How To Emulate Web Traffic Using Standard Load Testing Tools

James F Brady

State Of Nevada, Carson City, NV 89701

jfbrady@admin.nv.gov

Neil J Gunther

Performance Dynamics, Castro Valley, Ca 94552

njgunther@perfdynamics.com

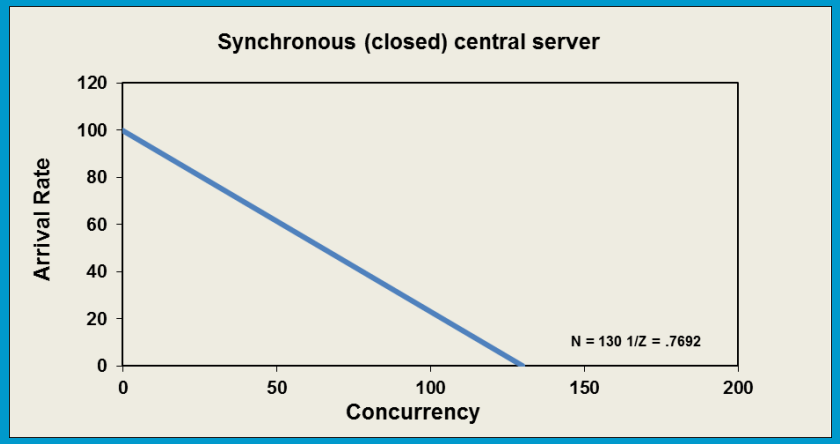
Introduction

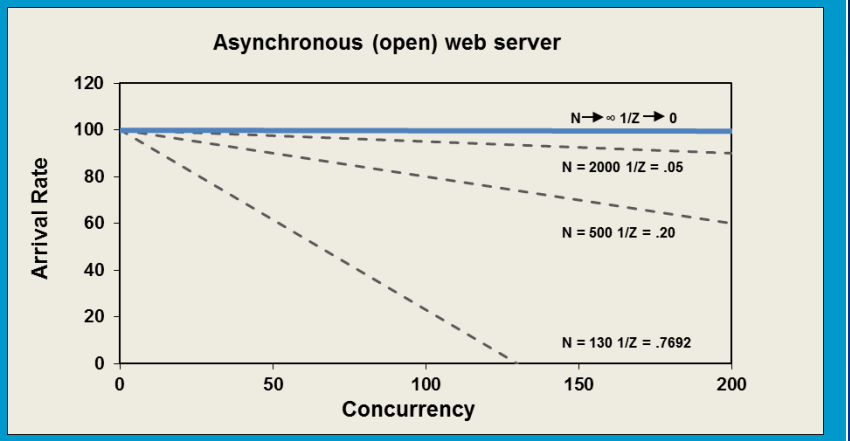
- We Describe The Problem.
- We Provide A Conceptual Solution To The Problem.
- We Show How To Verify That Solution Is Achieved.
- We Present a Detailed Case Study Based On Our Methodology.

The Problem

- Conventional Load Testing Tools Simulate A Fixed Set Of Users Accessing The System Under Test (SUT).
- Web Traffic Is Produced By A Transient User Population Who's Size Is Unknown At Any Given Point In Time.
- How Can Conventional Load Testing Tools Be Used To Produce Traffic Consistent With This Dynamic Web User Population?

Synchronous Vs Asynchronous Arrivals





Row	N	Z	1 / Z	Q	λ
1	130	1.30	0.7692	0	100
2	130	1.30	0.7692	130	0

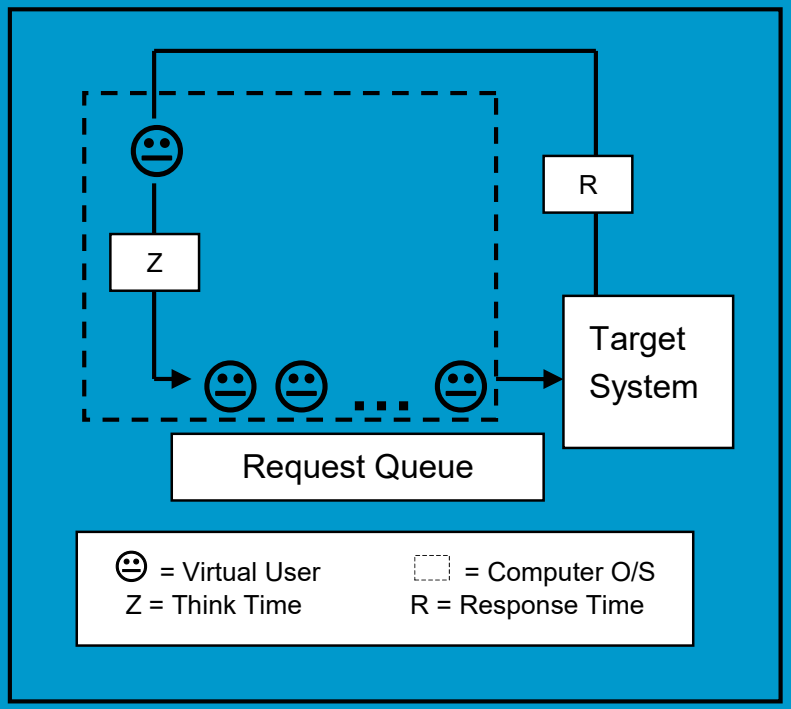
Row	N	Z	1 / Z	Q	λ
1	130	1.30	0.7692	0	100
2	130	1.30	0.7692	130	0
3	500	5.00	0.2000	0	100
4	500	5.00	0.2000	500	0
5	2000	20.00	0.0500	0	100
6	2000	20.00	0.0500	2000	0

