Resilience in Parallel Programming Environments

Christian Engelmann (ORNL)
Geoffroy R. Vallée (Sylabs, Inc)
Swaroop Pophale (ORNL)

Contact: engelmannc@ornl.gov

ORNL is managed by UT-Battelle, LLC for the US Department of Energy

Research sponsored by the Laboratory Directed Research and Development Program of Oak Ridge National Laboratory

Background

- Resilience is a key challenge in extreme-scale computing
  - Less reliable components
  - More components
  - More dependencies
- Heterogeneity adds significant complexity to the extreme-scale hardware/software ecosystem
  - No fine-grain protection domains & resilience at the component level
  - Coarse-grain checkpoint/restart at the job level is standard

Problem Statement

- The burden for providing resilience is currently on the user
  - Global checkpoint/restart is currently the only practical solution
  - Fault-tolerant MPI is experimental, low-level and *maybe* in the standard by 2024 at the earliest
  - There are some local checkpoint/restart libraries for accelerators
- A programming model needs to provide resilience with an easy to use interface to permit wide-spread adoption
  - Have clear error & failure models and corresponding abstractions
  - Hide the complexities of protection domains and resilience strategies
  - Offer efficient resilience with little programming burden

Proposed Solution

- Offer an easy to use and generic Quality of Service (QoS) interface for resilience
  - Use abstract, easy to understand, terms and programming constructs
  - Enable users to define their resilience needs
- Establish QoS contracts between the application and the system
  - Offer resilience QoS contract options for <u>accelerator offload</u>
  - Report contract breaches back to the application
- Embed QoS interfaces, coordination mechanisms and resilience strategies in the OpenMP language and runtime
  - An OpenMP that is resilient to accelerator errors/failures: **rOpenMP**

The Quality of Service (QoS) Approach

- Allow application developers to specify their resilience strategy without focusing on the implementation details
- Create a contract that maps application resilience requirements to the underlying hardware/software capabilities
- Specify the resilience strategy without focusing on implementation details

OpenMP QoS language extensions

- **QoS contract:** A set of QoS parameters that reflect resilience requirements by identifying resilience strategies
- **QoS parameters:** Generic *get/set* interface, using: (1) key/value pairs, (2) bounded values and (3) ranges of values
- **QoS parameter scope:** Code block and related data
- **QoS classes:** Offer coherent sets of parameters that achieve popular resilience strategies

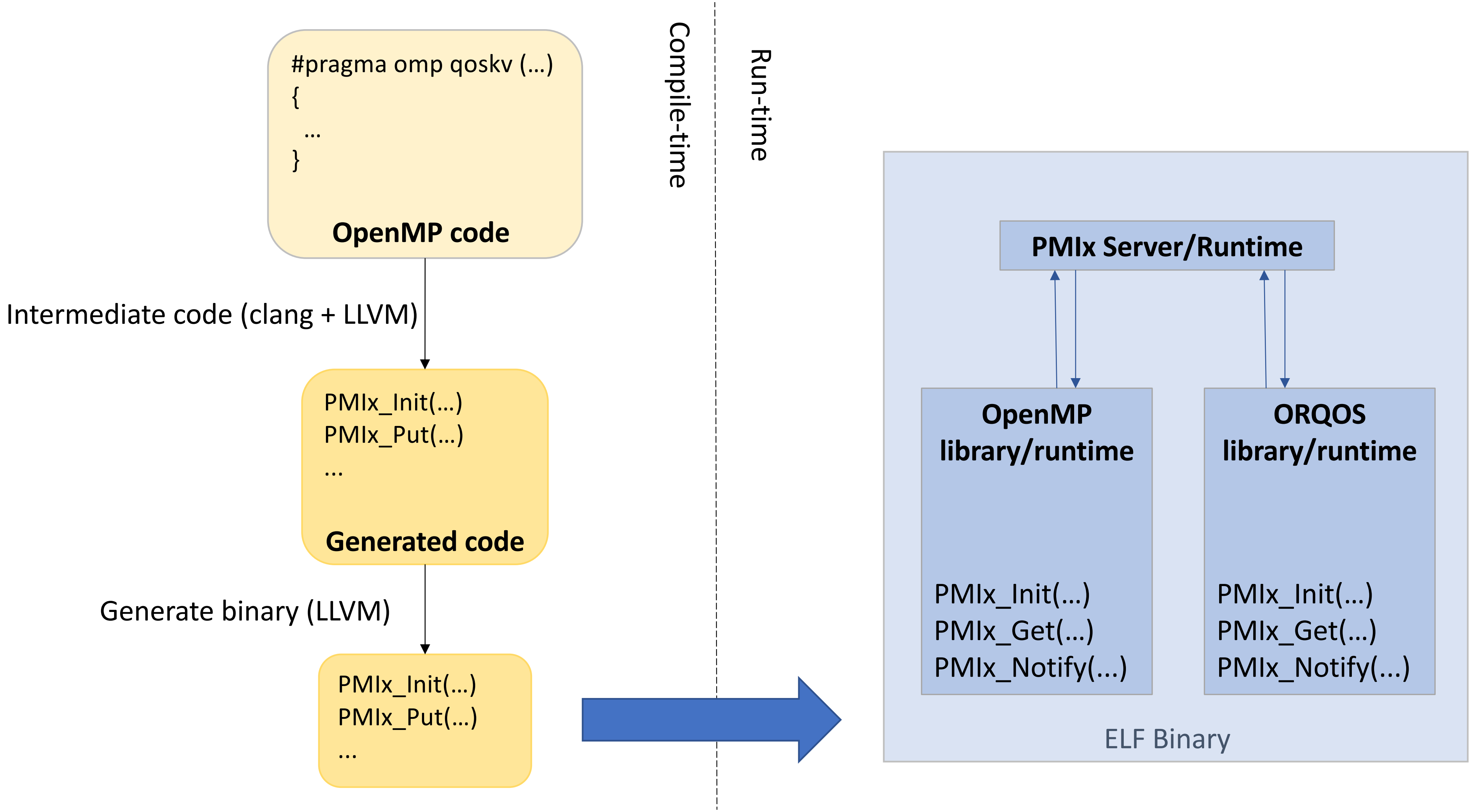
OpenMP QoS language extensions

```
#pragma omp qoskv resilience (TASK_REDUNDANCY, BOOL, TRUE)
#pragma omp qoskv resilience (TASK_REDUNDANCY_FACTOR, INT, 3)
#pragma omp qoskv resilience (TASK_REDUNDANCY_MAJORITY, INT, 2)
#pragma omp qoskv resilience (TASK_REDUNDANCY_COMPARE, BOOL, TRUE)
{
  #pragma omp target ...
  ...
}
```

