ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 2014

Seria: MATEMATYKA STOSOWANA z. 4 Nr kol. 1920

Zbigniew MARSZAŁEK, Dawid POŁAP, Marcin WOŹNIAK

Institute of Mathematics
Silesian University of Technology

ON BUBBLE SORT PERFORMANCE FOR LARGE DATA SETS AND NOSQL DATABASE SYSTEMS

Summary. In the paper we discuss performance of bubble sort algorithm. Research results discussed and described in this article help to evaluate this method when used in NoSQL database systems for large amounts of the input data. In the article is analyzed bubble sort algorithm for large scale data sets in two versions: classic version and modified version with logic control of order.

O WYDAJNOŚCI SORTOWANIA BĄBELKOWEGO DLA DUŻYCH ZBIORÓW DANYCH I SYSTEMÓW BAZODANOWYCH TYPU NOSQL

Streszczenie. Sortowanie jest jednym z ważniejszych problemów współczesnej informatyki. Obecnie komputery muszą pracować na coraz większych ilościach danych, dlatego w niniejszym artykule przedstawiamy analizę algorytmu sortowania bąbelkowego pod względem jego własności dla dużych zbiorów danych i baz tybu NoSQL. W analizie zbadaliśmy wersję klasyczną i zmodyfikowaną z funkcją kotroli ułożenia elementów.

2010 Mathematics Subject Classification: 68W40, 68Q25, 68P01, 68P10.
Keywords: computer algorithm, large data sets, NoSQL database systems.
Corresponding author: Z. Marszałek (Zbigniew.Marszalek@polsl.pl).
Received: 30.06.2014.
This project has been financed from the funds of Institute of Mathematics, Silesian University of Technology.

1. Introduction

Many publications discuss performance of sorting algorithms. In [21] is presented analysis of some dedicated versions of sorting algorithms for large data sets. While in [20] is discussed possible extension and improvements of sorting methods for large data sets and NoSQL database systems. Some research on sorting methods are presented in [15,16,22,23,25]. While in [15] is discussed novel version of merge sort and it's performance for NoSQL database systems. In [16] is presented possibility of efficient construction of NoSQL systems. Finally in [23] are discussed research results on sorting large data sets. Here we compare classic bubble sort (please see [20]) and bubble sort with logic control of order (please see [19]) performance for large data sets.

The bubble sort algorithm was examined and results were compared in terms of performance, like CPU usage and statistical stability of work (for more details please see [20,21]). The aim of the analysis and comparison is to determine which of the examined versions has better properties for large data sets and NoSQL systems. In [3,9,16,21,23,24] similar studies are shown for different methods and algorithms. Moreover some examples of performance analysis are presented in [1,2,10,11,18], where authors describe tests of various implementations of sorting algorithms. Interesting results and analysis of multi-thread algorithms are presented in [3,5–7,17]. Moreover authors of [4,8,12–14] discuss research results on adaptive and parallel algorithms, what is also important for NoSQL database systems (see [15, 16]). Therefore, here we make some assessment on bubble sort versions for large data sets.

2. Comparison and assessment

At the beginning we will compare theoretical time complexity. For more details on theory of complexity of computer methods please see [10,11,20,21]. In Table 1 we present comparison of descriptive characteristics that were determined during the study for examined methods. Classic bubble sort for large data sets was described in [20] and modified bubble sort with logic control of order was described in [19].

Compared values are presented in Figure 1. Charts in Figure 1 show that logic control function reduces complexity of bubble sort algorithm. Then, examined versions were compared in terms of other characteristics. Comparison of CPU clock cycles is shown in Figure 2. It shows that both versions behave similarly. However, for sequences of size close to 100000 elements, control function can extensively use processor. This is due to increased computational complexity of this modification.

