Proceedings of the 16th Conference on Computer Science and Intelligence Systems pp. 283–290 DOI: 10.15439/2021F68 ISSN 2300-5963 ACSIS, Vol. 25

The impact of vectorization and parallelization of the slope algorithm on performance and energy efficiency on multi-core architecture

Beata Bylina, Joanna Potiopa, Michał Klisowski, Jarosław Bylina
Institute of Computer Science, Marie Curie-Sklodowska University
Pl. M. Curie-Skłodowskiej 5
Lublin, 20-031, Poland
Email: {beata.bylina, joanna.potiopa, michal.klisowski, jaroslaw.bylina}@umcs.pl

***Abstract*—Calculation of land-surface parameters (e.g. slope, aspect, curvature) is an important part of many geospatial analyses. Current research trends are aimed at developing new software techniques to achieve the best performance and energy trade-off. In our work, we concentrate on the vectorization and parallelization to improve overall energy efficiency and performance of the neighborhood raster algorithms for the computation of land-surface parameters. We chose the slope calculation algorithm as the basis for our investigation. The parallelization was achieved through redesigning the the original sequential code with OpenMP SIMD vectorization hints for compiler, OpenMP loop parallelization, and the hybrid of these techniques. To evaluate both performance and energy savings, we tested our vector-parallel implementations on a multi-core computer for various data sizes. RAPL interface was used to measure energy consumption. The results showed that optimization towards high performance can also be an effective strategy for improving energy efficiency.**

***Index Terms*—power, energy efficiency, RAPL, slope, multicore**

I. Introduction

WITH the growing demand for computing power new, more performant computer architectures have emerged. At the same time, the development of computer systems entails an increase in electricity consumption. One way to achieve the reduction of electricity consumption is modern, more energy-efficient hardware. It allows for high computing performance with lower energy consumption. Another way of achieving energy efficiency in high performance computing (HPC) is rethinking the software, including both runtime environments and applications themselves. Efforts are constantly being made to study the impact of algorithm optimizations on energy consumption [11], [6]. Yet, there are still many algorithms formulated in the past that now need to be rewritten to make effective use of modern computer architectures.

Examples are various geospatial analysis algorithms. Many of them are very time-consuming and does not scale well with large data sets, but are applied for increasingly large areas or at increasing resolution. Traditional GIS (geographic information system) software implements sequential algorithms that do not use efficiently computing power of modern computer architectures.

The study of the effect of terrain profile on hydrological, geomorphological, and ecological phenomena and processes start with the calculation of topographic parameters from the digital elevation model (DEM) [4]. The slope is one of the basic primary parameters and it is used e.g. for computing flow velocity for both overland and channelized flow. Other parameters (like soil erosion and deposition, soil wetness, flow speed) are calculated based on the slope. The slope calculation algorithm plays a fundamental role in more advanced models, although it is only one of the input elements for advanced computational algorithms (like modeling rates of snowmelt and evapotranspiration) [10]. Therefore, the high-performance slope calculation algorithm in such complex geospatial analyses is the key to speed them up. Transformations similar to those applied by us to the slope algorithm can be used to other related geospatial algorithms based on the neighborhood relation (e.g. aspect, curvature, focal flow).

It is important to use performant and energy-efficient parallel processing on multi-core machines or hybrid cluster systems for spatial analysis such as slope. Currently, it is not possible to create an architecture-independent solution to the problem of optimization of algorithms in terms of energy efficiency because energy consumption is closely related to a specific architecture. However, some results show that techniques that optimize the performance of algorithms can also improve energy efficiency [11].

In this article, we investigate the impact of the optimization of the slope algorithm on performance, power and energy consumption, and the correlation between them on multicore architecture. We use this architecture because many cores enable running multiple processes at the same time with greater ease, increasing the performance of applications and programs, especially those operating on large data size. The slope algorithm acceleration is achieved by vector optimization for each core and parallel implementation for multi-core processors. The energy efficiency of the proposed solutions is assessed for data (DEM files) of various sizes. The Intel RAPL (Running Average Power Limit) [3], [9] interface was used as a source of information on energy consumption.

The main contributions of this article are following:

- Results of the tests and conclusions from the evaluation of the execution time and acceleration of different versions of the slope algorithm for various data sizes.
- Conclusions on the impact of the optimization techniques on power consumption.
- Conclusions on the impact of the effect of vectorization and parallelization on energy consumption.
- Analysis of the correlation between performance and energy consumption.

This paper is organized as follows. Section 2 presents related works. Section 3 is devoted to the slope algorithm. It explains what slope is and describes the algorithm used to calculate it. It also describes versions of the algorithm tuned to the architecture used: version that enables the use of vector registers and two parallel versions. In Section 4, we concentrate on the details of conducting tests and on the discussion and explanation of the results. In Section 5, we present the conclusions of the conducted experiment and further research directions.

