Решение задачи быстродействия в DocsVision 4.5

Антон Варфоломеев, DocsVision, директор по производству.

Цели и задачи быстродействия системы

В системах электронного документооборота и управления бизнес-процессами быстродействие является одним из важнейших потребительских свойств системы.

При этом ожидания пользователя достаточно высоки – система не должна выглядеть существенно медленнее других используемых менеджером приложений – Microsoft Office, Internet Explorer, интернет-почта. Реальное быстродействие системы при выборе ее потребителем может повлиять сильнее чем функциональность. Конкретная функция может кому-то и не требоваться, не быть критичной, и ее отсутствие снизит спрос на систему только частично. В том время как недостаточное быстродействие может быстро вытеснить продукт с рынка.

Таким образом, управление быстродействием системы становится ключевой частью системы управления качеством ее разработки.

Осенью прошлого года компания DocsVision провела исследование важности потребительских свойств системы в глазах наших клиентов и партнеров. В опросе участнику предлагалось оценить каждое свойство как критичное, важное или неважное. Первую тройку по сумме оценок «Критично» и «Важно» заняли следующие свойства:

Качество	Критично	Важно	Неважно
Быстродействие	73%	27%	0%
Удобный, простой и понятный интерфейс	55%	45%	0%
Широкая функциональность	28%	60%	12%

Из таблицы в частности видно что быстродействие является по настоящему критичным, в то время как функциональность – скорее важным свойством.

До версии 4.5 в разработке DocsVision требования по быстродействию не выделялись среди других (функциональных) требований. При начале формирования спецификаций на новую версию руководством компании было принято решение выделить управление быстродействием как отдельную ключевую задачу. Для этого необходимо было:

1. Определить набор сценариев пользователя по которым измеряется быстродействие системы
2. Определить и описать модель пользователя и эталонную вычислительную инфраструктуру, на которой производится измерение.
3. Построить методику и технологию измерения быстродействия в процессе тестирования системы

Эти три пункта в сумме являются методологией оценки быстродействия системы. Далее необходимо:

4. Провести измерение быстродействия текущей версии системы, определив точку отсчета
5. Задать целевые значения метрик быстродействия разрабатываемой версии
6. Разработать новую версию и добиться выполнения целевых значений метрик

При переходе на уровень архитектуры системы задача повышения быстродействия как потребительского свойства превращается в более широкую задачу повышения производительности, относящейся к комплексу программного обеспечения, компьютерных платформ и коммуникаций. При этом важнейшим вопросом является выявление факторов производительности в архитектуре ПО системы, т.е. определение наиболее «тормозящих» компонент системы. Это необходимо в частности для того чтобы перейти от конечных требования по быстродействию на пользовательском уровне к метрикам производительности в заданиях на разработку отдельных модулей и распределении этой задачи по коллективу разработчиков.

Для определения границ задачи управления быстродействием мы использовали собственную экспертизу и опросы пользователей и партнеров, для определения набора сценариев пользователя и определения целевых значений быстродействия – детальное интервьюирование очень крупного пользователя (по количеству рабочих мест, территориальной распределенности и интенсивности документооборота, места документооборота в ряду бизнес-критичных приложений).

Требования к быстродействию

Вопрос на пользовательском уровне кажется простым. Пользователя интересует, в первую очередь, сколь долго он вынужден будет смотреть на “песочные часы”, вызвав ту или иную команду в системе, и успеет ли он за это время выпить чашку чаю. И если для некоторых операций (например, резервное копирование базы данных) допустимо время до нескольких десятков минут – то для наиболее востребованных операций может оказаться критичным время и в десятые доли секунды. Поэтому для управления быстродействием требования по времени реакции надо «спроецировать» на набор базовых сценариев работы системы, чтобы не попасть в «западню комбинаторики» - измерять быстродействие при любом сценарии из всех технически возможных.

Таким образом, прежде чем приступать к каким-либо измерениям, необходимо перечислить и описать эти самые типовые сценарии, для которых будут производиться измерения. Это могут быть операции, выполняемые пользователями чаще всего, по многу раз в день; а могут быть редко выполняемые, но критичные для функционирования системы операции (в частности, связанные с администрированием). При решении задачи управления быстродействием на данном этапе мы исследовали только операционные сценарии, затрагивающие всех или почти всех пользователей и поэтому наиболее критичные с точки зрения потребительских свойств системы.

