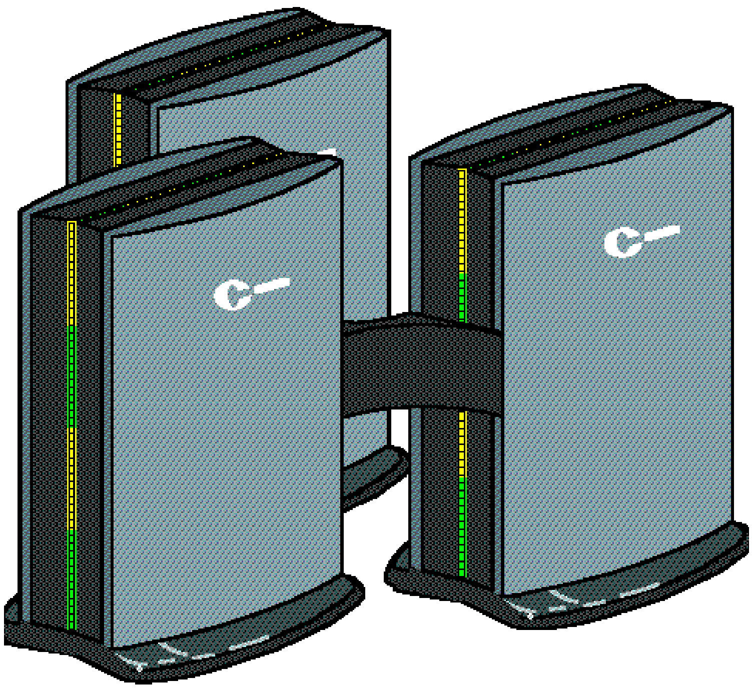
Die Convex SPP-1600

Michael Schröder

IMMD IV, Lehrstuhl für Betriebssysteme
Universität Erlangen-Nürnberg

1 Convex SPP am RRZE

- 6 Hypernodes mit jeweils 8 Prozessoren (= 48 Proz.)
- 6 Gigabyte Hauptspeicher
- 46 Gigabyte Plattenspeicher
- FDDI, DAT Tape
- Teststation



2 Architektur

- Prozessor: HPPA 7200 / 120 MHz.
- Prozessor-Cache: 1 MB Daten, 1MB Instruktionen.
- Verbindungshardware:
  - 8 Prozessoren über non-blocking Crossbarswitch (Hypernode)
  - max. 16 Hypernodes über 4 parallele Hochgeschwindigkeitsringe (CTI)
  - → Non-Uniform Memory Access (NUMA)
  - Network-Cache zum Beschleunigen der Internode Zugriffe.

3 cont.

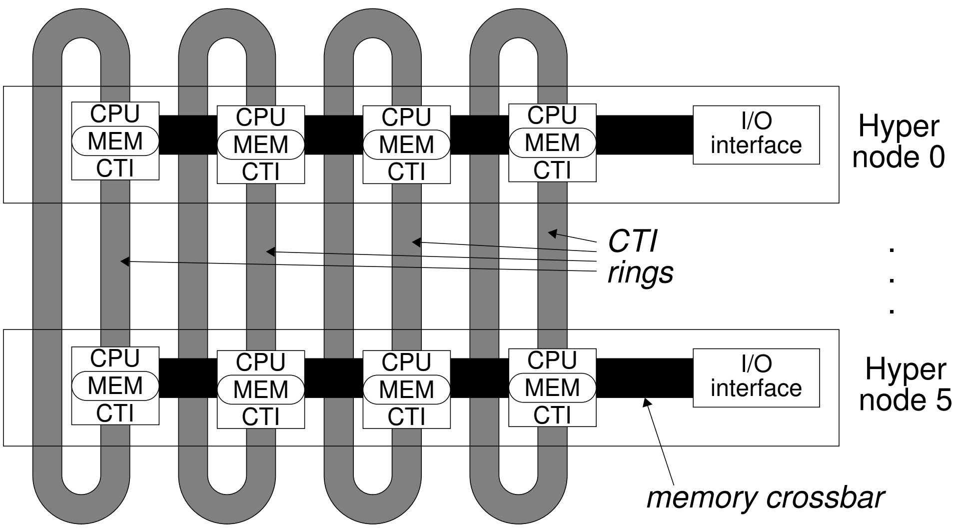
- I/O-System:
  - 250 Mbyte/s I/O Port an jedem Crossbar (also auf jedem Hypernode).
  - Pro IO Board 4 x IEEE-1496 Sbus Adapter.
  - Erweiterbar um weitere 4 Sbus/PCI Adapter.

