Analysis:

Step 1. Data collection.

Apparatus:
See figure 1.

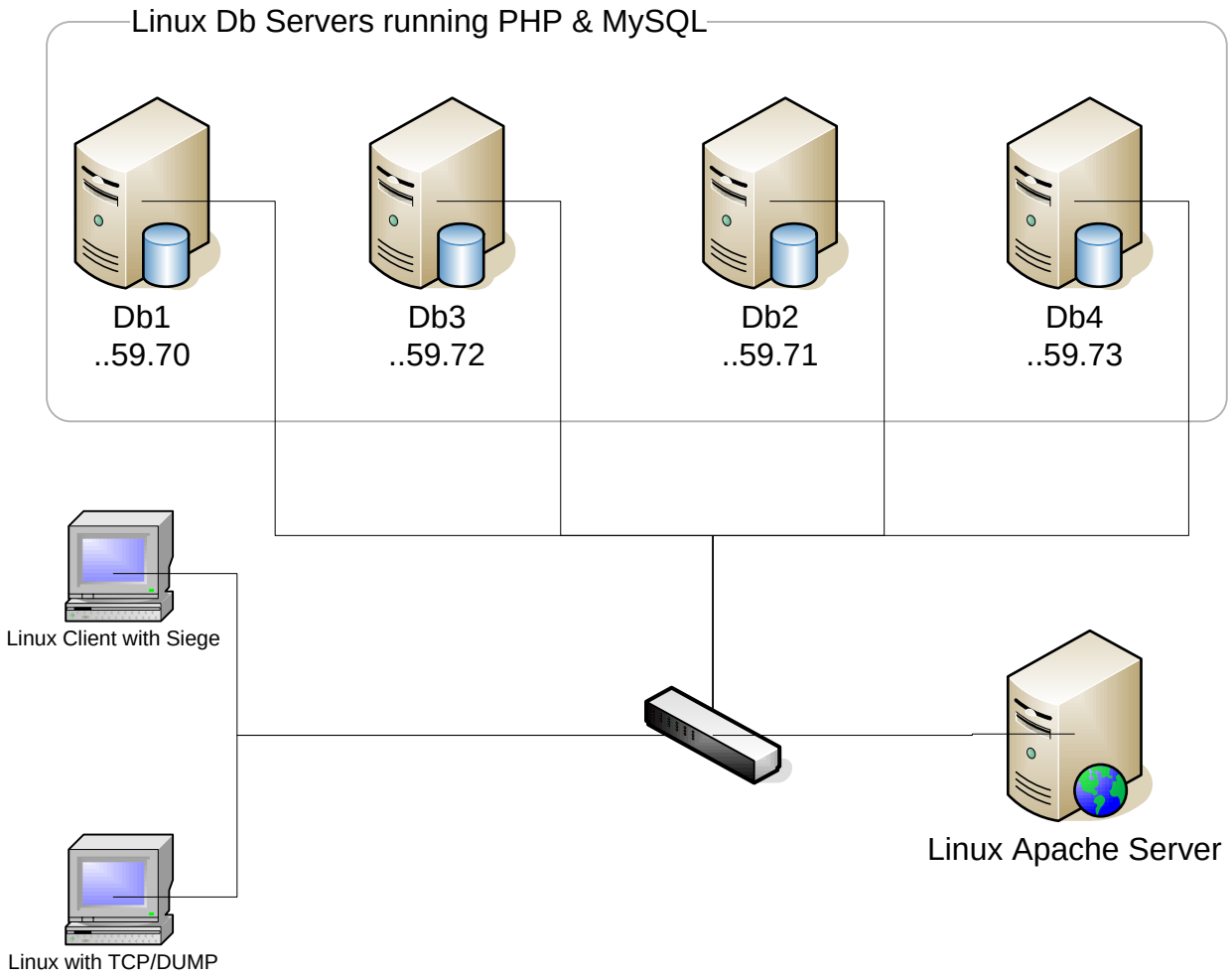


Figure 1

Description:
Siege uses the following URL's:

http://199.17.59.65/page/?function=random
http://199.17.59.65/page/?function=sequential
http://199.17.59.65/page/load.php?function=load

The Apache server would redirect based on criteria (sequential, random, load balanced.)

PHP:
Sample of the PHP code used on the Appache server:

```
<?php

//########################
// Settings
//########################

$servers = 4; //Number of Servers
//-- $test = true for testing output


//#########################
//Function app
//#########################
define("APP_DATA_FILE",
    "/tmp/application.data");

define("LOAD_DATA_FILE",
    "/tmp/mp.txt");

function application_start ()
{
    global $_APP;

    // if data file exists, load application
    //   variables
    if (file_exists(APP_DATA_FILE))
    {
        // read data file
        $file = fopen(APP_DATA_FILE, "r");
        if ($file)
        {
            $data = fread($file,
                filesize(APP_DATA_FILE));
            fclose($file);
        }

        // build application variables from
        //   data file
        $_APP = unserialize($data);
    }
}

function application_end ()
{
    global $_APP;

    // write application data to file
    $data = serialize($_APP);
    $file = fopen(APP_DATA_FILE, "w");
    if ($file)
    {
        fwrite($file, $data);
        fclose($file);
    }
}


function load_start ()
{
```

```
    global $_LOAD;

    // if data file exists, load application
    //   variables
    if (file_exists(LOAD_DATA_FILE))
    {
        // read data file
        $file = fopen(LOAD_DATA_FILE, "r");
        if ($file)
        {
            $data = fread($file,
                filesize(LOAD_DATA_FILE));
            fclose($file);
        }

        // build application variables from
        //   data file
        $_LOAD = $data;
    }
}

function load_end ()
{
    // write application data to file
    $data = "";
    $file = fopen(LOAD_DATA_FILE, "w");

    if ($file)
    {
        fwrite($file, $data);
        fclose($file);
    }
}

function reverse(&$inarray ) {
  for( $i = 0; $i < sizeof( $inarray , 1); $i++ )
   $outarray[ $i ] = $inarray[ sizeof( $inarray ) - $i - 1 ];
   $inarray = $outarray;
}


if ($function == "sequential") {
//##############################
// Sequential server selection
//##############################
       //echo "sequential";
       application_start();

       if ($_APP["serverID"]++ >= $servers+69) {
              $_APP["serverID"] = 70;
       }
       elseif ($_APP["serverID"] < 70) {
              $_APP["serverID"] = 70;
       }
       $URL = "http://199.17.59.".$_APP["serverID"] . "/?id=parameter";
       application_end();
       //echo $URL;
       header ("location: $URL");
}
elseif ($function == "load") {
//##############################
// Server selection by load
//##############################
```

```
if ($test=="true")
{
   echo "load (TeSt MoDe)<BR>==========<br>";
}
$lowestUtilization = "100";
$lowestUtilizationSvr = 1;
load_start();
//---------------
// Build Array
//---------------
$delim = "%\n \n";
$loadArray = explode($delim,$_LOAD);
reverse( $loadArray);
$i = 0;
//-- Initilization of server utilization Array
for ($count=1; $count <= $servers+1; $count++)
{
   $serverUtilization[] = "100";
}

//-- iterating through array
//--   • Finding most recent utilization values
$ellipsis = "...<br>";
while (list($IndexValue, $ElementContents) = each($loadArray))
{
   $i++;
   // -- Get Server
   $serverID =
str_replace("db","",str_replace(".","",strrev(strrchr(strrev($loadArray[$i]),".
")))));
   if ($serverID > 0)
   {
      // -- Get utilization percent
      $utilization = abs(strchr($loadArray[$i],"\t")/100);
      if ($test=="true" && $i < 20)
      {
         echo "Server($serverID) = $utilization<br>";
      }
      elseif ($test =="true")
      {
         echo "$ellipsis";
         $ellipsis = "";
      }
      if ($serverUtilization[$serverID] == "100")
      {
         $serverUtilization[$serverID] = $utilization;
      }
   }
}


//-- Saving utilization in Application Session Variables and finding lowest
utilization
application_start();
for ($count=1; $count < $servers+1; $count++)
{
   if ($serverUtilization[$count] == "100" && $_APP[$count] > 0)
   {
      $serverUtilization[$count] = $_APP[$count];
   }
   $_APP[$count] = $serverUtilization[$count];
   if ($test=="true")
```

```
   {
      echo "Application Session-->SERVER($count) = $_APP[$count]<BR>";
   }
   // -- Determining lowest utilization
   if ($lowestUtilization > $serverUtilization[$count])
   {
      $lowestUtilization = $serverUtilization[$count];
      $lowestUtilizationSvr = $count;
   }
}

if ($test=="true")
{
   echo "LOWEST UTILIZATION is SERVER $lowestUtilizationSvr @
$lowestUtilization Utilization<br>";
}
application_end();
$lowestUtilizationSvr += 69;
$URL = "http://199.17.59.".$lowestUtilizationSvr . "/?id=parameter";

//-- Check value of $i > 100 then flush load file
if ($i > 100)
{
   load_end();
}
if ($test=="true")
{
   echo "Redirect to ---> $URL";
}
else
{
   header ("location: $URL");
}


}
else {
//##############################
// Random server selection
//##############################
//echo "random";
       $r = rand(0,$servers-1);
       $URL = "http://199.17.59.7$r" . "/?id=parameter";
       //echo $URL;
       header ("location: $URL");
}



application_end();

exit;

?>
```

Output test screen on browser under no load (4 servers):

```
load (TeSt MoDe)
==========
Server(1) = 0.04
Server(2) = 0.02
Server(3) = 0.02
Server(4) = 0
Server(1) = 0
Server(2) = 0
Server(3) = 0.03
Server(4) = 0.01
Server(1) = 0.01
Server(2) = 0.01
Server(3) = 0.01
Server(4) = 0.01
Server(1) = 0
Server(2) = 0.01
Server(3) = 0.01
Server(4) = 0.04
Server(1) = 0.01
Server(2) = 0.04
Server(3) = 0.01
...
Application Session-->SERVER(1) = 0.04
Application Session-->SERVER(2) = 0.02
Application Session-->SERVER(3) = 0.02
Application Session-->SERVER(4) = 0
LOWEST UTILIZATION is SERVER 4 @ 0 Utilization
Redirect to ---> http://199.17.59.73/?id=parameter
```

Output test screen on browser under no load (2 servers):

```
load (TeSt MoDe)
==========
Server(3) = 0.01
Server(4) = 0.01
Server(1) = 0.04
Server(2) = 0
Server(3) = 0.03
Server(4) = 0.01
Server(1) = 0
Server(2) = 0.01
Server(3) = 0.01
Server(4) = 0
Server(1) = 0.01
Server(2) = 0.01
Server(3) = 0.01
Server(4) = 0.04
Server(1) = 0.01
Server(2) = 0.01
Server(3) = 0.01
Server(4) = 0.02
Server(1) = 0.04
...
Application Session-->SERVER(1) = 0.04
Application Session-->SERVER(2) = 0
```

```
LOWEST UTILIZATION is SERVER 2 @ 0 Utilization
Redirect to ---> http://199.17.59.71/?id=parameter
```

Output test screen on browser under full load (8 iterations of 400 concurrent sessions using 4 servers):

```
load (TeSt MoDe)
==========
Server(3) = 0
Server(4) = 0.01
Server(3) = 0.05
Server(2) = 0.69
Server(1) = 0.97
Server(4) = 0.01
Server(3) = 0.01
Server(2) = 0.49
Server(4) = 0.04
Server(1) = 0.01
Server(3) = 0.01
Server(2) = 0.01
Server(4) = 0.01
Server(1) = 0.04
Server(3) = 0
Server(2) = 0.04
Server(4) = 0
Server(1) = 0
Server(3) = 0.02
...
Application Session-->SERVER(1) = 0.97
Application Session-->SERVER(2) = 0.69
LOWEST UTILIZATION is SERVER 2 @ 0.69 Utilization
Redirect to ---> http://199.17.59.71/?id=parameter
```

Output test screen on browser under full load (8 iterations of 400 concurrent sessions using 2 servers):

```
load (TeSt MoDe)
==========
Server(3) = 0
Server(2) = 0.01
Server(4) = 0.01
Server(3) = 0.02
Server(2) = 0.01
Server(1) = 0.99
Server(4) = 0
Server(3) = 0
Server(2) = 0.01
Application Session-->SERVER(1) = 0.99
Application Session-->SERVER(2) = 0.01
Application Session-->SERVER(3) = 0
Application Session-->SERVER(4) = 0.01
LOWEST UTILIZATION is SERVER 3 @ 0 Utilization
Redirect to ---> http://199.17.59.72/?id=parameter
```

The following grid was used as a template for data collection:

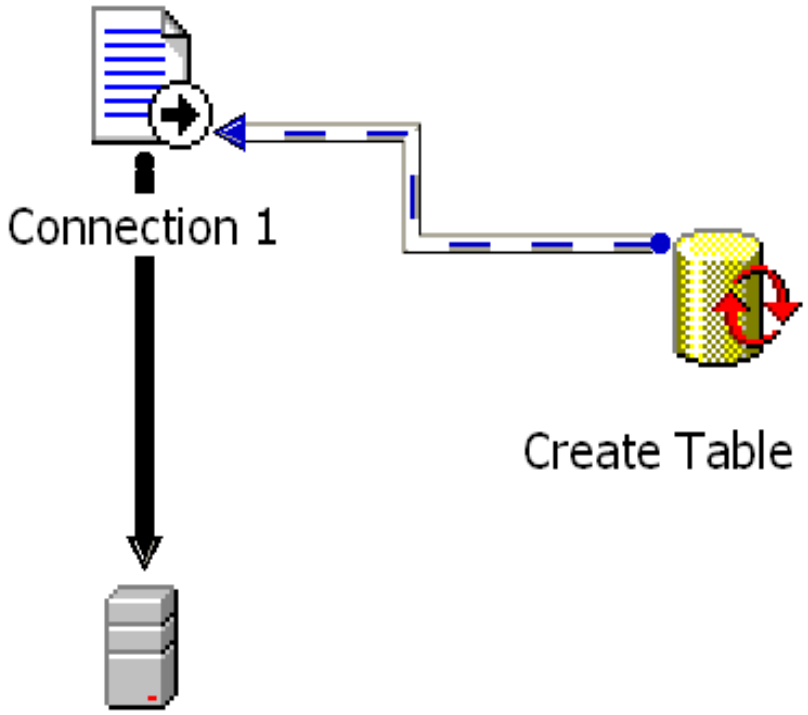
NOTE: Data grid can also be found in Appendix A (Macro Analysis)

Client (A) client 1 (B) zeus	Query Distribution Type	Sequential Iterations	Server Nodes	Concurrent Client Sessions	Average Delay (ms)	Throughput	Packet Intensity	Total Process Time (s)	Peak Process Count
A	Sequential	8	1	50	2.07193316	92758.467	241.321	55	280
A	Sequential	8	2	50	0.74688173	191275.131	669.450	27	72
A	Sequential	8	4	50	0.39466387	312233.329	1266.901	16.42	65
A	Sequential	8	1	100	3.88490715	44360.966	128.703	> 200	-
A	Sequential	8	2	100	0.71031160	197973.048	703.916	55.52	193
A	Sequential	8	4	100	0.37551237	361269.535	1331.514	30.01	90
A	Sequential	8	1	200	4.80601741	17878.602	104.036	> 736	> 285
A	Sequential	8	2	200	5.19456850	21524.983	96.254	> 787	285
A	Sequential	8	4	200	0.34005095	330987.386	1470.368	75.52	180
A	Sequential	8	1	400	4.20020566	23200.616	119.042	> 300	233
B	Sequential	8	1	400	21.56025828	5609.872	23.191	> 300	233
A	Sequential	8	2	400	0.62360851	216110.392	801.785	456.99	283
B	Sequential	8	2	400	11.54540789	8978.663	43.307	456.99	283
A	Sequential	8	4	400	0.31362455	385330.204	1594.263	336.6	230
B	Sequential	8	4	400	0.71562455	166563.402	698.690	336.6	230
A	Random	8	2	50	0.71621023	167432.264	698.119	20.19	58
A	Random	8	4	50	0.47683332	275472.700	1048.584	17.09	62
A	Random	8	2	100	0.63998721	202004.413	781.266	60.1	200
A	Random	8	4	100	0.44903326	312168.375	1113.503	32.01	106
A	Random	8	2	200	13.61430900	8529.467	36.726	>2244	230
A	Random	8	4	200	0.89513973	157894.511	558.572	58.57	200
A	Random	8	2	400	-	-	-	>500	280
B	Random	8	2	400	7.02728106	14320.478	71.151	>500	280
A	Random	8	4	400	-	-	-	336.6	230
B	Random	8	4	400	5.76863794	19649.033	86.676	336.6	230
A	Load Balanced	8	2	50	0.17195826	252105.315	2907.682	122.09	128
A	Load Balanced	8	4	50	0.08090560	522526.965	6180.042	50.96	92
A	Load Balanced	8	2	100	0.13790886	318776.122	3625.583	50.96	92

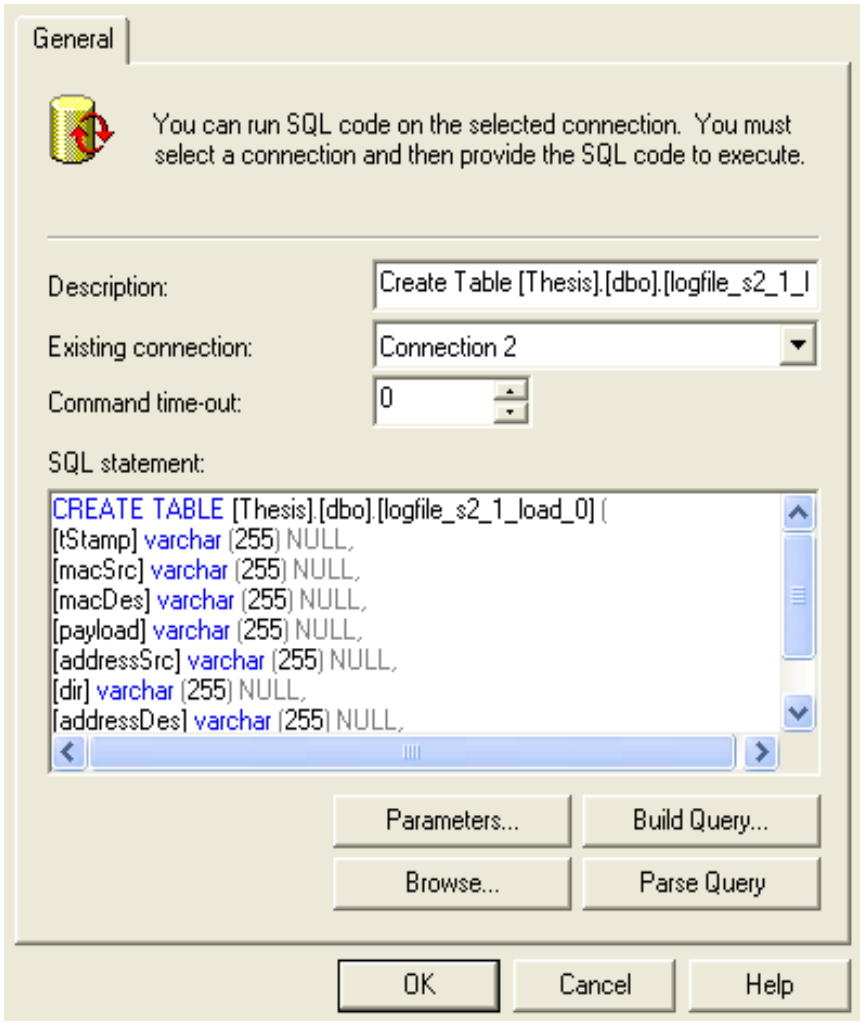
A	Load Balanced	8	4	100	0.08812921	484603.770	5673.488	33.98	103	test 1
					0.07656717	556347.453	6530.214	23.06	122	test 2
A	Load Balanced	8	2	200	0.10743538	424712.763	4653.961	41.64	163	
A	Load Balanced	8	4	200	0.05969465	724683.596	8375.961	19.28	138	
A	Load Balanced	8	2	400	0.26592366	170338.792	1880.239	60.95	209	test 1
					0.24486395	211604.990	2041.950	43.03	211	test 2
					0.13944143	358677.363	3585.735	23.34	160	test 3
					0.43814796	99091.412	1141.167	90.04	277	test 4
A	Load Balanced	8	4	400	0.09072802	474901.359	5510.977	24.29	163	test 1
					0.12549306	346656.873	3984.284	30.25	188	test 2
					0.60850789	87940.314	821.682	108.09	257	test 3
					0.12308939	350594.854	4062.089	27.88	191	test 4
A	Load Balanced	8	2	50	0.17195826	252105.315	2907.682	122.09	128	
A	Load Balanced	8	4	50	0.08090560	522526.965	6180.042	50.96	92	
A	Load Balanced	8	2	100	0.13790886	318776.122	3625.583	50.96	92	

Step 2:Import TCPDUMP into MS SQL Server.