Table 1

CPU clock cycles for bubble sort

Time complexity	Number of elements				
[sec]	3	4	5	6	7
lower bound	3	5,33	8,33	12	16,33
bubble sort with control	1,67	5,54	9,21	13,54	18,68
bubble sort	3	6	10	15	21
[sec]	8	9	10	11	
lower bound	21,33	27	33,33	40,33	
bubble sort with control	24,67	31,5	39,17	47,66	
bubble sort	28	36	45	55	

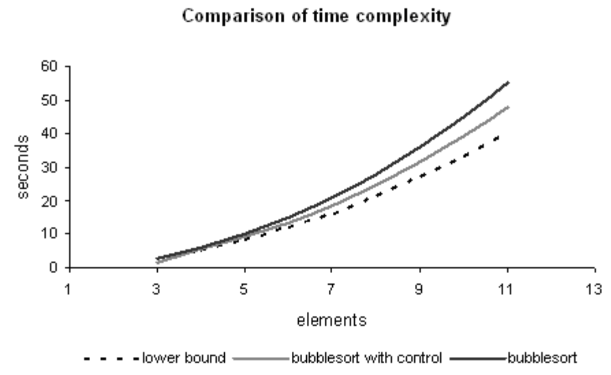

Fig. 1. Comparison chart of theoretical time complexity of bubble sort
Rys. 1. Wykres porównawczy teoretycznej złożoności czasowej badanych wersji algorytmu sortowania bąbelkowego

Analysis shows that control function speeds up sorting for large data sets. This means that the logic control of order has a positive effect on the acceleration of the sorting process. Figure 3 shows comparison of standard deviation for CPU clock cycles. Classic bubble sort seems to be more stable version. However applied control function may reduce stability of the algorithm.

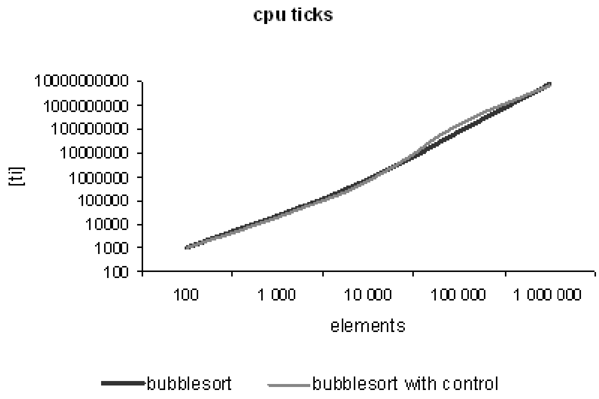

Fig. 2. Performance comparison chart of CPU clock cycles for tested versions of bubble sort algorithm
Rys. 2. Przybliżenie charakterystyki cykli zegarowych dla badanych wersji algorytmu sortowania bąbelkowego

Figure 4 shows a comparison of time standard deviation. Diagram of these characteristic indicate that both examined versions operate in similar manner.

Research results make it possible to estimate the difference between examined versions. In Figure 5–6 are shown differences between characteristics of performance and time. As a reference method was chosen classic version of bubble sort algorithm described in [19].

Analysis of CPU ticks percentage difference shown in Figure 6 shows that the control function can reduce the use of computer resources for small collections up to 10000 elements. However, for sets of size about 100000 elements positioning control function may slow down the algorithm. In Figure 6 is shown comparison of sorting time.

Based on a comparative chart in Figure 6, we conclude that the use of control function can accelerate the algorithm for large NoSQL systems. However adverse effects of proposed function can effect sorting very small collections of less than 100 elements. Despite the increase in complexity, control function (see [20]) allows to speed up sorting large data sets of more than 10000 elements. Moreover logic control of order significantly accelerates the process of sorting for sequences of more than 1000000 elements. Analysis confirms that control function speeds up the algorithm for large data sets. Therefore there comes a question, which of

