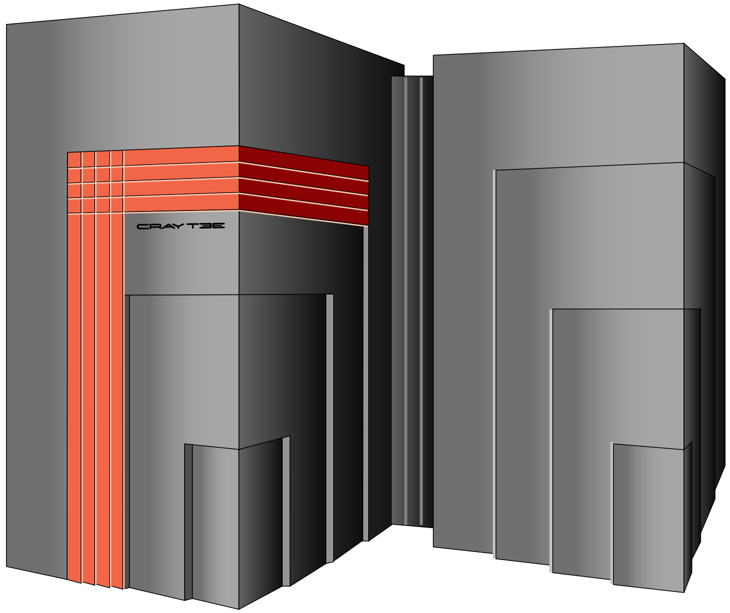
Anwender-Software
für

Cray Research
Massiv-Parallele Prozessorsysteme



CRAY RESEARCH
A Silicon Graphics Company

Sprachen und Message Passing Bibliotheken

- **Compiler**

  Fortran 90

  HPF in Zusammenarbeit mit Portland Group

  HPF_CRAFT - Shared Memory Programmiermodell Erweiterung in HPF

  C / C++

- **Ausführung der Compiler**

  CRAY T3E: Auf einem der Kommando-PEs (kein Cross-Compiling)

  CRAY T3D: Host-Rechner (Y-MP, C90 oder T90)

- **Message Passing**

  PVM (auf Hardware angepaßte Implementierung)

  MPI (auf Hardware angepaßte Implementierung für CRAY T3E)

  Shared Memory Routinen zum direkten, d.h. ungepufferten Transfer zwischen lokalem und entfernten Speicher; einfache Handhabung und sehr effizient

Fortran 90 auf CRAY T3E

- **Fortran 90 Compiler für Single-PE Code**
- **Ganzworte (REAL, INTEGER, COMPLEX, DOUBLE PRECISION) und Halbworte (REAL*4, INTEGER*4)**
- **Wichtige Optionen:**

-e A	Instrumentierung für Apprentice
-O unroll2	Loop Unrolling zur optimalen Füllung der internen Pipelines
-O split2	Schleifenaufspaltung zur optimalen Nutzung der Stream Buffer
-O aggress	'aggressive' Skalaroptimierung
-O nojump	Verzweigen zu Unterprogrammen statt Springen (bei großen Codes)
-s default32	INTEGER, REAL, LOGICAL werden auf 32 Bit Größen gesetzt; COMPLEX und DOUBLE PRECISION auf 64 Bit
-r 2	Compiler Listing in file.lst
-X m	Erzeugen eines 'mallable' (verformbaren) Executables (-X #PEs statisch für #PEs; benötigt kein mpprun)
-Wl"loader_opt"	Anweisungen an den Lader (cld)

