Werkzeuge zur parallelen Programmierung: Leistungsvorhersage, -analyse und -debugging

Roland Wismüller

LRR-TUM
Institut für Informatik
Technische Universität München
e-mail: wismuell@informatik.tu-muenchen.de

Inhaltsübersicht

Leistungsanalyse sequentieller Programme

Motivation:

- Amdahl's Gesetz:
  Speedup $S \leq 1/f$, wobei f nicht-parallelisierter (Laufzeit-)Anteil des sequentiellen Programms ist

$\Rightarrow$ Zuerst die Teile mit der höchsten Laufzeit parallelisieren

Dazu nützlich: *Profiling*-Werkzeuge

- Beispiele: `gprof`, `quantify`, `cvperf` (ProDev WorkShop)
- Gemessene Daten:
  - Laufzeit/Zeitanteil einzelner Prozeduren
  - Dynamischer Aufrufgraph mit Aufrufhäufigkeit
  - Teilweise: Zeitanteil für einzelne Codezeilen
- Darstellungsarten:
  - Textuelle Zusammenfassungen
  - Annotationen des Quellcodes
  - Aufrufgraphen mit Annotationen

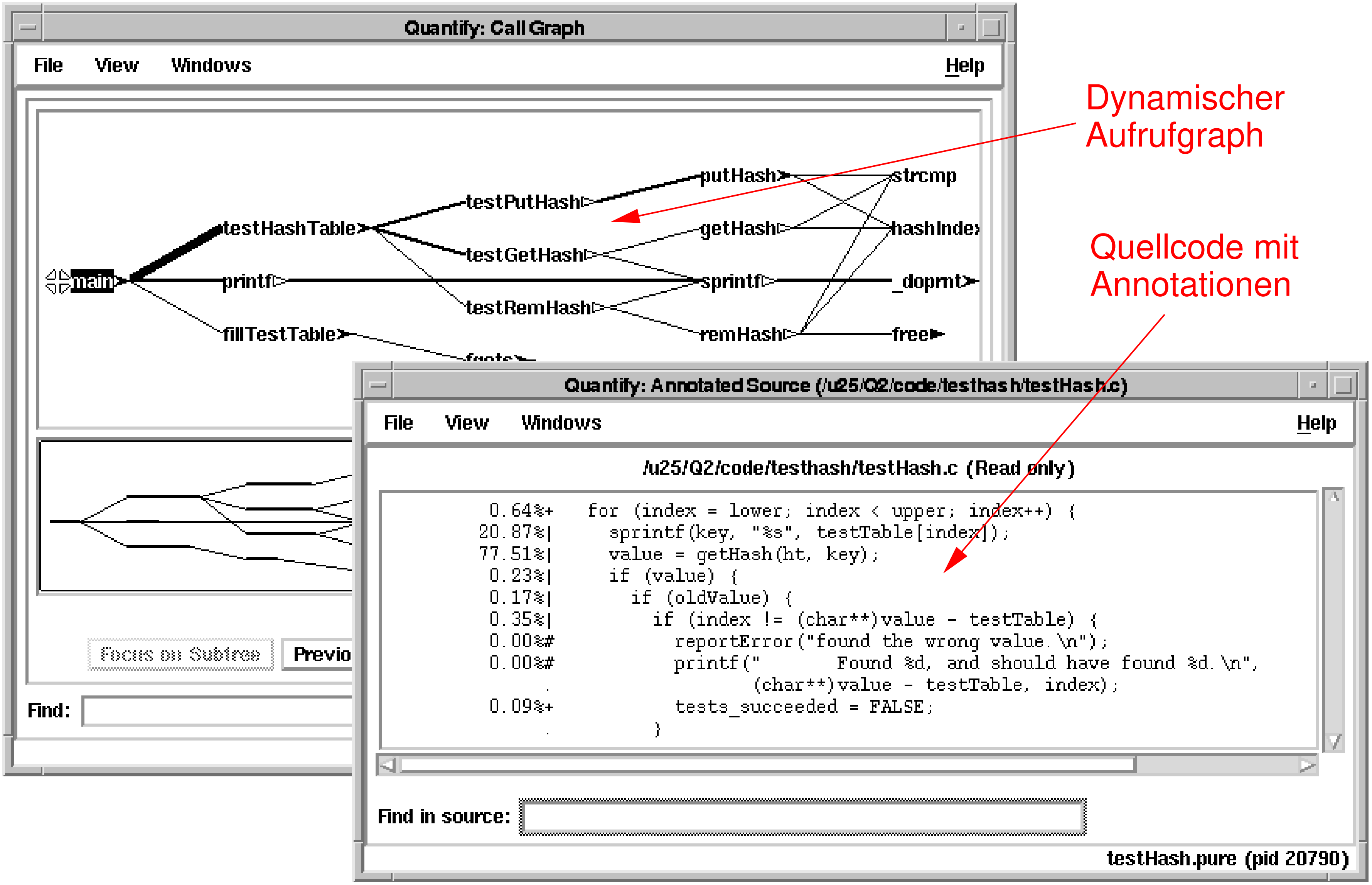
Beispiel: gprof-Ausgabe

```
                                called/total       parents
index  %time    self descendents  called+self    name          index
                                called/total       children

                0.09        0.48    9504/9504          st_get_next_sym [6]
[7]      27.4   0.09        0.48    9504           st_transform_dbx_sym [7]
                0.02        0.27    2687/2687          _st_transform_stabs [9]
                0.03        0.07    9521/11127         _st_strdup [21]
                0.00        0.05    5686/13750         _st_new_sym_entry [20]
                0.00        0.03    9504/16412         _st_free [31]
```

```
    %  cumulative    self              self     total
 time    seconds   seconds    calls  ms/call  ms/call name
 26.7       0.56      0.56                            internal_mcount [8]
  9.5       0.76      0.20       40     5.00     5.00  _read [13]
  9.5       0.96      0.20                            _sbrk_unlocked [14]
  8.6       1.14      0.18     5669     0.03     0.03  st_sort_line_in_func [16]
  4.8       1.24      0.10       19     5.26     5.26  _write [23]
  4.3       1.33      0.09     9504     0.01     0.06  st_transform_dbx_sym [7]
```

