



# Fabric Management



The European DataGrid Project Team

<http://www.eu-datagrid.org>

---



# Contents

- Definition
- Fabric Management architecture overview
- User job management overview
- Present status (R1.2)
  - LCFG
  - LCAS + edg\_gatekeeper



# What is Fabric Management

## ➤ Definitions:

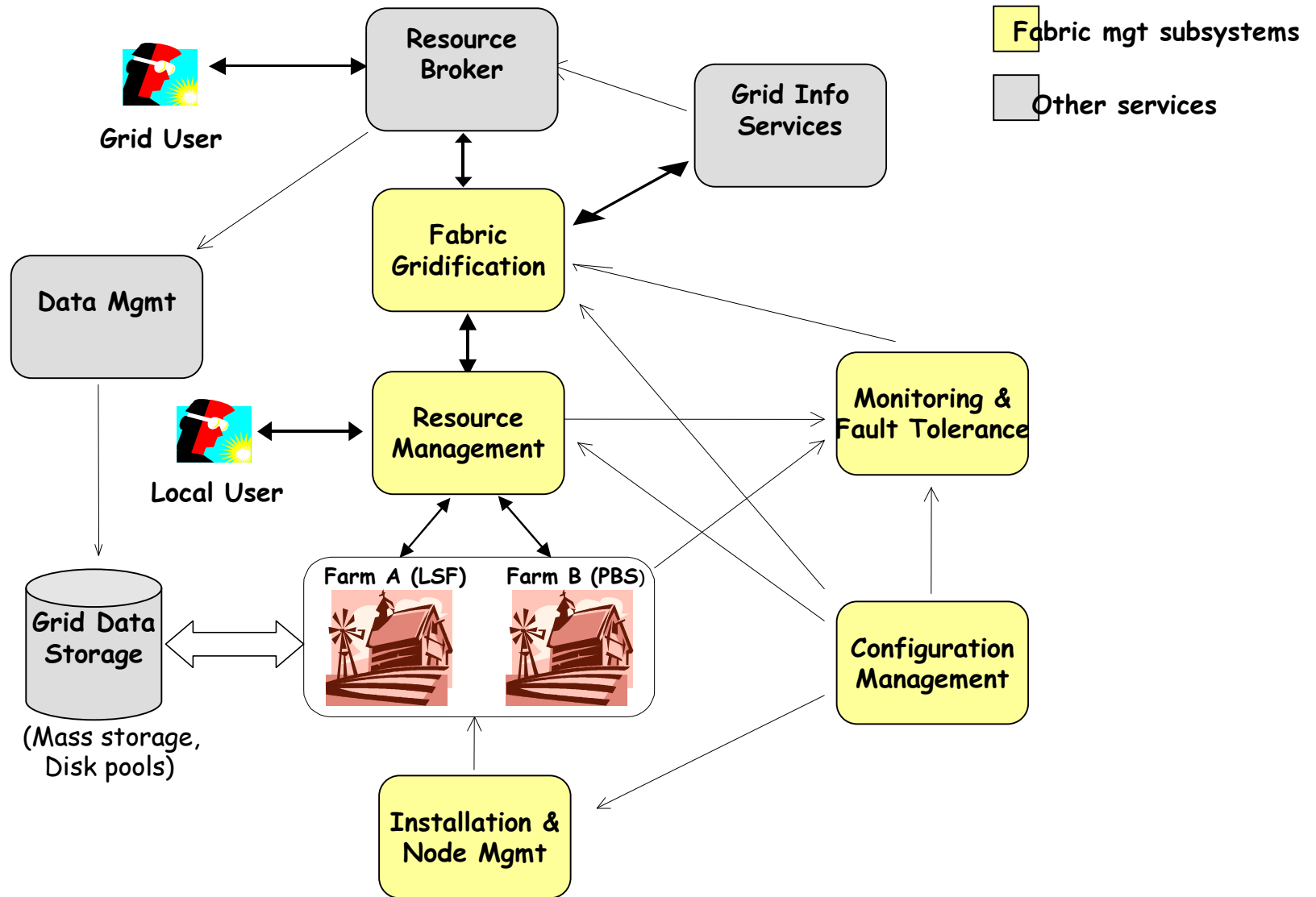
- **Cluster (or Farm):** "A collection of computers on a network that can function as a single computing resource through the use of tools which hide the underlying physical structure".
- **Fabric:** "A complete set of computing resources (processors, memory, storage) operated in a coordinated fashion with a uniform set of management tools".

## ➤ Functionality:

- Enterprise system administration - scalable to ~10K nodes
- Provision for running grid jobs
- Provision for running local jobs



