Into the Job: Gaining Insight into Your Workloads Using OGRT

HPC Knowledge Meeting 2016

21.04.2016

Barcelona, Spain

http://goo.gl/7DMegZ

Hello. Who am I?

Georg Rath

Systems Engineer at IMP/IMBA

IMBA
Institute of Molecular Biotechnology
of the Austrian Academy of Sciences

IMP
Research Institute of Molecular Pathology

What is happening inside the job?

program execution

shared libraries

environment

Why would you want to know?

- What software do my users run?
  - Unoptimized Python from the home directory?
  - Binary build of some software?
- Some BLAS library had a bug - are my users affected?
  - Which jobs did use that library?
- Is the environment "sane"?
  - Are there problematic environment variables set?

How would you do it?

Existing Solutions

Asking the users

1. "We use this pipeline: 'mnseq_4_custom_3.Copy 2.sh'."
2. Go through the shell script, check the programs, module loads without versions, hardcoded paths, everything you could and could not imagine.
3. Rinse, repeat

Existing Solutions

Hooking module loads

A sample ~/.profile:

```
module load cd-hit
module load emboss
module load hmmer
module load ncbi-blast
module load ncbi-blast+
module load mafft
module load muscle
```

$$load \neq use$$

Existing Solutions

XALT

- Needed a launcher (was true in 2014, parts of 2015)
- Not designed to track everything
- Tailored for HPC (TACC) needs
- Quite complex to deploy

What does OGRT do?

- Tracks execution of all programs in a job
- Track every shared object a program loads
- Embed a signature into programs and shared objects
- Outputs data to Elasticsearch/Splunk in near-realtime

What makes OGRT unique?

- Works without a launcher
- Lightweight
- Transparent
- Resistant to outside influence
- No runtime dependencies
- Easy to deploy

How does it work?

Tracking Programs

LD_PRELOAD

The loader "preloads" a shared object
when loading a dynamic executable.

...combine with a GCC 'constructor':

No Launcher/Wrapper

Tracking shared objects

"The **dl_iterate_phdr()** function allows an application to inquire **at run time** to find out which **shared objects** it has **loaded**."

With a signature

- **dl_iterate_phdr()** provides ELF program headers
- can we get our signature into a program header?
  - link section into target program and mark it allocatable

Signature

- Link in an object file at compile time
- Creates a note section in ELF (GCC does this too)
  - gets loaded into memory on execution
  - embeds an UUID
  - can be read by readelf/OGRT

Why the signature?

- same path - different executable
  - recompile of software
- discern user generated programs

Are we lightweight?

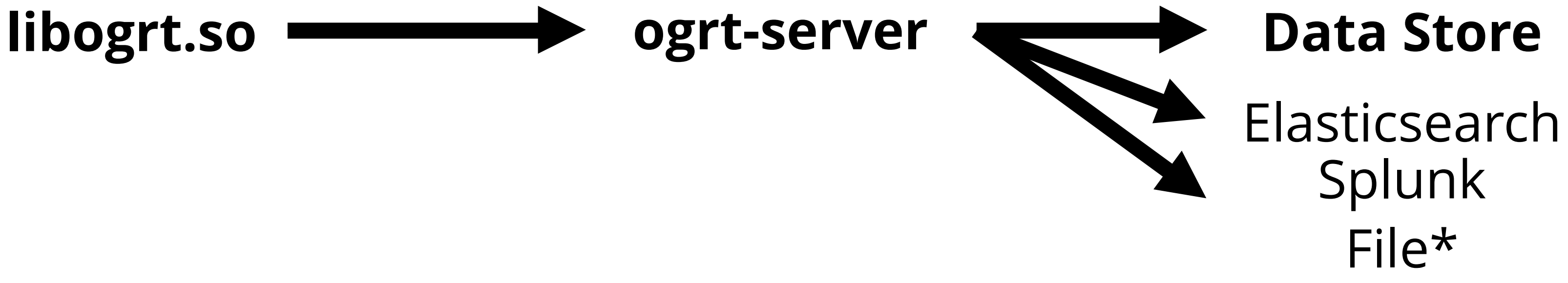
We are doing everything in memory.

Are we transparent?

OGRT is barely noticeable when active.

How do we persist the gathered data?

The Transport



*for debugging only

Demo

1. Deploying the client
2. Deploying the server
3. Playing with the client
4. Signatures and linking
5. Getting the data into ELK

Conclusion

OGRT is

- giving you deep insight into what runs on your machine
- a versatile tool for the sysadmins toolbox
- configurable to your needs
- very easy to deploy (literally in 10 minutes)

Outlook

- Syslog transport
- Filtering in preload library
- DB Level XALT compatibility
- eBPF evaluation
- Symbol level tracking (has the function x() been used)

Fin

https://github.com/IMPIMBA/ogrt