

#shareEGU20

4 - 8 May 2020

Refactoring the memory access pattern to improve computational performance in NEMO

I.Epicoco, F. Mele, S. Mocavero, M. Chiarelli, A. D'Anca, G. Aloisio



OS4.8 - Numerical modelling of the ocean: new scientific advances in ocean models to foster exchanges within NEMO community and contribute to future developments



NEMO Ocean Model

Single core performance of the NEMO model is limited by memory access and poor exploitation of vector processing units on modern HPC architectures

The analysis of the memory access pattern shows that many repeated accesses occur for reading values not available in cache

-> **high rate of cache miss!!!**



Goal of the work: enhance the exploitation of the cache memory of the modern parallel architectures through the loop fusion approach

Single core performance - Loop fusion

- **Loop fusion technique** aims at better exploiting the cache memory by fusing DO loops together

```
DO j=1, n-1
  DO i=1, n
    b(i,j) = in(i,j+1) - in(i,j)
  END DO
END DO

DO j=2, n-1
  DO i=1, n
    out(i,j) = b(i,j) - b(i,j-1)
  END DO
END DO
```

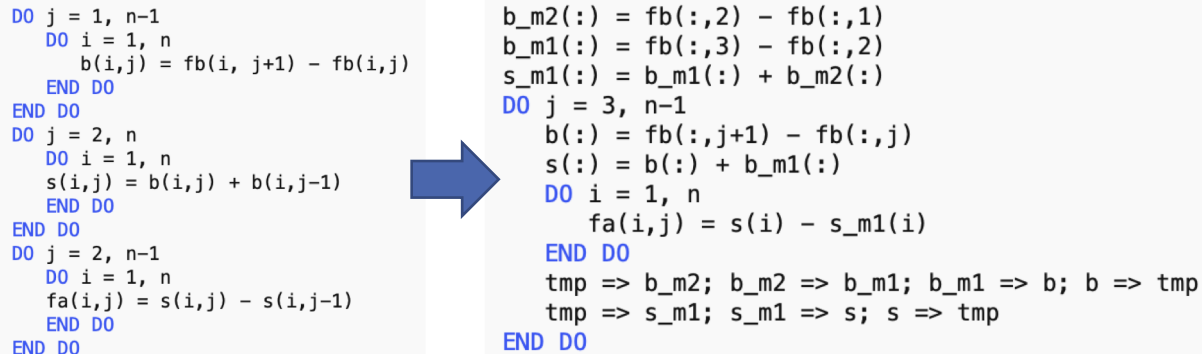
```
DO j=2, n-1; DO i=1, n
  b_0 = in(i,j+1) - in(i,j) ! correspond to b(i,j)
  b_m1 = in(i,j) - in(i,j-1) ! correspond to b(i,j-1)

  out(i,j) = b_0 - b_m1
END DO; END DO
```

- advantages: reduction of cache misses and reduction of the memory footprint
- disadvantage: due to data dependencies redundant operations are needed

Single core performance - Loop fusion

- Loop fusion approach has been applied on the NEMO MUSCL advection kernel
- Three different levels of fusion have been implemented
 - ❑ prototype1: has the maximum level of fusion with redundant operations
 - ❑ prototype2: introduces the buffers rotation¹ in the outer loop
 - ❑ prototype3: uses the buffers rotation in the outer and middle loop



```
DO j = 1, n-1
  DO i = 1, n
    b(i,j) = fb(i, j+1) - fb(i,j)
  END DO
END DO
DO j = 2, n
  DO i = 1, n
    s(i,j) = b(i,j) + b(i,j-1)
  END DO
END DO
DO j = 2, n-1
  DO i = 1, n
    fa(i,j) = s(i,j) - s(i,j-1)
  END DO
END DO
```

➔

```
b_m2(:) = fb(:,2) - fb(:,1)
b_m1(:) = fb(:,3) - fb(:,2)
s_m1(:) = b_m1(:) + b_m2(:)
DO j = 3, n-1
  b(:) = fb(:,j+1) - fb(:,j)
  s(:) = b(:) + b_m1(:)
  DO i = 1, n
    fa(i,j) = s(i) - s_m1(i)
  END DO
  tmp => b_m2; b_m2 => b_m1; b_m1 => b; b => tmp
  tmp => s_m1; s_m1 => s; s => tmp
END DO
```

¹buffers rotation technique avoid redundant operations by adopting pointers to arrays and implementing a rotation at each loop iteration as shown in the figure

First approach – prototype1

- The halo exchange is moved before all DO-loops (at the beginning of the routine)
 - The halo region needs to be extended up to two halo lines
- The advective trend is computed for each single (ji, jj, jk, jn) grid cell within a single big DO-loop
 - This approach implies also a duplication of the calculus up to a factor 3

First approach – prototype1

BaseLine

```
DO jn = 1, kjpt      !== loop over the tracers ==!
  !!-- initial slop
  DO jk = 1, jpkml
    DO jj = 1, jpjm1
      DO ji = 1, fs_jpim1
        initial_slop_i(zwx(ji,jj,jk), ji)
        initial_slop_j(zwy(ji,jj,jk), jj)
      END DO
    END DO
  END DO

  CALL mpp_lnk_3d(zwx, 1); CALL mpp_lnk_3d(zwy, 1)

  !!-- Slopes of tracer
  DO jk = 1, jpkml
    DO jj = 2, jpj
      DO ji = fs_2, jpi ! vector opt.
        tracer_slop(zslpx(ji,jj,jk), zwx(ji,jj,jk), zwx(ji-1,jj,jk) )
        tracer_slop(zslpy(ji,jj,jk), zwy(ji,jj,jk), zwy(ji,jj-1,jk) )
      END DO
    END DO
  END DO

  !!-- Slopes limitation
  DO jk = 1, jpkml
    DO jj = 2, jpj
      DO ji = fs_2, jpi ! vector opt.
        limitation_slop(zslpx(ji,jj,jk), zslpx(ji,jj,jk), zwx(ji-1,jj,jk), zwx(ji,jj,jk) )
        limitation_slop(zslpy(ji,jj,jk), zslpy(ji,jj,jk), zwy(ji,jj-1,jk), zwy(ji,jj,jk) )
      END DO
    END DO
  END DO

  !!-- MUSCL horizontal advective fluxes
  DO jk = 1, jpkml
    DO jj = 2, jpjm1
      DO ji = fs_2, fs_jpim1 ! vector opt.
        vertical_adv_flux_i(zwx(ji,jj,jk), ji, zslpx(ji,jj,jk), zslpx(ji+1,jj,jk))
        vertical_adv_flux_j(zwy(ji,jj,jk), ji, zslpy(ji,jj,jk), zslpy(ji,jj+1,jk))
      END DO
    END DO
  END DO

  CALL mpp_lnk_3d(zwx, 1); CALL mpp_lnk_3d(zwy, 1)

  !!-- Tracer advective trend
  DO jk = 1, jpkml
    DO jj = 2, jpjm1
      DO ji = fs_2, fs_jpim1 ! vector opt.
        zbtr = 1. / ( ele2t(ji,jj) * fse3t(ji,jj,jk) )
        ztra_i = zwx(ji-1,jj,jk) - zwx(ji,jj,jk)
        ztra_j = zwy(ji,jj-1,jk) - zwy(ji,jj,jk)
        pta(ji,jj,jk,jn) = pta(ji,jj,jk,jn) + zbtr * ( ztra_i + ztra_j )
      END DO
    END DO
  END DO
```