II. Related works

Many studies are targeted at the prediction of power consumption and energy savings for various processing units. These issues can be addressed at both the hardware and software levels. The software-level approach involves redesigning numerical algorithms from various fields in terms of energy efficiency. Some examples one can find in [7], [6], [8], [2], [1] and [11]. In this, paper we also take the software-level approach. We focus on general-purpose processors (CPUs) with vector units and selected geospatial algorithms.

To take full advantage of the computing potential of CPUs while ensuring energy efficiency, both vector and multicore processing should be used. The works [7], [6], and [8] showed advantages and limitations of vectorization for energy efficiency. They describe the techniques of automatic and manual vectorization of Gauss elimination and Gram-Smith orthogonalization algorithms for multicore computers. In our work, we investigate the energy efficiency of the code manually vectorized with compiler directives. The directives inform the compiler about the vectorizable instructions in an appropriately transformed code.

In [2] and [1], the authors study energy efficiency in the context of high-performance dense linear algebra libraries for multicore computers and multithreading. Both works used transformed block matrix decomposition algorithms. Energy consumption measurements were reported along with parallel performance numbers on multi-socket machines. The conclusion was that the use of block linear algebra algorithms results in energy savings. In our work, we study the energy efficiency of algorithms employing loop fission transformations.

The paper [11] examines the impact of performance optimization on the power and energy consumption of Intel Xeon Scalable processors. The studies were conducted on the example of the MPDATA application (a finite-difference solver for geophysical flows). MPDATA is a memory-bound application and it needed optimization to utilize both vector and multicore processing. The research showed that improving memory access (i.e. cache reusing and data locality) for such memory-bound applications also improves energy efficiency. Additionally, the authors show that SIMD vectorization can lead to energy consumption reduction and, at the same time, increase the efficiency of calculations. They also evaluate the CPU frequency scaling as a tool for balancing energy savings with admissible performance losses.

In our research, we investigate the effect of vectorization and combining it with multithreading on energy efficiency for geospatial raster algorithms on a multi-core machine with vector units.

III. HPC Slope Algorithm

Digital elevation model (DEM) is a digital representation of earth's surface. DEM and its derivatives (land-surface parameters) are the basis of various geomorphometric analyses [4]. The slope is one of the most important and most frequently computed land-surface parameters.

DEM is most frequently represented as a two-dimensional regular grid of cells. Each cell contains an elevation value. The same representation is used for land-surface parameters, such as the slope.

All the algorithms discussed in this section use this representation for input and output data.

Various ways of calculating slope from DEM are discussed in [12] and [13]. Our implementation uses the method described in [5]. This method is also implemented in most popular GIS software packages (ArcGIS, QGIS, GRASS GIS, SAGA GIS).

In this section, we discuss some details of our implementation and its improvements, i.e. vectorization and parallelization.

A. Basic algorithm

Slope at a given point can be described as the maximum rate of change of elevation value at that point. It can be expressed as a slope angle (between 0 and 90 degrees). The method of determining the slope is presented by Algorithm 1. The algorithm requires two arrays: dem stores elevation values, and $slope$ — computed slope values in each cell. dem is an input array, $slope$ is an output array. These arrays have got the same dimensions. Additionally, we also need information about cell size (Δx, Δy) and the number of rows n and columns m of arrays. Δx means the grid interval from west to east, expressed in the same units as the elevation in dem. Δy means the grid interval from south to north, expressed in the same units as the elevation. In the basic implementation of slope, all instructions are executed sequentially, one by one.

B. Vectorization

The slope algorithm reads its input data from the main memory. It also writes its output to the main memory. This results in intensive data movement to and from the main memory. This movement severely limits the performance. To prevent this and make better use of cache we developed a

Algorithm 1: Base: Basic sequential algorithm

```
Input: $dem$ — input DEM
$\Delta x$ — west-to-east cell size
$\Delta y$ — south-to-north cell size
$n$ — number of rows of $dem$
$m$ — number of columns of $dem$
Output: $slope$ — output of the same size as $dem$
for $r \leftarrow 1 \ldots (n-2)$ do
    for $c \leftarrow 1 \ldots (m-2)$ do
        $p \leftarrow ((dem[r-1][c+1] + 2dem[r][c+1]$
            $+ dem[r+1][c+1])$
            $- (dem[r-1][c-1] + 2dem[r][c-1]$
            $+ dem[r+1][c-1]))$
            $/(8\Delta x)$
        $q \leftarrow ((dem[r+1][c-1] + 2dem[r+1][c]$
            $+ dem[r+1][c+1])$
            $- (dem[r-1][c-1] + 2dem[r-1][c]$
            $+ dem[r-1][c+1]))$
            $/(8\Delta y)$
        $slope[r][c] \leftarrow \arctan(\sqrt{p^2 + q^2})$
return $slope$
```

