

Memory Traffic and Complete Application Profiling with PAPI Multi- Component Measurements

Daniel Barry, Heike Jagode,
Anthony Danalis, Jack Dongarra

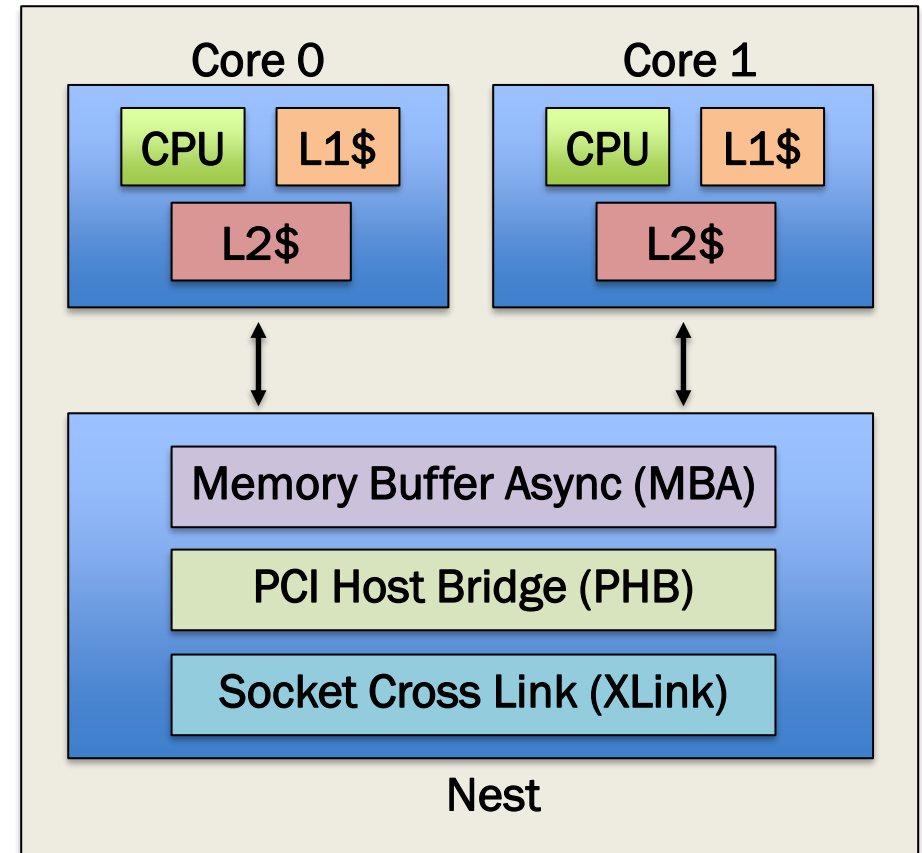
May 15, 2023



THE UNIVERSITY OF
TENNESSEE
KNOXVILLE

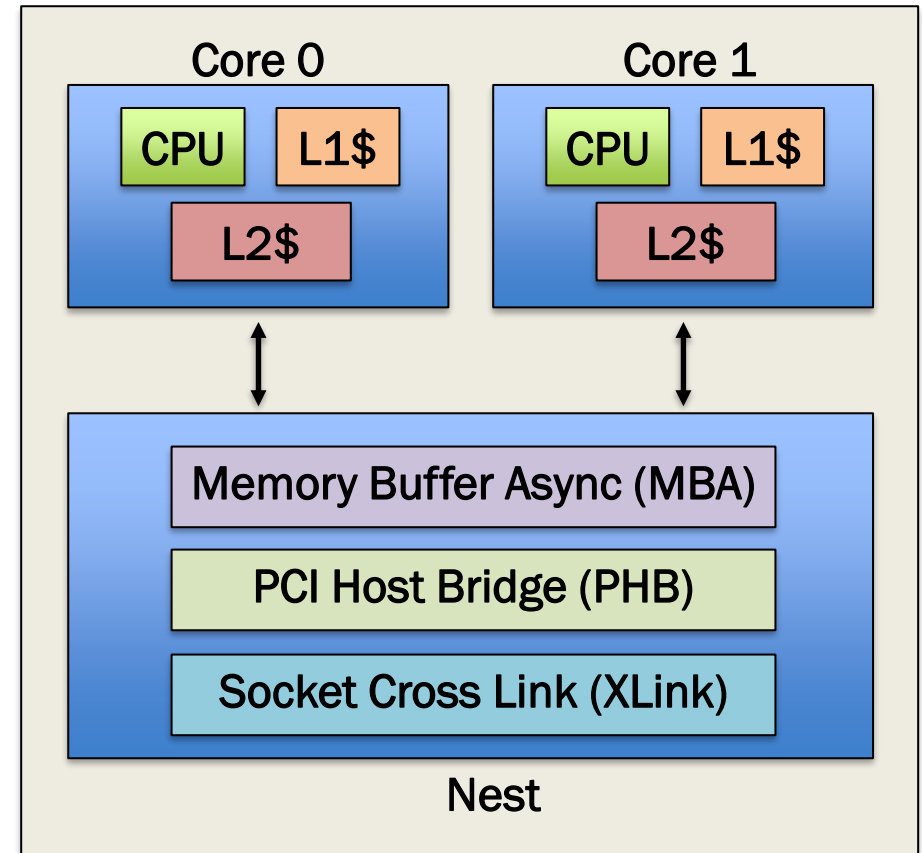
Motivation

- Accurately monitor mem. traffic



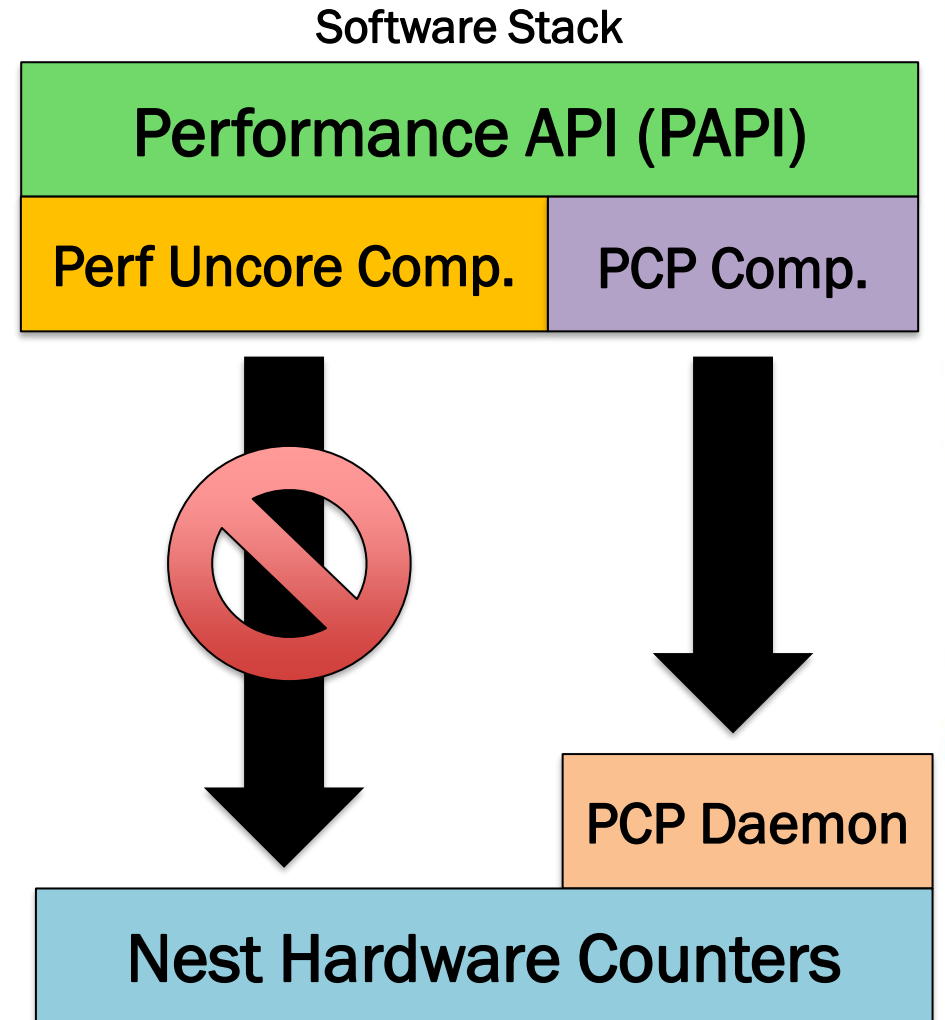
Motivation

- Accurately monitor mem. traffic
- Mem. traffic counters are in the *nest*



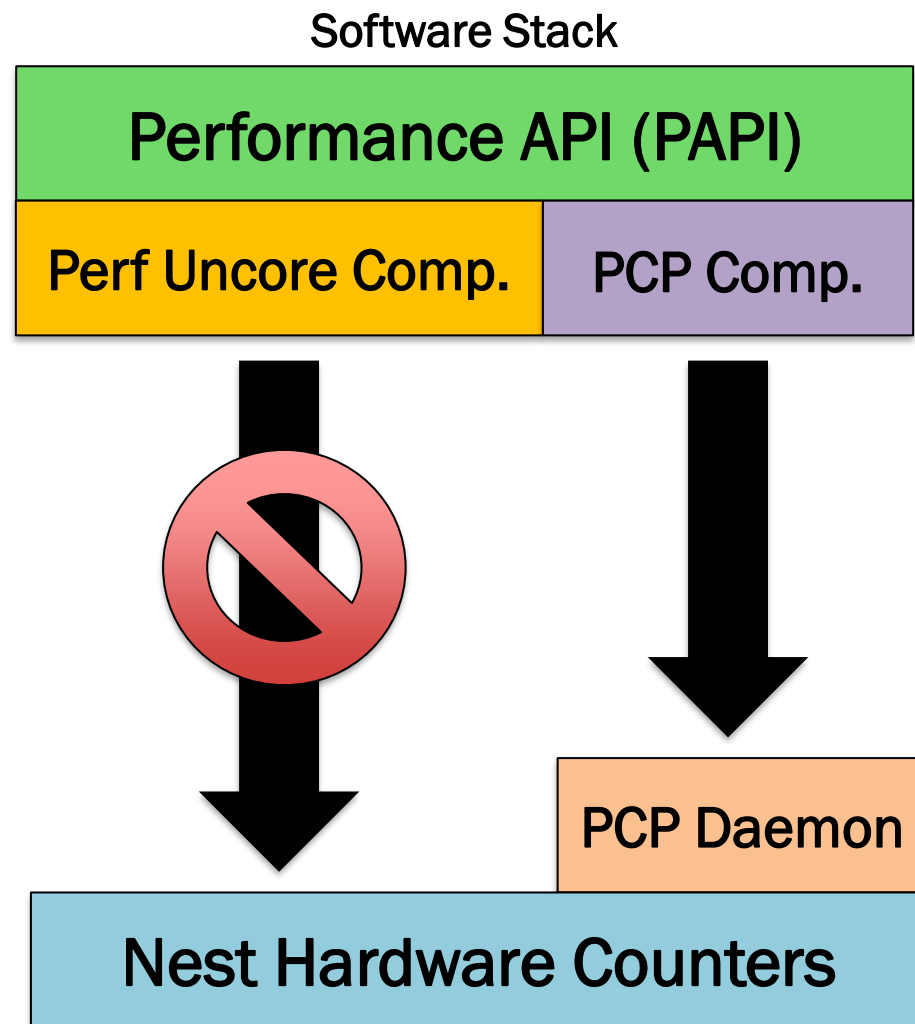
Motivation

- Accurately monitor mem. traffic
- Mem. traffic counters are in the *nest*
- Require elevated privileges to access



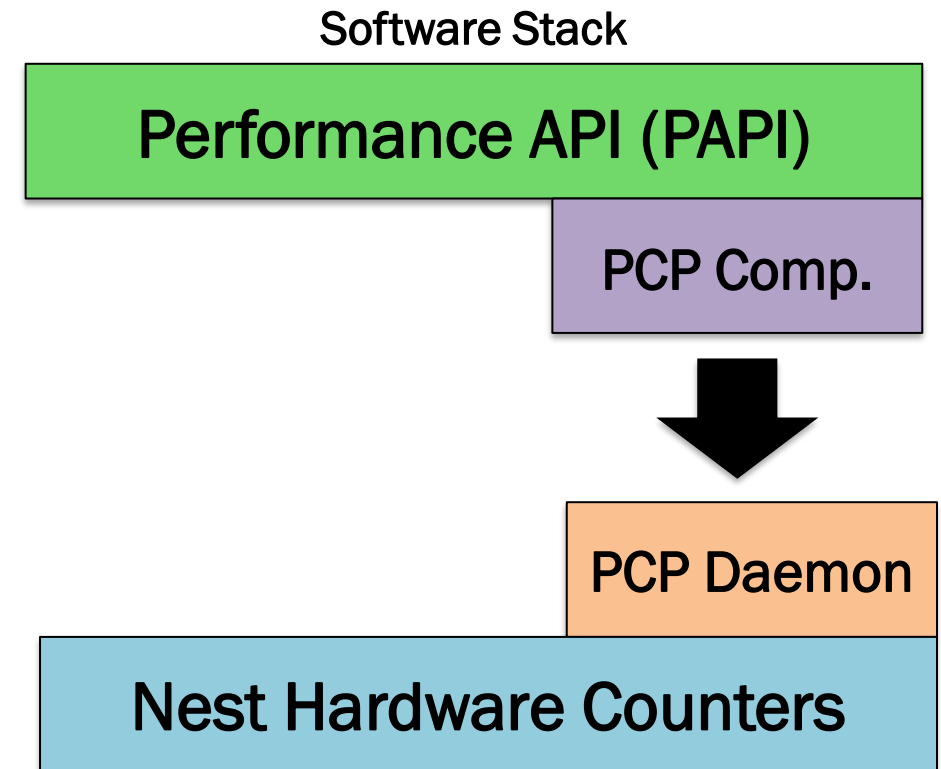
Motivation

- Accurately monitor mem. traffic
- Mem. traffic counters are in the *nest*
- Require elevated privileges to access
- IBM chose Performance Co-Pilot (PCP)
 - Third-party performance monitoring infrastructure



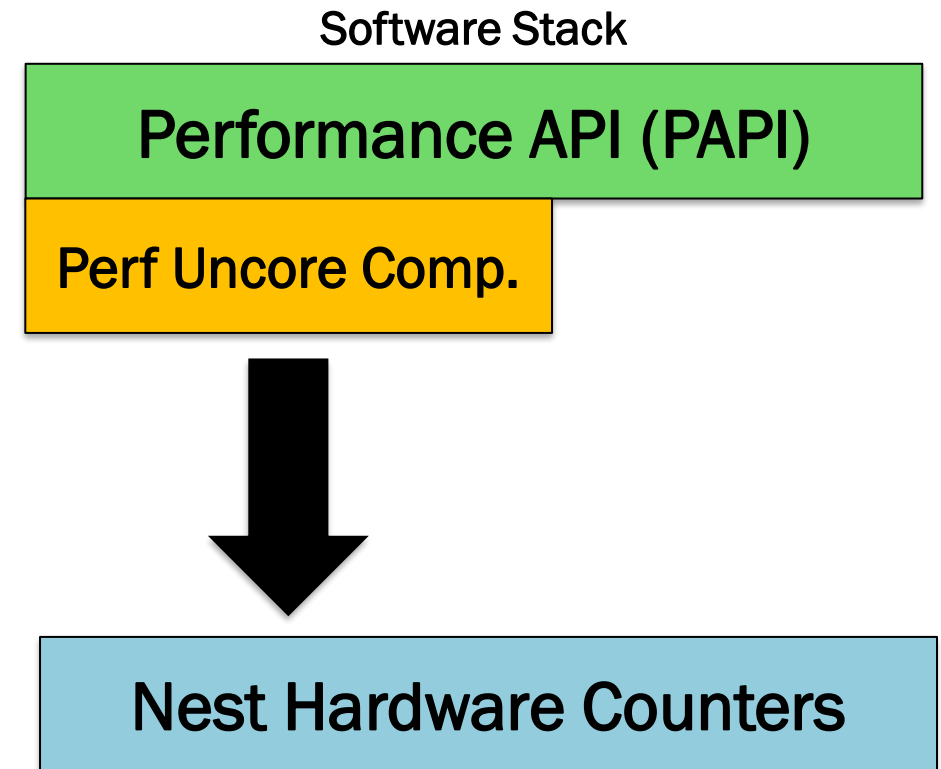
Computing Environments

- Summit (ORNL)
 - Nest access via PCP daemon.
- Tellico (UTK)
 - Direct access to nest counters (elevated privileges).
 - Baseline measurement to gauge overhead of PCP.



