THE UNIVERSITY OF ALABAMA ENGINEERING

UAB

CUDACL: A Tool for CUDA and OpenCL Programmers

Ferosh Jacob[1], David Whittaker[2], Sagar Thapaliya[2], Purushotham Bangalore[2], Marjan Memik[32] , and Jeff Gray[1]

[1]Department of Computer Science, University of Alabama
[2] Department of Computer and Information Sciences, University of Alabama at Birmingham
[3] Faculty of Electrical Engineering and Computer Science, University of Maribor

Contact: fjacob@crimson.ua.edu

Challenges to GPU programmers

1. **Accidental complexity**
   - **Improper level of abstraction**

   1. Is it possible to allow programmers to use GPU's as just another tool in the IDE?

   2. A programmer often selects a block of code and specifies the device on which it is to execute in order to understand the performance concerns. Should performance tuning be at a high-level of abstraction, such as that similar to what a Graphical User Interface (GUI) developer sees in a What You See Is What You Get (WYSIWYG) editor?

Challenges to GPU programmers

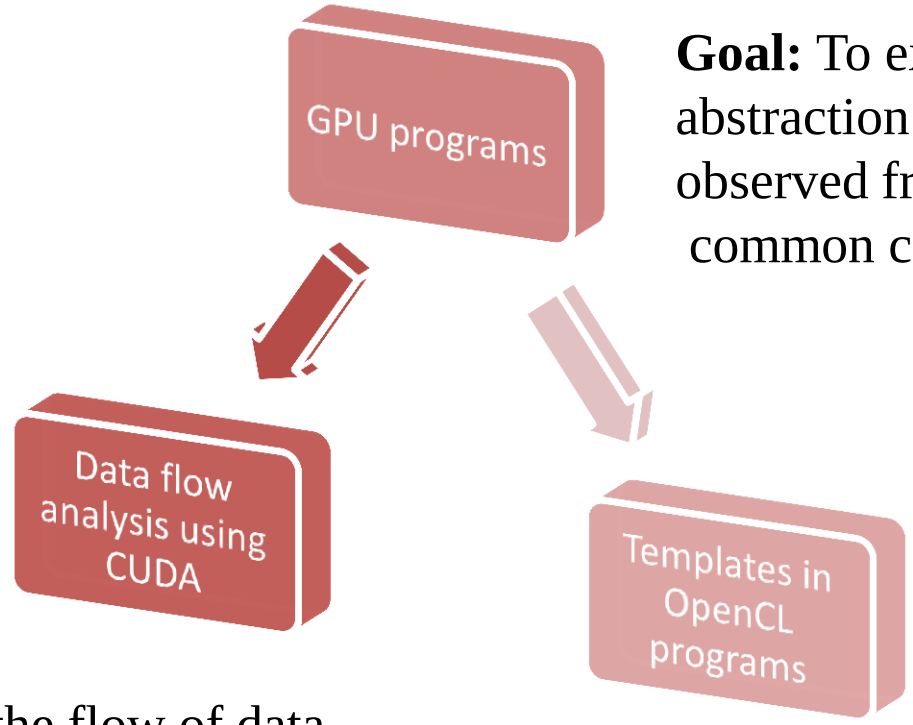
2. Accidental complexity

- Configuration details

1. A simple analysis for CUDA or OpenCL reveals that there are few configuration parameters, but many technical details involved in the execution. How far can default values be provided without much performance loss?

2. How much information should be shown to the user for performance tuning and how much information should be hidden to make programming tasks more simplified?

GPU program analysis



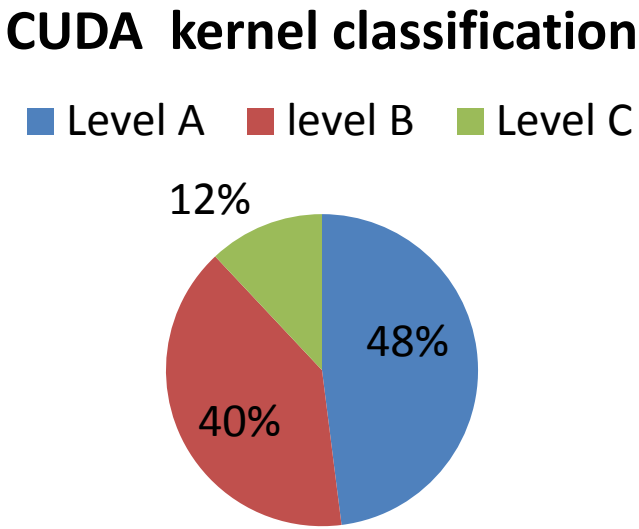
Goal: To explore the abstraction possibilities observed from their common capabilities.

