



Universität Hamburg

DER FORSCHUNG | DER LEHRE | DER BILDUNG

informatik
die zukunft

Praktikum «Paralleles Programmieren»

Thema

Seiler Tide Modell

Autor

Anna Fuchs

Datum

Oktober 2012

Betreuer

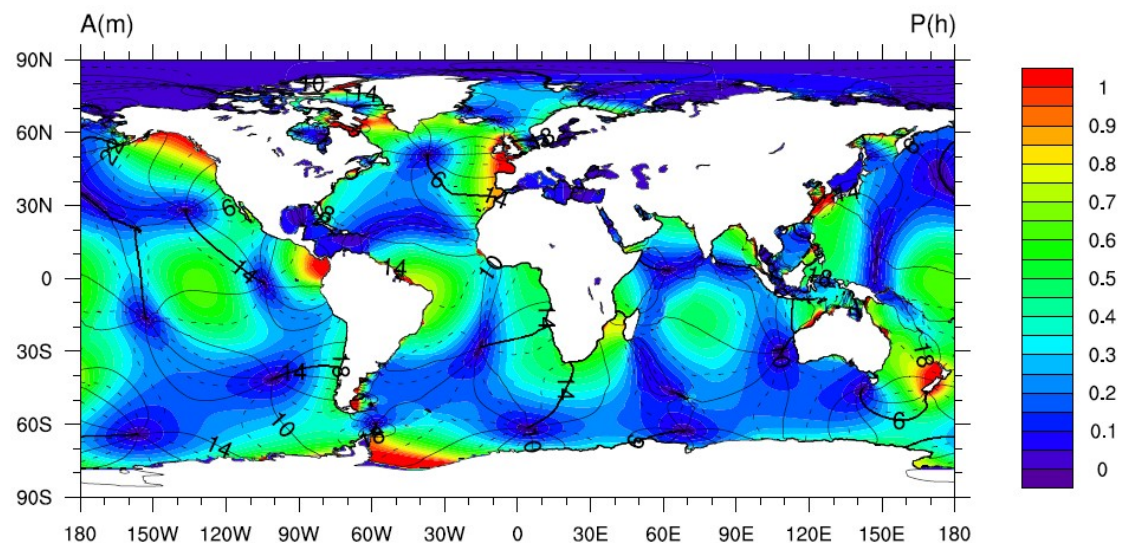
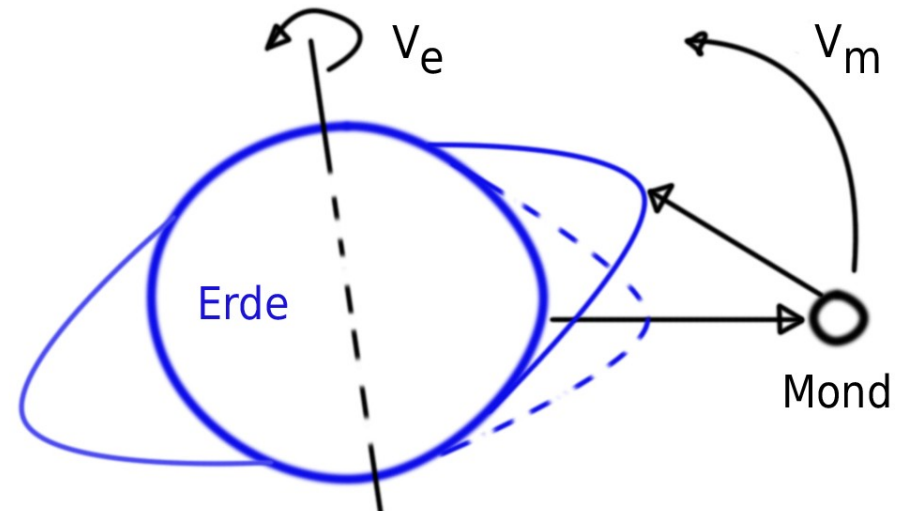
Petra Nerge

Überblick

- Gezeiten - Einführung
- Seiler Tide Modell
- Implementierung
- Parallelisierungsschema
 - Topologie
 - Nachrichtenaustausch
- Lastverteilung
- Schwierigkeiten
- Quellen

