Computational Requirements of Lattice QCD Applications

Hubert Simma

21.3.2011

Zeuthen Cluster User Meeting

Plan:

- ❑ The questions
- ❑ Tentative answers for the case of LQCD
- ❑ Advanced analysis

Computational Requirements

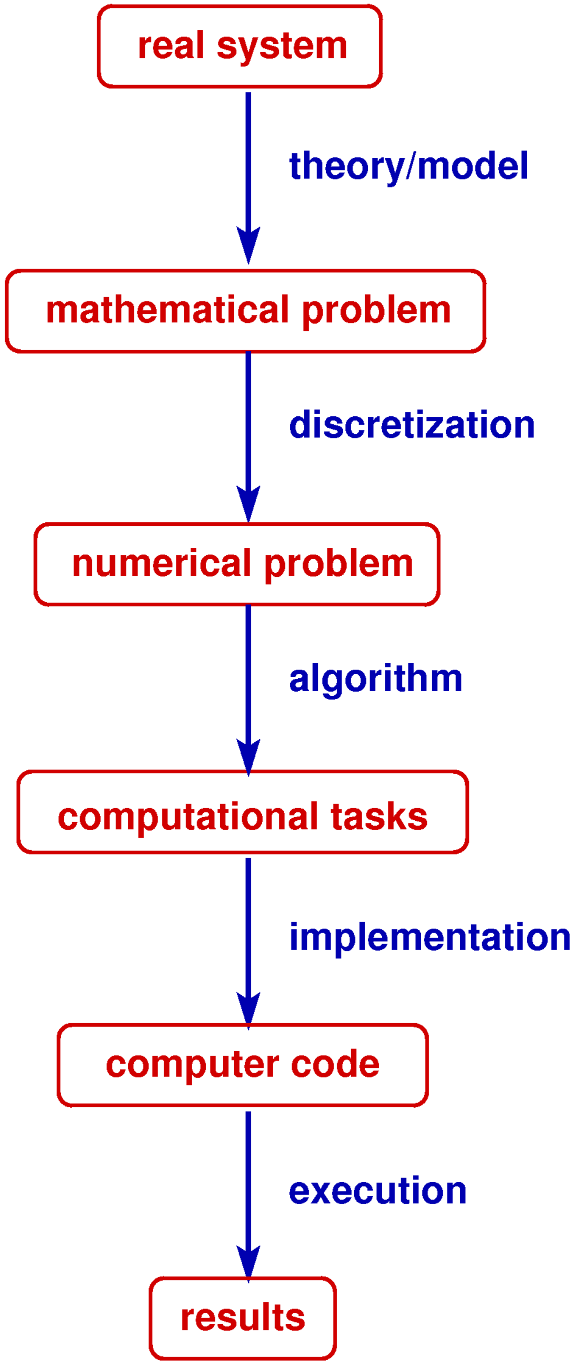
Aim: investigate and evaluate relation between

applications and **architectures**

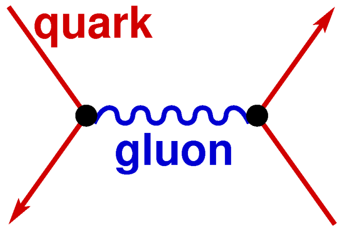
Questions: [Bertinoro 2006]

- Application domain
- Algorithms and computational kernels
- Basic computational requirements
  - problem size
  - storage requirements
  - computing requirements
- Advanced analysis
  - parallelism
  - communications
- Computer Architectures

Scientific Computing Applications

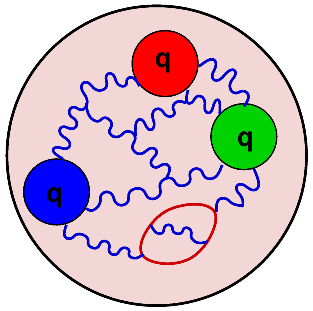


QCD: Strong interactions of quarks and gluons



- Relativistic quantum field theory
  - 4d space-time
  - path integral over all field configurations

$$\langle O \rangle \sim \int_{\text{fields}} O \cdot e^{-S_E}$$



- Coupling grows at low energies (large distances)
  - non-perturbative methods
  - fundamental fields $\neq$ observable bound states
    (quarks, gluons) $\quad$ (p, n, π, . . .)

