DAHL: Energy-Efficient Machine Learning on Heterogeneous CPU using Task-based Programming

Andrew Mary Huet de Barochez
Stéphan Plassart
Sébastien Monnet

LISTIC – Université Savoie-Mont-Blanc

July 6, 2026

Outline

How about AI ecological impact?

Literature:

- Electricity consumption [Tripp et al. 2024]
- Carbon emissions [Patterson et al. 2021]
- Water usage [Li et al. 2025]
- Resource depletion [Berthelot et al. 2024; Falk et al. 2025]

→ Here we focus on **electricity consumption**

Problem: growing hardware heterogeneity

Heterogeneity at different levels:

- Computing units (CPU, GPU, TPU, FPGA...)
- Network speed in distributed setups
- Memory hierarchies, bandwidth differences

→ **Frugal** usage of machine's resources is **harder**

Reusing what we have?

- Device use phase = **40%** of carbon emissions in its life [Brilland et al. 2025]
  → Buying new machines lead to strong impacts
- Existing machines may be heterogeneous, we have to use them **efficiently**
  → We first **focus on CPUs**, as they are widespread
- In the long-term we want to study more heterogeneous contexts (CPU+GPU, distributed)

Heterogeneous CPU topologies

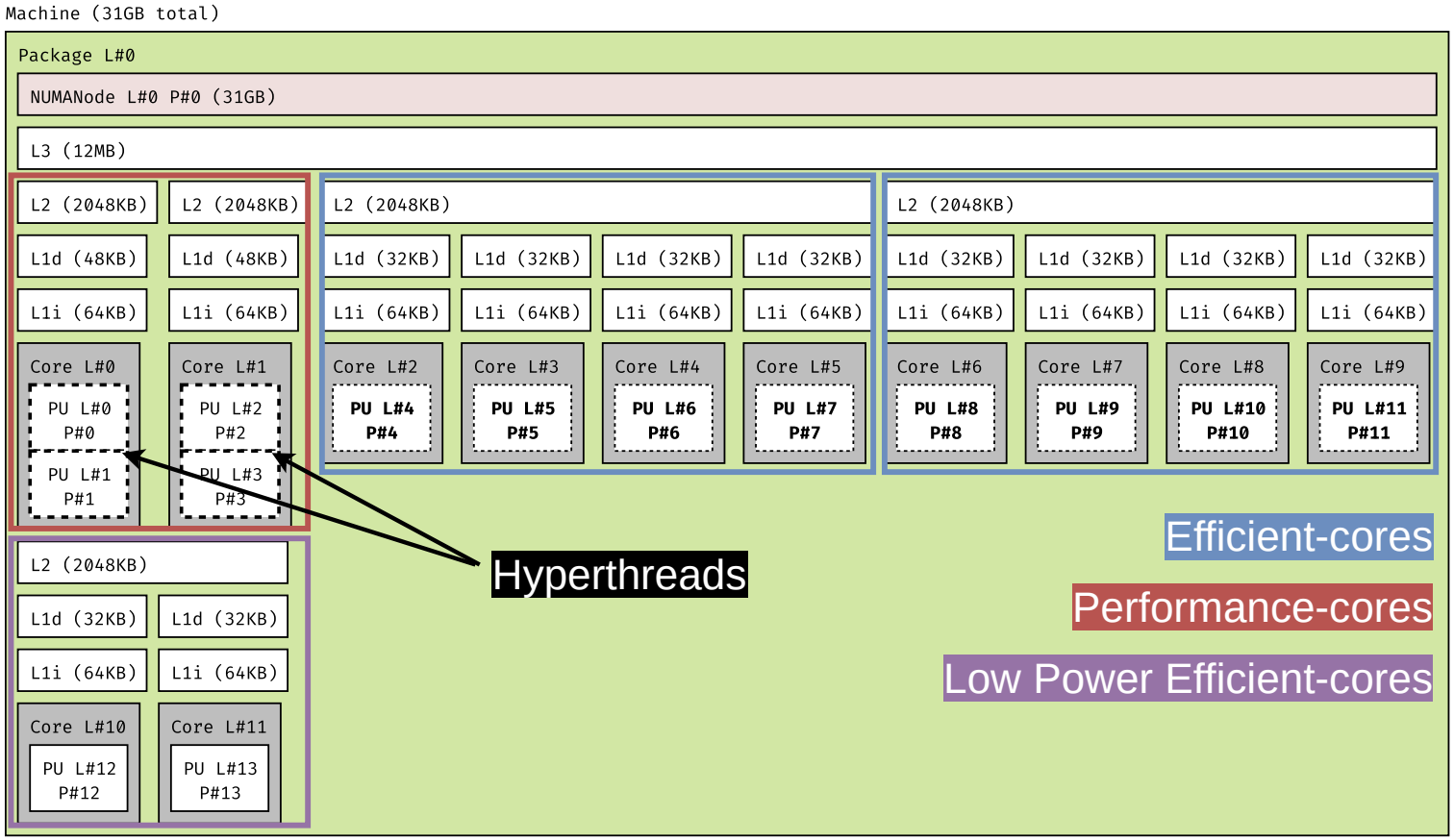


Figure: Intel® Core™ Ultra 5 Processor 125U topology (Q4'23).

Topology aware scheduling?

Task Graph Computing System (TGCS) can **dynamically dispatch** tasks based on **hardware** at runtime

