



ellexus

Application-Level Debugging and Profiling: Gaps in the Tool Ecosystem

Dr Rosemary Francis, Ellexus

For years instruction-level debuggers and profilers have improved in leaps and bounds. Similarly, system-level and network monitoring tools are increasingly ubiquitous. There is, however, a gulf between the two worlds which is widening as more and more processes and applications share system resources.

In this talk I will present the concepts of application-level tracing and what kind of information is available. I will run through the techniques for combining run-time data from different levels before moving onto some case studies. In particular I will present how a lack of application-level monitoring can lead to file-system bottlenecks and undetected network issues in distributed Linux HPC IT systems.



ellexus

Application-level Tracing

Instruction-level
code debugging



?



System-level monitoring
of physical resources





ellexus

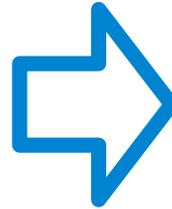
Who am I?

PhD in Semiconductor Architecture
and Electronic Design Automation
(CAD)

Embedded Software and
Digital Design

Too much time spent installing and
configuring the CAD tools.

Too much time reading scripts.



Technical Director, Ellexus Ltd

Specialists in:

- Tracing Linux applications
- Profiling application file IO

We work out why your application
doesn't work where it should.

We take the guess-work out of build
issues and installation problems.



ellexus

What am I going to talk about?

