Evaluation of Heterogeneous Scheduling Algorithms for Wavefront and Tile Parallelism in Video Coding

Natalia Panagou[1], Maria Koziri[1], Panos K. Papadopoulos[2], Panagiotis Oikonomou[1], Nikos Tziritas[1], Kostas Kolomvatsos[3], Thanasis Loukopoulos[2(✉)], and Samee U. Khan[4]

[1] Computer Science and Telecommunications Department, University of Thessaly, Volos, Greece
{napanagou, mkoziri, paikonom, nitziri}@uth.gr
[2] Computer Science and Biomedical Informatics Department, University of Thessaly, Volos, Greece
{ppapadopoulos, luke}@uth.gr
[3] Informatics and Telecommunications Department, University of Athens, Athens, Greece
kostasks@di.uoa.gr
[4] Electrical and Computer Engineering Department, North Dakota State University, Fargo, ND, USA
samee.khan@ndsu.edu

Abstract. Video is by far the "biggest" Big Data, stretching network and storage capacity to their limits. To handle the situation, video compression has been an active field of study for many years, producing output of huge commercial interest, e.g., MPEG-2 and DVD. However, video coding is a computationally expensive process and for this reason, parallelization was proposed at various granularity levels. Of particular interest, are block level methods implemented in HEVC (High Efficiency Video Coding) which was designed to be the successor of H.264/AVC for the 4K era. Parallelization in HEVC is supported by the following three modes: slices, tiles and wavefront. While considerable research was conducted on the parallelization options of HEVC, it was focused on the case of homogeneous processors. In this paper we consider video coding parallelization when the processing elements are heterogeneous. In particular, we focus on wavefront and tile parallelism and measure the performance of scheduling schemes for the induced subtasks. Through simulation experiments with dataset values obtained from common benchmark sequences, we conclude on the relevant merits of the evaluated scheduling algorithms.

Keywords: Scheduling · Heterogeneous processors · Video coding · Parallelism · Wavefront · Tiles · HEVC

1 Introduction

As we are rapidly moving towards realizing fully interconnected smart IoT environments at a large scale, many challenges are being posed. Of particular importance is data collection and stream processing [1] which consumes both network and

V. Issarny et al. (Eds.): ICIOT 2019, LNCS 11519, pp. 16–27, 2019.
https://doi.org/10.1007/978-3-030-23357-0_2

computational resources. Among all data sources, video feeds from street cameras and mobile smart devices, undoubtedly pose the "biggest" Big data size wise. As an example, Cisco indicated in [2] that video related traffic accounted for 55% of the total mobile traffic in 2015 and was increased by 75% compared to the previous year. Such trends will likely remain in the foreseeable future due to the wide use of 4K in smart phone cameras and TV sets. Therefore, in order to efficiently deploy intelligent video processing applications, e.g., car plate recognition [3] etc., as components of a smart IoT ecosystem, it is of paramount importance to reduce the size of video streams without affecting quality (referred to as video compression or video coding).

One of the most successful video coding standards is the popular H.264/AVC [4] which was designed to tackle the challenges of the FullHD era. Nevertheless, with the dawn of 4K resolutions H.264/AVC saw its age, resulting in the development of a new generation of video coding standards. Examples consist of High Efficiency Video Coding-HEVC [5] (also termed H.265) by the MPEG group in 2012, VP9 [6] by Google in 2013 and the recently launched AV1 [7] by AOMedia. Simultaneously, the successor of HEVC, termed VVC (Versatile Video Coding) or H.266 [8] is under development and scheduled for release in 2020. While the battle for the succession of H.264/AVC still wages, what is common on the newer standards is the fact that they offer improved compression ratio (for the same quality) compared to H.264/AVC. Nevertheless, to achieve such performance improvement, particularly for 4K compression, increased computational cost is involved which can only be alleviated through parallelism.

