Optimization Strategies for WRF Single-Moment 6-Class Microphysics Scheme (WSM6) on Intel Microarchitectures.

Presented by:
T.A.J.Ouermi, Aaron Knoll, Mike Kirby, Martin Berzins

User Productivity Enhancement, Technology Transfer, and Training (PETTT)

Weather Physics Optimization

Outline

- Motivation
- NEPTUNE and WSM6
- KNL Architecture
- Methodology
- Standalone experiments
- WSM6 results
- Discussion Conclusion

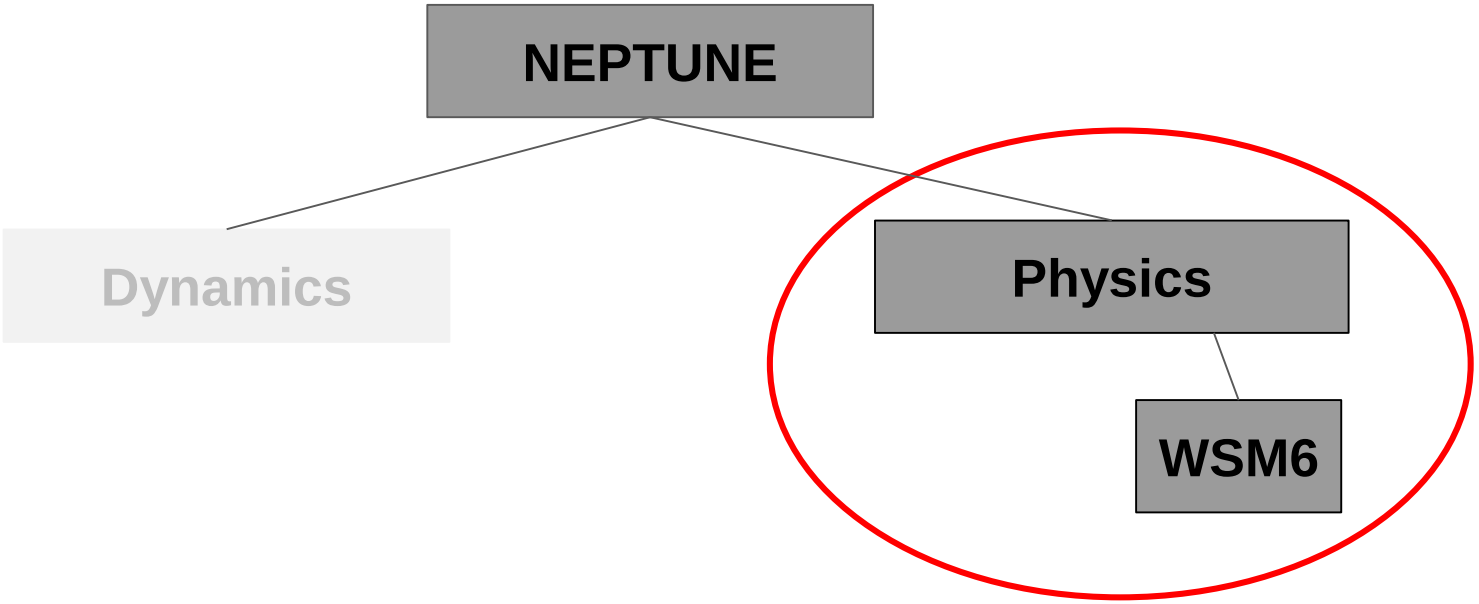
Motivation

- Faster weather physics for operational Navy Environmental Prediction sysTem Utilizing the NUMA corE (NEPTUNE)

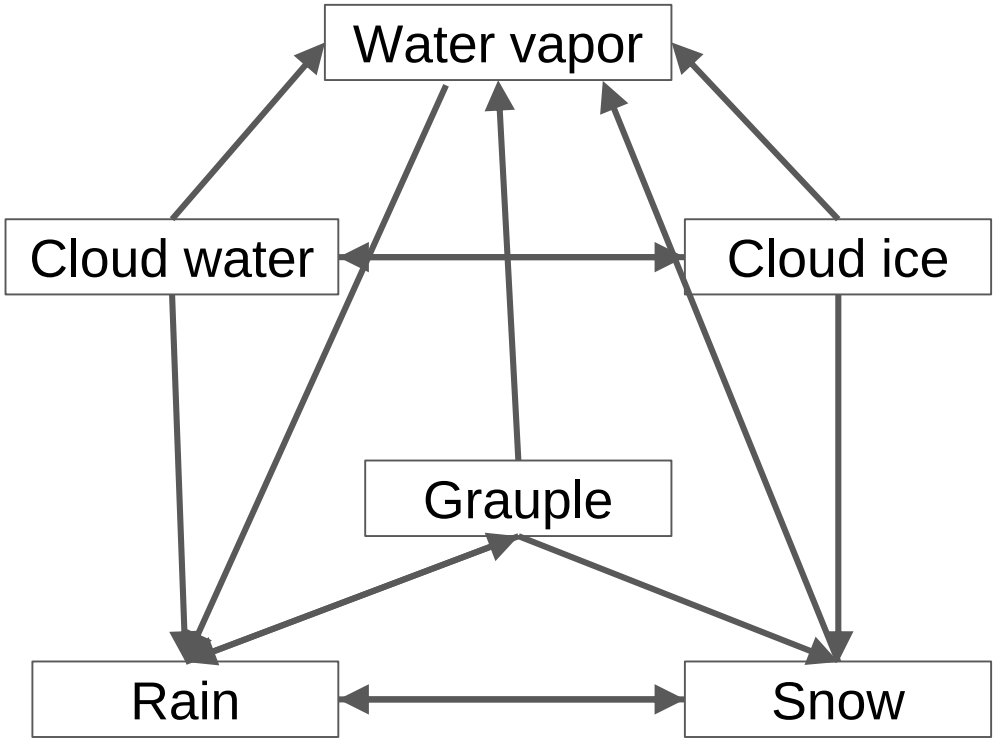
- Target architectures: MIC
  - Intel Knights Landing (KNL), Knights Hill(KNH), …

- Portability with OpenMP

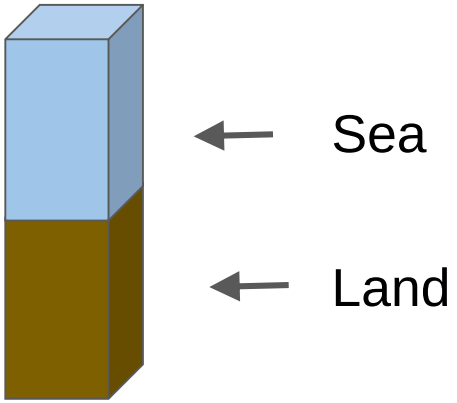
NEPTUNE



Physics Optimization Challenges



WRF single-moment 6-class Microphysics Scheme (WSM6)

