



CONQUEST:
Concurrent Queries over
Space and Time

Silvia Nittel

University of California, Los Angeles



Overview

- ⌘ Motivation
- ⌘ Conquest Overview
- ⌘ Operators in Conquest
- ⌘ Trigger-Based Re-optimization in Conquest
- ⌘ Conclusions



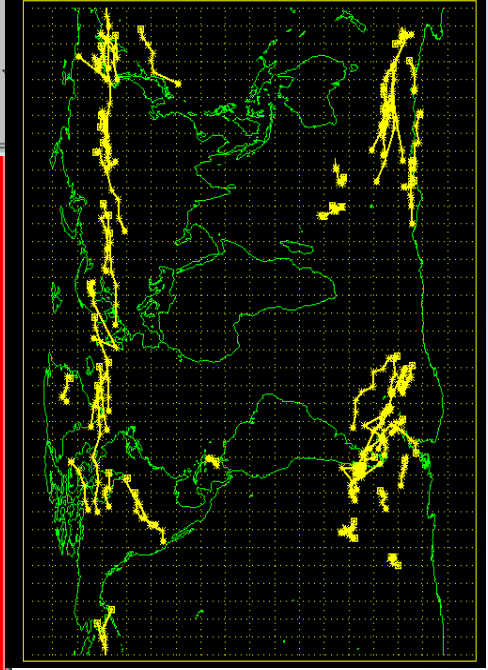
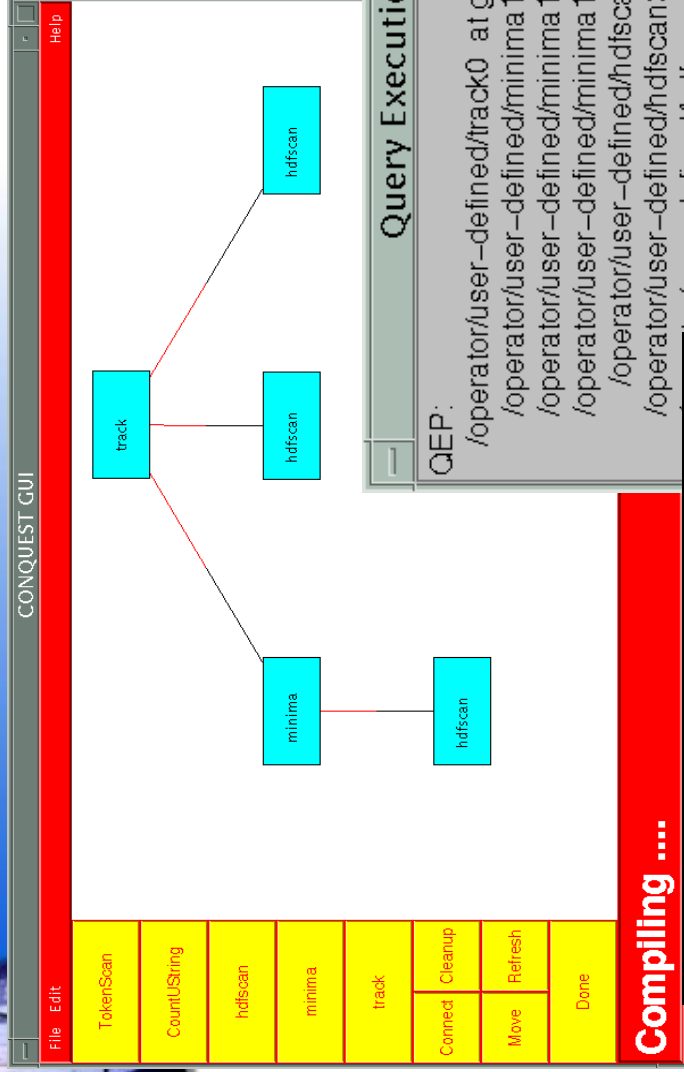
Motivation

- ⌘ Availability of large amounts of satellite raster data (commercial or government)
- ⌘ Need to efficiently mine the data, ie. identify patterns and features, in
 - ⌘ weather data
 - ⌘ land-use data
 - ⌘
- ⌘ A typical 'query' can involve large amounts of data
- ⌘ Creation of new, derived data sets



Requirements

- ⌘ **Efficiency** of query execution
- ⌘ **Extensibility** of data model and operators
- ⌘ **Re-use** of existing analysis code
- ⌘ **Ease of use**
 - ☑ Implementation of operators (wrapping)
 - ☑ Integration of operators
 - ☑ Definition of queries
- ⌘ **'Cheap' execution environment**



