

# Scaling Mission Critical Applications

Using Internet based advanced distributed techniques

By **Joe Ellsworth** - [joe@pybiz.com](mailto:joe@pybiz.com) (408) 364-1741



CEO & Lead Architect  
PyBiz, Inc.  
(Silicon Valley)

## Our Focus

Applying advanced Internet techniques that make it easier to cope with rapid change.

## Our Mission

Enable rapidly evolving & collaboration centric businesses to manage the impact of diversity and business changes on IT infrastructure.

## Distributed Internet Techniques

- **Open / Cross Platform Techniques**
  - http, XML, XSL, etc
- **Scale at the architecture Level**
- **Use Auction Techniques to optimize.**
- **Design to scale Horizontally but allow vertical**

**Scaling Mission Critical Applications**

**Using Auction Techniques**



# Keywords & Audience

---

## ■ Watch Words

- E-Business,
- Scalability,
- Mission Critical, Enterprise Class,
- E-Commerce,
- B2B,
- Auction,
- Provisioning,
- XML,
- Internet

## ■ Who Should Attend

- IT Managers
- Directors of IT
- Web Development managers
- Senior Architects
- Technologists responsible for delivering & maintaining next generation mission critical e-business

**This course is semi - Technical. So you do not have to have an in-depth technical background**



# Class Protocol

---

- The success of this class depends on your interaction.
  - Please Ask questions as we GO.
  - I might defer an answer if there is a place where the answer will make more sense.
- My goal is to help you understand where these techniques apply to your business.
  - Class FAQ – I Need a volunteer to capture the class questions and I will send out detailed answers as part of the class materials.



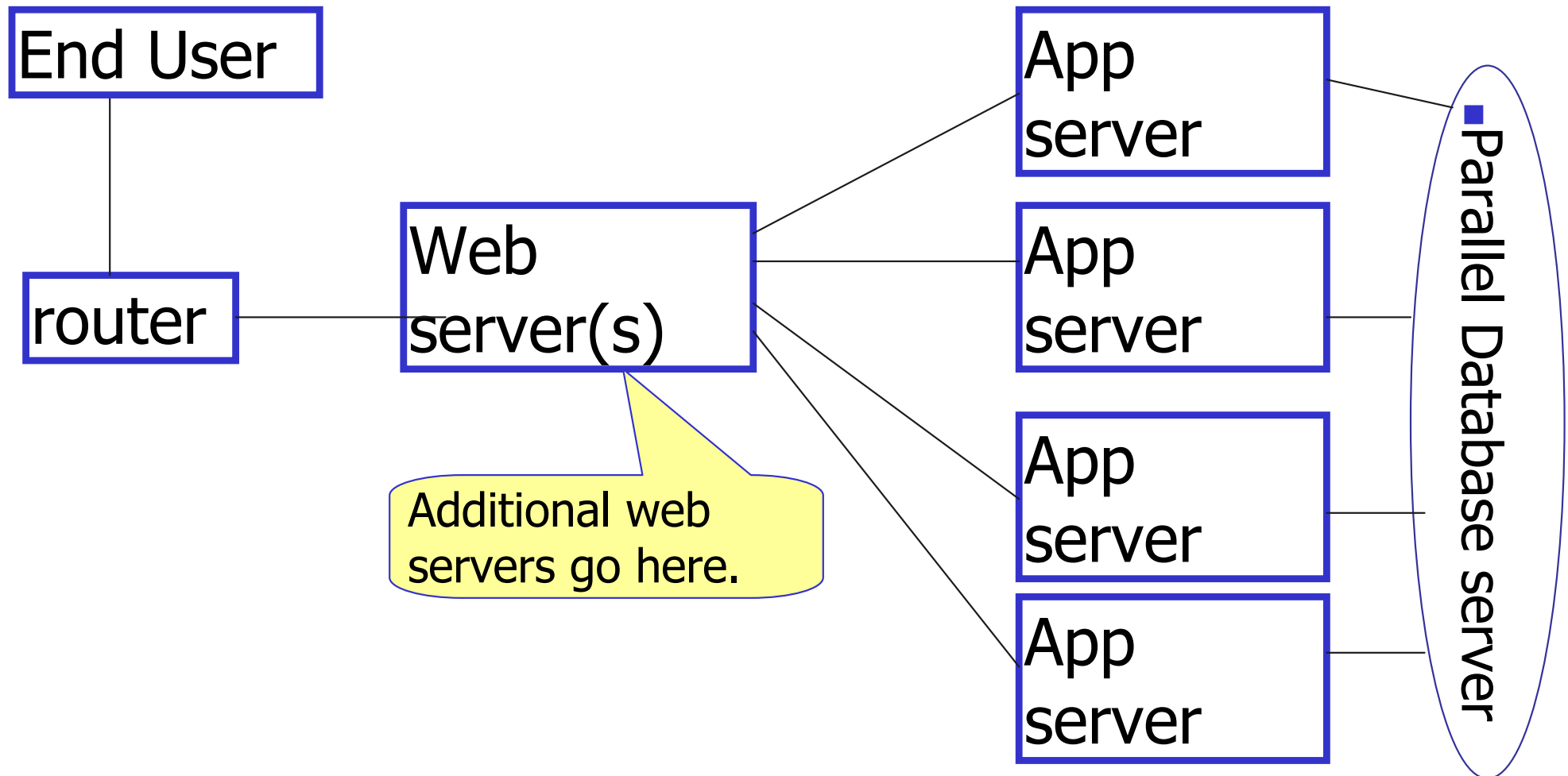
# About this seminar

---

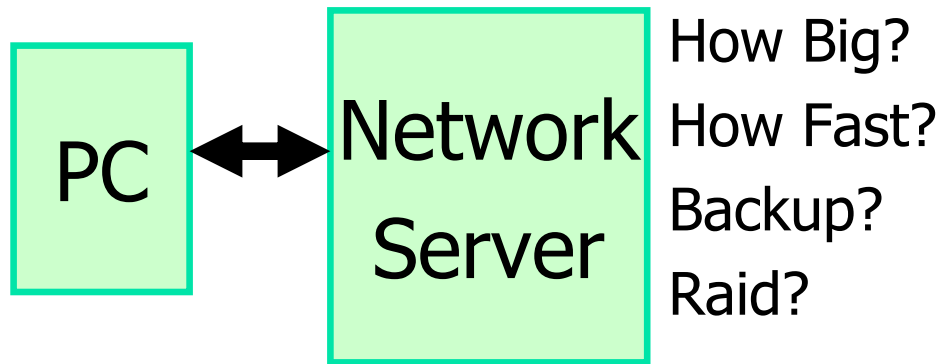
- Share our experience gained while building and deploying mission critical experience.
- This seminar is about demystifying the technology not our products.
- I will share lessons learned during 20 years of experience building and delivering software products.
- Best practices from the last 6 years have been delivering extranet corporate B2B and B2C solutions.

# An Example Mission Critical System

- Assuming the heavy domain processing is done on app servers.



# Picture of the World Today



## ■ Shared Storage Scenario


- Corporate IT has to cover heavy managements costs.
- Once a IT group takes responsibility they tend to drive up costs.
- Drives the cycle between local systems, mini and mainframe computers.

## ■ Today:

- Customers / users awarded by management decree
- Each time new services are requested internal IT finds way to increase cost.
- Management is forced to impose draconian cost control techniques.
- No reasonable way to allocate costs so we get to traditional over charges.

## ■ Desired States:

- New services added as needed by end users.
- Use natural competition to control costs.
- Move or Lay off those that do not compete effectively.



# My History

Why I am qualified to comment

- XDisect & XDisect Forms
  - E-Speak Broker
  - Dynamic Security Cells
  - ESN – Enterprise Class Extranet Portal
  - QOA – Web Configuration.
  - Watson Server
  - RPG to C++ translator.
  - Distributed workman's compensation System.
  - Municipal Billing Systems
  - Books Store POS
  - 3d robotic control
- History starting in 1982 software Development.
  - 5 years self employed building fixed price solutions.  
6 years as a preiminant Object oriented Architect Consultant.
  - 6 Years as manager and thought leader at H.P.
  - CEO – PyBiz, Incorporated.



