Preserving Type Information in Memory Usage Measurement of C++ Programs

Zsolt Parragi Zoltan Porkoláb

Eotvos Lorand University, Faculty of Informatics

zsoltparragi@caesar.elte.hu gsd@inf.elte.hu

Memory profilers are essential tools to understand the dynamic behavior of complex programs. Unfortunately, in C++ we have no runtime reflection capabilities, and allocators in the standard library also hide type information, limiting the information available for profilers. Most tools provide information about the location and call stack of the allocation, but recovering type information only from those is practically impossible in many cases.

Previous methods to solve this problem[1] used preprocessor hacks to preserve type information during the compilation process. The complex syntax of new and delete operators limits the use of this technique, and requires several workarounds, as it conflicts with certain valid C++ constructs, which are used for example in the C++ standard library.

This paper discusses a different transformation technique: a source to source compiler based on clang tooling, making it possible to transform the code based on it's abstract syntax tree. This method allows us to deduce every required property about allocations during the compilation process, allowing the tool to work without reflection or even RTTI.

References

[1] J. Mihalicza, Z. Porkolb, and A. Gabor, Type-preserving heap profiler for C++ (2011)