Netezza meets MapReduce
Abstractions for data intensive computing

Severin Hacker and Robert J. Simmons and Carsten Varming

October 8, 2007

Abstract

The high-level goal of our project is to better understand parallel data-intensive computing. In particular we want to explore, compare and evaluate two different systems: the Netezza platform and Google’s MapReduce, in part to understand the class of problems that can be efficiently implemented across both architectures. We then plan to develop a common way of specifying problems across both architectures, and using this system, compare the systems against a single machine and against each other with respect to usability and performance. We will do our evaluation against large data sets such as housing data from Allegheny county.

1 Introduction

The Netezza platform is a unique approach to the problem of “data warehouse appliances,” providing (cheap) parallel computational power very close to storage. This is an interesting and very general model - for instance, the idea of placing special-purpose computational power very close to the hard drive goes back at least to NASD [GNA+98]. It is furthermore an architecture that has been successful at performing massively parallel, IO intensive computations, and Google’s success with MapReduce [DG04] demonstrates that with suitable software interfaces, it can also be an intuitive model for programmers to work with.

However, MapReduce is designed for a file-based environment where each disk is attached to a commodity general-purpose computer, whereas Netezza is a database-based environment where each disk is attached to a very special-purpose computing unit that consists of a CPU, a FPGA, and dedicated memory. We believe that there is a class of problems that are well-suited to both environments. One example is relational queries within databases that would be impossible to predict and index ahead of time such as looking for all housing records where a house was sold to someone with the same last name as the previous owner within some time range. Another example is aggregate queries that would be similarly difficult to index, for instance, asking “What is the average change in property values over the last two decades?” on a database of property values based on an arbitrary grid provided by the user.

We plan to demonstrate that a simple framework can be used to efficiently program to a cluster computing environment running a MapReduce implementation such as Hadoop [Pro07] as well as the more impoverished environment of a FPGA assisted disk array that exists on the Netezza. We will show that this framework can specify an important class of problems and solve them without significantly degrading performance when compared to a platform-specific solution.
2 Programming the Netezza

The Netezza allows the application programmer to write C++ code that will be executed in parallel on a small computational device close to storage; Netezza’s advertising material refers to this combination of a FPGA, PowerPC processor, dedicated memory as a “Snippet Processing Unit” or SPU. Instructions are given to these devices by two user-defined functions, one which acts essentially like an intelligent filter and another that aggregates data; the behavior of these two functions seems at least loosely analogous to the Map and Reduce functions of Google’s MapReduce.

However, the computational model of the Netezza is more complicated than seems necessary, requiring both C++ class inheritance and an explicit state passing style. The explicit state passing style of programming is unfamiliar to most programmers, as almost all languages handle this. We propose to construct a language in which you can program the Netezza in way more natural to programmers, and also hopefully in a way that deals more naturally with the restrictions imposed by the computation environment. In doing so, the programs that we specify should should be able to target not only the Netezza in an efficient manner, but also other distributed data architectures (for instance, the cluster model that MapReduce assumes). Furthermore, it will be important for debugging and testing purposes that the implementation be able produce a design that can run on a workstation where all the data is contained on a single drive.

3 Evaluation

We will evaluate our solution by developing a number of examples that require processing of a large amount of data, such as a large set (> 1 GB) of housing data from Allegheny county. The data set includes information about the current and previous owner, exact location, sale prices, market value, number of bedrooms, and condition of the house. An example record is given in Figure 1.

Given this data, we want to answer high-level questions that would be time-consuming for a single-machine database to handle, but that can be more efficiently queried in a distributed architecture. Questions that we have considered which may fall into this category include:

- What is the average change in property values over the last two decades (by neighborhood, by grid, by street, etc.)?
- Which houses were sold to family members (people with the same last name), and at what times?
- How long do people stay in the same house before they sell it (on average)?

