UVA HPC & BIG DATA COURSE

Service oriented Architecture and Web services

Adam Belloum

Web services are everywhere

- "If you hit Amazon.com gateway page, the application calls more than **100 Web services** to collect data and construct the page for you"

  Wenrner Vogels, CTO Amazon, ACM Queue 2006

- "More than **1 6 000 000** job submission were processed using Web Services during 2006 (30% of all jobs run at the EBI)"

  European Bioniformatics institute

Bringing HPC to the web

- "We want to make HPC resources accessible and useful to **scientists who are more comfortable with the web** than they are with compilers, batch queues, or punched cards. Thanks to Web 2.0 APIs, we believe HPC can speak the same language as the web."
  - NEWT is a web service that allows you to access computing resources at NERSC through a simple RESTful API

the National Energy Research Scientific Computing Center (NERSC) is the primary scientific computing facility for the **Office of Science** in the **U.S. Department of Energy**.]

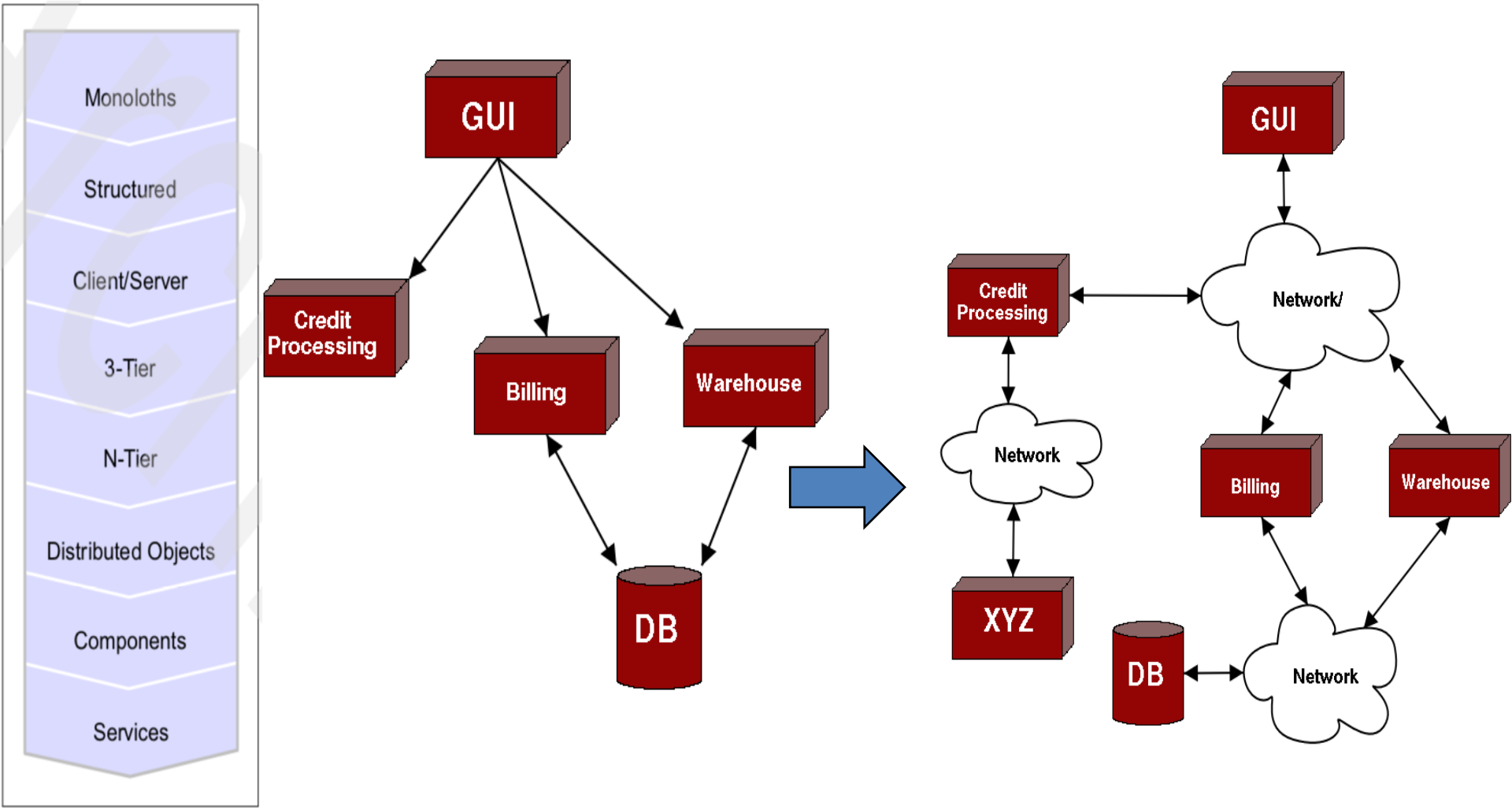
Web services are used entry to HPC

- High Performance Computing (HPC) with Amazon Web Services [1]
- Microsoft HPC Web service API reference [2]

[1] https://aws.amazon.com/hpc/
[2] https://docs.microsoft.com/en-us/previous-versions/windows/desktop/hh560258(v=vs.85)

Content

- Service Oriented Architecture
- Web services
  - SOAP (Web) services
  - Web Service Reference Framework (WSRF)
  - REST (Web) service / REST API



Source: Chapter 2, IBM Redbook, **Patterns: Service- Oriented Architecture and Web Services**

Problem

Cumulative effect of decades of **growth** and evolution has produced severe **complexity**

- Redundant and **non-reusable** programming
- Real **integration** killer –multiplicity of interfaces

> "Before SOA emerged in the late 1990s**, connecting an application to data** or **functionality housed in another system** required complex point-to-point integration—integration that developers had to recreate, in part or whole, for each new development project"

