A FAST HEURISTIC FOR TILE PARTITIONING AND PROCESSOR ASSIGNMENT IN HEVC

Panos K. Papadopoulos, Dept. of Computer Science and Biomedical Informatics, Univ. of Thessaly

Maria Koziri, Dept. of Computer Science, Univ. of Thessaly

Thanasis Loukopoulos, Dept. of Computer Science and Biomedical Informatics, Univ. of Thessaly

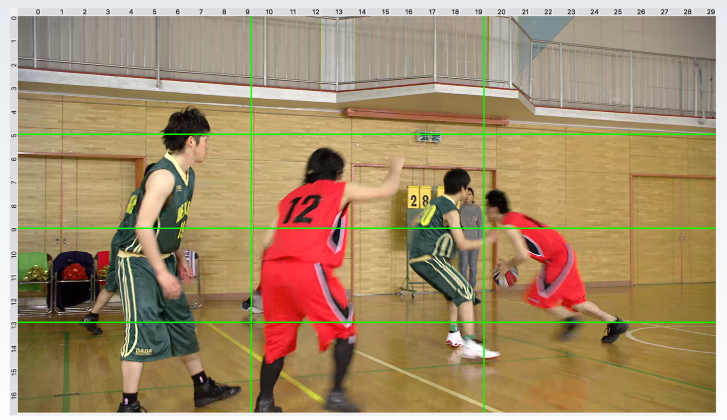
MOTIVATION

- HEVC achieves high compression efficiency, at the cost of computationally complexity.
- Solution: Parallelization. HEVC offers different primitives: Slices, Tiles, Wavefront.
- Most works in tile parallelism assume a one on one tile to thread and thread to processor assignment. With this assumption they aim at load balancing processors by balancing tile sizes.
- *What happens if the number of processors is less than the number of tiles?*

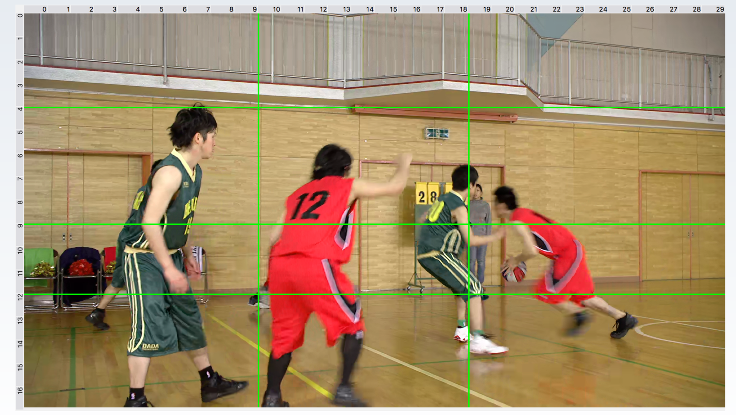
GOAL

- Assume a one on one thread to processor assignment but assign multiple tiles per thread, in order to balance processor load.
- Resize tiles and perform scheduling upon each frame.
- FAST (Fast Adaptive Scheduling of Tiles) algorithm offers solution to the combined problem of both tile partitioning in an adaptive manner and scheduling the resulting tiles to the available processors.

EXAMPLE OF TILE PARTITIONING



Static algorithm



FAST algorithm

- Each frame starts with a uniform $M \times N$ tile grid (M and N are part of the input). In Static Algorithm the uniform tile partitioning does not change.
- In FAST algorithm, tile boundaries are adapted so as to balance processor load.

PRELIMINARIES

- Load balancing
  - The compression time of a tile is the aggregation of the compression times of the CTUs it contains.
  - At the start of each frame the expected CTU compression time is estimated using previously recorded times. The LDE method (LowDelay Estimator) presented in (Koziri et al., Eusipco17) is used for the estimation.
- Scheduling
  - The load of each processor is the aggregate cost of the tiles assigned to it.
  - Processor assignment is performed using the *MaxMin* approach (Brown et al., JPDC01).
  - The heaviest tile (Max) is assigned to the least loaded processor (Min) in an iterative fashion until all tiles are assigned.

