Desarrollo de aplicaciones distribuidas: Tendencias

Grupo ARCOS

Desarrollo de Aplicaciones Distribuidas

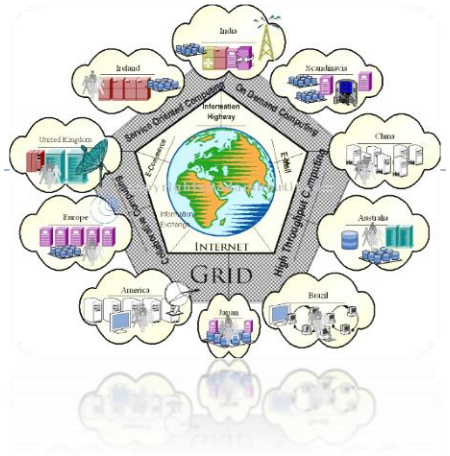
Ingeniería Informática

Universidad Carlos III de Madrid

Contenidos

1. Plataformas:
   1. Supercomputadores
   2. Clusters
   3. Grid
   4. *Cloud computing*
2. Entornos de desarrollo
   1. JSON+REST vs XML+SOAP
   2. Peer-to-peer y redes sociales

Contenidos



1. **Plataformas:**
   1. **Supercomputadores**
   2. **Clusters**
   3. **Grid**
   4. *Cloud computing*
2. Entornos de desarrollo
   1. JSON+REST vs XML+SOAP
   2. Peer-to-peer y redes sociales

Se incrementa la necesidad de cómputo…

- Supercomputación distribuida (*Distributed supercomputing*)
- Computación de alta productividad (*High throughput computing*)
- Computación bajo demanda (*On-demand computing*)
- Computación intensiva de datos (*Data-intensive computing*)
- Computación colaborativa (*Collaborative computing*)
- Computación multimedia (*Multimedia computing*)

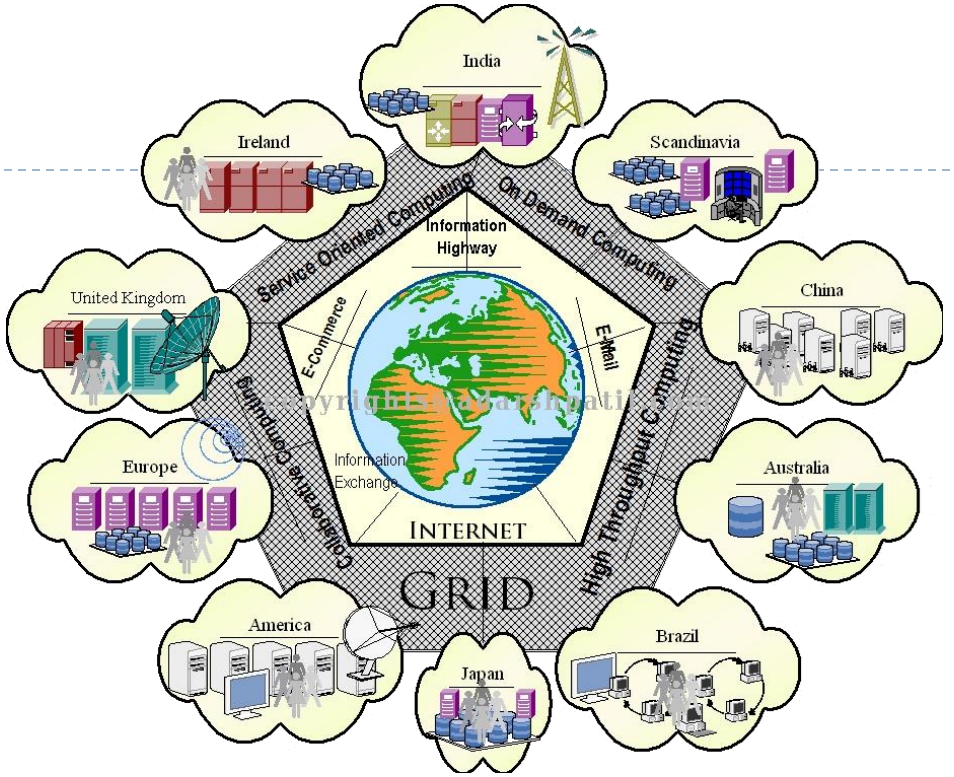
Supercomputadores

- Evolución de los sistemas de altas prestaciones:
  - Inicialmente: uso de supercomputadores
  - Sistemas **centralizados** basados en multiprocesadores o multicomputadores
  - Caros y difíciles de ampliar

Clusters

- Evolución de los sistemas de altas prestaciones:
  - Siguiente: *clusters*
  - Sistema **centralizado** basado en la agrupación de computadores genéricos de forma barata y ampliable
  - Necesidades de electricidad, refrigeración y espacio para un mayor número de elementos de cómputo

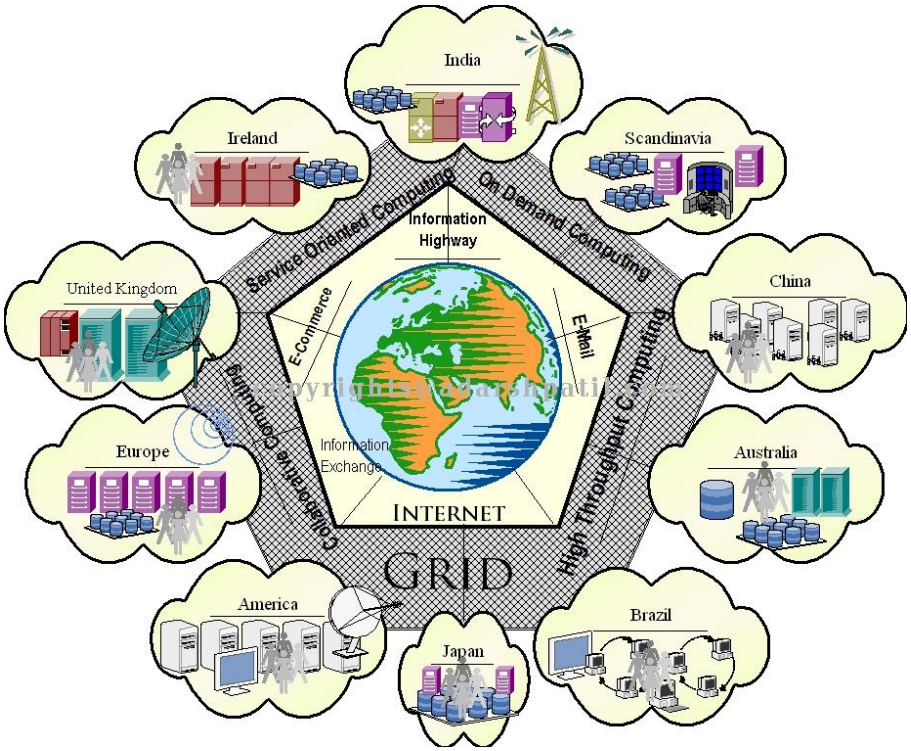
Grid (1/2)



- Evolución de los sistemas de altas prestaciones:
  - Siguiente: *Grid*
  - Sistema **distribuido** de *clusters* y otros recursos federados que permite construir un sistema con la suma de la potencia de sus recursos
  - Se facilita compartir recursos: los usuarios ven y pueden acceder a los diferentes recursos compartidos (incluyendo los de su plataforma)

http://www.adarshpatil.com/pictures/Cloud_Grid_Computing_adarsh.jpg

Grid (2/2)



- Tecnología cuyo objetivo es la compartición de recursos en Internet de forma uniforme, transparente, segura, eficiente y fiable
  - **Ofrecen** un **único punto de acceso** a un conjunto de recursos distribuidos
  - Geográficamente **en diferentes dominios de administración**

http://www.adarshpatil.com/pictures/Cloud_Grid_Computing_adarsh.jpg

Contenidos

1. **Plataformas:**
   1. Supercomputadores
   2. Clusters
   3. Grid
   4. ***Cloud computing***
2. Entornos de desarrollo
   1. JSON+REST vs XML+SOAP
   2. Peer-to-peer y redes sociales

Se incrementa la necesidad de cómputo…

- Supercomputación distribuida (*Distributed supercomputing*)
- Computación de alta productividad (*High throughput computing*)
- Computación bajo demanda (*On-demand computing*)
- Computación intensiva de datos (*Data-intensive computing*)
- Computación colaborativa (*Collaborative computing*)
- Computación multimedia (*Multimedia computing*)

