Ediflow: data-intensive interactive workflows for visual analytics

Véronique Benzaken\textsuperscript{2}, Jean-Daniel Fekete\textsuperscript{1}, Pierre-Luc Hémery\textsuperscript{1}, Wael Khemiri\textsuperscript{1,2} and Ioana Manolescu\textsuperscript{1,2}

\textsuperscript{1}INRIA Saclay – île de France
\textsuperscript{2}LRI, Université Paris Sud

ICDE 2011
Outline

- Motivation
- Ediflow architecture
- Isolation management
- Use cases
- Robustness evaluation
- Conclusion and perspectives
Motivation – Visual analytics field

“Visual analytics is especially focused on situations where the huge amount of data and the complexity of the problem make automatic reasoning impossible without human interaction”

Current visual analytics tools have some drawbacks:

- Scalability issues
- No multi-user environment
- Data cannot be shared and reused
Scientific workflows vs. visual analytics

Scientific workflow systems share many characteristics with visual analytics

- Complex analysis tasks backed by persistent storage

Differently from visual analytics platforms, scientific workflows:

- are designed to carry automated analytical processes to completion
- do not manage dynamic data
- offer little or no visualization

Our goals:
1) Integrating scientific workflows with visual analytics
2) Managing dynamic data
Ediflow architecture
Process model

Core process model:
- Structured processes
- Workflow management coalition model
  - Sequence
  - OR split, OR join
  - AND split, AND join
  - IF-THEN
  - Procedure

Extension: reactive processes
- Reactive: propagate changes between the database and the running workflow instances through the process
Process model zooms

- Computation unit
- Black box developed outside the DB engine (Java, C++, Matlab etc.)
- E.g. clustering algorithms, statistical analysis tools

Procedure

- Helper procedures used to reflect the impact of data changes on process execution
- Ediflow recuperates the result of handler invocation and injects it into the process
- The implementation of handlers is opaque to the process execution framework

Delta handler
Reactive process model

Update propagation in reactive processes:

- Ignore $\Delta R$ for the execution of all processes which had started executing before $t_{\Delta R}$

- Ignore $\Delta R$ for the execution of all activities which had started executing before $t_{\Delta R}$

- $\Delta R$ are propagated to instances of all activities that are yet to be started in a running process

- Propagate the update $\Delta R$ to all the terminated instances of a given activity

- Propagate the update $\Delta R$ to all the running instances of a given activity, whether they had started before $t_{\Delta R}$ or not
Isolation management

- Process- and activity-based isolation

P1

\[ a_1 \rightarrow a_2 \rightarrow a_3 \rightarrow \text{black circle} \]

P2

\[ a_4 \rightarrow a_5 \rightarrow a_6 \rightarrow \text{black circle} \]
Isolation management

Time-based isolation

- Data visible to a given activity or process instance may depend on the starting time of that instance

Associating to each application table R a creation timestamp

Problem with tuple deletion:

- Tuples are not actually deleted from R until the end of the process execution
- Tuples are added to a deletion table R (tid, tdel, pid, __)
- Rewriting queries
Use case 1: WikiReactive scenario

**Goal:** Proposing to Wikipedia readers and contributors some measures of the history of an article

Compute the differences between successive versions of each article

For each user, maintain the total number of characters added, deleted and moved

Compute the contribution table storing the identifier of the user who entered it

Compute the number of distinct contributors

Maintain the total number of characters
Use case 2: publication cleaning scenario

**Goal:** Detecting and helping remove duplicated author entries in a large database of publications.

- Compute the similarities between the inserted author and all the other already in the table
- Show the results of similarities and co-publications graph of an author
- Allow the user to decide whether two authors are identical or not
Visualization views management

- Ediflow can maintain several visualization views for one visualization.
Visualization views management

Benefits of this architecture:
- It allows sharing visual attributes by several views
- The computation of visual attributes is done only once
- In line with visual analytics recommended software architecture

Example of co-publications graph in the WILD:
- A cluster of 16 machines to display the graph over 32 screens
- Each machine controls two screens
- Each machine runs an Ediflow instance
Visualization views management
Ediflow tool implementation