Algorithm 2: Transformed with SIMD: Transformed sequential algorithm

```
Input: $dem$ — input DEM
$\Delta x$ — west-to-east cell size
$\Delta y$ — south-to-north cell size
$n$ — number of rows of $dem$
$m$ — number of columns of $dem$
Output: $slope$ — output of the same size as $dem$
for $r \leftarrow 1 \ldots (n-2)$ do
    /* calculating $p$ */
    #pragma omp simd
    for $c \leftarrow 1 \ldots (m-2)$ do
        $p[c] \leftarrow dem[r-1][c+1] - dem[r-1][c-1]$
    #pragma omp simd
    for $c \leftarrow 1 \ldots (m-2)$ do
        $p[c] \leftarrow p[c]$
            $+ 2(dem[r][c+1] - dem[r][c-1])$
    #pragma omp simd
    for $c \leftarrow 1 \ldots (m-2)$ do
        $p[c] \leftarrow (p[c] + (dem[r+1][c+1]$
                $- dem[r+1][c-1]))/(8\Delta x)$
    /* calculating $q$ */
    #pragma omp simd
    for $c \leftarrow 1 \ldots (m-2)$ do
        $q[c] \leftarrow dem[r+1][c-1]$
            $+ 2dem[r+1][c] + dem[r+1][c+1]$
    #pragma omp simd
    for $c \leftarrow 1 \ldots (m-2)$ do
        $q[c] \leftarrow (q[c] - (dem[r-1][c-1]$
                $+ 2dem[r-1][c]$
                $+ dem[r-1][c+1]))/(8\Delta y)$
    #pragma omp simd
    /* calculating $slope$ */
    for $c \leftarrow 1 \ldots (m-2)$ do
        $slope[r][c] \leftarrow \arctan(\sqrt{p[c]^2 + q[c]^2})$
return $slope$
```

new version of the algorithm. The modification consisted in transforming the inner loop. We use the loop fission technique [14]. Algorithm 2 describes the transformed slope algorithm. The modified version improves cache usage, reduces main memory access, and enables the use of vector registers found in every modern CPU. It employs the vectorization along the `c`-dimension using the **`#pragma omp simd`** directive from the OpenMP standard.

C. Parallelization

Algorithms 3 and 4 describe parallelized algorithms 1 and 2 (base and transformed). In both parallel implementations, the outermost loop is processed in parallel using the OpenMP standard. These versions make use of data parallelism in `r`-dimension with the **`#pragma omp parallel for`** directive.

IV. Numerical experiment – methodology and results analysis

A. Methodology

We benchmark four versions of the slope algorithm:

- Base (Algorithm 1) – basic sequential algorithm,
- Parallelized base (Algorithm 3) – parallel version of Base,
- Transformed with SIMD (Algorithm 2) – transformed sequential algorithm with vectorization hints for compiler,
- Parallelized transformed with SIMD (Algorithm 4) – parallel version of Transformed with SIMD; The code is transformed so that it can be executed in parallel on multiple cores and that it can be efficiently vectorized.

All versions have been implemented in C++.

For tests, we used 1-meter resolution GeoTIFF files. The area of the smallest one we denote by R. We consider GeoTIFF files in 10 different sizes: from R to 10R, covering areas from 200 km^2 to 2000 km^2. The use of multiples of R facilitates analysis of scalability in data size. The sizes of the test data are presented in Table I. Each test area is an $m \times n$ matrix (m columns, n rows). The data type of each element of input and output arrays, as well as the type used for all calculations, is a single-precision floating-point type (4 bytes).

The performance and power measurements presented in this work were performed on a computing platform equipped with a modern multi-core processor code-named Haswell and with the following parameters:

Algorithm 3: Parallelized base: Parallelized basic algorithm

```
Input: $dem$ — input DEM
$\Delta x$ — west-to-east cell size
$\Delta y$ — south-to-north cell size
$n$ — number of rows of $dem$
$m$ — number of columns of $dem$
Output: $slope$ — output of the same size as $dem$
1 #pragma omp parallel for
  for $r \leftarrow 1 \ldots (n-2)$ do
2     for $c \leftarrow 1 \ldots (m-2)$ do
3         ...
          /* loop body as in Alg. 1 */
4         ...
5 return $slope$
```

Algorithm 4: Parallelized transformed with SIMD: Transformed sequential algorithm

```
Input: $dem$ — input DEM
$\Delta x$ — west-to-east cell size
$\Delta y$ — south-to-north cell size
$n$ — number of rows of $dem$
$m$ — number of columns of $dem$
Output: $slope$ — output of the same size as in $dem$
1 #pragma omp parallel for
  for $r \leftarrow 1 \ldots (n-2)$ do
2     ...
      /* loop body as in Alg. 2 */
3     ...
4 return $slope$
```

```
processor: 2x Intel Xeon E5-2670 v3 @ 2.30GHz
(2x12 cores with HT)
RAM: 128GB (8x16GB DDR4 2133MHz ECC)
```

The machine is equipped with 2 processors 12 cores each. Thus the program can be run in 24 threads (the number of threads equals the number of cores). In addition, this processor supports Intel Hyper-Threading (HT) technology and allows concurrent execution of 2 threads on one processor core. However, we do not use it during our tests.

