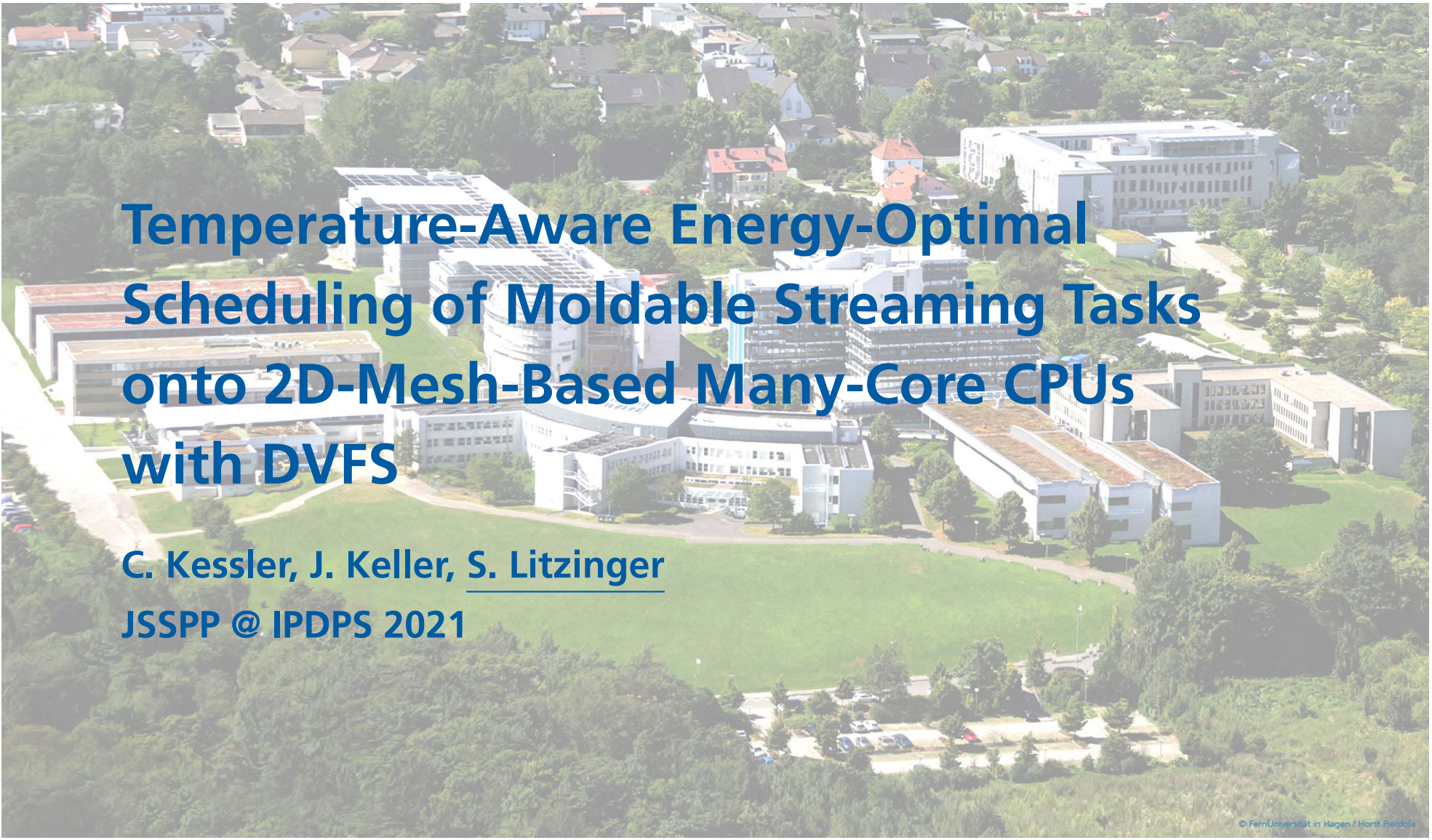


Temperature-Aware Energy-Optimal Scheduling of Moldable Streaming Tasks onto 2D-Mesh-Based Many-Core CPUs with DVFS

