The 39th International Symposium on Lattice Field Theory (Lattice 2022)

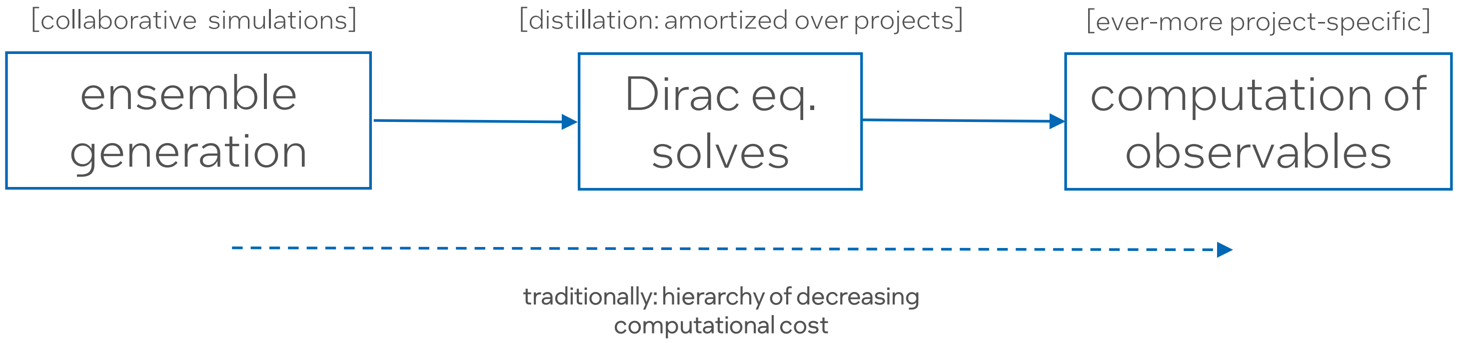
Performance Optimization of Baryon-block Construction in the Stochastic LapH Method

Phuong Nguyen* (TU Munich/Intel)
Ben Hoerz (Intel)

intel®

Overview

1. Motivation



- Traditionally, comparatively moderate cost to compute observables
- Balance can shift for modern multi-hadron spectroscopy
  - e.g. Baryon systems with the stochastic LapH method [Morningstar et al. 1104.3870]
- **Challenge**: scaling to state-of-the-art ensembles
  - Design with CLS E250 in mind: 96^3 x 192, a = 0.064 fm [talk by W. Soeldner Tue 15:40]
  - First multi-hadron results in mesonic sector [talk by S. Paul Fri 15:10]
  - Projected cost of baryon blocks exceeds Dirac solves severalfold

2. Baryon-block Construction

$$\mathcal{B}_{\mathbf{p}}^{d_1 d_2 d_3} = \sum_{\mathbf{x}} \sum_{a,b,c} \mathrm{e}^{-\mathbf{ipx}} \, \varepsilon_{abc} \, q_{\mathbf{x}a}^{d_1} q_{\mathbf{x}b}^{\prime d_2} q_{\mathbf{x}c}^{\prime\prime d_3}$$

- q : set of (LatticeColorVector-valued) LapH quark fields.
- d1, d2, d3 = 1,…,nDil
- Kernel called many times with same momentum set, different q (spin, noise combinations)
  - #momenta ≪ V ⇒ precompute momentum phases for SFT
- Seek to exploit high arithmetic intensity
  - O(nDil^3) compute with O(nDil) memory transfer.

3. Reference Implementation (1)

```
input: q1, q2, q3            // 3D array, [nD,nX,nColor]
       momBuf               // 2D array,  [nMom, nX]
output: res                  // 4D array, [nMom, nD1, nD2, nD3]
intermediate: diq            // 2D array, [nX, nColor]
              singlet        // 1D array, [nRows, nX]
for d1 = 1 to nD1; do        // dilution index 1
  for d2 = 1 to nD2; do      // dilution index 2
    for iX = 1 to nX; do
      calculate diq(iX,:) = q1(d1,iX,:) * q2(d2,iX,:)
    end for
    rowIndex = 0;
    for d3 = 1 to nD3; do  // dilution index 3
      for iX = 1 to nX; do
        calculate singlet(rowIndex, iX) = diq(iX,:) * q3(d3,iX,:)
      end for
      // blocked momentum projection
      rowIndex++;
      if rowIndex == nrows
        // With BLAS Level 3 - GEMM
        res(:, d1, d2, d3 - nrows : d3) = momBuf * singlet
        rowIndex = 0
      end if
    end for
  end for
end for
```

$$\mathcal{B}_p^{d_1 d_2 d_3} = \sum_x e^{-i\mathbf{p}\mathbf{x}} \sum_{a,b,c} \varepsilon_{abc}\, q_{xa}^{d_1} q'^{d_2}_{xb} q''^{d_3}_{xc}$$

diq * q3

momBuf singlet

- pre-compute **diq** = q1 x q2 once for all d3
- pack multiple **singlet** vectors into a matrix, then using BLAS-3 to calculate **momBuf** * **singlet**.

Problems ???

