

An Algebraic Approach for Data-Centric Scientific Workflows

Eduardo Ogasawara^{1,2}, Jonas Dias¹, Daniel de Oliveira¹
Fabio Porto³, Patrick Valduriez⁴, Marta Mattoso¹

¹ Federal University of Rio de Janeiro, Brazil

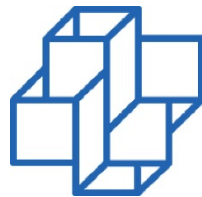
² CEFET/RJ

³ LNCC, Petrópolis, Brazil

⁴ INRIA & LIRMM, Montpellier, France



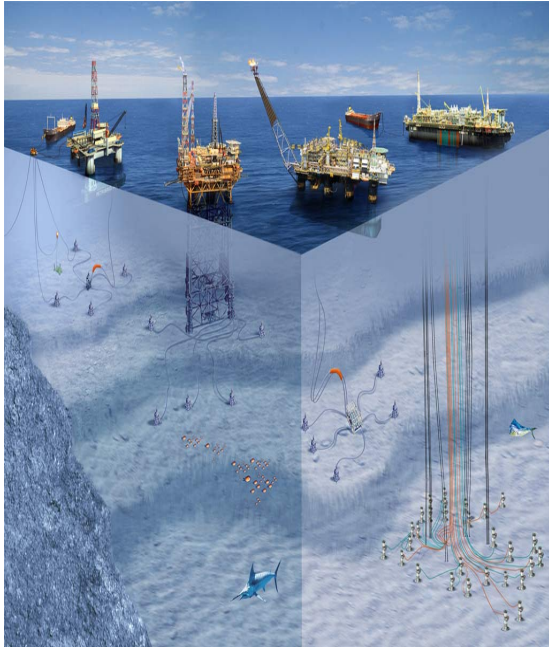
Federal
University
Rio de Janeiro



National
Laboratory
Scientific
Computing

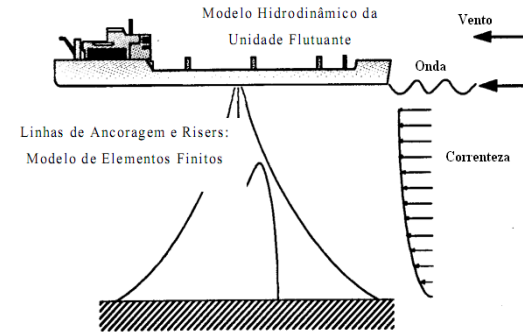


Risers' Fatigue Analysis in Ultra-Deep Waters

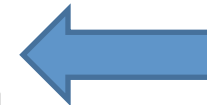
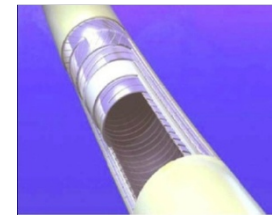
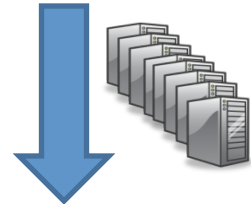


Input Data to simulate
Environment conditions:
Waves, wind, currents,
bathymetry, etc.

1. Coupled movement Analysis (TPN or Prosim)



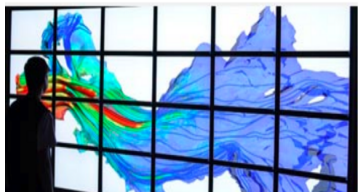
Generates large
amount of data ...
(finite element meshes)



2. ... to do Structural Analysis
of Risers (ANFLEX)