C. Kessler, J. Keller, S. Litzinger

JSSPP @ IPDPS 2021

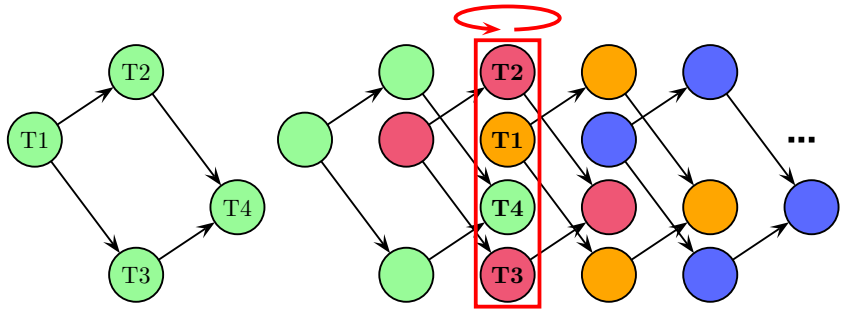
Parallelism and VLSI Research Group
Sebastian Litzinger

Overview

- Motivation
- Background
- Temperature-aware crown scheduling with core buddying
- Experimental results
- Summary

Motivation

- Chip temperature increasingly becomes a concern
- Negative impact on power consumption, durability of the hardware
- OS governor has no knowledge about application
- Heat diffusion must be factored in
- Streaming computations:



- Minimum data rate imposes deadline on single round
- Parallelization, DVFS in static scheduling
- Minimization of power consumption

Background: architecture and application

Architecture

- Generic multi-/manycore architecture with 2D mesh core layout
- Upper limit for core temperature
- Core-individual DVFS
- No specific assumptions regarding power function
- Cores are considered ‘hot’ or ‘cold’, bearing implications on core power consumption
- Heat flow to neighboring cores is modeled via discretized and linearized set of equations

Background: architecture and application

Architecture

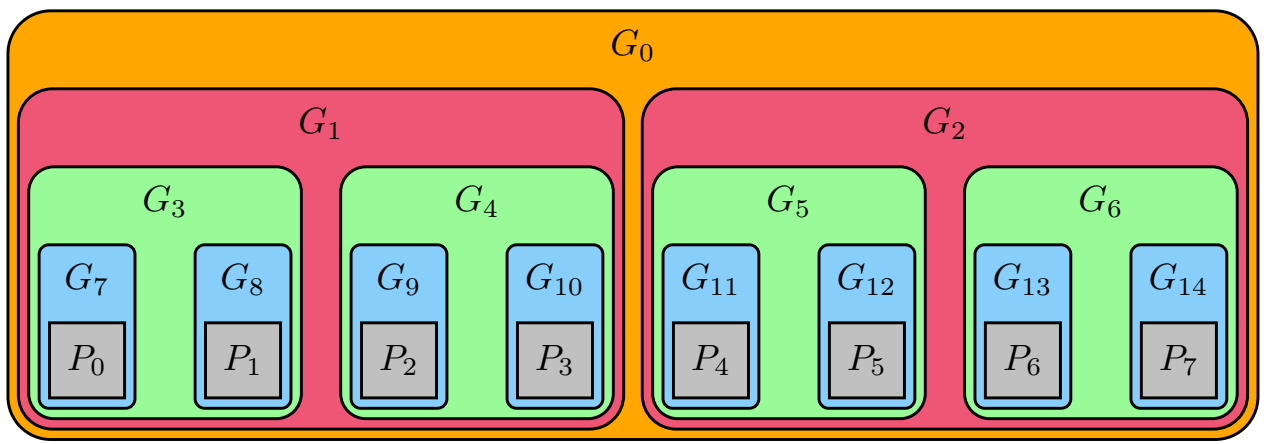
- Generic multi-/manycore architecture with 2D mesh core layout
- Upper limit for core temperature
- Core-individual DVFS
- No specific assumptions regarding power function
- Cores are considered ‘hot’ or ‘cold’, bearing implications on core power consumption
- Heat flow to neighboring cores is modeled via discretized and linearized set of equations

