Common Coding Strategies for Lattice QCD

Albert Deuzeman

University of Bern

Lattice 2013
Mainz, 02.08.2013

Introduction

PRACE-2IP WP8

Incrementally update scientific numerical tools to innovative computational solutions.

Includes codes in astrophysics (3), material science (4), climate science (5), particle physics (1) and engineering (5).

Compared to other fields...

- ...the lattice community is *small*.
- ...typical problems rely less on *input data*.
- ...there are *no real 'standard' codes*.

Talk by Claudio Gheller.

Challenges

Lattice QCD codes

- High optimisation levels needed.
- Massive investment of time.
- High developer turnover.
- Divergent goals.
- Limited payoff.

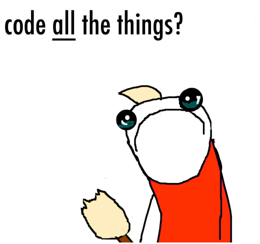
Main challenge

Given the research we want to do, how to make the process of developing high performance codes as **efficient** as possible?

Challenges

Lattice QCD codes

- High optimisation levels needed.
- Massive investment of time.
- High developer turnover.
- Divergent goals.
- Limited payoff.



Main challenge

Given the research we want to do, how to make the process of developing high performance codes as **efficient** as possible?

Developments

Name	Architecture	$R_{\max}$ [TFlops]	Efficiency [MFlops / W]
Tianhe-2	Xeon + Xeon Phi	33862.7	1901.5
Titan	Opteron + Tesla	17590.0	2142.7
Sequoia	Blue Gene/Q	17173.2	2176.5
K Computer	SPARC64	10510.0	830.1
Mira	Blue Gene/Q	8586.6	2176.5

Developments

	Units $[\times 10^6]$	time [years]	architecture
Playstation	103	6	MIPS R3051
Playstation 2	155	6	Sony Emotion Engine
Playstation 3	78	7	Cell Broadband Engine
Xbox	24	4	Intel Pentium III Coppermine
Xbox 360	77	8	PowerPC tri-core Xenon
Playstation 4	??	?	AMD Jaguar APU
Xbox One	??	?	AMD Jaguar APU

Developments

Diverse processor architectures

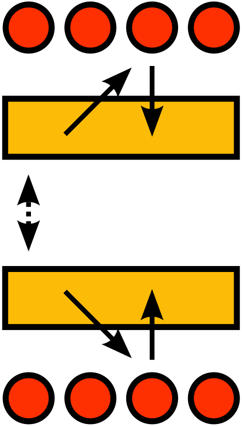
- Quad processing extensions
- SPARC
- ARM

But also, *helpful* new developments on the software side!

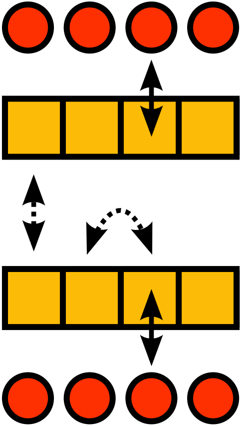
- ILDG format as standard.
- Distributed source control management (*e.g.* `git`, `Mercurial`).
- Development tracking platforms (*e.g.* github, gitorious, Google code).
- Improving compiler quality.
- Novel compiler architecture (`llvm`).

Hybrid codes

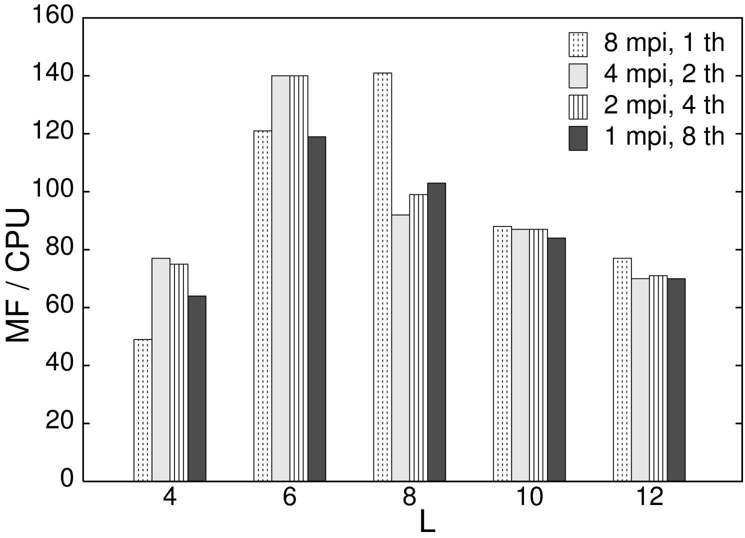
OpenMP (+ MPI)