- All predictions fixed by only $1 + N_f$ parameters!

$$\mathcal{L}_{\text{QCD}}(g_0, m_0^{(u)}, m_0^{(d)}, \ldots) = \frac{1}{4} G_{\mu\nu} G_{\mu\nu} + \sum_{f=1}^{N_f} \overline{\psi}(i \not{D} - m_0^{(f)})\psi$$

Scientific Key Problems in (lattice) QCD

- Precise determination of the fundamental QCD parameters (e.g. from hadron masses) and comparison with determinations from different processes
- Spectroscopy of (well-known and exotic) Hadrons
- Ab-initio computation of "hadronic matrix elements" that enter in the interpretation of experimental data from many electroweak processes, e.g.

$$\Gamma(B \to \tau\nu) \sim |V_{ub}|^2 \cdot f_B^2$$

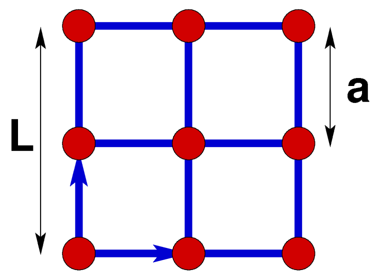
vs.

$$d\Gamma(B \to \pi\ell\nu) \sim |V_{ub}|^2 \cdot (\text{form factors})^2$$

- QCD thermodynamics (quark-gluon plasma)

Discretization of QCD on the Lattice

Finite 4-d euclidian space-time lattice:



quark field $\psi(x)$: 12 complex numbers at each site

gauge field $U(x,\mu)$: 9 complex numbers at each link

e.g. lattice size $\mathbf{V = L^3 \times T = 64^3 \times 128}$ ➔ $\mathbf{0.3 \cdot 10^8}$ sites

Lattice Action:

$$\mathcal{S} \;=\; S_g(U) \;+\; \overline{\psi}\, M(U)\, \psi \;+\; O(a)$$

$\Rightarrow$ rigorous non-perturbative definition of the QCD path integral

Theoretical research topics:

- improved actions to reduce discretisation errors, e.g. $O(a) \to O(a^2)$
- preserve/restore symmetries of continuum theory

LQCD Algorithms

Monte-Carlo method:

Estimate path integral by

$$\langle O \rangle \;\rightarrow\; \frac{1}{\#U} \sum_{U} O(U)$$

with gauge configurations U generated according to distribution (unquenched)

$$P(U) \;\sim\; e^{-S_g(U)} \cdot det M(U)$$

Algorithmic task:

Generate independent configurations

$$U_n \rightarrow U_{n+1} \rightarrow U_{n+2} \rightarrow \cdots$$

Algorithmic research topics:

- computation of the fermion determinant, e.g. $det\, M(U) \sim \int D[\phi]\, e^{-\bar{\phi}(M^{\dagger}M)^{-1}\phi}$
- reduction of correlations between subsequent configurations

Key LQCD Kernels

Typically more than 80 % of CPU time is spent for

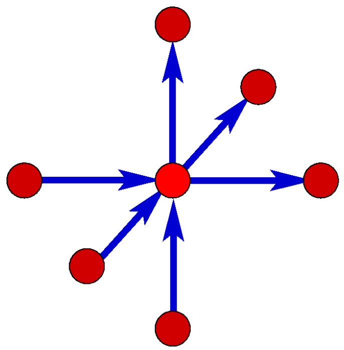
solve $M(U)\phi = b$

➔ Krylov-space methods, polynomial approximations, . . .

➔ Pre-conditioning (e/o, Schwarz alternating procedure, SSOR, . . .) and deflation

Wilson-Dirac Operator:

- sparse
- regular



$$
\begin{aligned}
[\mathbf{M}\phi]_x &= (D_\mu \gamma_\mu + m + a\cdots)\phi \\
&\sim \phi_x - \kappa \sum_{\mu=\pm 1}^{\pm 4} \mathbf{U}_{\mu,\mathbf{x}} \otimes (1-\gamma_\mu)\cdot \phi_{x+\hat{\mu}}
\end{aligned}
$$

➔ 1320 floating-point operations per lattice site

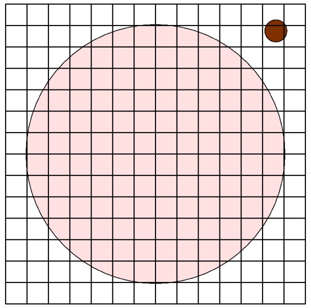
LQCD Computing Requirements

"The computational requirements voiced by these European groups sum up to more than 1 sustained Petaflop/s by 2009."

["Scientific case for European HPC infrastructure HET"]

Multiple simulations with different parameters are required to gain full control on systematic errors from

- extrapolation to continuum limit $a \to 0$
- approach to physical (light) quark masses $m_q \to 0$
- finite volume: L^4
- heavy quarks (or other big scale differences)



LQCD Computing Requirements (cont.)

