An Algebraic and Parallel Approach for Scientific Workflows using Chiron

First Brazil-France Workshop

On High Performance Computing and Scientific Data Management Driven by Highly Demanding Applications

Eduardo Ogasawara [2,1], Jonas Dias [1]

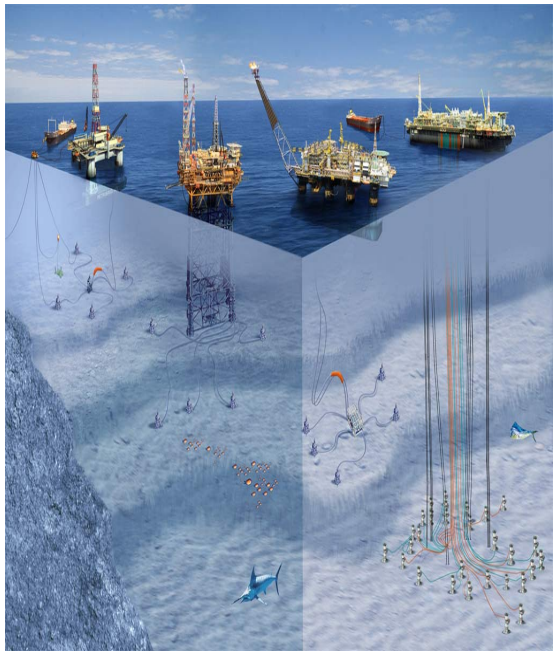
[1] Federal University of Rio de Janeiro, Brazil

[2] CEFET/RJ

Federal University Rio de Janeiro

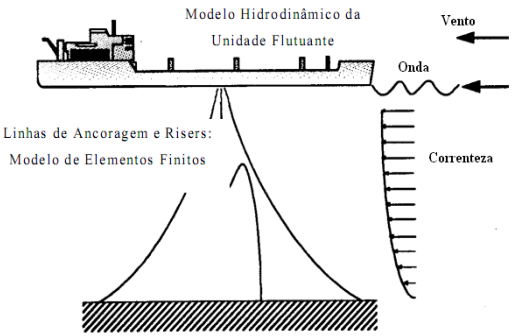
CEFET/RJ

Risers' Fatigue Analysis in Ultra-Deep Waters

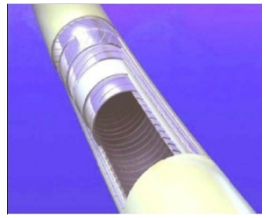


Input Data to simulate Environment conditions: Waves, wind, currents, bathymetry, etc.

1. Coupled movement Analysis (TPN or Prosim)

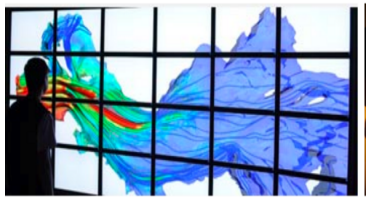


Generates large amount of data … (finite element meshes)



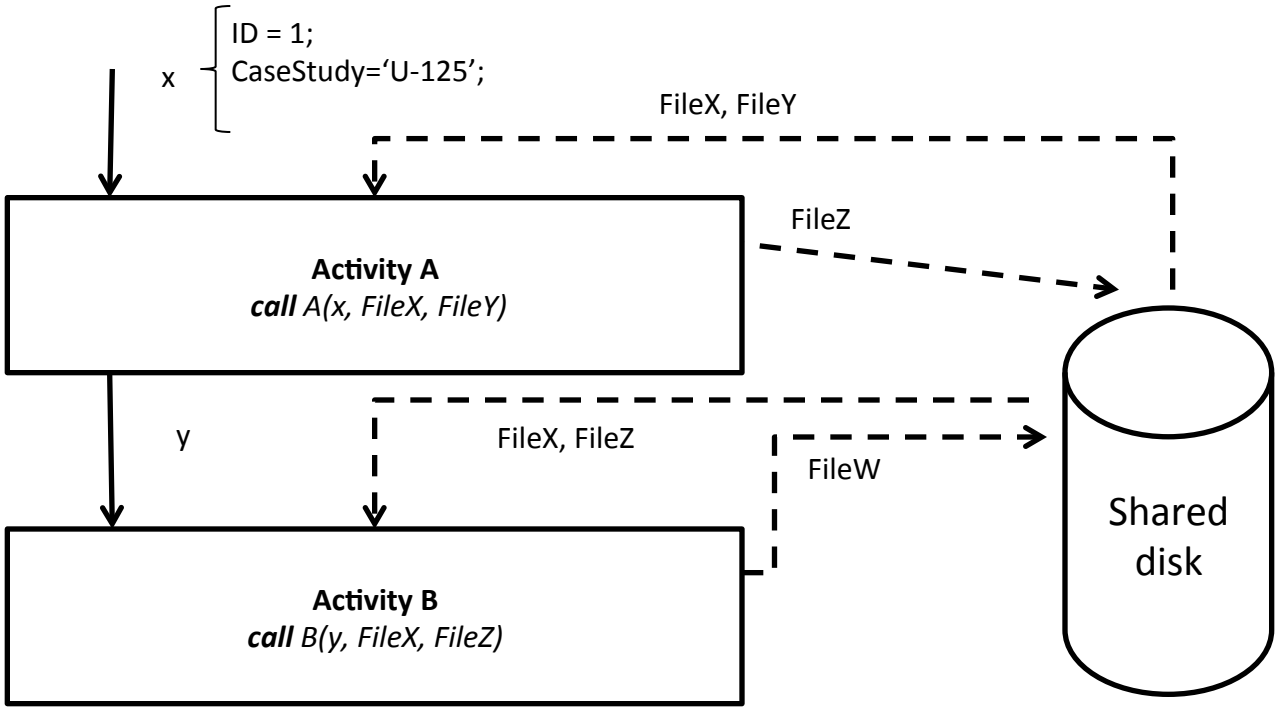
2. … to do Structural Analysis of Risers (ANFLEX)

3. Results are analyzed POSFAL



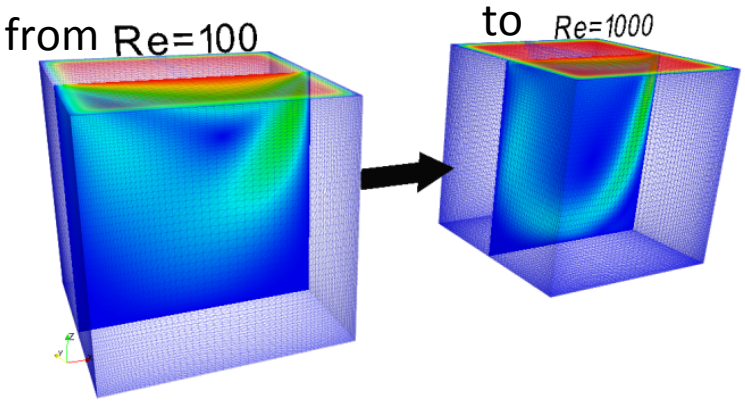
Estimate risers lifetime

Dissecting a Scientific Workflow



Data-centric experiments

- 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 data-centric experiments we have multiple inputs for the workflow.



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

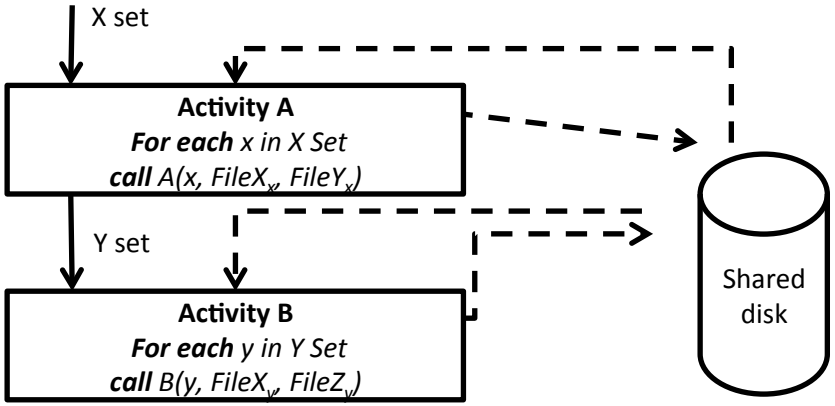
Current Approaches for Data Centric Workflows

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

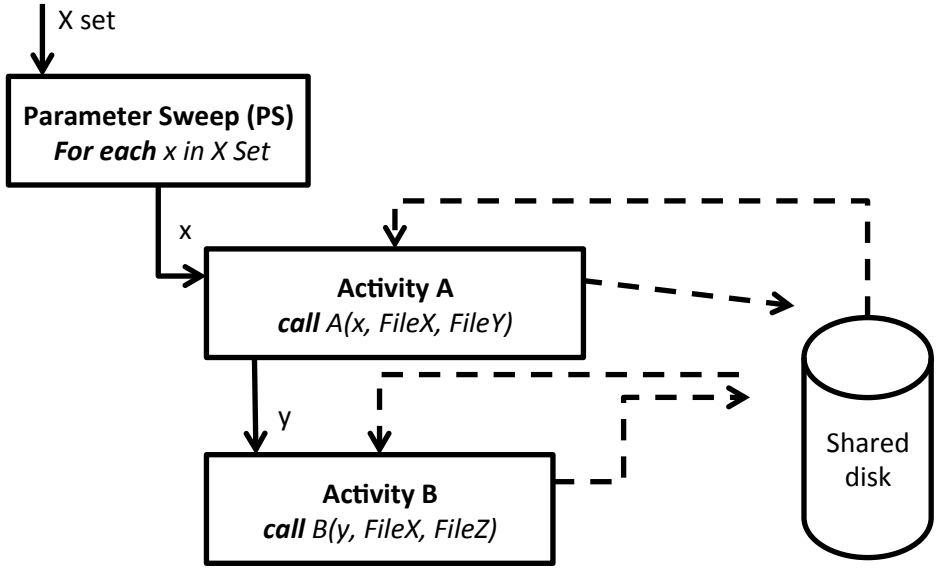
Data-centric workflows are natural candidates for parallel processing

