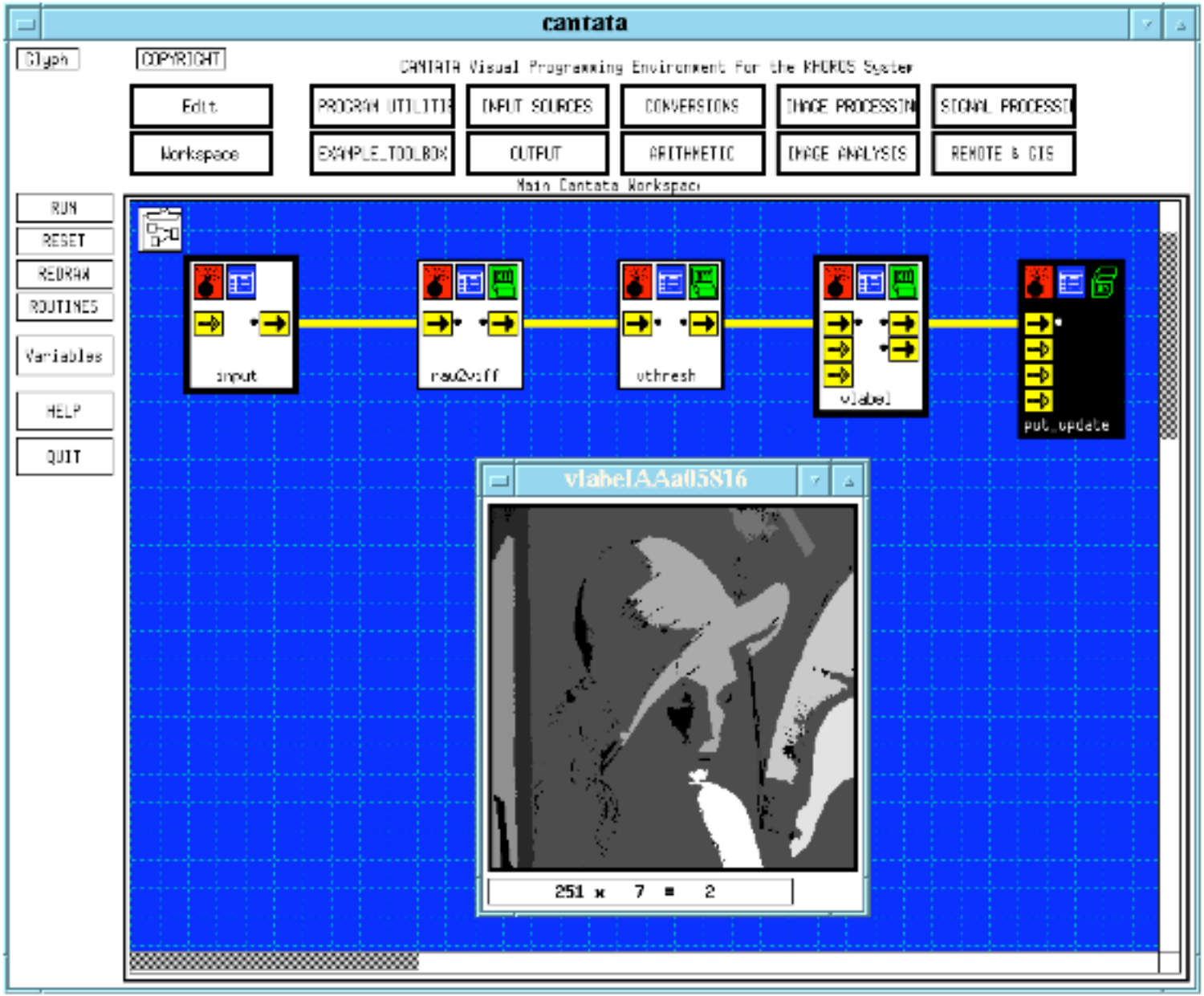
Lessons learnt in distant files

- Examples of MiddleWares and MiddleForces for WorkFlow
  1. One ancestor: Khoros @NMU -a tool for learning image processing, was used at INRIA (project PASTIS and others). C++ Toolkit was complex.
  2. CAST in Decision - INRIA-led project about optimization for genetic etc. algos on clusters (project OPALE, born from former project SINUS+)
     http://www.inria.fr/rapportsactivite/RA2004/opale/uid22.html
  3. QSO Tool at CFHT for LS - very simple but robust operational workflow: handles loops, but branching validated at runtime by SOs. Immediate human interaction available AND needed for exception handling.
- Issues
  - Design issue: Size of vision and granularity (ex: Astrogrid Web App now going to a WorkBench because of req changes)
  - Horizontal Integration Issue: with Queue Scheduling systems for access to data and computing resources (ex: clusters, DBs, telescopes+). “la file d’attente peut-elle casser l’apport de la parallelisation ?”
  - Vertical integration - existing systems and pipelines, their own level of dev
  - **Human issue** = convergence of traditionnaly closed communities (ex: F95 specialists doing SQL92). Education and interfaces (ex: SP): Middle(Work)Force accompagning MiddleWare
  - Visualization is the key for both process and human communication


