

CAN CO-DESIGN OF SYSTEMS AND APPLICATIONS DELIVER SUSTAINABILITY?

Ana-Lucia Varbanescu

CAES @ EEMCS

a.l.varbanescu@utwente.nl



CAN CO-DESIGN OF SYSTEMS AND APPLICATIONS DELIVER SUSTAINABILITY?

Ana-Lucia Varbanescu

CAES @ EEMCS

[a.l.va](mailto:a.l.varbanescu@utwente.nl)

Most work & results by Merijn Verstraaten,
Duncan Bart, Jeffrey Spaan, Kevin Nobel, Skip Thijssen



ModSim relevant topics I will not cover

- We built a new GPGPU simulator for NVIDIA GPUs
 - Rust-based, parallel, (somewhat) more accurate than AccelSim
 - Extensible, yet limited to existing systems
- We studied end-to-end sustainability for data-centers
 - Model combining models from utilization to utilities
 - Focusing on the R's of sustainability – from Re(f)use to Recycle and Refurbish
- We work on Energy Labels for digital services in the computing continuum
 - Assess the energy consumption of different energy systems
 - Towards proposing energy labels
 - Extended iFogSim to support both multi-app and computing and networking energy modelling

Computing is everywhere ... and it's not free!

- Top 10 videos on YouTube* consumed as much as 600-700 EU persons per year (or about 400 North America persons)
- Training Alpha-Zero for a new game consumes as much as 100 EU persons per year
- A mid-size datacenter alone consumes as much energy as a small town
 - And that is not considering purchasing and secondary operational costs (e.g., cooling)
- In 2019 Dutch datacenters combined consumed 3-times more energy than the national railways
 - And consumption increased by 80% in 3 years
- The ICT sector is predicted to reach 21% of the global energy consumption by 2030

*https://en.wikipedia.org/wiki/List_of_most-viewed_YouTube_videos#Top_videos

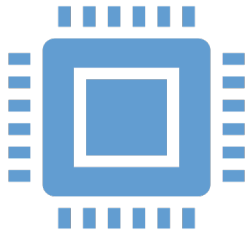
Computing is everywhere ... and it's not free!

- Top 10 videos on YouTube* consumed as much as 600-700 EU persons per year (or about 400 North America persons)
- Training Alpha-Zero for a new game consumes as much as 100 EU persons per year
- A mid-size datacenter alone consumes as much energy as a small town
 - And that is not considering purchasing and secondary operational costs (e.g., cooling)
- In 2011, the United States alone consumed more energy than the entire country of China
 - And that is not considering purchasing and secondary operational costs (e.g., cooling)
- The ICT sector is predicted to reach 21% of the global energy consumption by 2030

The energy consumption of computing is substantial and constantly increasing!

*https://en.wikipedia.org/wiki/List_of_most-viewed_YouTube_videos#Top_videos

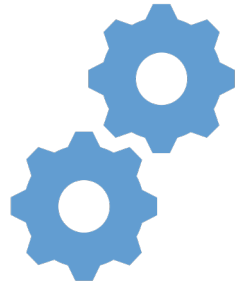
Three types of stakeholders



Developers and users

Improve the energy efficiency of their own codes, making use of algorithmic, programming, and hardware tools

Design and implement applications able to adapt to the available system resources



System integrators

Offer the right mix of resources for the application developers and system operators.

Include efficient hardware to enable different application mixes.



System operators

Ensure efficient scheduling of workloads on system resources.

Harvest energy where resources/systems are massively underutilized.

Agenda

- Different views on performance
 - Zero-waste computing
- Two case-studies
- (re)Defining systems codesign
- Graph processing and GraphMassivizer
 - Challenges and opportunities
- Take home message



Some relevant performance metrics*

- Speed-up: how much faster do we get with new machines, algorithms, ...

$$S(\text{workload}) = \text{Perf}(\text{Old}) / \text{Perf}(\text{New})$$

High-performance computing

- Efficiency: how efficient are we in getting performance

$$E(\text{workload}) = \text{Perf} / \text{Resources}$$

- Energy efficiency: how energy efficient are we in getting performance

$$EE(\text{workload}) = \text{Perf} / \text{Energy}$$

High-efficiency computing

- Utilization: how efficient are we utilizing our resources

$$U(\text{resource}) = \text{Achieved} / \text{Peak}$$

**please accept the naïve notation and pseudo-definitions*

Waste in computing

Unnecessary time (or energy) spent in (inefficient) computing is **compute waste**.

To reduce compute waste, we must focus on **efficiency-to-solution**

Detecting and reducing waste

- We assume computing waste is a consequence of underutilized resources.

- Informally, assume:

`system1` > `system2`

`P1` = performance(algorithm, workload, `system1`)

`P2` = performance(algorithm, workload, `system2`)



- “Strict” definition:

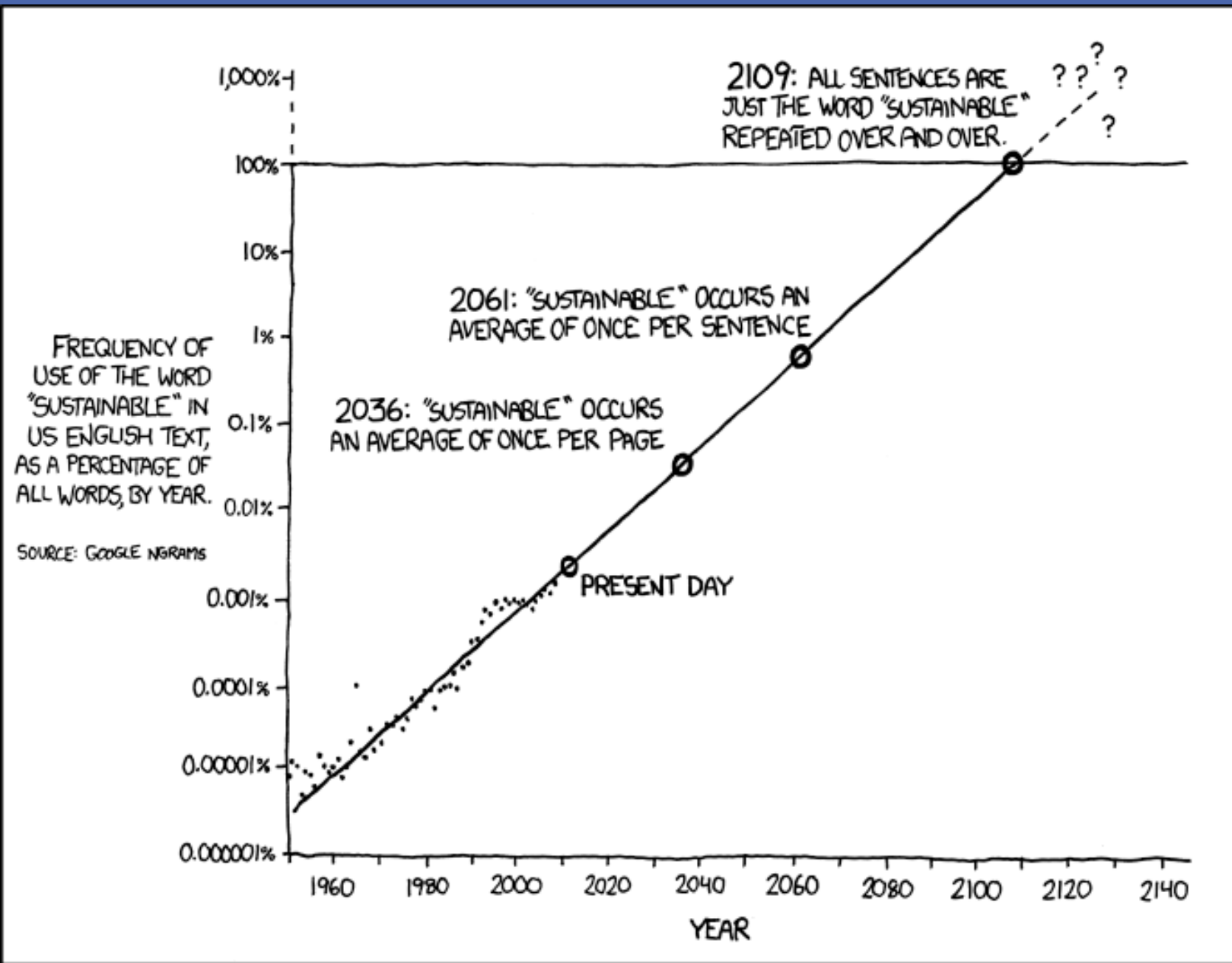
if (`P1` == `P2`) => waste in `P1`

- “Relaxed” definition:

if (abs (`P1` - `P2`) < T) => waste in `P1`

with T = threshold for performance loss

**performance is not necessarily runtime.*



THE WORD "SUSTAINABLE" IS UNSUSTAINABLE.

