Grid computing for wildfire danger assessment: porting RISICO in glite

Stefano Dal Pra (INFN)
Mirko D'Andrea(CIMA)

EGEE User Forum
February 11, 2007
Clermont-Ferrand, France

Co-Authors
M.Verlato (INFN),
F.Gaetani, P. Fiorucci (CIMA)
V. Angelini (CNR-IMAA)

Layout

- RISICO as CYCLOPS use case.
- RISICO outline
- Grid porting
- Submission strategy
- Results
- Future plans

CYCLOPS and RISICO

CYCLOPS:

- European project. Main goal is "to bridge the gap between Grid and Civil Protection communities"

RISICO:

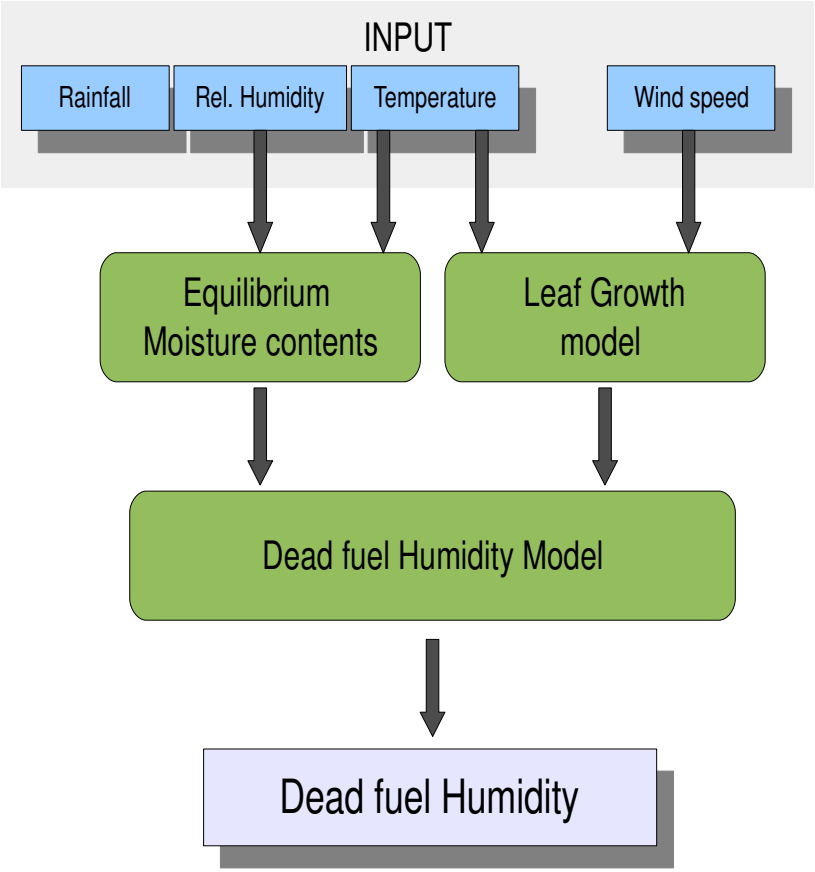
- RISICO (RISchio Incendi e Coordinamento) is a Fire Risk prediction software developed by CIMA for Italian Civil Protection Agency
- Selected as Use Case in Cyclops
- Goal: Run with higher resolution and lower latency

About RISICO

- developed by the CIMA Foundation for the Italian Civil Protection Agency
  - Authors: P. Fiorucci, P. Gaetani, F. Manciardi
- Goal: computing detailed wildland fire risk forecasts relevant to the whole national territory.
- two main modules,
  - fuel moisture model
  - potential fire spread model.
- Static and dynamic input data:
  - topographic and vegetation cover data (GIS)
  - Dynamic: meteorological forecast

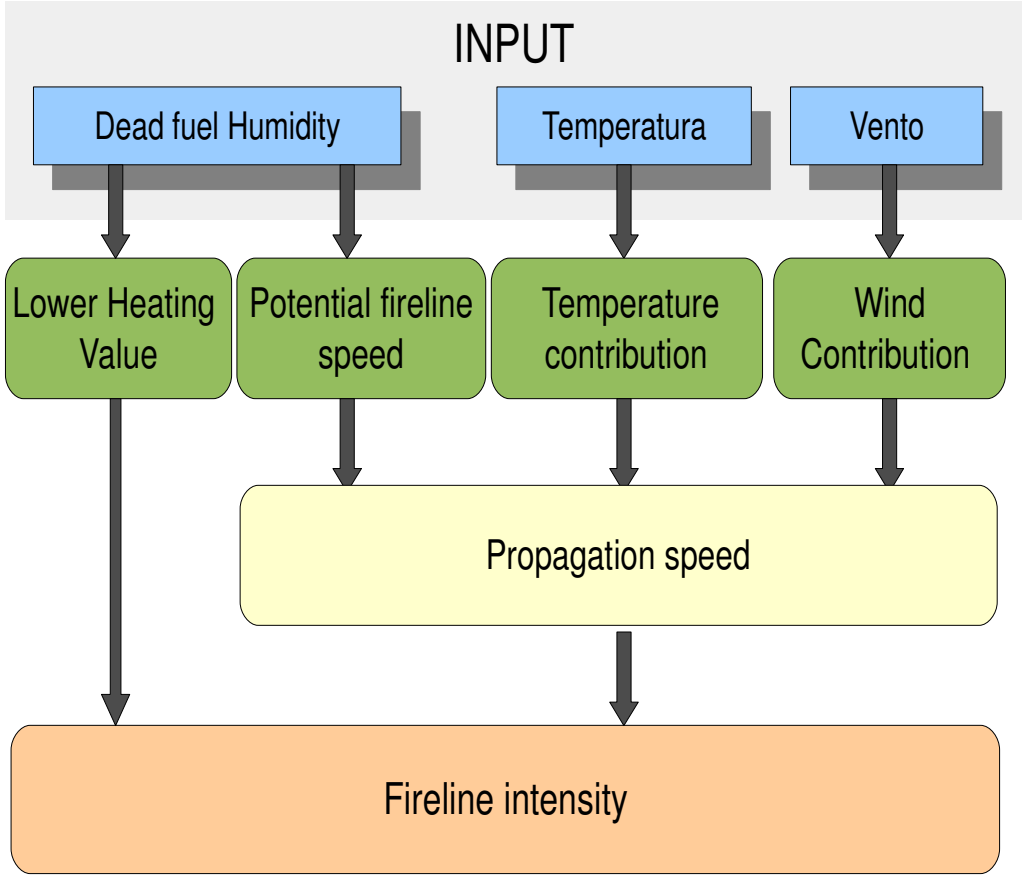
Fuel moisture model

- fire ignition risk depends on dead fuel status
- First order model
- Meteorological input used



