Improving Crystal Reports Performance in Visual Studio .NET Applications

Applies to:

Crystal Reports 9.1 to Crystal Reports 2008 (12.0.x) when used in applicable versions of Visual Studio .NET.

For more information, visit the Business Objects homepage.

Summary

This article discusses techniques, tips and best practices that should lead to improved report performance in Visual Studio .NET applications using the Crystal Reports SDK for Visual Studio .NET.

Author: Ludek Uher

Company: SAP

Created on: 18 February, 2009

Author Bio

Ludek Uher is a Senior Engineer with Technical Customer Assurance, SAP Business Objects. He specializes in the SDKs supplied with Crystal Reports and BusinessObjects Enterprise.

Table of Contents

Expected Performance 3
Comparison of Report Performance in the Crystal Reports Designer and Custom Application 3
Using Progress Bar to Monitor Report Progress 4
Troubleshooting Slow Performance 4
Performance Impact is on Report Load 5
- Report Design Issues 5
- Report Options 6
- Old or Incorrect Runtime 6

Performance Impact is on Database Connection / Data Retrieval 7
- ADO .NET Dataset 7

Coding Issues 9
Performance Issues when Printing 9
Other Tips and Tricks when Using the Crystal Reports .NET SDK 10
- Using Sessions 10
- Too Many Reports Being Processed by the Report Engine 10

Related Content 11
Copyright 12

Expected Performance

An application will only be as fast as the reports it uses. A large part of tuning an application for scalability is ensuring that all the reports are designed in a way such that they will perform at an optimum speed. Report size is one of the obvious factors in report performance. When designing for scalability, consider using smaller reports. Also, reports designed for desktop applications may not be appropriate for web-based applications.

Typically, the load of the first report in an application will be slower than all subsequent report loads. Depending on the complexity of the report, the time to load the first report may be as much as 20 to 30 seconds or longer. Sometimes, report load time will be significantly longer. This is a normal and expected behavior as a number of Crystal Reports assemblies, com files etc., have to be loaded and initialized before a report can be processed. Once these have been loaded, they are reused and thus the performance improves on subsequent report loads. The only way to improve the end user experience on first report load is to offset the initial load time on application load or some other part of the application where the performance hit is acceptable.

Comparison of Report Performance in the Crystal Reports Designer and Custom Application

It is always useful to compare the performance of a report running in an application to performance in the Crystal Reports designer. Slow report performance in the CR designer will usually not improve when that same report is ran from a custom application. Typically, the opposite will be the case.

This is because setting of a database connection in the CR designer happens immediately on report load when it prompts for the database logon parameters. At runtime, the logon method for the tables doesn't make the connection right away. Rather it happens on the call to the view / print to printer / export method. Thus these comparisons need to be done with consideration.

Using Progress Bar to Monitor Report Progress

The **processingIndicatorText** and **processingIndicatorDelay** settings, new in Crystal Reports 2008, can be used in web applications to indicate report processing activities that are longer than 500 milliseconds. For more details, search for **processingIndicatorText** or **processingIndicatorDelay** in the Crystal Reports 2008 .Net Developer Guide.

Troubleshooting Slow Performance

Irrespective of the application being Web or Windows based, consult the article Optimizing Reports for the Web.

Even though the article specifies “Web” in the title, many of the points for improving report performance will apply to Windows applications.

Next, determine where the performance hit is coming from. With Crystal Reports, there are at least four places in your code where slow downs may occur. These are:

- report load
- connection to the data source
- setting of parameters
- actual report output, be it to a viewer, export or printer.

Temporarily inserting the following code into your project may help to determine the slow down points:

```
private void button_View_Click(object sender, System.EventArgs e)
  {
   System.DateTime startTime, endTime;
   System.TimeSpan diffTime;

   if(crReportDocument != null)
   {
    crReportDocument.Close();
    crReportDocument.Dispose();
    crReportDocument = null;
   }

   // reseting the display text back to zeros
   this.textBox_Start.Text = "0";
   this.textBox_End.Text = "0";
   this.textBox_Diff.Text = "0";

   // instantiate a new ReportDocument object
   crReportDocument = new ReportDocument();

   // mark the time before we Load the report from disk
            startTime = System.DateTime.Now;
   this.textBox_Start.Text = startTime.TimeOfDay.ToString();

   if (this.textBox_Filename.Text.Length<1)
    return;

   crReportDocument.Load(this.textBox_Filename.Text,
CrystalDecisions.Shared.OpenReportMethod.OpenReportByDefault);

   // mark the time after the report was successfully loaded
   endTime = System.DateTime.Now;
   this.textBox_End.Text = endTime.TimeOfDay.ToString();
```

```
    // calculate the time difference, and display the time in TotalMilliseconds
    diffTime = endTime.Subtract(startTime);
    this.textBox_Diff.Text = diffTime.TotalMilliseconds.ToString();

  }
```

Performance Impact is on Report Load

Slow report load may be caused by report design issues, report options, and old or incorrect runtime.

