The QMP Message Passing API on QCDOC

Chulwoo Jung

(Brookhaven National Laboratory/Columbia University/SciDAC)

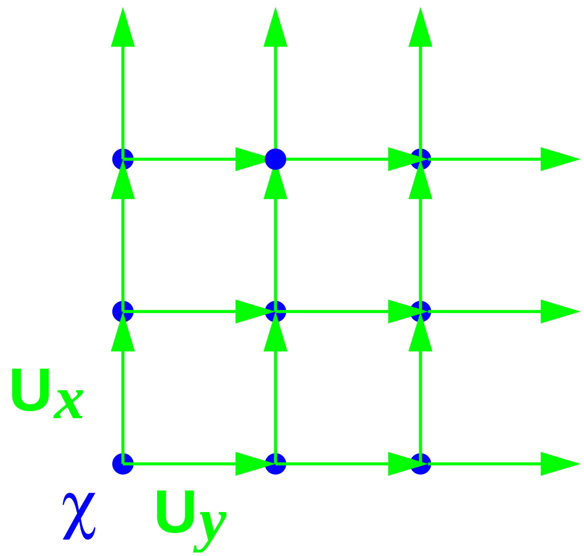
For QCDOC collaboration

Mar 26, 2003

- Overview of QCD-API
- QMP on QCDOC
- Non-nearest neighbor communication on QCDOC
- Optimization of existing programs for QMP

Brief description of lattice QCD

Euclidean path integral is evaluated numerically on the hypercubic lattice.



$$(\chi(x), A_\mu(x)) \rightarrow$$
$$(\chi(n), U_\mu(n) = \exp(-iA_\mu))$$
$$Z = \int [dU][d\bar{\chi}][d\chi] \ \exp(-(S_g + S_f))$$
$$= \int [dU][d\bar{\chi}][d\chi] \det(\not{D} + m) \exp(-(S_g))$$

In most calculations, repeated application of Dirac matrix is the most time consuming part.

Discretized Dirac matrix is a sparse matrix of rank $10^7 \sim 10^8$ in typical lattice QCD calculation.

SciDAC Software Program

Software objective: Create a unified programming environment that will enable US lattice community to achieve high efficiency on diverse multi-teraflop scale hardware.

Existing codes for lattice QCD (CPS, MILC, SZIN, ...) can be run on various hardwares by adopting to programming environment provided by SciDAC Software effort.

Participant list

Arizona	Doug Toussaint	Illinois	Celso Mendez
Arizona	Eric Gregory	Illinois	Daniel Reed
Boston U.	Richard Brower	Jlab	Robert Edwards
Boston U.	Hartmut Neff	Jlab	Chip Watson
BNL	Chulwoo Jung	Jlab	Walt Akers
BNL	Konstantin Petrov	Jlab	Jie Chen
Columbia	Bob Mawhinney	MIT	Andrew Pochinsky
Fermilab	Don Holmgren	Utah	Carleton DeTar
Fermilab	Jim Simone	Utah	James Osborn
Fermilab	Amitoj Singh		

SciDAC related activities in the QCDOC collaboration

- Level 3: Highly optimized routines for Lattice QCD
  - Dirac matrix inverter
    - Wilson, Clover (P. Boyle)
    - Staggered (C. Cristian)
    - Asqtad (Improved staggered) (CJ, H. Lin)
  - HMC Asqtad force term (CJ, C. McNeile, Z. Sroczynski)
- Level 2: QCD Data Parallel (QDP): Lattice-wide operations with both communication and computation (K. Petrov).
  example: parallel transport $\chi'(x) = U_\mu(x)\chi(x + \hat{\mu})$
- Level 1:
  - QMP : Internode communication (CJ)
  - QLA: Single node linear Algebra

Brief description of QMP

The goal of QMP is to provide portable, low-latency, high-bandwidth communication routines suitable for Lattice QCD.

- Point-to-point communication
  - Nonblocking
  - Simultaneous, multidirectional transfer capability
  - Chained block/strided transfer capability
  - Separate routines for the initialization and commencement of transfers $\longrightarrow$ opened communication channels can be reused, minimize overhead for repeated transfers.
- Global operation
  - Global reduction
    {Sum, Max, Min} {Single precision, Double precision} {Scalar, Vector}
    General binary reduction available
  - Broadcast
  - Barrier

Implementation of QMP on QCDOC

- Point-to-point communications
  For nearest neighbor, the requirements for QMP are well satisfied by QC-DOC physics network $\rightarrow$ Small software overhead.
  Both C/C++ Binding implemented over QCDOC system calls.
- Global operations
  Store-and-forward capability of QCDOC is used.
  Both C/C++ Binding implemented over QCDOC system calls.
  Communication time grows as $\sim L \times N_d$
  L = Number of sites N_d = Number of Dimension.
- Implementation complete except non-nearest communication.

