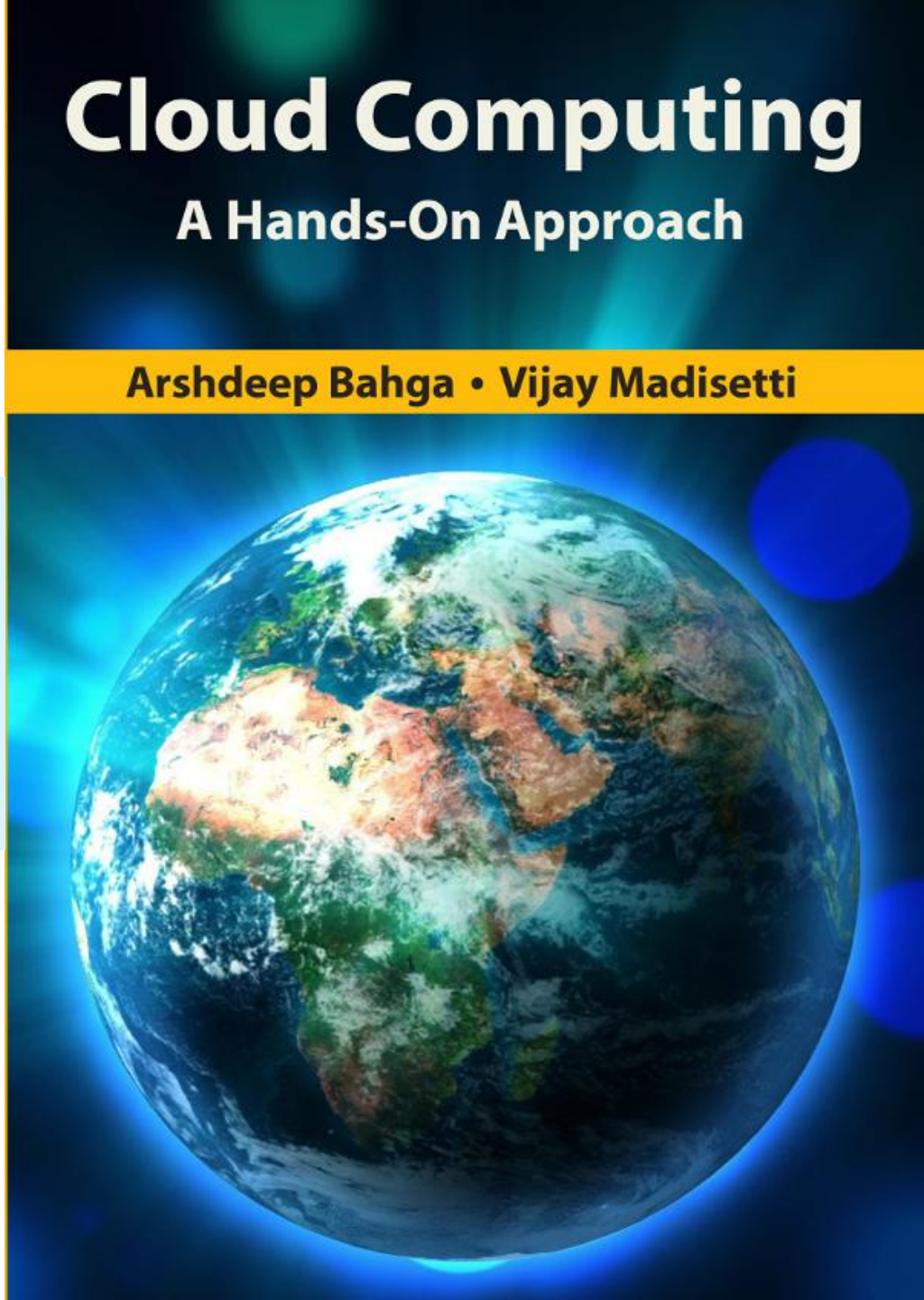
Chapter 11

Cloud Application Benchmarking & Tuning



Bahga & Madisetti, © 2014

Outline

- Cloud application workload characteristics
- Performance metrics for cloud applications
- Cloud application testing
- Performance testing tools
- Load test and bottleneck detection case study

