Modern Computer Architecture (brief overview)

Jarmo Rantakokko

Aim: Give an overview of different types of systems and to give understanding on what to do and how to optimize the performance

Need to understand the hardware to be able to optimize the software!

1

UPPSALA UNIVERSITET

Modern computer architecture

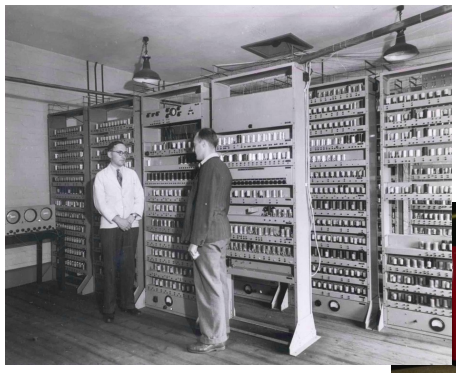
"All" modern computers are parallel computers (even single core computers)

Compare (historic overview):

- EDSAC (1949) 500kHz 100 Flop/s
- Cray 1 (1979) 80 MHz 4 Mflop/s
- Cray 1 (1983) 80 MHz 12 Mflop/s
- Intel P4 (2004) 3.8 GHz 7.6 Gflop/s
- Intel i7 (2010) 2.3 GHz 37 Gflop/s
- Nvidia (2012) 732MHz 3.95 Tflop/s

Explanation: Interior parallelism!

2



EDSAC
(Priceless, one of a kind)

Cray 1 ($5,000,000)

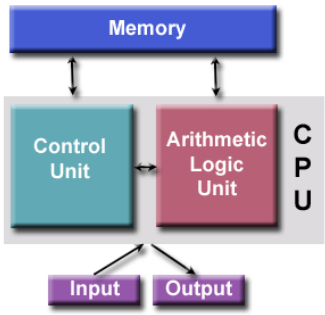
Nvidia ($3000)

3

The von Neumann model

For each instruction:

1. Fetch instruction
2. Decode instruction
3. Fetch operands
4. Execute instruction
5. Write result back



Observations (problems)

1. Each stage is performed sequentially
2. A lot of traffic to and from memory

> Several cycles for an instruction to complete.
> Slow devices/operations stalls execution.
> Contention on the data bus to memory.

4

UPPSALA UNIVERSITET

Solutions to run faster:

1. Increase clock rate
   ⇒ Slow devices (memory) still stalls execution
   ⇒ More contention on the memory bus
   ⇒ High power consumption
   Increase freq, increase voltage, $p \sim f * v^2$
   (Energy cost, cooling problem, battery time)

2. Introduce parallelism
   a. Within a CPU
   b. Multiple CPUs/cores

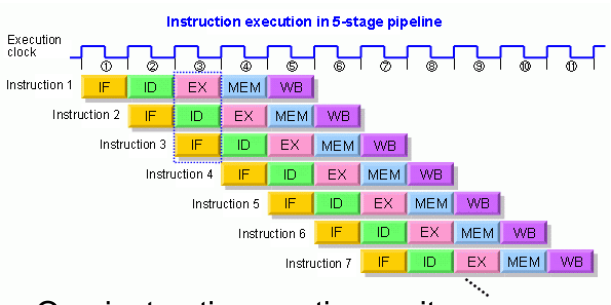
5

UPPSALA UNIVERSITET

Parallelism within a processor:

1. **Parallel busses**
   Wider data bus
   Separate data and instruction bus

2. **Instruction pipeline**

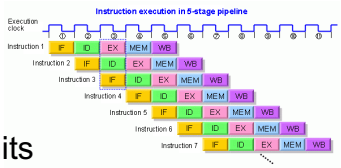


⇒ One instruction per time unit

6

UPPSALA UNIVERSITET

How does the performance depend on the pipeline?



- Largest machine instruction limits the time unit (the speed of pipeline)
- Uniform stages (no dead time, higher efficiency)
- Number of stages (more parallelism/speedup)
- Number of consecutive instruction (startup time minor)

Short, uniform instruction length => RISC processor (Reduced Instruction Set Computer)

Super-pipelined processor: Extremely short and simple instructions (long pipelines). Can run with higher frequency.

