World Wide Web 2003
Budapest, May 20, 2003

Benchmarking of Locally and Geographically Distributed Web-Server Systems

Michele Colajanni°, Mauro Andreolini*, Valeria Cardellini*
°Dipartimento di Ingegneria dell'Informazione
Università di Modena, Italy

***Dipartimento di Informatica, Sistemi e Produzione**
Università di Roma "Tor Vergata", Italy

Web-based services have new requirements

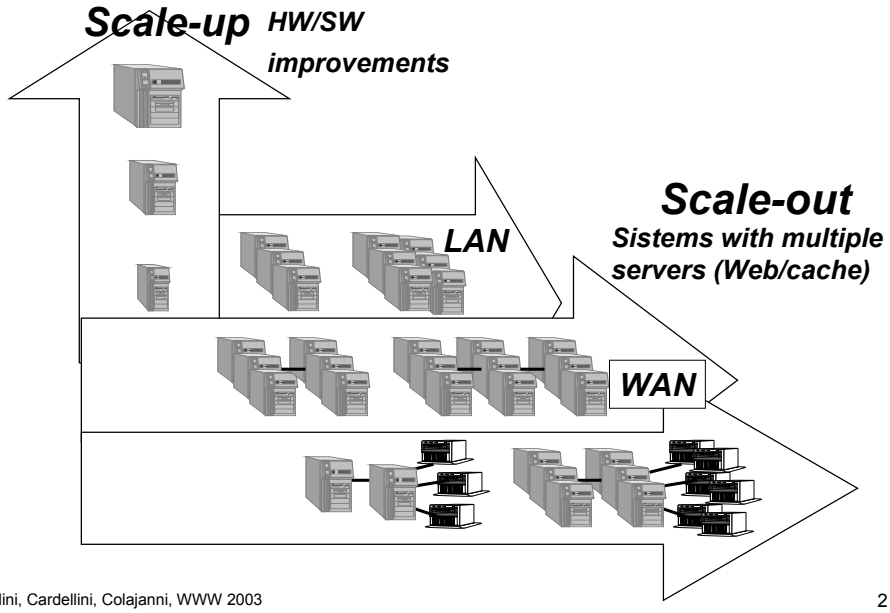
First generation

- An economic channel for not critical information
- 90 percent of information represented by text and some images
- Occasional maintenance and updating
- Highly variable performance
- No guarantee on availability
- Security not important

Second generation

- An important channel for critical information
- Always larger percentage of dynamic and personalized content
- Direct or indirect (say, publicity) costs
- The quality of services provided by a Web site changes user's view on a company
- *Differentiated services*
- *Content adaptation*

Pressure on Web-based architectures



Pressure on performance evaluation models and tools

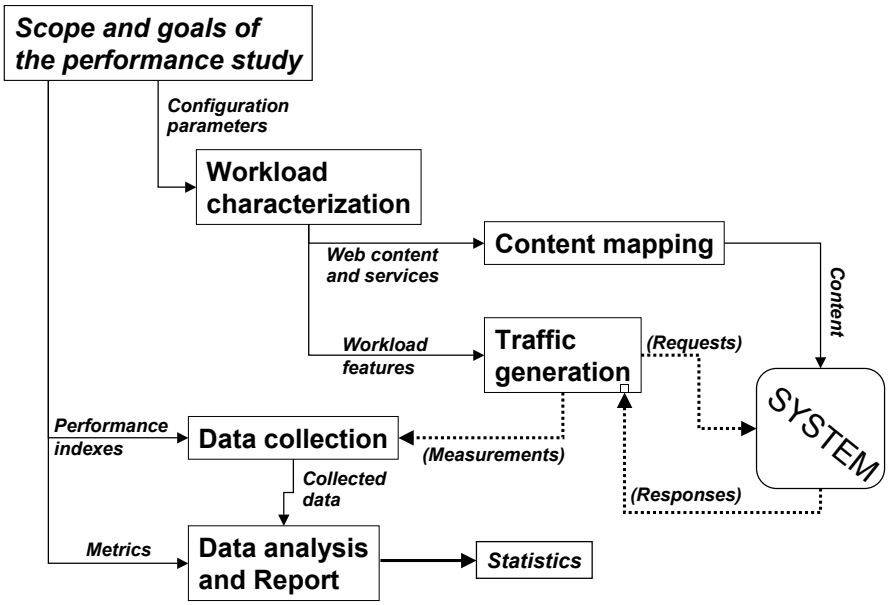
- **The significance of designing for scalability and high performance cannot be understated**
- The success of **any high performance Web-based service** depends on the ability to design and implement a system that yields: high performance, availability and reliability expected to support the business objectives for revenue and customer satisfaction

Benchmark (*def.*)

- ***Dictionary definition***
  - A point of reference by which something can be measured
- ***Computer industry definitions***
  - A set of conditions against which a product or system is measured
  - A set of performance criteria which a product is expected to meet
- ***"Old hacker's saying"***
  - "In the computer industry, there are three kinds of lies: lies, damn lies, and benchmarks."

Three lessons: - We need a *point of reference*
- Benchmark is related to *measurement*
- Benchmark may be affected by (*un*)*intentional errors*

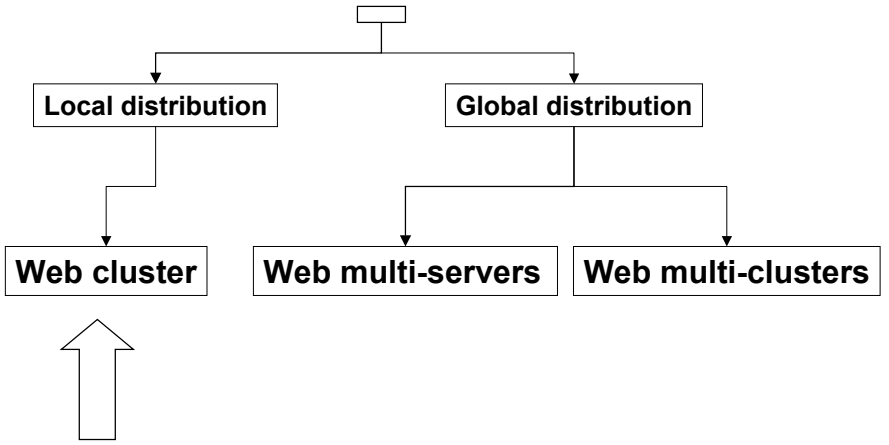
Main components



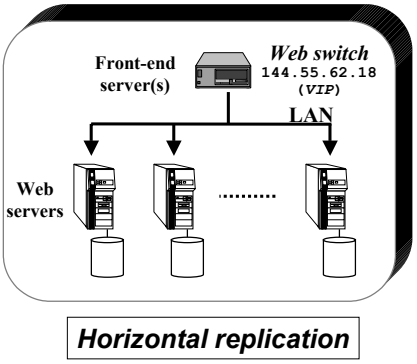
There is a lot of expectations, but …

- **Current benchmarks for Web systems based on a single server are affected by many limits**
  - Focus on specific workloads
  - Focus on micro-optimizations (no overall performance)
  - Unrealistic (no network delays and errors, correlations between emulated clients, …)
- **Benchmarking of distributed Web-server systems is even a bigger problem**
  - Content mapping on server nodes
  - Sustained traffic generation
  - Data collection
  - When necessary to test geographically distributed systems: supports for request routing mechanisms

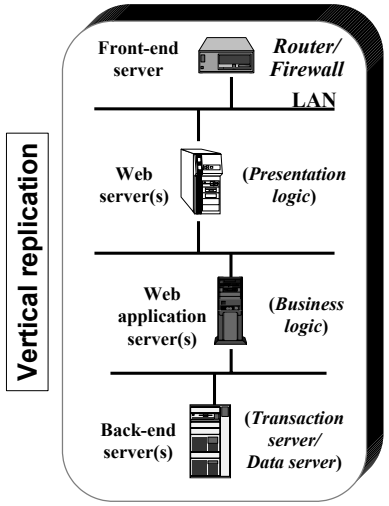
Scalable Web systems (multiple servers)



