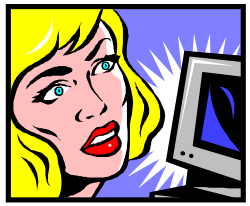
The EU DataGrid
Fabric Management

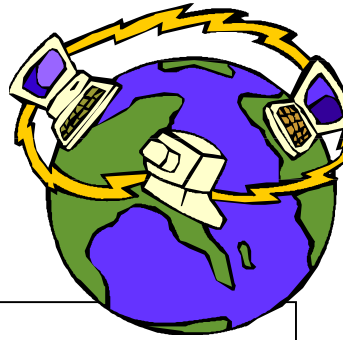
The European
DataGrid Project Team

http://www.eu-datagrid.org

EDG Tutorial Overview

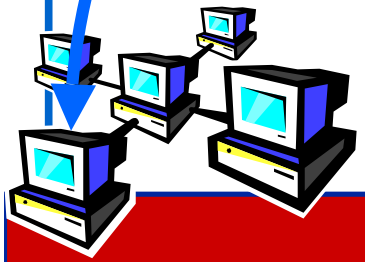


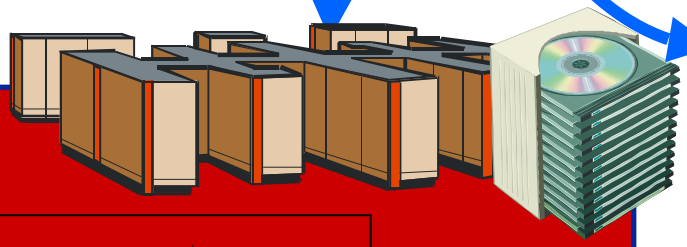
Workload Management Services



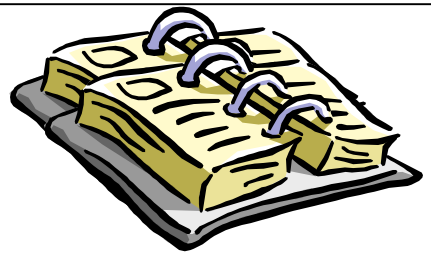
Data Management Services

Networking





Information Service



Fabric Management

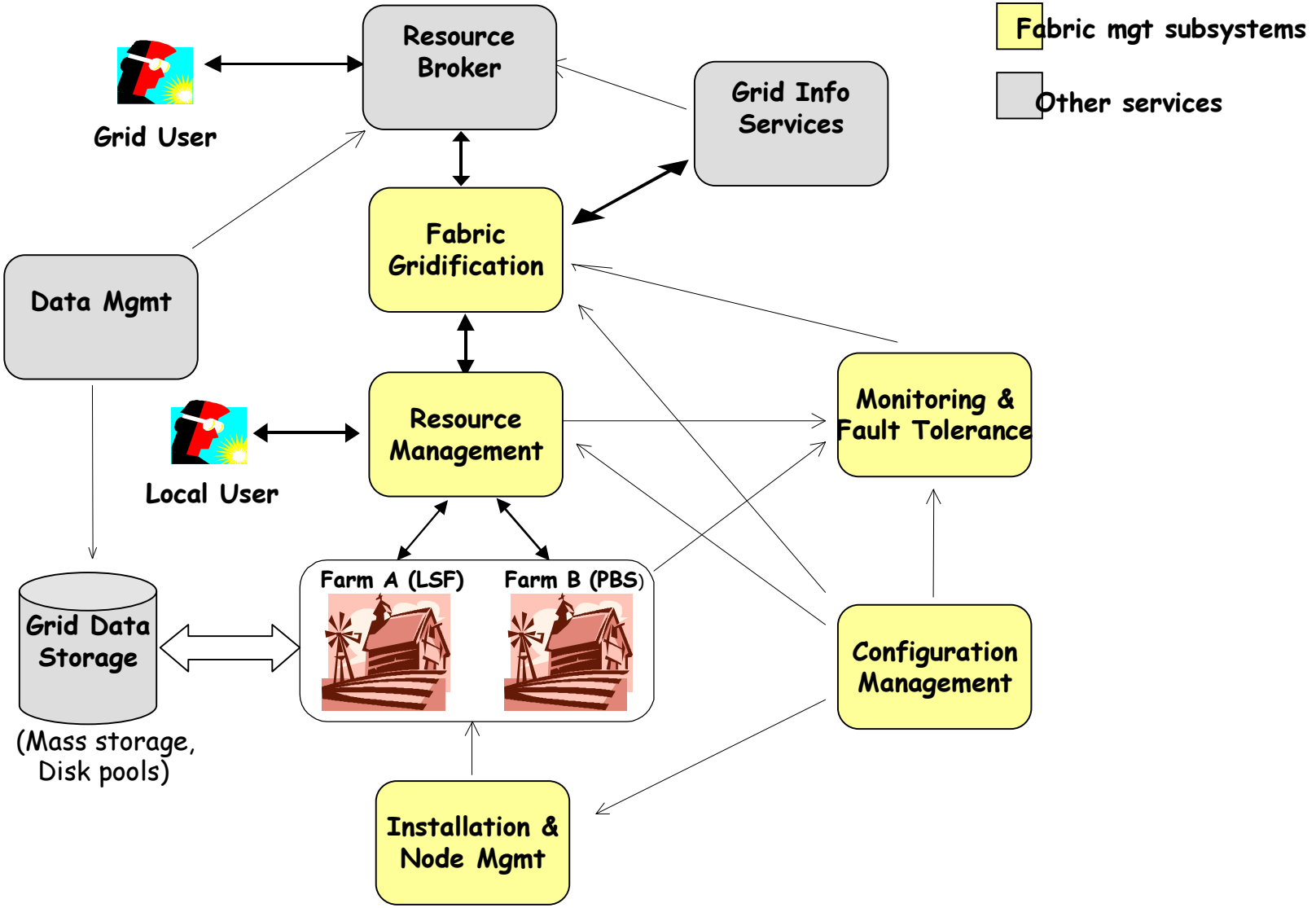
Contents

- Definition
- Fabric Management architecture overview
- User job management overview
- Present status (EDG2.0)
  - LCFG
  - LCAS + edg_gatekeeper
  - Fmon (Fabric MONitoring)
  - RMS (Resource Management System)

