Satellite Science Data Processing with PostgreSQL

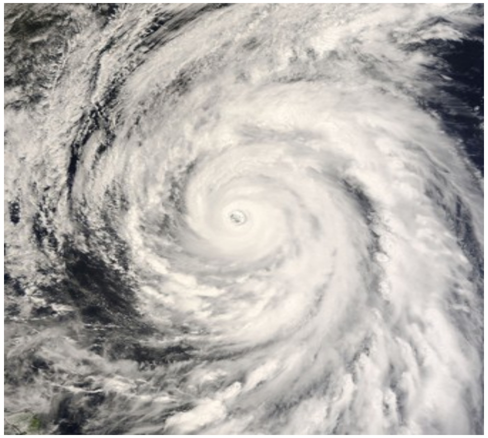
Curt Tilmes

Curt.Tilmes@nasa.gov

PGCon 2008
May 22, 2008

- Background – MODIS and Ozone Processing
- Science Data Processing
- Architecture Evolution
- Metadata and Archiving
- Spatial Searching
- Reprocessing
- Algorithms and Production Rules
- Provenance Tracking
- Process on Demand

- MODIS
  - Moderate Resolution Imaging Spectroradiometer
  - On Terra (1999) and Aqua (2002) spacecraft
  - Views (most of) the earth at 250m, 500m and 1km resolution every day in 36 spectral bands
  - MODAPS Processing System

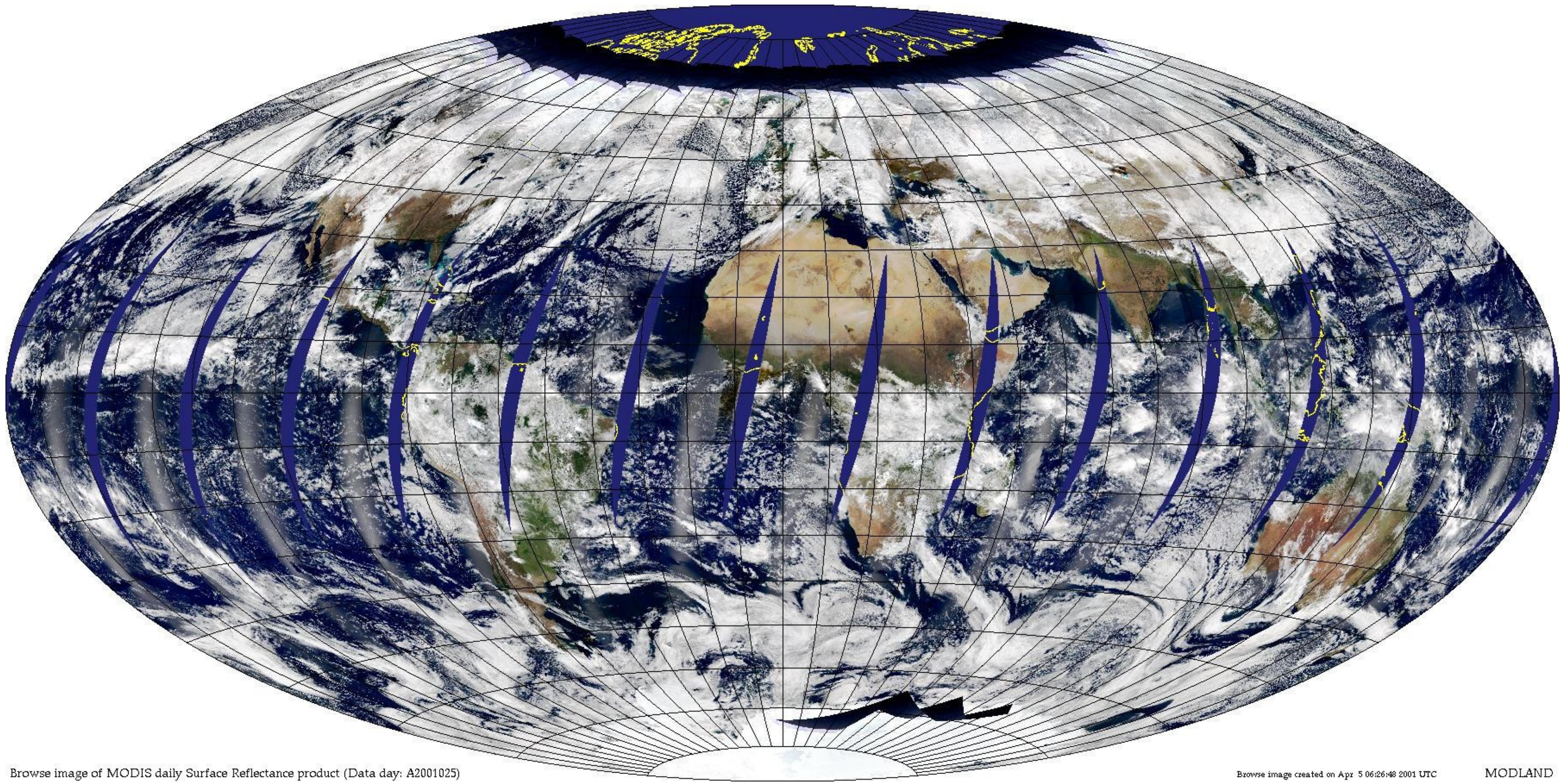


Typhoon Rammasun
2008-05-11

Burma (Myanmar)
2008-04-15

Burma (Myanmar)
2008-05-05
after Cyclone Nargis

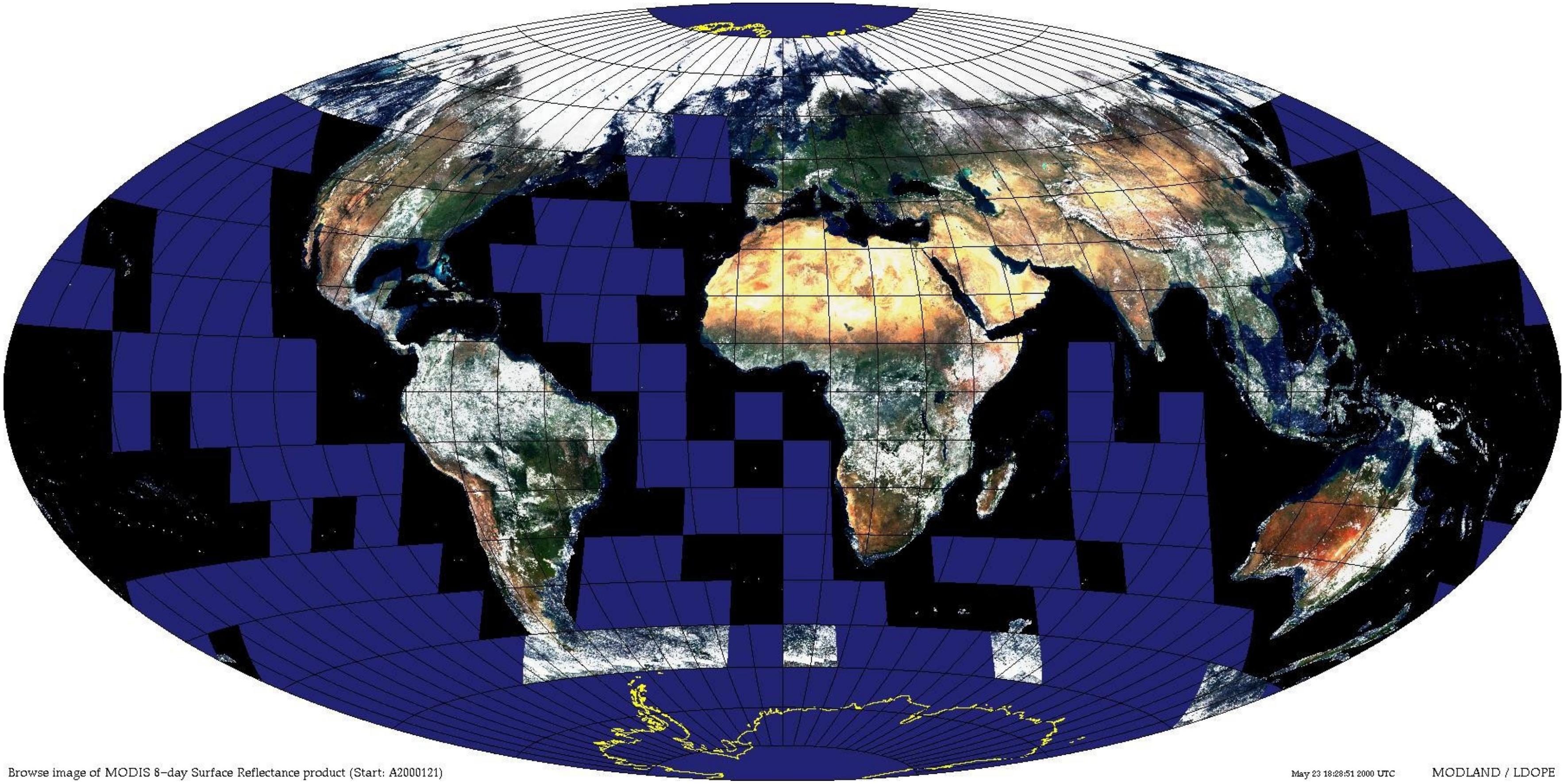
*Images courtesy Jeff Schmaltz, MODIS Land Rapid Response Team


Browse image of MODIS daily Surface Reflectance product (Data day: A2001025)
Browse image created on Apr 5 06:26:48 2001 UTC
MODLAND

