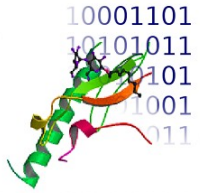
Baking RDKit on a Pi

- Tips and gotchas

Jan Holst Jensen
CEO, Biochemfusion



biochemfusion
- Enabling biochemformatics

Raspberry Pi

Image from http://www.raspberrypi.org/faqs

An ARM-based, networked, credit-card sized computer capable of running Linux. For $35!

RDKit on the Pi

- Really smooth sailing.
- Compiles *almost* out-of-the-box.
- We will look at
  - The needed OS and HW configuration tweaks.
  - (Minimal) source code corrections.
  - What performance you can expect.

Setup

- Using the standard Debian Wheezy distro.
- Add RDKit build prerequisites:
  - apt-get install the following packages
    - cmake flex bison python-imaging
    - libboost-dev libboost-regex-dev libboost-python-dev
- The python-numpy package is already installed in this Pi distro.

Configure and go...

```
pi@pi-dev ~/RDKit_2012_06_1 $ export RDBASE=/home/pi/RDKit_2012_06_1
pi@pi-dev ~/RDKit_2012_06_1 $ export LD_LIBRARY_PATH=$RDBASE/lib
pi@pi-dev ~/RDKit_2012_06_1 $ export PYTHONPATH=$RDBASE
pi@pi-dev ~/RDKit_2012_06_1 $ mkdir build
pi@pi-dev ~/RDKit_2012_06_1 $ cd build
pi@pi-dev ~/RDKit_2012_06_1/build $ cmake ..
-- The C compiler identification is GNU 4.6.3
-- The CXX compiler identification is GNU 4.6.3
...
...
-- Generating done
-- Build files have been written to: /home/pi/RDKit_2012_06_1/build
pi@pi-dev ~/RDKit_2012_06_1/build $ make
Scanning dependencies of target fastentropy
[  0%] Building CXX object External/cmim-1.0/CMakeFiles/fastentropy.dir/
[  1%] Building CXX object External/cmim-1.0/CMakeFiles/fastentropy.dir/
...
```

Now, get coffee, read a good book, rent a movie, clean the house...

Ouch! Compile crash

```
...
...
[  4%] Building CXX object Code/RDBoost/Wrap/CMakeFiles/rdBase.dir/RDBase.cpp.o
c++: internal compiler error: Killed (program cc1plus)
Please submit a full bug report,
with preprocessed source if appropriate.
See <file:///usr/share/doc/gcc-4.6/README.Bugs> for instructions.
make[2]: *** [Code/RDBoost/Wrap/CMakeFiles/rdBase.dir/RDBase.cpp.o] Error 4
make[1]: *** [Code/RDBoost/Wrap/CMakeFiles/rdBase.dir/all] Error 2
make: *** [all] Error 2
pi@pi-dev ~/RDKit_2012_06_1/build $
```

Oops – out of memory

Memory configuration

Low default swap space

```
root@pi-dev ~ $ free
...
Swap:        102396           0      102396
root@pi-dev ~ $
```

Add 512 M more for the compile session...