SOA (Service-Oriented Architecture)
By: IBM Cloud Education[1] July 2019

[1] https://www.ibm.com/cloud/learn/soa#toc-what-is-an-sqoa7ntn

Code evolution: dependency graph

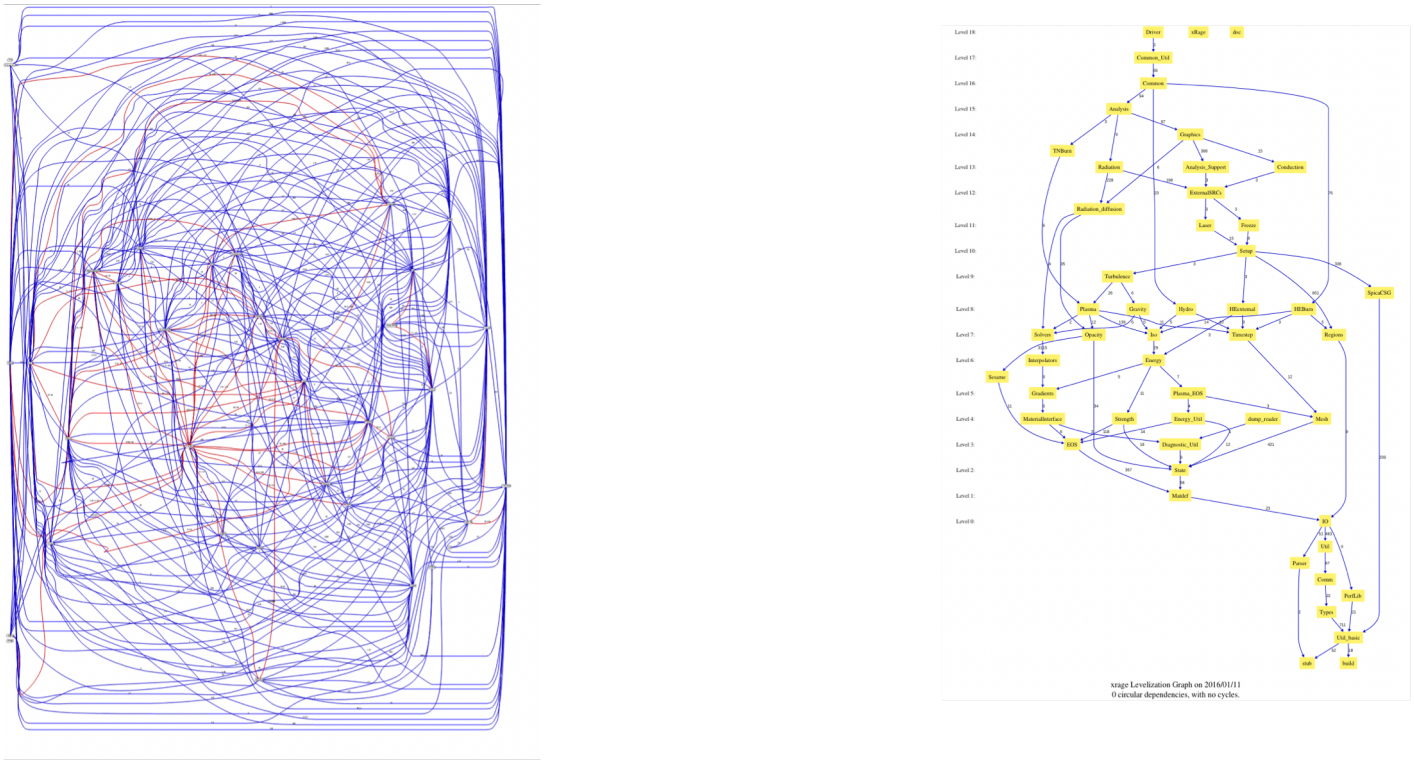


Figure 1: xRage dependency graphs. Left: The "hairball" graph, October 2014. Right: The levelized graph, January 2016.

"… an **Eulerian AMR radiation-hydronamics code**. xRage is descended from Sage, a code originally written around **1990**. It contains about **470K lines of source code**, not counting numerous third-party libraries from LANL and elsewhere. It is written mostly in **Fortran 90**, with **some C and C++**, and uses **MPI-only parallelism**."

Source: http://sc16.supercomputing.org/sc-archive/tech_poster/poster_files/post196s2-file3.pdf

Service Oriented Architecture
--New computing models

- Allow for **incremental** implementations & migration of code
  → Include a development environment that will be built
    - around a standard component framework,
    - promote better reuse of modules and systems,
    - allow legacy code to be migrated to the framework,
    - allow for the timely implementation of new technologies.

Allow implementation of new computing models; specifically, portal-based models, Grid computing, and on-demand computing (cloud)

A service-oriented architecture -
- not just Web services

- First, it must be understood that *Web services* does not equal *service-oriented architecture*.
- Web services is a **collection of technologies**, including HTTP, XML, SOAP, WSDL, and UDDI,

Service Oriented Architecture is "an application architecture within which all functions are defined as **independent** services with well defined **invocable** interfaces".

