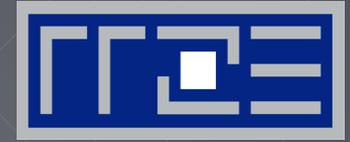


ERLANGEN REGIONAL COMPUTING CENTER



<http://tiny.cc/NLPE-VSC>

Node-Level Performance Engineering

Georg Hager, Gerhard Wellein
Erlangen Regional Computing Center (RRZE)
University of Erlangen-Nuremberg

Quiz

- What does “clock frequency” mean in computers?

The “heartbeat” of the CPU. A clock cycle is the smallest unit of time on a CPU chip. Typically $< 1\text{ns}$ $\rightarrow f \gtrsim 1\text{GHz}$

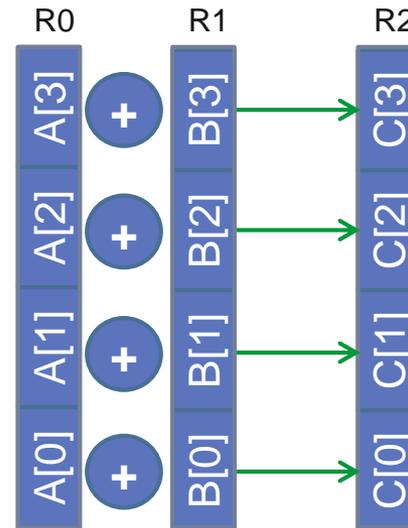
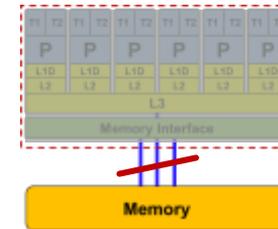
- What is “memory bandwidth”?

Rate of data transfer between main memory (RAM) and CPU chip. Typical $b_S \approx 10 \dots 100\text{GB/s}$

- What is SIMD vectorization?

Single Instruction Multiple Data.
Data-parallel load/store and execution units.

- What is ccNUMA?



Quiz cont.

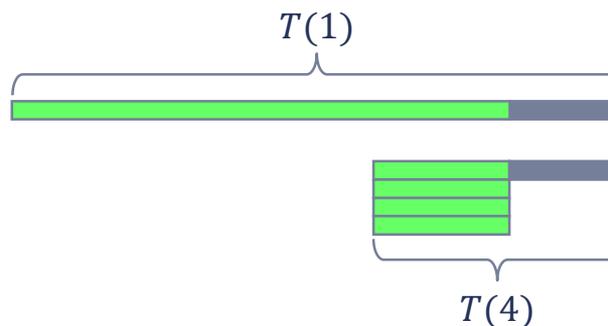
- What is a register?

A storage unit in the CPU core that can take one single value (a few values in case of SIMD). Operands for computations reside in registers.

rax	ymm0
rbx	ymm1
rcx	ymm2
rdx	ymm3
rsi	ymm4

- What is Amdahl's Law?

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



- What is a pipelined functional unit?

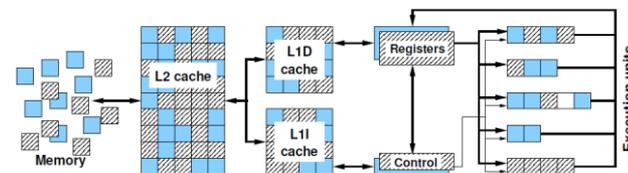
An instruction execution unit on the core that executes a certain task in several simple sub-steps. The stages of the pipeline can act in parallel on several instructions at once.



Quiz cont.

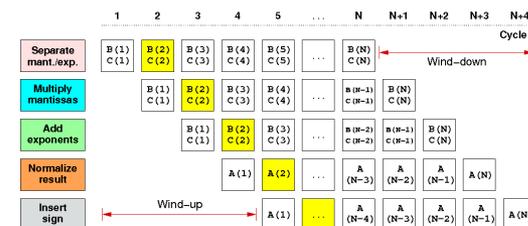
■ What is SMT?

Simultaneous **M**ulti-**T**hreading, a.k.a. hyper-threading. A CPU core can execute multiple threads concurrently. Such threads share all execution resources except the registers.



