Performance-Cost Optimization of Moldable Scientific Workflows

Marta Jaros

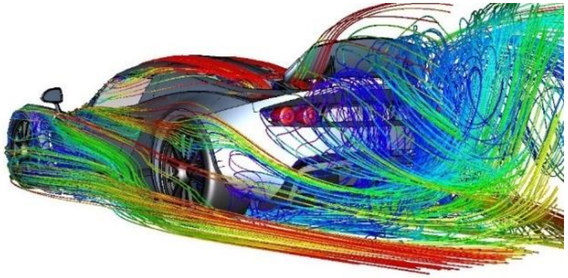
Biomedical Ultrasound Group (BUG)
Department of Medical Physics and Biomedical Engineering
University College London

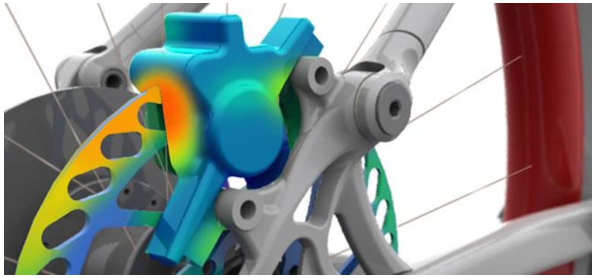
About Me

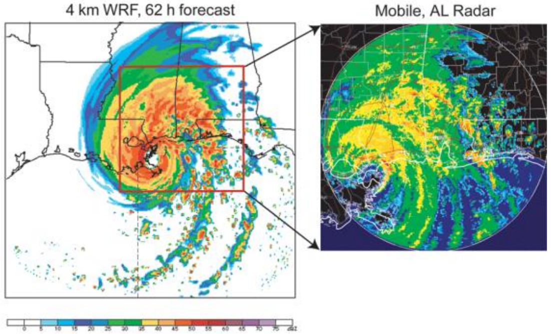
- Research assistant at Biomedical Ultrasound Group at UCL
- PhD student at the Faculty of Information Technology at Brno University of Technology (Supercomputing Technologies Research Group)

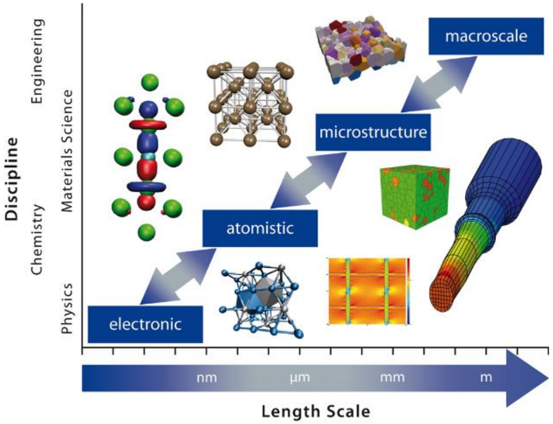
Motivation

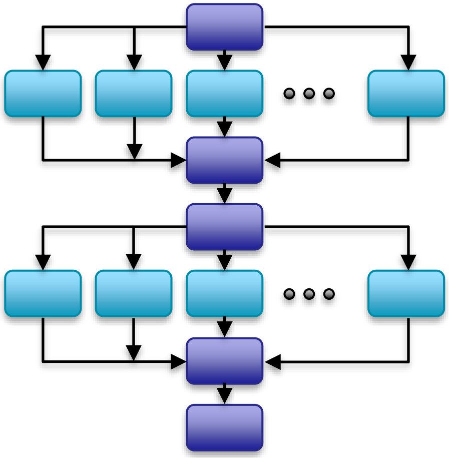
Simulations of Complex Natural Phenomena





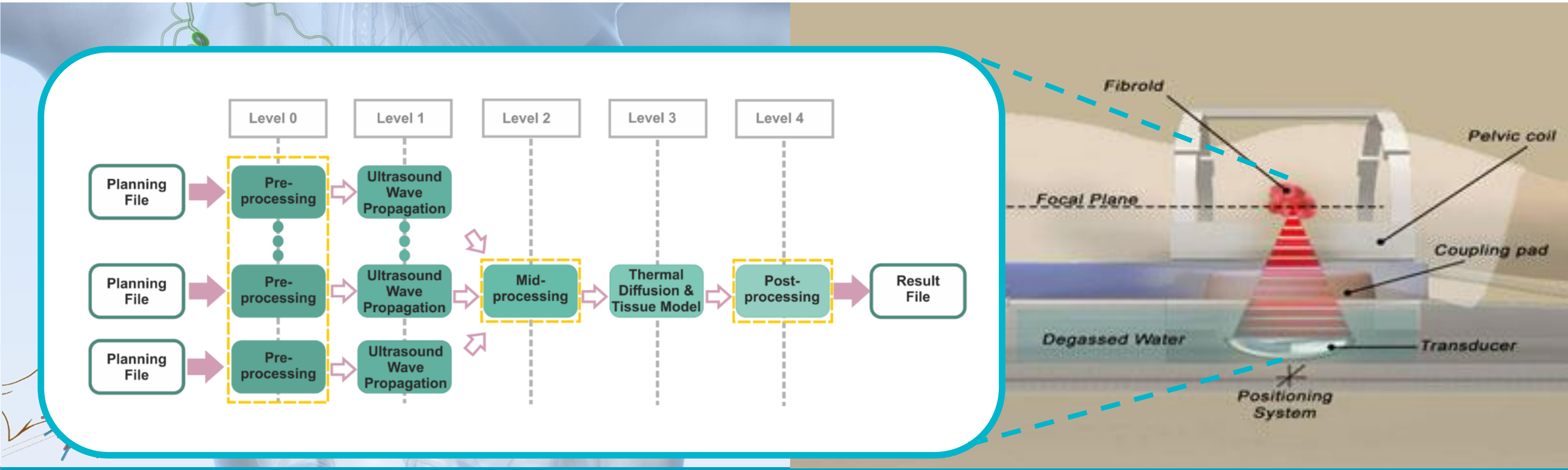






Case Study

- Ultrasound treatment planning
- Examples of medical applications:
  - Surgery planning
  - Targeted drug delivery
  - Neurostimulation