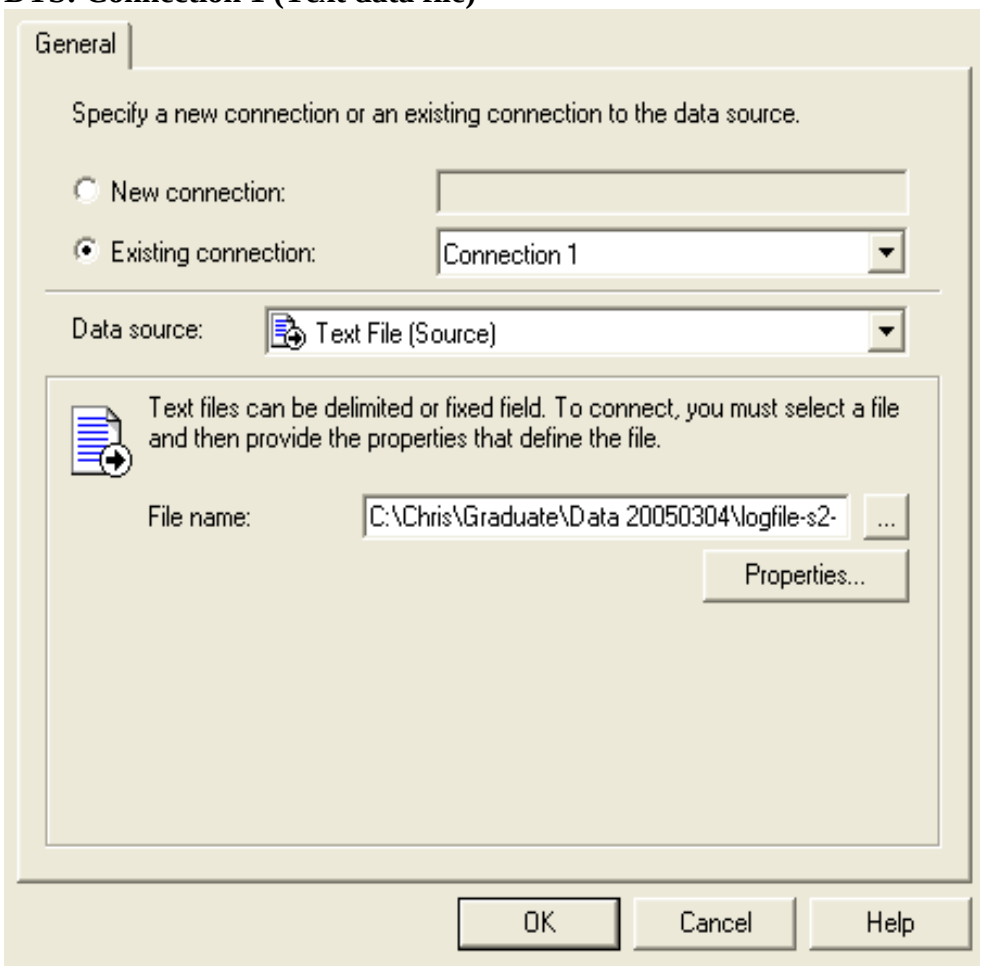
DTS Package



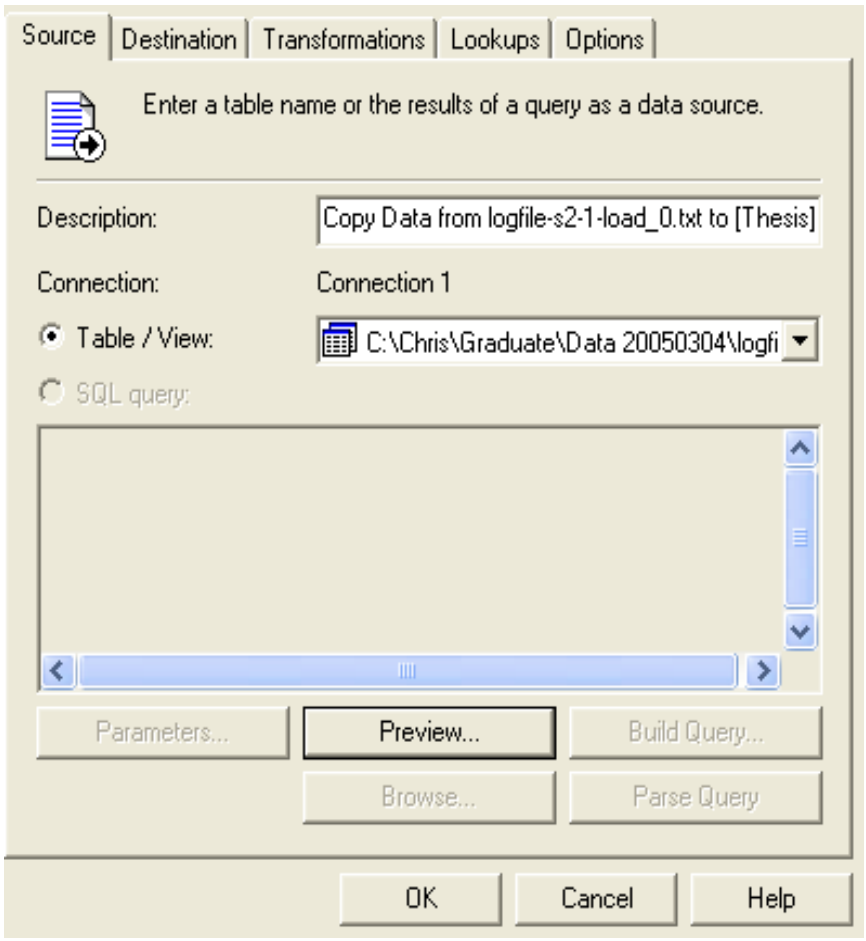
DTS: Create Table

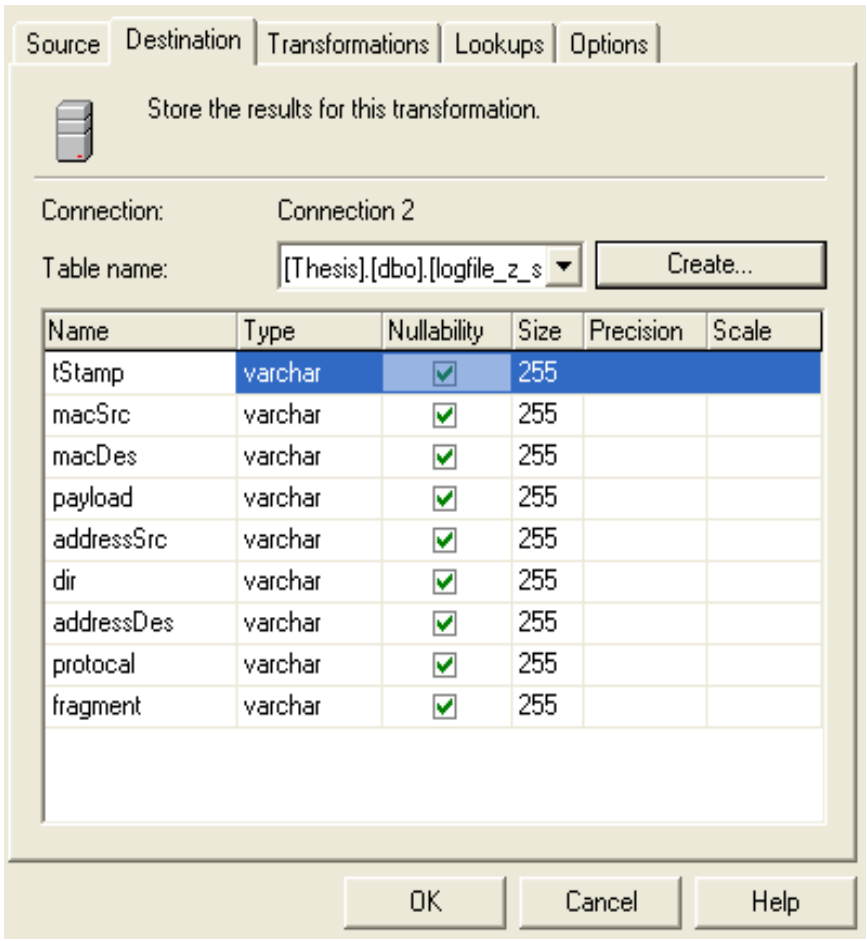

General
You can run SQL code on the selected connection. You must select a connection and then provide the SQL code to execute.
Description:
Create Table [Thesis].[dbo].[logfile_s2_1_l
Existing connection:
Connection 2
Command time-out:
0
SQL statement:
CREATE TABLE [Thesis].[dbo].[logfile_s2_1_load_0] (
[tStamp] varchar (255) NULL,
[macSrc] varchar (255) NULL,
[macDes] varchar (255) NULL,
[payload] varchar (255) NULL,
[addressSrc] varchar (255) NULL,
[dir] varchar (255) NULL,
[addressDes] varchar (255) NULL,
Parameters...
Build Query...
Browse...
Parse Query
OK
Cancel
Help

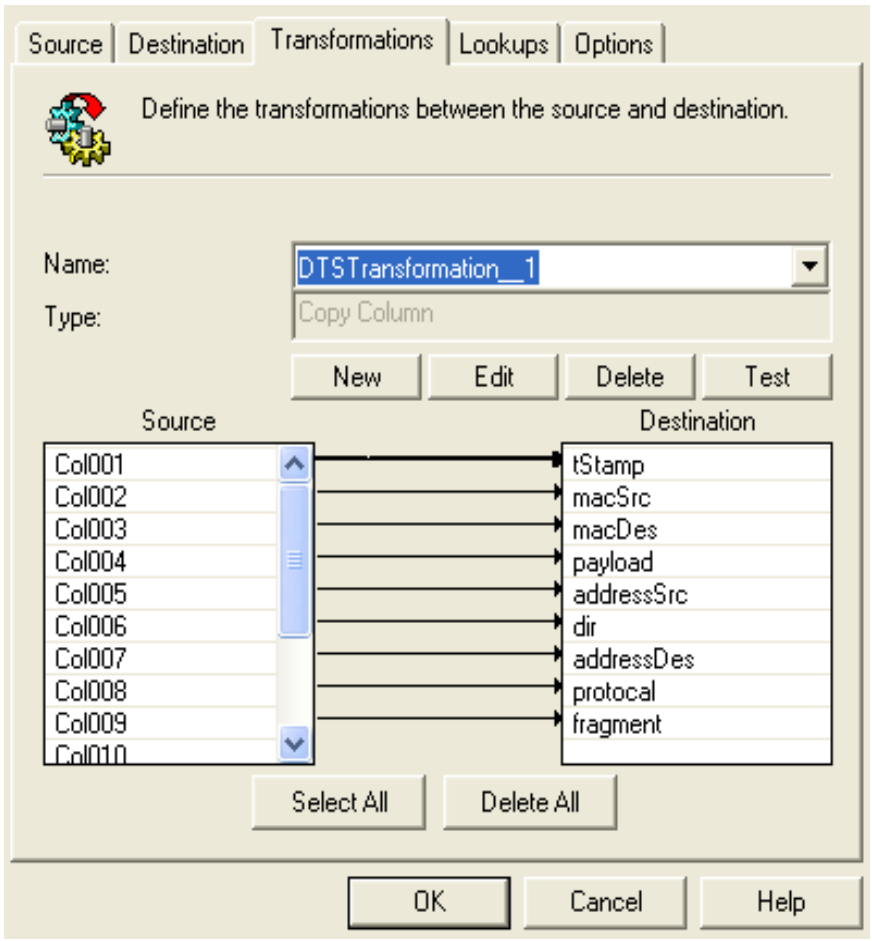
DTS: Connection 1 (Text data file)



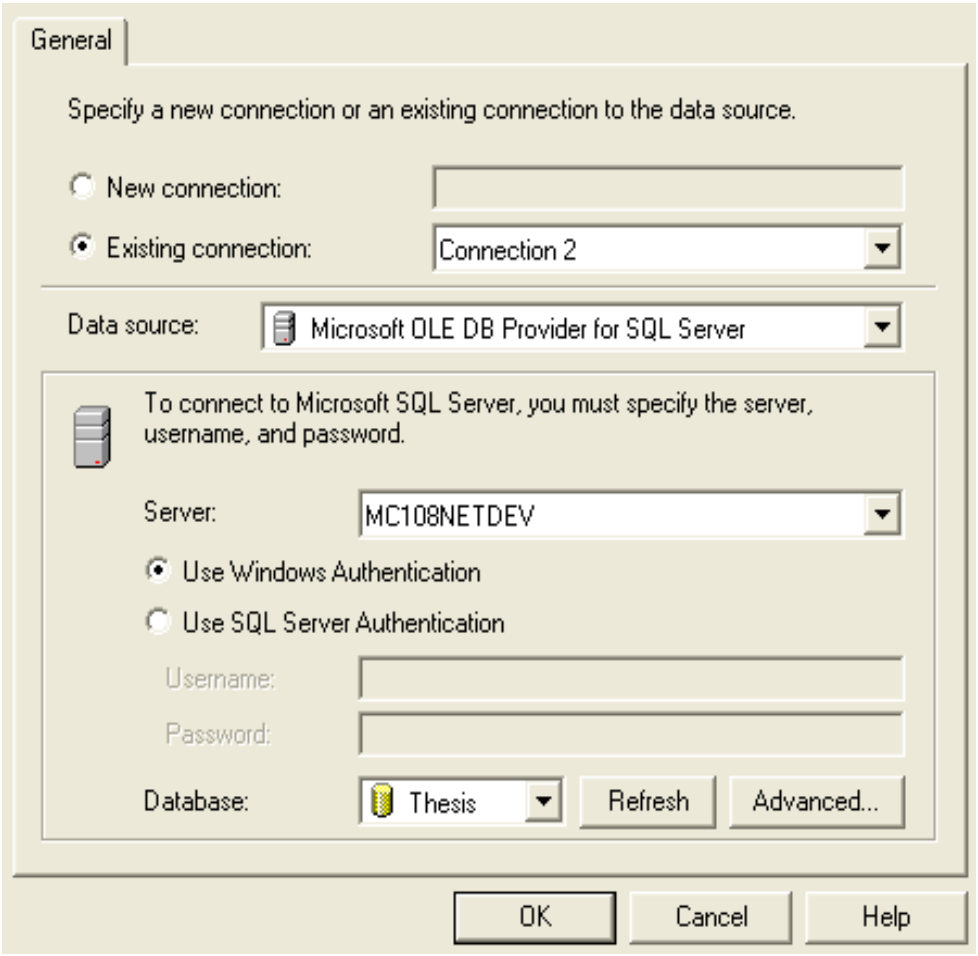
DTS: Transformation




Source
Destination
Transformations
Lookups
Options
Store the results for this transformation.
Connection:
Connection 2
Table name:
[Thesis].[dbo].[logfile_z_s
Create...
Name
Type
Nullability
Size
Precision
Scale
tStamp
varchar
255
macSrc
varchar
255
macDes
varchar
255
payload
varchar
255
addressSrc
varchar
255
dir
varchar
255
addressDes
varchar
255
protocal
varchar
255
fragment
varchar
255
OK
Cancel
Help

