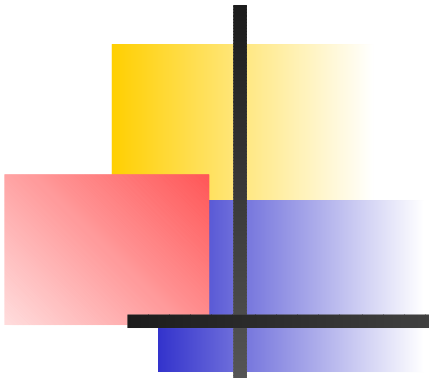

Lattice QCD on GPUs

M. D'Elia (for the I.S. PI12)
Università di Genova & INFN - Genova

Incontro di lavoro della CCR
INFN Napoli 25-27 gennaio 2010

Our Research Goals

We investigate the non-perturbative physics of strong interactions. Numerical simulations of Quantum Chromodynamics (QCD) on a lattice are the best available tool to study the following issues:

- **Confinement/Deconfinement of Quarks and Gluons**
- **Nature of the Deconfinement Transition / QCD Phase Diagram**
- **Relevant to:**
  - Theoretical understanding of QCD
  - Phenomenology of Heavy Ion Collisions
  - Evolution of the Early Universe

Our Numerical Task

Sampling of gauge field configurations by dynamic Monte-Carlo

- **Stochastic variables:** 3×3 **unitary complex matrices** $U_\mu(n)$ **(gauge link variables) associated to each elementary link of a (typically cubic) 4d space-time lattice**
- **Equilibrium distribution:** $\mathcal{D}U e^{-S_G[U]} \det M[U]$

  S_G **is the pure gauge term: takes into account gluon-gluon interactions and is the sum of products of links over elementary loops, e.g. plaquettes**

  $\det M[U]$ **is the determinant of the fermion matrix: takes into account dynamical fermions.** M **is a** $N \times N$ **sparse matrix**

  $N = \mathrm{Lattice\_sites} \cdot \mathrm{N\_of\_Colors} \cdot \mathrm{Dirac\_components} \sim 10^5 - 10^6$

Our Numerical Task

The typical algorithm: Hybrid Monte Carlo

- **auxiliary stochastic variables are introduced:**

  $$\mathcal{D}U e^{-S_G[U]} (\det M[U])^2 \rightarrow \mathcal{D}U\mathcal{D}H\mathcal{D}\Phi^\dagger\mathcal{D}\Phi\, e^{-\mathcal{H}}$$

  $$\mathcal{H} = S_G[U] - \Phi^\dagger (M[U]M[U]^\dagger)^{-1}\Phi + \tfrac{1}{2}\sum_{n,\mu} \mathrm{Tr} H_\mu^2(n)$$

- **Pseudofermion fields Φ and conjugate momenta H_μ updated through a heatbath step (Gaussian distribution)**

- **U_μ and H_μ evolved by Molecular Dynamics eqs. ($d\mathcal{H}/dt = 0$)**

  $$U_\mu(n, t+\delta t) = e^{i\delta t H_\mu(n,t)} U_\mu(n,t),$$

  $$H_\mu(n, t+\delta t) = H_\mu(n,t) + \delta t \dot{H}_\mu(n,t)$$

  Integration errors corrected by a Metropolis accept-reject step

Our Numerical Task

- **The most expensive task is the evalution of $\dot{H}_\mu$, which requires $(MM^\dagger)^{-1}\Phi$, i.e. the inversion of the fermion matrix.**
  A conjugate gradient algorithm is used typically

- **The condition number of $MM^\dagger$ rapidly increases at low quark masses, i.e. approaching a physical quark spectrum**

- **Computational requirements rapidly increase when increasing the physical volume, as necessary when studying the nature of the deconfinement transition**

Why GPUs?

- **Our present computational resources:** mostly apeNEXT (PC clusters and GRID help ...). **About 2 Teraflops available** vs **50-100 Teraflops available to competitor groups** in the world.
- **Mid-long term solutions for italian supercomputing strongly needed!** But we were looking for a soon-available and cheap improvement to accomplish our short term projects:
  We decided to have a bet on GPUs
- **Our efforts come as part of the PI12 experiment:**
  C. Bonati (Pisa), G. Cossu (Pisa $\longrightarrow$ Japan), A. Di Giacomo (Pisa), M. D'E. (Genova), F. Negro (Genova, new entry)

Why GPUs?

