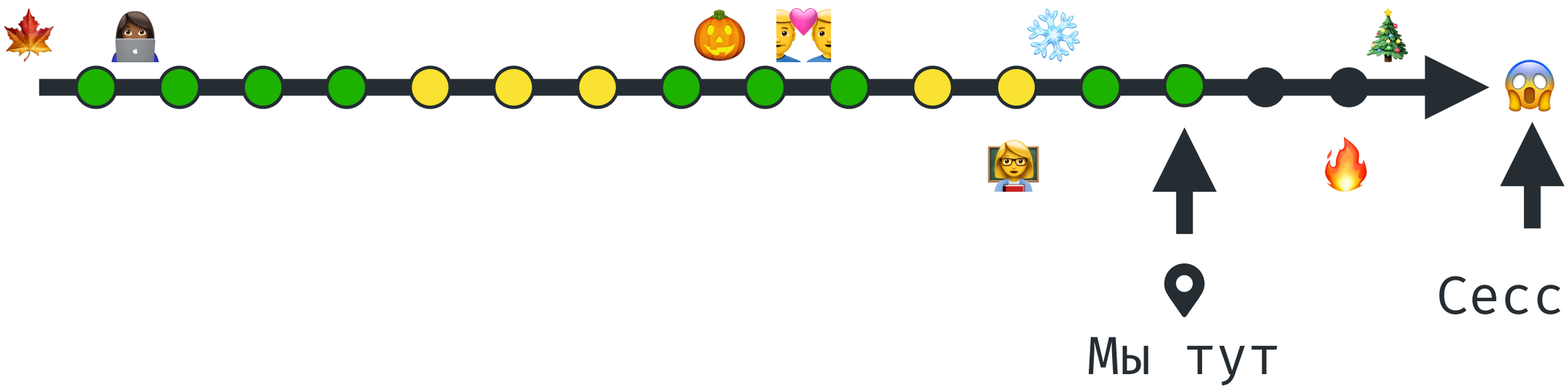
ИНФОРМАТИКА

```
~/polytech $ date
осень 2019
~/polytech $ name
константин кориков
~/polytech $ topic
Элементы конкурентного программирования
```

ГДЕ МЫ?



О ЧЁМ ПОГОВОРИМ?

1. Parallelism ≠ Concurrency
2. Threads и processes
3. Синхронизация
4. Actors
5. GPU: параллелизм данных
6. Лямбда-архитектура

Parallelism ≠ Concurrency

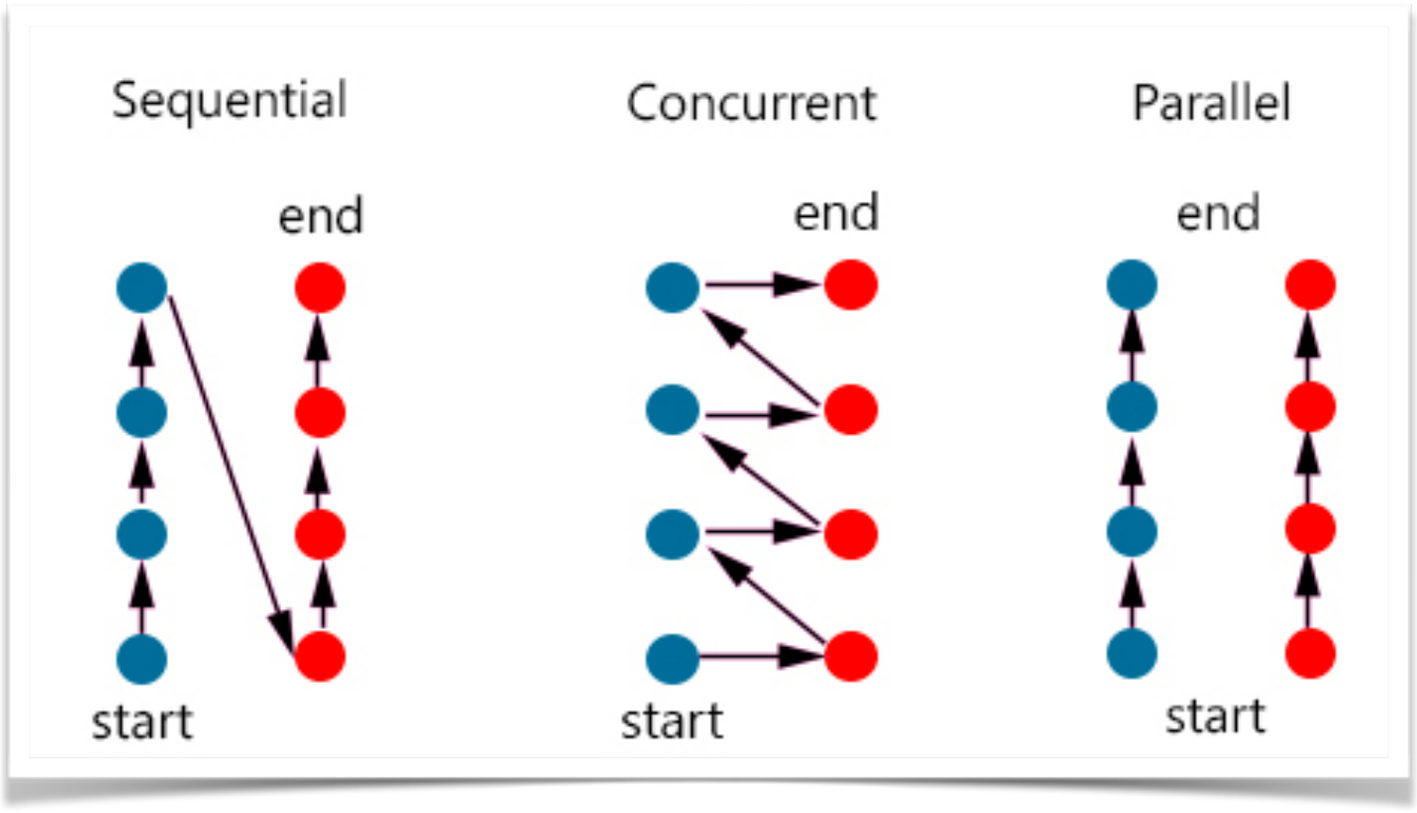
Parallelism ≠ Concurrency

Конкурентные задачи:
пересекаются во времени,
могут выполняться параллельно
или порционно