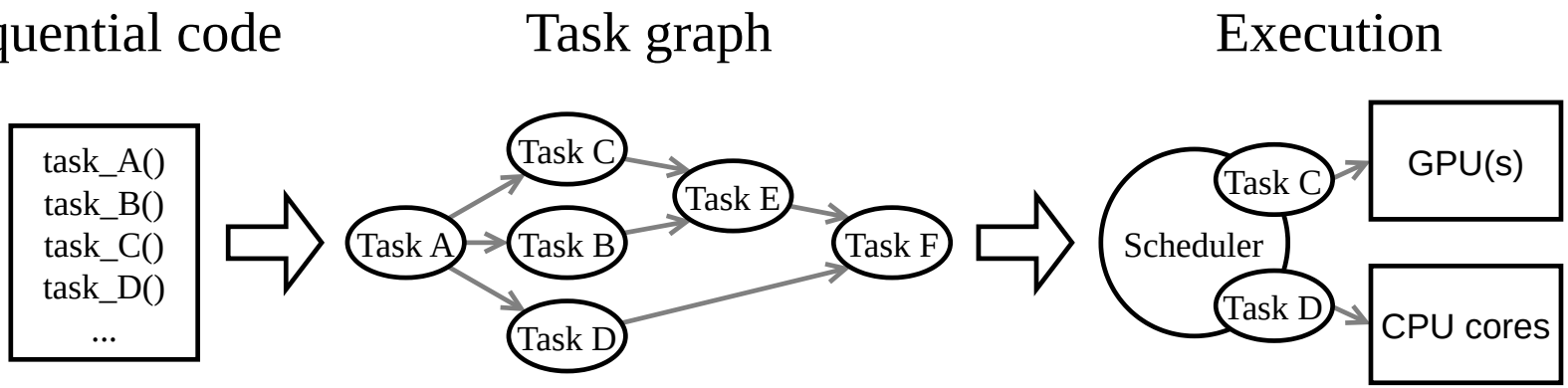
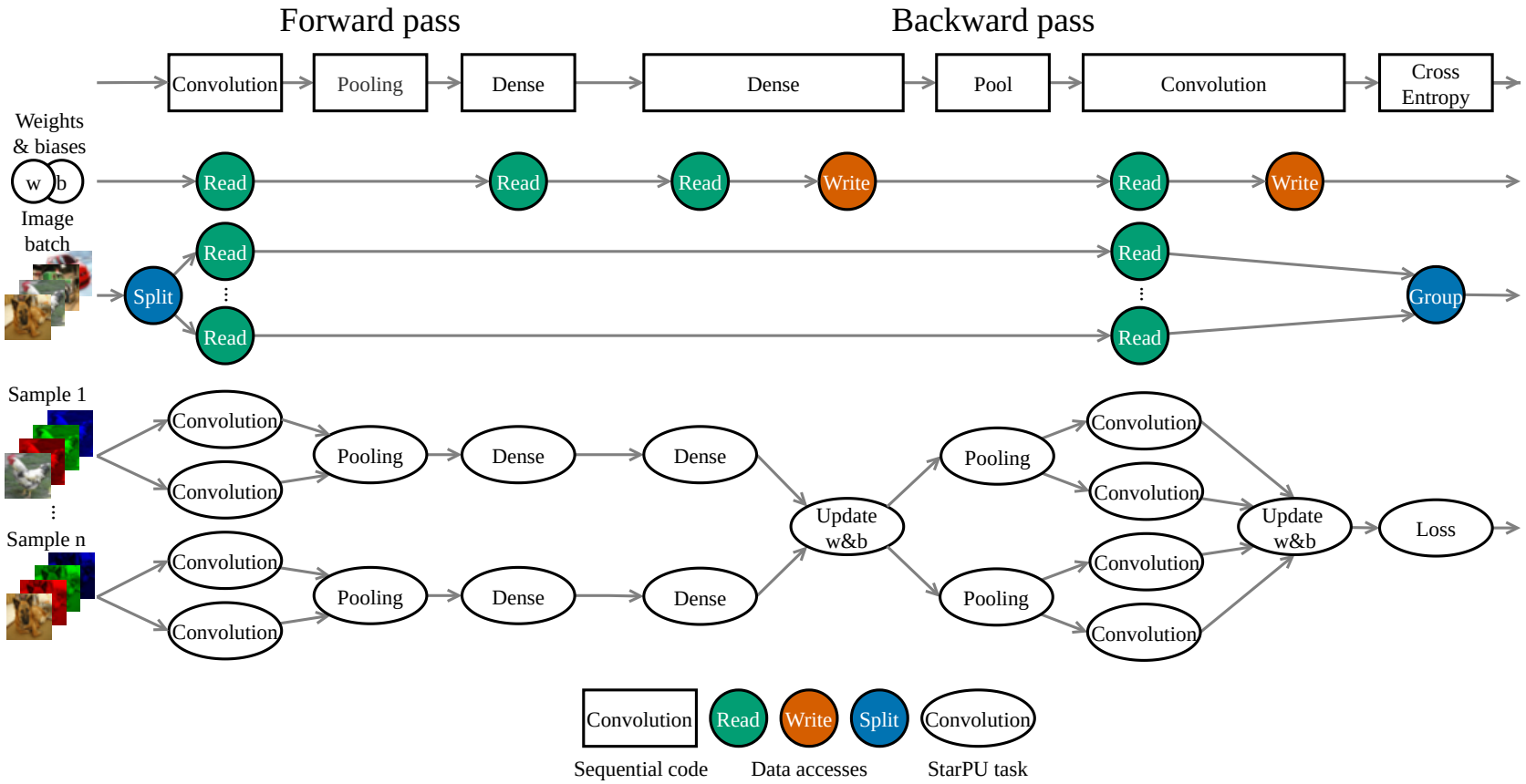


Figure: Basic principle of a TGCS

How the task graph is infered?



Our proof-of-concept: DAHL

Distributed, Arbitrary, and Heterogeneous Learning

What is DAHL?

- TGCS-based machine **Learning** framework
- **Arbitrary** implement any parallelization
- Supports **Heterogeneous** CPU topologies
- **Distributed** and GPU support (WIP)

Goals:

- Study **energy-efficiency** in heterogeneous contexts
- Explore **software optimizations**

Methodology: platform

Table: CPU specifications of machines used in experiments.

Machine	CPU	#Cores	#Threads	Frequency
Rammus	i7-14700K	8 P-cores 12 E-cores	28	5.6 GHz
Senna	125U	2 P-cores 8 E-cores 2 LP E-cores	14	4.3 GHz
Malphite	E3-1240V6	4	8	4.1 GHz
Poppy	i5-7500	4	4	3.8 GHz

Methodology: use case

Image classification with a Convolutional Neural Network (CNN)

High definition dataset similar to Fashion-MNIST, **scaled down** linearly to multiple sizes

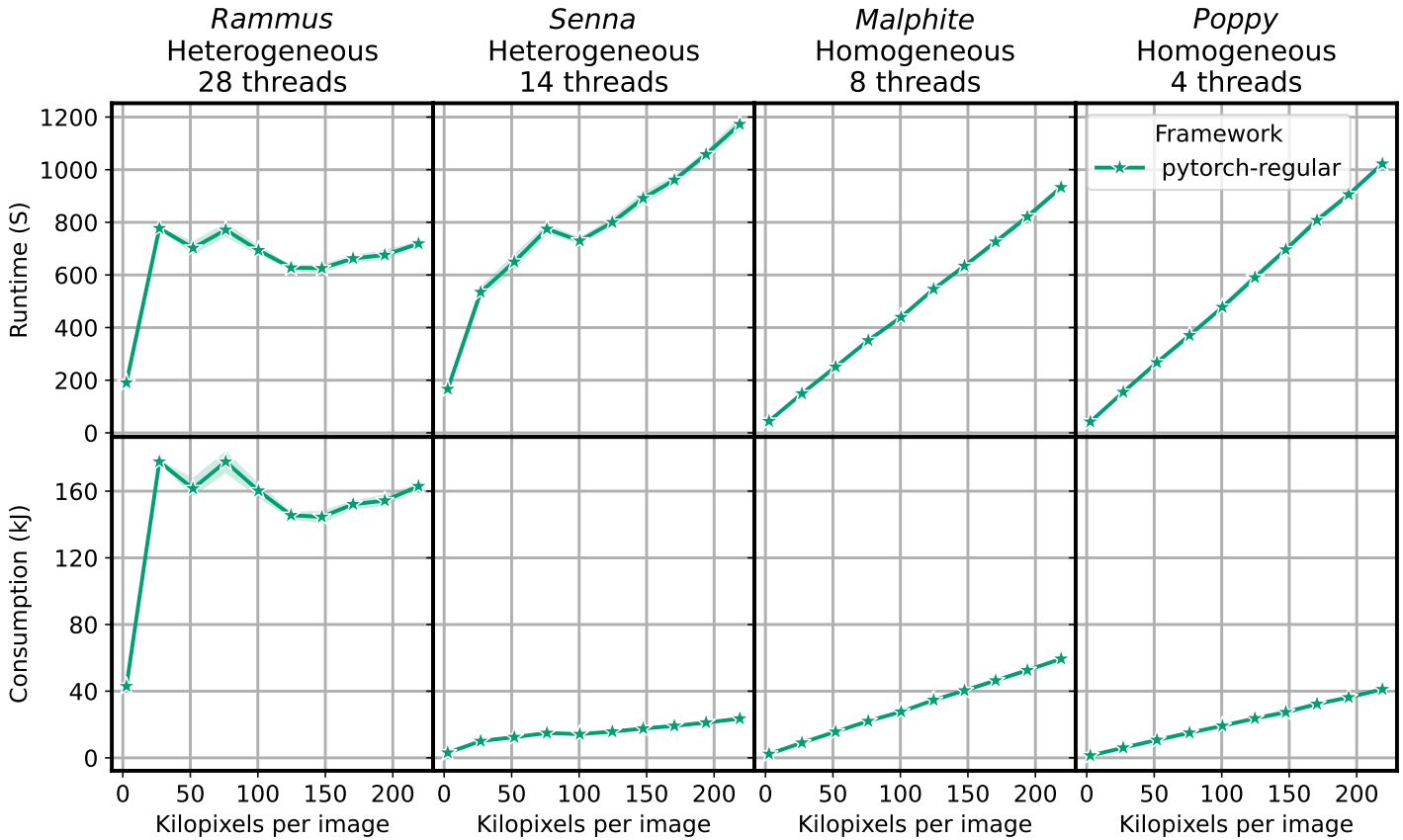
Figure: Example of scaled down images replicated to mulitple datasets