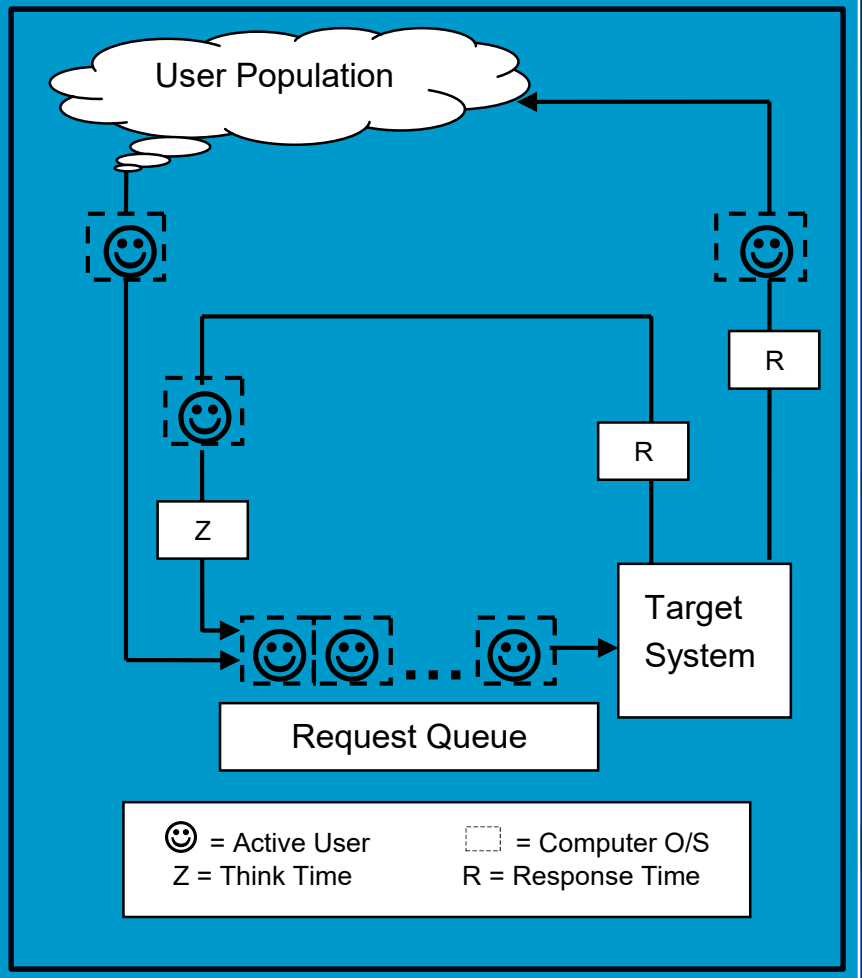
Where:
N = Number of Virtual Users
Z = Think Time Setting
Q = Number In Residence
λ = Arrival Rate

Virtual Users And Web Users

Steps Each 😐 Performs In Loop:

1. **Make A Web Request**
2. **Wait For & Time Response (R)**
3. **Sleep For Think Time (Z)**
4. **Wake Up & Wait On Run Queue**
5. **Compute And Return Results**





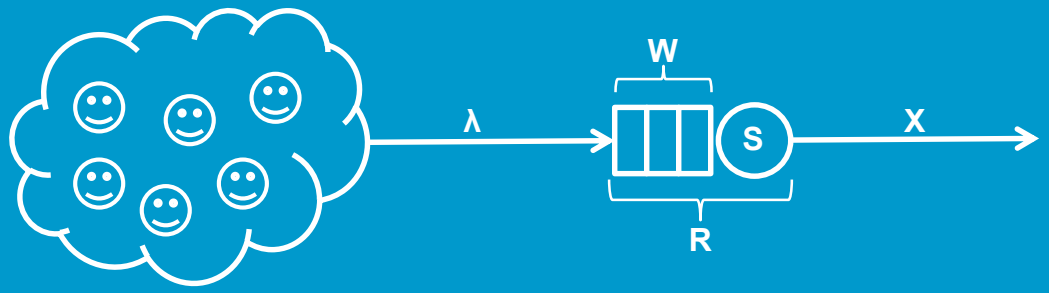
Web Testing Methodology

The Key Insights Come From Comparing The Load-test Simulation Models With The Appropriate Queue-theoretic Models

1. **Open Queue - The Number Of Users Generating Requests Is Unknown**
2. **Closed Queue - The Number Of Users Generating Requests Is Known And Fixed**

The Dynamics Of These Queues Differ Dramatically So They Need To Be Characterized In Completely Different Ways

Asynchronous Open Users



Where:
λ = Arrival Rate
W = Time Waiting In Line
S = Service Time
R = Residence Time
X = Service Rate
Q = Number In Residence

Residence Time: $R = W + S$

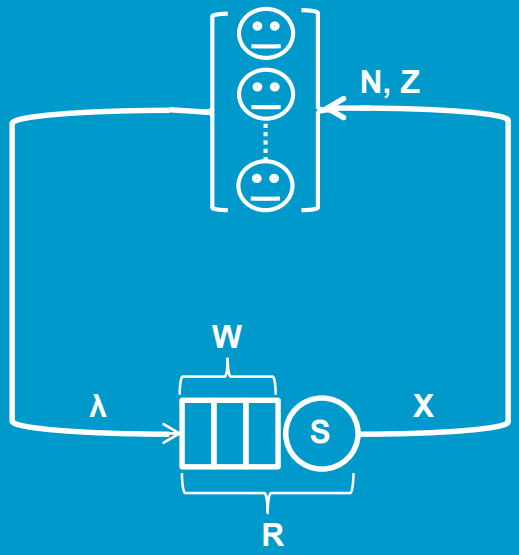
Number In Residence: $Q = \lambda R = \lambda W + \lambda S$

Steady State: $\lambda = X$

M/M/1 Queue: **Random Arrivals**
Exponentially Distributed Service Times

Synchronous Closed Users

Where:
N = Number of Users
Z = Think Time
λ = Arrival Rate
W = Time Waiting In Line
S = Service Time
R = Residence Time
X = Service Rate
Q = Number In Residence



Round Trip Time: $R_{TT} = R + Z$

Number Of Users: $N = \lambda R_{TT} = \lambda R + \lambda Z$

Number In Residence: $Q = \lambda R$

Steady State: $\lambda = \frac{N}{Z} - \frac{Q}{Z} = X$

M/M/1/N/N Queue: **Quasi-Random Arrivals**
Exponentially Distributed Service Times

Theoretical Comparison

Where:
λ = Arrival Rate
N = Number of Users
Z = Think Time
Q = Number In Residence

$$\lambda = \frac{N}{Z} - \frac{Q}{Z}$$

Constant Z virtual users

N	Z	N / Z	λ	Q	Q / Z
100	10	10	9.8	2.01	0.2
200	10	20	18.46	15.37	1.54
400	10	40	19.95	200.5	20.05
600	10	60	19.98	400.2	40.02
800	10	80	19.99	600.1	60.01
1000	10	100	19.99	800.1	80.01

