.5 28:03.56 /usr/bin/python2 -m ipykerne
119849 wennikuo 20 0 11.9G 2772M 56652 S 14.5 0.5 27:58.15 /usr/bin/python2 -m ipykerne
119867 wennikuo 20 0 11.9G 2772M 56652 S 14.5 0.5 27:57.59 /usr/bin/python2 -m ipykerne
119902 wennikuo 20 0 11.9G 2772M 56652 S 14.5 0.5 27:59.54 /usr/bin/python2 -m ipykerne
119927 wennikuo 20 0 11.9G 2772M 56652 S 14.5 0.5 18:59.13 /usr/bin/python2 -m ipykerne
120215 wennikuo 20 0 11.9G 2740M 56888 S 14.5 0.5 27:08.11 /usr/bin/python2 -m ipykerne
120290 wennikuo 20 0 11.9G 2740M 56888 S 14.5 0.5 16:33.98 /usr/bin/python2 -m ipykerne
119897 wennikuo 20 0 11.9G 2772M 56652 S 13.8 0.5 28:03.96 /usr/bin/python2 -m ipykerne
119850 wennikuo 20 0 11.9G 2772M 56652 S 13.8 0.5 27:59.46 /usr/bin/python2 -m ipykerne
119896 wennikuo 20 0 11.9G 2772M 56652 S 13.8 0.5 28:00.01 /usr/bin/python2 -m ipykerne
119848 wennikuo 20 0 11.9G 2772M 56652 S 13.8 0.5 27:57.45 /usr/bin/python2 -m ipykerne
119888 wennikuo 20 0 11.9G 2772M 56652 S 13.8 0.5 27:58.98 /usr/bin/python2 -m ipykerne
119878 wennikuo 20 0 11.9G 2772M 56652 S 13.8 0.5 27:58.31 /usr/bin/python2 -m ipykerne
119881 wennikuo 20 0 11.9G 2772M 56652 S 13.8 0.5 27:55.29 /usr/bin/python2 -m ipykerne
119872 wennikuo 20 0 11.9G 2772M 56652 S 13.8 0.5 27:59.62 /usr/bin/python2 -m ipykerne
120255 wennikuo 20 0 11.9G 2740M 56888 S 13.8 0.5 16:46.81 /usr/bin/python2 -m ipykerne
119914 wennikuo 20 0 11.9G 2772M 56652 S 13.2 0.5 18:45.90 /usr/bin/python2 -m ipykerne
119898 wennikuo 20 0 11.9G 2772M 56652 S 13.2 0.5 28:00.94 /usr/bin/python2 -m ipykerne
119883 wennikuo 20 0 11.9G 2772M 56652 S 13.2 0.5 28:00.66 /usr/bin/python2 -m ipykerne
120212 wennikuo 20 0 11.9G 2740M 56888 S 13.2 0.5 27:07.61 /usr/bin/python2 -m ipykerne
120227 wennikuo 20 0 11.9G 2740M 56888 S 13.2 0.5 27:11.39 /usr/bin/python2 -m ipykerne
120208 wennikuo 20 0 11.9G 2740M 56888 S 13.2 0.5 27:10.41 /usr/bin/python2 -m ipykerne
119951 wennikuo 20 0 11.9G 2772M 56652 S 13.2 0.5 18:54.41 /usr/bin/python2 -m ipykerne
120229 wennikuo 20 0 11.9G 2740M 56888 S 13.2 0.5 27:07.94 /usr/bin/python2 -m ipykerne
120191 wennikuo 20 0 11.9G 2740M 56888 S 13.2 0.5 27:10.28 /usr/bin/python2 -m ipykerne
F1Help F2Setup F3Search F4Filter F5Tree F6SortBy F7Nice - F8Nice + F9Kill F10Quit