Web cluster architectures



Horizontal replication

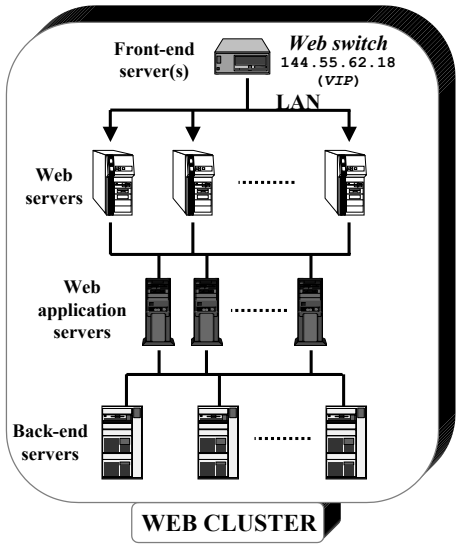


Vertical replication

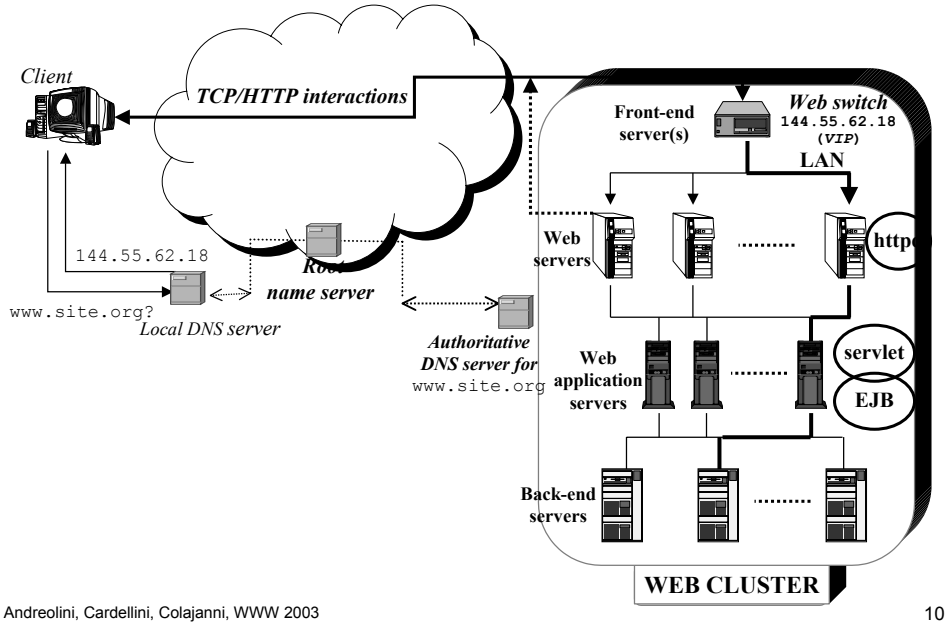
With all possible combinations …

Web cluster architecture:
A myriad of (*complex*) technologies

- Network/OS technology
- Web server technology
- Middleware technology
- Database technology



Interaction with a Web cluster architecture



Tutorial goals

- **Main issues to be considered when analyzing the performance of high-performance Web server systems**
- **Critical analysis of the most popular (academic/research) benchmarking and load generator tools with respect to main components**
- **A look to the future needs**

Tutorial's overview and presentations

- **Part 1** (*Colajanni*)
  - **Introduction and motivations**
  - **Web systems with multiple servers**
  - **Goals of a benchmarking study**
- **Part 2** (*Cardellini*)
  - **Workload characterization**
  - **Content mapping**
- **Part 3** (*Andreolini*)
  - **Traffic generation**
  - **Data collection and analysis**
- **Part 4** (*Cardellini*)
  - **Impacts of WAN effects**
- **Part 5** (*Colajanni*)
  - **Summary**
  - **Research perspective**

Performance evaluation

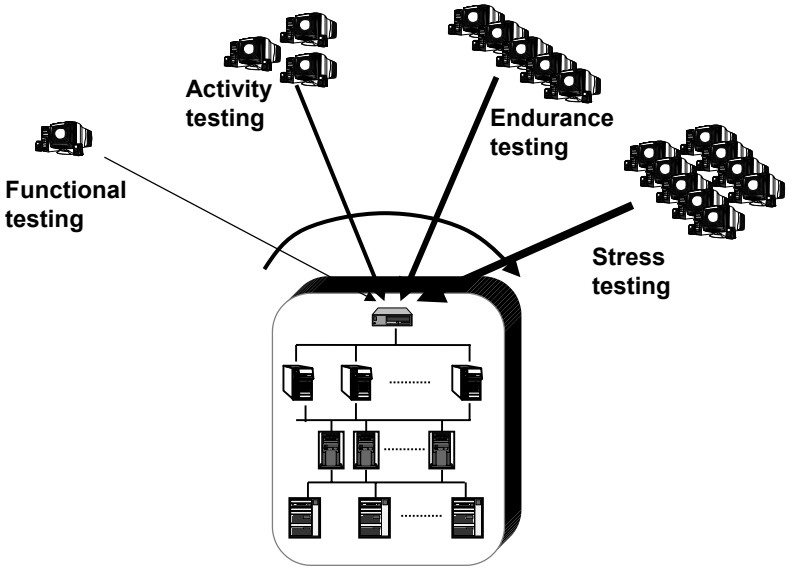
- **Analytical methods**
- **Simulation**
  - **Distribution-driven simulation**
  - **Trace-driven simulation**

They work on a **model of the system**

- **Benchmarking**
- **Load testing**

They operate on a **real system** or a working prototype

Types of testing



Benchmarking *vs.* Load testing

- To evaluate the performance of a given Web-server system by using a well-defined (possibly standardized) workload model
- Performance objectives and workloads are measurable and repeatable on different system infrastructures

- To evaluate the performance of a specific Web site on the basis of actual user behavior
- To provide performance results of a given site under specific application and workload conditions
- Load testing tools capture real user requests for further replay, possibly modified by test parameters

Load testing services

Hosted load testing

- Services offered by test package vendors and other companies
- The Web site is tested from **remote client machines**, that are typically distributed over the Internet
- Hosted load tests are not repeatable and not reproducible due to the unpredictable and variable nature of the Internet traffic

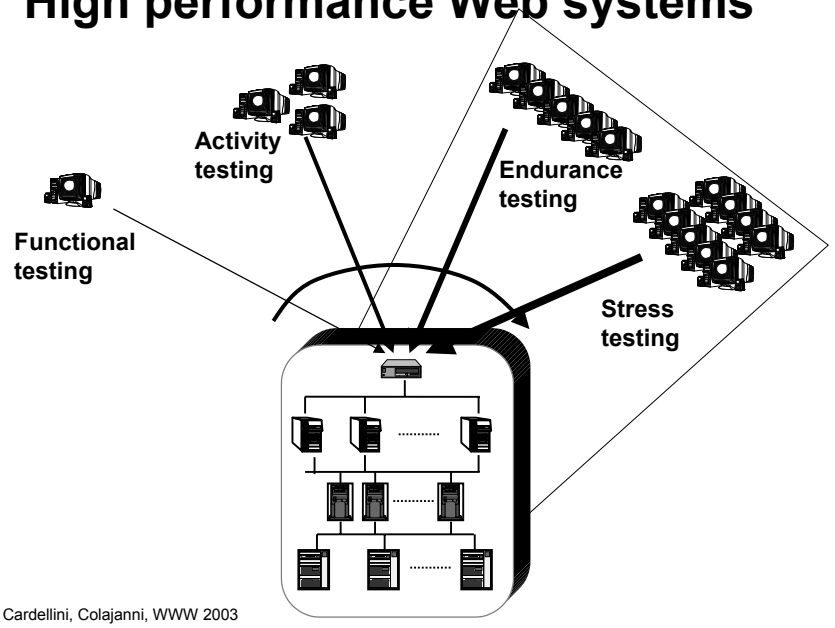
Application performance monitoring

- Services oriented to monitor response time and application functionality under typical loads in the real world

Load testing tools and services

- **Low-end and enterprise-class tool packages that are available on the market**
- **Tools**
  - Cyrano Test
  - Empirix e-TEST suite
  - Mercury Interactive LoadRunner
  - Segue Silk family suite
  - Web Performance Trainer
  - …
- **Services**
  - Empirix
  - Keynote Systems
  - Mercury Interactive
  - Web Performance
  - …

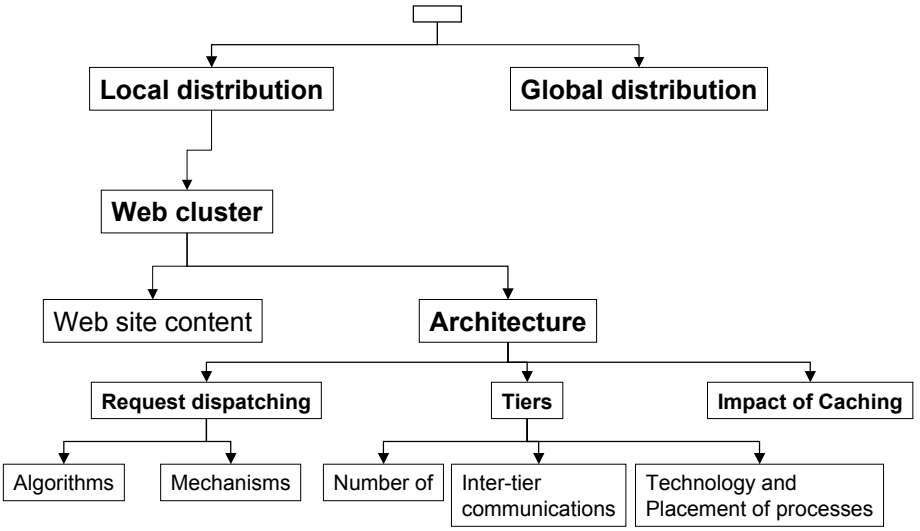
Main focus of testing for High performance Web systems