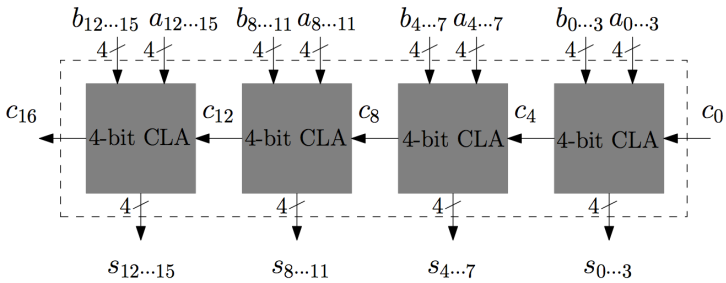
Problem: Branches in code, disrupts the pipeline
Branch prediction in hardware

7

UPPSALA UNIVERSITET

3. Arithmetical pipelines

Floating point operations are heavy operations, can be decomposed into smaller operations, for example, decomposing an adder to adding a few bits at a time.



Useful for loops with arithmetic operations

```
for (i=0;i<n;i++)
    s[i]=a[i]+b[i];
```

Without pipeline: 4n t.u.
With pipeline: 4+n-1 t.u.

8

UPPSALA UNIVERSITET

4. Multiple operating units

Multiple units for floating point operations (add,mult), integer arithmetic unit, logic unit, branch unit, etc,

⇒ Parallel instruction stream with 2-4 flops/cycle

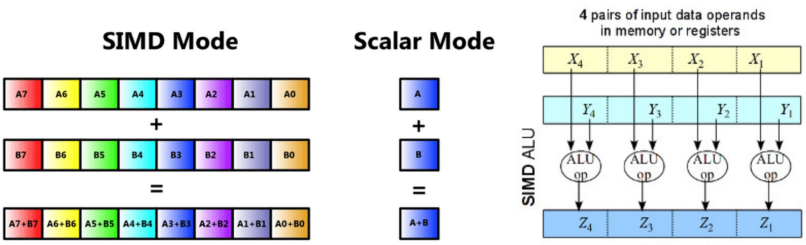
Problem:
Serial sections (ordered instructions) limit performance.

- Use *out-of-order* and *speculative* execution in hardware, i.e., precompute results without knowing if the instruction will execute or not.
- Use multiple software threads and fill the units with instructions from the parallel threads*, multithreading.*

9

UPPSALA UNIVERSITET

5. Vector operations, SIMD



SIMD operation adding 8 elements simultaneously

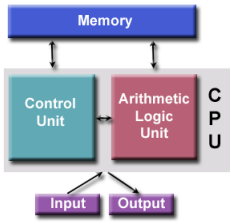
SIMD: Single Instruction Multiple Data, the same operations are to be performed on multiple data elements simultaneously using vector registers MMX, SSE, AVX, AVX512.

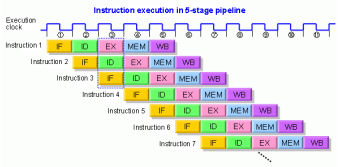
10

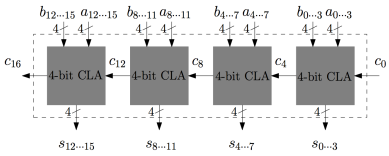
UPPSALA UNIVERSITET

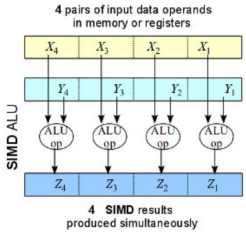
Parallelism within a processor:

1. **Parallel busses**
2. **Instruction pipeline**
3. **Arithmetical pipelines**
4. **Multiple operating units**
5. **Vector operations SIMD**









11

UPPSALA UNIVERSITET

MEMORY PROBLEM:
Enhancements (1) - (5) => Memory can't keep up delivering data at processor speed!

