NWZSUPERDOME

Einführung für Benutzer

Christian Mück-Lichtenfeld

Organisch-Chemisches Institut

May 16, 2007

Überblick

Themen

- Hardwareresourcen
- Installierte Software
- Programmierung
- Queueing System

Zielgruppe: High Performance Computing (HPC) User

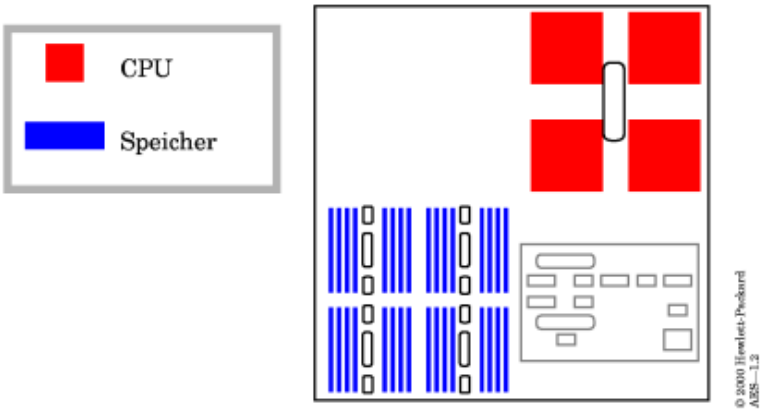
HP Superdome

Itanium2 Prozessor

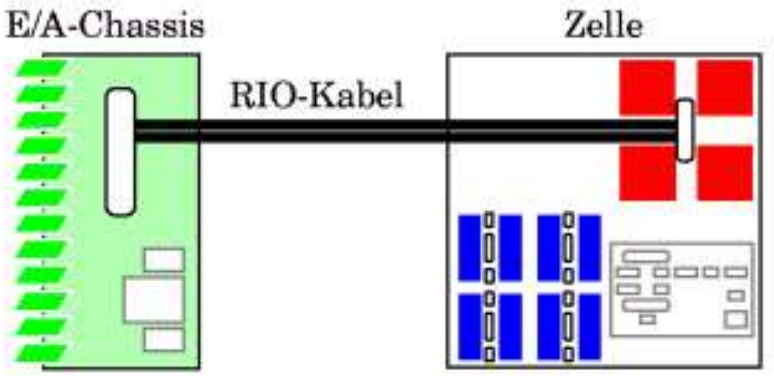
- Montecito
- 1.6 GHz
- 18 MB L3 Cache
- 2 Cores (Dual Core) pro CPU

Segmentierung der Superdome

- Superdome besteht aus Cellboards (CBs)
- jedes CB ist mit bis zu 4 CPUs und 16 GB RAM bestückt



