Performance

2023

This presentation is based on

- Chapter 3: “Optimizing Performance of Enterprise Web Application”

  Book: “Architecting High Performing, Scalable and Available Enterprise Web Applications” by: Shailesh Kumar Shivakumar, 2015

Chapter 8: Performance

Book: Software Architecture in Practice, 3th Ed

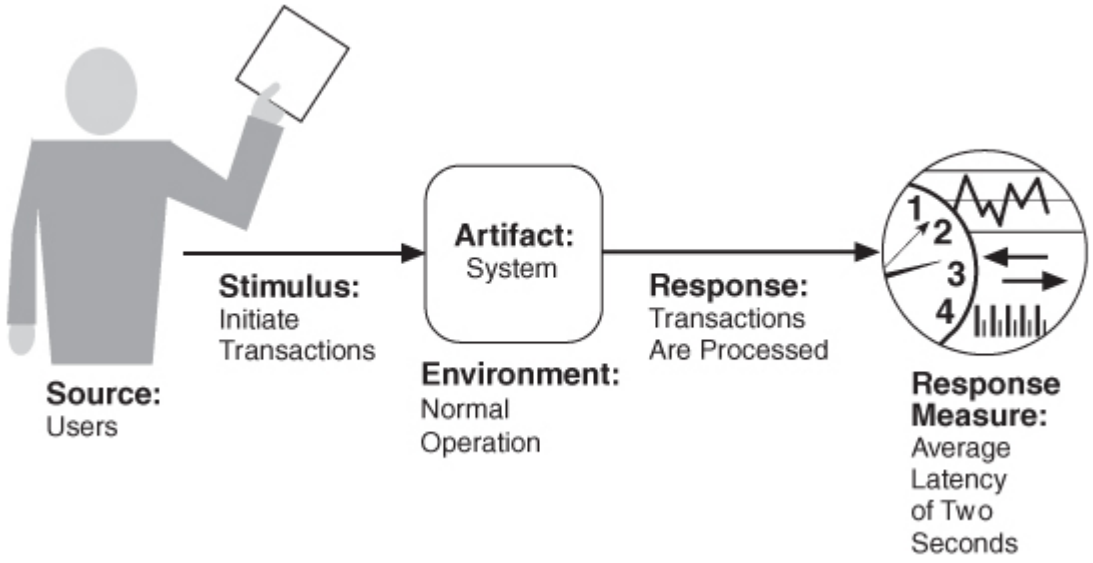
Definition

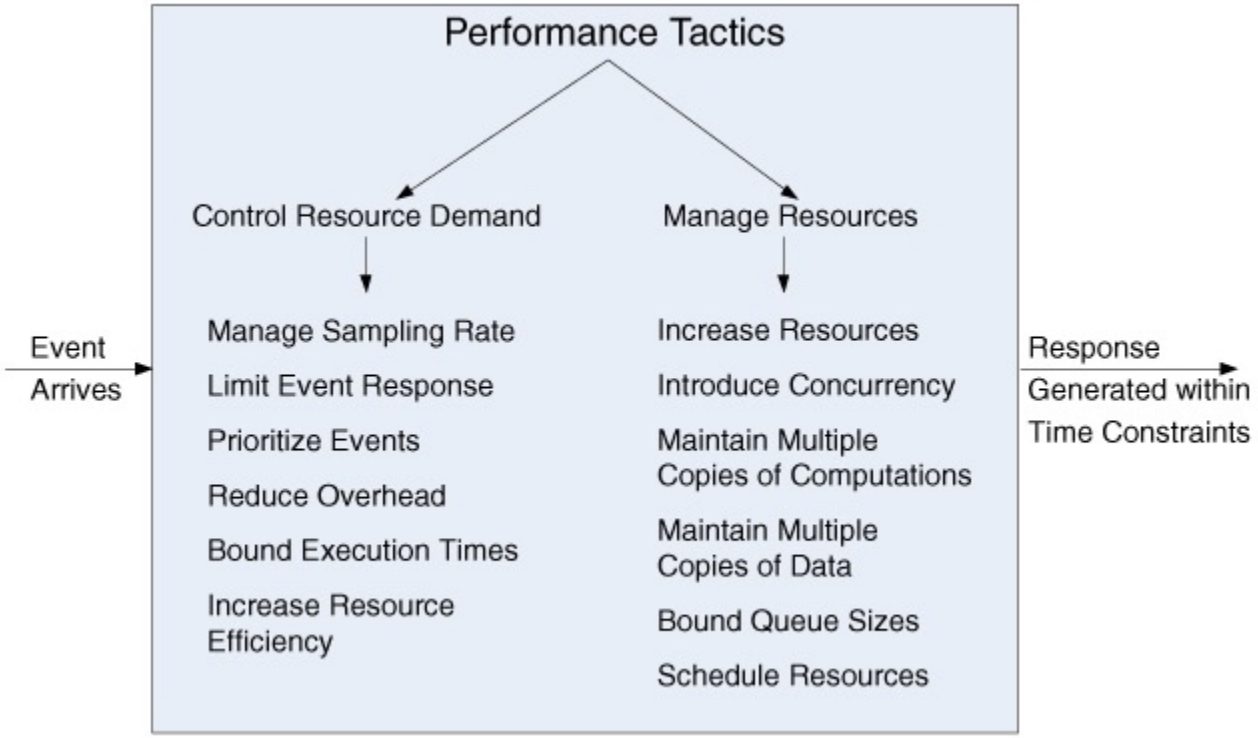
- It’s about time and the software system’s ability to meet timing requirements

Requiremens Spec by Scenarios

- Source:
  - Concurrency
  - Events can arrive in predictable patterns or mathematical distributions, or be unpredictable.
- Stimulus:
  - Event arriving at the system
- Response:
  - *Latency*
  - *throughput*
  - *jitter*