Methodology: reproducibility

- Energy: **wattmeters** + Intel **RAPL**
  Machines only used for experiments
- Thermal biases: **random experiment order** + warmup + air-conditioned room
- Confidence intervals: 5 runs per experiment
- Software: NixOS to **replicate** configuration on all machines

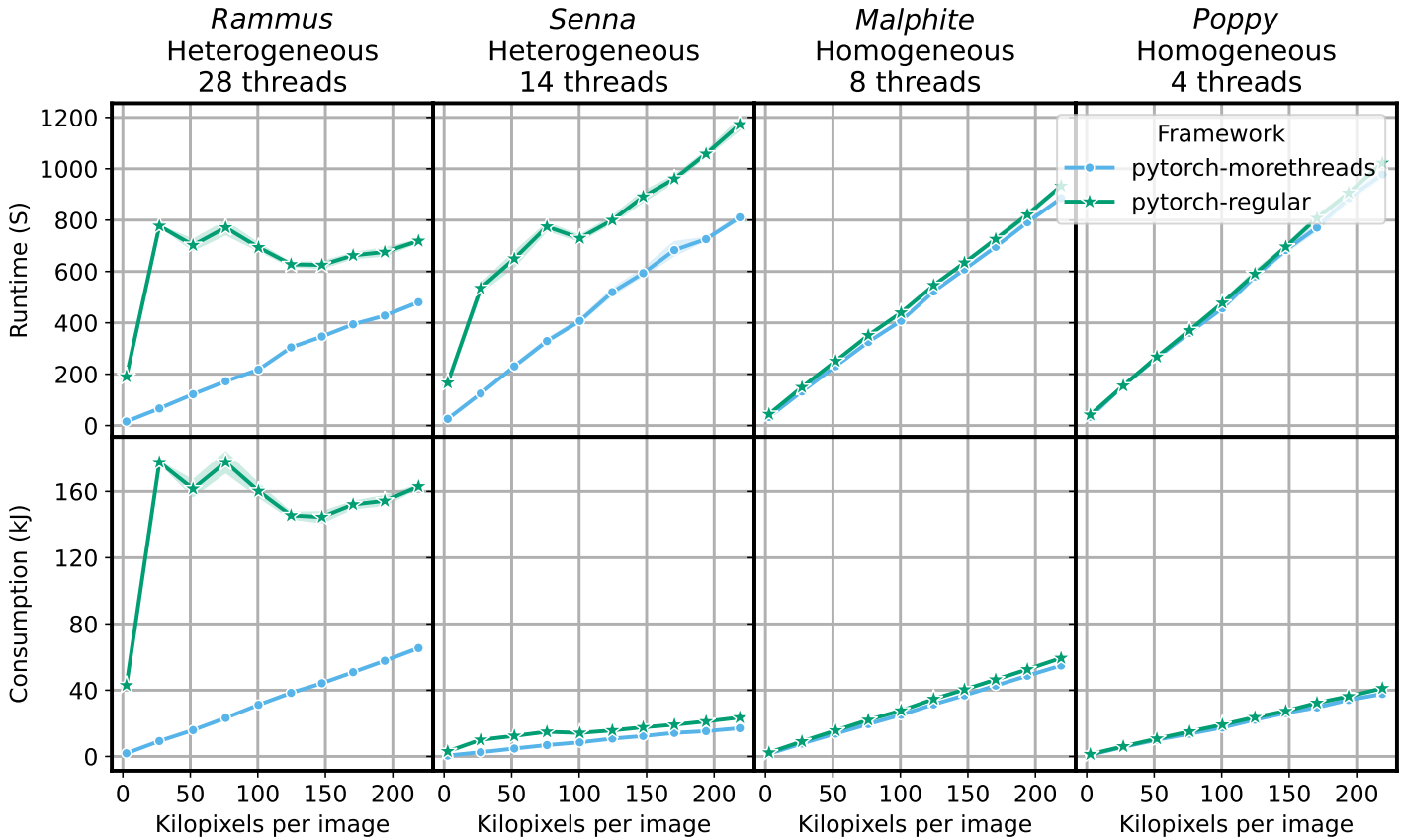
How Pytorch adapts to heterogeneity?

- Default Pytorch config bad when hetero-geneous



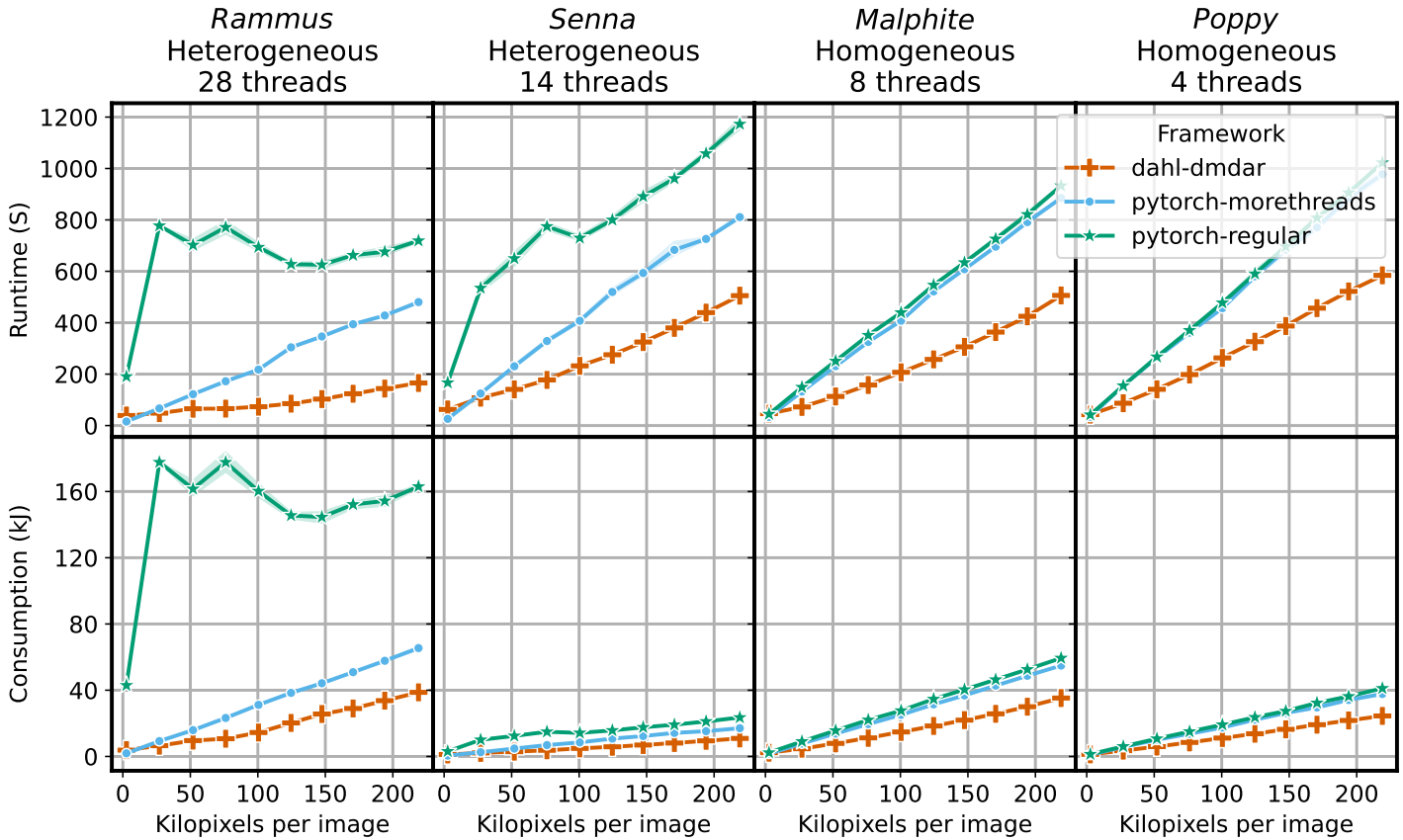
How Pytorch adapts to heterogeneity?

