Методики обеспечения качества:
Независимое тестирование Informix Direct Driver для Clipper

Алексей Лукутин,
Исполнительный директор компании Amphora Quality Technologies,
e-mail: lav@in-amphora.com, тел: (095) 737-0225

Amphora Quality Technologies, российское подразделение Amphora LLC (Денвилл, шт. Калифорния), предоставляет полный спектр услуг по тестированию и обеспечению качества программных продуктов. AQT помогает разрабатывать высококачественное, отвечающее современным требованиям программное обеспечение, соблюдая поставленные сроки. Для обеспечения качества Internet-приложений и решений клиент-сервер используются современные методики и средства автоматизации, ведётся активная исследовательская работа.

Программирование на стыке тысячелетий

В нашей жизни рано или поздно наступает момент, когда надо подводить итог сделанному, оценить достигнутые результаты, поставить перед собой новую, ещё более интересную цель. Смена веков и тысячелетий всегда была хорошим поводом для людей охватить взглядом прошлое и заглянуть в будущее, отвлекшись от сиюминутных забот. Думаю, не ошибусь, если скажу, что люди занятые в ИТ индустрии являются одними из наиболее загруженных работой, начиная от глав транснациональных корпораций и заканчивая службами технической поддержки небольших компаний. Ничего удивительного в этом для человека находящегося в самой гуще событий нет. ИТ отрасль зародилась совсем не так давно, наверное нет ещё и полувека, а за этот период был сделан колоссальный скачок в развитии. Сотни байт информации, обрабатываемой допотопными ламповыми ЭВМ, сменились сотнями гигабайт, перемалываемыми сверхскоростными современными компьютерами. И это далеко не предел, аналитики предсказывают в ближайшее десятилетие ускорение вычислений в сотни раз и во столько же увеличение объёмов обрабатываемых данных. Помимо увеличения объёмов информации, усложняются и методы её обработки, создаются всё более изощрённые алгоритмы, пишутся миллионы строк программного кода. Неудивительно, что человеческий мозг перестаёт справляться с обрушевшейся на него нагрузкой и рядовой ИТ специалист уже не может ориентироваться в мегабайтах кода и гигабайтах данных, хранящихся в СУБД. **Сложность систем становится основным сдерживающим фактором прогресса.** Внесение даже небольших изменений в существующие программы грозит непомерными расходами, нестабильной и непрогнозируемой работой информационных систем.

Эта проблема софтверной индустрии была замечена специалистами достаточно давно. Развитие CASE технологий, средств быстрой разработки приложений помимо повышения эффективности труда программистов включало в себя и меры, направленные на поддержку коллективной работы, средств декомпозиции и структуризации программ. Парадигма структурного программирования плавно сменилась объектноориентированным подходом. Последний, копируя восприятие действительности обычным человеком, достаточно прост для изучения и применения, хотя конкретные реализации могут быть достаточно сложными, в первую очередь из за огромного разрыва в уровнях абстракции объектов, доступных для обработки человеком и типичной цифровой машиной. Попытки создания вычислителей более высокого уровня пока не принесли результатов. Многие из нас были свидетелями взлёта и падения амбициозного японского проекта ЭВМ пятого поколения, а также многочисленных нейрокомпьютеров. Впрочем, развитие нанотехнологий и других передовых отраслей современной науки в самом ближайшем будущем может привести к гигантским изменениям в области методов вычислений, а пока мы все ещё используем компьютеры, работающие по старому доброму принципу машины Тьюринга и всё ближе подходим к той черте, когда наши представления о работе программы и её реальное функционирование всё больше и больше описываются формулами теории вероятностей. Это уже нашло своё отражении в недавно появившемся термине **«достаточно хорошее ПО»**. Обратите внимание, что субъективно соответствие ожиданий пользователя программы и функционирования ПО, оценивается как качество последнего. Возвращаясь к утверждению, что виной такой ситуации является сложность программных систем, можно сделать вывод, что высокое качество программы – это победа над её

сложностью, что ещё раз подтверждает народную мудрость: чем проще, тем надёжнее. Конечно, не следует ударяться в примитивизм, но современное ПО должно стать ясным, а его работа – очевидной для пользователя.

