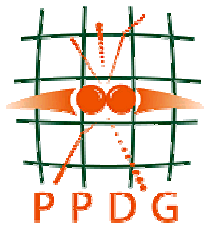
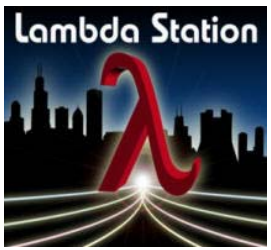
Grid Analysis Environment (GAE) (overview)

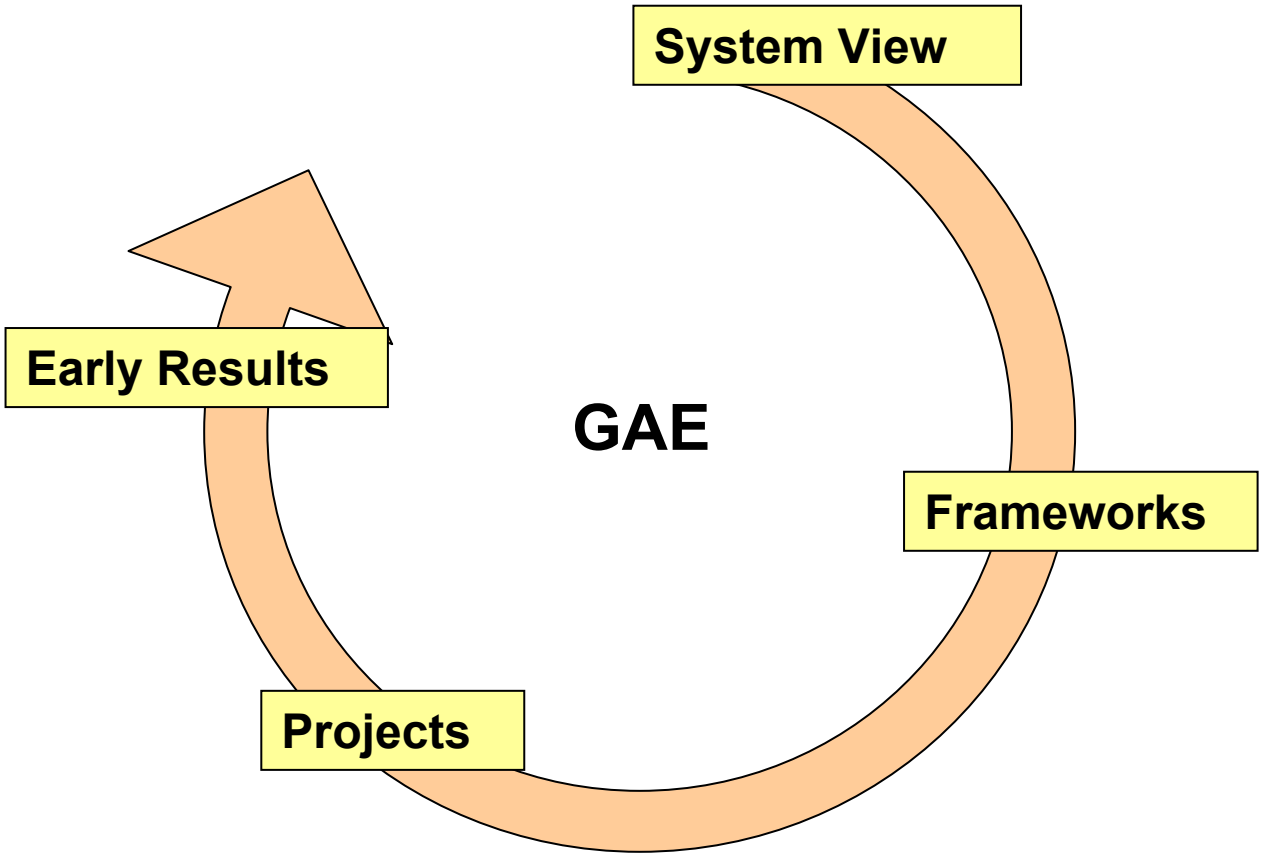
Frank van Lingen (fvlingen@caltech.edu)

(on behalf of the GAE group)





Outline



Goal

Provide a transparent environment for a physicist to perform his/her analysis (batch/interactive) in a distributed dynamic environment: Identify your data (Catalogs), submit your (complex) job (Scheduling, Workflow,JDL), get "fair" access to resources (Priority, Accounting), monitor job progress (Monitor, Steering), get the results (Storage, Retrieval), repeat the process and refine results

Support data transfers ranging from the (predictable) movement of large scale (simulated) data, to the highly dynamic analysis tasks initiated by rapidly changing teams of scientist