Computing Environments

- Summit (ORNL)
 - Nest access via PCP daemon.
- Tellico (UTK)
 - Direct access to nest counters (elevated privileges).
 - Baseline measurement to gauge overhead of PCP.



Validation Experiments

1. Common BLAS Kernels
2. 3D-FFT Case Study
3. Profiling Full Applications with Multiple Components of PAPI

Validation Experiments

1. Common BLAS Kernels
2. 3D-FFT Case Study
3. Profiling Full Applications with Multiple Components of PAPI

How Reliably Do Memory Events Correspond to BLAS Kernels?

GEMM

```
for ( i = 0; i < N; i++ )  
  for ( j = 0; j < N; j++ ) {  
    sum = 0.0;  
    for ( k = 0; k < N; k++ )  
      sum += A[i][k]*B[k][j];  
    C[i][j] = sum;  
  }
```

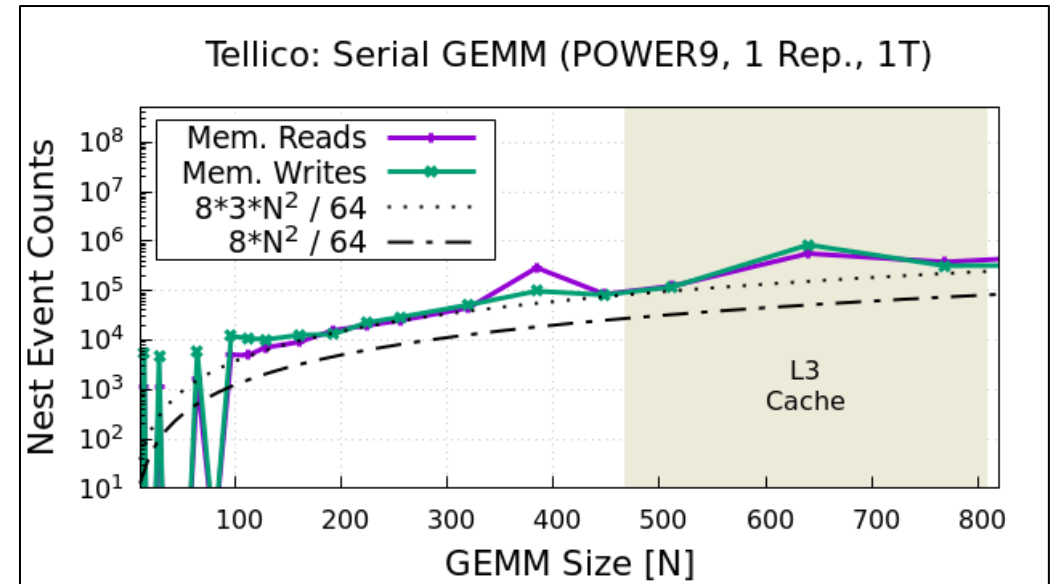
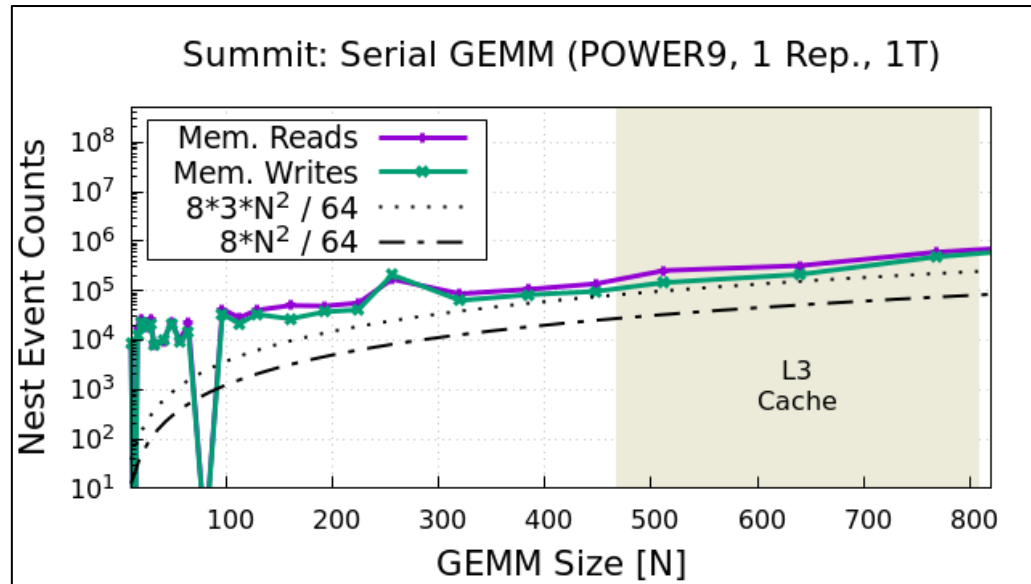
$$C = A * B$$

GEMV

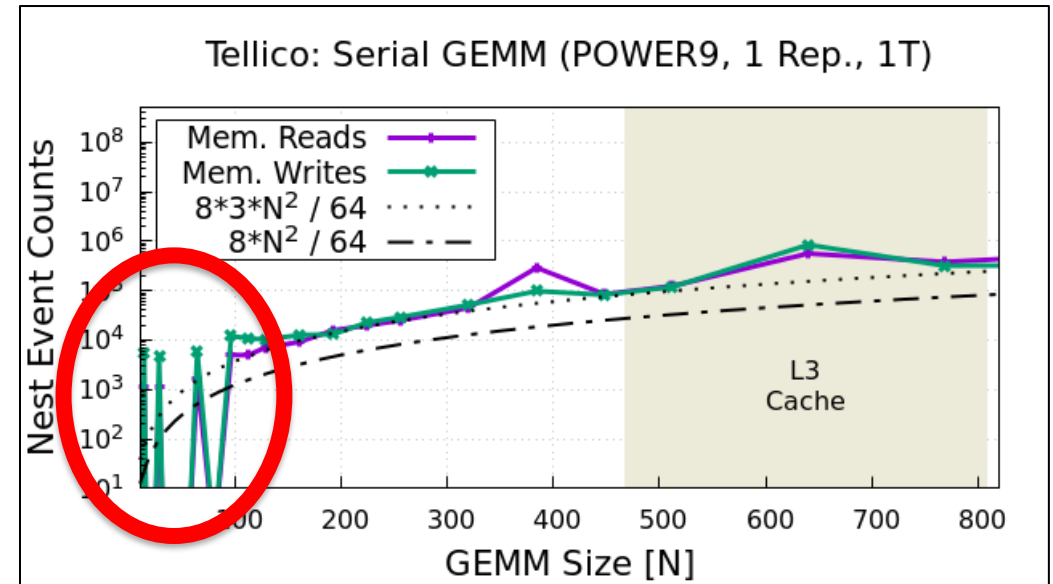
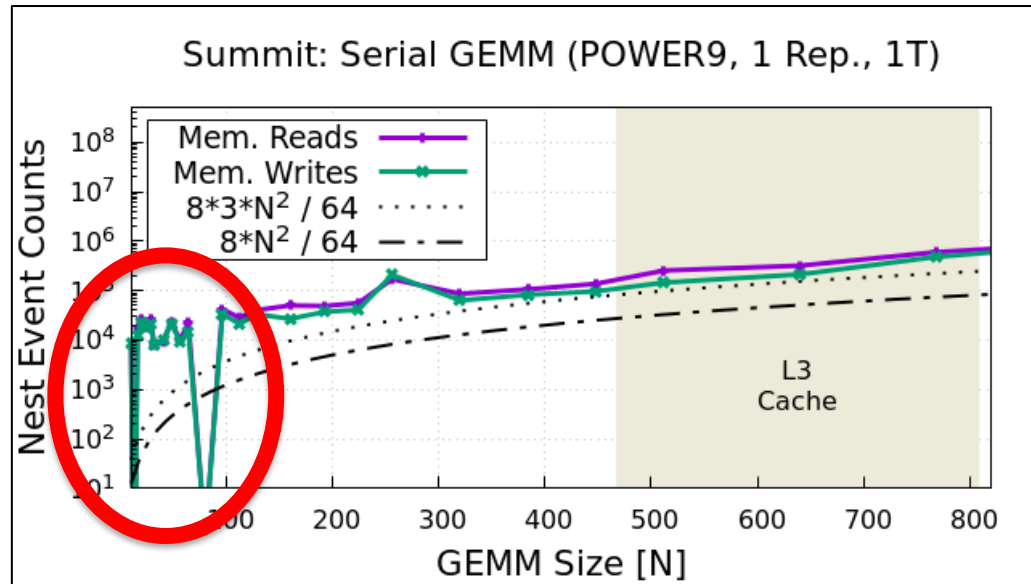
```
for ( i = 0; i < M; i++ ) {  
  sum = 0.0;  
  for ( k = 0; k < N; k++ )  
    sum += A[i][k]*x[k];  
  y[i] = sum;  
}
```

$$y = A * x$$

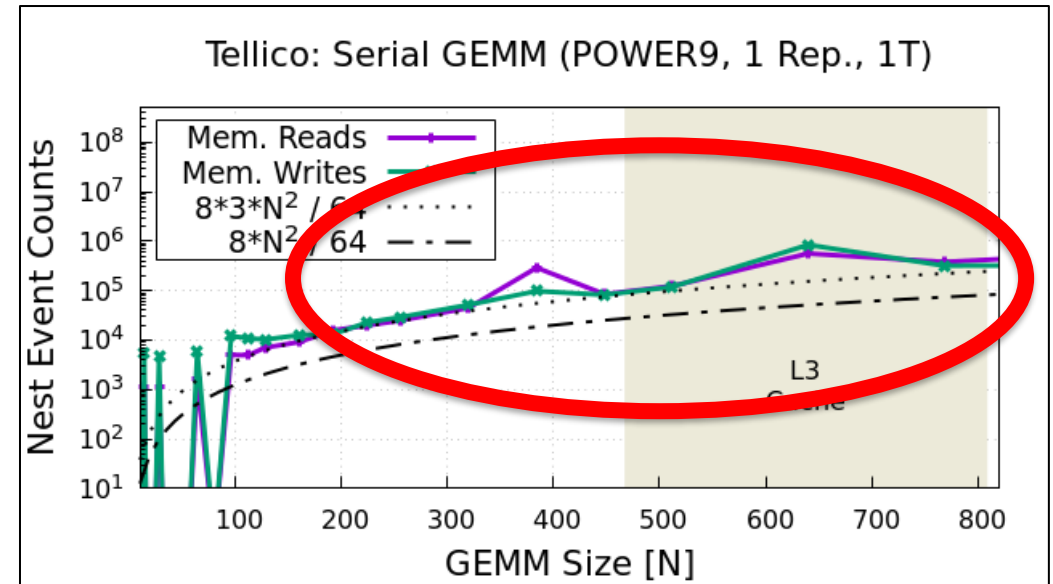
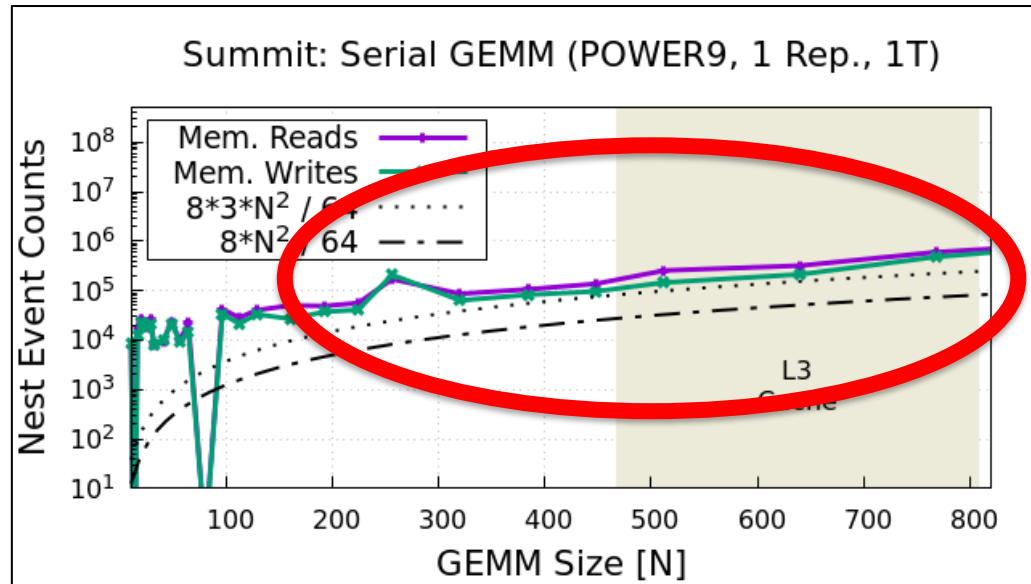
Does Infrastructure Make a Difference?



