Digital Library Grid Initiatives: Cheshire3 and the Grid

Ray R. Larson

University of California, Berkeley

School of Information Management and Systems

Rob Sanderson

University of Liverpool

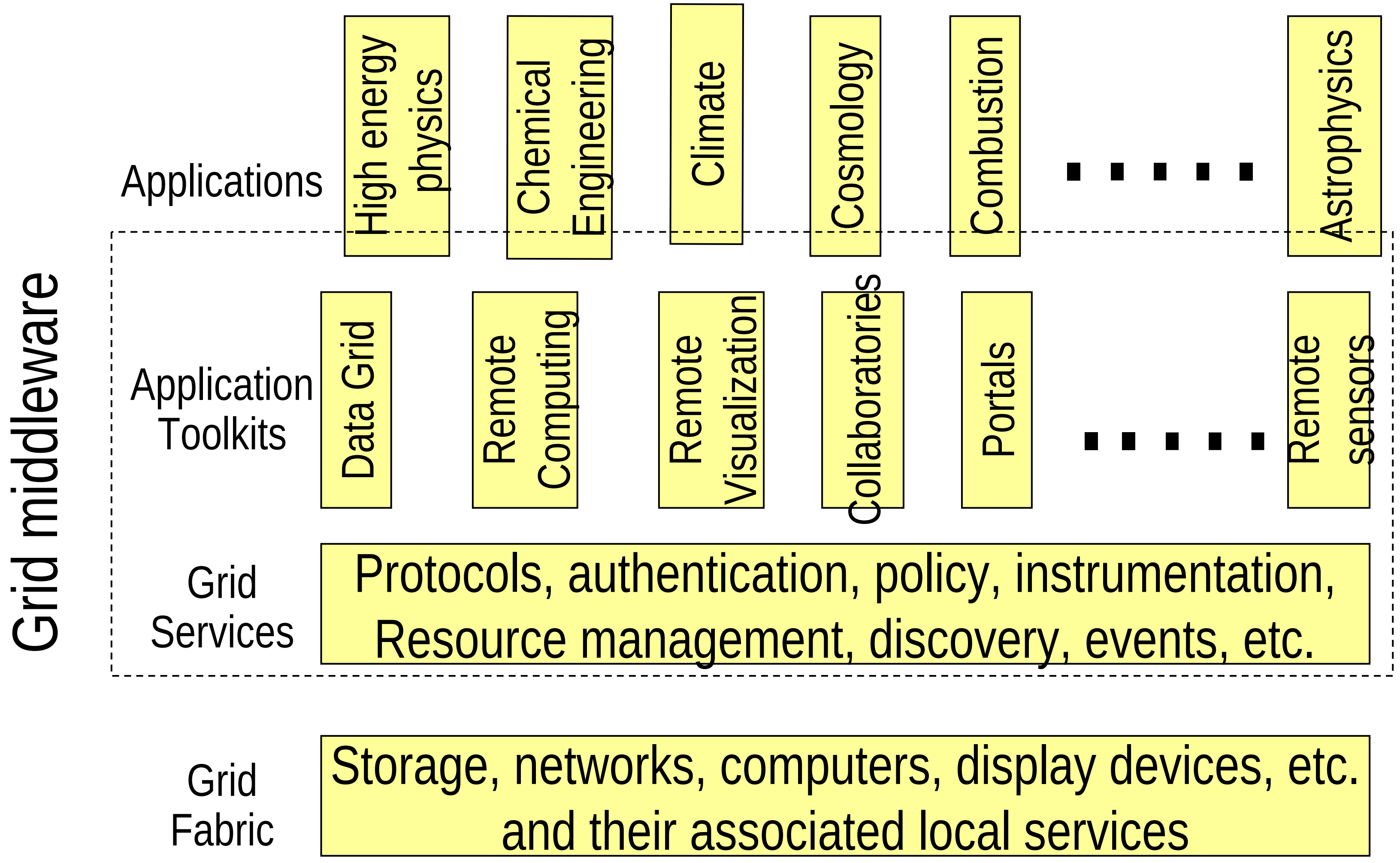
Dept. of Computer Science

Thanks to Dr. Eric Yen and Prof. Michael Buckland for parts of this presentation

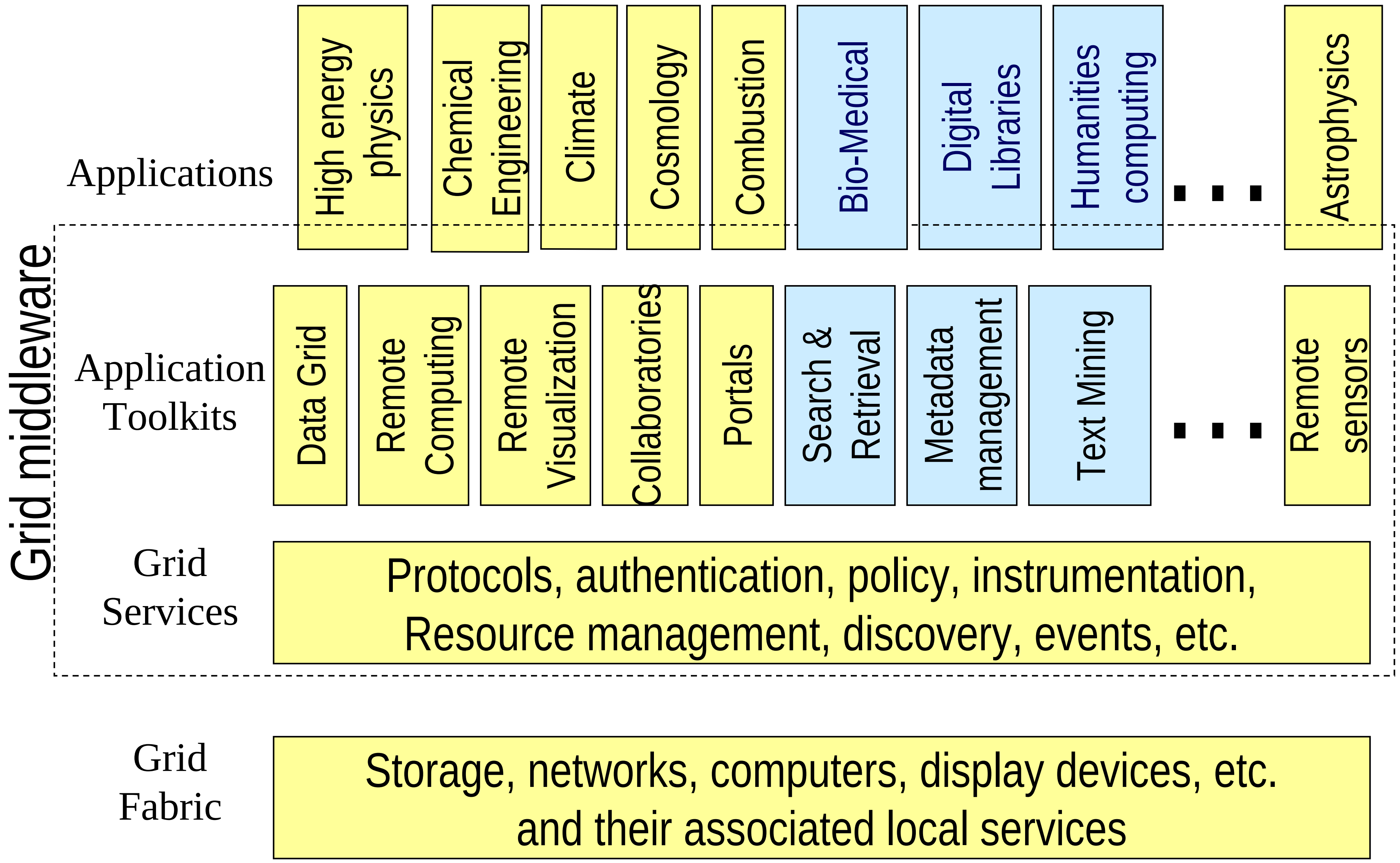
Overview

- The Grid, Text Mining and Digital Libraries
  - Grid Architecture
  - Grid IR Issues
- Cheshire3: Bringing Search to Grid-Based Digital Libraries
  - Overview
  - Grid Experiments
  - Cheshire3 Architecture
  - Distributed Workflows

Grid Architecture -- (Dr. Eric Yen, Academia Sinica, Taiwan.)



Grid Architecture (ECAI/AS Grid Digital Library Workshop)



Grid-Based Digital Libraries

- Large-scale distributed storage requirements and technologies
- Organizing distributed digital collections
- Shared Metadata – standards and requirements
- Managing distributed digital collections
- Security and access control
- Collection Replication and backup
- **Distributed Information Retrieval issues and algorithms**