- Default Pytorch config bad when hetero-geneous
- Increasing threads helps



How Pytorch adapts to heterogeneity?

- Default Pytorch config bad when hetero-geneous
- Increasing threads helps
- DAHL shows better performances overall

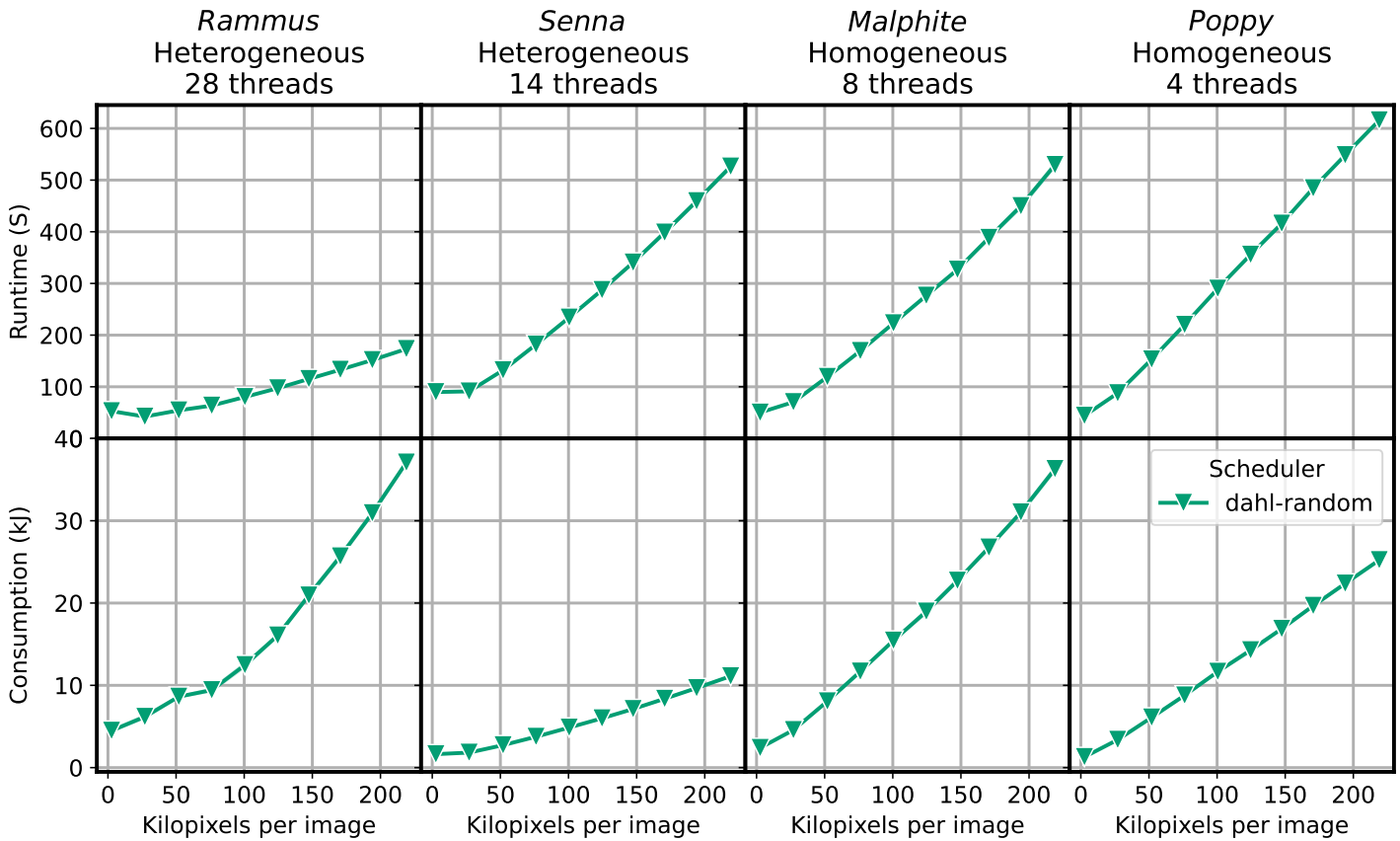


Comparing StarPU schedulers

- `Random`: schedules tasks **randomly**
- `Eager`: workers pulls tasks from a **central queue**
- `Peager`: same but **performance model-based**
- `WS`: workers can **steal** from other queues
- `LWS`: **locality-aware** work stealing
- `DMDAR` (deque model data aware ready): find **optimal task placement** with **performance model** informations: runtime, data transfer time, and readiness.

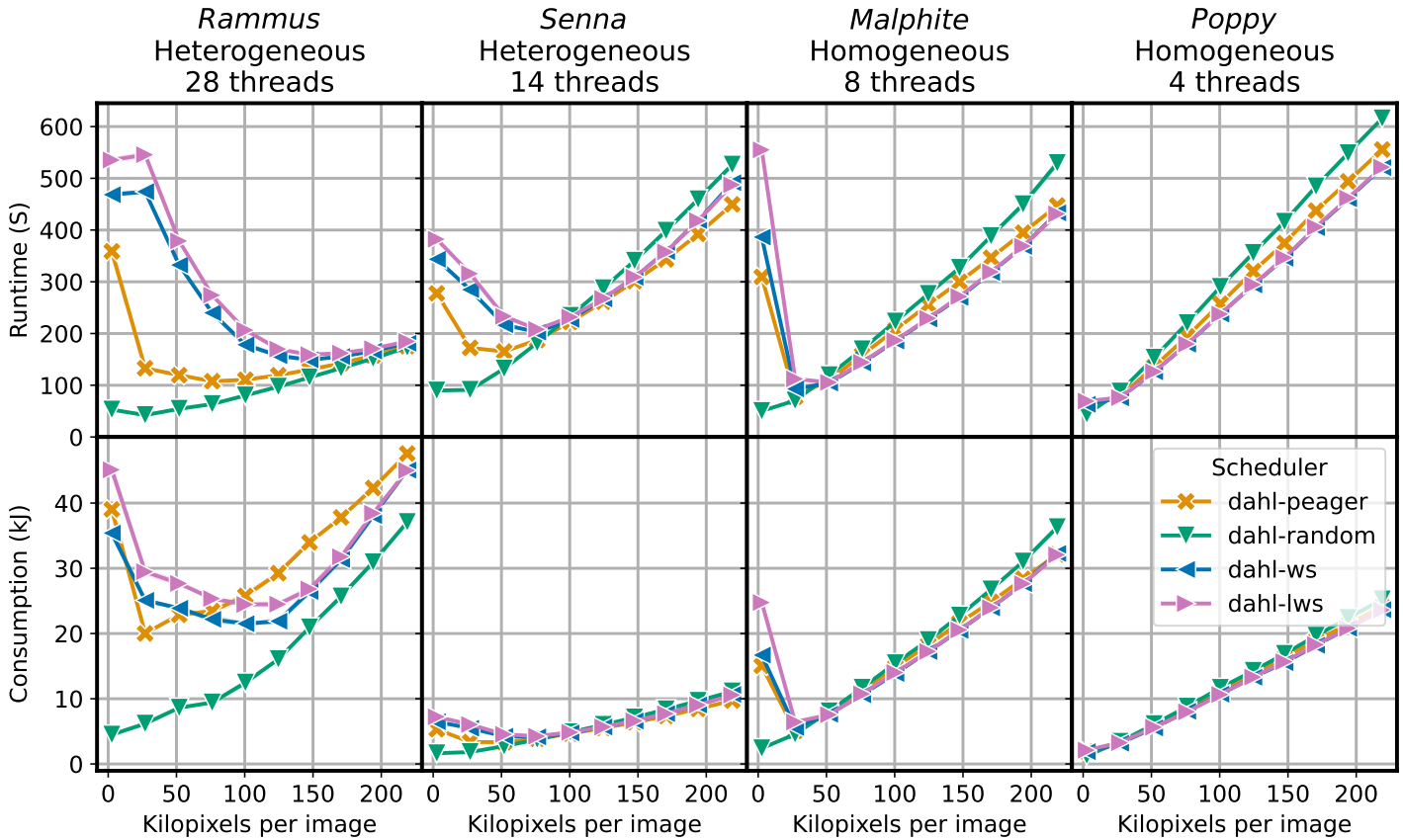
Which scheduler to use?