Bahga & Madisetti, © 2014

Benchmarking

- Benchmarking of cloud applications is important or the following reasons:
  - Provisioning and capacity planning
    - The process of provisioning and capacity planning for cloud applications involves determining the amount of computing, memory and network resources to provision for the application.
    - Benchmarking can help in comparing alternative deployment architectures and choosing the best and most cost effective deployment architecture that can meet the application performance requirements.
  - Ensure proper utilization of resources
    - Benchmarking can help in determining the utilization of computing, memory and network resources for applications and identify resources which are either under-utilized or over-provisioned and hence save deployments costs.
  - Market readiness of applications
    - Performance of an application depends on the characteristics of the workloads it experiences. Different types of workloads can dead to different performance for the same application.
    - To ensure the market readiness of an application it is important to model all types of workloads the application can experience and benchmark the application with such workloads.

Bahga & Madisetti, © 2014

Cloud Application Benchmarking - Steps

- Trace Collection/Generation
  - The first step in benchmarking cloud applications is to collect/generate traces of real application workloads. For generating a trace of workload, the application is instrumented to log information such as the requests submitted by the users, the time-stamps of the requests, etc.
- Workload Modeling
  - Workload modeling involves creation of mathematical models that can be used for generation of synthetic workloads.
- Workload Specification
  - Since the workload models of each class of cloud computing applications can have different workload attributes, a Workload Specification Language (WSL) is often used for specification of application workloads. WSL can provide a structured way for specifying the workload attributes that are critical to the performance of the applications. WSL can be used by synthetic workload generators for generating workloads with slightly varying the characteristics.
- Synthetic Workload Generation
  - Synthetic workloads are used for benchmarking cloud applications. An important requirement for a synthetic workload generator is that the generated workloads should be representative of the real workloads.

Bahga & Madisetti, © 2014

Synthetic Workload Generation Approaches

- Empirical approach
  - In this approach traces of applications are sampled and replayed to generate the synthetic workloads.
  - The empirical approach lacks flexibility as the real traces obtained from a particular system are used for workload generation which may not well represent the workloads on other systems with different configurations and load conditions.

- Analytical approach
  - Uses mathematical models to define the workload characteristics that are used by a synthetic workload generator.
  - Analytical approach is flexible and allows generation of workloads with different characteristics by varying the workload model attributes.
  - With the analytical approach it is possible to modify the workload model parameters one at a time and investigate the effect on application performance to measure the application sensitivity to different parameters.

Bahga & Madisetti, © 2014

User Emulation vs Aggregate Workloads

The commonly used techniques for workload generation are:

- User Emulation
  - Each user is emulated by a separate thread that mimics the actions of a user by alternating between making requests and lying idle.
  - The attributes for workload generation in the user emulation method include think time, request types, inter-request dependencies, for instance.
  - User emulation allows fine grained control over modeling the behavioral aspects of the users interacting with the system under test, however, it does not allow controlling the exact time instants at which the requests arrive the system.
- Aggregate Workload Generation:
  - Allows specifying the exact time instants at which the requests should arrive the system under test.
  - However, there is no notion of an individual user in aggregate workload generation, therefore, it is not possible to use this approach when dependencies between requests need to be satisfied.
  - Dependencies can be of two types inter-request and data dependencies.
  - An inter-request dependency exists when the current request depends on the previous request, whereas a data dependency exists when the current requests requires input data which is obtained from the response of the previous request.

Bahga & Madisetti, © 2014

Workload Characteristics

- Session
  - A set of successive requests submitted by a user constitute a session.
- Inter-Session Interval
  - Inter-session interval is the time interval between successive sessions.
- Think Time
  - In a session, a user submits a series of requests in succession. The time interval between two successive requests is called think time.
- Session Length
  - The number of requests submitted by a user in a session is called the session length.
- Workload Mix
  - Workload mix defines the transitions between different pages of an application and the proportion in which the pages are visited.

Bahga & Madisetti, © 2014

Application Performance Metrics

The most commonly used performance metrics for cloud applications are:

- Response Time
  - Response time is the time interval between the moment when the user submits a request to the application and the moment when the user receives a response.

- Throughput
  - Throughput is the number of requests that can be serviced per second.