The following software was installed during tests:

```
operating system: CentOS 7.6
kernel: Linux 3.10.0
GCC: 8.3.1 z OpenMP 4.5
GDAL: 2.4.0
```

The programs were compiled with the GCC compiler and the optimization flag `-O3` turned on. The GDAL (Geospatial Data Abstraction Library) [3] library was used to read from and write to GeoTIFF files. In each version of the algorithm, to help the optimization, data alignment in memory was used.

In this chapter, we also describe the impact of the algorithm modifications on power and energy consumption. The measurements were performed using the Intel's Running Average Power Limit (RAPL) interface, which is available for all Intel processors, starting with the Sandy Bridge architecture. RAPL uses machine-specific registers to monitor and control power consumption in real-time. On multi-socket systems, RAPL provides the results for each socket (each package) separately. RAPL also provides separate measurement values for the memory modules (DRAM) associated with each socket. Starting from Haswell processors which are equipped with fully integrated voltage regulators, the accuracy of the measurements returned by RAPL has significantly improved [9].

TABLE I: Characteristics of test areas

Area	Number of rows (n)	Number of columns (m)	Number of cells ($m \times n$)	[GB]
R	4000	50000	200000000	0.75
2R	8000	50000	400000000	1.49
3R	12000	50000	600000000	2.24
4R	16000	50000	800000000	2.98
5R	20000	50000	1000000000	3.73
6R	24000	50000	1200000000	4.47
7R	28000	50000	1400000000	5.22
8R	32000	50000	1600000000	5.96
9R	36000	50000	1800000000	6.71
10R	40000	50000	2000000000	7.45

B. Execution Time

Each of the implemented algorithms requires reading the input data and writing the results. The times of these operations do not differ between algorithms for the same data size. Table 1 shows the average time of reading and writing data for each input data size.

TABLE II: Average read and write time [s] for each input data size

Area	Reading [s]	Writing [s]
R	0.84	4.04
2R	1.51	8.45
3R	2.24	11.50
4R	2.81	15.56
5R	4.65	20.63
6R	4.58	24.41
7R	5.98	30.17
8R	5.66	32.52
9R	7.40	37.99
10R	7.08	40.37

First, we measured the execution time of each algorithm for different sizes of input data. Execution times are given for calculations only — without reading and writing data. The results are presented in Figure 1. Tests show that the transformation of the algorithm accompanied with vectorization hints shortens the execution time by about 30% compared to the base version. The use of parallelism reduces the execution time much more: parallelization alone reduces the time by more than 80% compared to the base version, while the parallelized and vectorized version shortens execution time by almost 90% for each test data size.

Table III shows the speedup gained by vectorization and parallelization for selected data sizes. In table III, the pa-

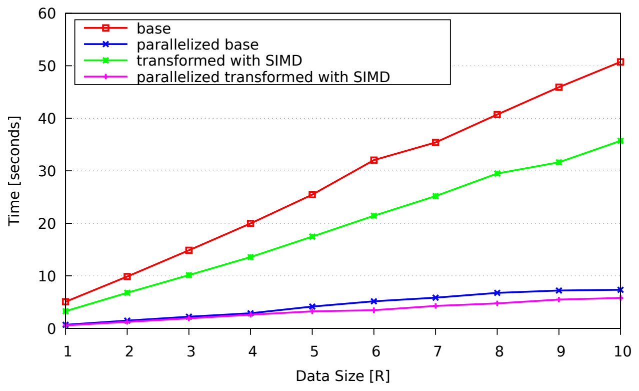


Fig. 1: Execution time of slope algorithms for different data sizes

TABLE III: Relative speedup of the slope algorithms

	R	5R	10R
T_{Base} / $T_{Transformed}$	1.55	1.46	1.42
$T_{ParallelizedBase}$ / $T_{ParallelizedTransformed}$	1.22	1.29	1.27
T_{Base} / $T_{ParallelizedBase}$	7.33	6.11	6.90
$T_{Transformed}$ / $T_{ParallelizedTransformed}$	5.81	5.40	6.16
T_{Base} / $T_{ParallelizedTransformed}$	8.98	7.87	8.76

rameter $T_{Algorithm}$ denotes the execution time of the given algorithm. The table shows that the vectorization of the base version gives a greater improvement (from 1.42 to 1.55) compared to the vectorization of the parallelized version (from 1.22 to 1.29). The very parallelization of the base version of the algorithm speeds up the execution about 7 times, while the program using OpenMP pragmas and the use of SIMD extensions (Parallelized Transformed with SIMD) is performed on average 8 times faster and for some sizes even 9 times faster.