Common approaches for supporting data parallelism

Adaptation to include for each in all activities



Adaptation to include a for each invoking a group of activities



Fixed (rigid) execution plan

Problems

- Lack of uniform data model
  - Demands the adoption of an uniform data model to represent workflows that are agnostic to execution environment
- Execution models for data centric workflows
  - Demands the optimization of the parallel workflow executing considering this agnostic model
  - Without optimization, scientists should code their workflows using low level primitives
- How, when and in what granularity we should store provenance

Objectives

- Propose an uniform data model and an agnostic workflow representation
- Evaluate opportunities for workflow optimization that consider the entire workflow
  - Propose an optimization process for workflow execution
- Consider the execution of workflows in the same way as query execution plans in databases

Solution: An Algebraic Approach

- Data-Centric algebra for scientific workflows
  - Relations as data model for consumption and production
  - Algebraic operators that provide semantics to activities
  - Algebraic expressions provide an agnostic workflow representation
  - 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($\mathcal{R}$)*

RID	CaseStudy	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 ← SplitMap(Y, a, R)

R

RID	RdZip
1	Project1.zip
2	Project2.zip

T ← SlipMap(extractRD, 'RdZip', R)

T

RID	Study	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

Reduce Activity (Reduce)

$T \leftarrow Reduce(Y, g_A, R)$

R

RID	Study	SsSai	DdSai	MEnv
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 ← Reduce(CompressRD, {'RID'}, R)

T

RID	RdResultZip
1	ProjectResult1.zip
2	ProjectResult2.zip

Single Relation Query Activity (SRQuery)

T ← SRQuery(qry, R)

R

RID	Study	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

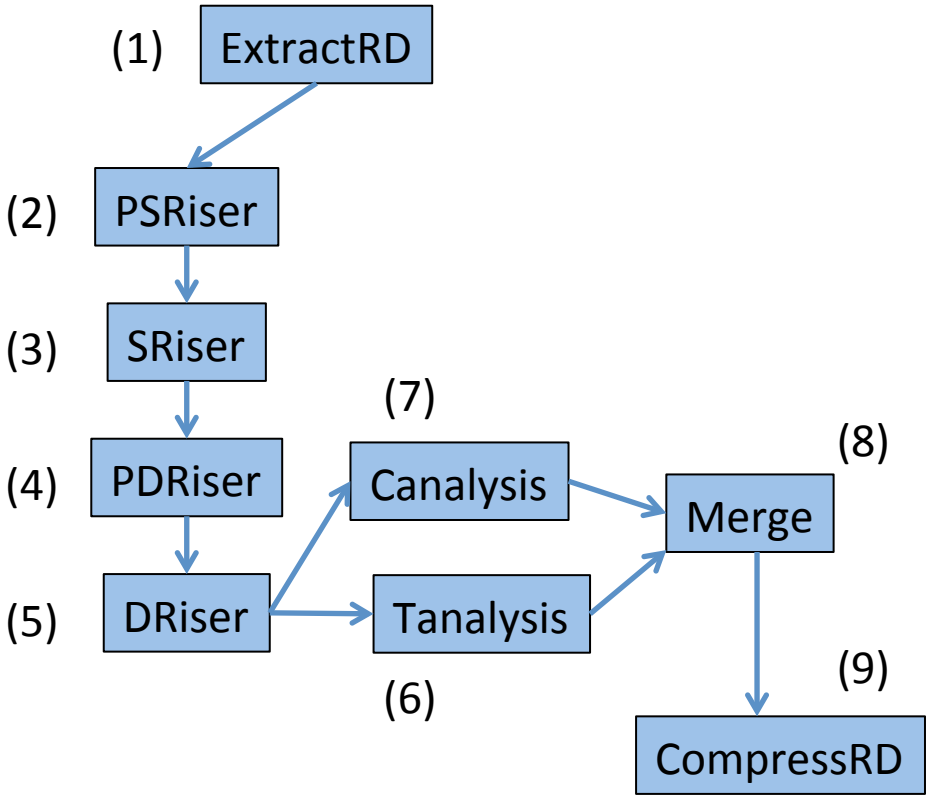
$$T \leftarrow SRQuery(\pi_{RID,\ Study,\ SsSai,\ Curvature}(\sigma_{Curvature>1}(R)),\ R)$$

T

RID	Study	SsSai	Curvature
1	U-125	U-125Ss.SAI	1.5
1	U-127	U-127Ss.SAI	1.2

Workflow specification expressed as algebraic expressions

Workflow



Algebraic expressions

$T_1 \leftarrow \text{SplitMap}(\text{ExtractRD}, R_1)$

$T_2 \leftarrow \text{Map}(\text{PSRiser}, T_1)$

$T_3 \leftarrow \text{Map}(\text{SRiser}, T_2)$

$T_4 \leftarrow \text{Map}(\text{PDRiser}, T_3)$

$T_5 \leftarrow \text{Map}(\text{DRiser}, T_4)$

$T_6 \leftarrow \text{Filter}(\text{Tanalysis}, T_5)$

$T_7 \leftarrow \text{Filter}(\text{Canalysis}, T_5)$

$T_8 \leftarrow \text{MRQuery}(T_6 \bowtie T_7, \{T_6, T_7\})$

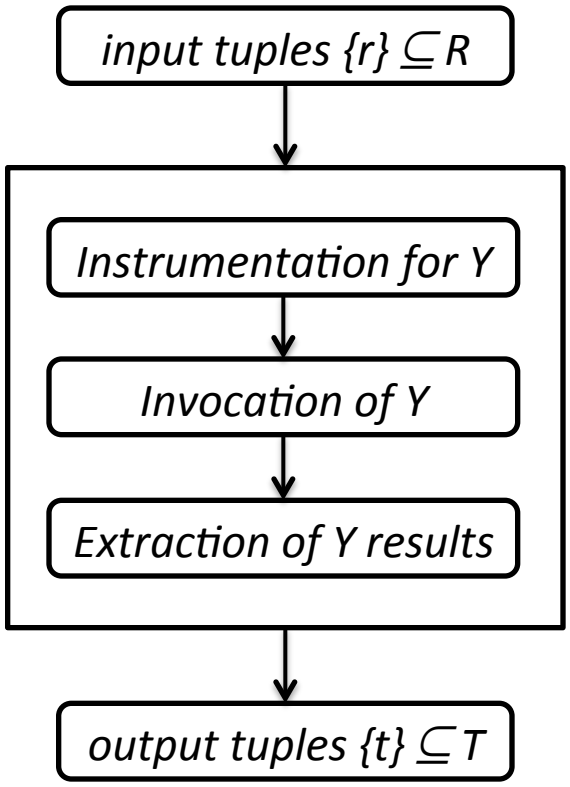
$T_9 \leftarrow \text{Reduce}(\text{CompressRD}, T_8)$

Workflow Execution Model

- Activity Activation
- Strategies for execution
  - Dataflow Strategy
  - Dispatching Strategy
- Algebraic optimization

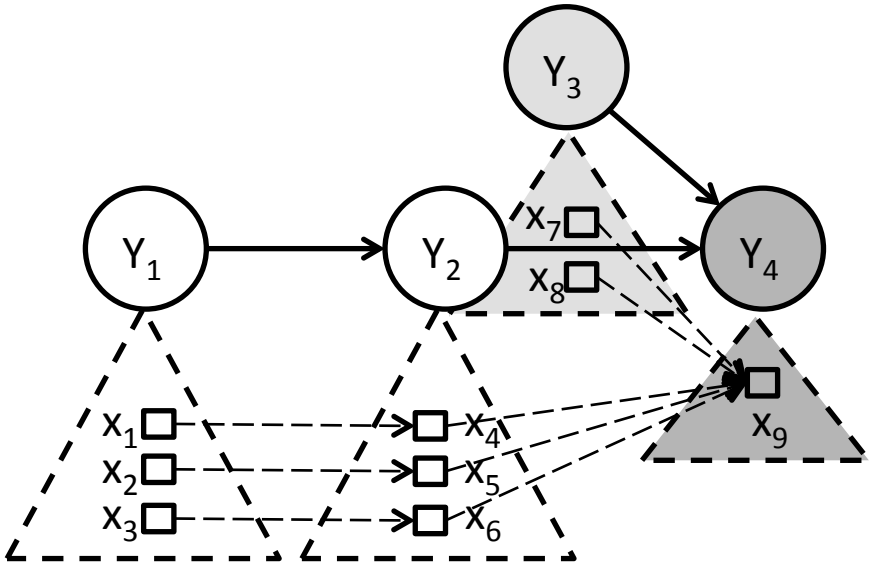
Activity Activation

- 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



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.

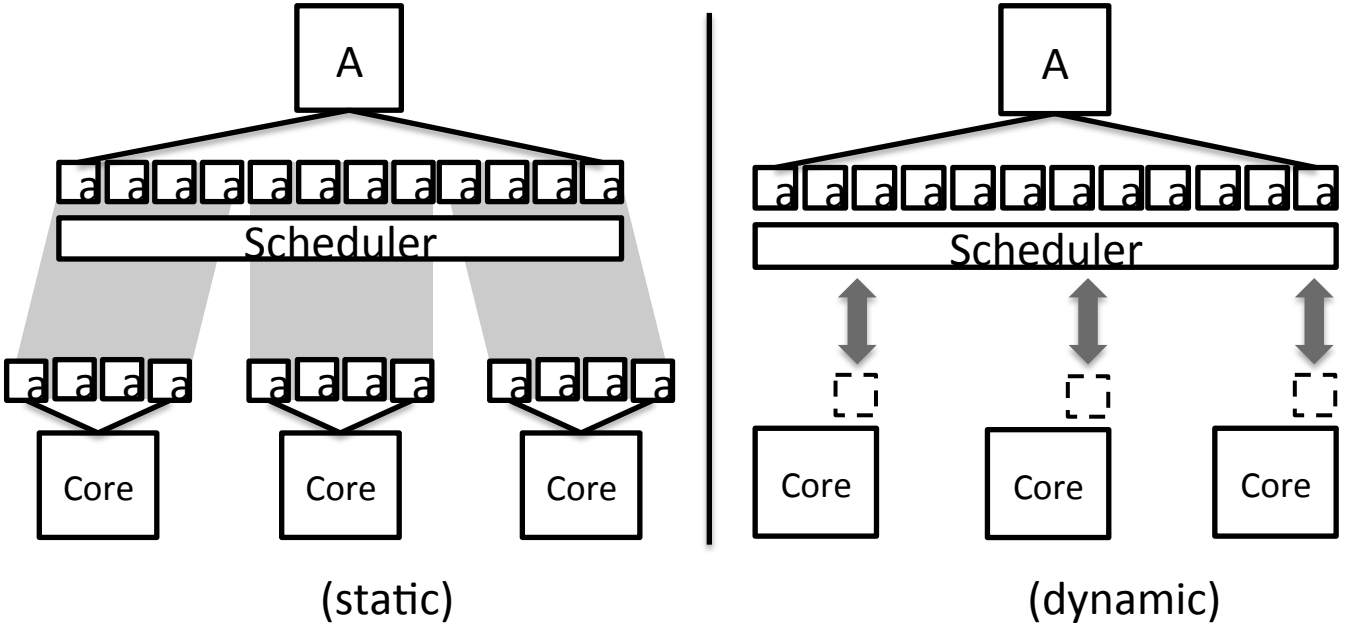


FTF:
{<x_1, x_4>, <x_2, x_5>, <x_3, x_6>}

FAF:
{<x_1>,<x_2>,<x_3>,
<x_4>,<x_5>,<x_6>}

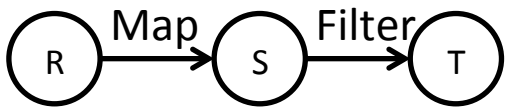
Dispatching Strategy

- 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.

