MPI-Datentypen

07.06.2021

Gliederung

- Was ist MPI?
- Wofür wird MPI eingesetzt?
- Wofür braucht man MPI-Datentypen?
- Welche Datentypen gibt es
- Wie sind sie aufgebaut?

Was ist MPI?

MPI = Message Passing Interface

- Strukturvorgabe für Kommunikation zwischen Computern
  - Keine Programmiersprache, meistens in C oder FORTRAN
- Optimiert für Effizienz & Schnelligkeit
- Aktuelle Version: 4.1.1

Wofür wird MPI eingesetzt?

- PC-Cluster
  - Mehrere, zusammengehörende Rechner
  - z.B. Rechenzentren
- Parallelrechner
  - Ein Rechner mit vielen Prozessoren
  - z.B. Supercomputer

Parallelisierung

- Teilprozesse auf Prozessoren verteilt
  - Keinen Zugriff auf Daten der anderen Prozesse → Datenaustausch
- Berechnungen sehr viel schneller als Datenaustausch
  - Datenaustausch muss so effizient wie möglich gestaltet werden

Kommunikation

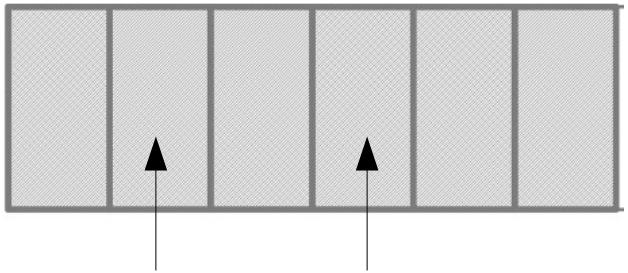
- int MPI_Send (void* buf, int count, MPI_Datatype datatype, int dest, int tag, MPI_Comm comm)
  - buf: Zeiger auf den Datenbeginn im Speicher
  - count: Zahl der Elemente, die zu senden sind
  - datatype: Datentyp der Elemente
  - dest: Rang des Zielprozesses
  - tag: Markierung der Nachricht
  - comm: Kommunikator der Prozessgruppe

Kommunikation

- int MPI_Recv (void* buf, int count, MPI_Datatype datatype, int source, int tag, MPI_Comm comm, MPI_Status* status)
  - buf, count, datatype, comm, tag: Wie bei send
  - source: Rang des Quellprozesses
  - status: Zeiger auf eine Statusstruktur, in der Informationen über die empfangene Nachricht abgelegt werden sollen

Wofür braucht man MPI-Datentypen?

- Effizientere Gestaltung des Sendens von Daten
  - Kein seperates “Vorspeichern” mehr nötig
  - Übermittelung der Länge und Interpretation der Daten



Vereinfachte Darstellung von Daten im Speicher.
Quelle:https://pages.tacc.utexas.edu/~eijkhout/pcse/html/mpi-data.html

Wofür braucht man MPI-Datentypen?

- Komplexere Datenstrukturen können einfacher abgebildet werden
  - z.B. Matrizen
- Mögliche Problemfaktoren durch Unterschiede zwischen Systemen fallen weg
  - MPI übersetzt Daten korrekt
  - z.B. Big / Little Endian

Welche Datentypen gibt es?

- Elementare Datentypen (vordefiniert)
- Array-Datentyp (Contigous)
- Vektor-Datentyp
- Indexed-Array-Datentyp
- Struct-Datentyp
- Benutzerdefiniert

Elementare Datentypen

`char`	*MPI_CHAR*
`unsigned char`	*MPI_UNSIGNED_CHAR*
`char`	*MPI_SIGNED_CHAR*
`short`	*MPI_SHORT*
`unsigned short`	*MPI_UNSIGNED_SHORT*
`int`	*MPI_INT*
`unsigned int`	*MPI_UNSIGNED*
`long int`	*MPI_LONG*
`unsigned long int`	*MPI_UNSIGNED_LONG*
`long long int`	*MPI_LONG_LONG_INT*
`float`	*MPI_FLOAT*
`double`	*MPI_DOUBLE*
`long double`	*MPI_LONG_DOUBLE*
`unsigned char`	*MPI_BYTE*

Vergleich von Typen in C und MPI.
Quelle: https://pages.tacc.utexas.edu/~eijkhout/pcse/html/mpi-data.html

Array-Datentyp (Contiguous)

- Arrays elementarer Datentypen
  - Elemente eines Typs
- Speicherzellen direkt hintereinander

Elemente eines Arrays im Speicher.
Quelle: https://pages.tacc.utexas.edu/~eijkhout/pcse/html/graphics/data-contiguous.jpeg

Array-Datentyp (Contiguous)

int MPI_Type_contiguous

(int count, MPI_Datatype oldtype, MPI_Datatype *newtype)

- Input count: Anzahl der Elemente
- Input oldtype: Alter Datentyp
- Output newtype: Neuer Datentyp