Constraints

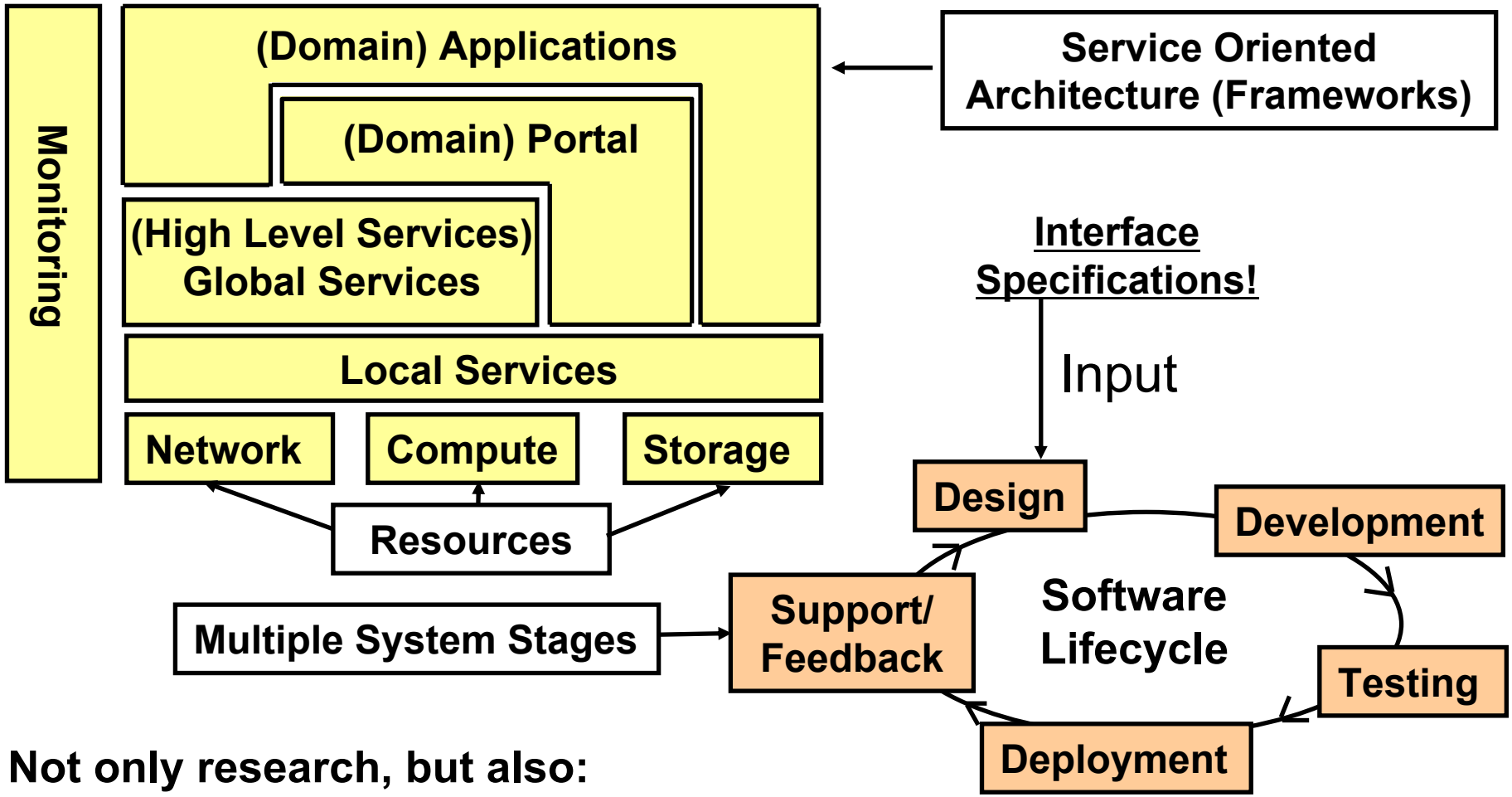
The "Acid Test" for Grids; crucial for LHC experiments

- Large, diverse, distributed **community of users**
- **Support for 100s to 1000s of analysis tasks, shared among dozens of sites**
- Widely varying **task requirements and priorities**
- **Need for priority schemes, robust authentication and security**

Operates in a resource-limited and policy-constrained global system

- **Dominated by collaboration** policy **and** strategy **(resource usage and priorities)**
- **Requires** real-time monitoring**; task and workflow tracking; decisions often based on a** global system view

System View



Not only research, but also:

- **Software Engineering**
- **Sociology**

System View (Details)

Domains

- Virtual Organization and Role management

Service Oriented Architecture

- Authorized Access
- Access Control Management(groups/individuals)
- Discoverable
- Protocols (XML-RPC, SOAP,….)
- Service Version Management
- Frameworks: Clarens, MonALISA...