Types of memory

DRAM: Dynamic Random Access Memory
Charged based devices, needs to be refreshed at read/write – takes time ~10^{-8} s
Cheap but slow, use for main memory.
(Improvements: SDRAM, DDR SDRAM, RDRAM using multiple memory banks, can overlap accesses)

SRAM: Static Random Access Memory
Gate based devices (transistors)
No need for refreshment, fast but expensive, use for registers and cache

12

UPPSALA UNIVERSITET

Memory Hierarchies

To keep costs down, create memory hierarchy:

Speed	Registers	Head
	Caches	Address book
	Main memory	Phone book
Size	Virtual mem (disk)	Data base

Analogy: Memory hierarchy – Phone numbers

13

UPPSALA UNIVERSITET

Caches: L1 Cache (~128kB, on chip)
L2 Cache (~4MB, on/off chip)
L3 Cache (+8MB, off chip)

Types of caches (techniques to store/replace data):
- direct mapped – pages alphabetical order
- fully associative – blank pages
- set associative – free sections, each section ordered

Memory hierarchies makes it very important with **data layout** and **data accesses** in your codes!

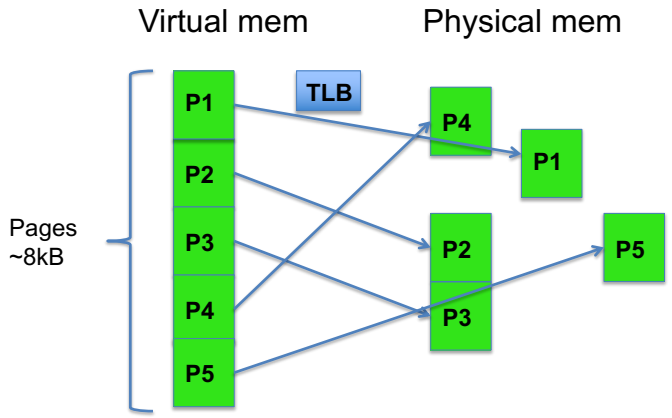
Register
Cache — Cache miss
Main memory — Page fault, TLB miss
Virtual memory

14

UPPSALA UNIVERSITET

TLB: Translation Lookaside Buffer

Fast translation table between virtual and physical mem

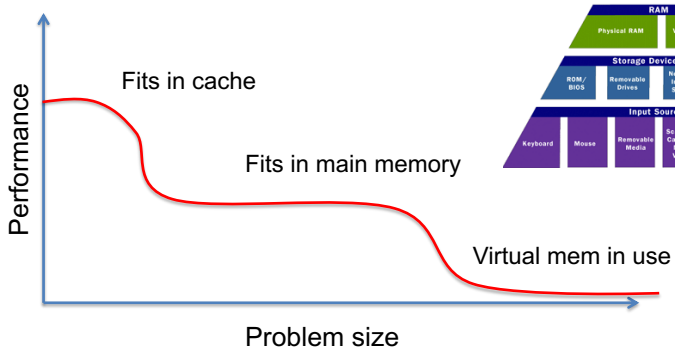


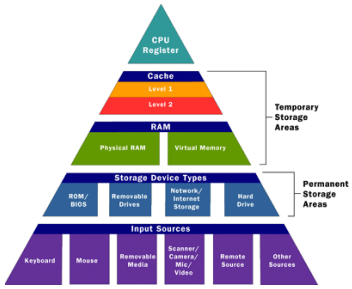
If the page's location is not in TLB we get a miss and need to find the page in the physical memory, costly!

15

UPPSALA UNIVERSITET

Performance vs memory





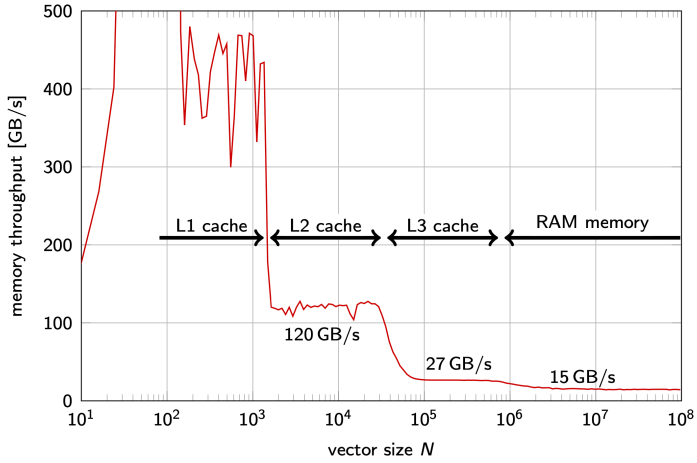
Note: Bad access pattern (bad cache reuse) in code will move performance location to the right and good access pattern (cache optimization) will move performance location to the left!

