Relatório

David Pinheiro *FCTUC, DEI*
Universidade de Coimbra
Coimbra, Portugal
2021246865@student.uc.pt

João Antunes *FCTUC, DEI*
Universidade de Coimbra
Coimbra, Portugal
uc2021220964@student.uc.pt

Gabriel Pinto *FCTUC, DEI*
Universidade de Coimbra
Coimbra, Portugal
uc2021220925@student.uc.pt

March 25, 2025

Software Architecture

GoogolClient

- **Role:** Acts as the user interface through which end-users interact with the search engine.
- **Functionality:** Provides search query input, displays results, and manages user sessions.
- **Importance:** As the point of contact for users, it must be both responsive and intuitive, ensuring quick interactions with the system.

Gateway

- **Role:** Serves as the central coordination hub.
- **Responsibilities:**
  - Routing search queries to the appropriate backend services.
  - Managing URL submissions.
  - Facilitating failover and load balancing.
- **Design Considerations:** Implements fault-tolerance mechanisms, such as maintaining an active list of indexing barrels and a recursive retry mechanism in case a barrel fails.

IndexStorageBarrel

- **Role:** Distributed storage of the indexed data.
- **Responsibilities:**
  - Managing a distributed index across multiple instances.
  - Storing persistent data using SQLite databases.
- **Indexing Approach:**
  - Each indexed word maps uniquely to a semicolon-separated list of URLs.

  - This design ensures that the storage is both space-efficient and supports rapid retrieval of search results.

– **Scalability:** The use of multiple IndexStorageBarrel instances allows for horizontal scaling, where the workload can be balanced and replicated across different storage units.

Downloader

– **Role:** Handles the crawling and indexing of web pages.

– **Functionality:**

  - Retrieves URLs from the centralized URLQueue.
  - Utilizes libraries like JSoup to process HTML, ensuring accurate extraction of textual content.
  - Normalizes the text by removing special characters, accents, and converting text to lowercase, which aids in building a consistent and reliable index.

– **Algorithm Complexity:** The sophisticated text processing algorithm ensures that the crawler can efficiently handle noisy or malformed data while maintaining high indexing accuracy.

URLQueue

– **Role:** Manages the URLs that are queued for crawling.

– **Key Features:**

  - Centralized URL management ensures that URLs are processed in a coordinated manner.
  - Persistence is achieved through an SQLite-based queue, ensuring that the system can recover from failures without losing URL processing progress.
  - Prevents duplicate URL processing, which is essential for maintaining efficiency and ensuring that the system does not waste resources on redundant tasks.

Communication Architecture

The system employs Java RMI to establish a robust remote procedure call mechanism between components. This design facilitates:

- **Decentralized Processing:** Each component operates independently while still being tightly integrated via RMI interfaces.
- **Defined Communication Channels:**
  1. **Gateway ↔ IndexStorageBarrel:** Handles search queries and indexing operations.
  2. **Gateway ↔ URLQueue:** Manages URL submissions and removals.
  3. **Downloader ↔ URLQueue:** Retrieves URLs in an efficient manner for further processing.
  4. **Downloader ↔ IndexStorageBarrel:** Transfers processed and normalized content for indexing.

By using RMI, the system not only supports remote method invocations but also abstracts away the complexities of network communication, making it easier to build and maintain distributed services.

Indexing Strategy

The indexing mechanism in Googol is designed with both performance and consistency in mind:

- **Distributed Indexing:** The index is partitioned across multiple IndexStorageBarrel instances, allowing for parallel processing and query handling.
- **Storage Technology:** SQLite databases are used for persistent storage. The lightweight nature of SQLite, combined with its reliability, makes it an ideal choice for a distributed index.
- **Data Structure:** Each word in the index is associated with a unique, semicolon-separated list of URLs. This ensures that lookups are fast and that the index remains compact and efficient.

Fault Tolerance and Load Balancing

Gateway Failover Mechanism

- **Redundancy:** The gateway maintains a dynamic list of active IndexStorageBarrel instances. This list is continually updated to reflect the health of each node.
- **Load Distribution:** The gateway randomly selects an active barrel for processing requests, effectively balancing the load across the network.
- **Resilience:** In the event of a failure, the gateway removes the failed barrel from its list and utilizes a recursive retry mechanism to ensure that no request goes unprocessed.

