



# Future CAF directions

**Frank Wurthwein**

*UCSD*

- **Condor Issues**
- **Grid Issues**
- **Application Issues**





# Condor Issues

- **Hierarchical Fair Share**
  - **Highest prio because we need it to get rid of fbsng**
  
- **Headnode failover**
  - **Low priority because headnodes are quite stable**



# Grid Issues

Glide-in

global brokering

CafMon on the grid

global resource utilization monitoring

global status monitoring



# Glide-in

Our way of accessing any grid site as long as it has a vanilla GRAM.

## Issues:

We need to use user cert for glide-in submission.

We need to decide on architecture:

Where does the glide-in submitter live?

Do we submit glide-ins in bulk? (e.g. SAM-Grid)

## Assumptions:

cdfsoft is dynamically installed

DH (e.g. SAM station) is dynamically installed

DB (e.g. FroNtier) is dynamically installed

**i.e. We have no responsibility for any of the above!**



# Global Brokering

We want an electronic broker instead of a manual one.

Can we simply reuse JIM broker ?

Or are we better off with something from glite or LCG ?

E.g., how does the resource ClassAd of a random grid site reach our collector ? GLUE schema? How is it advertized ?  
Information Service ?



## CafMon on the Grid

***With glide-in we get CafMon functionality for free (we hope).  
However, should we offer to develop CafMon functionality  
as a standalone service that plugs in neatly into SAM-Grid  
and whatever CMS chooses to use?***



# Accounting

***We want global accounting of resource utilization.***

**We expect to get it by having each glide-in submission site hooked up with MonALISA to a central accounting server via the CAF's XML monitoring server.**



# Global State Monitoring

**We want global state monitoring of all services.  
Terrence started this for SDSC using NAGIOS.  
If this is successful we can consider it for all our  
services.**



# Application Support

**Reliability & robustness via retry**

**concatenation & workflow**

**skip event**



## Reliability & robustness via retry

Concerned that data volume & compute volume increases faster than CAF system reliability, especially in grid context where we do not control the site infrastructure!

To isolate the user from pain due to failure bookkeeping, we need to be able to retry failed segments.

To do so requires well controlled work environment that keeps track of itself.

Should we take responsibility for it or is this somebody else's job?



## Concatenation & workflow

If we took responsibility for a more elaborate “process wrapper” then we could take responsibility for more complex workflows, e.g. Concatenation, or MC generation, or reconstruction-validation chain.

Imagine providing easy to configure workflows for the user.



## Skip event

The functionality Elliot produced for the production farm is another example of a workflow manager.

This might be of interest for general users.

How far should one push this ?

All the way towards standard tcl fragment management (recall mcProd)?

Or should we keep out of this business altogether?