Invenio Technology

Selected Practical Software Development Lessons From A Large Digital Library System

Tibor Šimko
<tibor.simko@cern.ch>

Department of Information Technology
CERN

August 2010 / openlab talk

Invenio Technology
Tibor Šimko
Introduction
Digital Library
Invenio
Case Studies
Episode 1: Python
Episode 2: Git
Episode 3: Test Suite
Episode 4: Building Efficient Indexes
Episode 5: Load-balancing
Conclusions

Outline

What is Digital Library?

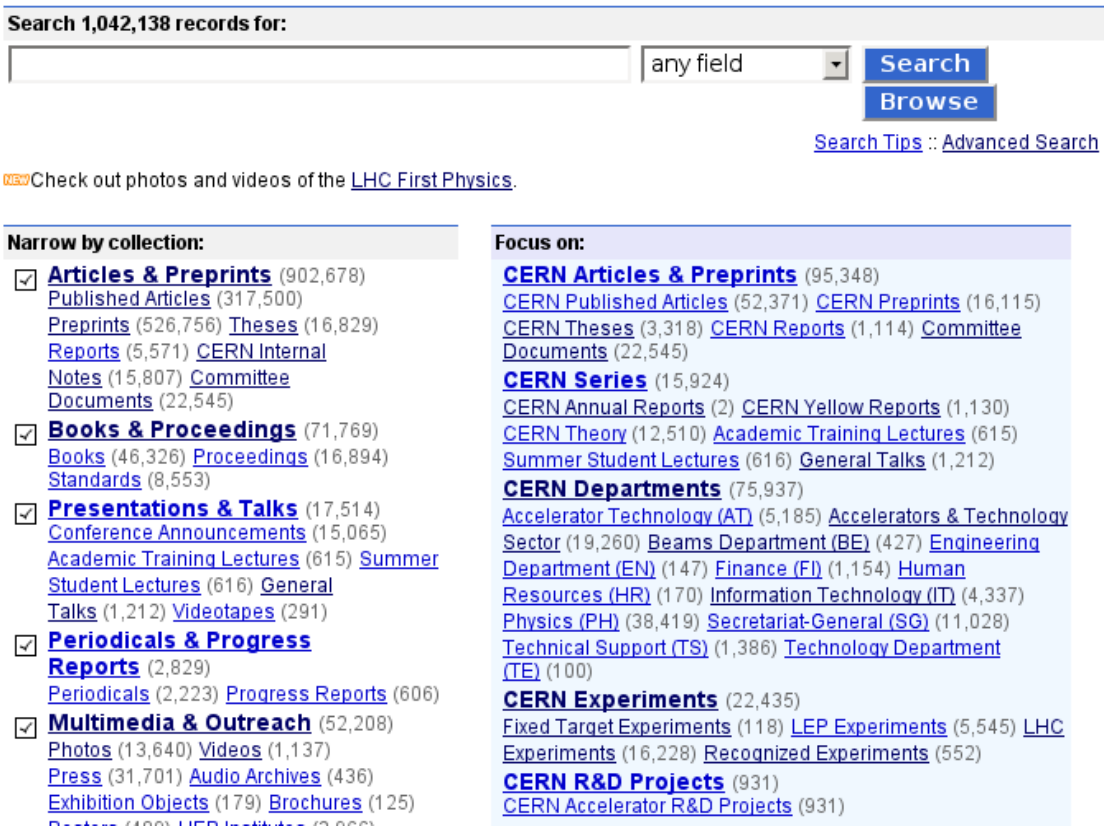
- *"library in which collections are stored in digital formats (as opposed to print, microform, or other media) and accessible by computers"*
- (1) institutional document repositories
- (2) world-wide subject-based information systems

Example: CERN Document Server

- managing CERN and selected non-CERN high-energy physics and related documents since ∼1993
- more than 1,000,000 records
- articles, books, theses, photos, videos, and more
- powered by Invenio, free digital library software
- `http://cdsweb.cern.ch/`

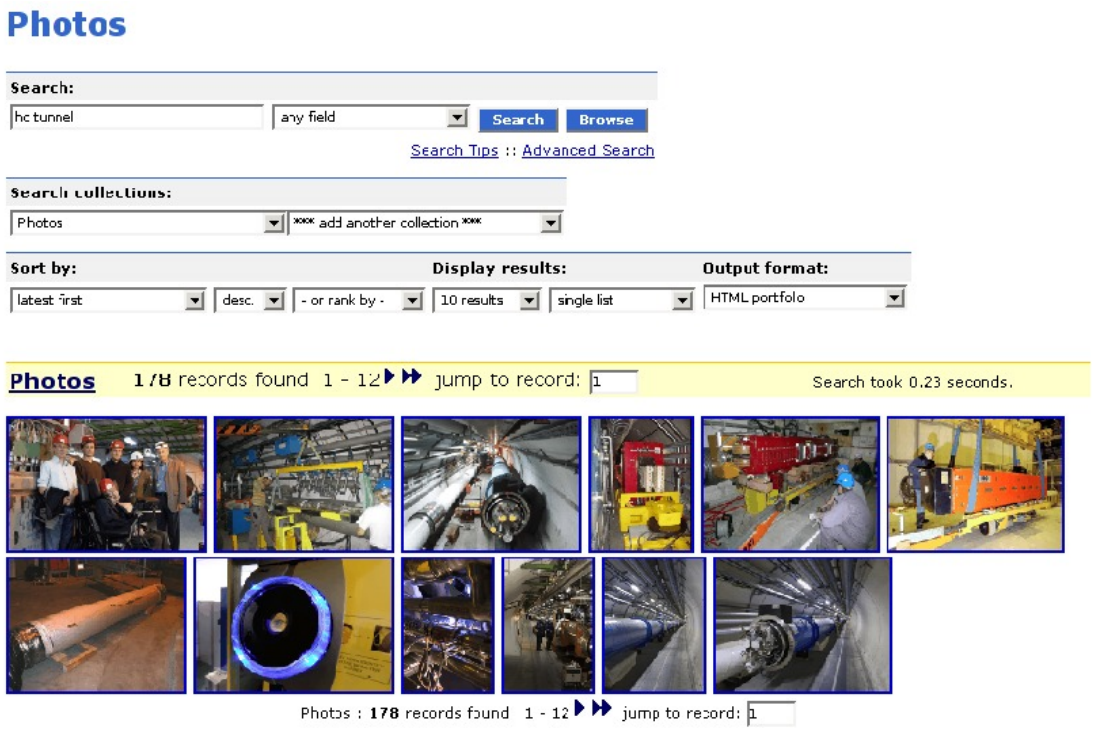
Invenio Technology
Tibor Šimko
Introduction
Digital Library
Invenio
Case Studies
Episode 1: Python
Episode 2: Git
Episode 3: Test Suite
Episode 4: Building Efficient Indexes
Episode 5: Load-balancing
Conclusions