`Random` **scales well**, no matter **topology** or **image size**



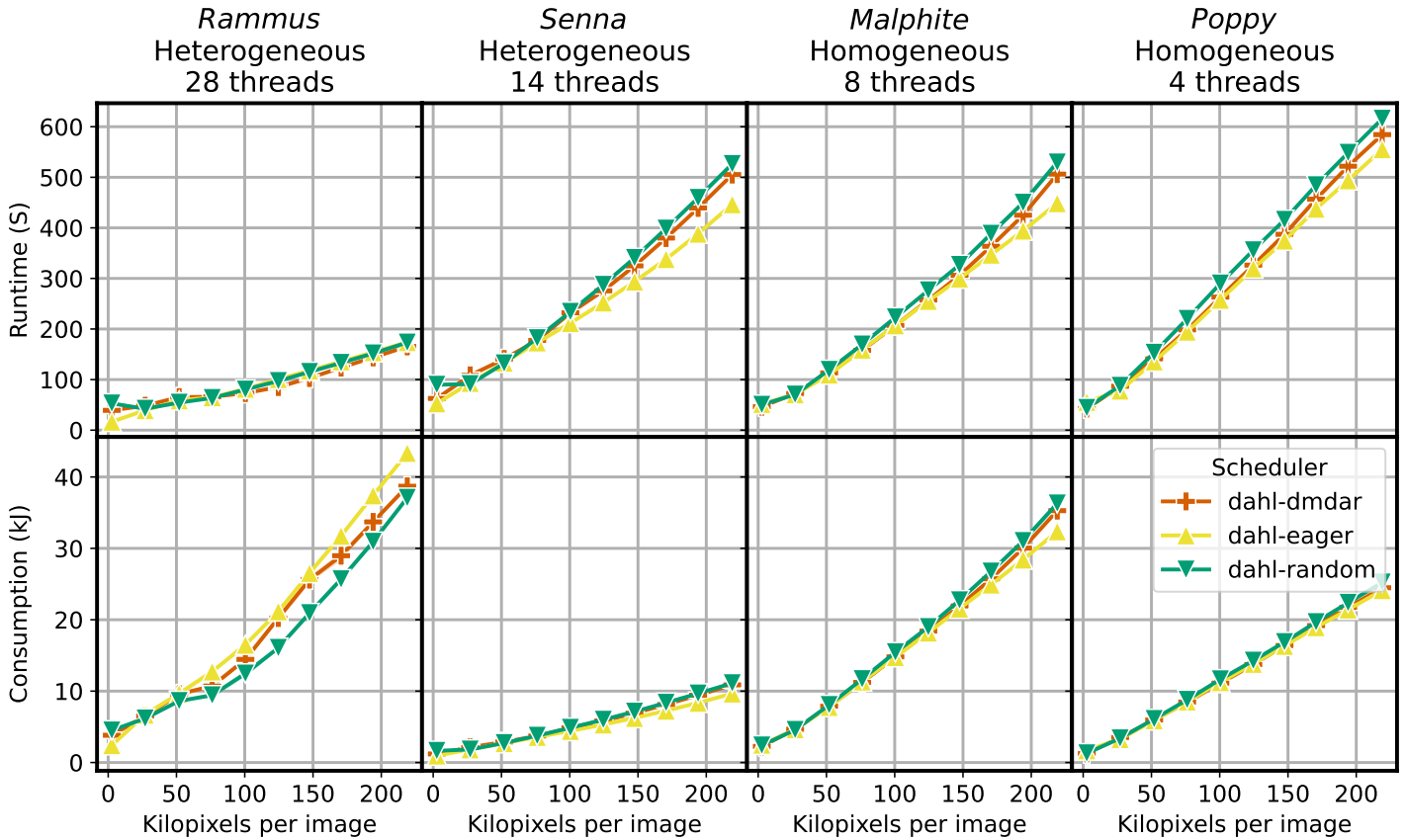
Which scheduler to use?

WS, LWS and Peager scales bad on small images + heterogeneous CPU



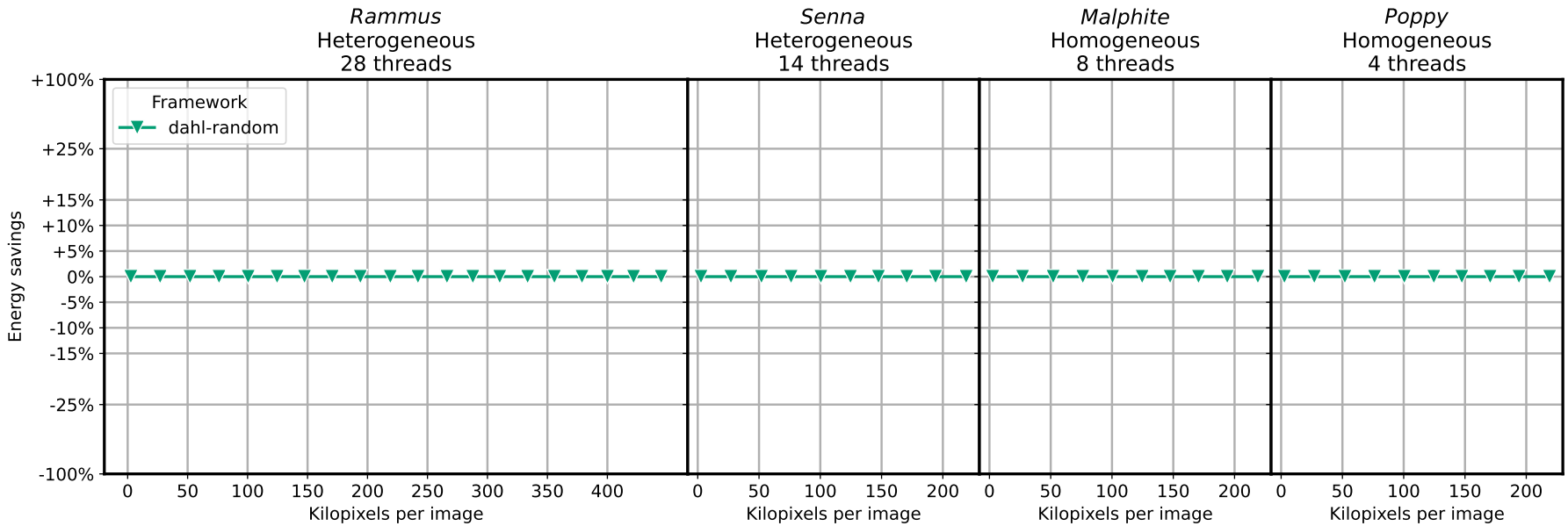
Which scheduler to use?

Eager and DMDAR provides **better performances** regardless of topology!



