— On the Factor Refinement Principle and its Implementation on Multicore Architectures —

Mohsin Ali, Marc Moreno Maza and Yuzhen Xie

ORCCA Ontario Research Centre for Computer Algebra

Western

Overview

The *factor refinement principle* turns a partial factorization of integers (or polynomials) into a more complete factorization represented by basis elements and exponents, where basis elements are pairwise coprime. There are many applications of this refinement technique such as simplifying polynomial systems and, more generally, to algebraic algorithms generating redundant expressions during intermediate computations.

In this work, we design parallel adaptations of factor refinement algorithms. We consider several optimization techniques such as data locality analysis, balancing subproblems, etc. to fully exploit multicore architectures. The `Cilk++` implementation of our parallel algorithm achieves linear speedup for input data of sufficiently large size.

Definitions

Let $n_1, n_2, \ldots, n_s$ be integers (or polynomials). We say that they form a GCD-free basis whenever $\gcd(n_i, n_j) = 1$ for all $1 \leq i < j \leq s$.

Let $m_1, m_2, \ldots, m_r$ be other elements of the same type as the n_i's and let m be the product of the m_j's. Let $e_1, e_2, \ldots, e_s$ be positive integers. We say that the pairs $(n_1, e_1), (n_2, e_2), \ldots, (n_s, e_s)$ form a refinement of $m_1, m_2, \ldots, m_r$ if the following conditions hold:

(i) $n_1, n_2, \ldots, n_s$ is a GCD-free basis,

(ii) for every $1 \leq i \leq r$ there exists non-negative integers $f_1, \ldots, f_s$ such that we have $\prod_{1 \leq j \leq s} n_j^{f_j} = m_i$,

When this holds, we also say that $(n_1, e_1), (n_2, e_2), \ldots, (n_s, e_s)$ is a coprime factorization of m. For instance, $5^1, 6^2, 7^1$ is a refinement of 30 and 42 while $5^1, 6^2, 7^1$ is a coprime factorization of 1260.

The contribution of Bach, Driscoll and Shallit

Given a partial factorization of an integer m, say $m = m_1 m_2$, we compute $d = \gcd(m_1, m_2)$ and write

$$m = (m_1/d)(d^2)(m_2/d).$$

If this process is continued until all the factors are pairwise coprime, one obtains a coprime factorization within $O(\mathrm{size}(m)^3)$ bit operations.

In their landmark 1988 paper, Bach, Driscoll and Shallit observed that, by keeping track of the pairs (n_j, n_k) in an ordered pair list such that only elements adjacent in the list can have a nontrivial GCD, one computes a coprime factorization within $O(\mathrm{size}(m)^2)$ bit operations.

In their proof, they simply rely on plain (or quadratic) arithmetic. Using asymptotically fast algorithms for the case of univariate polynomials, D. Bernstein on one hand and, Dahan, Moreno Maza, Schost, Xie on another, have obtained an estimate of $O(d \log_2^4(d) \log_2(\log_2(d)))$, where d is the sum of the degrees of the input polynomials.

A Divide and Conquer Approach

We have developed a divide and conquer algorithm illustrated hereafter.

