Efficient dynamic and static tracepoints in kernel space

Rafik Fahem
Michel Dagenais
Department of Computer and Software Engineering

December 8, 2011
École Polytechnique, Montreal

Content

- Goals
- GDB tracepoints
- KGTP kernel module
- Work on dynamic tracepoints
- Static tracepoints integration
- Results
- Future work
- Conclusion

Goals

- Insert dynamic and static tracepoints in the kernel
- Associate conditions to tracepoints
- Use dynamic expressions to define conditions
- Evaluate and collect dynamic expressions using tracepoints
- Collect the static data defined by TRACE_EVENT using static tracepoints

Expressions

- May be complex
- Use the kernel variables
- Arithmetic and logical operators
- C syntax

GDB tracepoints

- Initially developed for user-space tracing
- Debug a program without interrupting it
- Locations in the program to which data collecting probes are associated
- Probes execute dynamic actions
- Tracepoints may be conditional
- 2 types: dynamic and static(UST)
- Work only in remote mode

Agent Expressions

- Used to define conditions
- Can also be evaluated and collected in the trace
- Support arithmetic and logical operators
- Use code variables
- Transformed into bytecode by GDB
- Bytecode is interpreted each time the tracepoint is hit by the remote stub

KGTP

- KGTP is a GDB stub that implements tracepoints in kernel-space
- Only dynamic tracepoints are supported
- Bytecode is interpreted each time the tracepoint is hit
- Dynamic tracepoints are based on kprobes

Work on dynamic tracepoints

- A bytecode to native code converter was implemented in order to increase performance
- Used for conditions and actions
- X86 architecture is supported
- This converter is similar to the GDBServer converter
- An assembly code snippet corresponding to each opcode
- Code snippets are copied into an executable buffer

Kernel Static Tracepoints

- Connect to kernel static tracepoints defined using TRACE_EVENT
- KGTP is now able to list, enable and disable static tracepoints
- Static tracepoint data is collected
- Static tracepoints have the same capabilities than dynamic tracepoints

Kernel Static Tracepoints

- Static tracepoints detected automatically
- No manual integration is needed(Systemtap)
- Static data simply collected using the “trace $_sdata” command
- TRACE_EVENT was modified in order to collect the registers at the tracepoint site
- Static tracepoints use the bytecode to native code converter

Results: dynamic tracepoints

	KGTP with native code support(cycles)	Systemtap(cycles)
Condition	202	351
Expression	500	1035
Condition + Expression	602	1061

Results: TRACE_EVENT data

	KGTP with native code support(cycles)	Systemtap(cycles)
Condition	154	223
Data	1216	1252
Condition + Data	1368	1336

Future Work

- Optimize the bytecode produced by GDB: unnecessary operations
- Work on the data structures used to store the trace: ring buffers
- File tracing mode
- Integrate this work in other tracing tools

Conclusion

- GDB tracepoints provide a quick solution to trace in kernel space
- Some improvements are needed
- More details in my thesis

Questions?