LIZARDTECH™
a celartem Company

The JPEG 2000 & GML Connection

Part III – Advanced Topics

GeoWeb 2006
Vancouver BC

Michael P. Gerlek
LizardTech

We Make Imaging Work. Everywhere.

Agenda

- *Introductions and perspectives*
- ***Part I*** *– Core Technology*
- ***Part II*** *– Metadata*
- **Part III – Advanced Topics**
  - **WCS + GMLJP2**
  - **JPIP**
  - **WCS + JPIP**
  - **Databases**
- *Opinions and Conclusions*

LIZARDTECH™
a celartem Company

Agenda

- **W*S + GMLJP2**
- *JPIP*
- *WCS + JPIP*
- *Databases*

Quick review: W*S

OGC Web Services

- "W*S": providing geospatial data, in response to queries
  - WMS – Web Map Service
  - WCS – Web Coverage Service
  - WFS – Web Feature Service

All use classical HTML request/response paradigm

Web Map Service (WMS)

Conceptually: front-end for a GIS system

- returns "maps"
  - generally to a human-like client
  - performs layering of vector, raster data
- input: layer name, CRS, bounding box, …
  - CGI strings
- output: image (JPG, PNG, …)

Web Coverage Service (WCS)

Conceptually: interface to an image archive

- returns coverage data
  - as GeoTIFF, NITF, GML, DTED, HDF-EOS
- uses XML/GML for queries and responses
  - "describe coverage"
  - "get coverage"
    - bounding box, reprojection, …

Web Feature Service (WFS)

Conceptually: interface to a geo database

- uses XML/GML for queries and responses
  - "describe feature type"
  - "get feature"
    - query uses "filters" to describe which properties to return from which features
  - return is a GML document

WMS vs. WCS vs. WFS

- each designed for a different purpose
  - though some overlap
  - WCS and WFS both can return "coverages"
- bigger picture examples:
  - a WFS can serve to another WFS
  - a WMS can front for a WFS and a WCS
  - …

Uses of GMLJP2

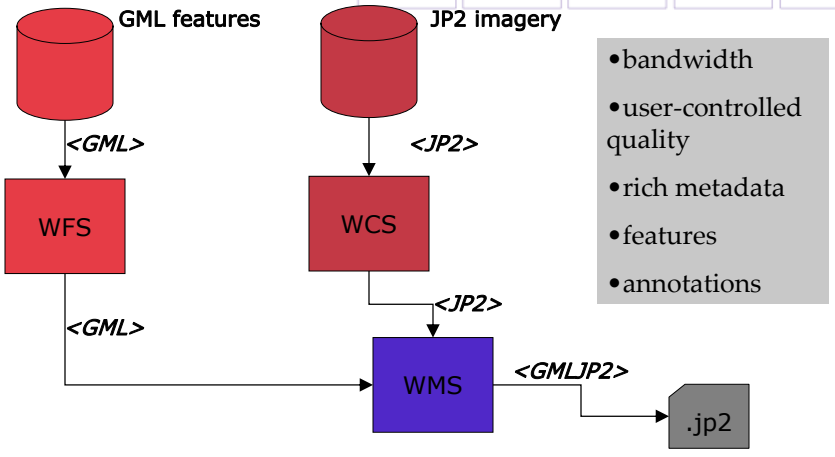
Recall: JP2/GMLJP2 gives us…

- better image-format features (compression, pyramids, …)
- georeferencing
- arbitrary feature data

- back-end storage (databases, flat files
- interchange format
- stand-alone output file

OGC/OWS-4 projects…

Example



- bandwidth
- user-controlled quality
- rich metadata
- features
- annotations

Agenda

- *WCS + GMLJP2*
- **JPIP**
- *WCS + JPIP*
- *Databases*

A Problem?

Consider workflows that are bandwidth-constrained

- small pipes…
  - disaster response
  - battlefield conditions
- or big data…
  - GoogleEarth / VirtualEarth / WorldWind
  - too much data for twitchy users

Incremental delivery of imagery?