Report Design Issues

- Linked OLE object inserted into a report is not where the report expects it to be. If this is the case, the report will attempt to locate the object, potentially consuming large amounts of time.
- The subreport option **Re-import when opening** is enabled (right-click the subreport(s), choose format subreport, look at the subreport tab). While this is a great design time feature, it is a time consuming process and should be used judiciously.
- The number of subreports the report contains and in which section the subreports are located will impact report performance. Minimize the number of subreports used, or avoid using subreports if possible. Subreports are reports within a report, and if there is a subreport in a detail section, the subreport will run as many times as there are records, leading to long report processing times. Incorrect use of subreports is often the biggest factor why a report takes a long time to process.
- Use of **Page N of M**, or **TotalPageCount** special fields. When the special field **Page N of M** or **TotalPageCount** is used on a report, it will have to generate each page of the report before it displays the first page. This will cause the report to take more time to display the first page of the report.
- Graphics, while a powerful feature of Crystal Reports, need to be used with care. When displaying a report using the Web Forms report viewer, for each chart or image on the report page, the report engine must create a temporary JPG file and the browser must make a separate request to the web server to retrieve it.

  The Web Forms viewer also periodically searches through the images folder for old images and deletes them. This disk access and creation of temporary files will reduce report performance.
- Remove any unused object from the report. This includes database tables, formulas and parameters and running total formulas. Even if these objects are not used in a report, the report engine will attempt to evaluate the objects, thus affecting performance.
- Suppress unnecessary report sections. Even if a report section is not used, the report engine will attempt to evaluate the section, thus affecting performance.
- If summaries are used in the report, use conditional formulas instead of running totals when ever possible.
- Whenever possible, limit records through Record selection Formula, not suppression.
- Use SQL expressions to convert fields to be used in record selection instead of using formula functions. Since SQL Expression Fields are added to the SELECT clause of the SQL Query the SQL expressions are evaluated server side.

  For example, if you need to concatenate two fields together, instead of using a formula, create a SQL Expression Field.
- Using one command table or Stored Procedure or a Table View as the datasource can be faster if you return only the desired data set.
- Perform grouping on the database server. This applies if you only need to return the summary to your report but not the details. It will be faster as less data will be returned to the reports.

Report Options

- The **Verify on First Refresh** option (File > Report Options) forces the report to verify that no structural changes were made to the database. There may be instance when this is necessary, but once again, the option should be used only if really needed. Often, disabling this option will improve report performance significantly.
- The **Verify Stored Procedure on First Refresh** option (File > Report Options) is essentially the same function as above. However, this option will only verify stored procedures.

Old or Incorrect Runtime

If at all possible, use the latest runtime, be it with a custom application or the Crystal Reports Designer.

- The latest updates for the current versions of Crystal reports can be located on the SAP support download page.
- Incorrect Crystal Report version will cause incompatibility with Microsoft Visual Studio .NET. For details of which version of Crystal Reports is supported in which version of VS .NET, please refer to the Crystal Reports assembly versions and Visual Studio .NET Wiki.
- For the correct Crystal Reports runtime files please refer to the Crystal Reports for Visual Studio .NET Runtime Distribution Wiki.

Performance Impact is on Database Connection / Data Retrieval

Database fine tuning, which may include the installation of the latest Service Packs for your database, must be considered. The report engine internally attempts to perform a certain level of database connection pooling. That is, if three users hit the same ASPX page with the same report on it, the same database connection will be used to perform all three queries.

One thing the report engine does not do is pooling of the query itself. For each report job (each **ReportDocument** instance), a database query must be performed. If the query takes a while to process, or if the set of data returned is very large, then performing this query on a per-user basis could turn out to be the bottleneck of the application. Using ADO .NET datasets may be an option in this case. See the section in this Article titled ADO .NET Dataset for more detailed discussion of ADO .NET Datasets. Other factors affecting data retrieval:

- network traffic
- the number of records returned

  If a SQL query returns a large number of records, it will take longer to format and display than if was returning a smaller data set. Ensure you only return the necessary data on the report, by creating a Record Selection Formula, or basing your report off a Stored Procedure, or a Command Table that only returns the desired data set.
- the amount of time the database server takes to process the SQL query.