CDS: Collection Tree

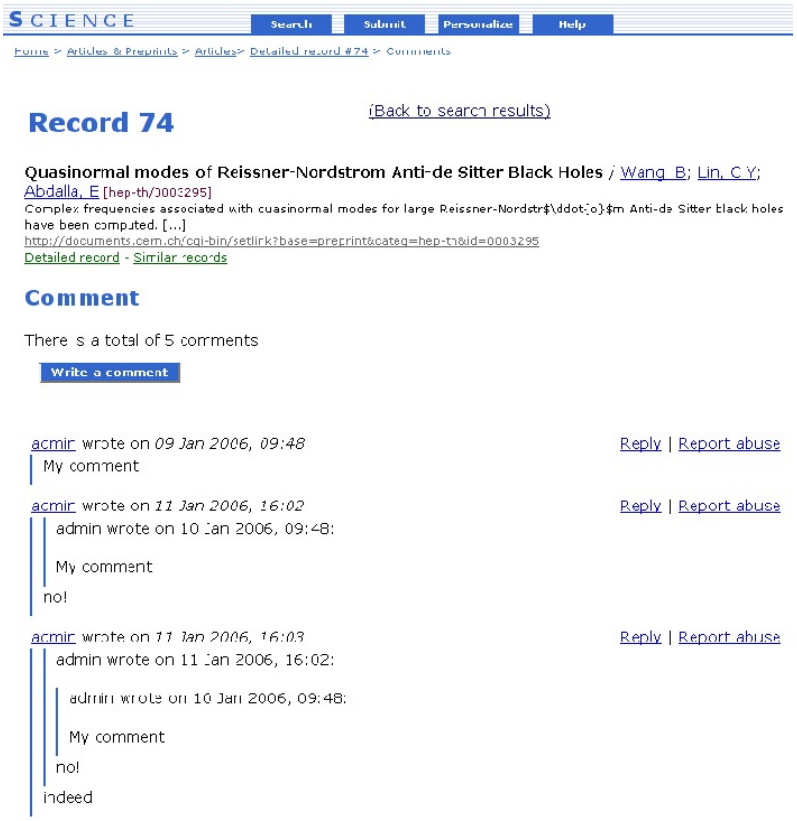


CDS: Search for Books

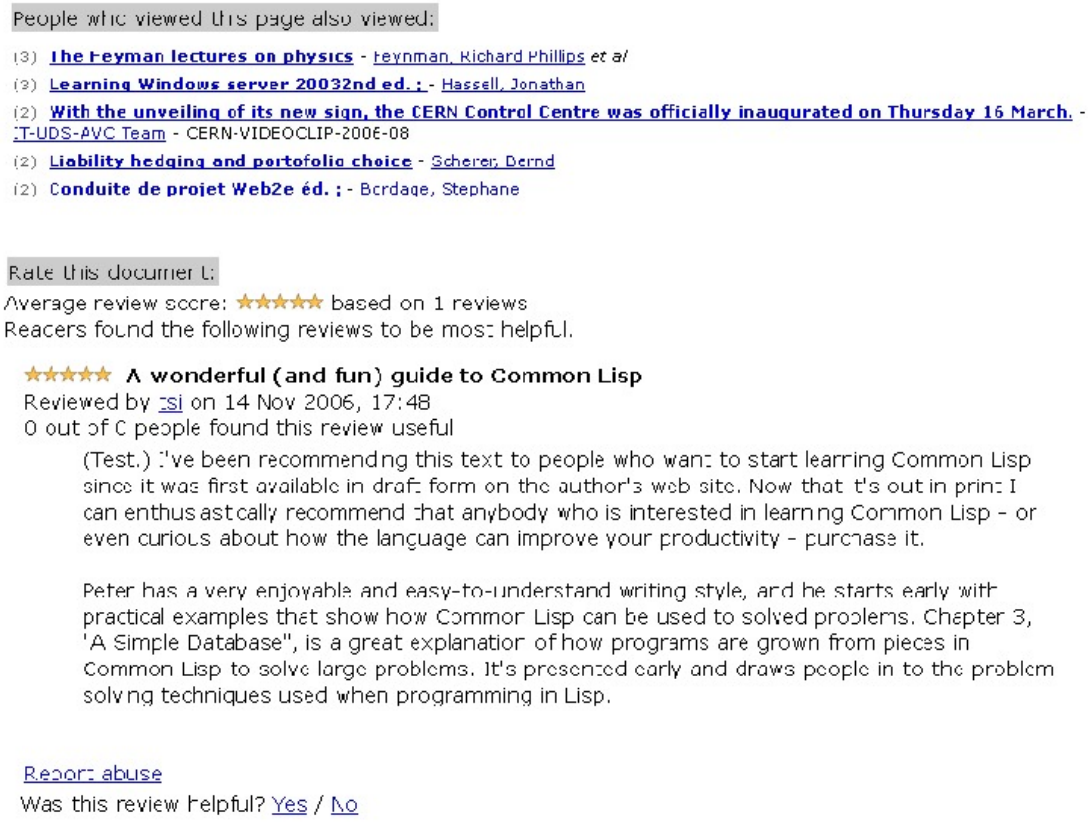
CDS: Search for Photos



CDS Features: Commenting



Invenio Features: Reviewing



CDS: Create Personal Alert

Search:

neutrino mixing | any field | Search | Browse

Search Tips :: Advanced Search

Results overview: Found **4,236** records in 0.07 seconds.

Articles & Preprints, **4,193** records found
Books & Proceedings, **19** records found
Presentations & Talks, **11** records found
Multimedia & Outreach, **13** records found

1. **Constraining sterile neutrinos with a low energy beta-beam** / Agarwalla, Sanjib Kumar
Task hep-ex
We study the possibility to use a low energy beta-beam facility to search for sterile neutrinos by measuring the disappearance of electron anti-neutrinos. This channel is particularly sensitive since it allows to use inverse beta decay as detection reaction; thus it is free from hadronic uncertainties, provided the neutrino energy is below the pion production threshold. [...]
arXiv:1006.1640; VPI-IPNAS-10-10.- 2010 - Published in : Published in AIP Conf.Proc.: 1222 (2010) , pp. 169-173 Preprint
Detailed record - Similar records

Interested in being notified about new results for this query?
Set up a personal email alert or subscribe to the RSS feed.

