Introduction to High Performance Computing

parallel operations

Example:
1000 nodes with 2GHz, 2 ALUs, 4 processor cores each:

Peak performance = 1000*$[2 * 4 * (2 * 10^9)] = 16{,}000$ GFLOPS

True for peak performance – but what about application performance?

Speedup

Comparision between performance on a single processor (serial, sequential) and parallel performance.
system performance: based on elapsed time
CPU performance: based on user CPU time

Speedup: $S_s(\mathrm{p}) = \frac{T_1}{T_p}$

based on system performance.

Speedup, Efficiency

Efficiency: $E(p) = \frac{S}{p}$

Ideal speedup: S=p

Amdahl‘s Law

Gene Amdahl, geb. 1922, Computer-Architekt und Hi- Tech-Unternehmer. Zu internationaler Bekanntheit kam er durch seine Arbeit im Bereich der Großrechner bei IBM und durch seine später gegründeten Unternehmen, vor allem die Amdahl Corporation.

In every program exist code that is ideal parallelisable (fraction f) and not parallelisable at all (fraction (1-f)). Thus a serial run lasts:

$$T_1 = T_1 \times [(1 - f) + f]$$

Amdahl‘s Law

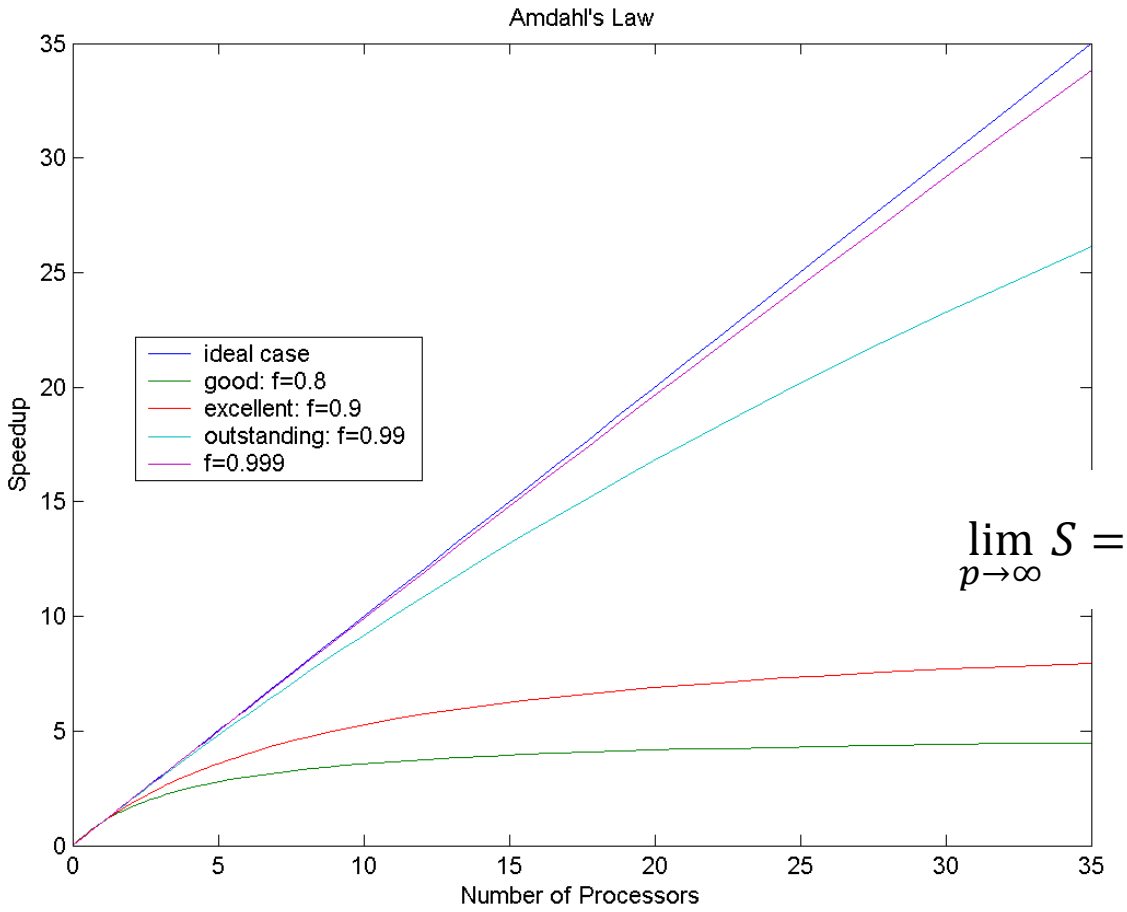
Time for parallel run:

$$T_p = T_1 \times \left[(1-f) + \frac{f}{p}\right]$$

Only the parallel fraction is speeded up by parallelization.

$$S(p) = \frac{T_1}{T_p} = \frac{1}{\left[(1-f) + \frac{f}{p}\right]} = \frac{p}{[(1-f)p + f]}$$

practical aspects



$$\lim_{p\to\infty} S = \frac{p}{[(1-f)p+f]} = \frac{1}{1-f}$$

practical aspects

Numerical integration on interval [0,1] of the function

$$f(x) = \frac{4}{1+x^2}$$

The integration interval is subdivided into N pieces of size

$$h = \frac{1-0}{N}.$$

Approximation of the integral by the sum:

$$Int = h\sum_{i=1}^{N} f(x_i)$$

at points $x_i = 0 + ih$.

practical aspects

Each evaluation of $f(x_i)$ may be performed independently in parallel.

- Calculation of $f(x_i)$ is the f-fraction in Amdahl's law. It lasts appr. 10xN operations.
- Calculation of the sum and multiplication at h is the (1-f)-fraction. It lasts p operations (may be $\log_2 p$)

How large is the best achievable speedup according to Amdahl's law?

Answers:

- for small N, the linear increase with p prevents speedup
- for large N, the speedup is almost ideal

poor granularity

Measuring performance

Various characteristics require different benchmarks

- Close to CPU-Peakperformance: LINPACK
  - Basis of the TOP500 list
- Network performance tests
  - Intel MPI Benchmark
  - cbench
- Filesystem benchmarks
  - bonnie
  - iozone
- Performance of memory system
  - stream benchmark
- Application benchmarks
  - NAS Parallel
  - SPEC benchmarks
  - …..

Measuring performance

Ideas to your own benchmark:

1. What should be measured?
    1. How long will it last?
    2. Which side effects are possible?
2. Measuring (elapsed) time
    1. Which resolution has the measurement?
3. Statistical average to exclude unforeseen side effects

Scalability

Numerical integration on interval [0,1] of the function

$$f(x) = \frac{4}{1 + x^2}$$

The integration interval is subdivided into N pieces.

- The integral is calculated in 10s on 4 processor kernels.
- Accuracy is not enough – need to double N.
- Increase in calculation time is not tolerable on 4 processor kernels.
- Would 8 processors do the job in 10s?

Does the speedup scale with the number of processors?

Gustafson‘s Law

John Gustafson, geb. 1955, ist mit seinen Arbeiten bei Intel Labs-SC, Massively Parallel Technologies, Inc. und Clearspeed Technologies Spezialist für das massive Parallelrechnen und den Einsatz und Entwicklung von Beschleunigerkarten.

$$T_p = T_p * [(1 - f) + f]$$

Looks like Amdahl‘s law, but from the opposite site.

Gustafson‘s Law

Let’s consider an algorithm with

- $\mathcal{O}(n^4)$ in an ideal parallelisable region (1h)
- $\mathcal{O}(n)$ in a not at all paralleilisable region (12min)
- total time T_1: 72 min
- 16 procs: 15.75 min
- Speedup (strong scaling): 4.6

  The speedup is poor.

