A tool for innovative interleaved execution and compilation scenarios: HybroGen

Why should we generate binary code at runtime?

Computing architectures are complex
Complex memory hierarchy, deep pipelines,

Datasets are complex
Sparse data, indirect values, run-time values

Static Compilers fail to reach peak performances
Binary codes should be adapted to run time conditions

How to modify functions?

By using a complete tool which will optimize
the code at run-time.

A complete is a code generator that will:
- generate the binary code at run-time
- use data characteristics as optimizing parameter
- use the available accelerators

Compilation Scenarios

Compilation
(a) Static
(b) Dynamic

Code generation
Code execution
Execution time

Program init
Kernel init
Application context

Heterogeneous architecture
Edge node
dT node

Code generation scenarios:
at the top (a) part, the classical static compilation scheme.
The yellow bar represents the compilation time, the red bars
the kernel execution time.
On the (b) part we represent the interleaving code generation
part and kernel execution part for multiple scenarios we want to implement.

Demonstrators

At program initialization
Code specialization based on
- Program parameters
- Available accelerators
- Fixed data values

At kernel initialization
Code specialization based on
- Function parameters
- Fixed data values

Controlled by program
Code specialization based on
- Program decision (variable precision)
- Security features

Heterogeneous architecture
Code specialization based on
- Multi ISA architectures
- Example on Computational SRAM

Technical informations

- Supported architectures
  - IBM / Power
  - ARM / aarch64
  - RISC V
  - CEA / RISC-V + C-SRAM

Access

https://github.com/CEA-LIST/HybroGen

State of the art on compilation scenarios

Classical compilation scenarios using
iterative profiling compared to our proposition.
1. use static profile gathering and manual optimization,
2. use static profiling with automatic optimization,
3. use dynamic profiling to guide JIT optimization.

Our solution uses:
- dataset characteristics
- and architecture description.