ERLANGEN REGIONAL COMPUTING CENTER

http://tiny.cc/EOCOE-PE

Performance Evaluation Workshop

Georg Hager[1], Jan Eitzinger[1], Judit Gimenez[2]

[1] Erlangen Regional Computing Center (RRZE), FAU Erlangen-Nuremberg
[2] Barcelona Supercomputing Center (BSC)

FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

ERLANGEN REGIONAL COMPUTING CENTER

http://tiny.cc/EOCOE-PE

Part 1: Node-Level Performance Engineering

Georg Hager[1], Jan Eitzinger[1]

[1] Erlangen Regional Computing Center (RRZE)
FAU Erlangen-Nuremberg

FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

Quiz

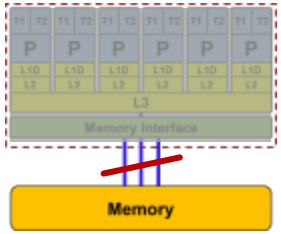
http://tiny.cc/EOCOE-PE

- 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 < 1ns → $f \gtrsim 1\ \mathrm{GHz}$

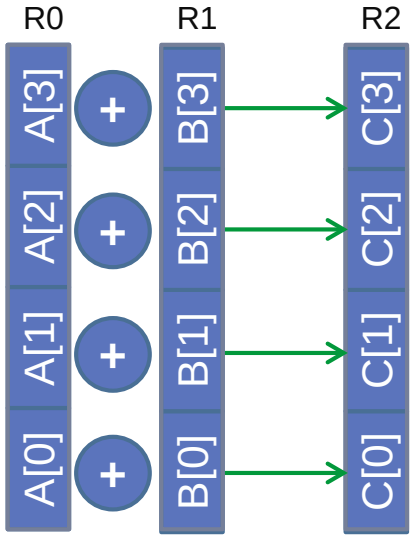
- What is “memory bandwidth”?

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

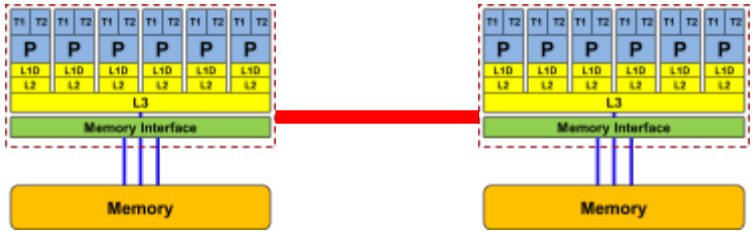


- What is SIMD vectorization?

  Single **I**nstruction **M**ultiple **D**ata.
  Data-parallel load/store and execution units.



- What is ccNUMA?

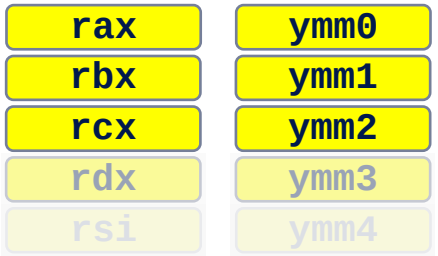


Quiz cont.

http://tiny.cc/EOCOE-PE

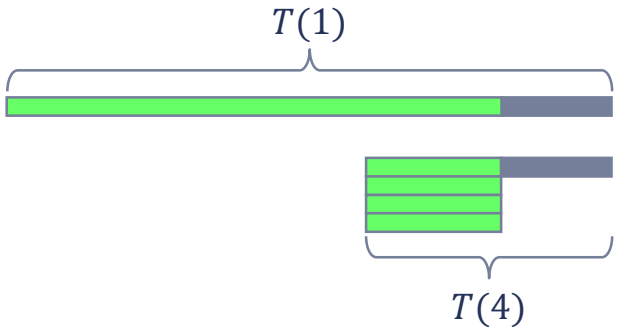
- What is a register?

  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.



- 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?

