Writing efficient code with QDP(C): benchmarks and optimization

SciDAC All Hands Meeting

BNL

Mar. 27, 2003

James C. Osborn

Improved Staggered "Asqtad" D-slash

```
/* shift source from forward directions */
  for(mu=0; mu<4; mu++) {
    QDP_V_eq_sV(temp_fwd[mu], src, QDP_neighbor[mu], QDP_forward, subset);
    QDP_V_eq_sV(temp_fwd3[mu], src, neighbor3[mu], QDP_forward, subset);
  }

/* multiply source by adjoint links */
  for(mu=0; mu<4; mu++) {
    QDP_V_eq_Ma_times_V(tempvec[mu], fatlinks[mu], src, othersubset);
    QDP_V_eq_Ma_times_V(tempvec3[mu], longlinks[mu], src, othersubset);
  }

/* shift multiplied source from backward directions */
  for(mu=0; mu<4; mu++) {
    QDP_V_eq_sV(temp_bck[mu], tempvec[mu], QDP_neighbor[mu], QDP_backward, subset);
    QDP_V_eq_sV(temp_bck3[mu], tempvec3[mu], neighbor3[mu], QDP_backward, subset);
  }

/* multiply forward shifted source by links and accumulate */
  for(mu=0; mu<4; mu++) {
    QDP_V_peq_M_times_V(dest, fatlinks[mu], temp_fwd[mu], subset);
    QDP_V_peq_M_times_V(dest, longlinks[mu], temp_fwd3[mu], subset);
    QDP_discard_V(temp_fwd[mu]);
    QDP_discard_V(temp_fwd3[mu]);
  }

/* subtract backward fields */
  for(mu=0; mu<4; mu++) {
    QDP_V_meq_V(dest, temp_bck[mu], subset);
    QDP_V_meq_V(dest, temp_bck3[mu], subset);
    QDP_discard_V(temp_bck[mu]);
    QDP_discard_V(temp_bck3[mu]);
  }
```

