ERLANGEN REGIONAL COMPUTING CENTER

http://goo.gl/41Qacc

Node-Level Performance Engineering

Georg Hager, Jan Treibig, Gerhard Wellein
Erlangen Regional Computing Center
University of Erlangen-Nuremberg

Specialist Workshop on Scientific Computing
KU Leuven, Belgium
2015-04-9/10

FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

Agenda

http://goo.gl/41Qacc

Intro & Quiz

Multicore & computer architecture

Topology & affinity

Microbenchmarking

The Roofline Model (with examples)

Case study: Jacobi smoother

Optimal use of parallel resources: ccNUMA, SIMD, SMT

Optional: The Execution-Cache-Memory (ECM) model

Pattern-guided performance engineering

Quiz

http://goo.gl/41Qacc

- What is a “write-allocate” (a.k.a. read for ownership)?

  A: Many cache architectures allocate a cache line on a store miss.

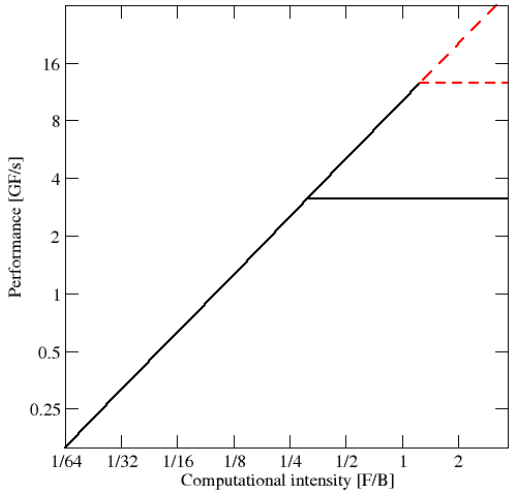
- What is Amdahl’s Law?

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

- What is the Roofline Model?

[1] W. Schönauer: Scientific Supercomputing: Architecture and Use of Shared and Distributed Memory Parallel Computers. (2000)

[2] S. Williams: Auto-tuning Performance on Multicore Computers. UCB Technical Report No. UCB/EECS-2008-164. PhD thesis (2008)

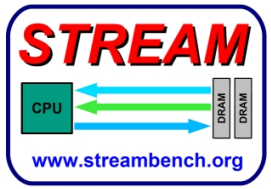


Quiz cont.

http://goo.gl/41Qacc

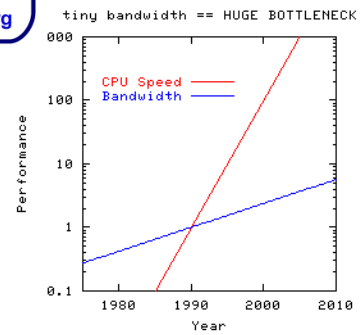
- How many cycles does a double-precision ADD/MULT/DIV take?

  A: on Intel Sandy Bridge:
  ADD 3 cycles, MULT 5 cycles , DIV 22 cycles



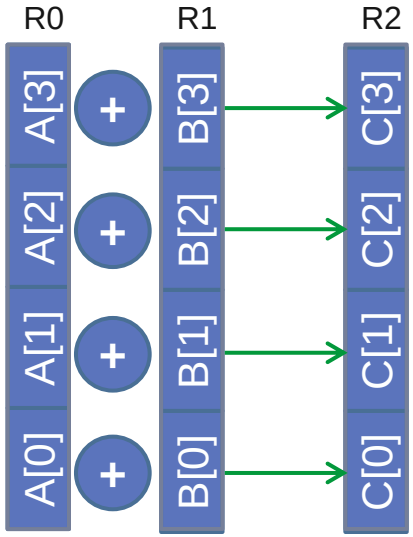
- Do you know the STREAM benchmarks?

  A: De facto standard HPC benchmark
  for (memory) bandwidth.

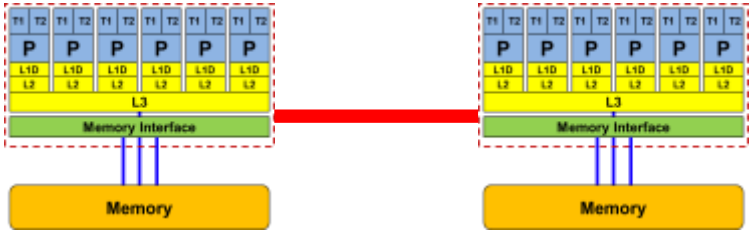


- What is SIMD vectorization?

  A: Single Instruction Multiple Data.
  Data-parallel execution units.



- What is ccNUMA?

