	GDB	UDB
Time-to-fix	**SLOW** • Lengthy, complex multiple- iteration debugging process reliant on guesswork • Build and test multiple hypotheses all whilst reproducing the failure over and over again	**FAST** • Single debug cycle • Put a watchpoint on the variable containing bad data and run backwards to go straight to the line of code that most recently modified it
Workflow	**REPETITIVE** • Unlikely to gather all key pieces of information first time - multiple reruns needed to get to root cause	**EFFICIENT** • Capture the failure once and debug it in one session - time travel backward from crash/error to root cause
Ease of debugging	**COMPLEX** • Program behavior may differ with repeated restarts/reruns - making debugging unnecessarily complicated	**EASY** • Ability to navigate forward and backward in a single debug session making code observability easier
Time Travel Debugging capability	**UNUSABLE** • Technically available in GDB, but too slow to be usable on real-life complex applications	**PERFORMANT** • Purpose-built time travel debugger. Performant and usable for debugging real-life, complex programs (including cloud / VMs / multithreaded)
Code-level observability	**UNUSABLE** • Impractical due to slow performance when navigating in reverse	**RESPONSIVE** • Watch memory changes and inspect global and local variable values in order to fully understand unintended code behavior as you navigate forward and in reverse