Beispiel: `quantify`



Leistungsvorhersage

Einfache Abschätzungen:

- Vor Festlegung der Parallelisierungsstrategie:
  Amdahl's Gesetz
- Für gegebene Parallelisierungsmethode:
  Bestimmung des Kommunikations- / Berechnungsverhältnisses
- Bei vorhandener Implementierung:
  - Bei einigen parallelisierende Compilern:
    symbolischer Ausdruck für Laufzeit von z.B. Schleifen
  - Forschungsthema: Automatische Leistungsvorhersage für unterschiedliche Zielsysteme
    Kombination aus statischer Analyse, sequentiellem Profiling und Mikrobenchmarks auf der Zielmaschine

Beispiel zum Kommunikations- / Berechnungsverhältnis

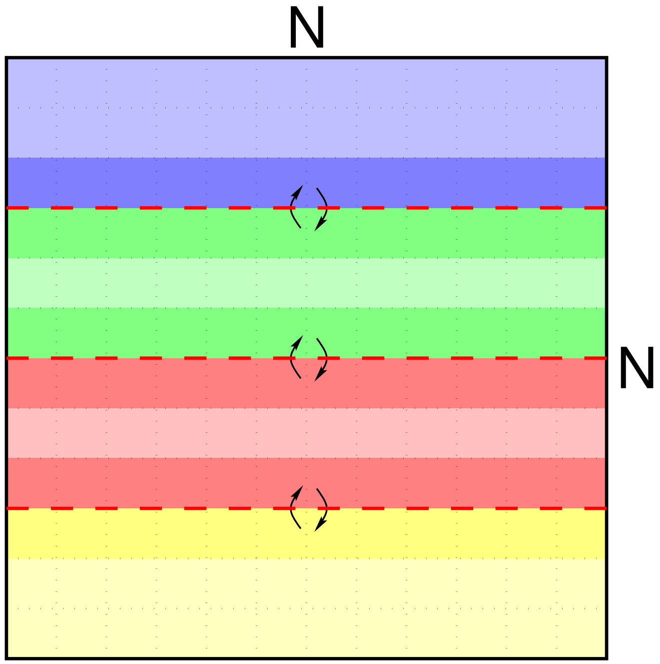
Anwendung: Gauß-Seidel-Relaxation auf $N \times N$ Matrix

- Berechnungsaufwand:
  $5 * (N-2)^2$ Gleitpunktoperationen pro Iteration
  Auf HP 9000/720: $B(N) = (N-2)^2 * 0.68 \mu s$
- Kommunikationsaufwand:
  - Bei streifenweiser Aufteilung: $2 * (p-1)$ Nachrichten mit N Gleitpunktzahlen pro Iteration
    Auf HP 9000/720 mit PVM: $K(N,p) = (p-1) * (2400 + N * 8) \mu s$
  - Bei blockweiser Aufteilung: $2 * \sqrt{p} * (\sqrt{p} - 1)$ Nachrichten mit $N/\sqrt{p}$ Gleitpunktzahlen pro Iteration
    Auf HP 9000/720 mit PVM: $K(N,p) = (\sqrt{p} - 1) * (\sqrt{p} * 2400 + N * 8) \mu s$
- Speedup:

$$S = \frac{t_s}{t_p} = \frac{p}{1 + K(N,p)/B(N)}$$

Matrix-Aufteilung bei Gauß-Seidel-Relaxation

Streifenweise Aufteilung

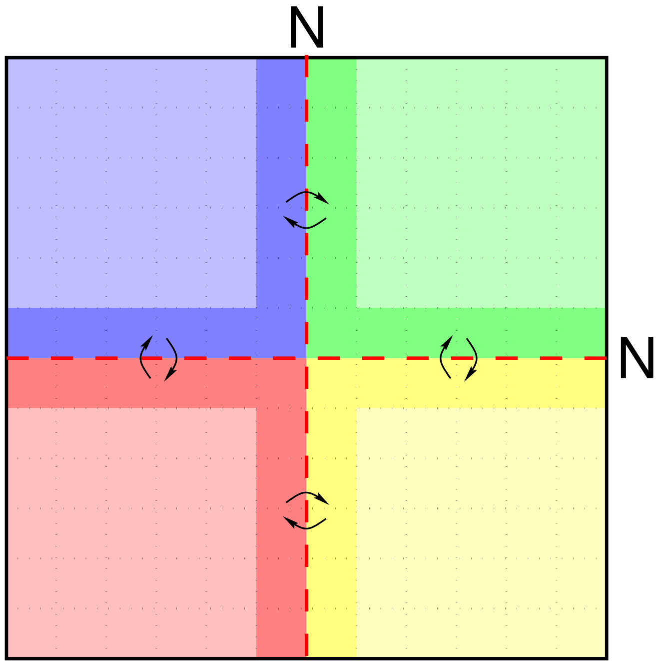


Kommunikation:

