RUPRECHT-KARLS-
UNIVERSITÄT
HEIDELBERG

Introduction to High Performance Computing

Lecture 03 – Applications, Performance Increase, Top500

Holger Fröning

Institut für Technische Informatik

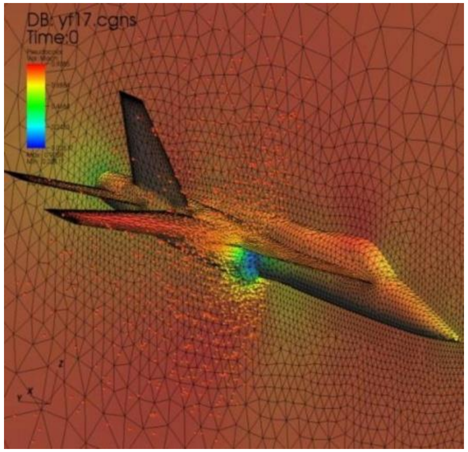
Universität Heidelberg

RUPRECHT-KARLS-
UNIVERSITÄT
HEIDELBERG

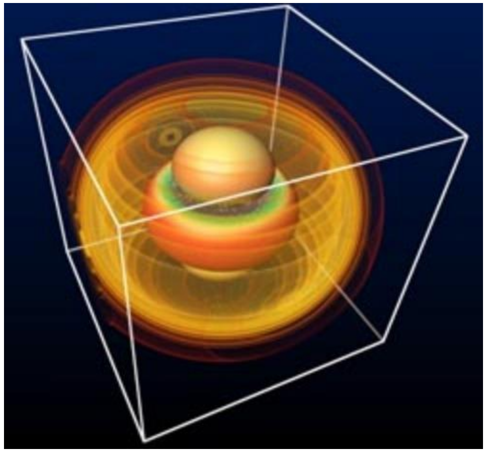
Example Applications Fields

Example application fields

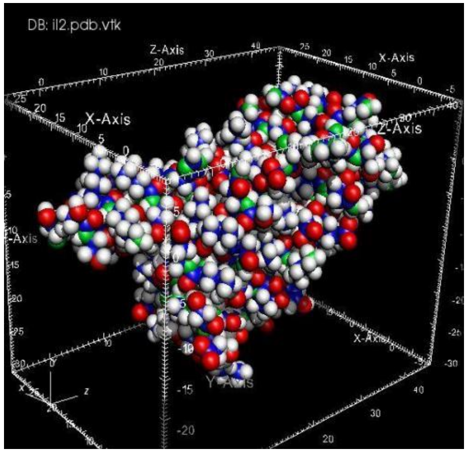
- Oil and gas
  - Seismic processing
- Financial services
  - Automated option pricing and trading
- Bioscience
  - Genetic sequencing and chemistry
- Government
  - Searching and encryption engines
- Digital content creation
  - Movie animation
- Scientific research
  - Astrophysics, particle physics
  - Biology: molecular dynamics
- Industry
  - Fluid dynamics
- Meterology & Climatology
- Electronic design automation
  - Chip design
- Finite element methods
  - Crash simulations
- …



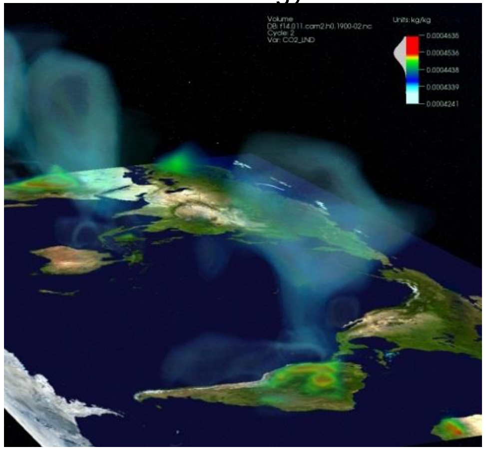
CFD



Cosmology



Molecular Dynamics (MD)

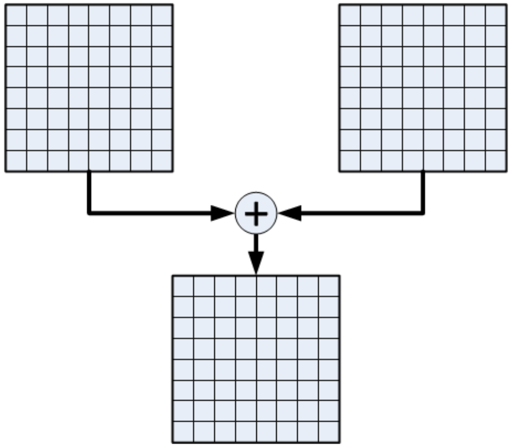


Weather/Climate Research

A more abstract view…

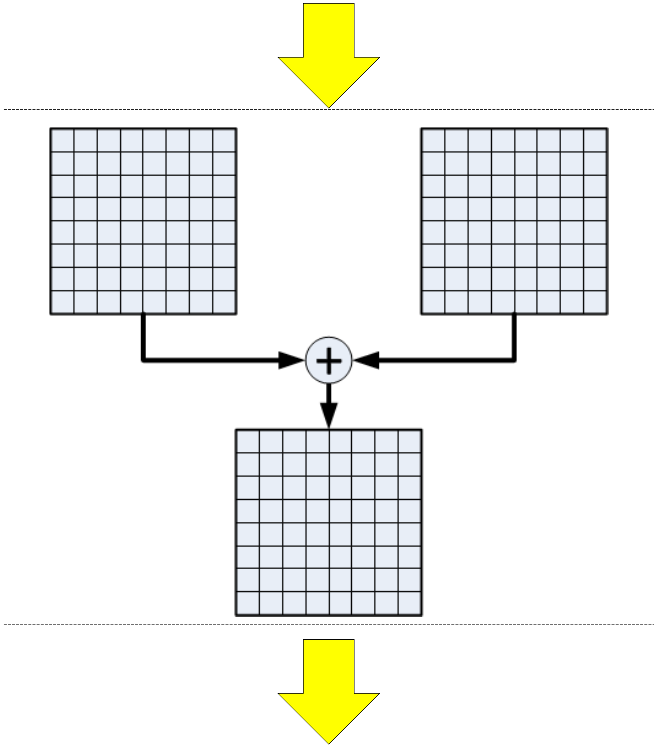
- Basic operation types in HPC
  - Complex processing of regular data structures
    - Vectors
    - Arrays
    - Elements
  - High degree of parallelism

- Trivial example: Matrix addition
  - “Domain decomposition”

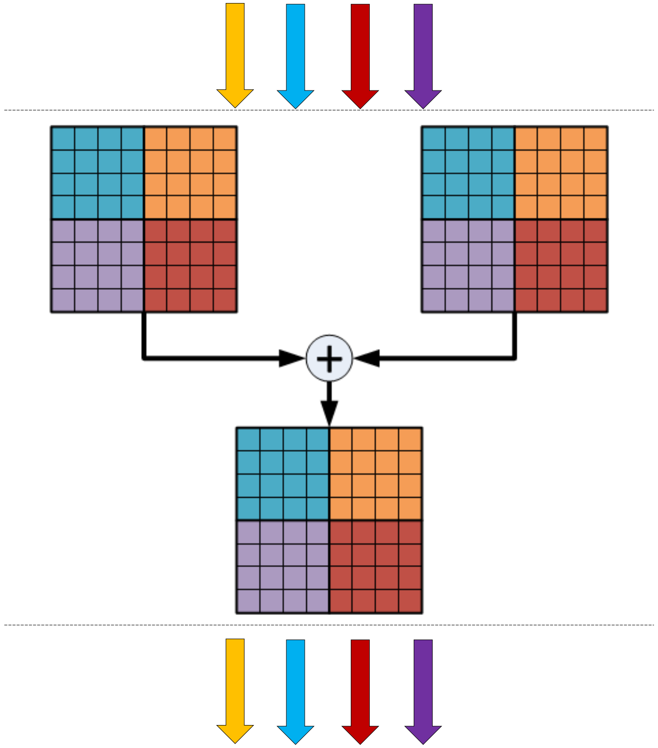


Simple parallelization example

- Serial execution
  - No communication
  - Sequential processing of elements

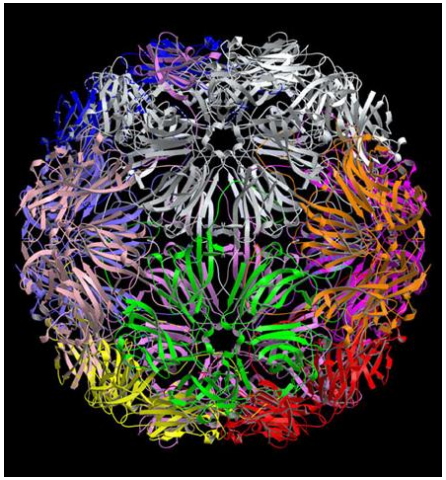


- Parallel execution
  - Communication?
  - Synchronization?
  - Parallel processing of (blocks of) elements