Level 0
Level 1
Level 2
Level 3
Level 4
Planning File
Pre-processing
Ultrasound Wave Propagation
Planning File
Pre-processing
Ultrasound Wave Propagation
Mid-processing
Thermal Diffusion & Tissue Model
Post-processing
Result File
Planning File
Pre-processing
Ultrasound Wave Propagation

Scientific Workflows

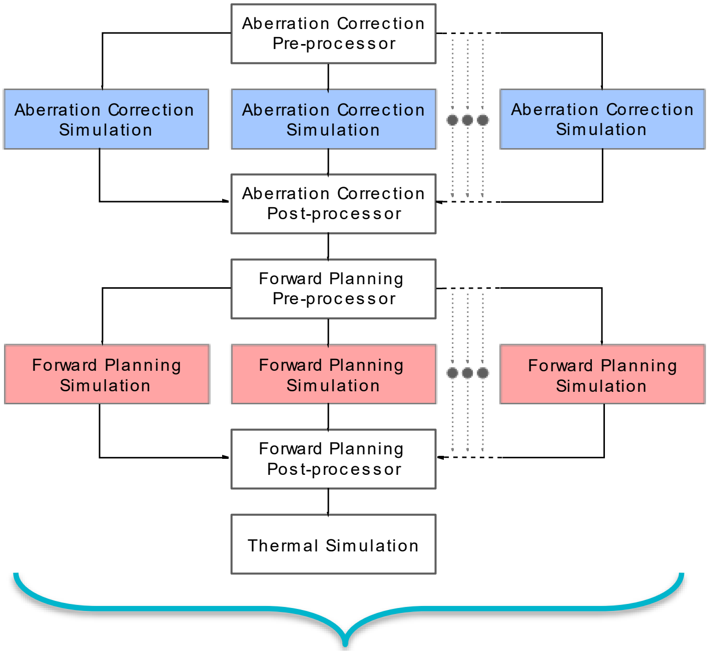
Execution Parameters:

Memory requirements

HW selection

Number of nodes / cores

Runtime definition


Aberration Correction Pre-processor
Aberration Correction Simulation
Aberration Correction Simulation
Aberration Correction Simulation
Aberration Correction Post-processor
Forward Planning Pre-processor
Forward Planning Simulation
Forward Planning Simulation
Forward Planning Simulation
Forward Planning Post-processor
Thermal Simulation

Workflow
(directed acyclic graph)

Challenges:

How to set the execution parameters?

How to fit into disk quotas?

How to provide error detection and recovery?

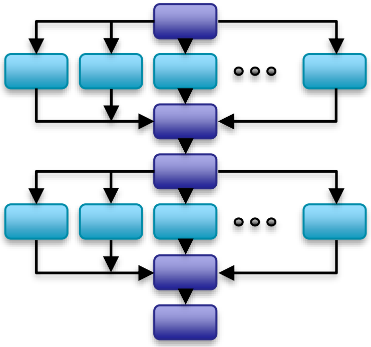
Workflow Executions on Remote Resources

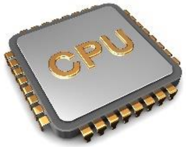
Define the workflow

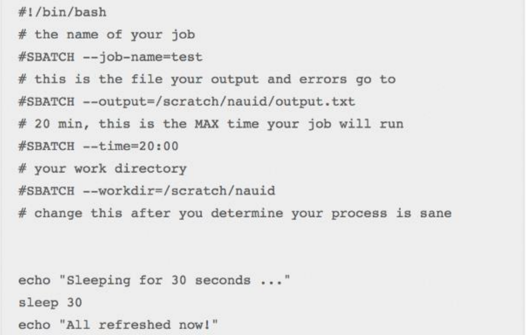
Define job dependencies

Define compute requirements

Manage file transfers







```
#!/bin/bash
# the name of your job
#SBATCH --job-name=test
# this is the file your output and errors go to
#SBATCH --output=/scratch/nauid/output.txt
# 20 min, this is the MAX time your job will run
#SBATCH --time=20:00
# your work directory
#SBATCH --workdir=/scratch/nauid
# change this after you determine your process is sane


echo "Sleeping for 30 seconds ..."
sleep 30
echo "All refreshed now!"
```

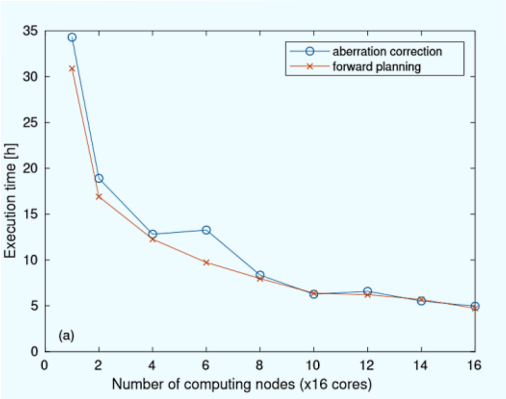
Execute the workflow

Monitor the workflow calculation

Take care of possible errors

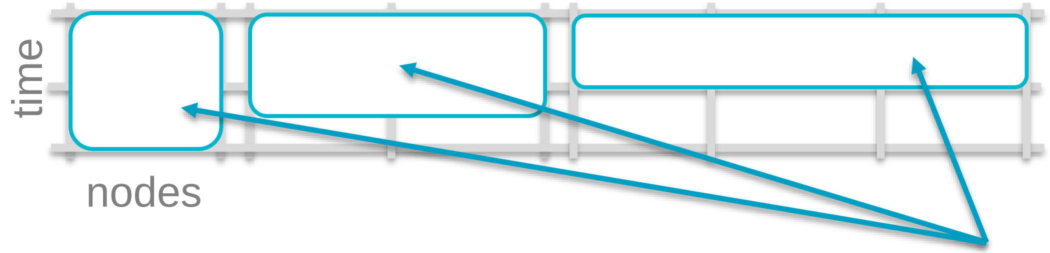
Download the results

Moldability

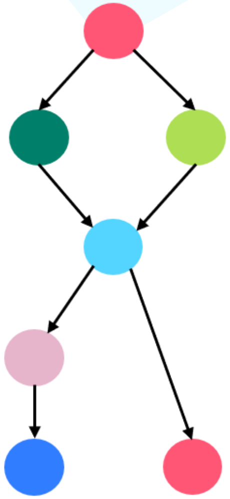


- Constant amount of useful work *W* (Amdahl’s law)
- More computing resources (nodes) -> shorter execution time
- Task scaling is never perfect -> growing cost