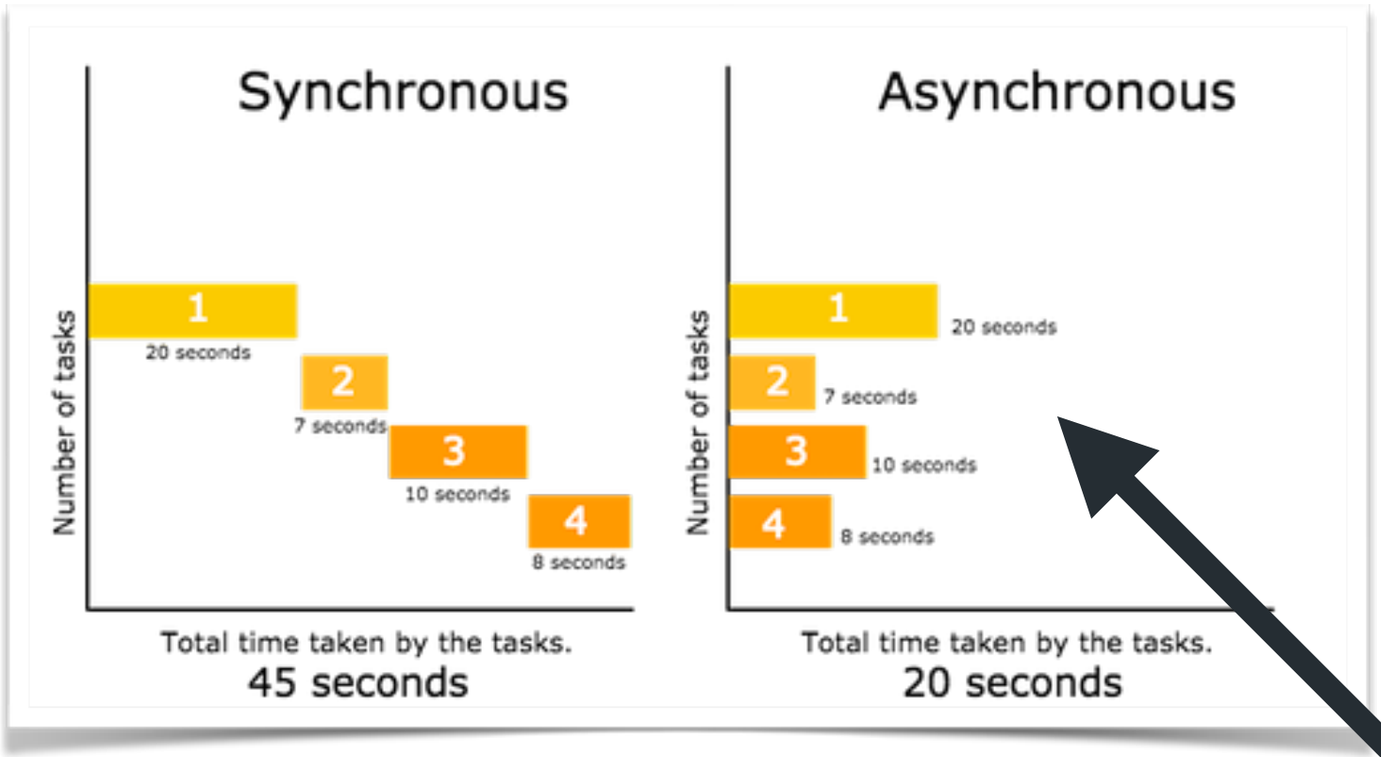
Parallelism ≠ Concurrency

Параллельные задачи:
выполняться одновременно

Parallelism ≠ Concurrency

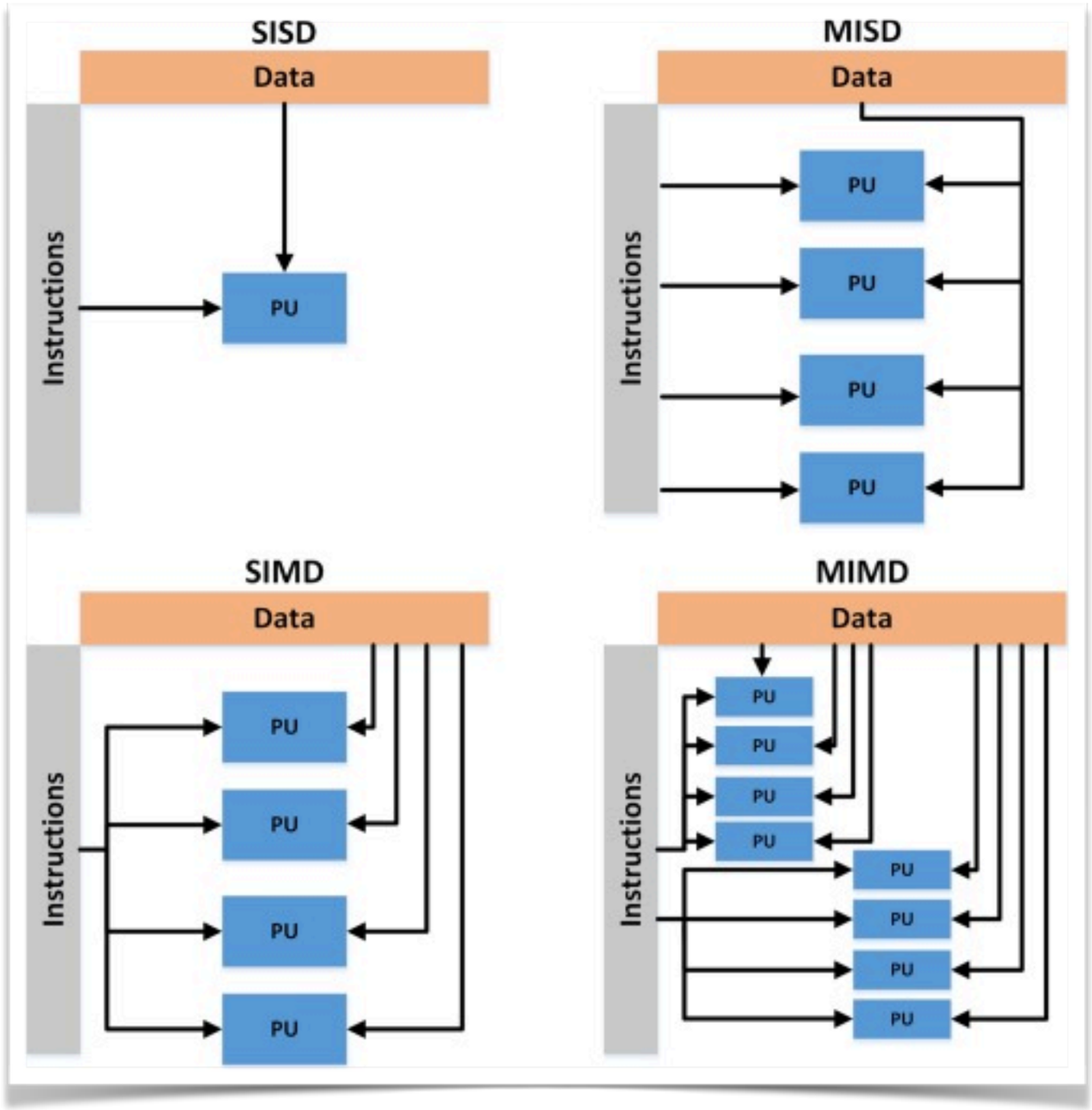


Asynchronous и synchronous



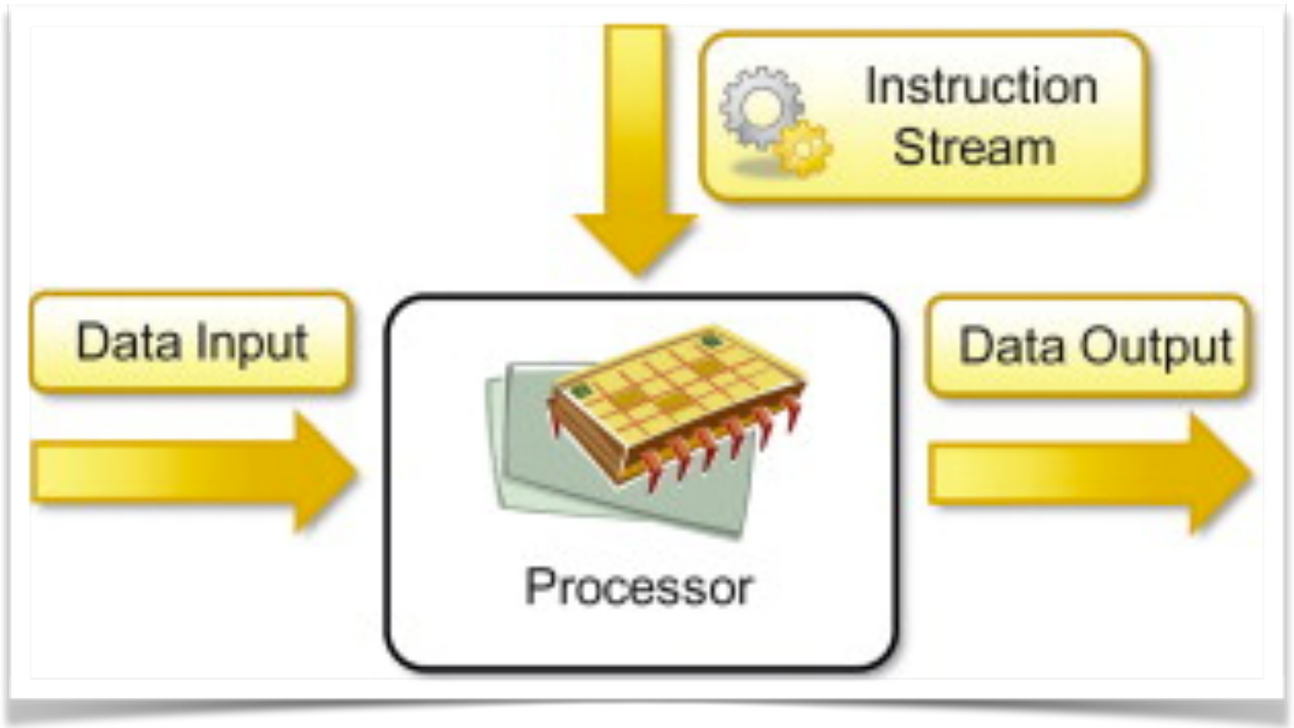
Могут вызывать callback-функции

Железо



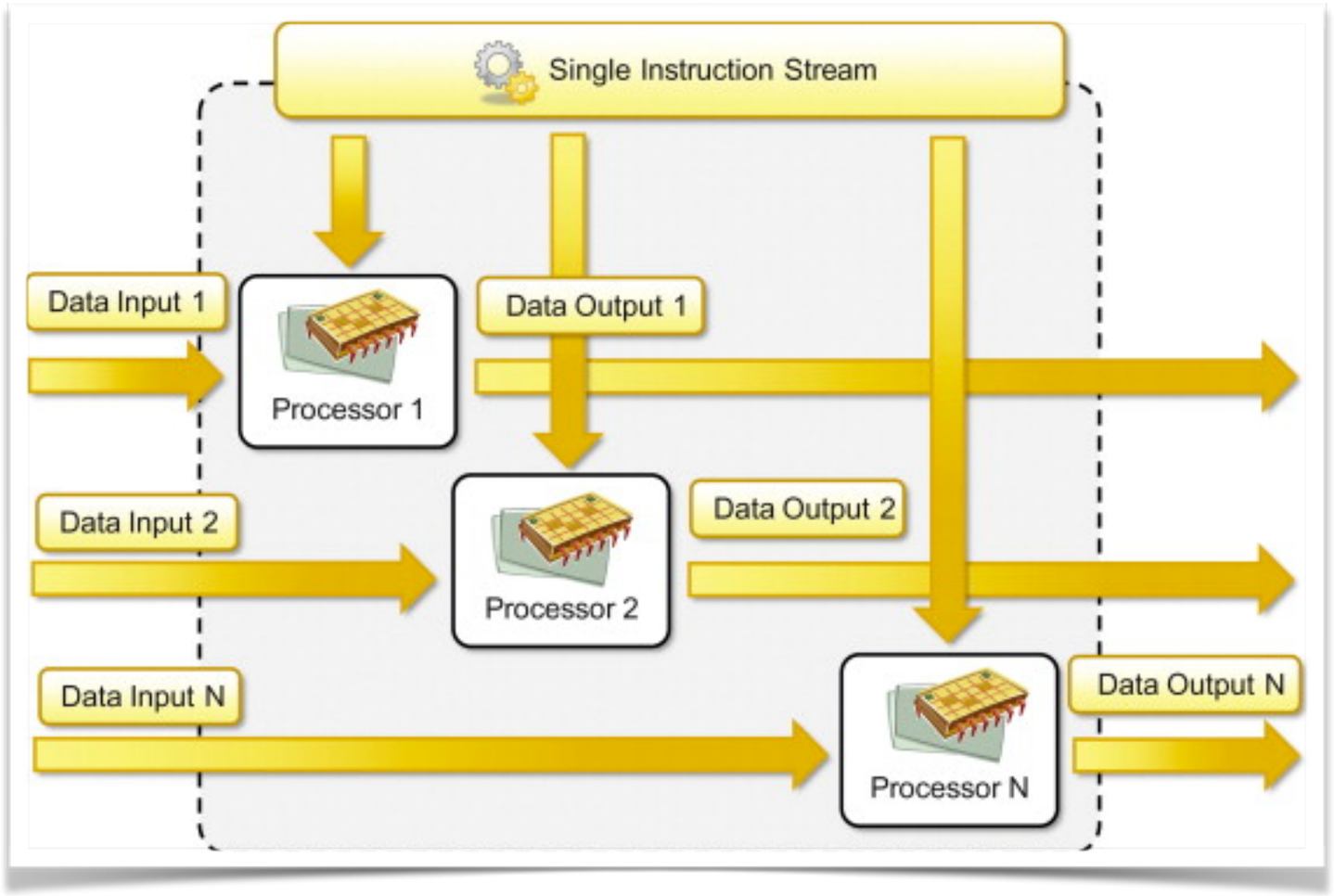
Классификация Flynn’a

Железо



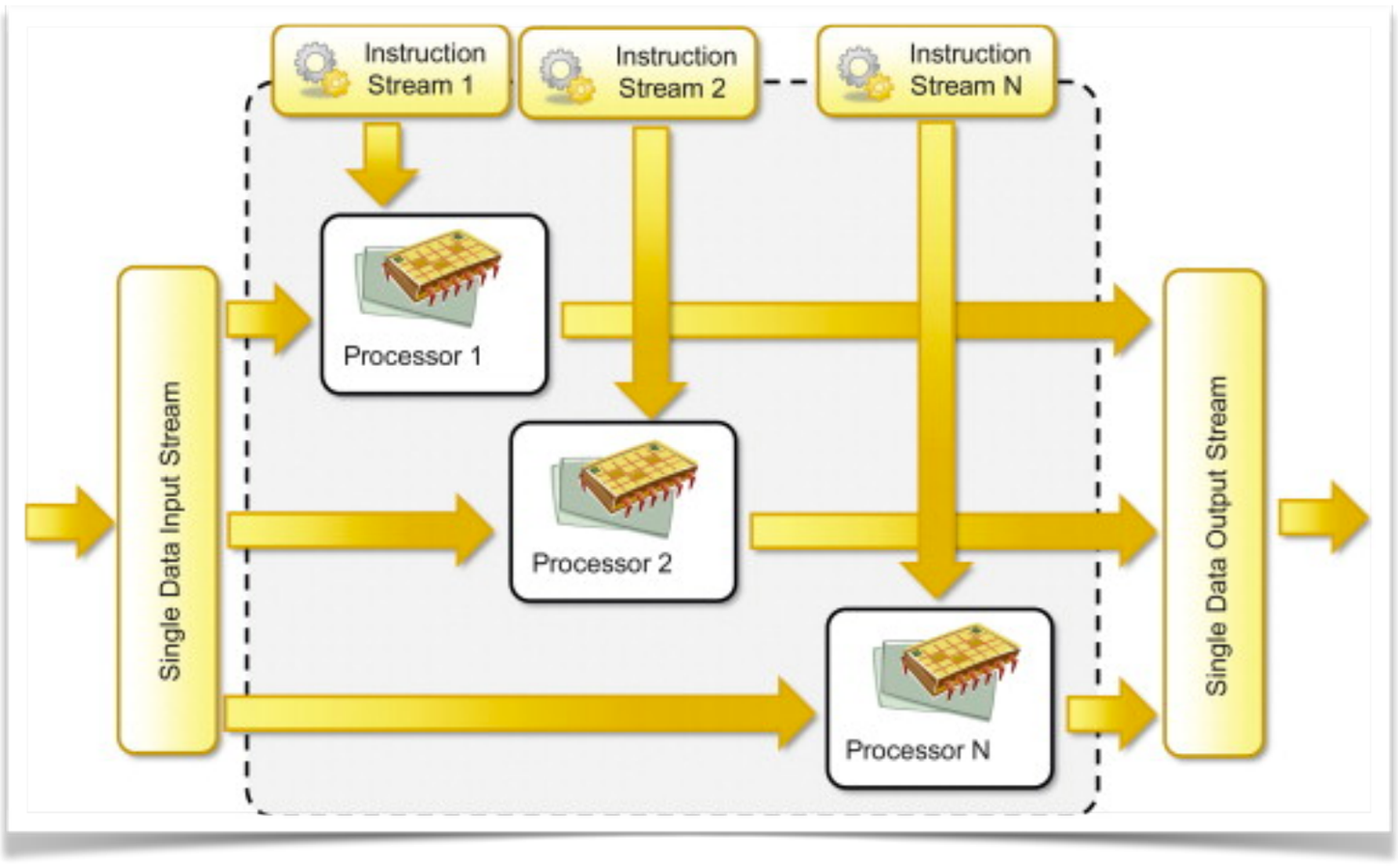
Single-instruction
Single-data
(SISD)

Железо



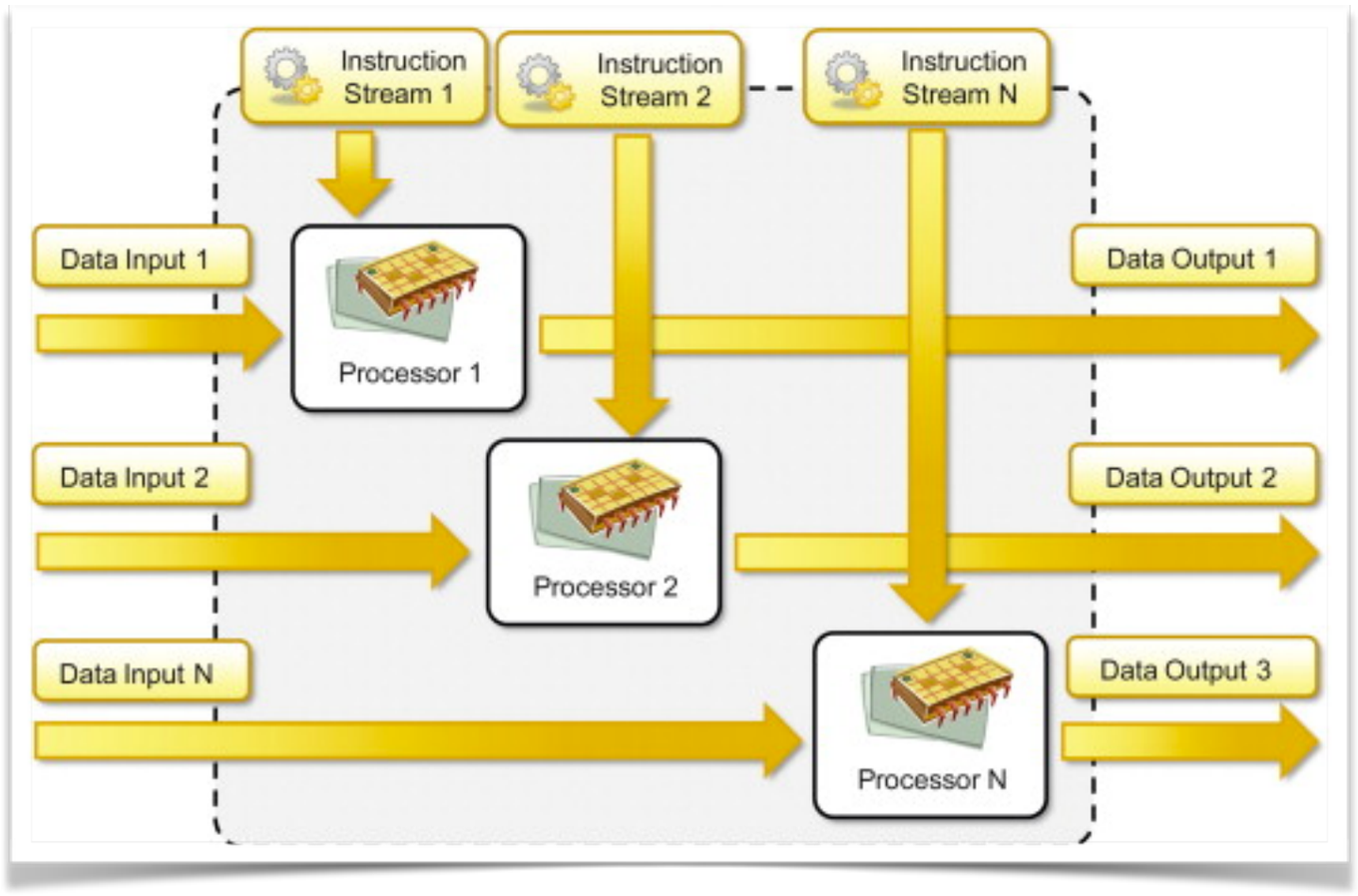
Single-instruction
Multiple-data
(SIMD)

Железо



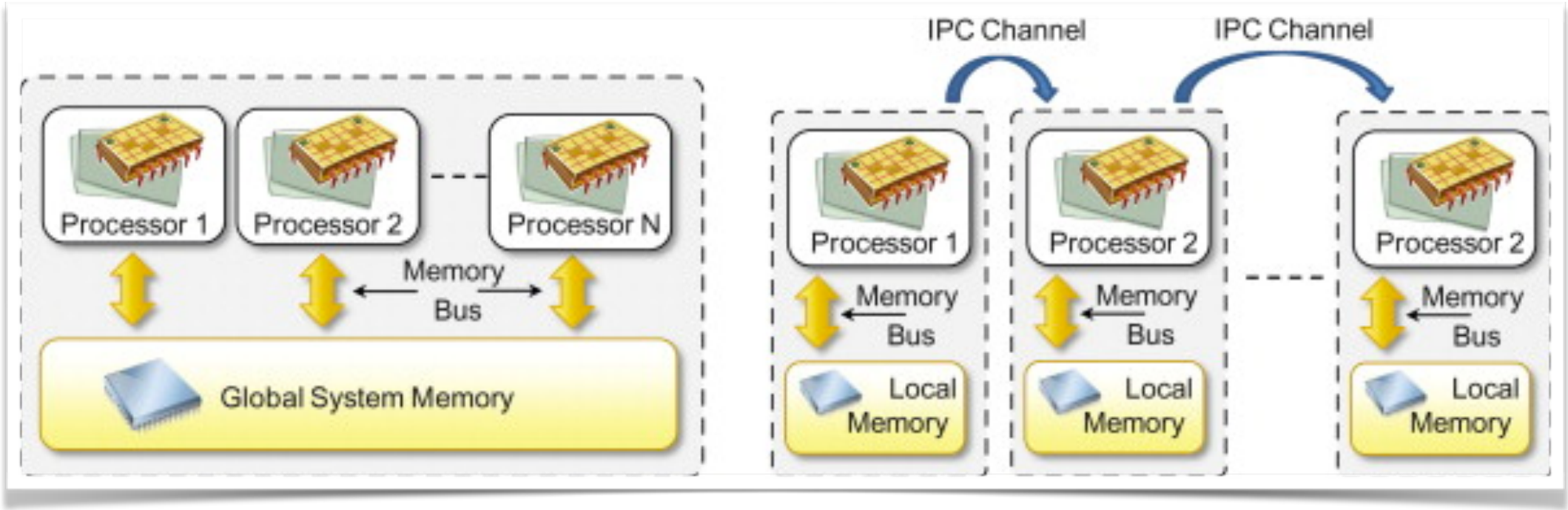
Multiple-instruction Single-data (MISD)