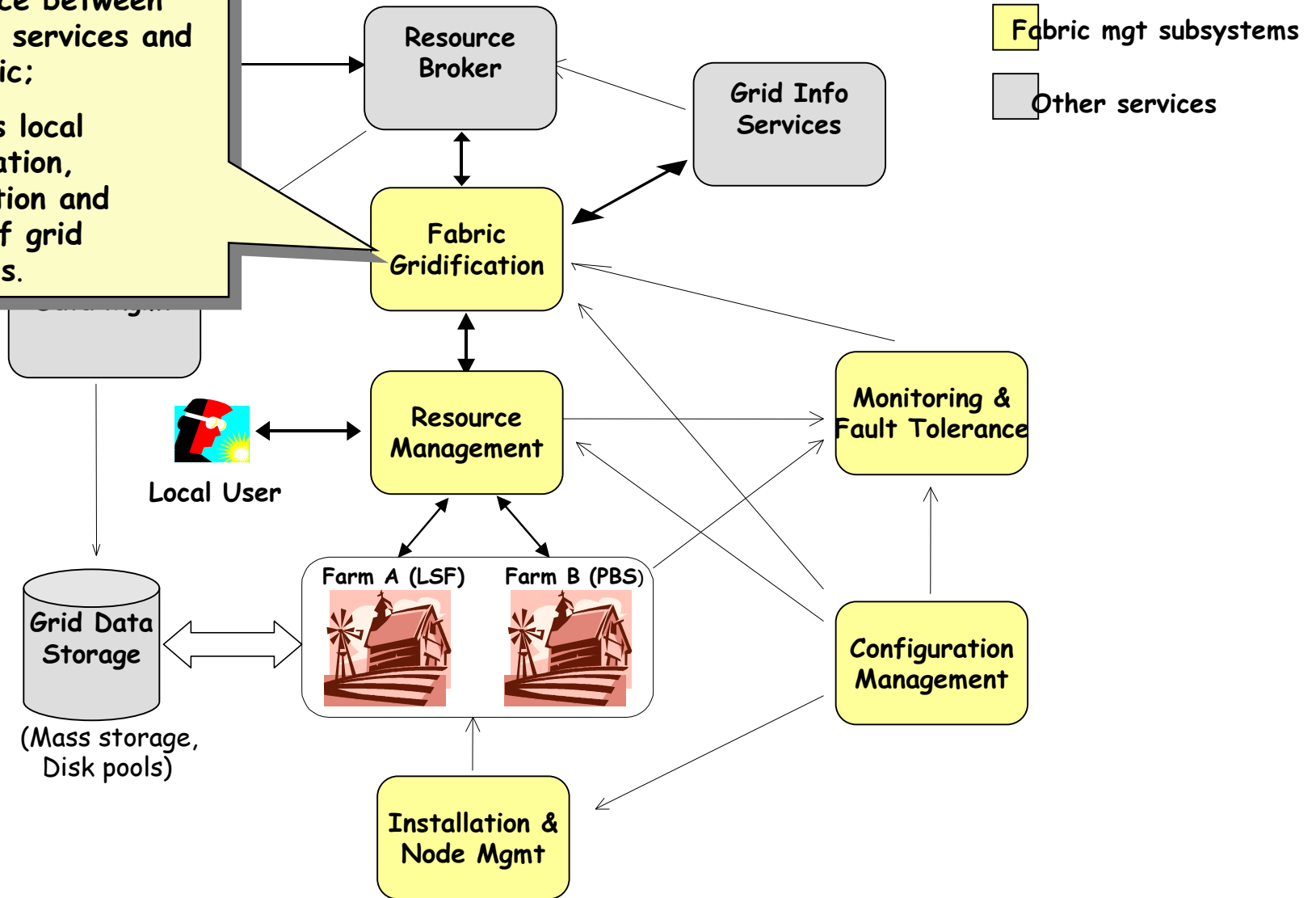
# Architecture logical overview





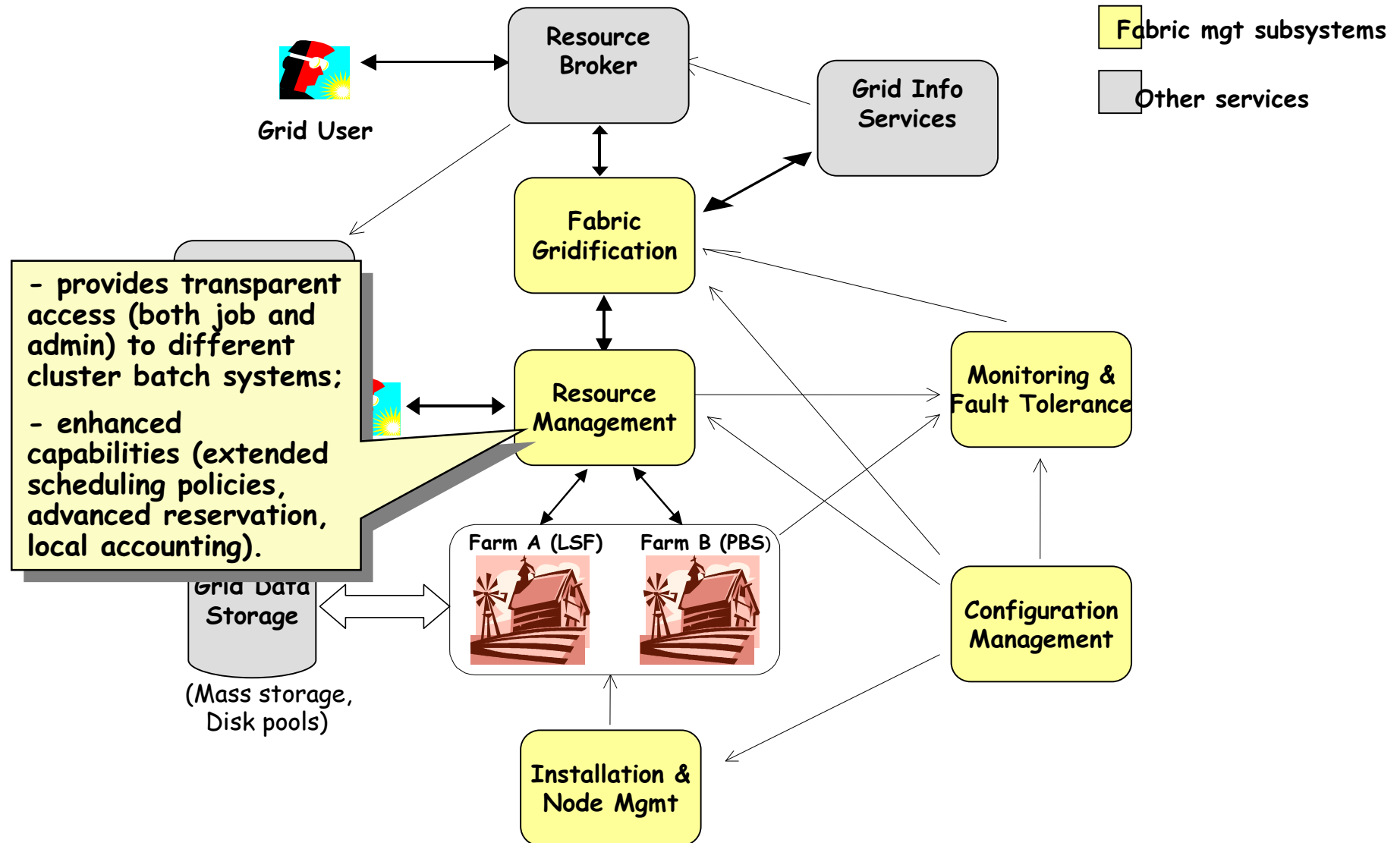
# Architecture logical overview

- Interface between Grid-wide services and local fabric;  
- Provides local authentication, authorization and mapping of grid credentials.



