INDIAN INSTITUTE OF MANAGEMENT CALCUTTA

WORKING PAPER SERIES

WPS No. 718/ December 2012

The IIM Calcutta Data Centre: A Retrospective, Programming Tutorial and Future Directions

by

P.Srikant
Doctoral Student IIM Calcutta, D. H. Road, Joka P.O., Kolkata 700 104 India

&

B.B.Chakrabarti
Professor, IIM Calcutta, Diamond Harbour Road, Joka P.O., Kolkata 700 104 India

The IIM Calcutta Data Centre : A Retrospective, Programming Tutorial and Future Directions

P. Srikant
IIM Calcutta

I describe the development from scratch of a high-frequency financial research data centre, using freely available tools. The design provides an acceptable level of performance at a low cost. I provide examples of usage, and suggest possible paths for the further development of the data centre.

Address for Correspondence:

P. Srikant
Fellow Programmes and Research, IIM Calcutta
Diamond Harbour Road, Joka, Kolkata - 700 104, India
phone: +91 91636 56187
email: p.srikant@gmail.com

(Questions and comments are welcome at the above address)

1 Introduction

The Data Centre was at the Finance Research and Trading Laboratory, IIM Calcutta, was initiated in April 2010. This data centre is now a unique repository of Indian tick data, from which a stream of academic research in asset pricing and microstructure has started emerging. The data is also used by students inclined towards trading careers to devise strategies in partnership with financial firms. The time now seems appropriate to take stock of the data centre's achievements over the past two and a half years, and to critically re-examine the rationale for various design choices made at the time. Also, this article provides short programming vignettes which demonstrate best practices for using this data. It concludes with some suggestions on how to develop this resource in the years to come.

2 Context and Motivations

In the US, CRSP and Compustat databases are easily available in most institutions through distributors like Wharton Research Data services. As [Anshuman and Badrinath, 2007] points out, empirical research in Indian financial markets is constrained by the lack of a proper database for financial research. While some financial data is indeed available through the Prowess database developed by the Centre for Monitoring the Indian Economy, the data quality is highly

suspect and the data is exceedingly slow to access. Also, intraday data is typically not available in commercial databases.

The development of databases is an important precursor to financial research. There have been several instances where academics have taken the lead in establishing such databases. To cite a few examples, the Center for Research in Security Prices at Chicago was created when a professor was asked by the industry to determine historical stock market return for which they had to create a portfolio, which later on became the CRSP equally and value weighted indexes and the basis for almost all event studies in the US. Empirical market microstructure was born when Bob Wood at the Univ. of Memphis created the ISSM database from intraday trading data. Ghon Rhee created the PACCAP database at URI which provided a unique competitive advantage for their Ph.D. students.[1]

IIM Calcutta was the logical place to try to attempt to build and intraday centre, since it already possessed a Trading Lab. The Trading Lab has several dealing workstations which receive a combined price feed for several exchanges through Newswire-18, which aggregates feeds from various exchanges and redistributes them in a common format. Therefore, we already were receiving (but not recording) high-frequency data on Indian financial markets.

At the time, the Trading Lab was used to conduct MBA classes and some workshops. Our goals in establishing a data centre were twofold. Firstly, we wanted to build a resource which would further underscore IIM Calcutta's research leadership amongst Indian schools. Secondly, we sought to impart more of a research orientation to the Trading Lab.

3 Strategy and Initial Approaches

As a first step, I wanted to examine if I could listen in to data from the Newswire18 feed. The Newswire18 feed was received by a server in the server room of the finance lab, and broadcast to other workstations. I discovered that we already had some documentation and example code on how to listen to the feed. Essentially, the example program initialized communication with the network, and repeatedly polled the network to read and process data.

I first customized the example program in C to listen in to all messages on the network channel, and to rewrite these messages to a file. This was relatively straightforward to do. While this demonstrated that data could be recorded, it was first of all critical to institutionalize a process of recording data. The recording process needed to start in the morning, and be terminated after market closing hours. To record data systematically, the process of recording needed to be standardized and automated. I created a number of batch files which could be either run manually or as scheduled tasks, to start and stop recording data. I also trained the laboratory support personnel in how to perform these tasks.