In the relevant literature parallelism was applied at various granularity levels of the video coding process such as: (i) within a block of pels (e.g., at the motion estimation or filtering steps [9]); (ii) at a block or group of blocks level (e.g., slice parallelism [10]) and (iii) at a frame level (e.g., by assigning different Groups of Pictures (GOPs) to different processors [11]). With the exception of the coarser granularity level (per GOP) most of the relevant literature on video coding parallelism assumed a homogeneous set of processors. Nevertheless, heterogeneous computing scenarios are becoming more and more important, e.g., with big.LITTLE processors [12]. In this paper we turn our attention to block level video coding parallelism assuming heterogeneous processors. Since a number of works, e.g., [13], outlined that among the parallelization granules implemented in HEVC (slices, tiles and wavefront), the last two offer the best coding performance we restrict our study to tile and wavefront parallelism.

In normal mode, the encoding of the blocks of a frame (termed Coding Tree Units-CTUs in HEVC) is done in raster order (row by row). In wavefront, CTU encodings are again done from left to right but different CTU rows can be processed in parallel, as long as the following constraint is respected: a CTU can commence encoding once the upper and upper-right CTUs are encoded. Figure 1 provides an example whereby the greyed CTUs are the ones that are already encoded. Clearly, by considering each CTU coding as a separate task, the encoding of a frame can be modeled as a DAG (Directed Acyclic Graph). Thus, in order to complete the frame's encoding at the minimum possible time, suitable scheduling algorithms are important, particularly in the presence of heterogeneity.

Similarly, efficient scheduling can prove useful in the case of tile parallelism where a frame is split into a grid of rectangular areas (tiles) comprised of multiple CTUs (see

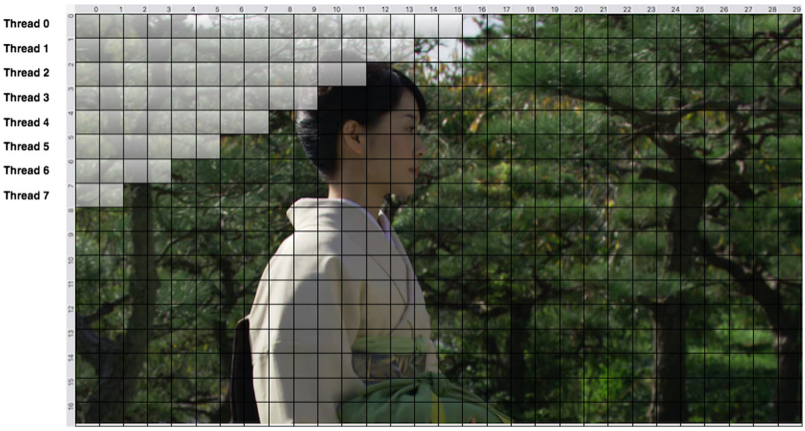

Fig. 1. Example of wavefront parallelism with 8 threads (Kimono sequence).

Fig. 2. Example of partitioning a frame into 12 tiles (Kimono sequence).

Fig. 2). Since the encoding of each tile is independent from others, independent task (tile) scheduling techniques are applicable. Even if a plethora of results exist on task scheduling (e.g., see [14] for a comparative study), the particular case of scheduling algorithms for video coding parallelism was typically overlooked.

Our primary contribution rests on evaluating scheduling algorithms for wavefront and tile parallelism under heterogeneous assumptions concerning processors' computational power. The evaluation is based on simulation experiments using a realistic dataset for CTU coding times, obtained by encoding common test video sequences [15] using the HM 16.15 reference encoder [16] of HEVC. Results indicate that compared to random scheduling decisions (as is the current practice by many encoders) reduction in makespan of even 50% is achievable. To the best of our knowledge this is the first paper tackling heterogeneous scheduling in video coding at a finer granule than the GOP level.

The rest of the paper is organized as follows. Section 2 discusses related work. Problem details together with the DAG formulation and the tested scheduling schemes are presented in Sect. 3. Performance evaluation results are illustrated in Sect. 4 together with a summary of our findings. Finally, Sect. 5 includes our concluding remarks.