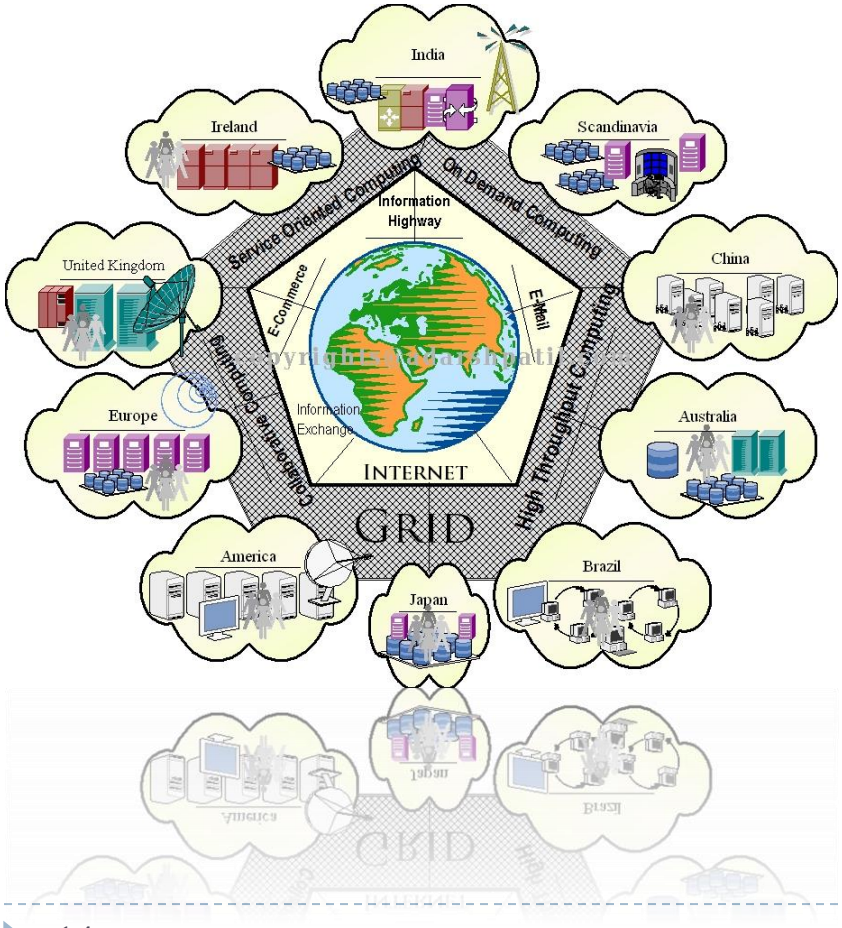
Ej.: ¿Qué ropa ponernos hoy?

Ej.: ¿Qué ropa ponernos hoy?

- Tenemos un armario con diversa ropa para las ocasiones que se nos presenta:
  - Verano, invierno, …
  - Trabajo, deporte, noche, …
- Seleccionamos la apropiada
- Fácil adaptarse con un buen fondo de armario
- Clave: actualización del armario
  - Compra de nueva ropa
  - Alquiler de ropa ocasional
  - Venta/donación de ropa usada

¿Qué entorno usaremos hoy?

¿Qué entorno usaremos hoy?

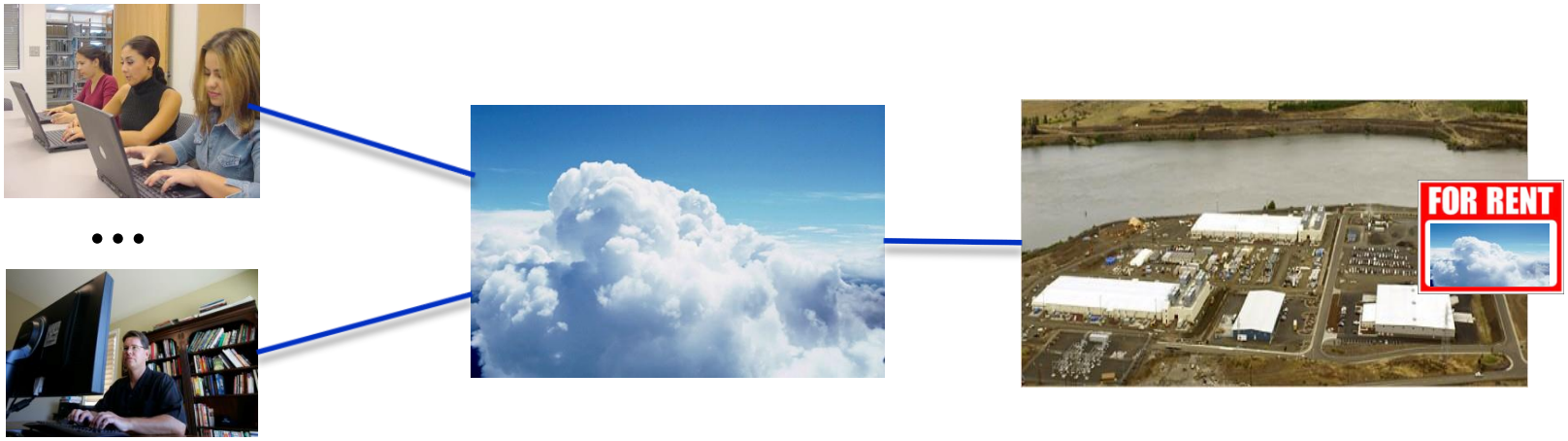


- Hay diversos recursos distribuidos que pueden ser accedidos
- Se busca usar el apropiado
- Fácil adaptarse si se tiene un Grid rico en recursos
- Problemas: adaptación del Grid a nuevos requisitos
- Evolución gracias a *utility computing* y virtualización

Mercado de cómputo (*utility computing*)

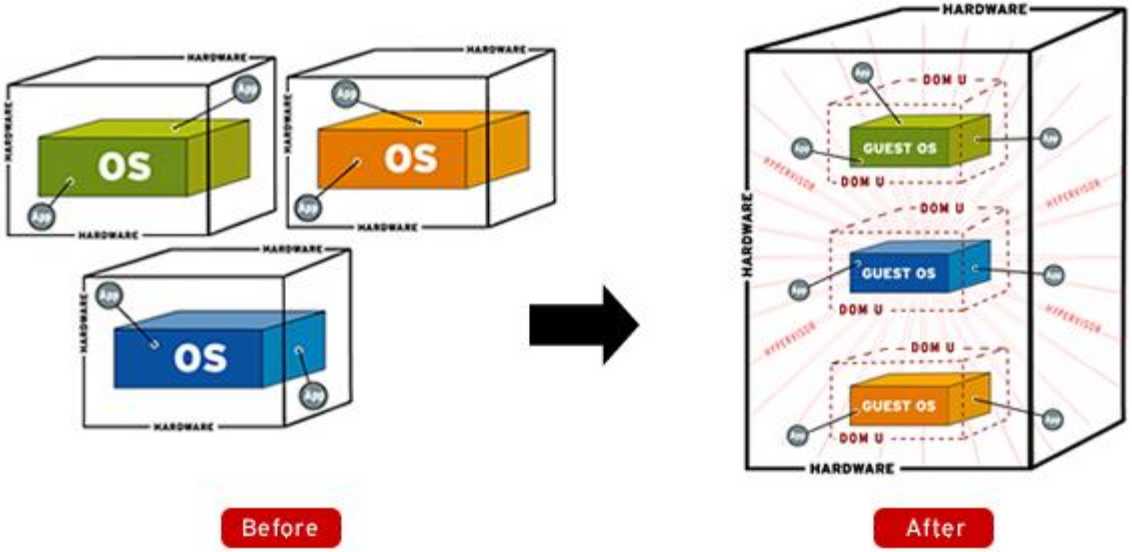
- Empresas con **exceso** de capacidad de cómputo pueden de forma rentable dejar usar sus sistemas a distintos clientes

Mercado de cómputo (*utility computing*)



- Empresas con **exceso** de capacidad de cómputo pueden de forma rentable dejar usar sus sistemas a distintos clientes
- Empresas con **demanda** de capacidad de cómputo pueden buscar alquilar la infraestructura de quién le ofrezca mejor precio o servicio (o relación entre ellos)
  - No hay que pagar por construir grandes centros de datos
  - No hay que pagar por la compleja administración de sistemas
  - No hay que pagar el elevado consumo eléctrico

