Ein MPI Beispiel

```
#include <mpi.h>
int numInstances, myRank;
int main(int argc, char *argv[]) {
  // initialize MPI
  MPI_Init(&argc, &argv);
    // get number of instances and my rank (id)
  MPI_Comm_size(MPI_COMM_WORLD,&numInstances);
  MPI_Comm_rank(MPI_COMM_WORLD,&myRank);
  printf("Found %d instances; I'm #%d\n",
          numInstances,myRank);
  doWork();
  // finished
  MPI_Finalize();
}
```

MPI Programmskelett

```
void doWork()
{ char msg[50];      MPI_Status state;
  if (myRank == 0) {                    // I'm the sender
    MPI_Send(msg,50,   // = size of the mesage
             MPI_CHAR, // type
             1,        // rank of target process
             0,        // message tag
             MPI_COMM_WORLD);
  } else if (myRank ==1) {             // I'm the receiver
    MPI_Recv(msg,50,MPI_CHAR,0,0,MPI_COMM_WORLD,
             &state);
    printf("I rcvd. the msg. of len. %d from %d, tag %d.\n",
           state.st_length,state.MPI_SOURCE, state.MPI_TAG);
    printf("Error code is %d.\n", state.MPI_ERROR);
  }
}
```

Kompilieren/Starten

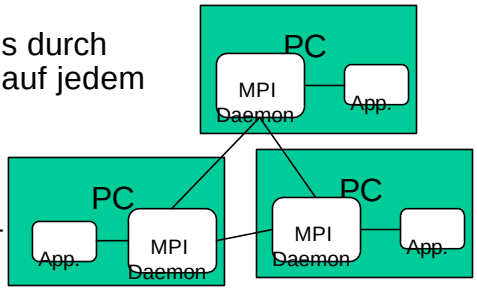
- Kompilation mit einem Wrapper:
  `> hcc demoMPI.c –o demoMPI`
- Hochfahren des MPI-Systems (später...)
- Starten des Programms

```
> mpirun –np 2 demoMPI
Found 2 instances; I'm #0
Found 2 instances; I'm #1
I rcvd. the message of length 50 from 0 with
tag 0.
Error code is 0.
```

- Runterfahren des Systems (später...)

LAM MPI

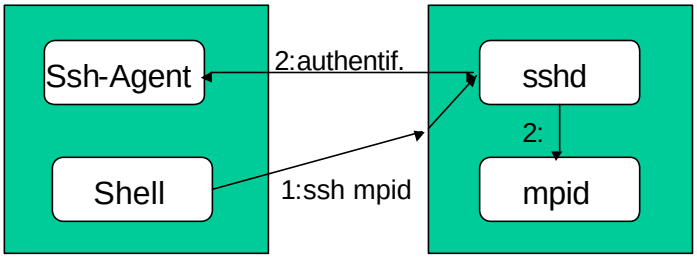
- Hochfahren des Systems durch Starten eines Daemons auf jedem Knoten
- Starten einer Applikation durch
  - Transfer der Applikations-teile zu den entsprechen-den Knoten
  - Verbinden mit dem lokalen Daemon
  - Starten der einzelnen Applikationsteile



- Bei SPMD sind alle Applikationsteile gleich.
- Problem:
  SSH verkompliziert die Nutzung entfernter Rechner

SSH ohne Passworteingabe

- Statt Passwortauthentifikation Einsatz eines Public-Key Verfahrens
- Entfernte Rechner erhalten öffentlichen Schlüssel
- Lokal wird ein Dämon gestartet,dieser erhält privaten Schlüssel
- Authentifikation nun durch Challenge-Response Verfahren



Schlüsselerzeugung

```
pc150:~)ssh-keygen
Generating public/private rsa1 key pair.
Enter file in which to save the key
  (/home/bwana/tnfink/.ssh/identity): <RETURN>
Enter passphrase (empty for no passphrase): XXXXX
Enter same passphrase again: XXXXX
Your identification has been saved in
  /home/bwana/tnfink/.ssh/identity.
Your public key has been saved in
  /home/bwana/tnfink/.ssh/identity.pub.
The key fingerprint is:
c1:2a:6d:0b:4a:2d:95:79:a7:2b:ad:e4:d8:ed:8a:77
  tnfink@pc150
```

