Evaluation of Different Storage Formats for Geospatial Data Caches Including a Sample Implementation to Enable ArcGIS Portal to Export Caches

Introduction

Exporting cached maps might take days. Airbus intended to analyze the performance of: *Compact Cache, GeoPackage, MBTiles* & *LuciadFusion*, by developing an algorithm in a web application based on WPS.

Compact Cache & *LuciadFusion* contain multiple files, while *GeoPackage* & *MBTiles* are made of single database file. It is tested whether the combination of multiprocessing & multithreading optimizes Compact Cache to achieve a similar speed to the database formats.

Algorithm

•**If drew polygons, transform coordinates from EPSG 3857 to 4326:** the geoportal's map is Web Mercator 3857 → 4326.

•**Convert from 4326 into tiles mechanism and from tiles into 4326:** create bbox (tile mechanism) around polygons (4326).

Inside bbox, check if tile (→ 4326) is intersected with polygon → (saved as tiles), 3 loops are executed: **Z (zoom) level, Y & X coordinates of each tile → Higher Z → More iterations of (X, Y) → More tiles to be calculated, More intersection tests.** Three polygons are processed with different LOD - level of detail: world - lowest Z, national - higher Z, regional - highest Z.

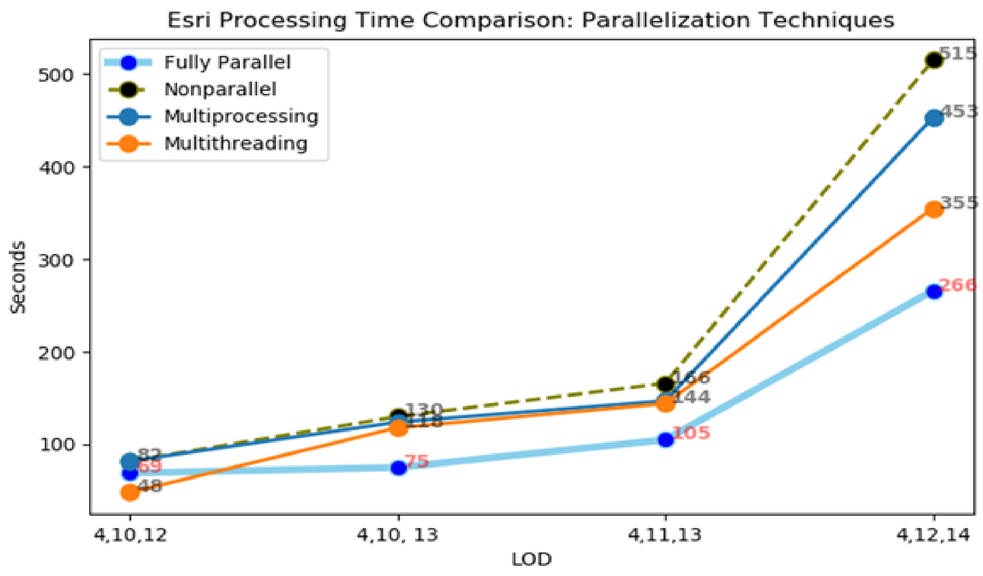

Fig. 1: Paralelization Techniques

Karlsruhe University of Applied Sciences
Faculty IMM • Geomatics Master
Author: Yair Preiss E-Mail-Adresse: pyair86@gmail.com
Supervisor: Prof. Dr.-Ing. Reiner Jäger (HSKA)
Prof. Dr. Ángel Marqués-Mateu (UPV)
Dipl.-Inf. (FH) Andreas Degwerth (Airbus)

Check if tile exists in cache and copy tiles: generate files on server, zip files, open file in browser by JavaScript & send link by email.

Each cache format contains a distinct data structure → **GeoPackage & MBTiles:** store tiles in a list → insert list into database per transaction. Locked database → no parallelization possible

LuciadFusion: Luciad provides special methods to store tiles faster → more effective under high zoom → employs Java → wrapped for Python → multiprocessing isn't possible through Pyjnius, which wraps the code.

Esri Compact Cache: Mapproxy blocks multiprocessing → assign relevant tiles to each bundle file → split bundles to be processed in multiple cores. Python's GIL blocks multithreading → use Numba's nogil decorator → in each core, chunks of data are processed in parallel by multiple threads.

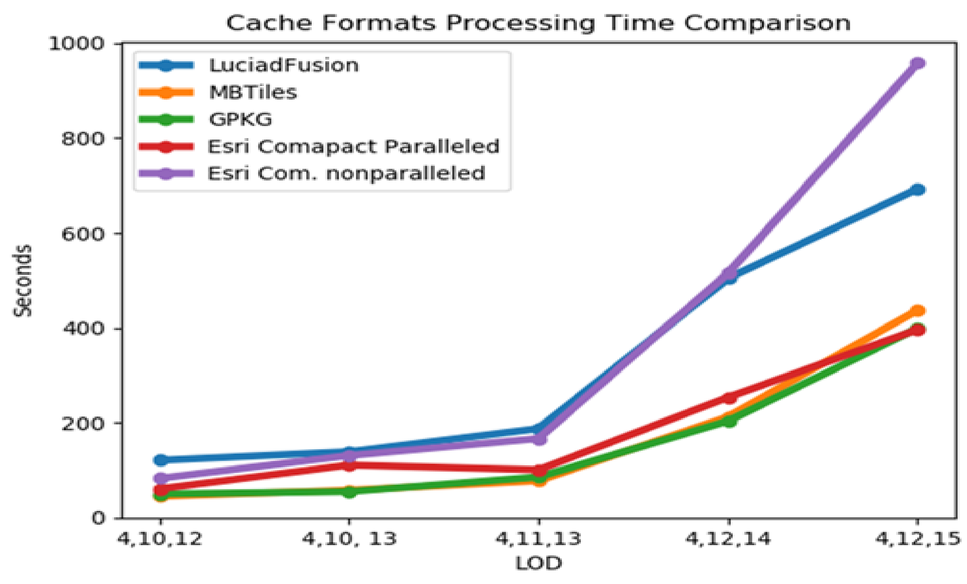

Fig. 2: Cache Formats Processing Time Comparison

Conclusion

When Compact Cache is optimized, it's at least as fast as the database formats - by combination of multiprocessing & multithreading, else it is the slowest → one should consider employing database logic, or combination of multiprocessing & multithreading to export cache files.