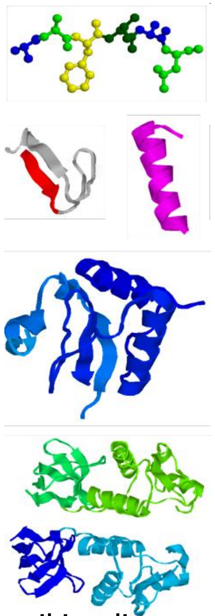


Molecular dynamics

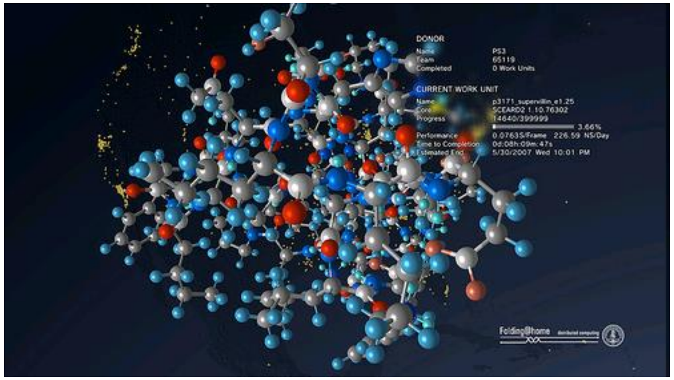
- Motivation
  - Protein Folding
  - Digital simulation instead of biochemics
  - Computationally intensive
  - Runtime of several months
  - STMV: 160 genes, 100ns/day for Petascale-class
- Goal: Calculation of the molecule's shape
  - Double precision floating point
  - Calculation of forces in between the atoms of the molecule and the surrounding
    - Forces: Electrostatic (Coulomb) & Van der Waal
  - Time step = 1 femto (10^{-15}) second
- "N-body problem"



nasa.gov



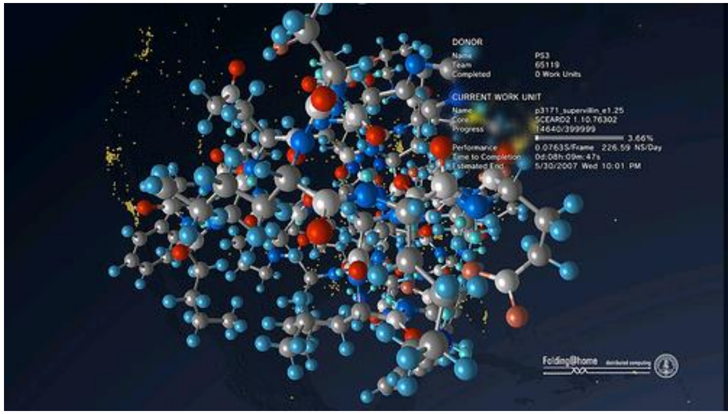
wikipedia.org



Molecular dynamics: N-body

- N-body problem
  - For each atom in a 3D system

```
Do repeat
   Increase time step t
   Foreach atom i
      Foreach atom j (j != i)
         Compute force(s) from j to i
         Sum all forces on i
      Next j
      Compute next position of i
   Next i
Repeat until stable
```



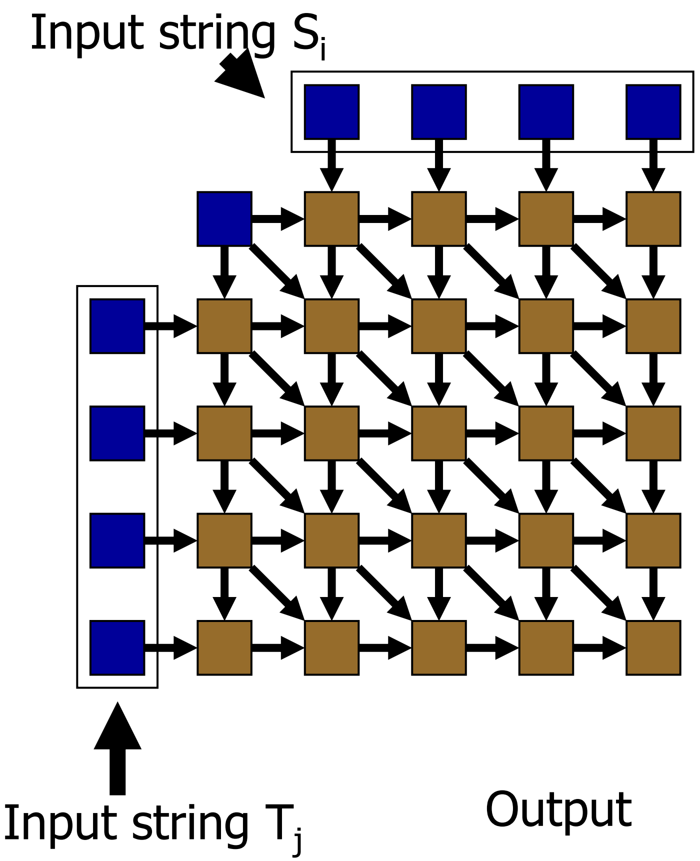
forces ~ $1/d^2$
~82% time
~52 FP ops

  - No special treatment of borders here…
    - Approx. 60 variants

RUPRECHT-KARLS-UNIVERSITÄT HEIDELBERG

Bioscience

- Genetics related research
- String distance computation
- Using Smith-Waterman
  - Exact string matching algorithm
  - Finds optimal local alignment
  - Computes a matching score ***H(i,j)*** of two input strings ***S*** and ***T*** using a 2D matrix
- Other applications:
  - Motif discovery, data mining

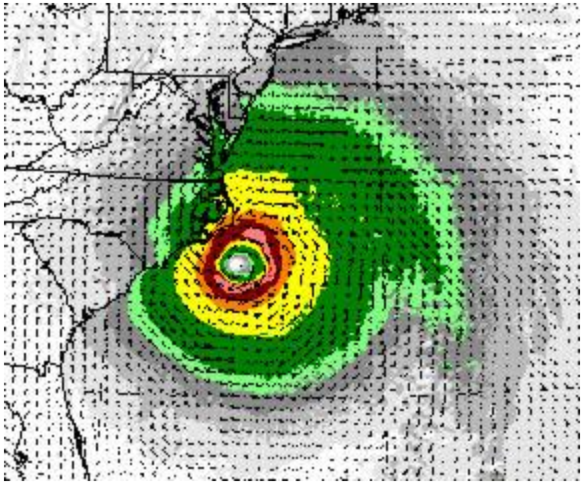


$$H(i,j) = \max \left\{ \begin{array}{cc} 0 & \\ H(i-1,j-1) + \ w(a_i,b_j) & \text{Match/Mismatch} \\ H(i-1,j) + \ w(a_i,-) & \text{Deletion} \\ H(i,j-1) + \ w(-,b_j) & \text{Insertion} \end{array} \right\}, \ 1 \le i \le m, 1 \le j \le n.$$

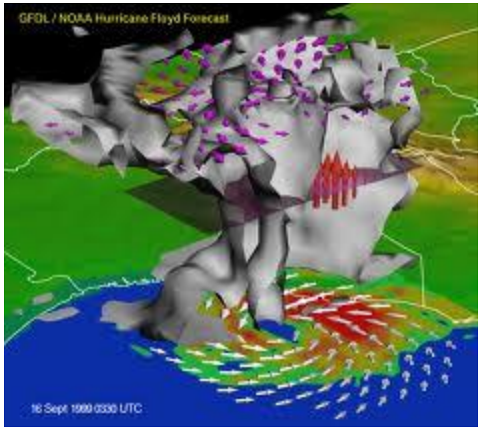
Computational Fluid Dynamics (CFD)

