Introduction to Parallel Computing

Definition

Parallel algorithm - an algorithm that can be implemented in parts on a variety of different computing devices, followed by combining the results and obtaining the correct result

Parallel algorithms

Reasons for using parallel algorithms:

1. Reduction of time for solving problems
2. Providing the ability to solve larger tasks in a given time

The crisis of efficiency

Thanks to Moore's Law - increasing productivity of processors all the time led to a decrease in the time to perform some tasks.

The answer to the clock speed limit is multiprocessor systems.

However:

Serial programs cannot run faster on multiprocessor systems than on serial ones

You cannot **automatically convert** serial programs to parallel programs

In this regard, it is necessary to develop new algorithms using the advantages of modern hardware.

Motivation

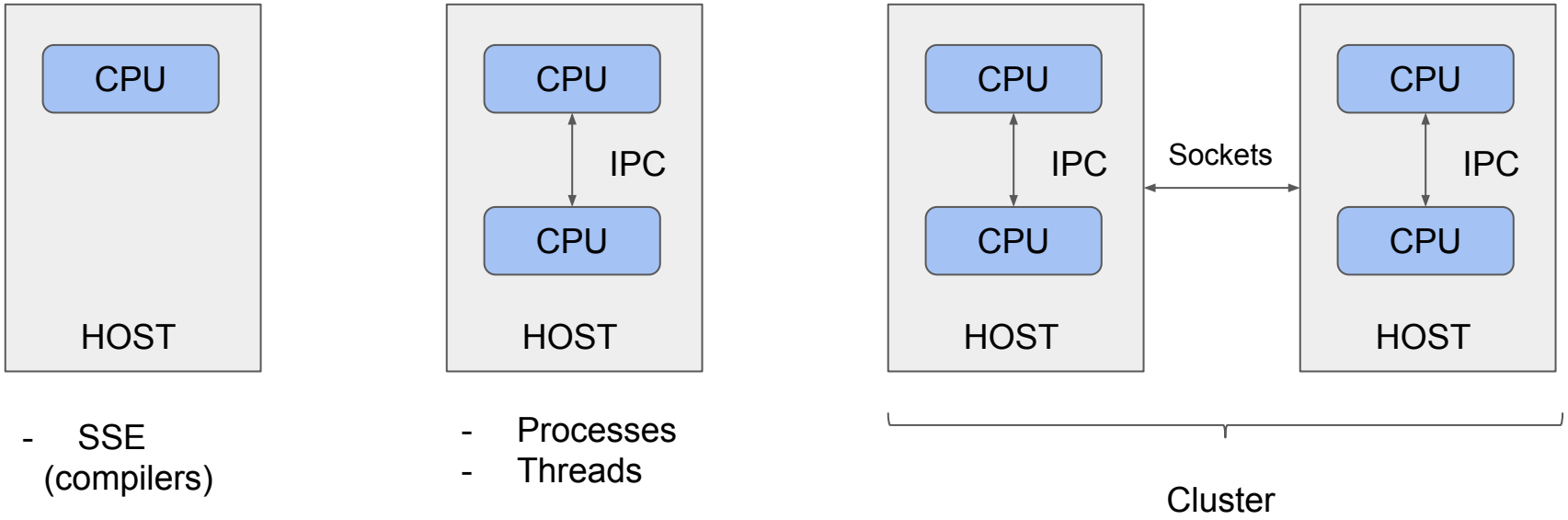
- The limit clock frequency

- High energy physics
- Climatology
- Genetic engineering
- Bank transactions

- Big Data
- Machine Learning (AI)

Facebook processes about 1 petabyte of data per day

Parallel computation



- SSE (compilers)

- Processes
- Threads

SSE. Streaming SIMD extension

```
vec_res.x = v1.x + v2.x;
vec_res.y = v1.y + v2.y;
vec_res.z = v1.z + v2.z;
vec_res.w = v1.w + v2.w;
```

```
movaps xmm0, [v1]
addps xmm0, [v2]
movaps [vec_res], xmm0
```

```
__declspec(align(16)) float a[4] = { 300.0, 4.0, 4.0, 12.0 };
__declspec(align(16)) float b[4] = {   1.5, 2.5, 3.5,  4.5 };

__asm {
    movups xmm0, a
    movups xmm1, b
    mulps xmm0, xmm1




    movups a, xmm0
};
```