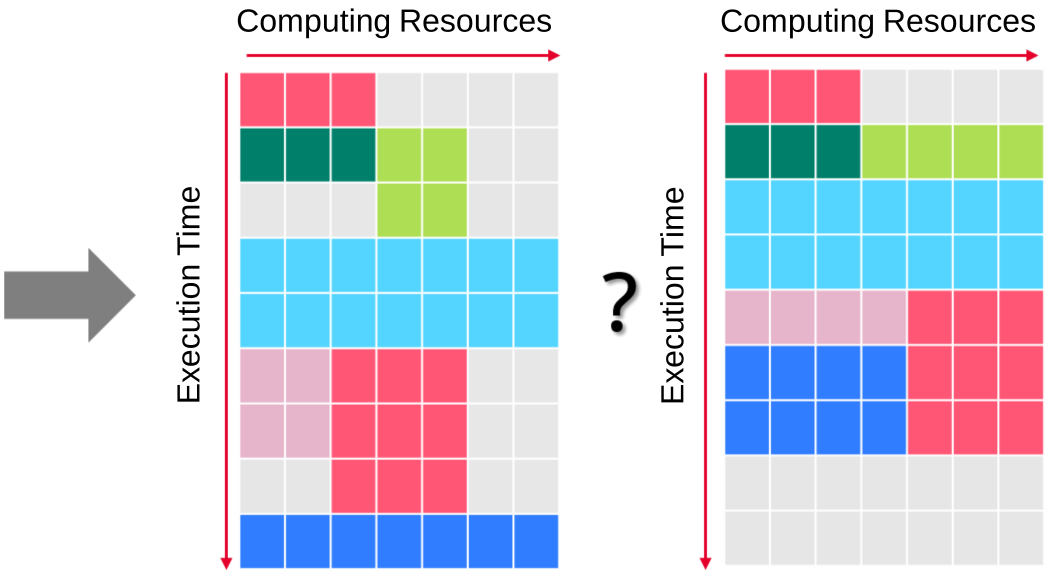
*cost = execution time * number of nodes*



Achieved computational efficiency impacts the size of the area

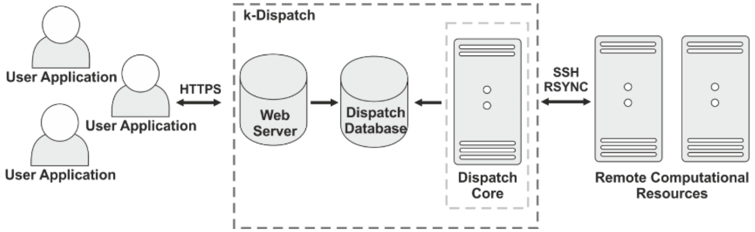


Task Graph
(Workflow)

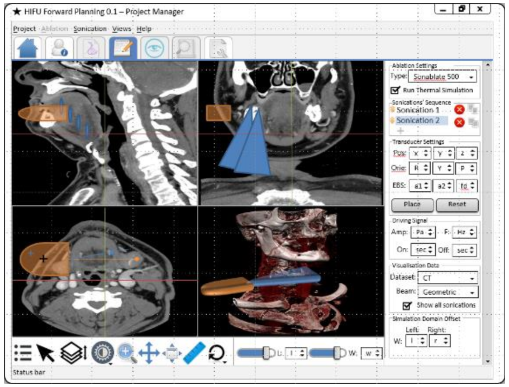


Tasks Mapping to Computational Resources and Free Time Slots

k-Dispatch



- Middle-ware
- HPC as a service
- Provides job submission, monitoring, reporting, fault tolerance, accounting, reporting


★ HIFU Forward Planning 0.1 – Project Manager
Project Ablation Sonication Views Help
Place
Reset
Status bar

Ablation Settings
Type: Sonablate 500
☑ Run Thermal Simulation

Sonications' Sequence
Sonication 1
Sonication 2

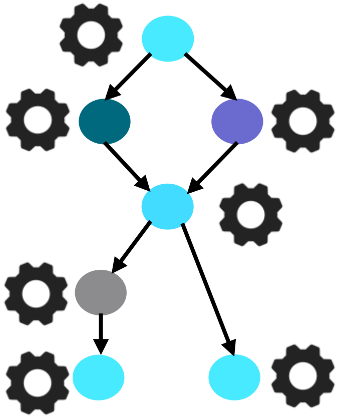
Transducer Settings
Pos: X Y Z
Orie: R Y P
EBS: a1 a2 fd

