A REPORT

ON

BOLT Broadcast Status Reporting and Analysis System

BY

Name of Student	ID No	Discipline
Tejaswini V S	2009H112035G	M.E Software Systems

AT

CMC Ltd, Mumbai

A Practice School-II Station of

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI

July, 2011

A REPORT

ON

BOLT Broadcast Status Reporting and Analysis System

BY

Name of Student	ID No	Discipline
Tejaswini V S	2009H112035G	M.E Software Systems

Prepared in partial fulfillment of the

Practice School – II,

Course No. BITS G639

AT

CMC Ltd, Mumbai

A Practice School-II Station of

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI

July, 2011

Acknowledgement

I would like to acknowledge the support of Dhairyasheel Pawar, Project Manager at CMC Ltd, my team lead Navneet Haldankar, his team and practice school instructor Vijay Bhasker Reddy.

Tejaswini. V.S

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE
PILANI (RAJASTHAN)

Practice School Division

Station: CMC LTD, Mumbai	Centre: Mumbai
Duration: From July 04, 2011	To: December 14, 2011

Date of Submission: December 14, 2011

ID No.	Name(s) of student(s)	Discipline
2009H112035G	Tejaswini V S	M.E Software Systems

Name(s) of expert	Designation
Navneet Haldankar	Team Lead

Name of PS Faculty:

Key words: Tandem broadcast, analysis and reporting, volatility index

Project Area(s) : Software Engineering, Software Architecture, Performance, Networking, Inter Process Communication

Abstract:

CMC Limited is a systems engineering and integration company. The BSE Online Trading system (BOLT) is CMC's on-line Trading System for Trading in Stocks. The System is operational at Bombay Stock Exchange. Two different projects are dealt with in this report. BOLT broadcast status reporting and the Calculation of Volatility Index, former being the main project. The trading system broadcasts various data for different purposes. Presently, information about activity in the trading system is obtained by querying the trading system. To relieve the trading system of handling such queries, a Linux machine is to be set up that keeps a replica of offline data from tandem. One such data is the broadcast information. This broadcast data can further be used to derive meaningful information that can help in analyzing the health of the trading machine. The Volatility Index is calculated to represent market's expectation of volatility or fluctuation in price. It uses Sensex future and options, bid and ask prices. The methodology used for calculation is similar to the one used by Chicago Board of Exchange.

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE

PILANI (RAJASTHAN)

Practice School Division

RESPONSE OPTION SHEET

Station: CMC Ltd. Centre: Mumbai

IDNO. & Name : 2009H112035G, Tejaswini VS

Title of the Project: BOLT Broadcast Status Reporting and Analysis

Usefulness of the project to the on-campus courses of study in various disciplines. Project should be scrutinized keeping in view of the following response options. Write Course No. and Course Name against the option under which the project comes.

Refer Bulletin for course No. and course Name.

Code No.	Response Option	Course No & Name
1.	A new course can designed out of this project	NA
2.	The project can help modification of the course content of some of the existing courses	NA
3.	The project can be used directly in some of the existing Compulsory Discipline courses (CDC)/ Discipline Courses Other than Compulsory (DCOC) Emerging Area (EA) etc. courses.	NA
4.	The Project can be used in preparatory courses like Analysis and Application Oriented Courses (AAOC) Engineering Science (ES), Technical Art (TA) and Core courses.	NA
5.	This project cannot come under any of the above mentioned options as it relates to the professional work of host organisation	YES

Contents

1. Introduction

The trading in BSE is managed by two of HP NonStop Integrity Servers / Tandem with 16 single-core Intel Itanium Processors and 4 GB of memory on each processor. Applications that are run are NonStop SQL supporting a set of custom programs called the Central Trading Engine (CTE). BOLT is part of the CTE. Given below is a brief introduction to the BOLT system.

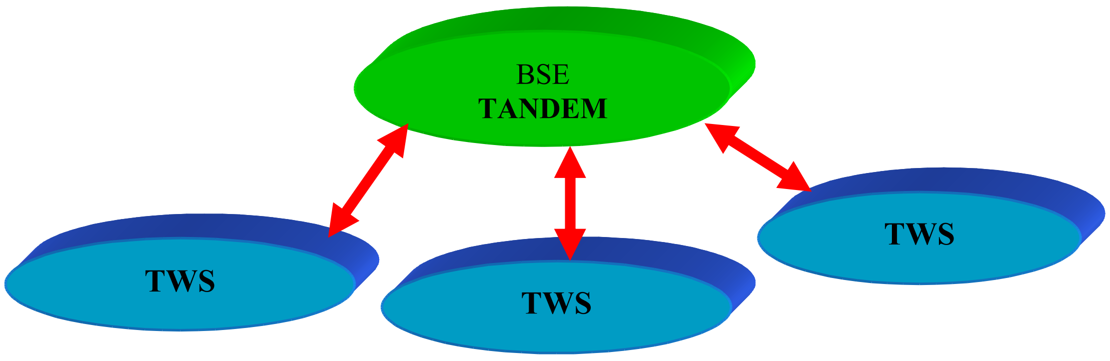


Figure a: BOLT

In the figure above, we can see Trader Work Stations (TWS) connected to the Tandem Nonstop Servers. The trader with a trading application typically can place buy/sell orders and track the orders. Various other functionalities have been implemented in CMC's custom application for client front-end called the TWS. BOLT has the provision for a work station to be third party software too. Services provided by the trading engine is handled by various servers, there are different servers for trades, orders, communication. Details of trade-activity are stored in the Tandem database and in the logs. Various data such as Market picture data is broadcast periodically from the live server.