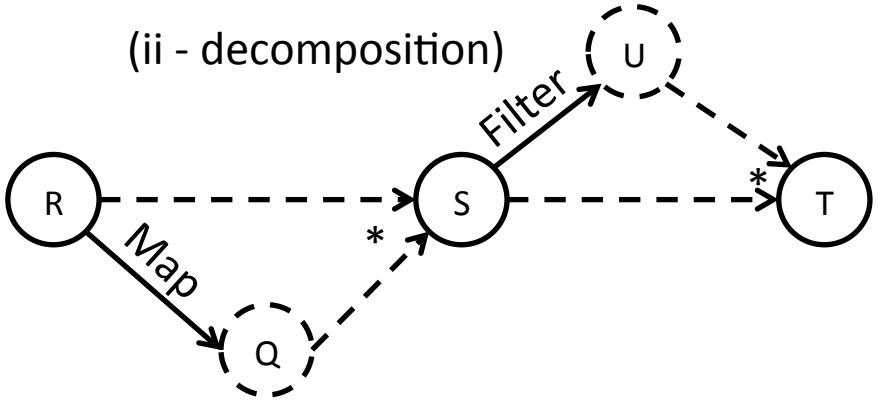


Algebraic transformation

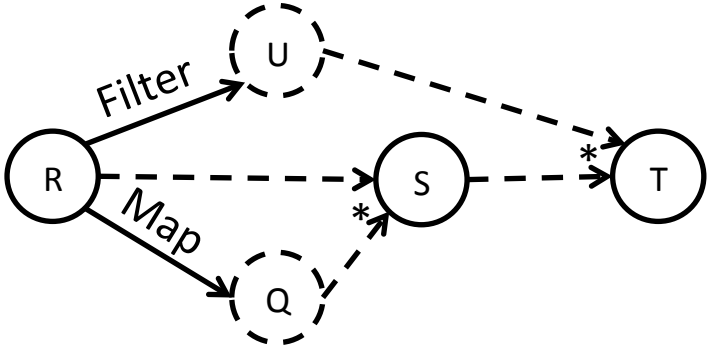
(i - *workflow* – relation perspective)



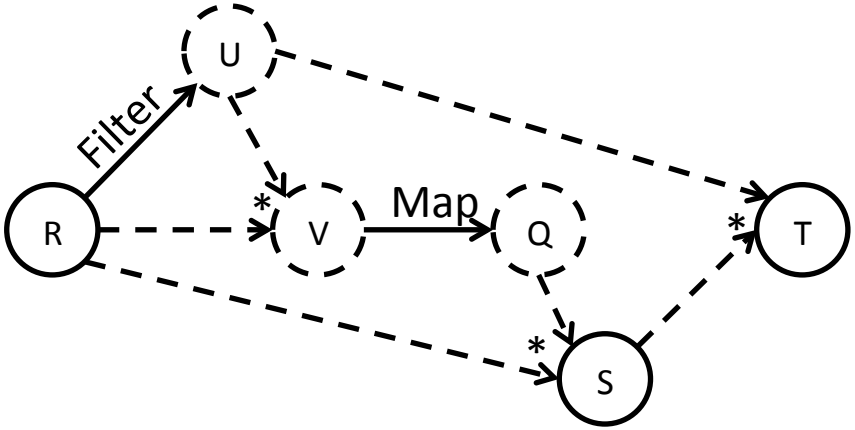
(ii - decomposition)



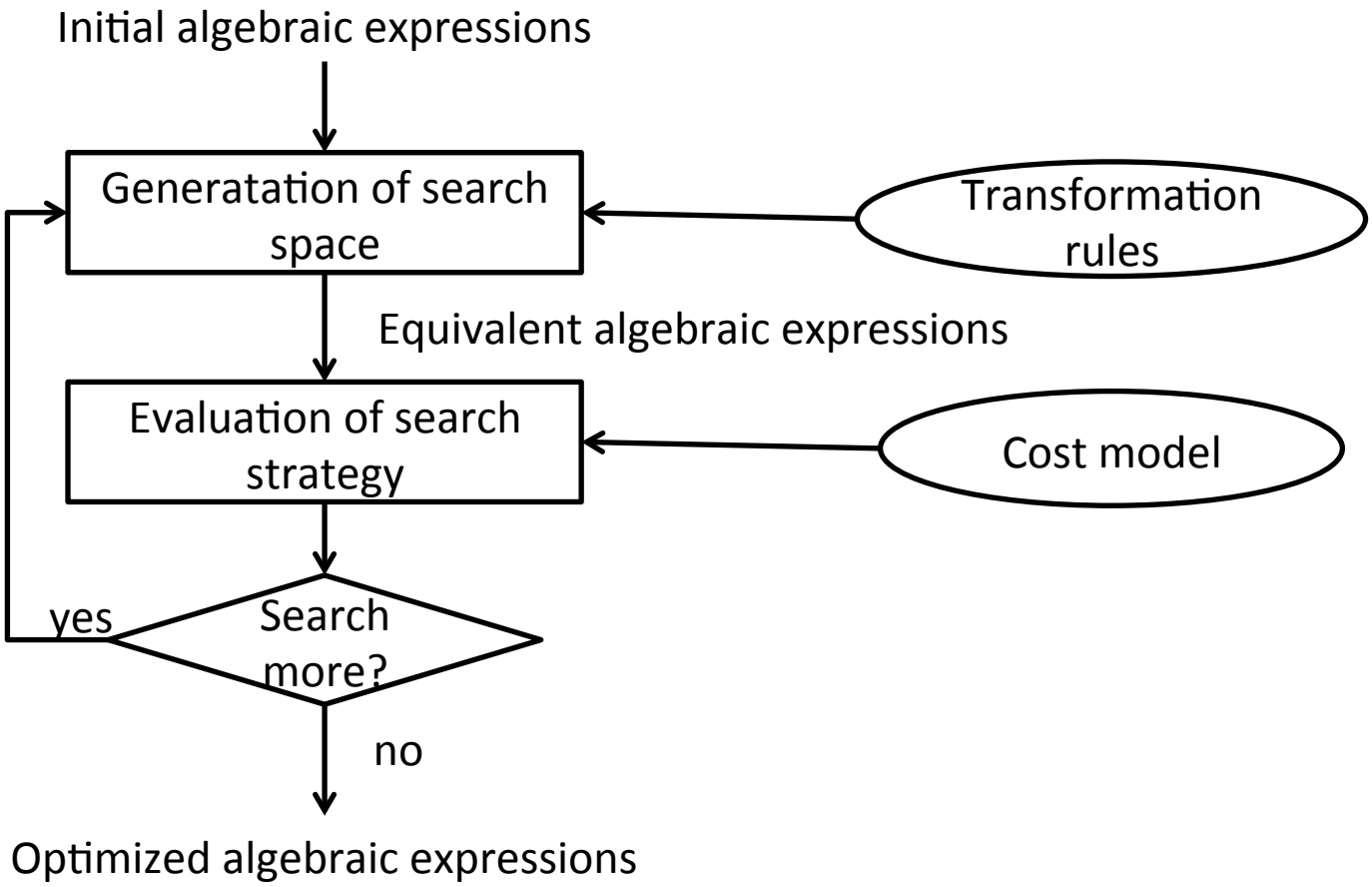
(iiii - anticipation)



(iv - procastination)

