Verteilte Algorithmen und Datenstrukturen

Kapitel 4: Einführung in verteilte Programmierung

Prof. Dr. Christian Scheideler

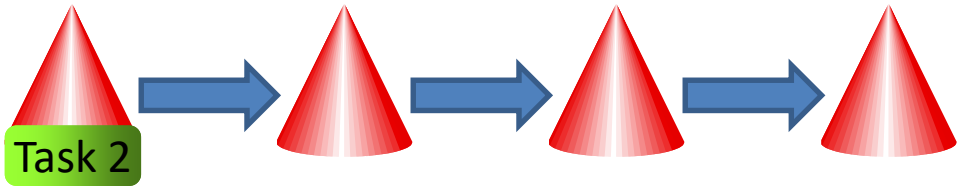
Insitut für Informatik

Universität Paderborn

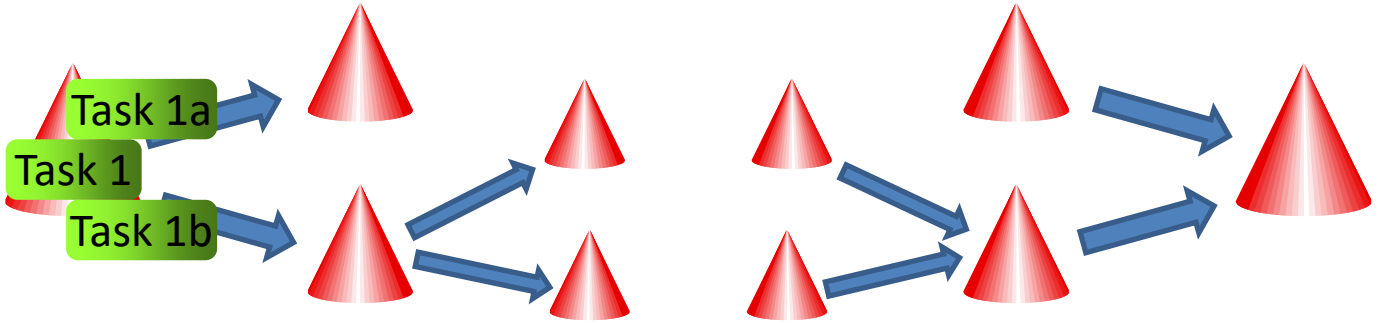
Verteilte Programmierung

Grundprinzip: Arbeitsteilung

- Sequentielle Arbeitsteilung (Flow-Based Progr.)



- Parallele Arbeitsteilung (MapReduce Paradigma)



Verteilte Programmierung

Zoo an Programmiersprachen:

Actor Modell:

- Axum, Elixir, Erlang, Janus, SALSA, Scala, Smalltalk,…

Dataflow:

- CAL, E, Joule, LabView, Lustre, Signal, SISAL,…

Verteilt:

- Bloom, Emerald, Go, Julia, Limbo, MPD, Oz, Sequoia, SR,…

Funktional:

- Concurrent Haskell, Concurrent ML, Clojure, Elixir, Erlang, Id, MultiLisp, SequenceL,…

Multithreaded:

- C=, Cilk, Clojure, Go, Java, ParaSail, SequenceL,…

Objektorientiert:

- mC++, Ada, C*, C++ AMP, Charm++, D, Emerald, Go, Java, ParaSail, Smalltalk,…

Siehe z.B. http://en.wikipedia.org/wiki/Concurrent_computing

Verteilte Programmierung

Unser Ansatz:

- Prozesse, die sich beliebig miteinander vernetzen können
- Prozesse können eigenständig sequentiell Aktionen ausführen
- Prozesse arbeiten und kommunizieren asynchron
- Daten müssen Prozessen zugeordnet sein, da es keinen Speicher außerhalb von Prozessen gibt

Pseudocode

Wie in objektorientierter Programmierung:

Subject ⟨Subjektname⟩: // deklariert Prozesstyp
    lokale Variablen
    Aktionen

Allgemeine Formen einer Aktion:

⟨Aktionsname⟩(Objektliste) →
    Befehle in Pseudocode

⟨Aktionsname⟩: ⟨Prädikat⟩ →
    Befehle in Pseudocode

Spezielle Aktionen:

init(Objektliste) → // Konstruktur
    Befehle in Pseudocode

timeout → // bei jedem timeout (was periodisch erfolgt)
    Befehle in Pseudocode

Pseudocode

- Zuweisung durch :=
- Schleifen (for, while, repeat)
- Bedingtes Verzweigen (if – then – else)
- (Unter-)Programmaufruf/Übergabe (return)
- Kommentar durch { }
- Blockstruktur durch Einrückung
- Aufruf einer Aktion über Subjektreferenz: $\leftarrow$
- Referenzvariable leer: $\perp$, Menge leer: $\varnothing$
- Erzeugung neuer Subjekte und Objekte: new
  (new aktiviert init im Subjekt)