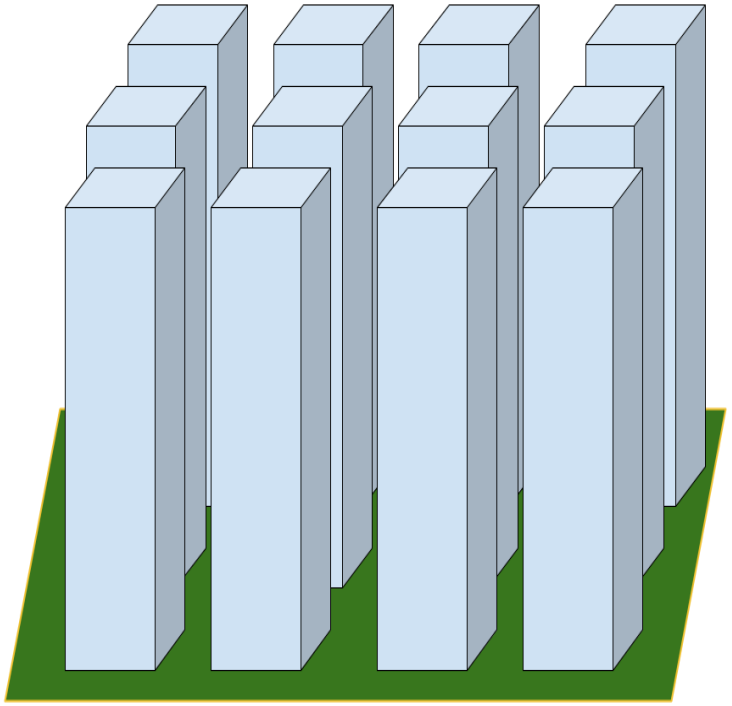


Grauple Particles

Vertical Physics Advantage

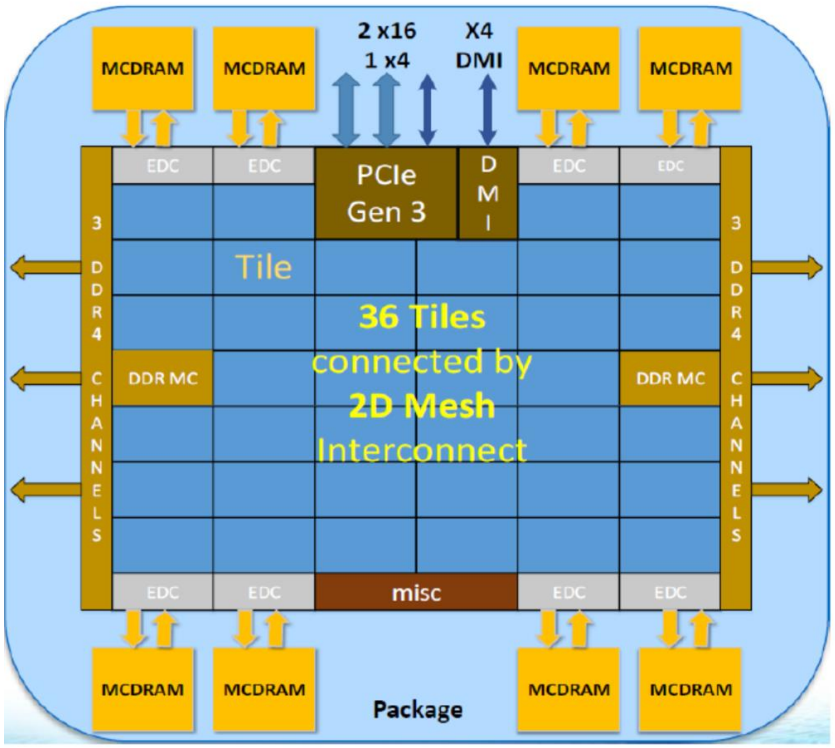
- Dependencies within columns

- No dependencies between columns



Vertical Physics representation

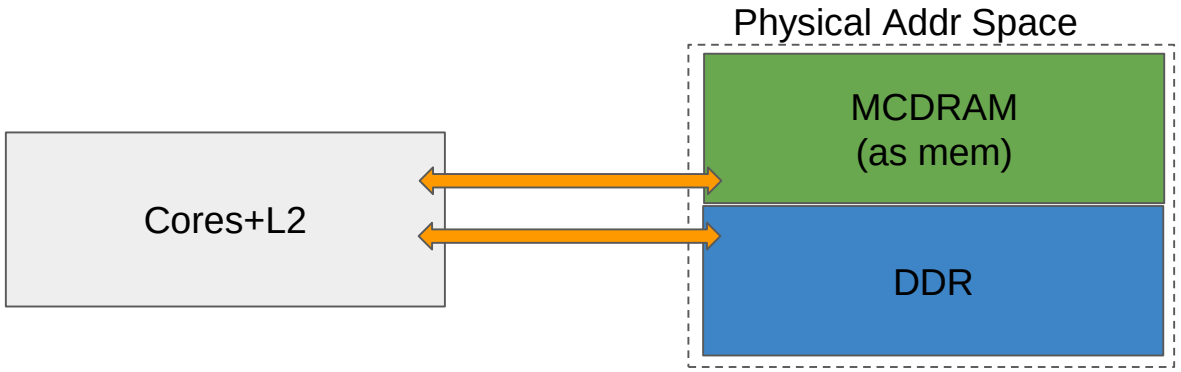
KNL Architecture



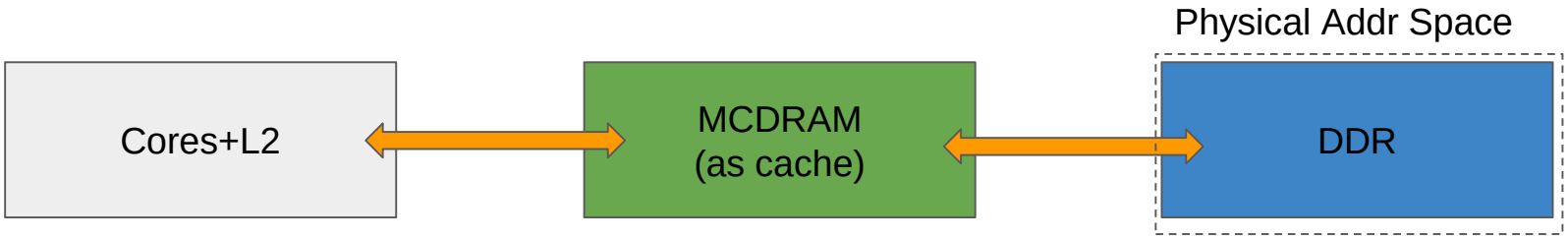
Tile

- MCDRAM:16GB, High BW
- Peak 3 teraflops double precision
- 512 bit vectors

KNL Configurations

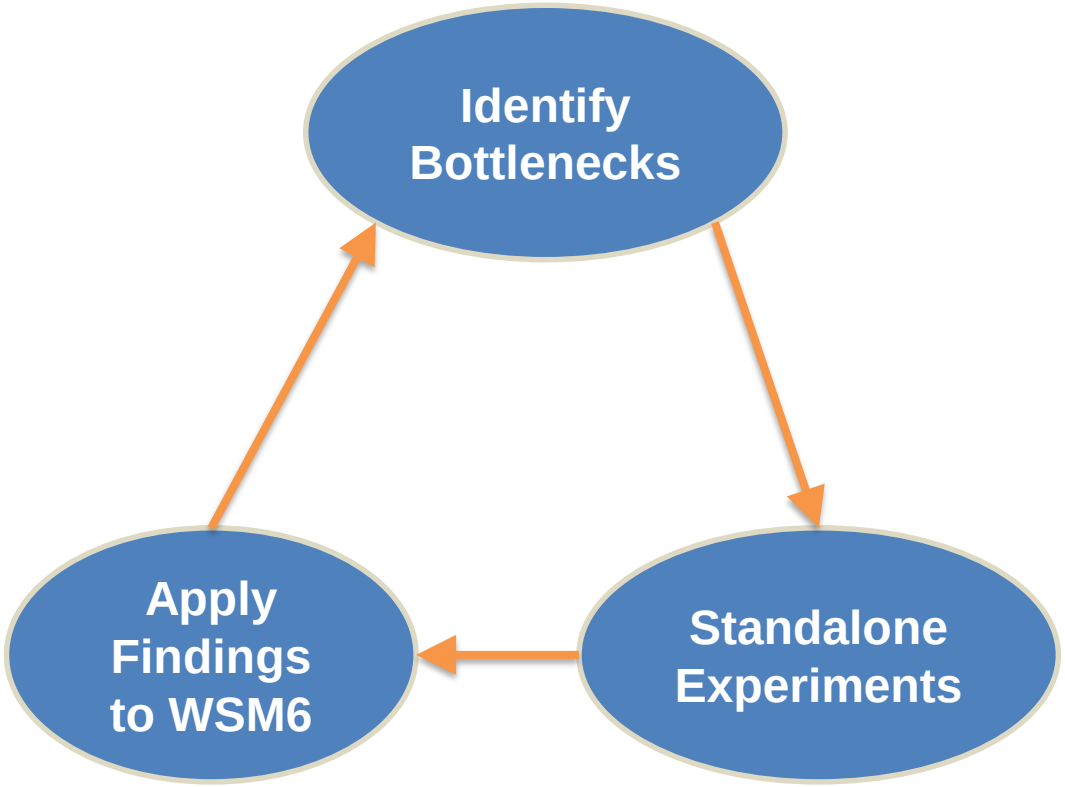


Flat mode



Cache mode

Methodology

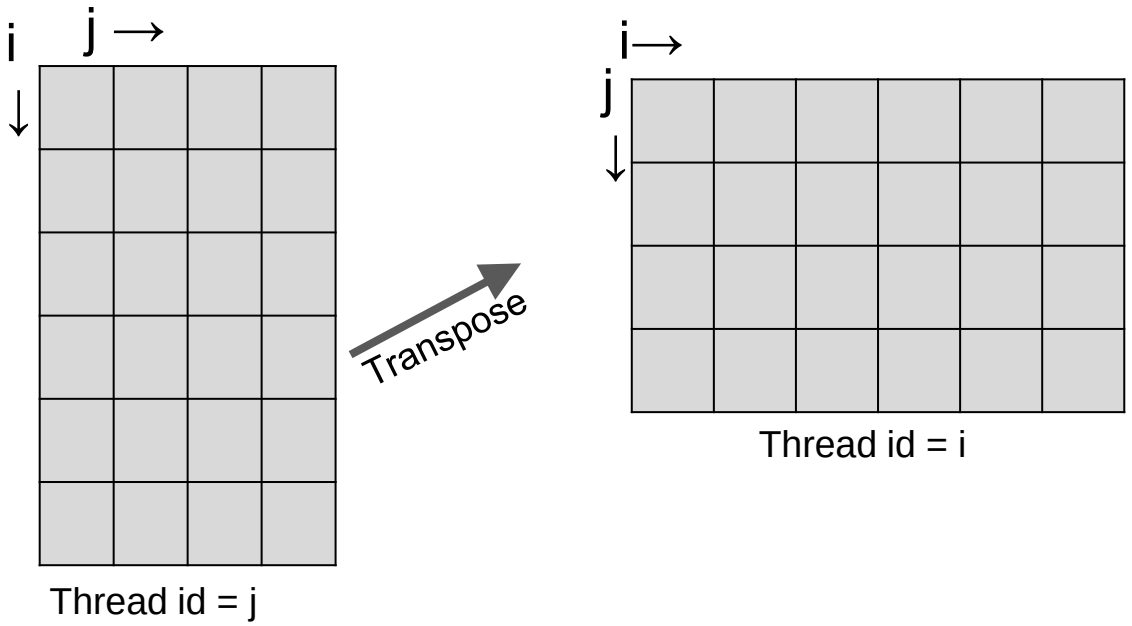


- Identify bottlenecks
  - -Wall Clock, Vtune
  - -Advisor, optrpt
- Standalone experiments
  - -Controlled environment
- Apply findings to WSM6
  - -Thread
  - -SIMD