Scaled Z web users

N	Z	N / Z	λ	Q	Q / Z
100	5	20	17.8	11.02	2.2
200	10	20	18.46	15.37	1.54
400	20	20	18.93	21.38	1.07
600	30	20	19.14	25.95	0.86
800	40	20	19.26	29.78	0.74
1000	50	20	19.34	33.15	0.66

$$\lambda_{sat} = \frac{1}{S_{max}}$$

$$\lambda_{rat} = \frac{N}{Z}$$

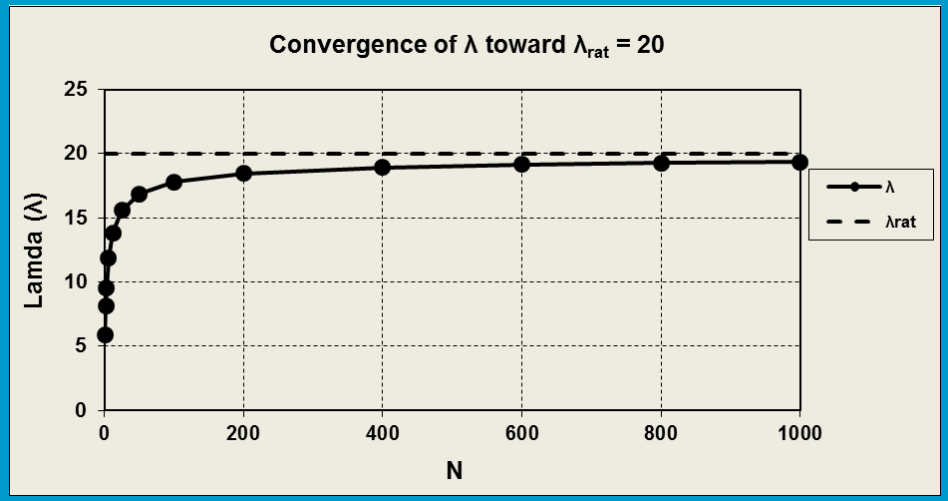
$$\lambda_{rat} \leq \lambda_{sat}$$

Web Testing Principle A

Principle A

To efficiently emulate web traffic the mean think-time, Z, in each of the N load generators should be scaled with N such that the ratio N / Z remains constant thereby ensuring the request rate λ approaches the soft bottleneck λ_{rat} .

Visualizing Principle A



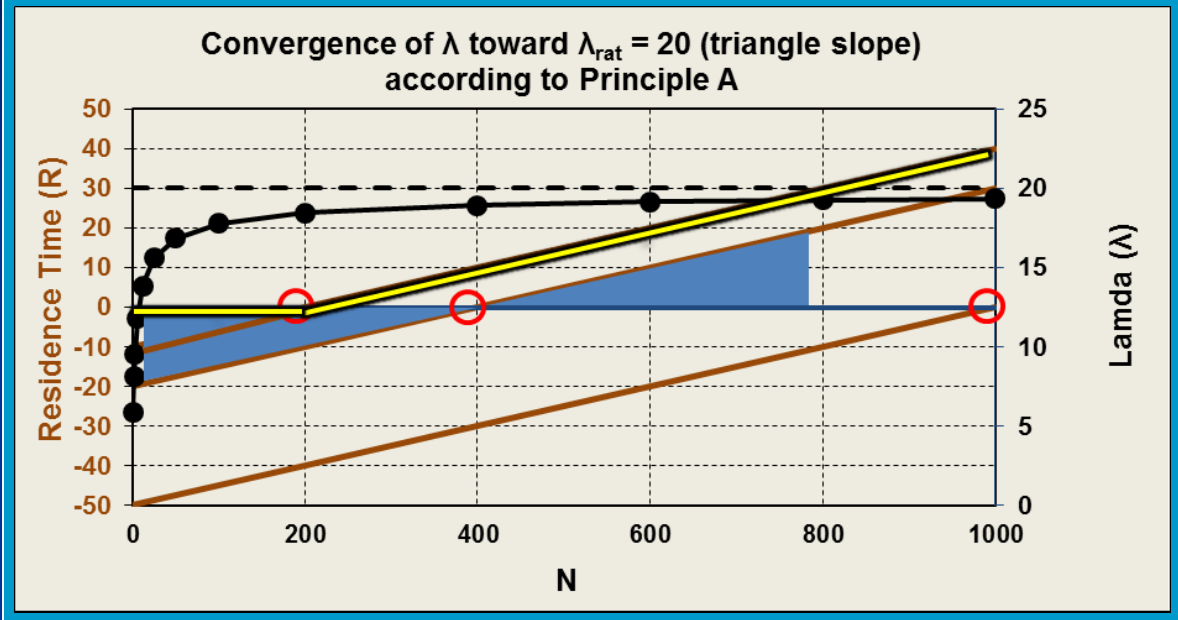
Row	N	Z	N / Z	λ	Q	Q / Z
1	1	0.05	20	5.88	0.71	14.12
2	2	0.10	20	8.17	1.18	11.83
3	3	0.15	20	9.55	1.57	10.45
4	6	0.30	20	11.83	2.45	8.17
5	12	0.60	20	13.85	3.69	6.15
6	25	1.25	20	15.59	5.51	4.41
7	50	2.50	20	16.86	7.85	3.14
8	100	5.00	20	17.80	11.02	2.20
9	200	10.00	20	18.46	15.37	1.54
10	400	20.00	20	18.93	21.38	1.07
11	600	30.00	20	19.14	25.95	0.86
12	800	40.00	20	19.26	29.78	0.74
13	1000	50.00	20	19.34	33.15	0.66

Where:
N = Number of Users
λ = Arrival Rate
λ_{rat} = Soft Bottleneck
R = Residence Time
Q = Number In Residence
Z = Think Time

$$\lambda = \frac{N}{Z} - \frac{Q}{Z}$$

$$\lambda_{rat} = \frac{N}{Z} = 20$$

Visualizing Principle A



$For\ \lambda = \lambda_{rat} = 20$

$R = .05\,N - Z$

Row	N	Z Lines 10	20	50
1	0	-10	-20	-50
2	200	0	-10	-40
3	400	10	0	-30
4	600	20	10	-20
5	800	30	20	-10
6	1000	40	30	0

