science + computing

A Bull Group Company

Post-Mortem Debugging with Heap-Dumps

Anselm Kruis | EuroPython 2014

science + computing ag

IT-Services for Complex Computing Environments

Tübingen | München | Berlin | Düsseldorf

Post-Mortem Debugging with Heap-Dumps

science + computing
A Bull Group Company

Outline

- About me
- Outset
- Previous works
- The project pyheapdump
- Live demonstration / usage
- How does it work? Challenges.
- Future goals
- Q & A

© 2014 science + computing ag

Who and Why

science + computing
A Bull Group Company

Who

Name: Anselm Kruis

Profession: Senior Architect at science + computing ag

Location: Munich

Why

- Python is fun, EuroPython is fun
- Let's push the limits.
- And make programs usable: fight bugs.

© 2014 science + computing ag

Outset

science + computing
A Bull Group Company

- Every serious computer program is buggy.
- Some program failures occur very infrequent or are hard to reproduce.
- They are best analysed post-mortem.
- Most common approach:
  - Create a core-dump
  - Analyse it later
- Python lacks (usable) core dumps

→ Chance for a cool little project:

https://pypi.python.org/pypi/pyheapdump

© 2014 science + computing ag

Previous Work

science + computing
A Bull Group Company

Conventional OS-level memory/register dumps

- The origins date back into the time of magnetic core memory.
  - SHARE Operating System, IBM 1959, debug macro “CORE” [1]
- Today almost every operating system supports dumps under various names. Most important:
  - Core-dumps on UNIX, Minidumps on MS-Windows
- People used OS-level dumps to analyse „interpreted“ programs running within a native-code interpreter-process.
  - Various Python related examples with mixed results [2, 3, 4, 5].

© 2014 science + computing ag

Previous Work 2

science + computing
A Bull Group Company

OS independent dump methods

- For some „interpreted“ languages OS-independent dump methods were developed [6].
  - A prominent example are Java heap dumps [7].

- Python related
  - In 2012 Eli Finer released “pydump” [8]
    - Catch an exception
    - Pickle the traceback
    - Use pdb.post_mortem to analyse the unpickled traceback
    - Pydump fails on pickling / unpickling errors.

© 2014 science + computing ag

Pyheapdump

science + computing
A Bull Group Company

- Name
  - Pydump was already used
  - analogous to Java heap dumps
- Status
  - Experimental work
  - Currently 2.7 only, porting to 3.x seems possible
- Building Blocks
  - Exception handling code
  - Dump creation
  - Debugger glue code

I copied some code from Eli Finers pydump module.

© 2014 science + computing ag

Demonstration

science + computing
A Bull Group Company

- Situation:
  - You installed a little Python game for your <*partner, kids, customer ...*>. She/he complains about infrequent crashes.
  - Now you have to catch the bug...

Note: I purposely introduced the bug into the demo program (BlockFortress). The upstream version is OK.

© 2014 science + computing ag

Usage

science + computing
A Bull Group Company

Basic application of pyheapdump

1. Set up an exception handler
   - Usually dump_on_unhandled_exceptions(...)
   - Low level functions are also available
   - See http://pythonhosted.org//pyheapdump/pyheapdump.html
2. Instruct your customer / operator to send you any python_heap*.dump files.
3. Wait …
4. Analyse the exception using a common debugger.

© 2014 science + computing ag

How does it work?

science + computing
A Bull Group Company

I'll try to answer simpler questions:

- What's the content of a python heap dump?
- How does pyheapdump create this content?
  - Basic idea
  - Challenges
- Debugger support

© 2014 science + computing ag

How does it work?

science + computing
A Bull Group Company

What's the content of a python heap dump?

- It is a MIME message.
  - Informational header lines
  - Binary content
- The binary content is a compressed pickle of a dictionary
  - The trace back of an exception
  - Stack frames of selected / all Python-threads
    (Stackless Python: selected / all tasklets)
  - All objects in the transitive closure of the frames / tasklets
  - Optionally: relevant Python sources
  - Other interesting objects: process id, path module, thread ids

© 2014 science + computing ag

How does it work?

science + computing
A Bull Group Company

How does pyheapdump create this content?

- Basic idea:
  - create a dict with the content and pickle it
- Challenges
  - You can't pickle <*put the name of you favorite class here*>
  - What about multithreading?

