Performance Analysis of Web Programming Languages

Himanshu Kumar
Columbia University
New York, NY USA
hk2409@columbia.edu

Abstract

Using a benchmark script that renders a table from an SQL database, the performance of several different programming languages, such as Perl, ASP.Net, and PHP, is analyzed.

Introduction

As preparation, we configured Internet Information Services (IIS) on a Windows machine, installed SQL Server 2005 and MySQL server on it, and then installed the binaries for the programming languages.

The performance analysis consists of four test cases. Each language was scripted to run those four tests and the response time is analyzed and recorded using a benchmark tool.

Related Work

Yu Song was working on the performance analysis of the languages Python, Java, Tcl and Ruby. The test cases were coordinated so as to be able to directly compare the test results. The benchmarking tool was also used in both the tests for consistency.

Configuration

Server Configuration

- HP Proliant Server
- Dual Intel Pentium 4 CPU 3.06 GHz
- 2.17 GB Ram, 3 GB Virtual Memory
- Dual Gigabit Ethernet Server Adapters
- Microsoft Windows XP Pro

Software Installation

- IIS 5.1
- .Net Framework 2.0
- Active Perl 5.8.8
- PHP 5.2.5
- Microsoft SQL Server 2005

- MySQL Server 5.0
- MySQL GUI Tools 5.0
- TortoiseSVN/Subversion 1.4.5
- Microsoft Visual Studio 2005
- Open Perl IDE 1.0.11
- MySQL connector 5.0.8.1
- MySQL ODBC 3.51 driver
- Microsoft SQL connection driver

Project Source Code

- ASP.Net C# Source
  - Contains one set of scripts. Query strings for the two databases: sql=mysql & sql=mssql
  - dbEntry.aspx: Script used to populate the databases with random data. The number of records added to the tables can be set in the script.
- Benchmark application source
  - Visual Studio 2005 Project (C#)
  - Compiled executable inside the archive: /WebPageResponseTime/bin/Release/WebPageResponseTime.exe
- Perl Source
  - Contains two sets of scripts, one each for the two databases
- PHP Source
  - Contains two sets of scripts, one each for the two databases
- Results
  - PDF
  - CSV Archive

Test Configuration

Database

Two databases were used in the testing. The database contained three tables, with 10,000 records each in two of the tables and 20,000 in the third table. An ASP.Net script was written to populate the tables with random data.

- Microsoft SQL Server 2005: The default setup was used without any changes to the security or performance
- mySQL 5.1.11: The default configuration was used, with MySQL GUI Tools 5.0 providing the front end for managing the databases.

Database Table Layout								
Student	ID	Name (100 Char)	Track (50 Char)	AdvisorID (int)				
PersonalInfo	ID	Name (100 Char)	Type (20	Age (int)	Address (200	Phone (10	WebAddress (100 Char)	Email (50

			Char)		Char)	Char)		Char)
ProfessorInfo	ID	Department (50 Char)	Office (100 Char)	Hours (50 Char)				

Test cases

Test Case	Description	# of Records
1	Print "Hello World " 10,000 times	10000
2	Student, professorInfo, personalInfo tables read from database and loaded into memory before rendering the web page	40000
3	Inner Join on Student and PersonalInfo tables and load the results into memory before rendering the database	10000
4	Table from #3, quicksort in code on "Name" column	10000

Benchmarking tool

We tested many benchmarking tools available on a website [1] that lists numerous web site testing and performance analysis tools. Most of these tools were commercial software geared towards testing performance of websites to help developers optimize their sites. We decided against using these software due to their extreme complexity and somewhat irrelevant results. The tools were not easy to configure and sometimes returned unexpected results when tests were repeated multiple times, for example, returning less than a 10 ms interval for a page of size more than 30 MB that displayed 40,000 records on screen. So we developed a small application using Microsoft Visual Studio 2005 & Visual C# to request a webpage and measure the time taken for the request. The measurement was taken as the time between the HTTP request was made and the last byte of data received. The information gathered by the application is displayed on the application window as well as stored as a log file in the executable's directory.