CDS: Add to Personal Basket

☐ 8. **Bridging flavour violation and leptogenesis in SU(3) family models** / Calibbi, Lorenzo (Max-Planck-Institut fuer Physik) ; Chun, Eung Jin (Korea Institute for Advanced Study)
We reconsider basic, in the sense of minimal field content, Pati-Salam x SU(3) family models which make use of the Type I see-saw mechanism to reproduce the observed mixing and mass spectrum in the neutrino sector. [...]
arXiv:1005.5563 ; KIAS-P10014 ; IC-2010-021 ; MPP-2010-58. - 2010.
Preprint
Detailed record - Similar records

☐ 9. **Rare muon and tau decays in A4 Models** / Feruglio, Ferruccio ; Paris, Alessio
We analyze the most general dimension six effective Lagrangian, invariant under the flavour symmetry A4 x Z3 x U(1) proposed to reproduce the near tri-bimaximal lepton mixing observed in neutrino oscillations. [...]
arXiv:1005.5526 ; DFPD-10-TH-9. - 2010.
Preprint
Detailed record - Similar records

☐ 10. **Quark and lepton mixing angles with a dodeca-symmetry** / Kim, Jihn E ; Seo, Min-Seok
The discrete symmetry D_12 at the electroweak scale is used to fix the quark and lepton mixing angles. [...]
arXiv:1005.4684. - 2010.
Preprint
Detailed record - Similar records

ADD TO BASKET

CDS: Display Personal Basket

Display baskets

Personal baskets > Physics Back to Your Baskets | Create basket | Edit topic

Standard Model (3)

Standard Model Edit basket | Delete basket
3 items, 2 notes
last update: 11 Jun 2010, 14:21

1. **Non-Abelian Flat Directions in a Minimal Superstring Standard Model** / Cleaver, G B ; Faraggi, A E ; Nanopoulos, Dimitri V ; Walker, J W [ACT-2000-1] [CTP-TAMU-2000-2] [OUTP-2000-03-P] [TPI-MINN-2000-6] [hep-ph/0002060]
   Recently, by studying exact flat directions of non-Abelian singlet fields, wedemonstrated the existence of free fermionic heterotic-string models in whichthe SU(3)_C x SU(2)_L x U(1)_Y-charged matter spectrum, just below the stringscale, consists solely of the MSSM spectrum. [...]
   Published in **Mod. Phys. Lett. A: 15 (2000) pp. 1191-1202**
   Fulltext: PDF; PS.GZ
   Detailed record - Notes (2) Copy item | Remove item

2. **Precise calculation of parity nonconservation in cesium and test of the standard model** / Dzuba, V A ; Flambaum, V V ; Ginges, J S M [hep-ph/0204134]
   We have calculated the 6s-7s parity nonconserving (PNC) E1 transition amplitude, E_{PNC}, in cesium. [...]
   Fulltext: PDF; PS.GZ
   Detailed record - Add a note... Copy item | Remove item

CDS: Organize and Share Your Baskets

Display baskets

Personal baskets | Group baskets | Public baskets

Physics (1)
Standard Model

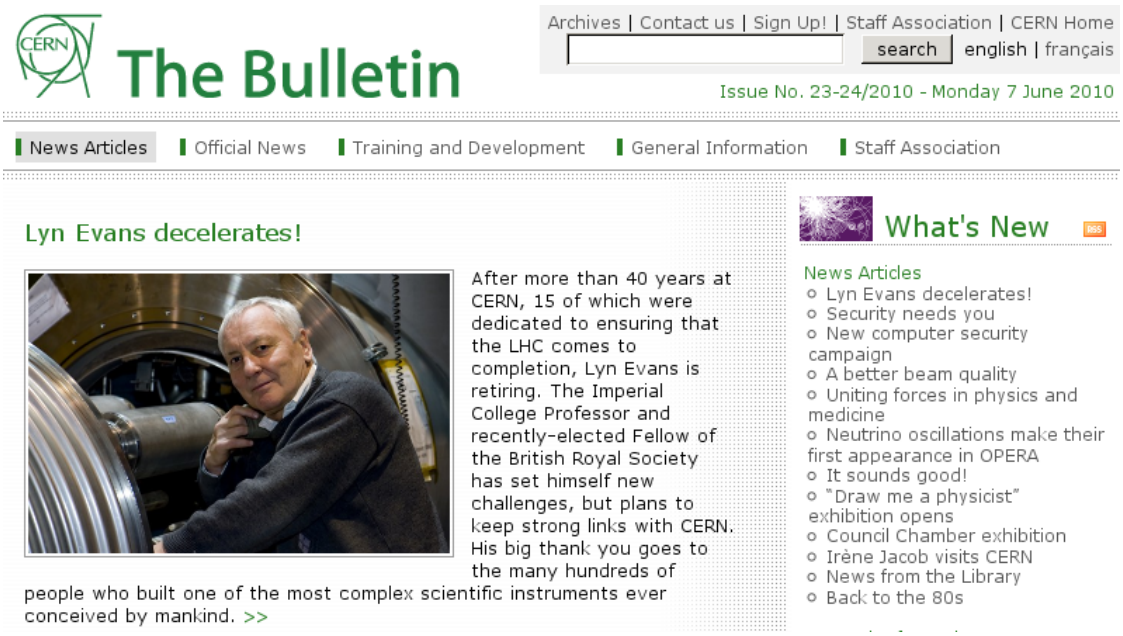
Programming (3)
Linux, Python, SQL

Search baskets for:

in Your personal baskets Search

☐ Search also in notes (where allowed)

