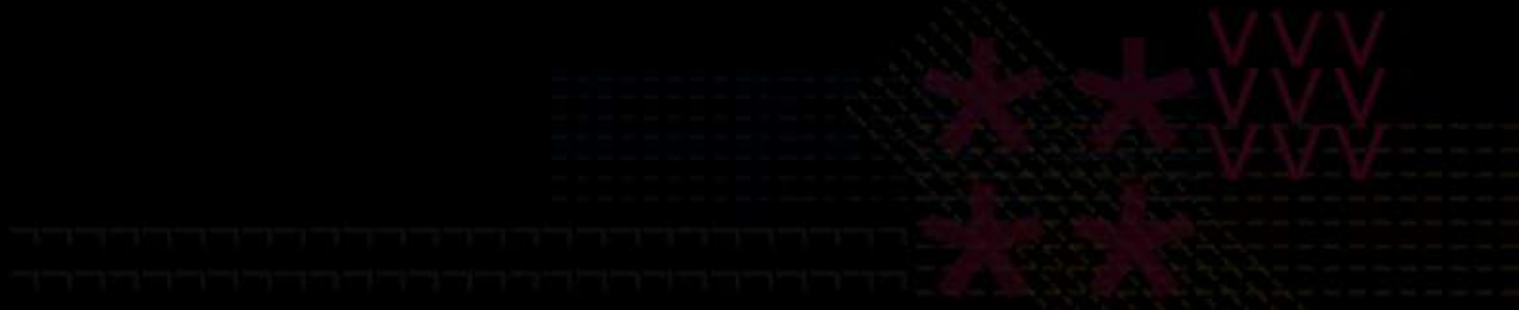
Best Practice of Building Scalable and Available Web Site on Windows Platform

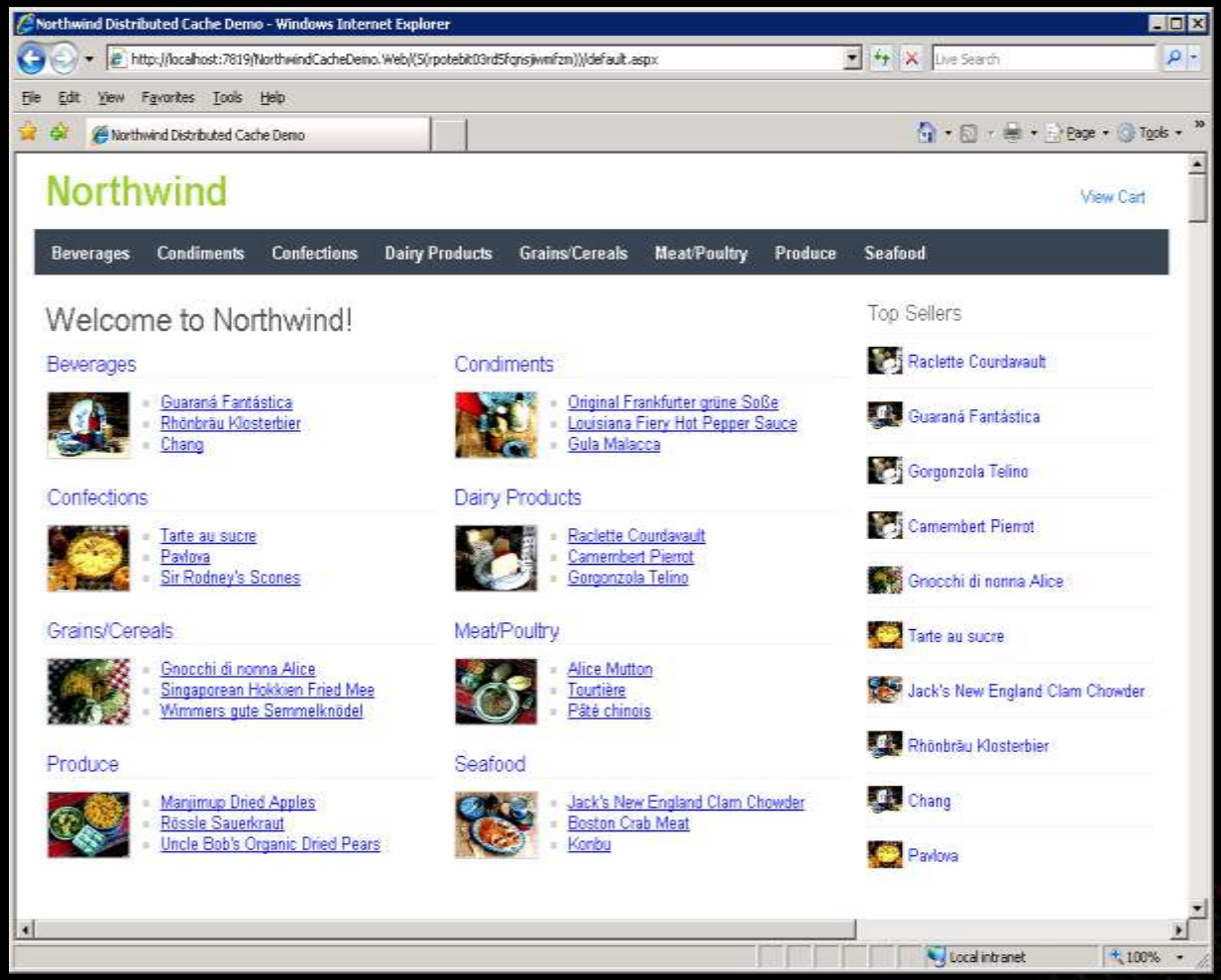
Robert Hu
IT Advisor
Developer & Platform Evangelism
Microsoft Taiwan

Overview

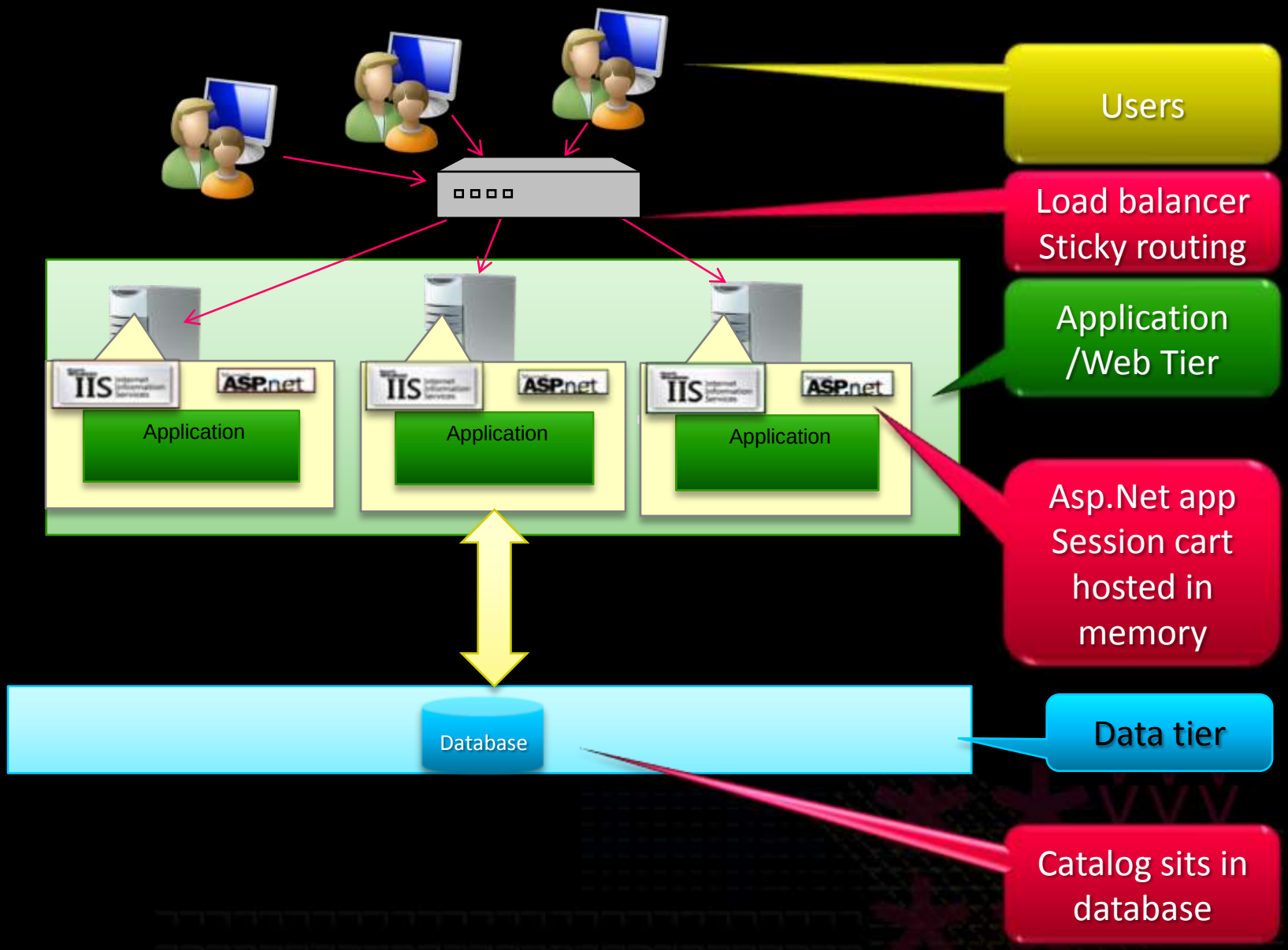
- The Problem
- The Solution – Load Balance
- Application Request Routing
- Break
- The Solution – Cache
- Project Velocity
- Q & A



