DOI: 10.14529/jsfi180101

AlgoWiki Project as an Extension of the Top500 Methodology

Alexander Antonov[1], *Jack Dongarra*[1,2], *Vladimir Voevodin*[1]

The AlgoWiki project is dedicated to describing the parallel structure and key features of various algorithms. The descriptions are intended to provide complete information about algorithms' properties, which are needed to adequately assess their implementation efficiency for any computing platform. This work sets out the key areas for further development of the project which were recently developed based on working with the AlgoWiki encyclopedia. We are suggesting an approach to extend the Top500 methodology, which is commonly used to compare various computing platforms.

Keywords: AlgoWiki, parallel structure, algorithm's properties, Top500 methodology, problems, methods, algorithms, implementations, computing platforms.

Introduction

The computing world is constantly changing, and there are numerous reasons for this. New problems appear regularly that require increasingly powerful computing platforms. New ideas appear that are reflected in the computer architecture and help to improve performance or have a positive impact on energy consumption and cost. Technological progress results in developing and utilizing new computing environments which, unlike classical computers, can be highly heterogeneous and distributed. Ultimately these changes result in the need to carefully review algorithm structure and properties in order to answer the main question: can a problem be solved within a certain predefined level of efficiency (e.g., within a reasonable time), and if so, how can this be done? If the answer is obviously positive, all other questions are irrelevant; otherwise a solution needs to be found. The main question is: does the algorithm have properties that match well with specific features of the computing system's infrastructure? Is it possible to find the minimum spanning tree of a graph with 2^{28} nodes on a vector computer? Is it possible to effectively solve large sparse linear equation systems in distributed computing environments? These questions cannot be answered without understanding the algorithm's properties.

Today, descriptions of various algorithms can be found in numerous books, systems, online resources and other sources [1–6]. Some focus on a mathematical formulation, others show a possible software implementation or study serial complexity. However, the main feature of modern computing platforms is a high degree of parallelism and special memory structure. This is what should be considered first of all if we wish to talk about efficiently implementing algorithms on computing systems at any level, from mobile devices to supercomputers.

1. About the AlgoWiki Project

The AlgoWiki project [7–9] is dedicated to describing the structure and key features of various algorithms. All descriptions follow the same structure (`http://algowiki-project.org/en/Description_of_algorithm_properties_and_structure`), which allows for easy comparison of various algorithms. The descriptions are intended to provide complete information about algorithm's properties, which is needed to ensure their efficient implementation on any computing platform. Each algorithm description in AlgoWiki is divided into two parts. The first part

[1] Lomonosov Moscow State University, Moscow, Russian Federation
[2] University of Tennessee, Knoxville, USA

of the description contains information that does not depend on the software implementation or computing platforms used. This is the theoretical potential of the algorithm determined by mathematics which we can rely on for implementations on any computing platform. The second part of the algorithm's description is oriented towards practical application, as it considers the interconnection between the algorithm's properties, specific parallel programming technologies and various classes of computing systems.

The project has been implemented using Wiki technologies, similar to the Wikipedia project: existing algorithm descriptions are available to everyone, and at the same time all experts can contribute their knowledge to AlgoWiki by adding descriptions of new algorithms or by making the information more exact for the existing ones.

2. The AlgoWiki Project and Top500 Methodology

The project currently presents a multitude of algorithms from various areas: linear algebra, graph algorithms, sorting algorithms, quantum system modeling algorithms, etc. The project is continually growing, covering more and more new areas including descriptions of new algorithms. Whenever a scientist works on an algorithm, he or she creates a new article in AlgoWiki, which contains a description of the algorithm's theoretical potential and particular features regarding its implementation on various computers.

In this regard, AlgoWiki offers a good basis for the natural extension of the Top500 methodology used to compare computing systems today. Linpack [10] was historically the first widely adopted approach which led to the creation of the list of the most powerful supercomputers (`http://top500.org/`). However, Linpack only reflects one aspect of computing platforms. That is why other tests were suggested later which became the basis for the Graph500 [11] and HPCG [12] benchmarks. All three ratings use the same technique: a basic algorithm is chosen and its software implementation is written and executed on each computing system in question, which results in a number that is used to judge the computer's properties. The number helps easily compare different computers to one another, including a compilation of the ratings described above.