- Numerical fluid simulations
- Lattice-Boltzmann method (LBM)
  - Simulation at particle scope
  - Discretization by grid, at microscopic level solution of mutual reaction as described by the Boltzmann equation
- Boltzmann equation in the case of large average free path lengths
  - Currents in (depleted) gases
  - Neutron distribution in atomic reactors
- Otherwise: E.g. Navier-Stokes equation
  - Much simplier
  - Current in liquids
- Rather few communication, large messages between pairs of two

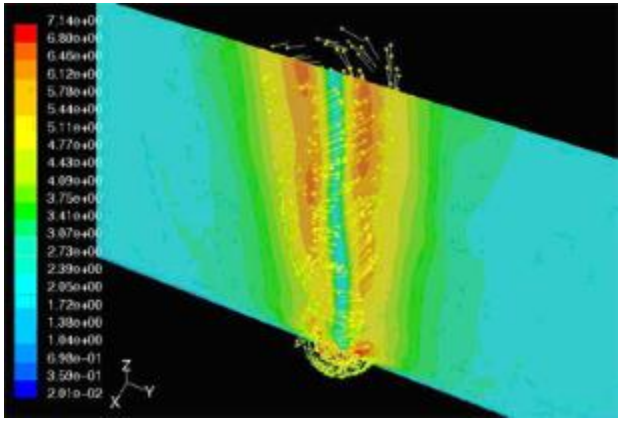
Weather and Climate Research



ucar.edu

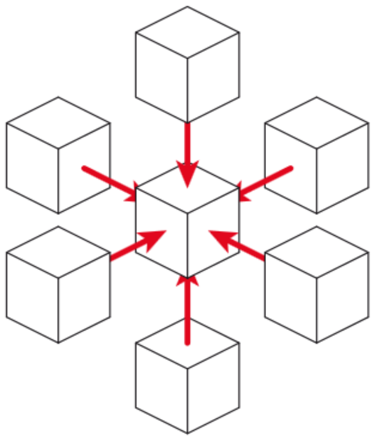


- See CFD
- Examples
  - Hurricane prediction
  - Tornado simulation
  - Local weather forecast
  - Global Climate Modeling

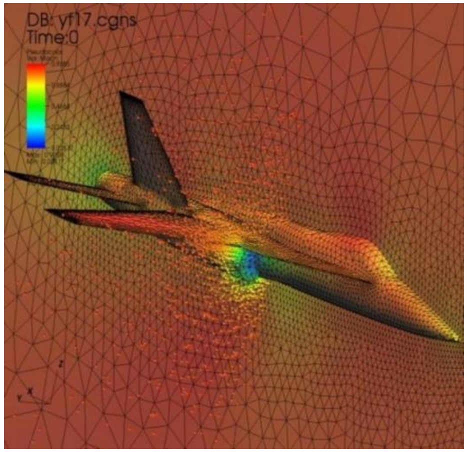


Implementations

- Solving/Approximation of Partial Differential Equations (PDE) or CFD
- Stencil codes
  - Iterative kernels
  - Regular, invariable structure
- Finite Element Methods (FEM)
  - Irregular structures
  - For complex or variant structures
  - Different accuracies
  - Adaptive Mesh Refinement (AMR)
- Technical and scientific computing is mostly modeling
  - Based on time steps
  - Iterative solution until results are stable or simulation period is over