Fire spread model

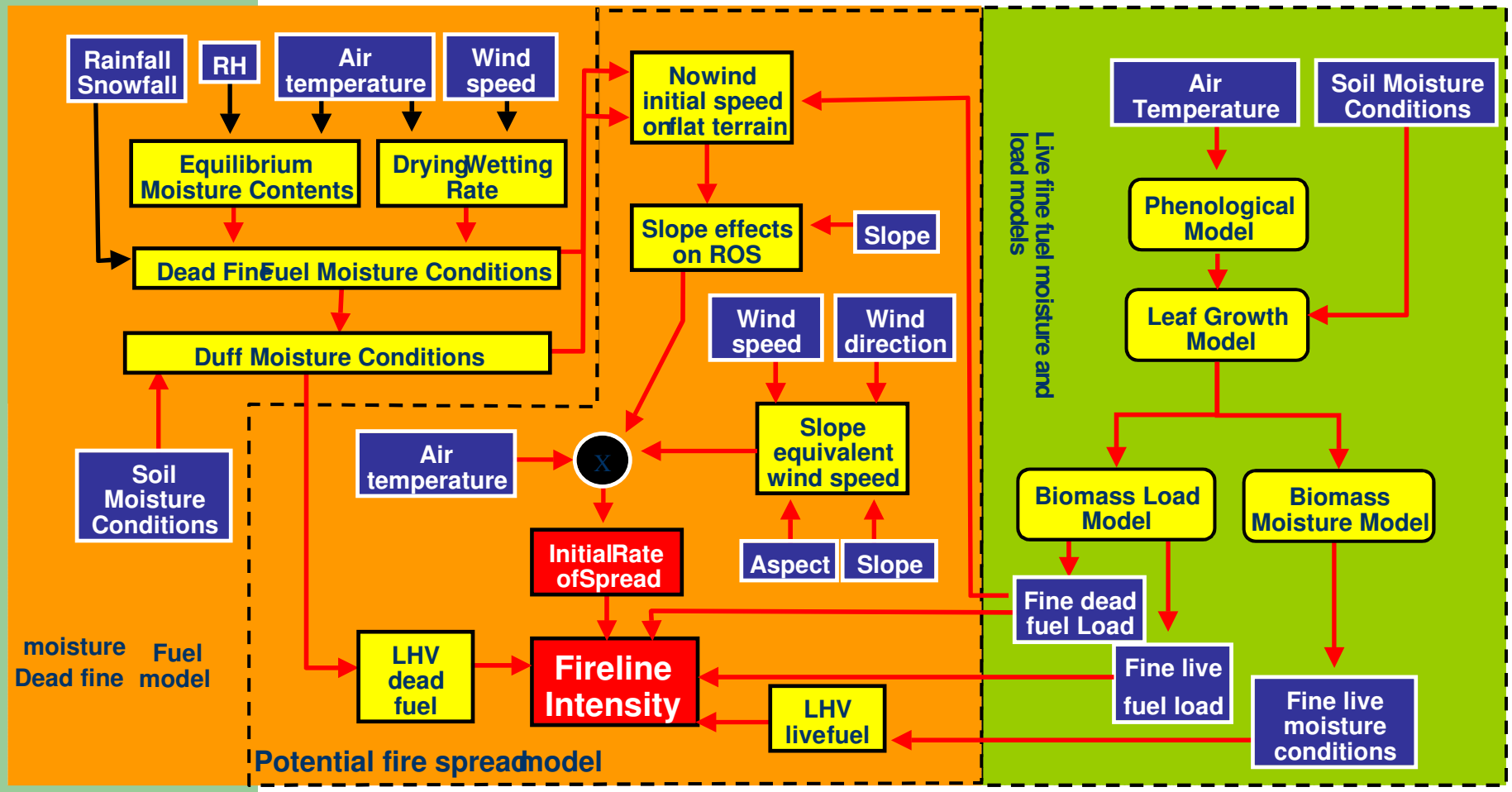
- Combines output of previous model with meteo data
- Provides estimation of potential risk in case of ignition



Which data RISICO uses

- Atmospheric:
  - Air Temperature and humidity,
  - Wind speed components,
  - Rainfall;
- LAM (Local Area Model), meteorological forecasting over 72h, used for risk prediction.
- Interpolated Maps of earth sensors providing data used to initialize the system