SSE. Streaming SIMD extension

cat /proc/cpuinfo

```
processor       : 1
vendor_id       : GenuineIntel
cpu family      : 6
model           : 63
model name      : Intel(R) Xeon(R) CPU E5-2640 v3 @ 2.60GHz
stepping        : 2
microcode       : 0xffffffff
cpu MHz         : 2593.989
cache size      : 20480 KB
physical id     : 0
siblings        : 2
core id         : 1
cpu cores       : 2
apicid          : 1
initial apicid  : 1
fpu             : yes
fpu_exception   : yes
cpuid level     : 13
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflu
sh mmx fxsr sse sse2 ss ht syscall nx lm constant_tsc rep_good nopl eagerfpu pni pclmulqdq ssse
3 fma cx16 sse4_1 sse4_2 movbe popcnt aes xsave avx f16c rdrand hypervisor lahf_lm abm fsgsbase
 bmi1 avx2 smep bmi2 erms xsaveopt
bugs            :
bogomips        : 5187.97
clflush size    : 64
cache_alignment : 64
address sizes   : 42 bits physical, 48 bits virtual
power management:
```

Multiple processors

Primitives:

- Processes
- Threads
- Interprocess Communication (IPC)
- Synchronization primitives (resource race/race condition)

Technologies: OpenMP, Boost threads, Java threads, Intel TBB, Java.util.concurrent, Fork/Join Framework, CUDA, OpenCL

Multiple computers

Primitives:

- Sockets

Technologies:

MPI, PVM, MapReduce (Hadoop, Spark, ...)

Example

Consider the problem - you need to find all Prime numbers from 1 to 10^N

How to solve the problem as efficiently as possible?

An ideal parallel algorithm

What distinguishes the parallel algorithm from the serial?

There is no overhead in a sequential algorithm:

- Synchronization
- Data exchange
- Duplication of operations

An ideal parallel algorithm

An ideal parallel algorithm has the following properties:

- There is a possibility of simultaneous execution of operations (reserve of parallelism)
- Allows for the possibility of uniform distribution of the computational operations among processors
- Has a low level of overhead

Indicators of efficiency of a parallel algorithm

- Acceleration
- Efficiency
- Cost

Acceleration

The acceleration (speedup) obtained by using a parallel algorithm for P processors is determined by the value compared to the sequential version of calculations

$$S_p(n) = T_1(n) / T_p(n)$$

Efficiency

The efficiency of usage of parallel algorithm processors in solving the problem is determined by the ratio

$$E_p(n) = T_1(n) / (pT_p(n)) = S_p(n) / p$$

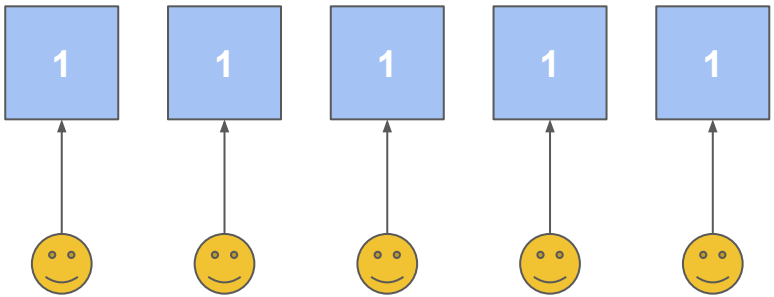
The efficiency of the parallel algorithm shows the proportion of time during which one processor is busy with calculations on average (the rest of the time it is idle)

Cost

When choosing a proper parallel method of solving the problem, it may be useful to estimate the cost of calculations, defined as the product of the time of parallel solution of the problem and the number of processors used.

$$C_p = pT_p$$

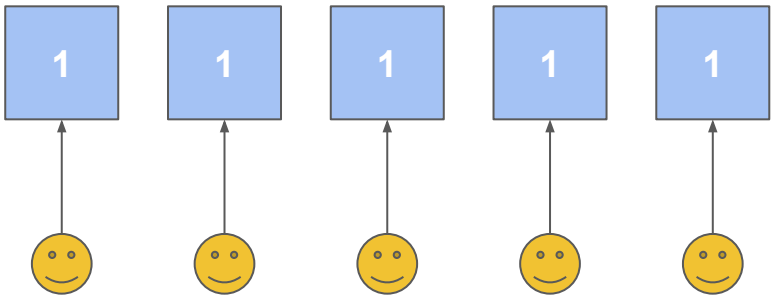
Evaluation of maximum achievable parallelism



Acceleration:

S = ?

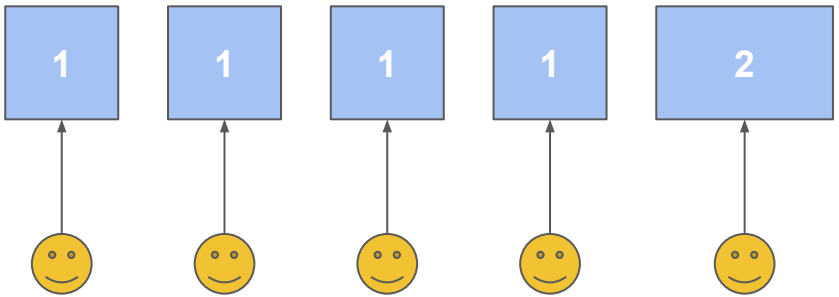
Evaluation of maximum achievable parallelism



Acceleration:

S = 5/1 = 5

Evaluation of maximum achievable parallelism



Acceleration:

S = ?