6-point 3D stencil, courtesy: wikipedia.org



RUPRECHT-KARLS-
UNIVERSITÄT
HEIDELBERG

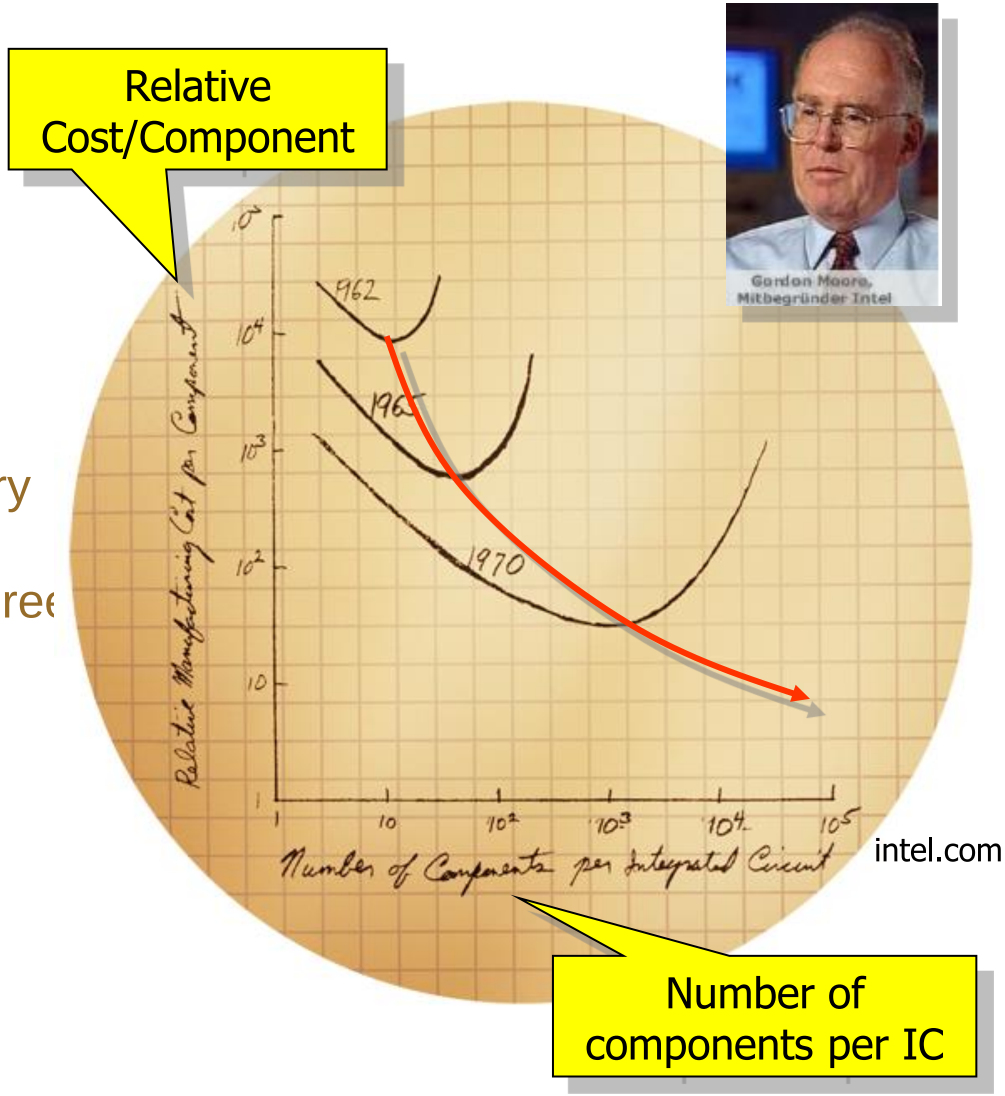
Basics of Performance Increase

Performance Increase of
Technologies and Applications

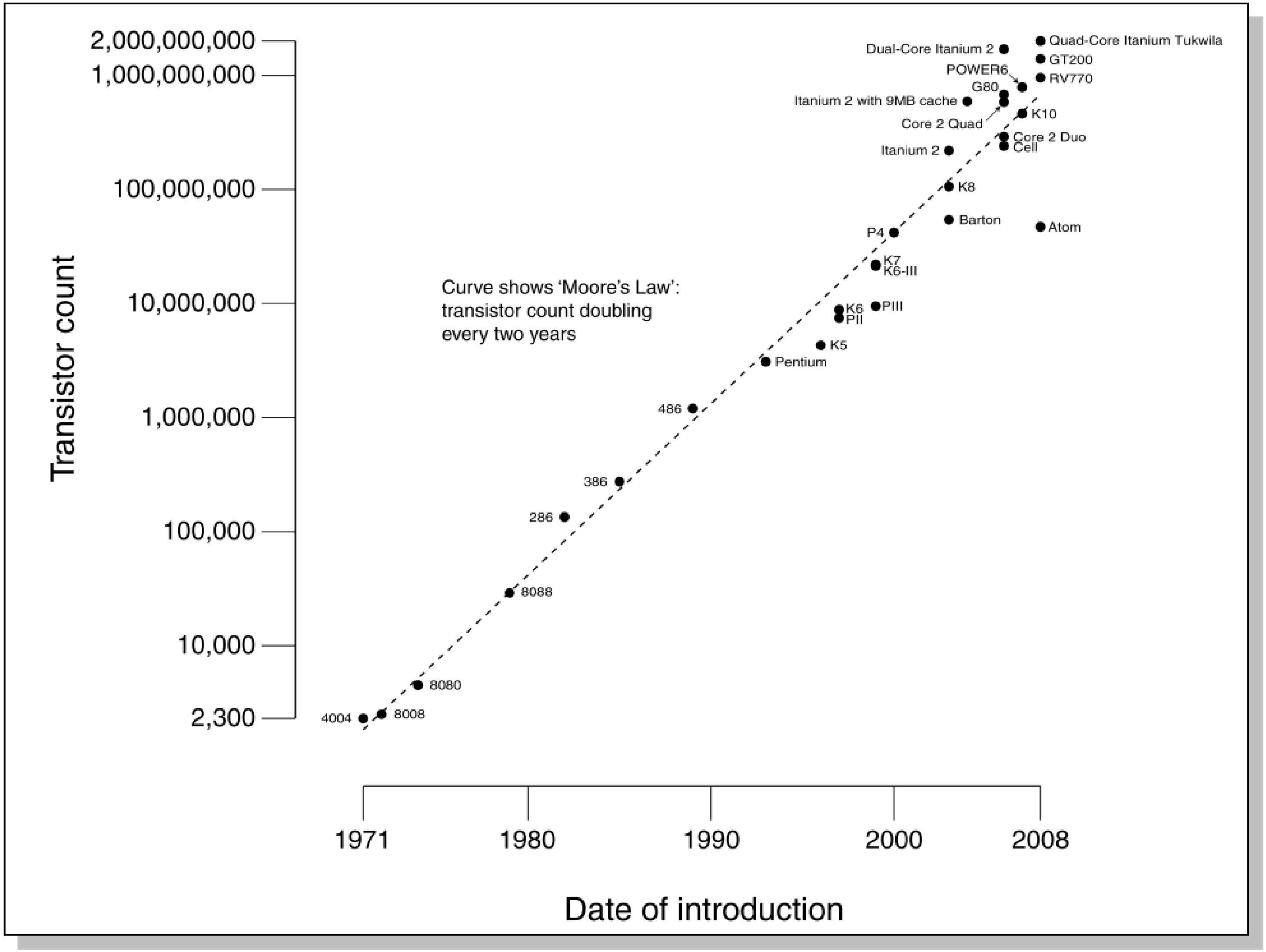
Moore vs. Amdahl

Moore’s Law

- Gordon Moore
  - 1965: Doubling each year
  - 1975: Transistor count of ICs doubling every two years
- Derived “laws”
  - CPU performance doubling every 18 months
  - Memory size four times every three years
  - Memory performance doubling every 10 years
  - At the same costs double performance every two years



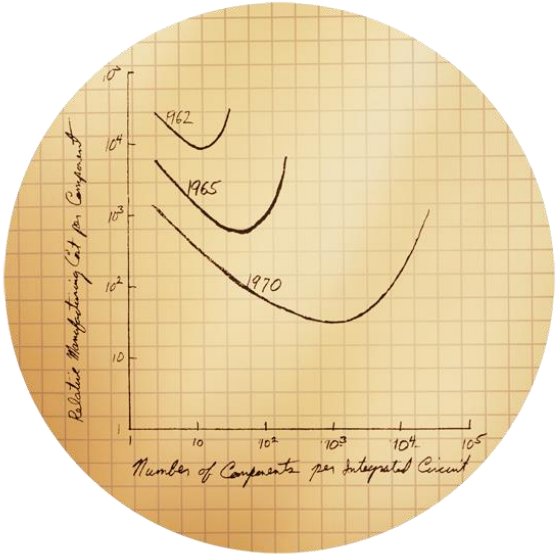
Moore’s Law



wikipedia.org

Moore’s Law

- Industry is trying to keep the pace
  - Self-fulfilling prophecy
  - “positive feedback between belief and behavior”
- Atoms as fundamental lower bound
  - Even then, increase of die size can maintain the law
  - Intel’s statements about end of Moore’s law
    - 2003: 2013-2018
    - 2005: until 2015
    - 2008: until 2029
- Bernie Meyerson (IBM):
  7-9nm is the limit
  - Quantum mechanics effects



Speed-up

- Speed-up: „How much faster can one program be executed“
- Assumption: instead of one resource, N identical resources are available
- Naive: More resources, faster execution
- A bit more realistic: N resources yield an execution time of 1/N
  - No overhead assumed
- Reality: significant loss
  - Break-even point when execution time starts to increase again

Speed-up - Definitions

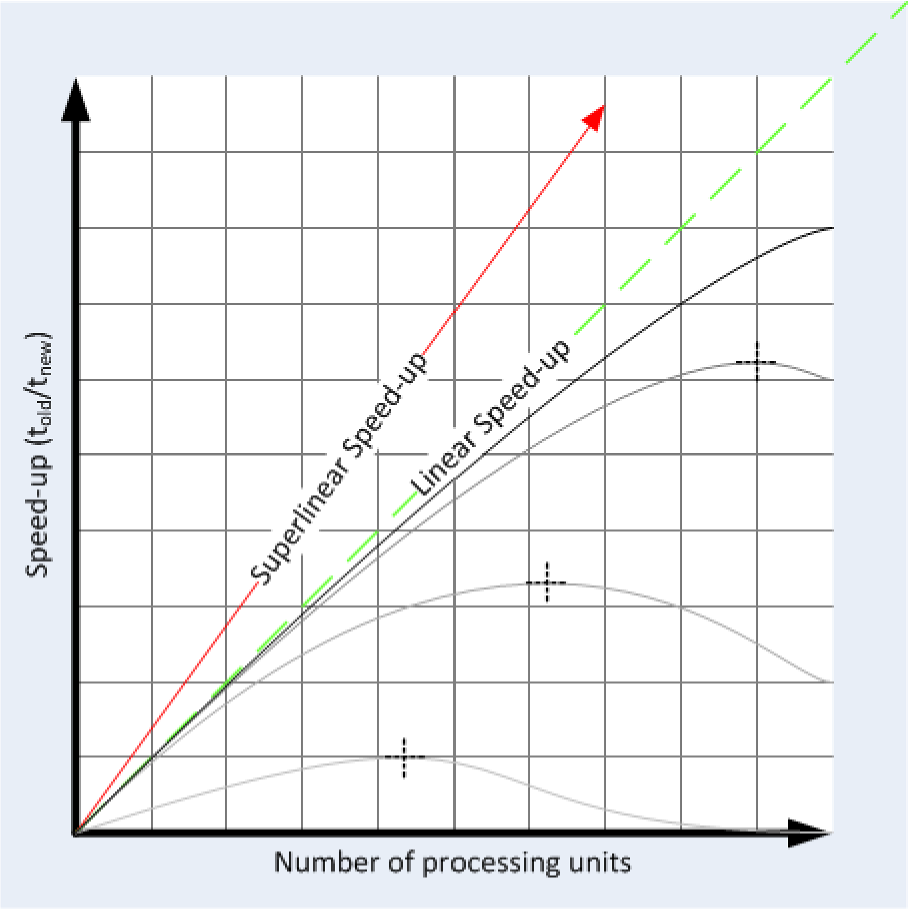
- For a given algorithm:
  - SerTime(n) = time of the best serial implementation for an input of size n
  - ParTime(n,p) = time of the parallel implementation, using p parallel computing units
- Sanity check: SerTime(n) >= ParTime(n,1)
  - The other case is not uncommon
- Speed-up:
  - Speedup(p) = SerTime(n) / ParTime(n,p)
  - Efficiency(p) = SerTime(n) / (p * ParTime(n,p))

