Performance Tools

IgProf in parallel systems

Lassi Tuura & Giulio Eulisse, Fermilab

Forum on Concurrent Programming Models and Frameworks, CERN, 29 February 2012

IgProf 1-2-3 for those unfamiliar with it...

1. **igprof.sourceforge.net**
   - igprof.sourceforge.net**/install.html**
   - igprof.sourceforge.net**/running.html**
   - igprof.git.sourceforge.net/git/gitweb.cgi?p=igprof/igprof
   - git://igprof.git.sourceforge.net/gitroot/igprof/igprof
2. **cmssw.cvs.cern.ch/cgi-bin/cmssw.cgi/CMSSW/IgTools/IgProf/plugins/**
   - More elaborate run-time control from within application, such as dropping heap dumps / performance snapshots at specific points
3. We'd be happy to give tutorials!
   - **cern.ch/lat/esc11** has a nice set of exercises

IgProf is multi-process, thread-ready

Automatically captures all threads and sub-processes for profiling, can target processes by name substring. For threads works both static and dynamic thread pools.

- This is not new, it's worked like this since a long time ago.
- Analyser supports merging results across processes and runs, and making comparisons; igprof-book adds more comparisons.

Supports x64 and ia32 linux, although CMS unlikely to notice any ia32 rotting since we only use x64. Have a raw sketch of OS X support, not in usable state at this time.

In CMS integrated with twice-daily integration build and release validation system: some number of performance and memory profiles automatically executed for a subset of validation matrix, results published to a web site.

But there are devils in the details...

Almost all current x64 system libraries have some fraction of invalid unwind info.

- Only recent compilers (GCC 4.5+patches) generate correct info.
- Cannot fix errors in stack unwinder, but can protect from crashes by performing fairly expensive validity checks.
- But... x64 stack unwinding is easily x5-10 slower than on ia32 in heavy use, so compounding the cost not acceptable for a profiler.
- We use an accelerated unwinder ("fast trace") with the validation checks mostly disabled. Very fast, but risky – conscious choice.

Result: **estimated 2-7% of performance profile tasks die of random crashes in system libraries.**

- Compiling everything with a better compiler is the only known fix. Yes, this implies rebuilding libc/libm is the only known final fix.
- Memory profiling fortunately rarely crashes.

And even more devils in the details...

Cloning complex 2GB+ process virtual memory can take >30 ms. If the clone is interrupted by an async signal, the kernel aborts the operation and user-space retries it.

- IgProf now enters a slow motion mode around fork() and system() to avoid triggering an infinite clone() loop in complex processes
- 5.9.3 fixes bug with fork() child not getting profiled until exec()

The GLIBC interface to walk shared libraries present, dl_iterate_phdr(), is not async signal-safe.

- A performance profile run will deadlock on a dynamic linker mutex with probability <1%; memory profiling not affected.
- Often can avoid the lock-up by executing a warm-up run, which is recommended anyway when targeting reproducible statistics.
- Can only be fixed with async signal safe interface, in talks with GLIBC, but also considering using dl_debug_state() instead.

Planned profiler improvements

#1 Restore profiling in fork() child without exec() [in 5.9.3]

#2 Support malloc() other than the one from GLIBC

#3 Arbitrary function profile, maybe -finstrument-functions

#4 Internals + dump format improvements, OS X support

#5 Attach profiler into already-running processes

Planned tools improvements

#1 Reintroduce regexp filter/collapse support in analyser

#2 Parallelise igprof-analyse

#3 Dump format + symbol lookup improvements

#4 REST API for igprof-navigator, unify with igprof-book

#5 igprof-book further development

Summary

Mostly positive experience using IgProf in applications which stretch or exceed the limits of all other tools.

Architecture very much suited for profiling multi-threaded, multi-process applications, including merging results from many runs together, and comparisons via igprof-book.

Intention is to do well what it does, but keep it basic and not overlap much with other existing tools. User feedback drives what gets done.

Inherent awkward tool chain issues on the x64 platforms. Essential for user community to pressure for system fixes at the root cause.