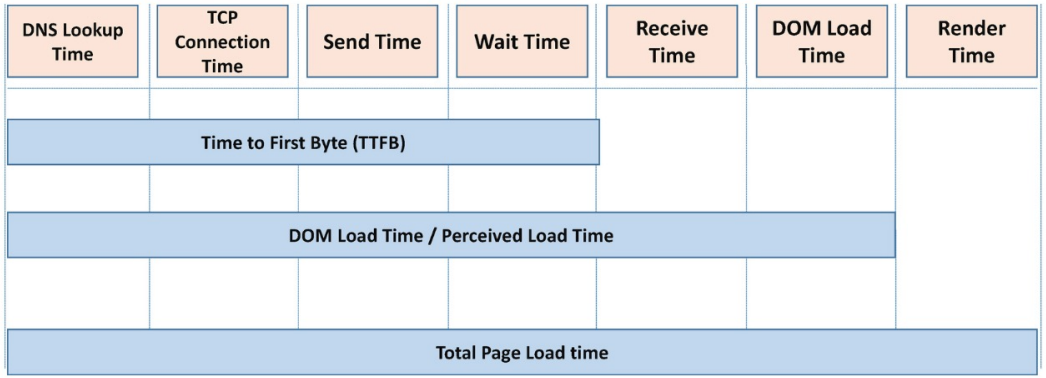
Portion of Scenario	Possible Values
Source	Internal or external to the system
Stimulus	Arrival of a periodic, sporadic, or stochastic event
Artifact	System or one or more components in the system
Environment	Operational mode: normal, emergency, peak load, overload
Response	Process events, change level of service
Response Measure	Latency, deadline, throughput, jitter, miss rate



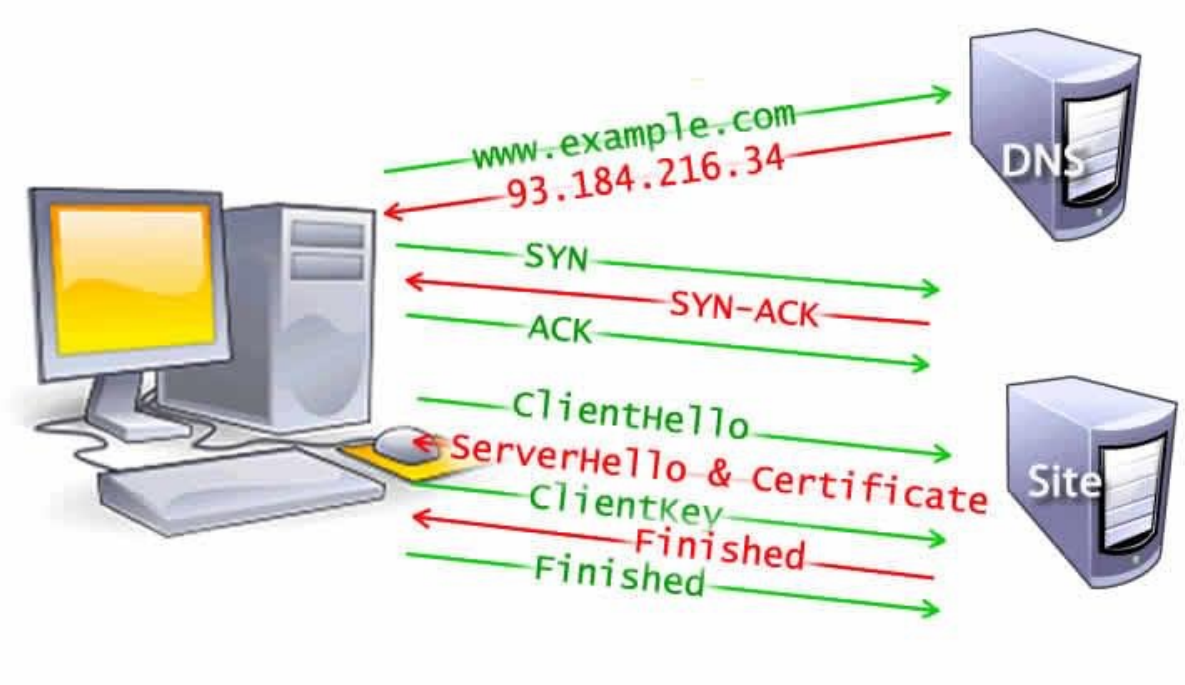

Performance Tactics
Control Resource Demand
Manage Resources
Manage Sampling Rate
Limit Event Response
Prioritize Events
Reduce Overhead
Bound Execution Times
Increase Resource Efficiency
Increase Resources
Introduce Concurrency
Maintain Multiple Copies of Computations
Maintain Multiple Copies of Data
Bound Queue Sizes
Schedule Resources
Event Arrives
Response Generated within Time Constraints

Web Page Metrics

- **Page response time (PRT) or page load time (PLT)** is the overall time taken for rendering the page Document Object Model (DOM). It is the total time between initial request and the time when all page objects are downloaded