Gezeiten

- Anziehungskräfte zwischen Erde und anderen Himmelskörpern
- Massenbewegungen auf der Erde
- hier: halbtägige Mondgezeit
- Wichtig für Astronomie, Schiffahrt, Ozeanologie etc.
- Ziel: Wasserstände und Geschwindigkeiten berechnen
- Viele physikalische Größen zu berücksichtigen, mit Navier-Stokes-Gleichungen darstellbar



Gezeiten - Mathematik

- Nichtlineares Differentialgleichungssystem
- u bzw. v sind dabei die vertikal gemittelten zonale bzw. meridionale Geschwindigkeiten, ζ Höhe des Wasserstandes
- Vertikale Geschwindigkeiten angenommen als vernachlässigbar gegenüber u und v
- Berücksichtigt Rotationen, Druckgradient, Reibung, Advection etc.
- Numerische Lösung

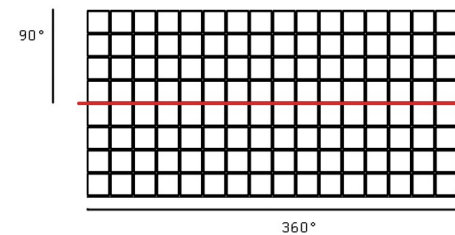
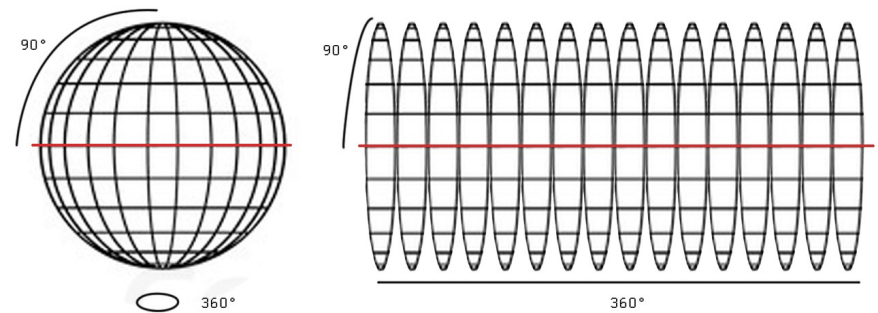
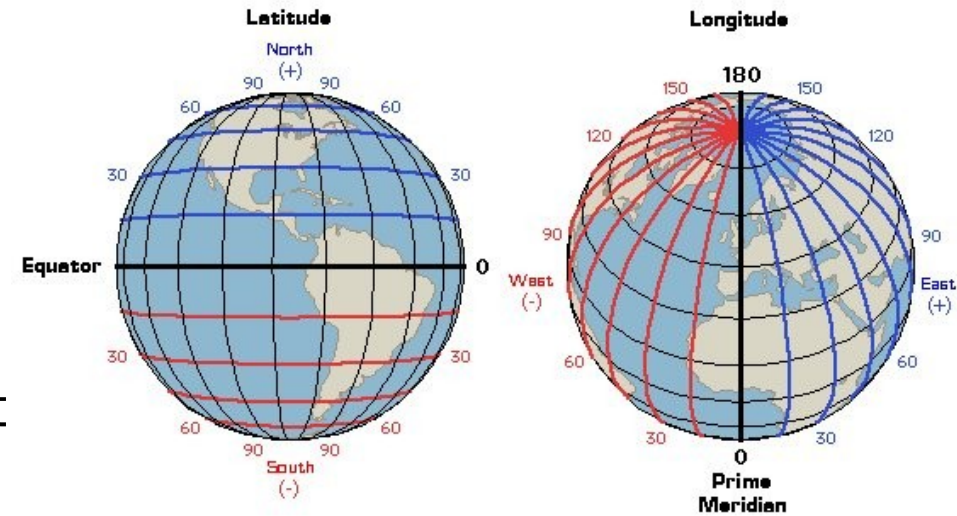
$$\frac{\partial u}{\partial t} + \frac{u}{R \cdot \cos\varphi} \frac{\partial u}{\partial \lambda} + \frac{v}{R} \frac{\partial u}{\partial \varphi} - 2\Omega \sin(\varphi)v = -\frac{g}{R \cdot \cos\varphi} \frac{\partial \zeta}{\partial \lambda} + \frac{1}{R \cdot \cos\varphi} \frac{\partial \Phi_j^r}{\partial \lambda} + \Gamma_{B,\lambda} + \Gamma_{E,\lambda}$$