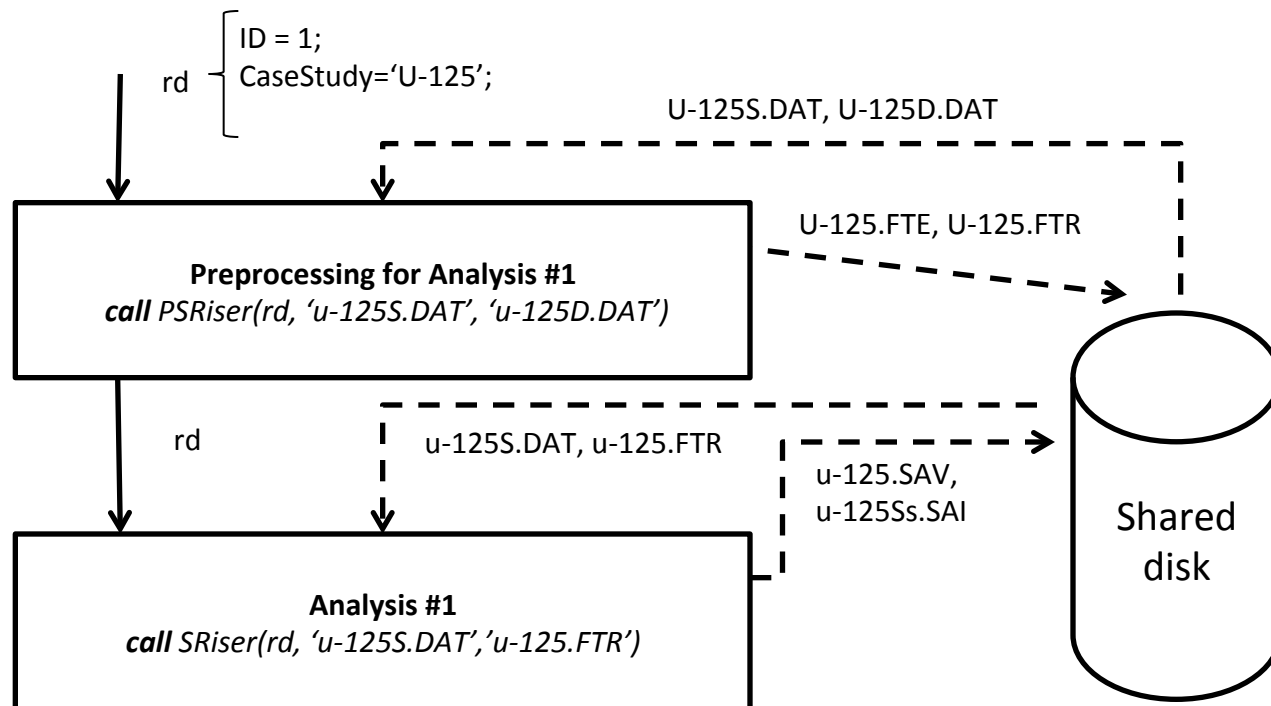
3. Results are analyzed
POSFAL



Estimate risers
lifetime

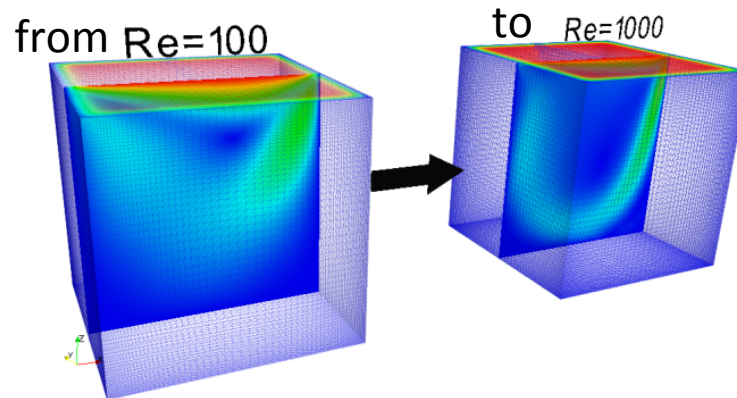


Example of Small Scientific Workflow



Parameter Sweep

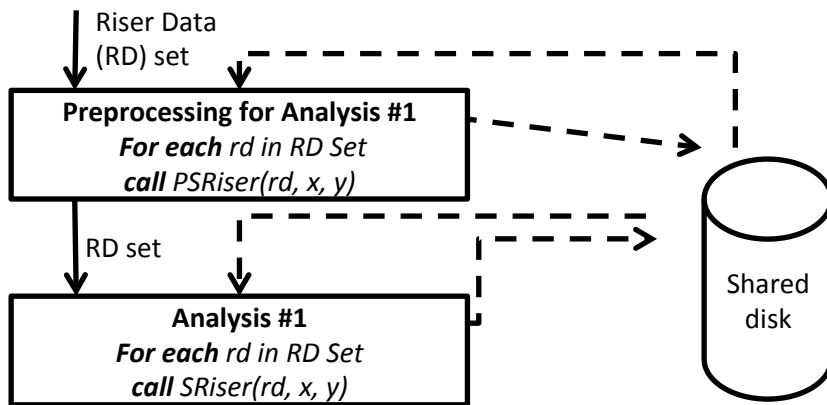
- Scientists have to explore the behavior of their model under different inputs.
 - This occurs in many areas such as computational fluid dynamics, bioinformatics, uncertainty quantification, dark energy analysis
- In parameter sweep we have multiple inputs for the workflow.



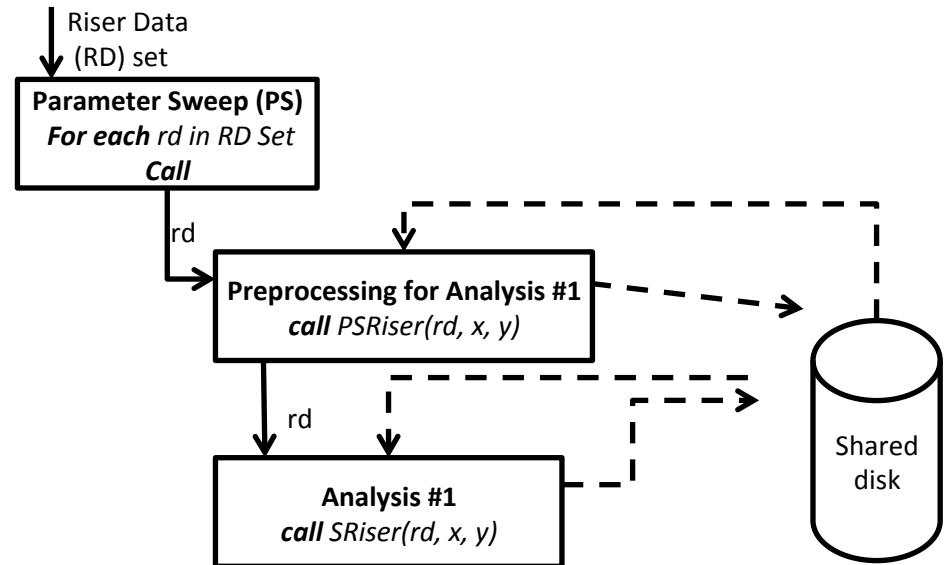
- These data-centric workflows becomes also computationally intensive and they may run for hours/days

Supporting Parameter Sweep

for each in all activities



for each invoking a group of activities



Parameter sweep are natural candidate for parallel processing

Current Approaches

- Parallel SWfMS
 - Swift: allows scientists to specify parallel workflows using a scripting language
- SWfMS Integration with Hadoop
 - Kepler+Hadoop: allows activities of a particular type to be parallelized
- SWfMS Integration with specialized middleware
 - VisTrails+Hydra: allows activities of a particular type to be parallelized

Current Approaches

- Parallel SWfMS
 - Swift: allows scientists to specify parallel workflows using a scripting language
- SWfMS Integration with Hadoop
 - Kepler+Hadoop: allows activities of a particular type to be parallelized
- SWfMS Integration with specialized middleware
 - VisTrails+Hydra: allows activities of a particular type to be parallelized

Fixed (rigid) execution plan

Problems

- Scientists must code low-level parallelization, thus limiting opportunities for automatic optimization
 - Huge amount of data consumed/produced
 - Ad-hoc and labor-intensive
- Engines are focused on scheduling activities of a fixed execution plan

Objectives

- Evaluate opportunities for optimization considering the entire workflow specification
- Transparent parallelization through strategies using automatic optimization
- View the workflow execution plan as a query optimization problem

Solution: An Algebraic Approach

- Data-Centric algebra for scientific workflows
 - Relations as data model for consumption and production
 - Operators that provide semantics to activities
 - Workflow execution model for this algebra based on activity activation

Relations as Data Model for Consumption and Production

- Relations are defined as sets of tuples of primitive types (integer, float, string, date etc) or complex data types (e.g. references to files)
- *Example: $R(\mathcal{R})$*

<u>RID</u>	<u>CaseStudy</u>	sdat	ddat
1	U-125	U-125S.DAT	U-125D.DAT
1	U-127	U-127S.DAT	U-127D.DAT
2	U-129	U-129S.DAT	U-129D.DAT

