Programming Techniques for Supercomputers:

Basics – Parallelism, Scalability and parallel efficiency

Basic limitations of parallel computing

Prof. Dr. G. Wellein[a,b] , Dr. G. Hager[a]

[a] Erlangen National High Performance Computing Center (NHR@FAU)
[b] Department für Informatik

Friedrich-Alexander-Universität Erlangen-Nürnberg, Sommersemester 2025

Basics: Motivation

- Identify basic limitations of implementations or algorithms for parallel processing

- Assumptions:
  - Underlying hardware is perfectly scalable (no saturation effects etc.)

  - Basic workload may have pure serial and pure parallel contributions

  - **N „workers“** have to perform either
    - Fixed amount of work as fast as possible → Amdahl‘s law
    - Increasing amount of work (~N) in constant time → Gustfson‘s law

- Metrics:
  - Parallel speed-up
  - Parallel efficiency

Basics: Motivation

- Absoulte runtime based view: N workers need $Time(N)$
  - Absolute time to execute $(N = 1)$ workload on one worker: $Time\ (1)$
  - Basic assumption: workload consists of pure serial (s) and perfectly parallelizable (p) „timefraction“

$$Time(1) = Time_s(1) + Time_p(1)$$

Can not be parallelized

Can be perfectly parallelized

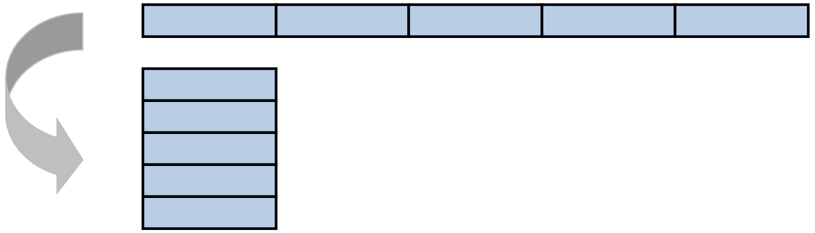
- Relative runtime („fraction“) based view:
  - All runtimes are measured realtive to $Time(1) \rightarrow T(N) = \frac{Time(N)}{Time(1)} \rightarrow T(1) = 1$
  - Serial fraction $s = \frac{Time_s(1)}{Time(1)}$ - parallel fraction: $\mathrm{p} = \frac{Time_p(1)}{Time(1)}$

$$T(1) = 1 = s + p$$

Can not be parallelized

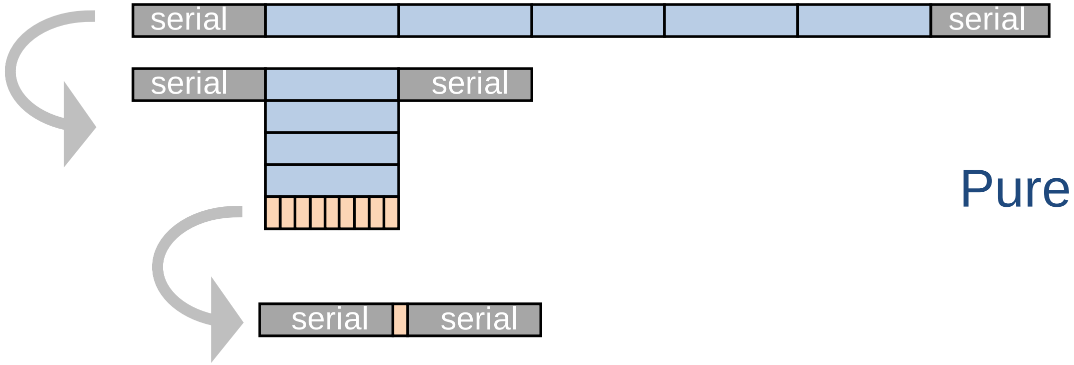
Can be perfectly parallelized

Basic: The ideal world and reality



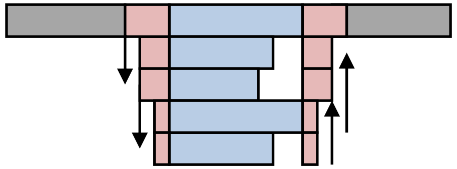
Ideal world

All work is perfectly parallelizable



First correction towards reality

Purely serial parts limit maximum speedup



Reality

Communication / synchronization / load imbalance…

Limitations of Parallel Computing: Metrics to quantify the efficiency of parallel computing

- Assume $T(N)$ is the time to execute „some workload“ with N workers

- How much faster do I execute the given workload on N workers?

  →Parallel Speed-Up: $\boxed{S_P(N) = \frac{T(1)}{T(N)}}$

- How efficient do I use the workers in average?

  →Parallel Efficiency: $\boxed{\varepsilon_P(N) = \frac{S_P(\mathrm{N})}{N}}$

- Warning: These metrics are relative to the time (performance) of a single worker → These metrics are not performance metrics!

Basic limitations of parallel computing

Amdahl’s law (“strong scaling”)