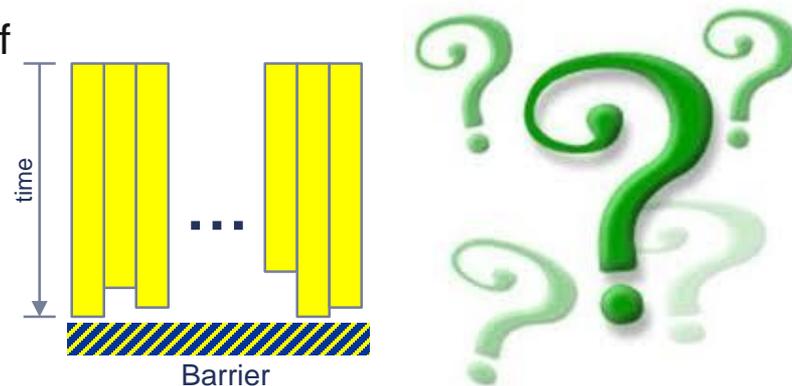
■ How long does a single double-precision floating-point multiplication take?

3-5 clock cycles, depending on the CPU



■ How long does an OpenMP barrier take?

Depending on the OpenMP runtime, the number of cores or threads, and where they are running, a barrier can take tens of thousands of cycles (typically a few 1000s on a full socket).



- 1 cycle = smallest unit of time on a CPU (“heartbeat”)
 - Clock speed of typical CPU: 3.0 Gcy/s (or GHz)
- Basic unit of work: Floating-point operation (Flop)
 - Typical peak performance of 8-core CPU: $P_{\text{peak}} = 192 \text{ Gflop/s}$

- How many Flops per cycle per core is that? $\frac{192 \cdot 10^9 \frac{\text{Flops}}{\text{s}}}{8 \text{ cores} \cdot 3.0 \cdot 10^9 \frac{\text{cy}}{\text{s}}} = 8 \frac{\text{Flops}}{\text{cy} \cdot \text{core}}$

- Typical duration of a double precision multiply: 5 cycles

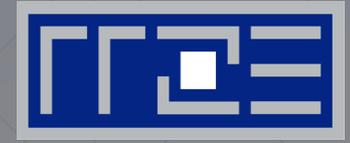
- How much time is that? $\frac{5 \text{ cy}}{3.0 \cdot 10^9 \frac{\text{cy}}{\text{s}}} = 1.67 \cdot 10^{-9} \text{ s} = 1.67 \text{ ns}$

- Basic unit of traffic: Byte
- Unit of bandwidth: Bytes/s
 - Typical memory bandwidth: 48 Gbytes/s = $4.8 \cdot 10^{10} \text{ Bytes/s}$

- How many bytes per cycle is that? $\frac{48 \cdot 10^9 \frac{\text{Bytes}}{\text{s}}}{3.0 \cdot 10^9 \frac{\text{cy}}{\text{s}}} = 16 \frac{\text{Bytes}}{\text{cy}}$



PRELUDE: SCALABILITY 4 THE WIN!



How to ask the right questions



From a student seminar on “Efficient programming of modern multi- and manycore processors”

Student: I have implemented this algorithm on the GPGPU, and it solves a system with 26546 unknowns in 0.12 seconds, so it is really fast.

Me: What makes you think that 0.12 seconds is fast?

Student: It is fast because my baseline C++ code on the CPU is about 20 times slower.