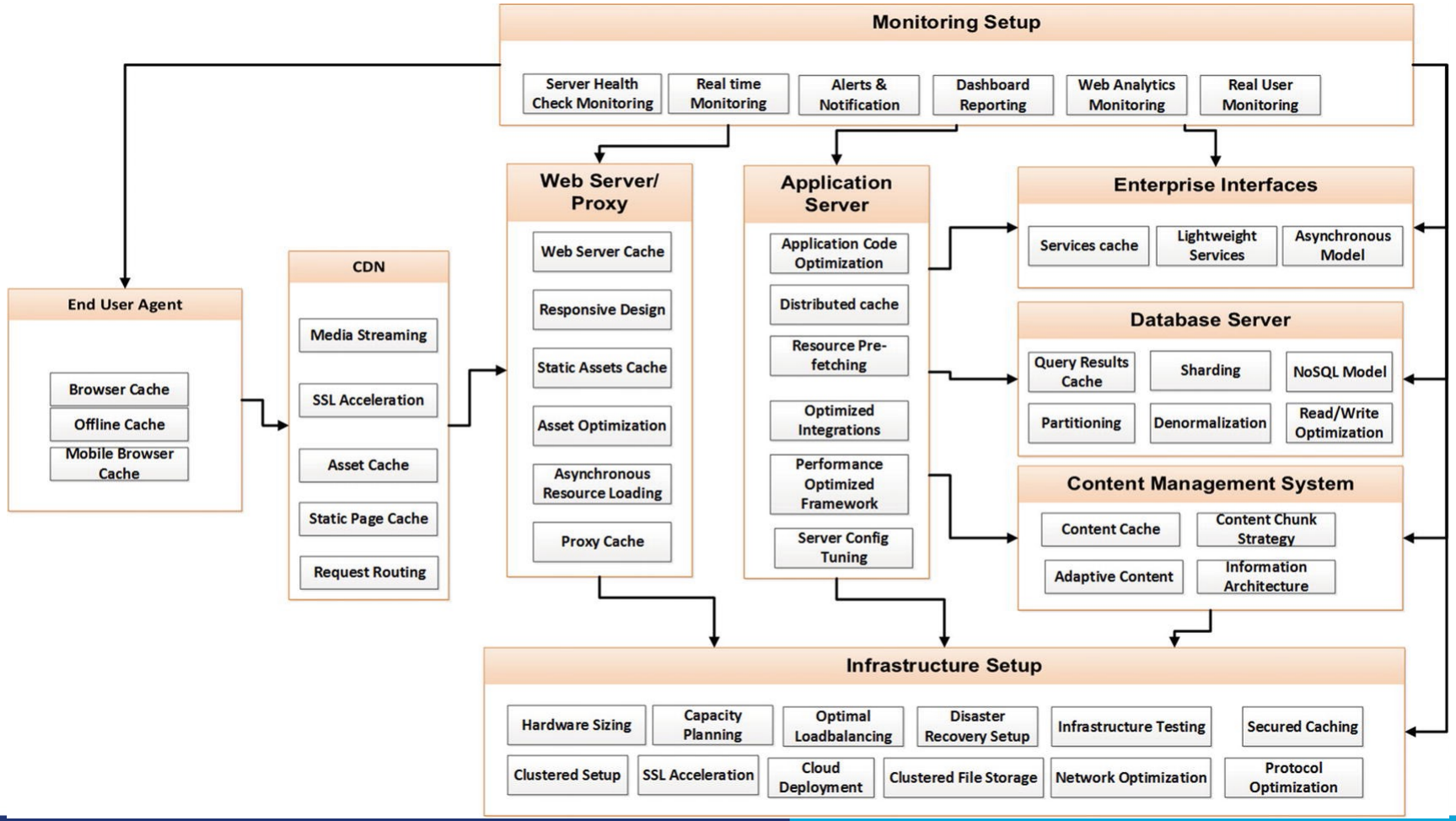


Web Page Metrics (2)



Page response time = DNS lookup time + TCP Connect time+ server response time (time needed for sending, waiting, and receiving) + Object download time.

Web Performance Optimization End-To-End Flow



Strategy-1

1. Define Non-functional requirements of Performance
2. Design
    1. Patterns
    2. best practices
    3. Tactics
3. Build
    1. App
4. Deploy
    1. InfraTI (HW, SW Base, Network)
    2. App

Strategy for legacy systems

1. Define Non-functional requirements of Performance
2. Establisment of Base Line of Performance
    1. i.e: jmeter
3. Work on
    1. InfraTI
        1. Hardware
        2. Network
        3. Software (SO, Databases, security servers, etc)
            1. Tactics: New Versions, PERFORMANCE TUNING
        4. Introduces LB, Data Replication/Redundant, Computing Replication/Redundant
    2. Application
        1. Review the Sw Architecture -> apply patterns and best practices
        2. Code Optimization
        3. Algorithm Optimization
        4. Introduce: Cache, Queues, Parallel or Distributed computing
4. Measure your system related with Base Line Performance and SLA
    1. Testing