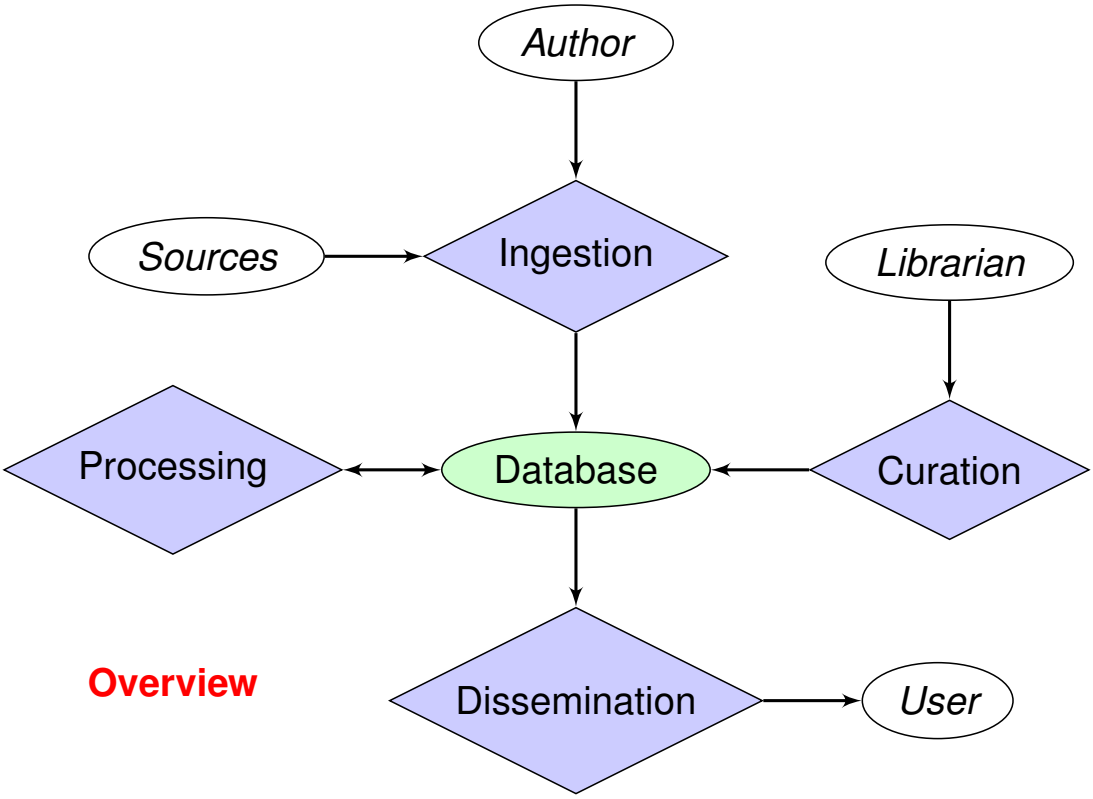
CDS: Journals and Bulletins



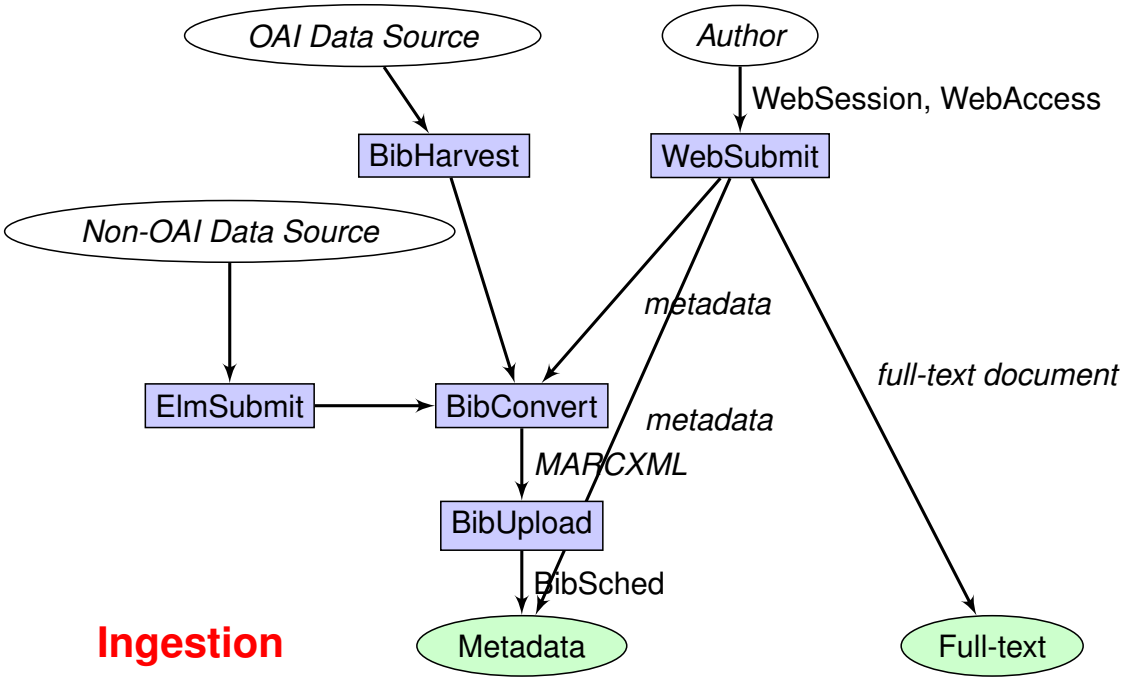
Invenio Key Features

- **navigable collection tree** (regular, virtual)
- **powerful search engine**
  - Google-like speed for up to 5M records
  - combined metadata, reference and fulltext search
- **flexible metadata** (MARC, OA)
  - handling any kind of document (multimedia)
  - customizable input, formatting and linking
- **personalization** and **collaborative** features:
  - alerts, baskets, groups, reviews, comments
  - internationalization (26 languages)
- **open source**, GNU General Public License
  - co-developed by CERN (2002–), EPFL (2004–), DESY/FNAL/SLAC (2008–), CfA (2009–)
  - installed at ∼30 institutions world-wide

