# Высокопроизводительный генератор нагрузки для тестирования систем автоматизированной торговли

Дмитрий Гурьев[1], Мария Гай[1], Иосиф Иткин[2], Александр Терентьев[3]

[1] ООО «Инновационные Трейдинговые Системы», Россия, 115088, г. Москва, 2-й Южнопортовый проезд, 20А/4
Dmitry.Guriev@exactpro.com, Maria.Gai@exactpro.com
[2] Exactpro Systems LLC, 4040 Civic Center Drive, Suite 200, San Rafael, CA 94903, USA
Iosif.Itkin@exactpro.com
[3] Саратовский государственный технический университет имени Гагарина Ю.А., Россия, 410054, Саратов, ул. Политехническая, 77
a_a_terentyev@mail.ru

**Аннотация.** В связи с существенным ростом числа торговых заявок, вызванным развитием высокочастотной торговли, ставится задача тестирования биржевых и брокерских систем в режимах, приближенных к реальным. Для обеспечения качества высоконагруженных трейдинговых систем высокой доступности применяются специализированные инструменты тестирования. Основные требования к таким инструментам - это способность создавать высокие реалистичные нагрузки, используя ограниченную аппаратную базу. В данной статье описывается разработанный генератор нагрузки для тестирования систем автоматизированной торговли. Представлен подход, обеспечивающий высокую производительность.

***Ключевые слова:*** нагрузочное тестирование, высокочастотная торговля (HFT), инструменты для тестирования.

## 1 Введение

Высокочастотная электронная торговля финансовыми инструментами (HFT), позволяющая минимизировать временные задержки при совершении сделок, выросла в последние годы и составляет в настоящее время около 30% от всего объёма торговли акциями в Великобритании и, возможно, более 60% от всего объёма торговли акциями в США [1]. В результате этого брокерские и биржевые системы испытывают всё большую нагрузку от потока транзакций, генерируемого системами автоматизированной торговли. Операторы трейдинговых платформ, регулирующие органы и участники торгов должны быть уверены в надёжности программного обеспечения и инфраструктуры торговых площадок [2] в условиях постоянно растущих нагрузок.

В ходе разработки программного обеспечения для выявления максимальной пропускной способности, возможных узких мест и определения проблемных

элементов системы применяют методы нагрузочного тестирования. Под нагрузочным тестированием понимается процесс отправки системе большого количества запросов, проверка своевременности и корректности полученных от неё откликов, а также проверка внутреннего состояния системы.

В настоящее время для нагрузочного тестирования программного обеспечения широко используются различные коммерческие и свободно распространяемые генераторы нагрузки. В качестве примера можно привести следующие продукты: Apache JMeter, HP Load Runner, IBM Rational Performance Tester, Borland Silk Performer и другие [3-6]. Основной концепцией, используемой в этих продуктах, является создание множества виртуальных пользователей, эмулирующих поведение реальных пользователей для моделирования условий, при которых программа/система будет функционировать в реальности. При имитации и поддержке соединения с большим количеством пользователей и высокой нагрузке у инструментов тестирования могут возникать ограничения производительности, разобранные во второй части данной статьи.

В компании Exactpro Systems LLC разработан инструмент для тестирования высоконагруженных трейдинговых систем, обладающий необходимой производительностью и использованный на практике при проверке некоторых из крупнейших биржевых технологических инфраструктур в Западной Европе [7; 8]. Разработанный инструмент поддерживает протоколы: FIX (все версии), ITCH, LSE Native, SOLA SAIL & HSVF, HTTP, SOAP, а также различные бинарные протоколы трейдинговых систем. Многопротокольность является одной из архитектурных особенностей разработанного инструмента для тестирования, вследствие чего добавление новых протоколов и новых версий уже поддерживаемых протоколов представляет собой относительно малозатратную задачу. В третьей части статьи рассмотрены особенности подготовки данных для нагрузочного тестирования. В четвёртой части представлены возможности по настройке инструмента, включая задание профиля нагрузки.

