Performance Optimization on the Intel Xeon Phi

Presenter: Dhananjay Brahme
Parallelization and Optimization Center of Excellence
TATA Consultancy Services, SahyadriPark Pune, India

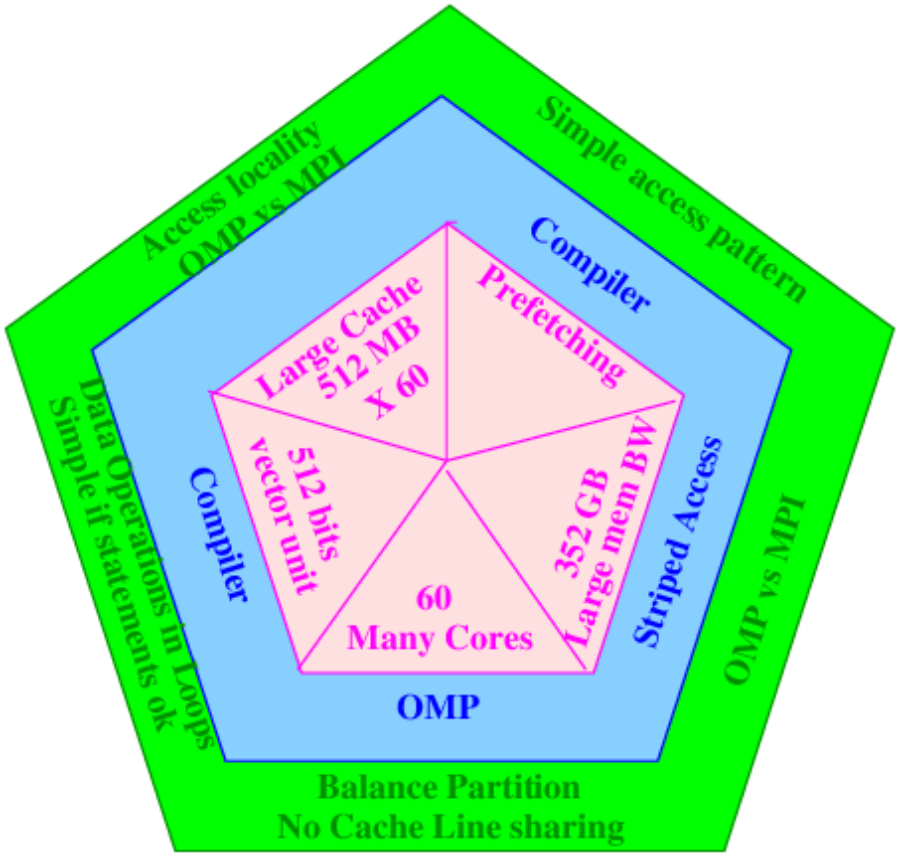
July 8, 2014

Platform, Applications, Performance

Parameter Name / Processor	Cores	Clock	L2 Cache	Vector size	Max Memory Size	Memory Type	Memory BW GB/s
Xeon Phi 7120	61	1.233Ghz	30.5MB	512 bits	16GB	GDDR5	352
Xeon e5 2697	12X2	2.7Ghz	30MB X 2	256 bits	64GB	DDR3	59.7 X 2
Xeon E5-2650	8x2	2.6 GHz	20MB X 2	256 bits	64 GB	DDR3	51.2 GB/s

Application	Description	Configuration/Parameters			Performance		
		Size	No of Time steps	No of Threads	Xeon	Xeon Phi	Xeon Phi / Xeon
Amber	Molecular Dynamics Simulation	Nucleosome 25095 atoms	2500	180	1216 ms	1959.6 ms	1.61
		Myoglobin 2500 atoms	25000	180	18 ns/d	5.65 ns/d	3.19
OpenFoam	Computational Fluid Dynamics	4.8 million cells	100	60	213 secs	635 secs	2.98
ROMS	Ocean Modelling Software	2048 X 256 (Area) X 30 (depth)	30	120	37 sec	153 sec	4.14