Amdahl's Law

$$S_p = \frac{1}{\alpha + \frac{1 - \alpha}{p}}$$

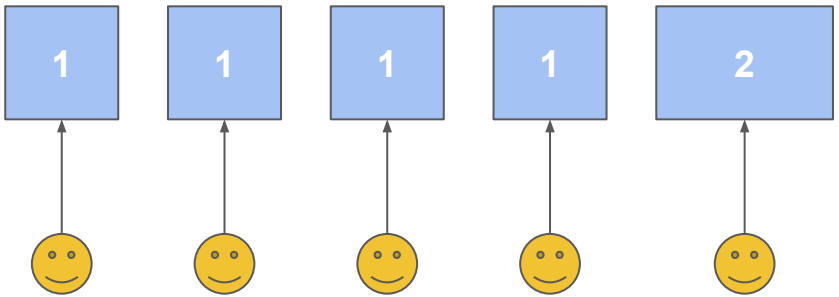
Sp - acceleration

a - percent of serial code

p - number of processors

1. The more ***a*** the less acceleration
2. The increase in computational efficiency depends on the algorithm and is bounded from above for any problem with ***a*** != 0
3. The more code to synchronize threads, the less acceleration

Evaluation of maximum achievable parallelism



Acceleration:

$1/S = 1/6 + 5/(6*5) = 1/3$

$S = 3$

Evaluation of maximum achievable parallelism

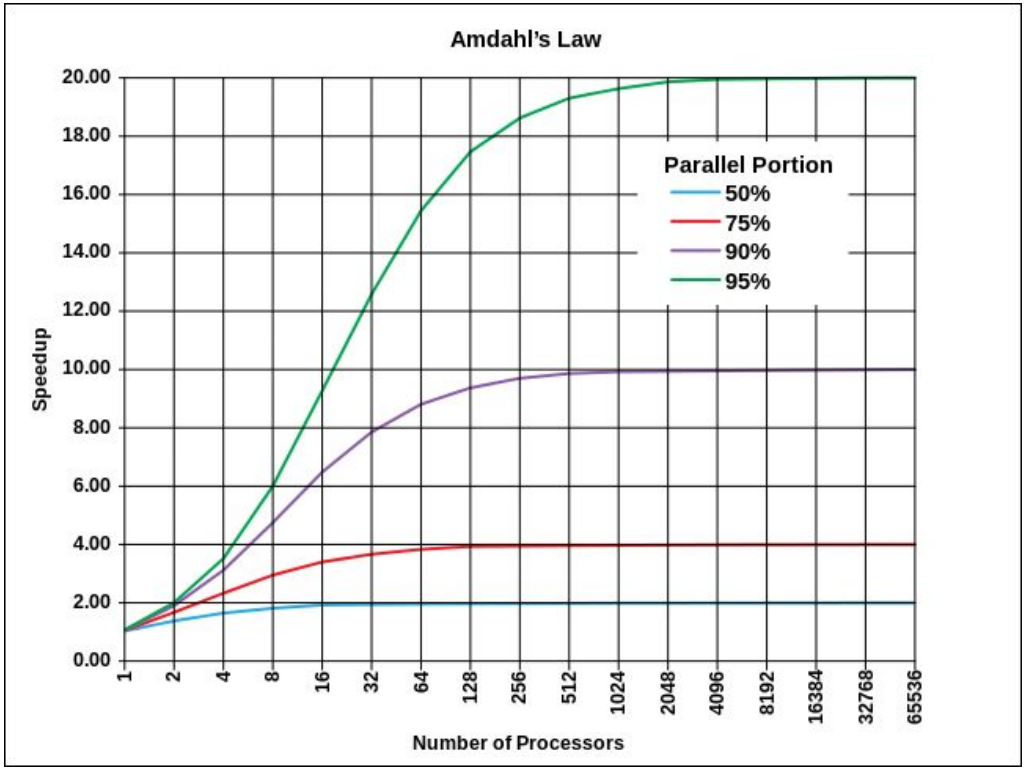
How to determine the quality of the parallel algorithm?

$$S=p \text{ и } E=1$$

Such an ideal variant is unattainable in practice, so it is considered to be effective such a parallel algorithm, the efficiency of which, at least, is limited to some constant, and does not tend to zero with the growth of **p**.

The maximum acceleration can be hindered by the presence of sequential steps in the algorithm, i.e. steps that cannot be parallelized.

Amdahl's Law


Amdahl's Law
Speedup
20.00
18.00
16.00
14.00
12.00
10.00
8.00
6.00
4.00
2.00
0.00
Parallel Portion
50%
75%
90%
95%
1
2
4
8
16
32
64
128
256
512
1024
2048
4096
8192
16384
32768
65536
Number of Processors

Literature

Maurice Herlihy. “The Art of Multiprocessor Programming”