DOI: 10.14529/jsfi210406

Scalability as a Key Property of Mapping Computational Tasks to Supercomputer Architecture

Alexander S. Antonov[1]

© The Author 2021. This paper is published with open access at SuperFri.org

When solving complex computational problems on modern supercomputers, an increasingly important role is played by the scalability property, which characterizes the ability of applications to adapt to various degrees of parallelism of computing systems.

Keywords: scalability, supercomputer, AlgoWiki, parallel structure, problems, methods, algorithms, implementations, computing platforms.

1. Description of the Algorithms Properties in the AlgoWiki Encyclopedia

The project of the AlgoWiki Open Encyclopedia [1] implements on the Internet the possibilities for uniting the efforts of the computing community to form a bank of descriptions of the computational algorithms properties according to a single universal scheme. Within the framework of this scheme, two parts of the description are distinguished – the first describes the properties of the algorithms themselves, and the second describes the properties of software implementations and the dynamic characteristics of their execution on specific computing systems [2].

Algorithm descriptions are the most important, but not the only part of the AlgoWiki Open Encyclopedia. Within the framework of the encyclopedia, hierarchical descriptions are built in the form of chains "problem–method–algorithm–implementation–computer" [3], corresponding to the scheme actively used in practice for solving problems on high-performance computing systems of various architectures. Currently, work is underway to close such chains with descriptions of supercomputers in the framework of the Algo500 project [4].

2. Scalability

The scalability property is one of the most studied properties of parallel programs [5]. Different authors interpret this concept in different ways. In the most general approach [6], *scalability* is understood as a property of a parallel program that characterizes the dependence of the entire set of dynamic characteristics of this program on the set of its launch parameters.

From the dynamic characteristics of programs in practice, the achieved value of performance is considered most often, and from the launch parameters – the number of parallel processes and the computational complexity of the problem being solved. Taking these three values together it will determine the most commonly used types of scalability. So, most often in practice, the dependence of performance on the number of application processes is considered, fixing the size of the task (computational complexity). This dependence is commonly referred to as *a strong scalability*. In many cases, unless a special caveat is made, it is strong scalability that is meant.

Many real-life problems are designed in such a way that their computational complexity can be regulated by setting a few simple parameters. If so, then it makes sense to simultaneously increase both the number of computational processes and the computational complexity of the

[1]Lomonosov Moscow State University, Moscow, Russian Federation

problem. If the system performance continues to grow, then the program is said to have *weak scalability*. In addition to strong and weak, some other types of scalability are also known, but it is these two types that are encountered most often in practice.

One of the well-known scalability metrics is the isoefficiency function [7], which shows the required level of growth in the computational complexity of a problem to maintain a given level of efficiency (understood as the ratio of acceleration to the number of processes used). In the framework of the AlgoWiki Open Encyclopedia, another version of the scalability metric of algorithm implementations has been proposed, based on empirical data.

3. Scalability Obstacles

Various obstacles that hinder the achievement of high scalability rates on various high-performance computing systems are of interest for study. It is important that scalability interference can occur at all stages of the mapping of the problem being solved to the target supercomputer, which we propose to fix using the mechanisms of the AlgoWiki encyclopedia. The reasons for poor scalability may lie in the structure of the problem itself, in the chosen method for solving it, in a specific algorithm, its software implementation, as well as on the side of the computer's software and hardware environment.

3.1. Algorithm-Level Obstacles

Sometimes the algorithm itself is designed in such a way that no implementation of it will effectively scale to the available computing system. This can be determined, for example, by a small number of operations of the considered algorithm or by the presence of true information dependencies between the available operations. Ultimately, this leads to the fact that the use of the available parallelism resource is justified only for a relatively small number of computational processes. When scaling to large configurations, the computational load on each process becomes unreasonably small, and the gain from parallelization is lost against the background of additional overhead costs. In this case, they say that the limit of decomposition of the computational domain has been reached, and no additional efforts will give a fundamentally better scalability without changing the algorithmic basis itself, which should be reflected in the description of the algorithm.