Profiling and debugging technologies

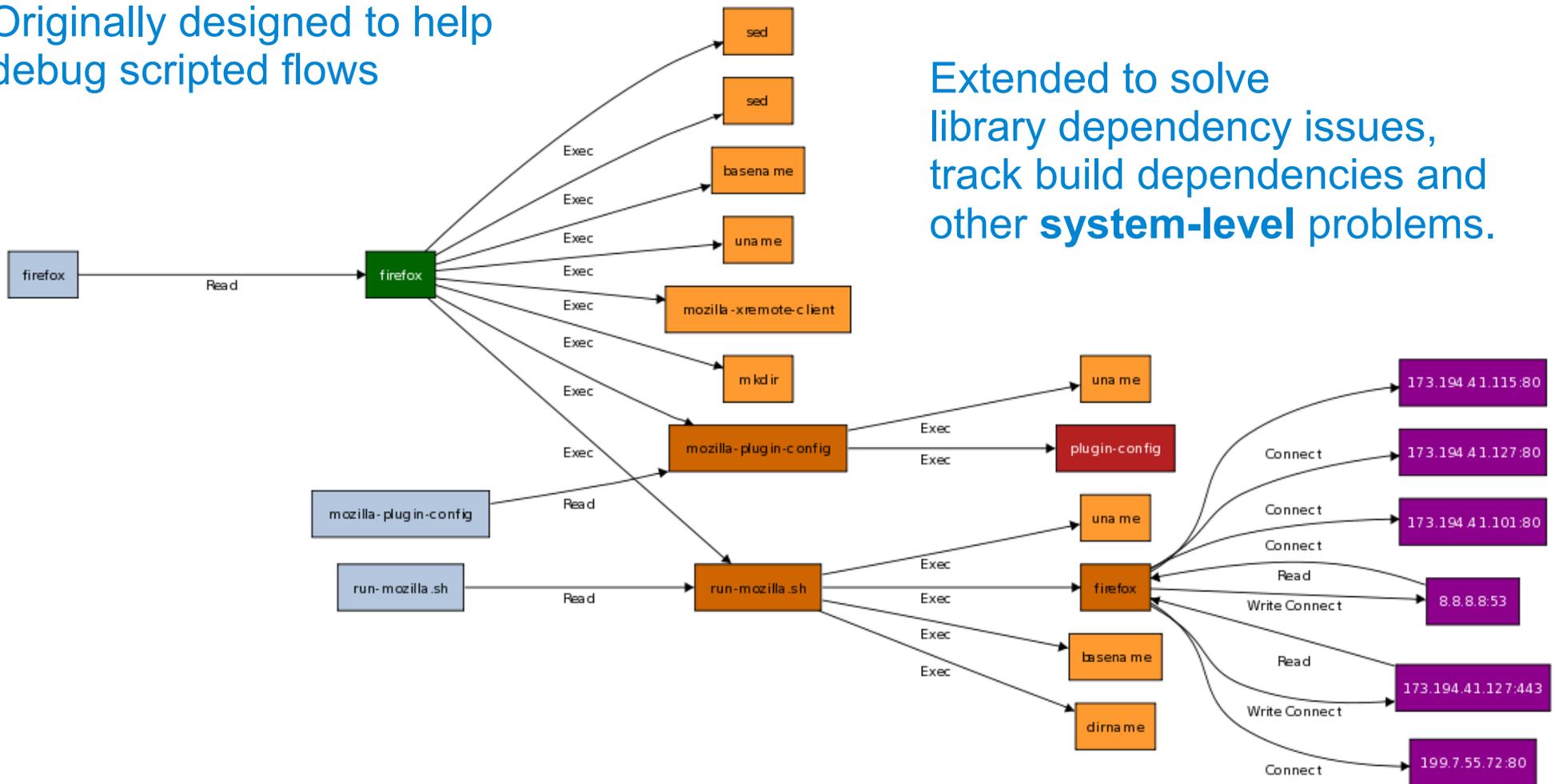
Our tools and how they work

Gaps in the tooling ecosystem

What the future might look like

Breeze

Originally designed to help debug scripted flows



Extended to solve library dependency issues, track build dependencies and other **system-level** problems.

Breeze lets you compare applications

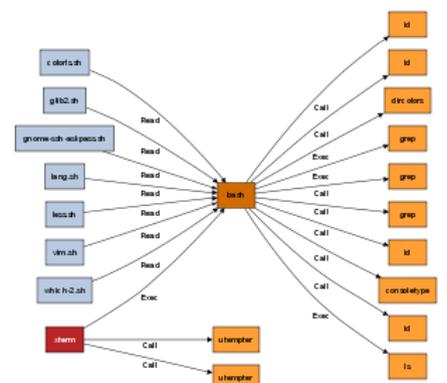
Ellexus - Breeze - Advanced Script Development

File Edit Trace Bookmarks Views EDA Help

Select run: 25 - xterm Debugger State: Done

Graph View

Graph 25: xterm



Commandline (bash)

```

/home/rosemary>
Graph View: Node View:
  
```

Compare Two Runs

This dialog lets you compare two runs. Step one will pair programs with matching names in the call hierarchy. Steps two and three let you compare the program pairs in more detail, looking at specific program properties.

Select run: 23 - xterm Select run: 25 - xterm

Select root node

Search entire trace

- /usr/bin/xterm - pid_29035 pid_29047
 - bash pid_29047 pid_29103 pid_29089 pid_29085 pid_29083 p
 - consoletype pid_29086
 - dircolors pid_29063
