



Universität Hamburg

DER FORSCHUNG | DER LEHRE | DER BILDUNG

Praktikum: Paralleles Programmieren für Geowissenschaftler

Prof. Thomas Ludwig, Hermann Lenhart, Ulrich Körner, Nathanael Hübbe



Dr. Hermann-J. Lenhart

hermann.lenhart@zmaw.de



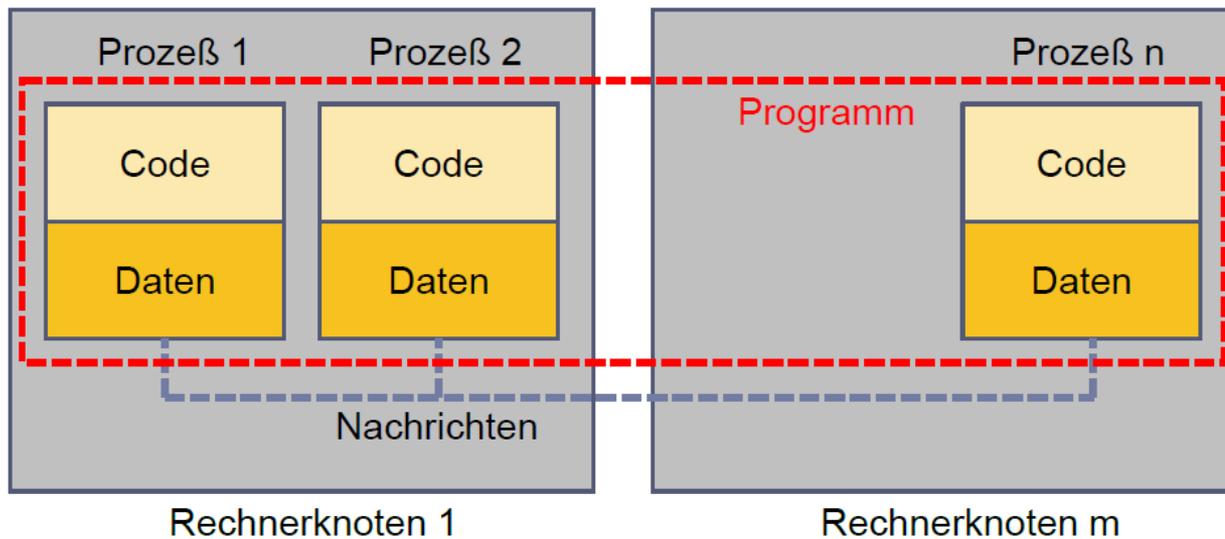
MPI Einführung II:

- Nachrichtenaustausch
- Send/Receive Syntax
- MPI Umgebungsvariablen
- Broadcast
- Reduce Operation
- Gather Operation



MPI – Hardware Voraussetzung

(nach Ludwig WS12/13)



- Keinen direkten Zugriff auf Memory (Daten) von anderen Prozessen.
- Datenverfügbarkeit über expliziten Datenaustausch (Senden/Empfangen) mit anderen Prozessen!



MPI Nachrichtenaustausch

MPI Nachrichten sind Datenpakete die zwischen Prozessen ausgetauscht werden.

Der Nachrichtenaustausch bedarf folgender Informationen:

- Sender Prozess
- Datentyp
- Datenlänge
- Empfangender Prozess
- Status der Nachricht
- Nachrichtenumgebung (z.B. wieviele Prozesse sind vorhanden?)