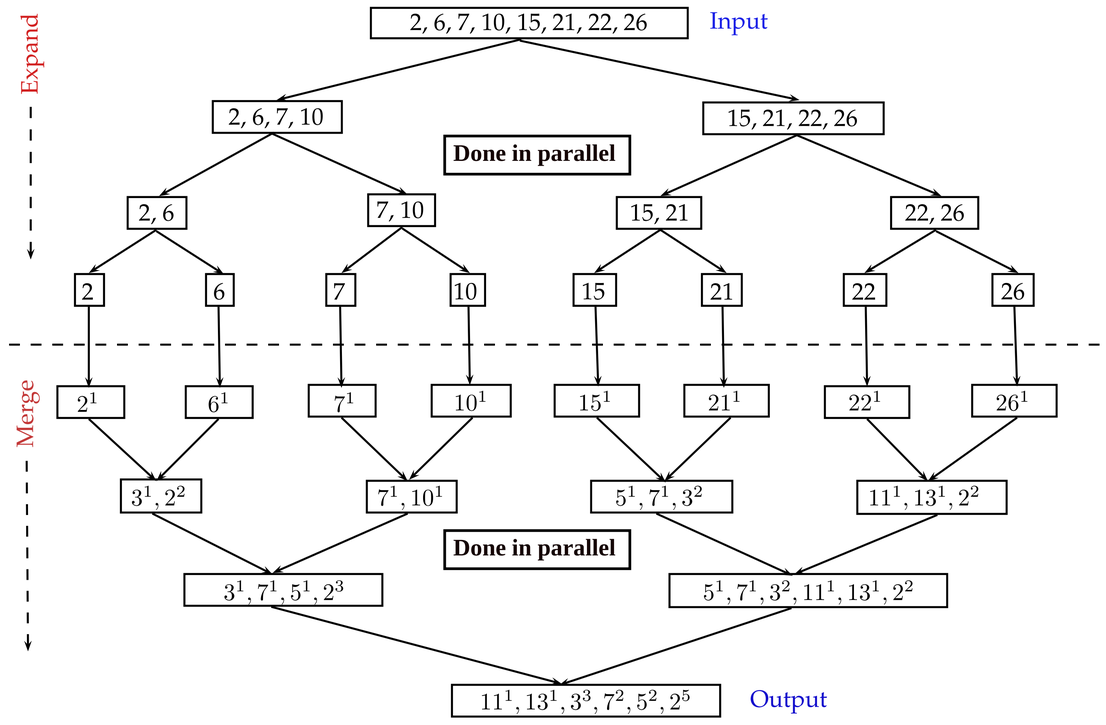


Figure 2: Example of algorithm execution.

On input data of size n, under the condition that each arithmetic operation (integer division and integer GCD computation) has a unit cost, this parallel algorithm features $O(n^2)$ work, $O(Cn)$ span, and thus $O(n/C)$ parallelism, where C is the size threshold below which serial code is run.

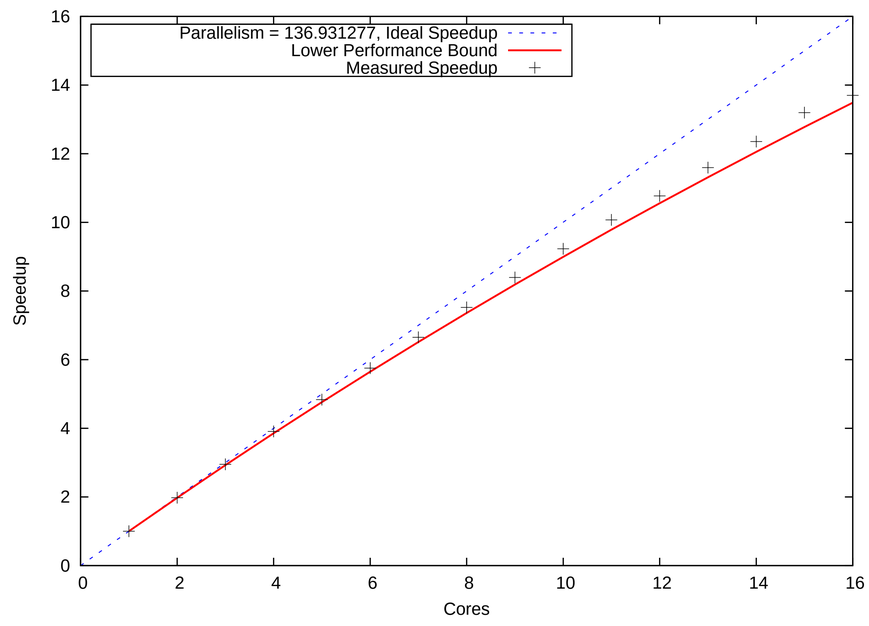
For an ideal cache of Z words, with L words per cache-line, with C small enough, for any input of size n, the number of cache misses is

$$Q(n) = O(n^2/ZL + n^2/Z^2),$$

under the assumption that each input or output sequence is packed, and each sequence item is stored in one machine word. These assumptions hold in our implementation for the case of univariate polynomials over a small prime field. Moreover, we enforce balancing of subproblems for optimizing the share of computer resources among threads.

Experimental Results

We have implemented our algorithm in `Cilk++` and tested successfully with both integers and polynomials. The figure below shows scalability results with an input data set of 5,000 polynomials of various degrees up to 200. Coefficients are taken modulo a machine word prime.



Acknowledgments

NSERC CRSNG

Maplesoft Mathematics • Modeling • Simulation

MITACS