Application

- Streaming task graph with multi-variant tasks (e. g. different algorithms, parameters)
- Each variant can be sequential, fully or partially moldable
- Scalability modeled via parallel efficiency values for variants and core counts

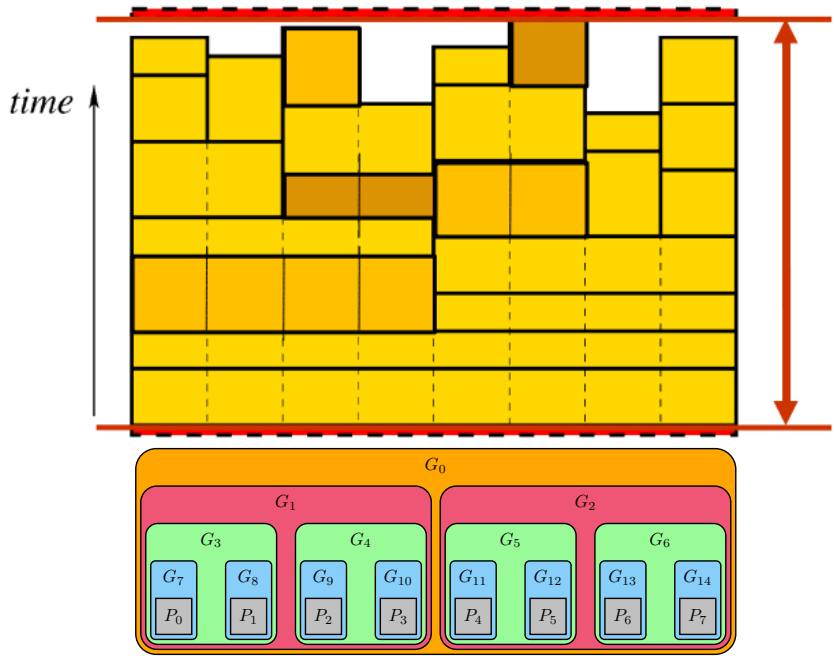
Background: scheduling

- Scheduling decisions for each task: variant, width, core operating frequency
- Deadline and heat constraints must be respected
- Minimize overall energy consumption
- Base: crown scheduling, map tasks to core groups, execute in order of descending width