Verbindungen zwischen Cellboards und IO

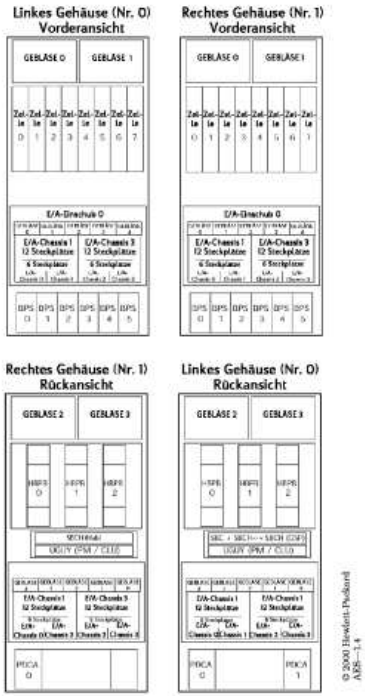


Rackaufbau

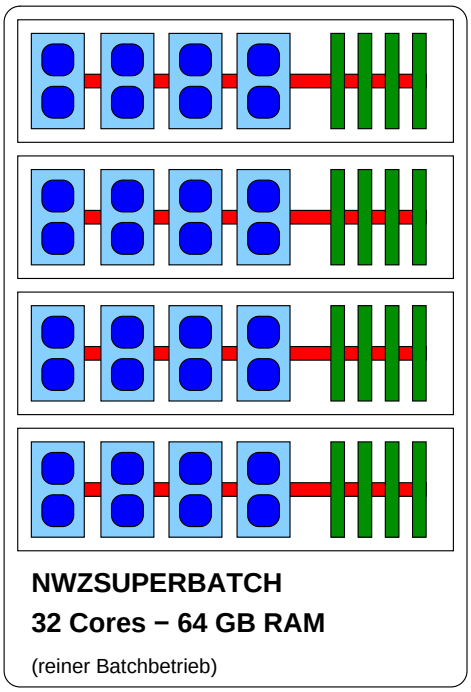
Superdome 16-Way:
Gehäuse #0, Zellen 0–3

Superdome 32-Way:
Gehäuse #0, Zellen 0–7

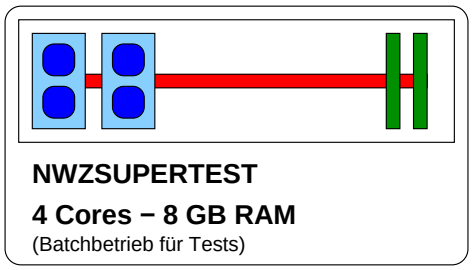
Superdome 64-Way:
Gehäuse #0–1, alle



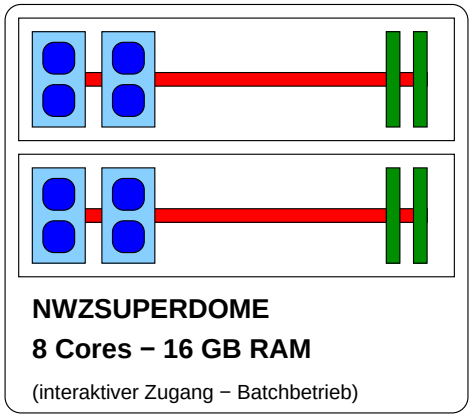
Partitionierung der Superdome



NWZSUPERBATCH
32 Cores – 64 GB RAM
(reiner Batchbetrieb)

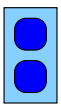


NWZSUPERTEST
4 Cores – 8 GB RAM
(Batchbetrieb für Tests)



NWZSUPERDOME
8 Cores – 16 GB RAM
(interaktiver Zugang – Batchbetrieb)

4 GB RAM

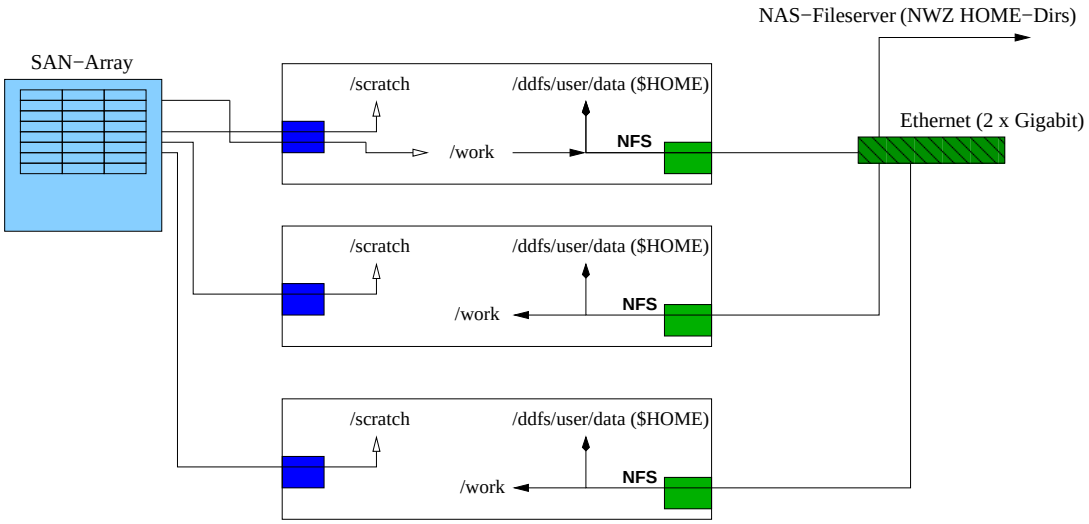


Itanium2 CPU
(dual core)

