The HP2C COSMO-CLM Project

Task 3 …and why you might be interested

Oliver Fuhrer, MeteoSwiss

…with results from Tobias Gysi, SCS

…initiated by Thomas Schulthess, CSCS

Questions

- Whatever I do, **computers will be twice as fast in two years** time and my performance problems will have vanished, right?
- **Why should I worry** about memory, I/O, bandwidth, caches et al. (when generations of PhD students before me didn't have to)?
- **Have CSCS and MeteoSwiss gone nuts?**

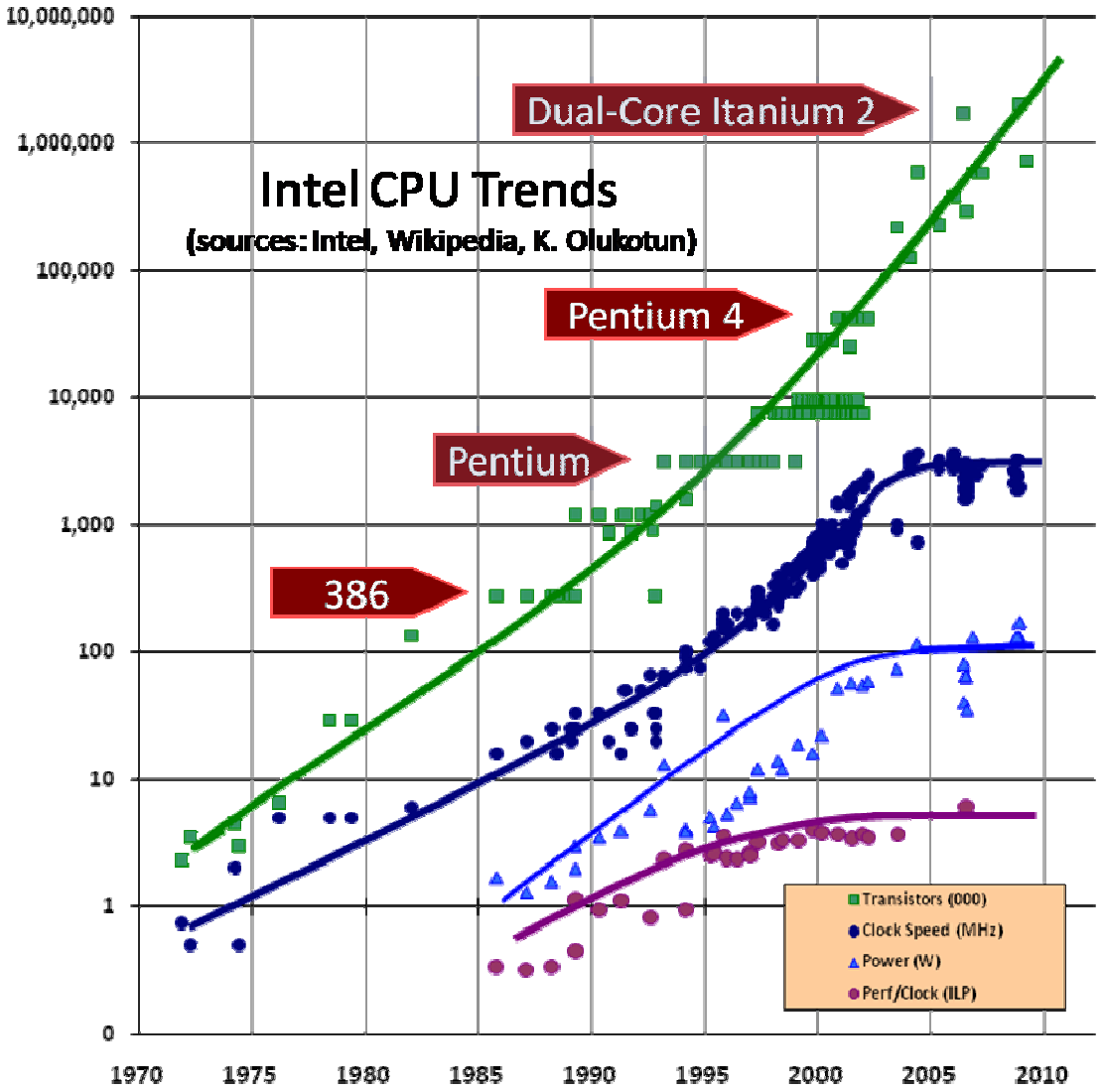
Example Code (Dynamics)