The AlgoWiki project can be used to extend on this methodology. In fact, AlgoWiki offers a multitude of descriptions for very diverse algorithms, for which we have execution data on multiple computing platforms. The underlying algorithms for Linpack, Graph500 and HPCG, among others, are represented in AlgoWiki and correspond to three points out of the total multitude of algorithms in the project. Giving the computing community an opportunity to save the execution results for any algorithm, we can substantially extend the possibilities for comparing computing platforms. Using the AlgoWiki potential, we can move from three points (corresponding to Linpack, Graph500 and HPCG) to an analysis based on dozens, if not hundreds of various algorithms. We do not have to select every AlgoWiki algorithm for a detailed analysis and comparison; instead we can focus on the most interesting ones. If the corresponding algorithm is missing from AlgoWiki, it can be added, establishing the first step for formulating the respective new rating.

This extension of the Top500 methodology within AlgoWiki encyclopedia has several important implications. We can not only compare the results of various algorithms on different computers, but also analyze and understand the reasons behind these results: detailed descriptions of all algorithms are always at hand in AlgoWiki. It is also important that AlgoWiki can be used to store more than just the results obtained for record-setting computer configurations and

very large sets of input data: lesser values can be of substantial practical interest and are also available for analysis. In this respect, ratings like the Top500 following any AlgoWiki algorithm are just the tip of the iceberg representing the entire multitude of data stored in AlgoWiki for each specific algorithm.

3. Problems, Methods, Algorithms, Implementations, Computing Platforms

In practice, the possibilities enabled by AlgoWiki for analyzing the Algorithm–Computer combinations are much wider. The classification of algorithms in AlgoWiki is structured to support the clear identification of three levels: problem, method, algorithm (`http://algowiki-project.org/en/Algorithm_classification`). These three levels are marked with special icons in the classification, but in reality AlgoWiki has five levels:

Problem → Method → Algorithm → Implementation → Computing platform.

When data on an algorithm's execution on a particular computing system are submitted to AlgoWiki, information about the entire chain is stored, from Problem to Computing Platform. This gives extra freedom to perform comparisons and analyses. In particular, a researcher armed with the data structure in this manner can query the AlgoWiki database to make the following comparisons:

- computer performance achieved using different methods to solve the same problem;
- computer performance achieved using different methods to solve the same problem with fixed data size;
- the time to solve a problem of fixed size using different methods to address the problem;
- computer performance achieved using different implementations of the same method to address the same problem;
- the time to solve a problem of fixed size using different methods to address it in clusters containing, for example, up to 128 nodes;
- methods that demonstrate the maximum/minimum/predefined efficiency for a given class of computers; and many others.

In addition to analysis based on parameters like time, performance and efficiency, it is also possible to conduct a different kind of qualitative analysis within AlgoWiki, in particular to find:

- algorithms used to solve a given problem;
- problems a given algorithm is used to solve;
- algorithms that are used to solve a given problem with serial complexity below $O(n^2)$;
- all method-computer pairs used to solve a given problem, where the method used has serial complexity below $O(n^2)$ and parallel complexity below $O(n)$, while the implementation efficiency on a computer exceeds 40%.

For any elements of the chain indicated above: Problem, Method, Algorithm, Implementation and Computing Platform, the relevant values can be set as constants, while the others can be variable; this allows new ratings to be built or to find combinations that best suit required conditions. For example, the data in Tab. 1 show the performance of various computers (in MTEPS) when solving the "Strongly Connected Components Search" problem for two graph sizes with 2^{18} and 2^{20} nodes. Each line in the AlgoWiki table additionally contains a detailed description of the computer environment from which this data were obtained: a link to the implementation (executable file or source code), the number of compute nodes and cores, the

Table 1. Strongly Connected Components: performance of various implementations on different computers for two types of graphs (RMAT and SSCA–2) and number of nodes equals to 2^{18} and 2^{20}

