Clarens Web Services Architecture

CHEP 2003 March 24 - 28,

Grid Architecture, Infrastructure & Middleware

Conrad Steenberg, Eric Aslakson, Julian Bunn, Harvey Newman

California Institute of Technology

Developed as part of the

Particle Physics DataGrid

Overview

- Client/server architecture (short)
  - 2 implementations
- Modularity
- Scalabilty and fault tolerance
- Security and Virtual Organizations
- Persistency and session management

Client/Server Architecture

- Web services builds upon the ubiquity of web technology:
  - Servers, clients, standards, large body of developers
- Marhals data using XML
  - Clarens NOT limited to XML, e.g. *file* module
  - Uses XML-RPC or SOAP
  - Other serialization layers can be used (e.g. *CORBA*)
- Main implementation is stateless
    - Based on *Apache* server with embedded *Python* interpreter
  - Smaller *pClarens* that is stateful, when needed

Client/Server Architecture II

- Clarens connects a client via a call routing mechanism to services
- Implements certain default services:
  - PKI-based security (authentication and authorization)
  - Service discovery
  - Persistent data store for session management
  - Logging
  - File publishing (RO with plans for R/W)

Modularity

- Services implemented via plug-ins
  - Location determines root of plugin-method name
    - e.g. system.* methods reside in system directory
    - Users can install modules under login directory, no system admin intervention needed. This can be disabled if needed!
- Plug-ins accessible without server restarts

{root}/system/__init__.py

system.auth
system.logout
system.*

/file/__init__.py

file.read
file.md5
file.*

/proxy/__init.py

proxy.store
proxy.retrieve
proxy.*

{home/user/clarens}/analysis/__init__.py

~user.analysis.init
~user.analysis.chi2
~user.analysis.*

/transform/__init.py

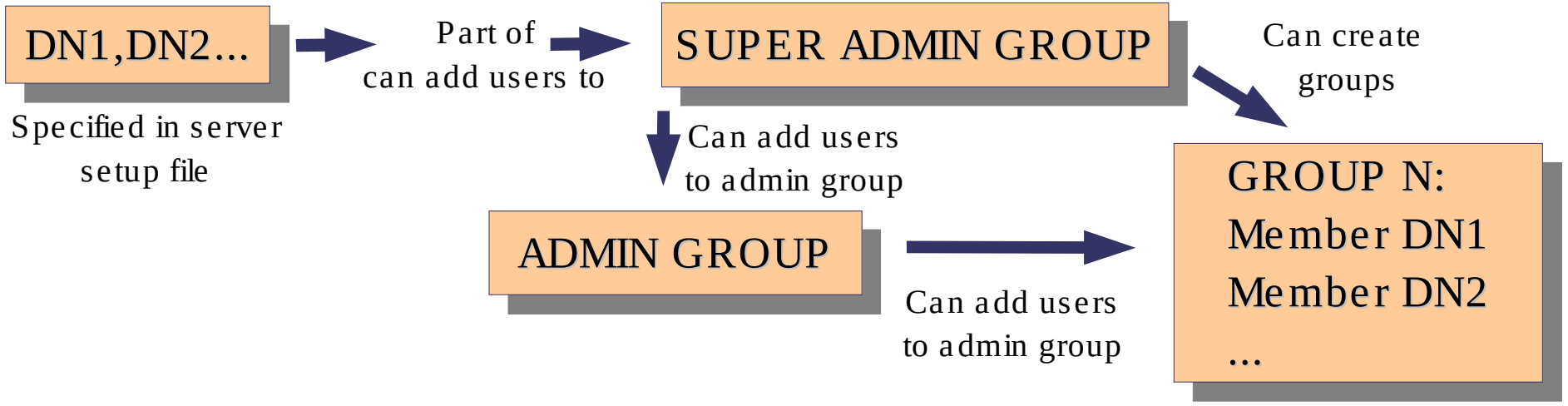
~user.transform.init
~user.transform.fft
~user.transform.*

Scalability and fault tolerance

- Each RPC is handled by own server process
  - Crashing module doesn't affect neighbours
  - Long-running requests does not block server
  - Leverages SMP when available
  - Server farm with load-balancing can appear as single virtual server
- Stateless protocol
  - Clients do not hold connection
- Session data stored in DB
  - Clients can survive server restarts, seen temporary server unavailability

Security and Virtual Organization

- Authentication via X509 certificates
  - Verifies certificate chain up to a list of accepted Certificate Authority certificates
  - Client identified internally by the certificate distinguished name (DN) – uniqueness ensured by CA
  - Authentication at the application layer
- Authorization done using an internal VO
  - VO consists of a hierarchy of groups and users
  - Does not need to store client certificates, uses Dns
  - VO data stored in DB



Security and Virtual Organization II

- Authorization for methods based on ACLs
  - ACLs bootstrapped from .clarens_access files in module directories
  - Store in DB, can be administered remotely
  - Based on model of Apache .htaccess files
- E.g. for system.auth() method which is required for login:
  - Order allow, deny
  - Allow all in specified group(s) or list of DNs to access method
  - Unless member of group(s) in deny list, or DN in deny list
  - Similar for order deny, allow
- Authorization is hierarchical based on method name
  - E.g. the ACL for 'system' has precedence over 'system.listMethods', making it easy to specify ACLs with the minimum information
- System ACL is special
  - Can specify access to all methods
  - Normal module .clarens_access files cannot specify access controls for other modules