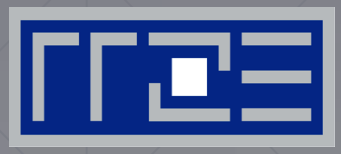


Cycle gymnastics

http://goo.gl/41Qacc

- 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_{peak} = 192 Gflop/s
  - How many Flops per cycle per core is that? $\frac{192 \cdot 10^9 \frac{Flops}{s}}{8\ cores \cdot 3.0 \cdot 10^9 \frac{cy}{s}} = 8 \frac{Flops}{cy \cdot core}$
  - Typical duration of a double precision multiply: 5 cycles
    - How much time is that? $\frac{5\ cy}{3.0 \cdot 10^9 \frac{cy}{s}} = 1.67 \cdot 10^{-9} s = 1.67\ ns$
- Basic unit of traffic: Byte
- Unit of bandwidth: Bytes/s
  - Typical memory bandwidth: 48 Gbytes/s = $4.8 \cdot 10^{10}$ Bytes/s
  - How many bytes per cycle is that? $\frac{48 \cdot 10^9 \frac{Bytes}{s}}{3.0 \cdot 10^9 \frac{cy}{s}} = 16 \frac{Bytes}{cy}$

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.

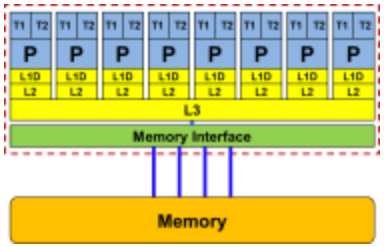
Me: What makes you think that 0.12 seconds is fast?

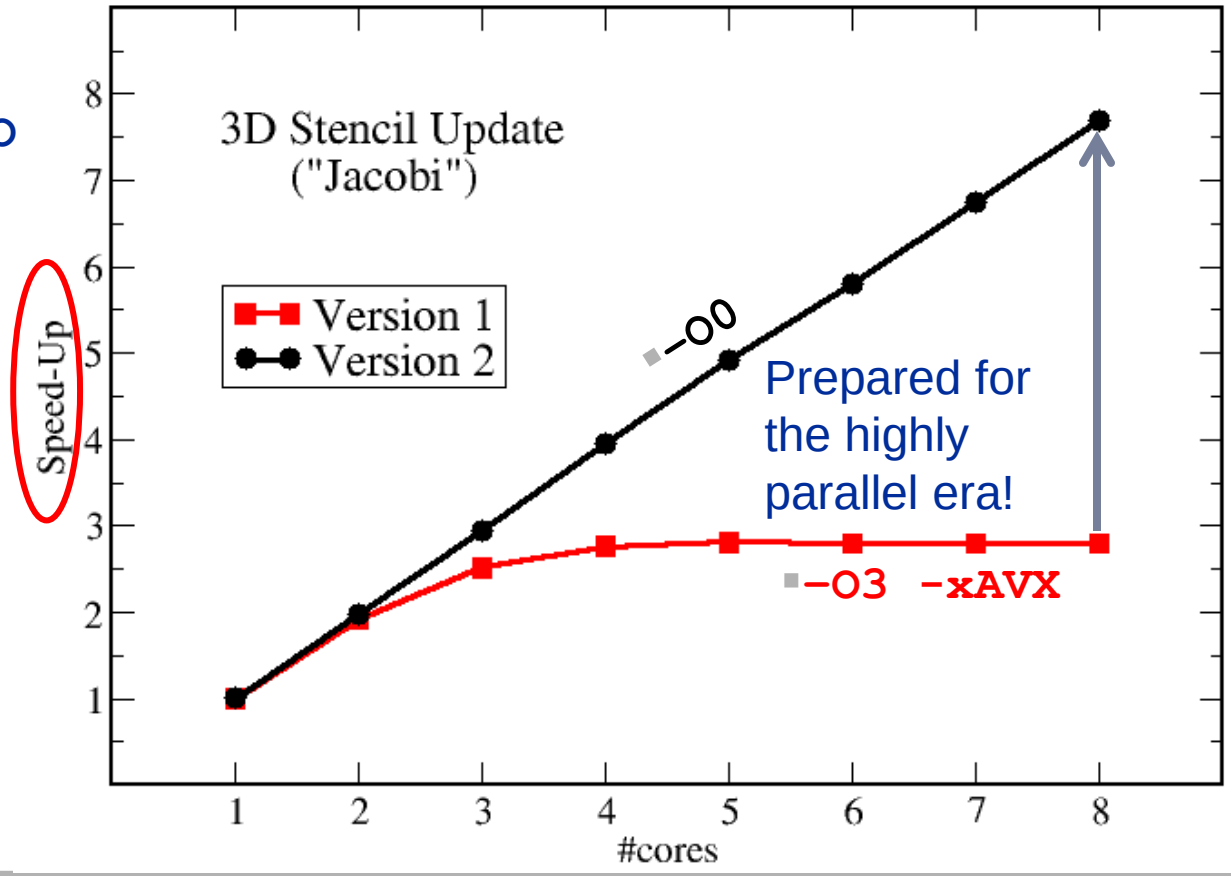
Student: It is fast because my baseline C++ code on the CPU is about 20 times slower.