- 1D diffusion equation (niter = 48, nwork = 4096000)

```
do j = 1, niter
  do i = 1, nwork
    c(i) = a(i) + b(i) * ( a(i+1) - 2.0d0*a(i) + a(i-1) )
  end do
end do
```

Machine	buin (XT4)	rosa (XT5)	palu (XE6)
Mountain			
CPU	AMD Opteron Barcelona	AMD Opteron Istanbul	AMD Opteron Magny-Cours
Introduction	April 2008	June 2009	April 2010
Runtime	0.802 s	0.836 s	0.631 s

Moore's Law



Number of transistors

Clock speed (MHz)

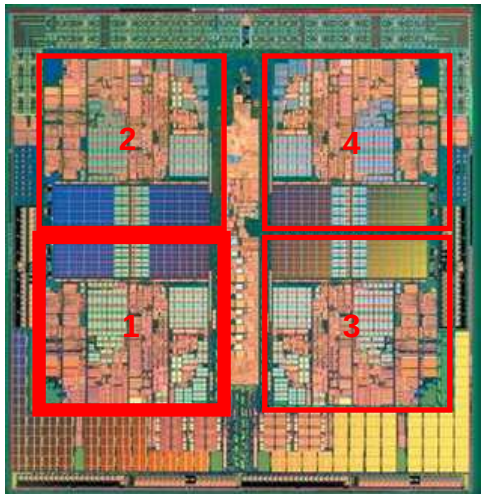
Power (W)

Instructions per Clock (ILP)

Multicore Chips

- Chip makers discover Copy & Paste…

buin (Barcelona)

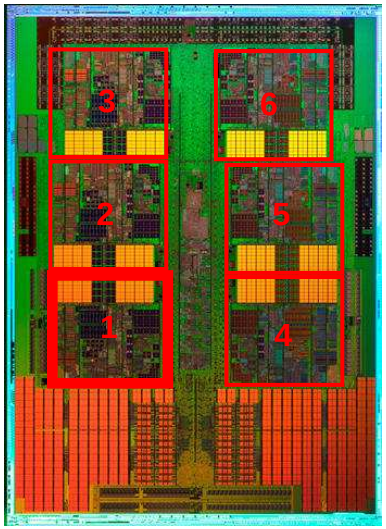


4 cores

0.558 s

Reference

rosa (Istanbul)

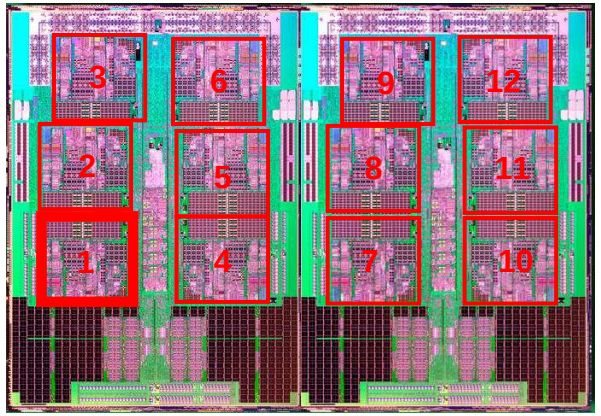


6 cores

0.384 s

Speedup 1.2

Palu (Magny-Cours)



12 cores

0.183 s

Speedup 3.1

Answers

- Your code will not get faster automatically. **You need to parallelize** in order to take advantage of increase in computer power.

