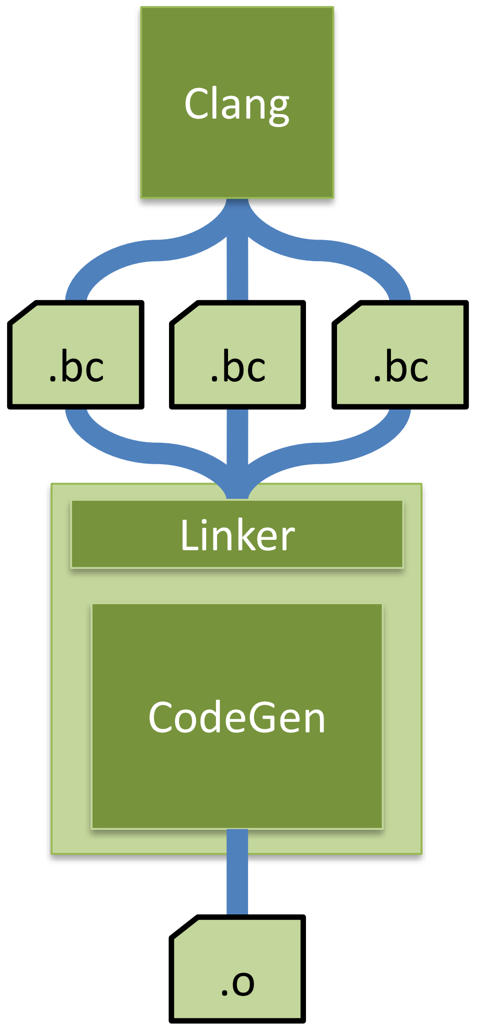
Supporting Regular and Thin LTO with a Single LTO Bitcode Format

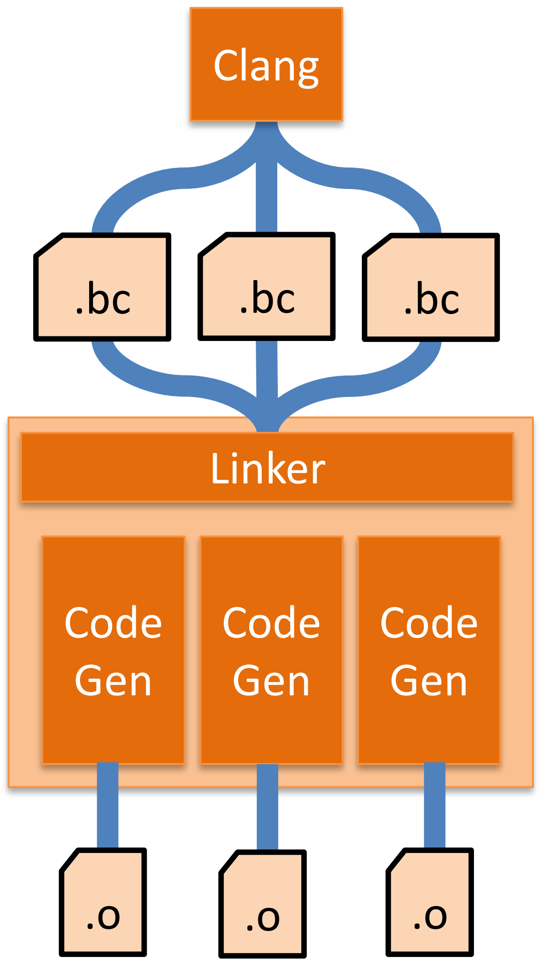
Matthew Voss, Sony Interactive Entertainment

