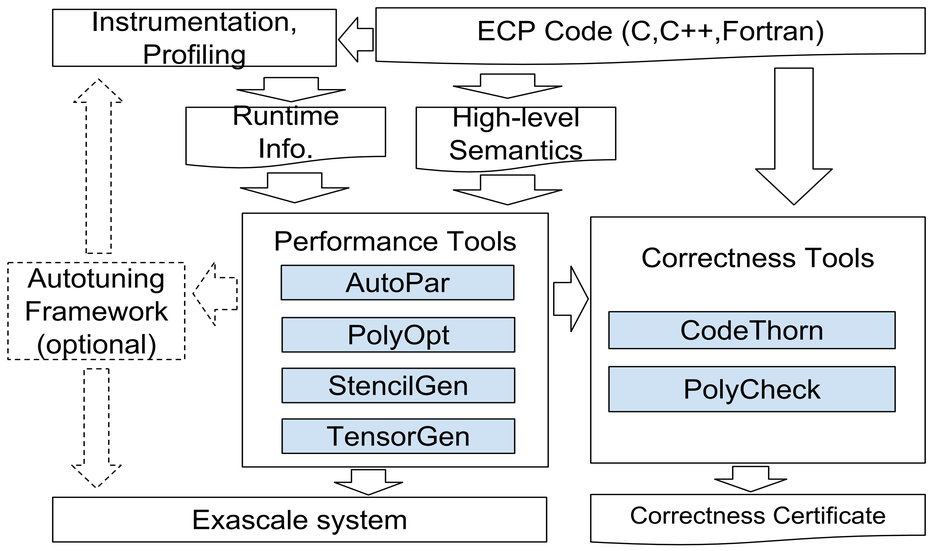
# Exascale Code Generation Toolkit

**PI: Dan Quinlan (LLNL) Partner sites: (PNNL, Ohio State University, Colorado State University )**

## Project Description

The goal of Exascale code generation toolkit is to support the automated generation of code for current and future computer architectures. The toolkit will enable performance portability across a wide range of vendor architectures and compilers. We will deliver source code generation tools required to automate the targeting of algorithm implementations to a wide range of architectures and vendor compilers (and tools).




## Correctness Tools

**CodeThorn** (compile-time verification): verifies iteration-space transformations (including polyhedral transformations) at compile time. CodeThorn checked 1487 variants of PolyOpt generated optimization variants.

| PolyBench/C Benchmark Suite | CodeThorn Verification |
|---|---|
| 2mm, 3mm, adi, atax, bicg, covariance, durbin, fdtd-2d, gemm, gemver, gesummv, jacobi-1d, jacobi-2d, lu, ludcmpm, mvt, seidel, symm, syr2k, syrk, trisolv, trmm, correlation, gramschmidt, fdtf-ampl, floyd-warshall | for 26 benchmarks all 53 optimization variants verified to be correct (1378 variants) |
| doitgen | all fusion variants verified (3 variants) |
| doitgen (tiling variants not generated by PolyOpt) | N/A |
| cholesky | errors found in fusion and tiling (53 variants) |
| reg_detect | all fusion variants verified (3 variants)<br>errors found in tiling (50 variants) |
| dynprog (no variants generated by PolyOpt) | N/A |

**PolyCheck** (runtime verification): a dynamic bug checker for affine programs transformed with arbitrary loop iteration reordering. It supports arbitrary loop transformations (tiling, interchange, etc.) and non-affine transformations (parametric tiling, etc.), and the analysis is not sensitive to the input dataset values. A polyhedral analysis at compile-time is done to create a lightweight checking code to be executed along with the transformed program. PolyCheck complements CodeThorn by enabling runtime verification.

## ECP Application Category and Software Tool Support

| ECP App. Category | Apps. | Relevant Software Tools Support |
|---|---|---|
| RAJA | LLNL ASC apps. | AutoPar, PolyOpt, StencilGen, CodeThorn, PolyCheck |
| Kokkos | | AutoPar, PolyOpt, StencilGen, CodeThorn, PolyCheck |
| Computational chemistry | NWChem | PolyOpt, TensorGen, CodeThorn, PolyCheck |
| Structured Grid | LBL apps. | AutoPar, PolyOpt, StencilGen, CodeThorn, PolyCheck |
| Unstructured Grid | LLNL ASC apps. | AutoPar, CodeThorn |
| Others | | TBD (Will work with selected application teams to determine the most relevant tools for them.) |

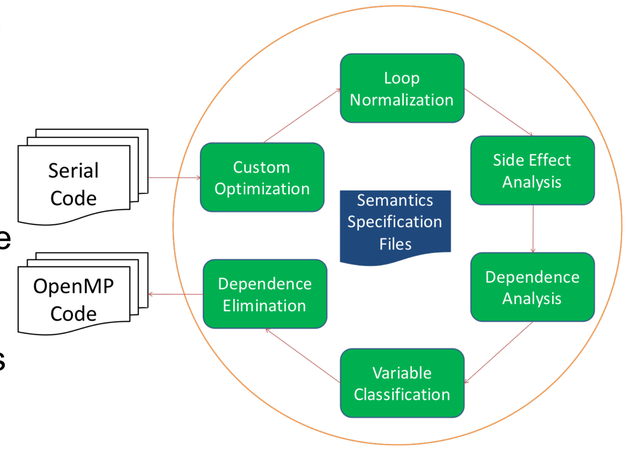
## Optimization Tools

**PolyOpt:** automatically recognizes and optimizes programs with the polyhedral model, a mathematical framework for complex transformations of program regions with regular loops, from C,C++,Fortran codes. Performs analysis and loop transformations, introduces OpenMP directives.

| | Laplacian 2d 64pts | Laplacian 2d 81pts | Laplacian 3d 125pts |
|---|---|---|---|
| **Simple codegen** | 10.53 GF/s | 10.36 GF/s | 5.51 GF/s |
| **+ parallelization** | 43.31 GF/s | 42.51 GF/s | 22.26 GF/s |
| **+ PolyOpt** | **75.17 GF/s** | **75.90 GF/s** | **43.6 GF/s** |
| **DSL input** | **~100 lines** | **~100 lines** | **~ 200 lines** |
| Generated code | 4367 lines | 4649 lines | 8247 lines |

**AutoPar**: automatically inserts OpenMP directives or checks correctness of existing directives.

- Semantics specification files assist parallelization detection.
- AutoPar recognizes more parallelizable loops with semantics info.: **+63.6%** (from 11 to 18 loops) for TinyHydro, **+208.3%** (from 12 to 37 loops) for Ares Loop suite, and **+62.5%** for ALE3D.




**StencilGen**: Takes as input (from PolyOpt) a stencil computation (on dense/regular grids), produces high-performance CUDA/OpenCL code for GPUs and vectorizable C or vector intrinsics code with OpenMP directives for Xeon Phi and multicore platforms.

**TensorGen**: Pattern-specific optimizer for tensors (interfaced with PolyOpt) that generates high-performance CUDA/OpenCL code for GPUs and vectorizable C or vector intrinsics code with OpenMP directives for Xeon Phi and multicore platforms.

Lawrence Livermore National Laboratory · Pacific Northwest National Laboratory · The Ohio State University · Colorado State University

EXASCALE COMPUTING PROJECT

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. IM release number: LLNL-POST-719921