CPU-Cost: current physics projects at level of several tens of Tflops $\times$ year

$$N_{flop} \;\sim\; L^{5\ldots6} \cdot \left(\frac{1}{a}\right)^{6\ldots7} \cdot \left(\frac{1}{m_q}\right)^{1\ldots2}$$

➔ mainly SIMD floating-point arithmetics (64-bit and 32-bit)

➔ predictable control flow (loops)

Storage (main memory):

$$S = \left(\tfrac{L}{a}\right)^4 \cdot (N_\phi \cdot 12 + N_U \cdot 36) \text{ complex words}$$

➔ strongly algorithm-dependent: $N_\phi = 6 \ldots 200$ and $N_U = O(3)$

➔ predictable access pattern (index tables)

Computer Architectures used for LQCD

- Commercial Super-Computers
  Cray, BlueGene, . . .

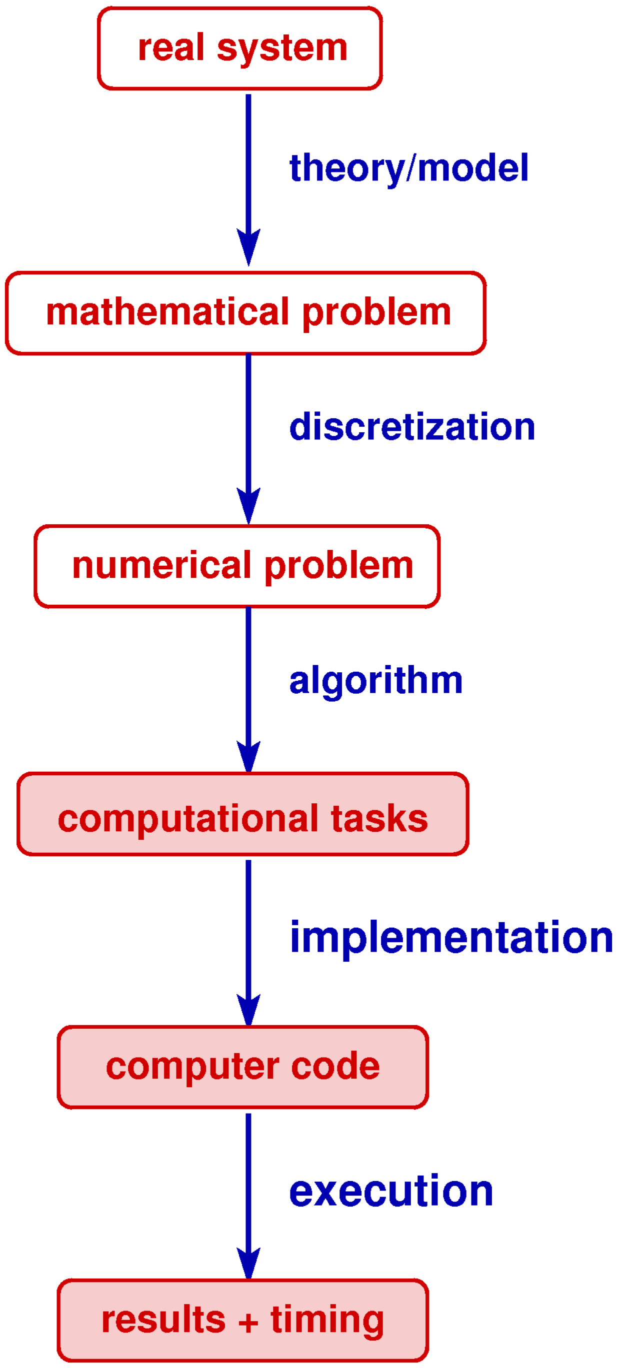
- LQCD-Optimized Architectures
  with custom designed network and/or processors
  APE, QCDOC, QPACE, . . .

- PC Clusters

- GPUs

- . . .

(provided e.g. through DESY / NIC, the Gauss Center, HLRN, or PRACE,. . .)


real system
theory/model
mathematical problem
discretization
numerical problem
algorithm
computational tasks
implementation
computer code
execution
results + timing

Advanced Analysis

Performance is a "convolution" of

application signatures $\otimes$ **hardware characteristics**

Example:

	linear algebra $\phi' = a \cdot \phi_1 + \phi_2$	Dirac operator $\phi' = M\phi$
FP operations / site	8×12	1320
Memory access (cword / site)	3×12	$\leq$ 180
ratio	2.6	$\geq$ 7.3
communications	no	yes
data re-use	no	yes

➔ Different application kernels usually depend on different hardware characteristics!

Information Exchange

$I_{xy}(N,\sigma) \equiv$	data exchange for specific computational task of size N between computer sub-systems x and y with storage σ

where x, y = registers (R), memory (M), cache (C), processors (P, P'), . . .