$$\frac{\partial v}{\partial t} + \frac{u}{R \cdot \cos\varphi} \frac{\partial v}{\partial \lambda} + \frac{v}{R} \frac{\partial v}{\partial \varphi} - 2\Omega \sin(\varphi)u = -\frac{g}{R} \frac{\partial \zeta}{\partial \varphi} + \frac{1}{R} \frac{\partial \Phi_j^r}{\partial \varphi} + \Gamma_{B,\varphi} + \Gamma_{E,\varphi}$$

$$\frac{\partial \zeta}{\partial \varphi} = -\frac{1}{R \cos\varphi} \left[\frac{\partial}{\partial \lambda} (H \cdot u) + \frac{\partial}{\partial \varphi} (H \cdot v \cdot \cos\varphi) \right]$$

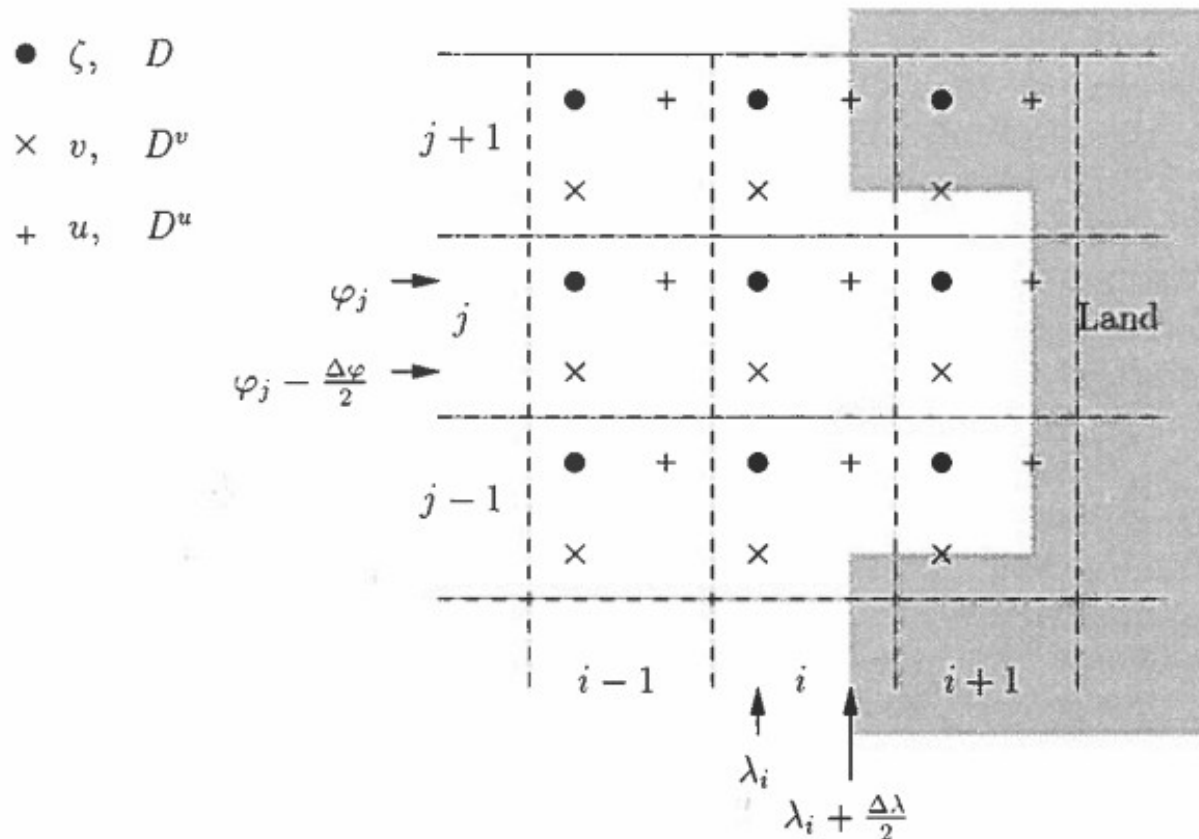
Seiler Tide Modell

- 1989 von Ulrike Seiler entworfen
Institut für Meereskunde - UHH
- Süd- und Nordhalbkugel abwechselnd durchiteriert — hinsichtlich Zeta
- Nordpolarkappe (4°) getrennt berechnet
- Differentialgleichungssystem 2. Ord.
 - Numerische Lösung
 - Gitterpunktmethodode
 - Semiimplizites Verfahren
 - Arithemische Mittelwerte



