Distributed prime sieve in heterogeneous computer clusters

Carlos Costa

Faculdade de Engenharia da Universidade do Porto

Rua Dr. Roberto Frias, s/n 4200-465 Porto, Portugal

[carlos.costa@fe.up.pt](mailto:carlos.costa@fe.up.pt) / carloscosta.cmcc@gmail.com

**Abstract.** Prime numbers play a pivotal role in current encryption algorithms and given the rise of cloud computing, the need for larger primes never been so high. This increase in available computation power can be used to either break the encryption or to strength it by finding larger primes. With this in mind, this paper provides an analysis of different sieves implementations that can be used to generate primes up to a given number. It starts by analyzing cache friendly sequential sieves with wheel factorization, then expands to multicore architectures using OpenMP and ends with a segmented hybrid implementation of a distributed prime sieve using OpenMP and OpenMPI, designed to efficiently use all the available computation resources of heterogeneous computer clusters with variable workload.

**Keywords:** distributed prime sieve, sieve of Eratosthenes, wheel factorization, distributed and parallel computing, OpenMP, OpenMPI, C++

1. Introduction

Prime numbers have been a topic of wide research given their important role in securing online transactions using asymmetric public key algorithms such as RSA [1]. In this context prime numbers are used to create the public key that is used by the sender to encrypt the message contents. The reason to use a product of two prime numbers to create the public key is because in the current computer architecture, it is computation infeasible to factor large prime numbers, and thus find the original primes used to create the public key, and used then to decrypt the message (with the progress made in quantum computers, this may not hold for very long). Besides encryption, prime numbers can also be used in hash functions to reduce the risk of collision and in pseudorandom number generators.

Although nowadays is more common to use primality test algorithms [2] to find large primes, sieves have been a known method to generate primes up to a given number. One of the most efficient prime number sieve was discovered by Eratosthenes [3] in the ancient Greece and can generate prime numbers up to a given number n with O(n log log n) operations. Other prime sieves were discovered since then, such as the sieve of Atkin [4] or the sieve of Sundaram [5], but a modified sieve of Eratosthenes is still considered to be the most efficient algorithm to use in the current computer architecture, and thus it was the one chosen to be used. This algorithm was implemented with 20 different variations, ranging from sequential form, passing to parallel on multicore architectures and ending in distributed computer clusters. Each of these 3 main implementations have several algorithm variations, to determine which strategy was more suitable for each usage. As such, it was developed algorithms variations that focused on using the minimum amount of memory, others focused on computing time, and others made a tradeoff between the two. Finally it was implemented segmented versions of both parallel and distributed algorithms to allow the computation of primes to the maximum number represented by the current computer architecture (2^64). Special attention was devoted to the implementation of the distributed algorithm version since is the one more suitable to be used to calculate primes up to 2^64 in reasonable time, because it was designed to scale very well to large clusters and was implemented to perform dynamic allocation of segments to nodes in order to efficiently use all the computation capacity of clusters with heterogeneous hardware and with variable workload.

On the next sections it will be presented each algorithm variation and their possible application, and it will be discussed the implementation results along with the speedup, efficiency and scalability analysis of the 3 main algorithms implementations.

1. Related work

In the course of the implementation of the several algorithm variations, it was performed a state of the art search to identify possible design improvements and to determined which implementation should be optimized in order to contributed to the already existing implementations publicly available.

From all the publicly implementations analyzed, special interest was devoted to the Prime Sieve [6] developed by Kim Walisch, since it is considered to be the fastest multicore implementation publicly available at the present date.

Other papers that influenced the strategies developed included [7] that details a very efficient use of the cache memory system for very large prime numbers, [8] [9] that explains how to use wheel factorization to speed up considerably the sieving process and [10] [11] that provide insights on how to implement the simple MPI version.

From this search was determined that at the present date, a distributed implementation optimized for heterogeneous clusters could be of public interest and as such, was the implementation that was devoted most of work in development.

### **Headings.** This is an example of a sub-sub-section. Headings should be capitalized (i.e., nouns, verbs, and all other words except articles, prepositions, and conjunctions should be set with an initial capital) and should, with the exception of the title, be aligned to the left. The font sizes are given in Table 1.

Here are some examples of headings: "Criteria to Disprove Context-Freeness of Collage Languages", "On Correcting the Intrusion of Tracing Non-deterministic Programs by Software", "A User-Friendly and Extendable Data Distribution System", "Multi-flip Networks: Parallelizing GenSAT", "Self-determinations of Man".

**Table 1.** Font sizes of headings. Table captions should always be positioned above the tables. This table provides the format that must be used for all tables. The font size of table text is 9pt. and left justified

|  |  |  |
| --- | --- | --- |
| Heading level | Example | Font size and style |
| Title (centered) | **Lecture Notes …** | 14 point, bold |
| 1st-level heading | **1 Introduction** | 12 point, bold |
| 2nd-level heading | **2.1 Printing Area** | 10 point, bold |
| 3rd-level heading | **Headings.** Text follows … | 10 point, bold |
| 4th-level heading | *Remark.* Text follows … | 10 point, italic |
| Table text | Example table text | 9 point |

Lemmas, Propositions, and Theorems. The numbers accorded to lemmas, propositions, and theorems, etc. should appear in consecutive order, starting with Lemma 1, and not, for example, with Lemma 11.