Waste == sustainability ??

Identifying waste leads to opportunities ...

- for better more sustainable system selection
- for efficient application collocation
- for DVFS and similar techniques to reduce energy consumption
- for better workload mapping and scheduling
- ...

The case of graph processing

Graph processing ...

... is / can be / will be everywhere!^{1,2}

- Social networks
- Bioinformatics
- Pandemic analysis³
- Fraud detection
- Neural networks
- ...

Bigger graphs and more complex computing
require more performance.

¹ Sherif Sakr et al. - “The Future Is Big Graphs: A Community View on Graph Processing Systems” – CACM Sept. 2021

² Tim Hegeman, Alexandru Iosup - “Survey of Graph Analysis Applications” - arXiv:1807.00382

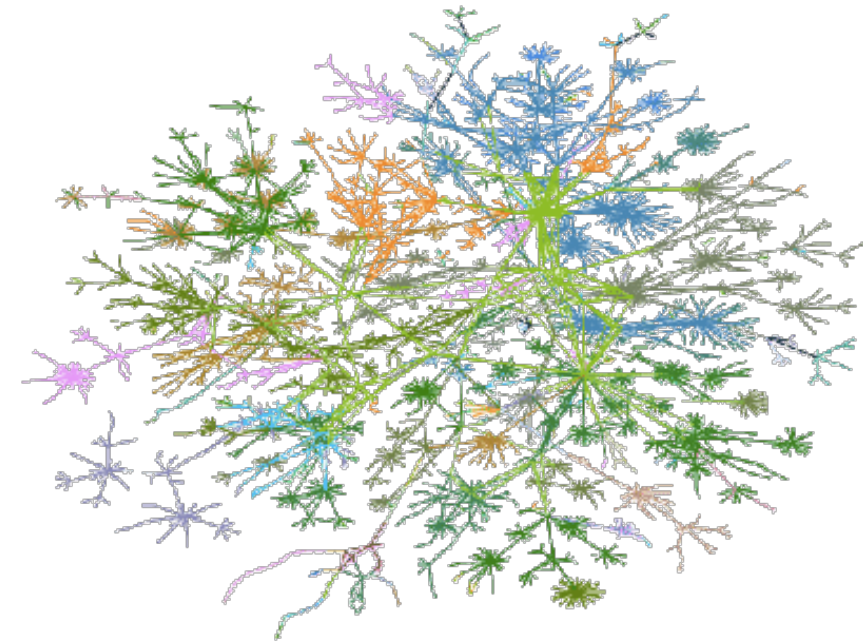
³ <https://neo4j.com/graphs4good/covid-19/>

Parallel graph processing

- Current *PUs
 - Massive (data) parallelism
 - Optimized for high throughput processing
 - Penalties for irregular execution
 - Penalties for load imbalance
- Graph processing ⁴
 - Data-driven computations
 - Irregular memory accesses
 - Poor data locality
 - Unstructured problems
 - Low computation-to-data access ratio



(mis)match?



⁴ Andrew Lumsdaine et al.

“Challenges in Parallel Graph Processing” – Parallel Processing Letters 2007

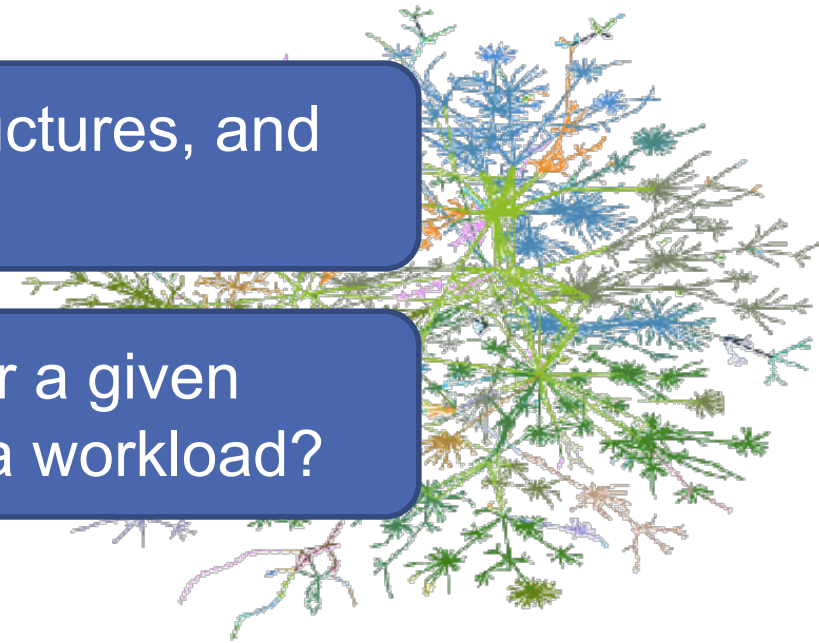
Parallel graph processing

- Current *PUs
 - Massive (data) parallelism
 - Optimized for high throughput processing
 - Penalties for irregular execution
 - Penalties for load imbalance



(mis)match?

- Graph processing
 - Data-driven
 - Irregular
 - Poor data locality
 - Unstructured
 - Low connectivity
- Parallelism \Leftrightarrow New algorithms, data-structures, and graph processing systems
- How do we select the best algorithm for a given workload? What is the right hardware for a workload?

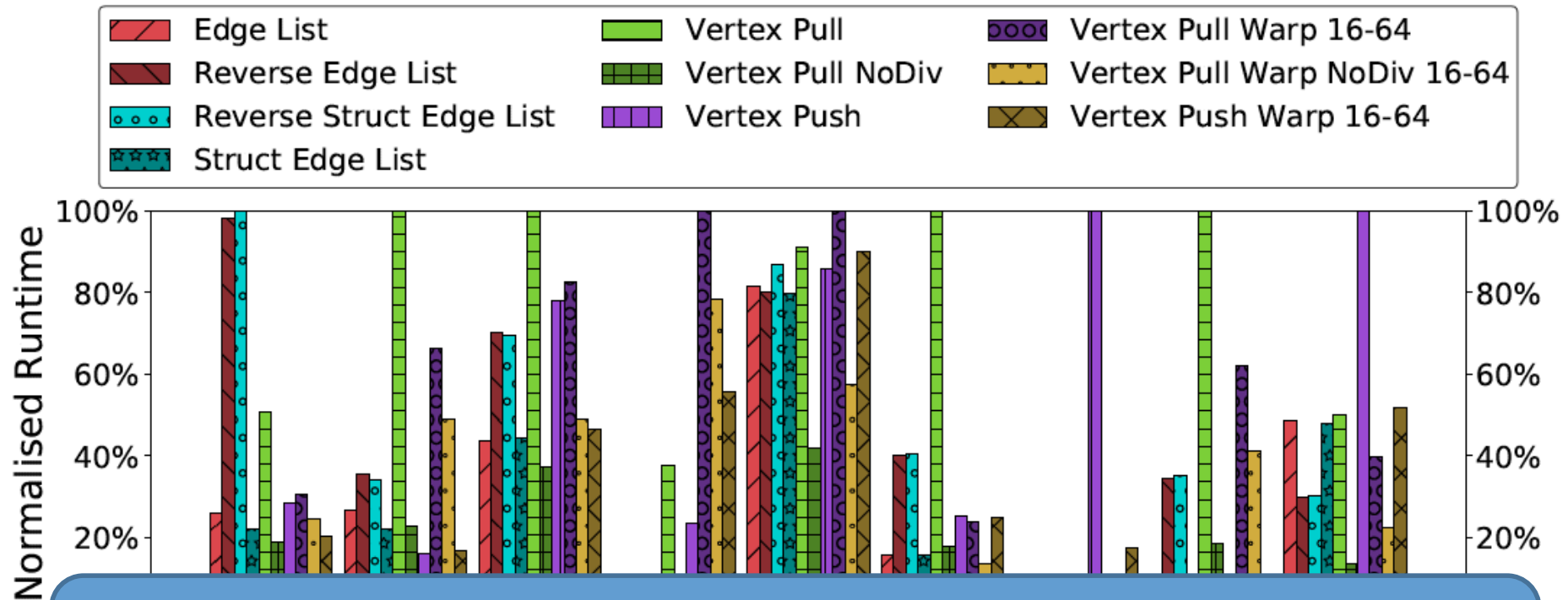


Some empirical evidence

- NVIDIA TitanX (revised on RTX2080Ti) + CUDA
- Two algorithms: PageRank and BFS
- Results presented on 9 graphs