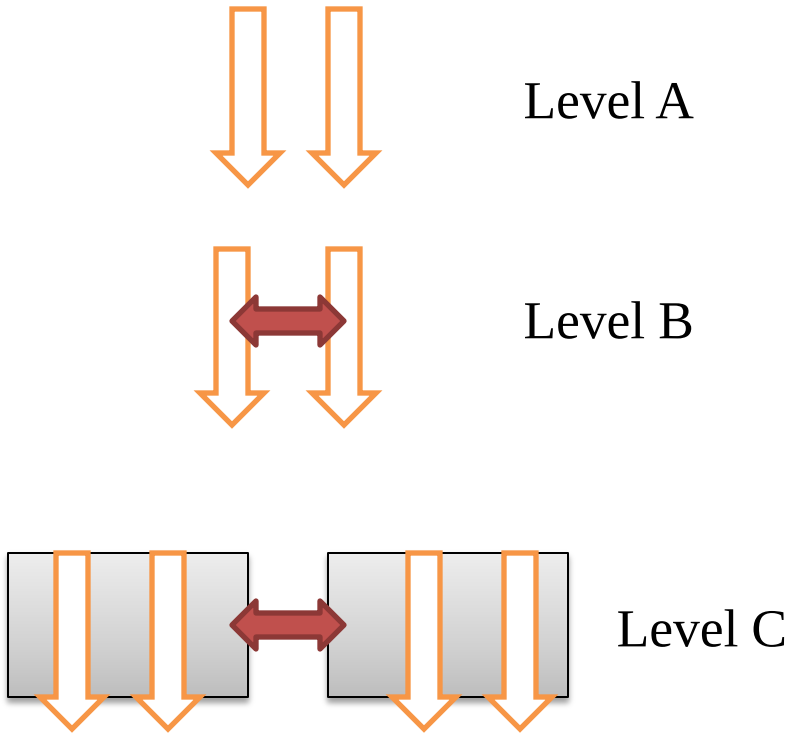
Goal: To study the flow of data from GPU to CPU or vice versa, the flow of data between multiple threads, and the flow of data within the GPU (e.g., shared to global or constant).

Goal: To extract the possible templates from an OpenCL program.

CUDA analysis

For the analysis, 42 kernels were selected from 25 randomly selected programs that are provided as code samples from the installation package of NVIDIA CUDA.





OpenCL analysis

Motivation: In a program oclMatrVecMul from the OpenCL installation package of NVIDIA, three steps – 1) creating the OpenCL context, 2) creating a command queue for device 0, and 3) setting up the program – are achieved with 34 lines of code.

If the steps for a general OpenCL program can be found, templates can be provided that can free the programmer from writing much of the common code manually. Furthermore, one or more steps can be abstracted to standard functions to simplify the development process.

Data: 15 programs were randomly selected from the code samples that are shipped with the NVIDIA OpenCL installation package.