MPI Nachrichtenaustausch

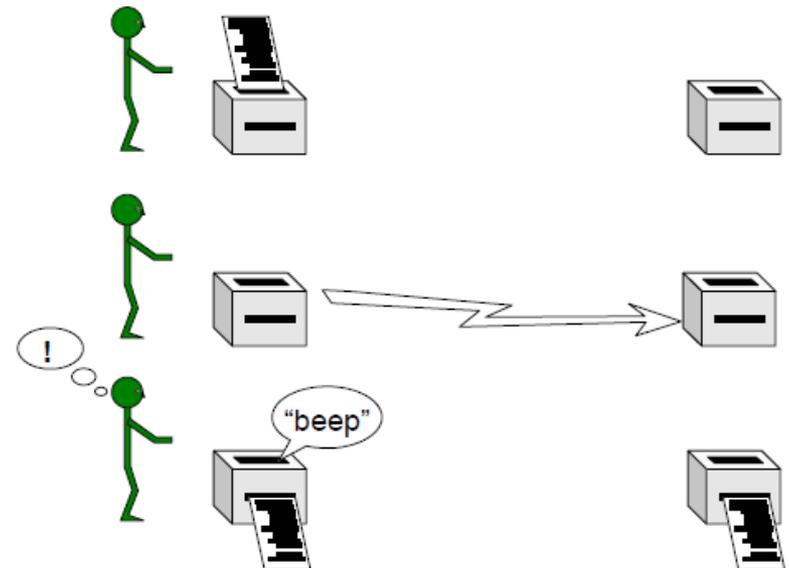
Der MPI Nachrichtenaustausch im Vergleich zum FAX

Für einfachste Art der MPI Kommunikation -
Point to Point Communication:

Ein Prozess sendet eine Nachricht
zu einem Anderen,

und erhält Information über die
ordnungsgemäße Zustellung (!)

(Wolfgang Baumann ZIB, 2009;
Parallel Programming with MPI)





MPI Send/Receive Syntax I

MPI_SEND(Message, Count, Datatype, Dest, Tag, Comm, lerror)

z.B:

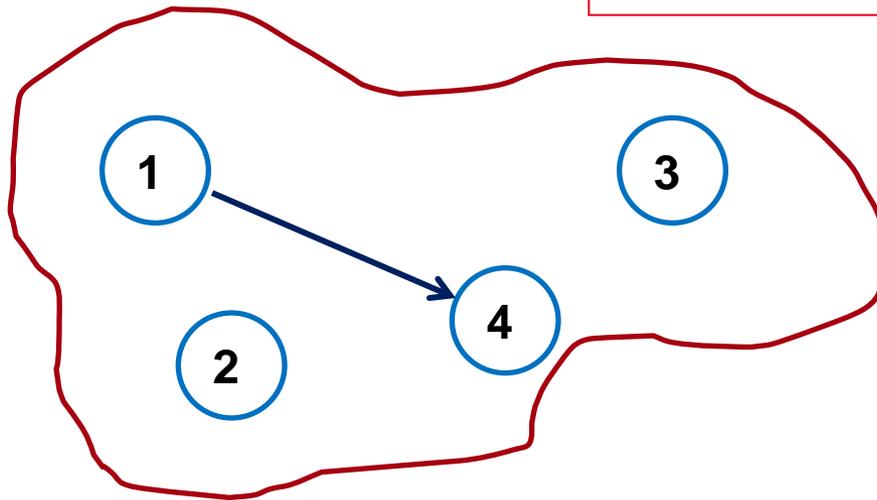
Call MPI_SEND(temp, 1, MPI_Real, dest, tag, MPI_COMM_World, lerror)

temp	Adresse des Sendepuffers; Real :: temp
1	Count – Anzahl der Elemente im Puffer
MPI_Real	Datentyp des gesendeten Elementes
dest	Angabe des Ranges des Zielprozesses; integer :: dest
tag	Nachrichtenkennung; integer :: tag
MPI_COMM_World	Kommunikator (Gruppe, Kontext)
lerror	Fehlerstatus; integer :: lerror



MPI Point to Point Communication:

Kommunikator: MPI_COMM_WORLD



Send -> Receive



MPI Send/Receive Syntax II

MPI Datentypen in Anlehnung an Fortran

MPI Datentyp

FORTRAN Datentyp

MPI_INTEGER

INTEGER

MPI_REAL

REAL

MPI_DOUBLE_PRECISION

DOUBLE PRECISION

MPI_LOGICAL

LOGICAL

MPI_CHARACTER

CHARACTER(1)