2 Related Work

HEVC offers three main parallelization approaches within a frame: wavefront, slices and tiles. Significant research exists for all three approaches. A comparative study examining the strengths of each one is provided in [13]. As it is demonstrated, in terms of quality and bitrate, wavefront is the winner. However, the achievable speedup from parallelization is rather restricted due to task precedence constraints. Furthermore, both slices and tiles provide other complementary strengths. Slices are essentially encoded as sub-frames, therefore they provide flexibility in the network transmission layer at the expense of higher header overhead, thus increased bitrate. Tiles on the other hand, allow for fine tuning the encoding parameters, e.g., the selected QP (quantization parameter), so as to increase video quality at ROIs (Regions of Interest).

In terms of slice and tile level parallelism most existing works assume homogeneous processors. In [17] the achievable speedup from slice parallelism was evaluated for HEVC assuming uniform static slice partitions. Adaptive slice resizing was discussed in [10] and [18]. The core idea was to estimate the time complexity for encoding each Macroblock/CTU of the next frame and based on these estimations resize slices to evenly distribute workload. In [10] CTU time estimation was done using weighted past average of the actual encoding times experienced in previous frames, while [18] provided a more sophisticated estimator tuned for Low Delay (LD) encodings that take advantage of GOP hierarchy.

A similar estimator to the one of [18] was used in [19] with the aim of feeding a tile resizing algorithm. The algorithm itself was based on iteratively applying the optimal one dimensional array partitioning algorithm [20] across the two frame dimensions. In [21] the authors considered tile and slice resizing based on a CTU cost estimation that used a weighting function that takes into account CTU encoding mode and depth. The tile resizing scheme was based on first defining a master tile size, apply it in one of the four frame corners and build the remaining grid using this first tile definition as basis.

The aforementioned research assumed a one on one slice/tile to homogeneous processor assignment in which case evenly splitting expected slice/tile workload is the crucial factor, while scheduling decisions are trivial. In [22] the authors proposed to increase the number of tiles a frame is split into so as to exceed the number of available processors. In this way the task granule is effectively reduced allowing for better load balancing. Tile partitioning was done in a static manner and processor assignment was performed by a greedy bin packing heuristic. Although the achievable speedup was increased compared to the one on one tile-processor assignment, using too many tiles results in lowering video quality as indicated in [23] where the authors provide experimental results on the "optimal" number of tiles that should be used given the number of available processors. Finally, in [24] the authors propose a fast heuristic to

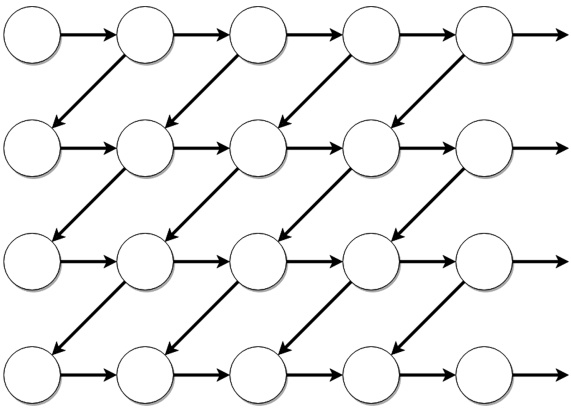

Fig. 3. Example of a wavefront DAG.

adapt tile size in case more tiles exist than processors. The heuristic is based on iteratively reducing the load of the most loaded processor by shrinking the area of one of its assigned tiles.

Overall, the case where fewer processors than tiles exist, gives rise to scheduling issues. Nevertheless, all the aforementioned works studied the case of homogeneous processors, whereas the focus of this paper is on heterogeneous processors both for tile and for wavefront parallelism.