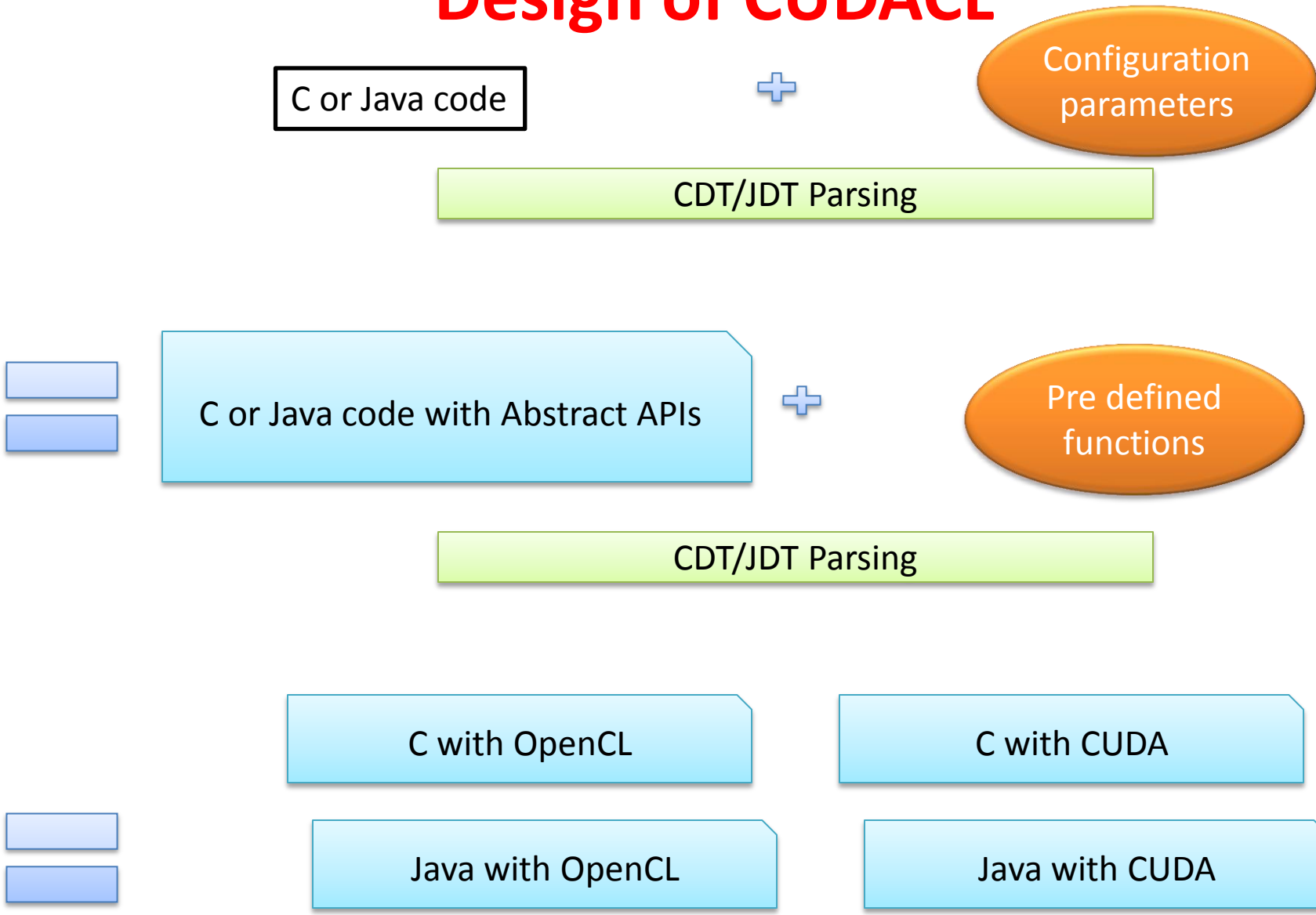
Analysis conclusions

- **CUDA**
  - **Automatic Code Conversion:** There are a fair amount of programs (48%), whose code can be automatically generated if the sequential code is available.

  - **Copy mismatch:** There exists programs (12%) where all the variables in the kernel are not copied. Even though the variables are not copied, memory should be allocated in the host code before the first access.

- **OpenCL**
  - **Default template:** Every OpenCL program consists of creating a context, setting up the program, and cleaning up the OpenCL resources.
  - **Device specification:** Each kernel could be specified with a device or multiple devices in which the kernel is meant to be executed.

