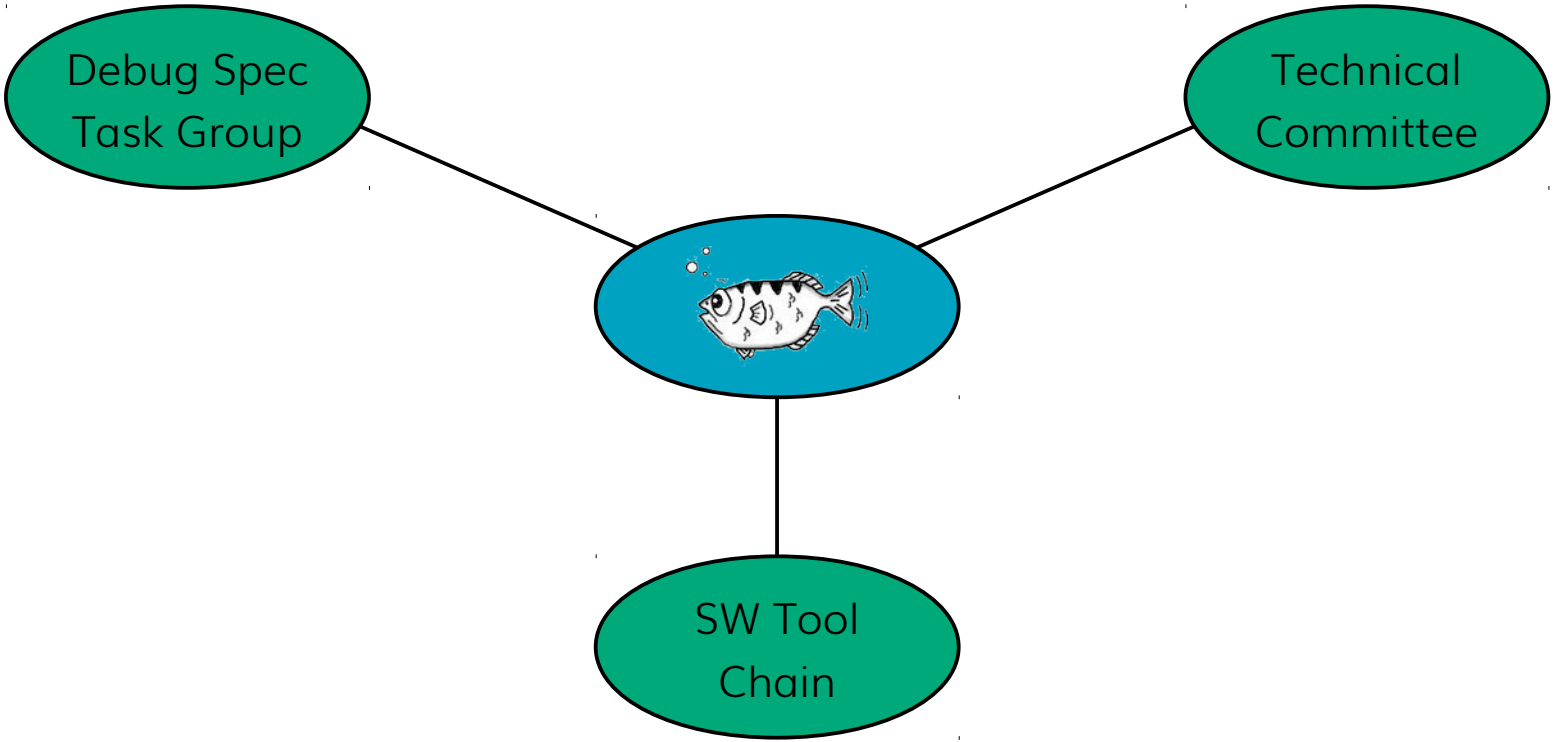

# GDB for RISC-V

## Extending Support for Bare Metal Multi-core Debugging

Jeremy Bennett
Andrew Burgess

# GDB in the RISC-V World



# Upstream GDB

- Committed upstream 6 March 2018
  - basic bare metal support
- Combined effort of several engineers:
  - Andrew Burgess, Tim Newsome, Albert Ou, Darius Rad
  - official maintainers are Andrew Burgess & Palmer Dabbelt
- Nightly regression testing
  - against GDBsim and remote GDBserver (wrapping GDBsim)
  - range of architectures from RV32IM thru' RM64IMFDC
  - >99% pass rate on all architectures