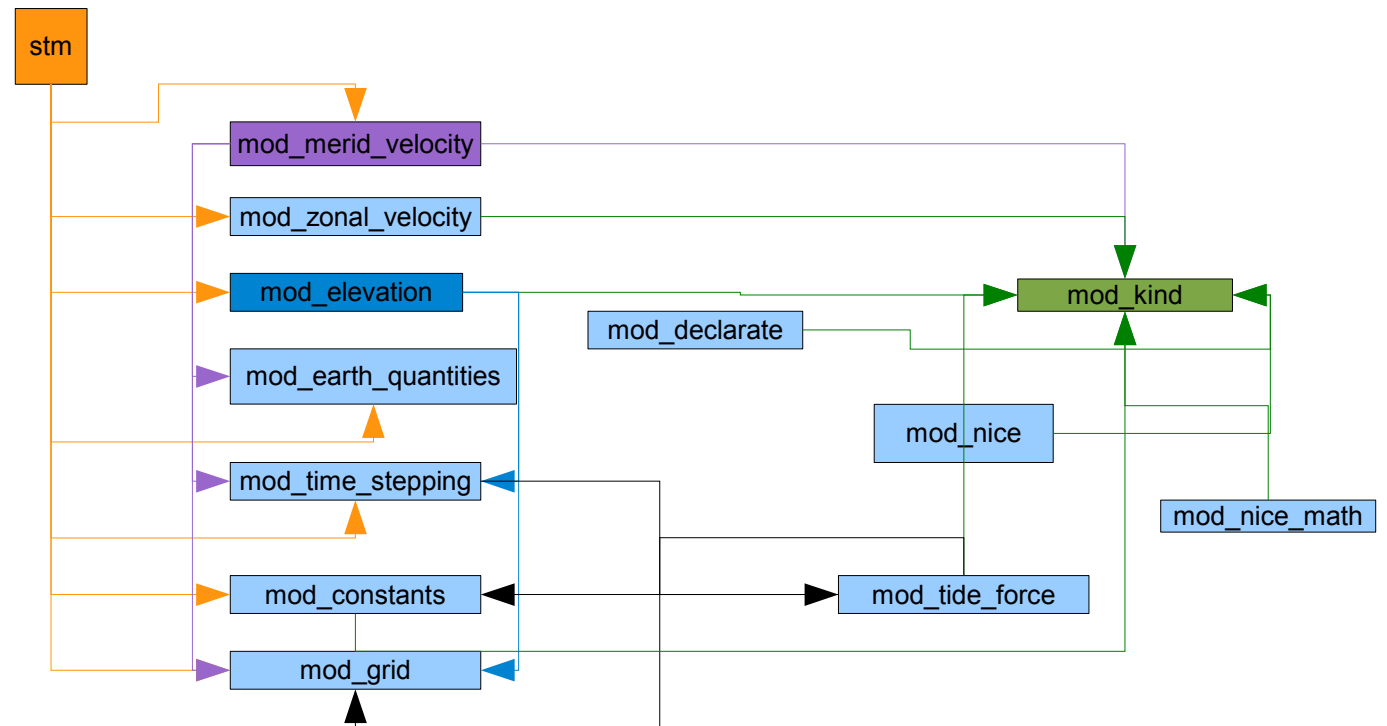
Seiler Tide Modell

- Aracawa C-Gitter
- $\Delta\varphi$ bzw. $\Delta\lambda$ - Winkelabstand der Wasserpunkte ζ und der Geschwindigkeitskomponente u, v in der geographischen Breite bzw. Länge
- Tiefenverteilung D an den ζ -Punkten



Implementierung

- Fortran
- Von Petra Nerge überarbeitet
- 12 Module
- -03
- 1 min 44 seq
- 1 min 30 seq