Monitoring

- End-to-end monitoring,collecting and disseminating information
- Provide Visualization of Monitor Data to Users

System View (Details)

Local Services (Local View)

- Local Catalogs, Storage Systems, Task Tracking (Single User Tasks), Policies, Job Submission

Global Services (Global View)

- Discovery Service, Global Catalogs, Job Tracking (Multiple User Tasks), Policies

High Level Services (“Autonomous”)

- Acts on monitor data and has global view
- Scheduling, Data Transfer, Network Optimization, Tasks Tracking (many users)

System View (Details)

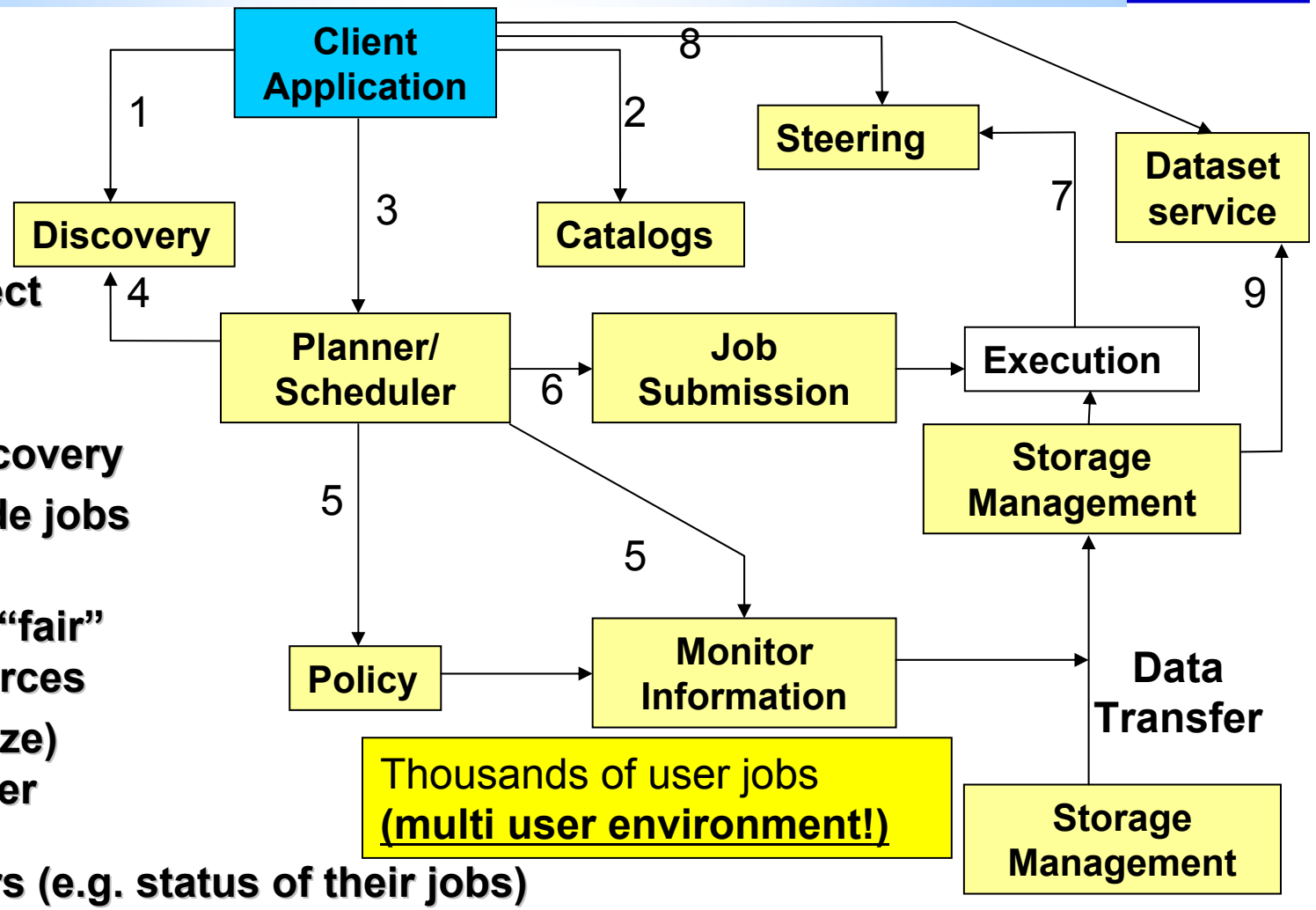
(Domain) Portal

- One Stop Shop for Applications/Users to access and Use Grid Resources
- Task Tracking (Single User Tasks)
- Graphical User Interface
- User session logging (provide feedback when failures occur)

(Domain) Applications

- ORCA/COBRA, IGUANA, PHYSH,….