More explicit: compute for several specific implementations, i, separately

- ${I^i}_{xy}(N)$ information exchange
- ${S^i}_x(N)$ storage requirement

Then

$$I_{xy}(N,\sigma) = \min_{\{i: S^i_x(N) \leq \sigma\}} I^i_{xy}(N)$$

N.B.: A typical optimisation tradeoff:

storage requirement $\leftrightarrow$ information exchange

e.g.

- S_C vs. I_{CM} (cache misses)
- S_M vs. $I_{PP'}$ (communication overhead)

Hardware Model

Devices for:

- control
- data storage
- data transport/processing

Parametrized by:

ISA, . . .

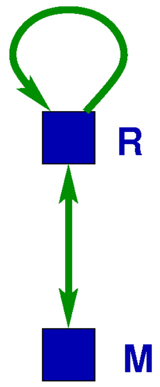
size: $0 \leq \sigma_i < \infty$

bandwidth: $\beta_{ij} < \infty$
latency: $\lambda_{ij} \geq 0$

Structure:

described by a "Hardware Architecture Graph" (HAG) with

- nodes = storage devices
- arcs = transport devices



Application Analysis

Computational Tasks:

- data set (input, output, temporary variables)
- data transport/processing tasks (equations)

Quantified by:

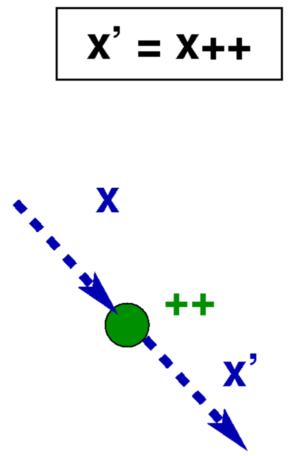
storage requirement: S_i

information exchange: I_{ij}

Data Dependencies:

described by a Directed Acyclic Graph (DAG) with

- arcs = RAW dependencies (variable lives)
- nodes = transport operations



Implementation

Main problems:

- **Code Selection**

  transport operations (DAG) $\longrightarrow$ HW instructions (DAG')

- **Resource Allocation**

  data set (variables) = arcs of DAG' $\longrightarrow$ storage devices = nodes of HAG

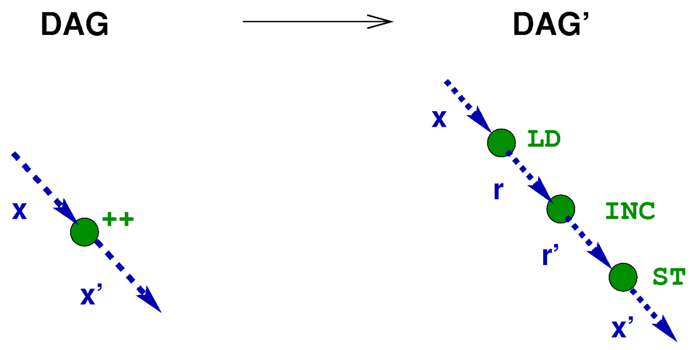
  operations (instructions) = nodes of DAG' $\longrightarrow$ transport devices = arcs of HAG

- **Scheduling**

☛ Allocation and scheduling are interrelated and NP-hard problems
(to be tackled by algorithm, programmer, compiler, hardware)

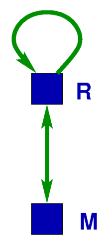
Example: $\boxed{x' = x + 1;}$

❒ Code selection

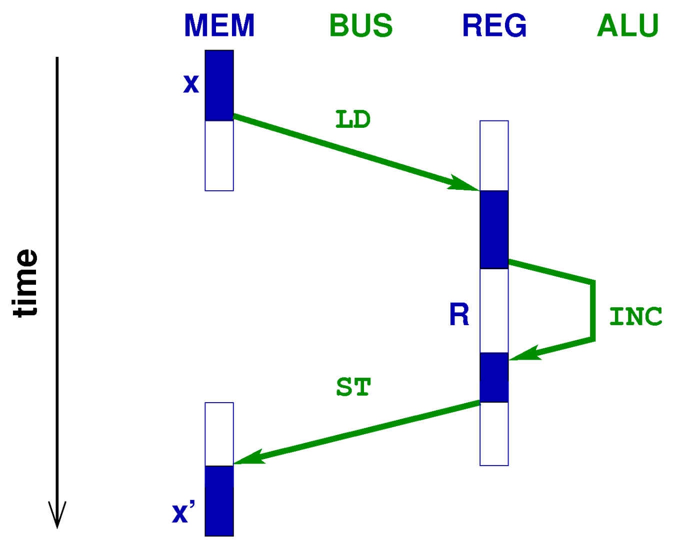


❒ Allocation of resources

DAG'		HAG
(x, r, r', x')	$\longrightarrow$	(M, R, R, M)
(LD, ST; INC)	$\longrightarrow$	(BUS; ALU)