Id	Graph	# Vertices	# Edges	Dataset
1	actor-collaboration	382,219	30,076,200	KONECT
2	amazon0601	403,394	3,387,390	KONECT
3	flixster	2,523,390	15,837,600	KONECT
4	jester1	73,512	8,272,720	KONECT
5	patentcite	3,774,770	16,518,900	KONECT
6	wikipedia_link_en	12,151,000	378,142,000	KONECT
7	wiki_talk_ru	457,017	919,790	KONECT
8	higgs-social_network	456,626	14,855,800	SNAP
9	sx-stackoverflow-c2q	1,655,350	11,226,800	SNAP

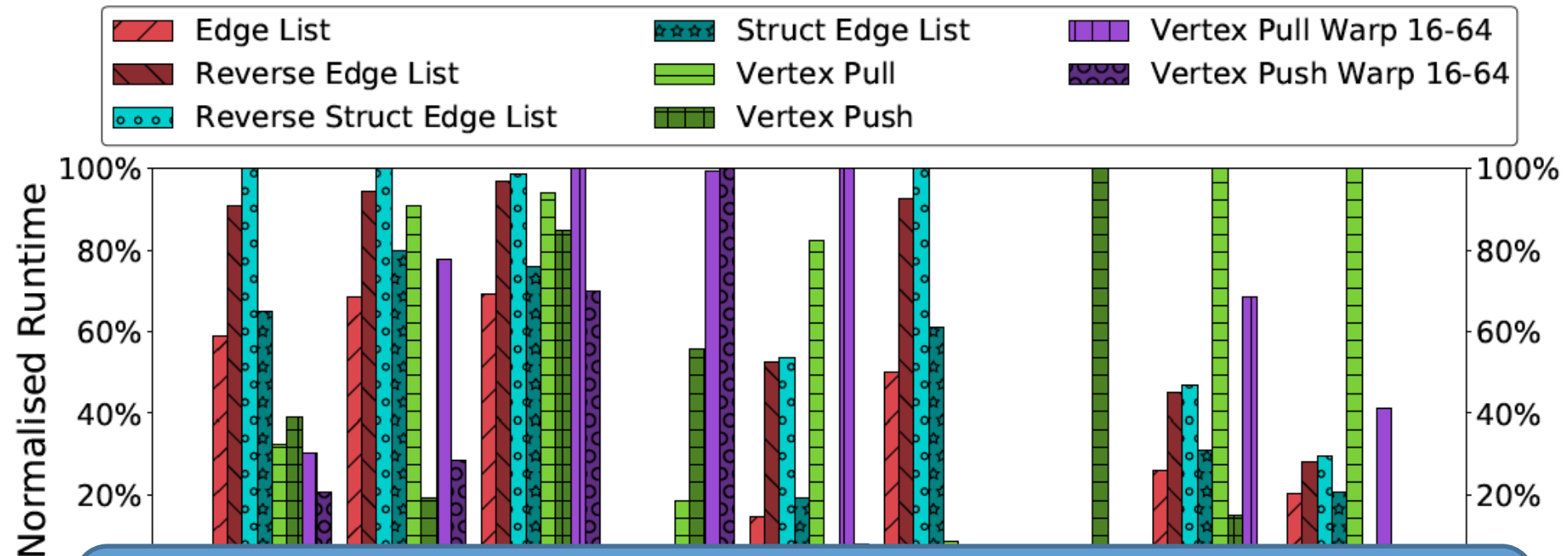
PageRank: results



- Different algorithms behave best.
- Different algorithms behave worst.
- The gap in execution time/energy cons. can be up to 2 orders of magnitude.

Choosing the wrong algorithm may lead to significant waste!

BFS: results



- Different algorithms behave best.
- Different algorithms behave worst.
- The gap in execution time/energy cons. can be up to 2 orders of magnitude.

Choosing the wrong algorithm may lead to significant waste!

Reducing waste in computing

Raise awareness

- Monitor (energy) efficiency
- Quantify waste

Performance analysis

Performance modeling

Performance optimization

Efficient scheduling and
resource sharing

Application-centric system
design

??

Improve efficiency

- Improve applications for the systems at hand
 - Make applications more efficient
 - Make applications share systems
- Improve systems for the applications at hand
- Co-design applications and systems

*Wishful thinking included.



Improve
applications for
the systems at
hand

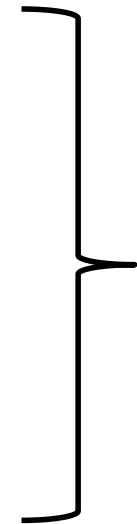


Dr. Merijn Verstraaten,
Skip Thijssen

Case study #1: Best graph-processing algorithm

Choose the best algorithm

- Model the **algorithm**
 - Basic analytical model (work & span)
- Calibrate to **platform**
 - GPU, CPU, ...
- Model the **dataset**
 - Size, dimension, topology ...
- Predict performance
 - Plug the platform and graph parameters into algorithm model
- Rank solutions and pick best.


$$T = f(P, A, D)$$

Choose the best algorithm

- Model the **algorithm**
 - Basic analytical model (work & span)
- Calibrate to **platform**
 - GPU, CPU, ...
- Model the **dataset**
 - Size, dimension, topology ...


$$T = f(\mathbf{P}, \mathbf{A}, \mathbf{D})$$

- Predict performance
 - Plug the platform and graph parameters into algorithm model

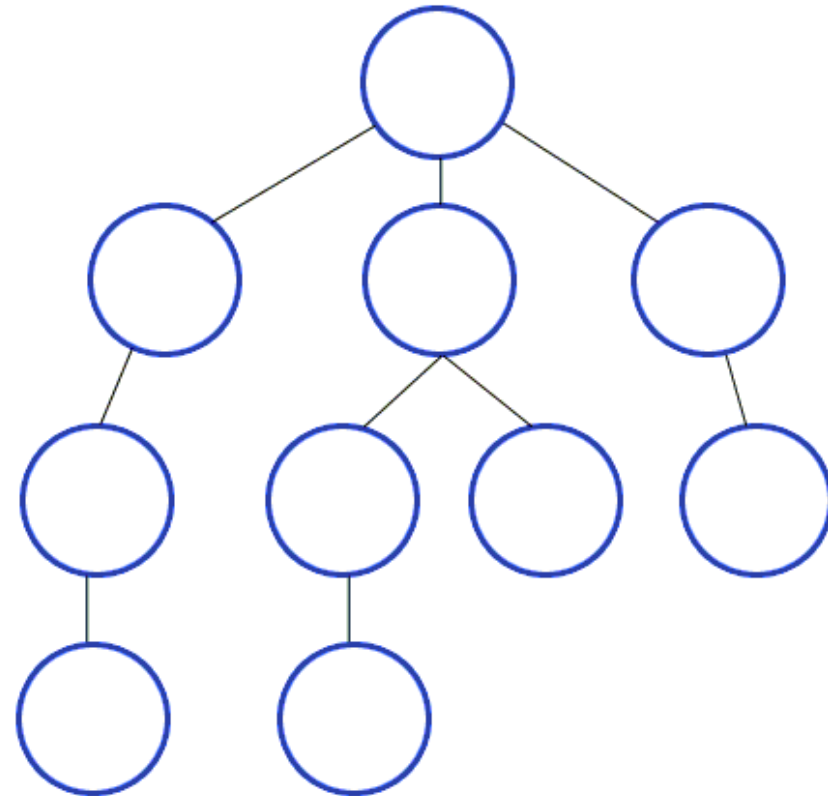
Many different attempts of analytical models failed!