Grocery Web Site



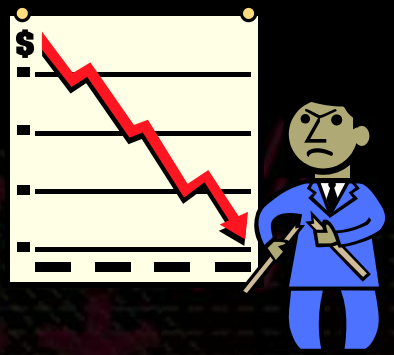
Typical Architecture



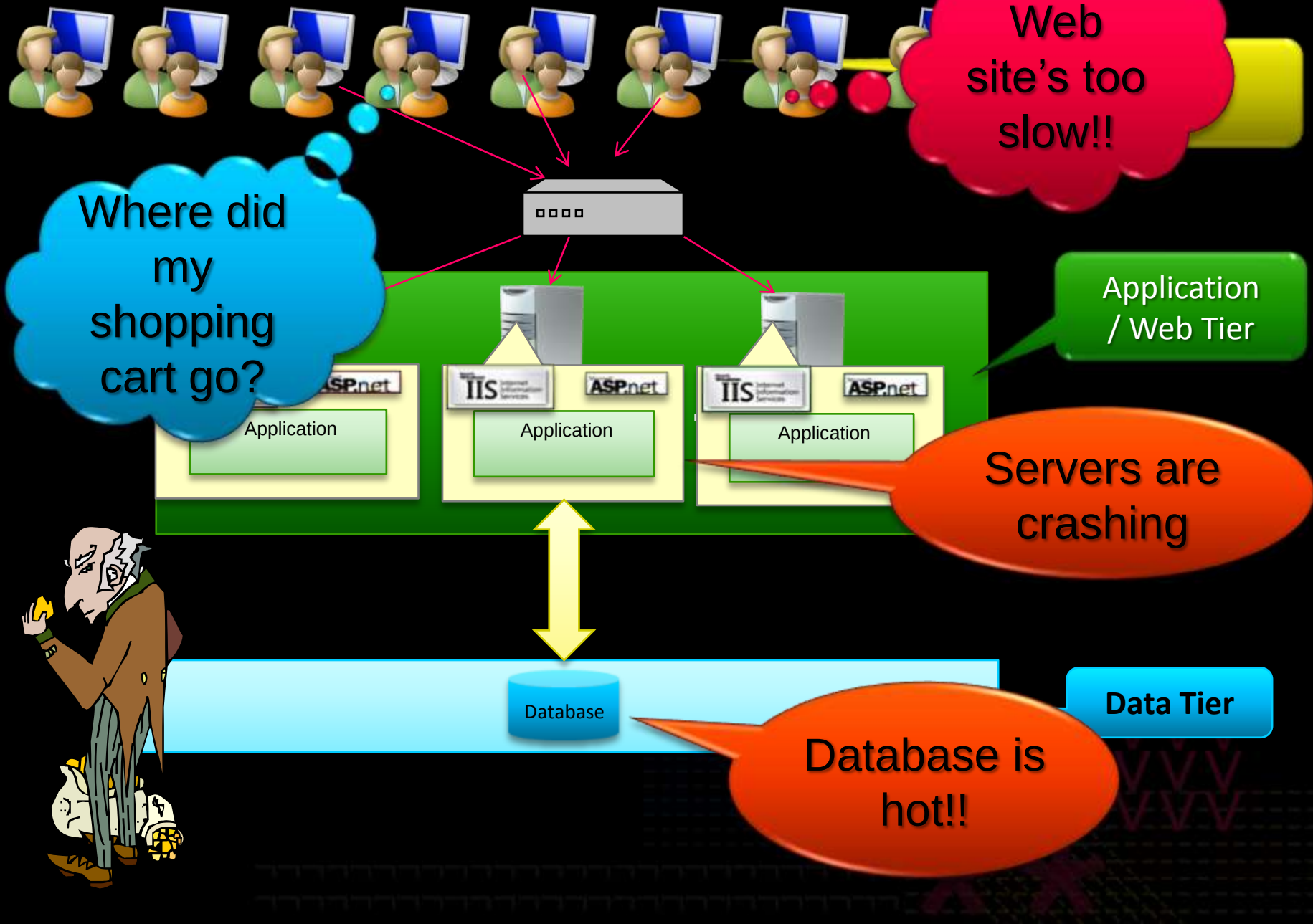
And Then…

Gas Prices Go up!!

Stock Price Goes Down!!



Your Web Site is now popular!!



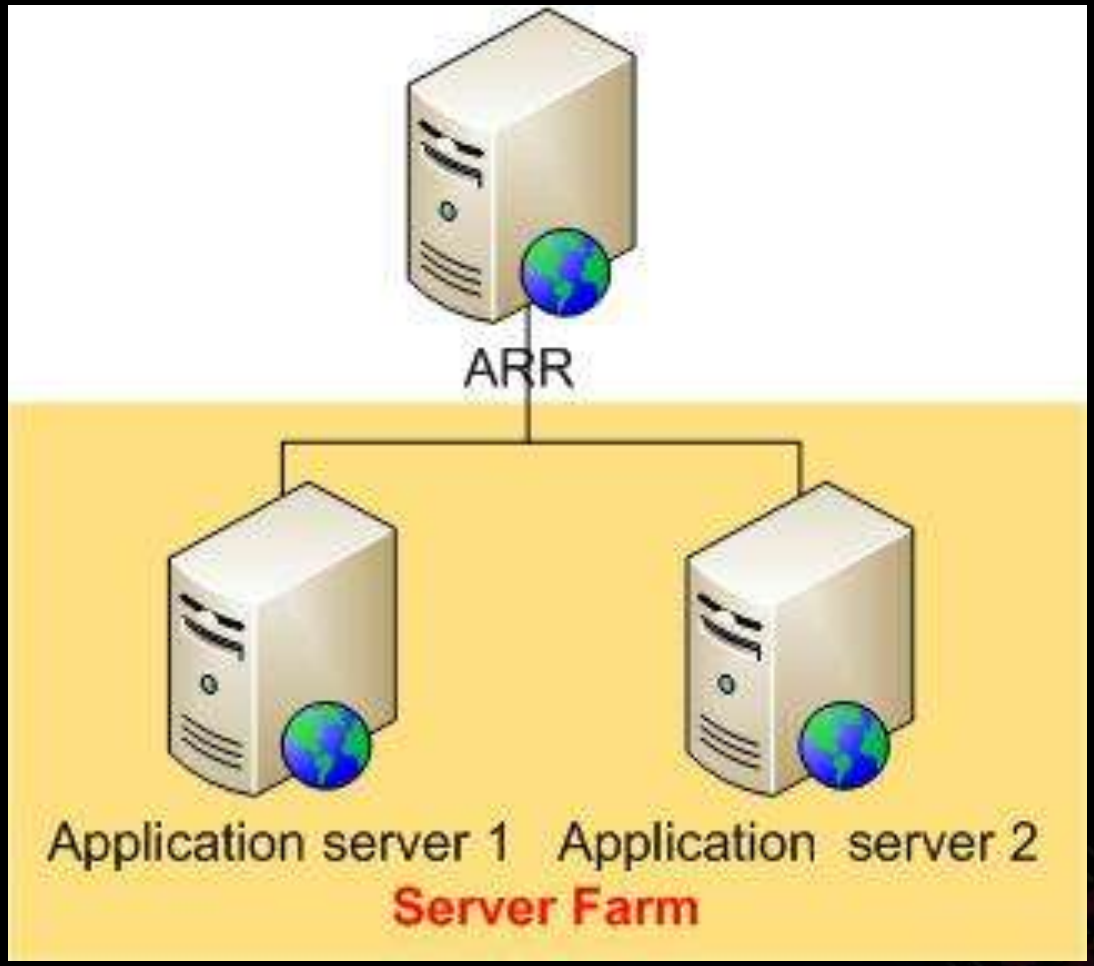
The Solution

Load Balance

Load Balance Solution

- Network Load Balance (Windows)
  - Good: Windows built-in
  - Bad: Affinity, Layer4 only, NAT, Converge time
- Hardware Load Balancer
  - Good: Layer3-7, Health Monitoring, Session
  - Bad: Expensive
- Application Request Routing (ARR)
  - Design to be Software “Hardware Load Balancer”

What is ARR?



http://www.iis.net/extensions/ApplicationRequestRouting

ARR Features

- HTTP based routing decision
- Load balancing algorithms
- Health monitoring
- Client and hostname affinity
- Multiple server farms for pilot and A/B testing scenarios.
- Management and monitoring
- Support for Failed Request Tracing Rules

Load Balance Algorithm

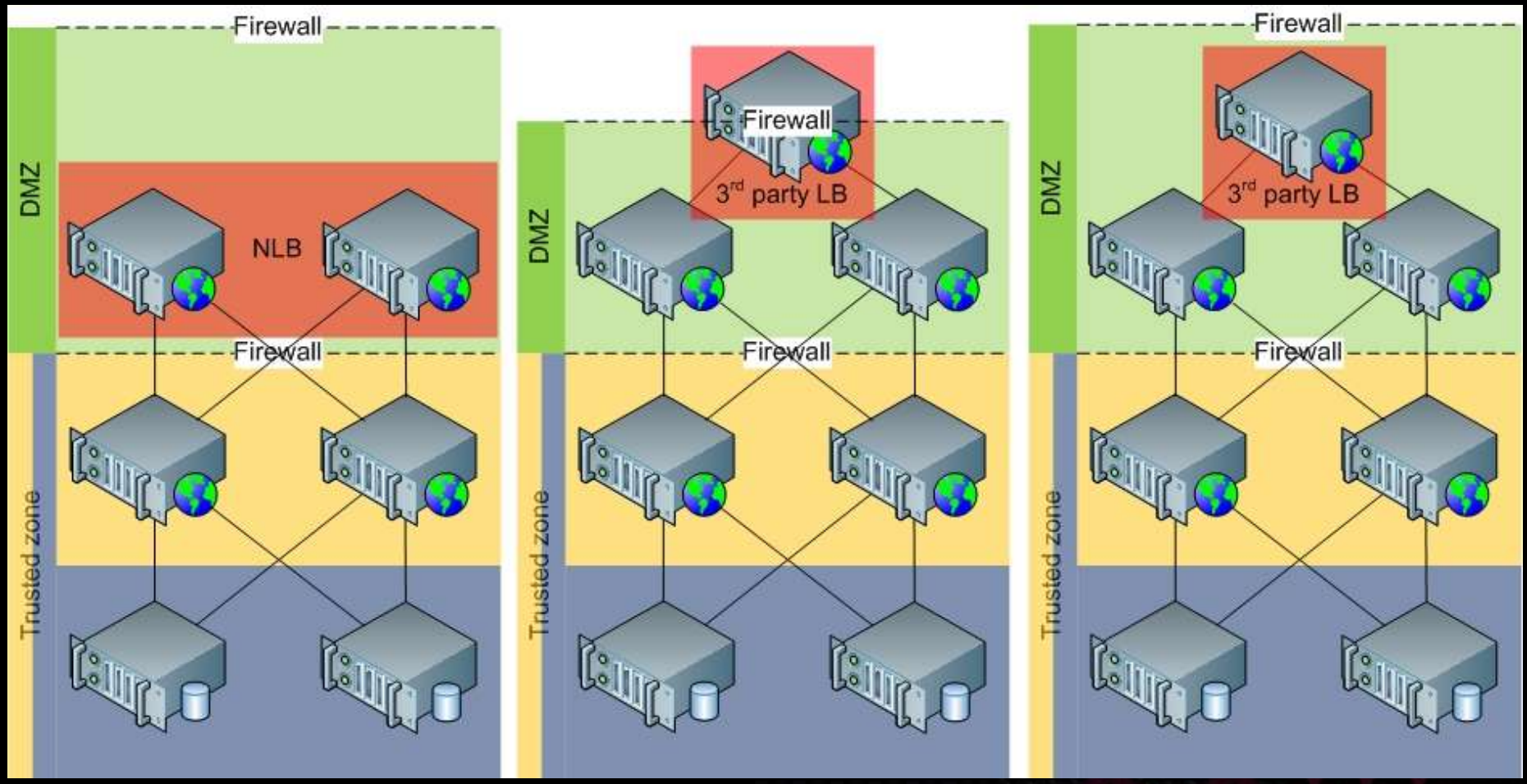
- Weighted round-robin
- Weighted total traffic
- Least current request
- Least response time
- Server variable hash
- Query string hash