Методики обеспечения качества

Что может обеспечить качество в современных условиях разработки ПО? Ответов на это вопрос может быть много, однако, когда программы создаются уже не гениальными программистами-одиночками, а коллективами состоящими из десятков и даже тысяч разработчиков существует единственный путь – глубокая методическая проработка, планирование и чёткое управление процессом создания программ. На это непосредственно указывают исследования американского рынка программных средств. Как показывает статистика, только порядка четверти всех ИТ проектов достигают успеха, большая же часть не укладывается ни в бюджет, ни в сроки, что, как было выяснено, является результатом в первую очередь непродуманной политики управления качеством программных продуктов.

Осознание мировым компьютерным и финансовым сообществами непосредственной связи между успешностью производства программных систем и методологией процесса производства привело к появлению на западе новой индустрии – **Software Quality Assurance** – что на русский язык можно перевести как «Обеспечение качества ПО». Методики гарантирования качества могут применяться на всех этапах производства ПО. По заведённой в нашей стране традиции, этим занимаются Отделы технического контроля, проверяющие соответствие произведённой продукции различным нормативным актам, ГОСТам, Техническим условиям, Техническим заданиям. Однако в результате многолетних исследований было выяснено, что наибольший эффект при создании программных средств даёт не пост контроль создаваемой продукции, а проведение мероприятий по контролю качества на всех этапах создания ПО, начиная с момента замысла нового программного продукта, и заканчивая его внедрением у заказчика. Такой подход к производству существенно снижает общие затраты на производство, риски неуспеха проекта в целом и вселяет уверенность в возврате сделанных инвестиций и получении прибыли. В результате все остаются в выигрыше – разработчик может привлечь больше средств под более выгодные проценты, а заказчик получает качественный продукт в установленные сроки. Полезность такого подхода стала настолько очевидной, что методики обеспечения качества были оформлены в виде стандартов ISO, а, в последствии, ГОСТ. Однако даже при наличии стандартов серии ISO 9000 остаётся вопрос об их внедрении в повседневную практику. Надо сказать, что стандарт определяет весьма общие подходы к процессу обеспечения качества и фактически не содержит детальных инструкций по их воплощению в жизнь. Выпущено довольно много литературы, толкующей стандарт ISO 9000 и призванной помочь ИТ специалистам пройти сертификацию, но в основном она издаётся на английском языке и малодоступна широкой российской общественности. Приходится признать, что современные методики обеспечения качества с трудом пробивают себе дорогу к применению в отечественной программной индустрии.

Rational Unified Process

В последние несколько лет ситуация с методиками обеспечения качества ПО начинает изменяться к лучшему, несмотря на общее ухудшение ситуации с качеством в ИТ индустрии. Усилиями **Object Management Group** (OMG) созданы методики объектного моделирования и разработки программных систем, повлекшие широкий резонанс в компьютерном мире. Практически все ведущие производители программного обеспечения поддержали инициативу OMG и взяли её на вооружение. Основой методики моделирования стал **Unified Modeling Language** (UML), язык графической нотации объектных моделей. Надо сказать, что наличие только одного лишь языка UML, хоть и играет огромную положительную роль, но всё же является недостаточным. Разработчикам по-прежнему приходится самостоятельно придумывать способы использования UML для повышения эффективности своего труда. Чтобы разрешить данную проблему в корпорации Rational была разработана новая методика создания программных систем. Она получила коммерческое название **Rational Unified Process** (RUP). Главным отличием от существующей технологии разработки ПО методом «водопада» стала её ориентация на использование объектного проектирования с помощью UML и применение многочисленных коротких итераций на каждом этапе разработки ПО, что позволяет избежать серьёзных ошибок при проектировании и программировании. RUP может использоваться и без

применения UML, однако их совместное использование даёт наиболее плодотворные результаты. Благодаря высокой проработанности и наличию конкретных рекомендаций по проведению всех этапов производства ПО, указанная методика становится всё более и более популярной среди разработчиков программ среднего и большого уровня сложности.