Transpose

```
!$OMP DO
do j=1,km
do i=1,im
  a(i,k)=b(i,k)-c(i,k)
end do
end do
```

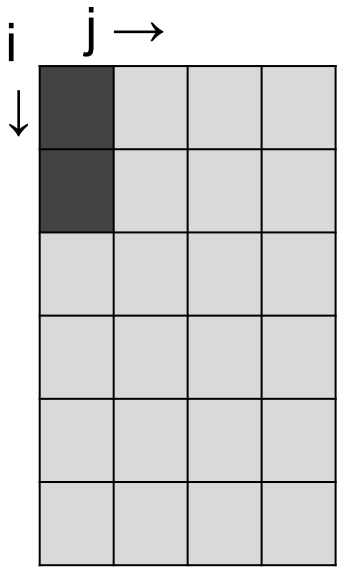
```
!$OMP DO
do i=1,im
do j=1,km
  a(k,i)=b(k,i)-c(k,i)
end do
end do
```



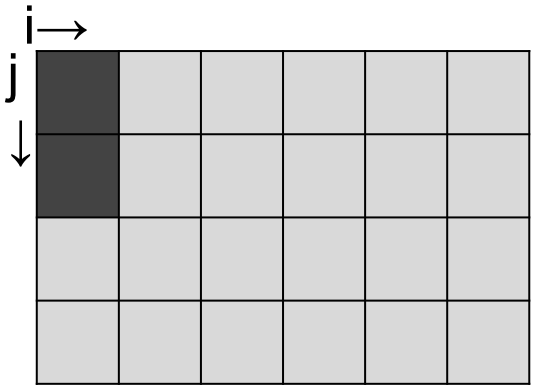
Vectorization

```
!$OMP DO
do j=1,km
!$OMP SIMD
do i=1,im
 a(i,k)=b(i,k)-c(i,k)
end do
end do
```

```
!$OMP DO
do i=1,im
!$OMP SIMD
do j=1,km
 a(k,i)=b(k,i)-c(k,i)
end do
end do
```

