REPORT

"First INFN course on practical application integration in GRID"

Applicazion Name: *Fire risk assessment and coordination*

Acronym: *RIS.I.CO.*

Applicazion's domain of interest: *Environmental Monitoring*

Involved Foundations: *CIMA Foundation, INFN Padova, IMAA*

Authors , workgroup: *Mirko D'Andrea – CIMA Foundation (mirko.dandrea@cima.unige.it), Stefano Dal Pra – INFN Padova (stefano.dalpra@pd.infn.it), Valerio Angelini – IMAA (angelini@imaa.cnr.it)*

1. Application general description.

RIS.I.CO. is a prevision system designed for the assessment of the distribution of dynamic wild land fire risk over the whole Italian territory which have been developed by the CIMA Foundation for the Italian Civil Protection Agency.
Such an assessment takes place on the basis of static information concerning the vegetation cover and the topography of the territory and of dynamic information consisting in the meteorological forecast, over a certain time horizon (72 hours), provided by a Limited Area Model.

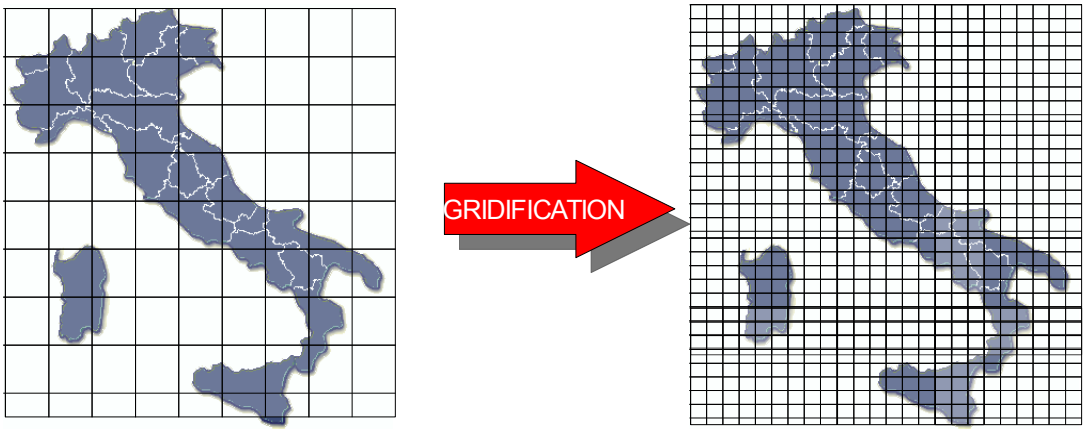
RISICO performs his prediction about potential dangerousness over a discretized domain of 1 km cells (these are about 330.000) covering the whole Italian territory. The model is intrinsically parallelizzable being that computation only takes into account cell status without considering correlations between them.
Input data for the model are provided by a network of meteo sensors as well as atmospherical models (Local Area Model). These data gets daily retrieved and aggregated.
RISICO also needs a static set of information describing cells respect to orographic and vegetation characteristics for the discretized domain. The adopted resolution for cells influences accuracy for static variables represented by the territory portion covered by cells. This has a strong impact especially for vegetation, being that Mediterranean regions have a great variety of different coexisting species as also a great importance in determining the fire risk estimation.
RISICO provides its computation output as a set of maps representing predicted behaviour in space and time for those variables which are useful in identifying risky situations. At present these variables indicates potential fire risk, i.e. the danger level of a fire, without considering probability of it to begin.

Output data produced for the whole Italian territory takes about a daily amount of 100MB, with an overall 40MB of needed input data, when using 1km cells. Needed computation time is in the order of 40 minutes using a common workstation.

2. Porting work description

Our goal.

As we said already, a good quality estimation strongly depends on properly sized cell. While the 1km cell size is a good one in terms of needed computation time and storage resources, running the model with 100m sized cells would provide much better estimations, but this would lead to increase a hundred of times both computation time and needed storage space.
Our goal is to run the RISICO model on GRID in order to achieve in a reduced time the results for simulation on 100m sized cells and manage produced output in an efficient way. The "reduced response time" is a quite important requirement for Civil Protection operators in order for them to be effective.



Grid porting strategy.

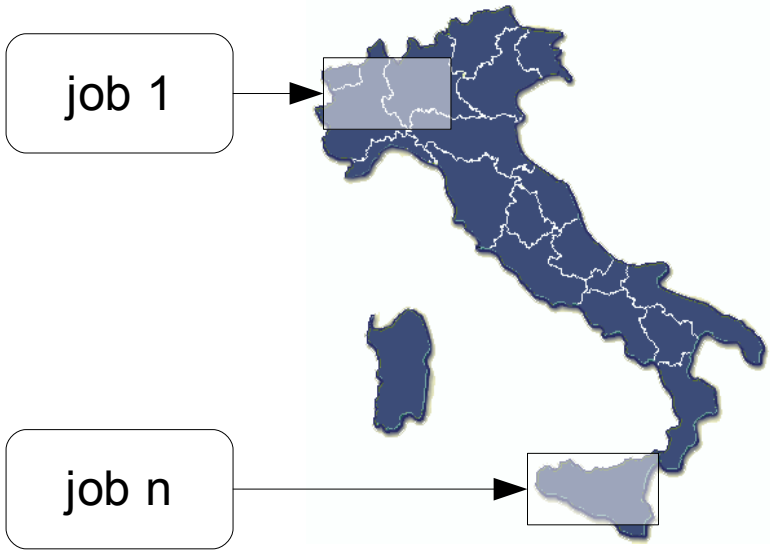
Thanks to the intrinsic parallel nature of RISICO, our first considered approach has been that of adopting a "divide et impera" strategy, which basically consists on submitting *N* risico instances, each one of them working on subsets of cells obtained by partitioning the whole set. A proper range choice for the *N* value permits more satisfying results. As a rule of thumb, too high values may increase queue times effects or the probability that a (single) job encounters a problem (i.e. it gets queued on a busy site with high wait time), while too little values obviously tend to reduce benefits of distributed computation. Experiments with *N* in the range 10, 30 has shown satisfying results.

