

Санкт-Петербургский государственный университет

Программная инженерия

Балашов Илья Вадимович

Разработка встроенного распределённого хранилища для системы алгоритмической торговли

Отчёт по учебной (ознакомительной) практике

Научный руководитель:
к.ф.-м.н., доцент кафедры СП Д.В.Луцев

Консультант:
ведущий архитектор ЧАО "Айтивити Софтвэр Ист Аб" А.В.Торопов

Санкт-Петербург
2021

Оглавление

Введение	3
1. Постановка задачи	4
2. Обзор предметной области	5
2.1. Системы алгоритмической торговли	5
2.2. Функциональные требования к СУБД	5
2.3. Обзор распределённых СУБД	6
2.3.1. GridDB	7
2.3.2. QuestDB	7
2.3.3. ScyllaDB	8
2.3.4. MongoDB	8
2.3.5. Tarantool	8
2.3.6. Итоги	9
3. Проведение экспериментов	10
3.1. Проведение замеров производительности	10
3.2. Проверка концепции внедрения в систему	11
3.3. Итоги	12
4. Дальнейшие планы	14
5. Промежуточные выводы	15
Список литературы	16

Введение

Во многих голливудских фильмах о финансовом рынке биржа часто изображается как огромное количество людей, собравшихся в одном зале с блокнотами, и пытающихся друг друга перекричать в попытках заключить сделку. Действительно, такая организация торгов существовала на протяжении многих веков. Однако, с развитием информационных технологий и Интернета на рубеже 20 и 21 веков, "живые" торги были стремительно вытеснены высокопроизводительными автоматическими системами, способными размещать заказы на порядок быстрее человека [1].

Ключевыми значениями для подобных систем являются задержка (latency) и пропускная способность (throughput), с которыми они могут совершать сделки на рынке, причём счёт идёт на миллисекунды. Как следствие, одна поломка оборудования или незапланированная остановка торговой системы могут полностью парализовать работу трейдера и привести к существенным финансовым потерям. По этой причине, разработчики систем алгоритмической торговли, как правило, предоставляют инструменты для обеспечения высокой доступности для созданных ими систем. Одной из таких систем является Tbricks от компании Itiviti [2].

Одним из ключевых элементов высокой доступности всей программной системы — высокая доступность баз данных [3]. Потеря данных или доступа к ним может привести как к финансовым потерям в моменте, так и к долгосрочным проблемам. К примеру, в 2018 году в Европейском Союзе в силу вступила регуляция Mifid 2, которая, помимо прочего, требует хранить данные о сделках на протяжении пяти лет [4].

В рамках данной работы будет осуществлена разработка и внедрение встроенной распределённой базы данных для системы алгоритмической торговли Tbricks для обеспечения её высокой доступности.

1. Постановка задачи

Целью данной работы является разработка встроенной распределённой СУБД (системы управления базами данных) для целей высокой доступности, а также её интеграция в систему Tbricks [2].

Для достижения указанной цели на учебную практику были вынесены указанные далее промежуточные задачи.

- Выполнить обзор:
 - основных функциональных требований к необходимой СУБД;
 - существующих на рынке распределённых СУБД, удовлетворяющих данным требованиям.
- Провести исследование производительности выбранных СУБД.
- Осуществить проверку концепции (PoC) интеграции Tbricks с СУБД, показавшими себя наилучшим образом.
- Проанализировать результаты экспериментов и интеграции, поставить задачи для дальнейшей реализации цели работы.

2. Обзор предметной области

В данной главе будут описаны функциональные требования к СУБД, реализующим принцип высокой доступности. Также будет представлен краткий обзор существующих СУБД, соответствующих указанным требованиям.

2.1. Системы алгоритмической торговли

Системы алгоритмической торговли, такие как Tbricks by Itiviti [2], позволяют на основе данных, получаемых от поставщиков данных, принимать решения о размещении заказов (ордеров) с использованием алгоритмических стратегий.

Основными техническими особенностями таких систем являются:

- низкая задержка, высокая пропускная способность;
- распределённость (в том числе географическая);
- физическое расположение оборудования у клиента, чаще всего в колокации с биржей. Таким образом, ресурсы ограничены и не могут быть быстро наращены;
- высокая сложность систем, большое количество различных алгоритмических стратегий.

