Master's Thesis Defense

Development of a Data Management Architecture for the Support of Collaborative Computational Biology

Lance Feagan
Acknowledgements

- Ed Komp—My mentor.
- Terry Clark—My advisor.
- Victor Frost—My “other” advisor.
- Heather, Jesse, Justin, Keith, Andrew—Project team members.
Presentation Overview

• Computational biology background
• Analysis of similar projects
• Design Objectives
• Implementation
• Conclusions
Computational Biology

- Small & Large data sets
  - Input: protein structure, genetic sequence
  - Output: MD simulation, BLAST

- Exponential data growth
  - New data becoming available at impressive rate
  - Curated vs. non-curated data

- Experiments can be represented as pipelines, possibly with feedback to earlier stages

- Runs frequently re-done
  - Parameter variation and re-analysis of results
  - Use of recently available data with previously used experimental configuration
Computational Biology

• Maintenance of provenance records required when publishing results
  – Challenging in fast-paced, high-volume environment

• Collaboration important

• Controlled information access levels

• Wet-lab techniques do not scale well

• Tools and formats used in analysis do not inherently provide integrated, end-to-end provenance trail
Why is provenance important?

- Provides context and specifications work was done in

- CS pioneer Jim Gray on provenance:

  “scientific data [to] remain available forever so that other scientists can reproduce the results and do new science with the data…To understand the data, those later users need the *metadata:* (1) how the instruments were designed and built; (2) when, where, and how the data was gathered; and (3) a careful description of the processing steps that led to the derived data products that are typically used for scientific data analysis.” [Gray02]
Provenance Taxonomy

Y.L. Simmhan, B. Plale, and D. Gannon
“A Survey of Data Provenance in e-Science”
Related Work

- NoteCards
  - Xerox D, LISP machine
  - Semantic network of related notes
  - Tree-like graphical browse/manipulation tool
  - Semantic network entirely user-maintained
    - Flexible, but very tedious and time-consuming
  - Search was limited to the title and content of a NoteFile
- Virtual Notebook Environment (ViNE)
- BioCoRE
- myGrid/Taverna
- Scientific Annotation Middleware (SAM)
Related Work: Shortcomings

- Limited or no provenance or other meta-data management
- Concept of workflow that other users can view, import, and alter
- Collaborative features don't include managed re-use that maintains provenance trail
- Weak search capabilities result in overly cluttered user interfaces
  - Search can be used as a filter to select visible data plane in navigation tools
- Integration of secure computational facility while keeping user's rights and restrictions on usage intact
Design Objectives

- Derived from analysis of related projects
- Identified five key areas as vital in aiding computational biologists
  1) Type Hierarchy & Data Abstraction
  2) Data Storage
  3) Provenance Management
  4) Data Security
  5) Data & Provenance Search
Type Hierarchy & Data Abstraction
Type Hierarchy & Data Abstraction

• Type Hierarchy: Design Goals
  – Provide core type definitions that could be used pervasively as basis
  – Extensible by users as well as administrators
  – Allow grouping of types for “namespaces”

• Data Abstraction: Design Goals
  – Allow interaction in generic fashion
  – Enable extensive search and computational usage capability
  – Maintain or improve performance
Data Model: Entry

- Electronic journal concept as basis
- Page <=> Entry
- Visible journal entry comprised of a single top-level entry (node) with arbitrary number of related nodes
- Entry (node) is an atomic unit of information associated with a single type that may be re-used
- Entry data structure composed of meta-data + content
Data Model: Entry

1) **EntryId** (INTEGER): PK for Entry table. Positive integers for all non-core entries. 0 reserved to prevent infinite recursion of type relationships.

2) **UserId** (INTEGER): Entry creator/owner. Joined with other tables to define permissions and search for entries.

3) **ContentTypeId** (INTEGER): References by **EntryId** the type of the information stored in this entry.

4) **Title** (VARCHAR2): Used to assign a title to an entry. *Titles* are not unique.

5) **JournalId** (INTEGER): References by **EntryId** another entry, which must be of content type journal. All journals are attached to a single root journal.
Data Model: Journal

- Collection of entries
- Entry may only be a member of a single journal
- Journals may be nested
- Multiple writers to a single journal
  - Furthers collaboration
Data Abstraction

• Achieved through:
  – Extensible type hierarchy
  – Plug-in architecture

• Dissociate content from meta-data
  – High-performance
  – Pervasive, comprehensive search capability
  – Attribution provenance

• Content type & content storage type
Type Hierarchy

- Extensibility: Addition of new types
  - By both administrators and users
  - Aids flexibility and re-use
- Flexibility: Changes to hierarchy should not break existing infrastructure
- Robustness: Operations that compromise stability are prevented
- Data-driven, plug-in architecture reduces coupling
- No nested types: simplifies processing of an entry
- Type hierarchy stored as a collection of entries, with relationship to special “master type” entry
Type Hierarchy: Design