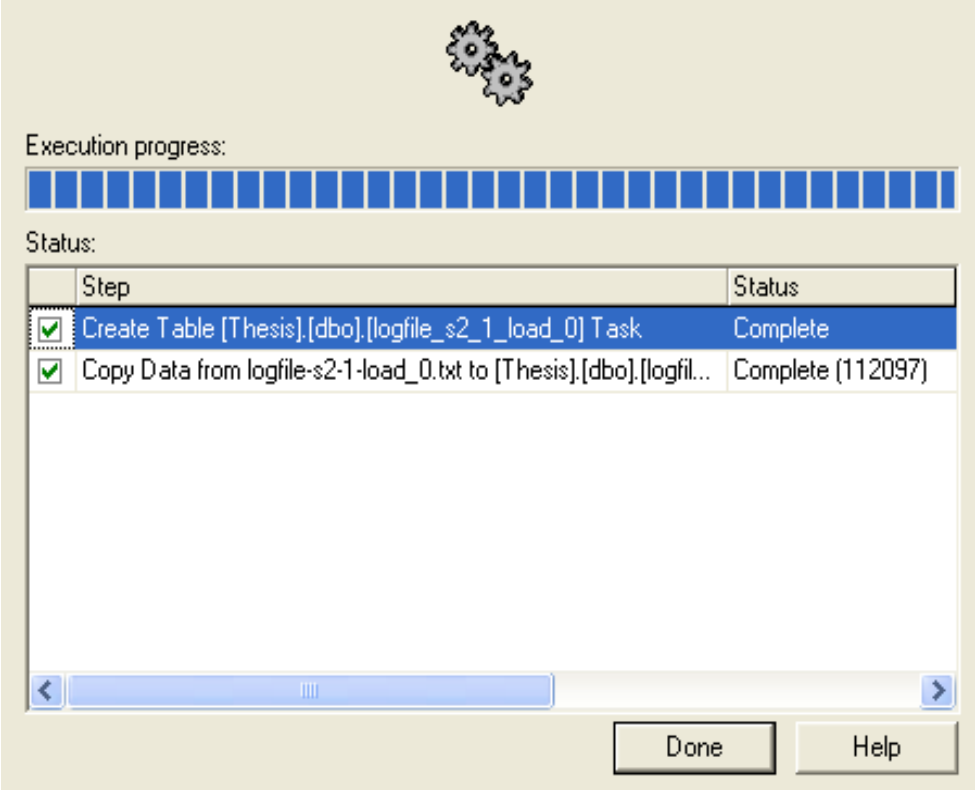

Source
Destination
Transformations
Lookups
Options
Define the transformations between the source and destination.
Name:
DTSTransformation__1
Type:
Copy Column
New
Edit
Delete
Test
Source
Destination
Col001
Col002
Col003
Col004
Col005
Col006
Col007
Col008
Col009
Col010
tStamp
macSrc
macDes
payload
addressSrc
dir
addressDes
protocal
fragment
Select All
Delete All
OK
Cancel
Help

DTS: Destination




```
-- SQL used to create a data table for each test (NOTE Table name
changes to match log file naming convention.)
CREATE TABLE [Thesis].[dbo].[logfile_s2_1_load_0] (
[tStamp] varchar (255) NULL,
[macSrc] varchar (255) NULL,
[macDes] varchar (255) NULL,
[payload] varchar (255) NULL,
[addressSrc] varchar (255) NULL,
[dir] varchar (255) NULL,
[addressDes] varchar (255) NULL,
[protocal] varchar (255) NULL,
[fragment] varchar (255) NULL
)
```

DST: Execution



Step 3: Create Stored procedures and views used in determining Average Delay, Throughput and Packet Intensity. One set of views, and stored procedure is created for each data set.

i. Drop standard view if already created

```
if exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[logfile_s2_1_loadView]') and OBJECTPROPERTY(id,
N'IsView') = 1)
drop view [dbo].[logfile_s2_1_loadView]
GO
```

ii. Create standard view

```
CREATE VIEW dbo.logfile_s2_1_loadView
AS
SELECT TOP 100 PERCENT
         tStamp,
         CAST(REPLACE(payload, ':', '') AS int) AS payload,
         addressSrc,
         addressDes,

   REPLACE(RIGHT(addressDes,CHARINDEX('.',REVERSE(addressDes))-1
),':','') as temp,
         CASE RIGHT(addressSrc, 3)
```

```
                WHEN '.80' THEN
REPLACE(RIGHT(addressDes,CHARINDEX('.',REVERSE(addressDes))-1
),':','')
                ELSE
RIGHT(addressSrc,CHARINDEX('.',REVERSE(addressSrc))-1 )
                 END
             AS portSrc,
         CAST(LEFT(tStamp, 2) AS decimal(10, 6)) * 3600 +
               CAST(SUBSTRING(tStamp, 4, 2) AS decimal(10, 6)) * 60
+
               CAST(SUBSTRING(tStamp, 7, 9) AS decimal(10, 6)) AS t
FROM         dbo.logfile_s2_1_load
ORDER BY portSrc, tStamp
GO
```

iii. Drop throughput view if already created

```
if exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[logfile_s2_1_loadView_tp]') and
OBJECTPROPERTY(id, N'IsView') = 1)
drop view [dbo].[logfile_s2_1_loadView_tp]
GO
```

iv. Create throughput view

```
CREATE VIEW dbo.logfile_s2_1_loadView_tp
AS
SELECT     TOP 100 PERCENT portSrc, MAX(t) - MIN(t) AS t,
SUM(payload) AS payload
FROM         dbo.logfile_s2_1_loadView
GROUP BY portSrc
ORDER BY portSrc
GO
```

v. Drop stored procedure if already created

```
if exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[lf_s2_1_load_stat]') and OBJECTPROPERTY(id,
N'IsProcedure') = 1)
drop procedure [dbo].[lf_s2_1_load_stat]
GO
```

vi. Create statistics stored procedure

```
CREATE PROCEDURE [dbo].[lf_s2_1_load_stat] AS
/*
Convention
==========
                                logfile s2 1 load
                                ------- - -- - ---
                                   |    |  | |  |
  logfile - initial filename -----/    |  | |  |
                                       |  | |  |
  z - data from zeus client -----------/  | |  |
                                          | |  |
  s# - 1 = using 1 server                 | |  |
       2 = using 2 servers                | |  |
       4 = using 4 servers ---------------/ |  |
                                            |  |
  # - 1 = 50  concurrent sessions           |  |
      2 = 100 concurrent sessions           |  |
      3 = 200 concurrent sessions           |  |
      4 = 400 concurrent sessions ----------/  |
                                               |
 Seq - seq = sequence                          |
       ran = random ---------------------------/
*/

--#################
--logfile_s2_1_load
--#################

--  AVERAGE DELAY
SELECT
   (MAX(t) - MIN(t))/count(portSrc) / 2 AS Average_Delay
FROM
   dbo.logfile_s2_1_loadView

--  THROUGHPUT
SELECT
   SUM(payload)/sum(t) AS Throughput
FROM
   dbo.logfile_s2_1_loadView_tp

--  PACKET INTENSITY
SELECT
   count(portSrc)/(MAX(t) - MIN(t)) AS Packet_Intensity
FROM
   dbo.logfile_s2_1_loadView
GO
```

vii. Execute stored procedure

```
lf_s2_1_load_stat
```

viii. Results

```
Average_Delay
---------------------------------
.0001662230122126350

(1 row(s) affected)

Throughput
-------------------------------------
298088.1711530463984497266

(1 row(s) affected)

Packet_Intensity
-------------------------------------
3008.0070944712852680828

(1 row(s) affected)
```

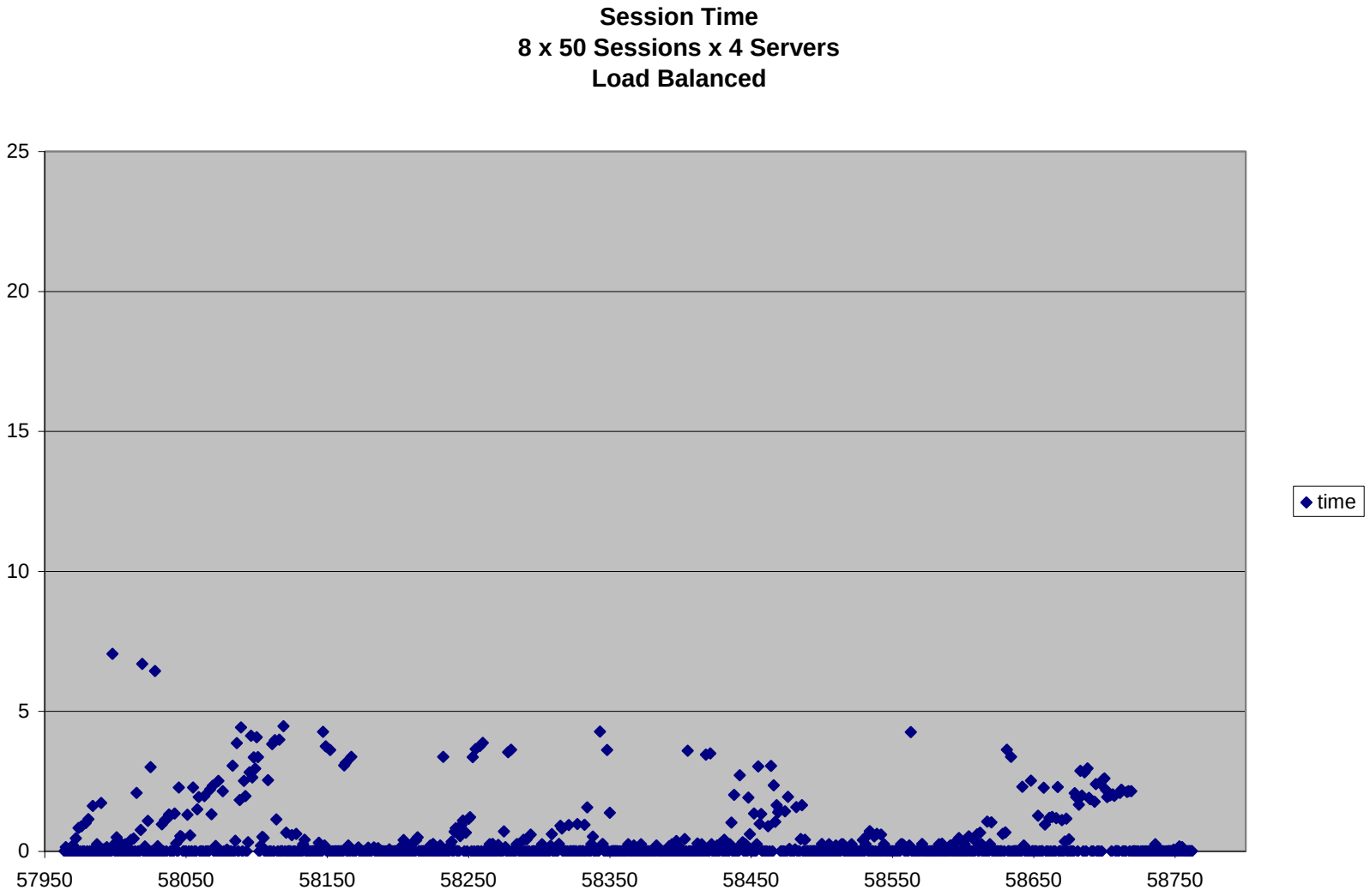
Step 5:load data into Excel

i. Sample Excel column of load balanced data from 4 servers using 8 iterations of 50 concurrent sessions.

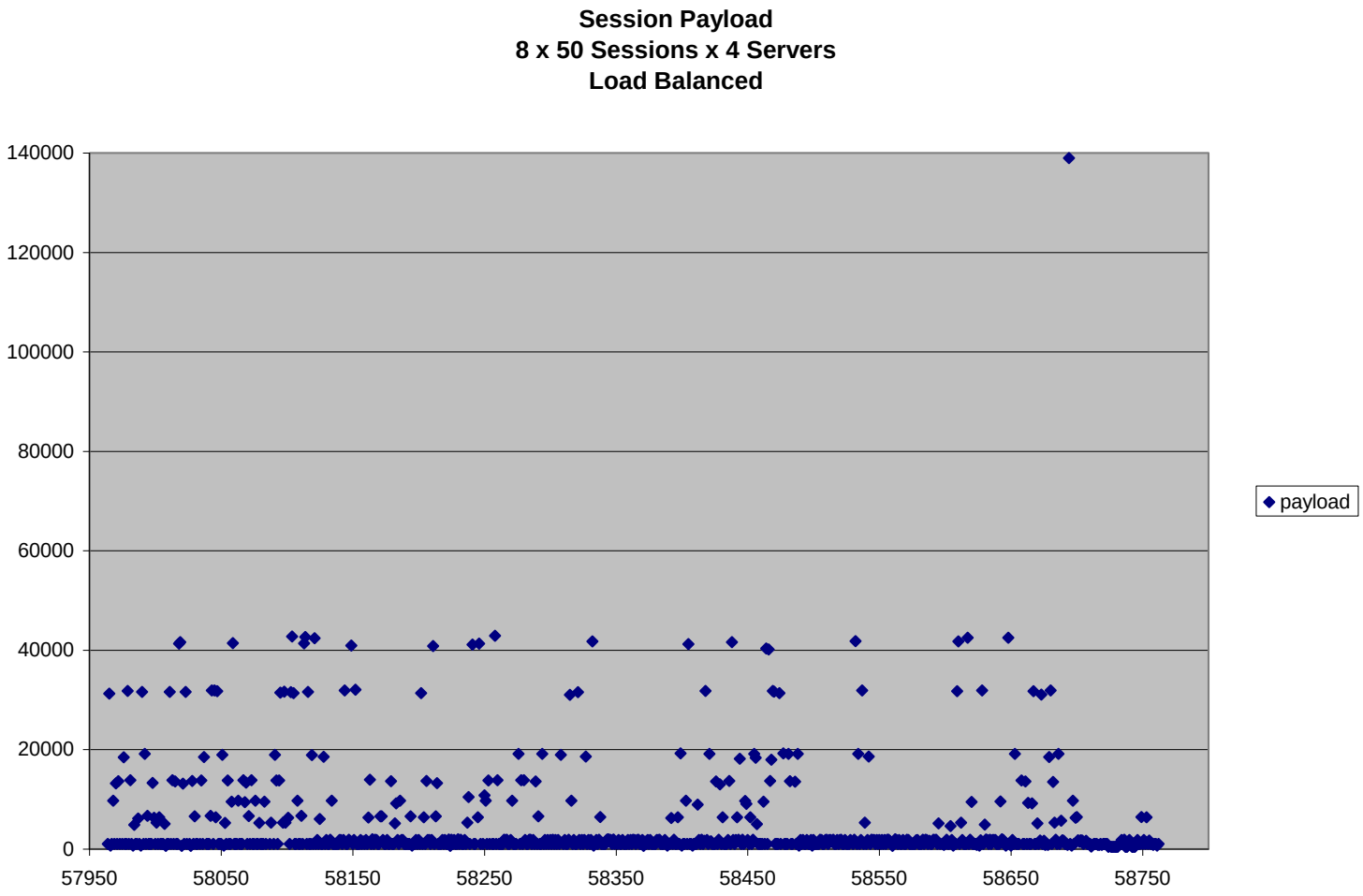
NOTE: Excel data is derived based on the throughput view in step 3 above.

portSeq	time	payload
57964	0.010289	1049
57965	0.147174	31238
57966	0.00319	696
57967	0.002593	1049
57968	0.092772	9705
57969	0.002683	1049
57970	0.217409	13198
57971	0.00261	1049
57972	0.468031	13664
57973	0.002636	1049
57974	0.830308	197832
57975	0.002736	1049
57976	0.895952	18450
57977	0.002433	1049
57978	0.002519	1049
57979	1.011154	31815
.	.	.
.	.	.
.	.	.

ii. Sample Excel graph of above data set noting session time.



iii. Sample Excel graph of above data set noting session payload.



Appendix A

Data Grid (Macro Analysis)

Client (A) client 1 (B) zeus	Query Distribution Type	Sequential Iterations	Server Nodes	Concurrent Client Sessions	Average Delay (ms)	Throughput	Packet Intensity	Total Process Time (s)	Peak Process Count
A	Sequential	8	1	50	2.07193316	92758.467	241.321	55	280
A	Sequential	8	2	50	0.74688173	191275.131	669.450	27	72
A	Sequential	8	4	50	0.39466387	312233.329	1266.901	16.42	65
A	Sequential	8	1	100	3.88490715	44360.966	128.703	> 200	-
A	Sequential	8	2	100	0.71031160	197973.048	703.916	55.52	193
A	Sequential	8	4	100	0.37551237	361269.535	1331.514	30.01	90
A	Sequential	8	1	200	4.80601741	17878.602	104.036	> 736	> 285
A	Sequential	8	2	200	5.19456850	21524.983	96.254	> 787	285
A	Sequential	8	4	200	0.34005095	330987.386	1470.368	75.52	180
A	Sequential	8	1	400	4.20020566	23200.616	119.042	> 300	233
B	Sequential	8	1	400	21.56025828	5609.872	23.191	> 300	233
A	Sequential	8	2	400	0.62360851	216110.392	801.785	456.99	283
B	Sequential	8	2	400	11.54540789	8978.663	43.307	456.99	283
A	Sequential	8	4	400	0.31362455	385330.204	1594.263	336.6	230
B	Sequential	8	4	400	0.71562455	166563.402	698.690	336.6	230
A	Random	8	2	50	0.71621023	167432.264	698.119	20.19	58
A	Random	8	4	50	0.47683332	275472.700	1048.584	17.09	62
A	Random	8	2	100	0.63998721	202004.413	781.266	60.1	200
A	Random	8	4	100	0.44903326	312168.375	1113.503	32.01	106
A	Random	8	2	200	13.61430900	8529.467	36.726	>2244	230
A	Random	8	4	200	0.89513973	157894.511	558.572	58.57	200
A	Random	8	2	400	-	-	-	>500	280
B	Random	8	2	400	7.02728106	14320.478	71.151	>500	280