RISICO computational model



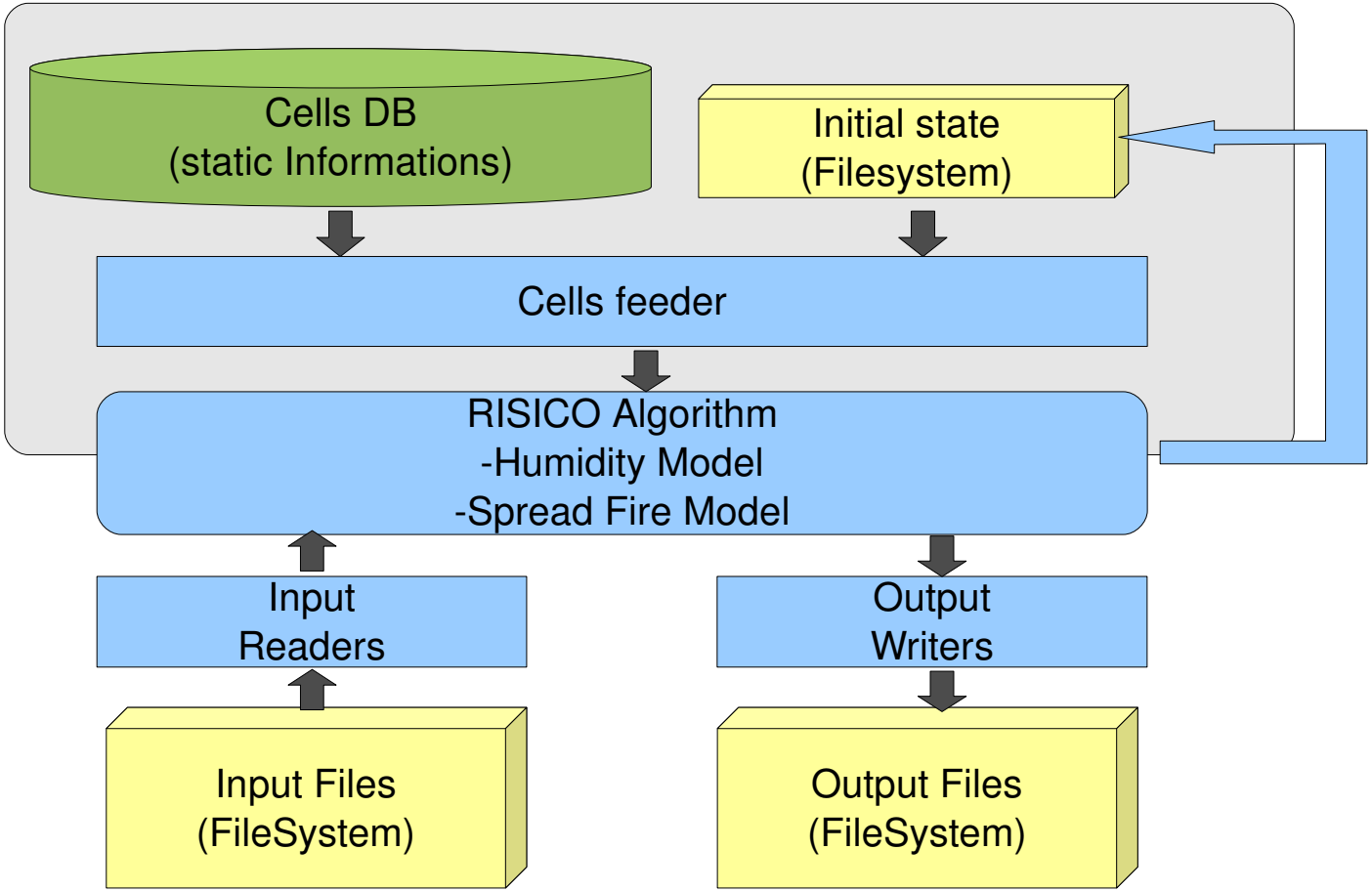
RISICO Algorithm

- Input files
  - X0 = stato0 (initial status)
  - C = set of cells (static dataset)
  - I = input (dynamic; meteorological data)
- Transformation
  - T = The RISICO executable
- Output files
  - Ou = output (Humidity and firerisk prediction maps)
  - (X1,Ou) := T(X0,C,I);
    - X0 := X1 for next execution
- T property
  - Each cell computed independently from others
    - Easy to split for distributed computing

