Recent years have seen the advent of serverless computing, where thousands of containers can be launched instantaneously, and paid in millisecond increments. For embarrassingly parallel burst tasks, such as compiling and linking software, these platforms can be used for massive speed-ups, while costing much less than a dedicated server.

Existing tools often require maintaining prohibitively expensive clusters, while existing serverless approaches such as gg have limited build system compatibility. By integrating into the Clang compiler itself, Cymbl provides four key advantages over existing tools: infinite parallelism without user-supplied infrastructure, drop-in compatibility with existing build-systems, deterministic builds, and a fine-grained compilation cache across users and codebases. Cymbl also has support for Clang’s tools and compiler plugins such as LLVM-based automatic differentiator (see the Enzyme poster).

We evaluate the performance of Cymbl by benchmarking compile times of several codebases. We evaluate 1 local core, 96 local cores, and 8000 simultaneous Clang tasks. We also evaluate the performance of Cymbl when the compilation has already been cached. Times are the mean of three runs taken from an Amazon c5.3metal instance.

We found roughly similar performance on 1 and 48-core builds, this isn’t a true apples-to-apples comparison. That said, it appears gg had a much more significant overhead than Cymbl. On the Chrome build, Cymbl also appears to be significantly more performant.

To try Cymbl out, please email us and visit https://cymbl.dev/.

Acknowledgments

William S. Moses was supported in part by a DOE Computational Sciences Graduate Fellowship DE-SC0019223.

References