Implementation	Computing Platform	MTEPS	GraphType	GraphSize
Ligra	Lomonosov-2 (x86)	830.0	RMAT	2^{20}
RCC for GPU	Lomonosov-2 (NVIDIA P100)	634.0	RMAT	2^{20}
GAP	Lomonosov-2 (x86)	547.0	RMAT	2^{20}
RCC for PU	Lomonosov-2 (x86)	418.0	RMAT	2^{20}
PBGL MPI	IBM BlueGene/P	232.9	RMAT	2^{20}
RCC for GPU	Lomonosov-2 (NVIDIA K40)	195.0	RMAT	2^{18}
RCC for PU	IBM Regatta	53.6	SSCA–2	2^{18}
PBGL MPI	IBM BlueGene/P	45.7	RMAT	2^{20}
RCC for PU	Lomonosov (x86)	41.0	RMAT	2^{20}
RCC for PU	IBM Regatta	36.9	RMAT	2^{18}
RCC for PU	Lomonosov (x86)	32.5	RMAT	2^{20}
PBGL MPI	IBM BlueGene/P	13.1	SSCA–2	2^{18}
RCC for PU	Lomonosov (x86)	10.1	SSCA–2	2^{20}
RCC for PU	Lomonosov (x86)	8.3	SSCA–2	2^{20}
PBGL MPI	Lomonosov NVIDIA 2090)	2.3	SSCA–2	2^{18}
PBGL MPI	IBM BlueGene/P	0.2	RMAT	2^{20}
PBGL MPI	IBM BlueGene/P	0.1	SSCA–2	2^{18}

Table 2. Source Shortest Paths: performance of different implementations of different algorithms on various computers for graphs with number of nodes equals to 2^{20} and 2^{21}

Method	Implementation	Computing Platform	MTEPS	GraphSize
Bellman–Ford	RCC for GPU	Lomonosov	1309.0	2^{20}
Bellman–Ford	Ligra	Lomonosov–2	1035.0	2^{21}
Delta–Stepping	PBGL MPI	Cluster “Angara”	809.5	2^{21}
Delta–Stepping	GAP	Lomonosov–2	616.0	2^{21}
Delta–Stepping	GAP	Lomonosov–2	512.0	2^{20}
Bellman–Ford	Ligra	Lomonosov–2	511.0	2^{20}
Bellman–Ford	RCC for GPU	Lomonosov	452.9	2^{20}
Bellman–Ford	RCC for CPU	Lomonosov	435.0	2^{21}
Bellman–Ford	RCC for CPU	Lomonosov–2	426.0	2^{21}
Bellman–Ford	RCC for CPU	Lomonosov–2	418.0	2^{20}
Bellman–Ford	Graph500 MPI	Lomonosov	350.0	2^{20}
Bellman–Ford	RCC for CPU	Lomonosov	204.1	2^{20}
Bellman–Ford	RCC for CPU	Lomonosov	183.5	2^{20}
Delta–Stepping	PBGL MPI	Lomonosov	174.0	2^{21}
Dijkstra’s	PBGL MPI	Cluster “Angara”	150.0	2^{20}
Delta–Stepping	PBGL MPI	Lomonosov	124.1	2^{21}
Bellman–Ford	Graph500 MPI	Lomonosov	120.0	2^{20}
Bellman–Ford	Graph500 MPI	Lomonosov	18.0	2^{20}
Bellman–Ford	Graph500 MPI	Lomonosov	11.8	2^{21}
Dijkstra’s	PBGL MPI	IBM BlueGene/P	8.9	2^{20}
Dijkstra’s	PBGL MPI	Lomonosov	5.3	2^{21}
Delta–Stepping	PBGL MPI	IBM BlueGene/P	3.8	2^{20}
Dijkstra’s	PBGL MPI	Cluster “Angara”	2.5	2^{21}
Delta–Stepping	PBGL MPI	IBM BlueGene/P	1.3	2^{20}
Dijkstra’s	PBGL MPI	IBM BlueGene/P	0.6	2^{20}

compiler used, compiler options, settings for generating input graphs and other relevant parameters. Details on the implementations presented in the table can be found here: Ligra [13], GAP [14], PBGL [15], “RCC for CPU/GPU” correspond to an AlgoWiki user’s own implementations for CPU/GPU. Table 2 compares different implementations of different algorithms for solving the “Single Source Shortest Paths” problem on different platforms, for graphs with 2^{20} and 2^{21} nodes.