 - firefox pid_29103 pid_29136 pid_29137 pid_29128 pid_291
 - basename pid_29106
 - mkdir pid_29126

- id pid_29243
- id pid_29247
- id pid_29268
- id pid_29275
- ls pid_29289
- utempter pid_29235
- utempter pid_29298

Step 1:
Choose two runs and click to pair matching programs:
Compare the program trees of each run. Programs with matching executable names will be paired.

Step 2:
Check program pairs by property:
Breeze will look for differences in the program pairs. Program pairs that differ in the property chosen will be highlighted.

Step 3:
Choose a program pair to diff by property:
Breeze will display property differences for the selected program pair.

Pair program selection automatically

Program Properties

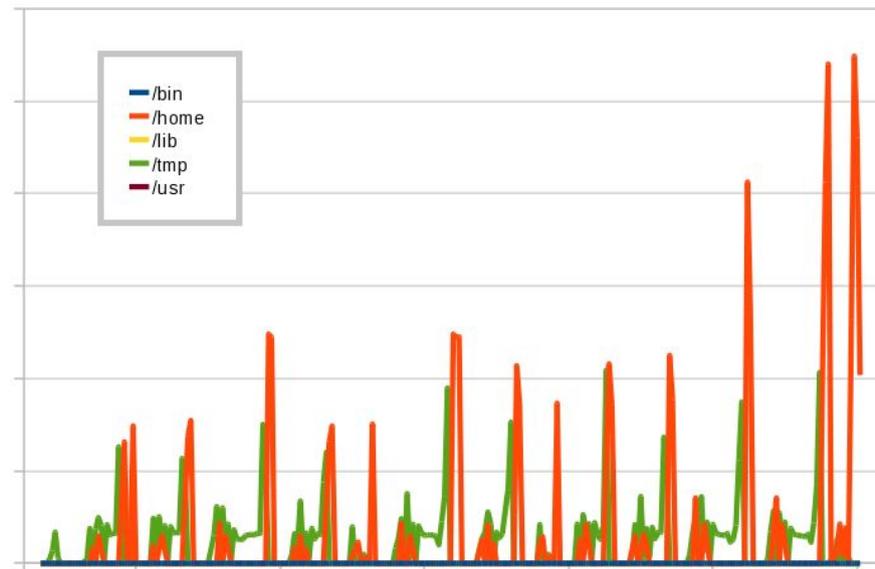
- Program args
- Program environment
- Environment change
- File system reads
- File system writes
- Network connections
- System Libraries
- Program actions over time

You can see what is different about a particular machine or user setting when it runs.



ellexus

Breeze Profiling File IO



In 2012 we added file IO profiling for remote file systems.

