NA3: Tips for SEE Trainers

Fotis Georgatos
GRNET

Training the Trainers
December 1st, 2006
Sofia, Bulgaria

Overview

- Trainers: Prerequisites
- Trainings: Organization
- Trainings: Practical(ities)
- Trainings: Suggestions
- Presentations: Exemplar material
- Technical Tips you should know

Trainers - Prerequisites

- You need to understand the system, in order to be able to teach about it
- But, you don't need to know *all* the details: The only real requirement is a robust knowledge of a skill-set which is enough to kick-start with.
- Read the gLite User Guide's (v3.0?) index every now and then. You should at least be able to tell people: "you can find the answer in the User Guide".
- Treat trainees as colleagues (~~they~~ we are, after all!)

Trainings - Organization

- **A good training involves both theory and practice; people need to:**
  - familiarize themselves with new concepts (theory)
  - have practical skills for their day-to-day needs (practice)
- **Plan early, work in advance. Most of the work for a training to happen, occurs before the event: checklist! Once that is done, the training goes smoothly**
- **Try to bring as much "grid innovation" as possible, during the Trainings, eg. by giving A4 hand-outs of ongoing projects. There are several reasons:**
  - dissemination of Grid capabilities and inventions
  - facilitate users in their quest to find solutions within the Grid
  - the impression of an active community
  - spirit of openness & encouragement of publishing results

Trainings - Practical(ities)

- **Hands-On (laboratory type of activities) are particularly attractive for the part of the audience which is most likely to develop applications on the grid. Include as many as possible, still, leave space for "wanderers".**
- **Pair-based practicals are a well-known tool for people who study Educational Sciences... have a 2nd projector!**
- **Stability of t-Infrastructure extremely critical, because it has to work, real-time and grid is a complex machine:**
  - A real-time activity is not an easy thing to do
  - people get a perception of grid stability, based on a single event's experience... you just entered danger zone :-)
  - for you it might be just another event, for them "this is the grid".
  - make multiple tests of infrastructure, early-on. Job submission and Data Management tests are absolutely essential.
  - It is customary to test follow-up practicals during breaks, in order to avoid having to change the training on-the-fly

Trainings: Suggestions

- **Don't hesitate to ask people questions and promote them firmly to do so..., check cultural aspect, too.**
- **If you have to do some waiting for the network to be fixed, processes to finish, etc, perform genuine people management: This is a great opportunity to ask people to introduce themselves, to explain their interests and expectations etc. Provide “general interest” as answer**
- **Consider in advance a "Plan B", for *any* kind of failure you can thing of, no matter what happens:**
  - t-Infrastructure failure, (could you train them from your account?)
  - room facility failure (eg. projector/air-conditioning problems, etc)
- **or at least be ready to "degrade gracefully" ;-)**
- **Things *will* go wrong, sooner or later, on one event or another... can you manage?**

Presentations: Exemplar material

- **The exemplar material is very useful resource, because it can serve both as a standard body of material to copy from, and as a point of reference for comparison.**
- **It is in English. You could translate eg. the Sofia 2006 training, and create a 3-4 day course which consists of:**
- **Day 1: Grasping the grid concept (Beginners)**
  - Generic knowledge (Grids, EGEE project)
  - T+Practicals on Workload Management and Data Management
  - Example Applications on the grid
- **Day 2: Extending skills (Intermediate Users)**
  - Advanced security aspects (eg. Myproxy)
  - Advanced applications' aspects (eg. we do MPI)
- **Day 3: Applications day (Advanced Users)**
- **Day 4: Training the Trainers, the Administrators etc.**

Tips: Applications SEE can run

- **Simulations (CPU intensive, SEE has >1000 of CPUs)**
- **Bulk data processing (Storage, SEE has >100TBs+CPU)**
- **Parallel jobs (SEE: quite a few MPI supporting clusters)**
  - MPI fully supported in many (huge) clusters within SEE
  - Interconnect is typically Gigabit, but some have Myrinet!
  - Constellation-like calculations should be possible in the future
  - HellasGrid is designed with Gbit provision e2e. Take advantage!
- **We'd like to see: workflows of ~medigrid applications**
  - Weather models, Fire behaviour, Flooding, Landslides etc
  - Anything with direct or indirect impact on level-of-living for SEE
- **We'd like to see: Scavenged resources & Applications**
  - We can do that, LiveWN proved that it is technically possible
  - Needs lots of evolution in the Information System
  - Needs lots of experimentation on resource & job matchmaking

Tips: How consistent is the IS?

- **Do clusters have WNs with...**
  - Same OS? Same CPU? Same clockrate? Same RAM? ...
- **There is some heterogeneity in the resources provided by a given cluster (eg. slightly different clockrates).**
- **It is imposed by the combination of hardware diversity and the capabilities of the IS, ie. GLUE schema v1.x!**
- **GLUE schema v2 will allow the description of "subclusters", so this issue will be overcome; to be released in early 2007.**
- **For the time being, ask users to put their minimum job requirements in a .jdl, and if they see an error, to report immediately to Operations (SA1)'s help-desk tools.**

Tips: Why do checkpointing?

- **Check-pointing is the technique of storing the state of your job, typically in a set of files, so that you can later "reanimate" it. It is very critical for “long jobs”.**
- **If jobs take more than 24 hrs, it is imperative for users to consider check-pointing for a number of reasons:**
  - User: software engineering reasons, ie. debugging a checkpointable job is simpler to debug on longer-term issues, since it is possible to restart it on eg. the 6th day of execution
  - User: overcome the limit of grid queues (infinite jobs!) and be able to pause and reanimate the job arbitrarily
  - User: by limiting the execution time of your jobs, you can now tap much more resources (queues), which will decrease the total time
  - Systems administration: cluster downtimes can happen for either scheduled reasons (m/w upgrade, hardware installation or reconfiguration etc) or various unscheduled reasons (power/network/airconditioning outages, security issues)

Tips: How to do checkpointing

- **There are currently two well-known techniques to implement checkpointing:**
  - Update some central service eg. some database like AMGA, about the current state of the job (easy, if state==integer)
  - Save the state of the job in a set of files, make a tarball (.tar.gz) & register it on a nearby SE with a predefined LFN!
- **Middleware is supposed to be able to assist, but in gLite v3.0 this is currently unsupported function (?)**
- **In effect, workflow and job management tools (ganga?) do state-maintenance at the large-scale of a gridified application, can this be integrated with checkpointing?**

Which user are you looking for?

Contact info

Thank You

gef@grnet.gr