Introduction to High Performance Computing

MPI (2)

Misc point-to-point

If not known

- data type, type of message
- sender
- size of data to be received

Probe for a message before receiving it

```
MPI_Probe(source, tag, comm, &status)
MPI_Iprobe(source, tag, comm, &flag, &status)
```

- Look into receive stack (blocking or non blocking) whether an appropriate message arrived.
- Allows inspection (status) first.
- Does not receive message!

Example - probing

```
// Receive any messages with tag 34
MPI_Probe(MPI_ANY_SOURCE,34,comm,&status);
// Inspect sender
if(status.MPI_SOURCE == 0)  {
   … // small control message
} else {
  // large data – alloc appropriate memory
  MPI_Get_count(&status, MPI_DOUBLE, &number);
  buffer = (double *)malloc(number*sizeof(double));
  MPI_Recv(buffer,number,MPI_DOUBLE,
           status.MPI_SOURCE,tag,comm,&status);
}
```

status.MPI_SOURCE (status.MPI_TAG) allows classification of messages
MPI_Get_count helps to get number of words required to store message

Probing – thread safety

Threaded application (hybrid with OpenMP)– one MPI Stack

- Thread 0 probes for a message
- Thread 1 probes for a message
- Thread 0 receives message
- Thread 1 ?

No identification of messages possible.
Probe, Iprobe are not thread safe.

Be careful !

Mprobe and Improbe

`MPI_Mprobe(source, tag, comm, &message, &status)`
`MPI_Improbe(source, tag, comm, &flag, &message, &status)`

- Look into receive stack (blocking or non blocking) whether an appropriate message arrived.
- Allows inspection (status) first.
- Does not receive message!
- Returns a message handle to identify a message in multi threaded context
- Receive data with matching receive operations:

`MPI_Mrecv`(&buf, count, datatype, &message, &status)
`MPI_Imrecv`(&buf, count, datatype, &message, &request)

- No interference from another thread possible -> thread safe

pack and unpack

How to exchange a structure?

```
struct Particlestruct {
 double c[3];  // coordinates
 double v[3];  // velocities
 int hydro;      // water molecules in complex
 double m;      // mass
 char name;   // name (type) of particles
} particle[10];
.
for(i=0;i<10;i++) {
      MPI_Send(particle[i].c,3,MPI_DOUBLE,rec,tag,comm);
      MPI_Send(particle[i].v,3,MPI_DOUBLE,rec,tag,comm);
      MPI_Send(particle[i].hydro,1,MPI_INT,rec,tag,comm);
      MPI_Send(particle[i].m,1,MPI_DOUBLE,rec,tag,comm);
      MPI_Send(particle[i].name,1,MPI_CHAR,rec,tag,comm);

}
```

pack and unpack

Questions:

- How is a structure organized in memory?

```
struct Particlestruct {
 double c[3];  // coordinates
 double v[3];  // velocities
 int hydro;    // water molecules in complex
 double m;     // mass
 char name;    // name (type) of particles
} particle[10];
```

c 3x8 Byte	v 3x8 Byte	8 Byte	8 Byte	8 Byte

consecutive ?