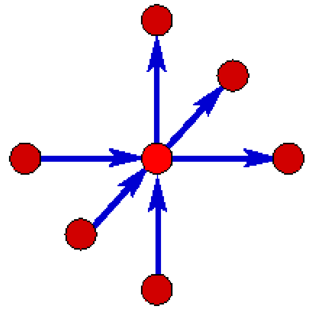
❒ Scheduling



■ positive storage life times

➛ transport in forward time

Analysis of the Wilson-Dirac Operator

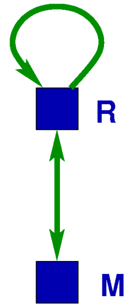


Hopping term (without even-odd preconditioning):

$$\phi' = \sum_{\mu=1}^{4} \{U(x,\mu)(1-\gamma_\mu)\phi(x+\hat{\mu}) + \cdots\}$$

$\Rightarrow$ every U link used twice and every ϕ field used 8 times

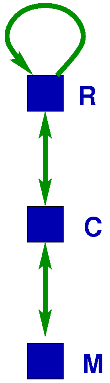
Implementation without cache



$$\begin{aligned} S_C &= 0 \\ I_{RM}/v &= (8+1)|\phi| + 8|U| = 180\ \mathit{cword} \end{aligned}$$

(v = number of lattice sites, $|\phi|$ = size of ϕ field per site, $|U|$ = size of U link)

Maximal cached implementation

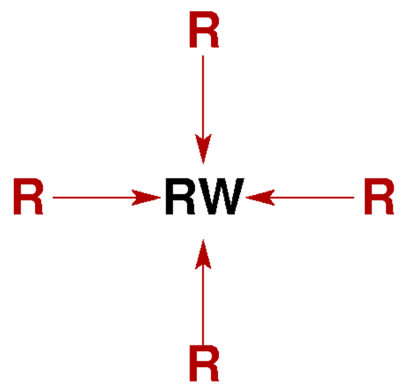


$$\begin{aligned} I_{RC}/v &= (8+1)|\phi| + 8|U| &&= 180\ \mathit{cword} \\ S_C/v &= 1|\phi| + 4|U| &&= 48\ \mathit{cword} \\ I_{CM}/v &= 2|\phi| + 4|U| &&= 60\ \mathit{cword} \end{aligned}$$

Scheduling Strategies for the Dirac Operator

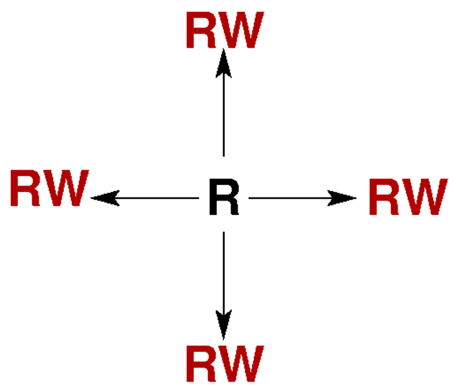
(1) Fixed ϕ':

foreach $x \in X$: compute $\phi'_x = [D\phi]_x$



(2) Fixed ϕ:

foreach $x \in X$: $\phi'_x = 0$
foreach $x \in X$:
  foreach μ:
    accumulate in $\phi'_{x\pm\hat{\mu}}$ contribution of ϕ_x



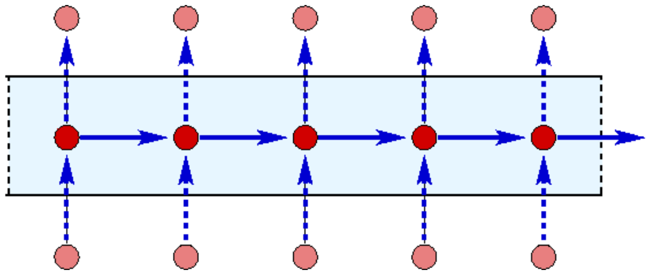
(3) Fixed U:

foreach μ:
  foreach $x \in X_\mu$:
    accumulate in $\phi'_{x\pm\hat{\mu}}$ contributions $U_{x,\mu}$

N.B.: The order for running through the sites $x \in X$ is free!