Testing procedure

The web server was restarted before the tests, and no extra services or applications were allowed to run during the test apart from important system services. Each test case was conducted five times using the order described below, while the test system was connected to university network using an ethernet cable to minimize the network delays.

Another test was later added to determine if the test results were linear in nature when the length of these tests were varied. The new tests added two more tests for each test case where a limit was placed on the number of records returned by the database. These limits were set to 50% and 25% of the records returned by the test scripts.

Caching

One problem faced was some type of caching taking place, probably caching by the IIS server,

when a web page is requested multiple times in sequence. To avoid this we used two strategies:

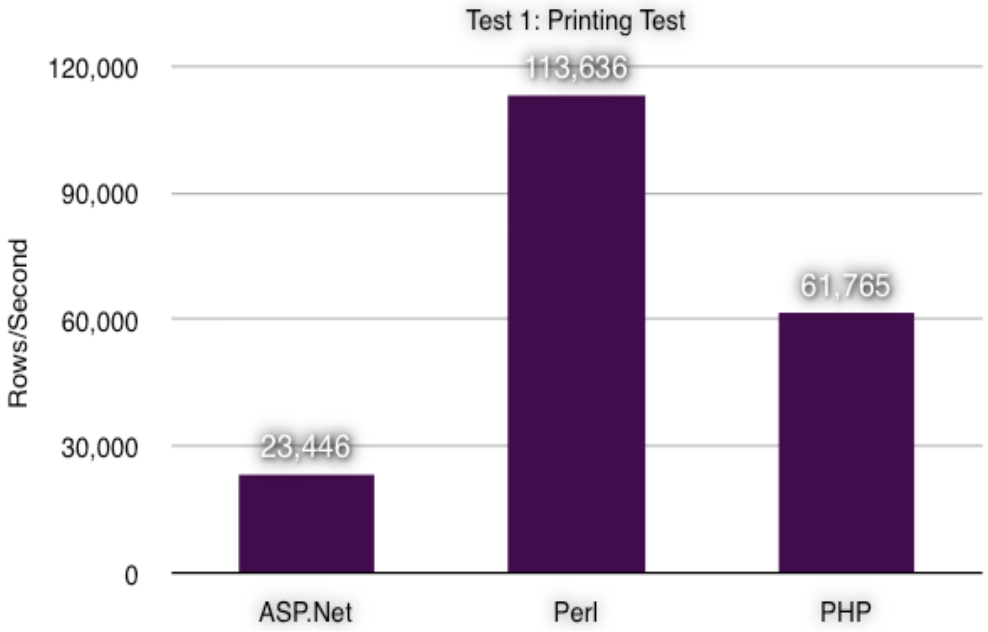
- Add a dummy parameter to the query string for the pages, and set its value to be different for each request. For example: test.aspx?n=1000, test.aspx?n=1001
- Perform the tests in a predetermined sequence to avoid caching on the database side or the web server side. The order chosen was: Test1, Test3, Test2, Test4. This order avoided having tests 3 and 4 next to each other since both tests requested the same query from the database.

Test Results

	ASP.NET	Perl	PHP
Test 1	23446	113636	61764
MSSQL			
	ASP	**PHP**	**Perl**
Test 2	27027	6976	6127
Test 3	63291	15625	14530
Test 4	54348	2664	8096
MSSQL			
	ASP	**PHP**	**Perl**
Test 2	19685	7186	17007
Test 3	46296	16611	34014
Test 4	39683	2745	12642