- Rank solutions

Data-driven (ML) models work a bit better – still not sufficient !

BFS traversal

- Traverses the graph layer by layer
 - Starting from a given node
- Sensitive to ...
 - High diameter
 - Graph density
 - (dis)connected components
 - ...
- Challenges
 - No computation
 - Load-balancing
 - Irregular memory accesses



BFS traversal

- Traverses the graph layer by layer
 - Starting from a given node
- Sensitive to ...
 - High diameter
 - Graph density
 - (dis)connectivity
 - ...
- Challenges
 - No compact representation
 - Load-balancing
 - Irregular memory accesses



Best algorithm changes per level!

We cannot predict ... but we can construct!

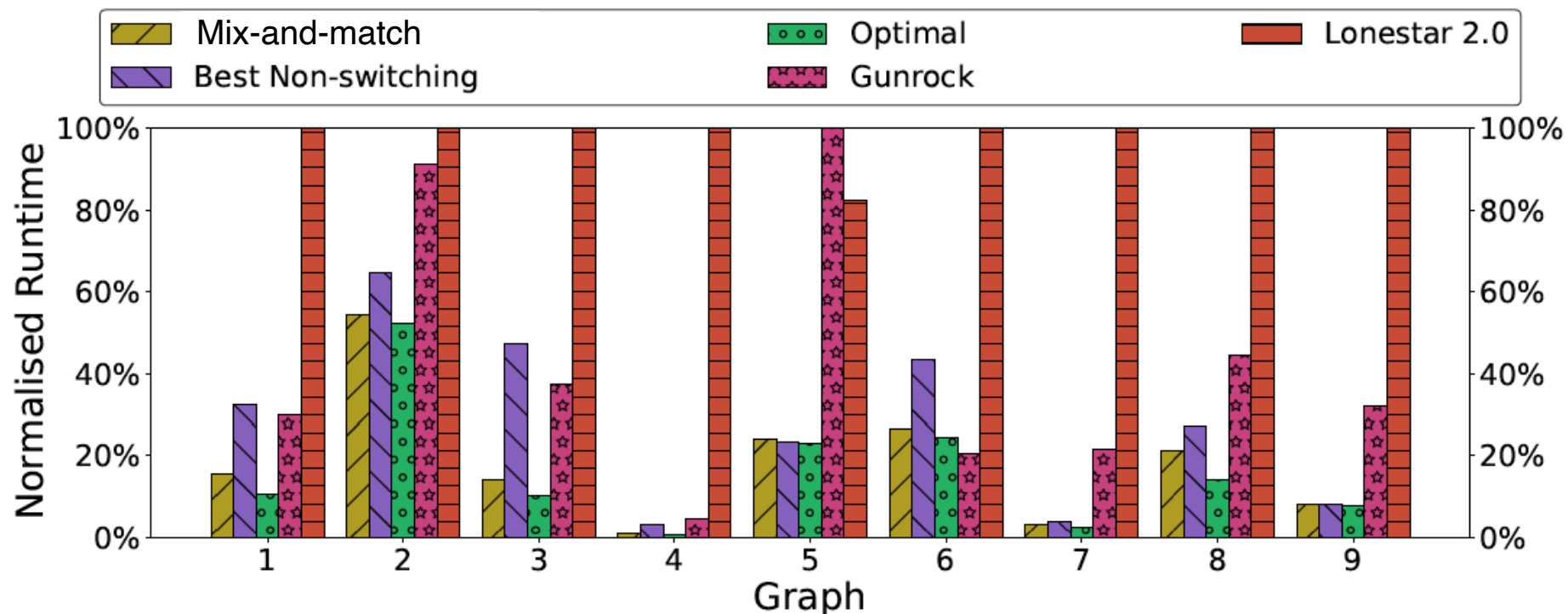
Constructing the best BFS

- Predict ranking
 - Determine the **best algorithm per level**
 - **Still** depends on platform and dataset ...
- **Construct** the best overall algorithm
 - Best algorithm per layer => best overall *by construction*
 - Switching between algorithms is a challenge
 - When?
 - How?

Mix-and-match: build the best algorithm at run-time by **switching to the best implementation** at every level*

*this is a generalization of the direction-switching BFS

Does it work?



- Runtime switching is possible, (currently) with some memory overhead
- We are faster than the state-of-the art, on average, by 3x

Mix-and-match uses performance variability to build the best BFS per graph!

Wait ... what about PageRank?

- Data-centric approach predicts the best performing algorithm with >95% accuracy
- It is simpler than BFS because it has no different steps and no incremental coverage of the graph
 - No need to construct a mixed algorithm
- So ... analytical models?
 - Still no luck ☹️



Lessons learned


- We can enable **some performance prediction using basic ML**
- We can provide the **fastest BFS algorithm by design**
- We can predict the **fastest PageRank using BDTs**
- **Prediction models are difficult to build for data-dependent kernels**
 - Data-centric approaches do offer a feasible alternative
- **Analytical modeling** failed because of the wrong granularity and the complexity of the hardware.
- Performance engineering can quickly become a big-data problem
 - We collected GBs of performance data

Lessons learned

- We can enable **some** performance prediction using basic ML
- We can provide the **fastest BFS algorithm by design**
- We can predict the **fastest PageRank using BDTs**
- **Prediction models are difficult to build for data-dependent kernels**

- High-efficiency algorithm
- High occupancy* for the compute cores
- Low utilization of the compute cores
- High utilization of the bandwidth

Wasted compute resources.



Improving systems
for the applications
at hand.

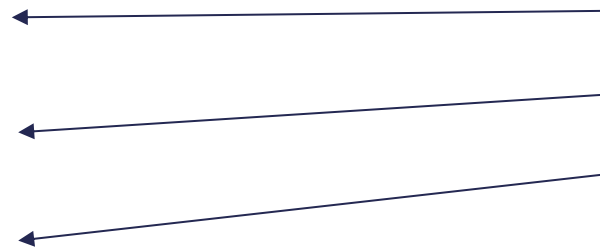


Jeffrey Spaan

Case study #2: Shrinking the platform

Possible workflow to identify waste

1. Pick a workload
2. Pick a baseline platform
3. **Reduce** resources
4. **Measure** performance
5. **Compare** performance








The devil is in the details.

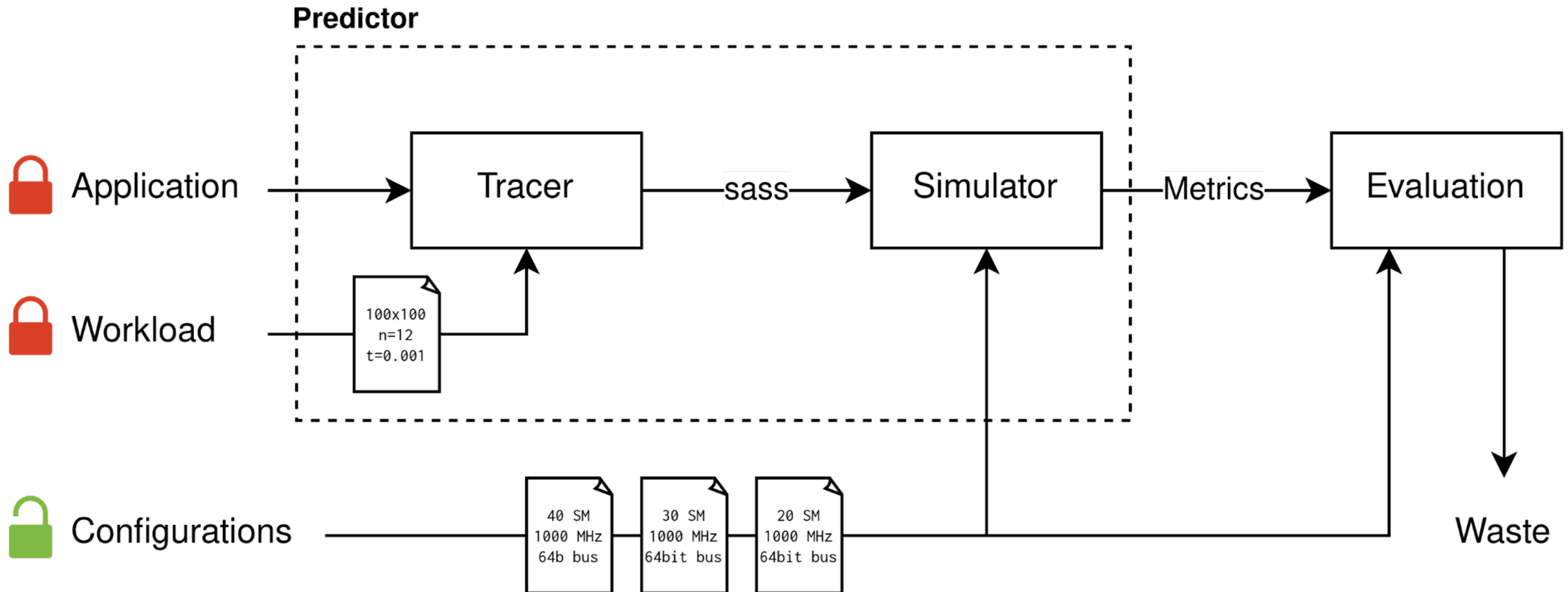
No difference or better performance? Found waste and/or a better system.

