Accelerating the Cloud Scheme Within the Unified Model for CPU-GPU Based High-Performance Computing Systems

Wei Zhang, Min Xu, Mario Morales Hernandez,

Matthew Norman, Salil Mahajan and Katherine Evans

2019 MultiCore 9 Workshop, Sep 25th 2019

ORNL is managed by UT-Battelle, LLC for the US Department of Energy

Thanks to the US Air Force, DOE OLCF, Met Office and EPCC for support and help!

Content

- Overview
  - Introduction of project and motivation
  - CASIM cloud scheme
  - OLCF Summit supercomputer

- CASIM on Summit, from CPU to GPU: current status and future plans

Overview

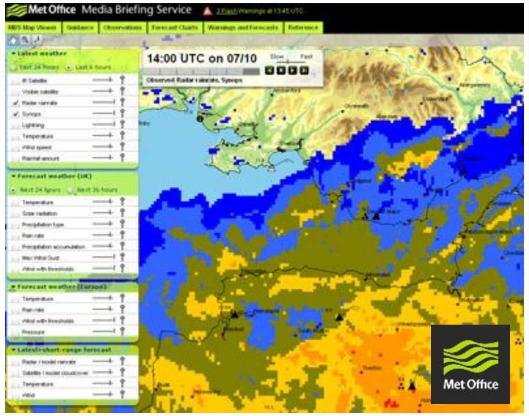
Forecast Extension to Hydrology – From Rainfall to Flood

Rainfall
- Weather Forecasting Models

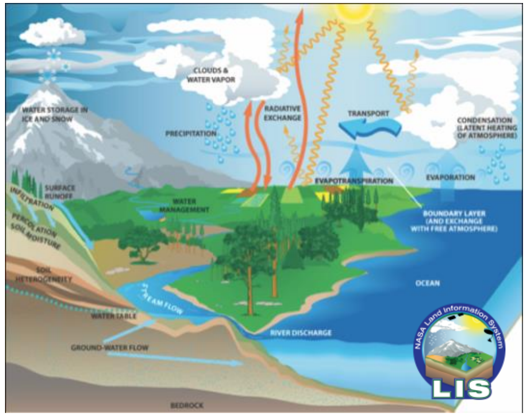
Runoff
- **NASA**
- Land Information System (LIS)

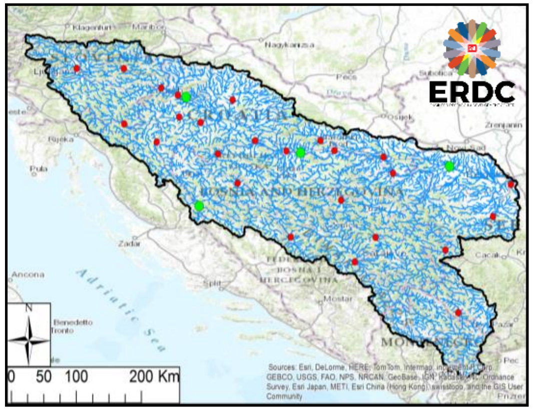
Streamflow
- **ERDC**
- Streamflow Prediction System (SPT)

Inundation
- **ORNL/TTU**
- TRITON-GPU

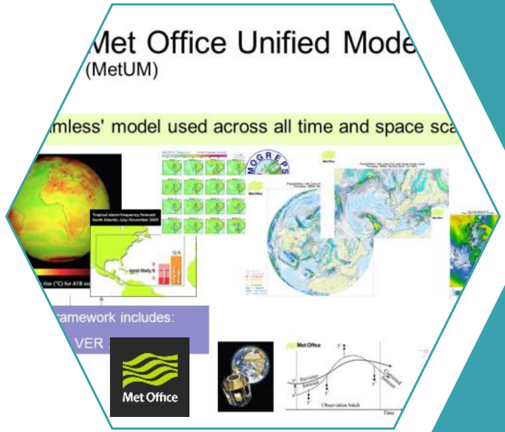


UM Optimization (Cloud Scheme, Radiation Scheme)





Overview



Air Force Weather and ORNL Collaboration

Met Office Unified Model (UM)

Cloud AeroSol Interacting Microphysics (CASIM)

OLCF Summit

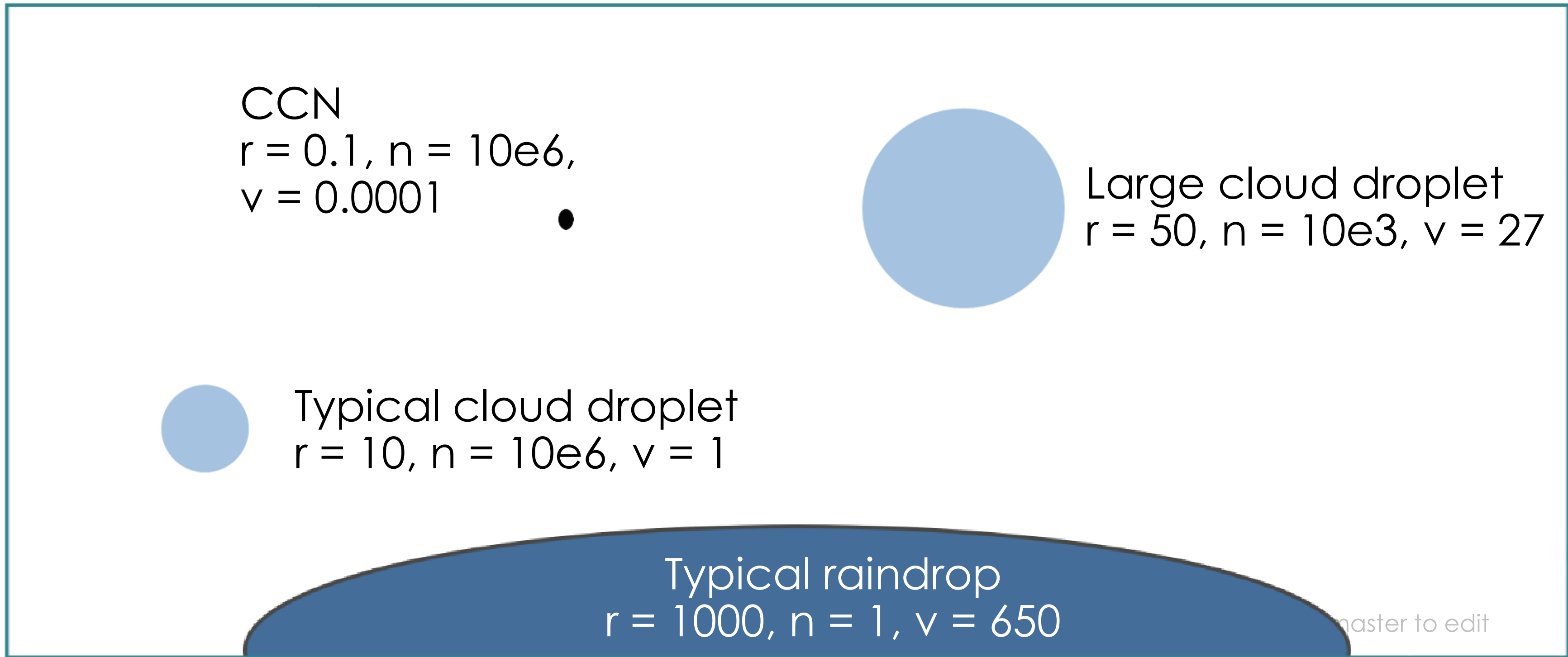
What is cloud microphysics?

Cloud microphysics concerns the mechanisms by which cloud droplets generated from water vapor and the particles in the air, and grow to form raindrops, ice and snow.

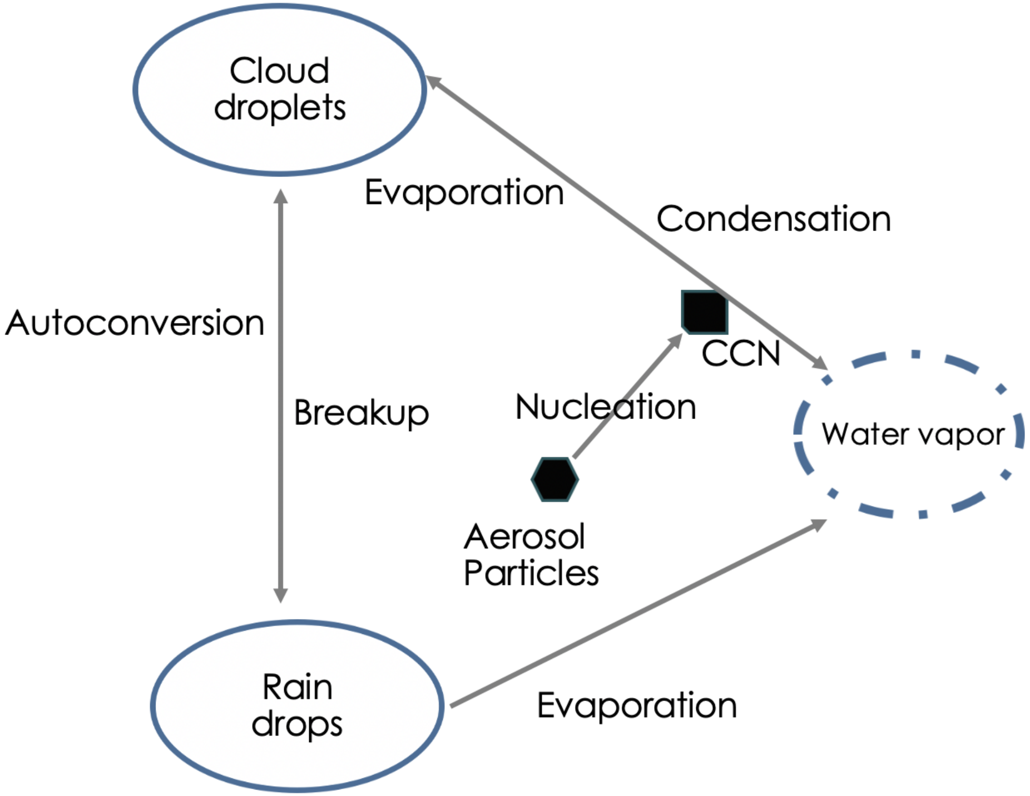
-- John M. Wallace, Peter V. Hobbs, in Atmospheric Science (Second Edition), 2006