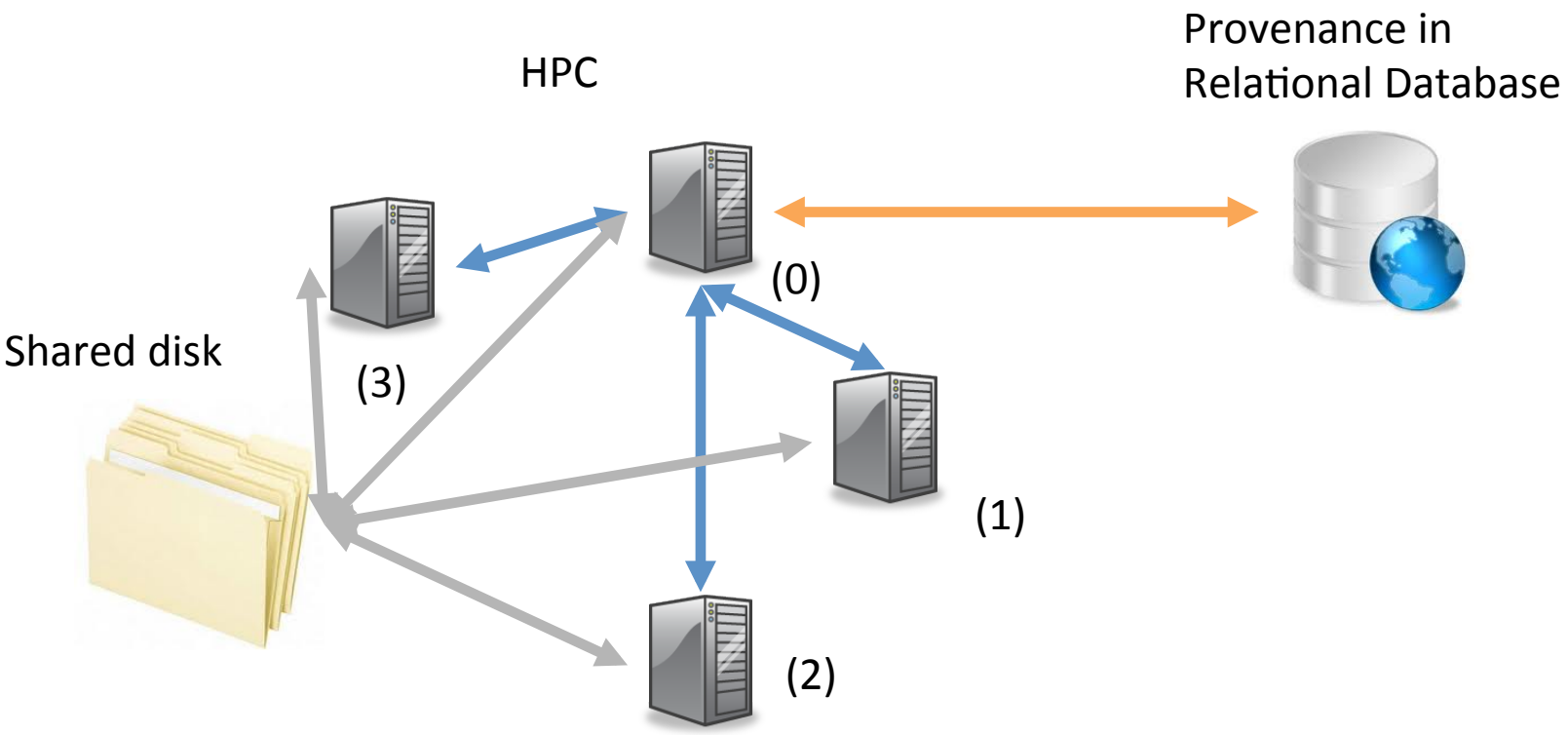


Workflow optimization process

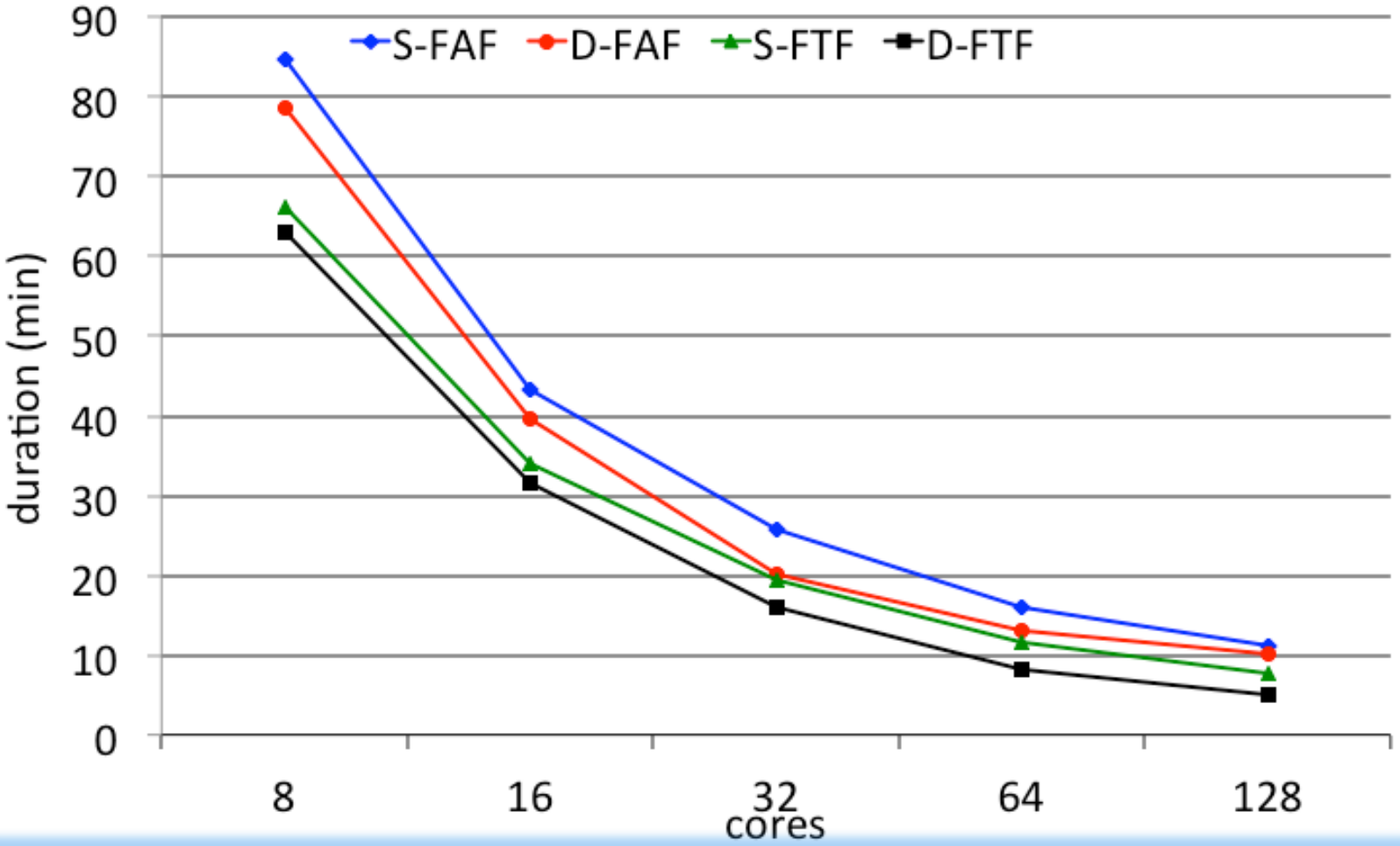


Chiron

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

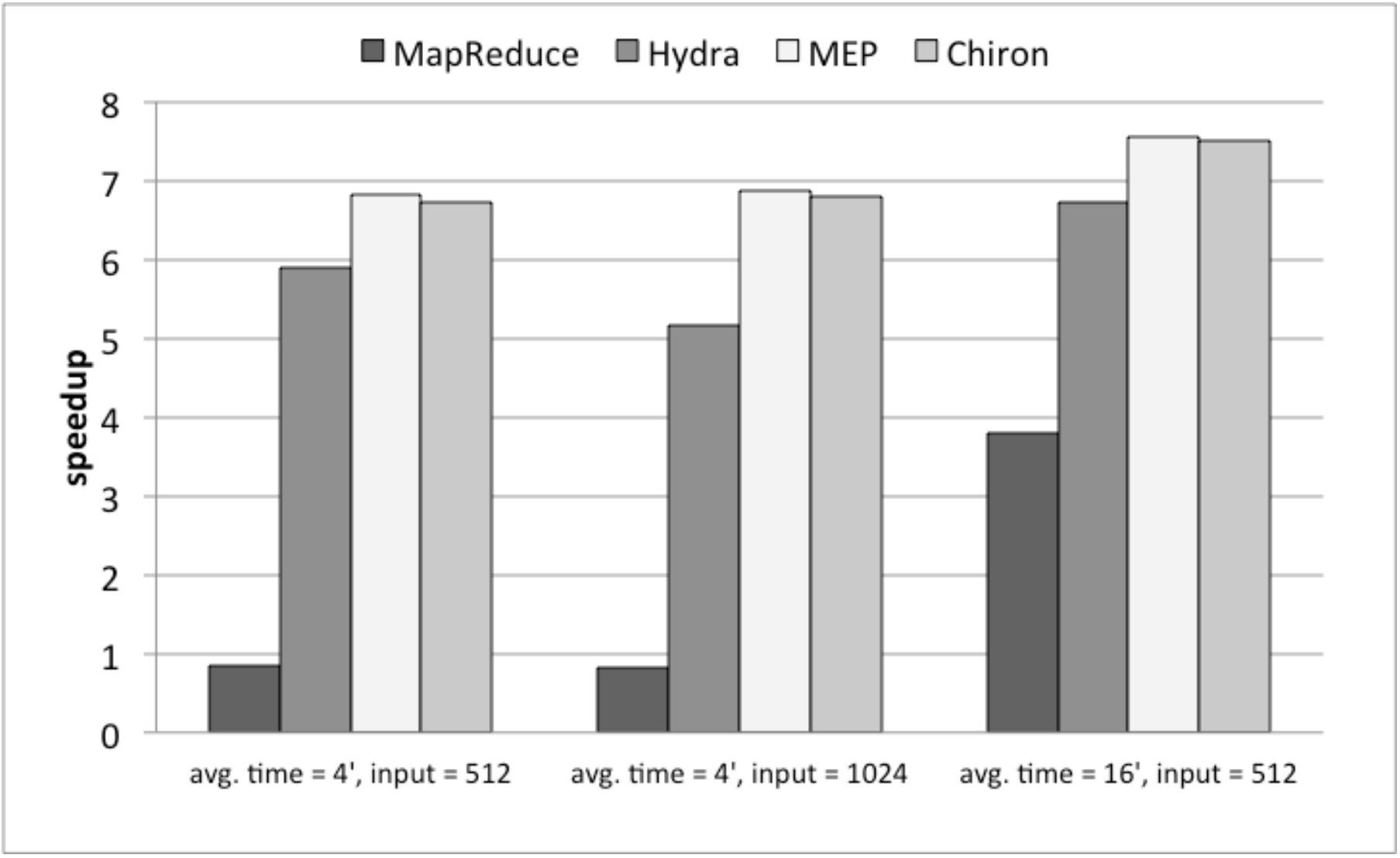


Evaluation of RFA Workflow with 358 Case Studies

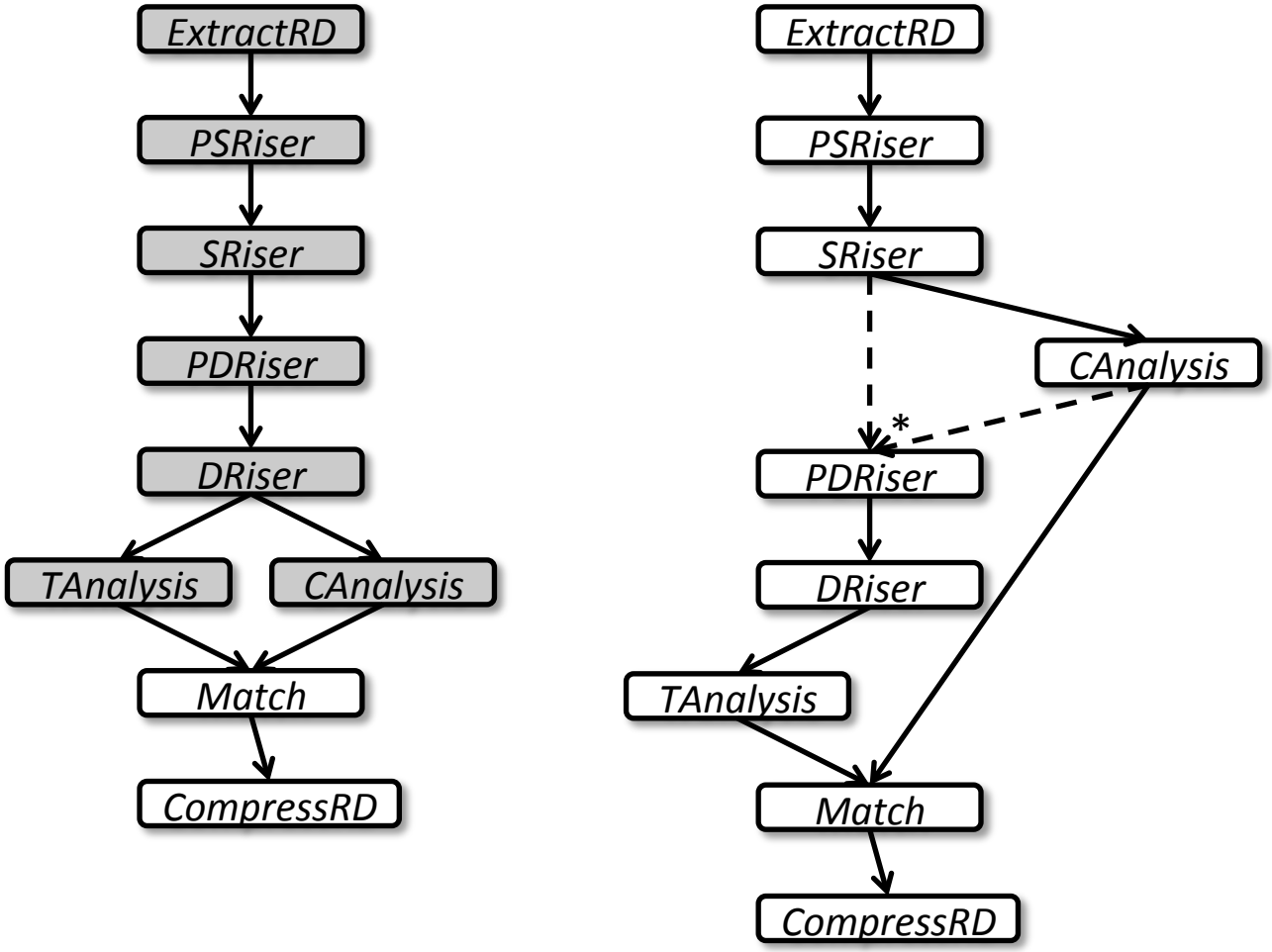


1438 activations, 16765 files
Performance difference of 226% between D-FTF versus S-FAF for 128 cores

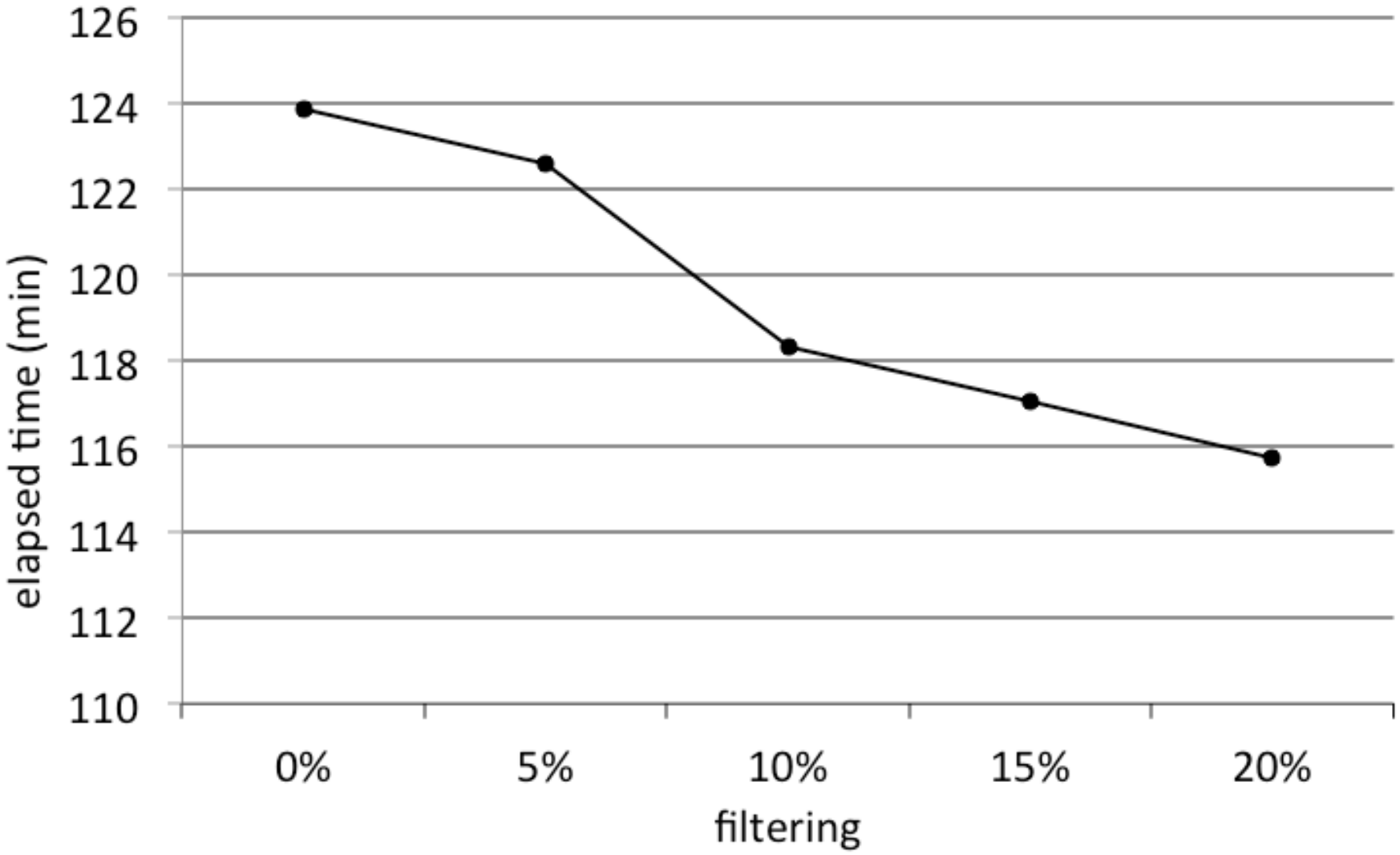
Comparison of Chiron against other approaches



Algebraic optimization in RFA workflow



Evaluation of algebraic optimization in RFA workflow



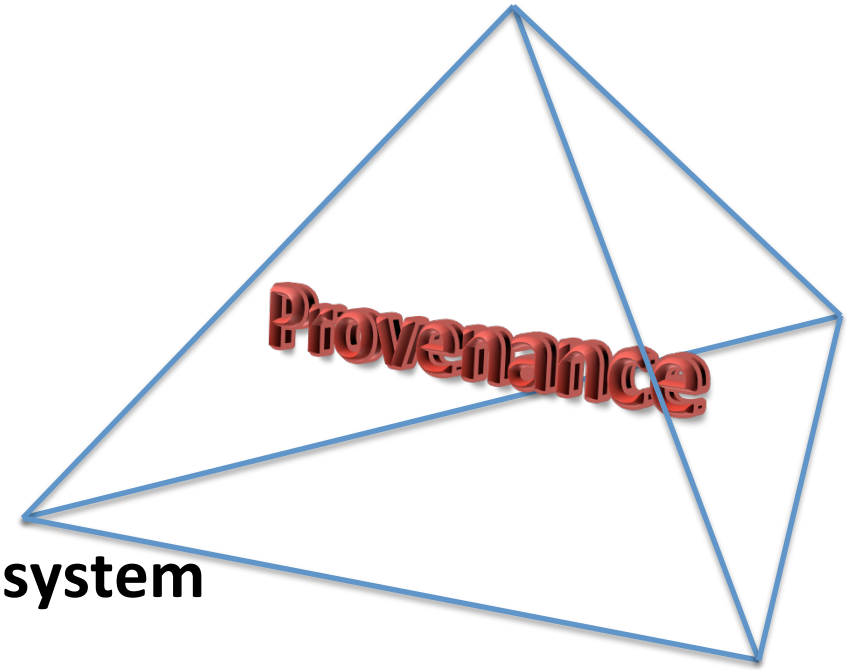
Exploratory Nature in Experiments

- The execution of the same experiment repeatedly
- Exploring parameters or input datasets
  - Parameter Sweeps
  - Fine-tuning
  - Iterative Methods
- Time consuming workflows
  - Analysis of Partial Results
- Provenance Data
  - Real time analysis
  - Reproducibility of the experiment

Dynamic Workflows

- The experiment life cycle is intrinsically dynamic
  - Workflows also need to be dynamic
    - Distributed and collaborative workflow design
    - Workflow adaptation based on *external events*
      - Human intervention and **dynamic steering**
    - **Efficient query system** in support to provenance data
      - Support provenance information browsing and traversing
    - Forms to **explore slices of the parameter space** and compare the results of different configurations
  - **Provenance** is the key for dynamic analysis
    - Access and query meta-data and some results from ongoing experiments

