



Wrocław  
University  
of Science  
and Technology

# Large Scale Data Processing

## Lecture 1 – Basic notation, definitions

dr inż. Tomasz Kajdanowicz, Roman Bartusiak, Piotr Bielak

November 9, 2019



HR EXCELLENCE IN RESEARCH

# Overview

Big data – 5Vs

Types of processing

Flynn's taxonomy

Compilers

Python - GIL

# Overview

Big data – 5Vs

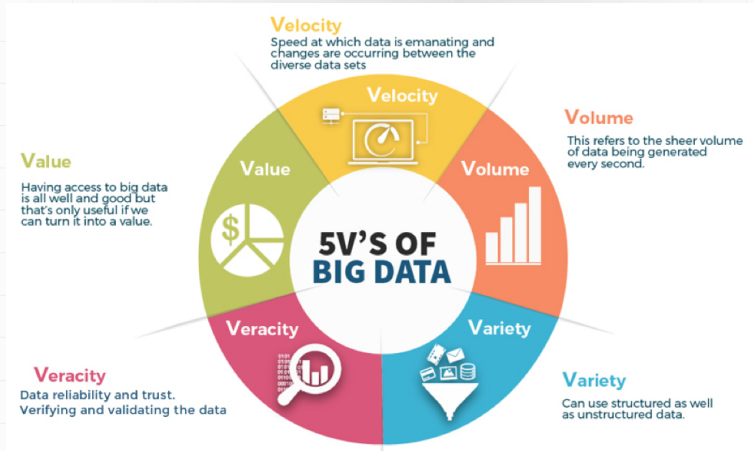
Types of processing

Flynn's taxonomy

Compilers

Python - GIL

# Big data – 5Vs



## Big data – 5Vs

- ▶ **Volume** – enormous volumes of data,
- ▶ **Velocity** – data flows in time from multiple sources and with varying speed,
- ▶ **Value** – data can be hard to obtain,
- ▶ **Veracity (wiarygodność)** – biases, noise and abnormality in data,
- ▶ **Variety** – many sources and types of data both structured and unstructured,

Sometimes this definition is extended to 7Vs:

- ▶ **Validity** – if data correct and accurate for the intended use,
- ▶ **Volatility** – how long is data valid and how long should it be stored,

# Overview

Big data – 5Vs

Types of processing

Flynn's taxonomy

Compilers

Python - GIL

# Types of processing

- ▶ Sequential processing
- ▶ Distributed processing
- ▶ Parallel processing
- ▶ Concurrent processing

# Sequential processing

## Types of processing

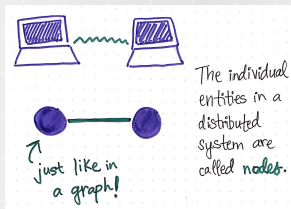
- ▶ processing that occurs in the order that it is received
- ▶ processor inevitably executes the same program



# Distributed processing

## Types of processing

- ▶ more than one computer (or processor) run an application
- ▶ memory is distributed!
- ▶ includes parallel processing in which a single computer uses more than one CPU to execute programs



- ▶ nodes run operations, that decomposes original large problem
- ▶ operations within a node are fast; communication between nodes is slow
- ▶ nodes operates on their own clocks

# Parallel processing

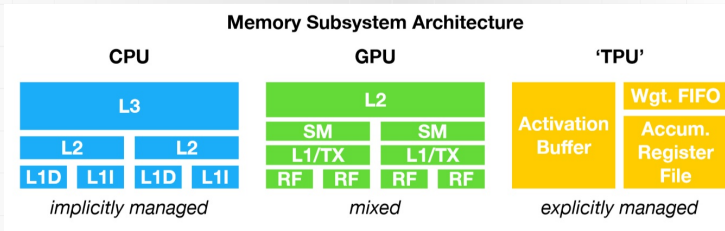
## Types of processing

- ▶ Programs use parallel hardwares to execute computation more quickly
- ▶ Possible hardware:
  - ▶ multi-core processors
  - ▶ symmetric multiprocessors
  - ▶ graphics processing unit (GPU)
  - ▶ field-programmable gate arrays (FPGAs)
  - ▶ computer clusters
- ▶ Parallel programming requires to think about:
  - ▶ How does code divide original huge problem into smaller sub-problems?
  - ▶ Which is the optimal use of parallel hardware?

