CSIS 4135

Caching and Tracing

Where are the bottlenecks?

First mile?

- Origin infrastructure – gets the most attention when designing web apps

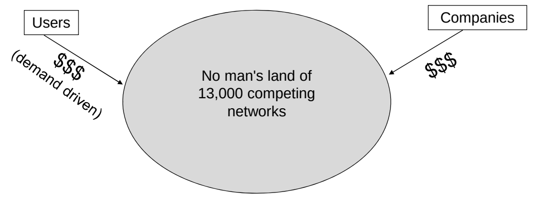
Last mile?

- User access – dial-up vs. broadband

Middle mile?

- The time data travels between server and client

Follow the money



Stuck in the middle

- Little incentive to build up capacity
  - Networks often want to minimize incoming traffic that they don't get paid for
- Reliability problems
  - Cable cuts, power outages, DDoS attacks
  - Border Gateway Protocol (BGP) vulnerability
- Increased traffic loads due to rich media, smart phones, etc.
- Distance and TCP protocol round trip time

Content delivery approaches

- Centralized hosting
  - Mirror sites closer to end users
- Content Delivery Networks (CDNs)
  - Offload cacheable content from origin server to shared networks (often on wrong side of middle mile)
- Highly distributed CDN
  - Puts content on the right side of middle mile (ISP)
- Peer-to-peer networks
  - Very highly distributed, but broadband generally fast in one direction only

Caching

- Temporarily store expensive resources in memory
  - Reduces load on web server
  - Faster access for clients
- Recent requests cached in memory
- Subsequent requests are served from in memory cache

Caching

- Many sites spend considerable effort generating the same web pages over and over
  - For example, a product catalog is updated each night, but is accessed tens of thousands of times a day
- Server-side caching can vastly improve performance and scalability
- ASP.NET provides support for
  - Page output caching
  - Data caching

Page Output Caching

- Entire web page output (HTML) is cached
- Must specify life of cached page (in seconds)

```
<%@ OutputCache Duration="25" VaryByParam="*" %>
```

- Can cache multiple versions of a page, by:
  - GET/POST parameters; use `VaryByParam`
  - HTTP header; use `VaryByHeader`
  - Browser type or custom string; use `VaryByCustom`

Data Caching

- Page level caching has a problem when pages are dynamically generated from database table information
  - What if table changes before cache expires?
  - Updated data won't appear until cache expires
- Fix this by enabling table level caching in the database

Table Level Caching

This requires running aspnet_regsql from the command line to enable caching on the database

Example:

```
aspnet_regsql -S .\SQLSERVER2005 -E -ed -d Basics -et -t Quotes
```

-S = server	-E = Windows authentication
-ed = enable db	(-U = SQL Server login)
-d = database	
-et = enable table	
-t = table	

Table Level Caching

- The aspnet_regsql command creates a cache table and a "trigger" that updates the cache table whenever the original table changes.
- This enables polling the database every couple seconds to see if anything has changed and updating the cache table

Polling

To use this feature, some configuration of the web application is needed

- Web.config:

```
<system.web>
  <caching>
    <sqlCacheDependency enabled="true"  pollTime="2500" >
      <databases>
        <add connectionStringName="Basics" name="Basics"/>
      </databases>
    </sqlCacheDependency>
  </caching>
```

- Page directive:

```
<%@ OutputCache Duration="25" VaryByParam="none"
                                        SqlDependency="Basics:Quotes" %>
```

Notification

- Relatively new feature for SQL Server that eliminates polling and only notifies the application when a change occurs
- Page directive:

```
<%@ OutputCache Duration="25" SqlDependency="CommandNotification" %>
```

- Add a `global.asax` file to the project, and in the `Application_Start method`:

```
System.Data.SqlClient.SqlDependency.Start(
            ConfigurationManager.ConnectionStrings["Basics"].ConnectionString);
```

Notification

- There is a gotcha when using notification
- Form SELECT statements like this example:

```
SELECT Quotation, Author, Category, Creation_date FROM dbo.[Quotes]
```

Note:

- Can't use *
- Table name must have owner prefix

Partial Page Output Caching

- Can cache a portion of a page by placing it in a User Control
- Can cache multiple versions of a User Control on a page, by including a page attribute or by class attributes
- The Substitution control can also be used to bypass caching for part of the page

Caching in the Browser

- Don't confuse server-side page output caching with how the browser and proxy servers cache the page
- Use `Response.Cache` to specify HTTP cache headers
  - Contains a `HttpCachePolicy` object

Tracing

- A convenient way to get lots of information about the current request
- Can be done at
  - Page level
  - Application level

Page Level Trace

- Add the following to the *@Page* directive at the top of the .aspx file
  - Trace="true"
- This results in trace output appended to the content of the page
  - Cookies, forms, query strings, etc.
- Don't want this visible to clients

Application Level Tracing

- More flexible and practical
- Logs trace output for review later
- Hides trace output from users of the page
- Enabled in Web.config

Application Level Tracing

In Web.config:

```
<trace
      enabled="true"
      requestLimit="10"
      pageOutput="false"
      traceMode="SortByTime"
       localOnly="true"
/>
```

Application Level Tracing

Access trace output using the applications URL and Trace.axd

http://localhost/basics/Trace.axd