- Relative sizes of cloud droplets, raindrops and cloud condensation nuclei (CCN)

r: radius (um)
n: number per liter of air
v: fall speed (cm/s)



Why cloud microphysics matters?



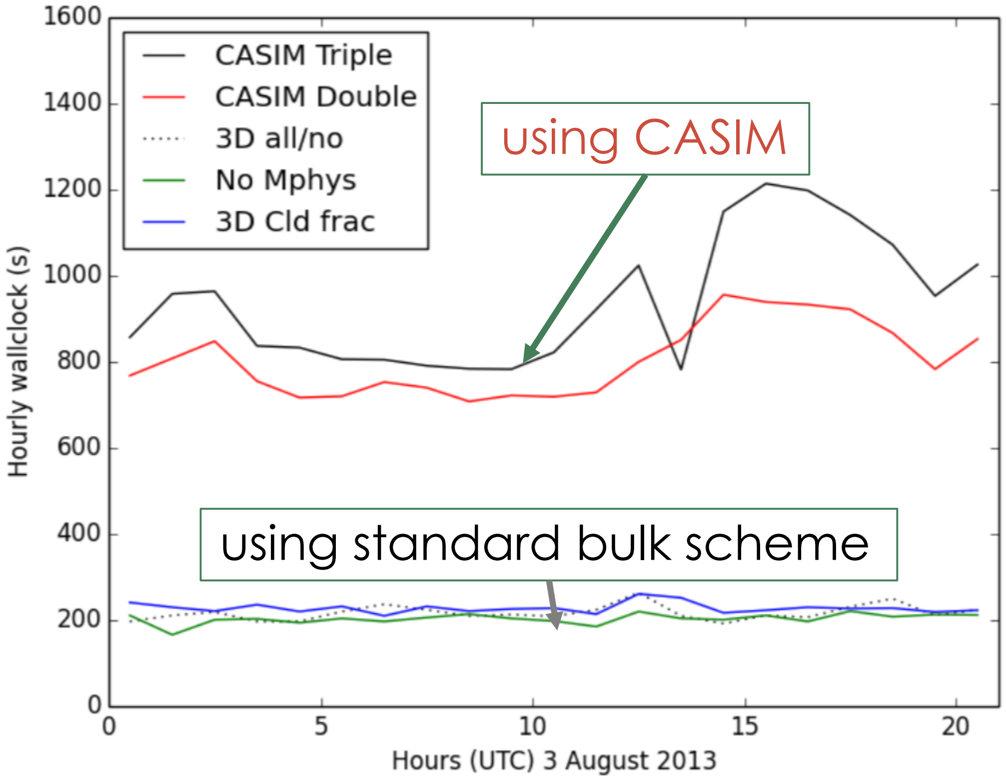
- Schematics of some of the warm cloud and precipitation microphysical processes

- The evolution of cloud/rain mass, the number concentration of droplets and particles
- Latent heating/cooling, Temperature
  - condensation, evaporation, deposition, sublimation, freezing, melting
- Affecting surface processes, radiative transfer, cloud-aerosol-precipitation interactions…

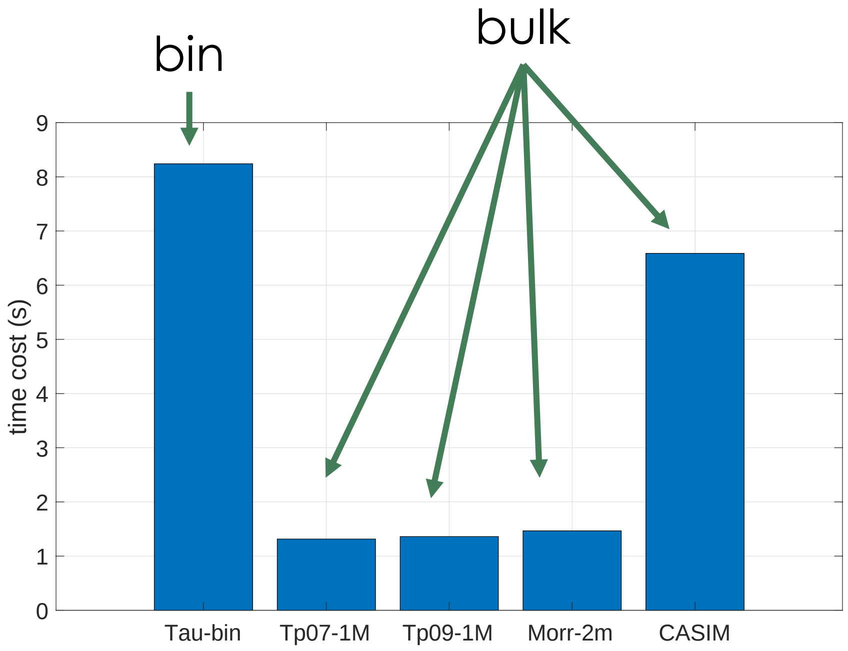
Cloud AeroSol Interacting Microphysics - **CASIM**

- Long-term replacement for UM microphysics and the default microphysics
- User definable
  - number of cloud species (e.g., cloud, rain, ice, snow, graupel)
  - number of moments to describe each species (1 - mass, 2: 1 + number, 3: 2 +radar reflectivity)
- Detailed representation of aerosol effects and in-cloud processing of aerosol
  - increase accuracy
  - more intensive calculation

- CASIM/src
  - Modern Fortran code
  - 16329 total lines, 116 subroutines



(Run in UM, same COPE case, different microphysics schemes, adopted from Met office technical paper)

