Particle Physics On The Couch: Using CouchDB To Help Unravel The Mysteries Of The Universe

Michael Marino
Technische Universität München
What are these mysteries and how are we looking for them?
Back to the beginning…
The difference between matter and antimatter was tiny…

‘right after’ the Big Bang
... but we can only explain

\[ 10^{10} \]

\[ 10^{10} + 10^{-8} \]
... but we can only explain $10^{10} < 1$!
We search for a particular characteristic of the neutron:

**neutron Electric Dipole Moment (nEDM):**
a (tiny!) separation of electrical charge within the neutron

Discovery of this helps to solve the matter/anti-matter problem
How small is \( d \)?

If we blow up the neutron to the size of the earth...

\[
\begin{align*}
\text{d} & \quad \text{<} \\
\text{Virus} & \\
\text{Image:} & \quad \text{NASA/NOAA/GSFC/Suomi NPP/VIIRS/Norman Kuring}
\end{align*}
\]

\[
\begin{align*}
\text{If we blow up the neutron to the size of the earth...} & \\
\text{Image:} & \quad \text{Wikipedia, Graham Colm}
\end{align*}
\]
How do we find it?

... with a bucket full of neutrons

(we can literally store neutrons in a bottle for ~100s of seconds, and then investigate them)
It’s a bit more complicated than that…

Very important: control/understand magnetic fields
It’s a bit more complicated than that...

Experimental setup at the TUM (Garching, Germany)
It’s a bit more complicated than that…

Magnetometer (Cs laser)

The neutron chamber

“Robot” for mapping out magnetic fields
Many different subsystems:
- Cs Laser + magnetometry system
- Hg Laser + co-magnetometer
- External magnetometry
- Active coil earth magnetic field compensation
- Temperature/humidity monitoring
- Neutron detection
- Valve monitoring/control
- B0 coil/current controls
- Vacuum monitoring
- Nuclear Magnetic Resonance System
- SQUID
- Degaussing
- etc...

All of these systems take and write data

Some of them must also be controlled
A lot of independent hardware systems, in general with very different requirements/dependencies/languages, etc.

But they all need to play together!
Basic concept

CouchApp

Data

Control

CouchDB

Hardware devices

ORCA

Control via changes feed

LabVIEW

python™

CASCADe
How do we use CouchDB’s features?
Why CouchDB?

RESTful Interface

= small hurdle

- Arduinos
- Raspberry Pi
- VMEbus
Why CouchDB?

- RESTful Interface
- Embedded web app

Control/monitor all sub-systems from a browser. Automatic cross-platform (-device)* support
Why CouchDB?

- **RESTful Interface**
- **Embedded web app**
- **Simplicity/Scalability**

Every subsystem is a database,

- allowing granularity of control (only some users get the rights to change a particular system)
- systems are kept separate
- new system = new database
- should be easy (fun!) for students (undergrads and PhDs) to build
Why CouchDB?

RESTful Interface

Embedded web app

Simplicity/scalability

Every subsystem is a database,
- allowing granularity of control (only some users get the rights to change a particular system)
- systems are kept separate
- new system = new database
- should be easy (fun!) for students (undergrads and PhDs) to build
Why CouchDB?

- RESTful Interface
- Embedded web app
- Simplicity/scalability
- Replication

Twofold:

Partial replication (a subset of databases/subsystems)
Deploying of a test apparatus at another site, etc.

Remote monitoring + control (bi-directional replication)
Writing data, and a subsystem example
Generally, only “slow” data is written to a database for a system (write frequency ~ 1/s to 1/min)

```json
{
  "_id": "2509b27958e8271bd18fc002610011a3",
  "_rev": "1-95680c33defe09e7a2fcc07c9cb755cc",
  "timestamp": "Sun, 28 Jul 2013 09:10:43 GMT",
  "value": {
    "sensor9": 1.3820271950588525,
    "sensor1": 1.4459014666715626,
    "sensor0": 0.18376116050815683,
    "sensor3": 1.868015638732785,
    "sensor2": 1.2473604704118044,
    "sensor5": 1.6353645865447695,
    "sensor4": 3.6022805562067934,
    "sensor7": 2.5801280996424794
  },
  "created_by": "mgmarino",
  "type": "data"
}
```

Timestamp in RFC 1123 (autofilled)

Dictionary of data taken at this timestamp

Who saved it (autofilled)

Slow-control data type

Unified “data” used across different subsystems
Writing to the database: views

We are interested in the time behaviour of variables, two main views:

Map:

1. key: ['varname', YYYY, MM, DD, HH, MM, SS]
   value: varvalue

2. key: [YYYY, MM, DD, HH, MM, SS, 'varname']
   value: varvalue

Reduce:

_stats

none
Writing to the database: views

is used more often, especially in our web interface, allowing us to easily look at the average (or extreme values) over time (and live, via changes)

Temperatures:

seconds

group_level=7

[‘varname’, YYYY, MM, DD, HH, MM, SS]
Writing to the database: views

is used more often, especially in our web interface, allowing us to easily look at the average (or extreme values) over time (and live, via changes).