MODIS Data Flow – Level 2

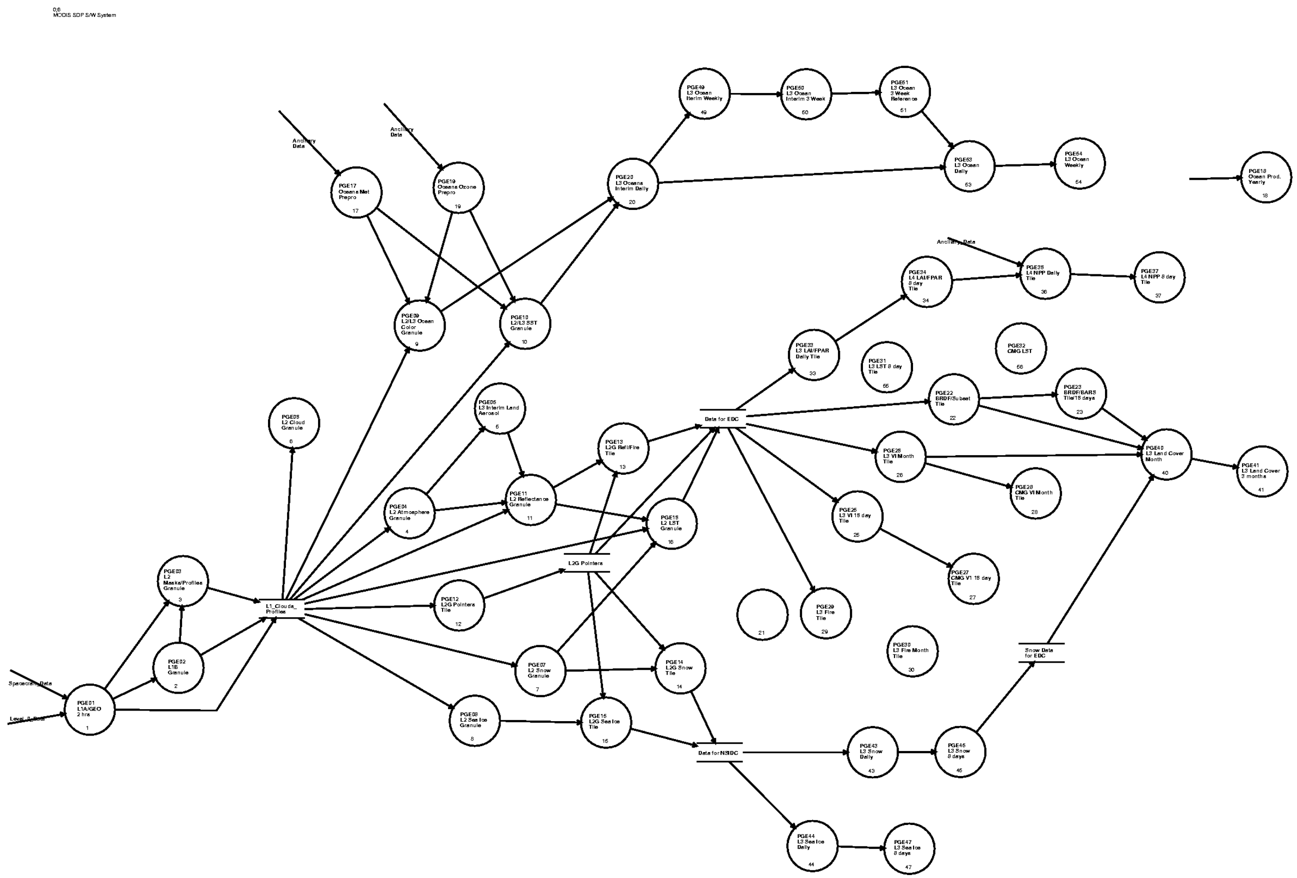
- 7 Level 2 Algorithms
- Input 5 minute granules, output 5 minute granules
- 288 x 5 minute granules per day
- 144 day-mode, 144 night-mode executions
- Input ~730MB per 5 minutes
- Output ~580MB day-mode and 175MB night mode
- Very suited to distributed processing

MODIS Data Flow – Level 3, Gridding



MODIS Data Flow

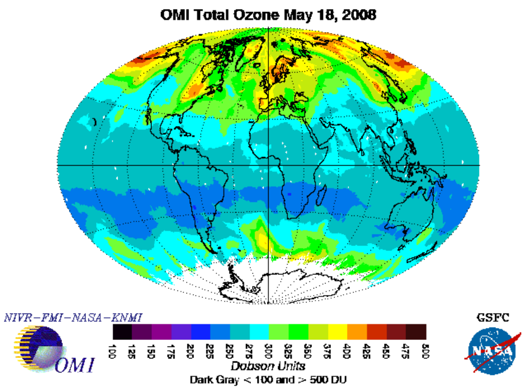
MODIS SDP S/W System

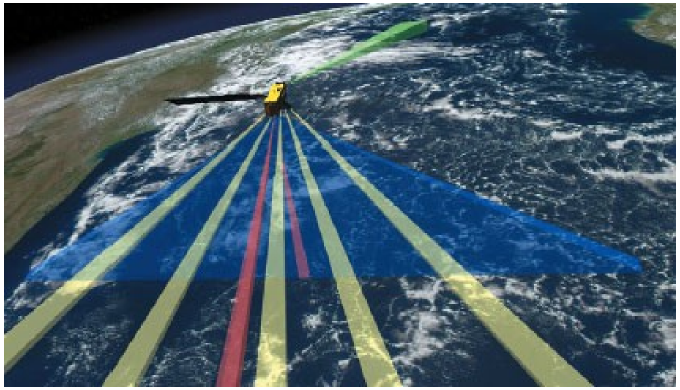


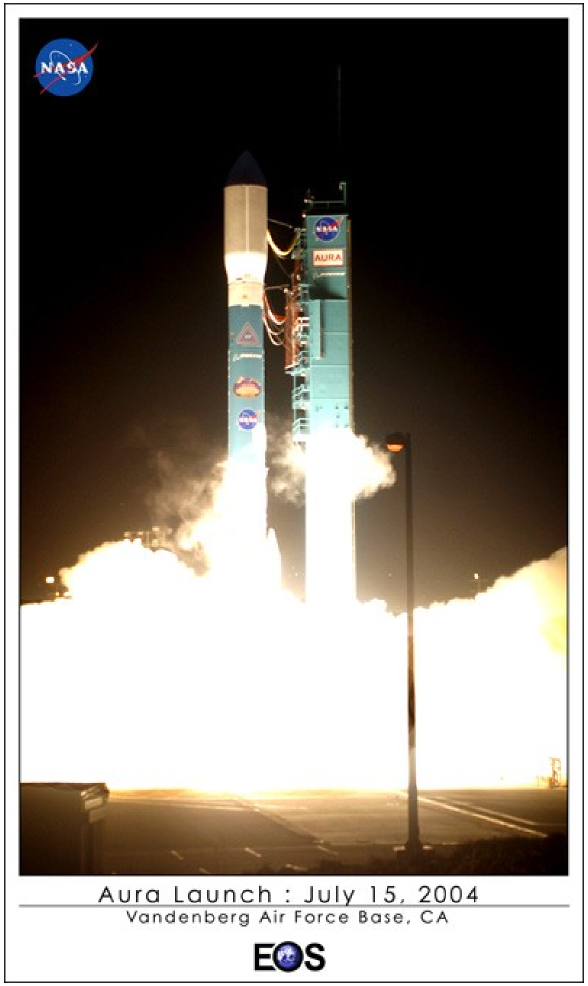
1

Ozone Processing

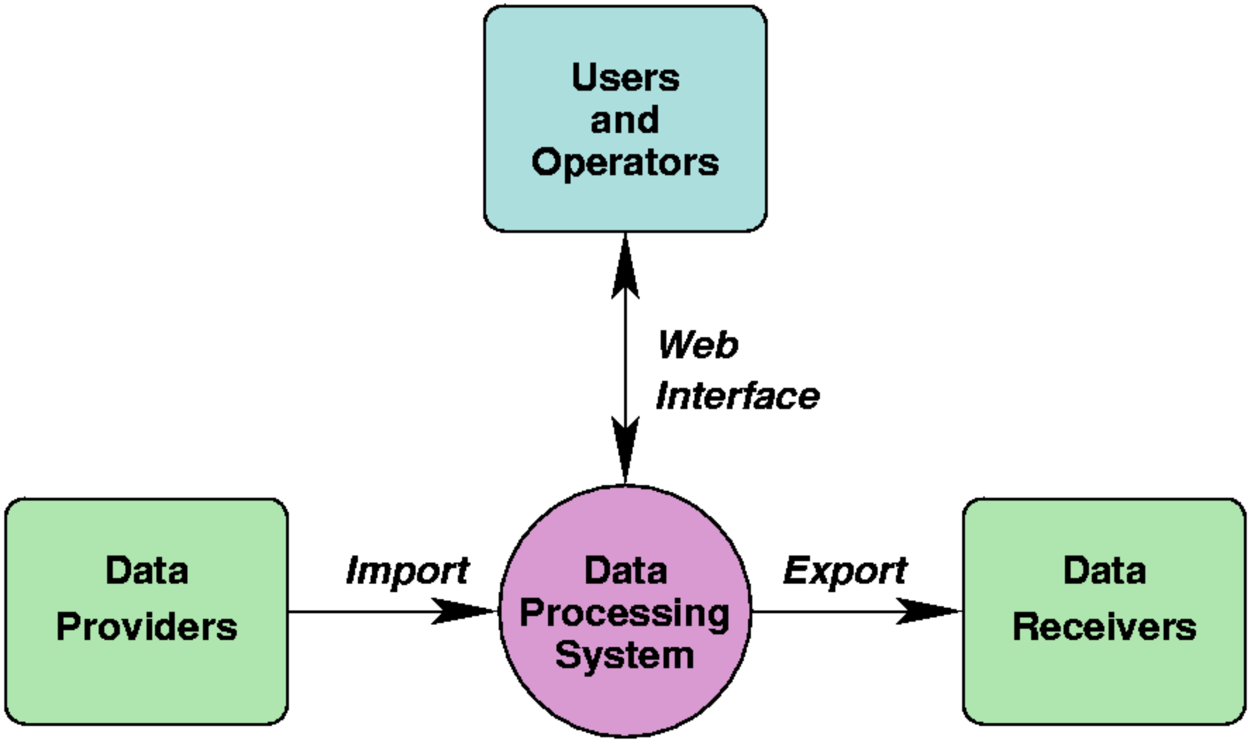
- OMIDAPS Processing System adapted from MODAPS
- TOMS
  - Total Ozone Mapping Spectrometer
  - On Nimbus-7 (1978), Meteor-3 (1991) and Earth Probe (1996)
- OMI
  - Ozone Monitoring Instrument
  - On Aura (2004) spacecraft
  - Netherlands Agency for Aerospace Programs (NIVR) in collaboration with the Finnish Meteorological Institute (FMI) and the Royal Netherlands Meteorological Institute (KNMI) sponsored OMI construction.