We are increasingly being asked to solve file system, networking, power and memory issues.

Common factor: matching physical resource use with application stats



ellexus

What else is available?

How does it work?



ellexus

Instruction-level Debuggers

GDB swaps machine instructions for special 'breakpoint' instructions

Valgind rewrites the memory allocation code of your program



Lets you see the program in instruction-level detail

Difficult to get an overview

Measurement can change application behaviour

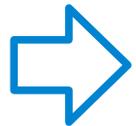


ellexus

New Instruction-level Debuggers

UNDO takes snapshots so you can run your program backwards

Allinea DDT can debug parallel code across many machines



Debug symbols match the instructions to the code

Research into debug symbols continues to improve

Focus is on embedded, but what is embedded?



ellexus

Application-level Debuggers: ptrace

Strace and Ltrace show system and library calls into the linux kernel

Both use a kernel facility called ptrace

They can solve some problems, but don't cope well with large applications



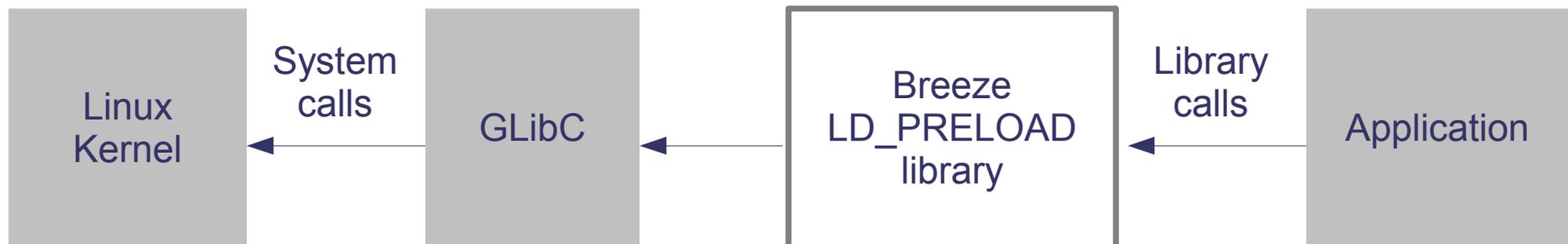


ellexus

Application-level Debuggers: LD_PRELOAD

Breeze and others like it use LD_PRELOAD instead of ptrace
This is a technique that involves overriding library functions

Provides a powerful interface that can trace and control the application
Can easily be combined with other profiling data





ellexus

LD_PRELOAD

Override library functions by matching the function signature

Use the `dlsym` function to get the next loaded library function with that name

```
int read(int fd, void *buf, int count)
{
    int (*next_read)(int fd, void *buf, int count)
    next_read = dlsym(RTLD_NEXT, "read")

    breeze_print(getpid(), "read", "%d", fd);

    return next_read(fd, buf, count);
}
```



ellexus

Multi-machine LD_PRELOAD

Override exec() and re-write command arguments to modify remote jobs

```
ssh user@host ls -l /tmp
```



```
ssh user@host -o SendEnv=LD_PRELOAD ls -l /tmp
```



ellexus

Switching file systems with LD_PRELOAD

Similar techniques can swap file systems, or suspend and resume jobs

```
int open(const char *pathname, int flags)
{
    ...

    char *alt_pathname;
    asprintf(&alt_pathname, "%s/%s", "/alt", pathname);

    return next_open(alt_pathname, flags);
}
```