$(p-1) * 2$ Nachrichten

N Elemente

Blockweise Aufteilung

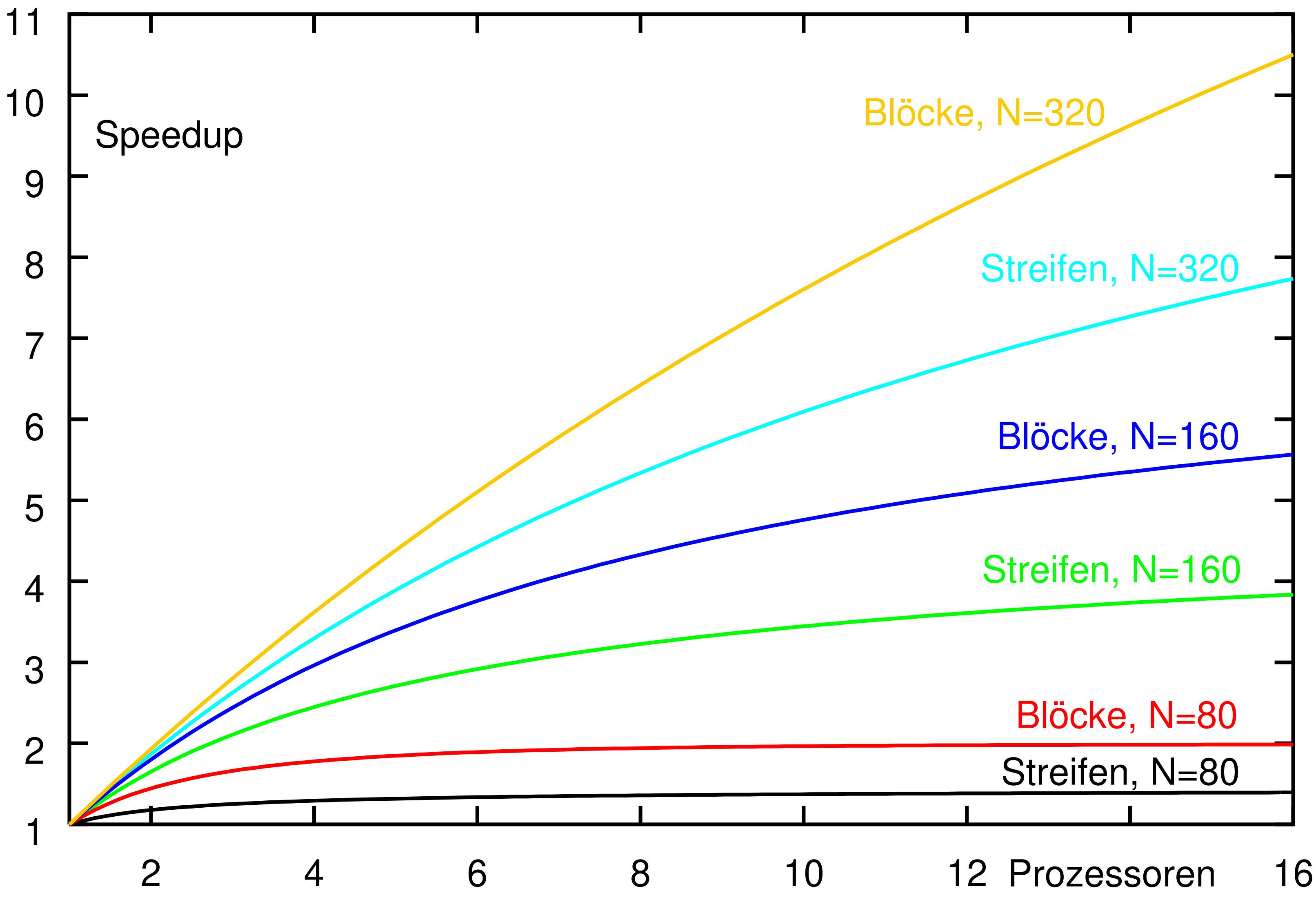


Kommunikation:

$\sqrt{p} * (\sqrt{p} - 1) * 2$ Nachrichten

$N / \sqrt{p}$ Elemente

Ergebnis der Analyse:



Leistungsdebugging

Leistungsdebugging: Aufspüren und Beseitigen der Ursachen mangelnder Leistung

Unterschiedliche Zielvorgaben:

- Laufzeitoptimierung des Anwendungsprogramms
  (auf exklusiven Ressourcen)
- Optimierung der Verweilzeit eines Anwendungsprogramms in einem belasteten System
- Optimierung von Auslastung bzw. Durchsatz der Ressourcen
  (CPU, Speicher, Netzwerk, Hintergrundspeicher)
- Gesichertes Einhalten von Echtzeitbedingungen

Ursachen für Leistungsmängel:

- **Zugriffsverluste:** bei der Überwindung von Distanzen beim Datenaustausch zwischen Systemkomponenten (z.B. Nachrichtenaustausch)
- **Konfliktverluste:** bei der gemeinsamen Benutzung von Systemkomponenten durch mehrere Programmeinheiten (z.B. Konflikte beim Netzzugriff)

Leistungsdebugging (2)

Ursachen für Leistungsmängel:

- **Auslastungsverluste:** entstehen durch einen zu geringen Parallelitätsgrad des zu bearbeitenden Programms (z.B. permanente oder zeitweise Lastungleichheit)
- **Bremsverluste:** entstehen beim Beenden gewisser Berechnungen, wenn eine Lösung gefunden wurde (z.B. bei Suchproblemen: Benachrichtigung aller anderen Prozesse)
- **Komplexitätsverluste:** entstehen durch den Zusatzaufwand, der bei der parallelen Abarbeitung einer Anwendung im Vergleich zur sequentiellen Abarbeitung entsteht (z.B. Berechnung der Matrizenaufteilung bei Lösern für Gleichungssysteme)
- **Wegwerfverluste:** ergeben sich durch Berechnungen, die im parallelen Programm mehrfach ausgeführt werden und nicht weiterverwendet werden (z.B. manchmal Doppeltberechnung von Ergebnissen in mehreren Prozessen)
- **Algorithmische Verluste:** können durch implizite Änderung des Algorithmus bei der Parallelisierung entstehen (z.B. schlechtere Konvergenz bei Iterationsverfahren)