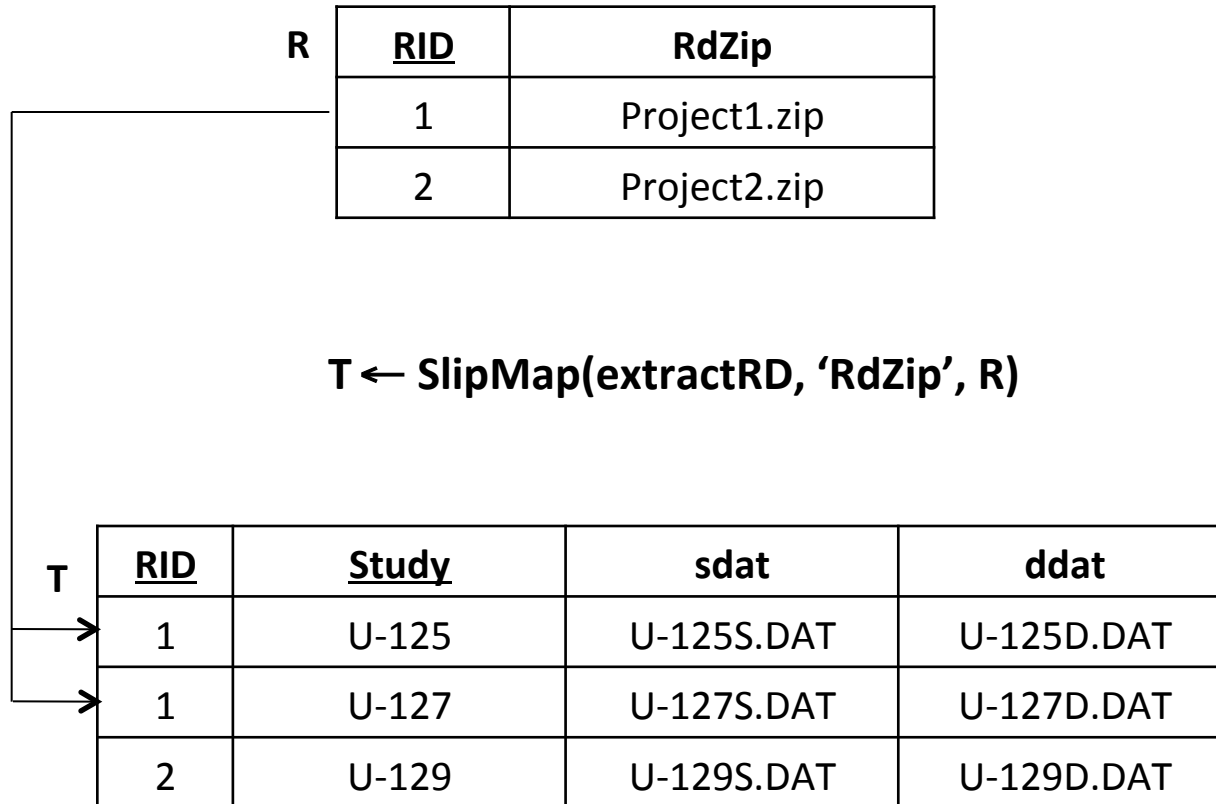
- $\mathcal{R} = (RID: Integer, CaseStudy: String; SDat: FileRef, DDat: FileRef)$

Algebraic Operators for Data-Centric Activities

- Program invocation
 - Map (1:1)
 - SplitMap (1:n)
 - Reduce (n:1)
 - Filter (1:0-1)
- Relational Algebra Expressions
 - SRQuery
 - MRQuery

Split Map Activity (SplitMap)

$$T \leftarrow \text{SplitMap}(Y, a, R)$$



Reduce Activity (Reduce)

$$T \leftarrow \text{Reduce}(Y, g_A, R)$$

R

<u>RID</u>	<u>Study</u>	<u>SsSai</u>	<u>DdSai</u>	<u>MEnv</u>
1	U-125	U-125Ss.SAI	U-125Dd.SAI	U-125.ENV
1	U-127	U-127Ss.SAI	U-127Dd.SAI	U-127.ENV
2	U-129	U-129Ss.SAI	U-129Dd.SAI	U-129.ENV

$$T \leftarrow \text{Reduce}(\text{CompressRD}, \{\text{'RID'}\}, R)$$

T

<u>RID</u>	<u>RdResultZip</u>
1	ProjectResult1.zip
2	ProjectResult2.zip

Single Relation Query Activity (SRQuery)

$$T \leftarrow SRQuery(qry, R)$$

R

<u>RID</u>	<u>Study</u>	SsSai	Curvature
1	U-125	U-125Ss.SAI	1.5
1	U-126	U-126Ss.SAI	0.9
1	U-127	U-127Ss.SAI	1.2

$$T \leftarrow SRQuery(\pi_{RID, Study, SsSai, Curvature}(\sigma_{Curvature > 1}(R)), R)$$

T

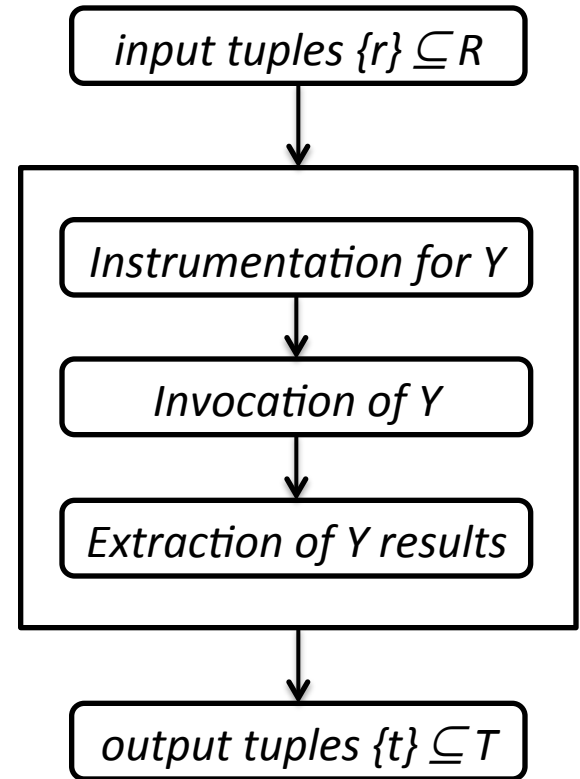
<u>RID</u>	<u>Study</u>	SsSai	Curvature
1	U-125	U-125Ss.SAI	1.5
1	U-127	U-127Ss.SAI	1.2

Workflow Execution Model

- Activity Activation
- Workflow Fragments
- Dataflow and Dispatching Strategies

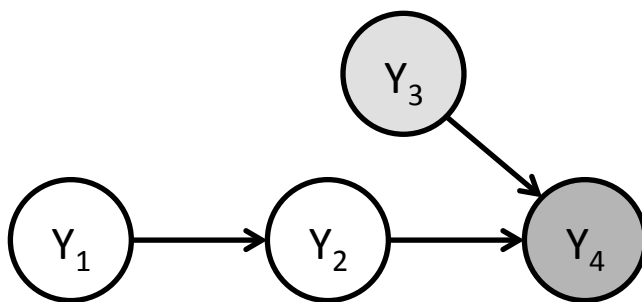
Activity Activation

- Activity activation is a self-contained object that holds all information needed (*i.e.* which program to invoke and which data to access) to execute an activity at any core
- Activations contain the finest unit of data needed by an activity to execute