4 cont.

- Betriebssystem SPP-UX 4.2.
  - ▲ MACH 3.0 Microkernel, HP-UX Server.
  - ▲ Auf jedem Node läuft ein Server
- HP-UX Application Binary Interface (ABI)
  - ▲ HP-UX Programme sind im allgemeinen ausführbar.
- Affinity Scheduler
  - ▲ Es wird versucht, Prozesse an den Prozessoren zu halten, an denen sie das letztens gelaufen sind.
  - ▲ Wichtig für die Cacheausnutzung.
  - ▲ Dies gilt insbesonders für NUMA Architekturen.

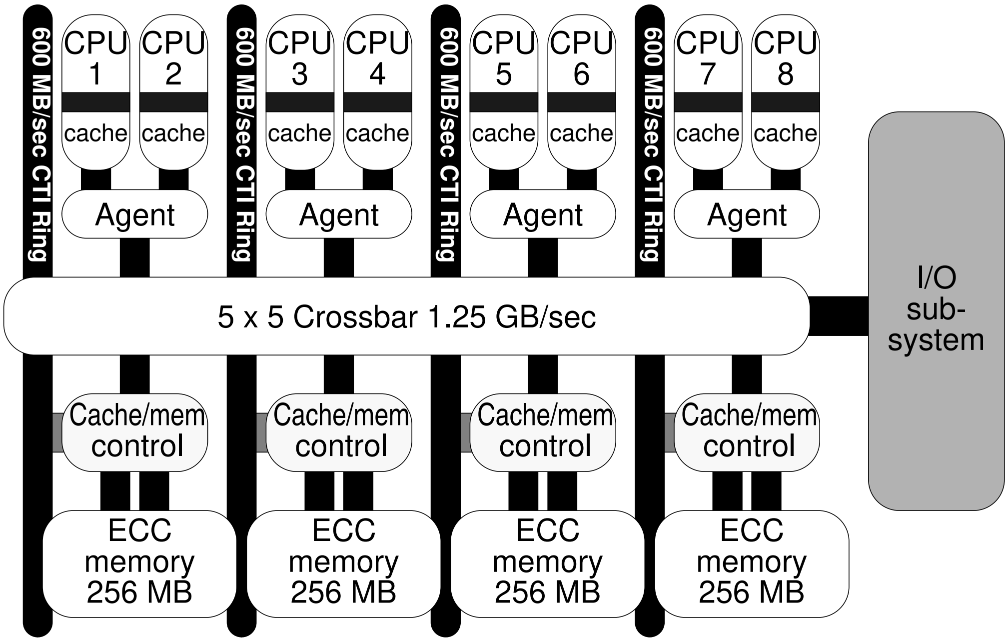
5 cont.

- Verbindungsnetzwerk:



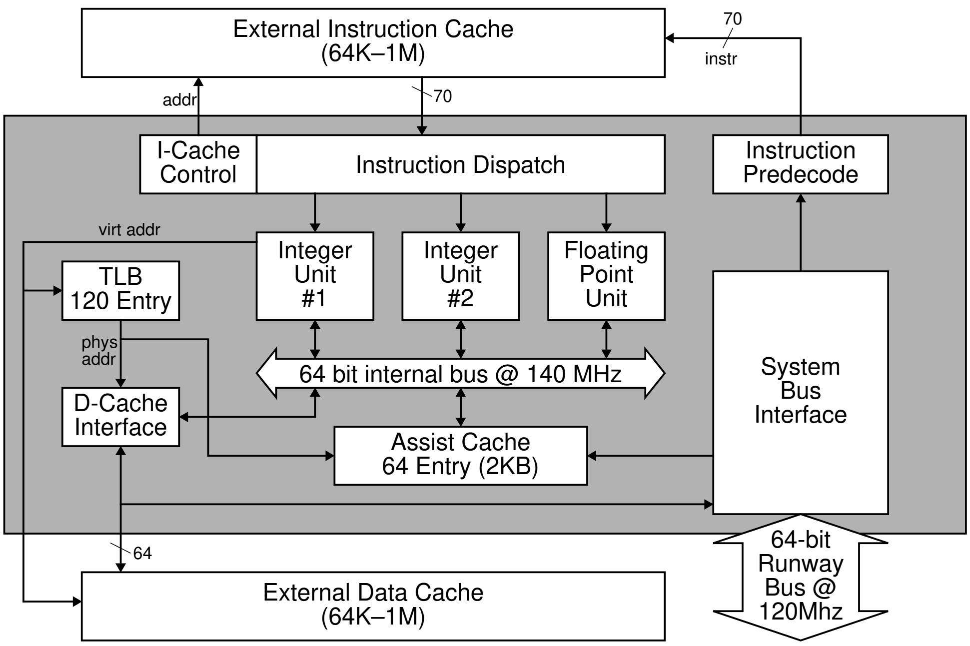
6 cont

■ Aufbau eines Nodes



7 Prozessorarchitektur HPPA 7200

■ Blockschaltbild



8 cont.

- CPU:
  - ▲ 120 MHz; CMOS 0.55 micron
  - ▲ 32 x 32 Bit-General-Register
  - ▲ 5 Stage Pipelines
- Datenformate
  - ▲ Signed & Unsigned 16-bit Integers
  - ▲ Signed & Unsigned 32-bit Integers
  - ▲ Single Word (32-bit) IEEE Floating Point
  - ▲ Double Word (64-bit) IEEE Floating Point
  - ▲ Quad Word (128-bit) IEEE Floating Point

9 Speicherarchitektur

- Prozessor-Cache:
  - ▲ 1 Mbyte Datencache / 1 Mbyte Instruktionscache.
  - ▲ 32 Byte Cachelines.
  - ▲ Direct mapped / virtually indexed / write back.
  - ▲ Abbildung Speicher->Cache:

```
cache_addr = virtual_addr & (1<<20 - 1)
```

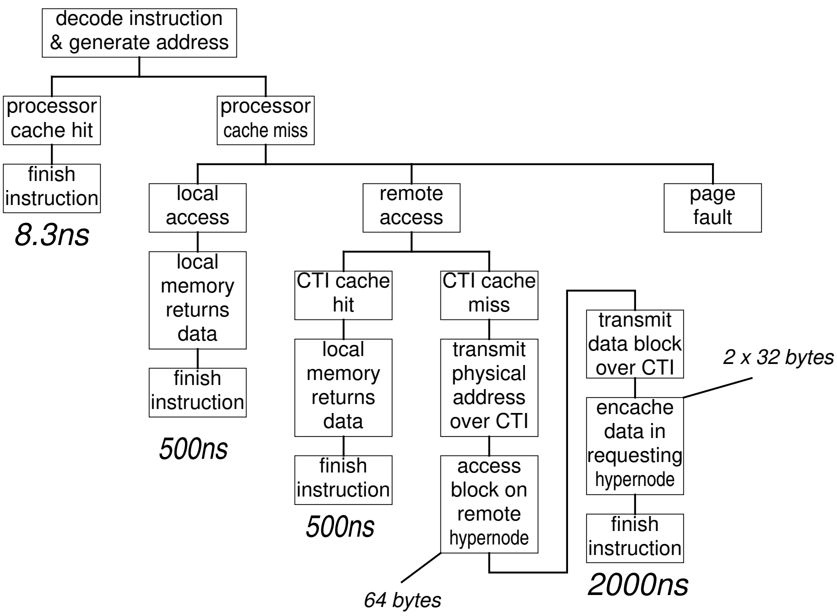
- Network-Cache:
  - ▲ Konfigurierbare Größe.
  - ▲ 64 Byte Cachelines.
  - ▲ physically indexed.

