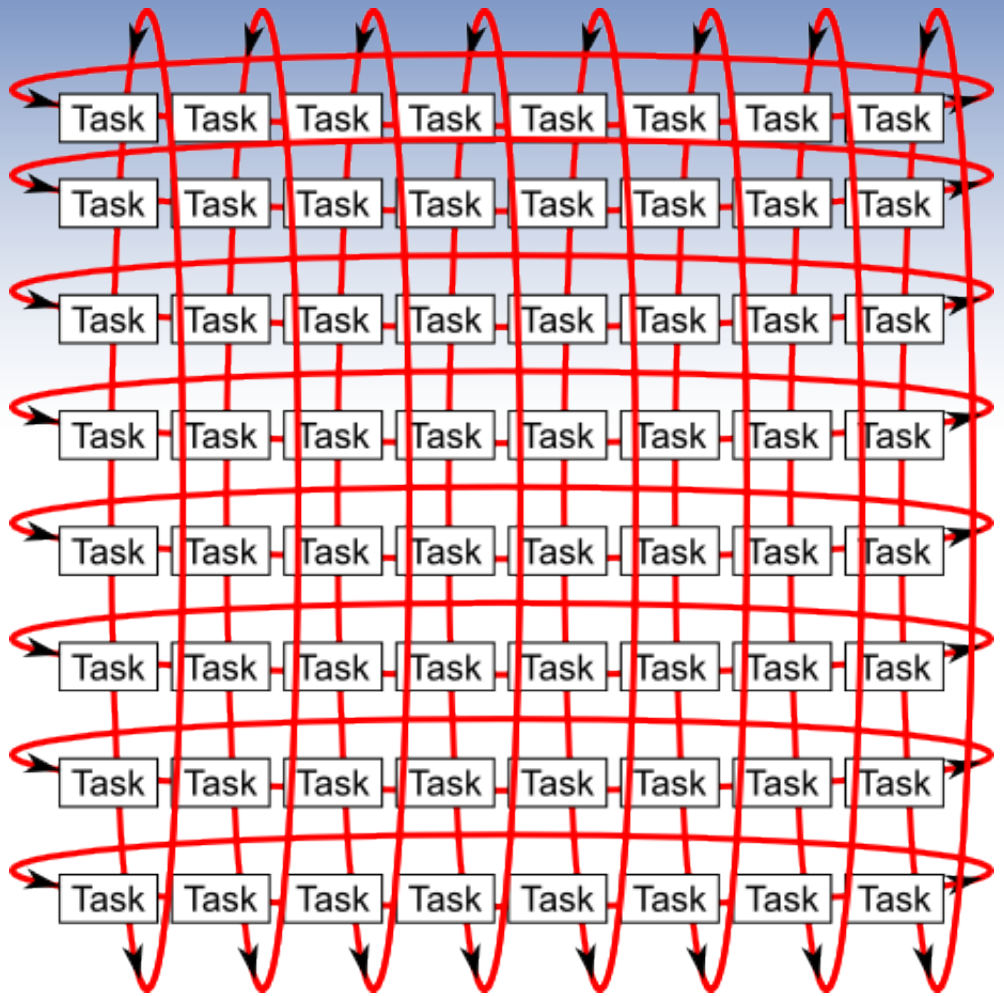
Parallelles Rechnen

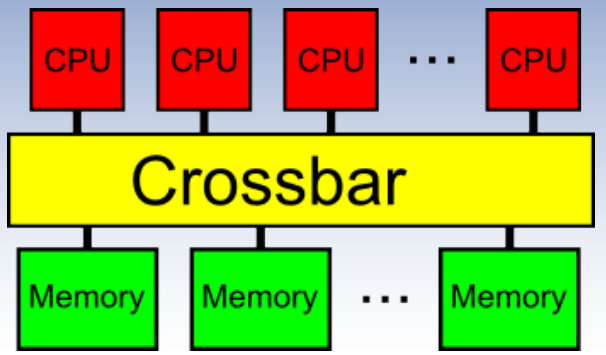


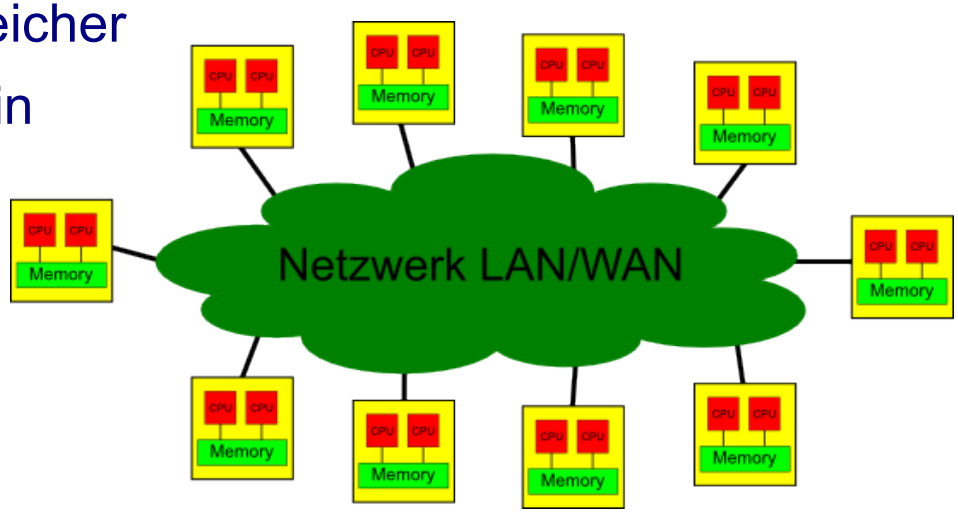
Warum Parallelprogrammierung

- große numerische Probleme (Simulation)
  - HPC – Molecular Dynamics, Monte Carlo Simulation, Gleichungslöser
- optische Bildverarbeitung
  - Qualitätssicherung bei Blechen (Parsytec)
- große WWW-Server
- Datenbanken
- Aufteilung der Speichers
- Verteilung IO

Hardware Unterschiede

- Hardware Besonderheiten
  - Shared Memory Systeme
    - Gemeinsamer Speicher
    - Kommunikation über gemeinsame Speicherbereiche
  - Distributed Memory Systeme
    - Jede Einheit hat eigenen Speicher
    - Kommunikation findet über ein geeignetes Netzwerk statt (Ethernet,InfiniBand,Myrinet)