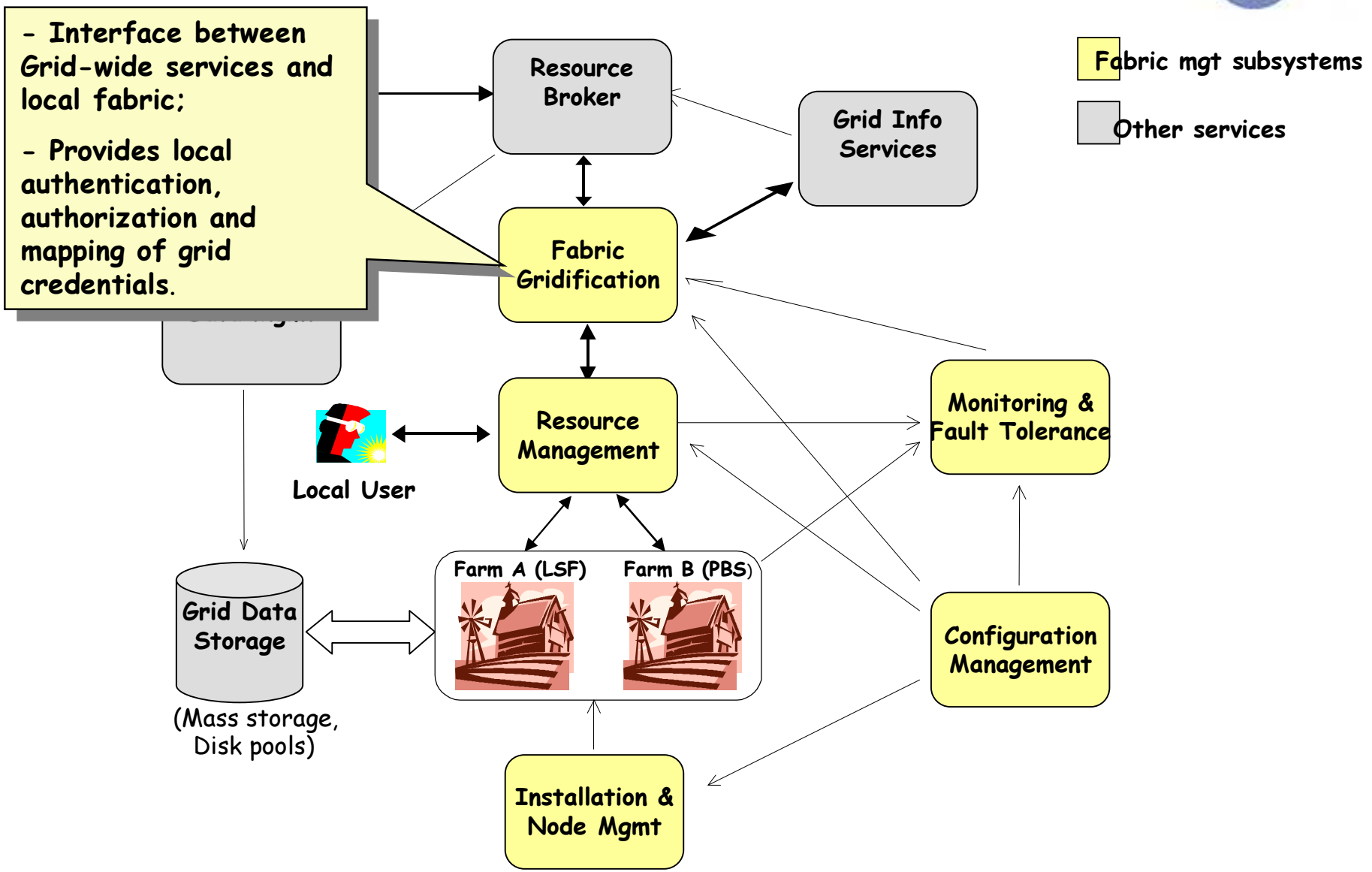
What is Fabric Management

- 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

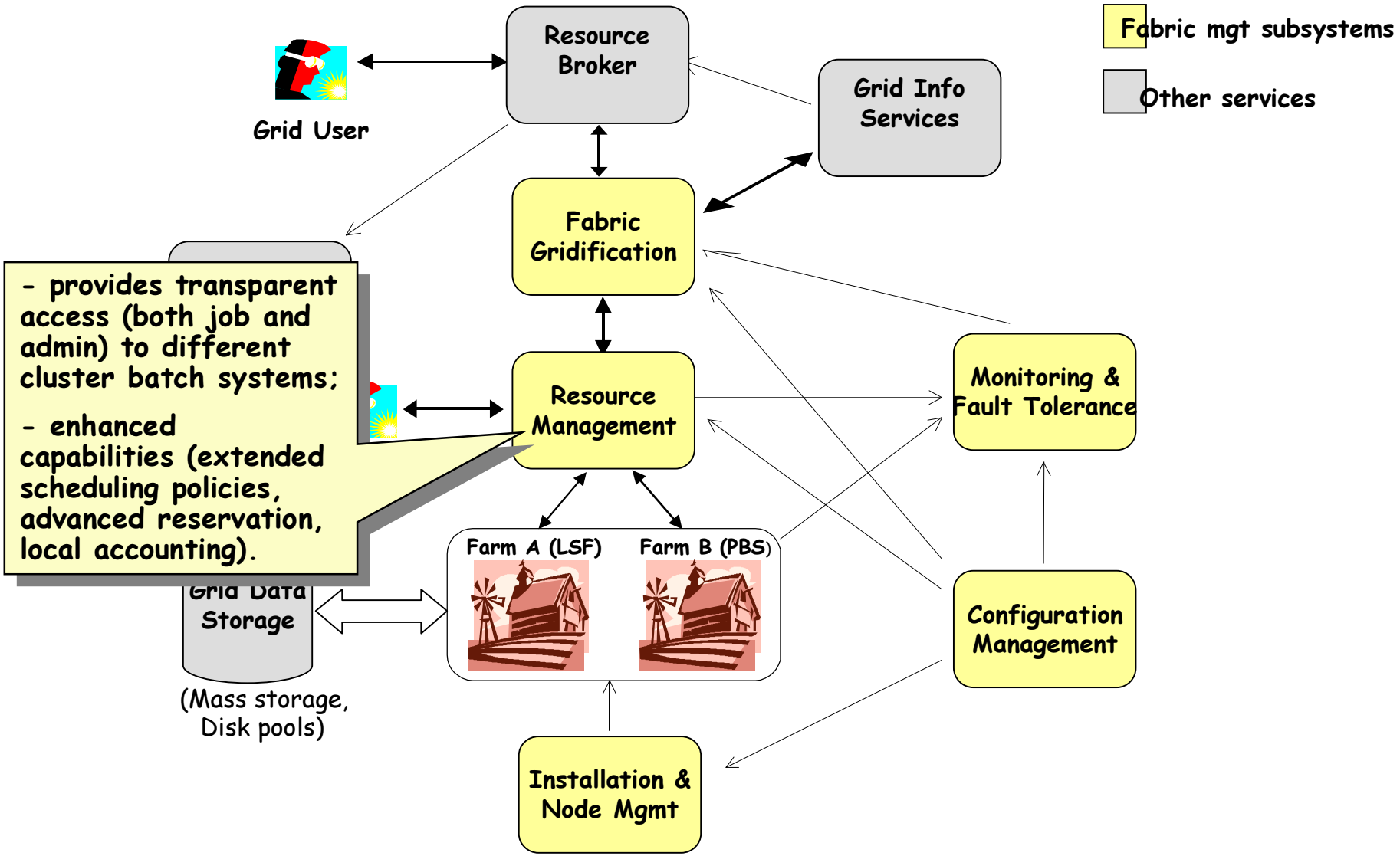
Architecture logical overview



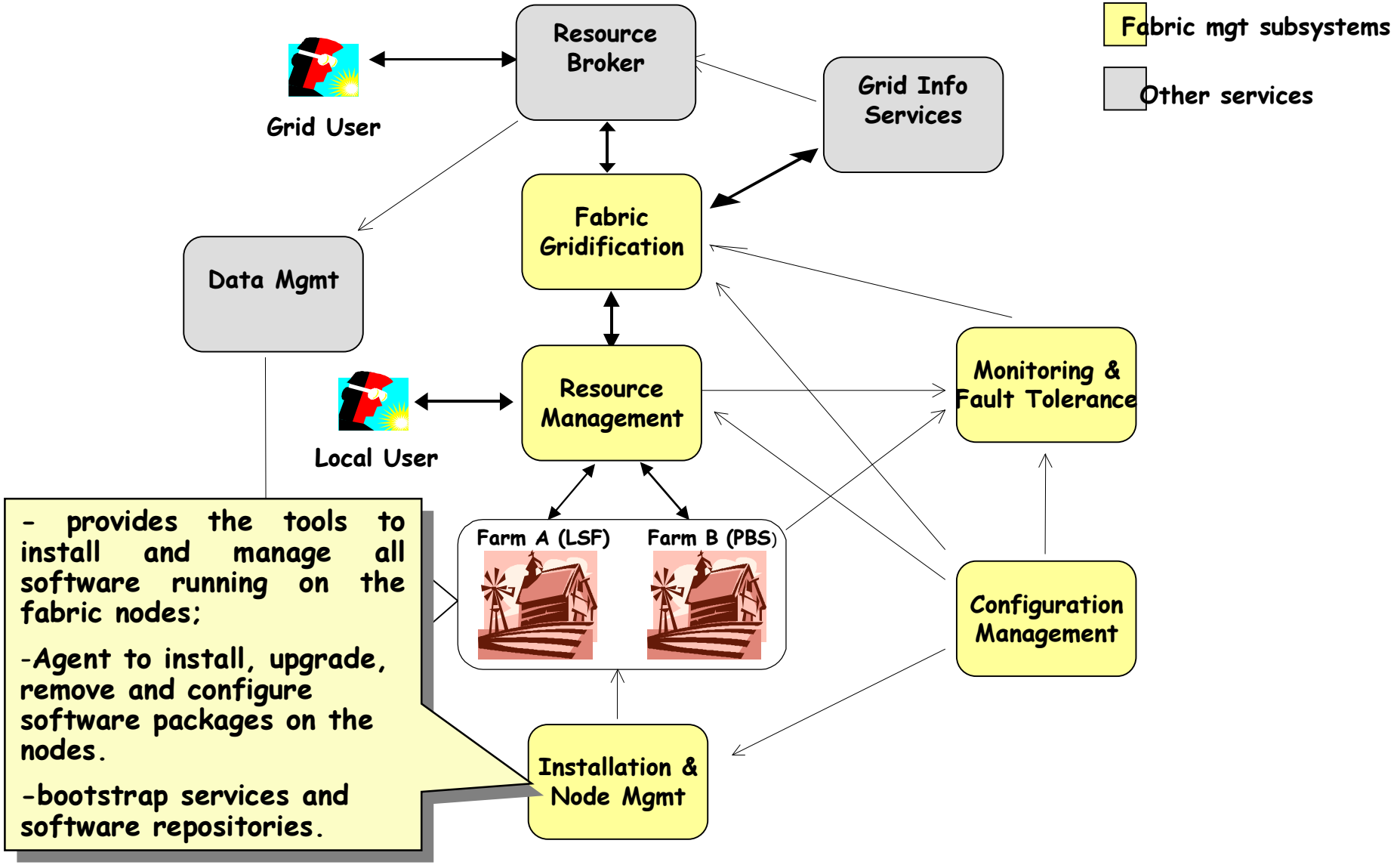
Architecture logical overview