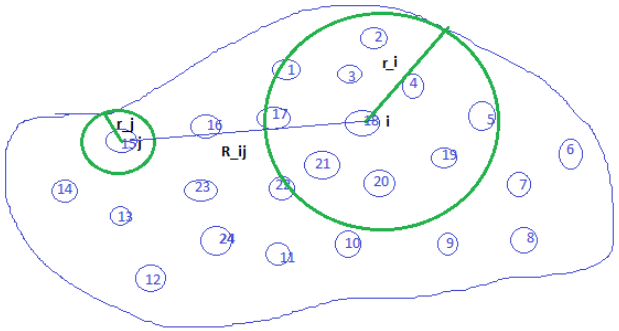
Intel Xeon Phi Hardware and Software Platform



1. Hardware: many core, large coherent L2 cache, 64 byte vector unit, bidirectional 64 byte data ring interconnect.
2. Software: vectorizing compiler, omp.
3. Tools: vtune profiler, debugger, etc.
4. User: balanced partitioning, simple statements within for loops to enable vectorization.

Amber: Generalized Born Model

1. Protein atoms in a solvent (water).
2. Allows computation over protein molecules.
3. Reduces amount of computation.
4. Born Radii and Energy.



Amber: Baseline

Hotspot	Problem	Baseline ms	% of Total
Born_radii	MPI, sparse set, Complex conditionals, reduction, merged data, unaligned access	159.8	8.16
Off-Diagonal Energy	MPI, Complex conditionals, indexed access, reduction, merged data, unaligned access	1573.1	80.28
Diagonal Energy	MPI, sparse set, Complex conditionals, reduction, merged data, unaligned access	190.6	9.73
Other		36.1	1.85
Total		1959.6	

$>$ 98% in top three hotspots

Amber: Born Radii and Diagonal Energy Computations

```
!born radii and diagonal energy computations
for(i=0; i < Num_of_atoms; i++){
  access_atom_info(i);
  for(j=0; j < Num_of_atoms; j++){
    access_atom_info(j);
    d= compute_distance(i, j);
    select_atoms(d < cut_of);
  }
  for_selected_set_do{
    vector_computes(k);
    compute_values(i) =
    nested_if_else_computations(k);
  }
  update_values(i);
}
```

```
!off diagonal energy
for(i=0; i < Num_of_atoms; i++){
  access_atom_info(i);
  for(j=i+1; j < Num_of_atoms; j++){
    access_atom_info(j);
     vector_computes(j);
     nested_if_else_statements(j);
  }
  update_values(i);
}
reduce_values_across_threads();
 }
```

Amber: Born Radii optimizations of nested if else

```
if (dij .le. rgbmax + sj) then

if ((dij .gt. rgbmax - sj)) then

uij = 1.d0 / (dij - sj)
reff_i = reff_i - 0.125d0 * dij1i * &
(1.d0 + 2.d0 * dij *uij + &
rgbmax2i * (r2 - 4.d0 * rgbmax * dij &
- sj2) + 2.d0 * log((dij - sj) * rgbmax1i))

else if (dij .gt. 4.d0 * sj) then

dij2i = dij1i * dij1i
tmpsd = sj2 * dij2i
dumbo = ta + tmpsd *
(tb + tmpsd * (tc + tmpsd *
(td + tmpsd * tdd)))

reff_i = reff_i - tmpsd * sj * &
dij2i * dumbo
.
.
.
```

Amber: Born Radii optimizations of nested if else

```
if (dij .le. rgbmax + sj) then

if ((dij .gt. rgbmax - sj)) then

<Computation1>.

else if (dij .gt. 4.d0 * sj) then

<Computation2>.
.
.
```

```
if (dij .le. rgbmax + sj) .and.

((dij .gt. rgbmax - sj)) then

< Computation1>

if (dij .le. rgbmax + sj) .and.
((dij .le. rgbmax - sj)).and.
(dij .gt. 4.d0 * sj) then

<Computation2>.
```

Amber: Off-Diagonal optimizations aligned access

```
for (i =0; i < NoOfAtoms; i++){
  for(j=i+1; j < NoOfAtoms; j++){
    <Computations>
  }
}
```

```
for (i =0; i < NoOfAtoms; i++){
  ib =(i/8+1)*8;

  for(j=i+1; j < ib; j++){
    <Computations>
  }

  for(j=ib; j < NoOfAtoms; j++){
    <Computations>
  }
}
```