Prototype 1

```
CALL mpp_lnk_4d(ptb, 2)

DO jn = 1, kjpt      !== loop over the tracers ==!
  DO jk = 3, jpk-3
    DO jj = nldj, nlej
      DO ji = nldi, nlei
        !!-- Initial slop !!!
        initial_slop_i(zzx2, ji-2)
        initial_slop_i(zzx1, ji-1)
        initial_slop_i(zzx, ji)
        initial_slop_i(zzxp1, ji+1)
        !!-- Tracer slop & Limitation slop !!!
        tracer_slop(zzslpxm1, zzx1, zzx2)
        tracer_slop(zzslpx, zzx, zzx1)
        tracer_slop(zzslpxp1, zzxp1, zzx)
        limitation_slop(zzslpxm1, zzslpxm1, zzx2, zzx1)
        limitation_slop(zzslpx, zzslpx, zzx1, zzx)
        limitation_slop(zzslpxp1, zzslpxp1, zzx, zzxp1)
        !!-- Horizontal advection flux (x-axes) !!!
        vertical_adv_flux_i(zzxf, ji, zzslpx, zzslpxp1)
        vertical_adv_flux_i(zzxfm1, ji-1, zzslpxm1, zzslpx)

        !!-- Initial slop !!!
        initial_slop_j(zzym2, jj-2)
        initial_slop_j(zzym1, jj-1)
        initial_slop_j(zzy, jj)
        initial_slop_j(zzyp1, jj+1)
        !!-- Tracer slop & Limitation slop !!!
        tracer_slop(zzslpym1, zzym1, zzym2)
        tracer_slop(zzslpy, zzy, zzym1)
        tracer_slop(zzslpy1, zzyp1, zzy)
        limitation_slop(zzslpym1, zzslpym1, zzym2, zzym1)
        limitation_slop(zzslpy, zzslpy, zzym1, zzy)
        limitation_slop(zzslpy1, zzslpy1, zzy, zzyp1)
        !!-- Horizontal advection flux (y-axes) !!!
        vertical_adv_flux_j(zzyf, jj, zzslpy, zzslpy1)
        vertical_adv_flux_j(zzyf1, jj-1, zzslpym1, zzslpy)

        !!-- Initial slop !!!
        initial_slop_k(zzwzm1, jk-1)
        initial_slop_k(zzwz, jk)
        initial_slop_k(zzwzp1, jk+1)
        initial_slop_k(zzwzp2, jk+2)
        !!-- Tracer slop & Limitation slop !!!
        tracer_slop(zzslpz1, zzwzm1, zzwz)
        tracer_slop(zzslpz, zzwz, zzwzp1)
        tracer_slop(zzslpz1, zzwzp1, zzwzp2)
        limitation_slop(zzslpz1, zzslpz1, zzwz, zzwzm1)
        limitation_slop(zzslpz, zzslpz, zzwzp1, zzwz)
        limitation_slop(zzslpz1, zzslpz1, zzwzp2, zzwzp1)
        !!-- vertical advection flux !!!
        vertical_adv_flux_k(zzwzf, jk, zzslpz1, zslpz)
        vertical_adv_flux_k(zzwzfp1, jk+1, zslpz, zslpzp1)

        ! add to the general tracer trends
        zbtr = 1. / ( ele2t(ji,jj) * fse3t(ji,jj,jk) )
        ztra_i = zzxfm1 - zzxf
        ztra_j = zzyf1 - zzyf
        ztra_k = zzwzfp1 - zzwzf
        pta(ji,jj,jk,jn) = pta(ji,jj,jk,jn) + zbtr * ( ztra_i + ztra_j + ztra_k )
      END DO
    END DO
  END DO
```

Second approach – prototype2

- Along the vertical direction:
 - The advective flux at level $jk=1$ is computed before the loop over jk (we call this flux F_{jk-1})
 - Inside the jk loop, the flux at level jk (which we call F_{jk}) is computed
 - We use F_{jk-1} and F_{jk} to calculate the advective trend at level jk and to update the RHS variable at level jk
 - Before incrementing the jk level, we update the flux at level $jk-1$: $F_{jk} \rightarrow F_{jk-1}$
- This approach reduces the number of redundant operations, but it introduces a data dependencies in the jk loop, hence the jk loop can not be vectorized, neither executed in parallel.

Second approach – prototype2

Prototype 1

```
CALL mpp_lnk_4d(ptb, 2)

DO jn = 1, kjpt                !== loop over the tracers ==!
  DO jk = 3, jpk-3
    DO jj = nldj, nlej
      DO ji = nldi, nlei
        !!! Initial slop !!!
        initial_slop_i(zxm2, ji-2)
        initial_slop_i(zxm1, ji-1)
        initial_slop_i(zzx, ji)
        initial_slop_i(zxpl, ji+1)
        !!! Tracer slop & Limitation slop !!!
        tracer_slop(zzslpxm1, zzxml, zxm2)
        tracer_slop(zzslpx, zzx, zxm1)
        tracer_slop(zzslpxp1, zxpl, zzx)
        limitation_slop(zzslpxm1, zzslpxm1, zxm2, zxm1)
        limitation_slop(zzslpx, zzslpx, zxm1, zzx)
        limitation_slop(zzslpxp1, zzslpxp1, zzx, zxpl)
        !!! Horizontal advection flux (x-axes) !!!
        vertical_adv_flux_i(zzxf, ji, zzslpx, zzslpxp1)
        vertical_adv_flux_i(zzxfm1, ji-1, zzslpxm1, zzslpx)

        !!! Initial slop !!!
        initial_slop_j(zzym2, jj-2)
        initial_slop_j(zzym1, jj-1)
        initial_slop_j(zzy, jj)
        initial_slop_j(zzyp1, jj+1)
        !!! Tracer slop & Limitation slop !!!
        tracer_slop(zzslpym1, zzym1, zzym2)
        tracer_slop(zzslpy, zzy, zzym1)
        tracer_slop(zzslpypl, zzyp1, zzy)
        limitation_slop(zzslpym1, zzslpym1, zzym2, zzym1)
        limitation_slop(zzslpy, zzslpy, zzym1, zzy)
        limitation_slop(zzslpypl, zzslpypl, zzy, zzyp1)
        !!! Horizontal advection flux (y-axes) !!!
        vertical_adv_flux_j(zzyf, jj, zzslpy, zzslpypl)
        vertical_adv_flux_j(zzyfm1, jj-1, zzslpym1, zzslpy)

        !!! Initial slop !!!
        initial_slop_k(zzwzm1, jk-1)
        initial_slop_k(zzwz, jk)
        initial_slop_k(zzwzp1, jk+1)
        initial_slop_k(zzwzp2, jk+2)
        !!! Tracer slop & Limitation slop !!!
        tracer_slop(zzslpzml, zzwzm1, zzwz)
        tracer_slop(zzslpz, zzwz, zzwzp1)
        tracer_slop(zzslpzpl, zzwzp1, zzwzp2)
        limitation_slop(zzslpzml, zzslpzml, zzwz, zzwzm1)
        limitation_slop(zzslpz, zzslpz, zzwzp1, zzwz)
        limitation_slop(zzslpzpl, zzslpzpl, zzwzp2, zzwzp1)
        !!! vertical advection flux !!!
        vertical_adv_flux_k(zzwzf, jk, zzslpzml, zzslpz)
        vertical_adv_flux_k(zzwzfp1, jk+1, zzslpz, zzslpzpl)

        ! add to the general tracer trends
        zbtr = 1. / ( e1e2t(ji,jj) * fse3t(ji,jj,jk) )
        ztra_i = zzxfm1 - zzx
        ztra_j = zzyfm1 - zzy
        ztra_k = zzwzfp1 - zzwzf
        pta(ji,jj,jk,jn) = pta(ji,jj,jk,jn) + zbtr * ( ztra_i + ztra_j + ztra_k )
      END DO
    END DO
  END DO
END DO
```