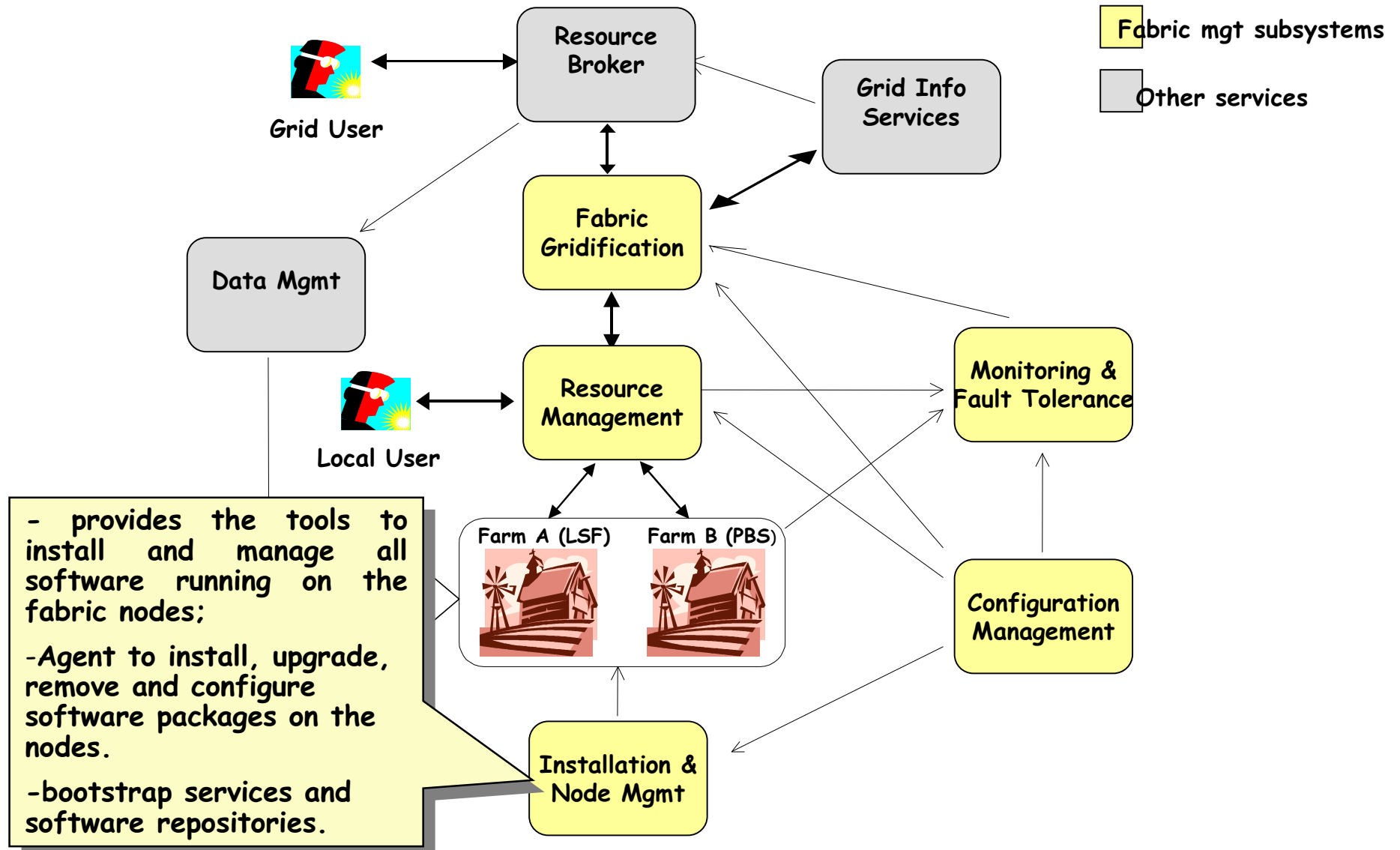


# Architecture logical overview



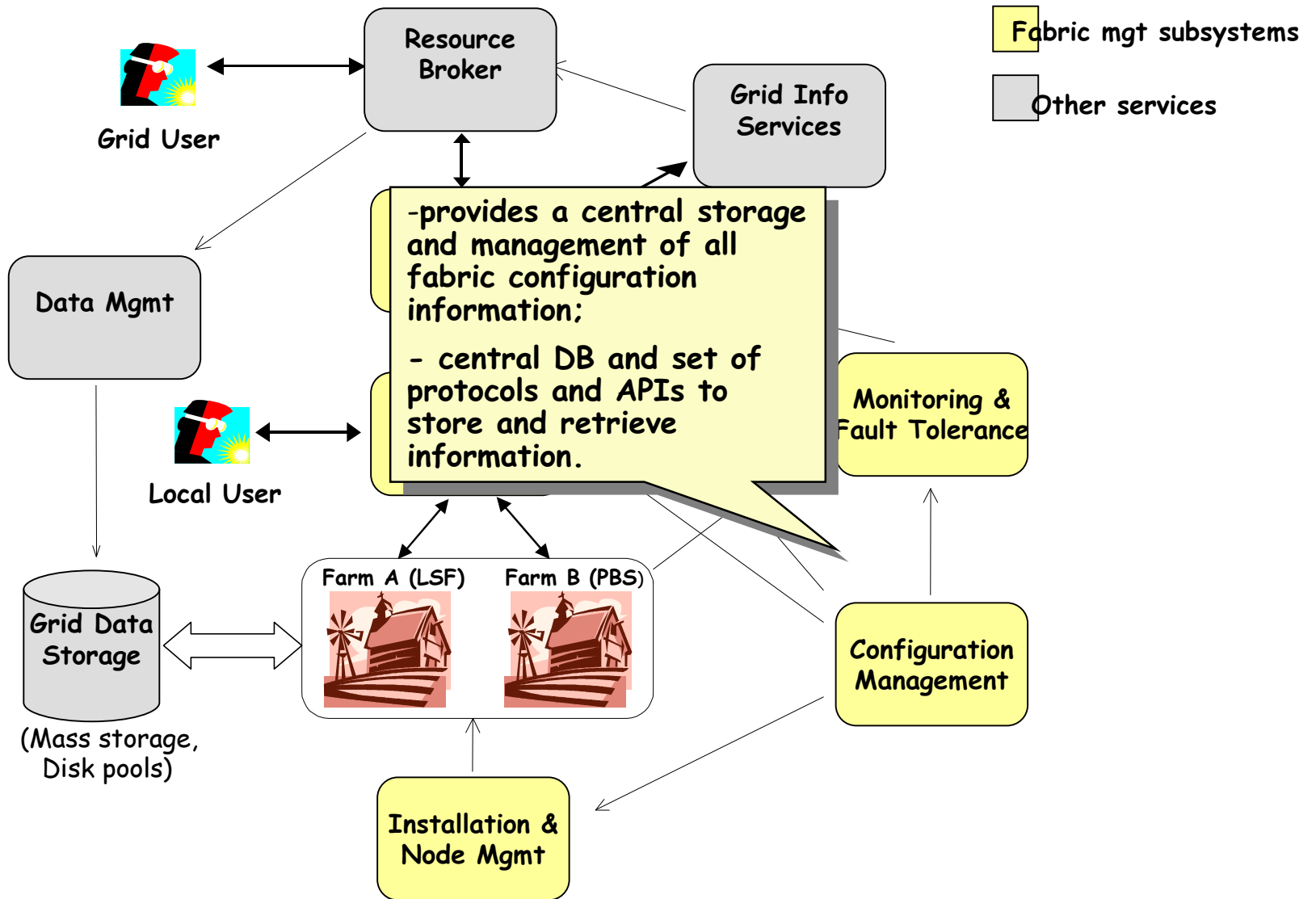


# Architecture logical overview





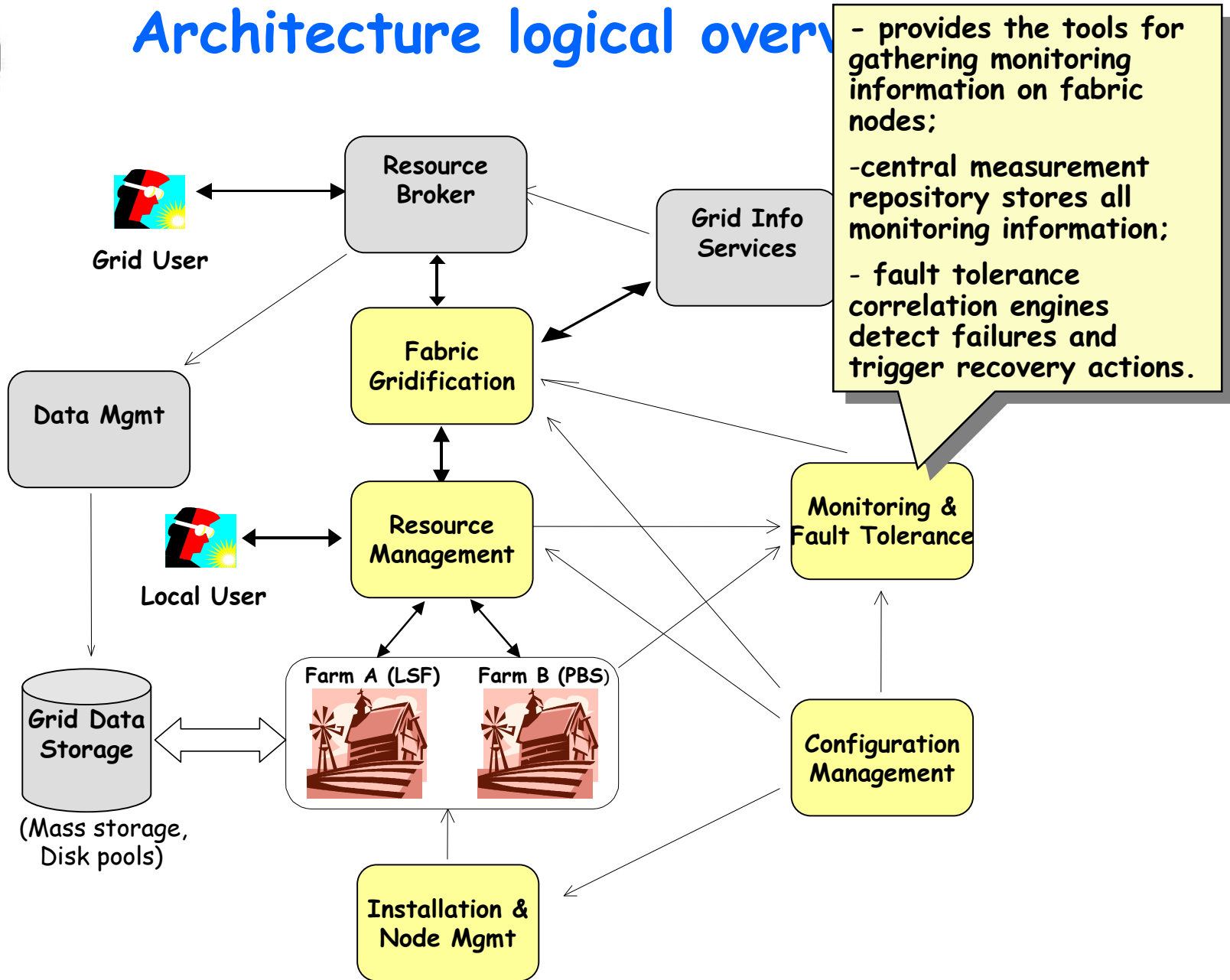
# Architecture logical overview





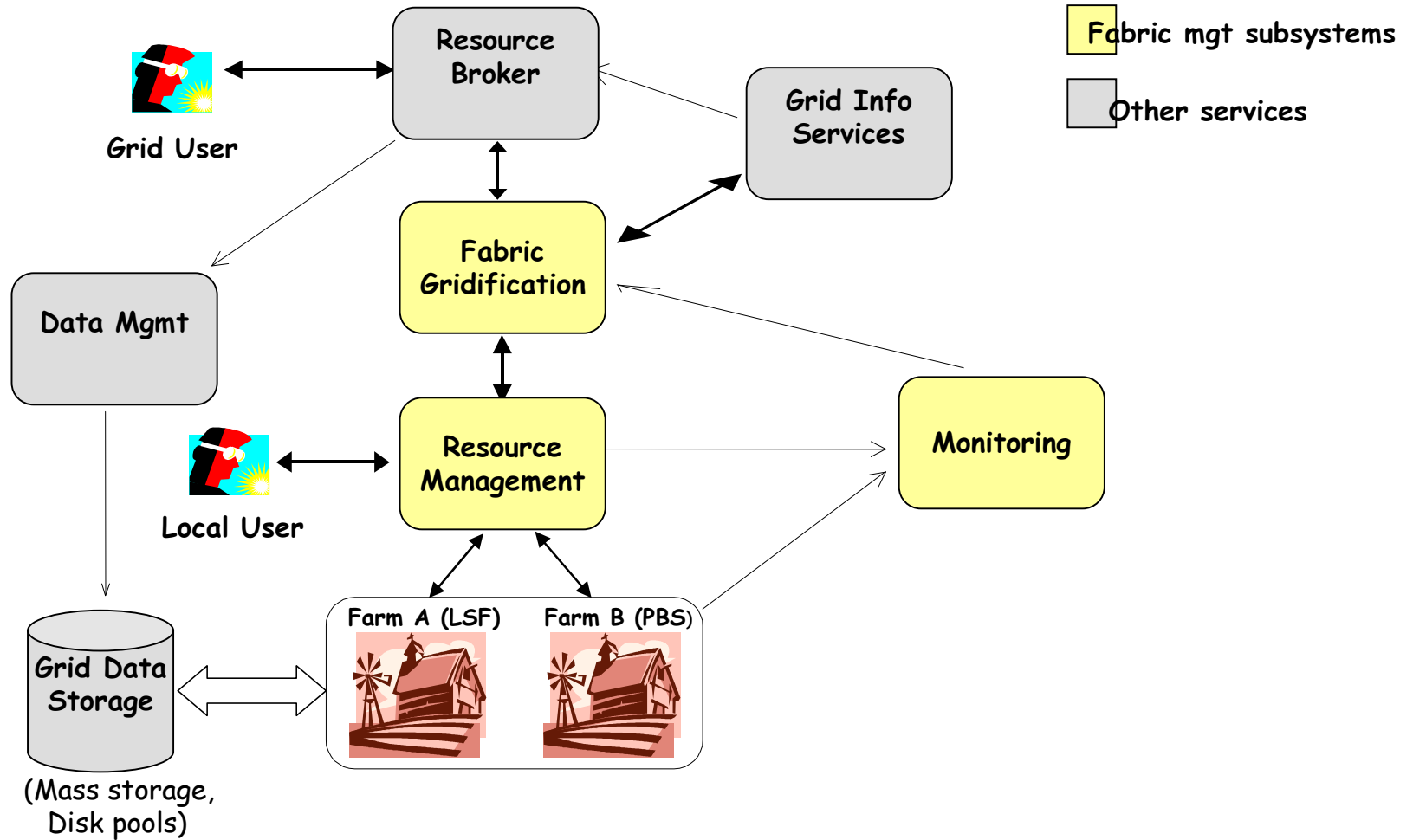


