IPRO 313 : Ultra-High-Speed Market Data Ticker System

Introduction

Main objective

Research and design a solution for our sponsor which sustains an optimal throughput of three million price quotes per second and minimizes latency while maintaining specific constraints

Approach

+ Research low latency discussions and reports
+ Explore competitors' solutions
+ Develop a functioning ticker plant system
+ Determine hardware requirements
+ Benchmarks & Prototype

Townsend Analytics

A direct-access trading-system vendor, provides a real time database management soultion but according to predicted market data for the near future, their solution won't stand ground for too long.

Problem Addressed

With the dawn of every new year, the speed of business is ever increasing.

What used to be performed in months, days or minutes, is now being done in miliseconds; and even that is not fast enough!

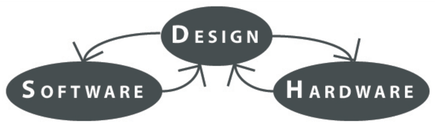
Businesses will have to transition to real time data management to keep a competative drive on the market.

+ Within one second, between 5,000 and 50,000 are expected to occur on a real time database system.
+ Within a year, terabytes of data are expected to occur on a real time database system

In order for a business to upgrade, a unique system has to be prepared to handle:

+ Large volumes of data
+ Very frequent requests for data
+ Increased rate of change of data

Methodology



Design Team:
Management of IPRO deliverables
Communicating with Townsend
Assigning and dividing tasks
Researching new technology
Analyzing market value

Philip Pannenko, CS – TL
Devaraj Ramsamy, BA
Kenneth Buddell, BA

Hardware Team:
Analyzing hardware performance
Defining components
Researching competitors' solutions

Mike Lenzen, MATH – STL
Jong Min Lim, ECE
Yunseok Song, ECE

Software Team:
Researching algorithm performances
Preparing benchmarks of algorithms
Optimizing algorithms

Jong-Yon Kim, B.A. – STL
Jesus Allan C. Tugade, C.S.
Jong Su Yoon, CS
Usman Jafarey, CS
Young Cho, MATH

Example

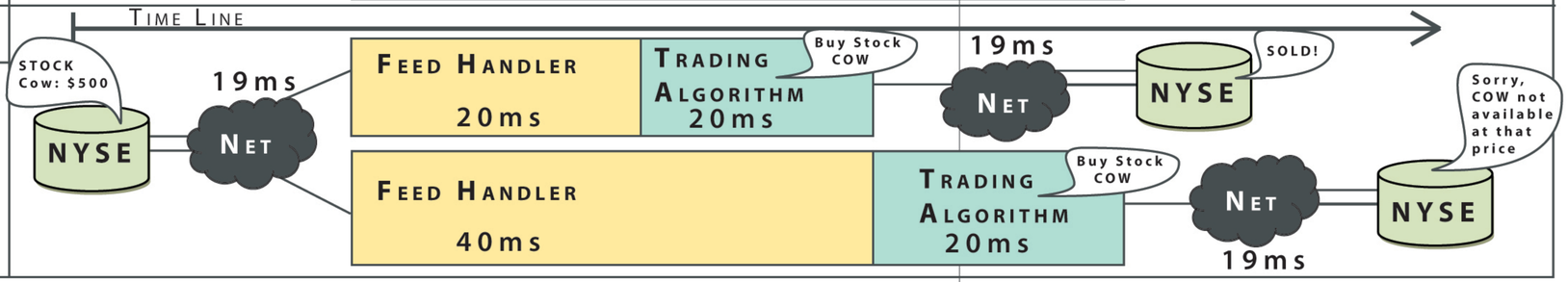
Assumptions

+ location of the feed handlers are at Stanford, CA
+ algorithmic trading system identical
+ New York Stock Exchange (NYSE) is located in New York City

Why 22ms travel time?

+ Speed of Light 300x10^6
+ Distance between Stanford & Boston 3858km
+ Speed of Light in fastest fibre cable: 200x10^6

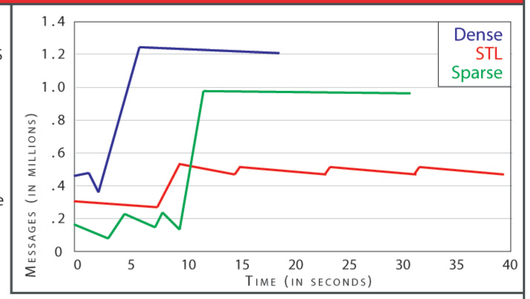
Total: 19.3ms



Unit Testing - Benchmarks & Results

Testing different hashing algorithms

+ Performance: Dense > Sparse > STL
+ Memory usage: Sparse > STL > Dense
+ STL: Table size and the character length has impact on the performance
+ Sparse, Dense: Character length has impact on the performance but less impact from the table size



Performance can be improved by optimizing the algorithm of the container and the hash function

System testing - Benchmarks & Results

Performance testing of various hashing algorithms on the system

Bandwidth of our network equiptment limited to 100mbps

+ Need to find ways to increase bandwidth limit

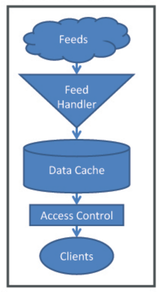
Hashing Algorithm	Data Generator (msg/sec)		Last Value Cache (msg/sec)	
	Case 1	Case 2	Case 1	Case 2
STL	82,000	120,000	82,000	109,00
Sparse	82,000	120,000	82,000	109,00
Dense	82,000	120,000	82,000	109,00

Hardware Requirements

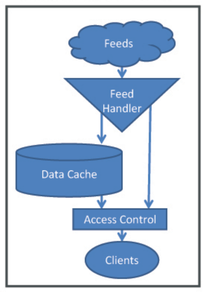
Defining Terms

+ Feeds: Options, Stocks, Futures
+ Feed Handlers: Translates incoming data to an internal format
+ Data Cache: Stores old files
+ Access Control: Handles client connections and permissions

General Architecture



Proposed Architechture



Pros of proposed design

+ Messages will be smaller requireing less bandwith
+ Missing data will not be filled in to create a complete record thus bypassing the Data Cache en route to the client

Cons of proposed design

+ Places the burden of applying the updates on the client-side software

Competitor Solutions

Company	S/W	Network	H/W	Consolidator	Direct Feed	Application
TA				●		
Reuters		●		●		●
Wombat	●				●	
Bloomberg		●		●		●
RTI		●		●		
Exegy			●		●	

Future Work

Hardware

+ Work around network bandwidth limit
+ Implement proposed architecture
+ Running tests on Playstation 3 cell processors

Software

+ Introducing efficient hash function such as SuperfastHash and CRC32
+ Implementing threads and parralel processing into code