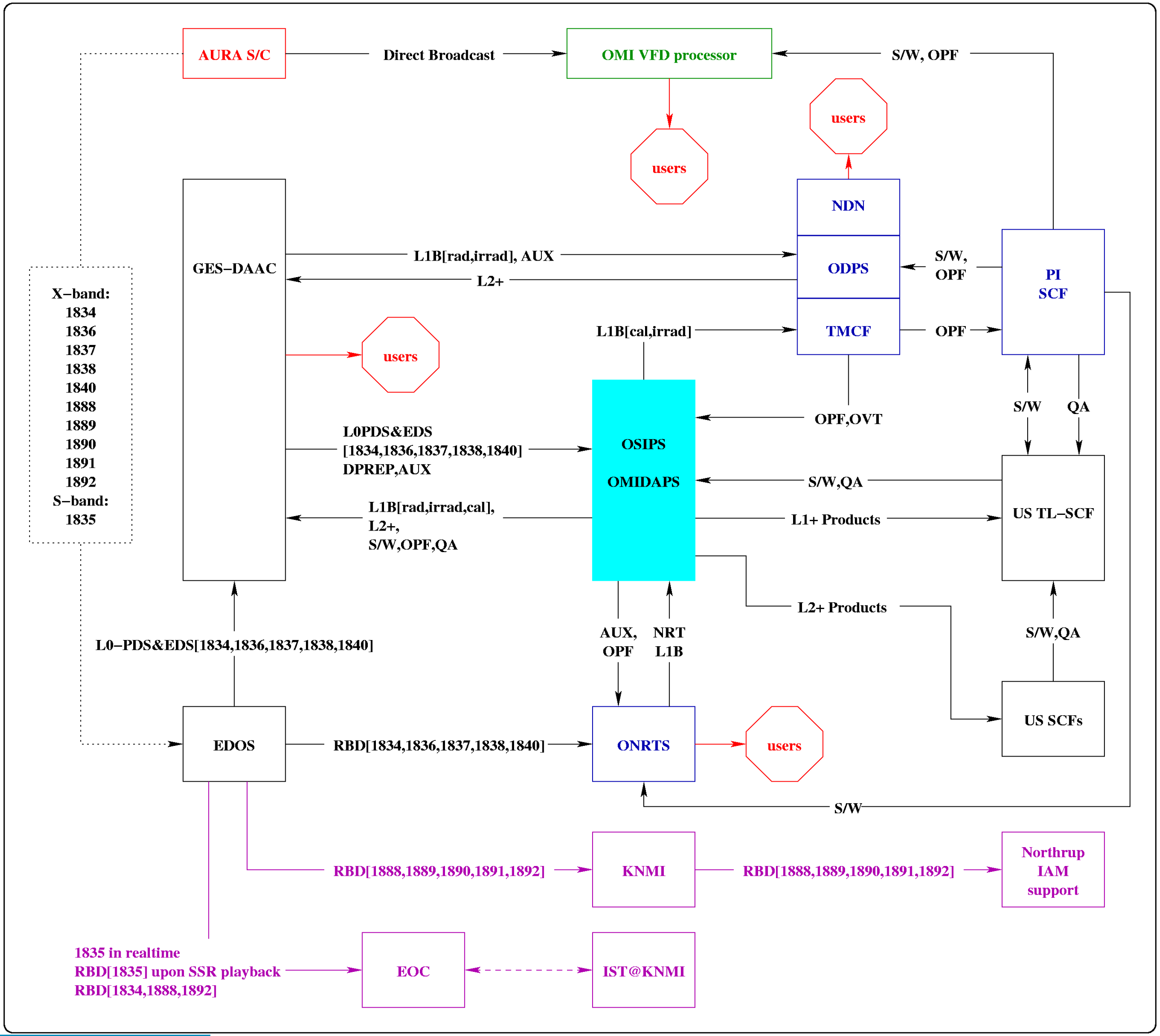




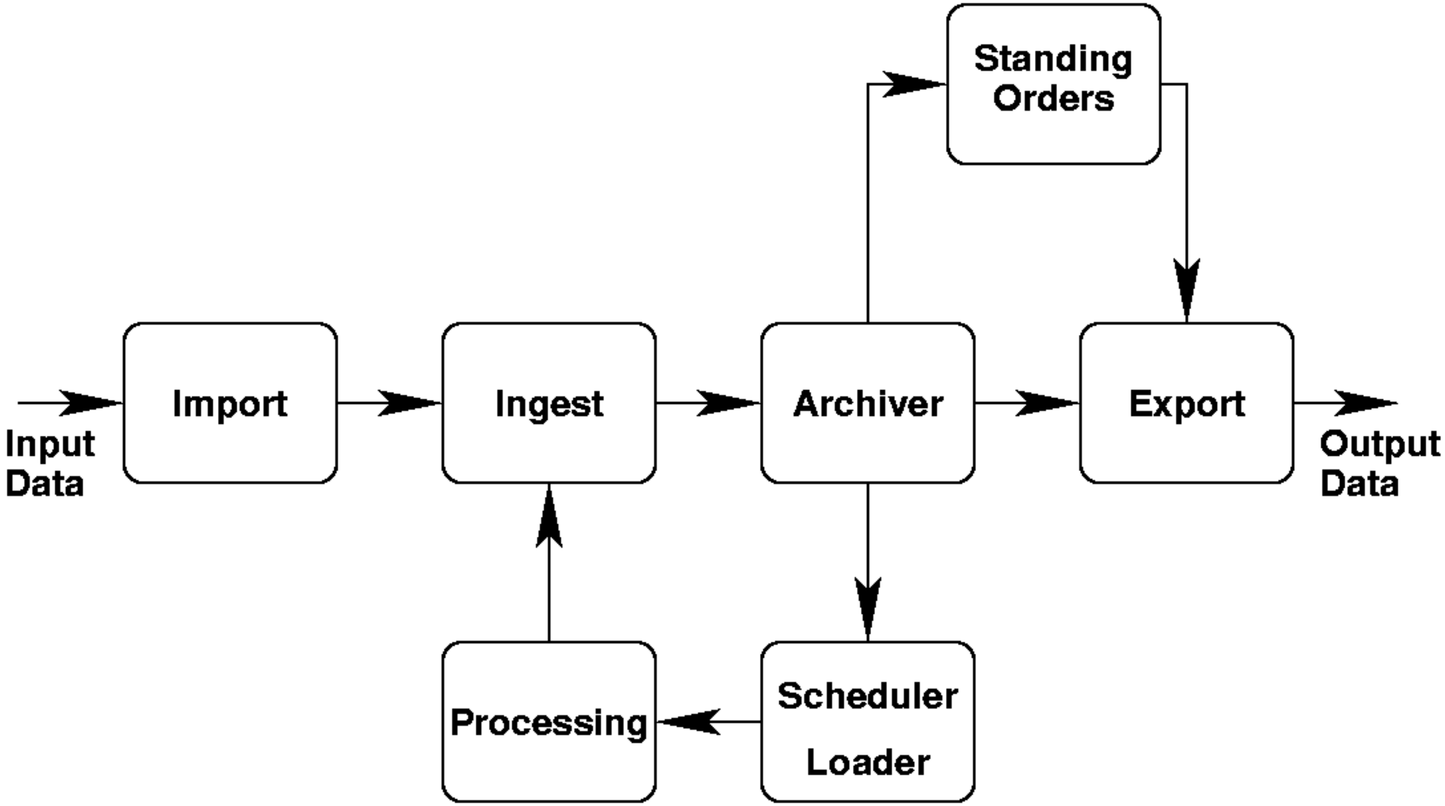
- System Context (simplified)