Máquinas virtuales (virtualización)

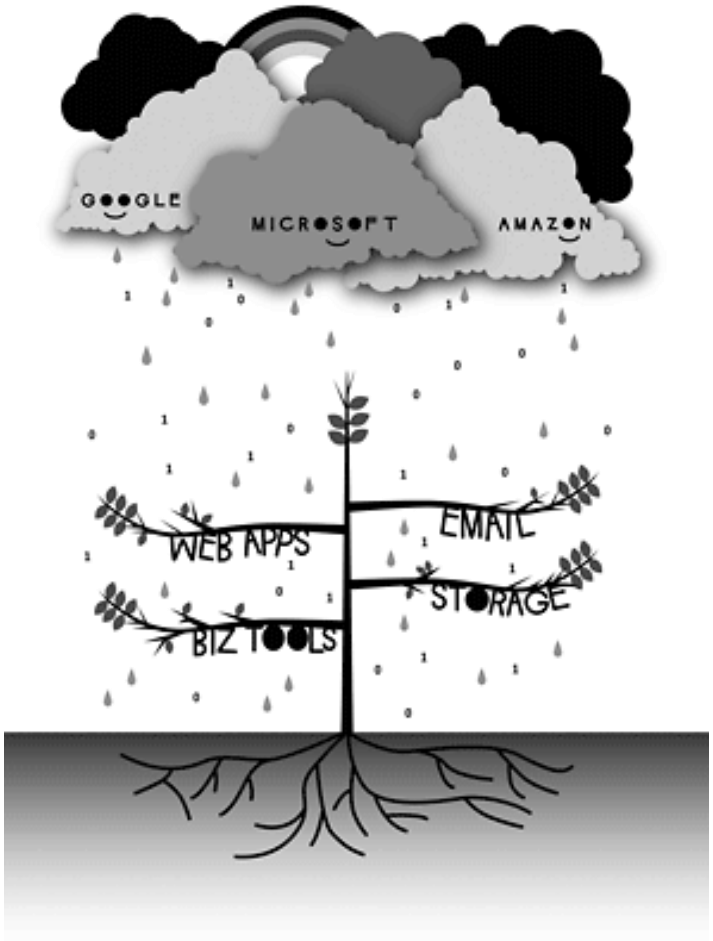


- Posibilidad de ejecutar en un computador (*host*) un programa que crea un computador virtual (*guest*) sobre el que ejecutar cualquier entorno.

Máquina virtual

- Posibilidad de ejecutar en un computador (*host*) un programa que crea un computador virtual (*guest*) sobre el que ejecutar cualquier entorno.

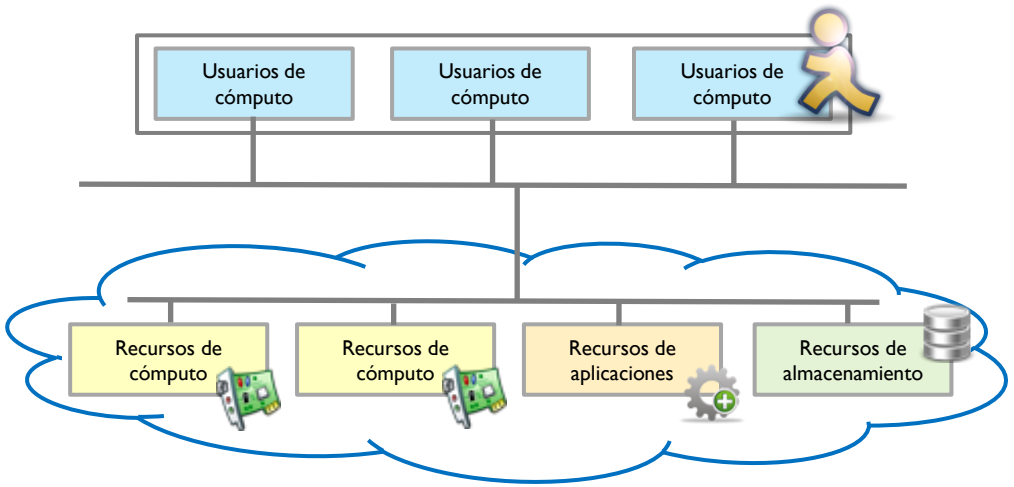
Cloud computing



http://blogs.msdn.com/jim_glass/archive/2008/10/27/microsoft-s-cloud-computing-strategy.aspx

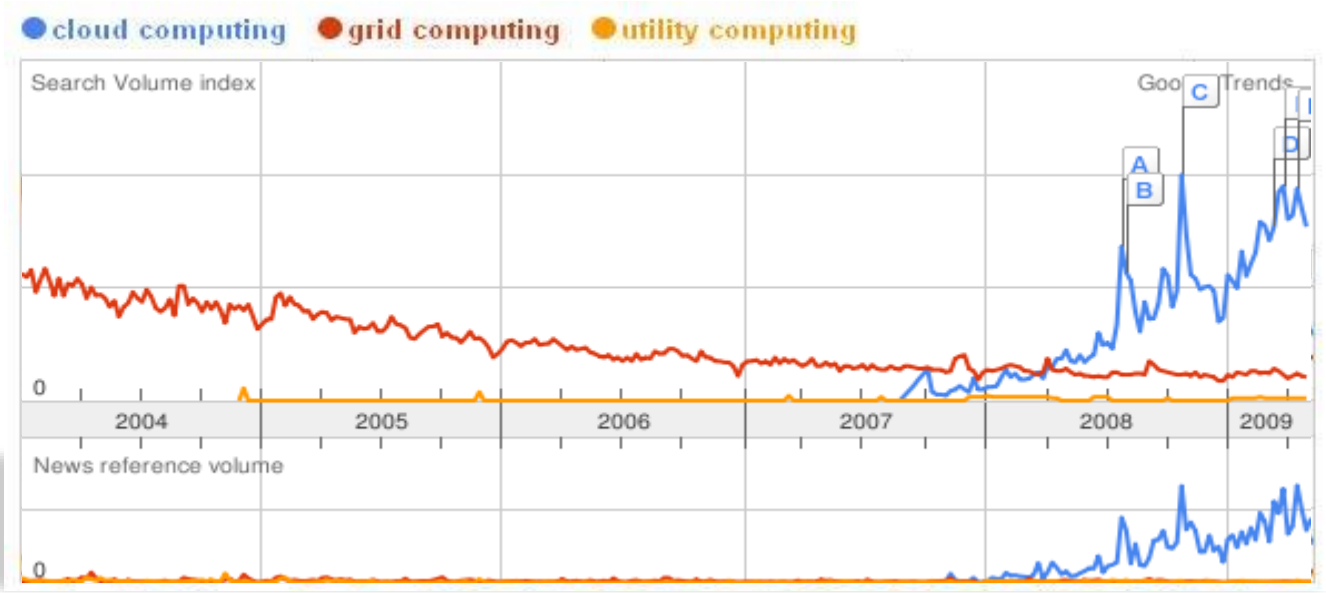
Definición inicial

- *Cloud computing* puede definirse de forma vaga como el uso de recursos computacionales escalables ofrecidos como un servicio desde fuera del entorno que los usa, a través de pago por uso.



