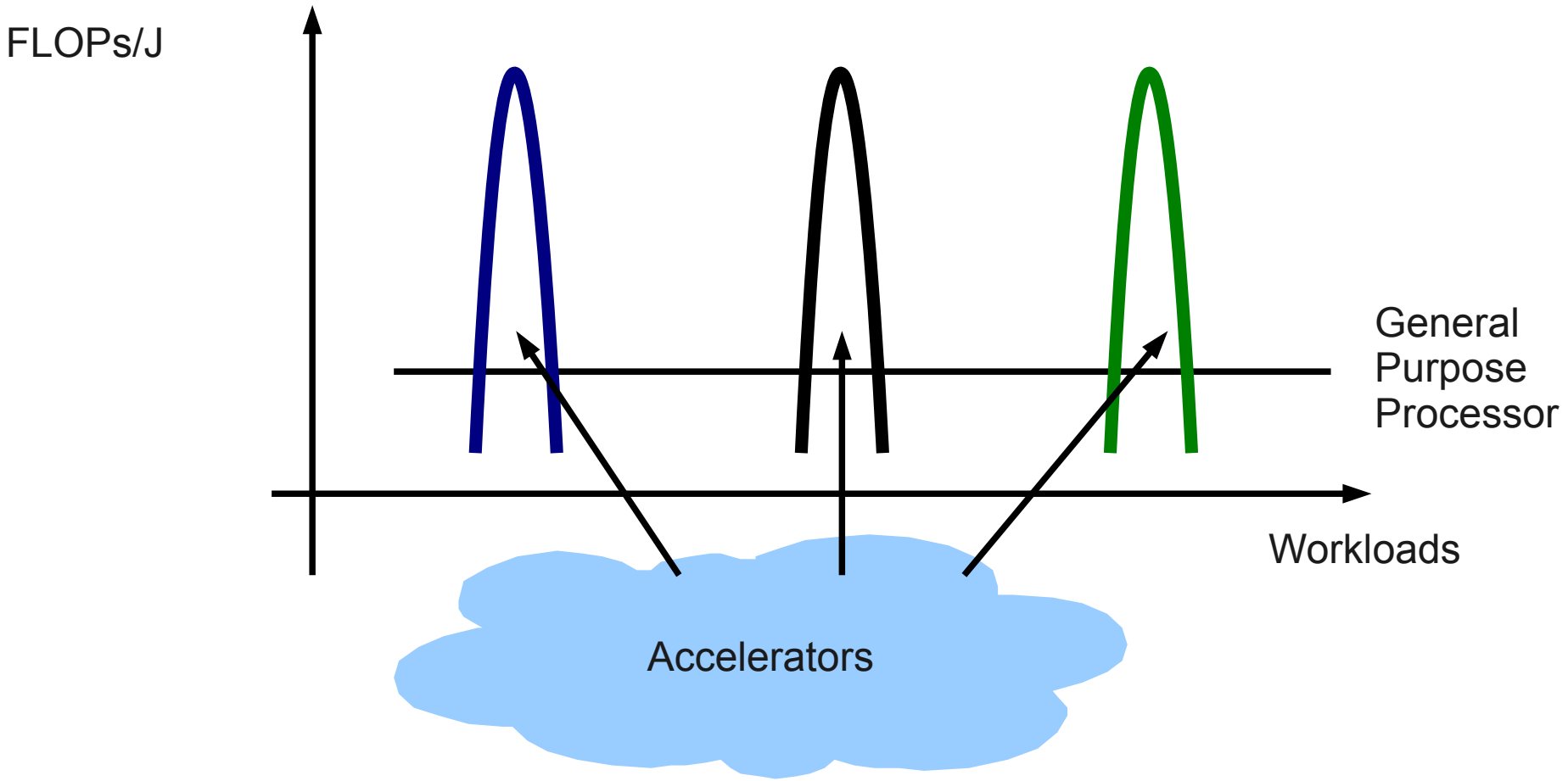
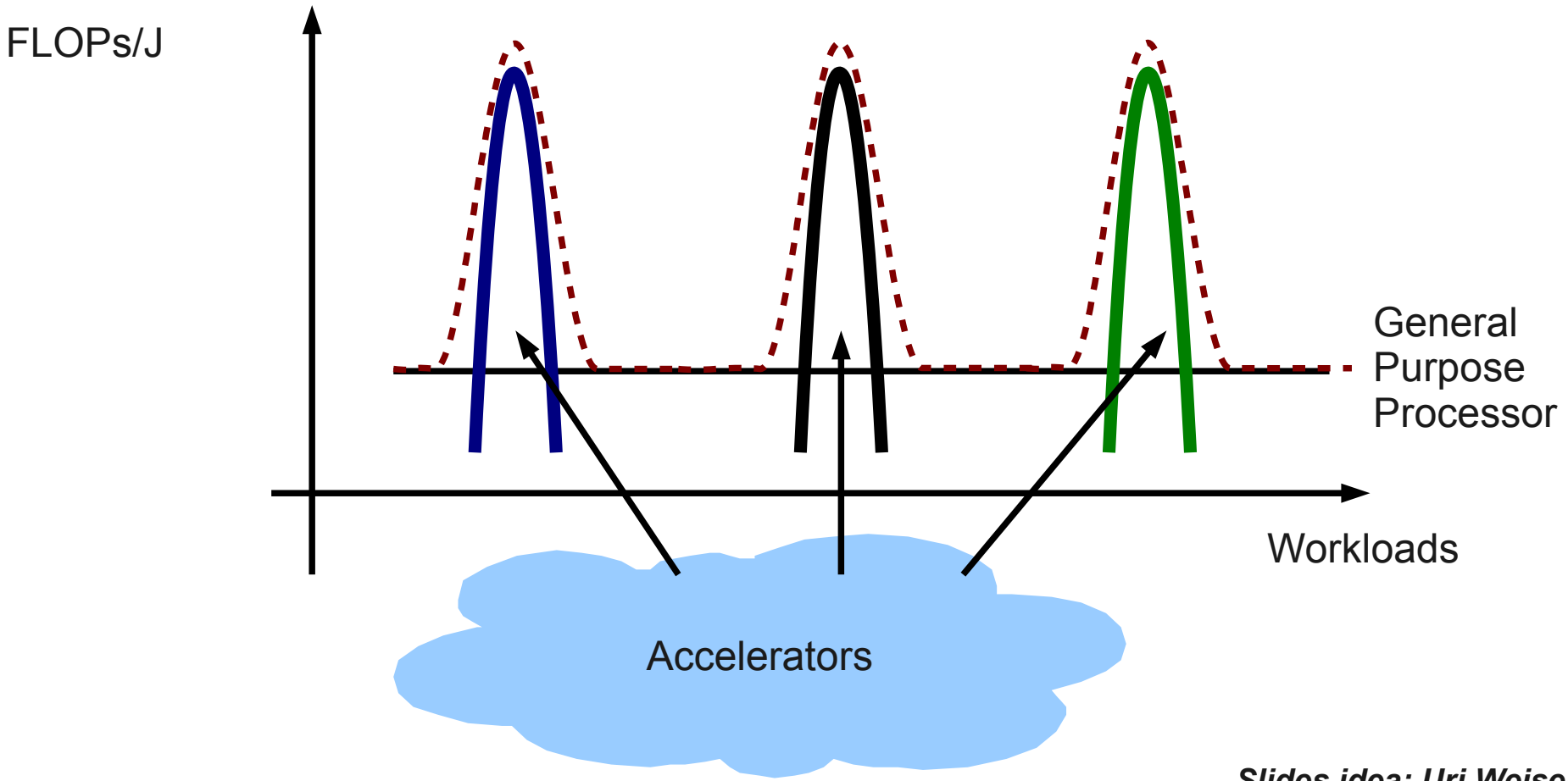
Energy-efficient acceleration of task dependency trees on CPU-GPU hybrids