Performance measured in Rows/Second

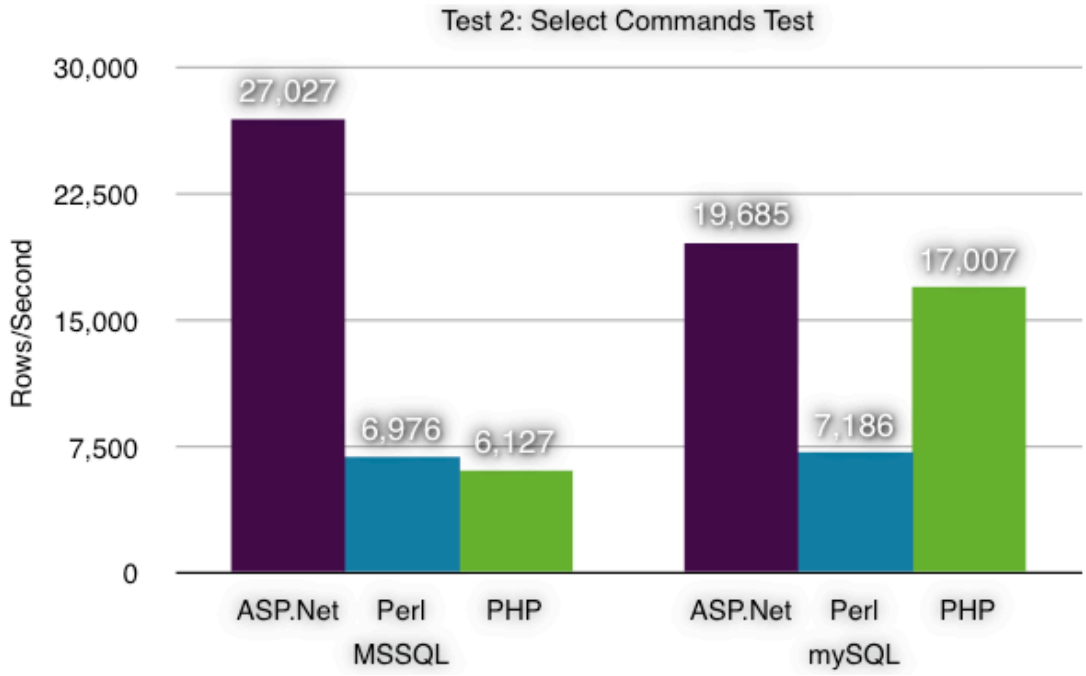
Test #1: Printing Test



Average Number of Rows displayed per second

Perl is the faster of the three languages in the printing test where the line "Hello World
" is printed onto the webpage. It had a performance advantage of 79% over ASP.Net and 46% over PHP.

Test #2: Select Commands Test



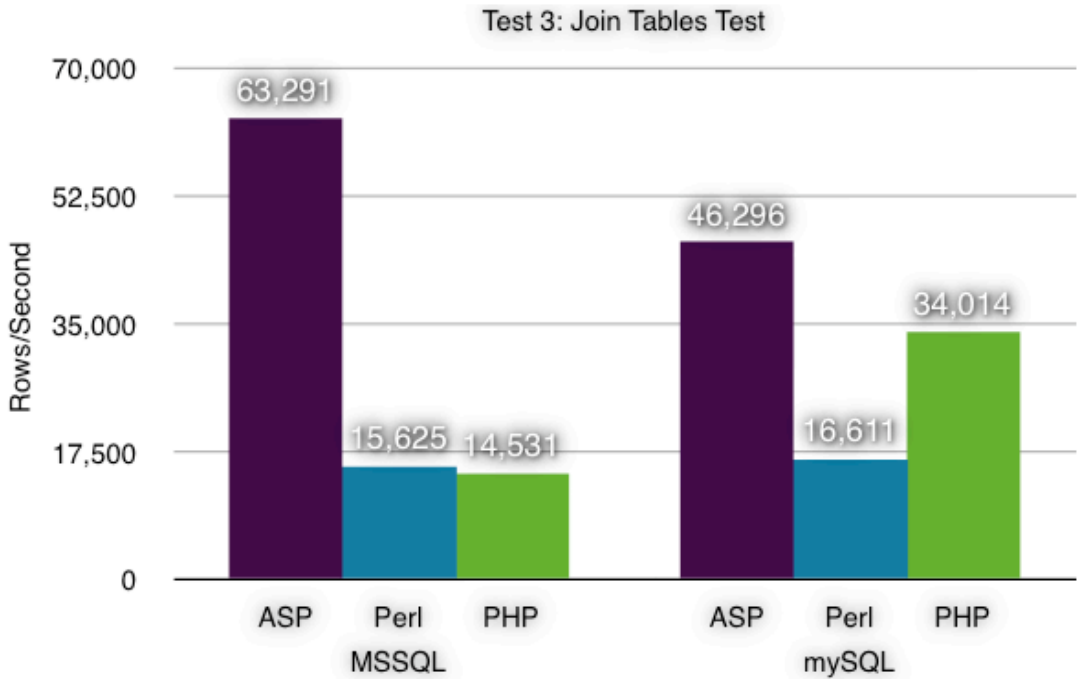
Average Number of Rows retrieved per second