# What is Scaling

- Production deployment
  - How do we respond to requests fast enough.
  - How do we provide services originally unanticipated.
  - How do we deal with semi stable back end servers.
  - How can we start small and grow incrementally.

- Scaling Development teams
  - **Small empowered groups deliver faster and cheaper.**
  - How do we tie them together.
  - How do we present a single face to the customer.
  - How do we enable for new groups that will solve new problems.
- Scaling Dollars
  - How do we enable maximum use of existing hardware without limiting deployment.
  - How do we use natural competition to control costs.





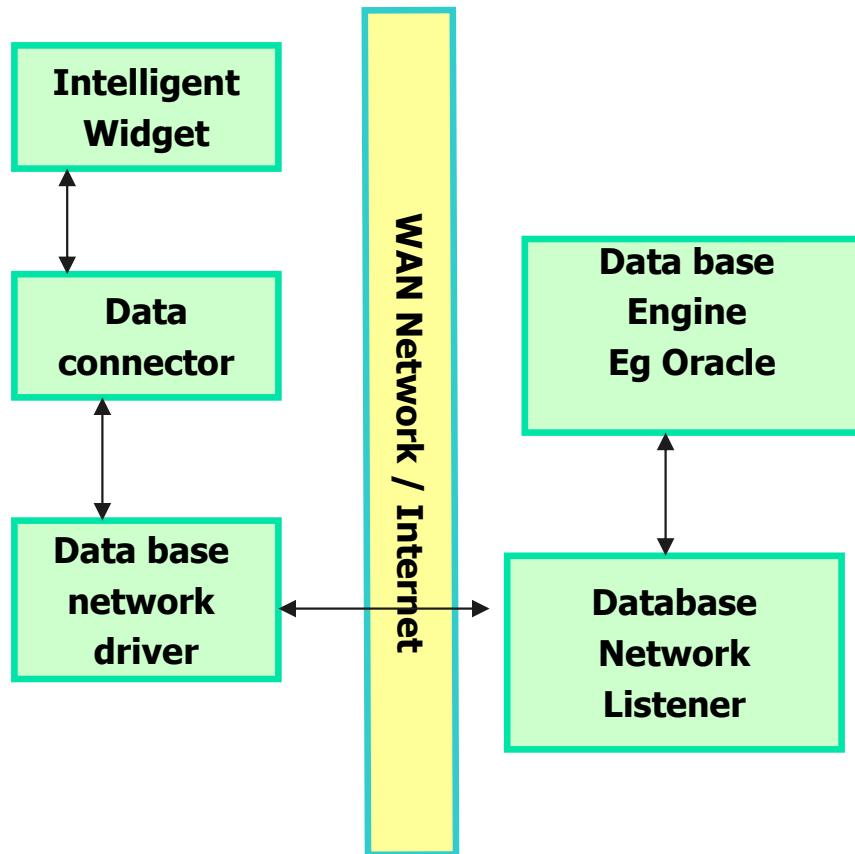
# Our Goal(s)

---

- Supporting partner focused integration activities.
- Providing a browser more versatile access to existing business data.
- Web enabling systems not originally designed to support web expectations.
- Supporting the new loads without massively upgrading existing legacy systems.
- Ensure high availability at all components.
- Ensure consistent loading across components.
- Make sure high impact requests are not allowed to substantially impact the rest of the system.

# Bandwidth & Latency Problem

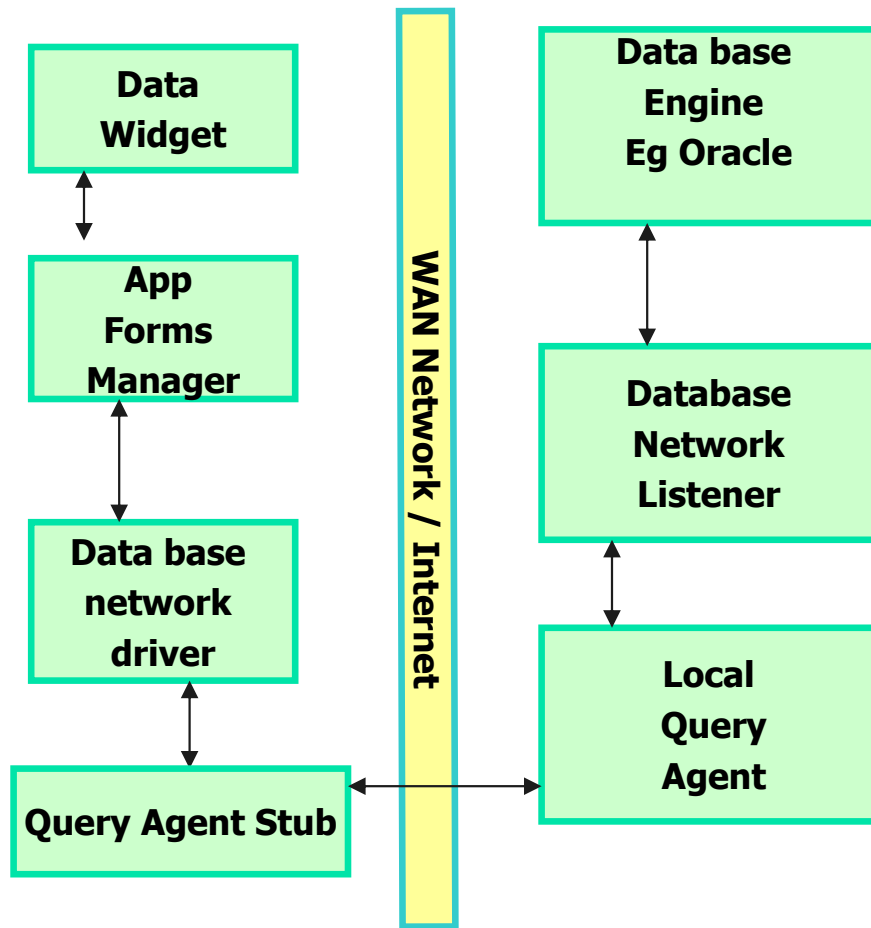
The intelligent Widget Fiasco for intermediate VBA and Java Widget programmers. The mistake for every generation.



- #1 problem I find when called in to diagnose systems.
- As many as 500 round trips on the network paint one screen.
- Comes back for every generation.
- It requires thinking but not hard to fix if done up front.