Speed-up - Notes

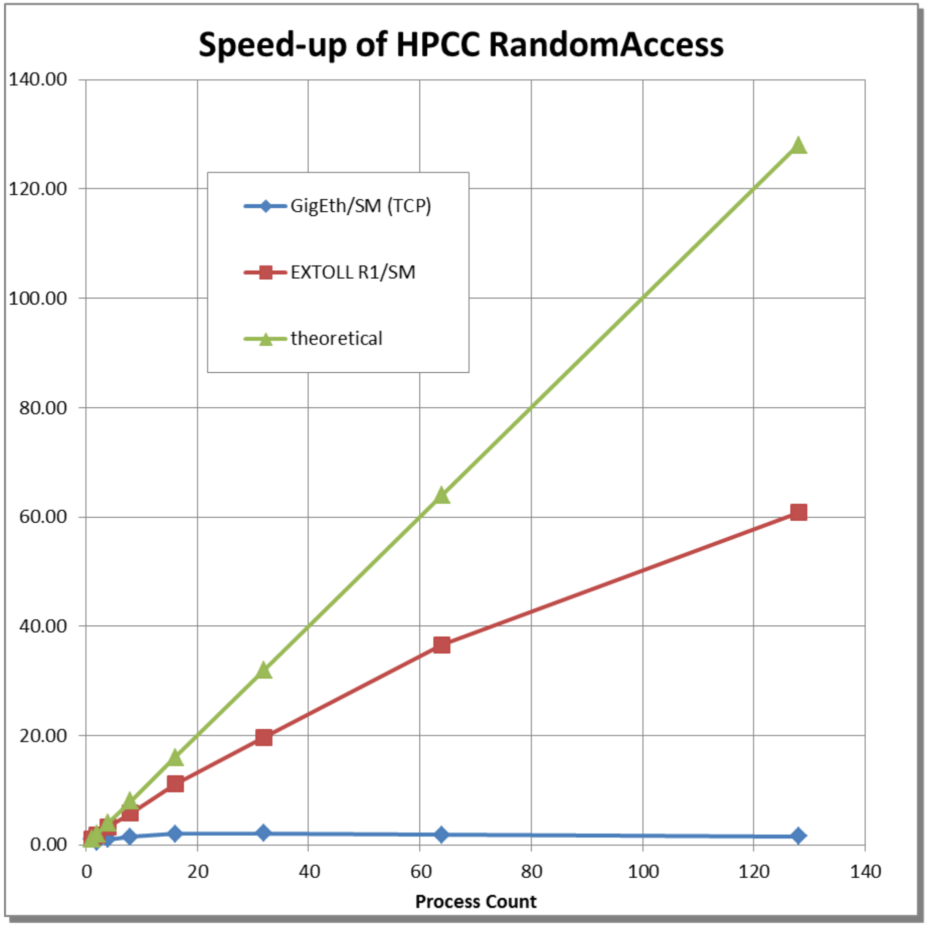
- 1 <= Speedup(p) <= p
- 0 <= Efficiency(p) <= 1

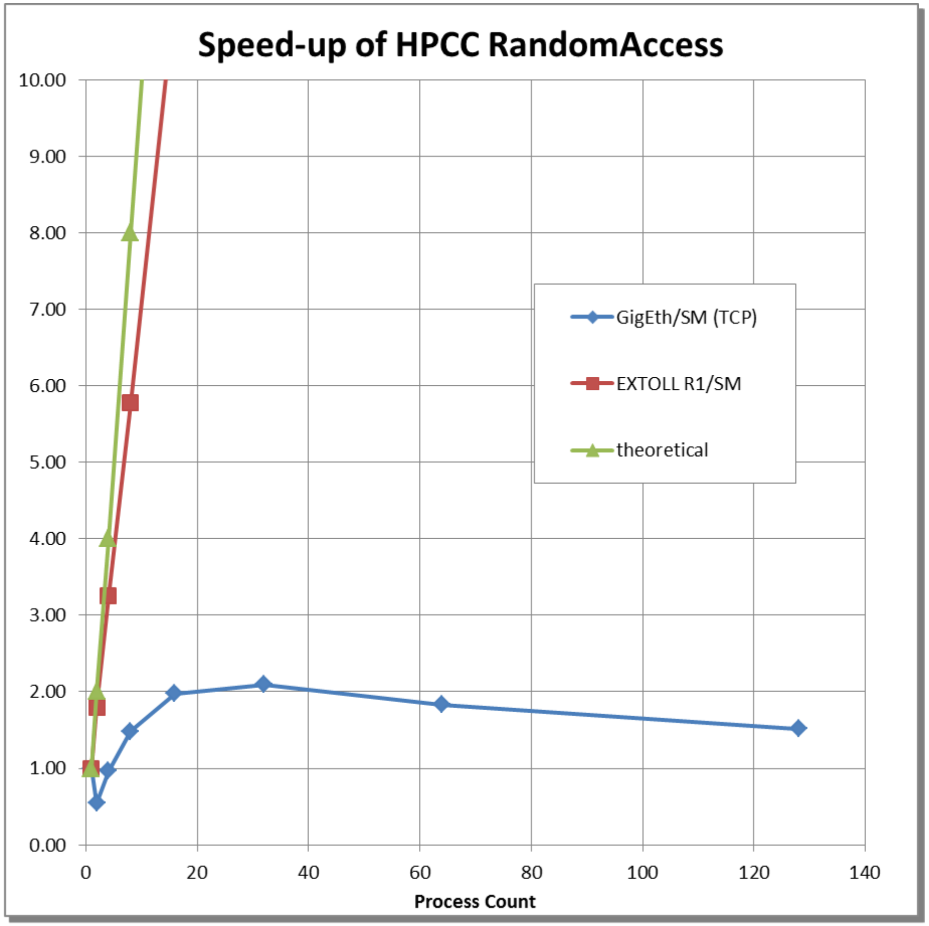
- Linear speed-up:
  Speedup(p) = p

- Superlinear speed-up:
  Speedup(p) > p
  - Usually not possible



Speed-up – Real experiment





Amdahl's Law

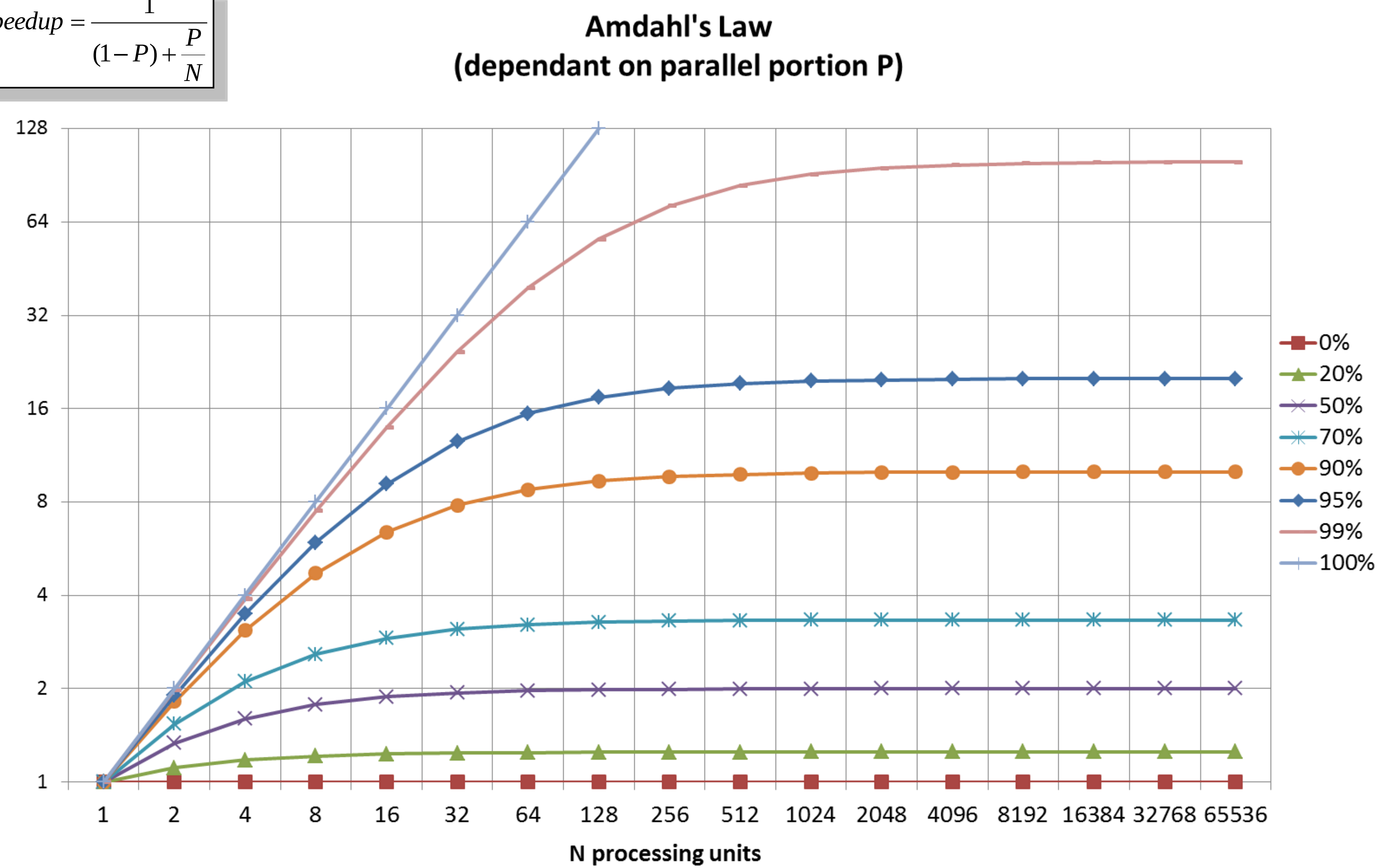
- Model to find the maximum improvement in terms of performance
  - Assumption: only a fraction of the runtime can be parallelized (parallel fraction ***P***).
  - Assumption that the other fraction is the serial one: serial fraction ***S***
  - Then: ***P*** + ***S*** = 1
  - As fraction ***P*** is processed in parallel, this fraction of time is reduced (***N*** parallel execution units)