The Tandem machine is also responsible for calculation of market indices. The most popular being SENSEX is an index to the health of the market. It is calculated using the market prices of around thirty Scrips that are major market players. One of the projects dealt with in this report involves the Volatility Index. This index is a benchmark to represent stock market volatility. The computation requires Sensex future and options bid and ask prices. Detailed description is further given in the later sections. The requirement specification documents and literature over the internet have served in understanding the underlying problem statement.

The other project deals with Broadcast status reporting and Analysis. The market picture data is a broadcast of scrips that are active in orders and trading. This data can be used to identify the activity of scrips in trading. Scrips are the names of companies/ a certificate which represents the commodity being is traded. Analyses of the broadcast data and information that can be derived from it can let us find how many scrips were active at a particular point in time in a particular server or which was the most active scrip of the day. Such status reports can help in identifying the root cause of abnormal behavior of the system when required.

The above project began with discussions with the client regarding the requirements, since the requirement was not completely frozen. The software development life cycle can be said to be iterative. At each stage, minor changes to the requirements were incorporated into the design. Extensive discussions with the experts were involved during the design of the system. The brainstorming would then lead to a suitable design that meets functional and performance requirements. A prototype of the system has been developed and further improvement on the implementation has been mentioned at the end of the design section.

2. Scope of the projects

There were two projects Broadcast Status Reporting and Analysis System and Calculation of Volatility Index, the former being the main project.

BOLT Broadcast Status Reporting and Analysis System:

- ✓ Create a client on Linux that collects data from Tandem and process it.
- ✓ Derive further information from the collected broadcast data.
- ✓ Create a database to store all the processed data and subsequently the analyzed data..
- ✓ Programming skills: Network programming, Inter process communication, SQL, scripting.

Calculation of Volatility Index:

- ✓ Understand the requirement, computation method
- ✓ Create a program that calculates the volatility index based on the equation given in the requirement specification.

3. Hardware and Software Specifications

- ✓ Linux SUSE Enterprise server
- ✓ GCC compiler
- ✓ MYSQL, POSTGRES
- ✓ C programming and Shell scripting

4. Project One: Broadcast Status Reporting and Analysis System

Aim: To display the basket and scrip wise activity of the trading machine (TANDEM) with respect to market picture broadcast. The display is a graph depicting the number of market picture data generated in the trading engine per basket and per scrip, within a particular interval. In addition to the above, the application counts the number of broadcasts received in all streams for all message types.

4.1 Requirement Specification

1. The Linux machine should act as a client that collects data from tandem.
    - 1.1 The client must collect all the message types. (viz., Market picture data, Time broadcast data, Sensex broadcast data and others)
        - 1.1.1 The data should be collected via UDP transport protocol by joining a multicast group.
        - 1.1.2 The broadcast data thus collected must be decompressed.
        - 1.1.3 A count of number of broadcasts per minute must be calculated. We can call this *Counter data*.
    - 1.2 The client must collect Basket master data.
    - 1.3 The client must collect Scrip master data.
2. The application should be able to provide data that can be used to plot the counter data along with other parameters and reference data.
    - 2.1 The parameters for Market picture data are Basket ID and System ID.
    - 2.2 The reference data for Market picture data are Averages of counts for last five days and last twenty five days.

2.3 Similar parameters and reference data can be added to the counts of other message types.

2.4 The process of arriving at the above mentioned integrated data should not impact the rate at which data is collected from tandem broadcast.

3 The application must provide *hooks* in itself at meaningful stages such that intermediate data within the application can be tapped by other applications or piece of code.

3.1 A different application or process should be able to access the continuous stream of Market picture data that is collected by this application.

4 The application must store all the data collected into the database. (Both directly collected broadcast data and derived data such as counter data).

5 An End-of-Day process calculates the averages (reference data) for that day. The process is run after the closing of the market.

6 The status of broadcasts in terms of counter data must be plotted and displayed on a web based client. There are two types of displays:

6.1 Display one-minute's data periodically on a graph. This is near-real-time.

6.2 A user-interface to query historical data and plot a graph accordingly.

6.3 A web-server must be employed to stream data for periodic display and to receive queries from clients and provide appropriate data by in turn querying the database.

7 The web-based client must provide facility to the user to subscribe to a particular stream of counter data, for example, stream of Market Picture Data counts and/or Sensex broadcast data counts.

Non-Functional Requirements: The process that receives the broadcast should not be block at any point in time.

4.2 Design

Based on the requirements, the following design was formulated. In the pages ahead are the Block diagram of the architecture, Schematic diagram and Data flow diagrams.