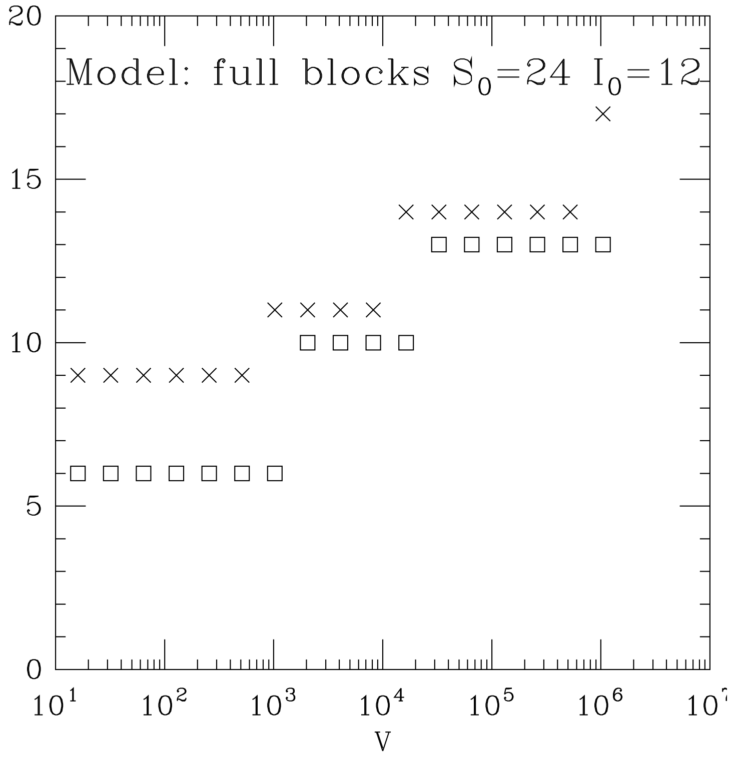
Data-Reuse in the Dirac Operator

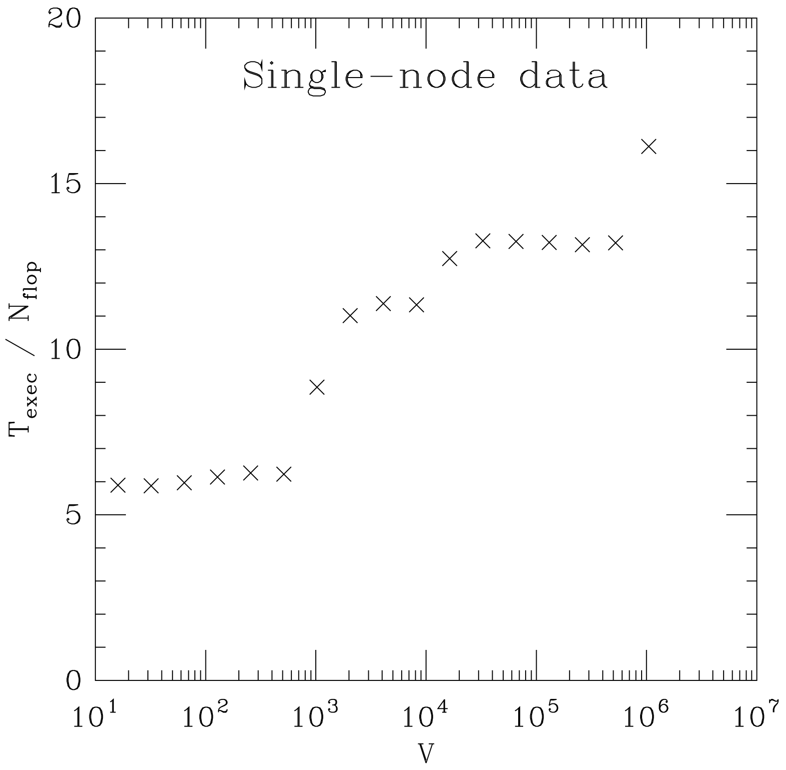
Implicit Caching of “disjoint blocks”



Model for information exchange between cache and memory

Measured execution time of CHROMA code on single Opteron node) [J. Grieger]

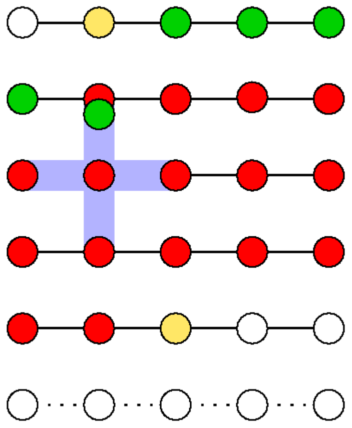




Data-Reuse in the Dirac Operator (cont.)

Explicit Caching of “moving 3d-slice”:

Sweeping along 0-direction through 3-d slices of $L_1 \times L_2 \times L_3$ sites



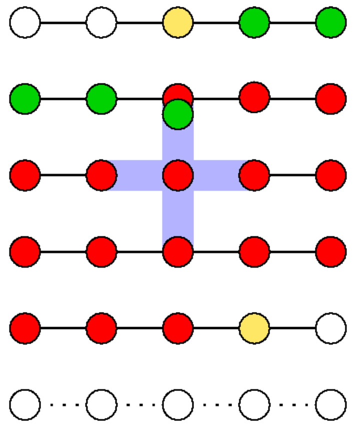
- load operands for one new slice
- computation with operands from 3 slices: $\phi' = D[U] \cdot \phi$
- store results of completed slice

Data-Reuse in the Dirac Operator (cont.)