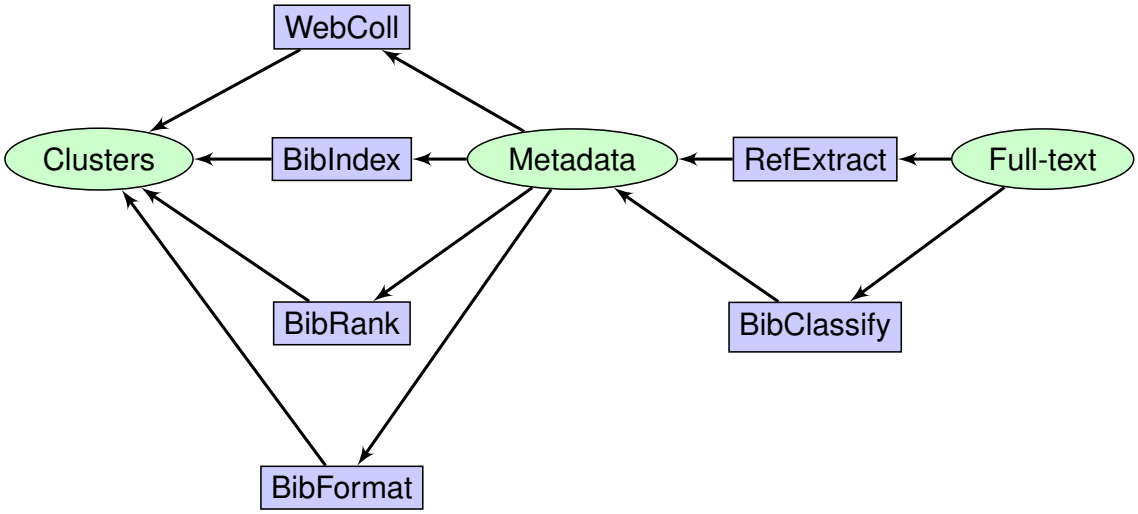
Invenio Architecture: Overview



Invenio Modules: Ingestion

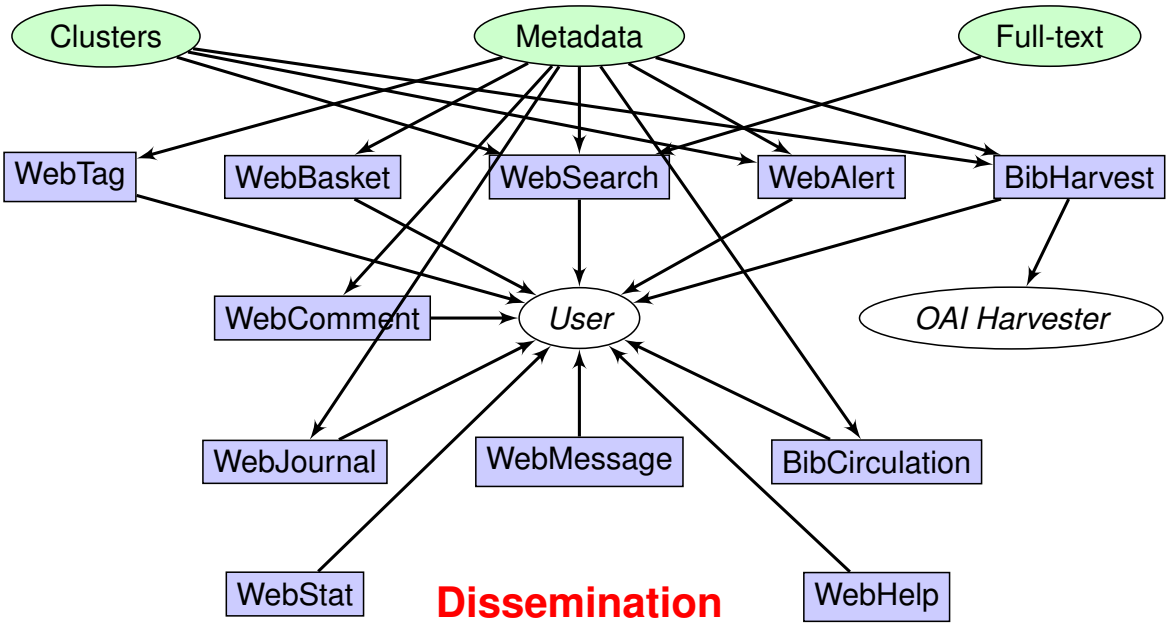


Invenio Modules: Processing



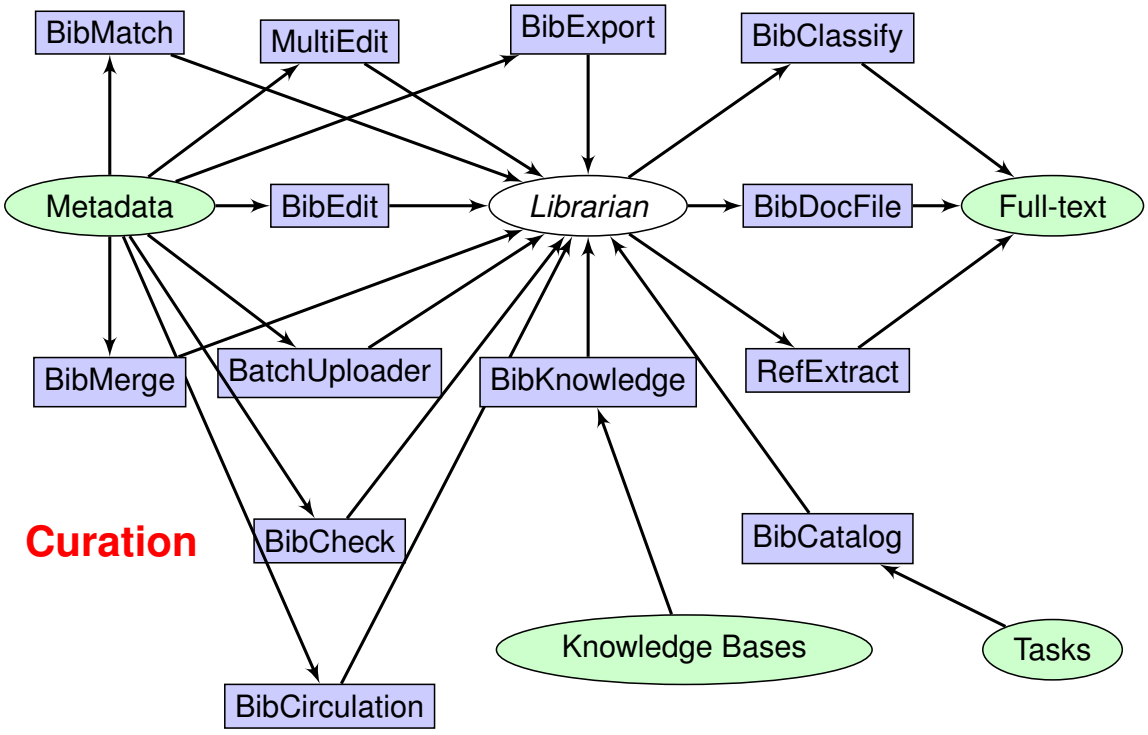
Processing

Invenio Modules: Dissemination



Dissemination

Invenio Modules: Curation



Invenio Modules: Summary

- ∼33 modules
- codebase
  - ∼250,000 lines of Python code
  - ∼10,000 lines of JavaScript code
  - ∼6,000 lines of XSL code
  - ∼5,000 lines of autotools code
- ∼40 authors
  - many short-term students
  - importance of *informal* coding standards
- ∼10 years of development
  - started at CERN, first release in 2002
  - now co-developed world-wide (EU, US)
- lego programming... but no silver bullet

Why Python?