Prototype 2

```
CALL mpp_lnk_4d(ptb, 2)

DO jn = 1, kjpt                !== loop over the tracers ==!
  DO jj = nldj, nlej
    DO ji = nldi, nlei
      initial_slop(zzwzm1, 2); initial_slop(zzwz, 3); initial_slop(zzwzp1_ptr(ji,jj), 4)
      tracer_slop(zzslpzml, zzwzm1, zzwz); tracer_slop(zzslpz, zzwz, zzwzp1_ptr(ji,jj))
      limitation_slop(zzslpzml, zzslpzml, zzwz, zzwzm1)
      limitation_slop(zzslpz_ptr(ji,jj), zzslpz, zzwzp1_ptr(ji,jj), zzwz)
      vertical_adv_flux(zzwzf_ptr(ji,jj), 3, zzslpzml, zzslpz_ptr(ji,jj))
    END DO
  END DO

  DO jk = 3, jpk-3
    DO jj = nldj, nlej
      DO ji = nldi, nldi
        initial_slop(zzwzp2_ptr(ji,jj), jk+2)
        tracer_slop(zzslpzpl, zzwzp1_ptr(ji,jj), zzwzp2_ptr(ji,jj))
        limitation_slop(zzslpzpl_ptr(ji,jj), zzslpzpl, zzwzp2_ptr(ji,jj), zzwzp1_ptr(ji,jj))
        vertical_adv_flux(zzwzfp1_ptr(ji,jj), jk+1, zzslpz_ptr(ji,jj), zzslpzpl_ptr(ji,jj))
      END DO
    END DO

    DO jj = nldj, nlej
      DO ji = nldi, nlei
        !!! Horizontal advection flux (x-axes) !!!
        initial_slop_i(zxm2, ji-2); initial_slop_i(zxm1, ji-1)
        initial_slop_i(zzx, ji); initial_slop_i(zxpl, ji+1)
        tracer_slop(zzslpxm1, zzxml, zxm2)
        tracer_slop(zzslpx, zzx, zxm1)
        tracer_slop(zzslpxp1, zxpl, zzx)
        limitation_slop(zzslpxm1, zzslpxm1, zxm2, zxm1)
        limitation_slop(zzslpx, zzslpx, zxm1, zzx)
        limitation_slop(zzslpxp1, zzslpxp1, zzx, zxpl)
        vertical_adv_flux_i(zzxf, ji, zzslpx, zzslpxp1)
        vertical_adv_flux_i(zzxfm1, ji-1, zzslpxm1, zzslpx)

        !!! Horizontal advection flux (y-axes) !!!
        initial_slop_j(zzym2, jj-2); initial_slop_j(zzym1, jj-1)
        initial_slop_j(zzy, jj); initial_slop_j(zzyp1, jj+1)
        tracer_slop(zzslpym1, zzym1, zzym2)
        tracer_slop(zzslpy, zzy, zzym1)
        tracer_slop(zzslpypl, zzyp1, zzy)
        limitation_slop(zzslpym1, zzslpym1, zzym2, zzym1)
        limitation_slop(zzslpy, zzslpy, zzym1, zzy)
        limitation_slop(zzslpypl, zzslpypl, zzy, zzyp1)
        vertical_adv_flux_j(zzyf, jj, zzslpy, zzslpypl)
        vertical_adv_flux_j(zzyfm1, jj-1, zzslpym1, zzslpy)

        zbtr = 1. / ( e1e2t(ji,jj) * fse3t(ji,jj,jk) )
        ztra_i = zzxfm1 - zzx
        ztra_j = zzyfm1 - zzy
        ztra_k = zzwzfp1_ptr(ji,jj) - zzwzf_ptr(ji,jj)
        ! add it to the general tracer trends
        pta(ji,jj,jk,jn) = pta(ji,jj,jk,jn) + zbtr * ( ztra_i + ztra_j + ztra_k )
      END DO
    END DO

    tmp => zzwzp1_ptr; zzwzp1_ptr => zzwzp2_ptr; zzwzp2_ptr => tmp
    tmp => zzslpz_ptr; zzslpz_ptr => zzslpzpl_ptr; zzslpzpl_ptr => tmp
    tmp => zzwzf_ptr; zzwzf_ptr => zzwzfp1_ptr; zzwzfp1_ptr => tmp
  END DO
END DO
```

! end of tracer loop

Third approach – prototype3

- Along the horizontal direction:
 - In the `jk` loop, before updating the RHS variable with the advective trend, the fluxes for the whole horizontal domain are computed
 - The fluxes are hence used to compute the advective trend and to update the RHS variable
- This approach further reduces the number of redundant operations.