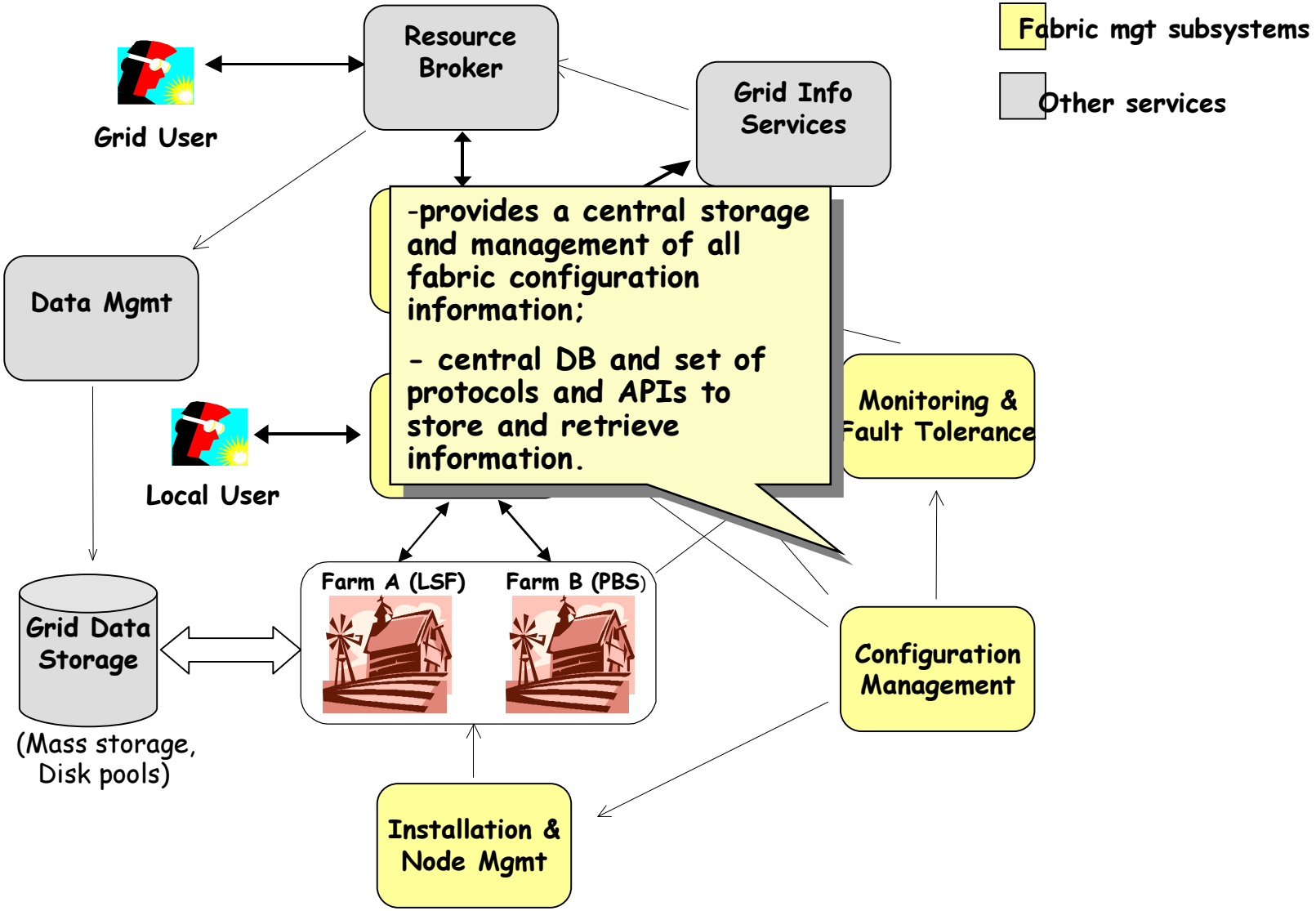
Architecture logical overview



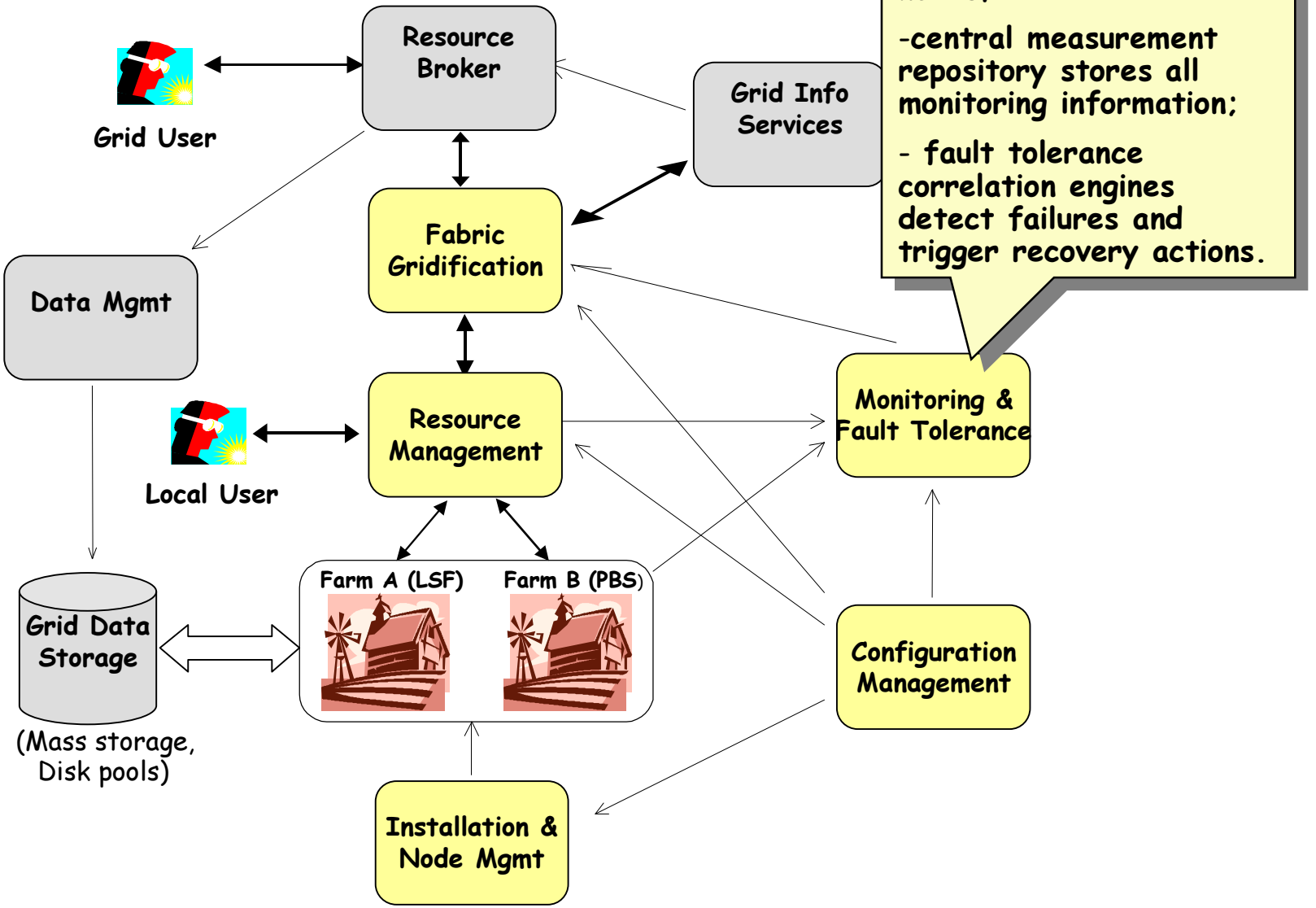
Architecture logical overview



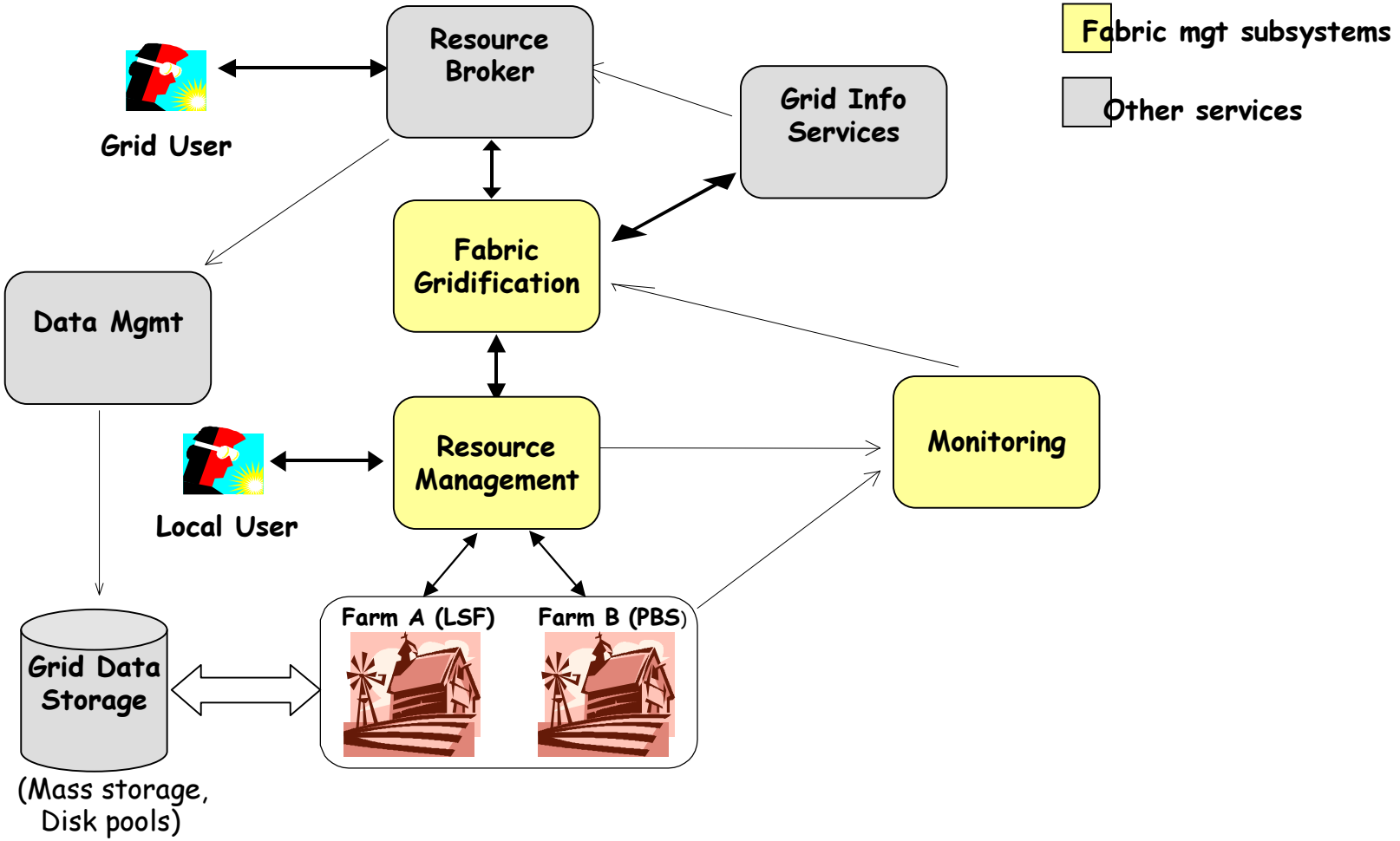
Architecture logical overview



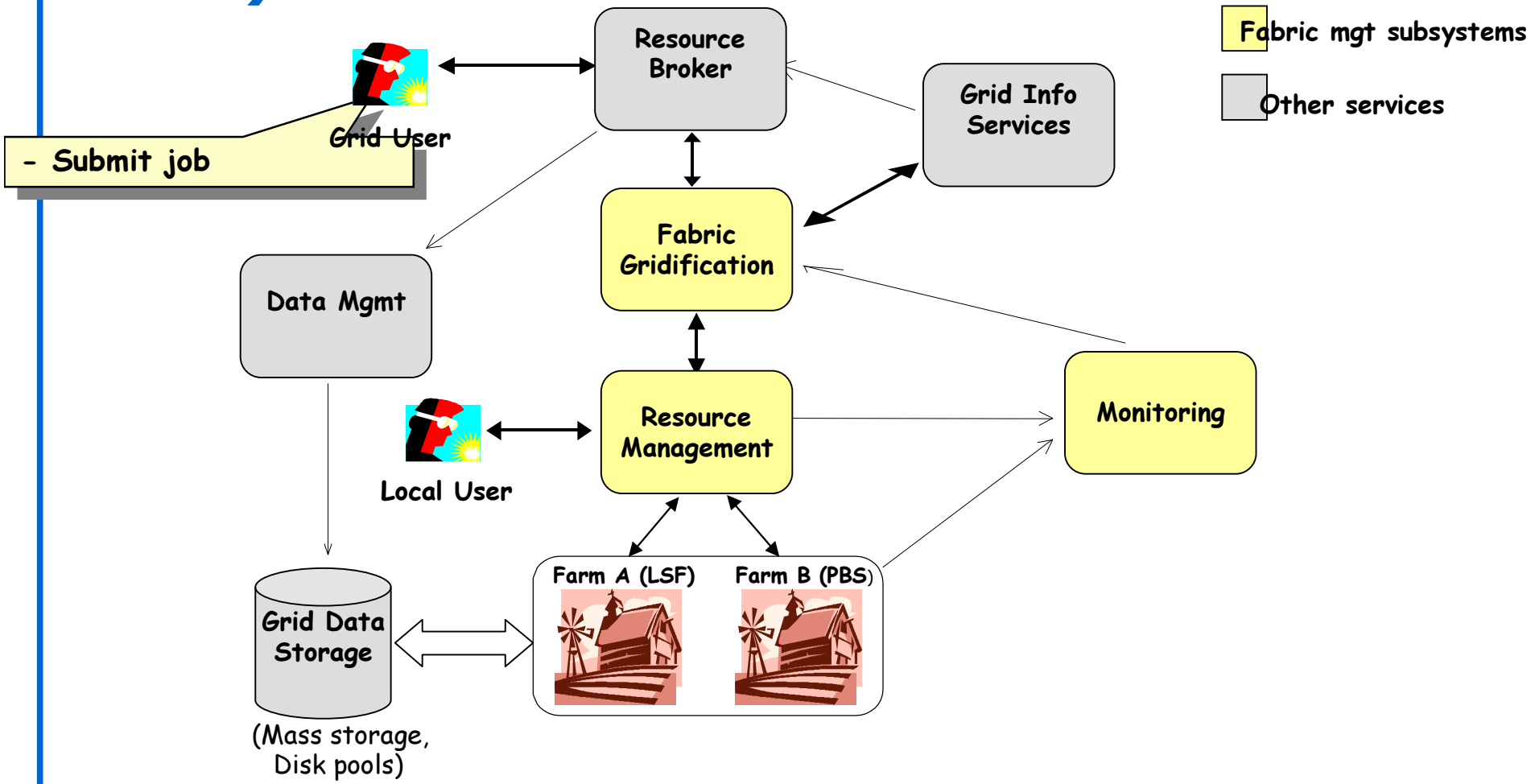