## 2 Оптимизация процесса создания нагрузки

Общее представление о процессе создания нагрузки даётся в ряде работ, в частности в [9]. Генераторы нагрузки делят на основанные на измерениях (measurement-based) и основанные на моделях (model-based) [10]. Основанные на измерениях генераторы удобны для нахождения пропускной способности тестируемой системы и построения зависимостей времён отклика от нагрузки. Генераторы нагрузки, основанные на моделях, направлены на симуляцию распределения входных данных, максимально приближенную к реальному промышленному использованию системы. В разработанном авторами высокопроизводительном генераторе нагрузки поддерживаются оба описанных варианта. Данные модели используются при создании конфигурационных файлов перед запуском генератора. Таким образом, инструмент может не

тратить ресурсы на обработку информации, относящейся к модели непосредственно во время выполнения тестов.

Генераторы нагрузки подразделяются на работающие по принципам закрытого цикла (closed-cycle) и открытого цикла (open-cycle) [11]. После посылки сообщения исполняемый поток в генераторе закрытого типа дожидается ответа от системы, прежде чем приступить к посылке следующих запросов. Генератор открытого цикла продолжает посылку сообщений не дожидаясь ответа от тестируемой системы. Большинство инструментов для тестирования веб-приложений являются генераторами закрытого цикла. Это связано с использованием концепции виртуальных пользователей, каждый из которых последовательно выполняет шаги определённого сценария. Генератор закрытого цикла требует существенно большего количества потоков исполнения и переключений между ними в сравнении с генератором открытого цикла. В генераторах закрытого цикла часто обработка исходящих из системы ответов происходит в том же потоке, что и посылка входящих сообщений, дополнительно снижая производительность инструмента и иногда даже влияя на его аккуратность. Таким образом, генераторы открытого цикла требуют меньшей аппаратной базы для создания требуемого уровня нагрузки. Они также не требуют порождения лишних потоков и их синронизации для производства фиксированного уровня нагрузки. Представленный в статье инструмент работает как генератор открытого цикла.

При разработке инструмента для нагрузочного тестирования рассматривался вопрос о необходимости привязки исполняемых потоков к ядрам процессора для сглаживания распределения входящих в систему сообщений по времени [12]. Авторы пришли к выводу, что миллисекундное разрешение системных таймеров, присутствующее в большинстве современных Linux-системах, достаточно для создания реалистичной трейдинговой нагрузки, а отсутствие привязки к ядрам процессора освобождает некоторое дополнительное количество аппаратных ресурсов на генераторе нагрузки, позволяя централизованно запускать оптимальное количество потоков. Разработанный инструмент использует центральный контроллер, позволяя заранее задать в конфигурации количество и состав протокольных соединений, которые будут работать в каждом потоке. Наличие центрального контроллера также позволяет выдавать команду на выполнение скоординированных действий всеми потоками. Например, одновременный старт потока сообщений или одновременное отключение установленных соединений.

При тестировании трейдинговой системы генератор нагрузки заменяет большое количество систем автоматизированной торговли, использующих множество серверов. Однако аппаратная база, доступная для размещения инструментов тестирования, всегда ограничена соображениями экономии [13]. В условиях продолжающейся финансовой нестабильности даже крупнейшие финансовые институты работают в режиме максимально возможной оптимизации затрат. Требуется существенно облегчить процесс создания исходящих сообщений. Для оптимизации процесса создания нагрузки необходимо до выполнения теста заготовить шаблоны сообщений, сократив процессорное время на серверах, содержащих инструменты для тестирования. Сходные соображения заложены в генератор нагрузки, созданный

разработчиками крупнейшего российского поисковика «Яндекс». Инструмент для тестирования с открытым кодом «Яндекс.Танк» предназначен для генерации огромных объёмов сообщений по протоколу HTTP [14]. Высокая производительность «Яндекс.Танк» достигается концентрацией нагрузки в одной сессии и одном потоке, а также использованием заготовленного файла со статическими запросами. Генераторы нагрузки для трейдинговых систем не могут использовать статические данные и обладают рядом других ограничений, рассматриваемых в следующей части.