# GDB Regression Tests (Sim Subset)

| | RV32IM | RV32IMFC | RV64IM | RV64IMFDC |
|---|---|---|---|---|
| PASS | 35,936 | 36,050 | 35,965 | 36,067 |
| FAIL | 309 | 190 | 303 | 194 |
| XPASS | 1 | 1 | 1 | 1 |
| XFAIL | 30 | 30 | 32 | 32 |
| KPASS | 3 | 3 | 3 | 3 |
| KFAIL | 53 | 53 | 51 | 51 |
| UNRESOLVED | 12 | 9 | 9 | 9 |
| UNTESTED | 231 | 231 | 229 | 229 |
| UNSUPPORTED | 326 | 326 | 326 | 326 |

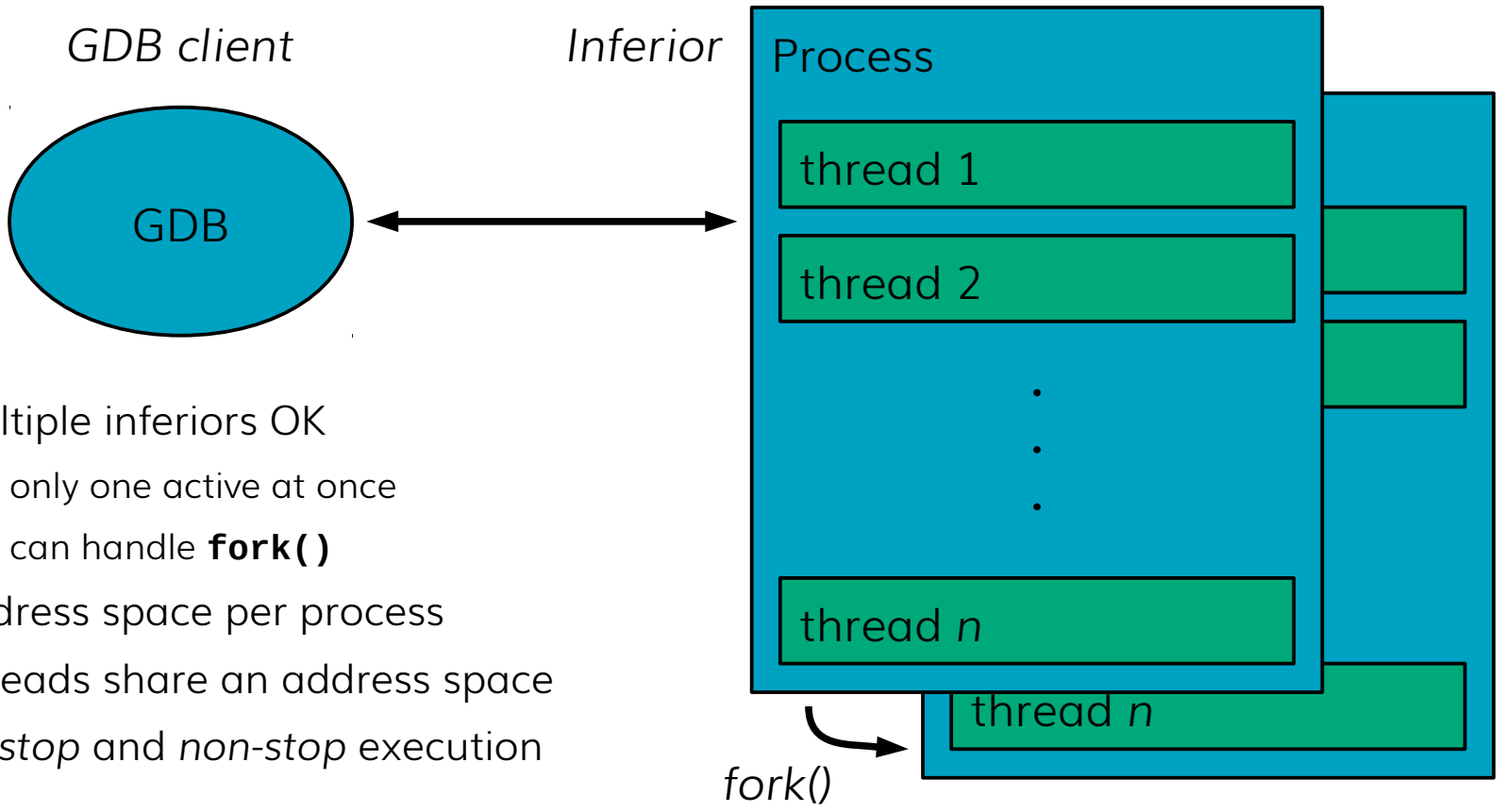
# GDB Regression Tests (GDBserver Subset)

