PROFILING AND PERFORMANCE

Perfecting Your Product with Deep Software Expertise

We've all had it happen: days before release you discover the memory leak that crashes your product, the test that makes your graphics sluggish, or the use case that brings your hardware to its knees.

While there are dozens of debugging, profiling, and performance tools to help, finding them is just the start. All those powerful tools require knowledge – which tools to use when, how to interpret their results, and how to customize them to catch complex problems.

At KDAB, hard problems are our specialty. We know how to use the most powerful debugging and profiling tools around – in many cases because we've created them or we maintain them. We're happy to help your team slay bugs, speed bottlenecks, and thrift memory on mobile, desktop, and embedded platforms. If you'd prefer to develop superior debugging and optimization competency in-house, we have customized training courses for your team or scheduled classes around the globe.

Tool user… and creator

KDABians aren't just expert tool users – we're expert tool developers as well. We've created many of the high-powered tools used by advanced development teams to clean, debug, profile, and optimize their code, like GammaRay, clazy, HotSpot, Heaptrack, and massif-visualizer. As part of our philosophy to help and educate the development community, these tools were all created as open source software projects. Our developers also make ongoing contributions to many other tools, standards, libraries, and frameworks in C++, Qt, and OpenGL. This means that we not only know these tools intimately, we can also add features, fix bugs, or adapt them as needed.

Development and debugging

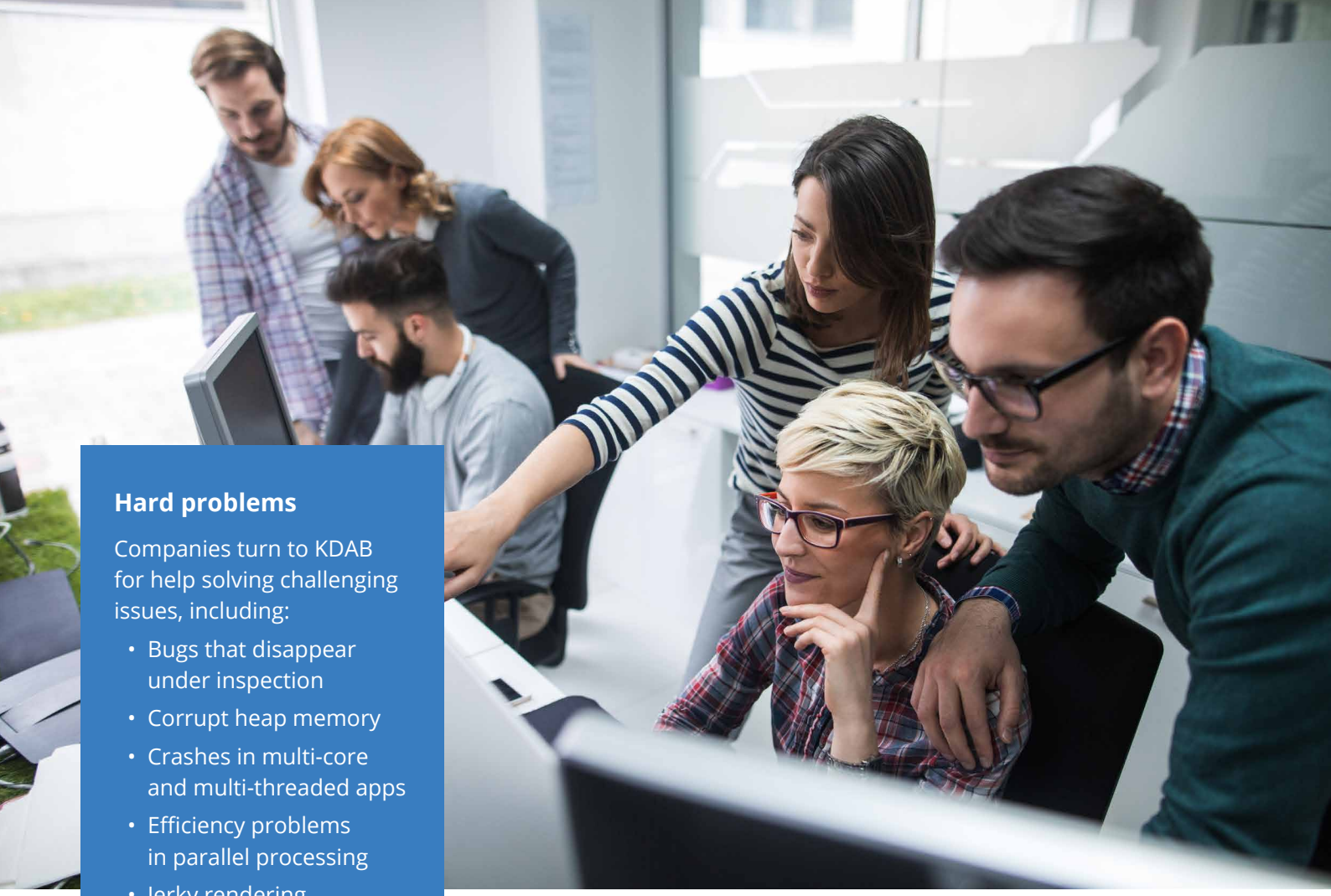
Every developer has experienced the pain of interpreting crufty old code to find a bug, pouring through endless memory dumps because the debugger doesn't understand the data, or struggling with build processes that impose structure at the expense of developer time. Some of these problems can be solved by using the right productivity tools early in the development cycle – let us help train your team or work side-by-side on fixing problems using a number of different tools, including:

- **Clang Tidy** – compile-time tool that points out places where C++ code can be improved with easier to read, better performing, and less fragile C++11/14/17 semantics
- **GammaRay** – Qt-aware debugger plug-in that understands Qt structure and idioms, and lets you easily examine and edit values in many Qt-specific components
- **Clazy** – Qt-aware compiler plug-in that analyzes source for helping fix misused APIs, inefficient constructs, and extraneous allocations
- **Buildbot, CruiseControl, Jenkins** – tools that help produce clean code that's tested and functional with minimal intrusion to workflow

Performance optimization

When code is slow, finding where bottlenecks occur is a significant part of the challenge. Even after you've located the source, you need to see if the API redesign, replacement algorithm, or new caching strategy has a consistently positive effect. Profiling tools are essential when used properly. We're experts in finding the right profiling tool for the circumstance and figuring out how to insert it without breaking your code. We work with many, including:

- **VTune** – Intel's performance tool that lets you do sampling, measurement, and analysis for thread, JIT, and hardware tuning for the most popular languages
- **Perf and Oprofile** – tools that let you examine Linux kernel performance, providing statistical profiling and scheduler tracing on x86, ARM, PPC, and SH processors
- **HotSpot** – visualization tool for graphically viewing and analyzing perf reports with call graphs, hot-spots, caller/callee, and event timelines
- **System Profiler** – tool for capturing and visualizing QNX system performance between all system tasks or within a single application
- **QML Profiler** – add-in within Qt Creator that lets you profile Qt Quick applications, identifying dropped frames or stuttering animations



Hard problems

Companies turn to KDAB for help solving challenging issues, including:

- Bugs that disappear under inspection
- Corrupt heap memory
- Crashes in multi-core and multi-threaded apps
- Efficiency problems in parallel processing
- Jerky rendering
- Limited disk space
- Memory fragmentation
- Non-responsive UIs
- Overheating or other thermal issues
- Poor peripheral throughput
- Processor-intensive applications
- Restrictive memory budgets
- Slow system boot-up times
- Rapidly draining batteries
- Rare or non-reproducible bugs

Memory optimization

It's often not practical to conserve RAM simply by using less – you'll need to determine allocation patterns and pooling strategies that shrink your memory consumption. Just like locating heap leaks or code that corrupts memory, each requires a detailed custom analysis for every particular application. We've helped clients squeeze into tiny RAM budgets, reduce memory fragmentation, and find memory problems that resist location with several different tools, including:

- **Heaptrack** – tool that shows peak memory usage, leaking functions, biggest allocators, and excessive temporary allocations
- **Valgrind** – instrumentation framework that detects memory and threading bugs, profiles your memory use, and finds stack overruns
- **Massif Visualizer** – tool that graphically displays valgrind's massif snapshots, providing call-tree, memory consumption, and hot-spot views

Five modules in A typical KDAB profiling workshop

1. **Assessment**: Assessing which aspects of performance are most important for your project
2. **Tool selection**: Finding and configuring the right tools for performance analysis in each application
3. **Benchmarking**: Creating a reliable, repeatable infrastructure for running benchmarks
4. **Profiling**: Identifying hotspots and performance issues
5. **Updates**: Suggesting code changes and assessing their outcome

Graphics optimization

GPU resources are powerful but not infinite: excessively off-loaded CPU computations, ineffective pipelines, and redundant draws can kill your frame rate. But squandered graphics cycles are hidden under layers of abstraction that are incredibly hard to peel back. Our graphics team can help you deliver buttery-smooth rendering and screaming frame rates. Among others, they use the following tools:

- **apitrace** – tools to trace OpenGL and DirectDraw APIs, replay saved traces, and inspect graphical system state, textures, and framebuffers throughout
- **GL Trace** – real-time OpenGL debugger and analyzer for forward and backward call-by-call OpenGL API stepping
- **NVIDIA Nsight, AMD CodeXL, and Vivante VProfile** – vendor-specific tools that expose hardware internals to enable GPU profiling and debugging

Contact us for more information at info@kdab.com.

www.kdab.com

About the KDAB Group

The KDAB Group is the world's leading software consultancy for architecture, development and design of Qt, C++ and OpenGL applications across desktop, embedded and mobile platforms and is one of the biggest independent contributors to Qt. Our experts build run-times, mix native and web technologies, and solve hardware stack performance issues and porting problems for hundreds of customers, many among the Fortune 500. KDAB's tools and extensive experience in creating, debugging, profiling and porting complex applications help developers worldwide to deliver successful projects. KDAB's trainers, all full-time developers, provide market leading, hands-on, training for Qt, OpenGL and modern C++ in multiple languages. Founded in 1999, KDAB has offices throughout North America and Europe.