$$Speedup = \frac{1}{(1-P) + \frac{P}{N}}$$

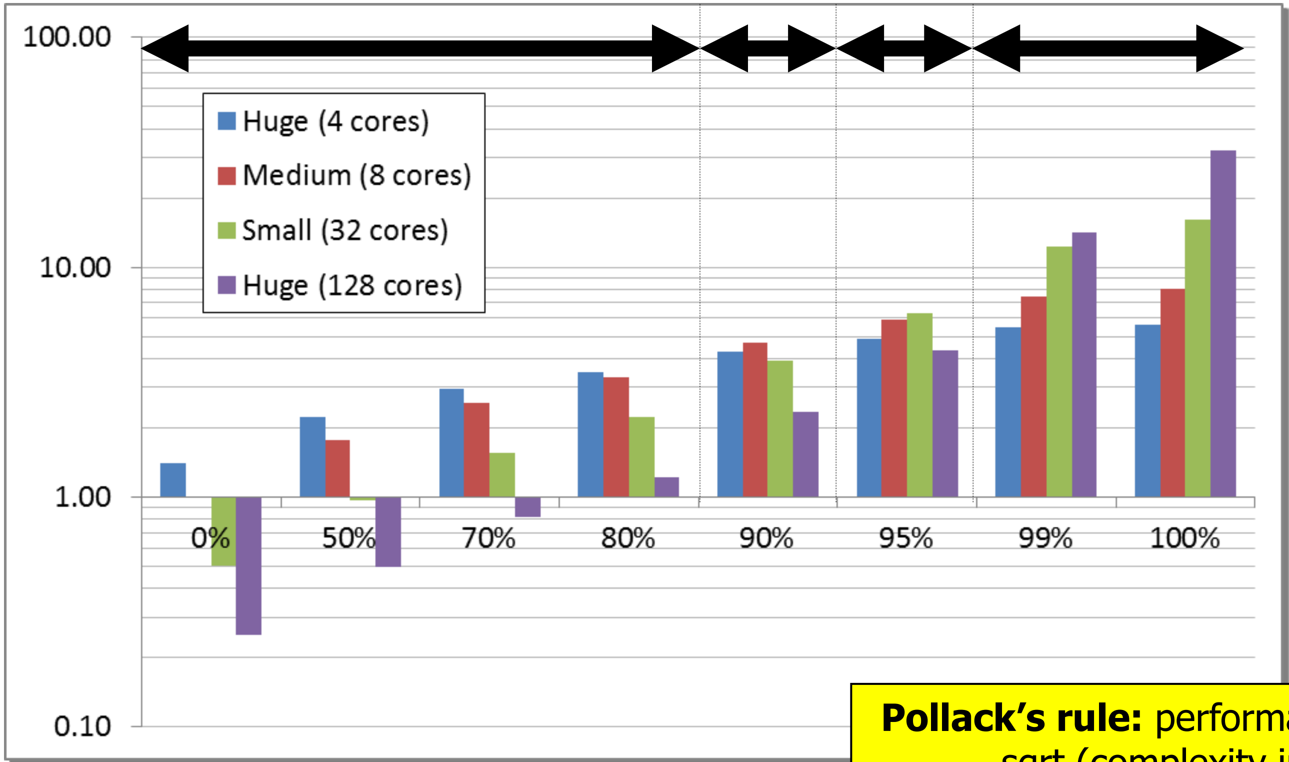
- Notes:
  - Speed-up has an upper limit dependent on ***S***, not on ***N***!

Amdahl's Law

$$Speedup = \frac{1}{(1-P) + \frac{P}{N}}$$



Amdahl's Law - Implications



Assumptions 80W & **200mm²** for cores

Pollack's rule: performance increase ~ sqrt (complexity increase)

Number of Cores	**4**	**8**	**32**	**128**
Power (W) / Core	**20**	**10**	**2,5**	**0,6**
Area (mm²) / Core	**50**	**25**	**6**	**1,5**
Relative Performance R	**140 %**	**100 %**	**50 %**	**25 %**

Notes on Amdahl and his law

- Amdahl himself …
  1. … wanted to claim that **parallel computing is not viable**
     - "Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities“, *AFIPS Conference Proceedings*, 1967.
  2. … was an **optimist**
     - Extra work is required for parallelization
       - Synchronization, communication, management, …
     - In this regard his law is too optimistic
  3. … was a **pessimist**
     - We can (have to?) scale the problem size with ***N***
       - Gustafson‘s law – superlinear speedup (1988)
     - Parallel algorithms exist that reduce fraction ***S***
     - Superlinear speed-up due to caching effects

Performance increase - Summary

- Increase of performance according to Moore's Law
  - Technology ☺
- Increase of performance according to Amdahl's Law
  - Limited by serial fraction ☹

"Everyone knows Amdahl's Law, but quickly forgets"
Dr. Tom Puzak, IBM Research, 2007

- Sources for serial fraction
  - Data dependencies
  - Communication & synchronisation is costly

⇨Optimize these components 😐
⇨Increase problem size ☺
  - Increases percental fraction of ***P***

