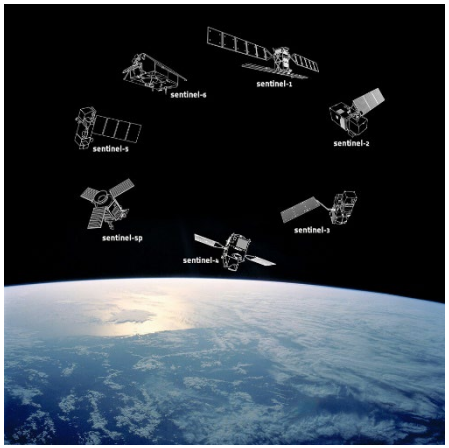
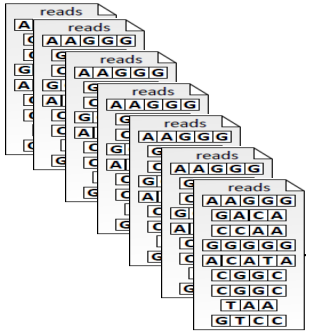
Scheduling of Scientific Workflows

Ulf Leser, Fabian Lehmann

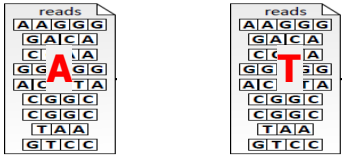
Big Scientific Data

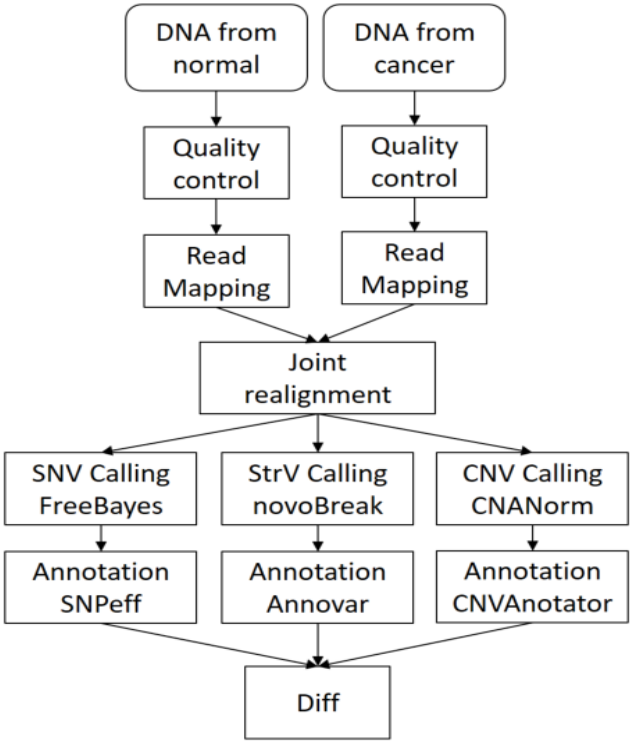


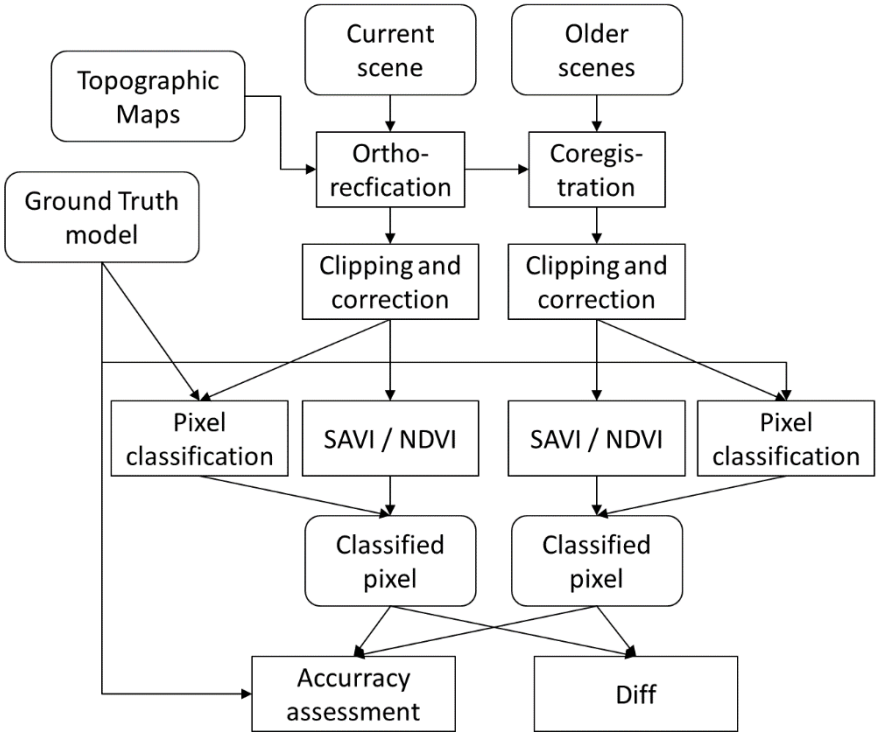


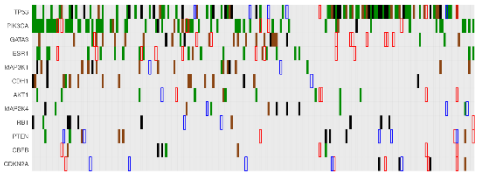
Note petabytes every day, but easily a few terabytes per week

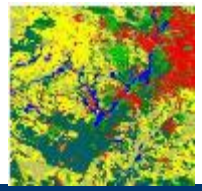
Data Analysis Workflows (DAWs)



