GNOME Memprof

A Look at the Implementation
COSCUP 2010

Our Agenda

- What is GNOME memprof?
- How is memory allocated?
- How to intercept the calls?
- How to get a backtrace?
- Summary

What Is GNOME Memprof?

- A tool to analyze memory usage.
- Visualize memory allocations
- Visualize memory fragmentation
- Visualize memory usage over time

How Does It Work?

- Tracks **only** calls to malloc/new/free/delete
- Generates a backtrace for each allocation
- Forgets about the allocation on free

How Does It Work?

- Will look into the implementation now.
- How to find an allocation?
- How to generate a backtrace?

How Is Memory Allocated?

- anonymous mmap
- sbrk
- This is for the actual allocator and not tracked by memprof.

How Is Memory Allocated?

- malloc/free
- new/delete
- new[]/delete[]
- Anything else?

How To Intercept The Calls?

Recompile glibc and call a different function?

Replace all calls to malloc and recompile?

Both are not practical!

How To Intercept The Calls?

- Create a library that contains a special malloc...
- Use dlsym to get the real malloc
- Make the special malloc call the real malloc
- Do the same for free and more allocator functions
- Only works when there is an external binding for the alloc calls

How To Get A Backtrace?

- On GLIBC backtrace(3) is available. For ARM compile with -fno-omit-frame-pointer.
- Or GCC's __builtin_frame_address(0)
- Or use a lib..
- Or walk the stack yourself
- Now we have a list of program counters..

How To Get A Backtrace?

- Demangle C++ names...
- Translate addresses to function names.
- libbfd is doing the heavy lifting.

Summary

- A small LD_PRELOADable library to intercept and backtrace.
- Uses libbfd to give us shiny names.`
- A GUI to navigate through the allocations.

Work Needed

- Memprof needs to be ported to ARM (backtrace)
- Memprof has issues with a custom allocator (inline functions)
- Maybe the lib should be done with systemtap..

- Questions?
- 謝謝