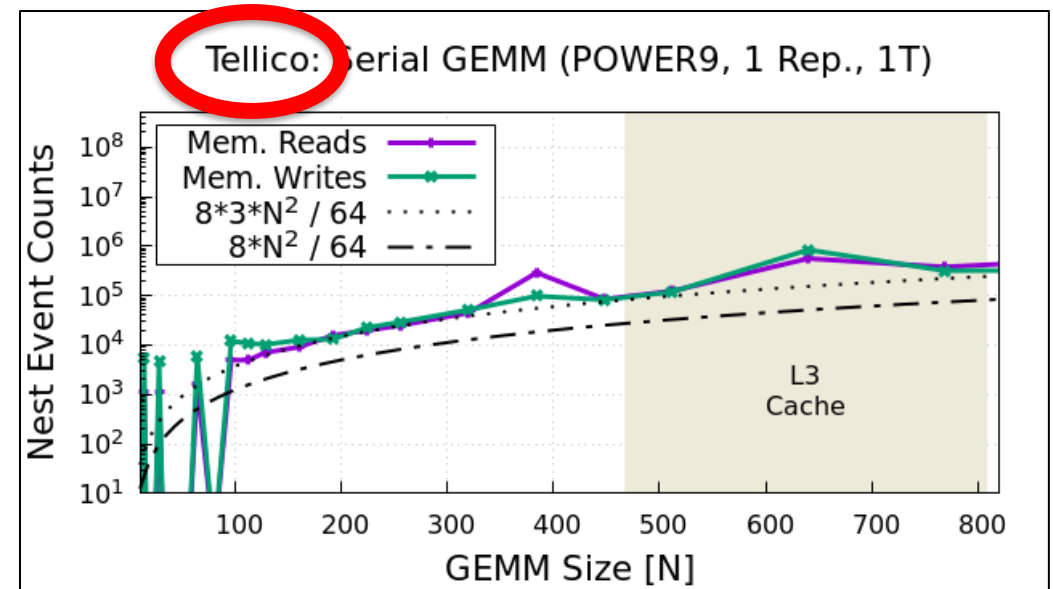
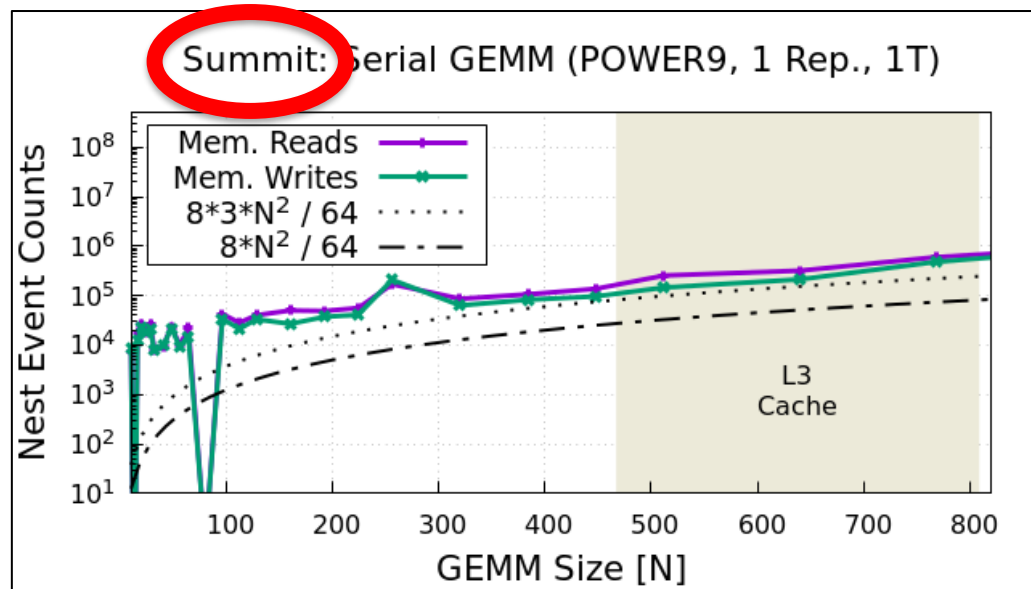
Does Infrastructure Make a Difference?



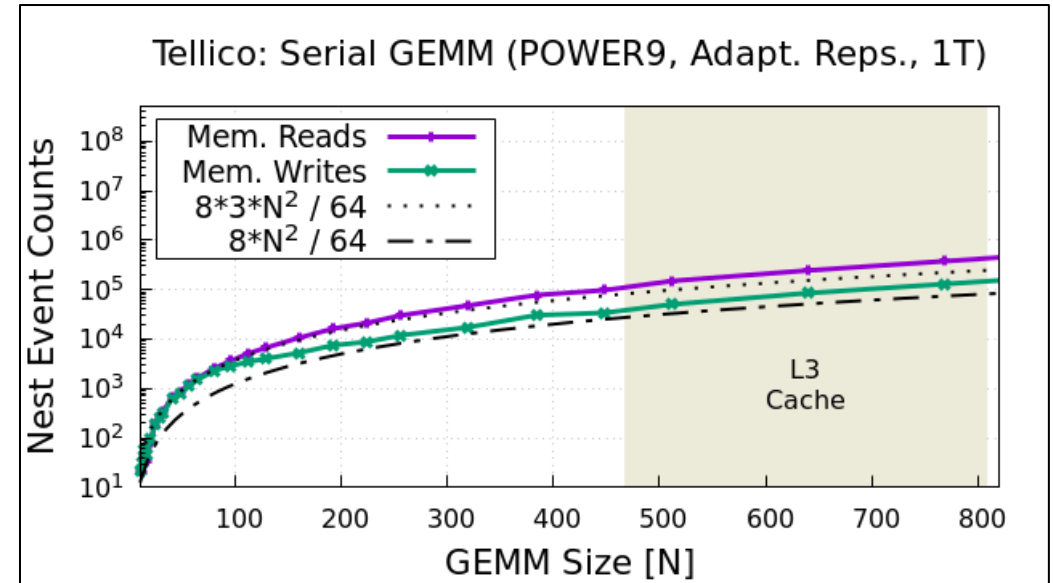
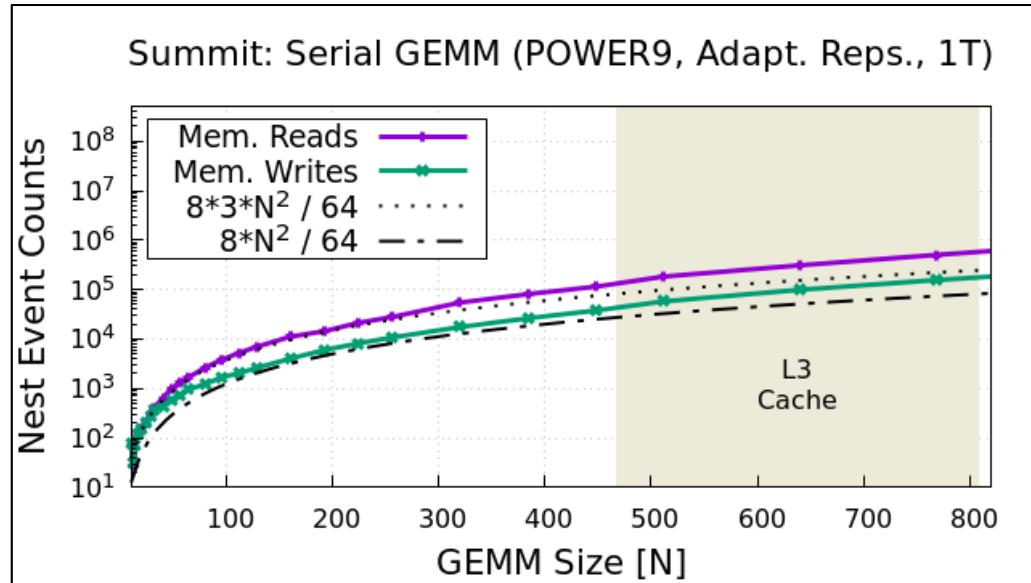
Does Infrastructure Make a Difference?



Does Infrastructure Make a Difference?



The More GEMMs the Merrier

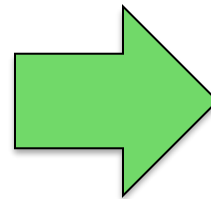


Using a Batched GEMM

- “Batched” = Each core executes a 1-thread GEMM (using OpenMP).

GEMM

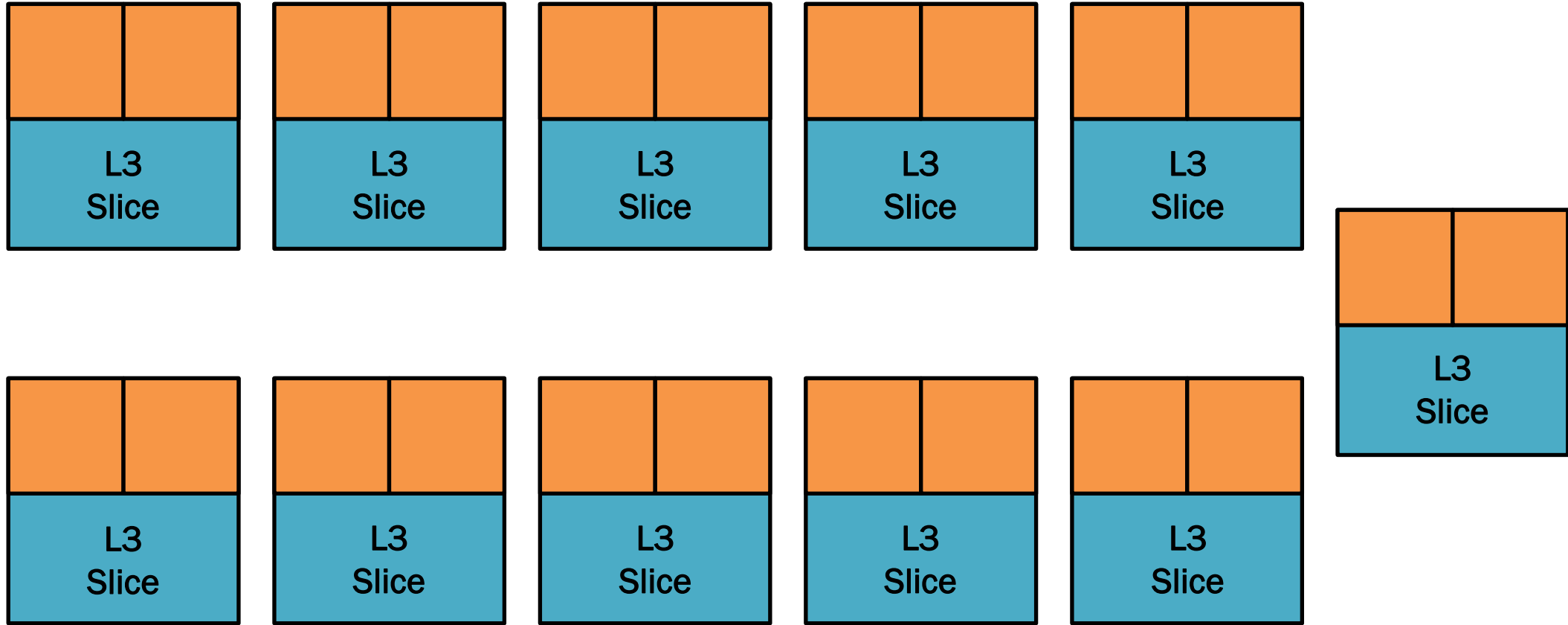
```
for ( i = 0; i < N; i++ )  
  for ( j = 0; j < N; j++ ) {  
    sum = 0.0;  
    for ( k = 0; k < N; k++ )  
      sum += A[i][k]*B[k][j];  
    C[i][j] = sum;  
  }
```




Batched GEMM

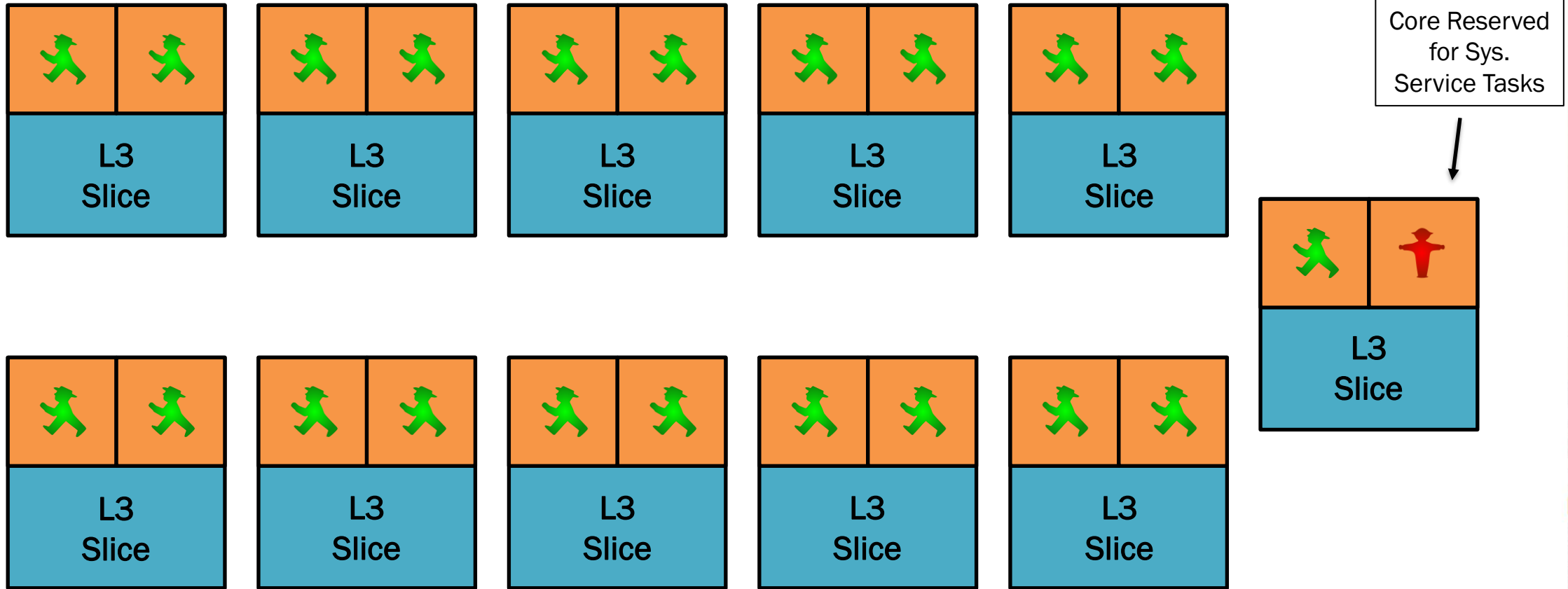
```
#pragma omp parallel for schedule(static)  
for ( idx = 0; idx < numThreads; idx++ )  
  for ( i = 0; i < N; i++ )  
    for ( j = 0; j < N; j++ ) {  
      sum = 0.0;  
      for ( k = 0; k < N; k++ )  
        sum += A[idx][i][k] * B[idx][k][j];  
      C[idx][i][j] = sum;  
    }
```