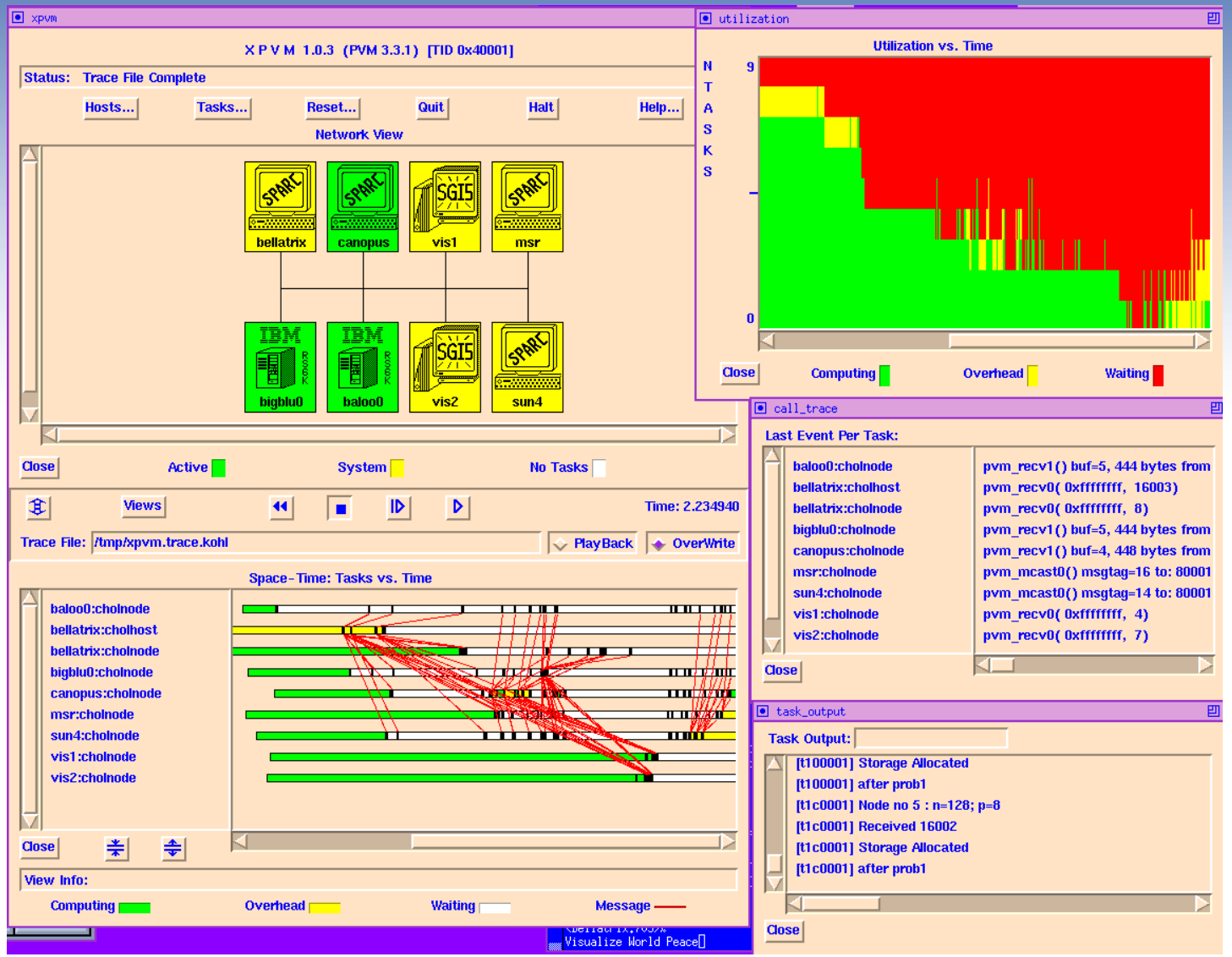




Software-Systeme

- Verschiedene Software-Systeme
  - MPI – **M**essage **P**assing **I**nterface
    - Mpich, Intel-MPI, HP-MPI
    - Lam
    - OpenMPI
  - PVM – **P**arallel **V**irtual **M**achine
    - Wird nicht mehr verwendet
  - Linda
    - Einige Chemieprogramme nutzen es
    - Tuple-Space
  - OpenMP – Compiler-Zusatz zur halbautomatischen Paralellisierung
    - HPF – High Performance Fortran (Adaptor)
  - Thread-Libraries

Graphische Tools

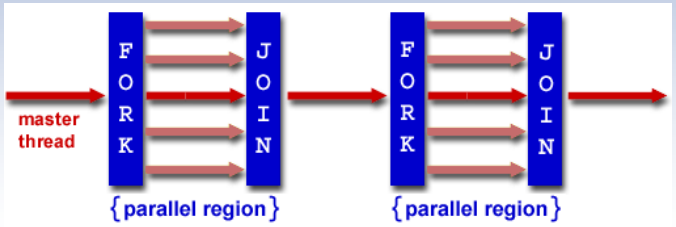


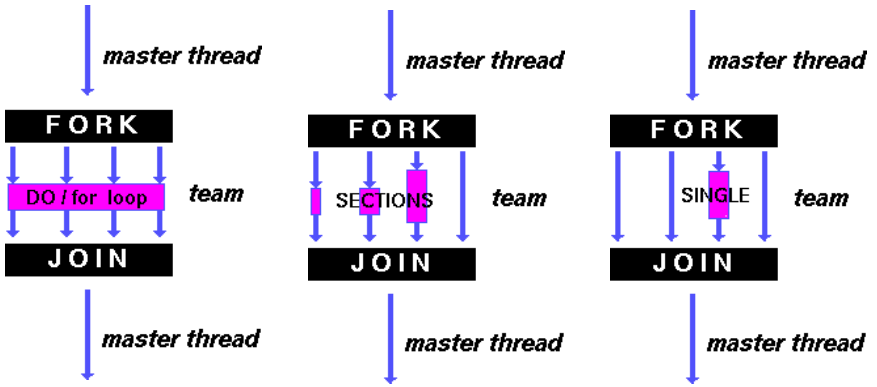
OpenMP

- Open specifications for Multi-Processing
- Weiterentwicklung in den Händen eines Architecture Review Board aus führenden IT-Firmen und Universitäten
- Spezifiziert für C/C++ und Fortran
- Unterstützung vieler Compiler: GNU Collection, Intel Compiler, PG Compiler, HP, SUN
  - Compiler versuchen halbautomatisch, bestehenden C – oder Fortran-Code zu parallelisieren
  - Hinweise des Benutzers in Form von Compiler-Directiven
  - Automatische Erzeugung von Threads

OpenMP - Directive

- Parallele Regionen werden mit durch Compiler-Directiven gekennzeichnet
  - Fortran: !$OMP
  - C/C++: #pragma omp
  - z.B. Parallel oder Do – Region
- Synchronisation
  - Master: nur Haupt-Thread
  - Critical: nur 1 Thread zur gleichen Zeit z.B. gleich Speicherbereiche
  - Barrier, Atomic, Ordered, Taskwait usw.



