AOT compilation in OMR: Relocations

Georgiy Krylov, Petar Jelenkovic,
Gerhard W. Dueck, Kenneth B. Kent
University of New Brunswick, Faculty of Computer Science
{georgiy.krylov, petar.jelenkovic, gdueck, ken}@unb.ca
Daryl Maier
IBM Canada
maier@ca.ibm.com

Eclipse OMR: Runtime creation toolkit

OMR, a toolkit to support runtime construction, instruments runtime builders with hooks to useful modules: garbage collector or just-in-time compiler. Our focus: developing infrastructure for adding ahead-of-time (AOT) compilation and loading.

Ahead of Time Compilation

Compiling a program code before execution and persisting it for future reuse is one way to define ahead-of-time compilation. Necessary components are: code generation, program loading, relocation, and symbol resolution.

AOT in OMR

There are two possible paths for AOT in OMR: using the system linker paired with UNIX ELF files, or using OMR's own infrastructure (the subject of our research). The **table below** describes the possible **components** of each process:

Ahead of Time Compilation: Relocations

Relocations in the classical sense are defined both as:

> *A process of modifying addresses to satisfy the memory context of the current program, or, a process of resolving references to external symbols defined within libraries or by other means, for instance, using the **extern** keyword*.

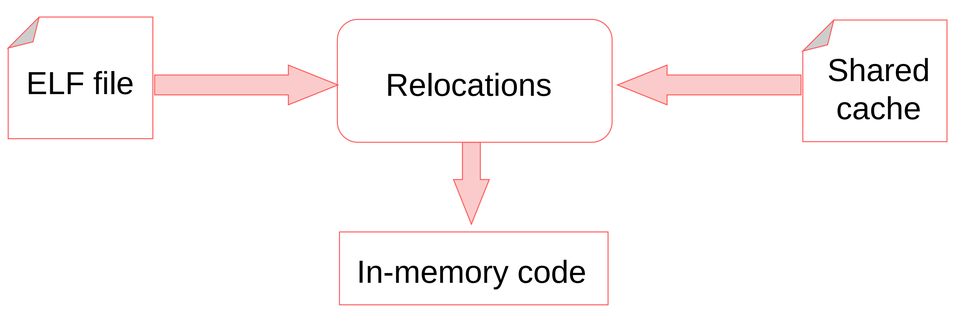
ELF Relocations in OMR

Does not:

- Have global and local static data
- Support dynamic libraries for executable files
- Provide support for dynamic and object oriented languages

Does:

- Generate **relocatable** and **executable** object files
- Support custom **metadata**



OMR Specific Relocations

The relocations, **as they are in OMR,** have their own class hierarchy that has a range of responsibilities:

Kind	Status
ELF-like relocations	Present, used
Label relocations	Used by JIT label relocations
External relocations	At a starting stage
Validations	Can be implemented

Research Directions

- **Implement external** relocations: address VM values
- I**nvestigate validations** w.r.t language agnostic runtime environment, if desired, **port to OMR**

	With UNIX support	OMR specific	
Code generation	Output of **TR::JIT Just in Time** compilation is reused		
Storage	Hard disk stores generated .o file (ELF format)	**Shared cache,** represented as a memory mapped file or shared memory region (work in progress)	
Loading	Is performed using ld UNIX loader	Not needed – runs within a process	Retrieved from a **code cache** region inside the **shared cache**
Relocations and **Symbol Resolution**	Falls to dlopen(), has limitations	Subject of ongoing research, work in progress	