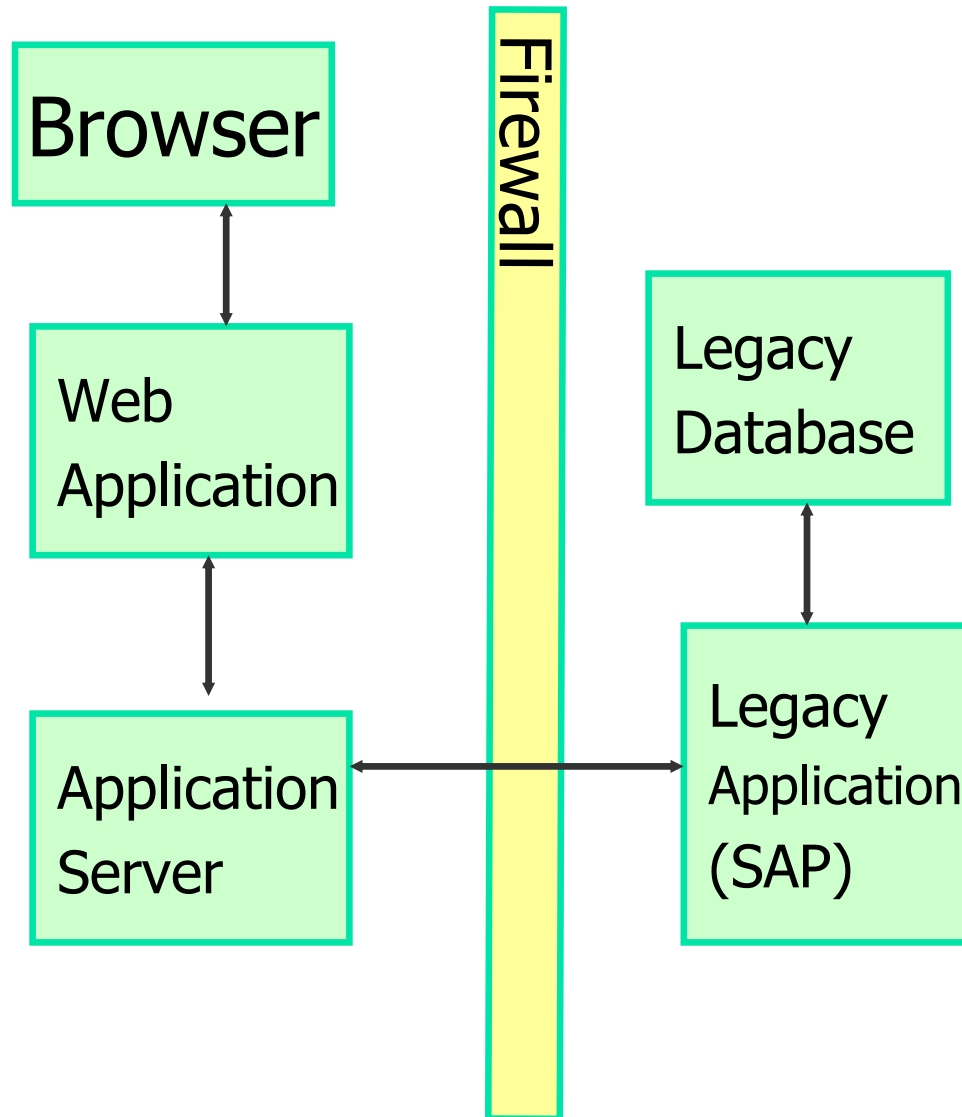
# Bandwidth & Latency Done Right

Using Query agents close to data source alleviates the problem.



- Works in most instances.
- Requires 1 to 3 round trips on average rather than 500.
- Actually lowers load on the legacy database.
- Not as easy to understand for programmers.
- No automated tools to support.

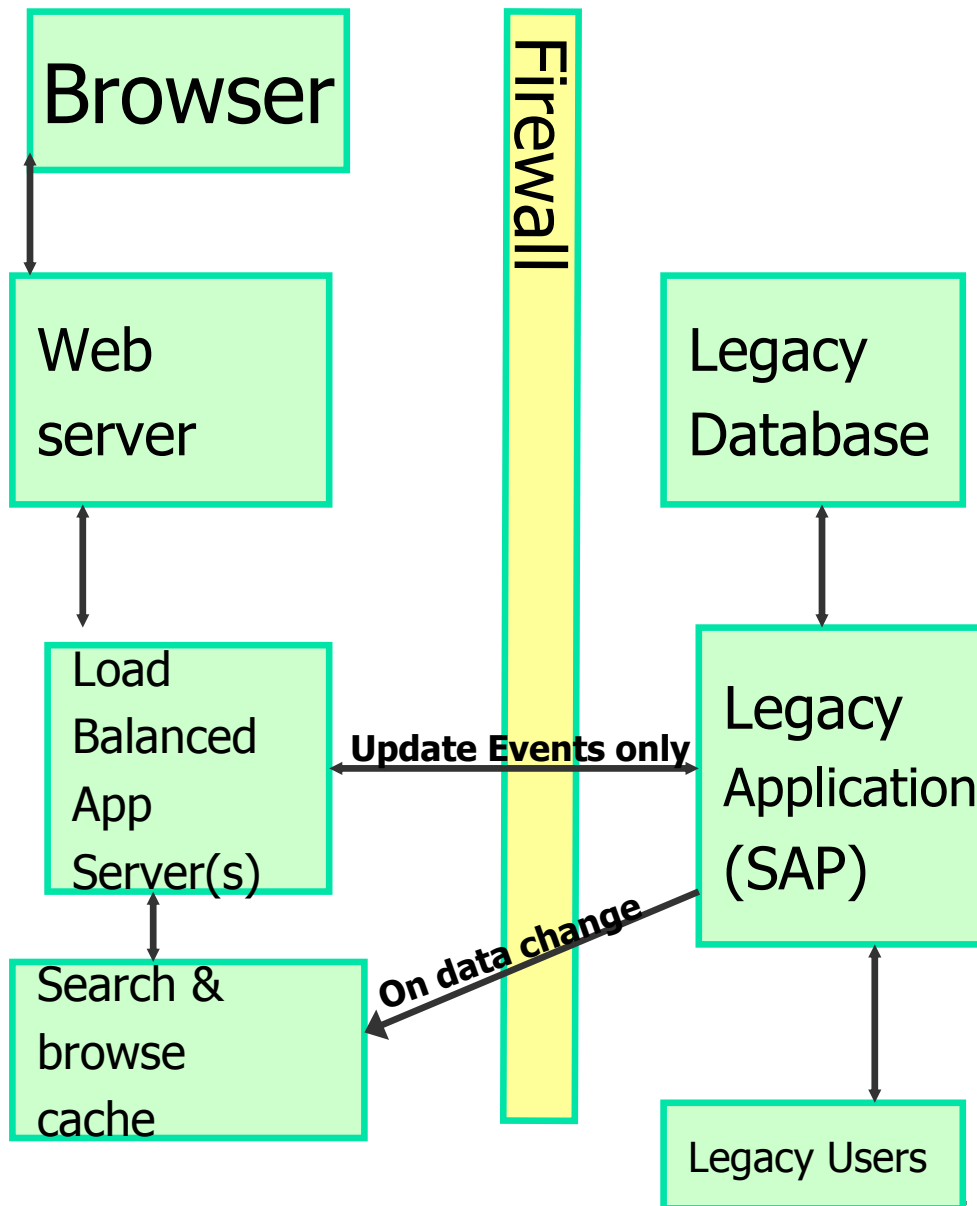
# Web Front Ends For Legacy Systems



- Problems With the Obvious
  - Existing schema can not support Web user expectations.
  - Existing systems can not scale to meet additional user loads.
  - Schema re-design in place will kill main line performance.

# Bringing them to the Web

[www.pybiz.com/products/xdisect/scenarios/xd\\_sap\\_scenarios.htm](http://www.pybiz.com/products/xdisect/scenarios/xd_sap_scenarios.htm)



- Horizontal scaling of app server makes meeting users needs feasible.
- Only those operations that require a change to the legacy database actually require legacy loads.
- XML Search Cache provides much stronger search semantics to make web user happy.
- Prevents having to update Legacy SAP instance which is Big Bucks and big hassle.

Scanning Mission Critical Applications  
Using Auction Techniques



# Current Scaling Strategies

---

- Buy a Big Box such as a HP V-Class and run the application on it.
- Buy a bunch of smaller web servers to serve the web and then a big app server and a big database server box.
- Use a 3 tier application engine such as Bluestone or Web Logic and buy a Big Database server.
- Buy a few smaller app servers and forward cache the bulk of the work load to Search & browse caches.



# Common Mistakes

- Monolithic thought processes. I am the center of the world and will have all my own processes such as registration.
  - Buying monolithic applications and then trying to force them to work as a key node in a much broader worldview.
  - Huge project teams and long timelines.
- Single point of failure.
  - Lots of network round trips.
  - Closed / proprietary systems that almost meet business needs.
  - Not anticipating change.
  - Not designing to accommodate unanticipated change.
  - Taking inappropriate shortcuts like direct connect data widgets.