The application receives broadcast data from tandem. Broadcast for all messages and all streams are received. There are four streams and each stream broadcasts sixteen types of messages as of now. The four streams, broadcast data at different rates. The rates at which they are broadcast depend on the purpose for which it is used. The broadcast sent to media such as television are slower when compared to broadcasts sent to traders who use Algorithmic trading or low latency trading. The frequency at which each message type in each stream is broadcast varies too. For example, the Market Picture broadcast is sent at the rate of one thousand broadcast packets per minute, Tandem server also broadcasts its time every one minute, details of the opening and closing of market is broadcast only in the morning and evening respectively. Figure 1. below is the block diagram of the architecture and gives a high level view of the system. We can see that there are different processes collecting broadcasts of different streams. Within each stream, different message types are in turn handled by different processes as shown in Figure 2. Message types with highest broadcast frequency are handled by a single separate process for example, Market Picture broadcast. While message types with lesser frequencies are handled together by one single process. In Figure 2, consider the end to end flow of the market picture data. The main receiver processes joins a multicast group and receives the broadcast for that stream, say FCAST and puts it in a message queue. The filtered market picture data is taken up by the receiver and sent to the next message queue. The message queues act as a buffer to collect broadcast data and not block the sender at any time or lose received packets incase the receive end of the message queue is not fast enough.

The size of the message queue is appropriately set to act as a buffer.