But, wait a minute…

- Why does my code does not scale well onto the available cores?

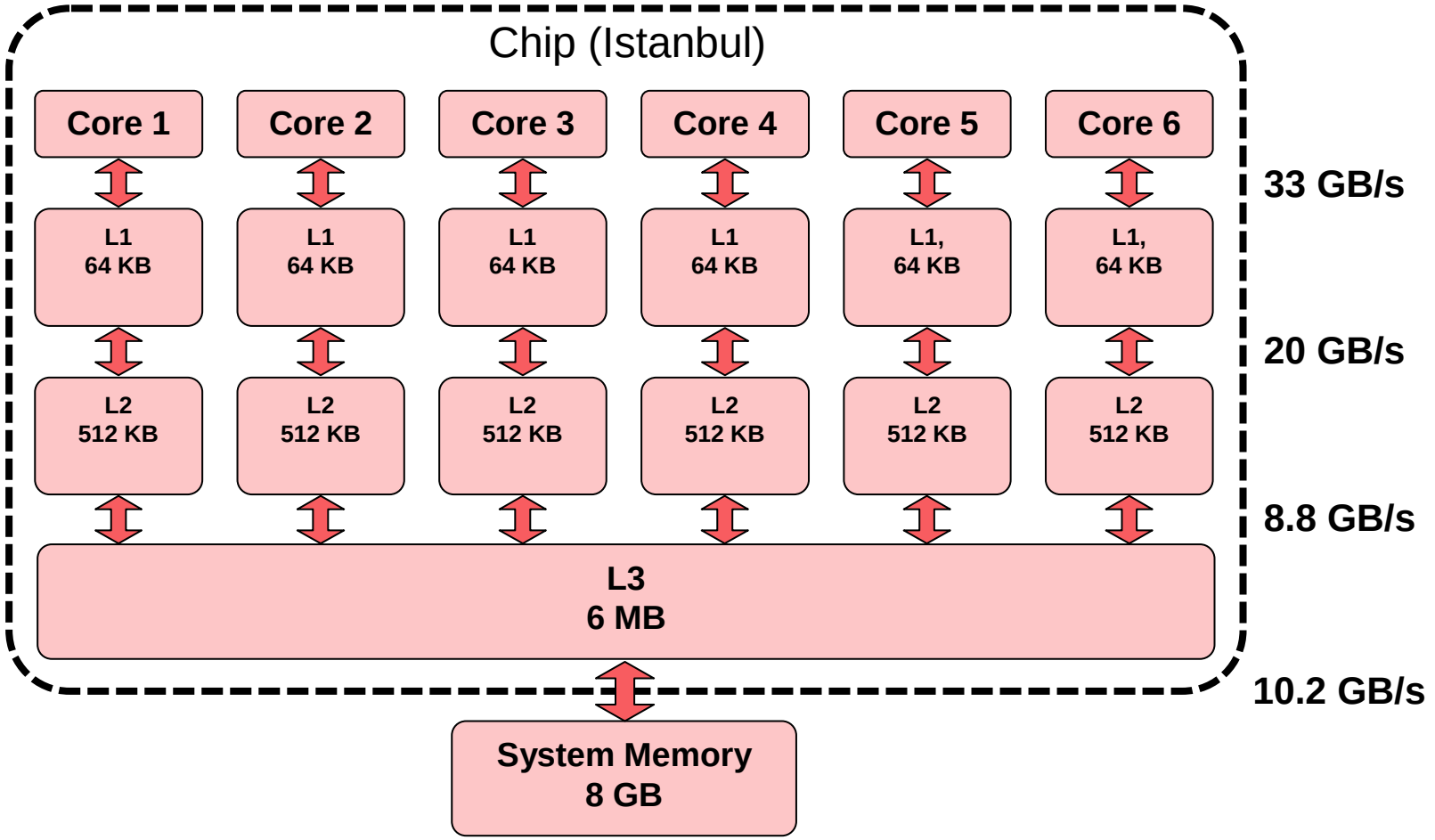
Maschine	**buin**	**rosa**	**palu**
# cores	4	6	12
Single core	0.802 s	0.836 s	0.631 s
All cores	0.558 s	0.459 s	0.183 s
Speedup	**1.4**	**1.8**	**3.4**

