Computation of Pi using CUDA

Dan Padgett
University at Buffalo
Background

- Want to find a way of utilizing CUDA to help improve times for computing digits of pi

- First attempt used numerical integration
  - Proved to be unhelpful
  - Rate of Convergence
Obstacles

- Original series converged too slowly
- Only double precision supported under CUDA 1.3 compute capability
Overcoming our Obstacles

- Found new series with fast convergence

\[ \pi = \sum_{i=0}^{\infty} \left( \frac{1}{16^i} \right) \left( \frac{4}{8i+1} - \frac{2}{8i+4} - \frac{1}{8i+5} - \frac{1}{8i+6} \right) \]
Next Steps of Action

- Implemented new series
  - Sum converged to full precision in 8 iterations
  - Looked for higher precision library
    - Why has no one written one for CUDA?
    - We will soon find out...
Implementing Higher Precision

- Started with sequential C
- Modeled after IEEE 754 floating point specs
- Left precision as #define variable
- Was able to compute precisions up to 2600 integers per number on a worker node
Stop... CUDA Time!

- Compiled vanilla C source in nvcc CUDA compiler
- Several issues
  - Incompatible low-level memory hacks
  - CUDA functions using structs are inlined
  - Limited CUDA memory, registers
CUDA Difficulties

- Replaced memory hacks with new memory hacks (maximum memset, extracting bits)
- Other issues not satisfyingly resolvable
  - Inlining → 10 minute compile time
  - Executable size neared 1MB
  - Limited shared memory → limited precision
Other Difficulties

• Using higher precisions caused the compiler to simply crash
• Found precision = 12 uses maximum number of CUDA registers
• Nowhere near the capability of the sequential code
Results Cont.

- After the usual 6-8 second CUDA initialization time, code ran far faster than sequential equivalent (up to number parallel processors)
- Asymptotic behavior was as desired, even though the approximation wasn't as good as desired.
Accuracy of Approximation

# Summation Terms

Value of Approximation

3.14159265359000000000
3.14159265358900000000
3.14159265358920000000
3.14159265358940000000
3.14159265358960000000
3.14159265358980000000
3.14159265359000000000

# Summation Terms

3 5 10 15 20 25 30 35

pi approx
CUDA Runtime vs Number of Sum Terms

Log Scale!

# Summation Terms

Time (Seconds)
Conclusions

- CUDA is not well-suited to problems which require a moderate amount of memory
- For pure computation, CUDA offers enormous speedups through parallelism
- \( \pi \approx 3.14 \)