Железо



Multiple-instruction Multiple-data (MIMD)

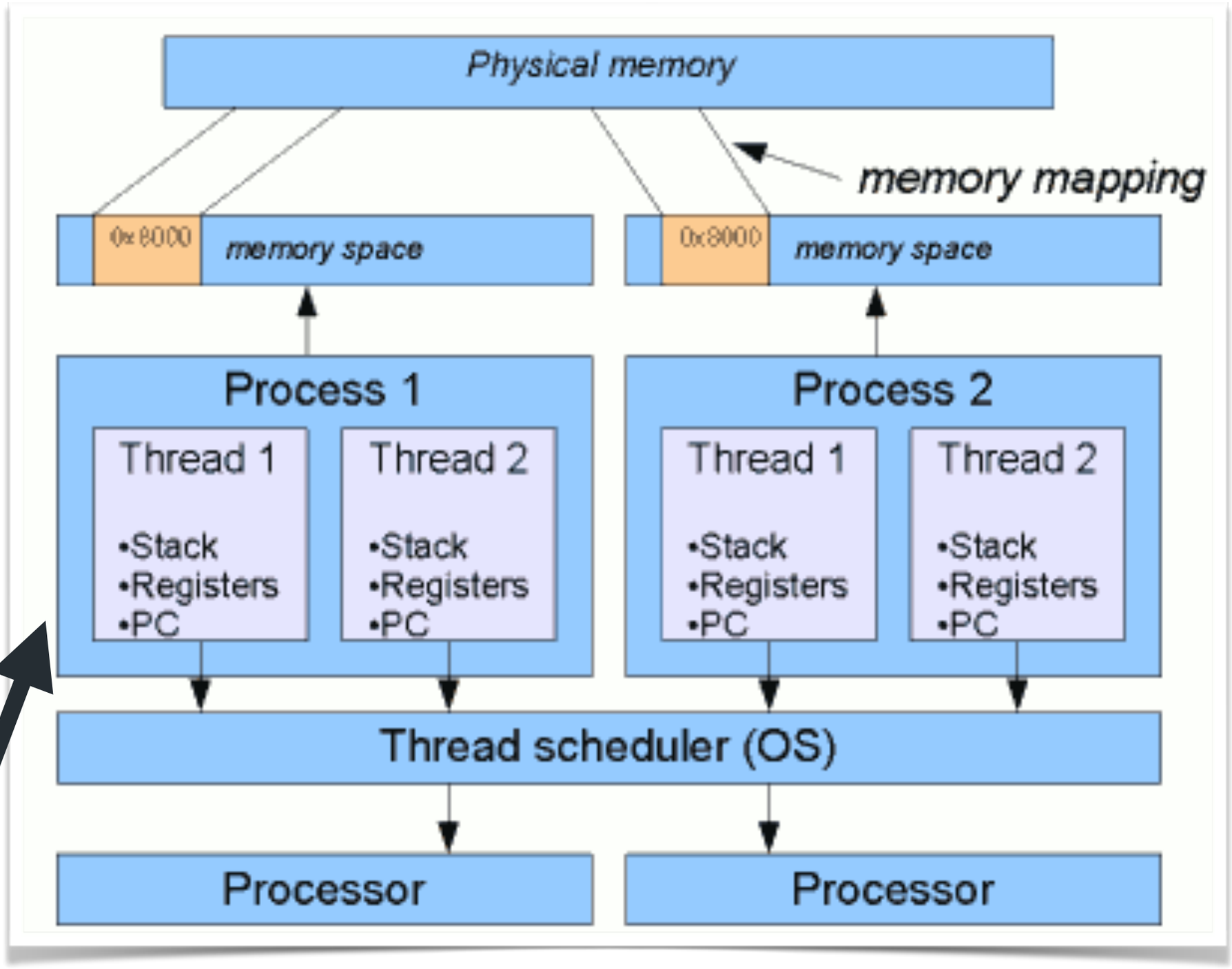
Железо



shared memory
MIMD

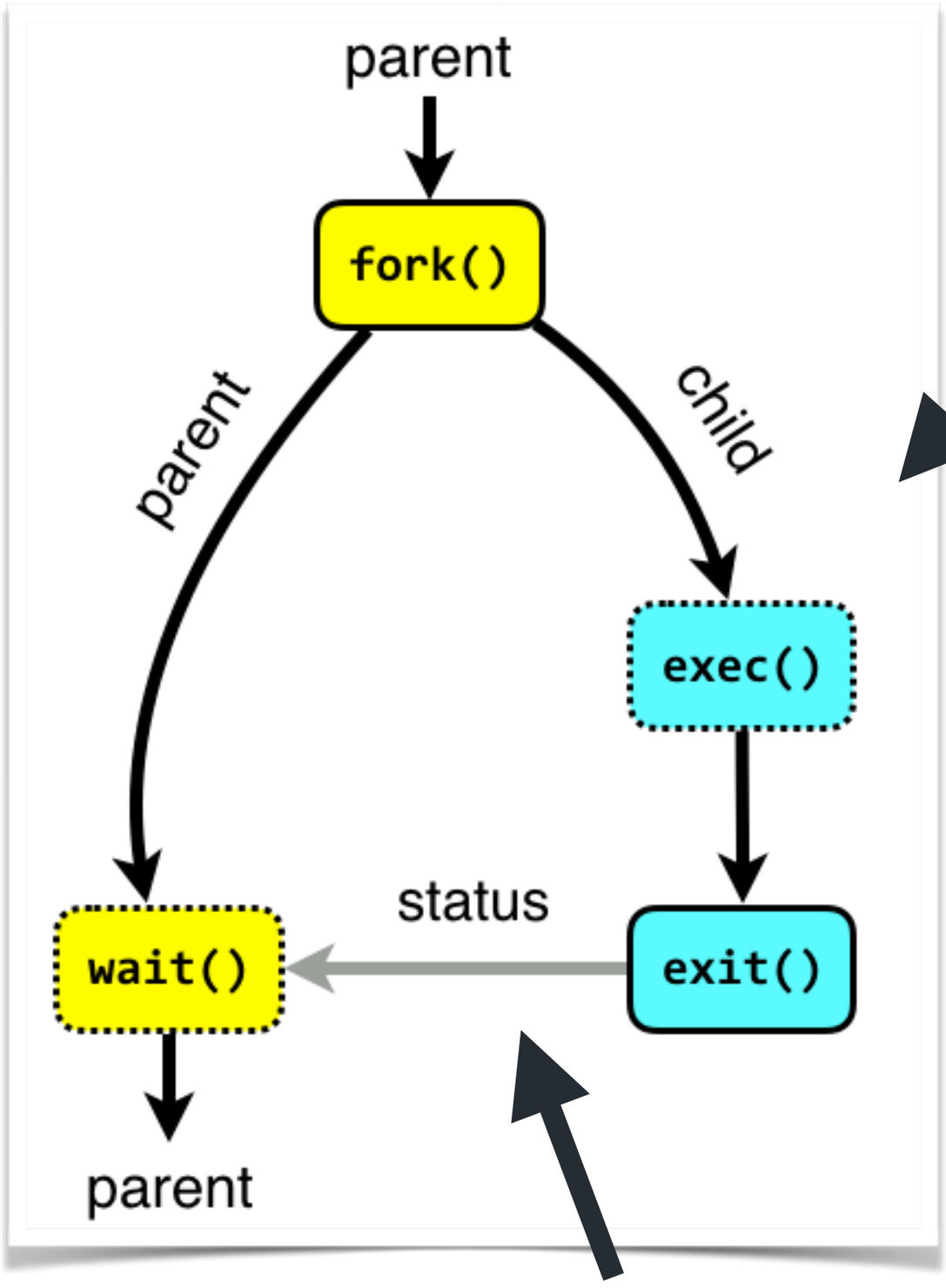
distributed memory
MIMD

Threads и processes



Можно сделать свои threads (кооперативные)

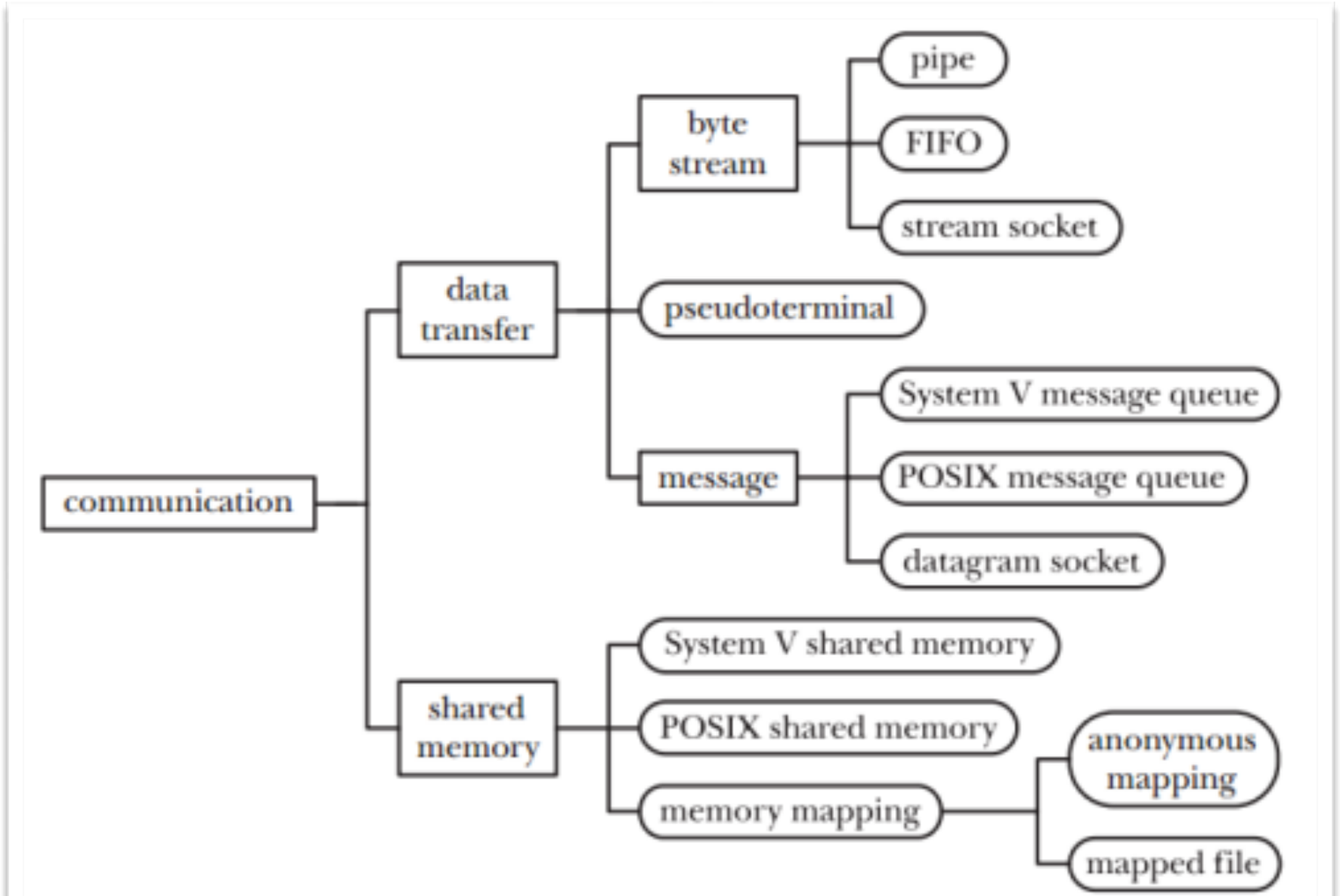
«жизнь» процесса



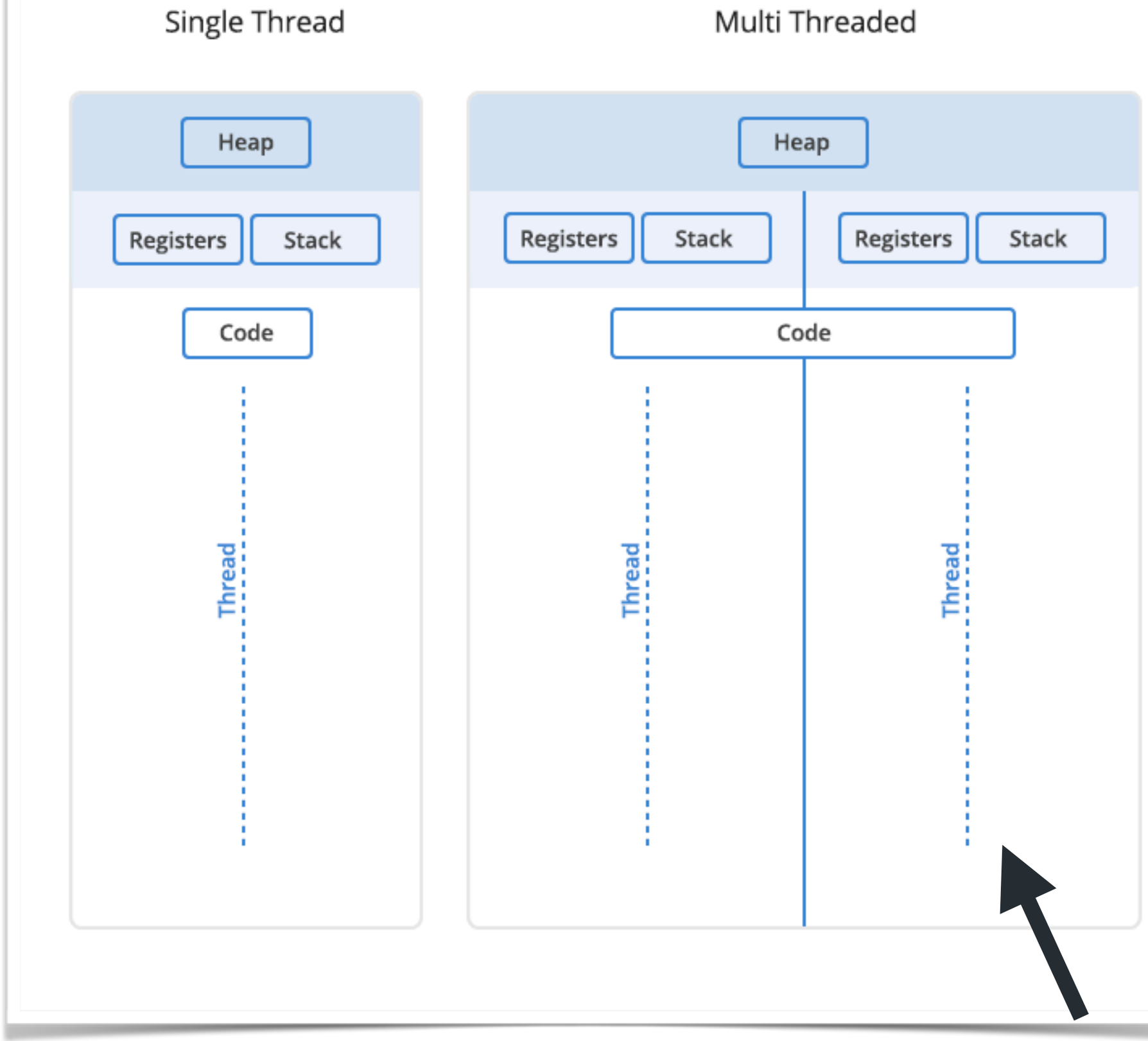
Иначе будет зомби 

```python
from multiprocessing import Process

def fun_in_child():
    print('Hello from child!')

p = Process(target=fun_in_child)
p.start()
p.join()
```