Mark Silberstein - Technion
Naoya Maruyama – Tokyo Institute of Technology

The case for heterogeneous hardware

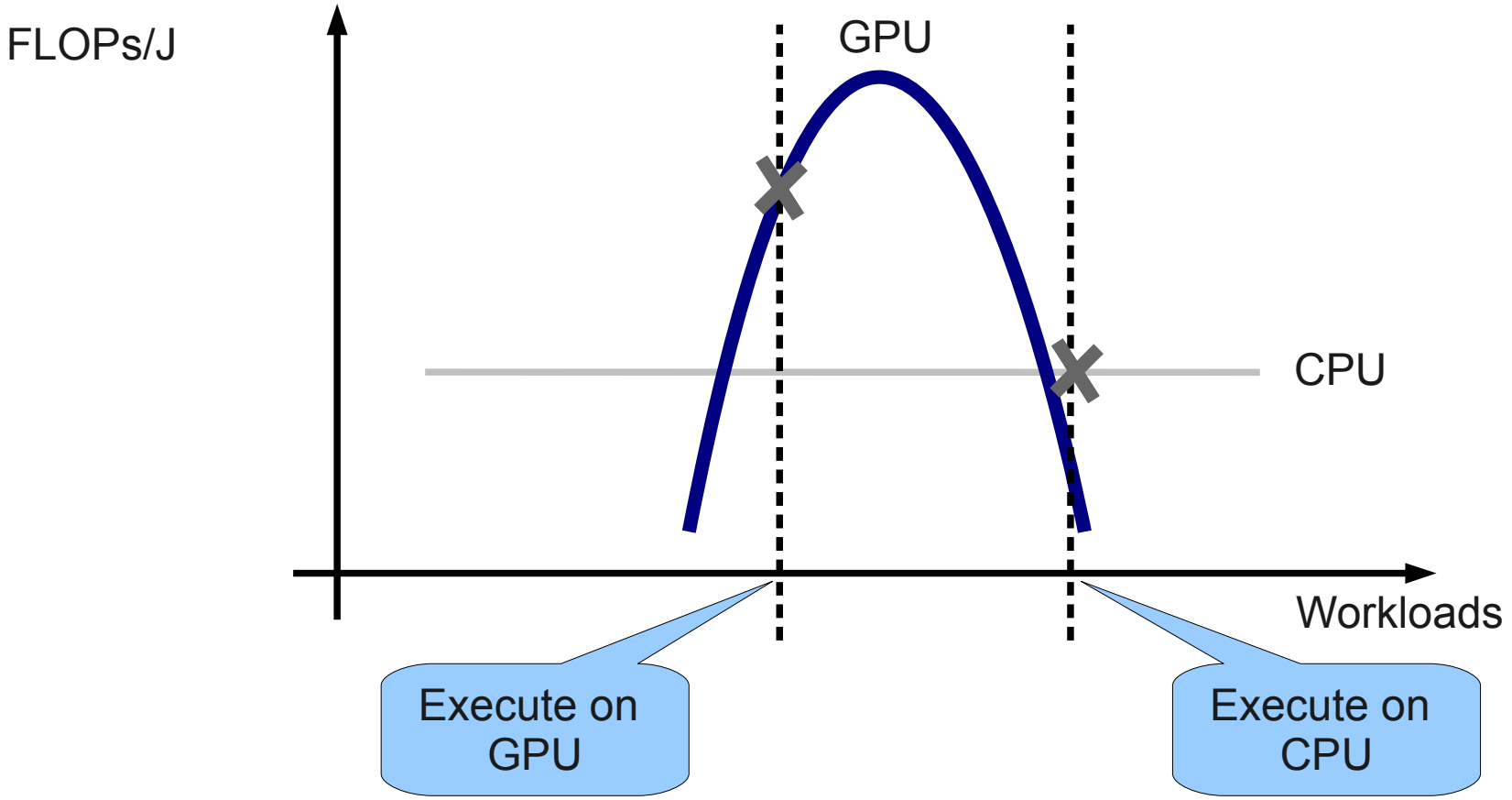


The case for heterogeneity-aware software

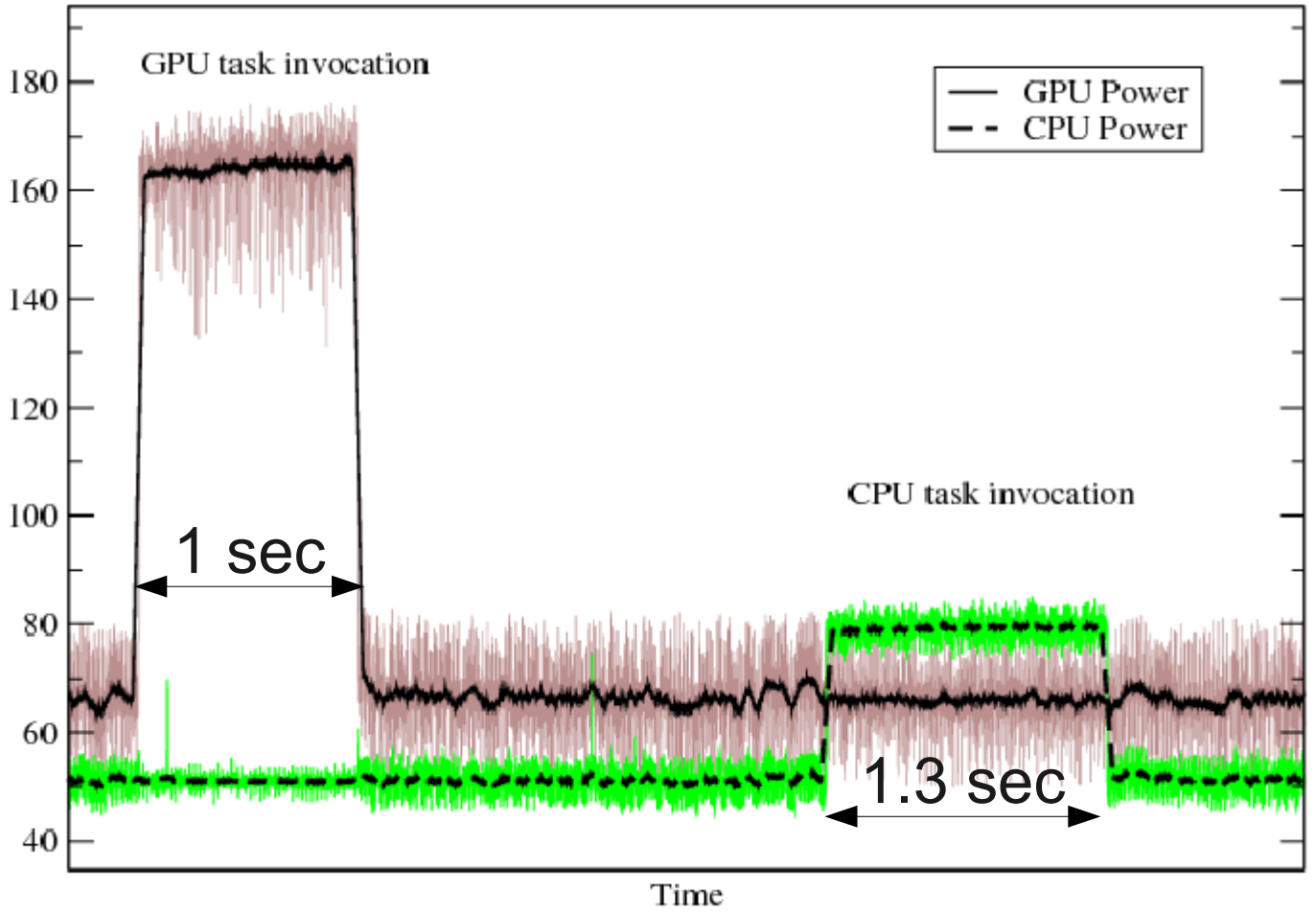


Slides idea: Uri Weiser

The case for energy-aware task assignment



Experiment: GPU vs. CPU for the same task



CPU: AMD Phenom Quadcore
GPU: NVIDIA GTX285

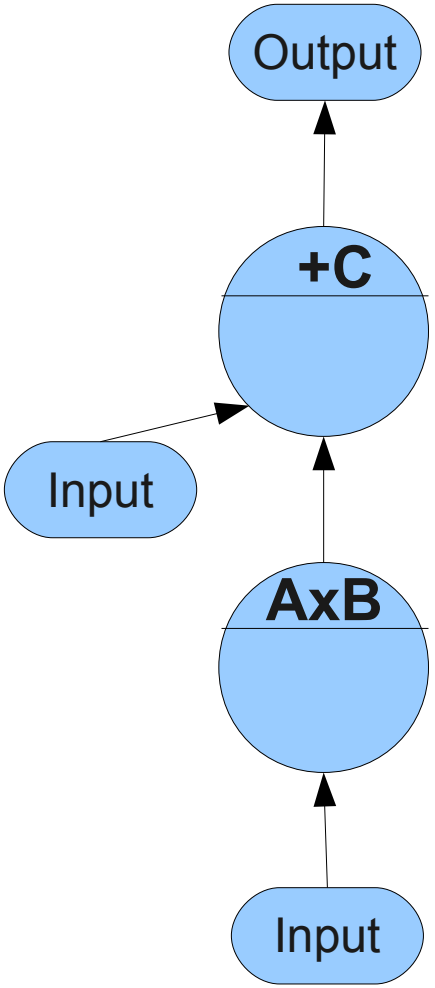
Outline

- Task dependency trees
- Energy-efficient acceleration assignment problem
- Optimal algorithm
- Evaluation on probabilistic networks workload

Input: task dependency trees

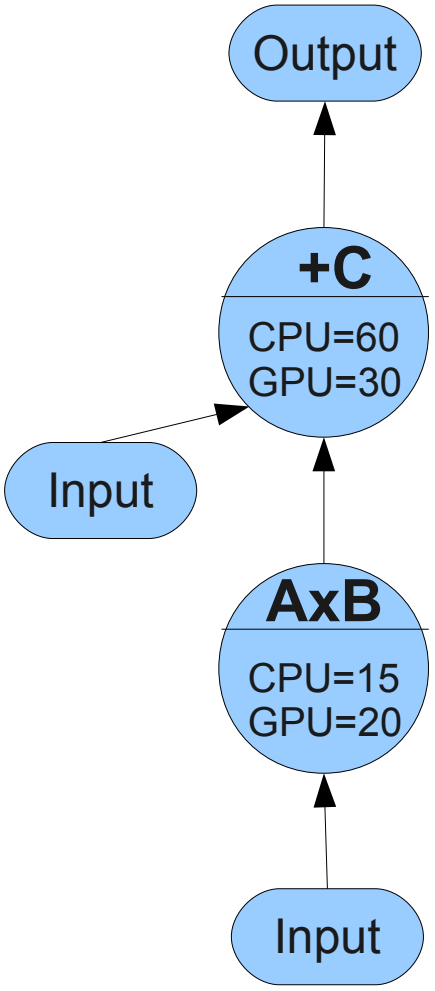
- Algebraic expression evaluation
- Divide and conquer strategies
- Inference in probabilistic networks

Dependency tree for **AxB+C**



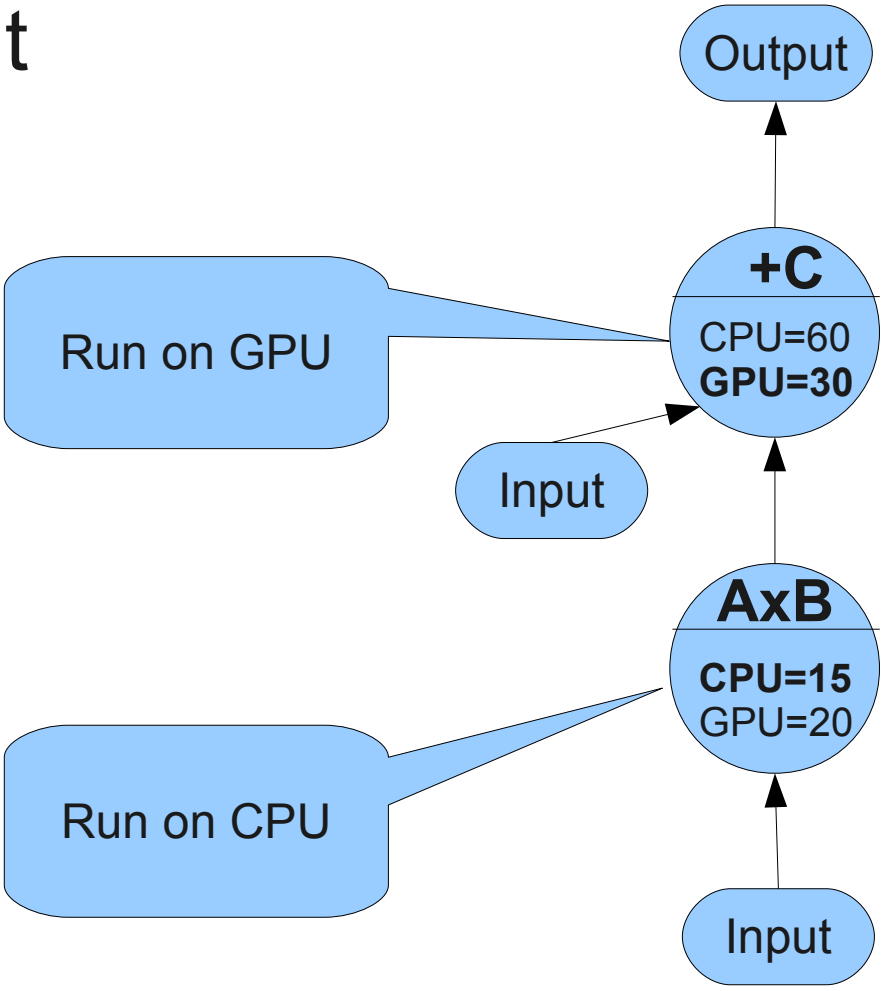
Goal: minimize power consumption

- Greedy assignment

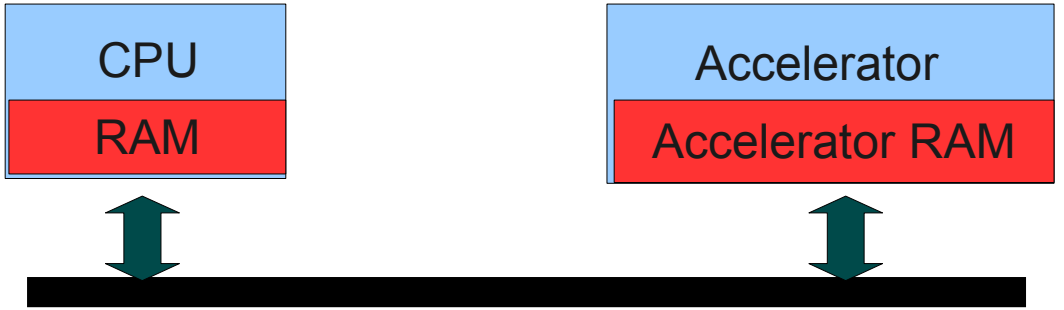


Goal: minimize power consumption

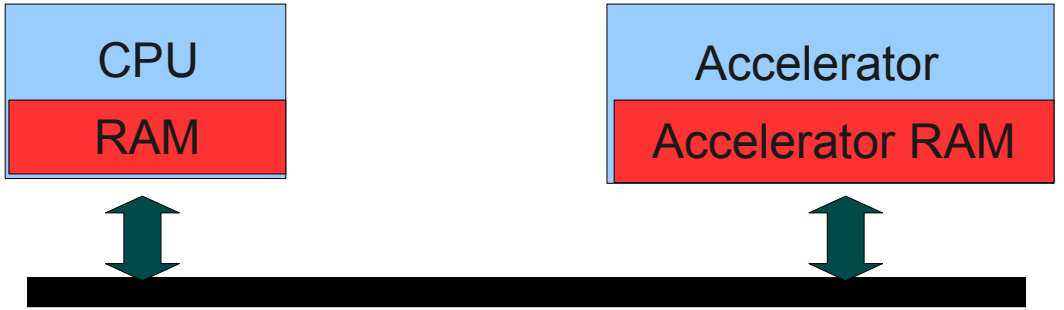
- Greedy assignment

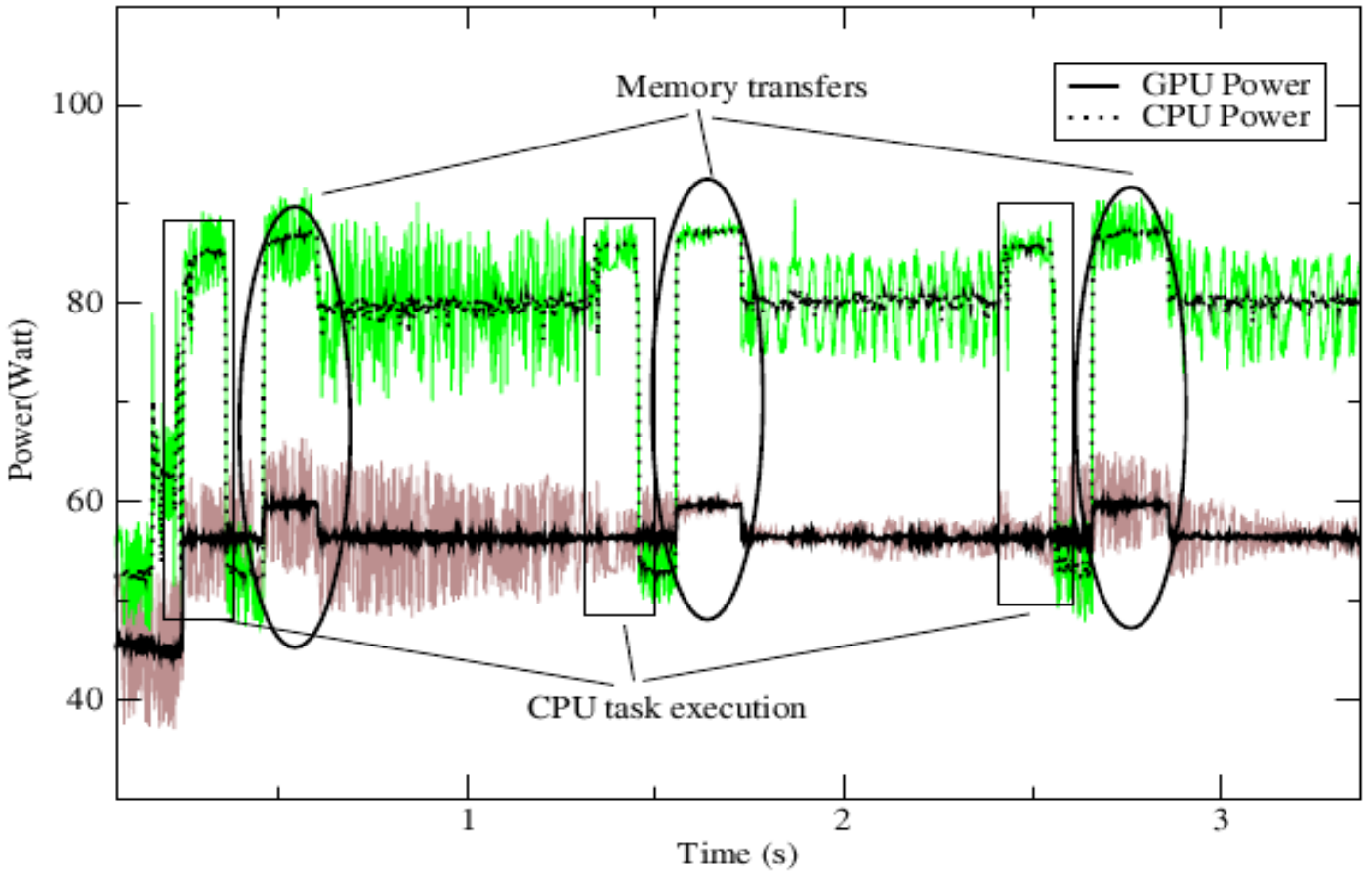


Communication is not free!



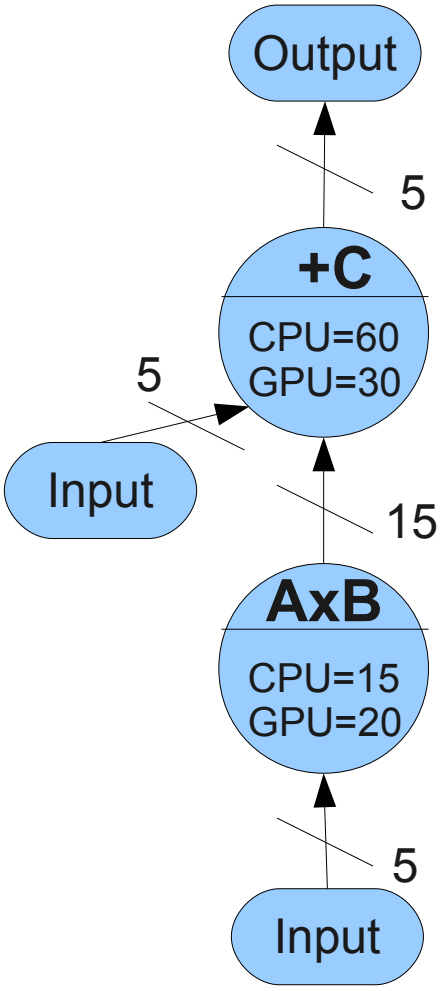
Communication is not free!





