Task Coarsening Through Polyhedral Compilation for a Macro-Dataflow Programming Model

Alina Sbirlea, Louis-Noël Pouchet, Vivek Sarkar

Rice University
Ohio State University

January 19, 2014

IMPACT’15
Amsterdam
DFGR and HC

**Overview:**

**User View**
- DFGR
- Tools for graph transformations
  - DFGR Translator
- CnC and other data-flow representations
- User kernels
- Optional sources (e.g., libraries, hand-written HC, CUDA code, etc.)

**Underlying tools**
- Habanero-C (HC) Library
- DFGR Runtime
- Auto-generated C, HC files and Makefile
  - hcc: HC compiler
  - gcc
- Object files (*.o)
- gcc
- Executable
Task Coarsening Through Polyhedral Compilation for a Macro-Dataflow Programming Model

Alina Sbirlea¹, Louis-Noël Pouchet², Vivek Sarkar¹
¹Rice University, ²Ohio State University

DFGR: Data-Flow Graph Representation

Transforming DFGR graphs for task+data coarsening

DFGR regions as iteration spaces: a hierarchy of concepts

Key Features

- Steps are functional
- Item collections implement Dynamic Single Assignment form
- Data type in collections can be arbitrary (w serializers)
- Dependence between steps with step-to-step dependence or via data dependence
- Use tags as unique identifiers for step instances in collections
- Tag values may be known only at runtime or at compile-time
- Natively represent task-level, pipeline and stream parallelism

Overview:

Performance results on 16-core Intel E7330 @ 2.4 GHz

(a) Input sequence sizes: 400k-400k
(b) Input sequence sizes: 10k-10k
(c) Input sequence sizes: 50k-50k

C code

```
A[0][0] = corner[i];
for (int i=1; i<NH; i++) {
    for (int j=0; j<NW; j++) {
        A[i][j] = center(i, j, A[i-1][j-1], A[i-1][j], A[i][j-1], A[i][j]);
    }
}
```

```
\textbf{Figure 1: Smith-Waterman execution for small input sets}
```

```
\textbf{Figure 2: Smith-Waterman execution for large input sets}
```

```
	extbf{Table 3: DFGR for Smith-Waterman.}
```

```
\textbf{Smith-Waterman example}
```

```
\textbf{Dependencies}
```

```
\textbf{Performance Reports.}
```

```
\textbf{Overview:}
```

```
\textbf{Performance results on 16-core Intel E7330 @ 2.4 GHz}
```

```
(a) Input sequence sizes: 400k-400k
(b) Input sequence sizes: 10k-10k
(c) Input sequence sizes: 50k-50k
```

```
\textbf{Overview:}
```

```
\textbf{Performance results on 16-core Intel E7330 @ 2.4 GHz}
```

```
(a) Input sequence sizes: 400k-400k
(b) Input sequence sizes: 10k-10k
(c) Input sequence sizes: 50k-50k
```

```
\textbf{Overview:}
```

```
\textbf{Performance results on 16-core Intel E7330 @ 2.4 GHz}
```

```
(a) Input sequence sizes: 400k-400k
(b) Input sequence sizes: 10k-10k
(c) Input sequence sizes: 50k-50k
```

```
\textbf{Overview:}
```

```
\textbf{Performance results on 16-core Intel E7330 @ 2.4 GHz}
```

```
(a) Input sequence sizes: 400k-400k
(b) Input sequence sizes: 10k-10k
(c) Input sequence sizes: 50k-50k
```

```
\textbf{Overview:}
```

```
\textbf{Performance results on 16-core Intel E7330 @ 2.4 GHz}
```

```
(a) Input sequence sizes: 400k-400k
(b) Input sequence sizes: 10k-10k
(c) Input sequence sizes: 50k-50k
```

```
\textbf{Overview:}
```