## 3 Особенности нагрузочного тестирования торговых систем

В [15] разобраны основные требования к моделированию нагрузки для систем высокочастотной торговли. Логика работы таких систем существенно затрудняет использование статичных, ранее записанных или предопределенных данных. В этой секции рассматриваются некоторые из особенностей создания нагрузки и подготовки входных данных для тестирования трейдинговых систем.

### 3.1. Создание сообщений из шаблонов

Вследствие того, что торговля не анонимна, для поддержания сессии сервер должен получать имена существующих пользователей, корректные порядковые номера сообщений, а также время отправки каждого сообщения. Наш анализ показал, что построение сообщений непосредственно перед отправкой с использованием словарей обходится очень дорого с точки зрения используемых системных ресурсов. Поэтому было решено использовать заготовки, в которых порядок полей и набор ключевых значений заданы до начала теста. Например, в FIX-сообщении NewOrderSingle перед моментом отправки будут изменяться всего несколько параметров, которые должны быть уникальны (например, ClOrdID(11) – номер клиентской заявки) и зависеть от текущего времени (ExpireTime(126) – время истечения срока заявки, ExpireDate(432) – день истечения срока заявки). Остальные параметры новой заявки не изменяются. Для поддержания сессии изменяются служебные параметры:
*BodyLength*(9) – длина сообщения;
*SenderCompID*(49) – название компании, которая отправила заявку;
*TargetCompID*(56) – название компании, которой была отправлена заявка;
*MsgSeqNum*(34) – уникальный номер сообщения;
*SendingTime*(52) – текущее время;
*CheckSum*(10) – проверочное число.

В сообщения OrderCancelReplaceRequest и OrderCancelRequest подставляются все необходимые параметры, взятые из сообщения NewOrderSingle, такие как:
*OrderID*(37) – идентификатор заявки;
*Price*(44) – цена заявки;
*Quantity*(53) – размер заявки;
*Side*(54) – сторона заявки (покупка или продажа);
*Symbol*(55) – символическое обозначение инструмента.

Предположение, что все значения параметров верны, даёт возможность существенной экономии времени на проверку значений полей и правильности их последовательности в сообщении.

### 3.2. Воспроизведение ранее записанных данных

Отправка заранее подготовленных записанных данных приводит к искажению теста. При построении данных для тестирования необходимо придерживаться той же пропорции торговых событий, которую мы наблюдаем в реальной жизни. Анализ реальных торгов показывает, что на одну сделку может приходиться более 20 изменений заявок.

Записанные данные представляют собой совокупность новых заявок, их изменения и отмены. Так как заявки отправляются из разных потоков на одну и ту же книгу заявок, они могут приходить на рынок в порядке, отличном от того, каким он был при записи. Даже если время появления заявки на рынке будет отличаться незначительно, это может привести к нежелательным последствиям. Например, к моменту получения рынком заявки, другие заявки могут располагаться в книге заявок в порядке, отличном от того, который был при записи, и они могут проторговаться в другой последовательности. Из этого следует, что последующие изменения или отмена заявки будут невозможны, так как заявка проторговалась и была удалена из книги заявок. Такая ситуация влечёт за собой сбой в последовательности сценария тестирования, и в дальнейшем будет получен сценарий, отличный от записанного. Таким образом, на рынке будут происходить события, отличные от тех, что были при записи сценария. Например, рынок будет отправлять значительно больше отказов на изменение или отмену заявок. При этом отличия от первоначального сценария могут накапливаться, и реальная пропорция торговых событий может сильно отличаться от исходной.