На уровне платформы мы выделили около 30 сценариев работы пользователя, включая:

- Запуск пользовательского интерфейса (навигатора) системы;
- Открытие встроенных справочников системы;
- Атрибутивный поиск;
- Операции экспорта представлений и журналов;
- Создание карточки со 100 свойствами;
- Добавление, сохранение и удаление файлов (размером от 100Кб до 10Мб).
- Отображение папок по дайджесту и представлению (содержащих от 10 до 70 000 строк)

Помимо сценариев пользователя, исполняемых платформой на интерфейсном уровне пользователя, мы выделили как часть архитектуры подсистему управления бизнес-процессами (WorkFlow), быстродействие которой также влияет на сценарии пользователя в интерфейсе приложения. Для измерений быстродействия WorkFlow использовался единственный сценарий

- Полная обработка эталонного бизнес-процесса, состоящего из 20 функций вычислительной направленности (обмен данными и их обработка), с момента его запуска и до полного окончания обработки.

Поскольку DocsVision является платформой для задач СЭДО и управления бизнес-процессами, то пользователь часто работает не с самой платформой, а с приложением, построенным на ее базе. Поэтому правильно разделить сценарии между платформой и приложением и выбрать достаточно типичное приложение. Мы использовали для моделирования приложение DocsVision «Административное делопроизводство». Задача управления быстродействием решалась одновременно и для платформы и для приложения, оптимизации подверглись и код платформы, и код приложения.

В приложении «Административное производство», использующем собственные регистрацинные карточки документов, были выделены 16 сценариев работы пользователя, включая:

- Создание (первичное и повторное) новой регистрационной карточки приложения
- Открытие (первичное и повторное) заполненной регистрационной карточки приложения
- Сохранение регистрационной карточки приложения
- Создание и открытие карточки задания
- Утверждение и отзыв резолюций
- Регистрация (выделение номера) карточки приложения

Исследование литературы по требованиям к быстродействию офисных приложений показало что в этой отрасли считается приемлемым быстродействие сценариев на уровне 3 секунд, и хорошим на уровне 1,5 секунд. Эти значения использовались в качестве целей при оптимизации кодов платформы и приложения.

Задачи измерения быстродействия

Факторы, влияющие на быстродействие системы естественным образом разделяются на две группы:

1. Факторы инфраструктуры (архитектура и качества кода программного обеспечения системы, вычислительная инфраструктура серверов, рабочих станций и сетей передачи данных)
2. Факторы нагрузки в конкретном внедрении (количество одновременно работающих пользователей, частота применения ими различных сценариев)

Поэтому мы выделяем две задачи управления быстродействием - тестирование производительности и нагрузочное тестирование.

Тестирование производительности представляет собой измерение скорости выполнения основных операций в системе в некоторых идеальных условиях. Данный вид тестирования преследует следующие цели:

- Оценка влияния инфраструктуры (аппаратного и программного обеспечения) на быстродействие системы;
- Оценка максимально возможной скорости отклика системы в типовых сценариях, и соответствия этих показателей предъявляемым требованиям «в идеальных условиях».

В то же время, **нагрузочное тестирование** дает иные оценки:

- Оценка соответствия конкретной аппаратной и программной инфраструктуры прогнозируемой нагрузке
- Оценка влияния нагрузки на быстродействие в «в реальных условиях».

Эти два вида тестирования в совокупности позволяют всесторонне оценить все факторы, влияющие на производительность. Далее рассмотрим каждый из видов тестирования по отдельности.

Тестирование производительности

При тестировании производительности необходимо получить оценки:

1. Влияния инфраструктуры (аппаратного и программного обеспечения) на быстродействие системы
2. Максимально возможной скорости отклика системы в типовых сценариях, и соответствия этих показателей предъявляемым требованиям

Говоря об оценке влияния инфраструктурных показателей, прежде всего необходимо выделить те из них, которые могут в принципе влиять на производительность.

Например, система DocsVision архитектурно реализована по классической трехуровневой модели:

- База данных – представляет собой реляционное (OLTP) хранилище информации;
- Сервер приложений – реализует всю бизнес-логику по обработке данных, хранящихся в базе; обрабатывает запросы клиентов и возвращает результат;
- Клиентская часть реализует пользовательский интерфейс для интерактивного взаимодействия с системой.

