Part II

Designing Workflows
What Is In a Workflow
Similar but not Same:
What We Do **NOT** Mean by “Workflow”

- **Workflows in human organizations**, eg Process handbook [Malone 95]
  - Task dependencies and flow of information in human organizations

- **Workflows that represent process models**, eg PSL [Grunninger et al 05]
  - Graphs of activities and subactivities expressing complex temporal and resource constraints

- **Workflows for e-business and e-commerce transactions**
  - Interactions among processes that support negotiations, multi-party coordinated activities, etc.
Simple Examples of Workflows for Machine Learning Experiments

1. **Very specific, limited applicability**
   - **Weather-SM-2007-Data.csv** → **ID3-MODELER** → **Weather-SM-2007-Model.csv**
   - **Weather-PAS-2007-Data.csv** → **ID3-CLASSIFIER** → **Weather-PAS-2007-Predict.csv**
   - ClassIndex = 5

2. **Very general, very broad applicability**
   - **Training Data** → **ID3-MODELER** → **Model**
   - **Test Data** → **ID3-CLASSIFIER** → **Classification**

3. **Very general, very broad applicability**
   - **Training Data** → **DISCRETIZE** → **SAMPLE** → **BAYES-MODELER** → **Model**
   - **Test Data** → **BAYES-CLASSIFIER** → **Classification**

4. **General but customized to a specific use**
   - **Weather-SM-2007-Data.csv** → **SAMPLE** → **Weather-SM-2007-DataSample.csv**
   - **Sampling Interval = 20**
   - **ClassIndex = 5**
   - **Weather test data** → **ID3-MODELER** → **Weather-SM-2007-Model.csv**
   - **Weather prediction** → **ID3-CLASSIFIER**
Computational vs Service-Based Workflow

- Workflow consists of
  - Nodes: computations
  - Links: dataflow among nodes

### COMPUTATIONAL WORKFLOW
- Data stored in files
- Metadata catalogs of dataset properties
- Data catalogs of physical replicas
- Components catalogs of code locations and code execution requirements

### SERVICE-BASED WORKFLOW
- Message-passing paradigm for dataflow
- Service registries show availability
- Standing (3rd party) services
  - Computations managed by service provider
- Services are invoked remotely through potentially complex workflow orchestration
Designing Workflow Structure

- Complexity of application logic is ideally hidden within components
  - Components are designed and implemented by programmers
    - Full-fledged programming constructs
    - May be implemented in parallel languages

- Workflow logic is streamlined
  - Workflows are designed by non-programmers and should have simple structure
    - Focus is on selecting application components and data
Designing and Implementing Workflow Components

- **Workflow components as encapsulated software**
  - Declare I/O data
  - Declare parameters
  - Declare execution dependencies (e.g., runtime libraries)
  - Declare execution architecture requirements (e.g., OS, processor)

**Basic Component Encapsulation (BCE) Schema**

- **Workflow components as services**
  - Request and result messages

**Web Services Description Language (WSDL)**
Where to Start: A Workflow Sketch

- Computations as nodes
- Dataflow as links
  - Represent intermediate data
Terrible Workflow Sketches (I): The “I-Too-Can-Sketch” Workflow Sketch

User Specifies Parameters → Rough Image Analysis → Data Storage

Data Storage → Image Filtering → Detailed Image Analysis

Detailed Image Analysis → Results

Results
Terrible Workflow Sketches (II):
The “I-Am-Organized” Workflow Sketch
Terrible Workflow Sketches (II):
The “I-Am-Organized” Workflow Sketch
Terrible Workflow Sketches (III):
The “I-Am-In-Control” Workflow Sketch

Code1 → “go” → Code2 → “go” → Code3
Terrible Workflow Sketches (III):
The “I-Am-In-Control” Workflow Sketch

Code1 → “go” → Code2 → “go” → Code3
Terrible Workflow Sketches (IV): (Alternative Version of) The “I-Am-In-Control” Workflow Sketch
Terrible Workflow Sketches (V): (YAA Version of)
The “I-Am-In-Control” Workflow Sketch