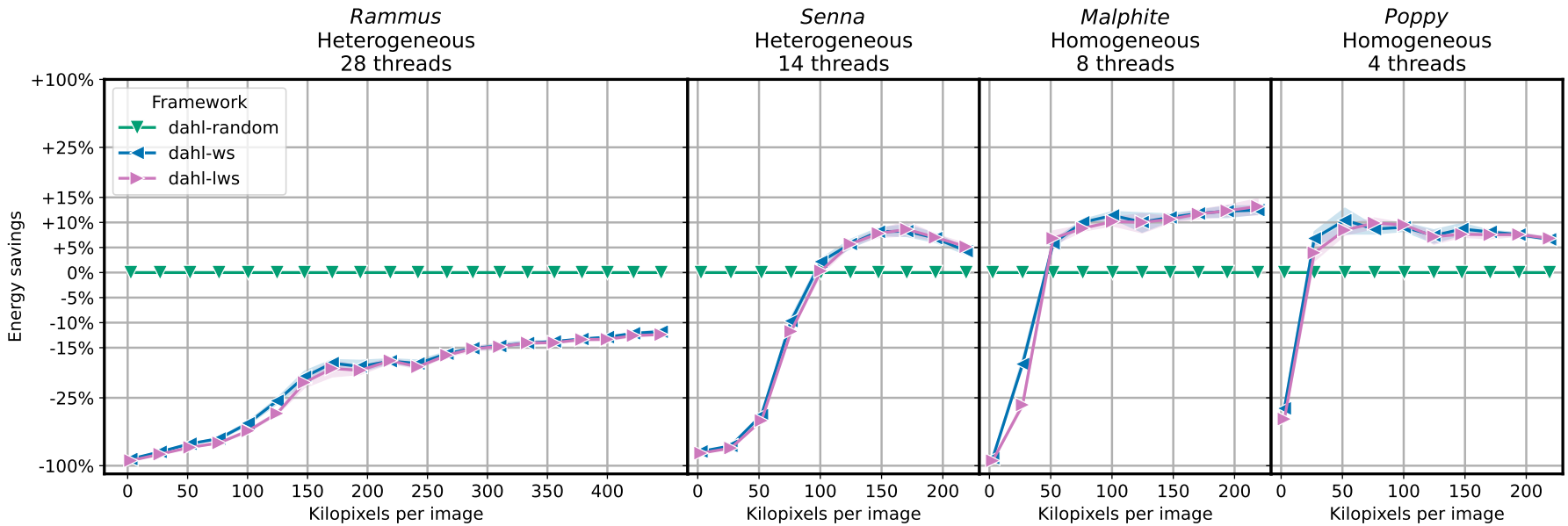
What are the gains?

- Need a baseline?
- Let's use the `Random` scheduler!



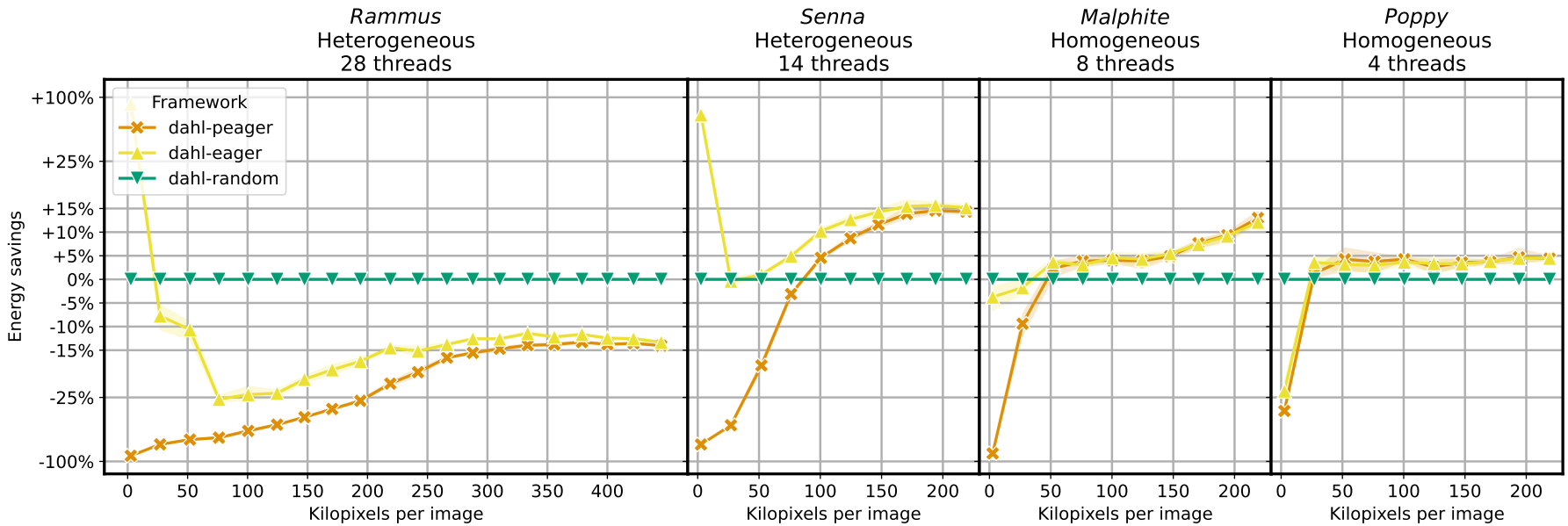
What are the gains?

- Work-stealing `WS` and `LWS` save up to 13% energy
- More consistent on **homogeneous CPUs**



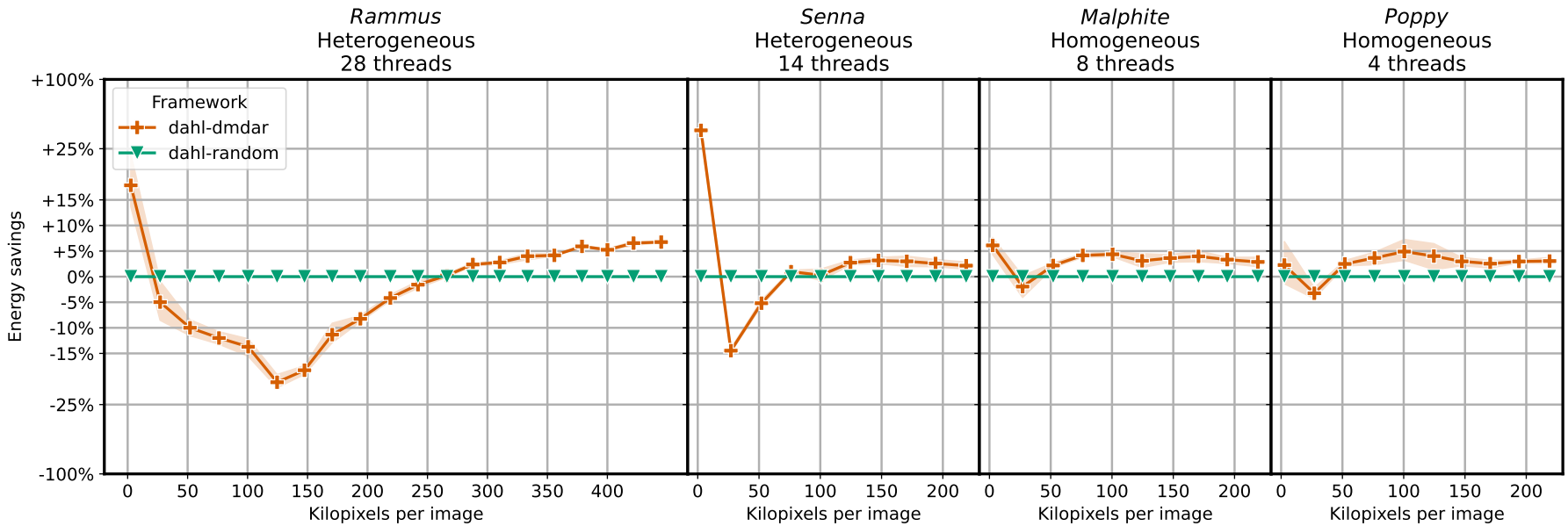
What are the gains?