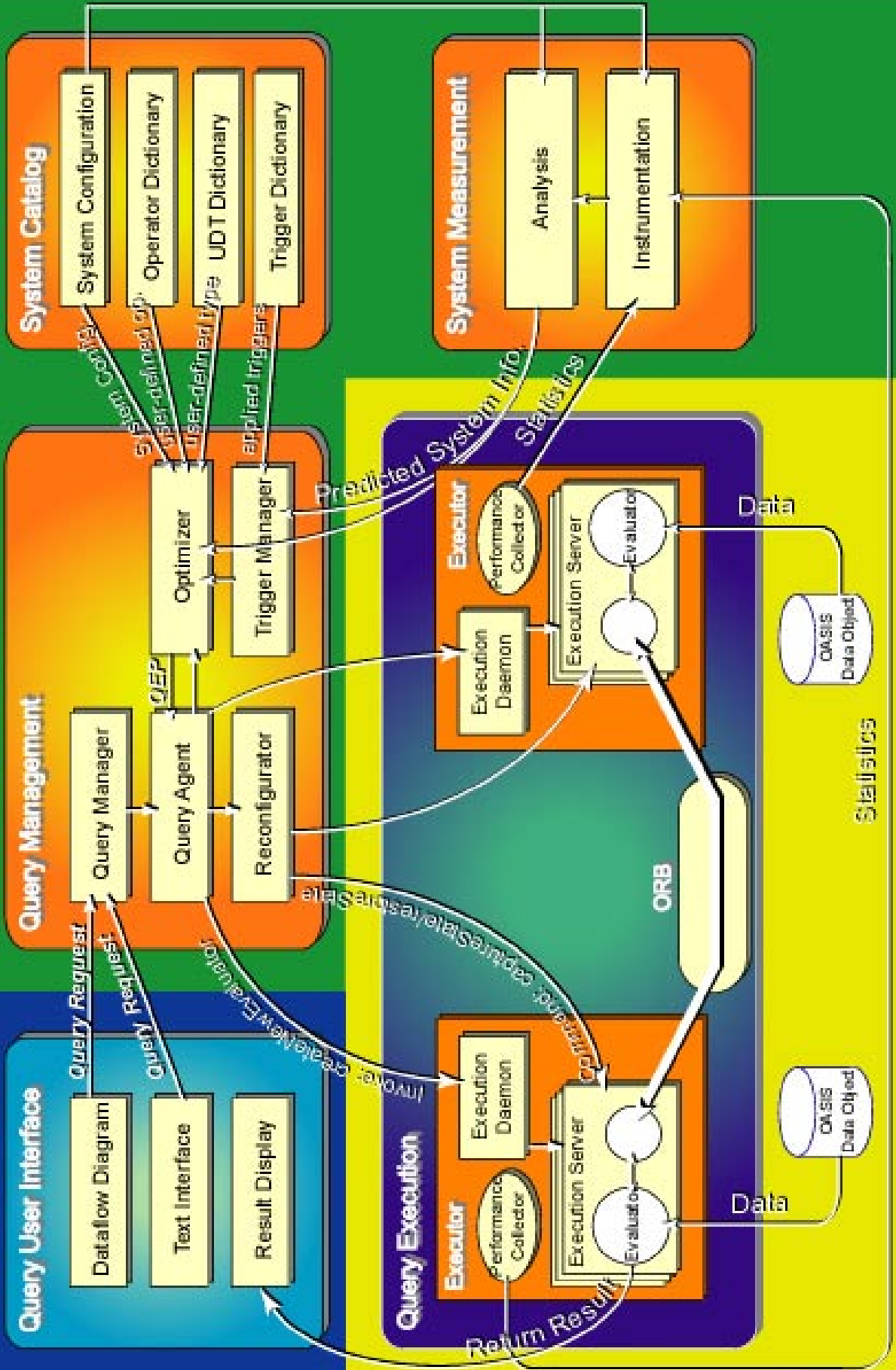


Conquest

Characteristics:

- ⌘ **Data streaming paradigm** efficient for query execution (full intra-query parallelism)
- ⌘ **User-defined operators** and data types
- ⌘ **Automatic re-optimization** of long running queries
- ⌘ **Java/CORBA-based runtime** environment using heterogeneous workstation farms