16

UPPSALA UNIVERSITET

STREAM benchmark (https://www.cs.virginia.edu/stream/)

STREAM triad is a benchmark of the **memory hierarchy**!



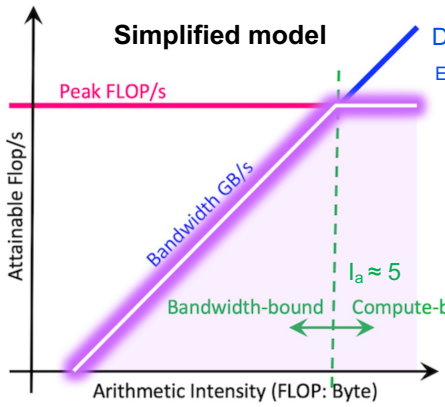
STREAM triad: A(1:N)=B(1:N)+s∗C(1:N)

17

UPPSALA UNIVERSITET

Roofline model

(https://en.wikipedia.org/wiki/Roofline_model
https://www.youtube.com/watch?v=lx1oEGtZNK8)



Data transfer = I_a x B_w

Ex: 0.1Flop/B x 10GB/s = 1GFlop/s

Performance = min { Data transfer, Computation }

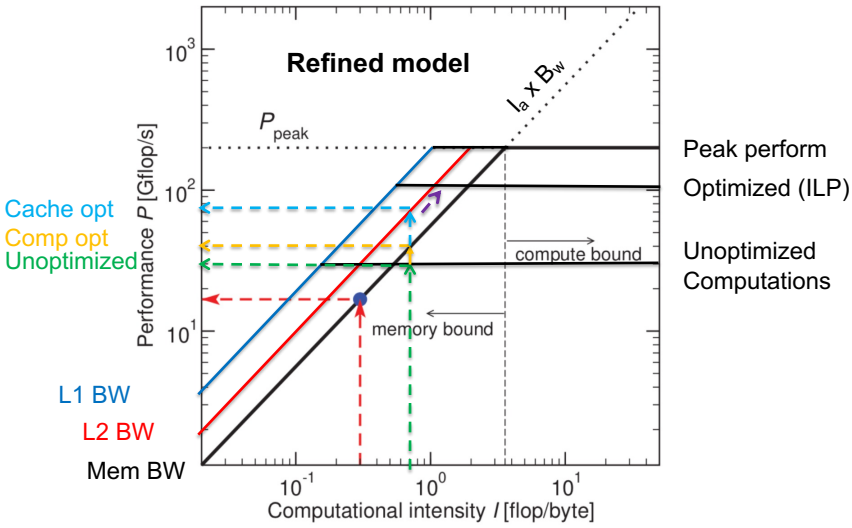
(Note: The model is simplifying both hardware and software, no latency, no startup, long loops, no integer ops, no branches, etc. Hard to get numbers correct for an application. But it gives us important insight and understanding.)

$$\text{arithmetic intensity } I_a = \frac{\text{number of floating point operations}}{\text{memory access}}$$

Triad: $I_a = 2N/(3Nx8) = 1/12$ (Bandwidth-bound)
MxM: $I_a = 2N^3/(3N^2x8) = N/12$ (Compute-bound – if fits in mem)

18

Roofline model



Note: The computational intensity can also increase with optimization, i.e., we re-use the data in cache and do more computations before preplacing

19

In summary:

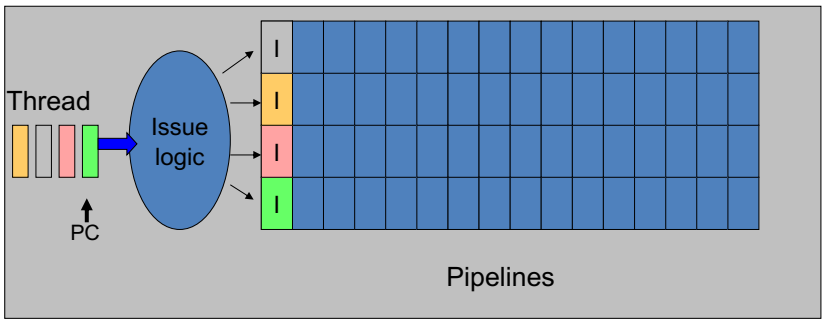
A “traditional” processors is very parallel!
But, the parallelism is on a low level and “hidden” for programmer. Compiler exploits it with loop unrolling, reordering, merging, splitting etc., if using aggressive optimization flags (-fast, -O3, -O5, -unroll, or similar). **Manual tuning of code** is still necessary!

Moreover, **cache optimization** becomes increasingly important, e.g., by exploiting temporal and spatial data locality and cache blocking, we can in data intensive cases improve the performance with a factor of 10x.

Applications can either be compute-bound or memory-bound depending on the computational intensity, need to know what to optimize for (**Roofline model**).

20

Multicore processor design



In the “traditional” processor we run one thread and issue instructions from this to the multiple pipelines.

21

Problems with “traditional” processor design:

#1: Running out of ILP
Not enough instructions to feed pipelines

#2: Wire delay is starting to hurt
Thinner wires, higher resistance, slower speed

#3: Memory is the bottleneck
Slow memory accesses stalls the execution

#4: Power is the limit, $P \sim F*V^2$
Cooling problem, high energy cost, low battery time

22

UPPSALA UNIVERSITET

Solving all the problems, exploring parallelism:

#1: Running out of ILP
> Feed one CPU with instructions from many threads

#2: Wire delay is starting to hurt
> Multiple small cores with private L1$, shorter paths

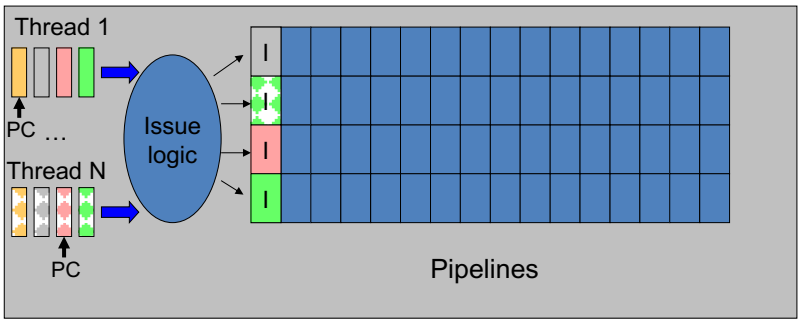
#3: Memory is the bottleneck
> Overlap memory accesses from many threads

#4: Power is the limit
> Multiple cores, lower F and V, better performance

23

UPPSALA UNIVERSITET

SMT: Simultaneous MultiThreading

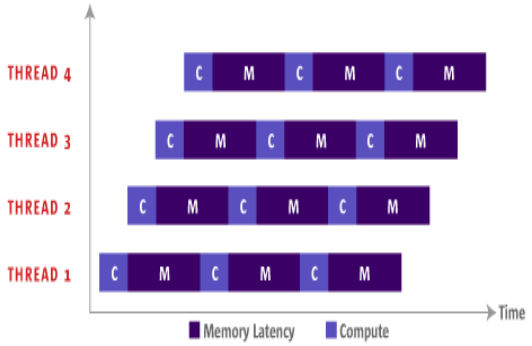


Can feed the multiple pipelines with instruction from many threads. Can have hardware and software threads. Typically, improves performance with 10-20%.