MPI Send/Receive Syntax III

MPI_RECV(Message, Count, Datatype, **Source**, Tag, Comm, **status**, lerror)

Call MPI_RECV(temp, 1, MPI_Real, dest, tag, MPI_COMM_World, status, lerror)

temp Adresse des Sendepuffers; Real :: Vector

1 Count – Anzahl der Elemente im Puffer

MPI_Real Datentyp des gesendeten Elementes

source **Angabe des Ranges des Sendeprozesses; integer :: source**

tag Nachrichtenennung (Reihenfolge); integer :: tag

MPI_COMM_World Kommunikator (Gruppe, Kontext)

status **Empfangsstatus der Nachricht (angekommen?); integer status(MPI_STATUS_SIZE)**

lerror Fehlerstatus; integer :: lerror



MPI Send/Receive Syntax III

MPI_RECV(Message, Count, Datatype, **Source**, Tag, Comm, **status**, lerror)

Call MPI_RECV(temp, 1, MPI_Real, dest, tag, MPI_COMM_World, status, lerror)

temp Adresse des Sendepuffers; Real :: Vector

1 Count – Anzahl der Elemente im Puffer

MPI_Real Datentyp des gesendeten Elementes

source **Angabe des Ranges des Sendeprozesses; integer :: source**

tag Nachrichtenennung (Reihenfolge); integer :: tag

MPI_COMM_World Kommunikator (Gruppe, Kontext)

status **Empfangsstatus der Nachricht (angekommen?); integer status(MPI_STATUS_SIZE)**

lerror Fehlerstatus; integer :: lerror



MPI Umgebungsvariablen I

Für die „Bestückung“ der Send/Receive Aufrufe sowie für allgemeine Infos stehen folgende Befehle zur Verfügung um die MPI Umgebung zu erfragen.

MPI_Comm_size Wieviele Prozesse sind aktiv

MPI_Comm_rank Welchen Rang hat der aktuelle Prozess



MPI Umgebungsvariablen II

Program hello

```
use mpi
```

```
INTEGER :: ierr, rank, size
```

```
CALL MPI_INIT(ierr)
```

```
CALL MPI_COMM_RANK(MPI_COMM_WORLD,rank,ierr)
```

```
CALL MPI_COMM_SIZE (MPI_COMM_WORLD, size,ierr)
```

```
Print*, ' I am ',rank,' of ',size
```

```
CALL MPI_FINALIZE(ierr)
```

End

Alle Prozesse starten gleichzeitig!



MPI Kollektive Operationen

MPI verfügt über umfangreiche Operationen zum kollektiven Bewegen von Daten.

Dazu gehören:

MPI_BROADCAST

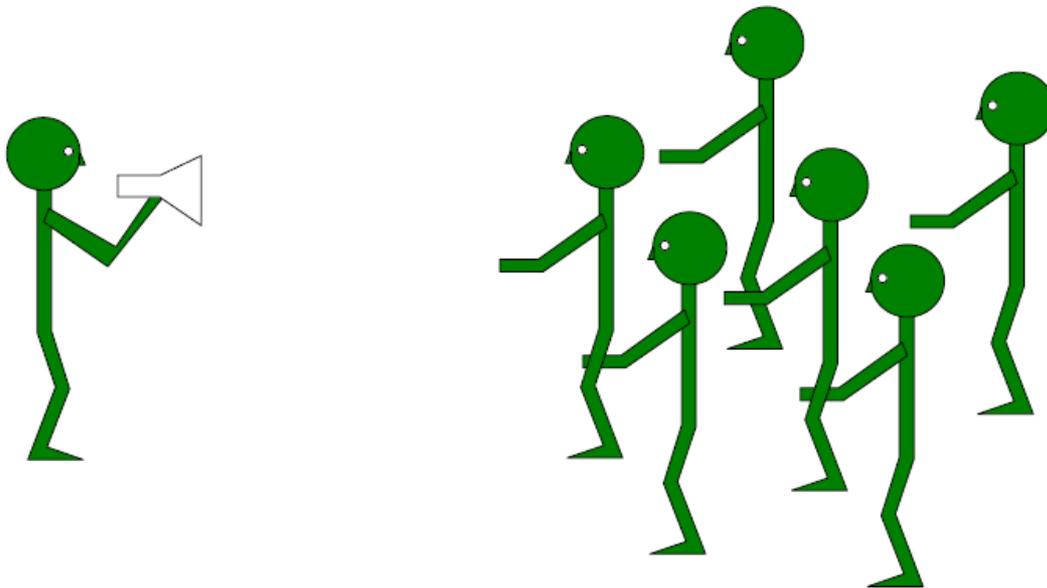
MPI_REDUCE

MPI_GATHER

MPI_SCATTER



MPI Broadcast I



(Wolfgang Baumann ZIB, 2009;
Parallel Programming with MPI)



MPI Broadcast

Neben dem Versenden von Nachrichten zwischen einzelnen Prozesse mittels Call `MPI_SEND(temp, 1, MPI_Real, dest, tag, MPI_COMM_World, lerror)`

gibt es auch die Möglichkeit **eine Nachricht an alle anderen Prozesse** zu senden:

`MPI_BCAST(Message, Count, Datatype, Root, Comm, lerror)`

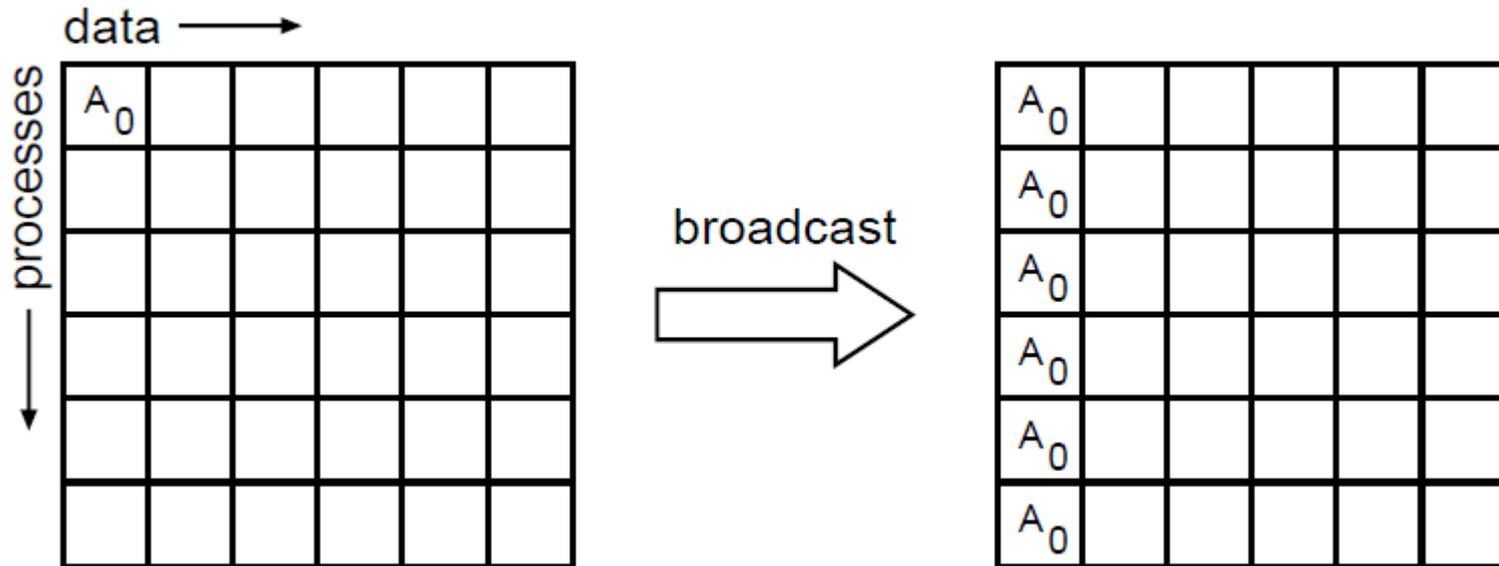
Call `MPI_BCAST(temp, 1, MPI_Real, source, MPI_COMM_World, lerror)`

Für Initialisierung oder zum Programmabbruch genutzt.