Explicit Caching of "moving 3d-slice":

Sweeping along 0-direction through 3-d slices of $L_1 \times L_2 \times L_3$ sites

- load operands for one new slice
- computation with operands from 3 slices: $\phi' = D[U] \cdot \phi$
- store results of completed slice

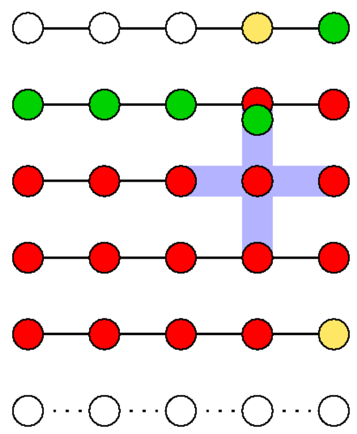


Data-Reuse in the Dirac Operator (cont.)

Explicit Caching of "moving 3d-slice":

Sweeping along 0-direction through 3-d slices of $L_1 \times L_2 \times L_3$ sites

- load operands for one new slice
- computation with operands from 3 slices: $\phi' = D[U] \cdot \phi$
- store results of completed slice

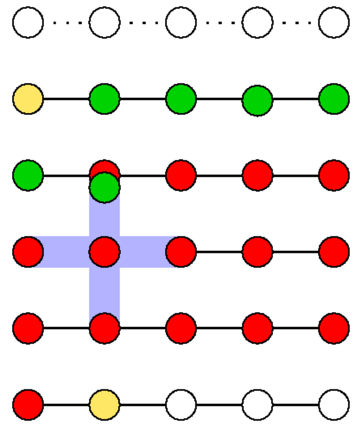


Data-Reuse in the Dirac Operator (cont.)

Explicit Caching of "moving 3d-slice":

Sweeping along 0-direction through 3-d slices of $L_1 \times L_2 \times L_3$ sites

- load operands for one new slice
- computation with operands from 3 slices: $\phi' = D[U] \cdot \phi$
- store results of completed slice



yields optimal data re-use with limited cache requirement

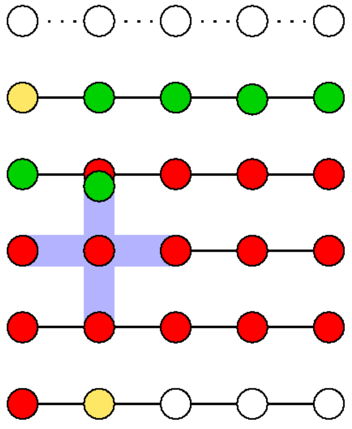
$$
\begin{aligned}
S_C &= \frac{V}{L_0} \cdot (3|\phi| + 4|U|) \\
I_{CM} &= V \cdot (2|\phi| + 4|U|)
\end{aligned}
$$

Data-Reuse in the Dirac Operator (cont.)

Explicit Caching of "moving 3d-slice":

Sweeping along 0-direction through 3-d slices of $L_1 \times L_2 \times L_3$ sites

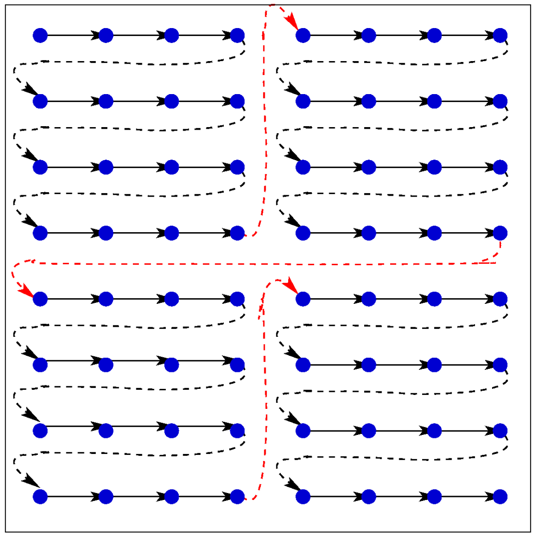
- load operands for one new slice
- computation with operands from 3 slices: $\phi' = D[U] \cdot \phi$
- store results of completed slice



yields optimal data re-use with limited cache requirement

$$
\begin{aligned}
S_C &= \frac{V}{L_0} \cdot (3|\phi| + 4|U|) \\
I_{CM} &= V \cdot (2|\phi| + 4|U|)
\end{aligned}
$$