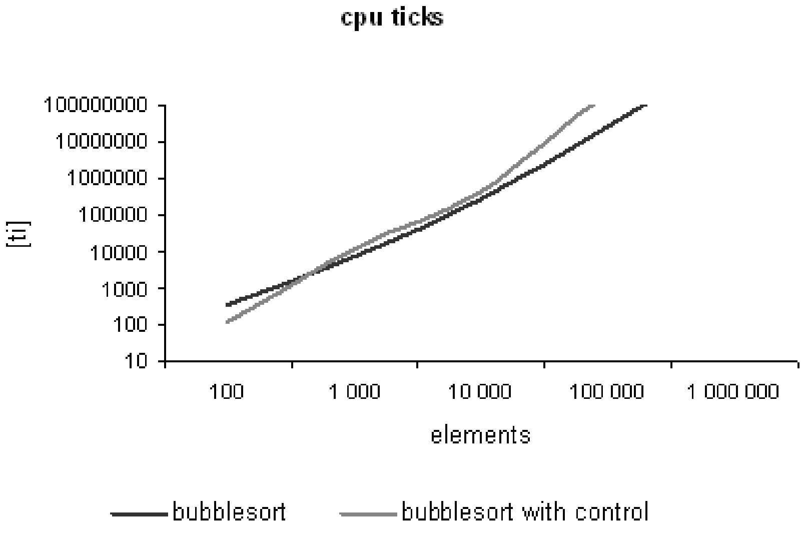

Fig. 3. Performance comparison chart of standard deviation of CPU clock cycles for examined versions of bubble sort algorithm
Rys. 3. Wykres porównawczy charakterystyki odchylenia standardowego cykli zegarowych dla badanych wersji algorytmu sortowania bąbelkowego

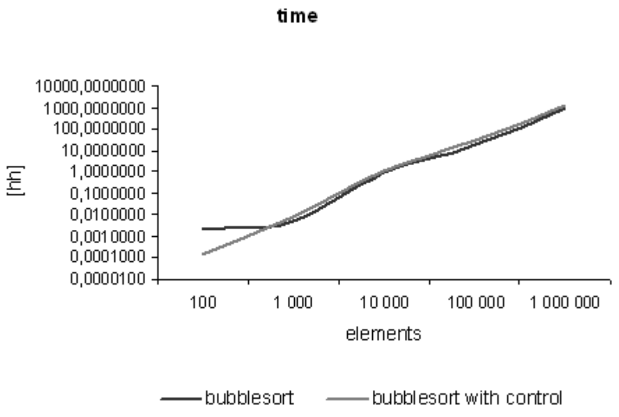

Fig. 4. Performance comparison chart of time standard deviation for examined versions of bubble sort algorithm
Rys. 4. Wykres porównawczy charakterystyki odchylenia standardowego czasu rzeczywistego dla badanych wersji algorytmu sortowania bąbelkowego

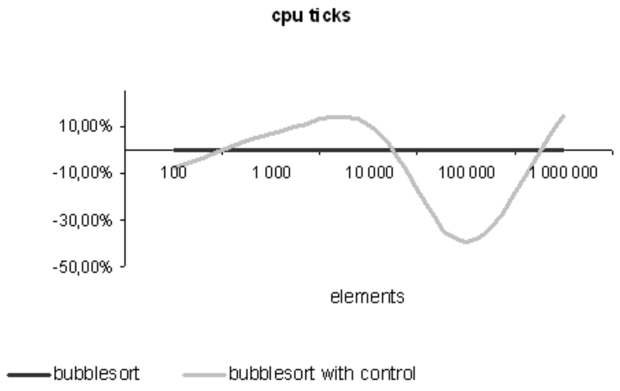

Fig. 5. Rate difference of CPU clock cycles during bubble sort with logic Control of order in relation to classic bubble sort
Rys. 5. Ilustracja procentowej różnicy cykli zegarowych procesora w trakcie algorytmu sortowania bąbelkowego z funkcją kontroli w stosunku do postaci klasycznej

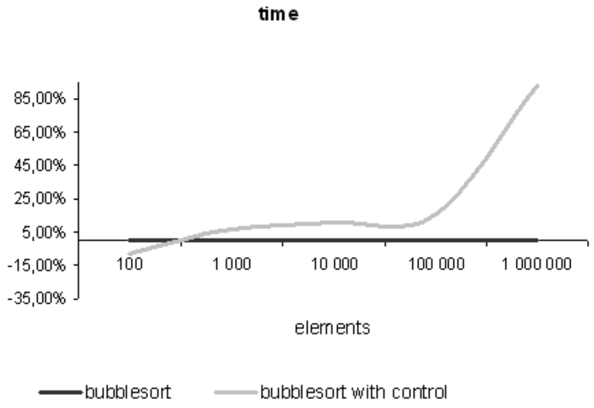

Fig. 6. Rate difference of time during bubble sort with logic control of order in relation to classic bubble sort
Rys. 6. Ilustracja procentowej różnicy czasu rzeczywistego algorytmu sortowania bąbelkowego z funkcją kontroli w stosunku do postaci klasycznej

examined versions is more stable. Comparison of coefficient of variation, calculated according to [1, 11, 15, 18, 20, 21], for both versions is shown in Figure 7–8. Table 2 shows summary comparison of algorithm variability for CPU clock cycles. Results from Table 2 were plotted in Figure 7.

