Bulletin of Networking, Computing, Systems, and Software – www.bncss.org, ISSN 2186–5140
Volume 1, Number 1, pages 13–15, January 2012

Evaluating Memory Size for Energy Efficient Sorting

Cheng-Jen Tang, Hui-Chin He
and Miau-Ru Dai
Graduate Institute of Communication Engineering
Tatung University
Taipei, Taiwan 104
Email: ctang@ttu.edu.tw, hche@ttu.edu.tw
d9610002@ms2.ttu.edu.tw

Chi-Cheng Chuang
Smart Network System Institute
Institute for Information Industry
Taipei, Taiwan 105
Email: polon@nmi.iii.org.tw

***Abstract*—Sorting processes are responsible for a large portion of energy consumed by computers. This paper proposes an energy efficiency evaluation for the required memory size of a sorting algorithm. An experiment result shows the energy consumption of different kinds of sorting algorithms with different memory sizes and various input sizes. This study then determines the memory size of a sorting algorithm for achieving better energy efficiency.**

I. Introduction

In order to make the best use of precious electricity resource, the essential step is to know how much energy will be required under a certain circumstance. To effectively evaluate the energy efficiency of a computing process needs to consider the energy usage conditions on many issues. Software solutions avoid a lot of hardware investments, which make them very attracting. A work that targets at the energy efficiency issues of sorting algorithms[1] inspires this study, since sorting has been considered as the foundation of many other algorithms[2], and occupied a lot of CPU cycles[3]. If there is an approach capable of dimming energy required for sorting in a computer, some energy saving is expectable. Through the conducted experiment, this study finds that memory size of a computer has a great impact on energy efficiency. Therefore, this paper presents a memory size evaluation for energy efficient sorting.

II. Related Work

Besides designing a new energy-efficient hardware, some researches attack the energy efficiency issue regarding computation from the following perspectives:

- Finds the most energy-efficient algorithm by comparing different algorithms, such as Bunse et al.[1]
- Makes compilers to generate energy-efficient codes or use a energy-efficient library, such as Zhong et al.[4], Ayala et al.[5], and Segmund et al.[6]

Bunse et al.[1] define a set of trend functions that chooses a sorting algorithm according to the given conditions. In their work, bubblesort, heapsort, insertionsort, mergesort, quicksort, selectionsort, shakersort, and shellsort are evaluated. Insertionsort is identified as the most energy-efficient sorting algorithm in this work, if the number of input items is large enough.

TABLE I
Hardware Specification

Hardware	Sorting Machine	Control Recording Machine	Network Arrached Storage(NAS)
Processor	AMD Athlon 64, 1GHz	IntelR CeleronR, 2.66GHz	AMD Athlon 64, 1GHz
Memory	SAMSUNG 512MB DDR	SAMSUNG 512MB DDR	SAMSUNG 512MB DDR

Zhong et al.[4] propose AcovSA (Analysis of Compiler Options via Simulated Annealing) that can find a good set of compiler options for a particular CPU and software. Although this tool is not particularly designed for optimizing energy usage, it is helpful for finding a set of compiler options that produces an executable image consuming less energy than other sets. Ayala et al.[5] tune the settings of register file with some code profiles. The main challenge of adopting such mechanism is the necessity of modifying ISA (Instruction Set Architecture). Segmund et al.[6] propose an energy feature library that is developed with many energy-saving techniques. These shared object libraries replace applications code with the code of the library or (de)activate the necessary hardware components.

III. Experiment Setup

The study conducts an experiment that measures consumed time and current of computers sorting different data sets using different memory sizes. The surveyed sorting algorithms include Bubble Sort, Insertion Sort, Selection Sort, Merge Sort, Recursive Quick Sort, and Non-recursive Quick Sort. The mounted memory sizes in the testing computers are 500 MB, 1 GB, 1.5 GB, and 2GB. Each experiment set-up consists of a digital multimeter, Kaito M9803R, and three computers as shown in Fig. 1. Three devices are named as **Current Measure Machine**, **Sorting Machine**, **Control-Recording Machine**, and **Network Attached Storage** according to their functions. Table I shows the hardware specifications in the experiment. The following describes the features of the used devices.