MPI



Hybrid codes



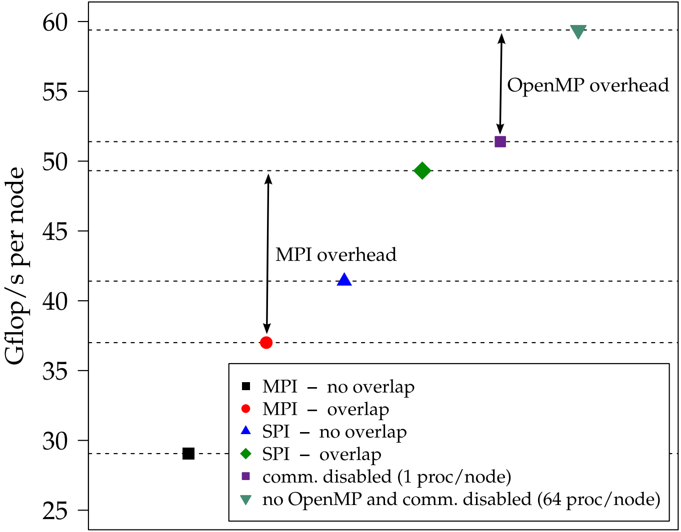
S. Gottlieb and S. Tamhankar, Nucl.Phys.Proc.Suppl. 94 (2001) 841-845

Hybrid codes

- Breaking the balance between threads gives opportunities for performance gain.
- Main gains are in overlapping communication and computation.
- May alleviate bottlenecks due to shared resources between threads.
- By necessity architecture and system dependent, tricky to optimise at lower level.

Talks by Michele Brambilla and Bartosz Kostrzewa.

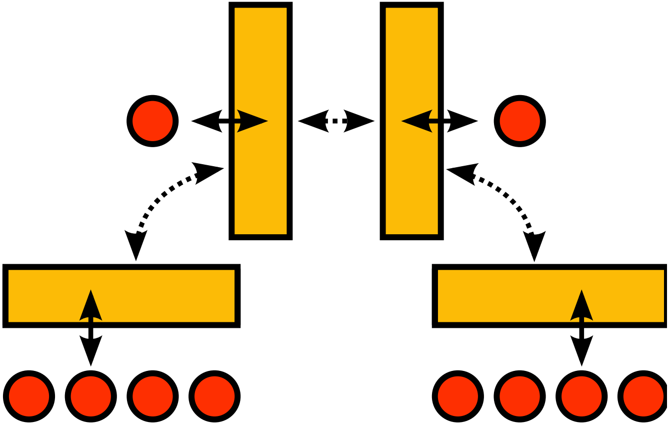
Hybrid codes



Courtesy of Bartosz Kostrzewa

Accelerators

MPI + Accelerator



Frameworks

- **CUDA**
  Platform and code framework for off-loading to Nvidia GPU's.
- **OpenCL**
  Framework and API for heterogeneous computing.
- **OpenMP**
  Compiler directive based API for shared memory multiprocessing.
- **OpenACC**
  A compiler directive based API for shared memory multiprocessing, allowing also access to GPU's.

Talks by Matthias Bach and Pushan Majumdar.

Strategies for efficiency

One Code To Rule Them All

- Not practical!
- Not wanted!
- Not needed?