# Architecture logical overview



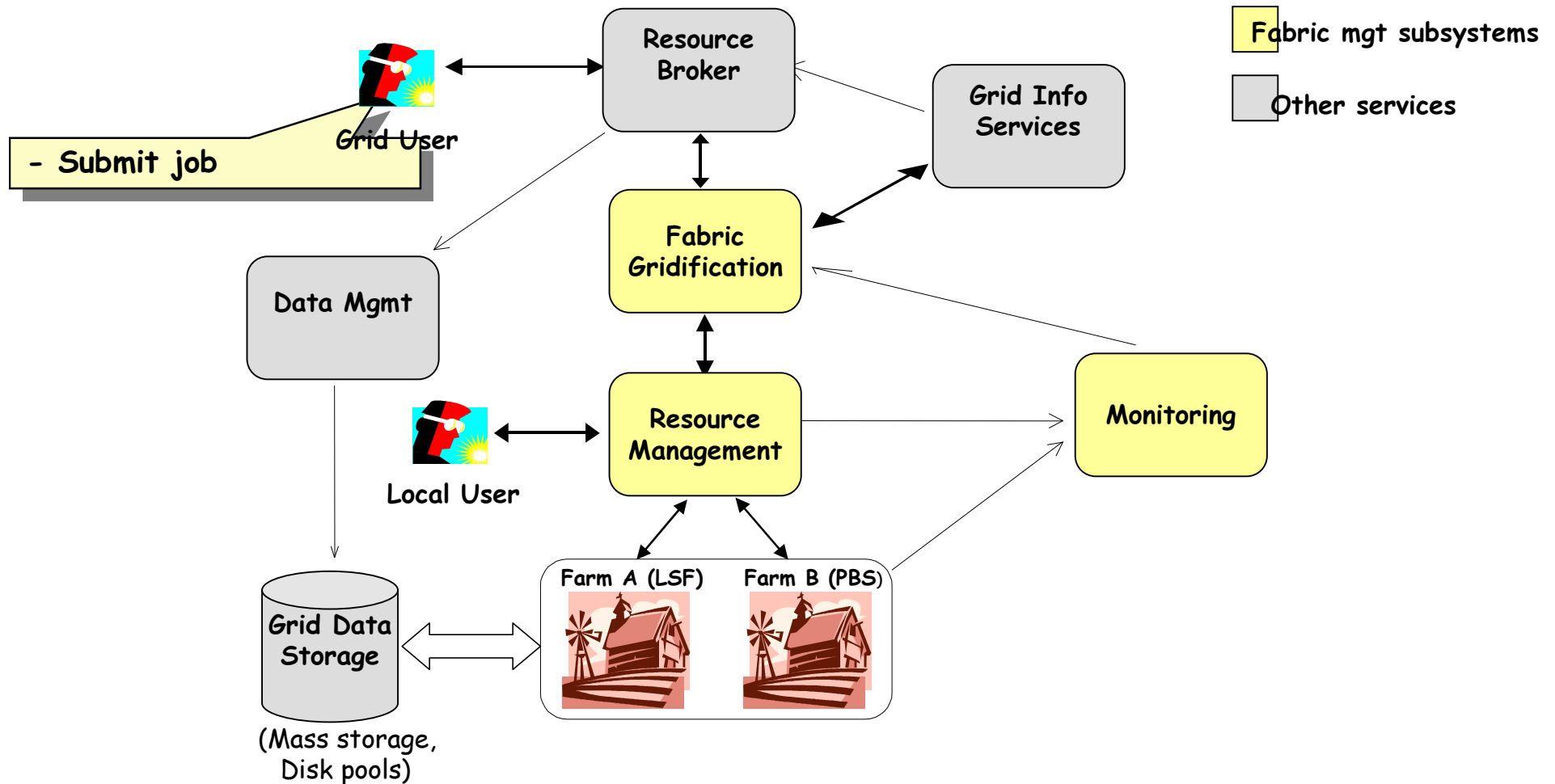


# User job management (Grid and local)



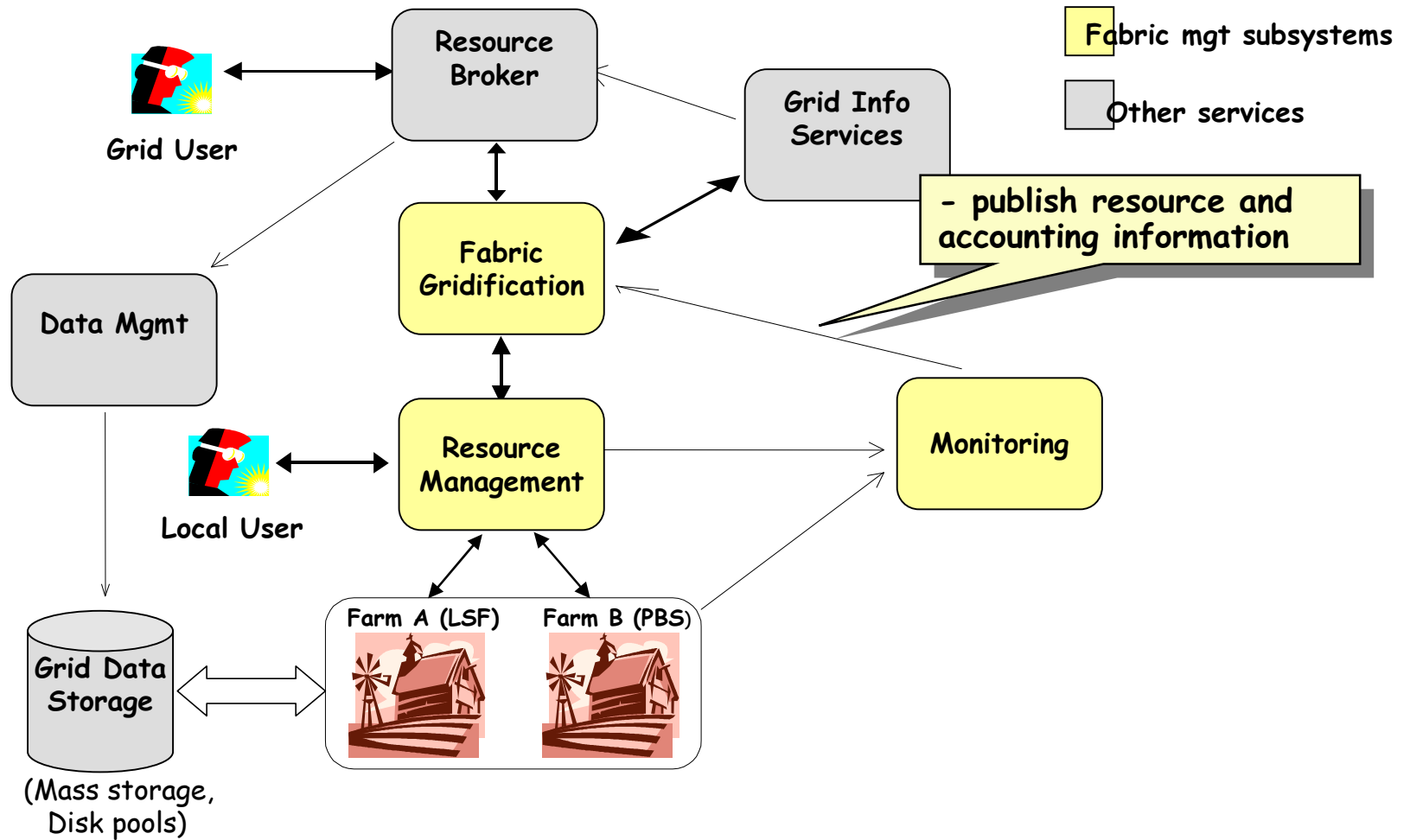


# User job management (Grid and local)



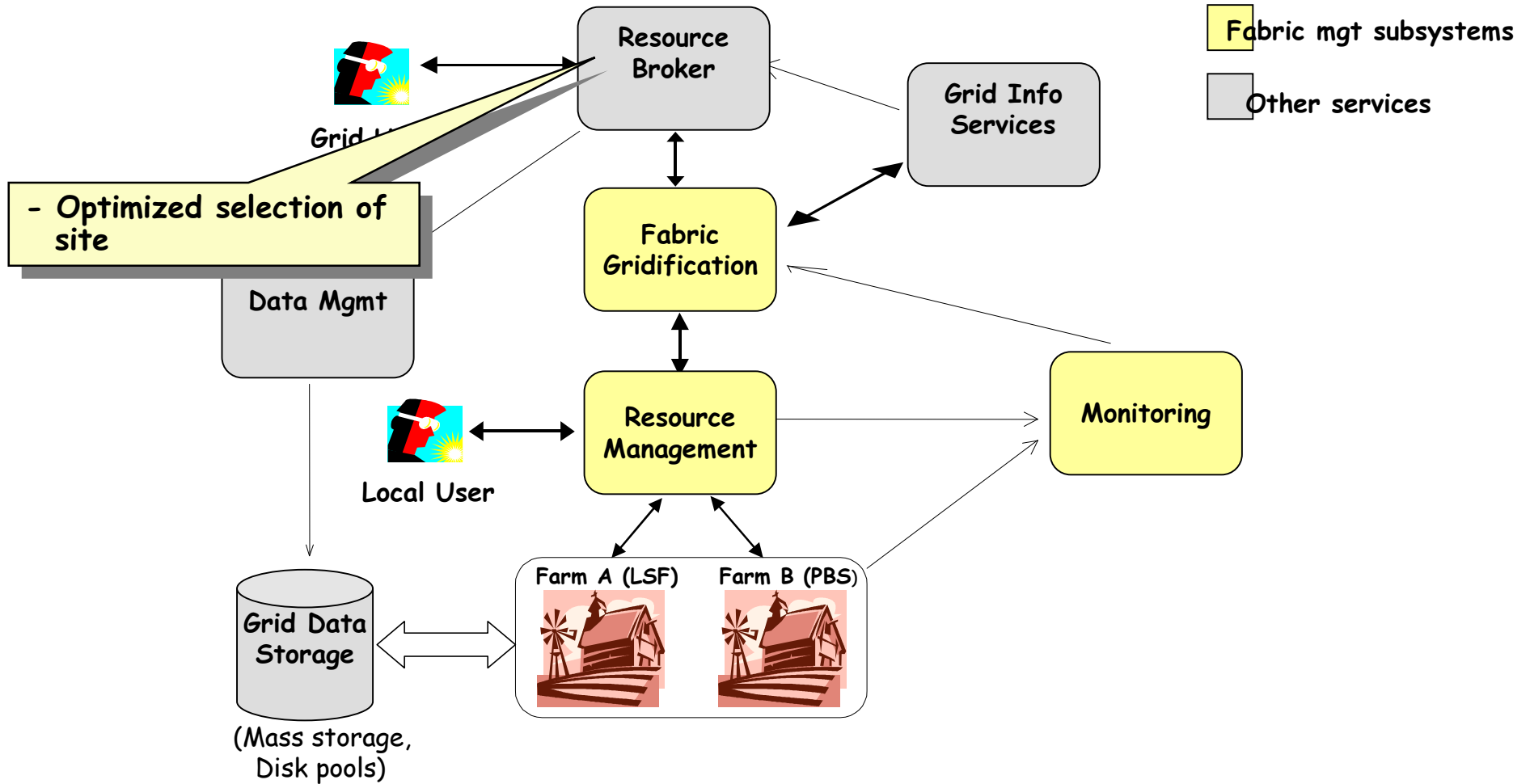


# User job management (Grid and local)



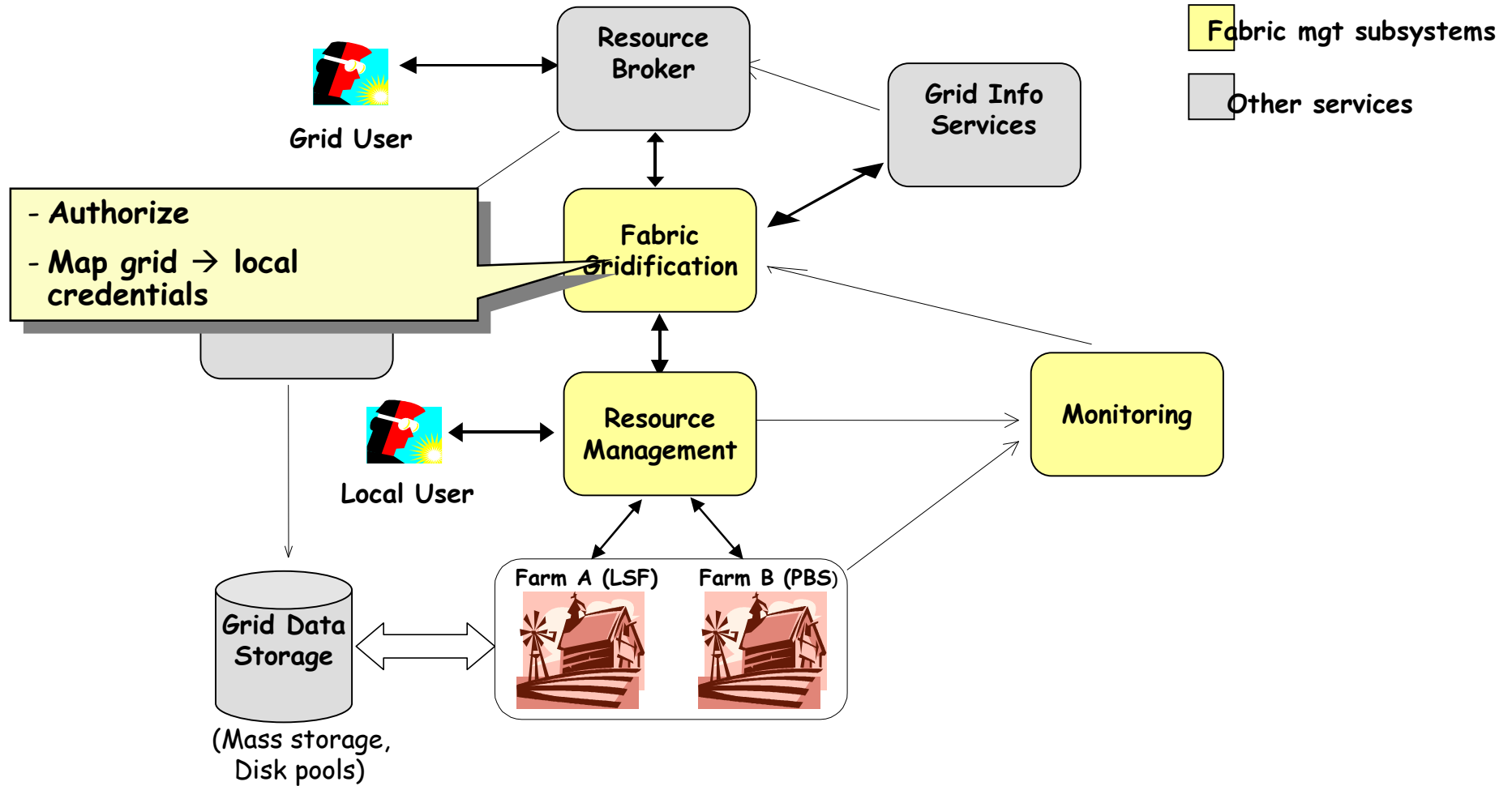