Third approach – prototype3

Prototype 2

```
CALL mpp_lnk_4d(ptb, 2)
DO jn = 1, kjpt
  ! == loop over the tracers ==!
  DO jj = nldj, nlej
    DO ji = nldi, nlei
      initial_slop(zzwzm1, 2); initial_slop(zzwz, 3); initial_slop(zzwzp1_ptr(ji,jj), 4)
      tracer_slop(zzslpzm1, zzwzm1, zzwz); tracer_slop(zzslpz, zzwz, zzwzp1_ptr(ji,jj))
      limitation_slop(zslpzzm1, zzslpzm1, zzwz, zzwzm1)
      limitation_slop(zslpzz_ptr(ji,jj), zzslpz, zzwzp1_ptr(ji,jj), zzwz)
      vertical_adv_flux(zzwzf_ptr(ji,jj), 3, zslpzm1, zslpzz_ptr(ji,jj))
    END DO
  END DO

  DO jk = 3, jpk-3
    DO jj = nldj, nlej
      DO ji = nldi, nldi
        initial_slop(zzwzp2_ptr(ji,jj), jk+2)
        tracer_slop(zzslpzm1, zzwzp1_ptr(ji,jj), zzwzp2_ptr(ji,jj))
        limitation_slop(zslpzzp1_ptr(ji,jj), zzslpzm1, zzwzp2_ptr(ji,jj), zzwzp1_ptr(ji,jj))
        vertical_adv_flux(zzwzfp1_ptr(ji,jj), jk+1, zslpzz_ptr(ji,jj), zslpzzp1_ptr(ji,jj))
      END DO
    END DO

    DO jj = nldj, nlej
      DO ji = nldi, nlei
        !!! Horizontal advection flux (x-axes) !!!
        initial_slop_i(zzxm2, ji-2); initial_slop_i(zzxm1, ji-1)
        initial_slop_i(zzx, ji); initial_slop_i(zzxp1, ji+1)
        tracer_slop(zzslpxm1, zzxm1, zzxm2)
        tracer_slop(zzslpx, zzx, zzxm1)
        tracer_slop(zzslpxp1, zzxp1, zzx)
        limitation_slop(zzslpxm1, zzslpxm1, zzxm2, zzxm1)
        limitation_slop(zzslpx, zzslpx, zzxm1, zzx)
        limitation_slop(zzslpxp1, zzslpxp1, zzx, zzxp1)
        vertical_adv_flux_i(zzxf, ji, zzslpx, zzslpxp1)
        vertical_adv_flux_i(zzxfm1, ji-1, zzslpxm1, zzslpx)

        !!! Horizontal advection flux (y-axes) !!!
        initial_slop_j(zzym2, jj-2); initial_slop_j(zzym1, jj-1)
        initial_slop_j(zzy, jj); initial_slop_j(zzyp1, jj+1)
        tracer_slop(zzslpym1, zzym1, zzym2)
        tracer_slop(zzslpy, zzy, zzym1)
        tracer_slop(zzslpyp1, zzyp1, zzy)
        limitation_slop(zzslpym1, zzslpym1, zzym2, zzym1)
        limitation_slop(zzslpy, zzslpy, zzym1, zzy)
        limitation_slop(zzslpyp1, zzslpyp1, zzy, zzyp1)
        vertical_adv_flux_j(zzyf, jj, zzslpy, zzslpyp1)
        vertical_adv_flux_j(zzyfm1, jj-1, zzslpym1, zzslpy)

        zbtr = 1. / ( ele2t(ji,jj) * fse3t(ji,jj,jk) )
        ztra_i = zzxfm1 - zzxf
        ztra_j = zzyfm1 - zzyf
        ztra_k = zzwzfp1_ptr(ji,jj) - zzwzf_ptr(ji,jj)
        ! add it to the general tracer trends
        pta(ji,jj,jk,jn) = pta(ji,jj,jk,jn) + zbtr * ( ztra_i + ztra_j + ztra_k )
      END DO
    END DO

    tmp => zzwzp1_ptr; zzwzp1_ptr => zzwzp2_ptr; zzwzp2_ptr => tmp
    tmp => zslpzz_ptr; zslpzz_ptr => zslpzzp1_ptr; zslpzzp1_ptr => tmp
    tmp => zzwzf_ptr; zzwzf_ptr => zzwzfp1_ptr; zzwzfp1_ptr => tmp
  END DO
END DO
! end of tracer loop
```