3 Scheduling Heuristics

3.1 Formulation

Since video coding tasks are CPU bound we decided to adopt a simple model for heterogeneity that is based solely on processor speed. Let T_{i} denote the running time of the i^{th} encoding task (assuming a total order of them) over a baseline processor. Let processor speedup (SP_{j}) denote how faster the j^{th} processor is in running video coding tasks compared to the baseline processor. Calculating the running time of the i^{th} task on the j^{th} processor is done as follows:

$$Time_{ij} = T_i / SP_j \tag{1}$$

In the experiments we assume a one on one task processor mapping. In case multiple tasks should be allocated for execution on a particular processing core, we assume a FCFS policy and calculate task completion times accordingly.

Under tile level parallelism each tile is considered a separate encoding task. Notice, that these tasks are independent. In wavefront parallelism, each task corresponds to the encoding of a single CTU. In order for a CTU compression to commence, the upper and upper right CTUs as well as the left CTU (that resides in the same row) must have finished compression. These dependencies can be captured by means of a DAG. Figure 3 illustrates the DAG structure for an example 4×5 CTU grid.

3.2 Heuristics

We evaluated two scheduling heuristics for tile parallelism namely, MaxMin and MinMin. Given a set of available tasks, i.e., tasks that can commence immediately, MaxMin selects the heaviest task and assigns it to the processor where it will experience the earliest completion time. MinMin also performs a similar processor assignment, i.e., based on earliest completion time, but starts by assigning the most lightweight task first. Notice, that in tile parallelism all tasks are available for execution at the beginning of the process.

MaxMin and MinMin heuristics were also evaluated for wavefront parallelism, with tasks becoming available depending on DAG constraints. Furthermore, we also evaluated two other heuristics that are inspired by the DAG structure namely, MaxMin-Row and MinMin-Row. These heuristics work in a similar manner to MaxMin and MinMin with the exception being that instead of ordering tasks depending on estimated load, they order tasks according to the CTU row they reside. MaxMin-Row selects tasks in a lowest row first fashion and MinMin-Row does the opposite.

Table 1. Video sequences.

Name	Resolution	Total frames	CTUs per frame
PeopleOnStreet	2560 × 1600	150	1000
Traffic	2560 × 1600	150	1000
BasketballDrive	1920 × 1080	500	510
BQTerrace	1920 × 1080	600	510
Cactus	1920 × 1080	500	510
Kimono	1920 × 1080	240	510
ParkScene	1920 × 1080	240	510

4 Experiments

4.1 Simulation Setup

We used class A and class B common test sequences [15] with characteristics described in Table 1. We encoded the sequences using the HM 16.15 reference software [16]. Low Delay setting was selected with an initial I frame followed by P frames. The remaining parameters were as follows: GOP size = 4, CTU size = 64 × 64, bit depth = 4, max CTU partitioning depth = 4, QP = 32 and search method was TZ search. These parameters are commonly used in the related literature, e.g., [17, 18] and [19]. Video compression tasks were performed on a Linux machine with Intel Xeon E5-2650 processor running at 2.2 GHz.

We recorded the time it took to compress each CTU in all the frames of a particular sequence. We should mention that the aggregation of these times differed from the total encoding time spent by at most 5%. In the experiments, the values obtained from the actual coding of the video sequences constituted the performance of the baseline processor. In practice, CTU compression times can be estimated before the encoding of

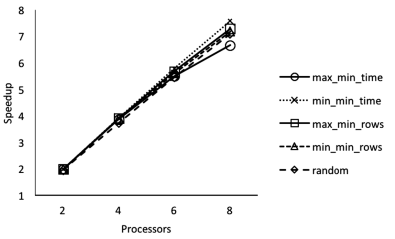

Fig. 4. Wavefront speedup (PeopleOnStreet sequence).

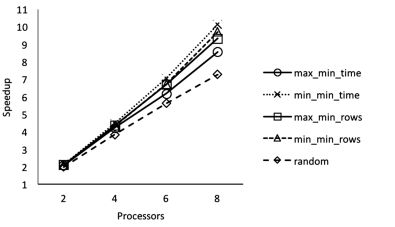

Fig. 5. Wavefront speedup (Traffic sequence).