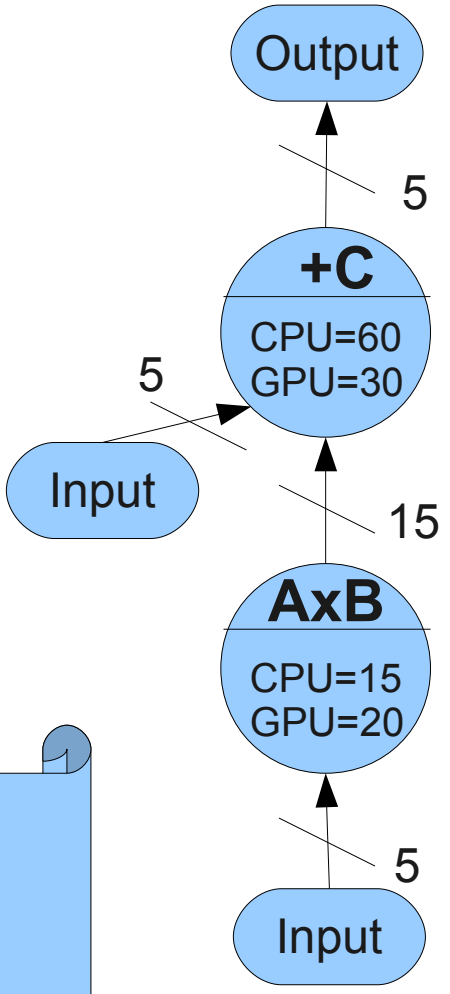
Greedy algorithm ignores communications

- Input and output should be moved to/from GPU
- Greedy (CPU-GPU) = 70J
- Optimal (GPU-GPU)= 65J



Greedy algorithm ignores communications

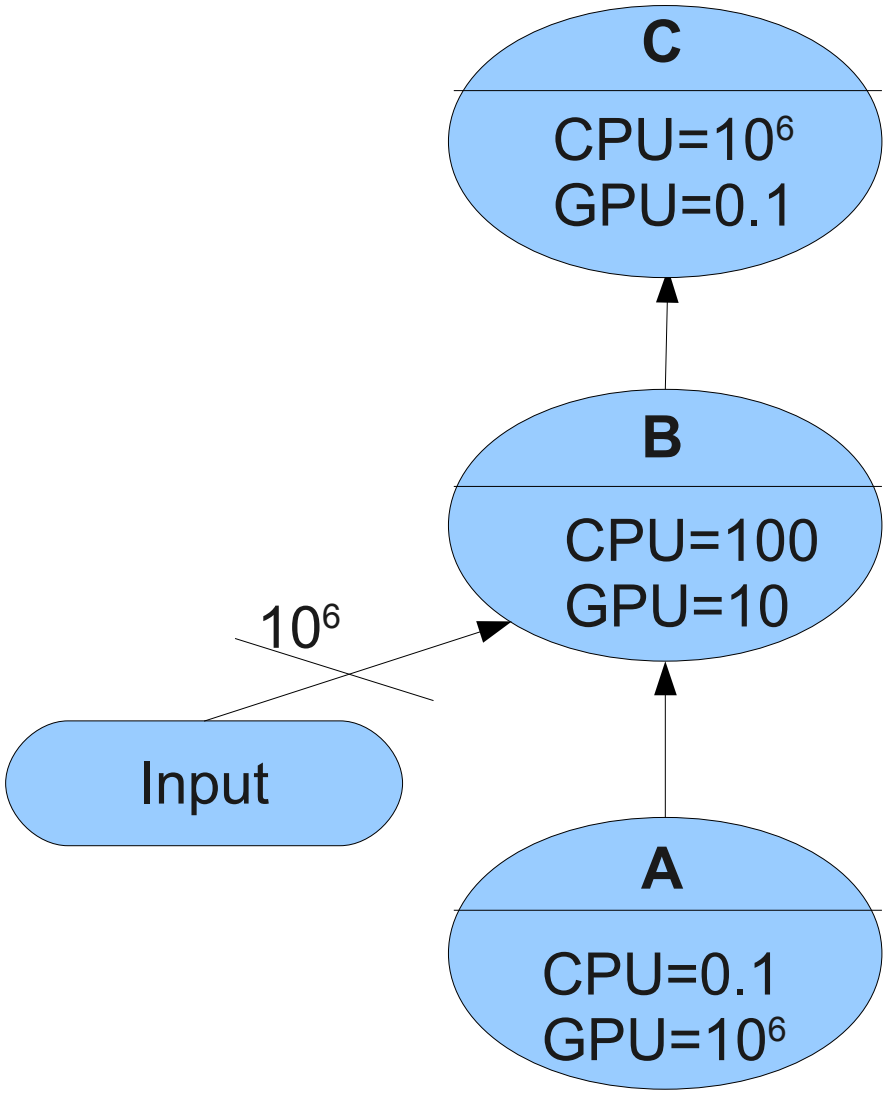
- Input and output should be moved to/from GPU
- Greedy (CPU-GPU) = 70J
- Optimal (GPU-GPU)= 65J



Needed: **communication-aware energy-optimized assignment**

How bad can it get?

How bad can it get?



As bad as we want...

More formally ...

- Task dependency tree T(**V,E,P,D**)

  V – tasks, **E** – data dependencies

  P_v(i) – cost of execution on processor *i*

  D_v(i→j)– cost of data transfer between *i* and *j*

- Goal: find *S:* **V** → {CPU:GPU} to minimize

$$\sum_{v \in V} \left(\underbrace{P_v[S(v)]}_{\text{Execution of task } v} + \underbrace{\sum_{w \in N_v} D_v[S(w) \rightarrow S(v)]}_{\text{Communications to task } v} \right)$$

Appears to be NP-hard...

- ***Resembles*** communication-aware parallel scheduling of DAGs on multiprocessors

- NP-hard even for
  - dependency **trees**,
  - for fixed number of processors,
  - with non-unit execution and communication costs

Appears to be NP-hard...

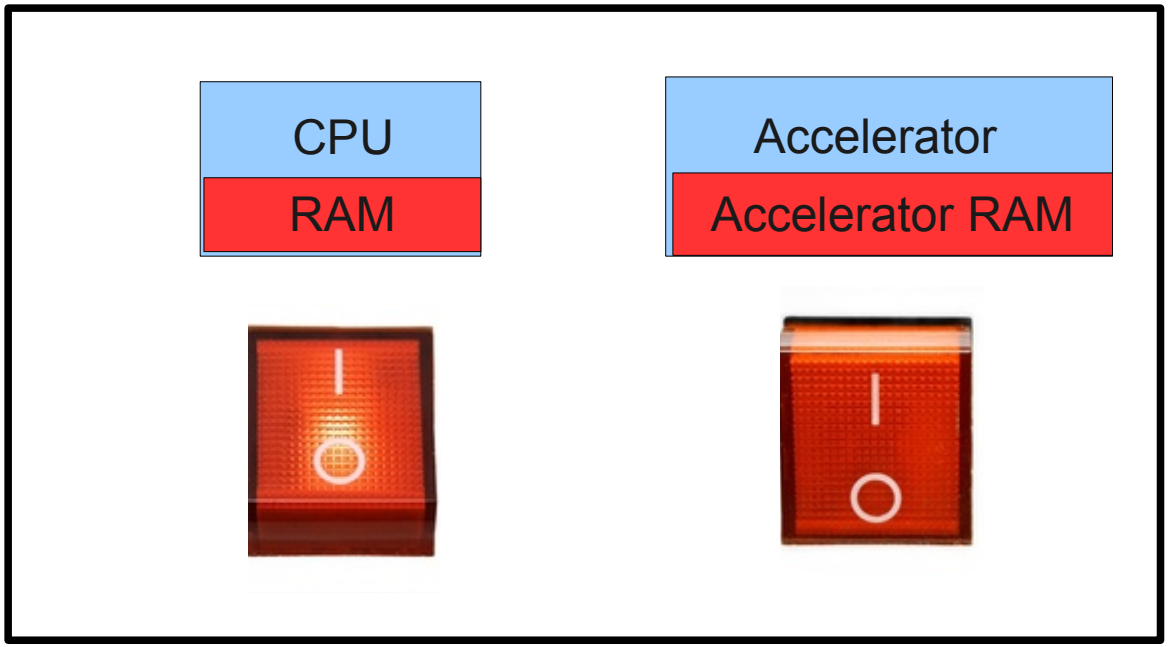
- ***Resembles*** communication-aware parallel scheduling of DAGs on multiprocessors

- NP-hard even for
  - dependency **trees**,
  - for fixed number of processors,
  - with non-unit execution and communication costs

But is it really the same problem?

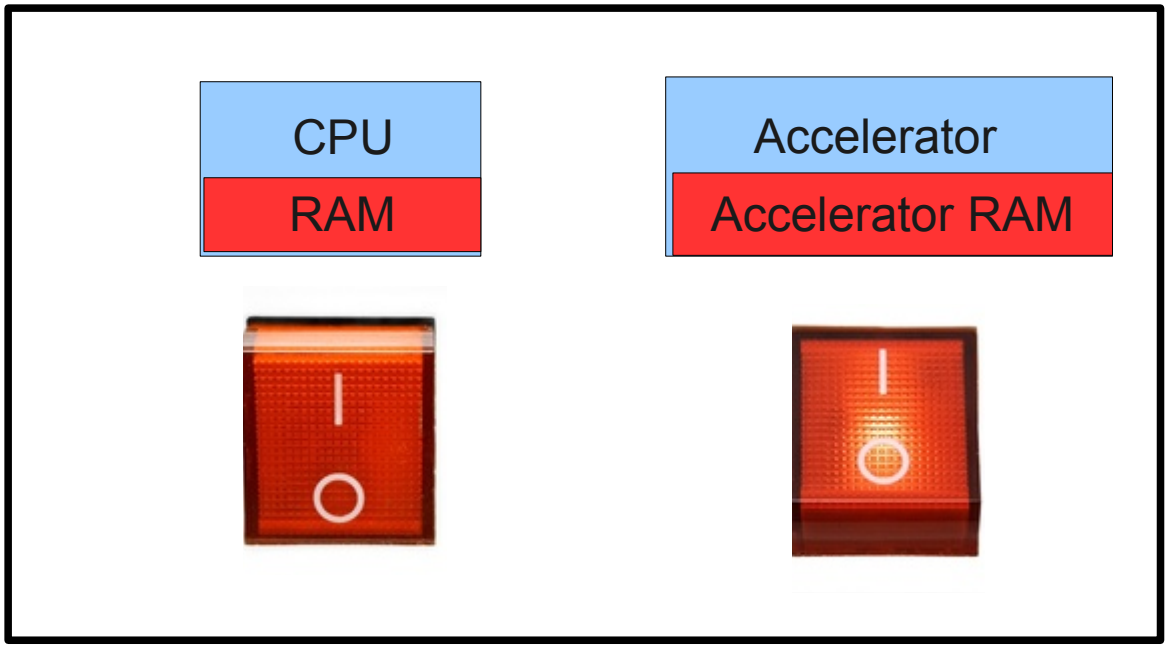
Energy-efficient architecture

CPU-intensive workload

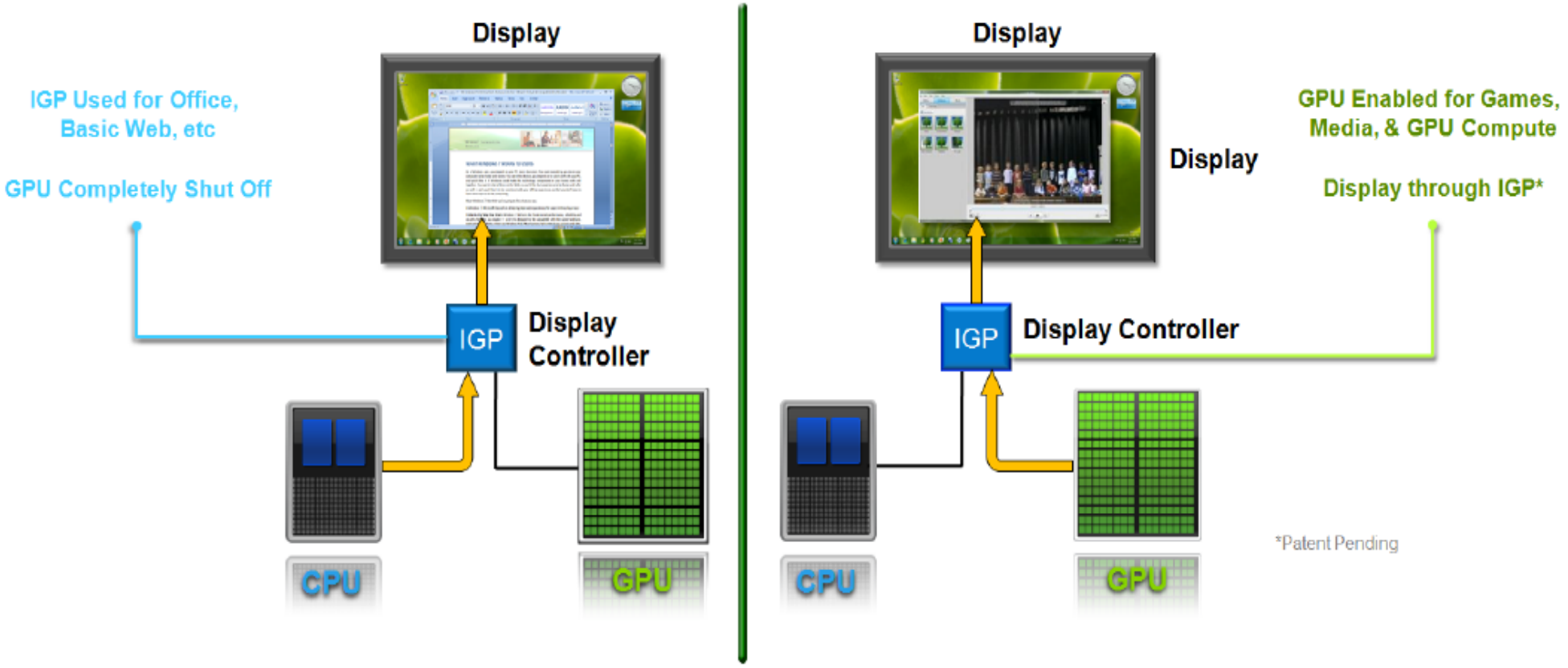


Energy-efficient architecture

Accelerator-intensive workload



Example – NVIDIA OPTIMUS



From NVIDIA OPTIMUS White Paper

Energy optimization is easier!

- Assume we found minimum-energy schedule
- Two executions are **energy-equivalent:**
  - All processors are used **concurrently**
  - Processors used **one-at-a-time,** unused are **powered off**

Energy optimization is easier!