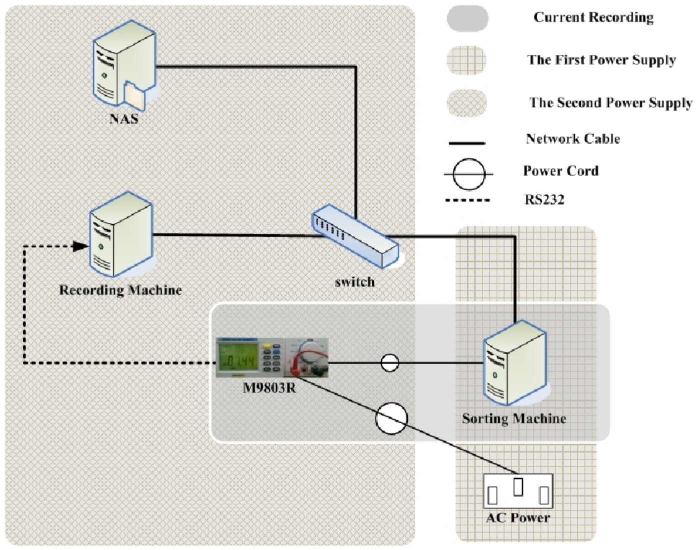

Fig. 1. Experiment Setup

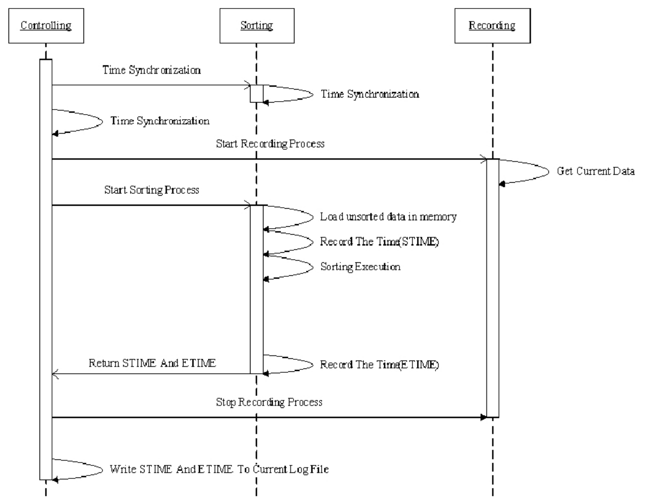

Fig. 2. Experiment Flow

- **Current Measure Machine** is Kaito M9803R bench digital multimeter. It generates four records per second, and transmits to the recording machine through *RS-232*. According to *Kirchhoff's Current Law*, the meter connects with the sorting system in series for reading current values.
- **Sorting Machine** is responsible for performing sorting processes. The consumed power is measured by **Current Measure Machine**.
- **Control-Recording Machine** preforms two tasks. The first is controlling the operations of the whole experiment. The second is recording the measured values from **Current Measure Machine**.
- **Network Attached Storage** stores the input data sets. These data sets are sorted in **Sorting Machine**.

The flow of this experiment is shown in Fig. 2. This experiment tests 49 different input sizes that are $1K$ to $10K$, and $10M$ to $490M$. Each number in a data set is 4 byte long. The numbers in a data set are reset for each test and re-generated randomly. For a sorting algorithm, each input size is tested 10 times with a randomly generated data set for each test.

IV. Experiment Result and Analysis

The collected records are analyzed from two perspectives. One is to see how the algorithm itself affects the energy usage. The other inspects the impact of memory size on energy usage.