3.2. Obstacles at the Software Implementation Level

In some cases, the considered algorithm has a sufficiently large parallelism resource to achieve good scalability, but the situation is spoiled by an ineffective software implementation.

So, in many algorithms that work with matrices of a special type (for example, triangular), the parallelism resource is sufficient to scale to large computing systems. However, when using parallelism of only the outer loop using a static distribution (Fig. 1), the operations of the algorithm will be extremely unevenly distributed among the processes. Some processes will get significantly more iterations of the inner loop than others, and such uneven load will lead to downtime of some processes, which will significantly affect scalability in general.

In other cases, scalability can be affected, for example, by poor choice of functionality provided by a particular parallel programming technology. For example, the overuse of blocking communications in MPI can lead to downtime waiting for data or even deadlocks. Figure 2 shows

```
#pragma omp for schedule (static)
for(i=0; i<N; i++)
   for(j=i; j<N; j++)
      a[i][j]=...
```

Figure 1. Parallelizing a fragment of triangular matrix computations

a typical example of an inefficient implementation of transfers from all communicator processes to process number `0` in a program using MPI technology. This fragment implements a strict order of receiving data in the order of increasing process numbers in the communicator. In many cases, it would be more efficient to abandon this ordering by accepting data from ready-made processes, for which you can use the `MPI_ANY_SOURCE` option. This can especially strongly affect the performance of large configurations of computing systems when the rest of the program is executed asynchronously.

```
if(rank != 0)
   MPI_Send(&a, 1, MPI_DOUBLE, 0, 1, MPI_COMM_WORLD);
else
   for(int i = 1; i < size; i++)
      MPI_Recv(&a, 1, MPI_DOUBLE, i, 1, MPI_COMM_WORLD, &status);
```

Figure 2. Sending data from all processes to one (MPI technology)

Excessive process synchronization also often affects efficiency and scalability. Too many spawned parallel threads can add a significant amount of overhead. Using unsuccessful communication schemes can result in additional wasted time. There can be a lot of such reasons; it is planned to devote a separate study to their analysis.

3.3. Obstacles at the System Software Stack Level

Choosing the right software and configuring it efficiently is also critical to achieving a high level of scalability. Here it is important to correctly configure not only compilers and user libraries, but also, for example, the resource manager used on the computing system – the efficiency of its execution can greatly depend on how it distributes the processes of a user program among computational nodes. Other system settings inaccessible to an ordinary user are also important, for example, routing settings in a communication network. Using different routing algorithms for a large number of processes or for long messages may result in better scalability of applications.

3.4. Obstacles at the Hardware and Environmental Level

In some cases, the very choice of the target supercomputer may be unfortunate in terms of achieving a high level of scalability. For example, the computational structure of a fragment can be effectively implemented on an existing supercomputer, but the amount of RAM and the required speed of working with it for large tasks may be insufficient. In such cases, it is necessary to formulate the requirements for the target computing system, taking into account the tasks to be solved, using the principles of supercomputer code design [8].

Good scalability can be hindered by the use of quite suitable computing systems. Thus, most supercomputers are used in a shared access mode. Most often, computing nodes in such systems are exclusively allocated for the task, but a number of important system resources (such as a communication network, storage system) are shared by all users of the system. Therefore, any currently running task can affect the scalability of our application by organizing heavy traffic over the network or actively engaging in data I/O.

4. AlgoWiki Encyclopedia and Scalability

The description of the scalability properties in the Open Encyclopedia AlgoWiki is given in the section 2.4 of the unified algorithms description. Until now, a scheme for describing the scalability properties of the algorithm itself has not been proposed, therefore, in most descriptions of algorithms, special attention is paid to the scalability of the algorithm implementation. For this, experimental data are obtained from runs of the existing implementation of the algorithm on some high-performance computing system. The key point of this section is to show the real scalability parameters of the implementation of this algorithm on various computing platforms, depending on the number of processors and the size of the task. In this case, it is important to choose such a ratio between the number of processors and the size of the task in order to reflect all characteristic points in the behavior of a parallel program, in particular, the achievement of maximum performance, as well as subtle effects arising, for example, due to the block structure of the algorithm or the memory hierarchy.