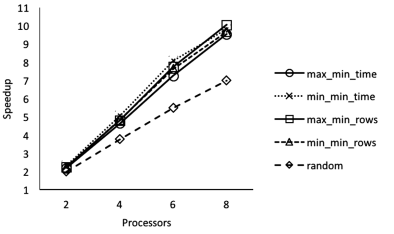

Fig. 6. Wavefront speedup (BasketballDrive sequence).

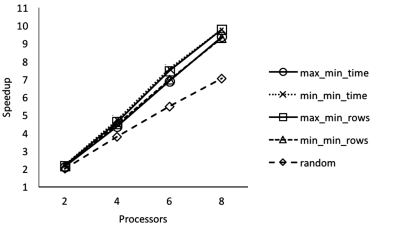

Fig. 7. Wavefront speedup (BQTerrace sequence).

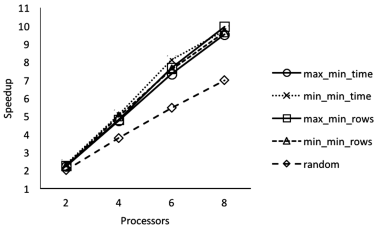

Fig. 8. Wavefront speedup (Cactus sequence).

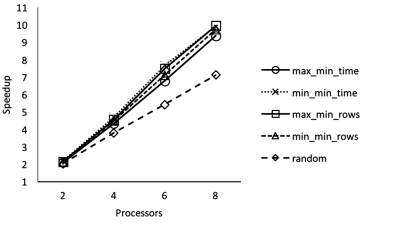

Fig. 9. Wavefront speedup (Kimono sequence).

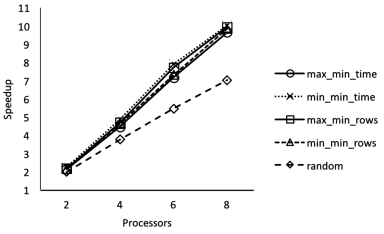

Fig. 10. Wavefront speedup (ParkScene sequence).

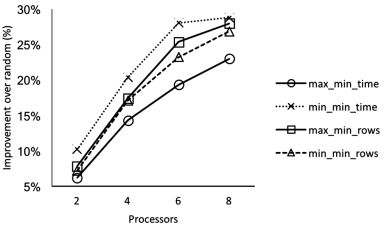

Fig. 11. Improvement over random (average of all sequences).

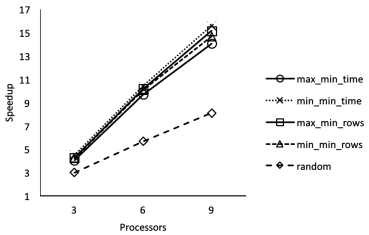

Fig. 12. Wavefront speedup (average for all sequences, processor speedups: 1, 2, 4).

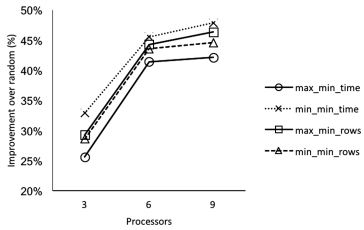

Fig. 13. Improvement over random (average for all sequences, processor speedups: 1, 2, 4).

a frame commences by one of the estimators proposed in the literature. In the paper the estimator used in [19] for LowDelay coding is adopted. Under the scheme, statistics from the first GOP (4 frames) are obtained, before being able to estimate the rest. For this reason, we only record the performance of scheduling schemes from the 5th frame onwards.