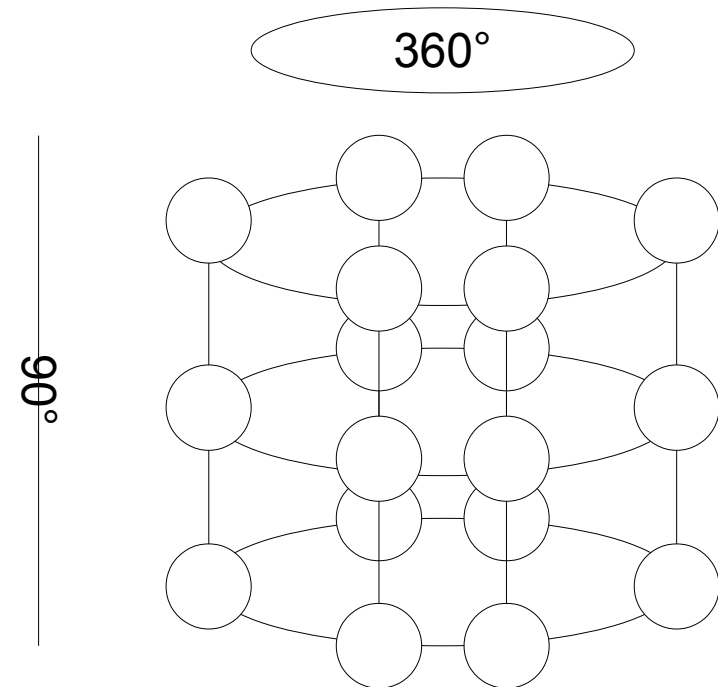


Profiling

- Flat profile: **1min 44**
- Each sample counts as 0.01 seconds.
- % cumulative self self total
- time seconds seconds calls s/call s/call name
- 83.21 83.02 83.02 2001 0.04 0.04 **__mod_elevation_MOD_elevation_iteration_relaxation**
- 1.81 84.83 1.81 1 1.81 99.79 MAIN__
- 1.77 86.60 1.77 2001 0.00 0.00 __mod_elevation_MOD_elevation_iteration_matrix_prepare
- 1.01 87.61 1.01 2001 0.00 0.00 __mod_elevation_MOD_elevation_iteration_zeta_coefficient
- 1.01 88.62 1.01 126783360 0.00 0.00 __mod_merid_velocity_MOD_merid_vel_bottom_friction
- 0.92 89.54 0.92 126783360 0.00 0.00 __mod_zonal_velocity_MOD_zonal_vel_bottom_friction
- 0.87 90.41 0.87 2001 0.00 0.00 __mod_merid_velocity_MOD_merid_vel_advection
- 0.76 91.17 0.76 2001 0.00 0.00 __mod_zonal_velocity_MOD_zonal_vel_advection
- ...
- **elevation_iteration_relaxation – entscheidende Schleife – 120 Zeilen**

Parallelisierung - Topologie

- Soll bequemere Kommunikation ermöglichen
- Leichter kommunizieren mit Nachbarn
- Horizontal im Ring kommunizieren
- Vertikal nicht periodisch
- Zylindertopologie
- Nord-/Südhalbkugel bis auf Nordpolarkappe identisch
- Abwechselnde aber volle Ausnutzung
- `MPI_Cart_create(COMM_OLD, NDIMS, DIMS, PERIODS, REORDER, COMM_CART, IERROR)`



Parallelisierung - Topologie

- `MPI_Cart_create(COMM_OLD, NDIMS, DIMS, PERIODS, REORDER, COMM_CART, IERROR)`
- `MPI_CART_SHIFT(COMM_CART, DIRECTION, DISPL, RANK_SOURCE, RANK_DEST, IERROR)`
- Im Ring:


```
CALL MPI_Cart_shift(cylinder_comm, 0, 1, src_rank, dst_rank, ierror)
```
- Vertikal:


```
CALL MPI_Cart_shift(cylinder_comm, 1, 1, src_rank, dst_rank, ierror)
```
- Kein Nachbar da - `MPI_PROC_NULL`
 - Kann als *source* und *destination* verwendet werden
- Verschiedene Aufteilungsstrategien - mehr in Lastausgleich
 - Nur in Scheiben - aktuell
 - Bis max 90 nprocs
 - Quadratisch
 - Entsprechend dem Seitenverhältnis der globalen Daten