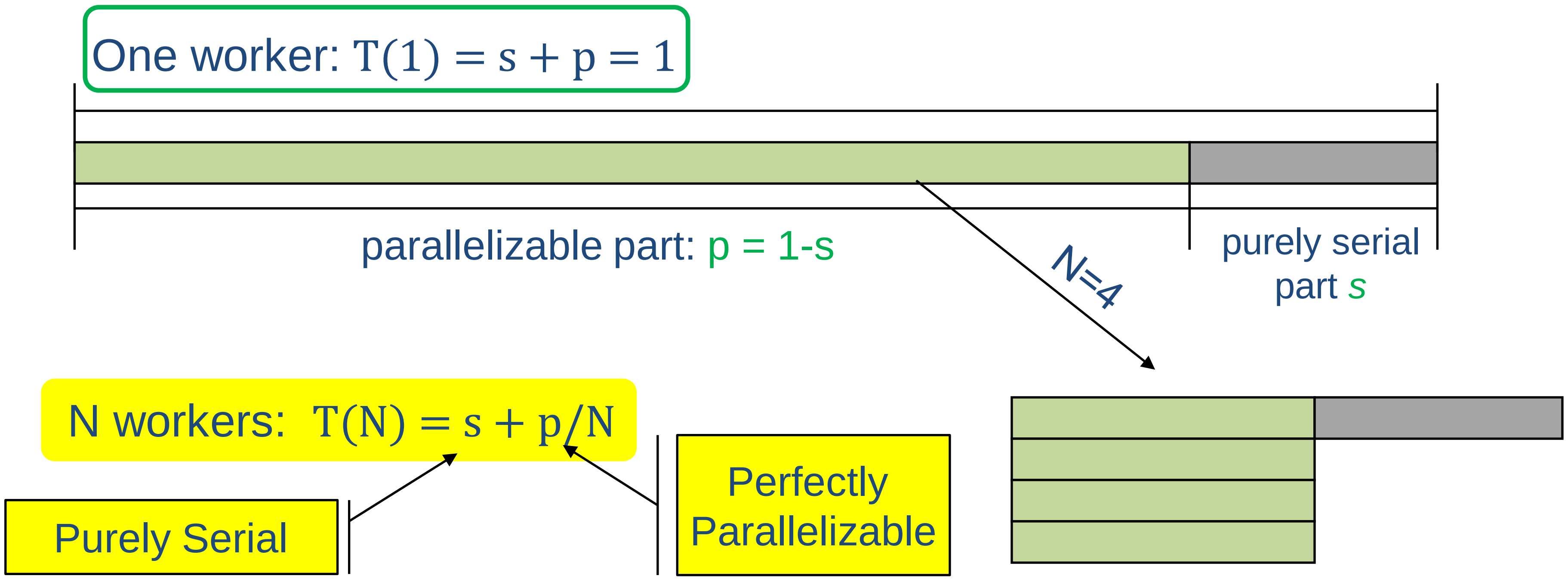
Gustafson’s law (“weak scaling”)

Applying Amdahl’s law

Limitations beyond Amdahl/Gustafson

Limitations of Parallel Computing:
Calculating Speedup in a Simple Model ("strong scaling")

Assumption: Constant workload („strong scaling“)



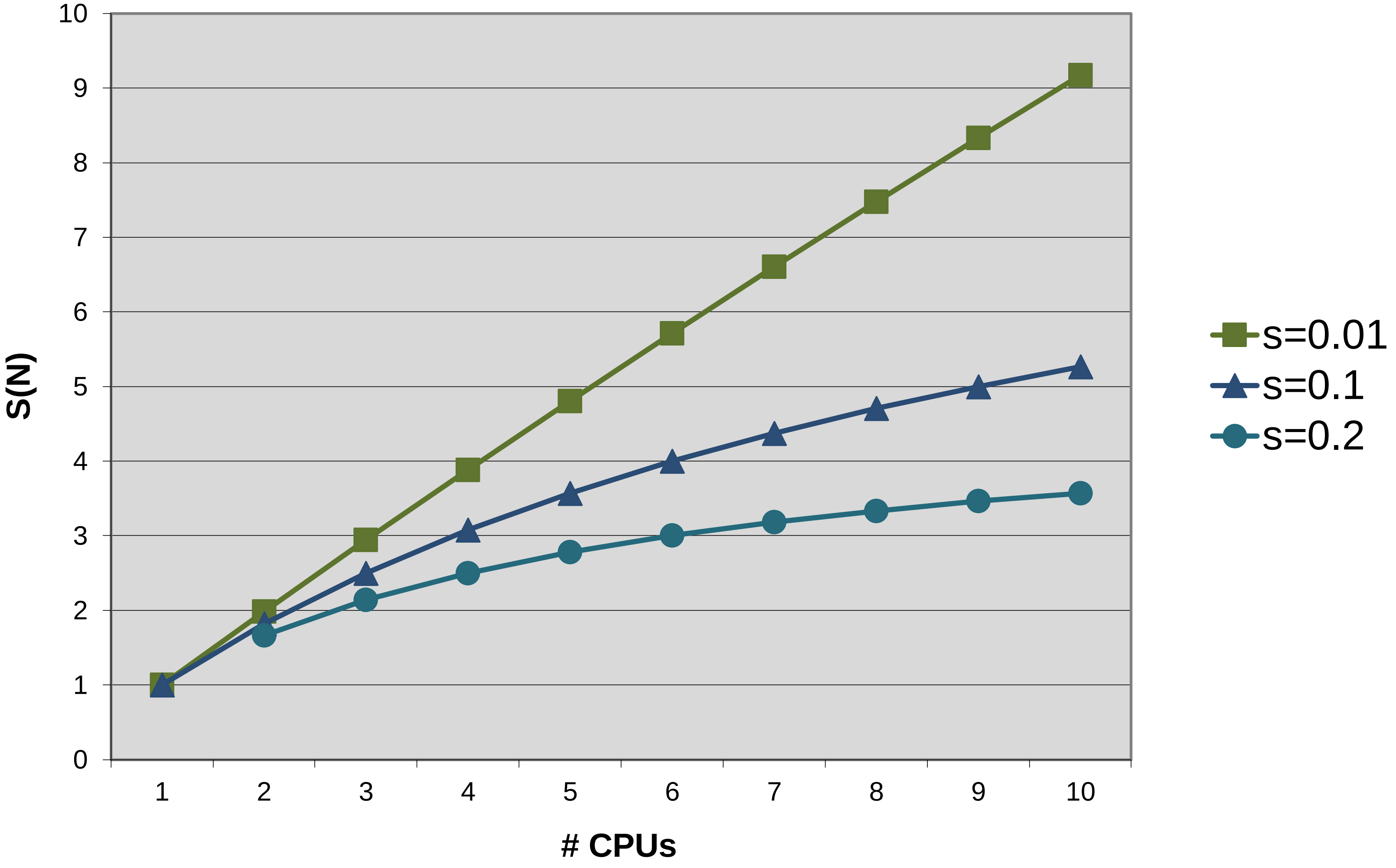
One worker: $T(1) = s + p = 1$

N workers: $T(N) = s + p/N$

→Parallel speedup: Amdahl's Law

$$S_P(N) = \frac{T(1)}{T(N)} = \frac{1}{s + \frac{1-s}{N}}$$

Limitations of parallel computing:
Amdahl´s Law (“strong scaling”)