  Crystal Reports send the SQL query to the database, the database process it, and returns the data set to Crystal Reports.

Where is the Record Selection evaluated? Ensure your Record Selection Formula can be translated to SQL, so that the data can be filter down to the server. If a selection formula cannot be translated into the correct SQL, the data filtering will be done on the local client computer which in most cases will be much slower.

One way to check if a formula function is being translated into a SQL is to look at **Show SQL Query** in the CR designer (Database > Show SQL Query). Many Crystal Reports formula functions cannot be translated into SQL because there may not be a standard SQL for it. For example, control structure like IF THEN ELSE cannot be translated into SQL. It will always be evaluated on the client computer. For more information on IF THEN ELSE statements see Note 1214385 in the Notes database.

Whenever possible, link tables on indexed fields. While linking on non-indexed fields is possible, it is not recommended. Database performance can also be compared to that of other programs or utilities. However, it must be realized that no two applications will perform identically.

ADO .NET Dataset

Reporting from datasets has its advantages, but it shouldn't necessarily be used as an exclusive data access mechanism. If your application creates a dataset for the sole purpose of the report, and especially if the query is simple such as `SELECT field1, field2 FROM table`, then there is little value in using a dataset.

Having Crystal Reports connect directly to the database and perform the query would be faster and require less memory. Datasets are useful when data from multiple data sources needs to be combined, or perhaps the data inside the dataset needs to be massaged after the query is run.

The Crystal Reports ADO.NET (XML) driver (**crdb_adoplus.dll**) does not use the table links or keys defined in the ADO .NET dataset. Instead, it will use the table linking defined in the report and will link the tables internally when the report is processed. This process will load a copy of the entire ADO .NET dataset into memory, which results in slower performance.

To work around the linking behavior, design the report to use a single table, and use an ADO .NET dataset that is also based on a single table with the same field structure. As the database engine creates its own copy of the dataset, performing a per-user query into a dataset could result in poor performance due to the memory overhead. One way to work around this limitation is use cache for the query results. Instead of creating a dataset on each hit to the aspx page, perform the query the first time. Then store the dataset in the ASP.NET cache, retrieving it from there later. This allows all report jobs to share a single dataset and

thus reduce the number of queries to the database. The C# code listing below illustrates how this could be accomplished.

```
DataSet ds;
// use the filename of the report as the cache lookup key
string datasetCacheKey = report.FilePath;
if (Cache[datasetCacheKey] != null)
{
ds = (DataSet) Cache[datasetCacheKey];
}
else
{
ds = new DataSet();
// perform query and fill dataset
...
// insert the dataset into the cache
Cache.Insert(datasetCacheKey, ds);
}
// set the dataset as the data source for the first table in the
report
report.Database.Tables[0].SetDataSource(ds);
```

The first access to the above page will not be any faster than without using caching, but on subsequent returns to the page will result in performance increase.

Table linking and number of records in the dataset also impact report performance.

- **Table linking** – Crystal Reports does not use the table links or keys defined in the ADO .NET. The report will use the table linking defined in the report and will link the tables internally when the report is processed. Indexes are not used by the ADO .NET (XML) driver. This process will load the entire ADO .NET dataset into memory, which results in slower performance. Reports that are intended to use ADO .NET datasets should be built off of single tables, stored procedures or command tables.
- **Ensure the number of records returned is reasonable** – Obviously the more records, the slower the report generation. However due to the table linking limitation and the need to create and use a copy of the dataset, the number of records used in a dataset becomes critical. Datasets of tens of thousands of records will take an inordinate amount of time to process, sometimes even leading to report or application failure.
- **DontVerifyAttachedRecordset** – This is an option set in the registry. The option is equivalent to **Verify on First Refresh** and **Verify Stored Procedure on First Refresh** options. By default, the option is set to **No**. Setting this option to **Yes** stops the Crystal Reports engine from verifying the ADO .NET dataset and this may improve performance.

