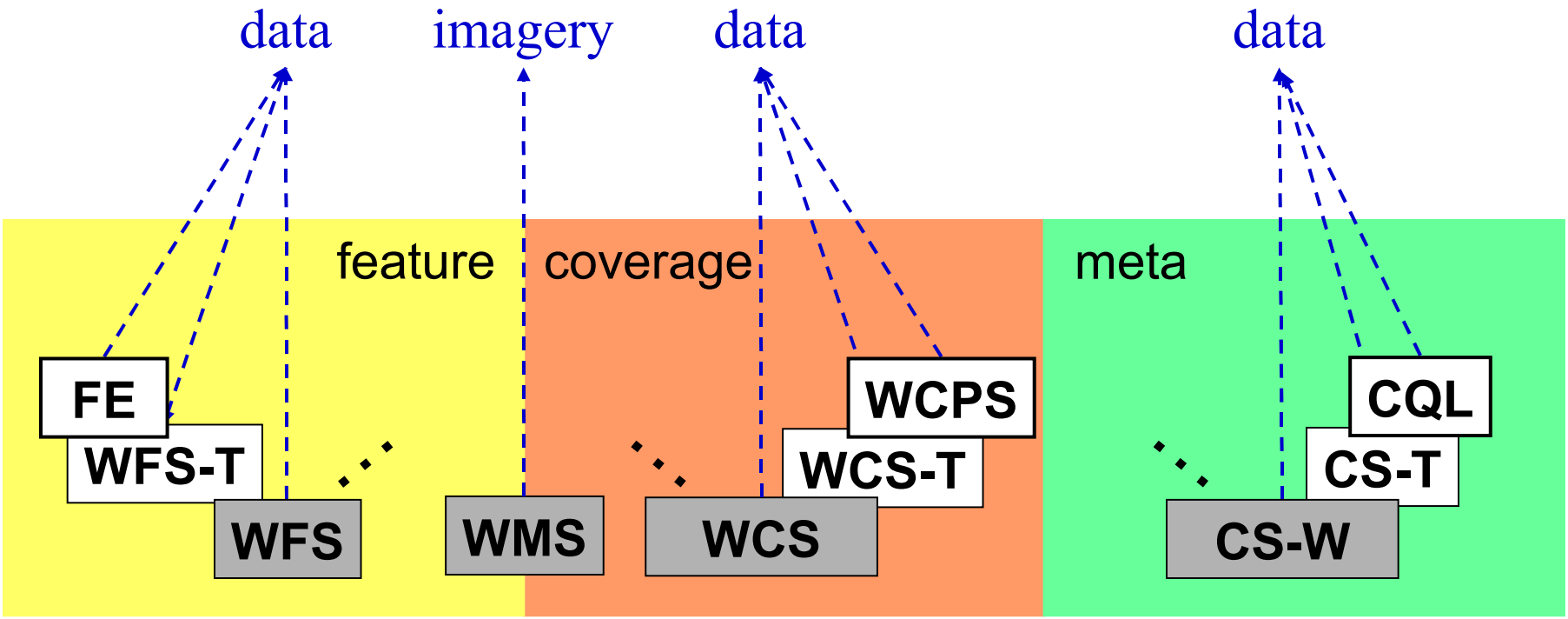
JACOBS UNIVERSITY

HMA-FO Project Meeting

ESRIN / Frascati, 2009-09-18

(Part of) The OGC Quilt

JACOBS UNIVERSITY



WCPS

JACOBS UNIVERSITY

- = standard for multi-dimensional raster language
  - Web Coverage Service (WCS) & Web Processing Service (WPS) embeddings
- „SQL for coverages“: ad-hoc navigation, extraction, aggregation, analysis
  - Formal mathematical evaluation model
  - Expressive power: image & signal processing, statistics …except recursion →safe

```
for cov1 in ( coverageList ),
    ...
    covn in ( coverageList )
[ where condition(cov1,…,covn) ]
return processingExpr(cov1,…,covn)
```

Example

JACOBS UNIVERSITY

- "From MODIS scenes M1, M2, and M3, the absolute of the difference between red and nir, in HDF-EOS"

```
for c in ( M1, M2, M3 )
return
    encode(
        abs( c.red - c.nir ),
        "hdf"
    )
```

```
(hdf_A,
 hdf_B,
 hdf_C)
```

Example

JACOBS UNIVERSITY

- "From MODIS scenes M1, M2, and M3, the absolute of the difference between red and nir, in HDF-EOS"
  - …but only those where nir exceeds 127 somewhere

```
for c in ( M1, M2, M3 )
where
    some( c.nir > 127 )
return
    encode
        abs( c.red - c.nir ),
        "hdf"
    )
```

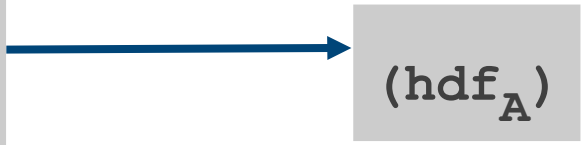
(hdf_A, hdf_C)