An average day generates approximately 70 million records, or 15 GB of data. Managing this data presents its own challenges. As a first concern, the free hard disk space on the server was only 100 GB, and its capacity would have been exhausted in two weeks. Therefore, I was

[1] I thank Prof. Malay Dey for these examples

compelled perforce to compress the files. I considered various possible compression formats, including writing my own, before selecting gzip. Gzip [Nelson and Gailly, 1995] is a widely used file compression program, which reduced the file sizes to about 1.2 GB each (The average size of compressed files now is 2.5 GB). This bought us some critical breathing time - I could now store a couple of months of data on the server. Later, the institute procured a 1 terabyte drive for some $150, which could hold two years of data. This was the only financial cost incurred by the project at the time.

4 Design Choices

The design of a large database requires various design choices. At 15 GB a day, a year's data would occupy some 4 TB of space. Even compressed, the data would occupy approximately half a terabyte.

[Jacob and Shasha, 1999] list various models for time series systems. A tick database for financial instruments is typically characterized by irregular time series of non-periodic data. The level of activity across stocks varies widely - some stocks have thousands of trades each hour, while others may only have a few dozen. The design of an appropriate architecture needs to consider the nature of data, the frequency of updates, and the use cases for such data.

I therefore planned to always retain the compressed daily files as a backup, while investigating other storage formats. Since the archival process was organized around a daily recording routine, I decided that this would also be the unit of storage.

Some specific design choices I considered are listed below:

1. *All information or select fields:* Different exchanges publish different kinds of information. For example, the NSE publishes only one level of bid and ask, while the BSE publishes 5 levels. While I received suggestions to standardize and store only a few fields, which would have led to smaller files, I decided not to throw away any information - while unnecessary information could always be ignored, it would be difficult if not impossible to recreate data fields which we dropped.
2. *One file or many files:* Live data is received in the form of messages, ordered by time. Messages from all exchanges and all instruments are broadcast on the same network. In general, the communication network does not guarantee that messages will be sent in the order they were received. Our timestamp on messages has a minimum resolution of one second. If all messages are stored in the same file, potentially valuable sequence information is retained. This sequence information - the ordering of events within a second - would be irretrievably lost if each ticker was stored in a separate file.
   It is worth mentioning here that I had considered other attempts to deal with this issue. For example, NEEDS provided a sequence count for its data on the TSE - this is timestamped to the nearest minute. The NYSE TAQ dataset sorts all trades in a day by ticker, and uses an indexed sequential file to access a desired ticker. (I understand a project was subsequently undertaken to split files by ticker, which seems an inelegant and lossy solution.)
   Financial research is rarely done on one stock, so a frequent case would be to run a

program for all stocks, or a list of stocks, for a day at a time. Any user with the rudiments of programming knowledge could use elementary data structures like a hashtable for each stock to store intermediate computations of interest. In addition, one file most closely mimics a live data stream, so this allows for easy translation of strategies from a historical to a live environment.

3. *Choice of database:* While compressed files can be used for many research tasks, a database typically provides indexes that speed up queries for specific instruments. Unlike in an investment bank, I did not have an unlimited budget at my disposal. Commercial time-series databases typically cost over a hundred thousand dollars. They also need hardware which was better than what was available at the Trading Lab at the time. Based on a thorough evaluation of existing technologies, I chose to store a copy of the data in HDF5 [Folk et al., 1999], a freely available database developed at the National Centre for Supercomputing Applications (NCSA). HDF5 has been successfully used to manage large data sets like NASA's Earth Science Data and Information System. Soem institutions that use HDF5 include the Argonne National Laboratory, Deutsche Bank and Mathworks. [2] I was therefore confident that HDF5 was a reliable and scalable format with an established user base. I used PyTables, a Python interface to HDF5, to create and read from HDF5 databases. PyTables uses a table-oriented approach with advanced indexing [Alted and Vilata, 2007]and reasonable compression featues. At the time, some of these features were only available in the commercial version (PyTables Pro), but now these functionalities have been merged into the regular product.
   While I was working on this, we were approached by Xenomorph via OptiRisk. The possibility was that we would get their tick data software, so I discontinued further development on the HDF5 front. The Xenomorph offer never materialized, and we had space limitations which prevented us from keeping a parallel copy of the database in HDF5 format, so, apart from a proof of concept, the bulk of the data remains in compressed zip files. HDF5 via PyTables remains my choice for a free tick database with acceptable performance.
4. *Access to data:* It was agreed at the time that the recorded files would be deployed on a Linux server which would support multiple user logins, so that people could remotely access the files. This has unfortunately not happened yet, and obtaining data still involves physically going to the financial lab and copying files to other media. I typically carry a portable hard drive to the finance lab, and again upload files to a UNIX server (which the MIS department have kindly granted me access to
5. *Tools to replay data:* The gzip decompression library is available in most programming languages, including C, Perl and Python. Since uncompressing and decompressing data is a resource-intensive operation, it is simpler to use on-the-fly decompression in most cases. I developed scripts in Perl and Python which could read these compressed files. The fastest way to read these files is typically by opening a pipe to an operating system command:

```
srikant@sasux2.iimcal.ac.in
bash-2.05b$ ls -l /home/srikant/flashcrash
-rw-r--r--
```