We will then quantitatively compare the performance of compiled vs. hand-coded solutions to the selected queries on each of the architectures (single-machine, Hadoop cluster, Netezza), as well as qualitatively comparing the ease which these questions can be specified on a single architecture vs. being specified in our general system that targets each of the various architectures.
4 Timeline

This project will take roughly eight weeks, and we plan to divide our time as follows:

- **Week 1-2**: Sort out single-machine and cluster computing (Hadoop) environment, run hand-coded tests in this environment.
- **Week 1-4**: Figure out Netezza computing environment, hand code tests and measure performance of Netezza on tests.
- **Week 3-5**: Solidify design of domain-specific language.
- **Week 4-7**: Implement domain-specific language.
- **Week 7-8**: Evaluate solution, generate final presentation and report.

5 Related Work

One concern of this project is understanding programming models that target a computational environment where hard disk intensive operations are distributed across many computational units (CPUs or FPGAs). A major success in this area has obviously been Google’s MapReduce [DG04] [Läm06]. Work on stream languages has similar goals but targets a different environment - individual multicore systems; [GTA06] mentions several such languages. In general, programming models for distributed programming programming have a long history, such as work on remote procedure calls [BN84]. Stored procedures [Eis96], which allow a database query optimizer to see a large body of work as a unit, may provide another way of thinking about this problem; we do not yet have an understanding of what the relationship is between stored procedures and Netezza’s C++-coded user defined functions.

Run-time code generation will likely be something that we will consider, even if its implementation is outside the scope of this project. Run-time code generation has been implemented in languages from ML with the Fabius system [LL96] to the ’C (tick-C) extension of ANSI C [PHEK99]. ’C is also interesting because of its similarity to standard C - this makes it easily usable for systems programmers comfortable with C, and also makes converting existing codebases simpler. We will probably want to do something similar in order to make the transformation process simpler - this will hopefully allow us to leverage a tool like CIL [NMRW02], a code parser/generator for C.

The idea that certain tasks are well-suited to compilation onto FPGA units instead of general purpose processors has been termed “spatial programming” by Seth Goldstein’s Tartan project [MCC+06] - most of the work in that area is concerned with integrating FPGAs within a single-computer architecture rather than a distributed computing environment.

References


Municipal Code:114 PITTSBURGH - 14TH WARD
Parcel ID:0175-P-00070-0000-00
School District:City Of Pittsburgh
Neighborhood Code:11413
Owner Name:SCHUMACHER HENRY B
Property Location:305 EAST END AVE, PITTSBURGH, PA 15221
Tax Code:Taxable
Sale Date:8/6/2004
Owner code:Regular
Sale Price:$83,500
State Code:Residential
Deed Book:12149
Use Code:TWO FAMILY
Deed Page:476
Homestead:No
Abatement:No
Farmstead:No
Lot Area (SQFT):2,880
2006 Market Value:$69,200
Tax Bill Mailing:PA HOUSING FINAN AGENCY, FINANCE DIV, PO BOX 15057, HARRISBURG, PA 17105-,
Change Notice Mailing:305 EAST END AVE, PITTSBURGH, PA 15221-0000,
Total Rooms:10
Style:MULTI-FAMILY
Bedrooms:3
Stories:2.5
Full Bathrooms:2
Year Built:1900
Half Bathrooms:0
Exterior Finish:Brick
Heating:Central Heat
Roof:Slate
Cooling:Basement:
Fireplace(s):0
Grade:C+
Garage:0
Condition:Poor
Finished Living Area:2923 Square Foot
History Owner 1:SCHUMACHER HENRY B
History Sale Date 1:08/06/2004
History Sale Price 1:$83,500
History Owner 2:THOMPSON ROBERT BRUCE & ARETI L (W)
History Sale Date 2:09/28/2000
History Sale Price 2:$57,000
History Owner 3:MARY E FORTNER & LEON P (H)
History Sale Date 3:04/29/1993
History Sale Price 3:$48,000
Geo Status:Found
Geo Accuracy:8
Geo Latitude:40.444926
Geo Longitude:-79.894697
Geo Address:305 EAST END AVE, PITTSBURGH, PA 15221

Figure 1: An example record from the housing data set