pack and unpack

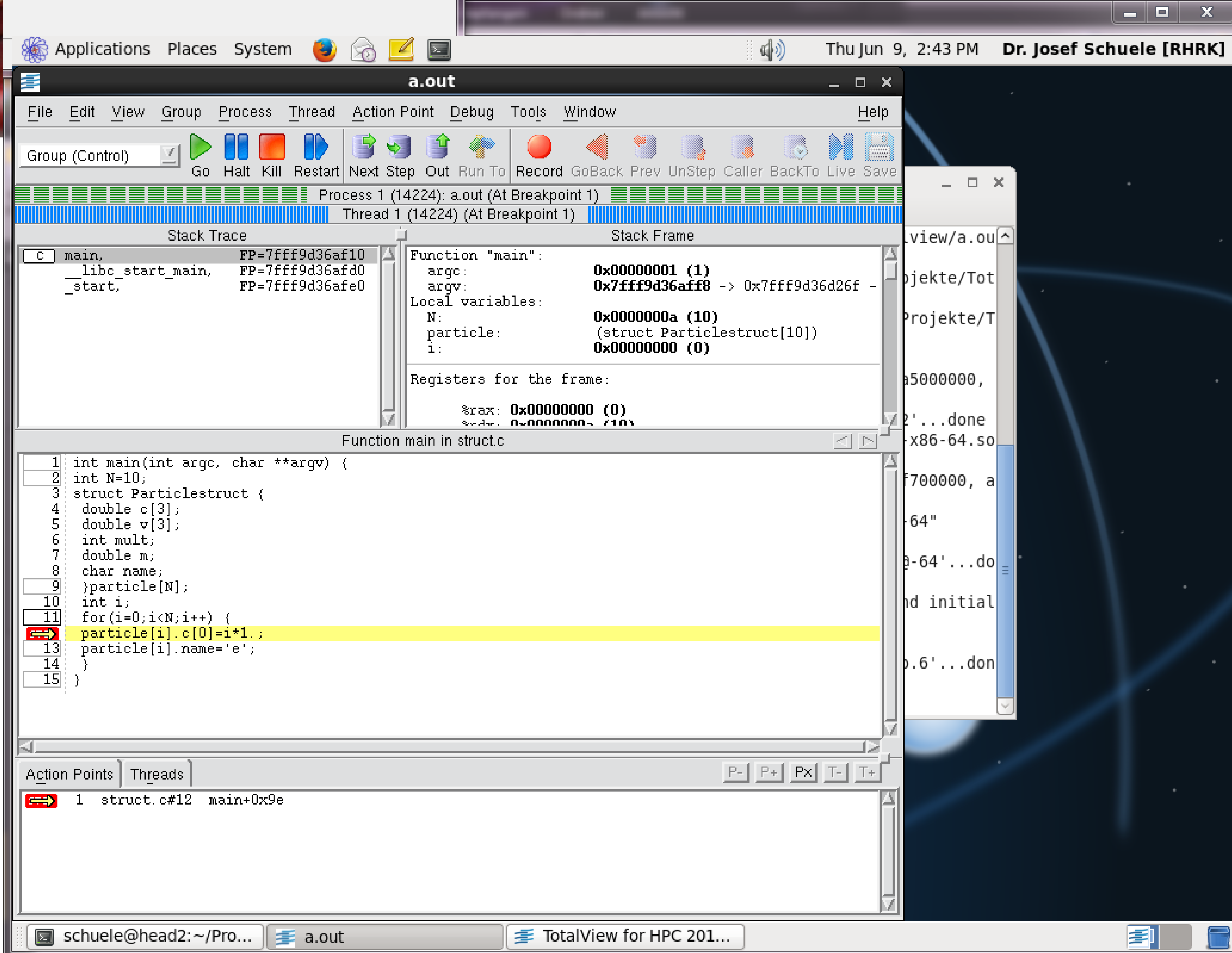
Questions:

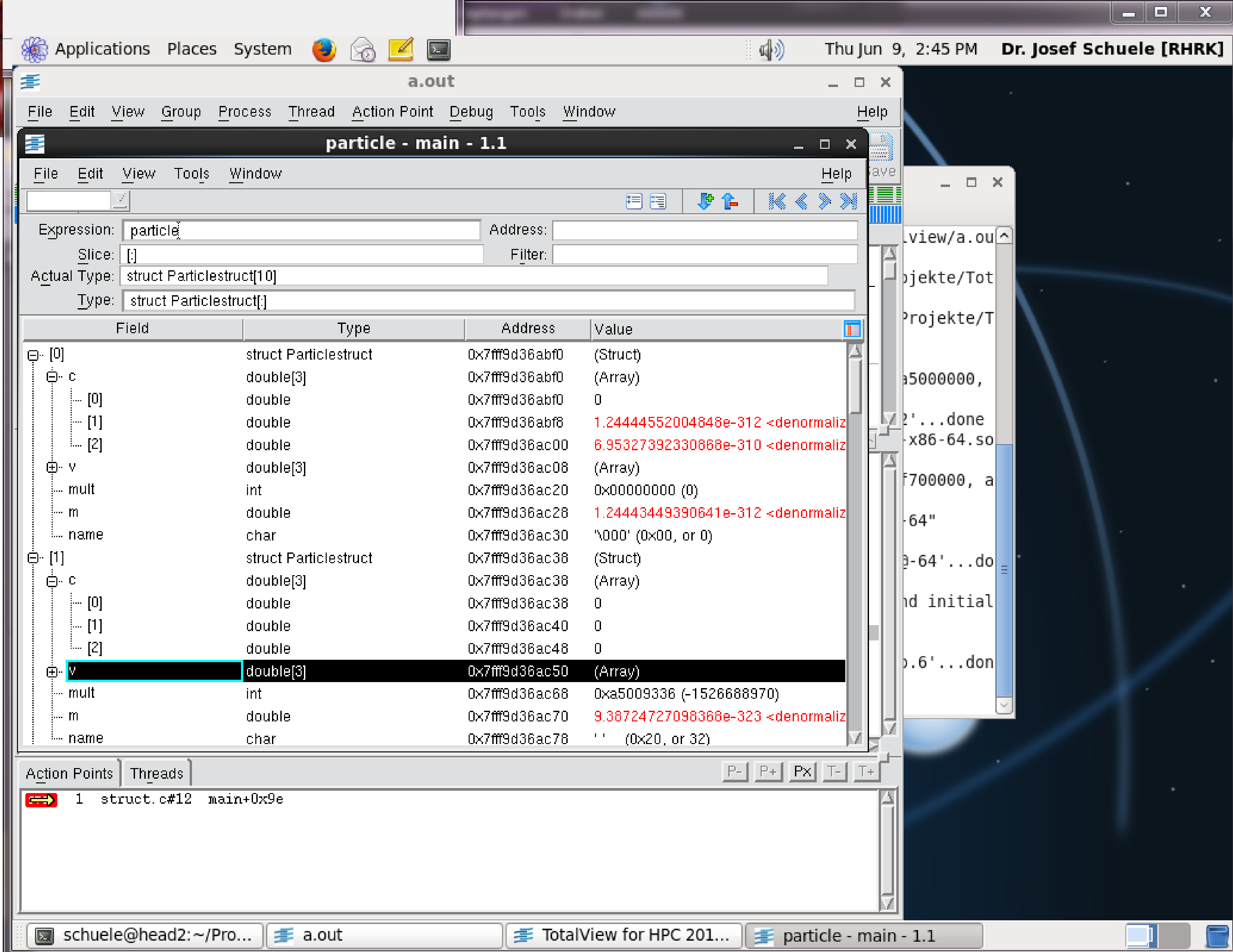
- How is a structure organized in memory?
- How many bytes are occupied by this structure?

```
struct Particlestruct {
 double c[3];  // coordinates
 double v[3];  // velocities
 int hydro;    // water molecules in complex
 double m;     // mass
 char name;    // name (type) of particles
} particle[10];
```

9x8=72 Byte (whereof 61 are used)

To control extends of structures – use debugger totalview.


Applications Places System
Thu Jun 9, 2:43 PM
Dr. Josef Schuele [RHRK]
a.out
File Edit View Group Process Thread Action Point Debug Tools Window Help
Group (Control)
Go Halt Kill Restart Next Step Out Run To Record GoBack Prev UnStep Caller BackTo Live Save
Process 1 (14224): a.out (At Breakpoint 1)
Thread 1 (14224) (At Breakpoint 1)
Stack Trace
C main, FP=7fff9d36af10
__libc_start_main, FP=7fff9d36afd0
_start, FP=7fff9d36afe0
Stack Frame
Function "main":
argc: 0x00000001 (1)
argv: 0x7fff9d36aff8 -> 0x7fff9d36d26f
Local variables:
N: 0x0000000a (10)
particle: (struct Particlestruct[10])
i: 0x00000000 (0)
Registers for the frame:
%rax: 0x00000000 (0)
Function main in struct.c
1 int main(int argc, char **argv) {
2 int N=10;
3 struct Particlestruct {
4 double c[3];
5 double v[3];
6 int mult;
7 double m;
8 char name;
9 }particle[N];
10 int i;
11 for(i=0;i<N;i++) {
12 particle[i].c[0]=i*1.;
13 particle[i].name='e';
14 }
15 }
Action Points Threads
P- P+ Px T- T+
1 struct.c#12 main+0x9e
schuele@head2:~/Pro...
a.out
TotalView for HPC 201...