2.2. Функциональные требования к СУБД

Высокая доступность системы в общем случае формируется из 4х основных составляющих, указанных далее [5].

1. Восстановление после инцидентов — возможность легко перенести отказ компонента и восстановиться после него.
2. Поддерживаемость — способность системы предоставлять информацию о проблемах и их источниках, а также способах их решения.

3. Управляемость — способность системы создать и поддерживать условия, ограничивающие негативное воздействие, которое пользователи могут оказать на систему.
4. Устойчивость — возможность системы работать в установленных условиях установленное количество времени. В контексте СУБД под этим понимаются следующие технологии [3]:
 - репликация — синхронизация нескольких копий одних и тех же данных [6];
 - шардирование — разбиение базы данных на несколько независимых частей (шардов) [3].

Кроме того, учитывая технические особенности систем алгоритмической торговли, добавляются следующие требования к высокодоступной СУБД:

- высокая производительность и приемлемое потребление ресурсов;
- соответствие принципам ACID [7] (атомарность, согласованность, изолированность, прочность);
- поддержка обработки информации в потоковом режиме работы (стриминга) для обеспечения минимальных задержек и максимальной пропускной способности;
- распределённость;
- приемлемая трудоёмкость интеграции (на примере системы Tbricks).

2.3. Обзор распределённых СУБД

В этом разделе будет проведён обзор существующих распределённых СУБД, удовлетворяющих поставленным функциональным требованиям.

2.3.1. GridDB

GridDB представляет собой гибридную СУБД, работающую как в резидентном режиме (БД в оперативной памяти), так и в режиме хранения БД на постоянном накопителе [8].

Можно отметить указанные далее положительные черты GridDB в соответствии с функциональными требованиями.

- Хорошая документация.
- Простота установки.
- Удобный интерфейс с языком Си.
- Поддержка индексации, репликации, шардирования и других средств обеспечения высокой доступности.

Однако, существуют и отрицательные особенности:

- репликация и шардирование только в платной версии;
- кластер прекращает работу, если больше половины узлов недоступны.

2.3.2. QuestDB

QuestDB — реляционная, колонно-ориентированная база данных [9].

Из документации [9] выяснены следующие особенности QuestDB:

- + простота установки;
- + наличие REST/Postgres/InfluxDB интерфейса;
- + отличное Java API;
- плохая документация;
- репликация и шардирование только в платной версии;
- высокое потребление ресурсов.

2.3.3. ScyllaDB

ScyllaDB — СУБД с максимальной совместимостью с CassandraDB, однако переписанная с Java на C++ [10].

Из документации [10] выяснены следующие особенности ScyllaDB:

- + удобное CassandraDB API;
- + поддержка асинхронной репликации;
- + продвинутая поддержка шардирования;
- громоздкая документация;
- по оценкам инженеров компании Itiviti, ScyllaDB будет тяжело интегрировать в систему алгоритмической торговли Tbricks.

2.3.4. MongoDB

MongoDB — документоориентированная СУБД, не требующая описания схемы таблиц [11].

Из документации [11] выяснены следующие особенности MongoDB:

- + отличная документация;
- + продвинутая поддержка асинхронных репликации и шардирования;
- + поддержка зонирования шардов [12];
- слишком громоздкая, большинство функций в случае интеграции использоваться не будет;

2.3.5. Tarantool

Tarantool — гибридная платформа вычислений (резидентный и с хранением на постоянном накопителе) для эффективного создания высоконагруженных приложений. Платформа включает в себя базу данных и сервер приложений.

Из документации [?] выяснены приведённые ниже особенности Tarantool.

- + Простота и разнообразие способов установки (исходные коды, пакеты, из менеджеров пакетов).
- + Адекватная документация.
- + Поддержка первичных/вторичных индексов, транзакций, сжатия и другого.
- + Продвинутая поддержка репликации.
- +/- Шардирование поддерживается не напрямую, а через сторонний модуль vshard.

2.3.6. Итоги

Подводя итог, был сделан вывод, что все рассмотренные базы данных в целом соответствуют описанным функциональным требованиям.

