Automatic Parallelization for Graphic Processing Units

Alan Leung
6th Workshop on Compiler-Driven Performance

D. R. Cheriton School of Computer Science, University of Waterloo, Waterloo, ON, Canada

October 22, 2007
Graphical Processing Units (GPUs)

$50 to $500+ massively parallel devices specialized for large throughput

Prototype JIT (JikesRVM) that utilizes GPUs for up to 5.4x speed ups

---

GPU SPMD Computing Model

- SPMD (Single Program Multi-Data) program model
- Gather (array reads from arbitrary index) is supported
- Scatter (array writes to arbitrary index) is NOT supported
GPGPU Example

Method() {
  ...
  for (int k = 0; k < K; k++) {
    ...
    for (int x = 0; x < X; x++) {
      for (int y = 0; y < Y; y++) {
        A[x][y] = ....
      }
    }
  }
} // end Method()

Method() {
  ...
  for (int k = 0; k < K; k++) {
    ...
    Program P = {
      ...
      for (int l = 0; l < L; l++) {
        ....
      }
    } // Apply P to each element of A.
    A = P(A);
  }
} // end Method()
Loop Classification

Method() {
  ...
  for (int k = 0; k < K; k++) {
    ...
    for (int \(x\) = 0; \(x\) < X; \(x\)++) {
      ...
      for (int \(y\) = 0; \(y\) < Y; \(y\)++) {
        // CPU
        C \(x\)[\(y\)] = ....
      }
    }
  }
} // end Method()

Method() {
  ...
  for (int k = 0; k < K; k++) {
    ...
    Program P = {
      ...
      for (int \(l\) = 0; \(l\) < L; \(l\)++) {
        ....
      }
    }
    // Apply P to each element of A.
    A = \(P(A)\);
  }
} // end Method()
Implementation Overview

bytecode

Jikes RVM

OPT_ArrayAnalysis

OPT_GlobalDepAnalysis

OPT_Parallelization

IA32 / PPC Assembler

Rapidmind

Cg

JNI Rapidmind Wrapper

Single Core

Multi Core

NVIDIA

CELL

ATI

OPT_GlobalDepAnalysis

OPT_Parallelization

IA32 / PPC Assembler

Rapidmind

Cg

JNI Rapidmind Wrapper

Single Core

Multi Core

NVIDIA

CELL

ATI
Definition

Given a loop $L$ in the loop nesting tree, $\text{WRITEINDICES}(L)$ is defined as

- A vector of indices for all array writes in $L$’s body
- $\top$, if there is more than one unique vector
- $\bot$, if there are no array writes
Loop Classification: TNLoops

Definition

For a loop $L$ in the loop nesting tree, let $\text{TNLOOPS}(L)$ be a list of all loops that are tightly nested within $L$.
Algorithm PARALLELIZE(loop L):
1: if WRITEINDICES(L) = (i₁, . . . , iₙ) and \{i₁, . . . , iₙ\} ⊆ TNLOOPS(L) and no dependencies are carried by loops i₁, . . . , iₙ and TNLOOPS(L) can be interchanged so the outermost n loops are i₁, . . . , iₙ, in this order then
2: interchange TNLOOPS(L) in this way
3: generate GPU program for body of loop iₙ
4: replace loop i₁ with code to execute GPU program
5: else
6: for each child loop L' of L in the loop nesting tree do
7: PARALLELIZE(L')
- Transferring data to/from GPU is slow
- Too many combinations of CPUs and GPUs available on the market....
Cost Modeling

Definition

\[\begin{align*}
Cost_{\text{cpu}} &= t_{\text{cpu}} \times insts_{\text{cpu}} \times A_{\text{out.size}} \\
Cost_{\text{gpu}} &= t_{\text{gpu}} \times insts_{\text{gpu}} \times A_{\text{out.size}} + \text{copy} \times \sum_{A \in A_{\text{inout}}} A_{\text{size}} + init
\end{align*}\]

- \(t_{\text{cpu}}, t_{\text{gpu}}\): Average time per CPU/GPU instruction (Installation)
- \(insts_{\text{cpu}}, insts_{\text{gpu}}\): Estimate number CPU/GPU inside the loop (Compile time)
- \(A_{\text{out.size}}\): Output array size (Runtime)
- \(A_{\text{size}}\): Size of all the array that needs to be copied to/from the GPU (Runtime)
- \(\text{copy}\): Average time for copying a single float to/from the GPU (Installation)
- \(\text{init}\): Constant initialization factor (Installation)
$$t_{\text{gpu}} = 7.81 \times 10^{-8} \text{ ms / instruction}$$

$$\text{copy} = 1.01 \times 10^{-4} \text{ ms / element}$$

$$\text{init} = 66.57 \text{ ms}$$
Performance Evaluation

- Intel Pentium 4 CPU running at 3.0 GHz with 1 GB of Memory
- NVIDIA GeForce 7800 GPU with 256 MB of GPU memory
- Modified JikesRVM O2
- Rapidmind 2.0.0.6546
Other Issues

- Exceptions (ArrayIndexOutOfBoundsException ... etc..)
- No real multi-dimensional arrays in Java
- Intra-array aliasing
- Non-rectangular arrays
- Branching in bytecode
Conclusion

- Algorithm for detecting GPU executable loops
- Cost model that is close to ideal performance
- 5.4x speed up in one of the benchmarks
Future work

- Other backends (Cell, multi-core CPU)
- CUDA / newer generation of GPUs
- Model improvements
- Limit CPU-GPU texture transfer
Questions

- Email: acleung@plg.uwaterloo.ca
- Thank you!
- Questions and Answers