Parallelisierung - Nachrichten



```
!--- MPI
!communicator for MPI
INTEGER :: old_comm, cylinder_comm, ndims, reorder, IERROR, nprocs, &
           my_rank, faktor, long_end

INTEGER dim_size(2)
INTEGER :: help_dim = 0
LOGICAL periods (0:1)
ndims = 2

CALL MPI_Comm_Size(MPI_COMM_WORLD, nprocs, IERROR)

old_comm = MPI_COMM_WORLD

dim_size(1) = 1           !longs = 360
dim_size(2) = nprocs     !lats = 90

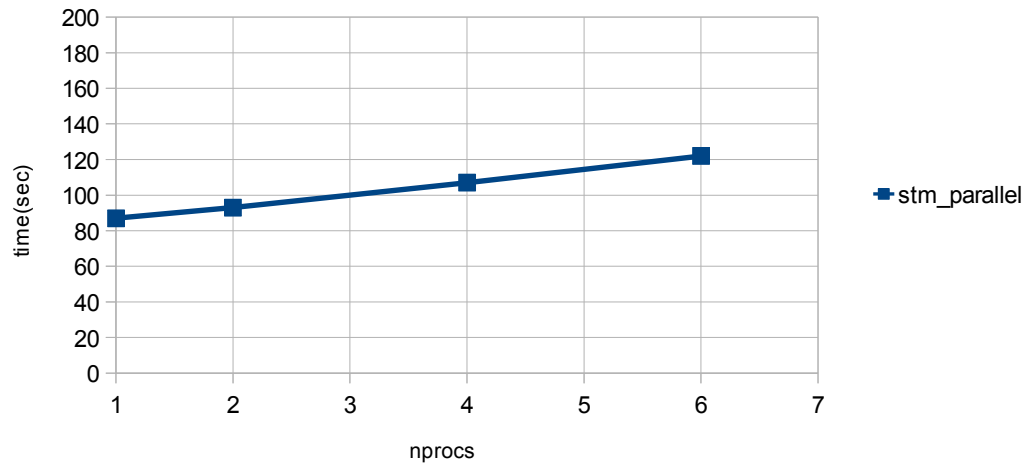
periods(0) = .true.      !row-periodic
periods(1) = .false.    !column-nonperiodic
reorder   = 1

CALL MPI_Cart_create(old_comm, ndims, dim_size, periods,      &
                    reorder, cylinder_comm, IERROR)
CALL MPI_COMM_SIZE(cylinder_comm, nprocs, IERROR)
CALL MPI_COMM_RANK(cylinder_comm, my_rank, IERROR)
```

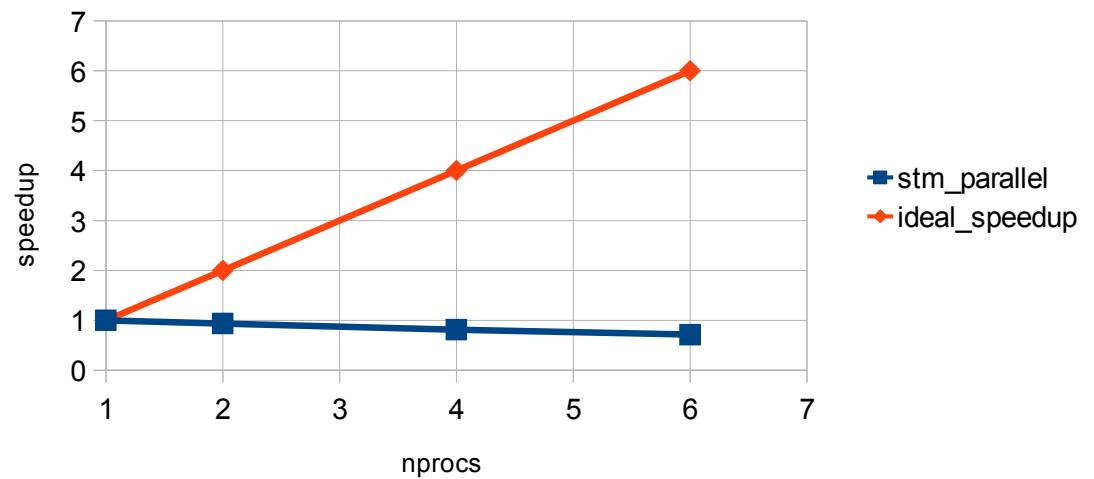
```
1 lon = longs_start + (rank * 90 / nprocs)    !Beginn der Längen abhängig vom Rang
2 mpi_end = lon + (90 / nprocs)              !Ende der Längen abhängig vom Rang
3
4
5 !--- start iteration
6 DO count_checks = 0 , max_convergence_checks
7   DO count_iteration = 1 , number_iterations
8     !--- set southerly hemisphere to iterate first
9     hemisphere = 1
10    start_lat = start_south_lat + 1
11    end_lat   = end_south_lat
12    DO WHILE ( hemisphere < 3 )
13      !--- latitudes without boundaries
14      DO lat = start_lat, end_lat
15        DO lon = longs_start, mpi_end
16          residuum (lon , lat) = &
17            coefficient_north(lon , lat) * zeta_new(lon , lat+1) + &
18            coefficient_south(lon , lat) * zeta_new(lon , lat-1) + &
19            coefficient_east (lon , lat) * zeta_new(lon+1 , lat ) + &
20            coefficient_west (lon , lat) * zeta_new(lon-1 , lat ) - &
21            zeta_new(lon , lat ) + &
22            coefficient_zeta (lon , lat)
23
24          count = count +1
25        END DO
26      END DO
27
28      CALL MPI_Alltoall (residuum(lon,start_lat), count, MPI_REAL, residuum,      &
29                        count * (nprocs-1), MPI_REAL, cylinder_comm, IERROR)
30
31      !--- set northerly hemisphere to iterate next
32      hemisphere = hemisphere + 1
33      start_lat = start_north_lat
34      end_lat   = end_north_lat - 1
35    END DO
36
37
38
39
```

