Appendix A. xml templates

```
<submodel id="" stateful="">
   <instance class="" id="" type="">

      <restrictions>
         <cpu number="" min="" max="" />
      </restrictions>

      <available_resources>
         <resource name="" nodeType=""/>
      </available_resources>

   </instance>
</submodel>
```

Listing 1: Submodel definitions (matrix.xml), this template shows the part of defining a single scale. Resource name and nodeType specifies the resource name in the EEE and the type of the node (see Table 1) in this resource. The complete submodels definition is explained in section 4 and a full description with an example is shown in Appendix E.

```
<multiscale>
   <info>
      <job appID="" project="">
         <computing></computing>
         <modeltime></modeltime>
         <task persistent="" taskId="">
            <numberofcores min="" max="" />
         </task>
      </job>
   </info>

  <topology>
     <!--instances for each submodel-->
     <instance id="" submodel=""/>
     ...
     <!--coupling of instances-->
     <coupling from="" to=""/>
     ...
   </topology>

    <middleware name="">
       <execution type="">
          <executable>
```

Listing 2: Multiscale coupling file (multiscale.xml), the template contains input requirements to run the multiscale application, such as minimum and maximum number of cores required by the user to run a simulation, instances, coupling topologies and middleware specific requirements. The complete multiscale coupling description is explained in section 4 and a full description with an example is shown in Appendix E.

```
<performance>
   ...
   <instance id="">
      <benchmark>
         <iterations></iterations>
            </benchmark>

            <resources name="">
                  <nodeType></nodeType>
                  <numberOfCores></numberOfCores>
                  <wallClockTime></wallClockTime>
            </resources>
      </instance>
</performance>
```

Listing 3: Template of the performance section (in matrix.xml), which is, as described in 3.1, an output of the Translation Component. The complete performance section is explained in section 4 and a full description with an example is shown in Appendix E.

```
   <kernels>
      <kernel id="">
         <helper id=""/>
      </kernel>
   </kernels>

   <classes>
      <class id="">
         <node host=""></node>
      </class>
   </classes>

   <plans>
      <plan id="">
         <criteria>
            <time></time>
         </criteria>
         <group>
```

```
19          <kernel refid="">
20             <class refid="">
21                <cores></cores>
22             </class>
23          </kernel>
24          </group>
25       </plan>
26    <plans>
```

Listing 4: Template of Pattern-Driven Planner output. The first part addresses the kernels and helper references. The second provides the references for the classes of nodes. The last part is the set of allocation plans. a plan is a group of single scale models (can be the whole multiscale model or part of it) with a number of classes of nodes and cores assigned to each submodel in a group. Note that the chosen cost criteria is time to completion. This file is input to the Execution component. The complete plan structure is explained in section 3.3 and a full description is given in Appendix F.