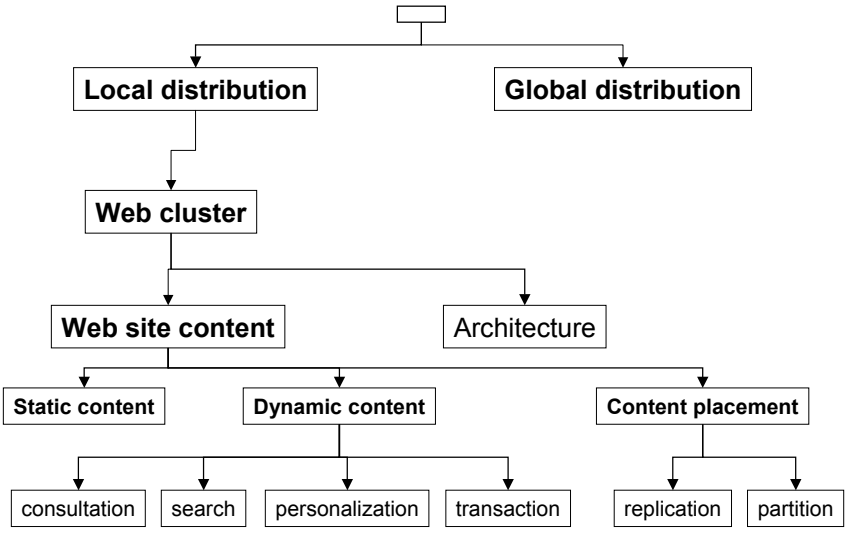
Main goals (*part 1*)

- **COMPARISON OF ALTERNATIVES**
  - To measure the performance and scalability of request dispatching algorithms and request routing mechanisms that are subject to a well-defined workload
  - To assess the impact of changes in the information system:
    - system architecture
    - distribution of content and services
- **TUNING** (basically, other alternatives to be compared)
  - To tune the system components, parameters of request dispatching algorithms and request routing mechanisms
- **CAPACITY PLANNING**

Where the alternatives come from...
(Distributed architecture)



The other sources of alternative ...
(Distribution of content and services)



Web site classification [source: IBM]

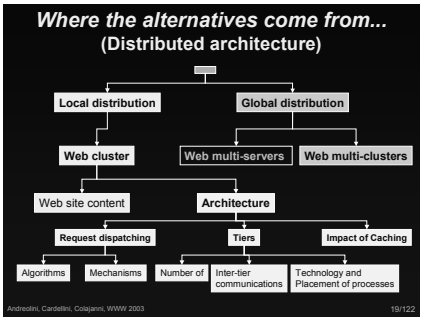
- **"Old plain" Web sites**
- **Publish Web sites**
  - **Provide users with information**
  - Site content tend to change frequently
  - Include search engines and multimedia files
  - Minor security concerns
  - Little or no connection to legacy systems
- **Online shopping sites**
  - **Let users browse and buy**
  - Site content are relatively static, with parts of the catalog where items change frequently (e.g., *promotions*, *special discounts*)
  - Of the total transactions, typically between 1% and 5% are *buy transactions*
  - *Buy transactions* require significant security, privacy, integrity, nonrepudation, authentication
  - Little connection to legacy systems

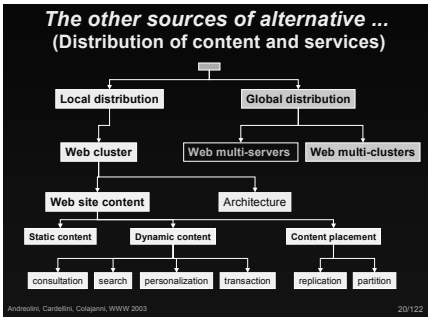
Web site classification (*cont.*)

- **Customer self-service sites**
  - **Let users help themselves** (e.g., *banking from home*, *travel arrangements*, *tracking packages*)
  - High level of personalization
  - Security concerns can be significant
  - High connection to legacy systems
- **Trading sites**
  - **Let users buy and sell** (e.g., *auction sites*, *stock exchanges*)
  - Highest volatile content
  - Complex transactions; nearly all transactions interact with the back-end server(s); can be extremeley time sensitive
  - Significant security, privacy, nonrepudation, integrity, authentication
- **Business-to-business sites → *Web services***

  (Shifting is in course. Any characterization is preliminary)

Putting all together …





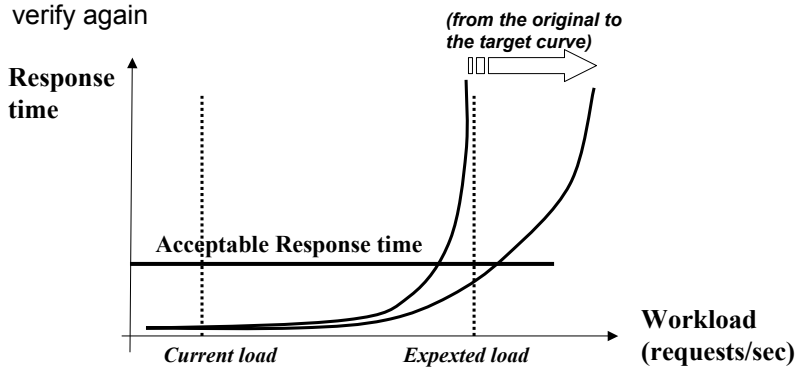
Main goals (*part 2*)

- **COMPARISON OF ALTERNATIVES**
- **TUNING**
- **CAPACITY PLANNING**
  - To evaluate the system capacity and/or response time with respect to an existing or expected workload
  - To conduct a **bottleneck analysis** with the goal of detecting the infrastructure element(s) that may cause performance problems

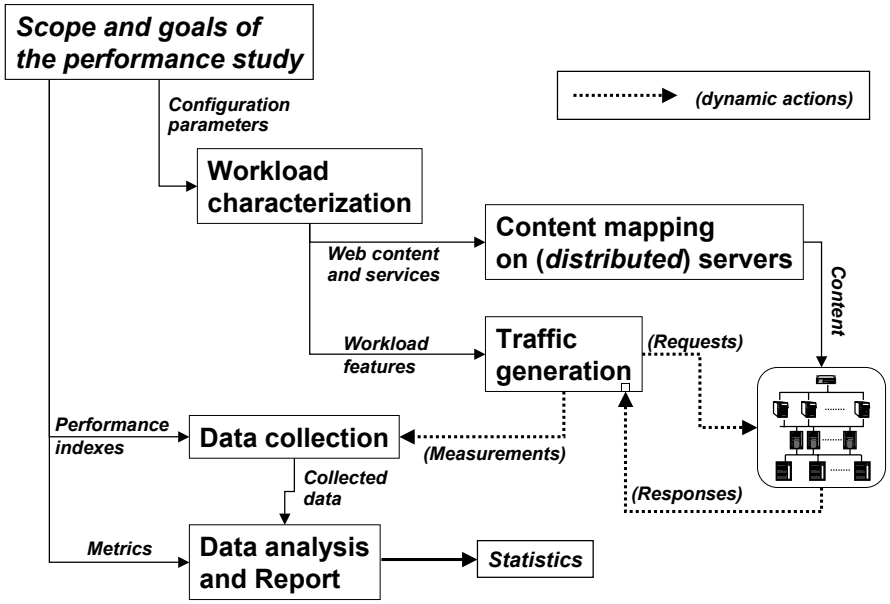
Capacity planning

- **Verification: does the system meet the acceptable response time?**
- **If not:**
  - find the bottleneck
  - find a solution for a more scalable system
  - verify again

Scalability of a distributed system depends on the abilityof <u>each component</u> to scale to meet increasing demand



Outline



Considered tools

Load generator tools

- httperf, HP (Mosberger-Jin)
- S-Client, Rice University (Banga-Druschel)
- Geist, Kant-Tewari-Iyer

Benchmarking tools

- WebStone, Mindcraft
- WebBench, Ziff Davis Media
- SURGE, Boston University (Barford-Crovella)
- Web Polygraph, The Measurement Factory (*main sponsor*)
- SPECweb99*, SPEC organization
- TPC-W* (*specification*), TPC organization

* Standard workload

Tools for transaction-based Web sites (not considered)

