Overview

Elaine Wah
ewah@umich.edu

Updated: October 14, 2013

1 Introduction

The HFT simulation system employs agent-based modeling and discrete-event simulation to model communication latencies and current U.S. securities regulations.

2 Overview

The simulation system consists of:

Markets: Two types of markets are implemented in the system, a continuous double auction (CDA) market and a call market (which matches orders at regular, fixed intervals).

Agents: There are two general types of agents in the system, which are distinguished by whether or not they have fast access to more than one market.

Background agents: These agents have fast (potentially undelayed) access to only a single market (which is specified at agent creation as their primary market). See ZIAgent.

High-frequency trading agents: These agents have access to multiple markets (usually every market), which models the direct feeds that many HFTs have to exchanges. See LAAgent.

3 Discrete-Event Simulation

In our system, we employ *discrete-event simulation*, a paradigm that allows the precise specification of event occurrences. It is particularly effective for modeling current U.S. securities regulation, specifically Regulation NMS—which led to the creation of the Security Information Processor (SIP) and which mandates the routing of orders for best execution. Components of the simulation system include:

Entity: Objects present in the simulation system, e.g., traders, markets, and the SIP, that perform actions that effects on other entities.

Activity: Actions that entities can execute.

Activity	Description
AgentArrival	agent arrives in a market (or markets)
AgentStrategy	agent executes its trading strategy
Clear	market clears any matching orders
Liquidate	agent liquidates any net position
ProcessQuote	IP (or SIP) updates its best market quotes
SendToIP	market sends a new quote to an IP (SIP)
SubmitOrder	agent submits a single-point order to a market
SubmitNMSOrder	agent submits single-point order, routed for best execution
WithdrawOrder	agent withdraws a specific order from a market

Table 1: List of activities in the simulation system.

Event: A sequence of activities happening at the same time. Maintains the order in which they occur.

Event Queue: Queue ordered by activity time that executes activities in the "proper" order, sequentially until empty. Multiple activities may occur during the same time step, and in most circumstances the order they are executed in is psedo-random according to the random number generator of the simulation. The random nature is meant to simulate the slight timing differences that occur in real life i.e. nothing actually occurs at the same time.

To summarize, an *event* consists of a sequence of *activities* that are to be executed by various *entities* (traders, markets, and the SIP).

3.1 Activities

Each activity has a timestamp and each is associated with at least one entity present in the simulation system. Note that activities are chained (the next immediate activity is inserted at the end of the current one). See Table 1 for a partial list of activities within the system.

3.2 Example

To control the latency of the SIP as well as general market access, we specify two activities: SendToIP and ProcessQuote. The SendToIP activity is inserted when a market updates its quote at time t. Once the information processor (IP) gets the information it inserts a ProcessQuote activity to execute at time $t + \delta$ in the future to account for the delay caused by processing the information. When ProcessQuote is executed, the IP updates its stored information on the best market quotes. When agent's query the SIP for market information they will only get the most recent information that it's processed after $t + \delta$, not all of the market quotes at the current time.

4 Simulation Specification File

The parameters in the specification file are those that can be adjusted in simulations. Below is the "default" `simulation_spec.json` file found in the docs folder:

```
{
    "assignment": {
    },
    "configuration": {
        "modelName" : "mySimulation",
        "modelNum" : "1",
        "simLength": "15000",
        "presets" : "TWOMARKETLA",
        "CDA" : "num_0",
        "CALL" : "num_0",
        "BASICMM": "
           num_0_sleepTime_200_numRungs_10_rungSize_1000",
        "ZI": "num_250_bidRangeMin_0_bidRangeMax_5000",
        "ZIR": "num_0_maxqty_5",
        "ZIP": "num_0",
        "AA": "num_0",
        "tickSize": "1",
        "nbboLatency": "100",
        "arrivalRate": "0.075",
        "reentryRate": "0.005",
        "meanValue": "100000",
        "kappa": "0.05",
        "shockVar": "150000000",
        "privateValueVar": "100000000"
    }
}
```

An explanation of select parameters follows:

`assignment:` Assignment of strategies to players. Relevant only for EGTA

`configuration:` Specify simulation configuration.

`modelName:` The name of the model. This is only used in the output file.

`modelNum:` The model number. This is output in the log file and can be used to merge several log files.

`simLength:` Length of simulation in time ticks (usually milliseconds).

`presets:` This is a way to get easy access to "standard" market configurations. Currently there are four options:

`CENTRALCDA:` A single CDA Market.

`CENTRALCALL`: A single CALL Market that clears at the `nbboLatency`.

`TWOMARKET`: Two CDA markets and no latency arbitrageur.

`TWOMARKETLA`: Two CDA markets and a latency arbitrageur.

Markets: The next set of options allow manually specifying market configurations if a preset is not used. Currently there are two markets you can set configurations for and those are `CDA` and `CALL`. The string after is a configuration string which takes the format `<key1>_<value1>_<key>_<value2>_...`, and can contain several comma separated configurations. The following entry would create three CALL markets. Two will clear every tenth of a second, and one will clear every second:

```
"CALL" : "num_2_clearFreq_100,num_1_clearFreq_1000"
```

Agents: The next set of options allow manually specifying agent configurations (NOTE: LA configurations can only be set if a `preset` is not being used). The current agents you can create are `BASICMM`, `ZI`, `ZIR`, `ZIP`, `AA`, and `LA`. Each agent takes a configuration string identical in style to a market only agents will use different parameters.

`tickSize`: Tick size of pricing. Prices are integers, so the smallest tick size is 1.

`nbboLatency`: The latency of the nbbo in time ticks (usually milliseconds).

5 Running a Simulation

1. To run a basic simulation, create a directory to store all of your simulations (e.g. `simulations`) and then create a directory for this specific simulation inside it:

   ```
   pwd # should print your hft folder
   mkdir simulations
   mkdir simulations/test
   ```

2. Then copy the default `simulation_spec.json` file into the folder you just created:

   ```
   cp docs/simulation_spec.json simulations/test/
   ```

3. Make any tweaks to the specification you want using your favorite text editor.

4. Use the following command to run your simulation:

   ```
   ./run_hft.sh simulations/test <number of simulations to run>
   ```

   For example:

   ```
   ./run_hft.sh simulations/test 100
   ```

5. All simulations should be saved within subfolders in the `simulations/test` (or other) directory!