Driving Signal
Amp: Pa F: Hz
On: sec Off: sec

Visualisation Data
Dataset: CT
Beam: Geometric
☑ Show all sonications

Simulation Domain Offset
Left: Right:
W: l r

L: l W: w



.HDF
.HDF
.HDF
DSPTCHR
SERVER

k-Dispatch: Under the hood

1 RECEIVE AND PARSE THE PLANNING FILE

2 CREATE A WORKFLOW

3 SELECT THE EXECUTION PARAMETERS

4 STORE THE WORKFLOW STRUCTURE AND EXECUTION DETAILS TO THE DATABASE

5 GENERATE JOB SCRIPTS AND PROVIDE FILE TRANSFERS

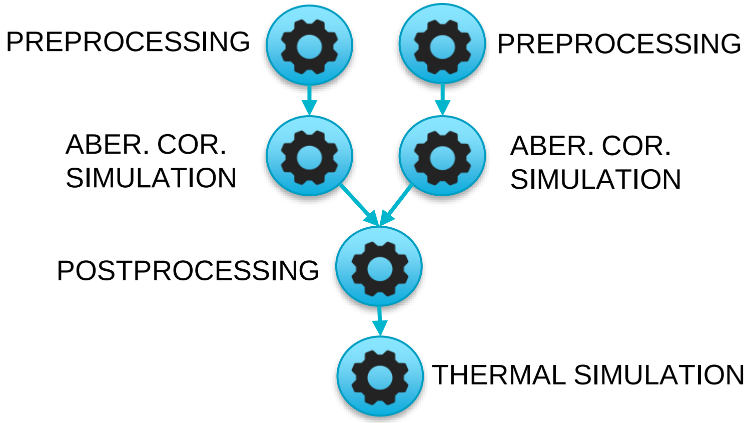
6 MONITOR THE REMOTE CALCULATION

7 PROVIDE ERROR DETECTION AND RECOVERY IF NEEDED

8 FILE TRANSFERS AND REMOTE CLEAN UP

9 ACCOUNTING AND REPORTING

.HDF



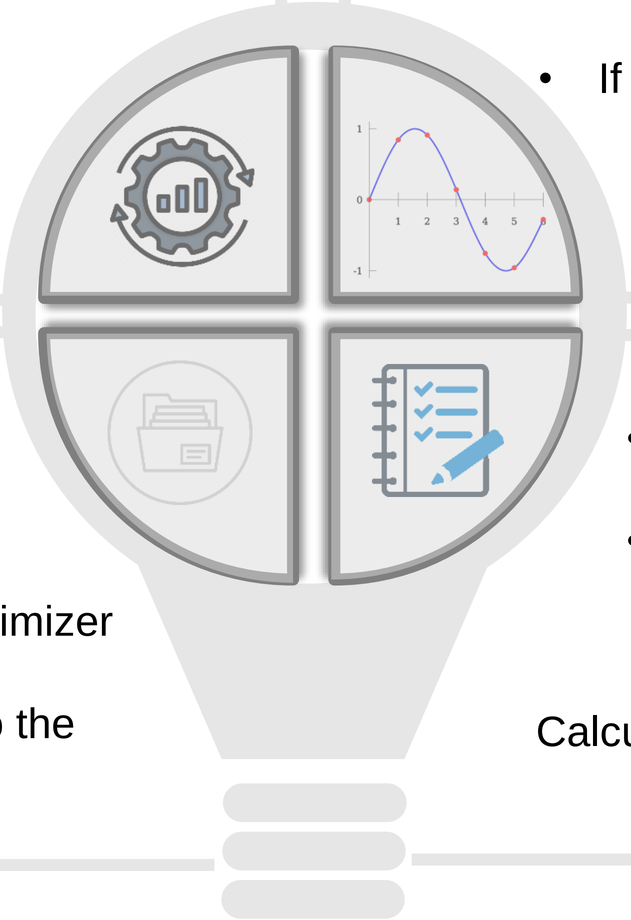
k-Dispatch’s Performance Modules

Optimizer

- Finds execution parameters based on task inputs
- Increases performance data diversity by small perturbations of the execution parameters
- Can be based on Genetic algorithms or Simulated annealing

Interpolator

- Estimates the execution time and cost for a given amount of resources
- Uses linear and cubic spline interpolation
- If fails, maximum execution time and associated cost are used



Collector

- Updates performance data in the database after each successful run
- Provides feedback to the optimizer after successful run
- Adapts optimizing process to the cluster workload variations

Evaluator

- Calculates the makespan (execution + queueing time)
- Runs the simulator (e.g. ALEA)

ALEA

Calculates the final quality criteria:

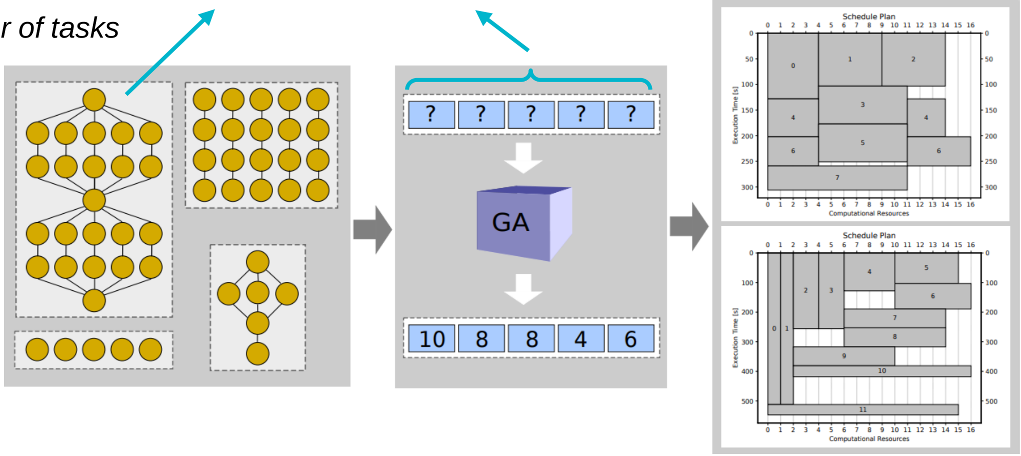
$$f = w * (t + q) + (1 - w) * c$$

Optimizer: Genetic Algorithm

Find the execution parameters (number of nodes, computational time) for individual tasks in the workflow so that given optimization constraints are met (minimal total execution time including queuing times and computational cost).