3. Reference Implementation (2)

Testing system

Hardware Info:

- 2 x Intel(R) Xeon(R) Platinum 8358 CPU @ 2.60GHz.
- 64 cores in total.

Theoretical Peak Performance:

- Single core: 2.6 GHz * 2 FMA * 2 Flops/FMA * 8 Flops/ AVX512 DP = **83.2 GFLOPs**
- Node performance: 64 x 83.2 GFLOPs = **5324 GFLOPs**

3. Reference Implementation (3)

Profiling for single core

- Only 33% of Peak Performance
- ~27% Non- FP

```
$ vtune -collect hpc-performance ./kernel
…
CPU
  DP GFLOPS: 27.490

Vectorization: 99.9% of Packed FP Operations
    Instruction Mix
        DP FLOPs: 73.5% of uOps
            Packed: 99.9% from DP FP
                128-bit: 26.1% from DP FP
                256-bit: 0.0% from DP FP
                512-bit: 73.9% from DP FP
            Scalar: 0.1% from DP FP
        x87 FLOPs: 0.0% of uOps
        Non-FP: 26.5% of uOps
    FP Arith/Mem Rd Instr. Ratio: 2.342
    FP Arith/Mem Wr Instr. Ratio: 12.229
…
```

Low Arithmetic Intensity

```
$ vtune –collect memory-access ./kernel
…
CPU Time: 10.515s
    Memory Bound: 37.2% of Pipeline Slots
        L1 Bound: 1.1% of Clockticks
        L2 Bound: 10.3% of Clockticks
        L3 Bound: 3.5% of Clockticks
        DRAM Bound: 15.0% of Clockticks
Elapsed Time
        Store Bound: 1.5% of Clockticks
Time
    Loads: 19,793,093,775
    Stores: 4,173,125,190
    LLC Miss Count: 97,540,950
        Local DRAM Access Count: 39,002,730
        Remote DRAM Access Count: 45,503,185
        Remote Cache Access Count: 0
    Average Latency (cycles): 41
…
```

- DRAM bound & large number of stalled cycles for LLC Misses

4. Single-core Optimization (1)

Reference Implementation

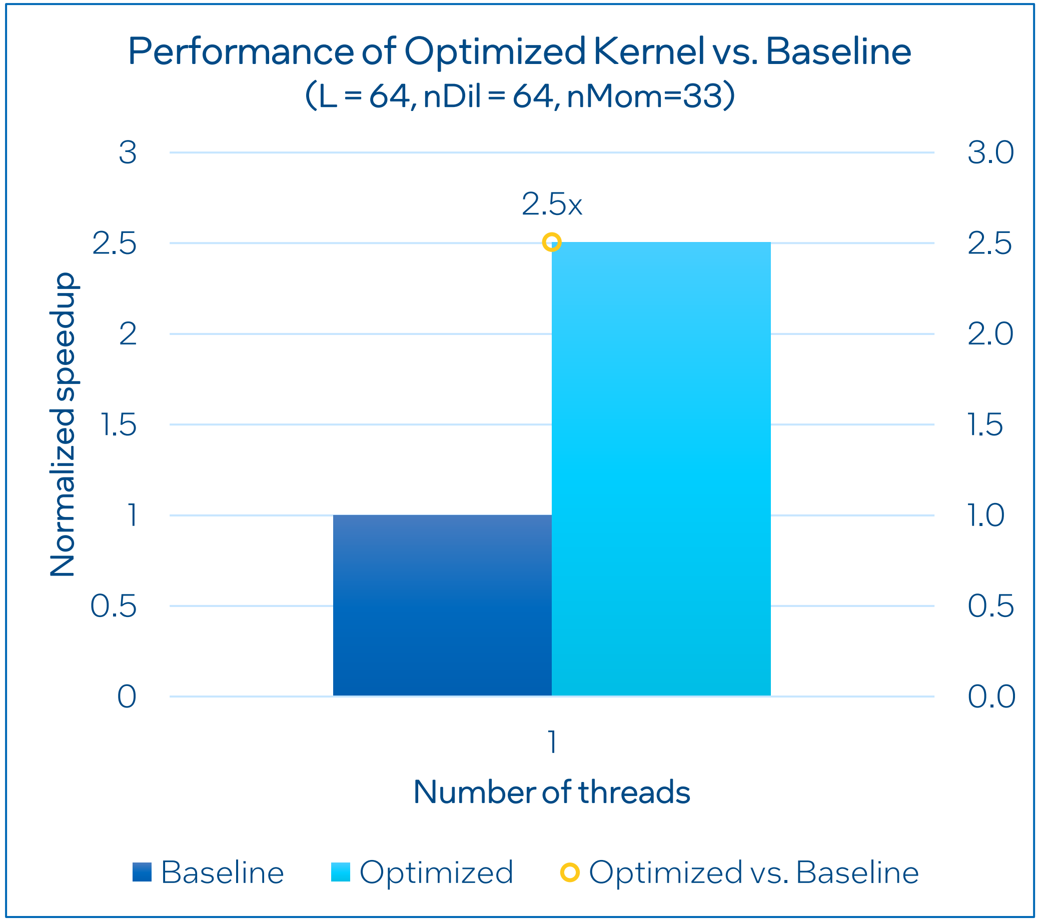
```
input: q1, q2, q3          // 3D array, [nD,nX,nColor]
       momBuf              // 2D array,  [nMom, nX]
output: res                  // 4D array, [nMom, nD1, nD2, nD3]
intermediate: diq          // 2D array, [nX, nColor]
                 singlet     // 1D array, [nRows, nX]
for d1 = 1 to nD1; do      // dilution index 1
  for d2 = 1 to nD2; do    // dilution index 2
    for iX = 1 to nX; do
      calculate diq(iX,:) = q1(d1,iX,:) * q2(d2,iX,:)
    end for
    rowIndex = 0;
    for d3 = 1 to nD3; do   // dilution index 3
      for iX = 1 to nX; do
        calculate singlet(rowIndex, iX) = diq(iX,:) * q3(d3,iX,:)
      end for
      // blocked momentum projection
      rowIndex++;
      if rowIndex == nrows
         // With BLAS Level 3 - GEMM
         res(:, d1, d2, d3 - nrows : d3) = momBuf * singlet
         rowIndex = 0
      end if
    end for
  end for
end for
```

