Clarens Web Services Framework: Services and Development

Conrad Steenberg

California Institute of Technology

On behalf of the GAE Collaborators
at Caltech, CERN, NUST and CERN

Developed as part of the

Particle Physics DataGrid

Overview

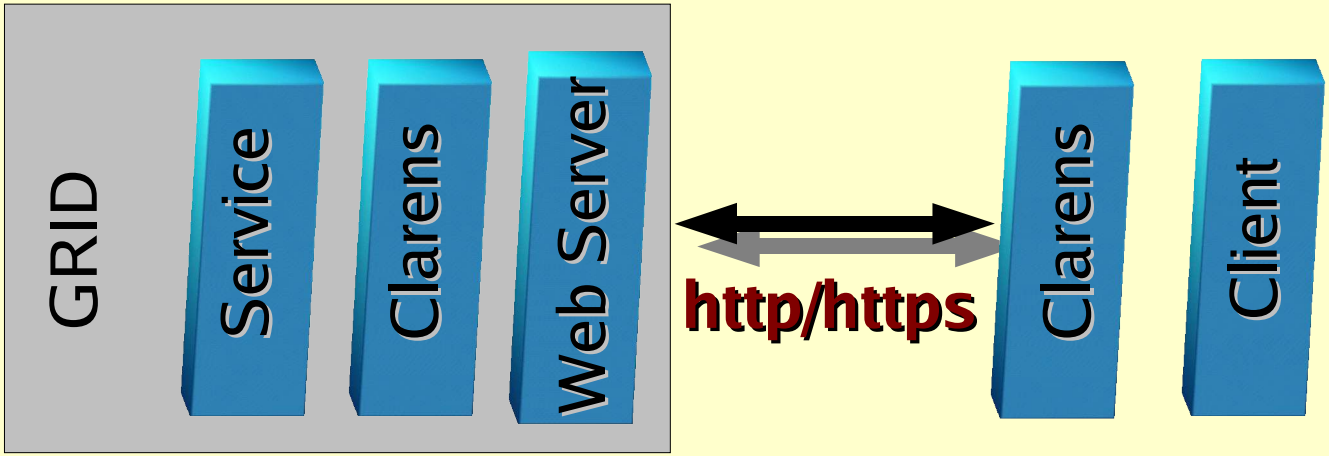
- Background
- Architecture
- Implementations
- GAE Service view
  - Higher level service examples
  - Note on performance
  - Global system view: discovery service
- New protocol: solving the interactive analysis problem
- Conclusion
- ➔ See paper 182 in T5 by *van Lingen et al.*, 89 in T4 by *Le Grand et al.*

Background

- Massive volumes of data distributed between sites of different storage/CPU/network capacities
- Grid model built to support top-down, production-oriented processing: queue response times ~minutes, jobs run hours -> days
- A multitude of 'service' access protocols. Consider
  - LDAP, FTP, Gatekeeper, RLS, GIIS, GRIS, RLS, SAZ, rootd, web pages, e-mail messages, word-of-mouth...
- Most of these protocols can be augmented/replaced by consistent, lightweight access protocol implemented within a common framework
- Result: RPC-based web services framework called *Clarens*
- Leverage universally available tools and standards: XML, HTTP, SSL, X509 certs, web server