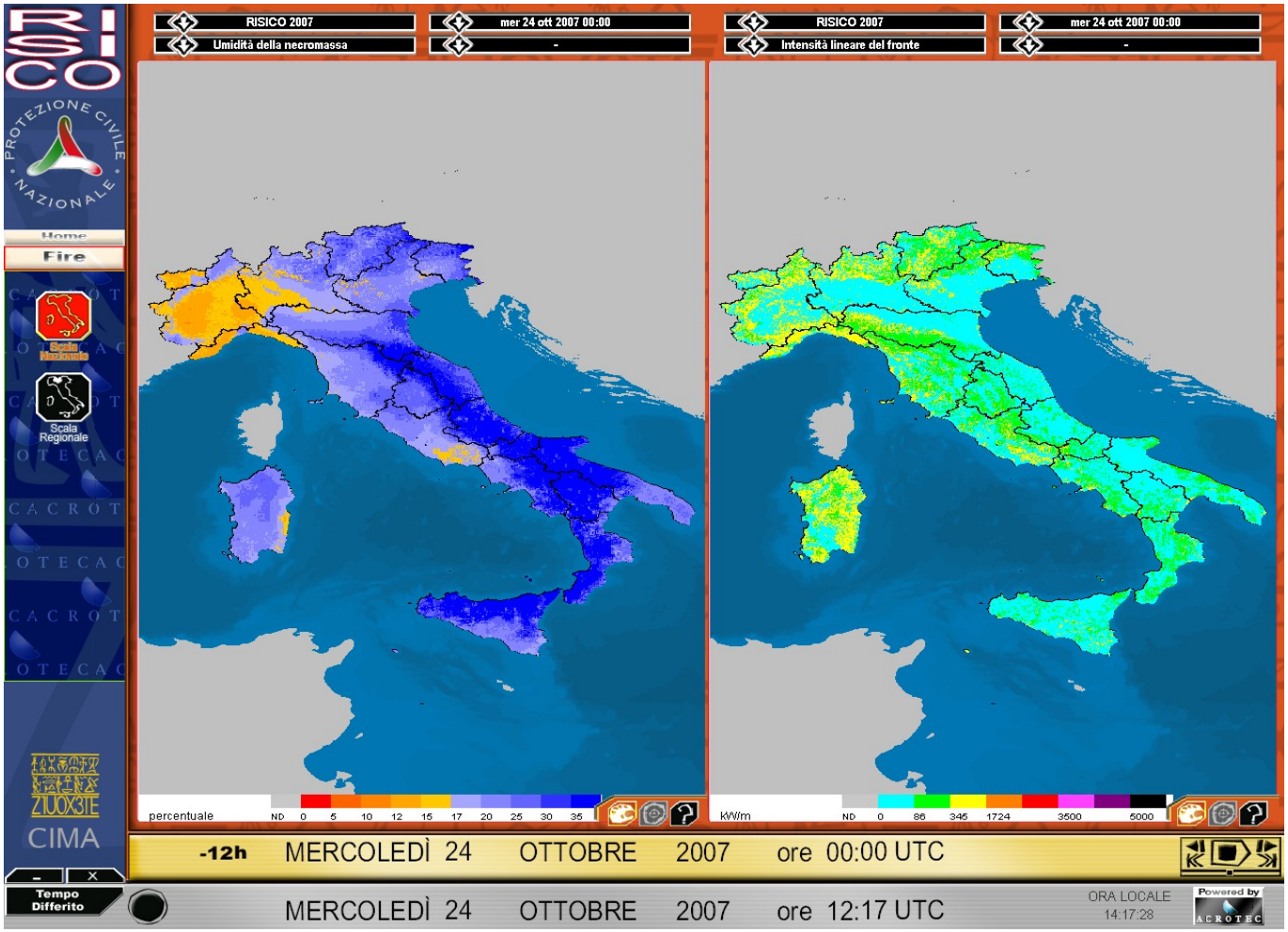
RISICO Computational domain

- Geographical domain is discretized into equally sized cells at 0.01 degrees (1 km^2) .
- 330000 cells are needed to cover Italy.
- Every cell is represented by:
  - Center coordinates
  - Vegetation parameters (dead fuel),
  - Orographic description,
  - Pointers into data in Input file I;
- 150MB input data

RISICO Functional scheme

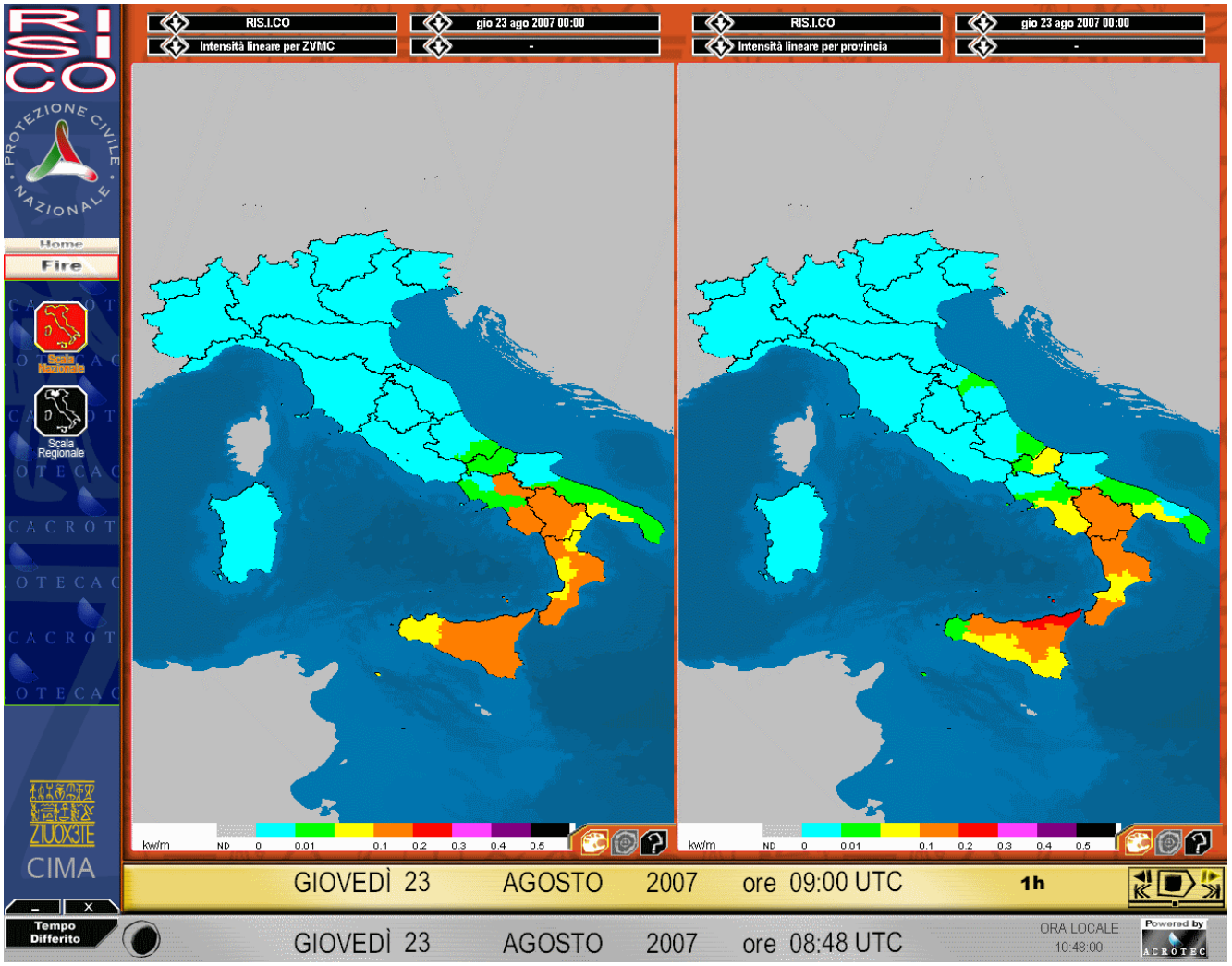


RISICO
Final output on graphical interface (Oct. 2007)



- Humidity and firerisk prediction maps

RISICO
Final output on graphical interface (Aug. 2007)