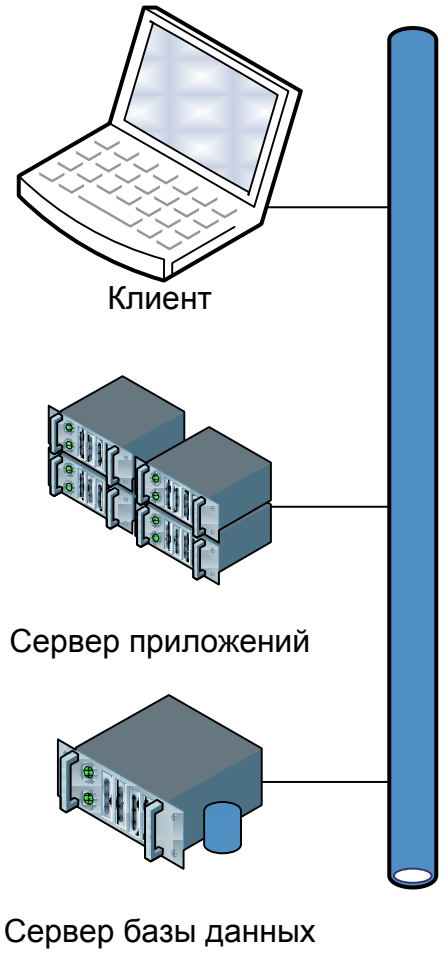


Рис. 1. Архитектура клиент-серверного ПО

С аппаратной точки зрения, все три уровня могут быть расположены как на одном компьютере (простейший случай), так и на разных компьютерах. Это позволяет гибко распределять нагрузку в зависимости от особенностей конкретной аппаратной инфраструктуры. Кроме этого, подобная архитектура имеет следующие преимущества:

- Гибкие возможности масштабирования - поддерживается возможность кластеризации как серверов приложений (NLB Web-farm), так и серверов баз данных;

- Независимость от платформы СУБД – потенциально сервер приложений может работать с любой реляционной базой данных независимо от платформы (SQL Server, ORACLE, MySQL, и т.д.)

Очевидно, что с точки зрения производительности играют роль характеристики инфраструктуры на всех уровнях системы:

- Аппаратное и программное обеспечение клиентского компьютера
- Характеристики канала связи между клиентом и сервером приложений
- Аппаратное и программное обеспечение сервера приложений, его настройки
- Характеристики канала связи между сервером приложений и базой данных
- Аппаратное и программное обеспечение сервера баз данных, его настройки, степень наполненности и характеристики самой базы данных

Варьируя каждый из этих показателей, и выполняя повторные измерения метрик, можно оценить влияние данного показателя на те или иные метрики, и на производительность системы в целом. Например, в качестве одного из таких тестов, искусственно занижалась пропускная способность сетевого канала между клиентом и сервером приложений (с этой целью использовалась утилита NetLimiter) в пограничных диапазонах 128kbps/512kbps/1mbps.

Кроме того, можно варьировать и иные аппаратные составляющие: объем оперативной памяти и частоту процессора на клиентском компьютере; тип дискового массива на сервере базы данных; и т.д.

Кроме времени операций, можно выделить несколько других метрик, которые могут оказаться полезными для принятия решений по инфраструктуре, либо для выполнения последующих оптимизаций. Среди них:

- Количество клиент-серверных вызовов, выполняющихся в рамках каждого сценария. Данный показатель важен, прежде всего, в условиях низкоскоростных сетевых каналов, и позволяет выделить сценарии, которые в таких условиях будут работать неэффективно. Количество вызовов измерялось с помощью утилиты Fiddler2, которая позволяет отслеживать и фиксировать все вызовы по протоколу http (и впоследствии детально анализировать весь трафик).
- Количество входящего/исходящего трафика клиент-сервер порождаемого каждой операцией – не менее важный показатель для низкоскоростных каналов связи. Измерения выполнялись также с помощью утилиты Fiddler.

Помимо этих прямых метрик, могут быть полезными также и “вторичные” метрики. Например, к ним можно отнести счетчики производительности, показывающие величину нагрузки на аппаратные составляющие всех уровней системы (клиент, сервер приложений, сервер БД):

- Загрузка процессора (%)
- Размер задействованной оперативной памяти (Mb)
- Размер свободной оперативной памяти (Mb)
- Длина очереди к диску (число команд)

Данные метрики могут сниматься с помощью встроенного в ОС Windows средства “Perfomance monitor”, и позволяют сохранить значения метрик за весь период тестирования, и представить результаты в виде удобных графиков.

Впоследствии с помощью данных метрик можно оценить, какие операции (из выполнявшихся в ходе тестирования) вызывали наибольшую нагрузку; и какая аппаратная часть (процессор, память, диск) на каждом из уровней является “узким местом” в быстродействии всей системы.

