Miłosz Zdybał

GRID: Data access and management for Belle II

Abstract

- Latest works
  - Data handling functions
  - Bookkeeping stress test
- Encountered issues
  - Reasons
  - Possible solutions
- What next?

Data handling functions

- Initially using Python 2.4
  - Grid Python APIs
    - LCG-Util (lcg_utils)
    - LFC (lfc)
  - No easy way for multiprocessing – *fork*
- Decision – Python 2.6
  - Nice multiprocessing with *Pool* from *multiprocessing* library (went from 18 lines of code to 2!)
  - LCG-Util and LFC not compatible – using *subprocess.Popen* for commandline calls

There is no good way

```
files = self.listDataset(dataset, False)
parent_pid = os.getpid()
subparent = 0
try:
    for se in ses:
        if os.getpid() == parent_pid:
            pid = os.fork()
            if pid == 0:
                subparent = os.getpid()
                for file in files:
                    if os.getpid() == subparent:
                        pid = os.fork()
                        if pid == 0:
                            self.replicate(se, file)
                if os.getpid() == subparent:
                    for i in xrange(len(files)):
                        os.wait()
    if os.getpid() == parent_pid:
        for i in xrange(len(ses)):
            os.wait()
```

```
(res, surl, errmsg) = lcg_util.lcg_rep3(lfn, self.SES[se], ...)
```

There is no good way

```
files = self.listDataset(dataset, False)
pool = multiprocessing.Pool(len(ses)*len(files))
pool.map(self.replicate, itertools.product(ses, files))
```

```
name = 'lcg-rep'
args = ['--vo', self.VO, '-d', self.SES[se], lfn]
cmd = [name]
cmd.extend(args)
(our, err) = subprocess.Popen(cmd).communicate()
```

Data handling – features

- Parallel replication of files and datasets
- Listing datasets
- LFN → GUID conversion
- GUID → LFN conversion
- Listing replicas of file
- Uploading files and datasets to SE
- Deleting files and datasets