IBM POWER9 Core Topology



 = 1 Core




IBM POWER9 Core Topology



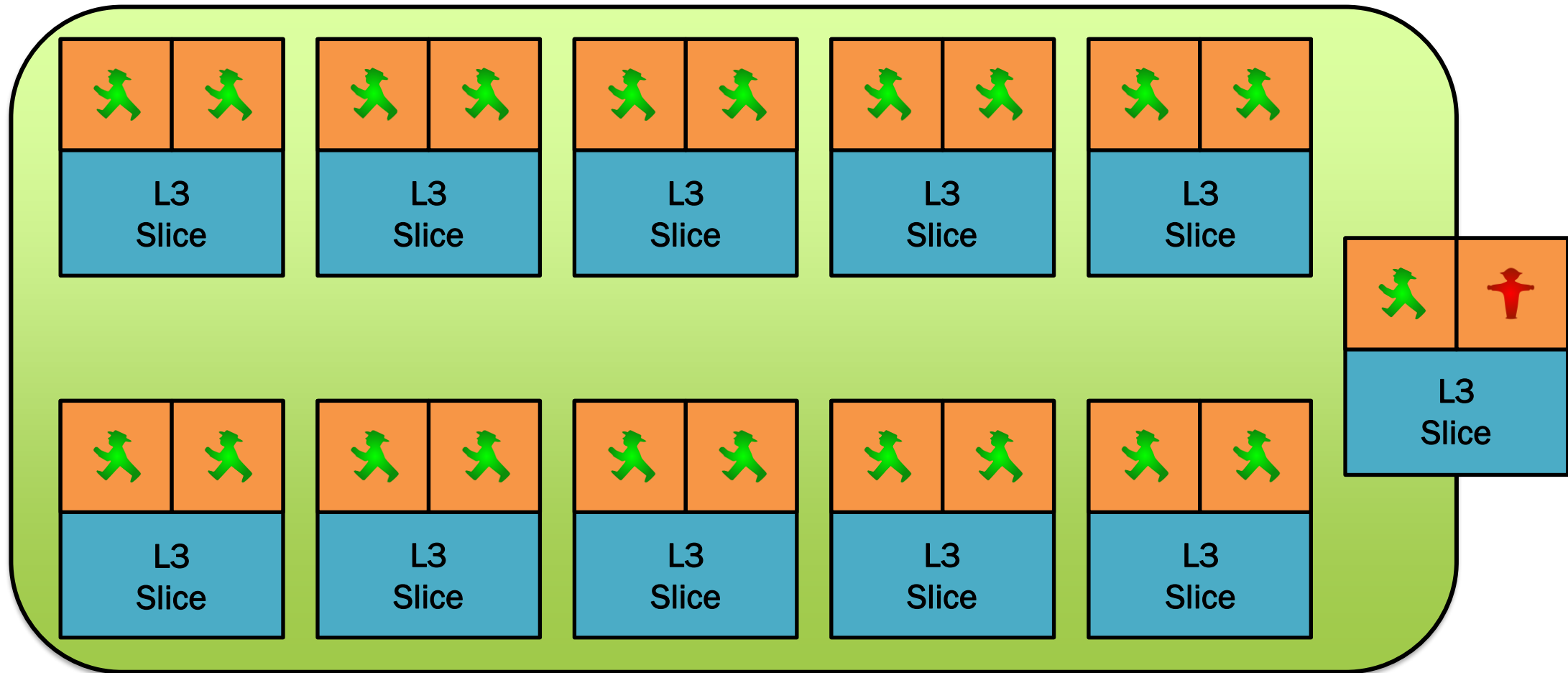
 = 1 Core  = 1 Thread

Batched GEMM



 = 1 GEMM  = 1 Core  = 1 Thread

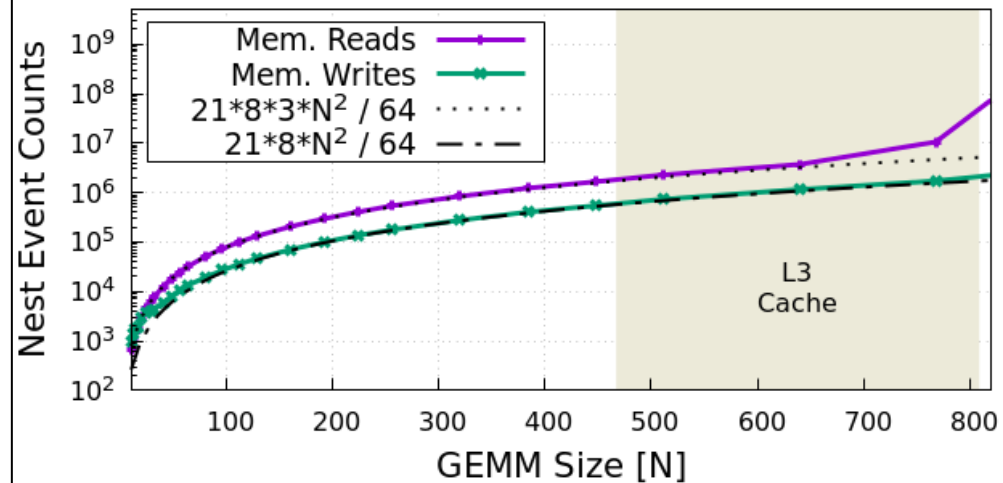
Shared-Work GEMM



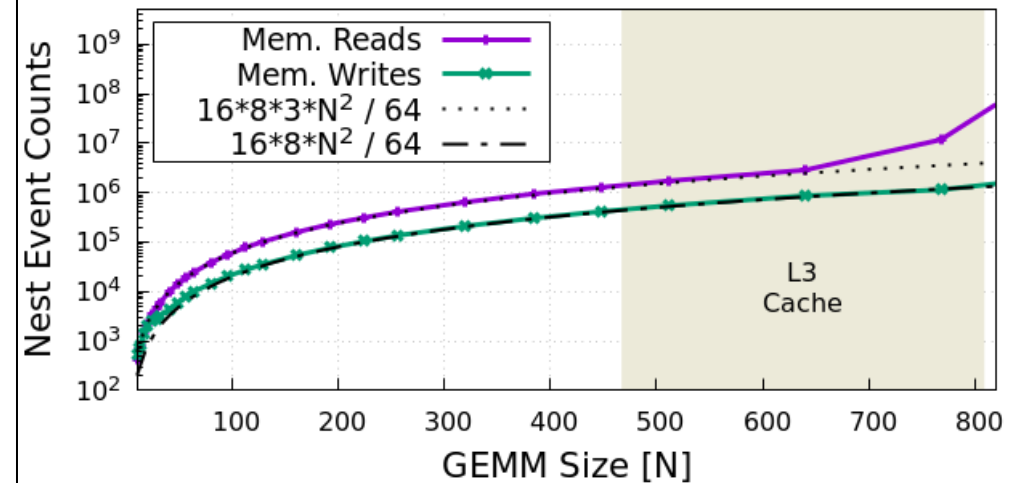
 = 1 GEMM  = 1 Core  = 1 Thread

Results from Batched GEMM

Summit: Batched GEMM (POWER9, Adapt. Reps., 21T)



Tellico: Batched GEMM (POWER9, Adapt. Reps., 16T)



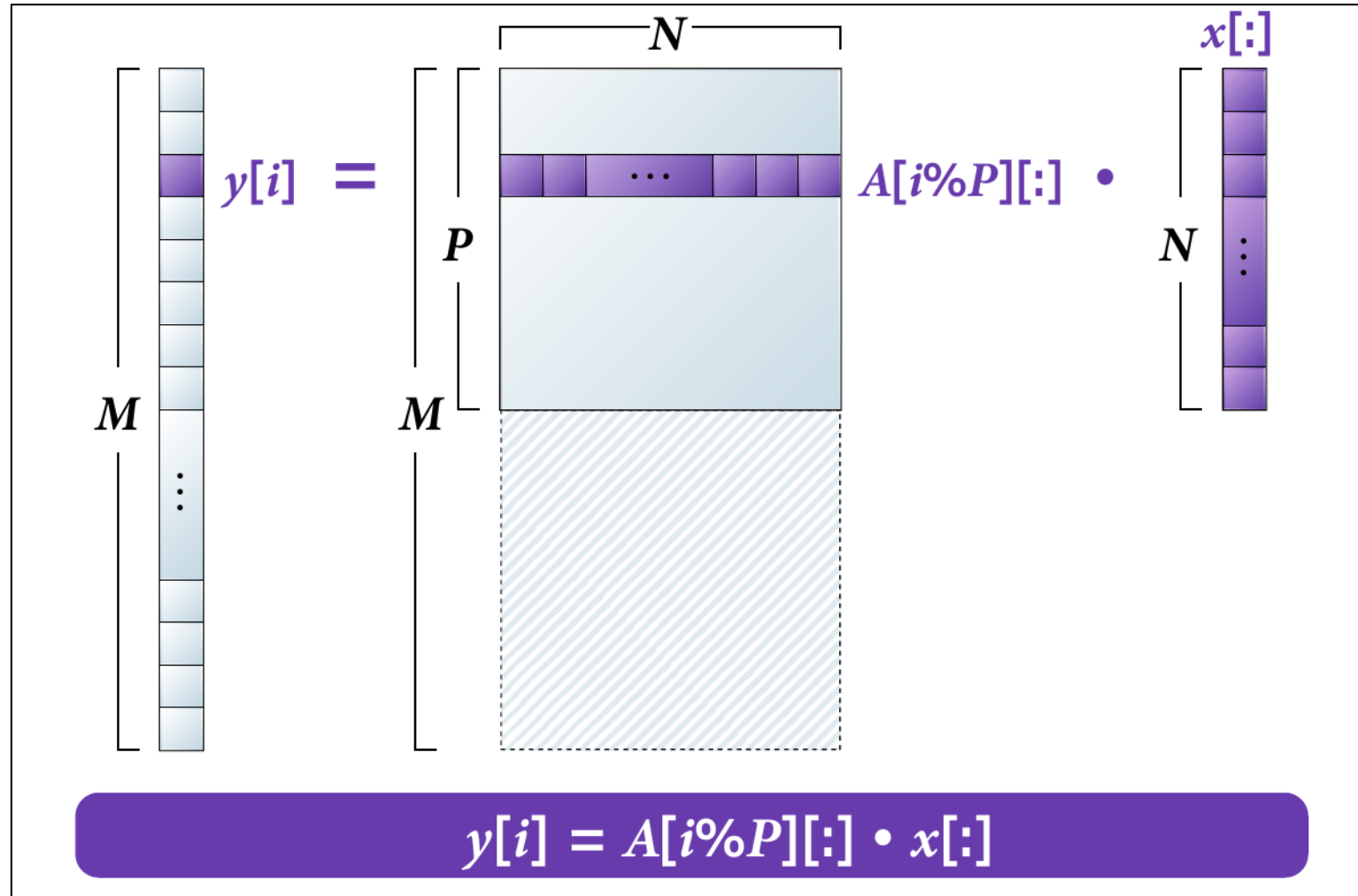
Caveats of the GEMV

- GEMM experiments have shown need for lots of work.
- GEMV does even less work than GEMM:

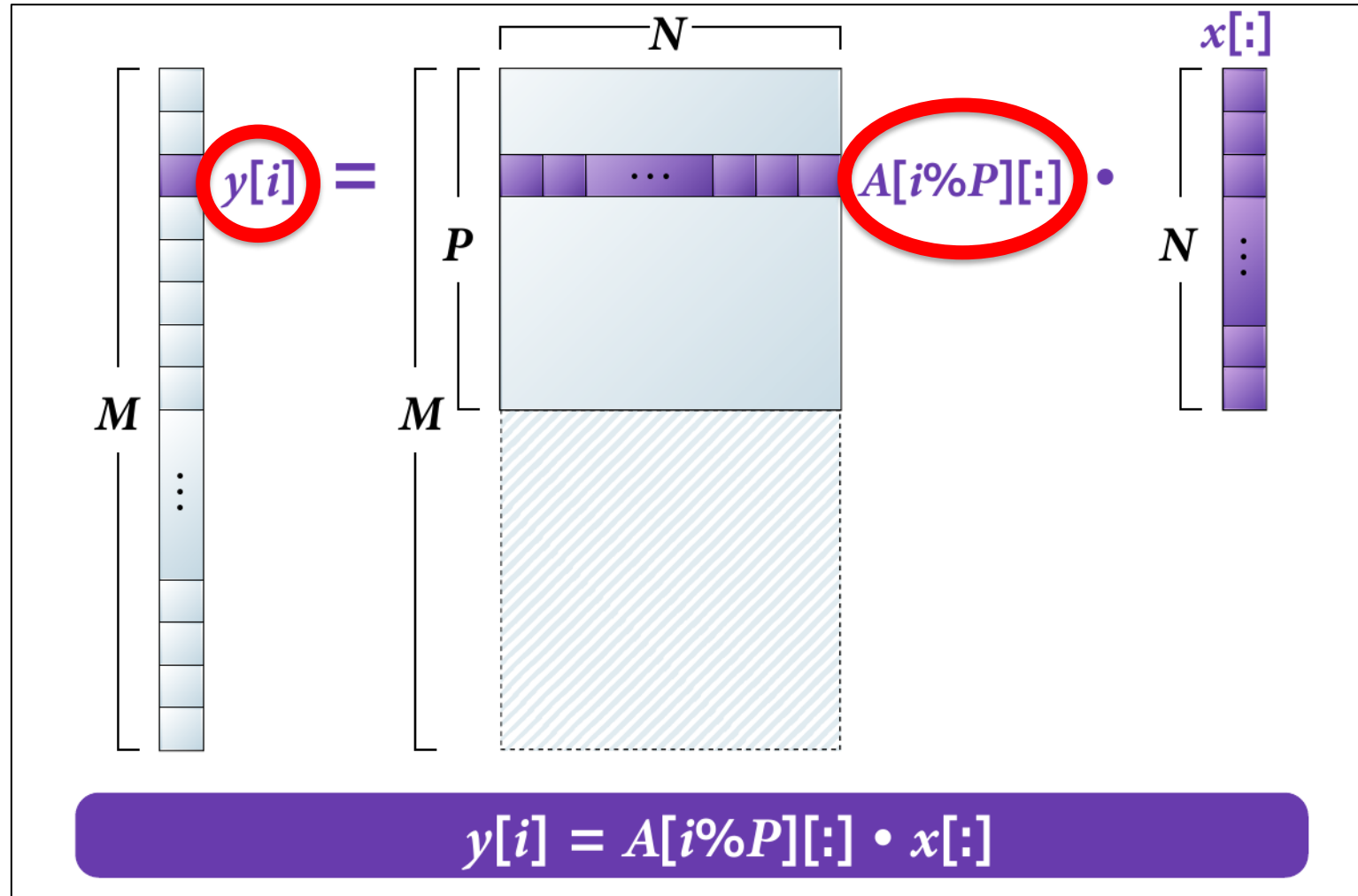
$$y = A * x$$

- For an $M \times N$ matrix A , GEMV's reading is $MN+M+N$.
- But writing *is only* M .
- *We run out of memory before M is large enough for meaningful measurements.*