Temperatures:

minutes

group_level=6

['varname', YYYY, MM, DD, HH, MM]
is used more often, especially in our web interface, allowing us to easily look at the average (or extreme values) over time (and live, via changes)

Temperatures:

hours

\texttt{group\_level=5}

[‘varname’, YYYY, MM, DD, HH]
Temperature subsystem

~ 45 Temp Sensors
+ 2 Humid. Sensors

Master’s project for 2 students!

https://github.com/nEDM-TUM/Temperature-Sensor-System
Controlling a subsystem
Basic concept

CouchApp

Control

Hardware devices

Control via changes feed

CouchDB
Subsystem saves a document with the list of available commands:

```json
{
   "_id": "commands",
   "keys": {
      "test_func": {
         "Info": "test_func(*args)\n            Demonstrate how well this works.\n            This is also automatically included in the "commands" document.\n"
      },
      "stop": {
         "Info": "stop = stop_listening(stop=True)\n            Request the listening to stop. Code blocked on wait() will proceed.\n"
      }
   },
   "uuid": "233758443303732",
   "created_by": "raspberry_34",
   "timestamp": "Tue, 21 Oct 2014 14:54:55 GMT"
}
```
Subsystem waits for command document, listens to changes feed

/db/_changes?feed=continuous&heartbeat=5000&filter=execute_commands/execute_commands

filter on “command” documents
Overview

A command document is saved by a user, either from web interface or from another program.
Overview

```
{
    "id": "0505ecf69487bb625fff117144685520",
    "rev": "2-d48f090ea53a2006db8667c9274d5336",
    "type": "command",
    "execute": "test_func",
    "arguments": [
        "2",
    ],
    "created_by": "nedm_user",
    "timestamp": "Tue, 21 Oct 2014 11:16:31 GMT",
    "response": {
        "content": "'test_func' success",
        "timestamp": "Tue, 21 Oct 2014 11:16:30 +0000",
        "return": [
            ["2",
        ],
        "ok": true
    }
}
```
Caveats:
- This is not a “real-time” command system!
- The “glue” for each subsystem must be written

Most of the time, we can write this in Python, and we have a module that handles this...

```python
import pynedm

def test_func(*args):
    ""
    Demonstrate how well this works. This is also automatically included in the "commands" document.
    ""
    print "test_func called"
    print args
    return [args]

func_dic = { "test_func" : test_func }

pynedm.listen(
    func_dic, # dictionary of functions to call
    "nedm%2Fhg_laser", # database name
    uri="http://raid.nedm1:5984", # server
    username="username", # username
    password="password" # password
)

pynedm.wait() # Wait in daemon mode, may be safely exited with
              # CTRL-C, or a command "stop" from the server
```

pynedm: https://github.com/nEDM-TUM/Python-Slow-Control
Uses: https://github.com/cloudant-labs/cloudant-python!
... but we also have something for Objective-C for ORCA (an open-source, Mac OS X-based data acquisition software)

http://orca.physics.unc.edu/~markhowe/Database_Support/CouchDB_Listener.html
... and also for straight C (when we have to)

Pillowtalk

Original [https://github.com/jubos/pillowtalk](https://github.com/jubos/pillowtalk)

Forked [https://github.com/mgmarino/pillowtalk](https://github.com/mgmarino/pillowtalk)
(with changes feed notifications, Windows + *X support)

(Closed source, but can load C libraries)

Have implemented, tested, but until now have always been able to use Python in favor of the pure C “glue”
A few more points

The command system doesn’t need to “export” the entire functionality of the subsystem. Generally, only a small subset of commands are necessary.

ToDo: Locking mechanisms to ensure only “one user” is controlling system. Designed, not yet implemented.
Some more “goodies”
Alarms/Notifications

All good monitoring/slow-control systems must provide notifications if certain events occur:

i.e. a value exceeds a limit, a valve opens, etc.

Our best solution: OS daemon

Auto started/configured by CouchDB
Polls to determine condition exceptions

https://github.com/nEDM-TUM/Slow-Control-Misc/blob/master/couchdb_alarm_daemon.py
Alarms/Notifications

All good monitoring/slow-control systems must provide notifications if certain events occur:

- a value exceeds a limit,
- a valve opens,
- etc.

Our best solution: OS daemon auto started/configured by CouchDB!

Polls to determine condition exceptions

https://github.com/nEDM-TUM/Slow-Control-Misc/blob/master/couchdb_alarm_daemon.py
(Almost all) devices live on our local experiment network and many support SCPI:

   e.g. "*IDN?", "*CAL", etc.

Almost all of these now run over Ethernet (instead of GPIB, RS-232, USB, etc.)

Desktop = overkill
Raspberries

Runs a python daemon (HimbeereCouch):

https://github.com/nEDM-TUM/HimbeereCouch

1. On boot: request code from CouchDB (identified by MAC)
2. Run “readout” code
3. Listen for changes (restart “readout” code with changes)
Putting it all together…

An excerpt from our HTML interface:

https://github.com/nEDM-TUM/nEDM-Interface
Putting it all together…

![Image of a Measurements interface with a test name, saved data, and coil check details]

- **Name:** Test
- **Saved:** 5fef603251db43bb3c947ef623ee776
- **Check coil 11**
  - **Active Coil Compensation:** Check
  - **Variable name:** Coil_0
  - **Min:** 4
  - **Max:** 7
  - **Status:** value (4.579999923706055) in range (4 -> 7)

**Start Fluxgate**
Putting it all together…
A (shrunk-down) read-only example

https://nedmtum.cloudant.com/nedm_head/_design/nedm_head/_rewrite/

(Continuous) replication of a subset of our systems, with some live examples

Kindly hosted by:

Cloudant

UN + Password: apachecon2014
Summary

• The nEDM experiment at TUM is trying to help explain the matter-anti-matter asymmetry in the universe.

• CouchDB is an essential piece of the data acquisition system used by the nEDM experiment. It allows the integration, control and readout of many distinct sub-systems.
nEDM-TUM collaboration


Students who have contributed significantly:

W. Chen, F. Kaspar, T. Reschenhofer, R. Schönberger, T. Stolz, B. Waltl

This work supported partially by SPP 1491 funds
Thanks for your attention!

Questions?