Non-Nearest Neighbor Communication(NNC) on QCDOC

While repeated communications are often to/from nearest-neighbor, NNC gives more flexibility and portability.

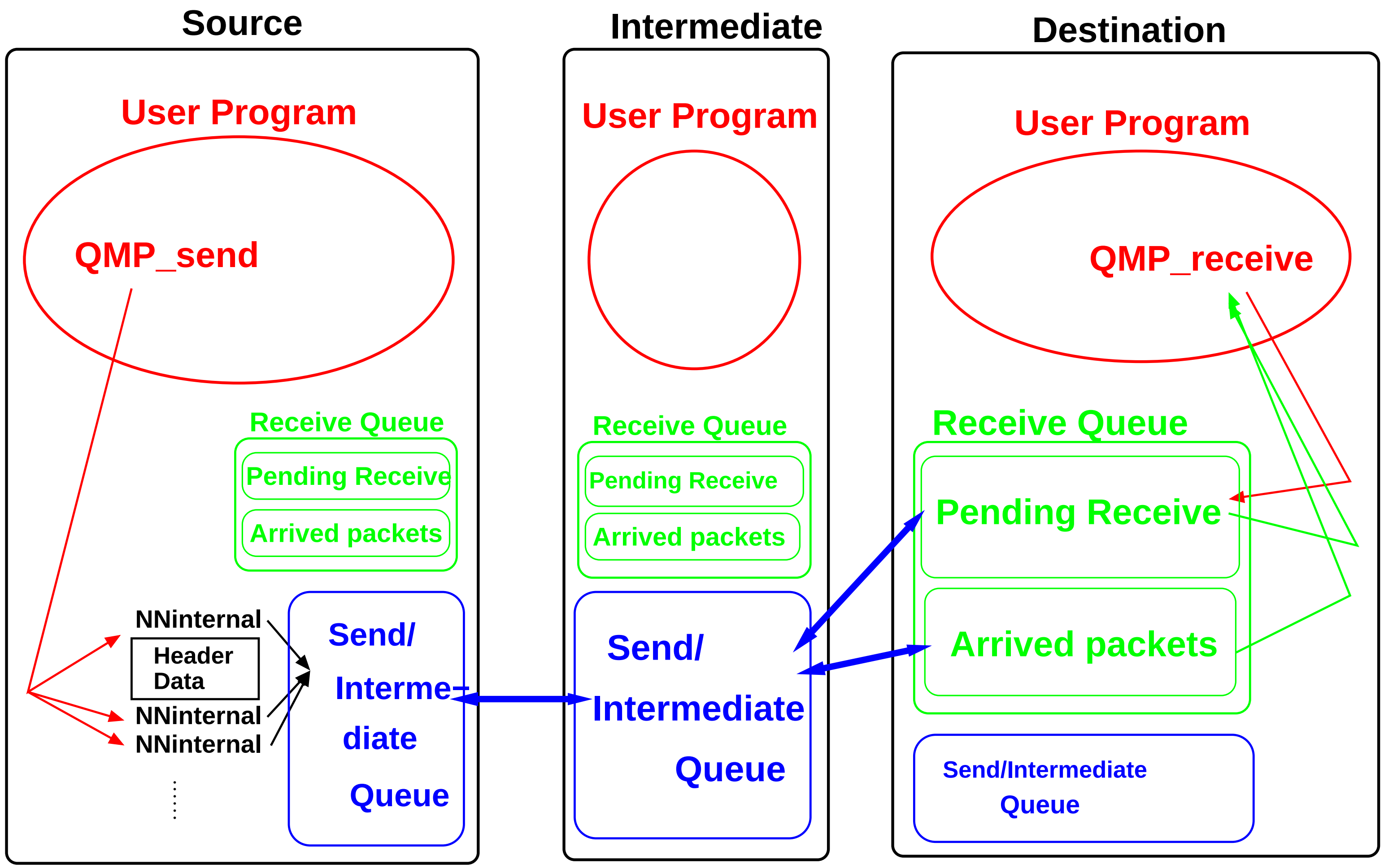
Desired features:

- Minimum number of copying
- Minimal overhead when destination is nearest
- Should be able to co-exist with nearest neighbor communication

Implementation strategy:

- Interrupting communication channel (Supervisor) used for the header transfer
- Non-interrupting physics network used for user data transfer
- Static, Manhattan-style routing to prevent lock-ups/long delays