24

UPPSALA UNIVERSITET

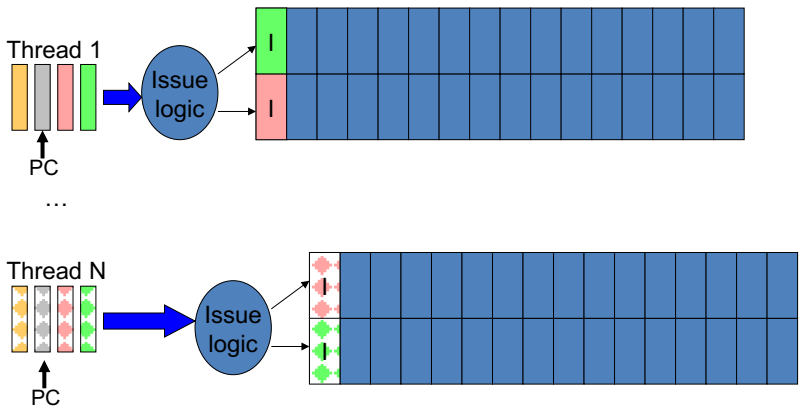
Overlap memory accesses with computations



25

UPPSALA UNIVERSITET

CMP: Chip MultiProcessor (MultiCore)



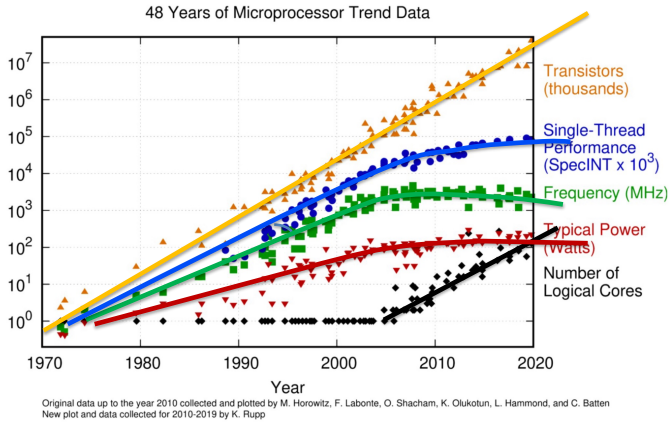
Up to N times performance improvment.

26

UPPSALA UNIVERSITET

Consequences:

- Non-parallelized programs will not utilize the full compute potential
- Non-parallized programs can run slower on the new CPUs (lower freq)

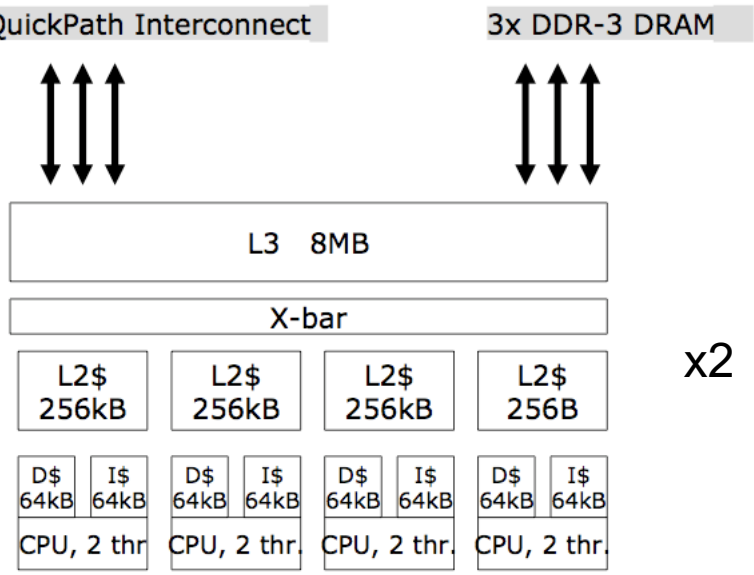


=> Need parallel programming even for one single CPU !

27

UPPSALA UNIVERSITET

Intel Core i5/i7, Xeon E-serie (IT Linux Servers)