- Eager and Peager save up to 15% energy
- **Best on the laptop**, *Senna*



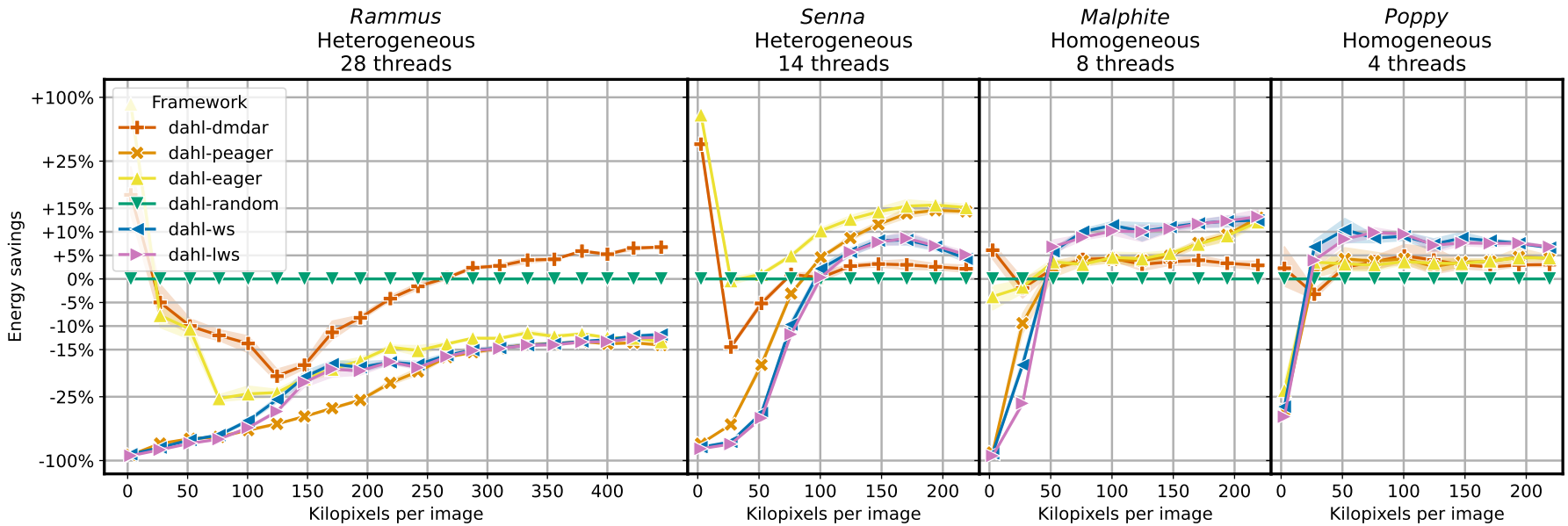
What are the gains?

- Scheduler DMDAR performs **great overall**, with smaller gains
- It is the only scheduler to save energy on *Rammus*!



What are the gains?

- **Difficulty** saving energy on **heterogeneous servers**
- Chosing the **correct scheduler** may save up to **15% energy**



Conclusion

- Default `Pytorch` configuration on heterogeneous CPUs may be **7.5× slower** and **consume 17× more energy**
- **Scheduler choice**: up to **22%** speedup, and **15%** greenup
- **Work-stealing** schedulers great on **homogeneous** CPUs
  Eager schedulers seems better on **heterogeneous** CPUs

Perspectives

- Distributed training, memory allocator, task granularity...
- Perform similar expirements on **GPU**
- Try to train on **GPU** + **CPU**

Perspectives

- Distributed training, memory allocator, task granularity...
- Perform similar expirements on **GPU**
- Try to train on **GPU** + **CPU**

Thanks for you attention!

References


Berthelot, Adrien et al. (2024). "Estimating the environmental impact of Generative-AI services using an LCA-based methodology". In: *Procedia CIRP* 122, pp. 707–712. DOI: 10.1016/j.procir.2024.01.098.


Brilland, T et al. (2025). "Évaluation de l'impact environnemental du numérique en France". In.


Falk, Sophia et al. (2025). "More than Carbon: Cradle-to-Grave environmental impacts of GenAI training on the Nvidia A100 GPU". In: *arXiv preprint arXiv:2509.00093*. DOI: 10.48550/arXiv.2509.00093.

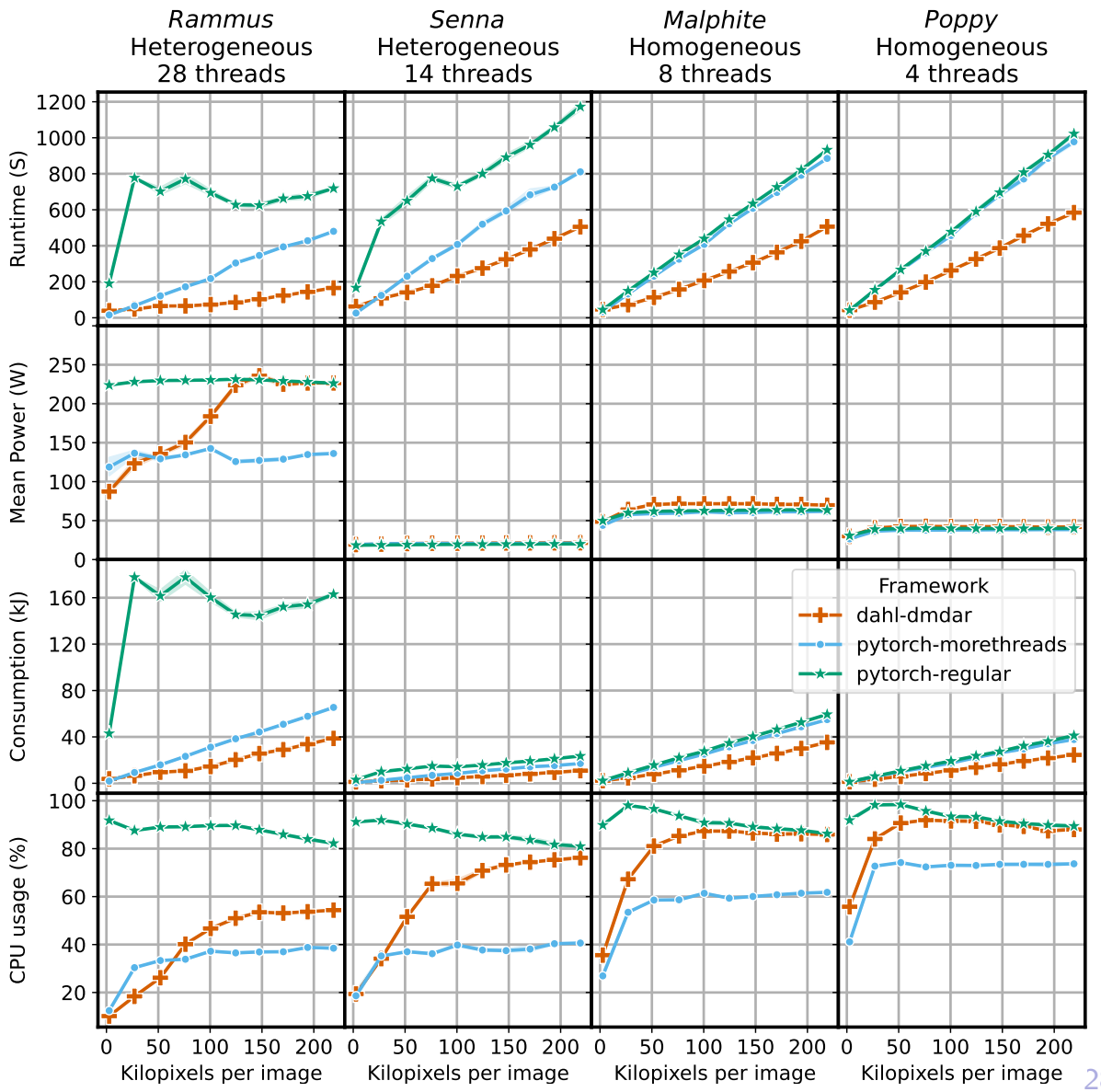

Li, Pengfei et al. (2025). "Making AI Less 'Thirsty'". In: *Commun. ACM* 68.7, pp. 54–61. DOI: 10.1145/3724499.


Patterson, David et al. (2021). "Carbon emissions and large neural network training". In: *arXiv preprint arXiv:2104.10350*. DOI: 10.48550/arXiv.2104.10350.


Tripp, Charles Edison et al. (2024). "Measuring the energy consumption and efficiency of deep neural networks: An empirical analysis and design recommendations". In: *arXiv preprint arXiv:2403.08151*. DOI: 10.48550/arXiv.2403.08151.