Code Analysis

```
do j = 1, niter
  do i = 1, nwork
    c(i) = a(i) + b(i) * ( a(i+1) - 2.0d0*a(i) + a(i-1) )
  end do
end do
```

- for **i** in **1** to **4096000**
  - load **a(i+1)**, **b(i)** from memory
  - computation (4 FLOPs)
    - 3 x add/subtract
    - 1 x multiply
  - store **c(i)**to memory
- **computation takes only 0.025 s**
- **memory access** of 3 fields with 4096000 entries of 8 Bytes = 94 MB

Memory Bandwidth



- 94 MB * 48 iterations / 10.2 GB/s = 0.43 s (measured 0.45)

Answers

- Your code will not get faster automatically. **You need to parallelize** in order to take advantage of increase in computer power

- Memory bandwidth does not increase at same speed as compute power. It is shared between increasing number of cores. **For stencil computations, memory bandwidth is a critical bottleneck**.

Example Code (Physics)

- Increase computational intensity (136 FLOPs)

```
do j = 1, niter
  do i = 1, nwork
    c(i) = a(i)**b(i) + sin(b(i)) * log(a(i))
  end do
end do
```

Maschine	buin	rosa	palu
# cores	4	6	12
Single core	17.6 s	17.3 s	18.8 s
All cores	4.4 s	2.9 s	1.6 s
Speedup	**4.0**	**6.0**	**12.0**

Answers

- Your code will not get faster automatically. **You need to parallelize** in order to take advantage of increase in computer power
- Memory bandwidth does not increase at same speed as compute power. It is shared between increasing number of cores. **For stencil computations, memory bandwidth is a critical bottleneck**.
- **For typical "physics" routines computation is the bottleneck.** For data coming from system memory, you need at least ~20 FLOPs per memory access, to be compute bound.

Loop Fusion

- Version 1

```
do j = 1, niter
  do i = 0, nwork+1
    b(i) = a(i+1) - 2.0d0*a(i) + a(i-1)
  end do
  do i = 1, nwork
    c(i) = a(i) + 0.1d0*( b(i+1) - 2.0d0*b(i) + b(i-1) )
  end do
end do
```

5 memory accesses
7 FLOPs

- Version 2

```
do j = 1, niter
  b(0) = a(1) - 2.0d0*a(0) + a(-1)
  b(1) = a(2) - 2.0d0*a(1) + a( 0)
  do i = 1, nwork
    b(i+1) = a(i+2) - 2.0d0*a(i+1) + a(i)
    c(i) = a(i) + 0.1d0*( b(i+1) - 2.0d0*b(i) + b(i-1) )
  end do
end do
```

4 memory accesses
6 FLOPs

Maschine	**buin**	**rosa**	**palu**
Version 1	3.7 s	4.5 s	3.7 s
Version 2	2.8 s	3.6 s	3.0 s
Speedup	**1.25**	**1.25**	**1.25**