Diagram of NNC



Performance of MILC + QMP on QCDOC

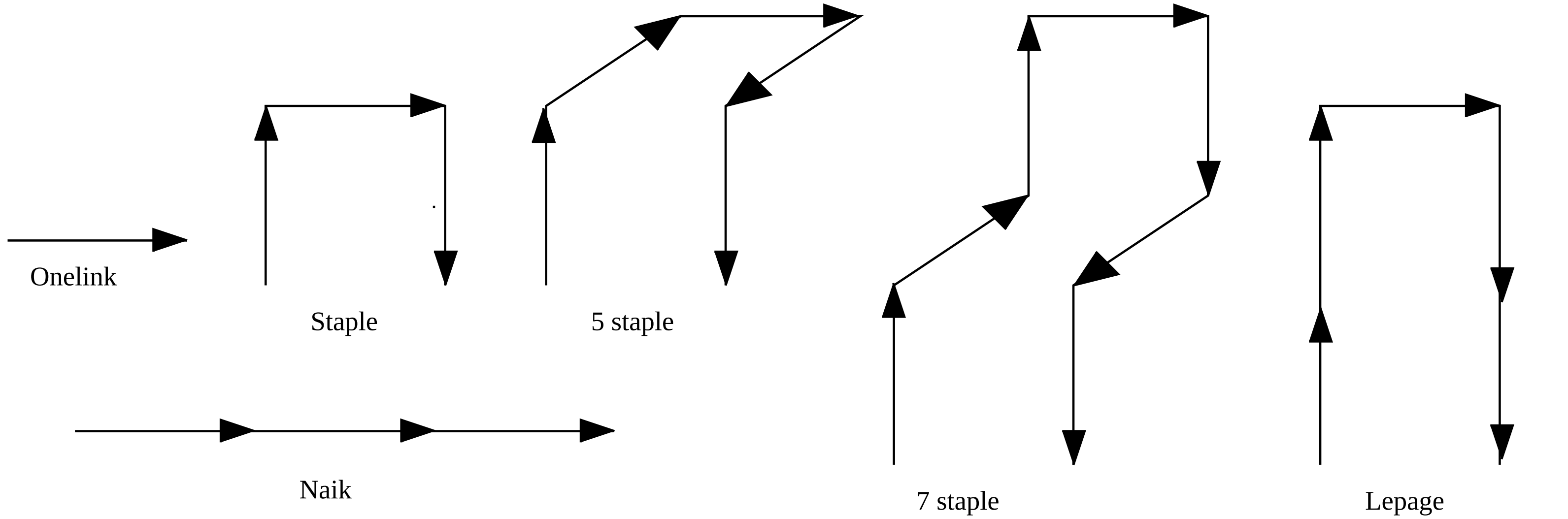
MILC code: C codes for SU(3) Gauge theory, developed and maintained my MILC collaboration. Ported to various hardware platforms and communication protocols, including QMP and MPI.

For single precision, $15 \sim 20\%$ of peak performance are observed from QCDOC simulator in many computationally intensive MILC routines (XLC compiler).

	Sites/ node	Machine optimized (double precision)		Sites/ node	Optimized (double precision)	MILC (single precision)
Wilson Dirac	2^4 4^4	47% 54%	Staggered Dirac	2^4 4^4	36%	17% 21%
Clover Dirac	2^4 4^4	56% 59%	Asqtad Dirac	4^4	33%	15%
			Asqtad force	2^4 4^4		$(3 \rightarrow 14)\%$ $(6 \rightarrow 20)\%$

Improved action(Asqtad) force term

Asqtad action is one of the improved staggered fermion action, with smaller lattice spacing error and flavor mixing.



Calculation of the force term mostly consists of parallel transport and SU(3) vector outer products.

Parallel transport: $\chi'(x) = U_\mu(x)\chi(x+\hat{\mu})$

Vector outer product: $P(x) = \chi(x)\chi'^{\dagger}(x)$

Total computation: $\sim$ 250,000 FLOPs per site

(Dirac inverter: $\sim (1150 \times (n \sim 1000))$FLOPs per site)

Optimization of Asqtad force term

- Elimination of repetitive overhead
  - Reusing Communication channel in each parallel transport routine
    $30 \sim 60\%$ of total cycles was for creation and destruction of communication channel.
  - Overlapping communication and computation
- Inlining small functions
  Overhead for function call is on the order of 100 cycles. function(s) with less than hundreds of computations should be avoided!
  $2^4 : 3 \rightarrow 12\%$, $4^4 : 6 \rightarrow 13\%$.
- Unrolling of loops in computation routines
  Unrolling the loops enables user to load variables as early as possible, alleviating cache misses and pipeline conflicts. This improves the performance of computation routines by $\times(1.5 \sim 1.7)$.

  $2^4 : 12 \rightarrow 14\%$, $4^4 : 13 \rightarrow 20\%$.

Conclusions

- QMP provides portable, efficient communication routines for lattice QCD.
- Existing C/C++ programs can be ported for various hardware platforms and run reasonably well without machine-specific optimizations by using QMP .
- To take full advantage of hardware capabilities provided via QMP, the user programs should be written in such a way that minimizes repetitive overheads.