Also other explicit blocking methods can yield very high re-use



Parallelisation of LQCD

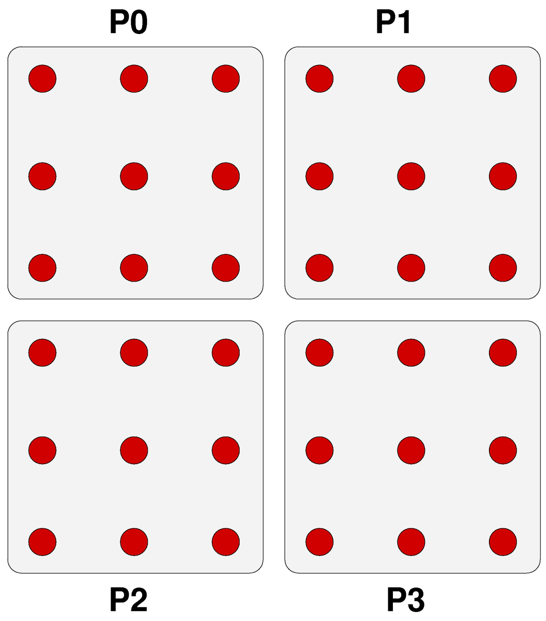
Exploit trivial data parallelism

Processor grid: $P_0 \times P_1 \times P_2 \times P_3 = P$

Local lattice: $L_0 \times L_1 \times L_2 \times L_3 = V/P$

➜ Simple geometric data decomposition:

- uniform
- static

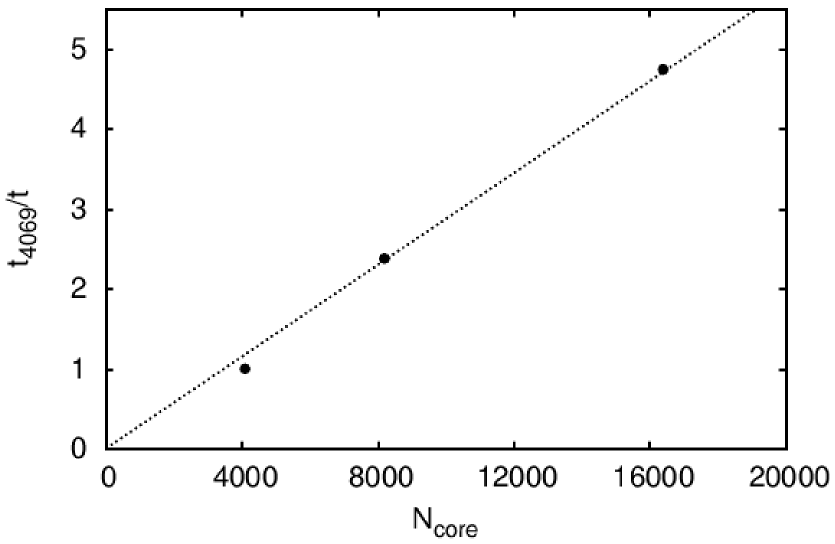


➜ Communications:

- mainly nearest neighbour
- information exchange is proportional to number of remote neighbour sites per node

$$A = 2\frac{V}{P}\sum_i \frac{1}{L_i} \qquad (P_i > 1)$$

➜ Strong scaling up to thousands of processes



Optimisations at Algorithm Level

Iterative solvers:

Combine diffrent point-operations while data in registers/cache

Example: Update of vectors (spinor fields) in CG iteration

- $s \leftarrow r + \beta \cdot s$
- $q \leftarrow A \cdot s$
- global (s, q)
- set $\alpha \leftarrow \frac{(r,r)}{(s,q)}$

➜

- compute locally

$$\begin{array}{lcl} s & \leftarrow & r + \beta \cdot s \\ q & \leftarrow & A \cdot s \\ (s,q)_{loc} & & \end{array}$$

- compute global (s, q) and set

$$\alpha \leftarrow \frac{(r,r)}{(s,q)}$$

$I_{CM}/v = 7|\phi|$ ➜ $I_{CM}/v = 4|\phi|$

Schwarz Alternating Procedure:

- natural decomposition into cache-friendly blocks
- reduced data dependencies between blocks (Dirichlet BC)
- reduced information exchange I_{CM} and $I_{PP'}$

Summary

- ❑ LQCD has huge but relatively simple computing requirements
  - many FP operations per memory access
  - regular control flow and data access patterns (memory, communications)

- ❑ Theoretical methods have been refined and tested to
  - analyse interplay between application and hardware
  - evaluate new architectures
  - guide algorithmic choices and implementation strategies