A service-oriented architecture

-- services are independent

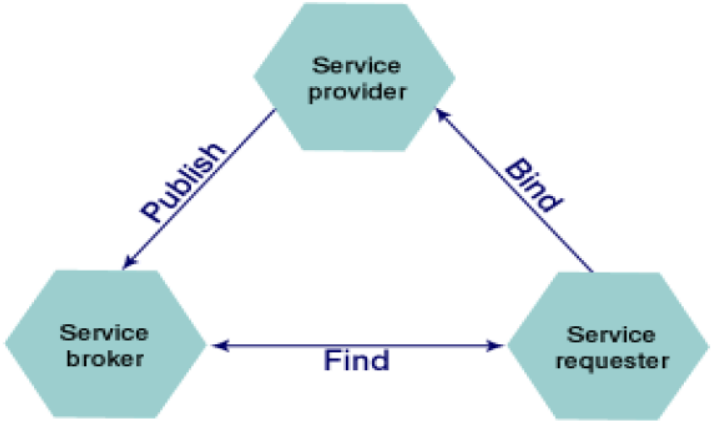
- All functions are defined as **services**.
- All **services** are **independent**
  - Operate as "black boxes";
    - external components neither know nor care how boxes are executed
- The **interfaces** are **invocable**
  - it is irrelevant whether the service are **local** or **remote**
    - what **interconnect scheme** or **protocol** is used to effect the invocation,
    - what **infrastructure components** are required to make the connection.

A service-oriented architecture
-- service Interface

- **Interface** is the key, & the focus of the calling application.
  - It defines the **required parameters** and the **nature** of the **result**
- It is the **system's responsibility** to manage the invocation of the service,
- This allows two critical characteristics to be realized:
  - Services are **truly independent**,
  - They can be **managed**: Security, Deployment, Logging, Dynamic rerouting, and Maintenance

A service-oriented architecture
-- Model

- A *service provider*
  - provides a service interface for a software asset that manages a specific set of tasks.
- A *service requester*
  - discovers and invokes other software services to provide a business solution..
- A *service broker*

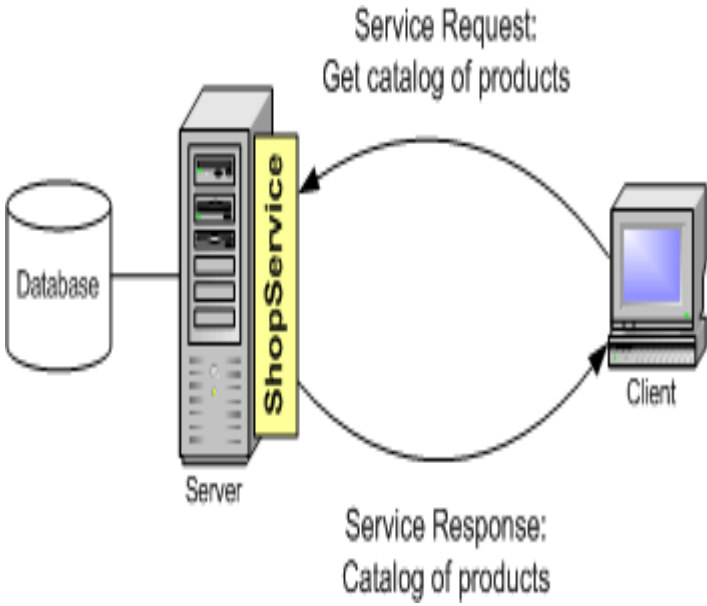


Content

- Service Oriented Architecture
- Web services
  - SOAP Based Web services
  - REST (Web) service
- Example of usage of Web Service in Scientific applications

sketchy example of how a Web Service works

- The *clients* (the PCs at the store)
  - contact the *Web Service* on remote *server*
  - send a *service request* asking for the catalog
  - The server returns the catalog through a *service response*.



Web Services have certain advantages over other technologies

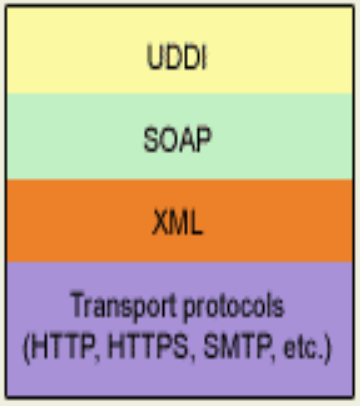
what makes Web Services special?

- Web Services are **platform-independent** and **language-independent**

- Web Services use **HTTP** for transmitting.

Enabling technologies

- XML: The Extensible Markup Language
- SOAP:
  - Simple Object Access Protocol is an XML-based lightweight protocol for the exchange of information in a decentralized,
- WSDL:
  - The Web Services Description Language is an XML vocabulary that provides a standard way of describing service IDLs.
- UDDI:
  - The Universal Description, Discovery, and Integration specification provides a common set of SOAP APIs that enable the implementation of a service broker.