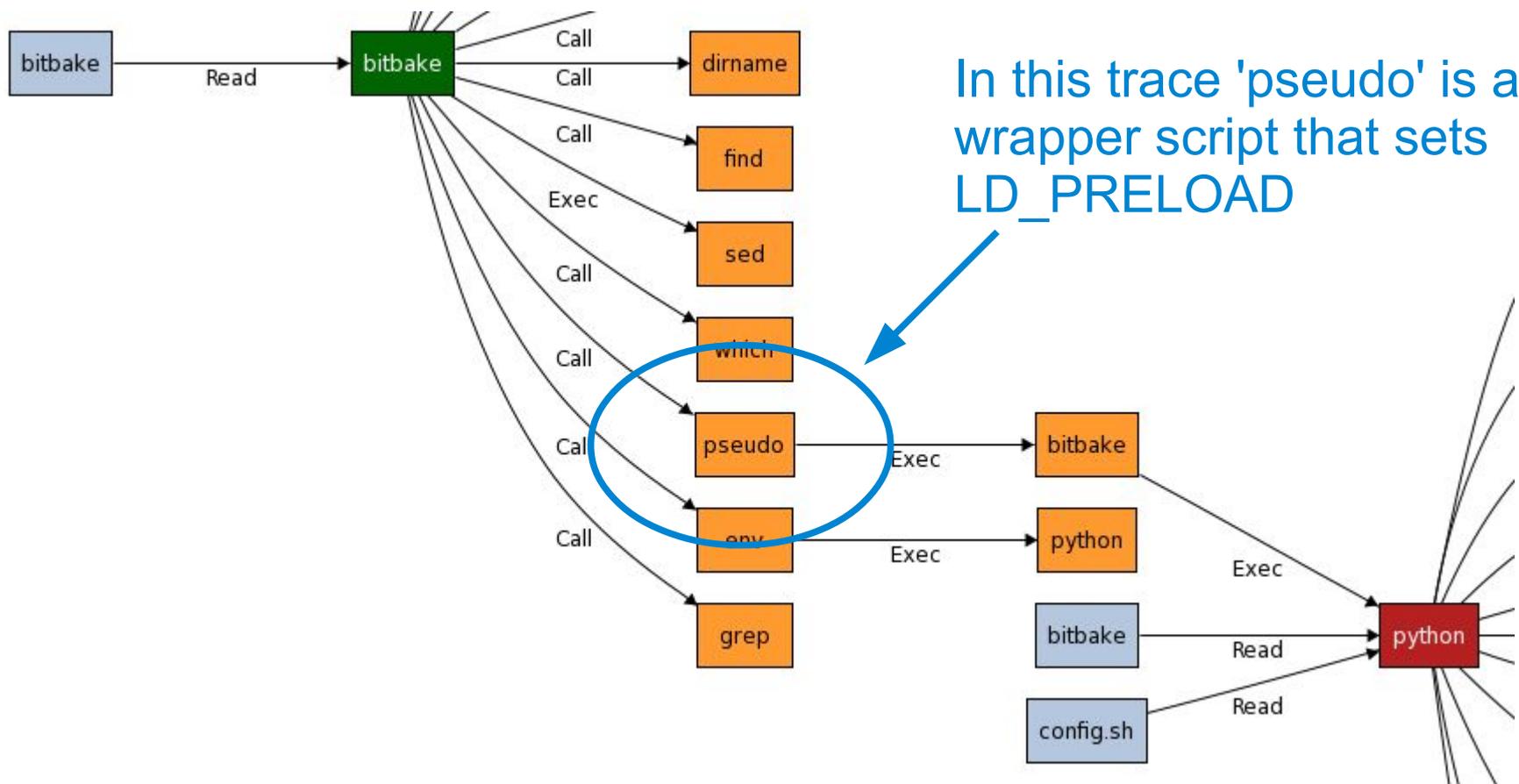


ellexus

Yocto 'Pseudo'

Yocto is a Linux build system

It uses an LD_PRELOAD tool called Pseudo ('sudo') to fake root access



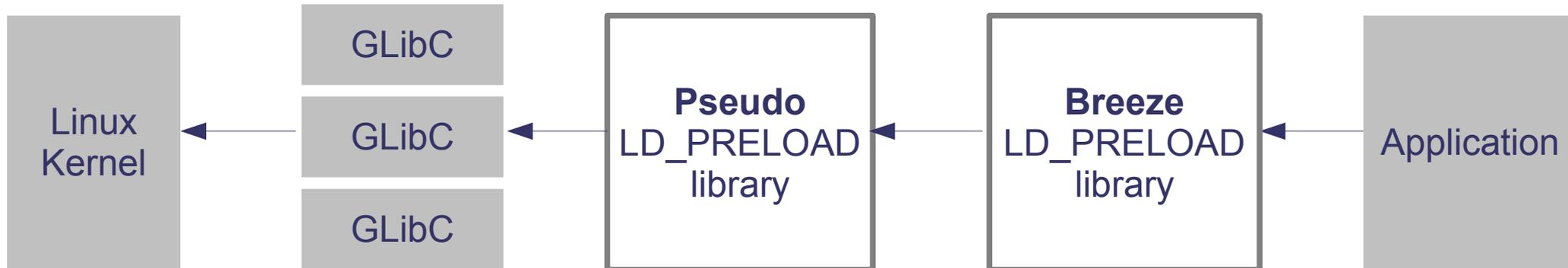


Getting it right

Glibc can have multiple versions of the same function.
This is even more complicated when another preload library is used.