Amber: Optimizations

Hotspot	Optimization	Baseline ms	% of of Total	Optimized ms	% of of Total	Improvement
Born_radii	MPI to OpenMP, Simplify conditionals, Remove False sharing	159.8	8.16	50	9.47	3.2
Off-Diagonal Energy	MPI to OpenMP, Simplify conditionals, Remove false sharing, Aligned Access, Simd reduction, split data structures, precompute array index	1573.1	80.28	381	72.16	4.13
Diagonal Energy	MPI to OpenMP, Simplify conditionals, Remove False sharing	190.6	9.73	69	13.07	2.76
Other		36.1	1.85	28	5.3	1.3
Total		1959.6		528		3.71

Amber: Nucleosome Offload Mode

Arch	Configuration		Baseline		Optimized		Improvement
	Cores	Threads	ns/day	ms	ns/day	ms	
Xeon	24	24	0.18	1216.4	0.37	464.4	2.62
Xeon Phi	60	180	0.09	1959.6	0.31	528	3.71
Offload Xeon + Xeon Phi*	24+60	24+180			0.51	325	1.43**

*Only off Diagonal is offloaded from 1 Xeon rank to 180 Xeon Phi ranks.
**Xeon/(Xeon+Xeon Phi)

OpenFoam: Computational Fluid Dynamics

1. Flow analysis of Motor Bike and Car.
2. Gauss Siedel Solver used for solving $AXx = b$.
3. $(L + D + U) * x = b => (L + D) * x = b - U * x$.
4. $(D^{-1} * L + I) * xnew = D^{-1} * (b - U * x)$.

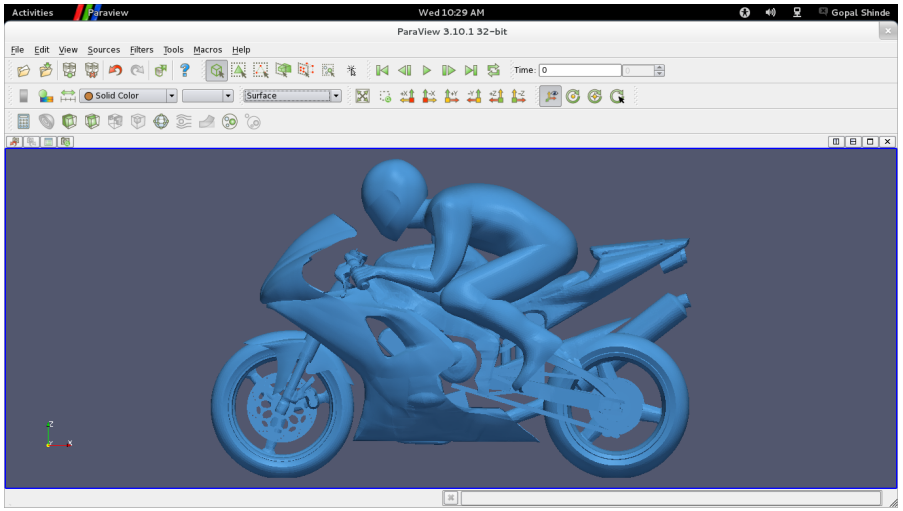


Figure: Motor Bike CFD*

*Source: Openfoam tutorial

OpenFoam Baseline

Hotspot	Problem	Baseline (ms)	% of Total
Gauss Siedel	Indexed access , short loops, unaligned data, division	101	16
Residual	short loops	26	4.1
Other Hotspots	short loops	254	40
Total		635	

OpenFoam: Optimizations

Optimization	Optimization Category	Time (s)
Baseline + Running from /home instead of /lfs		635
Secondary logging off	I/O and file system	520
renumberMesh utility	Cache locality	495
GaussSeidelSmoother	Precomputing + Vectorization + Data Rearrangement Vectorization Xscatter Intrinsics Prefetching	 470 452 445
Residual	Cache reuse	435
Loop Unrolling	Loop Unrolling	400
Other	Vectorization Prefetching Loop Split	 390
Compiler flags Compiler flags Compiler Flag	ansi-alias, -no-ansi-alias-check -opt-streaming-stores-always, -restrict -opt-prefetch-distance=72,"	 368 358
Use of 2MB pages for TLB pressure	Huge memory pages	 335
Post Processing Off	I/O and Post computation	307
Partitioning	(10,6,1) instead of (60,1,1)	286
Compiler Flag	-fimf-precision=low -fimf-domain-exclusion=31 Prefetch Intrinsic	 278
Nsweeps = 2	General	271

OpenFoam: Computational Fluid Dynamics

1. $(L + D) * x^{new} = b - U * x$.
2. U is a sparse matrix represented as an array of column, value pairs.
3. A cumulative count of total number of elements till i^{th} row is maintained.