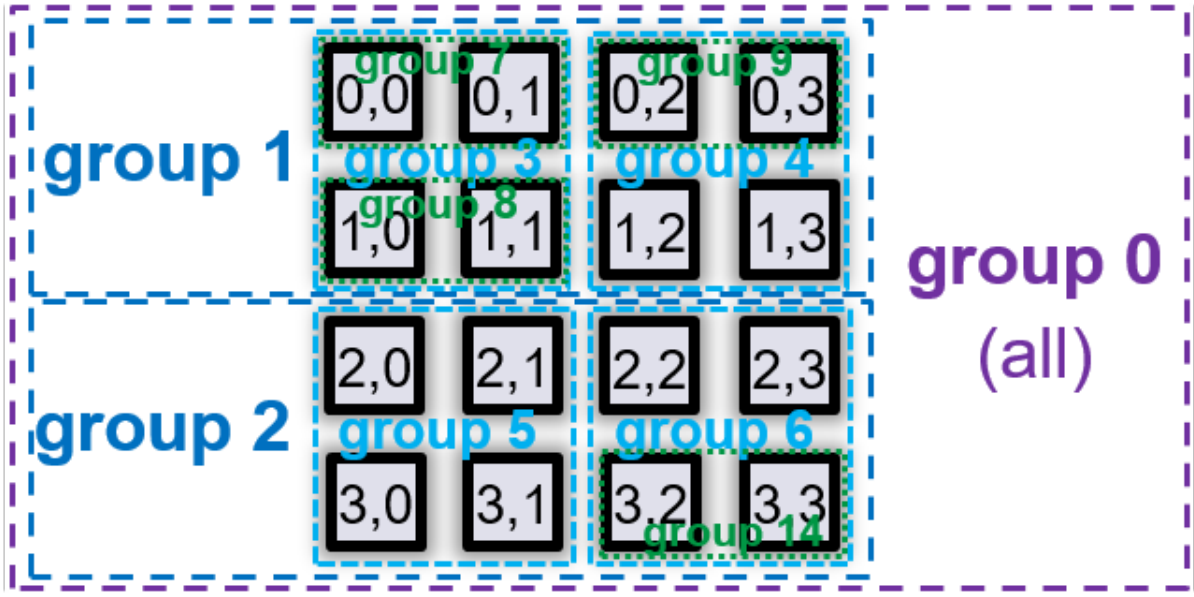
Background: scheduling

- Scheduling decisions for each task: variant, width, core operating frequency
- Deadline and heat constraints must be respected
- Minimize overall energy consumption
- Base: crown scheduling, map tasks to core groups, execute in order of descending width



Background: scheduling

- Scheduling decisions for each task: variant, width, core operating frequency
- Deadline and heat constraints must be respected
- Minimize overall energy consumption
- Base: crown scheduling, map tasks to core groups, execute in order of descending width



Background: scheduling

- Scheduling decisions for each task: variant, width, core operating frequency
- Deadline and heat constraints must be respected
- Minimize overall energy consumption
- Base: crown scheduling, map tasks to core groups, execute in order of descending width
- Extension to temperature-aware scheduling
- ILP vs. heuristic approach

Temperature-aware crown scheduling with buddying: basics

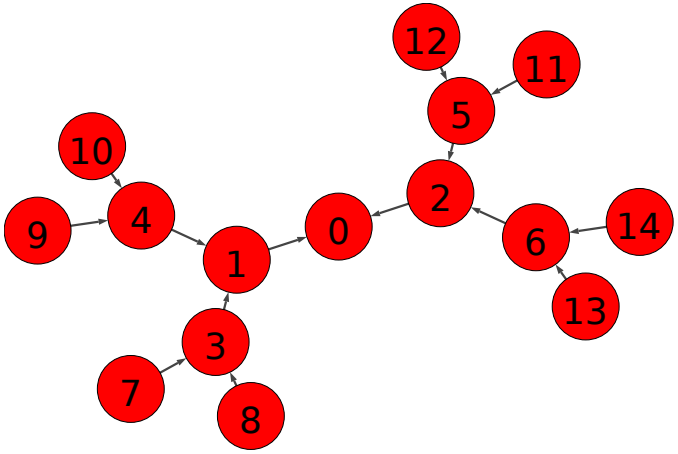
- Some tasks may produce a lot of heat, e. g. large sequential tasks, which must be run at high frequencies
- Such ‘hot’ tasks can lead to heat constraint violations
- Solution: map hot tasks to two core sets and alternate execution between these from round to round
- Integrate decision on additional core set mapping into scheduling to preserve solution quality
- Implementation:
  - map tasks to core group twice the size
  - for each core in that group, define (physically neighboring) buddy core
  - for each task, add new variant for buddy core execution
- Every core executes all of its ‘cool’ tasks each round, and its hot tasks every second round
- Shared caches should minimize switching penalty