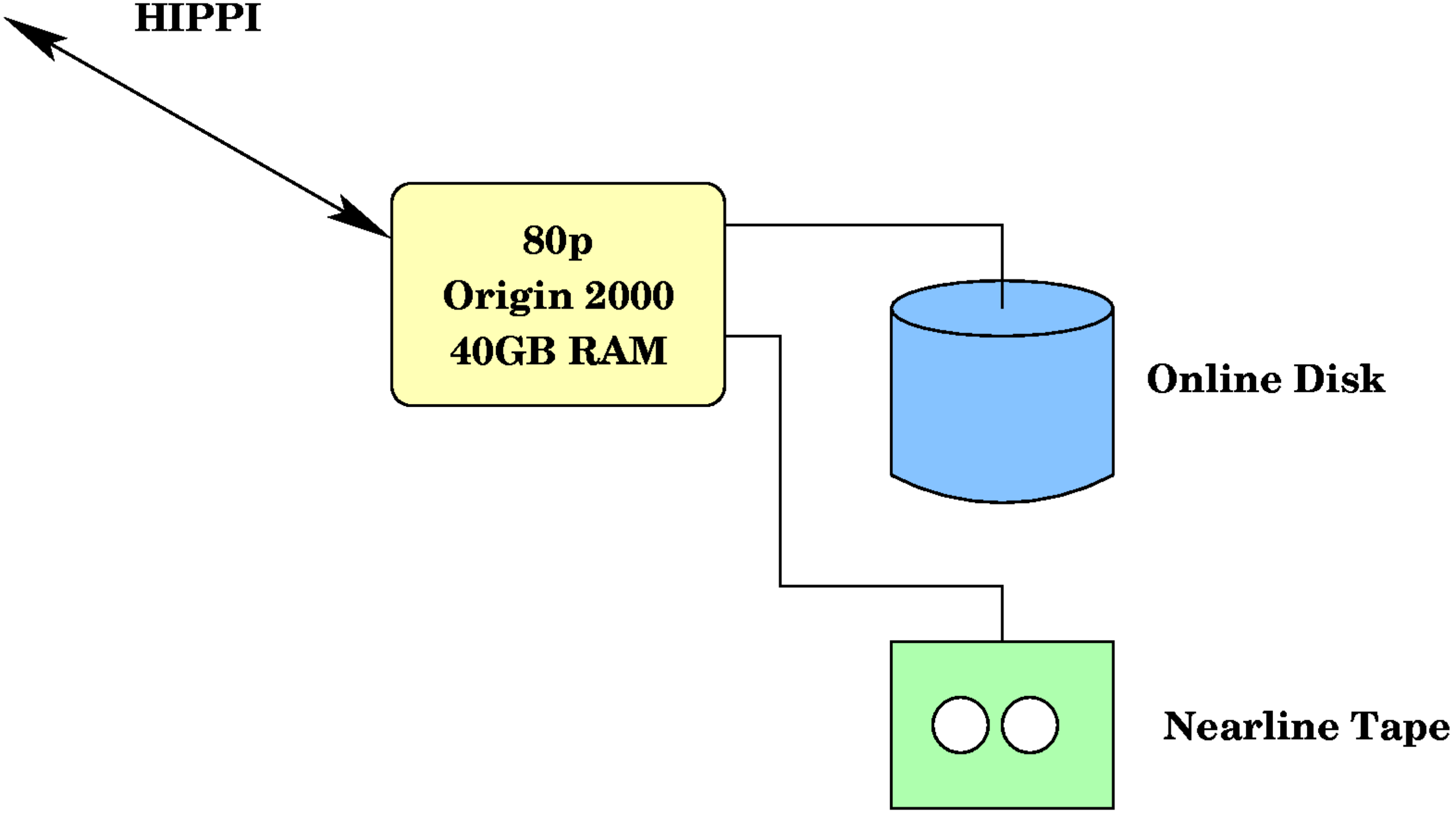
System Context (actual)



System Functions



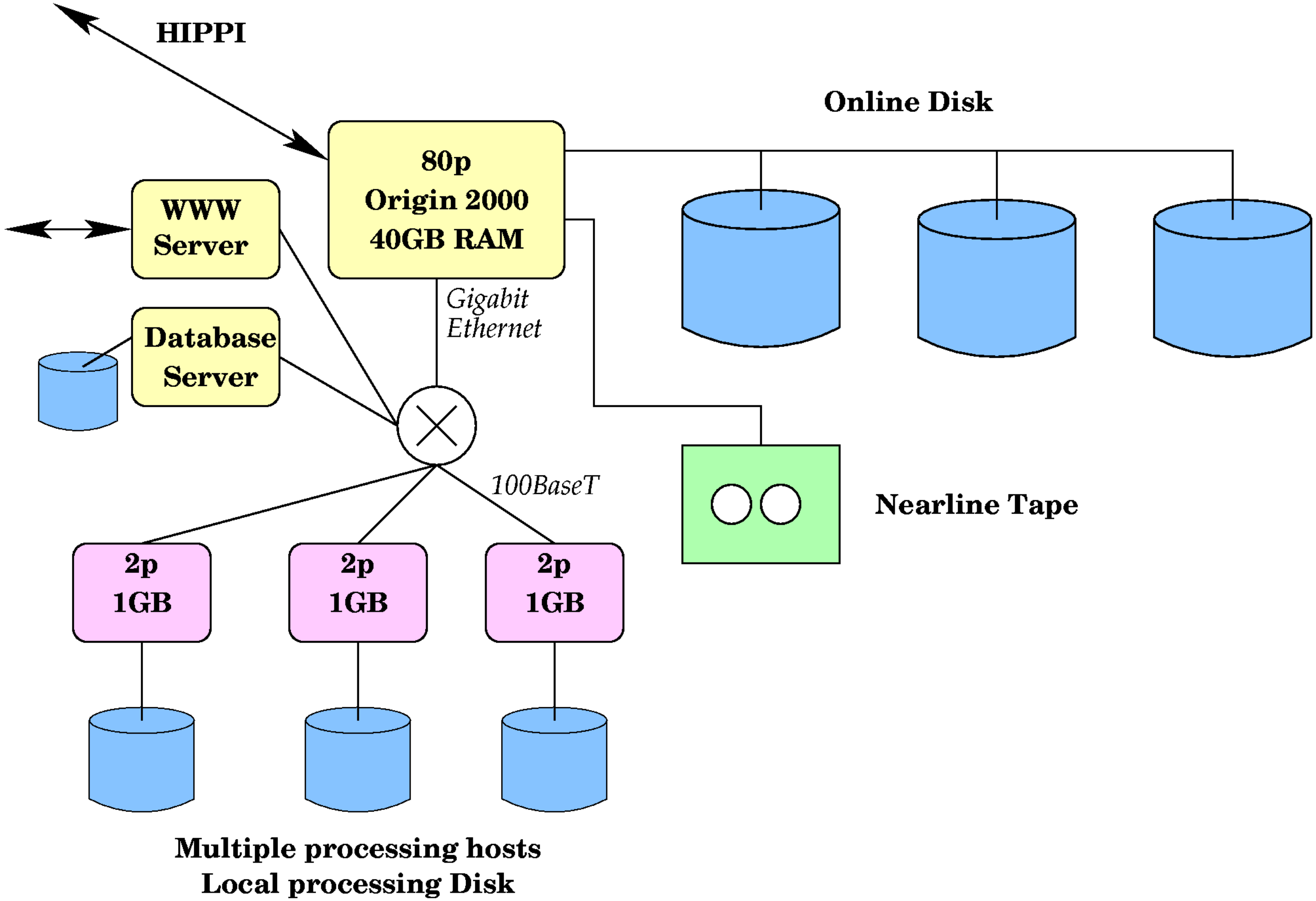
MODAPS at-launch (circa 1999)



