SCAS

Site Central Authorization Service

Oscar Koeroo by proxy of Gerben Venekamp
JRA1

index

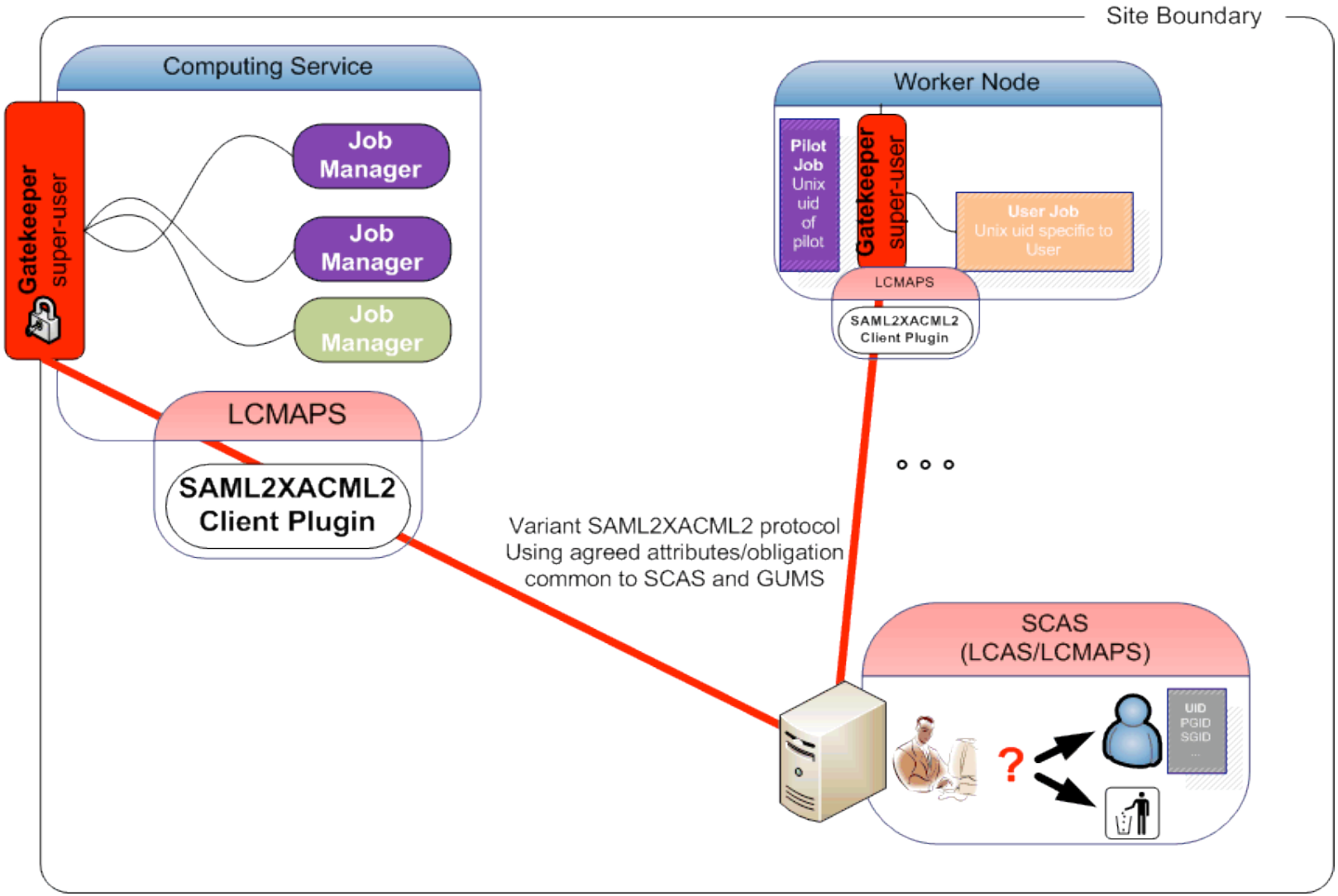
- **What's SCAS**
  - What is and what isn't SCAS?
  - Why this protocol particularly?
- **Interoperable components**
- **The implementation**
  - About the Request and Response messages
- **Performance**
- **Planning**

What is SCAS?