Werkzeugunterstützung für die Leistungsanalyse

Funktionalität der Werkzeuge:

- Messung leistungsrelevanter Größen
- Analyse dieser Größen (Aufgliederung)

Meßgrößen:

- Zeit (*wall clock*, CPU-Zeit, Blockiertzeit)
- Anzahl des Auftretens verschiedener Ereignisse
  (z.B. Gleitpunktoperationen, Cache-Fehler, Kommunikationen, Unterbrechungen, ...)
- Auslastung von Systemressourcen
  (z.B. CPU, Speicher, Kommunikationsnetz)

Werkzeugunterstützung für die Leistungsanalyse (2)

Aufgliederung der Meßwerte:

- System-Bezug:
  - Gesamtsystem, einzelne Knoten, Knotenmengen
  - Alle Prozesse, einzelne Prozesse, Kommunikationspartner
  - Operationen des Programmiermodells
- Quellcode-Bezug:
  - Gesamtprogramm, einzelne Funktionen
  - Verteilung auf einzelne Schleifen oder Codezeilen
  - Benutzerdefinierte Bereiche
- Zeit-Bezug:
  - gesamte Laufzeit, zeitlicher Verlauf
  - Schnappschüsse (genauer: kleines Intervall)

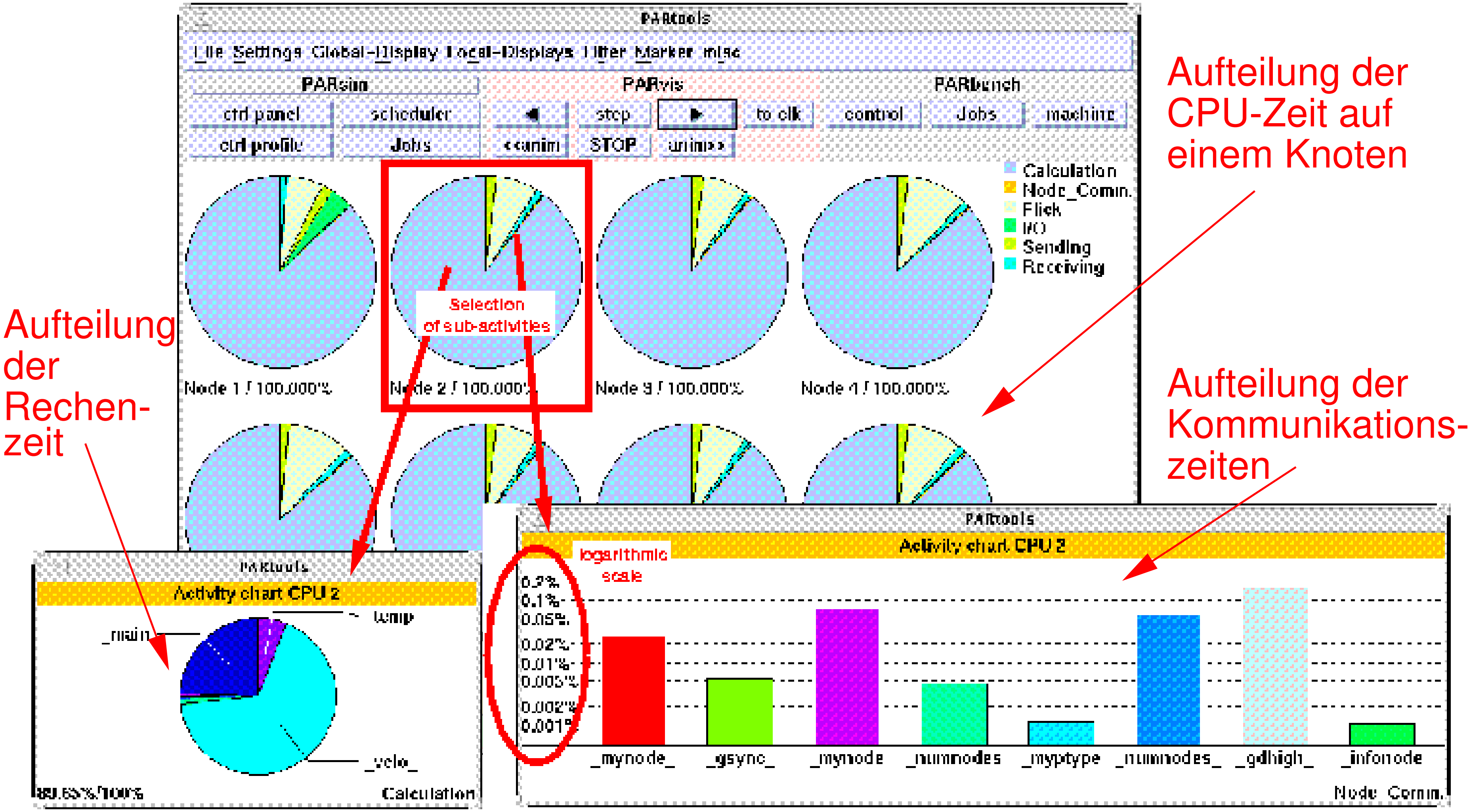
Bei System- und Quellcode-Bezug: i.a. auch Einschränkung der Messungen möglich