⇨

```
input: q1, q2, q3      // 3D array, [nD,nColor,nX]
       momBuf          // 2D array, [nMom, nX]
output: res            // 4D array, [nMom, nD1, nD2, nD3]
intermediate: diq      // 4D array, [bsizeD1, bsizeD2, nColor, bsizeX]
              singlet  // 4D array, [bsizeD3, bsizeD1, bsizeD2, bsizeX]
              tmpBuf   // 4D array, [nD3, bsizeD1, bsizeD2,nMom]
for blockD1; do
  for blockD2; do
    init(tmpBuf, 0.)
    for blockX; do
      for d1 = 1 to bsizeD1; do
        for d2 = 1 to bsizeD2; do
          for iX = 1 to bsizeX; do
            calculate diq(:) = q1(~d1,:,iX) * q2(~d2,:,iX)
          end for
        end for
      end for
      for blockD3; do
        for d3 = 1 to bsizeD3; do
          for diq_elem in diq; do
            for iX = 1 to bsizeX; do
              calculate singlet(:) = diq_elem * q3(~d3,:,iX)
            end for
          end for
        end for
       // With JIT GEMM
       tmpBuf += singlet * momBuf
     end for // blockX
     storing res <- tmpBuf
  end for // blockD2
end for // blockD1
```

- Change data layout to have contiguous memory accesses.
- Cache Blocking in d1, d2, d3, nX.
- Intel MKL Just-in-Time (JIT) Code Generation for Small Matrix Multiplication

4. Single-core Optimization (2)



4. Single-core Optimization (3)

Profiling with a small test case (L = 32, nDil = 32)

Reference

```
$ vtune -collect hpc-performance ./kernel
…
CPU
     DP GFLOPS: 27.490

Vectorization: 99.9% of Packed FP Operations
    Instruction Mix
        DP FLOPs: 73.5% of uOps
            Packed: 99.9% from DP FP
                128-bit: 26.1% from DP FP
                256-bit: 0.0% from DP FP
                512-bit: 73.9% from DP FP
            Scalar: 0.1% from DP FP
        x87 FLOPs: 0.0% of uOps
        Non-FP: 26.5% of uOps
    FP Arith/Mem Rd Instr. Ratio: 2.342
    FP Arith/Mem Wr Instr. Ratio: 12.229
…
```

⇨

After Optimization

```
$ vtune -collect hpc-performance ./kernel
…
CPU
        DP GFLOPS: 46.463

Vectorization: 99.9% of Packed FP Operations
    Instruction Mix
        DP FLOPs: 96.6% of uOps
            Packed: 99.9% from DP FP
                128-bit: 11.5% from DP FP
                256-bit: 0.0% from DP FP
                512-bit: 88.5% from DP FP
            Scalar: 0.0% from DP FP
        x87 FLOPs: 0.0% of uOps
        Non-FP: 3.4% of uOps
    FP Arith/Mem Rd Instr. Ratio: 3.583
    FP Arith/Mem Wr Instr. Ratio: 34.985
…
```

- 1,7x DP GFLOPS
  ~ 56% Peak Performance
- Only 3% Non- FP
- 3x Arithmetic Intensity

4. Single-core Optimization (4)

Profiling for memory-access

Reference

```
$ vtune -collect memory-access ./kernel
…
CPU Time: 10.515s
    Memory Bound: 37.2% of Pipeline Slots
        L1 Bound: 1.1% of Clockticks
        L2 Bound: 10.3% of Clockticks
        L3 Bound: 3.5% of Clockticks
        DRAM Bound: 15.0% of Clockticks
Elapsed Time
        Store Bound: 1.5% of Clockticks
Time
    Loads: 19,793,093,775
    Stores: 4,173,125,190
    LLC Miss Count: 97,540,950
        Local DRAM Access Count: 39,002,730
        Remote DRAM Access Count: 45,503,185
        Remote Cache Access Count: 0
    Average Latency (cycles): 41
…
```