$$I = (R_1, R_2, \ldots, R_N)$$

N is the number of tasks in the workflow

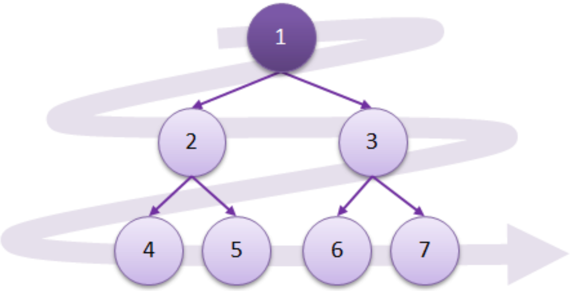


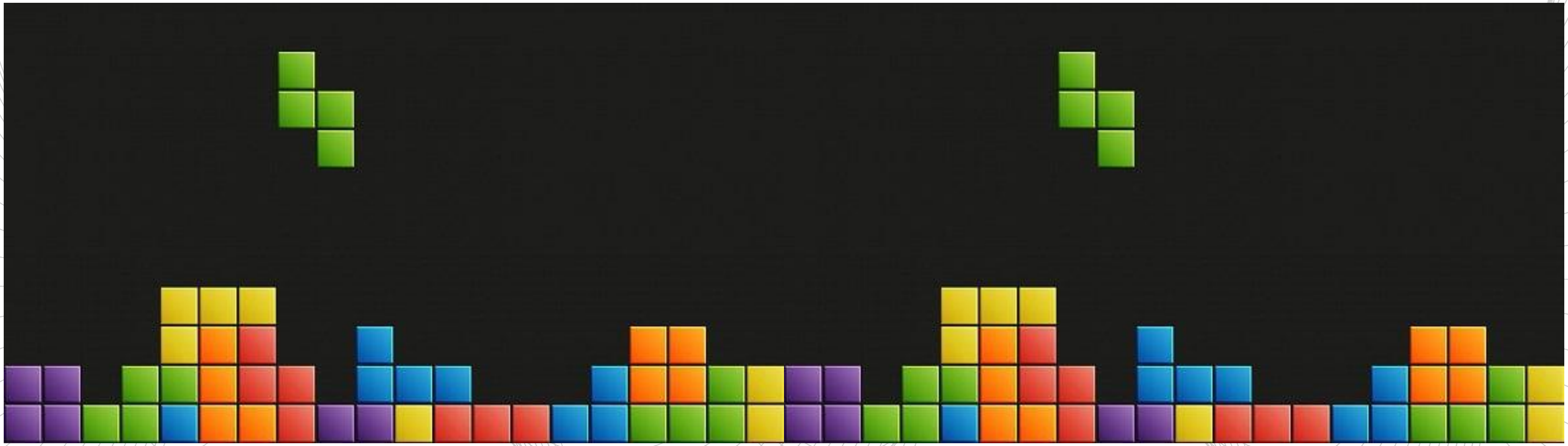
Three objective functions:

- **Local task optimization**
  - Total execution time is only given by the sum of the execution times of all tasks.
- **Global optimization with on-demand resource allocation**
  - The total execution time is given by the sum of the execution times of the tasks along the critical path in the workflow graph *(makespan)*.
  - Computational cost is given by the sum of the costs of all tasks.
- **Global optimization with static resource allocation**
  - The total execution time is given by the sum of the execution times of the tasks along the critical path in the workflow graph *(makespan)*.
  - Computational cost is fixed (given by the allocation). The goal is to minimize the amount of unused resources.

Evaluator: Tetrisator

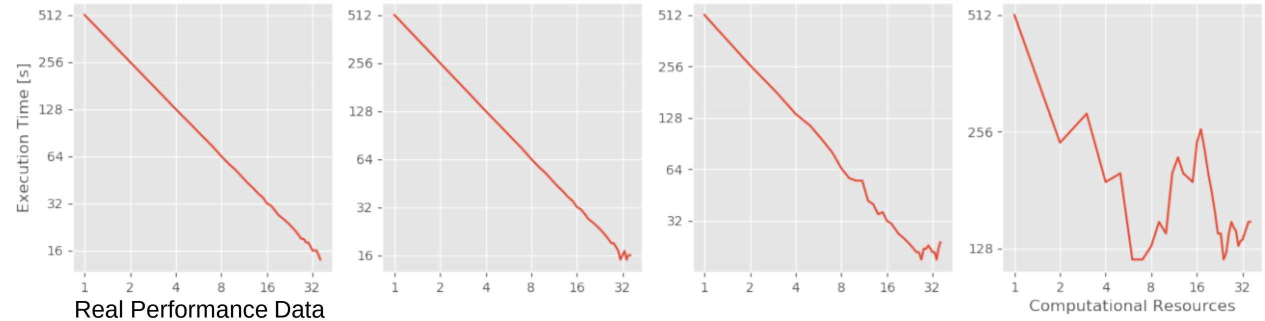
- Artificial HPC cluster simulator with a predefined number of nodes
- Tasks' queuing times are omitted
- Tasks are submitted to the simulator in the same order as defined in the solution encoding *(a breadth-first top-down traversal)*



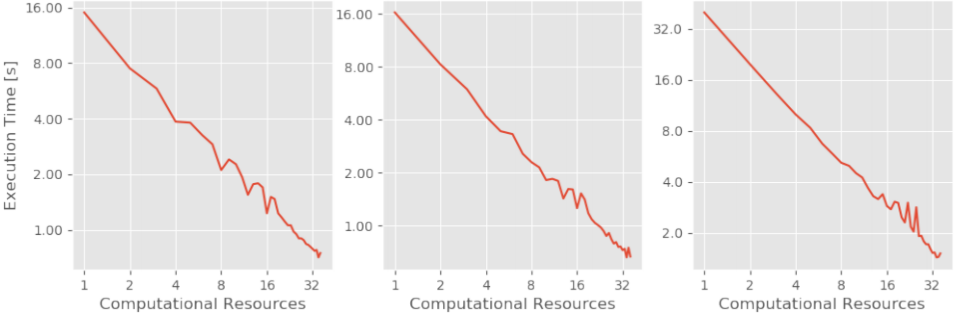


Experimental Data

Artificial Performance Data

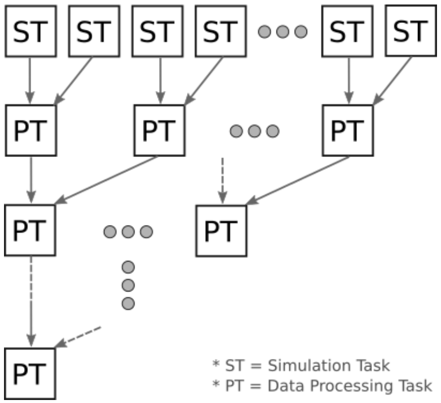


Real Performance Data

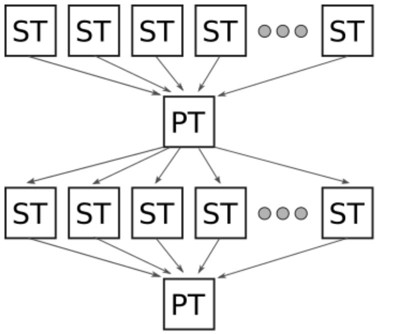


Performance data sets

- Artificial strong scaling data
- Real strong scaling data (Barbora@IT4I, 1-36 nodes, C++/MPI k-Wave toolbox, input data 500^3, 512^3, 544^3, GCC/Intel compiler)