Rich Ecosystem

- When possible, use existing programs.
- When coding, use existing libraries / interfaces.
- When experimenting, use a high level approach.

Needs *awareness* of the existing options.

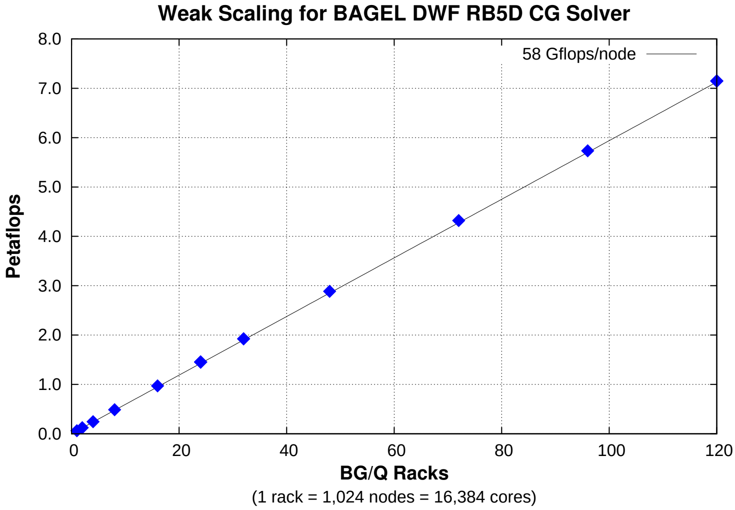
SciDAC

- `Bagel` is the most highly optimised kernel available for the Blue Gene/Q.
- It generates instructions for a range of architectures, however, including
  - Power & PowerPC
  - BG/LP Hummer
  - BG/Q QPX
  - Alpha
  - Sparc
  - C++
- Intel MIC support is forthcoming.

SciDAC

`QLA`	QPX + OpenMP
`QMP`	SPI
`QMT`	pthreads
`QDP/C`	pthreads

SciDAC



Courtesy of Peter Boyle

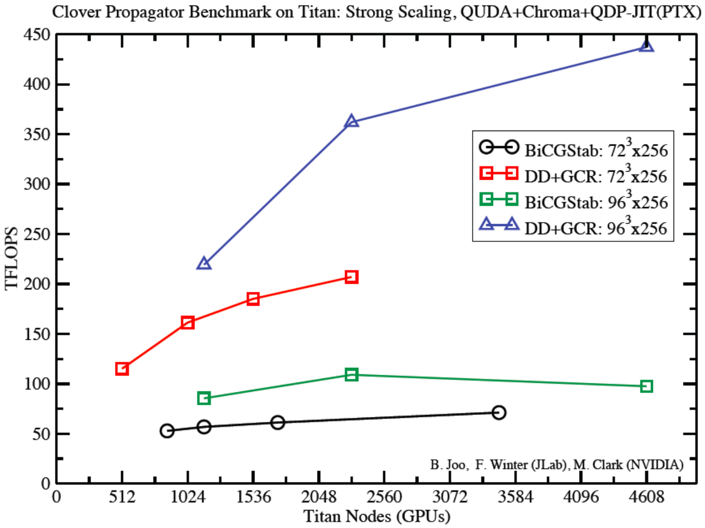
SciDAC

CUDA based GPU acceleration in the SciDAC stack.

- QUDA
  - Implements solvers and performance critical gauge generation routines.
  - Highly optimised: mixed precision methods, autotuning, cache blocking.
- QDP/JIT
  - Moves core QDP functionality to the GPU.
  - Just-In-Time compilation to PTX.
  - Works in conjunction with QUDA.
  - QDP++ as an interface.

Talks and poster by Mike Clark, Alejandro Vaquero and Frank Winter.