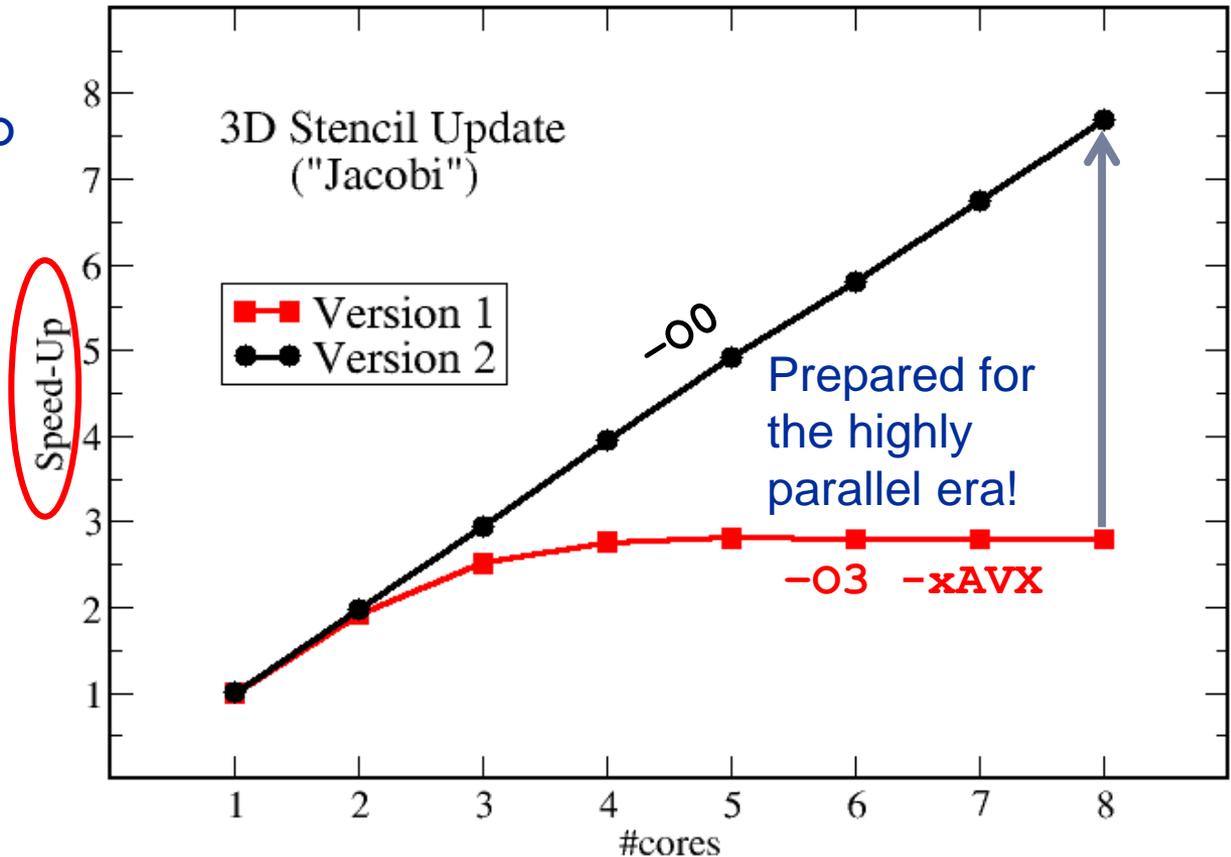
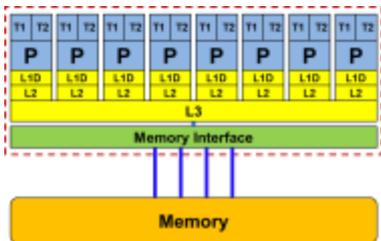
Scalability Myth: Code scalability is the key issue



```

!$OMP PARALLEL DO
do k = 1 , Nk
  do j = 1 , Nj; do i = 1 , Ni
    y(i,j,k) = b*( x(i-1,j,k)+ x(i+1,j,k)+ x(i,j-1,k)+
                  x(i,j+1,k)+ x(i,j,k-1)+ x(i,j,k+1))
  enddo; enddo
enddo
!$OMP END PARALLEL DO
    
```