Applications Places System
Thu Jun 9, 2:45 PM
Dr. Josef Schuele [RHRK]
a.out
File Edit View Group Process Thread Action Point Debug Tools Window Help
particle - main - 1.1
File Edit View Tools Window Help
Expression: particle
Address:
Slice: [:]
Filter:
Actual Type: struct Particlestruct[10]
Type: struct Particlestruct[:]
Field Type Address Value
[0] struct Particlestruct 0x7fff9d36abf0 (Struct)
c double[3] 0x7fff9d36abf0 (Array)
[0] double 0x7fff9d36abf0 0
[1] double 0x7fff9d36abf8 1.24444552004848e-312 <denormaliz
[2] double 0x7fff9d36ac00 6.95327392330868e-310 <denormaliz
v double[3] 0x7fff9d36ac08 (Array)
mult int 0x7fff9d36ac20 0x00000000 (0)
m double 0x7fff9d36ac28 1.24443449390641e-312 <denormaliz
name char 0x7fff9d36ac30 '\000' (0x00, or 0)
[1] struct Particlestruct 0x7fff9d36ac38 (Struct)
c double[3] 0x7fff9d36ac38 (Array)
[0] double 0x7fff9d36ac38 0
[1] double 0x7fff9d36ac40 0
[2] double 0x7fff9d36ac48 0
v double[3] 0x7fff9d36ac50 (Array)
mult int 0x7fff9d36ac68 0xa5009336 (-1526688970)
m double 0x7fff9d36ac70 9.38724727098368e-323 <denormaliz
name char 0x7fff9d36ac78 ' ' (0x20, or 32)
Action Points Threads
P- P+ Px T- T+
1 struct.c#12 main+0x9e
schuele@head2:~/Pro...
a.out
TotalView for HPC 201...
particle - main - 1.1

pack and unpack

Questions:

- How is a structure organized in memory?
- How many bytes are occupied by this structure?
- How much time is required to send (model time τ_c)

```
struct Particlestruct {
 double c[3];  // coordinates
 double v[3];  // velocities
 int hydro;      // water molecules in complex
 double m;      // mass
 char name;   // name (type) of particles
} particle[10];
```

for each communication: Tc(L) = Ts + L/BB

In total:

Tc = (5Ts + (24+24+4+1+8)/BB)*10

pack and unpack

Questions:

- How is a structure organized in memory?
- How many bytes are occupied by this structure?
- How much time is required to send (model time τ_c)
- Why is the following better?

```
struct Particlestruct {
 double c[3*10];  // coordinates
 double v[3*10];  // velocities
 double m[10];     // mass
  int hydro[10];     // water molecules
 char name[10];   // name (type) of particles
} particle;


     MPI_Send(particle.c,30,MPI_DOUBLE,rec,tag,comm);
     MPI_Send(particle.v,30,MPI_DOUBLE,rec,tag,comm);
     MPI_Send(particle.m,10,MPI_DOUBLE,rec,tag,comm);
     MPI_Send(particle.hydro,10,MPI_INT,rec,tag,comm);
     MPI_Send(particle.name,10,MPI_CHAR,rec,tag,comm);
```

pack and unpack

```
MPI_Pack(inbuf, incnt, datatype, outbuf, outcnt, &pos,
comm);
MPI_Unpack(inbuf,incnt,&pos,outbuf,outcnt,datatype,comm);
```

Pack/Unpack data into a buffer (inbuf). This buffer may be sent in one piece.

```
pos=0;
for(i=0;i<10;i++) {
      MPI_Pack(particle[i].c,3,MPI_DOUBLE,&buf,size_buf,&pos,comm);
      MPI_Pack (particle[i].v,3,MPI_DOUBLE,&buf,size_buf,&pos,comm);
      MPI_Pack (particle[i].m,1,MPI_DOUBLE,&buf,size_buf,&pos,comm);
      MPI_Pack(particl[i].hydro,1,MPI_INT,&buf,size_buf,&pos,comm);
      MPI_Pack (particle[i].name,1,MPI_CHAR,&buf,size_buf,&pos,comm);
}
MPI_Send(buf,size_buf,MPI_PACKED,rec,tag,comm);
```

cancel

Messages may be cancelled thus
- freeing buffers (resources) and
- allow to remove dangling send operations to broken threads.

```
MPI_Cancel(&request)
```

End of (most) information about point-to-point communication.

Einführung in das Hochleistungsrechnen

Introduction to High Performance Computing

VIELEN DANK
THANK YOU