Where:
N = Number of Users
λ = Arrival Rate
λ_{rat} = Soft Bottleneck
R = Residence Time
Q = Number In Residence
Z = Think Time

$$\lambda = \frac{N}{Z} - \frac{Q}{Z}$$

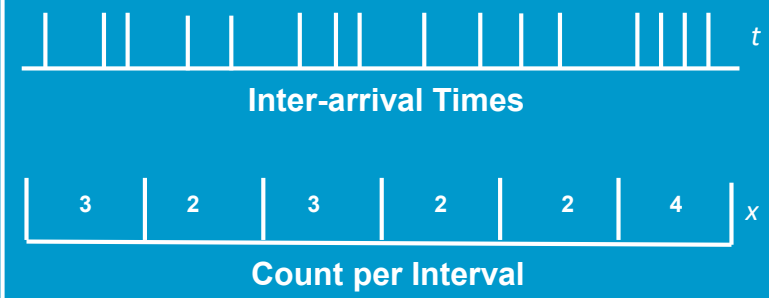
$$Q = \lambda R$$

$$R = \frac{N}{\lambda} - Z$$

Applying the Methodology Mimicking a Poisson Process

- The Number Of Arrivals In Each Fixed Time Interval Is A Random Variable That Is Poisson Distributed.
- The Time Periods Between Arrivals Are A Random Variable That Is Exponentially Distributed.
- Combining Arrivals From N Poisson Processes, Each Having Mean Request Rate λ, Produces An Aggregated Poisson Process Having A Mean Request Rate Nλ.
- Combining N non-Poisson Processes Asymptotically Approaches An Aggregated Poisson Process Provided N Is Large And Each λ Is Small. (Palm-Kintchine Theorem)

Visualizing a Poisson Process



$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}, s^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}, s = \sqrt{s^2}$$

Where:

$x_i = the\ i^{th} sample\ value$

$n = sample\ size$

$$f(t) = \frac{1}{\mu} e^{-\frac{t}{\mu}}$$

$$p(x) = \frac{\left(\frac{1}{\mu}\right)^x}{x!} e^{-\frac{1}{\mu}} \qquad x = 0, 1, \ldots$$

Where:

$\mu = mean\ time\ between\ arrivals, t$

$\frac{1}{\mu} = mean\ arrivals, x, per\ time\ interval$

$For\ f(t), \mu = mean = sdev = \sqrt{variance}$

$For\ p(x), \quad \frac{1}{u} = mean = variance$

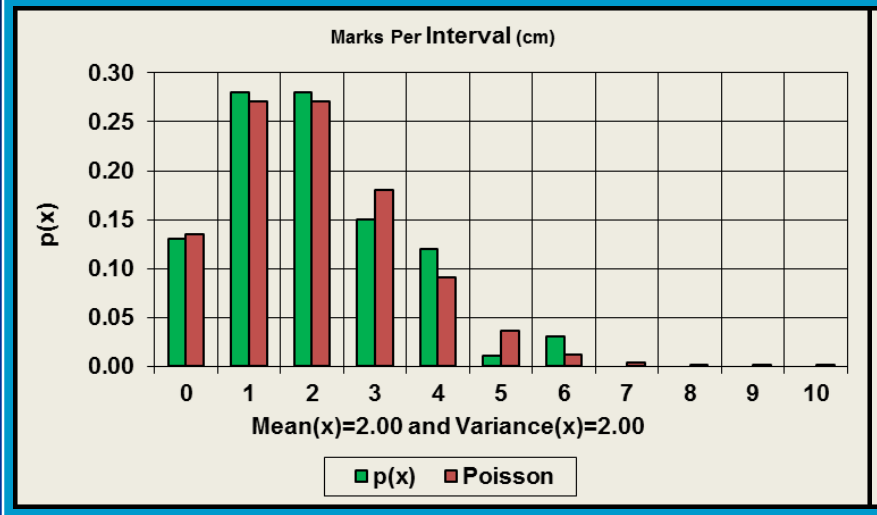
Exponential

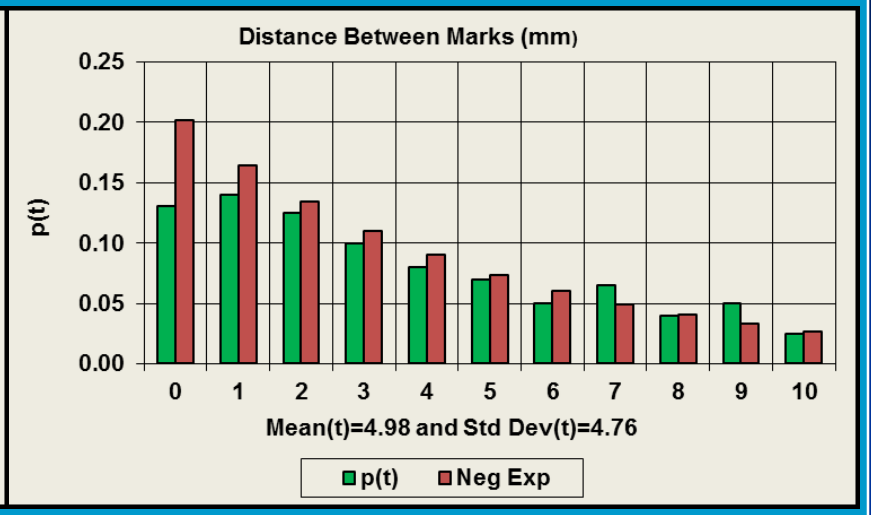
Poisson

Mean = Sdev

Mean = Variance

One Meter Ruler Illustration

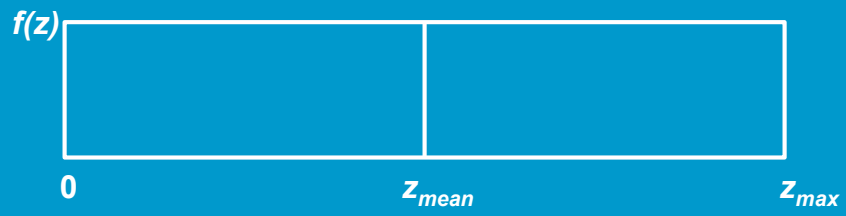




Sampling				Marks Per Interval (cm) - Poisson Distributed								Distance Between Marks (mm) - Neg Exp Distributed							
Sample		Ruler		Ruler		Histogram			Statistics		Theory	Ruler		Histogram			Statistics		Theory
Seq No.	Random Number	mm	cm	cm	cm freq	x	f(x)	p(x)	x*p(x)	x^2*p(x)	Poisson	mm sort	mm diff	t	f(t)	p(t)	t*p(t)	t^2*p(t)	Neg Exp
1	0.43181	431	43	0	4	0	13	0.130	0.00	0.00	0.1353	0	0	0	26	0.130	0.00	0.00	0.2010
2	0.58983	589	58	1	4	1	28	0.280	0.28	0.28	0.2707	0	0	1	28	0.140	0.14	0.14	0.1644
3	0.33311	333	33	2	2	2	28	0.280	0.56	1.12	0.2707	2	2	2	25	0.125	0.25	0.50	0.1345
4	0.21824	218	21	3	2	3	15	0.150	0.45	1.35	0.1804	2	0	3	20	0.100	0.30	0.90	0.1100
5	0.41661	416	41	4	2	4	12	0.120	0.48	1.92	0.0902	14	12	4	16	0.080	0.32	1.28	0.0900
6	0.16398	163	16	5	1	5	1	0.010	0.05	0.25	0.0361	16	2	5	14	0.070	0.35	1.75	0.0736
7	0.68319	683	68	6	2	6	3	0.030	0.18	1.08	0.0120	18	2	6	10	0.050	0.30	1.80	0.0602
8	0.97592	975	97	7	2	7	0	0.000	0.00	0.00	0.0034	19	1	7	13	0.065	0.46	3.19	0.0492
9	0.64382	643	64	8	0	8	0	0.000	0.00	0.00	0.0009	28	9	8	8	0.040	0.32	2.56	0.0403
10	0.21030	210	21	9	2	9	0	0.000	0.00	0.00	0.0002	28	0	9	10	0.050	0.45	4.05	0.0329
11	0.67479	674	67	10	2	10	0	0.000	0.00	0.00	0.0000	31	3	10	5	0.025	0.25	2.50	0.0269

Combining N non-Poisson Processes

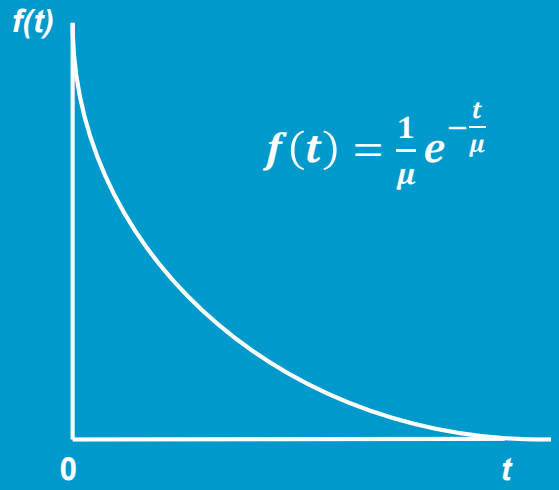
Uniform Distribution Of Z Values



$$f(z) = \frac{1}{z_{max}}$$

Where:
z_{mean} = mean think time

Exp Dist Of Times Between Requests - Superposition Of N Threads



$$f(t) = \frac{1}{\mu} e^{-\frac{t}{\mu}}$$

Where:
μ = mean time between requests
For f(t), mean = sdev

Web Testing Principle B

Principle B

Verify that the requests generated by applying Principle A (i.e., large N with low rate 1/Z) closely approximate a Poisson process by measuring the coefficient of variation of the inter-arrival periods and demonstrating that CoV ~ 1 within acceptable measurement error.

Where:
CoV = Sdev / Mean = 1 for an Exponential Distribution

Website Case Study

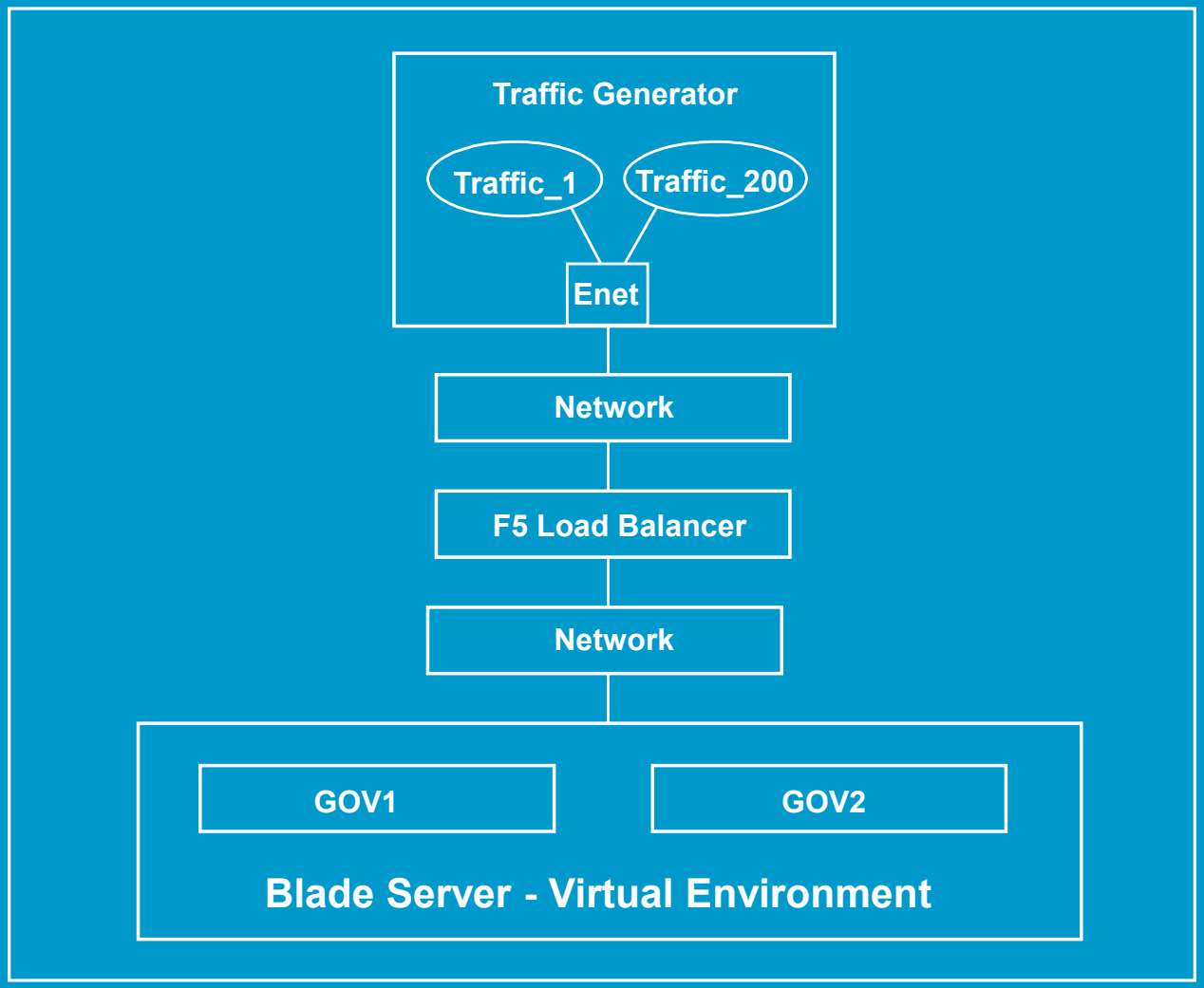
- **Government Web Site – WEB.gov**
  - Citizens obtain government statistics
  - Reconfiguring site from standalone servers to a virtualized load sharing environment