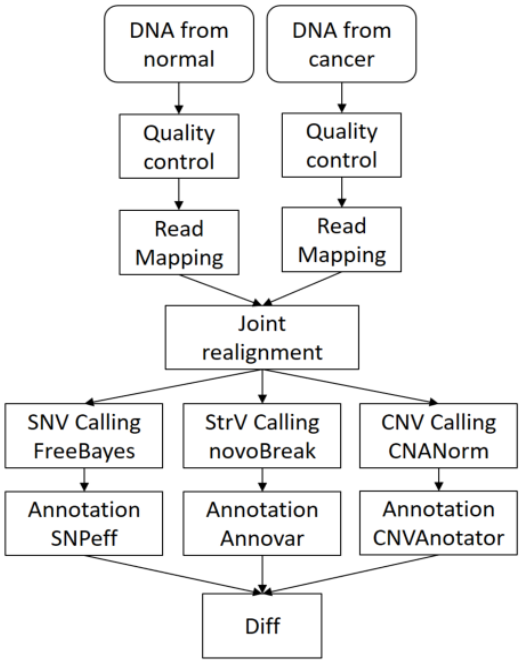








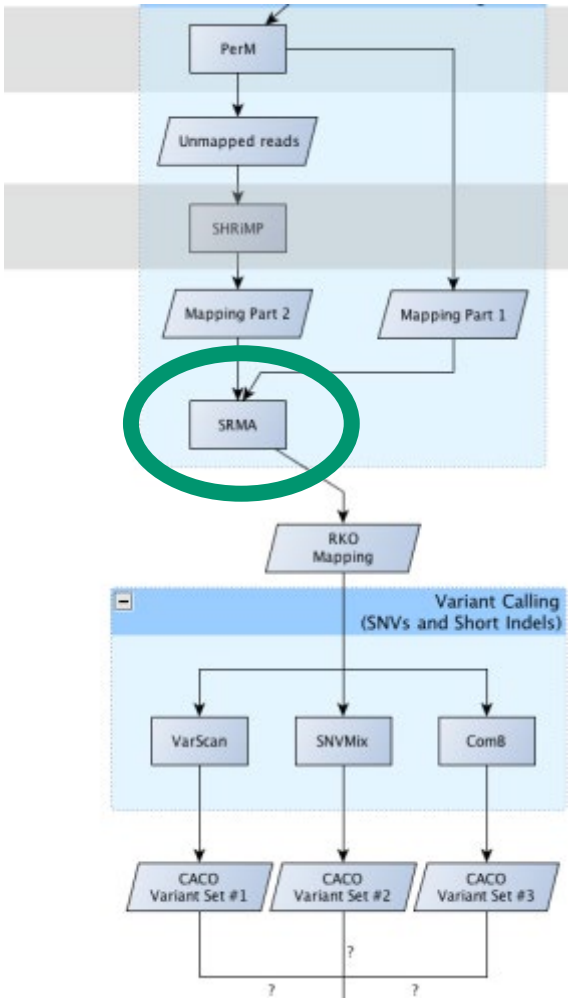
Distributed DAW Infrastructure



Workflows

Tasks

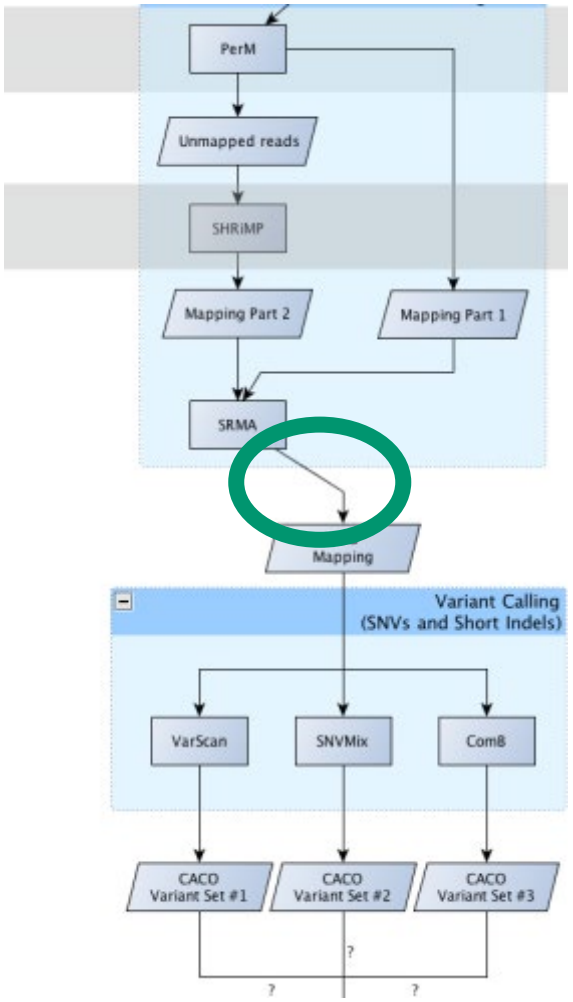
- Have (multiple) input and output "ports" (parameter)
- Must be executable
  - Or web services – deprecated in the Big Data area
- Black box model: Infrastructure has no notion on what a task does



Workflows: Tasks and Dependencies

Tasks

- Have (multiple) input and output "ports" (parameter)
- Must be executable
  - Or web services – deprecated in the Big Data area
- Black box: Infrastructure has no notion on what a task does

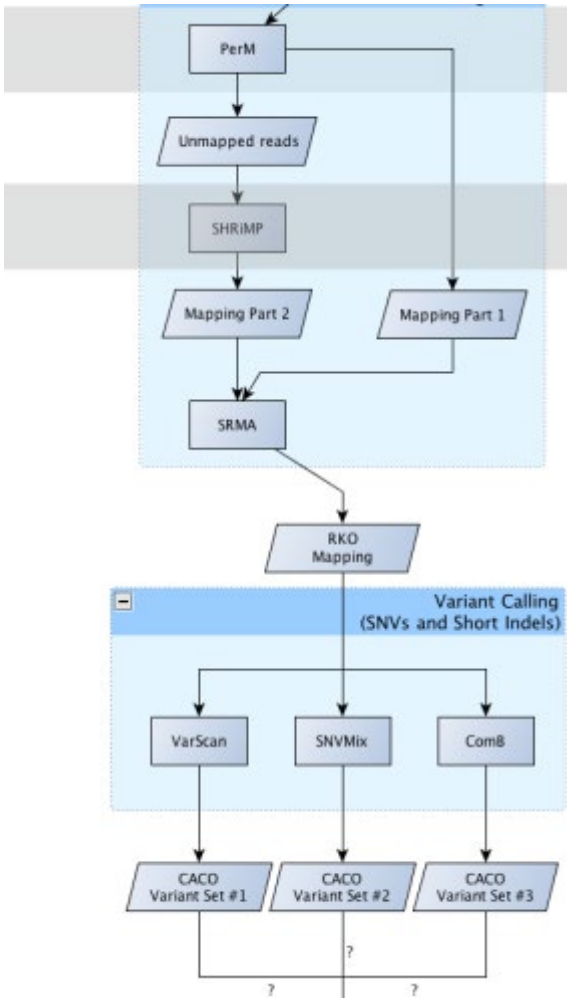


Dependencies

- Connect one input (upstream) with one output (downstream) port
- Implemented as files, pipes, in-memory, …
- Constrain the possible order of execution
- Black box model: Infrastructure has no notion on content (or format)