Для автоматизации процесса измерения метрик и получения оценки максимально возможной скорости отклика системы в типовых сценариях необходимо для каждого такого сценария сделать его подробное описание уже в терминах интерфейса системы (последовательность нажатия кнопок и выполнения команд, необходимая для достижения цели). Необходимо также определить границы измерений (какой момент

считать началом сценария, и по какому признаку судить, что сценарий завершился успешно).

Измерения времени выполнения операций выполнялись с помощью средства автоматизированного тестирования TestComplete. Данный инструментарий позволяет автоматизировать процесс измерений, и тем самым существенно сократить время, затраченное на данный вид тестирования. Среда TestComplete позволяет описать все сценарии, по которым выполняются измерения, прямо в терминах пользовательского интерфейса. Процесс описания сценариев напоминает запись макросов в приложениях Microsoft Office – включается запись, и выполняются необходимые действия в системе. По окончании сценария запись прекращается, и готовая последовательность сохраняется для последующего использования. Записанный сценарий представляет собой скрипт (Jscript,VBScript), который может впоследствии быть изменен или дополнен вручную для выполнения различных вспомогательных действий. Отметим, что на практике такая доработка требуется для большинства записанных сценариев, чтобы обеспечить их адекватность.

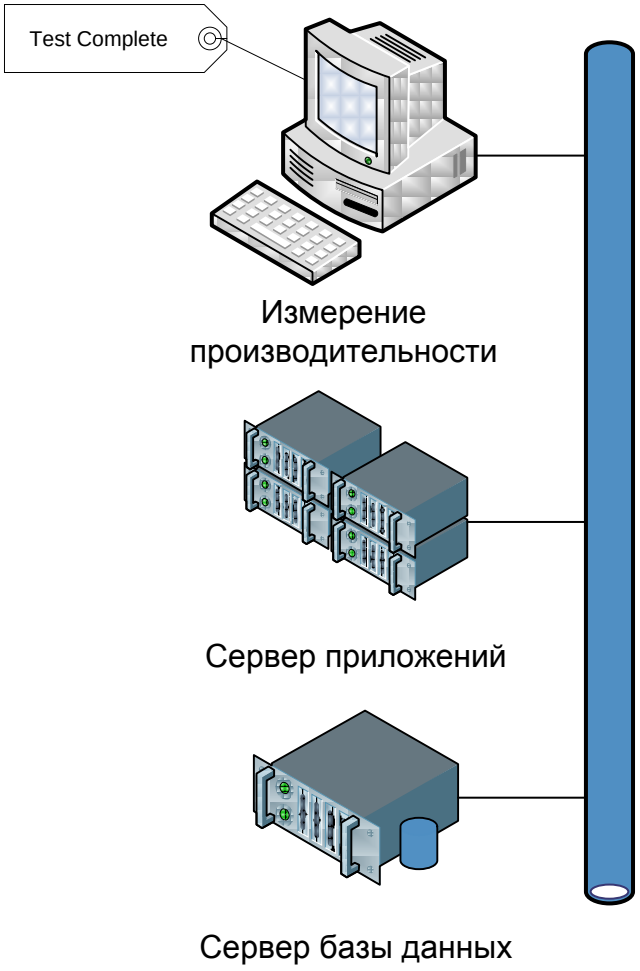


Рис.2. Архитектура испытательного стенда для тестирования производительности

Во время выполнения собственно тестирования, все заранее сохраненные таким образом сценарии ”проигрываются” на работающей системе. При исполнении сценария, можно измерить время, затраченное на его выполнение, и зафиксировать его. Каждый сценарий может быть выполнен и измерен произвольное число раз.

Нужно также обеспечить достаточную точность измерения таких величин, которая не может быть достигнута ручным измерениями. Во-вторых, измерения нужно проводить несколько раз, чтобы исключить случайную погрешность. Мы посчитали достаточным выполнение 5 независимых измерений для каждого сценария, и усреднение полученных результатов.

Измерив скорость выполнения (и вторичные метрики) для некоторого набора операций, можно сделать вывод, какие из них не укладываются в заявленные рамки, и нуждаются в оптимизации. Оптимизации могут заключаться в модификациях исходного кода, направленных на снижение количества вызовов и трафика, оптимизацию вычислительных алгоритмов для снижения нагрузки на аппаратные составляющие и т.п.

Описанная выше методология тестирования позволила выполнить сравнение показателей быстродействия между разными версиями системы, с тем чтобы определить эффективность произведенных улучшений и оптимизаций. Вот, например, результаты сравнения быстродействия двух версий системы DocsVision – 4.3 и 4.5, в последней из которых были проведены работы по оптимизации производительности.