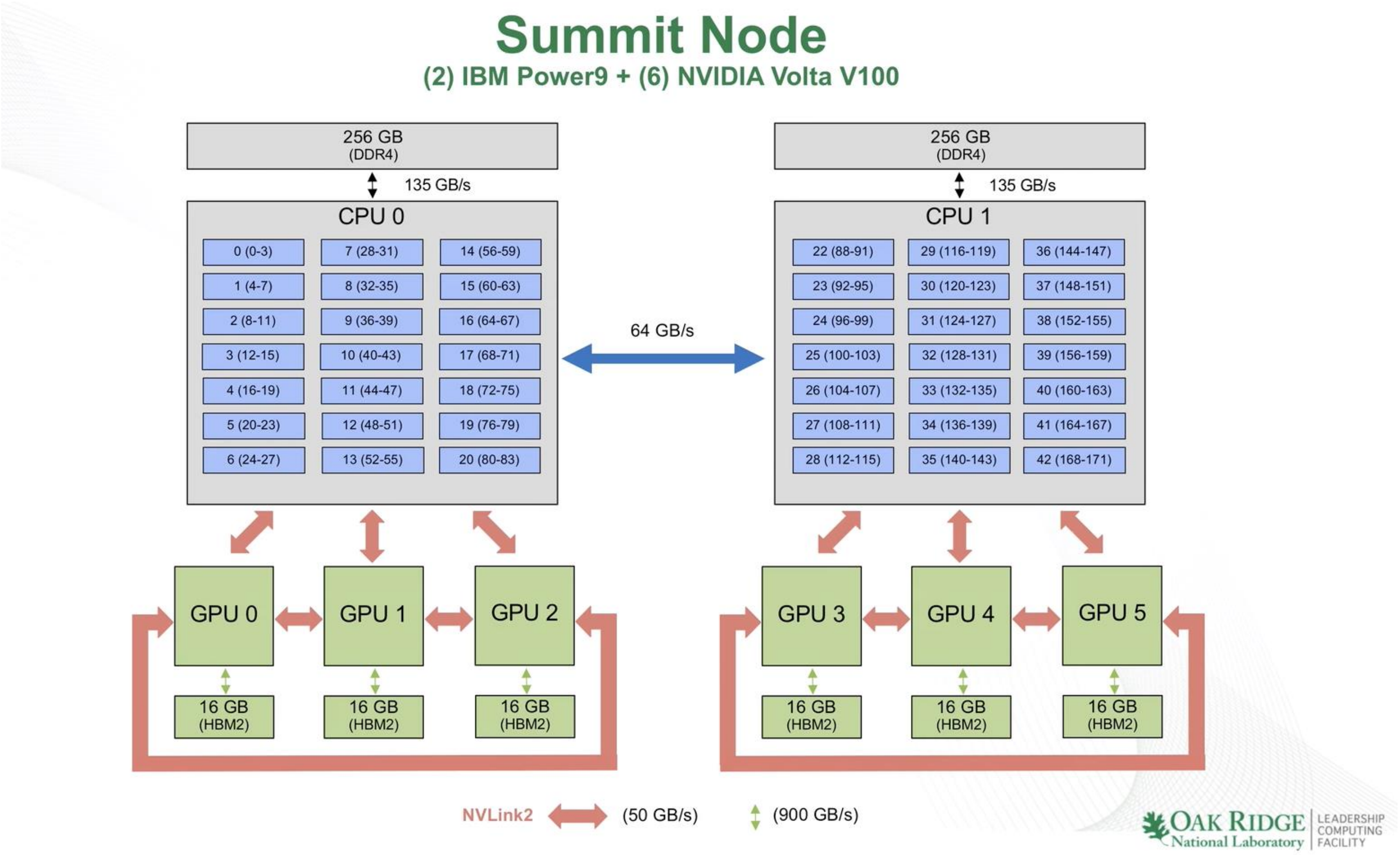


Wallclock for KiD_1D Simulations on Summit (no parallelism)

(Same model, same cumulus case, different microphysics schemes)

HPC + GPU Computing

Oak Ridge Leadership Computing Facility (**OLCF**) Summit



OAK RIDGE
National Laboratory

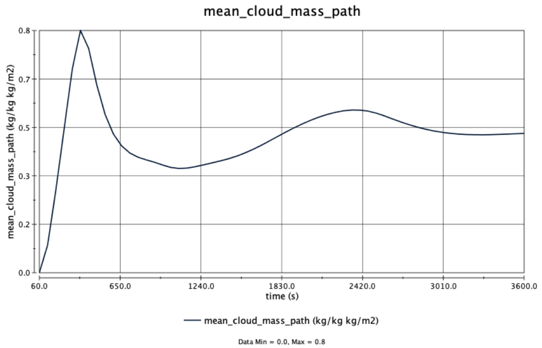
Open slide master to edit

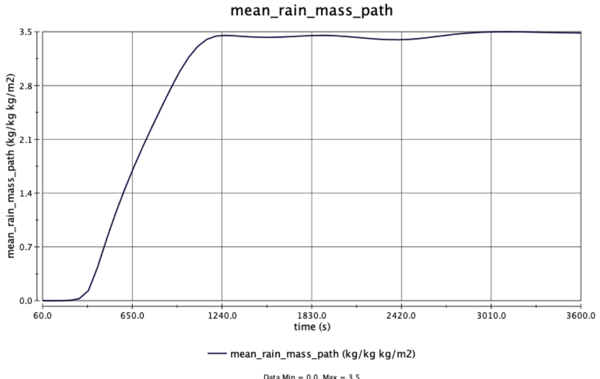
- Objects
  - Applying new coding to CASIM for GPUs
  - Developing algorithms that will be suited for accelerated machines (Summit now, Frontier, in the future)
- Compilers
  - PGI (19.7 on Summit)
  - Cray (will be available when Frontier comes out)
  - CLAW (source-to-source translator, Produces code for the target architectures and directive languages, https://github.com/claw-project/claw-compiler)
- Directive
  - OpenACC
- Considerations
  - Portability limitations, CPU-GPU communication
  - Validation & Verification, Robust testing
  - The software stack for these new computing systems

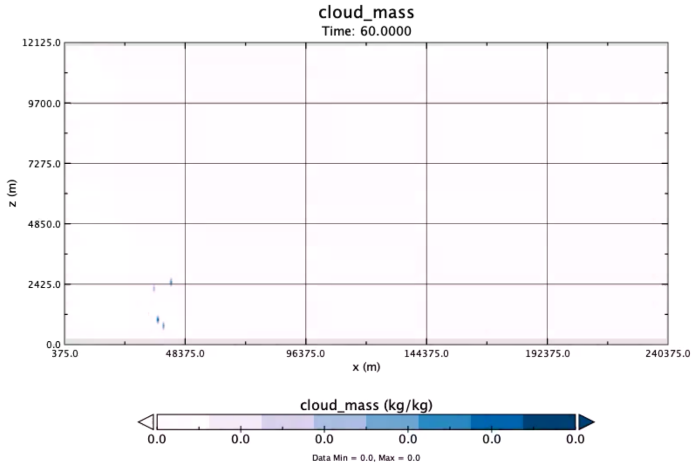
CASIM on Summit

- Parent model: The Kinematic Driver Model (**KiD**, Shipway and Hill, 2011)
  - Kinematic framework to constrain the dynamics and isolate the microphysics
  - Original KiD has no parallelization directives

- Baseline case: 2D squall line case
  - nx = 320, dx = 750 m, nz = 48, dz = 250 m
  - dt = 1 s, t_total = 3600 s, output saved every 60 s







- Step 1. Access KiD-CASIM 2D-SQUALL Performance on CPU
  - Profiling tool: General Purpose Timing Library (**GPTL**) https://jmrosinski.github.io/GPTL/

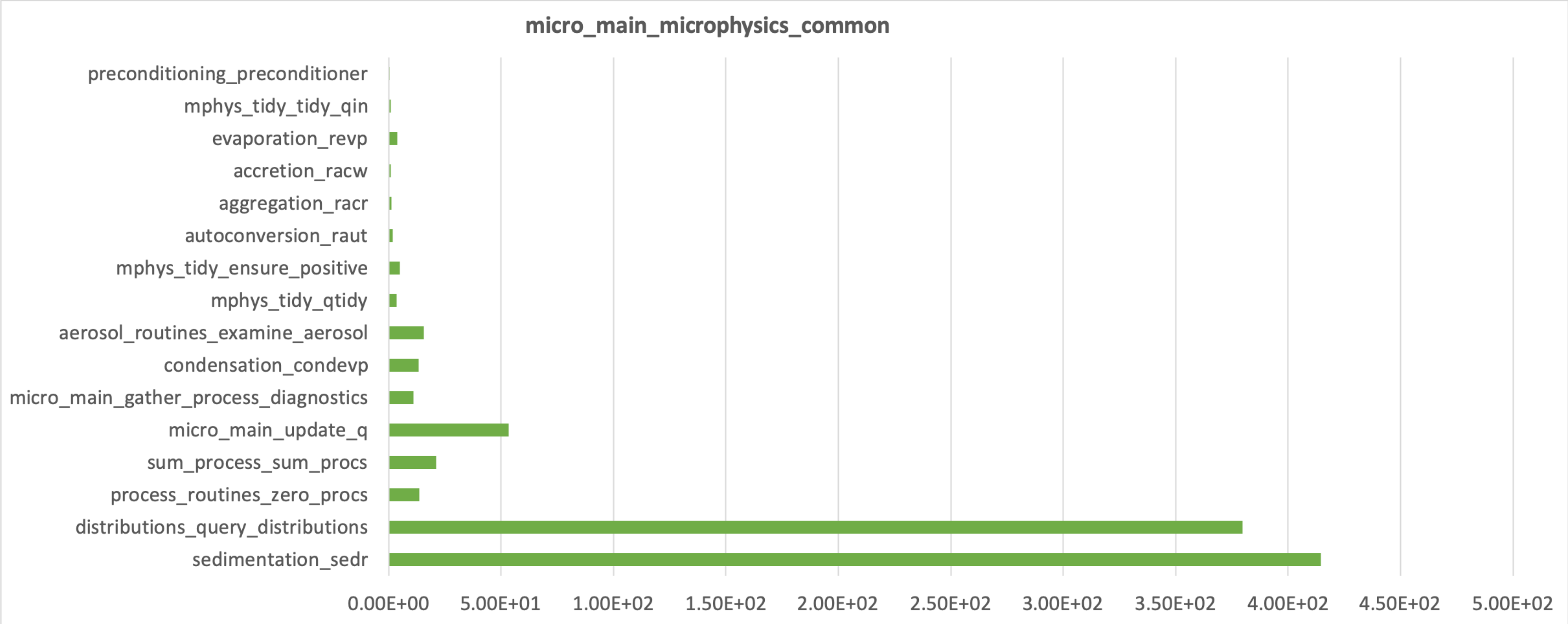
	Called	Recurse	Wallclock	max	min	self_OH	parent_OH
total	1	-	1187.963	1187.963	1187.963	0.000	0.000
main_main_loop_	1	-	1187.963	1187.963	1187.963	0.000	0.000