C. Evaluation of Power and Energy Consumption

1) Power Consumption for Different Slope Versions: Next, we evaluate the impact of the applied performance optimization steps on the power consumption, which is measured with the RAPL interface. RAPL counters are updated once every 1 ms and have an adjustable sampling rate that has been set to 100 ms.

Figure 2 shows the power profiles of the Base algorithm implementation for three data sizes: R, 5R, 10R. The experiment was conducted on a dual-socket system, therefore RAPL returns the measurement results separately for packages attached to each socket: Package0 and Package1, and separately for the memory attached to the package's integrated memory controller: DRAM0 (attached to Package0) and DRAM1 (Package1). Figure 2 shows power profiles for the execution of the Base algorithm successively for sizes R, 5R, and 10R. Between the runs, the system is idle for 3 seconds. The Base algorithm is sequential, so only one core from the selected socket is being employed at any time. The figure shows the power consumption of the system components during program execution for the data size R (3–15 sec.), 5R (17–71 sec.), and 10R (73–174 sec.), respectively. Data reading times (R_{read}, $5R_{read}$, $10R_{read}$ — marked in green), computation times (R_{comp}, $5R_{comp}$, $10R_{comp}$ — marked in red), and results writing times (R_{write}, $5R_{write}$, $10R_{write}$ — marked in blue) are shown above the power consumption graphs.

Figure 3 shows the power profiles of the execution of four versions of the slope algorithm: Base, Transformed with SIMD, Parallelized, and Parallelized transformed with SIMD. Sequential algorithms (Base and Transformed with SIMD) use exactly one core out of 24 available. Parallel versions of algorithms (Parallelized base and Parallelized transformed with SIMD) utilize all 24 available cores. As you can see in Figures 3a–3d, the system is idle before and after the execution of a program. In this state, it consumes about 39 W (about 20 W for each socket: Package + DRAM). This is the minimum power necessary to keep the system idle. After the start of the program, more system components are involved (cores, memory), which increases the power consumption. The system returns to the idle state when it completes the program execution. One can observe that for modified versions (Transformed with SIMD, Parallelized, Parallelized transformed with SIMD), the system returns to the idle state faster (the program execution takes less time), and because the reading and writing time is constant for a given size, the calculation time itself is shorter. Graphs 3a and 3b show that the levels of power consumption for the sequential versions of the algorithm are close to each other. The maximum instantaneous power of the entire system (both sockets: Package0 + DRAM0, Package1 + DRAM1) is 83 W for the Base version and 82 W for the Transformed with SIMD version. Graphs 3c and 3d show the power consumption for the parallel versions. They show that only when calculations are performed, the power consumption increases and the cores on both sockets work simultaneously. The maximum instantaneous power for the Parallelized base version is 190 W, and for the Parallelized transformed with SIMD version it is 219 W. Reading and writing data does not differ between versions (and these are operations performed sequentially), so the power consumption for these operations is the same for all algorithms.

Regardless of the version of the algorithm, reading and writing data takes the same time and power, and thus when analyzing the average power and energy consumption, we consider the computation alone.

Figure 4 shows a comparison of the average power consumption of the considered system components (Package0 + DRAM0, Package1 + DRAM1) by different versions of the slope algorithm and for different data sizes. It also shows the value of the system idle power. The graph shows that for the sequential versions (Base, Transformed with SIMD) the average power consumption is almost identical for each size. Parallel versions of algorithms run faster (Figure 1) but have higher average power consumption. The average power consumption of the Parallel base version is on average 43% higher than the Base version, and the average power consumption of the Parallelized transformed with SIMD version is on average 31% higher than the Base version. Despite having the highest