administrator

internal use

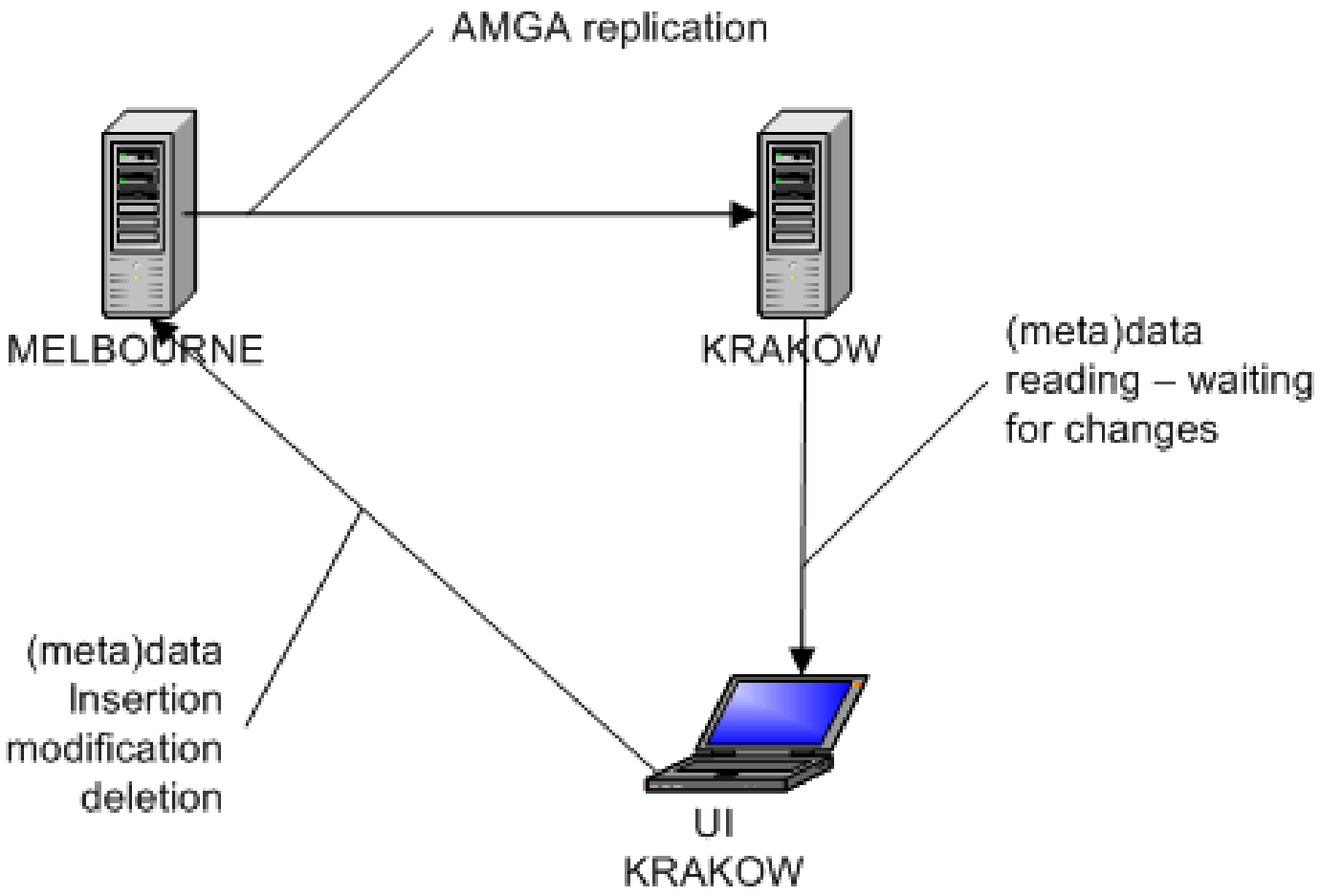
Latest problem

- No unit tests when writing code ☹
- Tests inspired by Tom
  - We have a problem – functions are failing!
- Technical issue – classes need to be redesigned for serialization
- Solutions
  - Adding missing serialization functions
  - Finding another multiprocessing mechanism

Bookkeeping using AMGA

- Why?
  - We know it
    - Configuration
    - API
  - It has built in-replication
- Why not?
  - It's mainly for metadata
  - No full synchronization, only master-slave replication
  - Maybe PostreSQL could be better?
- What we need to do?
  - Check if it fulfils our needs
  - Test it!

Test dataflow



Test procedure

- Procedure
  - Prepare „workspace”
  - Insert new entry and wait
  - Modify entry and wait
  - Delete entry and… wait
  - Do it many times and show results
- Test 1
  - Simple entry – one integer value
- Test 2
  - Complex entry – integer and varchars
- Test 3 – stress test
  - Complex entry (about 2.5 kB each) – parallel operations

Test results

- Data containing only one *int (seconds):*
  - new entry - `min: 2.71` **`avg: 2.97`** `max: 3.02`
  - value change - `min: 2.58` **`avg: 2.95`** `max: 3.01`
  - deletion - `min: 2.96` **`avg: 2.99`** `max: 3.03`
- Data containing *int, varchar(32), 2 varchar(128),2varchar(512) (seconds):*
  - new entry - `min: 3.01` **`avg: 3.16`** `max: 3.74`
  - value change - `min: 2.59` **`avg: 2.93`** `max: 3.02`
  - deletion - `min: 2.97` **`avg: 3.01`** `max: 3.03`
- Insertion of new entry: **1.5 –2.0 seconds**

Stress test results

- Insertion:
  - `min: 3.11` **`avg: 3.44`** `max: 3.73`
- Modification:
  - `min: 0.42` **`avg: 0.93`** `max: 1.97`
- Deletion:
  - `min: 0.82` **`avg: 1.12`** `max: 1.45`

About the same as non-stress (it took longer, but entries are biger now)

3 times faster? Maybe some external conditions changed. Keeping connections alive?

Issues

Parallel execution of processes (1 process – 1 test)

- Sometimes test pool couldn't go to the end
  - Most of processes succeeded, few hang
    - 100% utilization of all cores, massive memory usage
  - < 25 – no problems at all, all OK.
  - 25 – random
  - > 25 – always failing
  - Using faster slave machine was pushing up the limit! Even over 100 processes doing fine!
- 34 Postgres processes running while testing
  `postgres: arda metadata [local] idle`
- Possible reasons
  - Configuration of PostgreSQL and AMGA
  - Machine performance

What next?

- Solve multiprocessing issues
- Put things together! Make system work!