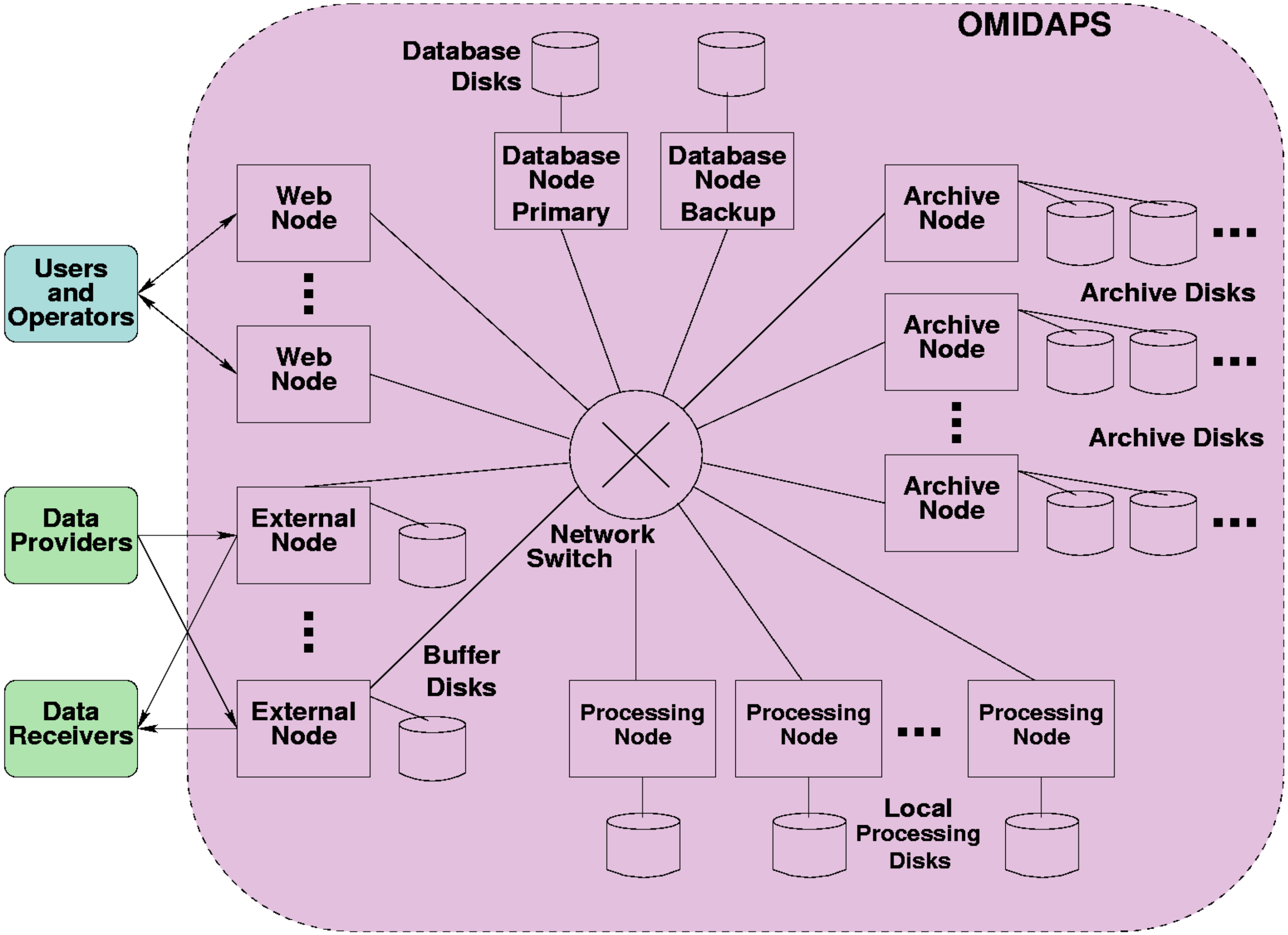
MODAPS Physical Architecture

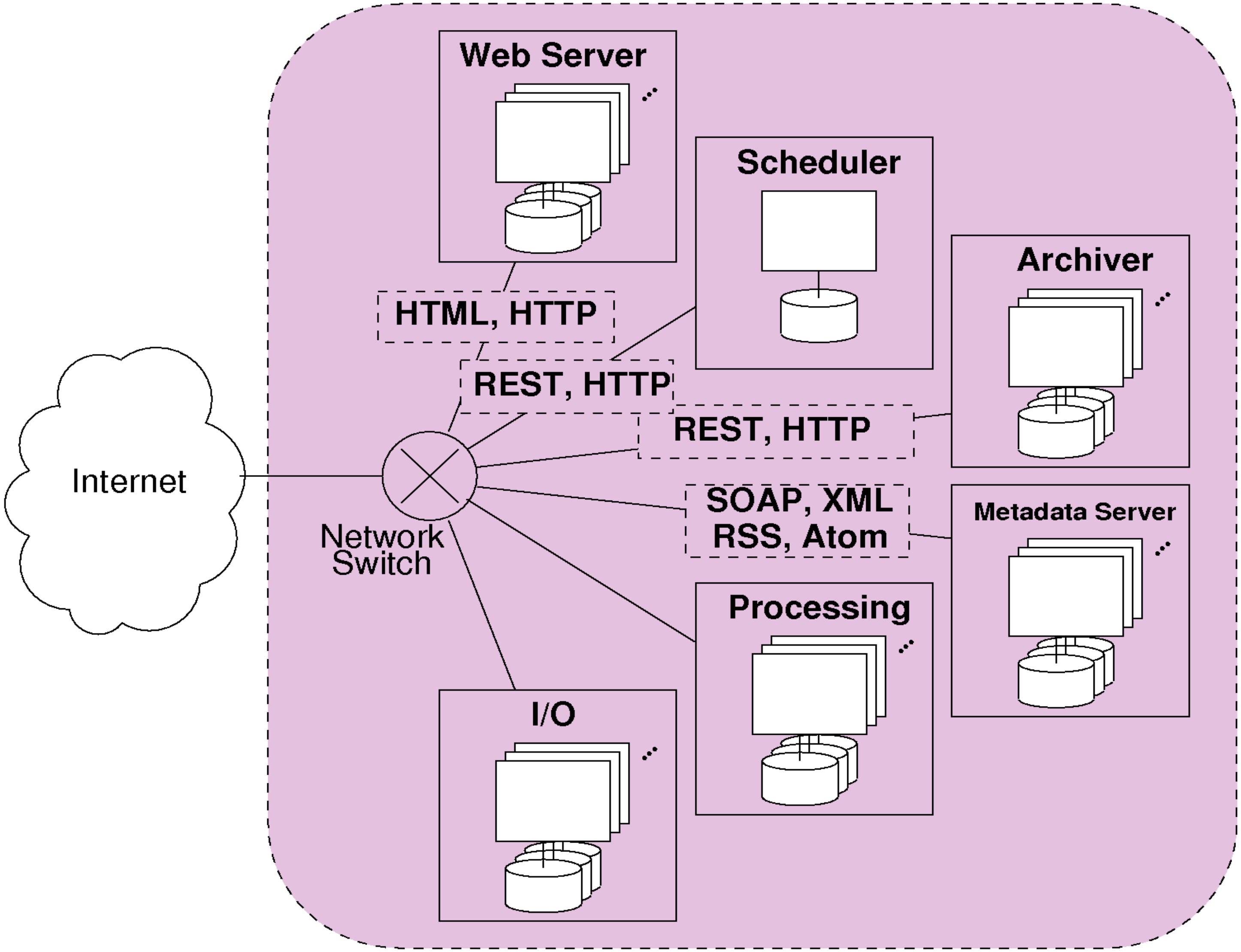
- At Terra launch (circa 1998-9)
  - 80p SGI Origin 2000, 40 GB RAM
  - 1TB disk -> 3TB disk
  - Nearline storage on tape
  - Sybase Database on the SGI
- Changes for Aqua launch (circa 2001-2)
  - Moved Sybase to dedicated Linux server
  - Added 2p Linux hosts, offloaded level 2 and level 3 processing
  - Added 35TB disk forward processing, 28TB reprocessing
- Later added multiple SGI O2000, O3000 and hundreds of 2p Linux hosts for Aqua processing, testing, reprocessing
- Eventually hundreds of processors, > 1PB of disk, no tape at all

MODAPS (circa 2001)

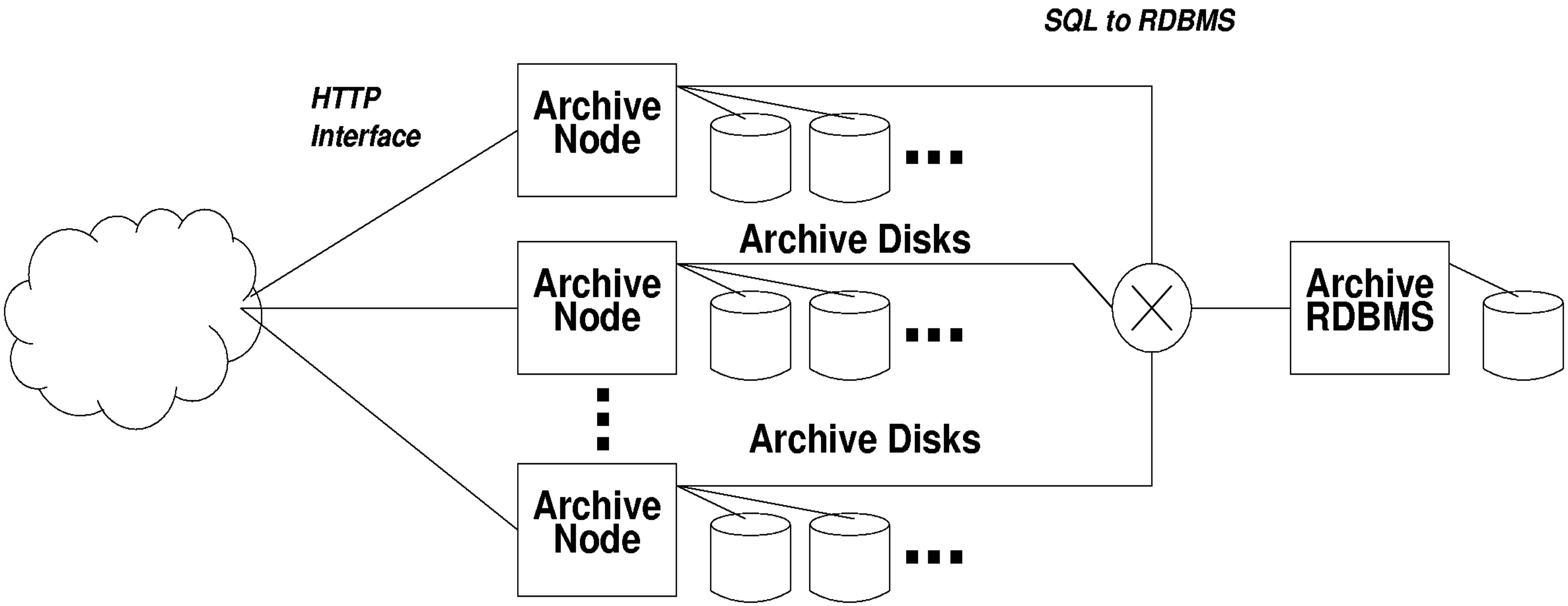


OMIDAPS Physical Architecture




Web Server
Scheduler
Archiver
HTML, HTTP
REST, HTTP
REST, HTTP
Internet
SOAP, XML
RSS, Atom
Metadata Server
Network
Switch
Processing
I/O

Modular Archive Server



Use Case Example : Finding a file

- Initial Monolithic architecture
  - Every process has direct access to the database
  - Every file is local
  - DB Query (“select <file location> where <metadata>=<file I want>”)
  - open(filename)
- Intermediate Hybrid architecture
  - Every process has direct access to the database
  - Files are now on a remote host
  - DB Query (“select <file location> where ...”)
  - rcp archivehost:/path/filename .
  - open(filename)
- New architecture
  - Files distributed across many remote hosts
  - SOAP request to Metadata Server: Find file with <metadata>
    - Metadata Server does DB Query to Metadata database
  - HTTP GET http://anyarchhost/filename
    - archhost does DB Query to Archive database
    - if filename is local, return it, else redirect to the host that has it
  - open(filename)

- Archive two parts of each data file:
  - Data – The actual data itself.
    - The files get copied onto big disks.
    - Data files are always retrieved explicitly by unique name(*).
    - We refer to the smallest chunk of individual data as a “granule” of data. It could be a month of data, a day of data, an orbit of data, 5 minutes, or even 30 seconds of data.
  - Metadata – Information that describes or relates to the data.
    - Stored in a relational database (PostgreSQL)
    - Used for search and browse to find the data itself.
    - Determining metadata can be complex
      - Different for every type of data.
      - Use “Plugins” that know how to determine metadata for various types of files.
- Give each part its own permanent distinct URL

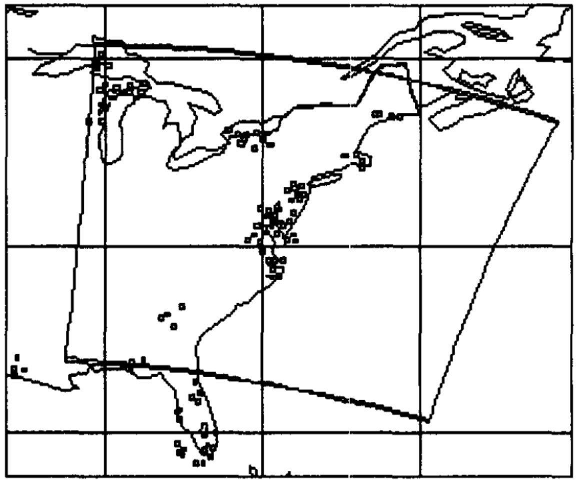
- “Collection level Metadata”
  - The same for every granule within a collection.
    - Spacecraft, Instrument, Contact Info, etc.
- “Granule level Metadata”
  - Different for each granule.
    - Orbit Number, Data capture time, etc.