Web Services Protocol Stack

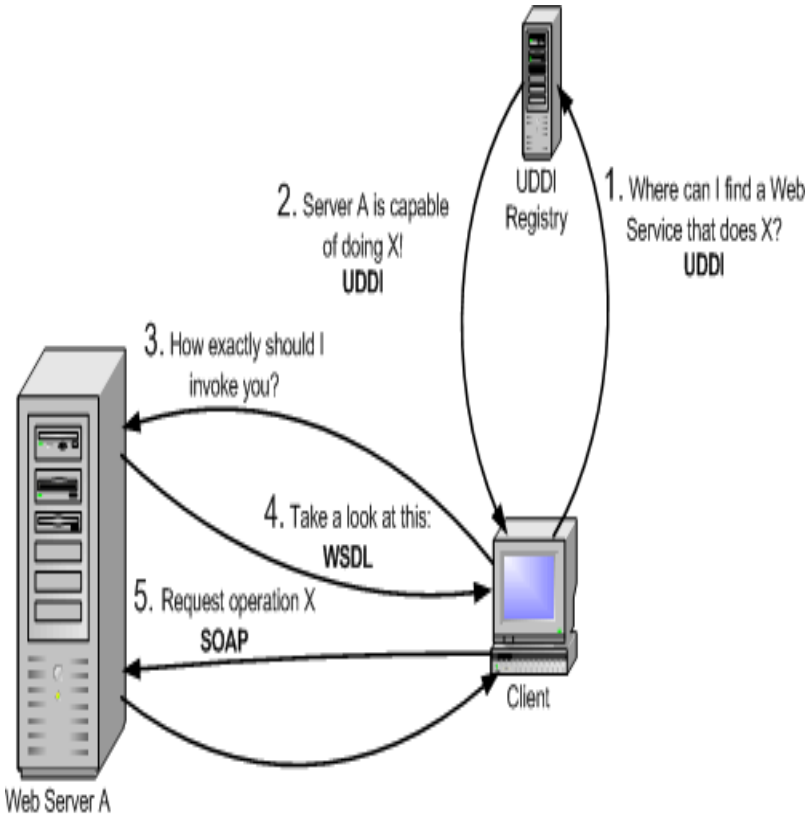
Source: Java™ Web Services Unleased

A Typical Web Service Invocation

1. First step will be to ***find*** a **Web Service** that meets our requirements: **contact a registry**.
2. The registry will reply, telling what servers **can** provide the service required.
3. the **location** of a Web Service is now **known**, but the actually invocation method is still unknown.

   The second step is to ask the Web Service to *describe* itself
4. The Web Service replies using WSDL.
5. The Web Service is **located** and **invocation** method is known.

   The invocation is done using SOAP.
6. The Web Service will reply with a ***SOAP response*** which includes the information we asked for, or an error message



Web Services also have some disadvantages

- **Overhead.** Transmitting all data in **XML** is not as efficient as using a proprietary binary code.
  - What you win in **portability**, you lose in **efficiency**.
  - This overhead is usually acceptable for most applications, but you will probably never find a critical real-time application that uses Web Services.
- **Lack of versatility**. Web Services are not very versatile, since they allow for very basic forms of service invocation.
  - Do not offer a standardized support for **persistency**, **notifications**, **lifecycle management**, **transactions**, etc.

- Service Oriented Architecture
- Web services
  - SOAP Based Web services
  -  RESTful Web service
- Usage of Web Service in Scientific applications

What is REST

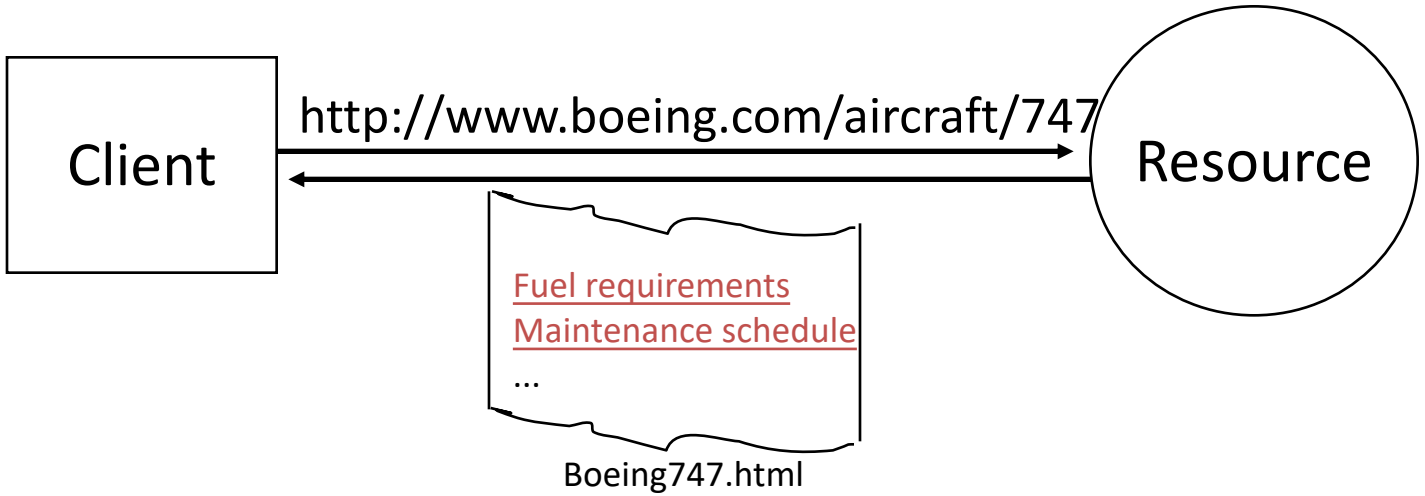
- REST stands for **Re**presentational **S**tate **T**ransfer.

- REST was coined by Roy Fielding in 2000 in his Ph.D. dissertation to describe **a design pattern for implementing networked systems**.

In many ways, the World Wide Web itself, based on HTTP, can be viewed as a REST-based architecture

[1] http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

Why is it called "Representational State Transfer? "



- The Client references a Web resource using a URL.
- A representation of the resource is returned (in this case as an HTML).
- The representation (e.g., Boeing747.html) places the client in a new state.

When the client selects a hyperlink in Boeing747.html, it accesses another resource.
The new representation places the client application into yet another state.
Thus, the client application transfers state with each resource representation.

Representational State Transfer

"Representational State Transfer is intended to evoke an image of **how a well-designed Web application behaves**: a network of web pages (a virtual state-machine), where the user progresses through an application by selecting links (**state transitions**), resulting in the next page (the next state of the application) being transferred to the user and rendered for their use."

- Roy Fielding

Roy Thomas Fielding, PhD DISSERTATION Chapter 5: Representational State Transfer
https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

REST - Not a Standard

- REST is NOT a standard
- REST is just a design pattern /an architecture style/
  - You can only understand and follow its principles/contraints.