Design of CUDACL



CUDACL in action

```
13 for (  j = 0; j < N ; j++ ) {
14
15     c [ j ] = a [ j ] + b [ j ];
16
17 }
```

+

```
startline 13
endline 17
name ArrayAdd
```

=

```
_GPUcopy( a , true);
_GPUcopy( b , true);

_GPUcall("ArrayAdd",params( a , b , N  ));
_GPUcopy( c,false);
_GPUrelease( a );
_GPUrelease( b );
_GPUrelease( c );
```

Host code of Arrayadd

CUDACL in action

```
for (j = 0; j < N; j++) {
    c[j] = a[j] + b[j];
}
```

+

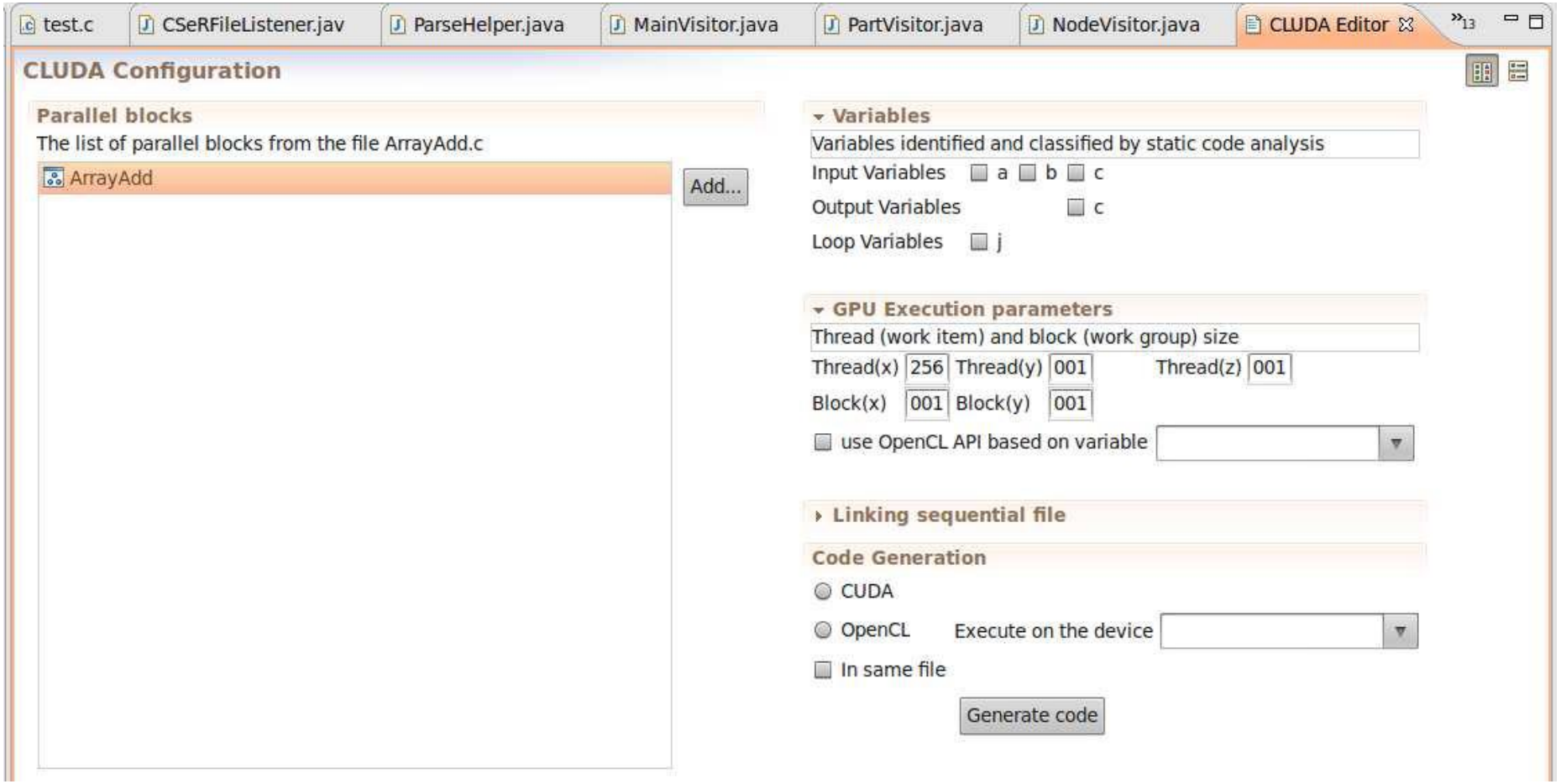
```
startline 13
endline 17
name ArrayAdd
```

=

```
void GPU_Method_ArrayAdd(int[] a, int[] b, int[] c, int N) {
    int j = getGlobalId();
    if ( j < N ) {
        c[j] = a[j] + b[j];
    }
}
```