WEB.gov test objects	
Object	**Definition**
010_Home	Home page
012_Home_jpg	Background image
020_Depart	Department data
022_Depart_jpg	Department image
030_Demographics	Demographic data
040_Statistics	Government user data

Mapping between web pages and objects	
Web Page	**Objects**
Home	010_Home + 012_Home_jpg
Department	020_Depart + 022_Depart_jpg
Demographics	030_Demographics
Statistics	040_Statistics

- **Test Objectives**
  - How many web users will real WEB.gov support?
  - Do the virtual servers scale?
  - Is the load shared evenly across the virtual servers?

Test Configuration



Test Conditions

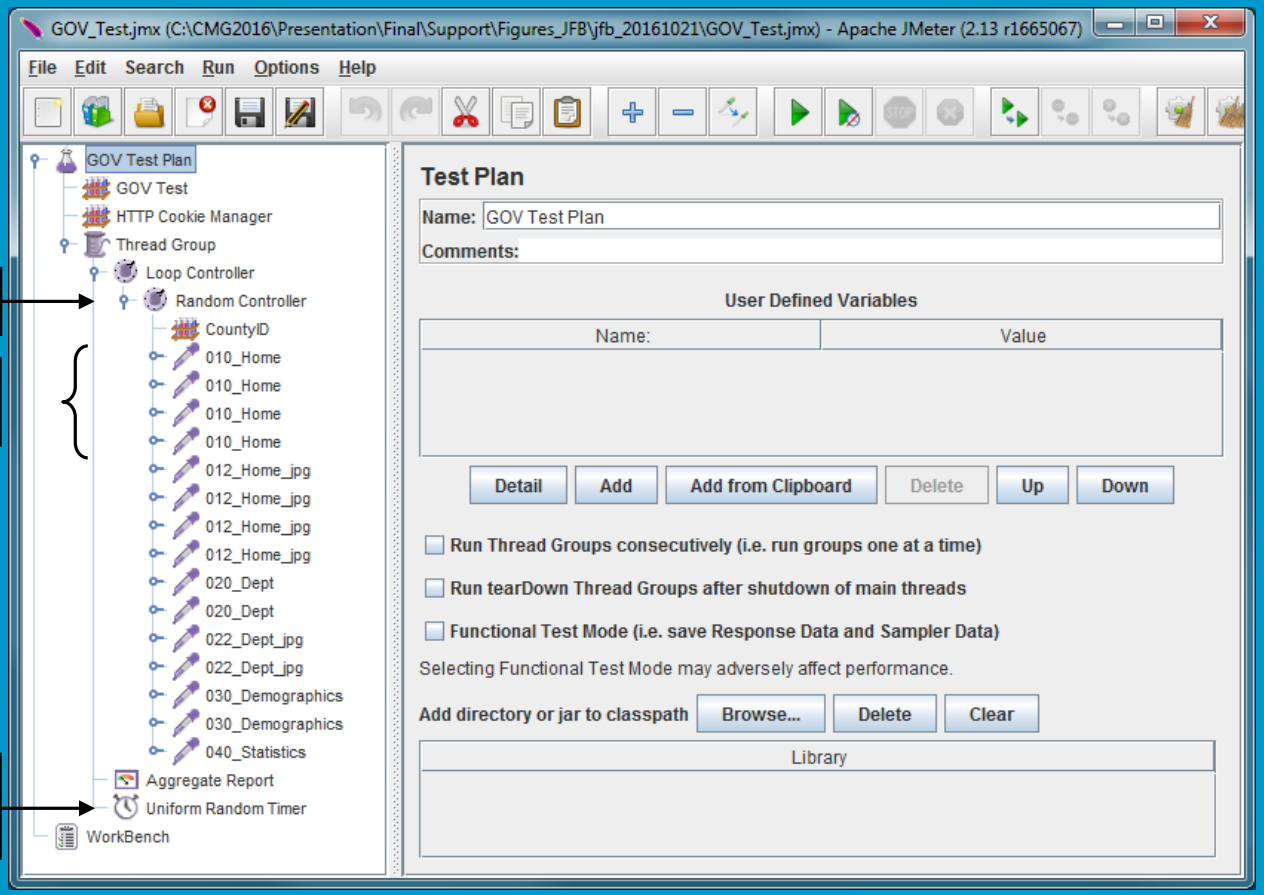
1. All Tests Are Run With 200 JMeter Threads
2. One Uniform Random Timer For Think Times
3. Load Increased By Reducing Mean Think Time
4. Web Objects Are Accessed In Random Order
5. Seven 25 Minutes Tests Performed
   - First Two And Last Three Minutes Excluded To Ensure Steady State
   - Test Start Times Based On 24 Hour Clock - 1800, 1830,1900, 1930, 2000, 2030, 2100

Test Script - JMeter

Random Order

Home Page Objects (4)

Uniform Random Timer



WEB.gov Simulated Web User Loads

JMeter Parameters				JMeter Metrics		Queue Metrics		Inter-arrival Time Statistics		
No	Run	N	Z (ms)	λ (obj/s)	R (ms)	Q (obj)	N (obj)	Mean	Sdev	CoV
1	1800	200	12500	15.91	53.17	0.85	199.72	62.81	61.75	0.98
2	1830	200	6250	31.73	52.74	1.67	199.99	31.52	31.59	1.00
3	1900	200	4200	46.79	54.06	2.53	199.05	21.37	21.54	1.01
4	1930	200	3250	60.26	59.64	3.59	199.44	16.60	16.78	1.01
5	2000	200	2500	77.56	74.97	5.81	199.71	12.89	13.22	1.03
6	2030	200	1563	118.03	132.67	15.66	200.14	8.47	9.12	1.08
7	2100	200	1000	159.16	253.38	40.33	199.49	6.28	7.37	1.17