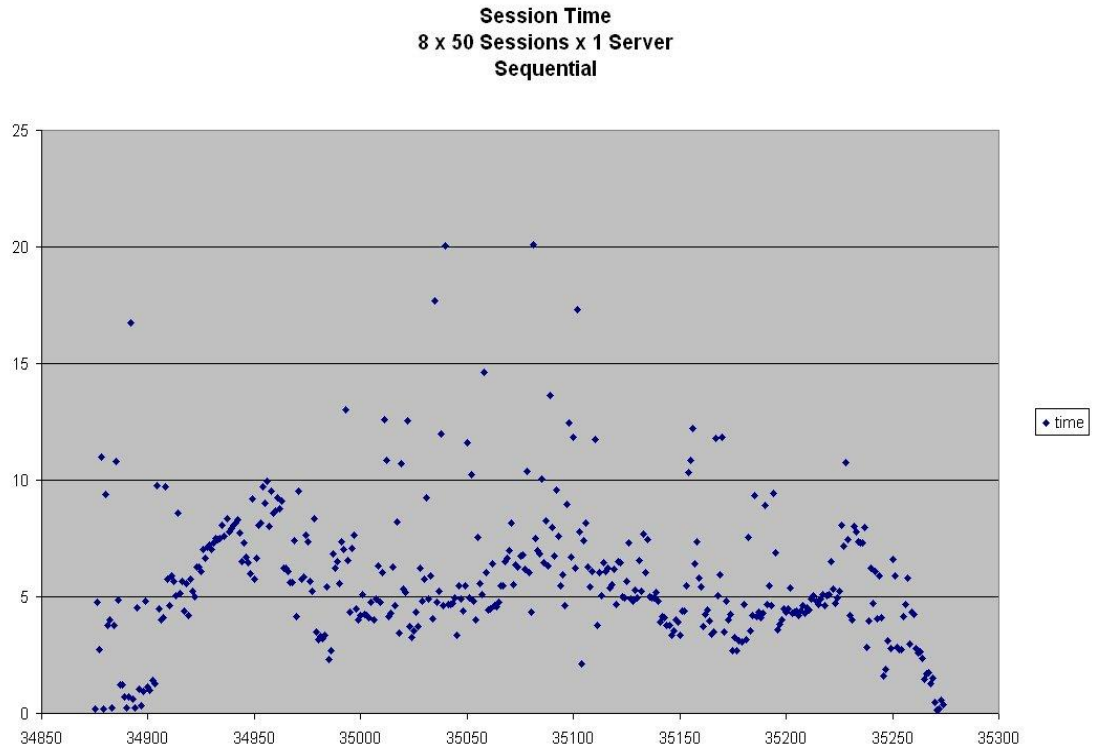
Client (A) client 1 (B) zeus	Query Distribution Type	Sequential Iterations	Server Nodes	Concurrent Client Sessions	Average Delay (ms)	Throughput	Packet Intensity	Total Process Time (s)	Peak Process Count	
A	Random	8	4	400	-	-	-	336.6	230	
B	Random	8	4	400	5.76863794	19649.033	86.676	336.6	230	
A	Load Balanced	8	2	50	0.17195826	252105.315	2907.682	122.09	128	
A	Load Balanced	8	4	50	0.08090560	522526.965	6180.042	50.96	92	
A	Load Balanced	8	2	100	0.13790886	318776.122	3625.583	50.96	92	
A	Load Balanced	8	4	100	0.08812921	484603.770	5673.488	33.98	103	test 1
					0.07656717	556347.453	6530.214	23.06	122	test 2
A	Load Balanced	8	2	200	0.10743538	424712.763	4653.961	41.64	163	
A	Load Balanced	8	4	200	0.05969465	724683.596	8375.961	19.28	138	
A	Load Balanced	8	2	400	0.26592366	170338.792	1880.239	60.95	209	test 1
					0.24486395	211604.990	2041.950	43.03	211	test 2
					0.13944143	358677.363	3585.735	23.34	160	test 3
					0.43814796	99091.412	1141.167	90.04	277	test 4
A	Load Balanced	8	4	400	0.09072802	474901.359	5510.977	24.29	163	test 1
					0.12549306	346656.873	3984.284	30.25	188	test 2
					0.60850789	87940.314	821.682	108.09	257	test 3
					0.12308939	350594.854	4062.089	27.88	191	test 4

Appendix B

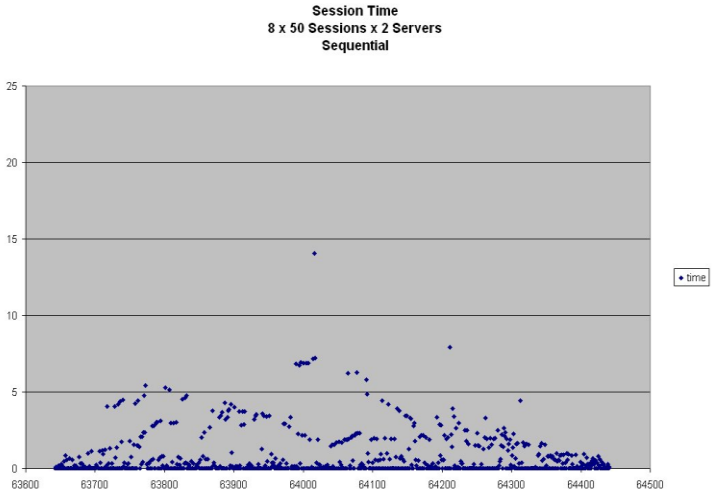
Session Time

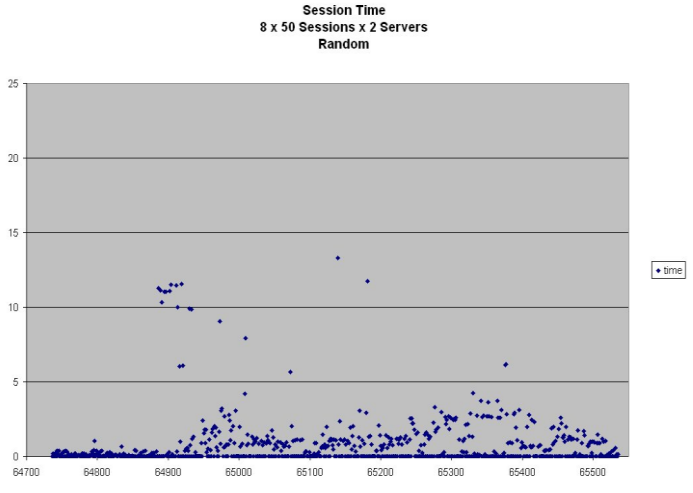
8 consecutive intervals of 50 concurrent sessions.

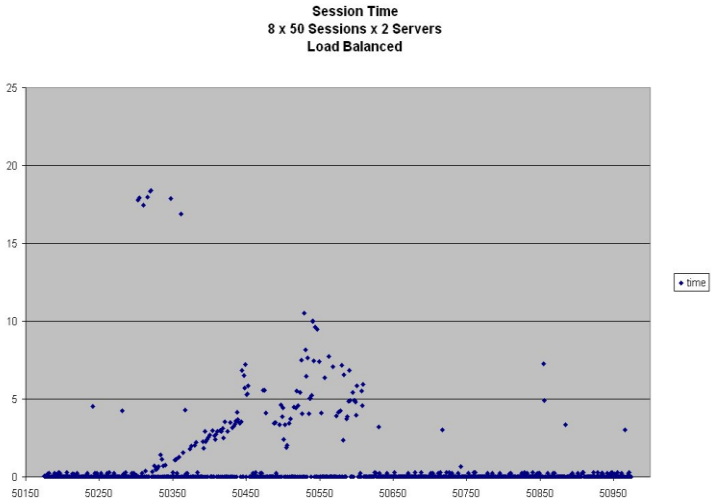
1 Server:



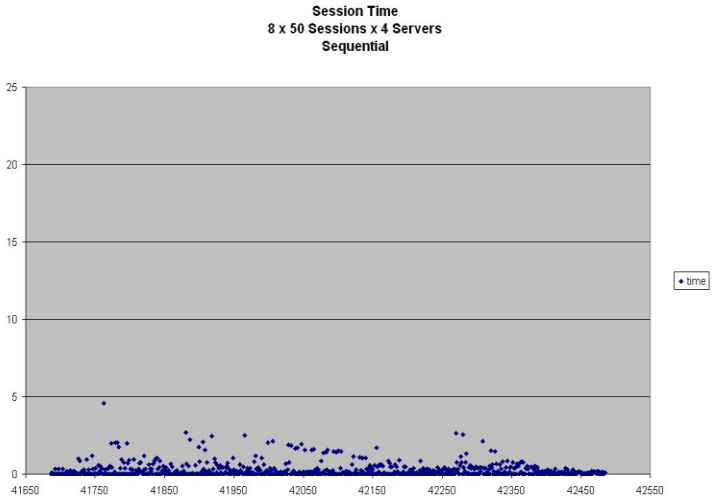
2 Servers:

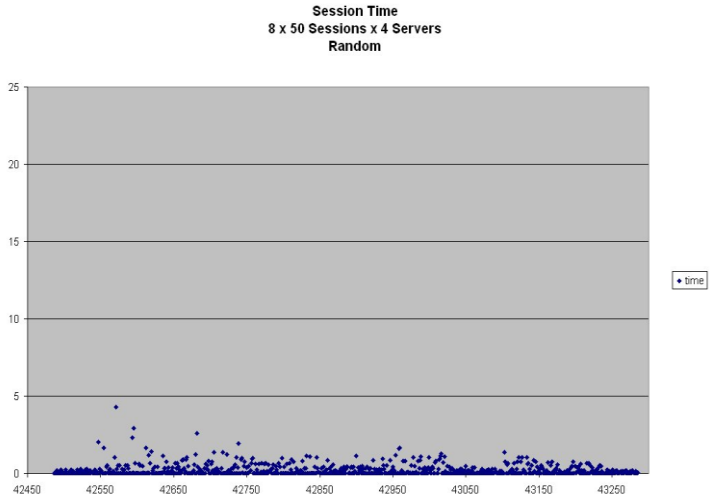

Session Time
8 x 50 Sessions x 2 Servers
Sequential
25
20
15
10
5
0
63600
63700
63800
63900
64000
64100
64200
64300
64400
64500
time

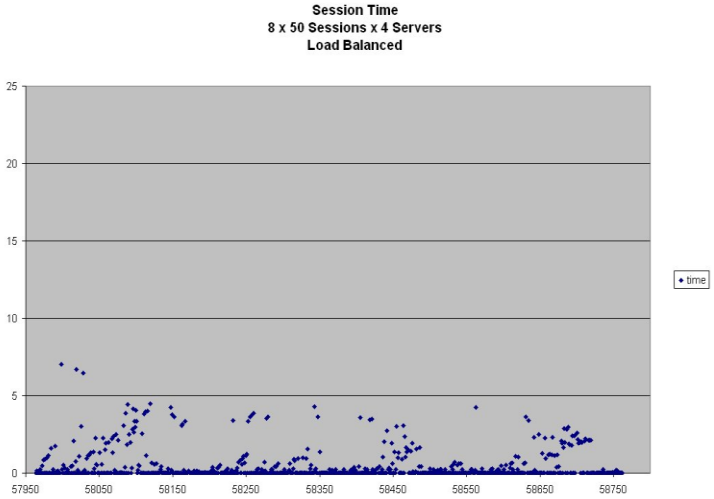

Session Time
8 x 50 Sessions x 2 Servers
Random
25
20
15
10
5
0
64700
64800
64900
65000
65100
65200
65300
65400
65500
time


Session Time
8 x 50 Sessions x 2 Servers
Load Balanced
25
20
15
10
5
0
50150
50250
50350
50450
50550
50650
50750
50850
50950
time

4 Servers:

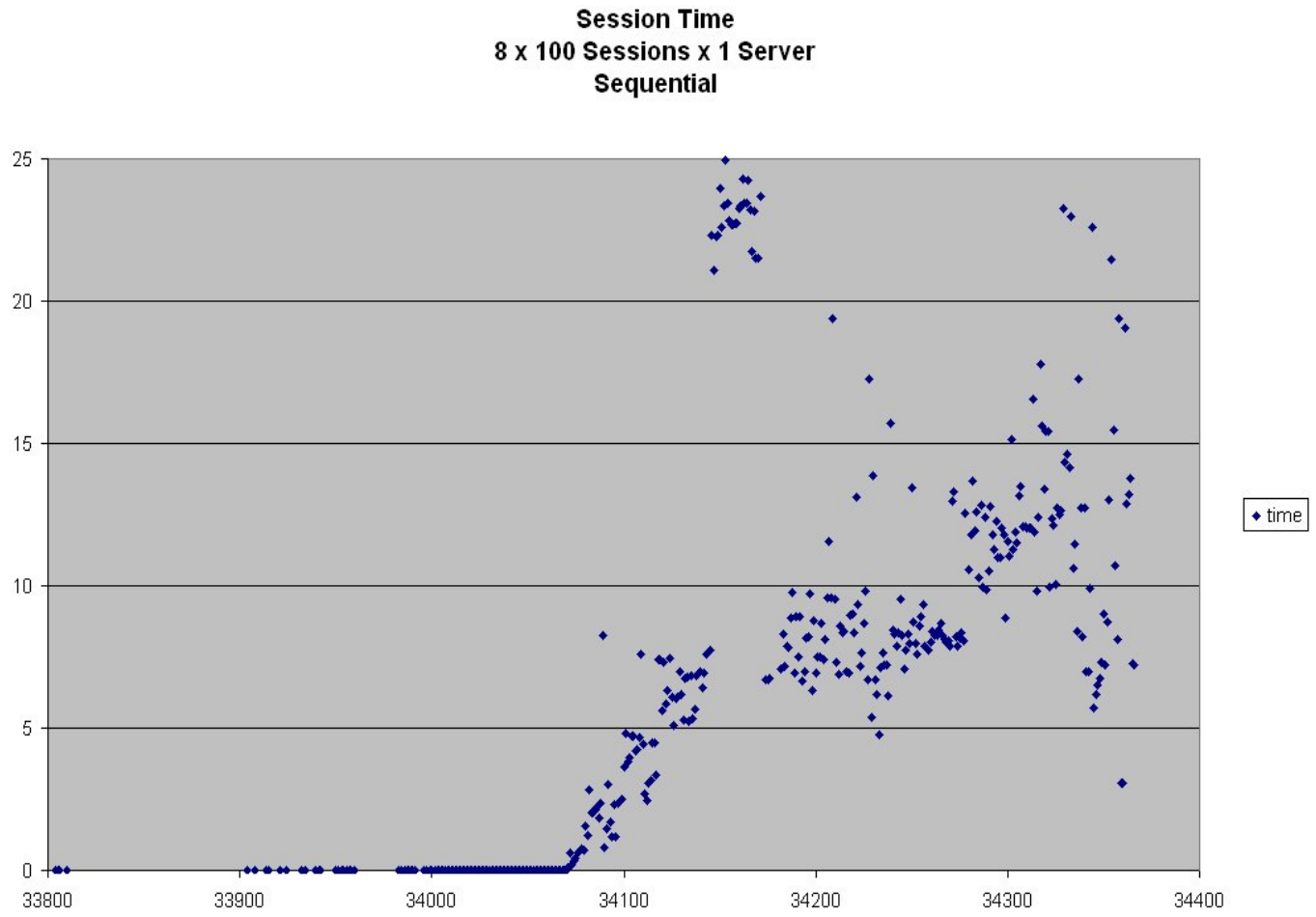

Session Time
8 x 50 Sessions x 4 Servers
Sequential
25
20
15
10
5
0
41650
41750
41850
41950
42050
42150
42250
42350
42450
42550
time


Session Time
8 x 50 Sessions x 4 Servers
Random
25
20
15
10
5
0
42450
42550
42650
42750
42850
42950
43050
43150
43250
time

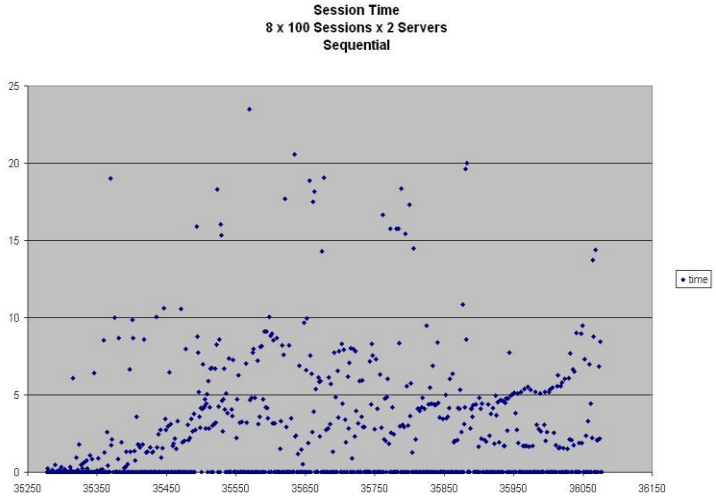

Session Time
8 x 50 Sessions x 4 Servers
Load Balanced
25
20
15
10
5
0
57950
58050
58150
58250
58350
58450
58550
58650
58750
time

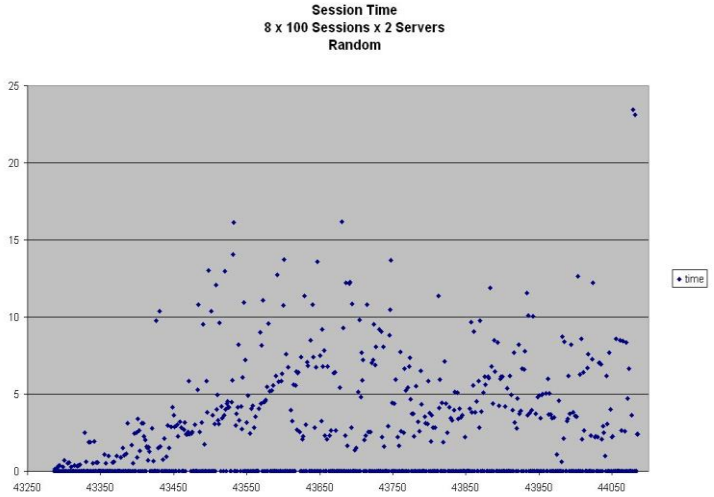
8 consecutive intervals of 100 concurrent sessions.