TOP500 List

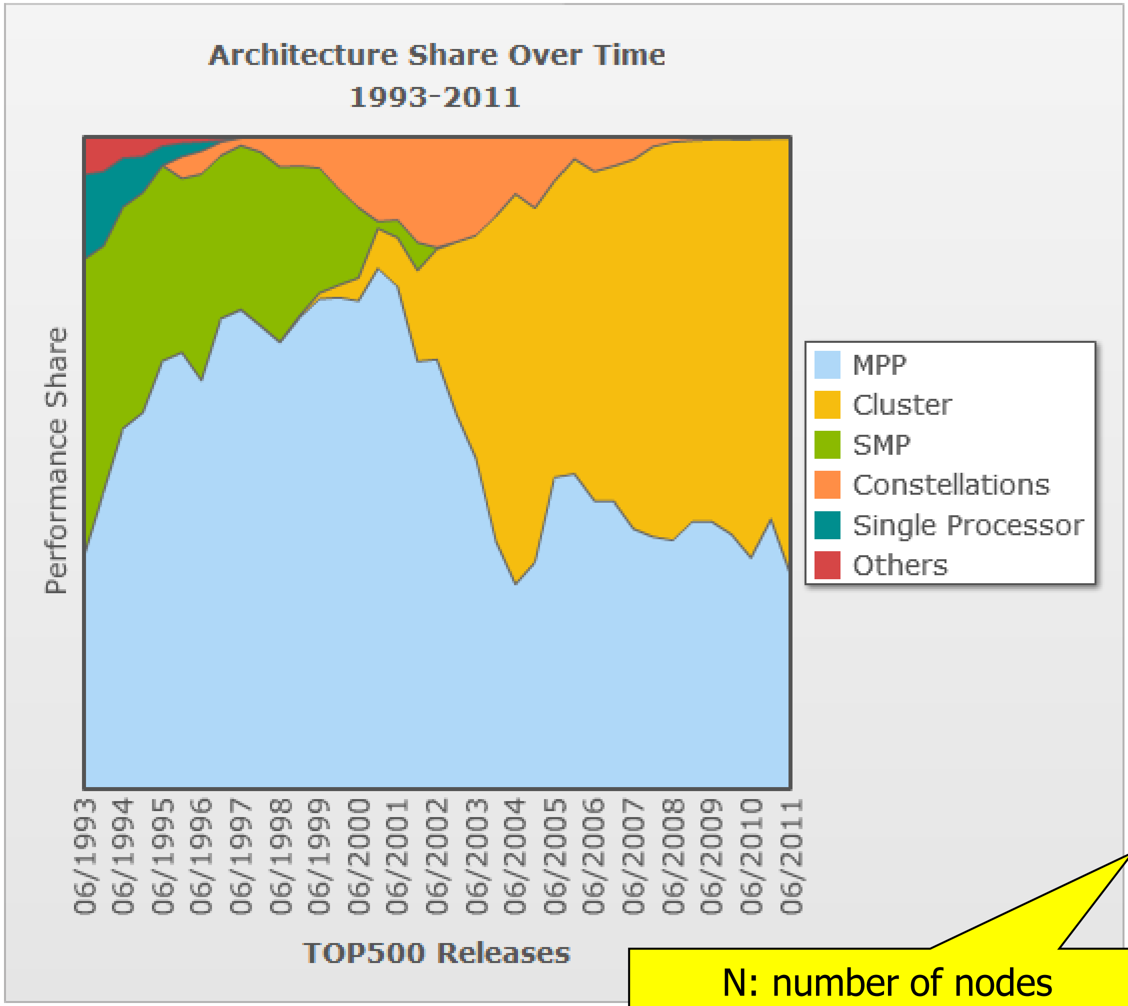
TOP500

- http://www.top500.org
- Biannual list: 500 fastest computer systems world-wide
  - LinPACK benchmark
    - „Dense system of linear equations"
    - $2/3\ n^3 + O(n^2)$ double precision floating point operations
    - Highly scalable, problem size can be chosen arbitrary
- Computationally intensive, not memory-bound
  - Little requirements on memory bandwidth and capacity
- Old lists available on-line
  - History and trends

TOP500 – List of 06/2014

Rank	Site	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	National Super Computer Center in Guangzhou China	**Tianhe-2 (MilkyWay-2)** - TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.200GHz, TH Express-2, Intel Xeon Phi 31S1P NUDT	3,120,000	33,862.7	54,902.4	17,808
2	DOE/SC/Oak Ridge National Laboratory United States	**Titan** - Cray XK7 , Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x Cray Inc.	560,640	17,590.0	27,112.5	8,209
3	DOE/NNSA/LLNL United States	**Sequoia** - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom IBM	1,572,864	17,173.2	20,132.7	7,890
4	RIKEN Advanced Institute for Computational Science (AICS) Japan	K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect Fujitsu	705,024	10,510.0	11,280.4	12,660
5	DOE/SC/Argonne National Laboratory United States	**Mira** - BlueGene/Q, Power BQC 16C 1.60GHz, Custom IBM	786,432	8,586.6	10,066.3	3,945
6	Swiss National Supercomputing Centre (CSCS) Switzerland	**Piz Daint** - Cray XC30, Xeon E5-2670 8C 2.600GHz, Aries interconnect , NVIDIA K20x Cray Inc.	115,984	6,271.0	7,788.9	2,325
7	Texas Advanced Computing Center/Univ. of Texas United States	**Stampede** - PowerEdge C8220, Xeon E5-2680 8C 2.700GHz, Infiniband FDR, Intel Xeon Phi SE10P Dell	462,462	5,168.1	8,520.1	4,510
8	Forschungszentrum Juelich (FZJ) Germany	**JUQUEEN** - BlueGene/Q, Power BQC 16C 1.600GHz, Custom Interconnect IBM	458,752	5,008.9	5,872.0	2,301
9	DOE/NNSA/LLNL United States	**Vulcan** - BlueGene/Q, Power BQC 16C 1.600GHz, Custom Interconnect IBM	393,216	4,293.3	5,033.2	1,972
10	Government United States	Cray XC30, Intel Xeon E5-2697v2 12C 2.7GHz, Aries interconnect Cray Inc.	225,984	3,143.5	4,881.3	