CASIM in KiD: 1019.095/1187.963 = 85.79%

mphys_interface_mphys_column_	3600	-	1019.095	0.784	0.071	0.000	0.000
mphys_casim_mphys_casim_interface_	3600	-	1019.088	0.784	0.071	0.000	0.000

micro_main in CASIM: 987.515/1019.095 = 96.90%

micro_main_shipway_microphysics_	3600	-	987.515	0.767	0.063	0.000	0.000
process_routines_zero_procs_	1.2e+07	-	13.576	1.80e-03	1.00e-06	0.288	0.899
passive_fields_set_passive_fields_	1.2e+06	-	16.373	1.44e-03	1.30e-05	0.029	0.090
qsat_funs_qsaturation_	1.7e+08	-	10.206	2.41e-03	0.00e+00	4.147	12.939
micro_main_microphysics_common_	1.2e+06	-	966.310	6.96e-03	1.78e-04	0.029	0.090
process_routines_zero_procs_	1.2e+07	-	13.576	1.80e-03	1.00e-06	0.288	0.899
mphys_tidy_tidy_qin_	1.2e+06	-	0.813	1.01e-04	0.00e+00	0.029	0.090
mphys_tidy_recompute_qin_constants_	2	-	3.00e-06	3.00e-06	0.00e+00	0.000	0.000
distributions_query_distributions_	2.5e+07	-	379.726	2.65e-03	1.00e-06	0.634	1.977
lookup_get_slope_generic_	4.0e+08	-	302.323	2.61e-03	0.00e+00	10.071	31.422
lookup_get_lam_n0_2m_	4.0e+08	-	256.817	2.61e-03	0.00e+00	10.071	31.422
special_gammalookup_	3.8e+09	-	172.311	0.168	0.00e+00	94.951	296.246


micro_main_microphysics_common
preconditioning_preconditioner
mphys_tidy_tidy_qin
evaporation_revp
accretion_racw
aggregation_racr
autoconversion_raut
mphys_tidy_ensure_positive
mphys_tidy_qtidy
aerosol_routines_examine_aerosol
condensation_condevp
micro_main_gather_process_diagnostics
micro_main_update_q
sum_process_sum_procs
process_routines_zero_procs
distributions_query_distributions
sedimentation_sedr
0.00E+00
5.00E+01
1.00E+02
1.50E+02
2.00E+02
2.50E+02
3.00E+02
3.50E+02
4.00E+02
4.50E+02
5.00E+02

- Step 2: Get CASIM ready for GPU (ongoing)

- General idea:
  - Optimize most time-consuming parts
  - Avoid/minimize data transfer between CPU and GPU

Idealized solution: GPU region sandwiched between two CPU calculation regions

but ….

```
do i = is, ie
    do j = js, je
        call cpu_calculation1()
    end do
end do
!---------------------------------------------------------------
do i = is, ie
    do j = js, je
        call gpu_calculation()
    end do
end do
!---------------------------------------------------------------
do i = is, ie
    do j = js, je
        call cpu_calculation2()
    end do
end do
```

– Challenge 1: Derived Data Type

1) -ta=tesla:deepcopy (testing)
2) change to flat array (bit-for-bit on CPU confirmed)

```fortran
type :: process_rate
        real(wp), allocatable :: column_data(:)
end type process_rate
…
type(process_rate), allocatable :: procs(:,:)
…
allocate(procs(ntotalq, nprocs))
…
call micro_common(…, procs, …)
```

→

```fortran
type :: process_rate
        real(wp), pointer :: column_data(:)
end type process_rate
…
type(process_rate), allocatable :: procs(:,:)
real(wp), target, allocatable :: procs_flat(:,:,:)
…
allocate(procs(ntotalq, nprocs)
allocate(procs_flat(nz, ntotalq, nprocs)
do iprocs=1, nprocs
     do iq=1, ntotalq
          procs(iq, iprocs)%column_data => &
                procs_flat(1:nz, iq, iprocs)
      end do
end do
call micro_common(…, procs_flat, …)
```

3 levels of nested loops

– Challenge 2:

n-loop and k-loop are not parallelable now;

Hotspots locate deep in the call tree

```
do i = is, ie
   do j = js, je
      ...
      do n = 1, nsubsteps
         ...
         !! early exit if no hydrometeors and subsaturated
         if (.not. any(precondition(:))) exit
         !! do the business
         do k = 1, nz
            ...
            ...
         end do !! k
         ...
      end do !! n
      ...
   end do !! j
end do !! i
```

Not parallelable

Hotspots and vertical dependence

- Former work done in EPCC and UK Met Office:
  - *Porting the microphysics model CASIM to GPU and KNL Cray machines (Brown et al., 2016)*

- Parent model: the Met Office NERC Cloud Model (**MONC)**
- Compiler: Cray
- Directive: OpenACC
- Offloaded the whole CASIM onto GPU on Piz Daint XC50 and XC30

```
subroutine CASIM()
  !$acc parallel async(ACC_QUEUE)
  !$acc loop collapse(2) gang worker
     vector
  do i = is, ie
    do j = js, je
      call microphysics_common(i,j, ...)
    end do
  end do
  !$acc end loop
  !$acc end parallel
end subroutine CASIM

subroutine microphysics_common(i,j, ...)
  !$acc routine seq

  ...
end subroutine microphysics_common
```

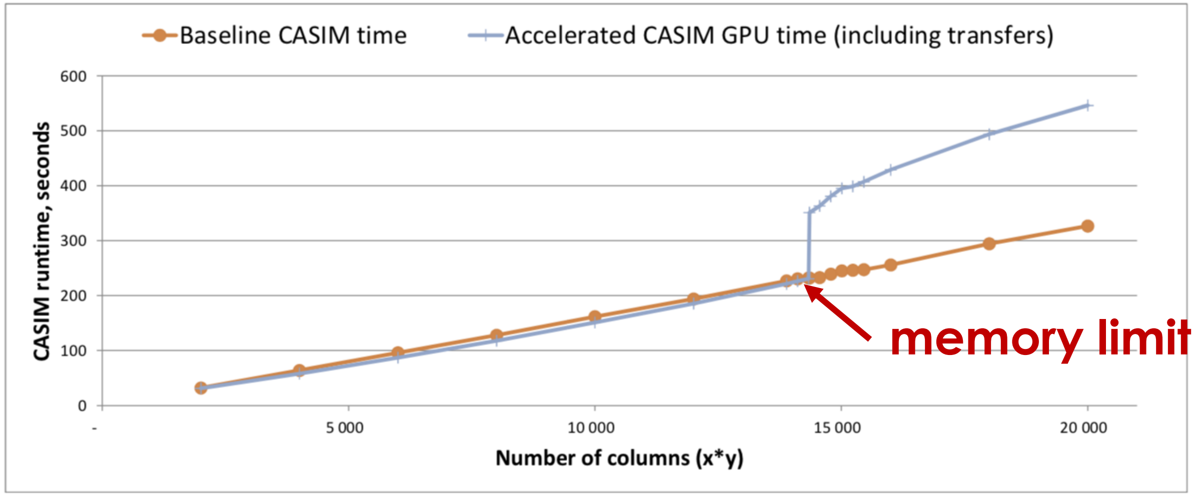


Figure 5.1: Running time measurements for the CASIM's accelerated and CPU-only versions. Data for the accelerated version includes both the computation time on GPU and the time spent on host-GPU memory transfers.

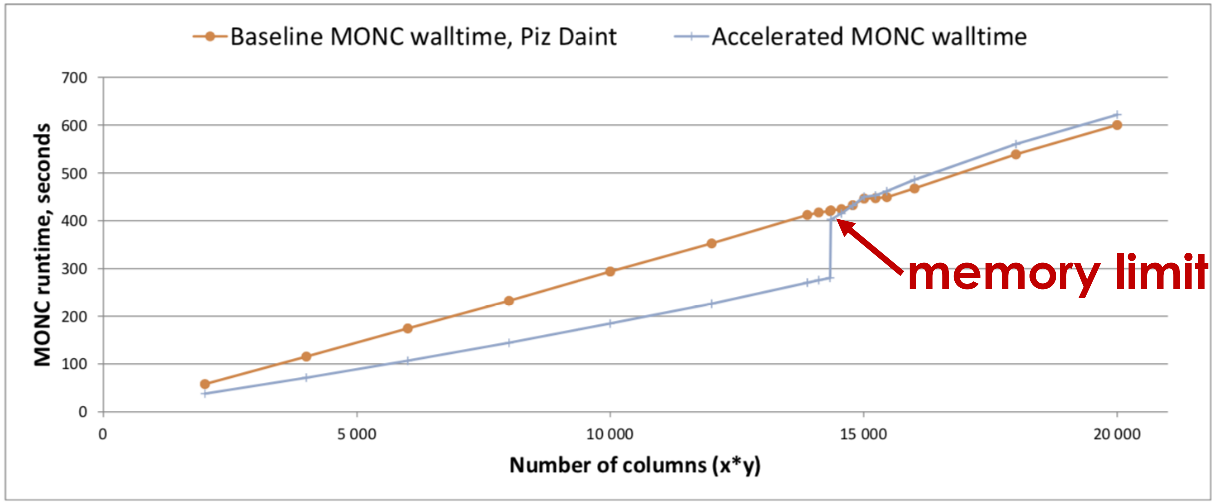


Figure 5.4: MONC running time versus the number of grid columns.

