Speed up execution of

Django tests

Igor Davydenko
UA PyCon 2011

I am...

- Python developer at oDesk Professional Services
- Like write tests, but not only in TDD manner
- github.com/playpauseandstop
- djangonaut.blogspot.com
- djangonaut.posterous.com

Django tests

What I've used?

- Django 1.3.1
- SQLite 3.7.5
- PostgreSQL 9.1
- MySQL 5.5.15
- Did not try to speed up tests on other database backends like Oracle or NoSQL solutions

Test settings

- TEST_RUNNER
- DATABASES
- TEST_NAME
- TEST_USER
- TEST_CHARSET
- TEST_COLLATION
- TEST_DEPENDECIES
- TEST_MIRROR

Test settings

```
$ python manage.py test --settings=test_settings app ...
```

test_settings.py

```
from settings import *

DATABASES = { ... }
EMAIL_BACKEND = 'django.core.mail.backends.locmem.EmailBackend'

if 'south' in INSTALLED_APPS:
    INSTALLED_APPS = list(INSTALLED_APPS)
    INSTALLED_APPS.remove('south')

try:
    from local_test_settings import *
except ImportError:
    pass
```

Test project

- 2 test packages
- 34 test modules
- 621 tests
- near 11300 lines of code in test modules

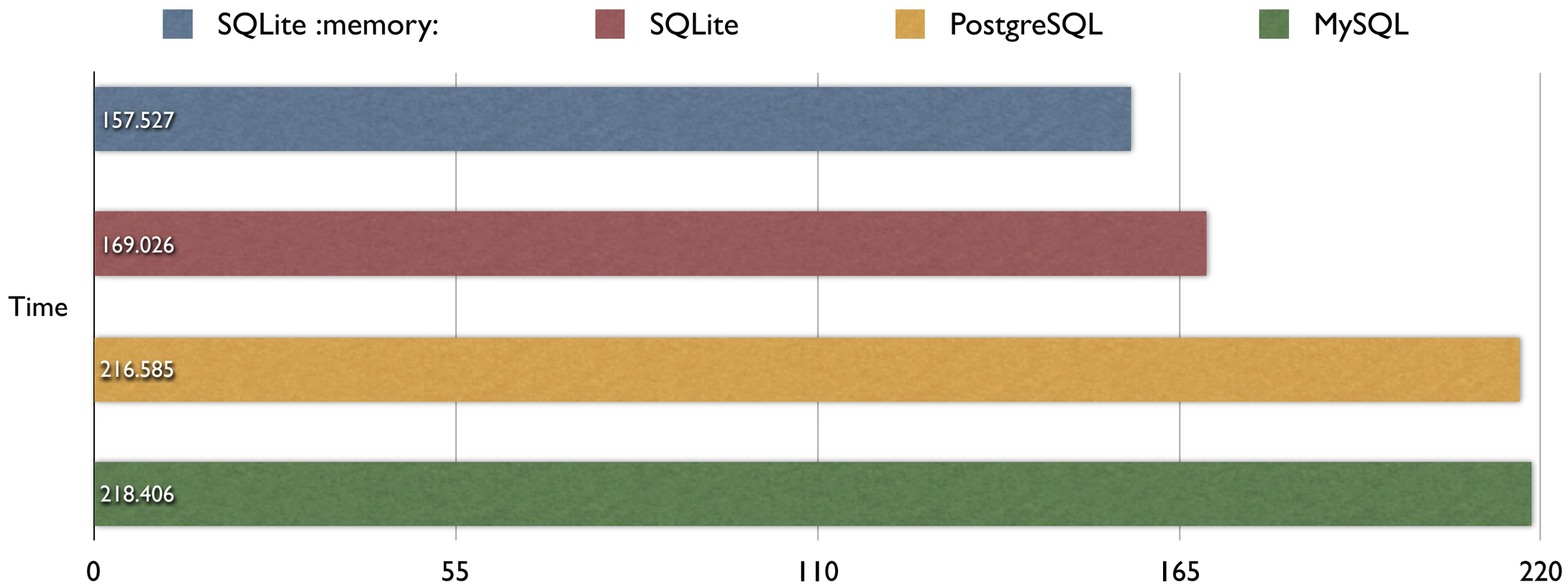
Test machine

- Intel Core 2 Duo E6300 1.86 GHz
- 4GB Ram
- 7200rpm HDD
- openSUSE 11.4 x86_64
- Python 2.7