Deadlock

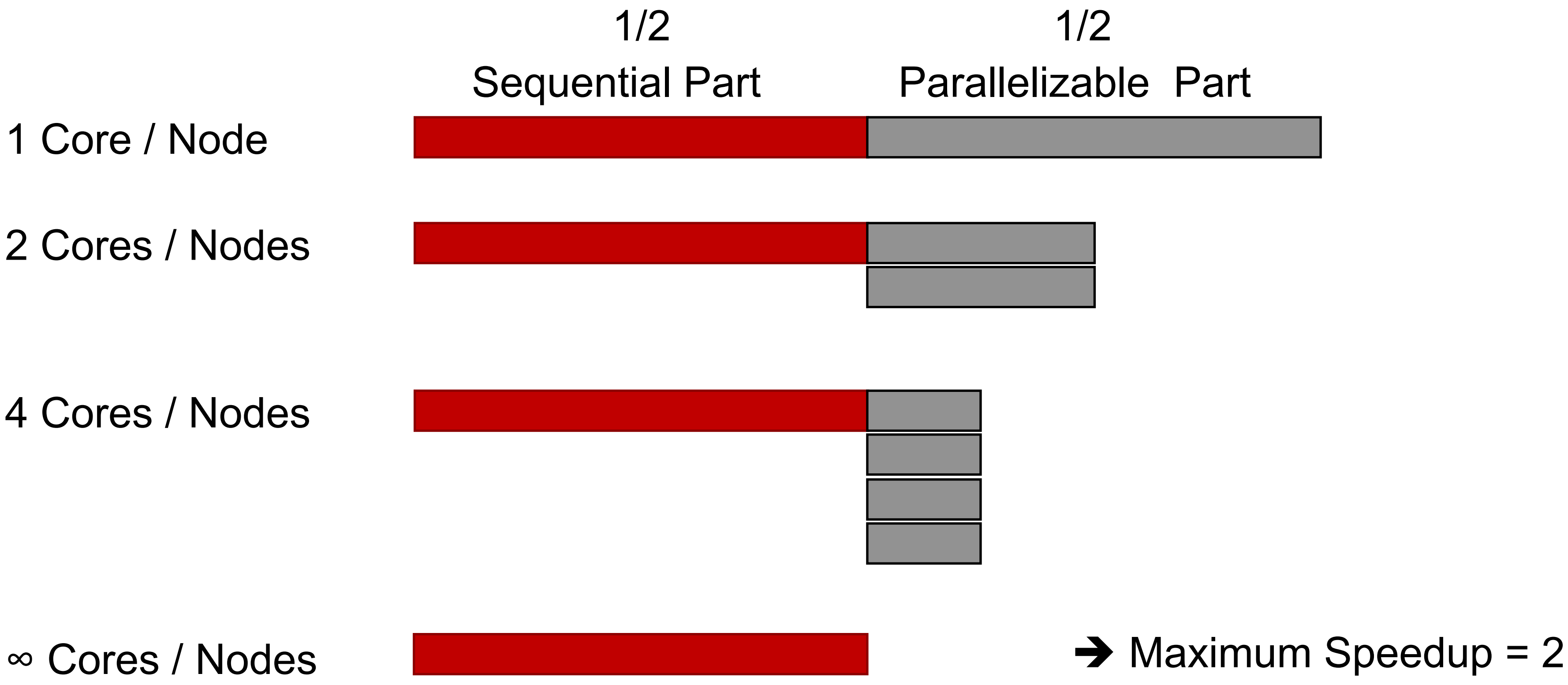

Process
1
waiting for
Process
2
waiting for
Process
3
waiting for

Pure Functions and Parallel Processing

```
x = 10
y = function1(x)
z = function2(x)
a = x + y
```

- Step 2 and 3 and be only parallelized if `function1` and `function2` are pure functions
  - No side effects
  - No internal state
  - Result is only determined by the input parameter

Amdahl‘s Law Visually Explained



Ahmdal's Law

- T_1 Total runtime of a Task (with one CPU)
- T_s Runtime of sequential part (can not be parallelized)
- T_p Runtime of parallelizable part