Limitations of Parallel Computing: Amdahl's Law (“strong scaling”)

- Benefit of parallelization is strongly limited by serial part (s)
  - Maximum Speed-Up which can be attained: $\lim_{N\to\infty} S_P(N) = \frac{1}{s}$
- Parallel Efficiency: $\varepsilon_p = \dfrac{1}{s(N-1)+1}$
  - For large number of workers $\lim_{N\to\infty} \varepsilon_P(N) = 0$
- Reality: No task is perfectly parallelizable
  - Shared resources have to be used serially
  - Task interdependencies must be accounted for
  - Communication overhead (but that can be modeled separately)

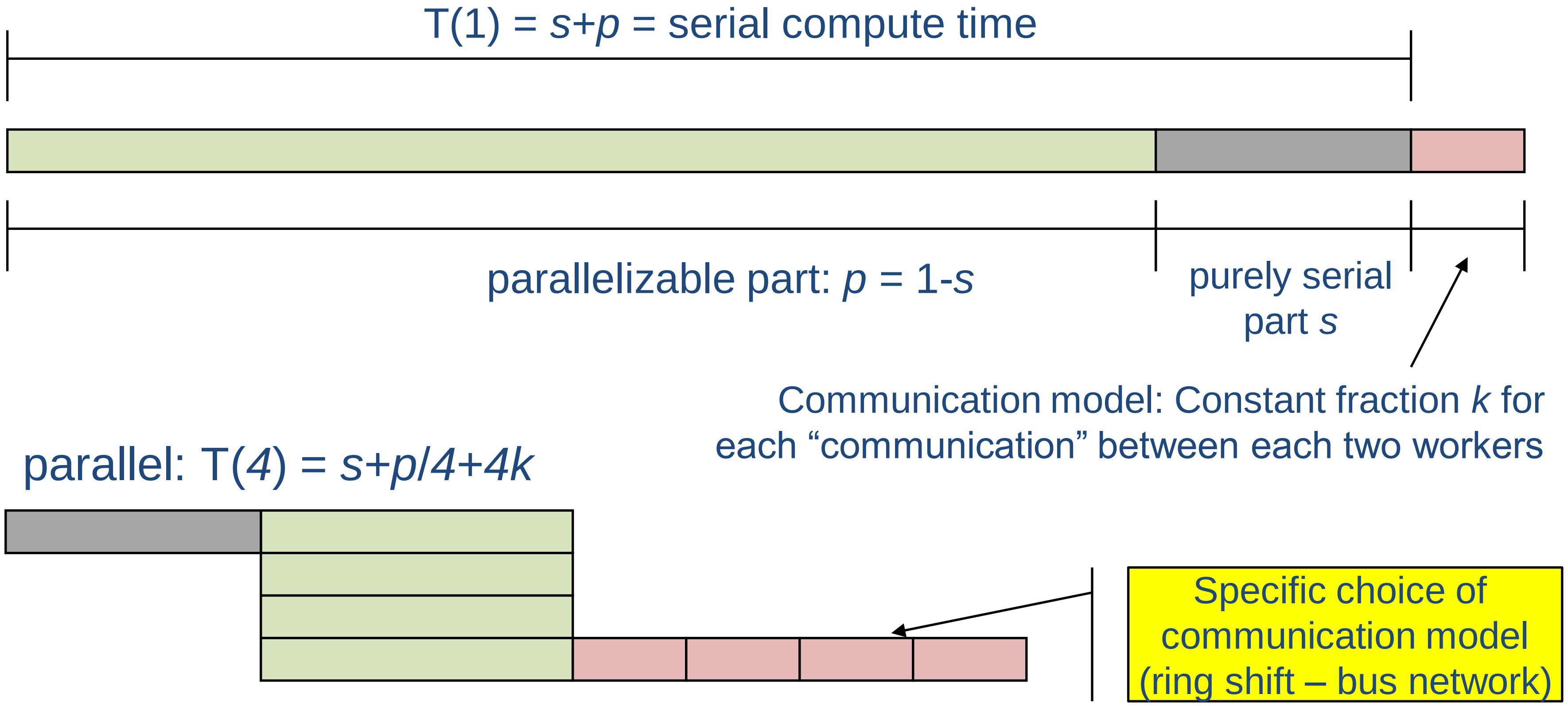
Limitations of Parallel Computing: Extended Amdahl's Law with Communication

- Assume that $c(N)$ is the communication time when using N processors with $c(1) = 0$

→ $T(N) = s + {}^{p}/_{N} + c(N)$

- Communication time may depend on many factors:
  - Network topology
  - Communication pattern
  - Message sizes
  - …
- Typical scaling of communication times:
  - Global communication, e.g. barrier: $c(N) = \mathrm{k}\ \log N$
  - Every process sending message over bus based network or serialization of communication in application code: $c(N) = \mathrm{k\ \ N}$ (see next slide)

Limitations of parallel computing:
Amdahl with (simple) communication Model: Extended Amdahl



T(1) = *s*+*p* = serial compute time

parallelizable part: *p* = 1-*s*

purely serial part *s*

Communication model: Constant fraction *k* for each “communication” between each two workers

parallel: T(*4*) = *s*+*p*/*4*+*4k*

Specific choice of communication model (ring shift – bus network)

Extended Amdahl model
(for specific communication model):
(*k*=0 → Amdahl's Law)

$$S_p^k = \frac{T(1)}{T(N)} = \frac{1}{s + \frac{1-s}{N} + Nk}$$

Limitations of parallel computing: Amdahl’s Law

- Large N limits

  - Amdahl's Law predicts (k=0)