Metadata Types: Primary

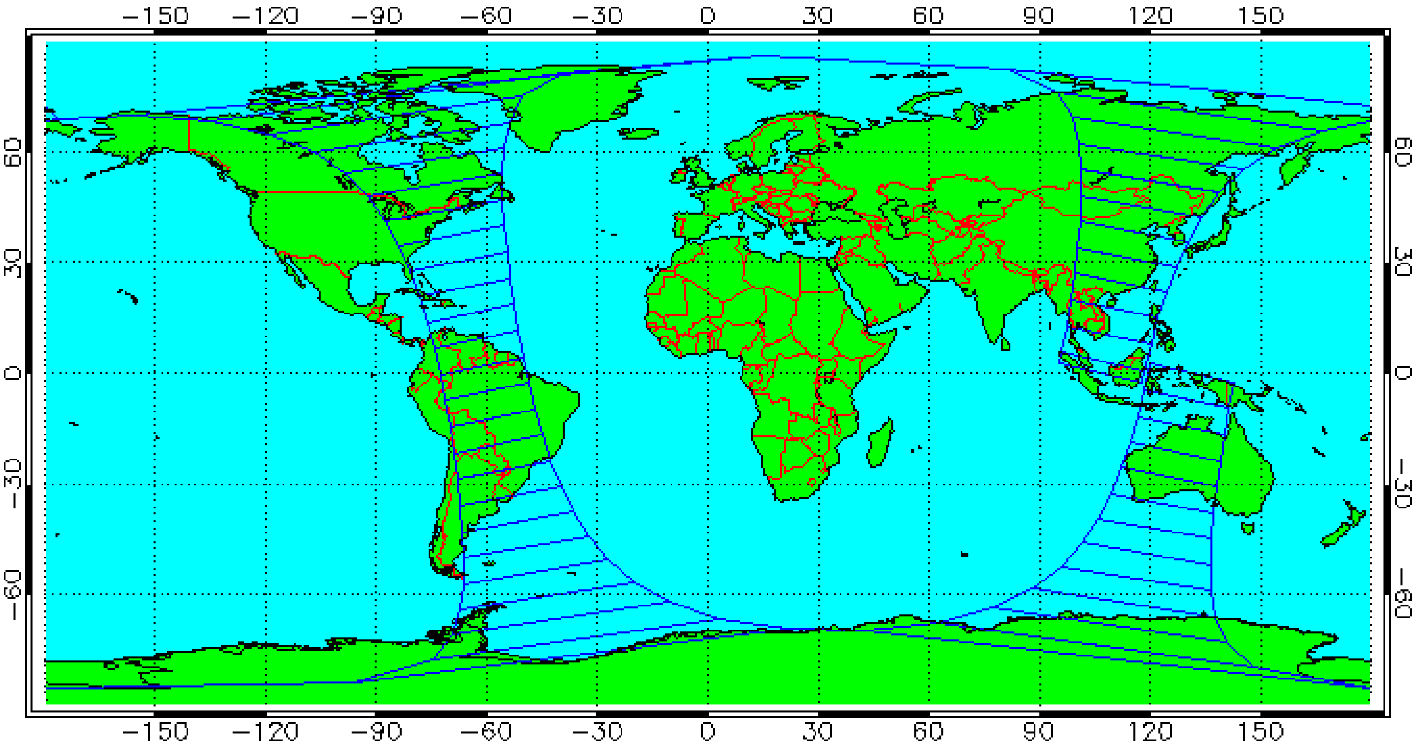
- Primary
  - The set of metadata that uniquely identifies the data of interest.
  - Comprises a set of metadata sufficient to distinguish a file:
    - Orbital { OrbitNumber }
    - TimeRange { StartTime, EndTime }
    - DailyGridded { Date, GridCoordinates }
  - Construct a unique “Key” from the primary metadata
  - A common DB table stores the FileType + Key for every type of file
  - Separate tables for each category of data, index key fields
  - Examples:
    - OMTO3_18418
    - MODL1B_20081331405 (2008, julian day 133, 14:05)
    - MODVI_2008133_10,10 (2008, julian day 133, grid (10,10))

- Secondary
  - Other interesting information about the file.
  - Some useful for searching by criteria or refining search from primary metadata.
    - Geographic information – spatial data searching
    - Quality information – cloud obscured, spacecraft maneuver flag, etc.
  - Some fields are just extra information the user wants to know about the file.
    - File Size, Checksum, List of input products
  - Hundreds of parameters – some very specific for certain types of files.
    - %Cloud Covered, Instrument Mode
  - Annotations can be added after production
  - Stored in very general "Parameter=Value" tables

- MODIS geolocation calculates {latitude,longitude,altitude} of every ground observation, summarized in granule level metadata
- Level 2 (5 minute granules) granule level metadata include
  - G-Ring: List of 4 corners (Lat,Long)
  - Bounding Box: Highest and Lowest Lat and Long
- Looking into PostGIS Generalized Search Trees (GiST), but not really using it yet...



- Aura uses active station keeping to maintain a standard orbital repeat cycle of 233 defined paths, ~2 minute blocks
- Path #156:

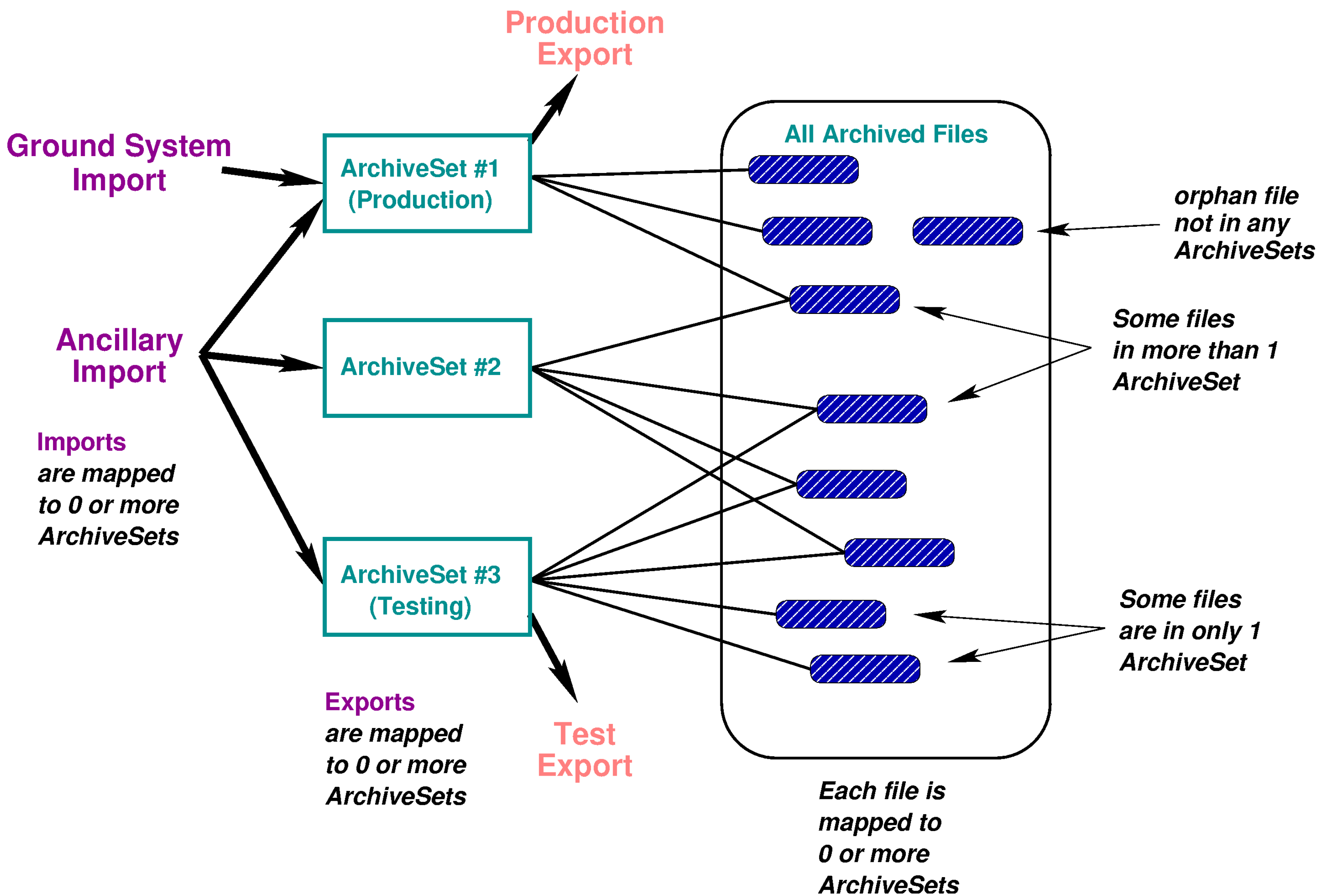


* Image courtesy KNMI

- Forward processing is easy.
  - Have a whole day to process each data day (1X)
- Science keeps marching forward
  - MODIS had an average of one new science algorithm version update delivered per day for its first year!
- Do you start processing with the new software immediately each time you find a bug?
  - Sometimes it is better to keep a dataset consistent with known problems than inconsistent.
- Periodically need to correct old data to make a new “baseline”
- At 1X reprocessing, 7 years of MODIS data would take 7 years – way too long. Even at 10X, it takes over 8 months..
- Must keep track of multiple versions of the “same” file (and process)