Application Request Routing

ARR v2 Beta 1

- What's New
  - Enhanced v1 with disk cache
  - Edge cache in CDN/ECN scenario
- Feature
  - Disk-based cache
  - Cache-proxy
  - Cache hierarchy management (CARP)
  - Caching while serving response
  - Cache compressed object

ARR in Production Environment




Break

The Solution

Cache

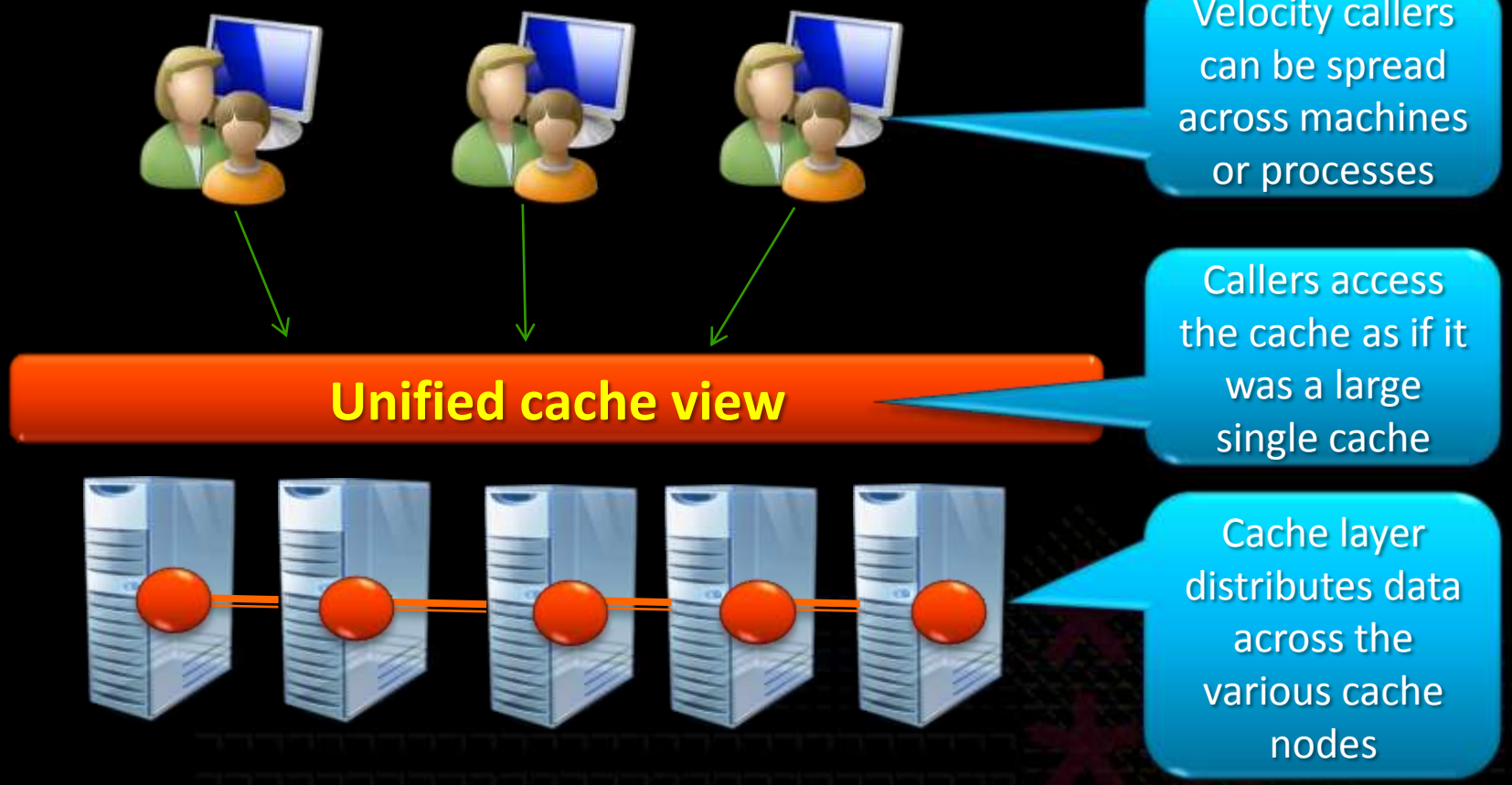
Cache Solutions

- ASP.Net cache
  - Good: built-in page cache, object cache
  - Bad: ASP.Net only, local cache
- IIS 7 output cache
  - Good: all IIS output (ASP, PHP, …)
  - Bad: limited option, local cache
- Project Velocity
  - Distributed memory cache
  - ASP.Net session provider

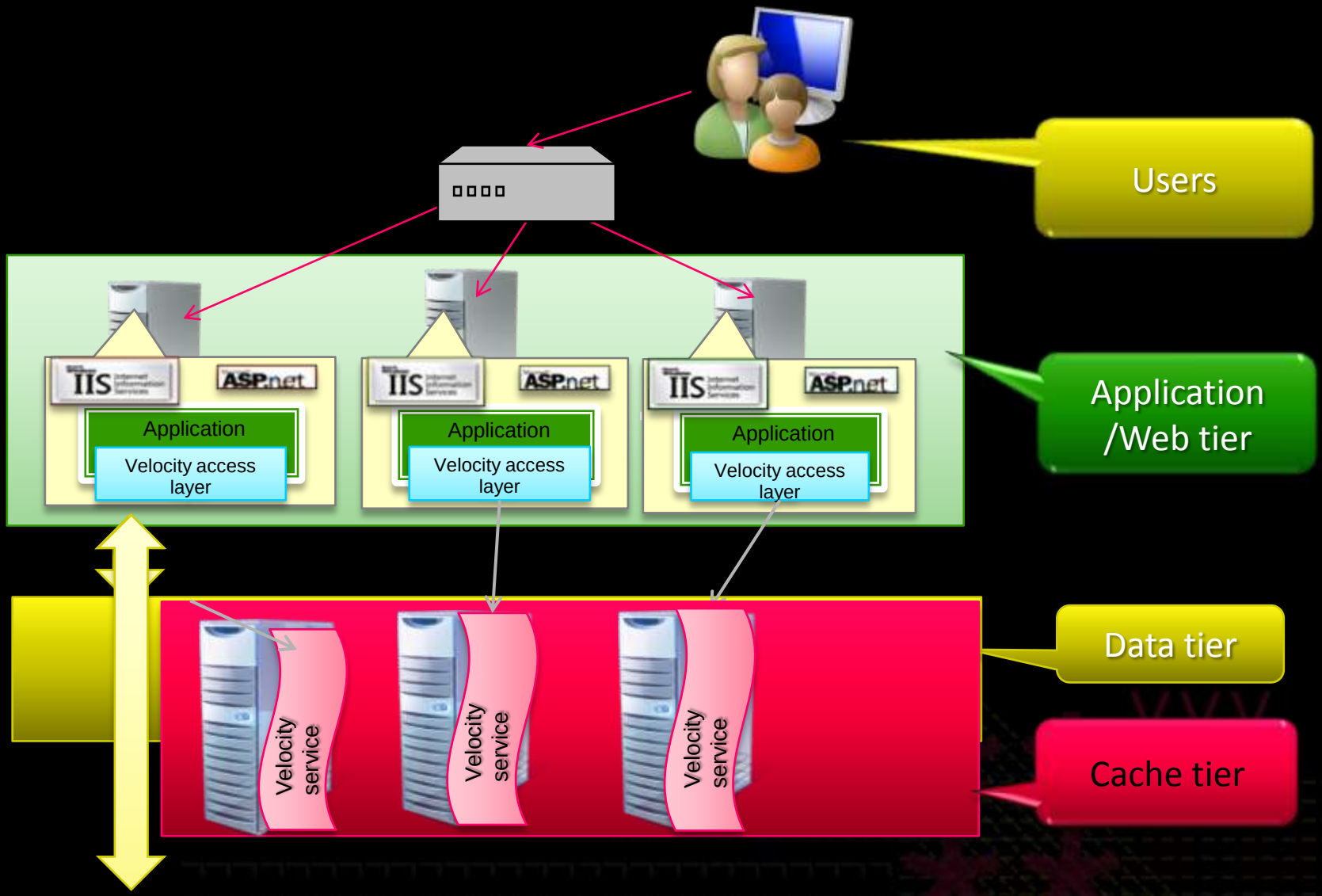
http://msdn.microsoft.com/en-us/data/cc655792.aspx

What is "Velocity"?