The Site Central Authorization Service

- **It implements a client/server architecture to query:**
  - Authorization decisions (LCAS), allow/ban:
    - From a trusted resource
    - From an authorized pilot job executor
    - For an authorized user
  - Centralized grid identity to Unix ID translations (LCMAPS)
    - Full LCMAPS support
      - *VOMS pool and local accounts mappings*
      - *Non VOMS pool and local account mappings*
- **Uses mutual authenticated SSL/TLS**

Site Central Access Control



What isn't SCAS?

It's not a centralized authentication service (....yet)

... although the option is left open for future investigation

- Clients must authenticate credentials before it goes on the wire:
  - Requirement on clients:
    - *CA certificates and VOMS authorization files (.lsc)*

SAML-XACML lib requirements

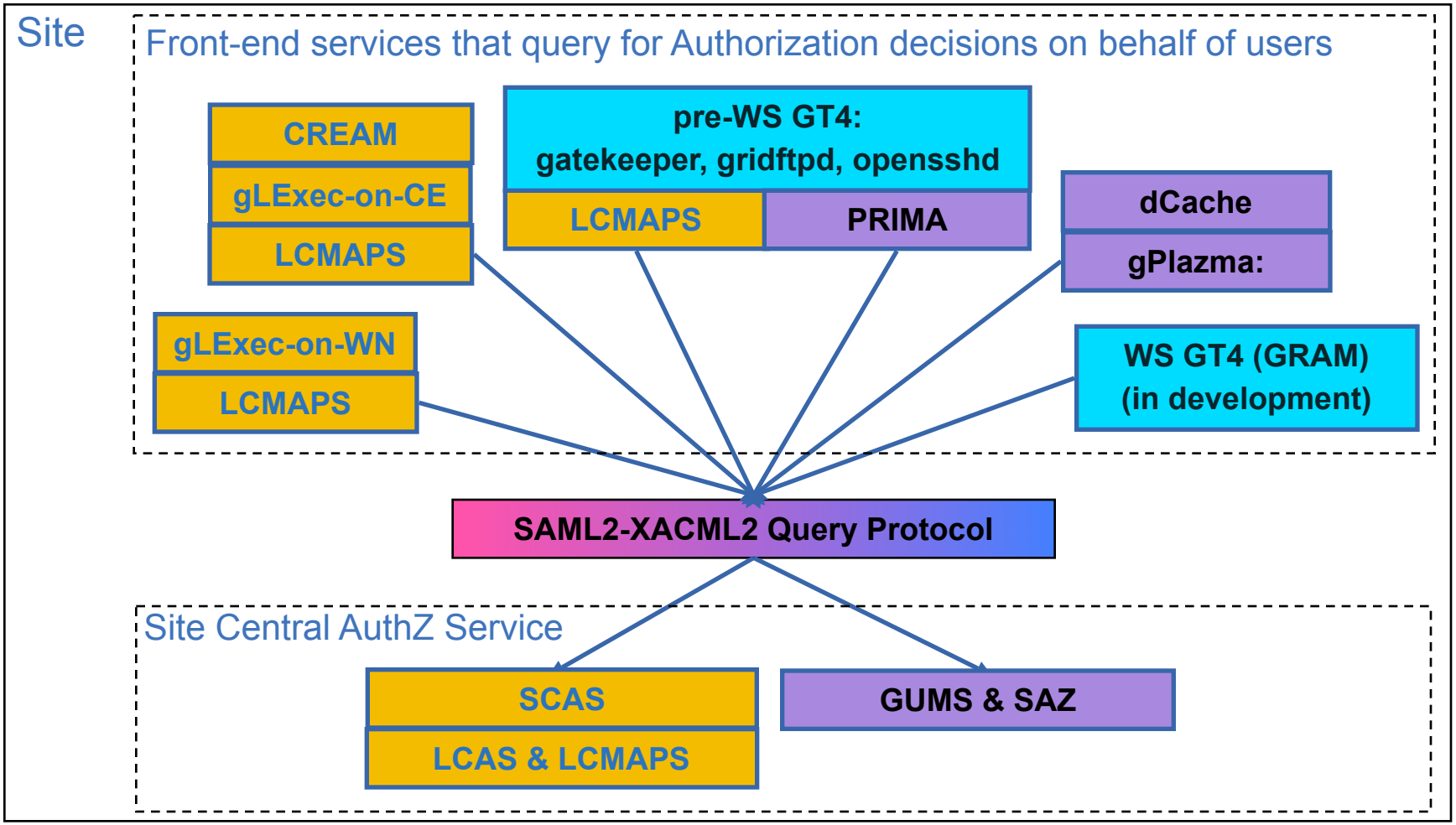
- **Requirements to SCAS dev:**
  - Easy interoperation
    - Understand a common set of obligations and its attributes
  - Scalability
    - Low network traffic
    - Low overhead at the end points
  - Keeping compatibility with existing LCAS and LCMAPS plug-ins and their functionalities