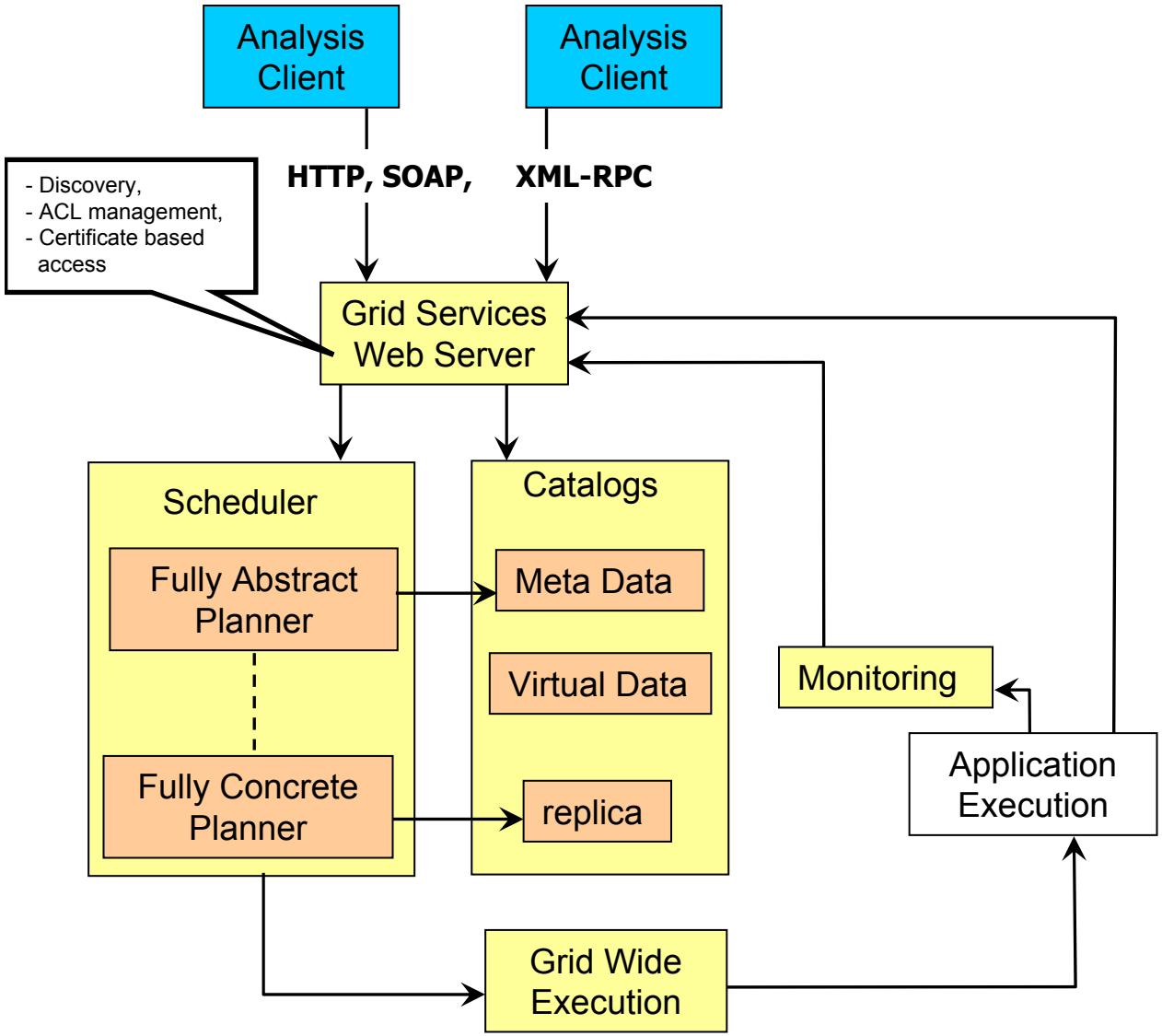
Single User View



- Catalogs to select datasets,
- Resource & Application Discovery
- Schedulers guide jobs to resources
- Policies enable "fair" access to resources
- Robust (large size) data (set) transfer

- Feedback to users (e.g. status of their jobs)
- Crash recovery of components (identify and restart)
- Provide secure authorized access to resources and services.

Service Oriented View



- Clients talk standard protocols to "Grid Services Web Server",
- Simple Web service API allows simple or complex analysis clients
- Typical clients: ROOT, Web Browser, ….
- Clarens portal hides complexity
- Key features: Global Scheduler, Catalogs, Monitoring, Grid-wide Execution service.