Query Execution Supporting Subsystem





Publications on Conquest

- ⑩ "Parallelizing User-Defined Functions in Distributed Object-Relational DBMS",
Kenneth Ng, Richard Muntz
International Database Engineering and Applications Symposium, Montreal, August, 1999.
- ⑩ "Dynamic Query Re-Optimization",
Kenneth Ng, Zhenghao Wang, Richard Muntz, Silvia Nittel
SSDBM99, Cleveland, Ohio, July, 1999.
- ⑩ "Conquest: CONcurrent Queries over Space and Time",
Silvia Nittel, Kenneth Ng, Zhenghao Wang, Richard Muntz
International Workshop "Integrated Spatial Databases: Digital Images and GIS" (ISD'99),
Portland, Maine, June, 1999.
- ⑩ "The Design and Implementation of the Conquest Query Execution Environment",
Frank Fabbrocino, Eddie C. Shek, and Richard R. Muntz
UCLA CSD Technical Report #970029, July 1997.
- ⑩ "Scalable Exploratory Data Mining of Distributed Geoscientific Data",
E.C. Shek, R.R. Muntz, E. Mesrobian, and K. Ng
Second Intl Conference on Knowledge Discovery and Data Mining, Portland, Oregon, Aug. 1996.
- ⑩ "On Heterogeneous Distributed Geoscientific Query Processing",
E.C. Shek, E. Mesrobian, and R.R. Muntz
RIDE-96, New Orleans, Louisiana, Feb. 1996.
- ⑩ "Fast Spatio-Temporal Data Mining of Large Geophysical Datasets",
P. Stolorz, E. Mesrobian, R.R. Muntz, E.C. Shek, J.R. Santos, J. Yi, K. Ng,
S.Y. Chien, H. Nakamura, C.R. Mechoso, and J.D. Farrara
1st Intl Conference on Knowledge Discovery and Data Mining, Montreal, Canada, Aug 1995.



Overview

⌘ Motivation

⌘ Conquest Overview

⌘ **Operators in Conquest**

⊠ User-Defined Operators

⊠ Re-Optimizing Operators

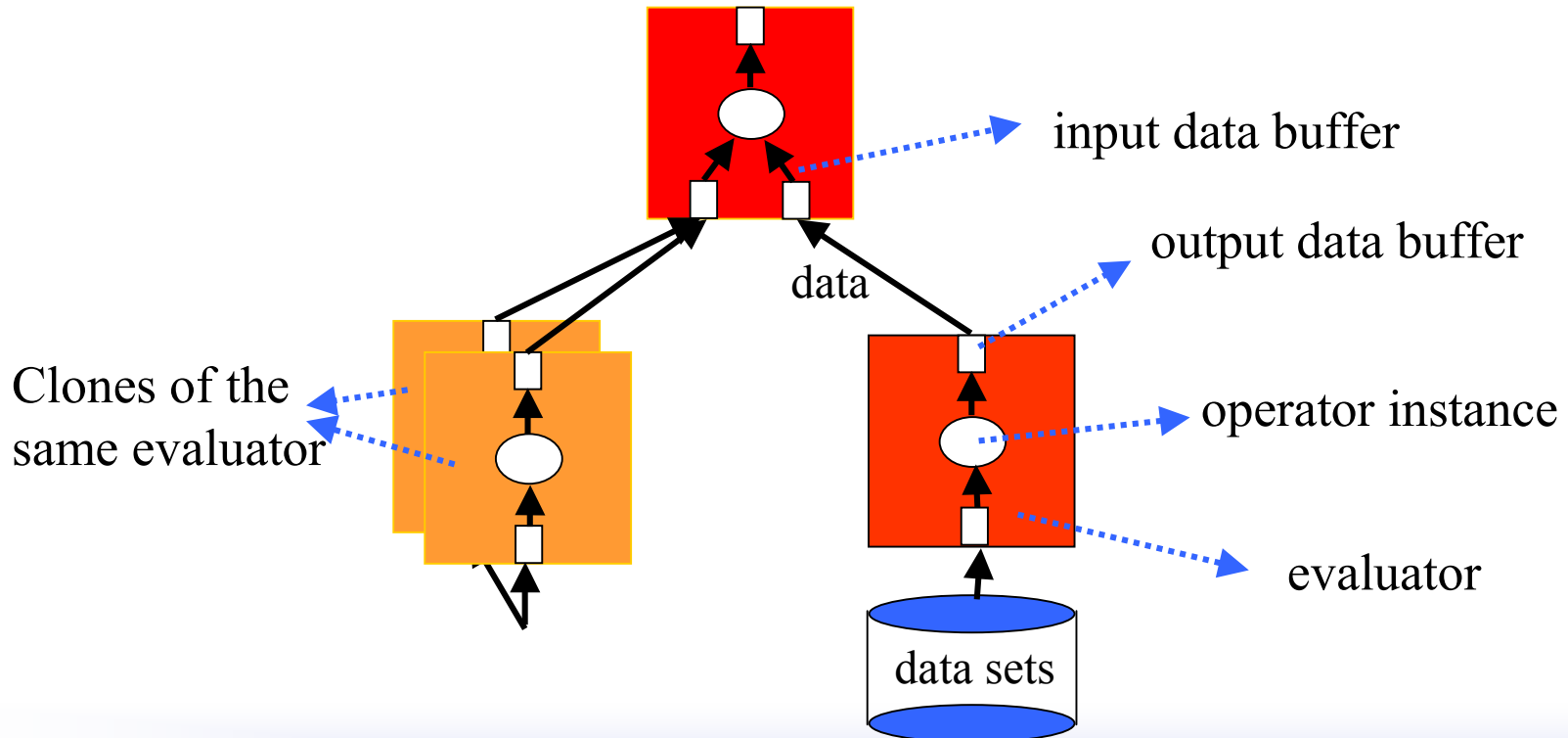
⌘ Trigger-Based Re-optimization in Conquest