28

UPPSALA UNIVERSITET

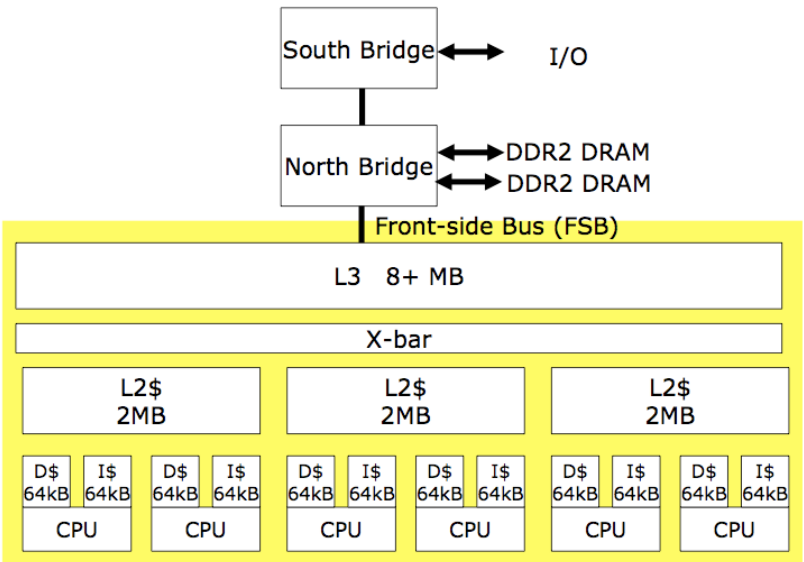
IT Linux servers:

```
jra13969@arrhenius:~$ lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                16
On-line CPU(s) list:   0-15
Thread(s) per core:    2
Core(s) per socket:    4
Socket(s):             2
NUMA node(s):          2
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 26
Model name:            Intel(R) Xeon(R) CPU           E5520  @ 2.27GHz
Stepping:              5
CPU MHz:               2400.270
CPU max MHz:           2268,0000
CPU min MHz:           1600,0000
BogoMIPS:              4533.84
Virtualization:        VT-x
L1d cache:             32K
L1i cache:             32K
L2 cache:              256K
L3 cache:              8192K
NUMA node0 CPU(s):     0-3,8-11
NUMA node1 CPU(s):     4-7,12-15
```

29

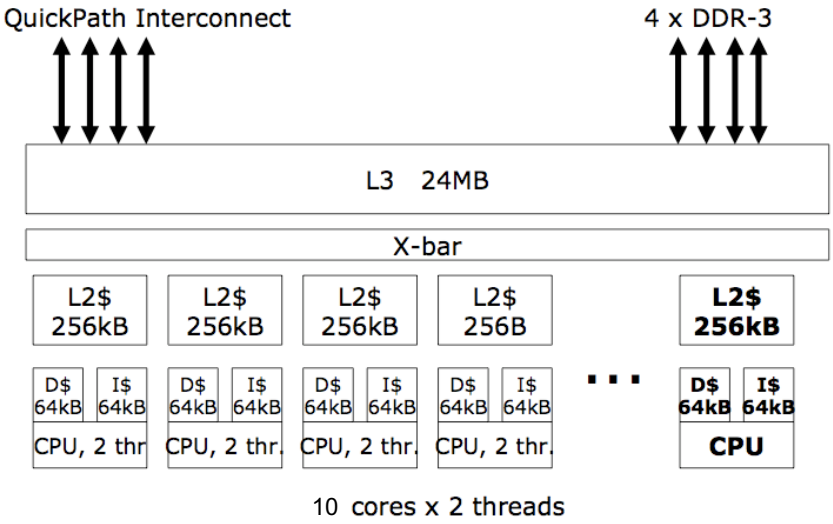
UPPSALA UNIVERSITET

Intel Core i7, 6 core (Macbook pro)



30

Intel Core i7/i9, Xeon V-series (UPPMAX)



31

Parallel Computers at UU

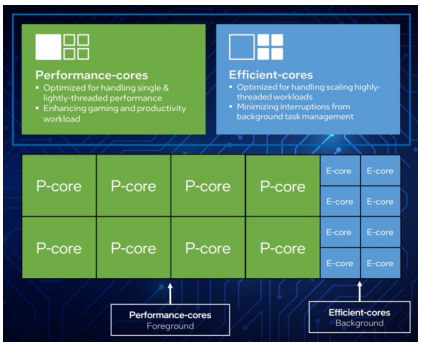
UPPMAX, Rackham (2017)

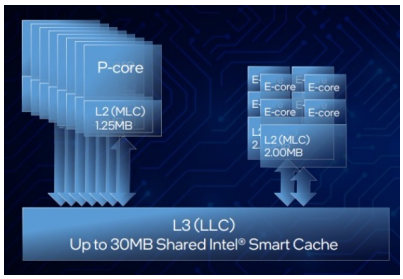
- 486 nodes x 2 proc x 10 cores = 9720 cores
- FDR Infiniband interconnect (high speed data)
- Intel Xeon E5-2630v4, 2.2GHz (3.1GHz)
- Peak performance ~ ½ PetaFlop/s
- Total memory 70 TeraByte RAM
- Will be used in PDP-course, period 4!