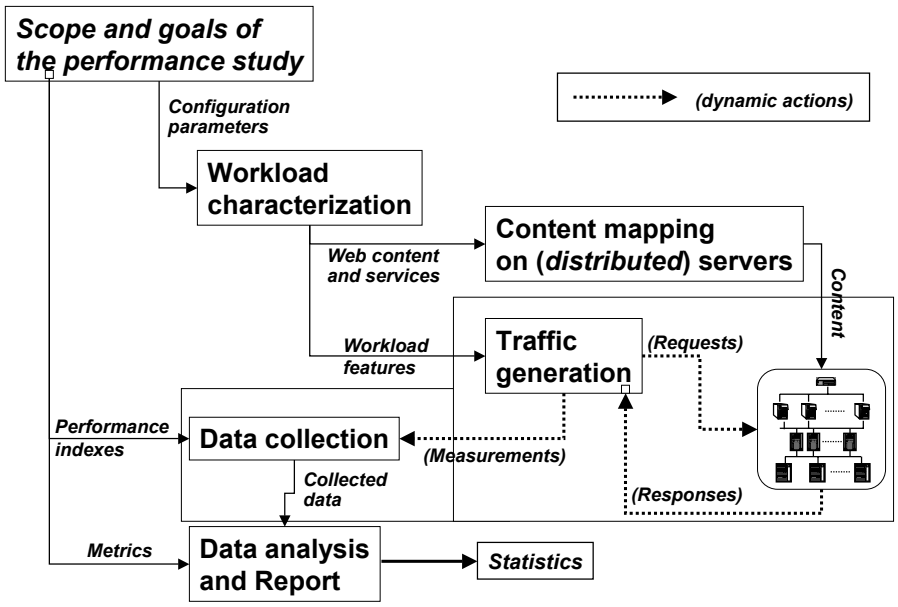
Commercial

- Technovations' Websizer
- Neal Nelson's Web Server Benchmark

Vendor-based

- BEA Systems WebLogic Benchmark
- SAP Standard Application Benchmarks
- Oracle Applications Standard Benchmark
- IBM WebSphere Performance Benchmark Sample

Part 2a: Workload characterization



Importance of workload characterization

- **Represents a distinguishing core feature of existing (and future) benchmarking tools**
- **Aims at reproducing as accurately as possible the characteristics of real traffic patterns**
  - At least the most relevant characteristics of the real workload
- **Not a trivial task as Web traffic exhibits:**
  - **Some peculiar statistical features** (i.e., burstiness and self-similarity)
  - **Less homogeneity than other applications**: large variety, that tends to augment with the increasing number of dynamic requests and classes of Web sites

Web workload is different

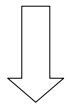
- **The Web is a highly variable system**
  - Geographical location
  - Day of the week and time of the day
  - Responses vary across multiple orders of magnitude
- **Workload is heavy-tailed distributed**
  - Very large values are possible with non-negligible probability
- **Unpredictable nature of information retrieval and service request**
  - Highly variable load and dramatically different access patterns
  - Difficulty of sizing system capacity to support load spikes
- **Web traffic is bursty in several time-interval scales**
  - Peak rates are much larger than the average rate

Aspects of workload characterization

- **Web-based service characterization**
  - Types of services being requested to the Web-server system
  - Benchmark related issues:
    - How realistic is the workload generated by the benchmark?
    - Capability of the benchmark tool of generating different types of traffic
    - As workload characteristics change over time, benchmarks must too
- **Request stream characterization**
  - Methodology used to generate the stream of requests
- **Web client characterization**
  - Model of the emulated Web client

Issues in service characterization

- **Server workload characteristics depend on the class of the Web site**
- **The category of the Web site affects the distribution of the load over time**
  - E.g., online shopping and B2B sites have different peak and busy load periods



- **The workload for "the Web site" does not exist**

  For example, IBM identifies five classes of high-volume Web sites

Web site classification

- **"Old plain" (simple browsing) Web sites**
  - Content is mainly static

Large volumes of dynamic transactions for remaining classes

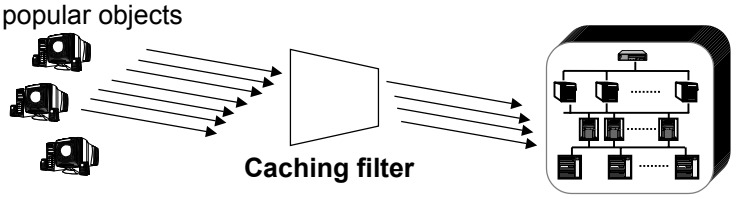
- **Publishing Web sites**
  - Content change frequently
    - But fairly static information sources
  - Security considerations are minor
- **Online shopping sites**
  - Content can be relatively static and dynamic
  - Significant amount of secure transactions
  - Typical requests such as browse, search, select, add, and pay
- **Customer self-service sites**
  - Complexity of transactions
  - Multiple data sources, consistency issues are significant
  - Significant amount of secure transactions

Web site classification *(cont.)*

- **Trading sites**
  - A great deal of rapidly changing (volatile) content
  - Complexity of transactions
    - Most transactions interact with back-end servers
  - Significant amount of secure transactions
  - Typical requests such as browse, select, bid
- **Business-to-business / Web services sites**
  - Complexity of transactions (substantial purchasing activity)
  - Multiple data sources, consistency issues are significant
  - Nearly all transactions are secure

Evolution of server workload characteristics

- **Growing popularity of Web-based services that require dynamic content generation**
- **Trickle-down effect for static content due to Web caching and Content Distribution Networks**
  - Caches absorb the majority of static object requests to the most popular objects



New issues for benchmarking studies:
- Generation of dynamic content
- Effects due to internal and external caching

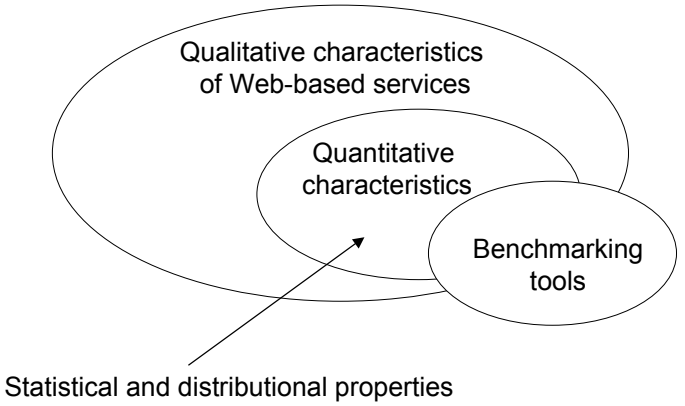
Dynamic content characterization

- **Studies on Web workload characterization have been focused on static content**
  - Non e-commerce Web traffic → a large amount of requests are for static content
- **Still few studies consider Web sites with prevalent dynamic (and personalized) Web content**
  - Difficulties in obtaining data for dynamic content characterization
    - Unavailability of representative data
    - Privacy, competitive concerns
  - No consensus as to what constitutes a representative dynamic workload
- **Even some conclusions on static content need to be revisited** (e.g., *academic sites* vs. *commercial sites*)

Classification of Web sites and workload characterization

Class of Web site	*Known results and characterization*
Simple browsing	
Publishing	
Online shopping	
Customer self-service	
Trading	
B2B	
Web services	

Workload characterization and tools



Workload characterization and tools *(cont.)*

- **Some benchmarking tools oriented to *stress testing* (e.g., S-Client) do not aim to provide a realistic workload characterization**
- **Some tools (e.g., SPECweb99, WebStone, WebBench) provide a limited characterization**
- **Some tools attempt to provide a realistic characterization (e.g., SURGE, TPC-W, Web Polygraph), at least for a class of Web site**
- **For some tools the characterization depends on the configuration (e.g., httperf, Geist)**

Characterization of simple browsing sites

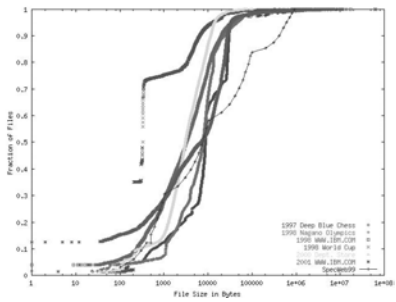
- **Many research results capture the characteristics of static Web content**
  - Including high-volume Web sites

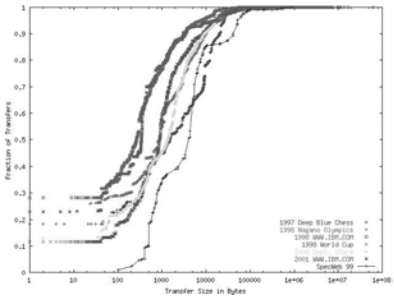
> Arlitt and Williamson, *IEEE/ACM Trans. on Networking*, Oct. 1997
> Crovella and Bestavros, *IEEE/ACM Trans. on Networking*, Dec. 1997
> Barford and Crovella, *Proc. Performance/ACM Sigmetrics 1998*, July 1998
> Iyengar et al., *World Wide Web Journal*, Mar. 1999
> Pirolli and Pitkow, *World Wide Web Journal,* Mar. 1999
> Barford and Crovella, *Proc. ACM Sigmetrics 1999*, May 1999
> Liu et al., *Performance Evaluation*, 2001

- **A wide range of investigated characteristics**
  - *User behavior characteristics*
    - Session and request arrivals, clicks per session, request interarrival times
  - *Object characteristics* ⇦ Our example
    - Sizes, content types, resource popularity, resource changes, temporal locality, embedded objects
  - *HTTP characteristics*

Simple browsing sites: object characteristics

- **Object size**
  - Unique and transferred objects
  - Sizes have Lognormal distribution for body, Pareto for tail (some controversy)
  - Most responses are small, most of the bytes are from large transfers
  - Although well studied, size distributions are reproduced relatively poorly by some benchmarking tools (SPECweb99 analysis by Nahum at WCW02)