- Humidity and firerisk prediction maps

Porting RISICO to Grid: motivation

- A tipical RISICO run takes 20 to 30 mins for Italian territory with 1km cells On a common workstation.
  - Complex topography with heterogenous vegetation needs higher resolution (100m sized cells)
  - This increases of two magnitude orders both dataset size and needed computation time.
- Initial Goal: Simulation over Italy with 100m sized cells.
- Constraint: Maximum execution time < 1h.
- Desired: getting Output as quickly as possible.
  - Speed over completness

Porting RISICO to Grid: considerations

- RISICO is written in C++ with no need for nonstandard libraries
  - Straightforward compilation on Linux (RHEL, SL4)
- Computational domain can be conveniently splitted into *N* nonoverlapping slices
  - This means executing *N* distinct jobs.
- Quite large I/O (must use Storage Elements)
- Be Robust, when possible
  - Prevent or handle undesired events (failing or “sick” components, etc).
    - Submission strategy, Execution monitor

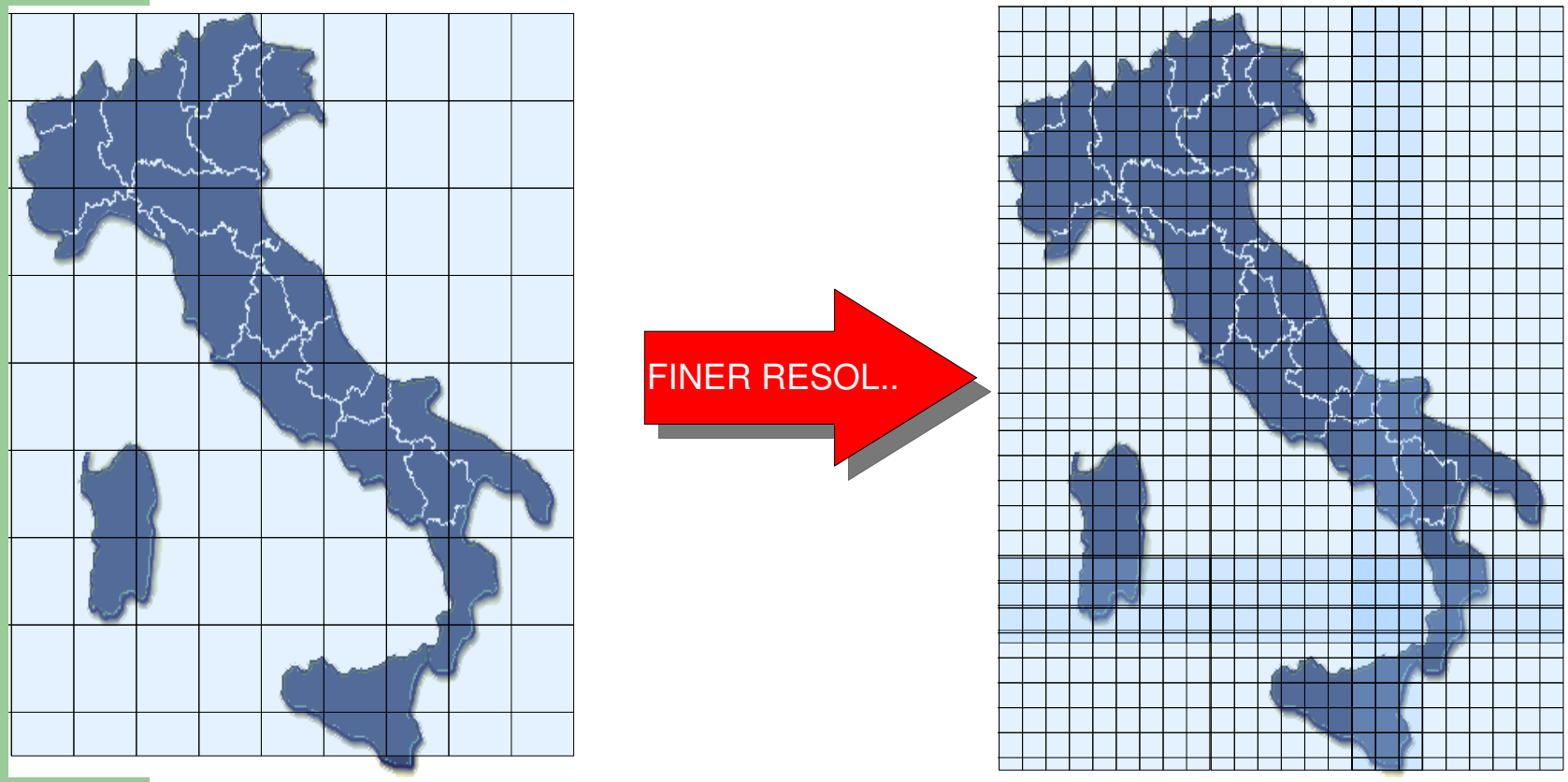