Interactive Workflows



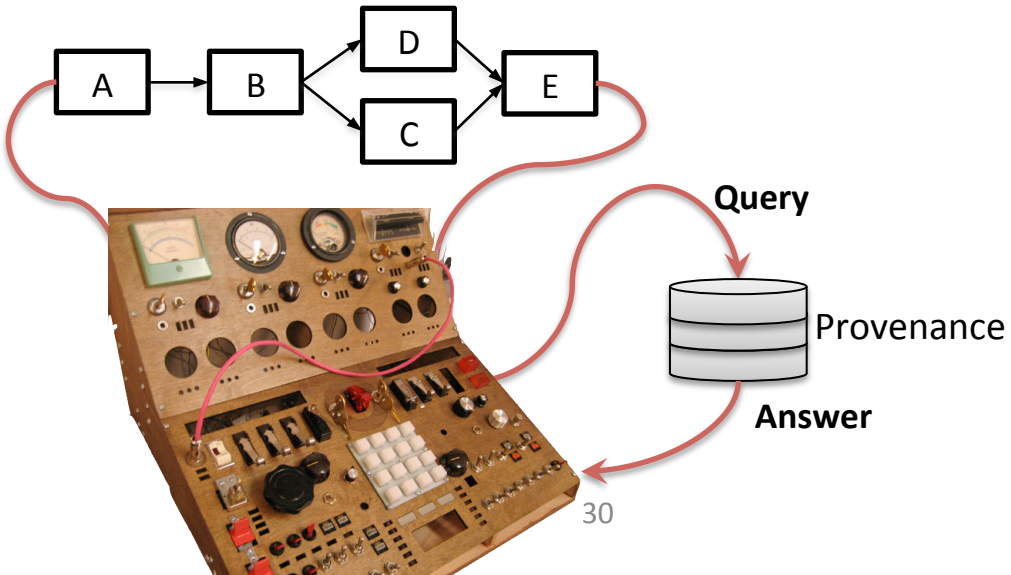
Dynamic Steering

Efficient query system

Explore slices of the parameter space

Steering the workflow

- Purpose
  - Visualize and analyze **provenance** data to steer the Wf
  - Make adjustments during the execution
    - Parameter refinements
    - Filtering options
    - Number of iterations of a loop
- How to be in control
  - Runtime Provenance
    - Real time analysis
  - "Adjustable knobs"
    - Adjust Parameters

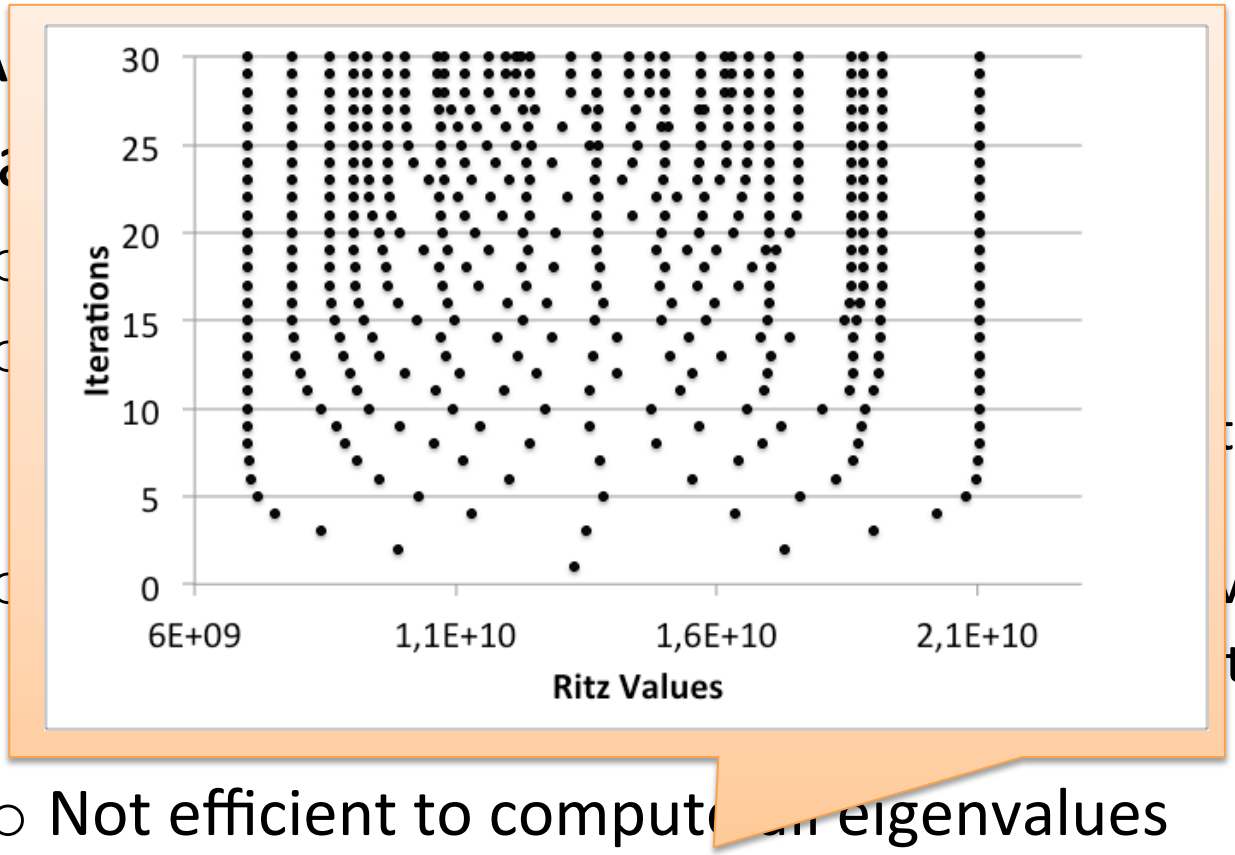


Evaluation

- Adjusting a runtime parameter
- Lanczos algorithm
  - **R**educed **O**rder **M**odel scenario
  - Simplest Krylov sub-space method
    - Construct **iteratively** an orthonormal basis in the Krylov sub-space
  - Computes approximate eigenvalues (Ritz values)
    - The number of eigenvalues is associated with the number of iterations
  - Not efficient to compute all eigenvalues
    - The iteration is usually **truncated**
      - After obtaining a given number of eigenvalues

Evaluation

- A
- La
  - ○
  - ○
  - ○
  - ○ Not efficient to compute all eigenvalues
    - The iteration is usually **truncated**
      - After obtaining a given number of eigenvalues



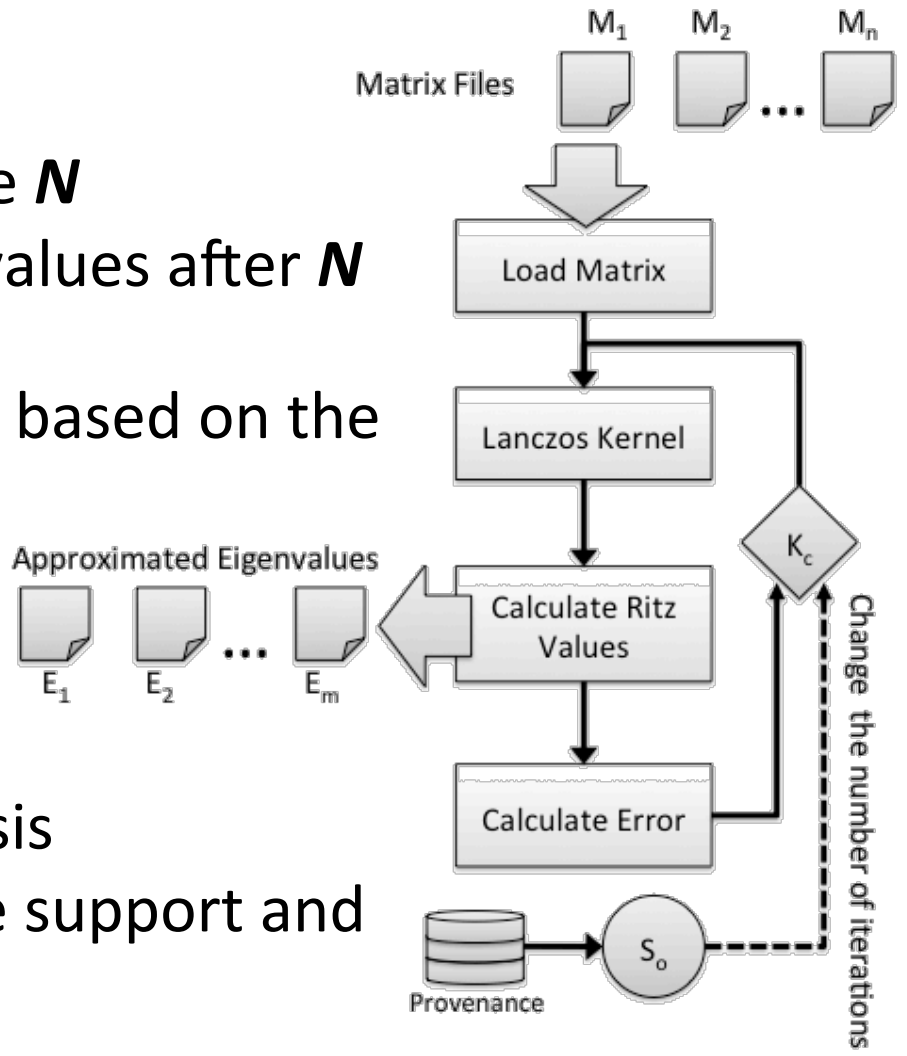
the Krylov sub-

values)

