Collaborative Grant Application form: Compute component of the Flemish Tier-1 supercomputing platform

Title of the application:
Applicant name, first name:
E-mail address:
Consortium information (research groups, departments, institutions):
VSC id of all mandated persons, separated by commas:
Core hours applied for: GPU hours applied for:
Largest amount of scratch disk required (in TiB):
Largest associated number of files:
List of simulation codes and their version numbers:

1. Include a short description of your research project, in layman's terms wherever possible, with a view to dissemination. Explicitly mention the scientific questions that you are planning to address and the overall scientific goals of the project. (max. 1 A4 in Arial 12)

2. Persons mandated to compute on the Tier-1 within the framework of the present project. Please provide for every person:

- Name, first name
- VSC id
- Institution
- Research group / department:
- Experience with using particular HPC resources (i.e. Tier-0/Tier-1/Tier-2 infrastructure) in Belgium and abroad. Specify both the name of infrastructure and number of years it was used.
- List of computing time allocations received during the past two years, on the Flemish Tier-1 systems, as well as other Tier-1 and Tier-0 systems.

3. Why does this project need to run on a Tier-1 system? What is the added value of a collaborative grant for your consortium over one or more regular proposals?

4. Provide information for each software package that will be used.
   - If centrally installed on Tier-1 compute or a Tier-2 system within VSC, state the module name.
   - If not open source software, state that the associated license can be validly used by all mandated users on the desired Tier-1 compute partition (BrENIAC/Hortense). Add a copy of the signed license to this application.

5. Provide the results of parallel efficiency tests for each software package that will be used.
   - Perform these benchmark tests on the Tier-1 compute infrastructure (using, e.g., a Starting Grant), on the desired partition (Hortense/BrENIAC). Make sure to mention this particular partition.
   - Use system/problem sizes that closely reflect those of the intended computational tasks (e.g., same mesh size, actual molecular system, similar I/O pattern, same communications patterns, etc.). If a different system/problem size is used in the tests, describe how it relates to the problem size in the application. State if I/O has been included in the tests. For example, simply run your application tasks for a limited number of iterations.
   - List the results in a table and plot efficiency versus number of cores or number of GPUs using a log scale x-axis (see example Table 1 and Plot 1).
   - Start the scaling tests of your code using the smallest number of cores or GPUs possible. If possible, the baseline is using 1 core or 1 GPU on a dedicated node. If not possible, explicitly state why.
   - Wall clock times are preferably obtained by averaging the timing results of several similar simulations for each node/core/GPU configuration. This is required when task farming jobs that vary significantly in run time. In that case, give an indication of variation (e.g. standard deviation). Otherwise, report the minimum time of at least three separate runs.

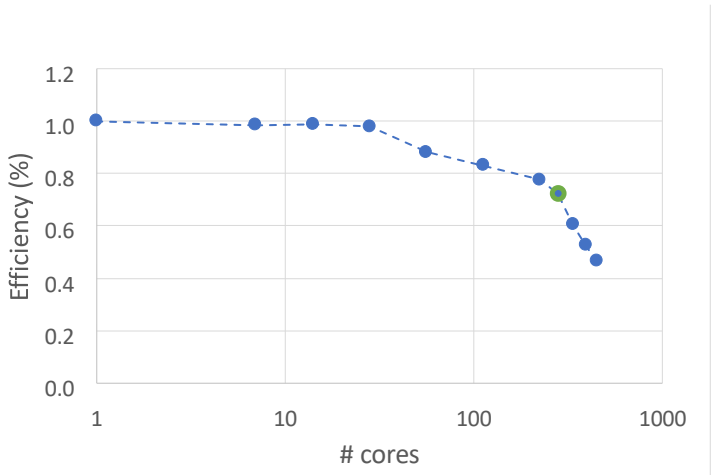
- Task loads that don’t use the maximum number of cores/GPUs per node are preferentially packed together, using the worker framework, atools, …
  Explain anomalies in plot and table.
- Clarify, on the basis of the parallel efficiency plot and table, which number of nodes and cores/GPUs you plan to use for your computational tasks (cf. Section 7) and explain why.

Example Table 1

Number of nodes	Total number of cores	Wall clock time (s)	Speed-up (w.r.t. baseline)	Efficiency
$A_{baseline}$	$B_{baseline}$	$C_{baseline}$	*1.00*	*1.00*
A1	*B1*	*C1*	$C_{baseline}/C1$	$(B_{baseline}*C_{baseline})/(B1*C1)$
A2	*B2*	*C2*	$C_{baseline}/C2$	$(B_{baseline}*C_{baseline})/(B2*C2)$
Baseline = minimal configuration with which your computational task can be carried out on Tier-1.				
Wall clock time is difference between start/end of the computational task, including any I/O operations as part of that task.				

Tier-1 partition on which benchmark was performed: …				
Number of nodes	Total number of cores	Wall clock time (s)	Speed-up (w.r.t. baseline)	Efficiency
1	1	4000	1.00	1.00
1	7	580	6.90	0.99
1	14	289	13.84	0.99
1	28	146	27.40	0.98
2	56	81	49.38	0.88
4	112	43	93.02	0.83
8	224	23	173.91	0.78
10	280	19.8	202.02	0.72
12	336	19.7	203.05	0.60
14	392	19.4	206.19	0.53
16	448	19.2	208.33	0.47

Example Plot 1



The optimal number of cores in this example is 280, as parallel efficiency quickly drops below 70% when more cores are used.

6. Justify the number of core-hours and GPU-hours, and storage volume applied for.
   Describe your planned computational tasks and the sequence in which these tasks will be performed. Start from the examples in Table 2 and Table 3 and adjust them to your project.