Caches

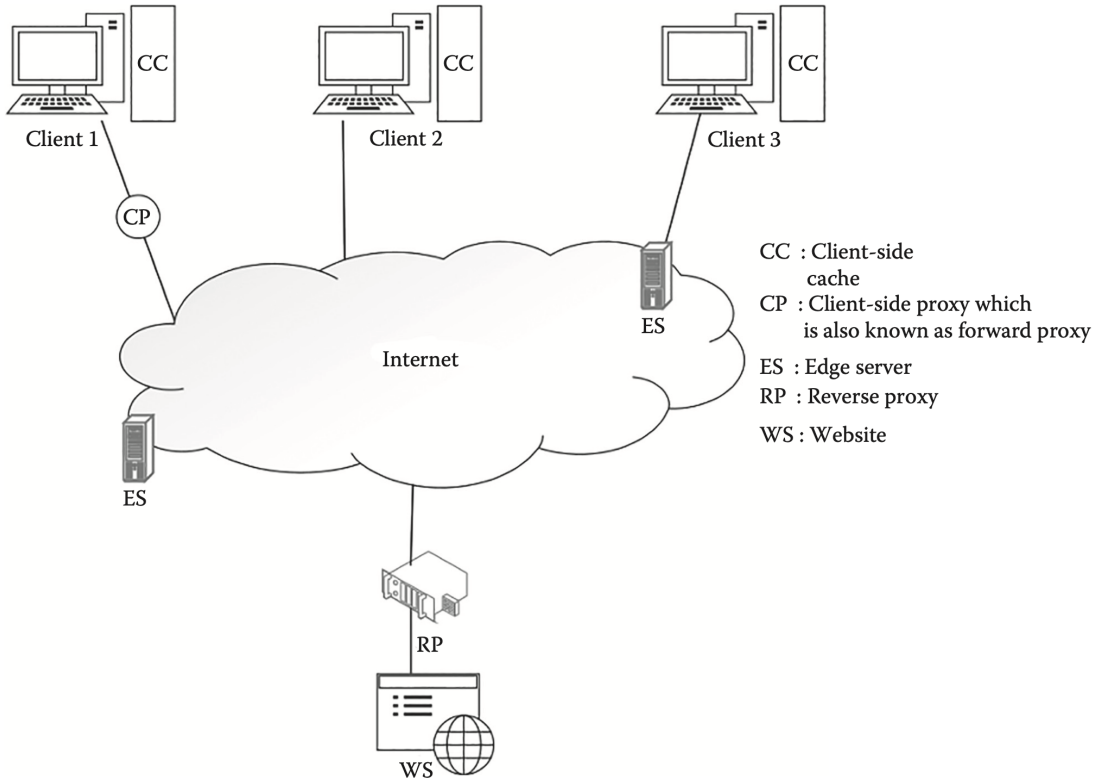
Caches

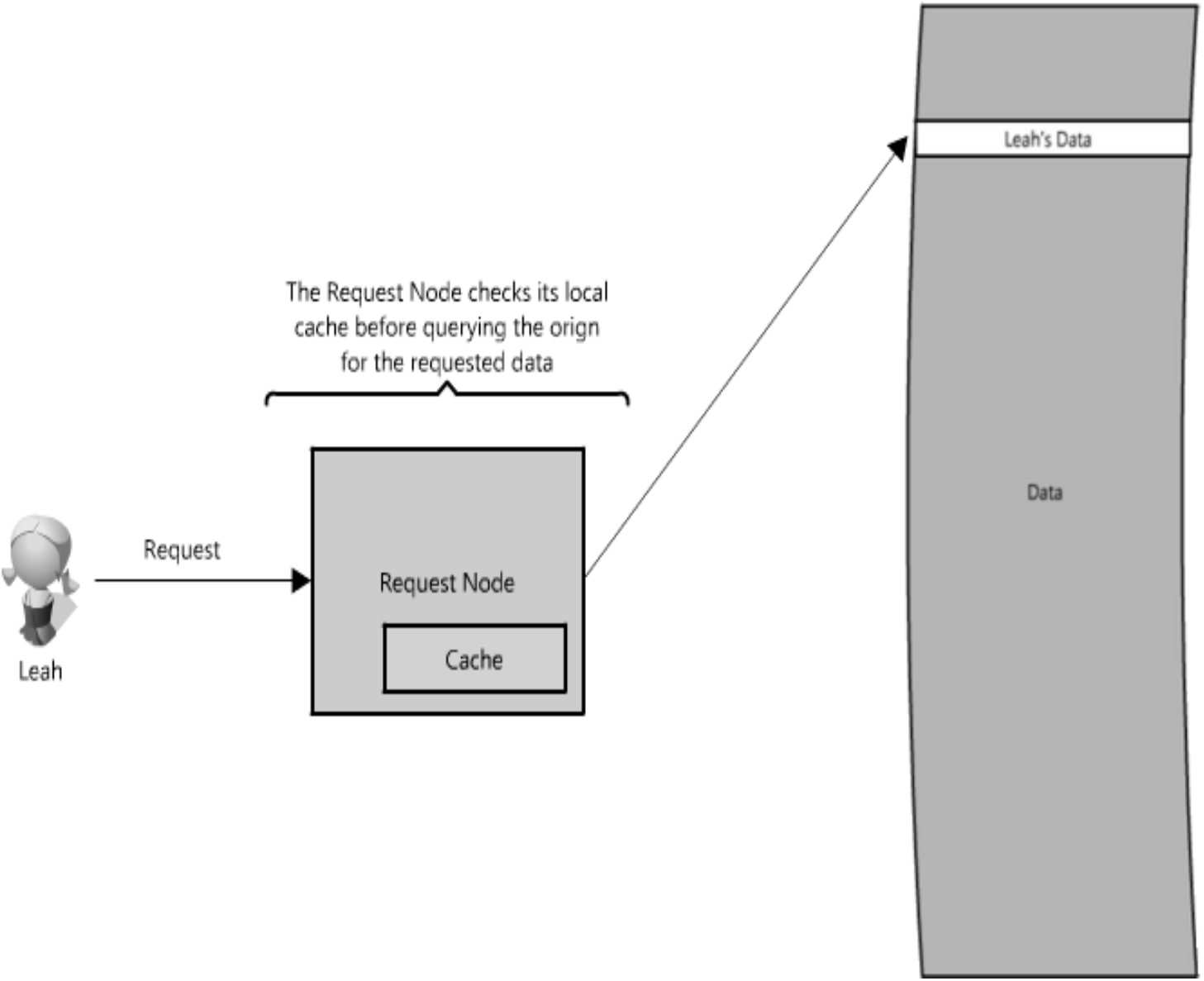
- Locality of reference principle: recently requested data.
- Many uses: hardware, OS, Web Browsers, webapps, etc.
- Cache: short-term memory
  - Amount of space
  - Data consistency
- Principle: nearest to the front end.

Caches

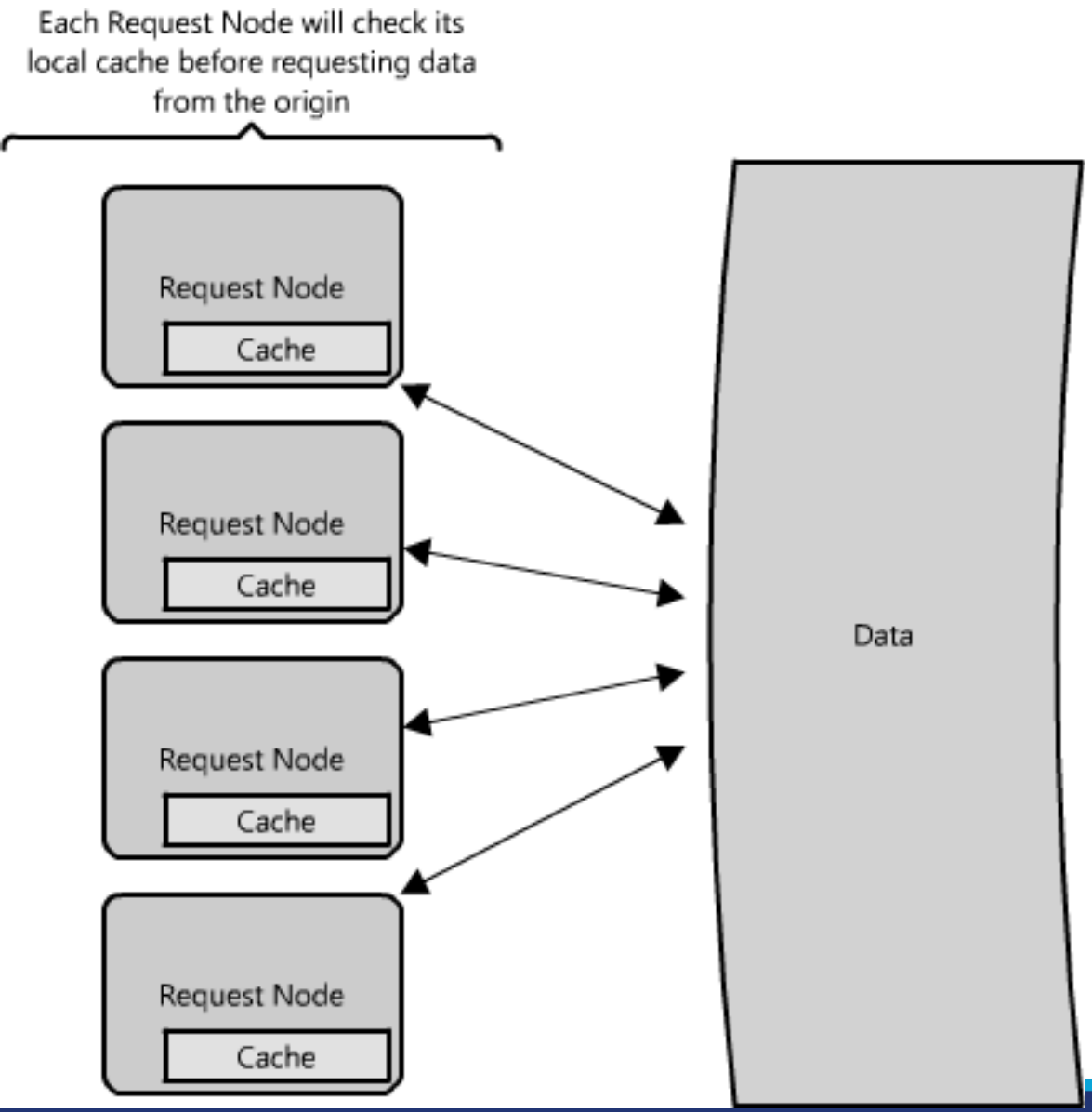
- ImgWebApp:
  - Insert a cache on request layer node
  - In memory or disk