Kernel code of Arrayadd

CUDACL in action



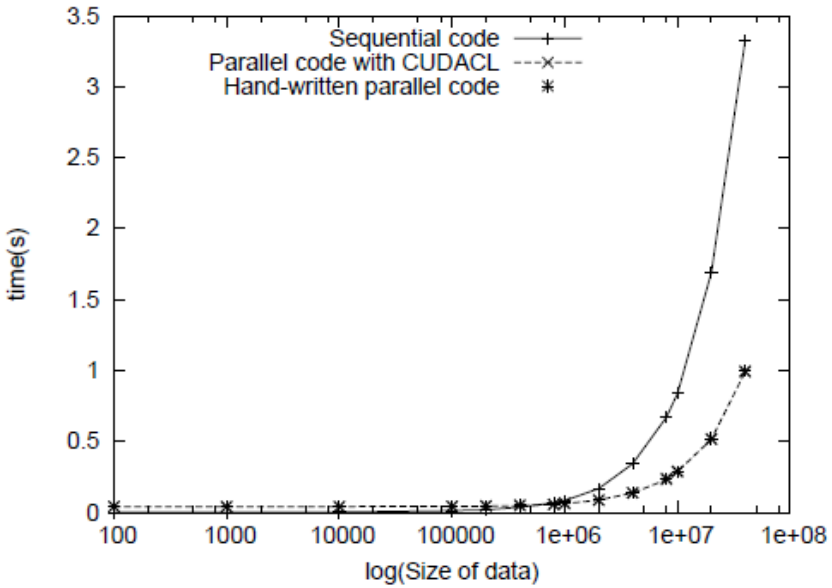
Eclipse plug-in editor for configuration

Case study 1

1. Creation of Octree with CUDA

Octree is a popular tree data structure that is often used to represent volumetric data. Volumetric 3D data is widely used in computer graphics