How to reduce resources? **How** to **measure** performance? **How** to compare?

~~Measuring~~ Predicting performance

- Benchmarking  Impossible
- Co-location  Difficult to setup
 - Simultaneous execution with a (specific) resource-consuming application
- Partitioning  Not available on many systems
 - Partitions with isolated GPU resources
- Analytical modelling
- Statistical modelling  Not sufficiently accurate
- Simulation  best option (currently)

Proposed workflow



Experimental setup

Applications:

- 5 Rodinia kernels:
 - **Compute-bound:** hotspot, k-means (2)
 - **Memory-bound:** k-means (1), backpropagation (1), backpropagation (2)

Systems:

- Baseline: RTX 2060 Super
- Variables:
 - **SMs:** 25, 30,, 40
 - **Core clock:** 1000, 1150,, 1900
 - **Memory clock:** 800, 1250, ..., 3500

Simulation run-time \approx 24-40 hours

Simulated with:

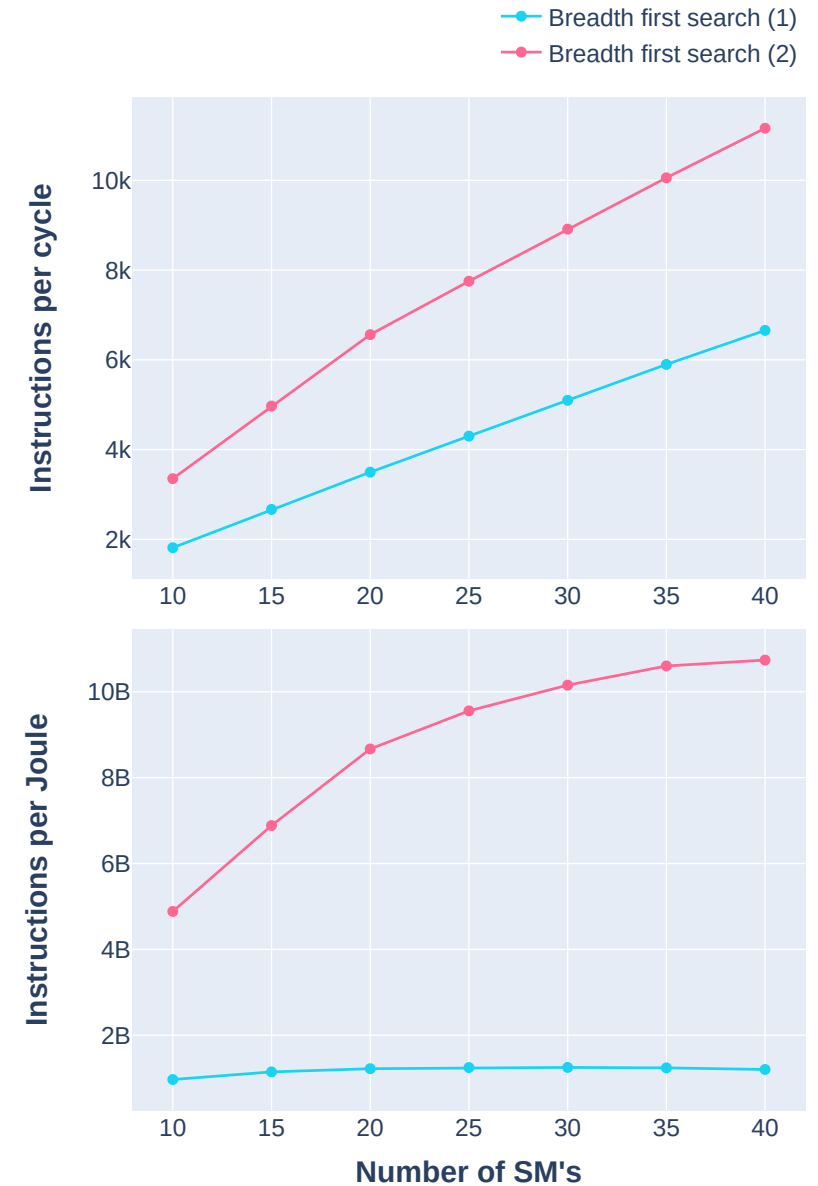
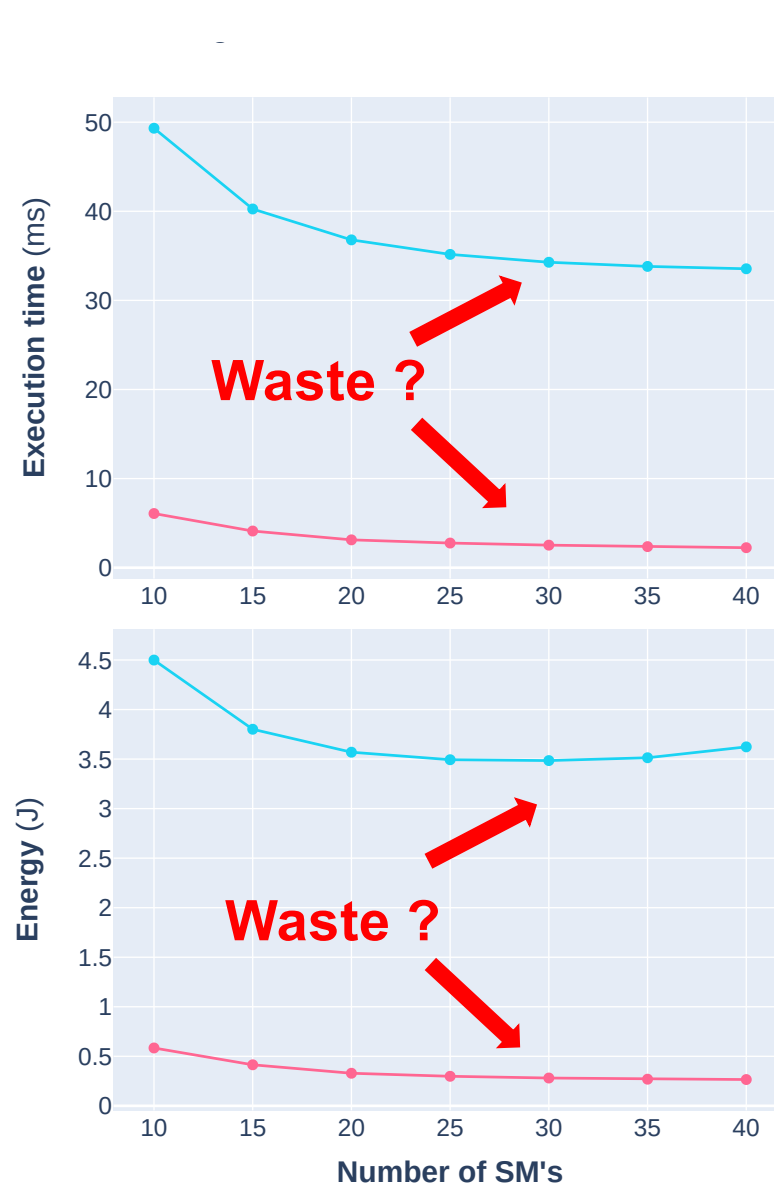


Ask me
more!