Appendix B. Fusion xMML description

The following is the Submodels definitions for the fusion application [9]. For more information about the xMML format, we refer to [10].

```
<?xml version="1.0" encoding="UTF-8"?>
<model xmlns="http://www.mapper-project.eu/xmml" id="TTE" name="turbulence-transport-equilibrium_workflow" xmml_version="0.4">
  <definitions>
    <mapper id="dupEquil" type="fan-out">
      <ports>
        <in id="equilibrium_in" />
        <out id="equilibrium_out1" />
        <out id="equilibrium_out2" />
        <out id="equilibrium_out3" />
      </ports>
    </mapper>
    <mapper id="dupCorep" type="fan-out">
      <ports>
        <in id="coreprof_in" />
        <out id="coreprof_out1" />
        <out id="coreprof_out2" />
        <out id="coreprof_out3" />
      </ports>
    </mapper>
    <submodel id="initialState" init="yes">
      <timescale delta="0" total="0" />
      <spacescale delta="0" total="0" />
      <ports>
        <out id="inputCPOs" operator="Of" />
      </ports>
    </submodel>
    <submodel id="transport">
      <timescale delta="1E-0" total="1E+1" />
      <spacescale delta="1E-2" total="1E+1" />
      <ports>
        <in id="inputCPOs" operator="finit" />
        <in id="coreprof_in" operator="S" />
        <in id="coretransp_in" operator="S" />
        <in id="equilibrium_in" operator="S" />
        <out id="coreprof_out" operator="Oi" />
        <out id="equilibrium_out" operator="Oi" />
      </ports>
    </submodel>
    <submodel id="turbulence" stateful="yes">
      <timescale delta="1E-6" total="1E-4" />
```

```
      <spacescale delta="1E-3" total="1E-2" />
      <ports>
        <in id="coreprof_in" operator="finit" />
        <in id="equilibrium_in" operator="finit" />
        <out id="coretransp_out" operator="Of" />
      </ports>
    </submodel>
    <submodel id="equilibrium">
      <timescale delta="0" total="0" />
      <ports>
        <in id="equilibrium_in" operator="finit" />
        <out id="equilibrium_out" operator="Of" />
      </ports>
    </submodel>
    <submodel id="fluxesToDVs">
      <timescale delta="0" total="0" />
      <ports>
        <in id="coreprof_in" operator="finit" />
        <in id="equilibrium_in" operator="finit" />
        <in id="coretransp_in" operator="finit" />
        <out id="coretransp_out" operator="Of" />
      </ports>
    </submodel>
  </definitions>
  <topology>
    <instance id="init" submodel="initialState" />
    <instance id="transp" submodel="transport" />
    <instance id="turb" submodel="turbulence" />
    <instance id="equil" submodel="equilibrium" />
    <instance id="dupEquil" mapper="dupEquil" />
    <instance id="dupCorep" mapper="dupCorep" />
    <instance id="f2dv" submodel="fluxesToDVs" />
    <coupling from="init.inputCPOs" to="transp.inputCPOs" />
    <coupling from="transp.coreprof_out" to="dupCorep.coreprof_in" />
    <coupling from="transp.equilibrium_out" to="equil.equilibrium_in" />
    <coupling from="dupCorep.coreprof_out1" to="turb.coreprof_in" />
    <coupling from="dupCorep.coreprof_out2" to="f2dv.coreprof_in" />
    <coupling from="equil.equilibrium_out" to="dupEquil.equilibrium_in" />
    <coupling from="dupEquil.equilibrium_out1" to="turb.equilibrium_in" />
    <coupling from="dupEquil.equilibrium_out2" to="f2dv.equilibrium_in" />
    <coupling from="turb.coretransp_out" to="f2dv.coretransp_in" />
    <coupling from="f2dv.coretransp_out" to="transp.coretransp_in" />
  </topology>
</model>
```

Listing 5: xMML description for fusion application.

Appendix C. Fusion MUSCLE2 configuration file (cxa ruby file)

Below we list the Muscle2 configuration file for the fusion application. Key features are the 'declare kernels' where the single scale models and other relevant kernels are defined, and the 'configure coupling' section where the wiring between the single scale models is specified.

```
# total simulation time
$env["max_timesteps"] = "1.0"
# declare kernels
init = Instance.new('init')
ETS = Instance.new('ETS')
GEM = Instance.new('GEM')
CHEASE = Instance.new('CHEASE')
equildup = Instance.new('equildup')
corepdup = Instance.new('corepdup')
coreprof2equilibrium = Instance.new('coreprof2equilibrium')
# parameters
init['dt']="0.0e0"
init['T']="0.0e0"
init['dx']="0.0e0"
init['X']="0.0e0"
ETS['dt']="1.0e0"
ETS['T']="1.0"
ETS['dx']="1.0e-2"
ETS['X']="1.0"
GEM['dt']="1.0e-6"
GEM['T']="1.0"
GEM['dx']="1.0e-3"
GEM['X']="1.0e-2"
CHEASE['dt']="0.0e0"
CHEASE['T']="1.0"
coreprof2equilibrium['dt']="0.0e0"
coreprof2equilibrium['T']="1.0"
# configure coupling
init.couple(ETS, 'initCPOs')
ETS.couple(corepdup, 'coreprof' => 'corep_in')
corepdup.couple(coreprof2equilibrium, 'corep_out1' => 'coreprof')
corepdup.couple(GEM, 'corep_out2' => 'coreprof')
coreprof2equilibrium.couple(CHEASE, 'equilibrium' => 'equilibrium_in')
CHEASE.couple(equildup, 'equilibrium_out' => 'equil_in')
equildup.couple(GEM, 'equil_out1' => 'equilibrium')
equildup.couple(ETS, 'equil_out2' => 'equilibrium')
GEM.couple(ETS, 'coretransp')
```

Listing 6: Ruby file for fusion application.