[Source: Nahum, *Proc. WCW 2002*]

- **Object popularity**
  - Zipf-like distribution

Characterization of publishing sites

- **Some studies**

Arlitt and Jin, *IEEE Network*, May 2000
Padmanabhan and Qiu, *Proc. ACM Sigcomm*, Aug. 2000
Shi et al., *Proc. WCW 2002*, Aug. 2002
Shi et al., *Proc. IEEE Globecom 2002*, Nov. 2002

- **Object characteristics**
  - Exponential distribution of objects sizes embedded in a dynamic page (not heavy-tailed) [Shi02]
    - Some controversy: certainly some big transfers
- **Peculiar characteristics of dynamic content**
  - Freshness time (Weibull distribution)
  - Content reusability

Characterization of online shopping sites

- **Some studies**

Menascé et al., *Proc. ACM Conf. on Electronic Commerce,* Oct. 2000
Arlitt ed al., *ACM Trans. Internet Technology*, Aug. 2002
Vallamsetty et al., *Proc. Wecwis 2002*, June 2002
Shi et al., *Proc. IEEE Globecom 2002*, Nov. 2002

- **E-commerce traffic is significantly more complex than simply-browsing traffic**
  - A variety of activities (Menascé and Almeida)
  - A high level of Online Transaction Processing activity
  - A high proportion of dynamic requests
- **Arrival characteristics**
  - Arrival traffic is more bursty than normal
- **Object characteristics**
  - ***Size*** of transferred objects is ***not heavy-tailed***
    - But response times show heavy-tailed behavior (due to server processing and back-end data retrieval times
  - Popularity of search terms (*Zipf-like* distribution)
  - Freshness time (*Bimodal* distribution)

Characterization of trading and B2B sites

- **Very few preliminary results**
  - For trading sites

    Menascé et al., *Proc. ACM Conf. on Electronic Commerce,* Oct. 2000

  - For B2B sites

    Vallamsetty, Kant, et al., *Proc. Wecwis 2002*, June 2002
    Vallamsetty, Kant, Mohapatra, *Electronic Commerce Research,* Jan. 2003

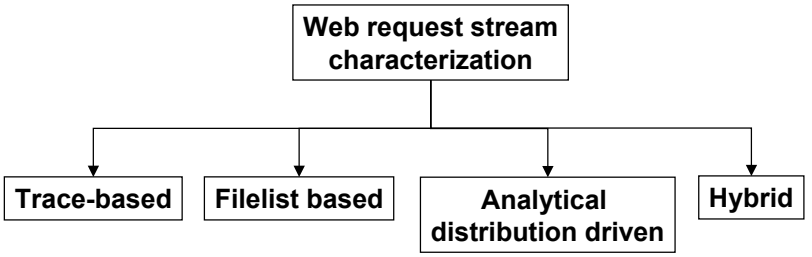
- B2B sites
  - Heavy-tailed distribution of response times
  - Lower number of embedded objects
  - Secure pages are simpler
- Lack of back-end transactional characteristics and their relationship with front-end transactions

Service characterization

Tools	**Class of Web site**
httperf	*Trace-based workload*
WebStone	Simple browsing
SPECweb99	Simple browsing
Web Polygraph	Simple browsing
SURGE	Simple browsing
S-Client	-
WebBench	Simple browsing Online shopping
TPC-W	Online shopping
Geist	*Trace-based workload*

Characterization of the request stream

- **How to generate the request stream issued to the system under testing**



Trace-based approach

- **Request generation is driven from an actual pre-recorded or synthetically generated trace of server activity; two alternatives:**
  - Replay the requests as logged in the trace
  - Extract session-oriented information through a preliminary trace analysis
- **Typically used in load testing tools**

- **<u>Pros and cons</u>**
  - Ability to use actual traces from a live site
  - The trace reflects a real but specific workload
  - Difficulties in obtaining the traces
  - Trace generation can be a quite expensive process
  - Not enough data in the Web server log files to reproduce requests
  - Identification of user sessions is not a trivial task

Filelist based approach

- **The benchmark defines a given set of files (divided in classes) with their relative access frequency**
  - File sizes and access frequency based on the analysis of logs from several large Web sites
  - E.g., SPECweb99 has 4 classes of file sizes with Poisson distribution within each class
- **Time characteristics are typically not taken into account**
- **Pros and cons**
  - Simplicity of implementation
  - Lack of flexibility
  - No user-session oriented

Analytical distribution-driven approach

- **Mathematical distributions to represent characteristics of the workload**
  - E.g., SURGE captures size distributions, object popularity, embedded objects, temporal locality of reference, user think times
- **Pros and cons**
  - Different workload parameter values to model a variety of data sets
  - Possibility to change workload parameters one at time
  - Model of user behavior
  - Degree of representativeness
    - Related to the availability of workload characterization studies

Request stream

Tools	Request stream
httperf	Trace-based, hybrid
WebStone	Filelist based
SPECweb99	Filelist based
Web Polygraph	Hybrid
SURGE	Analytical distribution-driven
S-Client	Filelist based
WebBench	Filelist based
Geist	Trace-based

TPC-W is not included (it provides a specification)

Web client characterization
(*wish list*)

- **Support of HTTP specifications**
  - HTTP/1.0 and HTTP/1.1 features
    - Persistent connections and pipelining
    - Support for various request methods (GET, POST, HEAD, …)
    - Chunked encoding and range requests
    - Parallel connections
  - Client robustness to conditions
    - The client should capture response codes different from 200 OK
      - E.g., the client should handle conditional requests
- **Cookie handling**
- **Support for SSL/TLS encryption**

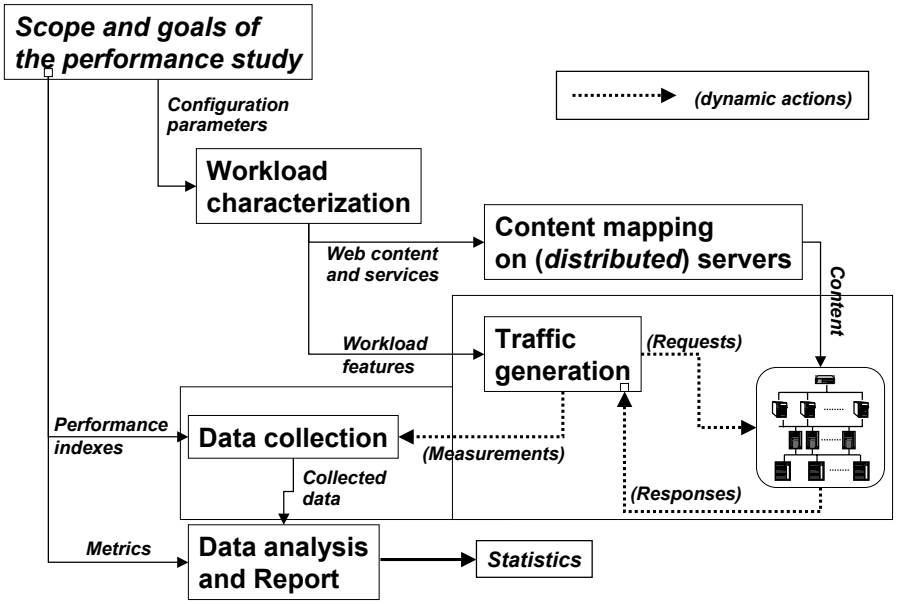
Client characterization

Tools	HTTP/1.1	Methods (no GET)	Cookies	SSL/TLS
httperf	✓✓	✓✓	✓✓	✓
WebStone	X	X	X	X WebStone SSL
SPECweb99	✓	✓	✓✓	X SPECweb99_SSL
Web Polygraph	✓	X	X	✓
SURGE	✓	X	X	X
S-Client	X	X	X	X
WebBench	✓	✓✓	✓✓	✓
Geist	X	X	X	✓

TPC-W is not included (it provides a specification)

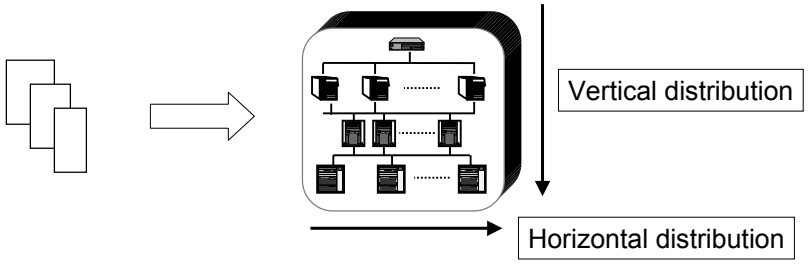
✓✓ Adequate
✓ Needs improvement
X Inadequate

Part 2b: Content mapping on servers



Content mapping

- **Replication (distribution) of the synthetic content among the multiple server nodes**
  - Among nodes at the same tier and at different tiers
  - Mapping of static and dynamic content



- **The replication strategy may differ on the basis of the system architecture**