⌘ Conclusions



Data Streaming

Use of stream processing paradigm in a distributed workstation environment





User-Defined Operators in a Parallel Execution Environment

⌘ Requirements

- ⊞ large investments in data analysis code
- ⊞ wrapping of legacy code (different PLs)
- ⊞ easy integration into parallel environment

⌘ Problems

- ⊞ complex execution environment (intra-query parallelism, (re-)optimization)
- ⊞ interdependence with execution and optimization environment -- system support code complex



User-Defined Operators

⌘ Related work

- ⊞ extensible optimizer: Volcano [Graefe93]
- ⊞ operator implementation support:
 - ⊞ **Conquest Operator Definition Language [DML95]**
 - ⊞ **Interface Definition Language (OMG)**

⌘ Conquest approach for operator support:

- ⊞ programming interface is object-oriented, procedural (instead of data streaming)
- ⊞ *Tool opGen* automates integration of operator and provides system support code for operator



opGen *Characteristics*

Goal: minimize amount of support code to be written for an operator

⌘ Assistance of operator development:

- ☑ all *system support* code for an operator is *automatically generated*
- ☑ system support code is *complete*, and *separated* from operator logic (changes in Conquest execution environment do not affect any operator)
- ☑ can be used *with different languages*



opGen *Operator Support*

⌘ 1) Specification of operator 'sections':

- ⊞input/output: in-/output data types definition
- ⊞parameter: definition of execution parameters
- ⊞state: accumulate execution information

⌘ 2) *opGen* generates support classes (Java/JNI code)

⌘ 3) Client programming:

- ⊞*init()*, *next()* for data streaming paradigm
- ⊞*isSuspensable()*, *getContext()*, *setContext()* for participation in run-time re-optimization



opGen (*cont.*)

⌘ 4) code generation

- ⊞ operator in C++ with required Conquest functions
- ⊞ 'make'
- ⊞ generation of Java executable class and C++ shared library

⌘ *Conquest run-time*: use of Java class, and calls of C++ functions.



Overview

⌘ Motivation

⌘ Conquest Overview

⌘ **Operators in Conquest**

⊠ User-Defined Operators

⊠ Re-Optimizing Operators

⌘ Trigger-Based Re-optimization in Conquest

⌘ Conclusions



Stream Operators

⌘ Problems:

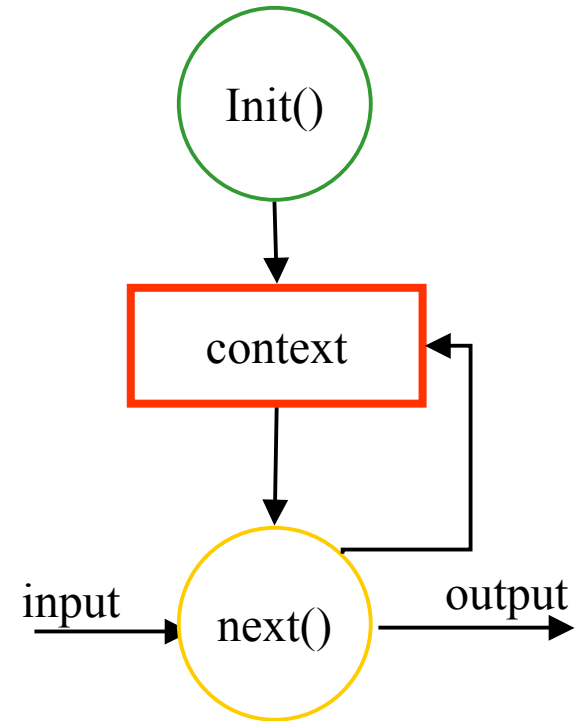
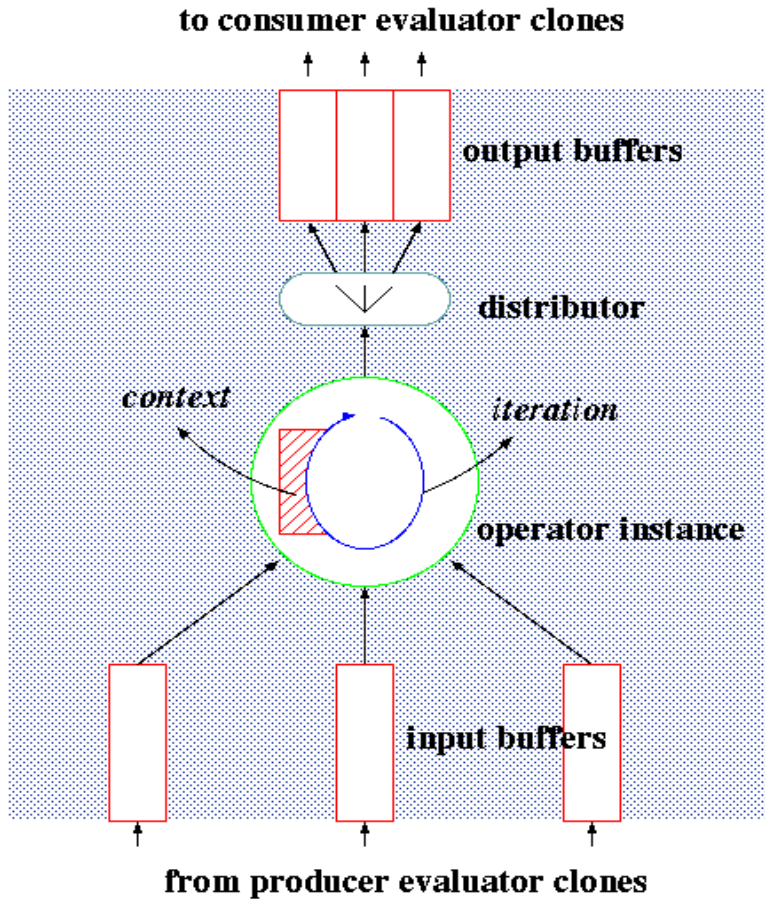
- ☑ Which operators are “good” to re-optimize?
- ☑ What is an efficient way to reconfigure an operator?