QUDA



Courtesy of Mike Clark and Balint Joo

Traditional codes

- Codes within the SciDAC stable have mainly improved through the developments within the underlying libraries.
- Integration of QUDA is available within *e.g.* Chroma, CPS, BQCD and the MILC code.
- Chroma, in particular has seen a lot of work on efficient threading, with optimisations for the Xeon Phi.
  - Pioneering implementations of Xeon Phi tuned invertor libraries show performance on par with GPU codes.
  - Code should be useful for X86 libraries and could eventually be ported to the Blue Gene/Q.
  - Specific Blue Gene/Q optimisation is a secondary target for the moment.

Talk by Chulwoo Jung.

Traditional codes

- `IroIro++`
  - Uses several IBM Japan developed libraries for message passing, threading and linear algebra.
  - Integrates Bagel for high performance inversions.
  - Set to be used in production by JLQCD, publicly available soon.
- `Bridge++`
  - A modern concept code, written for extendability, readability and portability.
  - Only MPI fully implemented, but support for OpenMP, pthreads and OpenCL is being worked on.
  - Architecture specific tuning is underway, not yet mature.
  - Development priorities guided by user base, focus on new features.

Talk and poster by Guido Cossu and Satoru Ueda.

Traditional codes

- `OpenQCD` Mainly algorithmic work (open boundaries), but some additional architecture tuning (AVX).
- `PLQCD` New Wilson operator inverter library with hybrid parallelisation.
- `tmLQCD`
  - Coding work has been focusing on efficiency on the Blue Gene/Q.
  - Use of SPI, QPX intrinsics and hybrid parallelisation have dramatically increased efficiency.
  - Somewhat limited from-scratch CUDA support is available.

Talks and poster by Abdou Abdel-Rehim, Bartosz Kostrzewa, Stefan Krieg, Stefan Schaefer and Carsten Urbach.

High Level Languages

Scripting languages are the natural medium for utilty and analysis codes.

- `qcd_utils`
  - Written in python and maintained by *Massimo di Pierro*.
  - Utilities for fetching and manipulating data files, analysis and visualisation.
- `QLUA`
  - Analysis code using `Lua` as glue around the SciDAC libraries.
  - Functions as a Domain Specific Language: flexibility with decent performance.

High Level Languages

The flexibility of scripting languages can also be leveraged for flexible Monte-Carlo codes.

- `QCL`
  - Designed for decent performance in a range of exotic scenarios.
  - Actions described as paths in high level Python, manipulated symbolically for efficiency and then translated into OpenCL.
- `FUEL`
  - Partner to `QLUA`, providing an API on top of the SciDAC libraries.
  - Designed for BSM physics and modern algorithms.
  - Preliminary indications of especially good performance for $N_c \neq 3$.

Talk and poster by Meifeng Lin and Massimo di Pierro.

High Level Languages

The underlying concept of flexibility can be taken different routes than interfaces from higher level languages.

- `QIRAL`
  - Uses LaTeXexpressions as native input.
  - Translated to formal logic (Maude), which is converted into C.
  - Uses OpenMP for shared memory parallelism.
  - `arXiv:1208.4035`
- `parmalgt`
  - Objects on single spacetime points in a D-dimensional lattice as a basis, not specific to QCD.
  - Hybrid parallelism through threads and MPI.
  - Templates and C++11 features for efficiency and flexibility.

Talks by Michele Brambilla, Mattia dalla Brida and Dirk Hesse.

Summary

- The hardware landscape is changing rapidly and adjusting is a challenge.
- A massive amount of work is being done and we need to use this.
- Libraries are perhaps the most promising route to synergy.
- There is much potential in using existing libraries as interface definitions.
- High level languages, offering agility and ease of use, are starting to be explored.

Acknowledgements

- Abdou Abdel-Rehim
- Matthias Bach
- Peter Boyle
- Michele Brambilla
- Mike Clark
- Guido Cossu
- Carleton DeTar
- Massimo DiPierro
- Claudio Gheller
- Balint Joo
- Chulwoo Jung
- Bartosz Kostrzewa
- Meifeng Lin
- Pushan Majumdar
- James Osborn
- Stefan Schaefer
- Satoru Ueda
- Carsten Urbach
- Alejandro Vaquero
- Frank Winter