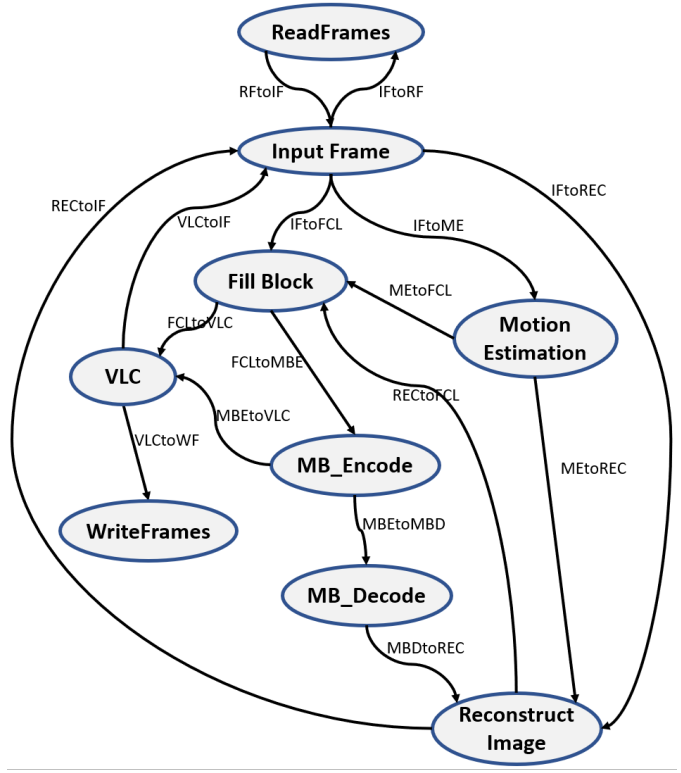
Temperature-aware crown scheduling with buddying: generalization

- Provide more flexibility to the scheduler by allowing arbitrary buddy cores
- Buddy group does not have to be same size or physically neighboring
- Penalty may be applied to account for cache misses or data movement
- Length of buddy core idle phase does not have to reflect task's duration
- ILP must now consider two subsequent rounds at once

Experimental setup

- Two applications: mergesort and H.263 encode from the Dataflow Benchmark Suite (DFbench)



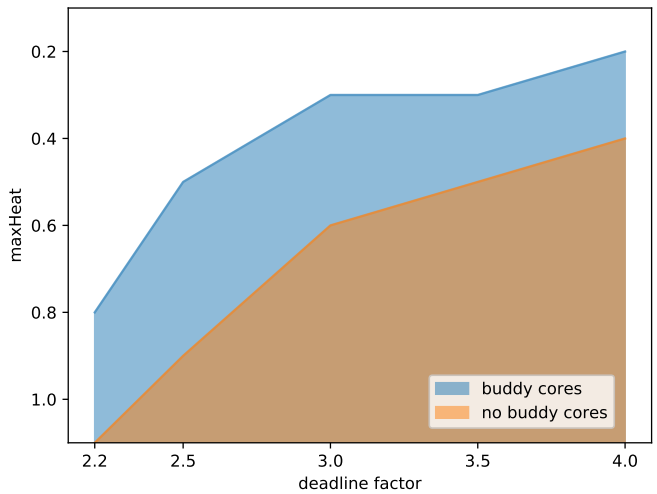


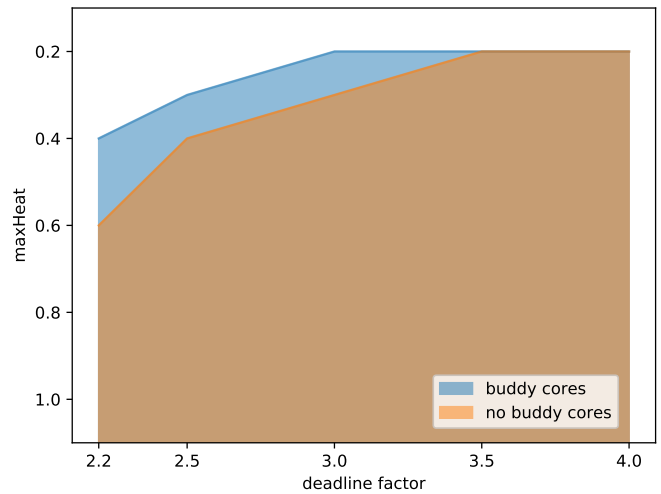
Experimental setup

- Two applications: mergesort and H.263 encode from the Dataflow Benchmark Suite (DFbench)
- Vary deadline tightness and heat limit
- Compute schedule for each parameter setting (45 combinations in total) via ILP or heuristic
- What is the effect of buddy cores in terms of feasible solutions?
- Heat diffusion, core-local heat only
- Multicore processor with 16 cores and power characteristics of ARM big.LITTLE (big cores)
- Gurobi 8.1.0 solver for ILPs, gurobipy
- AMD Ryzen 7 2700X, 8 cores, SMT