4.2 Wavefront Experiments

In a first experiment we evaluated the performance of the scheduling heuristics together with random scheduling, whereby each available task is assigned to one of the processors with probability following a uniform distribution. Figures 4, 5, 6, 7, 8, 9 and 10 plot the speedup from wavefront parallelization achieved for different number of processors assuming that half of them had processor speedup = 1 (baseline processor) and the other half processor speedup = 2 (i.e., twice fast to the baseline). As it can be observed, with the notable exception of PeopleOnStreet sequence, random scheduling has clearly worst performance compared to the other 4 options. It should be noted that in PeopleOnStreet sequence there exists avid movement throughout the frame thus, most CTU codings are equally complex and for this reason random choices perform in par with the remaining heuristics. It should also be noted that the achievable speedup in many cases exceeds the available processors. This is due to the way speedup is calculated: *Speedup* = *SequentialTime*/*ParallelTime*, whereby sequential time corresponds to the performance with processor speedup = 1 (in the experiments half of the processors are twice fast).

In order to better illustrate the overall performance of the heuristics we plot the improvement over random calculated as: (*TimeRandom*-*TimeHeuristic*)/*TimeRandom*. Figure 11 shows the average performance improvement (time reduction percentage) exhibited in all sequences. As it can be observed the MinMin heuristic is a clear winner followed by the heuristic that selects CTUs giving priority to the lowest CTU row.

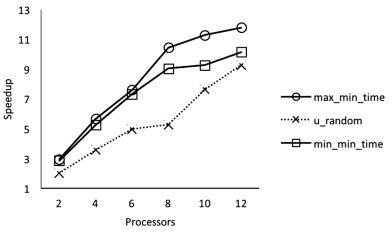

Fig. 14. Tile speedup (3 × 4 tiles, average for all sequences, processor speedups: 1, 2, static tile sizing).

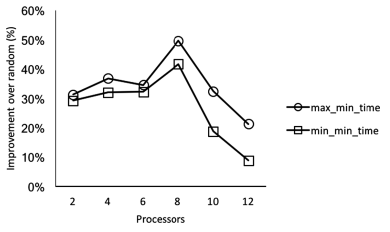

Fig. 15. Improvement over u_random (3 × 4 tiles, average for all sequences, processor speedups: 1, 2, static tile sizing).

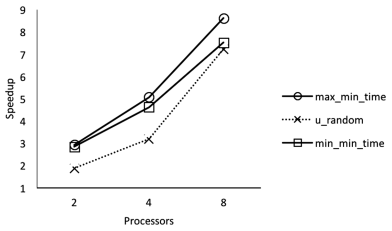

Fig. 16. Tile speedup (3 × 3 tiles, average for all sequences, processor speedups: 1, 2, static tile sizing).

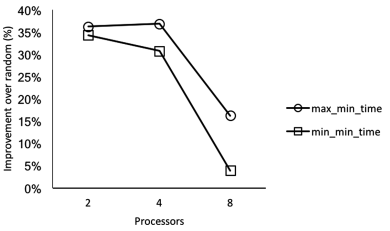

Fig. 17. Improvement over u_random (3 × 3 tiles, average for all sequences, processor speedups: 1, 2, static tile sizing).

To further confirm our findings, we run an experiment whereby 1/3rd of the processors had speedup of 1, 1/3rd speedup of 2 and another 1/3rd a speedup of 4. This scenario accounts for more heterogeneity. Thus, it is not surprising that the performance gap between random scheduling and the rest widens as shown in Fig. 12 which shows the achievable parallelization speedups and Fig. 13 which shows the performance improvement over random. Again notice that MinMin achieves the best performance reaching more than 45% of improvement, i.e., encodings are finished in almost half the time compared to random.

4.3 Tile Experiments

Next we evaluated the performance of MaxMin and MinMin on tile parallelism. We conducted experiments with tile partitioning of 3 × 4 and 3 × 3. Figures 14, 15, 16 and 17 plot the relevant performance results for the two cases. For comparison reasons we considered a modified random scheduling (u_Random) which assigns the same number of tiles on each processor but does so in a random uniform manner.