G-RISICO: The porting

OUTLINE

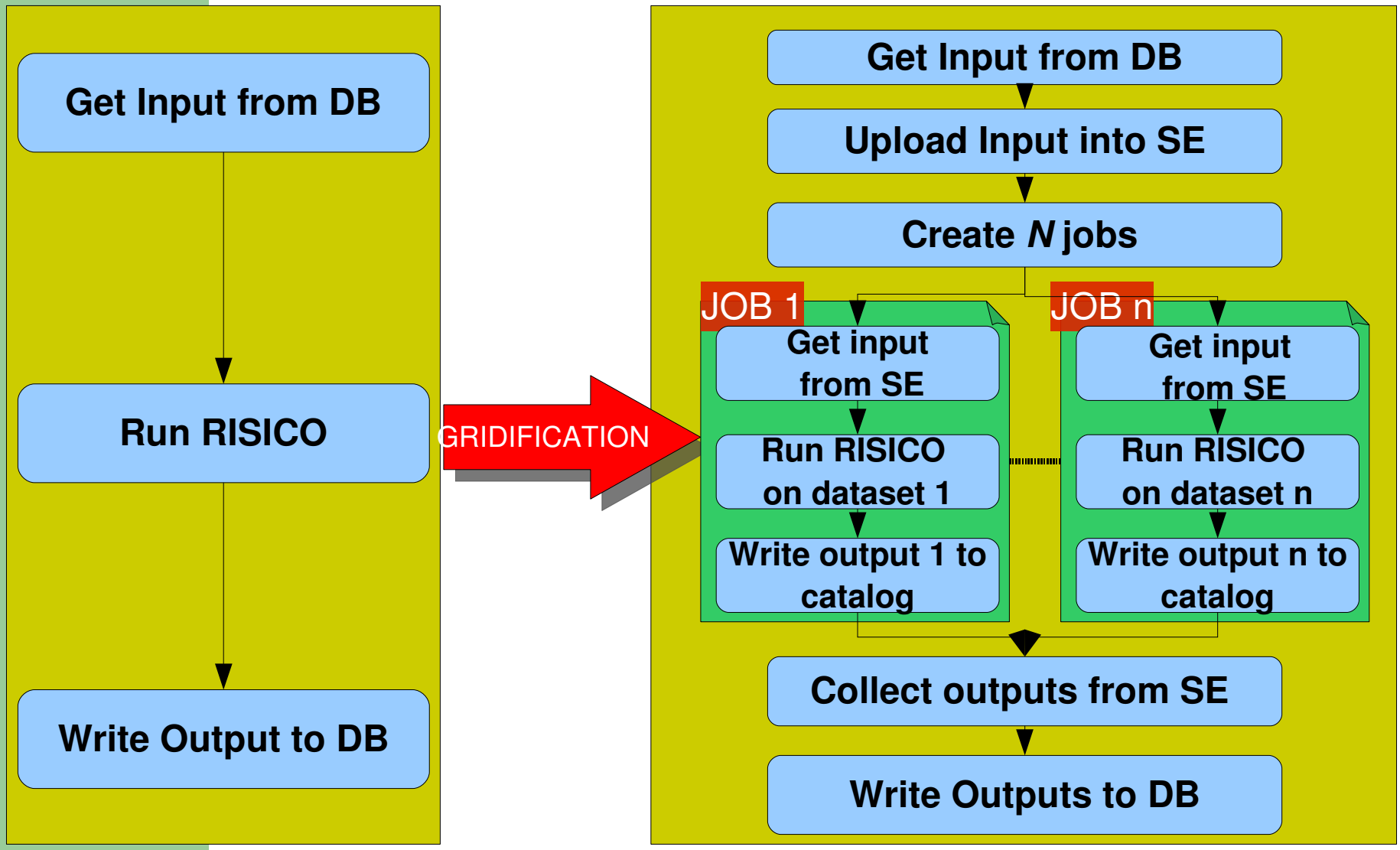
- Porting features;
- Jobs management;
- Implementation tests and results;
- Conclusions and further development.

G-RISICO: The porting

- Italy: 310000 km^2
- Current system: 300k regular cells, 1km side.
- Grid version: 30M regular cells, 0.1km side.



RISICO vs. GRID-RISICO



Job submission

- A RISICO's job is fully defined by a jdl (job description language) file and by a parameter file.
- Submitted jobs must end within a defined maximum time. Job activity is monitored at UI level by a “JobMonitor” sw module.
- The job submission procedure is handled by a JobSubmitter, which creates *N* jobs.

Job Monitor

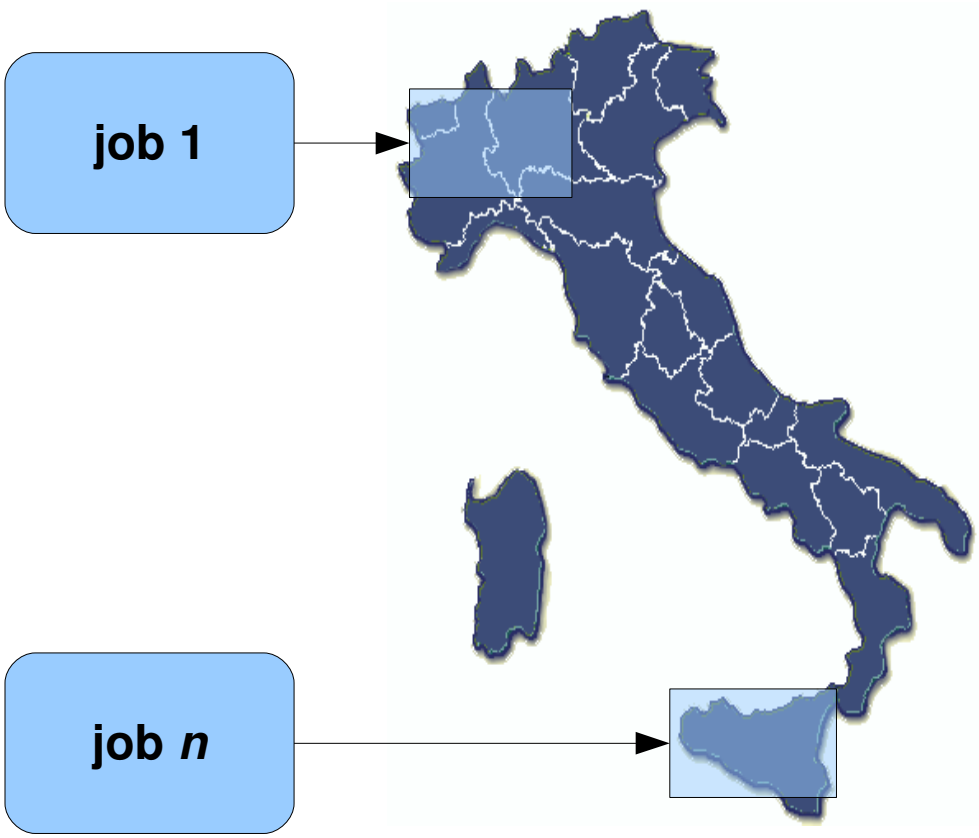
- All the jobs are monitored by an instance of a module called JobMonitor.
- The JobMonitor:
  - Checks the job status during execution;
    (It checks output file existence)
  - Retrieves the job output from catalog;
  - If the job fails, JobMonitor tries to resubmit it.
  - JobMonitor will log the error if the job fails to run correctly.