ASP.Net is the faster of the three languages in the test where three SELECT commands are sent to the SQL database and 40,000 records in total are returned to the script. ASP.Net was faster than Perl by 74% and PHP by 77% when accessing MS SQL and by 63% & 13% respectively when

accessing mySQL.

PHP was faster by 63% when accessing mySQL as compared to accessing MSSQL. ASP.Net was in turn faster by 27% when accessing the Microsoft SQL database as compared to the mySQL database.

Test #3: Join Tables Test



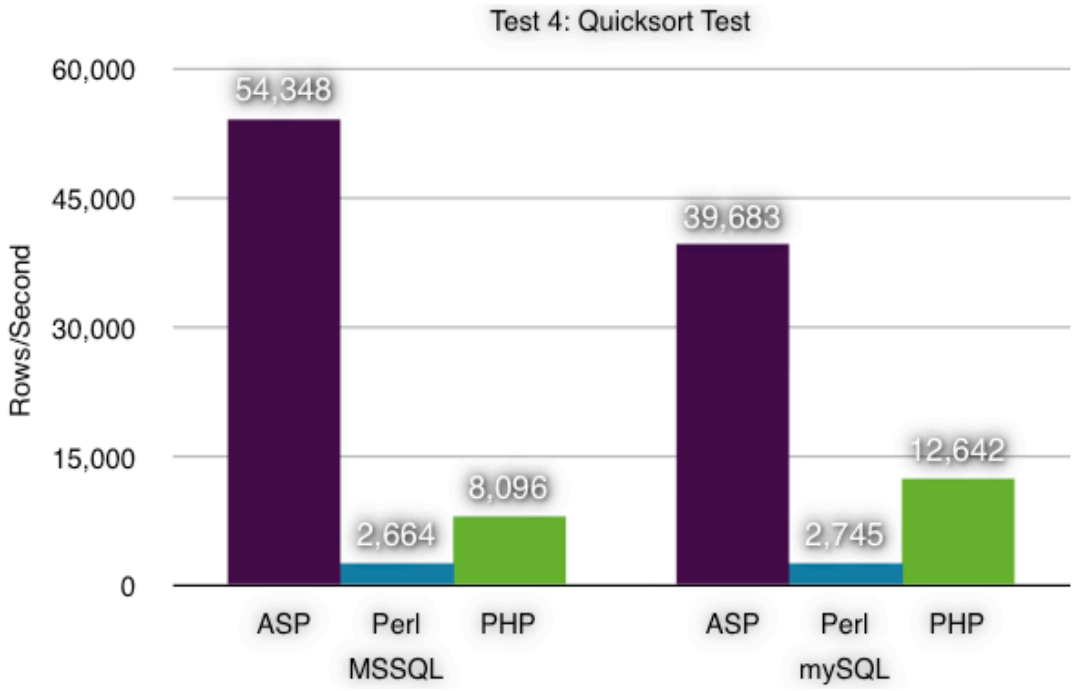
Average Number of Rows retrieved per second

ASP.Net is the faster of the three languages in this test as well where two tables are merged using a JOIN command in the SQL database and 10,000 records are returned to the script. ASP.Net was faster than Perl by 75% and PHP by 77% when accessing MS SQL and by 64% & 26% respectively when accessing mySQL.

PHP was faster by 57% when accessing mySQL as compared to accessing MSSQL. ASP.Net was in turn faster by 27% when accessing the Microsoft SQL database as compared to the mySQL database.