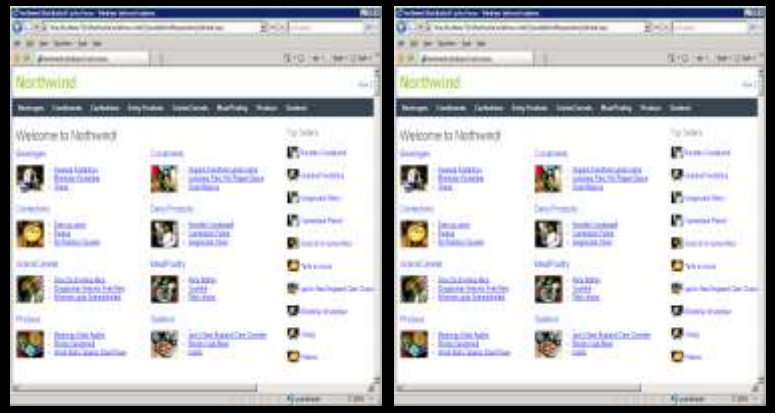
- **An explicit, distributed, in-memory application cache for all kinds of data**
  - **(CLR objects, rows, XML, Binary data etc.)**
  - Fuse "memory" across machines into a unified cache

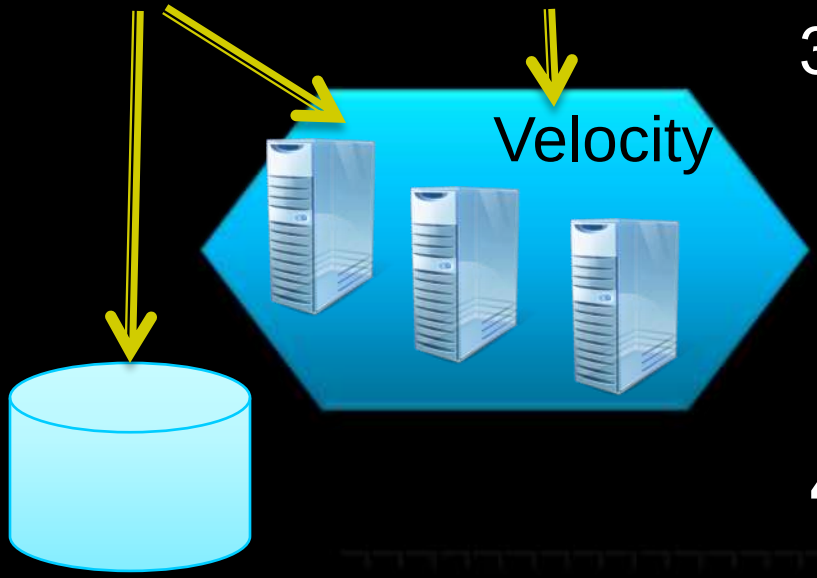


Where Does it Fit?



Why Velocity?





1. Share data across applications
   - No more sticky routing
2. Performance (quad core box)

Operation	Throughput	Latency
Read 2k	30,000 / sec	3 – 4 ms
Write 2k	18,000 / sec	3 ms

3. Scale out by adding more boxes

Operation	Servers	Throughput
Read 2k	1	30,000 / sec
Read 2k	2	58, 600 / sec
Read 2k	3	85, 500 / sec

4. High availability

Protect from web and cache server failure

And the Best Yet...

- Do all this at low cost!
  - Use inexpensive commodity hardware
  - V1 of velocity is currently planned to be a Free out of band release on MSDN *

* Requires windows server. Final SKUs/product positioning hasn't been figured out and some capabilities/number of nodes may require additional licenses

When Can I Get Velocity?

CTP1
- Teched

CTP2
- PDC 2008

CTP3
- End of Mar ‘09

RTM
- Mid 2009

Are Customers Using It?

Sector	Description
Media	Cache all media files and metadata; big media provider in US
Content	Big content provider in Europe
Content	MSDN forums is live on CTP2
Healthcare	One of the large health care ISV in the world; US
Finance	Big brokerage firm/banking; high performance computation
Gaming	Notifications for presence;
Enterprise	Largest mainframe migration to Windows
Manufacturing	Manufacturing process monitoring
Insurance	Session data for insurance purchase

Data to be Cached...

- Figure out which data to cache

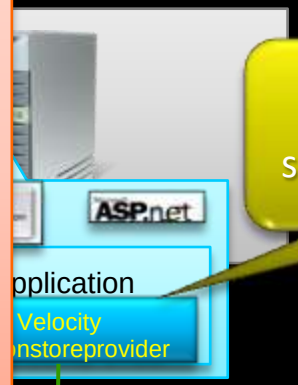
  Example
  - Catalog
  - Session/shopping cart
  - Inventory
- Cached data across sectors can be classified into
  - Reference
  - Activity
  - Resource

  Each of which has different characteristic

Session Cart Integration With ASP.Net

Load balance requests
No more sticky routing

```
<sessionState  mode="Custom"
customProvider="DataCacheSessionStoreProvider">
<providers>
    <add name="DataCacheSessionStoreProvider"
     type="Microsoft.Data.Caching.DataCacheSessionStoreProvider,
                ClientLibrary"
    cacheName="<YourNamedCache>"/>
</providers>
</sessionState>
```