- doubling problem size
- time now (T_1) : 16h 24min = 984min
- 16 procs: 1h 24min

- doubling problem size and increasing the number of procs according to the complexity gives (weak scaling):

$$S = 16 * \frac{1h12min}{1h24min} = 13.7$$

  Problem scales well!

Other prediction models

PRAM – (parallel random access machine)
unrestricted, conflict free access to main memory through all processors

simple timing model –

- uniform processing time for all processors
- no consideration of memory accesses
- linear time dependency for communication

simple performance model (Schüle) –

- addition of memory access times

SPM

Summation of two vectors

```
for(i=0;i<N;i++) z[i]=a*x[i]+b*y [i];
```

Which performance may we expect for this loop (sequentially)?
How to parallelize this loop?
Which performance may we expect in parallel?

SPM

Which performance may we expected for this loop (sequentially)?

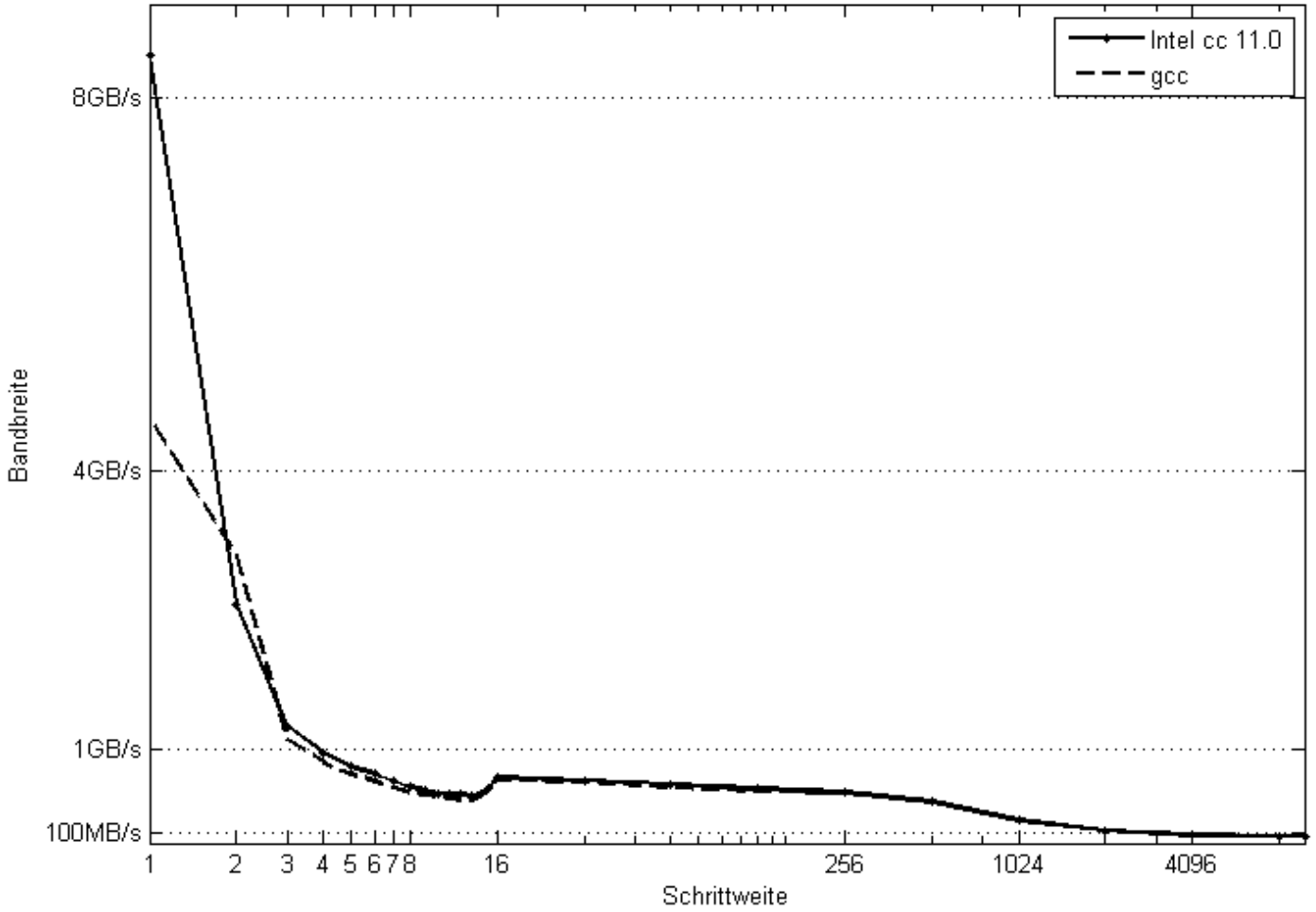
Assumptions:
1. Time for a single arithmetic operation: Ta
2. Time for a single memory operation: Tm