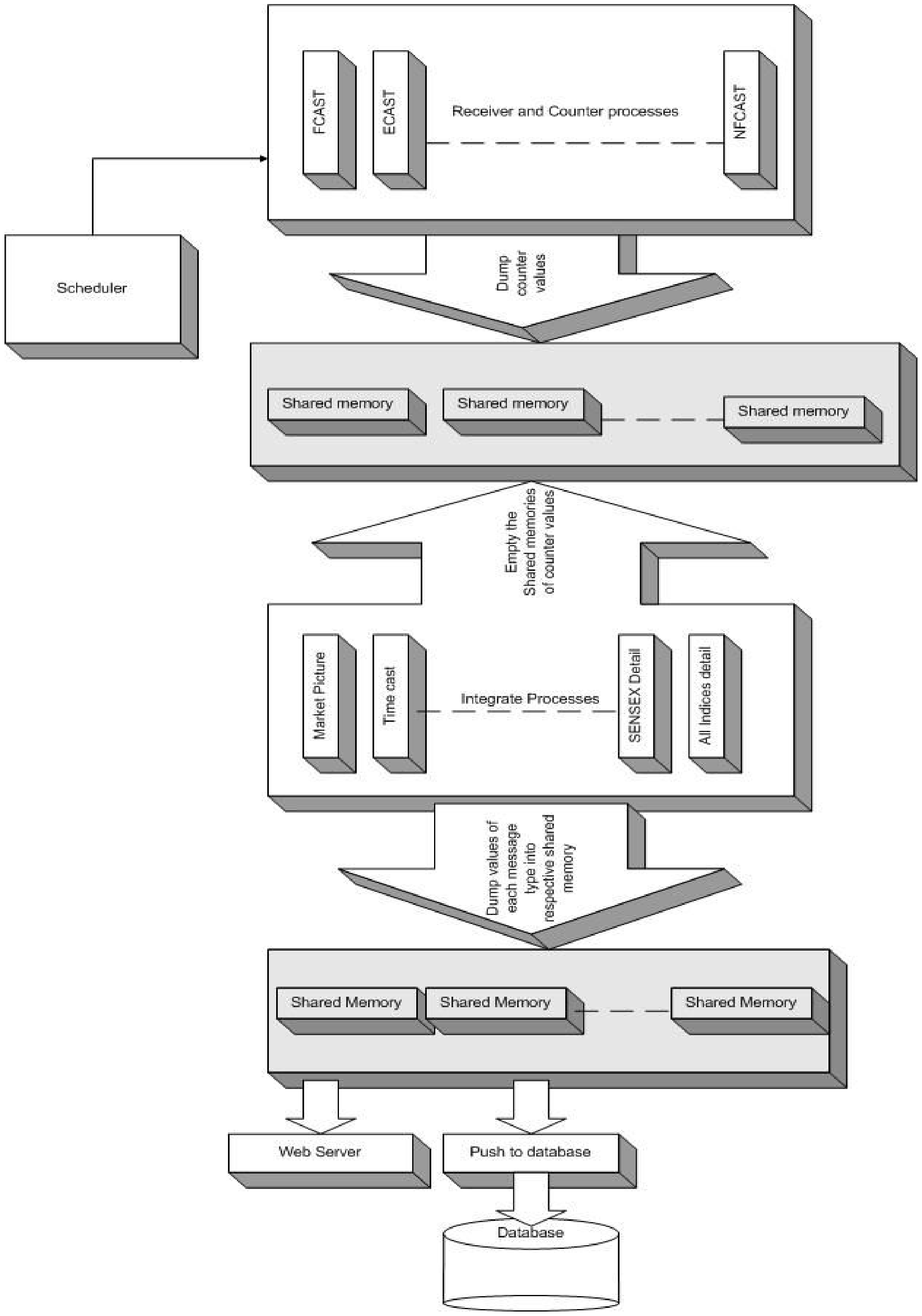


Figure 1: Block Diagram

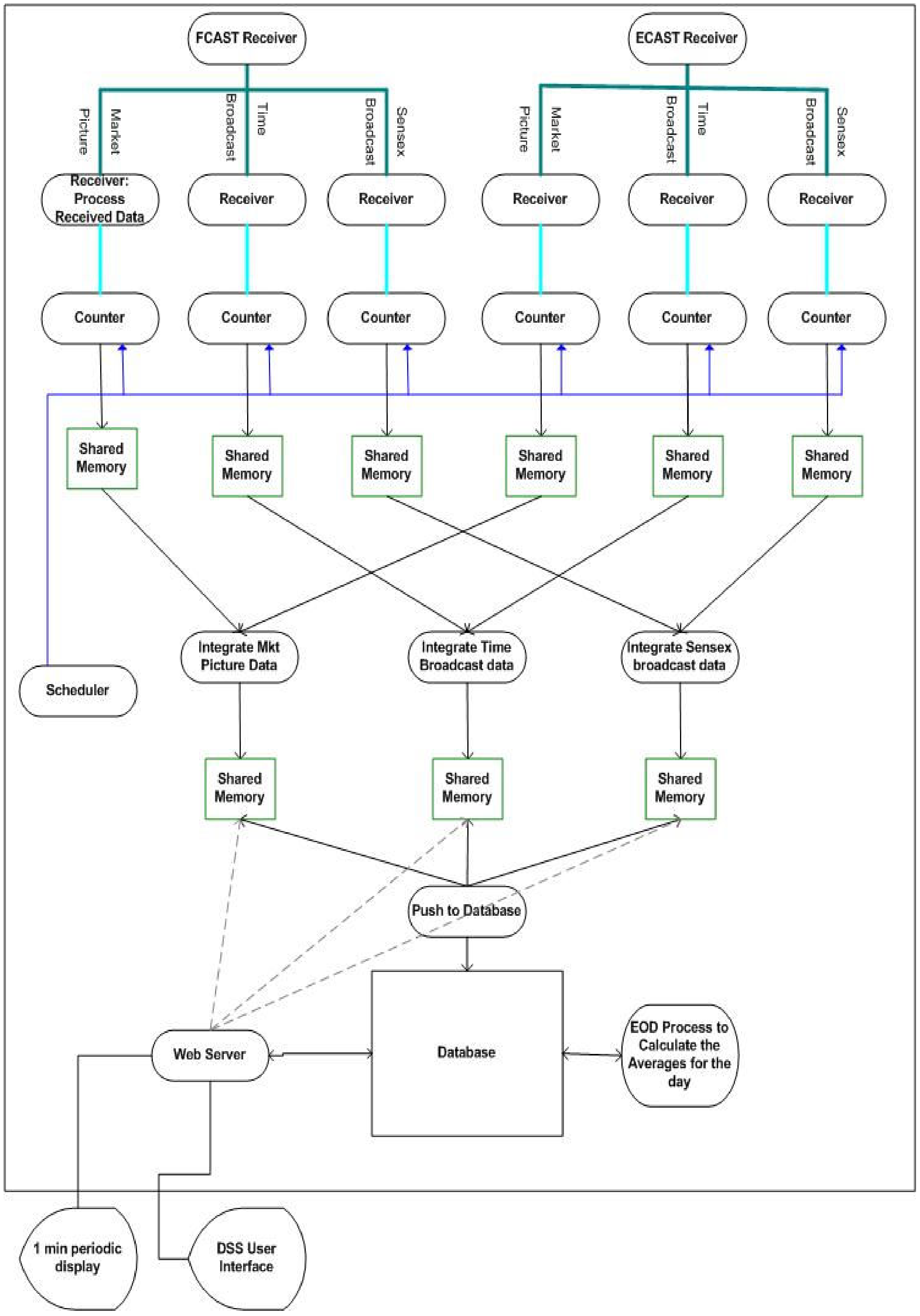


Figure 2: Schematic Diagram

The receive end of the second message queue in the hierarchy is implemented in a 'Counter' process. This process counts the number of market picture broadcasts received per scrip per minute. Only those scrips for which the broadcast is received are considered for counting.

The counter data consists of the timestamp of the latest received broadcast, scrip code for which the broadcast was received and the count of number of broadcasts. This counter data is collected for a minutes' duration before resetting the counters to zero to start counting for the next minute. At the end of one minute the counter data is dumped into a shared memory. The requirement is to display market picture counters for all streams, hence counts from all the streams for that message type, placed in separate shared memories are collected and integrated into a single structure before storing it into the database. This is done by the 'Integrate' process. In addition to the counts from all the streams, the integrated data consists of the basket ID, system ID, average of last five days of that scrip and average of last twenty five days of that scrip. The basket and system ID are parameters related to the trading system. This integrated data is again dumped into another shared memory. A different process collects this from the shared memory and pushes it into the database. Data in the same shared memory is also collected by the web server to be streamed to a web based client.