Аутсорсинг тестирования

Неотъемлемой частью любой технологии обеспечения качества, помимо методических рекомендаций, является проверка и тестирование созданного программного продукта. Как известно, пока ещё не изобретены полноценные системы автоматизированной генерации программного кода, т.е. программы по прежнему создают люди, которым свойственно ошибаться. Зачастую невысокая профессиональная подготовка специалистов, обусловленная высокими темпами развития технологий и отставанием мероприятий по переобучению, и несоответствие условий труда принятым нормам создают прекрасные условия для возникновения дефектов в программах. Выявить их может только последующее тестирование на соответствие ПО заданным спецификациям. Как водится, эту работу поручают тем же программистам, что и создавали программу. Безусловно, это необходимо, но по настоящему качественное тестирование может провести только профессионал, специально обученный для выполнения подобного рода работ. У программиста обычно складывается определённое мнение о том, как должен работать созданный им код, а многие детали и пограничные случаи упускаются из виду. Поэтому крупные софтверные компании организуют у себя специальные отделы тестирования ПО, призванные обеспечить выявление дефектов и проводить оценку общего качества ПО. Как показывает опыт зарубежных компаний, такой подход дополняется **аутсорсингом** тестирования и обеспечения качества, особенно по сложным видам тестов, что вызвано экономической нецелесообразностью содержания в отделе тестирования большого числа высокооплачиваемых специалистов. Не удивительно, что по этому пути пошла и компания Informix/Russia для проведения специализированного тестирования продукта Informix - Gateway to the Future.

Informix Direct Driver for Clipper

Является составной частью миграционного продукта Informix - Gateway to the Future и позволяет переносить унаследованные приложения, созданные для платформы Clipper на СУБД Informix. Система компонентов доступа к данным Informix Direct Driver для Clipper позволяет работать с реляционной СУБД Informix версий 7.30 и выше из приложений, написанных на языке Clipper версии 5.3. Кроме доступа к данным РСУБД Informix с помощью стандартных средств Clipper, в Direct Driver также существуют функции выполнения SQL-запросов из Clipper-приложений. Взаимодействие с сервером базы данных Informix-Dynamic Server обеспечивает специальный сервер промежуточного уровня (middleware), написанный на языке Informix-ESQL/C. Сервер выполняет все операции по подключению пользователей к серверу, преобразованию ISAM-запросов клиентских рабочих станций в SQL-запросы, преобразованию информации в формат данных Clipper и возвращению данных клиентским приложениям. Со стороны клиентского приложения на Clipper Informix Direct Driver реализован в виде библиотеки замещаемого драйвера базы данных, который преобразовывает обращения к стандартным функциям работы с данными в запросы к промежуточному серверу и возвращает в прикладной программе полученные результаты, а также коды ошибок, используя стандартный API Clipper для работы с ошибками. Сетевые коммуникации реализованы с помощью протокола IPX/SPX.

Целью тестирования являлось измерение характеристик производительности IDD, проведение сравнительного анализа производительности IDD и стандартного драйвера базы данных Clipper с БД в формате DBF, а также выявление узких мест в ПО, снижающих как общую, так и пиковую пропускную способность системы.

Методика проведения тестов

Как уже отмечалось выше, создание высококачественного ПО в современных условиях невозможно без методической поддержки. Специалисты компании Amphora Quality Technologies обладают большим опытом использования методики Rational Unified Process для проведения тестов и обеспечения качества программных систем. Тестирование IDD

выполнялось в соответствии с требованиями RUP и сопровождалось оформлением соответствующей документации и моделей UML.