Client/Server Architecture

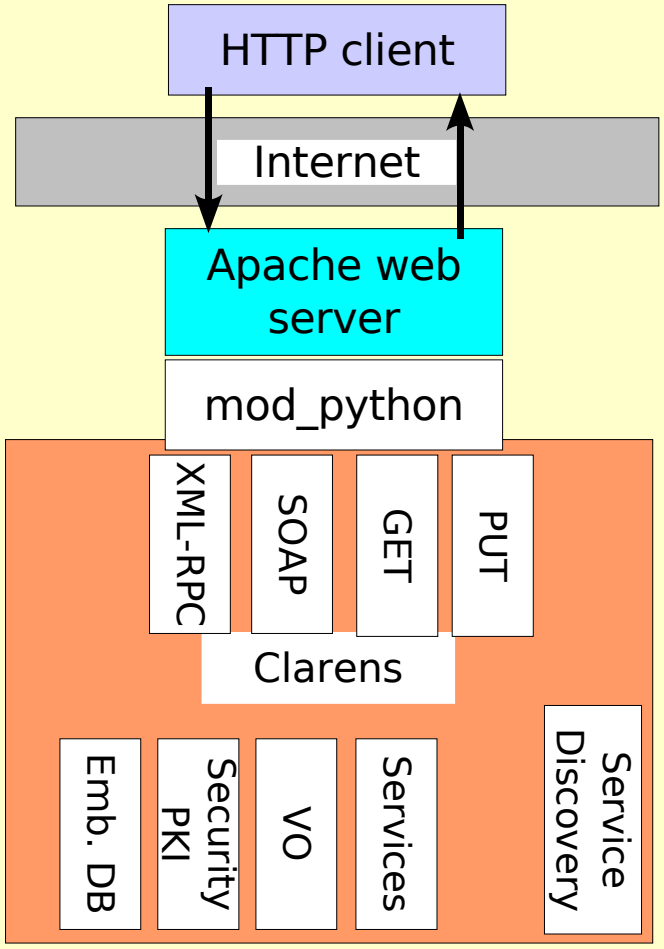
- Clarens connects a client to services



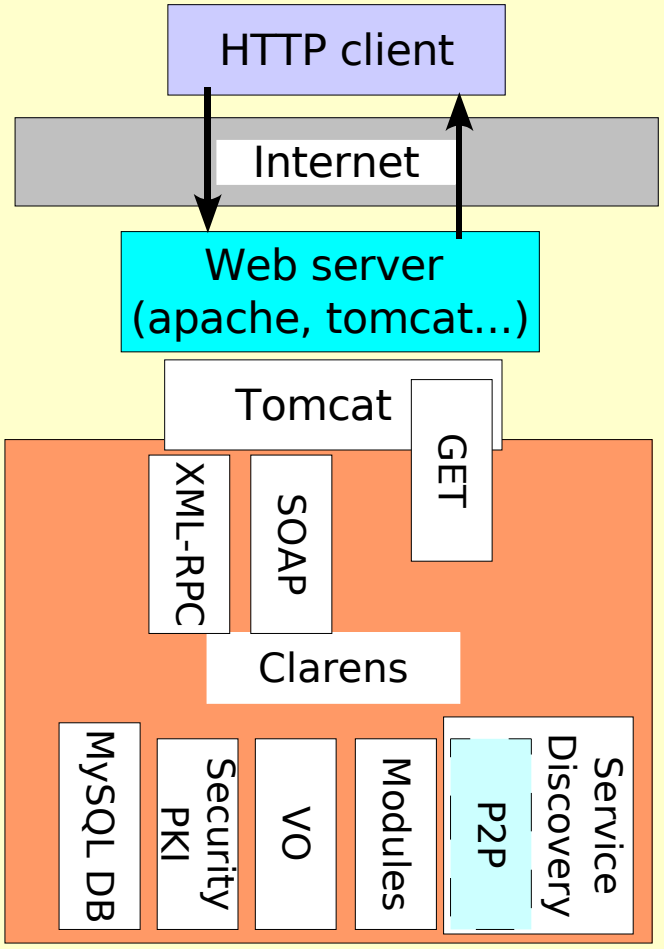
- Acts as a "portal" for Grid services
- Or act as "agent" contacting services on behalf of user

Implementations

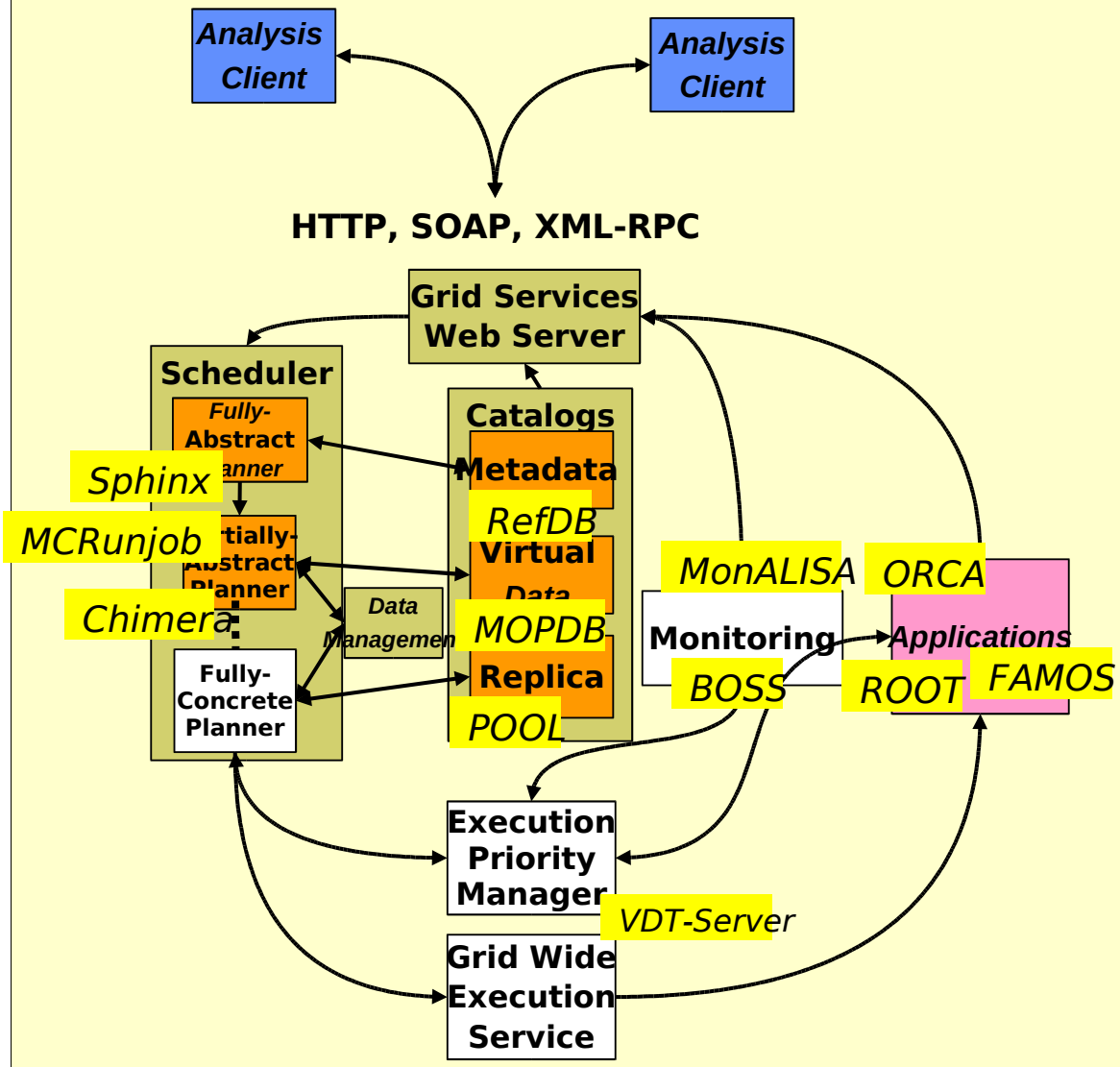
Apache/Python version:



Java/Tomcat



GAE Service View



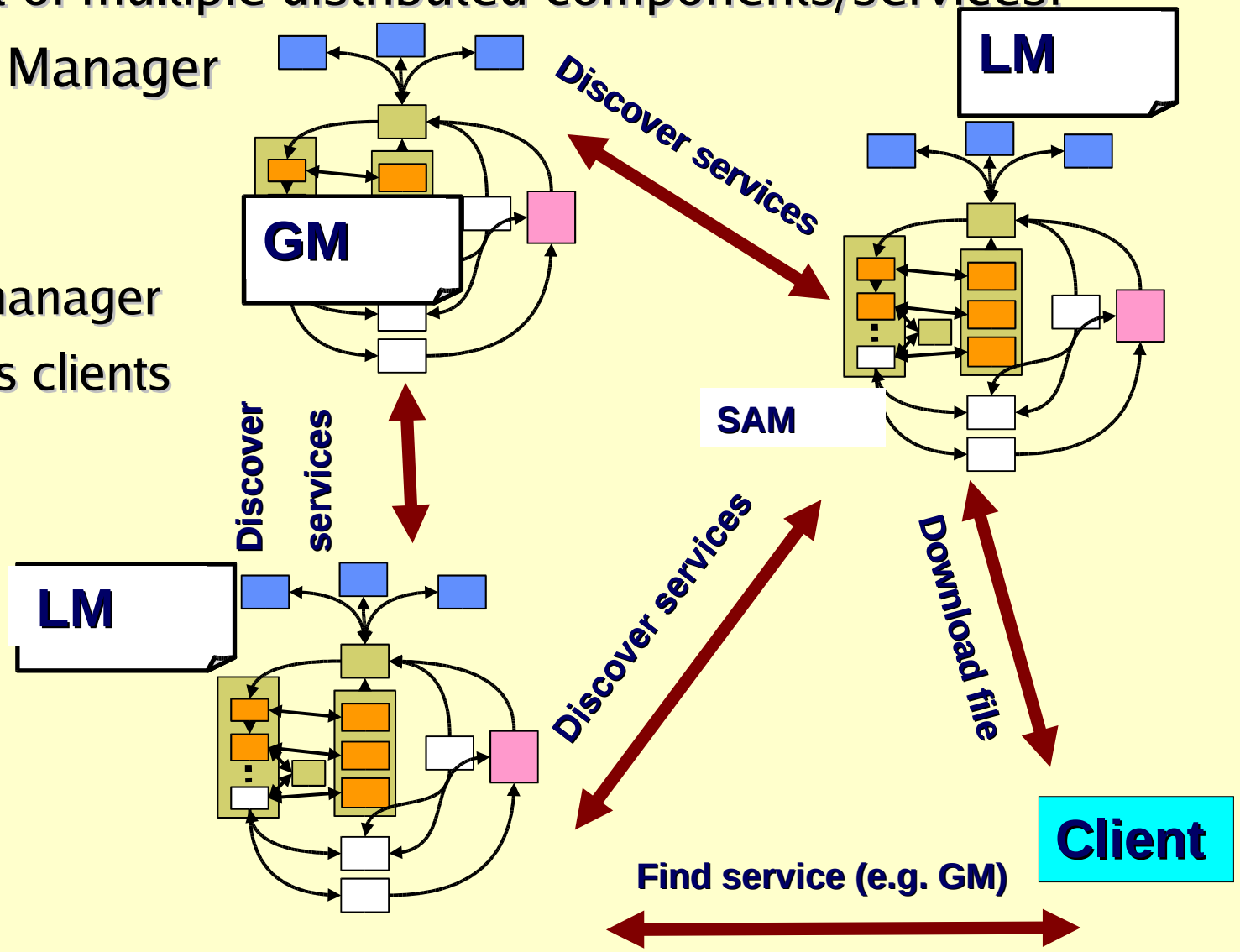
- Analysis Clients talk standard protocols to the the Clarens services portal.
- Simple Web service API allows Analysis Clients (simple or complex) to operate in this architecture.
- Typical clients: ROOT, Web Browser, IGUANA, COJAC
- The Clarens portal hides the complexity of the Grid Services from the client, but can expose it in as much detail as req'd for e.g. monitoring.
- Key features: *Global* Scheduler, Catalogs, Monitoring, and Grid-wide Execution service.

