UDC 004.75

M.Yu. YURICH, D.S. BARSUKOV, R.K. KUDERMETOV

Zaporizhzhya National Technical University, Ukraine

SORTING ALGORITHMS FOR DISTRIBUTION OF TASKS IN THE COMPUTER SYSTEM

The problem of the distribution of tasks in the computer system when the number of computers exceeds the number of tasks is distributed. The various sorting algorithms are applied for this purpose. Auto select of sorting algorithm which depending on the properties of data that represented by input arrays are proposed, in order to speed up work before the distribution of tasks based on the use of sorting algorithms. We show that the chosen solution to the problem is effective, because selection time of algorithm insignificant effect on the total time sorting. It is proved that the algorithm reduces the sorting time of input data, and so the time of the distribution of tasks in the computer system.

Keywords: *distribution of tasks, computer system, sorting, time sorting, sorting algorithm, auto select.*

Introduction

The development of research in GRID-technology in recent times is growing stronger [1 – 4]. This is closely related, primarily, the needs of a variety of scientific and technological tasks, for example:

– biologists want to simulate thousands of molecules – candidates for the drugs to see how these molecules will interact with certain proteins;

– high energy physics will soon annually about 10 Pb data. Thousands of physicists in many universities in the world will want to analyze these data;

– scientists who is study the planet, monitoring the level of atmospheric ozone, using data from satellites. To solve only one of the tasks they transfer from space to Earth a day, about 100 Gb;

– solution of mysteries of the human genome would be impossible without a computer analysis: DNA sequence consists of three billion chemical elements [5].

Therefore, create a need to have a powerful computing system, which would help to solve these and many other issues that necessary to develop a variety of industries. Such a system is the system of GRID, in particular UkrGRID. To solve any problem of applied industry, this task should be presented in the form of software code available to solve with computer technology. That is the primary task of applications is becoming a collection of tasks represented by software code, each of which is done of some time.

To work effectively the system must be considered that, on what the computer will be realize each of the tasks that will come to the system, it is necessary to solve the problem of efficient distributing of tasks among the computer of system.

1. Statement or problem and its solution

To further consideration the problem of distributing of tasks in the computer system is concerned, about what types of the system involved [6]. Consider the type of the system, if computers and tasks are different by capability. Then it would be logical for the most powerful computer to distribute the longest task. But as the number of computers and the number of tasks are differently, the task should be considered in detail from the viewpoint of the fact that there are two different cases:

1. The total number of tasks is less or equal than to the number of computers in the system;

2. The total number of tasks is much higher than the number of computers in the system. Then, all tasks are divided into streams, each of which subtasks (later – the task) is the same length of implementation.

Consider more particularly the first case.

Suppose n tasks of solutions time C_i and m computers that must solve these tasks. Each of m computers has its powers C_j.

Since then $n \le m$, it would be logical to choose a computer to solve the task is directly proportional. So, it is necessary that the more long-term task was given more powerful machine. Thus, the task is reduced to the sorting of array of task's time and of array of computer's power in the computer system. Then, working with the already sorted arrays, sending the first task of the array of task's time in the first computer, which is present in array of computer's power in the computer system.

This approach to solve the problem of optimal distribution of tasks makes important immediately algorithm of data sorting. It was optimization of sorting allows much to reduce the total time of all the tasks that have been in the system.

Thus, for the solution of the tasks mentioned above, it is necessary to find the optimal of the algorithm of data sorting.

Applying such an optimal sorting algorithm, the problem can be solved.

2. Auto select of sorting algorithm

Auto select of sorting algorithm was taken among algorithms by choice, by bubble, by Shaker, simple inserts, by Shell, pyramidal, quick [7 – 8].

Note that the functions implemented in Java in two ways: for arrays of type *int* and arrays of classes that implement the interface *Comparable*.