Sparse Matrix Multiplication:

count	0	1	2	3	4	5	6	7	8	9
0	8			4	4		2	1		
1		8	3	2	1					
2		1	8			4	4	2		
3	1	2		8		1	4	1		
4	1	1			8	1	4	2		
5			4	1	1	8			1	
6	2		4	4	4		8	1	1	
7	1		1	1	2		1	8	1	
8						2	2	1	8	1
9									1	8

X	XScatter
.1	.4
.2	.5
.3	.7
.4	.8
.5	.3
.6	.4
.7	.5
.8	.6
.9	.7
1.1	.8

Sparse Matrix Multiplication

columns: 3 4 6 7 2 3 4 5 6 7 5 6 7 5 6 7 8 7 8 8 9
values: 4 4 2 1 3 2 1 4 4 2 1 4 1 1 4 2 1 1 1 1 1
row_counts: 0 4 7 10 13 16 17 19 20 21 21
XVector: .1 .2 .3 .4 .5 .6 .7 .8 .9. 1.1

```
/* Problems:
 Indirect access of x.
 prefetching is hard,
 vectorization uses gather making it slow
*/
for(i=0; i < N; i++){
   jstart = jend;
   jend = row_count[i+1];
   for(j=jstart; j < jend; j++){
      result[i]= vals[j]*x[cols[j]];
   }
}
```

Sparse Matrix Multiplication

columns: 3 4 6 7 2 3 4 5 6 7 5 6 7 5 6 7 8 7 8 8 9
values: 4 4 2 1 3 2 1 4 4 2 1 4 1 1 4 2 1 1 1 1 1
row_counts: 0 4 7 10 13 16 17 19 20 21 21
XScatter: .4 .5 .7 .8 .3 .4 .5 .6 .7 .8 .6 .7 .8 .6 .7 .8 .9 .8 .9 .9 1.1

```
/* Use xscatter array
prefetching is easy.
vectorization is easy.
*/
for(i=0; i < N; i++){
   jstart = jend;
   jend = row_count[i+1];
   for(j=jstart; j < jend; j++){
      result[i]= vals[j]*xscatter[j];
   }
}
```

OpenFoam Results:

Hotspot	Optimization	Baseline ms	% of of Total	Optimized ms	% of of Total	Improvement
Gauss Siedel	Precomputation Scatter x Intrinsics Prefetching	101	16	53	8.4	1.9
Residual	Cache Reuse	26	4.1	18	2.8	1.46
Other Hotspots	Unrolling Prefetching Vectorization	254	40	190	30	1.33
Total		635		301		2.11

Symmetric Performance

	Configuration		Baseline	Optimized	Improvement
	Cores	Threads			
Xeon	24	24	213	201	1.06
Xeon Phi	60	60	635	301	2.11
*Symmetric Xeon + Xeon Phi	16+60	16+60		210	0.95**

*5X cells per Xeon Rank 1X cells per Xeon Phi Rank

**Xeon/(Xeon Phi)

ROMS: Ocean Modelling Software

1. Variables: pressure, salinity, currents, waves, temperature and density.
2. 3D grid, Compute over a certain area X depth.
3. Observation and Measurement over a set of points at certain times.
4. Model equations: interpolate values where observation is not available and predict future values.