Измерения производительности выполнялись в следующей инфраструктуре:

1. Клиент: Intel Core2 CPU 6420 2.1 GHz, 2Gb RAM, Win 2003 Server
2. AppServer+SQL Server (на одной машине): Intel Core 2 Duo CPU E8500, 3.16 GHz, 4 GB RAM, Windows 2008 Server R2 x64, SQL Server 2008 SP1.

В таблице указано время выполнения основных сценариев в секундах при различных скоростях канала, а также вторичные метрики – число серверных вызовов и трафик клиент-сервер при выполнении данного сценария.

Сценарий	4.3					4.5				
	Скорость канала			Число вызовов	Трафик (байт)	Скорость канала			Число вызовов	Трафик (байт)
	100 Mbps	512 kbps	128 kbps			100 Mbps	512 kbps	128 kbps		
Запуск системы (Навигатора)	3.6	17.8	70.9	584	1038698	2.8	3.4	7.0	99	91240
Развертывание дерева папок (2000 папок)	16	65	260	826	603005	4.3	5.0	11.2	17	121380
Отображение дайджеста на папке с 70000 карточек	17	17	23	5	24399	7	7	7	4	8946
Отображение сложного представления на папке с 10000 карточек	5	6	21	5	18836	3	3	6	7	10072
Создание регистрационной карточки АДП	3.5	7	20	131	264476	0.828	1	1.4	22	19112
Открытие регистрационной карточки АДП	2	3	9	51	100976	0.703	0,8	1.0	19	13498
Создание карточки задания АДП	2	2,5	4	60	72900	0.5	0.6	1.2	32	18612

Видно, что в результате работ по оптимизации быстродействия в версии DocsVision 4.5 удалось значительно (сократить время выполнения основных сценариев . Среднее ускорение сценариев на версии 4.5 произошло в 2, 4 и в 8 раз соответственно для каналов 100Мб, 512 и 128Кбит, т.е. наиболее значительный рост быстродействия произошел на медленных каналах связи 128Кб. При этом максимальное ускорение быстродействие для развертывания дерева папок на канале 128Кб составило 220%.

Видно также что при оптимизации удалось весьма значительно сократить и количество серверных вызовов при исполнении сценария и трафик клиент-сервер.

Нагрузочное тестирование

Напомним цели данного вида тестирования:

- Оценка соответствия конкретной аппаратной и программной инфраструктуры прогнозируемой нагрузке
- Оценка степени деградации быстродействия в зависимости от возрастания нагрузки на сервер

Для получения первой оценки необходимо прежде всего выявить ключевые требования по предполагаемой инфраструктуре, а также характеристики нагрузки. С этой целью выполняется обследование конкретных условий внедрения, в ходе которых фиксируются:

- Параметры аппаратно-программной инфраструктуры, схожей с предполагаемой при внедрении (аппаратные характеристики серверов, клиентов, пропускная способность сети, программная инфраструктура, и т.д.)
- Основные сценарии работы пользователей в системе (модели пользователей)

Модель пользователя

Очевидно, что в зависимости от специфики деятельности конкретной организации, варианты использования программного обеспечения в ней могут иметь существенные отличия от аналогичных предприятий. Особенно это касается таких сложных областей, как системы автоматизации электронного документооборота, которые могут применяться для решения крайне широкого спектра задач. Одни организации могут использовать такие системы просто в качестве хранилища совместных файлов (ECM); другие делают упор на автоматизацию бизнес-процессов (Workflow); третьи – пользуются исключительно канцелярскими функциями, и т.д. И очевидно, что каждый из видов такой деятельности затрагивает специфические функции системы, и порождает таким образом различную нагрузку на нее. Таким образом, для более точного моделирования поведения системы в конкретных условиях, необходимо как можно более подробно и точно описать все варианты ее использования.

Модель предприятия пользователя описывает масштабные факторы заказчика, в частности нами использовалось референтное предприятие со следующими модельными параметрами.

Число документов в день	10 000
Число резолюций в день	30 000
Число пользователей	4 000
Число подразделений (управления и отделы)	490

Для нагрузочного тестирование важным вопросом является частота применения пользователями выделенных сценариев работы. Для этого используется профилирование пользователей Профиль пользователя представляет собой описание всех действий, совершаемых типовым пользователем в системе в течении дня – с указанием их частоты. Например, профиль пользователя “Делопроизводитель” может выглядеть следующим образом:

- Вход в систему – 2-3 раза в день
- Регистрация входящих документов – 20-30 раз в день
- Создание резолюций – 5-8 раз в день
- Открытие файлов до 1 Mb – 10-12 раз в день, более 1Mb – 2-3 раза
- И т.д.