- Assume we found minimum-energy schedule
- Two executions are **energy-equivalent:**
  - All processors are used **concurrently**
  - Processors used **one-at-a-time,** unused are **powered off**
- We refer to this problem as

  Energy-efficient acceleration assignment
- Runtime becomes secondary optimization

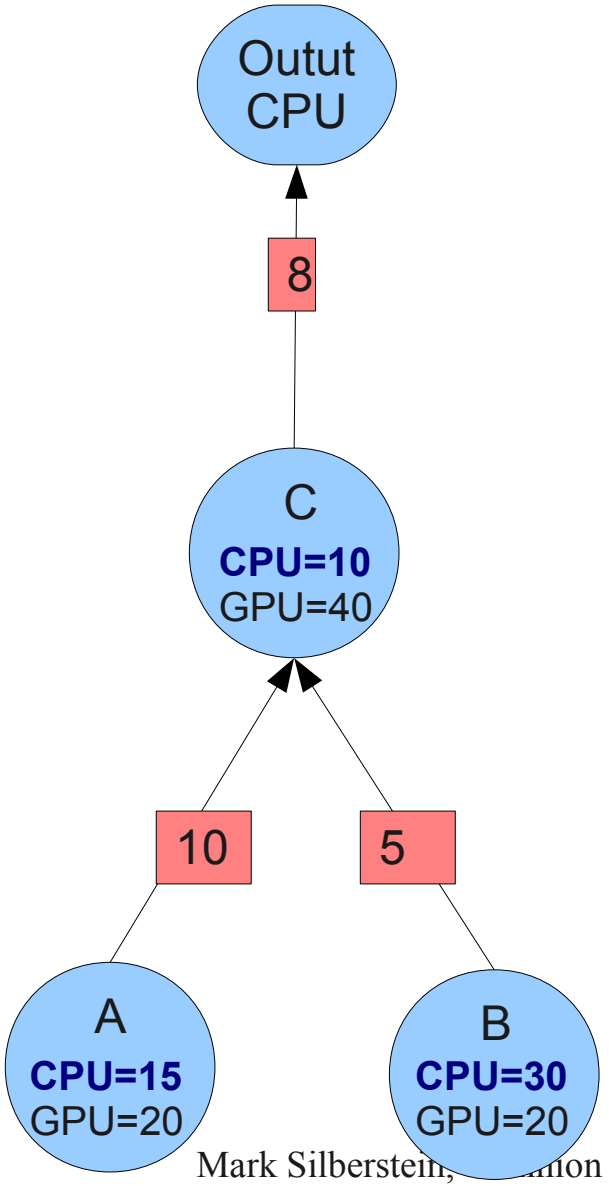
Dynamic programming

- Observation: assignment in one branch does not affect the assignment in another
- Forward step:
  - Traverse tree from the leaves to the root
  - Update the best costs for node execution on CPU and GPU, given best costs of its descendants

Dynamic programming

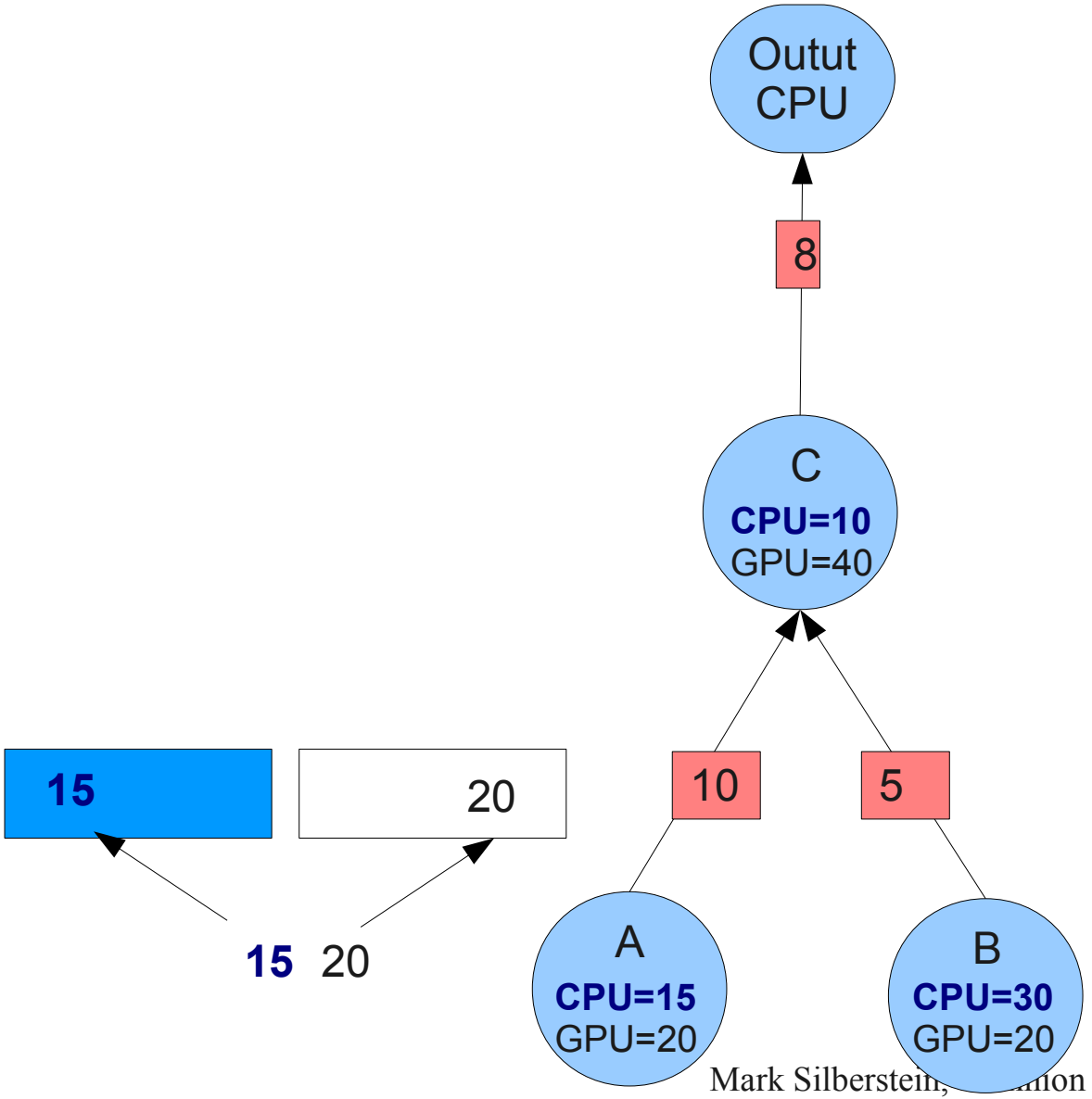
- Observation: assignment in one branch does not affect the assignment in another
- Forward step:
  - Traverse tree from the leaves to the root
  - Update the best costs for node execution on CPU and GPU, given best costs of its descendants
- Backward step:
  - Traverse tree from the root to the leaves
  - Fix the best cost given the root on a CPU