Lesson we learned: Much more code refactoring is needed to

- Maximize the number of parallelization in GPU
- Minimize the amount of data transfer between CPU and GPU

From: *Accelerating the microphysics model CASIM using OpenACC, Alexandr Nigay, 2016*

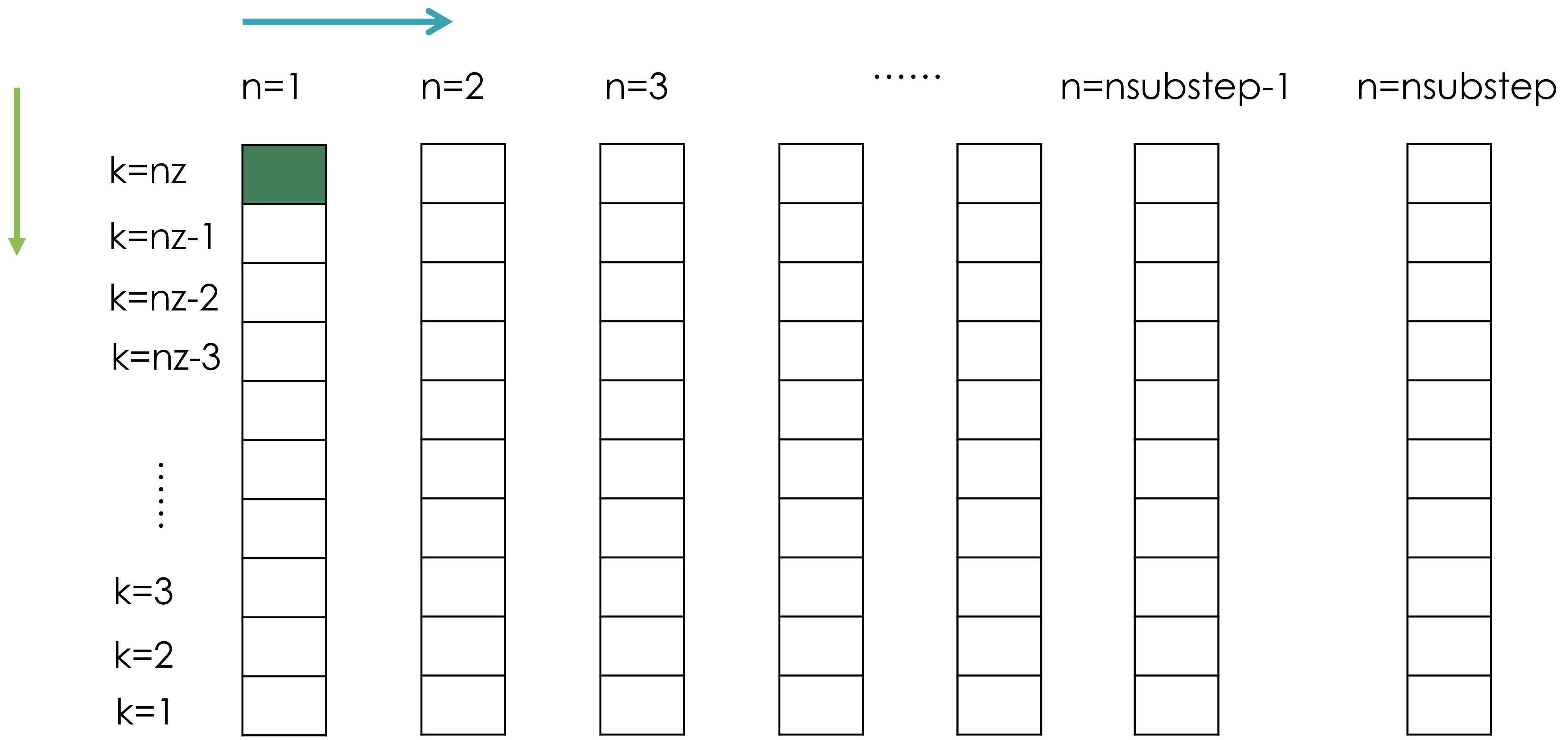
? How to increase the parallelization?

```
do n = 1, nsubsteps

    if (.not. any(precondition(:))) exit

    call function(qfields(1:nz))
    update qfields(1:nz)

end do !! n
```

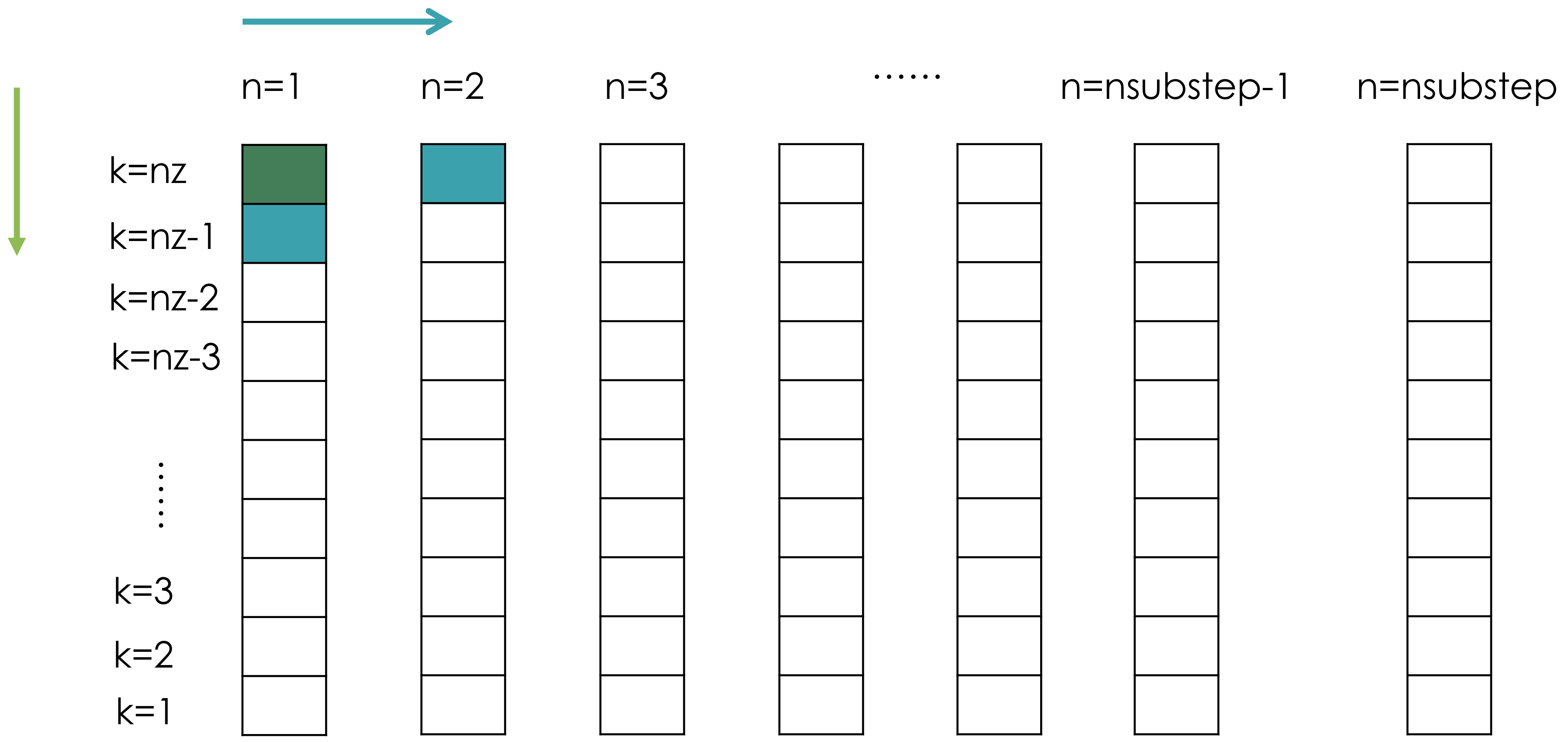
```
do k = nz-1, 1
    ...
    flux(k) = functions(flux(k+1))
    ...
end do !! k
```