- **Requirements to Globus:**
  - Must be separate from the existing Globus Toolkit (low dependency overhead)

Interoperable components

Legenda: Color code indicates component developers:

Globus	EGEE	OSG / Privilege Project	Globus, EGEE, OSG / Privilege Project



The Query Protocol: Request (0)

- **Setups up SSL/TLS connection to authz service:**
  - SCAS:
    - From CE/SE: Host credential
    - From WN (pilot job): Pilot job executor credential
  - GUMS
    - Using host credential in all cases (CE/SE/WN)

The Query Protocol: Request (1)

- **Request message payload to authz service:**
  - Subject
    - User info for who is the authorization request
  - Action
    - Send a queue-able job, execute now (fork/glexec) or accessfile
  - Resource
    - From which (type of) node (WN, CE, SE, RB) and host id (if avail.)
  - Environment
    - Advertise PEP supported obligation handlers
    - Job invoker (replicate subject) and type (could be unprivileged Condor daemon or pilot)

The Query Protocol: Response

- **Basic: Yes/No**

- **EGEE Obligations:**
  - UID + GID
  - Optional multiple secondary GIDs
  - Optional AFS token (type string)

Performance

- **Part of the Nikhef cluster to one old service machine:**
  - Lots of dual-quad xeons for clients
  - One old dual Opteron for the SCAS service
- **Goals for the service:**
  - Stability and 6Hz nominal rate authz decisions and mappings

	# clients	# cum. server load	# cum. server peak	# Mappings in Hz
1 SCAS daemon	15	10%	13%	~24Hz
1 SCAS daemon	30	15%	18%	~31Hz
2 SCAS daemon	30	20%	25%	~42Hz
2 SCAS daemons	60	40%	48%	~66Hz
4 SCAS daemons	60	60%	65%	~80Hz

Final words

- **The site central solution allows for improved emergency response**
  - Central blacklist
  - Consistent mappings across a cluster or a site for all the supported services

- **Profiled document on the used attributes:**
  - "An XACML Attribute and Obligation Profile for Authorization Interoperability in Grids"
    - https://edms.cern.ch/document/929867/2

Planning

- **Patch #2511: SCAS client and service:**
  - Finally found the bug in the SAML2-XACML2 C library
    - "You shall prefix the endpoint with "**http{s}://**"
    - New SAML2-XACML2 C library version is more careful
- **In the meanwhile:**
  - Fixing memory leakage and thread safety
  - Attending patch #2381 (glexec install/config)
    - Open issue: logging-only mode

?