IPC в *nix


communication
data transfer
byte stream
pipe
FIFO
stream socket
pseudoterminal
message
System V message queue
POSIX message queue
datagram socket
shared memory
System V shared memory
POSIX shared memory
memory mapping
anonymous mapping
mapped file

```python
from multiprocessing import Process, Queue

def f(q):
    q.put([42, None, 'hello'])

if __name__ == '__main__':
    q = Queue()
    p = Process(target=f, args=(q,))
    p.start()
    print(q.get())    # prints "[42, None, 'hello']"
    p.join()
```

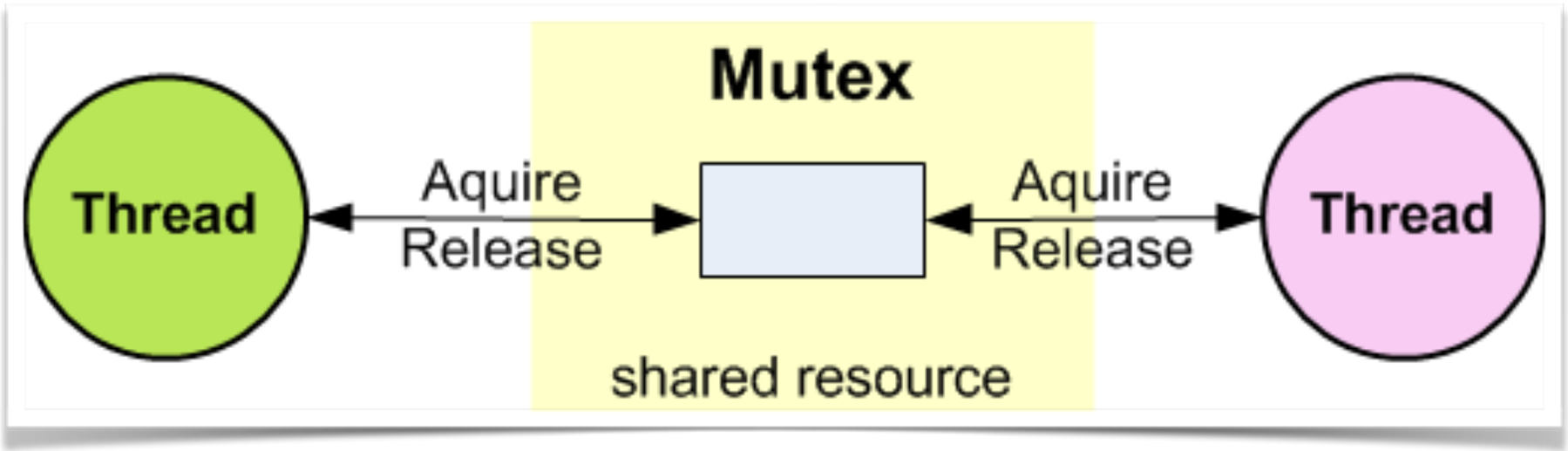


Можно сделать демона 😈

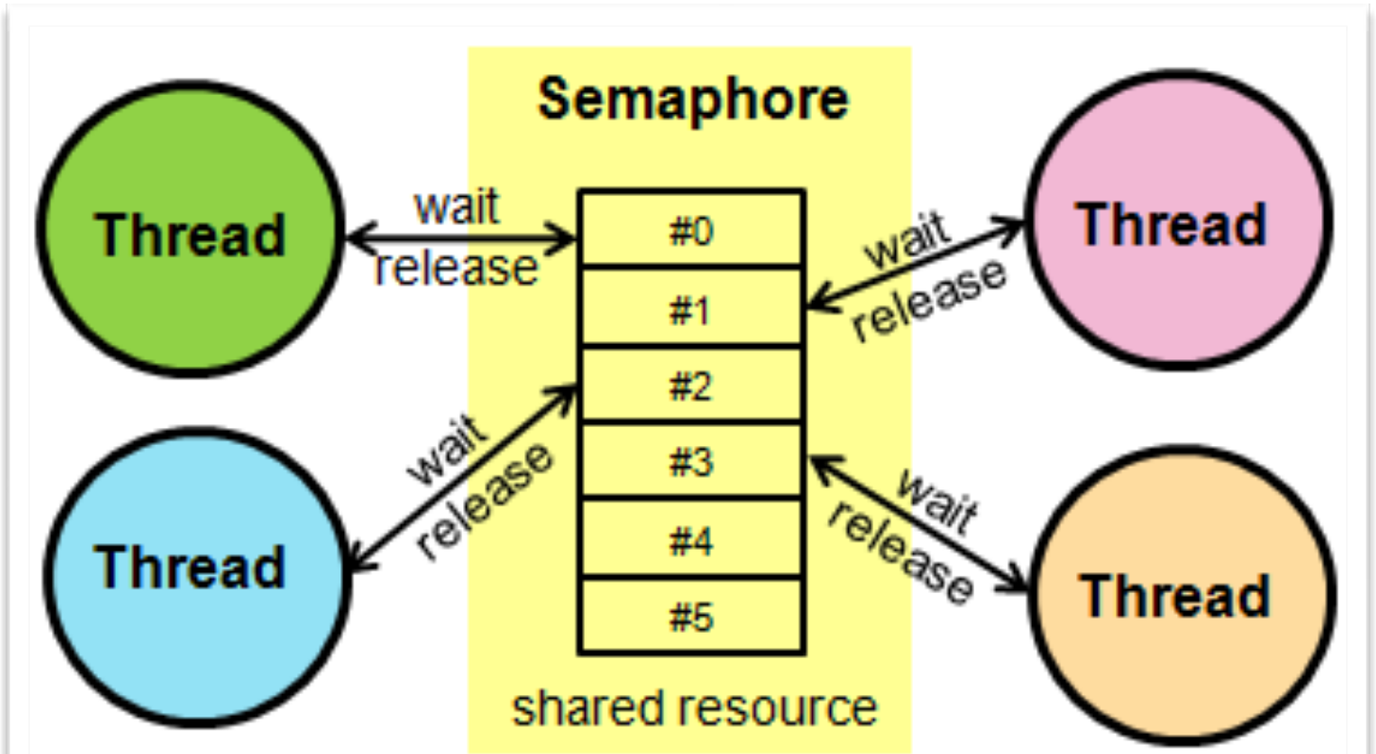
```python
from threading import Thread

def subproc(n: int):
    [print(i) for i in range(n)]

if __name__ == "__main__":
    thread1 = Thread(target=subproc, args=(5,))
    thread2 = Thread(target=subproc, args=(5,))

    thread1.start()
    thread2.start()
    thread1.join()
    thread2.join()
```

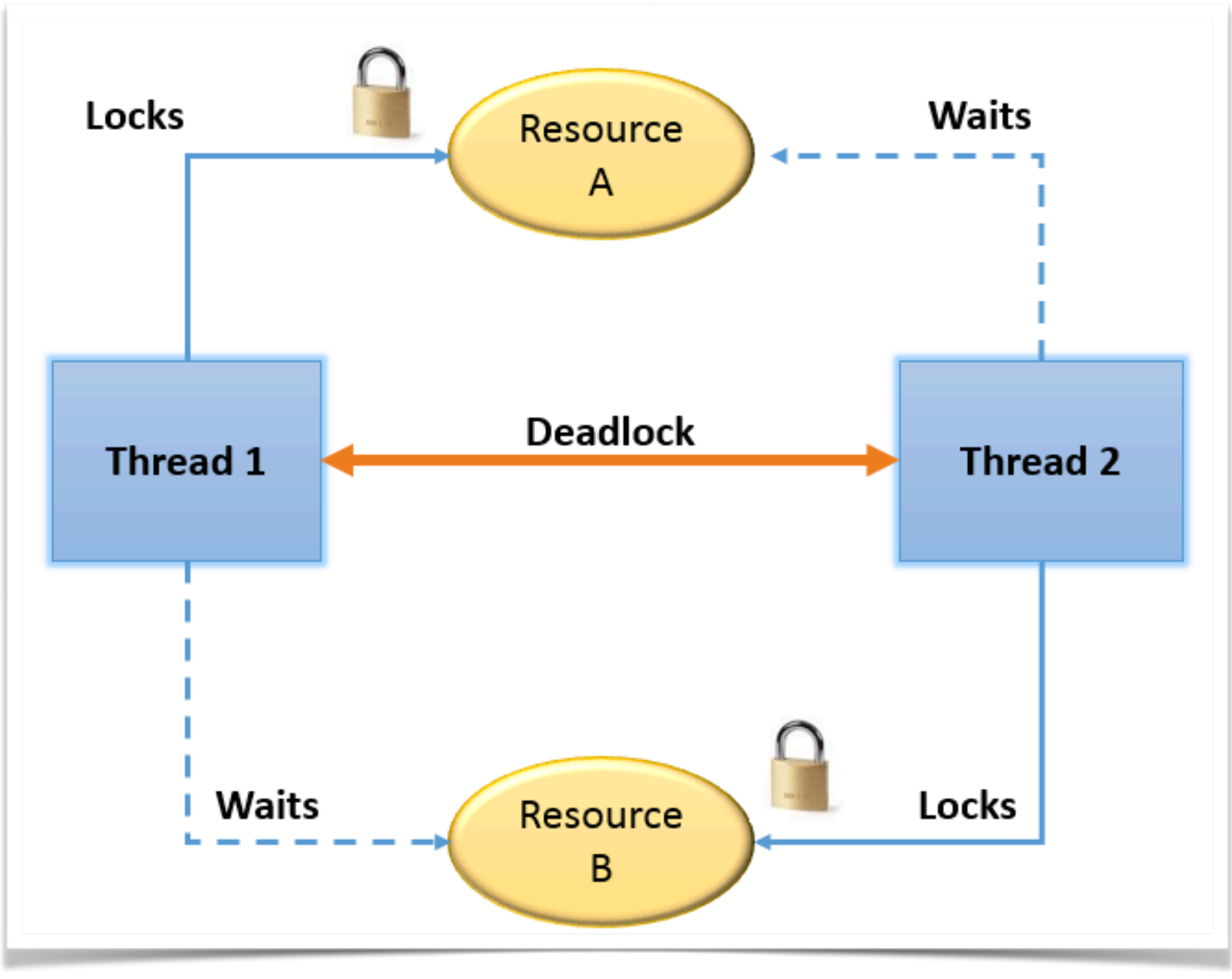
Синхронизация

Knock knock.
Race condition.
Who’s there?