3. Проведение экспериментов

3.1. Проведение замеров производительности

Одним из основных функциональных требований, предъявляемых к СУБД для системы алгоритмической торговли, является высокая производительность и умеренное потребление ресурсов. Поэтому, при выборе наиболее оптимальной СУБД для внедрения в систему Tbricks необходимо учесть время работы и затраты памяти на как можно более приближенных к реальным условиям данных и конфигурациях. Все результаты приведены в сравнении со встроенными в систему Tbricks базами данных (Audit/ParametersAudit).

В качестве данных для эксперимента была взята база данных, содержащая примерно 4 миллиона заказов (ордеров), экспортированная (в формате JSON) из системы Tbricks, развёрнутой внутри компании Itiviti. Замеры проводились на стенде со следующей конфигурацией: AWS m2.2xlarge, Intel® Xeon® Platinum 8175M (2.50GHz x 8), 32Gb RAM, RHEL 8.5.

В каждую из СУБД проводилась полная запись и полное чтение указанной базы данных. При этом проводились следующие замеры:

- полное время записи;
- полное время записи (при записи блоками по 1000 элементов);
- задержка при совершении одной транзакции;
- задержка при совершении одной транзакции (при записи блоками по 1000 элементов);
- полное время чтения;
- полное время повторного чтения;
- время чтения в 10 потоков (на 1 поток);
- время чтения в 30 потоков (на 1 поток);

- занимаемое место на диске.

Для реализации замеров производительности были разработаны скрипты на языке Python 3, код которых закрыт. Замеры времени исполнения осуществлялись с использованием утилиты `time` из стандартной поставки `bash`. Другие замеры также проводились с использованием стандартной библиотеки Python. Воспроизводимость проверена путём многократного исполнения замера производительности, наблюдаемая погрешность — около 5%. Каждая СУБД использовалась в режиме записи на постоянный носитель.

	Полная запись, с.	Полная запись (блоки по 1000), с.	Задержка на одной транзакции (на одну запись), мс.	Задержка одной транзакции (блоки по 1000), средн., мс.	Полное время чтения (первое чтение), с.	Полное время чтения (повторное чтение), с.	Полное время чтения (10 потоков, на поток), с.	Полное время чтения (30 потоков, на поток), с.	Место на диске
Audit	146	-	31	-	13	12	-	-	430мб
ParametersAudit	120	120	25	26500	19	18	-	-	320мб
Tarantool	250 (110 при записи блоками)	45	46	10000	21	6	65	215	500мб – 1.7гб
GridDB	290	150	55	34000	34	28	40	100	800мб – 1.5гб
ScyllaDB	960 (184 при параллельной записи)	84 (25-47 при параллельной записи)	230	-	33	31	92	270	350мб
QuestDB	800	330	120	82000	6	5	10	60	2.5гб
MongoDB	390	55	90	11000	4	3	3	5	750мб

Таблица 1: Результаты замеров производительности различных сторонних СУБД

Результаты замеров представлены в таблице 1.

Таким образом, можно сделать вывод, что наиболее производительным решением на выбранных данных являются Tarantool и MongoDB.

3.2. Проверка концепции внедрения в систему

Наиболее производительными и соответствующим функциональным требованиям для дальнейшей работы по совокупности свойств признаны СУБД Tarantool и MongoDB. Однако, эти СУБД, вероятно, могут оказаться негибким при работе как часть существующей сложной системы алгоритмической торговли. Также, само внедрение стороннего по отношению к системе решения может составить существенную трудность. Таким образом, перед внедрением указанных СУБД необходимо

оценить, насколько повысится сложность структуры системы в результате внедрения.

В качестве основных требований по интеграции инженерами и клиентами Itiviti были выдвинут ряд требований требования к количеству реплик шардов, перечисленных далее.

- 2 или 3 реплики на каждый шард.
- 1 или более реплики для каждого шарда в изолированной системе (от функционирования которой не зависит функционирование системы).
- Желательно, дополнительная инсталляция в облаке для хранения резервных копий.