⌘ Objective:

- ☑ *Categorizing operators so that re-optimization options can be systematically derived.*



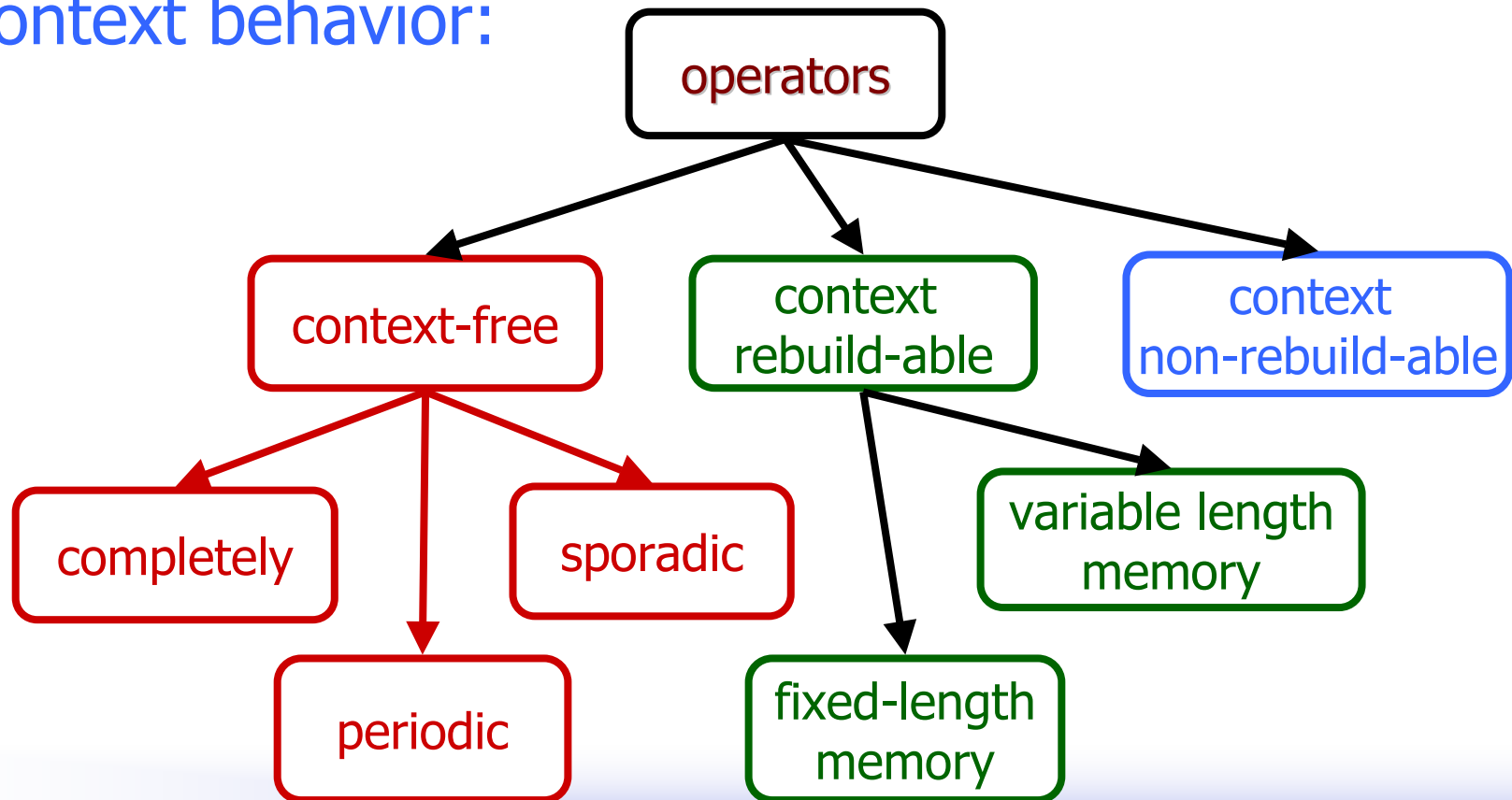
Stream Operators





Classification

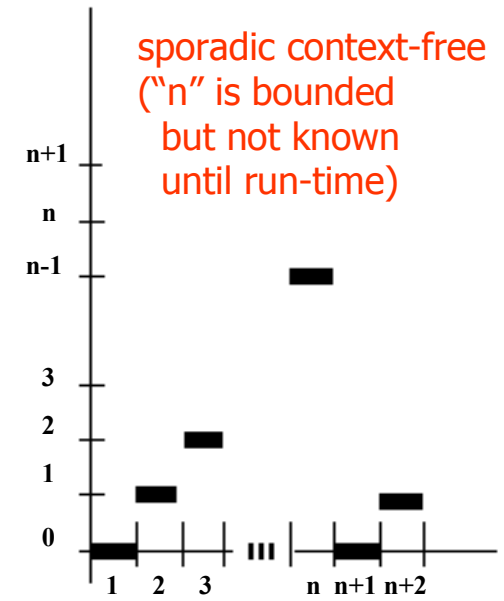
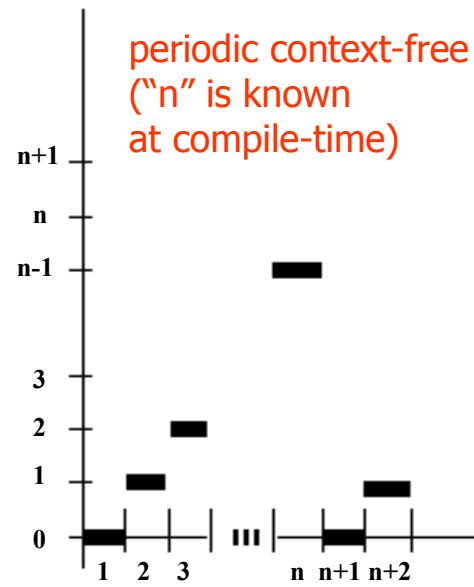
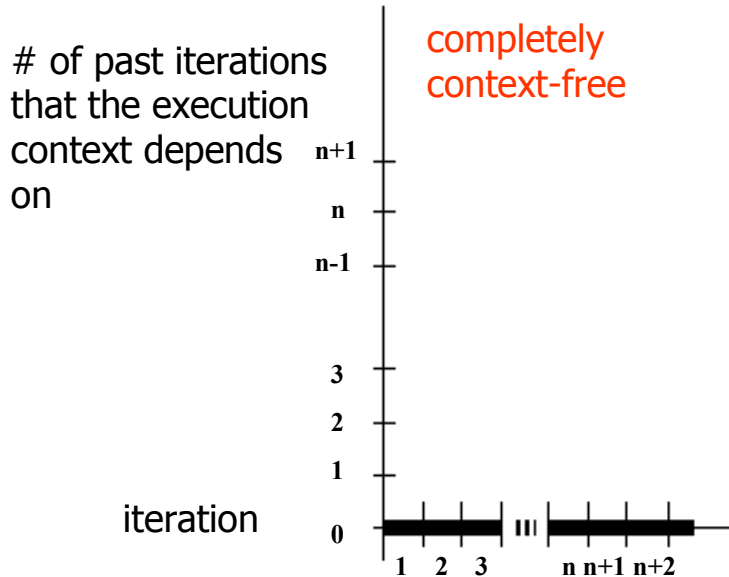
Framework definition to classify operators regarding context behavior:





Context-free Operators

Operators that can return to a null context at the end of some iterations:



Example:

selection

Weekly temperature avg.
(input data:
time-stamped regular
chronological order)

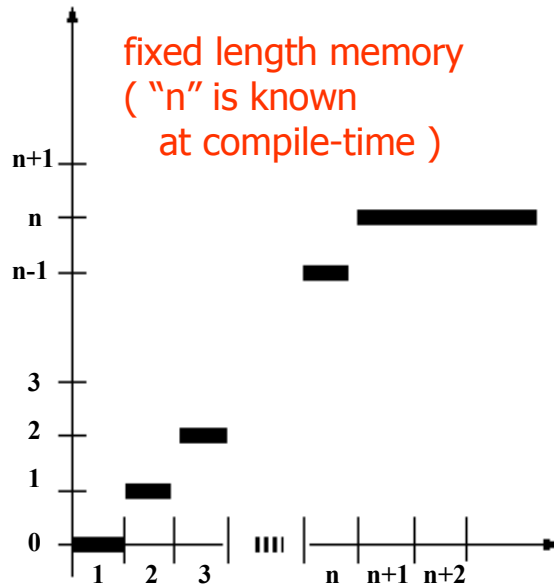
Weekly temperature avg.
(input data:
time-stamped irregular
chronological order)