Experimental results

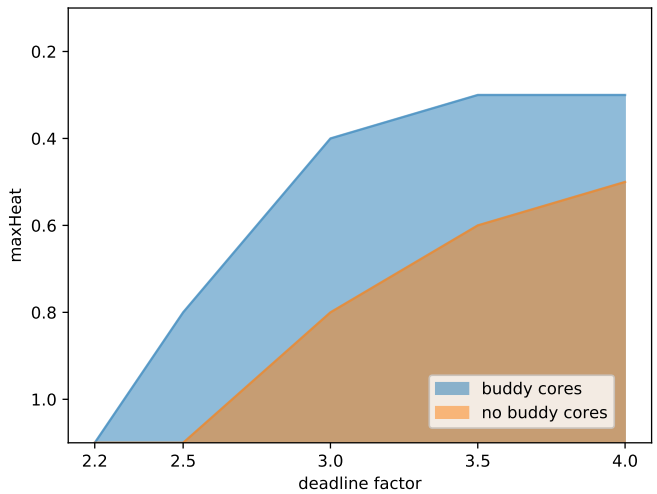
Feasible solutions for mergesort task set, left: core-local heat only, right: heat diffusion to neighboring cores

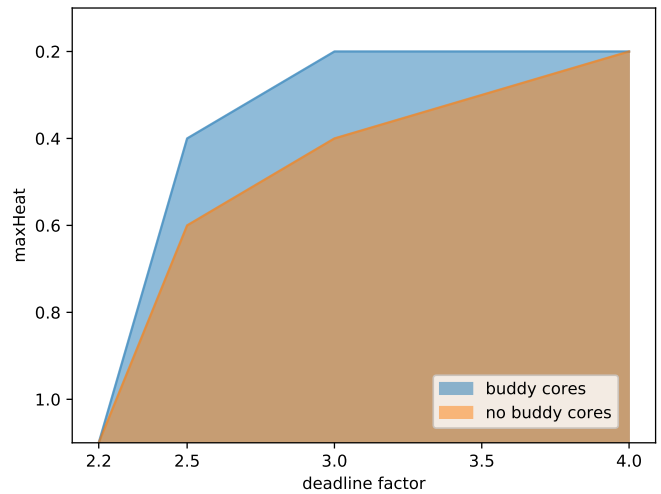




Experimental results

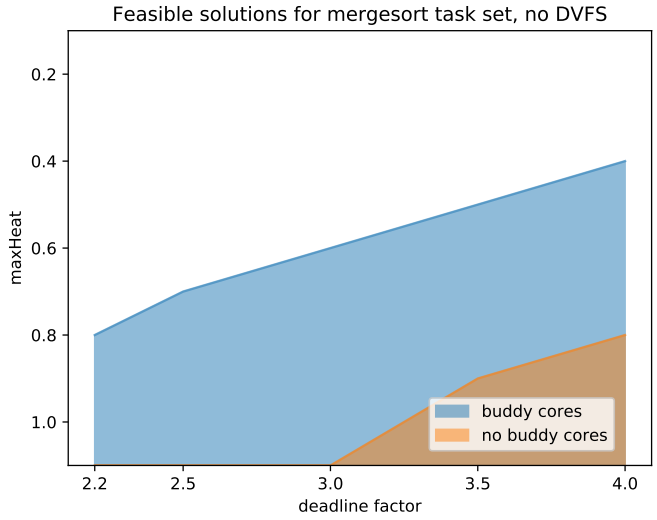
Feasible solutions for H.263 encode task set, left: core-local heat only, right: heat diffusion to neighboring cores