cantata
Glyph
COPYRIGHT
CANTATA Visual Programming Environment For the KHOROS System
Edit
Workspace
PROGRAM UTILITIES
EXAMPLE_TOOLBOX
INPUT SOURCES
OUTPUT
CONVERSIONS
ARITHMETIC
IMAGE PROCESSING
IMAGE ANALYSIS
SIGNAL PROCESSING
REMOTE & GIS
Main Cantata Workspace
RUN
RESET
REDRAW
ROUTINES
Variables
HELP
QUIT
input
vthresh
vlabel
vlabelAAa05816
251 x 7 = 2

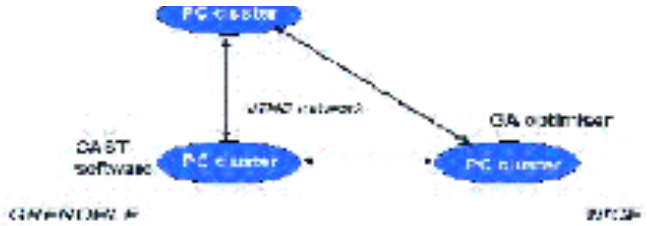


Figure 1. PC Cluster over the VTHD network

The main drawback still remains however in the deployment and control of complex distributed applications on grids by the end-users. Indeed, the deployement of the computing grid infrastructures and of the applications in such environments still requires specific expertise by computer science specialists. However, the users, which are experts in their particular application fields, e.g. aerodynamics, are not necessarily experts in distributed and grid computing. Being accustomed to internet browsers, they want similar interfaces to interact with grid computing and problem-solving environments. A first approach to solve this problem is to define component-based infrastructures, e.g. the Corba Component Model, where the applications are considered as connection networks including various application codes. The advantage is here to implement a uniform approach for both the underlying infrastructure and the application modules. Still however, it requires specific expertise not directly related to the application domains of each particular user. A second approach is to make use of grid services, defined as application and support procedures to standardise access and invocation to remote support and application codes. This is usually considered as an extension of Web services to grid infrastructures. A new approach, which is currently being explored by the OPALE project, is the design of a virtual computing environment able to hide the underlying grid-computing infrastructures to the users. An international collaborative project has been set up in 2003 on this subject involving the OPALE project at INRIA in cooperation with CNES and involving also EADS (Common Research Center) and Datamat (Italy).