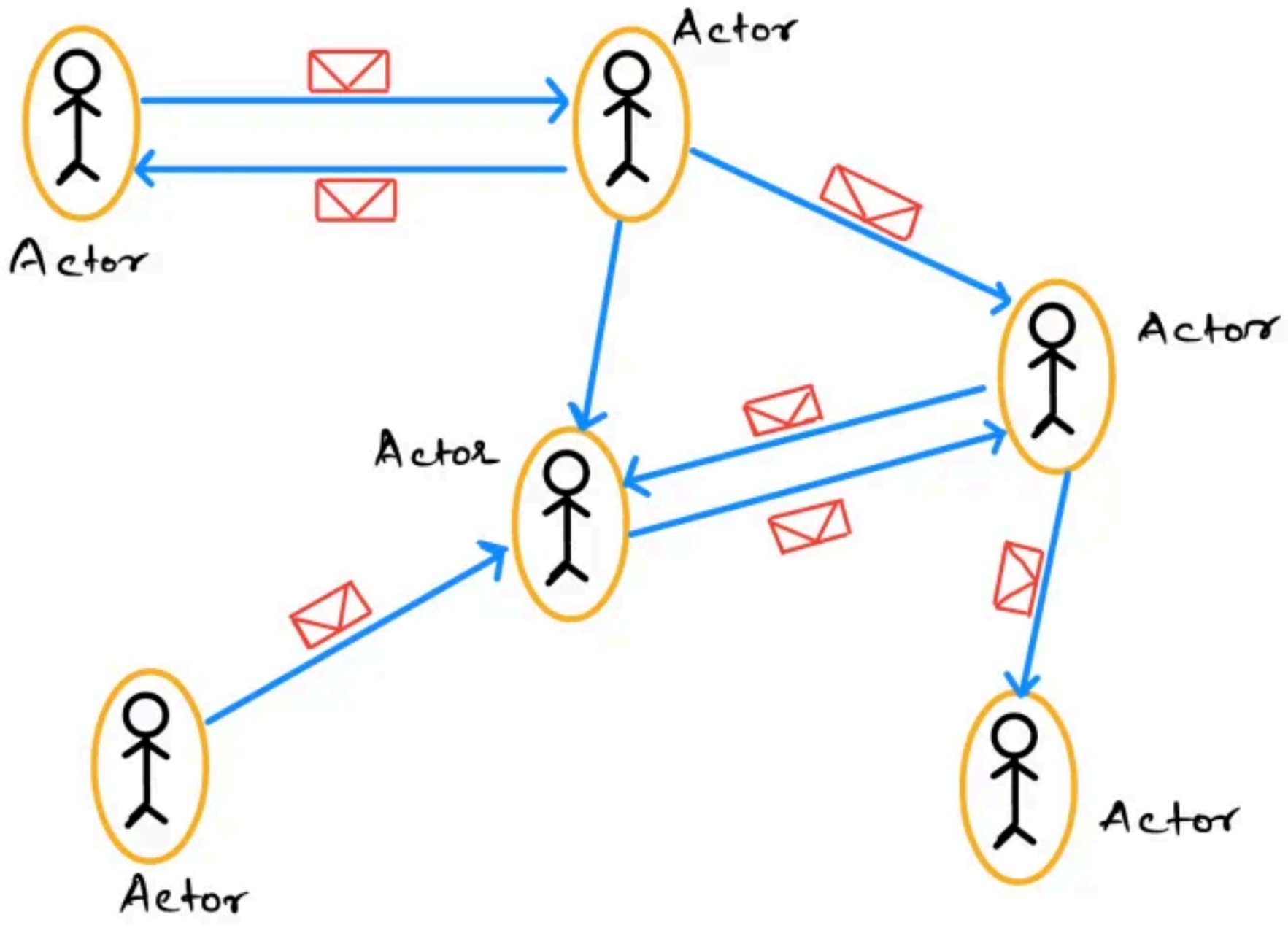

Mutex
Thread
Aquire
Release
Aquire
Release
Thread
shared resource

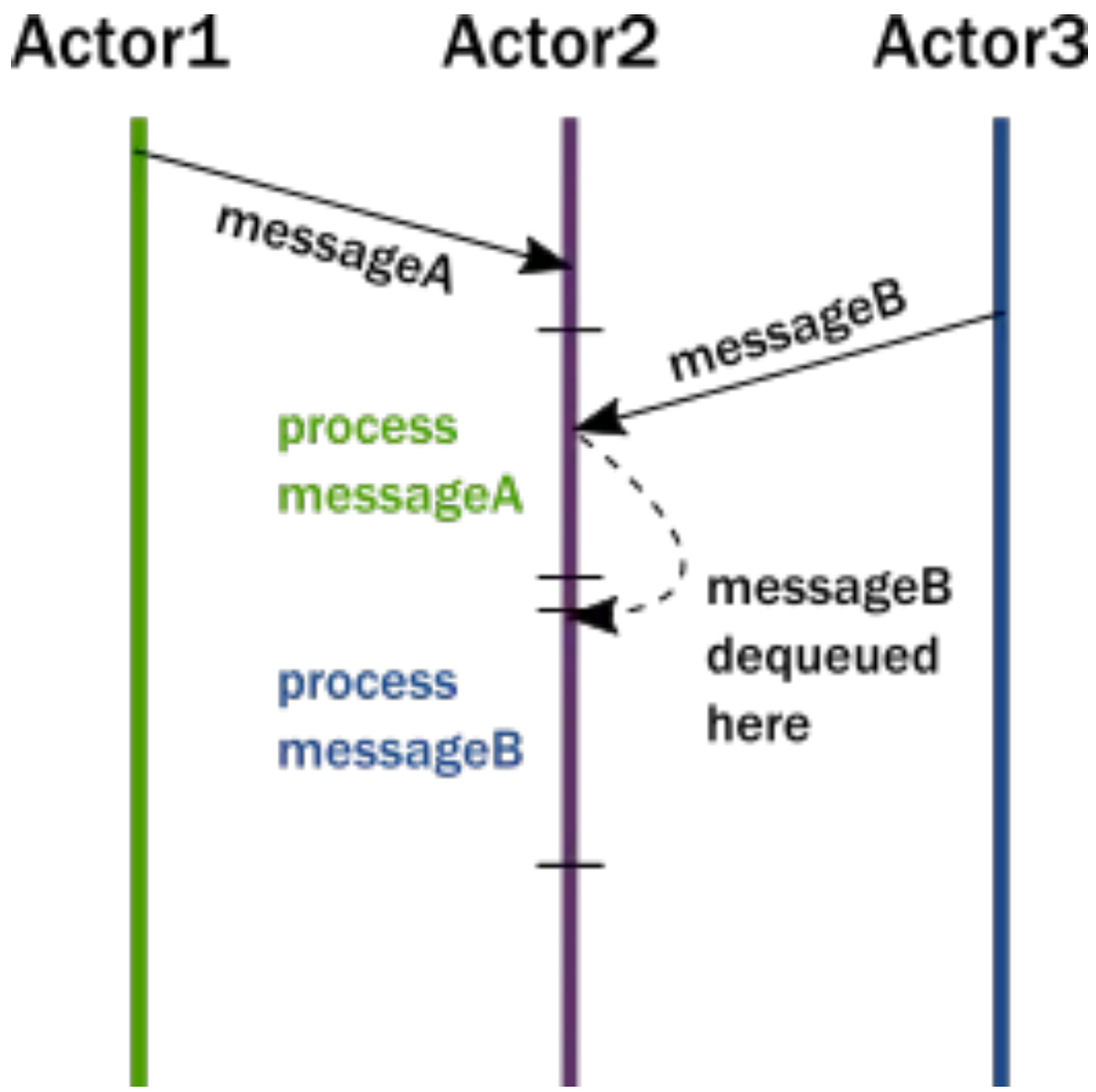
```python
from threading import Lock

mutex = Lock()

try:
    mutex.acquire()
    # Работа с общими ресурсами

except:
    # Обрабатываем исключения
    pass

finally:
    mutex.release()
```

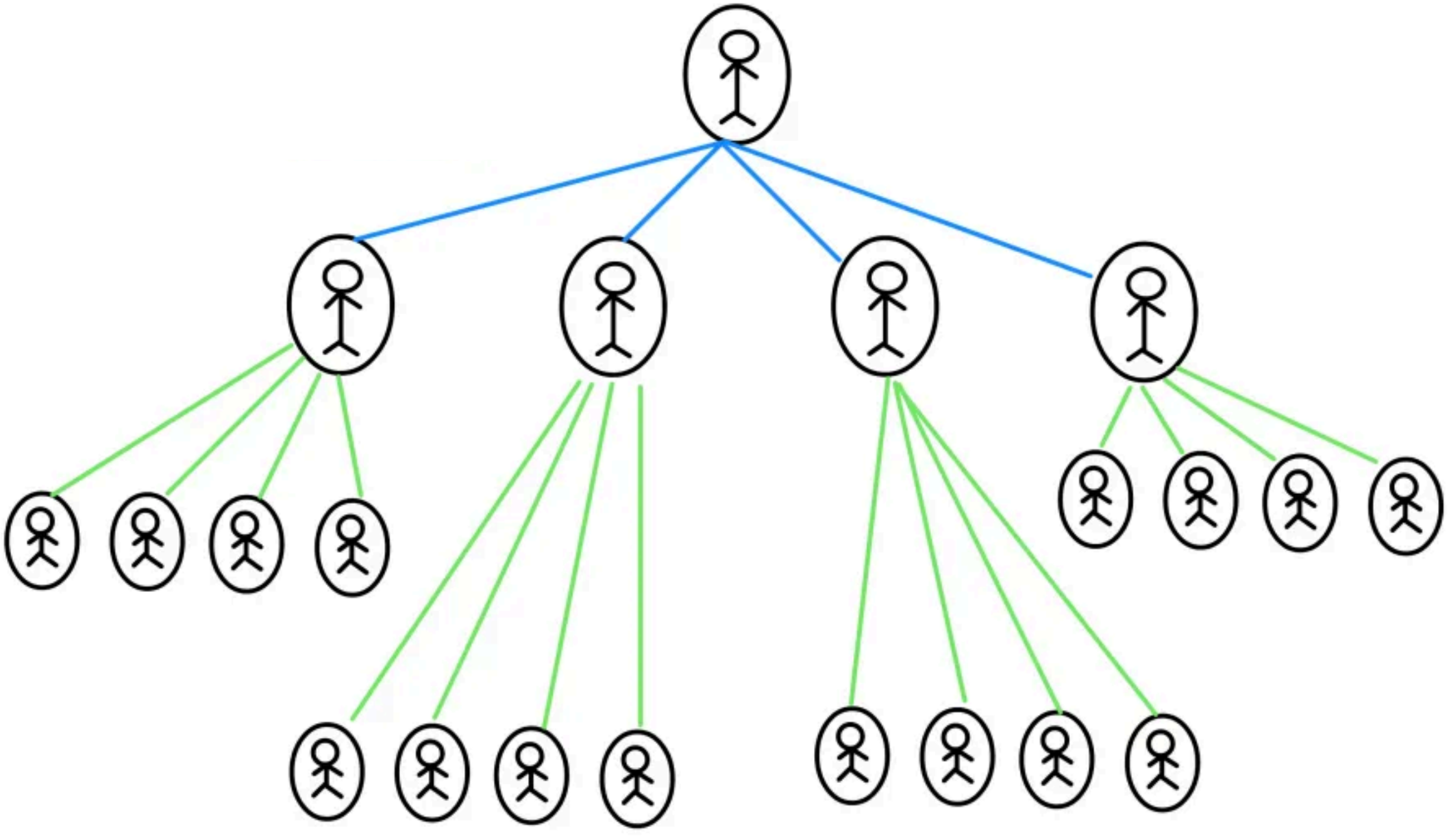

Semaphore
Thread
wait
release
#0
#1
#2
#3
#4
#5
Thread
wait
release
Thread
wait
release
Thread
wait
release
shared resource

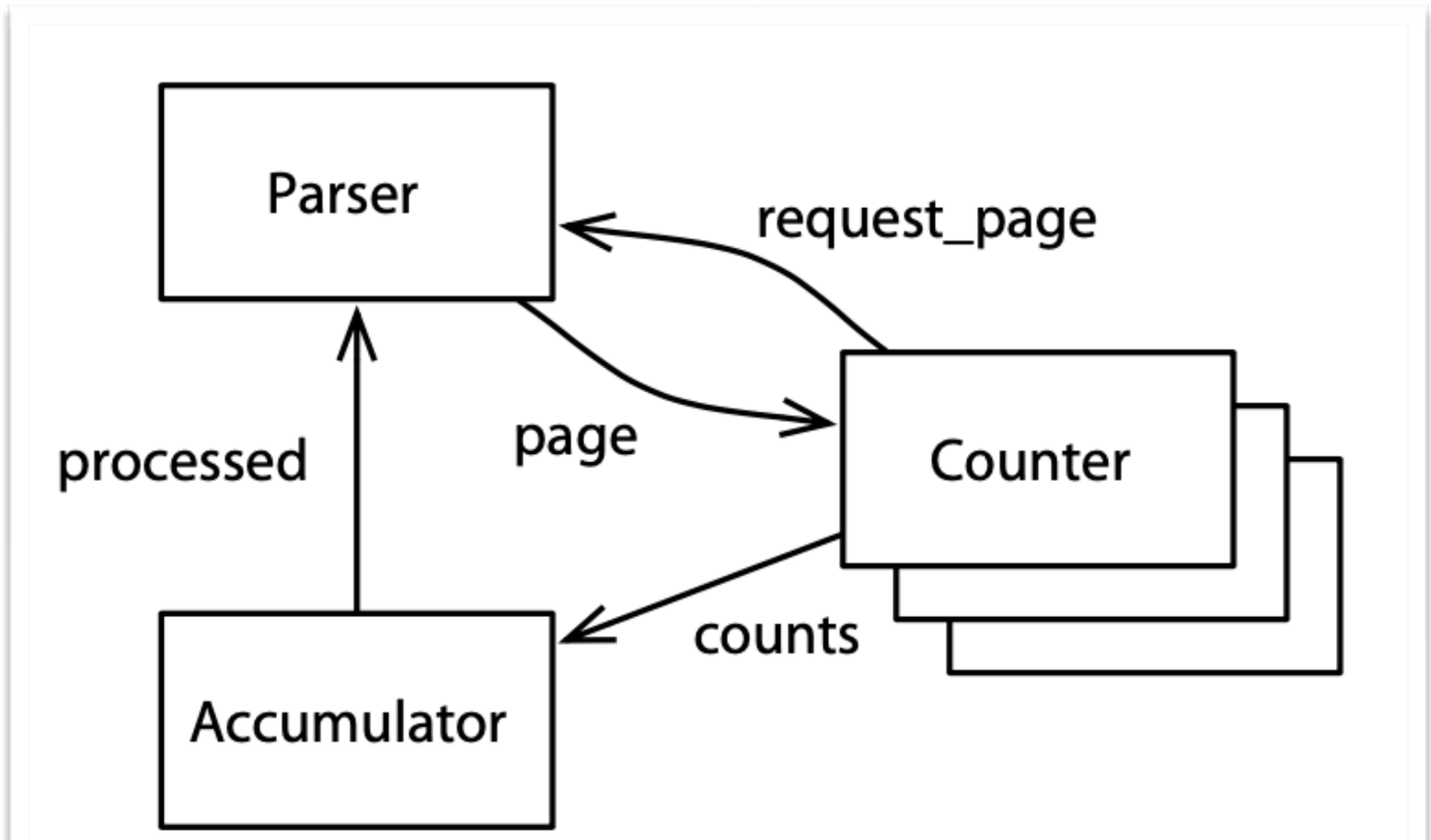

Locks
Resource
A
Waits
Thread 1
Deadlock
Thread 2
Waits
Resource
B
Locks

Actors

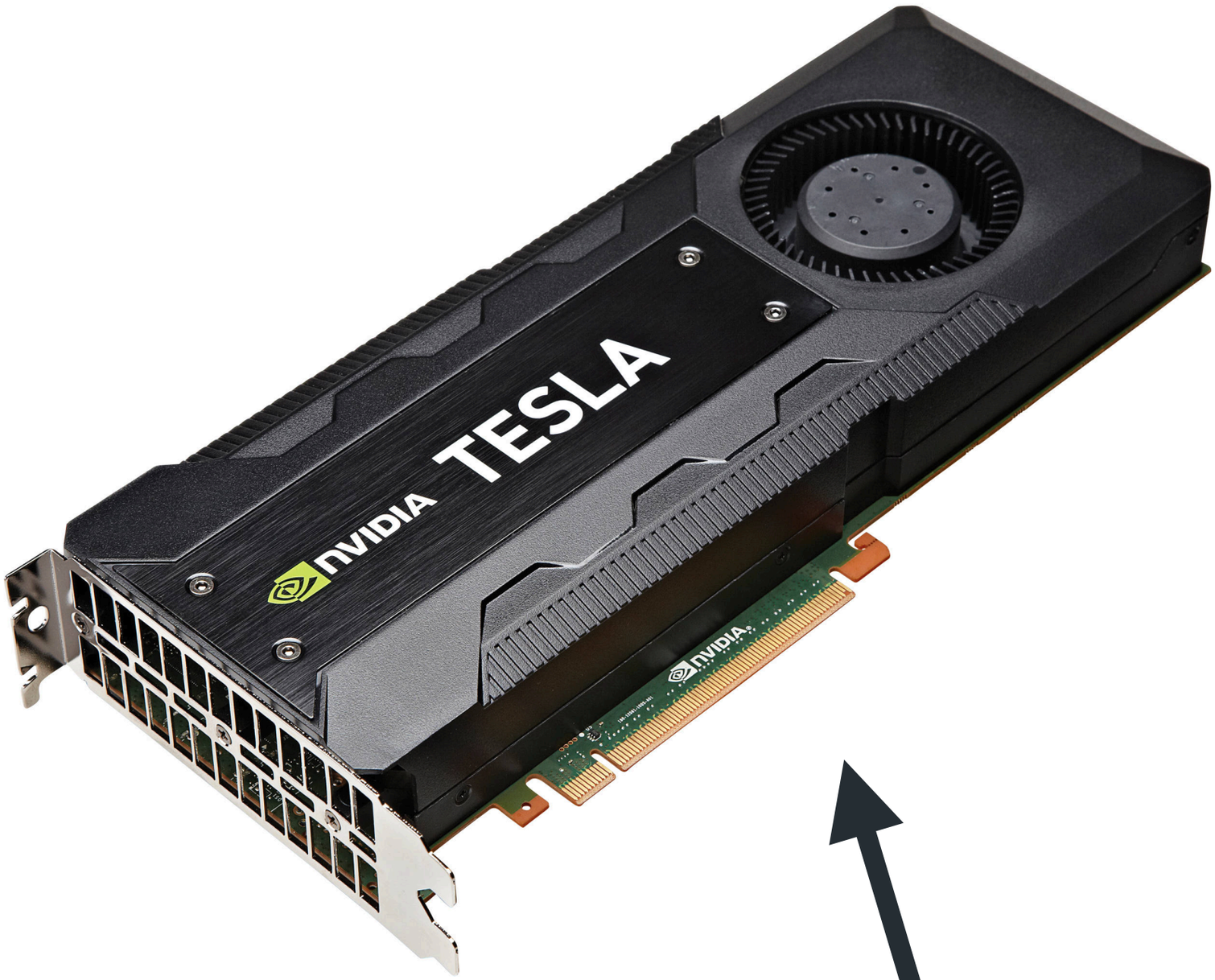

Actor
Actor
Actor
Actor
Actor
Actor
Actor


Actor1
Actor2
Actor3
messageA
messageB
process
messageA
messageB
dequeued
here
process
messageB

