QUASAR

*(GPU Programming Language)*
on GDaaS Accelerates Coding from Months to Days
OUTLINE

1. CAUSE
2. THE OFFER
3. HOW DOES IT WORK
4. DEMO
5. RESULTS
6. CONCLUSION
GPUs are everywhere
Each HW platform requires a new implementation

Long development lead times

Strong coupling between algorithm development and implementation

Low level coding experts are required

... ALMOST EVERYWHERE ???
OBSERVATION

- While breakthrough results are achieved, still limited usage in research
  - Scientific articles mentioning CUDA: 90K
  - Scientific articles mentioning a specific scripting language: 400K-1900K
- A continuous investment in easy access:
  - Optimized libraries: cuFFT, cuDNN, cuBLAS, ...
  - Tools: Digits
    - High level access increases potential market by a factor 7

Limited to specific applications
On GPU desktop as a service (GDAAS)

- HIGH LEVEL PROGRAMMING LANGUAGE
- IDE & RUNTIME OPTIMIZATION
- KNOWLEDGE BASE & LIBRARIES
QUASAR’s Value Proposition

- **Lowering barrier of entry**
  - Days iso of months to get started *
  - Algorithm development decoupled from implementation

- **Faster development**
  - Development cycle reduction by a factor of 3 to 10

- **Efficient code**
  - #lines of code reduction by a factor of 2 to 3
  - Same performance

- **Future proof**
  - Future proof code can also target other GPU models

- **Better algorithms**
  - Distinctive tools for coding and design analysis and exploration

- **Larger market and faster take up**

- **Earlier product launch**

- **Reducing R&D and maintenance costs**

- **Early access to highest performance**

- **Better products**

*Note: The asterisk (*) indicates a footnote or clarification for the term “days iso of months to get started.”*
$Y = \text{sum}(A + B \cdot C + D)$

Code analysis and target-dependent lowering
$Y = \textbf{sum}(A + B \cdot C + D)$

Code analysis and target-dependent lowering

```
function $out$:scalar = __kernel__
kernel$1$(A:vec'col'unchecked,B:vec'unchecked,C:scalar,
    D:vec'unchecked,$datadims$:int,blkpos:int,blkdim:int,blkidx:int)

    $bins$:vec'unchecked=shared(blkdim)
    $accum0$=0.
    for $m=$(blkpos+(blkidx*blkdim)).(64*blkdim)..($datadims-1)
        pos=$m
        $accum0+=($A[pos]+(B[pos]*C)+D[pos])
    end
    $bins[blkpos]$=$accum0
    syncthreads
    $bit=1$
    while ($bit<blkdim)
        if (mod(blkpos,(2*$bit))=0)
            $bins[blkpos]$=($bins[blkpos]+$bins[blkpos+$bit])
        endif
        syncthreads
        $bit*=2
        continue
    end
    if (blkpos==0)
        $out+=$bins[0]
    endif
end

$out$=parallel_do([(blksz.*[1,64,1]),$blksz],A,B,C,D,numel(A),kernel$1)
```
HOW DOES IT WORK?

\[ Y = \text{sum}(A + B \cdot C + D) \]

Code analysis and target-dependent lowering

```c
__global__ void kernel(scalar *ret, Vector_PA, Vector_PB, scalar_PC, Vector_PD, int_P_datadims)
{
    shmem shmem;
    shmem_init(&shmem);
    int blkpos = threadIdx.x, blkdim = blockDim.x, blkidx = blockIdx.x;
    Matrix o35, bins;
    scalar accum0;
    int m, _Lpos, bit;
    bins = shmem_alloc<scalar>(&shmem,blkdim);
    accum0 = 0.0f;
    for (m = (blkpos + (blkidx * blkdim)); m <= _P_datadims - 1; m += (64 * blkdim)) {
        accum0 += vector_get_at<scalar>(_PA, m) +
                  vector_get_at_checked<scalar>(_PB, _Lpos) * _PC +
                  vector_get_at_checked<scalar>(_PD, m);
    }
    vector_set_at<scalar>(bins, blkpos, accum0);
    __syncthreads();
    for (bit = 1; bit < blkdim; bit *= 2) {
        if (mod(blkpos, (2 * bit)) == 0) {
            scalar t05 = vector_get_at_safe<scalar>(bins, blkpos +
                              vector_get_at<scalar>(bins, blkpos);
            scalar t15 = vector_get_at<scalar>(bins, blkpos);
            vector_set_at<scalar>(bins, blkpos, (t15 + t05));
        }
    }
    if (blkpos == 0)
        atomicAdd(ret, vector_get_at<scalar>(bins, 0));
}
```