[2] A list of other users is available at http://www.hdfgroup.org/HDF5/users5.html

```
   1 srikant    users      2436360072 Oct  6 06:30 20121005.txt.gz
bash-2.05b$ gzcat 20121005.txt.gz
```

Using this, a 2.4 GB file can be processed on our HP-UX SAS server in IIM Calcutta in about 16 minutes. The clever use of OS pipes and UNIX text commands like awk for filtering can greatly speed up simple research tasks. For more complicated tasks, I developed example programs in scripted languages for Python and Perl - these were intended for quickly developing programs. For high-performance tasks, I developed a C++ library, using which required writing and compiling programs. To illustrate the simplicity of writing an interpreted script, I provide portions of an example script in Perl in the next section. By finetuning the buffer size, a perl script can read a day's worth of data in 25 minutes.

Many of these design choices, as already noted, were dictated by technological and financial constraints. While some of these constraints no longer hold, these design choices still appear to have been reasonable.

5 Tutorial Example 1: Reading the flush file with Perl

This section serves as a tutorial introduction to a user trying to use the tick data in the data centre for the first time. I used this to analyze the flash crash of 5 October 2012, when a broker mistakenly sent a large order on a Nifty basket.

I first populate a list of tickers of interest, and create and initialize a data structure (refer to the perldsc, the Perl Data Structures Cookbook) to hold the information I am interested in. (There are the inputs to the algorithm I used to reconstruct order flow)

```
use Data::Dumper;
$Data::Dumper::Terse =>1;
use Compress::Zlib;

@tickers = ‘cat niftytickers.txt‘;
chomp @tickers;

%StockHash =();
%fieldsOfInterest = (   bid => 0,
                        ask => 0,
                        bsz => 0,
                        asz => 0,
                        last => 0,
                        cvol => 0,
                        lvol => 0    );

foreach $t (@tickers) {
        $StockHash{$t} = {};
        foreach $f ( keys ( %fieldsOfInterest ) )         {
               $StockHash{$t}{$f} = 0;
        };
}
```

I then open the file, and based on the size of the file and the hours of trading , I skip to approximately half an hour before the basket was sent:

```
my $gz = gzopen("20121005.txt.gz", "rb") ;
$gz->gzseek( 1700 * 1024 * 1024 );
```

Each record in the file is written on a separate line. Various items are delimited by the pipe (|) character. An record consists of a line number, a timestamp, a source identifier, a ticker identifier, and various field=value pairs. This can be processed by the following Perl code that extracts the NSE ticker, and extracts key-value pairs for the record if the ticker is in the Nifty index :

```
while ( $gz->gzreadline($line) > 0) {
        chomp $line;
        next if ($line !~ "\<138\>" or $line !~ "Stock" );
        my @elems  = split(/\|/, $line );
        ($ticker) = ( $elems[1] =~ /\<138\>(.*)EQN/ );
        next if ( !exists ( $StockHash{$ticker} ));

       ($lineno, $hh, $mm, $ss, $chr) = split( /\:/, $elems[0] );
        $ctime = $hh * 3600 + $mm * 60 + $ss;
        shift @elems;         shift @elems;         shift @elems;

        %mH = ();
        foreach $e(@elems)         {
                next if ($e !~ /=/ );
                ($key, $val) = split (/=/, $e);
                $mH{$key} = $val;
        }

}
```

Finally, I close the file:

```
$gz->gzclose();
```

As can be seen through the above example, it is easy for a novice programmer to quickly develop a script to perform a research task of interest. Similar programs can be written in Python, and, for efficiency, in C++. The use of a programming language provides the maximum degree of flexibility. A simple script to extract rollups like minutely series of bids and asks in SAS / STATA format was also developed.