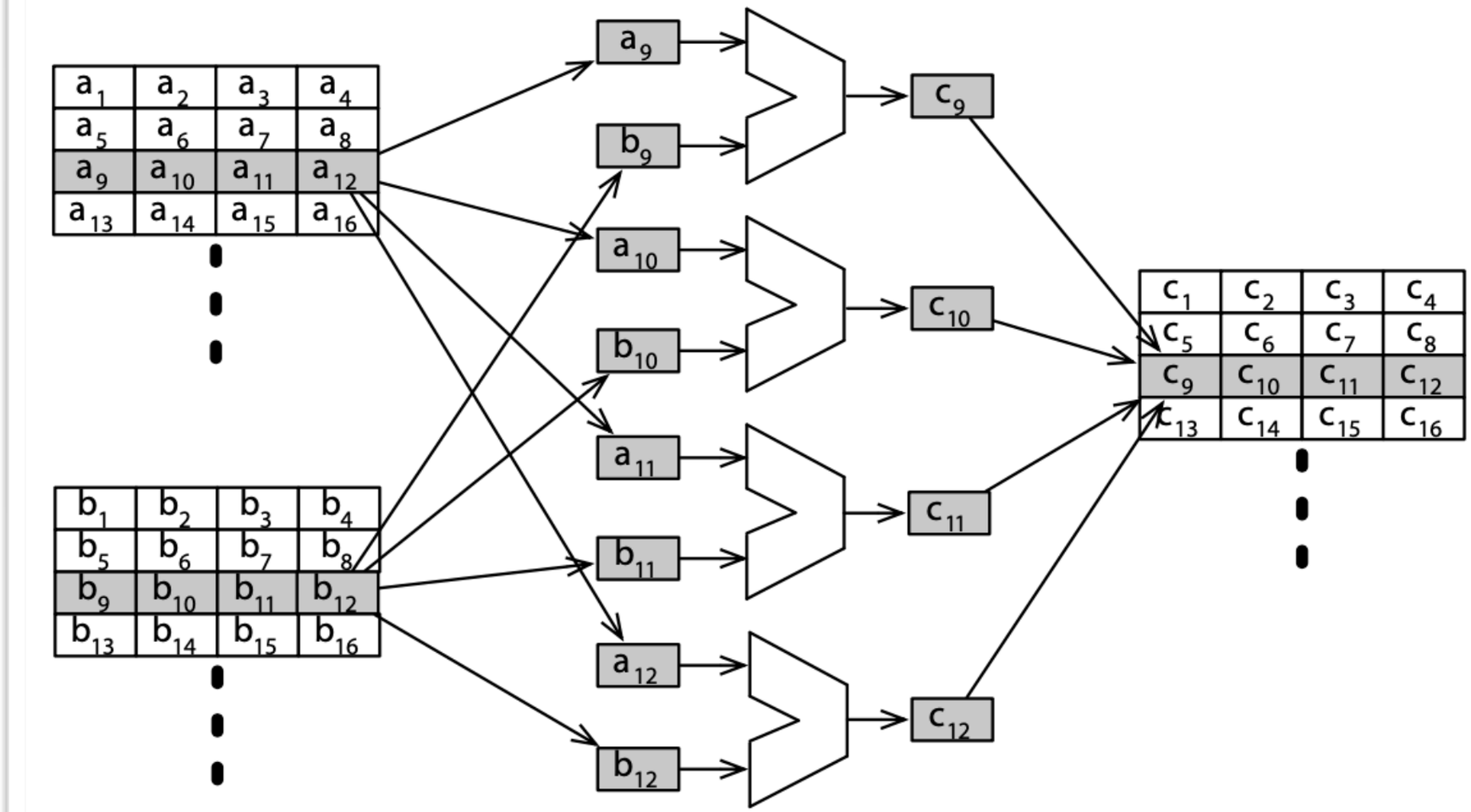



Parser
request_page
processed
page
Counter
counts
Accumulator

GPU: параллелизм данных



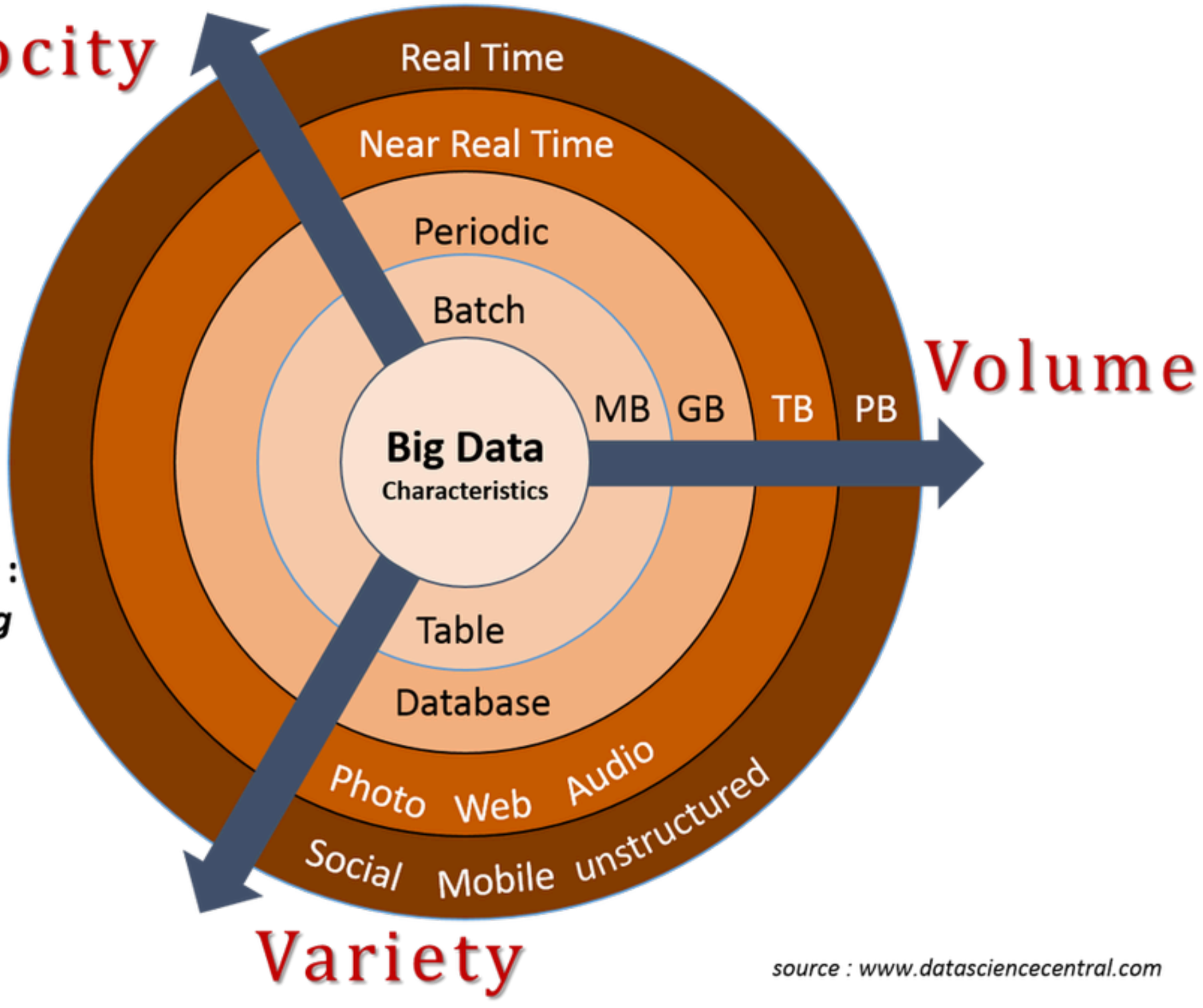
GPU


a1 a2 a3 a4
a5 a6 a7 a8
a9 a10 a11 a12
a13 a14 a15 a16
b1 b2 b3 b4
b5 b6 b7 b8
b9 b10 b11 b12
b13 b14 b15 b16
a9
b9
a10
b10
a11
b11
a12
b12
c9
c10
c11
c12
c1 c2 c3 c4
c5 c6 c7 c8
c9 c10 c11 c12
c13 c14 c15 c16