Peer-2-Peer View

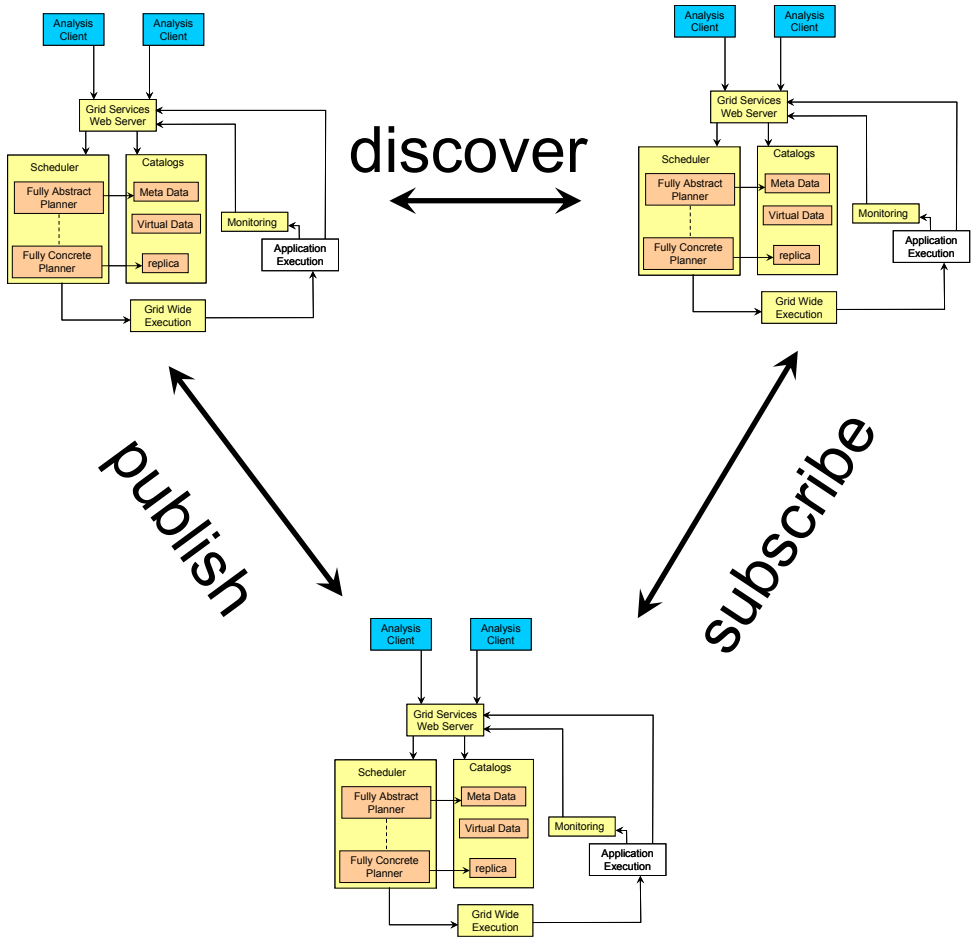
"Peer-2-Peer" configuration enhances:

- **robustness**
- **Scalability**

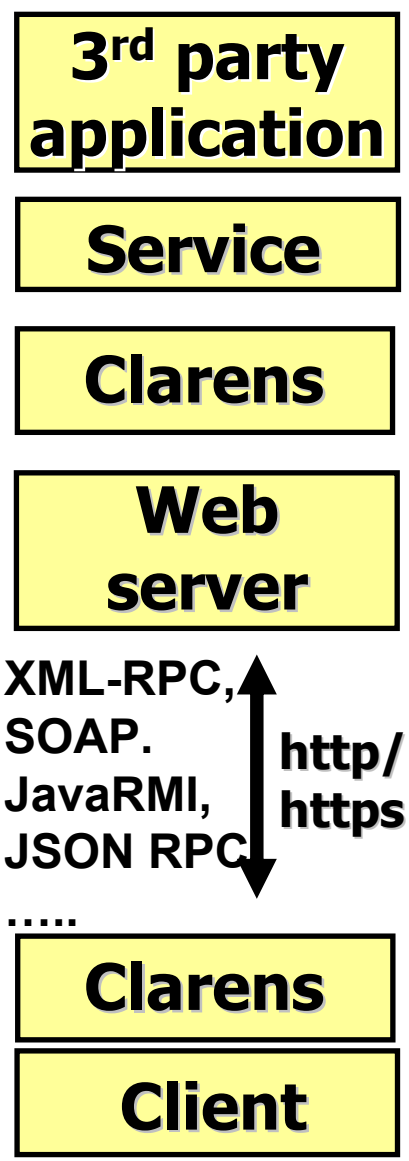
Provide:

- **Discovery**
  - **Services**
  - **Software**
  - **……**
- **Publication**
- **Subscription**