First results

Database	TEST_NAME	Time
SQLite	**:memory:**	**157.527s**
SQLite	test_sqlite3.db	169.026s
PostgreSQL	test_postgresql	216.585s
MySQL	test_mysql	218.406s

First results



Note: *In-memory SQLite database is fastest solution for running tests out of box*

Speeding it up

1st Solution

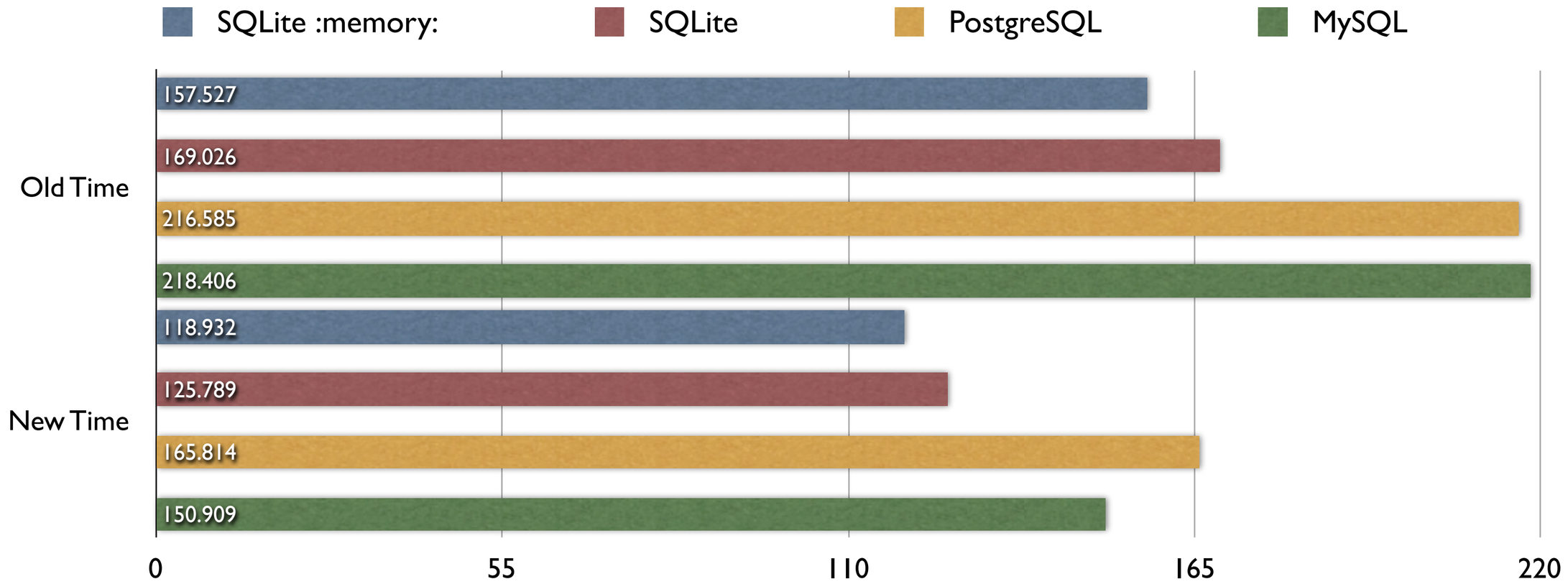
Obvious. Improve hardware

- *From Intel Core 2 Duo E6300 1.86 GHz to* Intel Core i5
- *From 4GB 667 GHz to* 4GB 1333 GHz DDR3
- *From 7200rpm HDD to* SSD
- *Do not forget to upgrade software too :) Tests in Django 1.3 run a* ***bit*** *faster than in 1.2 and a* ***lot*** *faster than in 1.1*