It is important to preserve the original behaviour of the program.

Most of our code is involved in picking the right 'next' library version to use





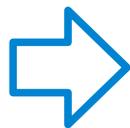
ellexus

**Why should you care
about application tracing?**



ellexus

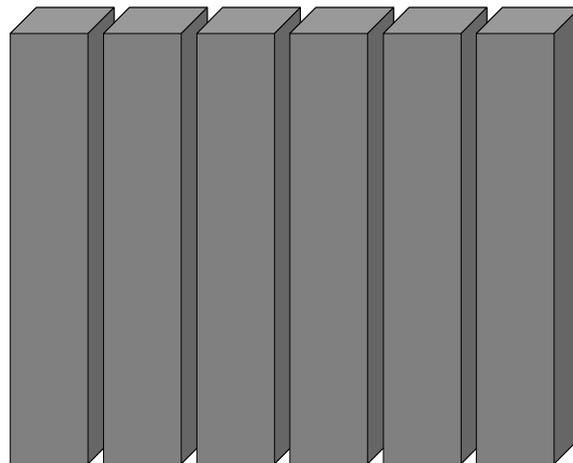
System Design



Job Scheduling

Virtualisation

Host OS



Multi-machine jobs

Suspend resume

Hierarchical file-systems

Virtualisation

Homogenous hardware?
- not cost effective

Bromium – custom stack
with one VM per app

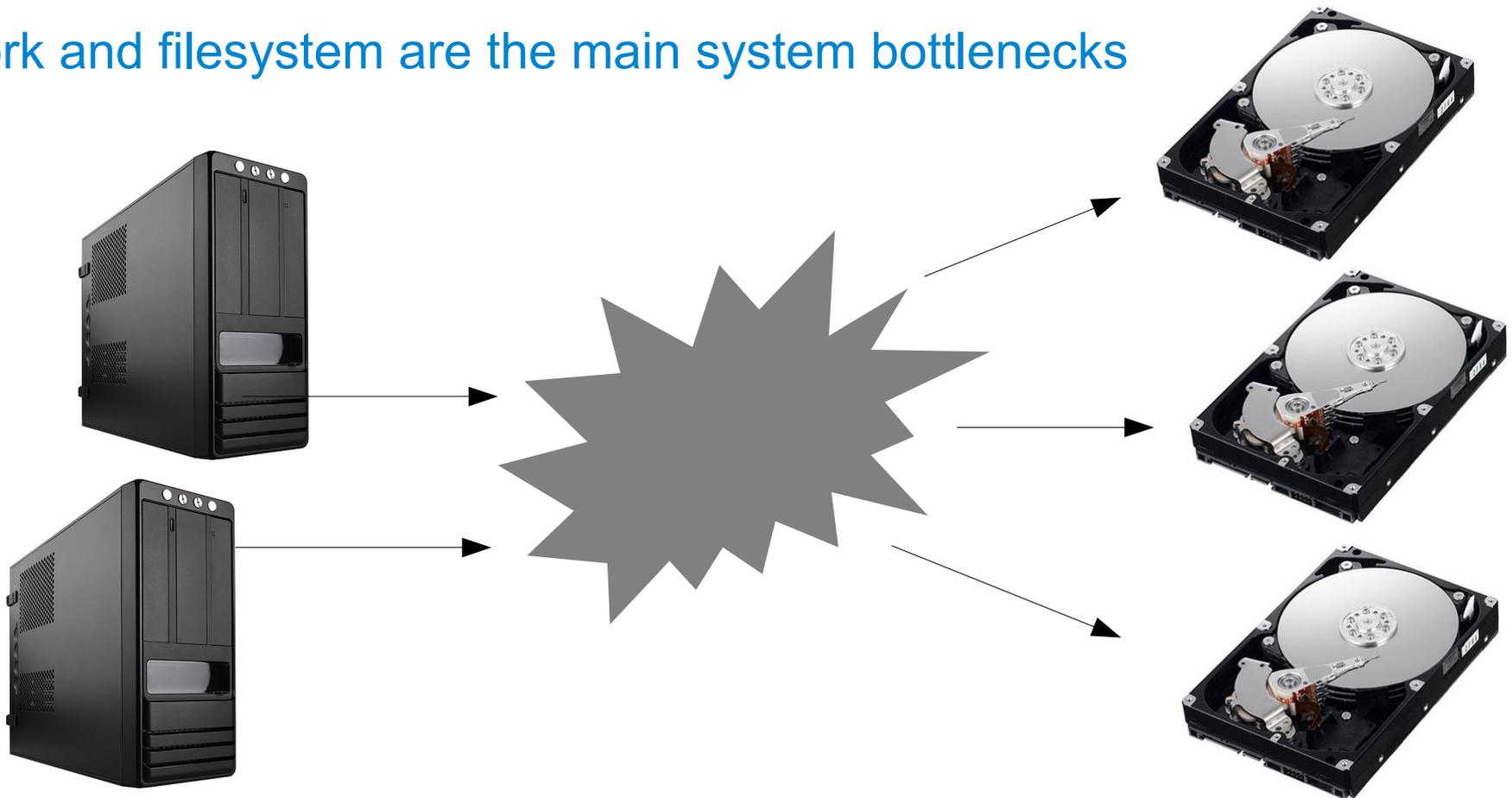


ellexus

Case Study 1: ARM filesystem profiling

Networked filesystem with different performance for different areas

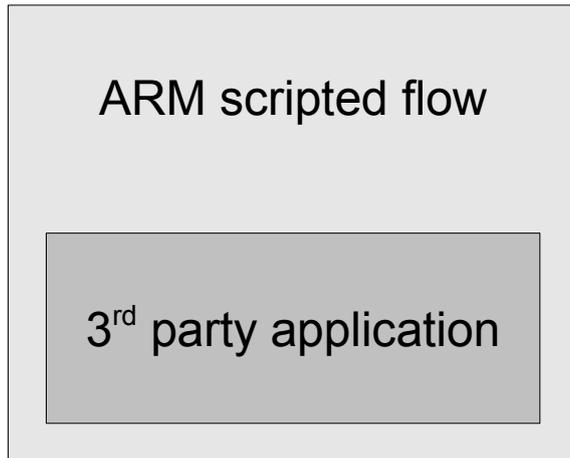
Network and filesystem are the main system bottlenecks





ellexus

Case Study 1: ARM CAD Flow



Script dependencies

Application dependencies

License settings

Is everything being stored in the right place?



ellexus

Case Study 1: Profiling file IO results

/scratch is medium-term storage and should only be used for results

We can see the writes at the end OK, but there are also lots of writes at the beginning. This is a problem.