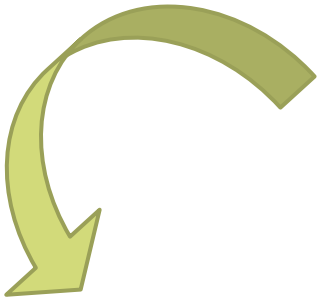
- Principales ventajas:
  - Solo se usa lo que se necesita y se paga solo por lo usado
  - Se puede acceder a cualquiera de los recursos que están en la nube en cualquier momento y desde cualquier sitio de Internet

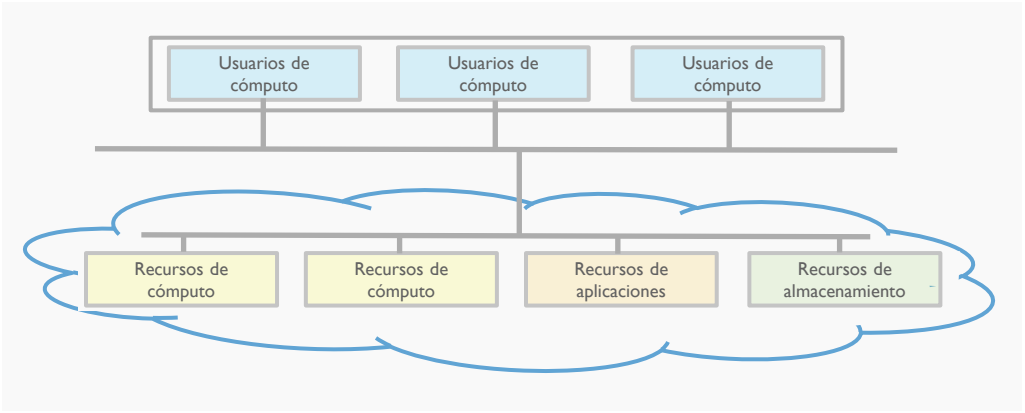
Evolución de *cloud computing*

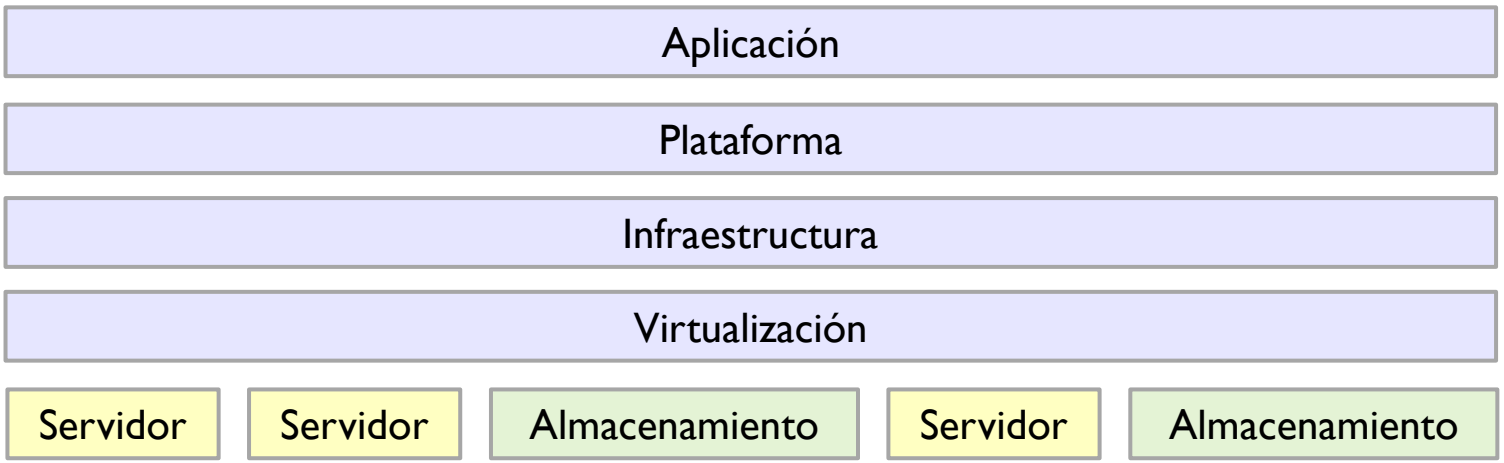


- Google trend nos da una pista de la evolución del término 'cloud computing' en las búsquedas
  - Más uso día a día

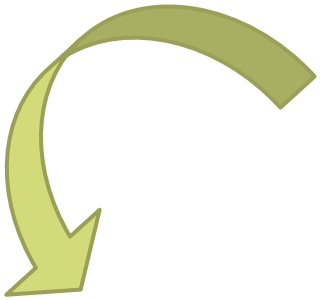
Anatomía de un *cloud computing*

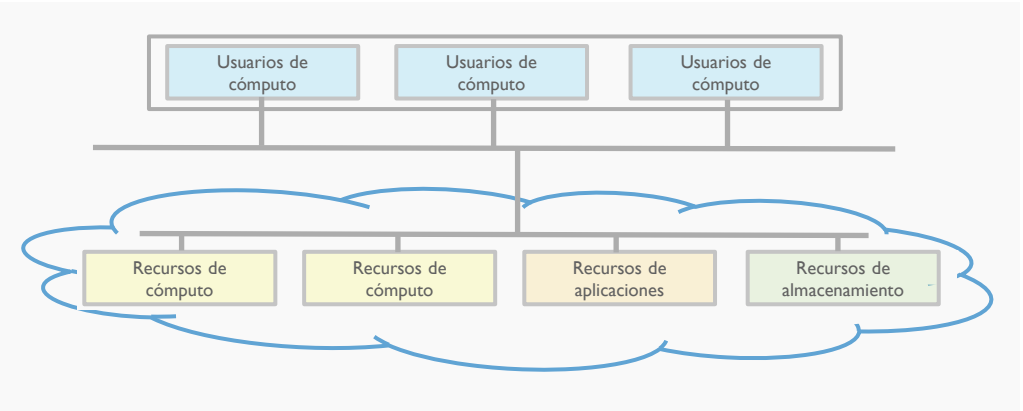






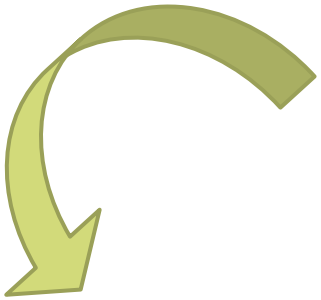
Anatomía de un *cloud computing*

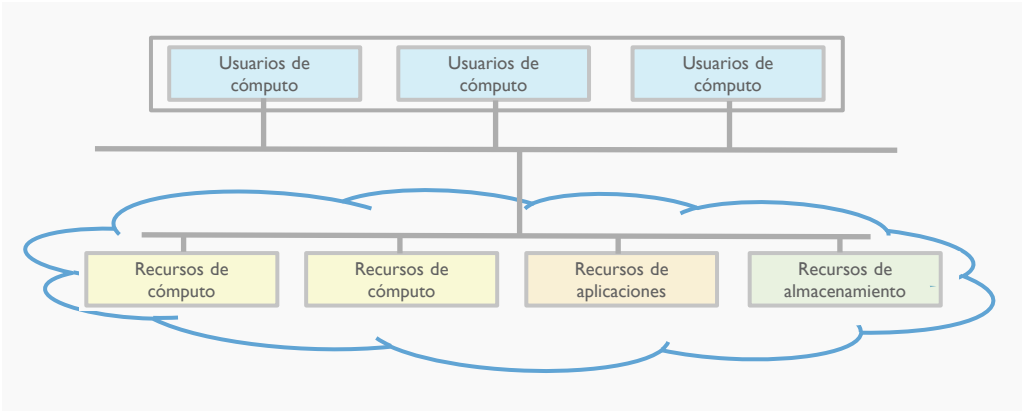




SaaS	Software como servicio	Google Apps, Microsoft “Software+Services”
PaaS	Plataforma como servicio	Google AppEngine, IBM IT Factory, Force.com
laaS	Infraestructura como servicio	Amazon EC2, Sun Grid, IBM Blue Cloud
dSaaS	Almacenamiento como servicio	Amazon S3, Nirvanix SDN, Cleversafe dsNet

Anatomía de un *cloud computing*





SaaS	Software como servicio	Google Apps, Microsoft "Software+Services"
PaaS	Plataforma como servicio	Google AppEngine, IBM IT Factory, Force.com
laaS	Infraestructura como servicio	Amazon EC2, Sun Grid, IBM Blue Cloud
dSaaS	Almacenamiento como servicio	Amazon S3, Nirvanix SDN, Cleversafe dsNet

- Servicio de almacenamiento muy simple de alta disponibilidad
- Tarificado por transferencia y almacenamiento

Amazon S3 (Simple Storage Service)