Bahga & Madisetti, © 2014

Considerations for Benchmarking Methodology

- Accuracy
  - Accuracy of a benchmarking methodology is determined by how closely the generated synthetic workloads mimic the realistic workloads.
- Ease of Use
  - A good benchmarking methodology should be user friendly and should involve minimal hand coding effort for writing scripts for workload generation that take into account the dependencies between requests, workload attributes, for instance.
- Flexibility
  - A good benchmarking methodology should allow fine grained control over the workload attributes such as think time, inter-session interval, session length, workload mix, for instance, to perform sensitivity analysis.
  - Sensitivity analysis is performed by varying one workload characteristic at a time while keeping the others constant.
- Wide Application Coverage
  - A good benchmarking methodology is one that works for a wide range of applications and not tied to the application architecture or workload types.

Bahga & Madisetti, © 2014

Types of Tests

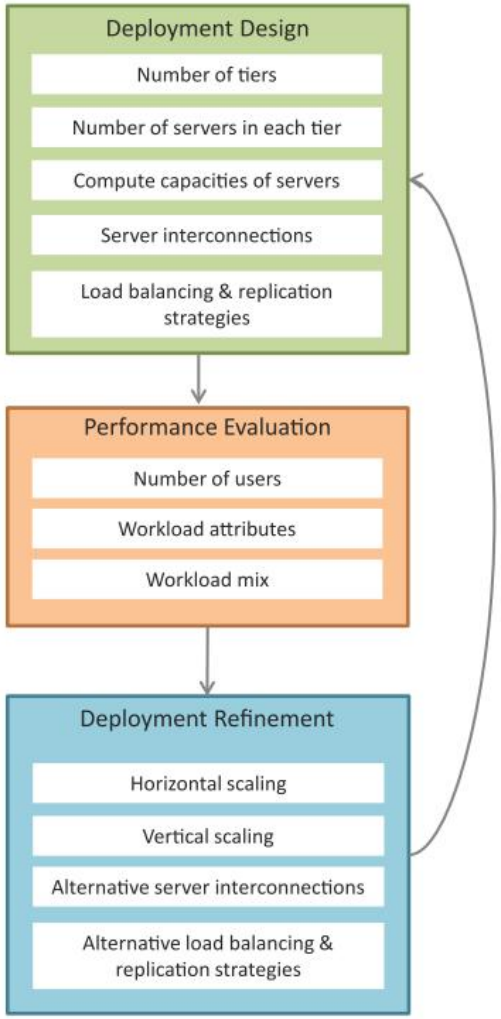
- Baseline Tests
  - Baseline tests are done to collect the performance metrics data of the entire application or a component of the application.
  - The performance metrics data collected from baseline tests is used to compare various performance tuning changes which are subsequently made to the application or a component.
- Load Tests
  - Load tests evaluate the performance of the system with multiple users and workload levels that are encountered in the production phase.
  - The number of users and workload mix are usually specified in the load test configuration.
- Stress Tests
  - Stress tests load the application to a point where it breaks down.
  - These tests are done to determine how the application fails, the conditions in which the application fails and the metrics to monitor which can warn about impending failures under elevated workload levels.
- Soak Tests
  - Soak tests involve subjecting the application to a fixed workload level for long periods of time.
  - Soak tests help in determining the stability of the application under prolonged use and how the performance changes with time.

Bahga & Madisetti, © 2014

Deployment Prototyping

- Deployment prototyping can help in making deployment architecture design choices.
- By comparing performance of alternative deployment architectures, deployment prototyping can help in choosing the best and most cost effective deployment architecture that can meet the application performance requirements.

- Deployment design is an iterative process that involves the following steps:
- Deployment Design
  - Create the deployment with various tiers as specified in the deployment configuration and deploy the application.
- Performance Evaluation
  - Verify whether the application meets the performance requirements with the deployment.
- Deployment Refinement
  - Deployments are refined based on the performance evaluations. Various alternatives can exist in this step such as vertical scaling, horizontal scaling, for instance.