Higher Level Services

- Need flexible distributed services to connect large N clients to M servers -> Global service organization *or chaos continues*
- Organize services into Virtual Organizations within common name space
- Need “service DNS” for service lookup and discovery
  - https://discover.gridservice.info/<protocol>
  - protocols: soap, xmlrpc
- Allows construction of “higher level” or composite services
- Clients (and “agents”) contact multiple servers to provide ultimate result
- Requirement: minimal latency or system becomes unusable

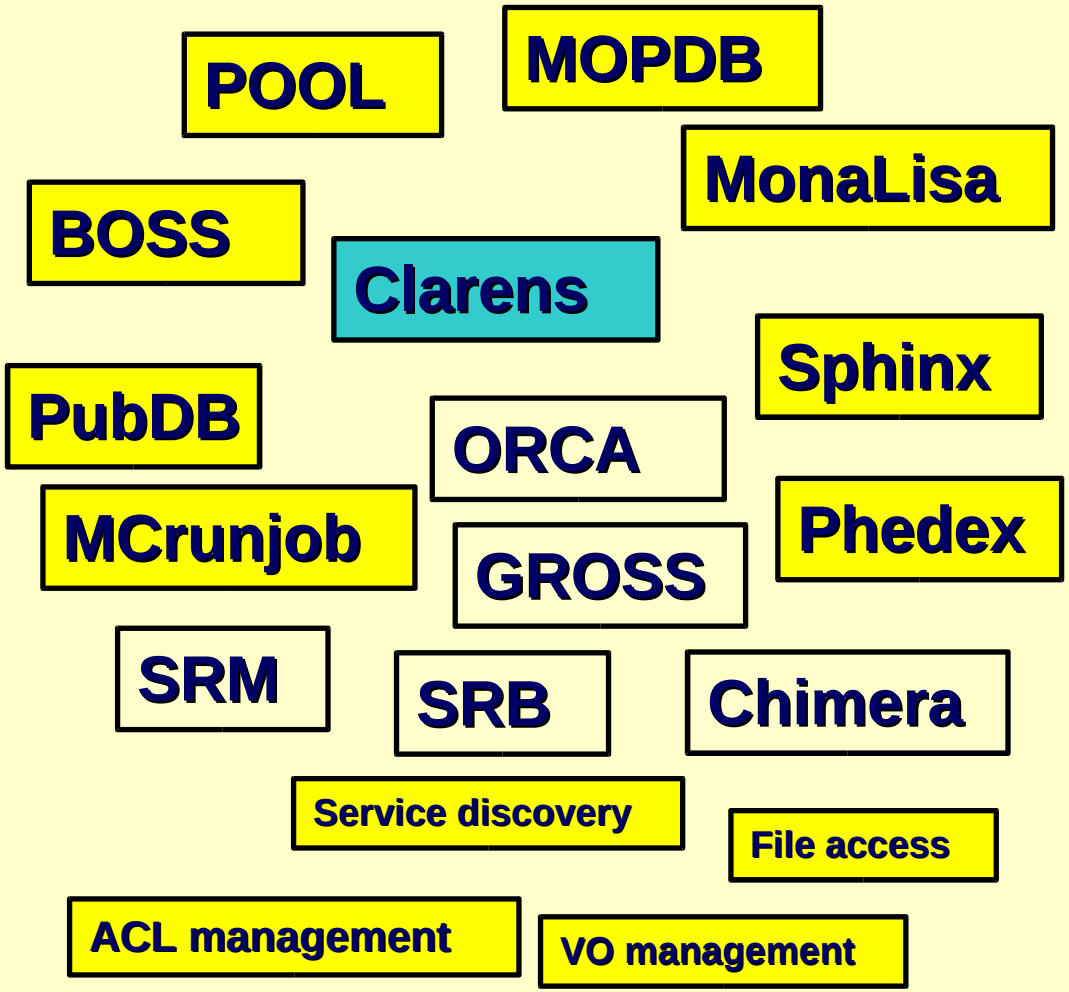
Example: PEAC

- Consist of multiple distributed components/services:
  - Global Manager
  - SAM
  - Proof
  - Local manager
  - Analysis clients



