TCAP Compiler/Optimizer

Template MetaProgramming

User Query Graph

```
Reader<Supervisor> ("db1", "allSup")
SelectComp<Supervisor, Supervisor>
Lambda <bool> getSelection (
    Handle<Supervisor> arg) {
    return makeLambdaFromMethod (arg,
getDept) !=
makeLambdaFromLiteral("EE");   }
SelectComp<Employee, Supervisor>
Lambda <bool> getSelection (
    Handle<Supervisor> arg) {
    return makeLambdaFromMethod (arg,
getDept) ==
makeLambdaFromLiteral("CS");   }
Lambda <Employee> getProjection (
    Handle<Supervisor> arg) {
    return makeLambdaFromMethod (arg,
getEmp); }
Writer<Employee> ("db1", "csEmp"
```

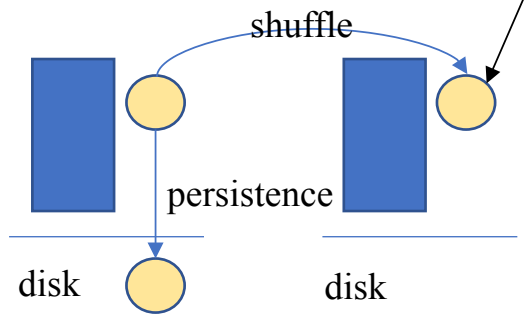
Optimized TCAP Program

```
//read in data
In(sup) <= ScanSet ("db1", "allSup");

//push down projection
AppliedMethodCall1Select1(sup, emp)
<= Apply (In(sup), In(sup, emp),
'Select1', 'getEmp');

//fusion the two filtering condition
AppliedMethodCall0Select1(sup, emp,
dept) <= Apply
(AppliedMethodCall1Select1(emp),
AppliedMethodCall1Select1(sup, emp),
'Select1', 'getDept');

Filtered0(sup, emp) <= Filter
(AppliedMethodCall0Select1(dept),
"CS", 'Select1');

//write the results
Out(emp) <= WriteSet("db1", "csEmp");
```

Compiled Pipeline Stages

```
Each thread:
for (…) {
  VecListPtr curVecList = getNextVectorList(…);
  for (PipelineStage q : pipeline) {
    try {
        curVecList = q->process(curVecList);
    }
    catch (NotEnoughSpaceInCurrentBlock e) {
        postprocess(…);
    }
  }
}
```

Distributed Execution:



PC Object has identical representation across disk and network, so no (de-) serialization overhead.

PC API/Lambda Calculus

PC Execution Engine

PC Object Model **(Support complex objects and virtual functions; No (de-) serialization overhead)**