# User job management (Grid and local)



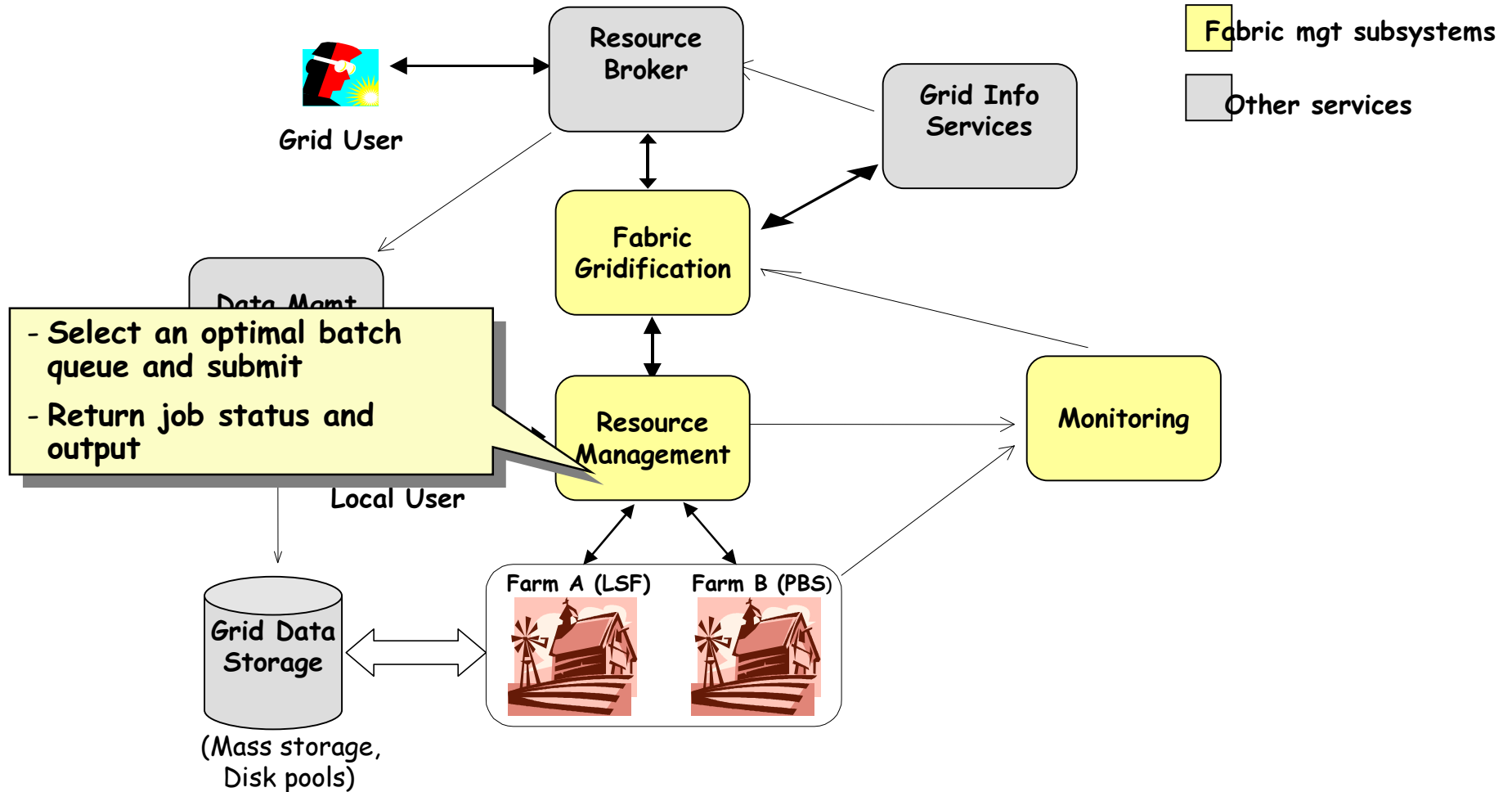


# User job management (Grid and local)





# User job management (Grid and local)





## Fabric Management @ Release 1.2

- Installation and configuration:  
LCFG (Local ConFiGuration system)
- Gridification:  
LCAS + edg\_gatekeeper