Prototype 3

```
CALL mpp_lnk_4d(ptb, 2)
DO jn = 1, kjpt
  ! == loop over the tracers ==!
  DO jj = nldj, nlej
    DO ji = nldi, nlei
      initial_slop_k(zzwzm1, 2); initial_slop_k(zzwz, 3); initial_slop_k(zzwzp1_ptr(ji,jj), 4)
      tracer_slop(zzslpzm1, zzwzm1, zzwz); tracer_slop(zzslpz, zzwz, zzwzp1_ptr(ji,jj))
      limitation_slop(zslpzzm1, zzslpzm1, zzwz, zzwzm1)
      limitation_slop(zslpzz_ptr(ji,jj), zzslpz, zzwzp1_ptr(ji,jj), zzwz)
      vertical_adv_flux_k(zzwzf_ptr(ji,jj), 3, zslpzm1, zslpzz_ptr(ji,jj))
    END DO
  END DO

  DO jk = 3, jpk-3
    DO jj = nldj-1, nlej+1
      DO ji = nldi-1, nlei+1
        !!! Horizontal - x slop !!!
        initial_slop_i(zzxm1, ji-1)
        initial_slop_i(zzx, ji)
        tracer_slop(zzslpxs, zzx, zzxm1)
        limitation_slop(zzslpx(ji,jj), zzslpxs, zzxm1, zzx)

        !!! Horizontal - y slop !!!
        initial_slop_j(zzym1, jj-1)
        initial_slop_j(zzy, jj)
        tracer_slop(zzslpys, zzy, zzym1)
        limitation_slop(zzslpy(ji,jj), zzslpys, zzym1, zzy)

        !!! Vertical - z slop !!!
        initial_slop_k(zzwzp2_ptr(ji,jj), jk+2)
        tracer_slop(zzslpzm1, zzwzp1_ptr(ji,jj), zzwzp2_ptr(ji,jj))
        limitation_slop(zslpzzp1_ptr(ji,jj), zzslpzm1, zzwzp2_ptr(ji,jj), zzwzp1_ptr(ji,jj))
        vertical_adv_flux_k(zzwzfp1_ptr(ji,jj), jk+1, zslpzz_ptr(ji,jj), zslpzzp1_ptr(ji,jj))
      END DO
    END DO

    !!! Horizontal advection flux !!!
    DO jj = nldj-1, nlej
      DO ji = nldi-1, nlei
        vertical_adv_flux_i(zzxf(ji,jj), ji, zzslpx(ji,jj), zzslpx(ji+1,jj))
        vertical_adv_flux_j(zzyf(ji,jj), jj, zzslpy(ji,jj), zzslpy(ji,jj+1))
      END DO
    END DO

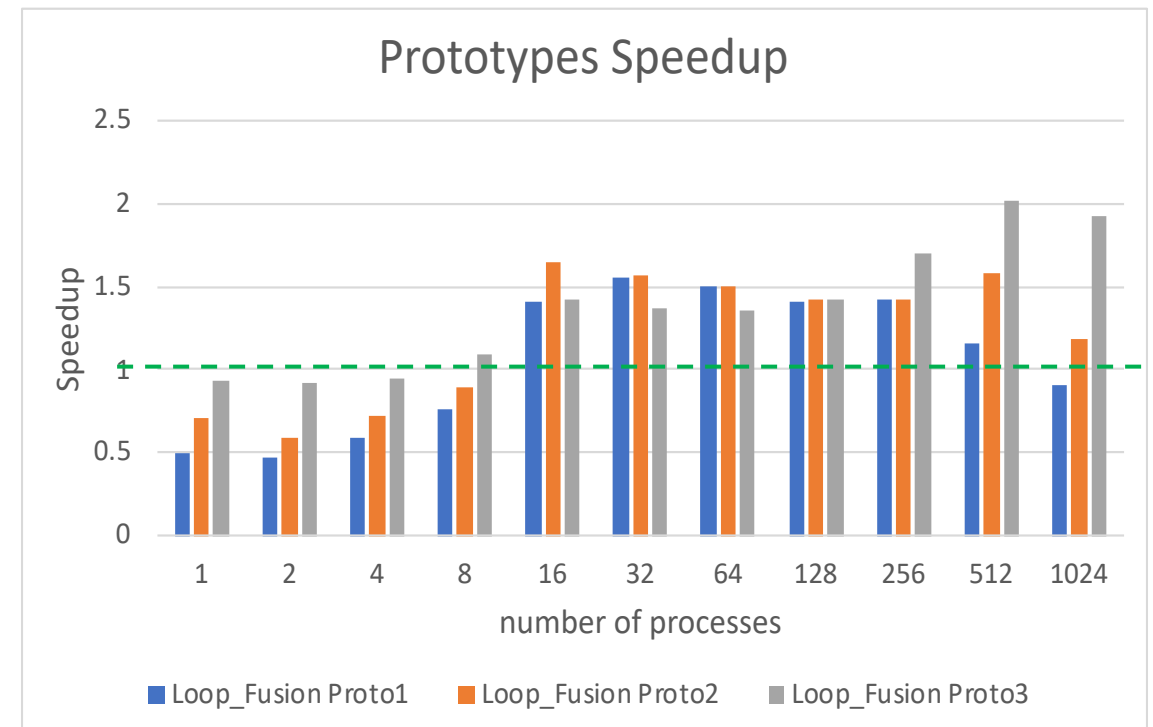
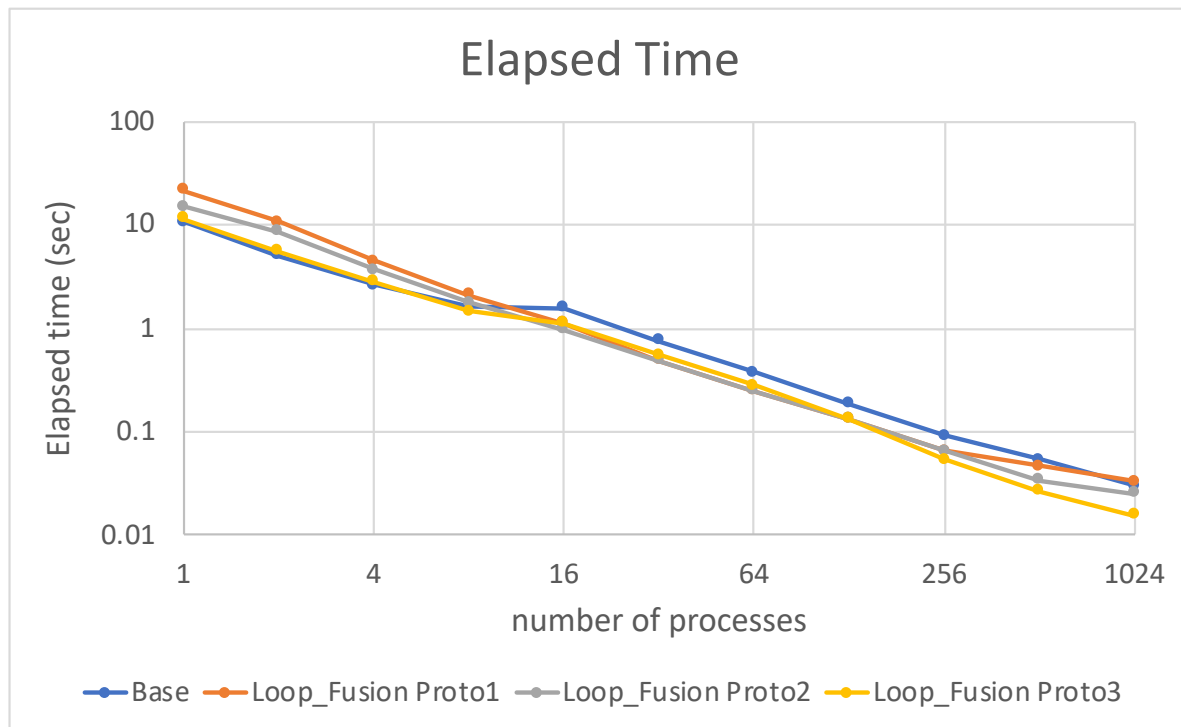
    DO jj = nldj, nlej
      DO ji = nldi, nlei
        zbtr = 1. / ( ele2t(ji,jj) * fse3t(ji,jj,jk) )
        ztra_i = zzxf(ji-1,jj) - zzxf(ji,jj)
        ztra_j = zzyf(ji,jj-1) - zzyf(ji,jj)
        ztra_k = zzwzfp1_ptr(ji,jj) - zzwzf_ptr(ji,jj)
        ! add it to the general tracer trends
        pta(ji,jj,jk,jn) = pta(ji,jj,jk,jn) + zbtr * ( ztra_i + ztra_j + ztra_k )
      END DO
    END DO

    tmp => zzwzp1_ptr; zzwzp1_ptr => zzwzp2_ptr; zzwzp2_ptr => tmp
    tmp => zslpzz_ptr; zslpzz_ptr => zslpzzp1_ptr; zslpzzp1_ptr => tmp
    tmp => zzwzf_ptr; zzwzf_ptr => zzwzfp1_ptr; zzwzfp1_ptr => tmp
  END DO
END DO
! loop over k
END DO
! loop over jn
```