Forward step

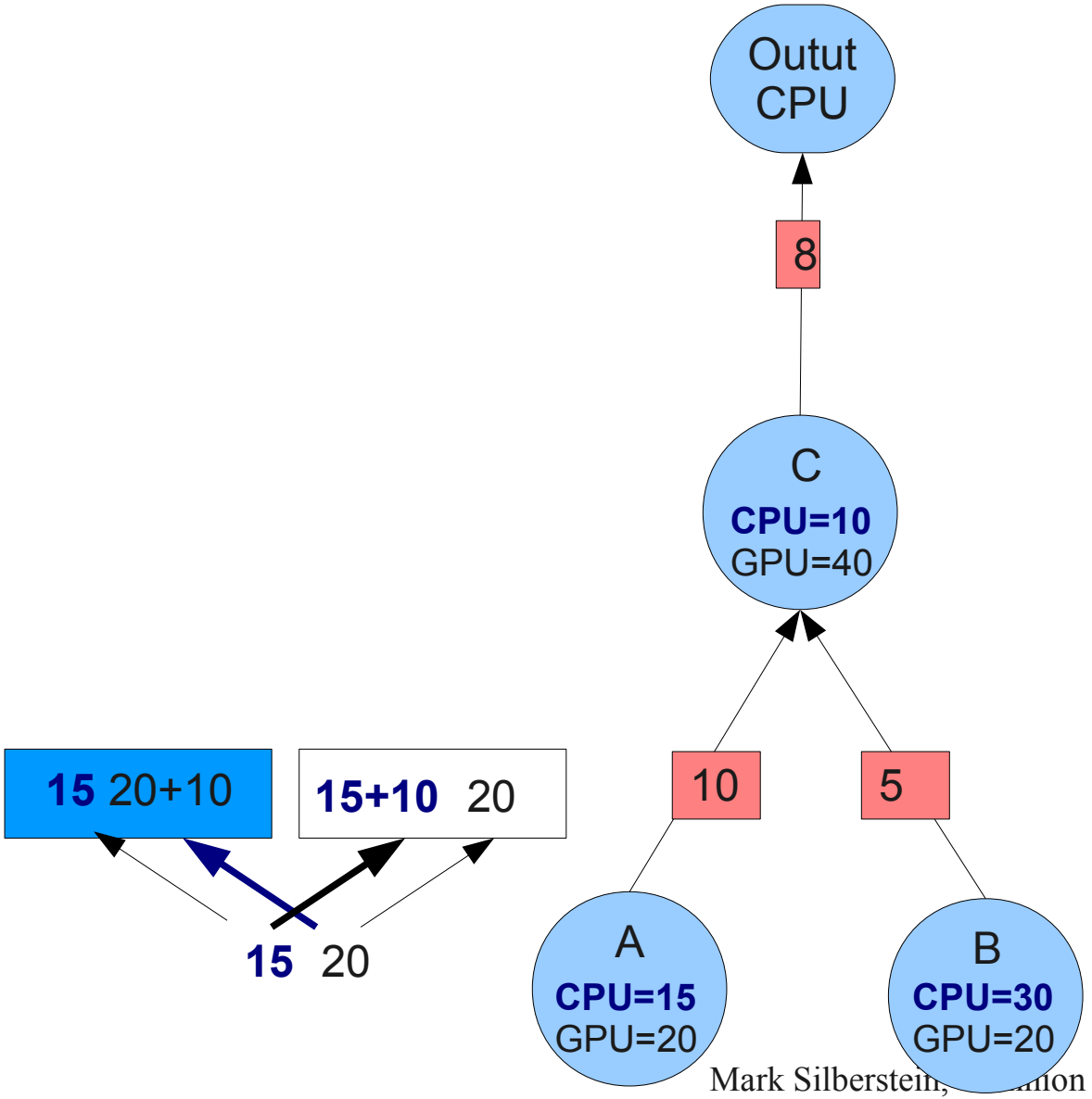


15 20

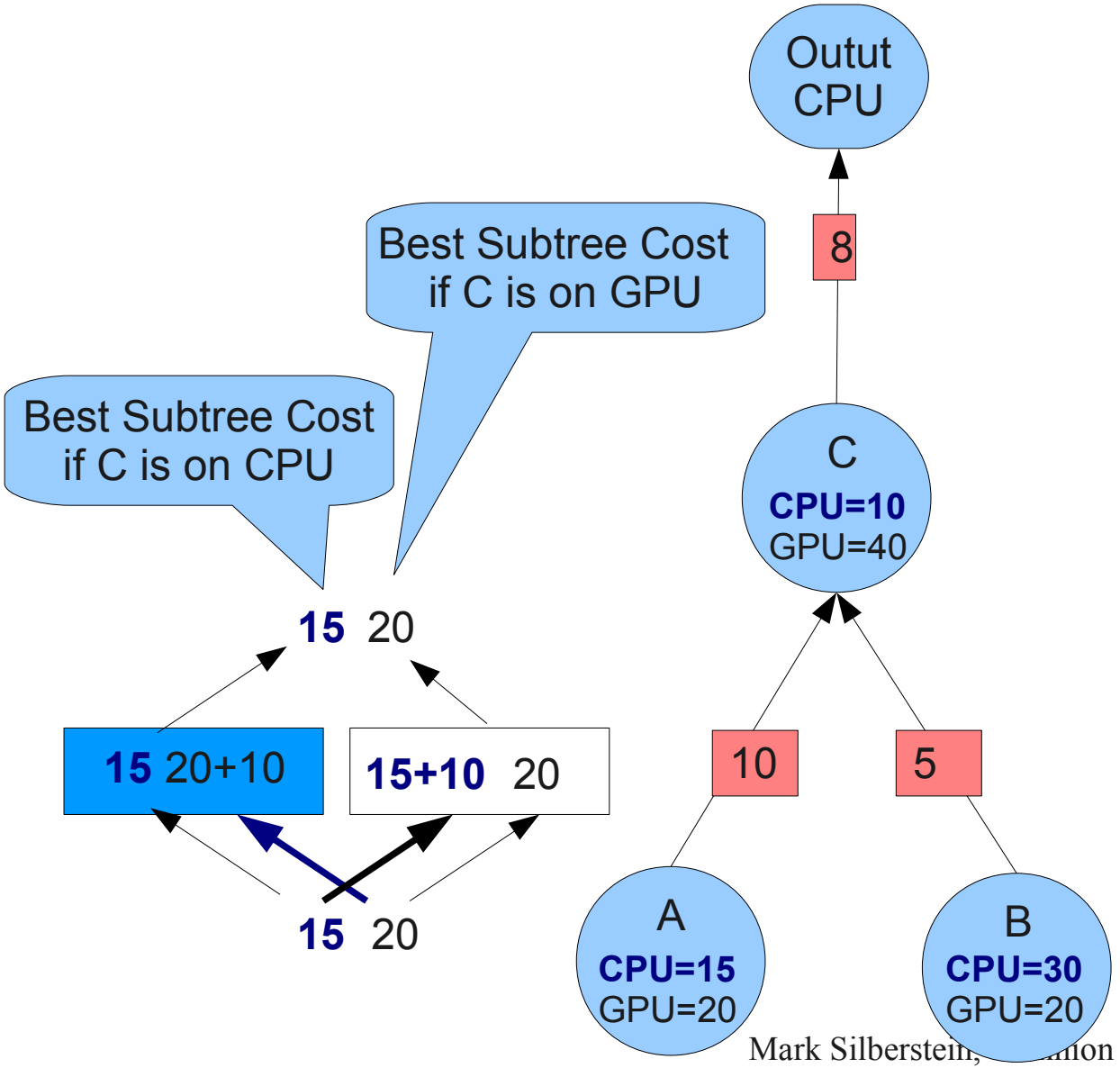
Forward step



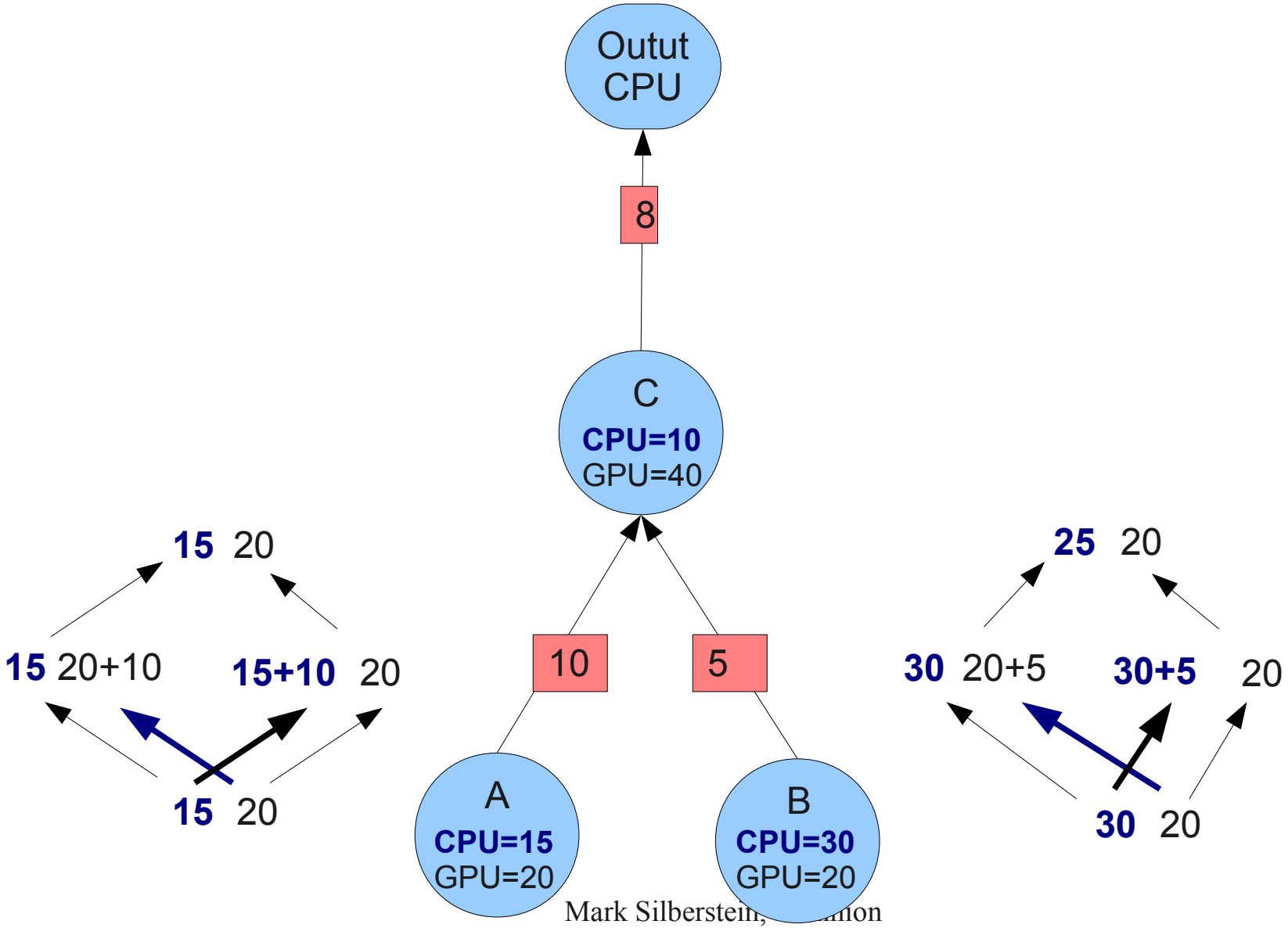
Forward step



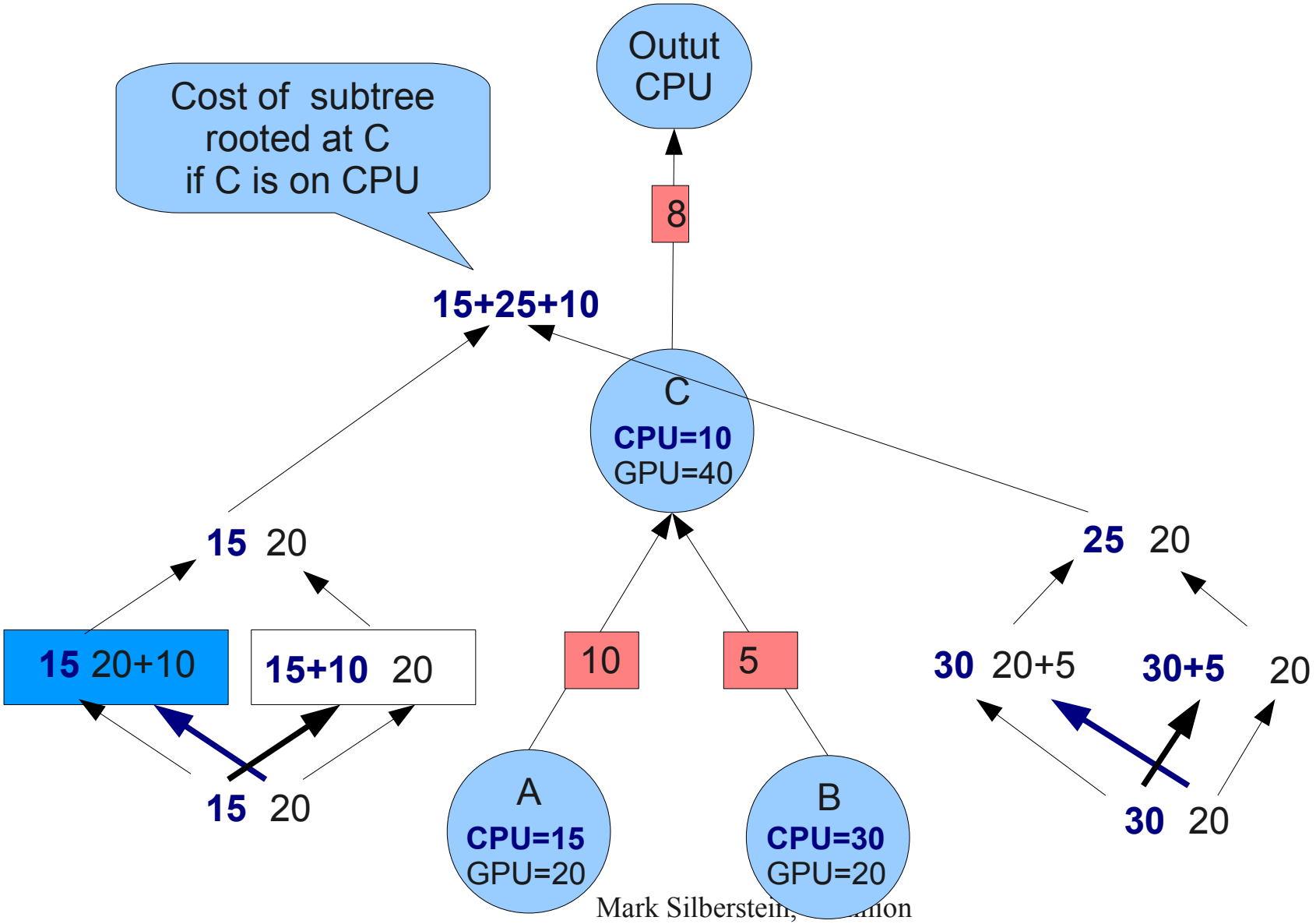
Forward step



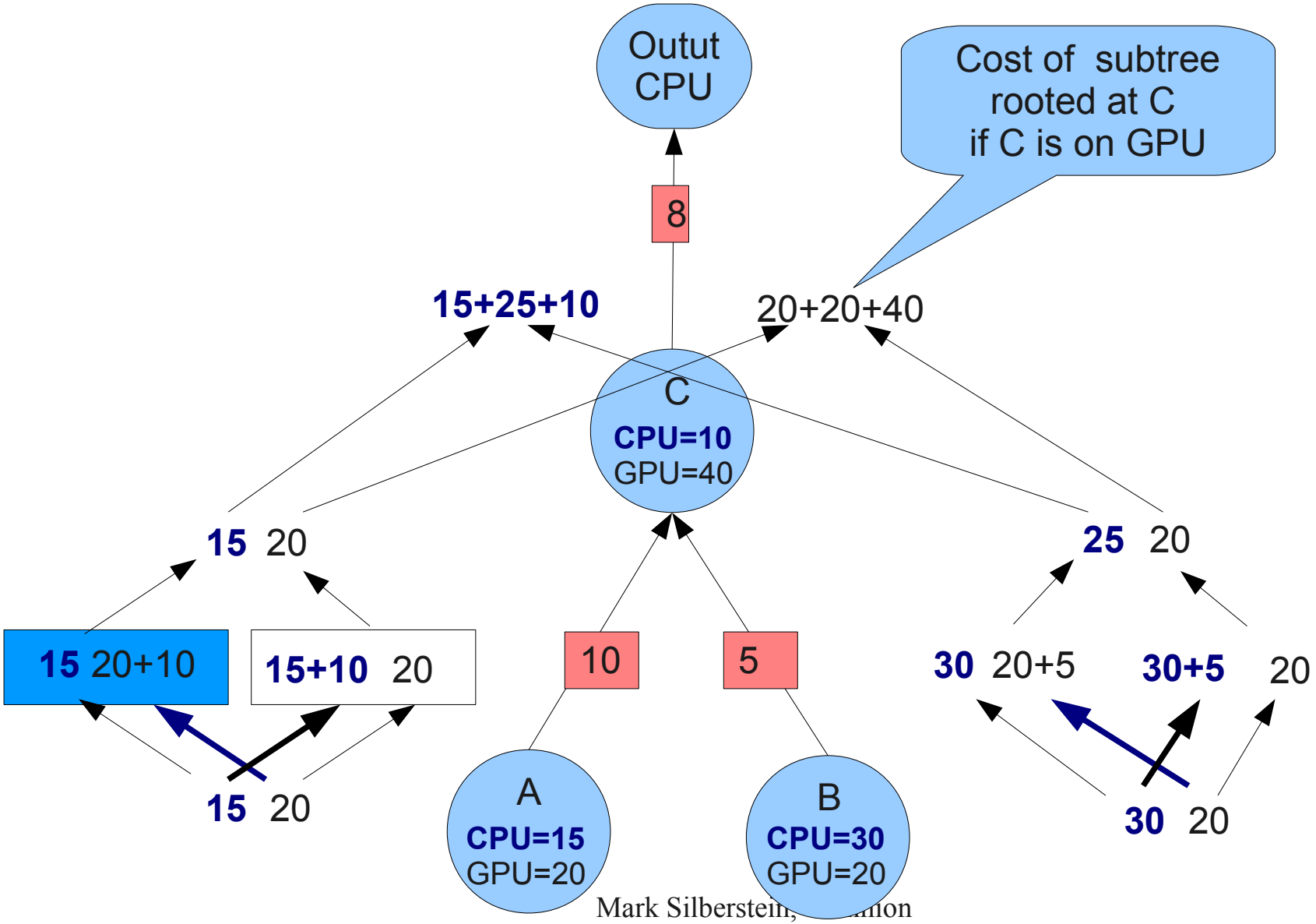
Forward step



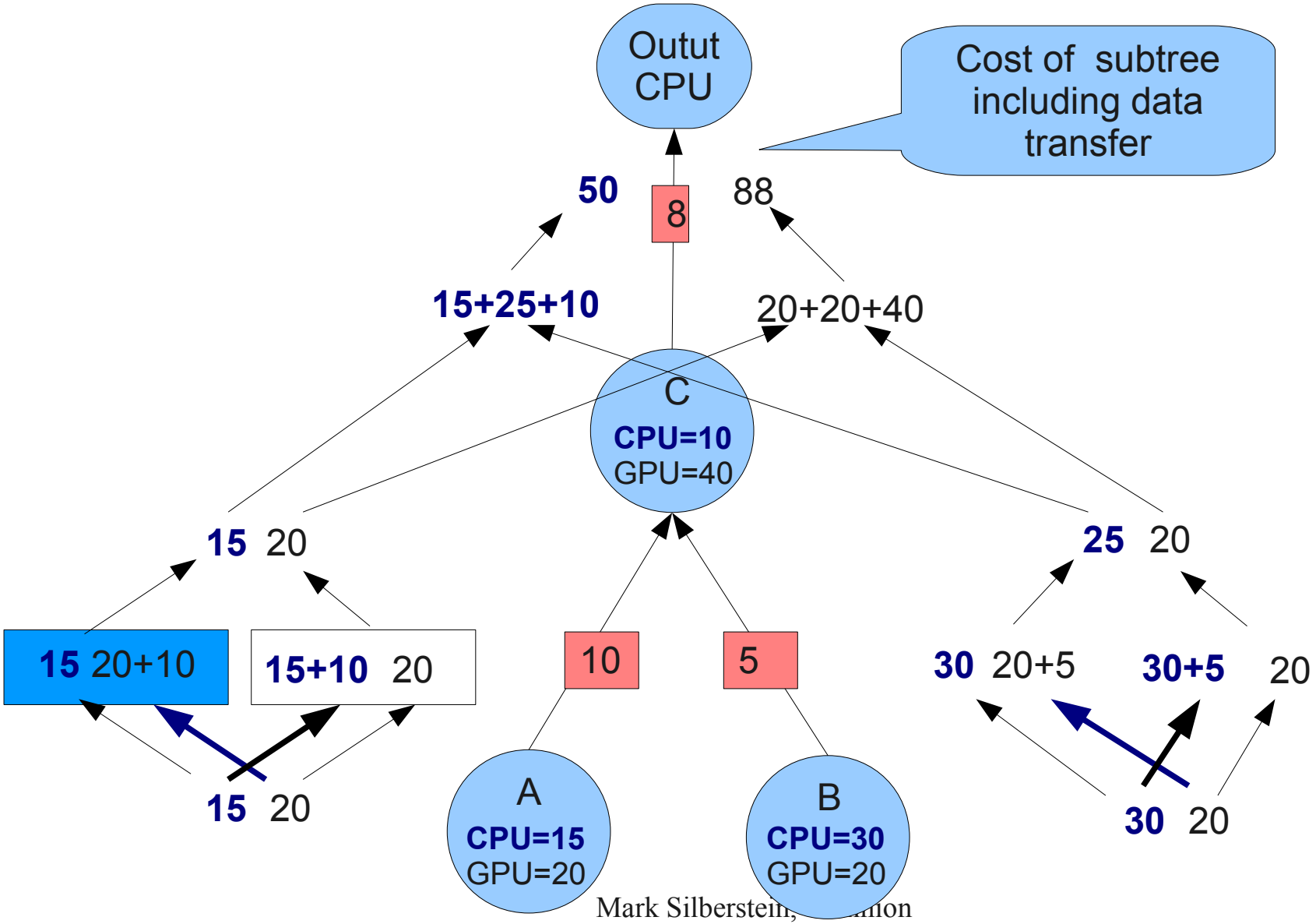
Forward step



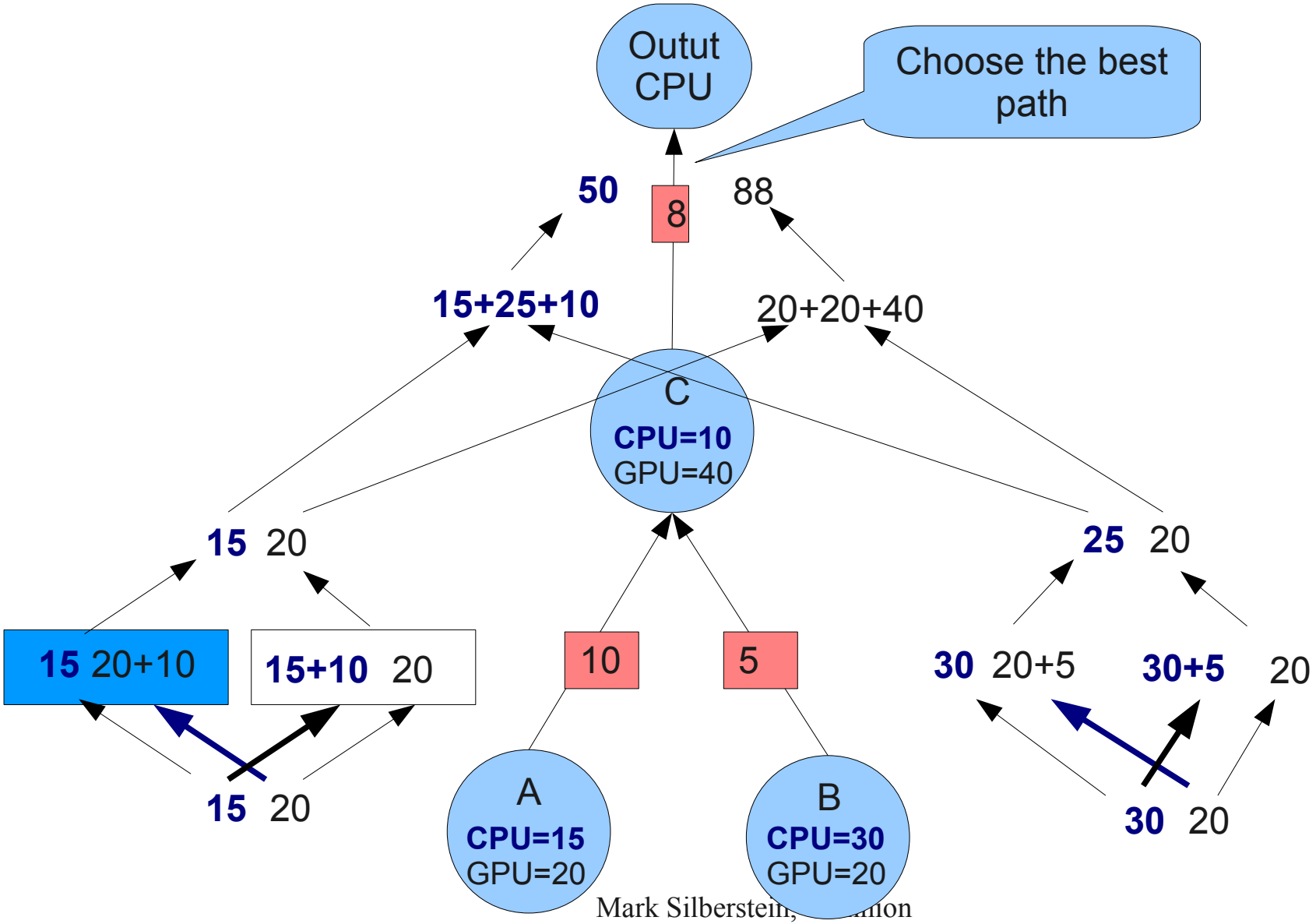
Forward step



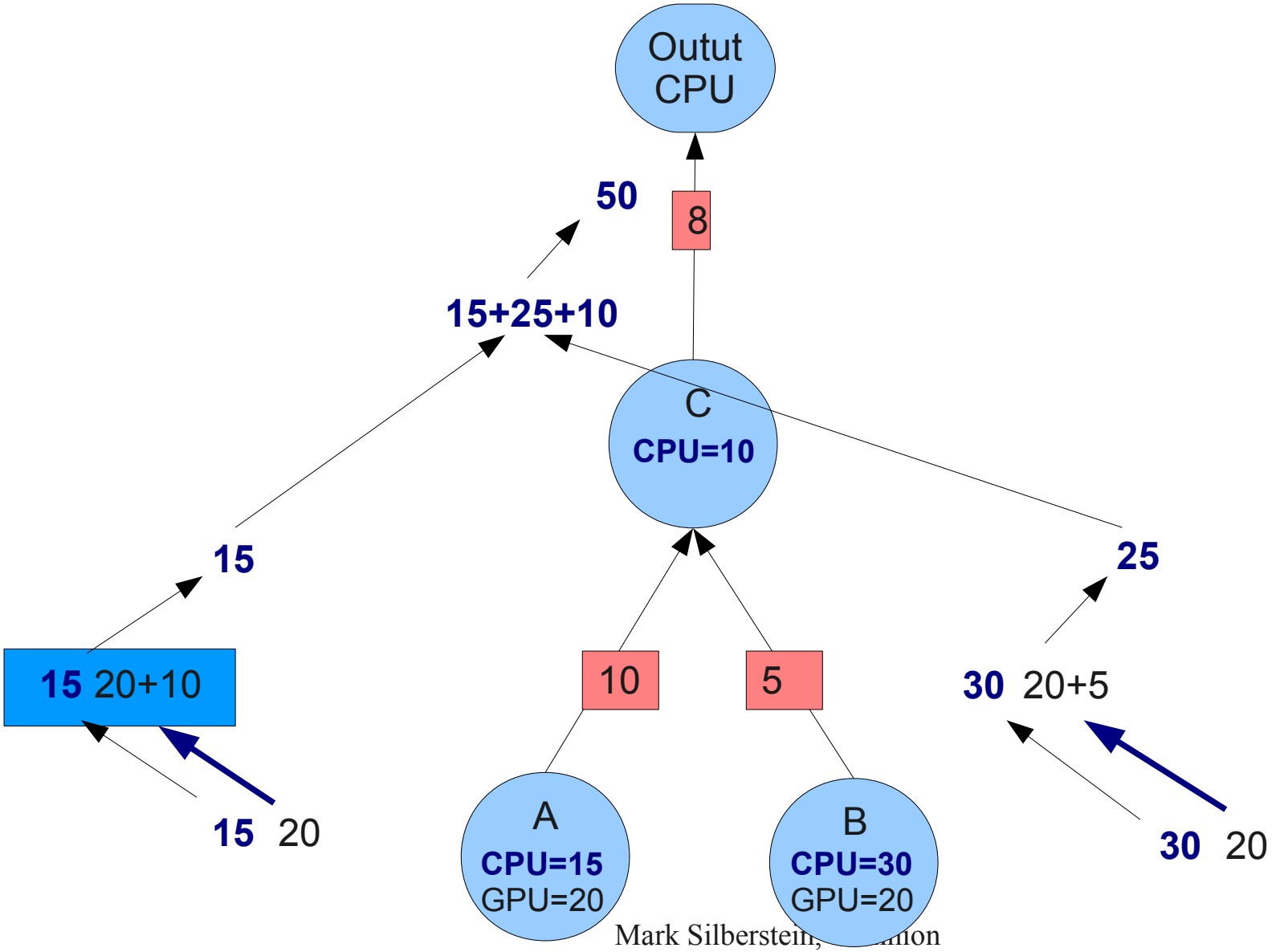
Forward step



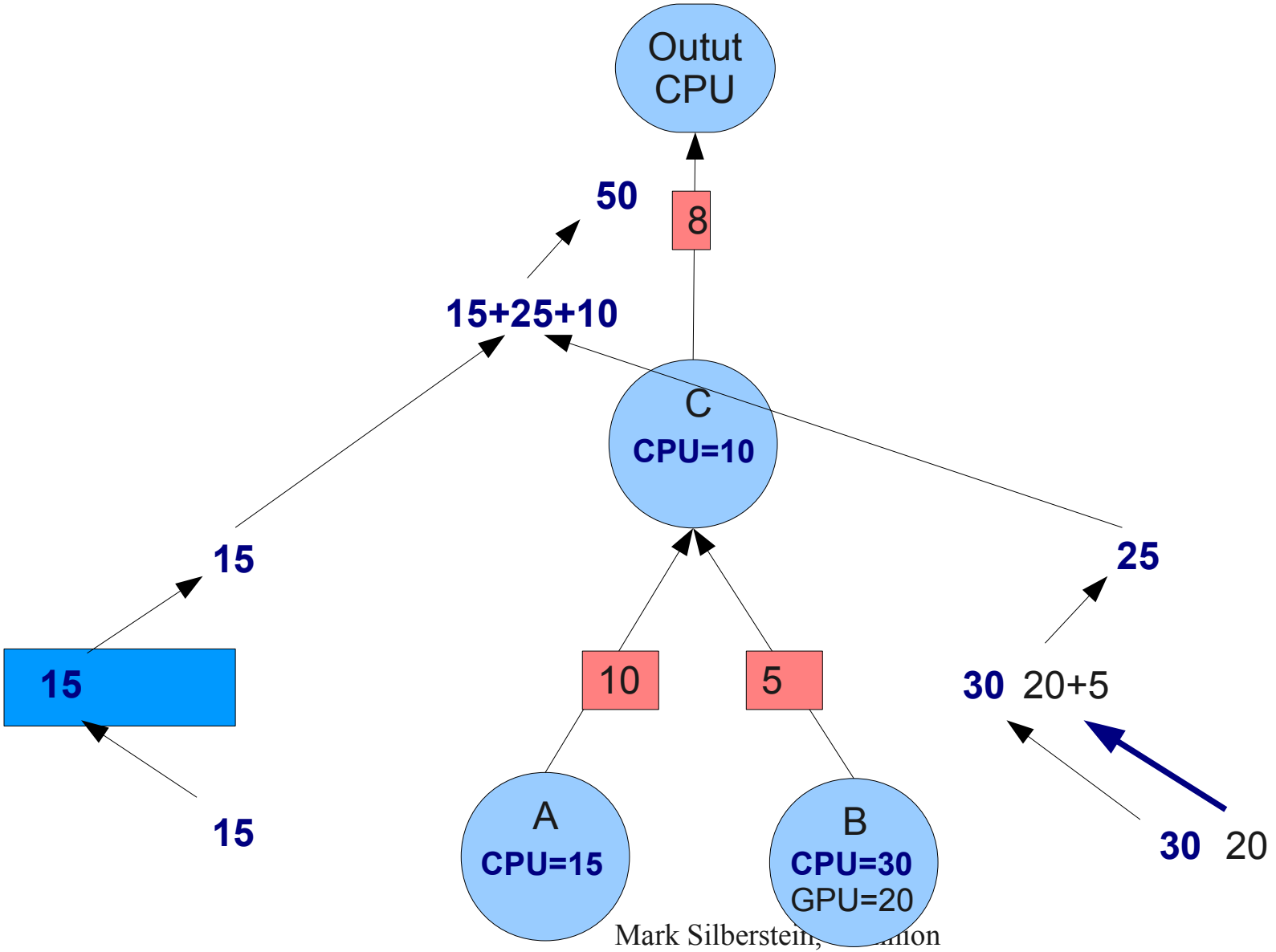
Backward step



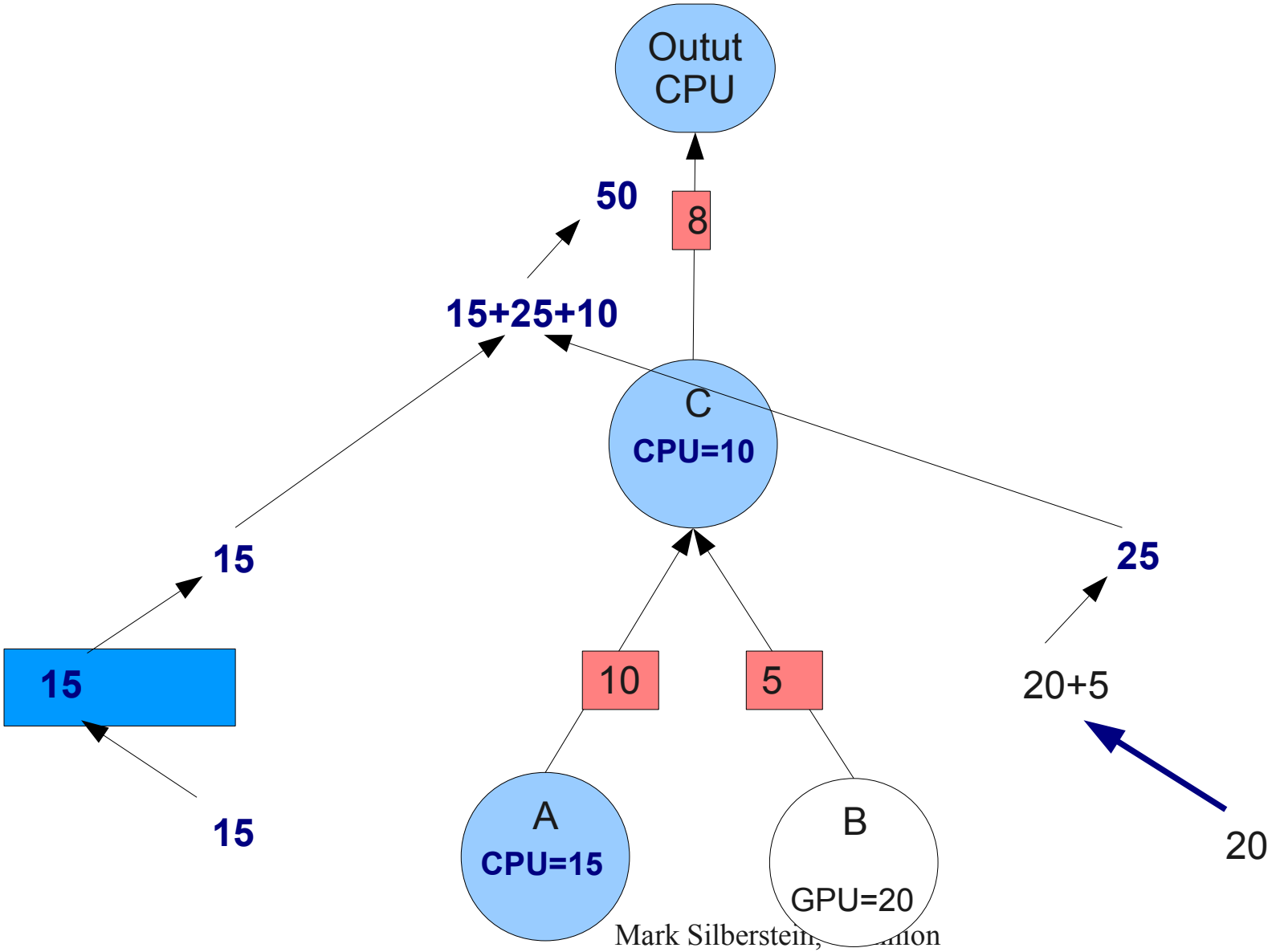
Backward step



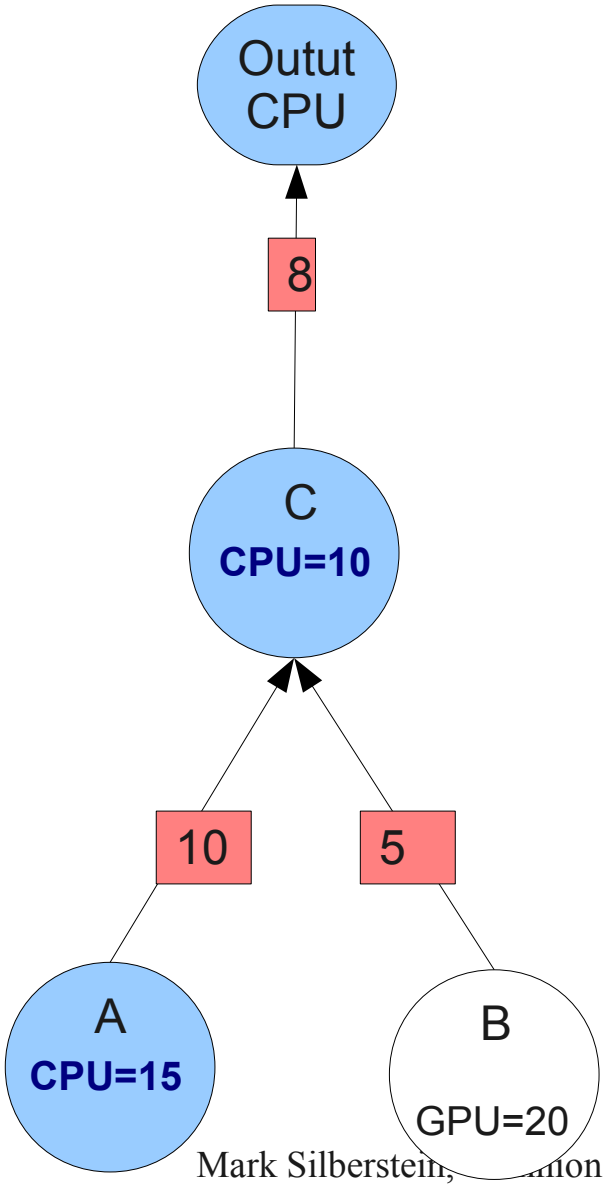
Backward step