### 3.3. Использование детерминированных сценариев

Другая возможность организовать тест заключается в подготовке двух групп заявок: к первой группе относятся заявки, о которых заранее известно, что они будут участвовать в сделках (активные заявки); вторая группа состоит из заявок, которые не должны торговаться (пассивные заявки). Однако в этом случае следует учитывать возможные осложнения со стороны системы мониторинга и контроля рынков (англ. *Market Surveillance System*). Одна из функций этого компонента заключается в том, что он должен в режиме реального времени отслеживать «договорные» заявки, т.е. как раз те заявки, которые должны проторговаться при проведении тестирования, и сообщать о таких событиях службе, следящей за манипулированием ценами на рынке. Очевидно, что такой вариант не подходит для создания сценария тестирования. В связи с этим, нашими специалистами был создан рандомизированный генератор нагрузки с использованием механизма обратной связи. Рандомизация используется для генерации новой цены при отправке изменения заявки. Для генерации используются три параметра: начальная цена, диапазон изменения цены и шаг изменения цены.

Начальная цена задается в массиве данных для каждого инструмента и для каждой стороны (покупки или продажи). Начальные цены должны удовлетворять следующим условиям:
- цена покупки должна быть меньше цены продажи;
- разница начальных цен продажи и покупки должна составлять около 2-3% от цены открытия рынка.

Диапазон изменения цены покупки и продажи выбирается таким образом, чтобы цены встречных предложений пересекались, обеспечивая торговлю, и чтобы цены предложений не выходили за 10%-й барьер - условие, при несоблюдении которого возможна остановка торговли на данном инструменте или на целом сегменте инструментов. Эти параметры позволяют гибко подбирать желаемое среднее соотношение количества сделок к количеству изменений, приходящихся в среднем на одну заявку. Чем меньше область пересечения цен покупки и продажи, тем меньше будет сделок, и заявка в среднем будет изменяться большее количество раз. Надо заметить, что это отношение нелинейно зависит от интервала пересечения цен. Поскольку цены задаются для каждого инструмента в отдельности, становится возможным настроить разное количество сделок на разных инструментах в одном тесте.

Шаг изменения цены задаётся исходя из конфигурации инструмента. Например, одни инструменты могут торговаться с шагом цены 0,05, другие - с шагом цены 0,10.

Механизм обратной связи необходим, чтобы отслеживать состояние заявки на рынке и обеспечивать возможность изменения или отмены заявки. Заявка может иметь следующие состояния:
*New* – заявка еще не приняла участие в торговле;
*PartFilled* – заявка выполнена частично;
*Filled* – заявка выполнена полностью;
*Canceled* – заявка отменена клиентом;
*Expired* – заявка с истекшим сроком действия;
*Rejected* – заявка отклонена биржей.

Только заявки в состояниях «New» и «PartFilled» могут участвовать в торговле. Как только заявка переходит в другое состояние, она перестает быть интересной, и информация о ней тут же удаляется.

## 4 Пример настройки разработанного генератора нагрузки

В этой части статьи подробнее остановимся на нескольких вариантах настройки разработанного авторами генератора нагрузки для тестирования трейдинговых систем на основе протокола FIX [16].

Настройка теста осуществляется посредством 4-х типов конфигурационных файлов, которые содержат:

- настройку параметров нагрузки;
- конфигурацию сессий;
- заготовку сообщений;
- распределение по сообщениям.