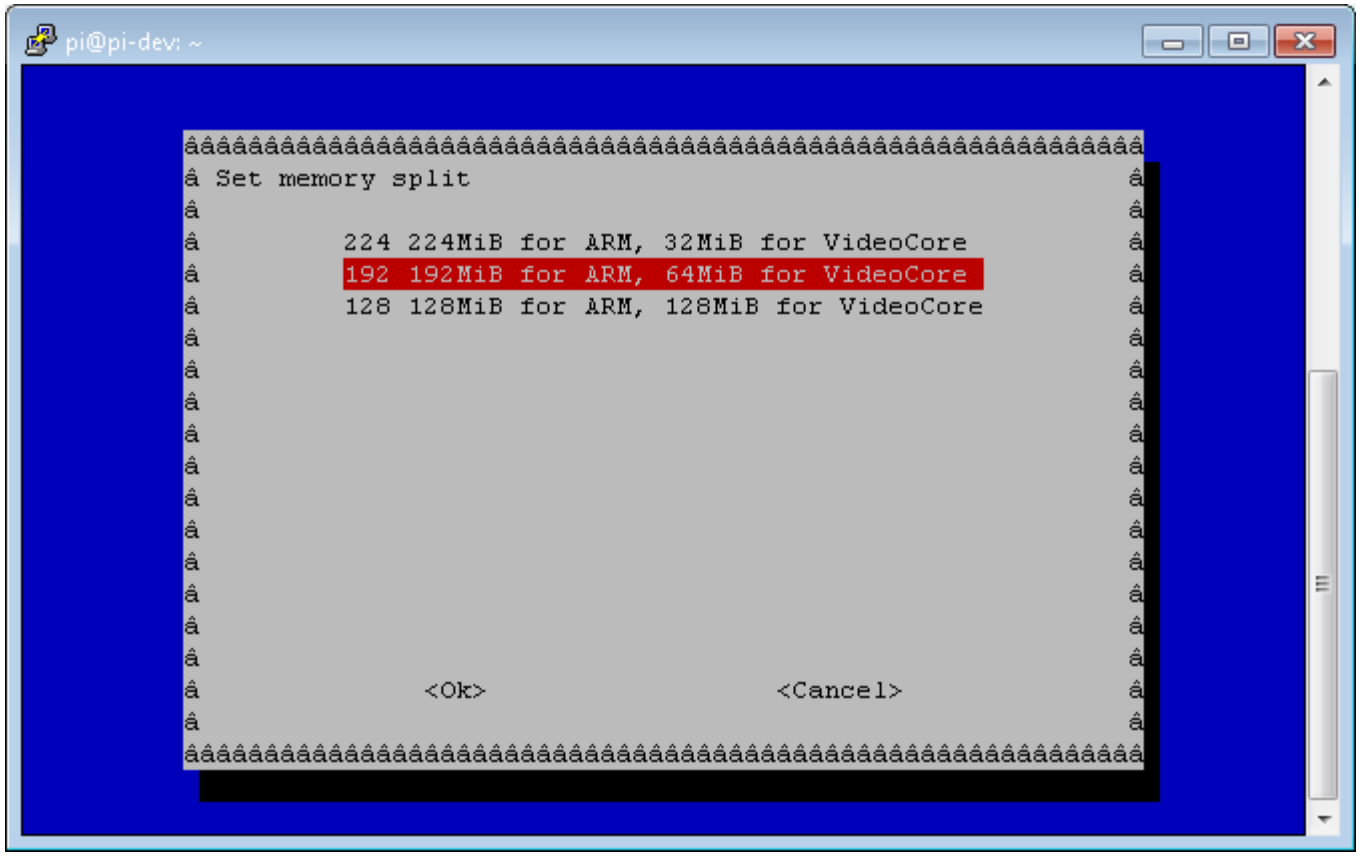
```
root@pi-dev:~# dd if=/dev/zero of=tmp-swapfile bs=1M count=512
512+0 records in
512+0 records out
536870912 bytes (537 MB) copied, 54.4397 s, 9.9 MB/s
root@pi-dev:~# mkswap tmp-swapfile
Setting up swapspace version 1, size = 524284 KiB
no label, UUID=689fbed2-bb8b-41d0-a228-1eea71705f18
root@pi-dev:~# swapon tmp-swapfile
root@pi-dev:~# free
...
Swap:        626680       15172      611508
root@pi-dev:~#
```

256 MB on-board RAM, but...

```
pi@pi-dev ~ $ free
             total       used       free
Mem:        188080     146852      41228
-/+ buffers/cache:      63568     124512
Swap:       626680       5480     621200
pi@pi-dev ~ $
```