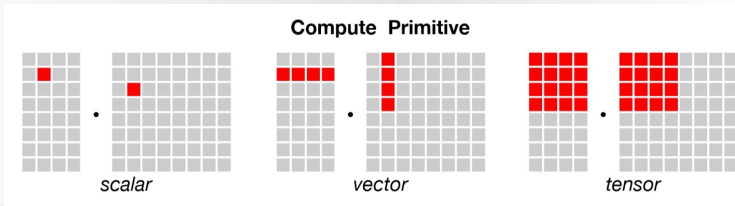
# CPU vs GPU vs TPU

## Types of processing

### ► Memory Subsystem Architecture



### ► Compute Primitive



# CPU vs GPU vs TPU

## Types of processing

- ▶ Dimension of data:
  - ▶ CPU: 1 X 1 data unit
  - ▶ GPU: 1 X N data unit
  - ▶ TPU: N X N data unit
- ▶ Performance
  - ▶ CPU can handle tens of operation per cycle
  - ▶ GPU can handle tens of thousands of operation per cycle
  - ▶ TPU can handle upto 128000 operations per cycle
- ▶ Purpose
  - ▶ CPU - designed to solve every computational problem in a general fashion; cache and memory optimal for any general programming problem
  - ▶ GPU - designed to accelerate the rendering of graphics
  - ▶ TPU - designed to accelerate deep learning tasks developed with TensorFlow

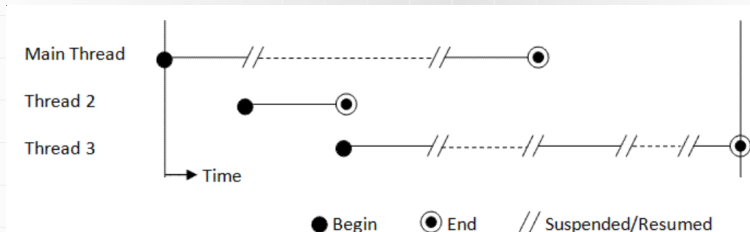
# Concurrent processing

## Types of processing

- ▶ concurrency is when multiple sequences of operations are run in overlapping periods of time
- ▶ task A and task B both need to happen independently of each other, and A starts running, and then B starts before A is finished
- ▶ address limits of resources
- ▶ taxonomy:
  - ▶ multitasking
  - ▶ multiprocessing
  - ▶ preemption: preemptive, cooperative