Dependency Graph

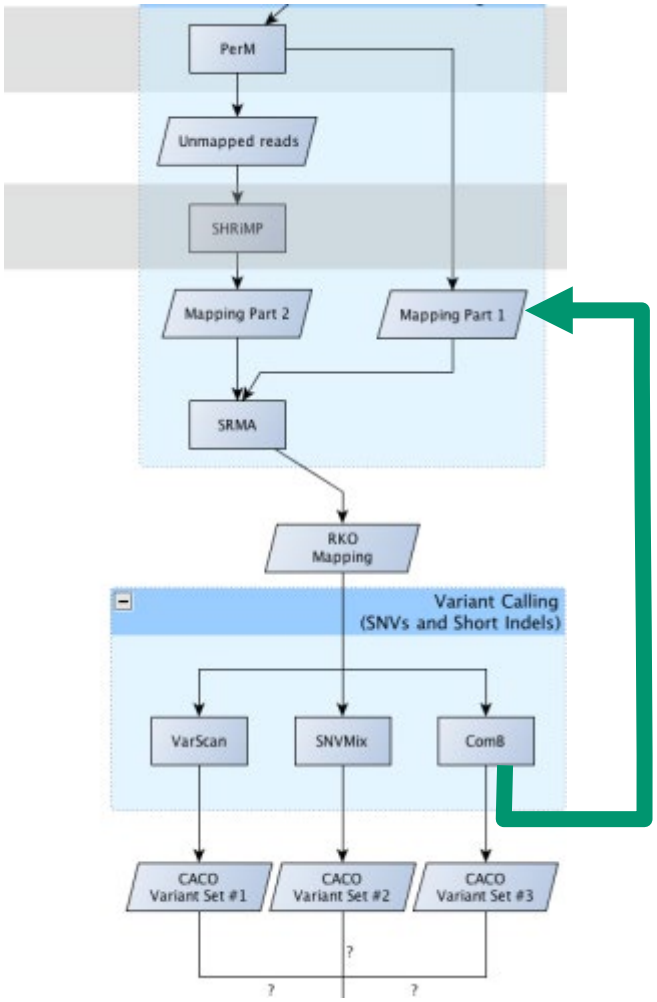
- Tasks become nodes (vertices)
- Dependencies become edges (arcs)
- Together a directed graph G = (T,D)



Dependency Graph

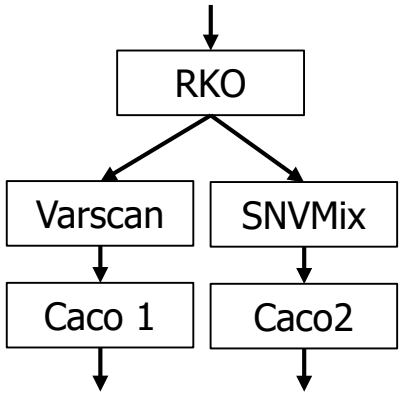
- Tasks become nodes (vertices)
- Dependencies become edges (arcs)
- Together a directed graph G = (T,D)

- Mostly a simple graph: No two arcs between the same pair of nodes
  - But: A task T1 may produce two files F1, F2 which both are input for task T2
- Mostly not a hyper-graph
  - But: Broadcasts can be modelled as n-ary arcs
- Mostly a DAG
  - But: Iteration / recursion introduces loops

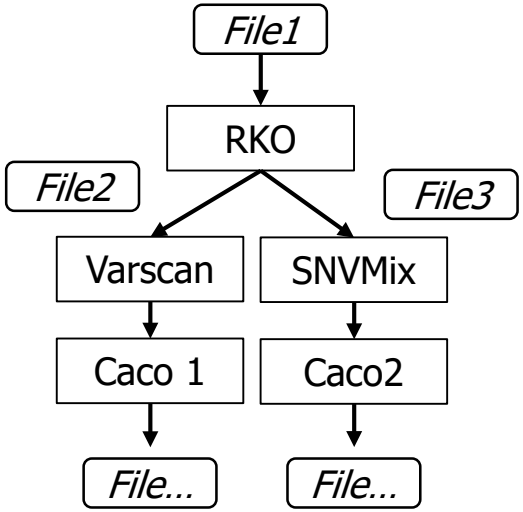


Abstract versus logical DAWs

- So far, our DAWs were abstract
  - Nodes have names, but there are no concrete files
  - Cannot be executed, but are easy to understand