For describe of array were elected to two criteria: the size of the array and the «primary selection». These two parameters are determined by other more easily and at the same time have the most significant impact on the time of the sorting algorithms. Primary selection, in this case is defined as the ratio of the number of large, than those who were before them, the total number of elements. To save time, was realize the function, which sort out, not all elements, but only those that meet with sort out by a given step.

The graphs on Fig. 1 – 7 are showed effect of array's size to the speed of a variety of sorting. Also on thus diagrams, it was noted time of the definition of the parameters of array (curves test1 and test2).

Fig. 1 shows that the bubble-algorithm and algorithm by Shaker-sorting, which is a modification of the bubble (bubbleSort and hashingSort) are much slower than others. So they may only be used if needed natural behavior. Good results show sorting by simple insertions (insertSort) and sorting by Shell (shellSort). Fig. 2 can also see the advantage of sorting by Shell.

On Fig. 3 you can see that in arrays with size less than half of million the good results show the quick-sort and Shell-Sort.

On Fig. 4 you can see a sharp increase in run-time quick-sort. This is related with overheads on recursion, therefore the option was implemented quick-sorting, which realize software recursion.

It also built the same graph for pre-ordered array (Fig. 5).

On Fig. 5-7 you can see leadership of sorting by simple inserts.

After examining the sorting time of different ranges in several ways sorting, made the following conclusions.