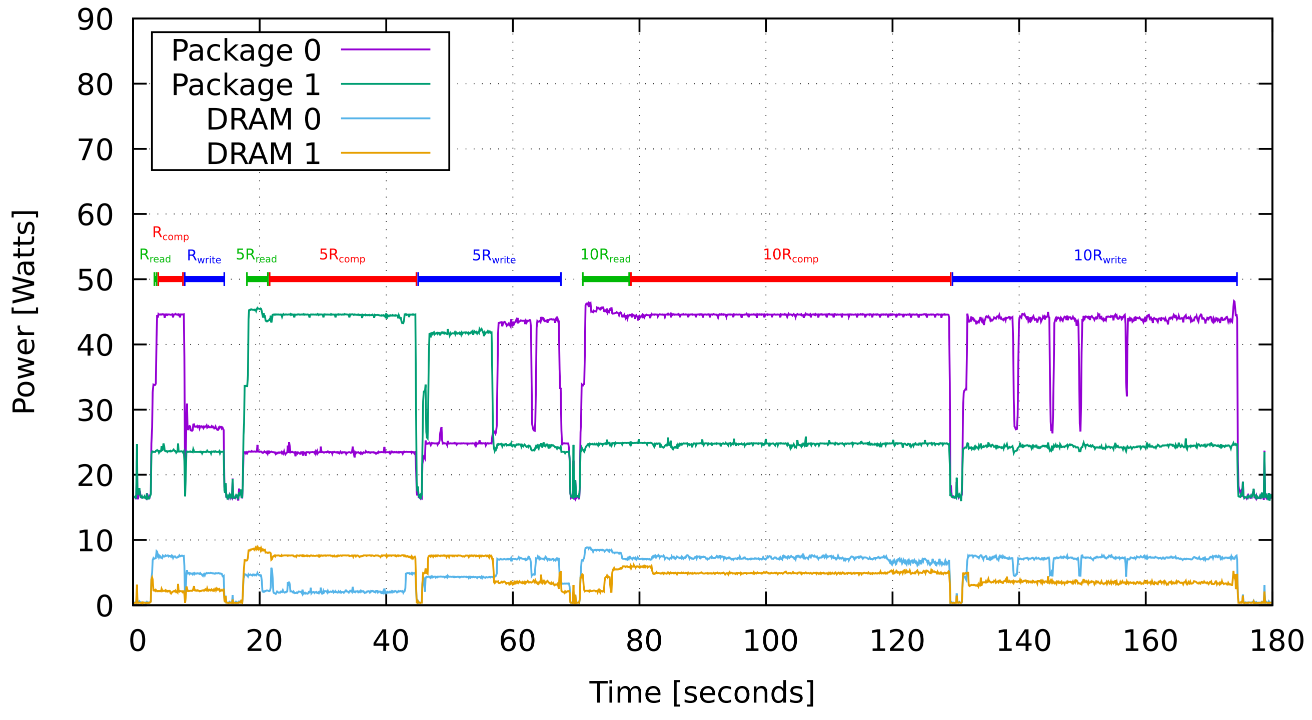


Fig. 2: The power profiling of Base algorithm for R, 5R, 10R with RAPL

instantaneous power, the Parallelized transformed with SIMD version has a lower (2–16%) average power consumption than the Parallelized base version.

2) Energy consumption: In this section, we show how the performance optimization steps affect the total computation energy consumption.

Table IV presents a comparison of energy and performance-related parameters for different versions of the slope algorithm for a data size of 10R. In addition to the execution time, the total energy consumption obtained from the RAPL interface is given. Moreover, the performance (in Mflop/s) and the energy efficiency (in Mflop/J) are given. Energy efficiency is expressed as the ratio of the number of floating-point operations to the total energy consumption.

Figure 5 shows the effect of different optimization mechanisms on the total energy consumption for different data sizes. One can see that the power consumption increases as the data size increases.

The modification used in the Transformed with SIMD algorithm, which removes the data dependency, allows for more efficient access to memory and the use of vectorization mechanisms, which results in a reduction of the computation time (Figure 1), but also a reduction in energy consumption (27% to 35%, 32% on average). This optimization affects neither the maximum instantaneous power consumption (Figure 3) nor the average power consumption (Figure 4).

The optimization used in the Parallelized base version, i.e. the use of OpenMP pragmas, allows all available cores in the system to work simultaneously. This causes momentary increases in power consumption (Fig. 3) and higher average power consumption (Fig. 4). However, it also results in a significant reduction of the computation time (Fig. 1) and energy consumption (74%–80%, 77% on average).