Darstellungsformen für Leistungs-Meßwerte

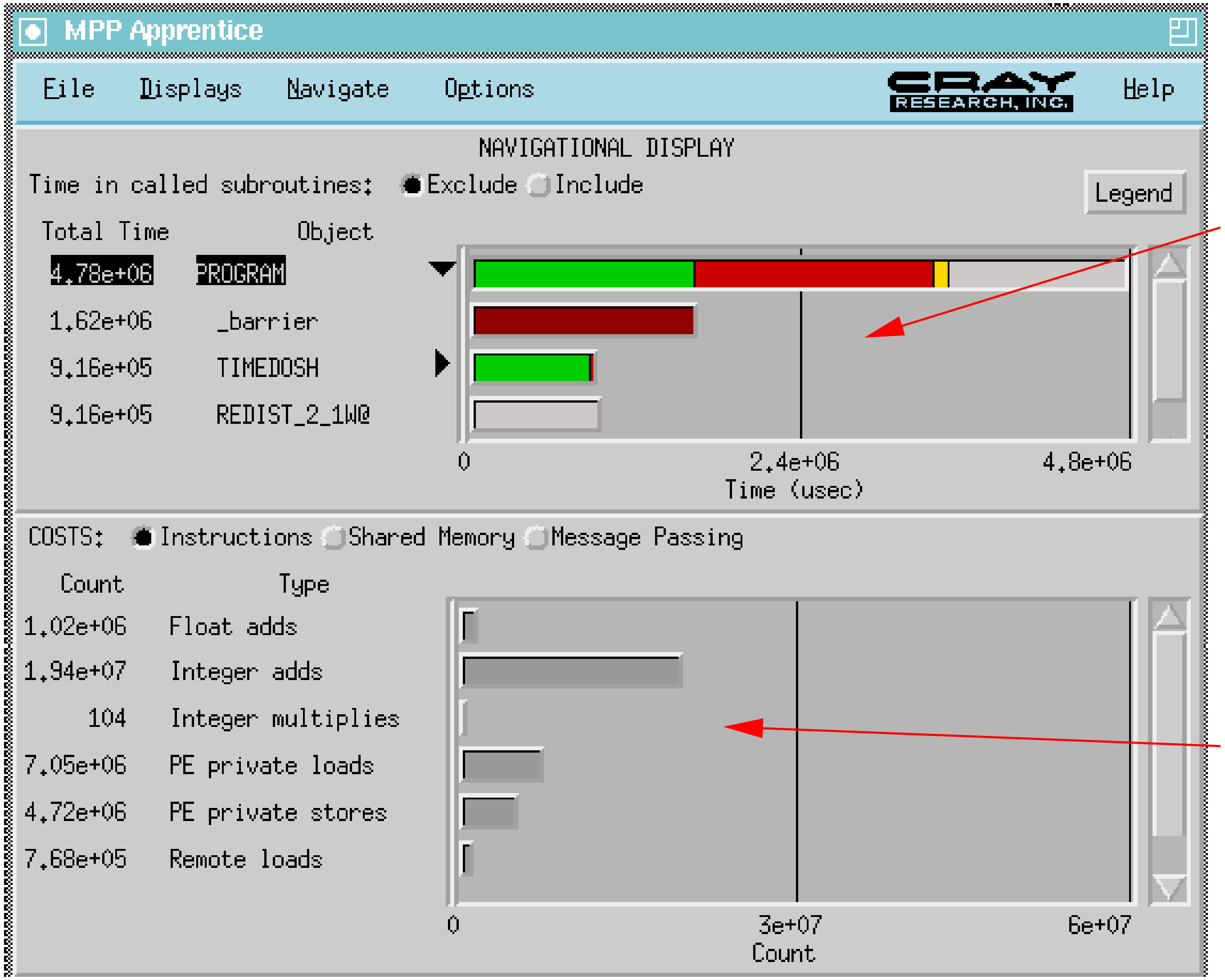
Sehr große Vielfalt:

- Textuelle Ausgaben (meist Zusammenfassungen)
- Balkendiagramme (einzelne Meßgrößen, Verteilung auf Bezugsobjekte)
- Tortendiagramme (Verteilung auf Bezugsobjekte)
- Kiviat-Diagramme (Meßgrößen und Verteilung)
- Topologische Darstellungen (Auslastung der Betriebsmittel)
- Kurven über der Zeit
- Gantt- bzw. Raum-Zeit-Diagramme
- ...

Beispiel: Aufgliederung der Meßwerte (parvis)