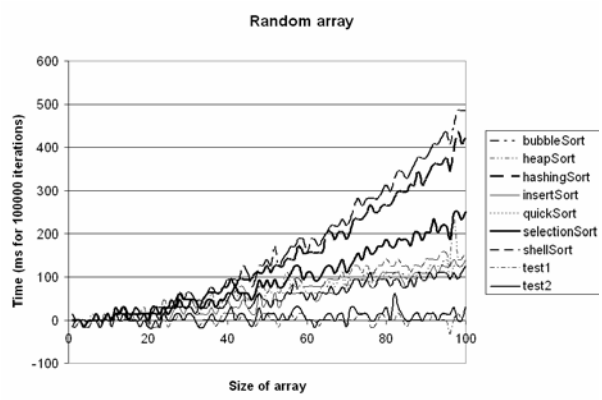

Fig. 1. Sorting time of small (up to 100 elements) random array

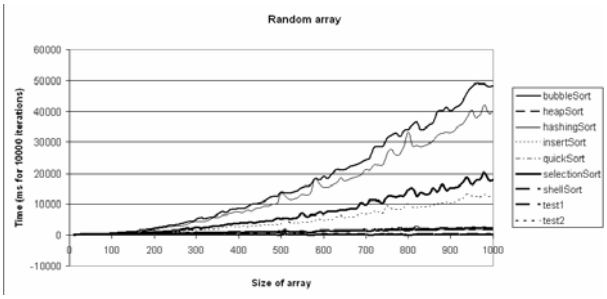

Fig.2. Sorting time of small (up to 1000 elements) random array

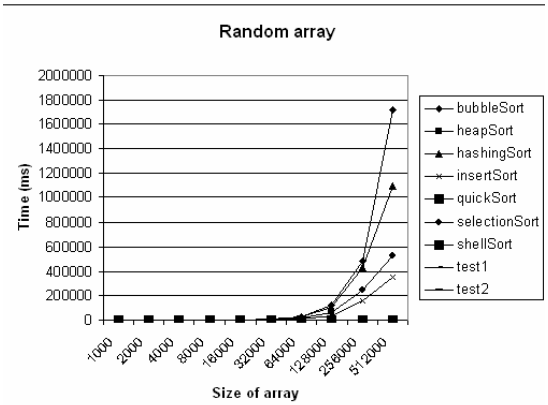

Fig. 3. Sorting time of average (up to 512,000 elements) random array

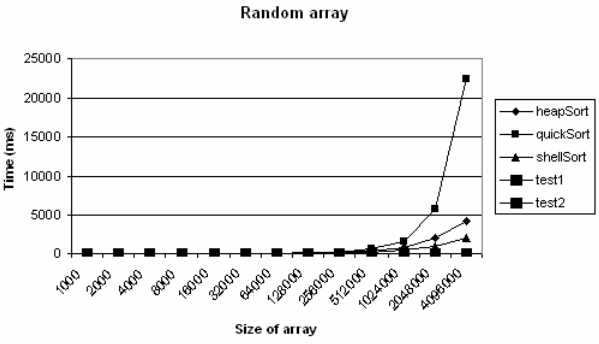

Fig.4. Sorting time of large (up to 4000000 elements) random array

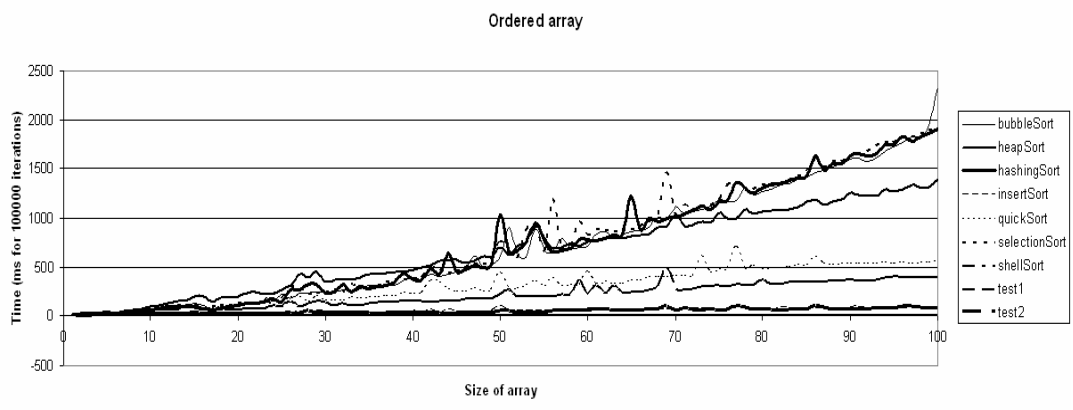

Fig. 5. Sorting time of small (up to 100 elements) ordered array

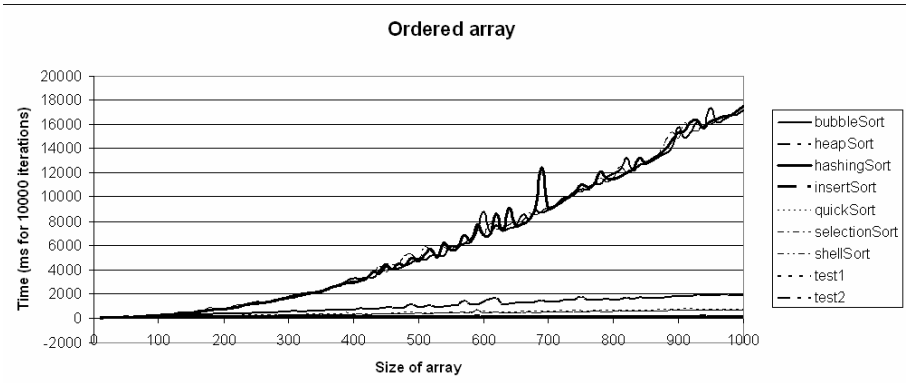

Fig. 6. Sorting time of small (up to 1000 elements) ordered array

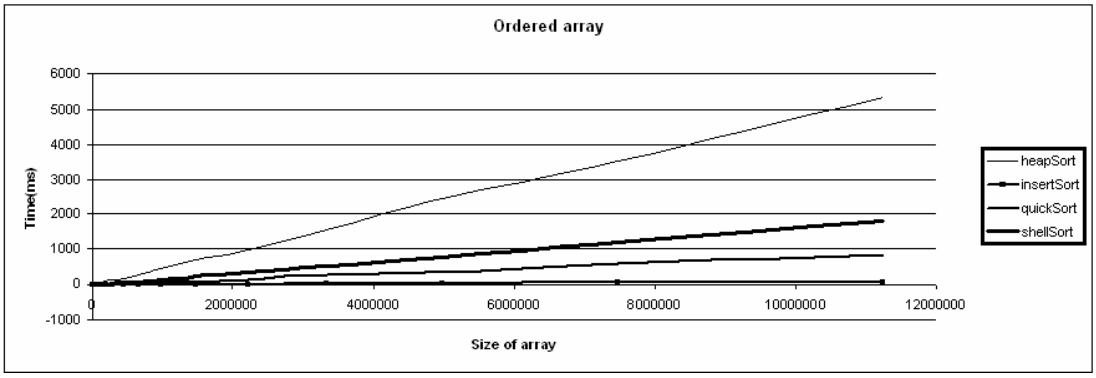

Fig. 7. Sorting time of large (up to 12000000 elements) ordered array

From the viewpoint of minimizing the time has the best sorting of arrays of random elements is the following sort:

– up to 50 elements – sort by simple inserts;

– up to 65536 elements – sort by Shell;

– more elements – a quick-sorting without using recursion (software implementation).

For almost selected array is better to use sorting by simple inserts.

3. Analysis of results of auto select of sorting algorithm

To demonstrate the achieved results measure time that spent on sorting through direct calling functions, and compare it with the time that spent on the call to Autoselect of sorting.

On Fig. 8-11 is built time of sorting dependence on array's size for a direct call function and for function Autoselect of sorting. As you can see from the Fig. 8 quick algorithms that do not have the properties of the natural, yield in the speed to algorithms, which it has. Auto select function that takes into account the property of the previous selection of array shows good results.

Small arrays are sorting very quickly. So, to measure the time of sorting, it is necessary to sort out a few arrays (in the case of Fig. 9 – 1000 iterations). In doing so, we generate an array of random elements in the same cycle, therefore their generation time

is added to the time of sorting. To take into account, we measured the time, only the generation array 1000 and subtract it from the sort of time. In view of the fact that the timer did not accurately measure short intervals (up to 30 ms), results of measurements of time sorting near zero received with great inaccuracy. Because of this, and through the small spacing between the points of measurement graph, first, is «waves», and secondly, passes through the negative values of time sorting. However, in Fig. 9 you can see that the functions Autoselect less than most other algorithms.

On Fig. 10 you can see that the sorting and selecting inserts on arrays of such size are very slow in comparison with other algorithms, and so build a graph without them.