Table 2
Comparison of CPU clock cycles variations

CPU tics	Number of elements		
[ti]	100	1000	10000
bubble sort	0,35	0,34	0,36
bubble sort with control	0,14	0,63	0,65
[ti]	100000	1000000	
bubble sort	0,35	0,35	
bubble sort with control	0,85	0,65	

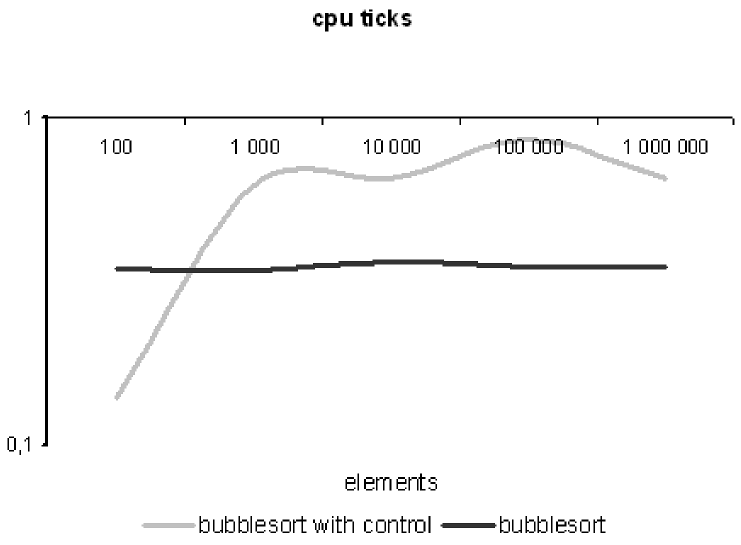

Fig. 7. Comparison of volatility of CPU clock cycles for examined bubble sort
Rys. 7. Ilustracja porównawcza zmienności cykli zegarowych procesora w trakcie algorytmu sortowania bąbelkowego

The values of variation coefficient are shown in Figure 7. Secondary axis with corresponding color enables better comparison. Analysis of Figure 8 shows that two types differ in stability of work. Classic version operates better on sequences ringing under 1000 and about 100000 elements. Version with logic control of order may behave unstable for collection close to 100000 elements. However, for large data sets of size over 1000000 elements the content control feature can reduce the use of resources and improves efficiency.

Values from Table 3 were plotted in Figure 8.

Table 3

Comparison of time variations

Time	Number of elements		
[sec]	100	1000	10000
bubble sort	0,26	0,34	0,36
bubble sort with control	0,18	0,63	0,65
[sec]	100000	1000000	
bubble sort	0,35	0,37	
bubble sort with control	0,65	0,56	

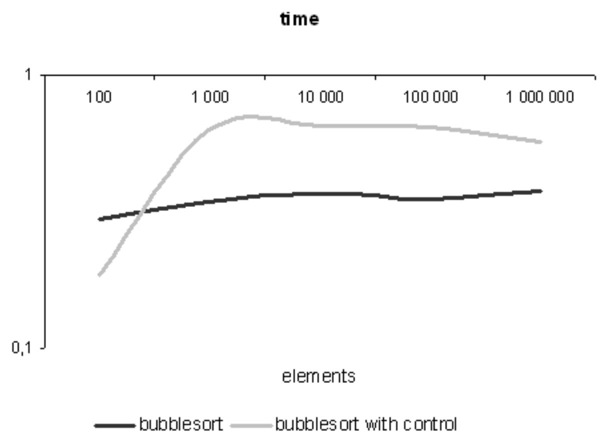

Fig. 8. Comparison of volatility of time for examined bubble sort

Rys. 8. Ilustracja porównawcza zmienności czasu rzeczywistego w trakcie algorytmu sortowania bąbelkowego

3. Conclusions

In conclusion, version of the bubble sort algorithm with logic control of order (see [19,20]) is more efficient for large data sets and allows acceleration of sorting. Volatility of measured values for modified version is similar to classic one. Therefore, it is appropriate to use it for large data sets. Interesting improvement for examined sorting algorithm can be multiprocessing or applying cloud computing.

