# You can't spell KABOOM without OOM

Anthony Towns

# Introduction

# What is this talk about?

Debugging system problems

- A walkthrough of a real problem
- Some strategies for understanding the problem
- Some tactics for dealing with “out of memory” problems

# Koji

- Build system used in Fedora
- Also used in Red Hat

- Database (postgresql)
- Hub (XMLRPC, python, apache)
- Web service (python, apache)
- Workers (mock, rpmbuild)

# Memory Usage (Background)

- Memory is important!
- NFS, disk, mmap, swap are all great, but not good enough
- OOM killer

# The Problem

- In August/September the koji hub had to be rebooted every few days.

- Symptoms:
  - Interactive requests started getting laggy
  - We got Nagios alerts because it was unresponsive to network requests
  - OOM killer messages in logs/console

# Easy Solution

# Basic Fix

- Sysadmins get a Nagios alert
- They reboot the system
- Everything works again!

# Basic Fix – Why it works

- “Stateless” server process
- Independent database as state store
- Network file system
- Independent workers

# Basic Fix – Why it doesn't work

- Not entirely stateless: workers need to be restarted too
- Sysadmins don't like getting Nagios alerts
  - ...especially at 3am
  - ...especially every other day

# Fixing the bug

# Past memory problems in koji

- Some OOM problems a few months earlier
- “List all history for everything forever”

- Results:
  - Nagios alerts
  - setrlimit() memory use enforcement
  - Loss of trust in code reliability

# Bug in koji?

- Could be an actual bug
- Could be overuse

- Enable debug logging
- Add more logging
- Troll through apache usage logs

# Long term memory leak?

- Maybe it's not a specific call, but a gradual memory leak of some sort?
- mod_python isn't great at letting memory get reclaimed

- Upgrade to RHEL 6 with newer mod_python ?
- Convert to wsgi ?
- Reduce apache MaxRequestsPerChild ?
- Python memory management debugging ?

# Number of processes?

- But anyway, setrlimit() should mean it can't be any single process
- avg_mem_per_process * n_process < mem
- Put a limit on n_process as well as memory usage?
- Reduce MaxClients in apache
- (But this can impact users)

# Test script (“crash_koji.py”)

- Open *n* sockets to the hub
- Write enough of the request to each socket to cause apache to assign a process to the connection
- Sleep for a bit
- Send the rest of the request to each socket
- Get the result from each socket
- Report how many of the requests succeeded

# Just give it more memory?

- Move the VM
- Double the RAM
- Update DNS

- ...

- Watch it keep crashing

# First Principles Debugging

# Debugging with Tuz


Need some debugging help, eh?

http://www.flickr.com/photos/stibbons/3208705403/

# Debugging with Tuz


So you're running out of memory? Are you sure?

# Debugging with Tuz


Why does the OOM killer happen?

# Debugging with Tuz


How does the kernel track memory?

# Debugging with Tuz


Can we monitor memory use by processes?

# Debugging with Tuz


How about kernel memory structures?

# Debugging with Tuz


Then let's do that!

# Debugging with Tuz


How'd it go?

# Debugging with Tuz


If it's not any of the processes, how about the kernel?

<mikeb> aj: is that 2.5G in nfs_inode_cache ?

# Debugging with Tuz


Well there you go. Now get me another beer.

# A Kernel Bug?

# A Kernel *Regression* in *RHEL 5*?

# Evidence for:

- It's in /proc/slabinfo, of course it's the kernel
- The problem only started when we updated to RHEL 5.7

# Evidence against:

- Plenty of other RHEL 5.7 systems work fine – internally and at customers
- Every other time a developer has tried fobbing their own bug off on the kernel or compiler or a library

# Verdict:

- Hung jury:

- Sysadmins think it's koji's fault
- koji developers think it's the kernel's fault
- Kernel guys tell us to give them an Oops message before they can form an opinion

# “Double jeopardy”

- No rule against it when debugging!
- So gather some more evidence and re-indict!