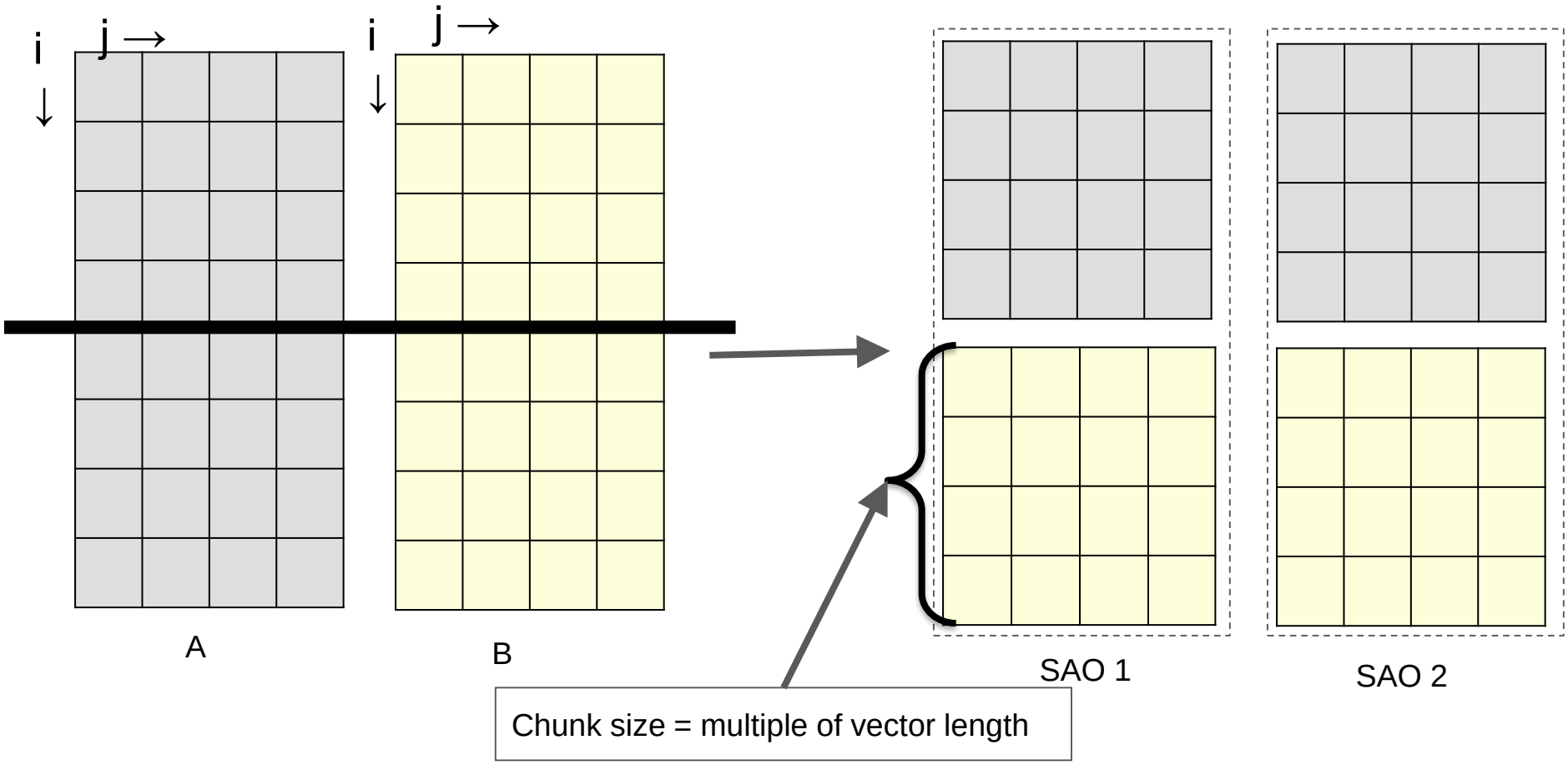


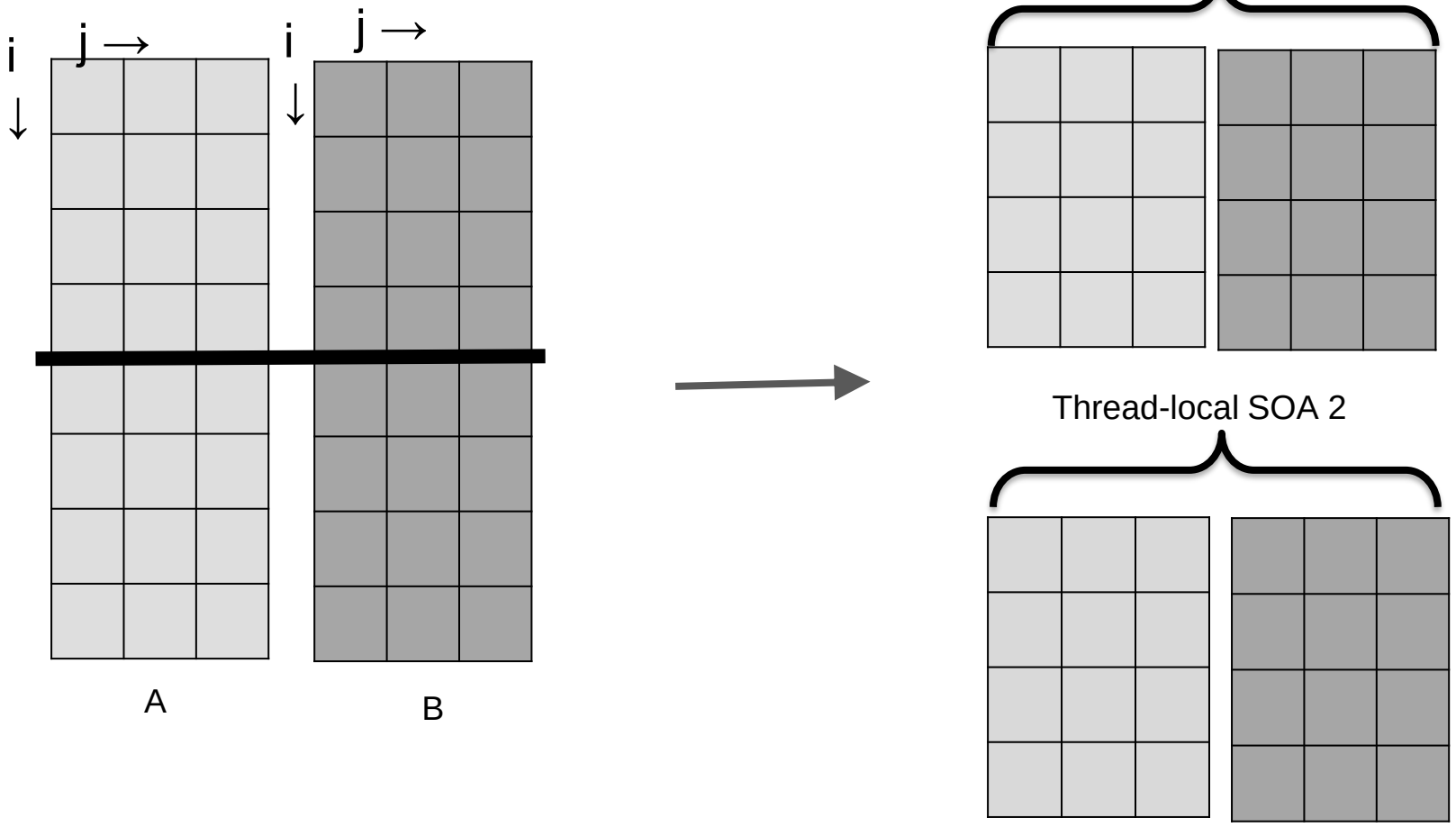
Thread id = j



Thread id = i

Structures of Arrays (SOA)




i
j →
i
j →
A
B
Thread-local SOA 1
Thread-local SOA 2

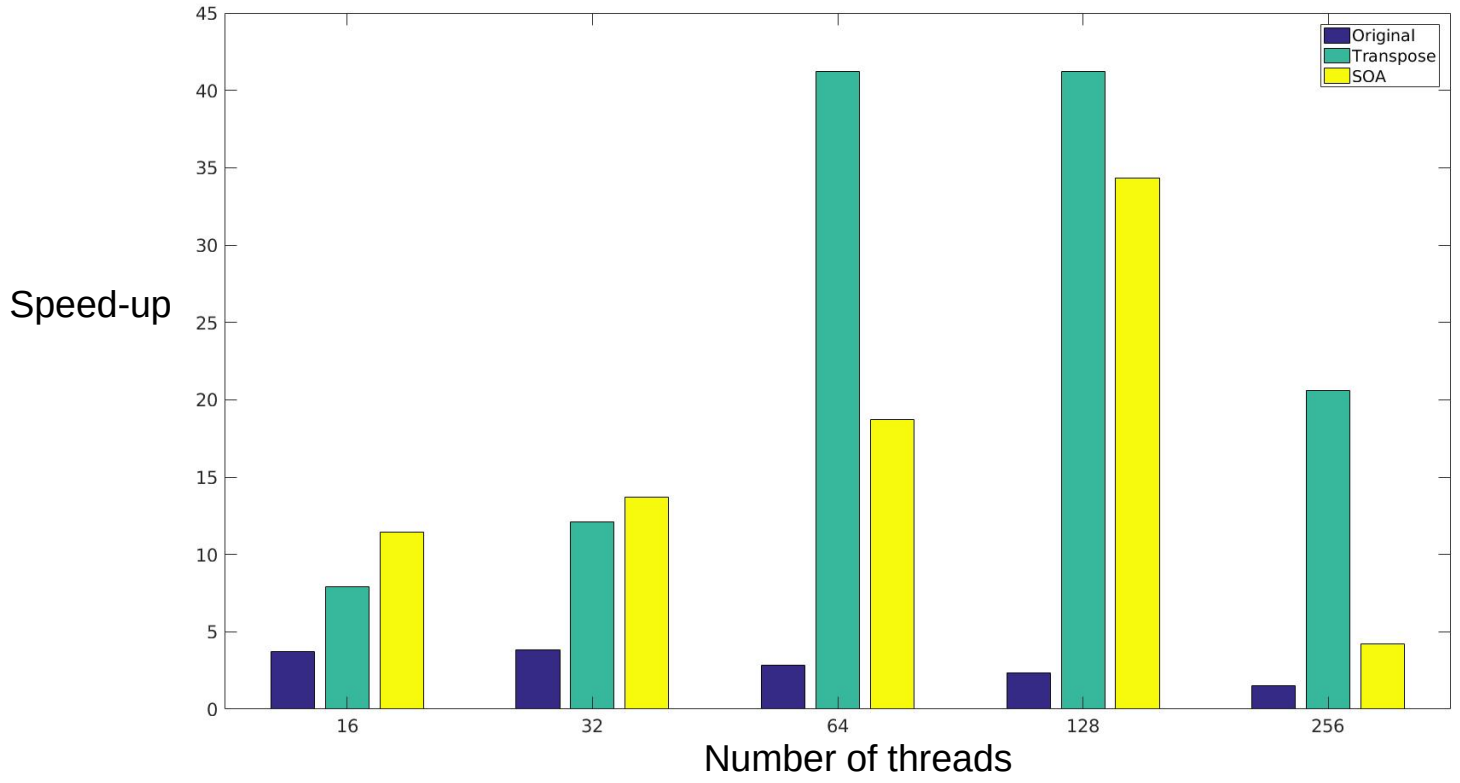
Complex Loop Parallelization

- No conditional 9.7x
- No function calls 30x
- Vectorization 41x

```
do k=kte,kts-1
do i=its,ite
 …
 if(t(i,k).gt.t0c)then
  …
  w(i,k) = venfac(p(i,k), t(i,k), den(i,k))
  if(qrs(i,k,2).gt.0)then
   …
   psmlt(i,k)=xka(t(i,k), den(i,k)…
  end if
  if(qrs(i,k,2).gt.0)then
  psmlg(i,k)=xka(t(i,k), den(i,k)…
  …
  end if
 end if
end do
end do
```

Loop 12 from WSM6

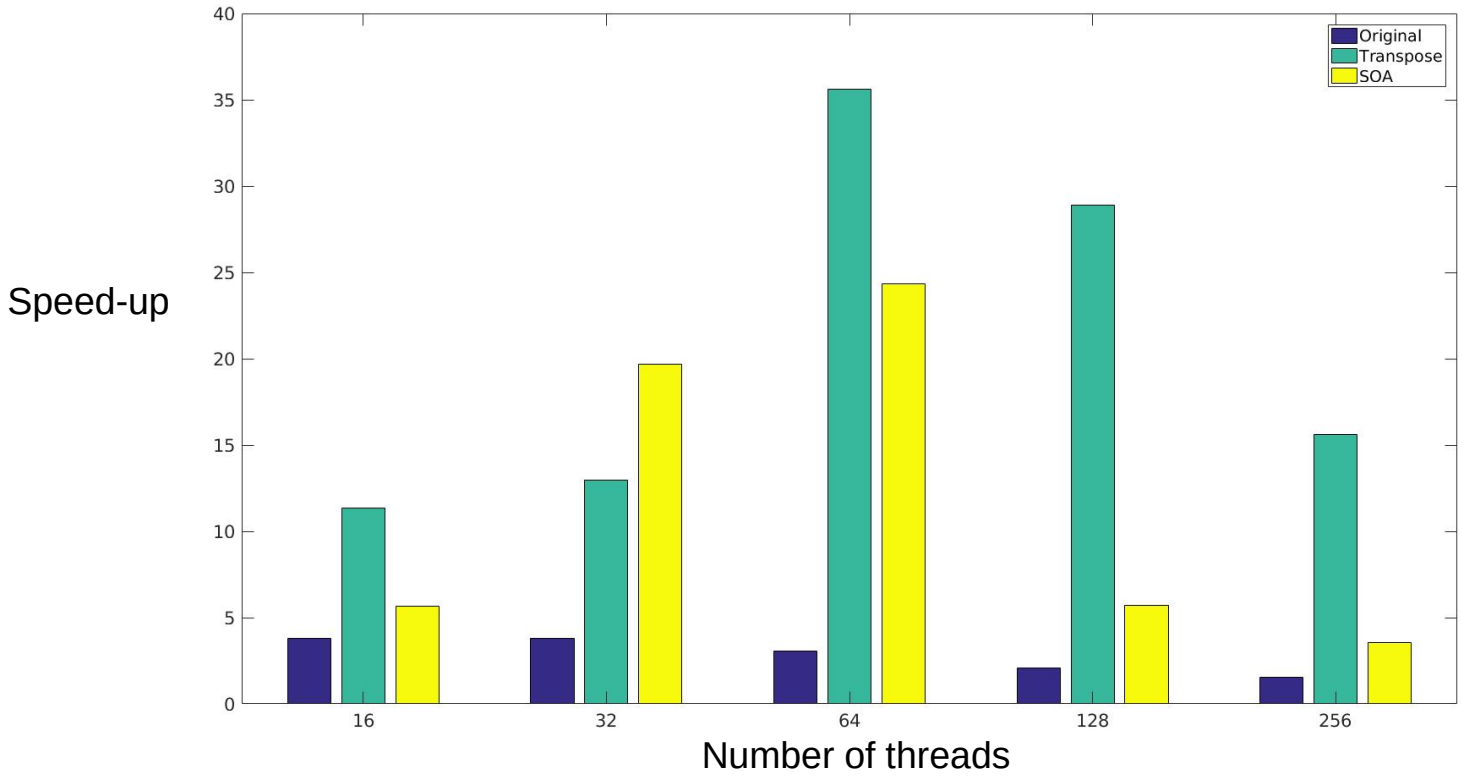
1D Arrays Experiments




```
!$OMP SIMD
do j=2,je-1
 a(j)=0.1+c(j)/d(j)
 b(j)=(0.2+c(j-1)-c(j))/(c(j)-c(j-1)+0.5)
end do
```