- Implemented in Java
- On top of Oracle 11g DBMS
- Procedures: Java modules in OSGi Service Platform
- A procedure instance is a concrete class implementing the Ediflow Process interface
- Ediflow process requires four methods:
  - initialize()
  - run(ProcessEnv env)
  - update(ProcessEnv env)
  - string getName()
Robustness evaluation

**Goal:** Study how the Ediflow event processing chain scales when confronted with changes in the data

The DBMS is connected via 100 MHz Ethernet connection to two Ediflow instances running on two machines

- The first Ediflow machine computes visual attributes (runs the layout procedures)
- The second machine extracts nodes from VisualAttributes table and displays the graph

Adding increasing numbers of tuples to the database
Robustness evaluation

Inserting tuples requires performing a sequence of steps:

- Parse the message involved after insertion in nodes table
- Insert the resulting tuples in the VisualAttributes table
- Parse the message involved after the insertion in the VisualAttributes table
- Extract the visual attributes of the new nodes
- Insert new nodes into the display screen of the second machine
Robustness evaluation

- Inserting tuples in VisualAttributes table
- Inserting new nodes into the display
- Message parsing (change in Author table)
- Message parsing (change in VisualAttributes table)
- Extracting new nodes from VisualAttributes (select)
- Total time

Time (in ms) vs. # inserted tuples
Robustness evaluation

Results:

- The times are compatible with the requirements of interaction
- They grow linearly with the size of inserted tuples
- The dominating time is required to write in the VisualAttributes table

The price to pay for having these attributes stored in a persistent database
Summary

- Design and implementation of Ediflow
- Workflow platform supporting visual analytics
- Ediflow unifies the data model used by all its components
- Supports standard data manipulation through procedures
- Reflects changes in the data through update propagations
- Several options are offered to react to such changes
Perspectives

- Improve the visual table schema
- Specify a collaboration management mechanisms
- Integration with current scientific workflow systems (Vistrail, Kepler, etc)
Thank you.
References

C. Scheidegger and T. VoHuy and D. Koop and J. Reire and C. T. Silva. **Querying and re-using workflows with VsTrails.**
SIGMOD 2008

SSDBM 2004.

I. Zachary and G. Todd and K. Grigoris and T. Nicholas and T. Val and T. Partha Pratim and J. Marie and P. Fernando
**The ORCHESTRA Collaborative Data Sharing System.**
SIGMOD 2008
Recent and well-developed scientific workflow project

Helps scientists and analysts to create, execute and share models

Provides a GUI to create scientific workflow

No mechanism to handle dynamic data

Visualization remains external
The Orchestra platform

Data-centric P2P platform for scientific applications

Dedicated to bioinformatics

Focuses on data exchange mapping across different sources

Each peer's DB is updated to reflect updates in the other peers

No visualization

No interactivity

Leaves out external computations
The Vistrail platform

Combines features of workflow systems and visual analytics

Manages exploratory activities

Iteratively refines computational tasks

Maintains detailed provenance of the exploration process

No support for data dynamicity
Use case 1: election scenario

**Goal:** Monitoring the results of the American presidential election

**US Elections workflow**

1. **Vote reporting**
2. **Aggregate calculation**
3. **Update visualization**

Retrieve the results of votes and update the database

Compute and store the number of votes of each party in each state in an aggregated table

Update the visualization and the view to reflect the new votes
Process model – Structured process

Process ::= Configuration Constant* Variable+ Relation+ Procedure* StructProcess
Configuration ::= DBdriver DBuri DBuser
Constant ::= name value name ∈ N, value ∈ V
Variable ::= name type name ∈ N, type ∈ T
Relation ::= name primaryKey RelType
RelationType ::= (attName attType)* attName ∈ N, attType ∈ T
Procedure ::= name classPath
StructuredProcess ::= Activity | Sequence | AndSplitJoin | OrSplitJoin | ConditionalProcess
Sequence ::= Activity, StructuredProcess
AndSplitJoin ::= AND-split (StructuredProcess)+ AND-join
OrSplitJoin ::= OR-split (StructuredProcess)+ OR-join
ConditionalProcess ::= IF Condition StructuredProcess
Activity ::= activityName Expression
Expression ::= askUser | callProcedure | runQuery