### 4.1. Формат файла параметров нагрузки

Для настройки параметров нагрузки используется файл, имеющий следующий формат:

```
#Конфигурационный файл с настройками сессий:
CONNECTIONS_CONFIG = fixConnections.cfg
#Указание используемых сессий из файла с сессиями:
CONNECTIONS_RANGE = 1-3, 5, 7-
#Файл с заготовками сообщений:
MESSAGE_TEMPLATES = fixMessageTemplates.dat
#Файл с распределением по сообщениям:
MESSAGE_RATES = messageRates.cfg
#Последовательность действий до начала теста:
INIT_CONFIG = connect(100ms), logon(3s)
#Конфигурация нагрузки:
LOAD_CONFIG = const(1000,5m)
#Задается постоянная нагрузка 1000 сообщений в секунду
#на протяжении 5-и минут.
#Количество повторений нагрузочного сценария, заданного
#параметром LOAD_CONFIG:
NUMBER_REPETITIONS = 10
#Последовательность действий после окончания теста:
SHUTDOWN_CONFIG = logout(1s), disconnect(10ms)
#Последовательность действий при внезапном обрыве
#соединения
ON_RECONNECT_CONFIG = connect(10ms), logon(3s)
#Флаг на выполнение действий, указанных в
#ON_RECONNECT_CONFIG при обрыве соединения:
HOLD_CONNECTION = 1
#Если значение = 0, действия в ON_RECONNECT_CONFIG не
#выполняются, и соединение не восстанавливается.
#Время задержки между авторизацией сессий в миллисекундах
LOGON_INTERVAL = 1000
```

Поскольку у клиентов есть возможность использовать собственные программы для торговли, существует вероятность того, что по какой-то причине, например, в случае отправки торговой системой большого объёма сообщений, клиентские программы не смогут вовремя прочитать эти данные. Это может негативным образом повлиять на поведение торговой системы. Для тестирования этой возможности в разработанном программном продукте существует специальное ограничение по количеству читаемых данных в секунду для эмуляции медленных клиентов.

### 4.2. Формат файла конфигурации сессий

Настройки параметров соединений задаются в файле следующего формата:
В секции COMMON задаются общие параметры соединений:

```
[COMMON]
HOST = 10.10.10.10
PORT = 5555
TARGET_COMP_ID = FGW
```

В секции [FIX] задаются уникальные параметры отдельного соединения:

```
[FIX]
SENDER_COMP_ID = LOAD_1
RESET_SEQ_NUM_AFTER_LOGOUT = 0
```

```
PARTY_ID = LOAD_1
```

Секция [FIX] должна повторяться столько раз, сколько предполагается использовать соединений. При этом соединения, которым предстоит участвовать в тесте, задаются параметром CONNECTIONS_RANGE в файле с параметрами нагрузки.

### 4.3. Формат файла заготовок сообщений

Файл содержит массив именованных сообщений-заготовок. Эти заготовки имеют правильные формат и последовательность полей в сообщениях. Некоторые поля будут заменяться корректными данными непосредственно перед отправкой сообщения.

```
Logon
8=FIXT.1.1|9=61|35=A|34=1|49=SenderCompID|56=TargetCompID|98=0|108=3600|
554=password|1137=9|10=135|EOM
NewOrderBuy
8=FIXT.1.1|9=199|35=D|34=1|49=SenderCompID|56=TargetCompID|1=CLIENT|11=C
lOrdID|38=200|40=2|44=9.8|54=1|55=Symbol|59=6|60=20130728-
13:34:03.194|432=20130730|528=P|581=3|1138=60000|9303=I|453=1|448=PartyI
D|447=D|452=76|10=047|EOM
NewOrderSell
8=FIXT.1.1|9=199|35=D|34=1|49=SenderCompID|56=TargetCompID|1=CLIENT|11=C
lOrdID|38=150|40=2|44=10.2|54=2|55=Symbol|59=6|60=20130728-
13:34:03.194|432=20130730|528=P|581=3|1138=60000|9303=I|453=1|448=PartyI
D|447=D|452=76|10=047|EOM
Cancel
8=FIXT.1.1|9=134|35=F|34=1|49=SenderCompID|56=TargetCompID|11=ClOrdID|41
=OrigClOrdID|54=1|55=Symbol|60=20130728-
13:34:03.178|9303=I|453=1|448=PartyID|447=D|452=76|10=050|EOM
Replace
8=FIXT.1.1|9=179|35=G|34=1|49=SenderCompID|56=TargetCompID|1=CLIENT|11=C
lOrdID|38=180|40=2|41=OrigClOrdID|54=1|55=Symbol|60=20130728-
13:34:03.178|432=20130730|1138=70000|9303=I|453=1|448=PartyID|447=D|452=
76|10=077|EOM
Logout
8=FIXT.1.1|9=29|35=5|34=111|49=SenderCompID|56=TargetCompID|10=249|EOM
```