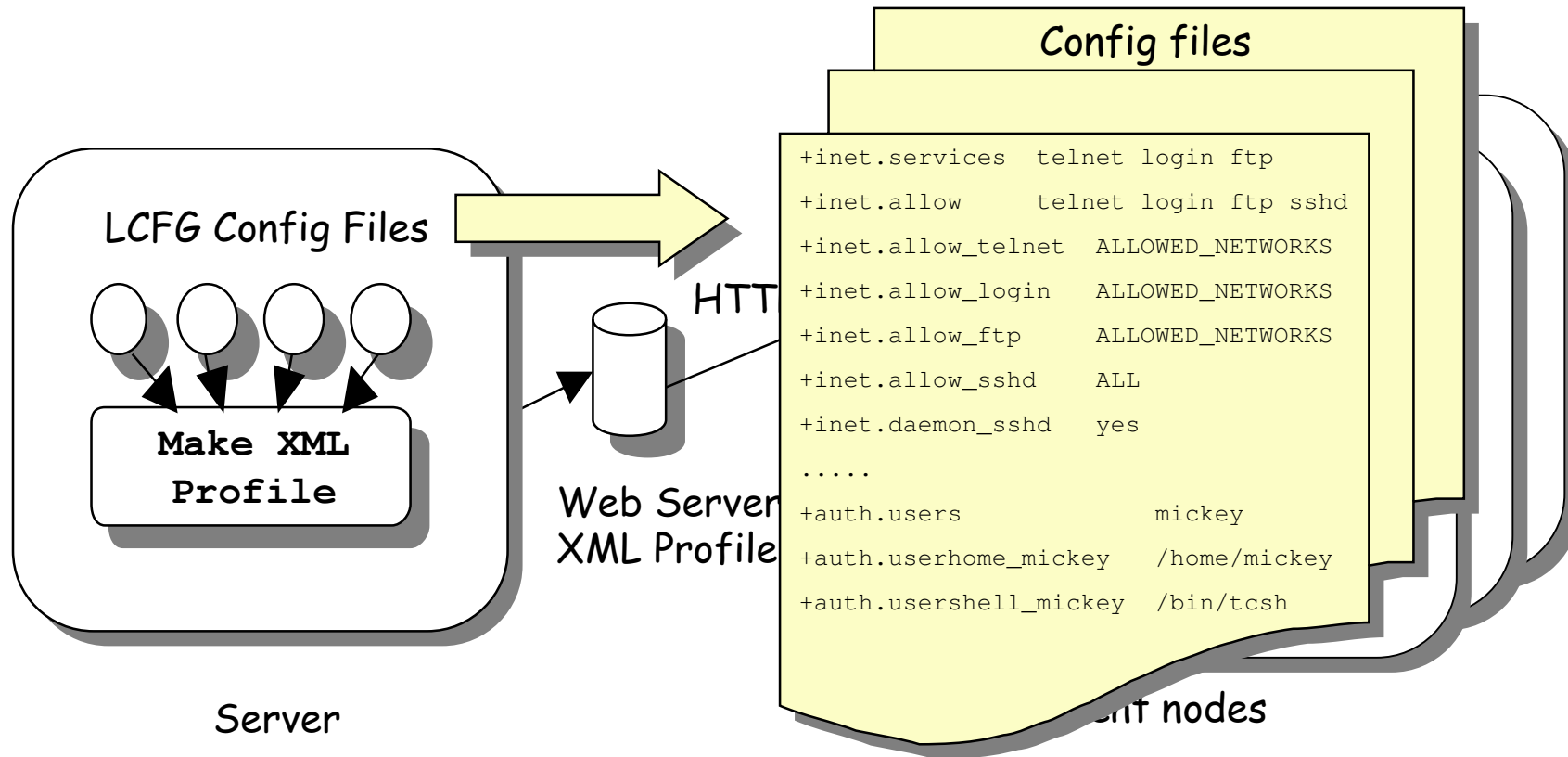


## LCFG (Local ConFiGuration system)

- LCFG is originally developed by the Computer Science Department of Edinburgh University
- Handles automated installation, configuration and management of machines
- Basic features:
  - automatic installation of O.S.
  - installation/upgrade/removal of all (rpm-based) software packages
  - centralized configuration and management of machines
  - extendible to configure and manage EDG middleware and custom application software



# LCFG system architecture

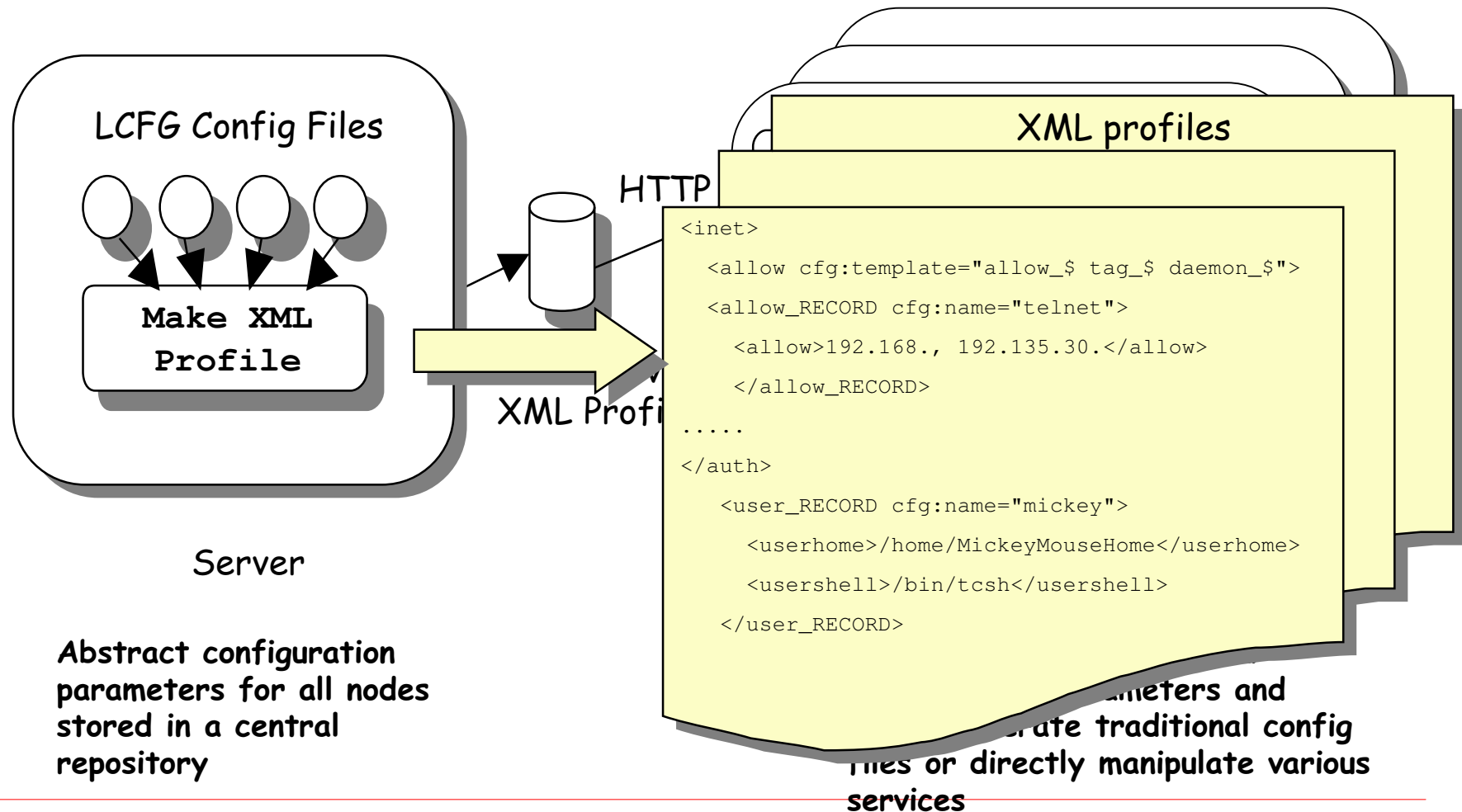


Abstract configuration parameters for all nodes stored in a central repository

A collection of agents read configuration parameters and either generate traditional config files or directly manipulate various services