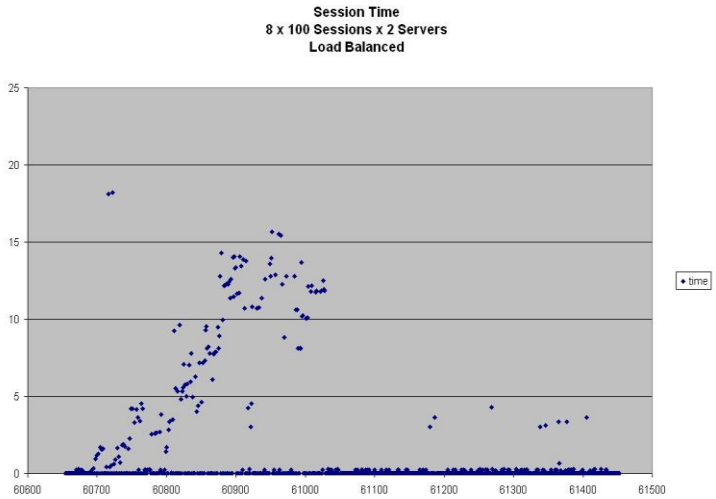
1 Server:



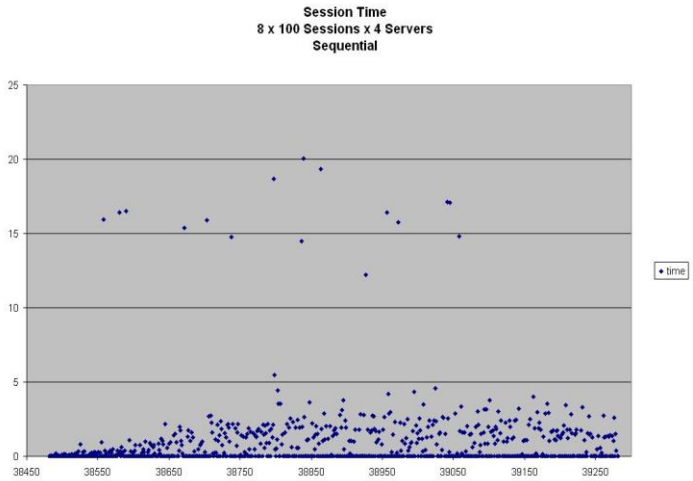
2 Servers:

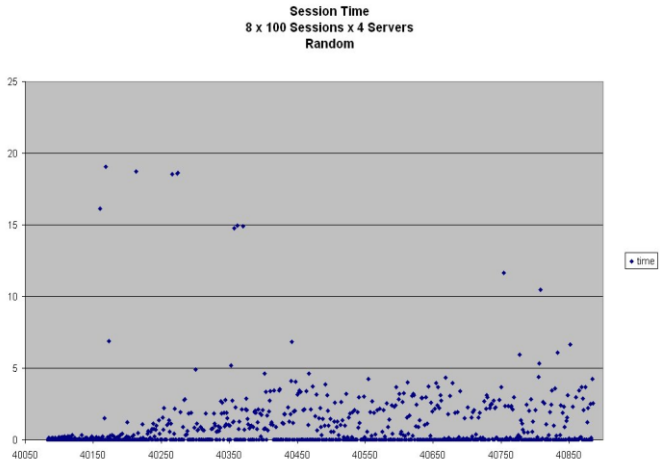

Session Time
8 x 100 Sessions x 2 Servers
Sequential
25
20
15
10
5
0
35250
35350
35450
35550
35650
35750
35850
35950
36050
36150
time

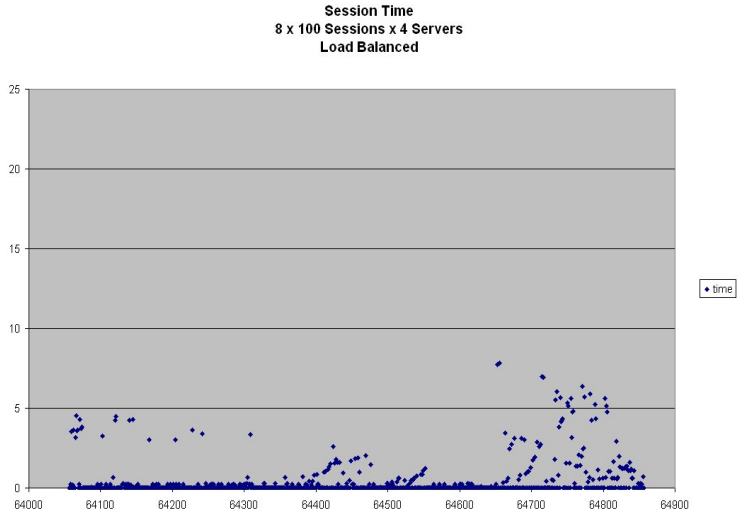

Session Time
8 x 100 Sessions x 2 Servers
Random
25
20
15
10
5
0
43250
43350
43450
43550
43650
43750
43850
43950
44050
time


Session Time
8 x 100 Sessions x 2 Servers
Load Balanced
25
20
15
10
5
0
60600
60700
60800
60900
61000
61100
61200
61300
61400
61500
time

4 Servers:

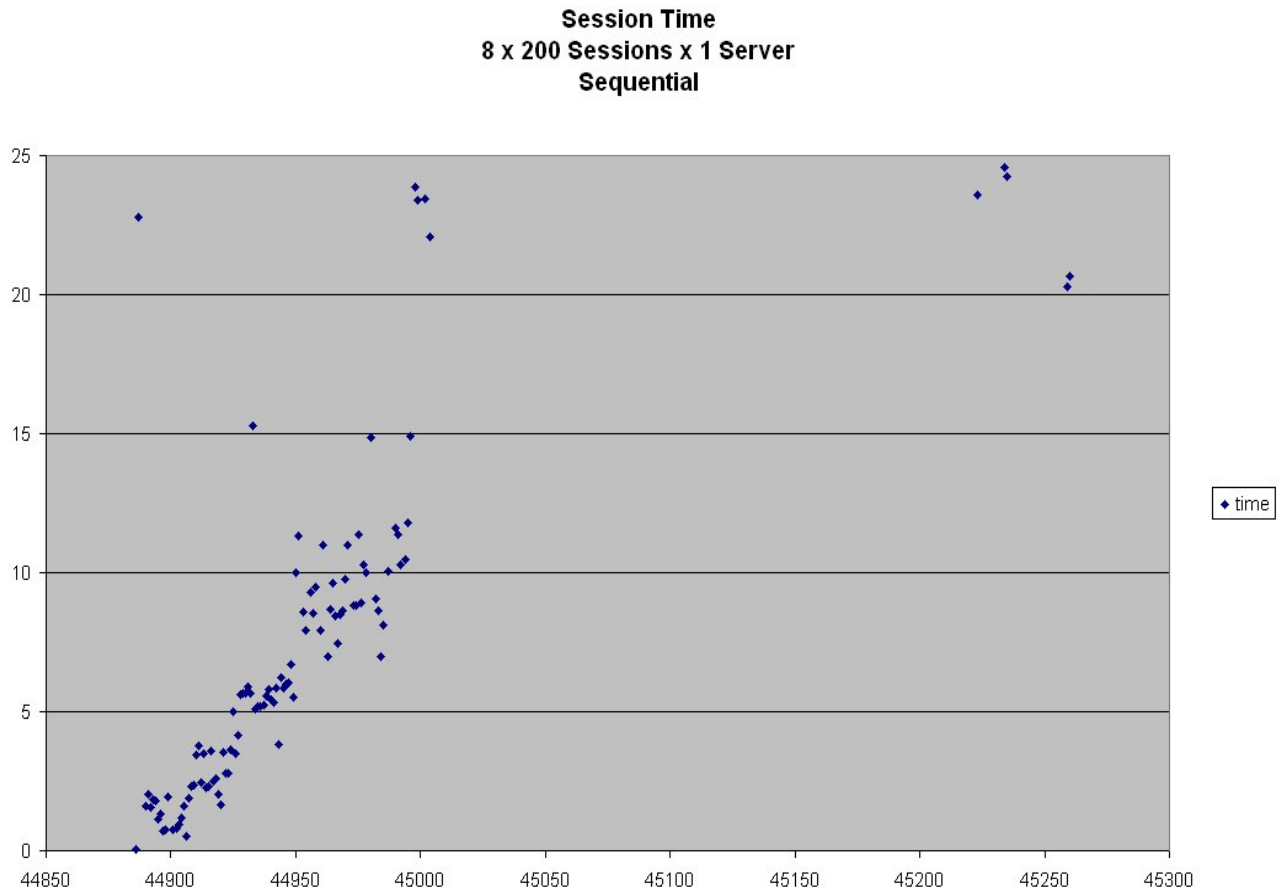

Session Time
8 x 100 Sessions x 4 Servers
Sequential
25
20
15
10
5
0
38450
38550
38650
38750
38850
38950
39050
39150
39250
time


Session Time
8 x 100 Sessions x 4 Servers
Random
25
20
15
10
5
0
40050
40150
40250
40350
40450
40550
40650
40750
40850
time

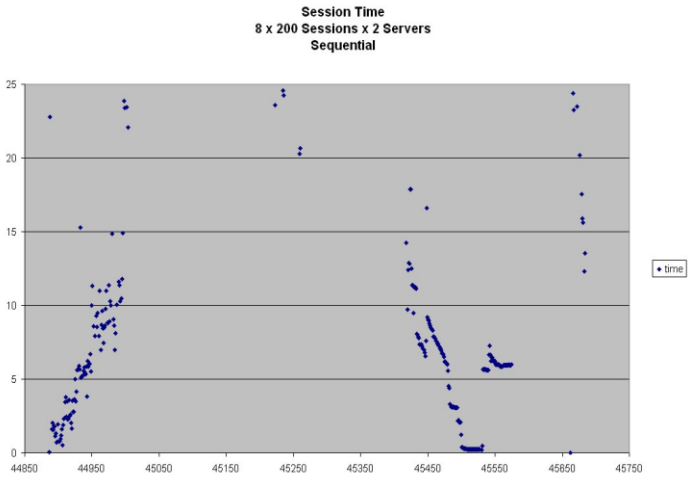

Session Time
8 x 100 Sessions x 4 Servers
Load Balanced
25
20
15
10
5
0
64000
64100
64200
64300
64400
64500
64600
64700
64800
64900
time

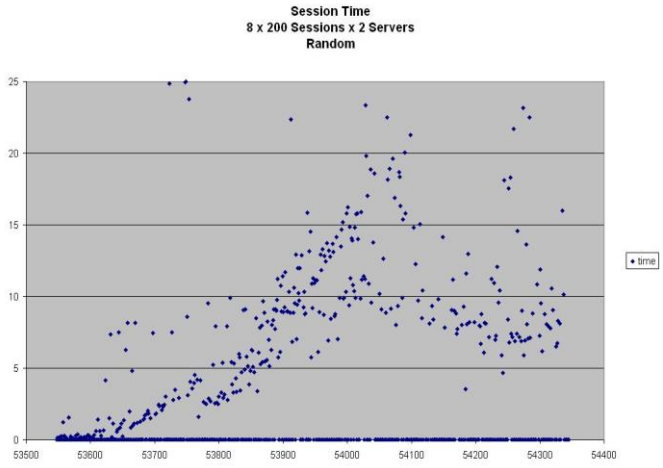
8 consecutive intervals of 200 concurrent sessions.

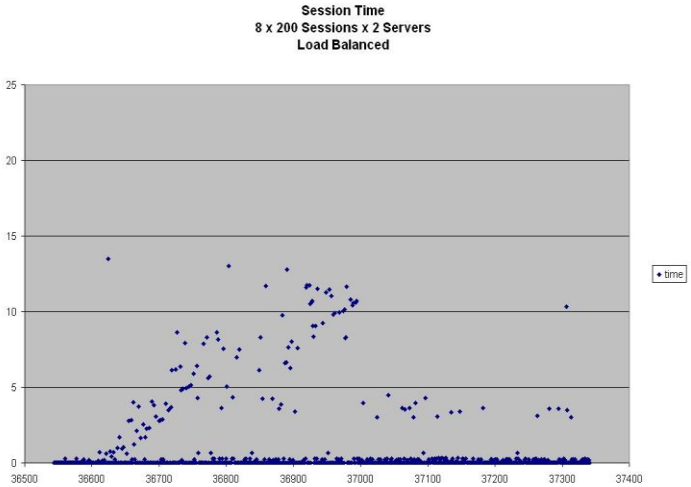
1 Server:



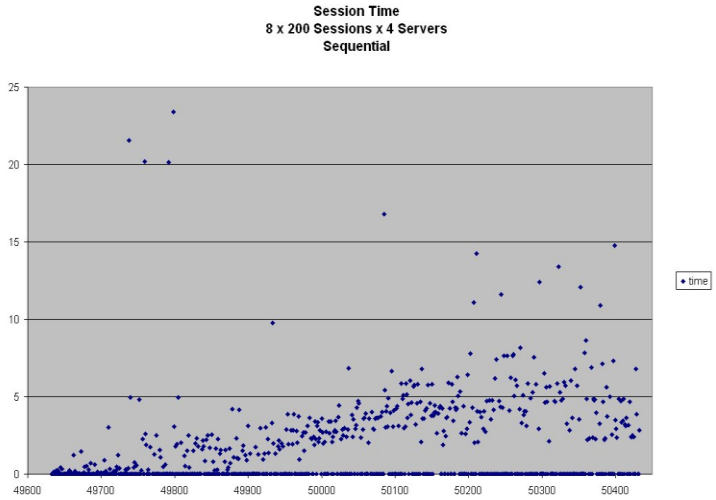
2 Servers:

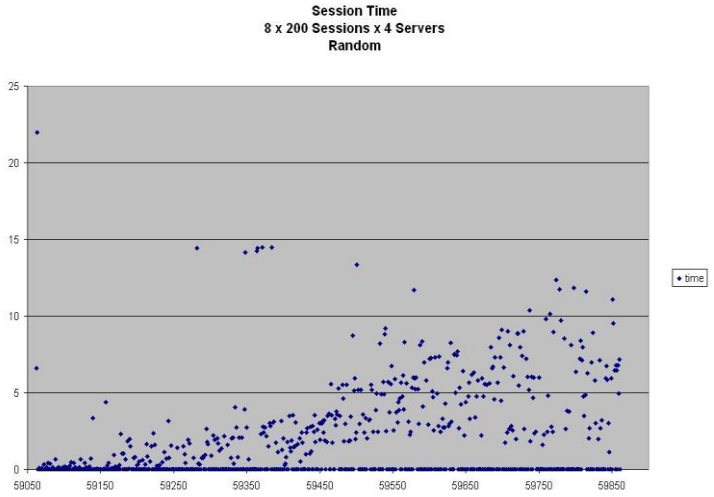

Session Time
8 x 200 Sessions x 2 Servers
Sequential
25
20
15
10
5
0
44850
44950
45050
45150
45250
45350
45450
45550
45650
45750
time

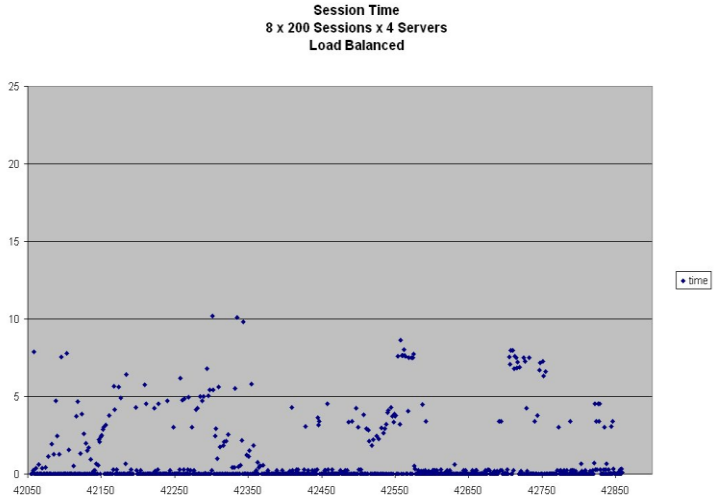

Session Time
8 x 200 Sessions x 2 Servers
Random
25
20
15
10
5
0
53500
53600
53700
53800
53900
54000
54100
54200
54300
54400
time


Session Time
8 x 200 Sessions x 2 Servers
Load Balanced
25
20
15
10
5
0
36500
36600
36700
36800
36900
37000
37100
37200
37300
37400
time

4 Servers:

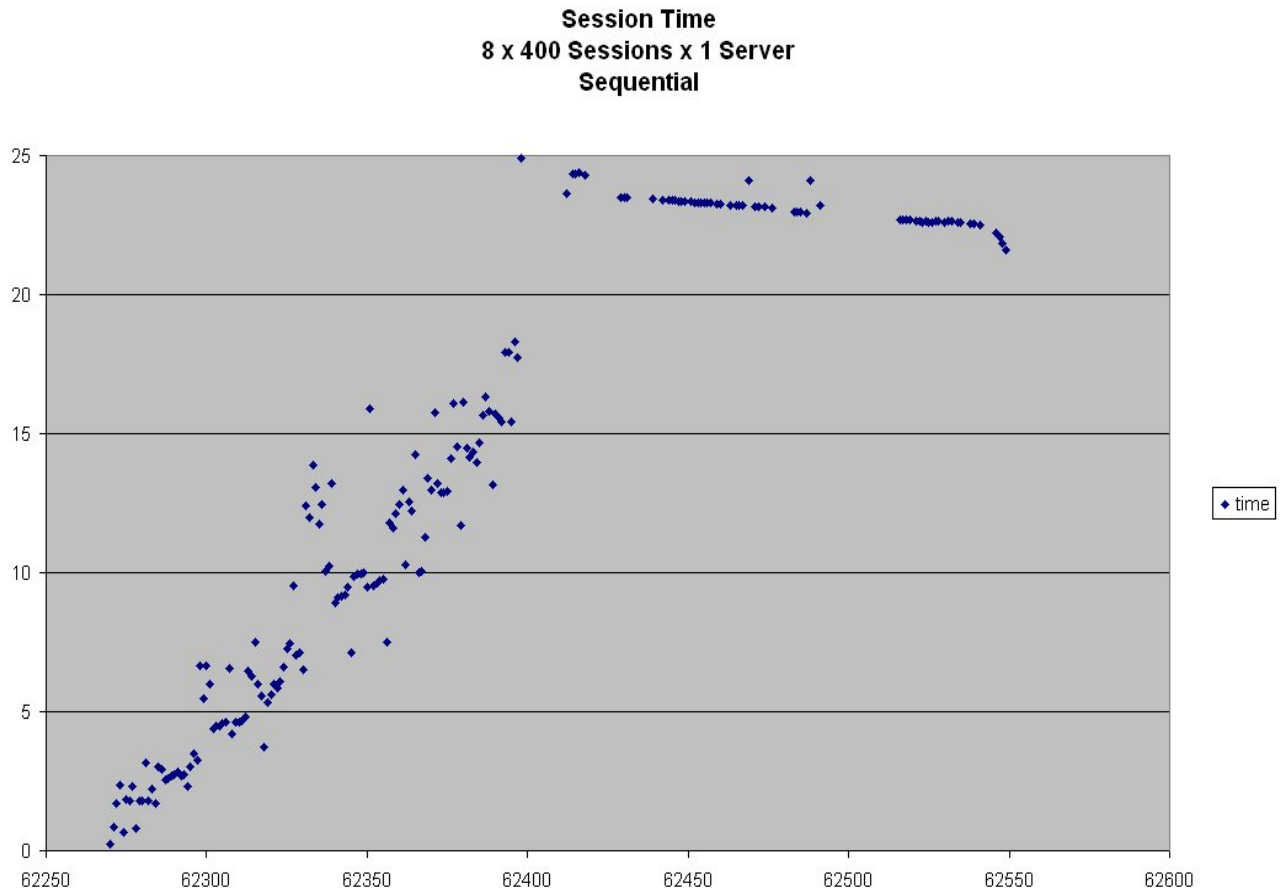

Session Time
8 x 200 Sessions x 4 Servers
Sequential
25
20
15
10
5
0
49600
49700
49800
49900
50000
50100
50200
50300
50400
time