Preliminary Performance Analysis

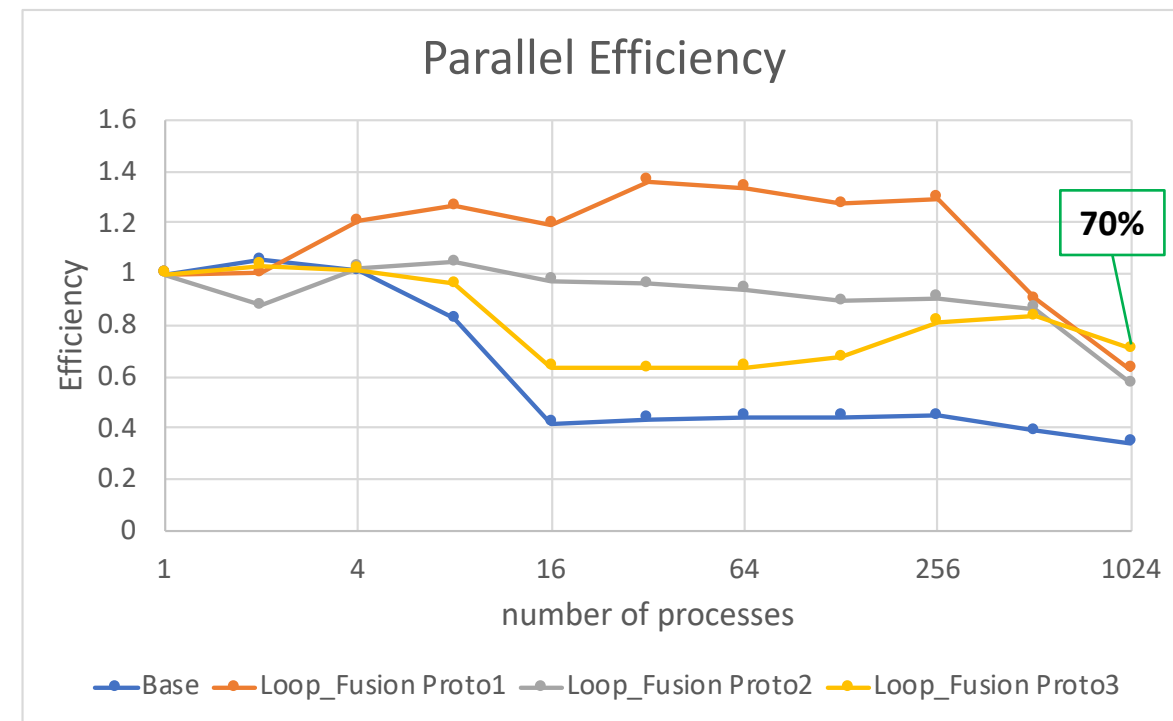
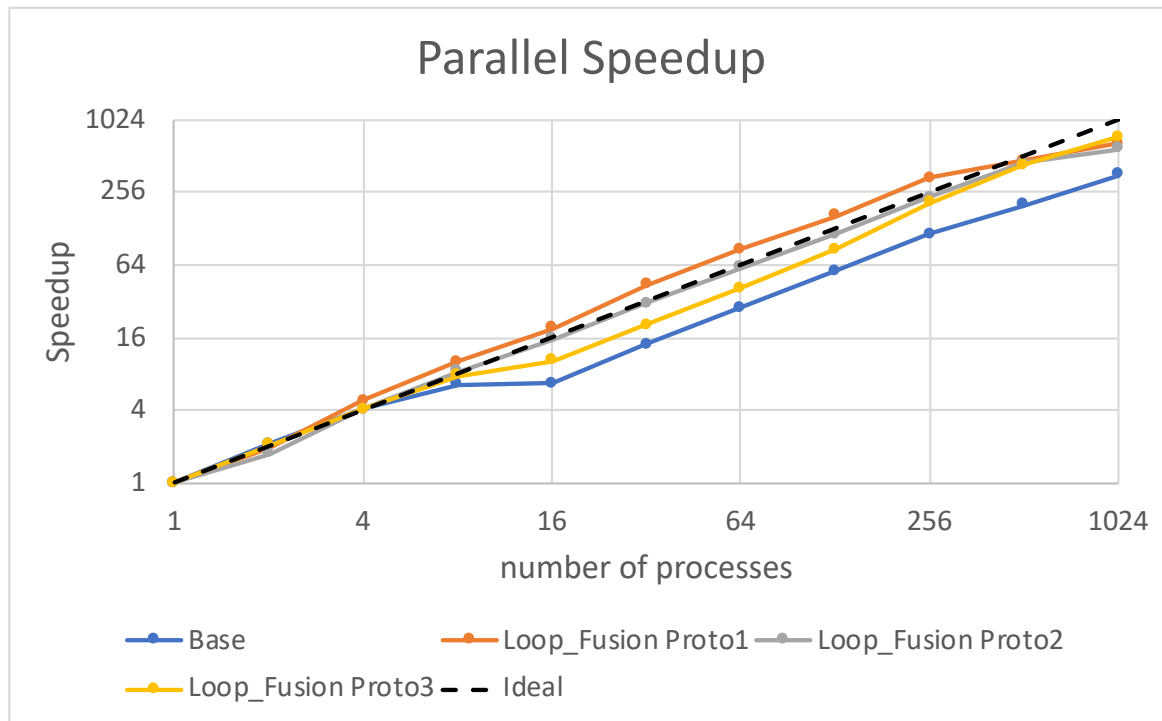
- Global domain: 2240 x 1500 x 31 points
- Smallest sub-domain (with 1024 cores): 74 x 51 x 31 points

*Test executed on Intel
Xeon based Architecture
16 cores per node*



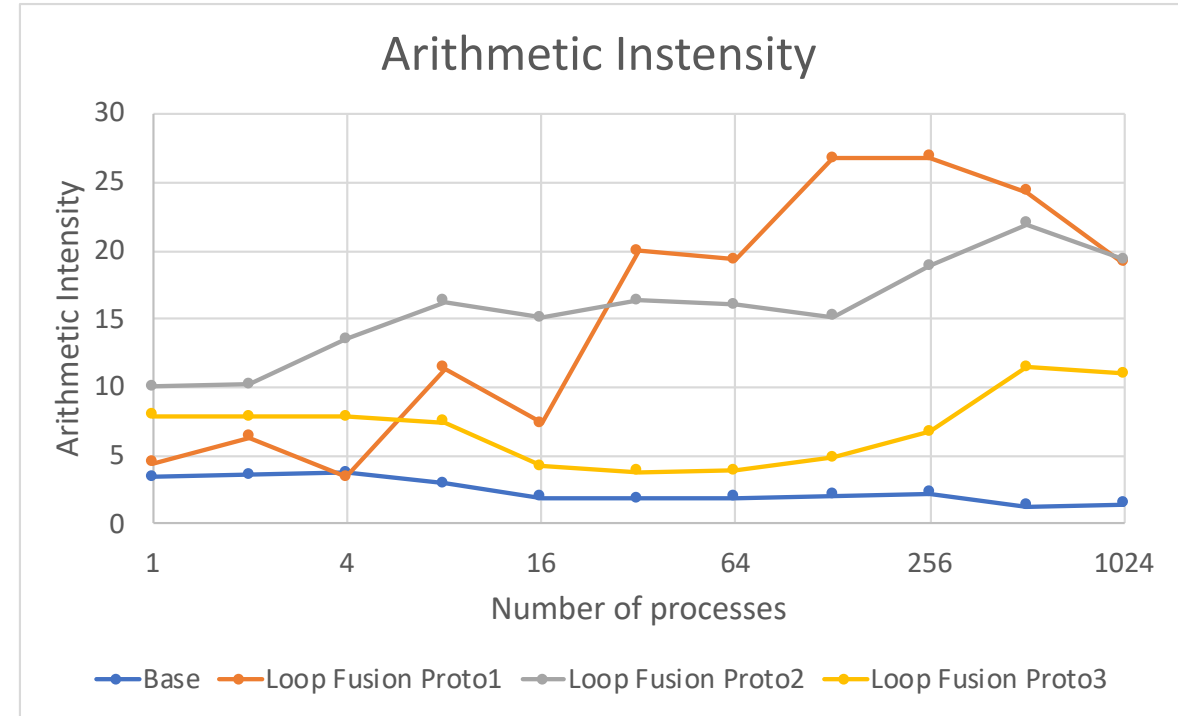
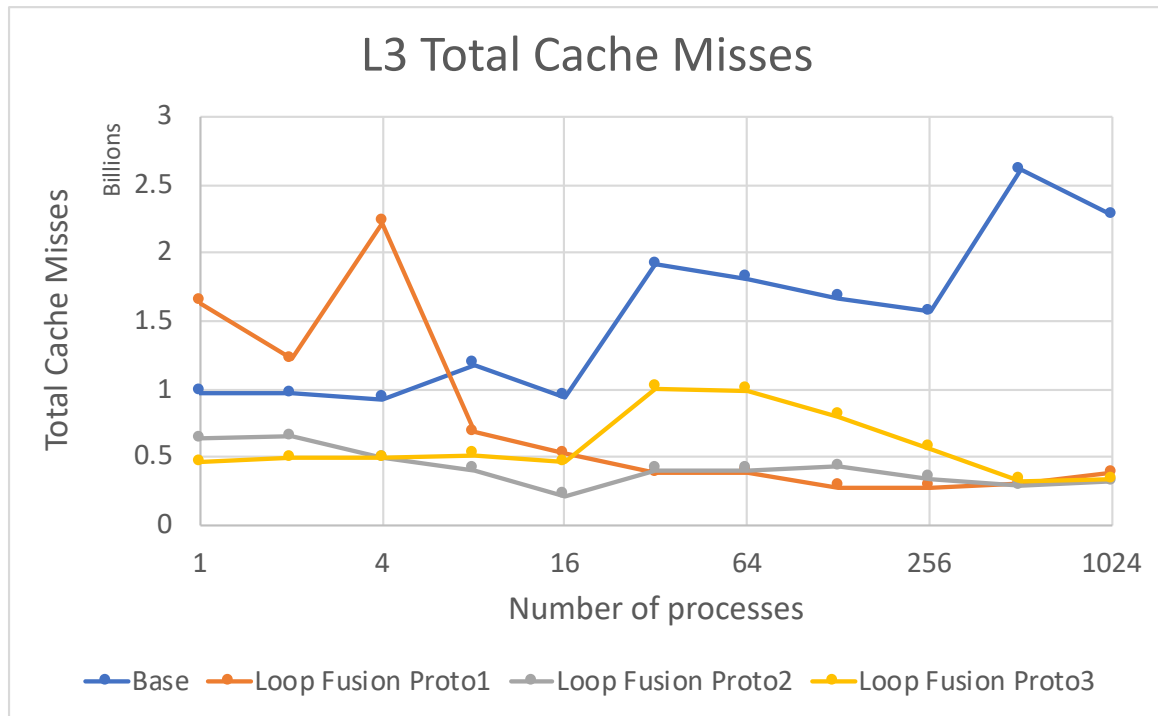
Preliminary Performance Analysis