Первым этапом было проведено создание детального плана работ с перечислением проводимых тестов, методов измерения, получаемых результатов, а также материальных и людских ресурсов, необходимых для проведения тестирования. Была выявлена необходимость проведения следующих категорий тестов:

- **Интегральные тесты:** данная категория тестов предназначена для выяснения пропускной способности системы (её производительности) как единого целого. Интегральные тесты имеют своей целью измерить характеристики производительности системы и провести сравнительный анализ для двух платформ БД – IDD и DBF. Тесты включают измерение производительности выполнения базовых функций доступа к данным: SEEK, SKIP.
- **Корреляционные тесты:** данная категория тестов предназначена для подтверждения гипотезы о возможности независимого тестирования компонентов комплексной клиент-серверной системы Informix Direct Driver с целью анализа распределения нагрузки между отдельными узлами и компонентами программного комплекса. Фактически корреляционные тесты подтверждают возможность создания программных «заглушек», позволяющих профилировать отдельные фрагменты программного кода. Данные тесты являются предварительными тестами перед тестами производительности отдельных компонент системы и эталонов, подтверждая или опровергая возможность применения последних для анализа распределения нагрузки между компонентами системы.
- **Эталонные тесты:** данная категория тестов предназначена для получения информации о теоретически предельном быстродействии программного комплекса Informix Direct Driver и составляющих его компонент в рамках используемых при тестировании программно-аппаратных средств. Как правило, данные тесты направлены на определение предельной пропускной способности отдельных узлов и компонент системы. Например, в случае с тестированием сетевого слоя клиент-сервер в качестве эталона используется простой эхо сервер, позволяющий определить предельную пропускную способность сети в терминах транзакций/сек. для используемой аппаратной конфигурации.
- **Компонентные тесты:**
    - **Тесты клиентского ПО IDD:** данная категория тестов предназначена для измерения времени, затрачиваемого клиентскими рабочими станциями в процессе работы с БД с помощью IDD. На рабочих станциях пользователей функционирует 3 компонента: Clipper DD, замещаемый драйвер доступа к данным IDD для Clipper и SPX-клиент IDD для DOS.
    - **Тесты коммуникационного слоя IDD:** данная категория тестов предназначена для измерения времени, затрачиваемого IDD в процессе работы с БД на передачу информации по сети. Данные тесты не учитывают время пре- и пост-обработки клиентского запроса, как на клиентской станции, так и на сервере.
    - **Тесты серверного ПО:** данная категория тестов предназначена для измерения времени, затрачиваемого IDD в процессе работы с БД на обработку запросов сервером промежуточного уровня и SQL-сервером IDS. В процессе тестирование проводится профилирование, позволяющее получить информацию о распределении времени обработки между сервером БД и MW.

Первым этапом тестирования являлось проведение **Интегральных** тестов. Фактически созданная тестовая конфигурация эмулировала работу небольшого подразделения организации или рабочей группы с централизованной базой данных. Затем, перед началом серии экспериментов с целью анализа распределения нагрузки между узлами системы и поиска узких мест был проведен тщательный анализ архитектуры построения IDD, а также изучение исходных текстов замещаемого драйвера базы данных для Clipper, SPX-клиента и Middleware для Windows NT.

Вторым этапом тестирования являлось проведение серии **Корреляционных** тестов, анализ результатов которых показал, что разработанная методика независимого тестирования компонент IDD, профилирования и сравнения производительности компонент с «эталонами» дает корректные результаты и может применяться.

Непосредственно вслед за этим, в рамках третьего этапа тестирования, были произведены **Эталонные** тесты для получения «идеальных» значений производительности компонент для используемых аппаратно-программных средств.

Четвертым этапом являлось проведение тестов **клиентского и серверного ПО,** а также тестов **коммуникационного слоя.** Программное обеспечение для тестов данного вида было создано на основе анализа исходного текста IDD и выделения подмножества процедур и функций, профилирование времени выполнения которых позволяет точно разграничить функциональные компоненты системы. Другими словами, было произведено разбиение исходного кода по выполняемой функциональности.

По завершении этой работы были созданы программные модули – заглушки, которые позволяют производить тестирование отдельных компонент системы (клиентской, серверной и коммуникационной) без вовлечения в работу функциональности других компонентов IDD.

Все виды тестов проводились на **«стабилизированной»** системе. Критерием достижения стабилизации являлось отличие получаемых результатов на нагрузочных серверах друг от друга не более чем на 2% при проведении серии последовательных тестов. Лишь затем проводились рабочие тесты.