the number of

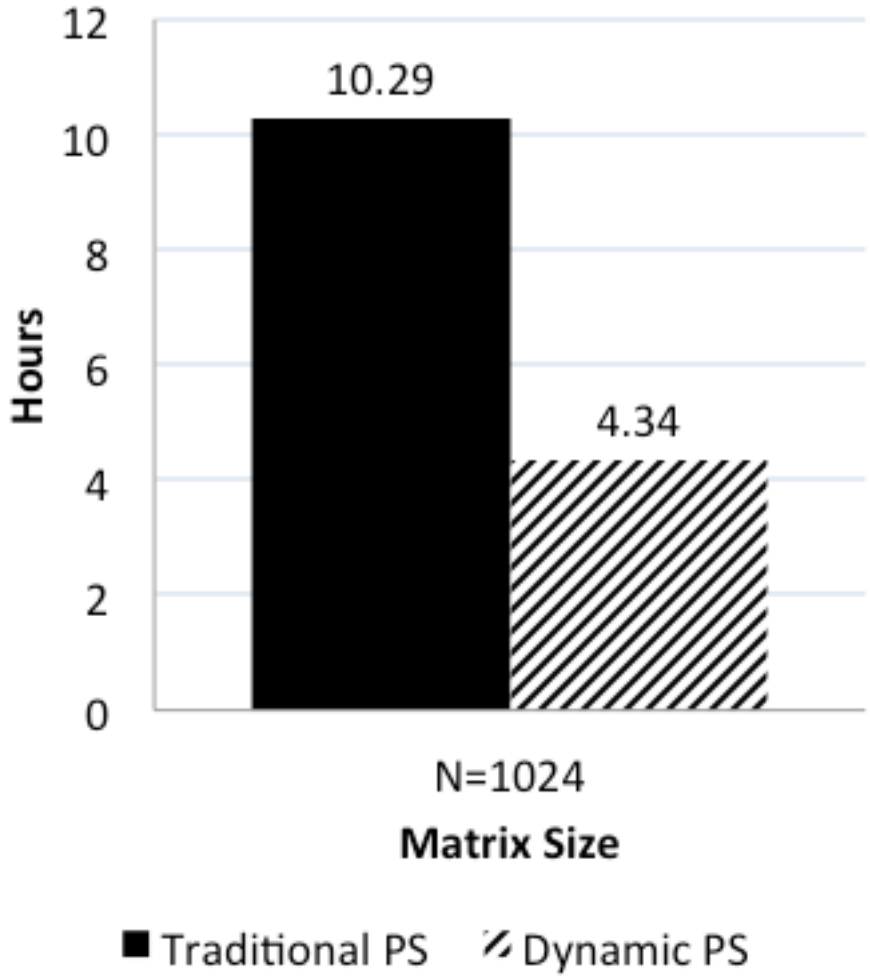
Lanczos Workflow

- Dynamic parameter sweep
  - Consumes ***m*** matrices of size ***N***
  - Can produce all the ***n*** eigenvalues after ***N*** iterations
  - S_o can truncate the iteration based on the error
- Modeled on Chiron
  - Parallel workflow engine
  - Distributed provenance
  - Real time provenance analysis
  - Modules to add the iterative support and the new controls



Case Study

- 128 ROMs with random material properties
  - Problem size: N=1024
  - Arbitrary truncation
    - After 180 iterations
  - Dynamic truncation
    - S_o truncates the iteration dynamically

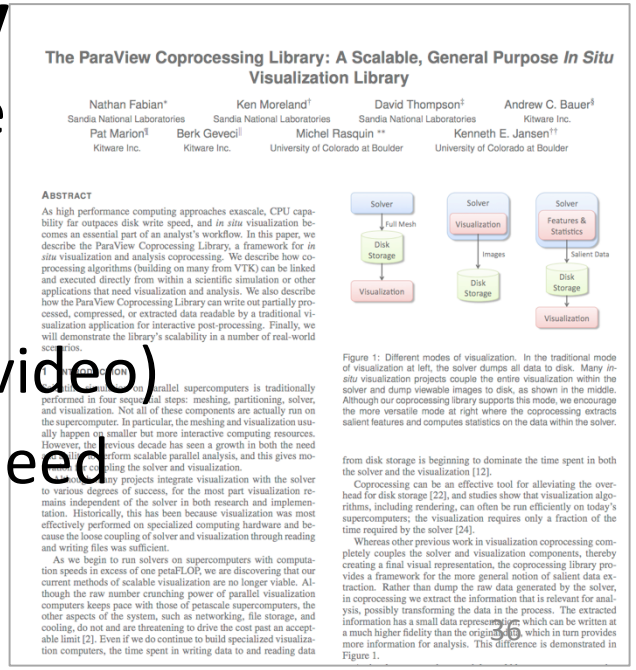


Current Goals

- Evaluation Loops
  - Support iterative experiments
  - Modify the behavior of the execution according to a given evaluation
  - Use provenance data extracted from results
    - Can analyze specific parameters
    - Or can analyze a behavior over experiment data
- Exploring slices of parameter space
  - Good for uncoupled analysis
  - Save computation after partial results
    - Discarding input data

Large Scale Visualization

- Dense meshes simulations
  - Can easily reach **Terabytes** of data
- Processor speed increased, but disk I/O and storage did not followed the same growth
- New **Co-Processing Paraview Library**
  - Co-Processing module developed in the simulator
  - **Runtime visualization**
  - Simulation data is not stored (only the video)
  - If you need to change something, you need to re-run the simulation



The ParaView Coprocessing Library: A Scalable, General Purpose *In Situ* Visualization Library

Nathan Fabian* Sandia National Laboratories
Ken Moreland† Sandia National Laboratories
David Thompson‡ Sandia National Laboratories
Andrew C. Bauer§ Kitware Inc.
Pat Marion¶ Kitware Inc.
Berk Geveci‖ Kitware Inc.
Michel Rasquin ** University of Colorado at Boulder
Kenneth E. Jansen†† University of Colorado at Boulder

ABSTRACT

As high performance computing approaches exascale, CPU capability far outpaces disk write speed, and *in situ* visualization becomes an essential part of an analyst's workflow. In this paper, we describe the ParaView Coprocessing Library, a framework for *in situ* visualization and analysis coprocessing. We describe how coprocessing algorithms (building on many from VTK) can be linked and executed directly from within a scientific simulation or other applications that need visualization and analysis. We also describe how the ParaView Coprocessing Library can write out partially processed, compressed, or extracted data readable by a traditional visualization application for interactive post-processing. Finally, we will demonstrate the library's scalability in a number of real-world scenarios.

Solver → Full Mesh → Disk Storage → Visualization
Solver → Visualization → Images → Disk Storage
Solver → Features & Statistics → Salient Data → Disk Storage → Visualization

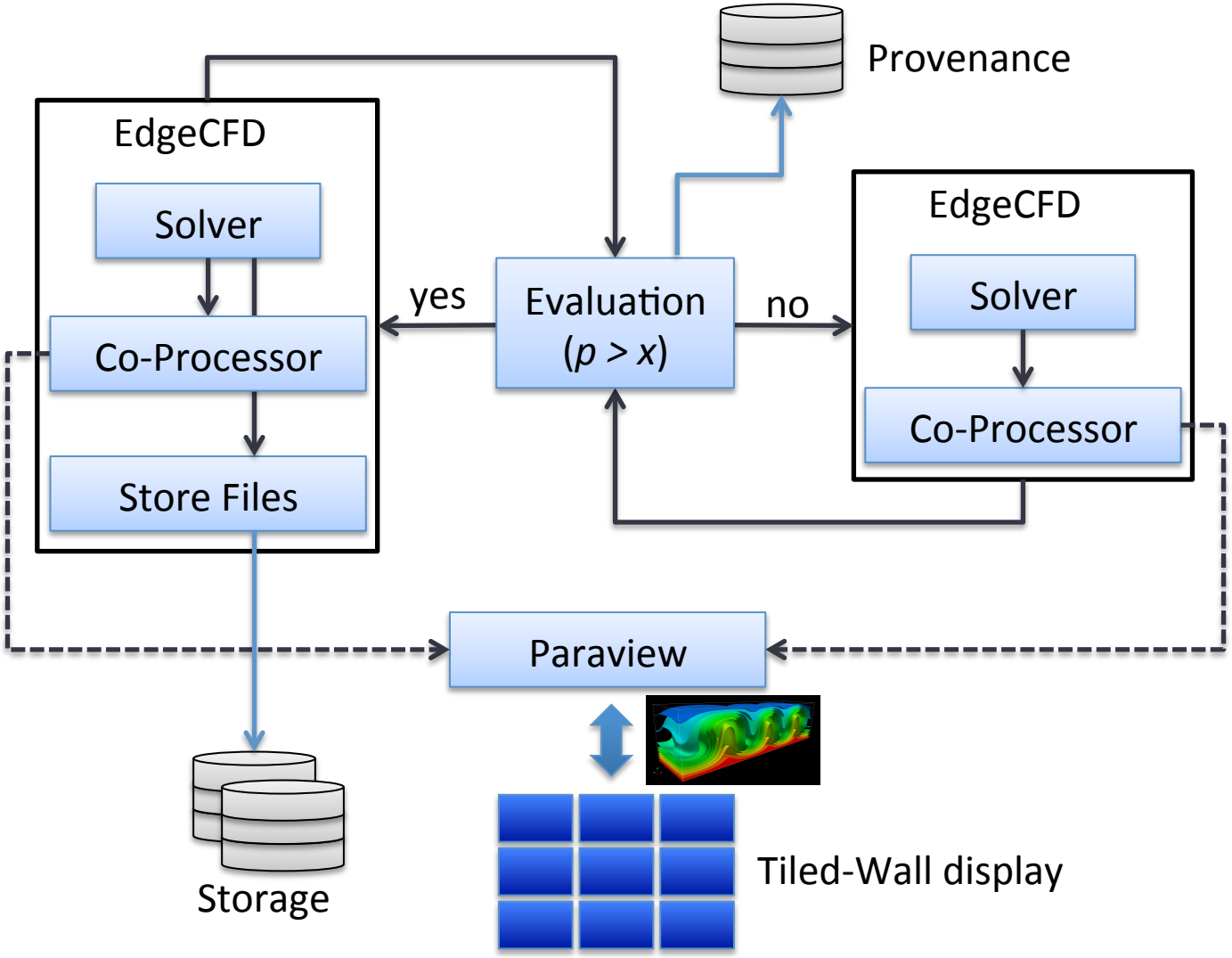
Figure 1: Different modes of visualization. In the traditional mode of visualization at left, the solver dumps all data to disk. Many *in situ* visualization projects couple the entire visualization within the solver and dump viewable images to disk, as shown in the middle. Although our coprocessing library supports this mode, we encourage the more versatile mode at right where the coprocessing extracts salient features and computes statistics on the data within the solver.

from disk storage is beginning to dominate the time spent in both the solver and the visualization [12].