Programmausführung auf CRAY T3E

- **Single PE Anwendungen**

  Laufen auf Kommando PEs

  Alle UNICOS/mk Kommandos

  Alle implizit oder explizit auf einem PE gestarteten Programme

- **Parallele Programme**

  Laufen auf Anwender PEs

- **Beispiel**

  ja #Starten von Job-Accounting

  SCACHE_D_STREAMS=1 ; export SCACHE_D_STREAMS #Stream Buffer aktiv

  f90 -o p.exe -O unroll2,aggress -X m a.f b.f c.f #Compilation

  mpprun -n 12 p.exe [Argumentliste] #Starten des Programms auf 12 PEs

  ja -chlt #Ausgabe und Beenden von Job-Accounting

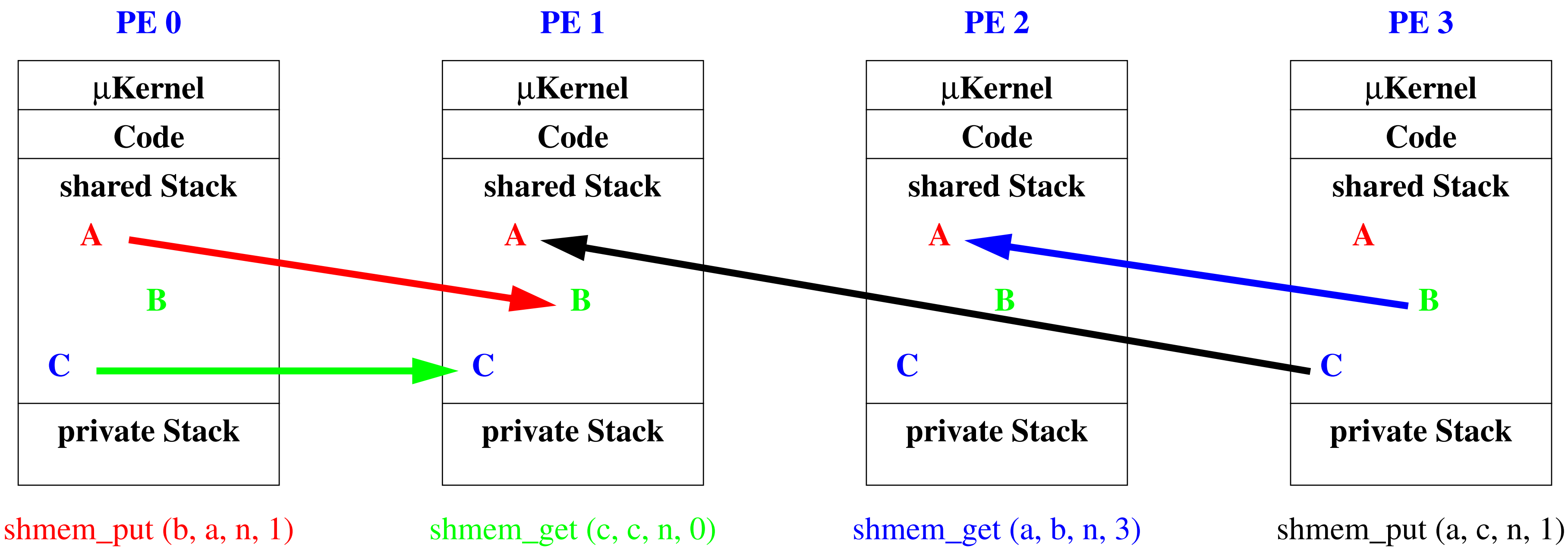
Arbeitsweise der Shared Memory Routinen

- **Ausnutzen der Shared Memory Adreß-Hardware**
- **Anwendung auf Daten im "Shared Stack" (identische Offsets)**

  Fortran COMMON Blöcke

  statische und globale Variablen in C

  dynamische Speicherplatzanforderung durch 'shmalloc'



Wesentliche Shared Memory Routinen

- **Statusinformation**

shmem_my_pe	Eigene, virtuelle PE Nummer (0 bis #PEs - 1)
shmem_n_pes	Anzahl der beteiligten PEs

- **Transferieren von Daten zwischen lokalem und entfernten Speicher**

shmem_get	Holen von n Ganzworten; Halbworte (shmem_get32 in C)
shmem_put	Schicken von n Ganzworten; Halbworte (shmem_put4 in Fortran)

- **Kollektive Routinen (alle beteiligten PEs haben den gleichen Zustand)**

shmem_broadcast	Broadcast von n Ganzworten von beliebigem PE aus
shmem_collect	Zusammenfügen von Teilblöcken zu einem Feld
shmem_sum	Bildung globaler Summen (für alle Datentypen)

- **Synchronisation**

shmem_fence	Warten auf vollständigen Abschluß einer Transferoperation
shmem_swap	Atomares Tauschen eines lokalen und entfernten Speicherinhalts
shmem_barrier	Synchronisation auf PE Untermenge (global durch barrier())