- easy to read and understand
  (good for many temporary developers)
- suitable for rapid prototyping
  (good for organic-growth software development model)
- write code to throw it away

Art of Ikebana

- Japanese art of flower arrangement
- “way of flowers”
- natural shapes, graceful lines
- minimalism
- *“disciplined art form in which nature and humanity are brought together”*

Art of Ikebana Programming

Java?

```
new Callable() {
    public Object call(Object x) {
        return x.times(k)
    }
}
```

Python!

```
lambda x: k * x
```

Speeding Up Python

- bytecode interpreted language
- but **Cython** permits to write C extensions easily
- combining efficiency of C with high-levelness of Python

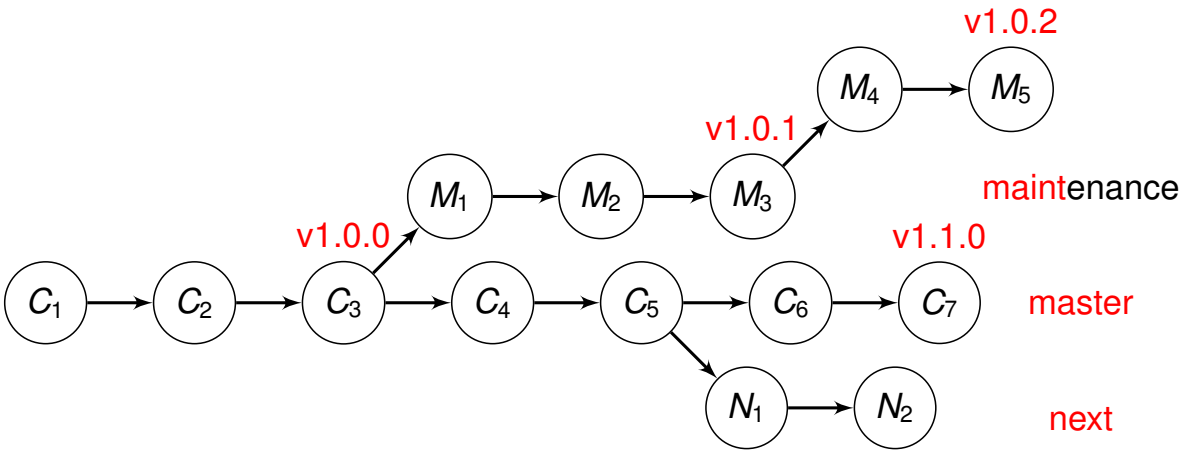
Example: intbitset.pyx

```
ctypedef unsigned long long int word_t

ctypedef struct IntBitSet:
    int size
    int allocated
    word_t trailing_bits
    int tot
    word_t *bitset
```

Why Git?

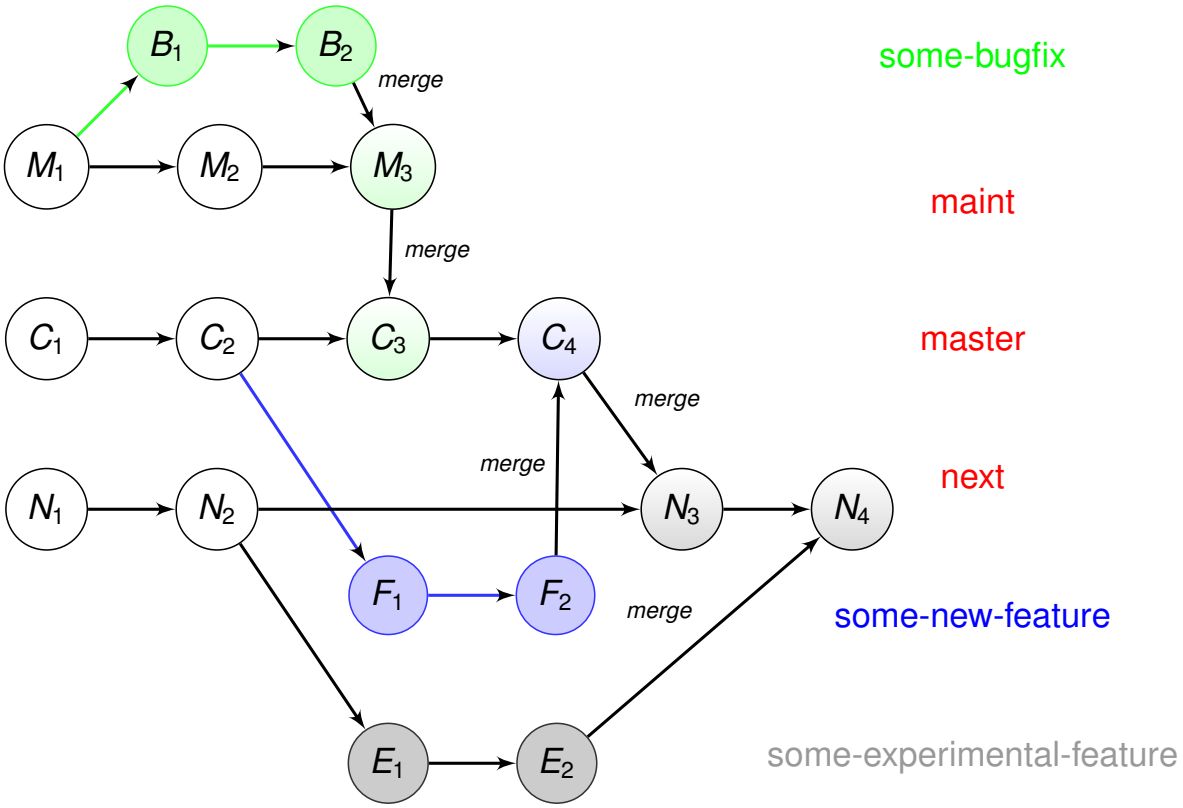
- good for distributed teams
- offline development possible
- “pull on demand” collaboration model (as opposed to “shared push” collaboration model)
  - inherent,natural code review process
- commit early, commit often (to private repositories)
- rebase and clean (before pushing for public consumption)
- interplay with SVN