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

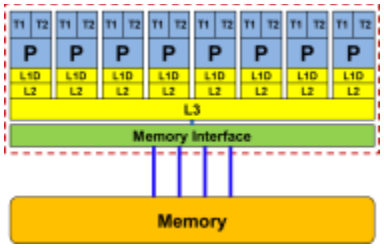


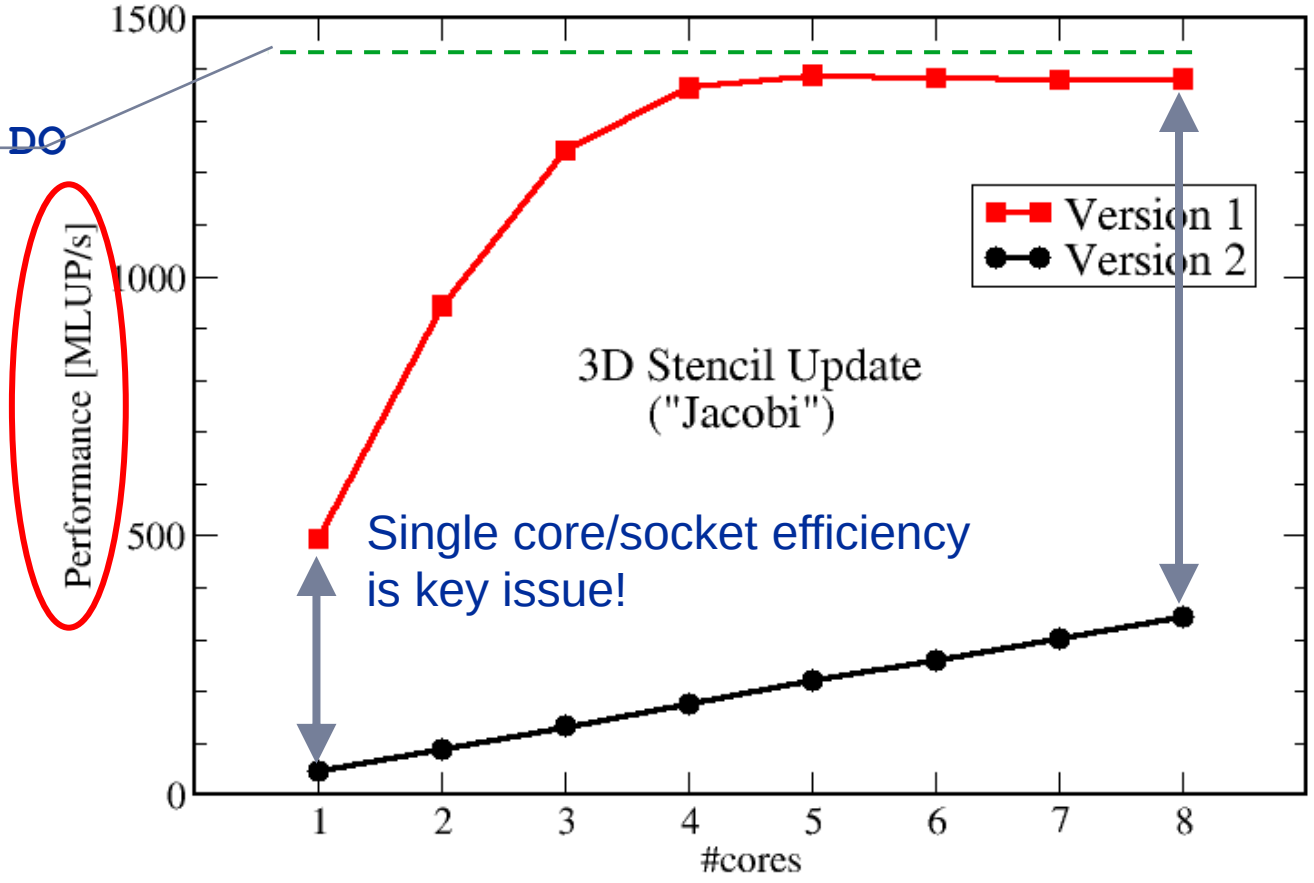


Scalability Myth: Code scalability is the key issue

```
!$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?**
  - 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.**