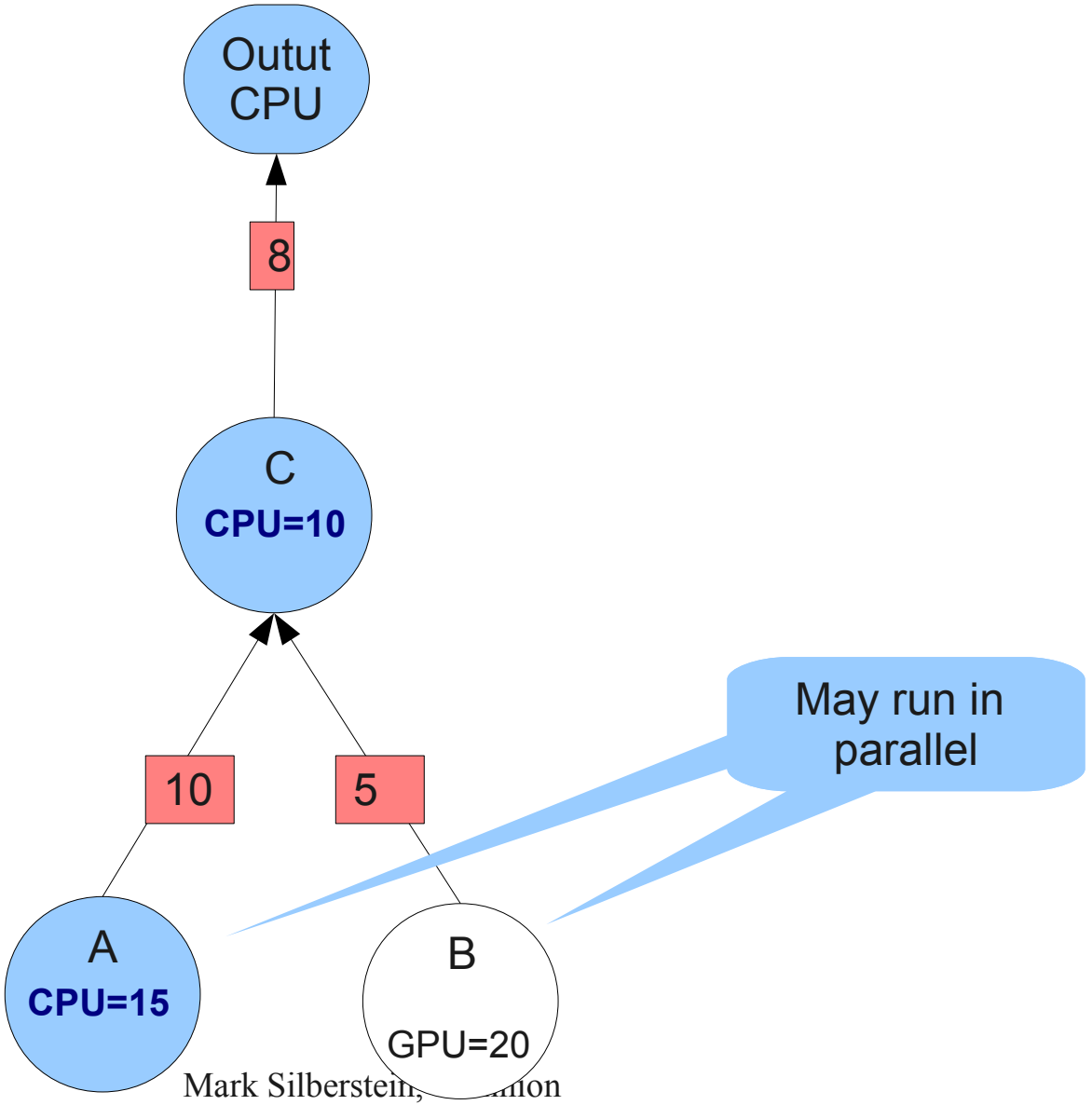
Backward step



Final assignment



Final assignment



Dynamic programming

- Forward step:
  - Traverse tree from the leaves to the root
  - Update the best costs for node execution on CPU and GPU, given best costs of its descendants
- Backward step:
  - Traverse tree from the root to the leaves
  - Fix the best cost given the root on a CPU
- Complexity: O(#Tasks x #Processors)

Experiments

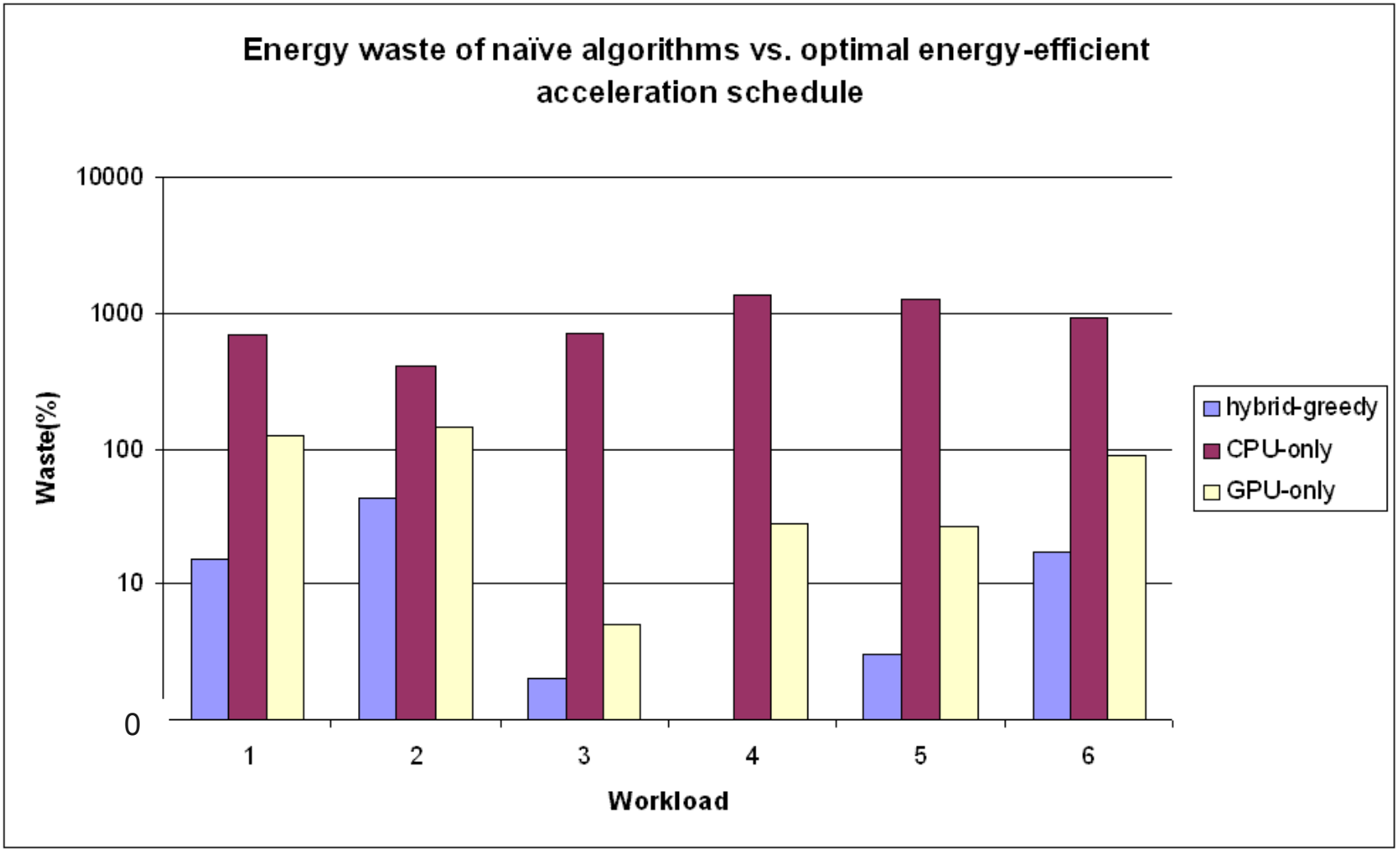
- 6 task dependency trees
- Source: inference in large probabilistic networks
  - Used in genetic analysis
- GPU: NVIDIA GTX285
- CPU: Quadcore AMD Phenom 9500