6 Tutorial Example 2: PyTables

PyTables is built on top of the HDF5 library, using the Python language and the NumPy package, and provides fast access to large amounts of hierarchical data. PyTables provides for advanced indexing and compression options. I provide here another tutorial example to calculate bid ask spreads for a particular ticker on a particular date. The code example below works with Enthought Python, and retrieves all trades for a stock on a given day. This is an example of an ’in-kernel’ query with PyTables, which allows efficient filtering of rows corresponding to a specific stock.

```
from tables import *

h5file = openFile("20121005.h5", mode = "r" )
```

```
table = h5file.root.mktdata.ticks

for row in table.where ( 'name␣==␣"Stock<139>500469"␣' ):
        if row['chr'] == 'T' :
                print row['time'], row['last'], row['lvol']

h5file.close()
```

Obviously, various design choices need to be made while storing the data. Message sizes vary significantly. A space-efficient solution for columns of variable length involves storing header information in a PyTable, and using pointers to a HDF5 EArray to access the relevant data. A compromise is to store a few common fields in separate columns, and the rest of the information in another column as shown below. The following program reads the compressed daily file, and stores it in a HDF5 database (some code is not shown) :

```
from tables import *
colFields = [ 'bid', 'ask', 'last', ... ,'asz', 'bsz' ]
from gzip import open as gzopen

class Tick( IsDescription ):
        l = UInt64Col()              #lineno
        timestamp = UInt64Col()       #timestamp
        readableTime = StringCol(8)
        name = StringCol(128)         #exchange and ticker
    ............
        asz = Float64Col()
        bsz = Float64Col()
        other = StringCol(256)

f = gzopen("20121005.txt.gz", "rb")
h5file = openFile("20121005.h5", mode = "w", title = "Test␣file")
group = h5file.createGroup("/", 'mktdata', 'Market␣Data')
table = h5file.createTable(group, 'ticks', Tick, title = 'TAQ', expectedrows=100e6
)

for line in f :
        elems = line.split('|')
        if len(elems) <= 4 :
                continue
        lineno, hh, mm, ss, c = elems[0].split(':')
        r = hh + ":" + mm + ":" + ss
        ticker = elems[1]
        exchange = elems[2]
        exchticker = elems[2] + elems[1]
        ts = int(hh) * 3600 + int(mm) * 60 + int(ss)

        t = table.row
        t['l'] = int(lineno)
        t['timestamp'] = ts
        t['name'] = exchticker
        o = ""
        for e in elems[3:-1] :
                fname, fval = e.split('=')
                if fname in colFields:
                        t[fname] = fval
                else:
```

```
                    o = o + fname + "=" + fval + "|"
        t['other']= o

        t.append()

table.flush()
h5file.close()
f.close()
```

7 Current State and Future Directions

I begin by noting that the current state of the data centre leaves much to be desired, Firstly, there is a lack of awareness on what data is available, and how to access and use it. This has led to the data being under-utilized. I attempt to partially address these issues through the tutorial examples in this article. Secondly, the recorded daily data can still be accessed only by going to the Trading Lab and copying files over. It would be preferable to run queries against the data stored in a central repository, instead of copying files into a separate computer. Research can only flourish in an open academic environment that allows participants to collaborate and share work. Thirdly, the data still resides primarily in compressed gzip files. This has led to frustration among some users who are interested in looking at data for a particular ticker instead of replaying the whole market. It was not possible at the time to replicate this information in HDF5 databases, because of a lack of adequate computing infrastructure - I was severely constrained by available memory. Much of my testing and development was actually performed on a laptop with under 20 GB of free space.

I believe that the recorded high frequency data sets need to be uploaded to a Linux server. It will be more efficient for users to run scripts directly on this server than trying to download or copy files back and forth. IIM Calcutta should provide logins on this server (like WRDS does) to researchers at other institutions at a moderate fee, so that the research in intraday asset pricing and microstructure on Indian markets develops. The data should continue to reside on our server, so FTP / SCP access of files can be limited to 20 MB at a time. We can perhaps provide STATA and SAS on this server at an additional fee, or R for free, to facilitate analysis on the server itself. For high performance, a variety of cluster computing solutions have been developed which provide for task scheduling and load balancing across multiple compute nodes. Such solutions are available for example at the University of Michigan's research computing grid, and have become more affordable.

The architecture I had in mind involves storing each day's data in a separate HDF5 database, and distributing these databases across multiple computation and storage nodes. HDF5 still looks a good choice for a free high-performance database.