10 cont.

- Physikalische Speicherklassen:
  - ▲ Prozessor Cache
    - – 2 x 1MByte / 1 cycle
  - ▲ Node private
    - – abh. vom Speicherausbau / 50 cycles
  - ▲ Network cache
    - – konfigurierbar / 50 cycles
  - ▲ Subcomplex global
    - – abh. vom Speicherausbau / 50 cycles bzw. 200 cycles

11 cont.

- Ablaufplan der Speicherzugriffe:



12 Convex SPP - Systemzugang

- Interaktiver Zugang
  - ▲ gsm.rrze (131.188.3.30).
  - ▲ Erreichbar über `rlogin` und `telnet`.
- Konfiguration (Subcomplexe)
  - ▲ System (2 Prozessoren, nur für Systemdienste)
  - ▲ **default** (6 Prozessoren, interaktive Arbeit / Tests)
  - ▲ gsm8 (8 Prozessoren, 1 GByte Speicher)
  - ▲ gsm32 (32 Prozessoren, 2 + 2 GByte Speicher)

13 cont.

- Batch Zugang:
  - ▲ Über NQS+ (`qsub`, `qstat`, `qjob`, `qdel` -> man pages)
  - ▲ Eingangsqueues: `gsm8@gsm` bzw. `gsm32@gsm`
  - ▲ Automatische Steuerung der Prioritäten durch Berücksichtigung der in letzter Zeit verbrauchten Rechenzeit.
  - ▲ **Nicht für Übungsaufgaben!**

14 Besonderheiten

- Spezielle Programme / Bibliotheken:
  - Die von Convex mitgelieferten Erweiterungen finden sich in:
    `/usr/convex/bin` (in `$PATH` aufnehmen)
    `/usr/convex/man` (in `$MANPATH` aufnehmen)
  - `syspic` zum Anzeigen der aktuellen Nodeauslastungen.
  - `fau_top` – wie top, gibt aber zusätzlich den Subcomplex aus.
- Subcomplexe
  - Nach Login automatisch im *default* Subcomplex.
  - Starten von Prozessen in anderen Complexen durch:
    `mpa -sc gsm32 ./testprg`

15 cont.

- Unterstützung von Kernelthreads
  - `cnx_ps -T` zeigt einzelne Threads eines Prozesses an (versteht alle `ps` Optionen)
  - `pot` – `top` auf Basis von Threads (–> Man Pages)
  - Zeitmessung über Kommandozeile:
    - timex oder /bin/time verwenden, time (Shell builtin) eine falsche Realzeit anzeigt.
    - Die User- und Systemzeit aller Threads eines Prozesses wird automatisch aufsummiert.
    - `timex /bin/echo`

```
real        0.09
user        0.00
sys         0.04
```

16 C Compiler

- HP C Compiler
  - ▲ `/bin/cc`
  - ▲ keine automatische Parallelisierung
  - ▲ kein ANSI C
- Convex C Compiler
  - ▲ `/usr/convex/bin/cc`
  - ▲ Automatische Parallelisierung ab `-O3`
  - ▲ Bessere Codeerzeugung
- Linker
  - ▲ Parallele Programme mit `/usr/convex/bin/ld` linken!

17 cont.

- Dokumentation:
  - ▲ Manpage
  - ▲ Convex C Guide (`/usr/doc/ps/CUG.ps`)
  - ▲ Exemplar Programming Guide (`/usr/doc/ps/epg.ps`)
  - ▲ PinPoint (`/usr/bin/pinpoint`)
- Kompatibel mit HP-C
  - ▲ HP und SPP Objektmodule können zusammengelinkt werden.
  - ▲ Ausnahme: HP Objektmodule, die Graphikinstruktionen enthalten (werden für Networkcache Implementierung verwendet)