It is proposed to implement an integrated approach to describing scalability in the Open Encyclopedia AlgoWiki, reflecting at all stages of the chains "problem–method–algorithm–implementation–computer" all possible obstacles for the effective adaptation of the properties of computational problems to an increase in the degree of parallelism of computing systems.

Conclusion

This article discusses the issue of studying the scalability property, which is one of the key properties characterizing the quality of parallel programs execution. The idea of expanding the description of the algorithms properties within the framework of the AlgoWiki project is proposed, which allows to present the most complete approach to the study of potential problems with scalability by analyzing all steps in solving problems. Possible obstacles to achieving good scalability can be observed at any stage of the mapping of the problems being solved to the architecture of the target supercomputers. A detailed study and description of such obstacles by the user community within the AlgoWiki framework will help to avoid them, achieving a high level of scalability of parallel applications.

Acknowledgements

Described results were obtained at Lomonosov Moscow State University with the financial support of the Russian Science Foundation, agreement № 20–11–20194. The research is carried out using the equipment of the shared research facilities of HPC computing resources at Lomonosov Moscow State University [9].

This paper is distributed under the terms of the Creative Commons Attribution-Non Commercial 3.0 License which permits non-commercial use, reproduction and distribution of the work without further permission provided the original work is properly cited.

References

1. Open Encyclopedia of Parallel Algorithmic Features. http://algowiki-project.org/en, accessed: 2021-11-22

2. Antonov, A., Voevodin, Vad., Voevodin, Vl., Teplov, A.: A Study of the Dynamic Characteristics of Software Implementation as an Essential Part for a Universal Description of Algorithm Properties. 24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing Proceedings, February 17-19, 2016. pp. 359–363. IEEE Computer Society (2016). http://dx.doi.org/10.1109/PDP.2016.24

3. Antonov, A., Frolov, A., Konshin, I., Voevodin, Vl.: Hierarchical Domain Representation in the AlgoWiki Encyclopedia: From Problems to Implementations. Communications in Computer and Information Science, vol. 910, pp. 3–15. Springer (2018). http://dx.doi.org/10.1007/978-3-319-99673-8_1

4. Antonov, A., Nikitenko, D., Voevodin, Vl.: Algo500 – a New Approach to the Joint Analysis of Algorithms and Computers. Lobachevskii Journal of Mathematics 41(8), 1435–1443 (2020). http://dx.doi.org/10.1134/S1995080220080041

5. Scalability. In: Padua, D. (eds) Encyclopedia of Parallel Computing. Springer, Boston, MA (2011). https://doi.org/10.1007/978-0-387-09766-4_2046

6. Antonov, A., Teplov, A.: Generalized approach to scalability analysis of parallel applications. Algorithms and Architectures for Parallel Processing - ICA3PP 2016 Collocated Workshops: SCDT, TAPEMS, BigTrust, UCER, DLMCS, Granada, Spain, December 14-16, 2016, Proceedings. Lecture Notes in Computer Science, vol. 10049, pp. 291–304. Springer (2016). http://dx.doi.org/10.1007/978-3-319-49956-7_23

7. Grama, A.Y., Gupta, A., Kumar, V.: Isoefficiency: measuring the scalability of parallel algorithms and architectures. IEEE Parallel Distrib. Technol. 1(3), 12–21 (1993). https://doi.org/10.1109/88.242438

8. Dosanjh, S.S., Barrett, R.F., Doerfler, D.W., et al.: Exascale design space exploration and co-design. Future Generation Computer Systems 30, 46–58 (2014). http://dx.doi.org/10.1016/j.future.2013.04.018

9. Voevodin, Vl., Antonov, A., Nikitenko, D., et al.: Supercomputer Lomonosov-2: Large Scale, Deep Monitoring and Fine Analytics for the User Community. Supercomputing Frontiers and Innovations 6(2), 4–11 (2019). http://dx.doi.org/10.14529/jsfi190201