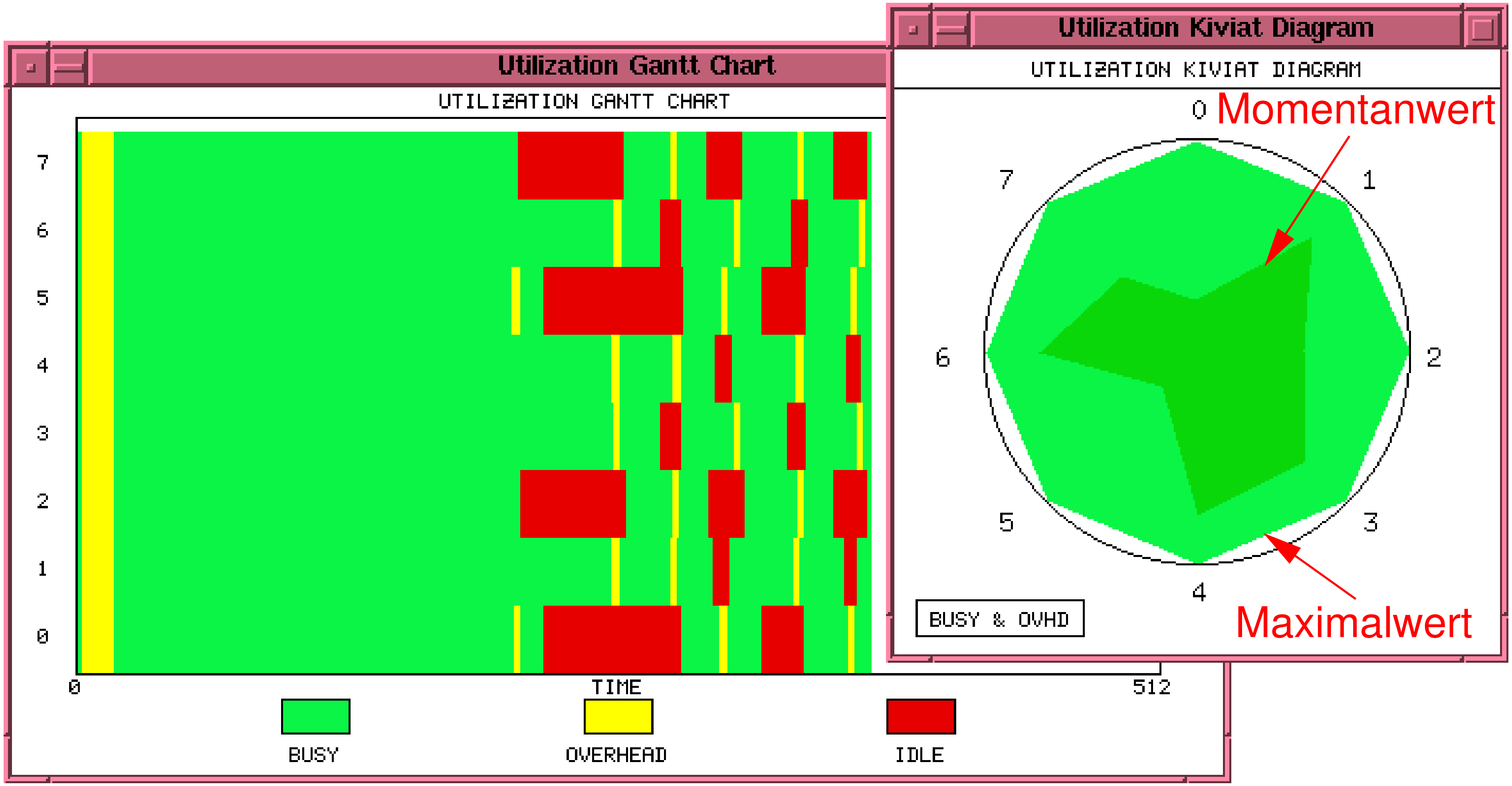
Beispiel: Balkendiagramme (`apprentice`)



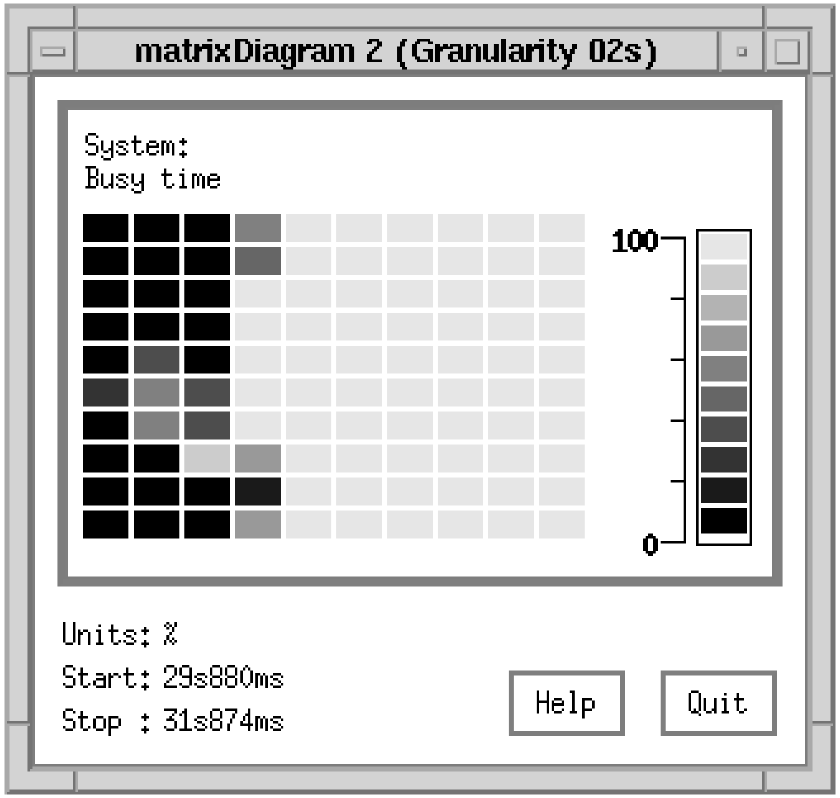
Verteilung von Rechen- und Blockiertzeiten auf einzelne Funktionen