# Architect & design mission critical applications

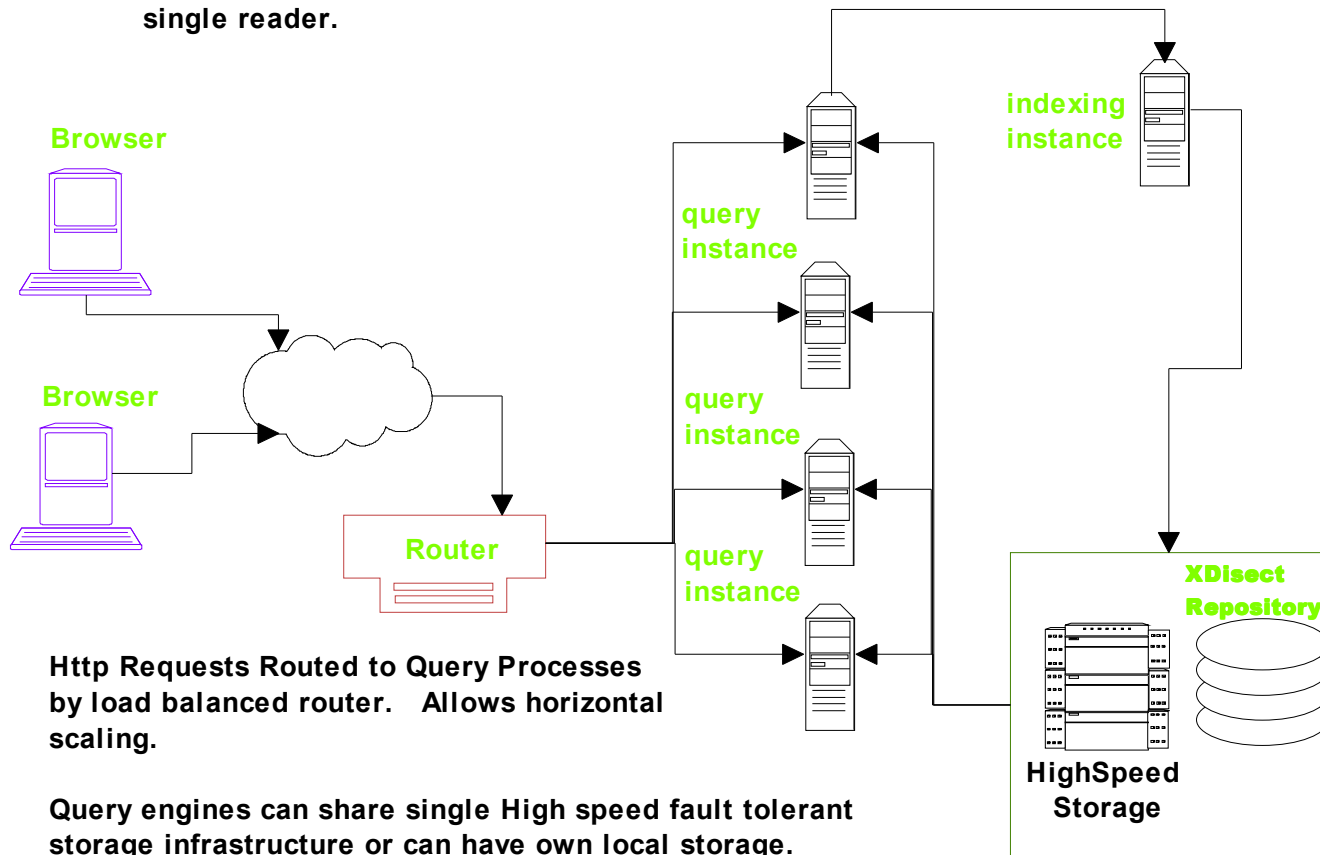
---

- Use the highest level communication semantics possible  
See.: Business XML
- Remain technology agnostic.
- Design to accommodate changing business requirements
- Build a bunch of small systems.
- Design to tie the small systems together into a larger whole.
- Use a number of smaller teams with short deliverables.
- Force teams to fully document interfaces between systems.



# Load Balancing

Any query instance can receive update commands but they are all processed by single reader.



Http Requests Routed to Query Processes by load balanced router. Allows horizontal scaling.

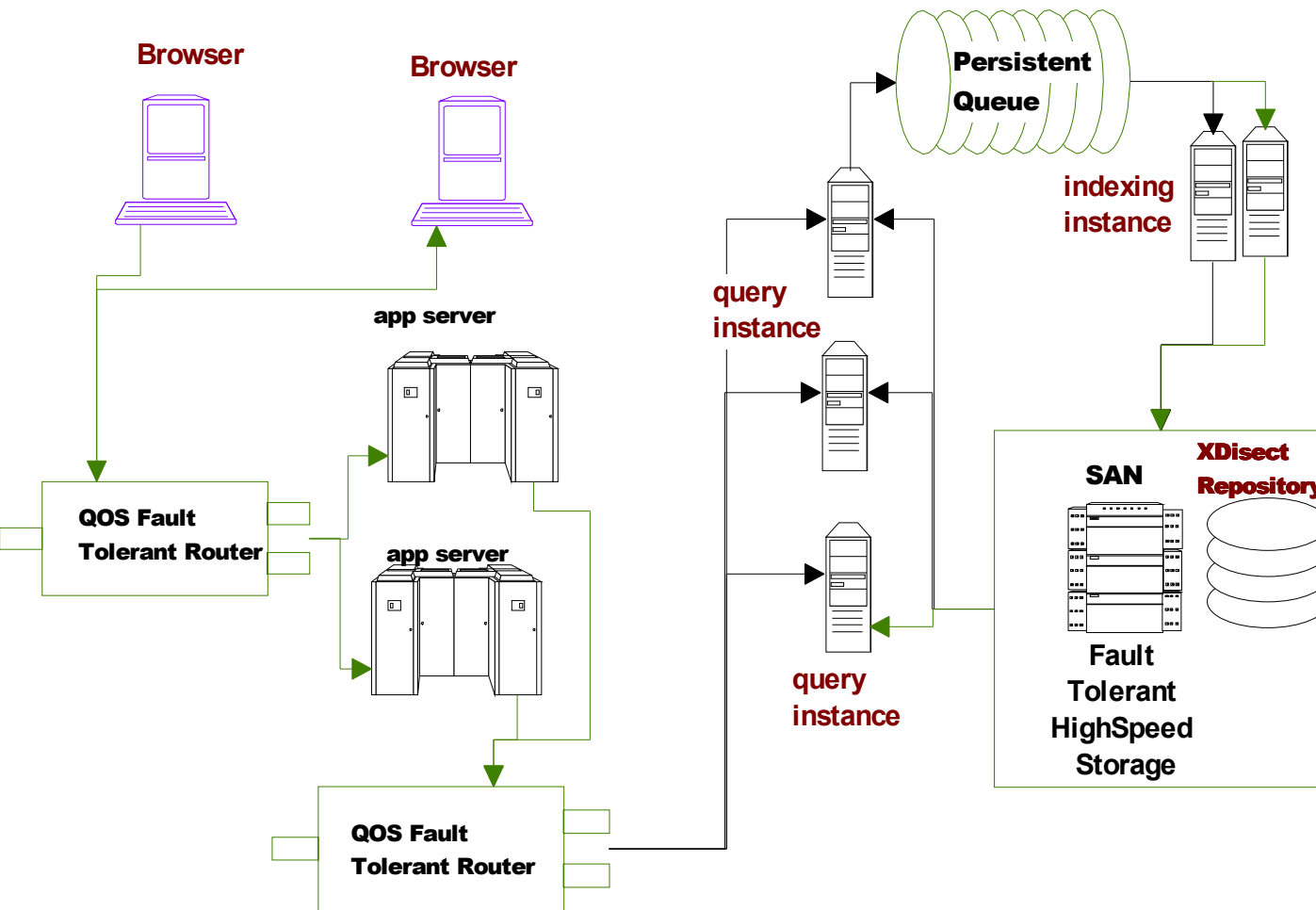
Query engines can share single High speed fault tolerant storage infrastructure or can have own local storage.

Many reader single writer design makes query scaling easy.

Support for synchronizing multiple data stores when query machines are using local storage.