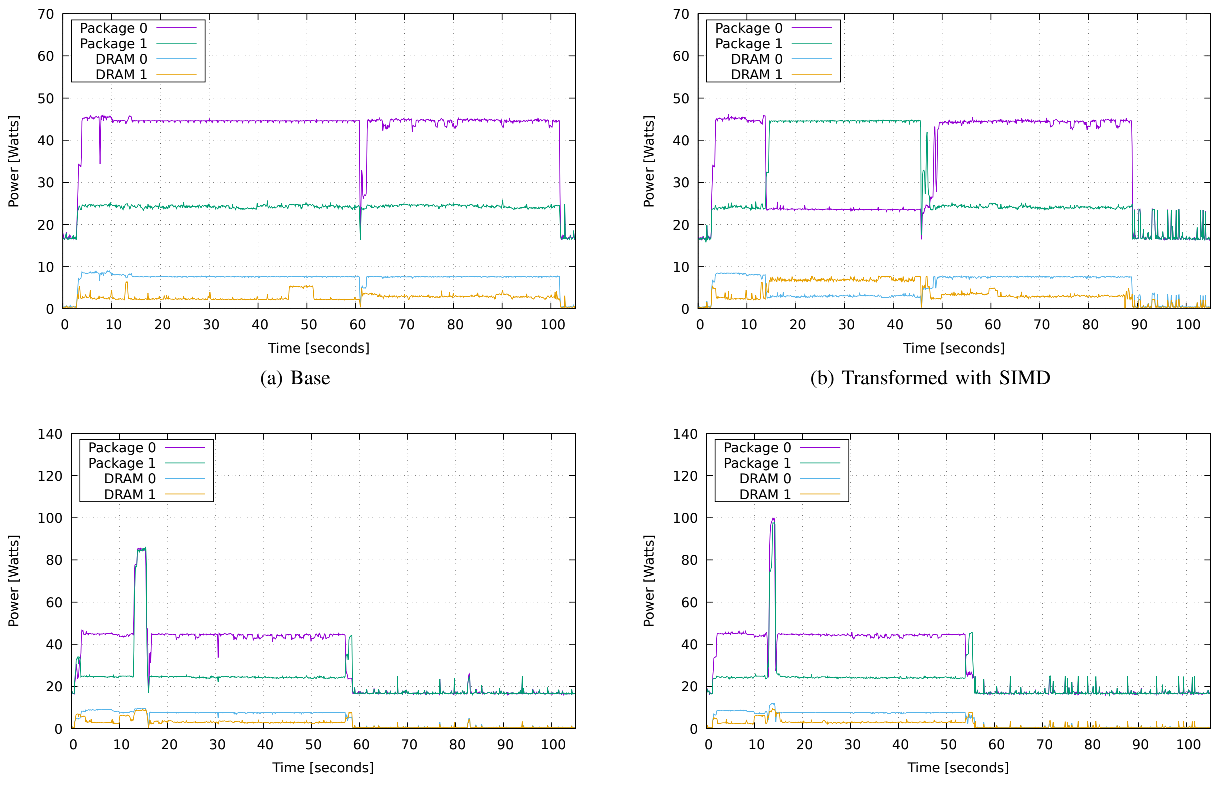
In the Parallelized transformed with SIMD version, the data dependencies are removed and vectorization is introduced into the parallel algorithm. Compared to Parallelized base, energy consumption is reduced by 28%. We also obtain even shorter computation times and lower average power consumption, with higher maximum instantaneous power.

However, the use of both parallelization and vectorization in the Parallelized transformed with SIMD version, reduces the execution time by almost 90% compared to the Base version (Figure 11). Despite the high maximum instantaneous power and higher average power consumption, we reduce energy consumption by an average of 84% compared to the Base version.

V. Conclusion

This article investigates four versions of the slope algorithm. Their execution time, power and energy consumption, and the correlation between performance and energy consumption are discussed. Measurements were made on a dual-socket machine with Intel Xeon E5-2670 processors using the Intel RAPL interface.

Both of the proposed transformations — vectorization and parallelization — reduce the computation time. The results



(a) Base (b) Transformed with SIMD

(c) Parallelized base (d) Parallelized transformed with SIMD

Fig. 3: The power profiling of Base, Transformed with SIMD, Parallelized base, Parallelized transformed with SIMD for 10R with RAPL

TABLE IV: Energy efficiency for various slope versions (10R)

Version	Time [s]	Total energy [J]	Performance [Mflop/s]	Energy efficiency [Mflop/J]
base	50.76	4028.18	788.00	0.20
parallelized base	7.35	867.68	5438.96	6.27
transformed with SIMD	35.72	2792.00	1119.58	0.40
parallelized transformed with SIMD	5.80	633.69	6901.30	10.89

show that the most performant version (parallel with vectorization) can shorten the computation time by more than 8 times. Through the analysis of power and energy consumption, one can see that vectorization alone, can slightly speed up the algorithm, without increasing average power consumption or maximum instantaneous power consumption. The reduction in the computation time allows for the reduction of energy consumption by about 30%.

The parallel version enables the simultaneous operation of all system components. This results in a significant reduction in computing time compared to the basic version (by almost 90%), but also an increase in both the maximum (more than 2 times) and average power consumption (by 30%–40%). Finally, however, we achieve a reduction in energy consumption by an average of 84% compared to the Base version. We can see that short periods of increased instantaneous power do not negatively affect the total energy consumption as long as the program computation time is shortened.

The conducted tests show that the proposed solutions respond well to increasing the size of the problem. For the largest data size tested, the energy efficiency improves (from 0.2 Mflop/J for the basic version to 10.8 Mflop/J for the most optimized version) along with the increase in performance.

The slope algorithm is not computationally intensive, but it is one of the basic components used in other geomorphometric analyses. The increase of performance that also causes the reduction of the energy consumption will improve the energy efficiency of the secondary analyses. Moreover, the proposed transformations can be applied to other raster analyses employing similar techniques (neighborhood relation) such as computation of aspect, curvature, and flow direction.