### 4.4. Формат файла распределения нагрузки по сообщениям

Файл содержит соотношение количества сообщений, указанных в долях для каждого типа сообщения:

```
NewOrderBuy = 15
Replace = 50
Cancel = 5
```

В зависимости от настройки, MESSAGE_SELECTION_ORDER = sequential или random, сообщения будут выбираться или последовательно, или случайным образом.

### 4.5. Упрощенная схема работы алгоритма

На рисунке 1 приведена блок-схема алгоритма по выбору и отправке сообщений. На первом этапе происходит чтение входящих данных. В случае, если данные есть, происходит анализ полученных ответов и изменение

состояния заявок или их параметров. После этого случайным методом или последовательным перебором выбирается новое сообщение. Если был выбран приказ на создание новой заявки, его параметры сохраняются в памяти для дальнейшего использования. В случае выбора сообщения для изменения или отмены заявки, выбирается заявка, для которой нет неотвеченного запроса. Её параметры подставляются в сообщение и посылаются в торговую систему. После этого вычисляется время, прошедшее с начала итерации. Оно сравнивается с расчётным средним временем на одну итерацию, и при необходимости делается пауза. Последовательность «читать-отправить-ждать» позволяет принимать во внимание все последние изменения внутри тестируемой системы и на их основе посылать корректные с функциональной точки зрения сообщения.

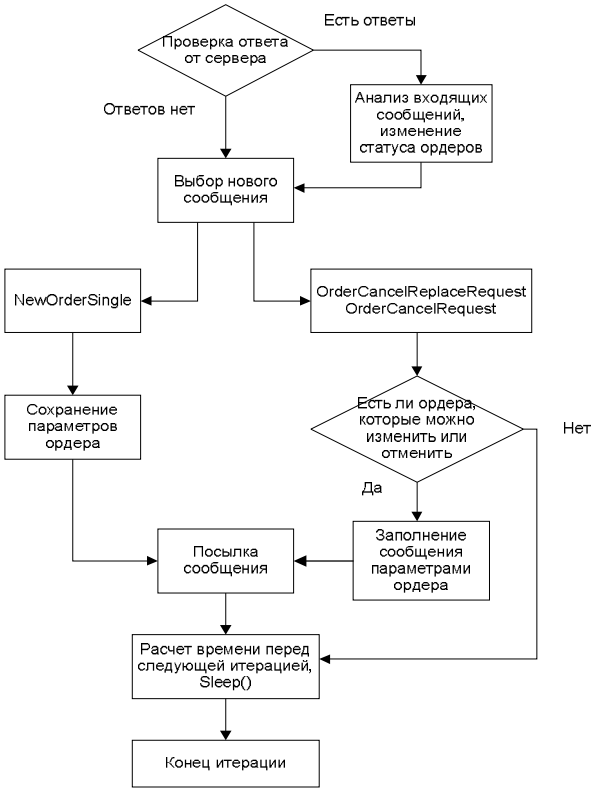


**Рис. 1**. Блок-схема получения и отправки сообщения.

Запуск теста с максимальной нагрузкой на Intel(R) Xeon(R) CPU X5570 @ 2.93GHz даёт на выходе с одного ядра до 70 000 сообщений в секунду и линейно масштабируется при использовании большего количества ядер. Результаты были подтверждены при распределении генерирующих потоков по 8 ядрам и использовании промышленной биржевой системы в качестве мишени.

Указанный объем нагрузки, создаваемой с одного сервера, превышает пропускную способность существующих систем торговли акциями. Показатель 70,000 исходящих сообщений в секунду с одного ядра соответствует максимальным показателям инструментов для тестирования веб-инфраструктур с помощью статических запросов [17].