These results were very similar to the Test 2 above apart from a slightly reduced performance of PHP when accessing mySQL.

Test #4: Quicksort Test



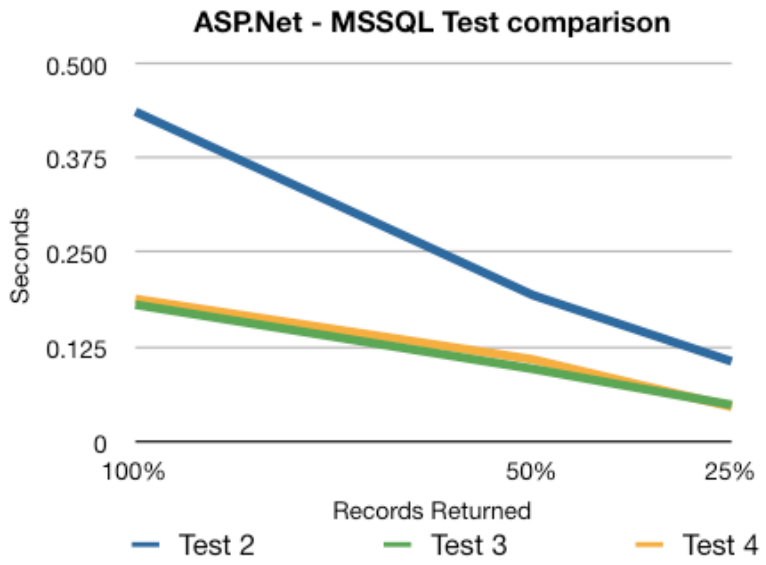
Average Number of Rows retrieved per second

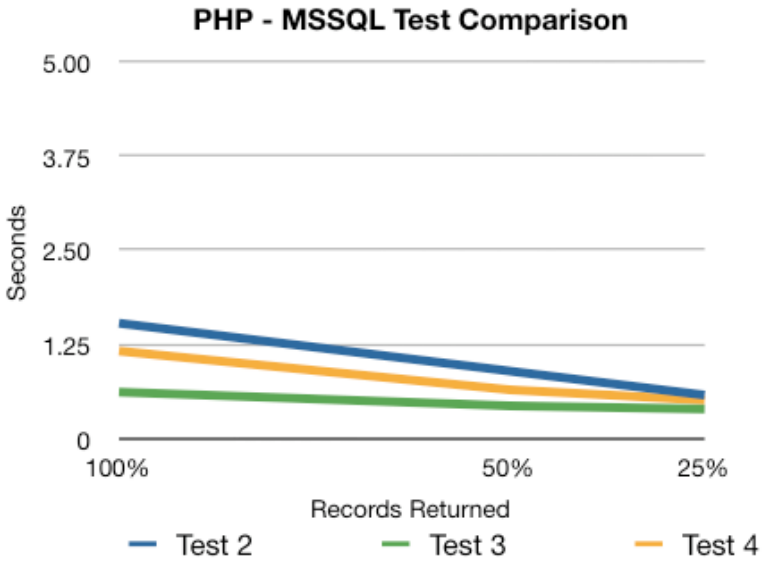
ASP.Net is the faster of the three languages in this test as well where two tables are merged using a JOIN command in the SQL database and 10,000 records are returned to the script. ASP.Net was faster than Perl by 95% and PHP by 85% when accessing MS SQL and by 93% & 68% respectively when accessing mySQL.