Where:

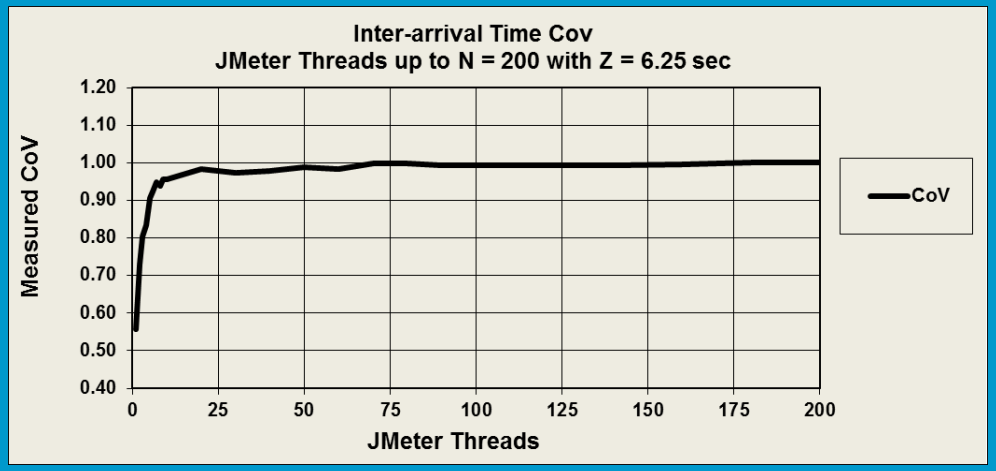
Q (obj) = λ x R

N (obj) = λ (R + Z)

CoV = Sdev / Mean of Inter-arrival Time

Inter-arrival Time Coefficient of Variation

(CoV By Thread Count For Run 1830)



JMeter Threads	Run 1830 Trans	λ (tps)	Inter-arrival Time (ms) Mean	Sdev	CoV
1	187	0.16	6391.96	3555.83	0.56
2	360	0.30	3323.87	2423.37	0.73
3	550	0.46	2175.62	1751.27	0.80
4	747	0.62	1603.27	1339.43	0.84
5	939	0.78	1275.45	1157.44	0.91
6	1122	0.94	1067.42	986.39	0.92
7	1317	1.10	909.37	862.32	0.95
8	1509	1.26	794.09	745.60	0.94
9	1694	1.41	707.37	676.15	0.96
10	1875	1.56	639.65	611.60	0.96
20	3846	3.21	311.90	306.66	0.98
30	5776	4.81	207.72	202.35	0.97
40	7685	6.41	156.12	152.82	0.98
50	9566	7.97	125.42	124.02	0.99
60	11425	9.52	105.02	103.31	0.98
70	13329	11.11	90.02	89.84	1.00
80	15251	12.71	78.67	78.64	1.00
90	17175	14.31	69.86	69.45	0.99
100	19089	15.91	62.86	62.55	1.00
120	22911	19.09	52.37	52.09	0.99
140	26693	22.24	44.95	44.64	0.99
160	30454	25.38	39.40	39.25	1.00
180	34243	28.54	35.04	35.11	1.00
200	38072	31.73	31.52	31.59	1.00

Where:
N = Number of Threads
Z = Think Time
λ = Arrival Rate
CoV = Sdev / Mean of Inter-arrival Time

WEB.gov Estimated Internet Web Users

Number Of Users:

$$N' = \lambda (R + Z')$$

JMeter Load Generator Statistics					Conversion pgs/obj= 3/5		Nominal Internet Think Time, Z' (ms)				
							10,000	20,000	30,000	40,000	50,000
No	Run	λ (obj/s)	R (ms)	Q (obj)	λ (pgs/s)	Q (pgs)	Calculated Internet Web Users, N'				
1	1800	15.91	53.17	0.85	9.55	0.51	95.97	191.43	286.89	382.35	477.81
2	1830	31.73	52.74	1.67	19.04	1.00	191.38	381.76	572.14	762.52	952.90
3	1900	46.79	54.06	2.53	28.07	1.52	282.26	563.00	843.74	1124.48	1405.22
4	1930	60.26	59.64	3.59	36.16	2.16	363.72	725.28	1086.84	1448.40	1809.96
5	2000	77.56	74.97	5.81	46.54	3.49	468.85	934.21	1399.57	1864.93	2330.29
6	2030	118.03	132.67	15.66	70.82	9.40	717.58	1425.76	2133.94	2842.12	3550.30
7	2100	159.16	253.38	40.33	95.50	24.20	979.16	1934.12	2889.08	3844.04	4799.00

95.50 x (.25338 + 30) = 2889.08

Where:
N' = Number of Internet Web Users
λ (pgs/s) = Arrival Rate in Pages/Sec
R = Residence (Response) Time
Z' = Nominal Internet Think Time

WEB.gov Estimated Internet Web Users

Number Of Users:

$$N' = \lambda\,(R + Z')$$

JMeter Load Generator Statistics					Conversion pgs/obj=	3/5	Nominal Internet Think Time, Z' (ms)				
							10,000	20,000	30,000	40,000	50,000
No	Run	λ (obj/s)	R (ms)	Q (obj)	λ (pgs/s)	Q (pgs)	Calculated Internet Web Users, N'				
1	1800	15.91	53.17	0.85	9.55	0.51	95.97	191.43	286.89	382.35	477.81
2	1830	31.73	52.74	1.67	19.04	1.00	191.38	381.76	572.14	762.52	952.90
3	1900	46.79	54.06	2.53	28.07	1.52	282.26	563.00	843.74	1124.48	1405.22
4	1930	60.26	59.64	3.59	36.16	2.16	363.72	725.28	1086.84	1448.40	1809.96
5	2000	77.56	74.97	5.81	46.54	3.49	468.85	934.21	1399.57	1864.93	2330.29
6	2030	118.03	132.67	15.66	70.82	9.40	717.58	1425.76	2133.94	2842.12	3550.30
7	2100	159.16	253.38	40.33	95.50	24.20	979.16	1934.12	2889.08	3844.04	4799.00