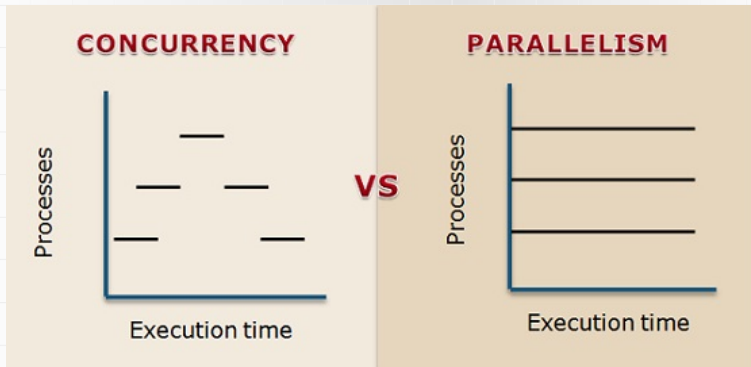
# Concurrency example

## Types of processing



# Concurrent vs parallel

## Types of processing



# Overview

Big data – 5Vs

Types of processing

Flynn's taxonomy

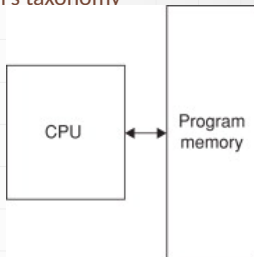
Compilers

Python - GIL

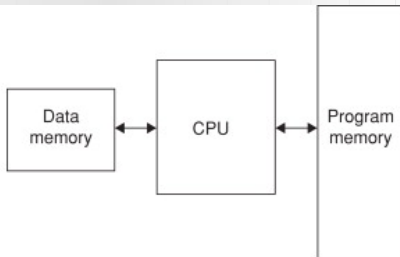


# Computer architecture recap

Flynn's taxonomy



(a) Von Neumann architecture



(b) Harvard architecture

Criterion	Architecture	
	(a)	(b)
Memory/Bus	one	two
Complexity	simple	complicated
Single instruction	two clock cycles	one clock cycle
Performance	low	high (pipelining)
Cost	cheap	high

# Data and Instruction streams

## Flynn's taxonomy

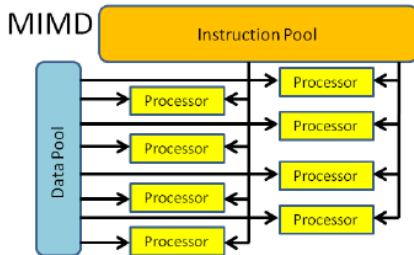
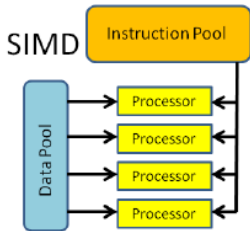
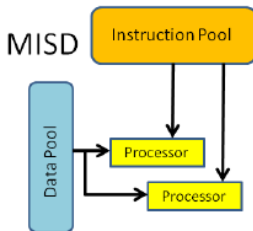
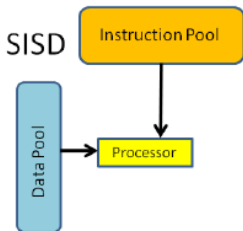
In Flynn's taxonomy we use following criteria to define system architectures:

- ▶ number of **instructions** stream(s) - single or multiple,
- ▶ number of **data** stream(s) - single or multiple,

Hence we get following acronyms: **(S/M) I (S/M) D**

# Architectures

## Flynn's taxonomy



# Examples

## Flynn's taxonomy

- ▶ SISD – sequential computer; von Neumann architecture; many PCs before 2010 and mainframes
- ▶ SIMD – GPU; modern CPUs with vectorization
- ▶ MISD – systolic computer; fault-tolerant systems
- ▶ MIMD – cluster, where each processor is programmed separately; Intel Xeon Phi; multi-core superscalar processors; distributed systems

# Overview

Big data – 5Vs

Types of processing

Flynn's taxonomy

Compilers

Python - GIL

# Compilation process, optimizations

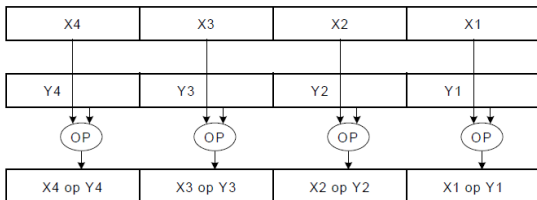
## Compilers

- ▶ we won't get into the details of the compilation process,
- ▶ programming languages:
  - ▶ interpreted (e.g., Python, JavaScript),
  - ▶ compiled (e.g., C, C++, Rust),
  - ▶ mixed (e.g., Java - Bytecode+JVM, Python in some cases),
- ▶ interpreted PLs are in general slower than compiled ones (however there is JIT),
- ▶ this is caused by heavy optimizations, which are applied in the compilation process, e.g.:
  - ▶ *removal of unused code* – if the compiler detects that some variable, function etc. is declared, but is never used, then all instructions concerning that variable are removed (can be problematic in some cases like embedded systems; see: *volatile* in C/C++)
  - ▶ *unrolling loops into vector operations* – ...