Vektor-Datentyp

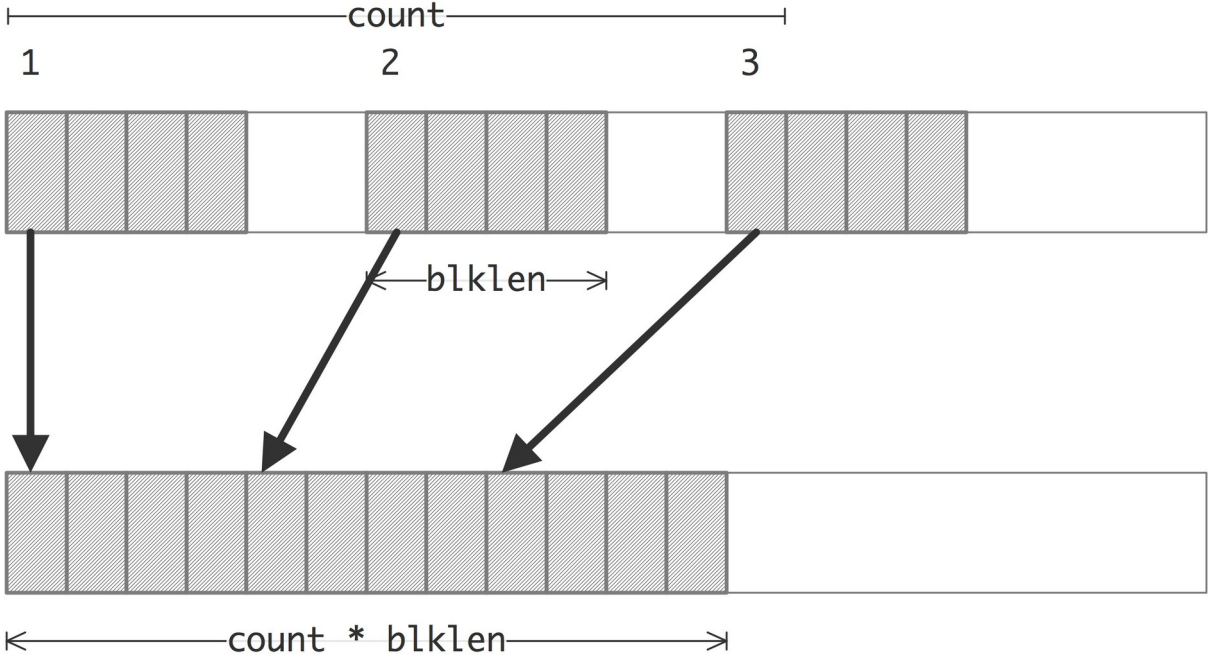
- Sehr ähnlich zu Arrays
  - Elemente eines Datentyps
- Elemente liegen nicht direkt nebeneinander im Speicher
  - Regelmäßige Abstände
- Empfänger erhält Elemente als Array

Vektor-Datentyp

```
int MPI_Type_vector
   (int count, int blocklength, int stride,
    MPI_Datatype oldtype, MPI_Datatype *newtype)
```

- Input blocklength: Anzahl Elemente pro Datenblock
- Input stride: Abstand des Starts der Blöcke

Vektor-Datentyp



Zusammenfassung der Elemente zu einem Array.
Quelle: https://pages.tacc.utexas.edu/~eijkhout/pcse/html/graphics/data-vector-to-contiguous.jpeg

Indexed-Array-Datentyp

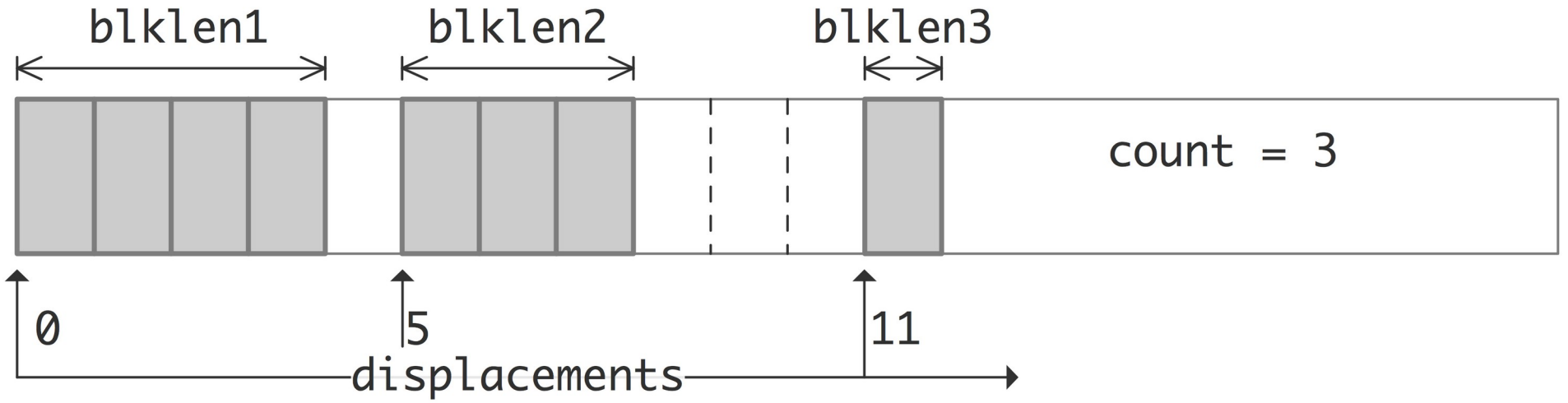
- Array aus Speicheradressen von Datenblöcken
  - Elemente eines Datentyps
- Elemente können unterschiedlich große Abstände haben