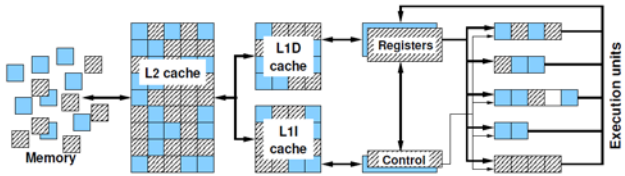
  An instruction execution unit on the core that executes
  a certain task in several simple sub-steps.
  The stages of the pipeline can act in parallel
  on several instructions at once.

Quiz cont.

http://tiny.cc/EOCOE-PE

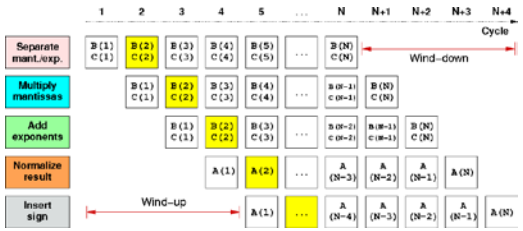
- What is SMT?

  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.



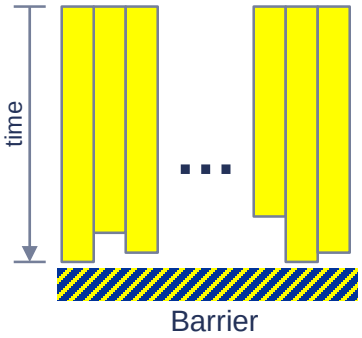
- How long does a single double-precision floating-point multiplication take?

  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).

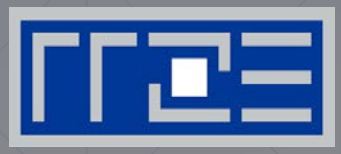


Cycle gymnastics

http://tiny.cc/EOCOE-PE

- 1 cycle = smallest unit of time on a CPU (“heartbeat”)
  - Clock speed of typical CPU: 2.0 Gcy/s (or GHz)
- Basic unit of work: Floating-point operation (Flop)
  - Typical peak performance of 20-core CPU: P_{peak} = 1280 Gflop/s
  - How many Flops per cycle per core is that? $\frac{1280 \cdot 10^9 \frac{\text{Flops}}{\text{s}}}{20\ cores \cdot 2.0 \cdot 10^9 \frac{\text{cy}}{\text{s}}} = 32\ \frac{\text{Flops}}{\text{cy} \cdot \text{core}}$
  - Typical duration of a double precision multiply: 4 cycles
    - How much time is that? $\frac{4\ \text{cy}}{2.0 \cdot 10^9 \frac{\text{cy}}{\text{s}}} = 2.0 \cdot 10^{-9}\text{s} = 2.0\ \text{ns}$
- Basic unit of traffic: Byte
- Unit of bandwidth: Bytes/s
  - Typical memory bandwidth: 105 Gbytes/s = $1.05 \cdot 10^{11}$ Bytes/s
  - How many bytes per cycle is that? $\frac{1.05 \cdot 10^{11} \frac{\text{Bytes}}{\text{s}}}{2.0 \cdot 10^9 \frac{\text{cy}}{\text{s}}} = 52.5\ \frac{\text{Bytes}}{\text{cy}}$

PRELUDE: SCALABILITY 4 THE WIN!



How to ask the right questions

A conversation

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.

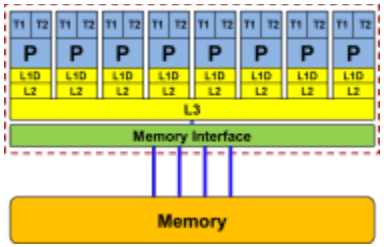
Me: What makes you think that 0.12 seconds is fast?