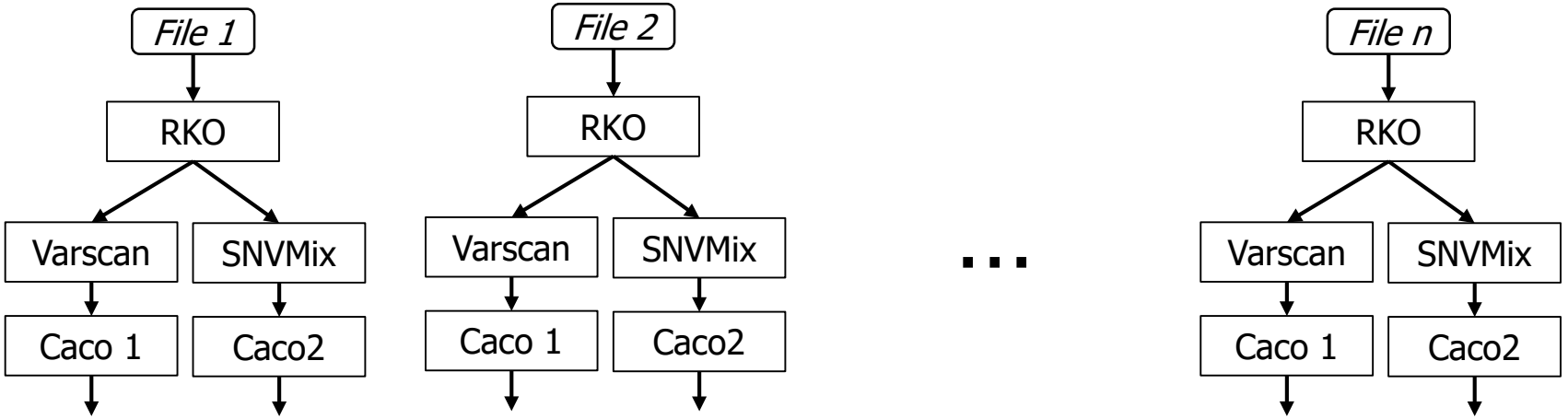


- A logical DAW also has concrete input files and produces concrete output files
  - A logical DAW instantiates an abstract DAW



Real Life

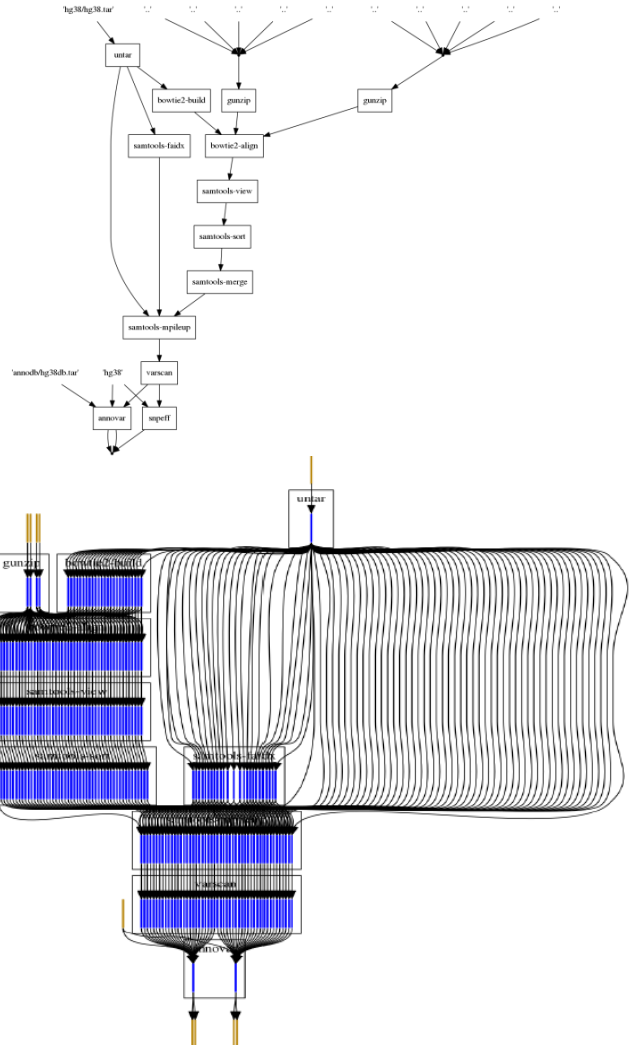
- Typically, DAWs are executed (from start or intermediate) over many files



- With multiple inputs, a single abstract (partial) DAW produces many logical (partial) DAWs
- More complicated as one may think, as the multiplicity of intermediate result files is only determined at runtime

Real Life with Real DAWs

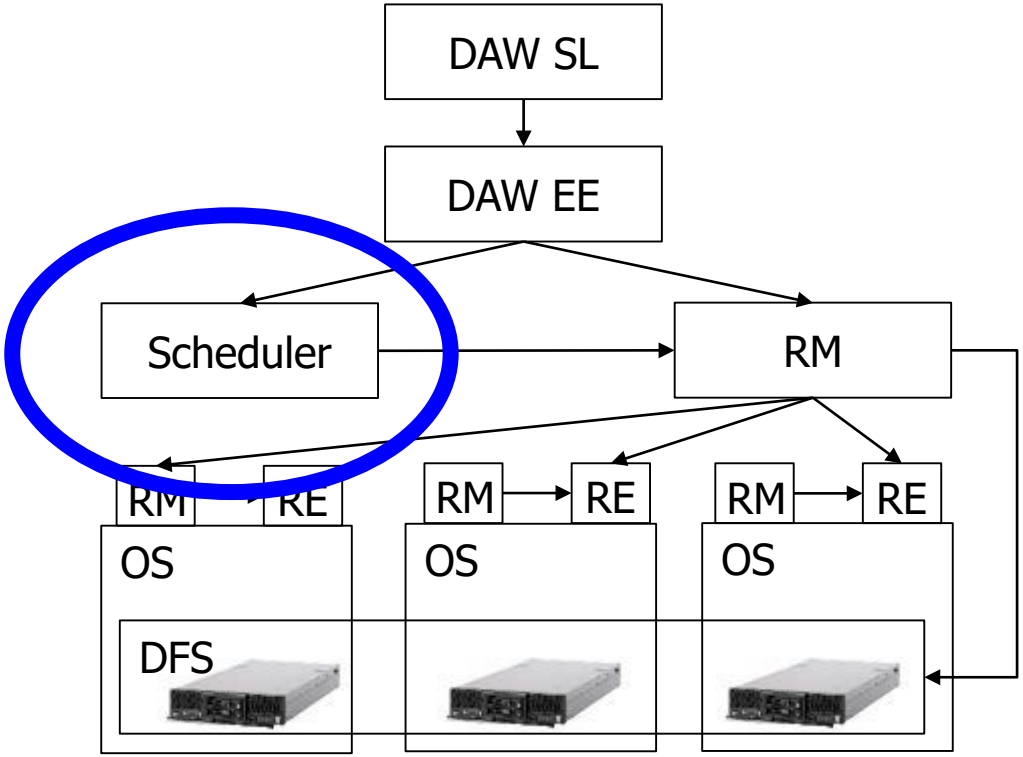
- Multiplicities
  - Some tasks read one file and produce one
    - Read image and produce normalized image
  - Some tasks read one file and produce many
    - Read large image and partition into smaller ones
  - Some tasks read many files and produce one
    - Read partitions and combine into one file