It should be noted that in these examples, specific performance values are obtained by AlgoWiki users and are largely determined by their experience and diligence. These are not necessarily the top performance figures: they show results obtained by people in practice. There is only one requirement: when submitting data in the AlgoWiki database, users must provide

all of the information that would be needed to reproduce and verify the results, and possibly improve upon them in the future.

Conclusion

The first stage of the AlgoWiki project was geared toward achieving the following two goals: developing a technology for describing the parallel structure of algorithms, and widening the project database with descriptions of actual algorithms. These goals have been accomplished, which enables further project development in a number of new areas. In particular, one of those areas is to extend the Top500 methodology used to compare high-performance computing systems. Implementing this will require expanding the project's functionality, giving AlgoWiki users a chance to save data on algorithm execution parameters in the project database. As a result, the project will not only provide detailed algorithm descriptions, but also enable the review and comparison of algorithm execution results on any computing platform.

Acknowledgements

The results were obtained in Lomonosov Moscow State University with the financial support of the Russian Science Foundation (Agreement № 14–11–00190). The research is carried out using the equipment of the shared research facilities of HPC computing resources at Lomonosov Moscow State University.

References

1. Press, W., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical Recipes in C. Cambridge University Press, second edition (1992)

2. Barrett, R., Berry, M., Chan, T.F., Demmel, J., Donato, J., Dongarra, J., Eijkhout, V., Pozo, R., Romine, C., Van der Vorst, H.: Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition, SIAM (1994), `http://www.netlib.org/linalg/html_templates/Templates.html`, accessed: 2018-03-22

3. List of algorithms. `https://en.wikipedia.org/wiki/List_of_algorithms`, accessed: 2018-03-22

4. Enabling AI in every Application. `http://algorithmia.com/`, accessed: 2018-03-22

5. ALGLIB. `http://www.alglib.net/`, accessed: 2018-03-22

6. A Library of Parallel Algorithms. `http://www.cs.cmu.edu/~scandal/nesl/algorithms.html`, accessed: 2018-03-22

7. Voevodin, Vl., Antonov, A., Dongarra, J.: AlgoWiki: an Open Encyclopedia of Parallel Algorithmic Features. In: Supercomputing Frontiers and Innovations, vol. 2, no. 1, pp. 4–18 (2015), DOI: 10.14529/jsfi150101

8. 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. In: 24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing Proceedings, 17–19 February 2016, pp. 359–363. DOI: 10.1109/PDP.2016.24

9. Voevodin, Vl., Antonov, A., Dongarra, J.: Why is it hard to describe properties of algorithms? In: Procedia Computer Science, vol. 101, pp. 4–7 (2016), DOI: 10.1016/j.procs.2016.11.002

10. Dongarra, J.J., Bunch, J.R., Moler, G.B., Stewart, G.W.: LINPACK Users' Guide. Society for Industrial and Applied Mathematics, 1979–1993.

11. Murphy, R.C., Wheeler, K.B., Barrett, B.W., Ang, J.A.: Introducing the Graph 500. In: Cray User's Group (CUG), May 5, 2010, vol. 19, pp. 45–74 (2010)

12. Heroux, M., Dongarra, J.: Toward a New Metric for Ranking High Performance Computing Systems. In: UTK EECS Tech Report and Sandia National Labs Report SAND2013–4744, June 2013.

13. Shun, J., Blelloch, G.E.: Ligra: a lightweight graph processing framework for shared memory. In: ACM Sigplan Notices, vol. 48, no. 8, pp. 135–146. DOI: 10.1145/2517327.2442530

14. Beamer, S., Asanovi, K., Patterson, D.: The GAP Benchmark Suite. arXiv:1508.03619 [cs.DC] (2015)

15. Parallel Boost Graph Library. `http://www.boost.org/doc/libs/1_51_0/libs/graph_parallel/doc/html/index.html`, accessed: 2018-03-22