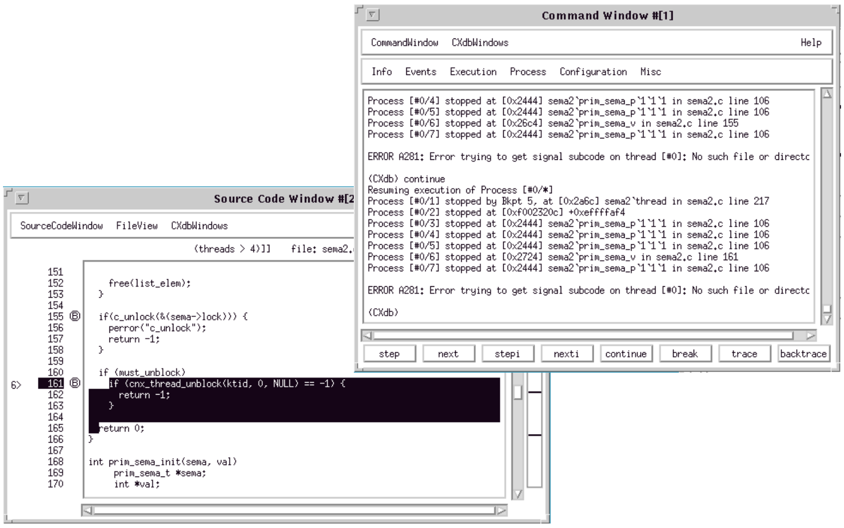
18 cont.

- Features:
  - ▲ ANSI-C kompatibel
  - ▲ Parallelisierend
  - ▲ Spezielle Datentypen
    - – 64 Bit Integer: long long
    - – SPP Speicherklassen (near_shared, thread_private...)
    - – Synchronisierung (critical_section...)
- Debugging und Profiling
  - ▲ cxdb (compilieren mit `-cxdb`)
  - ▲ cxpa (compilieren mit `-cxpa`)

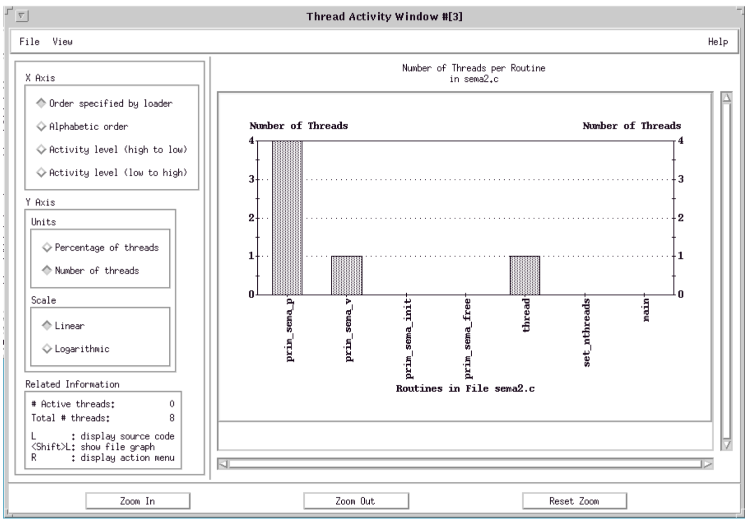
19 Cxdb - Debugger

- Sourcelevel Debugger:
  - ▲ C und Fortran
  - ▲ Multithread Support
  - ▲ Debuggen einzelner Threads
  - ▲ Debuggen von optimiertem Code
- X Window Interface
  - ▲ Command-, Source-, Threadwindow
- Aufruf:
  - ▲ Compilieren mit `-cxdb`
  - ▲ `/usr/convex/bin/cxdb program [args]`

20 cont.



21 cont.



22 Cxpa – Profiler

- Features:
  - ▲ C und Fortran
  - ▲ Multithread Support
  - ▲ Profiling von optimiertem Code
- X Window Interface
  - ▲ Command-, Source, 2 und 3D Display, Profiling Reports
- Aufruf:
  - ▲ Compilieren mit `-cxpa`
  - ▲ `/usr/convex/bin/cxpa program args`

23 cont.