P2666R0

Last Use Optimization

Bengt Gustafsson

Kona - 2022

Presentation contents

- Rationale
- Proposal contents
- Rules for compilers
- Rules for programmers
- Rules for guaranteed demotion

Rationale

- Whenever a variable is used for the last time it can be moved.
- The compiler easily figures this out in many cases.
- With (N)RVO we already allow move/copy differences to be visible.
- Generalize to any last-uses of variables improves performance.
- Mandatory for some cases to reduce cognitive load.

Proposal contents

- Allow compiler to demote copy to move when a last use has been detected.
- Includes local variables, parameters, rvalue reference parameters and members in rvalue qualified member functions.
- Includes variables provably assigned before next use.
- Require compiler to find some basic cases.

Rules for compilers

- Must find and demote basic cases.
- May find any more cases it can.
- Must not find cases which are not truly last uses.

Rules for programmers

- Must not rely on copies being done when moves are possible – same as for RVO but for types that are never returned.
- Should not rely on moves being demoted to copies outside of the mandated cases.

Rules for guaranteed demotion

- Must be defined together with compiler vendors
- If too complicated programmers will not feel safe to remove move/forward.
- Alternative: ”Always when function has no branches”.
- Alternative: ”Always when function has no loops”.
- Alternative: ”Always outside of loops”.
- Alternative: ”Always outside of loops, if reassigned.”.
- Alternative: ”Always, even if reassigned”.

Relation to other proposals

- P2667: Rule based parameter passing:

  Provides a way to implement all overloads at once.

- P2665: Overload selection:

  Let compiler select by value or by reference.

- Coming up later:

  Labelled types provides named parameters.
  Simplified declarators with all of the type first.