Tasks	CPU time per task (ms)	GPU time per task (ms)	Speedup per task	Transfer sizes between tasks (KB)
1194	(0.01 / 11 / 667)	(0.2 / 0.5 / 15.9)	0.06 / 7.7 / 104	0.07 / 633 / 12K

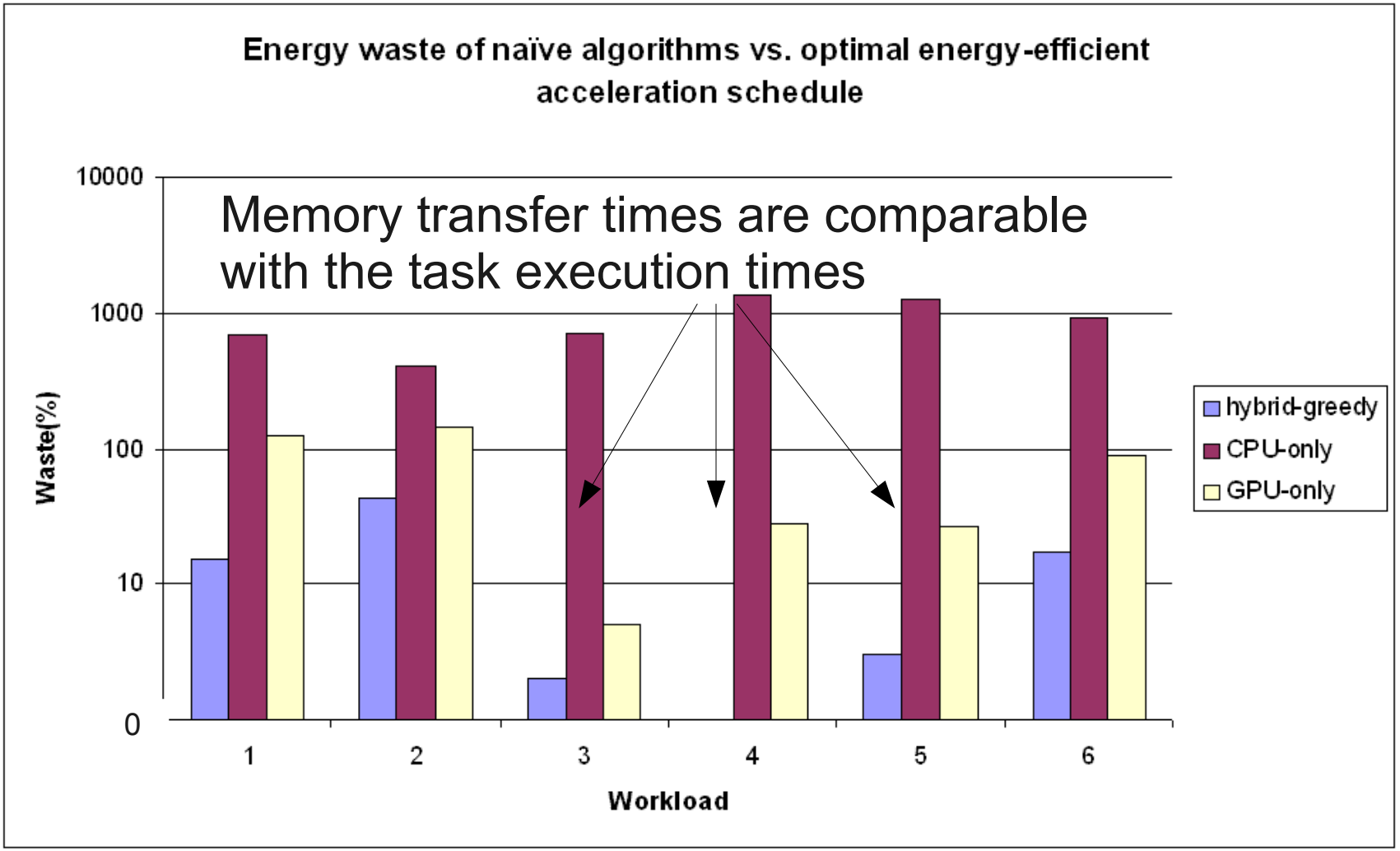
Methodology

- Predict the energy spent on each task and transfer
  - Runtime estimate x *observed* power consumption
  - Transfer time estimate x *observed* power consumption
- Run the assignment algorithm
  - Hybrid-greedy, only-GPU, only-CPU, exact
- Given the schedule, measure the actual execution and data transfer time of all tasks in the tree
- Scale by the observed power consumption