Security and Virtual Organization III

Example *.clarens_access* file for *system* module

```
access=[('system',
    [ORDER_DENY_ALLOW,          # Order
       ['/O=doesciencegrid.org/OU=People'],# Allow DOE certificates
       ['CMS'],                 # Allow group CMS
       [],                      # Deny individuals
       ['revoked_certs'],       # Deny group members
       [None,None,None]]),      # modtime,start_time,end_time
      ('system.updateMethods',
    [ORDER_ALLOW_DENY,          # Order
       ['/O=doesciencegrid.org/OU=People/CN=Conrad Steenberg'],# Allow
       ['admin'],               # Allow group admin
       [],                      # Deny individuals
       [] ,                     # Deny default
       [None,None,None]])]      # modtime,start_time,end_time
```

Security and Virtual Organization IV

Example *.clarens_access* file for *group* module

```
access=[('',                    # module name is prepended
    [ORDER_DENY_ALLOW,          # Order
        [''],                   # Allow
        ['Caltech', 'UFL'],         # Allow 2 groups
        [],                     # Deny individuals
        ['revoked_certs'],          # Deny group members
        [None,None,None]]),         # modtime,start_time,end_time
       ('delete_admin',             # method name
    [ORDER_ALLOW_DENY,              # Order
        ['/O=doesciencegrid.org/OU=People/CN=Conrad Steenberg'], # Allow
        ['admin'],                  # Allow group admin
        [],                     # Deny individuals
        [] ,                    # Deny default
        [None,None,None]])]         # modtime,start_time,end_time
```

Security and Virtual Organization V

- For normal modules, the module name is prepended to the method name
- Authorization does not require changes in the certificate structure
- ACLs and VOs can be remotely administered without system admin intervention
- VO administration allows for multiple group administrators
- Does not require certificate revocation lists – ACLs can be used to deny access to revoked certificates via the VO
- ACLs currently limited to method access, but can also be used for file access control

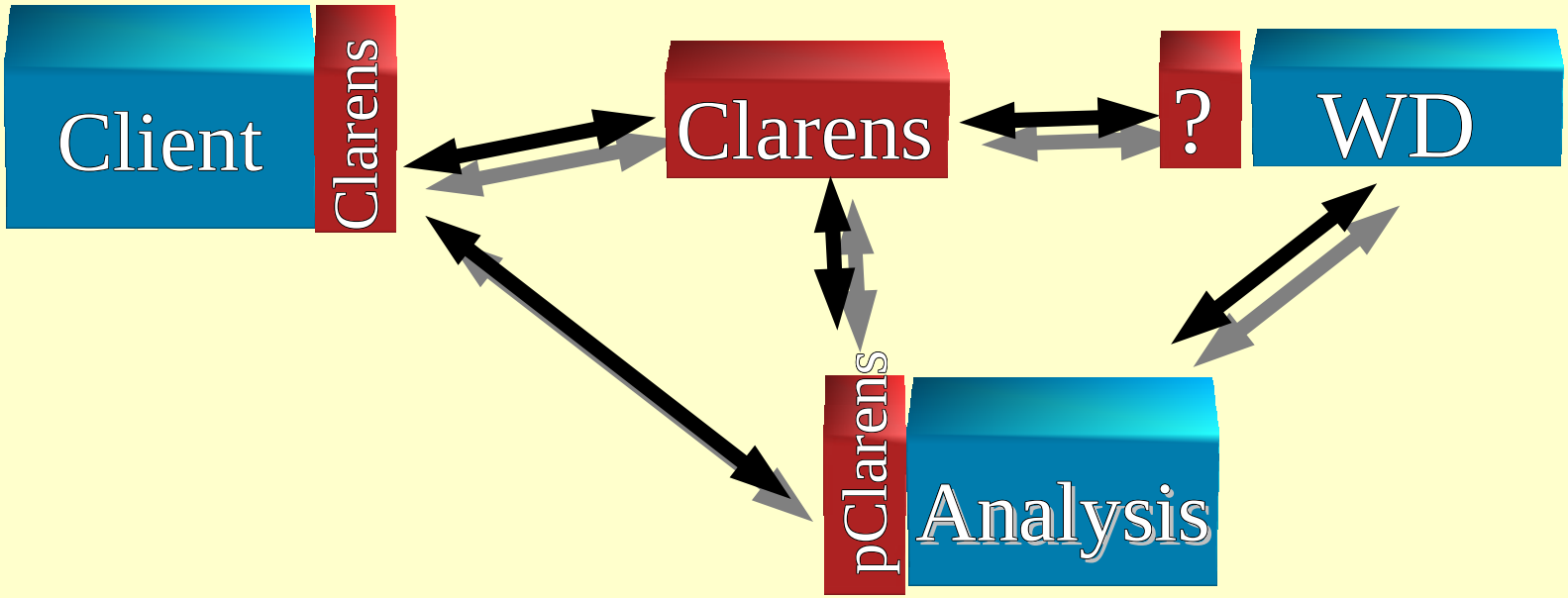
Future: Stateful Analysis

- Grid/distributed environment so far considered 2 extremes of proccessing/data access:
  - **A. Fetch remote data, process locally (0% "granularity")**
  - **B. Submit batch job to remote cluster, get processed data back (100% "granularity")**

- **What if another granularity is required:**
  - **Submit job**
  - **Get feedback**
  - **Request more data from same job**
  - **Data staging/job very time-consuming**

Future: Stateful Analysis

- Connect requesting client directly to analysis code via RPC
- Start analysis code using sceduler that can act as watchdog for process/resource management
- Use Clarens as RPC layer
- Python as scripting language already used
- May need an RPC-addressable scheduler



Summary

- The Clarens architecture presents users and developers with a scalable and relatively fault-tolerant way to implement web services in a Grid environment
- Benefits derived from the commodity Apache server platform
- VO and authorization (ACL) administration can be done remotely after bootstrapping essential information from text files once after installation
- Currently deployed in a variety of projects in the US, at CERN and Pakistan
- More info at http://clarens.sourceforge.net