SMs: BFS

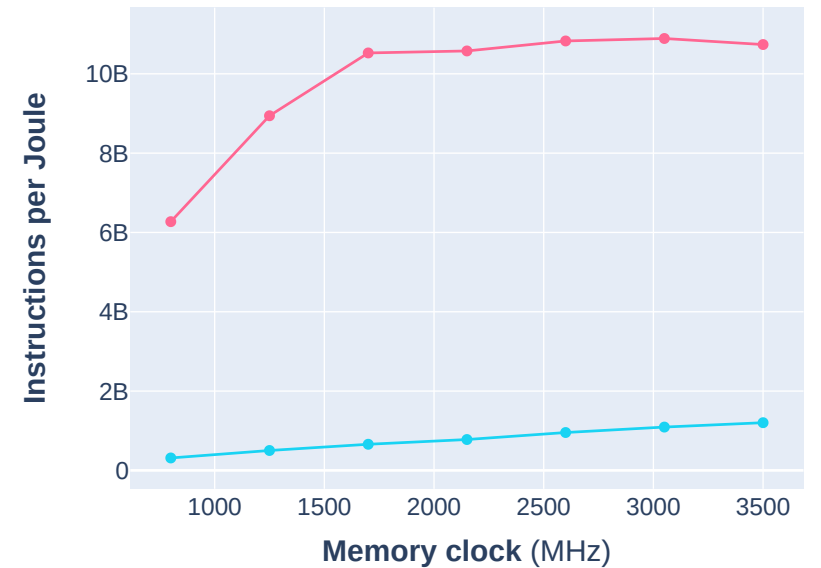
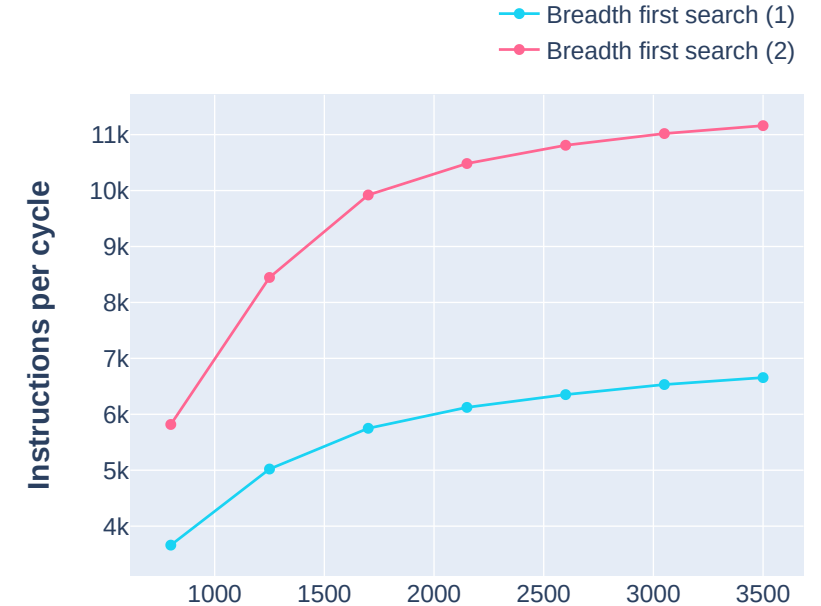
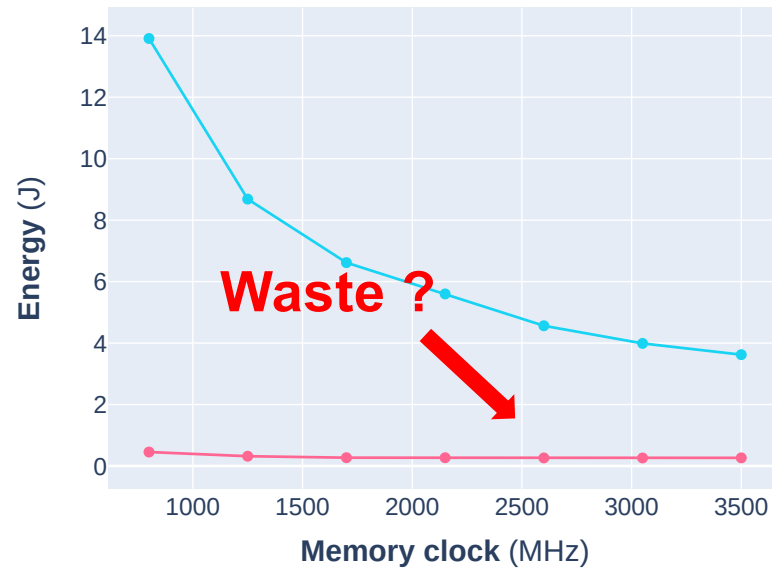
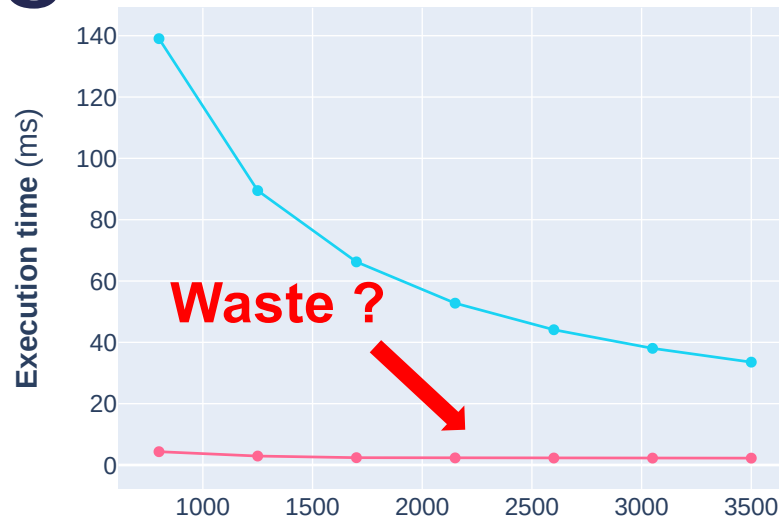
BFS is memory bound.

Is the strict definition reasonable? Should we use the relaxed definition?



Memory clock: BFS

As BFS is memory bound, we do expect to see performance gain when the memory clock speed increases.



Lessons learned

- We demonstrated **waste at resource-level can be significant**
- We demonstrated it is possible* (in simulation) to update platforms
- New opportunities for ...
 - Partitioning
 - Scheduling
 - Runtime systems

- Waste can (/should?) be investigated per resource.
- Difficult to model, trivial (but sloooow) to simulate.

Reconfiguring the system
can reduce compute waste.

Can we co-design?