Architecture logical overview



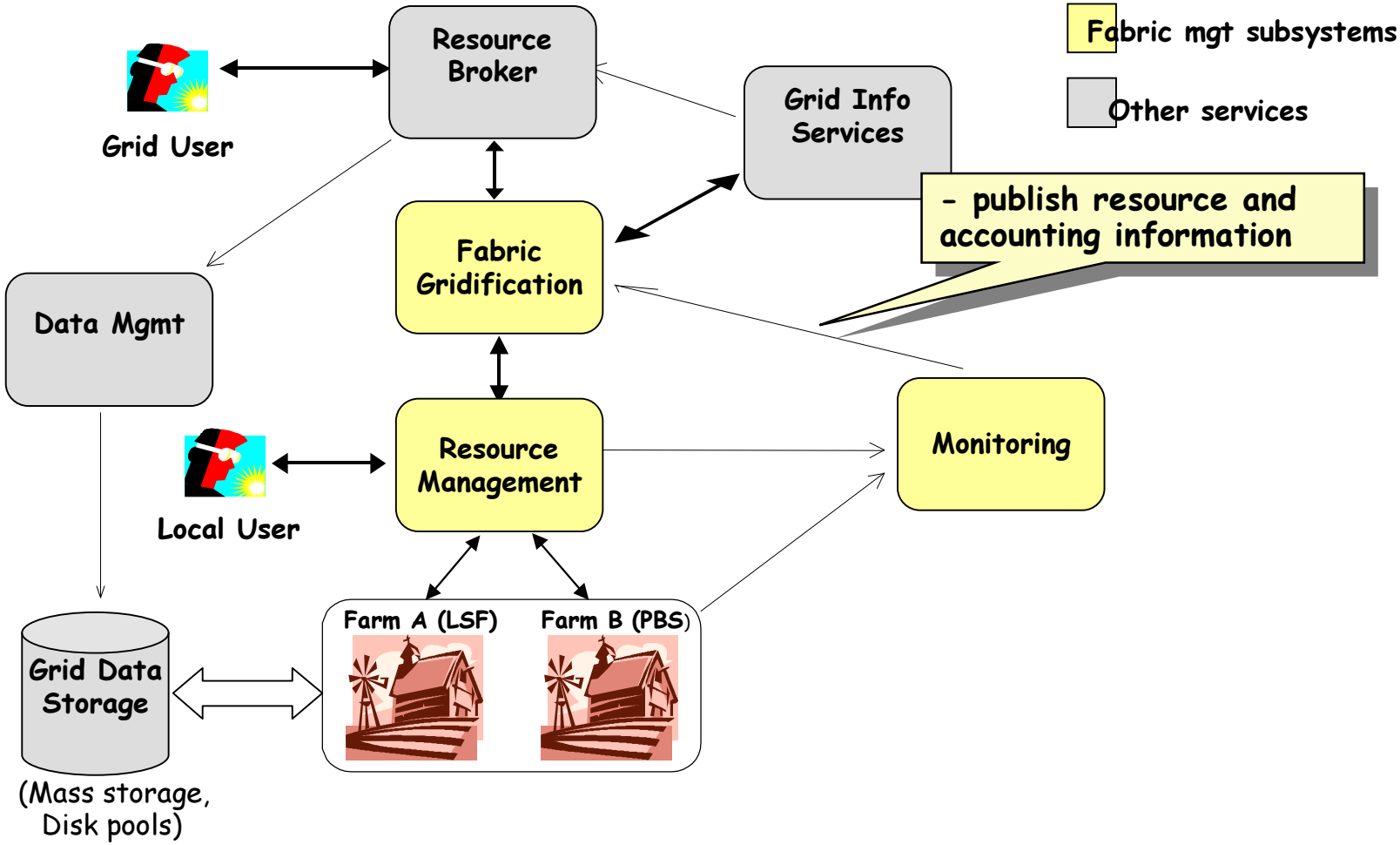
User job management (Grid and local)



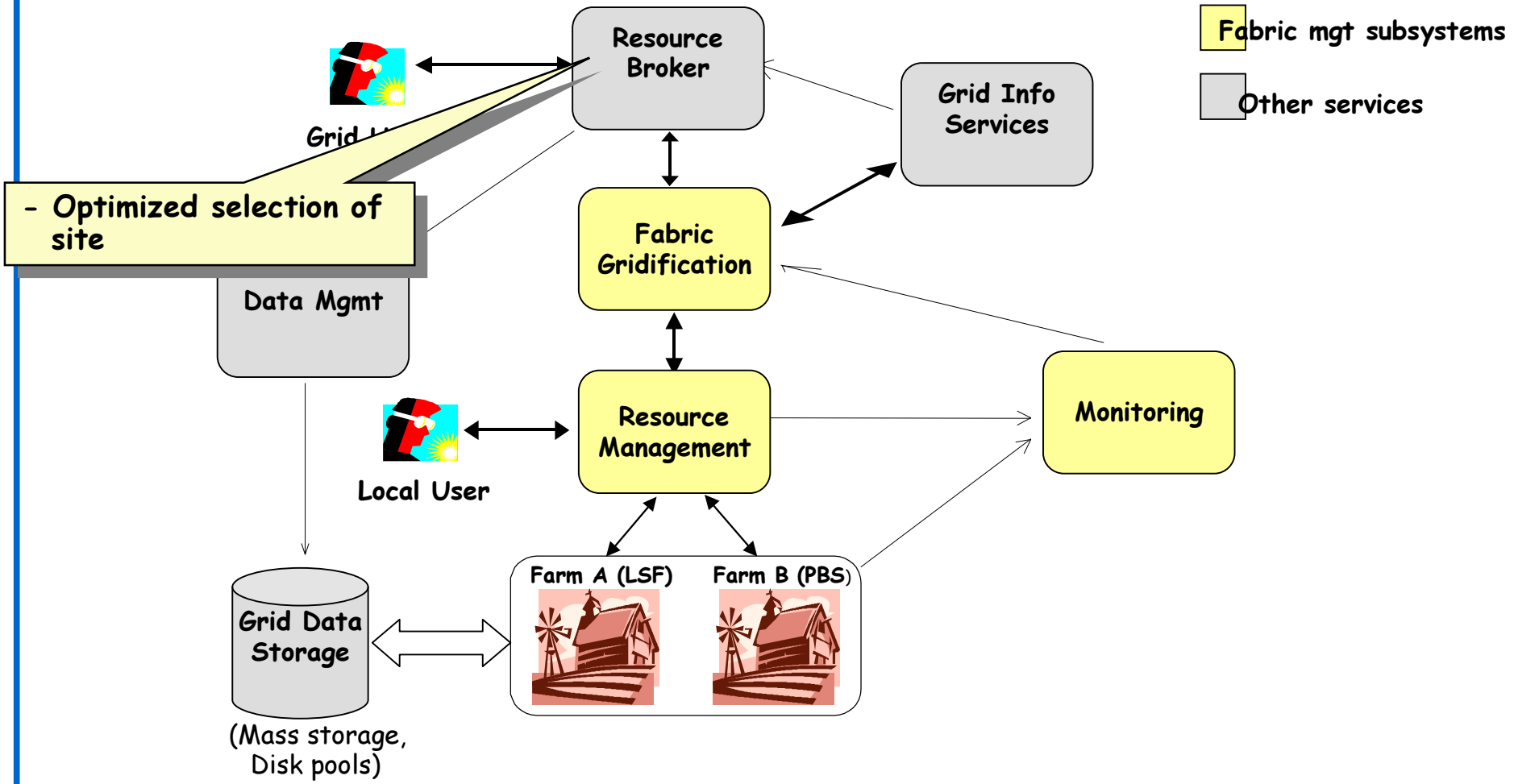
User job management (Grid and local)



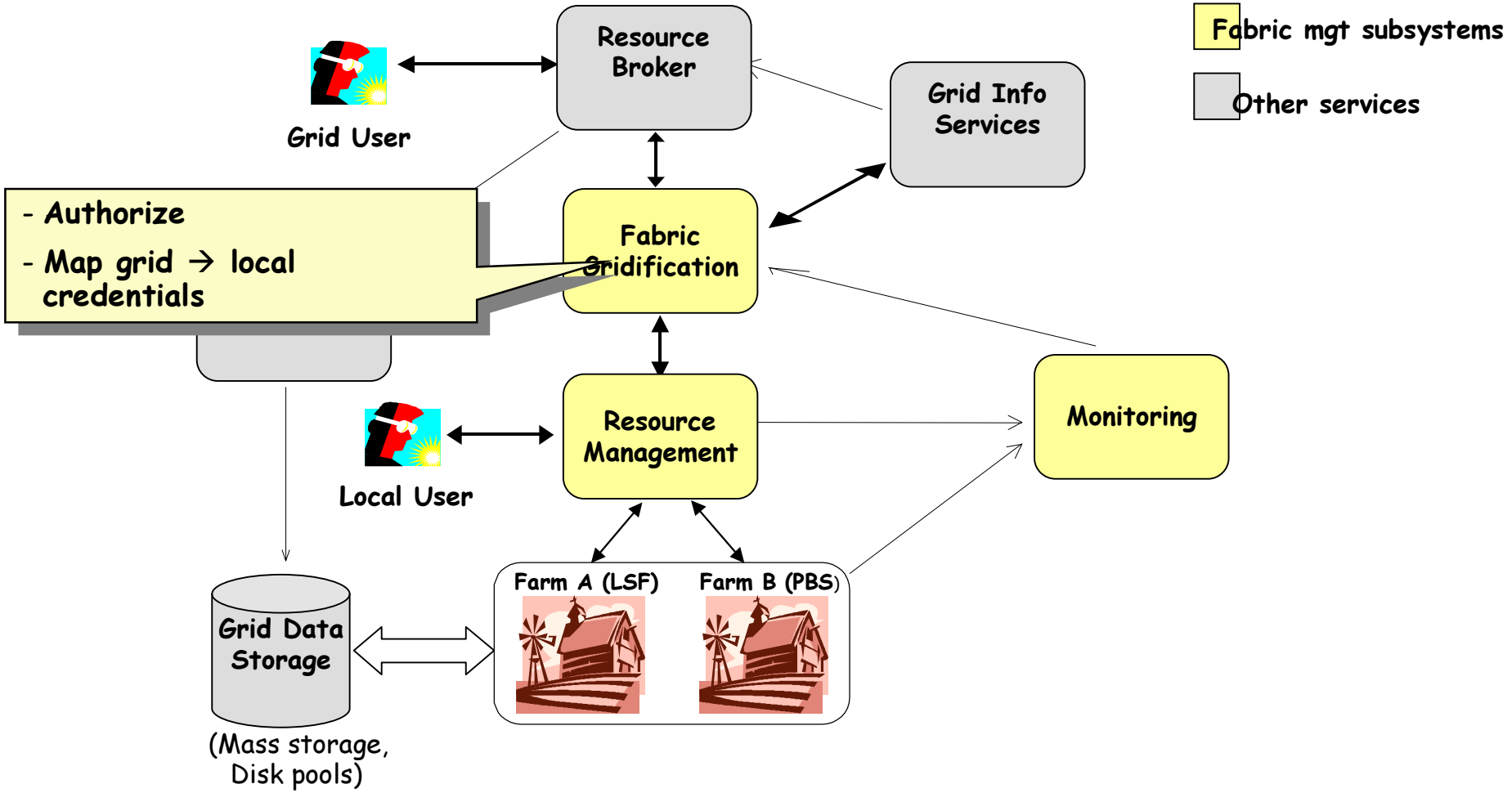
User job management (Grid and local)