Результаты тестов были предоставлены заказчику в обработанном виде, а именно в виде усредненных значений округленных до реально полученной степени точности. Значения для каждого теста были получены путем статистического усреднения данных 10 экспериментов, выполненных на «стабилизированной» системе. Погрешность измерений выполненных тестов не превысила 1,5%.

Тестовый стенд

Тестирование проводилось на специально собранном для этих целей испытательном стенде. Испытуемый серверный компонент драйвера, как и сам сервер Informix работали под управлением операционной системы Windows NT 4.0 Service pack 6. В качестве сервера использовалась двухпроцессорная серверная платформа Intel с двумя Pentium III 500. В сервере было установлено 256 мегабайт оперативной памяти с коррекцией ошибок и дисковый массив RAID 1, собранный с использованием контроллера Mylex. Сервер был подключён к специально собранной для тестирования локальной сети Fast Ethernet на скорости 100 мбит/сек с помощью хаба 3com. При проведении сравнительных тестов данный сервер использовался и в качестве файлового для хранения базы данных в виде файлов формата DBF. В последнем случае серверный компонент драйвера и СУБД Informix не запускались, все ресурсы компьютера выделялись для эффективной работы последнего в качестве файлового сервера, для чего также проводилась специальная настройка операционной системы.

Важным фактором получения точных и достоверных результатов нагрузочных тестов является правильная эмуляция нагрузки. Нагрузочный сервер должен обеспечивать одновременную работу десятков и сотен одновременно работающих сессий с тестируемым сервером и не вносить при этом существенной погрешности в измерения в связи с недостаточностью собственных ресурсов. Было принято решение использовать в качестве серверов нагрузки четырёх однопроцессорных компьютеров Pentium II 400, снабжённых 64 мегабайтами оперативной памяти, 8 гигабайтными дисками и сетевым интерфейсом Fast Ethernet каждый. Результаты тестов подтвердили правильность сделанного выбора, поскольку даже в самом сложном режиме тестирования загрузка ресурсов нагрузочных серверов не превышала 65 процентов. На компьютерах были установлены три операционные системы – DOS 6.22, Windows 98, Windows NT 4.0. Тесты проводились с использованием каждой операционной системы для выявления различий функционирования.

Для проведения тестов была создана специальная тестовая база данных, содержащая таблицы со структурой, типичной для систем автоматизации бухгалтерской и банковской деятельности. Таблицы содержали до 42 информационных полей и до 13 составных индексов.

Информационное наполнение базы данных было выполнено с помощью генерации псевдослучайных последовательностей символов и чисел с равномерным распределением. Основные тесты проводились на таблицах, содержащих порядка 200000 записей.

Итоги тестирования

Результаты тестов показывали, что платформа Informix обладает **существенным преимуществом перед DBF при любом числе одновременно работающих пользователей**. Более того, зависимость времени, необходимого для доступа к информации от количества пользователей для платформы DBF носит экспоненциальный характер, что не позволяет говорить о возможности реальной интенсивной работы более чем 3-4 пользователей и соответственно лишает смысла проведение сравнительного тестирования этого драйвера для большего числа пользователей. Результаты тестов наглядно иллюстрируют диаграммы 1 и 2.

В тоже время, при увеличении нагрузки, т.е. числа одновременно работающих пользователей падение производительности системы, использующей IDD, носит линейный характер с коэффициентом роста времени отклика существенно меньшим 1, что позволяет говорить о высоком уровне готовности решения к работе в многопользовательской среде.

Необходимо подчеркнуть, что при тестировании доступа к таблицам в формате DBF их открытие производилось в многопользовательском режиме, а не в монопольном, поскольку в последнем случае радикально меняется характер доступа к информации в файле (включается эффективное файловое кэширование на клиентской станции). Однако последний случай не предоставляет для нас интереса, поскольку цели тестирования для обеих платформ предполагают проведение тестирования в условиях, близких к реальной эксплуатации ПО IDD, с большой долей вероятности предполагающей многопользовательскую работу.