© 2014 science + computing ag

Challenge: Pickling arbitrary objects

science + computing
A Bull Group Company

- You all know pickling: https://docs.python.org/2/library/pickle.html
  - It's a data format
  - Standard implementation: pickle, cPickle
    - serialise data
    - portable between different Python versions
    - fast
  - Other implementation: sPickle
    - serialise all “well behaved” objects
    - not portable between Python versions
    - slow
  - Pyheapdump builds on sPickle
    - Adds fault tolerant pickling / unpickling

© 2014 science + computing ag

Challenge: Multithreading

science + computing
A Bull Group Company

How to make a snapshot of all objects in the presence of other threads?

- A perfect solution is not possible.
- Best efford solution:
  - You can block other threads as long as you don't release the GIL

    sys.setcheckinterval(sys.maxint)

    (Stackless: atomic context manager)
  - Pickling might release the GIL
  - Solution: shallow copy frames, then pickle the copies

© 2014 science + computing ag

Debugger Support

science + computing
A Bull Group Company

- Pdb and Pydevd already support post_mortem debugging.
  - Pdb has a nice API method post_mortem(traceback)
  - Pydev requires some hacking
- Pydevd supports inspection of additional stack-frames that do not belong to a thread: “custom frames”
  - Originally invented to inspect Stackless tasklets
  - I reuse it for unpickled frames
- Pydevd should add an API for advanced debugger features like
  - post-mortem debugging
  - adding custom frames
  - print to the debugger console

© 2014 science + computing ag

Future goals

science + computing
A Bull Group Company

- Gain experience
  - Up to now I got very few heap dumps caused by real bugs. :-)
  - Open questions:
    - Memory usage.
    - Reliability
    - Information security
- Propose debugger APIs, create patches for pydevd
- Support Python 3.x

© 2014 science + computing ag

Acknowledgements

Thanks to

- My boss Arno Steitz

  for approving the publication of code and know how
- My colleague Tanja Huthmacher

  for testing
- My spouse Esther

  for many many hours of patience ...

© 2014 science + computing ag

References

science + computing
A Bull Group Company

References

(1) C. E. Homan (IBM), G. F. Swindle (SDC), 1959-10-13, Programmers Manual for the SHARE OPERATING SYSTEM, Preliminary Version, Section 01.04.01 Debugging, PDF http://www.piercefuller.com/scan/sos59.pdf?id=sos59

(2)The Python Wiki, 2014-05-31, *DebuggingWithGdb* at https://wiki.python.org/moin/DebuggingWithGdb

(3) Brian Curtin, 2011-09-29: *minidumper - Python crash dumps on Windows*, blog at http://blog.briancurtin.com/posts/20110929minidumper-python-crash-dumps-on-windows.html

(4) David Malcolm, Fedora Feature, 2010-04-06: *Easier Python Debugging* at http://fedoraproject.org/wiki/Features/EasierPythonDebugging

(5) Andraz Tori, Python, 2011-01-16: *gdb and a very large core dump*, blog at http://www.zemanta.com/blog/python-gdb-large-core-dump/

(6) David Pacheco, ACM Queue - Programming Languages Volume 9 Issue 10, October 2011: *Postmortem Debugging in Dynamic Environments*, PDF http://dl.acm.org/ft_gateway.cfm?id=2039361&ftid=1050739&dwn=1&CFID=290171300&CFTOKEN=95099236

(7) Chris Bailey, Andrew Johnson, Kevin Grigorenko, IBM developerWorks, 2011-03-15: *Debugging from dumps - Diagnose more than memory leaks with Memory Analyzer*, PDF http://www.ibm.com/developerworks/library/j-memoryanalyzer/j-memoryanalyzer-pdf.pdf

(8) Eli Finer, Github-Project, 2012: *pydump* at https://github.com/gooli/pydump

(9) Anselm Kruis, EuroPython 2011: *Advanced Pickling with Stackless Python and sPickle*, archived talk at https://ep2013.europython.eu/conference/talks/advanced-pickling-with-stackless-python-and-spickle

(10) Fabio Zadrozny, 2013-12-12: *PyDev 3.1.0 released*, blog at http://pydev.blogspot.de/2013/12/pydev-310-released.html

© 2014 science + computing ag

science + computing

A Bull Group Company

Many thanks for your kind attention.

Anselm Kruis
science + computing ag
www.science-computing.de

Phone +49-7071-9457-0
info@science-computing.de