Session Time
8 x 200 Sessions x 4 Servers
Random
25
20
15
10
5
0
59050
59150
59250
59350
59450
59550
59650
59750
59850
time

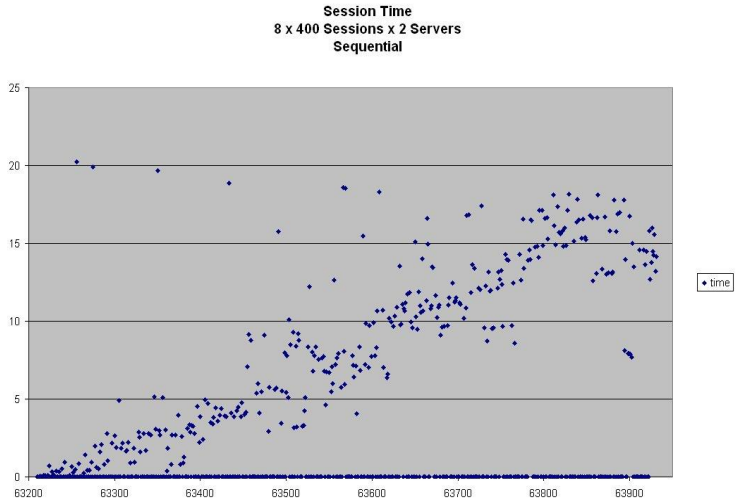

Session Time
8 x 200 Sessions x 4 Servers
Load Balanced
25
20
15
10
5
0
42050
42150
42250
42350
42450
42550
42650
42750
42850
time

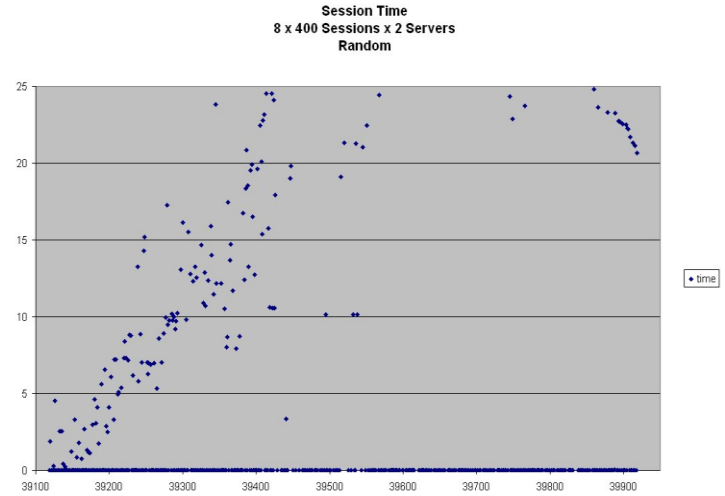
8 consecutive intervals of 400 concurrent sessions.

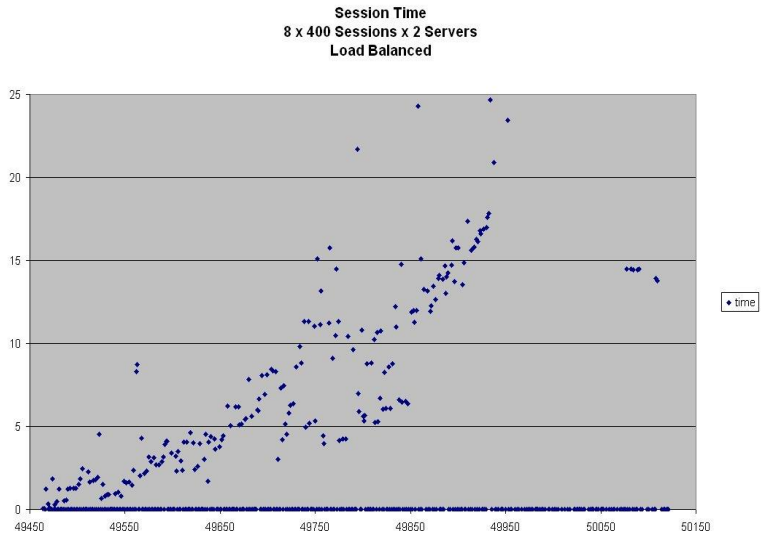
1 Server:



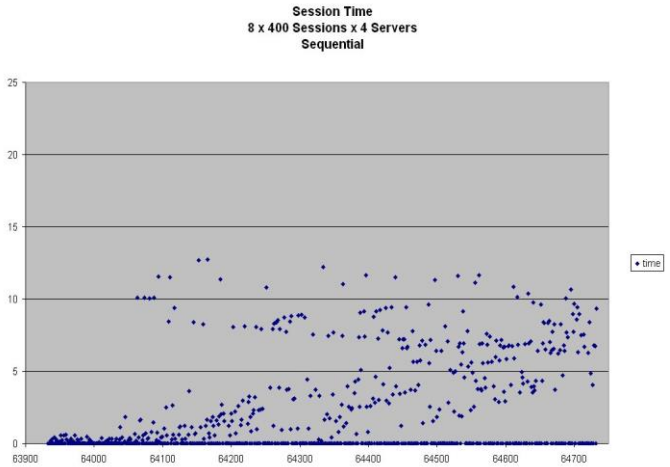
2 Servers:

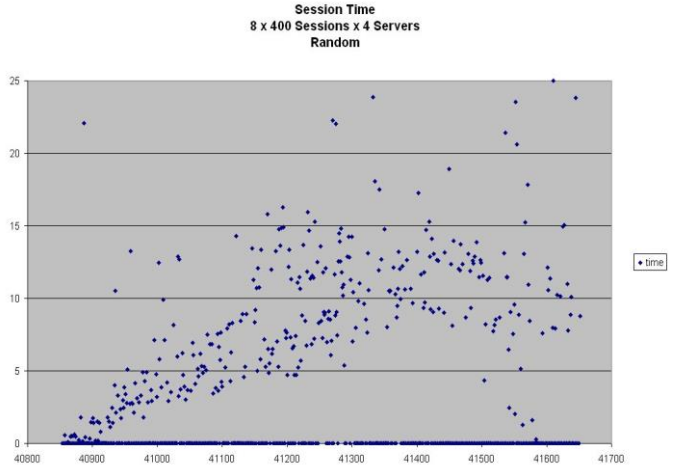

Session Time
8 x 400 Sessions x 2 Servers
Sequential
25
20
15
10
5
0
63200
63300
63400
63500
63600
63700
63800
63900
time

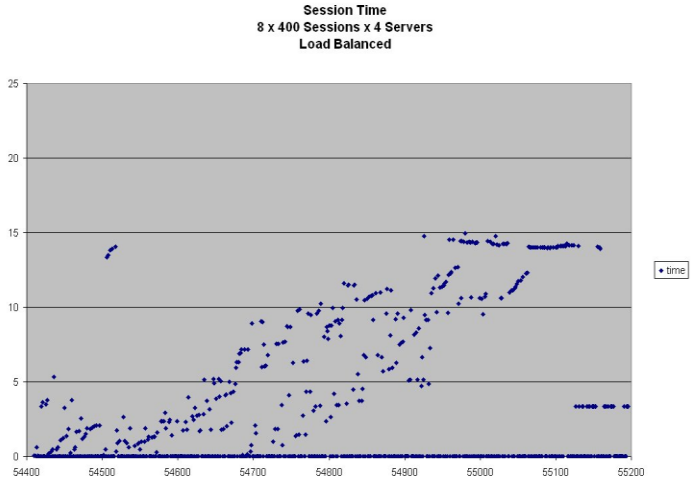

Session Time
8 x 400 Sessions x 2 Servers
Random
25
20
15
10
5
0
39100
39200
39300
39400
39500
39600
39700
39800
39900
time


Session Time
8 x 400 Sessions x 2 Servers
Load Balanced
25
20
15
10
5
0
49450
49550
49650
49750
49850
49950
50050
50150
time

4 Servers:


Session Time
8 x 400 Sessions x 4 Servers
Sequential
25
20
15
10
5
0
63900
64000
64100
64200
64300
64400
64500
64600
64700
time


Session Time
8 x 400 Sessions x 4 Servers
Random
25
20
15
10
5
0
40800
40900
41000
41100
41200
41300
41400
41500
41600
41700
time

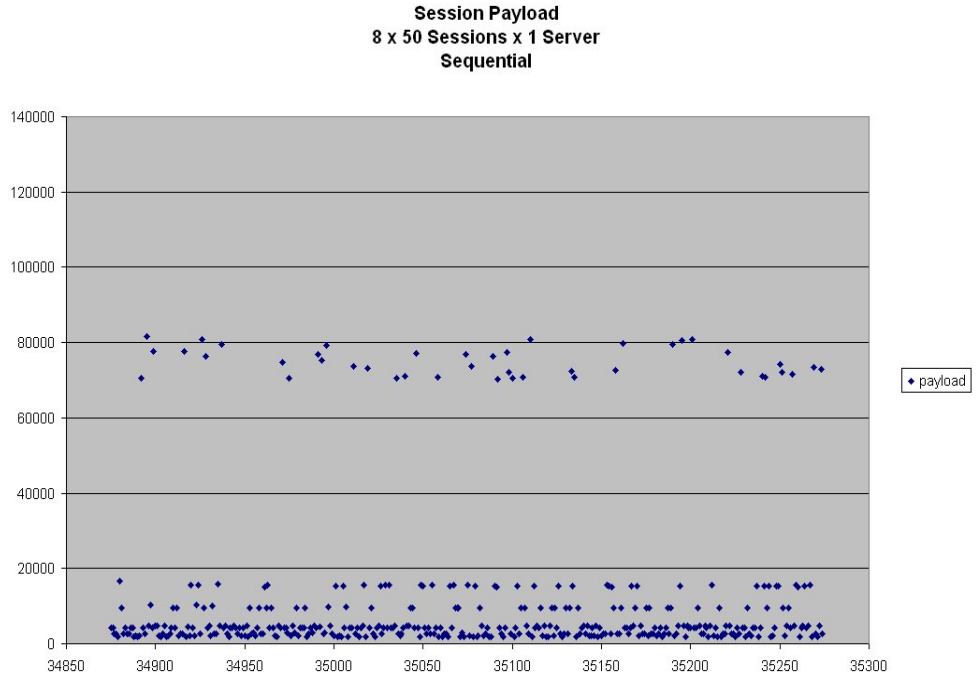

Session Time
8 x 400 Sessions x 4 Servers
Load Balanced
25
20
15
10
5
0
54400
54500
54600
54700
54800
54900
55000
55100
55200
time

Appendix C

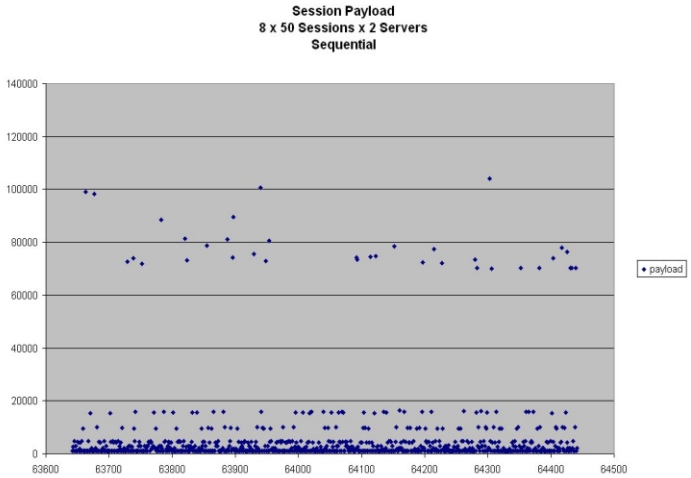
Session Payload

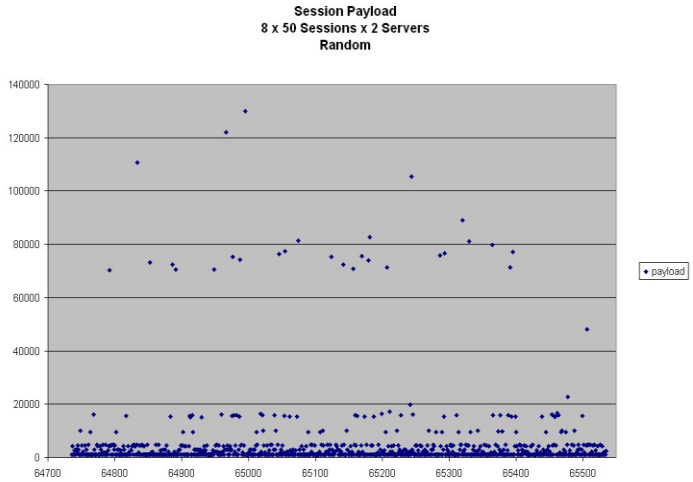
8 consecutive intervals of 50 concurrent sessions.

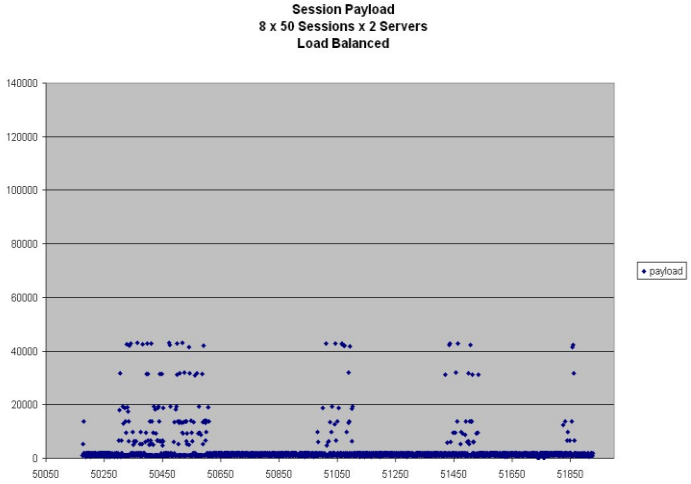
1 Server:



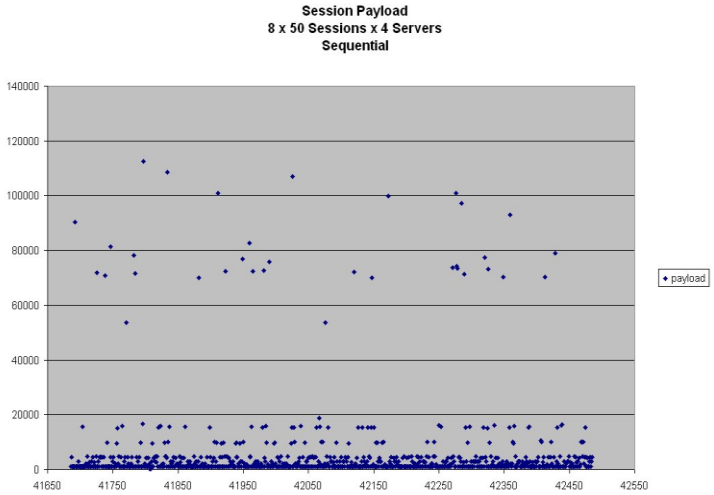
2 Servers:

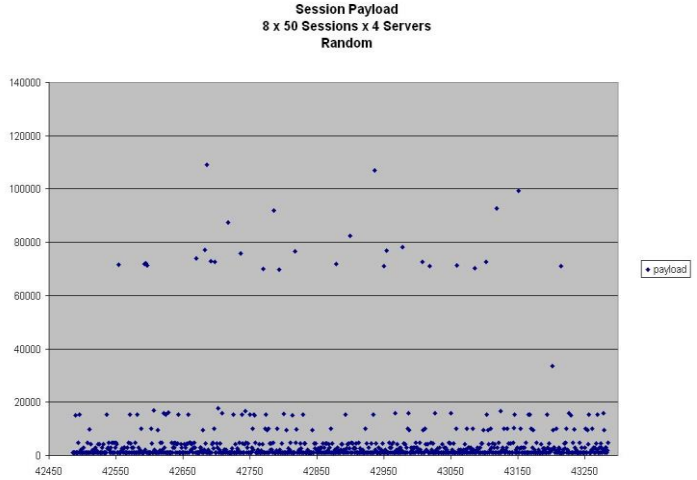

Session Payload
8 x 50 Sessions x 2 Servers
Sequential
140000
120000
100000
80000
60000
40000
20000
0
63600
63700
63800
63900
64000
64100
64200
64300
64400
64500
payload

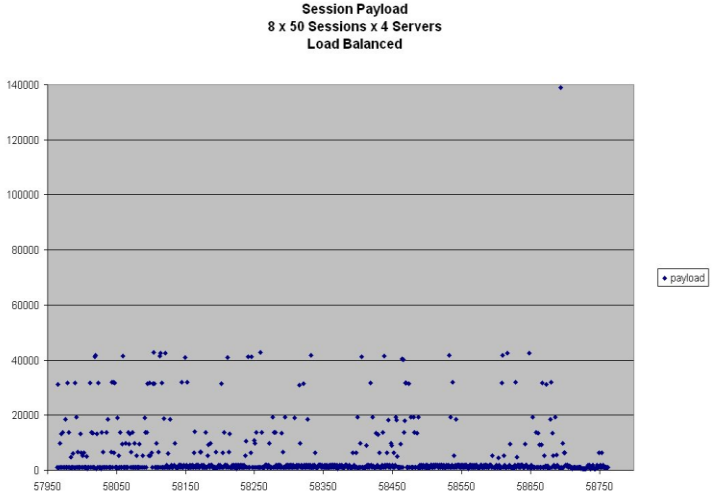

Session Payload
8 x 50 Sessions x 2 Servers
Random
140000
120000
100000
80000
60000
40000
20000
0
64700
64800
64900
65000
65100
65200
65300
65400
65500
payload