* ST = Simulation Task
* PT = Data Processing Task

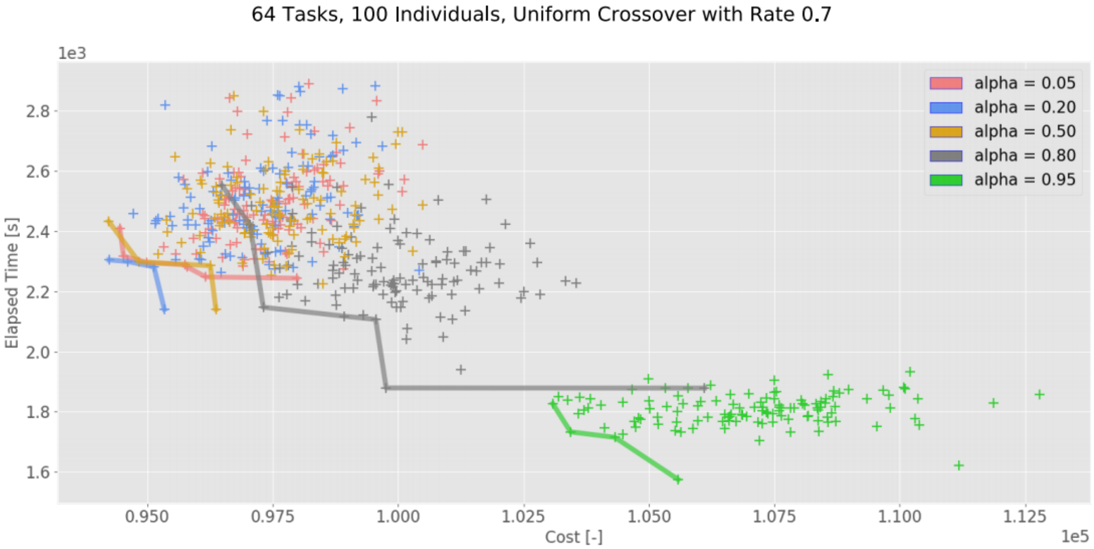


* ST = Simulation Task, PT = Data Processing Task

Investigated workflows

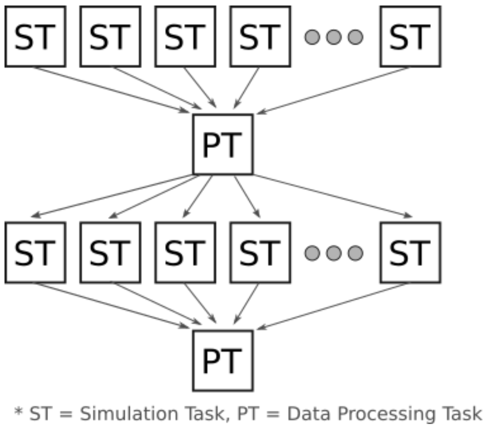
- Two task types are alternating (lightweight data processing tasks, heavy simulation tasks)
- Each workflow consists of 7-64 tasks