Indexed-Array-Datentyp

```
int MPI_Type_indexed(int count,
    const int array_of_blocklengths[],
    const int array_of_displacements[],
    MPI_Datatype oldtype, MPI_Datatype
    *newtype)
```

- Input displacement: Abstände der Blöcke in Vielfachen von old_type

Indexed-Array-Datentyp



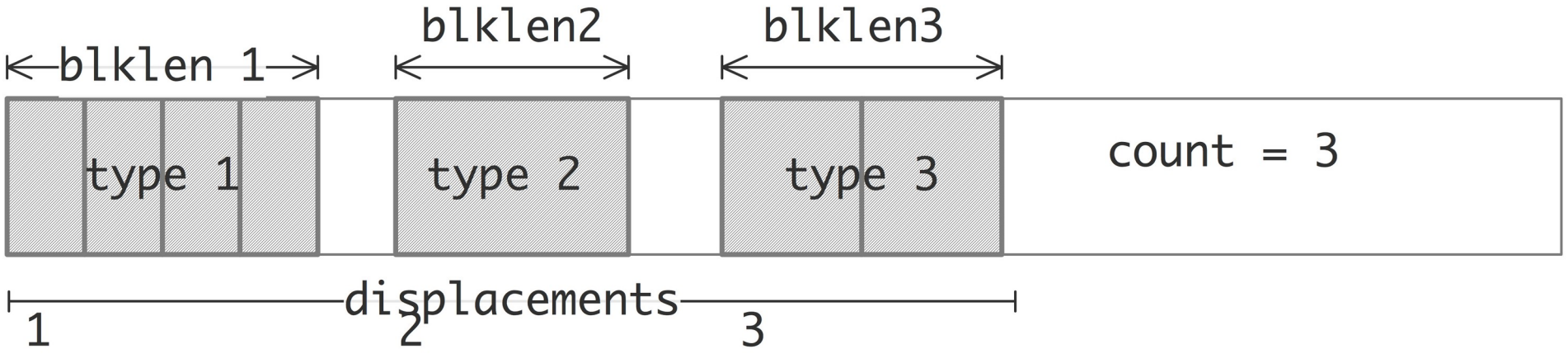
Zusammensetzung eines Indexed Array.
Quelle: https://pages.tacc.utexas.edu/~eijkhout/pcse/html/graphics/data-indexed.jpeg

Struct-Datentyp

- Struktur in C
- Kann mehrere Datentypen enthalten

```
int MPI_Type_create_struct(
  int count, int blocklengths[], MPI_Aint displacements[],
  MPI_Datatype types[], MPI_Datatype *newtype);
```

Struct-Datentyp



Zusammensetzung einer Datenstruktur.
Quelle: https://pages.tacc.utexas.edu/~eijkhout/pcse/html/graphics/data-struct.jpeg

Datentyp erstellen

- Für jeden nicht – elementaren Datentyp:

- Variable für den Datentyp erstellen
- Create call mit allen nötigen Eingabeparametern
- Commit call für interne Mitschrift
- Wenn nicht mehr benötigt: freigeben

Datentyp erstellen

```
MPI_Datatype newtype;
MPI_Type_something( < oldtype specifications >, &newtype );
MPI_Type_commit( &newtype );
/* code that uses your new type */
MPI_Type_free( &newtype );
```

Extent

- Alle Datentypen haben Attribut Extent
  - Abstand vom ersten zum letzten Element des Typs, inkl. Lücken
    - In Bytes gemessen
  - Kann über getter-Methode abgefragt werden

Beispiel-Kommando

int MPI_Send (void* buf, int count, MPI_Datatype datatype, int dest, int tag, MPI_Comm comm)

MPI_Send(message, strlen(message)+1, MPI_CHAR, 0, tag, MPI_COMM_WORLD);

- MPI_COMM_WORLD: Summe aller Prozesse

Zusammenfassung

- MPI ist Standard für Kommunikation zwischen Programmen auf mehreren Rechnern
- Wegen Optimierung bezüglich Effizienz und Performanz Datentypen wichtiger Bestandteil
  - Tragen maßgebend dazu bei
- Entweder elementare Datentypen oder aus ihnen zusammengesetzt

Quellen

- https://pages.tacc.utexas.edu/~eijkhout/pcse/html/mpi-data.html
- https://de.wikipedia.org/wiki/Message_Passing_Interface
- https://www.tutorialspoint.com/cprogramming/c_structures.htm
- http://wgropp.cs.illinois.edu/courses/cs598-s16/lectures/lecture27a.pdf
- https://www.mpi-forum.org/docs/mpi-3.1/mpi31-report.pdf
- https://www.open-mpi.org