Appendix D. Task graph for the fusion fast track application

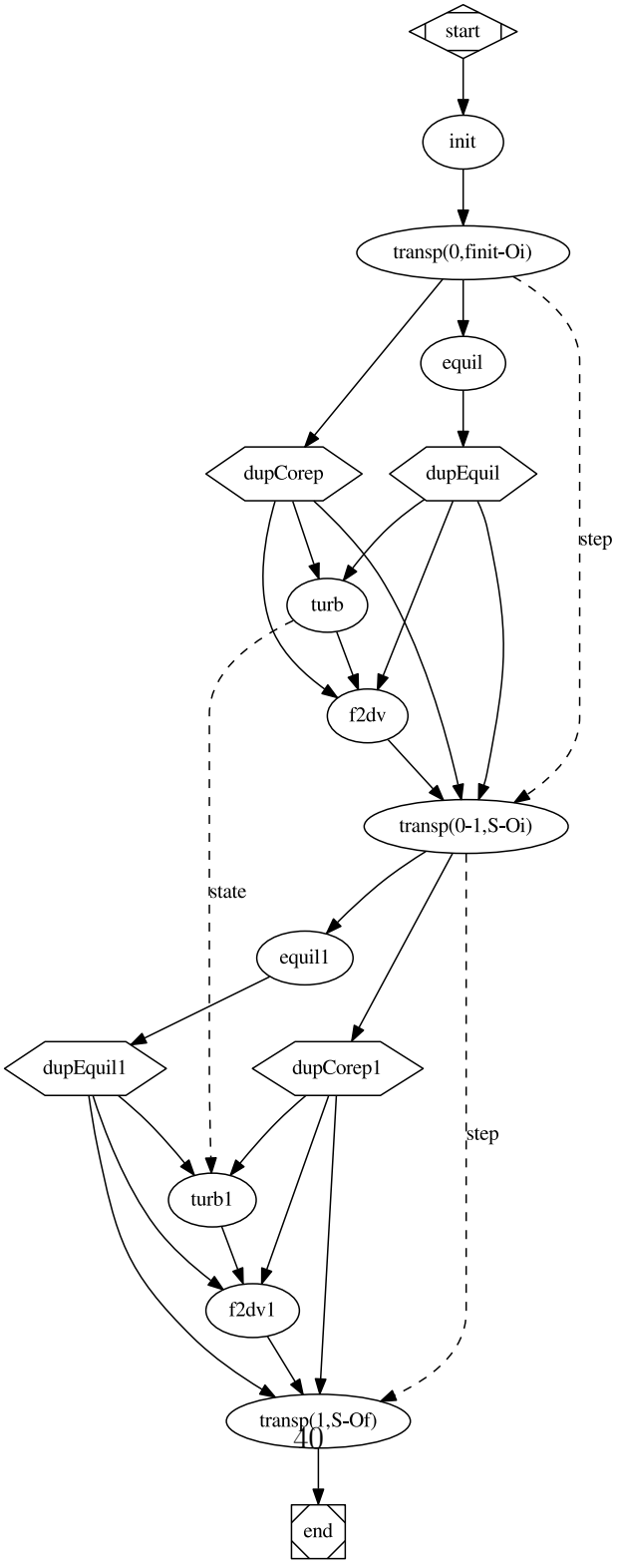

Figure D.8: Task graph for two multiscale iterations of fusion application.

Appendix E. matrix.xml and multiscale.xml for example of fusion application

matrix.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<submodels id="TTE" name="turbulence-transport-equilibrium_workflow"
    xmml_version="0.4">
  <submodel id="initialState">
    <instance class="NativeKernel" id="init" type="helper">
      <restrictions>
        <cpu number="1" min="1" max="1" />
      </restrictions>
      <available_resources>
        <resource name="supermuc" nodeType="thin" />
        <resource name="eagle" nodeType="haswell_128" />
      </available_resources>
    </instance>
  </submodel>
  <submodel id="transport">
    <instance class="NativeKernel" id="transp" type="helper">
      <restrictions>
        <cpu number="1" min="1" max="1" />
      </restrictions>
      <available_resources>
        <resource name="supermuc" nodeType="thin" />
        <resource name="eagle" nodeType="haswell_128" />
      </available_resources>
    </instance>
  </submodel>
  <submodel id="turbulence" stateful="yes">
    <instance class="MPIKernel" id="turb" type="kernel">
      <restrictions>
        <cpu number="(2**x)" min="64" max="2048" />
      </restrictions>
      <available_resources>
        <resource name="supermuc" nodeType="thin" />
        <resource name="eagle" nodeType="haswell_128" />
      </available_resources>
    </instance>
  </submodel>
  <submodel id="equilibrium">
    <instance class="NativeKernel" id="equil" type="helper">
      <restrictions>
        <cpu number="1" min="1" max="1" />
      </restrictions>
```