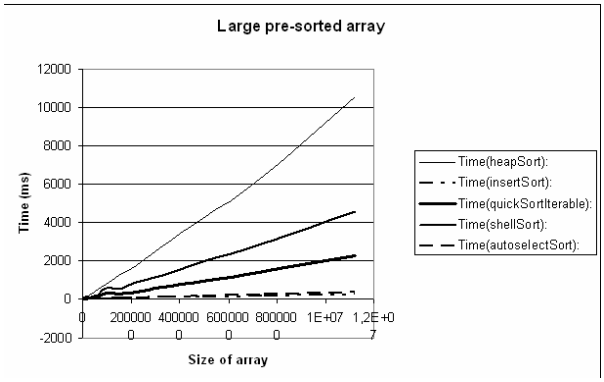

Fig. 8. Sorting time of pre-sorted array

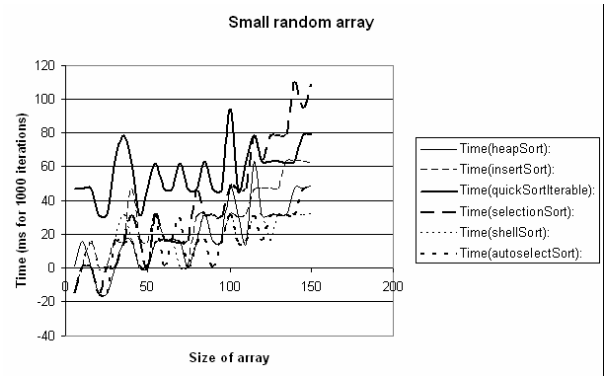

Fig. 9. Sorting time of small (up to 200 elements) random array

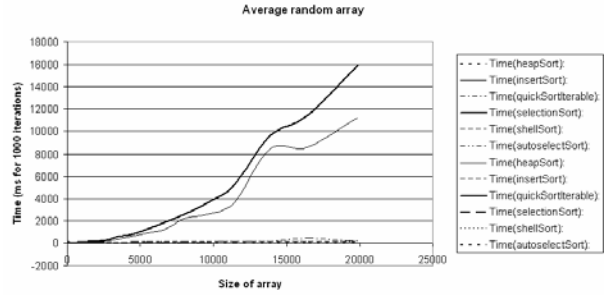

Fig. 10. Sorting time of average (up to 25,000 elements) random array

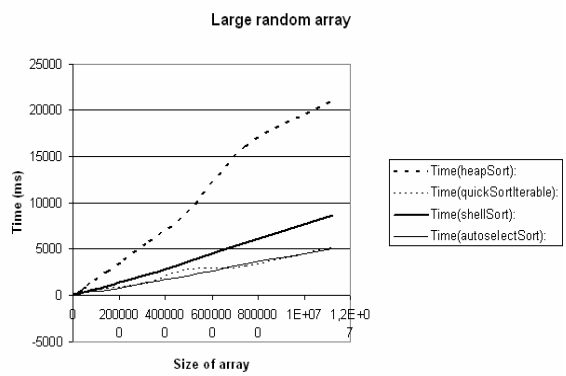

Fig. 11. Sorting time of large (up to 12000000 elements) random array

On Fig. 11 you can see, as does the work of various sorting algorithms. Autoselect shows only a small delay on the selection of algorithm, and the significant gains over time in comparison with other algorithms.

The graph noticeable that when the time sorting using Autoselect (t1) is not much more time (t2), the implementation of the algorithm, which shows the best results on some interval, then, in most cases, t1 and t2 is less than at the next interval. This demonstrates the success of the chosen approach: the selection time of algorithm insignificant effect on the total time sorting.

Conclusions

This paper demonstrates auto select of sorting algorithm for problem of the distribution of tasks in the computer system. The algorithm reduces the time of sorting of input data, and so the time distribution of tasks in the computer system. Developed library can be used not only in take up case, but in many others cases, where there is a need for speeding-up solving of tasks of data sorting, as well as save time and computer resources.

References

1. Корнеев В.В. Развитие инфраструктуры суперкомпьютерных вычислений МСЦ РАН на базе ГРИД-технологий / В.В. Корнеев // Труды 8-й Международной конференции «Высокопроизводительные параллельные вычисления на кластерных системах», 17-19 ноября 2008. – Казань: КГТУ, 2008. – С. 53-56

2. Интернет-портал по ГРИД-технологиям GRIDCLUB_RU – Grid for Windows & Unix [Электронный ресурс]. – Режим доступа к ресурсу: http://gridclub.ru/projects.

3. Каменщиков М.А. Сервисы GRID, как объекты стандартизации / М.А. Каменщиков // Электронный журнал «Журнал радиоэлектроники». – 2003. – №12 [Электронный ресурс]. – Режим доступа к журналу: http:// jre.cplire.ru.

4. Рузайкин Г.И. Философия ЦОД – Мир ПК / Г.И. Рузайкин // Открытые системы. – 2008 . – № 11

[Электрон. ресурс]. – Режим доступа к ресурсу: http://www.osp.ru/pcworld/2008/11/5688007/

5. Интернет-портал по ГРИД-технологиям GRIDCLUB_RU [Электрон. ресурс]. – Режим доступа к ресурсу: http://www.gridclub.ru/gridcafe/dreamers.html

6. Юрич М.Ю. Подход к оптимальному распределению заданий в вычислительной системе / М.Ю. Юрич // Комп'ютинг. – Тернопіль: Економічна думка, 2008. – Т. 7. – Вип. 1. – С. 27-34.