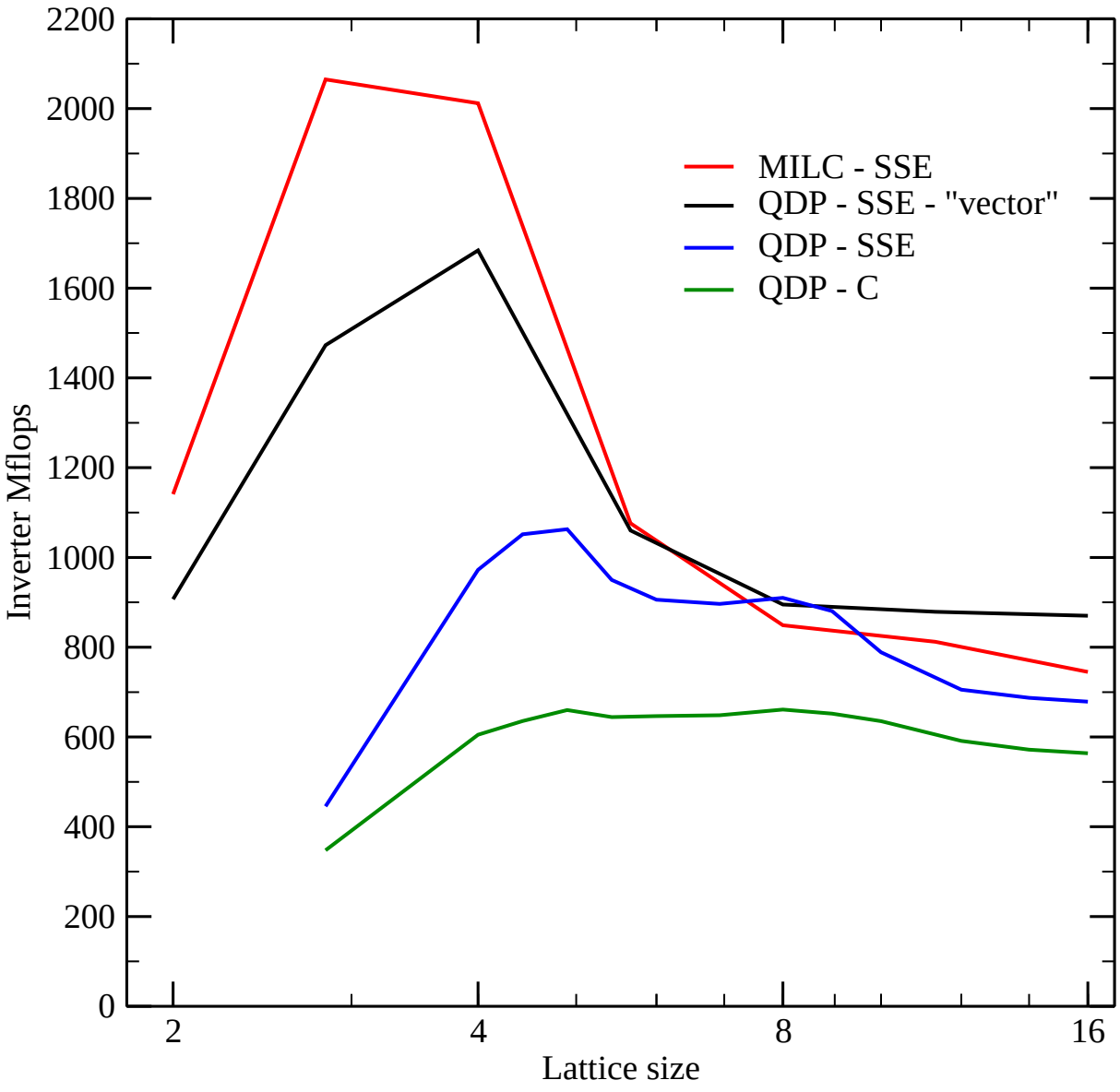
Use "vector" calls where possible

```
  for(mu=0; mu<4; mu++) srcvec[mu] = src;
  for(mu=0; mu<4; mu++) destvec[mu] = dest;

/* shift source from forward directions */
  QDP_V_veq_sV(temp_fwd, srcvec, QDP_neighbor, fwdvec, subset, 4);
  QDP_V_veq_sV(temp_fwd3, srcvec, neighbor3, fwdvec, subset, 4);

/* multiply source by adjoint links */
  QDP_V_veq_Ma_times_V(tempvec, fatlinks, srcvec, othersubset, 4);
  QDP_V_veq_Ma_times_V(tempvec3, longlinks, srcvec, othersubset, 4);

/* shift multiplied source from backward directions */
  QDP_V_veq_sV(temp_bck, tempvec, QDP_neighbor, bckvec, subset, 4);
  QDP_V_veq_sV(temp_bck3, tempvec3, neighbor3, bckvec, subset, 4);

/* multiply forward shifted source by links and accumulate */
  QDP_V_vpeq_M_times_V(destvec, fatlinks, temp_fwd, subset, 4);
  QDP_V_vpeq_M_times_V(destvec, longlinks, temp_fwd3, subset, 4);
  for(mu=0; mu<4; mu++) {
    QDP_discard_V(temp_fwd[mu]);
    QDP_discard_V(temp_fwd3[mu]);
  }

/* subtract backward fields */
  QDP_V_vmeq_V(destvec, temp_bck, subset, 4);
  QDP_V_vmeq_V(destvec, temp_bck3, subset, 4);
  for(mu=0; mu<4; mu++) {
    QDP_discard_V(temp_bck[mu]);
    QDP_discard_V(temp_bck3[mu]);
  }
```

Even more efficient

```
  for(mu=0; mu<8; mu++) srcvec[mu] = src;
  for(mu=0; mu<8; mu++) destvec[mu] = dest;

/* shift source from forward directions */
  QDP_V_veq_sV(temp_fwd, srcvec, shifts, fwdvec, subset, 8);

/* multiply source by adjoint links */
  QDP_V_veq_Ma_times_V(tempvec, links, srcvec, othersubset, 8);

/* shift multiplied source from backward directions */
  QDP_V_veq_sV(temp_bck, tempvec, shifts, bckvec, subset, 8);

/* multiply forward shifted source by links and accumulate */
  QDP_V_vpeq_M_times_V(destvec, links, temp_fwd, subset, 8);
  for(mu=0; mu<8; mu++) {
    QDP_discard_V(temp_fwd[mu]);
  }

/* subtract backward fields */
  QDP_V_vmeq_V(destvec, temp_bck, subset, 8);
  for(mu=0; mu<8; mu++) {
    QDP_discard_V(temp_bck[mu]);
  }
```