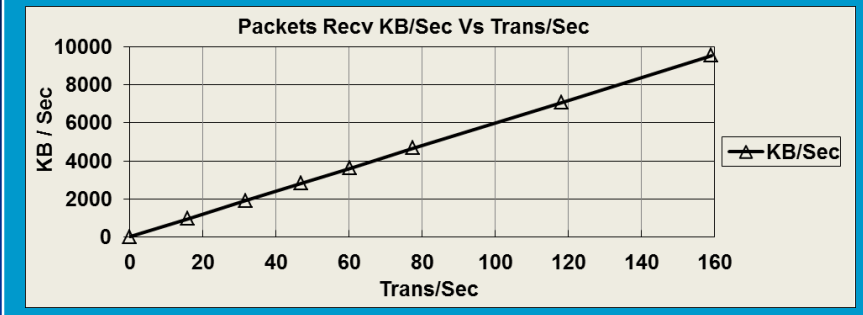
$$\lambda = \frac{N'}{Z'} - \frac{Q}{Z'}$$

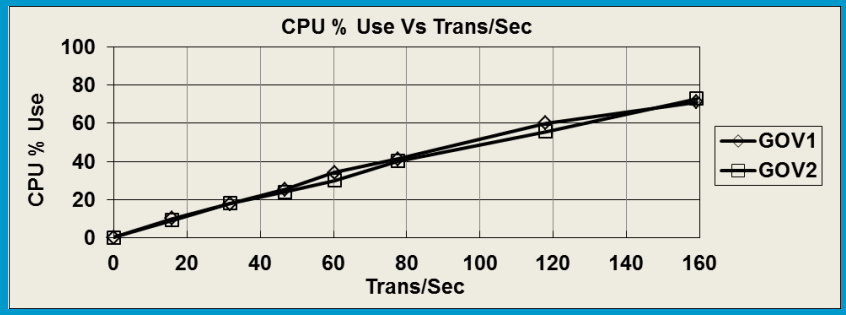
Where:
N' = Number of Internet Web Users
λ (pgs/s) = Arrival Rate in Pages/Sec
R = Residence (Response) Time
Z' = Nominal Internet Think Time
Q = Number In Residence

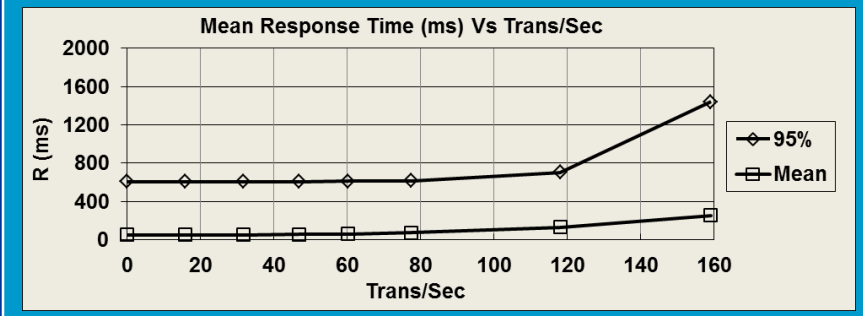
$$\lambda_{10} = \frac{363.72}{10} - \frac{2.16}{10} = 36.16$$

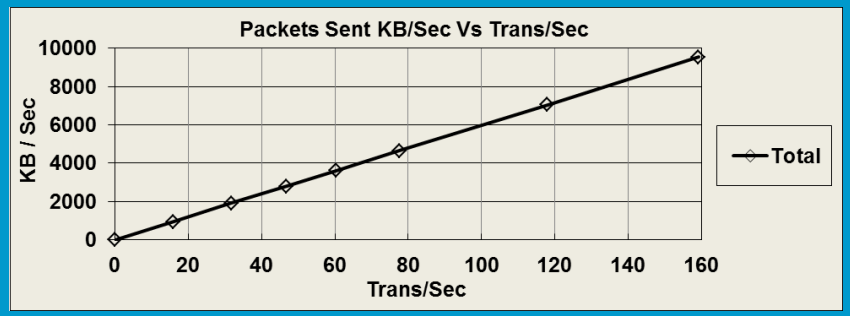
$$\lambda_{30} = \frac{1086.84}{30} - \frac{2.16}{30} = 36.16$$

Gov Test - SUT Scalability









JMeter Load Generator Statistics				
		R (ms)		Pkt Recv
Test Run	Trans/Sec	Mean	95%	KB/Sec
0	0.00	53	609	0
1800	15.91	53	609	956
1830	31.73	53	607	1910
1900	46.79	54	607	2811
1930	60.26	60	611	3625
2000	77.56	75	617	4670
2030	118.03	133	705	7080
2100	159.16	253	1440	9552

Target Servers GOV1 and GOV2				
CPU % Use		Pkt Sent (KB/Sec)		
GOV1	GOV2	GOV1	GOV2	Total
0	0	0	0	0
10.05	9.14	477	476	953
17.89	17.95	956	953	1909
25.27	24.10	1425	1370	2795
34.07	30.03	1881	1729	3610
41.23	40.14	2338	2313	4651
60.13	55.64	3684	3360	7044
71.26	72.89	4682	4856	9538

Conclusions

1. Conventional Tools Simulate A Fixed Set Of Users While Web Traffic Is Produced By A Dynamic User Population.
2. Our Methodology For Adapting Conventional Tools To Web Traffic Is Based On Two Fundamental Principles:
   - Principle A – Shows How To Scale Parameters N and Z To Produce A Poisson Process.
   - Principle B – Confirms These Parameter Settings Actually Mimic A Poisson Process.
3. Real Web Traffic Has Some Synchronization Properties But They Are Weak So Properly Implemented Conventional Tools Are Better Than Explicitly Asynchronous Tools For Mimicking Web Traffic.

Conclusions (cont)

4. **The Case Study Provides A Practical Example Of How Our Methodology Is Implemented And Highlightes Some Of Its Nuances:**
   - **It Illustrates That An Estimation Of Actual N' Internet Users Rather Than The Number Of Test Generator Threads May Need To Be Reported.**
   - **It Provides Insights Into How Many Load Tool Threads Are Needed To Mimic A Poisson Process**
   - **The Statistical Analysis Tools Used To Produce These Results, "web-generator-toolkit", Is Available Online (Free) With This Case Study As The Example.**

QUESTIONS

? ? ?