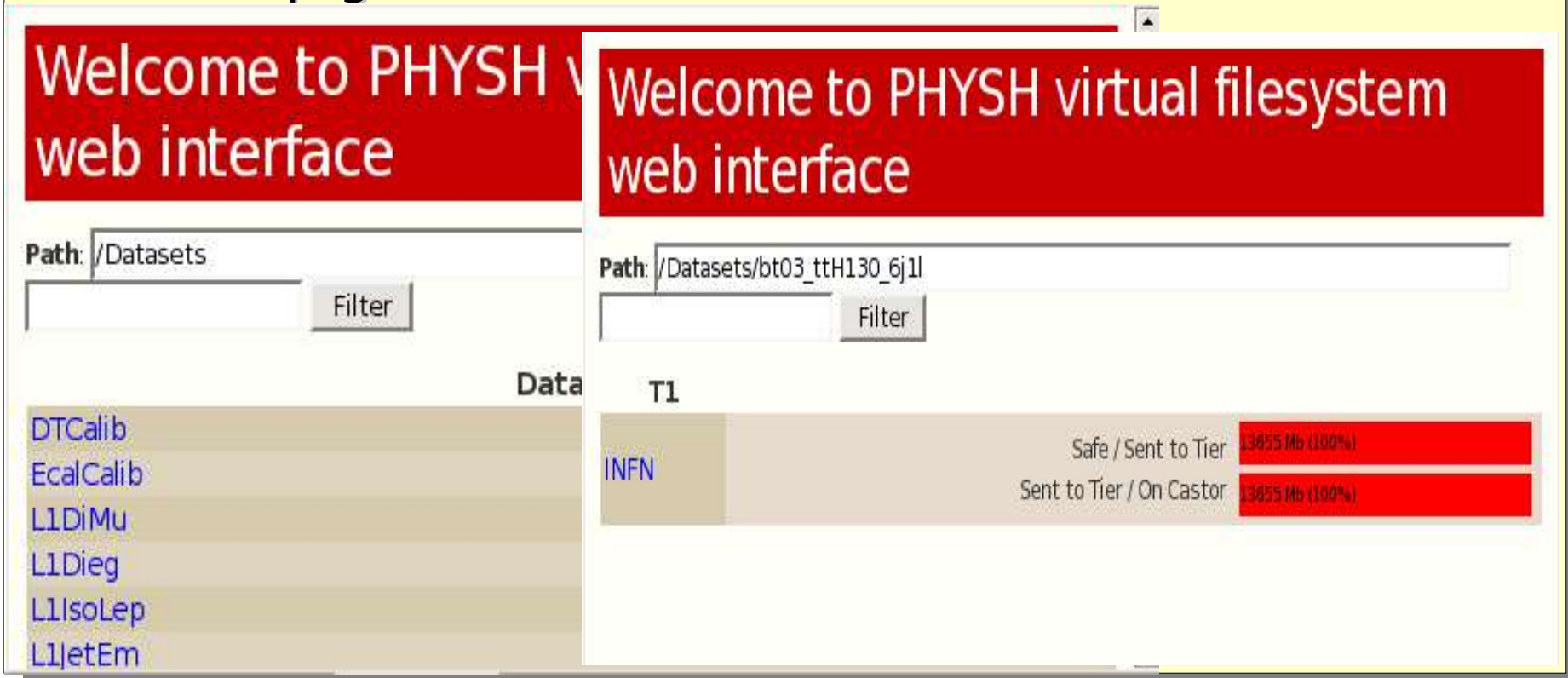
Example: CMS

Applications used within CMS:



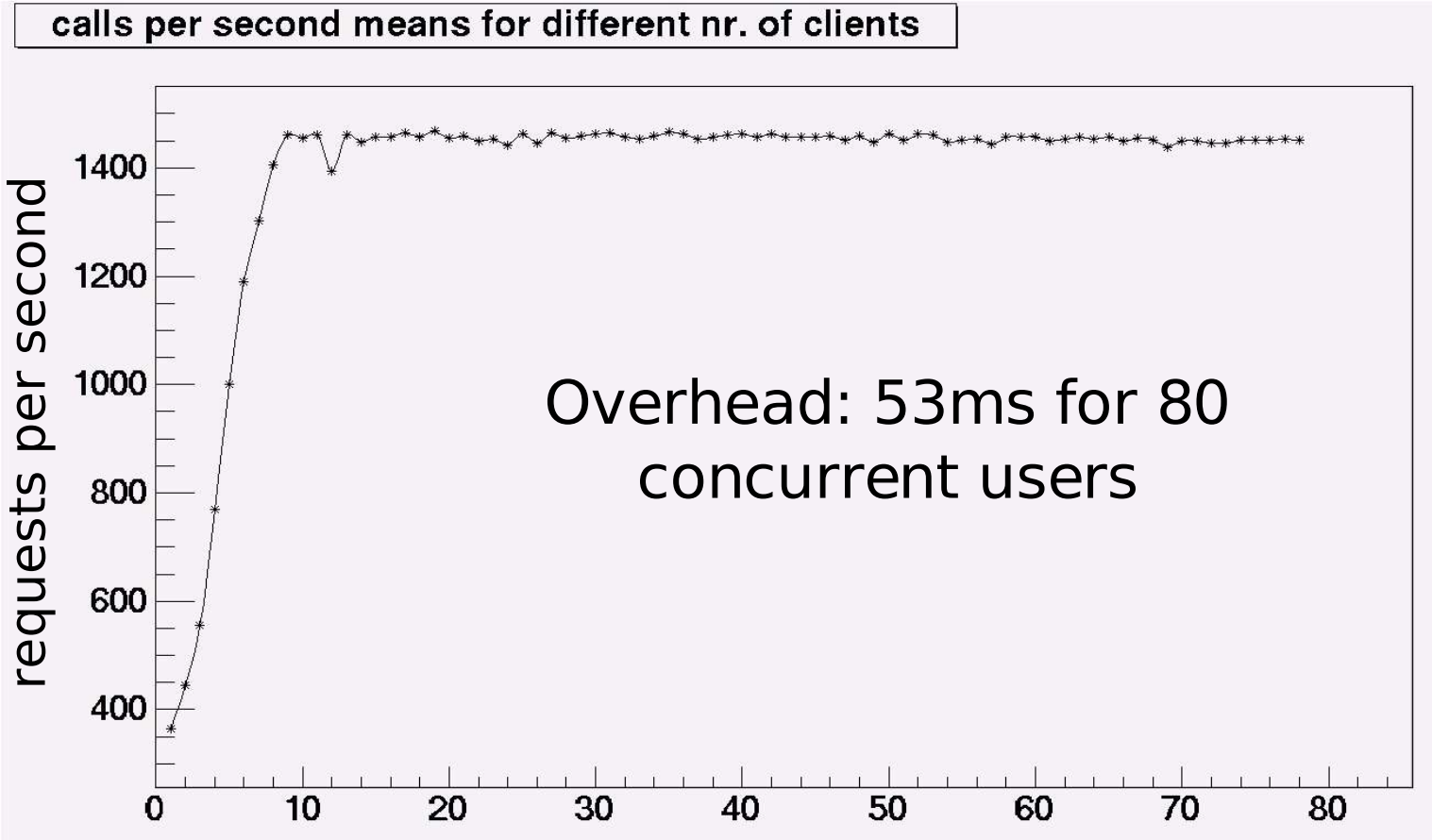
Example Interface: PhySH

- Filesystem-like access to PubDB, Phedex
- Familiar command-line interface
- Connects to multiple services through common interface to search for data, transfer data, monitor data transfer status
- Also web page interface:



Digression: performance

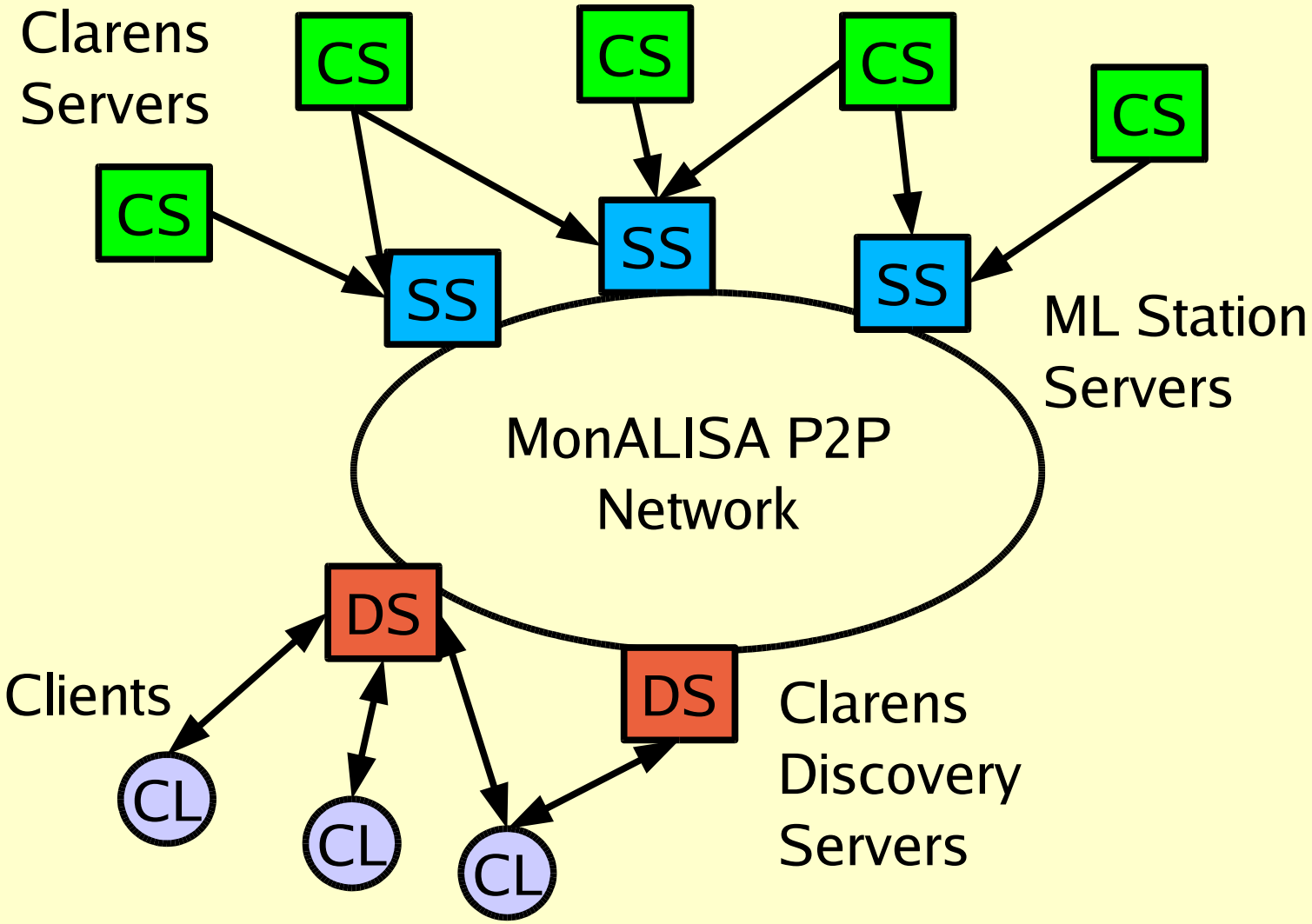
- For interactive analysis response times ~web page load times needed:



Discovery Service

- Leverage existing global MonALISA JINI-based monitoring network to publish and distribute service information
- Each server publish to a station server via lightweight UDP protocol
- Station server publishes to JINI network
- Rapid information propagation allows truly global view of system – both new servers and server time-outs
- Discovery server subscribes to monitoring data with predicates agreed upon by convention as JINI client
- Handles web service queries from clients through JClarens server

Discovery Service II

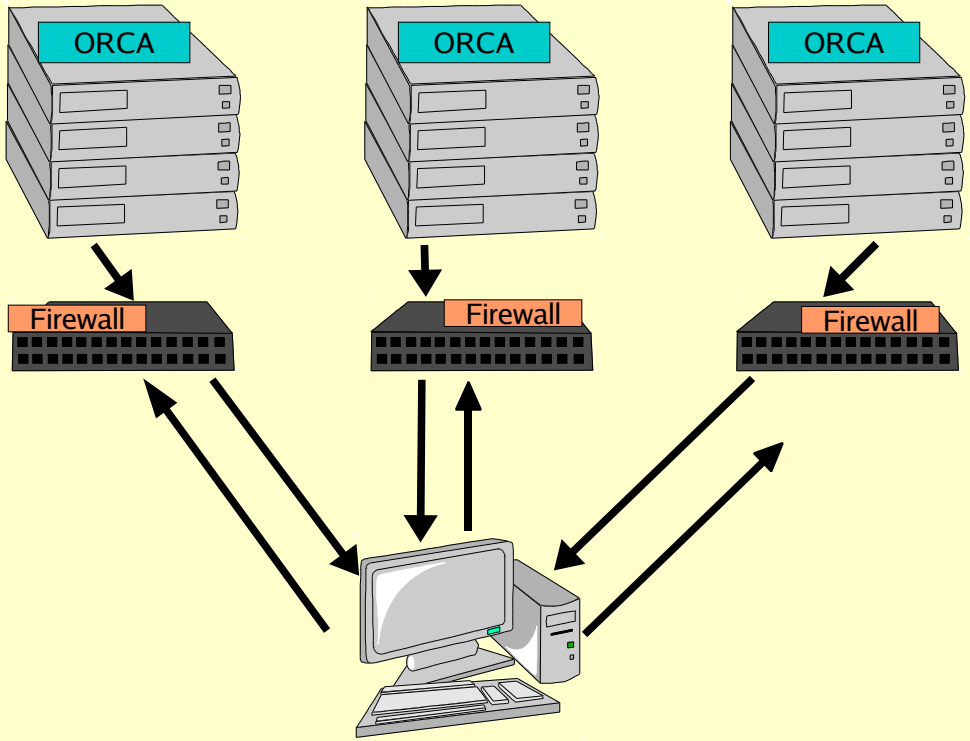


Discovery Service III

https://www.discover.gridservice.info

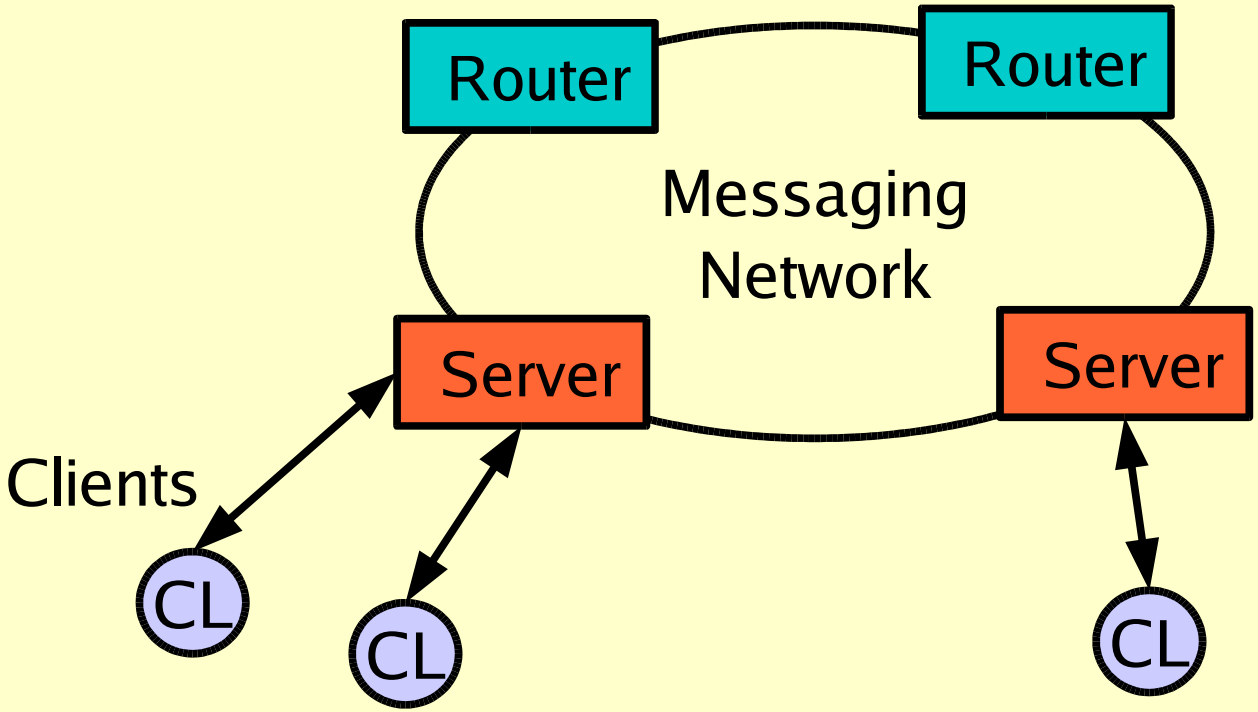
Interactive Analysis

- Presents challenge to client/server model:
  - Vincenzo: Would like to interact with running analysis jobs
  - Asynchronous status notification impossible
    (ML monitors jobs, but still requires client to poll ML repository)
  - F*rewalls!



Interactive Analysis II

- Possible solution: use persistent TCP connection through firewall to messaging network
- No need for client to listen on TCP port
- Server can run on firewall if needed

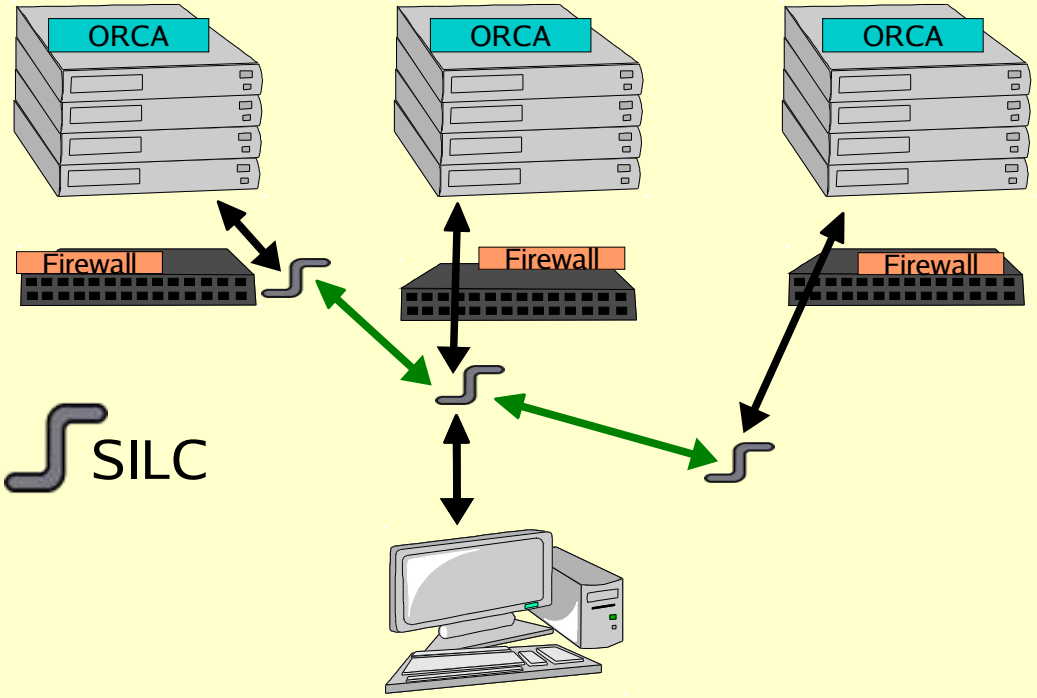


Interactive Analysis III

- Many existing message router implementations:
  - Spread, secure Spread (*), Elvin, JINI, MQSeries (*), Tibco (*), IRC
  - Problem with message security (* or licensing)