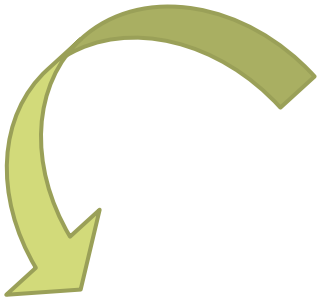
http://aws.amazon.com/s3/

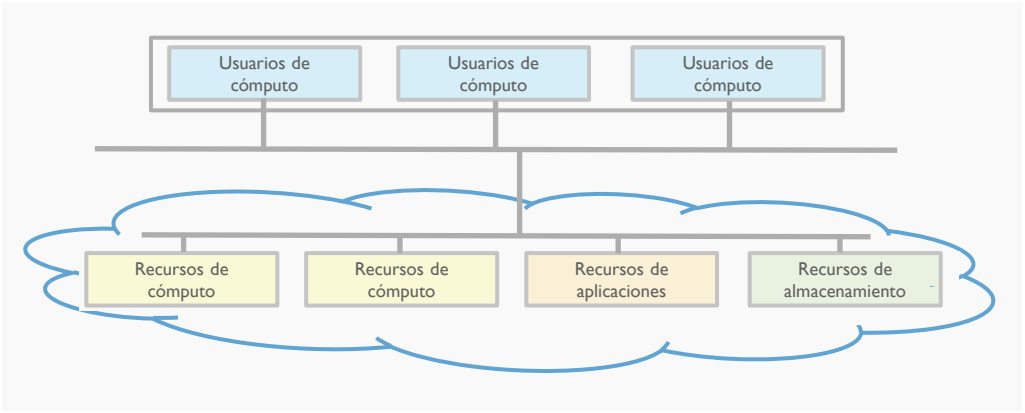
- El servicio S3 se factura por tres conceptos conjuntamente:
  - Cantidad almacenada:
    - Hay una tarifa por GB almacenado/mes.
    - Cuanto más almacenemos en S3, más pagamos.
  - Transferencia de Datos:
    - Hay otra tarifa por los GB transferidos.
    - Nos costará más barato cuanto más transfiramos.
  - Peticiones de acceso:
    - Tarifa por las peticiones sobre los ficheros (GET, PUT, LIST, etc.).
- A pesar de todo, los servicios de Amazon son competitivos:
  - Unos 2,5 GB de datos almacenados y una transferencia de 15GB al mes, no llegarán a los 4 dólares (2,69€) mensuales.

    http://www.maestrosdelweb.com/editorial/por-que-utilizar-s3-el-sistema-de-almacenamiento-de-amazon/
  - Smugmug afirma ahorrar 1 millón de dólares en doce meses usando Amazon S3

    http://www.error500.net/amazon-s3-ahorro-costes

Anatomía de un *cloud computing*





SaaS	Software como servicio	Google Apps, Microsoft "Software+Services"
PaaS	Plataforma como servicio	Google AppEngine, IBM IT Factory, Force.com
laaS	Infraestructura como servicio	Amazon EC2, Sun Grid, IBM Blue Cloud
dSaaS	Almacenamiento como servicio	Amazon S3, Nirvanix SDN, Cleversafe dsNet

- Una plataforma virtual completa que NO incluye toda la pila de software (como en PaaS)
- Sería posible administrar (el software de sistema) de la plataforma que se demanda

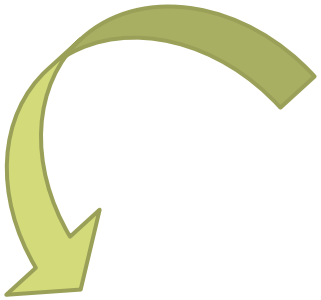
Amazon EC2 (Elastic Compute Cloud)

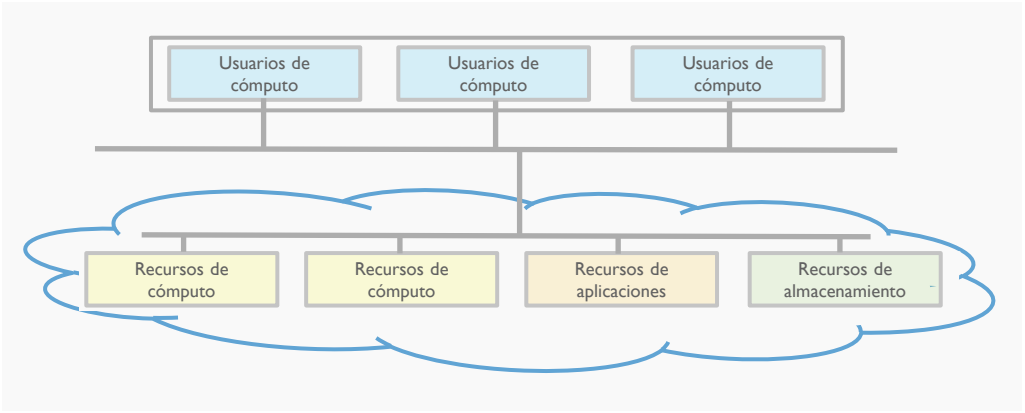
http://aws.amazon.com/ec2/

- El servicio EC2 se factura la capacidad de cómputo deseada:
  - Instancias de máquinas virtuales desplegadas:
    - Distintos tipos de instancias: pequeña, grande, extra-grande, etc..
  - Transferencia de Datos:
    - Tarifa por los datos transferidos a/desde EC2.
  - Almacenamiento:
    - Tarifa por las peticiones sobre los ficheros (GET, PUT, LIST, etc.).
  - Servicios adicionales:
    - Almacenamiento, monitorización, direcciones IP, reparto de carga, etc.
- Usos diversos:
  - Aplicaciones, Almacenamiento, distribución de contenido, comercio electrónico, etc.

    http://aws.amazon.com/solutions/case-studies/

Anatomía de un *cloud computing*





SaaS	Software como servicio	Google Apps, Microsoft "Software+Services"
PaaS	Plataforma como servicio	Google AppEngine, IBM IT Factory, Force.com
laaS	Infraestructura como servicio	Amazon EC2, Sun Grid, IBM Blue Cloud
dSaaS	Almacenamiento como servicio	Amazon S3, Nirvanix SDN, Cleversafe dsNet

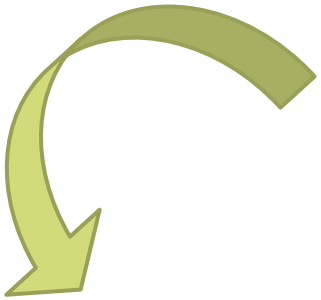
- Una plataforma virtual completa ('*outsourcing*' de la infraestructura de una empresa)
- Incluye servidores, sistemas operativos y aplicaciones específicas

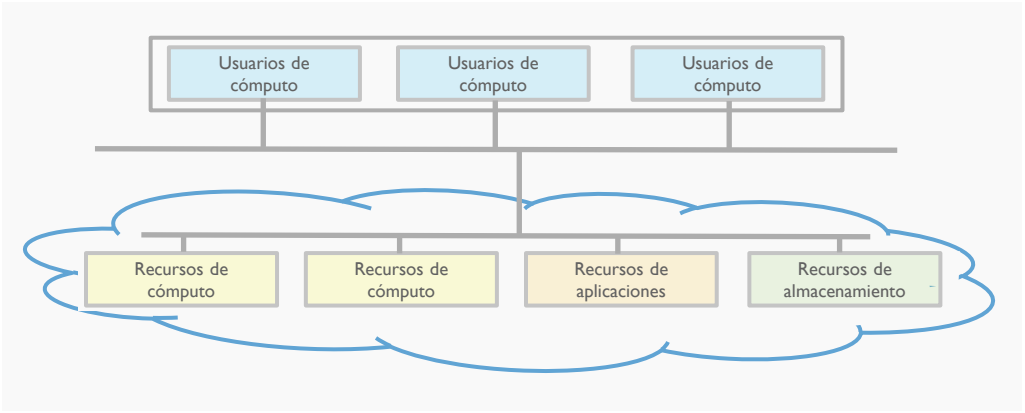
Google App Engine

code.google.com/appengine/

- *Google App Engine* permite ejecutar aplicaciones Web en los servidores de Google:
  - Dentro de la gigante granja de servidores de Google, lo que facilita escalabilidad
  - Integrable con otras aplicaciones de Google (ejemplo: utilizar la autenticación de Google en nuestras aplicaciones)
- Actualmente las aplicaciones Google App Engine se implementan mediante:
  - Los lenguajes de programación Python y Java.
  - El sistema de almacenamiento GQL (similar a SQL)
- Respuesta de Google a:
  - Amazon Web Services, Microsoft Azure Services, Heroku, etc.