<table>
<thead>
<tr>
<th>ENTRYID</th>
<th>USERID</th>
<th>CONTENTTYPEID</th>
<th>TITLE</th>
<th>JOURNALID</th>
</tr>
</thead>
<tbody>
<tr>
<td>-100</td>
<td>CJ</td>
<td>-100</td>
<td>&quot;MASTER_CONTENT_TYPE&quot;</td>
<td>0</td>
</tr>
<tr>
<td>-101</td>
<td>CJ</td>
<td>-100</td>
<td>&quot;MASTER_JOURNAL_TYPE&quot;</td>
<td>0</td>
</tr>
<tr>
<td>-102</td>
<td>CJ</td>
<td>-100</td>
<td>&quot;MASTER_ENTRY_RELATION_TYPE&quot;</td>
<td>0</td>
</tr>
<tr>
<td>-103</td>
<td>CJ</td>
<td>-101</td>
<td>&quot;ROOT_JOURNAL&quot;</td>
<td>0</td>
</tr>
</tbody>
</table>

Core Types

- **MASTER_CONTENT_TYPE**
  - EID=-100
  - CTID=-100
  - JID=0

- **MASTER_JOURNAL_TYPE**
  - EID=-101
  - CTID=-100
  - JID=0

- **ROOT_JOURNAL**
  - EID=-103
  - CTID=-101
  - JID=0

- **System Journal**
  - EID=1
  - CTID=-101
  - JID=-103

- **Annotation**
  - EID=2
  - CTID=-102
  - JID=1
Storage Architecture
Storage Architecture

- Objectives:
  - Rapid textual & meta-data search
  - High-performance (latency and B/W) cluster-based computing
- Split “atomic” entry into meta-data and content-data
- Hybrid content-data storage: in DBMS or in cluster file system

<table>
<thead>
<tr>
<th></th>
<th>Latency</th>
<th>Volume</th>
<th>Searchable</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLOB</td>
<td>Low</td>
<td>Low</td>
<td>Yes</td>
</tr>
<tr>
<td>BLOB</td>
<td>Low</td>
<td>Low</td>
<td>No</td>
</tr>
<tr>
<td>BFILE</td>
<td>High</td>
<td>High</td>
<td>No</td>
</tr>
</tbody>
</table>
Provenance
Provenance

1) Attribution
   1) Owner-only write model + link creation on copy + security model ensures attribution is maintained

2) Audit Trail

3) Data Quality

4) Informational

5) Replication
Provenance: Entry Fields

- Five Fields
  - UserId
  - CreateDate
  - ModifyDate
  - CommitDate
  - Committed
Provenance: Semantic Relationships

- RDF 3-tuple: <subject, predicate, object>
- Predicate defines an “IS-A” relationship
- <'Junior', 'son', 'Senior'>
  - “Junior IS A son OF Senior”
- Choice to require “IS-A” eliminates need for inverse relationship searches on “HAS-A”
- \textit{FromEntry IS A LinkType OF ToEntry}
Provenance: Semantic Relationship Integrity Constraint

- Second stage in ensuring durability of provenance (first is secured commit)
- Extension of Resource Description Framework (RDF) 3-tuple of \langle subject, predicate, object \rangle
- Depending on predicate, removal of subject or object may nullify ability to maintain definitive provenance trail
- Form 5-tuple with addition of two values describing subject's dependence on object and vice-a-versa
- Assurances of non-removal of content protected by SRIC encourages collaborative sharing and re-use of experiments & data from other users
Provenance: SRIC

- Flags:
  - ToRequiresFrom
  - FromRequiresTo

- Commit vs. SRIC – Describe how this plays out.

<table>
<thead>
<tr>
<th>ToRequiresFrom</th>
<th>FromRequiresTo</th>
<th>Valid?</th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>False</td>
<td>Yes</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>Yes</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>Yes</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>No</td>
</tr>
</tbody>
</table>
Provenance: Collection & Usage

- Two collection techniques:
  - Automatic: Done via DB triggers, Ex: create, update, commit of an entry
  - API: Done via BCJ plug-ins, Ex: input to and output from experiment, textual annotations

- Used by navigators and tools
  - Provide relevant information while reducing clutter
  - Filtering by author, create/modify/commit date, deprecated flag, rating, hidden, categorization by content types
  - Related entry view in navigator

- Open interface via views rather than purely functional interface
  - Allows complex join queries to be performed
Provenance: Conclusions

● Automated provenance management framework vital

● Eliminates burdensome folder-file management scheme to maintain association between input, experiment, and output

● Ensures attribution information is maintained from inception through multiple iterations and ending in the successful conclusion of work