- **Program written in C language**
- **Level A program**

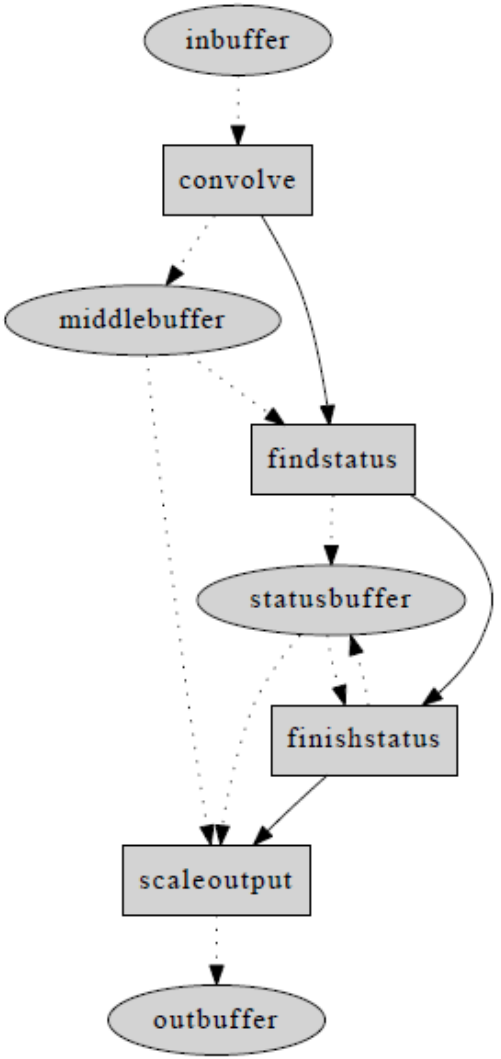


Case study 2

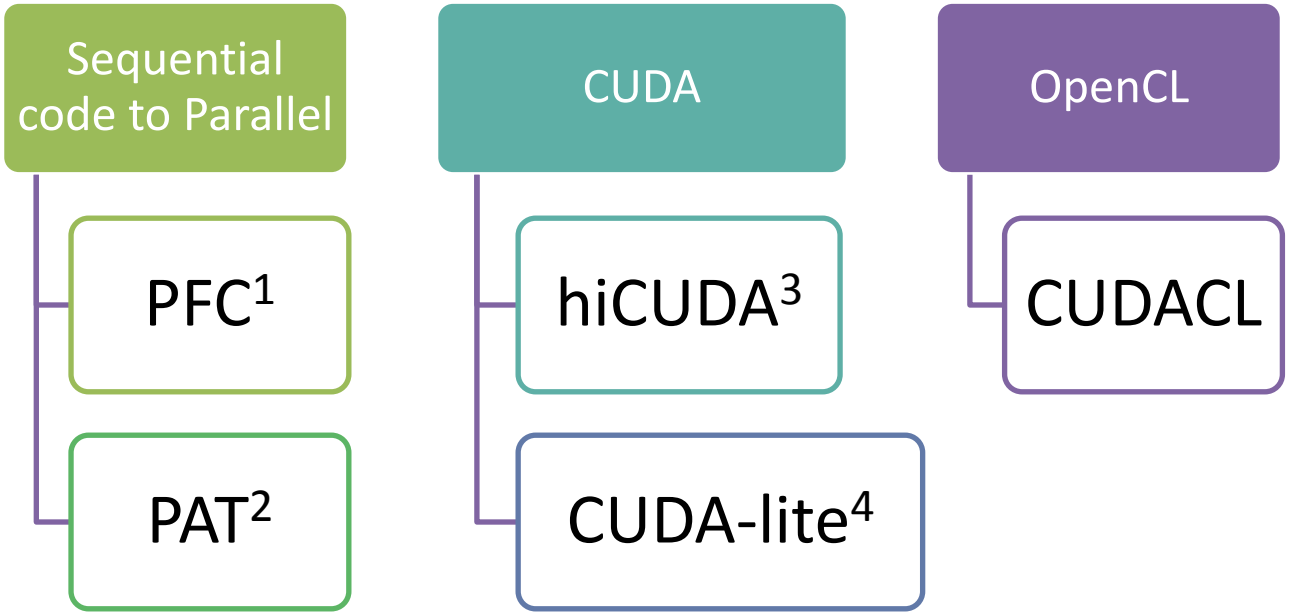
- **Edge detection with OpenCL**

Farmland is found by using a pattern matching algorithm to search for large, flat, contiguous squares and circles on the edge map of the land, which is
created by running a *Sobel* edge detector.

  - **Program written in Java language**
  - **Level C program**



Related works



1. R. Allen and K. Kennedy,“PFC: a program to convert fortran to parallel form,” In Supercomputers: Design and Applications, pp. 186–203, 1984
2. B. Appelbe, K. Smith, and C. McDowell, “Start/Pat: a parallelprogramming toolkit,” IEEE Softw., pp. 29–38, 1989.
3. T. D. Han and T. S. Abdelrahman, “hiCUDA: A high-level directivebased language for GPU programming,” in Proceedings of 2nd Workshop on General Purpose Processing on Graphics Processing Units, Washington, D.C., March 2009, pp. 52–61..
4. S.-Z. Ueng, M. Lathara, S. S. Baghsorkhi, and W.-M. W. Hwu, “CUDAlite: Reducing GPU programming complexity,” in Proceedings of the International Workshop on Languages and Compilers for Parallel Computing,Edmonton, Canada, July 2008, pp. 1–15.

Conclusion and Future work

Conclusion

- A graphical interface is available with default values to configure a GPU parallel block in ‘C’ and Java programs to generate CUDA and OpenCL programs.
- Two case studies were presented to show that CUDACL, and its associated tool support within Eclipse, can be useful for a wide range of applications.

Future work

- As an extension of the work, a Domain-Specific Language (DSL) is being designed to represent the interface for the configuration.
- More examples should be tried to evaluate the tool.

Thank You

- Questions?

Ferosh Jacob[1], David Whittaker[2], Sagar Thapaliya[2], Purushotham Bangalore[2], Marjan Memik[32] , and Jeff Gray[1]

[1]Department of Computer Science, University of Alabama
[2] Department of Computer and Information Sciences, University of Alabama at Birmingham
[3] Faculty of Electrical Engineering and Computer Science, University of Maribor

Contact: fjacob@crimson.ua.edu
http://cs.ua.edu/graduate/fjacob/