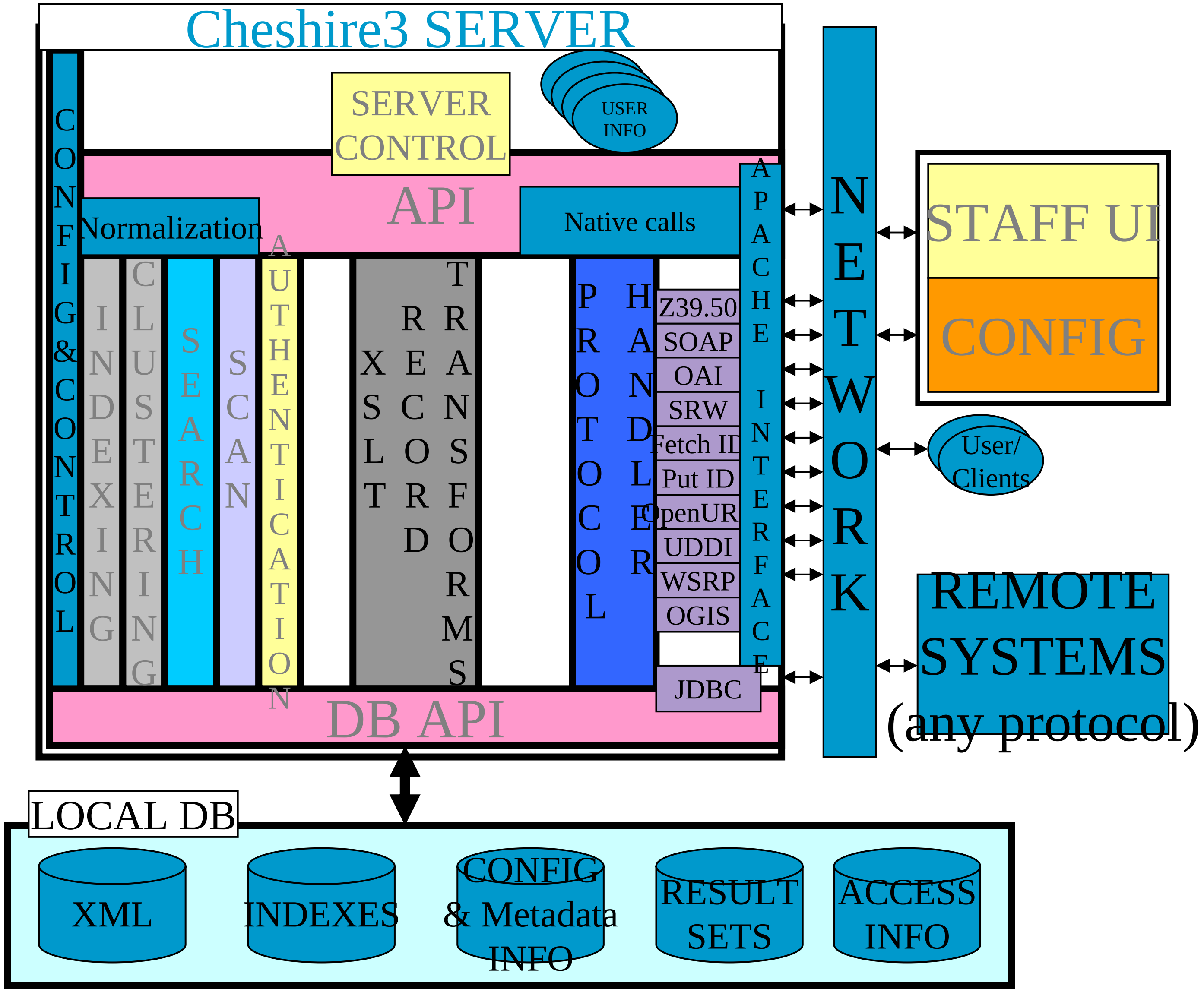
Grid IR Issues

- Want to preserve the same retrieval performance (precision/recall) while hopefully increasing efficiency (I.e. speed)
- Very large-scale distribution of resources is a challenge for sub-second retrieval
- Different from most other typical Grid processes, IR is potentially less computing intensive and more data intensive
- In many ways Grid IR replicates the process (and problems) of metasearch or distributed search

Cheshire3 Overview

- XML Information Retrieval Engine
  - 3rd Generation of the UC Berkeley Cheshire system, as co-developed at the University of Liverpool.
  - Uses Python for flexibility and extensibility, but imports C/C++ based libraries for processing speed
  - Standards based: XML, XSLT, CQL, SRW/U, Z39.50, OAI to name a few.
  - Grid capable. Uses distributed configuration files, workflow definitions and PVM (currently) to scale from one machine to thousands of parallel nodes.
  - Free and Open Source Software. (GPL Licence)
  - http://www.cheshire3.org/   (under development!)