Drop in velocity sessionstoreprovider

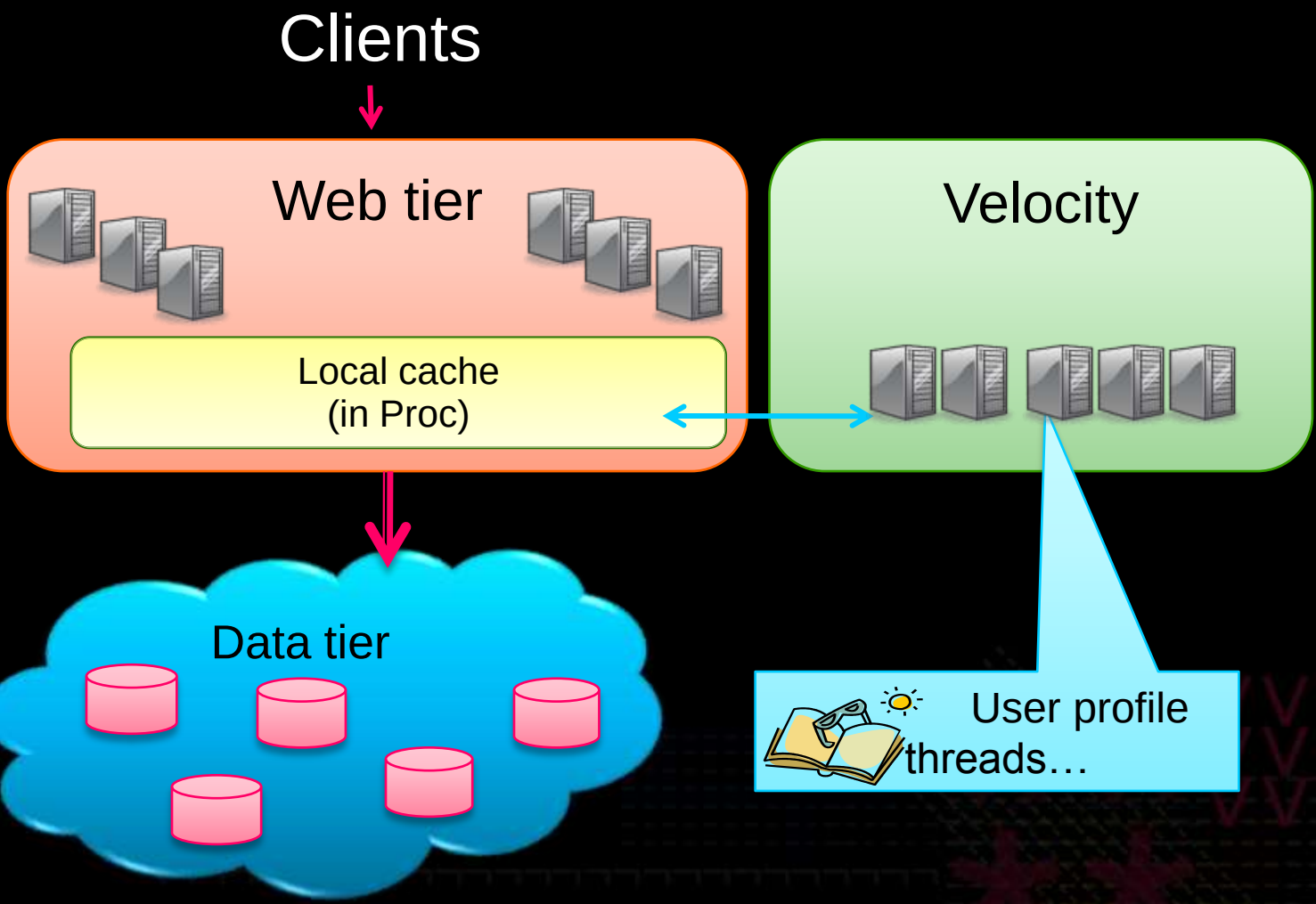
Session state stored in velocity

Scale your session store dynamically

Highly available

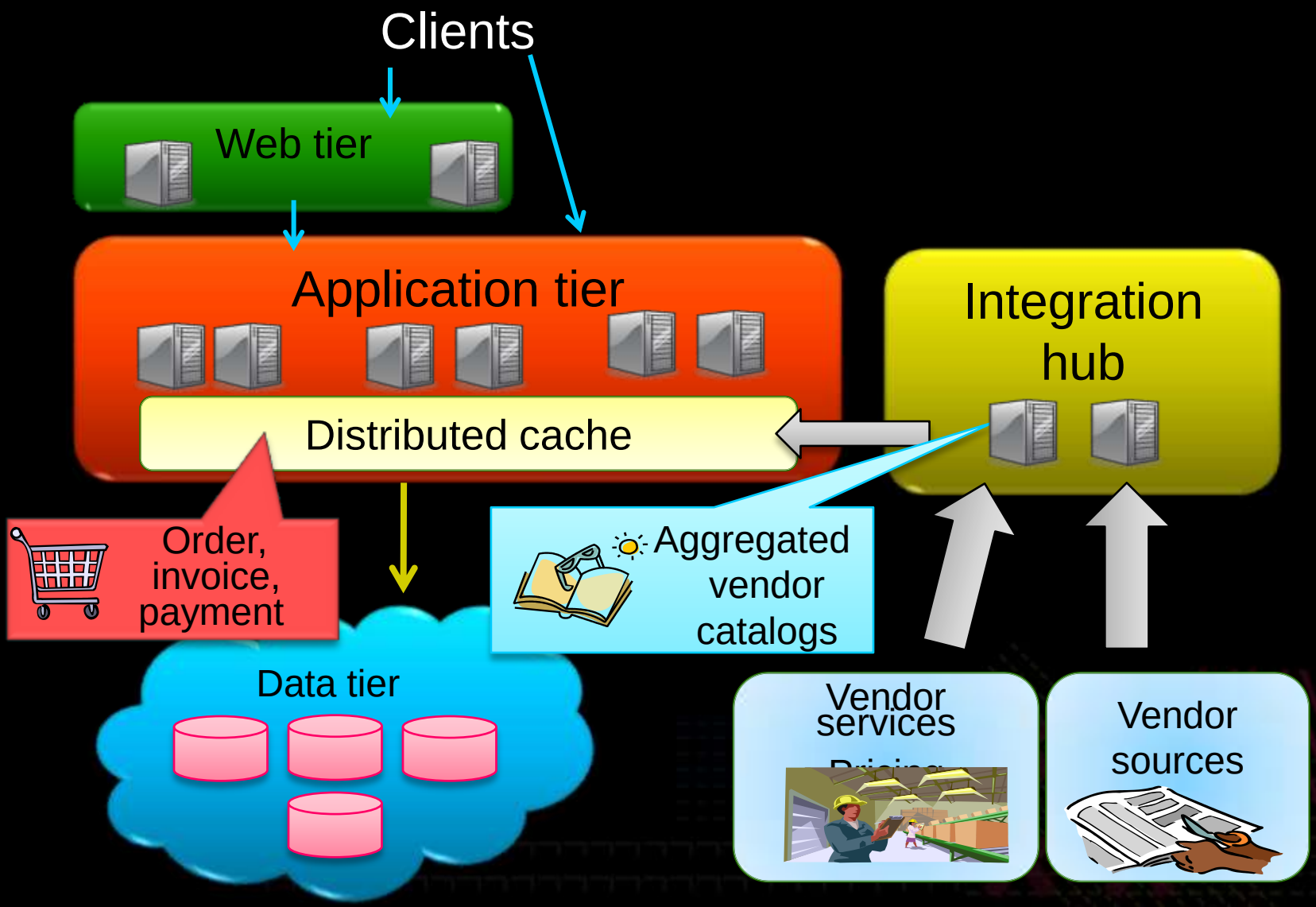
Caching Reference Data

Scenario: MSDN forums



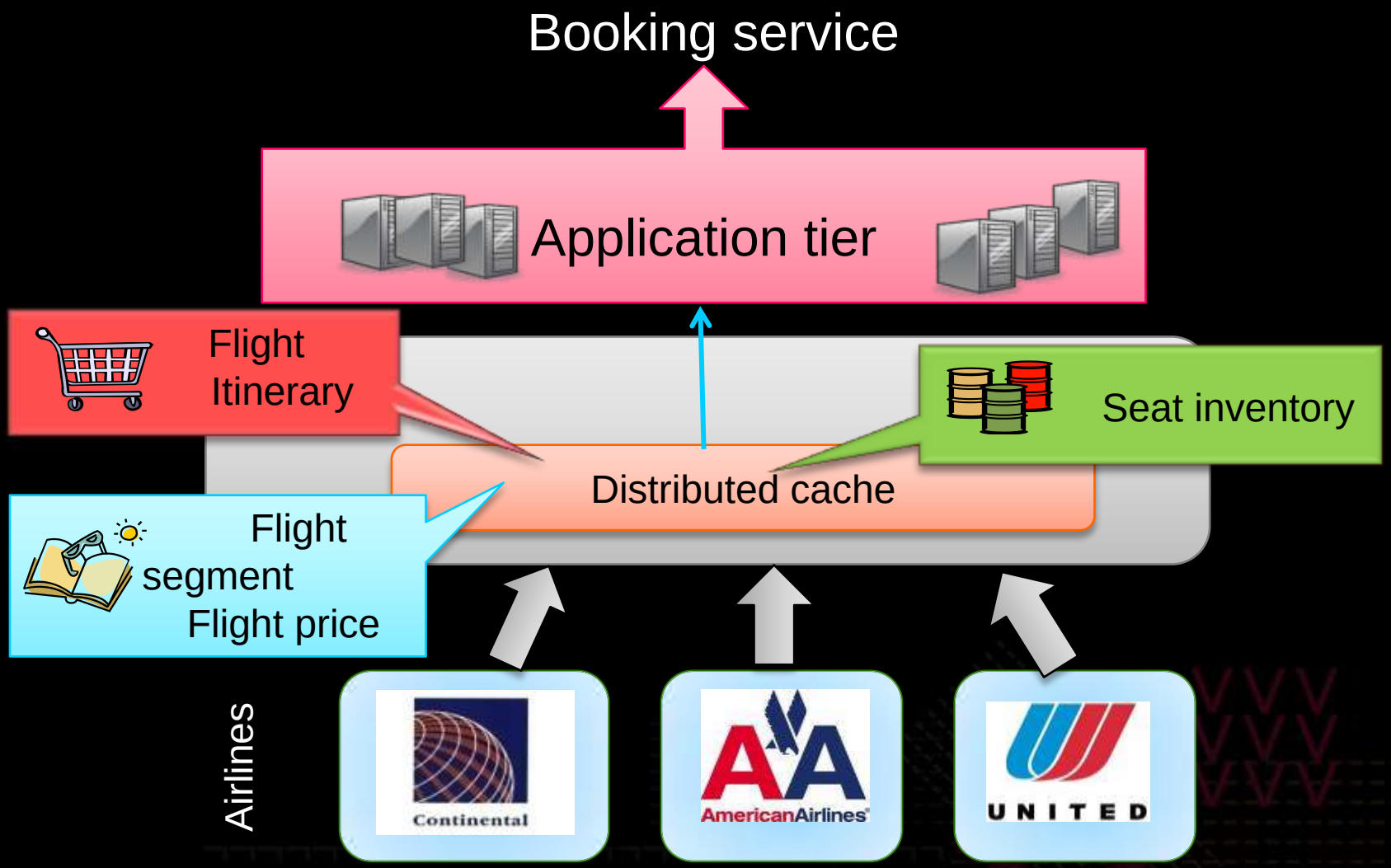
Caching Activity-Oriented Data

Scenario: Insurance

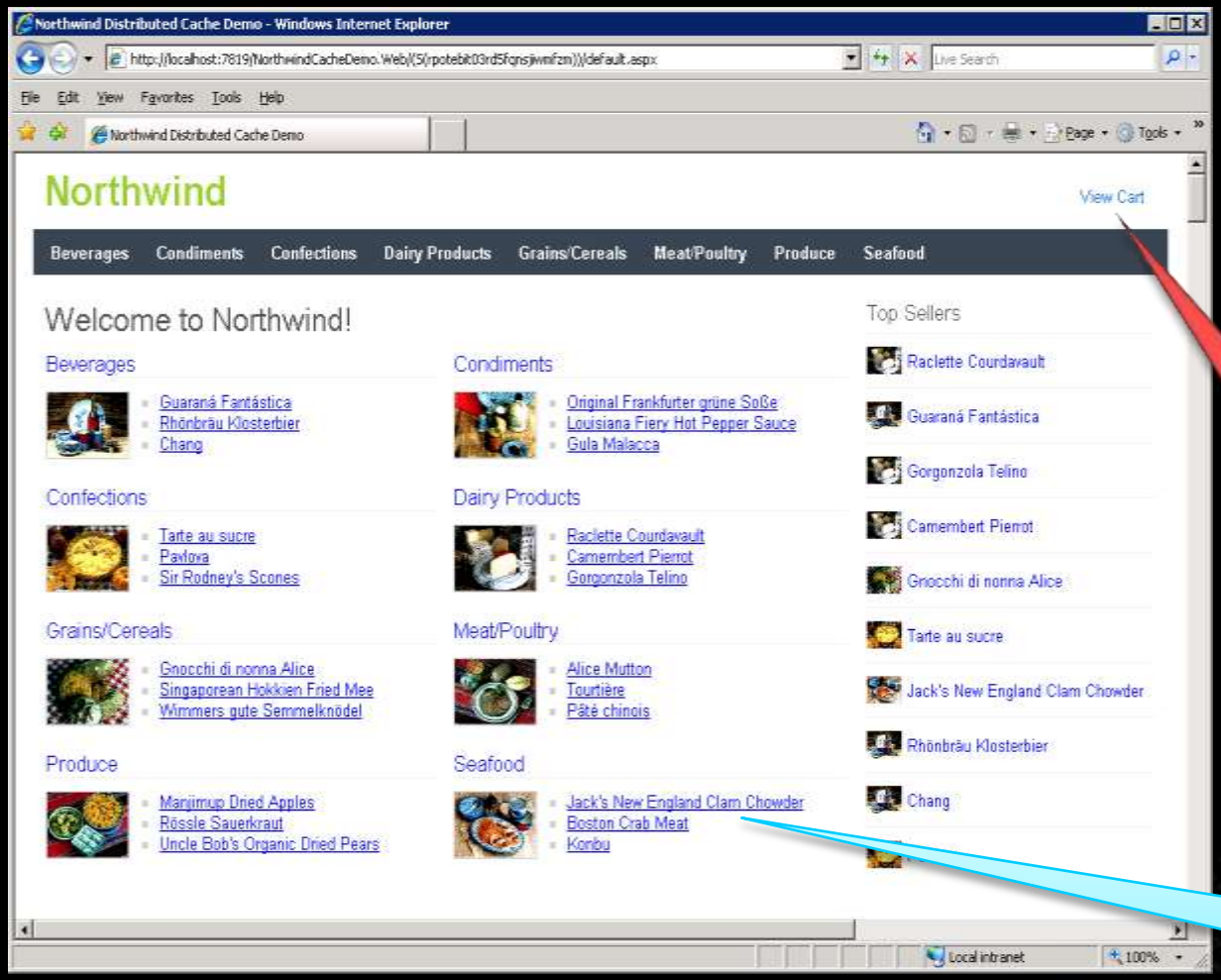


Caching Resource-Oriented Data

Scenario: Flight Inventory and Pricing



Data Classification



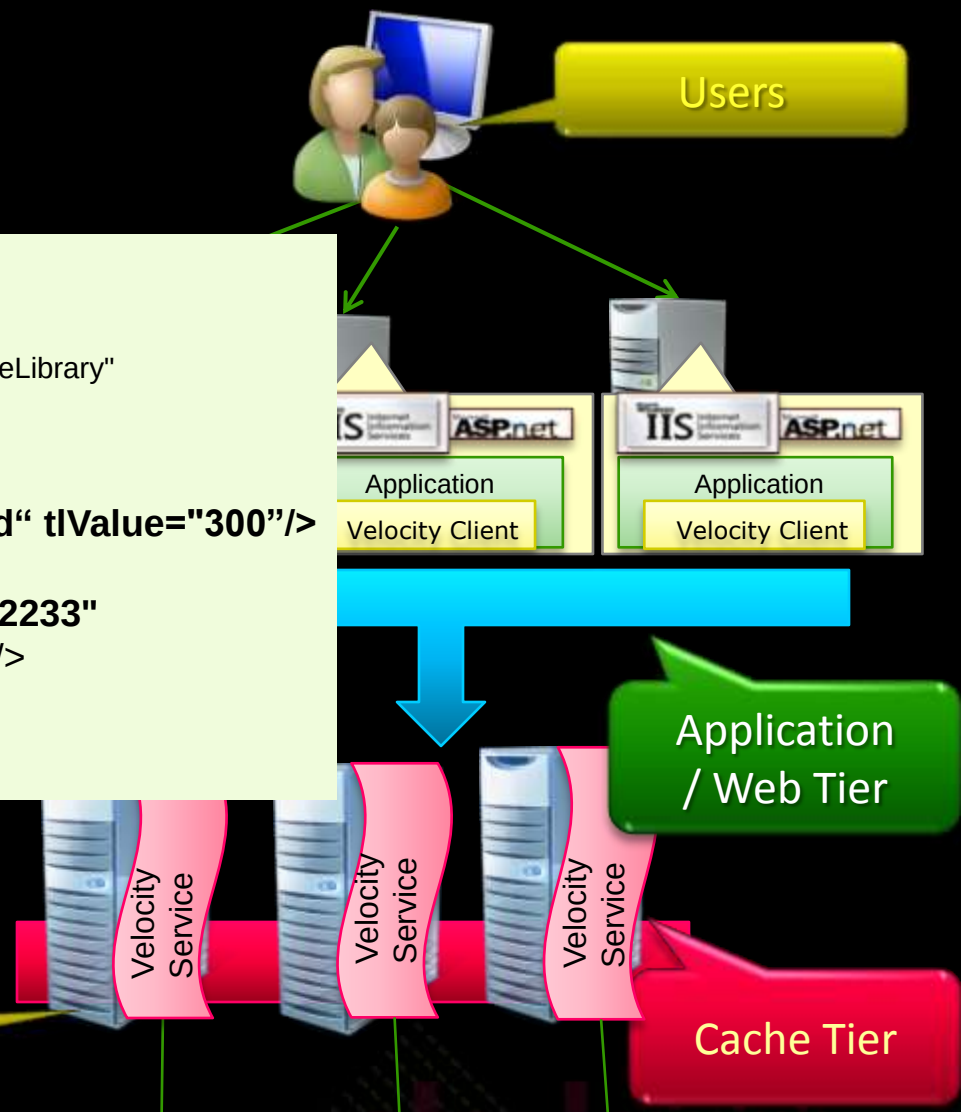
Understand Data Requirements

- Classify your data
  - Reference or activity or resource data
- Examine requirements
  - Performance
    - Throughput and latency
  - Consistency
    - How much staleness can be tolerated
  - Eviction
    - Is the data evictable?
  - Security
    - Is there any secure data?