⇨

After Optimization

```
$ vtune -collect memory-access ./kernel
…
CPU Time: 6.580s
    Memory Bound: 36.9% of Pipeline Slots
        L1 Bound: 5.7% of Clockticks
        L2 Bound: 2.4% of Clockticks
        L3 Bound: 14.7% of Clockticks
        DRAM Bound: 4.8% of Clockticks
Elapsed Time
        Store Bound: 0.0% of Clockticks
Time
    Loads: 12,038,361,140
    Stores: 1,235,037,050
    LLC Miss Count: 0
        Local DRAM Access Count: 0
        Remote DRAM Access Count: 0
        Remote Cache Access Count: 0
    Average Latency (cycles): 42
…
```

- L3 Bound instead of DRAM Bound

- 40% Less cycles for Loads
- 70% Less cycles for Stores

- 0 stalled cycles for LLC Misses

5. Node-level Optimization (1)

```
input: q1, q2, q3            // 3D array, [nD,nColor,nX]
       momBuf               // 2D array, [nMom, nX]
output: res                  // 4D array, [nMom, nD1, nD2, nD3]
intermediate: diq            // 4D array, [bsizeD1, bsizeD2, nColor, bsizeX]
              singlet        // 4D array, [bsizeD3, bsizeD2, bsizeD1, bsizeX]
              tmpBuf         // 2D array, [nD3, bsizeD2*bsizeD1,nMom]
for blockD1; do
  for blockD2; do
    init(tmpBuf, 0.)
    for blockX; do
      for d1 = 1 to bsizeD1; do
        for d2 = 1 to bsizeD2; do
          for iX = 1 to bsizeX; do
            calculate diq(:) = q1(~d1,:,iX) * q2(~d2,:,iX)
          end for
        end for
      end for
      for blockD3; do
        for d3 = 1 to bsizeD3; do
          for diq_elem in diq; do
            for iX = 1 to bsizeX; do
              calculate singlet(:) = diq_elem * q3(~d3,:,iX)
            end for
           end for
         end for
       tmpBuf += singlet * momBuf         // Using MKL JIT
      end for // blockX
      storing res <- tmpBuf
  end for // blockD2
end for // blockD1
```

- Which loop to parallelize?

5. Node-level Optimization (2)

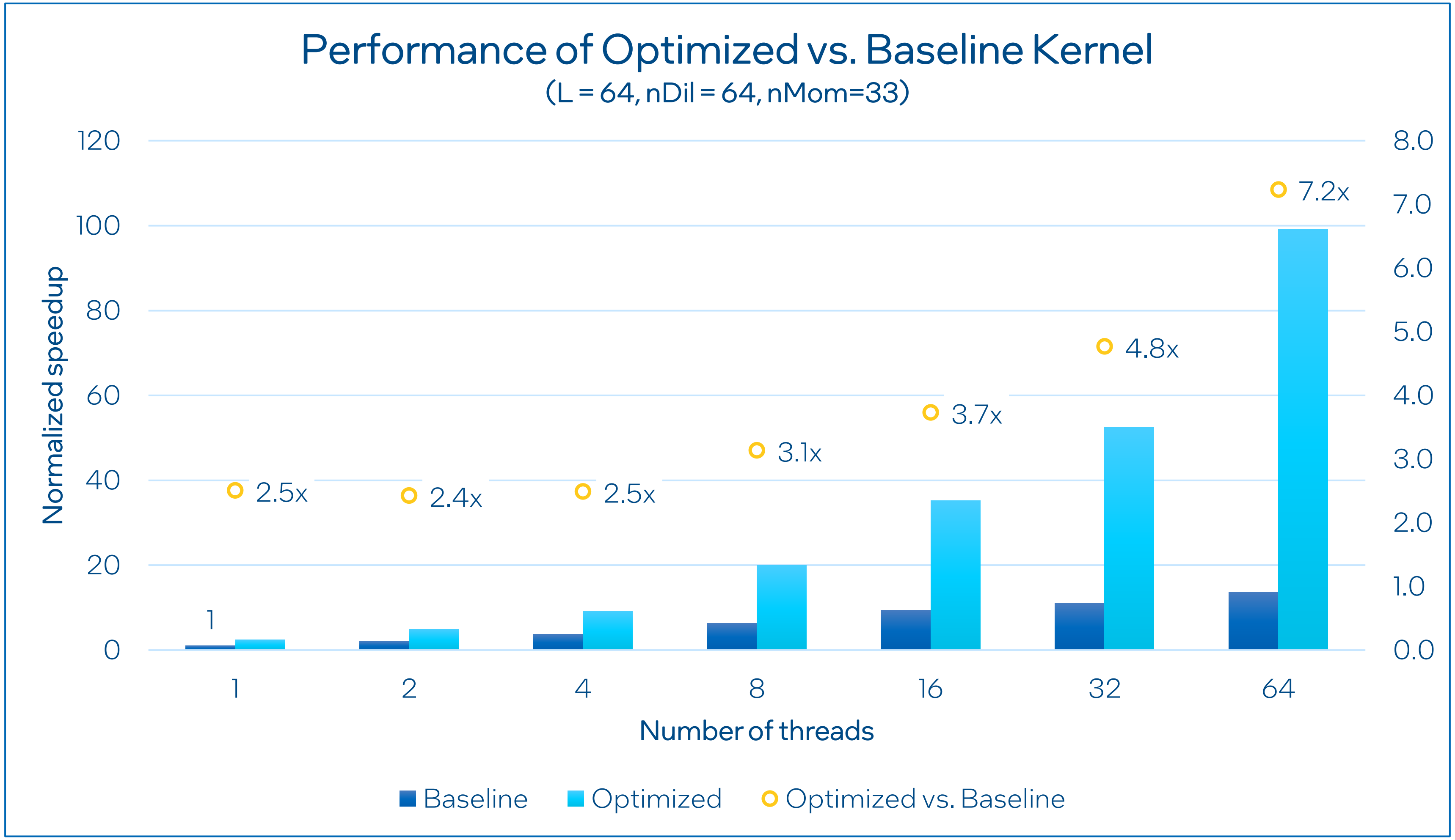
```
input: q1, q2, q3           // 3D array, [nD,nColor,nX]
       momBuf              // 2D array, [nMom, nX]
output: res                 // 4D array, [nMom, nD1, nD2, nD3]
intermediate: diq           // 4D array, [bsizeD1, bsizeD2, nColor, bsizeX]
              singlet       // 4D array, [bsizeD3, bsizeD2, bsizeD1, bsizeX]
              tmpBuf        // 2D array, [nD3, bsizeD2*bsizeD1,nMom]
#pragma omp parallel for collapse(2) firstprivate(q1, q2, q3)
for blockD1; do
  for blockD2; do
    init(tmpBuf, 0.)
    for blockX; do
      for d1 = 1 to bsizeD1; do
        for d2 = 1 to bsizeD2; do
          for iX = 1 to bsizeX; do
            calculate diq(:) = q1(~d1,:,iX) * q2(~d2,:,iX)
          end for
        end for
      end for
      for blockD3; do
        for d3 = 1 to bsizeD3; do
          for diq_elem in diq; do
            for iX = 1 to bsizeX; do
              calculate singlet(:) = diq_elem * q3(~d3,:,iX)
            end for
           end for
         end for
       tmpBuf += singlet * momBuf        // Using MKL JIT
     end for // blockX
     storing res <- tmpBuf
  end for // blockD2
end for // blockD1
```