Session Payload
8 x 50 Sessions x 2 Servers
Load Balanced
140000
120000
100000
80000
60000
40000
20000
0
50050
50250
50450
50650
50850
51050
51250
51450
51650
51850
payload

4 Servers:

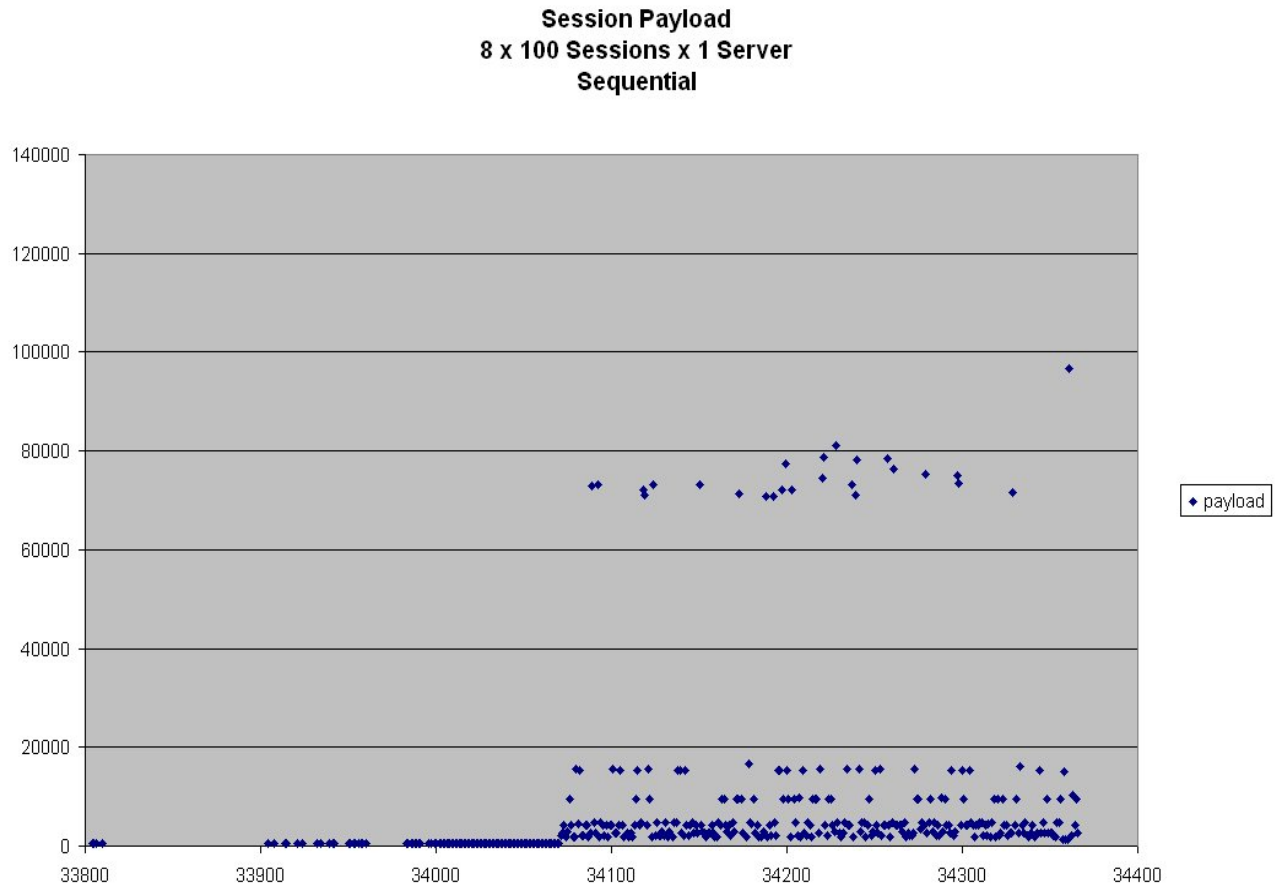

Session Payload
8 x 50 Sessions x 4 Servers
Sequential
140000
120000
100000
80000
60000
40000
20000
0
41650
41750
41850
41950
42050
42150
42250
42350
42450
42550
payload


Session Payload
8 x 50 Sessions x 4 Servers
Random
140000
120000
100000
80000
60000
40000
20000
0
42450
42550
42650
42750
42850
42950
43050
43150
43250
payload

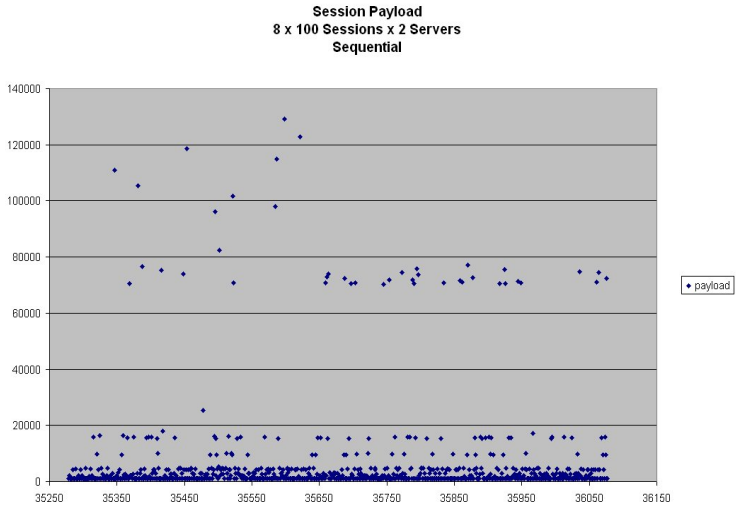

Session Payload
8 x 50 Sessions x 4 Servers
Load Balanced
140000
120000
100000
80000
60000
40000
20000
0
57950
58050
58150
58250
58350
58450
58550
58650
58750
payload

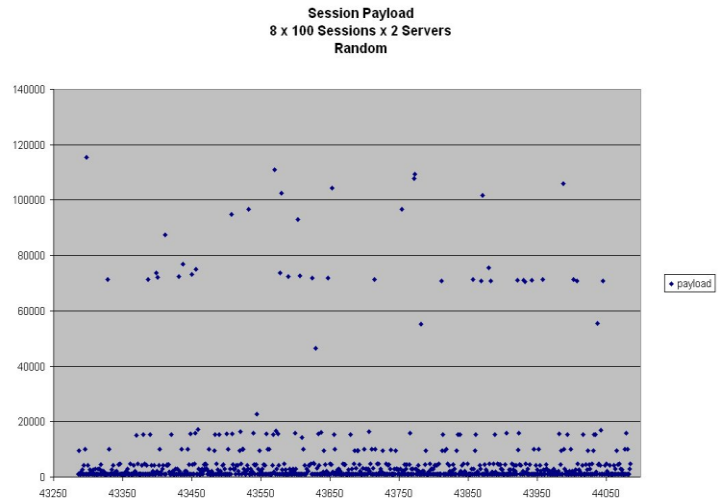
8 consecutive intervals of 100 concurrent sessions.

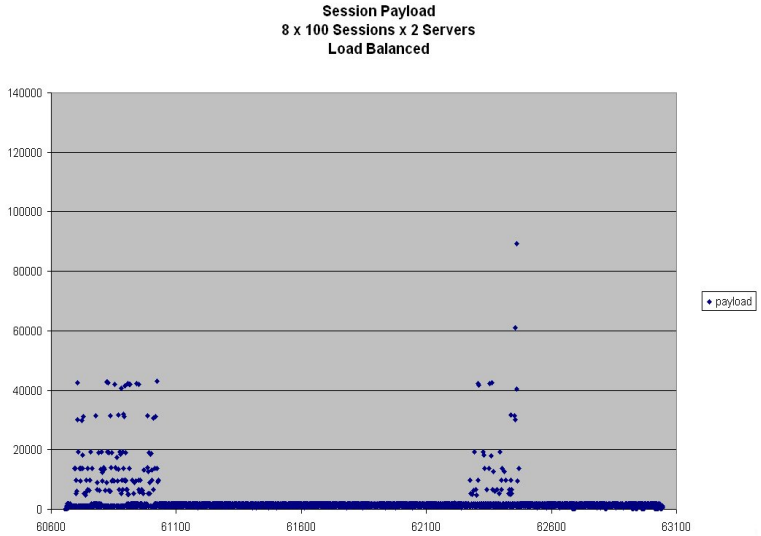
1 Server:



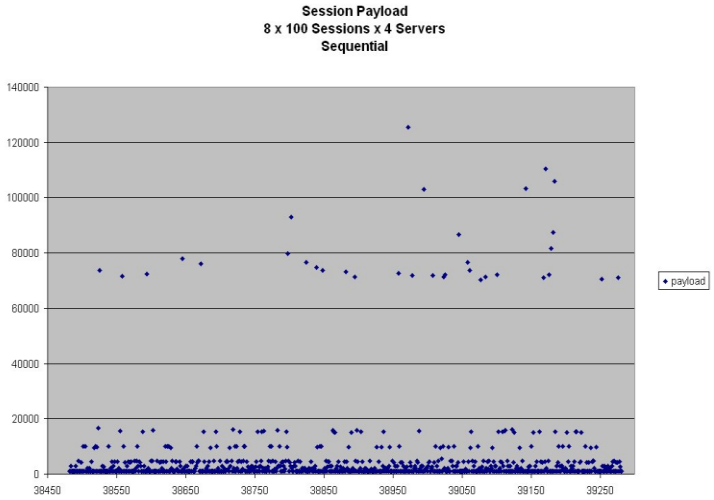
2 Servers:

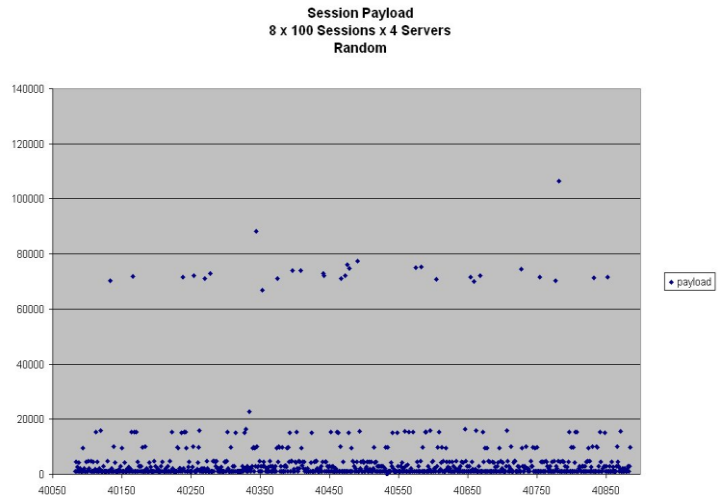

Session Payload
8 x 100 Sessions x 2 Servers
Sequential
140000
120000
100000
80000
60000
40000
20000
0
35250
35350
35450
35550
35650
35750
35850
35950
36050
36150
payload

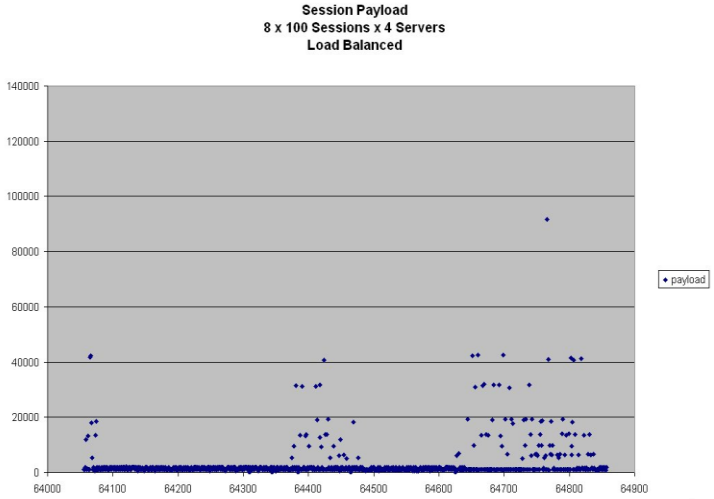

Session Payload
8 x 100 Sessions x 2 Servers
Random
140000
120000
100000
80000
60000
40000
20000
0
43250
43350
43450
43550
43650
43750
43850
43950
44050
payload


Session Payload
8 x 100 Sessions x 2 Servers
Load Balanced
140000
120000
100000
80000
60000
40000
20000
0
60600
61100
61600
62100
62600
63100
payload

4 Servers:

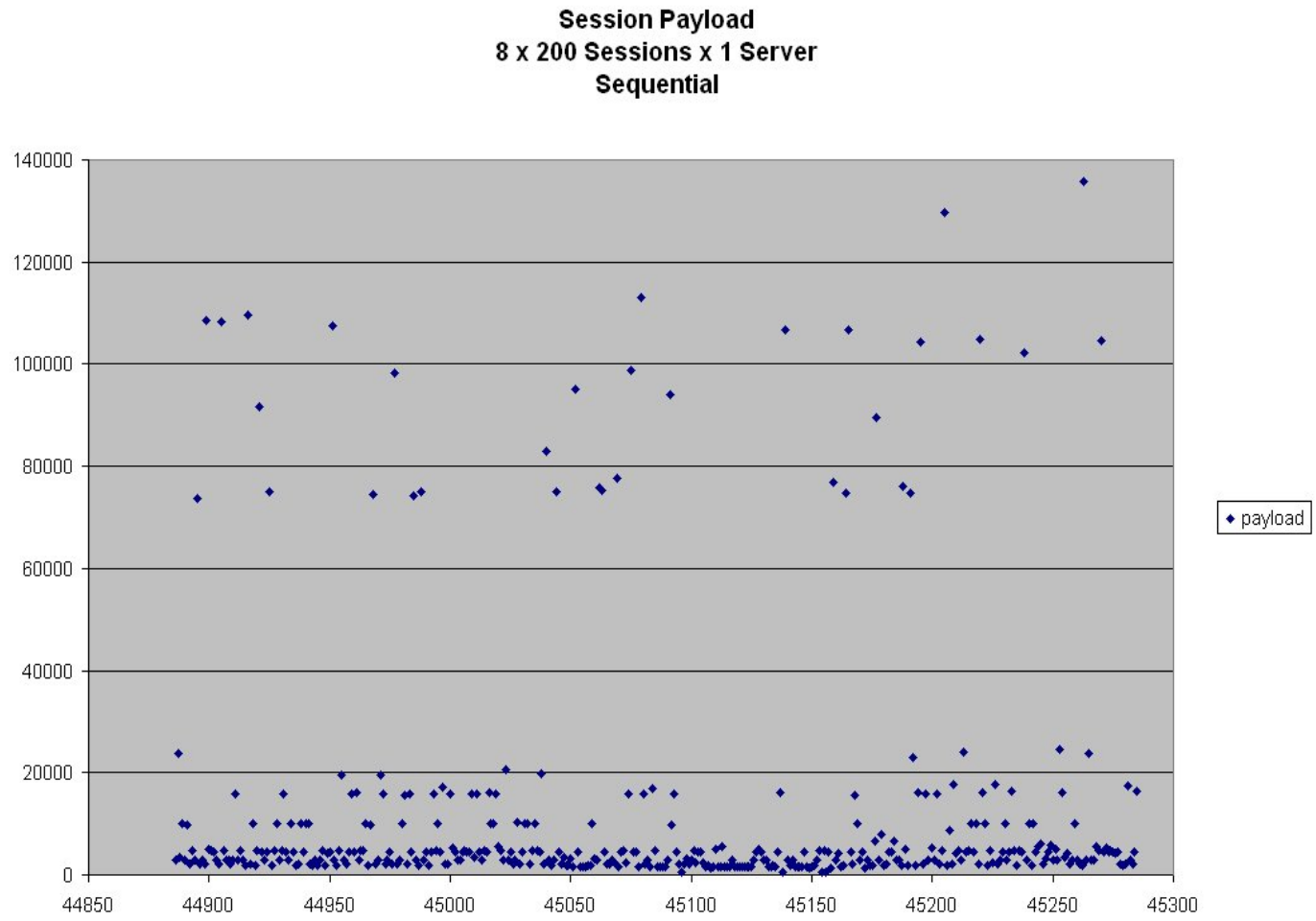

Session Payload
8 x 100 Sessions x 4 Servers
Sequential
140000
120000
100000
80000
60000
40000
20000
0
38450
38550
38650
38750
38850
38950
39050
39150
39250
payload


Session Payload
8 x 100 Sessions x 4 Servers
Random
140000
120000
100000
80000
60000
40000
20000
0
40050
40150
40250
40350
40450
40550
40650
40750
40850
payload

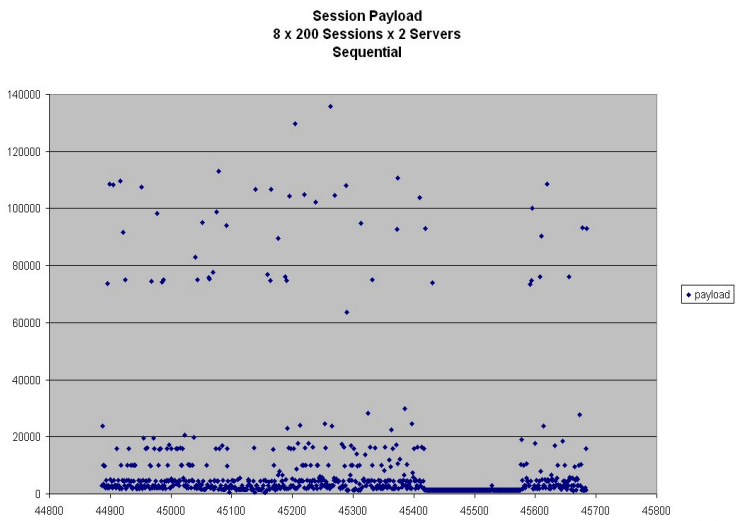

Session Payload
8 x 100 Sessions x 4 Servers
Load Balanced
140000
120000
100000
80000
60000
40000
20000
0
64000
64100
64200
64300
64400
64500
64600
64700
64800
64900
payload

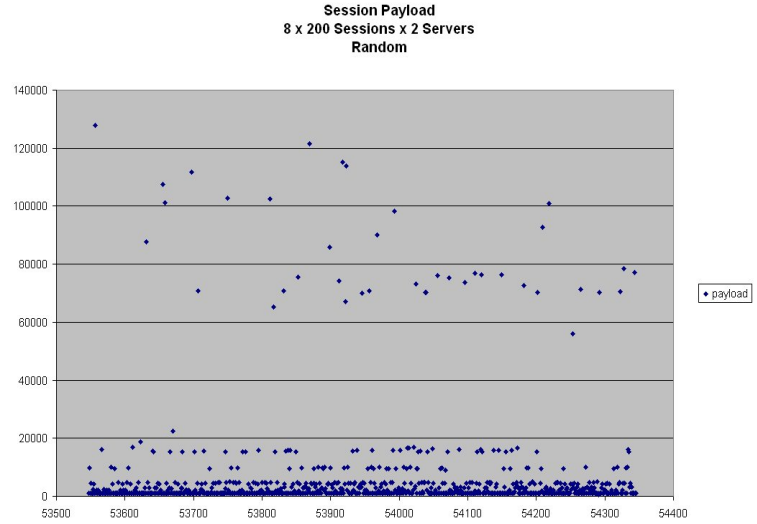
8 consecutive intervals of 200 concurrent sessions.

