Multi-variant User Functions for Platform-aware Skeleton Programming

August Ernstsson
Linköping University, Sweden

Christoph Kessler
Linköping University, Sweden

Skeleton Programming

- High-level parallel programming paradigm
- Skeletons are reusable components which may have efficient parallel implementations
- Skeletons encapsulate parallelism and memory management
- Represent computational patterns (control and data flow) such as:
  - Map  Data-parallel application of user function
  - Reduce  Reduction with 1D and 2D variations
  - MapReduce  Efficient combination of Map + Reduce
  - MapOverlap  Stencil operation in 1D and 2D
  - Scan  Generalized prefix sum

User Functions

- User-provided C++ functions or function templates
- Defined as free functions or C++11 lambdas
- Variadic parameter arity in three aspects:
  - Element-wise access container operands
  - Random access container operands (unrestricted read/write)
  - Uniform scalar operands (i.e., ordinary C++ parameter)
- Multi-variant user functions for targeting specific platforms
  - Multiple elements per user function enabling optimizations
  - Multiple variants for each user function, selectable directly or with SkePU auto-tuning

Tool Flow

- Directory-driven variant lookup, one directory per user function, one file per variant
- SkePU precompiler enables variants and assembles program

Example: Vectorized Addition

```cpp
// Main user function definition
float add(float a, float b) { return a + b; }

// Specialized variant of add
void add(float c, const float *a, const float *b) {
    __m256 av = _mm256_load_ps(a);
    __m256 bv = _mm256_load_ps(b);
    __m256 cv = _mm256_add_ps(av, bv);
    _mm256_store_ps(c, cv);
}
```

Performance

- Early experimental performance evaluation shows over 2x speedup with the selectable vectorized user-function variant

Selected SkePU Publications


SkePU Pre-compiler

```xml
<xml version="1.0" encoding="UTF-8">
<xpdlmodel xmlns="http://www.xpdl.com/xpdl_cpu"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemalocation="http://www.xpdl.com/xpdl_cpu.xsd">
    <xpdl_cpu name="Intel_Xeon_Gold_6130" num_of_cores="16"
num_of_threads="32" isa_extensions="avx avx2">
        <xpdl_cpu prefix="core_group" quantity="16">
            <xpdl_cpu frequency="2.11" unit="GHz"/>
            <xpdl_cache name="L1" size="32" unit="KiB" set="32"/>
            <xpdl_cache name="L2" size="1" unit="MiB" set="16"/>
        </xpdl_cpu>
    </xpdl_cpu>
</xpdlmodel>
```