Machine files

ESM-Tools recognises in which machine is operating (using the all_machines.yaml) and loads the content of the <machine>.yaml (esm_tools/configs/machines/mistral.yaml) under the computer section of the config python object

Feature variables

- name – name given to the HPC

Have a look at the esm_tools/configs/machines/mistral.yaml

config object

```
general:
    ...
computer:
    name: mistral
    additional_flags: "--men=0"
    use_hyperthreading: False
    ...
    accounting: True
    ...
    partitions:
        computer:
            name: "compute"
            cores_per_node: "24"
        ...
fesom:
    ...
```

This comes from the **mistral.yaml**, the **computer sections** in the **configuration files/runscript**, and the **environment_changes**

Machine files

Feature variables

SBATCH flags – list of featured variables that can be used from the computer section to set SBATCH flags (and their defaults):

- exclusive_flag: ”--exclusive"
- notification_flag: “--mail_type=<type> --mail_user=<email>”
- single_proc_submit_flag: "--ntasks-per-node=1” (**calculated automatically** by ESM-Tools, based on nproc, nproca/nprocb, omp_num_threads…)
- tasks_flag: "--ntasks=@tasks@” (**calculated automatically** by ESM-Tools, based on nproc, nproca/nprocb, omp_num_threads…)
- partition_flag: "--partition=@partition@” (**calculated automatically** by ESM-Tools with the info from partitions dictionary –next slide-)
- nodes_flag: "--nodes=@nodes@” (**calculated automatically** by ESM-Tools, based on nproc, nproca/nprocb, omp_num_threads…)
- time_flag: "--time=${compute_time}” (**calculated automatically** by ESM-Tools with the info from the experiment time variables -1st day presentation-)
- hyperthreading_flag: False
- **additional_flags**: "” (you can use this variable to add more SBATCH flags as a list)

```
<your_runscript>.yaml
computer:
    add_additional_flags:
        - "--reservation=esmtools"
```

Machine files

Feature variables

- accounting – if true, the user is required to define a general.account in the runscript
- batch_system – defines with <job_scheduler>.py module ESM-Tools will be importing
- sh_interpreter – defines the shebang of the comp-*.sh and the *.run files
- partitions – dictionary that defines the partitions available, including the name and the number of cores per node
- partition – string use to select the partition label to be used from the partitions dictionary

<machine>.yaml

```
partitions:
        <label_for_partition-1>:
                name: <actual_name_for_the_--partition_flag_in_sbatch>
                cores_per_node: <n_cores_per_node>
        <label_for_partition-2>:
                name: <actual_name_for_the_--partition_flag_in_sbatch>
                cores_per_node: <n_cores_per_node>
```

Machine files

Feature variables

- module_actions – a list of module actions to be included in the compilation and *.run files (i.e. module load netcdf, module unload netcdf, module purge, etc).

Syntax

- Omit the “module” word from the command
- Although it has nothing to do with the modules, “source” commands are also accepted here

<machine>.yaml

```
module_actions:
    - "purge”
    - ”source /FILE/PATH”
    - "load netcdf”
```

Machine files

Feature variables

export_vars – a dictionary containing all the variables (and their values) to be exported

Syntax

```
<machine>.yaml
export_vars:
    A_VAR_TO_BE_EXPORTED: the_value
```

```
comp-*.sh or *.run
export A_VAR_TO_BE_EXPORTED=the_value
```

- being a dictionary, export_vars is not allowed to have repeated keys
- could be a problem when environments are required to redefine a variable at different points of the script or from different yamls
- to overcome this limitation, repetitions of the same variable are allowed if the key is followed by an integer contained inside [(int)]:

Machine files

Feature variables

- export_vars – a dictionary containing all the variables (and their values) to be exported

Syntax

- being a dictionary, export_vars is not allowed to have repeated keys
- could be a problem when environments are required to redefine a variable at different points of the script or from different yamls
- to overcome this limitation, repetitions of the same variable are allowed if the key is followed by an integer contained inside [(int)]:

<machine>.yaml

```
export_vars:
    A_VAR_TO_BE_EXPORTED: the_value
    A_VAR_TO_BE_EXPORTED[(1)]: $A_VAR_TO_BE_EXPORTED:another_value
```

comp-*.sh or *.run

```
export A_VAR_TO_BE_EXPORTED=the_value
export A_VAR_TO_BE_EXPORTED=$A_VAR_TO_BE_EXPORTED:another_value
```

The index is removed!

environment_changes

Purpose

Modify the environment defined by the machine file in any section of other configuration files or in the runscript.

Syntax

Define one of the following dictionaries in any section of any yaml file (except the machine yamls):

- environment_changes (applied for both compilation and run time)
- compiletime_environment_changes (only for compilation)
- runtime_environment_changes (only during run time)

Note: this whole syntax might change in the coming year, to a more simple approach (i.e. **computer** section is used, instead of **environment_changes**)

environment_changes

Syntax

The environment_changes dictionaries can contain:

- add_module_actions: to expand the module_actions list in the <machine>.yaml
- add_export_vars: to expand the export_vars dictionary in the <machine>.yaml (redundant variables will be overwritten by the highest file in the hierarchy)
- Any other variable or choose to be resolved at the environment stage (i.e. select a MPI configuration for an MPI choose inside the <machine>.yaml)

environment_changes

Example

fesom-2.1.yaml

```
compiletime_environment_changes:
        add_export_vars:
                taken2from:      fesom2_compile
        choose_computer.name:
                juwels:
                        add_module_actions:
                                - "unload ParaStationMPI"
                                - "load ParaStationMPI/5.4.4-1"
                                - "load netCDF-Fortran/4.4.5"
```

comp-fesom-2.1.yaml in Juwels

```
#<module actions defined in juwels.yaml>
module unload ParaStationMPI
module load ParaStationMPI/5.4.4-1
module load netCDF-Fortran/4.4.5

#<exported variables as defined in the juwels.yaml>
export taken2from="fesom2_compile"
```

environment_changes

Coupled setups

- compiletime_environment_changes allows for different compilation environments under the same coupled setup (i.e. in AWI-ESM-2, the echam.yaml and the fesom-2.1.yaml both contain compiletime_environment_changes, resulting in different compilation environments during esm_master comp-awiesm-2.1)
- The runtime environment for online coupled setups is the result of combining:
  1. The environment information of the <machine>.yaml
  2. The environment information contained in the runtime_environment_changes in the component files (i.e. echam.yaml, fesom-2.1.yaml, ...)
  3. The environment information contained in the runtime_environment_changes in the setup file (awiesm.yaml)

environment_changes

Coupled setups

- To define a general environment_changes **for all the components of a setup**, include the environment_changes inside the general section of the setup file. **This will ignore all the** environment_changes **defined by the standalone files**. It is still possible to add component-specific environment_changes **from the component section inside the setup file**.