Increasing the speed of tasks solving by parallelization of tasks.

Parallelization method is using successfully in dealing with complex and time-consuming tasks. This method can significantly increase the speed of execution of the task.

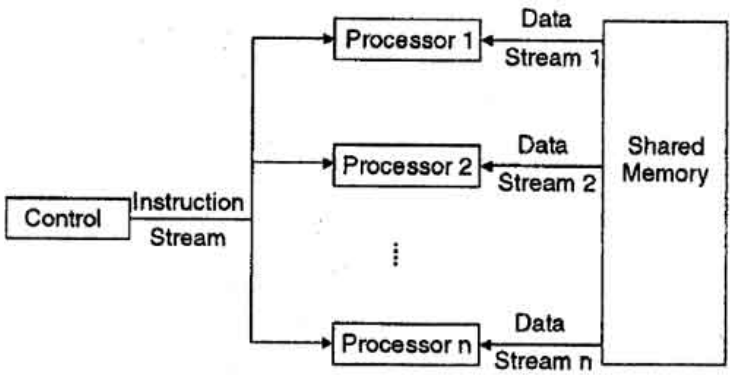
The method of parallel computing:

• A problem is broken down into several sub-tasks that are performed on different processors;

• Programmer is writing program for each subtask, which work in the background mode;

• Programs exchange the data with each other, and the programmer is following the progress of the programs and the exchanging of data stream;

• After completing the subtasks, the programmer makes data analyze.



At first glance, more CPUs are involved in solving the task, as many times and must win time. In practice, this acceleration is never achieved. The reason for this law Amdahl:

$$S \leq 1/(f+(1-f)/P)$$

where S - the acceleration of the program on P processors, and f - fraction non-parallel code in the program.

This formula implies that P-fold acceleration can be achieved only when the share of non-parallel code is 0. That is virtually impossible. Accelerating the program depending on the proportion of non-parallel code:

The number of processors	Share serial computing%				
	50	25	10	5	2
	Acceleration Program				
2	1.33	1.60	1.82	1.90	1.96
4	1.60	2.28	3.07	3.48	3.77
8	1.78	2.91	4.71	5.93	7.02
16	1.88	3.36	6.40	9.14	12.31
32	1.94	3.66	7.80	12.55	19.75
512	1.99	3.97	9.83	19.28	45.63
2048	2.00	3.99	9.96	19.82	48.83

Amdahl's law shows the maximum number of processors on which the program will be carried out with a heavy performance, depending on the proportion of non-parallel code.

Therefore, this method is not always speeds up the task. Execution speed is also dependent on an enabled computer architecture, operating parallel processors, competently composed algorithm for solving tasks and subtasks, and the style of writing programs.

When using this method, the programmer can use the following tools:

- *OpenMP* - a standard application interface for parallel systems with shared memory.
- *POSIX Threads* - standard implementation of solving flows.
- *Windows API* - threaded applications to C ++.
- *PVM (Parallel Virtual Machine)* - it connects computers to the general computational resource.
- *MPI (Message Passing Interface)* - standard messaging systems between the parallel working processes.

Therefore, to achieve maximum efficiency, you must first optimize all existing processes, analyze the technical capabilities of computers and determine whether to apply the method of parallelization of the problem in this situation.

References:

1. Alaa Ismail El-Nashar, «To parallelize or not to parallelize, speed up issue»

2. Sitkevich T.A. Syurin V.N., "Parallel Computing Environment"

3. Nemnyugin Sergey A., "Introduction to Parallel Programming with MPI»

4. V.E. Karpov, "Introduction to the parallelization of algorithms and programs"

5. V.N. Datsyuk, A.A. Bukatov, A.I. Zhegulo, "Programming

multiprocessor systems "

6. www.en.wikipedia.org/wiki/Parallel_processing

7. www.intuit.ru/studies/courses/4807/1055/lecture/16378

8. www.viva64.com/ru/b/0051/