The counters for other message types are implemented in a similar fashion. For example, counter data for time cast will consist of the number of broadcasts received in that minute and the timestamp of the latest received broadcast. The integrate process has to integrate this counter data from different streams before pushing it into the database.

Figure 2, depicts an EOD (End of Day) process, responsible for calculating the averages values that are used by the 'Integrate' processes of Market picture data.

The data flow in this design is linear. Figure 3, depicts the data flow in the system. As we can see from the diagram, there are events that trigger the system to change states, hence this is an event driven system. More so, an implicit invocation of state changes is brought about by the scheduler. In the diagram below the context diagram gives a level zero view of the data flow which is linear. The level one data flow diagram has three states represented by the three vertical data flows. The events are start of a minute and end of that minute. At the beginning of every minute, the ‘Counter’ processes in all the streams start a fresh count of broadcasts and at the end of every minute, the counter data structure is dumped into the shared memory and the counters are reset to zero. The ‘Integrate’ processes can change its state only when all the counter processes of a particular message type have finished dumping into the shared memory. These events are implemented using a couple of shared memory variables. Figure 4, gives the basic logic behind how the triggering occurs using these shared memory variables. This logic can be improvised to provide better performance of the system. The ‘Integrate’ process can be designed to be a blocking process until all the counter processes have completed its work.

The algorithm written for the above design can be referred to from the Appendix. The hooks in the application are provided in the form of shared memory. Any process that wants to tap into the system and fetch data can do it by attaching itself to the shared memory and fetching the required data when it is available.