- Global domain: 2240 x 1500 x 31 points
- Smallest sub-domain (with 1024 cores): 74 x 51 x 31 points



Preliminary Performance Analysis

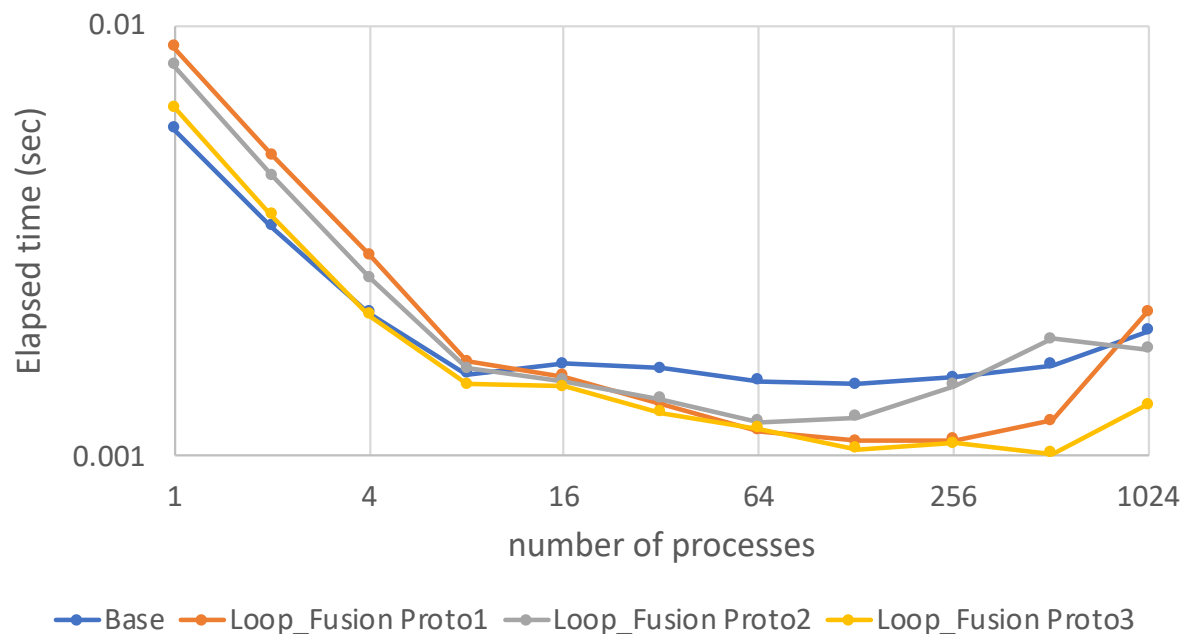
- Global domain: 2240 x 1500 x 31 points
- Smallest sub-domain (with 1024 cores): 74 x 51 x 31 points



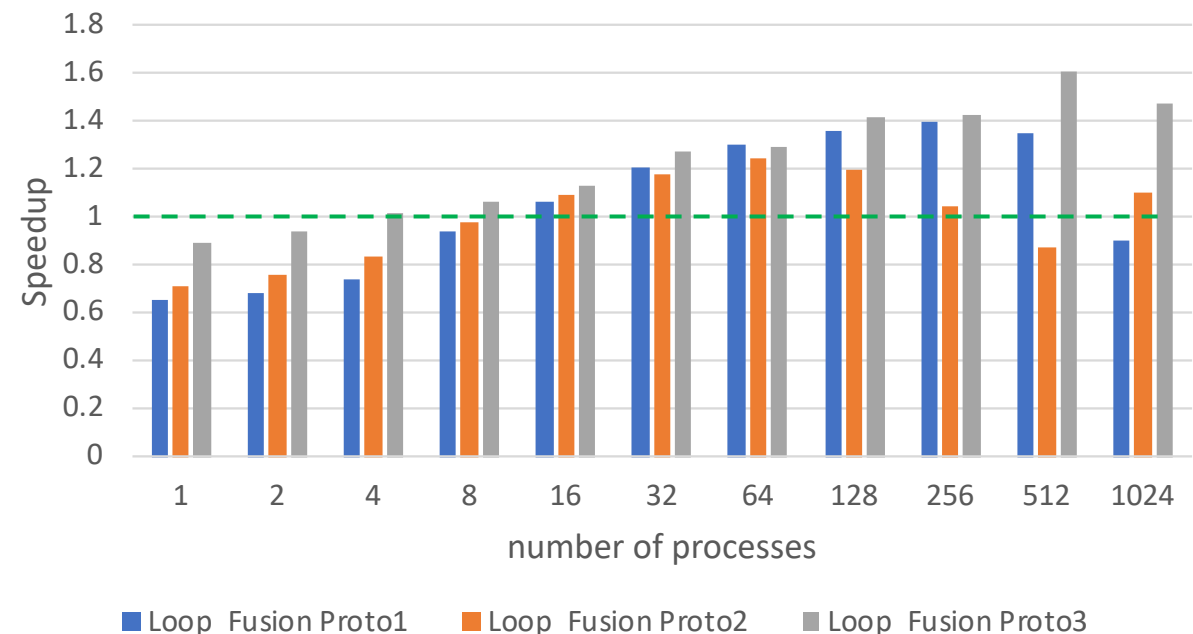
Preliminary Performance Analysis

- Global domain: 70 x 46 x 19 points
- Sub-domain with 64 cores: 13 x 10 x 19 points