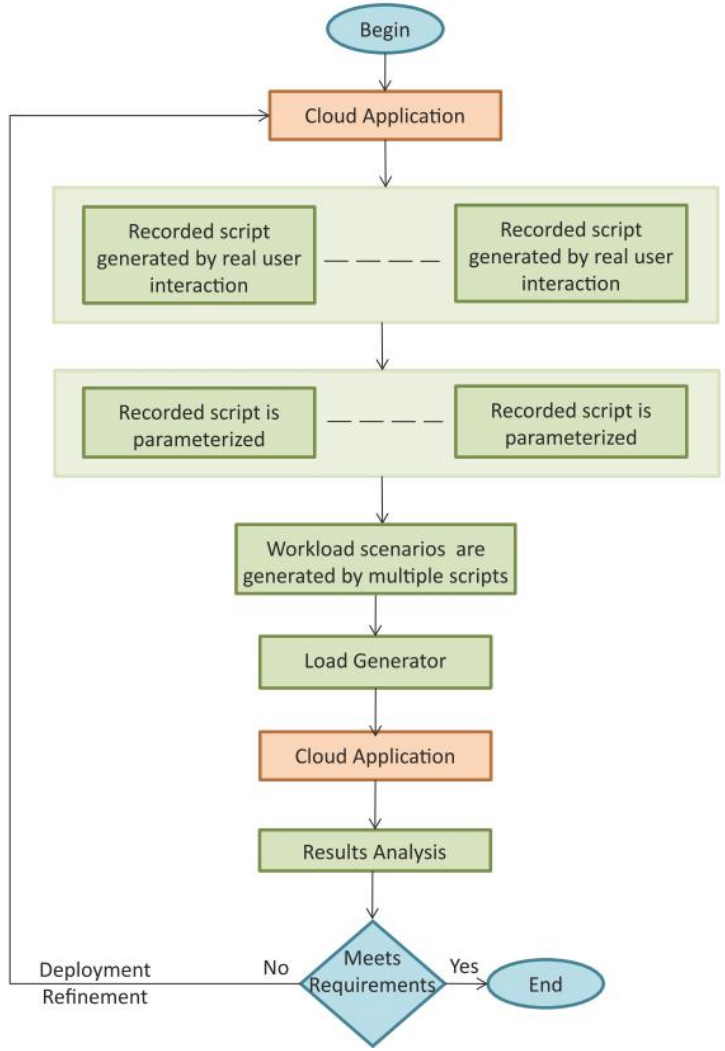


Bahga & Madisetti, © 2014

Performance Evaluation Workflow

Semi-Automated Workflow (Traditional Approach)

- In traditional approach to capture workload characteristics, a real user's interactions with a cloud application are first recorded as virtual user scripts.
- The recorded virtual user scripts then are parameterized to account for randomness in application and workload parameters.
- Multiple scripts have to be recorded to create different workload scenarios. This approach involves a lot of manual effort.
- To add new specifications for workload mix and new requests, new scripts need to be recorded and parameterized.
- Traditional approaches which are based on manually generating virtual user scripts by interacting with a cloud application, are not able to generate synthetic workloads which have the same characteristics as real workloads.
- Traditional approaches do now allow rapidly comparing various deployment architectures.