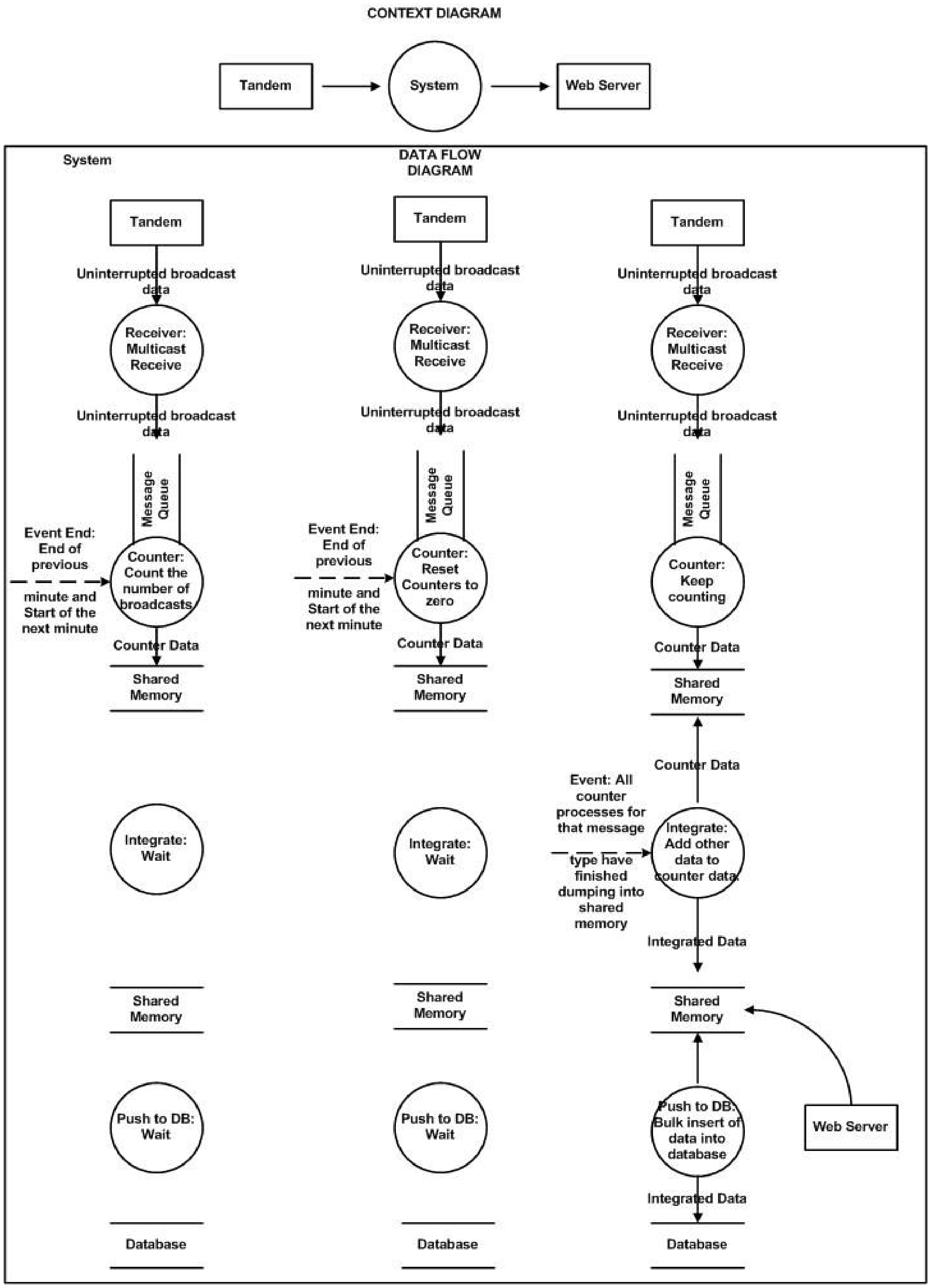
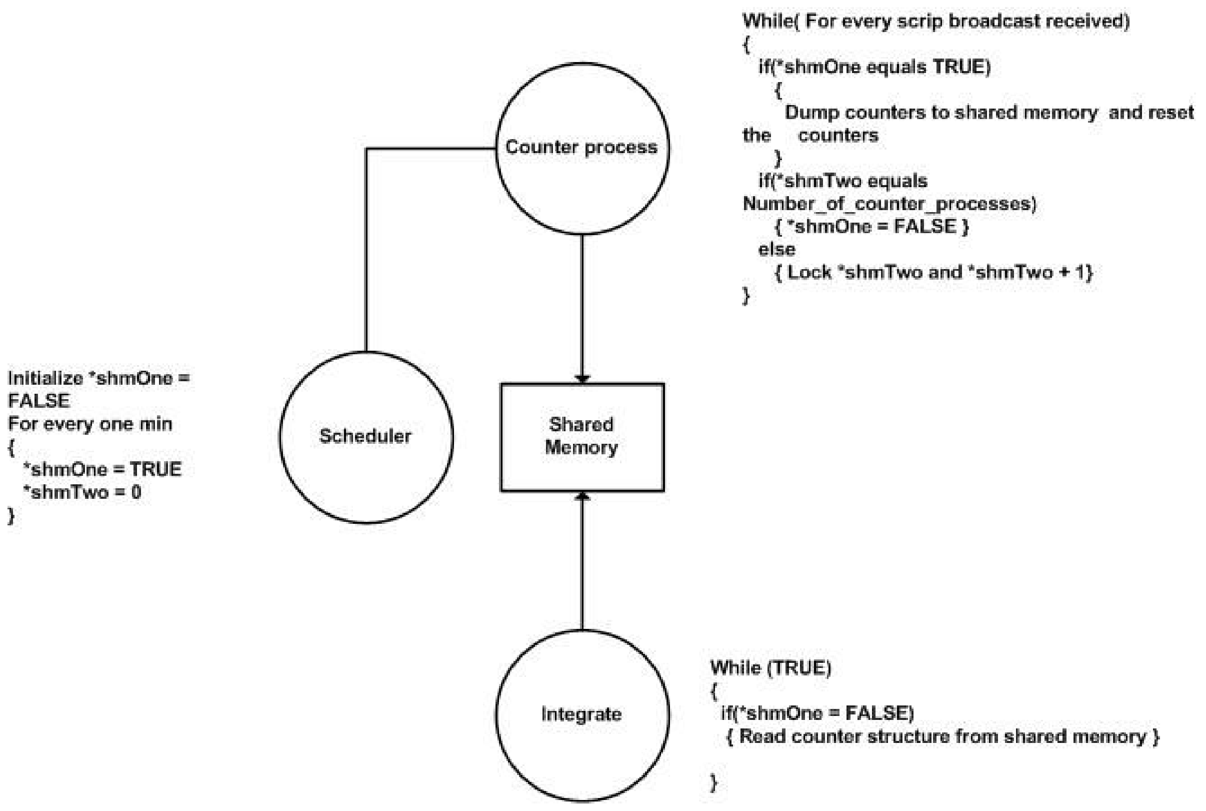

CONTEXT DIAGRAM
Tandem
System
Web Server
System
DATA FLOW DIAGRAM
Tandem
Uninterrupted broadcast data
Receiver: Multicast Receive
Uninterrupted broadcast data
Message Queue
Event End: End of previous minute and Start of the next minute
Counter: Count the number of broadcasts
Counter Data
Shared Memory
Integrate: Wait
Shared Memory
Push to DB: Wait
Database
Tandem
Uninterrupted broadcast data
Receiver: Multicast Receive
Uninterrupted broadcast data
Message Queue
Event End: End of previous minute and Start of the next minute
Counter: Reset Counters to zero
Counter Data
Shared Memory
Integrate: Wait
Shared Memory
Push to DB: Wait
Database
Tandem
Uninterrupted broadcast data
Receiver: Multicast Receive
Uninterrupted broadcast data
Message Queue
Counter: Keep counting
Counter Data
Shared Memory
Counter Data
Event: All counter processes for that message type have finished dumping into shared memory
Integrate: Add other data to counter data
Integrated Data
Shared Memory
Web Server
Push to DB: Bulk insert of data into database
Integrated Data
Database

Figure 3: Data Flow Diagram



```
While( For every scrip broadcast received)
{
  if(*shmOne equals TRUE)
    {
      Dump counters to shared memory and reset
the    counters
    }
  if(*shmTwo equals
Number_of_counter_processes)
    { *shmOne = FALSE }
  else
    { Lock *shmTwo and *shmTwo + 1}
}
```

```
Initialize *shmOne =
FALSE
For every one min
{
  *shmOne = TRUE
  *shmTwo = 0
}
```

```
While (TRUE)
{
 if(*shmOne = FALSE)
  { Read counter structure from shared memory }

}
```

Figure 4: Scheduler Detail

Improvements can be made to the above design of Scheduler and Integrate processes. In case of the Market Picture Data, the Integrate process can make use of Hash functions to map the incoming counter data with the other parameters such as the Basket ID and System Number.