Automatic generation of a kernel function

Parallel reduction algorithm using shared memory

Compile-time handling of boundary checks

Automatic generation of CUDA/OpenCL/C++ code
QUASAR’S WORKFLOW

DEVELOPMENT

SCRIPTING LANGUAGE → CODE ANALYSIS & LOWERING → COMPILATION → OPTIMAL RUNTIME EXECUTION

- High level scripting
- Compact, readable code
- Ideal for rapid prototyping

Runtime information

INPUT DATA

HARDWARE
QUASAR'S WORKFLOW

DEVELOPMENT

SCRIPTING LANGUAGE → CODE ANALYSIS & LOWERING → COMPILATION → OPTIMAL RUNTIME EXECUTION

- Optimization hints
- Automatic detection of parallelism

Runtime information

INPUT DATA

HARDWARE
QUASAR’S WORKFLOW

DEVELOPMENT

SCRIPTING LANGUAGE → CODE ANALYSIS & LOWERING → COMPILATION → OPTIMAL RUNTIME EXECUTION

OPENCL
NVIDIA CUDA
OPENMP

Runtime information

INPUT DATA
HARDWARE

Cloudalize
GEPURA
QUASAR’S WORKFLOW

DEVELOPMENT

SCRIPTING LANGUAGE → CODE ANALYSIS & LOWERING → COMPILATION → OPTIMAL RUNTIME EXECUTION

INPUT DATA

HARDWARE

Runtime information

HW-setup, load, memory state, scheduling
QUASAR ON GDaaS BENEFITS

1. Anyness (screen, device, GPU power)

2. Hourly model (1-4 GPUs)
   Monthly Quasar licenses

3. Instant app distribution
   Today: M60
   Coming: Multi-GPU
RESULTS

Lines of code

- Filter 32 taps, with global memory
- 2D spatial filter 32x32 separable, with global memory
- Wavelet filter, with global memory

CUDA-LOC: Green, QUASAR-LOC: Red

Development Time

- Filter 32 taps, with global memory
- 2D spatial filter 32x32 separable, with global memory
- Wavelet filter, with global memory

CUDA-Dev Time: Green, QUASAR-Dev time: Red

Execution time (ms)

- Filter 32 taps, with global memory
- 2D spatial filter 32x32 separable, with global memory
- Wavelet filter, with global memory

CUDA-time (ms): Green, QUASAR-time (ms): Red

Implementation of MRI reconstruction algorithm in <14 days using QUASAR versus 3 months using CUDA

More efficient code and shorter development times while keeping same performance
QUASAR APPLICATIONS

Quasar on GDaaS Accelerates Coding from Months to Days
CONCLUSION

- High level scripting language
  - Ideal for rapid prototyping
  - Fast development
  - Maintainable, compact code
- Optimal usage of heterogeneous hardware (multi-core, GPUs)
- Context aware execution
  - Build once, execute on any system
  - Different hardware, different optimization
  - Future proof code

Better, faster and smarter development thanks to GDaaS
Visit us on booth: 826

LEAVE YOUR BUSINESS CARD TO REQUEST YOUR FREE TRIAL OR
GO TO www.gdaas.com