Server specific setting

Here we explain how to install and submit jobs in Ohtaka and Kugui at ISSP in u-tokyo.

Use ISSP System B: Ohtaka

- Module settings
  OneAPI(Intel) + OpenMPI is recommended

```
module purge
module load openmpi/4.1.5-oneapi-2023.0.0-classic
```

```
#!/bin/sh
#SBATCH -p i8cpu
#SBATCH -N 1
#SBATCH -n 8
#SBATCH --exclusive

module purge
module load openmpi/4.1.5-oneapi-2023.0.0-classic
ulimit -s unlimited

./InstallAll.py --fc ifort --clean
```

> [!TIP]
> Although `mpirun` is used internally in `InstallAll.py`, it works (perhaps because it uses a single node).

PROF

Additional information on parallel computation

- Specifying the number of parallel processes
  We use MPI not OpenMPI.
  If you run out of memory, reduce the number of cores per node to ensure enough memory per core.

```
#SBATCH -p F16cpu
#SBATCH -N 16
#SBATCH -n 512
#SBATCH -c 4
#SBATCH --ntasks-per-node=32
#SBATCH -t 24:00:00
```

> [!TIP]
> If you encounter MKL (Intel Math Kernel Library) errors during execution, set `-c 1`.

modification of ecalj code for Ohtaka (just for developer)

At ohtaka it uses `srun` instead of `mpirun`.
The difference is written in `SRC/exec/MachineDependency.py`.

Use ISSP System C: Kugui

- Assuming the use of GPU
- Module settings
  nvfortran + OpenMPI

```
module purge
module load nvhpc-nompi/24.7 openmpi_nvhpc compiler-rt tbb mkl
```

> [!TIP]
> The Intel MKL math library is also used for CPU calculations. It can be used from NVIDIA or GNU Fortran.

- To use MPS (multiple MPI processes sharing a single GPU), add the following to `~/.bashrc`

```
if which nvidia-cuda-mps-control > /dev/null 2>&1 ; then
  export CUDA_MPS_PIPE_DIRECTORY=$(pwd)/nvidia-mps-$(hostname)
  export CUDA_MPS_LOG_DIRECTORY=$(pwd)/nvidia-log-$(hostname)
  echo "start nvidia-cuda-mps-control at" $(hostname)
  nvidia-cuda-mps-control -d
fi
```

PROF

- Job script

```
#!/bin/sh
#PBS -q F1accs
#PBS -l select=1:ncpus=64:mpiprocs=64:ompthreads=1
#
ulimit -s unlimited
id=inas6gasb6
gwsc -np 64 -np2 4 --gpu 5 $id > lgwsc
```

- Specify the number of parallel processes for CPU calculations with `-np`.
- Specify the number of parallel processes for GPU calculations with `-np2`. Usually, this is the number of available GPUs.
- `--gpu` uses the GPU version of the executable.

```
> [!TIP]
tip
The GPU version maybe not so effective for small systems, but it can
handle about 40 atoms with 4 GPUs
(it dependes on the computational conditions. Note that the
computational cost increases with the square of the number of k-points).
```