# Logging kernel behaviour

```
while true; do
    date; free; ps aux;
    cat /proc/slabinfo;
    echo ----;
    sleep 1;
done >> ~/memory-use.log
```

# Logging kernel behaviour
## (when your machine is out of memory)

- Can't run programs –

  they need to be reloaded from disk,

  which needs memory

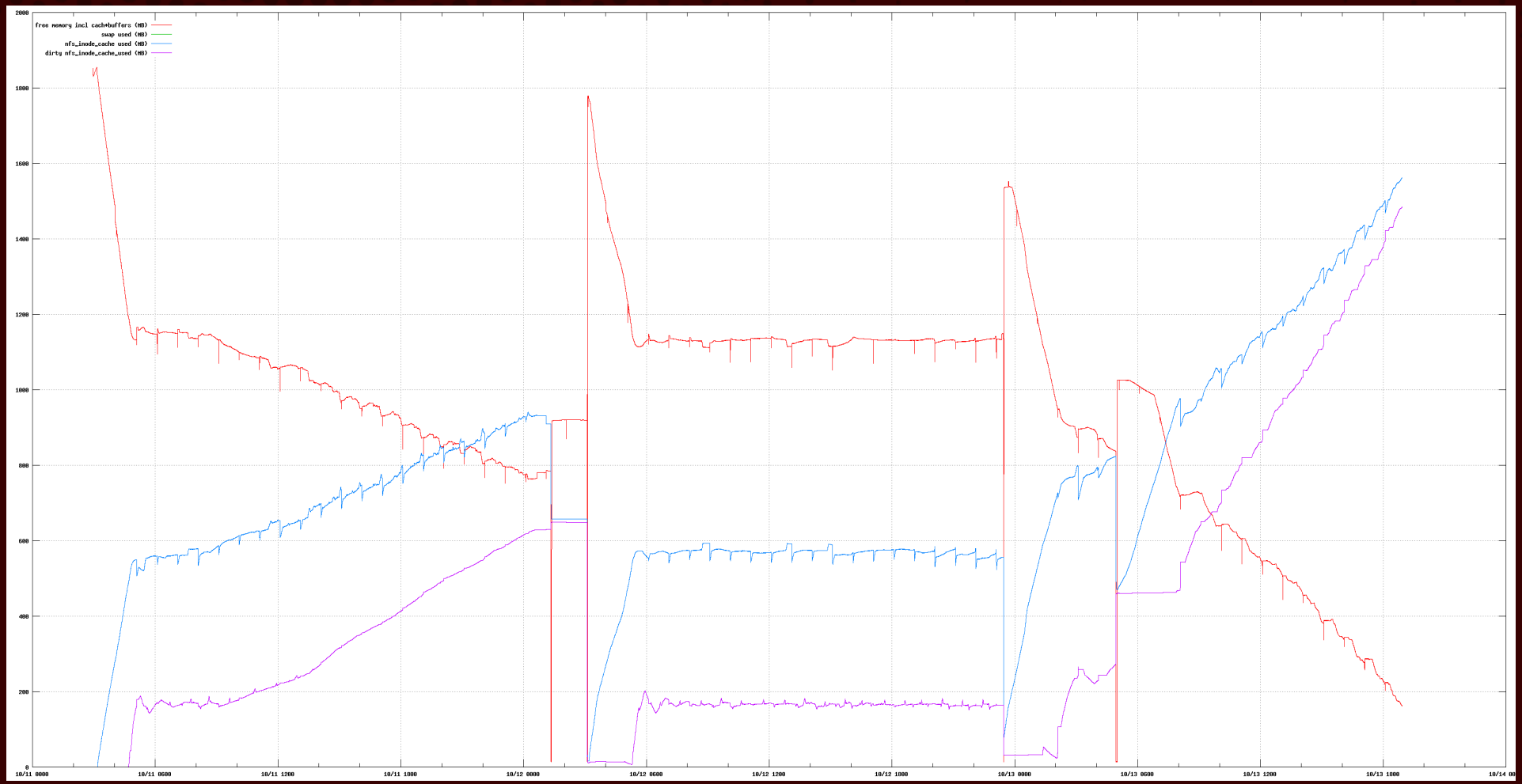
  but there isn't any

  also heavy I/O contention with swap

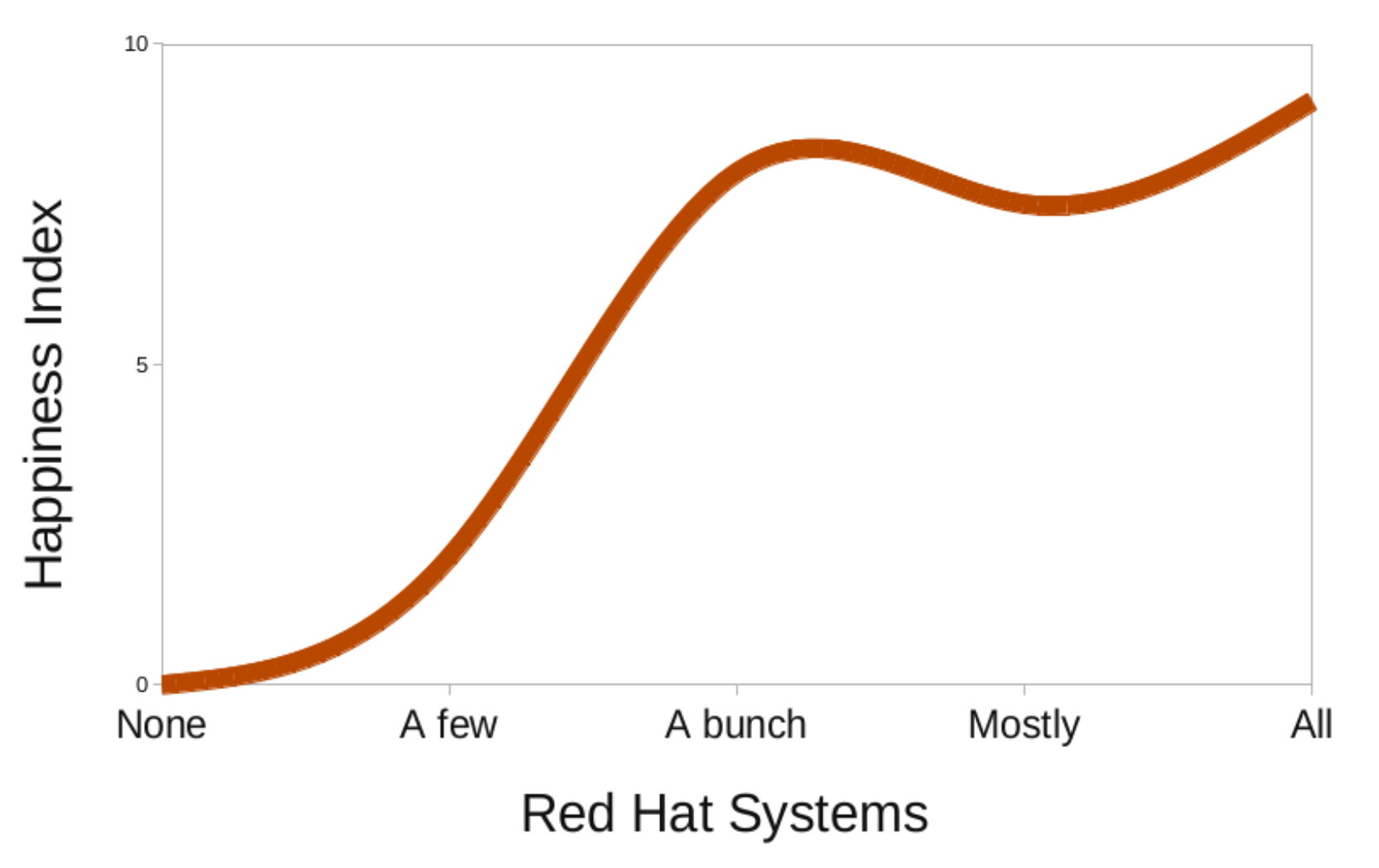
# monmemuse.py

- So write your own script
- Look in /proc/meminfo for “free”
- Look in /proc/12345/ for statm and cmdline for “ps aux”
- Look In /proc/slabinfo for kernel stuff
- Print out anything that's using significant memory, every few seconds
- “for I in range(100): time.sleep(0.1)”

# Analyse the data



# Pictures are persuasive!



# Conclusion

# Real fix?

- File a bugzilla
- Get your manager to ping Linux NFS developers
- (Or get your Red Hat support rep to do both those for you :)
- Install the update from the next RHEL release

# Test cases

- “building all Red Hat products” is not a very useful test case
- Managed to reproduce it by hand by cp'ing trees around to simulate regeneration of maven repositories
- (Filled up a shared NFS volume while doing this, ooops)
- Wrote a simulator (output of find, simulate a cp but with essentially random content)
- Clever kernel devs realise all that's needed is

```
i=0; while true; do
```

# Success!