For further performance improvements, we need to process several files in parallel. This is a further justification for keeping a separate file for each date, since each file can then be handled separately. In recent years, various open source cloud computing technologies developments have gained wide user bases, the most promising of which seems to be Apache's Hadoop [Borthakur, 2007]. This allows queries on multiple files to be processed simultaneously via map-reduce operations[Dean and Ghemawat, 2010]. The core idea is that instead of relying on monolithic databases, we try to split and parallelize operations across cheap compute nodes. A

combination of Hadoop and HDF5 should be an efficient yet relatively low-cost solution for research data services. Hadoop also comes with its own database, HBase, which might also prove to be an efficient solution.

One of my goals in writing this article at this time is also to caution against following unfruitful lines of work. I believe it is not advisable to split the files by ticker, or to use a relational database to store this data. Another caution is that a continued effort is required to ensure that the recording process continues to function smoothly, and that the data centre's recorded data is fully backed up.

I am fully conscious of the limitations of what I have developed. However, many of these design choices were dictated by resource limitations, some of which no longer hold. Also, some technologies have gained wide acceptance within the last two years. An exciting direction forward is the use of Apache Hadoop to parallelize computations so that files corresponding to multiple dates worth of data can be processed simulataneously. Deploying such a platform will need competent technology inputs. I understand cloud computing and technologies like Hadoop are actively being investigated by the MIS department, and we may be able to leverage on their expertise.

As future work, a library could be developed to provide programming interfaces which insulates the end user from the underlying storage format. A possible programming semantic to calculate vwap over an interval may look like this:

```
volume = value = 0;

sim = new Simulator(  );
sim.setStart(20121005:09:00:00);
sim.setEnd(20121005:09:30:00 );
sim.run();

while ( record = sim.getNextRecord() )
{
   if ( record.ticker == "ABB" and record.hasField( "lvol" ) )
   {
               volume += record.lvol;
               value += record.last * record.lvol;
    }
}

sim.stop();

print value / volume;
```

8 Acknowledgments

In early 2010, Prof. Dey visited IIM Calcutta and taught an course in Empirical Market Microstructure, which I attended. There was then a complete absence of high-frequency data in India, which made it impossible to perform research on intraday asset prices and market microstructure. Malay pointed out that the absence of a database was not just a constraint, but

also an opportunity to build one. I would like to thank Prof. Dey for suggesting the project, and for his support and guidance during the project.

Based on Malay's suggestion, I decided to investigate the technical feasibility of setting up a data centre. I am grateful to my faculty advisor, Prof. B. B. Chakrabarti, for agreeing that this project would also serve to meet my PhD summer research project requirement.[3] We believed at the time that creating such a centre would add to the institute's credentials as a premier finance school, and signal the institute's commitment to research.

I would also like to acknowledge the consistent and diligent work of Priyanka, the Assistant Manager of the finance laboratory, in ensuring that the recording process has continued to operate smoothly. In addition, interns at the lab at the time included Shishir Kumar, Rishiraj Diwakar, and Rahul Kumar, who helped developed a few applications using this data.

Finally, I would like to thank the University of Delhi, for my undergraduate education in computer engineering; and my former colleagues who taught me what I know about how to record, manage, and use high frequency data.

References

F Alted and I Vilata. *OPSI: The indexing system of PyTables 2 Professional Edition*. PyTables Pro. http://www. pytables.com, 2007.

VR Anshuman and SG Badrinath. A framework for a securities market database in india. *Economic and Political Weekly*, 2007. URL `http://www.jstor.org/stable/10.2307/40276721`.

D Borthakur. The hadoop distributed file system: Architecture and design. *Hadoop Project Website*, 2007.

J Dean and S Ghemawat. Mapreduce: a flexible data processing tool. *Communications of the ACM*, 2010. URL `http://dl.acm.org/citation.cfm?id=1629198`.

M Folk, RE McGrath, and N Yeager. Hdf: an update and future directions. *Geoscience and Remote*, 1999.

KJ Jacob and D Shasha. Fintime - a financial time series benchmark. *SIGMOD Record*, 1999.

M Nelson and JL Gailly. The data compression book 2nd edition. *M & T Books, New York, NY*, 1995.

9 Web References

1. http://www.selectorweb.com/algorithmic_trading.html An overview of technology options by a Wall Street developer

[3] The summer research project was executed under the guidance of Professors Ashok Banerjee, Malay Dey, and Gautam Mitra.

2. http://www.elitetrader.com/vb/showthread.php?threadid=81345 discussion in an online forum