INFORMATIK

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

Introduction to High Performance Computing

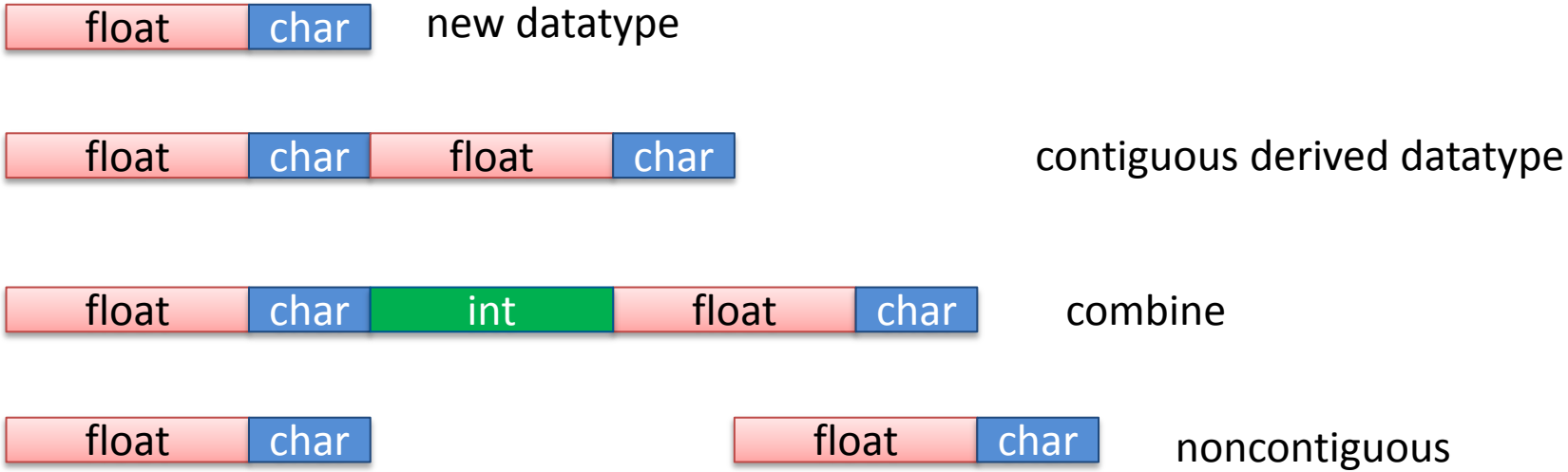
Dr. habil. Josef Schüle

Datatypes

- Basic datatypes like MPI_INT, MPI_FLOAT are not sufficient
- Structures may be packed
  - extra copy
  - no usage of scatter/gather hardware
- Noncontiguous data, like a sub-block of a matrix

Datatypes

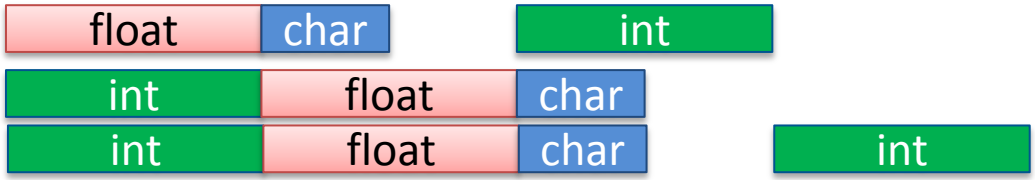
- **Definition of own derived datatypes**
  - **set together from basic datatypes**
  - **set together from other derived datatypes**



Datatypes

typemap = {(float,disp0),(char,disp1),(int,disp2)}
disps: relative postion (displacement)

{(float,0),(char,4),(int,8)}
{(newtype,0),(int,-4)}
{(int,8),(newtype,0),(int,-4)}

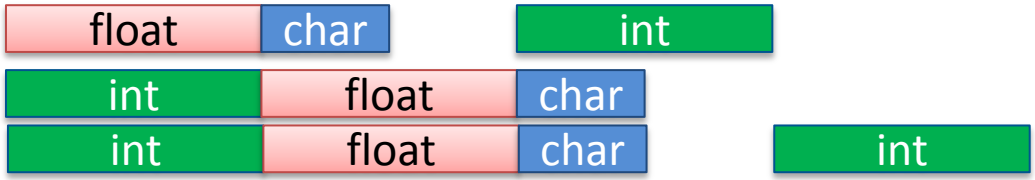


Datatypes

lb = smallest displacement
ub = largest + type + padding
extent =ub-lb

lb	ub	extent
0	12	12
-4	8	12
-4	12	16

{(float,0),(char,4),(int,8)}
{(newtype,0),(int,-4)}
{(int,8),(newtype,0),(int,-4)}