Physical DAWs

- A physical DAW is a logical DAW plus
  - Every (logical) task is assigned to a node for execution
  - Every dependency is assigned to a method of communication
- Logical DAW + schedule + data exchange = physical DAW

DAW Components



Scheduling of Workflows

- Management of queue of ready-to-run tasks
- Controlling data dependencies
- Mapping of tasks in queue to available resources
- Negotiation with resource manager
- Dealing with uncertainty
  - Task duration, memory requirements, …
- Provisioning of data at the right time at the right place
  - And cleaning up
- Exact or heuristic
- Achieving some optimization goal
  - Runtime, price, energy, …

Who should be here

- Master Informatik
- Ability to read English papers
- Knowledge in
  - Distributed systems (e.g. scheduling, file systems)
  - Algorithms (e.g. optimization, heuristics, search spaces)
- Willingness to work independently
  - Search suitable papers covering a topic, prepare presentations, write seminar thesis

How it will work

- Today: Presentation and choice of topics
  - If desired, we will group teams of 2 students
- 26.05.23: Send an outline of your topic (next slide)
- 02.06.23: Present your topic in 5min talk
- ~10.7.23: Meet your advisor to discuss slides
- ~15.7.23: Present your topic (30min) in a Blockseminar
- 30.08.23: Write seminar thesis (10-15 pages)

The outline

- Topics will be rather abstract
- Find yourself a set of suitable papers
  - A specific focus is allowed and welcome
- Extract the most important information
- Structure into an outline of your thesis
  - Abstract, chapters, sections,
  - 1-2 sentences per section to describe the content
- Abstract
  - Roughly 20 lines – what is the topic, what will the thesis describe?
- Send us outline + references
  - Mark your top-3 references – those that most likely will form the basis of your work

The 5-min flash talk

- Focus on marketing – sell your topic to gain audience
  - What is the topic?
  - Why is it challenging?
  - Why is it cool?
  - What are important applications?
  - What will your talk be about?
- At most 5 slides
- Focus on figures & examples; omit details or algorithms

Presentation

- 30min presentation
- German or English
- Explain topic, methods, maybe experimental results
- Compare different approaches (if enough time)
- Aim: Your audience should understand what you say
- No need to cover the topic entirely – a clear focus is helpful

Teams

- If a topic is addressed by a team of two students, I expect
  - Read more papers
  - Have more topics in your outline and thesis
  - Write longer thesis
  - Presentations times remain the same – choose wisely

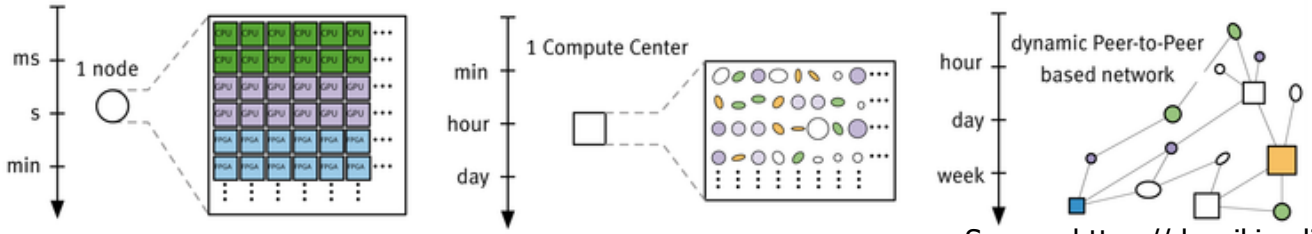
ToC

- Introduction
- Topics
- Assignment
- Hints on presenting your topic and writing your thesis

Topic	Advisor	Assigned to
Basic scheduling problem and variations	UL	
Task graph scheduling	FL	
Location-aware scheduling	FL	
Prediction of resource requirements	FL	
Prediction of task progression	UL	
Evaluation and simulation	FL	
Price-aware scheduling	UL	
Carbon-aware scheduling	UL	
Scheduling in real engines	UL	
Resource management with Slurm	FL	

Basic Scheduling Problem and its Variations

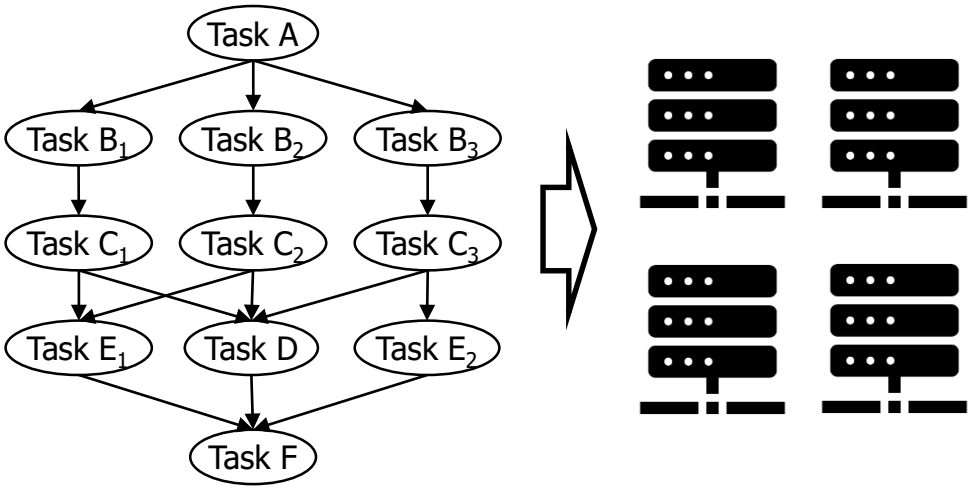
- Scheduling problems are everywhere
  - Zeitplanerstellung, Ablaufplanung, Reihenfolgeplanung, …
  - Logistics, production planning, traffic control, room assignment, …
- Optimal mapping "tasks" → "resources" under constraints
  - Task dependencies (acyclic, cyclic, linear)
  - "Fit" of resources for tasks
  - Optimization goals: Runtime, fairness, resource consumption, …
- Questions
  - What are basic scheduling models and there variations
  - Complexity classes and approaches to solution?
  - Classical heuristics for selected problems?