- REST does prescribe the **use** of standards:
  - HTTP
  - URL
  - XML/HTML/GIF/JPEG/*etc.* **(Resource Representations)**
  - text/xml, text/html, image/gif, image/jpeg, *etc.* (Resource Types, MIME Types)

Six constraint of a RESTful Architecture

Constraints

- Uniform Interface
- Stateless
- Client-server
- Cachability
- Layered system
- code on demand

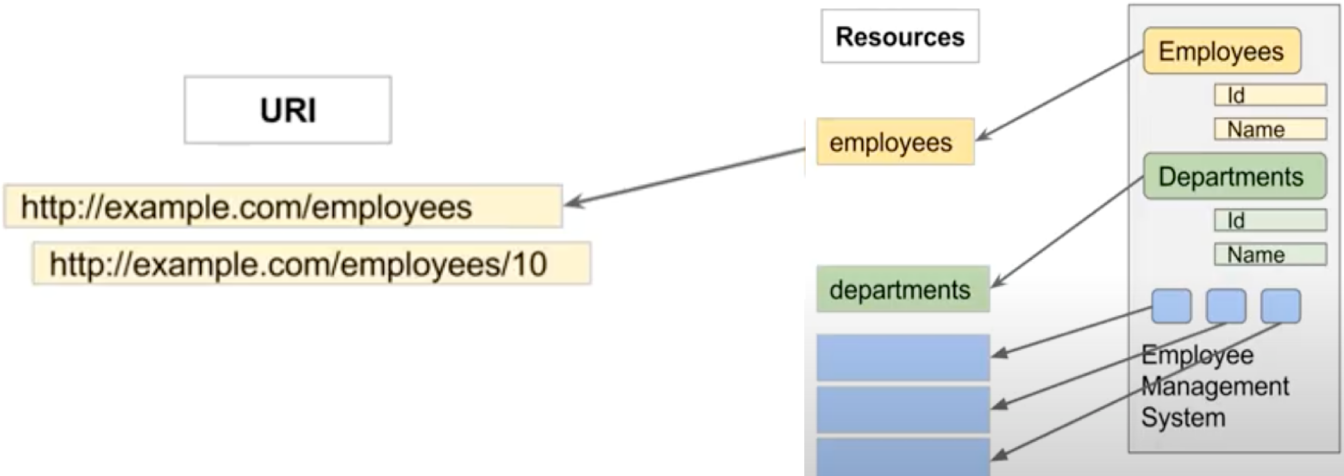
Features

- Scalability
- Simplicity
- Modifiability
- Visibility
- Portability
- Reliability

Roy Thomas Fielding, PhD DISSERTATION Chapter 5: Representational State Transfer
https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

Resource Based

- Resource is an object with a type, associated data, relationships to other resources, and a set of methods that operate on it
  - Identified by URIs
    - (Multiple URIs may refer to same resource)
  - separate from their representations



REST Fundamentals

1. Create a resource for every service.
2. Identify each resource using a URL.
3. The data that a Web service returns should link to other data.

Thus, design your data as a network of information.

The REST Design Pattern

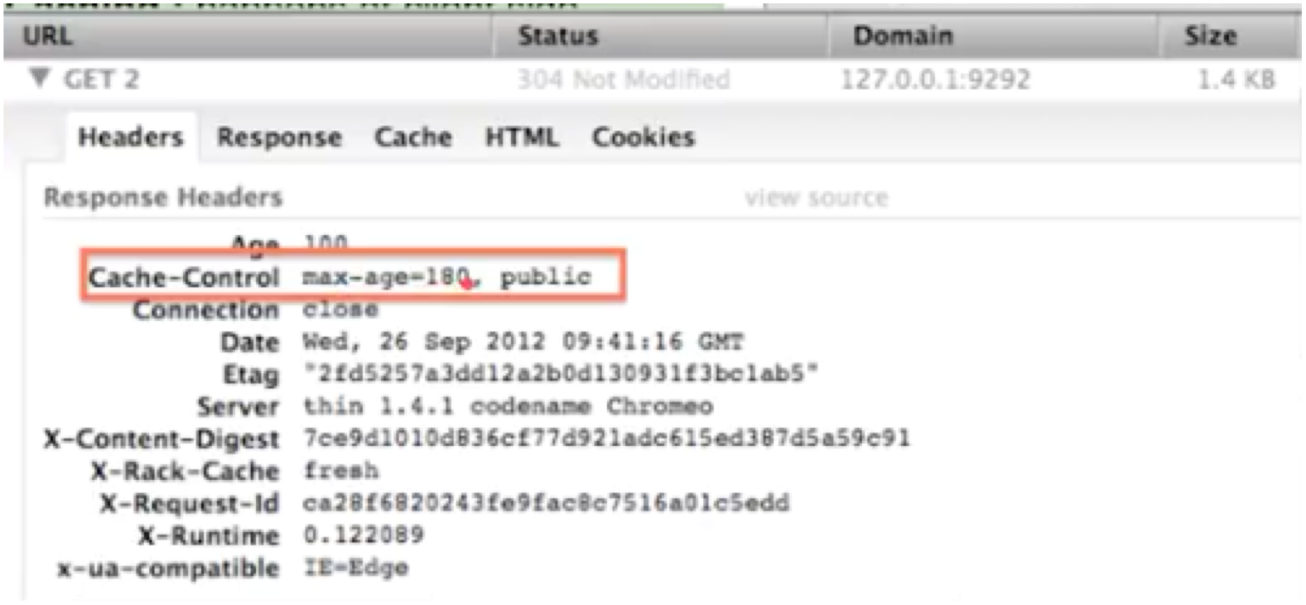
- Most web interactions are done using HTTP and just four operations:
  - Create information (HTTP PUT)
  - Retrieve information (HTTP GET)
  - Update information (HTTP POST)
  - Delete information (HTTP DELETE)

The REST Design Pattern (cont.)