Using:
- **Collapse:** parallelize multiple loops
- **Firstprivate**: declare "read-only" for q1, q2, q3.

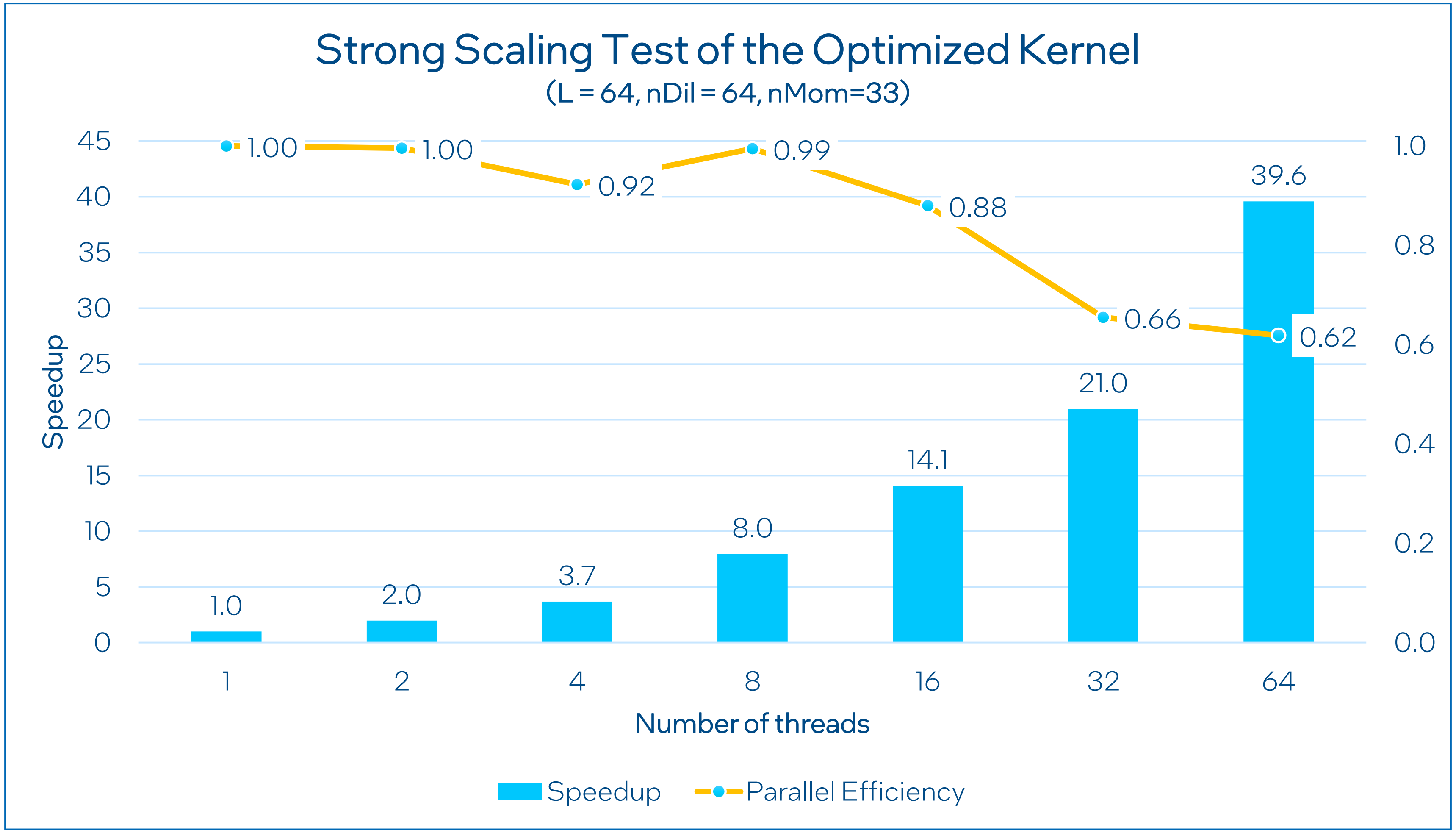
- Binding set:

$ OMP_PLACES=cores
$ OMP_PROC_BIND=close

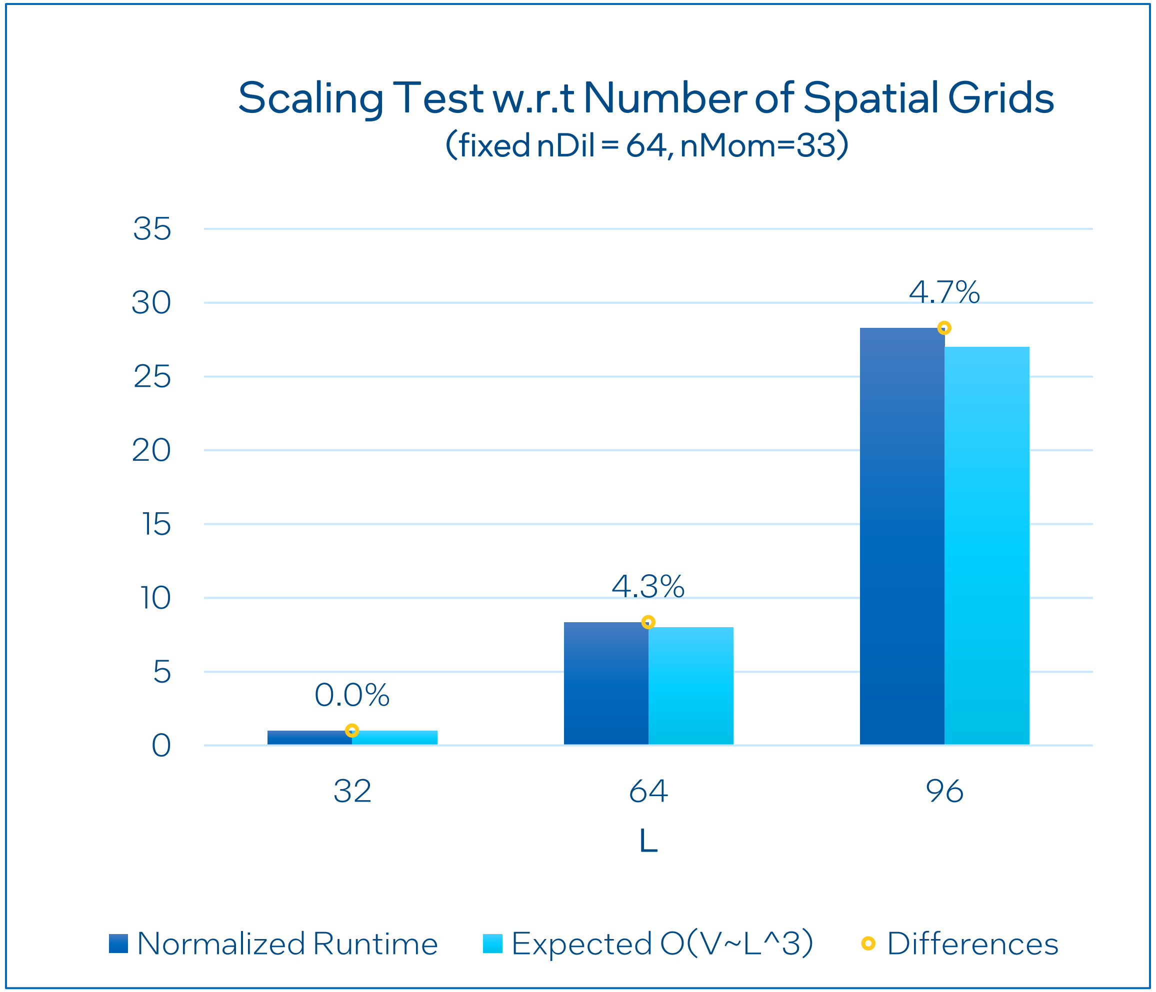
5. Node-level Optimization (3)