Source: https://de.wikipedia.org/wiki/Scheduling

Task graph scheduling (FL)

- Assign tasks to nodes
- Keep dependencies
- Minimize/maximize objective



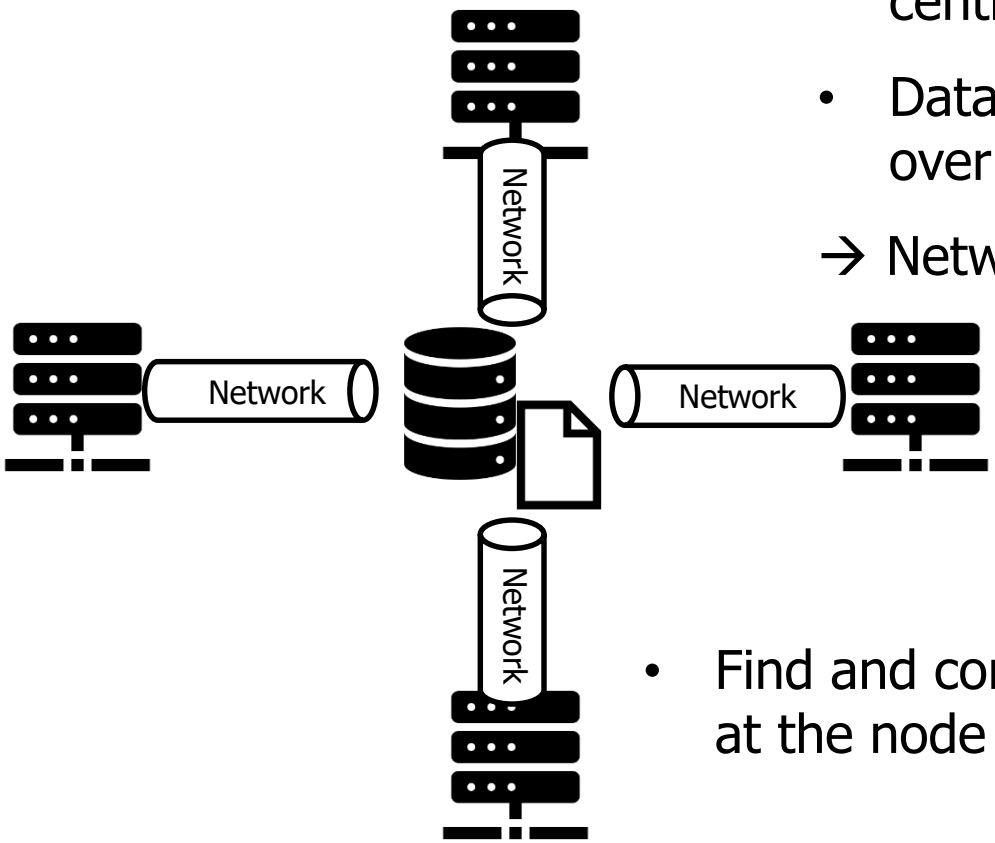
Task graph

Set of nodes

- Many algorithms exists
- All are heuristics
- Implement, simulate and compare a few

Where and when to start the task?

Location-aware scheduling (FL)

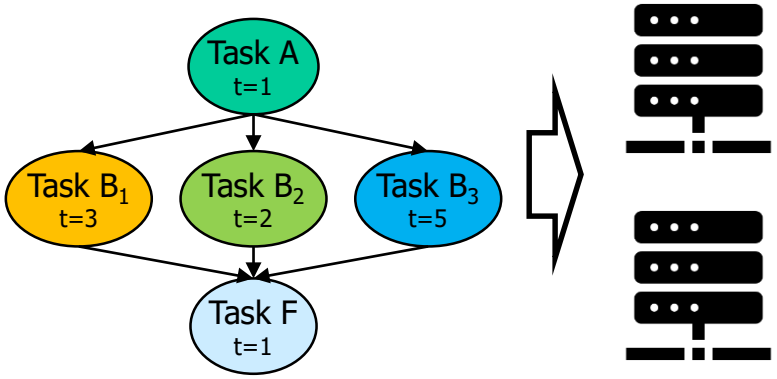


- Today's workflow systems use a central storage
- Data is always read and written over the network

→ Network becomes a bottleneck

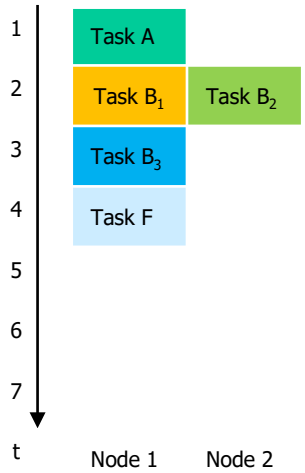
- Find and compare strategies keeping data at the node

Prediction of resource requirements (FL)

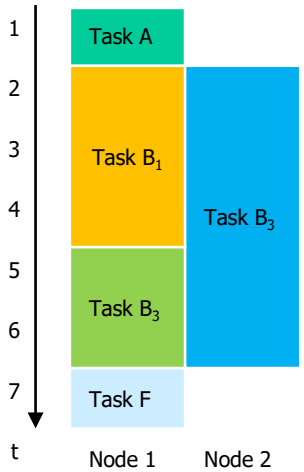


- Without knowledge of resource consumption scheduling decisions are inaccurate
- Resources are: time, memory, and CPU requests
- Tasks are black boxes → learn them

