The Quasar Value proposition

Quasar is a new programming language that:
- Has a low barrier of entry in GPU acceleration
- Results in a single (HW-agnostic) code base
- Ideal for sharing code
- Results in shorter development cycles
- Allows researchers to focus on algorithms, not on implementation
- Has differentiating development tools
  - e.g. interactive debugging

The tools

- A full-blown IDE:
  - Main target platform
  - Code editor with auto completion windows and help tooltips
  - Interactive command window
  - Real-time image viewers
  - Data inspection and watches

- Performance and profile analysis:
  - Bottle neck detection
  - CPU/GPU usage

- Code analysis and feedback:
  - Performance and profile analysis:
    - CPU/GPU usage
  - Code analysis and feedback:
    - Bottleneck detection

Check

http://quasar.ugent.be

Results

Execution time (ms)

Speed-up factor of 4 to 64

Fast development:
- 2 weeks vs. 3 months for a CUDA implementation of an MRI reconstruction algorithm

Fast execution using the GPU:
- 64 fps vs 2.91 fps for a template matching algorithm

Efficient code:
- 300 lines of Quasar code vs. 2700 lines of C++ code for a registration algorithm

The GPU conundrum for rapid prototyping

For research, an ideal programming language should enable fast development with compact, yet readable code. However, with the increasing complexity of state of the art algorithms, fast execution becomes paramount as well. GPU acceleration would be a major asset. Yet researchers often refrain from GPU acceleration due to the use of low-level programming languages and the perception of a steep learning curve.

The Quasar workflow

Algorithm (quasar code)

Code analysis

Compilation

Runtime

Data

Hardware characteristics

Heterogeneous hardware

CPU

Multi-core CPU

Many-core accelerator

GPU

SOC

Fast development:
- 2 weeks vs. 3 months for a CUDA implementation of an MRI reconstruction algorithm

Fast execution using the GPU:
- 64 fps vs 2.91 fps for a template matching algorithm

Efficient code:
- 300 lines of Quasar code vs. 2700 lines of C++ code for a registration algorithm

An example

Algorithm 1: Pseudo code for primal-dual optimization of active contour segmentation.

\[
\begin{align*}
\triangledown U_k &= \frac{\partial U}{\partial v}, \quad \triangledown V_k &= \frac{\partial V}{\partial v} \\
\nu_k &= \min (U_k, V_k) \\
V_{k+1} &= V_k + \alpha \nu_k \\
U_{k+1} &= U_k - \alpha \nu_k \\
end \end{align*}
\]

Boundary handling

Memory management
- Load balancing
- Scheduling
- Kernel parameter optimization

The tools

- A full-blown IDE:
  - Main target platform
  - Code editor with auto completion windows and help tooltips
  - Interactive command window
  - Real-time image viewers
  - Data inspection and watches

- Performance and profile analysis:
  - Bottle neck detection
  - CPU/GPU usage

- Code analysis and feedback:
  - Performance and profile analysis:
    - CPU/GPU usage
  - Code analysis and feedback:
    - Bottleneck detection

Check

http://quasar.ugent.be

Results

Execution time (ms)

Speed-up factor of 4 to 64

Fast development:
- 2 weeks vs. 3 months for a CUDA implementation of an MRI reconstruction algorithm

Fast execution using the GPU:
- 64 fps vs 2.91 fps for a template matching algorithm

Efficient code:
- 300 lines of Quasar code vs. 2700 lines of C++ code for a registration algorithm

The GPU conundrum for rapid prototyping

For research, an ideal programming language should enable fast development with compact, yet readable code. However, with the increasing complexity of state of the art algorithms, fast execution becomes paramount as well. GPU acceleration would be a major asset. Yet researchers often refrain from GPU acceleration due to the use of low-level programming languages and the perception of a steep learning curve.

The Quasar Value proposition

Quasar is a new programming language that:
- Has a low barrier of entry in GPU acceleration
- Results in a single (HW-agnostic) code base
- Ideal for sharing code
- Results in shorter development cycles
- Allows researchers to focus on algorithms, not on implementation
- Has differentiating development tools
  - e.g. interactive debugging

The tools

- A full-blown IDE:
  - Main target platform
  - Code editor with auto completion windows and help tooltips
  - Interactive command window
  - Real-time image viewers
  - Data inspection and watches

- Performance and profile analysis:
  - Bottle neck detection
  - CPU/GPU usage

- Code analysis and feedback:
  - Performance and profile analysis:
    - CPU/GPU usage
  - Code analysis and feedback:
    - Bottleneck detection

Check

http://quasar.ugent.be

Results

Execution time (ms)

Speed-up factor of 4 to 64

Fast development:
- 2 weeks vs. 3 months for a CUDA implementation of an MRI reconstruction algorithm

Fast execution using the GPU:
- 64 fps vs 2.91 fps for a template matching algorithm

Efficient code:
- 300 lines of Quasar code vs. 2700 lines of C++ code for a registration algorithm