/scratch
is written to at the
start of the trace



/tmp
is hardly used





ellexus

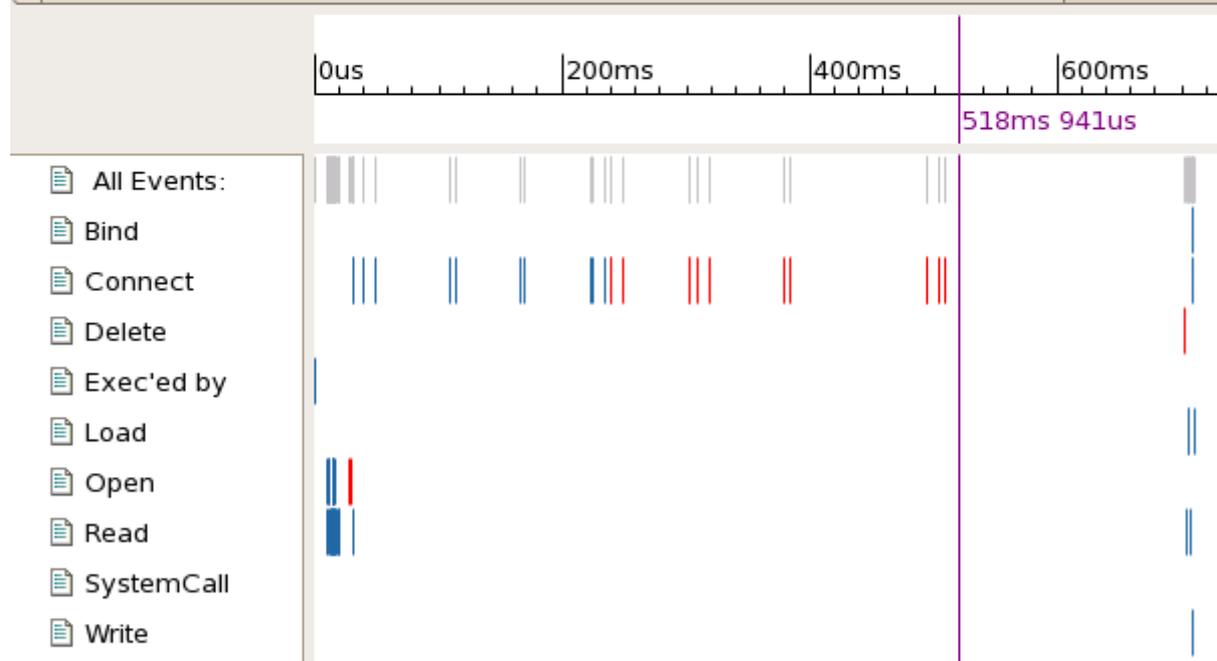
Case Study 2: Network IO

License server connections

This application spends
> 500ms connecting

Half of the connections fail
- it wastes 200ms

Time	Source PId	Event	Target
249ms 456us	9789	Connect	1.17.133.10.80.5221 (Network Location)
303ms 196us	9789	Connect	1.17.133.10.80.5221 (Network Location)
303ms 606us	9789	Connect	1.17.133.10.80.5222 (Network Location)
309ms 472us	9789	Connect	1.17.133.10.80.5221 (Network Location)
319ms 543us	9789	Connect	1.17.133.10.80.5221 (Network Location)
379ms 177us	9789	Connect	1.17.133.10.80.5221 (Network Location)
383ms 752us	9789	Connect	1.17.133.10.80.5231 (Network Location)
493ms 701us	9789	Connect	1.17.133.10.80.5227 (Network Location)
494ms 122us	9789	Connect	1.17.133.10.80.5227 (Network Location)
503ms 808us	9789	Connect	1.17.133.10.80.5227 (Network Location)
509ms 570us	9789	Connect	1.17.133.10.80.5280 (Network Location)





ellexus

Case Study 3: Cadence Client Technology

One user couldn't type anything in new login systems

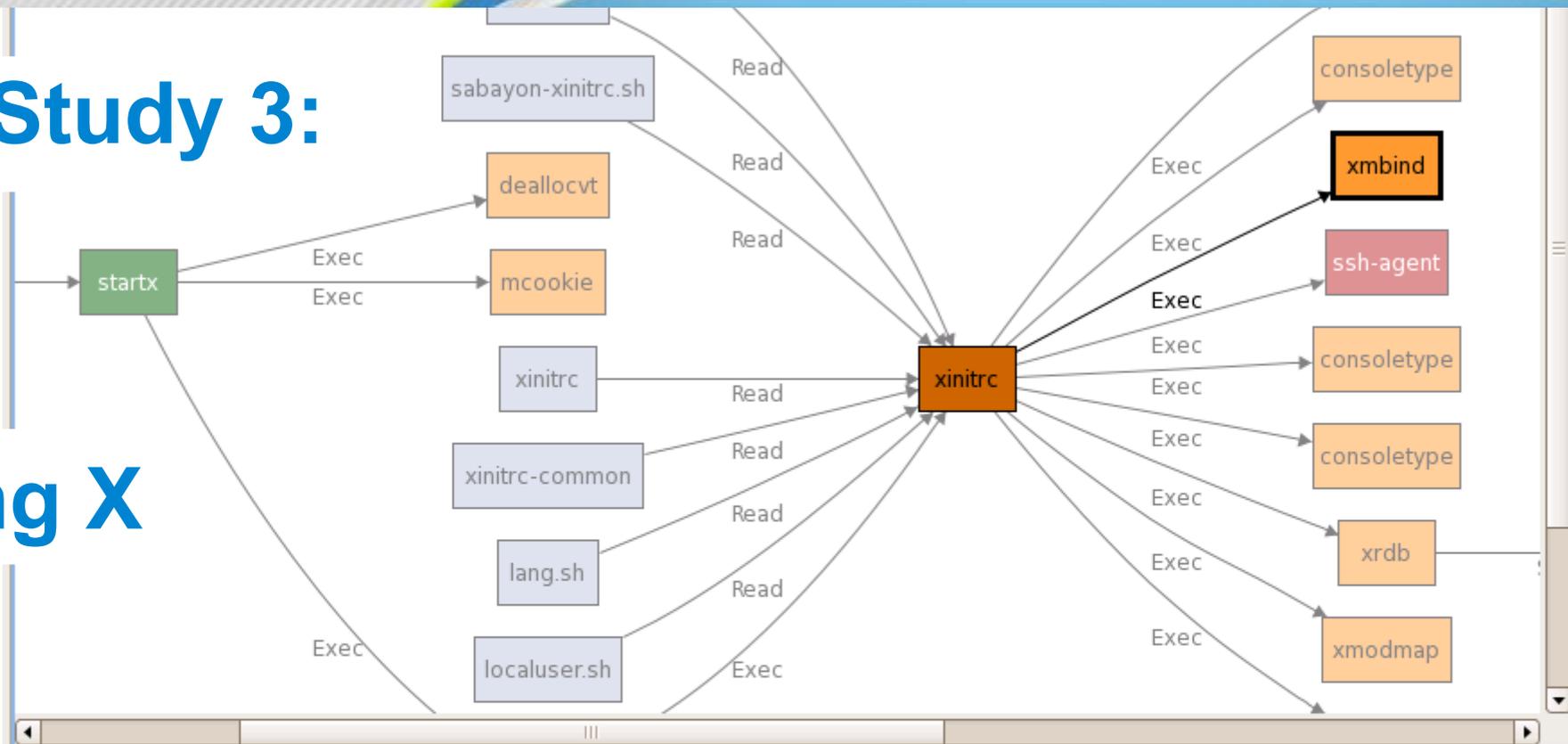
The IT manager traced a startx session for that user