Workflow

- Retrieve input files from remote meteo-data database, creates an archive and uploads it to Storage Element.
- Creates a jdl and parameters file for each job;
- Submits the jobs.
- Wait for jobs output. (polls for LFN existence)
- Retrieve job output from catalog and aggregates them.

Job Definition (1)

- Each job works with a specific dataset defining a spatial domain (subset).
- Such subsets are created off-line and stored on the catalog.
- A parameters file states the association between a job and a dataset.
- Each job produces an output, whose path in the catalog is a-priori known.



Job definition (2)

Job 1:
- *Domain*: **celle/celle_01.tar.bz2**
- *Status*: **celle/stato0_01.tar.bz2**
- *Input*: **input/input_20070119.tar.bz2**
- *Output*: **output/output_01_20071119.tar.bz2**

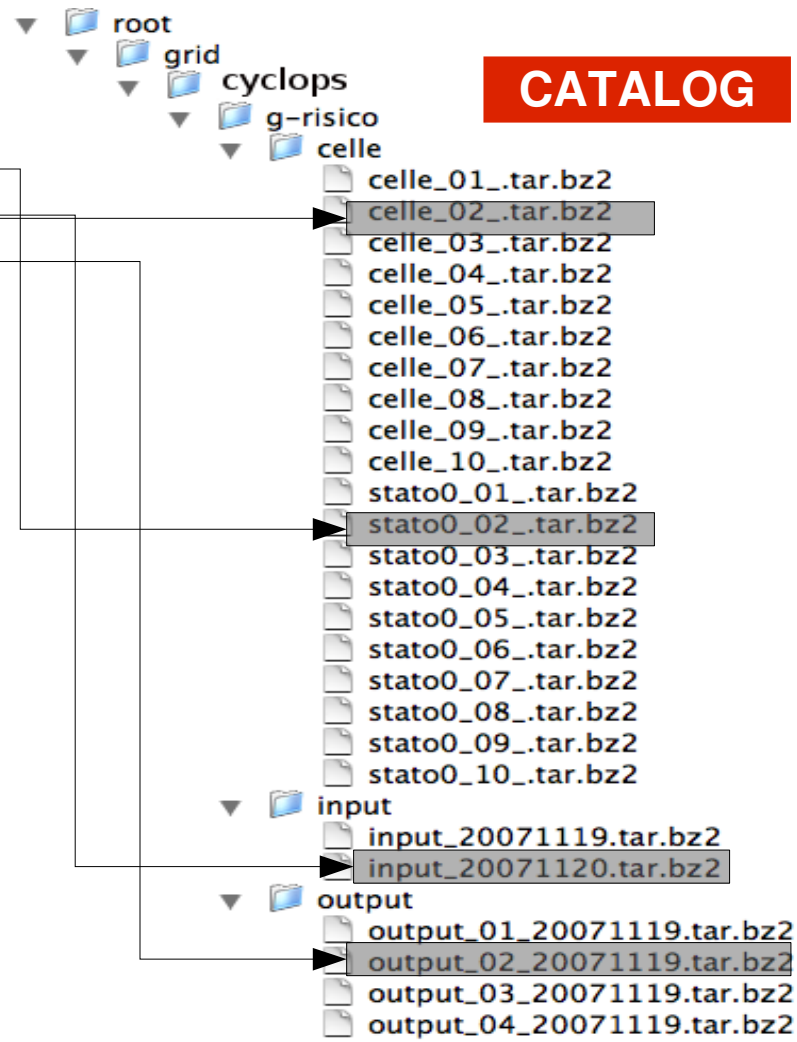
- Each job has its own domain.
- Job domain, status information and output are referred to the same geographical domain
- All jobs share the same input file.

Job and catalog

Job 1:
- *Domain*: **celle/celle_01.tar.bz2**
- *Status*: **celle/stato0_01.tar.bz2**
- *Input*: **input/input_20070119.tar.bz2**
- *Output*: **output/output_01_20071119.tar.bz2**

Job 2:
- *Domain*: **celle/celle_02.tar.bz2**
- *Status*: **celle/stato0_02.tar.bz2**
- *Input*: **input/input_20070119.tar.bz2**
- *Output*: **output/output_02_20071119.tar.bz2**

Job n:
- *Domain*: **celle/celle_nn.tar.bz2**
- *Status*: **celle/stato0_nn.tar.bz2**
- *Input*: **input/input_20070119.tar.bz2**
- *Output*: **output/output_nn_20071119.tar.bz2**



Job wrapping and splitting

- InputSandbox
  - {"grisicobin","grisico.py","optfile_*nn*.par"};
    - Python wraps real executable
- OutputSandbox
  - {"stdout.txt","stderr.txt","stats.txt"};
    - Timestamps about status execution advances are returned in stats.txt
    - These info are useful to tune the domain splitting
- Tests have been performed with 10 to 90 jobs, 100.000 cells each.
  - 10 jobs completes (output retrieved) in 7..15min after sequentially submitting them
  - 90 jobs completes in 40 to 60 min

Job execution times (WN , 90 jobs)

job_1 started at: Fri Feb 8 16:34:40 2008
job_90 started at: Fri Feb 8 17:00:25 2008
Max TT=170, Min TT=43, Avg TT=95

['grid001.roma2.infn.it', 21, 27, 51, 99], ...
['pccms34.ba.infn.it', 27, 7, 27, 61], ...
['prod-wn-001.pn.pd.infn.it', 12, 30, 41, 84], ...