Results

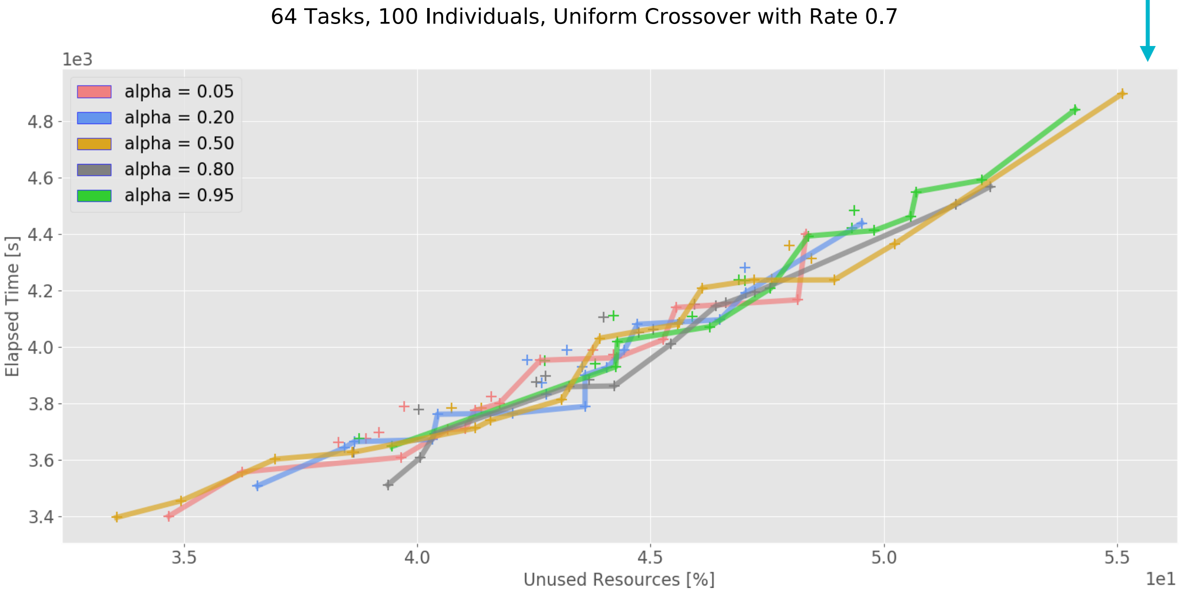


Global optimization with on-demand resource allocation

Global optimization with static resource allocation



* ST = Simulation Task, PT = Data Processing Task

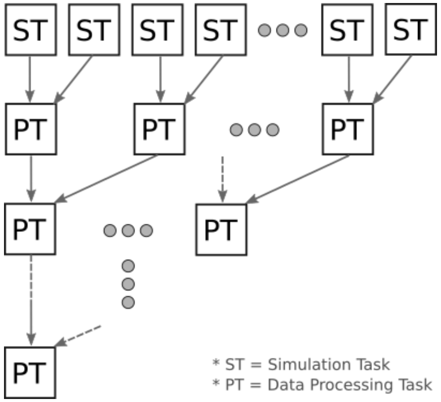


Evolved Schedules

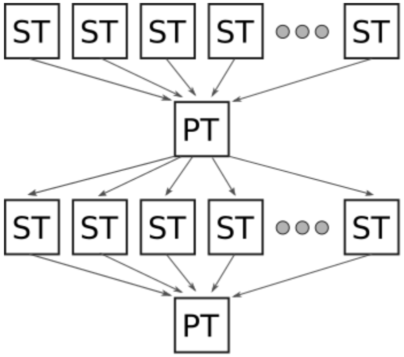
Local Optimization

Global Optimization
with On-Demand Allocation

Global Optimization
with Static Allocation



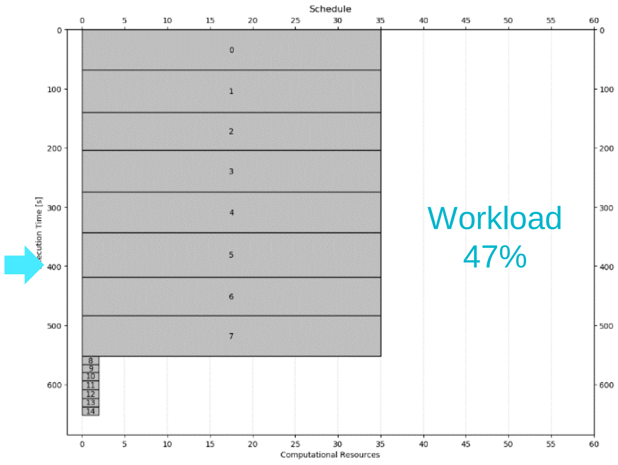
* ST = Simulation Task
* PT = Data Processing Task



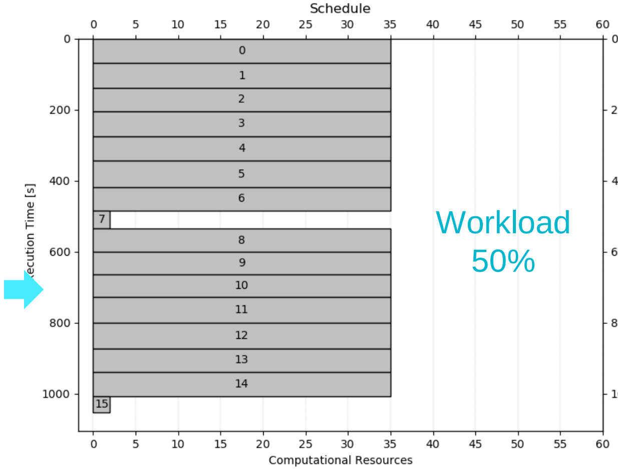
* ST = Simulation Task, PT = Data Processing Task

Evolved Schedules

Local Optimization

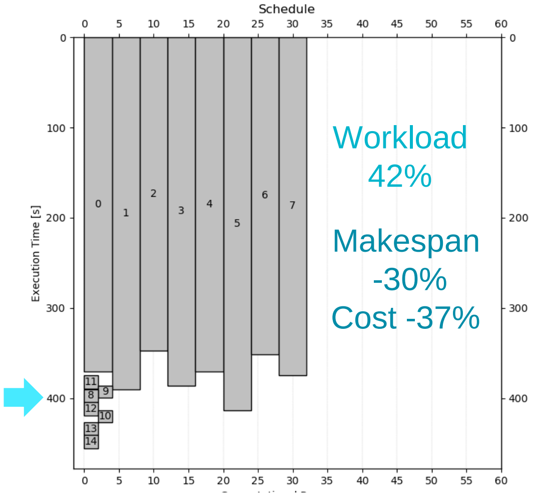


Workload 47%



Workload 50%

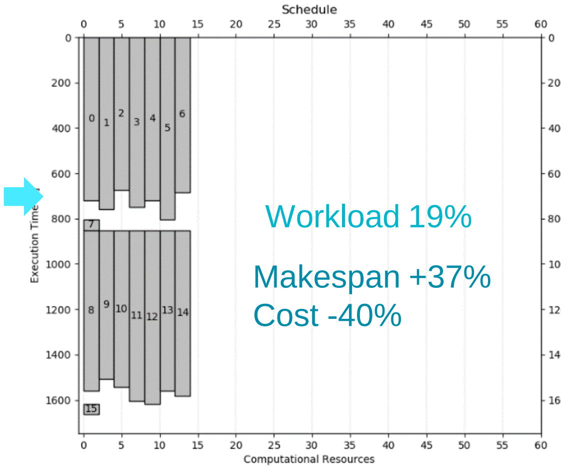
Global Optimization with On-Demand Allocation