Таких профилей, в зависимости от вариантов применения системы, может быть несколько. Все они должны быть схожим образом сформулированы для определения общей частоты использования сценариев для нагрузочного тестирования. В соответствие с приведенными выше масштабными факторами референтного предприятия, ролями и ролевыми профилями пользователей мы использовали для 4000 пользователей у нас сформировалась следующая нагрузочная модель.

В таблице приведены только наиболее «тяжелые» нагрузочные сценарии и указаны суммарные показатели нагрузки по всем сценариям.

Сценарий	Роли пользователей	Число за 8 часов	Частота исполь-зования сценария , %%	Число в минуту
Открытие РК, открытие и сохранение измененного файла (1 Мбайт).	1. Исполнители при создании внутренних и исходящих документов (до 10 на один документ) 2. Согласующие лица (до 2 циклов согласования с 5 лицами)	133 333	55%	278
Просмотр РК (по всем закладкам и дерева резолюций)	Сотрудники для подготовки других документов или поиске материалов (до 5 документов на каждый исходящий или внутренний)	33 333	14%	69
Открытие резолюции и расписание резолюции на 3 подчиненных	Помощник, руководитель при расписании резолюции	30 000	12%	63
Открытие резолюции и ввод информации о ходе исполнения	Исполнение резолюции (1 раз для каждой резолюции в неделю)	30 000	12%	63
.....				
Всего операций с карточками		243 333		507
Сценарии фильтрации представлений				
Поиск по справочнику сотрудников (100.000 записей, результат 10 записей)	50 раз каждый пользователь в день, 10 раз на каждый документ, 5 раз на каждую резолюцию, 2 раза на каждый отчет	459 800	56%	958
Поиск по справочнику контрагентов (200.000 записей, результат 10 записей)	50 раз каждый пользователь в день, 3 раза на каждый документ, 2 раза на каждый отчет	239 800	29%	500
.....				
Всего операция фильтрации		825 340		1 719

Следующим шагом является реализация модели пользователя в среде, которая будет использоваться для имитации нагрузки. В качестве такой среды мы использовали Microsoft Visual Studio 9.0 (2008), которая имеет встроенную подсистему описания и моделирования тестов. В данной среде необходимо описать каждую атомарную операцию, выполняемую в рамках модели, в терминах программного кода (на языке C# или VB.NET). Для этого используется объектная модель (API) прикладной системы. Здесь важно как можно более точно запрограммировать эти операции, чтобы их реализация в исходном коде соответствовала по набору команд (и порождаемой ими нагрузке) этой же операции, выполняемой через пользовательский интерфейс системы.

Готовые атомарные операции (блоки) объединяются с помощью интерфейса Visual Studio в сценарии. Для каждой операции указывается частота ее выполнения, и вероятность этого события. Эти сценарии могут быть целиком сформированы с использованием ранее описанных моделей пользователей. Таких сценариев также может быть несколько. Далее, в ходе собственно тестирования, каждый сценарий запускается в нескольких экземплярах – каждый экземпляр моделирует работу одного пользователя. Всего в среде Visual Studio возможно наличие до нескольких тысяч одновременно работающих сценариев (в зависимости от аппаратной мощности машины, на которой запускается такая нагрузка).

Параллельно этому, создается еще один выделенный стенд, на котором будут проводиться измерения метрик. Для выполнения измерений можно задействовать тот же набор средств, и ту же методику, что и для тестирования производительности. В частности, имеет смысл измерять время выполнения базовых операций (в среде TestComplete), а также снимать метрики второго порядка – счетчики производительности по нагрузке аппаратного обеспечения.