No Single point of failure



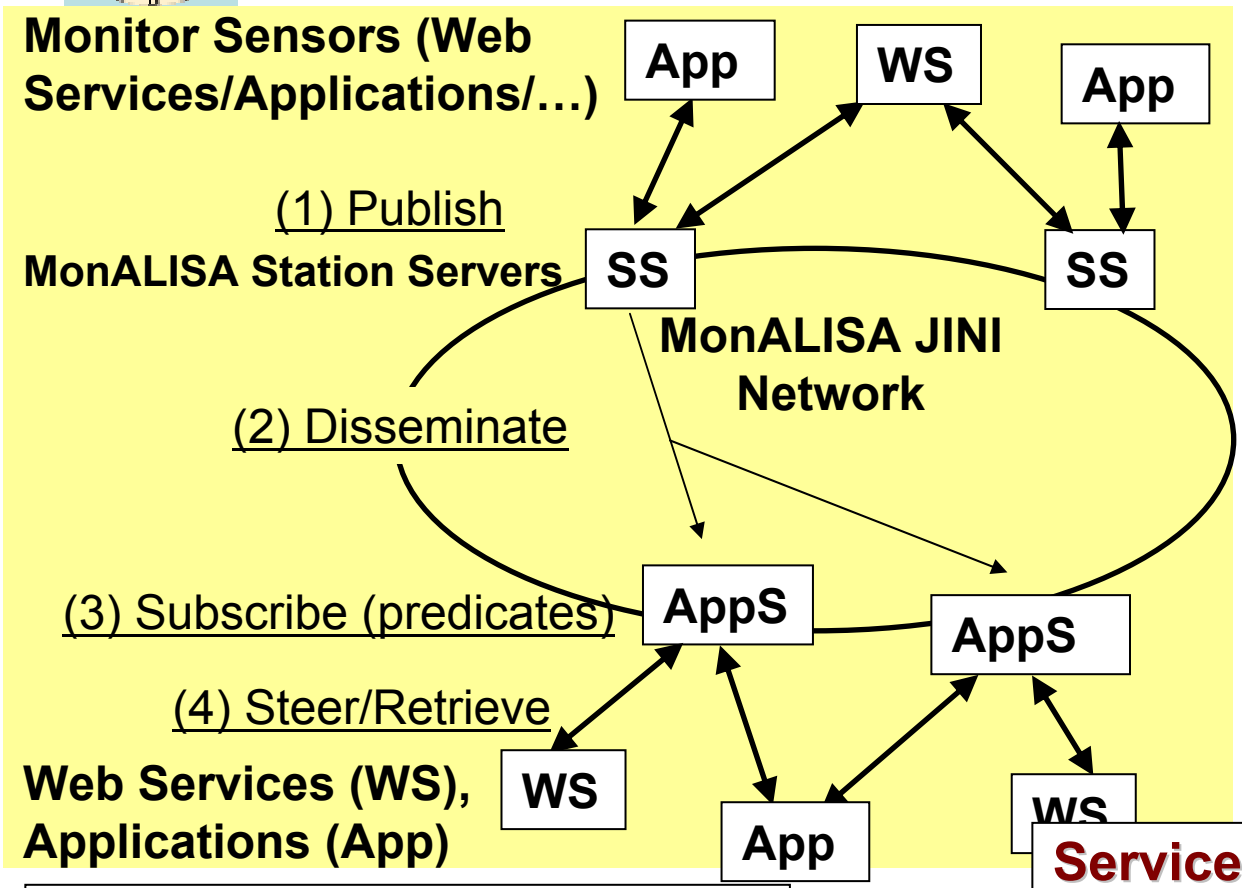
Framework clarens



- Authentication (X509)
- Access control on Web Services.
- Remote file access (with access control)
- Discovery of Web Services and Software
- Shell service. Shell like access to remote machines (managed by access control lists)
- Proxy certificate functionality
- Group management VO and role management
- Good performance of the Web Service Framework
- Integration with MonALISA

Framework

MonALISA
MONitoring Agents using a Large Integrated Services Architecture



Active filter agents
- Process data
- Application specific monitoring

Mobile agents
- decision support
- global optimisations

MonALISA able to dynamically
- register & discover

Based on multi-threaded engine
- Very scalable

Services are self describing

Code updates
- Automatic & secure

Dynamic configuration for services
- Secure Admin Interface

Fully distributed, no single point of failure!

