Parallel Programming with Synthesis 2.0

Shaon Barman  Rastislav Bodik  Sagar Jain
Yewen Pu  Saurabh Srivastava  Nicholas Tung

We are now working on exploiting synthesis throughout the HPC workflow, from design of parallel algorithms to kernel tuning. Ask us about details or come to our talk.

**Domain expert:**
- problem spec → dynamic programming → parallel scan
- O(2^n) → O(n)

**Parallel algorithm expert:**
- example of parallel scan network → SIMD algorithm

**GPU tuning expert:**
- SIMD algorithm → bank conflicts → index expressions

---

**How a classical code generator is developed:**

**Step 1:** Codegen developer writes a skeleton of how the generated code should look like.

**Step 2:** He writes the code generator that produces the desired code variant, by parameterizing the skeleton.

**How a synthesizing code generator is developed:**

**Step 1:** The developer writes only the template. No actual generator is developed. The synthesizer completes the template to make the code variants behave like the spec.

---

**Proposed solution:** Gradually refine the program

**Scalability:**
- Some programs are too hard to synthesize. Hence our proposal to use refinement, which provides modularity and breaks the synthesis task into smaller problems.

**What are the limitations behind the magic?**

**Sketch doesn't produce a proof of correctness:**
SKETCH checks correctness of the synthesized program on all inputs of up to certain size. The program could be incorrect on larger inputs. This check is up to programmer.

**Synthesis steps**

**Try SKETCH online at bit.ly/sketch-language**