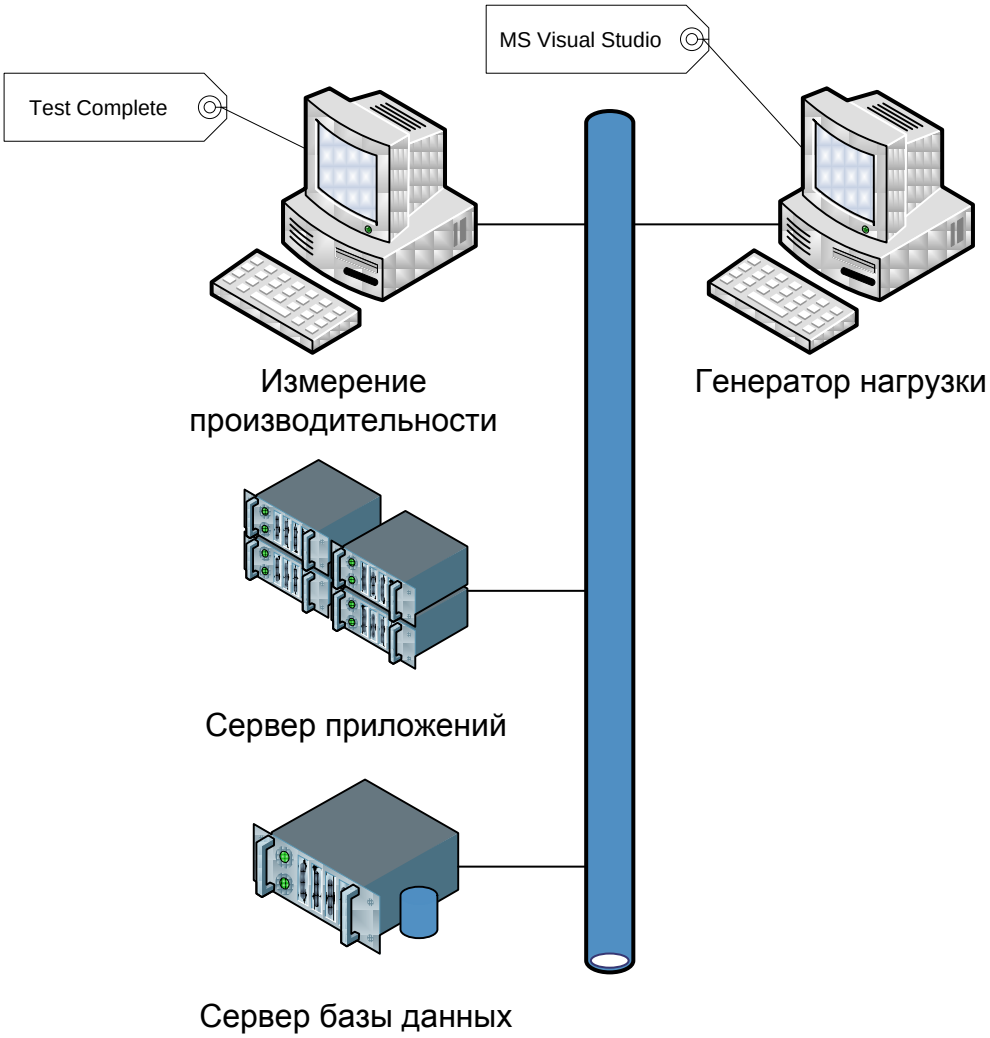
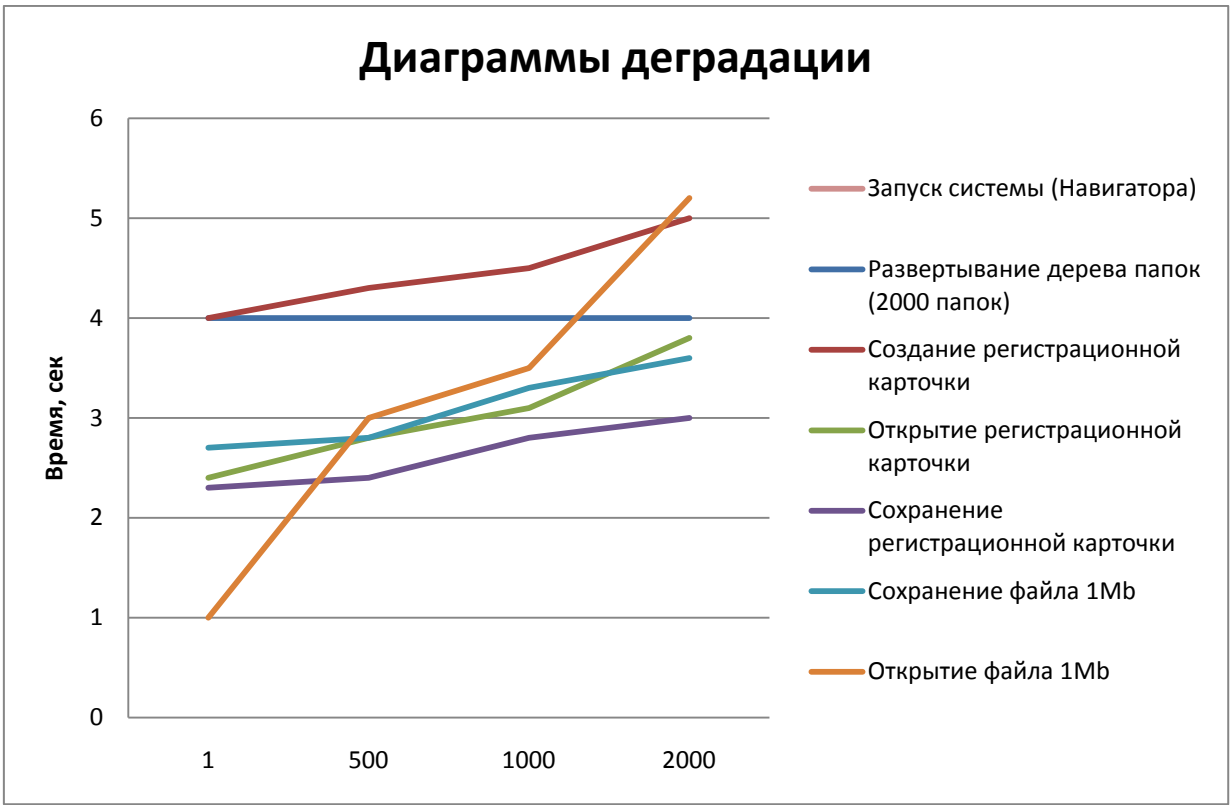


