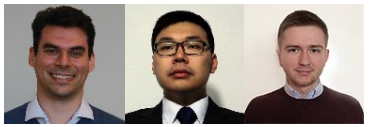
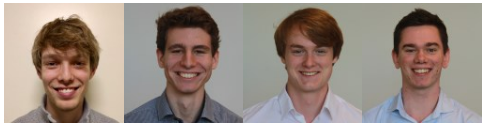
Modeling and Scheduling Complex Workflows with Dynamic, Non-Functional Requirements in Datacenters

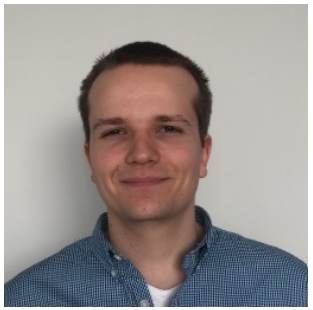
Slides developed jointly with Alexandru Iosup



VU Amsterdam:
Alexandru Iosup, Wing Ngai, and Mihai Neacsu



TU Delft:
Erwin van Eyk, Georgios Andreadis, Matthijs Bijman, and Leon Overweel



Ir. Laurens Versluis
Massivizing Computer Systems
@Large research,
https://atlarge-research.com

VU

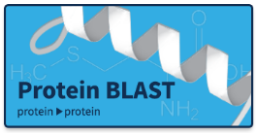
NWO

spec

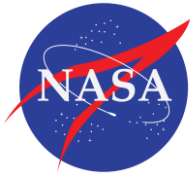
TU Delft
Delft University of Technology

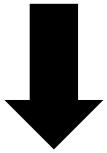
Today's society is increasingly using workflows, and increasingly complex

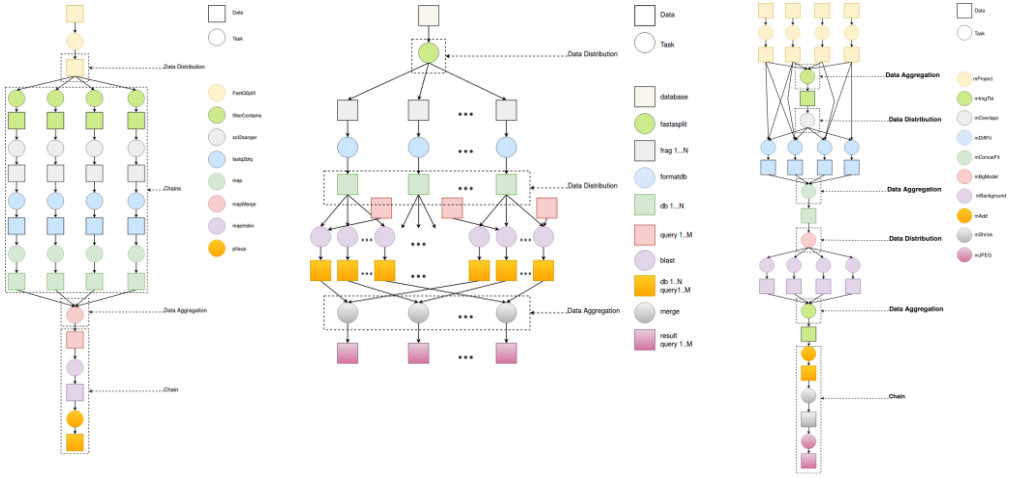
- Thousands of scientific applications
- Represented as workflows



