Parallel Programming

pauldj@aices.rwth-aachen.de

Time

- **Wall time** or “wall-clock time”: real time between the beginning and the end of a computation
- $T_p(n)$:= Wall time to solve a problem of size n by p processes
- **CPU-time** or “core time”: cumulative time spent by all processes in a computation

Performance

- **Performance**: Number of floating point operations per second performed while solving a given problem
- **Theoretical Peak Performance** (TPP): In ideal conditions, the highest number of floating point operations that a processor can perform in one second
- **Peak Performance** “Practical peak performance”: The performance attained by highly tuned matrix-matrix multiplication kernels
- **Efficiency**: The ratio between the performance attained while solving a given problem and the TPP

Scalability

- **Speedup**: $S_p(n) := \dfrac{T_1(n)}{T_p(n)}$ Typically: $0 \leq S_p(n) \leq n$

  When $S_p(n) > n$: "superlinear speedup"

- **Parallel efficiency**: $E_p(n) := \dfrac{S_p(n)}{p}$

- **Strong Scalability**: Behaviour of $T_k(n)$, as k increases (Fixed problem size, increasing number of processes)

- **Weak Scalability**: Behaviour of $T_k(m)$, as m and k increase so that the load per process stays constant. (Fixed load per process, incresing problem size and number of processes)

Amdahl's law

It quantifies the maximum possible speedup when only a portion of the code is improved.

- T_{seq}: strictly sequential portion of the algorithm (in secs)
- T_{par}: parallel portion of the algorithm (in secs)
- $T_1(n) == T_{\text{seq}} + T_{\text{par}}$
- β: fraction of the algorithm that is strictly sequential
- $\beta == \dfrac{T_{\text{seq}}}{T_{\text{seq}} + T_{\text{par}}}$
- $T_p(n) == \beta T_1(n) + (1-\beta)\, T_1(n)/p == T_1(n)\left(\beta + \frac{(1-\beta)}{p}\right)$
- $S_p(n) == \frac{T_1(n)}{T_1(n)\left(\beta + \frac{(1-\beta)}{p}\right)} == \frac{1}{\beta + (1-\beta)/p}$ $\qquad \lim_{p\to\infty} S_p(n) = 1/\beta$