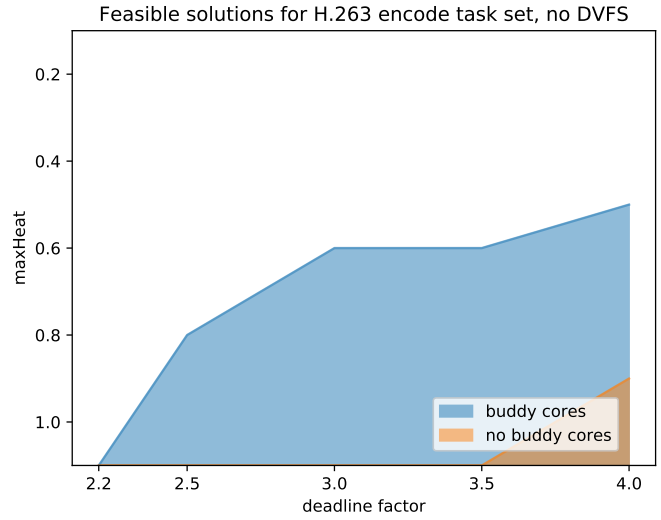




Experimental results

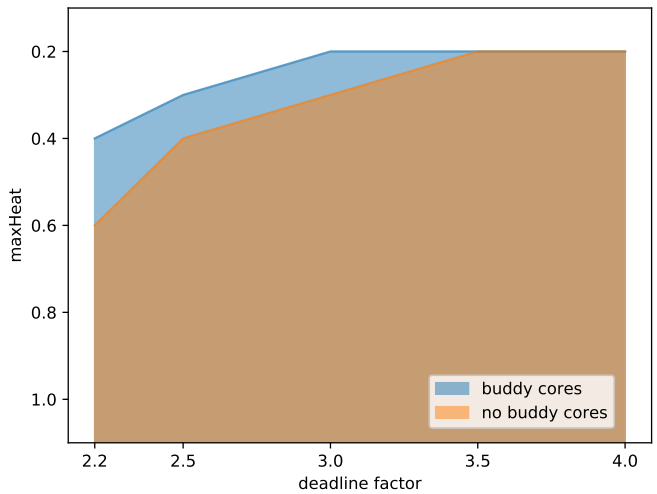
Feasible solutions without DVFS, left: mergesort task set, right: H.263 encode task set

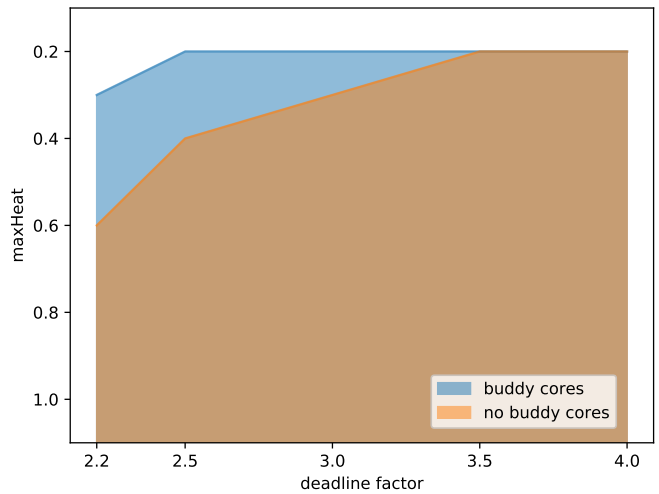




Experimental results

Feasible solutions for mergesort task set and heat diffusion to neighboring cores, left: fixed buddies, right: flexible buddy selection





Summary

- Energy-efficient scheduling of streaming task applications onto many-core CPUs with 2D mesh geometry
- Deadline and temperature constraints
- Temperature-aware scheduling with buddying technique extends the parameter range for which feasible solutions can be produced
- Heuristic approach can serve cases where static scheduling is not viable
- Next steps:
  - More applications
  - Validation of theoretical results on real system
  - Buddy core concept for heterogeneous architectures