Git Branches

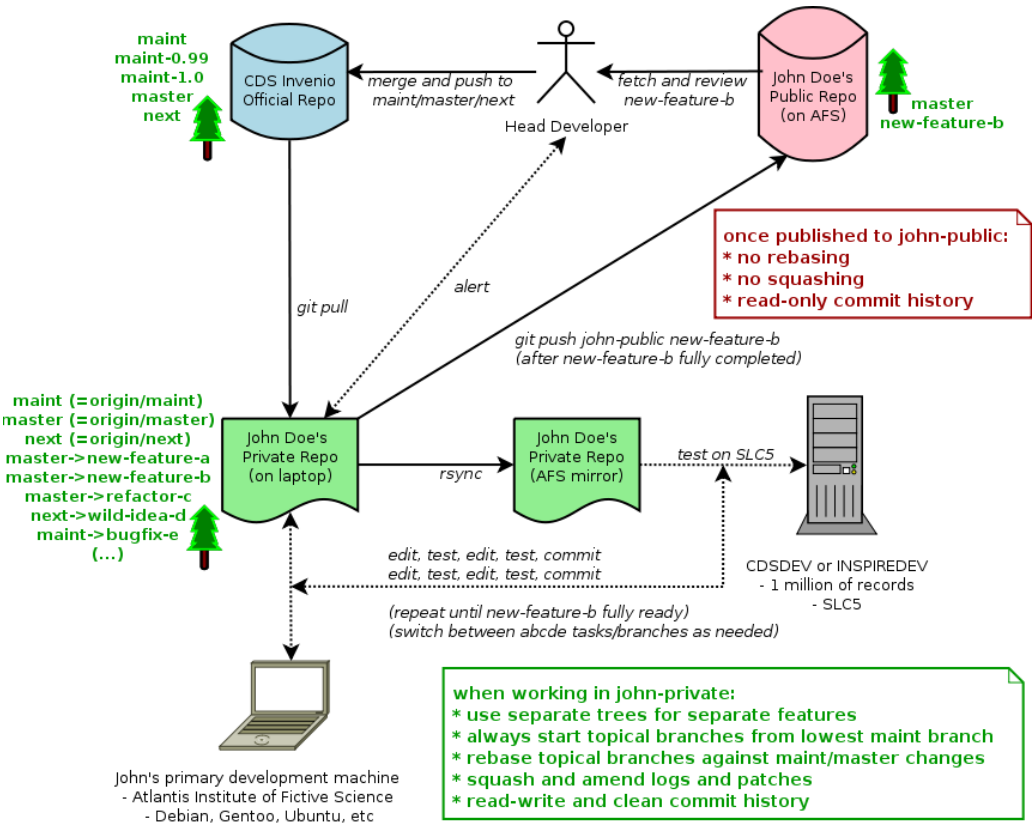


- **maint** — release maintenance branch
- **master** — new feature branch
- **next** — things not yet release-ready

Git Development



Git collaboration model



Unit testing

- **test-driven development** when appropriate
- e.g. before/while developing `strip_accents()`, write:

Example: search_engine_tests.py

```
class TestStripAccents(unittest.TestCase):
    """Test for handling of UTF-8 accents."""

    def test_strip_accents(self):
        """search engine - stripping of accented letters"""
        self.assertEqual("memememe",
             search_engine.strip_accents('mémêmëmè'))
        self.assertEqual("MEMEMEME",
             search_engine.strip_accents('MÉMÊMËMÈ'))
```

Functional testing

- functional/acceptance/regression testing
- testbed site (Atlantis of Institute Fictive Science)
- e.g. Python **mechanize** module to emulate browser

Example: websearch_regression_tests.py

```
class WebSearchSearchEnginePythonAPITest(unittest.TestCase):
    "Check typical search engine Python API calls on the demo data."

    def test_search_engine_python_api_for_failed_query(self):
        "websearch - search engine Python API for failed query"
        self.assertEqual([],
                         perform_request_search(p='aoeuidhtns'))

    def test_search_engine_python_api_for_successful_query(self):
        "websearch - search engine Python API for successful query"
        self.assertEqual([8, 9, 10, 11, 12, 13, 14, 15, 16, 17,
                          18, 47],
                         perform_request_search(p='ellis'))
```

Web testing

- sometimes we need to run tests in real browser
  - e.g. pages with heavy JavaScript
- using **Selenium IDE** extension for Firefox
  - record and replay browser actions
  - test for text existence or non-existence on pages
  - test for link labels and targets

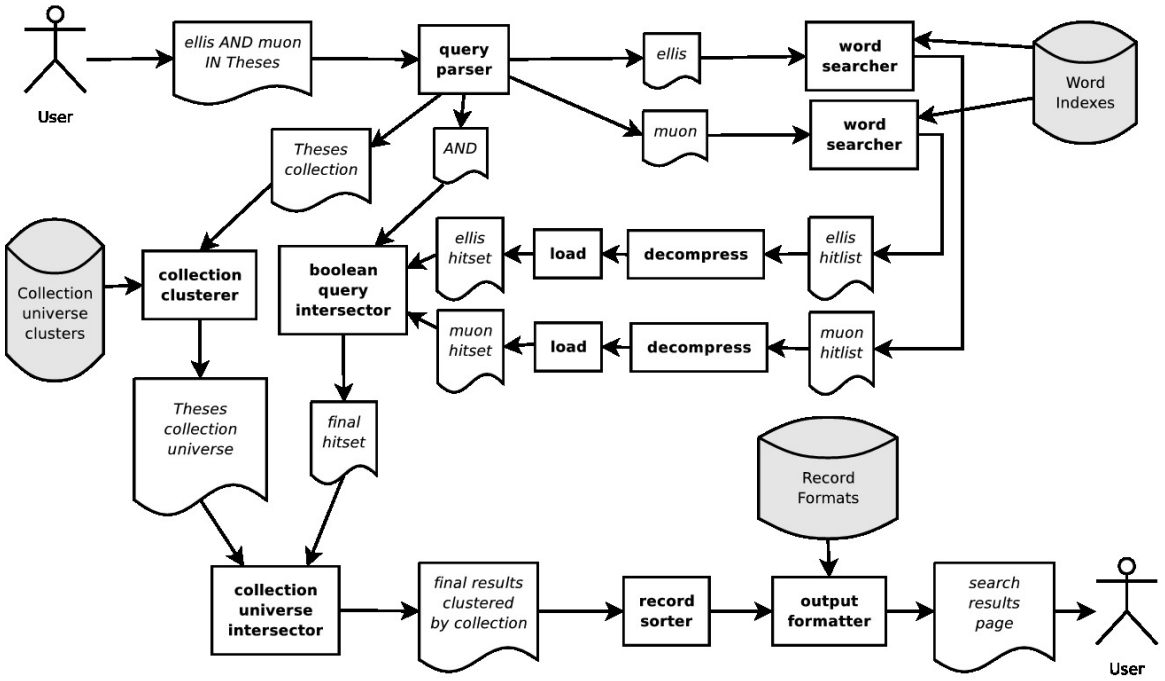
Example: test_search_ellis.html

```
<tr><td>open</td>
    <td>http://localhost</td>
    <td></td>                          </tr>
<tr><td>type</td>
    <td>p</td>
    <td>ellis</td>                      </tr>
<tr><td>clickAndWait</td>
    <td>action_search</td>
    <td></td>                           </tr>
<tr><td>verifyTextPresent</td>
    <td>1. Thermal conductivity of dense quark matter and cooling of stars</td>
    <td></td>                           </tr>
```

Designing A Search Engine

- **performance-driven design** assumptions:
  - high number of selects, low number of updates
  - fast searching, slow indexation
  - cache everything cacheable
- **search functionality:**
  - search for words, phrases, regular expressions
  - search in any field, authors, titles, etc
- **index design:**
  - forward indexes: $word1 \longrightarrow [rec1, rec2, \ldots]$
    $word2 \longrightarrow [rec2, rec7, \ldots]$
  - reverse indexes: $rec1 \longrightarrow [word1, word8, \ldots]$
    $rec2 \longrightarrow [word1, word2, \ldots]$
- **Zipf's law** on word frequency:
  - few words occur very often (e.g. *the*)
  - most words are infrequent (even e.g. *boson*)