Relation between Tm and Ta?

Ta may be approximated by peak performance.
Tm may be approximated by memory benchmark

- What is a benchmark?
- What aspects have to be considered?

SPM


Intel cc 11.0
gcc
8GB/s
4GB/s
1GB/s
100MB/s
Bandbreite
1
2
3
4
5
6
7
8
16
256
1024
4096
Schrittweite

SPM

$$t_m(S, L) = t_l + t_b \times S \times L$$
$$t_m(\infty, L) = (t_l + t_b) \times L$$

Relation:

$$\frac{t_m(S,L)}{t_a} = 2 \times S \times L \text{ for } S \leq 16$$
$$\frac{t_m(\infty,L)}{t_a} > 160 \times L \text{ for } S \gg 16$$

SPM

Summation of two vectors

```
for(i=0;i<N;i++) z[i]=a*x[i]+b*y [i];
```

Time for arithmetics: $N \times 3 \times T_a$

Time for memory (3 vectors): $3 \times 2 \times 1 \times N \times T_a$

Sequential time: $9 \times N \times T_a$

Peak time (theoretical 2 Ops per cyclce): $1.5 \times N \times T_a$

Efficiency: $1.5 \div 9 = 17\%$

SPM

Summation of two vectors

```
for(i=0;i<N;i++) z[i]=a*x[i]+b*y [i];
```

Sequential time: $9 \times N \times T_a$

Parallel time: $9 \times N_p \times T_a$ with $N_p = \max_p$ *their piece of* N

Parallel time depends on an equal distribution of the work

load balancing: The workload has to be distributed according to the computational capabilities to achieve good parallel performance.

SPM

inner product:

```
for(i=0,s=0.;i<N;i++) s+=x[i]*y[i];
```

algorithmic?

trivial:

```
s=0.;
for (i=0;i<Np;i++) s += x[i]*y[i];
for(i=0;i<p;i++) if(i!=myself) send (s to proc. i);
for(i=0;i<p;i++) if(i!=myself) s += receive(proc. i);
```

communication time?

$$T_c(L) = T_s + L \times T_b with\ T_b = {}^{1}/_{Bandwidth}$$

```
s=0.;
for (i=0;i<Np;i++) s += x[i]*y[i];
for(i=0;i<p;i++) if(i!=myself) send (s to proc. i);
for(i=0;i<p;i++) if(i!=myself) s += receive(proc. i);
```

communication time: $(p-1) \times T_c(1)$

not the bandwidth but the startup time (network latency) is important

```
s=0.;
for (i=0;i<Np;i++) s += x[i]*y[i];
for(i=0;i<p;i++) if(i!=myself) send (s to proc. i);
for(i=0;i<p;i++) if(i!=myself) s += receive(proc. i);
```