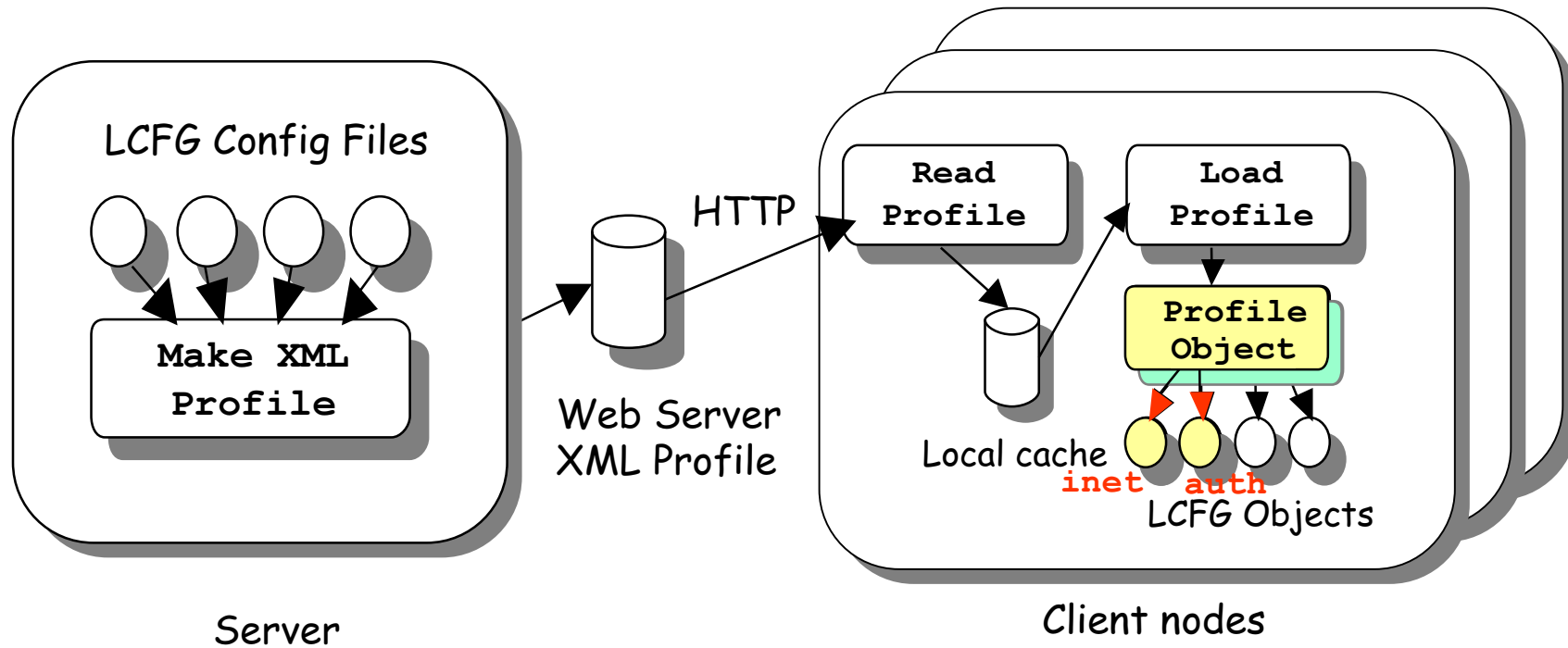


# LCFG system architecture





# LCFG system architecture

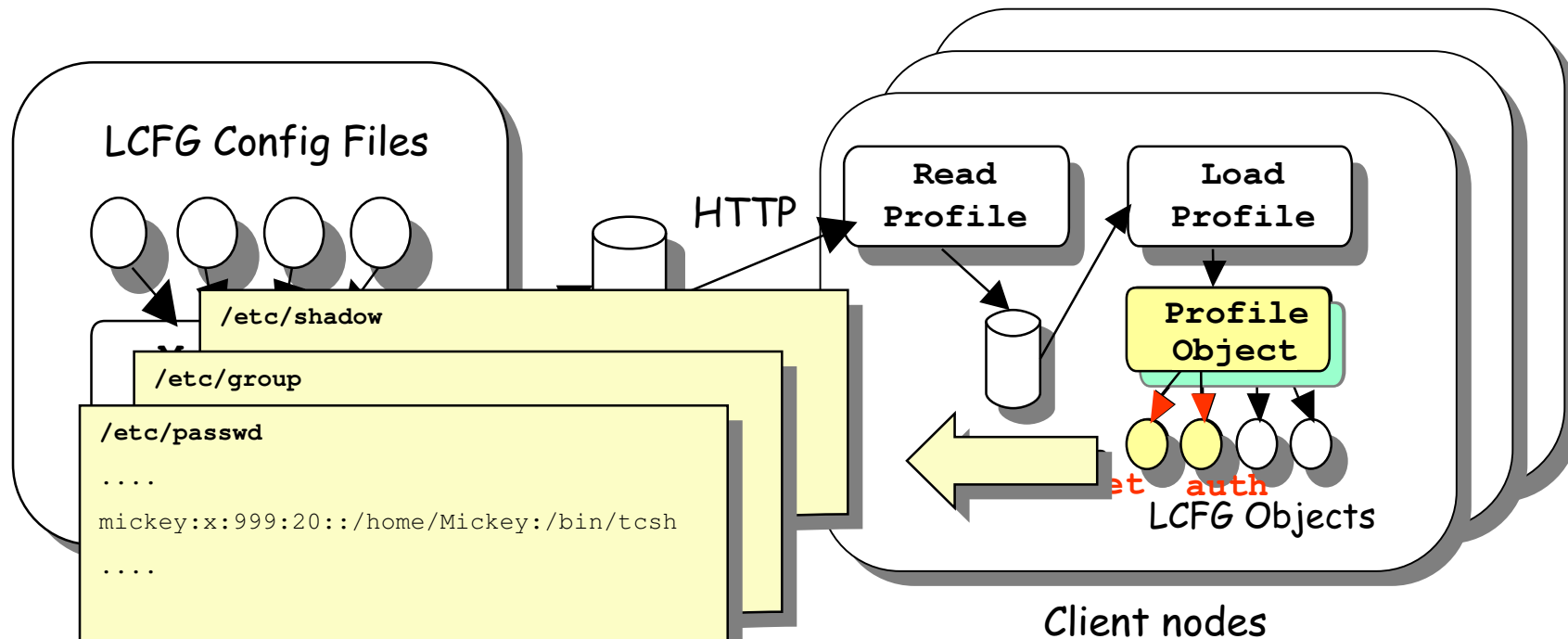


Abstract configuration parameters for all nodes stored in a central repository

A collection of agents read configuration parameters and either generate traditional config files or directly manipulate various services



# LCFG system architecture

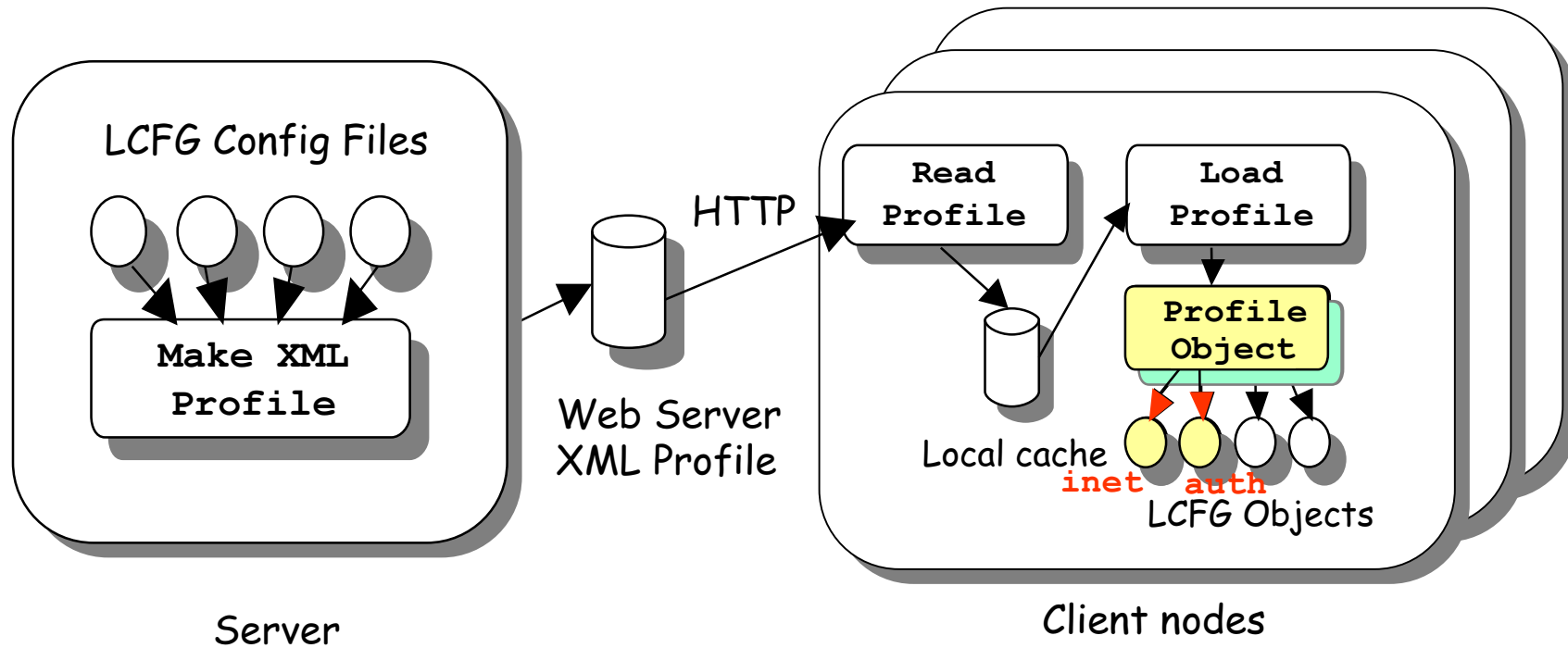


Abstract configuration parameters for all nodes stored in a central repository

A collection of agents read configuration parameters and either generate traditional config files or directly manipulate various services



# LCFG system architecture



Abstract configuration parameters for all nodes stored in a central repository

A collection of agents read configuration parameters and either generate traditional config files or directly manipulate various services



## LCFG: what's a component?

- Component == object
- It's a simple shell script (*... but moving to Perl*)
- Each component provides a number of "methods" (start, stop, configure,...) which are invoked at appropriate times
- A simple and typical component behaviour:
  - Started when notified of a configuration change
  - Loads its configuration (locally cached)
  - Configures the appropriate services, by translating config parameters into a traditional config file and reloading a service if necessary (e.g. restarting a init.d service).
- LCFG provides **components** to manage the configuration of services of a machine: inet, auth, nfs, cron, ...
- Admins/mw developers can build new **custom components** to configure and manage their own applications



## LCFG component skeleton

```
#!/bin/bash2
class=mycomp                                # set the name of the component (mycomp)
. /etc/obj/generic                          # include std methods and definitions
Start() {                                   # Start the component
}
Configure() {                               # Do the configuration
}
Case "$1" in                               # 'main' program
    configure) Configure;
    *) DoMethod "$@";
esac
```





# LCFG component methods

## ➤ Start () method:

```
Start() {                                     # 'Start' the component
    Generic_Start;                             # standard setup steps
    Configure;                                 # reconfigure the component
    if [ $? = 0 ]; then
        OK "Component mycomp started"
    else
        Fail "Starting component mycomp"
    fi
}
```

## ➤ Configure () method:

```
Configure() {                                 # Do the configuration
    LoadResources myresource1 myresource2 ...
    CheckResources myresource1 myresource2 ...
    # your code
    do_whatever ...
    return status;
}
```

**Only the RED parts are specific to a component!**



# LCFG component example (I)

ldconf component - Configures /etc/ld.so.conf (*Cal Loomis*)

## .def file:

```
class          ldconf
methods        start configure
conf file
paths
```

## Resources defined on server:

```
conf file      /etc/ld.so.conf
paths          /usr/local/lib /opt/globus/lib
```



## LCFG component example (II)

### Component: Configure() method

```
Configure() {  
  
    LoadResources  conffile paths  
    CheckResources conffile paths  
  
    # Update ld.so.conf file.  
    for i in $paths; do  
        line=`grep $i $conffile`  
        if [ -z "$line" ]; then  
            echo "$i" >> $conffile  
        fi  
    done  
    /sbin/ldconfig;  
    return $?;  
}
```