New results

Database	TEST_NAME	Time
SQLite	**:memory:**	**118.932s**
SQLite	test_sqlite3.db	125.789s
PostgreSQL	test_postgresql	165.814s
MySQL	test_mysql	150.909s

New results


SQLite :memory:
SQLite
PostgreSQL
MySQL
Old Time
New Time
157.527
169.026
216.585
218.406
118.932
125.789
165.814
150.909
0
55
110
165
220

2nd Solution

Run tests on internal machine

Jenkins continuous integration system

http://jenkins-ci.org/

```
# local machine
git push origin <branch>

# Jenkins machine
runtests.sh # or you should config how to run tests after each commit
```

Connect Jenkins and Django with **django-jenkins**

https://github.com/kmmbvnr/django-jenkins

2nd Solution

Run tests on internal machine

Local test server

Spend some money for new shiny hardware (Quad Core, SSD, etc)
and run tests there with Jenkins or git hook or whatever else.

3rd Solution

Obvious. Find for **fat** tests

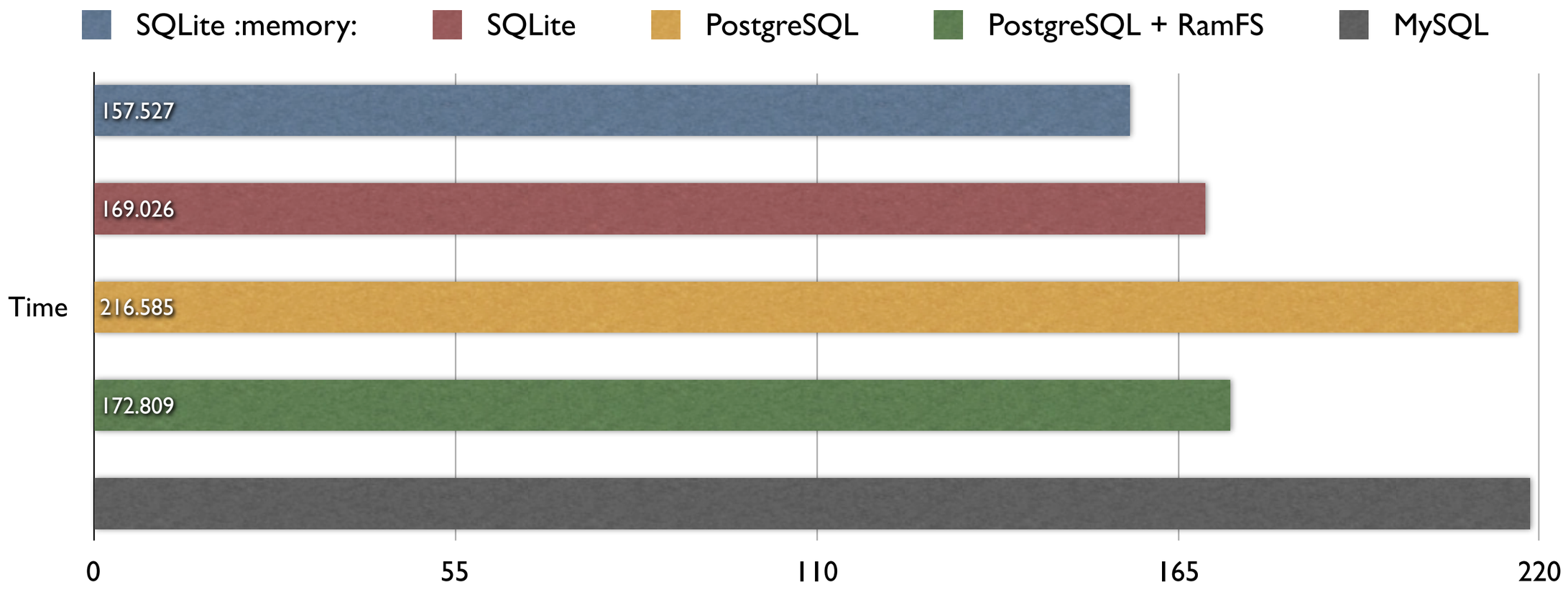
- Do not forget that tests is just a **Python** code
- All tips for speeding up Django ORM can be affected in tests too
- https://docs.djangoproject.com/en/1.3/topics/db/optimization/