Description:

Global RISICO output is obtained as result of many jobs, each one computing a given portion of the whole domain.
Jobs are identical and indepentent.

Every job makes use of a set of files which are available into one or more Storage Element (SE) and identified through their Logical File Name. These files are copied into the *closest SE* by a python script running on the User Interface. They are:

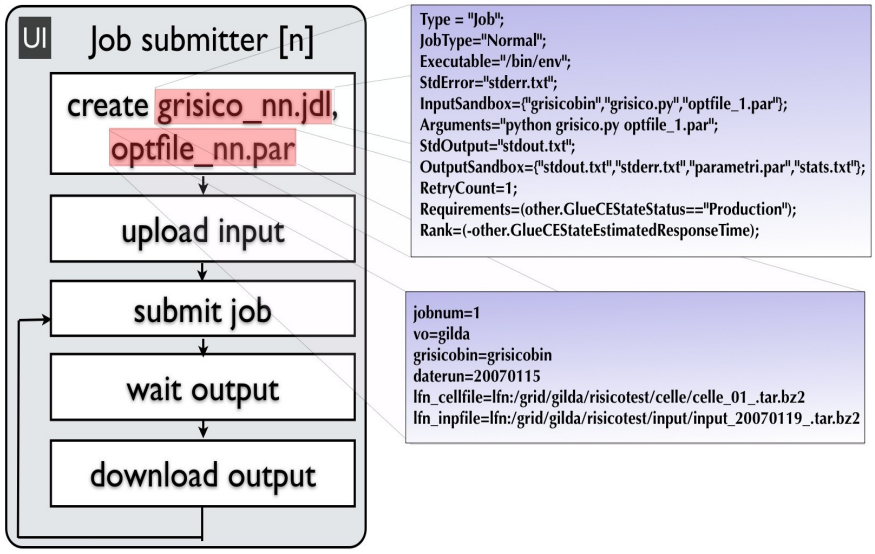
- *celle_**nn**.tar*.bz2: defines computation domain for the job numbered *nn* with static informations (cells description).
- *input_data.tar.bz2*: contains informations related to input variables (meteo data, dynamic). This file is common to all jobs;
- *output_**nn**_data.tar.bz2*: this is the output file produced by the job numbered *nn;*
- *stato0_nn.tar.bz2*: status informations for the job numbered *nn.* After computation, a new status file is produced, which will replace the old one and will be used in the next RISICO run.



A fundamental piece for the Grid-ified RISICO system is the gJobSubmitter module, which takes care of setting up all needed thing for jobs to run, basing on preferences specified in a configuration file. This module, when launched, uploads files into the Storage Elements, creates a jdl file for each job and submits the jobs itself. Then it follows their execution until it terminates, retrieving produced output for each succesfully terminated job.
Two methods are used by the gJobSubmitter module to keep track of the jobs status: a “direct control”, by regularly polling the Logging & Bookkeping server (glite-wms-job-status) and an alternative one, which consists in evaluating the presence on the SE for the expected output file (which LFN is known in advance). Quite often this permits to detect job's completion and retrieve its output file before it is “officially” sentenced to be finished by the LB server, and this saves a latency time who can quite commonly be in the order of a five to ten minutes.
Also with the intent of reducing latency times, it is optionally possible to specify in the configuration file one or more queues where jobs are to be directly submitted. This may permit to skip the MatchMaking phase in the Workload Management System. This can be useful to have jobs submitted on queues which are a-priori

known to be quickly available and offer a low latency service. It may happen that despite having specifyed a “lowest response time” as a requirement in the jdl file, the job is still delivered on a busy site or in a temporarily unefficient one.
In order to reduce the impact of this kind of problems it is possible to resubmit a job which is still waiting after another one has finished already to the same queue used by the succesfull one, which has proven to be effective.

GjobSubmitter Module:



Operations accomplished by the GjobSubmitter module are:

- Copy input files on a close SE using a *lcg-cr* call*;*
  Example: ***lcg-cr --vo cyclops file:/home/user8/grisico/input.tar.bz2 -l lfn:/grid/risico/input/input_20071127_.tar.bz2 -d prod-se-02.pd.infn.it***
- Creation and submission of *n* independent jobs;
  Example: ***glite-wms-job-submit -a -o jid_20071127.txt grisico_1.jdl***
- Instantiation of *n* control processes *jobSubmitter* and polls job termination.
- Output file retrieval for completed jobs.

Final considerations on the adopted submission strategy.
In Civil Protection context a “reduced response latency” is to be preferred respect to a “high productivity in the medium average period”. This is why the chosen submission strategy may optionally skip non strictly needed phases, such as MatchMaking in the WMS and “official” job status check from the LB point of view. A further potential advantage is that the number of “point of failure” gets reduced: if a component is in a critical condition, the ability to be robust respect to his undesirable behavior may be important. Practical performed tests have effectively proven the ability to retrieve job output minutes before LB declared it has done. Also, the job submitter module can decide to resubmit a still waiting job on a provenly effective queue.

3. Personal comments

This “application porting school” has given us a great oppportunity to learn about GRID architecture and to practice with its available instruments so to test what useful potential it offers for civil protection issues.