Workflow Fragments

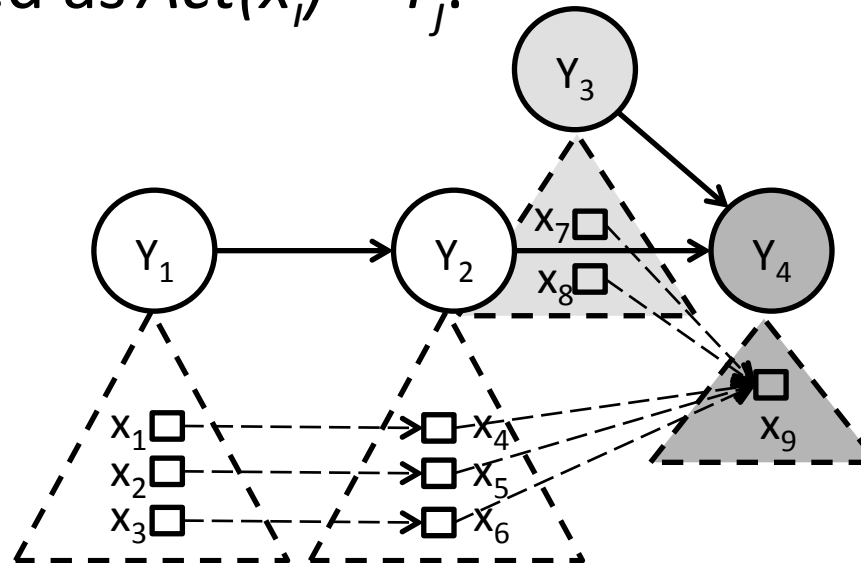
- A fragment F of a workflow is a subset F of the activities of a workflow W :
 - either F is an unitary set
 - or $\forall Y_j \in F, \exists Y_i \in F \mid (Dep(Y_i, Y_j)) \vee (Dep(Y_j, Y_i))$.



- Fragment F1: Y_1 and Y_2
- ◻ Fragment F2: Y_3
- ◼ Fragment F3: Y_4

Activations in Workflow Fragments

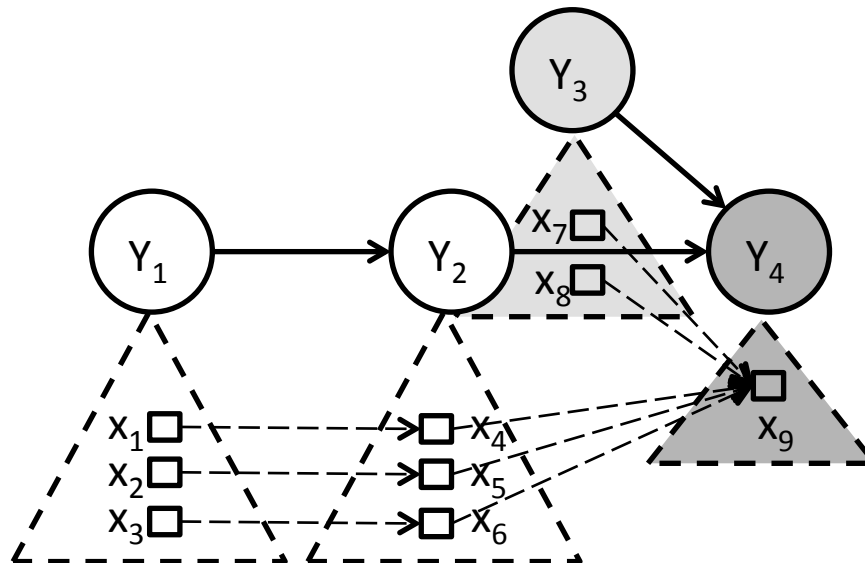
- Given a workflow W , a set $X = \{x_1, \dots, x_k\}$ of activations is created for its execution.
- Each activation x_i belongs to a particular activity Y_j , which is represented as $Act(x_i) = Y_j$.



The execution model obeys the Dataflow and Dispatching Strategies assigned to each fragment

Dataflow Strategies

- *First Tuple First (FTF)* partitions a set of activations in a fragment into a complete list of dependent activations;
- *First Activity First (FAF)* partitions a set of activations in a fragment into a complete list of independent activations ordered by activity dependence.



FTF:

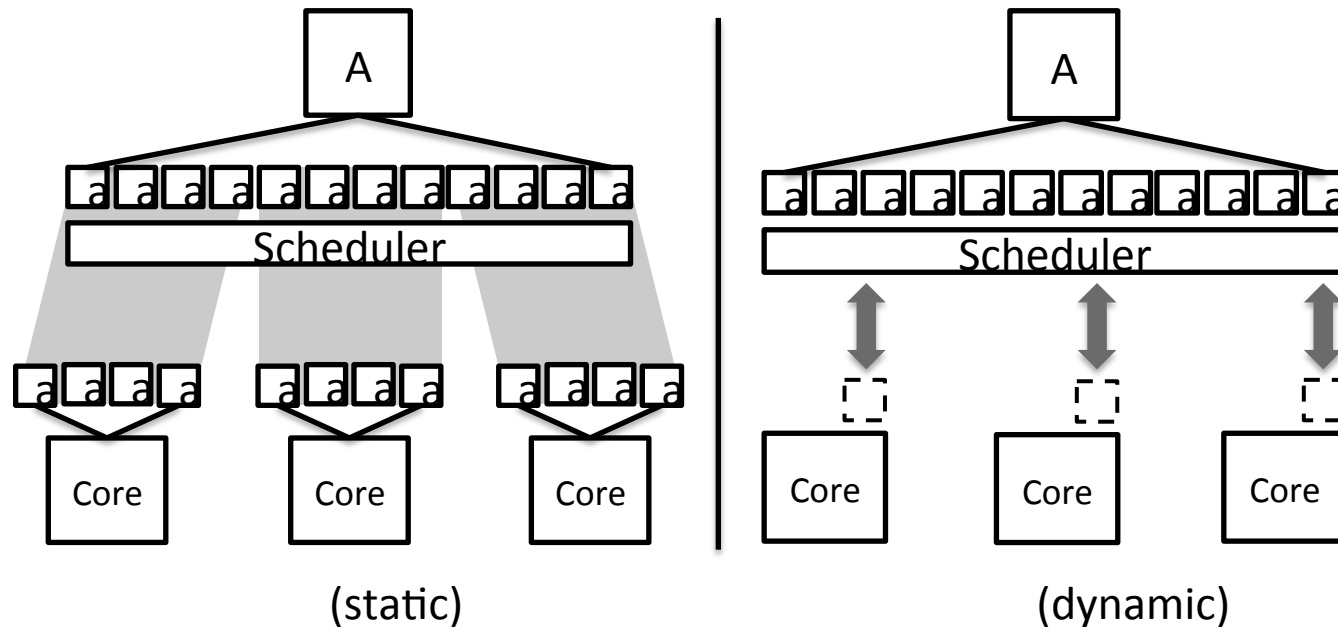
$\{ \langle X_1, X_4 \rangle, \langle X_2, X_5 \rangle, \langle X_3, X_6 \rangle \}$

FAF:

$\{ \langle X_1 \rangle, \langle X_2 \rangle, \langle X_3 \rangle, \langle X_4 \rangle, \langle X_5 \rangle, \langle X_6 \rangle \}$

