Agrios

Agrios integrates R and SciDB, minimizing data transfer between the two systems, through the use of a database-style cost-based optimizer employing query transformations. Contributions include:
- semantic mappings between the systems’ languages
- a cost-based inter-system interaction model
- performance results

Agrios is a “hybrid” system – a query’s operations can be performed at either R, or SciDB, and data may be stored on both systems.

Bonneville Optimizer

At the heart of Agrios lies Bonneville, an extension of the Columbia database optimizer. Bonneville utilizes Columbia’s methods for exploring the search space, but differs in several ways. Bonneville:
- is designed for use with an array data model. As such, the transformations, rules, etc., that guide exploration of the search space differ
- considers the location of data objects in assigning costs to plans

Compact representation

Bonneville compactly represents the search space through the use of a memo structure. The memo represents the plan space with two mutually recursive object types: groups and multiexpressions.
- A multiexpression is an operator with groups as inputs
- A group is a collection of equivalent multiexpressions

Staging and costing

A staging assigns execution locations to each of a query’s operations. A single query can have many stagings, and each staging may differ in its cost. The optimizer identifies the least-expensive staging.

Transformations

Transformations expose new staging opportunities, potentially uncovering the optimal plan. Traditional transformations are employed, such as associativity and commutativity, as well as array-model specific transformations, such as subscript pushdown.

Pruning the search space

The optimal plan can be identified in a large search space thanks to aggressive pruning techniques employed by Bonneville.

As a top-down optimizer, Bonneville can quickly determine an upper bound of an acceptable plan. This upper bound is used to halt exploration of the search space; if a plan under consideration exceeds the upper bound, no further exploration of that plan is attempted. This saves resources that would be wasted exploring suboptimal plans.

Many thanks to David Maier, Kristin Tufte, and the PSU Datalab. This research is funded in part through NSF Award # 1110917.