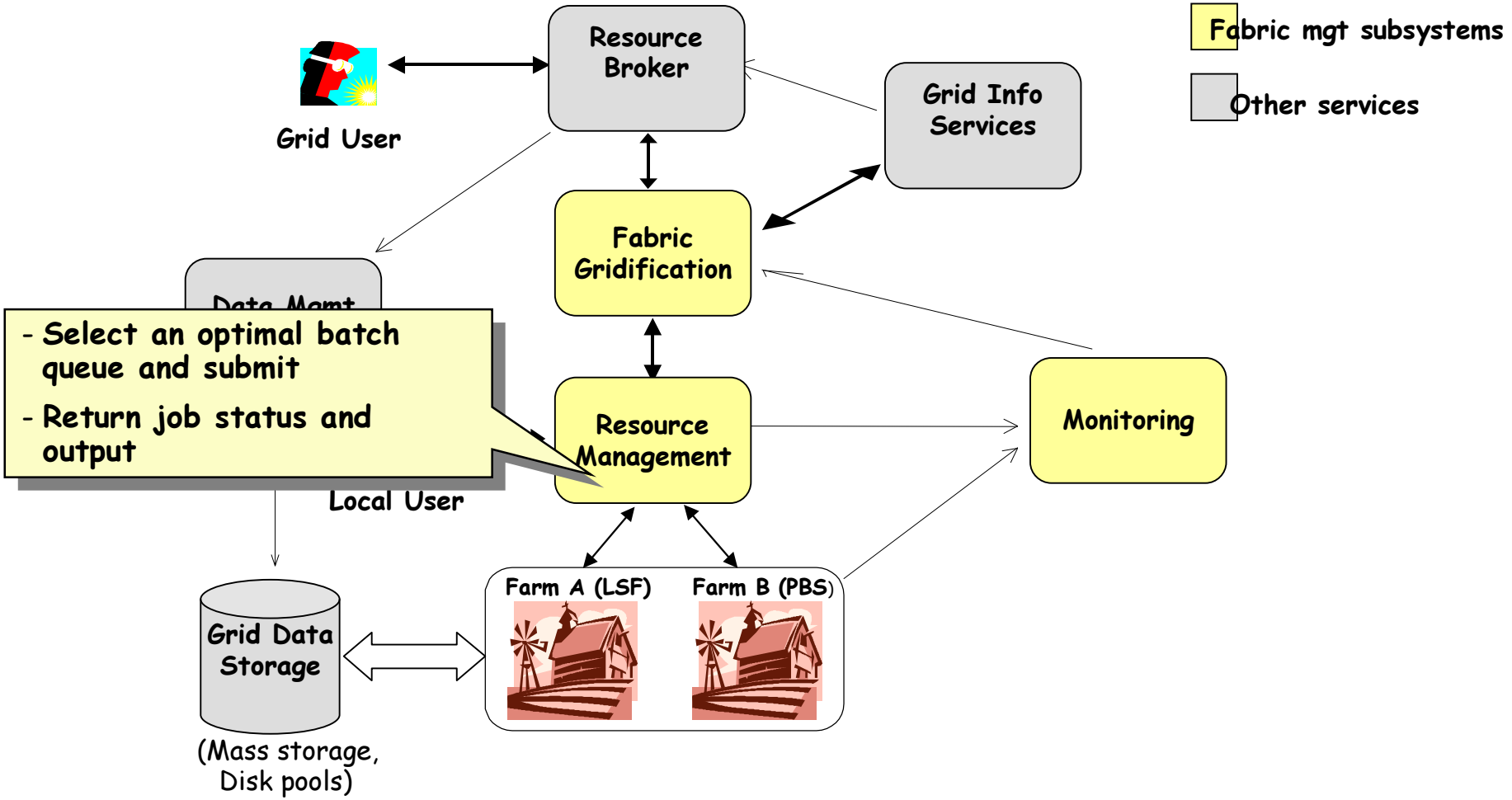
User job management (Grid and local)



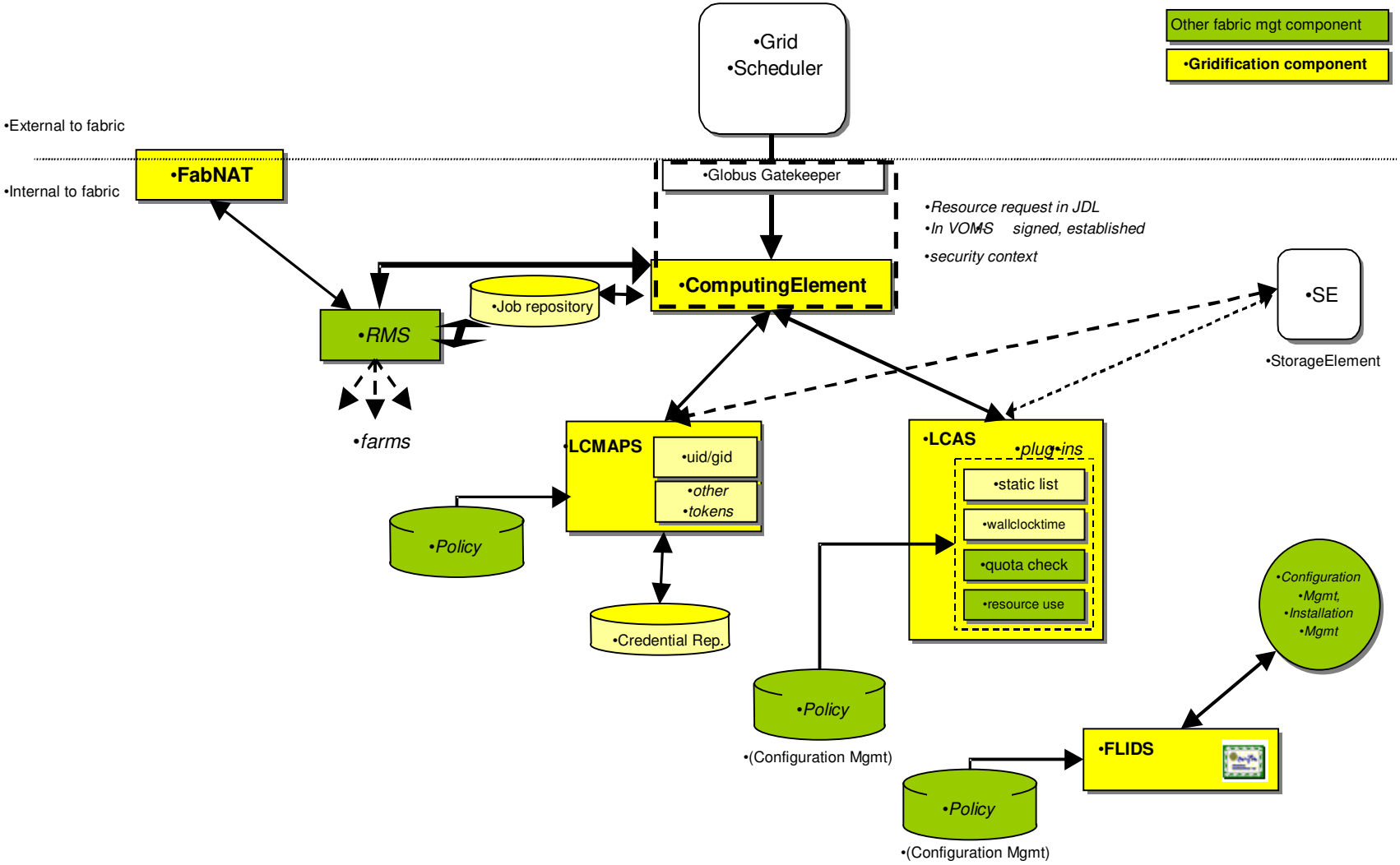
User job management (Grid and local)



User job management (Grid and local)



Gridification Architecture



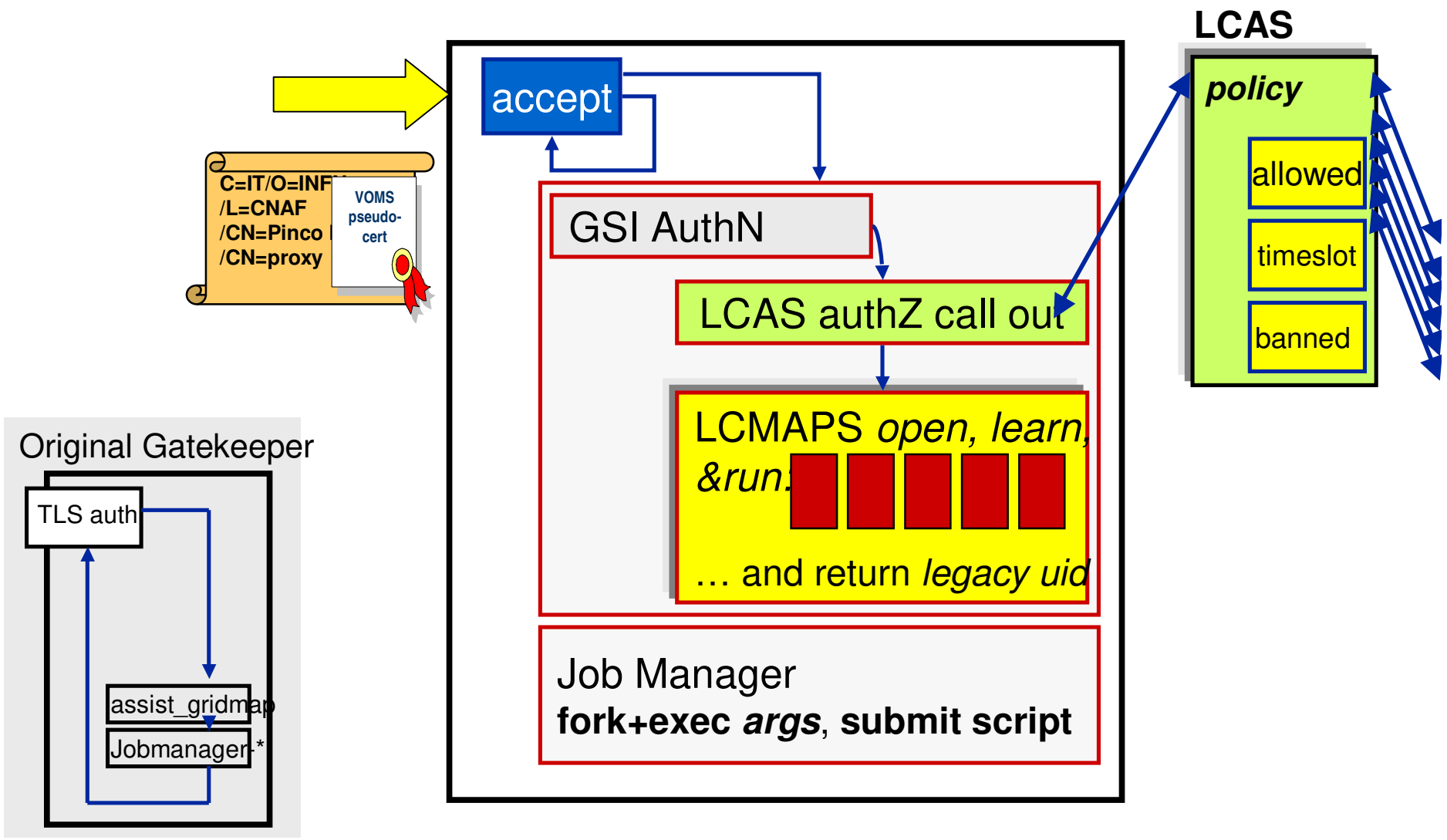
Fabric Management @ Release 2.0

- Gridification:
  - LCAS + edg_gatekeeper
- Monitoring
  - FMON
- Resource Management
  - RMS
- Installation and configuration:
  - LCFG (Local ConFiGuration system)