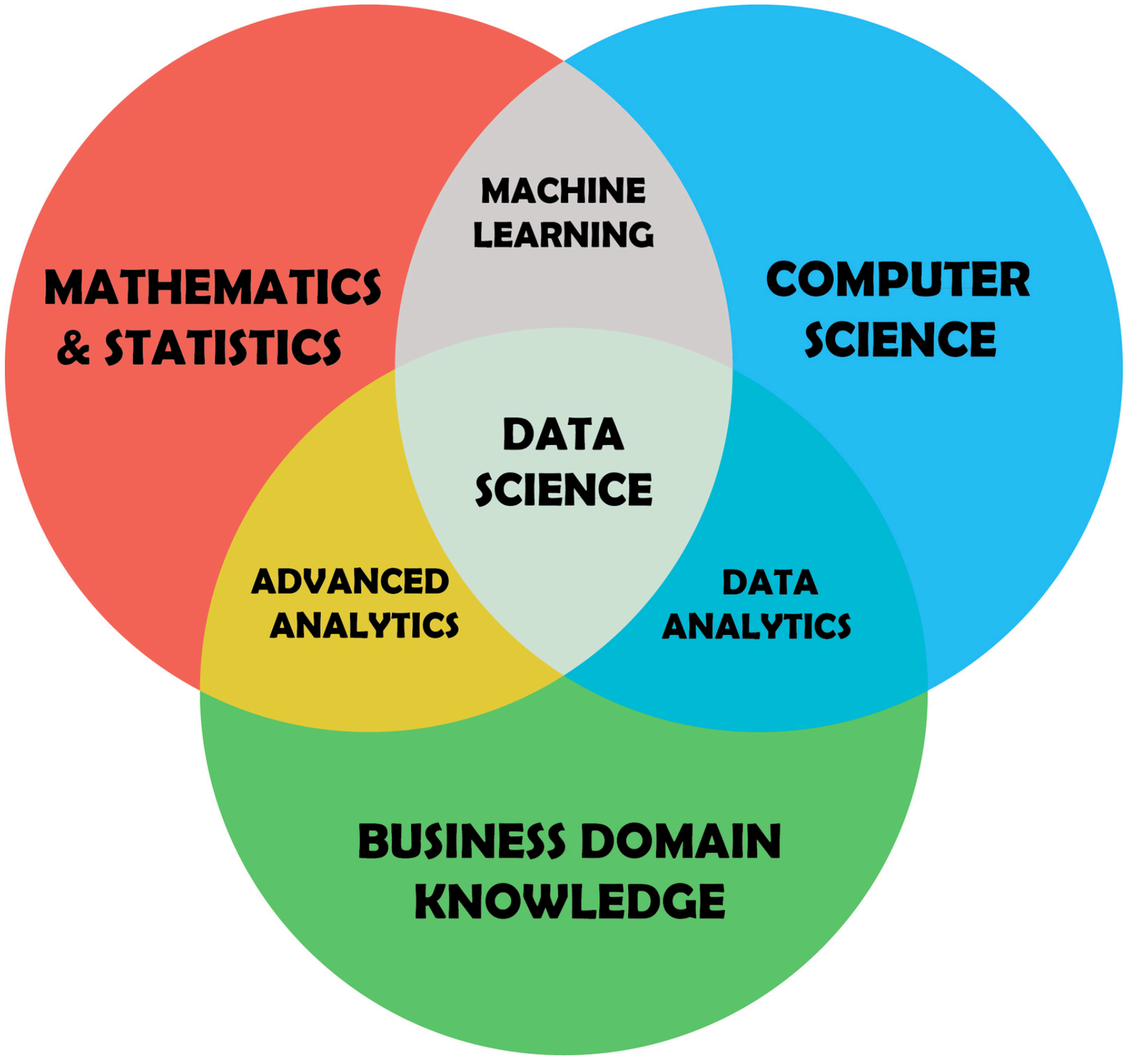
Язык OpenCL

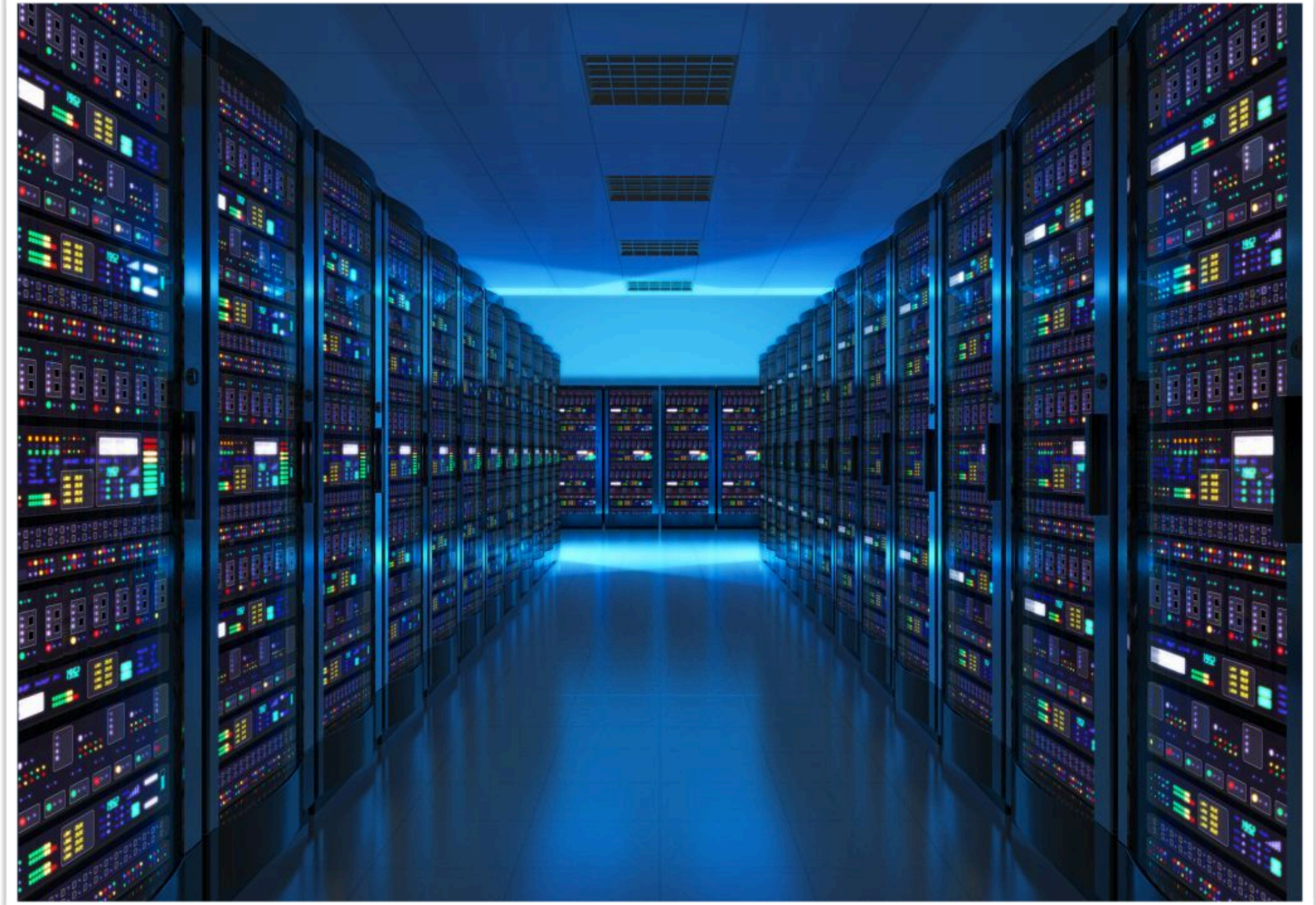
Лямбда-архитектура

Big Data Characteristics : ***Ever Expanding horizon***

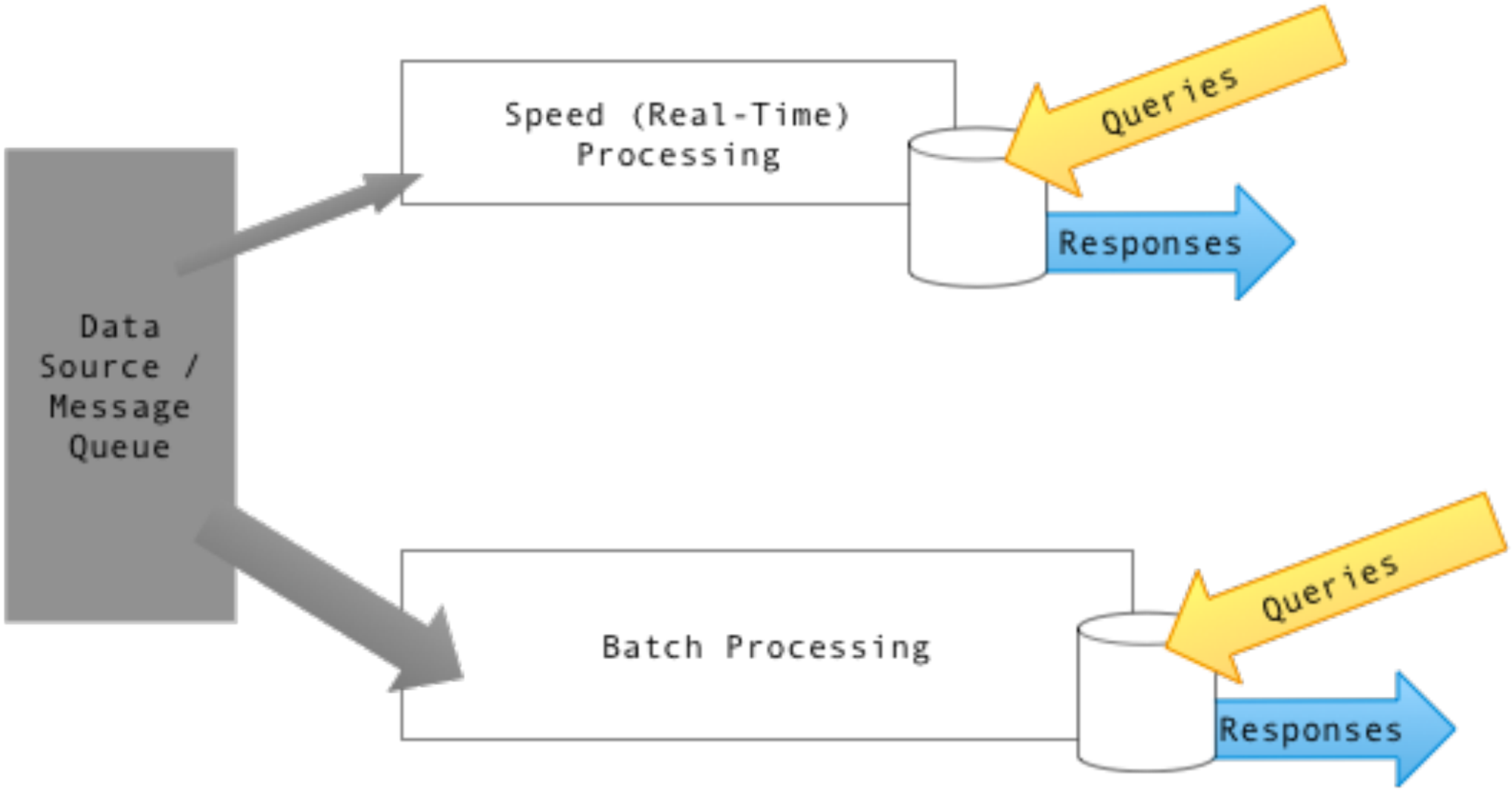


source : www.datasciencecentral.com


MATHEMATICS & STATISTICS
MACHINE LEARNING
COMPUTER SCIENCE
DATA SCIENCE
ADVANCED ANALYTICS
DATA ANALYTICS
BUSINESS DOMAIN KNOWLEDGE