Web caching locations





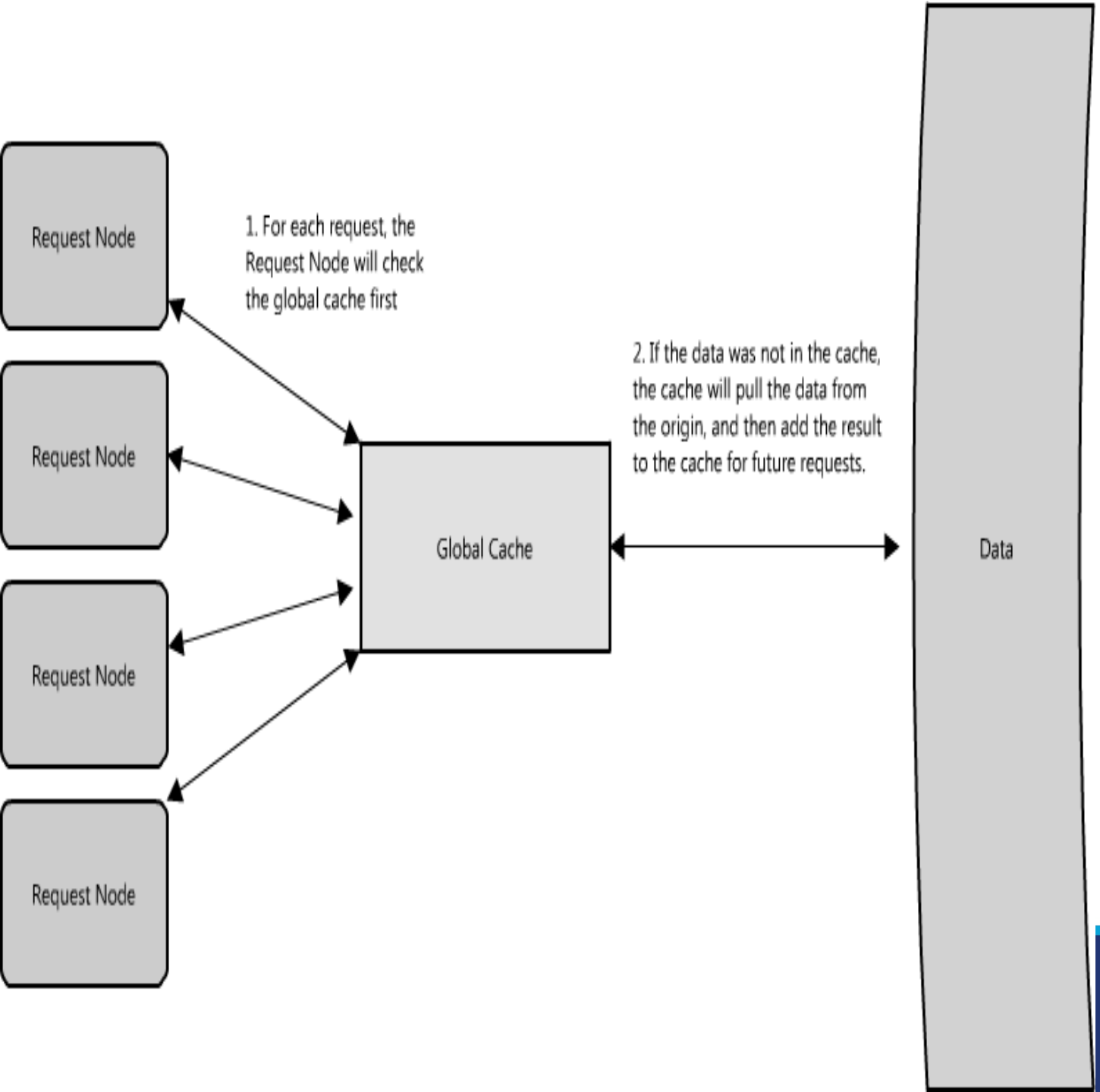
Inserting a cache on your request layer node