   Note that per requested GPU-hour on Hortense, you will automatically receive 12 core-hours on the CPU cores of the node containing that GPU unit. These core-hours do not need to be specified explicitly in Table 3.

   Provide additional descriptions for the computational tasks listed in the table. Resource estimates (wall clock time, number of nodes/cores/GPUs, memory, storage) should be based on the results of actual calculations on BrENIAC or Hortense (via, e.g., a Starting Grant) for system/problem sizes that match closely those of the intended computing tasks (e.g., same mesh sizes, actual molecular system, same I/O pattern, same amount of communications, etc.). If you plan to run the tasks concurrently, mention this in the description, so you can specify the correct total amount of scratch space required.

Example Table 2

	Core-hour calculation							Storage volume estimate	
Computational task	Number of such jobs	Wall clock time (in hours) per job	Number of Tier-1 nodes per job	Number of Tier-1 cores per node per job	Total core-hours per task	Memory usage (GiB) per node per job	OpenMP / MPI / OpenMP + MPI (hybrid) / worker framework / atools / etc.	Tier-2 DATA/HOME volume (TiB) + number of files	Tier-1 SCRATCH+ volume (TiB) number of files
Task • software X • parameters/conditions • system/mesh size • …	A	B	C	D	= A x B x C x D				
Task example CP2K • CP2K – MD • 100 ns runs • PBE functional • 1 -> 5 water molecules	5	48	10	28	67200	64	MPI	0 TiB 0 files	0.1 TiB 5000 files
Task example worker • MDTraj postprocessing • 5000 files	10000	0.5	1	1	5000	3 (84 GiB for 28 jobs in one node)	These single-core jobs will be packed within 1 node using worker framework	1 TiB 10000 files	0.1 TiB 5000 files
					Sum of core-hours applied for = …				Largest amount of scratch disk required + number of associated files = …

Important information:		*3 days is the maximum wall clock time for any job.*				*Memory limits (GiB/CPU node) BrENIAC: 128->256 Hortense: 256->512*			

The example CP2K task needs to run 5 times, for a molecular system containing 1 to 5 water molecules. Based on timing runs on Tier-1 compute partition BrENIAC, we found that one such job runs for 48 hours on 10 nodes, using all the cores

(28) in the node. The job needs 64 GiB RAM in each node and produces 20 GiB of SCRATCH storage (1000 files). Since the 5 jobs (for the 5 listed molecular systems) will be run concurrently, 5 x 20 GiB = 100 GiB of scratch disk space is required (and 5 x 1000 = 5000 files) for the entire task.

File postprocessing with the MDTraj tool is done in the example worker task, where 10000 jobs need to run on 5000 files on the SCRATCH volume to generate 10000 files on the Tier-2 DATA volume. Each job runs on 1 core. Based on 5 timing runs on Tier-1 compute partition BrENIAC, we found that the job duration varies between 25 and 28 minutes, and memory usage is 3 GiB at most. To be on the safe side, we foresee 30 minutes per job (0.5 hours). Wherever possible, 28 jobs will be packed on a single node using the worker framework, so 28 jobs require 1 full node for 30 minutes. For the entire task, 5000 core-hours are required.

Example Table 3

	GPU-hour calculation							Storage volume estimate	
Computational task	Number of such jobs	Wall clock time (in hours) per job	Number of Tier-1 nodes per job	Number of Tier-1 GPUs per node per job	Total GPU-hours per task	Memory usage (GiB) per node per job	OpenMP / MPI / OpenMP + MPI (hybrid) / worker framework / atools / etc.	Tier-2 DATA/HOME volume (TiB) + number of files	Tier-1 SCRATCH volume (TiB) + number of files
Task • software X • parameters/conditions • system/mesh size • …	A	B	C	D	= A x B x C x D				
Task example QE • Quantum Espresso • 1,500 compounds • SCF calculation	1500	8	1	2	24000	106	MPI & OpenMP	0.4 TiB 2500 files	1.2 TiB 7500 files
					Sum of GPU-hours applied for = …				Largest amount of scratch disk required + number of associated files = …

Important information:		*3 days is the maximum wall clock time for any job.*				*Memory limits (GiB/GPU node) Hortense: 256*			

The example QuantumEspresso task needs to run 1500 times, to perform an SCF calculation on 1500 different compounds. All tasks can be executed independently of each other. Based on timing runs on the GPU nodes of Tier-1 compute partition Hortense, we found that one such job runs for 8 hours on 1 node, using 2 GPUs along with 24 CPUcores in the GPU node. Each job requires 106 GiB of RAM, therefore two jobs can run simultaneously on a Hortense GPU node (256 GiB). The worker framework will be used to pack 2 tasks in one job that will make sure both end up on one GPU node, optimally using all GPUs of that node. Each job generates 5 files that total 0.8 GiB. For all tasks, this amounts to 1500 x 0.8 GiB = 1.2 TiB of SCRATCH storage (7500 files). These will be regularly offloaded to the Tier-2 DATA storage in a compressed format.

7. Describe how you will manage the workflow and the resources requested in the period during which the task is to be performed.

   In case you will launch a large number of computational tasks, describe how you will manage your jobs and provide details regarding job management, automation and dataflow. Will you make use of a task/workflow manager, such as the worker framework, atools or something similar? On which infrastructure or node will this manager run?

   Please present how you will manage your data. Describe how the transfer of files to/from the Tier-1 compute partition will be managed and automated, if data reduction and/or compression of files will be performed. If available, provide information about IOPS.