```
      <available_resources>
        <resource name="supermuc" nodeType="thin" />
        <resource name="eagle" nodeType="haswell_128" />
      </available_resources>
    </instance>
  </submodel>
  <submodel id="fluxesToDVs">
    <instance class="NativeKernel" id="f2dv" type="helper">
      <restrictions>
        <cpu number="1" min="1" max="1" />
      </restrictions>
      <available_resources>
        <resource name="supermuc" nodeType="thin" />
        <resource name="eagle" nodeType="haswell_128" />
      </available_resources>
    </instance>
  </submodel>
  <mapper id="dupEquil" type="fan-out" />
  <mapper id="dupCorep" type="fan-out" />
</submodels>
```

Listing 7: matrix.xml. Two architectures are shown, "supermuc" and "eagle". In these benchmarks, we consider the "thin" node type in SuperMUC and "*haswell*_128" node type on Eagle to measure the performance of the benchmark.

multiscale.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<multiscale>
  <info>
    <job appID="TTE" project="compat">
      <computing>ES</computing>
      <modeltime>transp + equil + turb</modeltime>
      <task persistent="true" taskId="task">
        <numberofcores min="64" max="2048" />
      </task>
    </job>
  </info>
  <topology>
    <!--instances for each submodel-->
    <instance id="init" submodel="initialState" />
    <instance id="transp" submodel="transport" />
    <instance id="turb" submodel="turbulence" />
    <instance id="equil" submodel="equilibrium" />
    <instance id="dupEquil" mapper="dupEquil" />
```

```
    <instance id="dupCorep" mapper="dupCorep" />
    <instance id="f2dv" submodel="fluxesToDVs" />
    <!--coupling of instances-->
    <coupling from="init" to="transp" />
    <coupling from="init" to="transp" />
    <coupling from="init" to="transp" />
    <coupling from="init" to="transp" />
    <coupling from="init" to="transp" />
    <coupling from="init" to="transp" />
    <coupling from="transp" to="dupCorep" />
    <coupling from="transp" to="equil" />
    <coupling from="dupCorep" to="turb" />
    <coupling from="dupCorep" to="f2dv" />
    <coupling from="dupCorep" to="transp" />
    <coupling from="equil" to="dupEquil" />
    <coupling from="dupEquil" to="turb" />
    <coupling from="dupEquil" to="f2dv" />
    <coupling from="dupEquil" to="transp" />
    <coupling from="turb" to="f2dv" />
    <coupling from="f2dv" to="transp" />
  </topology>
  <middleware name="QCG">
    <execution type="compat">
      <executable>
        <application name="muscle2" />
      </executable>
      <arguments>
        <value>short.cxa.rb</value>
      </arguments>
      <stdout>
        <directory>
          <location type="URL">gsiftp://qcg.man.poznan.pl/~/qcg-test-demo
              </location>
        </directory>
      </stdout>
      <stderr>
        <directory>
          <location type="URL">gsiftp://qcg.man.poznan.pl/~/qcg-test-demo
              </location>
        </directory>
      </stderr>
      <stageInOut>
        <!--Can contain <file> or <directory> tags, type=in or out-->
        <file name="short.cxa.rb" type="in">
```

```
61            <location type="URL">gsiftp://qcg.man.poznan.pl//Fusion/qcg-
                  test/short.cxa.rb</location>
62          </file>
63          <file name="inputs.tgz" type="in">
64            <location type="URL">gsiftp://qcg.man.poznan.pl///Fusion/
                  FastTrack/qcg-test/inputs.tgz</location>
65          </file>
66          <file name="extract_inputs.sh" type="in">
67            <location type="URL">gsiftp://qcg.man.poznan.pl//Fusion/
                  FastTrack/qcg-test/extract_inputs.sh</location>
68          </file>
69          <file name="prepare_outputs.sh" type="in">
70            <location type="URL">gsiftp://qcg.man.poznan.pl///Fusion/
                  FastTrack/qcg-test/prepare_outputs.sh</location>
71          </file>
72          <directory name="outputs" type="out">
73            <location type="URL">gsiftp://qcg.man.poznan.pl/~/qcg-test-demo
                  </location>
74          </directory>
75        </stageInOut>
76        <environment>
77          <variable name="QCG_MODULES_LIST">compat/apps/fusion/1.1</variable
                >
78          <variable name="QCG_PREPROCESS">extract_inputs.sh</variable>
79          <variable name="QCG_POSTPROCESS">prepare_outputs.sh</variable>
80        </environment>
81      </execution>
82    </middleware>
83  </multiscale>
```

Listing 8: multiscale.xml. Multiscale models specification in the targeted machines are shown.