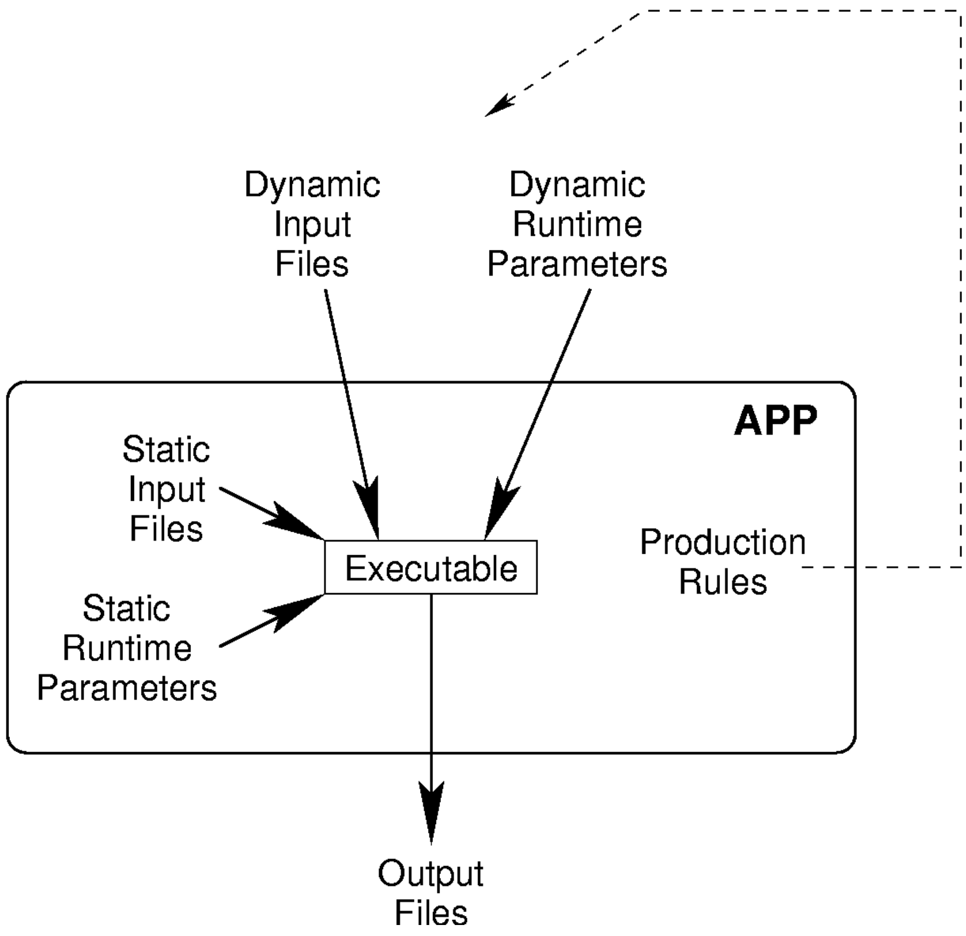
- Only one “instance” of a file can be in an ArchiveSet, with a unique set of metadata
  - i. e. for orbital files, only one file for each set of

    { FileType, OrbitNumber }
- Files can be in more than one ArchiveSet.
- When a new file with the same metadata is ingested for a given ArchiveSet, it will replace the old file in that ArchiveSet, but both files are still in the Archive. (The old file could be in another ArchiveSet.)
- DB table holds mapping of files to archivesets.
  - Maintained with PostgreSQL trigger function:
    - If file with same metadata exists in ArchiveSet, delete it from the ArchiveSet
- Each ArchiveSet acts like a “logical” processing system, providing the illusion of multiple “physical” processing systems.


Production
Export
Ground System
Import
ArchiveSet #1
(Production)
All Archived Files
orphan file
not in any
ArchiveSets
Ancillary
Import
ArchiveSet #2
Some files
in more than 1
ArchiveSet
Imports
are mapped
to 0 or more
ArchiveSets
ArchiveSet #3
(Testing)
Some files
are in only 1
ArchiveSet
Exports
are mapped
to 0 or more
ArchiveSets
Test
Export
Each file is
mapped to
0 or more
ArchiveSets

APP – Algorithm Plugin Package

- A well defined interface for inserting science algorithms into the processing framework.
- Allows scientists and developers to concentrate on science and getting algorithm working.
- APPs can be unit tested outside of the framework.
- Each APP encapsulates an algorithm:
  - Executable program(s)
  - Wrapper scripts that interface with the framework
  - Scheduling Rules – How often should the APP be run?
  - Production Rules – What input files and parameters should be used for a given run?


Dynamic
Input
Files
Dynamic
Runtime
Parameters
APP
Static
Input
Files
Executable
Production
Rules
Static
Runtime
Parameters
Output
Files

Production Rules

- Scheduling Rules – Divide the work into manageable chunks
  - Temporal (5 minutes, 1 orbit, 1 day, 8 days, 16 days, 1 month, etc.)
  - Geographic (tile schemes)
  - Profiles (different ways of running)
  - Etc. (Plugins allow extending to new methods)
  - Iterators (and nested iterators) can schedule many instances of processes at once
- Input Files and Parameters
  - What inputs do I need?
  - Optional inputs?
  - Alternate inputs?
  - Timer delays
  - Etc. (Plugins allow extending to new queries)

Production Rules

- Production Rules are defined for each APP
- Production Rules can “succeed” or “fail”
  - APP won’t run until all its rules succeed
- Production Rules can search the metadata database for needed dynamic input files
  - Mostly primary metadata tables, sometimes qualified by secondary metadata
- Production Rules search a specific ArchiveSet
  - join to archiveset table
- Since the rules search the current database they change with time.
- Production Rules can set dynamic runtime parameters
  - Runtime parameters can also be set by operations staff
- Static Input Files and Runtime Parameters are part of the APP
  - e. g. a lookup table or elevation map that is the same for every run

Runtime Parameters:

```
ECSCollection: '3'
EndTime: '2008-01-01T02:06:53.000000Z'
HDFCompress: '0'
InstrumentConfigAS: '10003'
OrbitNumber: '18418'
APP: 'OMTO3'
APPVersion: '1.1.0'
ProcessingCenter: 'OMI SIPS'
ReprocessingActual: 'processed 1 time'
SMFVerbosityThreshold: '2'
Source: 'OMI'
StartTime: '2008-01-01T00:28:00.000000Z'
TDOPFIntendedPurpose: 'Forward Processing'
TDOPFVersion: '1301'
ProcessingHost: 'Linux ominion607 2.6.22.6 i686'
```

