Parallele Anwendungen

Ein großer Vorteil von HPC-Systemen ist es Anwendungen parallel auf einer Vielzahl von Systemen laufen zu lassen. Dies ermöglicht es größere Rechnungen und Datenmengen zu verarbeiten.

Dabei ist es notwendig dem Batchsystem mitzuteilen, dass man eine parallele Anwendung starten will. Dazu gibt man im Select-Statement des Jobs die Option "mpiprocs" mit an.

Beispiel 1

```
#!/bin/bash
#PBS -l select=16:ncpus=1:mpiprocs=1:mem=1gb
```

Das Beispiel 1 würde dazu führen, dass einem 16 eigenständige Ranks / Chunks zugeteilt werden mit jeweils 1 Core. Diese Chunks können sich auf einem Knoten befinden, jedoch auch völlig verteilt über den Cluster sein.

Beispiel 2

```
#!/bin/bash
#PBS -l select=2:ncpus=12:mpiprocs=12:mem=1gb
#PBS -l place=scatter
```

Beispiel 2 führt dazu, dass man zwei Ranks / Chunks mit jeweils 12 Cores bekommt, welche jedoch zwingend auf verschiedenen Knoten sind.

Danach gibt es mehrere Möglichkeiten die Anwendung zu starten:

MPI

Das *Message Passing Interface* (MPI) ist ein offener Standard für die Kommunikation zwischen mehreren Prozessen einer parallelen Anwendung.

Aktuell ist dieser Standard in der Version 3.1 verfügbar und wird auch in den neuesten Implementierungen (z.B. Intel MPI) unterstützt.

Auf unserem HPC-System sind folgende MPI-Versionen verfügbar:

MPI	Version	module	Anmerkung
Intel MPI	4.1.0.024	intelmpi/4.1.0.024	
Intel MPI	5.0.2.044	intelmpi/5.0.2.044	**bevorzugt**
MPT	2.06	mpt/2.06	optimiert für SGI UV

pbsdsh

Dieses Tool ist Bestandteil des Batchsystems PBSPro und ermöglicht es ein Programm auf allen Nodes und allen Cores des Jobs zu starten. Dabei ist zu beachten, dass es grundsätzlich keine Möglichkeit zur Unterscheidung der Prozesse gibt. Man ist somit also gezwungen sich selbst darum zu kümmern, dass unterschiedliche Berechnungen ausgeführt werden.

Dies kann beispielsweise auch genutzt werden um einen Performancevergleich zwischen unterschiedlichen Architekturen im Cluster für das eigene Programm zu machen.

Triviales Template

job.sh

```
#!/bin/bash

#PBS -l select=3:ncpus=1:mem=20G
#PBS -l place=scatter
#PBS -l walltime=0:05:00
#PBS -A BenchMarking

cd $PBS_O_WORKDIR

pbsdsh -v -- bash -l $PWD/childs.sh
```

childs.sh

```
#!/bin/bash

echo "node: $PBS_NODENUM"
echo "task: $PBS_TASKNUM"

index=$(expr $PBS_TASKNUM - 2)

job_tmp=/gpfs/scratch/$USER/$PBS_JOBID
master_info=$job_tmp/master.info

mkdir -p $job_tmp

echo "rank: $index"

if [[ $index -eq 0 ]] ; then
        echo "running on master"
        port=$(shuf -i 2000-65000 -n 1)
        echo "" > $master_info
        echo "master=$HOSTNAME-ib" >> $master_info
        echo "port=$port" >> $master_info
        echo "wrote master info to $master_info"
        echo "starting master process"

        .... now you call/module load

else
        echo "running on slave"
        echo "waiting for master for some seconds"
        while [ ! -f $master_info ]
        do
                sleep 1 # or less like 0.2
        done
        echo "found info file $master_info"
        sleep 1

        master=$(grep master $master_info | awk -F'=' '{print $2}')
        port=$(grep port $master_info | awk -F'=' '{print $2}')

        echo "found master at $master:$port"

        ... now your call/module load

f
```