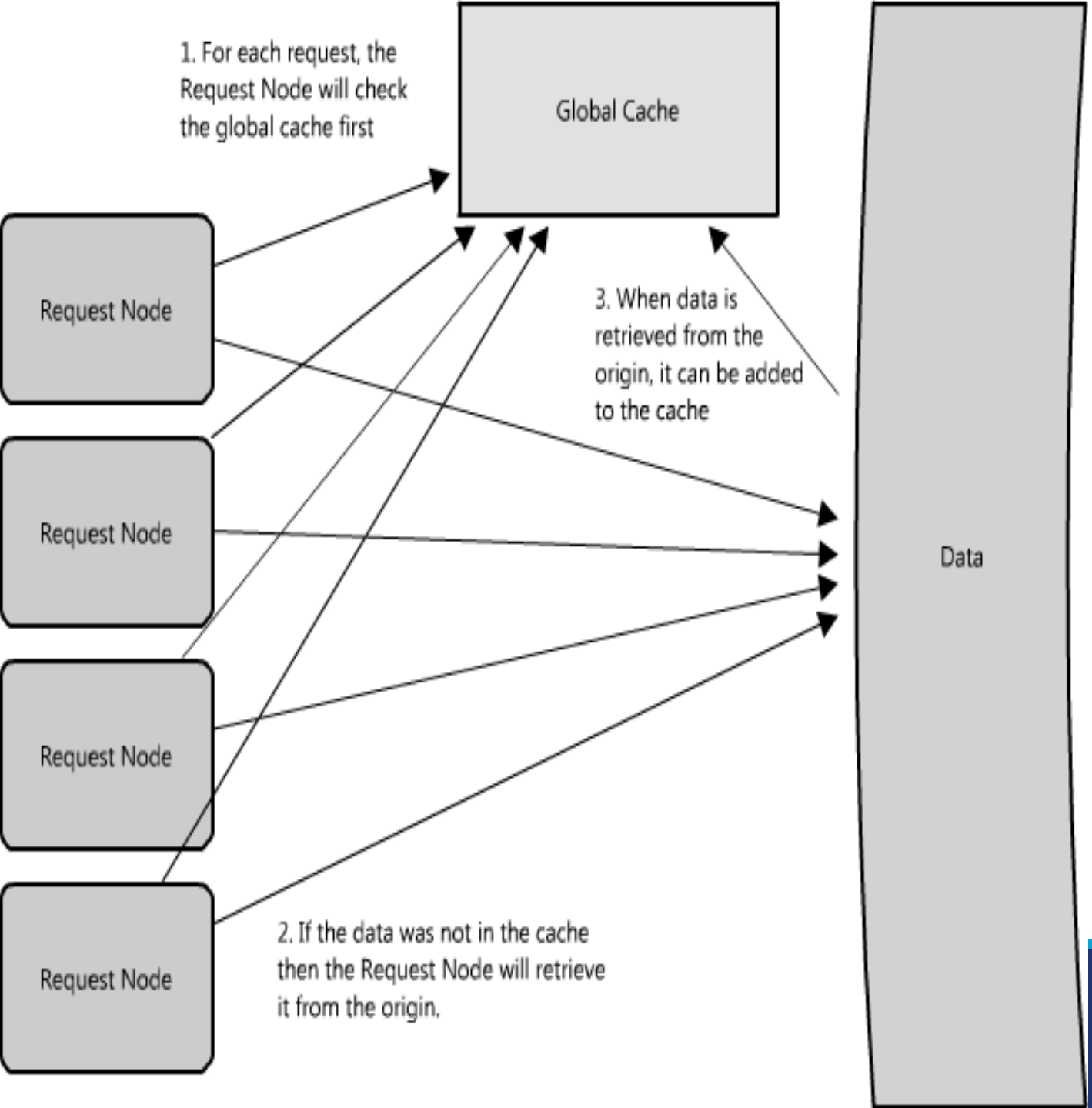
Multiple cache

Global Cache

- In multiple request nodes
  - Each node with each cache, but with LB?
  - Solution:
    - Global cache: all nodes use the same single cache space. Implies add a cache server.
    - Two common forms
      - Cache itself retrieve data when it miss



Global Cache where cache is responsible for retrieval

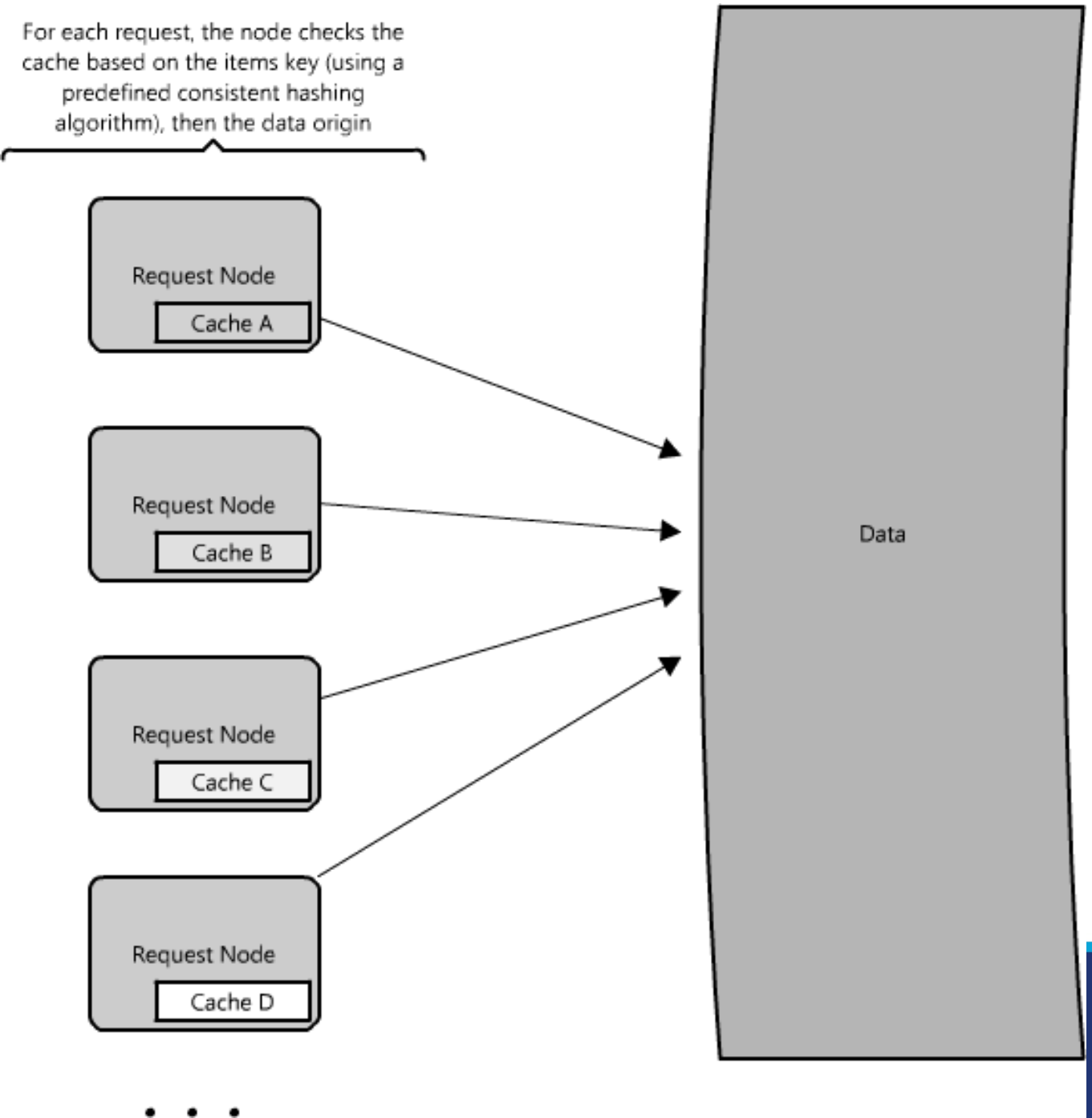


Global Cache where nodes are responsible for retrieval

Distributed Cache

- Each node own part of the cached data
- Cache divided with hash function
- Advantages: large caches
- Disadvantages: cache missing

