Distributed generation of image pyramids for web map services

## D. Eggert, M. Sips, D. Dransch

daniel.eggert@gfz-potsdam.de
mike.sips@gfz-potsdam.de
doris.dransch@gfz-potsdam.de

HELMHOLTZ CENTRE POTSDAM
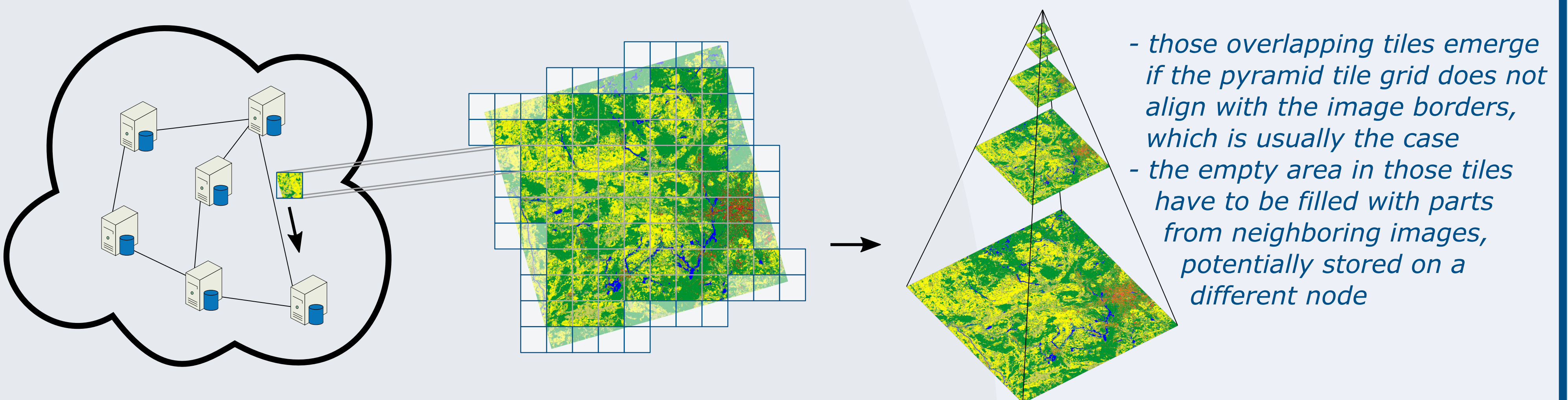**GFZ GERMAN RESEARCH CENTRE FOR GEOSCIENCES**

Introduction:

Web map services (WMS) are a common technique to publish high-resolution satellite images, allowing fast pan and zoom operations on the client-side. In order to reduce the computational costs on the server-side, the image data is usually processed into a tiled image pyramid in advance. The generation of such an image pyramid from a big high resolution source image, normally distributed over multiple storage and processing nodes in a cloud environment, causes a high network traffic when executed by a single node. Furthermore this single processing node has to meet high requirements regarding computation power, main memory and storage capabilities. We present a new approach that distributes the generation of the image pyramid across all nodes, reducing the induced network traffic by exploiting data locality and eliminating the need of a powerful processing node.

Problem: *all chunks of the distributed image are transferred to a single node to generate an image pyramid*

- stresses the network
- high requirements for the single processing node
- time consuming
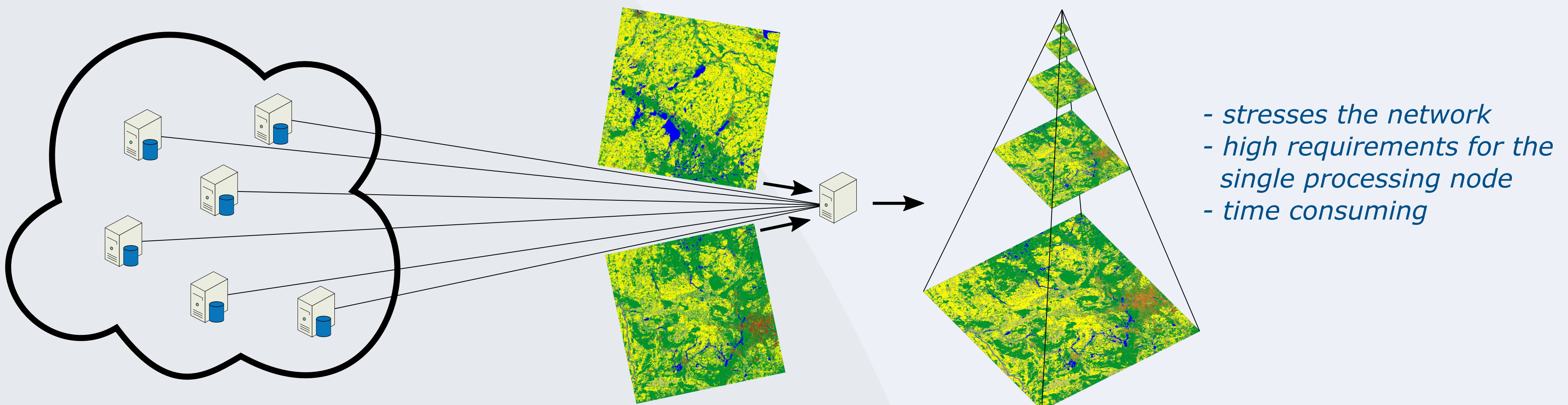
Proposed Solution: *transfer only small overlapping tiles in a distributed environment*

- those overlapping tiles emerge if the pyramid tile grid does not align with the image borders, which is usually the case
- the empty area in those tiles have to be filled with parts from neighboring images, potentially stored on a different node

MapReduce Workflow:

Warp *to target projection*	**Cut** *according to pyramid tile grid*	**Merge** *overlapping tiles*	**Resample** *to zoom level*
operator type: <map> data access: <local only>	operator type: <map> data access: <local only>	operator type: <reduce> data access: <global>	operator type: <map> data access: <local only>

Conclusion:

- eliminates the need of a powerful processing node -> increased scalability
- distributes the processing over multiple nodes -> faster generation
- exploits data locality -> reduced network traffic

HELMHOLTZ