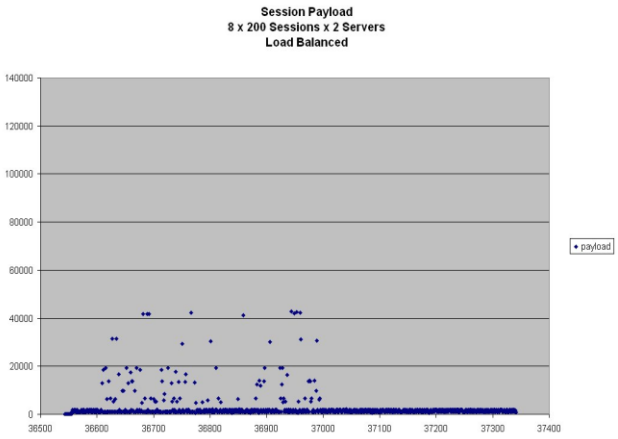
1 Server:



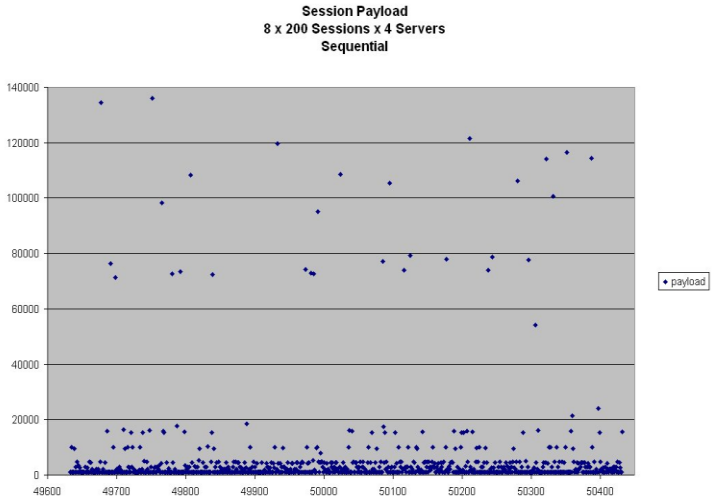
2 Servers:

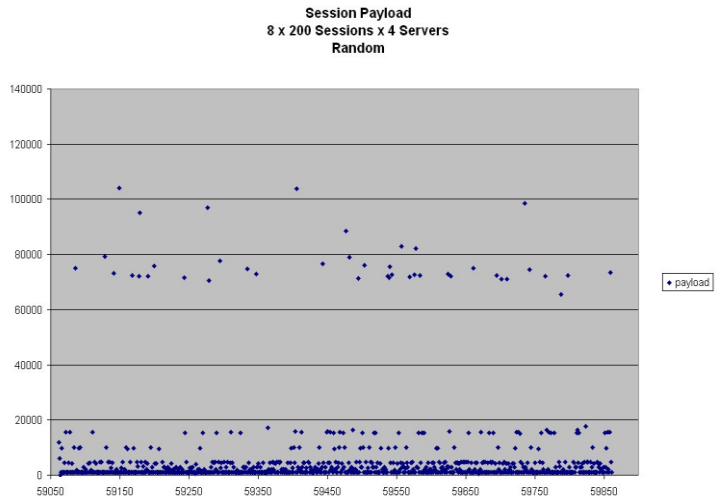

Session Payload
8 x 200 Sessions x 2 Servers
Sequential
140000
120000
100000
80000
60000
40000
20000
0
44800
44900
45000
45100
45200
45300
45400
45500
45600
45700
45800
payload

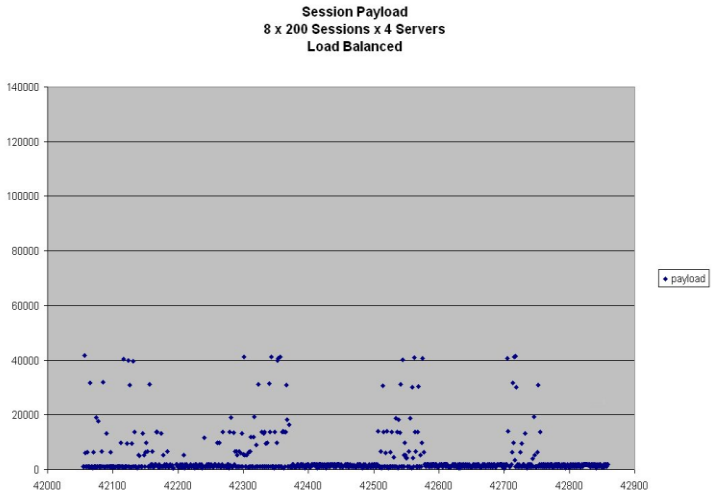

Session Payload
8 x 200 Sessions x 2 Servers
Random
140000
120000
100000
80000
60000
40000
20000
0
53500
53600
53700
53800
53900
54000
54100
54200
54300
54400
payload


Session Payload
8 x 200 Sessions x 2 Servers
Load Balanced
140000
120000
100000
80000
60000
40000
20000
0
36500
36600
36700
36800
36900
37000
37100
37200
37300
37400
payload

4 Servers:

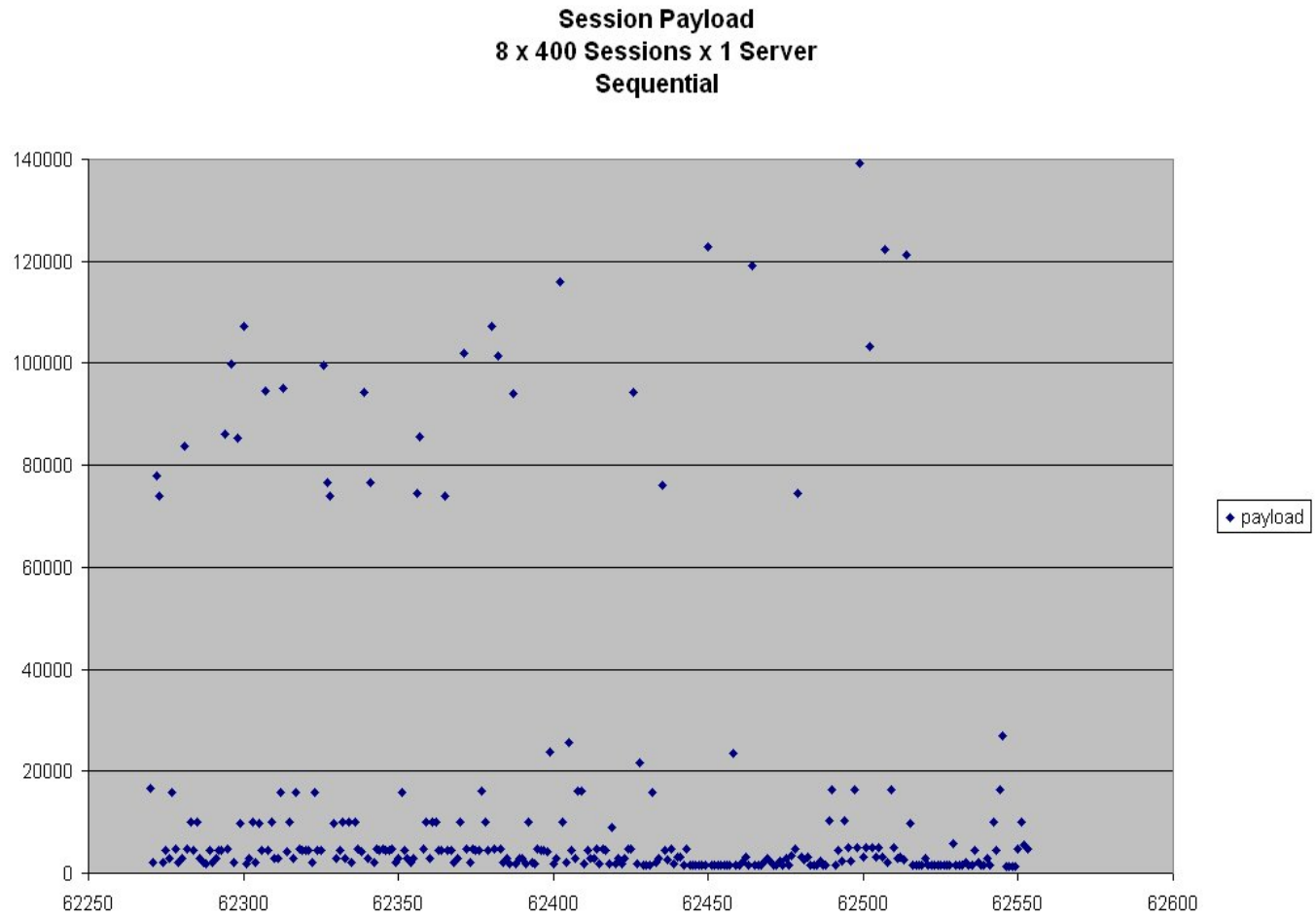

Session Payload
8 x 200 Sessions x 4 Servers
Sequential
140000
120000
100000
80000
60000
40000
20000
0
49600
49700
49800
49900
50000
50100
50200
50300
50400
payload


Session Payload
8 x 200 Sessions x 4 Servers
Random
140000
120000
100000
80000
60000
40000
20000
0
59050
59150
59250
59350
59450
59550
59650
59750
59850
payload

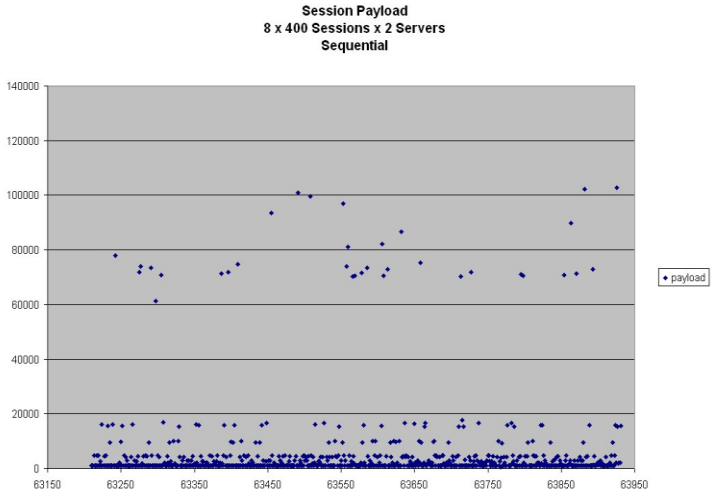

Session Payload
8 x 200 Sessions x 4 Servers
Load Balanced
140000
120000
100000
80000
60000
40000
20000
0
42000
42100
42200
42300
42400
42500
42600
42700
42800
42900
payload

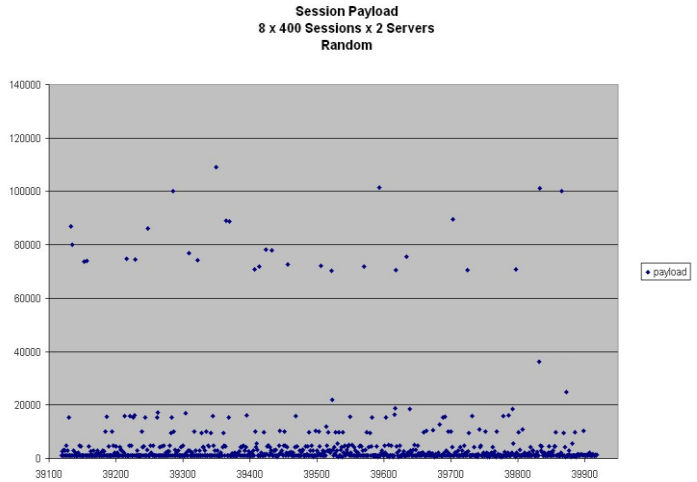
8 consecutive intervals of 400 concurrent sessions.

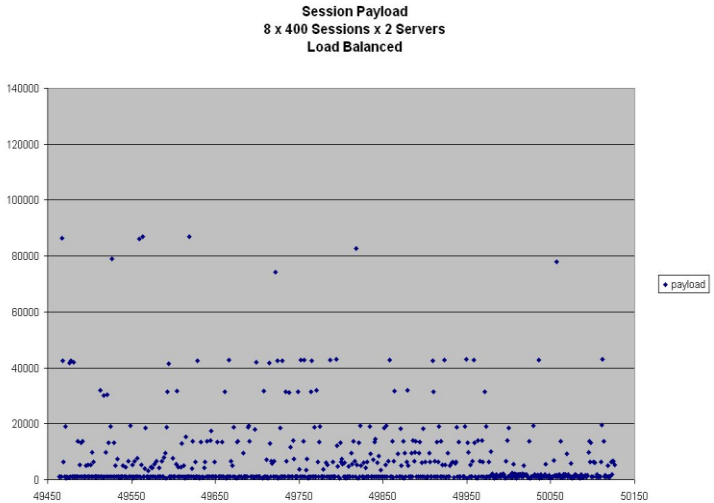
1 Server:



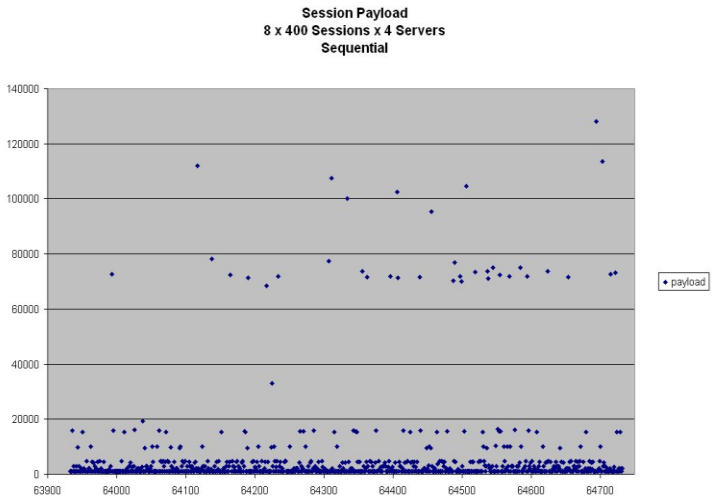
2 Servers:

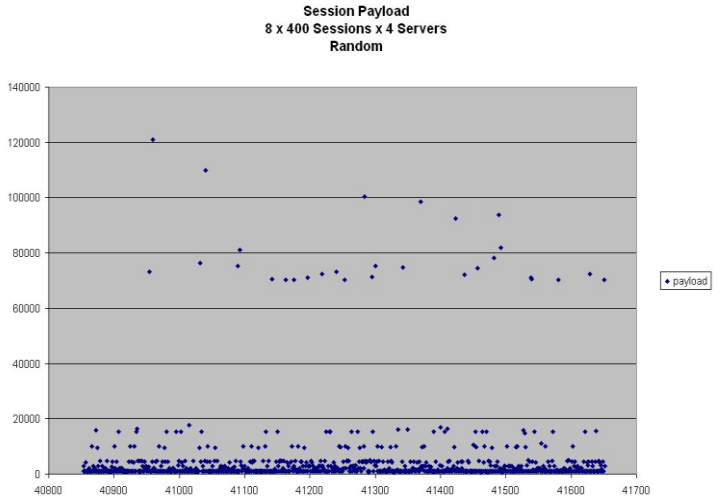

Session Payload
8 x 400 Sessions x 2 Servers
Sequential
140000
120000
100000
80000
60000
40000
20000
0
63150
63250
63350
63450
63550
63650
63750
63850
63950
payload

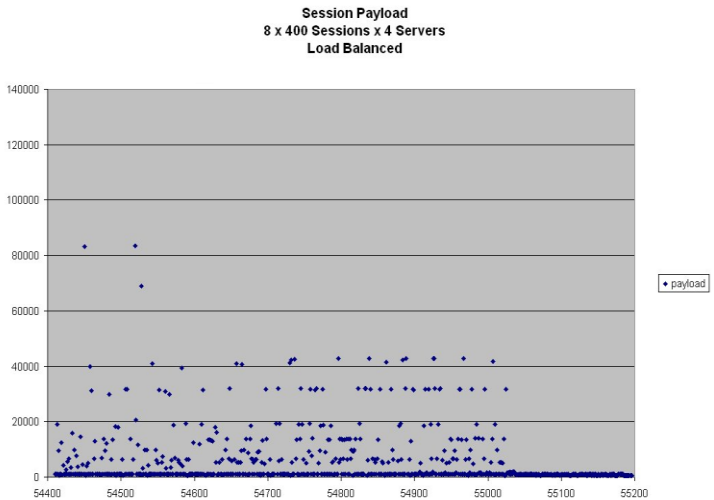

Session Payload
8 x 400 Sessions x 2 Servers
Random
140000
120000
100000
80000
60000
40000
20000
0
39100
39200
39300
39400
39500
39600
39700
39800
39900
payload


Session Payload
8 x 400 Sessions x 2 Servers
Load Balanced
140000
120000
100000
80000
60000
40000
20000
0
49450
49550
49650
49750
49850
49950
50050
50150
payload

4 Servers:


Session Payload
8 x 400 Sessions x 4 Servers
Sequential
140000
120000
100000
80000
60000
40000
20000
0
63900
64000
64100
64200
64300
64400
64500
64600
64700
payload


Session Payload
8 x 400 Sessions x 4 Servers
Random
140000
120000
100000
80000
60000
40000
20000
0
40800
40900
41000
41100
41200
41300
41400
41500
41600
41700
payload


Session Payload
8 x 400 Sessions x 4 Servers
Load Balanced
140000
120000
100000
80000
60000
40000
20000
0
54400
54500
54600
54700
54800
54900
55000
55100
55200
payload