1D case

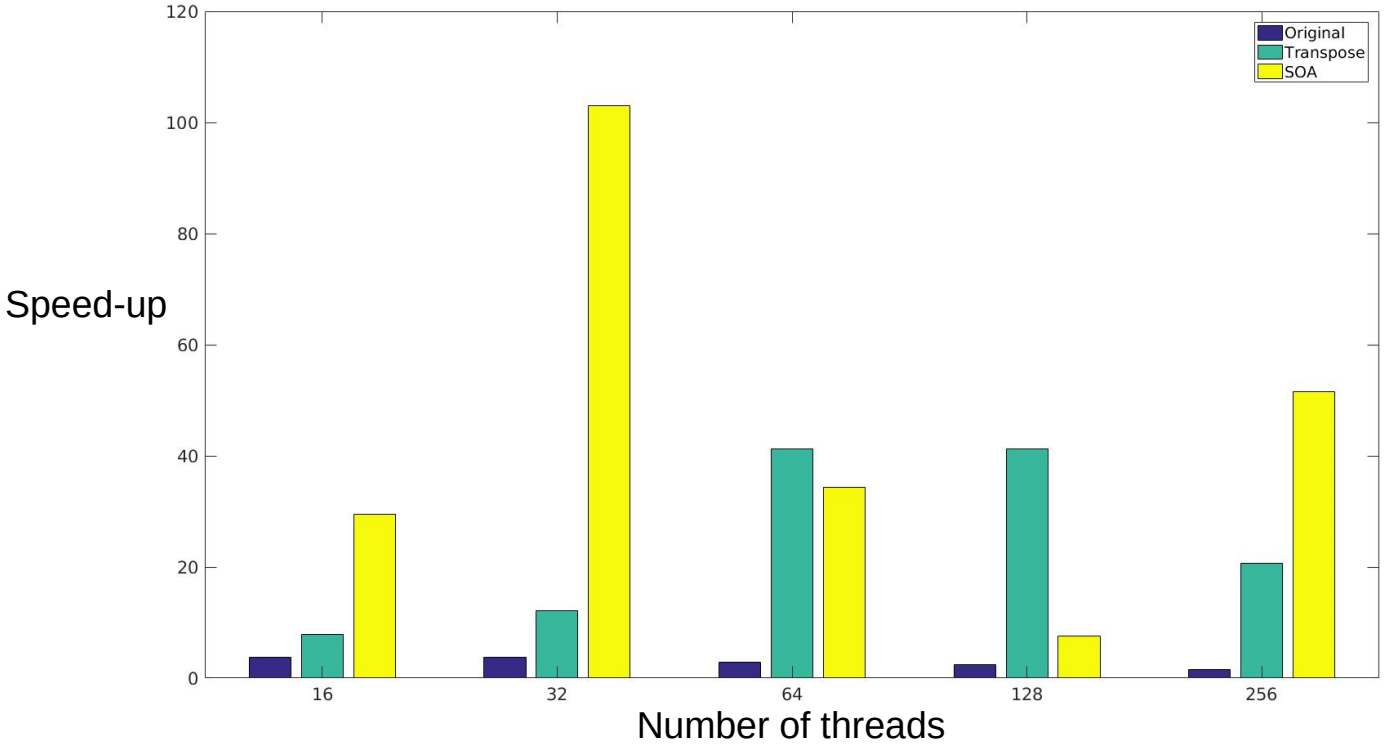
1D Arrays Experiments




```
!$OMP SIMD
do j=2,je-1
 a(j)=0.1+c(j)/d(j)
 b(j)=(0.2+c(j-1)-c(j))/(c(j)-c(j-1)+0.5)
end do
```

1D case with large array sizes

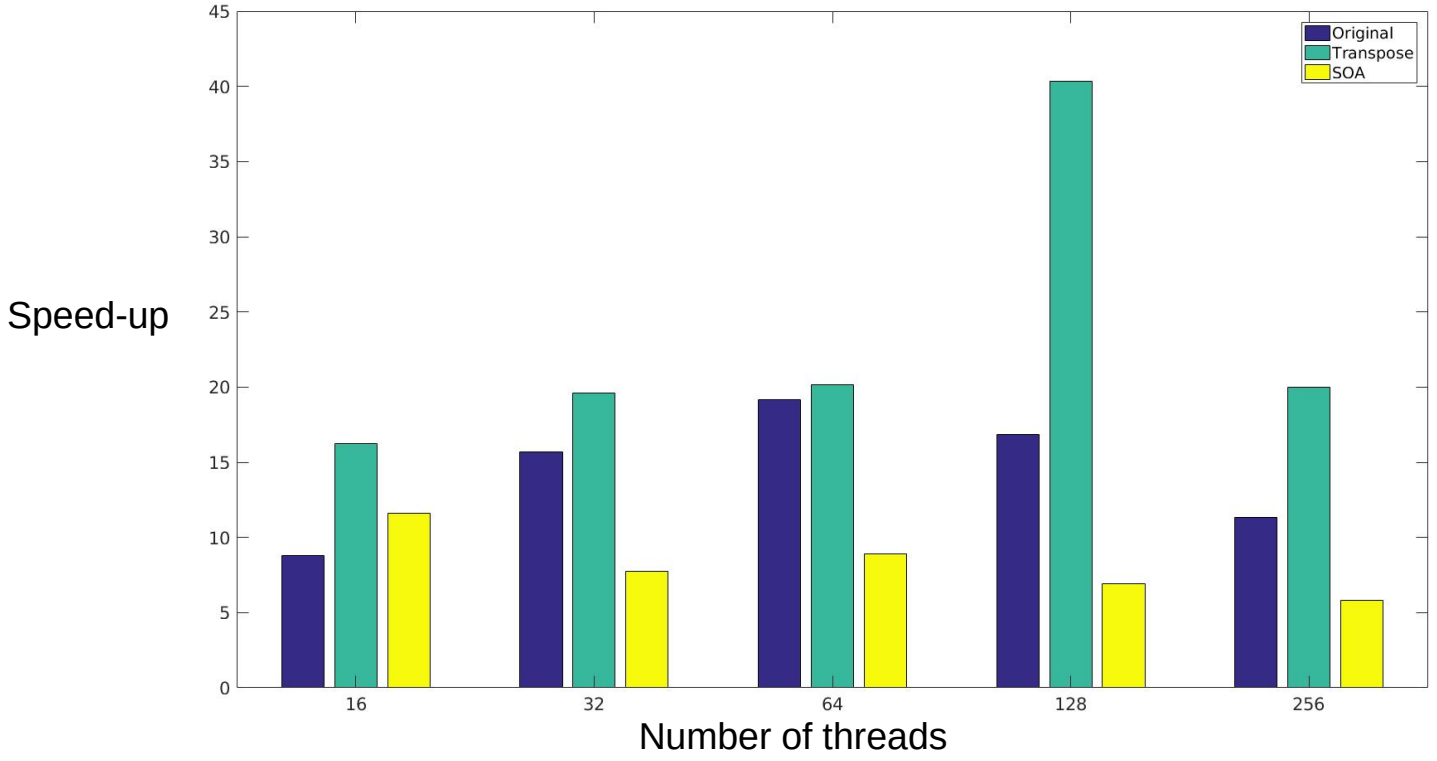
2D Arrays Experiments




```
do j=2,je-1
!$OMP SIMD
do i=1,ie
 a(i,j)=0.1+c(i,j)/d(i,j)
 b(i,j)=(0.2+c(i,j-1)-c(i,j))/(c(i,j)-c(i,j-1)+0.5)
end do
```