## Scaling Mission Critical Applications Using Auction Techniques

# Fault Tolerance



- Any query instance can receive requests for insert/update
- Requests are stored in a persistently queue.
- No single point of failure in the architecture.
- Total failure would require failure of all components
- Second router functionality can be done using software but we prefer the h/w router for performance.
- Only one instance of the XDisect Indexer is active at a time.



# High Level Communication Semantics

Always use the highest level semantics reasonable for the task at hand. (Very Common mistake in Junior and Intermediate XML programmers).

## Business XML

```
<person>
  <name>
    <first>joe</first>
    <last>ellsworth</last>
  </name>
  <member>
    <since>2001-10-30</since>
    <expires>2002-03-15</expires>
  </member>
</person>
```

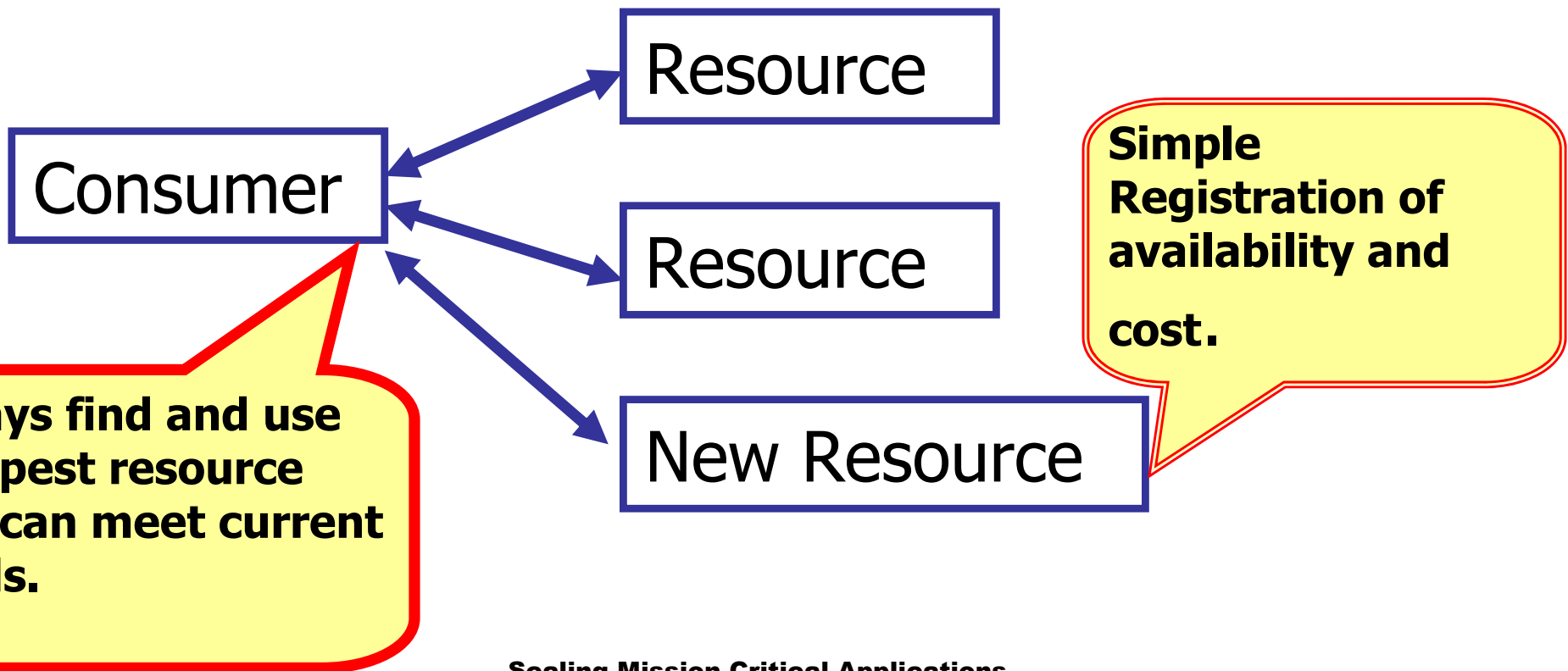
## Programmers XML

```
<record name="person" type="single">
  <element type="hash-container" name="name">
    <field type="string" length="40"
      name="first">joe</field>
    <field type="string"
      length = "40"   name = "last">ellsworth
    </field>
  </element>
  <element type="hash-container" name="member">
    <field type="date" storage="julian" size="4"
      format="ccyy-mm-dd"
      name="since">2001-10-30</field>
    </field>
    .....
  </element>
</record>
```

# Why Auction Techniques

## Automatic Discovery

- Transparently add new processing resources without making configuration changes to the consumers.





# Better Average Response

## At a lower cost

### ■ Better Allocation of resources

- More important / critical applications get first priority.
- Degrade less important applications first.
- Use more of the total machine resources available.
- Better balancing of space CPU cycles.

Pricing  
Server =  
2

Order  
Placement =  
3

Catalog  
Query =  
1.5

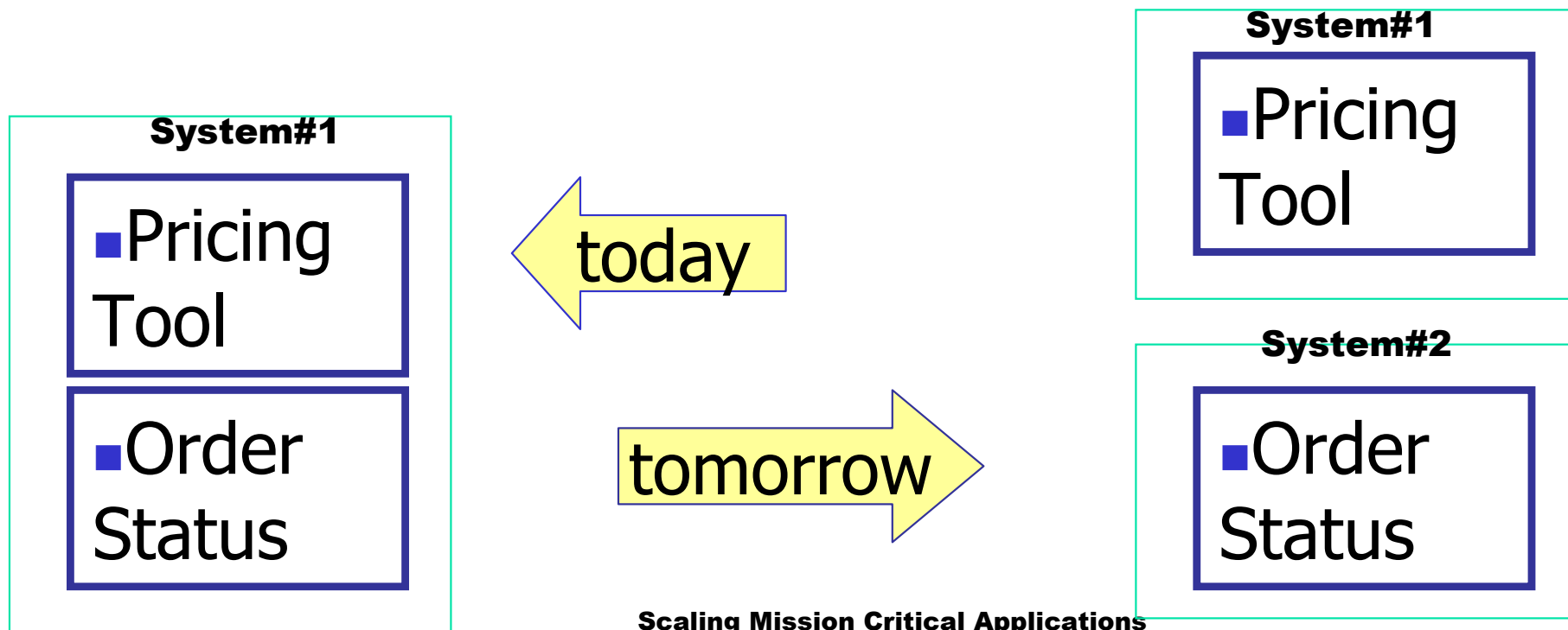
FAQ &  
Support =1

**Bid Rate For Service =**

**Current Average System Load \* Priority**

# Changing Configurations

- As services are moved around we can simply move the service and change the offer.
- We can make some machines more heavily allocated to some types of tasks by making their offers for a given service lower.