Seeing the improvement in energy efficiency after adapting the slope algorithm to a multi-core system, we plan to investigate the impact of the optimization methods applied to similar algorithms on the latest Ice Lake Intel Xeon processors and on the new generation of AMD Rome EPYC processors. We

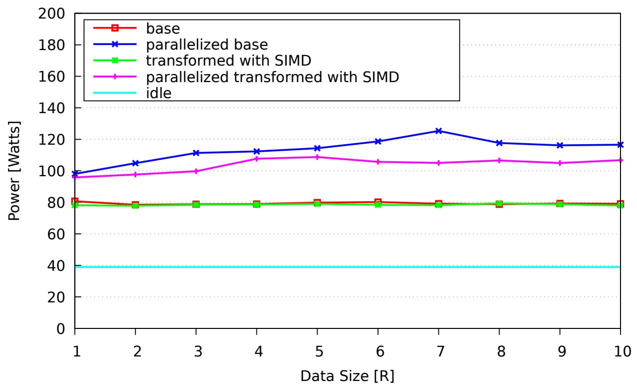


Fig. 4: Average power consumption as a function of different data sizes — and the idle power level

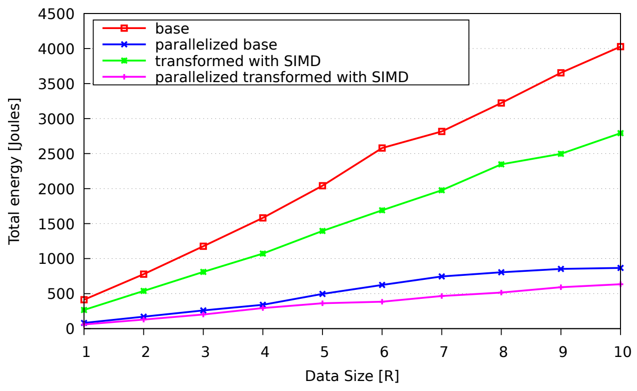


Fig. 5: Total energy as a function of different data sizes

also plan to investigate and improve other geospatial raster algorithms in terms of performance and energy efficiency.

References

[1] B. Bylina and J. Bylina. Studying OpenMP thread mapping for parallel linear algebra kernels on multicore system. *Bulletin of the Polish Academy of Sciences*, 66(6):981–990, 2018.

[2] J. Dongarra, H. Ltaief, P. Luszczek, and V. M. Weaver. Energy footprint of advanced dense numerical linear algebra using tile algorithms on multicore architectures. In *2012 Second International Conference on Cloud and Green Computing*, pages 274–281, 2012.

[3] D. Hackenberg, R. Schöne, T. Ilsche, D. Molka, J. Schuchart, and R. Geyer. An energy efficiency feature survey of the Intel Haswell processor. In *2015 IEEE International Parallel and Distributed Processing Symposium Workshop*, pages 896–904, 2015.

[4] T. Hengl and H.I. Reuter, editors. *Geomorphometry: Concepts, Software, Applications*, volume 33. Elsevier, Amsterdam, 2008.

[5] B. K. P. Horn. Hill shading and the reflectance map. *Proceedings of the IEEE*, 69(1):14–47, Jan 1981.

[6] T. Jakobs, B. Naumann, and G. Rünger. Performance and energy consumption of the SIMD Gram–Schmidt process for vector orthogonalization. *The Journal of Supercomputing*, 76:1999–2021, 2019.

[7] T. Jakobs and G. Rünger. Examining energy efficiency of vectorization techniques using a Gaussian elimination. In *2018 International Conference on High Performance Computing Simulation (HPCS)*, pages 268–275, 2018.

[8] T. Jakobs and G. Rünger. On the energy consumption of load/store AVX instructions. In *2018 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pages 319–327, 2018.

[9] K. Khan, M. Hirki, T. Niemi, J. Nurminen, and Z. Ou. RAPL in action: Experiences in using RAPL for power measurements. *ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS)*, 3, 01 2018.

[10] S.D. Peckham. Chapter 25 Geomorphometry and Spatial Hydrologic Modelling. In Tomislav Hengl and Hannes I. Reuter, editors, *Geomorphometry*, volume 33 of *Developments in Soil Science*, pages 579 – 602. Elsevier, 2009.

[11] L. Szustak, R. Wyrzykowski, T. Olas, and V. Mele. Correlation of performance optimizations and energy consumption for stencil-based application on Intel Xeon scalable processors. *IEEE Transactions on Parallel and Distributed Systems*, 31(11):2582–2593, 2020.

[12] J. Tang, P. Pilesjö, and A. Persson. Estimating slope from raster data – a test of eight algorithms at different resolutions in flat and steep terrain. *Geodesy and Cartography*, 39(2):41–52, 2013.

[13] S. Warren, M. Hohmann, K. Auerswald, and H. Mitasova. An evaluation of methods to determine slope using digital elevation data. *Catena*, pages 215–233, 12 2004.

[14] M. E. Wolf and M. S. Lam. A loop transformation theory and an algorithm to maximize parallelism. *IEEE Transactions on Parallel and Distributed Systems*, 2(4):452–471, 1991.