LLVM Developers' Meeting, October 2019

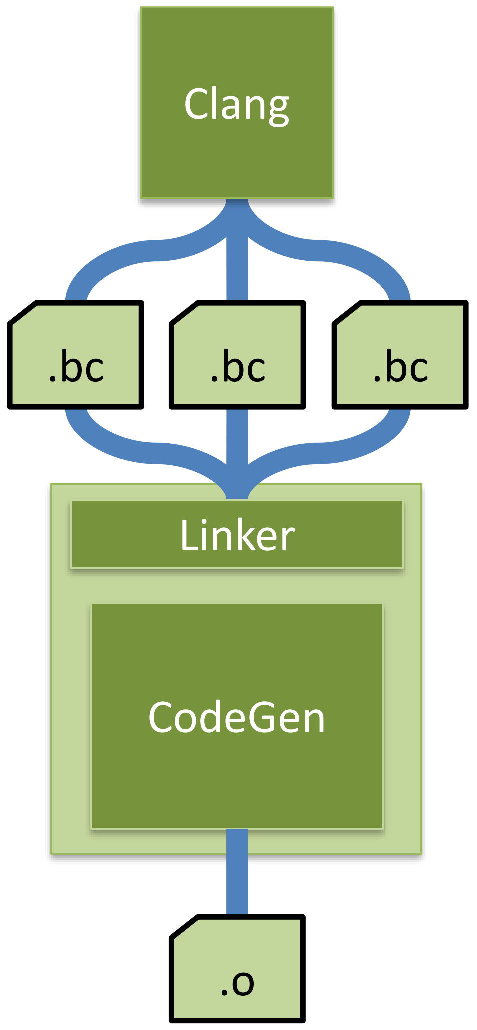
Regular LTO



ThinLTO

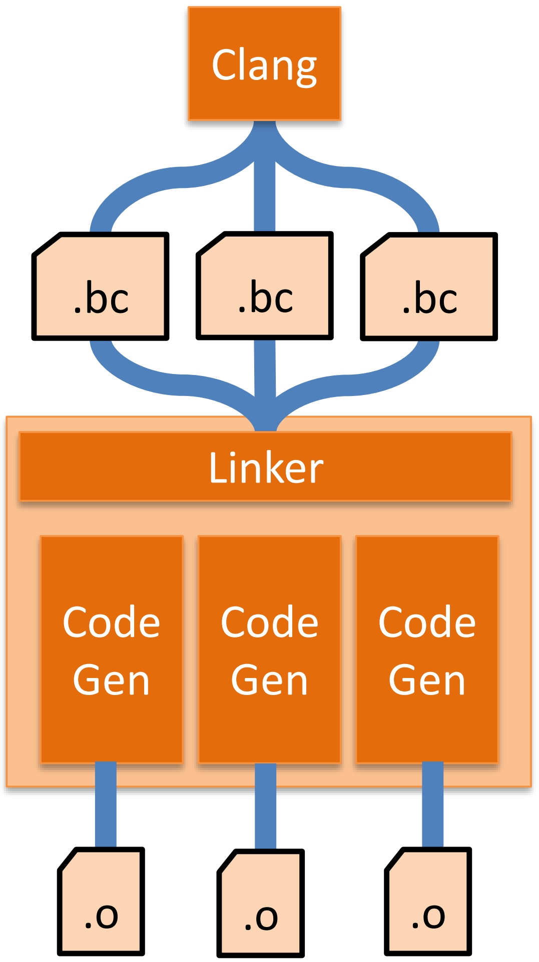


Regular LTO

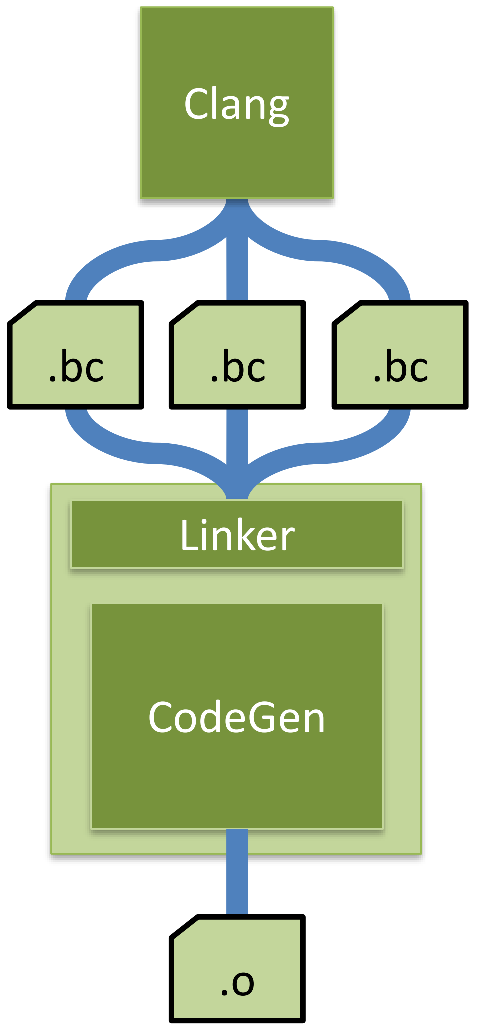


- LTO Backend chosen before link time.