- Investigate secure instant messaging router: SILC
- Security based on PKI: RSA pub/priv keys
- Multiple jobs can connect connections using one keypair, different nicknames
- Messages signed/encrypted in network
- To make into Grid-security compatible services network:
  - Make SILC PKI compatible with X509
  - Feed messages into existing Clarens server

Interactive Analysis IV

- Or more concretely:



- Status:
  - prototype exists, re-using Python server code
  - command-line client (chat with remote service)
  - presence notification

Summary

- Clarens presents users with a secure, low-latency and easily accessible way to implement and access Grid-enabled web services
- Maturing Jclarens implementation
- Focus during last year on horizontal and vertical scalability: network of servers, higher performance
- Facilitate fault-tolerant high-level (compound) services via discovery service, using MonALISA as back-end
- Rich set of services in development as part of CMS
- Investigating interactive analysis through messaging service
- More info at http://clarens.sf.net

GAE People

- Caltech:
  - Julian Bunn
  - Iosif Legrand
  - Harvey Newman
  - Michael Thomas
- CERN
  - Giulio Eulisse
- NUST
  - Ashiq Anjum
  - Tahir Azim
- UFL
  - Paul Avery
  - Dimitri Bourilkov
  - Richard Cavanaugh
  - Laukik Chitnis
  - Mandar Kulkarni
  - Jang Uk In