The JPIP Protocol

"streaming" … "progressive transmission" … better use of limited bandwidth … improved viewing experience

JPIP: a system for transmitting JP2 data

- protocol for serving compressed data
  - piece-wise file transfer
  - client requests scenes, renders data as becomes available
- "low-resolution" data sent and displayed first
  - image can be rendered at any point
  - more "quality" (detail) transferred, if you "hold still"
- client can cache data, to further minimize downloads
- (and the metadata, too)

JPIP Technology

Image decomposition happens along several dimensions…

RESOLUTION

SPATIAL

QUALITY

Protocol allows for transmission of individual *precincts* –

Image is rendered based on whatever data client has received so far

Databins

Data is transmitted in terms of "databins":

- **precinct databin:** all the packets from one precinct; canonically indexed by codestream number and precinct number
- **main header databin:** the codestream header marker data from the codestream; indexed by codestream number
- **meta databins:** all non-codestream boxes; indexed by (essentially) box number as determined lexically from the file

Any JP2 file can be decomposed into a set of databins

- without any data loss
- complexity of decomposition limited to that of file parsing

Demo

- Image data streaming from Seattle to Vancouver

(visualize demo here)

Think about what this means: progressive transmission, low-bandwidth links, intermediate node caching, …

Agenda

- *WCS + GMLJP2*
- *JPIP*
- **WCS + JPIP**
- *Databases*

JPIP for Geo

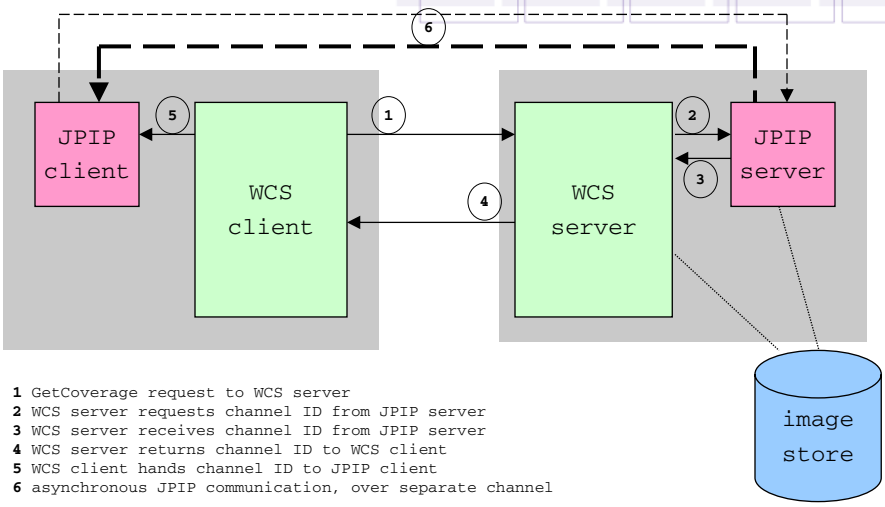
But:

- JPIP requests are not expressed with geo coords
- We already have a web service for "get pixels"
- WCS only knows how to do "static" images
- …

Can we make JPIP be a WCS return type?

another OGC/OWS-4 project

Architecture (DRAFT)



1 GetCoverage request to WCS server
2 WCS server requests channel ID from JPIP server
3 WCS server receives channel ID from JPIP server
4 WCS server returns channel ID to WCS client
5 WCS client hands channel ID to JPIP client
6 asynchronous JPIP communication, over separate channel

Problems to overcome

- Stateful vs. stateless..?
- Cache modeling..?
- "Best effort" response semantics..?
- "GMLJPIP"..?
  - Streamed metadata..?
  - Clipped metadata..?

Agenda

- *WCS + GMLJP2*
- *JPIP*
- *WCS + JPIP*
- **Databases**

"We get letters…"

Customers tell us:

- all their other (non-raster) data is in databases
- managing DBs is easier than large flat file systems
- some DBs have support for raster
  - but only raw (too big!) or jpeg (too blocky!)

- Also:
  - recall our discussion of progression orders
  - significant performance problems known with random access into large JP2 files

Precincts!

Precincts are a fully indexed storage unit

- codestream number
- spatial region
- resolution level
- …

So…

- decompose the file into precincts
- and make the DB manage them for us

JP2 in a Database

Advantages:

- no data loss
- solves the random-access performance problem
- no "artificial" compression artifacts due to tiling
- can extract pixels
- or can extract JP2-encoded data
  - (maps to JPIP really nicely!)

Disadvantage:

- the indexing scheme is nontrivial:
  - DB needs logic to map request scene to precinct index

JP2 Databases Today?

LizardTech and Oracle partnership

- *SpatialExpress plugin for Oracle 10g*
- performance numbers look good
  - example: against pyramid-generation time
  - example: against flat-files
  - example: time-to-extract