Just as expected, the time complexity of sorting algorithms does affect the energy usage greatly. It is easy to see that algorithms with time complexity $O(n^2)$, as shown in Fig. 4, consume a lot more energy than algorithms with time complexity $O(n \log n)$, as shown in Fig. 3. Since $Energy = Power \times Time$, it is obvious that algorithms with less time complexity consume less energy. Surprisingly and interestingly, Bunse et al. [7] indicates that the result does not always stand in an environment with a very limited memory capacity. The question is: how does memory size affect the energy usage of a sorting algorithm? Fig. 5 shows the comparison of energy efficient between **Merge Sort**, **Recursive Quick Sort**, and **Non-recursive Quick Sort** running on the **Sorting Machine** with 1.5 GB memory. The energy efficiency in this study is defined as:

$$EE = \frac{N}{E} \tag{1}$$

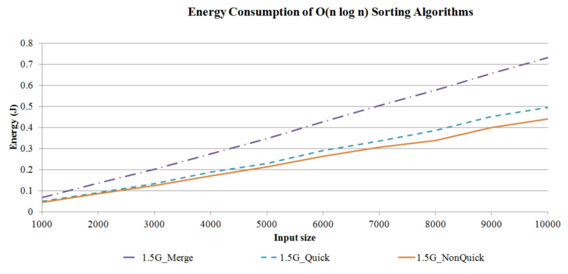

Fig. 3. Energy Consumption of Merge Sort, Recursive Quick Sort, and Non-recursive Quick Sort

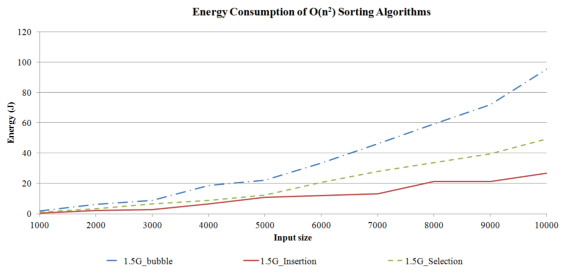

Fig. 4. Energy Consumption of Bubble Sort, Insertion Sort, and Selection Sort

EE is the energy efficiency of a test. N is the number of items being sorted in the test. E is the measured energy consumption , in $Joule$, of the test. The EE of both **Non-recursive Quick Sort** and **Recursive Quick Sort** fall dramatically with input size of 370 M (3.7×10^8). That is because that the size of input data set ($3.7\times10^8\times4 = 1.48\times10^9 \approx 1.5G$) exceeds the size of available physical memory. The reason of **Merge Sort** falling much earlier is that **Merge Sort** needs more memory for sorting data. When the required memory space exceeds the available physical memory, modern operating systems activate *Swapping*, which moves data to a certain area(s), so called *Swap Space*, on Hard Disk(s).

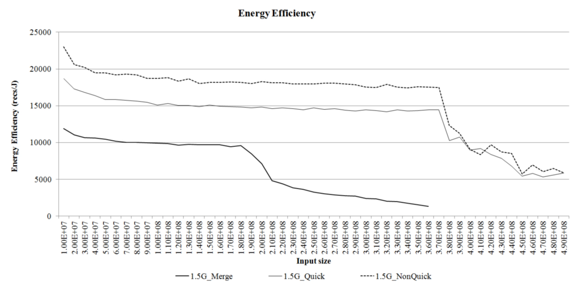

Fig. 5. Energy Efficiency of Merge Sort, Recursive Quick Sort, and Non-recursive Quick Sort

Therefore, the difference between the size of physical memory and the size of the input data is the deciding factor on the EE of a sorting algorithm. If there is enough memory, EE of a sorting algorithm is close to a constant when the input size is large enough, as shown in Fig. 6.

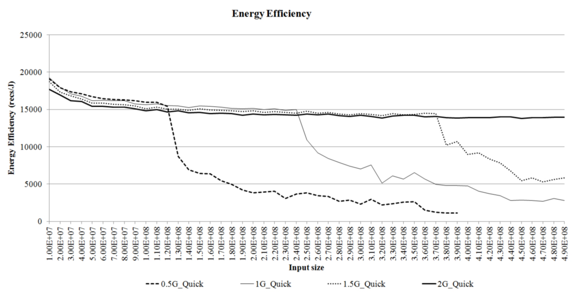

Fig. 6. Energy Efficiency of Recursive Quick Sort Running with 0.5 GB, 1 GB, 1.5 GB and 2 GB memory

Although larger memory size seems to have a better EE with a large input size, memory itself consumes a notable amount of energy. Fig. **??** shows that every 0.5 GB increases 0.025 Amperes that is about 2.75 Watts. On the other hand, this causes a lot of energy wasting if the system is not always busy on sorting things.