Duncan Bart,
Kevin Nobel

Co-designing systems and applications

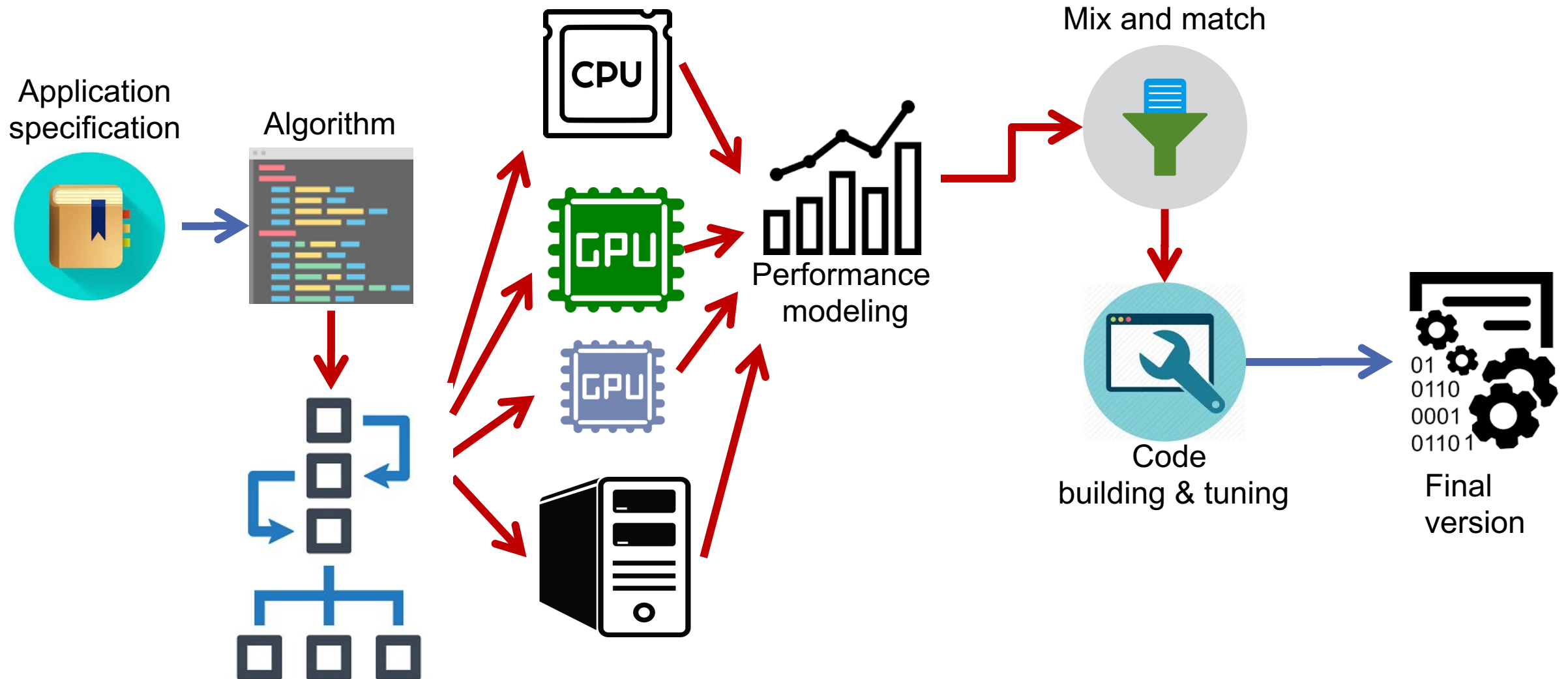
Main goal of **system co-design**

Determine the best-possible¹ system-configuration² for a given workload³.

1. Best-possible
 - Speed-up, utilization, energy efficiency, ...
2. System-configuration
 - Combination of resources: (fractions of) CPUs/ (fractions of) GPUs/heterogeneous ...
3. Workload
 - Processing to do _and_ representative data

Approach: model-based design-space exploration.

Co-design idea(s)

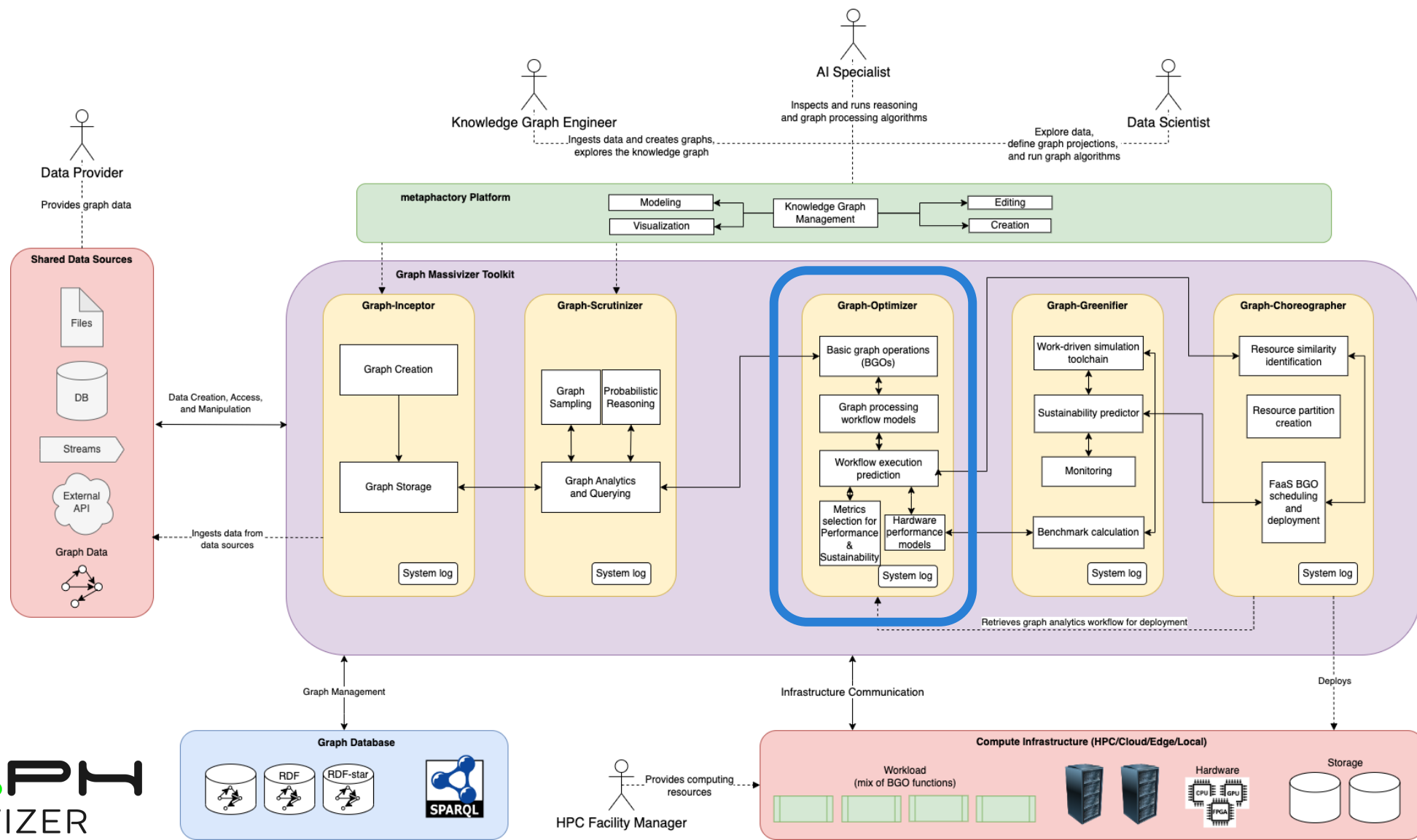


Open questions

- What is the right abstraction for the input?
- How do we split the workload in “basic units”?
- How do we build “basic units” performance models?
- How do we prune the search space?
- How do we do code building and tuning?
- What about the data?
- ...



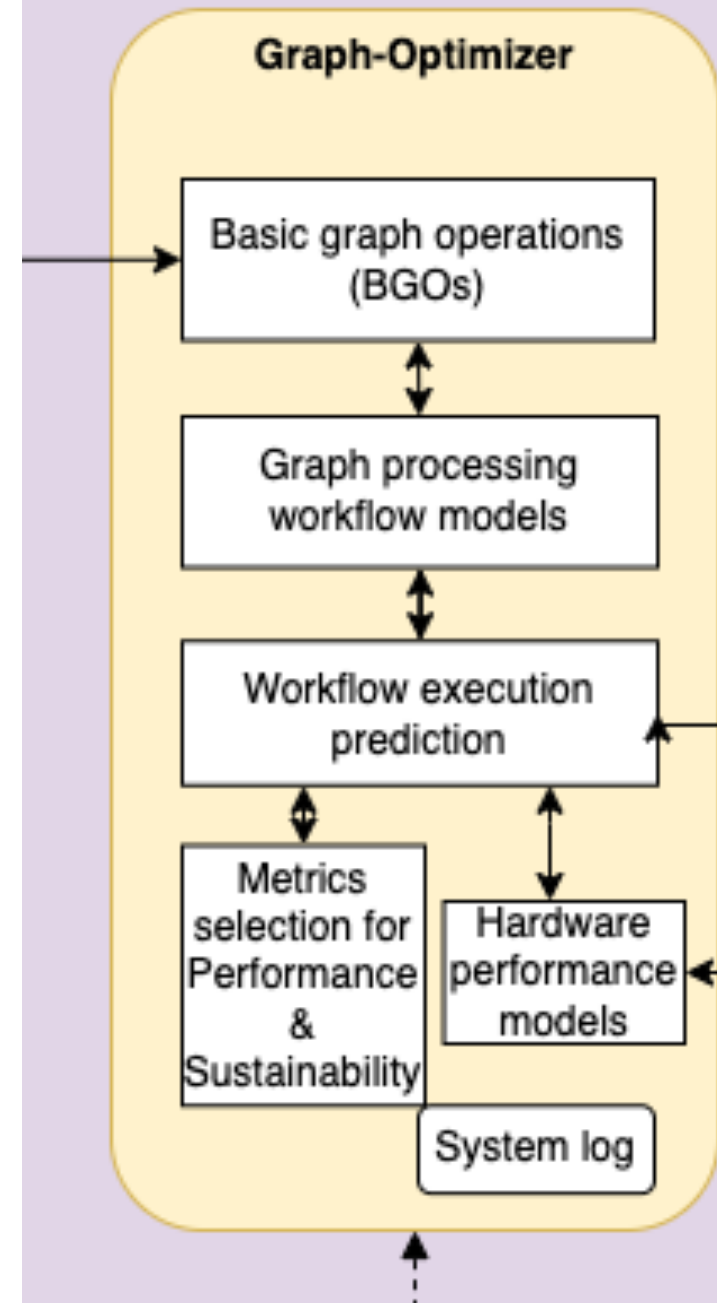
Graph-Massivizer: end-to-end graph processing