Generation of dynamic content

- **A variety of technologies for generating dynamic content on the server side**
  - EJB, ASP, JSP, PHP, Java Servlets, CGI, NSAPI, ISAPI…
  - The nature of dynamic content and its charactersitics tend to evolve in the future
- **A dynamic request is usually served by one Web application, usually an executable program or a script under some interpreters**
- **Generation of the set of data to be placed on the back-end servers**

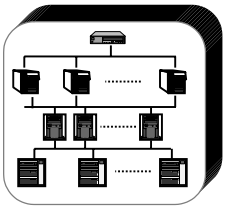
Approaches for content mapping

- **As content mapping is an error-prone operation, it should be automated by the tool as much as possible**
- **Main alternatives for content mapping**
  - **Full support**
    - Static and dynamic content
  - **Partial support**
    - Only replication (distribution) of static content is automated
  - **No support at all**
- **Few existing tools provide some facility for content mapping**

Content mapping on distributed Web systems

- **The site content may be**
  - Fully replicated
  - Partially replicated
  - Partitioned

} Increase secondary storage scalability
Use specialized servers



- **None of the existing tools includes any utility for partial content replication**

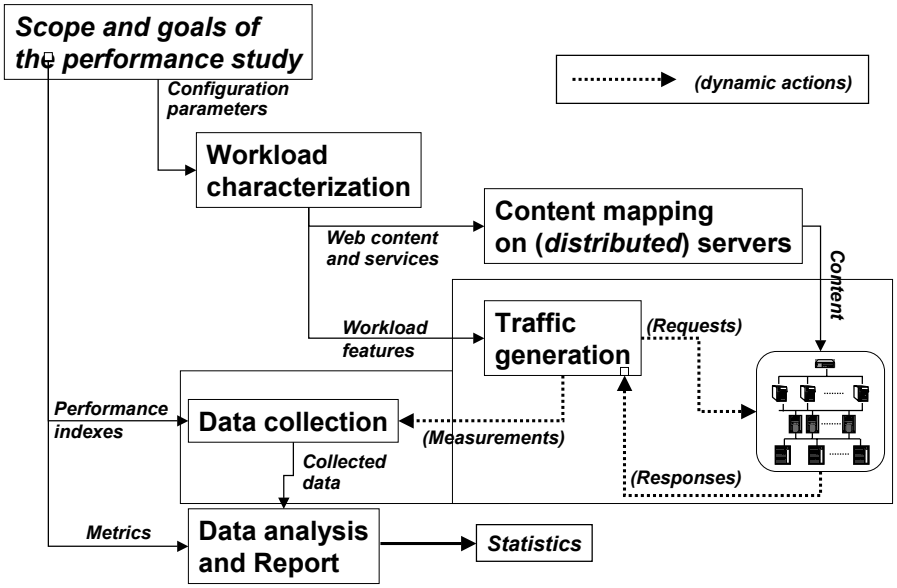
Content mapping

Tools	Single server	Multiple servers
httperf	X	X
WebStone	✓	✓
SPECweb99	✓	✓
Web Polygraph	✓	✓
SURGE	✓	✓
S-Client	X	X
WebBench	✓	✓
Geist	X	X

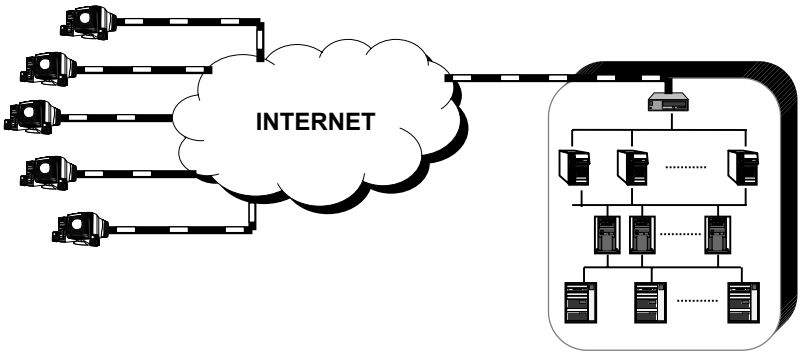
TPC-W is not included (it provides a specification)

✓✓ Adequate
✓ Needs improvement
X Inadequate

Part 3: Traffic generation



Traffic to a distributed Web system



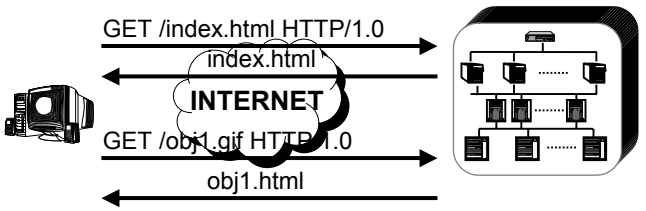
- Distributed Web server systems are motivated by huge numbers of concurrent accesses and/or complex operations per request

Traffic generator requirements for distributed Web systems

- **Scalability** of the traffic generator
  - **Clients never have to be a bottleneck**
  - **Generate the maximum amount of traffic for a given client node**

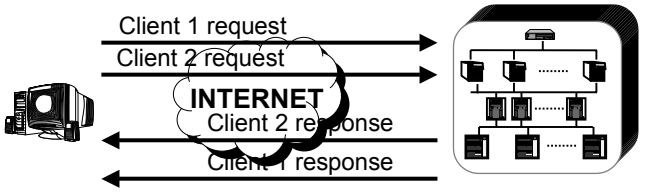
This is important for capacity planning and tuning studies, that typically require the generation of higher amounts of traffic

Traffic generator requirements for distributed Web systems



- Many benchmarking tools usually implement a *closed loop* request model
- **PROBLEMS**
  - It does not generate sustained load
  - Even worse, the frequency of client requests depends on the performance of the Web system

Traffic generator requirements for distributed Web systems



- For distributed Web systems an *open loop* request model is preferable
  - Client requests are issued following given characteristics, independently of server capacity
- This approach does generate sustained load
- More difficult to be implemented

Traffic generators of present benchmarking tools

- Oriented to *stress testing* of a single server (may be sufficient for tuning)
- Few benchmarking tools aim at generating sustained overload sufficient for distributed Web systems. E.g.,
  - **S-Client, httperf: event driven, non-blocking I/O**
  - **Geist: sender and receiver threads**

Classification of traffic generators

Two main characteristics

- **Engine architecture**
  - **Defines the number and type of computational units used to generate Web traffic**
- **Coordination scheme**
  - **Defines the ability of configuring and synchronizing the computational unit executions**

Classification of traffic generators

- Engine architecture: defines the number and type of computational units used to generate Web traffic
  - Single process
    - Closed loop and open loop implementation
  - Multiple processes
  - Multiple threads
  - Hybrid
- Coordination scheme: defines the ability of configuring and synchronizing the computational unit executions
  - Master-Client
  - Master-Collector-Client

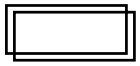
Engine architecture

- Notation:

Client node

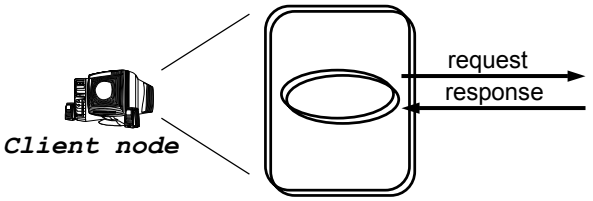
User process

Group of threads



Collector

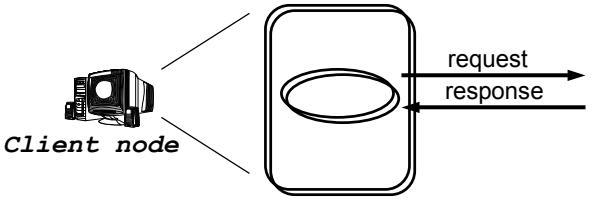
Engine architecture (*single process*)



- **Single process** – *closed loop* implementation
  - **Does not suffer from the context switch problem**
  - **Subject to per-process resource limits** (*file descriptors*)
  - **Clients do not send requests faster than the server can respond**

Examples: **WebBench (physical client)**

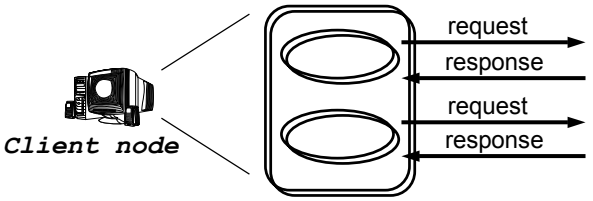
Engine architecture (*single process*)



- **Single process** – *open loop* implementation
  - **Does not suffer from the context switch problem**
  - **Clients send requests faster than the server can respond**
  - **Subject to per-process resource limits** (*file descriptors*)
  - **Does not allow more than one process per client node**

