複数スレッドでの共有オブジェクト

レッスン

このコードは奇妙なバグを引き起こします。時々バグが発生し、時々発生しません。

これは、`translate_markdown_file` 関数、特に `translate_front_matter` 関数が、適切な同期なしに共有データ構造（辞書やリストなど）にアクセスして変更している可能性があるためです。複数のスレッドが同じデータに同時にアクセスして変更すると、競合状態が発生する可能性があります。競合状態は、データの最終的な状態がスレッドの実行順序に依存する場合に発生します。これにより、データの破損、予期しない動作、そしてあなたが観察している断続的なバグが発生する可能性があります。

これを修正するには、スレッド間で可変データを共有しないようにするか、共有データを保護するために適切な同期メカニズム（ロックなど）を使用する必要があります。この場合、`front_matter_dict` がその場で変更されており、これはスレッドセーフではありません。修正方法は、辞書を変更する前にコピーを作成することです。これは既にコードで行われていますが、なぜこれが必要なのかを理解することが重要です。

コンテキスト

```
  with concurrent.futures.ThreadPoolExecutor(max_workers=MAX_THREADS) as executor:
        futures = []
        for filename in changed_files:
            input_file = filename

            for lang in languages:

                print(f"Submitting translation job for {filename} to {lang}...")
                future = executor.submit(translate_markdown_file, input_file, os.path.join(f"_posts/{lang}", 
                futures.append(future)

        for future in concurrent.futures.as_completed(futures):
            try:
                future.result()
            except Exception as e:
                print(f"A thread failed: {e}")
```

修正前

```
def translate_front_matter(front_matter, target_language, input_file):
    print(f"  Translating front matter for: {input_file}")
    if not front_matter:
        print(f"  No front matter found for: {input_file}")
        return ""
    try:
        front_matter_dict = {}
        if front_matter:
            front_matter_dict = yaml.safe_load(front_matter)
            print(f"  Front matter after safe_load: {front_matter_dict}")
        if 'title' in front_matter_dict:
            print(f"  Translating title: {front_matter_dict['title']}")
            if not (input_file == 'original/2025-01-11-resume-en.md' and target_language in ['zh', 'fr']):
                if isinstance(front_matter_dict['title'], str):
                    translated_title = translate_text(front_matter_dict['title'], target_language)
                    if translated_title:
                        translated_title = translated_title.strip()
                        if len(translated_title) > 300:
                            translated_title = translated_title.split('\n')[0]
                        front_matter_dict['title'] = translated_title
                        print(f"  Translated title to: {translated_title}")
                    else:
                        print(f"  Title translation failed for: {input_file}")
                else:
                    print(f"  Title is not a string, skipping translation for: {input_file}")
            else:
                print(f"  Skipping title translation for {input_file} to {target_language}")
        # Always set lang to target_language

        # Determine if the file is a translation
        original_lang = 'en' # Default to english
        if 'lang' in front_matter_dict:
            original_lang = front_matter_dict['lang']

        if target_language != original_lang:
            front_matter_dict['lang'] = target_language
            front_matter_dict['translated'] = True
            print(f"  Marked as translated to {target_language} for: {input_file}")
        else:
            front_matter_dict['translated'] = False
            print(f"  Not marked as translated for: {input_file}")
        result = "---\n" + yaml.dump(front_matter_dict, allow_unicode=True) + "---"
        print(f"  Front matter translation complete for: {input_file}")
        return result
    except yaml.YAMLError as e:
        print(f"  Error parsing front matter: {e}")
        return front_matter
```

修正後

```
def translate_front_matter(front_matter, target_language, input_file):
    print(f"  Translating front matter for: {input_file}")
    if not front_matter:
        print(f"  No front matter found for: {input_file}")
        return ""
    try:
        front_matter_dict = {}
        if front_matter:
            front_matter_dict = yaml.safe_load(front_matter)
            print(f"  Front matter after safe_load: {front_matter_dict}")

        front_matter_dict_copy = front_matter_dict.copy()

        if 'title' in front_matter_dict_copy:
            print(f"  Translating title: {front_matter_dict_copy['title']}")
            if not (input_file == 'original/2025-01-11-resume-en.md' and target_language in ['zh', 'fr']):
                if isinstance(front_matter_dict_copy['title'], str):
                    translated_title = translate_text(front_matter_dict_copy['title'], target_language)
                    if translated_title:
                        translated_title = translated_title.strip()
                        if len(translated_title) > 300:
                            translated_title = translated_title.split('\n')[0]
                        front_matter_dict_copy['title'] = translated_title
                        print(f"  Translated title to: {translated_title}")
                    else:
                        print(f"  Title translation failed for: {input_file}")
                else:
                    print(f"  Title is not a string, skipping translation for: {input_file}")
            else:
                print(f"  Skipping title translation for {input_file} to {target_language}")
        # Always set lang to target_language
        front_matter_dict_copy['lang'] = target_language
        front_matter_dict_copy['translated'] = True

        result = "---\n" + yaml.dump(front_matter_dict_copy, allow_unicode=True) + "---"
        print(f"  Front matter translation complete for: {input_file}")
        return result
    except yaml.YAMLError as e:
        print(f"  Error parsing front matter: {e}")
        return front_matter
```