- Download,unzip,untar
- Actual computation
- tar,zip,Upload
- Total Time
  - jobs sequentially submitted, 10^5 cells each
  - CPU Time is 25% to 50%

Job execution times (WN , 10 jobs)

job_1 started at: Thu Feb 7 17:49:46 2008
job_10 started at: Thu Feb 7 17:59:15 2008
Max TT=98, Min TT=55, Avg TT=78

['grid006.roma2.infn.it',	27,	24,	40,	91]
['pccms40.ba.infn.it',	34,	8,	33,	76]
['prod-wn-022.pn.pd.infn.it',	12,	21,	34,	68]

- Download,unzip,untar
- Actual computation
- tar,zip,Upload
- Total Time

Comments, ideas

- Submit fewer jobs with more cells
  - An optimal trade off between jobs num and cells per job can be determined.
- Distribute “near” cells among many jobs
  - Olistic effect: every job returns info about every portion of the overall domain
  - First retrieved output already provides info at a glance
- Let the job directly inform us he has done
  - WN can connect to external http services
    - Idea: Have at UI level a custom http listener. This can inform the Jobmonitor module about job progresses
    - CAVEAT: net security must be considered!

RISICO-Grid testbed: The CYCLOPS VO

Central VO services:

LFC: lfcserver.cnaf.infn.it

VOMS: voms2.cnaf.infn.it

WMS: prod-wms-01.pd.infn.it, glite-rb-00.cnaf.infn.it

Computing and Storage resources:

#CPU	Free	Total Jobs	Running	Waiting	ComputingElement
4	1	30	0	30	gridce.ilc.cnr.it:2119/jobmanager-lcgpbs-grid
70	1	40	1	39	gridba2.ba.infn.it:2119/jobmanager-lcgpbs-short
70	1	1762	34	1728	gridba2.ba.infn.it:2119/jobmanager-lcgpbs-long
70	1	2307	11	2296	gridba2.ba.infn.it:2119/jobmanager-lcgpbs-infinite
44	14	8	6	2	grid0.fe.infn.it:2119/jobmanager-lcgpbs-grid
44	0	36	15	21	prod-ce-01.pd.infn.it:2119/jobmanager-lcglsf-grid
539	4	30	4	26	gridce2.pi.infn.it:2119/jobmanager-lcglsf-grid4
25	9	15	12	3	grid001.ts.infn.it:2119/jobmanager-lcglsf-grid
12	2	14	10	4	gridce.sns.it:2119/jobmanager-lcgpbs-grid

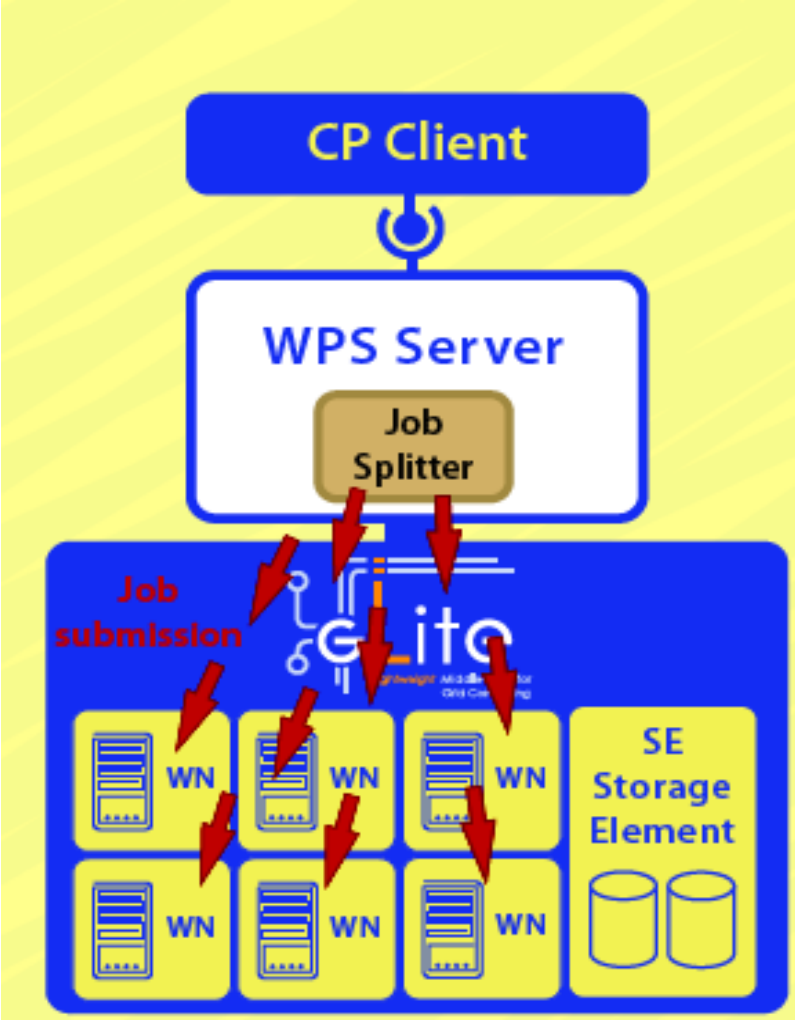
738 total CPU-cores (not dedicated)

Avail Space(Kb)	Used Space(Kb)	Type	SEs
395156380	4970248	n.a	gridse.ilc.cnr.it
445299512	1364038152	n.a	grid007g.cnaf.infn.it
3912509084	481669476	n.a	prod-se-01.pd.infn.it
52670212	656381412	n.a	gridit002.pd.infn.it
3840000000	250000000	n.a	prod-se-02.pd.infn.it
2322948608	19715584	n.a	grid002.ts.infn.it
67267572	3758208	n.a	gridse.sns.it

> 10 TB (not dedicated)

Next steps

Implement GRISICO as Open Geospatial Consortium
Web Processing Service
(See Poster session)



Conclusions

- RISICO-Grid can be effective in Civil Protection scenarios Fire Risk Assessment
- Improvements for current implementation
  - Parallel submission
  - Parallel output retrieval
  - sparse cells distribution between jobs
  - Event driven output retrieval (instead of polling).
- Integration with OGC WPS

The End

Thanks all

- Next event: CYCLOPS project conference jointly with IBERGRID 2008, with a live demo of RISICO through WPS.
- See http://www.ibergrid.eu/2008/