Cheshire3 Server Overview



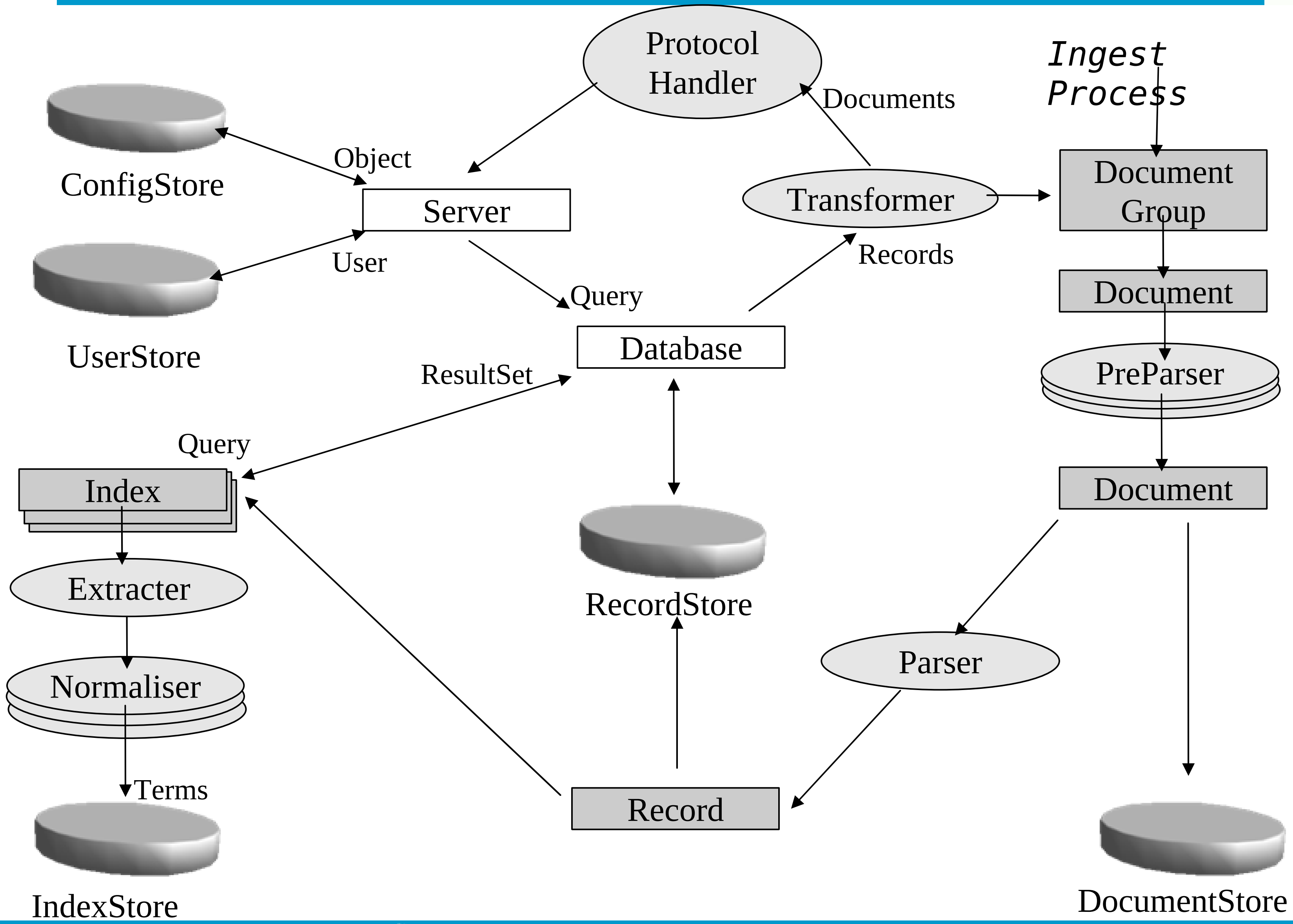
Cheshire3 Grid Tests

- Running on an 30 processor cluster in Liverpool using PVM (parallel virtual machine)
- Using 16 processors with one “master” and 22 “slave” processes we were able to parse and index MARC data at about 13000 records per second
- On a similar setup 610 Mb of TEI data can be parsed and indexed in seconds

SRB and SDSC Experiments

- We are working with SDSC to include SRB support
- We are planning to continue working with SDSC and to run further evaluations using the TeraGrid server(s) through a “small” grant for 30000 CPU hours
  - SDSC's TeraGrid cluster currently consists of 256 IBM cluster nodes, each with dual 1.5 GHz Intel® Itanium® 2 processors, for a peak performance of 3.1 teraflops. The nodes are equipped with four gigabytes (GBs) of physical memory per node. The cluster is running SuSE Linux and is using Myricom's Myrinet cluster interconnect network.
- Planned large-scale test collections include NSDL, the NARA repository, CiteSeer and the “million books” collections of the Internet Archive