  $T_1 = T_s + T_p$

- f Parallelizable fraction of program

  $f = \frac{T_p}{T_1}$ with $0 \leq f \leq 1$

- n CPUs / Cores / Nodes

- $T_n(n)$ Runtime with n CPUs / Cores / Nodes

$$T_n(n) = T_s + \frac{T_p}{n}$$

$$T_n(n) = (1 - f)T_1 + \frac{f}{n}T_1$$

- S(n) Speedup with n CPUs

$$S(n) = \frac{T_1}{T_n(n)}$$

$$S(n) = \frac{T_1}{(1-f)T_1+\frac{f}{n}T_1} = \frac{T_1}{T_1((1-f)+\frac{f}{n})}$$

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

Ahmdal's Law

- Speedup S(n) with n processors

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

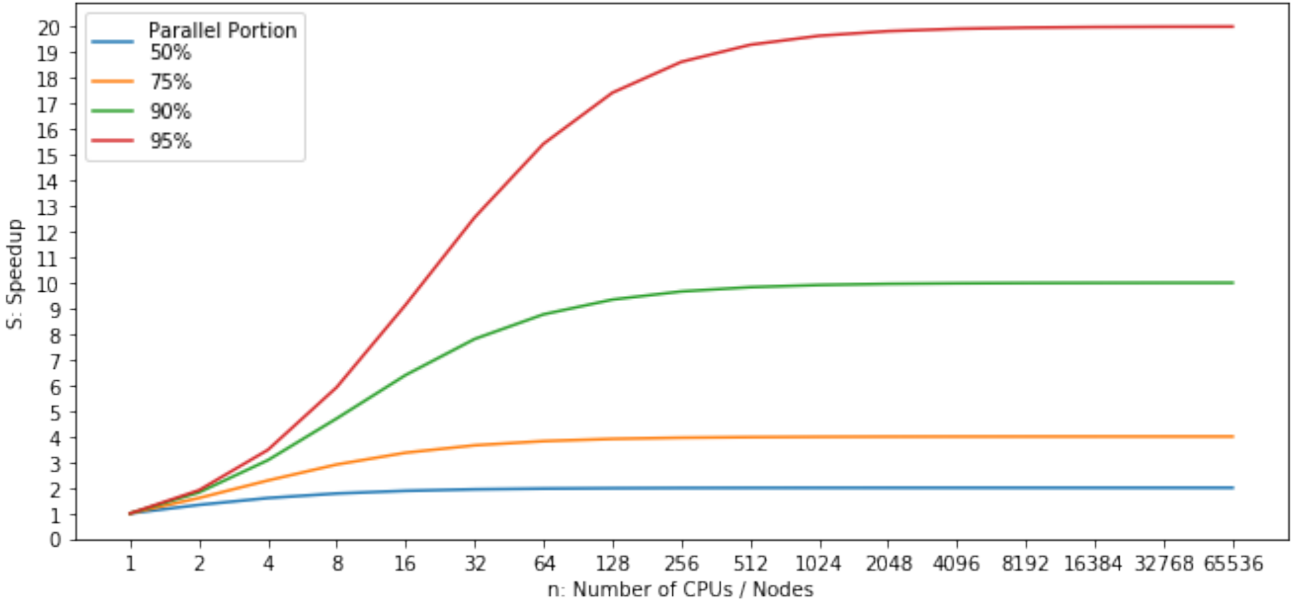
- Ideal Speedup: f=1 ➔ S(n)=n
- Usual f<1 ➔ S(n) is bound by constant

$$\lim_{n \to \infty} S(n) = \frac{1}{1-f}$$

- For f=0.9 and n=10 ➔ S(n)=5.3
- For f=0.9 and n→∞ ➔ S(n) =10

Parallel Speedup

- Implications of Amdahl's Law:
- If the parallel portion of the algorithm is 95% (f=0.95) the maximum speedup is 20



Embarrassingly Parallel Problems

- Little or no dependency and no need for a lot of communication between the parallel tasks
- Amdahl's Law: $f \approx 1$
- Examples:
  - Rendering Computer Graphics
  - K-fold Cross-Validation in Data Mining
  - Testing a Data Mining Algorithm with different parameters
  - Serving static files on a webserver to multiple users at once
  - Computer simulations comparing many independent scenarios, such as climate models
  - Tree growth step of the random forest
  - …