$$Flux_n^k = f(Flux_{n-1}^{k+1})$$

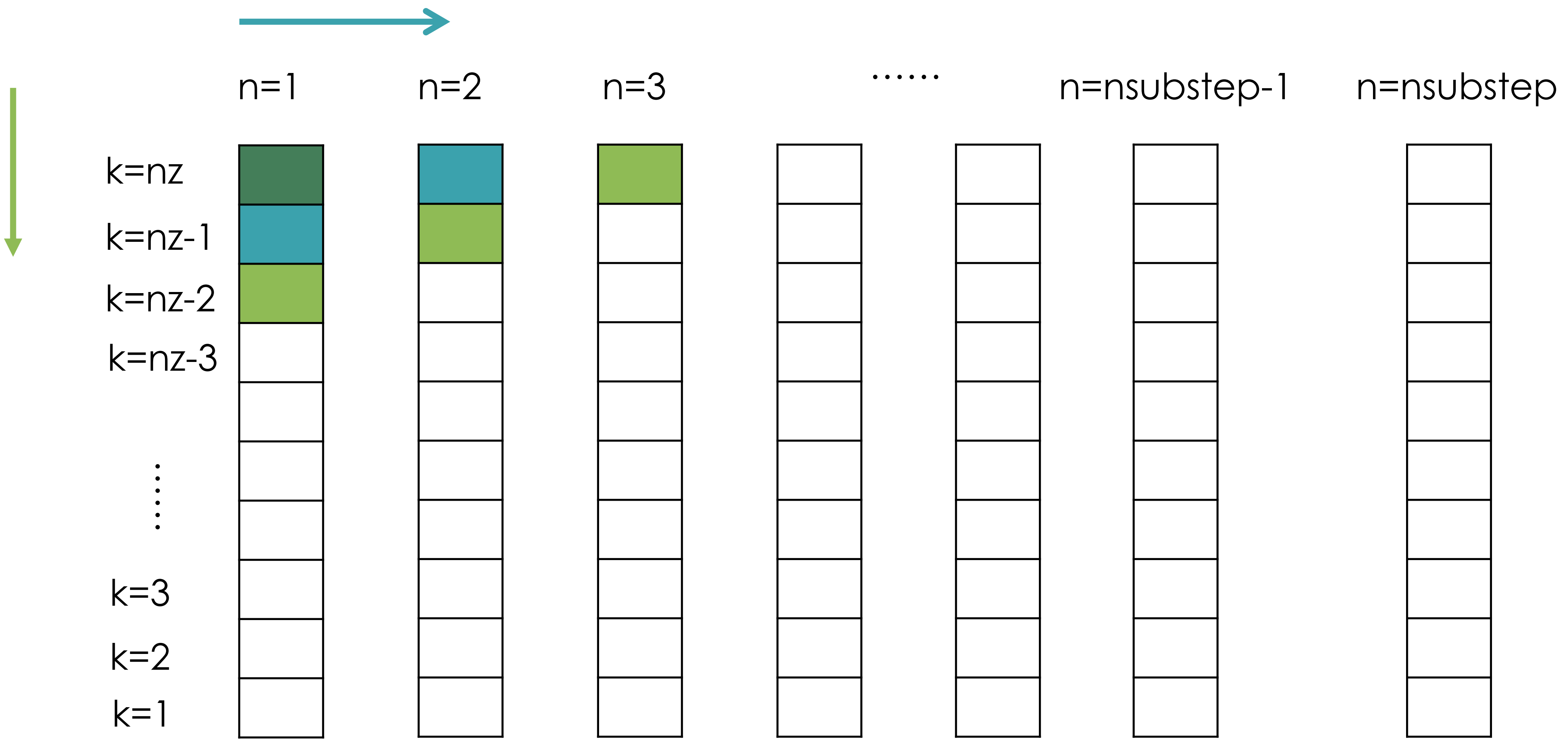
Possible new way for parallelizing n-loop and k-loop



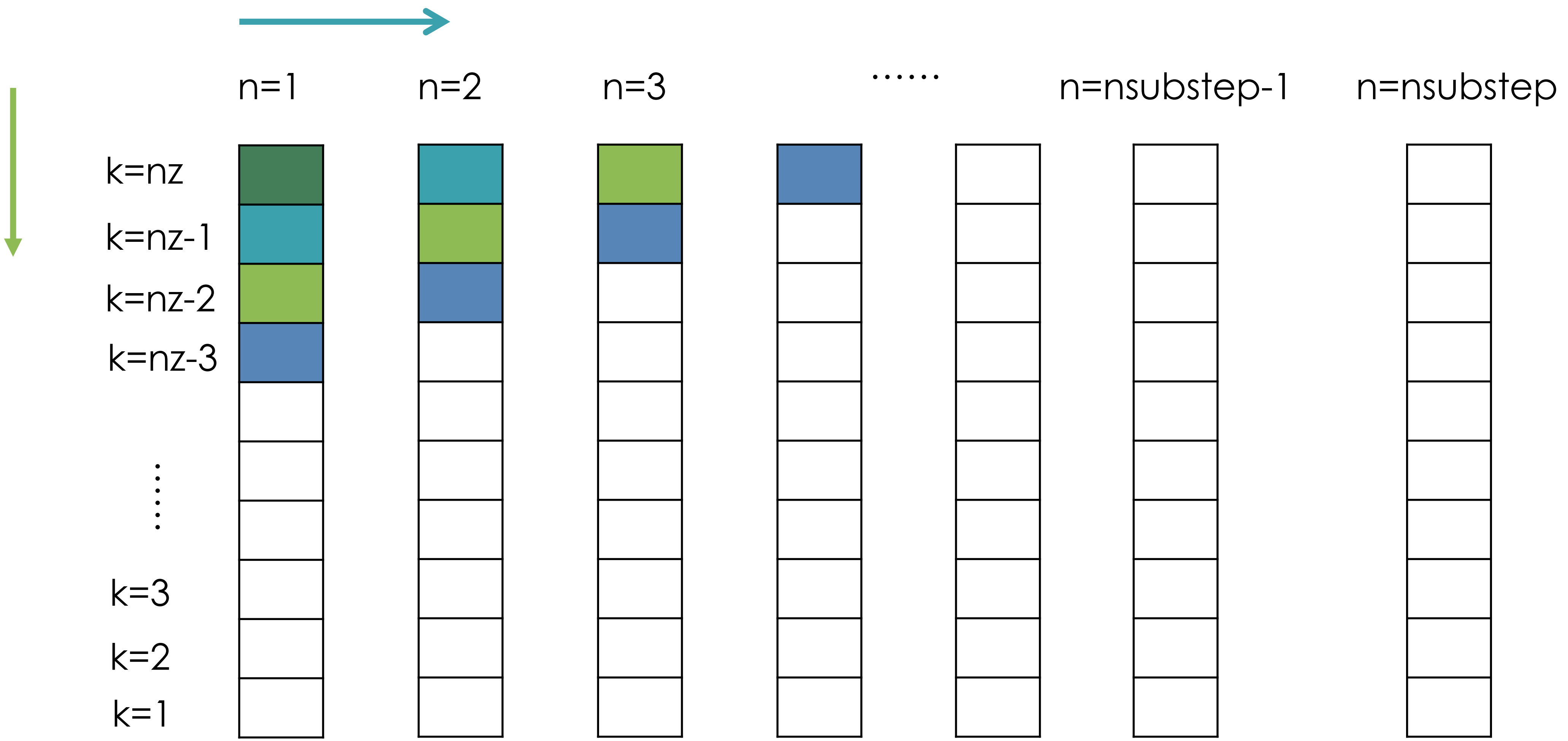
Possible new way for parallelizing n-loop and k-loop



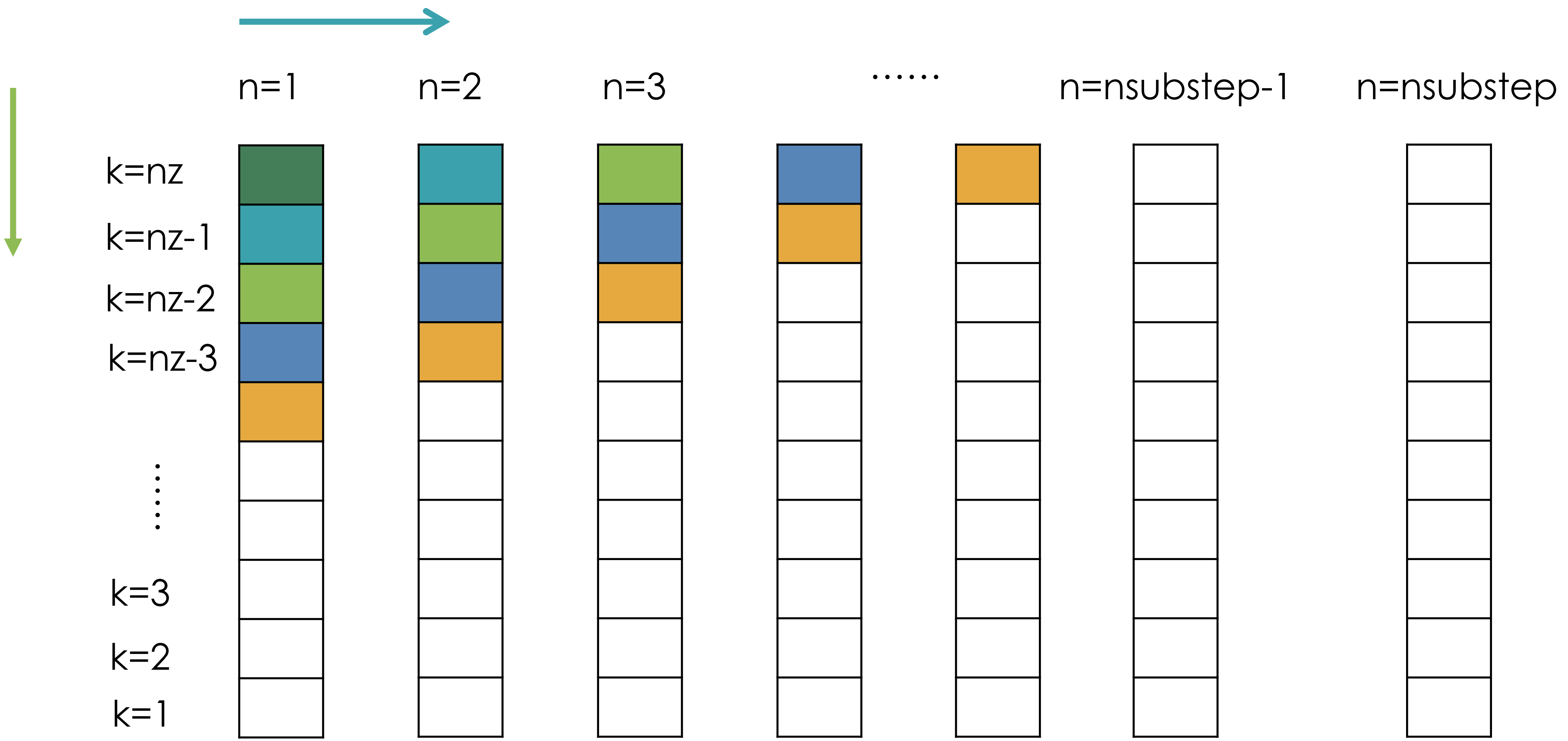
Possible new way for parallelizing n-loop and k-loop



