Fine-grained Parallelization of Lattice QCD Kernel Routine on GPU

Lattice QCD kernel routine
- Chromodynamics (QCD) is a theory that describes the strong interactions that bind quarks and gluons.
- For Lattice QCD, the space-time field is discretized into four-dimensional field to compute the actions of the Dirac operator on spinors.
- Most of the execution time is spent in the kernel routine Hopping Matrix.
- Parallelization of Hopping Matrix is based on decomposing the field into odd and even subfields (breaking dependency + outer-loop parallelization).

Objectives
- QCDNext: to propose the architecture of a low cost next generation computer system for highly demanding scientific applications.
- PARA: to study and develop optimization methods to better exploit all parallelism aspects coming from emerging architectures.

Fine-grained parallelization

Unit of work: computing half-spinor in one dimension of the space (~= 100 FP)

Difficult to SIMDize because of conditional execution!

Full spinor version: (16 threads)

Half spinor version: (2*8 threads)

Coarse-grained parallelization

Half spinor version: (2 threads)

Full spinor version: (1 thread)

Performance of fine-grained parallelization

Effect of removing control flow

Control flow effect is reduced by 5% for fine-grained half-spinor version and by 7% for fine-grained full-spinor version.

Effect of multiplexing spinors’ computation

• Full-spinor version has better performance than half-spinor version.
• Multiplexing improves performance up to 54 threads.
• Communication overhead is up to 40%.

Effect of post-processing on GPU

Sending post-processing of Lattice QCD improves performance by reducing the communication overhead to 31% (Lattice size 32^3).

Performance scaling on GPU

• Performance improves with the increase in lattice size.
• QCDNext GPU (Hopping plus post-processing) has a lower max performance (6.8 GFLOPS) but we achieve 6.3 GFLOPS while considering the overhead of communication.

Challenges for Lattice QCD on GPUs:
- SIMDizing code favors coarse-grained parallelism because the computation is not intervened by divergent control flow. This increases the frequency of accessing memory because of the large pressure on register file.
- To hide long memory latencies, large number of threads are needed which reduces the registers allocated per thread within a block.
- Creating fine-grained threads increases heterogeneity in the threads created and makes SIMDizing the code more difficult.

Fine-grained vs. coarse-grained

• Fine-grained full-spinor is the best over all other configurations by 29% to 51%.
• Even though fine-grained half-spinor version has same work granularity, it requires more memory accesses.

Summary

Fine-grained parallelization provides 8.3x speedup over optimized SSE2, at 6.3 GFLOPS.

Fine-grained parallelization allocates more resources per spinor computation while maintaining high degree of parallelism.

Partenaires: INRIA/IRISA (CAPS – Symbiose), CAPS-Entreprise, LPT Paris XI, INRIA (Futurs Labri - Futurs Orsay), Bull, IFP, INT, UVSQ.

Olivier Pene

Khaled Z. Ibrahim

F. Bodin

kibrahim@irisa.fr

bodin@irisa.fr

olivier.pene@th.u-psud.fr

Grand Colloque STIC

November 2007