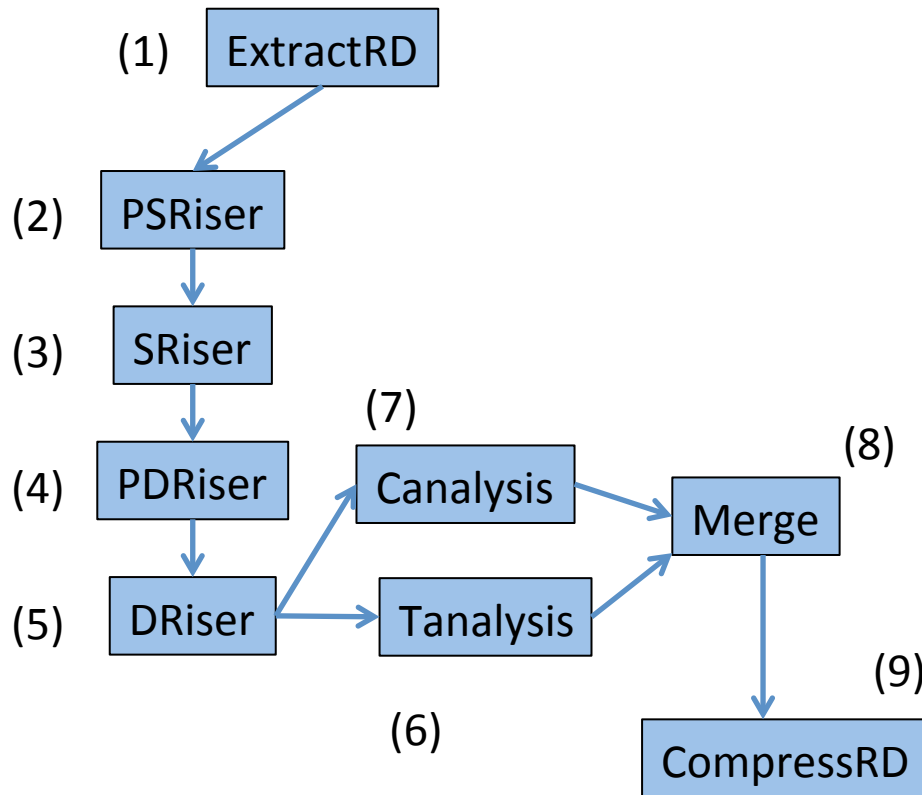
Dispatching Strategy

- In *static* dispatching strategy, activations are pre-allocated to each core before execution.
- In *dynamic* dispatching strategy activations are allocated to cores as a response to a request for activations.