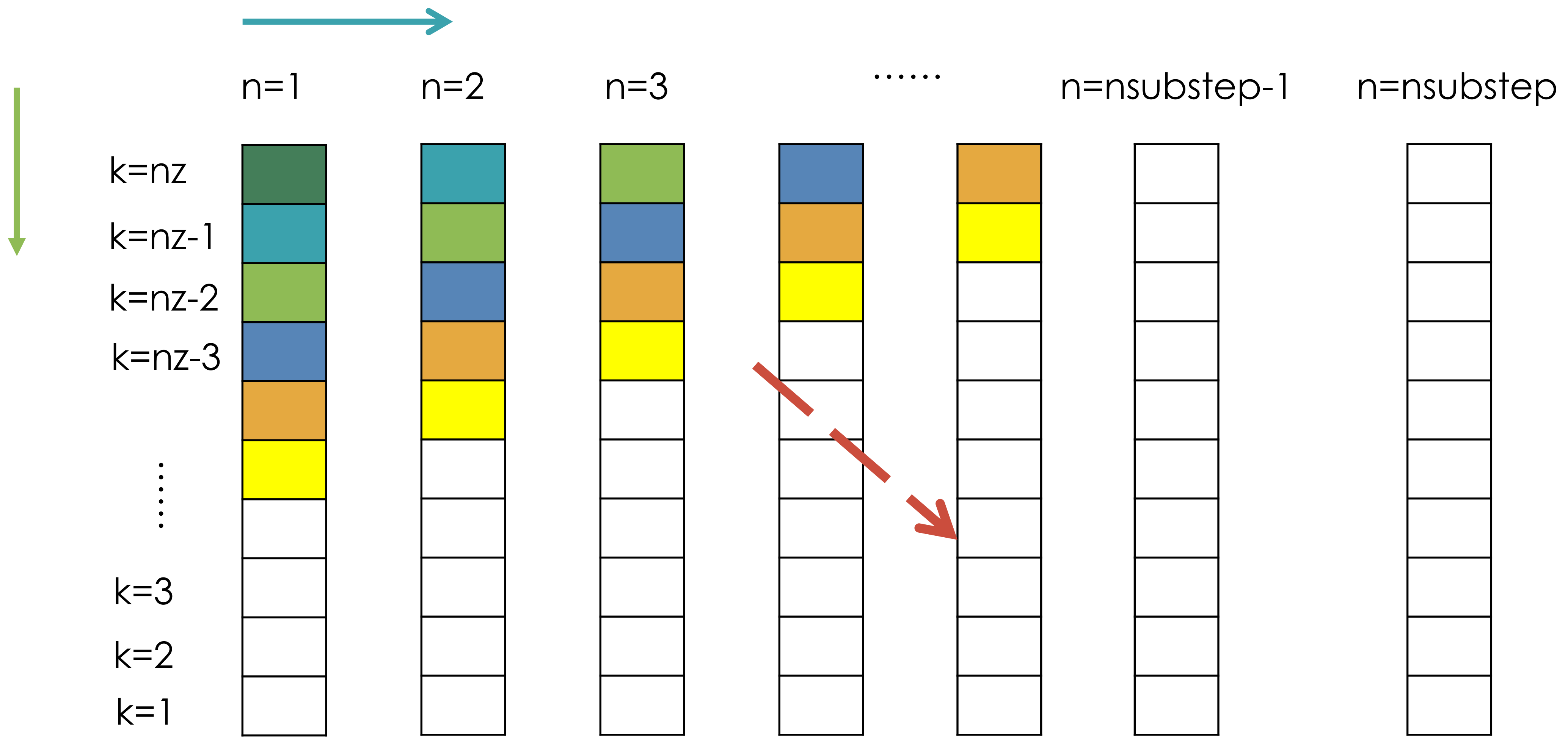
Possible new way for parallelizing n-loop and k-loop



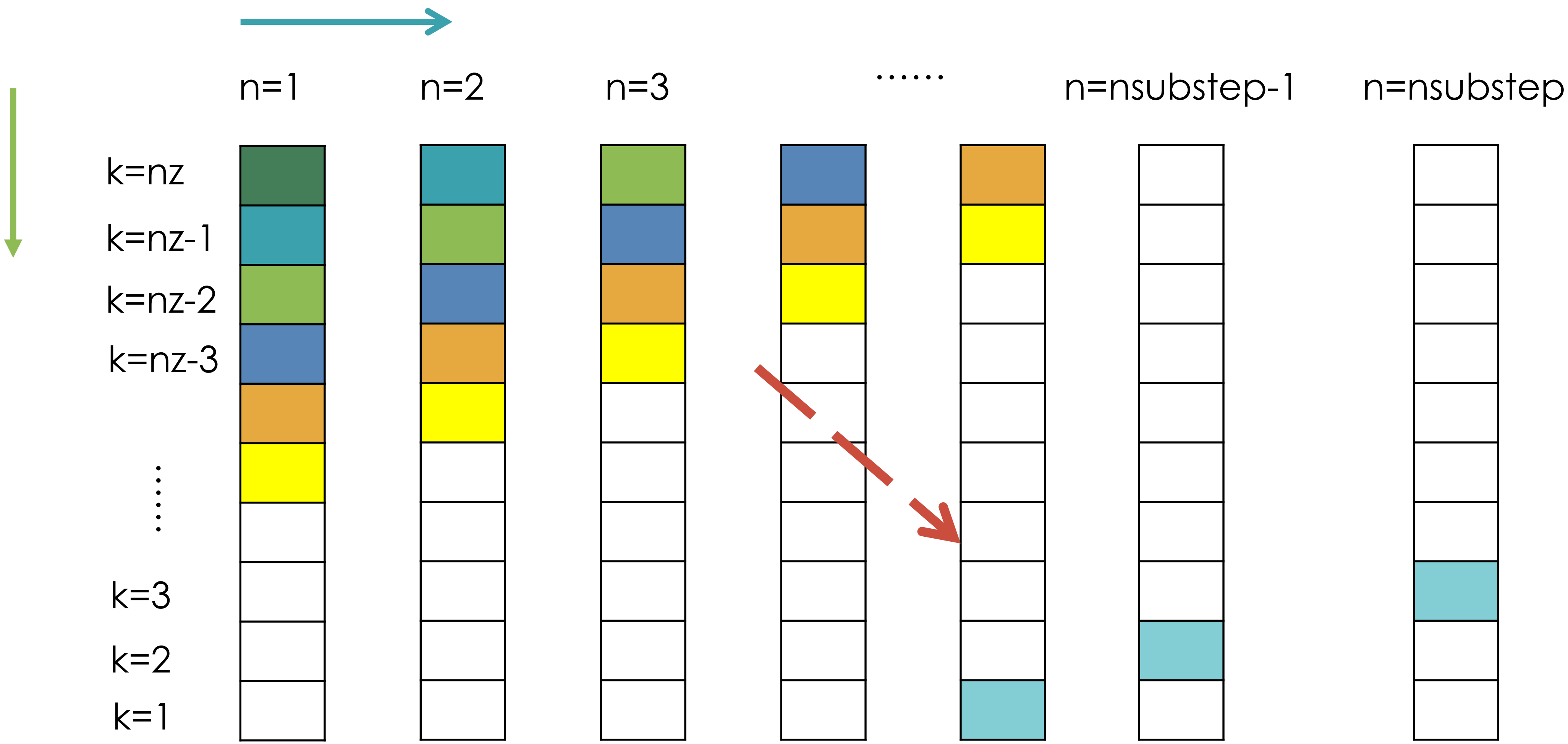
Possible new way for parallelizing n-loop and k-loop



