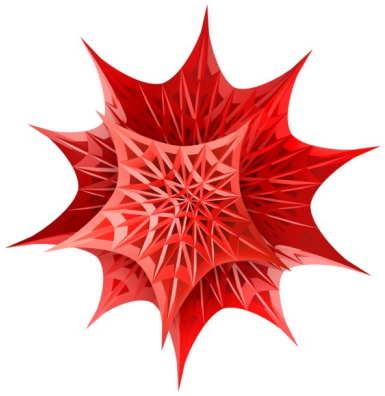
Mathematica im HPC - Betrieb



Zwei Beispiele sollen illustrieren, wie man Mathematica im Parallelbetrieb, einmal im notebook-Modus und einmal im batch-Modus betreiben kann.

Im Notebook:

Hierzu startet man Mathematica und öffnet das Notebook ctest7.nb:

1. Start der Kerne

```
Needs["SubKernels`RemoteKernels`"];

LaunchChrisKernels[] :=
  Module[{numCore, PBSKernel, hostfile, local, remote},
    numCore = 8;(*change to num cores per node*)

  PBSKernel[host_String] :=
      LaunchKernels[RemoteMachine[host,
          "ssh -x -f `1` \
/srv/software/products/Mathematica/8PBS/Executables/math -mathlink -linkmode \
Connect `4` -linkname '`2`' -subkernel -noinit", numCore]];
    hostfile = Environment["PBS_NODEFILE"];(*get all the nodes name*)

  local = ToString@Get["!echo $HOSTNAME"];(*get local node name*)

  remote = Complement[
    DeleteDuplicates@
     Import[hostfile, "List"], {local}];(*get remote node name*)

  If[hostfile =!= $Failed,(*if we're running in a batch job*)
   \
$ConfiguredKernels =
     Join[$ConfiguredKernels, LaunchKernels[], PBSKernel /@ remote];];
    Kernels[]
    ]

 LaunchChrisKernels[];

 DistributeDefinitions["Global`"];
```

2. Anzeigen, welche Kerne angesprochen werden

```
ParallelTable[Labeled[Frame[i], $KernelID], {i, 20},
 Method -> "FinestGrained"]
```

3.Beispiel zum Auffinden von Primzahlen

```
m2[n_] := PrimeQ[2^n - 1];

Print[Timing[Parallelize[Select[Range[10000], m2]]]];
```

Mit 'Evaluation/ParallelKernelStatus' kann man sich den aktuellen Status der Kerne ansehen

Im Batchmodus:

Hierzu habe wir das obig Progtramm in eine Text Datei als Programm für Mathematica ctest7.m geschrieben

```
Needs["SubKernels`RemoteKernels`"];
LaunchChrisKernels[] :=
 Module[{numCore, PBSKernel, hostfile, local, remote},
  numCore = 8;(*change to num cores per node*)
  PBSKernel[host_String] :=
   LaunchKernels[RemoteMachine[host,
     "ssh -x -f `1` /srv/software/products/Mathematica/8PBS/Executables/math -ma
thlink -linkmode Connect `4` -linkname '`2`' -subkernel -noinit", numCore]];
  hostfile = Environment["PBS_NODEFILE"];(*get all the nodes name*)
  local = ToString@Get["!echo $HOSTNAME"];(*get local node name*)
      remote = Complement[DeleteDuplicates@Import[hostfile, "List"], {local}];(*
get remote node name*)
      If[hostfile =!= $Failed,(*if we're running in a batch job*)
$ConfiguredKernels = Join[$ConfiguredKernels, LaunchKernels[], PBSKernel /@ remo
te];];
  Kernels[]
  ];
LaunchChrisKernels[];
DistributeDefinitions["Global`"];
m2[n_] := PrimeQ[2^n - 1];
Print[Timing[Parallelize[Select[Range[20000], m2]]]];
ParallelTable[Labeled[Frame[i],$KernelID],{i,20},Method->"FinestGrained"]
```

Beispiel für das submit-file „submit.pbs“

```
#PBS -l walltime=00:15:00
#PBS -l nodes=10:ppn=8
#PBS -o output.dat
#PBS -j oe
#PBS -q default
#PBS -N MathematicaJob
#PBS -m be
#PBS -V

module load mathematica/8PBS

cd $PBS_O_WORKDIR
echo "Information about the job:"
cat $PBS_JOBID
echo -------
cat $PBS_NODEFILE
echo -e "\n\n"

echo "Started batch processing at `date`."
math -noprompt -run -mathinput -mathoutput <ctest7.m>testout
echo "Ended batch processing at `date`."

exit 0
```

Abschicken des in die Warteschlange

Ein Job wird gestartet, indem die oben erzeugte submit-Datei mittels

qsub submit.pbs

abgeschickt wird, wobei submit.pbs der Name der submit-Datei ist.