Anatomía de un *cloud computing*





SaaS	Software como servicio	Google Apps, Microsoft "Software+Services"
PaaS	Plataforma como servicio	Google AppEngine, IBM IT Factory, Force.com
laaS	Infraestructura como servicio	Amazon EC2, Sun Grid, IBM Blue Cloud
dSaaS	Almacenamiento como servicio	Amazon S3, Nirvanix SDN, Cleversafe dsNet

- Un ejemplo inicial se encuentra en los antiguos ASP (*Application Service Provider*)
- Otro ejemplo: software ofrecido en forma de servicio Web que es usado por una aplicación local

Google Apps

Google apps
www.google.com/apps/

- Permite realizar trabajo colaborativo con un grupo de trabajo:
  - Colaboración en un mismo documento,
    en lugar de indicar cambios en documentos adjuntos.
  - Compartición de documentos y calendarios con compañeros de trabajo.
  - Acceso a toda la información desde cualquier equipo.
  - Invitación a miembros del equipo a unirse al servicios no compleja.

- Actualmente integra las siguientes aplicaciones colaborativas:

  - **Google Docs**: para crea documentos, hojas de cálculo y presentaciones.

  - **Google Calendar**: permite organiza y publicar eventos (Ej.: reuniones).

  - **Google Talk**: para envía mensajes instantáneos entre miembros del grupo.

Contenidos

1. Plataformas:
   1. Supercomputadores
   2. Clusters
   3. Grid
   4. *Cloud computing*
2. **Entornos de desarrollo**
   1. **JSON+REST vs XML+SOAP**
   2. Peer-to-peer y redes sociales

JSON

- JSON, acrónimo de *JavaScript Object Notation*
- Es un formato ligero para el intercambio de datos
  - JSON es un subconjunto de la notación literal de objetos de JavaScript
  - Es una alternativa simple y ligera de XML puesto que un analizador léxico, sintáctico y semántico es mucho más sencillo.
  - Dada su integración en JavaScript, es fácil su uso con AJAX puesto que con la función eval() es simple recrear el objeto representado por JSON

http://es.wikipedia.org/wiki/JSON

JSON vs. XML

- JSON:

```
{"menu": {
  "id": "file",
  "value": "File",
  "popup": {
    "menuitem": [
      {"value": "New", "onclick": "CreateNewDoc()"},
      {"value": "Open", "onclick": "OpenDoc()"},
      {"value": "Close", "onclick": "CloseDoc()"}
    ]
  }
}}
```

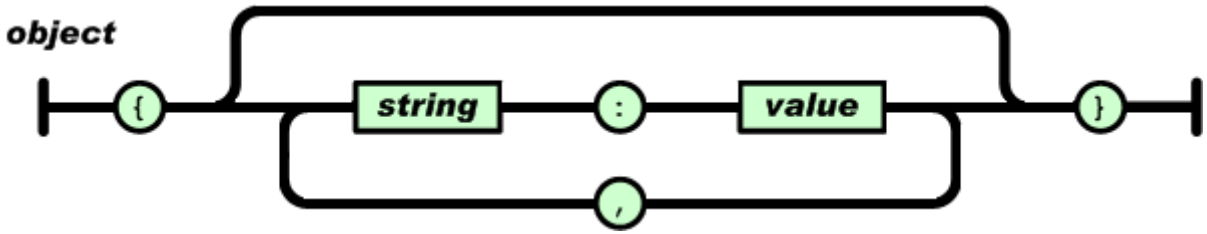
- XML:

```
<menu id="file" value="File">
  <popup>
    <menuitem value="New" onclick="CreateNewDoc()" />
    <menuitem value="Open" onclick="OpenDoc()" />
    <menuitem value="Close" onclick="CloseDoc()" />
  </popup>
</menu>
```

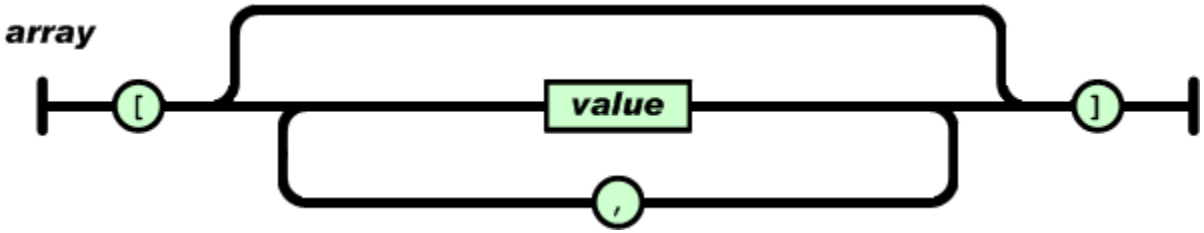

http://es.wikipedia.org/wiki/JSON

JSON: formato (1/4)

- Objeto:

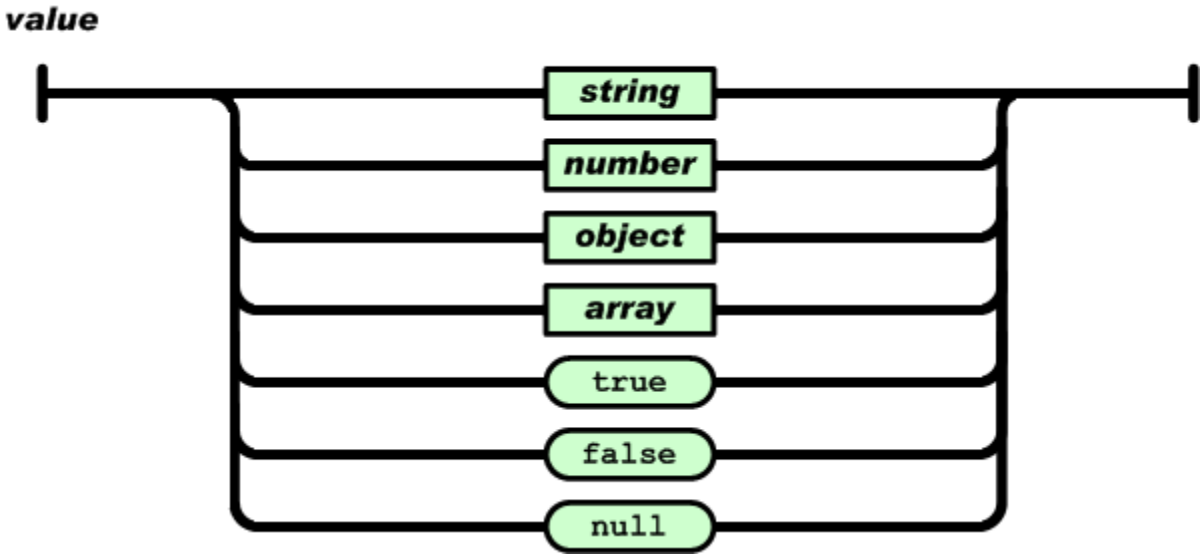


- Array:



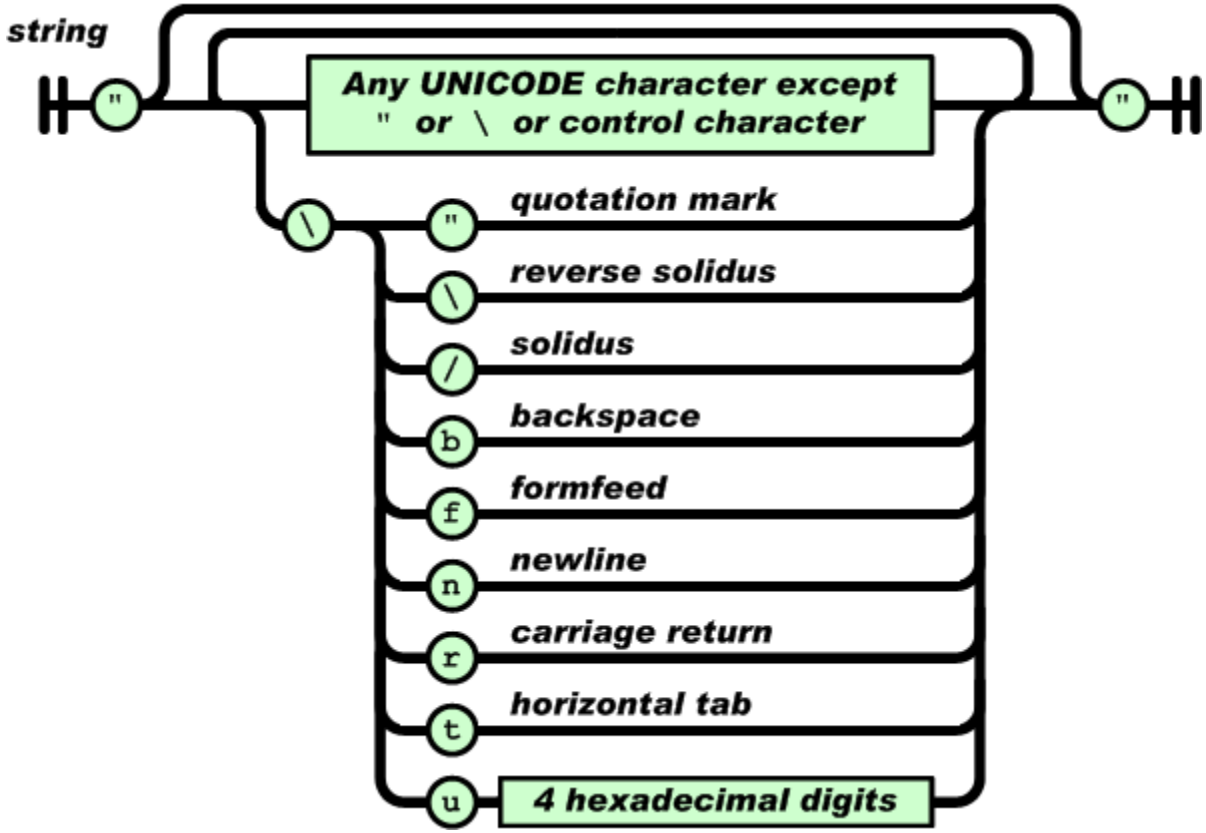
JSON: formato (2/4)