Context-Rebuildable Operators

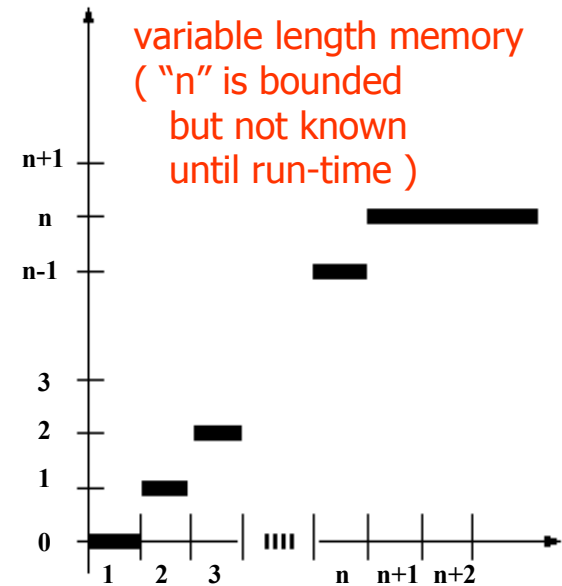
Operators whose context depends on the last iterations:

of past iterations
that the execution
context depends on



Example:

5-day temperature running avg.
(input data:
time-stamped regular
chronological order)



5-day temperature running avg.
(input data:
time-stamped irregular
chronological order)



Overview

- ⌘ Motivation
- ⌘ Conquest Overview
- ⌘ Operators in Conquest
 - ⊗ User-Defined Operators
 - ⊗ Re-Optimizing Operators
- ⌘ **Trigger-Based Re-optimization in Conquest**
- ⌘ Conclusions



When to Re-Optimize

⌘ Other Approaches

- ☒ when a sub-query is done (Kabra, D.J.DeWitt. SIGMOD 98).
- ☒ when delay is detected in accessing remotely stored data (T.Urhan, M.J.Franklin, L.Amsaleg. SIGMOD 98)

⌘ Shortcomings

- ☒ no timely response to changes / up-to-date knowledge;
- ☒ not considering operator's sensitivity to changes;
- ☒ not considering both system parameters and data characteristics.



Run-time Optimization

⌘ Problems

- ☒ Change of system configuration and resource availability can be unpredictable;
- ☒ Not possible to accurately estimate data characteristics in advance.

⌘ Requirements

- ☒ Timely response to the change (or up-to-date knowledge) is expected;
- ☒ Extensibility is desired in an extensible query processing system.



Triggered Run-Time Optimization

⌘ Triggering Approach

- ⊞ Self-responding processing system
- ⊞ Run-time system parameter measurement
- ⊞ Trigger rules evaluation
- ⊞ User-defined trigger rules (E-C-A structure)

⌘ Advantages

- ⊞ Response to the change (or up-to-date knowledge) in an active and timely manner;
- ⊞ New trigger rules can be defined any time to reflect the new requirements.



Trigger Rules

⌘ System Trigger Rules

- ⊞ Trigger rules that are **common to all queries**
 - ⊞ *e.g., trigger for a workstation shutdown, etc.*

⌘ Query Plan Trigger Rules

- ⊞ Trigger rules that are **common to all configurations** of a submitted query
 - ⊞ *e.g., trigger for a frequent empty input or output buffers of an operator, etc.*

⌘ Operator Trigger Rules

- ⊞ **Built-in Operator Rules**
 - ⊞ *e.g., trigger rule for “scan” operator*
- ⊞ **User-defined Operator Rules**
 - ⊞ *e.g., trigger rule for “minima” operator*



E-C-A Rules for Triggers

Events
occurring during
execution

System Parameters	workload change
	...
Data Characteristics	selectivity update
	...
Time	10:00 pm
	...
External Command	