Cheshire3 Object Model



Cheshire3 Data Objects

- DocumentGroup:
  - A collection of Document objects (e.g. from a file, directory, or external search)
- Document:
  - A single item, in any format (e.g. PDF file, raw XML string, relational table)
- Record:
  - A single item, represented as parsed XML
- Query:
  - A search query, in the form of CQL (an abstract query language for Information Retrieval)
- ResultSet:
  - An ordered list of pointers to records
- Index:
  - An ordered list of terms extracted from Records

Cheshire3 Process Objects

- PreParser:
  - Given a Document, transform it into another Document (e.g. PDF to Text, Text to XML)
- Parser:
  - Given a Document as a raw XML string, return a parsed Record for the item.
- Transformer:
  - Given a Record, transform it into a Document (e.g. via XSLT, from XML to PDF, or XML to relational table)
- Extracter:
  - Extract terms of a given type from an XML sub-tree (e.g. extract Dates, Keywords, Exact string value)
- Normaliser:
  - Given the results of an extracter, transform the terms, maintaining the data structure (e.g. CaseNormaliser)

Cheshire3 Abstract Objects

- Server:
  - A logical collection of databases
- Database:
  - A logical collection of Documents, their Record representations and Indexes of extracted terms.
- Workflow:
  - A 'meta-process' object that takes a workflow definition in XML and converts it into executable code.

Workflow Objects

- Workflows are first class objects in Cheshire3 (though not represented in the model diagram)
- All Process and Abstract objects have individual XML configurations with a common base schema with extensions
- We can treat configurations as Records and store in regular RecordStores, allowing access via regular IR protocols.

Workflow References

- Workflows contain a series of instructions to perform, with reference to other Cheshire3 objects
- Reference is via pseudo-unique identifiers … Pseudo because they are unique within the current context (Server vs Database)
- Workflows are objects, so this enables server level workflows to call database specific workflows with the same identifier

Distributed Processing

- Each node in the cluster instantiates the configured architecture, potentially through a single ConfigStore.
- Master nodes then run a high level workflow to distribute the processing amongst Slave nodes by reference to a subsidiary workflow
- As object interaction is well defined in the model, the result of a workflow is equally well defined. This allows for the easy chaining of workflows, either locally or spread throughout the cluster.

Workflow Example1

```
<subConfig id="buildWorkflow">
<objectType>workflow.SimpleWorkflow</objectType>
<workflow>
  <log>Starting Load</log>
  <object type="recordStore" function="begin_storing"/>
  <object type="database" function="begin_indexing"/>
  <for-each>
    <object type="workflow" ref="buildSingleWorkflow">
  </for-each>
  <object type="recordStore" function="commit_storing"/>
  <object type="database" function="commit_indexing"/>
  <object type="database" function="commit_metadata"/>
</workflow>
</subConfig>
```

Workflow Example2

```
<subConfig id="buildSingleWorkflow">
<objectType>workflow.SimpleWorkflow</objectType>
<workflow>
  <object type="workflow" ref="PreParserWorkflow"/>
  <try>
    <object type="parser" ref="NsSaxParser"/>
  </try>
  <except>
    <log>Unparsable Record</log>
    <raise/>
  </except>
  <object type="recordStore" function="create_record"/>
  <object type="database" function="add_record"/>
  <object type="database" function="index_record"/>
  <log>Loaded Record</log>
</workflow>
</subConfig>
```

Workflow Standards

- Cheshire3 workflows do not conform to any standard schema
- Intentional:
  - Workflows are specific to and dependent on the Cheshire3 architecture
  - Replaces the distribution of lines of code for distributed processing
  - Replaces many lines of code in general
- Needs to be easy to understand and create
- GUI workflow builder coming (web and standalone)

External Integration

- Looking at integration with existing cross-service workflow systems, in particular Kepler/Ptolemy
- Possible integration at two levels:
  - Cheshire3 as a service (black box) ... Identify a workflow to call.
  - Cheshire3 object as a service (duplicate existing workflow function) … But recall the access speed issue.

Conclusions

- Scalable Grid-Based digital library services can be created and provide support for very large collections with improved efficiency
- The Cheshire3 IR and DL architecture can provide Grid (or single processor) services for next-generation DLs
- Available as open source via:

http://cheshire3.sourceforge.net or

http://www.cheshire3.org/