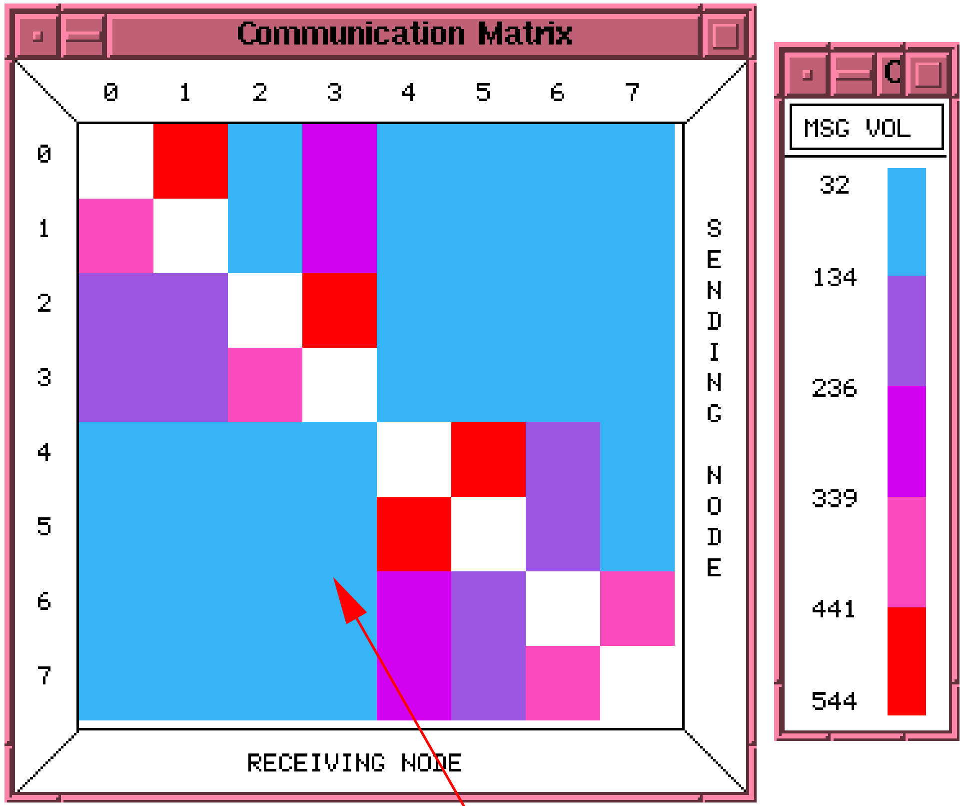
Werte verschiedener Ereigniszähler

Beispiel: Gantt- und Kiviatdiagramm (`paragraph`)



Beispiel: Matrixdiagramme (`patop`, `paragraph`)

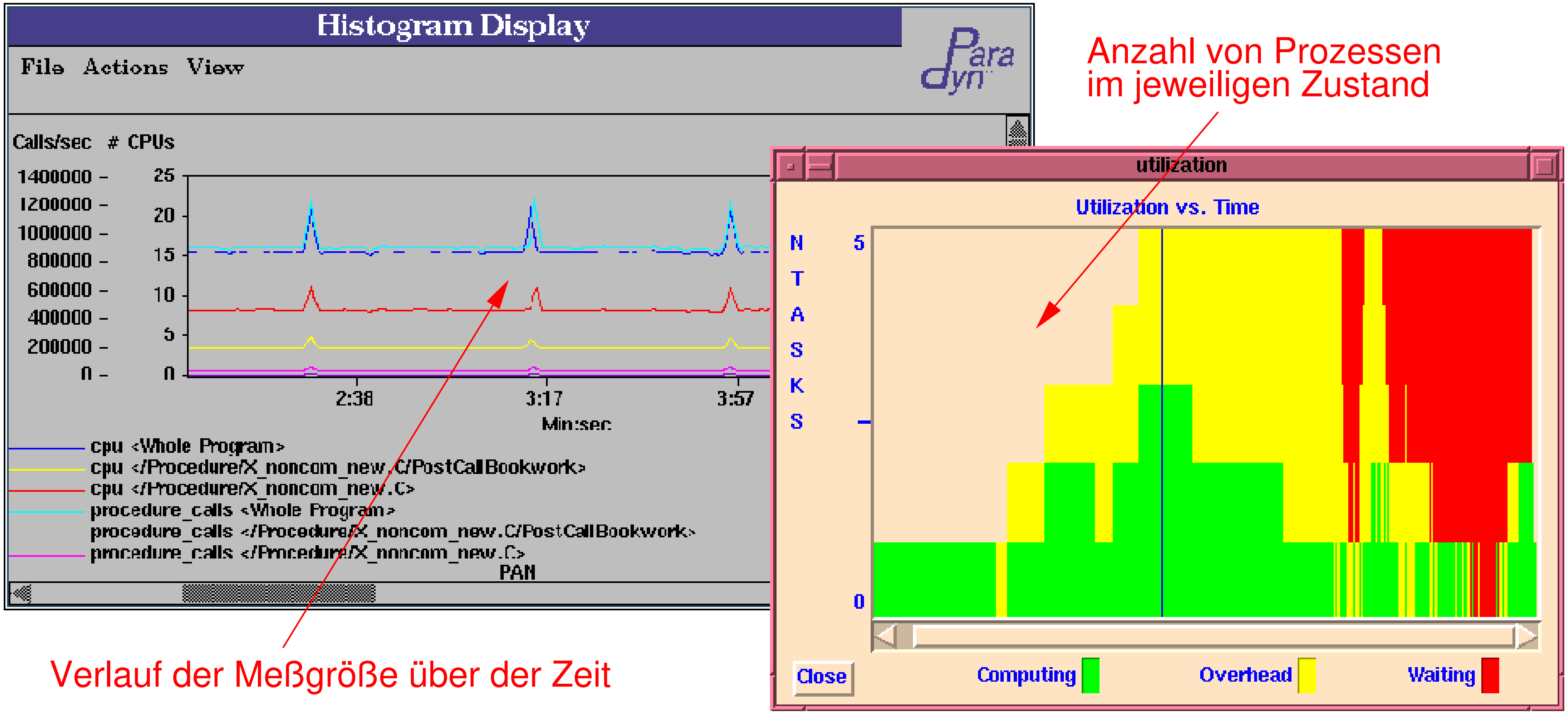




Kommunikationsvolumen zwischen Knoten

Beispiel: Zeitorientierte Diagramme (`paradyn`, `xpvm`)