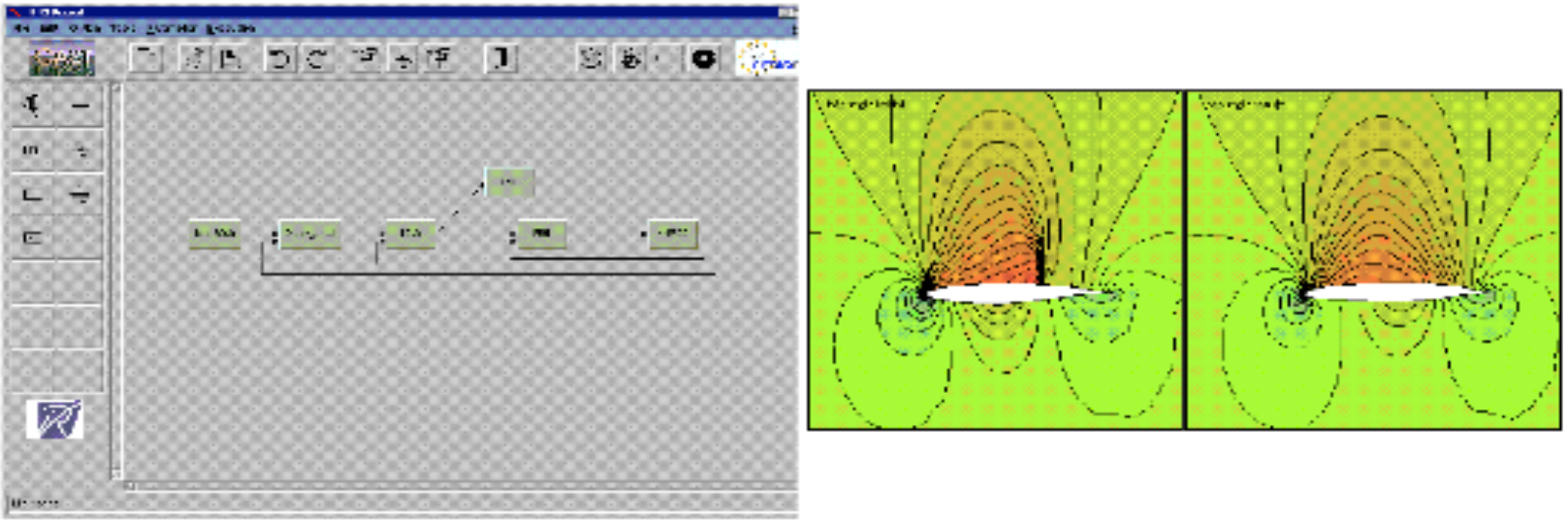
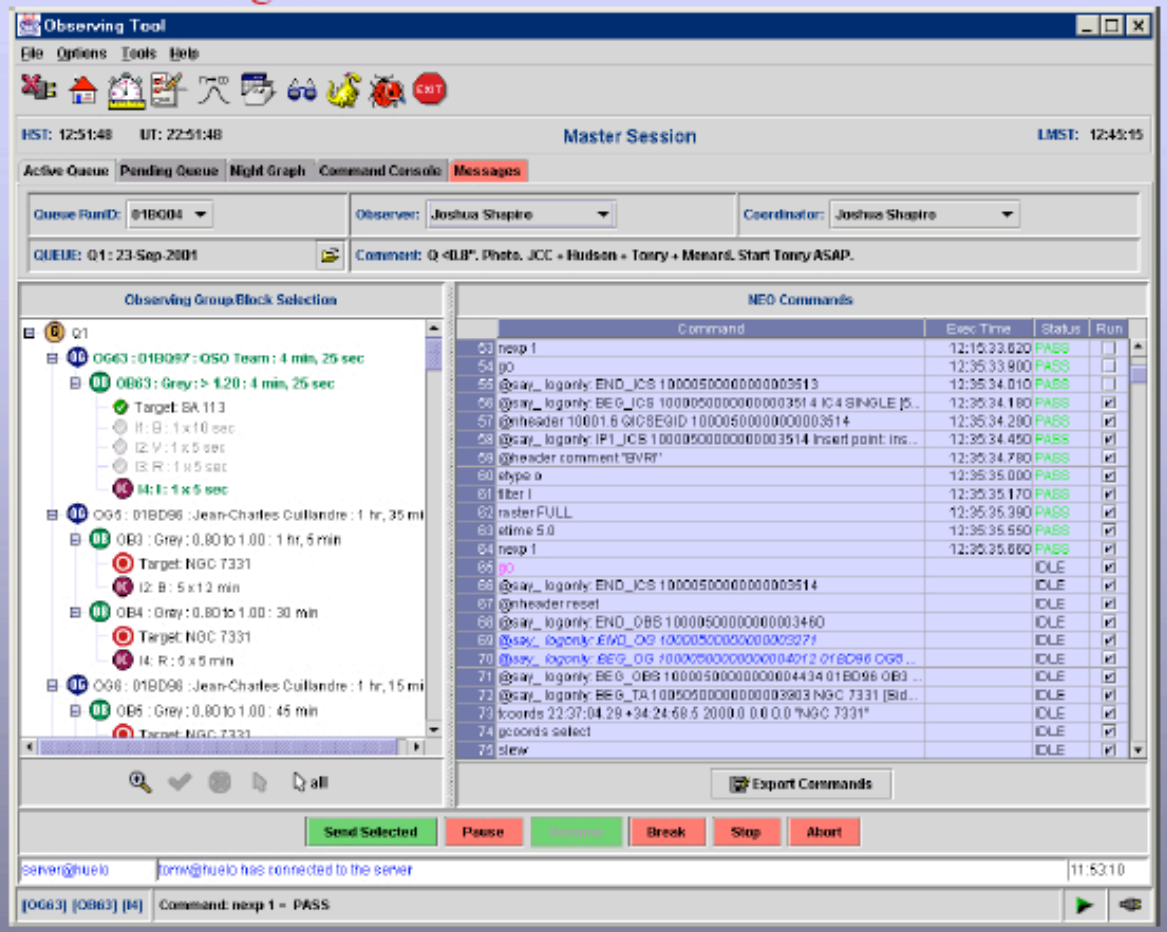


Figure 2. Hybrid Gradient-Genetic Optimization using the demonstration platform developed in the European Project DECISION; note the important

QSO Tools: A Brief Overview…

- Observing Tool



- Used by Observer during night
- Translate Phase 2 conceptual language into I+T command language
- Observe pre-built Qs AND on-demand RT calibration (FOCUS)
- Enforce program (vs. classical !) but flexible (iterations:repeats)

Workflow

Workflow: A middleware which gathers within a common (referentiel) the execution context of tasks necessary for accomplishing an action.

Costs and Risks

1. Integration of every step in the framework => open API vs. obscure codes
2. First a-posteriori approach (existing code) => learning curve for the GLUE
3. But we already have started via Astro WS and VOT

Benefits

1. Parallelization
2. Planning => end-to-end vision
3. Encapsulation => common unit testing environment
4. Centralization => WF engine => DB-centered for I/O/Temp params+state
5. Visualization of complex processes => helps transfer (part. large TO envs)
6. Piloting applications => Intelligent scheduling