Datatypes


```
MPI_Type_contiguous(count, old_t,&new_t)
```

count times old_t glued to one new_t regarding extent.

```
MPI_Type_vector(count, block, stride, old_t,&new_t)
```

count times (block times old_t) there the blocks have stride distance regarding extent.

```
MPI_Type_vector(2, 3, 4, old_t,&new_t)
```

(fl,0),(c,4), (fl,8),(c,12), (fl,16),(c,20), (fl,32),(c,36), (fl,40),(c,44), (fl,48),c(52)

Datatypes

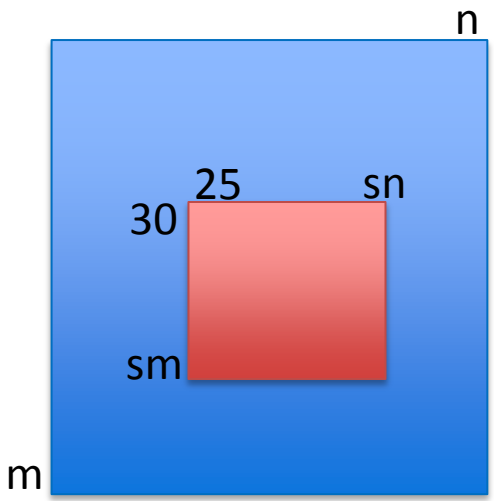

```
MPI_Type_create_struct(count, ar_block, ar_dis,
                       ar_typ,&new_t)
```

```
struct Particlestruct {
 double c[3];  // coordinates
 double v[3];  // velocities
 double m;     // mass
  int hydro;   // water molecules
 char name;    // name (type) of particles
} particle;
```

5,(3,3,1,1,1),(0,32,56,64,68),(d,d,d,i,c)

Datatypes


```
MPI_Type_create_subarray(ndims,&ars,&sus,&sst,ord,
                         o_t,&new_t)
```



Matrix with 2 dimensions
(ndims=2) of sizes n and m
(ars[0]=n, ars[1]=m, n running in x
(ord=MPI_ORDER_C)

Submatrix, beginning at element (25,30)
(sst[0]=25, sst[1]=30) of sizes sn and sm
(sus[0]=sn, sus[1]=sm)

Datatypes


```
ars[0]=n; ars[1]=m;
sus[0]=sn; sus[1]=sm;
sst[0]=25; sst[1]=30;
MPI_Type_create_subarray(2,ars,sus,sst,MPI_ORDER_C,
                         MPI_FLOAT,&new_t);
```

Datatypes

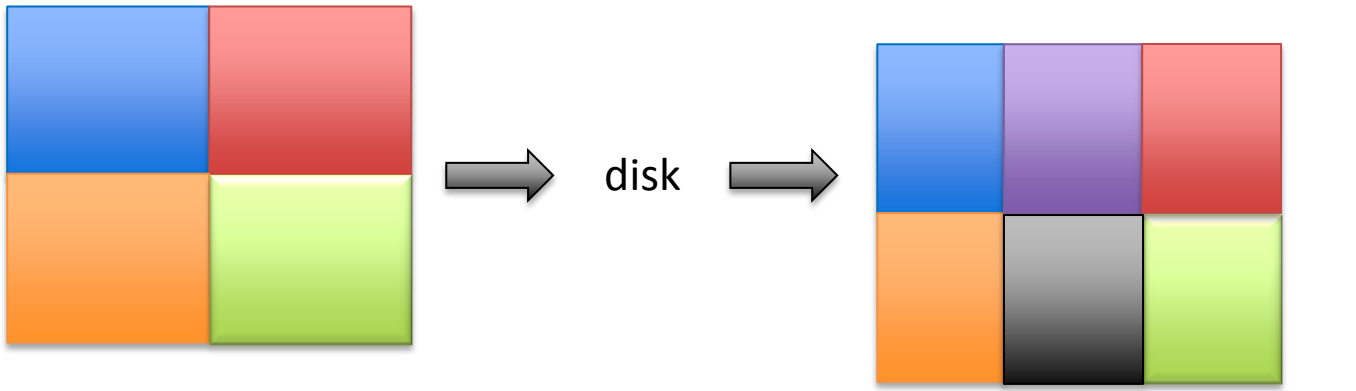
MPI_Type_commit(&datatype)

commits the datatype. It may be used in communications until

MPI_Type_free(&datatype)