4th Solution

PostgreSQL + RamFS *(tmpfs)*

- Mount a tmpfs
- Create second PostgreSQL instance on a different socket/port using this mount as data dir
- Tell Django to use it
- More info by click

Results

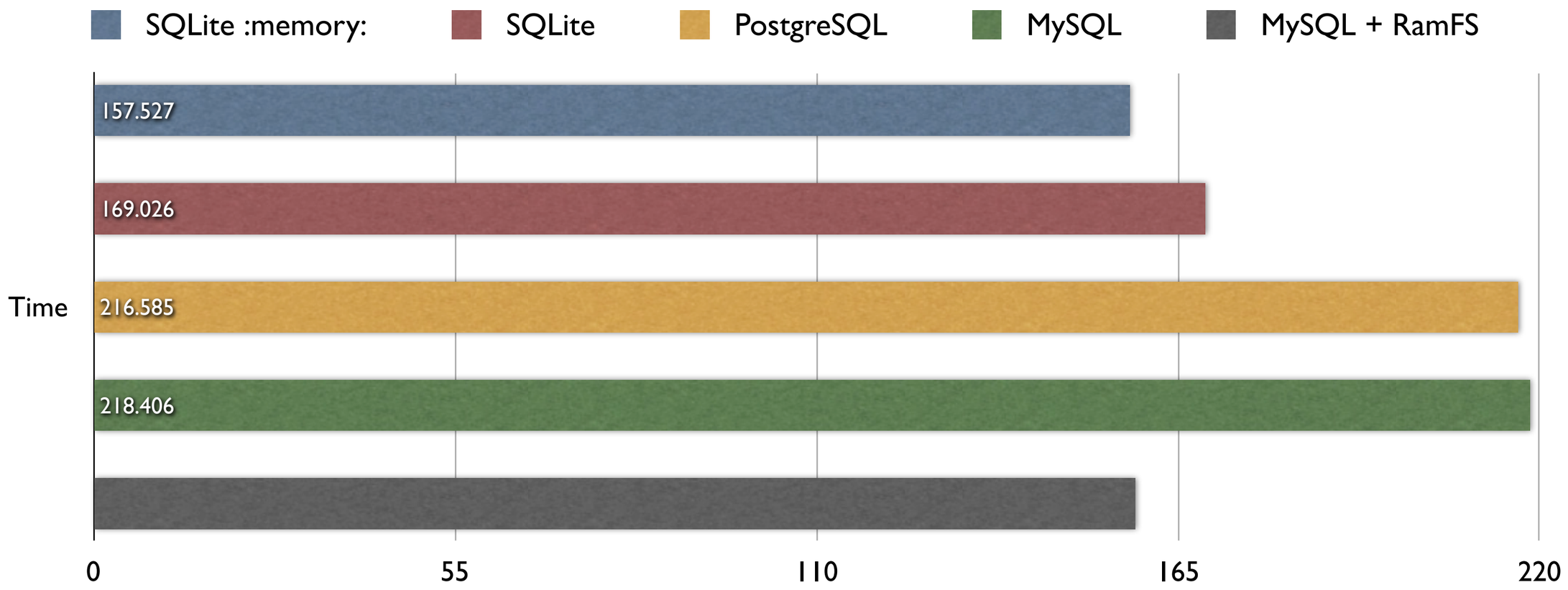

SQLite :memory:
SQLite
PostgreSQL
PostgreSQL + RamFS
MySQL
157.527
169.026
216.585
172.809
Time
0
55
110
165
220

4th Solution

MySQL + RamFS *(tmpfs)*

- Mount a tmpfs
- Create second MySQL instance on a different socket/port using this mount as data dir
- Tell Django to use it
- More info by click

Results


SQLite :memory:
SQLite
PostgreSQL
MySQL
MySQL + RamFS
Time
157.527
169.026
216.585
218.406
0
55
110
165
220