The same choice has already been done by other lattice QCD groups around the world, a partial list:

Egri et al. hep-lat/0611022 "Lattice as a video game" (OpenGL implementation...) Wilson kernel > 30 GFLOPs

C.Rebbi et al. (LATTICE08) "Blasting through Lattice Calculations using CUDA" Wilson kernel 100 GFLOPs

Kenji Ogawa (TWQCD) (Workshop GPU supercomputing 2009, Taipei) Wilson kernel 120 GFlops

K. Ibrahim et al. "Fine-grained parallelization of LQCD kernel routine on GPU" Speedup 8.3x on 8800GTX (Wilson kernel)

M. A. Clark et al., arXiv:0911.3191 "Solving Lattice QCD systems of equations using mixed precision solvers on GPUs" up to 150-200 Gflops for Wilson kernel on a GeForce GTX 280

Our implementation

- **Our production code for staggered fermions developed from JLAB's Chroma libraries - $O(10^6)$ code lines.**

  Parts of the code changed into CUDA kernels (thanks G. Cossu)

  Recently C++ code rewritten from scratch - same kernels, $O(10^4)$ lines - to simplify development (thanks C. Bonati)

  Single GPU code by now (less memory for finite T LQCD)

- **Conjugate Gradient solver ($(MM^\dagger)^{-1}\Phi$) takes ~ 90 % of CPU in the serial code: first part entirely put on device**

- **Once solver on device, remaining parts on CPU become relatively expensive: converting more and more code sections into CUDA kernels (matrix exponentiation, staples calculation, . . .)**

Our implementation - fine structure

$M\Phi$ (Dirac Operator) Kernel

- **Each thread reconstructs $M\Phi$ on one site: $U_\mu(n) \times \Phi(n+\hat{\mu})$**
- **Gauge and pseudofermion fields from CPU to threads:**
  - **first two rows of each link to pinned memory**
  - **host $\longrightarrow$ device to global (texture) memory, reordering for coalesced memory access (global $\longrightarrow$ threads)**
  - **pseudofermions to global memory with reordering**
- **Geometry: linear structure for threads (512 threads $\times O(10^3)$ Blocks on largest lattices and on C1060)**

Our implementation - fine structure

- **Thread operations:**
  - **read 3 float4 and 3 float2 from global memory**
  - **reconstruct third row of gauge links (unitary matrixes)**
  - **matrix $\times$ vector and store result in shared memory**
- **Shared memory:** **In present implementation it is not really shared among threads**
- **Precision:** **everything runs in single precision, apart from global sums (residue computation in inverter) which are done in double**

Our implementation - performance

Memory access (bandwidth) still bottleneck. Staggered fermions disfavoured with respect to Wilson fermions on this side (larger memory_transfer/float_operations ratio)

- **Dirac kernel ($M\Phi$) on a NVIDIA C1060 GPU:**
  sustained 60 GFLOPs and 70 GBytes/s (70 % bandwidth peak)).
- **Overall code performance (now, $48^3 \times 4, am = 0.01$):**
  - **1 C1060 $\sim$ 20-30 cores**
  - **Normalized power consumption (to single core): $\sim$ 0.15**
  - **Normalized price (to single core): $\sim$ 0.3**

Our installation

- 2 S1070 units (= 8 C1060) up and running in Pisa
- 4 S1070 units being installed in Pisa
- 2 S1070 units being installed in Genova

- Total of 8 S1070:
  - total investment $\sim$ 65 Keuros
  - with current code, perform as $\sim$ 800 cores
  - with current code, perform as $\sim$ 12 crates of apeNEXT

Perspectives and future improvements/developments

- First production results expected for summer conferences
- Put whole Molecular Dynamics trajectory on device
- Improve memory transfer requirements: 12 $\longrightarrow$ 8 parameters for SU(3) matrixes; single $\longrightarrow$ half precision in parts of the code (Clark et al., arXiv:0911.3191)
- Extend to different lattice discretizations: overlap fermions, improved actions
- Extend to multi-GPUs operation
- Try new GPU architectures