Experimental Evaluation with RFA

Workflow

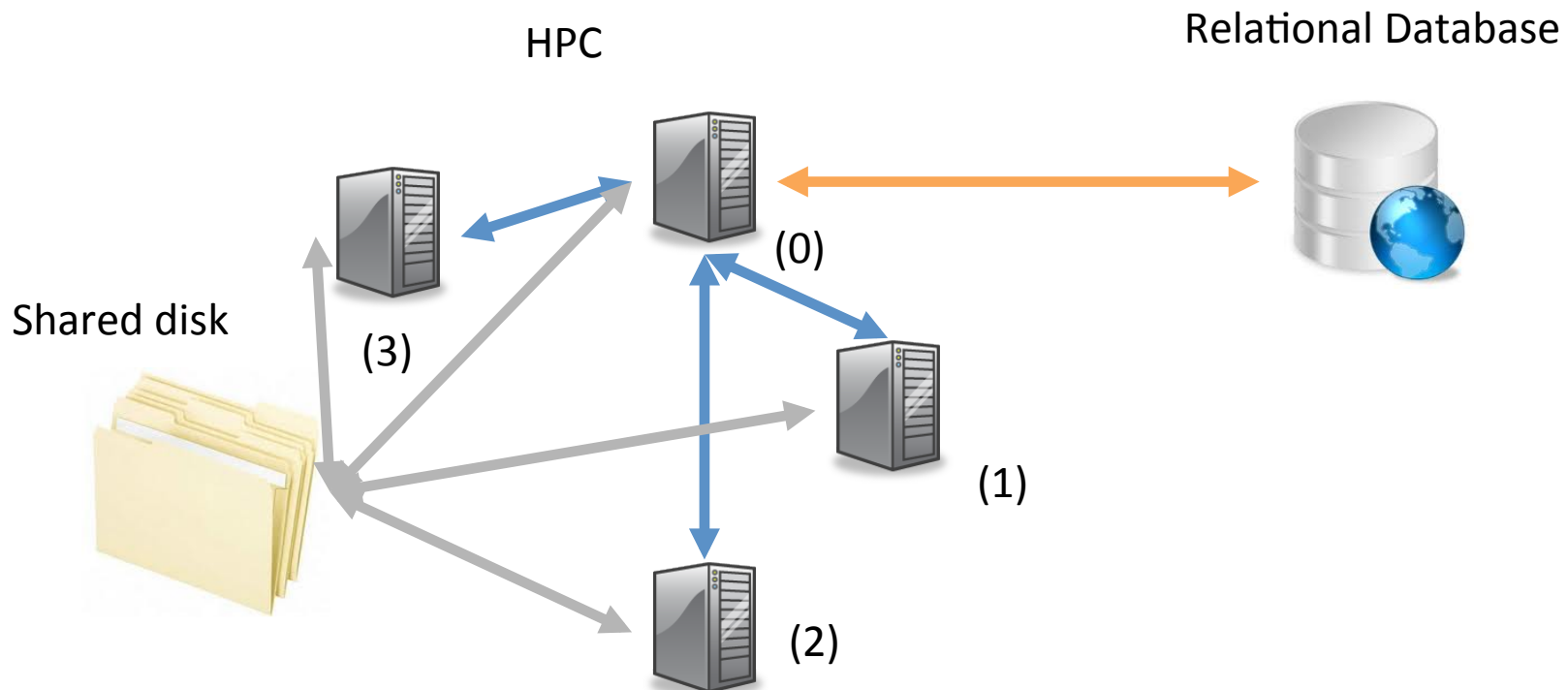


Workflow expressed as algebraic expressions

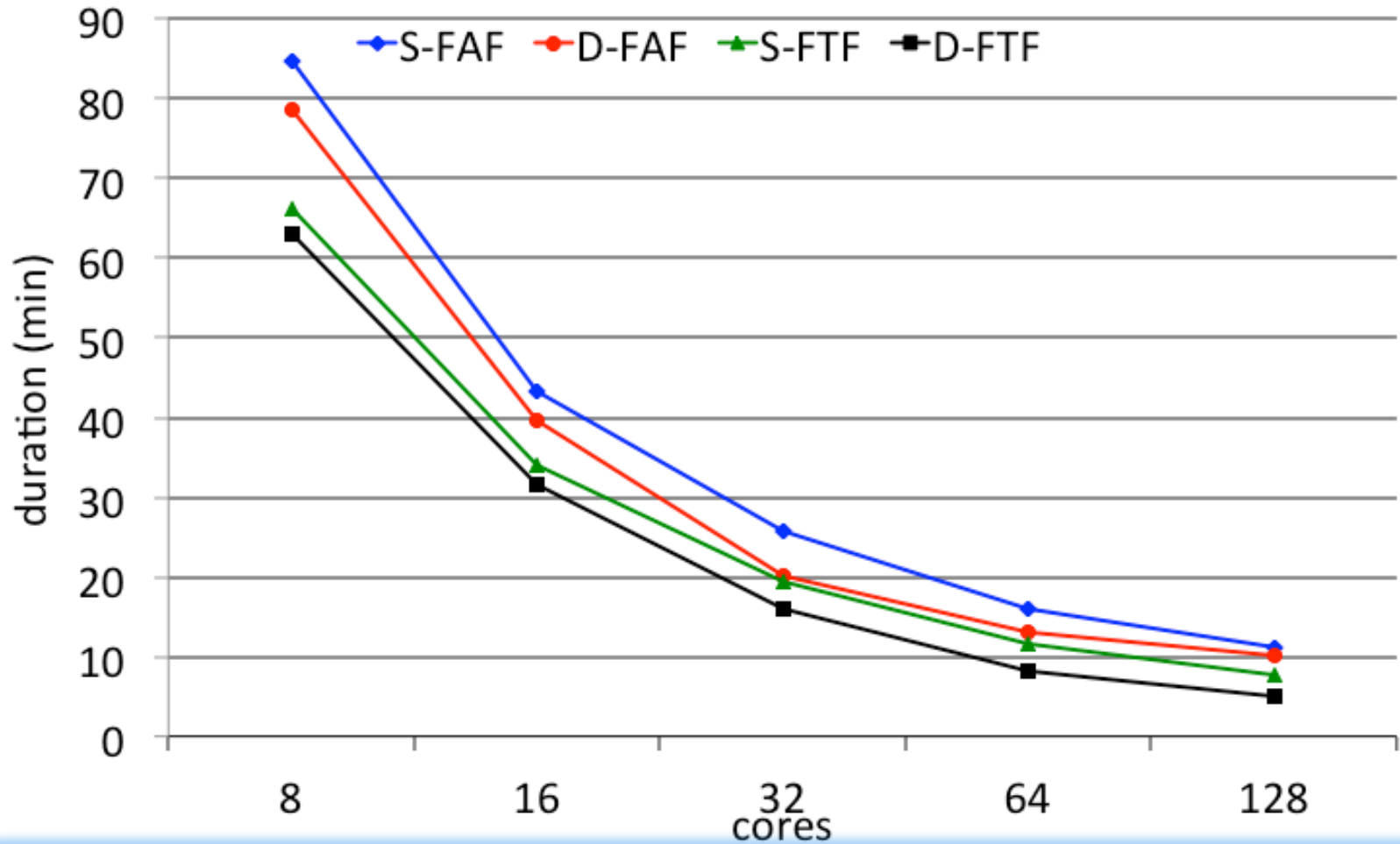
```
T1 ← SplitMap(ExtractRD, R1)
T2 ← Map(PSRiser, T1)
T3 ← Map(SRiser, T2)
T4 ← Map(PDRiser, T3)
T5 ← Map(DRiser, T4)
T6 ← Filter(Tanalysis, T5)
T7 ← Filter(Canalysis, T5)
T8 ← MRQuery(T6 ⋈ T7, {T6, T7})
T9 ← Reduce(CompressRD, T8)
```

Chiron

- Chiron is a data-centric scientific workflow engine
- Implemented in Java using MPJ
- Provenance Stored in Relation Database



Evaluation of RFA Workflow with 358 Case Studies

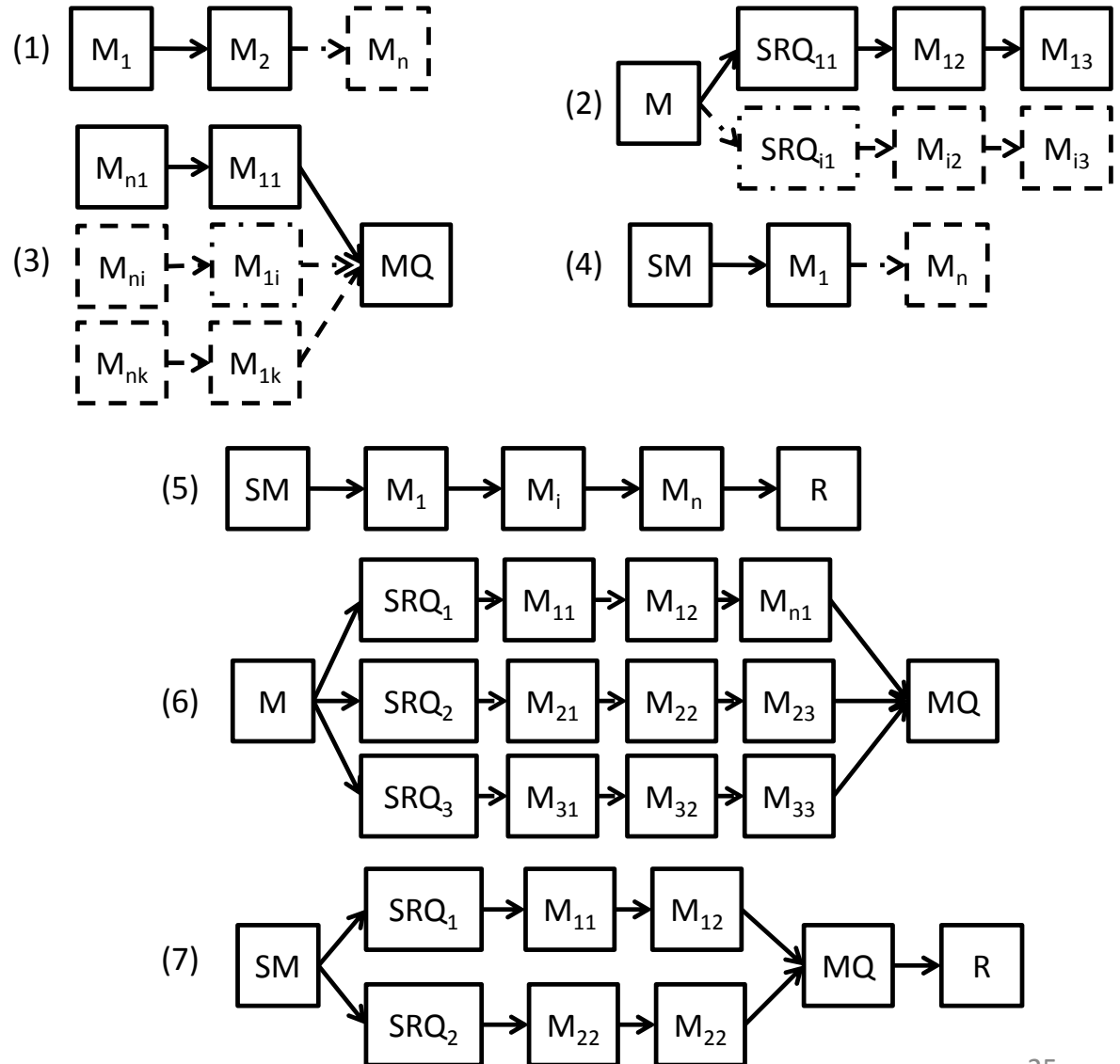


1438 activations, 16765 files
Performance difference of 226% between D-FTF versus S-FAF for 128 cores