● Non-repudiation assured

● Extended RDF 3-tuple encourages sharing and re-use among collaborators
Search
Search

- Pervasive: Fundamental activity
- Data and provenance
- Anything that user can access should be query-able
  - Content
  - Meta-Data
  - Combined content + meta-data query
- Used by navigators to define and refine presentation
- Used by workflow and experiment editors to find available and appropriate blocks and resources
- Sorting
Search: “Helper” Fields

• Four additional meta-data fields to assist search
  - `ContentLastAccessDate`
  - `Deprecated`
  - `Hidden`
  - `Rating`
Security

- LINK
  - CJ.LINK
  - CJ.ENTRY
    - metadata
    - CJ.CONTENT_BLOB
    - CJ.CONTENT_CLOB
    - CJ.CONTENT_BFILE
    - permissions
      - CJ.ENTRYGROUPS
- ENTRY
  - content
    - CJ.CONTENT
    - CJ.CONTENT_CLOB
    - CJ.CONTENT_BFILE
- GROUPS
  - CJ.GROUPS
    - CJ.USERS_GROUPS
- USERS
  - CJ.USERS
Security

- Coarse- vs. fine-grained
- Permission inheritance
- Goals
  - Create secure, private workspace for individuals
  - Encourage collaboration through the ability to selectively share information via find-grained access controls
  - Maintain the integrity of all provenance information collected
Security

- Actors: Users and Groups
- Group-based access control
  - Each user a member of one or more groups
  - Each group contains zero or more users
  - Each entry associated with one or more groups granted read-only access
- Read/Write Permission Levels
  - Owner-only write permission
Security

- “Commit” concept
  - Prior to commit, owner free to alter content
  - Committing entry locks content to prevent further writing
  - Meta-data fields affecting provenance also locked
  - First stage in ensuring durability of provenance (second step is SRIC)
Security: Metadata

- Two categories of metadata field access levels:
  - System-controlled, and
  - User-controlled
- Triggers used on entry metadata fields rather than functional interface
  - Maximize available information for search
  - hasEssentialDependents
- Delete compromise of strict rules

<table>
<thead>
<tr>
<th>System-Controlled</th>
<th>Fixed at Creation</th>
<th>Fixed at Commit</th>
</tr>
</thead>
<tbody>
<tr>
<td>EntryId</td>
<td>Yes</td>
<td>X</td>
</tr>
<tr>
<td>UserId</td>
<td>Yes</td>
<td>X</td>
</tr>
<tr>
<td>CreateDate</td>
<td>Yes</td>
<td>X</td>
</tr>
<tr>
<td>ModifyDate</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>CommitDate</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>ContentLastAccessDate</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>User-Controlled</th>
<th>Fixed at Creation</th>
<th>Fixed at Commit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Committed</td>
<td>No</td>
<td>Yes*</td>
</tr>
<tr>
<td>ContentType</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Deprecated</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Hidden</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>JournalId</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Rating</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>SystemGenerated</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Title</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
Security: BFILE

• Separate cluster FS created
  - Can only be accessed by 'oracle' and special user to submit PBS jobs
• Mode bits altered when an entry is (un)committed so that FS reflects meta-data
  - Uses triggers to Java Stored Procedures (JSP)
  - UNIX permissions of 660 and 440
  - Immediate deletion
Summary & Conclusions
Summary

- BCJ represents a comprehensive solution
- Areas addressed
  1) Type Hierarchy & Data Abstraction
  2) Data Storage
  3) Provenance Management
  4) Data Security
  5) Data & Provenance Search
Summary

• Type Hierarchy
  – Four core entries, three master types
  – Data decomposition: atomic unit of information: the entry
  – Each entry associated with a single type
  – Not a hierarchy, single master type, eases addition of new types
  – Journal as organizational structure

• Provenance
  – All five types of provenance can be collected
  – Collection through automated means (triggers or API) along with user-friendly tooling eliminates manual processes
Summary

- **Storage**
  - Global, shared workspace provides area for collaboration

- **Security**
  - Flexible, efficient, tools simplify use, enables powerful search and cluster-based computing while ensuring integrity of provenance
  - SRIC extension of RDF 5-tuple
Conclusions

- Crafting a comprehensive solution is not only time consuming but also challenging.
- End-product is vastly superior at lowering level of effort necessary to produce relevant biological research while maintaining vital provenance information.
- Pervasive use of search & filters simplifies access to basic information while retaining complex search capabilities.
Future Work

• SRIC
  – Link deletion strategy
  – Split ownership of link end-points
  – Owner of (very large) content linked-to by another user wants to delete the content

• Deployment needs to be simplified

• SRIC concept could be made extensible through the *LinkType Entry*

• Integrate SRIC into DBMS
Questions

- Wow! We made it to the end.
- Contact: lfeagan@us.ibm.com