ThinLTO

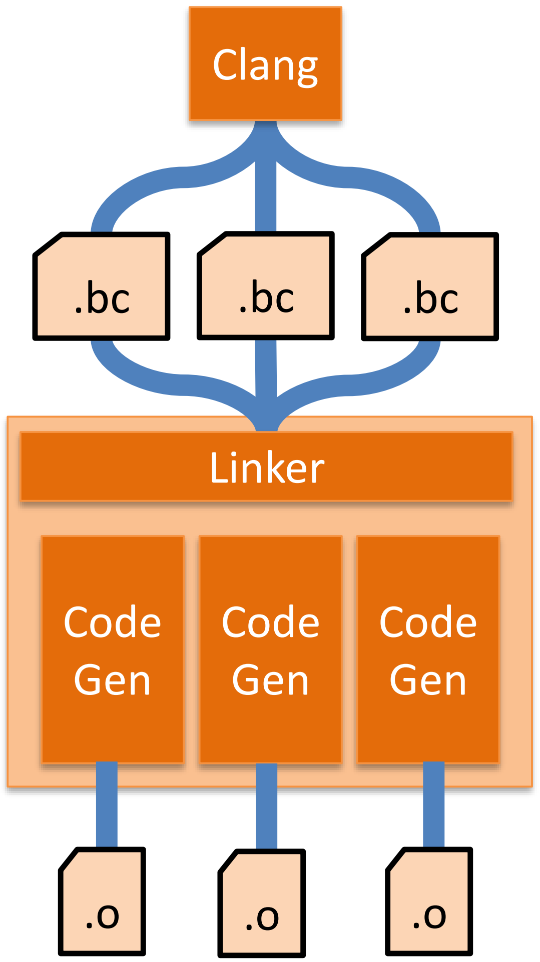


Regular LTO

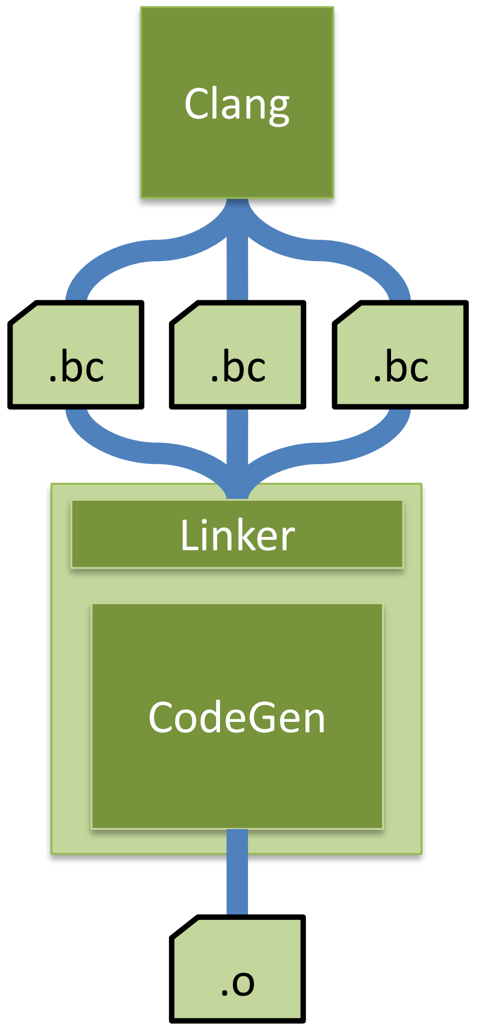


- LTO Backend chosen before link time.
- Thin and Regular Bitcode files can't cross-optimize.