Examples: **httperf, S-Client, Web Polygraph**

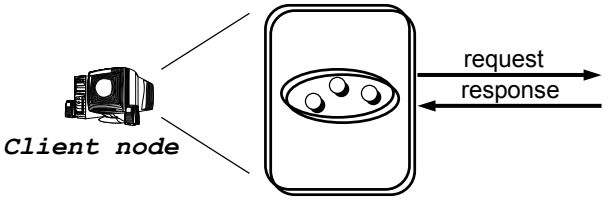
Engine architecture (*multiple processes*)



- **Multiple processes**
  - **Increase available resources**
  - **Suffers heavily from the context switch problem**
  - **Separated process address spaces**

Examples: **WebStone, SPECweb99**

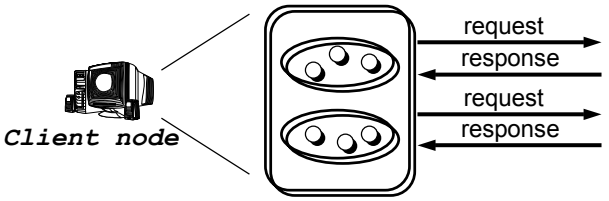
Engine architecture (*multiple threads*)



- **Multiple threads**
  - **Avoid (expensive) context switches**
  - **Much faster than processes**
  - **Process address space is shared**
  - **Subject to per-process resource limits**

Examples: **WebBench (logical client), SPECweb99**

Engine architecture (*Hybrid*)



- **Hybrid**
  - **Increases available resources (file descriptors)**
  - **Much faster than process-based architecture**
  - **May suffer from context switch problems**
  - **Does not allow more than one process per client node CPU**

  Examples: **SURGE, Geist**

Recommendations for engine architectures

- Most promising architecture:

  hybrid (very efficient on SMP machines)

- Implementation techniques for sustained overload:

  event-driven approach and **non-blocking I/O**

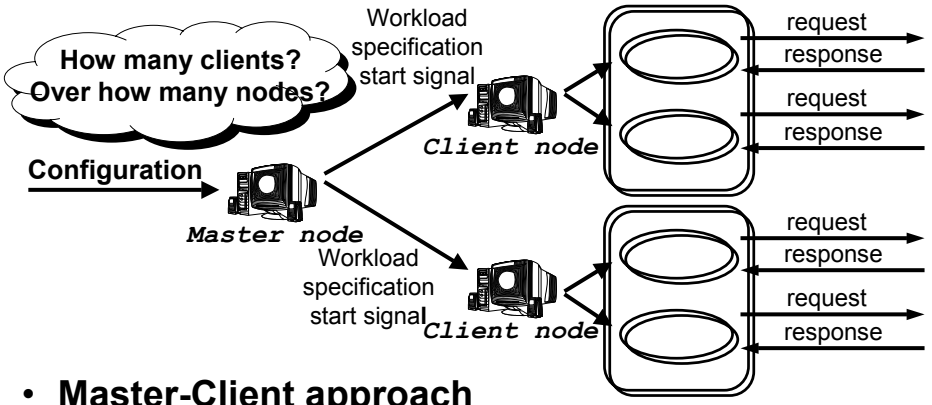
To achieve even higher scalability

⇩

Use several client nodes

An architecture with multiple client nodes opens a new issue:
How to coordinate processes running on distinct nodes?

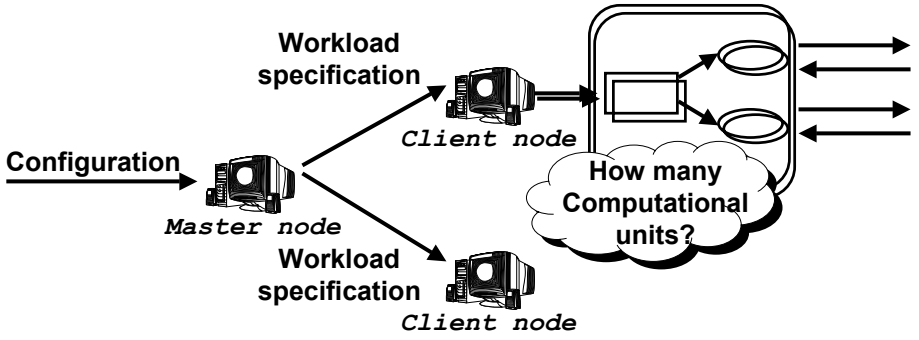
Coordination scheme *(cont.)*



- **Master-Client approach**
  - **Master must synchronize with *every client***
  - ***Sensible communication overhead with many clients***

Examples: **WebStone, WebBench (physical client), Geist**

Coordination scheme



- **Master-Collector-Client approach**
  - **Master synchronizes with *every collector***
  - ***Reduced communication overhead***

Examples: **SURGE, WebBench (logical client)**

Recommendations for coordination schemes

Scalable traffic generation requires many clients

Coordinate them automatically!

- The most promising coordination scheme:

Master-Collector-Client

Tool summary

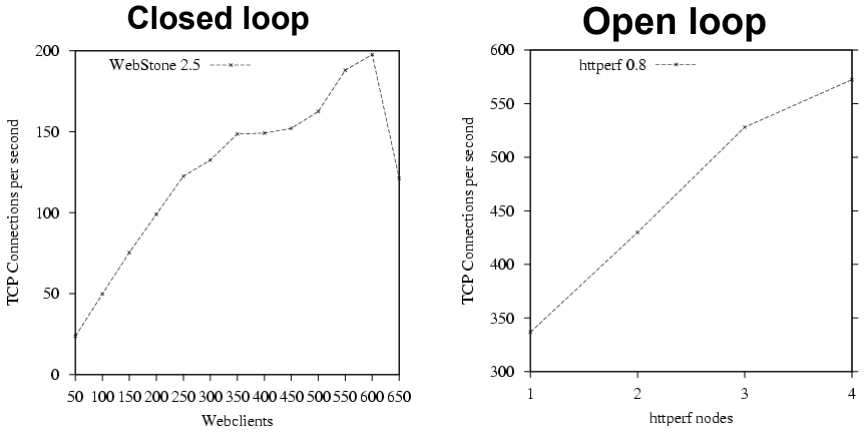
✓✓ **Adequate**

✓ **Not bad - needs improvement**

x **Inadequate**

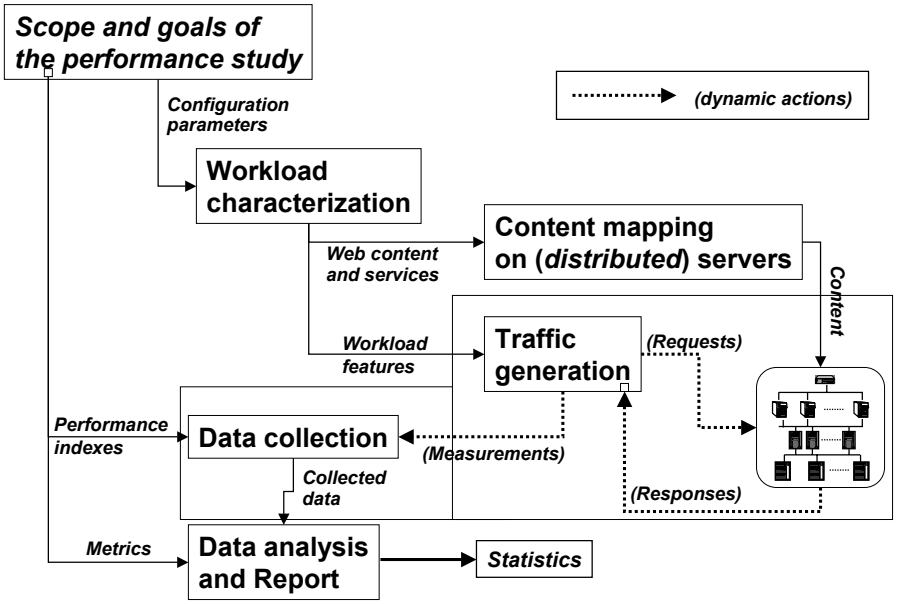
	Sustained workload
Geist	✓✓
httperf	✓
SURGE	✓
S-Client	✓
Web Polygraph	✓
WebBench	x
WebStone	x
SPECweb99	x

Scalability of traffic generators (example)



- **httperf is able to sustain almost three times the number of connections as Webstone**

Part3.b: Data collection and analysis



Web cluster architecture:
Performance indexes

Network/OS technology

Utilizations, error rate, connection rate, DNS lookup time, TCP connect time, network throughput

Web server technology

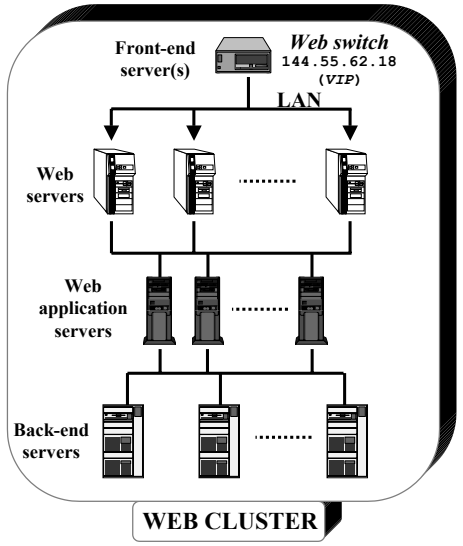
Request rate, reply rate,Web object latency and transfer time, Web object response time, Web page response time, Web session time