Changing only a the compile options makes this code scalable on an 8-core chip



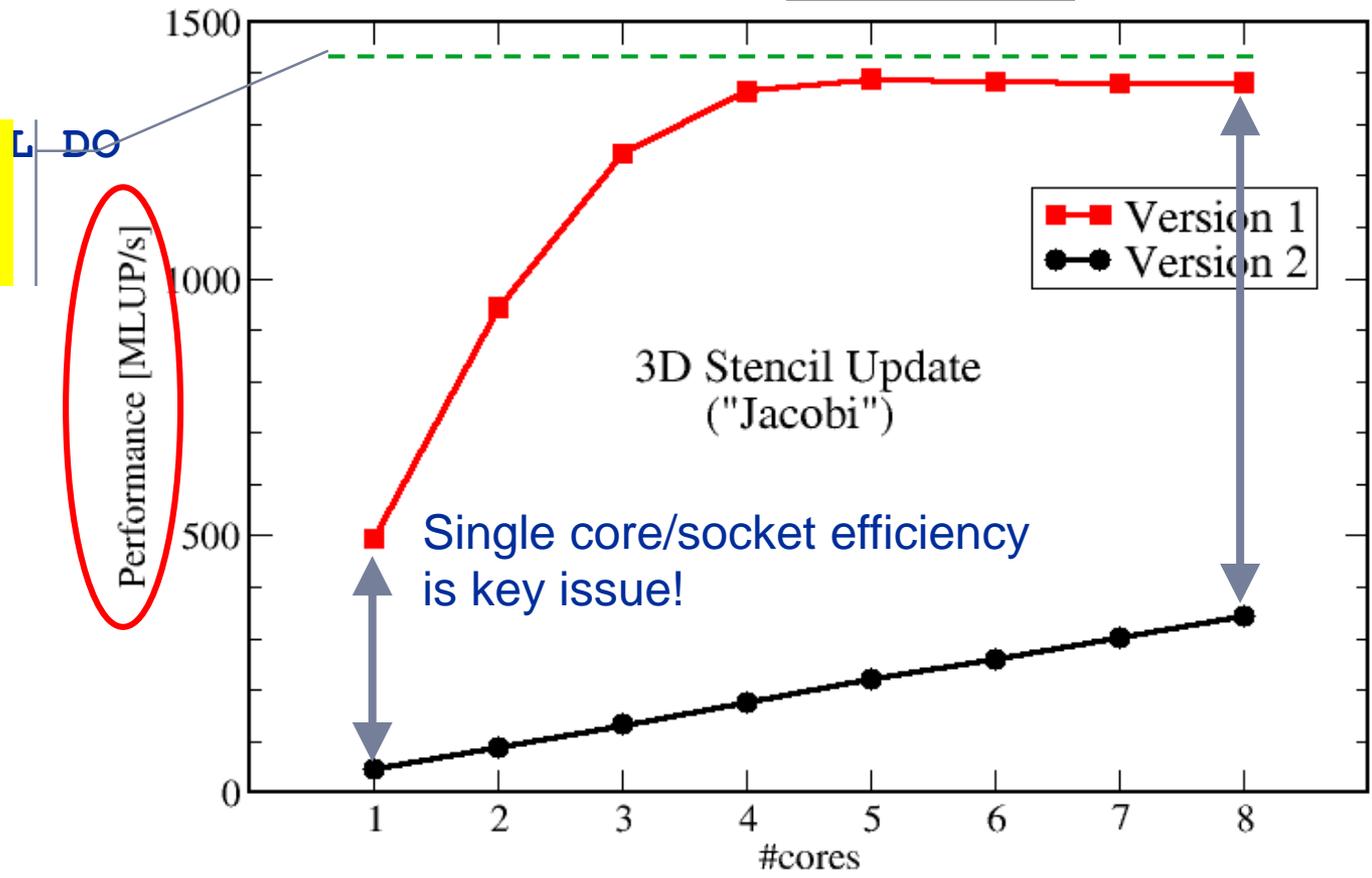
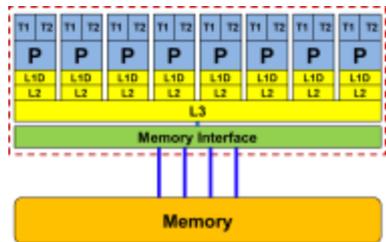
Scalability Myth: Code scalability is the key issue



```

!$OMP PARALLEL DO
do k = 1 , Nk
  do j = 1 , Nj; do i = 1 , Ni
    y(i,j,k) = b*( x(i-1,j,k) + x(i+1,j,k) + x(i,j-1,k) +
      x(i,j+1,k) + x(i,j,k-1) + x(i,j,k+1) )
  enddo; enddo
enddo
    
```

Upper limit from simple performance model:
35 GB/s & 24 Byte/update



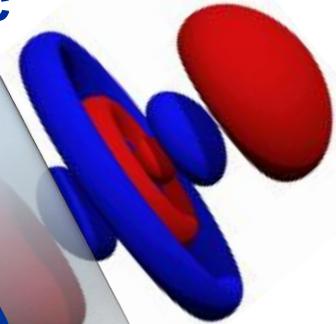
Newtonian mechanics



$$\vec{F} = m\vec{a}$$

Fails @ small scales!

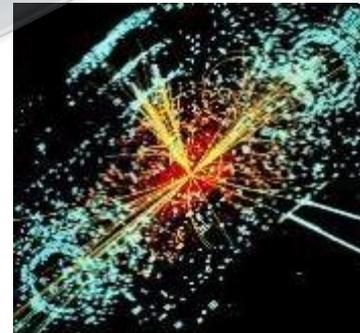
Nonrelativistic quantum mechanics



$$i\hbar \frac{\partial}{\partial t} \psi(\vec{r}, t) = H\psi(\vec{r}, t)$$

Fails @ even smaller scales!

**If a model fails,
we learn something!**



Relativistic quantum field theory

$$U(1)_Y \otimes SU(2)_L \otimes SU(3)_c$$



- **Do I understand the performance behavior of my code?**
 - Does the performance **match a model** I have made?
- **What is the optimal performance for my code on a given machine?**
 - **High Performance Computing == Computing at the bottleneck**
- **Can I change my code so that the “optimal performance” gets higher?**
 - Circumventing/ameliorating the impact of the bottleneck
- **My model does not work – what’s wrong?**
 - This is the good case, because **you learn something**
 - Performance monitoring / microbenchmarking may help clear up the situation
- **Use your brain!** Tools may help, but you do the thinking.