LCAS 1.0 (EDG release 1.2)

- The Local Centre Authorization Service (LCAS) handles authorization requests to the local computing fabric.
- Authorization plugin framework, library.
- The authorization decision of the LCAS is based upon the users' proxy 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



RMS architecture

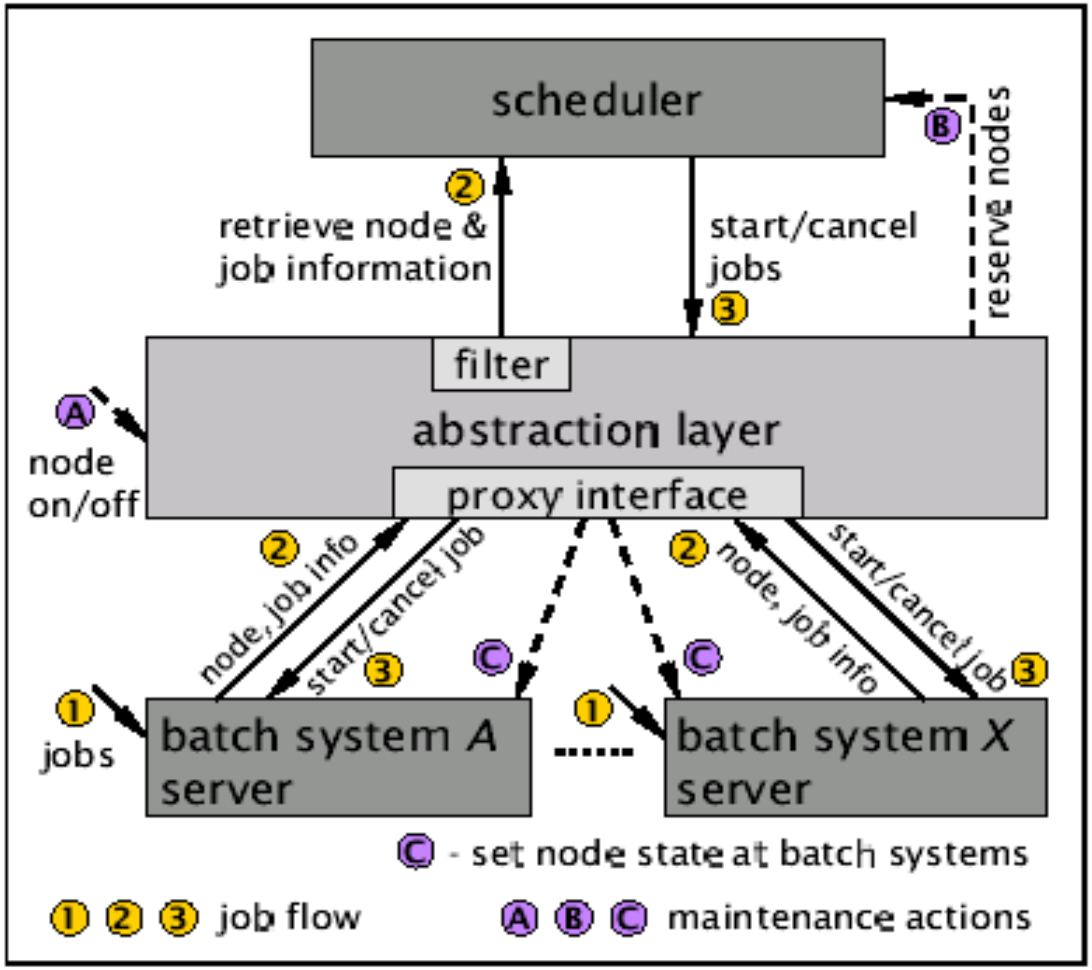
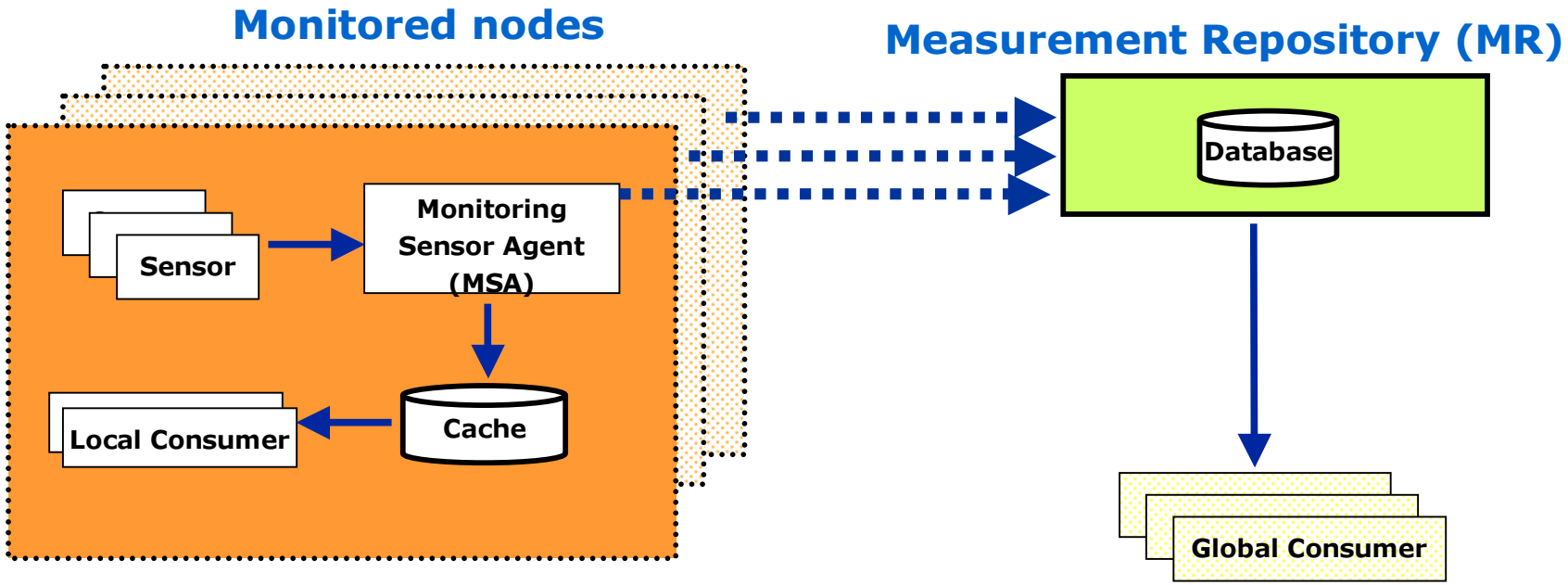


Figure 1: Architecture of the resource management framework

Fabric Monitoring architecture



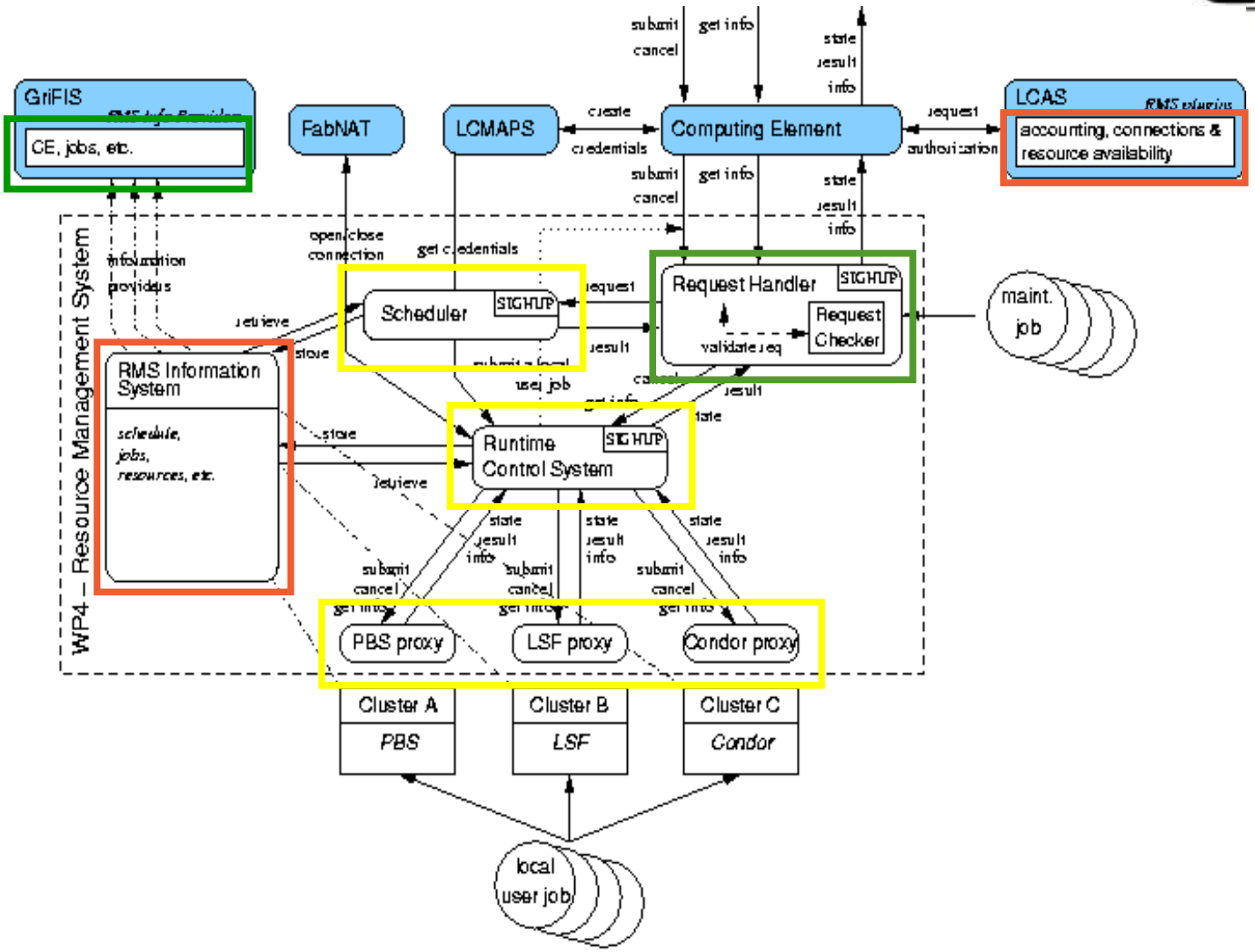
Monitored nodes

- Sensors measure Metrics mainly locally: CPU utilization, network throughput, daemons status, etc.
- MSA – Monitoring Sensor Agent:
  - collects sensors data, stores them in a local cache and sends them to the central repository
  - triggers measurements according to the configured schedule.
- Local Cache: allows local consumers to have fast access to monitoring information. Data can be collected even if the node istemporally isolated from central repository.
- Outgoing data consist of a value associated to a metric identifier (what), a timestamp (when), a target identifier (where), and an agent identifier (who).

