Tioga: Data Management Support for Scientific Visualization Applications

(from “Manteca” to “Big Oak Flat”)

Jolly Chen, Michael Stonebraker, Caroline Paxson

Nobuko Nathan, Jiang Wu, Alan Su

Current visualization systems

Examples: AVS, Data Explorer, Khoros

- "boxes and arrows" visual programming
- library of visualization modules
- creation, manipulation, and execution of visual programs (recipes)

So What's the Problem?

Weaknesses of existing visualization systems

- file orientation: data passed via files or shared memory
- limited data types
- all or nothing buffering of intermediate results
- limited browsing of results; VCR player instead of joystick, “flight simulator” model
- no zoom/abstracts capability
- no query system for recipes
- no concurrency control or crash recovery

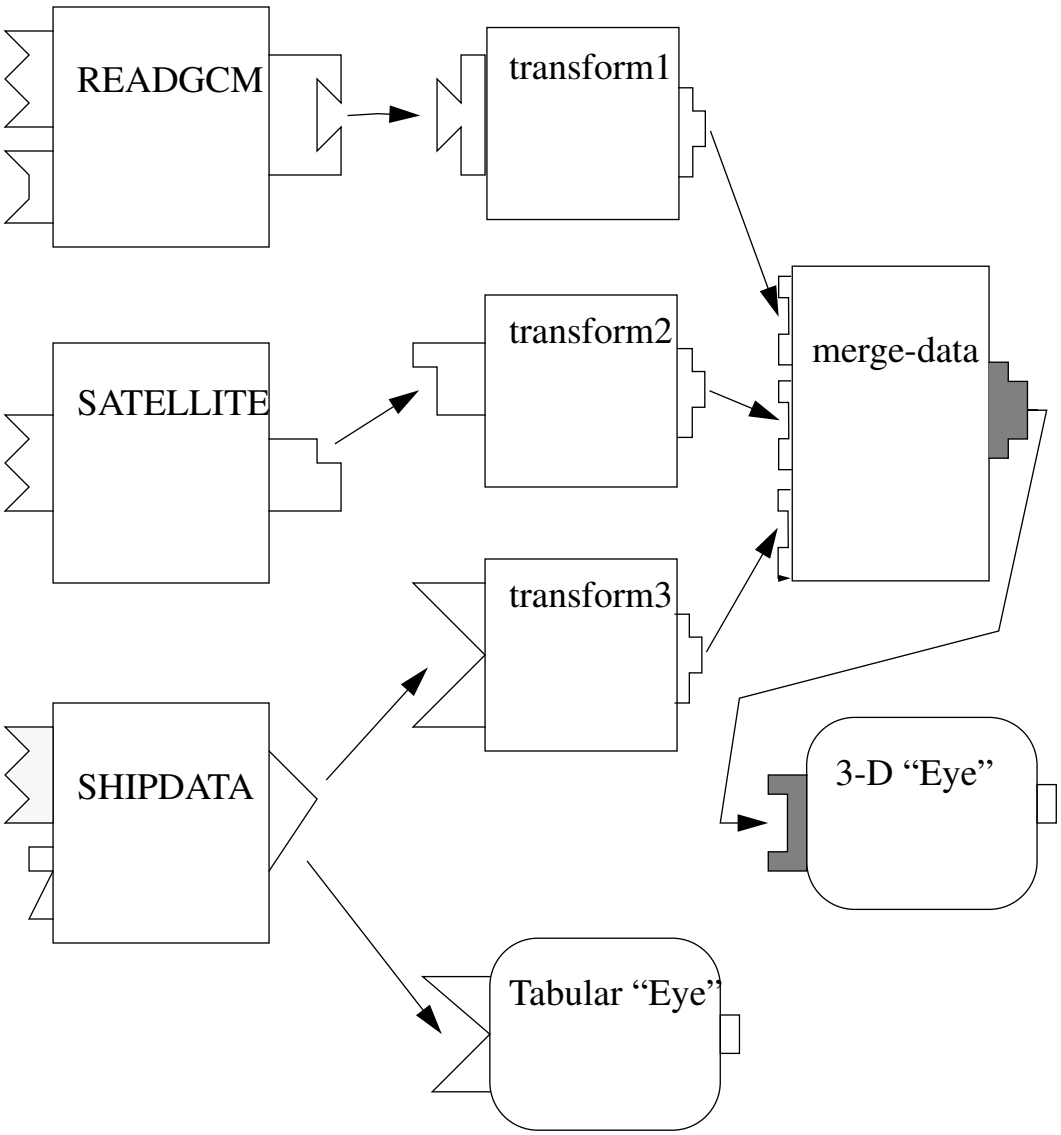
A database-centered approach to visualization

= Tioga

A Tioga recipe contains:

- ingredients
- browsers
- flow edges

An example:


READGCM
transform1
transform2
SATELLITE
merge-data
transform3
SHIPDATA
3-D “Eye”
Tabular “Eye”

Tioga's solutions

- ingredients are registered functions in the DBMS
- user can extend the DBMS type system
- recipes can be optimized through buffering and coalescing of ingredients
- user can *zoom* using the use of *abstracts*
- new application-DBMS protocol which incorporates a flight-simulator model

Design issues and research problems

- transaction model: not traditional transactions, not nested transactions, either.
- interface with foreign systems
- execution model: demand-driven/data-driven, run-time process scheduling, distribution.
- guaranteed rate of delivery of data
- exploit additional DBMS features -- e.g. rules

Example application: hydroclimatic database

in collaboration with Frank Gehrke from DWR:

- snow course data (~ 40,000 tuples)
- full natural flow data (~ 56,000 tuples)
- five-ingredient recipe to compare historical adjacent basin flows
- useful functionality currently implemented by difficult to maintain “legacy” code and tedious manual process
- data retrieval and visual analysis of results

Current status and demo

Working prototypes:

- graphical recipe editor
- recipe execution system
- multi-dimensional graphical browser
- recipe storage and retrieval

- graphical interfaces implemented in tk/tcl
- ingredients implemented in C and postquel
- processes communicate via tk/tcl *send*
- maximal buffering of intermediate results

Lessons and Future Work

Lessons:

- recipe paradigm is natural and suitable for database programming
- closely-coupled interaction with earth scientist and computer scientist is more effective
- postgres is not production quality

Future:

- database-native recipe execution engine
- recipe optimizations
- more ingredients -- integration with foreign systems
- more sophisticated visual browsers
- implement additional research ideas