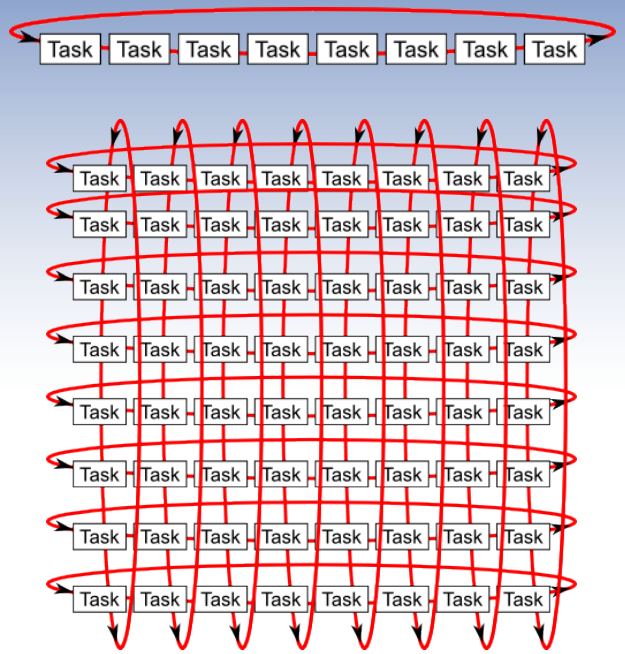


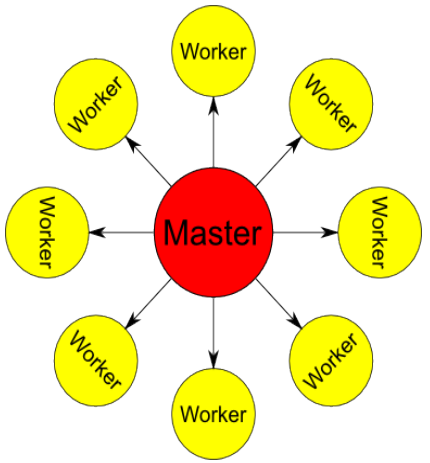
OpenMP-Funktionen und Environment-Variablen

- OMP_NUM_THREADS - Environment-Variable
  - Setzt die max. Anzahl von gleichzeitig laufenden Threads, z.B.
    setenv OMP_NUM_THREADS 8
- OMP_GET_NUM_THREADS(), OMP_SET_NUM_THREADS() - Funktion
  - Setzt bzw holt die max. Thread-Nummer
- OMP_GET_THREAD_NUM() - Funktion
  - Bestimmt die Nummer des akt. Threads im Gesamtsystem

MPI – Message Passing Interface

- Verschiedene Implementationen
  - MPICH
  - LAM/MPI
  - Open MPI
  - Intel MPI
  - HP MPI
- Unterstützung für spezielle Netzwerke
  - InfiniBand, bis 40 Gbit/s
  - MyriNet/10GBit, bis zu 20 Gbit/s
  - Herstellerspezifische Netzwerke, z.B. SGI
- Verschiedenen Netzwerk-Topologien denkbar
  - 1D-,2D- und 3D-Ringe
  - Gitterstrukturen
  - Master-Worker-Struktur





Struktur eines MPI-Programmes

- Initialisierung: MPI_Init()
- Berechnung, Kommunikation: MPI_...
- Ende und Aufräumen: MPI_Finalize()

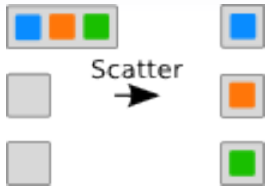
```
 call MPI_Init(ierr)
 call MPI_Comm_rank(MPI_COMM_WORLD, my_rank, ierr)
 call MPI_Comm_size(MPI_COMM_WORLD, p, ierr)
 if (my_rank  .NE.  0) then
    call MPI_Send(message, len_trim(message),
+   MPI_CHARACTER, dest, tag, MPI_COMM_WORLD, ierr)
 else
    do source = 1, p-1
       call MPI_Recv(message, 100, MPI_CHARACTER,
+      source, tag, MPI_COMM_WORLD, status, ierr)
    enddo
 endif

 call MPI_Finalize(ierr)
 end program greetings
```

Kollektive Operationen

- Broadcast – Datenverteilung

- Scatter – threadspezifische Datenverteilung



- Gather – Ergebnisse sammeln

- Reduce – Ergebnisse rechnerisch vereinen

Kennzahlen

- Amdahlsches Gesetz - max. Speedup

$$S_{max,n} = \underbrace{(1-P)}_{seq.} + \underbrace{\frac{P}{n}}_{parall.}$$

- Speedup

$$S_{P_n} = \frac{T_1}{T_n}$$

- Algorith. Speedup

$$S_{P_{alg.}} = \frac{T_{parall(1)}}{T_{parall(N)}} \cdot 100$$

- Effizienz

$$E_n = \frac{S_{P_n}}{n} \cdot 100$$