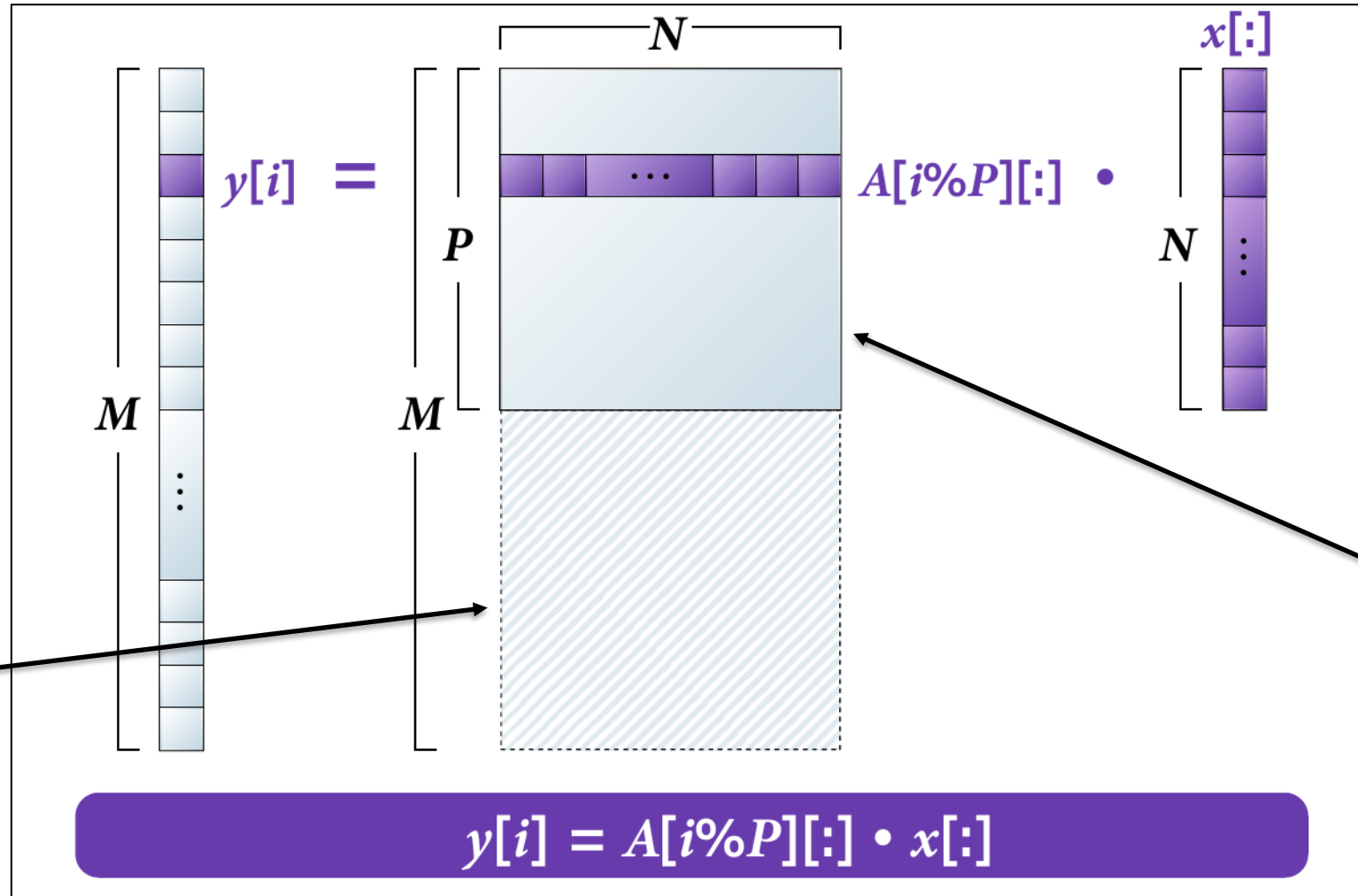
Capping the Size of GEMV



Capping the Size of GEMV



Capping the Size of GEMV

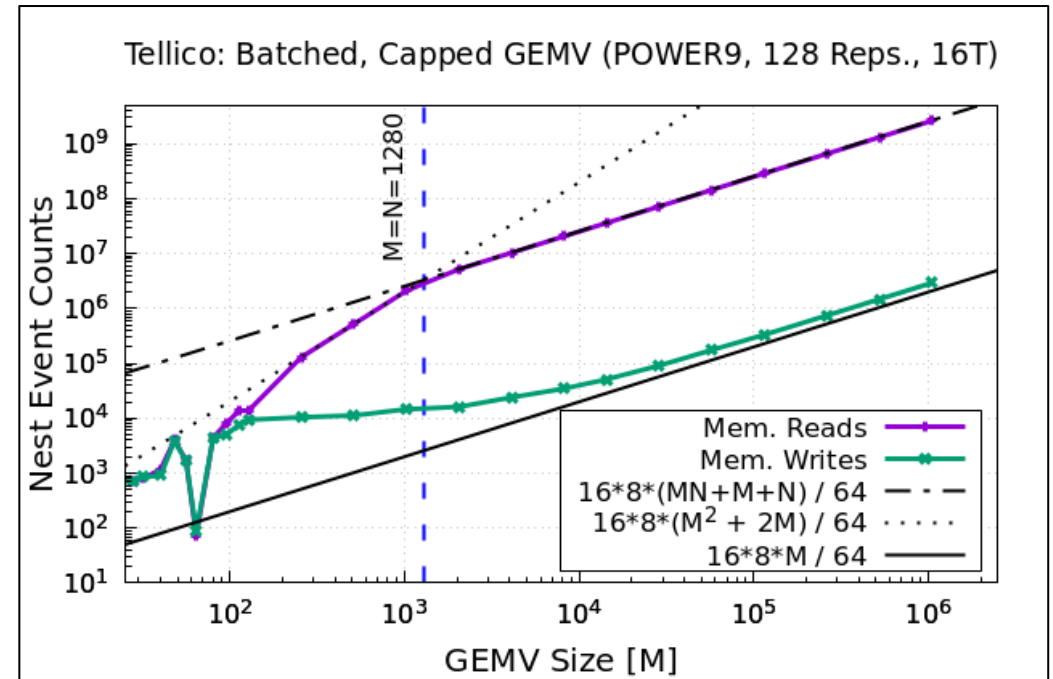
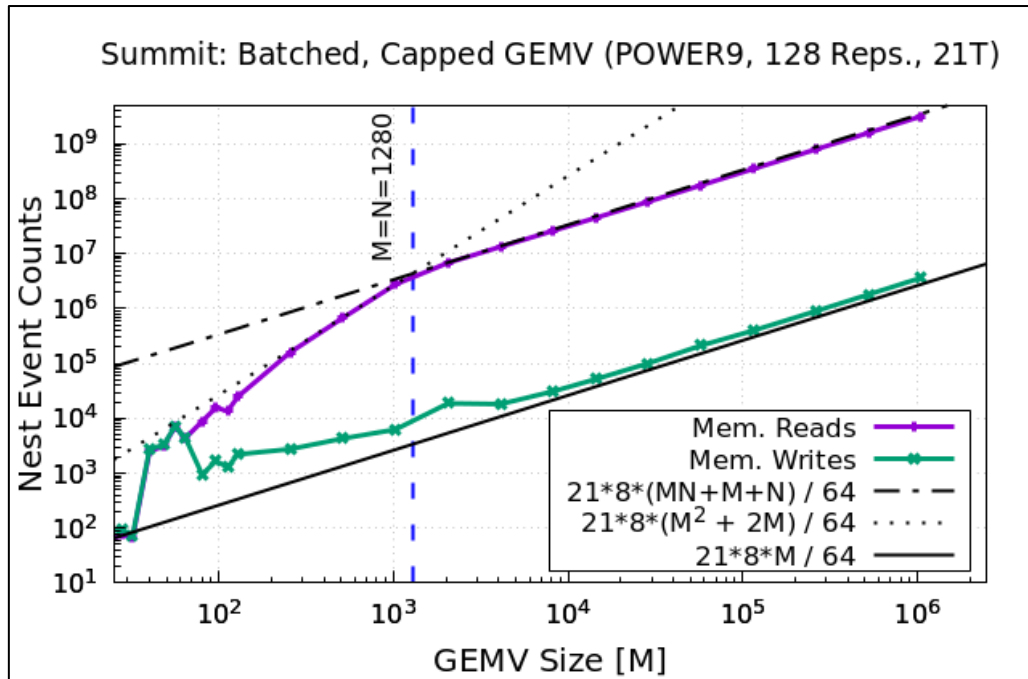


Memory Saved per A Matrix

Actual Memory Usage for A

$$y[i] = A[i\%P][:] \cdot x[:]$$

Batch of Capped GEMVs



Validation Experiments

1. Common BLAS Kernels
2. 3D-FFT Case Study
3. Profiling Full Applications with Multiple Components of PAPI

Case Study: 3D-FFT

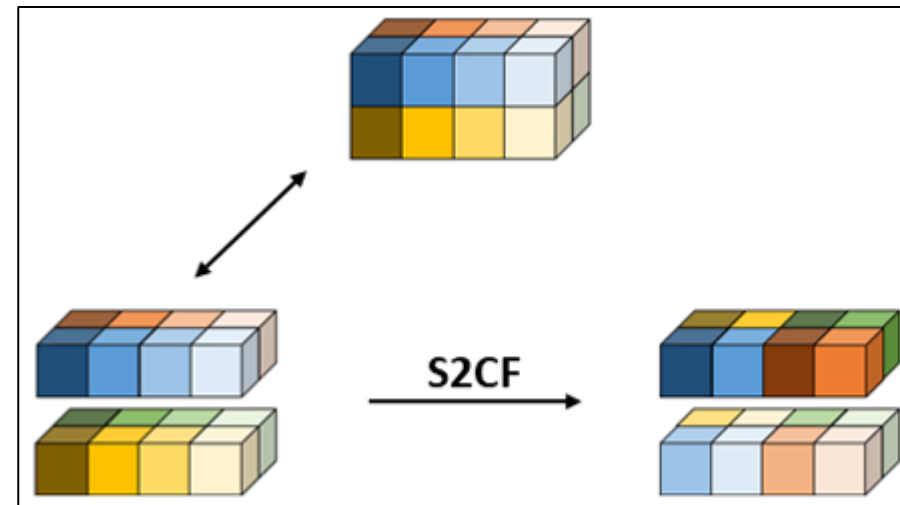
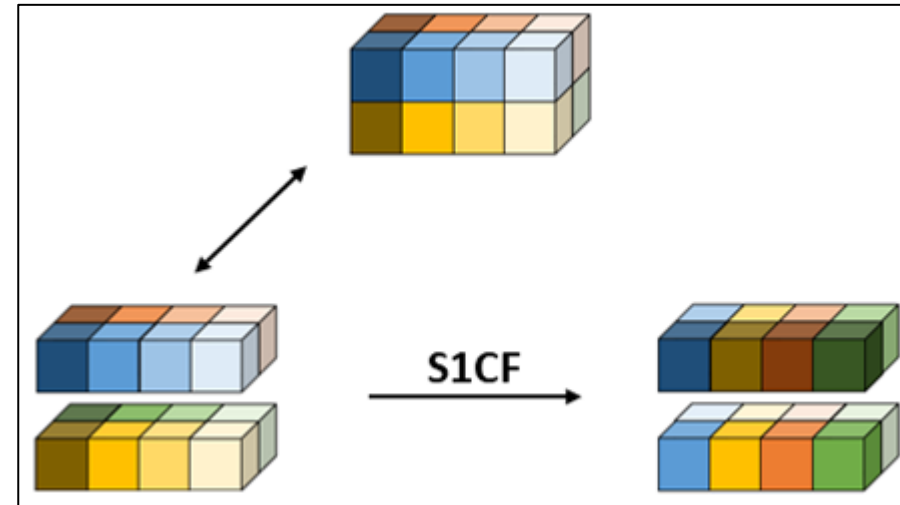
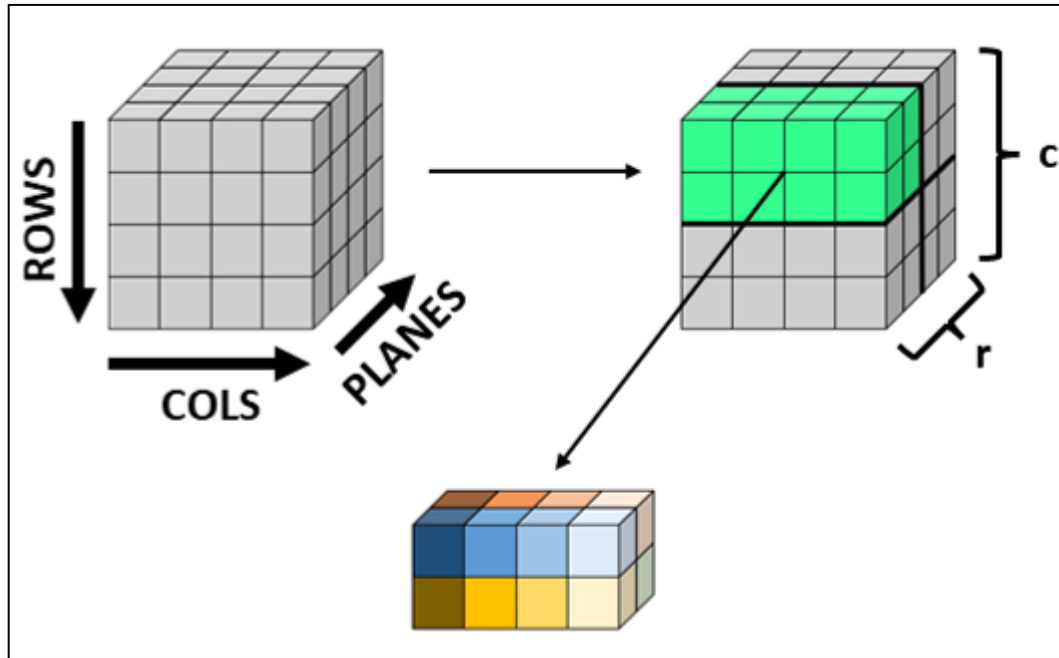
- 3D-FFT is a workhorse kernel:
 - Computationally heavy 1D-FFT phases
 - Memory re-sorting phases
 - All2All MPI communication phases
- Used in applications: HACC, GESTS, QMCPACK, etc.
- Memory re-sorting phases:
 - “S1CF” – store_1st_colwise_forward
 - “S2CF” – store_2nd_colwise_forward
 - “S1PF” – store_1st_planewise_forward
 - “S2PF” – store_2nd_planewise_forward

Case Study: 3D-FFT

- 3D-FFT is a workhorse kernel:
 - Computationally heavy 1D-FFT phases
 - Memory re-sorting phases
 - All2All MPI communication phases
- Used in applications: HACC, GESTS, QMCPACK, etc.
- Memory re-sorting phases:
 - **“S1CF”** – store_1st_colwise_forward
 - **“S2CF”** – store_2nd_colwise_forward
 - **“S1PF”** – store_1st_planewise_forward
 - **“S2PF”** – store_2nd_planewise_forward