# Vectorization

## Compilers

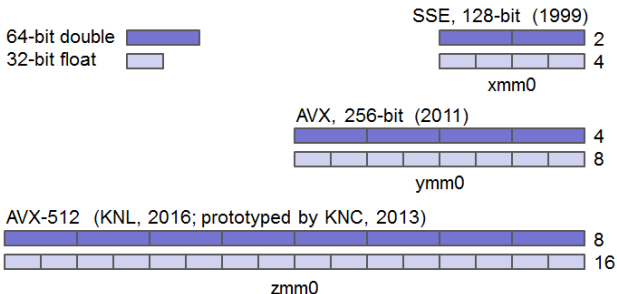
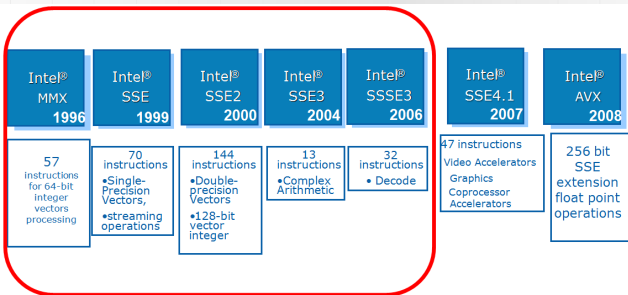
- ▶ 32/64-bit CPUs use general purpose registers with a capacity of 32/64 bits each,
- ▶ however there are some *special registers* with a size equal to the multiple of the architecture size (multiples of 32/64 bits),
- ▶ operations on these registers take one CPU cycle,
- ▶ hence we can speed up computations



OM15148

# Vector registers

## Compilers





# Overview

Big data – 5Vs

Types of processing

Flynn's taxonomy

Compilers

Python - GIL

# Python - GIL

---

```
1  import time
2
3
4  def countdown(n):
5      while n > 0:
6          n -= 1
7
8
9  def main():
10     n = 50000000
11
12     st = time.time()
13     countdown(n)
14     end = time.time()
15
16     print('Processing took:', end - st, '(s)')
17
18
19 if __name__ == '__main__':
20     main()
```

---

It takes about **3.09 seconds**

# Python - GIL

---

```
1  from threading import Thread
2  import time
3
4
5  def countdown(n):
6      while n > 0:
7          n -= 1
8
9
10 def main():
11     n = 50000000
12
13     t1 = Thread(target=countdown, args=(n//2,))
14     t2 = Thread(target=countdown, args=(n//2,))
15
16     st = time.time()
17     t1.start(); t2.start()
18     t1.join(); t2.join()
19     end = time.time()
20
21     print('Processing took:', end - st, '(s)')
22
23
24 if __name__ == '__main__':
25     main()
```

---

It takes about **5.37 seconds!**

# Python - GIL

- ▶ the main reason for that is the **Global Interpreter Lock**,
- ▶ from Python 3.2 there were some improvements,
- ▶ GIL ensures that only one thread in the interpreter runs at a given time,
- ▶ why? needed for implementation simplification (memory managements, calls to external C functions etc.),

- [illegible]

- When a thread is running, it holds the GIL
- GIL released on I/O (read,write,send,recv,etc.)

# Python - GIL

- ▶ the problem occurs especially for CPU bound threads (very little I/O),
- ▶ additionally there is no smart thread scheduling algorithm,
- ▶ this could lead to a situation where only one thread is running all the time and the others wait,
- ▶ together with a special *check* mechanism in the Python interpreter implementation this can cause extreme slowdown of running times

# Large Scale Data Processing

## Lecture 1 – Basic notation, definitions

dr inż. Tomasz Kajdanowicz, Roman Bartusiak, Piotr Bielak

November 9, 2019