| | RV32IM | RV32IMFC | RV64IM | RV64IMFDC |
|---|---|---|---|---|
| PASS | 35,746 | 35,856 | 35,793 | 35,896 |
| FAIL | 287 | 169 | 261 | 151 |
| XPASS | 1 | 1 | 1 | 1 |
| XFAIL | 30 | 30 | 32 | 32 |
| KPASS | 3 | 3 | 3 | 3 |
| KFAIL | 53 | 53 | 51 | 51 |
| UNRESOLVED | 10 | 10 | 10 | 10 |
| UNTESTED | 232 | 232 | 230 | 230 |
| UNSUPPORTED | 318 | 318 | 318 | 318 |

# Upstream GDB Next Steps

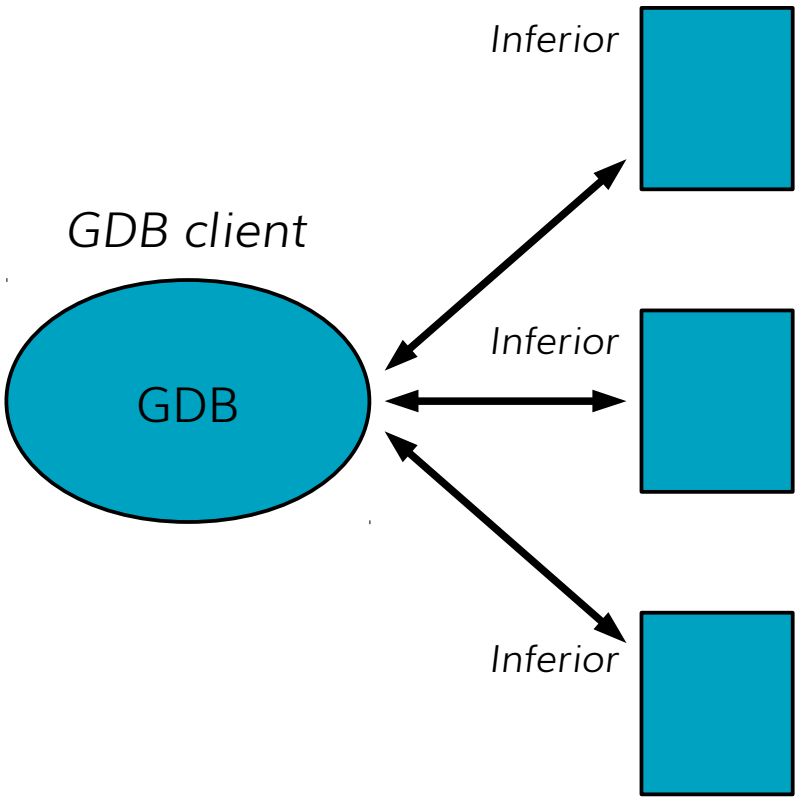
- Adding XML target description support
  - initial work removed for now, aim for OpenOCD compatibility
- Memory map support
- Remote I/O support
- Adding non-DWARF stack unwinding
- Upstreaming a GDB simulator——possibly CGEN based
  - see Mary Bennett's talk/poster later
- Linux application debugging
  - both native and via Linux *gdbserver* application

# GDB Historic View of Multicore Debug




- Multiple inferiors OK
  - only one active at once
  - can handle **`fork()`**
- Address space per process
- Threads share an address space
- *All-stop* and *non-stop* execution

# GDB Future View of Multicore Debug




- Multiple active inferiors
  - each a concurrent flow of control
- Each inferior associated with an address space
  - which memory can it see?
- Each inferior associated with a program space
  - which symbol table relates to its code?
  - breakpoints across multiple inferiors with the same program space
- All-stop and non-stop execution
  - inferiors not just threads

# Current Status

- Upstream GDB supports multiple concurrent inferiors
  - works for native/Linux and single remote target
  - generic work by Pedro Alves and others to support on multiple remote targets
- GDB for RISC-V has this functionality
  - tested with 36-core RV64IMC system
  - working on a public version based on PULP
- More work is needed for complex address spaces
  - for example where *some* memory is shared with other inferiors


# Thank You

## www.embecosm.com