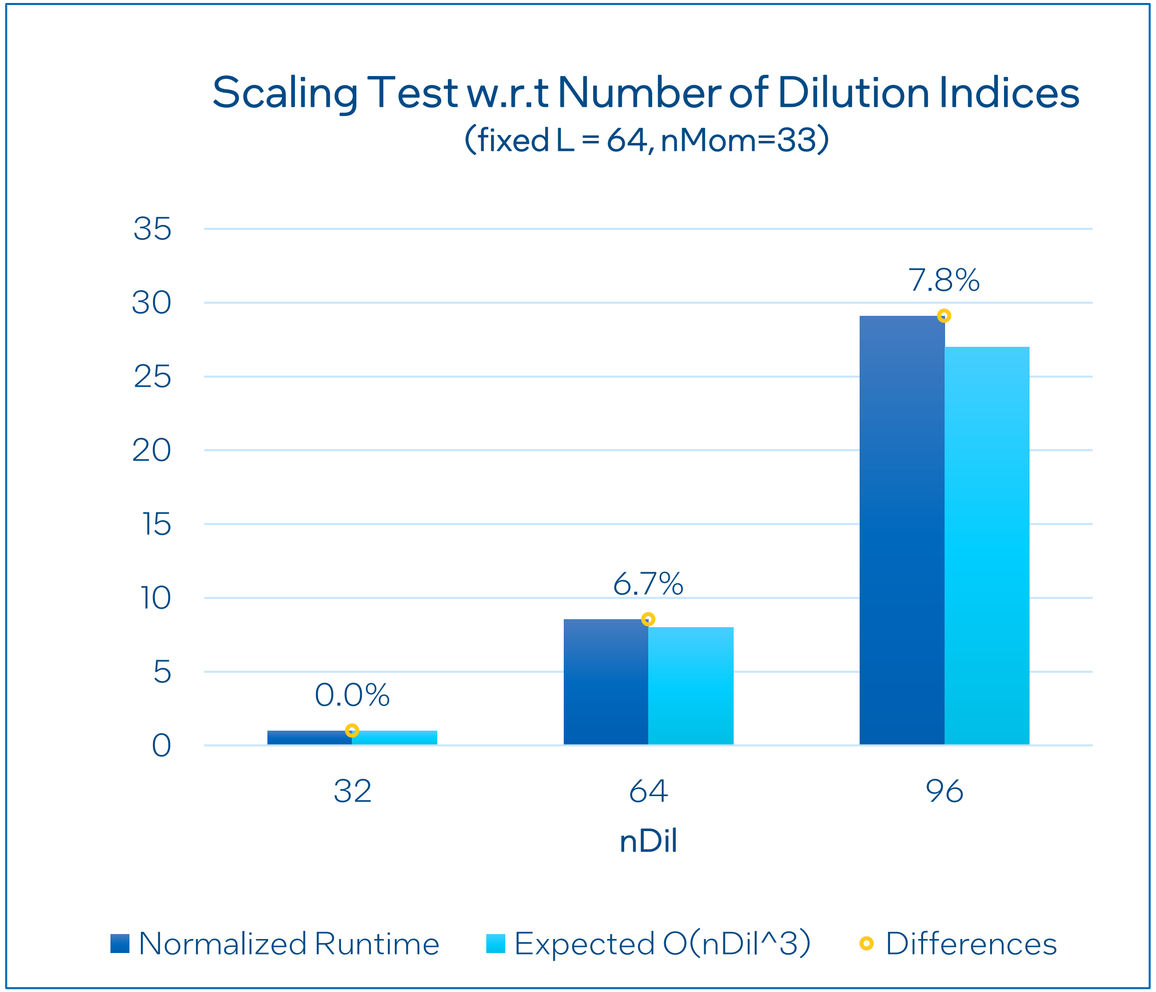


5. Node-level Optimization (4)



5. Node-level Optimization (5)





6. Conclusion

- Modern spectroscopy methods entail large measurement cost.
- Challenge: scale computation of observables to ever-larger problem sizes for simulations near the physical point.
- Optimized computation of stochastic LapH baryon blocks, achieving
  - up to **7.2x speedup** over previous QDP + gemm-based implementation
  - favorable scaling with core count, good parallel efficiency
  - good scaling with problem size (Ndil, V), no nasty surprises.
- Get ready for more ambitious production runs.

intel®

Backup (1)

$$\mathcal{B}_{\mathbf{p}}^{d_1 d_2 d_3} = \sum_{\mathbf{x}} \sum_{a,b,c} \mathrm{e}^{-\mathbf{i}\mathbf{p}\mathbf{x}} \, \varepsilon_{abc} \, q_{\mathbf{x}a}^{d_1} q_{\mathbf{x}b}'^{d_2} \, q_{\mathbf{x}c}''^{d_3}$$

$$\varepsilon_{abc} = \begin{cases} +1 & (abc) \in \{(123), (231), (312)\}, \\ -1 & (abc) \in \{(321), (132), (213)\}, \\ 0 & \text{otherwise.} \end{cases}$$