Single node Pentium 4



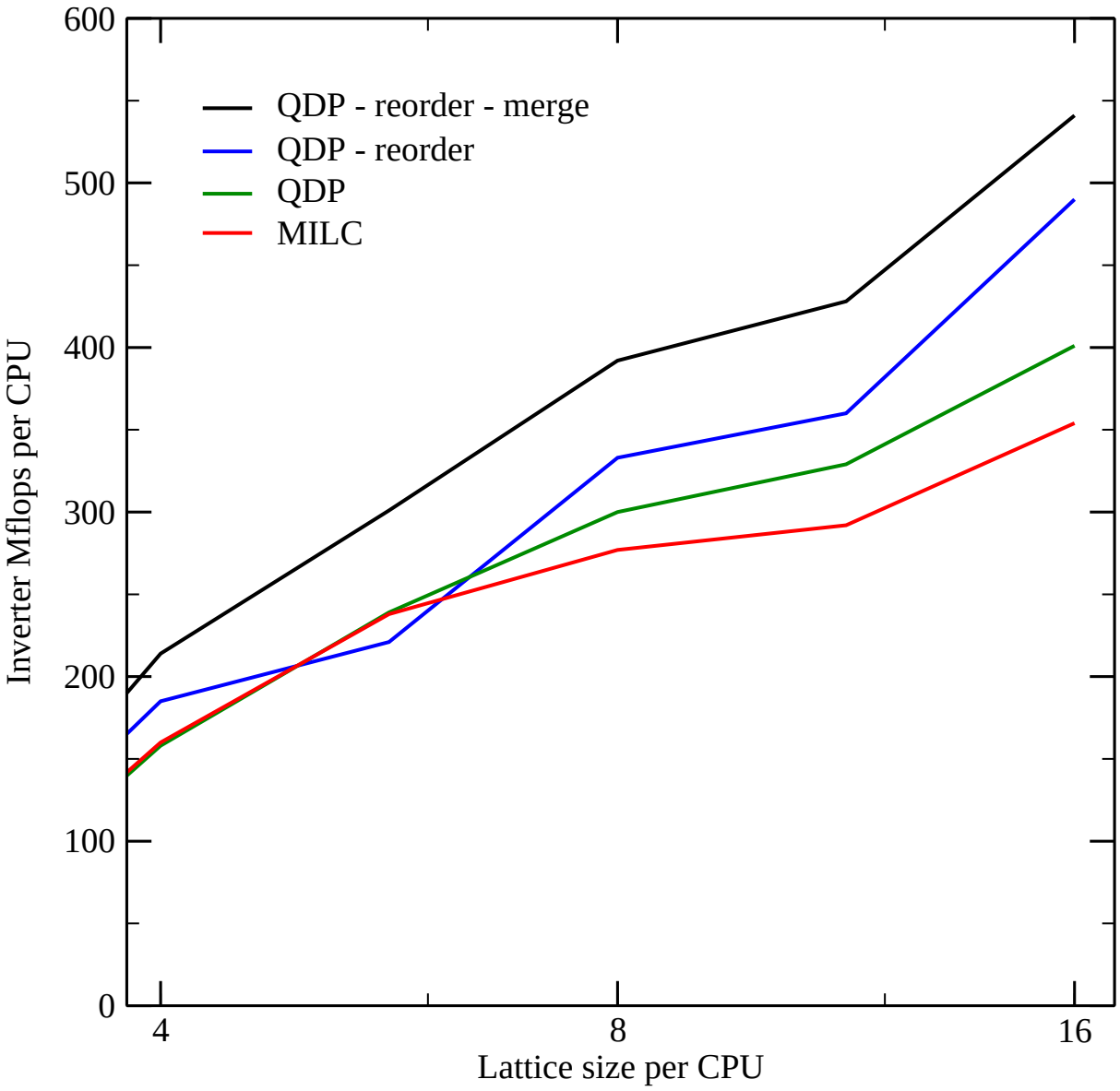
A possible reordering

```
  for(mu=0; mu<16; mu++) srcvec[mu] = src;
  for(mu=0; mu<16; mu++) destvec[mu] = dest;

/* shift source from all directions */
  QDP_V_veq_sV(temp, srcvec, shifts, shiftdirs, subset, 16);

/* multiply shifted source by links and accumulate */
  QDP_V_vpeq_M_times_V(destvec, links, temp, subset, 16);
  for(mu=0; mu<16; mu++) {
    QDP_discard_V(temp[mu]);
  }
```