ThinLTO

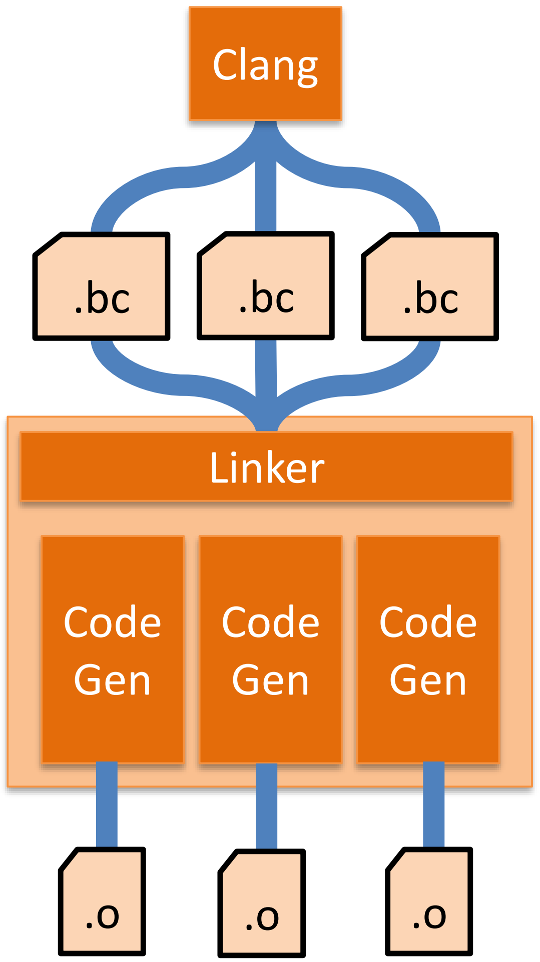


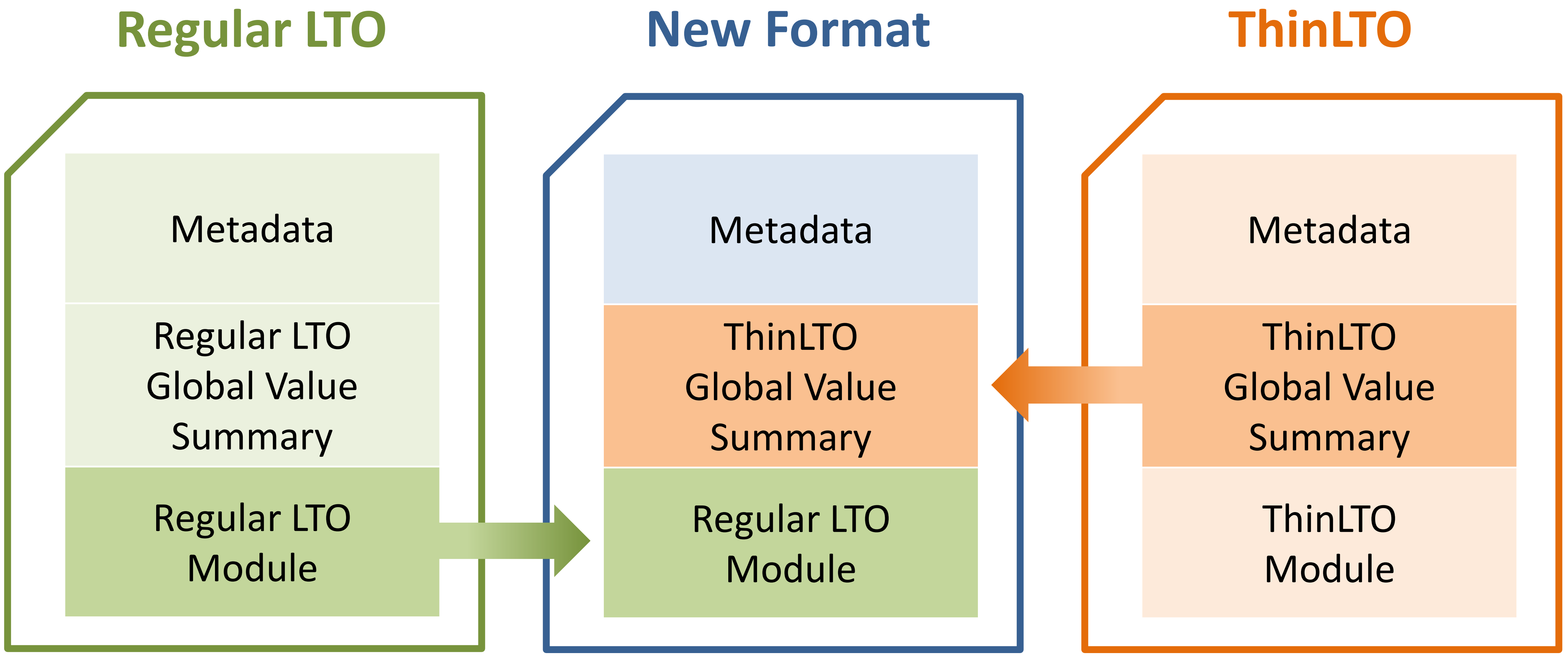
Regular LTO

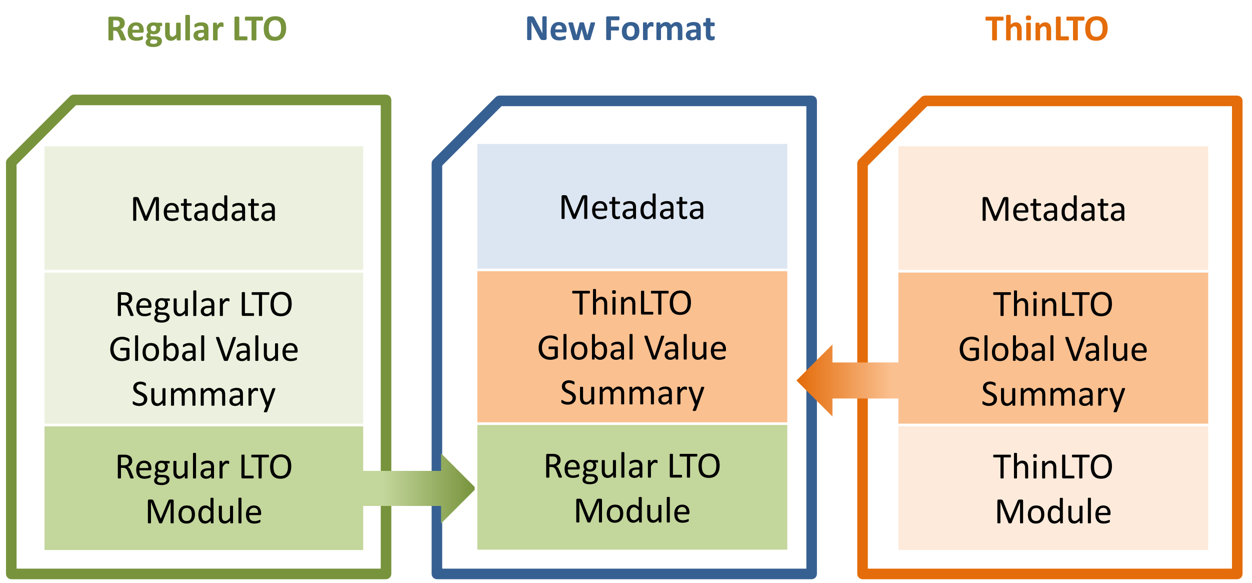


- LTO Backend chosen before link time.
- Thin and Regular Bitcode files can't cross-optimize.
- Library deployment becomes more complex.

ThinLTO




Regular LTO
New Format
ThinLTO
Metadata
Regular LTO Global Value Summary
Regular LTO Module
Metadata
ThinLTO Global Value Summary
Regular LTO Module
Metadata
ThinLTO Global Value Summary
ThinLTO Module



When comparing the new and the existing Regular LTO pipeline, we see *identical build time and code quality.*

ThinLTO Performance Tests

	Clang		
Pipeline	Existing ThinLTO	New ThinLTO	Δ%
Build	2020 sec	2051 sec	+2%
Run	1565 sec	1560 sec	<1%

ThinLTO Performance Tests

	Game 1		
Pipeline	Existing ThinLTO	New ThinLTO	Δ%
Build	145 sec	149 sec	+3%
Frame Time	21.74 ms	21.17 ms	-3%

	Game 2		
Pipeline	Existing ThinLTO	New ThinLTO	Δ%
Build	603 sec	632 sec	+5%
Avg. CPU Usage	35.9%	36.2%	<1%

Summary

- LTO Mode Chosen at Link-time
- LTO Libraries Support Both LTO Modes
- Equivalent Regular LTO Performance
- Small Differences in ThinLTO Build Time
- Equivalent ThinLTO Runtime Performance

Status and Future Work

- Released as a private change in the PS4™ compiler
- Update to use new LTO API (C++ API)
- Testing and performance tweaks
- Release the feature to LLVM for possible integration