Hybrid strategy pays off



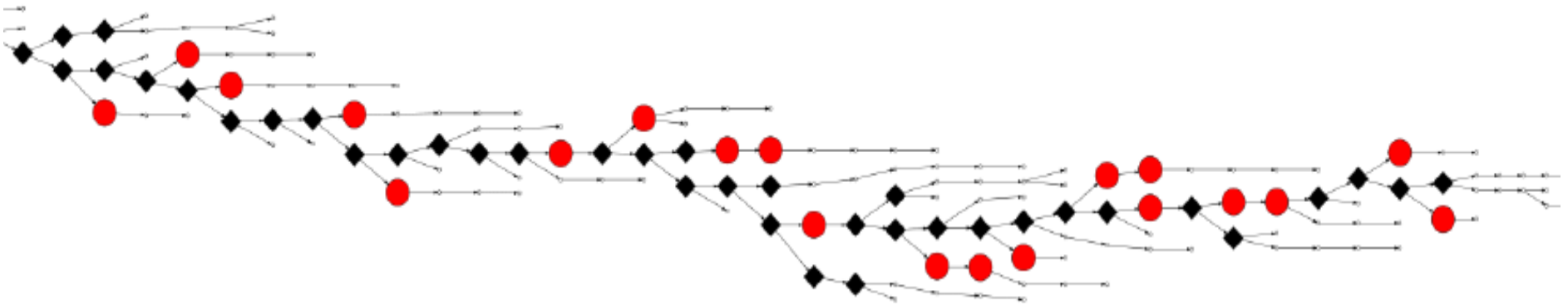
Hybrid strategy pays off



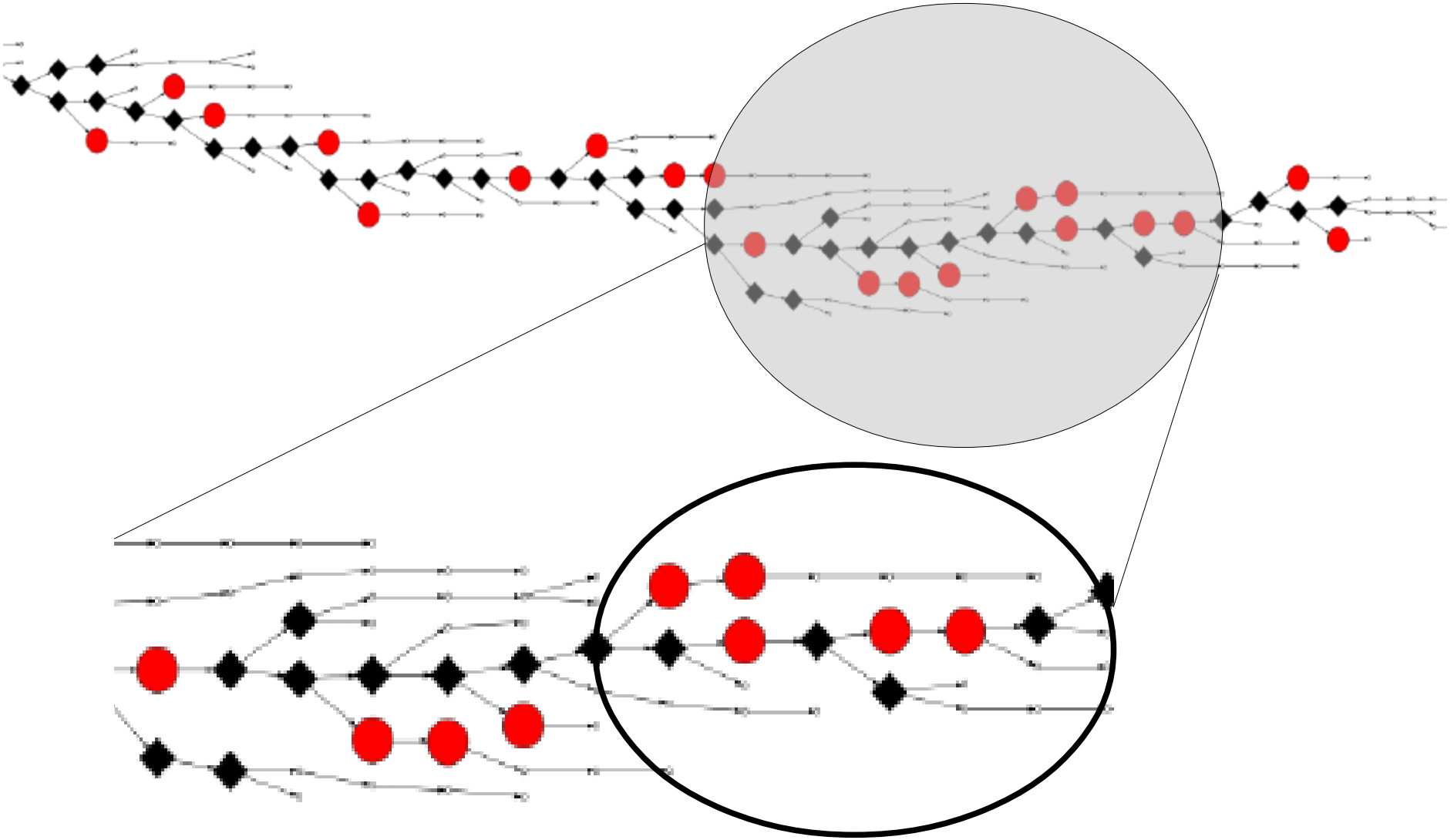
Hybrid greedy assigns fewer tasks to GPU

Total tasks	GPU tasks by Optimal	GPU tasks by Greedy
390	25	14
529	41	28
268	86	62
595	139	111
1194	301	230
505	46	21

Optimal algorithm avoids islands

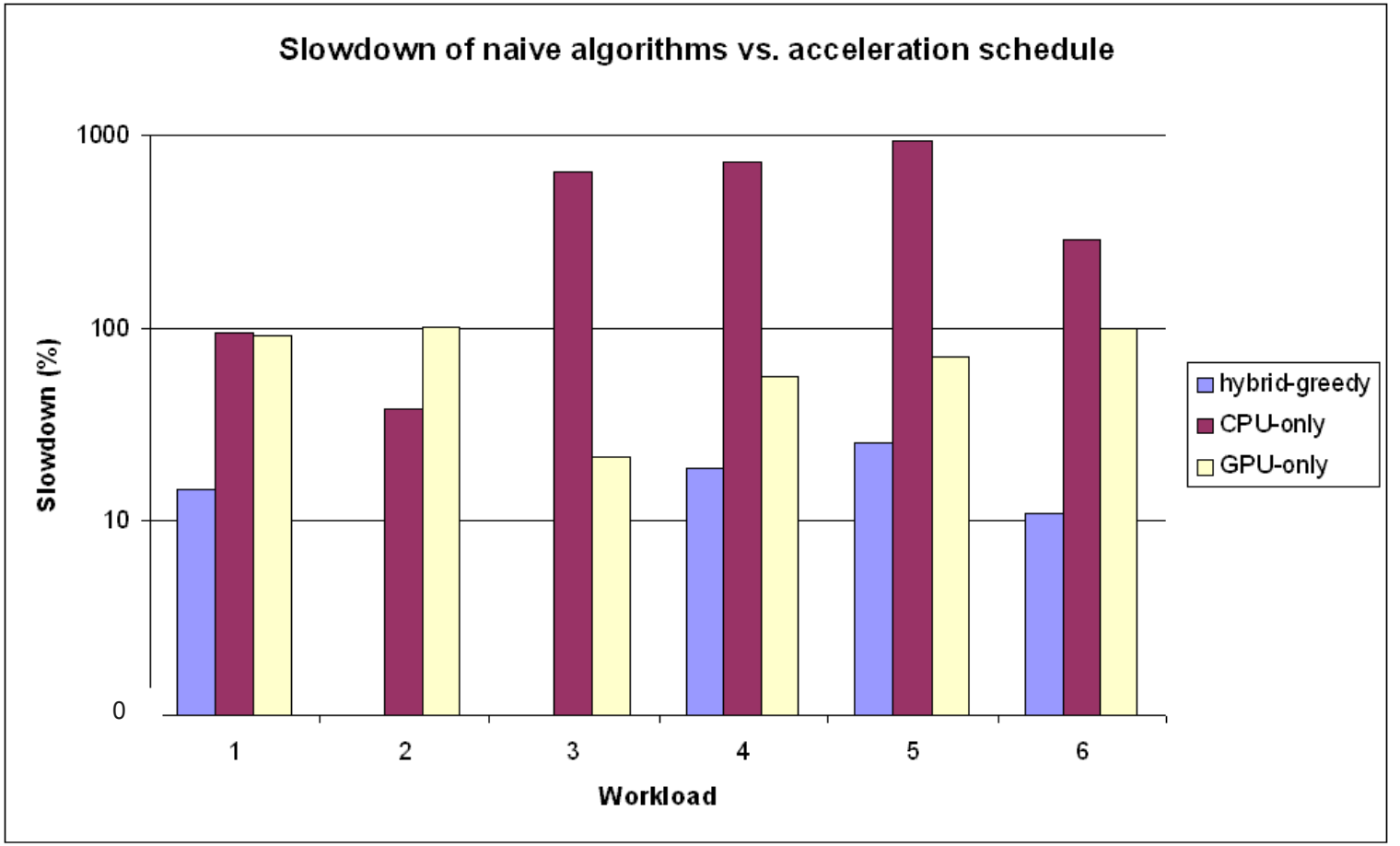


Optimal algorithm avoids islands



Runtime optimization

- The algorithm can be used to improve runtime
  - Exploiting the parallelism available in the schedule



Insights

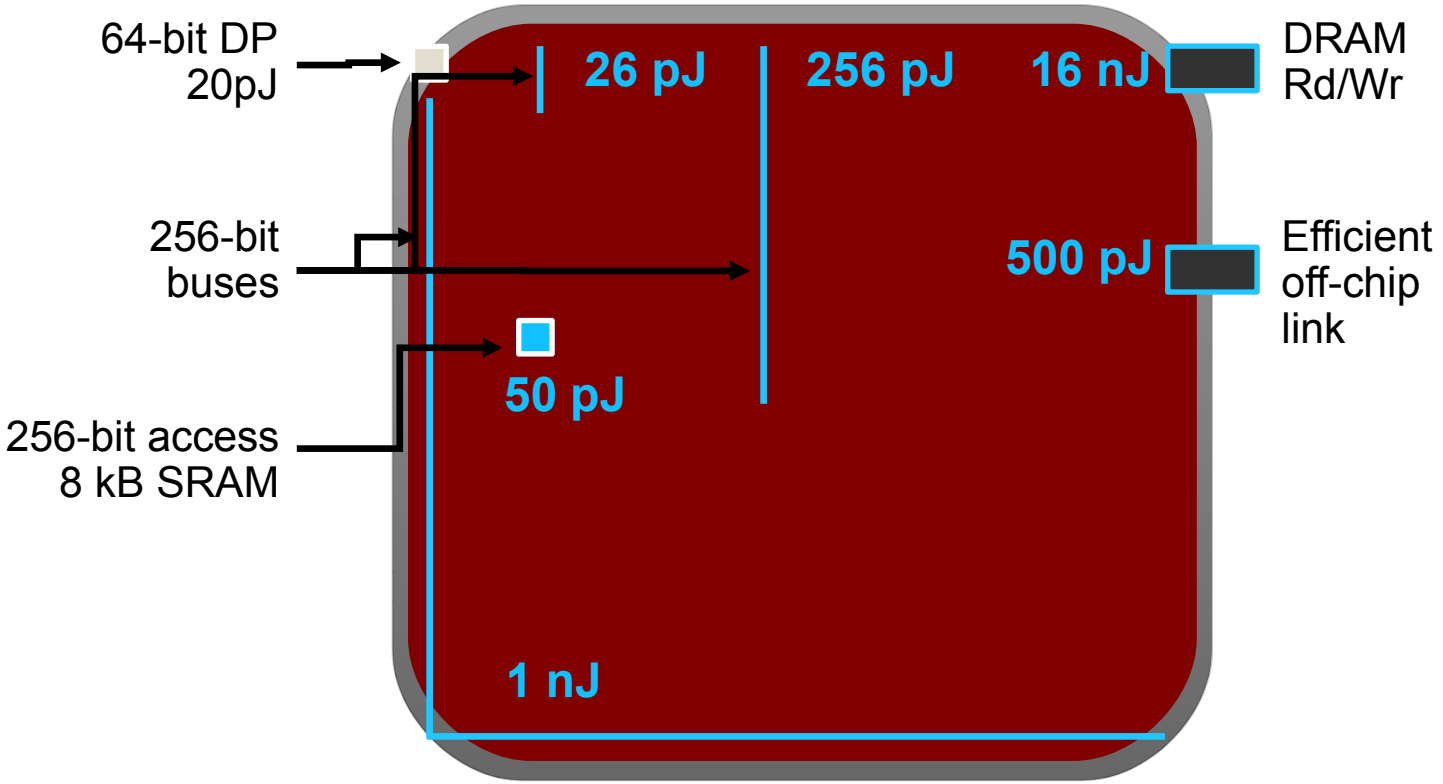
- Communication power cost may hinder GPU performance benefits
- “Power off” assumption simplifies scheduling algorithms
- Algorithm utility depends on the relative amount of communications

Future work

- Use as a part of a heuristic for DAG
- Joint runtime – power optimization
- Power measurements on real hardware
- Application to multicores

The High Cost of Data Movement

From the keynote IPDPS11, Prof. Dally, Stanford



Its not about the FLOPS
Its about data movement

Thank you!