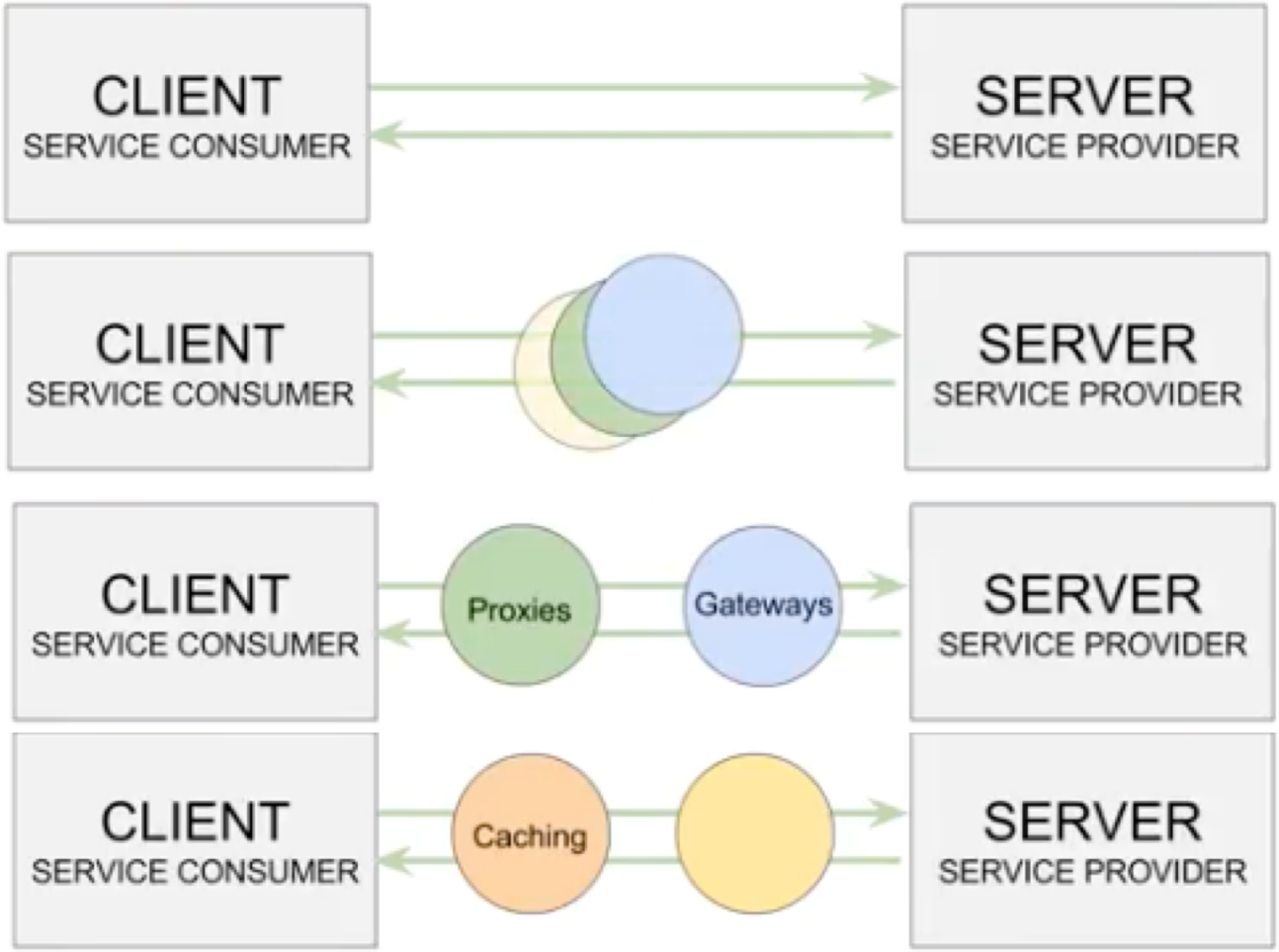
- **Web components** (firewalls, routers, caches) make their decisions based upon information in the **HTTP Header**.
- Consequently, the **destination URL** MUST be placed in the HTTP header for Web components to operate effectively.
  - Conversely, it is anti-REST if the HTTP header just identifies an intermediate destination and the payload identifies the final destination.

Cacheability

- Caching is done on the client side
- Following the information in the Response of server



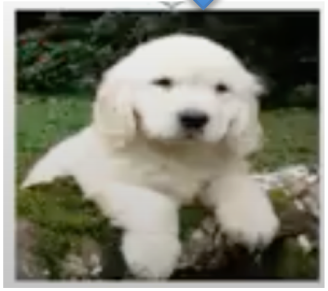
Layering


CLIENT
SERVICE CONSUMER
SERVER
SERVICE PROVIDER
CLIENT
SERVICE CONSUMER
SERVER
SERVICE PROVIDER
CLIENT
SERVICE CONSUMER
Proxies
Gateways
SERVER
SERVICE PROVIDER
CLIENT
SERVICE CONSUMER
Caching
SERVER
SERVICE PROVIDER

Resource Representations

GET /images/1
Accept : JPEG

Or GET /images/1.jpg



GET /images/1
Accept : XML

Or GET /images/1.xml

```
<pic>
 <name>mydog</name>
 <size>1080p</size>
</pic>
```

Real REST Examples

- Twitter has a REST API
  - https://dev.twitter.com/docs/api
- Flickr
  - http://www.flickr.com/services/api/
- Amazon.com offer several REST services, e.g., for their S3 storage solution

http://rest.elkstein.org/2008/02/real-rest-examples.html

Twitter has a REST API

Tweets

Tweets are the atomic building blocks of Twitter, 140-character status updates with additional associated metadata. People tweet for a variety of reasons about a multitude of topics.