5th Solution

Multiprocessing

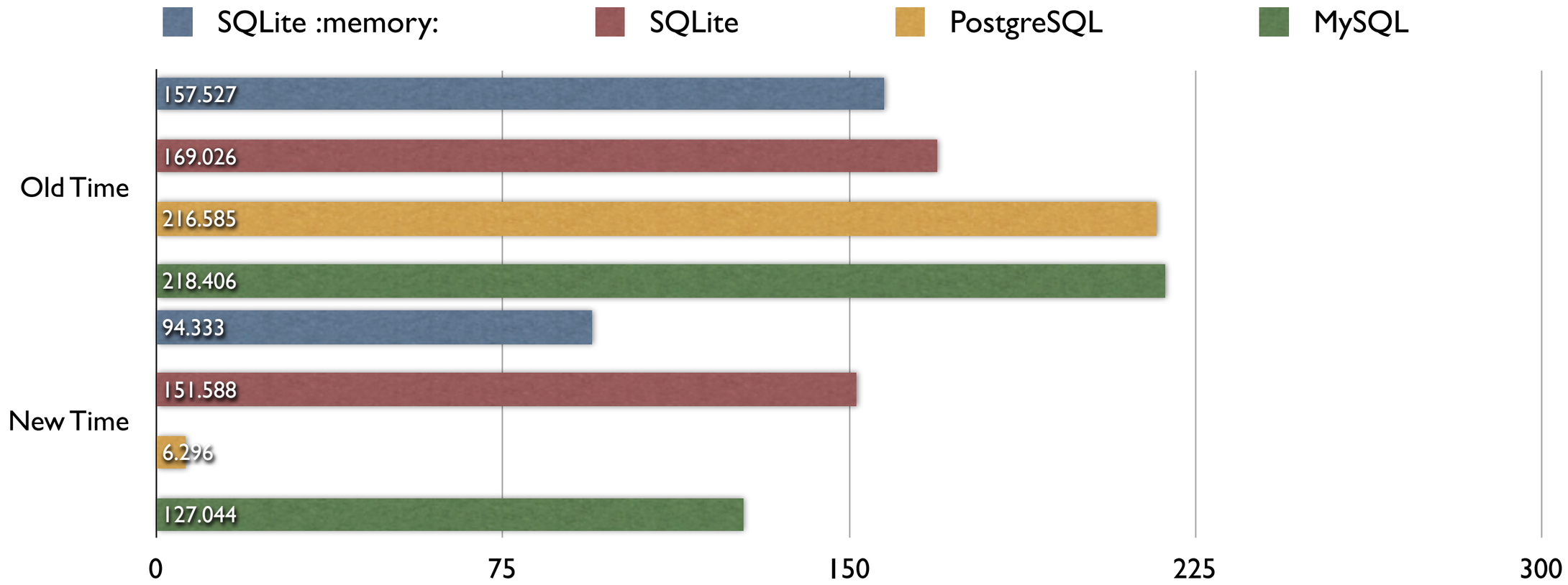
- pip install nose
- pip install django-nose
- Update test settings with:

```
TEST_RUNNER = ‘django_nose.NoseTestSuiteRunner’
NOSE_ARGS = [‘--processes’, ‘2’] # For dual-core CPU
```

New results

Database	TEST_NAME	Time
SQLite	**:memory:**	**94.333s**
SQLite	test_sqlite3.db	151.588s
PostgreSQL	test_postgresql	6.296s
MySQL	test_mysql	127.044s

New results


SQLite :memory:
SQLite
PostgreSQL
MySQL
Old Time
157.527
169.026
216.585
218.406
New Time
94.333
151.588
6.296
127.044
0
75
150
225
300

New problems

- It's totally mess with data, you don't know how things going on next test run
- "Database is locked" errors
- Needs only multi-core CPU
- Official BEWARE from nose documentation

6th Solution

Mix it up!

- The most obvious choice is to continue use SQLite in-memory database, but with multiprocessing
- Or, really, buy new MacBook Pro (that would be introduced on next week :))

Questions?