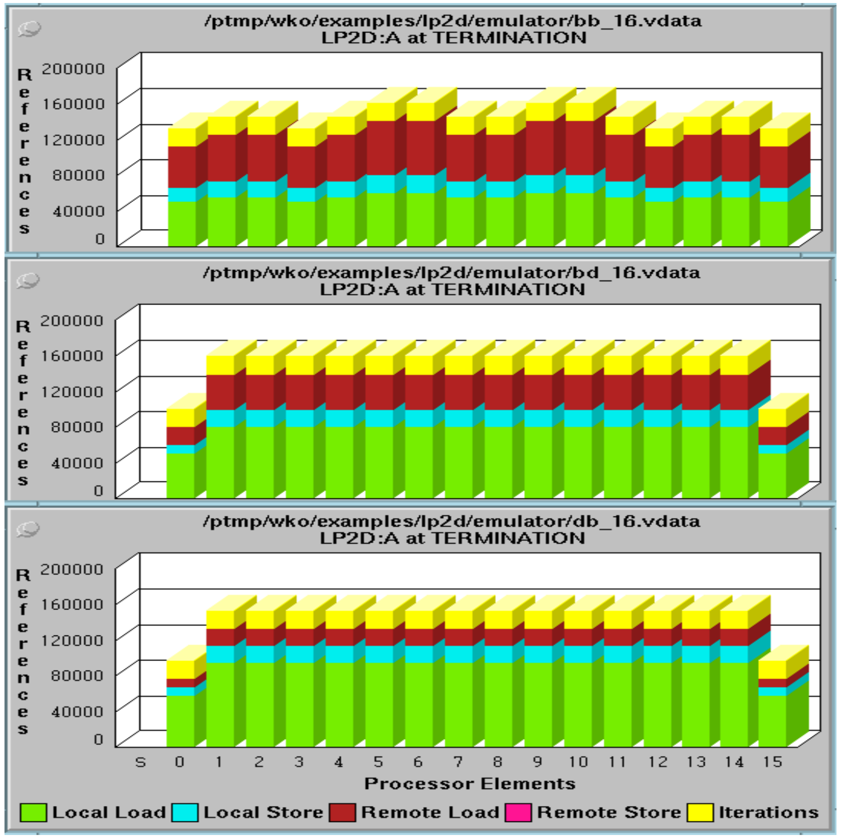
Informationen zur Laufzeit

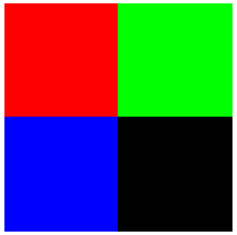
- **System Call sysconf (argument) gibt eine long Zahl zurück**
- **Argumente**

_SC_CRAY_MAXPES	maximale Anzahl für parallele Anwendungen
_SC_CRAY_NPES	Anzahl angeforderter PEs
_SC_CRAY_VPE	virtuelle PE Nummer (0 bis _SC_CRAY_NPES - 1)
_SC_CRAY_LPE	logische PE Nummer
_SC_CRAY_PPE	physikalische PE Nummer; ist als Hexadezimal mit je zwei Stellen für x, y und z Koordinate verschlüsselt (ZZYYXX)
_SC_CRAY_USRMEM	verfügbarer Speicherplatz für Anwender (in Bytes)
_SC_CRAY_SYSMEM	Speicherplatz für System (in Bytes)
_SC_CRAY_CPCYCLE	Taktzeit des Prozessors (in Pico-Sekunden)

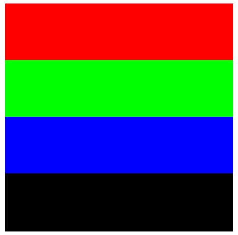
Auswirkung der Datenverteilung in CRAFT auf CRAY T3D

```
cdir$ do shared (j,i) on a(i,j)
       do j = 2,n-2,2
        do i = 2,n-1
         a(i,j) = 0.25*(a(i+1,j) + a(i-1,j) + a(i,j+1) + a(i,j-1))
        enddo
       enddo
```