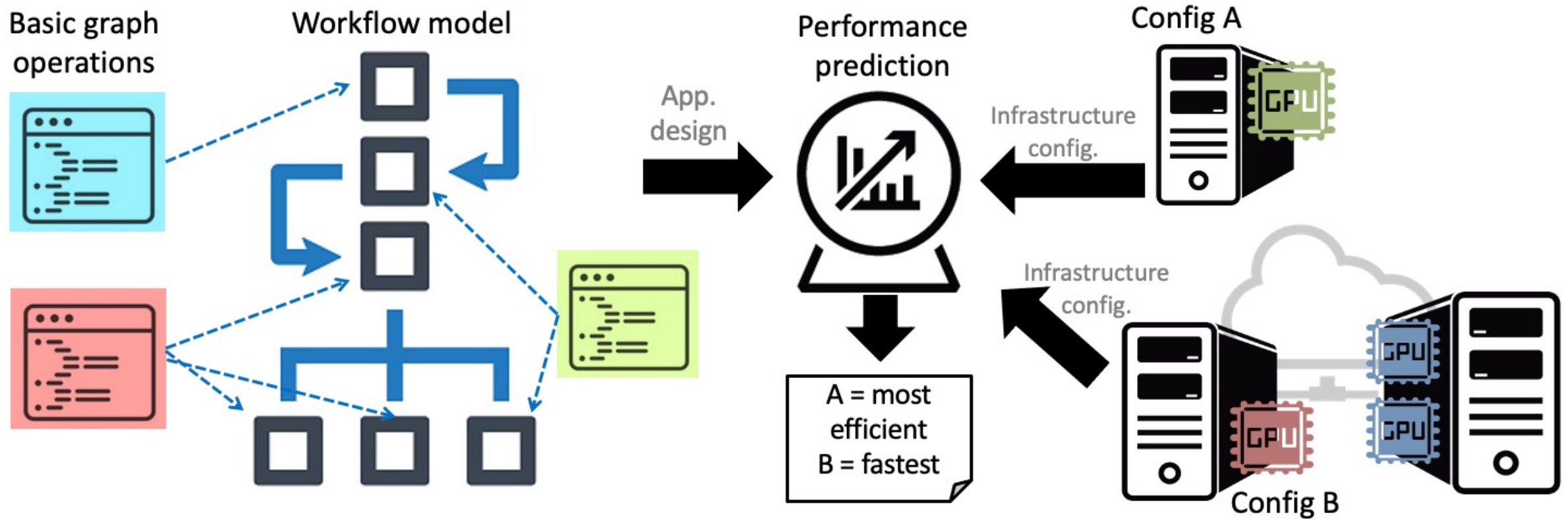
Graph-Optimizer & BGOs

BGOs = Basic Graph Operations that ...

- Can be implemented “independently” / “efficiently”
- Can be modelled for performance and energy
- Can be composed in “workflows”
- $\text{Workflow(BGOs)} + \text{Dataset} \Rightarrow \text{Workload}$
- Graph-Optimizer ...
 - uses a workflow model to compose BGOs
 - selects Best BGO for a given workload and platform
 - provides performance and energy consumption bounds per “execution plan” (aka, mapping)
 - selects “ideal system config”.



BGOs and Workflows



Co-design for sustainability [1]

- Sustainability = $f(\text{lower energy consumption, lower emissions energy})$
- Lower energy consumption depends on ...
 - Efficient hardware configuration
 - Efficient device selection
 - Efficient mapping (that is, "where" do we run)
- Lower-emissions energy depends on ...
 - Efficient location
 - Efficient scheduling (that is, "when" do we run)

Co-design for sustainability [2]

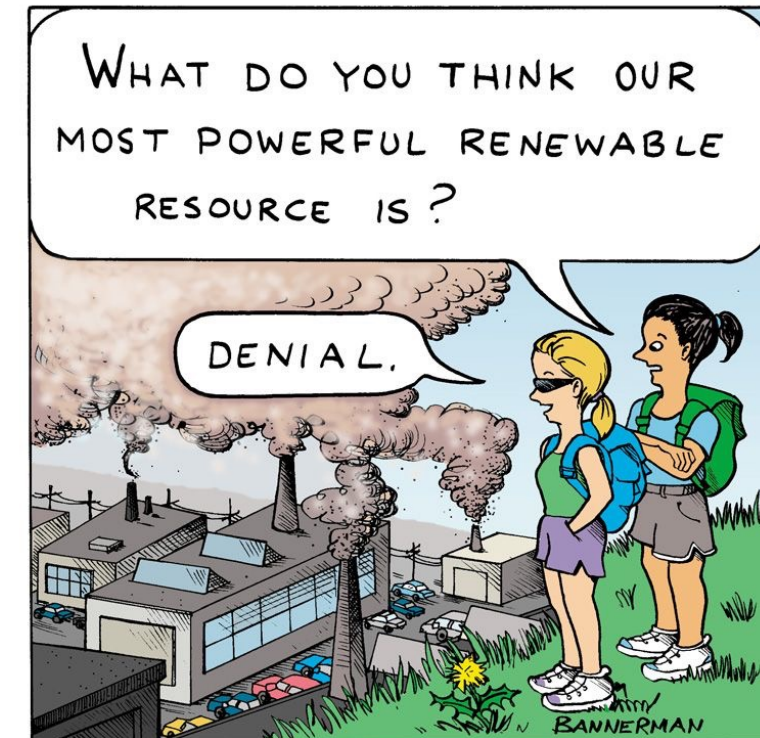
```
coDesign {  
  Input: H={HW components}, W=workload{algo, BGOs}, G=graph  
  G = read/load graph; // some in-memory representation  
  G = f(G{|V|,|E|, D, dD, ...}  
  G'= graphSample(G,size); // a small enough sample of G  
  forall (c in possibleSystemConfig{H}) { // naïve DSE  
    forall (mW in possibleMappings(W, c)) // mapping of tasks to HW  
      q = perfModel(mW, G, G') // performance modelling  
      if (q acceptable) { // best performance, or some margin  
        Systems += {(mW,q)};  
      } // add to feasible configs  
    }  
  }  
  Output = Systems // feasible configurations & predictions  
}
```

In summary ...

Co-design for sustainability



- From performance to zero-waste to sustainability
 - Zero-waste computing is a strong motivating example ...
 - ... but we need tools and methods for it.
- Tempting to co-design applications and systems
 - Need models for applications and performance/efficiency/waste
 - Need models/simulators for assessing system configurations
- Co-design with GraphMassivizer
 - Application models: from kernels to workflows
 - Providing performance and energy predictions
 - Focusing on *PU-based configurations



Relevant links

- Mix-and-Match BFS modeling
 - SC IA3 paper: <https://ieeexplore.ieee.org/document/8638408>
 - Full Phd thesis: <https://pure.uva.nl/ws/files/86139453/Thesis.pdf>
 - Github: <https://github.com/merijn/Belewitte>
- AI-based models for connected components:
 - Paper: <https://dl.acm.org/doi/10.1145/3528416.3530247>
- New work on modelling
 - MSc thesis available on demand
- GPGPU Simulator
 - MSc thesis available on demand
 - Github: <https://github.com/romnn/microgpusim>
- GraphMassivizer
 - Project: <https://graph-massivizer.eu/>
 - Paper on optimizer: <https://dl.acm.org/doi/10.1145/3578245.3585340>