Example

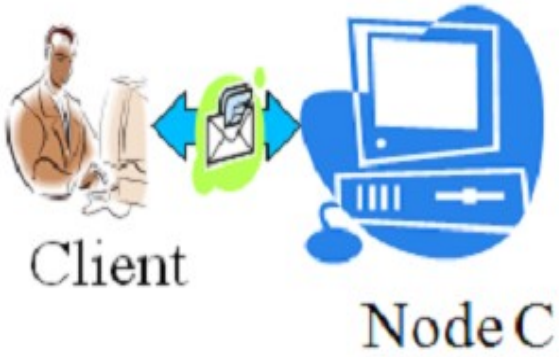
JACOBS UNIVERSITY

- "From MODIS scenes M1, M2, and M3, the absolute of the difference between red and nir, in HDF-EOS"
  - …but only those where nir exceeds 127 somewhere
  - …inside region R

```
for c in ( M1, M2, M3 ),
    r in ( R )
where
    some( nir > 127 and R )
return
    encode
        abs( c.red - c.nir ),
        "hdf"
    )
```



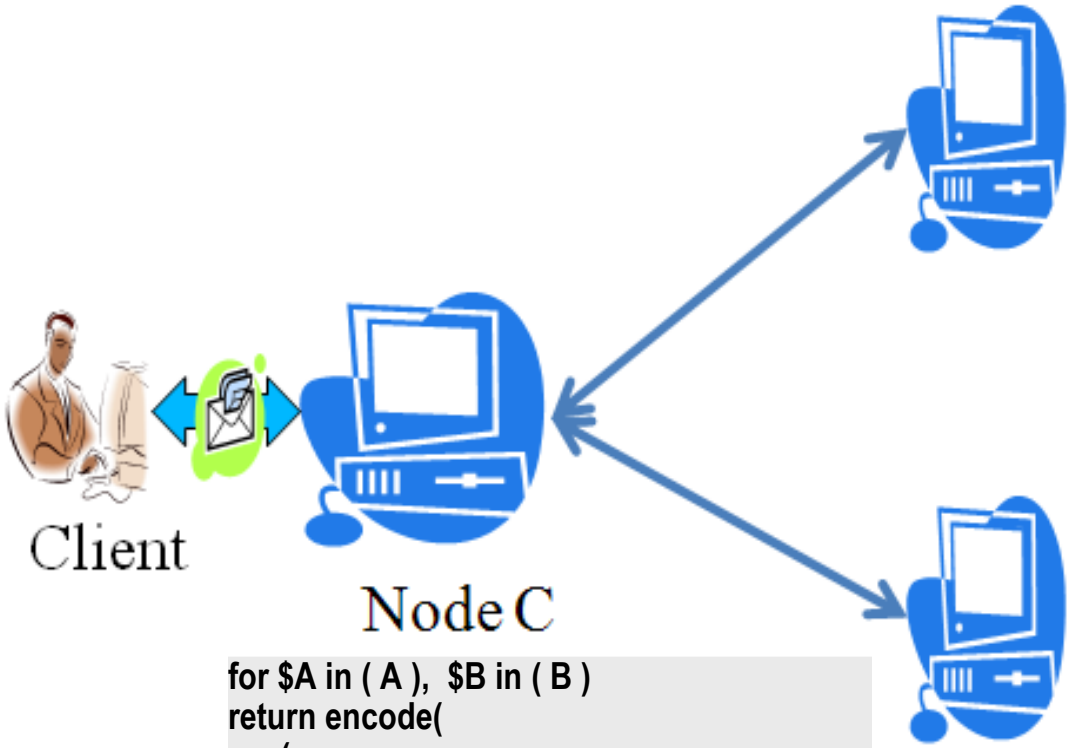
P2P Distributed Processing: Principle




```
for $A in ( A ),
      $B in ( B )
return
    encode(
      (
          ( ($A.nir - $A.red) / ($A.nir + $A.red) )
        - ( ($B.nir - $B.red) / ($B.nir + $B.red) )
      ) [ x(x1:x2), y(y1:y2) ],
      "tiff"
    )
```

P2P Distributed Processing: Principle

JACOBS UNIVERSITY




```
for $A in ( A ),  $B in ( B )
return encode(
    (
       ( ($A.nir - $A.red) / ($A.nir + $A.red) )
     - ( ($B.nir - $B.red) / ($B.nir + $B.red) )
    ) [ x(x1:x2), y(y1:y2) ],
    "tiff" )
```

```
for $A in ( A )
return encode(
    (
       ($A.nir - $A.red)
      /($A.nir + $A.red)
    ) [x(x1:x2),y(y1:y2)],
    "array" )
```

```
for $B in ( B )
return encode(
    (
       ($B.nir - $B.red)
      /($B.nir + $B.red)
    ) [x(x1:x2),y(y1:y2)],
    "array" )
```