### 4.6. Настройка профиля нагрузки

В этой части статьи описывается настройка профиля нагрузки. Нагрузка задается параметром:

```
LOAD_CONFIG = фаза1 [, фаза2, … фазаN]
```

Нагрузочная фаза может быть следующей:

`-const(freq, dur)` – постоянная нагрузка с частотой freq и длительностью dur. Возможно также использовать сокращенный формат – freq:dur;

`-step(freq, delta, steps, dur)` – увеличивающаяся нагрузка с начальной частотой freq, шагом изменения частоты delta, количеством шагов steps и длительностью одного шага dur;

`- connect(dur)` – все сессии должны установить соединение с задержкой dur;

`- disconnect(dur)` – все сессии должны оборвать соединение с задержкой dur;

`- logon(dur)` – все сессии должны послать сообщение с авторизацией с задержкой dur;

`- logout(dur)` – все сессии должны послать сообщение о прекращении сессии с задержкой dur;

Обрыв соединения сам по себе не является критической проблемой. Например, все web-соединения, и особенно соединения в мобильных приложениях, рассчитаны на то, что они будут прерываться. Для финансовых протоколов это событие может означать потерю участником торговли контроля над его заявками, и многие системы настроены на отмену всех открытых заявок. Если клиент активно ведёт торговлю и выставляет много заявок, потеря соединения приведет к тому, что его заявки будут массово отменены системой, что вызовет повышенную нагрузку на ядро системы. Также необходимо знать, как поведёт себя система при восстановлении соединения и авторизации под нагрузкой. Очень важно иметь возможность воспроизводить внезапный обрыв и восстановление соединения под нагрузкой. Для этой цели были созданы выше описанные фазы: connect, disconnect, logon, logout.

На рисунке 2. Изображены различные нагрузочные профили const, step и micro burst. Последний профиль создается при помощи фазы const с малой длительностью и высокой нагрузкой.

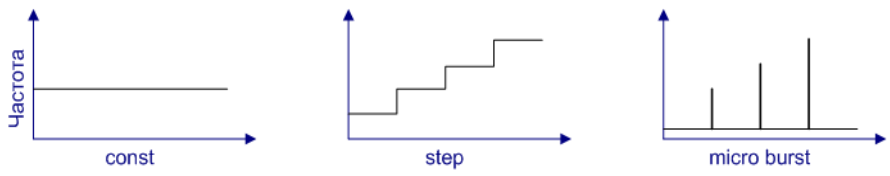


**Рис. 2**. Простейшие варианты профилей нагрузки.

```
const: LOAD_CONFIG=const(1000, 20m)
```

```
step: LOAD_CONFIG=step(500, 500, 4, 4m)
micro burst: LOAD_CONFIG=200:5m, 40000:10ms, 200:5m, 75000:10ms, 200:5m
```

Из опыта авторов, нагрузка в форме ступеней (step) в наибольшей степени подходит для определения максимальной производительности системы. Нагрузка в форме микро-всплекса в наибольшей степени воспроизводит поведение современных высоконагруженных трейдинговых систем.

## 5 Заключение

Созданный и описанный в данной статье инструмент тестирования используется при измерении пропускной способности и времён отклика крупномасштабных биржевых и брокерских платформ, обеспечивающих технологическую инфраструктуру финансовых рынков, в рамках проектов, осуществляемых при поддержке компании Exactpro Systems LLC. Достигнутые результаты подтверждают эффективность выбранных методов работы: управление исполняемыми потоками посредством центрального контроллера и использование заготовленных шаблонов при генерации потока сообщений.

Планируется дальнейшее расширение списка открытых и коммерческих протоколов коммуникаций, поддерживаемых данным инструментом для тестирования. Несмотря на то, что существующая производительность генератора нагрузки позволяет создать реалистичный поток данных, используя один сервер, достаточный для перегрузки любой из существующих торговых площадок, а также для обеспечения качества трейдиговых систем, которые появятся в ближайшие годы, планируется разработка масштабируемого модуля, позволяющего контролировать нагрузку, создаваемую с нескольких серверов.