Appendix F. QCG plans for fusion application

For more information on QCG middleware, please see [24].

```
<?xml version="1.0" encoding="UTF-8"?>
<qcgJob>
  <task persistent="true" taskId="ES-task-2">
    <requirements>
      <patternTopology>
        <kernels>
          <kernel id="turb">
            <helper id="init" />
            <helper id="transp" />
            <helper id="equil" />
            <helper id="f2dv" />
            <helper id="dupEquil" />
            <helper id="dupCorep" />
          </kernel>
        </kernels>
        <classes>
          <class id="c1">
            <node host="supermuc">thin</node>
            <node host="eagle">haswell_128</node>
          </class>
          <class id="c2">
            <node host="supermuc">thin</node>
          </class>
          <class id="c3">
            <node host="eagle">haswell_128</node>
          </class>
        </classes>
        <plans>
          <plan id="plan1">
            <criteria>
              <time>P0Y0M0DT0H10M</time>
            </criteria>
            <group>
              <kernel refid="turb">
                <class refid="c1">
                  <cores>1024</cores>
                </class>
              </kernel>
            </group>
          </plan>
          <plan id="plan2">
            <criteria>
```

```
43                <time>P0Y0M0DT0H05M</time>
44              </criteria>
45              <group>
46                <kernel refid="turb">
47                  <class refid="c1">
48                    <cores>2048</cores>
49                  </class>
50                </kernel>
51              </group>
52            </plan>
53            <plan id="plan3">
54              <criteria>
55                <time>P0Y0M0DT0H20M</time>
56              </criteria>
57              <group>
58                <kernel refid="turb">
59                  <class refid="c1">
60                    <cores>512</cores>
61                  </class>
62                </kernel>
63              </group>
64            </plan>
65          </plans>
66          <reservations>
67            <reservation host="eagle">eagle-res-1</reservation>
68            <reservation host="inula">inula-res-2</reservation>
69            <reservation host="zeus">zeus-res-2</reservation>
70          </reservations>
71          &gt;
72        </patternTopology>
73      </requirements>
74      <execution type="compat">
75        <executable>
76          <application name="muscle2" version="compat-1.2" />
77        </executable>
78        ...
79      </execution>
80    </task>
81  </qcgJob>
```

Listing 9: A set of QCG plans to run fusion application.

Appendix G. FabSim configuration files (YAML files) for BAC application

For more information on FabSim, we refer to [25].

```
production:
local_configs: "/net/dirac/mnt/store6/dave/janssen/out-stage/"
model: "NAMD"
name : namd
file_type: URL
benchmark_scaling: 1
iterations_scaling: 1
dependencies: namd
restrictions:
cpu:
min: 1
max: 512
execution:
arguments:
script:
"cd␣input/mineq_confs\n
namd_exec␣eq0-rep${PS_rep}.conf␣>␣../replicas/rep${PS_rep}/equilibration/
    eq0.log\n
namd_exec␣eq1-rep${PS_rep}.conf␣>␣../replicas/rep${PS_rep}/equilibration/
    eq1.log\n
namd_exec␣eq2-rep${PS_rep}.conf␣>␣../replicas/rep${PS_rep}/equilibration/
    eq2.log\n
cd␣../../sim_confs\n
namd_exec␣sim1-rep${PS_rep}.conf␣>␣../../input/replicas/rep${PS_rep}/
    simulation/sim${PS_rep}.log"
value: text
production2:
local_results: "~/work/fabsim/FabSim/results"
model: "Amber"
name: mmpbsa
restrictions:
cpu:
min: 1
max: 1
number: 1
dependencies: amber
execution:
arguments:
script:
"cd␣rep${PS_rep}/fe-calc
```

```
37 mpirun␣-n␣${QCG_PROCS}␣MMPBSA.py.MPI␣-i␣../../nmode.in␣-sp␣../../build/
       complex.top␣-cp␣../../build/com.top␣-rp
38 ../../build/rec.top␣-lp␣../../build/lig.top␣-y␣../simulation/sim1.dc/␣>␣
       amber.log"
```

Listing 10: Yaml desciption of BAC application.