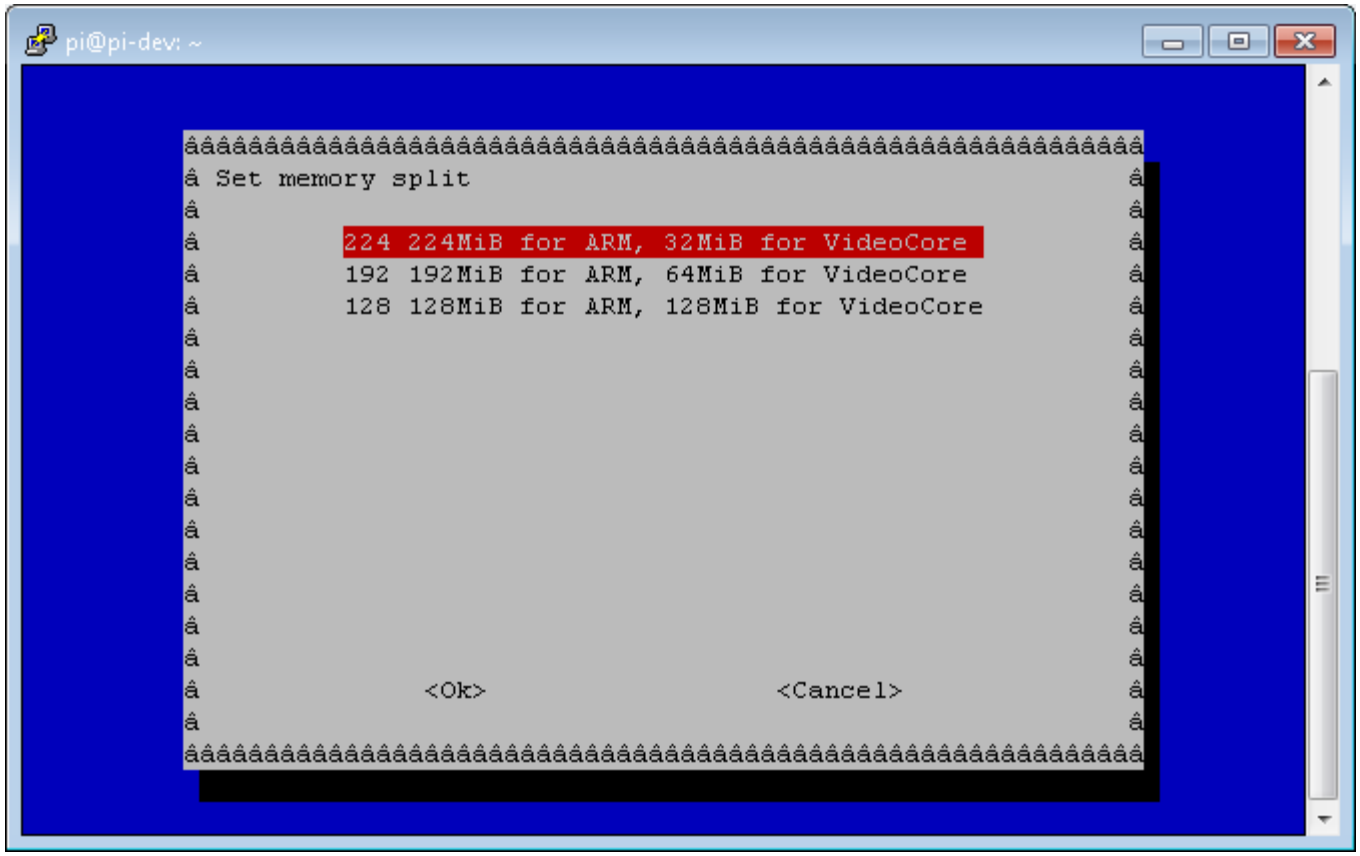
Who stole my RAM ?

Default configuration



Running the ‘raspi-config’ utility above

Console-only configuration



OK – then reboot

Much better

```
pi@pi-dev ~ $ free
             total       used       free
Mem:        220592      51020     169572
-/+ buffers/cache:      17720     202872
Swap:       626680          0     626680
pi@pi-dev ~ $
```

Compile warnings

```
[  2%] Building CXX object Code/RDGeneral/CMakeFiles/RDGeneral.dir/Dict.cpp.o
/tmp/ccgfzFB4.s: Assembler messages:
/tmp/ccgfzFB4.s:1525: Warning: swp{b} use is deprecated for this architecture
```

- Boost 1.49 issue generating suboptimal code for *smart_ptr*.
- See https://svn.boost.org/trac/boost/ticket/7141

Fixing boost 1.49

- In file

  `/usr/include/boost/smart_ptr/detail/sp_has_sync.hpp`

```
 #define BOOST_SP_HAS_SYNC

+ #if !defined(__GCC_HAVE_SYNC_COMPARE_AND_SWAP_4)
+
  #if defined( __arm__ )  || defined( __armel__ )
  #undef BOOST_SP_HAS_SYNC
  #endif

+ #endif // __GCC_HAVE_SYNC_COMPARE_AND_SWAP_4
+
  #if defined( __hppa ) || defined( __hppa__ )
  #undef BOOST_SP_HAS_SYNC
  #endif
```

Build completed (and tests too)

- Did you get that coffee ?!
  - 'time make' reports (*)

```
real    679m19.796s
user    641m45.310s
sys     12m36.180s
```

- For comparison:
  - Intel x86 Ubuntu VM running on a Core i5 2.4 GHz host
  - 'time make' reports

```
real    14m3.441s
user    13m24.326s
sys     0m31.790s
```

The Pi is almost 50 times slower – ouch.

(*) If you only configure the Pi with 188 M RAM instead of 220 M the build time rises to 750 mins.