Основным направлением исследовательской работы станет совершенствование механизмов обработки обратного потока данных для повышения сложности и реалистичности сценариев нагрузочного тестирования торговых платформ. При этом высокая экономичность и эффективность используемых для тестирования инструментов сохранятся.

## Источники


1. The Future of Computer Trading in Financial Markets. Final Project Report. / Foresight. The Government Office for Science, London. // [Электронный ресурс]. – Режим доступа http://www.bis.gov.uk/assets/foresight/docs/computer-trading/12-1086-future-of-computer-trading-in-financial-markets-report.pdf
2. Commission Roundtable on Technology and Trading: Promoting Stability in Today's Markets. / U.S. Securities and Exchange, October 2, 2012 // [Электронный ресурс]. – Режим доступа http://www.sec.gov/news/otherwebcasts/2012/ttr100212-transcript.pdf
3. Apache JMeter manual // [Электронный ресурс]. – Режим доступа http://jmeter.apache.org/usermanual/jmeter_distributed_testing_step_by_step.pdf
4. HP LoadRunner manual // [Электронный ресурс]. – Режим доступа ftp://ftp.itrc.hp.com/applications/HPSoftware/ONLINE_HELP/LoadRunner11.50_User.pdf

5. Rational Performance Tester // [Электронный ресурс]. – Режим доступа http://www-03.ibm.com/software/products/ru/ru/performance/
6. Silk Performer // [Электронный ресурс]. – Режим доступа http://www.borland.com/products/silkperformer/
7. Penhaligan,P.: Equity Trading: Performance, Latency & Throughput. / ExTENT Conference // [Электронный ресурс]. – Режим доступа http://www.slideshare.net/extentconf/extent3-turquoise-equitytrading2012
8. Benedetti E., Zanetti L.: London Stock Exchange - "The Focus Beyond Low Latency". / ExTENT Conference // [Электронный ресурс]. – Режим доступа http://www.slideshare.net/extentconf/extent-2013-obninsk-lse-the-focus-beyond-low-latency
9. Cong, J.: Load Specification and Load Generation for Multimedia Traffic Loads in Computer Networks. // Ph.D. Dissertation, FB Informatik, Univ. Hamburg, 2006; also published at: Shaker-Verlag, Aachen 2006
10. Jing Cong, Bernd E. Wolfinger: A Unified Load Generator Based on Formal Load Specification and Load Transformation // valuetools '06: Proceedings of the 1st international conference on Performance evaluation methodolgies and tools
11. Bodík P., Fox A., Franklin M., Jordan M., Patterson D.: Characterizing, Modeling, and Generating Workload Spikes for Stateful Services // SoCC'10, June 10–11, 2010
12. David Mosberger, Tai Jin: httperf—a tool for measuring web server performance // SIGMETRICS Performance Evaluation Review , Volume 26 Issue 3, December 1998
13. Иткин И.Л.: Тестирование биржевых систем в условиях высокочастотного трейдинга // SQA Days #10: http://sqadays.com/talk.sdf/sqadays/11151/talks/12196
14. Yandex.Tank Documentation // [Электронный ресурс]. – Режим доступа https://media.readthedocs.org/pdf/yandextank/latest/yandextank.pdf
15. Itkin Iosif: Theory of High Frequency Trading systems testing // Software Development & Analysis Technologies Seminar http://sdat.ispras.ru/2011/09/20-октября-модели-тестирования-систем/ http://www.slideshare.net/IosifItkin/theory-of-high-frequency-trading-systems-testing
16. Официальный сайт FIXprotocol // [Электронный ресурс]. - Режим доступа http://www.fixprotocol.org/
17. How to Generate Millions of HTTP Requests // [Электронный ресурс]. - Режим доступа http://dak1n1.com/blog/14-http-load-generate