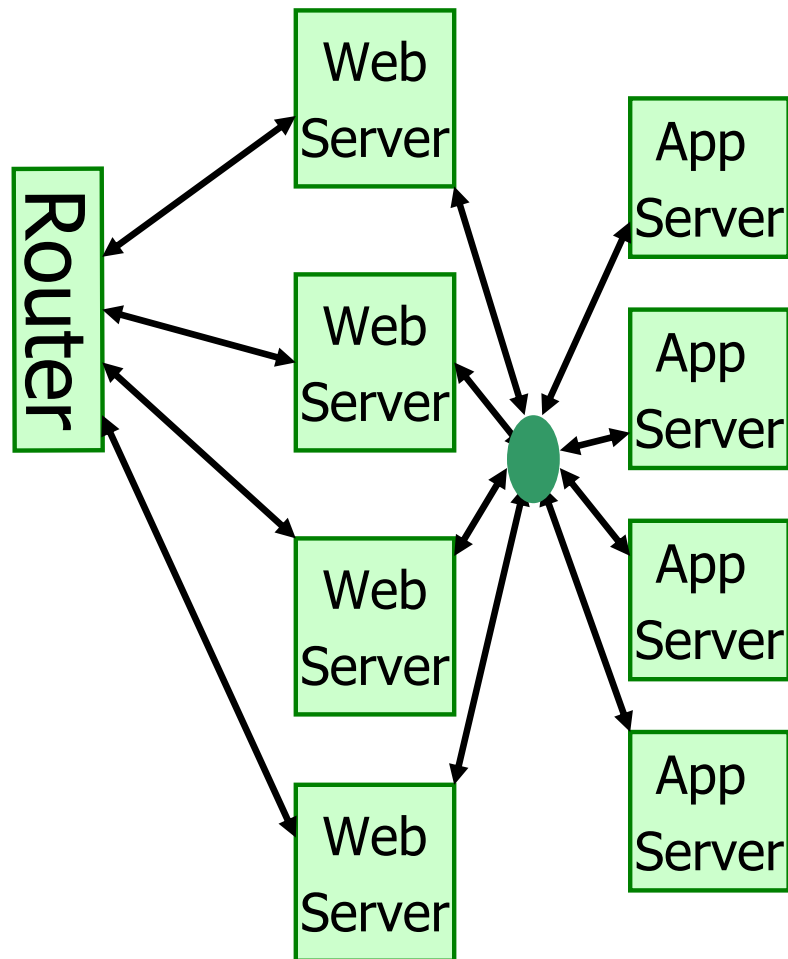
**Scaling Mission Critical Applications  
Using Auction Techniques**



# Overview of Auction Techniques

- **Normal Auction** – Seller makes available for purchase Buyers place a bid
  - **Reverse Auction** – Buyers indicate what they want to buy and the sellers bid on what they are willing to sell it for.
  - **Brokered Auction** – A middle man provides the services of receiving the initial request and matching it with offers.
- **Predicted Auction** – is similar to a reverse auction however a set of items that are likely to be requested by the buyer are advertised with the broker in regular bids. That way when the buyer decides they want to buy something they can look at the set of available bids.  
If the seller quits sending bids then his existing set will eventually time out and the they will not be considered.

# Applying Techniques to Enterprise Scaling



- When the web server receives a request it sends out a request to each of the app servers to see how much the app server will charge for that request and then it sends the request on to the app server that responds with the best bid.
- The app servers charge more as they get busy.
- The app server which recently dealt with a request for that user session will give a discount for another request from the same user.
- New app servers can join the family just by advertising their presence.
- In some instances the app servers will need to share state to allow one to take over for the other in other instances this can be transparent.
- This means the web server needs to be smart enough to auction it's service calls. Or use a broker to do it.
- For common or relatively cheap calls then a predicted auction strategy may work best.
- The auction adds one extra round trip but in many cases the auction request can be sent out via UDP and or the predicted auction can be used to predict the auction cost.





# Auction Features & Benefits

---

- Natural Load Balancing.
- Natural Fault Recovery.
- Graceful degradation.
- Easy to add new providers. (horizontal scaling).
- Optimal use of a wider range of systems.
- Easily take app servers off line for service & upgrade
  - Most of the auction behavior can be implemented with small amounts of code over open interfaces.
- Uses natural techniques to find the server resource best able to service current requests.
- Allows internal competition which can drive cost cutting
- Provides concrete book keeping and expense allocation.



# Problems with Auctions

---

- Careful design is needed or the auction handling overhead can be quite high.
- Contrary to hardware and OS vendors so it may be a while before it catches on.
- Still requires local server software than knows how to fulfill the service.
- Requires current data access across the set of possible service agents.
- Existing EJB interfaces do not support.
- Ready made clients do not know how to support it.
- The UDP and HTTP specs do not have a published standard.



# Auction Messages

---

- Offer / Seller Registration
- Bid Request
- Offer Termination
- Service Bid



# Registration Bid

---

- Sample Registration

```
<offer>
```

```
  <id>99188181</id>
```

```
  <url>http://127.0.0.1:8060/get_user.xsql</url>
```

```
  <terms>
```

```
    <charge>10.998</charge>
```

```
    <per>call</per>
```

```
    <expires>9000</expires>
```

```
    <active_add>5.2</active_add>
```

```
  </terms>
```

```
</offer>
```



# Explanation of an Offer

---

- **Offer.id** = refer to this when you send if service requests. Any new offers with the same ID replace the old one.
- **Offer.url** = is the URL where the service is actually available to be called. If not http: then could be a different location. The documentation for parameters to this call is handled separately see.: uddi, WSDL
- **Offer.terms.charge** = How much is service going to charge per unit delivered.
- **Offer.terms.per** = is the unit of charge. Call is each time service is called could also have record, page, etc.
- **Offer.term.expires** = the number of seconds to expiration. Once expired no new service requests will be sent.
- **Offer.term.active\_add** = how much cost to add per concurrent active request from this service.



# Delivery Of Bid

---

## **Delivery Bid Channels**

- HTTP Post / Call
- UDP broadcast
- Unicast Groups
- TIBCO – Messaging Events.
- Active Message Broker
- HP E-Speak Events Model.
- EJB - JMI



# When to Register / Advertise

---

- Delivery Events
  - On Start up Register.
  - Rebroadcast as per expiration policy.
  - Send update when availability of machine changes
  - If the service is restarted.
  - When the load of the machine causes a change of cost.



# Receiving an Offer

---

1. Inbound offer received via deliver agent.
2. Normalize the price. Normally force this to be sent normalized as per prior agreement E.G. Everybody sells at price per page.
3. Create internal key of zero padded Key concatenated from price + id.
4. Sort the keys
5. The best bid is the one with the lowest key.





# Choosing a Supplier

---

- 1. Find the lowest current cost.**
- 2. Increment the concurrent active request penalty to the current bid for that offer.**
- 3. Request the service.**
- 4. Get the results**
- 5. Log the usage**
- 6. decrement the concurrent active request penalty to the current bid for the offer.**

If another request needs to be fulfilled do not forget to resort the offer price.



# Case Studies

---

- ⑩ Two case studies of auction based scalable next generation mission critical applications
  - ⑩ Forms manager Service Portal
  - ⑩ Student Driven Sample



# Case Study Components

---

- Scenario Definition
  1. Defining the basic requirements
  2. Defining the Business Architecture
  3. Defining the Technical Architecture
    1. Basic user or screen flow
    2. Defining the Hardware Architecture
    3. Defining the Software Architecture
  4. Scalability Considerations



# Case Study Desired Results

---

1. Specific points used to maximize fault survivability.
2. Specific points used that maximize load carrying capacity.
3. Specific points used to maximize capability for future growth. (in particular growth with incremental expansion).
4. Specific points used to minimize cost which use or emphasize steps #1-#3.