Parallelisierung - Analyse

Zeitmessung



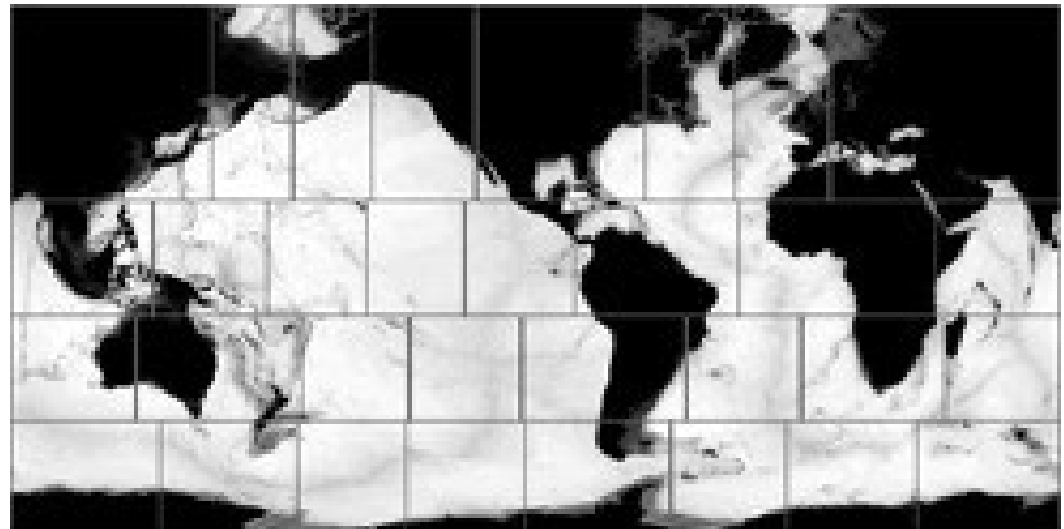
SpeedUP



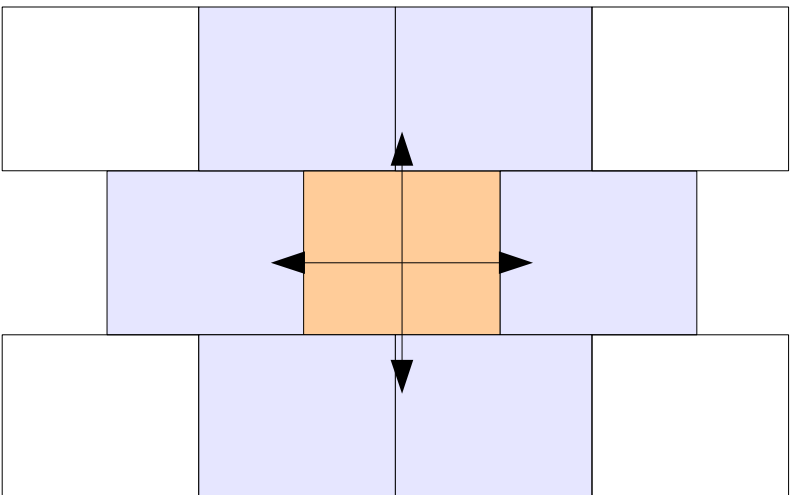
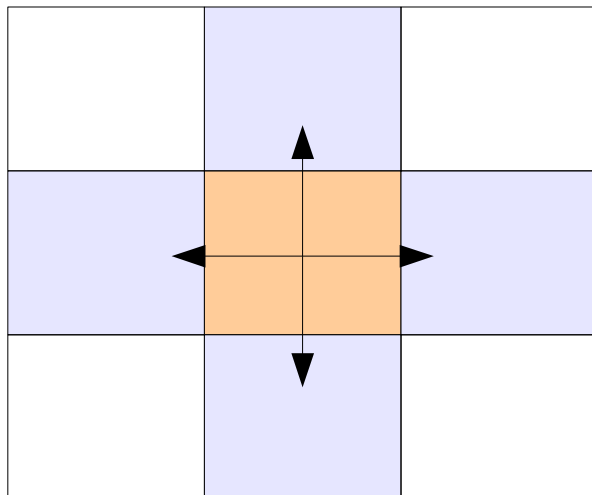
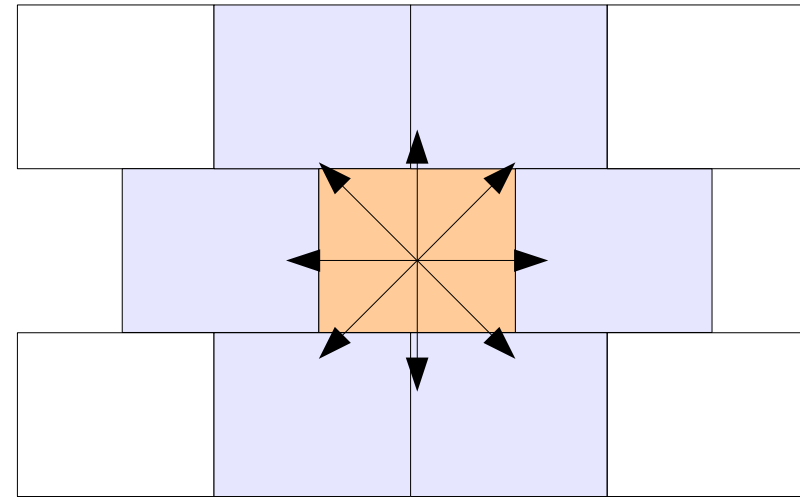
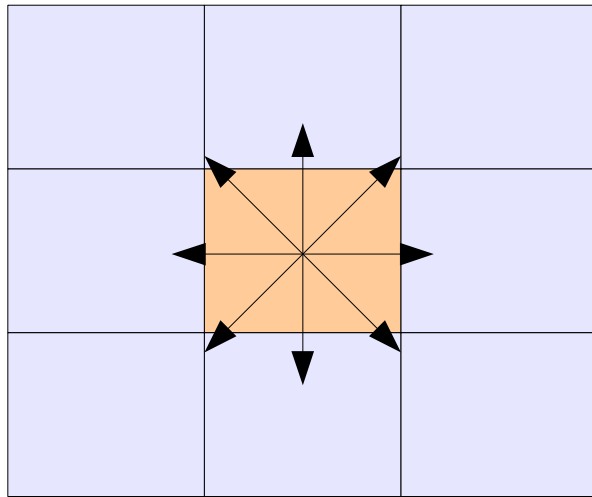
Parallelisierung - Lastausgleich

- 2 Gesichtspunkte
 - Land- / Wasserpunkteverteilung
 - Aufteilung in der Topologie
- Balance zwischen gutem Lastausgleich und angemessener Zahl der Nachbarn finden
- Immer das Ziel im Auge behalte : wenig Kommunikation

- Beispiel einer lastangepassten Domänendekomposition für das Ozean Modell MPIOM mit einem hierarchischen Ansatz (Scales)



Parallelisierung - Lastausgleich



Schwierigkeiten

- Keine Fortran Kenntnisse
- Anspruchsvoller Hintergrund (Mathe, Physik)
 - Fachliche Recherche gar nicht so einfach
- Fremdes Programm
- Keine Kapselung möglich
 - Kleinste Änderungen pflanzen sich fort
- Vollkommen unterschätzt

Anhang

- Quellen
 - Diplomarbeit Petra Nerge
 - Sonst siehe Ausarbeitung

- Werkzeuge:
 - Openoffice
 - doxygen

=)

Danke für die Aufmerksamkeit