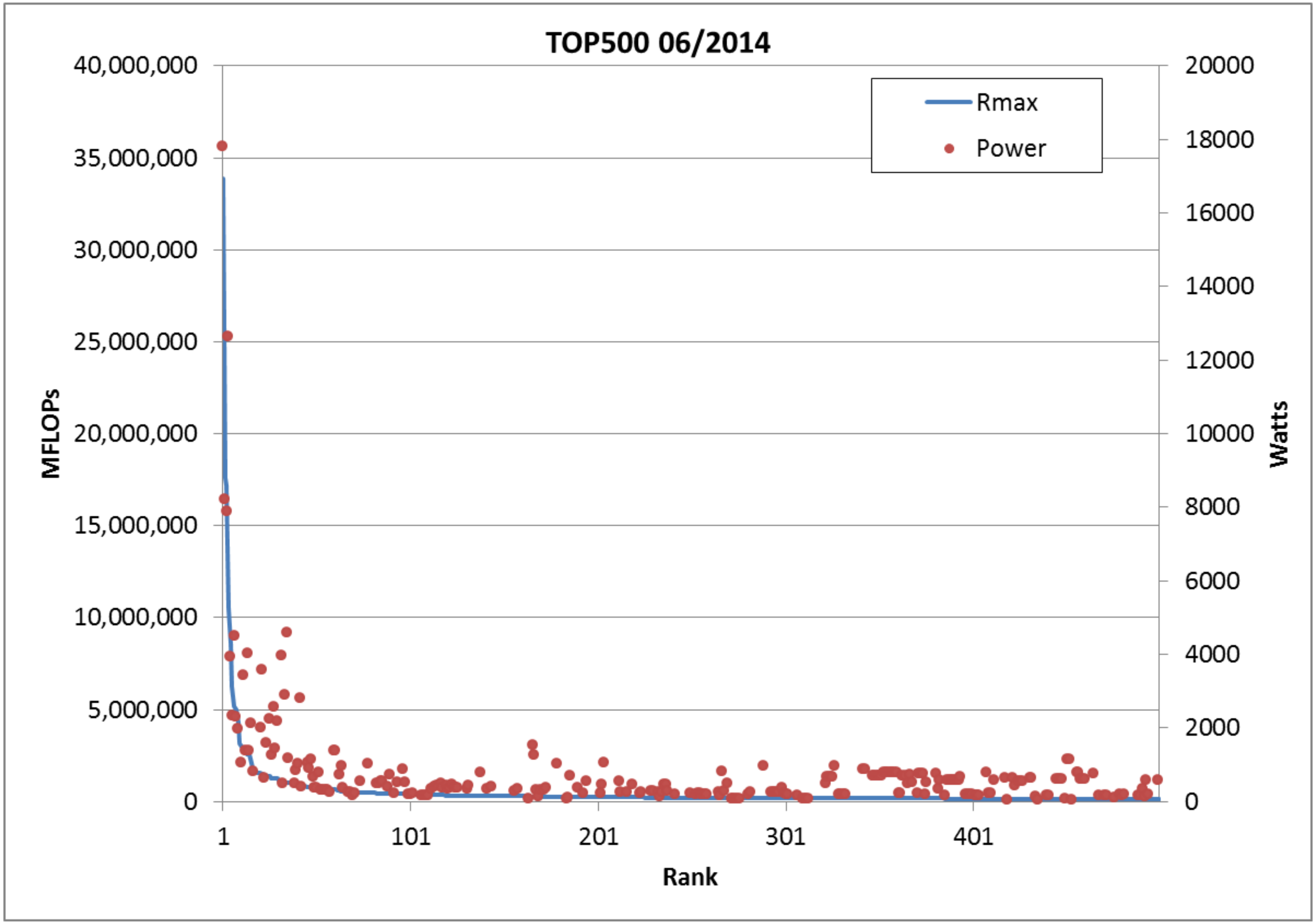
TOP500 – Architecture Share



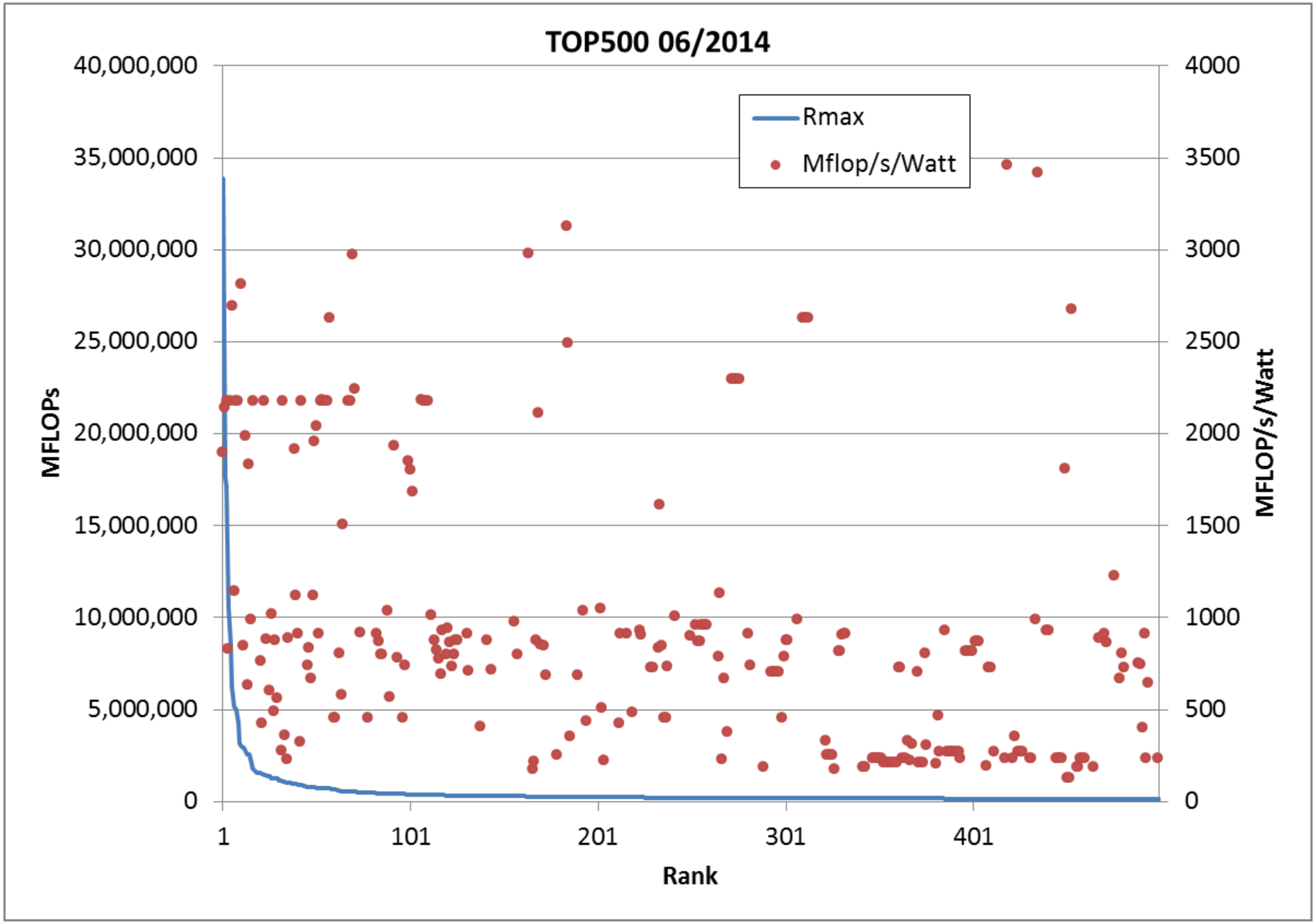
N: number of nodes
C: number of cores per node

- System with N nodes, each C cores
  - 1 node equals 1 address space
- Massively Parallel Processors (MPP)
  - N > C, N >> 1 (whatever that means)
- Cluster
  - N > C, N > 1
- Symmetric Multi-Processors (SMP)
  - N = 1
- Constellations
  - N < C, N > 1
- Single Processor
  - N = C = 1
- Others
  - Never seen

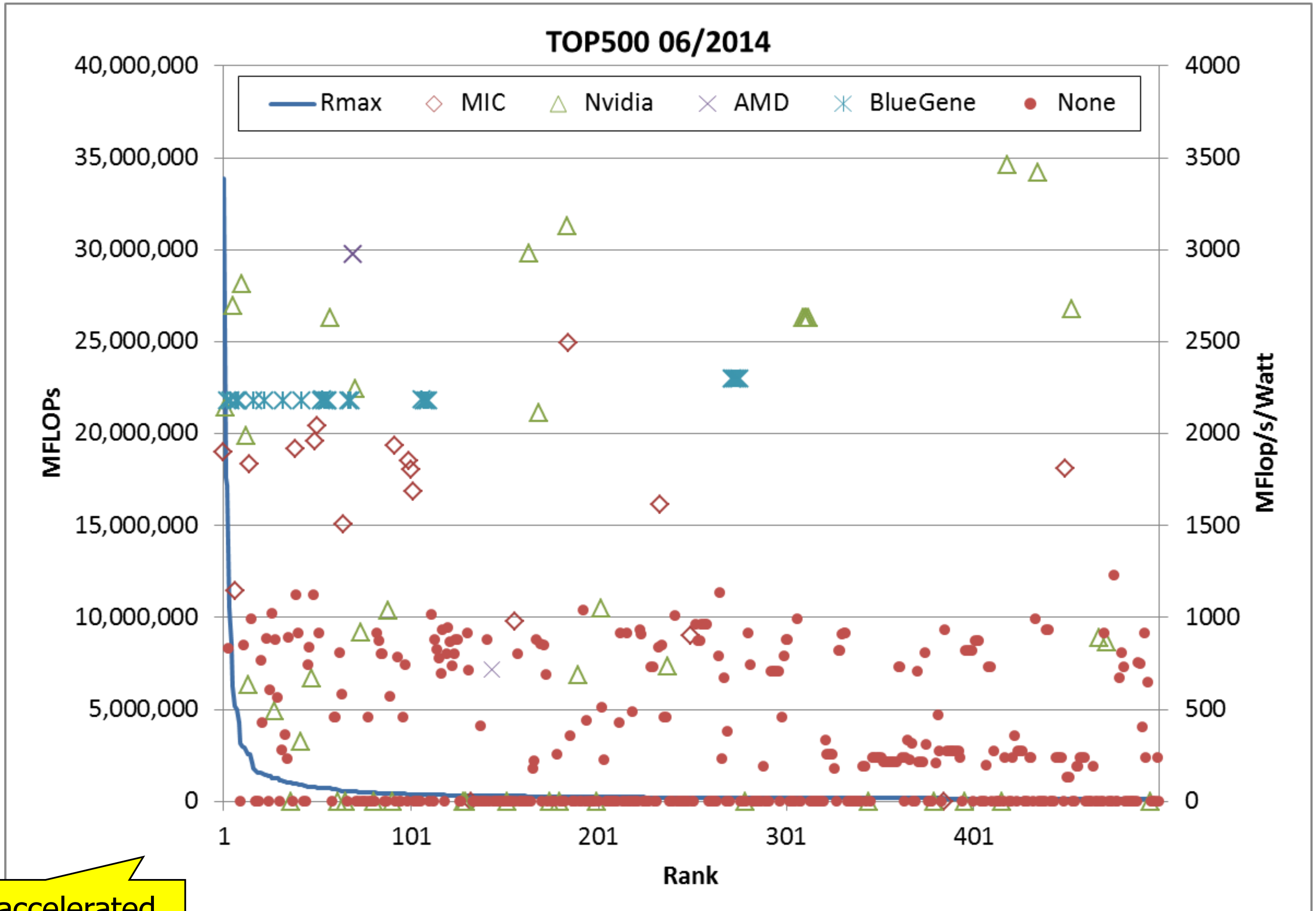
TOP500 – Performance & Power



TOP500 – Power Efficiency



TOP500 – Accelerated Systems



436 not accelerated

TOP500 - Summary

- Excellent tool for trend analysis
  - Introductions, motivating data
- Maybe too limited due to the single workload
  - No scalability worries
- Alternatives
  - Graph500: [http://www.**graph500**.org](http://www.graph500.org)
  - Green500: [http://www.**green500**.org](http://www.green500.org)
  - HPC-Challenge: [http://icl.cs.utk.edu/**hpcc**](http://icl.cs.utk.edu/hpcc)

- Exascale at 2GFLOPs/Watt (BlueGene):
  - 1,000,000,000,000,000,000 Flops (1,000,000,000 GFlops)
  - 500,000,000 Watts (500 MWatt)
  - @50MWatt: 20GFLOPs/Watt required

➔ Extreme specialization
  - What about too specialized?