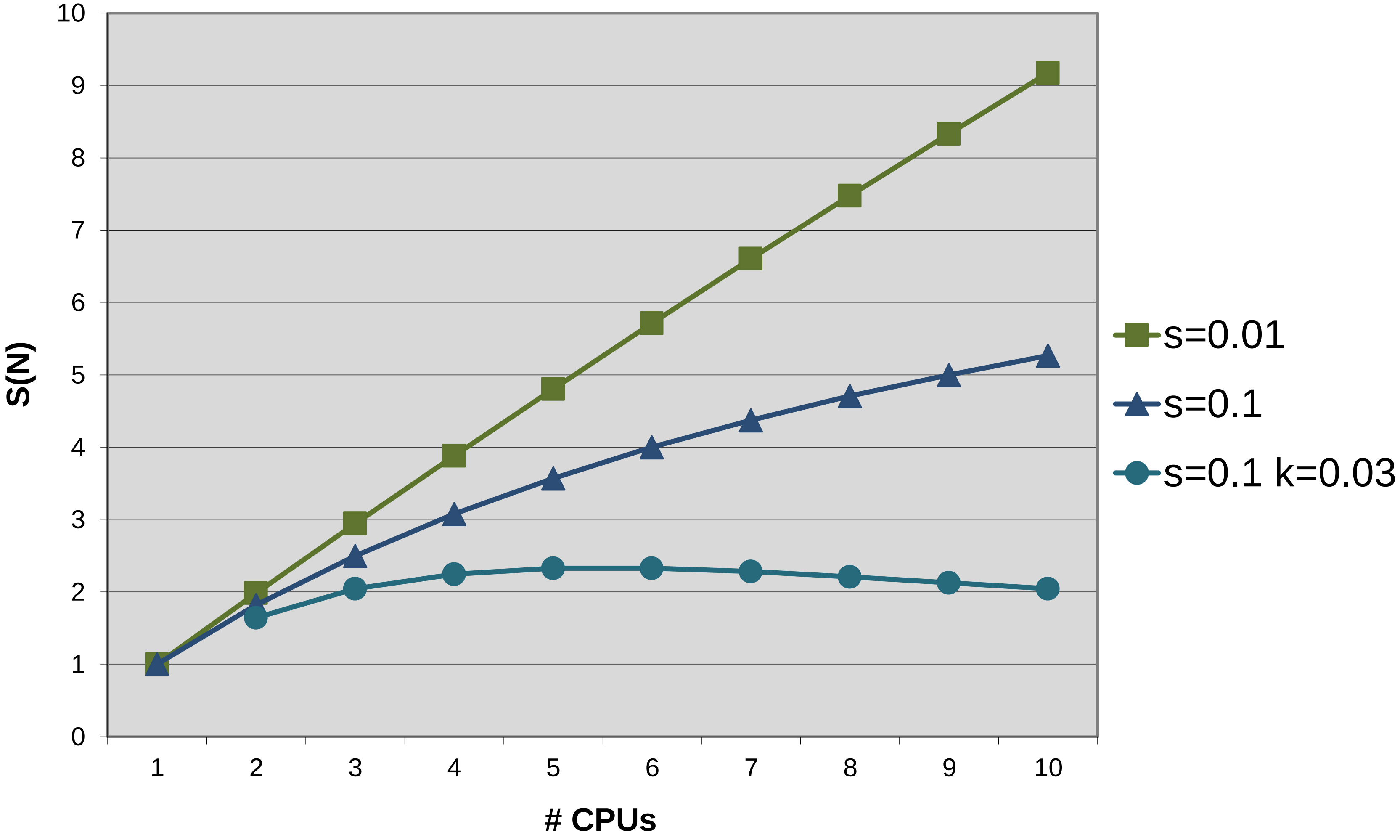
$$\boxed{\lim_{N\to\infty} S_p^0(N) = \frac{1}{s}}$$

(independent of *N*)

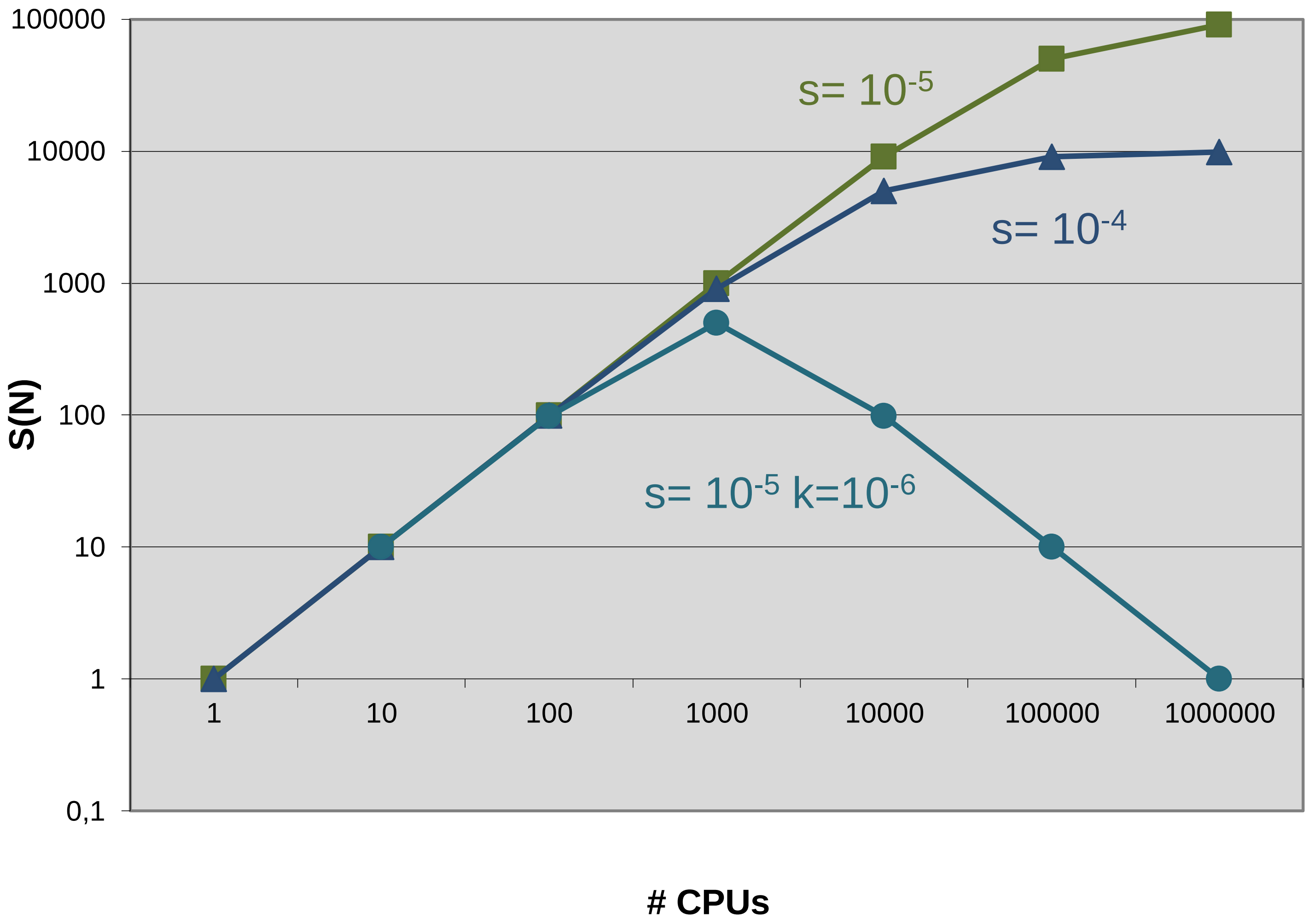
  - At k≠0, our simplified model of communication overhead yields a beaviour of