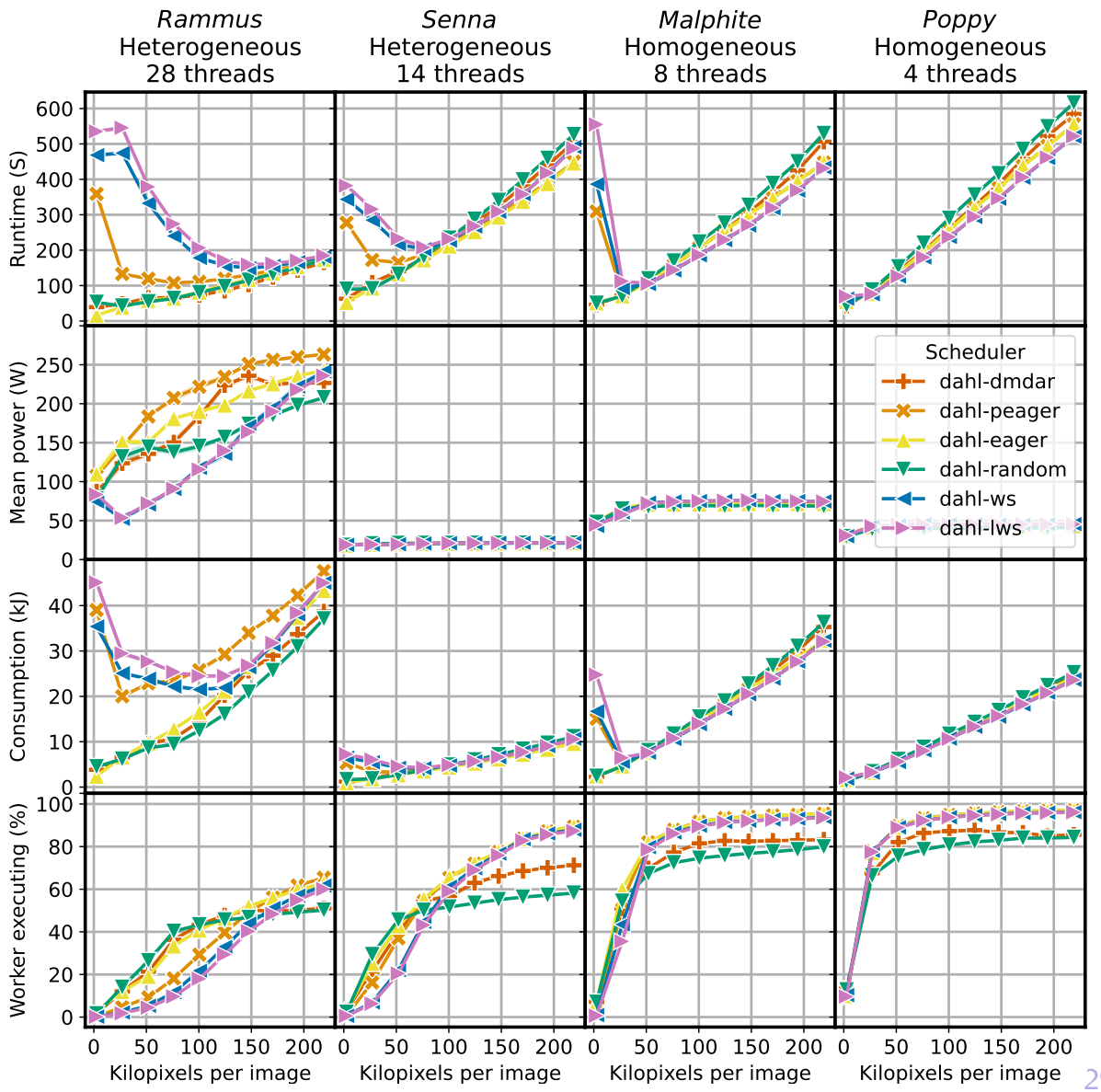
DAHL vs Pytorch

- Default Pytorch config bad when heterogeneous
- Increasing threads helps
- DAHL shows better performances overall



StarPU schedulers

- Random is decent overal on small tasks
- WS, LWS bad in heterogeneous setups
- Eager, DMDAR seems better



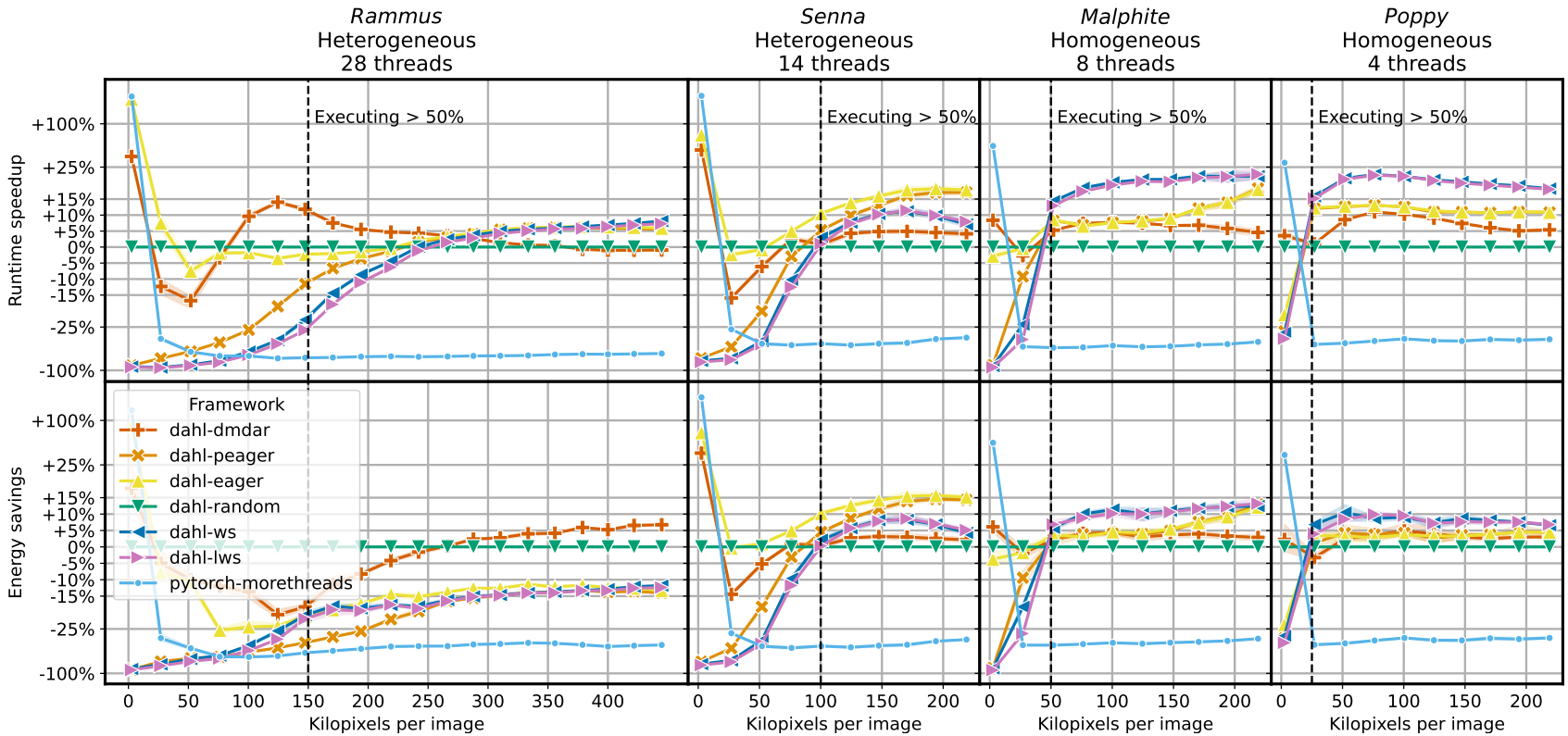


Figure: Relative runtime/energy gains between schedulers