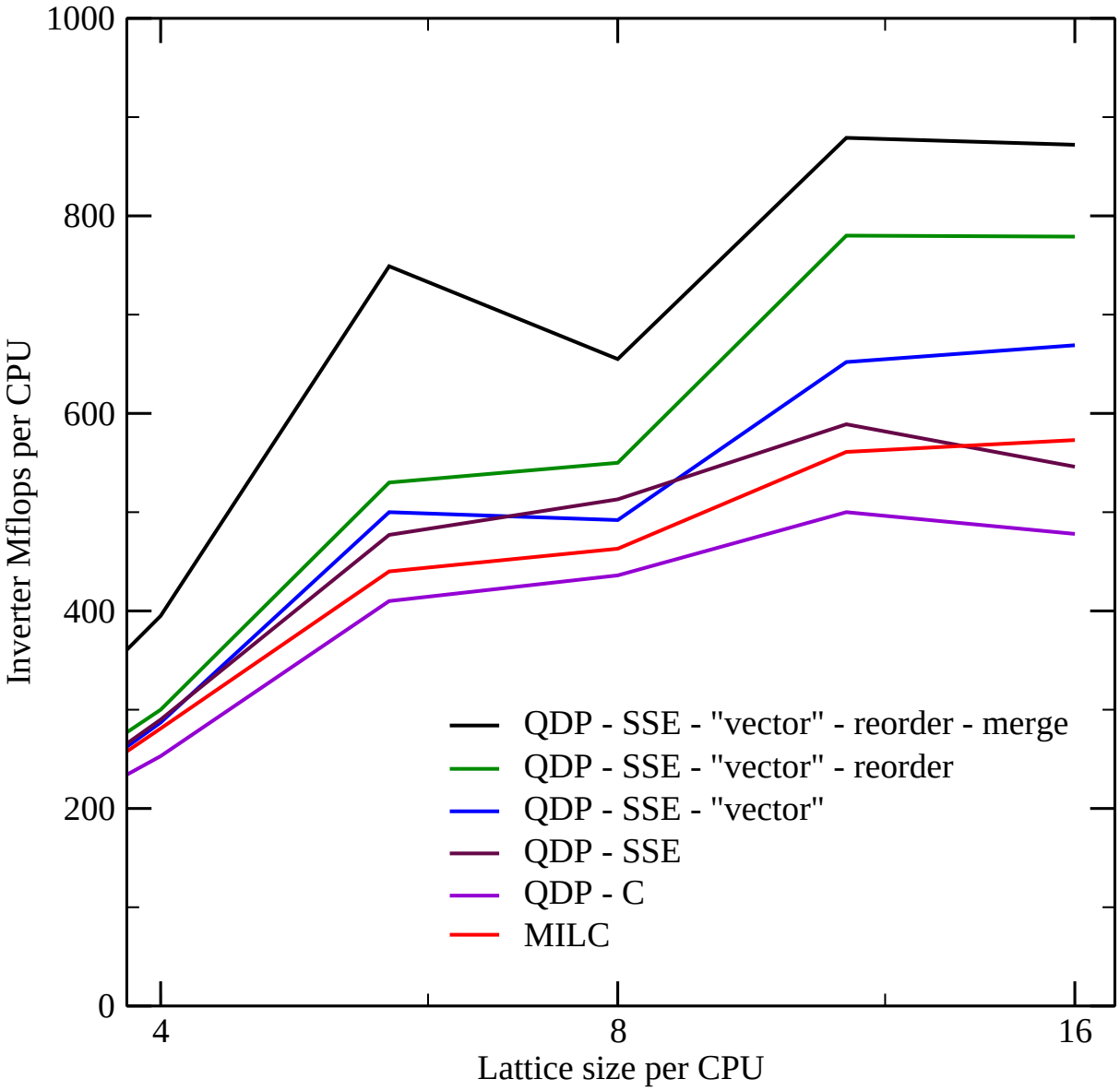
or perhaps this...

```
/* shift source from all directions */
  for(mu=0; mu<16; mu+=2) {
    QDP_V_veq_sV(&temp[mu], &srcvec[mu], &shifts[mu], &shiftdirs[mu], subset, 2);
  }

/* multiply shifted source by links and accumulate */
  QDP_V_eq_M_times_V(destvec, links, temp, subset);
  QDP_V_vpeq_M_times_V(&destvec[1], &links[1], &temp[1], subset, 15);
  for(mu=0; mu<16; mu++) {
    QDP_discard_V(temp[mu]);
  }
```

64 x 2 nodes - dual Pentium 4



16 x 1 nodes - Pentium 4



Summary

It is possible to write efficient code using SciDAC Level 2 library QDP.

1) Library optimizations (QDP, QLA, QMP, QIO)
   - Improved math libraries (SSE, etc.)
   - Lazy shifts
   - Restart shifts
   - "Vector" math operations
   - Combining shifts

2) Application code optimizations
   - Overlap math and communications
   - Discard shift results
   - Use "vector" math routines
   - Group shifts
   - Use profiling information