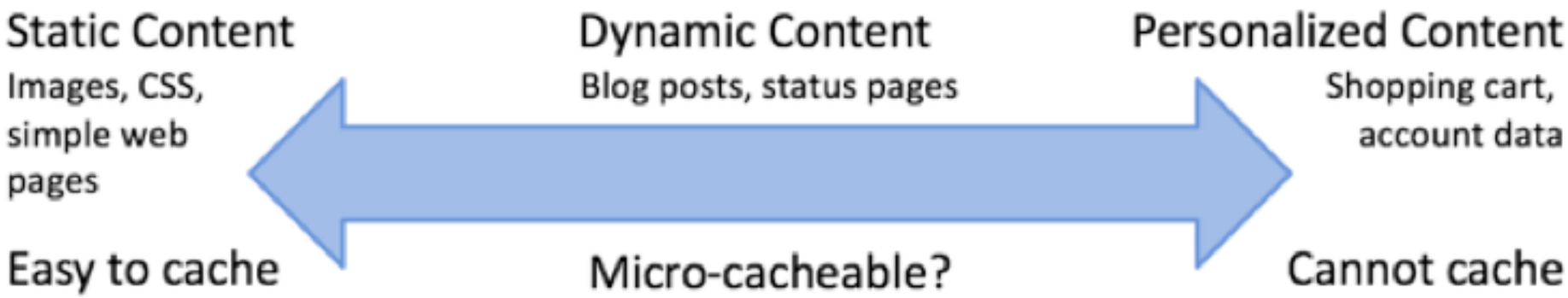
NGINX: Micro Caching in NGINX

- Caching **static content**, such as images, JavaScript and CSS files, and web content that rarely changes is a relatively straightforward process.
- Can we cache Dynamic Content too?



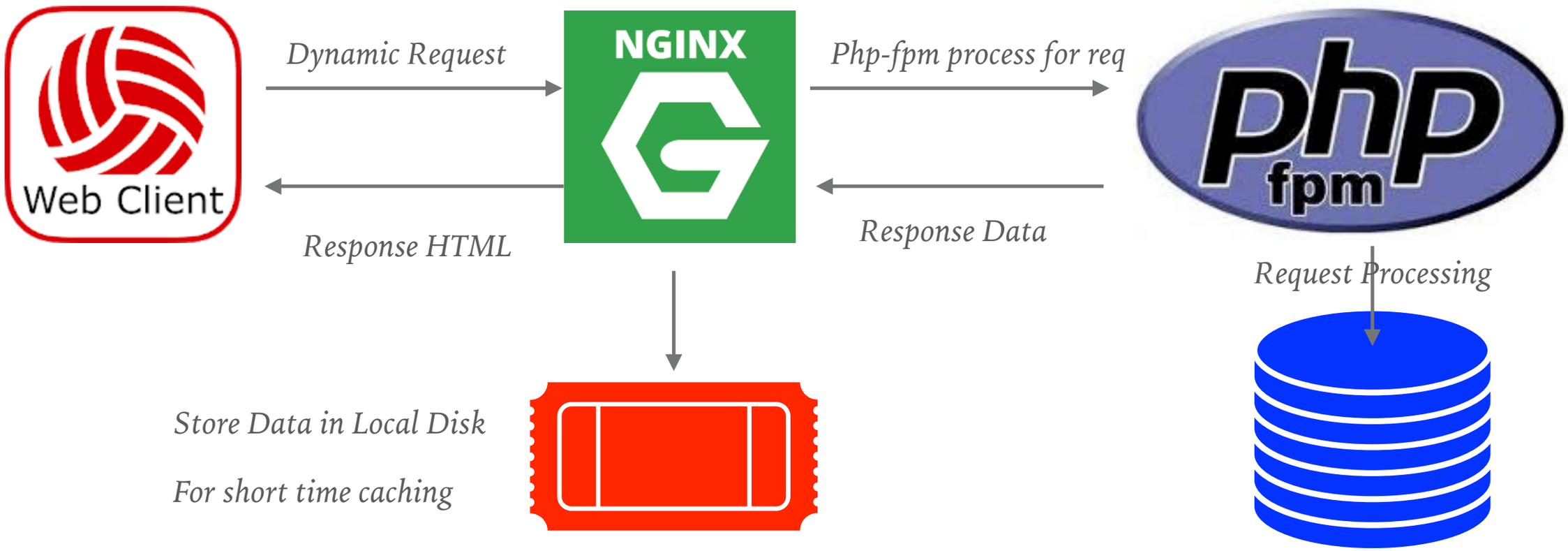
- Caching **personalized content** (that is, content customized for each user by the server application) is generally not possible, because the server's response differs for every request for the same resource.

- **Dynamic Content -** This Content can change unpredictably, but is not personalized for each user (or is personalized using JavaScript on the client device).
- Examples of dynamic content suitable for caching include:
  - The front page of a busy news or blog site, where new articles are posted every few seconds
  - An RSS feed of recent information
  - The status page for a continuous integration (CI) or build platform
  - An inventory, status, or fundraising counter
  - Lottery results
  - Calendar data
  - Personalized dynamic content is generated on the client device, such as advertising content or data ('Hello, *your name*') that is calculated using cookie data.

NGINX : Web-Server & Load Balancer

- **Micro Caching** is used to cache the dynamic content.
- *Microcaching* is a caching technique whereby content is cached for a very short period of time.
- **NGINX** uses a persistent disk-based cache located somewhere in the local file system.

Store Data in HTML Page



- **FastCGI** is a widely-used protocol for interfacing interactive applications such as **PHP** with web servers such as **NGINX**.
- The main advantage of **FCGI** is that it manages multiple CGI requests in a single process.
- Under NGINX, the FastCGI content cache is declared using a directive called **fastcgi_cache_path** in the top-level **http{}** context, within the NGINX configuration structure.
- User can also add the **fastcgi_cache_key** which defines a key (request identifier) for caching.

- **fastcgi_cache_path Directive Parameters -**
  - **/var/cache/nginx** – the path to the local disk directory for the cache.
  - **levels** – defines the hierarchy levels of a cache, it sets up a two-level directory hierarchy under /var/cache/nginx.
  - **keys_zone** (name:size) – enables the creation of a shared memory zone where all active keys and information about data (meta) are stored. Note that storing the keys in memory speeds up the checking process, by making it easier for NGINX to determine whether it's a MISS or HIT, without checking the status on disk.
  - **inactive** – specifies the amount of time after which cached data that are not accessed during the time specified get deleted from the cache regardless of their freshness.
  - **max_size** – specifies the maximum size of the cache.

- **fastcgi_cache_key Directive Parameters -**
  - NGINX uses them in calculating the key of a request. Importantly, to send a cached response to the client, the request must have the same key as a cached response.
  - **$scheme** – request scheme, HTTP or HTTPS.
  - **$request_method** – request method, usually “GET” or “POST”.
  - **$host** – this can be hostname from the request line, or hostname from the “Host” request header field, or the server name matching a request, in the order of precedence.
  - **$request_uri** – means the full original request URI.

Will see you in Next Lecture…

See you in next lecture …