Backup (2) – Different approaches for baryon constructions

$$q^{d}_{a\mathbf{x}} = \sum_{l=1}^{N_{\mathrm{ev}}} Q_{dl}\phi^{l}_{a\mathbf{x}}$$

$$\mathcal{T}^{lmn}_{\mathbf{p}} = \sum_{\mathbf{x}} \mathrm{e}^{-\mathbf{i}\mathbf{p}\mathbf{x}} \sum_{a,b,c} \varepsilon_{abc}\, \phi^{l}_{\mathbf{x}a}\phi^{m}_{\mathbf{x}b}\phi^{n}_{\mathbf{x}c}$$

$$\mathcal{B}^{d_1 d_2 d_3}_{\mathbf{p}} = \sum_{\mathbf{x}} \sum_{a,b,c} \mathrm{e}^{-\mathbf{i}\mathbf{p}\mathbf{x}}\, \varepsilon_{abc}\, q^{d_1}_{\mathbf{x}a} q'^{d_2}_{\mathbf{x}b}\, q''^{d_3}_{\mathbf{x}c}$$

$$\mathcal{B}^{d_1 d_2 d_3}_{\mathbf{p}} = \sum_{l,m,n=1}^{N_{\mathrm{ev}}} Q_{d_1 l} Q'_{d_2 m} Q''_{d_3 n} \mathcal{T}^{lmn}_{\mathbf{p}}$$

Backup (3)- Reference Implementation with OpenMP

```
input: q1, q2, q3          // 3D array, [nD,nX,nColor]
       momBuf             // 2D array,  [nMom, nX]
output: res                // 4D array, [nMom, nD1, nD2, nD3]
intermediate: diq          // 2D array, [nX, nColor]
              singlet      // 1D array, [nRows, nX]
for d1 = 1 to nD1; do      // dilution index 1
  for d2 = 1 to nD2; do    // dilution index 2
#pragma omp parallel for
    for iX = 1 to nX; do
      calculate diq(iX,:) = q1(d1,iX,:) * q2(d2,iX,:)
    end for
    rowIndex = 0;
    for d3 = 1 to nD3; do  // dilution index 3
#pragma omp parallel for
      for iX = 1 to nX; do
        calculate singlet(rowIndex, iX) = diq(iX,:) * q3(d3,iX,:)
      end for
      // blocked momentum projection
      rowIndex++;
      if rowIndex == nrows
         // With BLAS Level 3 – GEMM
         res(:, d1, d2, d3 – nrows : d3) = momBuf * singlet
         rowIndex = 0
      end if
    end for
  end for
end for
```

Backup (4) - Profiling for OpenMP

```
 CPU
     SP GFLOPS: 0.000
     DP GFLOPS: 2,462.079
     x87 GFLOPS: 0.000
     CPI Rate: 0.456
     Average CPU Frequency: 2.329 GHz
     Total Thread Count: 64
Vectorization: 100.0% of Packed FP Operations
    Instruction Mix
        DP FLOPs: 100.0% of uOps
            Packed: 100.0% from DP FP
                128-bit: 11.0% from DP FP
                256-bit: 0.0% from DP FP
                512-bit: 89.0% from DP FP
            Scalar: 0.0% from DP FP
        x87 FLOPs: 0.0% of uOps
        Non-FP: 0.0% of uOps
    FP Arith/Mem Rd Instr. Ratio: 4.066
    FP Arith/Mem Wr Instr. Ratio: 69.245

 Effective Physical Core Utilization: 82.8% (52.970 out of 64)
    Effective Logical Core Utilization: 41.9% (53.635 out of 128)
    Serial Time (outside parallel regions): 0.029s (0.4%)
    Parallel Region Time: 8.066s (99.6%)
        Estimated Ideal Time: 6.862s (84.8%)
        OpenMP Potential Gain: 1.204s (14.9%)
```

```
CPU Time: 446.248s
    Memory Bound: 22.5% of Pipeline Slots
        L1 Bound: 4.7% of Clockticks
        L2 Bound: 1.5% of Clockticks
        L3 Bound: 9.0% of Clockticks
        DRAM Bound: 0.7% of Clockticks
        Store Bound: 0.0% of Clockticks
        NUMA: % of Remote Accesses: 75.0%
        UPI Utilization Bound: 0.0% of Elapsed Time
Time
    Loads: 667,973,038,590
    Stores: 39,202,676,045
    LLC Miss Count: 0
        Local DRAM Access Count: 0
        Remote DRAM Access Count: 0
        Remote Cache Access Count: 0
    Average Latency (cycles): 23
    Total Thread Count: 64
    Paused Time: 2.456s

Bandwidth Utilization
Bandwidth Domain                  Platform Maximum  Observed Maximum  Average
--------------------------------  ----------------  ----------------
DRAM, GB/sec                      354                         46.900   13.052
DRAM Single-Package, GB/sec       177                         28.200   10.753
UPI Utilization Single-link, (%)  100                         34.500   11.405
```