V. Conclusion

This study finds that:

- The time complexity of sorting algorithms affects greatly on the energy usage. For example, algorithms with time complexity $O(n^2)$ consume a lot more energy than algorithms with time complexity $O(n \log n)$.

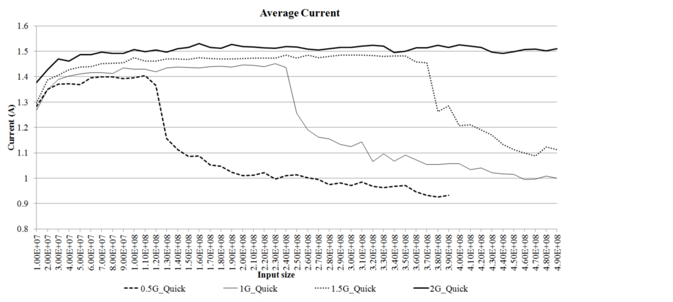

Fig. 7. Current of Recursive Quick Sort Running with 0.5 GB, 1 GB, 1.5 GB and 2 GB memory

- For a sorting algorithm, when the size of the input data exceeds the size of available physical memory, its energy efficiency drops dramatically.
- If there is enough memory, EE of a sorting algorithm is close to a constant when the input size is large enough.

Although larger memory size seems to have a better EE, a system with huge amount of memory causes a lot of energy wasting. Therefore, this situation brings up several research topics worth for further investigation, such as: balancing of EE and the required memory size, runtime detachable memory module, etc.

Acknowledgment

This study is conducted under the ”Advanced Metering Infrastructure(AMI) Enhancement Project” of the Institute for Information Industry which is subsidized by the Ministry of Economy Affairs of the Republic of China.

References

[1] C. Bunse, H. Höpfner, S. Roychoudhury, and E. Mansour, “Choosing the best sorting algorithm for optimal energy consumption,?” in *Proceedings of the International Conference on Software and Data Technologies (ICSOFT)*, 2009, pp. 199–206.

[2] S. S. Skiena, *The Algorithm Design Manual*, 2nd ed. pub-SV:adr: Springer-Verlag, 2008.

[3] D. E. Knuth, *The Art of Computer Programming, Vol 3, Sorting and Searching*, 2nd ed. Reading, USA: Addison-Wesley, 1998.

[4] S. Zhong, Y. Shen, and F. Hao, “Tuning compiler optimization options via simulated annealing,” in *Future Information Technology and Management Engineering, 2009. FITME ’09. Second International Conference on*, 2009, pp. 305 –308.

[5] J. L. Ayala, A. Veidenbaum, and M. Lpez-Vallejo, “Power-aware compilation for register file energy reduction,” *International Journal of Parallel Programming*, vol. 31, pp. 451–467, 2003, 10.1023/B:IJPP.0000004510.66751.2e. [Online]. Available: http://dx.doi.org/10.1023/B:IJPP.0000004510.66751.2e

[6] N. Siegmund, M. Rosenmüller, and S. Apel, “Automating energy optimization with features,” in *Proceedings of the 2nd International Workshop on Feature-Oriented Software Development*, ser. FOSD ’10. New York, NY, USA: ACM, 2010, pp. 2–9. [Online]. Available: http://doi.acm.org/10.1145/1868688.1868690

[7] C. Bunse, H. Höpfner, S. Roychoudhury, and E. Mansour, “Energy efficient data sorting using standard sorting algorithms,” *Software and Data Technologies*, pp. 247–260, 2011.