GAE Related Projects

DISUN (deployment)

- Deployment and Support for Distributed Scientific Analysis

Ultralight (development)

- Treating the network as resource
- "Vertically" Integrated Monitor Information
- Multi User, resource constraint view

MCPS (development)

- Provide Clarens based Web Services for batch analysis (workflow)

SPHINX (development)

- Policy based scheduling (global service) exposed a Clarens Web Service using MonALISA monitor information

SRM/Dcache (development)

- Service based data transfer (local service)

Lambda Station (development)

- Authorized programmability of routers using MonALISA & Clarens

PHYSH

- Clarens based services for command line user analysis

CRAB

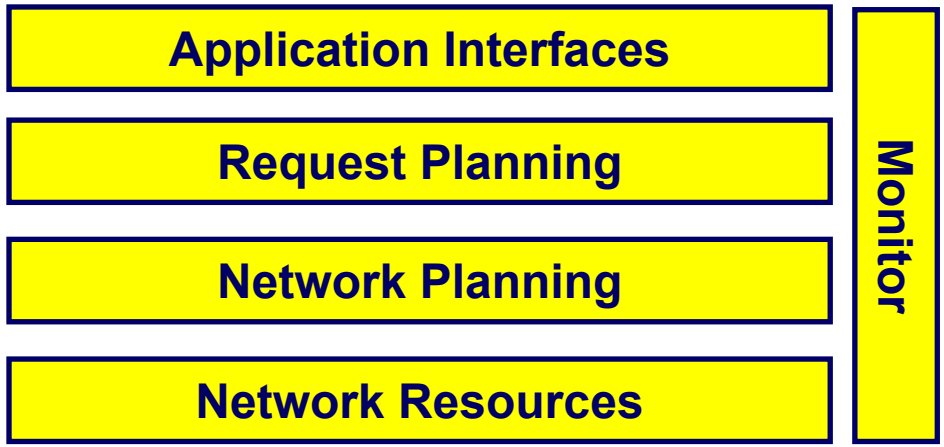
- Client to support user analysis using Clarens Framework

GAE and Ultralight
Make the Network an Integrated Managed Resource

Unpredictable multi user analysis

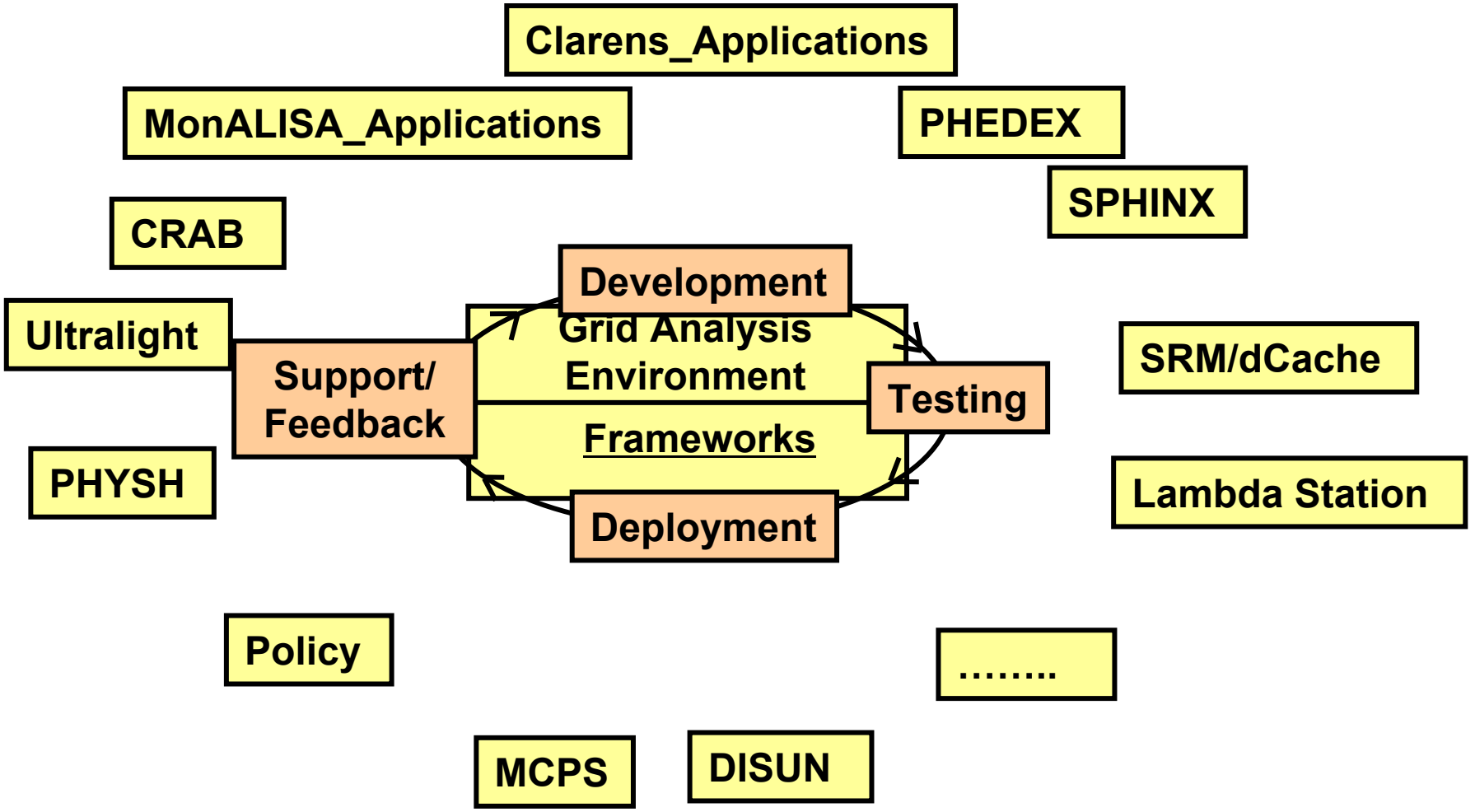
Overall demand typically fills the capacity of the resources