Hotspots transformed

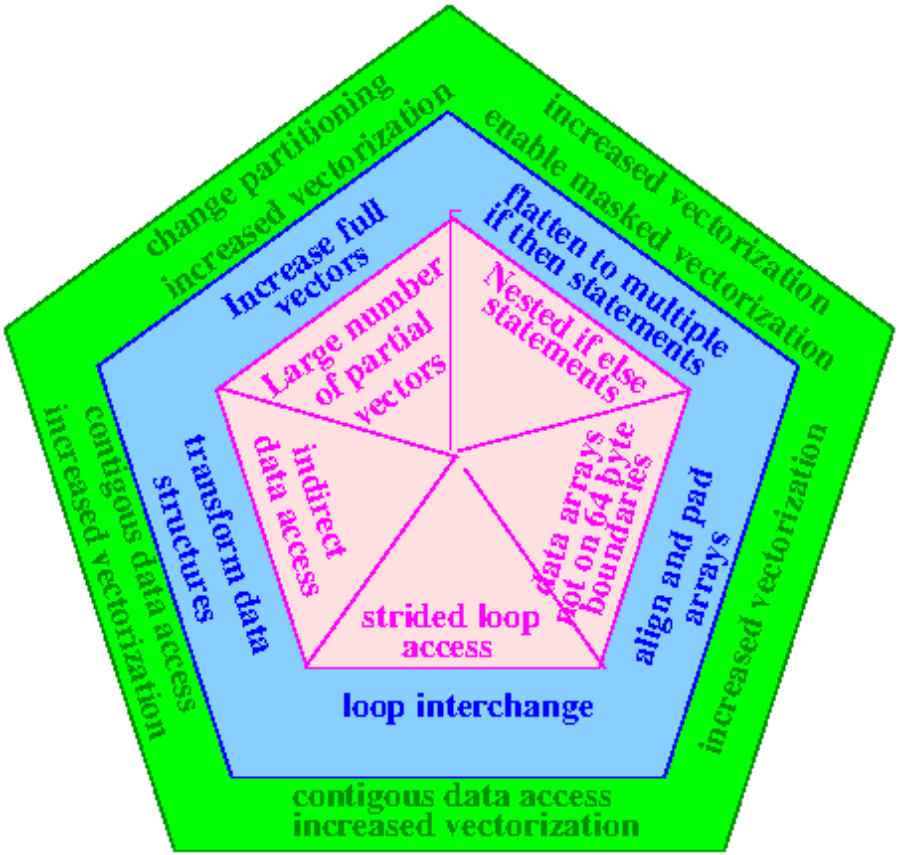
1. (FC) Flatten nested if else, 2. (LI) Induce contiguous access, 3. (AA) create Aligned access

Function	Baseline (B)	% of Total	Optimized (O)	% of Total	Improvement B/O
lmd_skpp_tile(FC,LI,AA)	15.51	10.14	2.74	4.42	5.66
t3dmix2_tile	12.2	7.96	3.07	4.95	3.97
uv3dmix2_tile	8.13	5.31	1.15	1.85	7.06
step2d_tile	7.07	4.62	6.65	10.73	1.06
prsgrd_tile	6.16	4.02	1.62	2.61	3.81
lmd_vmix_tile	5.77	3.77	1.32	2.13	4.37
step3d_uv_tile	5.34	3.49	3.76	6.06	1.42
diag_tile	4.13	2.7	1.86	3	2.22
pre_step3d_tile	4.09	2.67	4.73	7.62	0.87
step3d_t_tile	3.38	2.2	3.52	5.67	0.96
rhs3d_tile	3.2	2.09	3.27	5.27	0.98
omega_tile	2.14	1.39	2.42	3.9	0.88
All hotspots	77.12	50.41	36.11	58.24	2.14
Total	153		62		2.47

ROMS:Other Optimizations

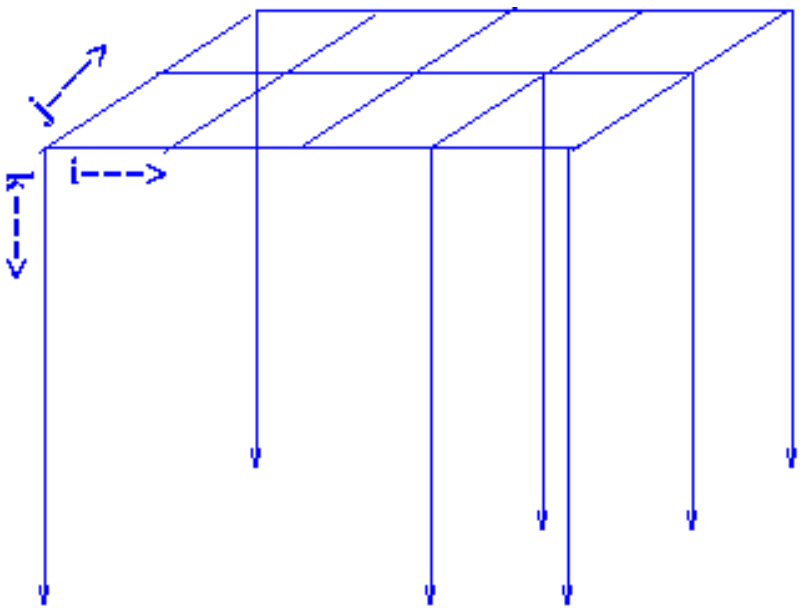
Optimization	Time (s)	Remarks
Baseline	153	Code run just as downloaded
Removing -ip flag	115	-ip flag for inter-procedural optimization was present by default,
		Causing no effect on xeon but adverse effect on MIC.
-fp model fast=1	100	Replaced -fp-model precise with -fp-model fast=1
FC loop optimizations	88	Loop interchange, loop splitting, alignment padding
Tile resizing	78	Redistributed the grid into 10x12 tiles from 30x04 to reduce trip
		Count of loops and lessen the amount of padding.
Other loops	76	Padding, alignment, loop interchange, loop split etc applied on SI_dpth, Ustar, ksbl, Drhs, momentumETA and momentumXI loops.
IVDEP	72	Put IVDEP on many medium impact loops in lmd_skpp.F, Step3d_uv.F and step2d_LM_AM3.h files.
2MB pages	62	Use of 2MB pages for TLB pressure