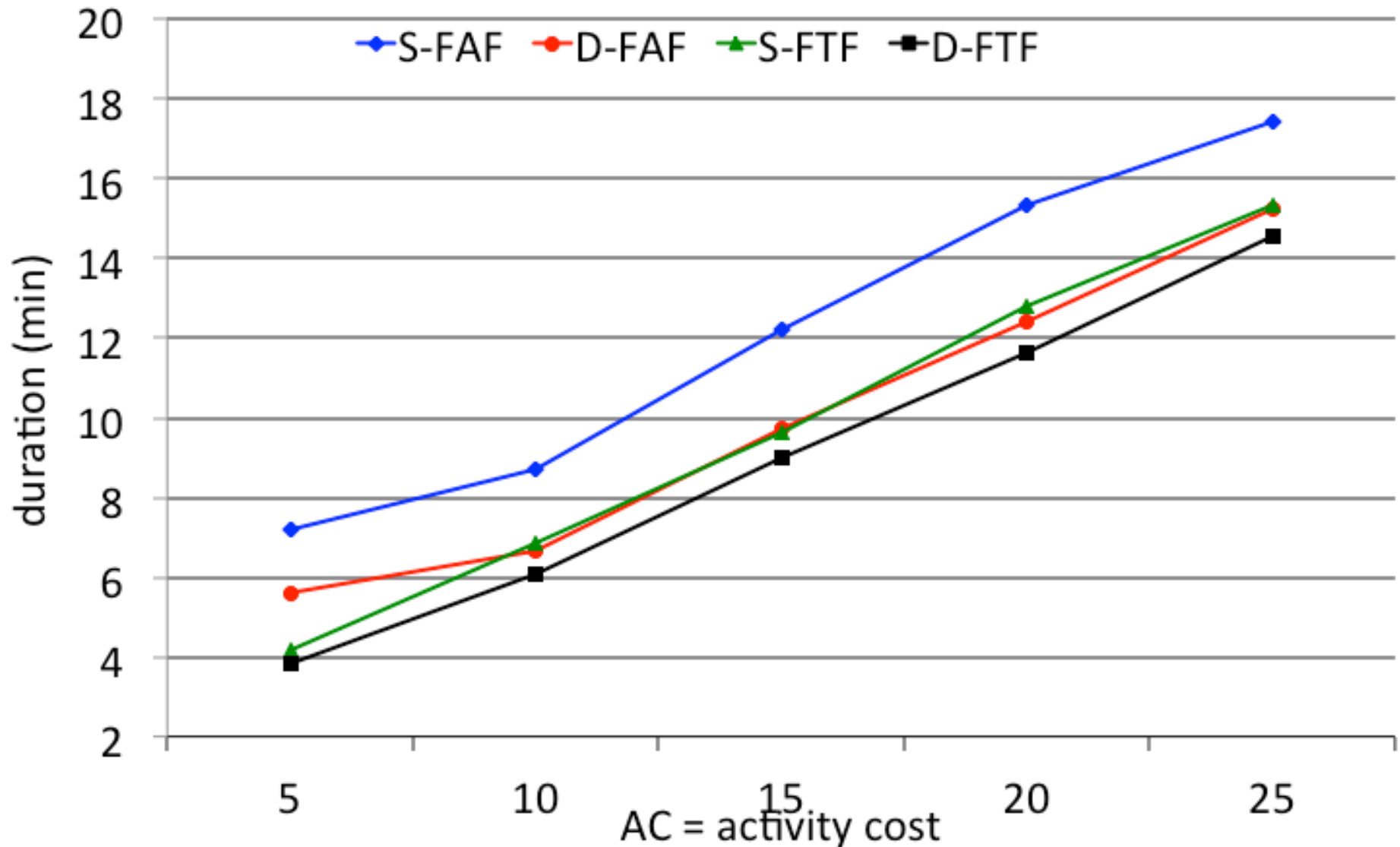
Experiments Using Synthetic Data

Studied variables:

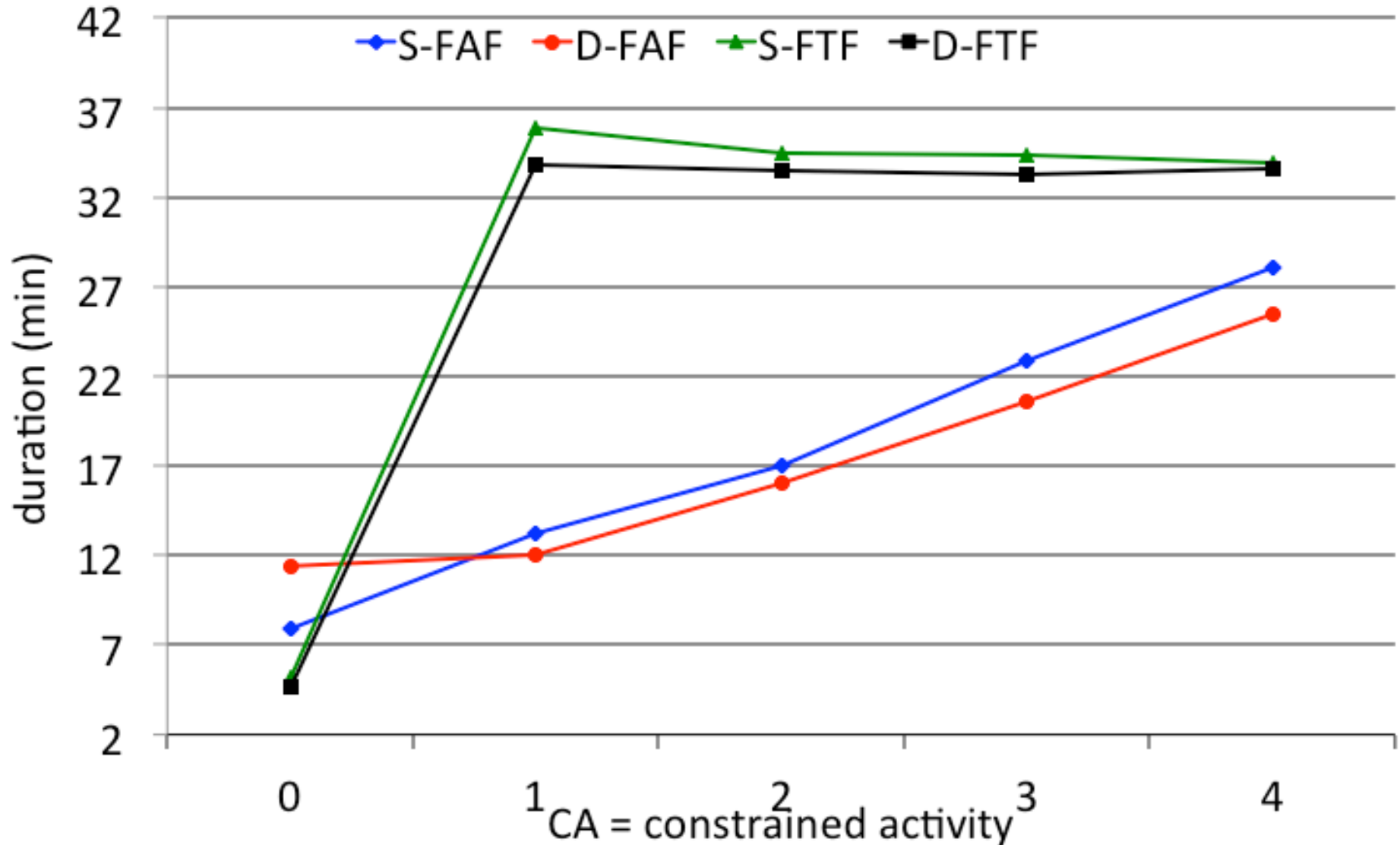
- Activity Cost
- Constrained Activity
- Input Tuples
- Sequence Length
- Fan-In/Fan-Out
- Split Factor/Reduce Factor