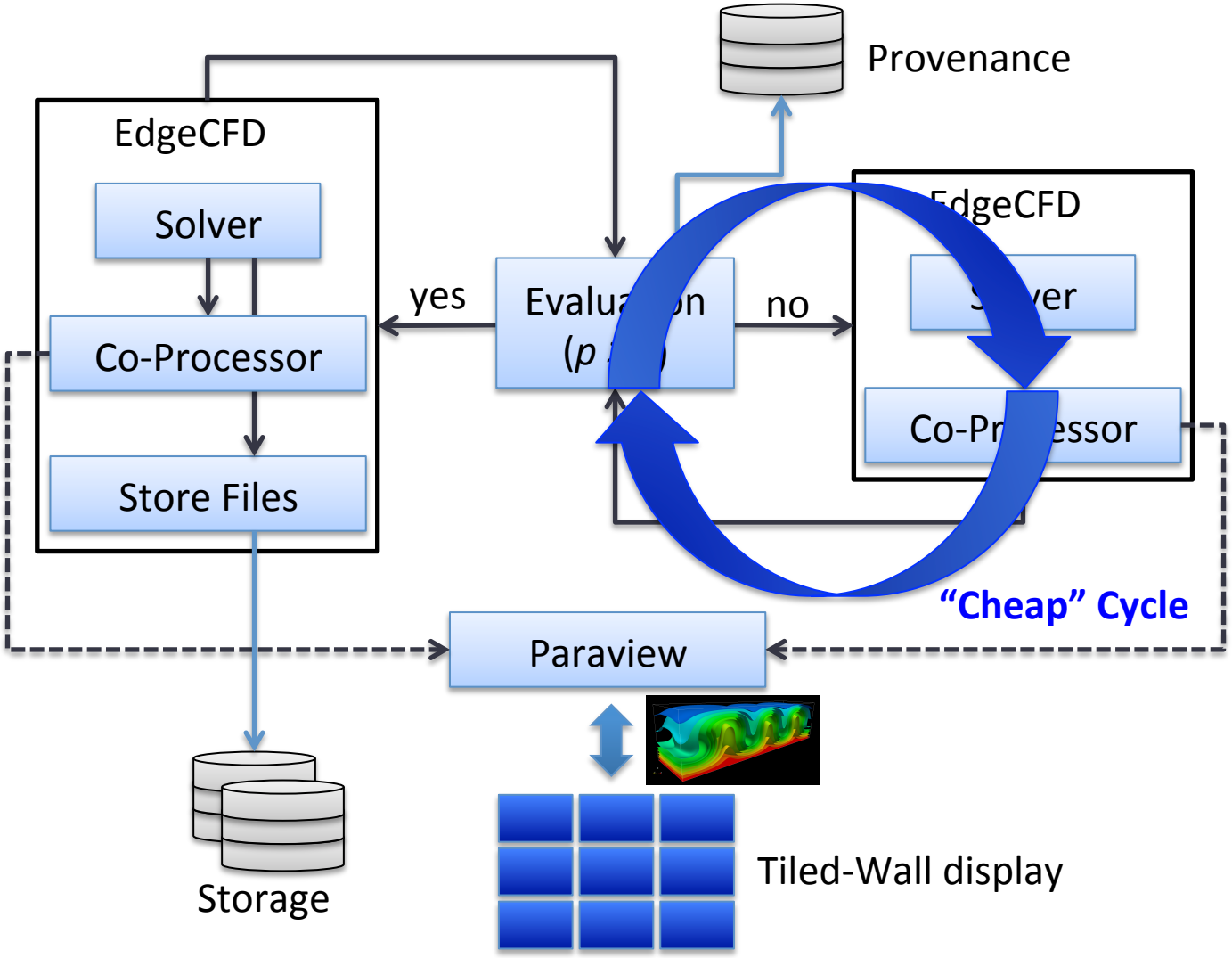
Coprocessing can be an effective tool for alleviating the overhead for disk storage [22], and studies show that visualization algorithms, including rendering, can often be run efficiently on today's supercomputers; the visualization requires only a fraction of the time required by the solver [24].

Whereas other previous work in visualization coprocessing completely couples the solver and visualization components, thereby creating a final visual representation, the coprocessing library provides a framework for the more general notion of salient data extraction. Rather than dump the raw data generated by the solver, in coprocessing we extract the information that is relevant for analysis, possibly transforming the data in the process. The extracted information has a small data representation which can be written at a much higher fidelity than the original data which in turn provides more information for analysis. This difference is demonstrated in Figure 1.

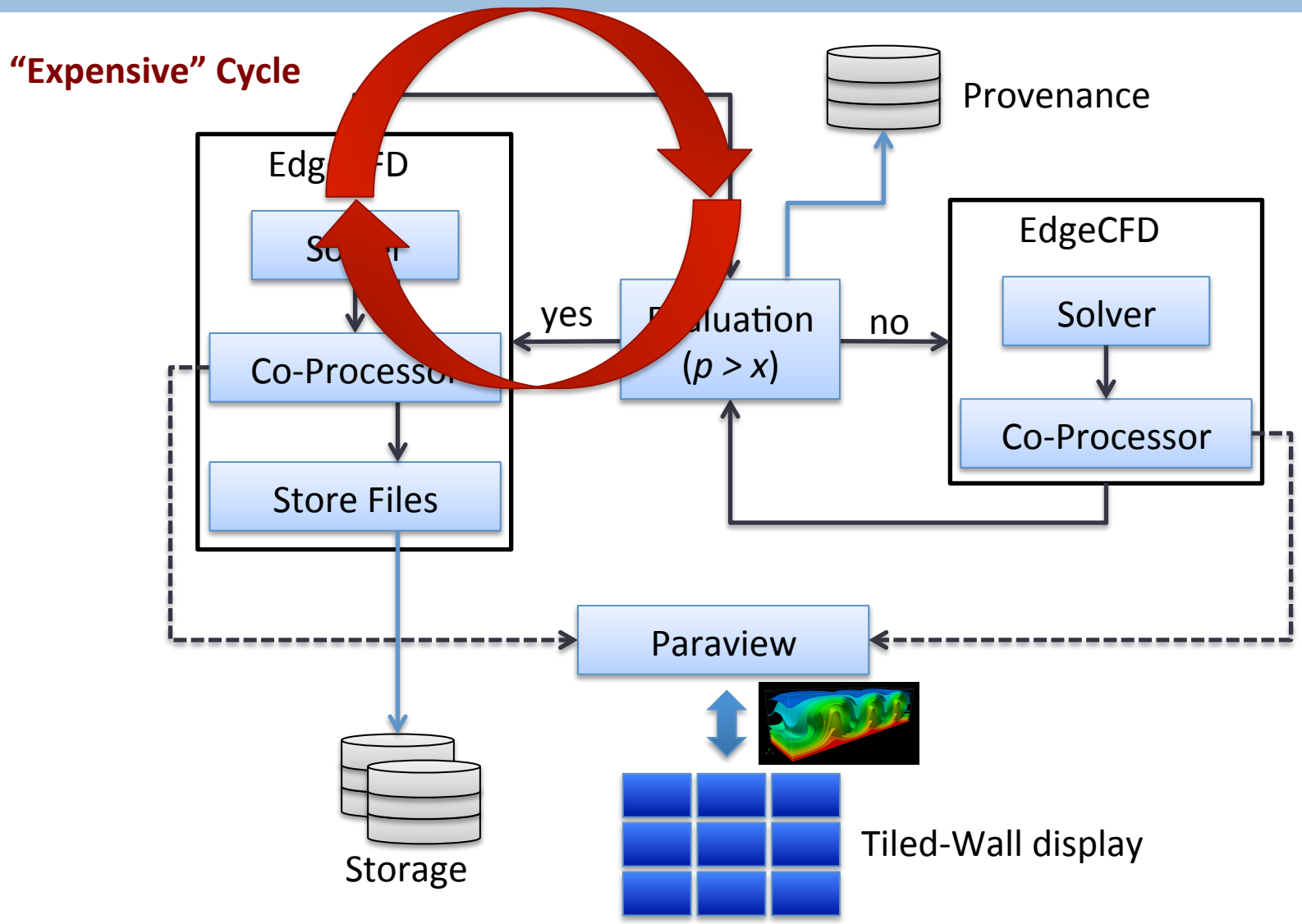
Evaluation Loops



Co-processing Workflow



Co-processing Workflow

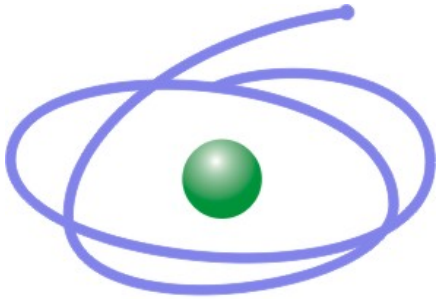


Conclusions

- Data centric experiments
  - *Example*: Large simulations with dense meshes
  - Parameter explorations
- Algebraic approach for scientific workflows
  - Algebraic and declarative workflow language
  - Allows for workflow optimizations
  - Runtime provenance analysis
- Interactive workflows
  - Fine-tuning adjustments during Wf execution
    - Real time provenance
  - Evaluation Loops
    - Iterative support
  - Exploring slices of parameter space
    - More dynamic analysis

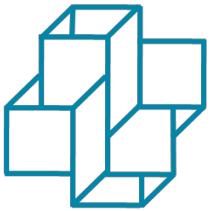
Acknowledgements

Conselho Nacional de Desenvolvimento Científico e Tecnológico



Federal University Rio de Janeiro

NACAD
High Performance Computing Center



Laboratório Nacional de Computação Científica

An Algebraic and Parallel Approach for Scientific Workflows using Chiron

First Brazil-France Workshop

On High Performance Computing and Scientific Data Management Driven by Highly Demanding Applications

Thanks!

Eduardo Ogasawara [2,1], Jonas Dias [1]

[1] Federal University of Rio de Janeiro, Brazil
[2] CEFET/RJ

Federal University Rio de Janeiro

CEFET/RJ