## Case Study 1 – *Enterprise Forms Portal*

- Our Portal owner has 25 separate web sites.
  - Each site is responsible for registering the end user.
  - Each site is built by separate teams using different technology.
  - The sites and teams generally do not know about each other.

- [http://www.xdfind.com/read\\_me/business\\_overview.html](http://www.xdfind.com/read_me/business_overview.html) - Business overview for Secure Portal.
- [http://www.pybiz.com/products/cef/scenarios/cef\\_nm\\_scenarios.pdf](http://www.pybiz.com/products/cef/scenarios/cef_nm_scenarios.pdf) - community portals and open integration protocols.
- [http://www.xdfind.com/read\\_me/programmer\\_overview.html](http://www.xdfind.com/read_me/programmer_overview.html) – Sample data structures and queries for such a portal.
- <http://www.pybiz.com/products/xdisect/market-thinking.html> – Other types of related portals.

### Some Integrated Sites

- Sales Information Request
- Service Contract Management
- Service request / dispatch system
- FAQ – Knowledgebase
- Customer Care & one to one marketing.
- Demonstration software download
- Demonstrate CD ordering
- Customer negotiated Pricing Tool
- Order Status



## *Enterprise Portal - **Business Problem.***

- Each site is responsible for registering the end user.
- If the office phone changes they have to go to every web site and change it.
- The user has to log in at each site.
- The user may have a different user ID and password for each site.
- The user has to talk to different support org for each site.
- We have different conflicting data for each user across site and do not know which ones are accurate.
- It is nearly impossible to enforce the data privacy rules since we have so many data owners.
- No Consistency in presentation of forms that gather the data.
- The user gets confused because another site for same company has already gathered the same data so why are we asking for it again.

## **Move duplicated responsibility common portal**

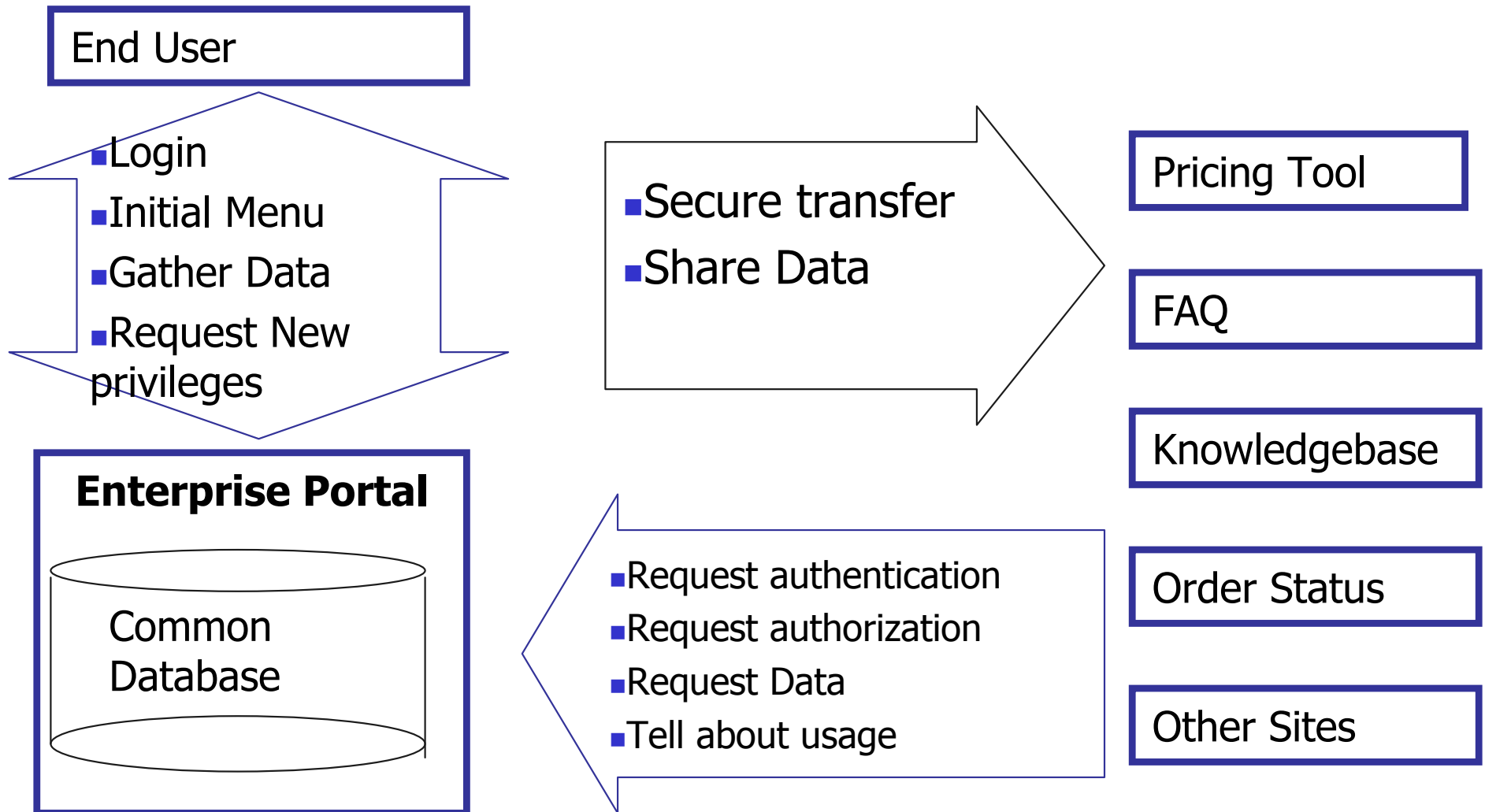
### ■ Central Portal Responsibilities

- Authenticate all users.
- Transfer uses to individual sites as needed.
- Provide security / authorization for all sites.
- Present forms and gather any data needed by sites.
- Share data needed by the remote sites.
- Store & Manage data over time.
- Make sure no single piece of data is requested more than once.
- Present a personal screen showing the users what they can access.

### ■ Individual Sites Responsibilities

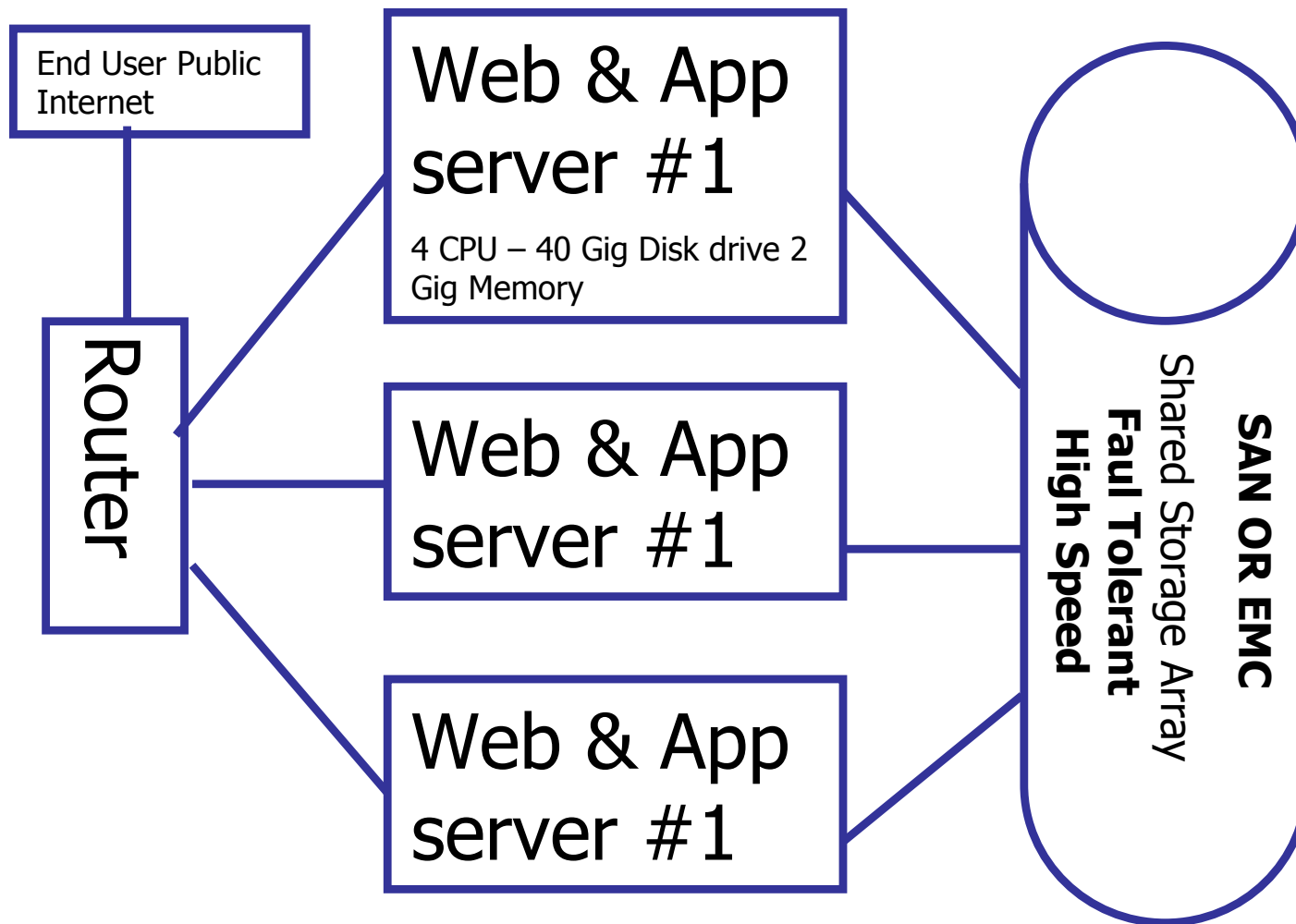
- Each site retains responsibility for it primary function.
- Each site will require a minimum of changes to participate.