OpenMP QoS language extensions: QoS classes

```
#pragma omp qoskv resilience (TASK_TRIPLE_REDUNDANCY, BOOL, TRUE)
{
  #pragma omp target ...
  ...
}
```

Compile-time Workflow and Run-time Interactions of the Implemented Prototype using LLVM 7



```
#pragma omp qoskv (…)
{
  …
}
```

OpenMP code

Intermediate code (clang + LLVM)

```
PMIx_Init(…)
PMIx_Put(…)
…
```

Generated code

Generate binary (LLVM)

```
PMIx_Init(…)
PMIx_Put(…)
…
```

Compile-time

Run-time

PMIx Server/Runtime

OpenMP library/runtime

```
PMIx_Init(…)
PMIx_Get(…)
PMIx_Notify(…)
```

ORQOS library/runtime

```
PMIx_Init(…)
PMIx_Get(…)
PMIx_Notify(…)
```

ELF Binary

Resilience Strategies

- Error and failure detection and notification
  - Detections by the device/OS must be reported to the OpenMP runtime
    - Language feature (such as a callback) for application feedback is needed to potentially decide on the course of action (such as if an error is acceptable or not)
  - Detections by the application must also be reported to the runtime
    - Language feature for raising notifications to the OpenMP runtime is needed as well

Resilience Strategies

- Fail-fast and graceful shutdown
  - Detection, notification and controlled termination as soon as possible
  - Graceful shutdown avoids error propagation and failure cascades
  - Also enables proper error/failure reporting and root-cause analysis
  - Should be the default behavior of OpenMP runtime and applications

Resilience Strategies

- Graceful degradation
  - Continue operation after an error or failure at the cost of performance or correctness that is deemed acceptable
  - May mean to continue with less or slower devices
  - Requires runtime support to dynamically remove devices
- Rollback recovery
  - This we know how to do: Save task data and re-execute if needed
    - VOCL-FT has done this for OpenCL-accelerated applications
  - Language feature to limit the maximum number of rollbacks needed

Resilience Strategies

- Redundancy
  - Dual- or triple-redundant execution of tasks
  - Language feature to specify redundancy and type needed
  - Output comparison for error detection and masking
- Redundancy in time
  - Execute the same task at the same time on multiple devices
- Redundancy in space
  - Execute the same task on the same device multiple times

Current Status

✓ Created OpenMP QoS language extensions to describe resilience needs

✓ Developed OpenMP runtime extensions to meet resilience needs

✓ Designed resilience strategies and corresponding protection domains

Christian Engelmann, Geoffroy R. Vallee, and Swaroop Pophale. **Concepts for OpenMP target offload resilience**. *15th International Workshop on OpenMP (IWOMP) 2019*, Auckland, New Zealand, September 11-13, 2019. doi: 10.1007/978-3-030-28596-8 6.

Future Work

→Create QoS policies to meet application needs with strategies

→Create the final prototype and demonstrate its capabilities

- Expand the QoS concept to
  - Other OpenMP features
  - Performance, resilience and energy trade-off
  - MPI and MPI+OpenMP
- Create intent-based QoS extensions to be architecture/strategy agnostic
- Develop an adaptive runtime with self-awareness (AI)

Resilience in Parallel Programming Environments

Christian Engelmann (ORNL)
Geoffroy R. Vallée (Sylabs, Inc)
Swaroop Pophale (ORNL)

Contact: engelmannc@ornl.gov

ORNL is managed by UT-Battelle, LLC for the US Department of Energy

Research sponsored by the Laboratory Directed Research and Development Program of Oak Ridge National Laboratory