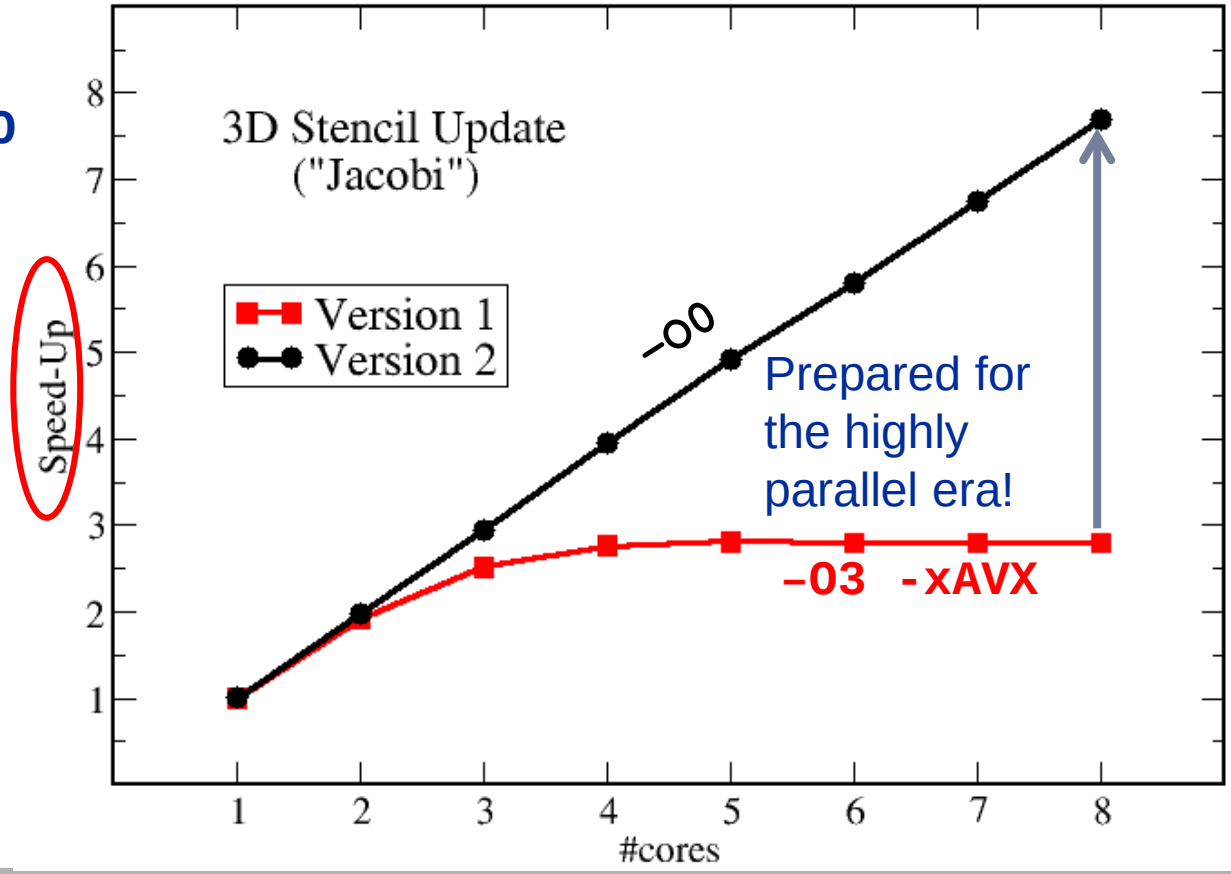
Student: It is fast because my baseline C++ code on the CPU is about 20 times slower.