По результатам компонентного тестирования были построены диаграммы распределения времени выполнения запросов между компонентами системы. Из диаграммы 3 видно, что большая часть времени при обработке запроса тратится SQL сервером. Это хороший результат, показывающий высокую эффективность работы драйвера базы данных (конечно при условии формирования им оптимальных запросов к СУБД). Особенно важно отметить крайне низкое время, которое тратится на сервере промежуточного слоя. **Исполняемый им код очень эффективен.**

После завершения комплекса тестов, предусмотренных планом тестирования, было принято решение провести дополнительный анализ исходного текста драйвера на предмет изучения транспортного протокола прикладного уровня, используемого для обмена данными между клиентом и сервером промежуточного уровня. Было произведено так называемое тестирование «белого ящика».

Анализ выявил наличие возможности оптимизации поведения системы при выполнении операций добавления и модификации записей в таблице БД со сложной структурой многосегментного индекса. В этих условиях возрастает нагрузка на процессор клиентской станции и существенно увеличивается сетевой трафик. Архитектурное построение драйвера дало возможность предположить, что часть нагрузки целесообразно перенести с рабочей станции на сервер промежуточного слоя, что может существенно ускорить работу указанных операций, поскольку снизит сетевой трафик, а обмен по сети является одним из наиболее медленных мест в системе.

Для проверки выдвинутой гипотезы были проведены дополнительные тесты с модифицированными версиями как клиентской части драйвера, так и серверной. Это было сделано с целью изменения логики работы программы в плане формирования значений ключевых полей на сервере промежуточного уровня. Как следствие, отпала необходимость передавать значения дополнительных индексных полей по сети.

Достигнутые результаты наглядно демонстрируются диаграммой 4. Видно, что выигрыш в производительности на операции добавления новых записей достигает **27%**. Вместе с этим необходимо отметить, что время, затрачиваемое коммуникационным слоем IDD на передачу

информации по сети уменьшилось на **69%**. Это говорит о существенном уменьшении сетевого траффика, что имеет очень важное значение с системо-технической точки зрения.

Подводя итог анализа результатов компонентного тестирования, можно сказать, что тесты показали в целом высокую производительность работы драйвера на операциях доступа к данным (SEEK, SKIP). По оценкам наших специалистов, существует лишь небольшая вероятность увеличения производительности системы IDD на данных операциях без существенного изменения заложенных при их реализации концепций.