- Valor:



JSON: formato (3/4)

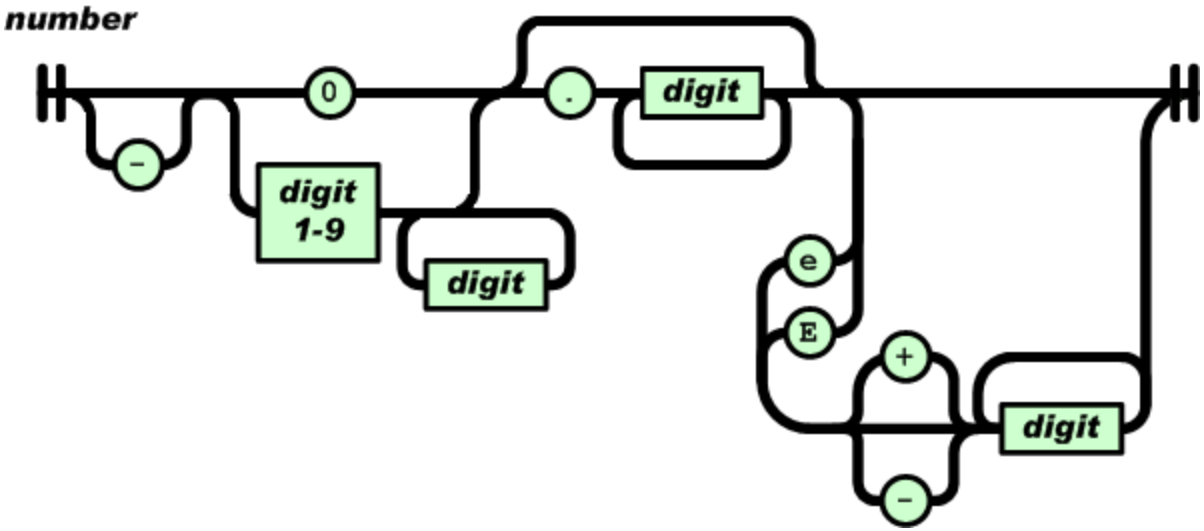
- String:



http://www.json.org/

JSON: formato (4/4)

- Número:



JSON: procesamiento

- De objeto *JavaScript* a JSON:

```
var myJSONText = JSON.stringify(myObject, replacer);
```

- De JSON a *JavaScript*:

```
var myObject = eval('(' + myJSONtext + ')');
```

```
var myObject = JSON.parse(myJSONtext);
```


http://json.org/js.html

Ejemplo de *JavaScript* que usa JSON…

```
var the_object = {};
var http_request = new XMLHttpRequest();

http_request.open( "GET", url, true );

http_request.onreadystatechange = function ()
{
    if ( http_request.readyState == 4 && http_request.status == 200 )
      {
          the_object = JSON.parse( http_request.responseText );
      }
      http_request = null;
};

http_request.send(null);
```


http://json.org/js.html

Ejemplo de JSP que genera JSON…

JSON-JSPExample.jsp

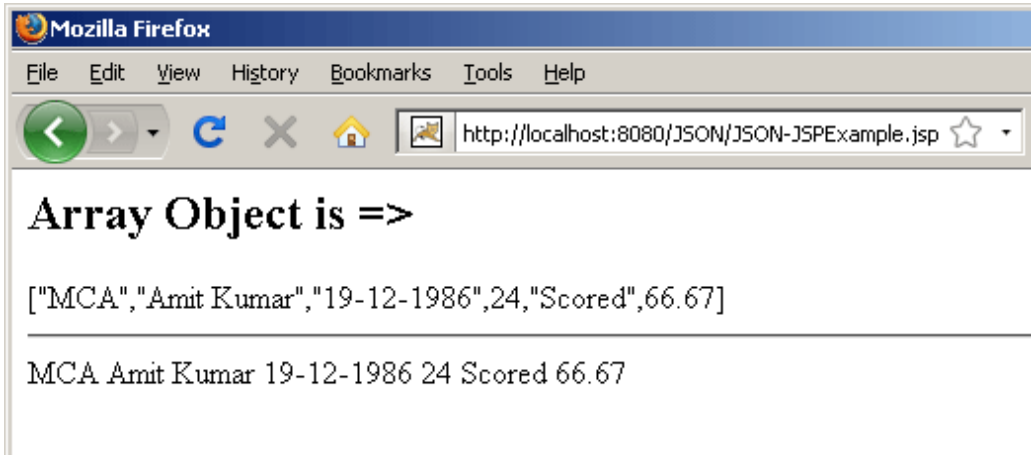
```
<%@ page language="java" import="net.sf.json.JSONArray" %>

 <%
   JSONArray arrayObj = new JSONArray();
   arrayObj.add("MCA");
   arrayObj.add("Amit Kumar");
   arrayObj.add("19-12-1986");
   arrayObj.add(24);
   arrayObj.add("Scored");
   arrayObj.add(new Double(66.67));
%>
<h2>Array Object is =></h2> <%=arrayObj%>
<br><hr>
<%  for(int i=0;i<arrayObj.size();i++){  %>
        <%=arrayObj.getString(i)%>
<%
  }
%>
```


http://www.roseindia.net/tutorials/json/json-servlet-example.shtml

Ejemplo de JSP que genera JSON…

- Crear el archivo JSON-JSPExample.jsp
- Colocarlo en el directorio WEB-INF
- Obtener las JSONLibraries y situarlas en el directorio lib de Tomcat
- Arrancar el servidor Tomcat
- Escribir la siguiente dirección en la barra de direcciones del navegador Web:
  - http://localhost:8080/JSON/JSON-JSPExample.jsp



http://www.roseindia.net/tutorials/json/json-servlet-example.shtml

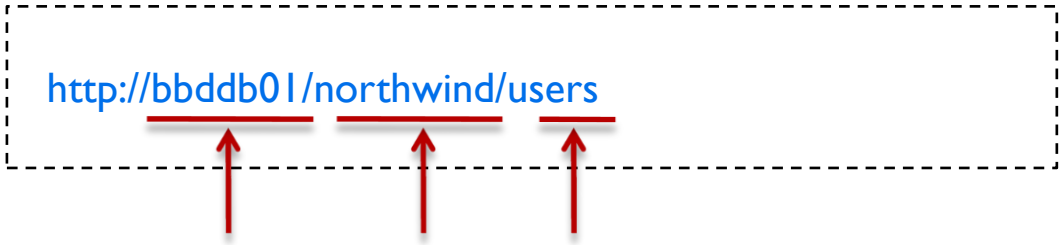
REST

- REST, acrónimo de
  Representational State Transfer
- Es un estilo de organizar la arquitectura software para sistemas distribuidos
  - Aparece como parte de la tesis doctoral de Roy Fielding en el año 2000
  - Se basa en el uso de URL para localizar los recursos y en HTTP para la gestión de los recursos

http://es.wikipedia.org/wiki/JSON

Ejemplo simple con REST (1/3)