Here is an example of a bullet list:

* Papers not complying with the LNCS style will be reformatted. This can lead to an increase in the overall number of pages. We would therefore urge you not to squash your paper.
* Another item.

Here is an example of a numbered list:

1. Item 1.
2. Item 2.
   1. Sub-item 1.
   2. Sub-item 2.
   3. Figures

Please check that the lines in line drawings are not interrupted and have a constant width.

To ensure that the reproduction of your illustrations is of a reasonable quality, we advise against the use of shading. The contrast should be as pronounced as possible.



**Fig. 1.** This is an example of a figure caption. Figure captions are positioned below the figure..

* 1. Formulas

Displayed equations or formulas are centered and set on a separate line (with an extra line or halfline space above and below). Displayed expressions should be numbered for reference. The numbers should be consecutive within each section or within the contribution, with numbers enclosed in parentheses and set on the right margin.

|  |  |
| --- | --- |
| x + y = z . | (**1**) () |

Equations should be punctuated in the same way as ordinary text but with a small space before the end punctuation mark.

* 1. Footnotes

The superscript numeral used to refer to a footnote appears in the text either directly after the word to be discussed or – in relation to a phrase or a sentence – following the punctuation mark (comma, semicolon, or period). Footnotes should appear at the bottom of the normal text area, with a line of about 5cm set immediately above them[[1]](#footnote-1).

* 1. Program Code

Program listings or program commands in the text are normally set in typewriter font, e.g., CMTT10 or Courier.

program Inflation (Output)  
 {Assuming annual inflation rates of 7%, 8%, and  
 10%,... years};  
 const MaxYears = 10;  
 var Year: 0..MaxYears;  
 Factor1, Factor2, Factor3: Real;  
 begin  
 Year := 0;  
 Factor1 := 1.0; Factor2 := 1.0; Factor3 := 1.0;  
 WriteLn('Year 7% 8% 10%'); WriteLn;  
 repeat  
 Year := Year + 1;  
 Factor1 := Factor1 \* 1.07;  
 Factor2 := Factor2 \* 1.08;  
 Factor3 := Factor3 \* 1.10;  
 WriteLn(Year:5,Factor1:7:3,Factor2:7:3,  
 Factor3:7:3)  
 until Year = MaxYears  
end.

* 1. Page Numbering and Running Heads

There is no need to include page numbers. If your paper title is too long to serve as a running head, it will be shortened. Your suggestion as to how to shorten it would be most welcome.

References

1. RSA Laboratories: PKCS #1 v2.2 - RSA Cryptography Standard (2012)
2. Wikipedia: Primality test, <http://en.wikipedia.org/wiki/Primality_test>, [Online, accessed 24-07-2013]
3. Wikipedia: Sieve of Eratosthenes, <http://en.wikipedia.org/wiki/Sieve_of_Eratosthenes>, [Online, accessed 24-07-2013]
4. Wikipedia: Sieve of Atkin, <http://en.wikipedia.org/wiki/Sieve_of_Atkin>, [Online, accessed 24-07-2013]
5. Wikipedia: Sieve of Sundaram, <http://en.wikipedia.org/wiki/Sieve_of_Sundaram>, [Online, accessed 24-07-2013]
6. Walisch, K.: Prime Sieve, <https://code.google.com/p/primesieve/>, [Online, accessed 24-07-2013]
7. Járai, A., Vatai, E.: Cache optimized linear sieve (2011)
8. Paillard,, G.: A fully distributed prime numbers generation using wheel sieve (2005)
9. Sorenson, J.: An Analysis of Two Prime Number Sieves (1991)
10. Wirian, D.: Parallel Prime Sieve – Finding prime numbers
11. Cordeiro, M.: Parallelization of the Sieve of Eratosthenes
12. May, P., Ehrlich, H.C., Steinke, T.: ZIB Structure Prediction Pipeline: Composing a Complex Biological Workflow through Web Services. In: Nagel, W.E., Walter, W.V., Lehner, W. (eds.) Euro-Par 2006. LNCS, vol. 4128, pp. 1148--1158. Springer, Heidelberg (2006)
13. Foster, I., Kesselman, C.: The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann, San Francisco (1999)
14. Czajkowski, K., Fitzgerald, S., Foster, I., Kesselman, C.: Grid Information Services for Distributed Resource Sharing. In: 10th IEEE International Symposium on High Performance Distributed Computing, pp. 181--184. IEEE Press, New York (2001)
15. Foster, I., Kesselman, C., Nick, J., Tuecke, S.: The Physiology of the Grid: an Open Grid Services Architecture for Distributed Systems Integration. Technical report, Global Grid Forum (2002)
16. National Center for Biotechnology Information, http://www.ncbi.nlm.nih.gov

Appendix: Springer-Author Discount

LNCS authors are entitled to a 33.3% discount off all Springer publications. Before placing an order, they should send an email to [SDC.bookorder@springer.com](mailto:SDC.bookorder@springer.com), giving full details of their Springer publication, to obtain a so-called token. This token is a number, which must be entered when placing an order via the Internet, in order to obtain the discount.

1. The footnote numeral is set flush left and the text follows with the usual word spacing. [↑](#footnote-ref-1)