Beispiele

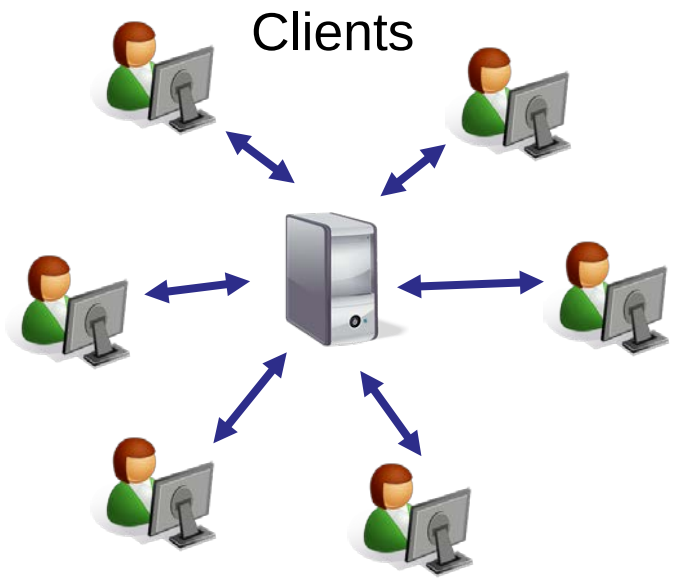
- Hello World: erzeugt Prozess, der „Hello World“ ausgibt
- Client & Server: erzeugt Server mit Broadcast Service und Clients
- Grid & Node: erzeugt ein Gitter und demonstriert das x-y Routing

Alle Programme dazu sind auf der Vorlesungs-webseite zu finden.

Hello World

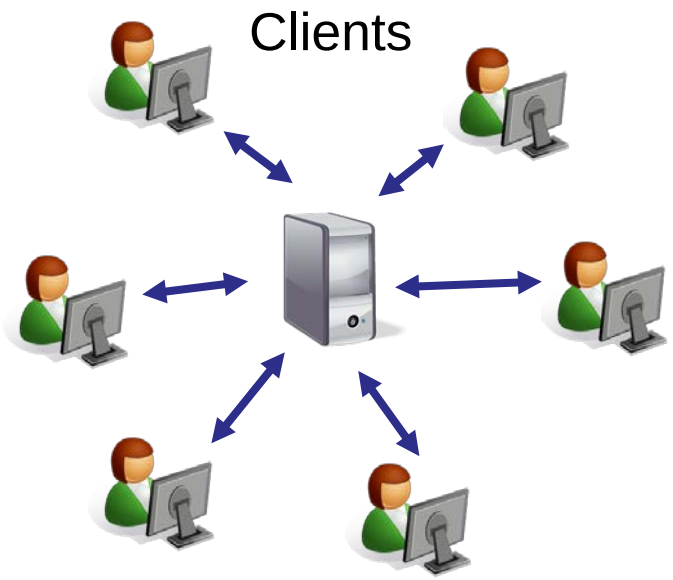

```
Hello_World {
    init(){
        print(“Hello World!”);
    }
}
```