Measurement Repository

- Receives samples from all the nodes
- Stores data:
  - Plain text DB
  - Oracle DB
- Follows data up to subscribers
- Answers queries
- information
- Consumers can subscribe to metrics and receive notification when new samples are available

RMS - Resource Management System



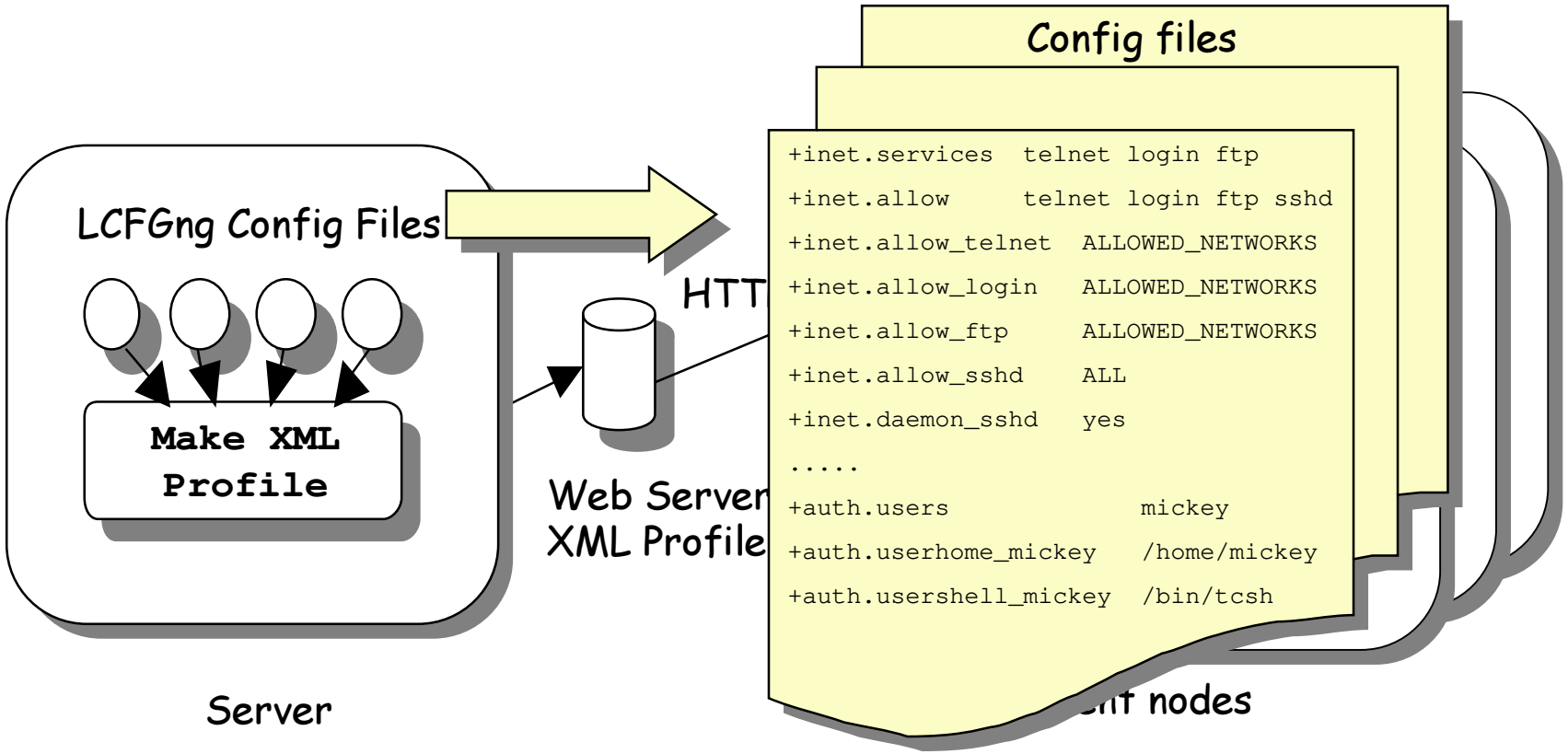
RMS functionality

- The Resource Management System (RMS) is responsible for the management of local user jobs, grid user jobs and maintenance jobs within a site, that is build of one to many clusters.
- It enhances capabilities of the Globus Toolkit job management that submits jobs to a specific resource.
- The enhancements enabled by the RMS include:
  - advanced scheduling mechanisms, e.g. backfill, fair share
  - support for advance reservations
  - support for maintenance tasks, e.g. node on/off.
  - Support to the EDG grid accounting infrastructure

LCFG (Local ConFiGuration system)

- LCFG is originally developed by the Computer Science Department of Edinburgh University
- Widely used fabric management tool, whose purpose is to handle automated installation and configuration in a very diverse and evolving environment
- 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
- LCFGng: updated version of LCFG, customized to EDG needs. This is the version used at EDG 2.0

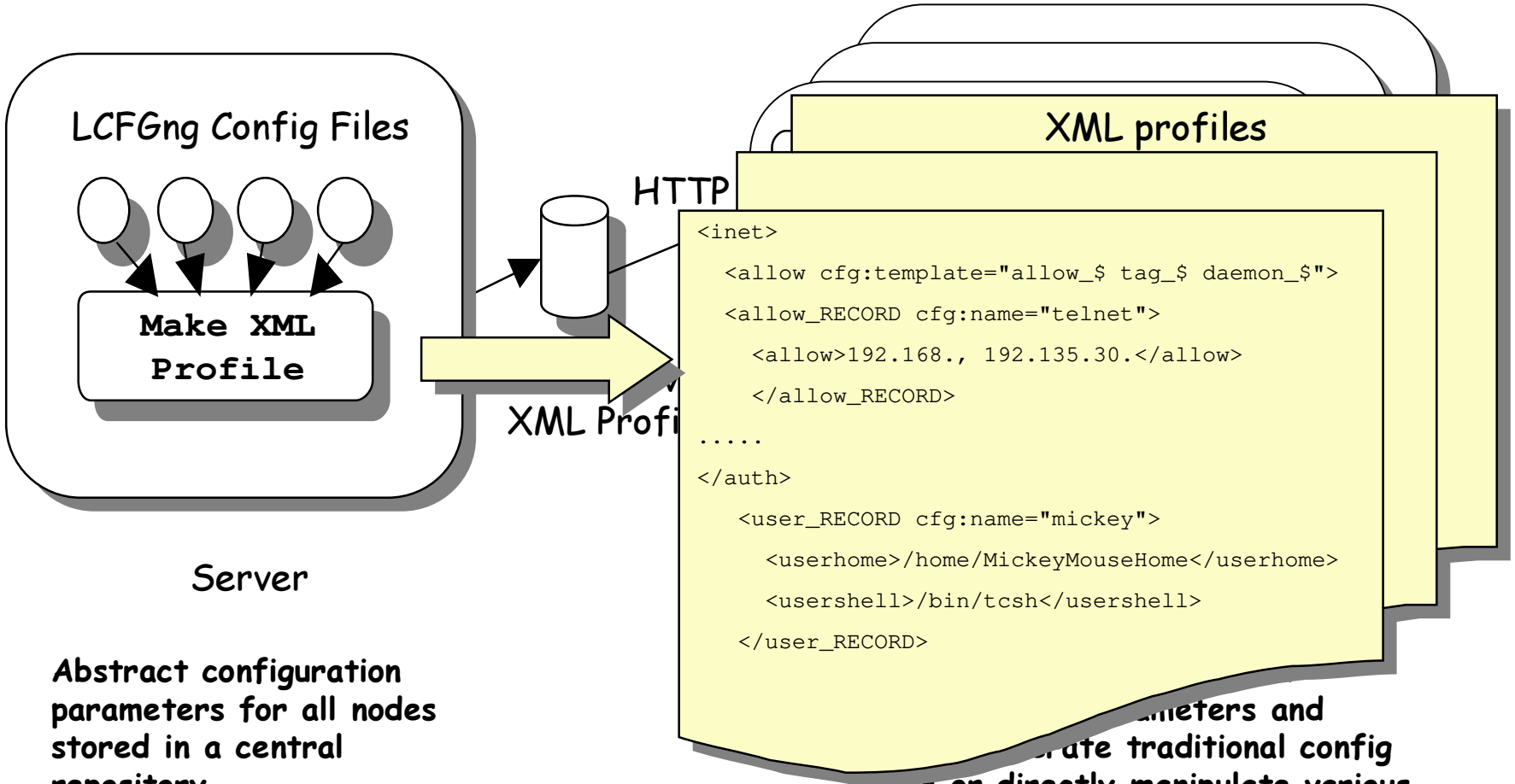
LCFGng system architecture



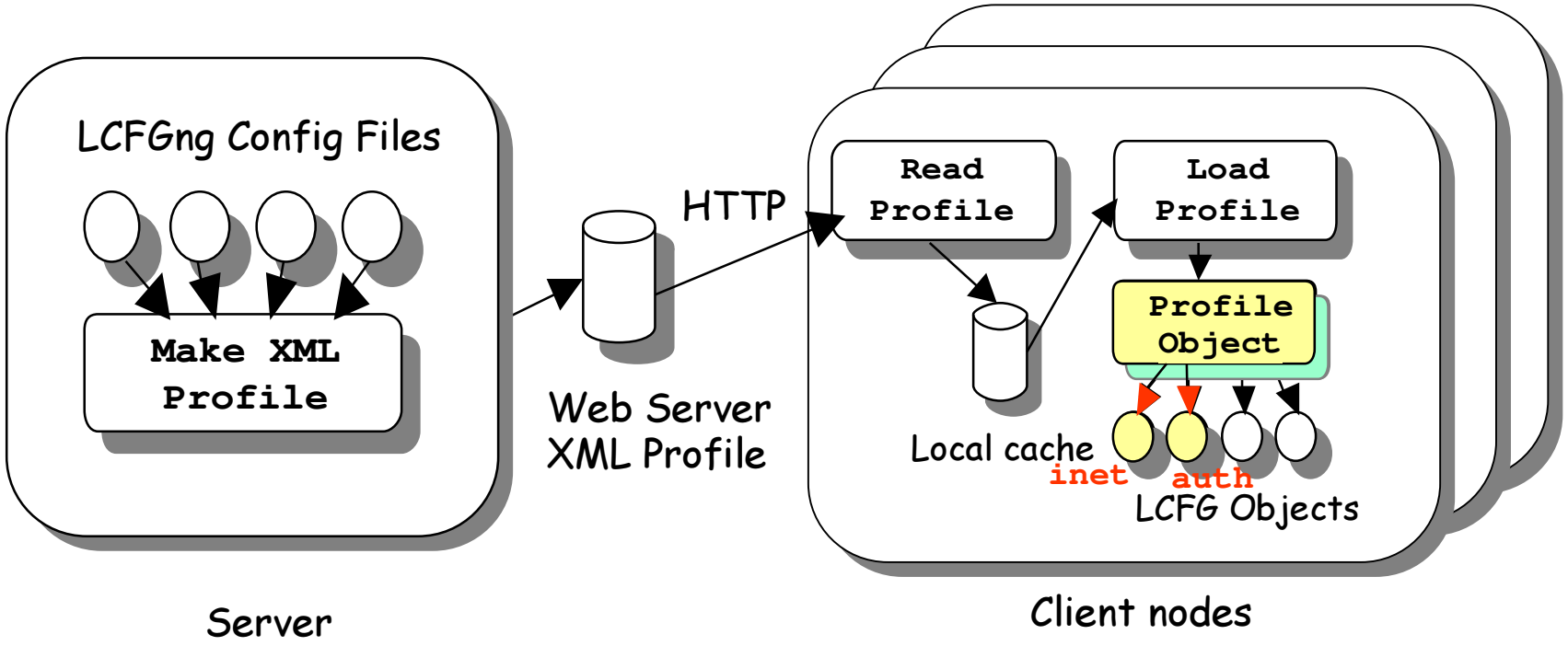
Abstract configuration parameters for all nodes stored in a central repository