  - Availability
    - Should data survive node or cluster failures?

Deployment

Copy Client DLLs and add app.config

```
<configuration>
  <configSections>
    <section name="dataCacheClient"
      type="Microsoft.Data.Caching.DCacheClientSection,CacheBaseLibrary"
      allowLocation="true" allowDefinition="Everywhere"/>
  </configSections>
  <dataCacheClient deployment="routing">
   <localCache isEnabled="false" sync="TTLBased" tlValue="300"/>
   <hosts>
     <host name="BL1CDB8083714" cachePort="22233"
           cacheHostName="DistributedCacheService"/>
   </hosts>
  </dataCacheClient>
</configuration>
```

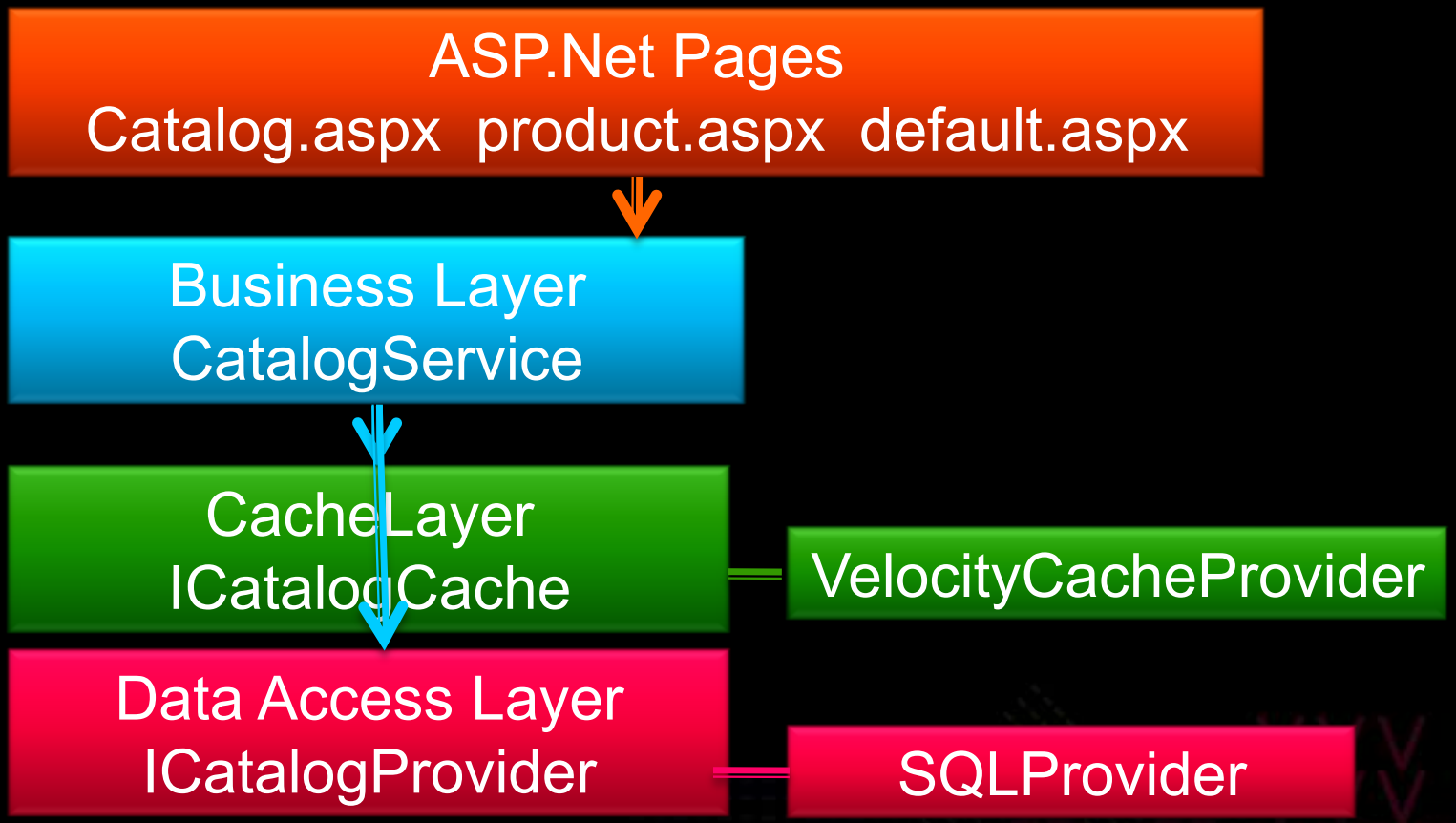


Install Windows Cache Service

Configuration Store
(Can be database, File share, etc.)
Stores Global Cache Policies
Stores Current Partitioning Information