2D case

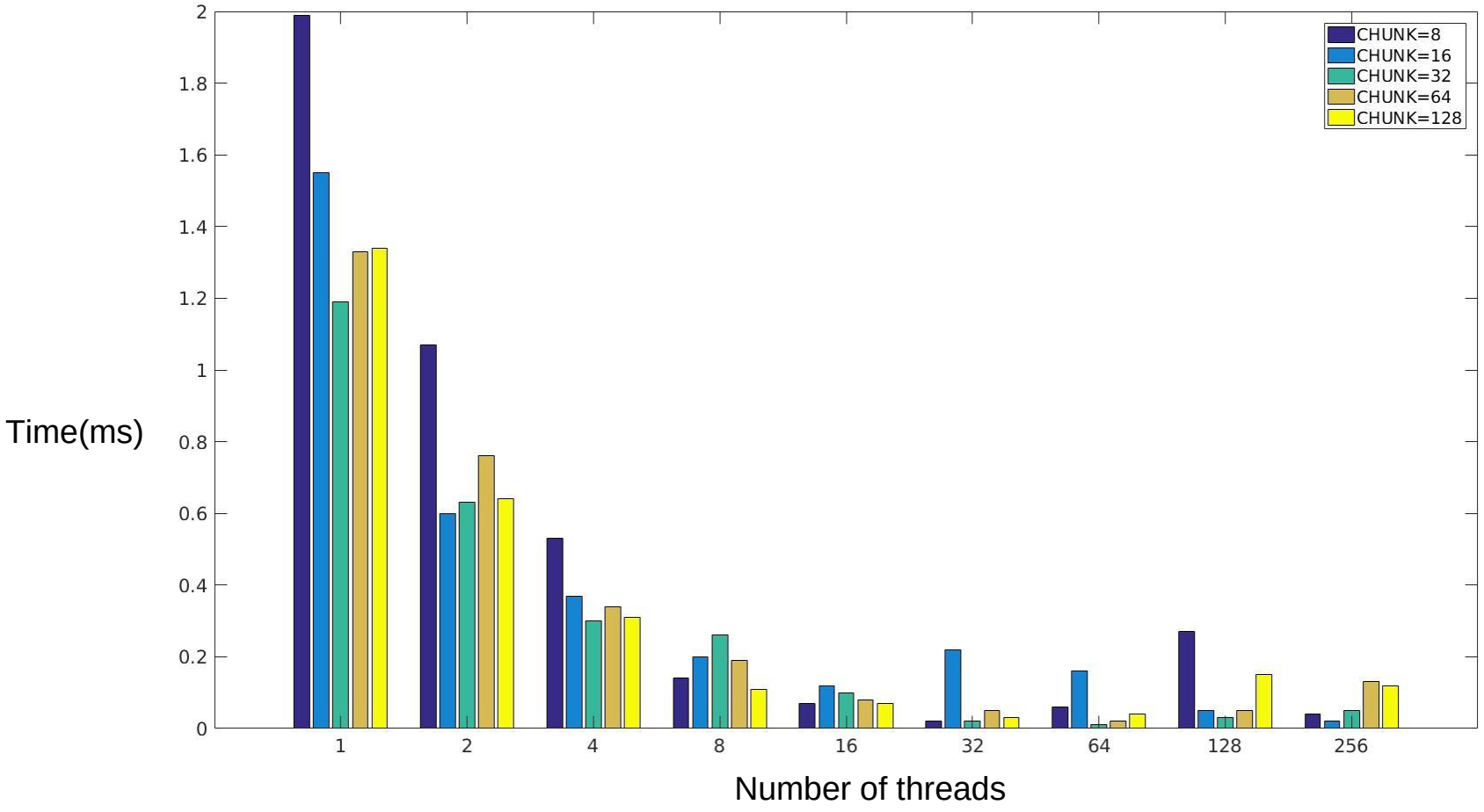
2D Arrays Experiments




```
do j=2,je-1
!$OMP SIMD
do i=1,ie
 a(i,j)=0.1+c(i,j)/d(i,j)
 b(i,j)=(0.2+c(i,j-1)-c(i,j))/(c(i,j)-c(i,j-1)+0.5)
end do
```

2D case with large array sizes

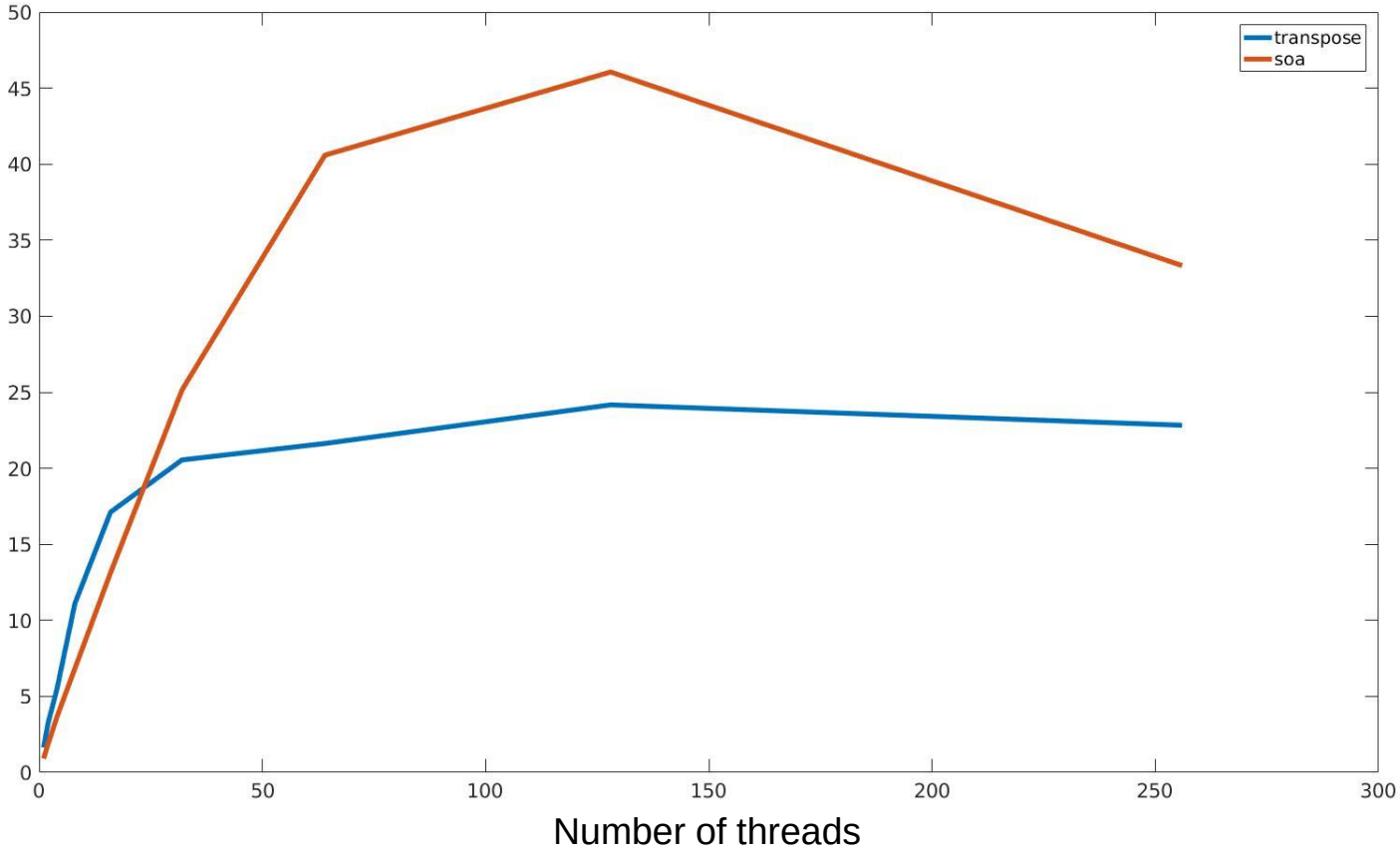
Chunk Size


Time(ms)
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
1
2
4
8
16
32
64
128
256
Number of threads
CHUNK=8
CHUNK=16
CHUNK=32
CHUNK=64
CHUNK=128