Distributed Cache



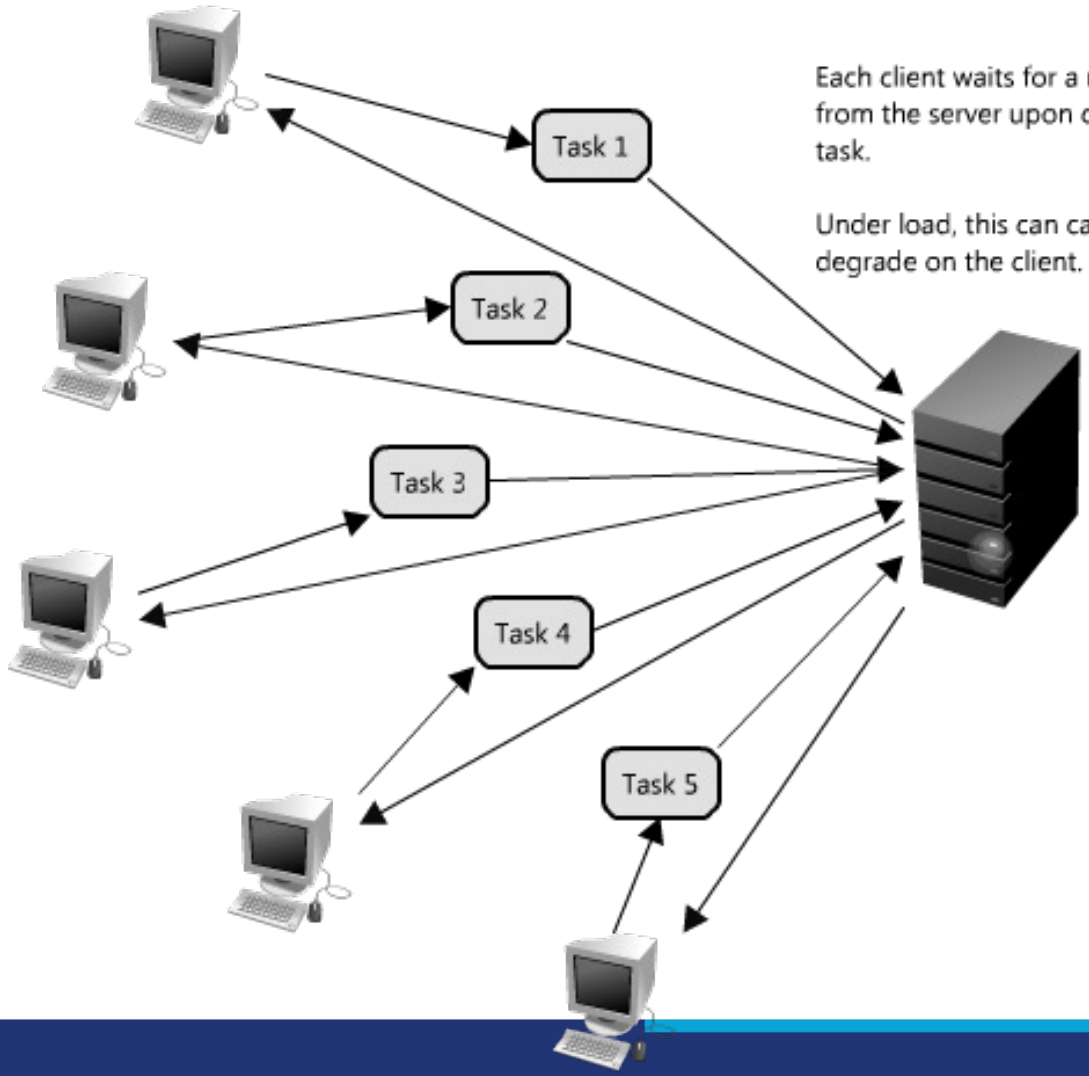
Queues

Async, MOMs

Queues

- Effective management of writes
- With high load of writes operations
  - Introduces asynchrony into the system with QUEUES
- Situation:
  - The server receives more requests than it can handle.
- Adding servers or load balancers don’t resolve this problem.