Results show that MaxMin and MinMin outperform for the largest part u_Random. Contrary to wavefront parallelism, MaxMin is the winner in tile parallelism. The performance difference between MaxMin, MinMin and u_Random is shown to initially rise as the number of processors increases, exhibits a peak and then drops as the number of processors tends to equal the number of tiles. Among MaxMin and MinMin,

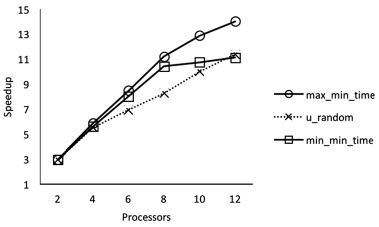

Fig. 18. Tile speedup (3 × 4 tiles, average for all sequences, processor speedups: 1, 2, adaptive tile sizing).

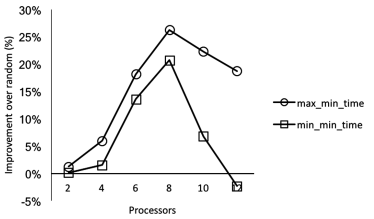

Fig. 19. Improvement over u_random (3 × 4 tiles, average for all sequences, processor speedups: 1, 2, adaptive tile sizing).

MinMin exhibits the sharpest drop. This can be explained by considering that in the extreme case where the number of processors equal the number of tiles, it will assign to the fastest processors the least heavy task, possibly leaving the heaviest ones to be assigned to low capacity processors.

In a final experiment we wanted to test the performance of tile scheduling schemes when adaptive tile resizing is performed. For this reason, we considered a scheme that aims at balancing tile sizes in a manner similar to the one proposed in [19]. Figures 18 and 19 plot the results for 3 × 4 tile partitioning. Again MaxMin is the winner, followed by MinMin in all but the case where the number of tiles and processors are equal (notice the negative improvement over u-Random for 12 processors in Fig. 19).

5 Conclusions

In this paper we evaluated scheduling heuristics for the case of video coding using wavefront and tile level parallelism, under heterogeneous assumptions concerning the available processor computational power. This issue, although crucial to performance was typically overlooked. Results demonstrate a different winner depending on the parallelization mode, with MinMin offering the best results for the wavefront case and MaxMin be the winner for tile parallelism. Concerning wavefront parallelism, placing preference depending on the row a CTU resides, bears no additional benefits to the basic MinMin scheme. At the same time, random scheduling decisions led to significantly inferior performance overall, even doubling in certain scenarios task completion time compared to other alternatives.

Acknowledgments. Panos K. Papadopoulos was supported by scholarship from IKY (State Scholarships Foundation) funded by the Act “Strengthening Human Resources Research Potential via Doctorate Research” of the Operational Program “Human Resources Development Program, Education and Lifelong Learning”, 2014-2020 co-financed by the European Social Fund (ESF) and the Greek Government.

Nikos Tziritas’ Post-doctoral research was carried out with an IKY scholarship funded by the Action “Supporting Post-doctoral Researchers” from EP resources “Development of Human Resources, Education and Lifelong Learning” with priority axes 6, 8, 9 and co-funded by the European Social Fund - ESF and the Greek government.

Samee U. Khan's work was supported by (while serving at) the National Science Foundation. Any opinion, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

