Many-Core HW is everywhere

But cannot be programmed well

Traditional Programs run only on a single core

Future massively parallel many-core processor

Embedded in Impala

Impala features a partial evaluator

Partial evaluation is triggered by annotations

fn main() {
  let mut arr: [float] = array(width, height);
  let mut out: [float] = array(width, height);
  let a = 0.2f, b = 1.0f - 4.0f * a;
  let stencil = [
    [0, b, 0],
    [b, a, b],
    [0, b, 0];
  ];
  foreach i in iteration(width, height) {
    out[i] = @apply_stencil(arr, stencil, i);
  }
}

Application-specific code

Applies the stencil to a single pixel

Scheduling & optimization

Target-specific implementations for the iteration function

Compiler exposes NVVM code generation through nvvm function

Also support for vectorization and SPIR

Composition process: Impala ➔ LLVM IR

Mapping for GPU execution:

Annotated LLVM IR ➔ NVVM IR for CUDA

SPIR for OpenCL 1.2

Results for Jacobi Kernel

<table>
<thead>
<tr>
<th></th>
<th>GTX 580</th>
<th>GTX 680</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUDA (hand-specialized)</td>
<td>0.32</td>
<td>0.35</td>
</tr>
<tr>
<td>CUDA (hand-tuned)</td>
<td>0.26</td>
<td>0.23</td>
</tr>
<tr>
<td>Impala (specialized)</td>
<td>0.32</td>
<td>0.35</td>
</tr>
<tr>
<td>Impala (+ tuned)</td>
<td>0.24</td>
<td>0.23</td>
</tr>
</tbody>
</table>

Time in ms for the Jacobi kernel an image of 2048x2048 pixels