32

UPPSALA UNIVERSITET

Intel Core i9-12900K, 12:th gen (2021)





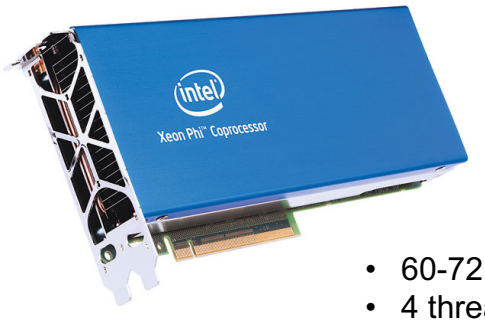
16 cores total, 8 P-cores run dual threaded (16 threads), 8 E-cores run single threaded (8 threads), in total 24 threads running simultaneously. Typically you would run non-homogenous interactive tasks on P-cores and computationally intensive homogenous task on E-cores.

Shared 30MB L3-cache for all cores, shared 4MB L2-cache for all E-cores and private 1.25MB L2-cache for each P-core.

33

UPPSALA UNIVERSITET

Intel Xeon Phi Co-processor (Many Integrated Core Arcitechture)



- 60-72 Compute cores
- 4 threads per core
- Up to 1.2 Tflops
- Tianhe-2 Supercomputer

34

UPPSALA UNIVERSITET

Graphical Processing Units (GPU)

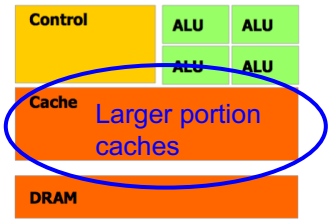
Main vendors NVIDIA, AMD

Architectural features:
- Many simple processing elements (16-2668)
- 1000s – 1 000 000s of threads
- Hardware thread scheduling (1 cycle)
- Focus on throughput (data parallel tasks)
- Limited memory (small on chip mem)
- Limited bandwidth CPU ⇔ GPU (bottleneck)

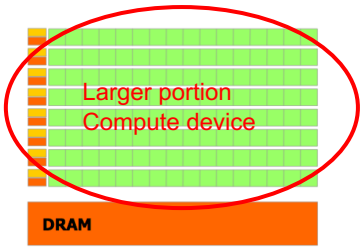
35

UPPSALA UNIVERSITET

CPU vs GPU



CPU



GPU

- Big caches, hierarchy
- Branch predictors
- Out-of-order
- Multiple-issue
- Speculative execution
- Double-precision

Reduce mem latency with caches and hide with other instructions

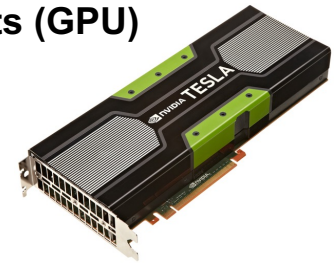
- None or small caches
- 1000's of threads
- 1 cycle context switch
- SIMT instructions (32 threads)
- Single precision

Hide mem latency with work from other threads

36

UPPSALA UNIVERSITET

Graphical Processing Units (GPU)



NVIDIA GeForce 650M
- 384 cores
- 650 Gflops SP
- 1 Gbyte Mem
- Power 64W
- Price $120

(C.f. CPU: 4 core, 2.3GHz, 4 flop/cycle => 37 Gflops DP)

NVIDIA Tesla K20X
- 2668 cores
- 3.95 Tflops SP (1.31 DP)
- 6 Gbyte Mem
- Power 235W
- Price > $3200

(Used in Titan supercomputer)

GPU is a significant source of computational power!
Accelerator based programming, 1TD054, period 1

37