MPI Broadcast III

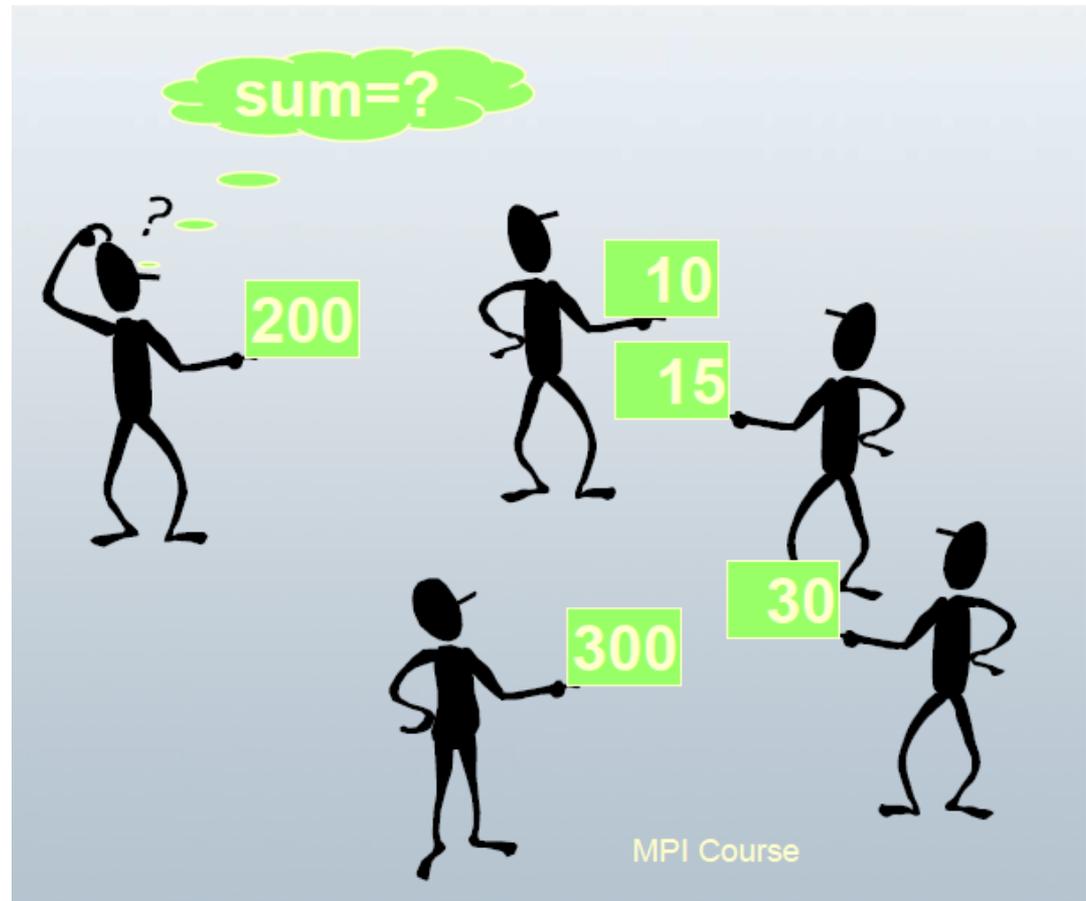
Verwendung zur Initialisierung



(William Gropp ANL,
MPI Tutorial)