Without Runtime Knowledge



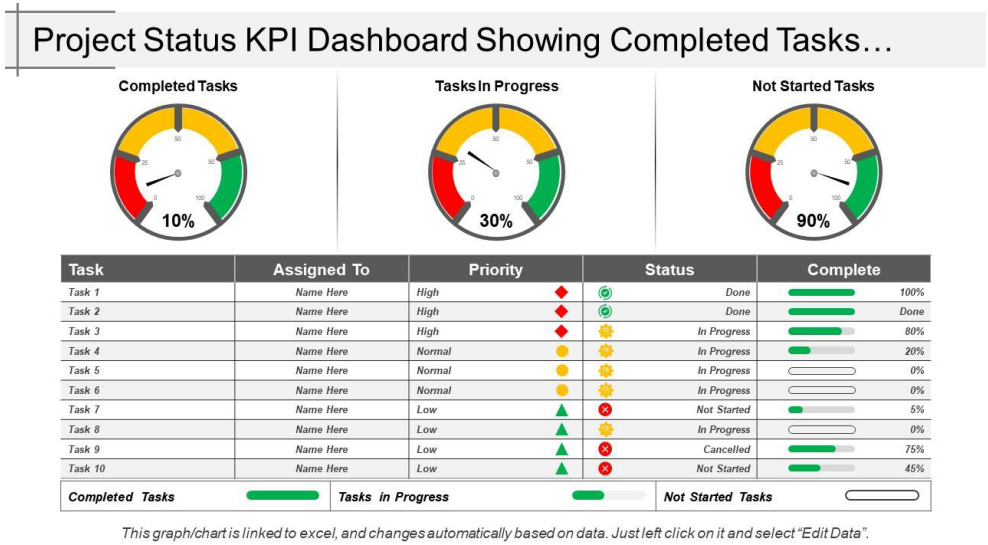
With



- Implement, simulate and compare a few strategies

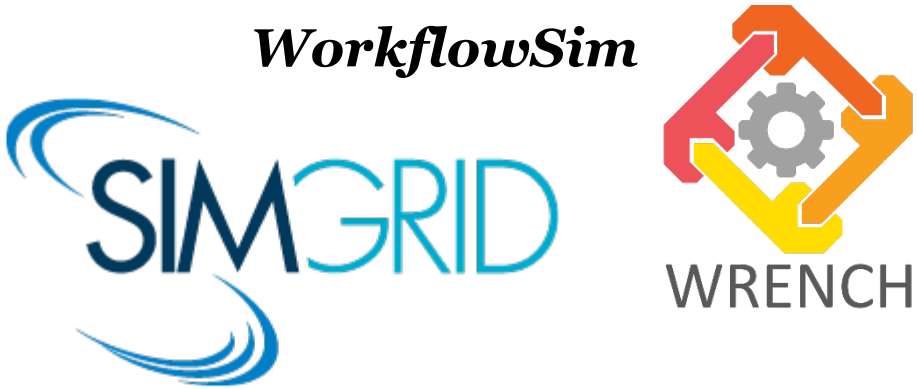
Prediction of Task Progression

- Task runtime is an essential information for any scheduler
- But a-priori estimates can go wrong
- Can we estimate task progression dynamically (time to finish?)
  - For dashboards – monitoring of progress
  - For adjusting schedules – updates runtime estimates
  - For coping with time-limited compute windows
- Questions
  - How to measure / predict task progression?
    - Much machine learning
  - Usage in scheduling
  - Other applications



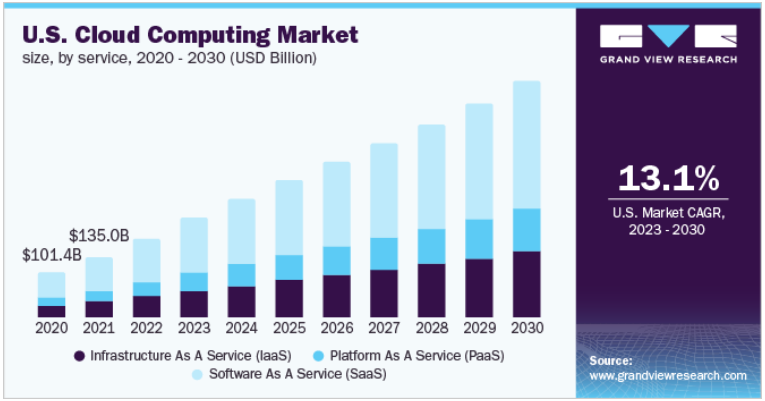
Evaluation and simulation (FL)

- Simulations are used to reduce time to results
- Compare different workflow simulation tools technically
- Test the same workflows in different tools
- Compare the results
- Limits of today's simulation frameworks



Price Aware Scheduling

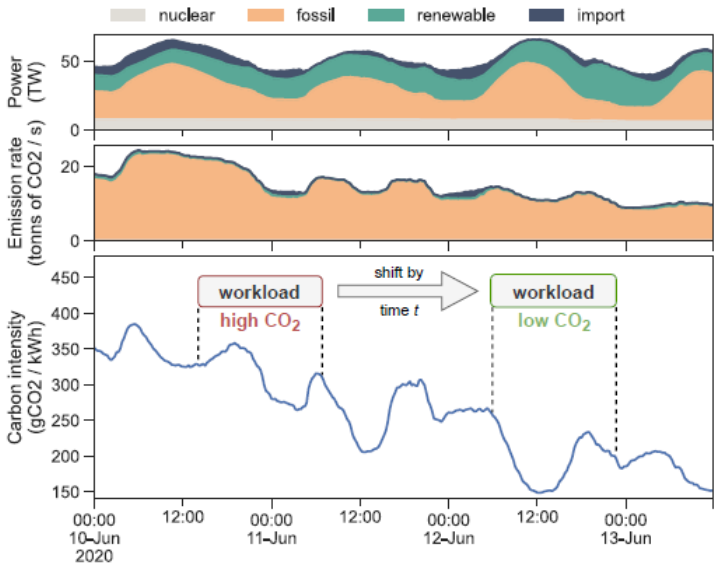
- Computation is never for free: Buy or rent?
- Very popular: Commercial clouds
  - AWS, Google, Telekom, …
- These have a pricing model: Pay by use
  - Available resources, time of day, quality of service, …
- Reverse optimization Optimal throughput at given resources or minimal cost for given compute power?
- Questions
  - Pricing models of cloud providers with pro/con?
  - Methods for optimizing price?
  - Cross-platform price optimization?