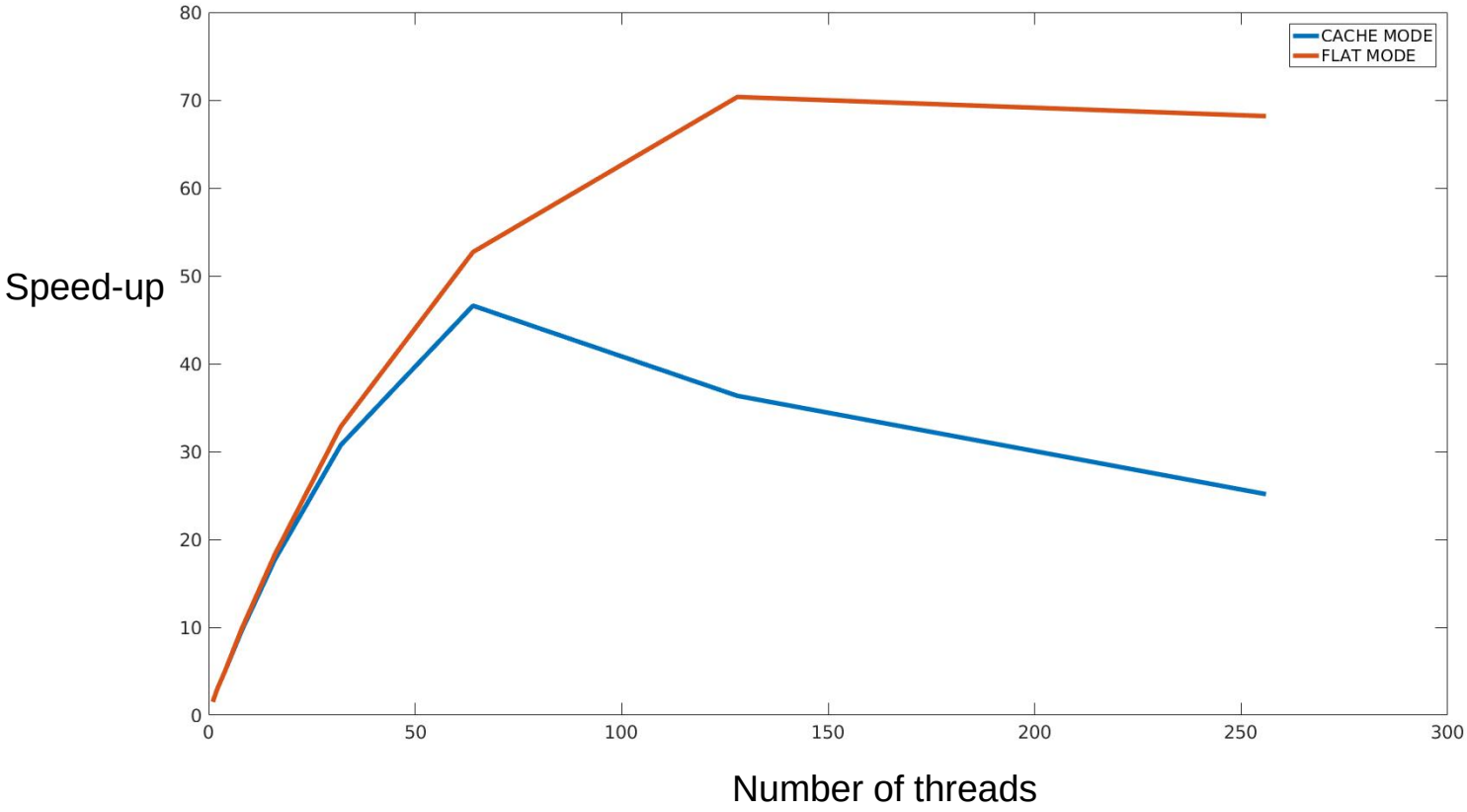
WSM6 Optimizations

- "Low-level"
  - Vectorization
  - Code restructuring
- "High-level"
  - Data reorganization
  - Thread parallelism

Transpose vs SOA



70x Speed-up on WSM6



Conclusion

- "Low-level" and "high-level"
  - Code restructuring
  - Data reorganization
  - Leverage architecture resources
  - Effective use of OpenMP features

- Performance
  - SOA best result
  - From 3x to **70x**
  - Vtune suggests 5.6% peak

Future Work

- GFS operational physics in NEPTUNE
- Mapping between physics and dynamics
- Scalability on large system (OpenMP+MPI)
- Lightweight runtime system
- Other data reorganization approaches

Acknowledgments

- DOD HPCMP PP-KY07-CWO-001-P3
- Intel Parallel Computing Center (IPCC)
- Alex Reinecke, Kevin Viner (NRL)
- John Michelackes (UCAR) Lars Koesterke (TACC)
- Rajiv Bendale, Hugh Thornburg (Engility)

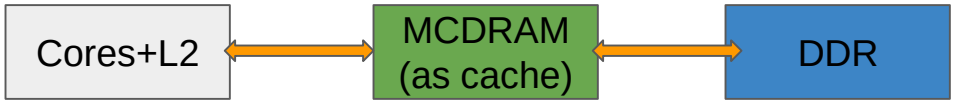
Thank you !!

Questions?

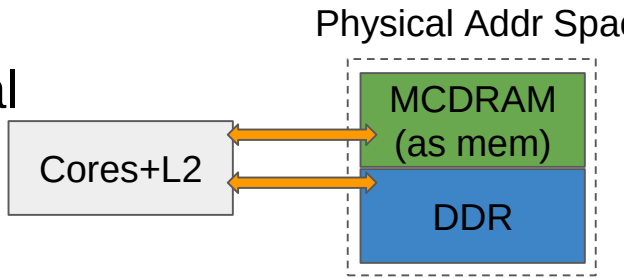
E-mail: touermi@sci.utah.edu

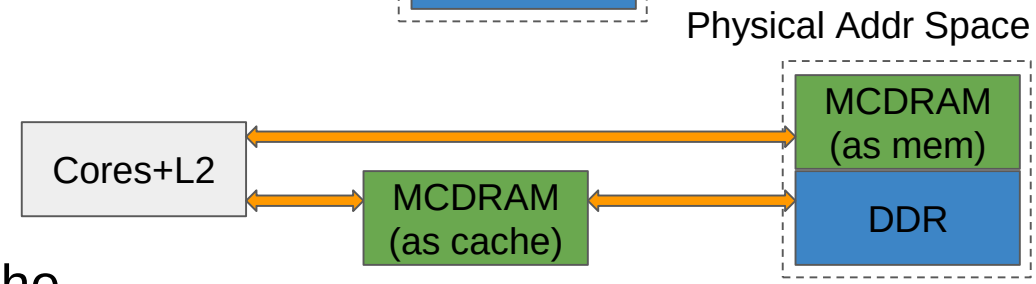
MCDRAM & Configurations

- **Cache Mode**
  - No source changes needed
  - Misses are expensive (higher latency)



- **Flat Mode**
  - MCDRAM mapped to physical address
    - use numactl -- for configuration
  - Exposed as NUMA node





- **Hybrid Mode**
  - Combination of flat and cache mode
    - eg: 8GB cache and 8GB flat

Threads	Time (ms)			Speed-up		
	Orig.	Transp.	SOA	Orig.	Transp.	SOA
1	2.06	3.3 6	3.33	1	0.61	0.62
2	1.59	1.97	1.74	1.30	1.05	1.18
4	0.91	1.44	0.84	2.26	1.43	2.45
8	0.67	0.5	0.41	3.07	4.12	5.02
16	0.55	0.26	0.18	3.75	7.92	11.44
32	0.54	0.17	0.15	3.81	12.12	13.73
64	0.72	0.05	0.11	2.86	41.20	18.73
128	0.87	0.05	0.06	2.37	41.20	34.33
256	1.35	0.1	0.49	1.53	20.60	4.20

Threads	Time (ms)			Speed-up		
	Orig.	Transp.	SOA	Orig.	Transp.	SOA
1	33.82	29.53	75.45	1.00	1.15	0.45
2	26.98	19.44	45.7	1.25	1.74	0.74
4	15.54	13.47	23.37	2.18	2.51	1.45
8	10.9	5.09	7.44	3.10	6.64	4.55
16	8.86	2.98	5.96	3.82	11.35	5.67
32	8.93	2.61	1.72	3.79	12.96	19.66
64	10.97	0.95	1.39	3.08	35.60	24.33
128	16.14	1.17	5.93	2.10	28.91	5.70
256	22.27	2.17	9.57	1.52	15.59	3.53

TABLE II

Threads	Time (ms)			Speed-up		
	Orig.	Transp.	SOA	Orig.	Transp.	SOA
1	2.06	3.36	1.99	1.00	0.61	1.04
2	1.59	1.97	1.07	1.30	1.05	1.93
4	0.91	1.44	0.53	2.26	1.43	3.89
8	0.67	0.5	0.14	3.07	4.12	14.71
16	0.55	0.26	0.07	3.75	7.92	29.43
32	0.54	0.17	0.02	3.81	12.12	103.00
64	0.72	0.05	0.06	2.86	41.20	34.33
128	0.87	0.05	0.27	2.37	41.20	7.63
256	1.35	0.1	0.04	1.53	20.60	51.50

TABLE III

Removal of Fortran Keywords

```
sum_precip: do k=1,km
        if(condition1)
           update precip
           cycle sum_precip
        elseif(condition2)
           update precip
           exit sum_precip
        end if
        exit sum_precip
end do sum_precip
```

```
sum_mask = 1

sum_precip: do k=1,km
        if(condition1 .and sum_mask)
           update precip
        elseif(condition2 .and. sum_mask)
           update precip
           sum_mask = 0
        else
        sum_mask = 0
end do sum_precip
```

```
100    continue
           .
           .
           .
       if(n .le. inter)
       goto 100
       end if
```

```
Do n=1,iter
      .
      .
      .
end do
```

Threads	Time (ms)			Speed-up		
	Orig.	Transp.	SOA	Orig.	Transp.	SOA
1	264.71	194.94	159.98	1.00	1.36	1.65
2	119.93	120.69	113.15	2.21	2.19	2.34
4	98.89	61.57	57.08	2.68	4.30	4.64
8	54.17	25.57	34.25	4.89	10.35	7.73
16	30.11	16.3	22.83	8.79	16.24	11.59
32	16.87	13.51	34.23	15.69	19.59	7.73
64	13.81	13.15	29.72	19.17	20.13	8.91
128	15.74	6.56	38.25	16.82	40.35	6.92
256	23.33	13.24	45.51	11.35	19.99	5.82

TABLE IV

Single vs Multiple Parallel Sections

- Environment variable to keep threads alive (KMP_BLOCKTIME). This is more useful for multiple parallel sections
- Multiple sections are not much worse, especially if work_loops are large
  - This lets us execute short serial code in between WSM6 loops, if needed.

```
work_loops
work_loops
work_loops
```