Code1 → “go” → Code2 → “go” → Code3

http://www
Terrible Workflow Sketches (VI):
The “I-Sure-Can-Program” Workflow

<The workflow version of spaghetti code>
Terrible Workflow Sketches (VII): The “I-Am-User-Friendly” Workflow
Terrible Workflow Sketches (VIII): The “I-Support-Collaboration” Workflow
Terrible Workflow Sketches (IX):
A Fine but Baffling Workflow Sketch
Terrible Workflow Sketches (X): Any Suggestions?
Parallel Processing in Workflows

- Exploit parallel computations when possible
  - Iteration over datasets
Designing Workflows: An Example from Image Analysis
Microscopy Image Correction [Saltz et al 05]

- Microscope takes image tiles, one at a time, for all the slices
  - tens of GBytes uncompressed

- Image is corrected, then warped to map to standard brain, then pre-processed for efficient querying

- Correction workflow: original image has repeated areas across tiles, this needs to be corrected. Steps:
  1. Z-projection: picks the best pixel from all the z-slices.
  2. Normalization

**MPI**

3. Auto-alignment: piece together the image mosaic from the tiles
   - Uses a feature matching algorithm, output is "offsets", the alignment is given a score, goal is to maximize the score

**MPI**

4. Stitching: remove overlapping regions, so the tiles can "transition" more smoothly by finding best match across tiles.
   - Optimization algorithm picks the right offsets and stitches them together
[Sharma et al 07]:
- 3D image layers
- Chunked into tiles
- Projected into the Z plane
- Normalized and aligned
- Stitched back into a single 2D image

Tiles and parallelism are not explicit!
Some restructuring of the workflow that helped…
Controllable parameter: Chunk size??

Now we could add a control parameter... But which?
Controllable parameter: Chunk size??

Separated “normalize” from generation of 2 “magic” tiles
Format conversion

Chunk size

Z-projection

Generate magic tiles

Normalize

Auto-align

MST

Stitch

Controllable parameter: Chunk size??

Separated “auto-align” from MST
Now we could see the controllable parameter…
Format conversion

Chunksize

Z-projection

Generate magic tiles

Normalize

Auto-align

MST

Normalize

Stitch

Controllable parameter: $p$ is number of chunks in an image layer

Identified which should be parallel codes (MPI) because of communication needs
Identified steps that could be parallelized as independent nodes

Controllable parameter: \( p \) is number of chunks in an image layer
Now we could see that the controllable parameter should be the number of chunks.

Controllable parameter: \( p \) is number of chunks in an image layer.
Now we could see the workflow!!!
Workflow Template Sketch

**Controllable parameter:** p is number of chunks in an image layer

1. **Format conversion**
   - Set of z layers each with t tiles (.IMG)
   - Set of z layers each with t tiles (.DIM)

2. **Chunkisize**
   - Set of stacks of chunks
     - Collections of z layers, each has p chunks, each chunk has t tiles

3. **Z-projection**
   - Sets of chunks for z-proj
     - p (one per chunk stack)

4. **Generate magic tiles**
   - Internally iterates over each chunk of the z-proj
   - 2 magic tiles

5. **Normalize**
   - Sets of normalized chunks for z-proj
     - p (one per chunk of z-proj)

6. **Auto-align**
   - Internally iterates over each chunk of the normalized z-proj
   - Scores per pairs of chunks in z-proj

7. **MST**
   - Graph

8. **Normalize**
   - Sets of normalized chunks for all layers
     - p * z (one per chunk per layer)

9. **Stitch**
   - Internally iterates over each chunk of each layer
   - Sets of stitched norm. chunks for all layers

**LEGEND**
- Regular codes
- Parallel codes (MPI) with one computation node
- Will be expanded to several nodes

**Set of z layers each with t tiles (.IMG)**
**Set of z layers each with t tiles (.DIM)**

**Sets of stacks of chunks**
- Collections of z layers, each has p chunks, each chunk has t tiles

**Sets of chunks for z-proj**
- p (one per chunk stack)

**Sets of normalized chunks for z-proj**
- Internally iterates over each chunk of the normalized z-proj
- Scores per pairs of chunks in z-proj

**Sets of normalized chunks for all layers**
- p * z (one per chunk per layer)

**Sets of stitched norm. chunks for all layers**
- Internally iterates over each chunk of each layer
Format conversion

Chunkisize

Z-projection

Generate magic tiles

Normalize

Auto-align

MST

Normalize

Stitch

Controllable parameter: p is number of chunks in an image layer

Sketch of Workflow Template

Legend:
- Regular codes
- Parallel codes (MPI) with one computation node
- Will be expanded to several computation nodes
Sketch of computations in the workflow:

- **Format conversion**
- **Chunkisize**
- **Z-proj**, **Z-proj**, ..., **Z-proj**
- **Generate magic tiles**
  - Internally iterates over each chunk of the z-proj
  - **Norm**, **Norm**, ..., **Norm**
    - **Auto-align**
      - Internally iterates over each chunk of the normalized z-proj
    - **MST**
- **Norm**, **Norm**, ..., **Norm**
  - Internally iterates over each chunk of each layer
- **Stitch**
  - Internally iterates over each chunk of each layer

Controllable parameter: $p$ is number of chunks in an image layer

**LEGEND**
- Regular codes
- Parallel codes (MPI) with one computation node
- Computation nodes
Summary
Blocks and Arrows A Workflow Do Not Make: A Well-Formed Workflow Sketch

- Components are models of software to be executed
- Components have all I/O data exposed
- All important parameter settings in components are exposed
  - So workflow system can reflect settings in provenance records
- Links reflect ALL data flow
  - Each dataset is produced by one component, can be consumed by several
- No side effects
  - No global context
Abstraction  Layers in Computational Workflows
Numerous *Interdependent* Decisions for Workflow Creation

- Many types/variants/implementations of application components, each with different storage and computational requirements
- Many alternative data collections with different degrees of pre-processing
- Many possible data replicas in the distributed environment
- Many possible resources to execute application components
Workflow Complexity

 Workflow complexity may be reflected in two major dimensions:

• **Scale**: measured by
  – amount of processes
  – compute times
  – data sizes
  – heterogeneity of code execution requirements

• **Scope**: measured by
  – Constraints on code I/O requirements
  – Constraints on data properties
Creation of Workflows in Layers of Increasing Detail

1. **Workflow Template** *(generic known-to-work recipes)*
   - Specifies application components and dataflow among them
   - No data specified, just their type

2. **Workflow Instance** *(data-specific)*
   - Specifies data files for a given template
   - Logical file names, not physical file replicas

3. **Executable Workflow** *(actual run)*
   - Specifies physical locations of data files (may be in data repositories)
   - Assigned hosts/pools for execution of each component
   - Includes data movements among execution sites and data repositories as well as data deposition steps
Creation of Workflows in Layers of Increasing Detail

1. **Workflow Template** *(generic known-to-work recipes)*
   - Specifies application components and dataflow among them
   - No data specified, just their type

2. **Workflow Instance** *(data-specific)*
   - Specifies data files for a given template
   - Logical file names, not physical file replicas

3. **Executable Workflow** *(actual run)*
   - Specifies physical locations of data files (may be in data repositories)
   - Assigned hosts/pools for execution of each component
   - Includes data movements among execution sites and data repositories as well as data deposition steps

---

*User guided*

*Automated*
Workflow Templates: No Data or Execution Information

Corpus

Split

Kernel_Rules

Filter_Rules

Prune_Rules

Binarize

Generate_Rule_Map

Compile

XRS_Rules

BRF_Rules

Lexicon_Dictionary

Workflow Templates: No Data or Execution Information
Workflow Instance: Compact Expression

Corpus ➔ WSJ-2001

Split ➔ Kernel_Rules ➔ KR-09-05

1…n

Filter_Rules ➔ Prune_Rules

Binarize ➔ Generate_Rule_Map

1…n

Compile ➔ XRS_Rules ➔ BRF_Rules ➔ Lexicon_Dictionary
Workflow Instance: Expanded Expression
Workflow Instance Expressions

- Compact expression for efficient search and matching
- Expanded expression when further details are needed

1...n
Executable Workflow: Maps Expanded Instance to Execution Resources
Workflow Creation then Workflow Mapping

**Workflow Creation Functions**

- Workflow formulation
- Workflow discovery
- Workflow elaboration

**Workflow Assembly Functions**

- Workflow assembly
- Workflow validation
- Data/parameter selection

**Workflow Completion Functions**

- Workflow completion
- Workflow sharing
- Metadata generation

---

**Workflow Mapping and Execution Functions**

- Site selection
- Replica selection
- Workflow submission

**Workflow Optimization Functions**

- Failure recovery
- Data management
- Workflow optimization