Westat

Blaise 5 in a Production Environment

Authors: Paul Segal, Mangal Subramanian, Ray Snowden,
Richard Frey, Mike Florcyzk

Presentation Overview

- Blaise 5 Server and Client Options.
- Blaise 5 API Use Case.
- Stress Testing Blaise 5 Application

Blaise 5 Server Roles

- Web server – processes requests and send responses to end users
- Resource Server – applies layout and formats Blaise 5 for different devices types
- Data Entry Server – interprets/applies Blaise rules
- Data Server – reads and writes to the Blaise data source
- Survey Manager supports Server Role administration

Blaise 5 Server Park

- A Server Park is the collection of physical servers on which the Blaise server roles are installed
- Considerations for Server Park design
  - Scalability – scale out or scale up
  - Continuity – eliminate single points of failure
  - Security – isolate servers from end-user access and practice secure server configuration and management practices
- Blaise 5 production configuration also requires a database server running a supported DBMS – SQL Server, MySQL, Oracle

Blaise 5 Client Options

- Connected options – supported in first release
  - Web browsers – various standard browsers and mobile browsers
  - Blaise 5 mobile app for iOS – thin client
- Disconnected options
  - Blaise 5 under Windows for CAPI (*Linux?*)
  - Blaise 5 mobile app for disconnected operations – thick client

Blaise 5 API: Overview

- Until Manipula becomes available in Blaise 5, the API becomes the mechanism for data manipulation.
- Blaise 5 has four API's organized functionally.
  - MetaAPI
  - DataRecord
  - DataLink
  - SessionData
- Blaise 5 API is built entirely with .NET Framework (.NET 4.0).

Blaise 5 API: Use Cases

- Typical use of Blaise API in a production environment may include the following use cases;
  - Preloading data into the instrument for a case.
  - Returning case-level status information.
  - Extracting case results.
- Blaise Agent;
  - .NET Wrapper that provided a generic implementation of the above three Blaise API functions.
  - Re-usable across projects.

Blaise 5 API: Observations

- The API has been re-organized compared to Blaise 4.
- The organizational by functions seems more logical and natural.
- Use of the new Blaise API for new and old users involves a bit of a learning curve.
- On the other hand there is the advantage of someone without have knowledge of Manipula being able to work with Blaise data using the .NET API.

Blaise 5 API: References by Function

- Initialization
  - MetaAPI to access DataModel object
  - DataLinkAPI to access DataLink object
- Preloading data for a case
  - DataRecordAPI to create DataRecords and set values from source
  - MetaAPI to access non-response values (SpecialAnswerNames)
  - DataLinkAPI to write DataRecords to DataSet
- Returning case-level status information
  - DataRecordAPI to retrieve DataRecord and read values
  - MetaAPI to access key fields
- Extracting case results (all records)
  - DataLinkAPI to access Blaise 5 DataSet
  - DataRecordAPI to process DataRecords in DataSet
  - MetaAPI to access key fields

Blaise 5 API Tips: Preloading data for a case

- To set the value for a Blaise 5 field (fld):
  - for an enumerated field, use the integer value

```
fld.DataValue.EnumerationValue = int.Parse(fieldValue)
```

  - for a date field, use a DataRecordAPI DataValue (dv) and its DateValue

```
dv.DateValue = DateTime.Parse(fieldValue)
fld.DataValue.Assign(dv)
```

  - for a non-response value use SpecialAnswers

```
fld.DataValue.SpecialAnswer =
MetaAPI.Constants.SpecialAnswerNames.DontKnow
```

Blaise 5 Stress Testing: Overview

- Factors affecting capacity and performance of the Blaise application are;
  - Number of users and expected distribution of users over time.
  - Size of the data model.
  - Size and configuration of the server hardware.
- Stress Testing Goals;
  - Define a consistent and reliable stress testing approach.
  - Determine key stress testing parameters.

Blaise 5 Stress Testing: Test Design Factors

- Test Server Configuration;
  - One Virtual Blaise 5 server running Windows 2008 server.
  - SQL server 2008 running on a separate database server.
- Record a test script with parameters and attributes that best replicates a real life user interaction scenario.
- Determine simultaneous users threshold.
- Record and monitor response times.

Blaise 5 Stress Testing: Tool and Scripts

- The tool used for stress testing was the Microsoft Visual Studio Test Manager.
  - Provides ability to record a Web Test script by recording a user session against a Blaise 5 web survey.
  - The Web Test can be associated with a Load Test.
  - Load Test allows one to specify various test parameters like - No. Of users, Test run time, Think time.
- The test tool allows to record all key strokes that a user inputs during a survey session including time taken between questions.

Blaise 5 Stress Testing: Factors

- Number of processors and amount of memory on the server has effect on performance (easier to make since they were in a Virtual environment).
- Configuration of client machines an important factor.
- Mixture of short and long running tests.

Blaise 5 Stress Testing: Metrics

- A key metric in web applications is the response time.
- Requests Queued – The more requests queued indicates server bog down, until you get a service unavailable error.
- Worker Process re-starts – Important monitor, each worker process has a set memory allocated, so if there are memory leaks it would re-start.

Blaise 5 Stress Testing: Tool Metrics

- In addition to these performance counters, following key metrics captured by the test tool were used to determine the health of the application under various loads.
  - Tests/Sec
  - Tests Failed
  - Avg. Test Time (sec)
  - Pages/Sec
  - Avg. Page Time (sec)
  - Requests/Sec
  - Requests Failed
  - Avg. Response Time (sec)