$$\boxed{S_p^k(N) \xrightarrow{N>>1} \frac{1}{Nk}}$$

Limitations of parallel computing: Amdahl´s Law

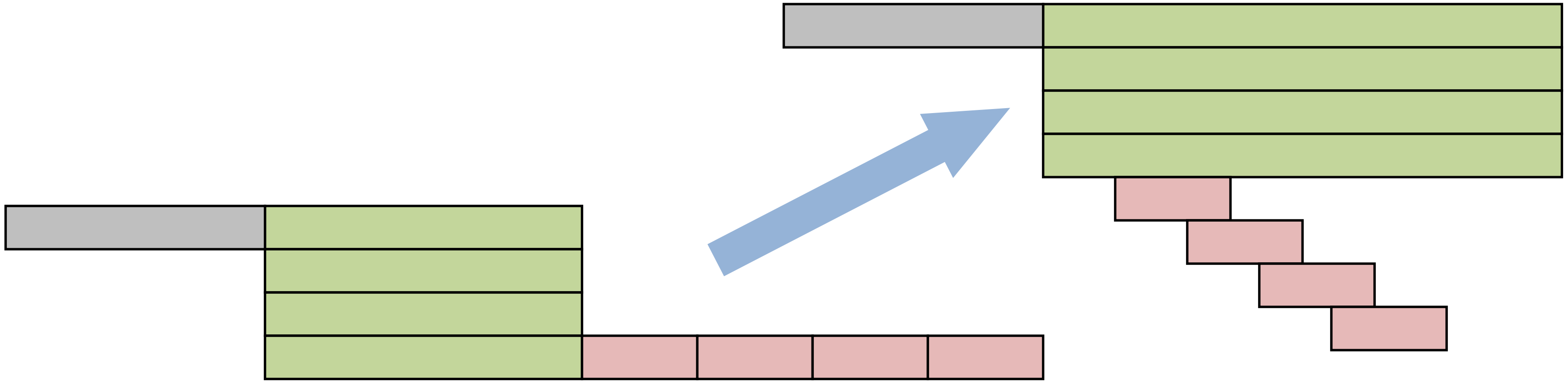


Limitations of parallel computing: Amdahl´s Law at scale



Limitations of parallel computing:
Impact of communication is not always as bad…

- Communication is not necessarily purely serial
  - Non-blocking networks can transfer many messages concurrently – factor *Nk* in denominator becomes *k*, which can be added to s
    (technical measure)
  - Sometimes, communication can be overlapped with useful work ("asynchronous communication"):



- But never forget $$\boxed{\lim_{N\to\infty} S_p^0(N) = \frac{1}{s}}$$

Basic limitations of parallel computing

Amdahl’s law (“strong scaling”)

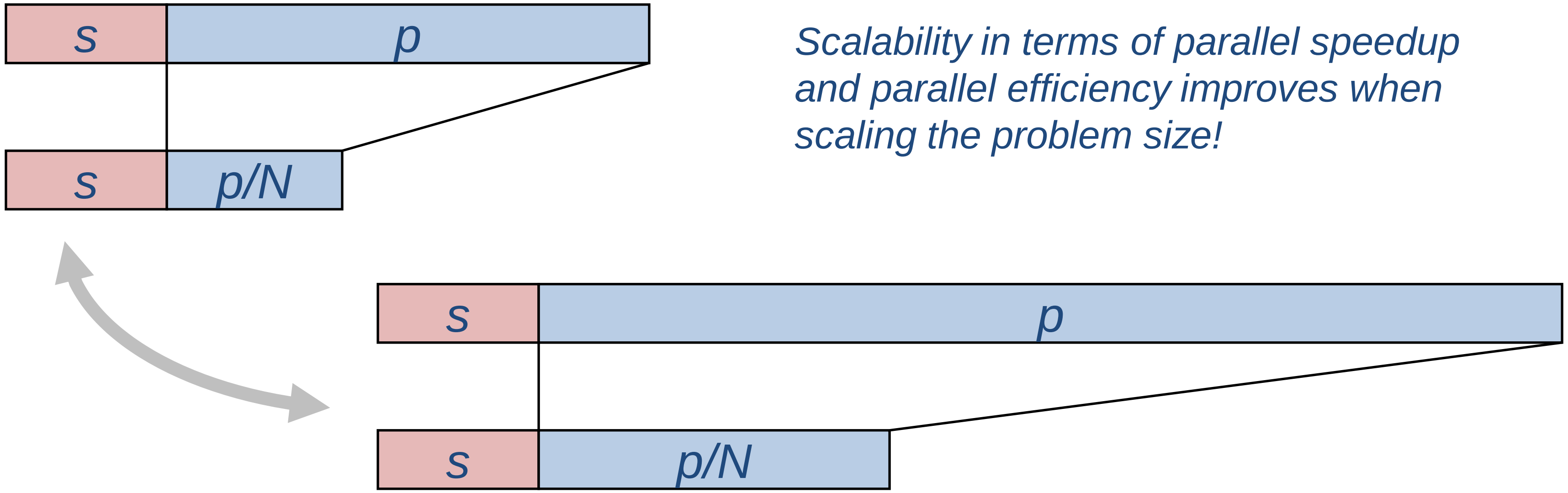
Gustafson’s law (“weak scaling”)