Using Breeze he saw that xmbind was sourcing the wrong key bindings



Case Study 3:

Tracing X



Node View

- Exec'ed by
- Open
- Open (failed)
- Read

/usr/bin/xmbind - pid_20126

- /grid/common/pkgsData/openmotif-v2.3.0/Linux/RHEL4.0-1H2006-x86_64/lib/X11/bindings/xmbind.alias (Data File)
- /home/tonyh/.Xauthority (Hidden File)
- + /usr/share/X11/locale/ * (Multiple files at this location)

This time it was easy,
but how long until we can't do this?



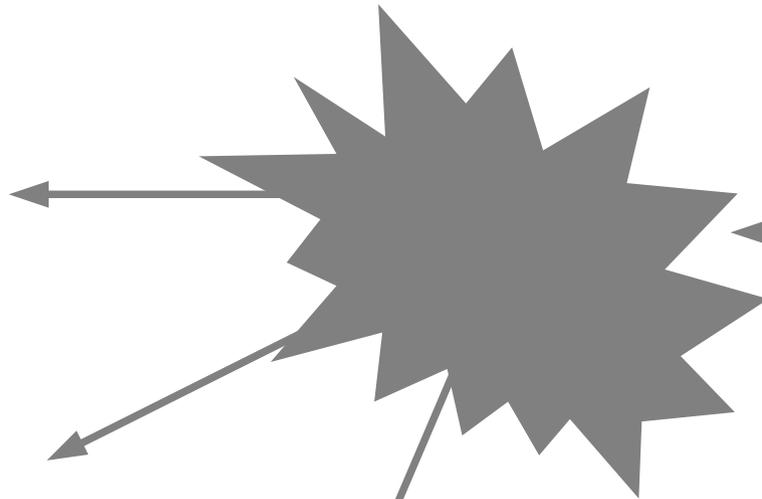
ellexus

Where do we go from here?



ellexus

Problem 1: Distributed applications



How do you profile an application that runs on many different systems?

Not necessarily any common factors between the platforms

Many components not fit for purpose



ellexus

Problem 2: What should we measure?

Hard to combine different data sets

We can't measure everything

We don't want to reverse engineer every system we need to profile

File System Daemon

Disk / Network IO

Job Scheduler

Virtual Machine

OS space

User Application

Application Framework

User space



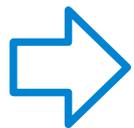
ellexus

Solution: Design for Profiling?

We have all heard of Design For Test...

... introducing Design For Profiling

We can build programs with debug symbols and debug code



This should be built into the high-level system design



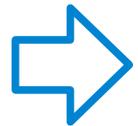
Profiling frameworks should be chosen at the start,
not as an afterthought



ellexus

Solution: Custom profiling frameworks

Breeze: *Trace



Customisable LD_PRELOAD framework

Jtrace : Java profiling and debugging



Open Source framework for inserting java snippets into classes
Lets you instrument the java class libraries

... what next??



ellexus

Big Data: Monitor everything and correlate

Grab information at all levels and look for trends

Difficult to get fine-grain coverage without harming performance

Could be the only way to improve performance in very complex systems

Cloud computing is only going to get more complicated

With enough data we could not only spot drops in performance, but we could predict resource needs such as memory consumption and power.



ellexus

Now it's time for your ideas...

Thanks for listening

Dr Rosemary Francis, Technical Director

Rosemary@ellexus.com