The prototype included implementing the design for the Market picture data for the FCAST stream. Testing involved Unit testing and Integration testing for whatever features were gradually integrated into the prototype.

5. Project Two: Calculation of Volatility Index

Aim: Develop an implied volatility index using the methodology developed by CBOE. VIX is a measure of market's expectation of volatility over the near term. Usually during periods of stress in the markets, the index returns tend to be large and the volatility index tends to rise. The volatility index is computed using the order book of Index options and is expressed as an annualized percentage data point. Investors use the index to hedge and speculate on volatility. "VIX" is a trademark of Chicago Board Options Exchange.

Methodology

$$\sigma^2 = \frac{2}{T}\sum \frac{\Delta K_i}{K_i^2} e^{RT} Q(K_i) - \frac{1}{T}\left[\frac{F}{K_o} - 1\right]$$

Where

σ	:	BSE Implied Vol. Index/100 → BSE Implied Vol. Index = σ *100
T	:	Time to expiration
Ki	:	Strike price of ith out-of-the-money option; a call if Ki > F and a put if Ki < F
∆Ki	:	Is the interval between strike prices - half the distance between strikes on either side of Ki

$$\Delta K_i = \frac{K_{i+1} - K_{i-1}}{2}$$

(Note: ∆Ki for the lowest strike is simply the difference between the lowest strike and the next higher strike. Likewise, ∆Ki for the highest strike is the difference between the highest strike and the next lower strike)

R	:	Risk-free interest rate to expiration

Q (Ki)	:	Midpoint of the bid ask quote for each option contract with strike Ki
F	:	Forward index taken as the latest available price of Sensex futures contract of corresponding expiry.
K0	:	First strike below the forward index level, F

The factors considered in the computation of CBOE VIX in NSE are mentioned below:

1) Time to expiry: The time to expiry is computed in minutes in order to arrive at a level of precision expected by professional traders.
2) Interest Rate: Risk free interest rate is prescribed by a regulatory body and depends on the Inflation rate and other factors.
3) The forward Index level: VIX is computed using out-of-the-money options contracts. out-of-the-money option contracts are identified using forward index level. The forward index level helps in determining the at-the-money (ATM) strike which in turn helps n selecting the option contracts which shall be used for computing. The forward index level is taken as the latest available price of future contract for the respective expiry month.
4) Bid-Ask Quotes: The strike price of options contract available just below the forward index level is taken as the ATM strike. Option Call contracts with strike price above the ATM strike and Option Put contracts with strike price below the ATM strike are identified as out-of-the-money options and best bid and ask quotes of such option contracts are used for computation of VIX.

After identification of the quotes, the variance (volatility squared) is computed separately for near and mid month expiry. The variance is computed by providing weightages to each of options contracts identified for the computation.

Steps followed by CBOE to calculate VIX are as follows:

Step 1: Select the options to be used in the VIX calculation

The selected options are out-of-the–money calls and out-of-the–money puts centered around an at-the-money strike price, K0.

For each contract month: Determine the forward index level, F, by identifying the strike price at which the absolute difference between the call and put prices is smallest. The call and put reflect the average of each option’s bid/ask quotation.

Next determine K0, the strike price immediately below the forward index level, F – for the near and next-term options.

Select out-of-the–money call options with strike prices < K0. Start with the put strike immediately lower than K0 and move to successively lower strike prices.

Next, select out-of-the-money call options with strike prices > K0. Start with the call strike immediately higher than K0 and move to successively higher strike prices. Finally, select both the put and call with strike price K0.

Step 2: Calculate the volatility for both near-term and next-term options

The volatility for both near month and next month options are then calculated by applying the formula for calculating the implied volatility index with time to expiration of T1 and T2, respectively.

$$\sigma_1^2 = \frac{2}{T_1} \sum \frac{\Delta K_i}{K_i^2} e^{RT} Q(K_i) - \frac{1}{T}\left[\frac{F}{K_o} - 1\right]$$ ---------Equation 1

$$\sigma_2^2 = \frac{2}{T_2} \sum \frac{\Delta K_i}{K_i^2} e^{RT} Q(K_i) - \frac{1}{T}\left[\frac{F}{K_o} - 1\right]$$
---------Equation 2