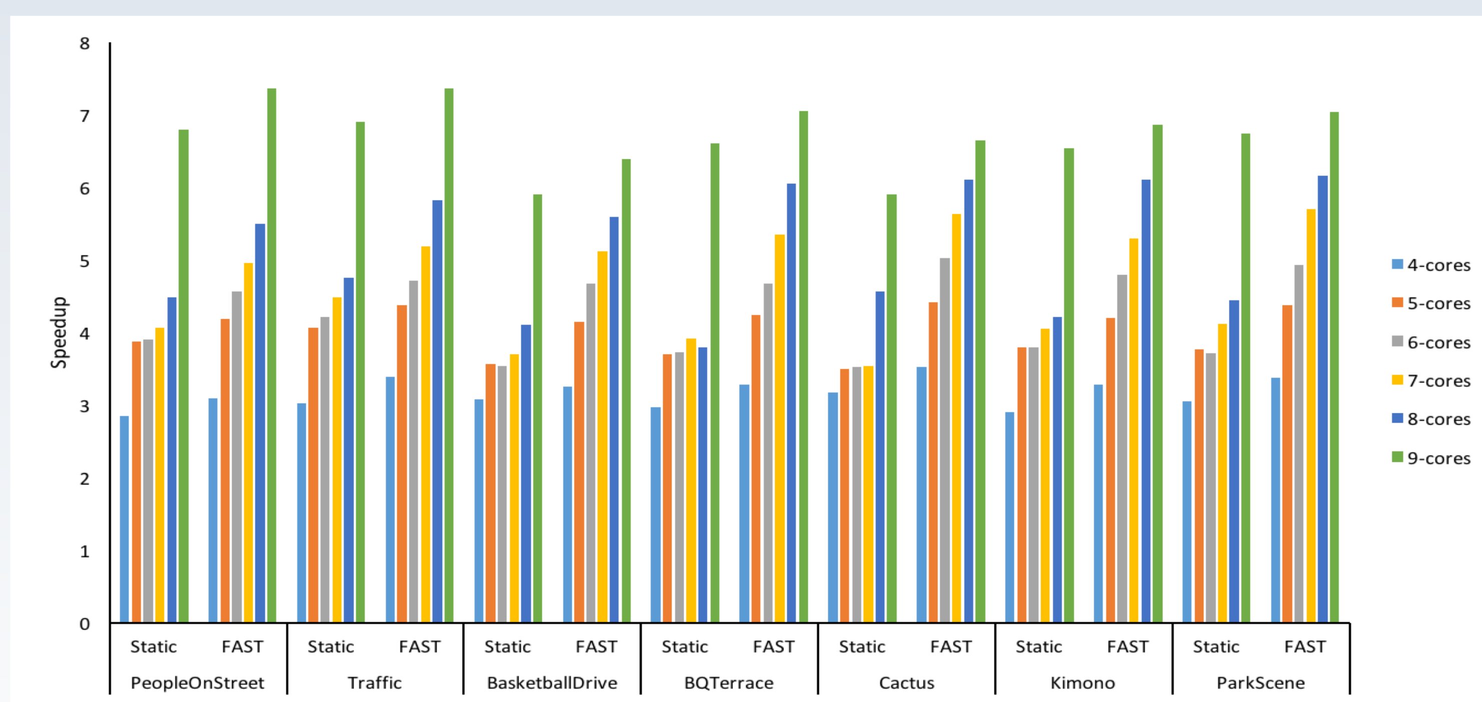
FAST ALGORITHM

Algorithm 1 FAST Pseudocode

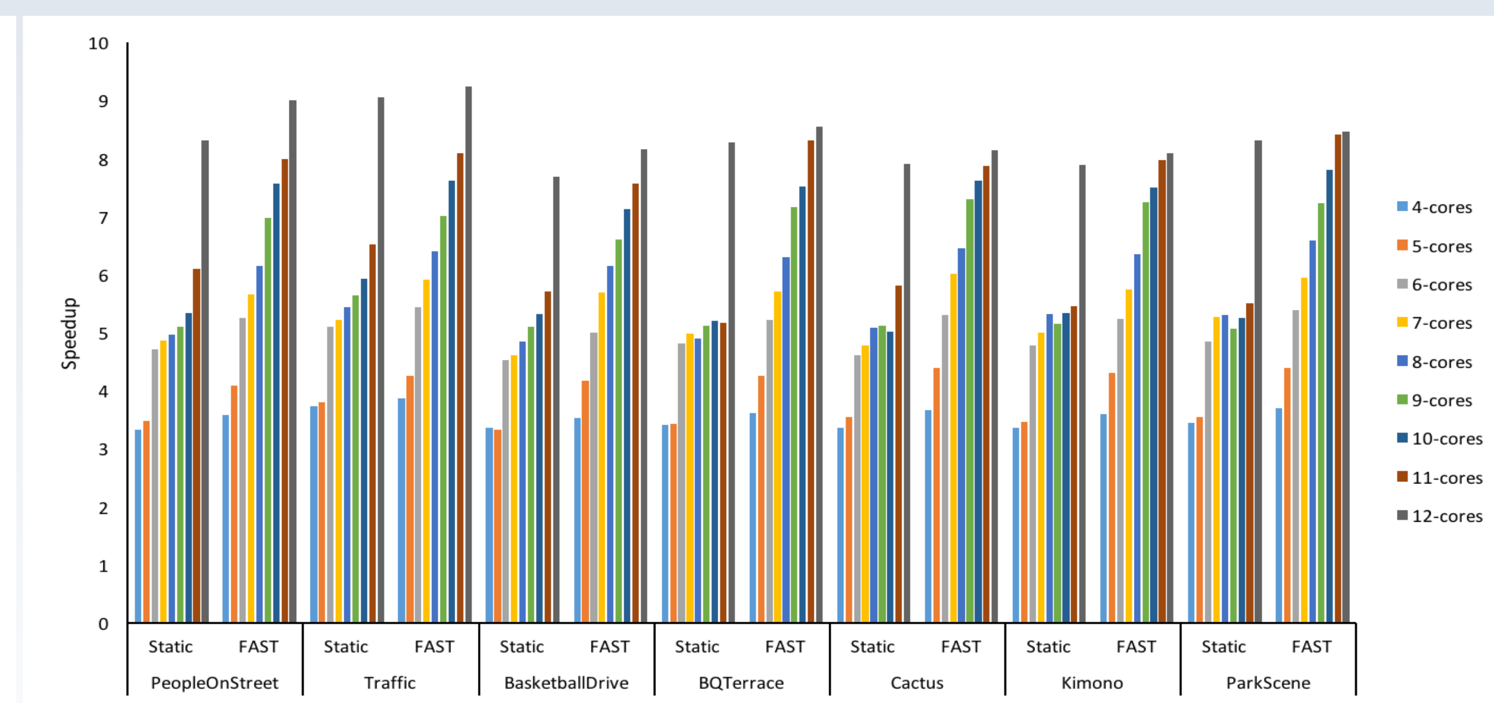
```
bestPartition ← Uniform
bestSol ← MaxMin(tilePartition)
found ← 1
while found do
    found ← 0
    candidateSol ← ∞
    candidatePartitions ← NULL
    candidateTiles ← heavyProcessorTiles
    for x ∈ candidateTiles do
        candidatePartitions ∪= {x shrinkings}
    end for
    for r ∈ candidatePartitions do
        candidateSol ← MaxMin(r)
        if candidateSol < bestSol then
            update bestSol, bestPartition
            found ← 1
        end if
    end for
end while
Implement(bestPartition)
```

- FAST algorithm starts with a uniform $M \times N$ tile grid.
- *MaxMin* allocates tiles to processors.
- Tiles at the most loaded processor change boundaries (by one CTU row or column) in an attempt to reduce its load.
- The new tile partitioning is reassigned to processors with *MaxMin*. If it reduces max processor load it is kept as candidate.
- The best candidate solution (the one with least max processor load) is selected and the whole process iterates until no further load reduction is possible.

EXPERIMENTS



Speedup for 3×3 tile partitioning



Speedup for 4×3 tile partitioning

SETUP

- Linux Server with two 12-core Intel Xeon E5-2650 running at 2.20GHz
- Class A and B test sequences
- Software: HM 16.15 + OpenMP
- Encoding parameters: QP 32, bit depth 8, CTU 64×64, max depth 4, TZ search

RESULTS

- FAST outperforms Static with large improvement in cases where the number of available processors is not a divisor of the number of tiles.
- Impact on video coding efficiency is negligible:
  - 3×3 tile partitioning: average PSNR difference of 0.016 dB, average Bitrate increase of less than 2%.
  - 4×3 tile partitioning: average PSNR difference of 0.002 dB, average Bitrate increase of less than 0.1%.
- The overhead of the algorithm is negligible (less than msec per frame).

CONCLUSIONS

- FAST algorithm decides tile sizing and processor assignment in an adaptive per frame fashion regardless of the number of processors available.
- Performance evaluation indicated a reduction in encoding time that reached 37% compared to static uniform tile partitioning.
- Experimental results indicate that FAST algorithm has negligible impact to coding efficiency.