PostgreSQL cartridge - build

- PostgreSQL 9.1 packages needed
  - postgresql-9.1 + postgresql-server-dev-9.1
- rdkit cartridge building
  - No issues at all.
    - cd $RDBASE/Code/PgSQL/rdkit/
    - make
    - sudo make install
  - Done ☺

PostgreSQL cartridge install

- Install into database

```
postgres@pi-dev:~$ psql -c 'CREATE EXTENSION rdkit'
CREATE EXTENSION
postgres@pi-dev:~$
```

- Edit /etc/postgresql/9.1/main/postgresql.conf

```
    #shared_preload_libraries = ''          # (change requi
+   shared_preload_libraries = 'rdkit'
```

- Restart database – done.

Performance

in real life, not compiling

Simple CPU-bound test

```
from rdkit import Chem
from rdkit.Chem import AllChem

for t in range(1, 100):
        m = Chem.MolFromSmiles('OC(=O)C1CCC(CC(=O)O)CC1')
        AllChem.EmbedMultipleConfs(m, 10)
        for i in range(m.GetNumConformers()):
                AllChem.UFFOptimizeMolecule(m, confId=i)
        m = None
```

Pi	Ubuntu x86
89 secs	3.3 secs

27 times slower

Database – load molecules

- Load 50 000 random PubChem molecules

```
postgres=# create table pubchem_mols (pubchem_no text, molecule mol);
CREATE TABLE
postgres=# create index molidx on pubchem_mols using gist(molecule);
CREATE INDEX
postgres=# \q

time psql -f /home/postgres/pubchem-smiles.sql > /dev/null
```

- Intel x86 VM, postgres 9.1.6
  - 3 mins 38 secs
- Raspberry Pi, postgres 9.1.5
  - 6 hours 1 min (x 100)

Database – search performance

- `select count(*) from pubchem_mols where molecule @> <smiles>;`

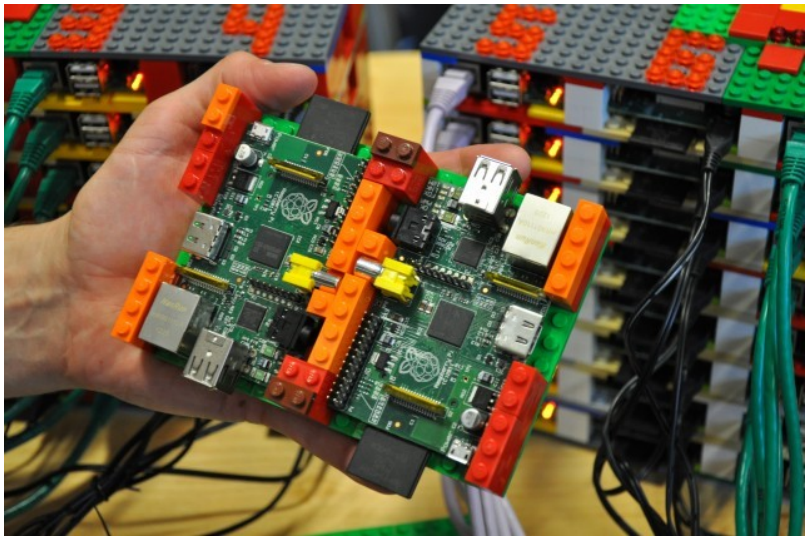
- <smiles> =
  - Triazine: 'c1ncncn1‘
    - Intel x86 VM 0.6 secs
    - Raspberry Pi 16.2 secs (x 27)
  - Coumarine: 'O=C1OC2=CC=CC=C2C=C1‘
    - Intel x86 VM 0.35 secs
    - Raspberry Pi 9.4 secs (x 27)

In conclusion

- Quite easy to get RDKit running on the Pi
  - Once you know a couple of tweaks
- No speed daemon!
  - But it is OK for non-heavy stuff
    - e.g. departmental chemical database
    - actually – perfect for testing! “If it runs well on the Pi...”
  - and it uses less than 3W !

Need more performance ?

- http://arstechnica.com/information-technology/2012/09/university-builds-cheap-supercomputer-with-raspberry-pi-and-legos/



Latest – new Debian image

- Per 2012-09-18 there is a new Debian image
- New firmware allows easy (and safe!) overclocking
  - "Turbo" mode – 1 GHz clock on-demand
  - Expect performance that is 50% to 100% better
    - General performance up 50% +
    - Build times almost halved – g++ upgraded too