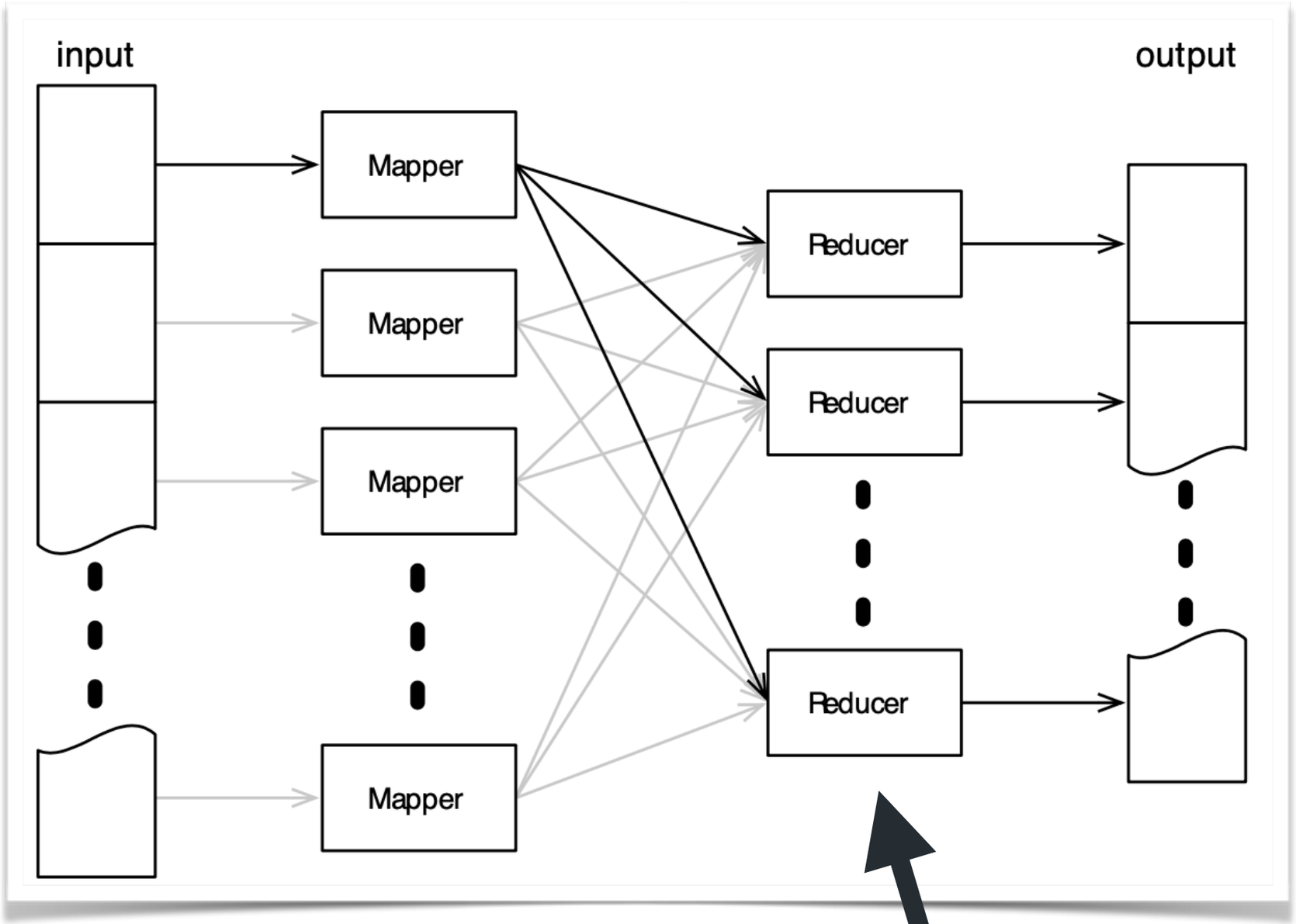


Кластер компьютеров



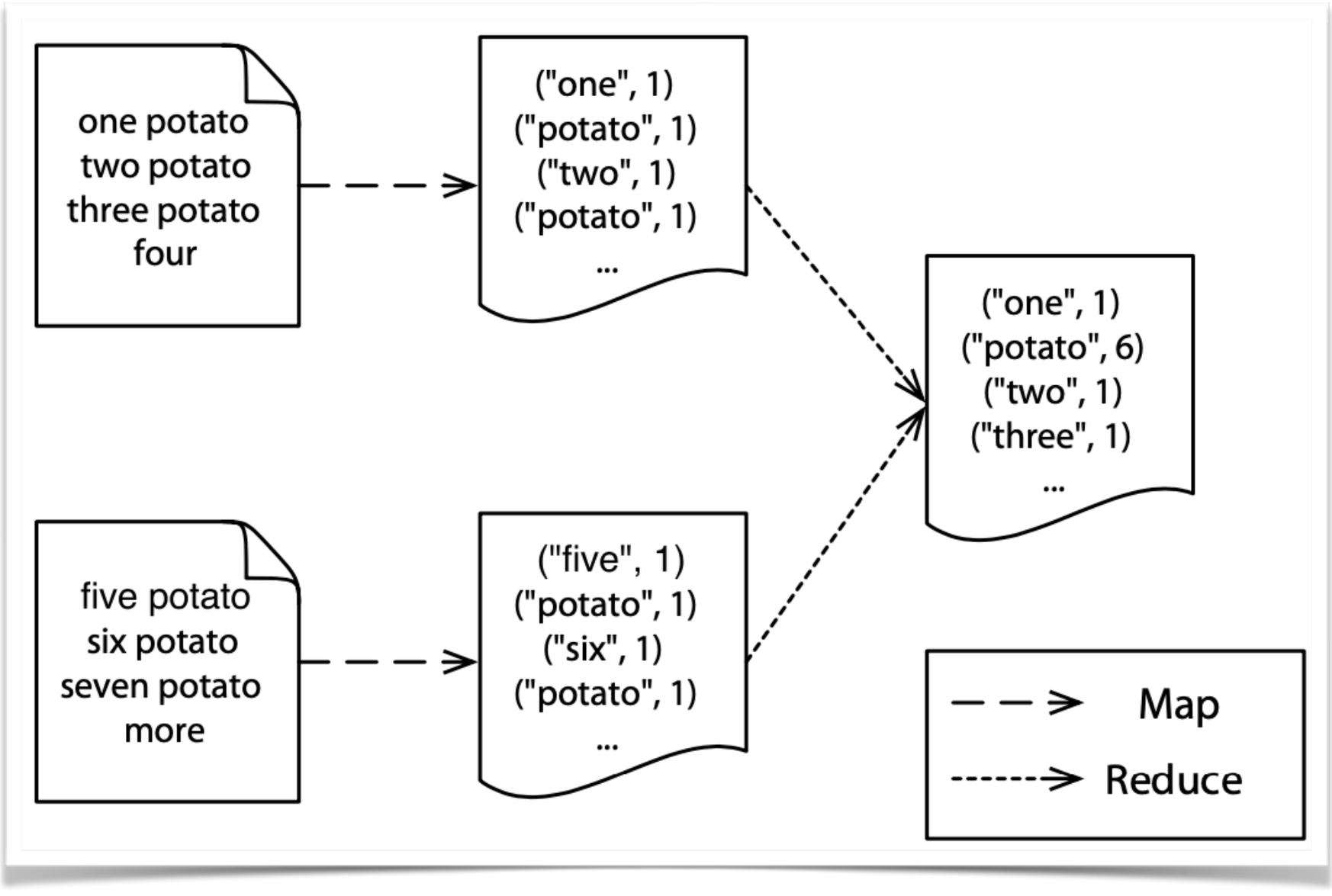
Лямбда-архитектура

Batch Processing

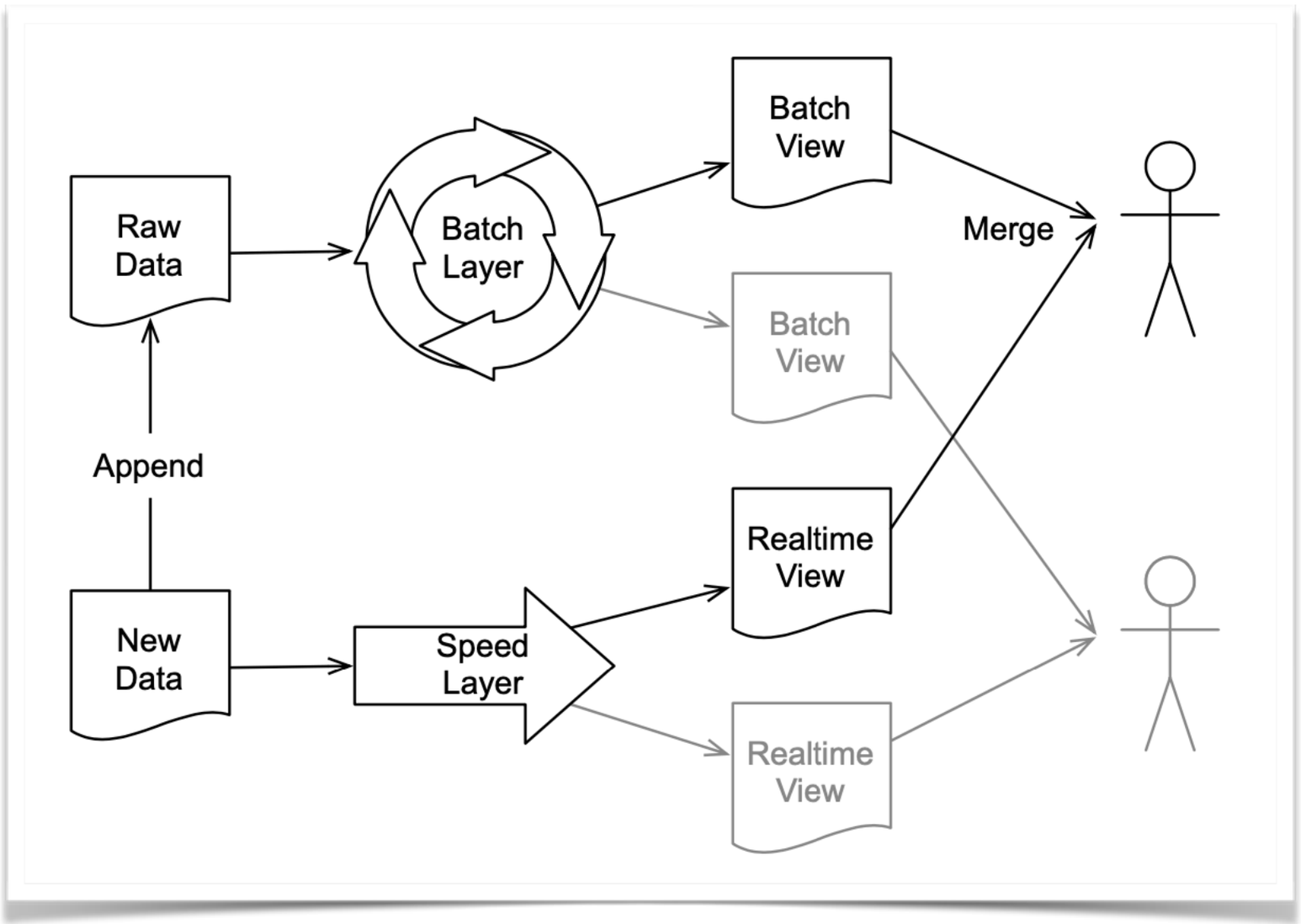


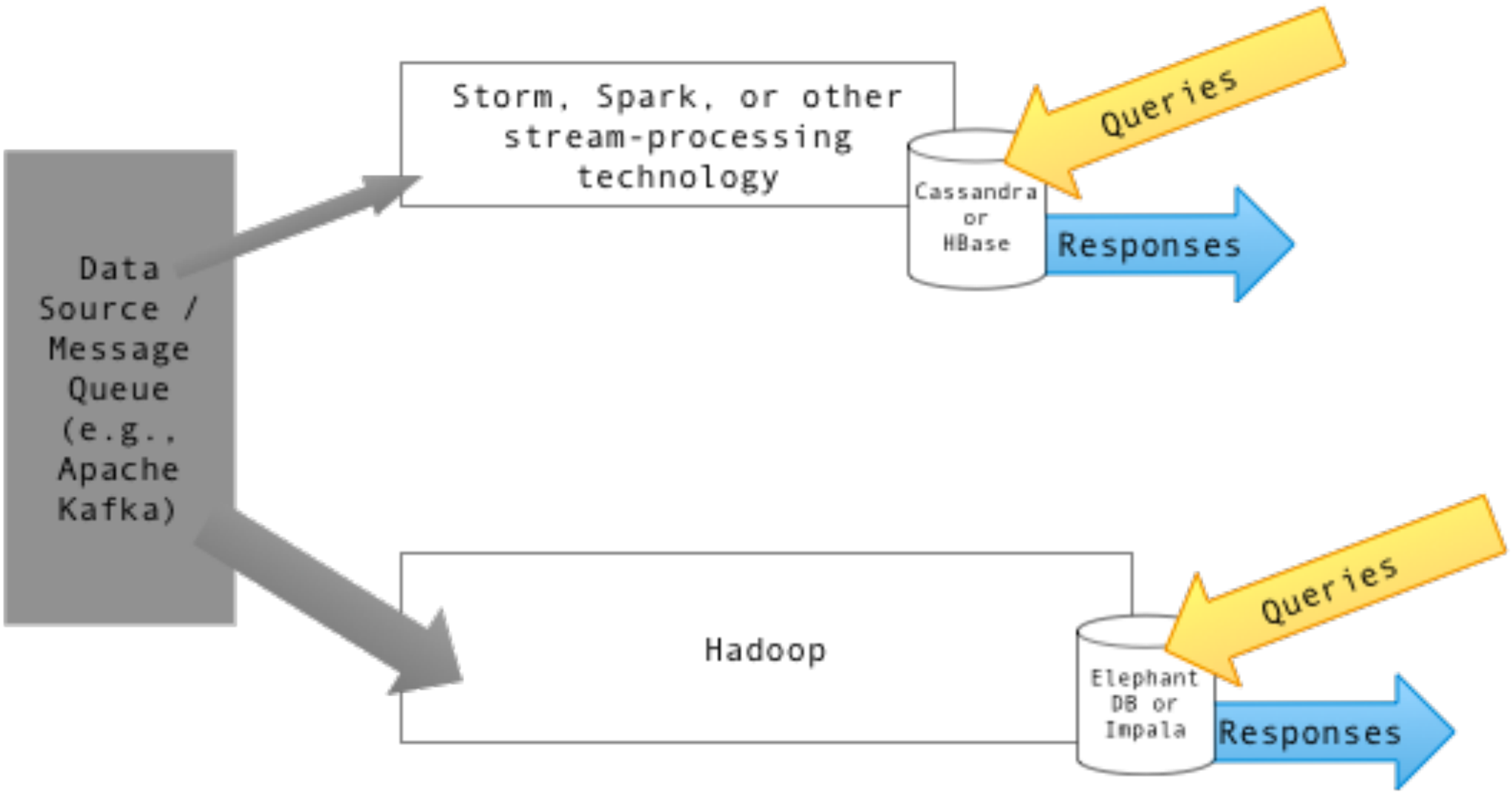
MapReduce

Batch Processing



Speed Processing




Storm, Spark, or other stream-processing technology
Data Source / Message Queue (e.g., Apache Kafka)
Cassandra or HBase
Queries
Responses
Hadoop
Elephant DB or Impala
Queries
Responses

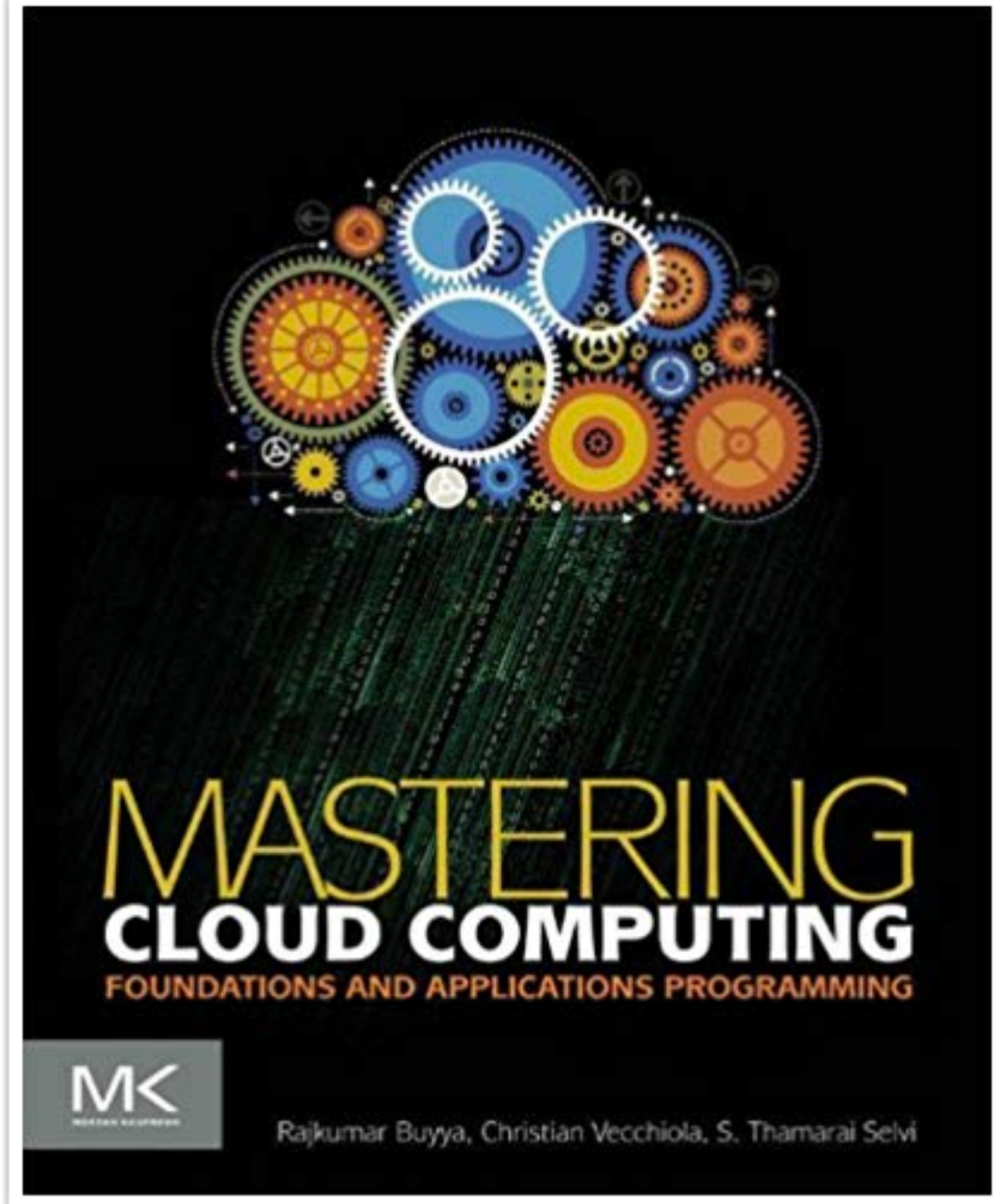
The Pragmatic Programmers

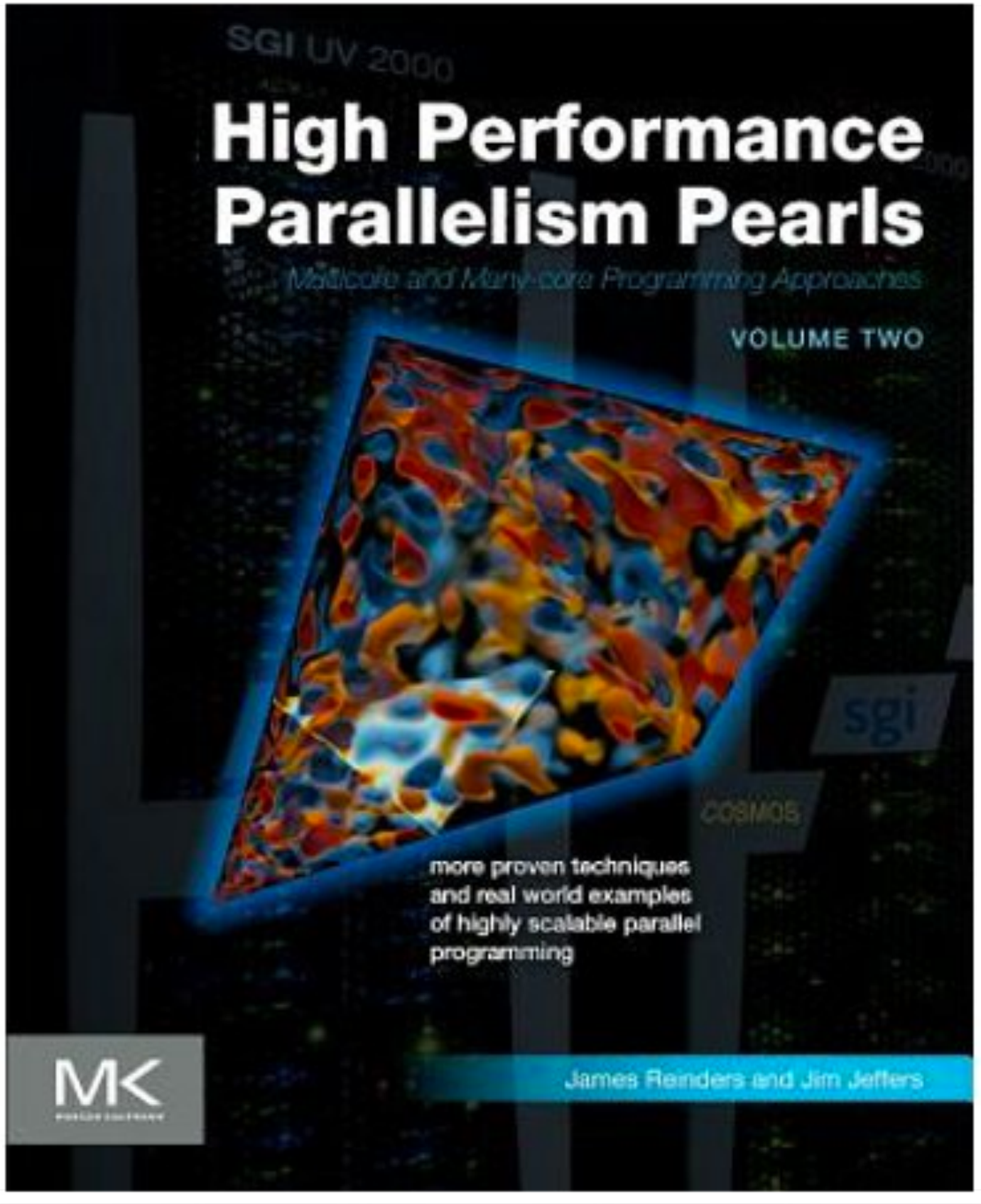
Seven Concurrency Models in Seven Weeks

When Threads Unravel

Paul Butcher

Series editor: *Bruce Tate*
Development editor: *Jacquelyn Carter*


MASTERING
CLOUD COMPUTING
FOUNDATIONS AND APPLICATIONS PROGRAMMING
MK
Rajkumar Buyya, Christian Vecchiola, S. Thamarai Selvi


SGI UV 2000
High Performance Parallelism Pearls
Multicore and Many-core Programming Approaches
VOLUME TWO
more proven techniques
and real world examples
of highly scalable parallel
programming
James Reinders and Jim Jeffers

Что дальше?