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

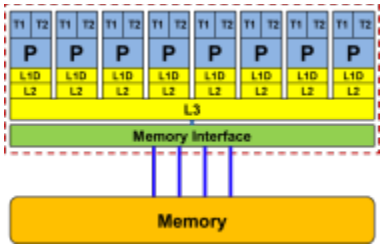


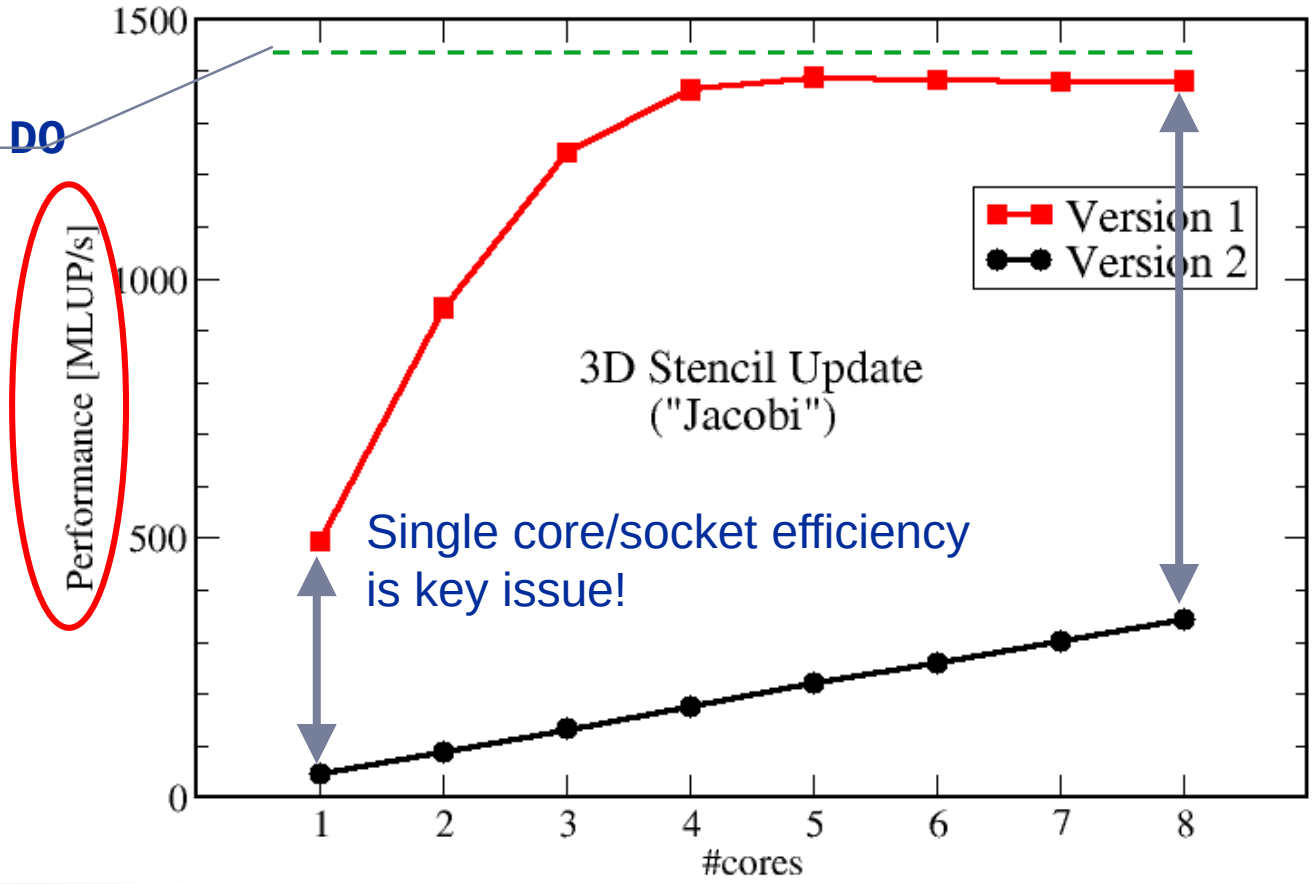


Scalability Myth: Code scalability is the key issue

```
!$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





Questions to ask in high performance computing

- **Do I understand the performance behavior of my code?**
  - What is (or should be) the principal hardware bottleneck?
  - 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.**