Synchronous Requests



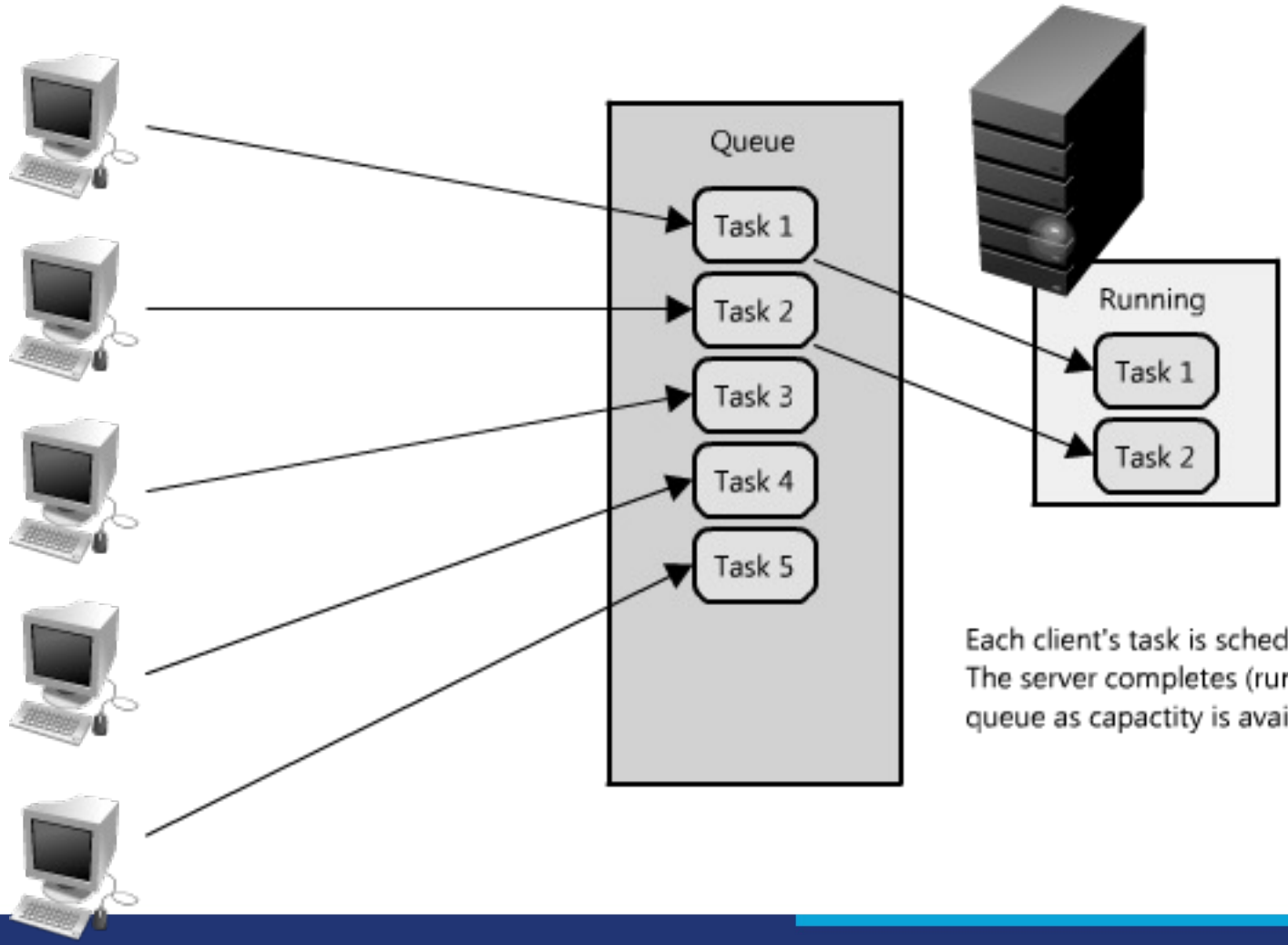
Each client waits for a response directly from the server upon completion of their task.

Under load, this can cause performance to degrade on the client.

Queues

- Queues introduces a decoupled model between client requests and task works.
- When a client submits task requests to a queue they don't wait for the results.
  - After the client request the result async.
- In a sync system, there is no differentiation between request and reply.
- In a async system
  - Clients sends request.
  - Do something else
  - Pulling for results
- Queue can also be used to support failure issues
  - Retry service requests
- Many systems: RabbitMQ, ActiveMQ, BeanstalkD, Zookeeper

Using queues to manage requests



Each client's task is scheduled on the queue. The server completes (runs) each one on the queue as capactity is available.

concepts - 1

- WPO (Web Performance Optimization) is a critical aspect of online success because it directly impacts the user experience, site usage, online revenue, and competitive advantage.
- **Assets** such as images, JS, and CSS provide a huge contribution of about 60% to the total page load time. These web components also consume a huge chunk of the page size.
- Performance should be used as a key design guideline from early stages of the project. This starts from the infrastructure architecture design and includes the design, development, testing, and deployment stages.

concepts – 2

- WPO includes these steps:
  - establishing performance objectives
  - performance modeling
  - establishing performance design guidelines
  - performance-based design
  - bottleneck analysis
  - continuous monitoring
  - performance governance.
- Performance modeling includes:
  - prioritizing business scenarios
  - workload modeling
  - identification of performance patterns.

concepts – 3

- **Key performance design guidelines** include:
  - Caching
  - distributed and parallel computing
  - lightweight design
  - asynchronous and on-demand data requests
  - Batching
  - standards-based technology
  - performance-based design and testing
  - modular design
  - omni-channel access
  - loose coupling
  - continuous and iterative build and testing.

concepts – 4

- Performance-based execution includes implementing performance design principles in all lifecycle stages of the project, starting with the requirements elaboration phase and continuing to the architecture and design phase, and development and validation phase.

concepts - 5

- Various dimensions of performance testing include:
  - load testing
  - process testing
  - infrastructure testing
  - omni-channel testing.
- A bottleneck creates a scenario that causes a data congestion and affects application performance.

concepts - 6

- A bottleneck can be identified using:
  - layer-wise decomposition
  - code profiling
  - call tracking
  - step-wise elimination.
- HTML 5 optimizations and RWD can be leveraged for providing a responsive web with faster performance on all devices.
- smart asset proxy, semantic progressive loading, and rapid rendering framework techniques for optimized delivery of static assets.
- Smart asset proxy employs different asset optimization techniques such as compression, asset caching, on-demand loading, DAC, and personalized content refreshing engine.

concepts - 7

- Progressive semantic asset loading iteratively loads various versions of the asset for optimal delivery.
- A rapid rendering framework provides components for both bottom-up and top-down performance optimization.
- A chunking strategy can be used for optimizing performance of content-driven pages.
- Internal system monitoring and end-user experience monitoring are required to assess the real-time insights into the performance of the web pages.
- Caching has to be enabled at various levels, including browser-based caching, web server caching, and object caching.

concepts - 8

- Because static assets such as images, JavaScript, and stylesheets play a major role in page performance, they must be optimally loaded using the following techniques:
  - Responsive design
  - On-demand loading
  - CSS sprites
  - Image compression
  - CDN caching
  - Merging and minifying
  - Web server caching
  - Elimination of duplicate assets
  - Asynchronous loading
  - Cloud hosting
  - Appropriate asset positioning
  - Distributed asset hosting

concepts - 9

- Web analytics can also be used for tracking key performance metrics such as page load time, asset load time, device-wise load time, and so on.
- A comprehensive performance governance framework should be designed to incorporate performance design guidelines in all phases of the project lifecycle.