Betriebssystem

- Red Hat Enterprise Linux AS release 4
  (Nahant Update 4)
- Anbindung an die Benutzerverwaltung des NWZnet
- Anbindung der Homeverzeichnisse des NWZnet
  (für Windows-User = Laufwerk **I:**)
- Zugang per ssh (von Windows, Linux, MAC aus)

Dateisysteme der Superdome-Partitionen



- $HOME und /work (200GB) auf allen drei Knoten identisch
- /scratch pro Knoten separat (50GB - wird turnusmässig gesäubert)

Shell-Umgebung

- BASH empfohlen, TCSH, PDKSH und ASH vorhanden.
- Vorgaben in `/etc/profile`:
  - Torque-Befehle und -libraries in den Pfad
  - MKL in den `LD_LIBRARY_PATH`
  - Intel Fortran und C Compiler in den Pfad setzen
  - `.bashrc` im HOME-Verzeichnis verarbeiten
- X11-Forwarding (ssh-Tunnel) möglich

Compiler und Libraries

- gcc/g++ 3.4.6
- Intel Fortran Compiler 9.1.036
- Intel C/C++ Compiler 9.1.042
- Intel MKL (Math Kernel Library) 9.0
- Intel vtune Performance Analyzer 8.0.4
- MPICH2 (1.0.5p4) / SMP

Anwendungen

1. Quantenchemie
   - Gaussian03 *geplant*
   - Turbomole (AK-Lizenz Grimme, OC)
   - MOLPRO (AK-Lizenz Grimme, OC)
2. andere folgen ...

Shared Memory Programmierung: OpenMP

Erzeugung und Lauf eines Parallelprogrammes:

- Pragmas im Quelltext steuern die Parallelisierung
  `!$OMP` *OpenMP-Anweisungen* oder
  `c$OMP` *OpenMP-Anweisungen*
- Compilieren
  `ifort -openmp -o` *prog* `source.f90`
- zur Laufzeit: `$OMP_NUM_THREADS` definiert Zahl der Parallelprozesse
  setzen mit z.B.
  `export OMP_NUM_THREADS=4`
- Systemtools (`ps`, `top` usw.) zeigen nur einen Prozess an, der aber $>$ 100% CPU-Leistung erhält.

Beispiel OpenMP-Subroutine

```
subroutine matrixmult(n,ma,mb,mc)
implicit none
real*8 ma(n,n), mb(n,n), mc(n,n)
integer i,j,k,n

!$OMP PARALLEL DO SHARED (ma,mb,mc)
do i = 1,n
 do j = 1,n
  mc(i,j) = 0
  do k = 1,n
   mc(i,j) = mc(i,j) + ma(i,k)*mb(k,j)
  end do
end do
end do
!$OMP END PARALLEL DO

return
end
```

Parallelprozess

Prozess mit OMP_NUM_THREADS=8

```
top - 08:57:22 up 20 days, 16:41,  3 users,  load average: 1.77, 0.41, 0.21
Tasks: 172 total,   2 running, 170 sleeping,   0 stopped,   0 zombie
Cpu(s): 50.0% us,  0.0% sy,  0.0% ni, 50.0% id,  0.0% wa,  0.0% hi,  0.0% si
Mem:  16529888k total,  4499584k used, 12030304k free,   273744k buffers
Swap:  2031584k total,        0k used,  2031584k free,  3540528k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
32560 hanswurst 25   0 85056  48m 2288 R  800  0.3   2:11.55 matmult-omp
32523 hanswurst 16   0 10544 4144 3168 R    1  0.0   0:00.20 top
32472 hanswurst 16   0 20816 6096 4128 S    0  0.0   0:00.02 sshd
```

Walltime (real) vs. CPUtime

Zahl der Threads führt zu unterschiedlichen Laufzeiten (gleiche CPU-Zeit !)

```
hanswurst@nwzsuperdome:~/Superdome> export OMP_NUM_THREADS=8
hanswurst@nwzsuperdome:~/Superdome> time ./matmult-omp

real    1m46.958s
user    13m28.524s
sys     0m1.497s
hanswurst@nwzsuperdome:~/Superdome> export OMP_NUM_THREADS=2
hanswurst@nwzsuperdome:~/Superdome> time ./matmult-omp

real    6m57.187s
user    13m47.253s
sys     0m0.429s
```