Workload 42%

Makespan -30%

Cost -37%

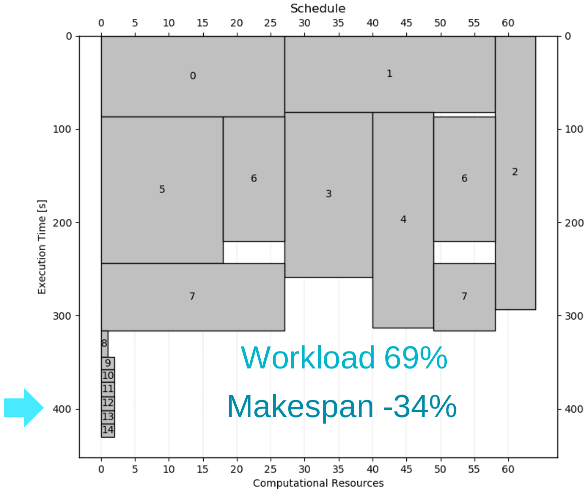


Workload 19%

Makespan +37%

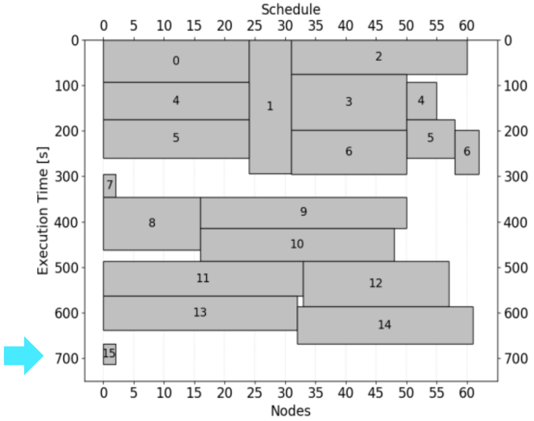
Cost -40%

Global Optimization with Static Allocation



Workload 69%

Makespan -34%



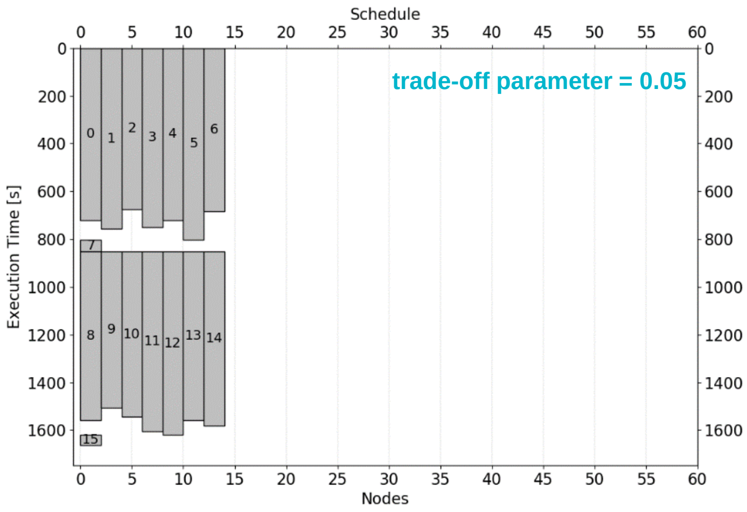
Workload 69% Makespan -32%

Ultimate Goal of k-Dispatch

Provide an **automated**, **effective** and **failure-free** workflow execution.

Minimal Cost

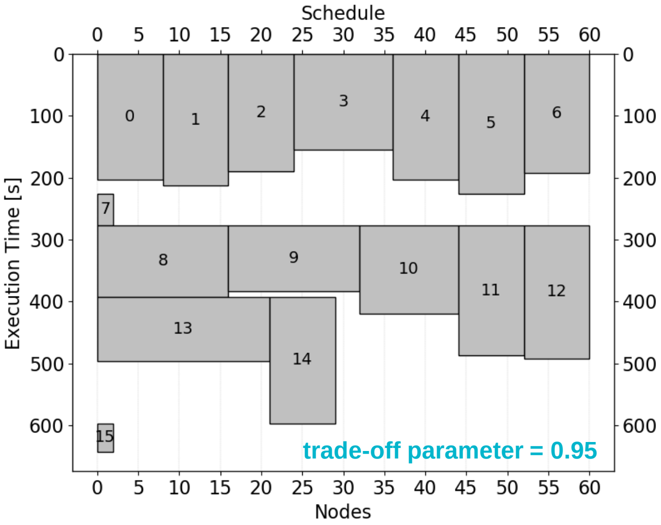
Minimal Execution Time



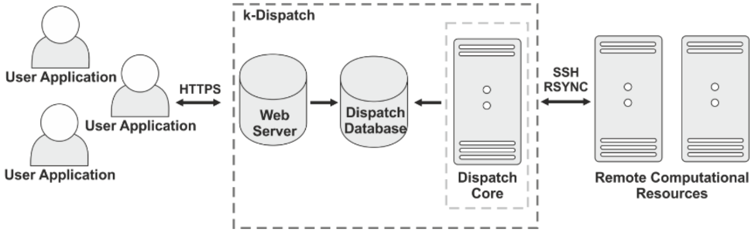
Makespan x2.5

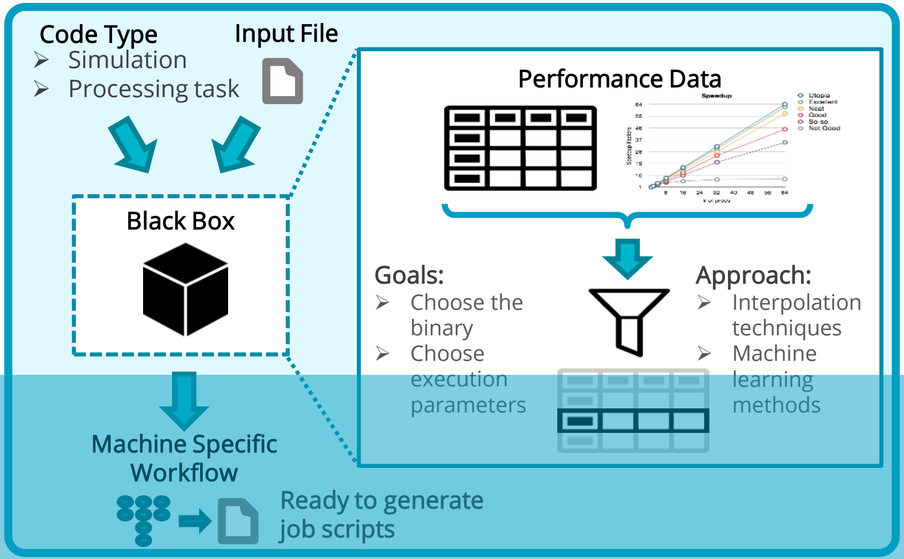
⟺

Cost x1.2



Conclusions





Thank you for the attention!