Broadcast Beispiel



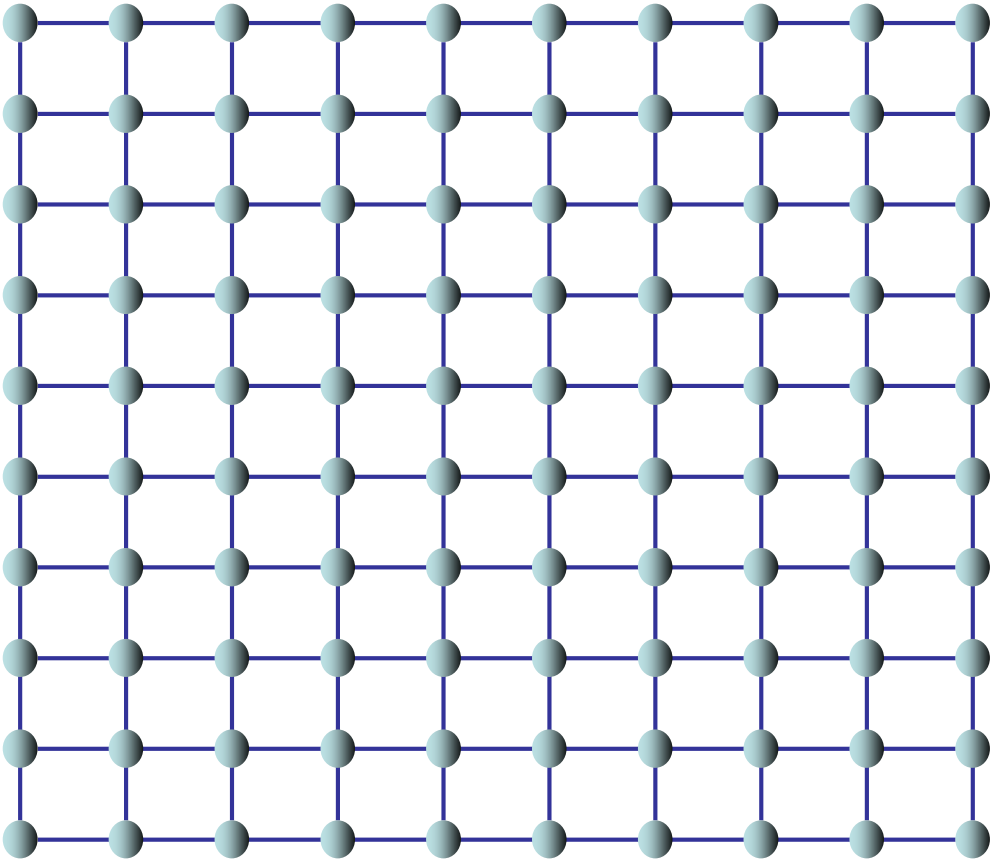

```
Server {
    node [] clients;

    entry(node C){
        int i;
        /* check for duplicate entries */
        for(i=0;i<length(clients);i++){
            if(clients[i]==C) {return;}
        }
        clients>>C; /*adds C to end of list*/
    }

    broadcast(string M){
        int i;
        for (i=0;i<length(clients);i++){
            clients[i] -> write(M);
        }
    }
}
```

Broadcast Beispiel




```
Client {
    node server;

    entry(node S){
        if (server == null){
            server = S;
            server -> entry(this);
        }
    }

    write(string M){print(M);}

    broadcast(string M){
            server -> broadcast(M);
    }
}
```

Gitter Beispiel

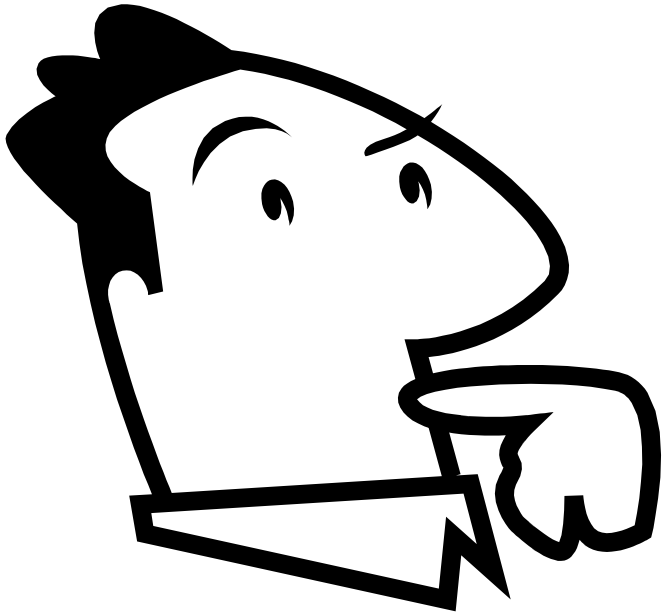


Gitter Beispiel

```
gridNode{
    /* Quadratwurzel der Gesamtzahl aller Knoten */
    int size = 5;
    node top; node left; node bottom; node right;
    int x_this; int y_this;

    init(){
        x_this=(id(this)-1)/size;
        y_this=(id(this)-1)%size;
    }

    entry(node n){
        int x_n=(id(n)-1)/size;
        int y_n=(id(n)-1)%size;
        if (x_n == x_this+1 & y_n == y_this) right=n;
        if (x_n == x_this & y_n == y_this+1) top=n;
        if (x_n == x_this-1 & y_n == y_this) left=n;
        if (x_n == x_this & y_n == y_this-1) bottom=n;
    }
```

Gitter Beispiel

```
message (string text, int to_id){
    int x_n=(to_id-1)/size;
    int y_n=(to_id-1)%size;
    if (x_this != x_n) {
        if (x_n < x_this)
            left -> message(text, to_id);
        else
            right -> message(text, to_id);
    } else {
        if (y_n < y_this)
            bottom -> message(text, to_id);
            if (y_n > y_this)
                top -> message(text, to_id);
            if (y_n == y_this)
                print("Message " . text . " received.");
    }
    mark(true);
}

/* um bei allen Knoten die Markierung zu entfernen.
   Sollte auf Knoten Nr. 1 ausgeführt werden. */
unmark(){
    mark(false);
    right->unmark(); top->unmark();
}
}
```



Fragen?