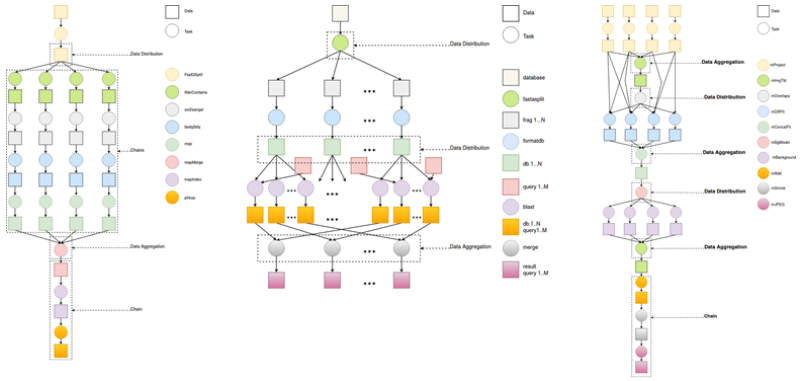
finding cancer early
epigenomics



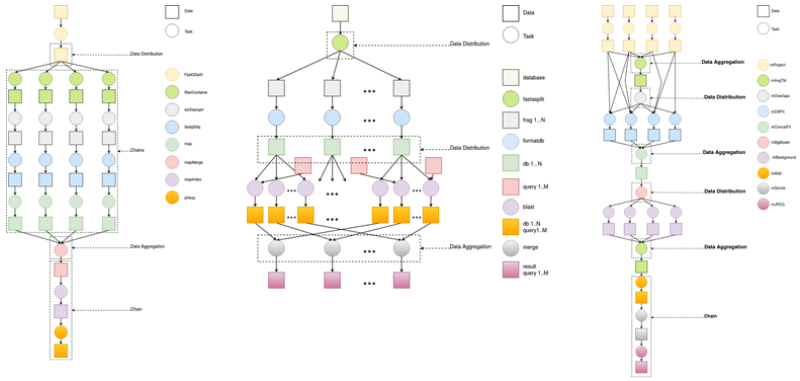




Workflows are executed in datacenters



Workflows are executed in datacenters



Scheduler

Workflows are executed in datacenters

Hundreds of schedulers exist => challenging problem

Scheduler

Scheduling is hard, also in real life
Example: planning a vacation

- Dependencies: money, date, (packing) belongings
- Agreements: What to visit/do and when?
- Constraints: food, accessibility, phobias...
- Coordination: meeting at the same place

Scheduling is hard, also in real life
Example: planning a vacation

- Dependencies: money, date, (packing) belongings
- Agreements: What to visit/do and when?
- Constraints: food, accessibility, phobias...
- Coordination: meeting at the same place

Scheduling is hard, also in real life
Example: planning a vacation

Complex workflows with dynamic, non-functional requirements.

My Ph.D. Roadmap

1. Design a model to support non-functional requirements at the task-based level for workflows
2. Incorporate the model in an (new) architecture
3. Create new schedulers to support dynamic, non-functional requirements

Why non-functional requirements at the task-based level?

- Currently, non-functional requirements are set on entire workflows.
  - Possibly wastes resources
  - Possibly increases costs
- Example: our vacation

Why non-functional requirements at the task-based level?

- Currently, non-functional requirements are set on entire workflows.
  - Possibly wastes resources
  - Possibly increases costs
- Example: our vacation

Why non-functional requirements at the task-based level?

- Currently, non-functional requirements are set on entire workflows.
  - Possibly wastes resources
  - Possibly increases costs
- Example: our vacation

Why non-functional requirements at the task-based level?

Q: Do we all need a seat at the emergency exit?

  - Possibly wastes resources
  - Possibly increases costs
- Example: our vacation

Why non-functional requirements at the task-based level?

Q: Do we all need a seat at the emergency exit?

A: no, but current approach gives no choice, and we pay for them.

Why non-functional requirements at the task-based level?

Q: Do we all need a seat at the emergency exit?

A: no, but current approach gives no choice, and we pay for them.

The seats are now marked as occupied as well, cannot be used where actually needed.

We need a formalism for dynamic, non-functional requirements

- Requirements
  - Allow non-functional requirements at a task-based level
  - Support dynamic changes
  - Generic: any non-functional requirement on all aspects of a workflow
  - Additional constructs such as groups
- Three possibilities:
  1. Such a formalism already exists
  2. We need to extend an existing formalism
  3. We need to design a formalism ourselves

VU

Investigate current formalisms

1. We created a library of complex workflows
   - Two with non-functional requirements
2. Comprehensive survey:
   - 11 conferences, last 5 years: 116 papers
3. Three formalisms:
   - BPMN
   - Petri Nets
   - Directed Acyclic Graphs (DAGs)

Main Finding: None of the three formalisms are capable of expressing non-functional requirements at a task-based level.

A new (extended) formalism: DAG+

- Metrics to qualitatively and quantitatively compare the formalisms
- Example metric: popularity
  - BPMN: 3 (3%)
  - Petri Nets: 5 (4%)
  - DAGs: 44 (38%)
- More information: please ask or read our CompSys 2017 article

Final outcome:

DAGs are most suitable to extend

My current focus

- Extending the formalism of DAGs to fulfill all requirements (DAG+)
- Annotated bibliography of scheduling approaches in datacenters and grids
- Tools to analyze trends and occurrence of keywords in (conference) papers
  - Feel free to ask about them!

VU

Future work: Resource management architecture and schedulers

- Investigate the support of resource management architecture
  1. No change needed
  2. Modifications required
  3. New architecture required
- Create new schedulers to support dynamic, non-functional requirements
- Compare the new schedulers against current state-of-the-art

Take-home message

1. Dynamic, per-task non-functional requirement are necessary
2. Existing popular formalisms for defining workflows do not support non-functional requirements
3. Proposed DAG+, a new DAG-based formalism for complex workflows with non-functional requirements
4. Proposed roadmap for my Ph.D.

Metrics used to compare formalisms

Properties	BPMN	Petri Nets	DAGs
Complexity	6	4	2
Utilization	$< 6\%$	100%	100%
Supports loops	Yes	Limited	No
SNFP	Limited	No	No
Domain	Business	Computer Science	Computer Science
Popularity	4%	3%	38%

Formalism occurrence per conference

Conference	Workflow	BPMN	Petri Net	DAG
HPDC	2	0	0	0
NSDI	1	0	0	0
SOCC	0	0	0	0
ICPE	1	1	1	0
Cluster	20	0	0	5
OSDI	1	0	0	0
SIGMETRICS	0	0	0	0
CCGRID	50	1	2	15
ICPP	12	0	0	7
IEEECLOUD	21	1	2	12
IPDPS	8	0	0	5
Total	116 (100%)	3 (3%)	5 (4%)	44 (38%)