sequential time (2 vectors):: $(2 \times N + 4 \times N) \times T_a$

parallel arithmetic time: $(6 \times N_p + (p-1)) \times T_a$

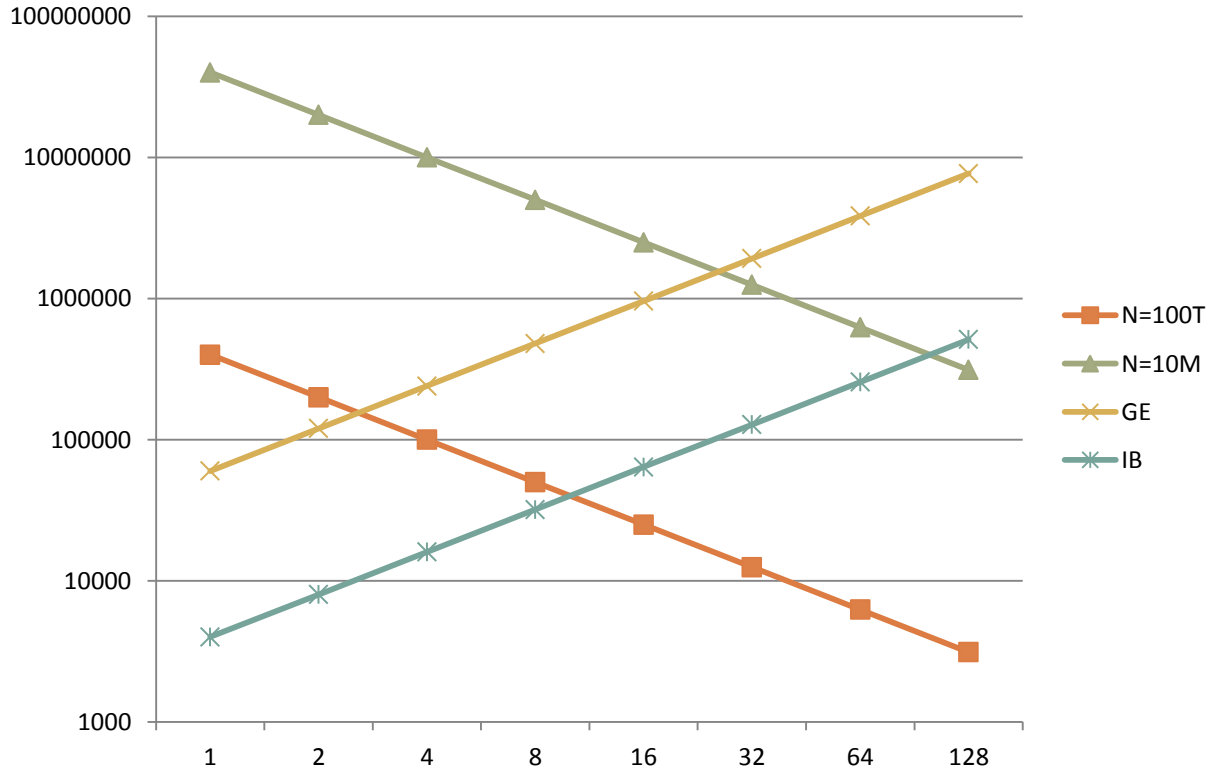
communication time: $(p-1) \times T_c(1)$

How large is $T_c(1)$?

Send a simple `ping`
In 1GE-network ca. 0.3ms

$$T_c(1) > 6 \times 10^5 \times T_a$$

$$T_p \approx (6 \times N_p + p \times 6 \times 10^5) \times T_a$$



Conclusions:

- network latency is very important
- a large granularity helps
- poor algorithm - linear dependency on p

measure for work to be done without communication

SPM

inner product:

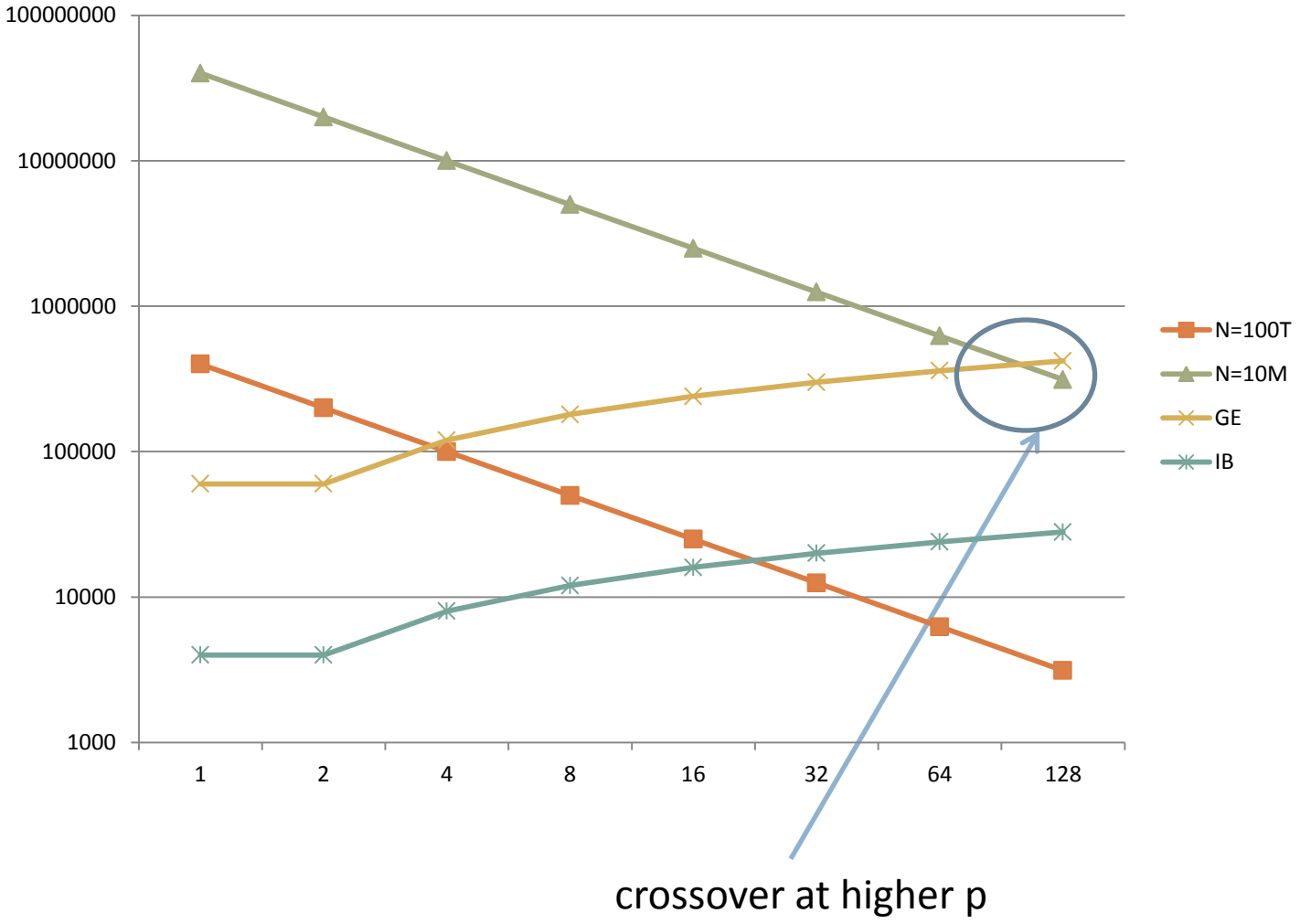
```
for(i=0,s=0.;i<N;i++) s+=x[i]*y[i];
```

improved:

```
s=0.;
for (i=0;i<Np;i++) s += x[i]*y[i];
if(p==odd) send (s to proc. p-1);
if(p==even) s += receive(proc. p+1);
if(p%4==2) send (s to proc. p-2);
if(p%4==2) s += receive(proc. p+2);
…
```

communication time?

$$T_{comm} = \log p \times T_c(1)$$


100000000
10000000
1000000
100000
10000
1000
1
2
4
8
16
32
64
128
N=100T
N=10M
GE
IB
crossover at higher p

Einführung in das Hochleistungsrechnen

Introduction to High Performance Computing

VIELEN DANK
THANK YOU