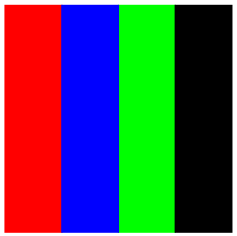




`cdir$ shared a(:block,:block)`



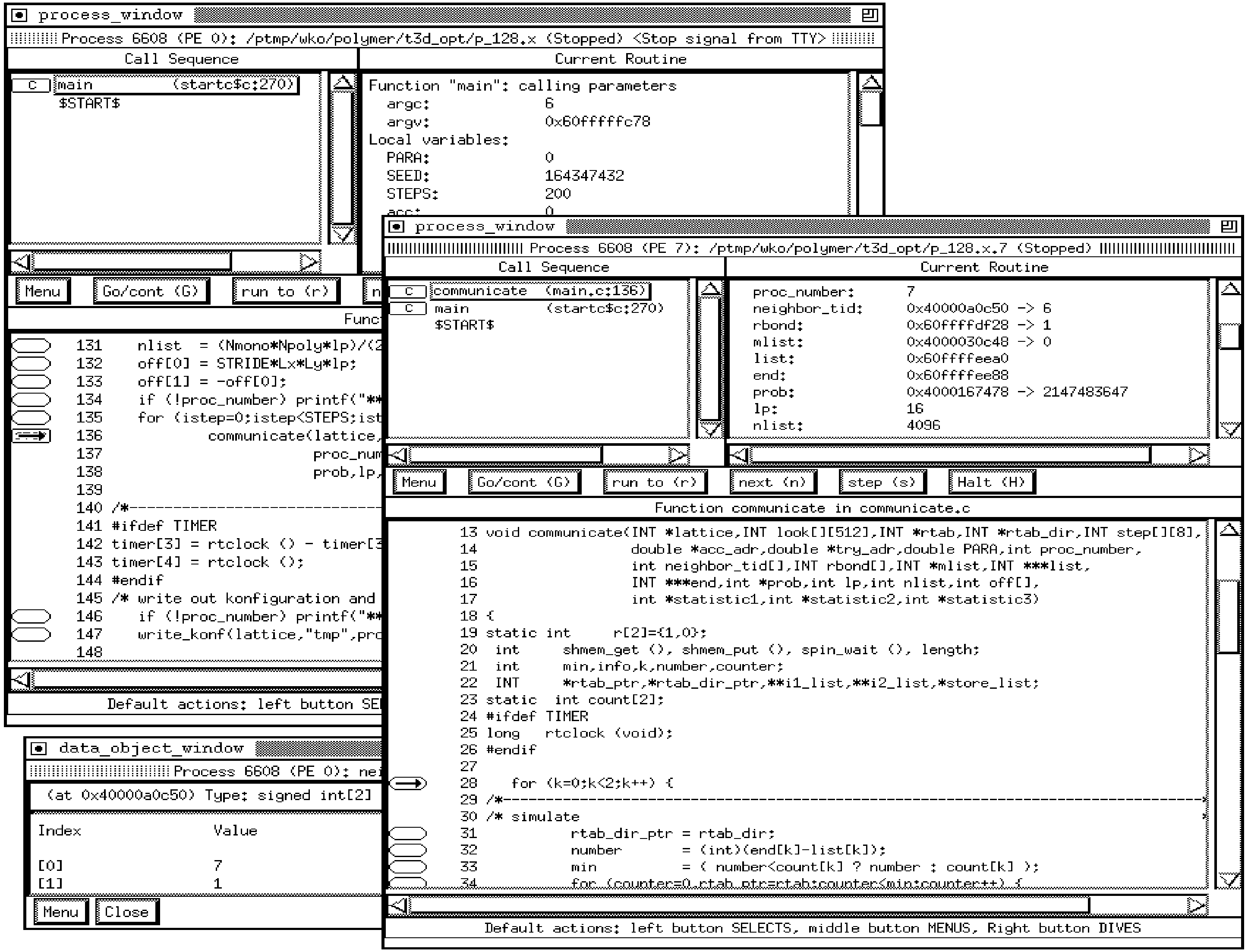
`cdir$ shared a(:block,:)`



`cdir$ shared a(:,:block)`

Totalview - Debugger

- **Interaktiv und Post-Mortem**
- **Bezug auf Quellcode**
- **Globale und lokale Breakpoints**
- **Process Windows für jedes PE**
- **Object Windows für Inhalte von Variablen**



A Silicon Graphics Company

Performance-Analyse mit Apprentice

- **Instrumen-tierung durch Compiler Option**
- **Bezug auf Quellcode**
- **Messung auf Ebene von Code Blöcken**
- **Zusammen-führung der Messung aller PEs**