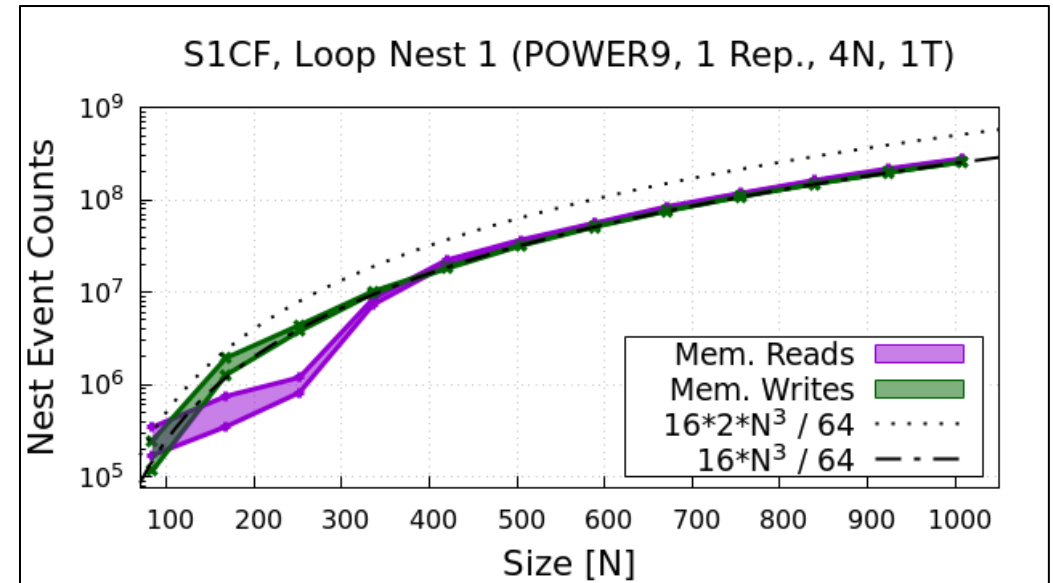
Domain Decomp. & Re-sorting Visualized

Domain Decomposition



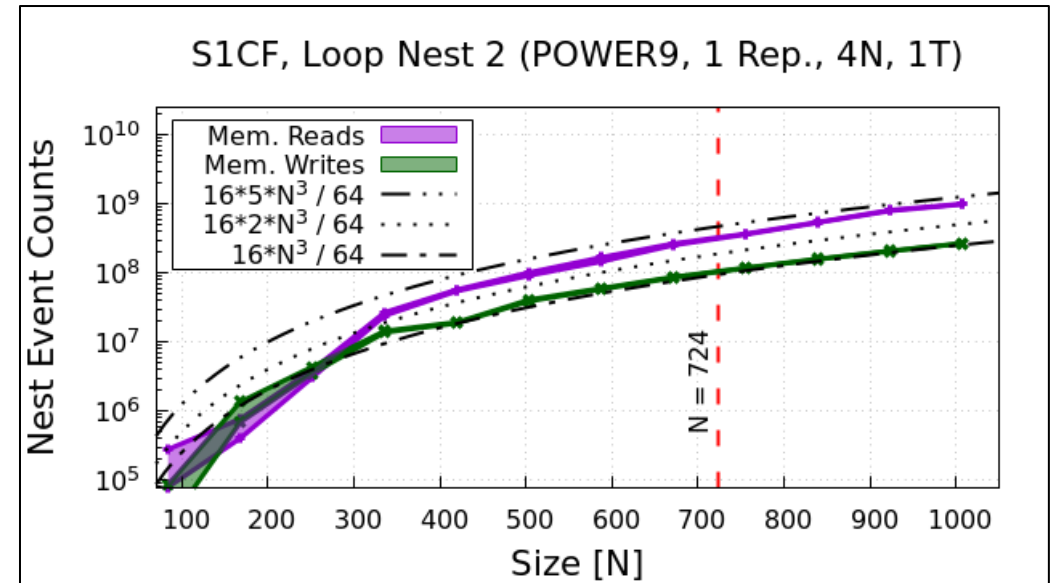
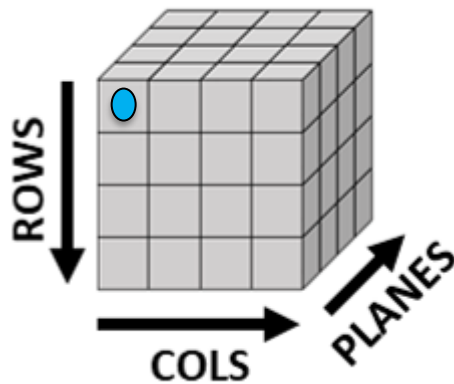
S1CF: Loop Nest 1

```
#pragma omp parallel for schedule(static)
for ( plane = 0; plane < PLANES; plane++ ) {
  for ( row = 0; row < ROWS; row++ ) {
    for ( col = 0; col < COLS; col++ ) {
      tmp[plane][row][col]
      = in[plane*ROWS*COLS + row*COLS + col];
    }
  }
}
```



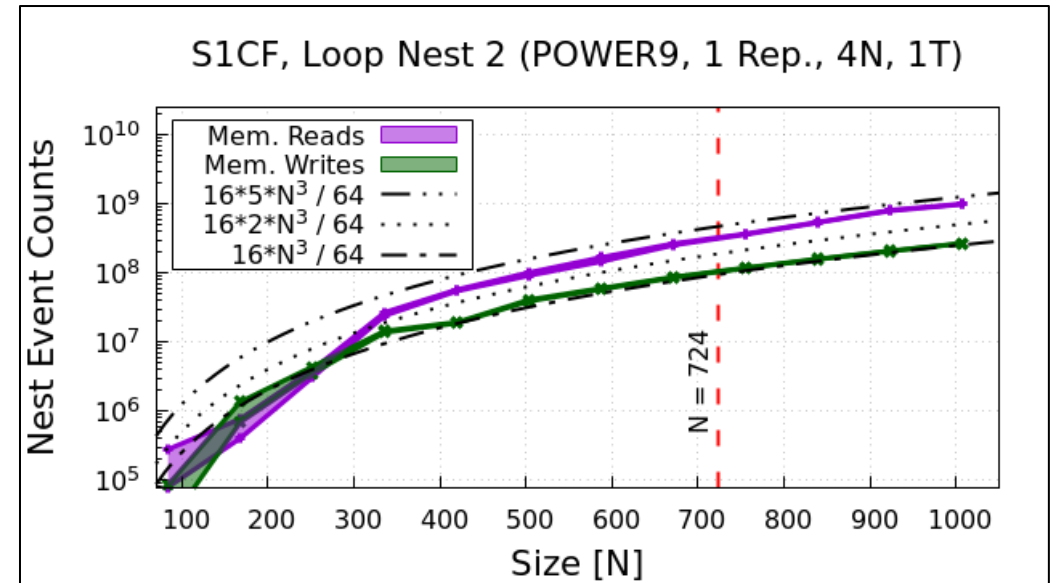
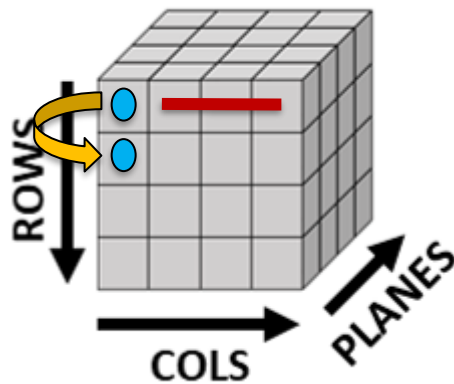
S1CF: Loop Nest 2

```
#pragma omp parallel for schedule(static)
for ( col = 0; col < COLS; col++ ) {
  for ( plane = 0; plane < PLANES; plane++ ) {
    for ( row = 0; row < ROWS; row++ ) {
      out[col*PLANES*ROWS + plane*ROWS + row]
      = tmp[plane][row][col];
    }
  }
}
```



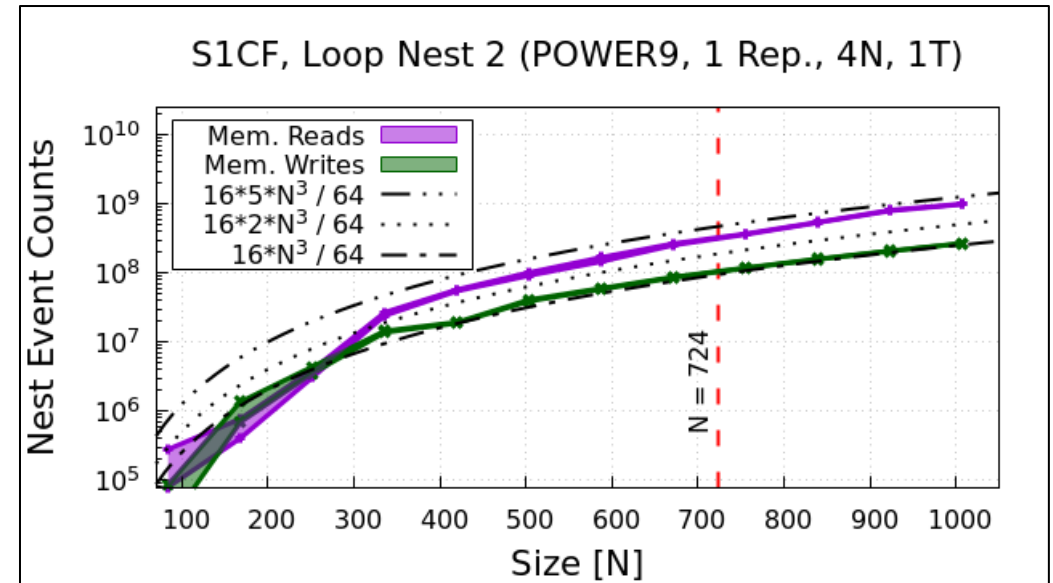
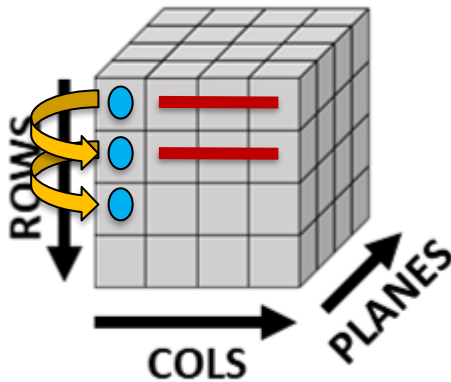
S1CF: Loop Nest 2

```
#pragma omp parallel for schedule(static)
for ( col = 0; col < COLS; col++ ) {
  for ( plane = 0; plane < PLANES; plane++ ) {
    for ( row = 0; row < ROWS; row++ ) {
      out[col*PLANES*ROWS + plane*ROWS + row]
      = tmp[plane][row][col];
    }
  }
}
```



S1CF: Loop Nest 2

```
#pragma omp parallel for schedule(static)
for ( col = 0; col < COLS; col++ ) {
  for ( plane = 0; plane < PLANES; plane++ ) {
    for ( row = 0; row < ROWS; row++ ) {
      out[col*PLANES*ROWS + plane*ROWS + row]
      = tmp[plane][row][col];
    }
  }
}
```



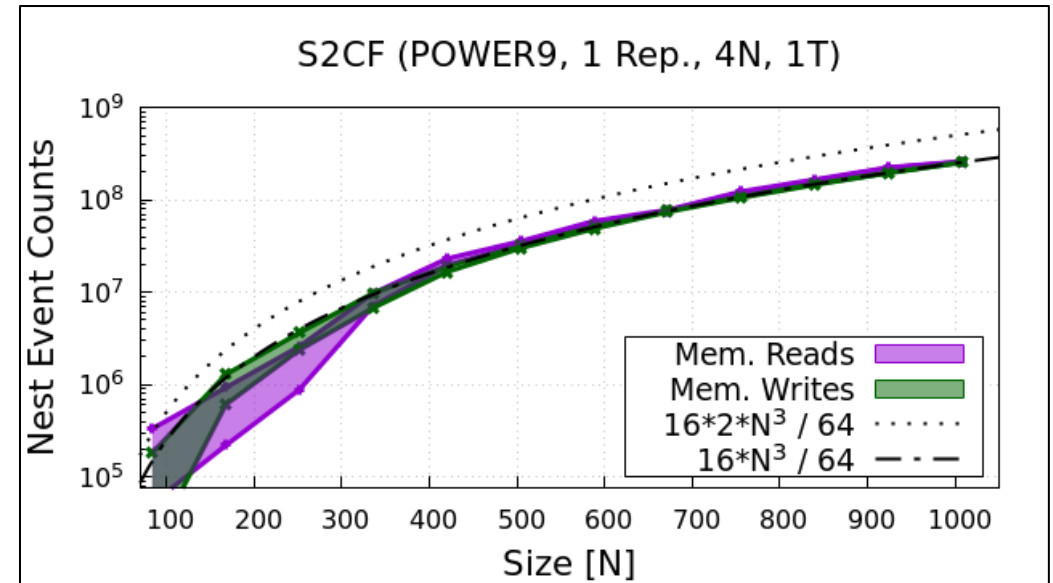
S2CF

```
X = COLS/r; Y = r;

#pragma omp parallel for schedule(static)
for ( plane = 0; plane < PLANES; plane++ ) {
    for ( x = 0; x < X; x++ ) {
        for ( y = 0; y < Y; y++ ) {
            for ( row = 0; row < ROWS; row++ ) {

                idx_out = plane*X*Y*ROWS + x*Y*ROWS \
                    + y*ROWS + row;
                idx_in = y*PLANES*X*ROWS + plane*X*ROWS \
                    + x*ROWS + row;

                out[idx_out] = in[idx_in];
            }
        }
    }
}
```



Assembly & Avoiding Cache on PowerPC

PowerPC

The PowerPC provides the following data prefetch instructions [14]:

| | |
|---------|----------------------------------|
| dcbt | Data Cache Block Touch |
| dcbstst | Data Cache Block Touch for Store |

There are no alignment restrictions on the address of the data to prefetch.

GCC: <https://gcc.gnu.org/projects/prefetch.html>

Assembly & Avoiding Cache on PowerPC

PowerPC

The PowerPC provides the following data prefetch instructions [14]:

| | |
|--------|----------------------------------|
| dcbt | Data Cache Block Touch |
| dcbtst | Data Cache Block Touch for Store |

There are no alignment restrictions on the address of the data to prefetch.

GCC: <https://gcc.gnu.org/projects/prefetch.html>

Assembly & Avoiding Cache on PowerPC

PowerPC

The PowerPC provides the following data prefetch instructions [14]:

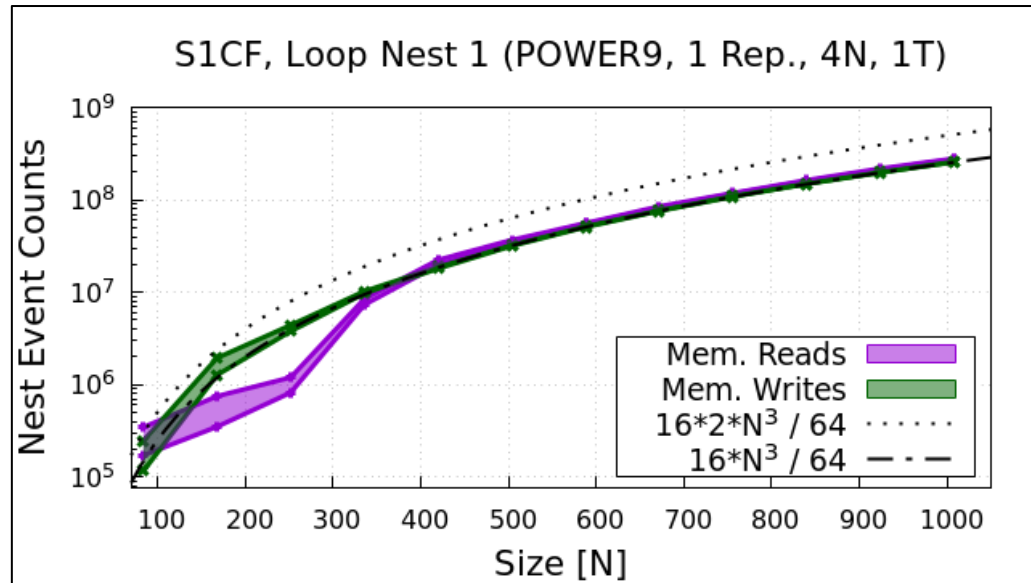
| | |
|---------|----------------------------------|
| dcbt | Data Cache Block Touch |
| dcbstst | Data Cache Block Touch for Store |

There are no alignment restrictions on the address of the data to prefetch.