1. Assuncao, M.D., Silva Veith, A., Buyya, R.: Distributed data stream processing and edge computing: a survey on resource elasticity and future directions. J. Netw. Comput. Appl. **103**, 1–17 (2018)
2. Cisco Systems Inc.: Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2015–2020 (White Paper). http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html
3. Anagnostopoulos, C.N., Anagnostopoulos, I., Psoroulas, I.D., Loumos, V., Kayafas, E.: License plate recognition from still images and video sequences: a survey. IEEE Trans. Intell. Transp. Syst. **9**(3), 377–391 (2008)
4. Wiegand, T., Sullivan, G.J., Bjontegaard, G., Luthra, A.: Overview of the H. 264/AVC video coding standard. IEEE Trans. Circuits Syst. Video Technol. **13**(7), 560–576 (2003)
5. Sullivan, G.J., Ohm, J.R., Han, W.J., Wiegand, T., et al.: Overview of the high efficiency video coding (HEVC) standard. IEEE Trans. Circuits Syst. Video Technol. **22**(12), 1649–1668 (2012)
6. De Cock, J., Mavlankar, A., Moorthy, A., Aaron, A.: A large-scale video codec comparison of x264, x265 and libvpx for practical VOD applications, p. 997116, September 2016
7. Alliance for Open Media. http://aomedia.org
8. Versatile Video Coding (VVC). https://jvet.hhi.fraunhofer.de
9. Hojati, E., Franche, J., Coulombe, S., Vzquez, C.: Massively parallel rateconstrained motion estimation using multiple temporal predictors in HEVC. In: 2017 IEEE International Conference on Multimedia and Expo (ICME), pp. 43–48 (2017)
10. Zhao, L., Xu, J., Zhou, Y., Ai, M.: A dynamic slice control scheme for slice-parallel video encoding. In: 2012 19th IEEE International Conference on Image Processing, pp. 713–716 (2012)
11. Rodriguez, A., Gonzalez, A., Malumbres, M.P.: Hierarchical parallelization of an H. 264/AVC video encoder. In: International Symposium on Parallel Computing in Electrical Engineering (PARELEC 2006), pp. 363–368 (2006)
12. Geng, Y., Yang, Y., Cao, G.: Energy-efficient computation offloading for multicore-based mobile devices. In: 2018 IEEE Conference on Computer Communications (INFOCOM), pp. 46–54 (2018)
13. Chi, C.C., et al.: Parallel scalability and efficiency of HEVC parallelization approaches. IEEE Trans. Circuits Syst. Video Technol. **22**(12), 1827–1838 (2012)
14. Chandio, A.A., et al.: A comparative study on resource allocation and energy efficient job scheduling strategies in large-scale parallel computing systems. Cluster Comput. **17**, 1349–1367 (2014)
15. Bossen, F.: Common test conditions and software reference configurations. Document JCTVC-H1100, JCT-VC, San Jose, CA, February 2012
16. HM 16.15 Reference Software. http://hevc.hhi.fraunhofer.de
17. Pinol, P., Gomis, H.M., Lopez, O., Malumbres, M.P.: Slice-based parallel approach for HEVC encoder. J. Supercomput. **71**, 1882–1892 (2014)

18. Koziri, M., Papadopoulos, P., Tziritas, N., Dadaliaris, A.N., Loukopoulos, T., Khan, S.U.: Slice-based parallelization in HEVC encoding: realizing the potential through efficient load balancing. In: 2016 IEEE 18th International Workshop on Multimedia Signal Processing (MMSP), pp. 1–6 (2016)
19. Koziri, M., et al.: Heuristics for tile parallelism in HEVC. In: 2017 25th European Signal Processing Conference (EUSIPCO), pp. 1514–1518 (2017)
20. Mingozzi, A., Ricciardelli, S., Spadoni, M.: Partitioning a matrix to minimize the maximum cost. Discrete Appl. Math. **62**(1–3), 221–248 (1995)
21. Ahn, Y.J., Hwang, T.J., Sim, D.G., Han, W.J.: Implementation of fast HEVC encoder based on SIMD and data-level parallelism. EURASIP J. Image Video Process. **2014**(1), 16 (2014)
22. Shafique, M., Khan, M.U.K., Henkel, J.: Power efficient and workload balanced tiling for parallelized high efficiency video coding. In: 2014 IEEE International Conference on Image Processing (ICIP), pp. 1253–1257. IEEE (2014)
23. Malossi, G., Palomino, D., Diniz, C., Susin, A., Bampi, S.: Adjusting video tiling to available resources in a per-frame basis in high efficiency video coding. In: 2016 14th IEEE International New Circuits and Systems Conference (NEWCAS), pp. 1–4. IEEE (2016)
24. Papadopoulos, P.K., Koziri, M., Loukopoulos, T.: A fast heuristic for tile partitioning and processor assignment in HEVC. In: 2018 25th IEEE International Conference on Image Processing (ICIP), pp. 4143–4147 (2018)