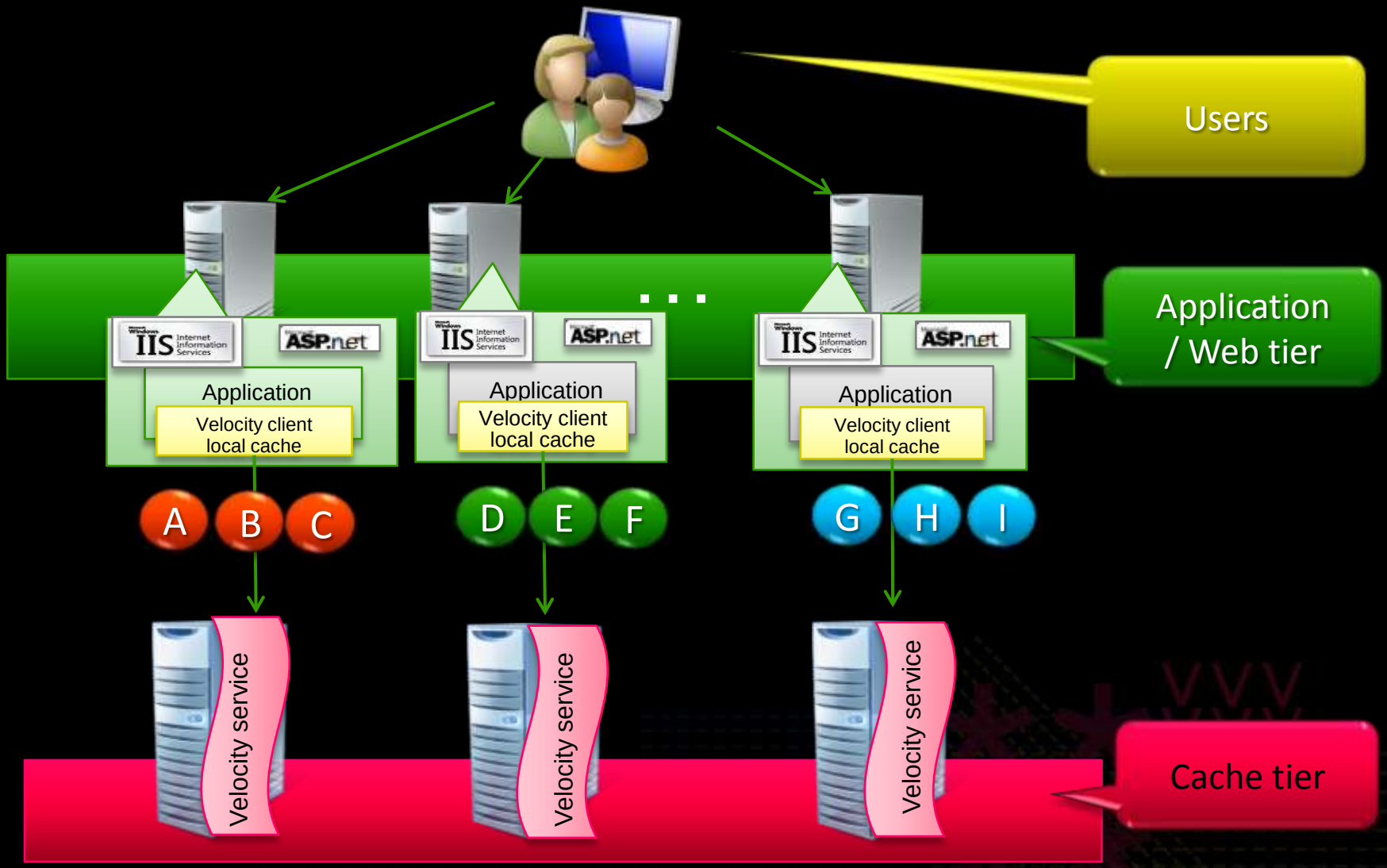
Grocery Web Site

Plugging velocity



Data Distribution

Partitioned cache



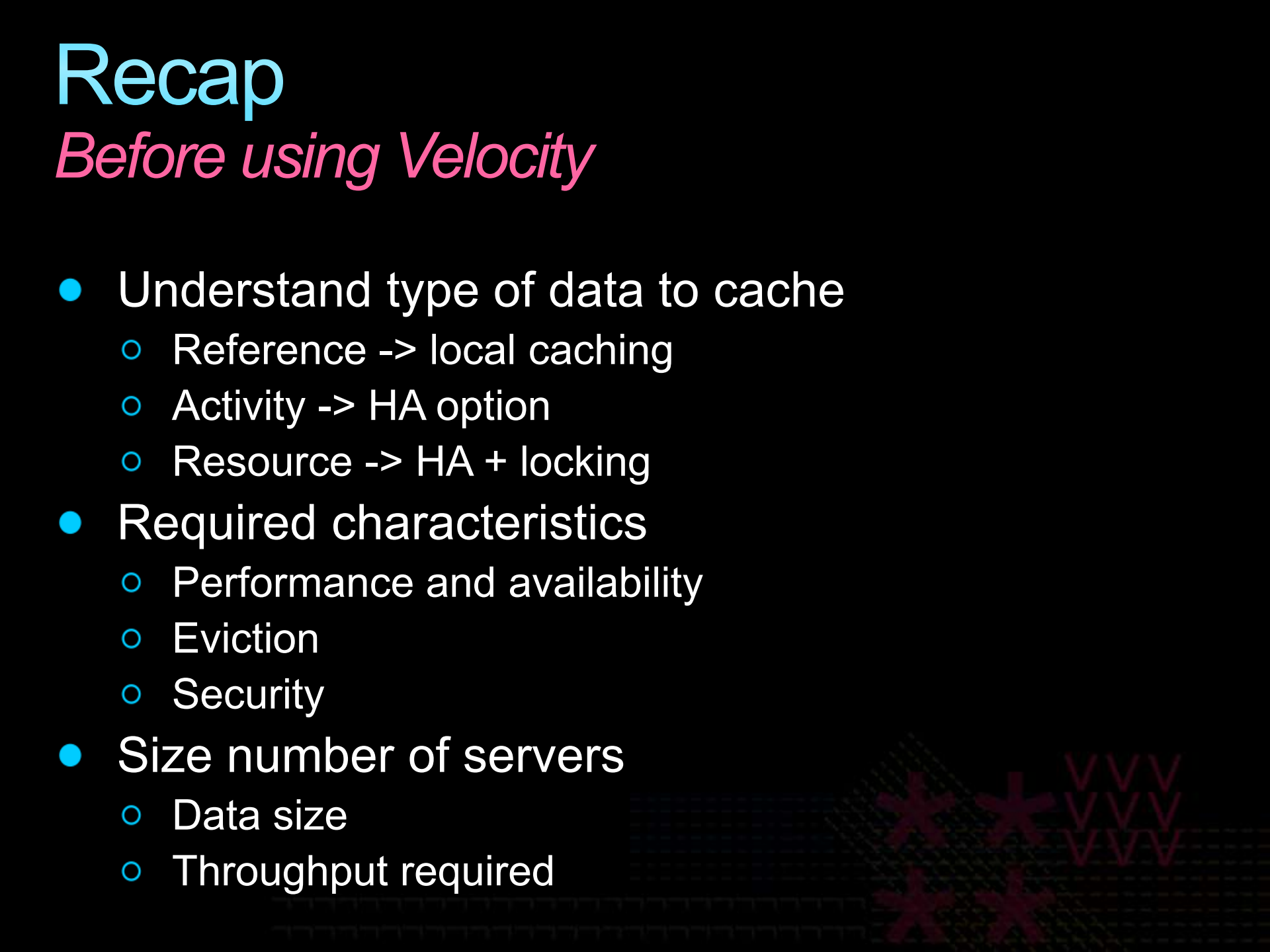
Size Your Servers

- Number of cache servers depend on
  - Amount of data to cache
  - Throughput required
  - Capacity of the cache server machines
- CTP3 saturation points
  - Less than 6k payload CPU saturates faster
  - More than 6k payload network saturates faster
- Assume linear scale for throughput
- Try it out – you can always add machines!!

Recap

Before using Velocity

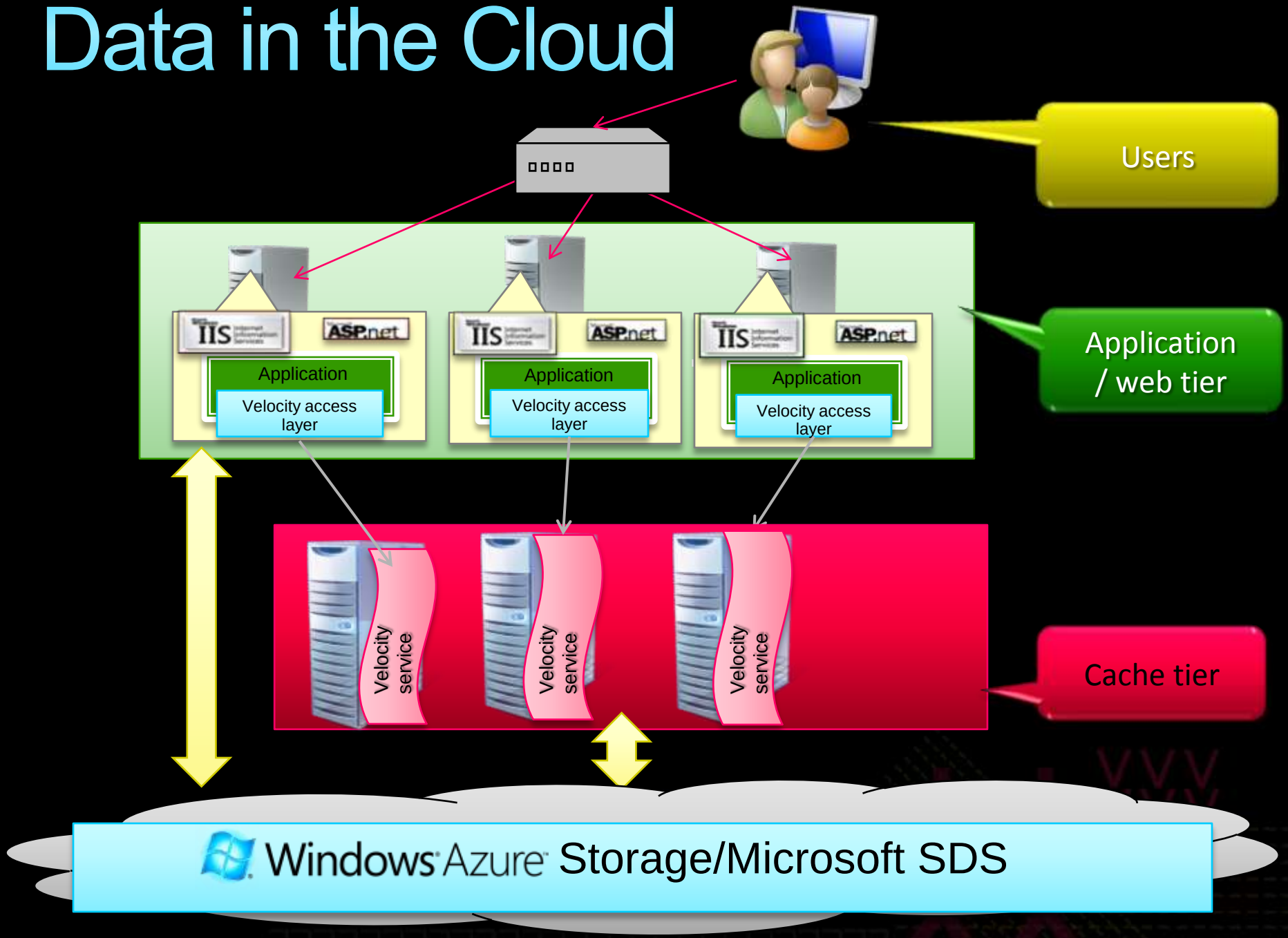
- Understand type of data to cache
  - Reference -> local caching
  - Activity -> HA option
  - Resource -> HA + locking
- Required characteristics
  - Performance and availability
  - Eviction
  - Security
- Size number of servers
  - Data size
  - Throughput required