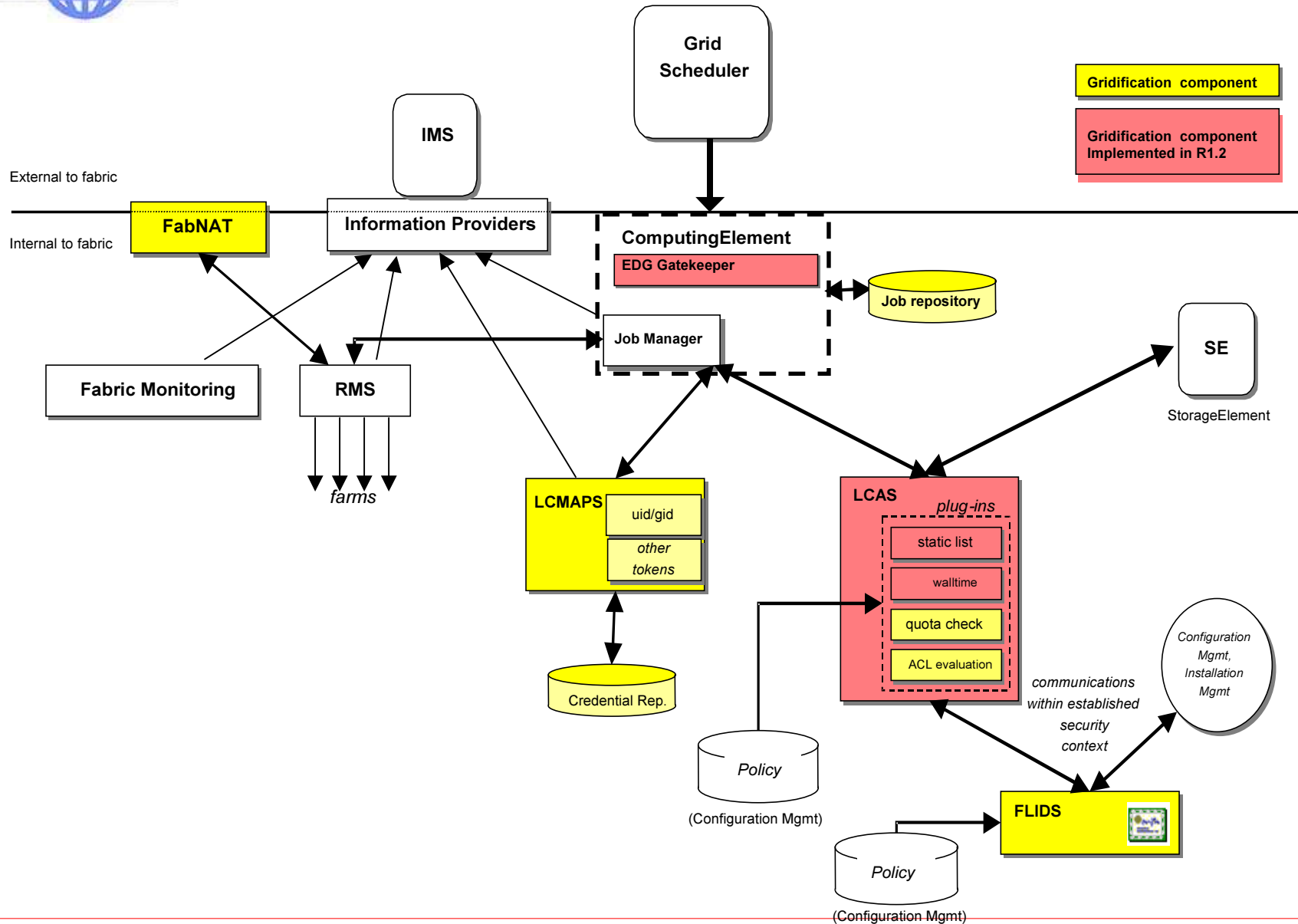


## Software distribution with LCFG

- It is done with an LCFG tool called `updaterpms`.
- The standard system packaging format is used: `rpm` for Linux
- It is managed via an LCFG component.
- Functionality:
  1. Compares the packages currently installed on the local node with the packages listed in the configuration
  2. Computes the necessary install/deinstall/upgrade operations
  3. Orders the transaction operations taking into account rpm dependency information
  4. Invokes the `packager` (RPM) with the right operation transaction set
- `Packager` functionality:
  1. Read operations (transactions)
  2. Downloads new packages from repository
  3. Executes the operations (installs/removes/upgrades)



# Gridification Architecture



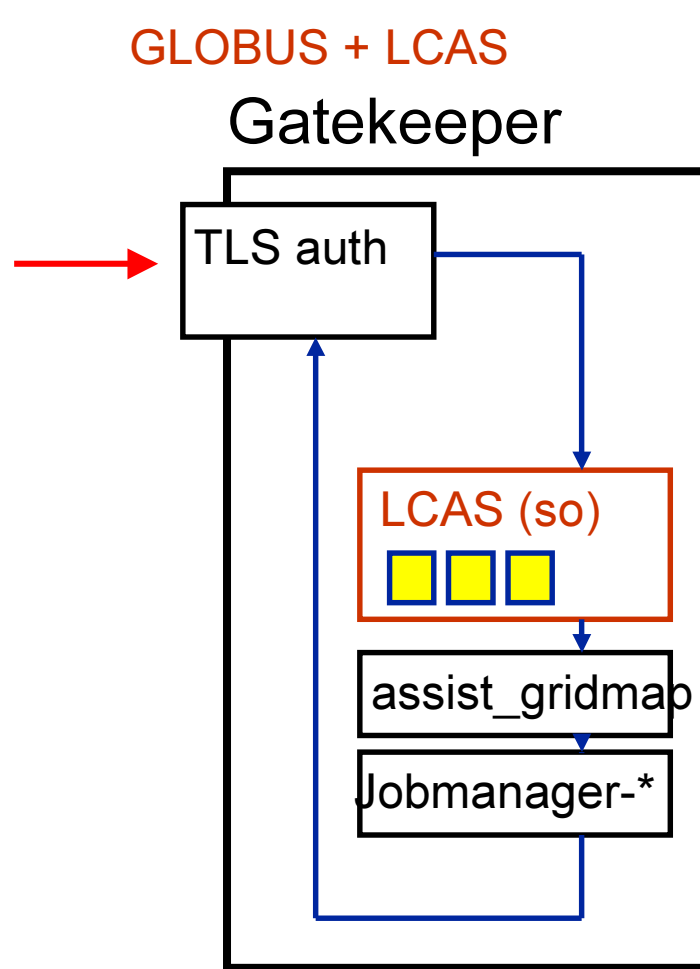
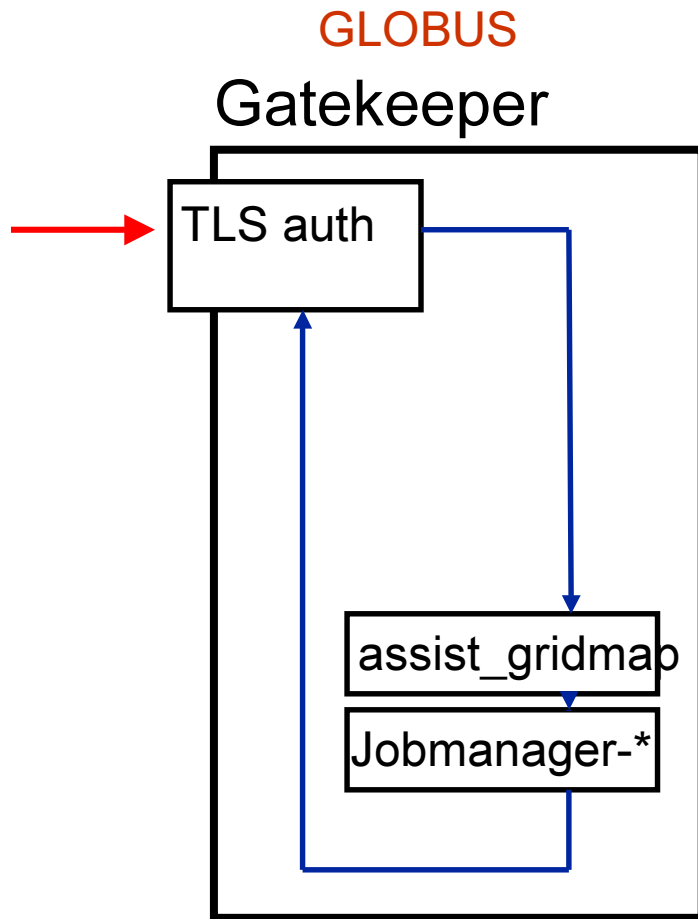


## LCAS 1.0 (EDG release 1.2)

- The **Local Centre Authorization Service (LCAS)** handles authorization requests to the local computing fabric.
- In this release the LCAS is a shared library, which is loaded dynamically by the globus gatekeeper. The gatekeeper has been slightly modified for this purpose and will from now on be referred to as **edg-gatekeeper**.
- The authorization decision of the LCAS is based upon the **users' certificate** and the **job specification in RSL (JDL)** format. The certificate and RSL are passed to (plug-in) authorization modules, which grant or deny the access to the fabric. Three standard authorization modules are provided by default:
  - `lcas_userallow.mod`, checks if user is allowed on the fabric (currently the gridmap file is checked).
  - `lcas_userban.mod`, checks if user should be banned from the fabric.
  - `lcas_timeslots.mod`, checks if fabric is open at this time of the day for datagrid jobs.



# Authentication control flow EDG gatekeeper



\* And store in job repository



## Summary

- Two main fabric management components deployed on release 1.2:
- Installation and Configuration management functionality:
  - **LCFG (Local ConFiGuration system)**: handles automated installation, configuration and management of machines.
- Gridification functionality:
  - **LCAS (Local Centre Authorization Service) + edg\_gatekeeper**: handle authorization requests to the local computing fabric.
- And more components coming in next releases in the areas of fabric monitoring, resource management, and configuration management.
- **Experience and conclusions from this release**: automatic installation and configuration of the DataGrid middleware was very useful for the testbed due to the complexity of the software.