I/O

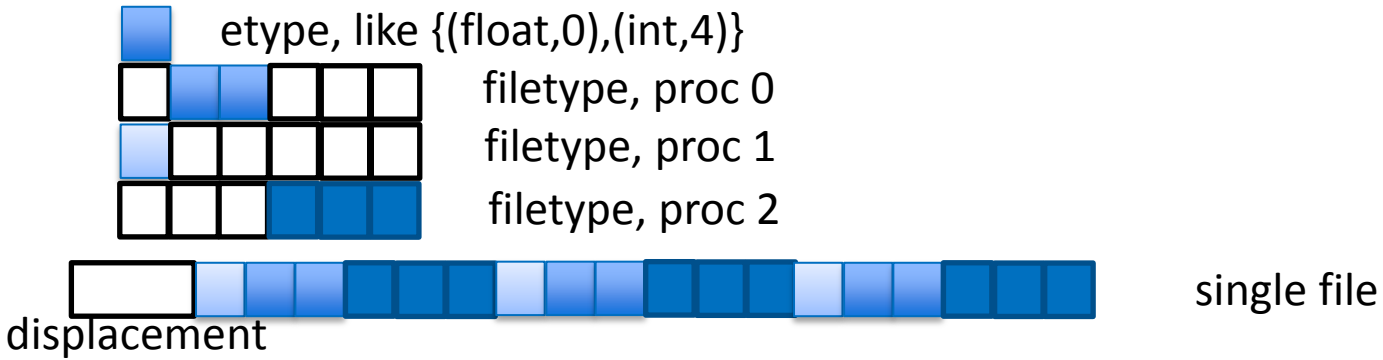
- **requirement for collective I/O**
- **realization with help of derived datatypes**



partitioned data (4 proc) single file partitioned data (6 proc)

I/O

- file (collective access)
- displacement – where to start
- etype – nondecreasing positioning, basic unit
- filetype – template build of etype(s)
- view – orderset set of etypes



I/O


```
MPI_File_open(comm,&filename,amode,info,&file_handle)
MPI_File_close(&file_handle)
```

I/O

positioning	synchronism	coordination	
		noncollective	*collective*
explicit offsets	*blocking*	MPI_FILE_READ_AT MPI_FILE_WRITE_AT	MPI_FILE_READ_AT_ALL MPI_FILE_WRITE_AT_ALL
	nonblocking	MPI_FILE_IREAD_AT MPI_FILE_IWRITE_AT	MPI_FILE_IREAD_AT_ALL MPI_FILE_IWRITE_AT_ALL
	split collective	N/A	MPI_FILE_READ_AT_ALL_BEGIN MPI_FILE_READ_AT_ALL_END MPI_FILE_WRITE_AT_ALL_BEGIN MPI_FILE_WRITE_AT_ALL_END
individual file pointers	*blocking*	MPI_FILE_READ MPI_FILE_WRITE	MPI_FILE_READ_ALL MPI_FILE_WRITE_ALL
	nonblocking	MPI_FILE_IREAD MPI_FILE_IWRITE	MPI_FILE_IREAD_ALL MPI_FILE_IWRITE_ALL
	split collective	N/A	MPI_FILE_READ_ALL_BEGIN MPI_FILE_READ_ALL_END MPI_FILE_WRITE_ALL_BEGIN MPI_FILE_WRITE_ALL_END
shared file pointer	*blocking*	MPI_FILE_READ_SHARED MPI_FILE_WRITE_SHARED	MPI_FILE_READ_ORDERED MPI_FILE_WRITE_ORDERED
	nonblocking	MPI_FILE_IREAD_SHARED MPI_FILE_IWRITE_SHARED	N/A
	split collective	N/A	MPI_FILE_READ_ORDERED_BEGIN MPI_FILE_READ_ORDERED_END MPI_FILE_WRITE_ORDERED_BEGIN MPI_FILE_WRITE_ORDERED_END

INFORMATIK

Introduction to High Performance Computing

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

Vielen Dank

Thanks