Middleware technology

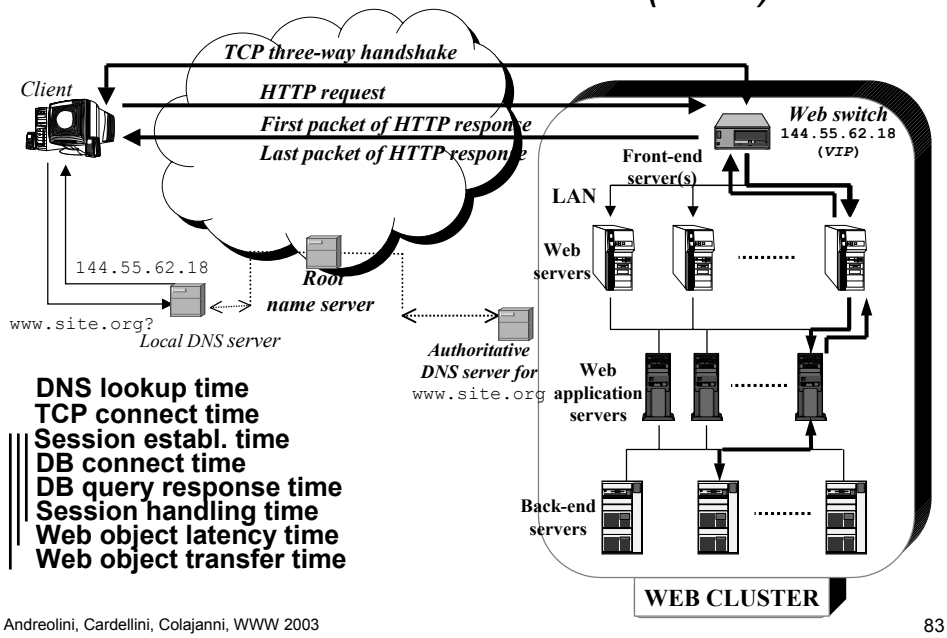
Session establishment time, session response time, session handling time

Database technology

Query throughput, database connect time, query response time



Performance indexes *(cont.)*

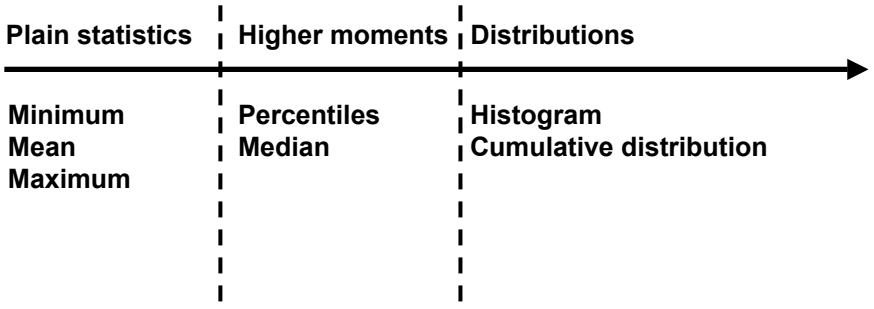


DNS lookup time
TCP connect time
Session establ. time
DB connect time
DB query response time
Session handling time
Web object latency time
Web object transfer time

Performance indexes

- *Performance indexes* may be considered at different levels: OS, network, HTTP, session
- Web workload is session- and transaction-oriented
  - ***Session-oriented indexes* (yield a realistic view of user perceived performance)**
  - ***Administration-oriented indexes* (give an idea of how cluster components are behaving)**
    - **Have to be measured on cluster components**

Performance metrics

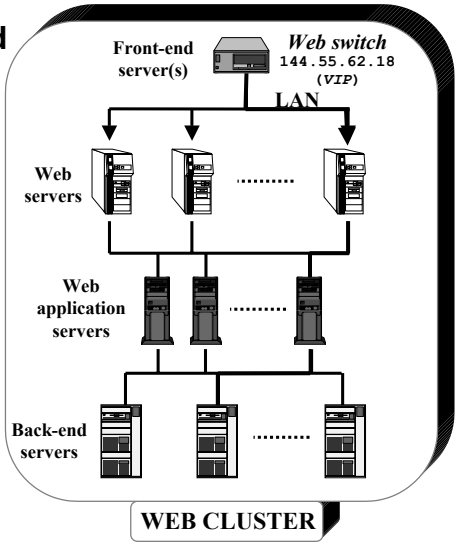


- Session oriented indexes exhibit heavy-tail behavior
  - ***Min, Mean, Max, StdDev* are often not representative**
  - ***Higher moments* should be computed at least**

Component monitoring

- **A client request is processed by several components**
- **Where is the bottleneck?**
- **Need of monitoring each cluster component**

- **Some characteristics and problems may result only through log analysis. E.g.,**
  - **Client trends**



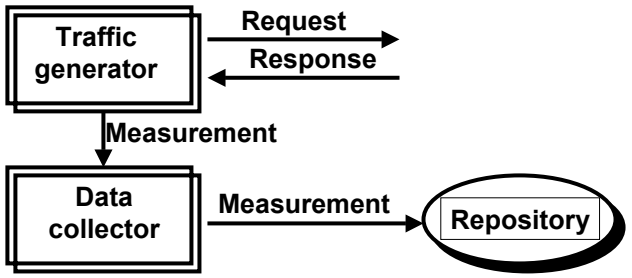
Data collection requirements for distributed Web systems

- **Aggregation of collected data** from distinct client nodes to obtain global reports and higher order statistics
- Use of **session-oriented** performance indexes and **appropriate metrics**
  - Give a better idea of cluster capacity in terms of users
- **Monitor facilities** for the main cluster components
  - Facilitate the task of finding bottlenecks
  - Useful in capacity planning and tuning studies
- **Log analyzers** for the main cluster components
  - Useful for trend analysis

Data collection of present benchmarking tools

- **Session-oriented performance indexes are often not collected**
- **Useful performance indexes are often *not* represented by means of appropriate metrics**
- **Aggregation of data collected from clients running on distinct client nodes is not always performed**
- **Different application server technologies have their monitoring and analyzer tools**
  - There is no standard
  - Some log formats are inadequate (typically, too coarse)

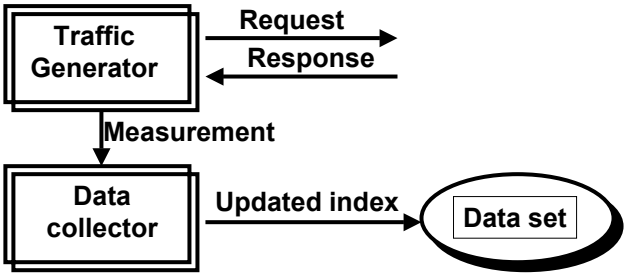
Strategy for data collection



- **Record storage**
  - **Allows for the computation of higher moments and histograms**
  - **Requires a big amount of resources**

Example: **SURGE**

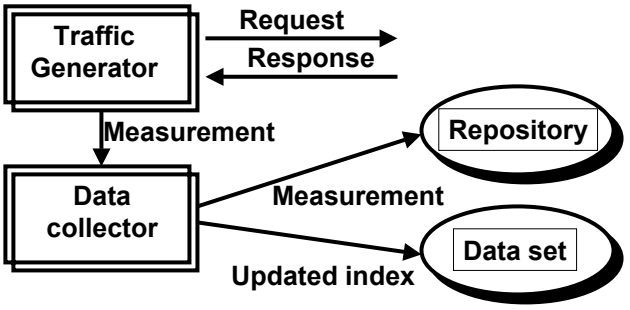
Strategy for data collection *(cont.)*



- **Data set processing**
  - **Does not require a large amount of system resources**
  - **Does hardly allow the computation of higher moments and histograms**

Example: **S-Client, WebBench, SPECweb99**

Strategy for data collection *(cont.)*



- **Hybrid**
  - **Does not require the big amount of system resources needed by record storage**
  - **Allows for the computation of higher moments and histograms**

Example: **httperf, WebStone, Web Polygraph**

Recommendations for data collection

Store records only
for heavy-tailed performance indexes
(since it is expensive)

Typically: Web session lengths, page and object response times

Provide session-oriented reports
Try to monitor cluster components
Analyze component logs

- Most promising data collection scheme:

Hybrid
(best tradeoff between precision and speed)

Tool summary

✓✓ **Adequate**

✓ **Not bad - needs improvement**

x **Inadequate**

	Data aggregation	Session metrics	Monitor facilities	Log analysis
httperf	x	✓	x	x
SPECweb99	✓✓	x	x	x
SURGE	x	✓✓	x	✓
S-Client	x	x	x	x
WebBench	✓✓	x	x	x
Web Polygraph	x	x	x	x
WebStone	✓✓	x	x	x
Geist	x	x	x	x