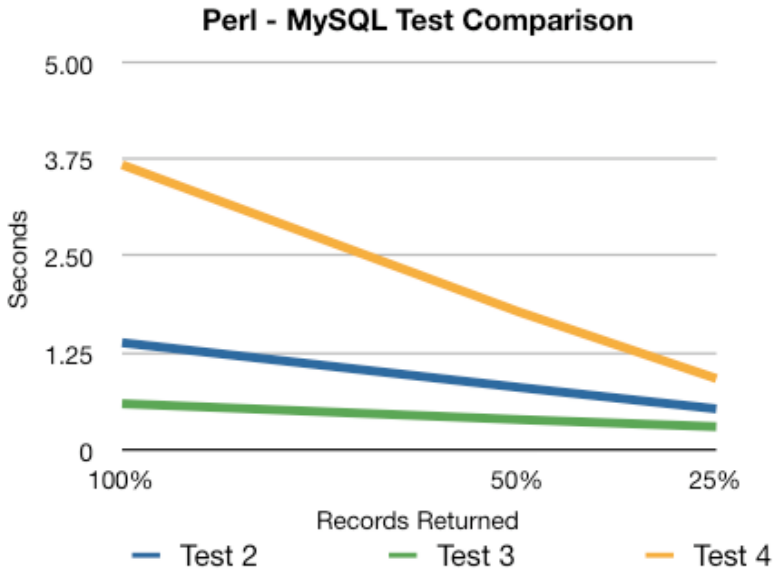
PHP was faster by 35% when accessing mySQL as compared to accessing MSSQL. ASP.Net was in turn faster by 27% when accessing the Microsoft SQL database as compared to the mySQL database, similar to test 2 & 3 above.

Verification of Test cases

Another set of measurements were taken to determine if the results of the tests were linear with varying lengths of the data-sets returned by the two databases. The graphs below show the linear nature of the test results when the tests were run for 50% and 25% of the records retrieved from the database.


ASP.Net - MSSQL Test comparison
0.500
0.375
0.250
0.125
0
Seconds
100%
50%
25%
Records Returned
Test 2
Test 3
Test 4


PHP - MSSQL Test Comparison
5.00
3.75
2.50
1.25
0
Seconds
100%
50%
25%
Records Returned
Test 2
Test 3
Test 4


Perl - MySQL Test Comparison
5.00
3.75
2.50
1.25
0
Seconds
100%
50%
25%
Records Returned
Test 2
Test 3
Test 4

Time taken in seconds to retrieve the data from the databases

Conclusion

The test results show that Perl was the fastest displaying large amount of text that was not pulled from a database, while PHP was the second placed language. In the database access and quick-sort tests, ASP.Net was the fastest with PHP coming in second place consistently. In the two SQL databases used for testing, MS SQL performed better when accessed from ASP.Net, while mySQL performed better when accessed from PHP. The performance difference in ASP.Net might be a result of the maturity of the custom connection drivers for the two databases used by ASP.Net. The difference in performance in PHP could lie in the use of an ADODB driver used by PHP to connect to the MSSQL database due to the absence of a custom MSSQL driver for PHP, while it used the mySQL connection driver for accessing the MySQL database. Perl performed equally well accessing both the databases.

Test Results Data

- PDF
- CSV Archive

Task List

- Himanshu Kumar
  - IIS installation & configuration
  - Microsoft SQL Server 2005 installation
  - MySQL database server installation
  - Database connection drivers (ODBC, ADODB, MySQL, MS SQL)
  - PHP installation
  - Perl installation
  - Version control installation (TortoiseSVN)
  - Script to fill databases with randomized data
  - Test and analyze data for PHP, Perl, and ASP. Net.
  - Develop the Benchmarking Application
- Yu Song
  - Tomcat5.5 installation
  - PostgreSQL server installation
  - Python installation
  - Ruby installation
  - Tcl installation
  - JDK installation
  - Test and analysis data on Ruby, Python, Tcl, and JSP.

References

1. Rick Hower, "Web Site Test Tools and Site Management Tools", *http://www.softwareqatest.com/qatweb1.html*
2. Michael Gossland and Associates, "Perl Tutorial Course",

http://www.gossland.com/course/index.html
3. Wikibooks, "Algorithm implementation/Sorting/Quicksort", *http://en.wikibooks.org/wiki/Transwiki:Quicksort_implementations*
4. Nik Silver, "Perl Tutorial: Start", *http://www.comp.leeds.ac.uk/Perl/start.html*
5. "Perl DBI/DBD::ODBC Tutorial", *http://www.easysoft.com/developer/languages/perl/sql_server_unix_tutorial.html*
6. "Configuring and Testing a PERL Script with Internet Information Server", *http://support.microsoft.com/kb/q150629/*