Serial

```
!$OMP PARALLEL
!$OMP DO
work_loops
!$OMP END DO

!$OMP DO
work_loops
!$OMP END DO

!$OMP DO
work_loops
!$OMP END DO
!$OMP END PARALLEL
```

Speedup: **16.2x**

```
!$OMP PARALLEL
!$OMP DO
work_loops
!$OMP END DO
!$OMP END PARALLEL

!$OMP PARALLEL
!$OMP DO
work_loops
!$OMP END DO
!$OMP END PARALLEL

!$OMP PARALLEL
!$OMP DO
work_loops
!$OMP END DO
!$OMP END PARALLEL
```

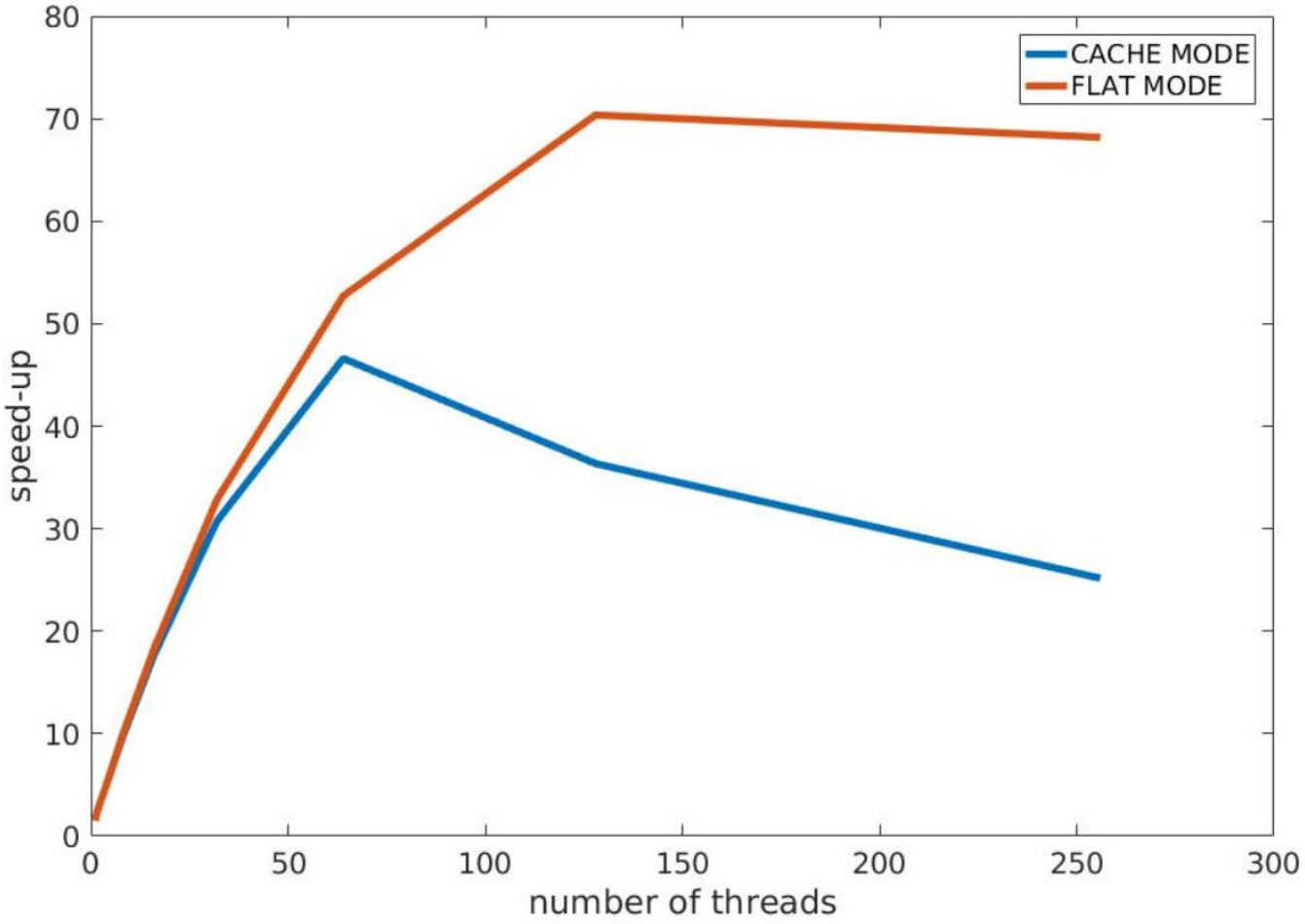
Speedup: **15.9x**

Speed-up per Loop

- slope_wsm6 has different speed-ups for the same routine. Thread invocation time and memory access impact runtime
- Loop 22 and 23 are simple with no complex logic
- Overall, good scalability to 64 cores on KNL
- Includes loop 12 (from standalone example). OMP SIMD enables 41x speedup, including nested conditionals, subroutines
- Final copy of the result arrays shows significant thrashing

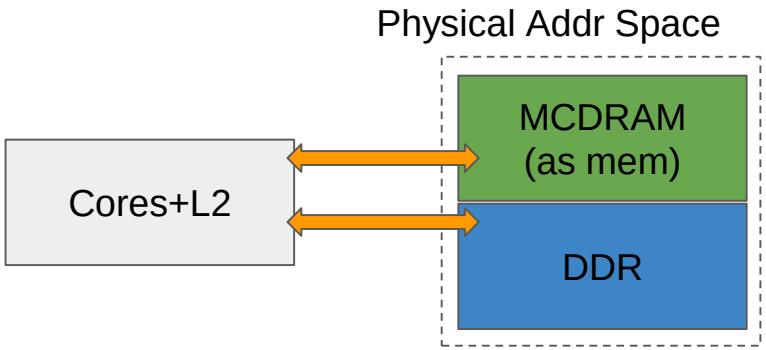
	Speedup	
Loop	**KNL**	**Haswell**
1	14	4.9
2-4	19	4.5
5-6	36	10
7	15	4.2
slope_wsm6	**55**	**8.7**
8	14	3.7
9-11	6.9	3.7
12-14	**41**	**3.0**
15-17	**74**	**19**
18-19	3.5	4.2
slope_wsm6	**45**	**5.6**
20-21	34	2.8
22	**98**	**13**
23	**100**	**5.5**
24-26	57	12
27	**.77**	**0.80**

WSM6 Results with Flat and Cache Mode

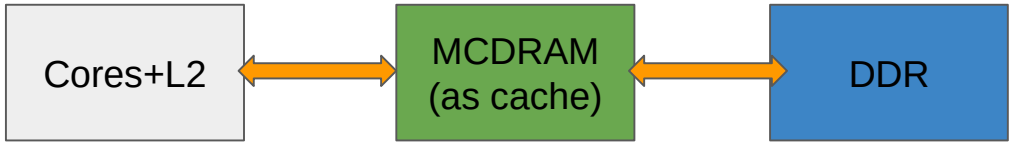


MCDRAM & Configurations

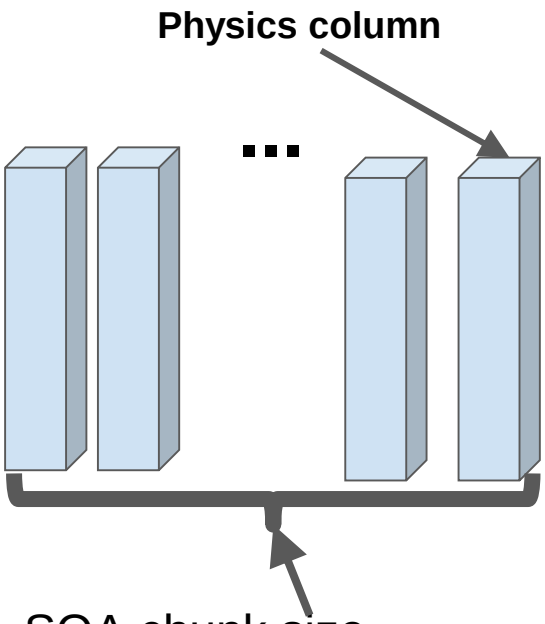
- **Flat Mode**
  - MCDRAM mapped to physical address
    - use numactl -- for configuration
  - Exposed as NUMA node



- **Cache Mode**
  - No source changes needed
  - Misses are expensive (higher latency)



Structure of Arrays (SOA)



SOA chunk size.
Corresponds to parts of i loop.

AOS

```
struct aos{
  double x, y, z;
};
struct aos
points[1024];
```

SOA

```
struct soa{
  double x[1024], y[1024];
  double z[1024];
};
struct soa points;
```

Basic AOS to SOA

```
!$OMP DO
for k=1, km
for i=1,im
  dza(i,k)=zi(i,k)-za(i,k)
enddo
Enddo
!$OMP END DO
```

```
!$OMP DO
for i=1,im
for k=1, km
  dza(k,i)=zi(k,i)-za(k,i)
enddo
Enddo
!$OMP END DO
```

Transpose example

- Simple example of SOA.
- Figure to the right shows actual SOA used in WSM6 optimization.
- Chunk size is chosen to be multiple of vector unit length.
- Top down optimization approach = From “high-level” to “low-level”