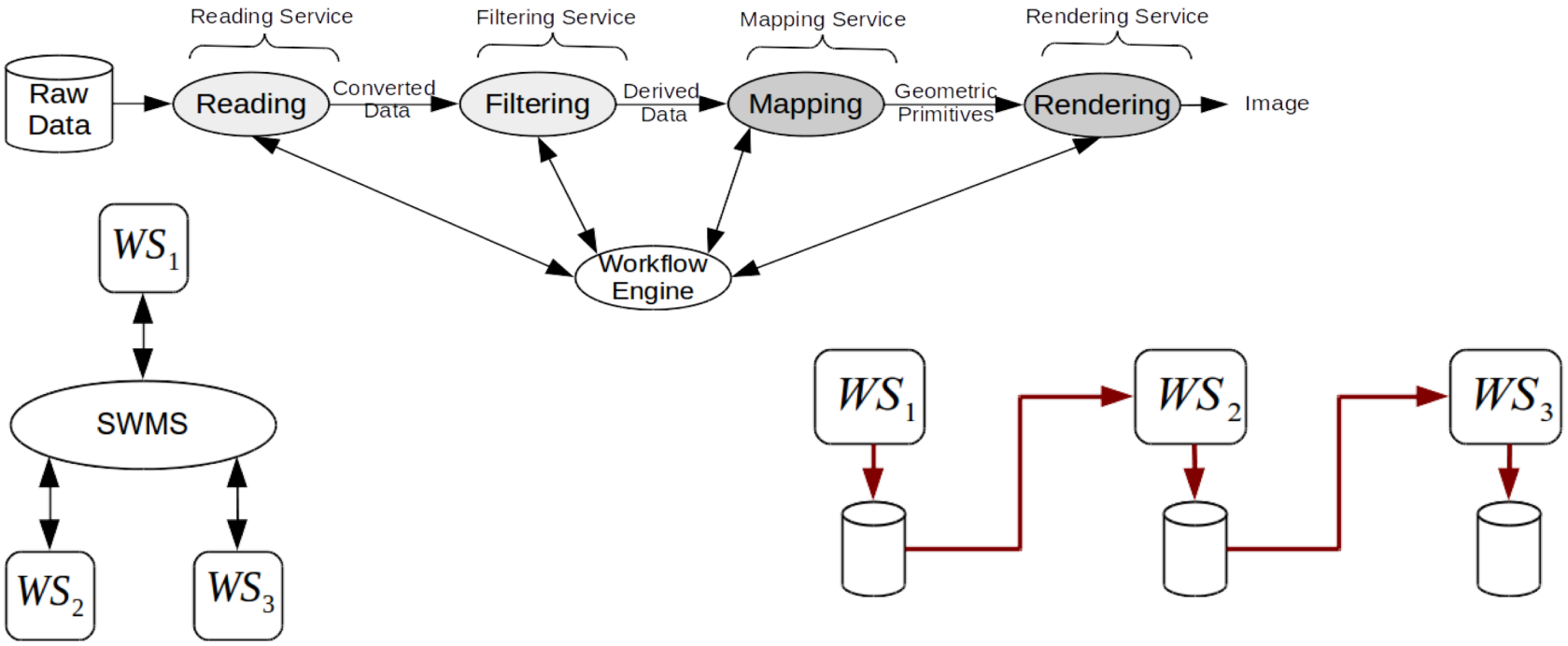
Resource	Description
GET statuses/retweets/:id	Returns up to 100 of the first retweets of a given tweet.
GET statuses/show/:id	Returns a single Tweet, specified by the id parameter. The Tweet's author will also be embedded within the tweet. See Embeddable Timelines, Embeddable Tweets, and GET statuses/oembed for tools to render Tweets according to Display Requirements.
POST statuses/destroy/:id	Destroys the status specified by the required ID parameter. The authenticating user must be the author of the specified status. Returns the destroyed status if successful.
POST statuses/update	Updates the authenticating user's current status, also known as tweeting. To upload an image to accompany the tweet, use POST statuses/update_with_media. For each update attempt, the update text is compared with the authenticating user's recent tweets. Any attempt that would result in duplication...
POST statuses/retweet/:id	Retweets a tweet. Returns the original tweet with retweet details embedded.
POST statuses/update_with_media	Updates the authenticating user's current status and attaches media for upload. In other words, it creates a Tweet with a picture attached. Unlike POST statuses/update, this method expects raw multipart data. Your POST request's Content-Type should be set to multipart/form-data with the media[]...
GET statuses/oembed	Returns information allowing the creation of an embedded representation of a Tweet on third party sites. See the oEmbed specification for information about the response format. While this endpoint allows a bit of customization for the final appearance of the embedded Tweet, be aware that the...

Content

- Service Oriented Architecture
- Web services
  - SOAP Based Web services
  - RESTful Web service
- Example of usage of Web Service in Scientific applications