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

LCFGng system architecture



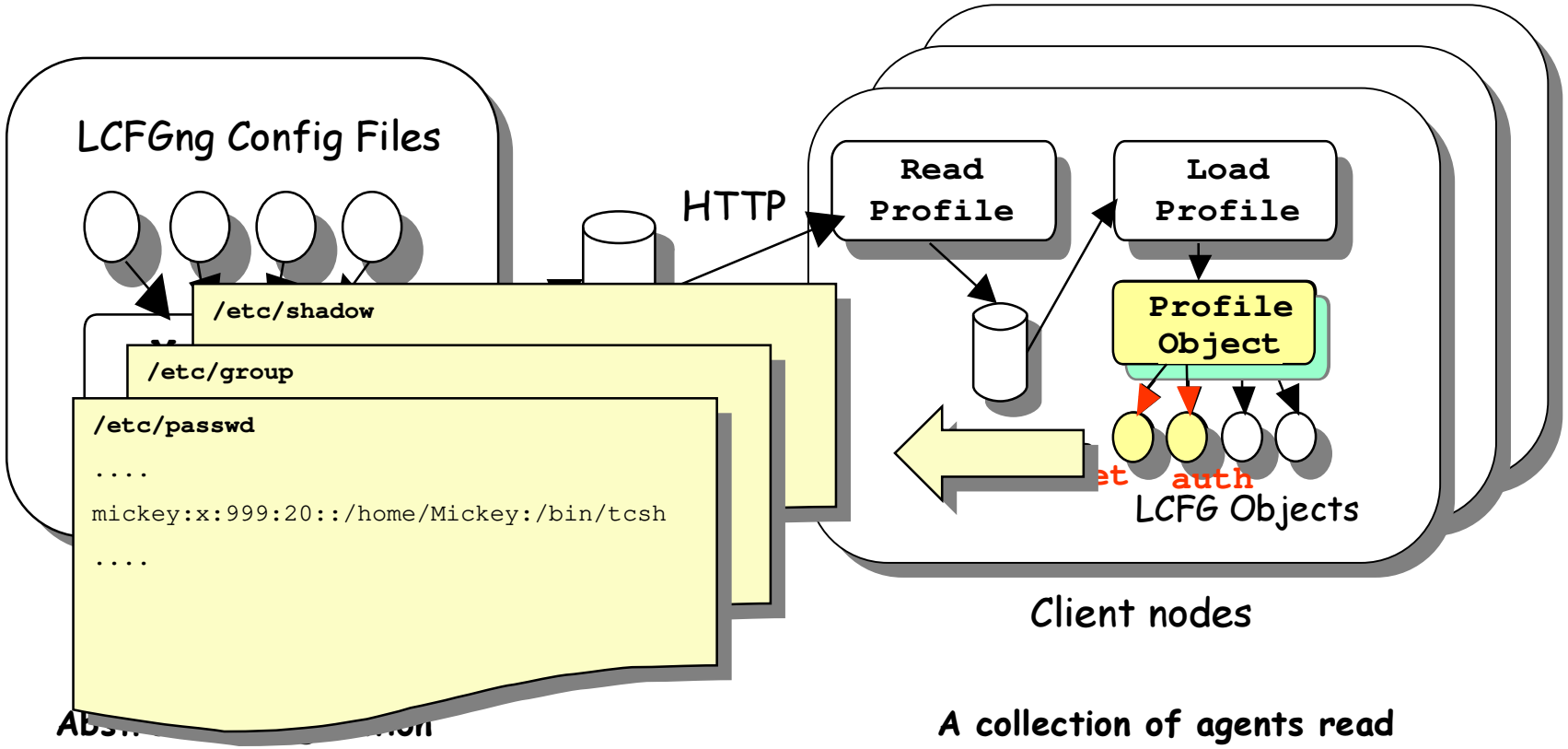
LCFGng system architecture



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

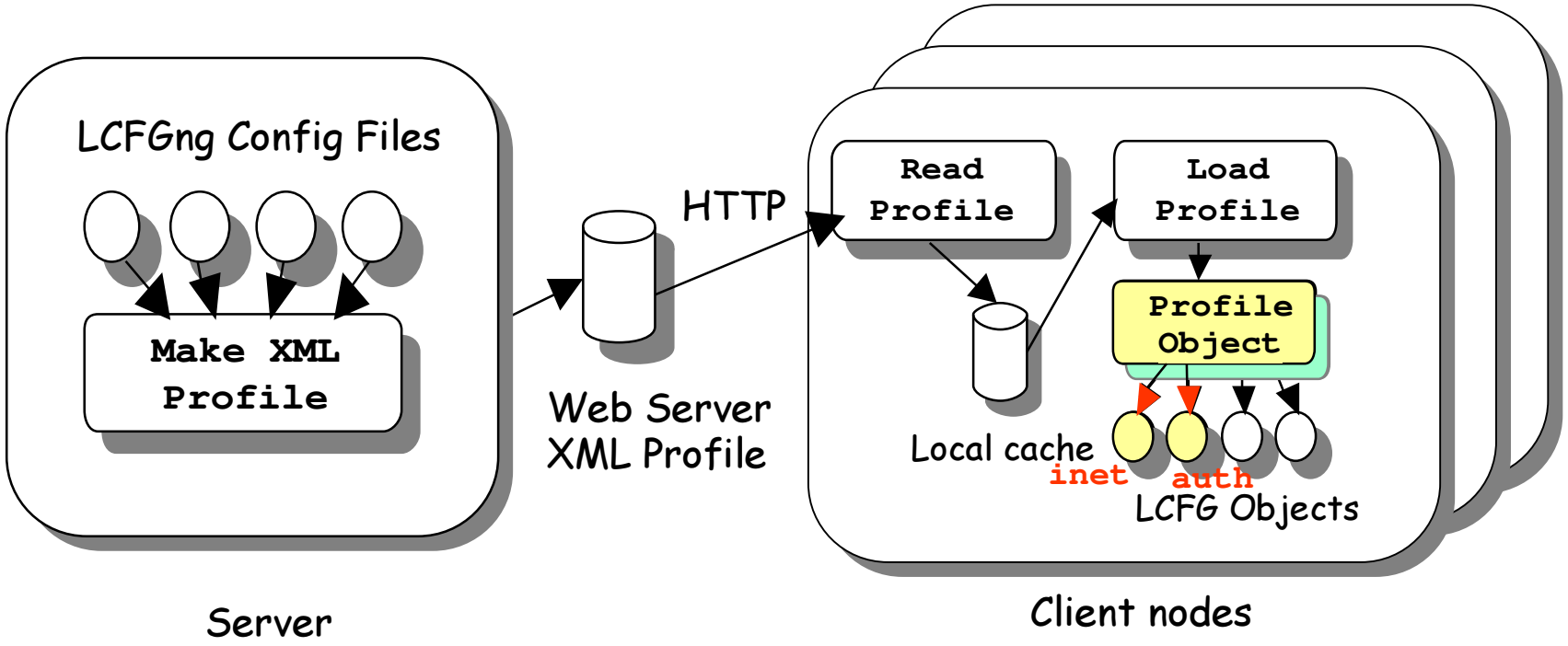
LCFGng system architecture



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

LCFGng 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

LCFGng: what's a component?

- It's a simple Perl script
- Each component provides a Configure() method invoked on startup or when configuration changes
- A simple and typical component behavior:
  - 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).
- LCFGng 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

LCFGng component template

```
#!/usr/bin/perl-w

package LCFG::MyComp;

@ISA = qw(LCFG::Component);


use strict;

use LCFG::Component;

use LCFG::Config; # EDG specific


sub Configure($$@) {

  my ($self,$res) = @_;

  my $config=LCFG::Config->new($res);

  my $age=$config->getValue('/persons/John/age');

  $self->Fail( "too young") unless ($age>18);

}

new LCFG::MyComp() -> Dispatch();
```

LCFGng component template (II)

- `Configure()` method:

```
# Do the configuration

sub Configure($$@) {

  my ($self,$res) = @_;

  my $config=LCFG::Config->new($res);

  # do whatever is necessary to reconfigure your
 service

}
```

LCFGng Component Example (I)

Edg-lcfg-syslog – configures /etc/syslog.conf

.def file:

```
@additions add_$

Additions

add_$
```

Resources defined on server:

```
syslog.additions                monitoring kernel

syslog.add_monitoring           local3.*
  |/var/obj/tmp/monitor.fifo

syslog.add_kernel               kern.*     /var/log/kernel.log
```

LCFGng Component Example (II)

Component: Configure() method. First part: template generation

```
sub Configure($$@) {

   my ($self,$res,@args)=@_;
    my $config=LCFG::Config->new($res);

    my $syslogconf='/etc/syslog.conf';

    my $status = LCFG::Template::Substitute
      ( '/usr/lib/lcfg/conf/syslog/template',
        '/var/obj/conf/syslog/config', 0, $res );

    unless (defined($status)) {
      $self->LogMessage($@);
      $self->Fail( "failed to create config file (see
 logfile)");
    }
```

LCFGng Component Example(III)

Component: Configure() method. Second part: add missing resources

```
    #extra values from 'additions', using NVA API

   my $additions=$config->getElement('/additions');
   my @add_array=();
   while ($additions->isNextElement()) {
     my $add=$additions->getNextElement();
     my $add_el=$config>getElement('/additions/'.$add.'/add');
     push(@add_array,$add_el->getValue());
   }


   if (scalar (@add_array)) {
     open (CFG, '>>/var/obj/conf/syslog/config') ||
       $self->Fail('cannot open config file (see
logfile)');
     print CFG join("\n",@add_array)."\n";
     close (CFG) ||
       $self->Fail('cannot close config file (see
logfile)');
   }
```

LCFGng Component Example(IV)

Component: Configure() method. Third part: copy config file, restart service

```
  # copy over to definitive location if files are different

  if (system("/usr/bin/cmp -s $syslogconf
/var/obj/conf/syslog/config")){
    $self->LogMessage("updating config file");

    system('cp -f '.$syslogconf.' '.$syslogconf.'.old');
    system('cp -f /var/obj/conf/syslog/config '.$syslogconf)
      && $self->Fail("copying template to $syslogconf: ". $?);

    # restart service
    if (system('/sbin/service syslog restart')) {
      $self->Fail('init.d syslog restart failed: '. $?);
    }
  }

  return 1; #OK

  }
```

Software distribution with LCFGng

- It is done with an LCFGng tool called updaterpms.
- The standard system packaging format is used: `rpm` for Linux
- It is managed via an LCFGng 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)

Summary

- Fabric management components deployed on release 2.0:
- 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.
- Fabric monitoring functionality:
  - FMON: provides a client (MSA) running sensors on each node to monitor, and a central server to collect data.
- Resource Management functionality:
  - RMS: management of local user jobs, grid user jobs and maintenance jobs within a site, built from one to many clusters