7. Кнут Дональд Э. Искусство программирования. Сортировка и поиск. / Дональд Э. Кнут. – М.: Издательский дом «Вильямс», 2008. – Т. 3. – 824 с.

8. Информационная интернет – база алгоритмов и методов [Электрон. ресурс]. – Режим доступа к ресурсу: http://algolist.manual.ru/sort/index.php

Поступила в редакцию 2.02.2009

Рецензент: д-р. физ.-мат. наук, проф., зав. кафедрой Г.В. Корнич, Запорожский национальный технический университет, Запорожье, Украина.

АЛГОРИТМЫ СОРТИРОВКИ ДЛЯ РАСПРЕДЕЛЕНИЯ ЗАДАНИЙ В ВЫЧИСЛИТЕЛЬНОЙ СИСТЕМЕ

М.Ю. Юрич, Д.С. Барсуков, Р.К. Кудерметов

Рассмотрена проблема распределения заданий в вычислительной системе для случая, когда количество компьютеров системы превышает количество заданий, которые необходимо распределить. Для этого применены различные алгоритмы сортировки. С целью повышения скорости работы рассмотренного механизма распределения заданий на основе использования алгоритмов сортировки предложен алгоритм автоматического выбора сортировки в зависимости от свойств данных, представленных входными массивами. Показано, что принятое решение проблемы является эффективным, т.к. время, потраченное на принятие решения о выборе алгоритма, несущественно влияет на общее время сортировки. Доказано, что разработанный алгоритм позволяет сократить время сортировки входных данных, а значит и время решения проблемы распределения заданий в вычислительной системе.

Ключевые слова: распределение заданий, вычислительная система, сортировка, время сортировки, алгоритмы сортировки, авто вибор.

АЛГОРИТМИ СОРТУВАННЯ ДЛЯ РОЗПОДІЛУ ЗАВДАНЬ В ОБЧИСЛЮВАЛЬНІЙ СИСТЕМІ

М.Ю. Юрич, Д.С. Барсуков, Р.К. Кудерметов

Розглянуто проблему розподілення завдань в обчислювальній системі для випадку, коли кількість комп'ютерів системи перевищує кількість завдань, які необхідно розподілити. Для цього застосовано різноманітні алгоритми сортування. З метою підвищення швидкості роботи розглянутого механізму розподілу завдань на базі використання алгоритмів сортування запропоновано алгоритм автоматичного вибору сортування у залежності від властивостей даних, що представлені вхідними масивами. Показано, що обране розв'язання проблеми є ефективним, бо час, витрачений на прийняття рішення про вибір алгоритму, несуттєво впливає на загальний час сортування. Доведено, що розроблений алгоритм дозволяє скоротити час сортування вхідних даних, а значить і час розв'язання проблеми розподілу завдань в обчислювальній системі.

Ключові слова: розподіл завдань, обчислювальна система, сортування, час сортування, алгоритми сортування, авто вибір.

Кудерметов Равиль Камилович – канд. техн. наук, доцент, заведующий кафедрой «Компьютерные системы и сети», Запорожский национальный технический университет, Запорожье, Украина, e-mail: krk@zntu.edu.ua.

Юрич Мария Юрьевна – аспирант, ассистент кафедры «Компьютерные системы и сети» Запорожский национальный технический университет, Запорожье, Украина, e-mail: masheryka@mail.ru; mashery@zntu.edu.ua.

Барсуков Дмитрий Сергеевич – студент кафедры «Компьютерные системы и сети» Запорожский национальный технический университет, Запорожье, Украина.