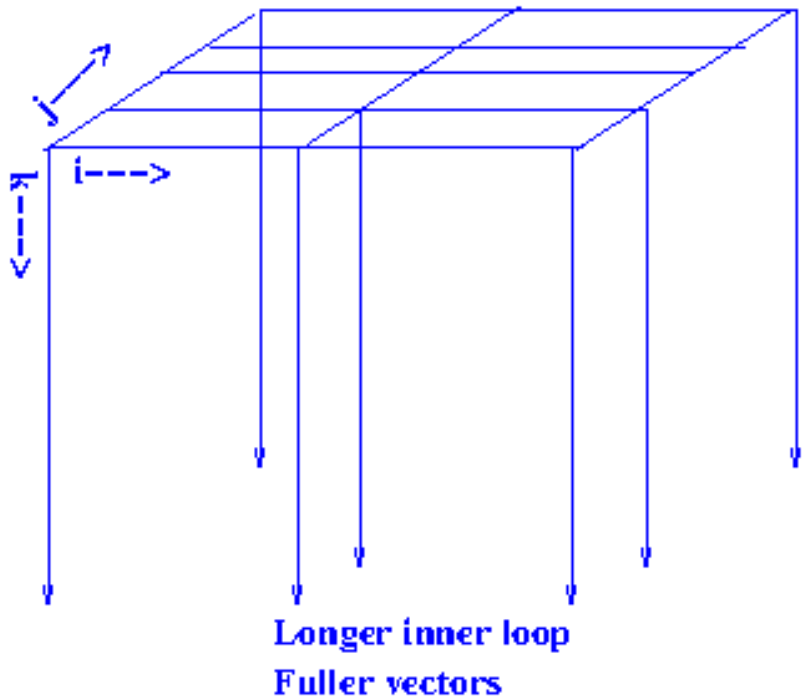
ROMS: Vectorization



1. Increase vectorization: reduce partial vector operation: Large loop lengths, aligned access,
2. Vectorization Enablers: Simple if then statements, contiguous data access, contiguous data storage.
3. Reduce gather used before vector operations.
4. Can get upto 8X performance improvement.

ROMS: Vectorization





Grid	2048X256	2048X256	Improv
Partition/Tiles	30X4	10X12	
Grid per tile	68X64	204X21	
Time (secs)	88	78	12.8

Performance comparision with Xeon

2048 X 256	Configuration		Baseline secs	Optimized secs	Improvement Baseline/Optimized
	Cores	Threads			
Xeon	12 X 2	24 X 2	37	25	1.48
Xeon Phi	60	120?	153	62	2.47

4096 X 512	Configuration		Baseline secs	Optimized secs	Improvement Baseline/Optimized
	Cores	Threads			
Xeon	12 X 2	24 X 2	159	136	1.17
Symmetric Xeon + Xeon Phi	24+60	24+120	351	104	3.375

Application Optimization Summary

Application	Optimization	Baseline	Optimized	Improvement
Amber	MPI to OpenMP, Simplify conditionals, Remove False sharing, Aligned Access, simd reduction, split data structures, precompute array index	1959.6 ms	528 ms	3.71
OpenFoam	Remove Indirect access, Intrinsics, Unrolling	635 secs	301 secs	2.11
ROMS	Flatten conditionals, aligned access, loop interchange	153 secs	62 secs	2.47

Thank You