Рис. 3. Архитектура испытательного стенда для нагрузочного тестирования

На завершающей стадии формирования нагрузки необходимо заполнить базу данных, так как степень наполненности базы вносит свои коррективы во времена отклика системы при интенсивной промышленной эксплуатации, соответствующей прогнозируемой нагрузке. В целом обычно используется такая степень наполненности, которая, как предполагается, будет достигнута через 3-4 года использования системы в рабочем режиме.

После выполнения всех этих условий можно приступать к собственно измерениям. Постепенно наращивая нагрузку (число одновременно выполняемых сценариев в Visual Studio), выполняются измерения типовых показателей в среде TestComplete. Прирост нагрузки между измерениями может быть как плавным (1,2,3,4 и т.д. пользователей) так и дискретным (1, 10, 500, 1000 пользователей).

Результаты нагрузочного тестирования представлены в виде диаграмм деградации, которые показывают падение быстродействия ключевых операций в зависимости от возрастающей нагрузки на сервер (по горизонтальной оси отложено количество пользователей, реализующих свои профили работы):



Видно, что в зависимости от специфики сценария деградация происходит различными темпами, при этом для 2000 пользователей большая часть сценариев исполняется быстрее 4 секунд, а до 500 одновременно работающих пользователей эти сценарии укладываются в целевые 3 секунды. Видны также сценарии по которым необходимо продолжать работы по оптимизации кода.

В свою очередь, нагрузочное тестирование подсистемы управления бизнес-процессами (Workflow) выполнялось по несколько иной методике. Сначала осуществлялось выведение сервиса управления процессами на рабочий уровень нагрузки путем запуска одновременно большого количества копий (10, 100, 1000) эталонного бизнес-процесса. После этого осуществлялся запуск еще одной копии бизнес-процесса, для которой уже выполнялось измерение времени обработки от начала до полного завершения. Такая методика позволяет судить о том, насколько быстро подсистема управления процессами реагирует на появление новых процессов, и насколько быстро их обрабатывает, при разных уровнях загруженности.

В таблице ниже приводятся данные нагрузочного тестирования подсистемы управления бизнес-процессами версий DocsVision 4.3 и 4.5.

Время обработки эталонного бизнес-процесса	DocsVision 4.3	DocsVision 4.5
При 10 активных процессах, сек	180	32
При 100 активных процессах, сек	460	345
При 1000 активных процессах, сек	2760	960

Видно что время исполнения бизнес-процесса удалось значительно сократить оптимизацией кода подсистемы управления бизнес-процессами.

Выводы

Главным результатом проведенных исследований и разработок является появление комплексной модели тестирования быстродействия решения на базе платформы DocsVision, включенной в основные производственные процессы компании DocsVision – разработку и тестирование. В результате при выпуске каждой версии

будут проводиться измерения производительности и нагрузочное тестирование и выполняться необходимая оптимизация кода. В конечном счете это дает нашим заказчикам гарантии того, что становясь сложнее функционально, система DocsVision будет становиться еще и быстрее.

Проведенный анализ вторичных метрик быстродействия позволил выделить компоненты и модули, требующие первоочередной оптимизации и выработать стратегию развития системы с учетом важнейшего потребительского качества.

Работы по оптимизации кода сделали систему DocsVision 4.5 значительно быстрее и надежнее, это самый наверное главный их результат.