GROPHECY:
GPU Performance Projection from CPU Code Skeletons

Jiayuan Meng
Vitali A. Morozov
Kalyan Kumaran
Venkatram Vishwanath
Thomas D. Uram
Motivation

- Project Performance
  - Without hardware
  - Without readapt code
- GPU
  - A wide variety
  - Various implementations

Current Tools

- GPU Simulations
- Code translation
- Meta-programming
- GPU performance models
- Cross-platform performance predictions

Approach

Skeletonize Transform Project
Skeletonization

Original Code: \( C = A \times B \)

```c
for(r=0; r<Rows; r++)
    for(c=0; c<Cols; c++)
        sum = 0.0;
        for(n=0; n<N; n++)
            sum += A[r][n]*B[n][c];
        C[r][c] = sum;
```

Skeleton

```c
parallel_for(Rows, Cols)
: r, c
    stream n = 0:N
    load A[r][n]
    load B[n][c]
    compute 3
    compute 2
    store C[r][c]
```

Original Code: \( C = A \times B \)

```c
for(r=0; r<Rows; r++)
    for(c=0; c<Cols; c++)
        sum = 0.0;
        for(n=0; n<N; n++)
            sum += A[r][n]*B[n][c];
        C[r][c] = sum;
```

Skeleton

```c
parallel_for(Rows, Cols)
: r, c
    stream n = 0:N
    load A[r][n]
    load B[n][c]
    compute 3
    compute 2
    store C[r][c]
```

Transformation

```c
parallel_for(Rows, Cols)
: r, c
    stream n = 0:N
    load A[r][n]
    load B[n][c]
    compute 3
    compute 2
    store C[r][c]
```

Code Layout

```c
Grid [Rows/BlockRows][Cols/BlockCols]
Block[BlockRows][BlockCols]
allocate A'[BlockRows][StageSize]
allocate B'[StageSize][BlockCols]
for n = 0:N:StageSize
    load A[r][n+thread_id] => A'
    load B[n+thread_id][c] => B'
    sync
for n' = 0:StageSize
    compute 3
    compute 2
    store C[r][c]
    sync
```
Transformation

Skeleton

parallel_for(Rows, Cols) : r, c
{ stream n = 0:N
  { load A[r][n]
    load B[n][c]
    compute 3
  }
  compute 2
  store C[r][c]
}

allocate A'[BlockRows][StageSize]
allocate B'[StageSize][BlockCols]
for n = 0:N:StageSize
  load A[r][n+thread_id] => A'
  load B[n+thread_id][c] => B'
  sync
  for n' = 0:StageSize
    compute 3
    compute 2
    store C[r][c]
  sync
}

Transformation

Skeleton

parallel_for(Rows, Cols) : r, c
{ stream n = 0:N
  { load A[r][n]
    load B[n][c]
    compute 3
  }
  compute 2
  store C[r][c]
}

allocate A'[BlockRows][StageSize]
allocate B'[StageSize][BlockCols]
for n = 0:N:StageSize
  load A[r][n+thread_id] => A'
  load B[n+thread_id][c] => B'
  sync
  for n' = 0:StageSize
    compute 3
    compute 2
    store C[r][c]
  sync
}
### Project

- **Threads per block**
- **Active blocks**
  - No. of blocks
  - No. of warps
  - Cache usage
- **No. of Memory instructions**
  - Coalesced
  - Non-coalesced
- **No. of computation instructions**

### Code Layout

```
Grid [Rows/BlockRows][Cols/BlockCols] {  
  Block[BlockRows][BlockCols] {  
    allocate A'[BlockRows][StageSize]  
    allocate B'[StageSize][BlockCols]  
    for n = 0:N:StageSize  
    load A[r][n+thread_id] => A'  
    load B[n+thread_id][c] => B'  
    sync  
    for n' = 0:StageSize  
    compute 3  
    compute 2  
    store C[r][c]  
    sync  
  }  
}  
```

---

### Project

- **Threads per block**
- **Active blocks**
  - No. of blocks
  - No. of warps
  - Cache usage
- **No. of Memory instructions**
  - Coalesced
  - Non-coalesced
- **No. of computation instructions**

### Code Layout

```
Grid [Rows/BlockRows][Cols/BlockCols] {  
  Block[BlockRows][BlockCols] {  
    allocate A'[BlockRows][StageSize]  
    allocate B'[StageSize][BlockCols]  
    for n = 0:N:StageSize  
    load A[r][n+thread_id] => A'  
    load B[n+thread_id][c] => B'  
    sync  
    for n' = 0:StageSize  
    compute 3  
    compute 2  
    store C[r][c]  
    sync  
  }  
}  
```
Experiment Setup

- **Benchmarks**
  - MatMul (dense linear algebra)
  - HotSpot (structured grid)
  - Green’s Function Monte Carlo
    - IspinEx (sparse linear algebra)
    - SpinFlap (spectral methods)

- **Graphics Processors**
  - FX5600
    - 16 Stream Multiprocessors
    - Mem. Bandwidth: 76.8 GB/s
  - C1060
    - 30 Stream Multiprocessors
    - Mem. Bandwidth: 104.2 GB/s
    - Improved coalescing

MatMul: Effect of Staging and Loop Unrolling

- **Manual Optimization**
- **Projected Performance**
- **Auto-tuning**

MatMul: Effect of Staging and Loop Unrolling

HotSpot: Effect of Coalescing

IspinEx: Explore Different High-level Designs

- \[ C = A^+ B \]
  - A: dense matrix, complex numbers
  - B: sparse matrix, real numbers
SpinFlap: Dealing with Indirect Accesses

- Exchange elements in a matrix

Conclusions and Future Work

- Projection without GPU programming & actual hardware
- Technical Contributions
  - Code skeletonization
  - Transform skeletons
  - Synthesize characteristics
- Future Work
  - Automatic code skeletonization
  - More accurate GPU Performance Modeling

Thank You!

Vitali A. Morozov
Kalyan Kumaran
Venkatram Vishwanath
Thomas D. Uram

Acknowledgements

- Steven Pieper
- Sunpyo Hong
- Richard Vuduc
- Anonymous Reviewers
- Gail Pieper
- Sponsors:
  - The U.S. Department of Energy (contract DE-AC02-06CH11357).
  - Leadership Computing Facility at Argonne National Laboratory
  - Office of Science of the U.S. Department of Energy (contract DE-AC02-06CH11357)