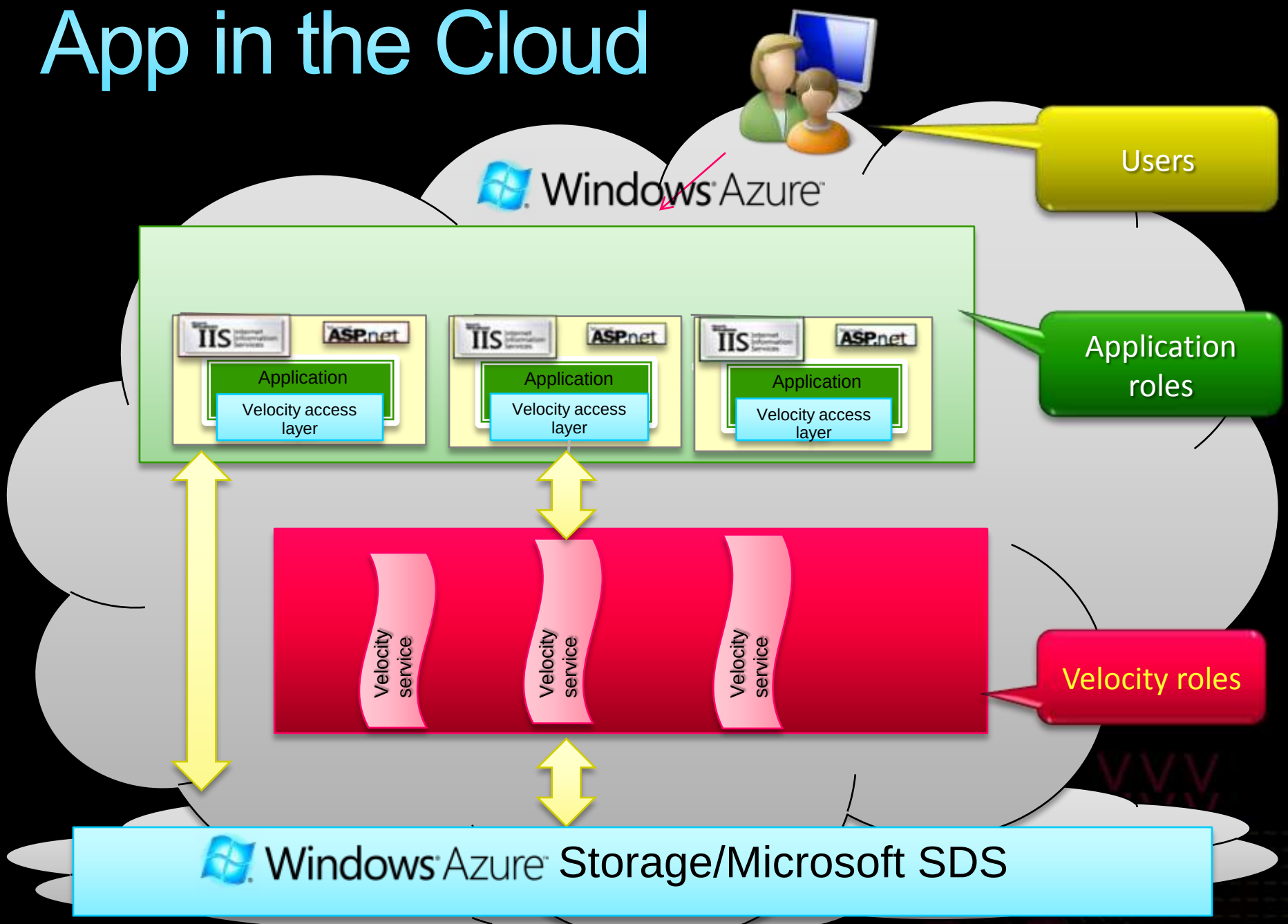
Application Growth

- You are a popular grocer!!
  - Lots of users!
- Irregular demands
  - Peaks during weekends or holidays
- Solution
  - Move to the Cloud!
    - Move data to the Cloud
      - Azure Storage, Microsoft SQL Services, etc.
    - Move the application to the Cloud
      - Azure compute

Data in the Cloud



App in the Cloud



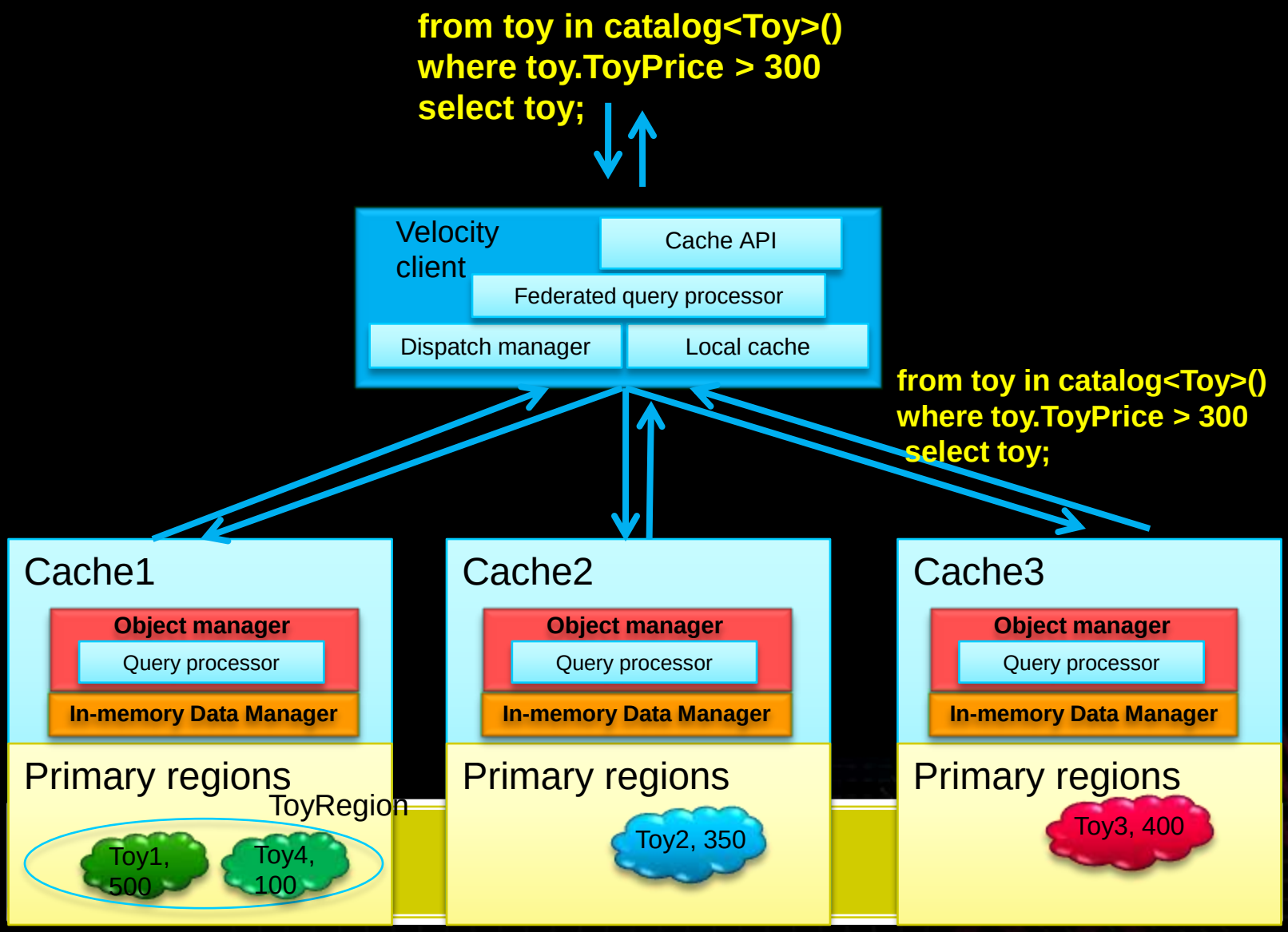
Better Integration with ASP.Net

- ASP.Net extends cache surface .Net 4.0
- Integrate with the application cache
  - System.Caching namespace
  - Velocity will be a provider for this namespace
- Granular session updates
  - IPartialSessionState lists keys changed
- Extensible output cache provider
  - Output caching not limited to single node

Persistence
Cache through

- Callback for read-through, write-behind
- Specified at named cache level
- Read-through
  - Called when item not present in cache
  - Callback returns the object/serialized bytes
- Write-behind
  - Writes to cache are queued
  - Callback called asynchronously in batches
  - Re-tries upon failure
- Bulk access APIs

Queriable Cache



Conclusion

- Velocity is an in-memory distributed cache for .NET
  - What is it? – a distributed in-memory cache
  - Why would I need it? – It can help your app scale, perform, and fault tolerant
- How to integrate Velocity in to a website
  - Understand the requirements and data to be cached
  - We saw how easy it is to integrate Velocity into a website
- Using Velocity can help your app scale, perform, and highly-available
  - We saw a how Velocity can scale and make app available
- Roadmap covers web, enterprise, and cloud scenarios
  - V1 Expected in Mid-2009
  - Velocity will be there when you go to the grid or cloud

Summary

- Load balance and Cache are key elements for scalable and available website
- Use Application Request Routing for load balance
- Use Project Velocity for ASP.Net session & application cache
- You‘ll have website grows with your business! ☺

Q & A

Microsoft®

Your potential. Our passion.™