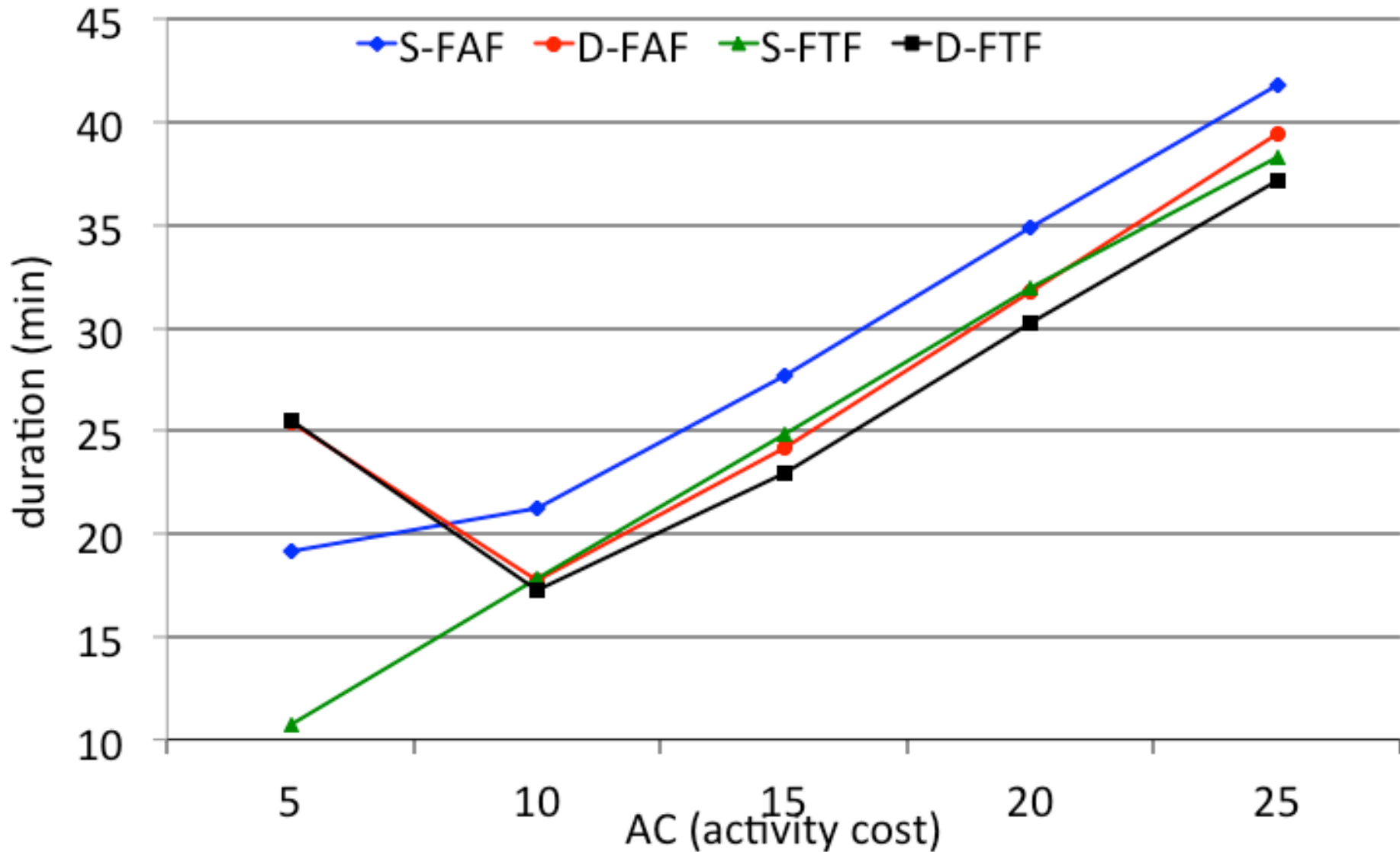
Scenario #1: Sequence of Activities



Scenario #1: Sequence with Constrained Activities



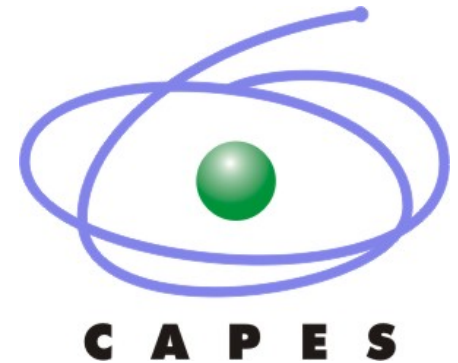
Scenario #6: Mixed Fan-Out/Fan-In



Conclusion

- We proposed an algebraic approach with an execution model for parallel processing
- We conducted a thorough experimental evaluation using Chiron, a data-centric scientific workflow engine.
- We evaluated our approach using Petrobras RFA application and synthetic data.
- The performance results show a variation of up to 226% when we compare the best with the worst performance results.
- As future work we intend to perform automatic optimization through algebraic transformations based on heuristics

Acknowledgements



**Federal
University
Rio de Janeiro**



NACAD
High Performance Computing Center