В соответствии с данными требованиям, была выполнена расстановка шардов СУБД MongoDB и Tarantool по компонентам системы. В результате были выявлены следующие проблемы:

- сложное развёртывание и настройка;
- новые точки отказа;
- ломается взаимодействие между компонентами;
- дополнительный "прыжок" по сети;
- существенное увеличение количества экземпляров СУБД на каждой компоненте.
 - Использование существенного количества ресурсов оборудования и памяти на диске.

3.3. Итоги

Подводя итог, сторонние СУБД удовлетворяют функциональным требованиям, а также в целом удовлетворяют требованиям по производительности. Однако внедрение сторонних СУБД может привести су-

щественные проблемы и усложнить архитектуру системы. По этой причине было принято решение рассмотреть возможность реализации собственного легковесного решения на базе некоторой встроенной СУБД. Сторонние же СУБД могут быть использованы по желанию для сохранения резервных копий.

4. Дальнейшие планы

В качестве дальнейших планов можно выдвинуть описанные далее задачи.

- Построить эксперименты по возможности внедрения существующие встроенных СУБД, с самостоятельной реализацией необходимой распределённой функциональности.
- Разработать схему базы данных для хранения заказов (ордеров) в выбранной СУБД.
- Внедрить выбранную СУБД в одну из компоненты системы Tbricks.

5. Промежуточные выводы

По результатам работы, проведённой в рамках учебной практики, были выполнены все поставленные задачи, перечисленные ниже.

- Выполнен обзор:
 - основных функциональных требований к необходимой СУБД;
 - существующих на рынке следующих распределённых СУБД, удовлетворяющих данным требованиям.
 - * GridDB.
 - * QuestDB.
 - * ScyllaDB.
 - * MongoDB.
 - * Tarantool.
- Проведено исследование производительности выбранных СУБД.
 - Сделан вывод о том, что на наборе данных, взятых из системы алгоритмической торговли, наиболее производительным является СУБД Tarantool.
- Осуществлена проверка концепции (PoC) интеграции Tbricks с СУБД MongoDB и Tarantool.
- Проанализированы результаты экспериментов.
 - Все рассмотренные распределённые СУБД в целом соответствуют функциональным требованиям.
 - Принято решение рассмотреть в дальнейшем существующие встроенные СУБД.
- Поставлены задачи для дальнейшей реализации цели работы.

Список литературы

- [1] History of automated trading systems. — Access mode: <http://www.automatedtrading.com/2014/01/13/history-trading-systems/> (online; accessed: 29.12.2021).
- [2] Tbricks official site. — Access mode: <https://www.itiviti.com/tbricks> (online; accessed: 29.12.2021).
- [3] Domaschka Jörg, Hauser Christopher B, Erb Benjamin. Reliability and availability properties of distributed database systems // 2014 IEEE 18th International Enterprise Distributed Object Computing Conference / IEEE. — 2014. — P. 226–233.
- [4] Directive 2014/65/EU (MIFID 2). — Access mode: <https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32014L0065> (online; accessed: 29.12.2021).
- [5] Gray Jim, Siewiorek Daniel P. High-availability computer systems // Computer. — 1991. — Vol. 24, no. 9. — P. 39–48.
- [6] Database replication techniques: A three parameter classification / Wiesmann Matthias, Pedone Fernando, Schiper André, Kemme Bettina, and Alonso Gustavo // Proceedings 19th IEEE Symposium on Reliable Distributed Systems SRDS-2000 / IEEE. — 2000. — P. 206–215.
- [7] Haerder Theo, Reuter Andreas. Principles of transaction-oriented database recovery // ACM computing surveys (CSUR). — 1983. — Vol. 15, no. 4. — P. 287–317.
- [8] GridDB documentation. — Access mode: <https://docs.griddb.net/> (online; accessed: 29.12.2021).
- [9] QuestDB documentation. — Access mode: <https://questdb.io/docs/introduction/> (online; accessed: 29.12.2021).

- [10] ScyllaDB documentation. — Access mode: <https://docs.scylladb.com/> (online; accessed: 29.12.2021).
- [11] MongoDB documentation. — Access mode: <https://docs.mongodb.com/> (online; accessed: 29.12.2021).
- [12] MongoDB zones support. — Access mode: <https://docs.mongodb.com/manual/core/zone-sharding/> (online; accessed: 29.12.2021).