Queueing System: Torque/MAUI

- Open Source Projekt (`www.clusterresources.com`)
- Weiterentwicklung von OpenPBS - kommerzielle Variante: PBSPro (Altair)
- Server: **pbs_server**, steuert die Jobausführung
- Scheduler: Torque-Scheduler (**pbs_Sched**) oder MAUI (open source)
- **pbs_server** und **pbs_sched** laufen nur auf NWZSUPERDOME nur dort können Jobs submittet werden.

Torque/MAUI aus Benutzersicht

Elementarer Befehlssatz:

- `qsub` - Job Submission
- `qstat` - Jobs verfolgen
- `qdel` - Jobs (vorzeitig) beenden
- `qrun` - Job starten
- `pbsnodes` - Node Eigenschaften abfragen

Batch-Job abschicken mit **qsub**

Zwei mögliche Vorgehensweisen:

1. alle Optionen in einem Jobskript (`job`) definieren und dieses mit `qsub` starten (*empfohlen*).

   `qsub job`

2. Optionen dem `qsub` Befehl in der Kommandozeile mitgeben, um `Skript` auszuführen:

   `qsub` *Optionen* `.....` `Skript`

qsub Optionen

Optionen werden entweder in der Kommandozeile oder im Jobskript übergeben:

- **-a date_time**: bestimmt die Zeit, nach der der Job starten soll
- **-e path**: Pfad der (Standard-)Fehlermeldungen
- **-I**: interaktiver Job (nicht im Jobskript)
- **-l resource_list**: definiert Jobresourcen (s.u.)
- **-m mail_options**: **a** bei Abbruch, **b** bei Beginn, **e** bei Ende des Jobs
- **-M user_list**: Email-Adresse für Benachrichtigung
- **-N name**: Name (Label) des Jobs. 15 Zeichen, erstes davon alphabetisches
- **-o path**: Pfad der (Standard-)Ausgabe
- **-q destination**: spezifische Queue, in der der Job laufen soll
- **-v variable_list**: Liste von Umgebungsvariablen, die an den Job übergeben werden
- **-V**: übergibt alle Umgebungsvariablen an den Job

PBS Jobresourcen

können durch Komma getrennt hinter `qsub -l ...` angegeben werden.
Auswahl:

nodes=1:ppn=8	Zahl der Knoten (hier immer 1) und der benötigten CPUs
cput=*HH:MM:SS*	Maximale CPU-Zeit des Jobs
pcput=*HH:MM:SS*	Maximale CPU-Zeit eines Prozesses des Jobs
walltime=*HH:MM:SS*	Maximale Laufzeit (Real-Zeit) des Jobs
pmem=*1000mb*	Benötigter physikalischer Speicher (hier 1 GB)
host=*NAME*	Name des Rechners, auf dem der Job laufen soll
file=*20gb*	maximale Grösse der von Job geschriebenen Files

Bsp:

```
#PBS -l cput=1:00:00,walltime=2:00:00,file=50gb,mem=15mb
```

Siehe auch *man pbs_resources*

PBS Queues

Befehl `qstat -Q` bzw. `qstat -Qf`:

```
Queue: batch
    queue_type = Execution
    total_jobs = 0
    state_count = Transit:0 Queued:0 Held:0 Waiting:0 Running:0 Exiting:0
    resources_max.walltime = 48:00:00
    resources_default.nodes = 1

Queue: test
    queue_type = Execution
    total_jobs = 0
    state_count = Transit:0 Queued:0 Held:0 Waiting:0 Running:0 Exiting:0
    resources_max.walltime = 04:00:00
    resources_default.nodes = 1:test

Queue: short
    queue_type = Execution
    total_jobs = 0
    state_count = Transit:0 Queued:0 Held:0 Waiting:0 Running:0 Exiting:0
    resources_max.walltime = 01:00:00
    resources_default.nodes = 1
```

qsub Shell-Variablen im gestarteten Job:

- **PBS_O_HOST** the name of the host upon which the qsub command is running.
- **PBS_O_QUEUE** the name of the original queue to which the job was submitted.
- **PBS_O_WORKDIR** the absolute path of the current working directory of the qsub command.
- **PBS_ENVIRONMENT** set to PBS_BATCH to indicate the job is a batch job, or to PBS_INTERACTIVE to indicate the job is a PBS interactive job, see -I option.
- **PBS_JOBID** the job identifier assigned to the job by the batch system.
- **PBS_JOBNAME** the job name supplied by the user
- **PBS_NODEFILE** the name of the file contain the list of nodes assigned to the job (for parallel and cluster systems).
- **PBS_QUEUE** the name of the queue from which the job is executed.

Jobskript für **qsub**

```
#!/bin/bash
# hier folgen PBS-Optionen:
#
#PBS -V
#PBS -N superdome-test
#PBS -M hanswurst@uni-muenster.de
#PBS -m ae
# PBE -q test       = Kommentar !!
#PBS -l nodes=1:ppn=16
#
# Queue-Variablen verarbeiten:
#
cd $PBS_O_WORKDIR
export PARNODES=‘wc -l $PBS_NODEFILE |gawk ’print $1’‘
#
# Jobs start here (uncomment and change)

export OMP_NUM_THREADS=PARNODES;
echo $OMP_NUM_THREADS" " >>out ; time ./matmult-omp  2>>out
```

Job verfolgen: `qstat`

Optionen:

- **-a** zeigt alle Jobs
- **-n** zeigt die Knoten an, auf denen der Job läuft
- **-f** gibt alle Informationen zu den Jobs aus
- **-Q** zeigt die laufenden Queues an

```
hanswurst@nwzsuperdome:~/Superdome> qstat
Job id              Name             User            Time Use S Queue
------------------- ---------------- --------------- -------- - -----
437.nwzsuperdome    superdome-test   hanswurst              0 R batch
```

Jeder Job hat eine eindeutige Jobnummer (`NNN.nwzsuperdome`).

Job (vorzeitig) beenden: `qdel`/`qsig`

Jeder Job hat eine eindeutige Jobnummer (`NNN.nwzsuperdome`).

Beenden des Jobs:

`qdel` *NNN*

Senden eines Unix-Signals an den Job:

`qsig -s` *Signal NNN*

NNN (Jobnummer) reicht aus, da es nur einen PBS-Server gibt.

`pbsnodes` - Belegung der Knoten

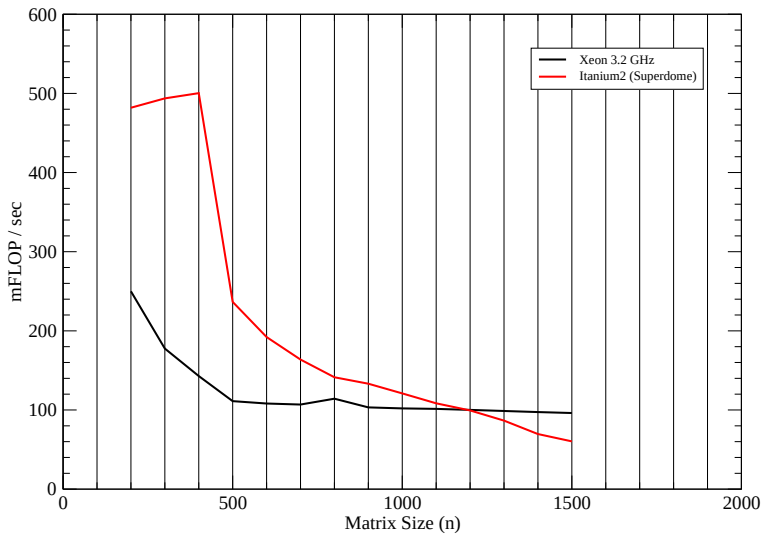
```
[hanswurst@nwzsuperdome ~]# pbsnodes -a
nwzsuperbatch
     state = free
     np = 32
     ntype = cluster
     status = arch=ia64,opsys=linux,uname=Linux nwzsuperbatch 2.6.9-42.EL #1 SMP Wed Jul 12 23:25:09 EDT 2006 ia64,
     sessions=5931,nsessions=1,nusers=1,idletime=1277899,totmem=68516672kb,availmem=66842848kb,physmem=66485088kb,
     ncpus=64,loadave=0.00,netload=1429621060,state=free,jobs=? 15201,rectime=1179228336

nwzsuperdome
     state = free
     np = 8
     ntype = cluster
     status = arch=ia64,opsys=linux,uname=Linux nwzsuperdome 2.6.9-42.EL #1 SMP Wed Jul 12 23:25:09 EDT 2006 ia64,
     sessions=3105 7822,nsessions=2,nusers=2,idletime=97952,totmem=18561472kb,availmem=17963472kb,physmem=16529888kb,
     ncpus=16,loadave=0.83,netload=5562323108,state=free,jobs=? 15201,rectime=1179228331

nwzsupertest
     state = free
     np = 8
     properties = test
     ntype = cluster
     status = opsys=linux,uname=Linux nwzsupertest 2.6.9-42.EL #1 SMP Wed Jul 12 23:25:09 EDT 2006 ia64,sessions=2679,
     nsessions=1,nusers=1,idletime=10401,totmem=10262864kb,availmem=9979920kb,physmem=8231280kb,ncpus=8,loadave=0.05,
     netload=1454144958,state=free,jobs=? 15201,rectime=1179228310
```