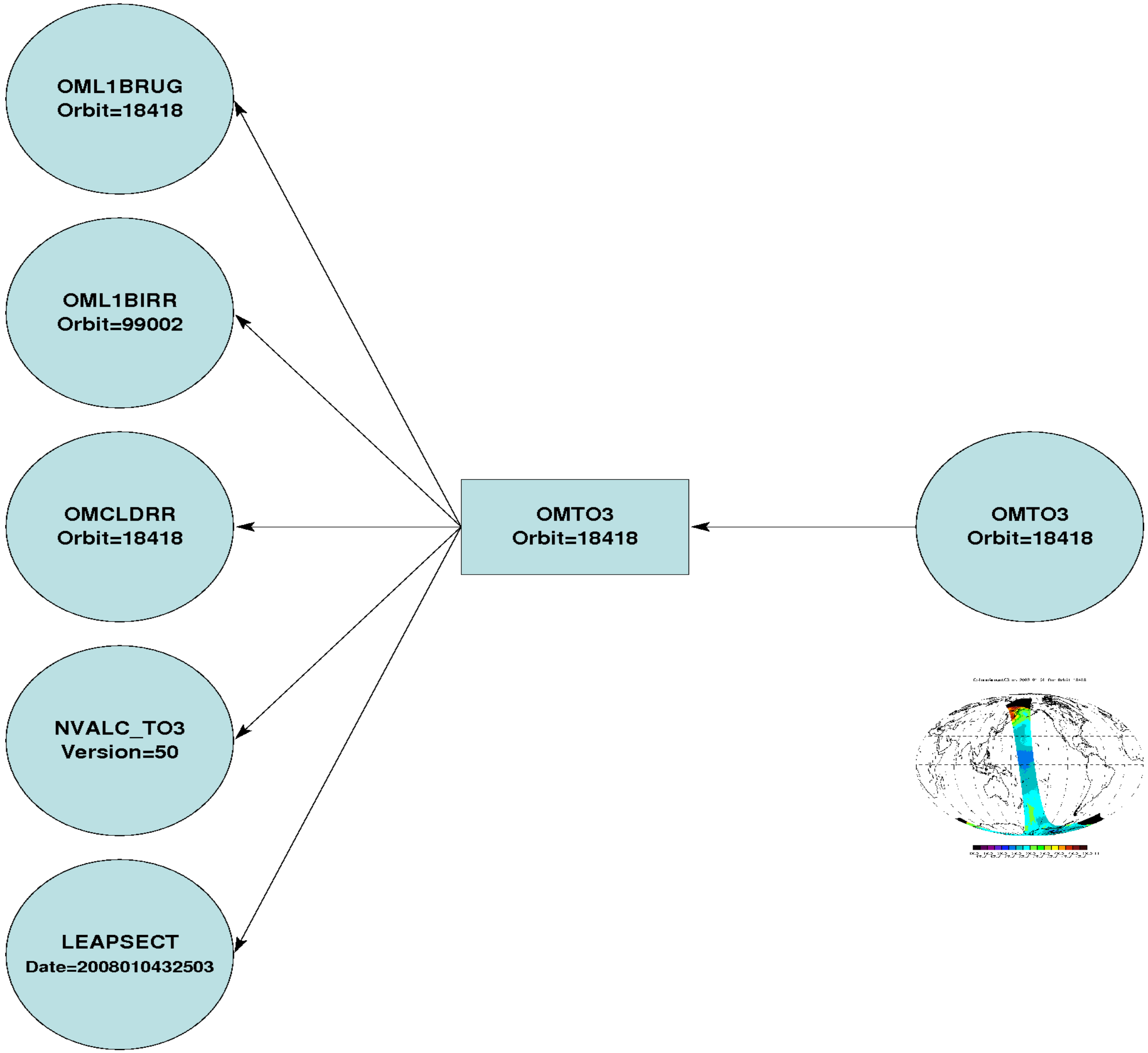
APP Example

Input Files:

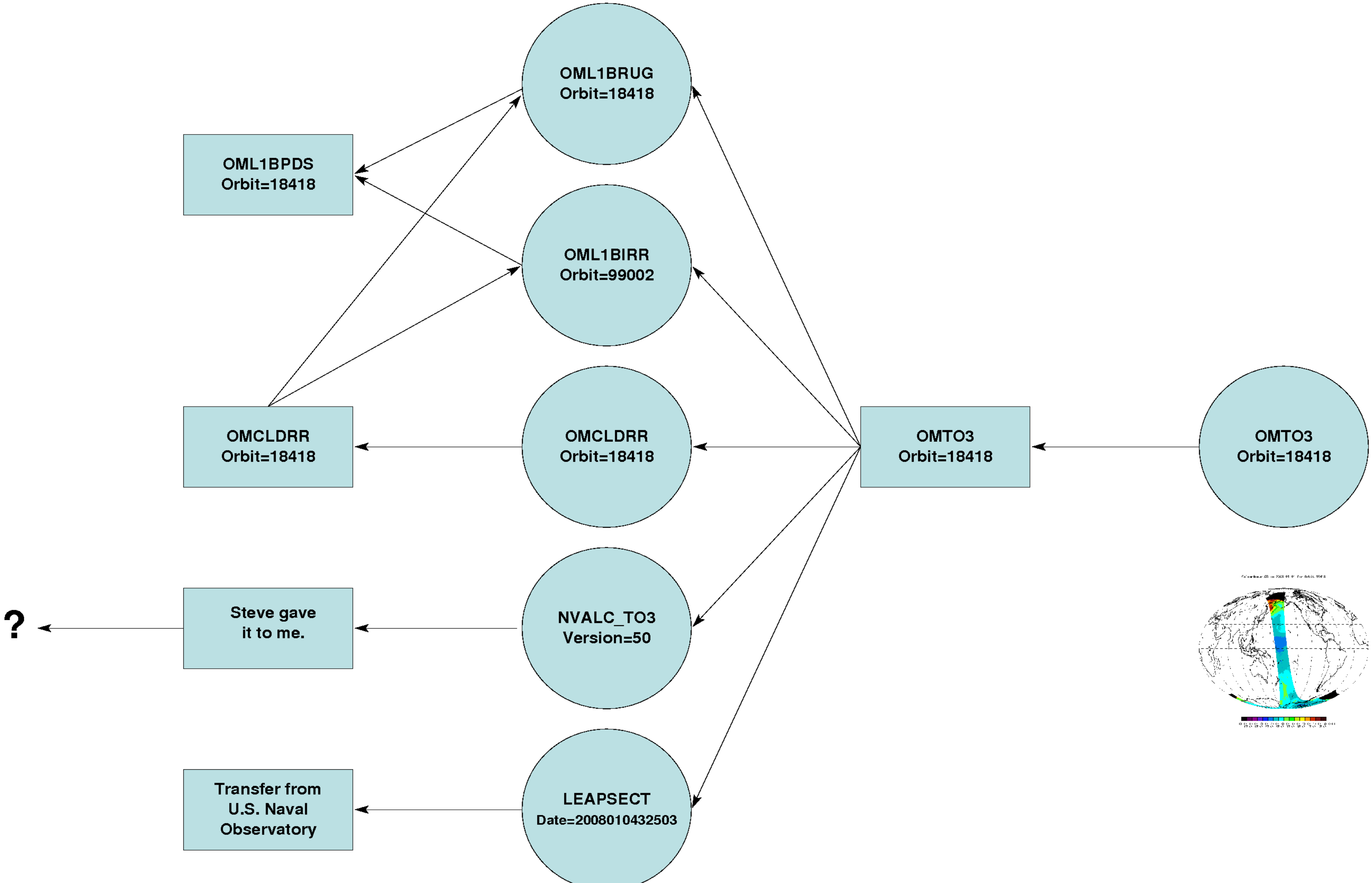
```
OML1BIRR:
  - OMI-Aura_L1-OML1BIRR_2004m1231t1248-o99002_v003-2007m0511t172858.he4
 OMCLDRR:
  - OMI-Aura_L2-OMCLDRR_2008m0101t0028-o18418_v003-2008m0108t202125.he5
 NVALC_TO3:
  - OMI-Aura_L2-NVALC_TO3_v00050.he4
 OML1BRUG:
  - OMI-Aura_L1-OML1BRUG_2008m0101t0028-o18418_v003-2008m0108t195451.he4
 LEAPSECT:
  - leapsec.dat.2008010432503
```

- Science is based on a principal of repeatability.
- Provenance: “The origin or source from which something comes”.
- Just as a laboratory experimenter must control and capture everything about the experiment environment, so should a science data processing system…
    - Algorithm Theoretical Basis Documents (ATBD)
    - Software Source Code, version
    - Software Build Environment, version
        - Static libraries, versions
        - Compiler versions
    - APP version
    - Execution Environment
        - Specific hardware
        - OS version
        - Dynamic libraries versions
    - Execution Instance
        - Runtime parameters
        - Input files and versions

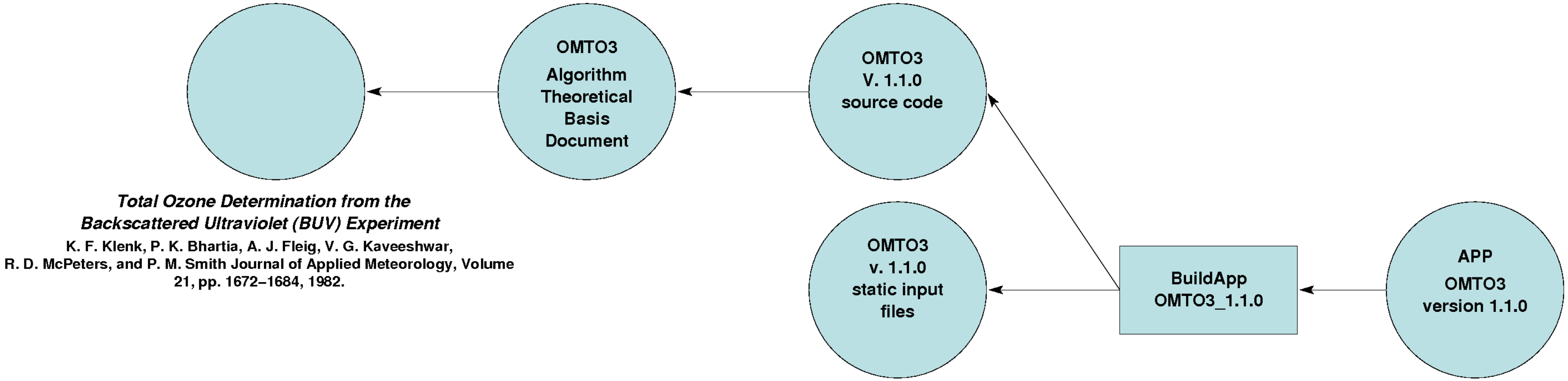
Example Provenance (1)



Example Provenance (2)



Example Provenance (3)



- Capturing complete and accurate provenance during data ingest and primary data processing
- Archiving provenance such that it can be easily retrieved and searched, even if the data are deleted
- Representing provenance to human users and providing tools for navigating graph to search and explore data provenance
- Representing provenance semantically to other systems at cooperating institutions with standard ontologies
- Allow agents to traverse inter-system provenance graphs and answer provenance questions
- Allow ***independent*** systems to mechanically reproduce data processing using the provenance information

- For valid science and complete “scientific reproducibility”, you must capture sufficient information to trace back the provenance of each product.
- Given such provenance and the ability to use it, do you still need the files in the archive at all?
  - There is a tradeoff between disk costs and processing costs
- “Extreme Compression”
  - Instead of storing the data product, just store the provenance.
  - When someone needs the file, just re-create it.
  - Given periodic reprocessing, many files are never needed again anyway..
- Allows much larger “virtual archives”
  - We make choices about which products to create, archive and distribute – intermediate products not always kept anyway

•Questions/Comments?

OMIDAPS Schema