Applying Amdahl’s law

Limitations beyond Amdahl/Gustafson

Limitations of parallel computing: The „weak scaling“ scenario

- Increasing problem size often mainly enlarges „parallel“ workload *p*
  - Then Speed-up increases with problem size



Scalability in terms of parallel speedup and parallel efficiency improves when scaling the problem size!

- For some application fields: Solve problems as big as possible

→ Increase (parallel) workload with available workers / processors

→ This is called „weak scaling“

Limitation of parallel computing: Increasing Parallel Efficiency ("*weak scaling*")

- Assume simple and optimistic scenario: Parallel Workload increases linearly with N, i.e. $p \rightarrow N\,p$

  → $T(N) = s + \frac{N\,p}{N} = s + p$

  → Runtime stays constant if workload is increased linearly with N

  → Performance increases linearly with N

- How long does it take to solve the workload of N processors on 1 processor

  → $T_N(1) = s + N\,p$

$$\boxed{\rightarrow S(N) = \frac{T_N(1)}{T\ (N)} = \frac{s+N\,p}{s+p} = \frac{s+N\,p}{T_S(1)} = s + (1-s)N}$$

Speed-Up increases linearly with N

Gustafson's Law
("weak scaling“ – performance scaling)

Basic limitations of parallel computing

Amdahl’s law (“strong scaling”)

Gustafson’s law (“weak scaling”)

Applying Amdahl’s law

Limitations beyond Amdahl/Gustafson

Limitations of parallel computing:
Applying Amdahl: Serial & Parallel fraction

Always remember model assumptions:

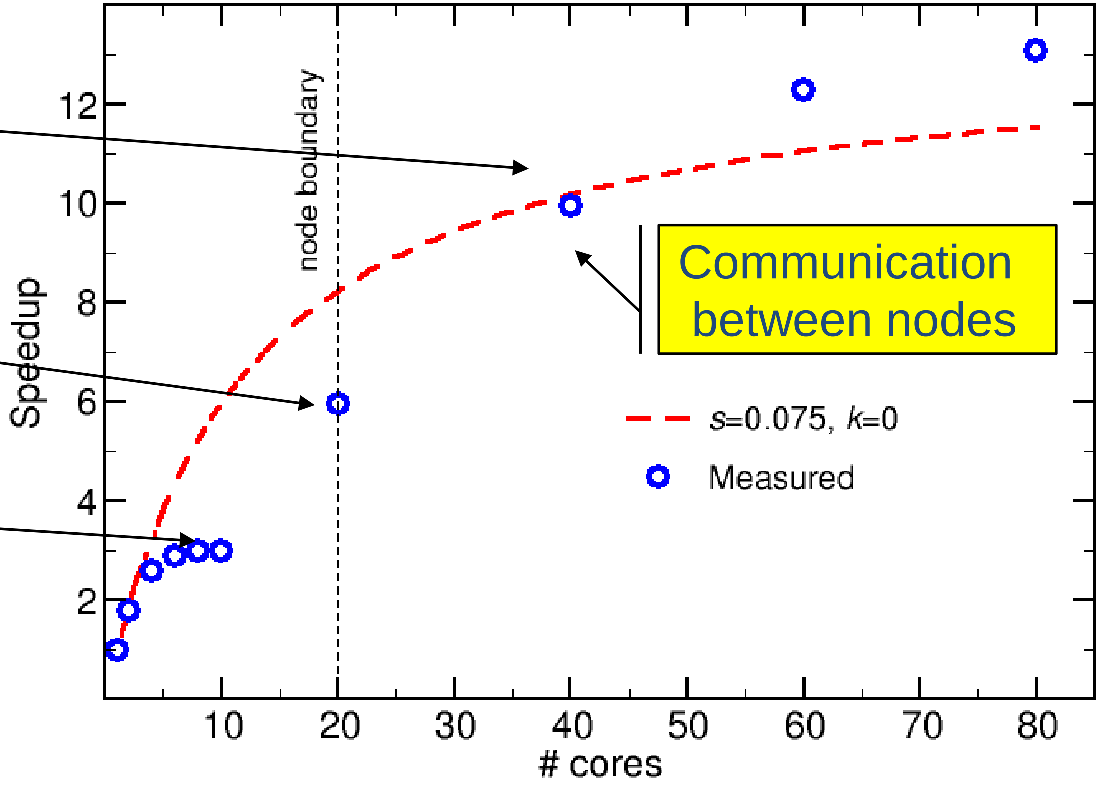
- Workload consists of
  - purely serial (s) and
  - perfectly parallelizable ($p \rightarrow \frac{p}{N}$) parts
- Scalability is limited by
  - serial fraction or
  - communication overhead (extended Amdahl).