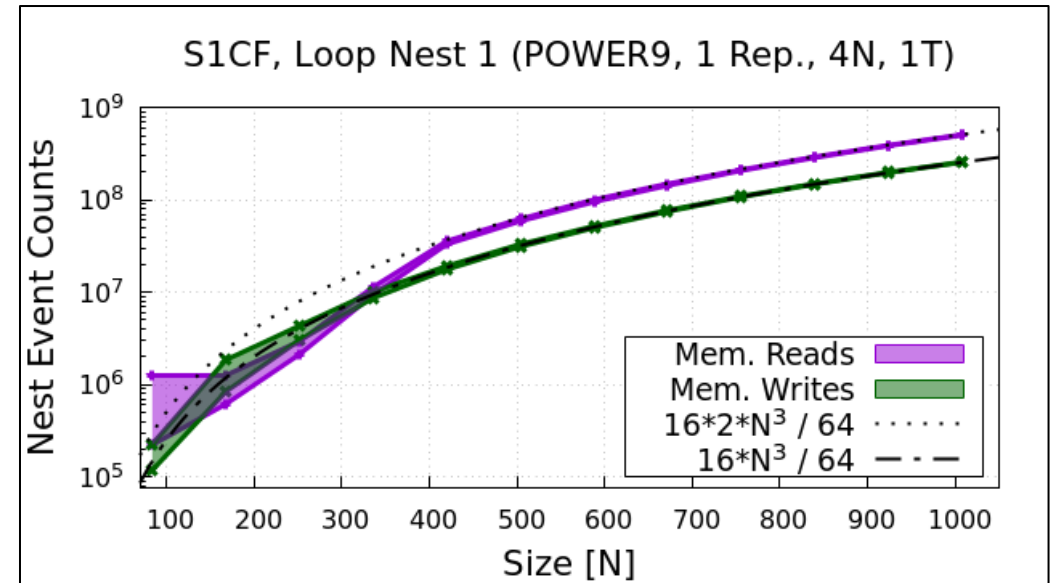
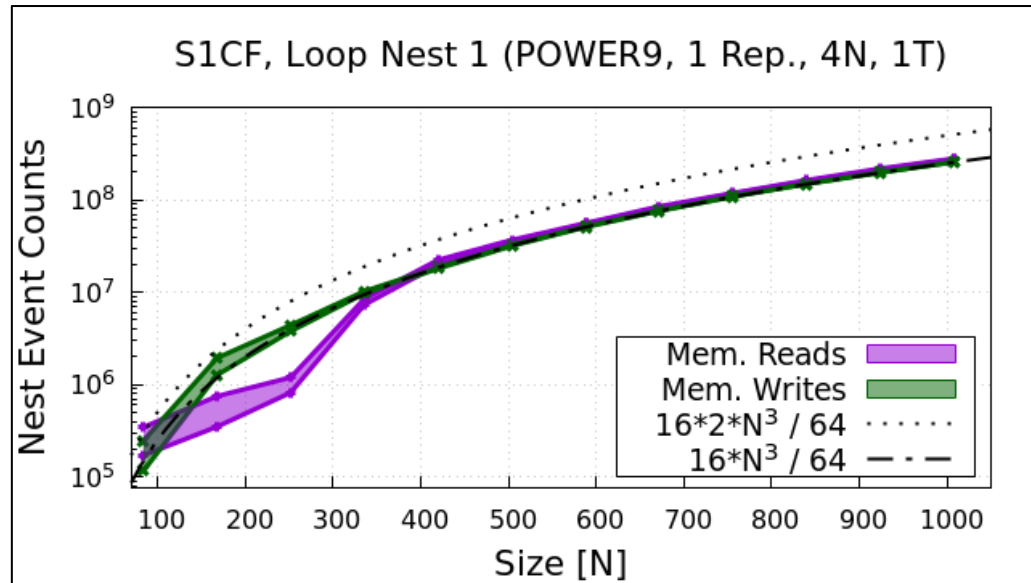
GCC: <https://gcc.gnu.org/projects/prefetch.html>

-fprefetch-loop-arrays

A Closer Look at S1CF: Loop Nest 1



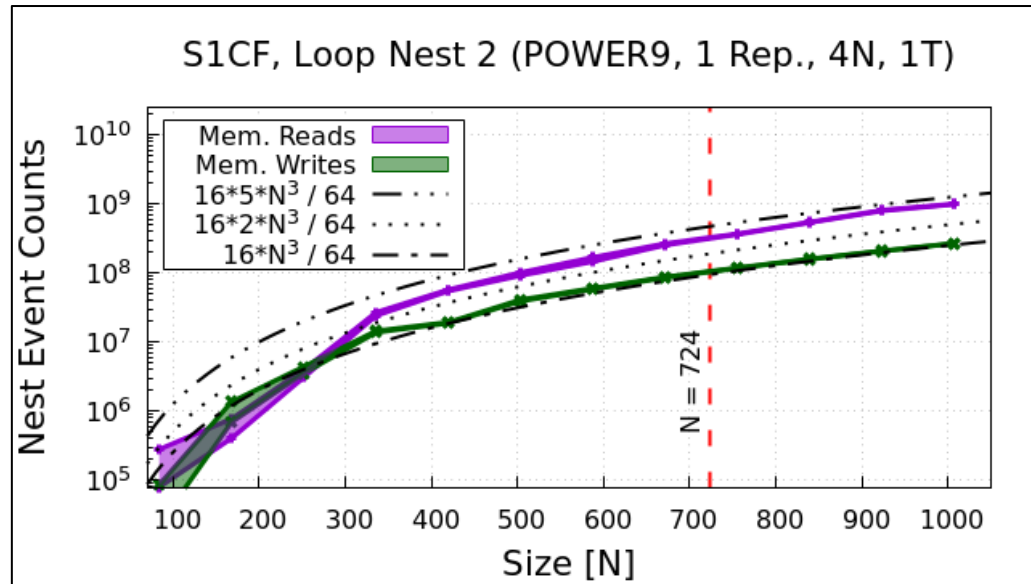
A Closer Look at S1CF: Loop Nest 1



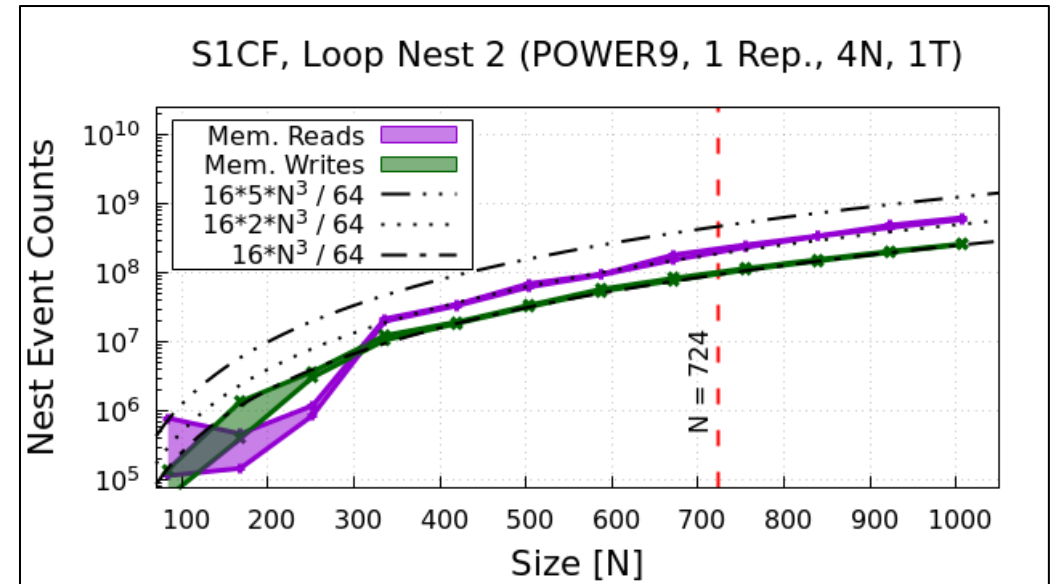
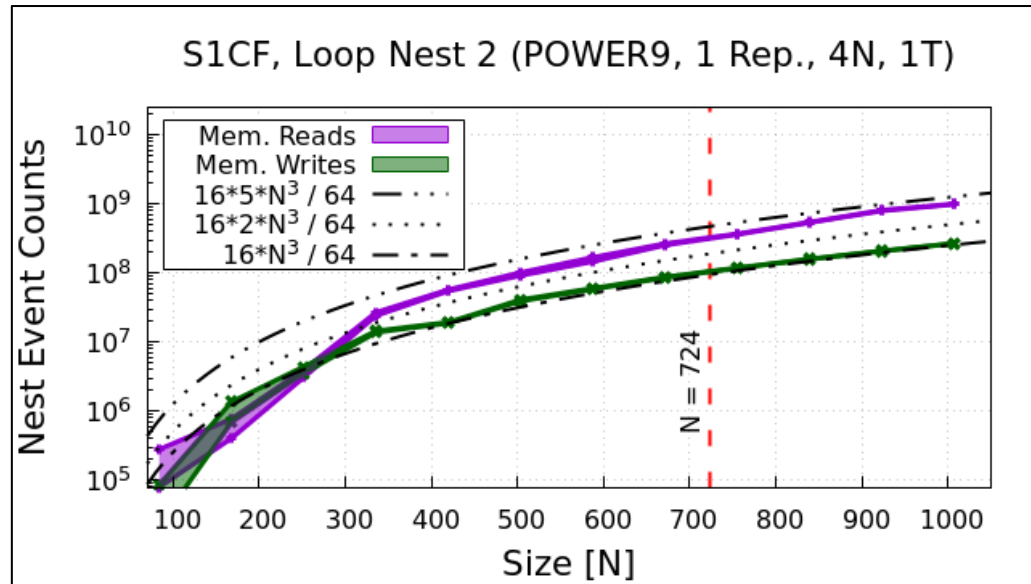
-fprefetch-loop-arrays

```
404: 2c 4a 00 7c dcbt 0,r9  
408: ec 41 00 7c dcbtst 0,r8
```

A Closer Look at S1CF: Loop Nest 2



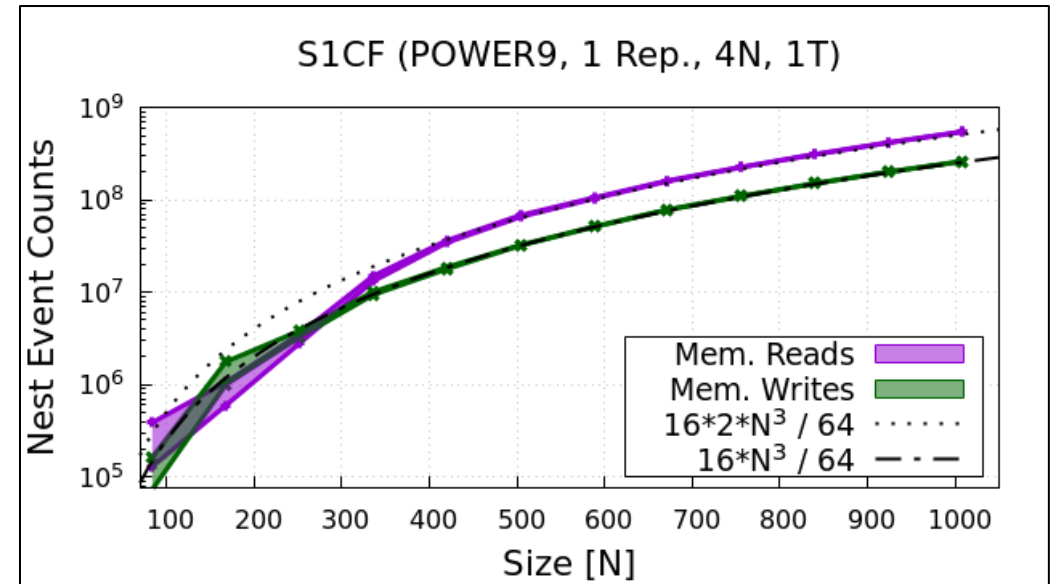
A Closer Look at S1CF: Loop Nest 2



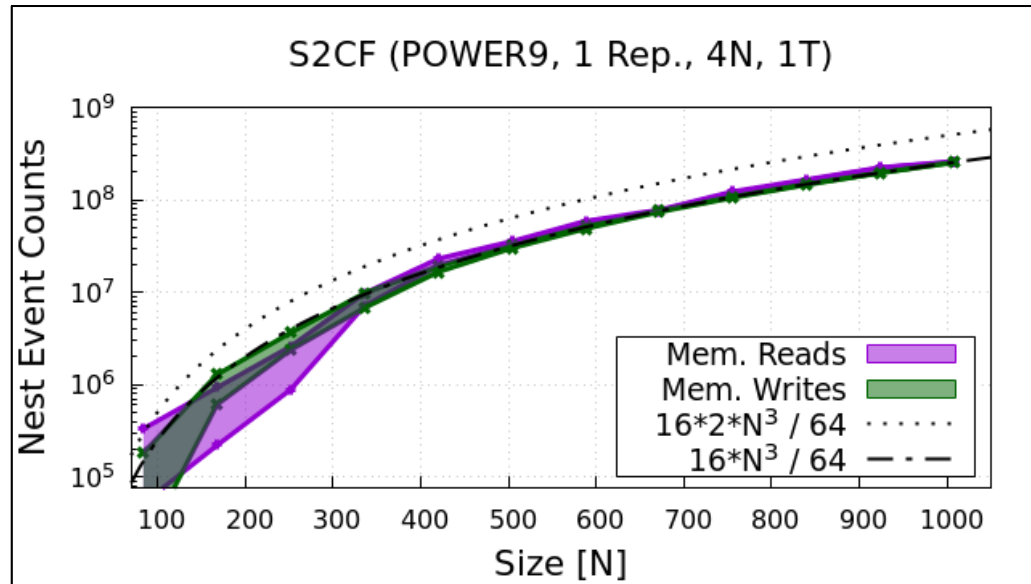
-fprefetch-loop-arrays

S1CF: Fused Loop Nests

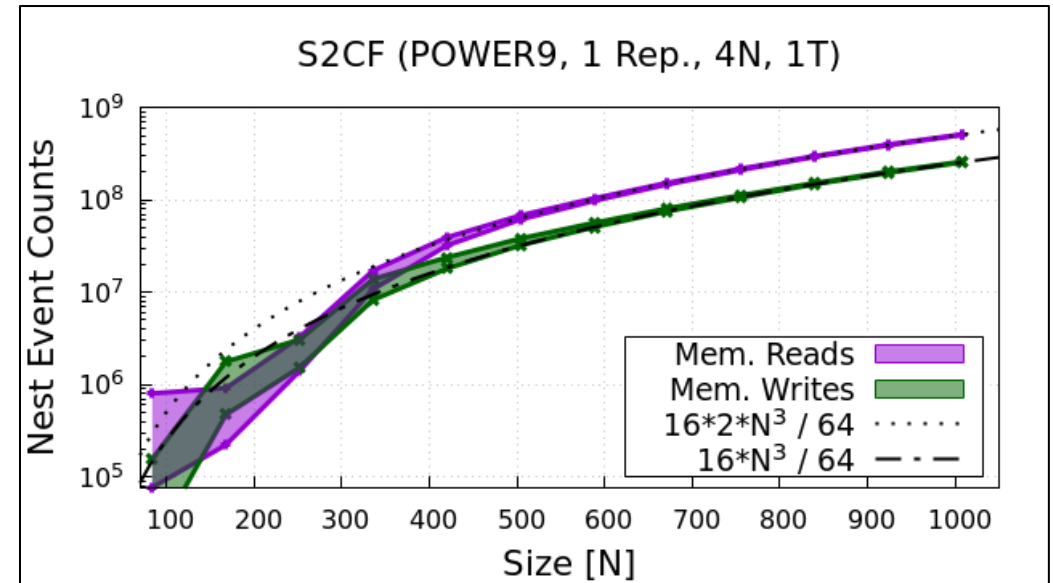
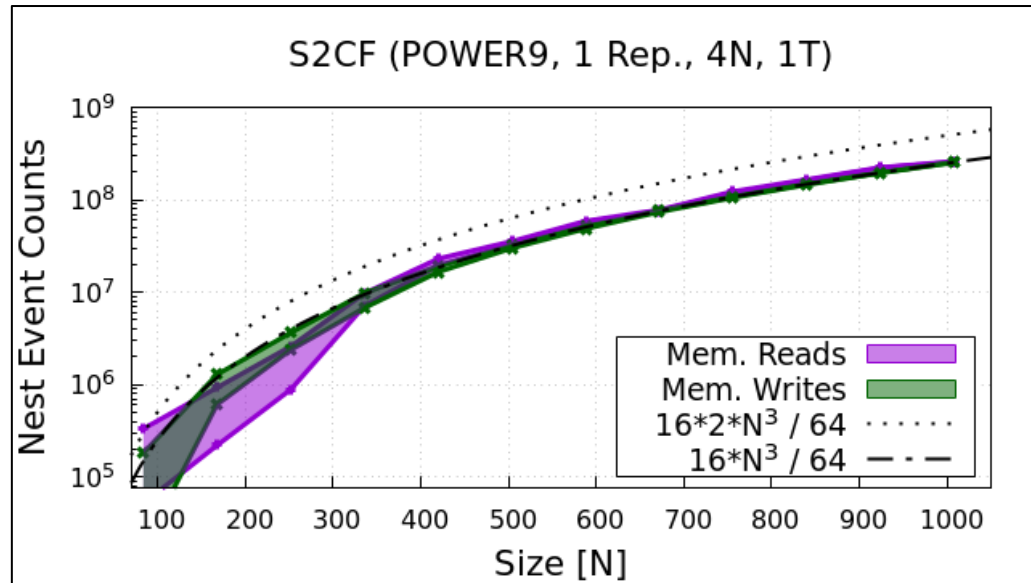
```
#pragma omp parallel for schedule(static)
for ( plane = 0; plane < PLANES; plane++ ) {
  for ( row = 0; row < ROWS; row++ ) {
    for ( col = 0; col < COLS; col++ ) {
      out[col*PLANES*ROWS + plane*ROWS + row]
      = in[plane*ROWS*COLS + row*COLS + col];
    }
  }
}
```