Real time monitor systems for networks, storage, computing resources,… : E2E monitoring



Support data transfers ranging from the (predictable) movement of large scale (simulated and real) data, to the highly dynamic analysis tasks initiated by rapidly changing teams of scientists

Combining Grid Projects into Grid Analysis Environment



GAE focuses on integration

GAE Deployment

- Installation of CMS (ORCA, COBRA, IGUANA,…) and LCG (POOL, SEAL,…) software on Caltech GAE testbed. Serves as environment to integrate applications as web services into the Clarens framework.
- Demonstrated distributed multi user GAE prototype at SC03, SC04 (using BOSS and SPHINX)
- Analysis Prototype in January 2005 currently upgraded to work with CRAB.
- PHEDEX deployed at Caltech, UFL, UCSD and transferring data

GAE Deployment

- Clarens **has been deployed on ~30+ machines. Other sites: Caltech, Florida, Fermilab, CERN, Pakistan, INFN (see Clarens Discovery Service)**
  - **Available in Java and Python**
- Core System: **Discovery, Proxy, Authentication, Remote File Access, Shell Service,….**
- **Clarens** Discovery Service **part of OSG (uses MonALISA)**
  - **Talking with** EGEE **on common interface**
  - Software Discovery **Service being developed (SCRAM based prototype available.**
- **GAE** distributed test framework **(Java, Python) available. Daily testing and verification.**
- Generic Catalog Service **developed to expose pooldbs, refdb, phedex, …and other DBs as entities with key/values**

GAE Deployment

PHEDEX and Pubdb Web Services deployed in Florida and Caltech.

- Part of the test framework

Working with the MCPS group to develop an Clarens based MCPS Web Service interface and browser based GUI.

- Supporting Production and Analysis

Web Service interface to BOSS (CMS Job/Task book-keeping system)

SPHINX: Distributed scheduler developed at UFL

Clarens/MonALISA Integration: Facilitating user-level Job/Task Interaction

- First version of a Steering Service

Java Webstart dashboard being developed for Clarens based services

- Also: Cross browser support for JavaScript browser GUI (Firefox, IE, Safari, ….) based on JSON

CAVES: Analysis code-sharing environment developed at UFL

Work with CERN to have the GAE components included in CMS software distribution.

GAE components being integrated in the DPE and VDT distribution used in US-CMS.

Lessons learned

- **Quality of (the) service(s)**
  - Lot of exception handling needed for robust services (gracefully failure of services)
  - Time outs are important
- **Need very good performance for composite services**
- **Discovery services**
  - enables location independent service composition.
  - semantics of services are important (different name, name space, and/or WSDL)
- **Web service design: Not every application is developed with a web service interface in mind**
- **Interfaces of 3rd party applications change: Rapid Application Development**
- **Overlapping functionality of applications (but not same interfaces!)**
- **Not one single solution for CMS**
- **Not every problem has a technical solution, conventions also important**

Future Directions

-Data movement using PHEDEX

-Integration of runjob into current deployment of services

- Full chain of end to end analysis

-Develop/deploy accounting service

-Improved GUI interface (Webstart client)

-Improve exception handling

-Integrate/interoperability mass storage (e.g. SRM) applications into/with Clarens environment

-Steering service

-Autonomous replication

-Trend analysis using monitor data

-E2E error trapping and diagnosis: cause and effect

-Strategic Workflow re-planning

-Adaptive steering and optimization algorithms

-Multi user distributed environment

More Information

GAE: http://ultralight.caltech.edu/gaeweb/portal

Ultralight: http://ultralight.caltech.edu

WIKI: http://ultralight.caltech.edu/gaeweb/wiki/

Thank You.

Questions?