- Impact of shared/saturating hardware resources is not modeled

- How to determine model parameters ($\boldsymbol{s}, \boldsymbol{p}$)?
  - First principles: Complete knowledge of application and hardware parameters required – too complex for most applications/kernels
  - Fit model parameters to speedup measurements

Limitations of parallel computing:
Applying Amdahl: Serial & Parallel fraction

- Naïve approach: Measure performance as a function of cores and fit (extended) Amdahl’s law (cf. slide 6/9)
- Hypothetical study on Emmy (i.e. 2-sockets 10 core each per node) – extended Amdahl



Experimental data does not match Amdahl

Communication within node

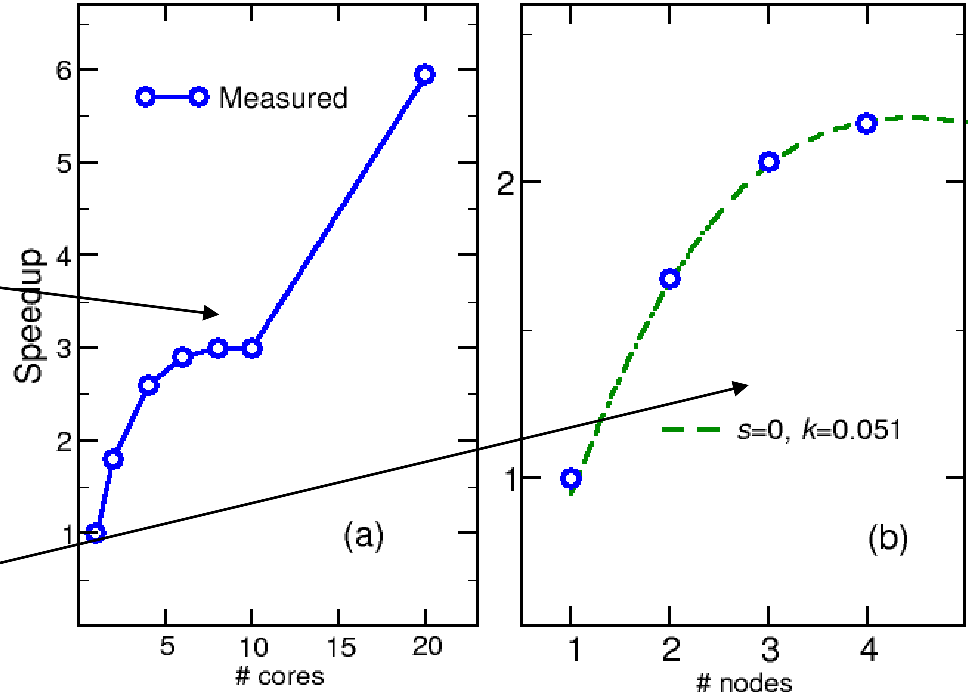
Socket/node scaling limited by bandwidth saturation

Communication between nodes

Multiple impact factors which are not covered by Amdahl!

Limitations of parallel computing: Applying Amdahl: Serial & Parallel fraction

- Better approach: Separation of concerns! Use well-defined basic building blocks as “workers”, which
  - are perfectly scalable (no shared resource in between)
  - restrict measured effects to model assumptions, e.g. use full nodes only (one communication path, serial fraction still visible)



Socket/node saturation to be modeled separately by e.g. ECM model

Amdahl’s modelling serial fraction and one communication speed

Limitations on parallel computing: Applying Amdahl: A more general view

- Amdahl's law can also be interpreted as follows
  - A fraction p of a given code/workload can be "accelerated" by a factor N through some "acceleration technique"
  - The remainder part s cannot be accelerated, i.e. $s + p = 1$
  - "Normalized" runtime of baseline code $T_{base} = 1$ (slide 6: $T(1)$)
  - "Normalized" runtime of accelerated code $T_{acc}(N, s) = s + p/N$ (slide 6: $T(N)$)

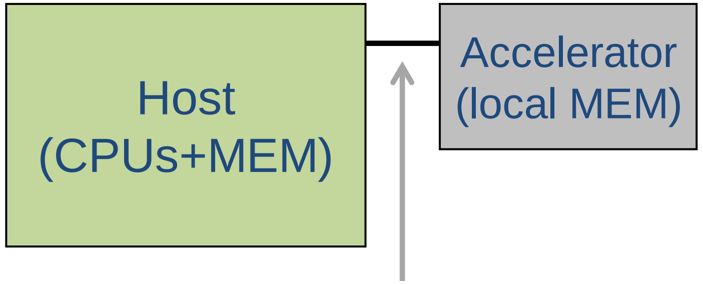
- The speed-up of the acceleration technique is

$$S_p(N) = \frac{T_{base}}{T_{acc}(N, s)} = \frac{1}{s + \frac{1-s}{N}}$$

- Potential "Acceleration factors"
  - Parallel processing with N processes assuming perfect speed-up on fraction p
  - Using an accelerator (e.g. GPGPU) which executes the fraction p of a code N times faster
  - Implementing a code transformation which speeds up a fraction p of a code by N times

Limitations on parallel computing: Applying Amdahl: A more general view

Application: GPGPU accelerated code



Host (CPUs+MEM)

Accelerator (local MEM)

Data exchange (e.g. via PCIe)

- Execution time of original code on host: T_{base}
- ”Accelerated execution” (offload)
  - A fraction p of the original code can be executed on GPGPU *N* times faster than CPU
  - The remaining part s is executed on host

  → $S_p(N) = \frac{1}{s+\frac{1-s}{N}}$

- Consider data transfer between host and accelerator: Extended Amdahl’s law

  → $S_p(N,k) = \frac{1}{s+\frac{1-s}{N}+k}$

Example:

- $T_{base} = 150\ s$
- 75% of that is put on GPGPU →$p = 0.75$
- GPGPU runs 15x faster than host →$N = 15$

→ $S_{0.75}(15) = 3{,}33$

- If total data transfer is 15 s → $k = \frac{15}{150} = 0.1$

→ $S_{0.75}(15{,}0.1) = 2{,}5$

where $k =$ (total data transfer time)/T_{base}

Basic limitations of parallel computing

Amdahl’s law (“strong scaling”)