URLQueue Management

- **Persistence:** By relying on an SQLite-backed queue, the system ensures that the URL processing state is preserved even in the event of component crashes.
- **Transactional Integrity:** The queue operations are transactional, which guarantees that URLs are processed exactly once and helps avoid race conditions.
- **Efficiency:** Preventing duplicate URL processing ensures that system resources are optimally utilized and that the crawling process remains efficient.

Task division

To have an equivalent work between colleagues, we've distributed the tasks by this way (Not totally static tasks):

David Pinheiro: Gateway, Downloader
João Antunes: Reliable Multicast, Barrels
Gabriel Pinto: Client, URLQueue, and Report
But to be more precise everyone worked a little bit on everything :)

Test Description

The main tests were always conducted by indexing the page: `https://pt.wikipedia.org/`. Indexing from this page allows covering most of the test cases, excluding "limit" tests.

Below is our proposed acceptance test plan for the basic functionalities:

Some Test Cases

Based on the current implementation of the Googol search engine, the following test cases can be defined:

Test 1 – URL Indexing

Given: GoogolClient, URLQueue, an active Downloader, and at least one IndexStorageBarrel.
When: Client adds a URL for indexing, such as `https://pt.wikipedia.org/`
Then:

- The URL is added to the URLQueue
- The Downloader processes the URL
- Words from the page are indexed in the IndexStorageBarrel's SQLite database
- Links from the page are captured in the links graph

Test 2 – Word Search

Given: Test 1 successful; GoogolClient, Gateway, and at least one active IndexStorageBarrel.
When: Client searches for a term present in the indexed content
Then:

- A list of URLs containing the searched word is returned
- The search results are retrieved from the IndexStorageBarrel
- The Gateway handles load balancing and barrel selection

Test 3 – URL Queue Management

Given: Active URLQueue and Downloader
When:

1. Multiple URLs are added to the queue
2. Downloader starts processing URLs

Then:

- No duplicate URLs are added to the queue
- URLs are removed from the queue after processing
- The SQLite database correctly tracks URL processing

Test 4 – Backlink Discovery

Given: Multiple URLs indexed by the system
When: Client requests backlinks for a specific URL
Then:

- A list of URLs linking to the specified page is returned
- Links are retrieved from the links graph in the IndexStorageBarrel

Test 5 – Distributed Indexing Resilience

Given: Multiple IndexStorageBarrel instances
When:

1. An IndexStorageBarrel is taken offline
2. Indexing and search operations continue

Then:

- The Gateway automatically redistributes load to active barrels
- System continues to function without interruption

Test 6 – Text Normalization

Given: A web page with complex text content
When: Downloader processes the page
Then:

- Special characters are removed
- Accents are normalized
- Text is converted to lowercase
- Words are correctly split and indexed

Test 7 – RMI Communication Reliability

Given: Fully operational Googol search engine components
When:

1. Remote method calls are made between components
2. Network conditions vary

Then:

- RMI calls are handled gracefully
- Components can recover from temporary communication failures
- No data loss occurs during remote method invocations

Requisitos Funcionais e Tolerância a Falhas

Requisito	**Pontuação**	
Requisitos Funcionais (Total: 52)		
Indexar novo URL introduzido por utilizador	8	Pass
Indexar iterativamente ou recursivamente todos os URLs encontrados	8	Pass
Pesquisar páginas que contenham um conjunto de palavras	8	Pass
Páginas ordenadas por número de ligações recebidas de outras páginas	8	Pass
Consultar lista de páginas com ligações para uma página específica	8	Pass
Página de administração atualizada em tempo real	8	Pass
Resultados da pesquisa agrupados de 10 em 10	4	Pass
Grupos de 3 alunos: Índice particionado em duas metades (-12)	0	Fail
Tolerância a Falhas e Processamento Paralelo (Total: 38)		
A informação é idêntica em todos os storage barrels (reliable multicast)	6	Pass
Serviço correto se funcionar pelo menos um storage barrel e a gateway	6	Pass
Os storage barrels recuperam o seu estado se avariarem (crash)	6	Pass
Avaria de um storage barrel não tem efeito visível nos clientes	4	Pass
Balanceamento da carga nas pesquisas sobre os storage barrels	4	Pass
Os downloaders executam em paralelo	4	Pass
A gateway recupera de quaisquer avarias internas (downloaders e barrels)	4	Pass
Pedidos de indexação são respondidos apenas por um downloader	4	Pass