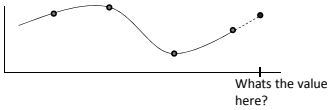
Algebraic Optimization of Computations over Scientific Databases

Risi Thonangi

CPS 296.1

Scientific Computation Example

- Extrapolation



  - Polynomial curve-fitting; gaussian mixtures; …
- Other examples
  - Interpolation, Selection, FFT, …

Running scientific computation

- Difficult because …
  - Various logical and physical operators exist
  - Coding is an involved job
  - Data format conversions need care

Database approach to scientific data

- Makes scientific computing easier
- Other goodies offered by database systems
  - Extensibility
  - Support for query optimization
  - Logical and Physical data independence

Next …

- Supporting scientific computing in Volcano database system
  1. Data types in the system
  2. Supported logical and physical operators
  3. Handling query optimization
     - Transformations and implementation rules
- An example for scientific query optimization

1. Data types

- Sets
  - Similar to relations
  - Logical properties: schema, cardinality, …
- Time series
  - Similar to a relation but contains a time attribute
  - Logical properties: start and stop times, and fixed time delta
- Spectra
  - Similar to a relation but contains a frequency attribute
  - Logical properties: frequency range and fixed frequency delta

2. Logical and physical operators

- List of relational and scientific operators

Type	Logical Operators	Physical Operators
Sets	Join Select, Project, Sample	Hash & Merge Join Filter
Time series	Digital Filter, Interpolation, Extrapolation, Regular Sampling Merge	Windowing operator Hash & Merge Join
Spectra	Spectral Filter Merge	Filter Hash & Merge Join
All Types	Math Function	Filter
Coerversion	Set→Time series, Set→Spectrum Time series→Set, Spectrum→Set Time series→Spectrum Spectrum→Time series	NOP (requires time or frequency tags) NOP FFT Inverse FFT
Enforcers		Sort

May ask for fixed output cardinality

Update every time position by considering values in a window around it

Two step procedure:
1. Do a bit-reverse sort
2. Build and execute an FFT processing tree

Support for physical operators

- Iterator-style execution
  - Volcano's existing iterators for most operators
  - New window-iterator added to support windowing operators

3. Handling query optimization

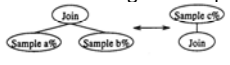
- Made easy by the extensible volcano query optimizer generator
- Optimizer generator accepts following inputs
  - Logical and physical operators
  - Transformation and implementation rules
  - Cost functions
  - Logical and physical properties
  - …

Handling query optimization: Transformations

- Logical transformations
  - Helps the optimizer find equivalent query expressions
  - Encoded as rules
  - Care required in order to handle effects of numerical accuracy and stability
- Example transformations
  - All standard relational transformations

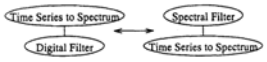
Handling query optimization: Transformations (contd.)

- Example transformations (contd.)
  - All standard relational transformations
  - Transformations involving the sampling operator



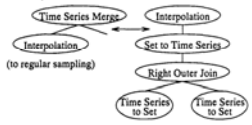
where a% × b% = c%

  - Time series and spectra operator transformations



Handling query optimization: Transformations (contd.)

- Example transformations (contd.)
  - Interpolations and merges

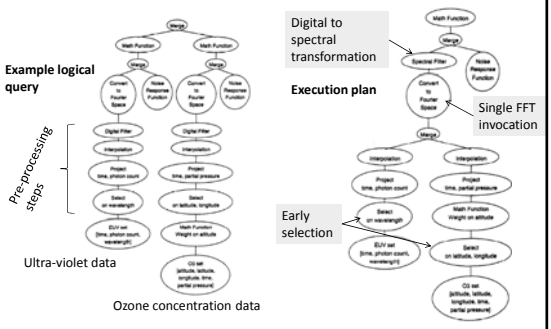


- Difficult to transform operators
  - Digital filtering operators

Handling query optimization: Implementation rules

- Implementation rules help convert a logical plan to physical plans
- Encoded as rules
- Examples
  - All relational rules
  - Polynomial curve fitting for interpolation
  - Special case execution to compute two FFTs in one invocation
- To resolve between multiple implementation rules, use
  - Cost functions
  - Heuristics

Query optimization example



Summary

- Database approach to time-series data
  - Benefits include data independence, query optimization, …
- Can accuracy be a part of query optimization?
  - How to account for loss of accuracies across multiple query operators?
- How about other types of scientific data?
  - Does relational approach work?