Anzahl von Prozessen
im jeweiligen Zustand



Verlauf der Meßgröße über der Zeit

Kategorien von Leistungsanalysewerkzeugen

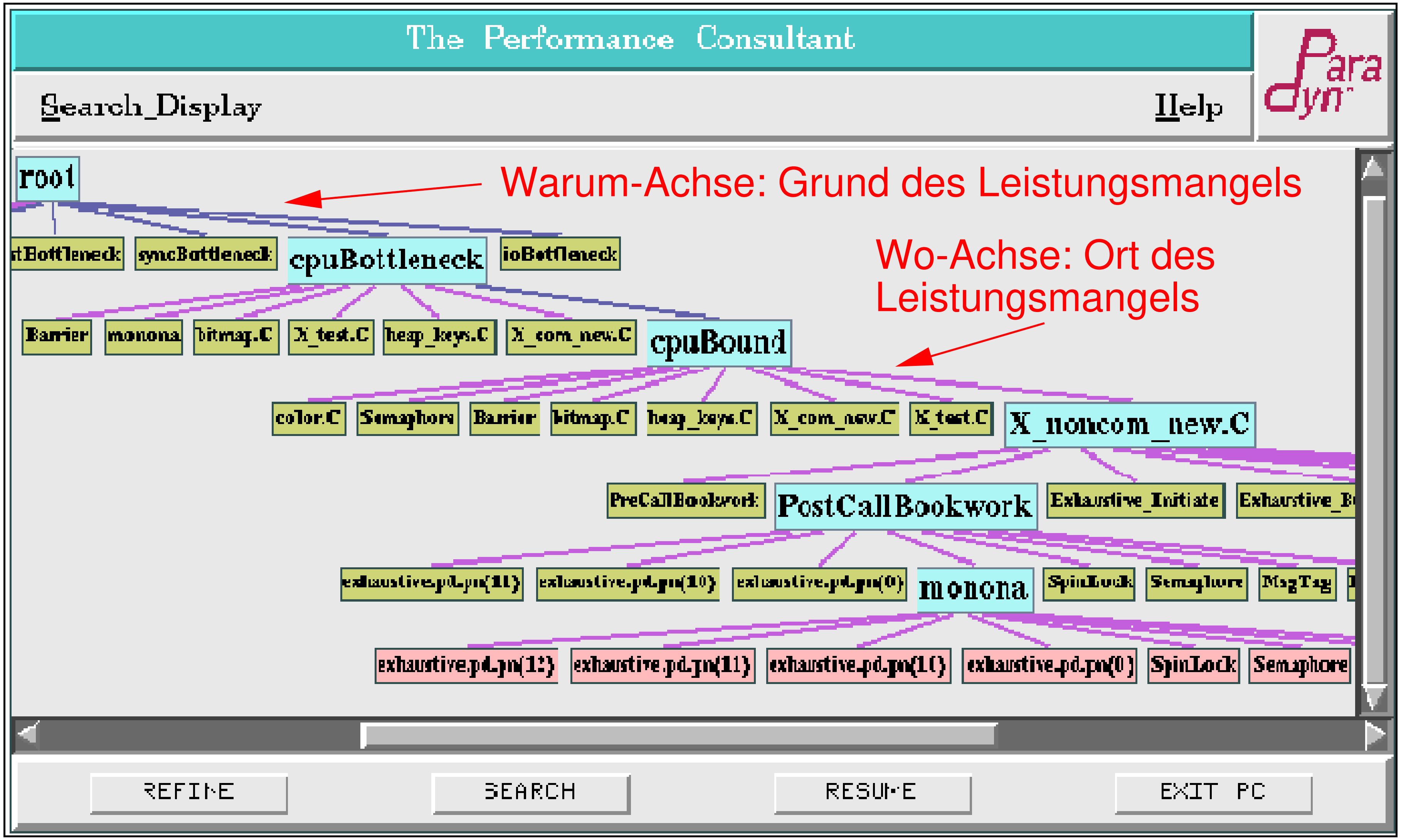
Interaktive, manuelle Leistungsanalyse:

- Off-line Werkzeuge
  - Spur-basiert (i.a. kombiniert mit Visualisierung)
  - Profil-basiert (kein Zeitbezug)
  - Problem: starke Beeinflussung bei feingranularen Messungen
- On-line Werkzeuge
  - Definition (Einschränkung) der Messungen zur Laufzeit möglich
  - Gut geeignet bei zyklischen Programmen: Neue Messungen aufgrund der bisherigen Ergebnisse möglich $\Rightarrow$ Automatisierung der Engpaßsuche möglich

Beispiel für (halb-)automatisches Werkzeug: `paradyn`

- Basis: W^3-Suchmodell: drei Achsen für mögliche Verfeinerung einer Messung (Wo?, Wann?, Warum?)
- Verfeinerungen entlang einer Achse automatisch:
  gleichzeitig, falls Beeinflussung des Programms unter Schwellwert, sonst nacheinander

Beispiel: `paradyn`



Erkennung von Leistungsmängeln

Zugriffsverluste: Messung von Kommunikationszeiten, schließt auch Konfliktverluste ein. Unterscheidung durch zeitliche Analyse.

Auslastungsverluste: Messung von CPU-Auslastung und Blockiertzeiten

Andere: nur mit Hilfe von Zusatzwissen möglich, z.B.:

- **Bremsverluste:** Blockierungen am Ende der Berechnung
- **Komplexitätsverluste:** Laufzeiten der entsprechenden Funktionen
- **Algorithmische Verluste:** Zählen der Iterationen