A Closer Look at S2CF

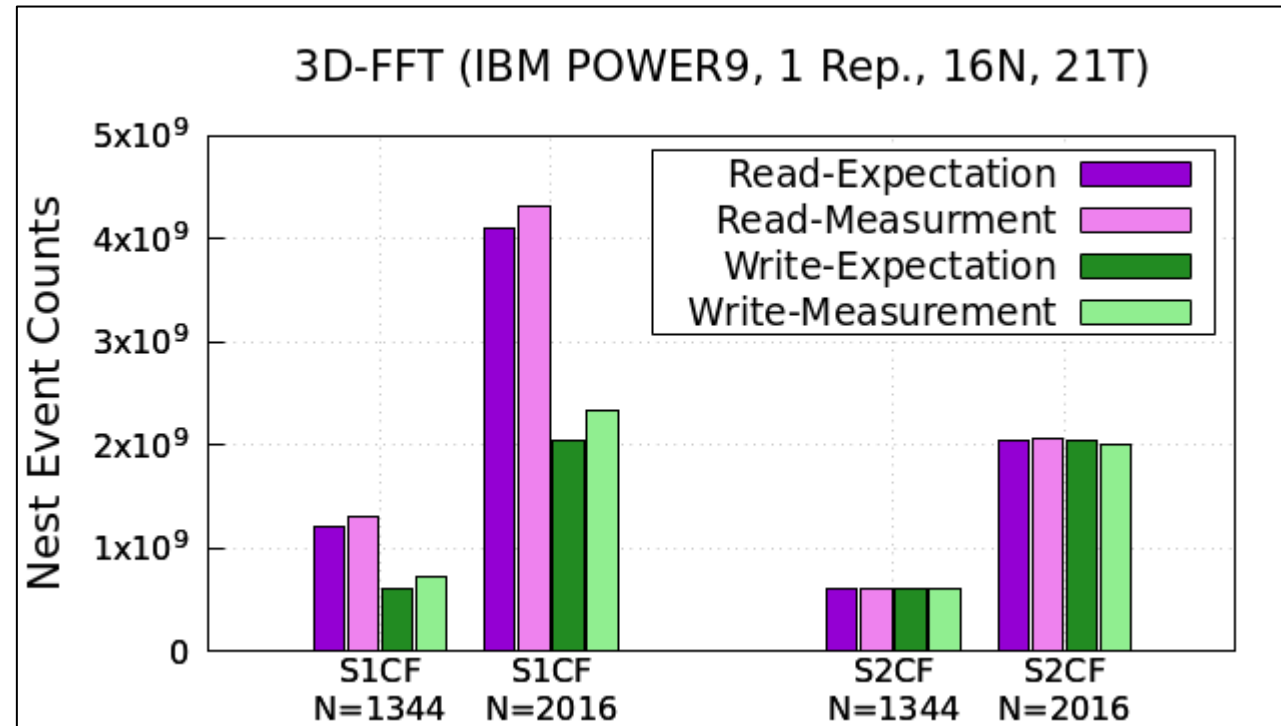


A Closer Look at S2CF



-fprefetch-loop-arrays

Examples of Larger Jobs

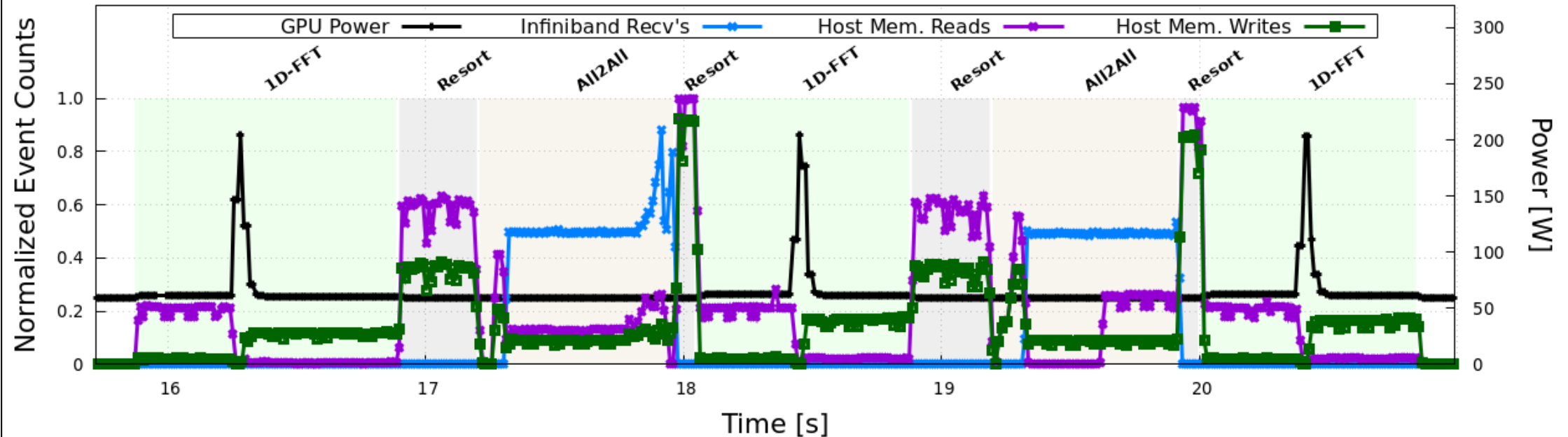


Validation Experiments

1. Common BLAS Kernels
2. 3D-FFT Case Study
3. Profiling Full Applications with Multiple Components of PAPI

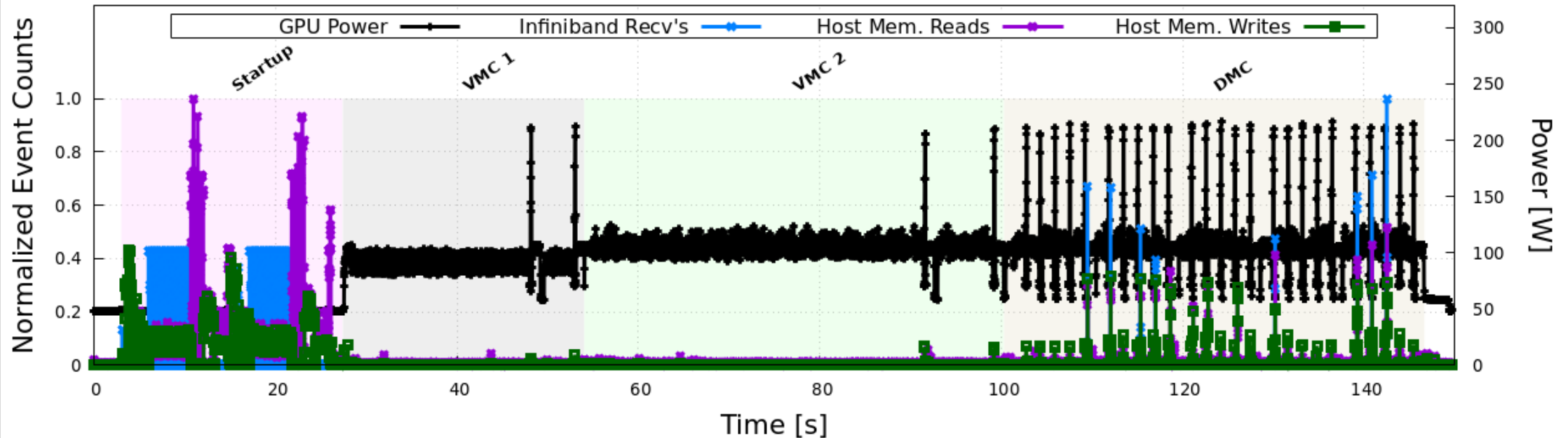
Profiling the 3D-FFT on Summit

3D-FFT: Memory, Power, and Network Traffic Profile
(N=2688, 64 MPI Ranks, IBM POWER9, NVIDIA Tesla V100)



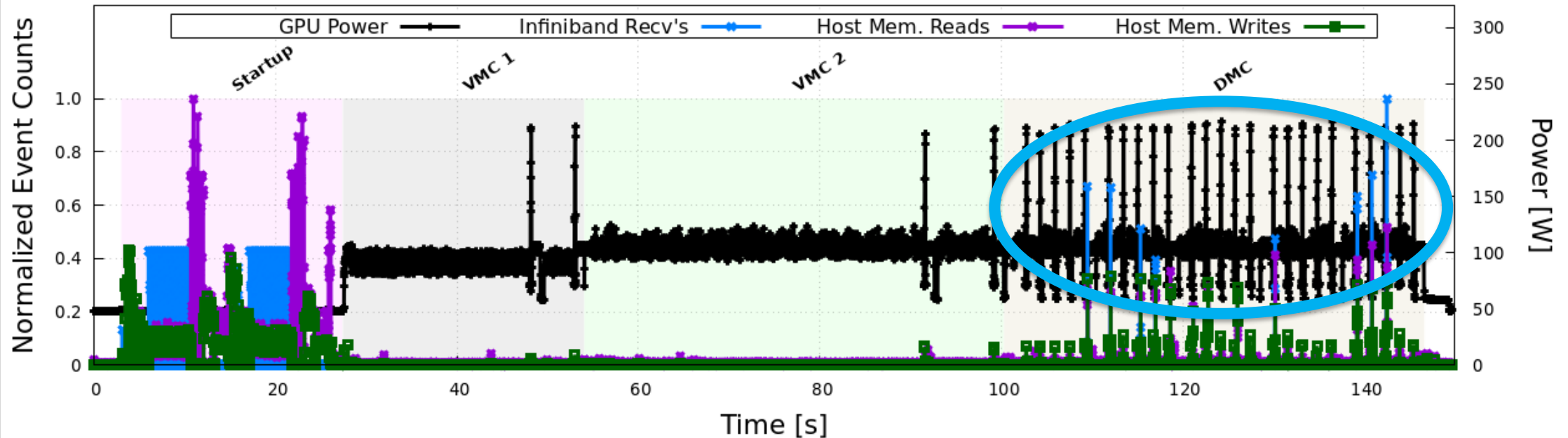
Profiling QMCPACK on Summit

QMCPACK: Memory, Power, and Network Traffic Profile
(NiO-fcc-S32-dmc, 32 MPI Ranks, IBM POWER9, NVIDIA Tesla V100)



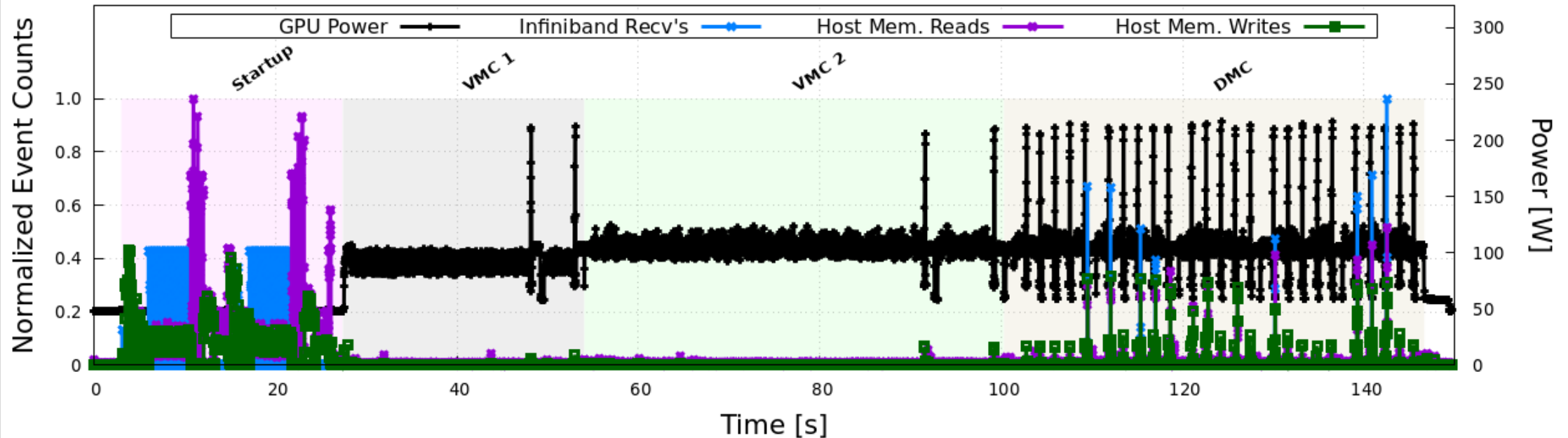
Profiling QMCPACK on Summit

QMCPACK: Memory, Power, and Network Traffic Profile
(NiO-fcc-S32-dmc, 32 MPI Ranks, IBM POWER9, NVIDIA Tesla V100)



Profiling QMCPACK on Summit

QMCPACK: Memory, Power, and Network Traffic Profile
(NiO-fcc-S32-dmc, 32 MPI Ranks, IBM POWER9, NVIDIA Tesla V100)



Conclusions

- Measurements of memory traffic for tiny data are unreliable.

Conclusions

- Measurements of memory traffic for tiny data are unreliable.
- Memory traffic measurements are sensitive to micro-architectural details
 - e.g. localized L3 slices, cache-avoidant writes

Conclusions

- Measurements of memory traffic for tiny data are unreliable.
- Memory traffic measurements are sensitive to micro-architectural details
 - e.g. localized L3 slices, cache-avoidant writes
- PAPI allows users to generate complete profiles for the *entire* system:
 - GPU Power
 - Memory Traffic
 - Infiniband Network Traffic
 - CPU, GPU, on- and off-chip memory, IO, networks, and more!

Future Work

- Focusing on other BLAS operations.
- Upcoming IBM architectures, such as POWER10.
- Categories of next hardware other than solely memory traffic.

Acknowledgements

This research was supported in part by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration; and by the National Science Foundation under award No. 1900888 “ANACIN-X.”