- Tenemos como recurso una tabla (users) de una base de datos (northwind) que reside en un servidor de base de datos (bbddb01).
- Localización del recurso:



- El contenido de los datos puede ser XML o JSON
  - En SOAP es obligatorio XML

Ejemplo simple con REST (2/3)

- Tenemos como recurso una tabla (users) de una base de datos (northwind) que reside en un servidor de base de datos (bbddb01).
- Gestión del recurso:

REST	CRUD (Create, Read, Update, Delete)
POST	Crear
GET	Leer
PUT	Actualizar o crear
DELETE	Borrar

- Se usan las acciones de HTTP
  - En SOAP el programador se crea métodos extras

http://www.slideshare.net/rmaclean/json-and-rest

Ejemplo simple con REST (3/3)

- Ejemplos de gestión del recurso:
  - http://bbddb01/northwind/users[firstname="rob%"]

POST	Error
GET	Obtenemos todos los usuarios que comienzan por rob
PUT	Error
DELETE	Borramos todos los usuarios cuyo nombre comience por rob

  - http://bbddb01/northwind/users + datos de entrada

POST	Crear un nuevo usuario
GET	Obtenemos los usuarios que cumplen el criterio
PUT	Creamos/Actualizamos un usuario
DELETE	Borramos todos los usuarios que cumplan el criterio

http://www.slideshare.net/rmaclean/json-and-rest

REST vs. SOAP

- Las acciones de HTTP pueden sustituir la creación de métodos HTTP (SOAP) a la hora de gestionar los datos:
  - Ahorro a la hora de codificar métodos por cada tipo de dato
  - Forma más familiar de tratar con datos:

```
recurso = new Recurso('http://bbddb01/northwind/users') ;
recurso.delete() ;
```

  - Se puede usar tanto JSON como XML
  - Se puede usar fácilmente desde *JavaScript*, Java, etc.

http://blog.feedly.com/2009/05/06/best-practices-for-building-json-rest-web-services/

Ejemplo en ASP.NET Ajax

- Llamada a un servicio Web remoto desde un cliente con JavaScript
- Uso del servicio Web *findNearByWeatherJSON* de *Geonames* que toma la longitud y latitud como parámetros y nos retorna un objeto JSON que describe el tiempo (situación meteorológica) obtenido del observatorio más cercano

http://geekswithblogs.net/JuanDoNeblo/archive/2007/10/24/json_in_aspnetajax_part2.aspx

WeatherService.cs

```
using System;
using System.Web.Script.Services;
using System.Web.Services;

namespace Mashup
{
    /* To allow this Web Service to be called from script using ASP.NET AJAX,
     * we need to set the following attribute. */
    [ScriptService]
    [WebService(Namespace = "Mashup")]
    [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
    public class WeatherService : WebService
    {
        [WebMethod]
        public string GetWeatherByLocation (double lat, double lng)
        {
            return GeonamesWeather.GetWeatherByLocation(lat, lng);
        }
    }
}
```


http://geekswithblogs.net/JuanDoNeblo/archive/2007/10/24/json_in_aspnetajax_part2.aspx

GeonamesWeather.cs (1/2)

```
using System;
using System.Net;
using System.Globalization;
using System.IO;

namespace Mashup {
   public class GeonamesWeather {

      private readonly static string
                  FindNearbyWeatherUrl = "http://ws.geonames.org/findNearByWeatherJSON?lat={0}&lng={1}";

      public static string GetWeatherByLocation(double lat, double lng)
      {
         string formattedUri = String.Format(CultureInfo.InvariantCulture, FindNearbyWeatherUrl, lat, lng);

         HttpWebRequest webRequest = GetWebRequest(formattedUri);
         HttpWebResponse response = (HttpWebResponse)webRequest.GetResponse();
         string jsonResponse = string.Empty;
         using (StreamReader sr = new StreamReader(response.GetResponseStream()))   {
             jsonResponse = sr.ReadToEnd();
         }
         return jsonResponse;
      }
```


http://geekswithblogs.net/JuanDoNeblo/archive/2007/10/24/json_in_aspnetajax_part2.aspx

GeonamesWeather.cs (2/2)

```
        private static HttpWebRequest GetWebRequest(string formattedUri)
        {
            // Create the request's URI.
            Uri serviceUri = new Uri(formattedUri, UriKind.Absolute);

            // Return the HttpWebRequest.
            return (HttpWebRequest)System.Net.WebRequest.Create(serviceUri);
        }

    }
}
```


http://geekswithblogs.net/JuanDoNeblo/archive/2007/10/24/json_in_aspnetajax_part2.aspx

WeatherService.asmx

```
<%@ WebService
        Language="C#"
        CodeBehind="WeatherService.cs"
        Class="Mashup.WeatherService"
%>
```


http://geekswithblogs.net/JuanDoNeblo/archive/2007/10/24/json_in_aspnetajax_part2.aspx

WeatherService.asmx

```
<%@ WebService
        Language="C#"
        CodeBehind="WeatherService.cs"
        Class="Mashup.WeatherService"
%>
```


http://geekswithblogs.net/JuanDoNeblo/archive/2007/10/24/json_in_aspnetajax_part2.aspx

JsonWeather.aspx (1/2)

```
<body>
<form id="form1" runat="server">
<div>
    <asp:ScriptManager ID="ScriptManager1" runat="server">
        <Services>
            <asp:ServiceReference Path="~/WeatherService.asmx" />
        </Services>
    </asp:ScriptManager>
    <div id="legend">
        <b>If you don't have a clue</b>:
        <br />
        <b>Rome</b>: Lat: 41.9 - Lng: 12.5                <br />
        <b>London</b>: Lat: 51.5 - Lng: 0                 <br />
        <b>Paris</b>: Lat: 48.84 - Lng: 2.35              <br />
        <b>Redmond</b>: Lat: 47.68 - Lng: -122.13    <br />
        <b>New York</b>: Lat: 40.7 - Lng: -74
    </div>

    <label for="txtLat“>            Latitude:</label><input type="text" id="txtLat" />
    <label for="txtLng“>           Longitude:</label><input type="text" id="txtLng" />
    <input type="button" value="Get Weather Info" onclick="callService();" />
    <div id="divResult“></div>
```


http://geekswithblogs.net/JuanDoNeblo/archive/2007/10/24/json_in_aspnetajax_part2.aspx

JsonWeather.aspx (2/2)

```
<script type="text/javascript">
        function callService()  {
            var latitude = $get('txtLat').value;
            var longitude = $get('txtLng').value;
            Mashup.WeatherService.GetWeatherByLocation(latitude, longitude,GetWeather_success, onFailed);
        }
        function GetWeather_success(e)   {
            var result = Sys.Serialization.JavaScriptSerializer.deserialize(e, true);  //var result = eval('(' + e + ')');
            var weatherData = new Sys.StringBuilder();
            var line;
            for(var property in result.weatherObservation)    {
                line = String.format("<b>{0}:</b> {1}<br/>", property, result.weatherObservation[property]);
                weatherData.append(line);
            }
            $get('divResult').innerHTML = weatherData.toString();
        }
        function onFailed()      {
            $get('divResult').innerHTML = 'Something went terribly wrong!';
        }
    </script>
  </div>
  </form>
</body>
```


http://geekswithblogs.net/JuanDoNeblo/archive/2007/10/24/json_in_aspnetajax_part2.aspx

Contenidos

1. Plataformas:
   1. Supercomputadores
   2. Clusters
   3. Grid
   4. *Cloud computing*
2. **Entornos de desarrollo**
   1. JSON+REST vs XML+SOAP
   2. **Peer-to-peer y redes sociales**

Tendencias…

- Consolidación y mejora del paradigma peer-to-peer.
  - Proyecto JXTA
    - http://www.jxta.org
- Computación colaborativa.
  - Entorno JSDT
    - http://www.java.sun.com

Tendencias…

- Aplicaciones para entornos de computación colaborativa: redes sociales.
  - Ejemplo: Facebook
    - http://junal.wordpress.com/2008/01/20/a-sample-facebook-application-with-codeigniter/

- Aplicaciones para entornos de computación colaborativa: pizarras compartidas.
  - Ejemplo: Google Wave
    - http://radleb.net/blog/2009/08/26/crear-un-robot-en-python-para-google-wave-i/

Desarrollo de aplicaciones distribuidas: Tendencias

Grupo ARCOS

Desarrollo de Aplicaciones Distribuidas

Ingeniería Informática

Universidad Carlos III de Madrid