Developer Velocity.

Accelerating the Developer Workflow and Time-to-results

Legal Statements

Developer Velocity.

Groq Tech Doc

Often when I talk with architects on the cutting edge of machine learning deployment models, there is a consistent theme to the conversations: developer velocity is slowed down by the tools we use. This is a challenge that needs to be overcome. We have the world's smartest innovators, but they can't try their new ideas fast enough because the development process is so laborious. This includes first the design of the algorithm, and then, the optimization process and finally, the process of profiling performance. And there's really no way around it: Workloads need to run in a real world environment to get a good statistical distribution of results. By profiling across many of these runs, developers gather insights into performance and algorithm efficacy. Then the manual optimization work that ensues forces this process to iterate over and over.

For example, we visited a top autonomous vehicle maker in Europe. Their developers sit in the back of the car with a laptop as it races around a test track. This is where they run their field tests and this is where they run their performance profiling until they can collect enough data. Then they head back to their office to revise the model to better meet requirements – which could take more days of hand optimization to run well on the target device (like GPUs, and FPGAs) and in the case of the latter, includes lengthy RTL simulations.

We have a fintech customer who has high value and expensive quants on their team who need to iterate quickly and converge on more optimal solutions. Every day it takes to deploy their latest brilliant idea equates to lost earnings from the better algorithm. The customer can't keep the quants iterating when they have to stop to spool up many racks of CPU servers and run a long simulation that profiles their algorithm performance. Their productivity diminishes because they're busy collecting more and more data, iterating and iterating.

Determinism

Historically, dynamic profiling uses a whole host of disparate tools like CPU tools (VTune), PyTorch, Tensorflow (TF Perf Analysis plug-in), GPU tools (gpu-smi, DLProf/PyProf/NSight), and for FPGAs – time consuming recompiles to include invasive JTAG probes. But they all have different levels of vendor support and different integration points, and they all require unique user expertise. With traditional devices, performance can't be guaranteed; there are too many variances in the underlying compute device nodes, and too much tail latency across the network of devices interconnected. Dynamic profiling attempts to capture a statistical distribution of performance across many runs of a workload by looking at points in the hardware that are physically measurable.

Not at Groq. Our chips, nodes, and networks are deterministic. Here, determinism equals 100% predictable performance—and that's what speeds up developers. Now a developer can hit compile on their code and within seconds, Groq's static profiling provides an ultra-detailed performance report and visualization of the entire chip's compute and memory usage for the entire program. Immediately, the developer has an accurate summary of performance metrics like latency, utilization and bandwidth, memory and FLOP occupancy, and power. All of the sudden, the slow, painful dynamic profiling process is gone.

With determinism, you can predict exactly how many clocks it will take to run a workload and exactly the order of numeric operations—this can help to better model important system metrics like latency and accuracy. Your silicon always runs the exact same way with no statistical variation. And not just in one chip, in a whole network of chips, with no variation from ethernet protocols or internet cables or extra processors. Because our networks are chip to chip, direct communication. No extra components.

Now you can compile a program knowing exactly what the performance will be—every single time. Which means you've cut off a whole chunk of the development cycle, since you get the result right away. And you can do all of this in your comfortable work area, without going to a noisy lab or a dusty race track. This is because you can accurately predict performance down to clock cycle resolution.

TCO

Now let's think about TCO: I can look at a static Groq programming file and can audit the silicon to easily see how utilized my compute cores are. Having this level-of-view is a powerful insight to a developer and to an infrastructure manager. And, since we know literally what's happening in every square millimeter of the chip at every clock cycle, we can predict power in a very, very controlled way too. I can see when the workload is at 0% or 100%. If I can predictively see power spikes, then I can optimize my kernels, altering power until my workload is flat and normalized. It's all because of our determinism— and predictable performance.

At Groq, we're fundamentally changing the developer's human experience. High performance, low power, low latency, all those things are wonderful technology features, but we're removing what limits how humans innovate—which happens to be the biggest bottleneck in the industry.

Interested in Learning More?

For more information on Groq technology and products, contact us, follow us on YouTube and Twitter, and connect with us on LinkedIn.

Additional Related Documents

- Accuracy Tech Doc
- Latency Tech Doc
- Predictability Tech Doc
- Scalability Tech Doc