Usage of Web Services in e-science

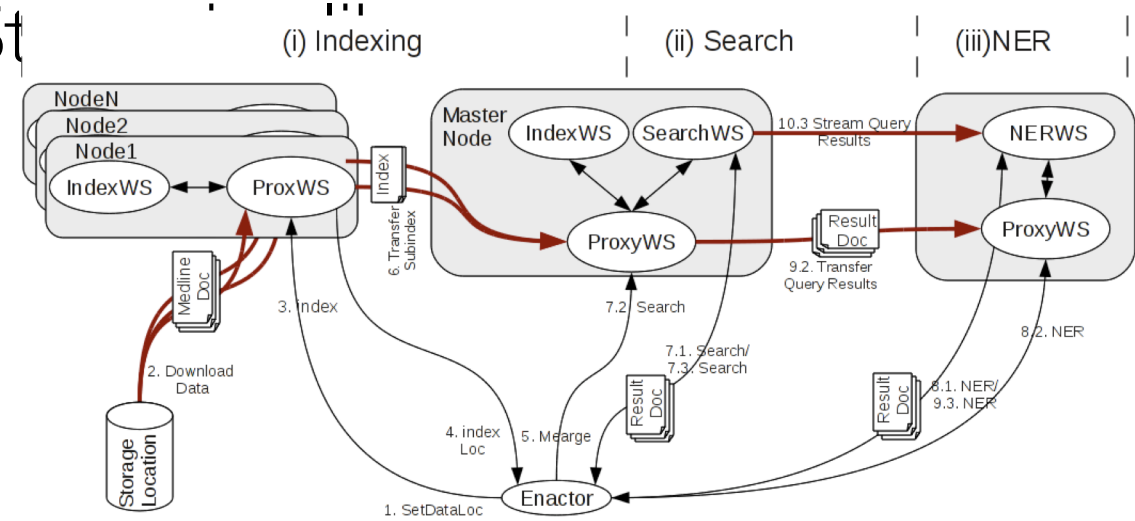
- Constraints:
  - Service orchestration, **all data is passed to the workflow engine** before delivered to a consuming WS
  - Data transfers are made through **SOAP**, which **is unfit for large data transfers**



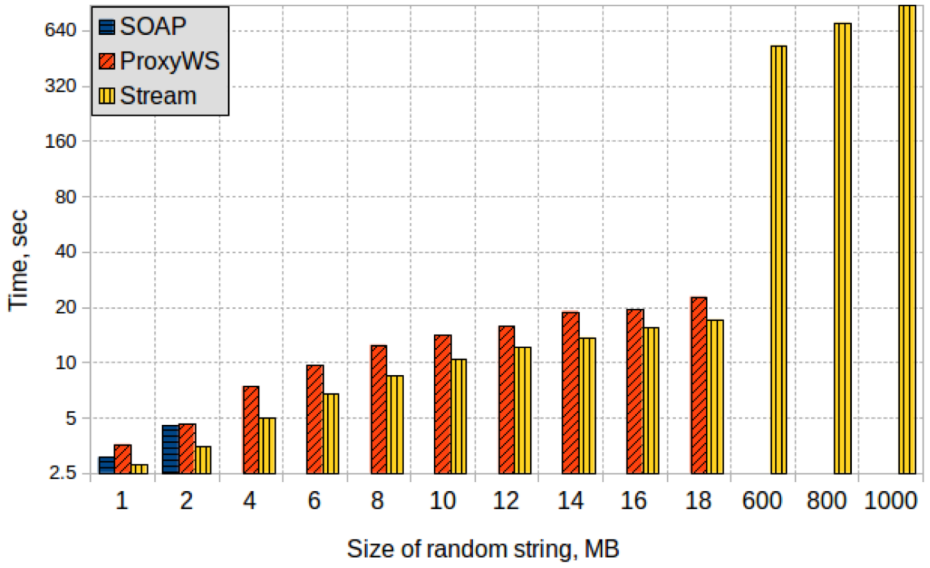
Enabling web services to consume and produce large distributed *datasets Spiros Koulouzis, Reginald Cushing, Konstantinos Karasavvas, Adam Belloum, Marian Bubak to be published JAN/FEB, IEEE Internet Computing, 2012*

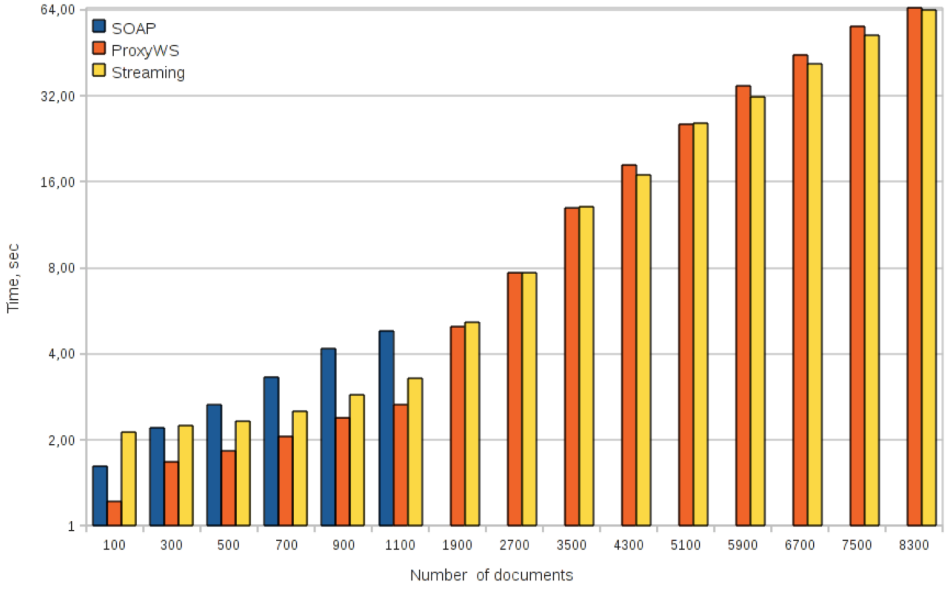
Indexing Name Entry Recognition

- AIDA provides a set of components which enable the indexing of text documents in various formats.
- AIDA's Indexer component, called IndexerWS is a WS able to index document with the use of the St



Results Indexing Web Services for Information Retrieval (NER)





Enabling web services to consume and produce large distributed *datasets Spiros Koulouzis, Reginald Cushing, Konstantinos Karasavvas, Adam Belloum, Marian Bubak to be published JAN/FEB, IEEE Internet Computing, 2012*