Диаграмма 1

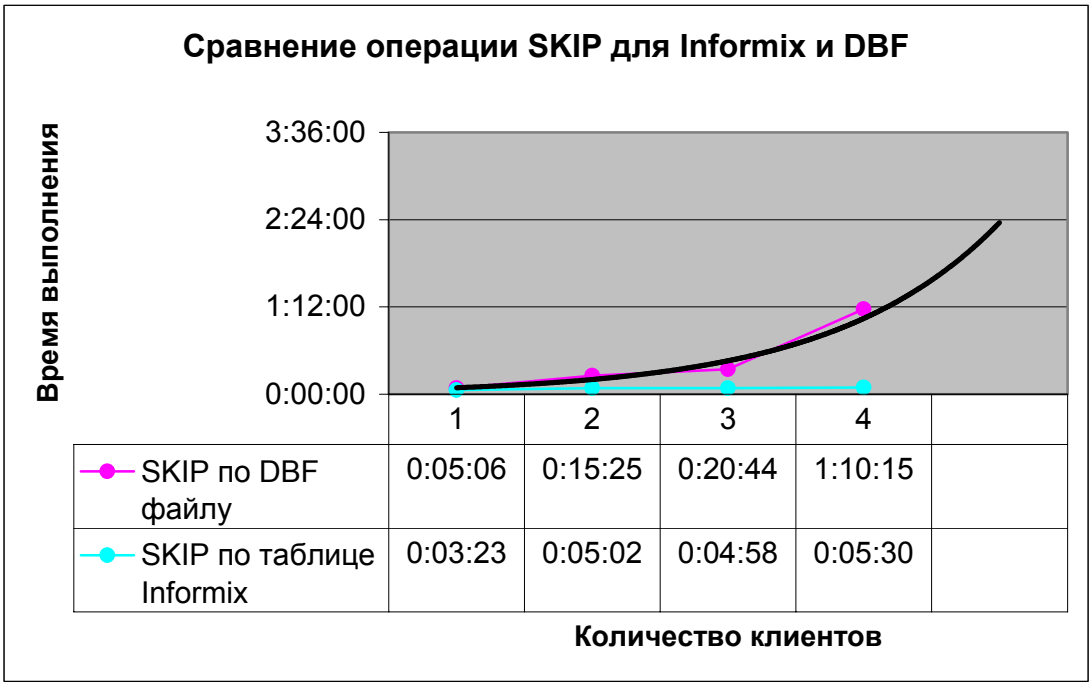


Диаграмма 2

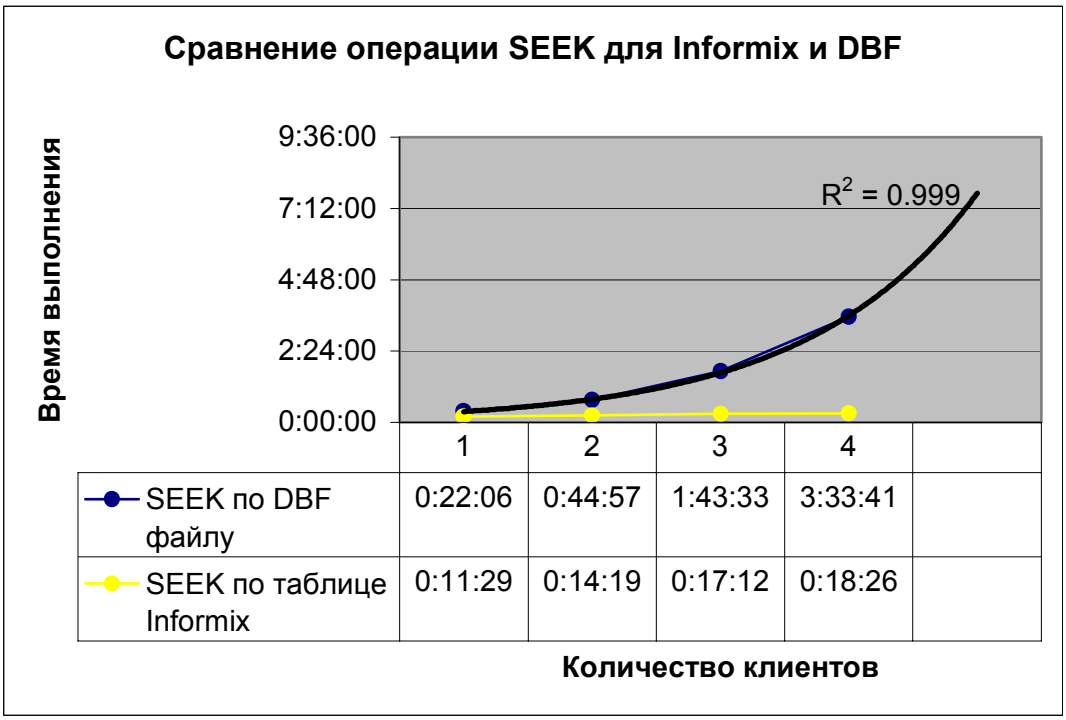


Диаграмма 3

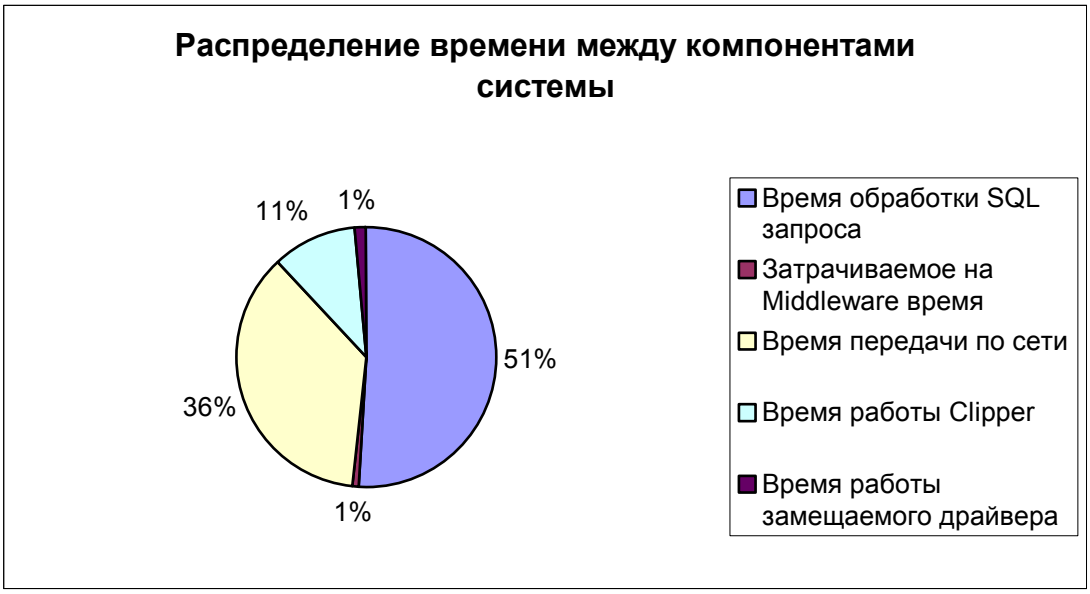