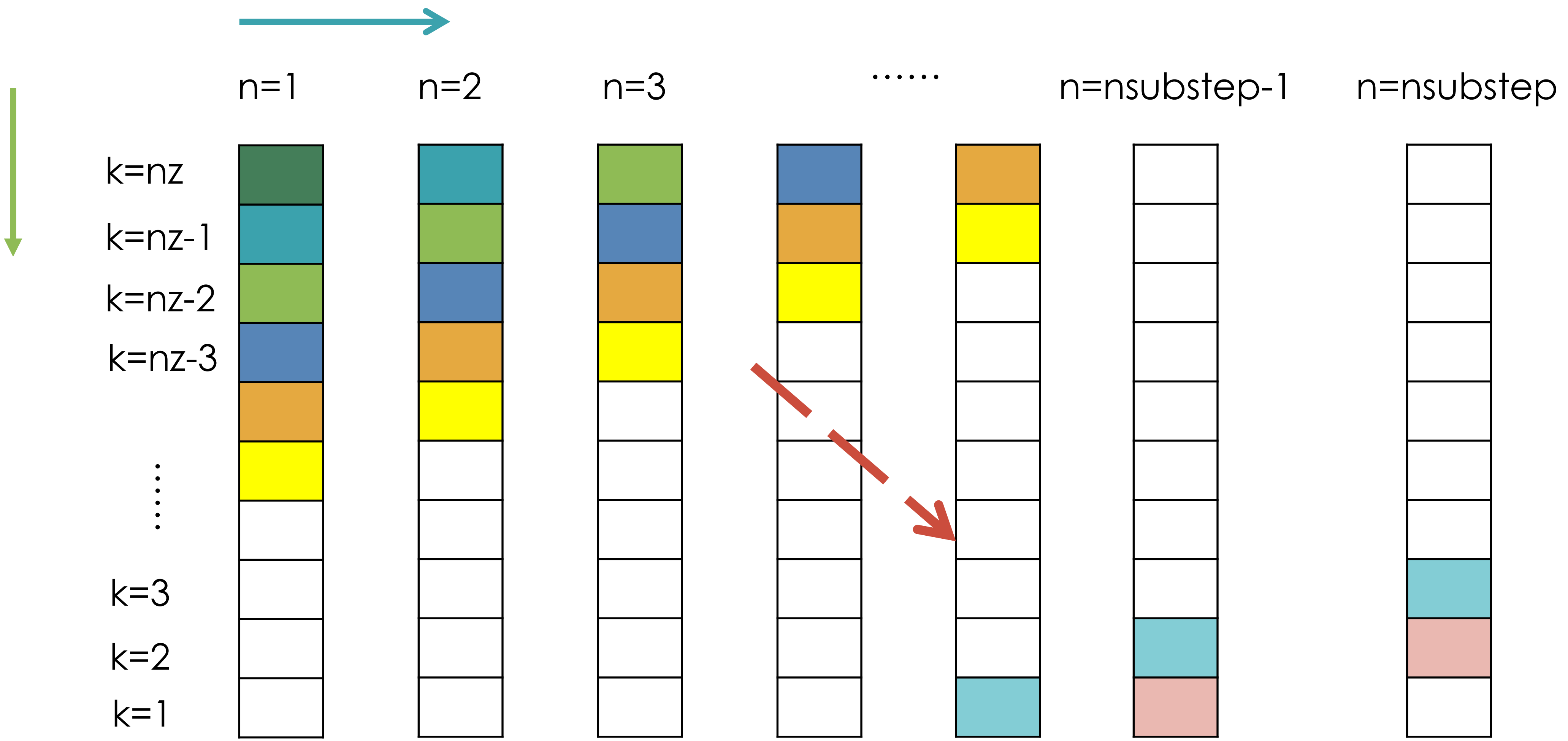
Possible new way for parallelizing n-loop and k-loop



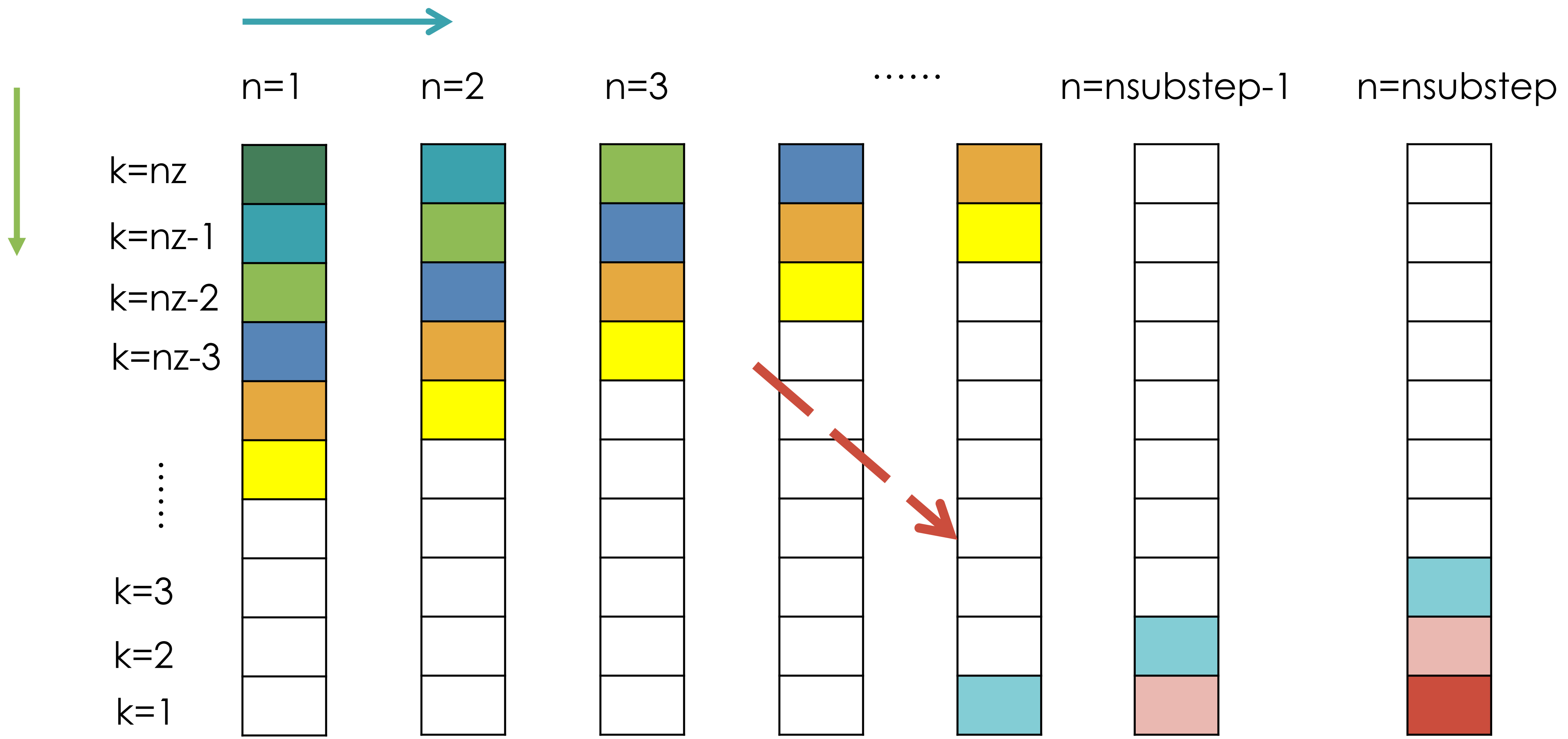
Possible new way for parallelizing n-loop and k-loop



Possible new way for parallelizing n-loop and k-loop



Possible new way for parallelizing n-loop and k-loop



limitation: nsubstep >= nz

?How to reduce the memory traffic?

– many conditional if branches

```
if (nq_l > 0) qfields(:,i_ql)=q1(ks:ke,i,j)
if (nq_r > 0) qfields(:,i_qr)=q2(ks:ke,i,j)
if (nq_l > 1) qfields(:,i_nl)=q3(ks:ke,i,j)
if (nq_r > 1) qfields(:,i_nr)=q4(ks:ke,i,j)
if (nq_r > 2) qfields(:,i_m3r)=q5(ks:ke,i,j)
if (nq_i > 0) qfields(:,i_qi)=q6(ks:ke,i,j)
if (nq_s > 0) qfields(:,i_qs)=q7(ks:ke,i,j)
if (nq_g > 0) qfields(:,i_qg)=q8(ks:ke,i,j)
if (nq_i > 1) qfields(:,i_ni)=q9(ks:ke,i,j)
if (nq_s > 1) qfields(:,i_ns)=q10(ks:ke,i,j)
```

– lookup table for gamma function in sedimentation.F90

```
  sedimentation_sedr_                          2.3e+07     -    414.461
    special_gammalookup_                       3.8e+09     -    172.311
      special_set_gammalookup_                       1     -      0.168
        special_gammafunc1_                    1.0e+06     -      0.077
```

Future Plan

- Continue to do code refactoring to expose more parallelism
  - Restructure the loops when it's necessary
- Continue to optimize the data locality
  - Reduce the data transfer between CPU and GPU
  - Reduce the number of system memory accesses
- First do the optimization with KiD-CASIM, then couple accelerated CASIM to UM for global simulation

Thank you