Elapsed Time

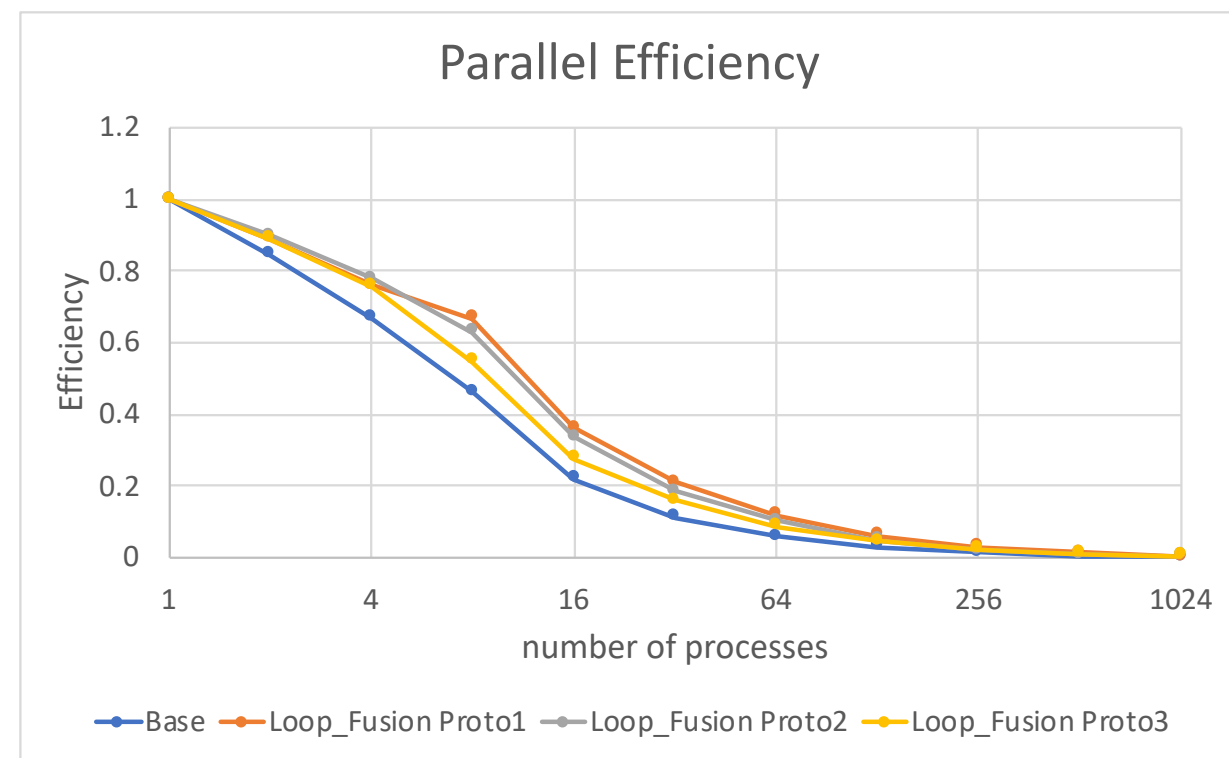
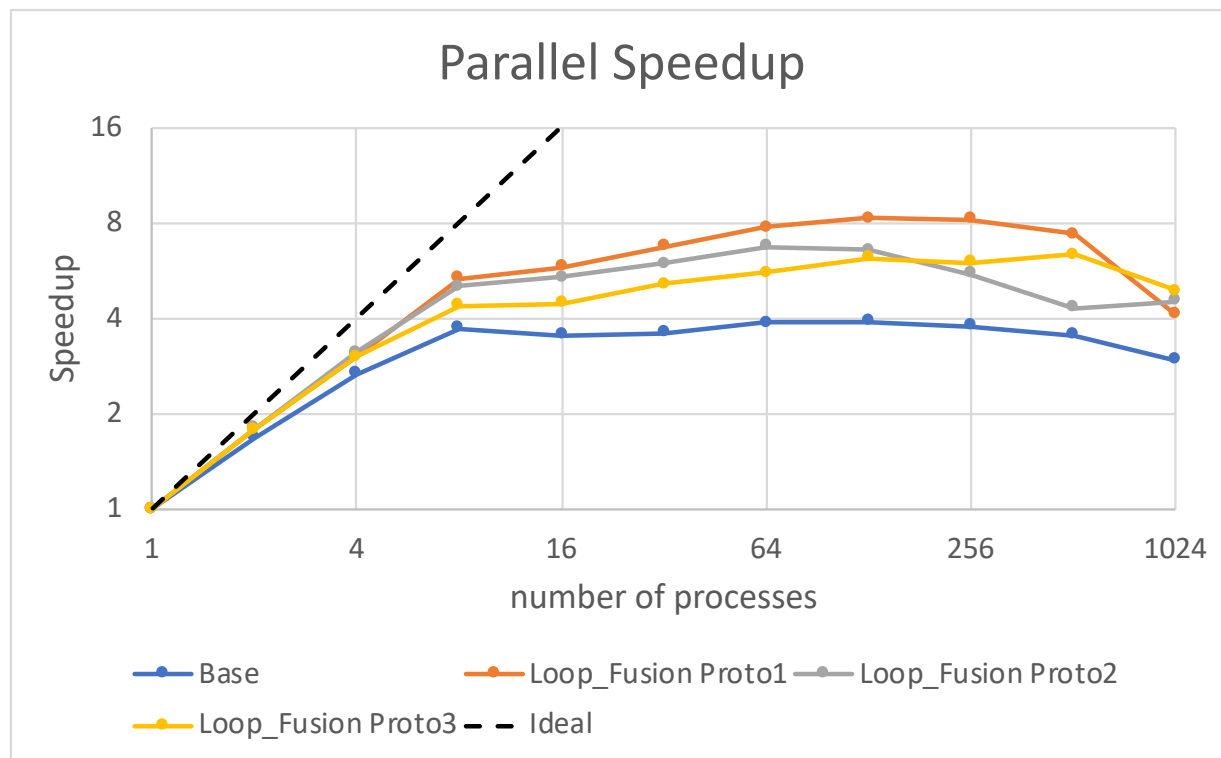


Prototypes Speedup



Preliminary Performance Analysis

- Global domain: 70 x 46 x 19 points
- Sub-domain with 64 cores: 13 x 10 x 19 points



Preliminary Performance Analysis

- Prototypes 1 and 2 provide a good improvement up to 256 cores then the redundant operations lead to a loss of performance
- Prototypes 3 improves parallel efficiency by ~30% on 1024 cores
- This approach enhanced the vectorization level and the cache reuse, reducing L3 Total Cache Misses by ~80% on 1024 cores
- Loop-fusion is strictly linked to the computing architecture → A fully portable performance improvement can be ensured by the adoption of a DSL.