Check if predefined
Condition
Is true

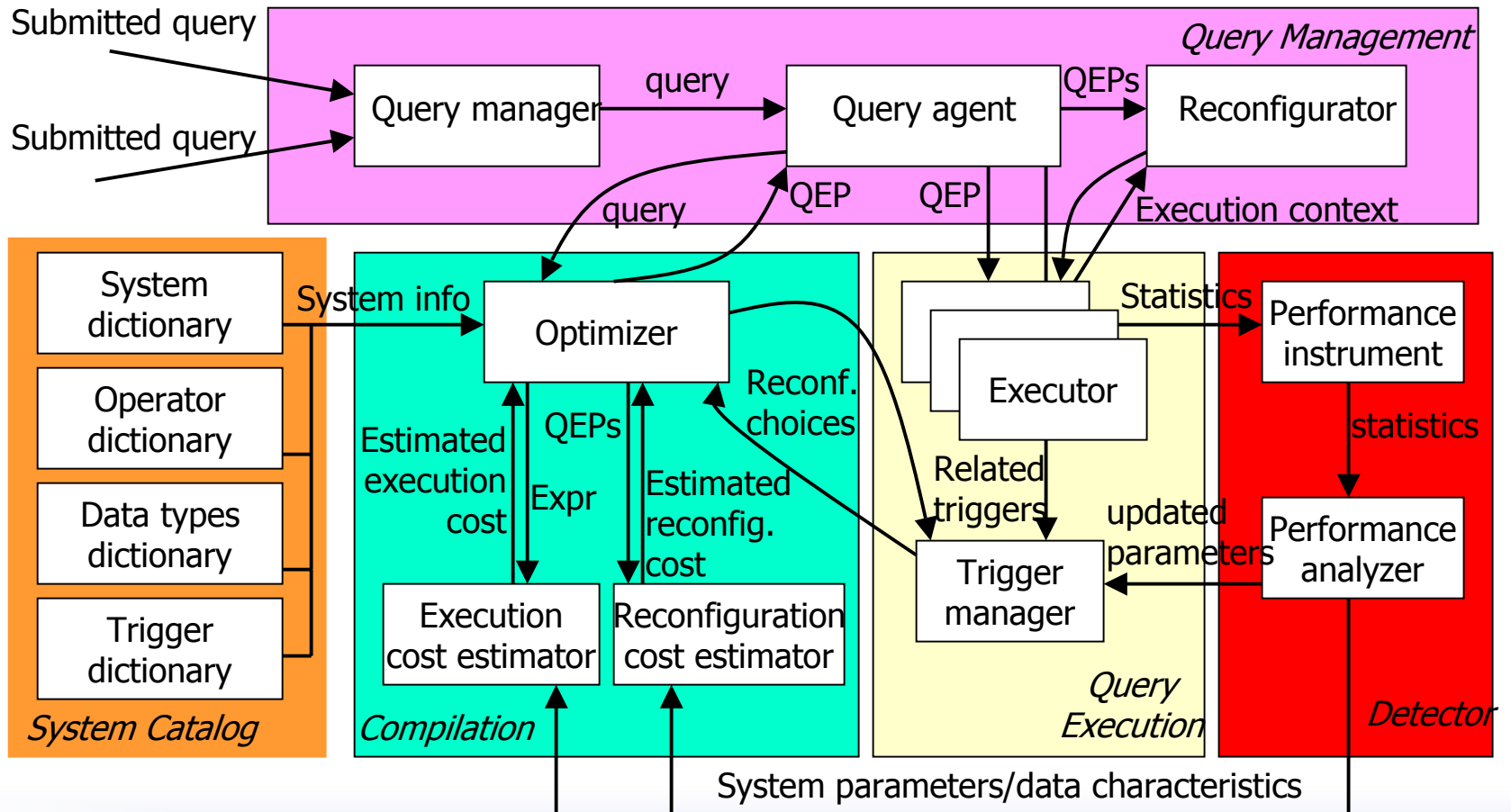
Workload on machine X: current > 1.2 * start-up
...

If Yes, execute
following
Action

Relocate operator A
Repartition data stream from A to B
...
Clone operator A to more instances
...



Query Processing





Reconfiguration Decision

$$\begin{aligned} & \textit{ResidualExecutionCost}(\textit{newQEP}) \\ & + \textit{ReconfigurationCost}(\textit{currentQEP}, \textit{newQEP}) \\ & + \textit{threshold} \\ & < \textit{ResidualExecutionCost}(\textit{currentQEP}) \end{aligned}$$



Conclusions

- ⌘ **Conquest:** Extensible, efficient data mining tool for querying huge data sets using data streaming paradigm
- ⌘ Paradigm to easily wrap legacy code as Conquest operators
- ⌘ Classification of operators for re-optimization
- ⌘ Run-Time re-optimization of long-running queries



Status

- ⌘ Software in 3rd generation (OrbixWeb 3.1, Java)
- ⌘ Execution environment, GUI, optimizer, re-optimizer complete
- ⌘ Use in NASA-sponsored **ESP²Net** for collaborative scientific experiment with JPL, Scripps, U Arizona and HRL

- ⌘ Software available at:
[Http://dml.cs.ucla.edu/projects/oasis/Conquest/conquest.html](http://dml.cs.ucla.edu/projects/oasis/Conquest/conquest.html)



Future Work

- ⌘ Use in distributed environment over internet using virtual private networks
- ⌘ XML-based interface for queries