# Enterprise Portal – Business Architecture





# Enterprise Hardware Architecture



- Cost Constraints limited hardware so we ended up without a second tier of hardware but we still had to support 25,000 users peak at an average of 3 pages a minute.

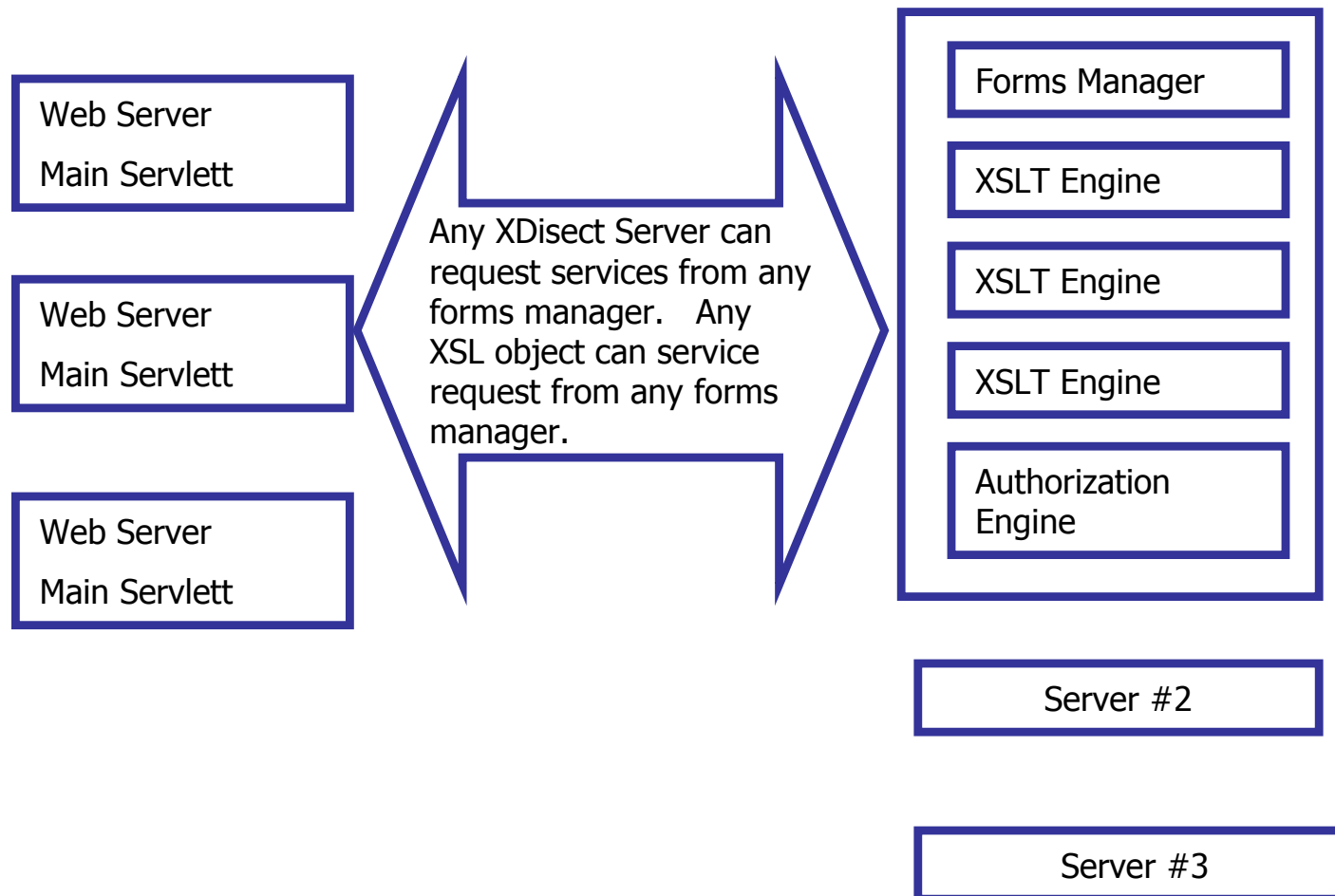


# *Enterprise Portal Scaling Questions*

---

- Since business conditions limited the solution to a total of three servers how do we best spread the work load across these servers while retaining an adequate fault recovery capability.
- Since XSLT can be slow under some conditions how do we cope design to cope with the fact that we may need to scale up our XSLT processing capacity.

# Enterprise Portal Software Architecture



**Scaling Mission Critical Applications  
Using Auction Techniques**



# *Enterprise Portal Using Auction Techniques.*

- Some repository queries for reports can consume lots of resources. The system load based cost functionality allows us to better route the requests around the busy instance.
  - By dramatically lowering the cost for expensive reporting functions on the server that is already processing a expensive report we prevent our other instances from also getting bogged down with these long running operations.
- The XSL instance that is least loaded for each transform.
  - UDB broadcast based predictive auction makes it easy to add more XSL processors with no configuration changes.



## *Enterprise Portal Summary.*

---

- Allowed us to start small.
- Allows incremental growth without jeopardizing existing investment.
- Allows incremental growth while the operational harder continues to run untouched.
- Gives users a consistent response rate even under heavy load conditions.
- Prevents long running expensive operations from negatively impacting end users.



## Case Study 2

- *Selected by seminar participants from a list of choices*

- **Scenario Definition**

1. Defining the basic requirements
2. Defining the Business Architecture
3. Defining the Technical Architecture
  1. Basic user or screen flow
  2. Defining the Hardware Architecture
  3. Defining the Software Architecture
4. Scalability Considerations



# Conclusion

- Auction or market oriented decision making techniques can be used to improve the efficiency of operating IT across the enterprise.

While there is not a wide range of tools readily available to support this activity it is not a terribly difficult add on in many instances. It will require demands from fortune 500 IT departments before we see these features show up in popular products such as Exchange.

- In addition to their general techniques can make a wider range of hardware available to a wider range of applications providing both better utilization of the existing resources and better overall response times for the end users.
- business impact bidding techniques provide an ideal basis for solving Scalability and robustness issues which have continued to plague IT implementation specialists. In addition these