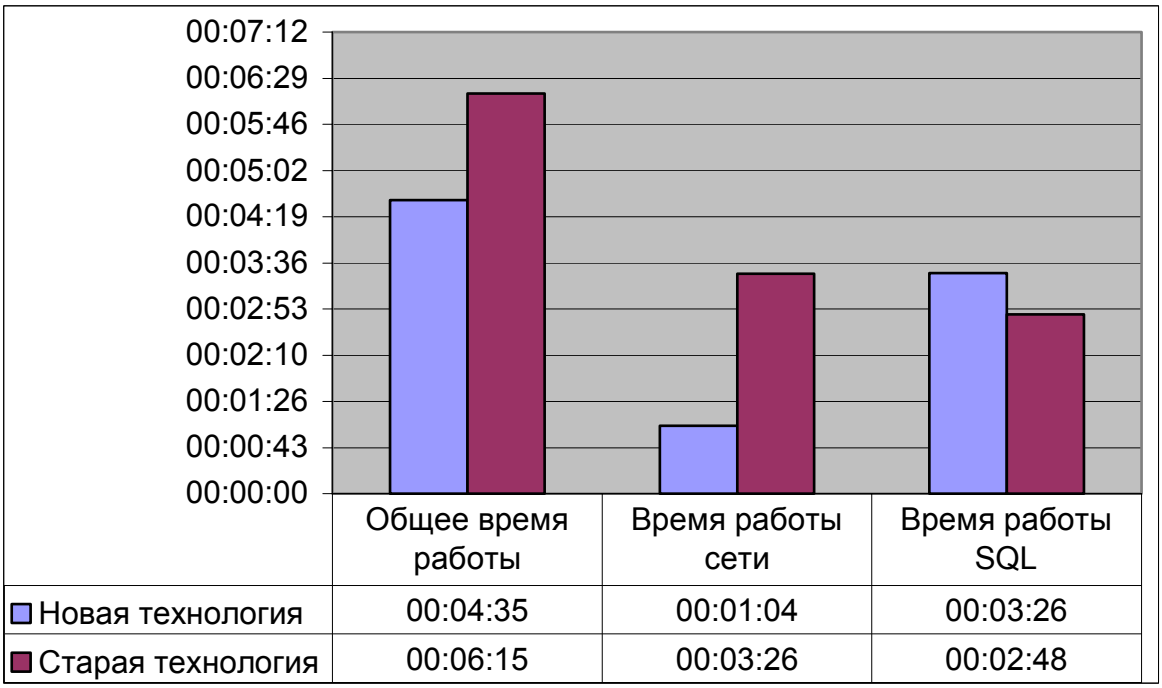


Диаграмма 4



	Общее время работы	Время работы сети	Время работы SQL
Новая технология	00:04:35	00:01:04	00:03:26
Старая технология	00:06:15	00:03:26	00:02:48