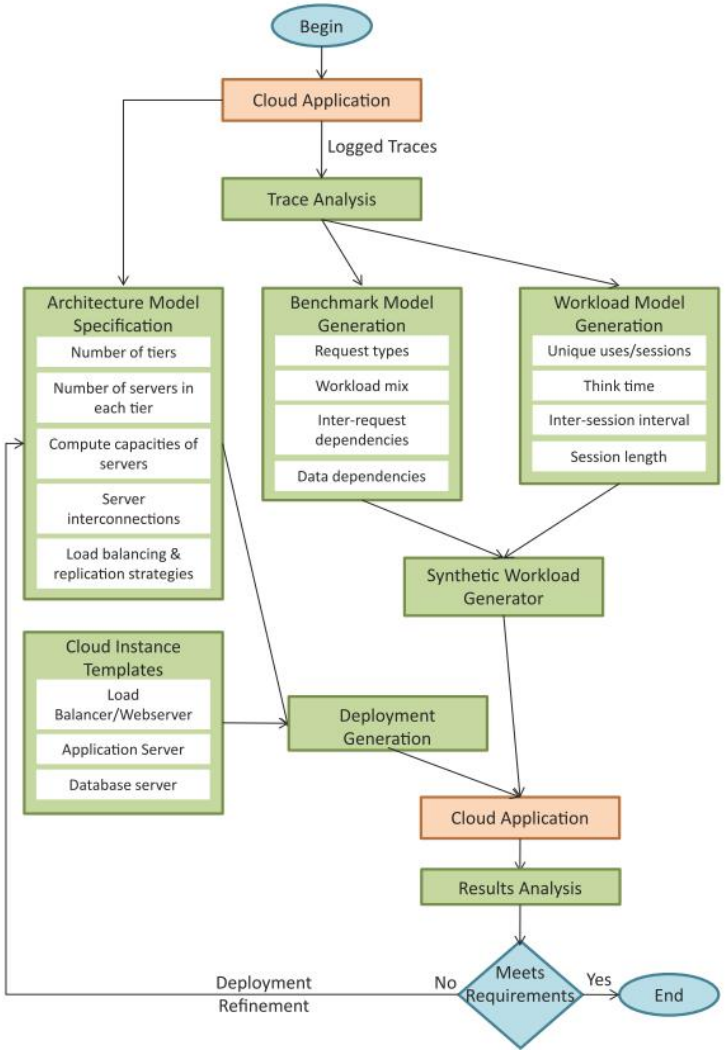


Bahga & Madisetti, © 2014

Performance Evaluation Workflow

Fully-Automated Workflow (Modern Approach)

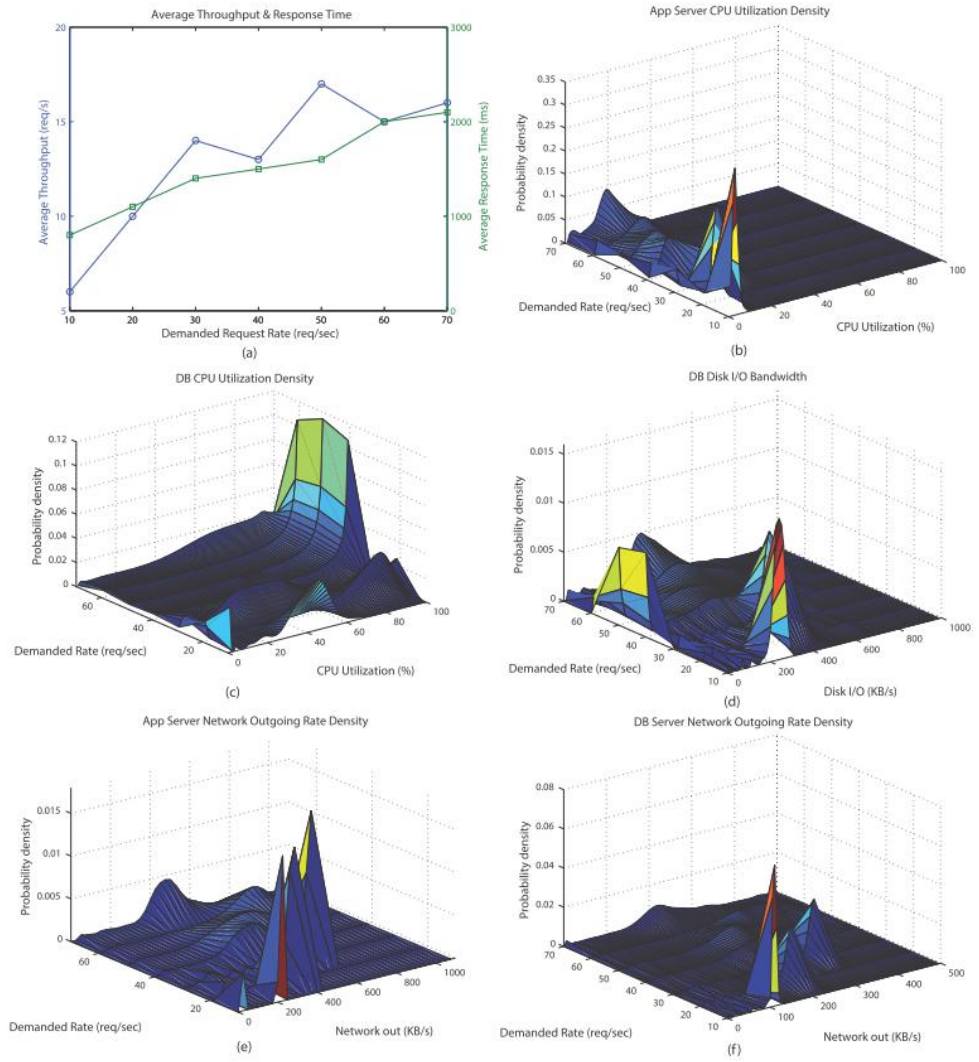
- In the automated approach real traces of a multi-tier application which are logged on web servers, application servers and database servers are analyzed to generate benchmark and workload models that capture the cloud application and workload characteristics.
- A statistical analysis of the user requests in the real traces is performed to identify the right distributions that can be used to model the workload model attributes.
- Real traces are analyzed to generate benchmark and workload models.
- Various workload scenarios can be created by changing the specifications of the workload model.
- Since real traces from a cloud application are used to capture workload and application characteristics into workload and benchmark models, the generated synthetic workloads have the same characteristics as real workloads.
- An architecture model captures the deployment configurations of multi-tier applications.