MPI Reduce I



DKRZ MPI Einführungs Kurs



MPI Reduce II

Um die Ergebnisse der einzelnen Prozesse zusammenzuführen gibt es eine Auswahl an „Reduce“ Operationen, z.B:

`MPI_REDUCE(Operand, Result, Count, Datatype, Operation, Root, Comm, Ierror)`

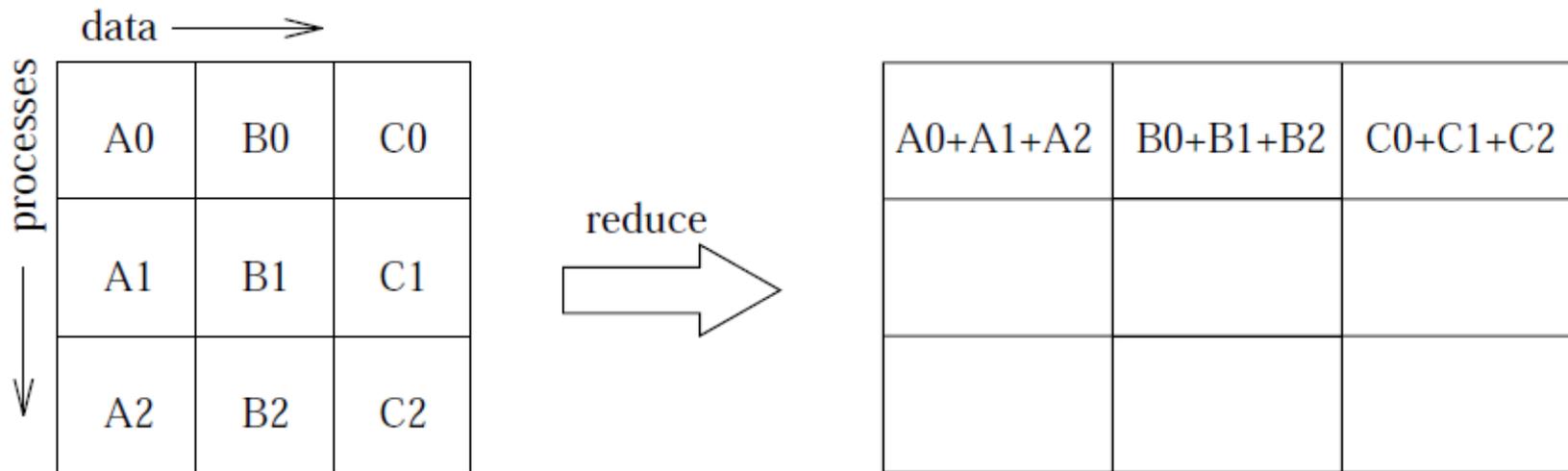
Call `MPI_REDUCE(temp, sum, 1, MPI_Real, MPI_SUM, 0, MPI_COMM_World, Ierror)`

Über die Operation `MPI_SUM` werden alle Resultate der Größe `temp` von allen Prozessen aufaddiert und in der Variable `sum` abgelegt.



MPI Reduce III

Operator SUM



(William Gropp ANL,
MPI Tutorial)



MPI Reduce IV

Übersicht der möglichen MPI_Reduce Operationen:

MPI_SUM	Summe
MPI_PROD	Produkt
MPI_MAX /MPI_MIN	Maximum/Minimum
MPI_MAXLOC	Maximum und Position des Maximums
MPI_LAND / MPI_LOR	Logical And / Logical Or



MPI Gather I

Eine weitere Möglichkeit Teilarrays der Prozesse (z.B. für I/O Zwecke) zusammenzuführen, bietet MPI_GATHER:

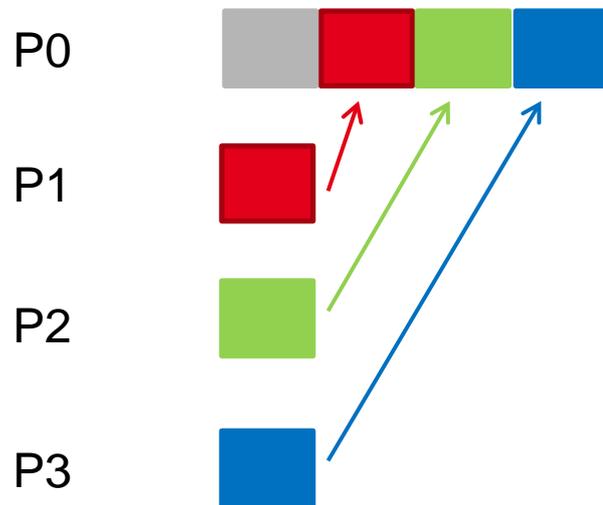
Syntax: MPI_Gather(Sendmessage, Sendcount, Sendtype,
Recvmessage, Recvcount, Recvtype,
Root, Comm, lerror)

Call MPI_GATHER (temp, 1, MPI_Real,
temps,1, MPI_Real, 0, MPI_COMM_World, lerror)



MPI Gather II

Call `MPI_GATHER(temp, 1, MPI_Real, tempAll, 1, MPI_Real, 0, MPI_COMM_World, lerror)`





MPI Scatter I

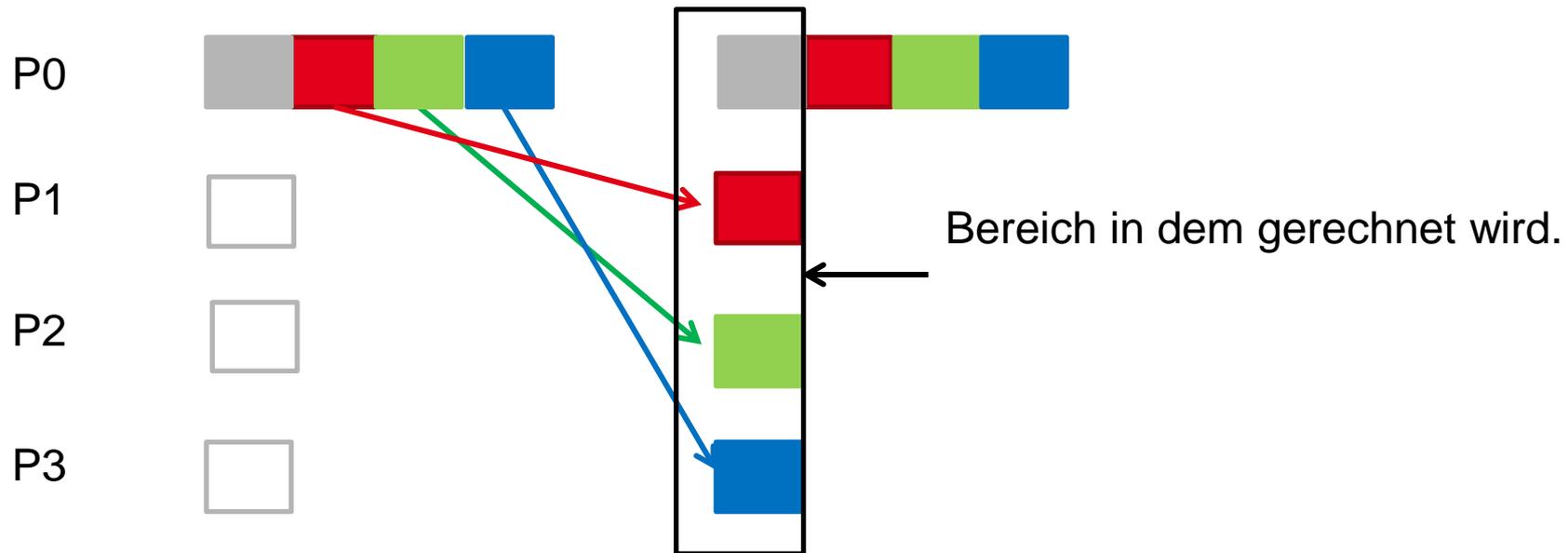
Eine Möglichkeit z.B. eine Anfangsbelegung auf die Teilarrays der Prozesse ,
zu übertragen bietet MPI_SCATTER:

Syntax: MPI_Scatter(Sendmessage, Sendcount, Sendtype,
Recvmessage, Recvcount, Recvtype,
Root, Comm, lerror)



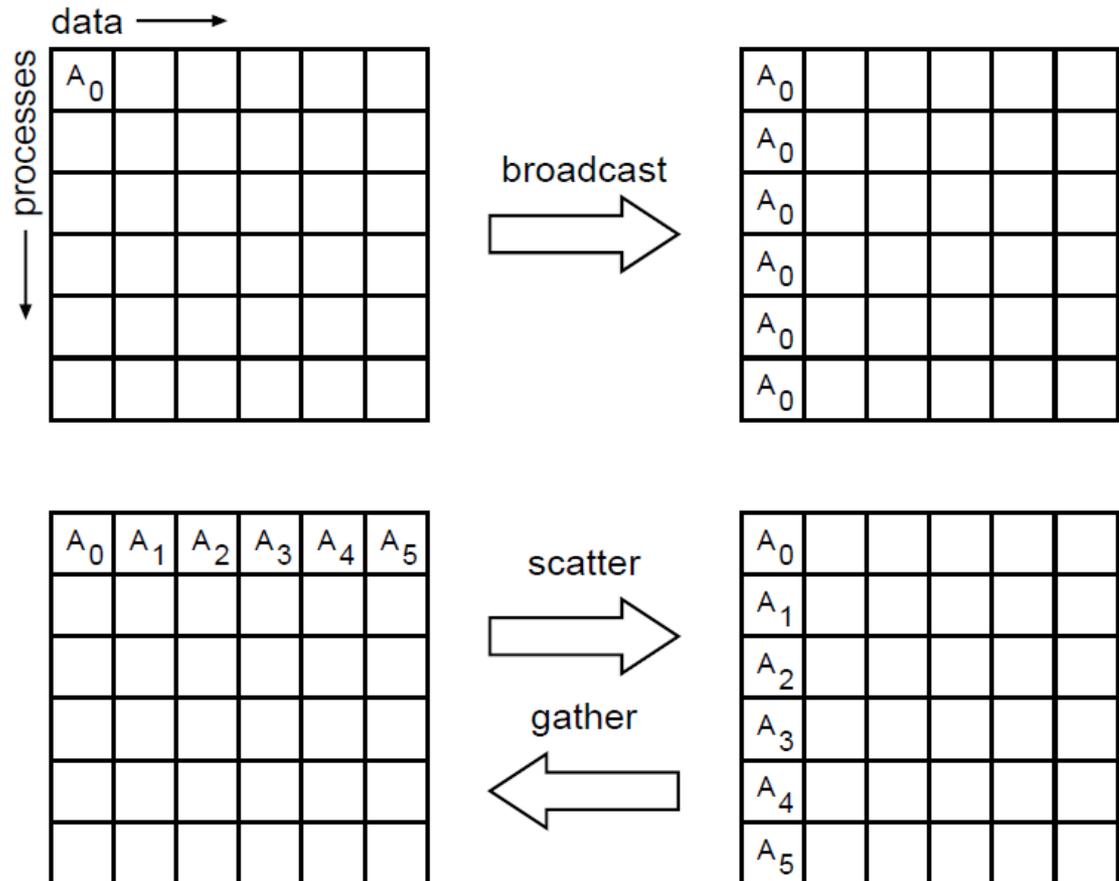
MPI Scatter II

Die Daten werden von P0 an die anderen Prozesse P1 – P3 gesendet.





MPI Scatter-Gather



(William Gropp ANL,
MPI Tutorial)



Danke das wars!