Gustafson’s law (“weak scaling”)

Applying Amdahl’s law

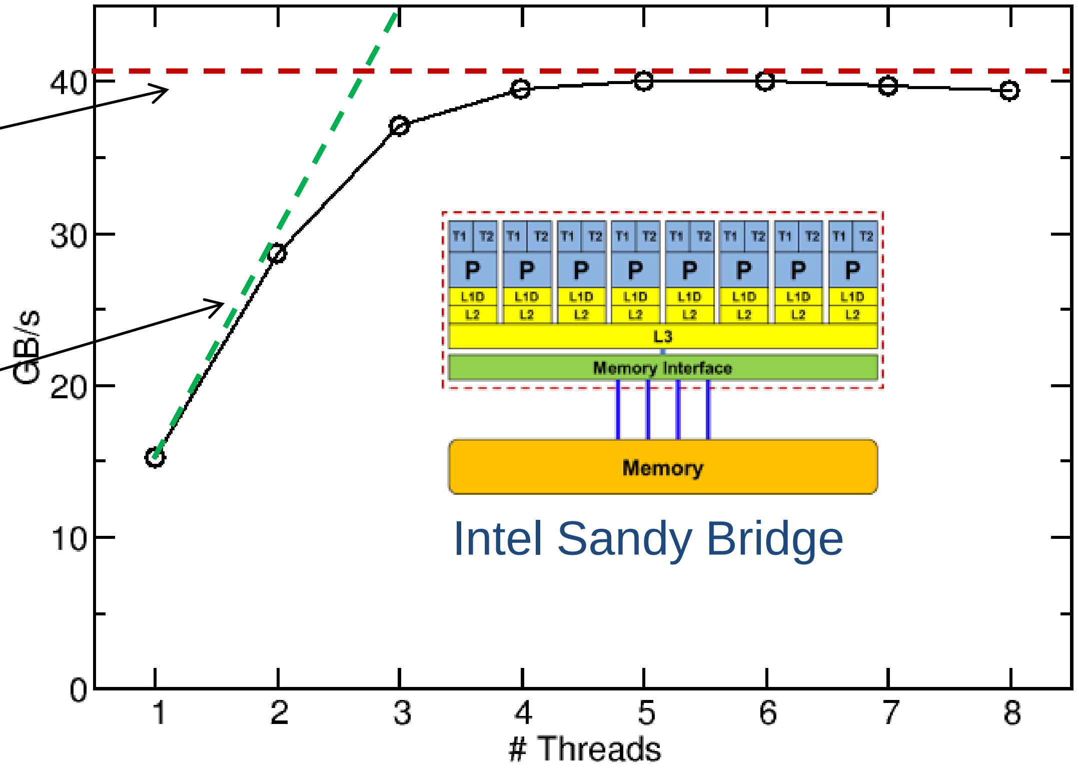
Limitations beyond Amdahl/Gustafson

Limitations of parallel computing – beyond Amdahl/G.
Shared/saturated hardware resources

- Saturations of shared hardware resources set limits to scalability not covered by Amdahl's / Gustafson's law

```
do i= 1 , 10 000 000
   a(i) = b(i) + s * c(i)
enddo
```

Assume perfect parallelization: **s=0**



Intel Sandy Bridge

- Technical limit imposed by hardware (40 GB/s)

- Parallel performance assuming perfect scalability (*p/N*)

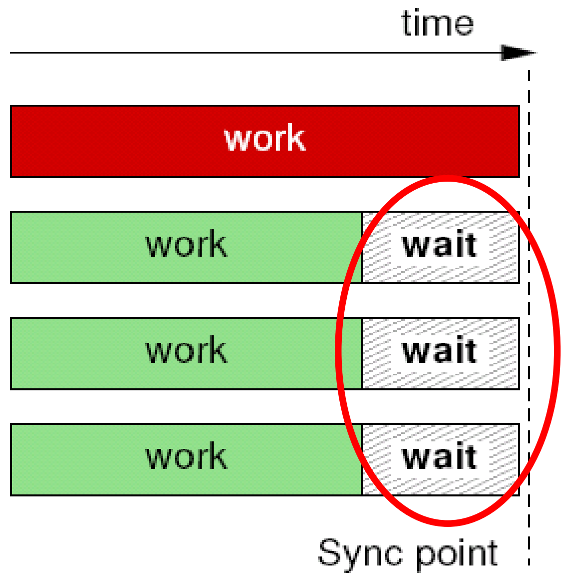
→Parallel scalability limited by saturated hardware resource

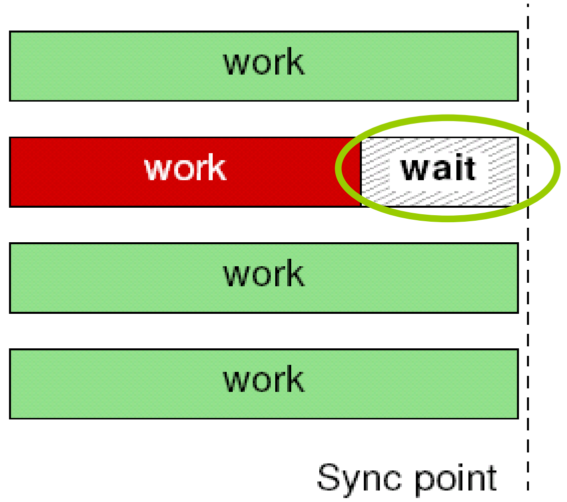
- Other potential HW bottlenecks: QPI, PCIe, networks (see next lecture)

Limitations of parallel computing – beyond Amdahl/G.
Synchronization points and load imbalance

- Load imbalance between “workers”
  → p/N assumption no longer valid (in general)

- Hard to model in a general way, but there are important special cases:
  - A few “laggers” waste lots of resources
  - A single (consistent) lagger could be modeled by increased serial fraction
  - A few “speeders” may be harmless





→ turning some “laggers” into “speeders” may boost performance a lot!