Optimization Practices Contract API

Joshua Van Hoesen

Lead Software Engineer

Accounting System Integrators

Jvanhoesen@asillc.com

Three points of interest

- Utilization of ‘ReturnBehavior’
- Avoid Graph logic by creating Generic Inquiry for data retrieval
- Multi-Threading

‘ReturnBehavior’ is your friend.

- ReturnBehavior.All - Returns all data points and child records on a request.
- ReturnBehavior.OnlySpecified – Returns only those data points specified, by default no child records.
- ReturnBehavior.OnlySystem – Returns only system data points such as ID and no child records.
- ReturnBehavior.None – Returns no data points and no child records.

267 Sales Orders

Number of records	ReturnBehavior	Details Included	Seconds
267	All	Yes	219.5
267	All	No	172.2
267	OnlySpecified [7]	Yes	3.2
267	OnlySpecified [7]	No	.56
267	OnlySystem	No	.43
267	None	No	.38

Scale of speed

$$\frac{|v_1 - v_2|}{\frac{(v_1 + v_2)}{2}} x100 = ?$$

Number of records	ReturnBehavior	Details Included	Minutes
5000	All	Yes	60
5000	OnlySpecified [7]	Yes	1
1,000,000	All	Yes	12k (200 Hrs)
1,000,000	OnlySpecified [7]	Yes	200 (3.3 Hrs)

between $V_1 = 60$ and $V_2 = 1$

$$\frac{|V_1 - V_2|}{\frac{(V_1+V_2)}{2}} \times 100$$

$$= \frac{|60 - 1|}{\frac{(60+1)}{2}} \times 100$$

$$= \frac{|59|}{\frac{61}{2}} \times 100$$

$$= \frac{59}{30.5} \times 100$$

$$= 1.93443 \times 100$$

$$= 193.443\% \; difference$$

Generic Inquiries

- Retrieving data through an inquiry can avoid costly Graph logic.
- No cache attached, no problem! Keep those selectors simple.

Multi-Threading, Why before the do

- Large external data store to import.
- Large number of records fitting separate criteria to retrieve.
- Large numbers of consecutive records to be processed.
- Large number of records need to be summarized and created.
- Greater utilization of server resources

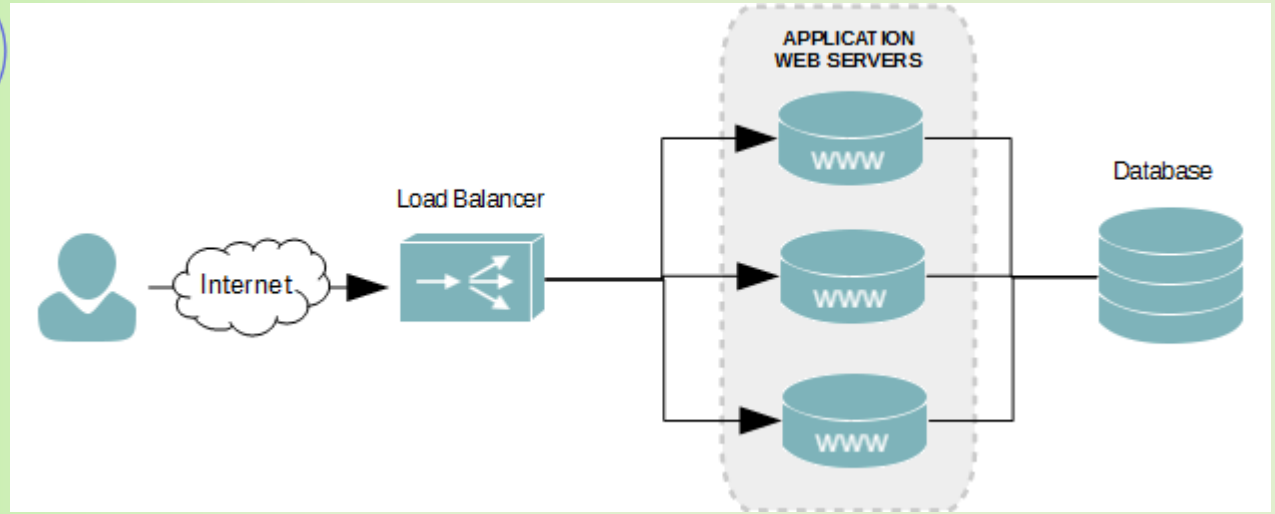
Multi-Threading, Is it faster?

Records to Create	Number of Threads	Time to Completion
500	1	113 Seconds
500	5	37 Seconds
500	10	35
500	50	47

- When applied to the proper criteria
- Non-Linear gain, overhead is a thing
- Limit number of threads in relation to processors available to instance.
- Reserve threads for session management and system processing

Sample Configuration

- Triple 8 core application servers
- Load balancer
- (24 cores * 2 threads per core) – 2 ‘reserved’ threads = 46 threads



Notes of implementation

- System.Net.ServicePointManager.DefaultConnectionLimit = 100;
- ClientBase<Acumatica.DefaultSoap>.CacheSetting = CacheSetting.AlwaysOff;
- Proper Implementation of IDisposable

ServicePointManager.DefaultConnectionLimit

- “By default the number of simultaneous connections to a server is controlled by the ServicePoint.ConnectionLimit property. This property is set on this object when the ServicePointManager creates the ServicePoint object for your connections to a host. Connections to the server are queued and will be served by the connections to the server as they become freed from use. The default number of connections is set to 2.” ~ http://msdn.microsoft.com/en-us/library/system.net.servicepoint.connectionlimit.aspx

- This value should be set relative to the number of threads you are utilizing for Instance requests.

CacheSetting.AlwaysOff

“WCF client applications use the ChannelFactory<TChannel> class to create a communication channel with a WCF service. Creating ChannelFactory<TChannel> instances incurs some overhead because it involves the following operations:

1. Constructing the ContractDescription tree
2. Reflecting all of the required CLR types
3. Constructing the channel stack
4. Disposing of resources

To help minimize this overhead, WCF can cache channel factories when you are using a WCF client proxy.”

- Default behavior will mean your code is actually almost always running **in one session.**

IDisposable

```
protected virtual void Dispose(bool disposing)
{
    if (disposing)
    {
        client.Logout();
        client.Close();
    }
}
```

- Important to logout before closing connection to avoid maxing out user sessions and forcing log-offs.

Remember!

- Utilization of ‘ReturnBehavior’
- Avoid Graph logic by creating Generic Inquiry for data retrieval
- Multi-Threading can be a boon.

Thank you for your time!

May your code be quick and your clients be happy