Starten des SSH-Agenten

- Sich selbst Zugriff erlauben:
  `cp ~/.ssh/identity.pub ~/.ssh/authorized_keys`
- Start des Agenten + Starten einer neuen Shell
  `pc150:~)ssh-agent tcsh`
  Alle weiteren Aufrufe müssen in **dieser** Shell stattfinden.
- Agent liest automatisch ~/.ssh/identity, falls keine Passphrase angegeben wurde.
- Explizites Hinzufügen der eigenen Identität
  `ssh-add ~/.ssh/identity`
- Starten von Programmen auf entfernten Rechnern ohne Passwort

```
pc150:~)ssh pc151 uname –a
Linux pc151 2.4.3 #5 SMP Mon Mai 14 15:51:04 CEST 2001 i686
unknown
```

Aufsetzen des MPI Systems Kompilation

- Voraussetzungen:
  - SSH-Agent wurde gestartet in der aktuellen Shell.
  - Umgebungsvariable LAMRSH ist auf „ssh –x“ gesetzt.
- Erstellung einer Datei `hosts` mit allen Knoten:

```
pc150
pc151
pc152
```

- Hochfahren des MPI Systems:

```
pc150) lamboot hosts
LAM 6.3.2/MPI 2 C++ - University of Notre Dame
pc150)
```

Starten/Beenden einer MPI-Applikation

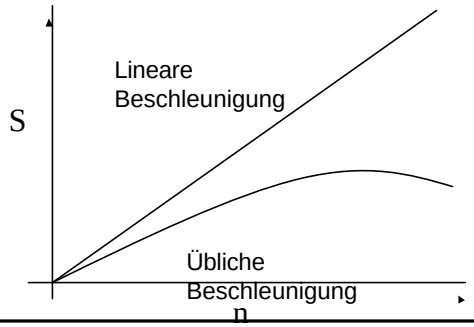
- Voraussetzung: MPI-System ist aufgesetzt.
- Start einer Applikation `a.out` im SPMD Modus mit 8 Kopien
  `mpirun –np 8 a.out`
- Terminierung, wenn alle Prozesse terminieren
- Aufräumen nach fehlerhaftem Verhalten der Applikation mit `lamclean`
- Runterfahren des MPI-Systems aufgrund Sitzungsende oder „schräger Situation“ mit
  `lamwipe –v hosts`
  Danach: Wiederaufsetzen des Systems
- Ausserdem: Statusanzeige mit `mpitask`
- Infos zu Funktionen, z.B.: `man MPI_Send`

Bewertung von parallelen Applikationen

- Beschleunigung (Speedup): $S_n = \frac{t_1}{t_n}$
- Effizienz (Efficiency): $E_n = \frac{S_n}{n}$
- Für t_1 sollte eine sequentielle Programmversion gemessen werden.
- Amdahls Gesetz:

$$t_1 = t_s + t_p \quad (t_s = a\ t_1)$$

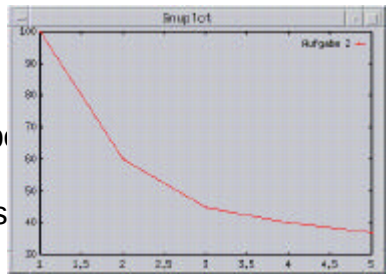
$$t_n = t_s + \frac{t_p}{n} \quad \Rightarrow \quad S_n \le \frac{t_1}{t_s} = \frac{1}{a}$$



GnuPlot in 5 Minuten

```
# n  t(ms)
1  100
2  60
3 45
4 40
5 37
```

- Zweidimensionale Messdaten in einem File `plot.dat`
- Starten von GnuPlot:




```
Hostname> gnuplot
gnuplot> plot “plot.dat“ title “Aufgab
gnuplot> set output “plot.eps“
gnuplot> set terminal postscript eps
gnuplot> replot
gnuplot> quit
Hostname>
```

Interne Hilfe mit “? <Kommando>“