Step 3: σ_1 and σ_2 are interpolated to arrive at a single value with a constant maturity of 30 days to expiration. The formula used for interpolation is as under:

$$\sigma = \sqrt{\left[\left\{T_1 \sigma_1^2 \left[\frac{N_{T_2} - N_{30}}{N_{T_2} - N_{T_1}}\right]\right\} + \left\{T_2 \sigma_2^2 \left[\frac{N_{30} - N_{T_2}}{N_{T_2} - N_{T_1}}\right]\right\}\right] \mathrm{X} \frac{N_{365}}{N_{30}}}$$

6. Conclusion

The primary goal of this report is to identify the project requirements and its implementation. The project work on the Linux machine can be extended to be used as a data replica of Tandem machine. This can be done by extracting other sources of data from Tandem, besides the broadcast information. The Web server can be extended to provide services and return queries regarding Tandem logs etc. Significant amount of time was spent on understanding the methodology of calculation of volatility index. In the course of its implementation, I have gained domain knowledge and improved my technical skills.

APPENDIX

Pseudo code for prototype of Project One: Broadcast Status Reporting and Analysis System

Input: Market Picture Broadcast

Output: Scrip wise count of the Broadcast

The pseudo code below is for a prototype of Broadcast Status Reporting and Analysis System. It restricts itself to handling the Market Picture Broadcast. It receives broadcast from the trading machine, decompresses it, and counts the number of broadcasts for one minute.

```
Receiver
   1. START Receiver
   2. Create UDP Socket
   3. Join the multicast group
   4. Receive broadcast packets
   5. Create another process
   6. IF child process
            Execute program that decompresses Market Picture Data
   7. IF parent process
            WHILE True
                  Send the broadcast packets to a loosely coupled message
            queue
            END WHILE
   8. END Receiver
Decompress
   1. START Decompress
   2. Open File for error logging.
   3. Attach to the shared memory variable created by Scheduler
   Attach to *SharedVariable
4. Initialize *SharedVariable to zero
5. Allocate memory to the data structure that collects Counter data
6. Create shared memory *Data to dump the counter structure
7. Create shared memory *NumberOfElements to send the number of
   elements of Counter data structure
8. Get the current time of the day
9. WHILE There are packets in message queue
         Decompress the Market picture data
         IF *SharedVariable is EQUAL to False THEN
               Initialize *NumberOfElements
               Dump counter data to shared memory
               Reset the counter structure
         ENDIF
         IF *SharedVariable is EQUAL to NumberOfCounterProcesses THEN
               Initialize *SharedVariable to zero
         ENDIF
         ELSE Increment *SharedVariable by one
         Search for the scrip code in the existing structure
         IF found THEN Increment its count by one
         ELSE make a new entry in the counter structure
10.      END WHILE
11.      END Decompress
```

Integrate

```
1. START Integrate
2. Attach to the shared memory *Data created by the Receiver process to
   retrieve the counter structure
3.  Attach to the shared memory variable created by Scheduler
```

```
   Attach to *SharedVariable
4. Allocate memory to retrieve the counter structure
5. WHILE True
         IF *SharedVariable EQUAL to zero
               Retrieve from the number of elements in counter
   structure
               Retrieve the counter structure from the shared memory
               Attach the basket ID and System no. for the scrip code
         ENDIF
6. ENDWHILE
7. END Integrate
```

Scheduler

```
1. START Scheduler
2. Create shared memory variable *SharedVariable
3. *SharedVariable EQUAL to zero
4. Change the value of *SharedVariable to True every one minute
5. END Scheduler
```

References

[1] http://www.cmcltd.com/case_studies/bolt.shtml

[2] http://www.itaniumsolutions.org/ , Bombay Stock Exchange Case Study.

[3] White paper on CBOE Volatility Index

[4] GCC, The Complete Reference Guide

[5] A Tale of Two Indices, Peter Carr and Liuren Wu (The Journal of Derivatives)

[6] BOLT Help file

[7] http://www.investopedia.com/

Glossary

1. BOLT: BSE Online Trading System
2. Streams of Broadcast: Broadcast data at different rates. The rates at which they are broadcast depend on the purpose for which it is used. The broadcast sent to media such as television are slower when compared to broadcasts sent to traders who use Algorithmic trading or low latency trading.
3. FCAST: It is a type of stream
4. Message types in Broadcast: Various types of messages that are broadcast such as Market picture data, Sensex and other indices, Tandem time etc.,
5. Market Picture Data: As the name suggests, it gives a picture of market in terms of variables such as Last Traded Price, Last Traded Quantity, best sell price, best sell quantity etc.
6. Scrip: Scrips are the names of companies/ a certificate which represents the commodity being is traded.
7. Basket ID and System Number: Terms are used in describing components in the trading system.
8. Volatility: Fluctuation in prices
9. Hooks: A place in code that allows you to tap into the module or data.
10. Bid-Ask Quote: A two-way price comprising a bid, or the price at which a dealer is willing to buy, and an ask (or offer) at which a dealer is willing to sell. The bid, by definition, is always below the ask and is always the first quoted price.
11. CBOE: Chicago Board of Exchange.
12. Out-of-the-money: For a call, when an option's strike price is higher than the market price of the underlying asset. For a put, when the strike price is below the market price of the underlying asset.