HP2C Task 3: Dycore Rewrite

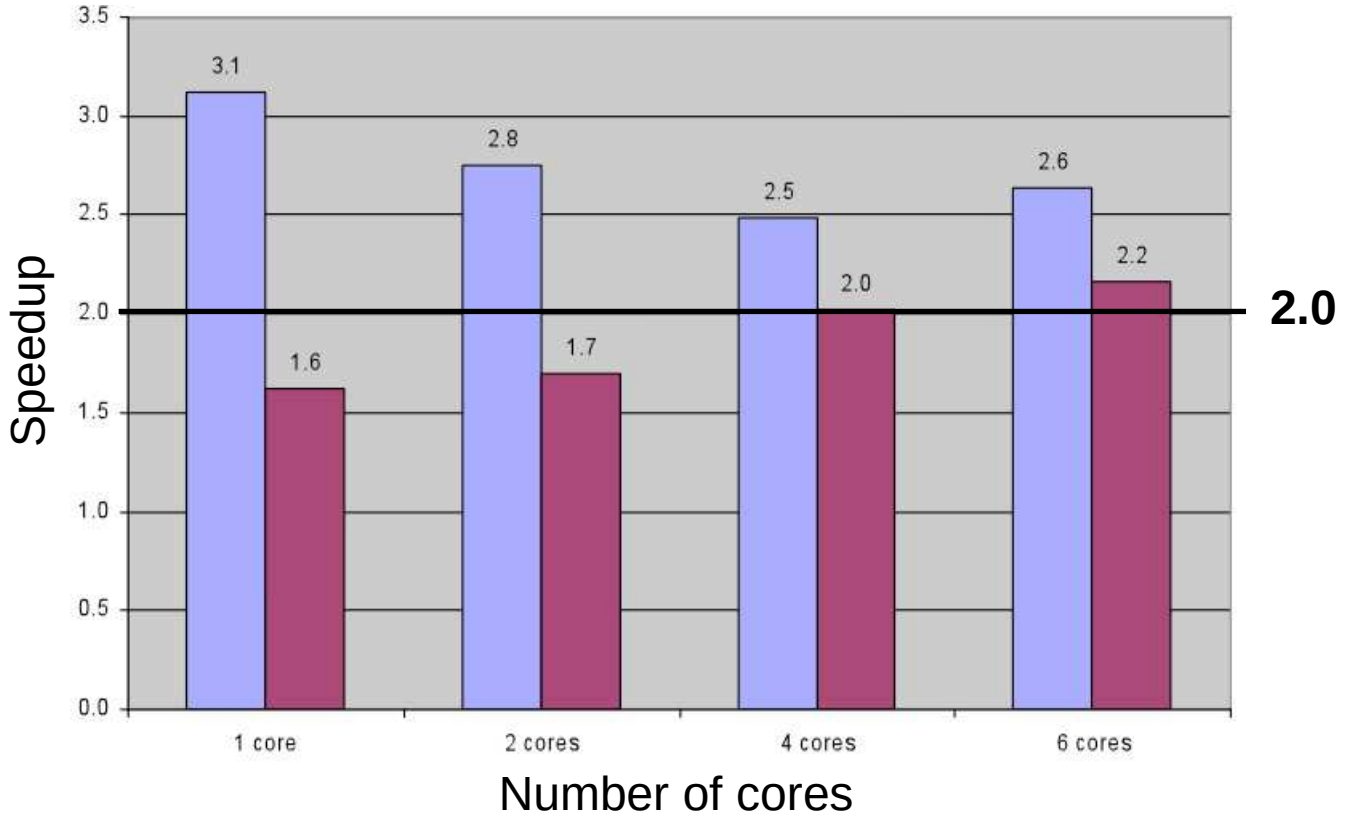
- Stencils are memory bandwidth limited
- Try to reduce memory traffic as much as possible
- Something like…

```
do j = 1, niter
  for 0 to nwork+1
    b = lap_1d( a )
  fuse
  for 1 to nwork
    c = a + 0.1d0 * lap_1d( b )
end do
```

- Language: C++
- Compile and run on both CPUs and GPUs
- High risk, high potential

HP2C Task 3: Feasibility Study

- Try out ideas on fast wave solver (~30% of runtime)



Summary

- Your code will not get faster automatically. **You need to parallelize** in order to take advantage of increase in computer power.
- Memory bandwidth does not increase at same speed as compute power. It is shared between increasing number of cores.
  - **“dynamics” routines ➜ memory bandwidth bound**
  - **“physics” routines ➜ compute bound**

  ➜ HP2C **Task 3**
- I/O bandwidth increases even slower rate than CPU or memory speed

  ➜ HP2C **Task 2**