Source: https://www.grandviewresearch.com/industry-analysis/cloud-computing-industry

Carbon-Aware Scheduling

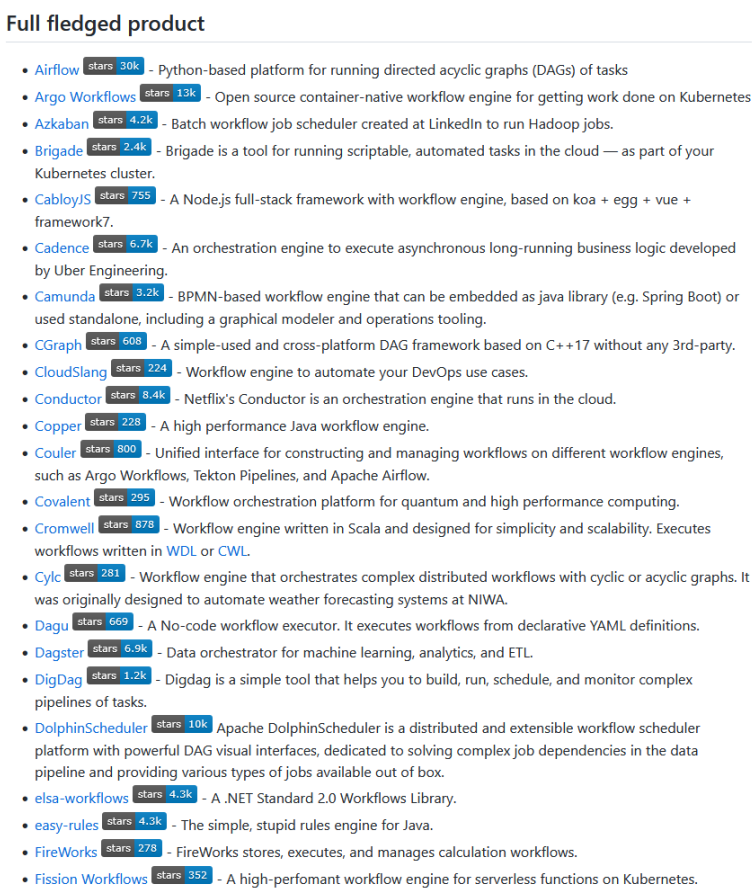
- Big data processing consumes millions of kilowatt
  - "Big data" – big storage, long computations, many nodes
- Energy consumption produces carbon
- How can we schedule workflows such that
  - Less energy is consumed
  - Less carbon is emitted?
- Questions
  - Difference between energy-aware and carbon-aware scheduling?
  - How to measure energy consumption of a workflow execution?
  - Concrete methods for carbon-aware scheduling?



Source: Wiesner et al. "Let's wait awhile", *Middleware Conference*. 2021

Scheduling in Real Systems

- Every workflow execution must be scheduled
- Every workflow engine must have / use a scheduler
- But how?
- Questions
  - How do engines determine schedules?
  - Static / dynamic / location-aware / round-robin … scheduling?
  - Extensible scheduling?
  - Which system uses which scheduling algorithm?



Full fledged product

- Airflow stars 30k - Python-based platform for running directed acyclic graphs (DAGs) of tasks
- Argo Workflows stars 13k - Open source container-native workflow engine for getting work done on Kubernetes
- Azkaban stars 4.2k - Batch workflow job scheduler created at LinkedIn to run Hadoop jobs.
- Brigade stars 2.4k - Brigade is a tool for running scriptable, automated tasks in the cloud — as part of your Kubernetes cluster.
- CabloyJS stars 755 - A Node.js full-stack framework with workflow engine, based on koa + egg + vue + framework7.
- Cadence stars 6.7k - An orchestration engine to execute asynchronous long-running business logic developed by Uber Engineering.
- Camunda stars 3.2k - BPMN-based workflow engine that can be embedded as java library (e.g. Spring Boot) or used standalone, including a graphical modeler and operations tooling.
- CGraph stars 608 - A simple-used and cross-platform DAG framework based on C++17 without any 3rd-party.
- CloudSlang stars 224 - Workflow engine to automate your DevOps use cases.
- Conductor stars 8.4k - Netflix's Conductor is an orchestration engine that runs in the cloud.
- Copper stars 228 - A high performance Java workflow engine.
- Couler stars 800 - Unified interface for constructing and managing workflows on different workflow engines, such as Argo Workflows, Tekton Pipelines, and Apache Airflow.
- Covalent stars 295 - Workflow orchestration platform for quantum and high performance computing.
- Cromwell stars 878 - Workflow engine written in Scala and designed for simplicity and scalability. Executes workflows written in WDL or CWL.
- Cylc stars 281 - Workflow engine that orchestrates complex distributed workflows with cyclic or acyclic graphs. It was originally designed to automate weather forecasting systems at NIWA.
- Dagu stars 669 - A No-code workflow executor. It executes workflows from declarative YAML definitions.
- Dagster stars 6.9k - Data orchestrator for machine learning, analytics, and ETL.
- DigDag stars 1.2k - Digdag is a simple tool that helps you to build, run, schedule, and monitor complex pipelines of tasks.
- DolphinScheduler stars 10k Apache DolphinScheduler is a distributed and extensible workflow scheduler platform with powerful DAG visual interfaces, dedicated to solving complex job dependencies in the data pipeline and providing various types of jobs available out of box.
- elsa-workflows stars 4.3k - A .NET Standard 2.0 Workflows Library.
- easy-rules stars 4.3k - The simple, stupid rules engine for Java.
- FireWorks stars 278 - FireWorks stores, executes, and manages calculation workflows.
- Fission Workflows stars 352 - A high-perfomant workflow engine for serverless functions on Kubernetes.

Source: https://github.com/meirwah/awesome-workflow-engines

Resource management with Slurm (FL)

- Resource manager executes tasks
- Slurm is commonly used
- Scales for thousands of nodes

- Describe the features of Slurm
- Slurm’s workflow awareness
- Slurm’s extendibility
- Test the Slurm simulation

Topic	Advisor	Assigned to
Basic scheduling problem and variations	UL	Stankov
Task graph scheduling	FL	Busch, Kaufmann
Location-aware scheduling	FL	Salek, Trogant
Prediction of resource requirements	FL	Riese, Patzak
Prediction of task progression	UL	Reinicke, Grund
Evaluation and simulation	FL	Cheng, Feng
Price-aware scheduling	UL	Cantepe, Gyuler
Carbon-aware scheduling	UL	
Scheduling in real engines	UL	Haase
Resource management with Slurm	FL	Kummer