Bahga & Madisetti, © 2014

Benchmarking Case Study

- Fig (a) shows the average throughput and response time. The observed throughput increases as demanded request rate increases. As more number of requests are served per second by the application, the response time also increases. The observed throughput saturates beyond a demanded request rate of 50 req/sec.
- Fig (b) shows the CPU usage density of one of the application servers. This plot shows that the application server CPU is non-saturated resource.
- Fig (c) shows the database server CPU usage density. From this density plot we observe that the database CPU spends a large percentage of time at high utilization levels for demanded request rate more than 40 req/sec.
- Fig (d) shows the density plot of the database disk I/O bandwidth.
- Fig (e) shows the network out rate for one of the application servers
- Fig (f) shows the density plot of the network out rate for the database server. From this plot we observe a continuous saturation of the network out rate around 200 KB/s.
- Analysis
  - Throughput continuously increases as the demanded request rate increases from 10 to 40 req/sec. Beyond 40 req/sec demanded request rate, we observe that throughput saturates, which is due to the high CPU utilization density of the database server CPU. From the analysis of density plots of various system resources we observe that the database CPU is a system bottleneck.



Bahga & Madisetti, © 2014

Further Reading

- D. Mosberger, T. Jin, httperf: A Tool for Measuring Web Server Performance, ACM Performance Evaluation Review, Vol. 26, No. 3, pp. 31-37, 1998.
- P. Barford, M.E. Crovella, Generating representative Web workloads for network and server performance evaluation, In SIGMETRICS 98, pages 151-160, 1998.
- D. Krishnamurthy, J.A. Rolia, and S. Majumdar, SWAT: A Tool for Stress Testing Session-based Web Applications, in Proc. Int. CMG Conference, pp.639-649, 2003.
- HP LoadRunner, http://www8.hp.com/us/en/software/softwareproduct.htmlficompURI=tcm:245-935779, 2012.
- A. Bahga, V. Madisetti, Performance Evaluation Approach for Multi-tier Cloud Applications, Journal of Software Engineering and Applications, Vol. 6, No. 2, pp. 74-83, Mar 2013.
- SPECweb99, http://www.spec.org/osg/web99, 2012.
- A. Mahanti, C. Williamson, D. Eager, Traffic Analysis of a Web Proxy Caching Hierarchy, IEEE Network, vol. 14, no. 3, pp. 16-23, 2000.
- S. Manley, M. Seltzer, M. Courage, A Self-Scaling and Self-Configuring Benchmark for Web Servers, Proceedings of the ACM SIGMETRICS Conference, Madison, WI, 1998.
- Webjamma, http://www.cs.vt.edu/ chitra/webjamma.html, 2012.
- G. Abdulla, Analysis and modeling of world wide web traffic, PhD thesis, Chair-Edward A. Fox, 1998.
- M. Crovella, A. Bestavros, Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes, IEEE/ACM Trans. Networking, vol. 5, no. 6, pp. 835-846, 1997.
- D. Garcia, J. Garcia, TPC-W E-Commerce Benchmark Evaluation, IEEE Computer, pages 4248, 2003.
- RUBiS, http://rubis.ow2.org, 2012. [15]
- TPC-W, http://jmob.ow2.org/tpcw.html
- httperfpy, https://github.com/jmervine/httperfpy

Bahga & Madisetti, © 2014