Performance

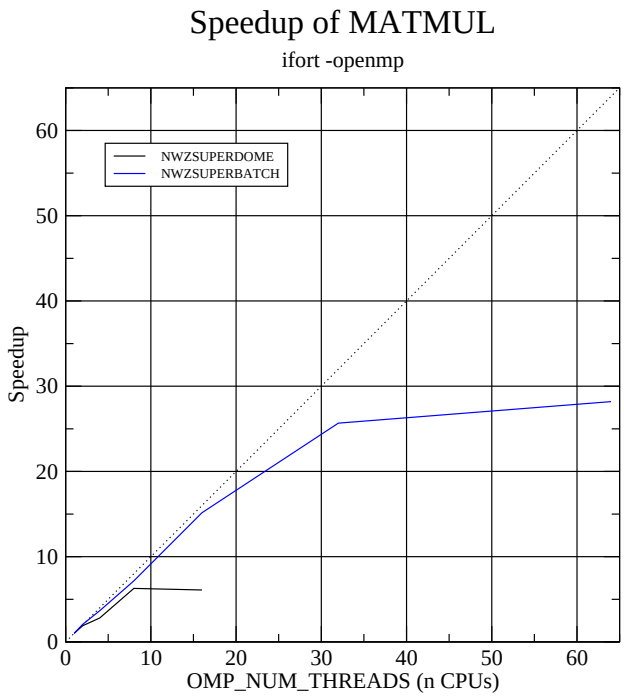
Eigenschaften des Itanium2 SMP-Systems:

- hohe Leistung bei Floating Point Operationen
- für Integer Operationen IA32, EM64T wirtschaftlicher
- Parallelisierung: Shared Memory Programme skalieren im Idealfall linear

Matmult-Prog - Vergleich mit Xeon CPU



Skalierungsverhalten



Info

1. Linux-Gruppe der IVV4 (Treffen: zweiter Mittwoch im Monat, OC)
2. Linux-IVV4 Mailingliste
   `http://listserv.uni-muenster.de/mailman/listinfo/linux-ivv4`
3. demnächst Webseite auf **NWZWiki.uni-muenster.de**

To Do

1. Anbindung des SAN-Array (WORK- und SCRATCH-Platten) kurzfristig (nächste Tage)
   Bis dahin: Arbeiten im HOME-Verzeichnis (keine exzessiven Fileoperationen)
2. testen, testen, testen
3. Nachinstallation von Libraries, Anwendungen
4. Rekonfiguration des Queueing System (in Absprache)
5. Installation von Gaussian03 (erfordert evtl. upgrade auf Rev D2)