References

1. Aho I.A., Hopcroft J., Ullman J.: *The Design and Analysis of Computer Algorithms.* Addison-Wesley, Indianapolis 1974.
2. Banachowski L., Diks K., Rytter W.: *Algorithms and Data Structures.* WNT, Warszawa 1996 (in Polish).
3. Brown M.R., Tarjan R.E.: *Design and analysis of data structures for representing sorted lists.* SIAM J. Comput. **9** (1980), 594–614.
4. Carlsson S., Chen J.: *An optimal parallel adaptive sorting algorithm.* Inform. Process. Lett. **39** (1991), 195–200.
5. Cook C.R., Kim. D.J.: *Best sorting algorithms for nearly sorted lists.* Comm. ACM **23** (1980), 620–624.
6. Dinsmore R.J.: *Longer strings for sorting.* Comm. ACM **8** (1965), 48–65.
7. Dlekmann R., Gehring J., Luling R., Monien B., Nubel M., Wanka R.: *Sorting large data sets on a massively parallel system.* In: Proceedings of the 6th IEEE Symposium on Parallel and Distributed Processing, Dallas 1994, 29–38.
8. Estivill-Castro E., Wood D.: *A genetic adaptive sorting algorithms.* Comput. J **35** (1992), 505–512.
9. Islam T., Lakshman K.B.: *On the error sensitivity of sort algorithms.* In: Proceedings of International Conference on Computing and Information, Toronto 1990, 81–85.
10. Knuth D.: *The Art of Computer Programming*, vol. 1–3. Addison-Wesley Professional, Indianapolis 2006.
11. Knuth D.E., Greene D.H.: *Mathematics for the Analysis of Algorithms*, Birkhuser, Boston 2007.
12. Levcopolos C., Petersson O.: *A note on adaptive parallel sorting.* Inform. Process. Lett. **33** (1985), 187–191.

13. Levcopolos C., Petersson O.: *An optimal parallel algorithm for sorting presorted files.* Lecture Notes in Comput. Sci. **338** (1988), 154–160.
14. Levcopolos C., Petersson O.: *Splitsort an adaptive sorting algorithm.* Inform. Process. Lett. **39** (1991), 205–211.
15. Marszałek Z., Połap D., Woźniak M.: *On preprocessing large data sets by the use of triple merge sort algorithm.* Proceedings of the International Conference on Advances in Information Processing and Communication Technologies (IPCT 2014), The IRED – Digital Seek Library, Santa Barbara 2014, 65–72.
16. Marszałek Z., Woźniak M.: *On possible organizing Nosql database systems.* Int. J. Information Science and Intelligent System **2**, no. 2 (2013), 51–59.
17. Sinha R., Zobe J.: *Cache-conscious sorting of large sets of strings with dynamic tries.* J. Exp. Algorithmics **9** (2004), article no. 1.5.
18. Weiss M.A.: *Data Structures and Algorithm Analysis in C++.* Prentice Hall, Indianapolis 2013.
19. Woźniak M., Marszałek Z.: *On some properties of bubble sort with logic control of order for large scale data sets.* Zesz. Nauk. PŚl., Mat. Stosow. **3** (2013), 47–58.
20. Woźniak M., Marszałek Z.: *Selected Algorithms for Sorting Large Data Sets.* Wyd. Pol. Śl. (Silesian University of Technology Press), Gliwice 2013.
21. Woźniak M., Marszałek Z.: *Extended Algorithms for Sorting Large Data Sets.* Wyd. Pol. Śl. (Silesian University of Technology Press), Gliwice 2014.
22. Woźniak M., Marszałek Z., Gabryel M.: *The analysis of properties of insertion sort algorithm for large data sets.* Zesz. Nauk. PŚl., Mat. Stosow. **2** (2012), 45–55.
23. Woźniak M., Marszałek Z., Gabryel M., Nowicki R.K.: *Modified merge sort algorithm for large scale data sets.* Lecture Notes in Artificial Intelligence **7895** (2013), 612–622.
24. Woźniak M., Marszałek Z., Gabryel M., Nowicki R.K.: *On quick sort algorithm performance for large data sets.* In: Looking into the Future of Creativity and Decision Support Systems, Skulimowski A.M.J. (ed.), Progress & Business Publishers, Cracow 2013, 647–656.
25. Woźniak M., Marszałek Z., Gabryel M., Nowicki R.K.: *Triple heap sort algorithm for large data sets.* In: Looking into the Future of Creativity and Decision Support Systems, Skulimowski A.M.J. (ed.), Progress & Business Publishers, Cracow 2013, 657–665.