Search Engine Under Cover



Measuring the Performance

- three important **speed factors** to consider:
  - speed of finding sets (DB Server)
  - speed of demarshaling sets (DB $\leftrightarrow$ Web App Server)
  - speed of intersecting sets (Web App Server)

Example: speed of various parts (2002, before optimization)

action / query:	"CERN 2002"	"of the this"
fetching	0.28 sec	0.34 sec
demarshaling	0.78 sec	1.10 sec
adding colls	0.37 sec	0.63 sec
intersecting	0.64 sec	1.19 sec
total search time	2.07 sec	3.22 sec

Optimizing Data Structures

- **data structures** tested:
  - 'sorted' (lists, Patricia trees)
  - 'unsorted' (hashed sets, binary vectors)
- **fast prototyping**: (Python, Lisp in 2002)
  - throw-away coding to test ideas

Example: lists vs dicts, 350K sets in 800K universe

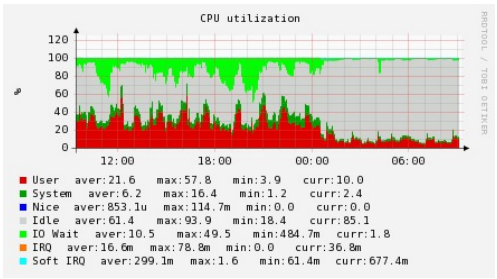
```
marshaling lists ..... 532616+532571 bytes in 1.33 sec
demarshaling lists ... 350000+350000 items in 0.10 sec
merging lists ........ 546965 items in 0.34 sec
intersecting lists ... 153035 items in 0.35 sec

marshaling dicts ..... 576491+576450 bytes in 0.87 sec
demarshaling dicts ... 350000+350000 items in 0.36 sec
merging dicts ........ 546965 items in 0.09 sec
intersecting dicts ... 153035 items in 0.15 sec
```

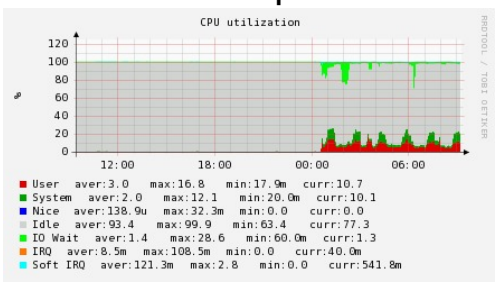
. . . and the winner is:

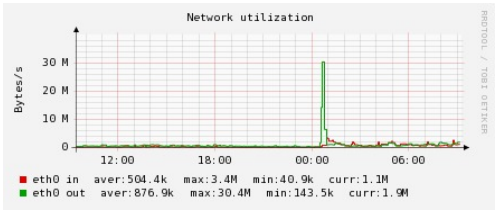
- **binary vectors** found the best compromise!
  - using `Numeric` Python module (in 2002)
  - typical search time gain: 4.0 sec $\rightarrow$ 0.2 sec (in 2002)
  - typical indexing time loss: 7 hours $\rightarrow$ 4 days (in 2002)
  - mostly spare data modelled via mostly dense data structure?
  - free your mind, think critically
- further optimization:
  - `Numeric` module not addressing real bits, only bytes
  - so home-made `intbitset` C extension in 2007
    - addressing real bits (factor of 8 already)
    - saving space, saving (indexing) time

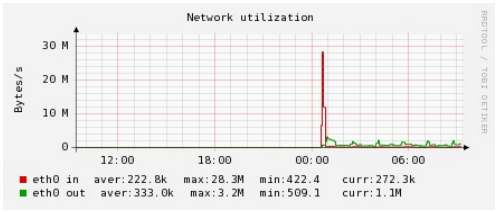
Splitting Web App Server and DB Server

- load of CDS Web and DB servers at the split time:







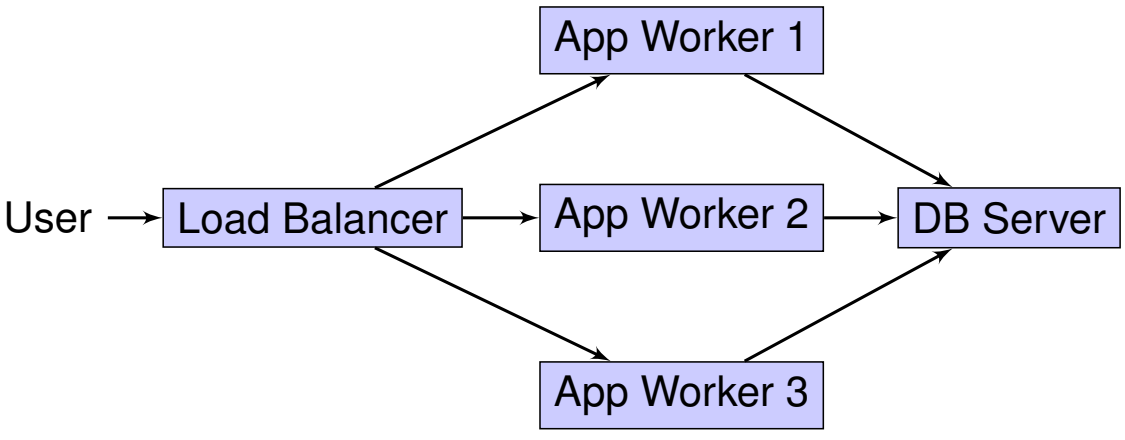


web + db $\rightarrow$ *web* *idle* $\rightarrow$ *db*

- split leads to efficient use of OS resources by lone, non-competing Web and DB daemon processes

Load-Balanced Setup

- useful for “LHC First Beam Day” rush situations with many concurrent visitors
- Apache **mod_proxy_balancer**



Measuring Scalability

- using **siege** to simulate concurrent users and to measure throughput on a sample of typical URLs

Example: inspirebeta.net under gentle siege

```
$ siege -d 1 -c 20 -t 1m -f inspirebeta_urls.txt
Transactions:                   1329 hits
Availability:                 100.00 %
Elapsed time:                  60.23 secs
Data transferred:              37.12 MB
Response time:                  0.41 secs
Transaction rate:              22.07 trans/sec
Throughput:                     0.62 MB/sec
Concurrency:                    8.96
Successful transactions:        1329
Failed transactions:               0
Longest transaction:            3.05
Shortest transaction:           0.01
```

Conclusions

- building Invenio digital library system
  - ∼250,000 LOCs from ∼40 authors over ∼10 years
- value of rapid prototyping
- value of organic-growth software development model
- value of coding aesthetics and minimalism
- morale from selected anecdotes?
  - *“Never Lose A Holy Curiosity”* (A. Einstein)