  Note that explicitly not verifying the dataset may lead to malfunctioning report if the structure of the dataset is changed from what the report was designed with. For Crystal Reports 2008 look for this option in the registry at:

  HKEY_CURRENT_USER\Software\Business Objects\Suite 12.0\Crystal Reports\DatabaseOptions

For more information regarding Crystal reports and ADO .NET datasets, see the following articles:

- Crystal Reports Guide To ADO.NET
- Reporting Off ADO.NET Datasets

Coding Issues

Code will affect performance. It is recommended that before using the Crystal Reports SDK, the developer is familiar with the Crystal Reports developer help files. Examples of inefficient code leading to slow performance include incorrect database logon, incorrect printer code, incorrect export code, unhandled report objects, and more. Code issues are beyond the scope of this article. However, becoming familiar with the Crystal Reports Developer Library should be a first step.

Also, see the following resources:

- Crystal Reports For Visual Studio 2005 Walkthroughs
- Advanced use of Crystal Reports for Visual Studio .NET's feature set and object model

Performance Issues when Printing

Performance issues as well as inconsistencies in printer output may occur if different printer drivers are used to create and print your reports. Printing inconsistencies are covered in other Articles (see Related Content). However, the following may impact performance:

- two identical printers, but each one is using a different printer driver
- two different printers using the same printer driver
- two different printers using different printer drivers
- two identical printers using the same printer driver, but the printer drivers are different versions
- no printer driver installed.

Other Tips and Tricks when Using the Crystal Reports .NET SDK

Using Sessions

If a report is used in a web application, when paging through the report, printing or exporting, the default behavior of the Crystal Reports print engine is to page through a report rerunning the report on each button click. Depending on the report design this may visibly affect report performance. To resolve this issue, place the report object into session and view, page, or print/export the session object from the viewer. The following code explains the use of a Crystal report in a session:

```
Dim _ReportDocument As ReportDocument
If Not Me.IsPostBack or Session("Report") = Nothing Then
 _ReportDocument = New ReportDocument

Dim _dtAttendanceDate As DateTime
_dtAttendanceDate = CDate(AttendanceYear & "-" & AttendanceMonth & "-01")
_ReportDocument.Load(sGlobalPath & "\Report_Attendance31.rpt")

Dim _Ds As New DataSet
_Ds = LoadDataset()
For iCount As Integer = 0 To _ReportDocument.Database.Tables.Count - 1
_ReportDocument.Database.Tables(iCount).SetDataSource(_Ds)
Next
Session.Add("Report", _ReportDocument)
Else
_ReportDocument = CType(Session("Report"), ReportDocument)
 End If
Me.CrystalReportViewer1.ReportSource = _ReportDocument
```

Too Many Reports Being Processed by the Report Engine

The Crystal Reports engine is limited to processing 75 print jobs by default. Note that one Print Job is not equal to one report. Nor is a Print Job equal to a Processor license. A report with a single subreport in the report header will, at minimum take two print jobs to run. Actions such as paging, drilling to a section or group, text searches, etc. are considered to be individual Print Jobs. It does affect the performance.

More users and more subreports in a repeating section will require more jobs to run the report. This will affect performance of the report engine and may even lead to server crashes. The number of Print Jobs the engine is allowed to process can be increased by modifying the **PrintJobLimit** registry entry. For Crystal Reports 2008, the registry key is here:

HKEY_LOCAL_MACHINE\Software\Business Objects\Suite 12.0\Report Application server\InprocServer

Changing the **PrintJobLimit** will impact the hardware behavior and may adversely impact the functioning of the server. If raising the **PrintJobLimit** does not resolve the performance issue, then other solutions need to be considered. These include:

- more memory/CPU
- Web farm/Web garden

  See the article Crystal Reports 10 for .NET Feature Overview, p. 10 for more details. Also, please refer to the Microsoft article How Do I Use Crystal Reports in a Web Farm or Web Garden?.
- high-end products such as the Crystal Reports Server or Business Objects Enterprise.

For more information on choosing the correct SDK, see the following:

- Choosing the Right Business Objects SDK for Your Needs
- Choose the Right SDK for the Right Task

Related Content

The following SAP BusinessObjects notes contain further information:

Passing multiple condition If-Then-Else date range to an SQL statement

Slow performance when previewing reports in CR 10 .NET application

Crystal Reports XI .NET application loads data slowly

Slow performance when using ADO .NET dataset in .NET application

Delay when changing data source connection in CR 10 .NET application

Report loads slowly in .NET application when development computer not accessible

Additionally, please refer to the following Articles:

Optimizing Reports for the Web

Crystal Reports 2008 Component Engine Scalability

Interactivity and Reports in Web Applications

How Printer Driver Options Affect a Report

Copyright