УДК 519.685

ОБОБЩЕННАЯ СТРУКТУРА УДАЛЕННОГО РЕПОЗИТОРИЯ ЯЗЫКА ПИФАГОР

Ермаков В. Н.
научный руководитель д-р техн. наук Легалов А. И.
Сибирский федеральный университет

Современный этап развития вычислительной техники характеризуется увеличением вычислительной мощности за счет экстенсивного наращивания количественных показателей ее структурных элементов с одновременным уменьшением технологических норм их производства и снижением ценовых характеристик.

Наличие нескольких вычислительных ядер в портативной вычислительной технике уже давно стало общепринятой нормой, не говоря уже о домашних персональных компьютерах и тем более о промышленных или научных центрах обработках данных, где их количество может доходить до нескольких сотен и даже тысяч на одну печатную плату [1].

Однако используются эти вычислительные ресурсы зачастую крайне редко, т.к. требуют специального программного обеспечения, поддерживающего параллельные вычисления. А это в свою очередь существенно увеличивает требования к квалификации программистов, которые разрабатывают эти программы, потому как современные языки программирования поддерживают многопоточность с помощью отдельных библиотек и расширений, возлагая обязанности по планированию вычислений в большей мере на программиста. Это в свою очередь вызвано тем обстоятельством, что данные языки создавались в то время, когда большое количество вычислительных ядер было исключением, а не правилом. Помочь разработчику в решении вопроса распараллеливания программ может язык функционально-потокового программирования Пифагор [2].

Большинство языков программирования ориентируется на текстовое представление программ. При этом программа обычно размещается в нескольких файлах, каждый из которых может нести различную ролевую нагрузку, зачастую определяемую спецификой построения конкретного языка программирования. Внутренняя структура этих файлов также бывает различной и зависит как от специфики языка, так и от особенностей использования в нем того или иного файла.

Для удобства разработки данные файлы помещается в репозиторий (хранилище данных), обеспечивая поддержку совместной работы разработчиков над проектом. Взаимодействие разработчиков с репозиториями может осуществляться различными способами с использованием как графического, так и программного интерфейса. При этом одним из важных является способ организации совместной работы и связанное с ним сопровождение версий (хранение нескольких версий одного и того же файла, возвращение к более ранним версиям, определение разработчика, сделавшего то или иное изменение, и т.д.). В общем случае, все множество существующих систем управления версиями (СУВ) можно разделить на три типа: локальные (нет средств удаленного доступа), централизованные (клиент-серверные системы, имеющие одно общее хранилище), распределенные (каждый клиент - отдельное хранилище). Каждый из перечисленных типов можно характеризовать положительными и отрицательными чертами, являющихся следствием их архитектурных особенностей (Таблица 1).

Таблица 1. Сравнительная таблица типов систем управления версиями

Характеристики	Типы		
	локальные	централизованные	распределенные
Достоинства	автономность	необходимо хранить только рабочую копию	гибкость
		блокировка файла или группы файлов	автономность
		слежение за определённым файлом или группой файлов	
		единая сквозная нумерация версий системы и/или файлов, в которой номер версии монотонно возрастает	
		локальная работа пользователя с отдельной, небольшой по объёму выборкой	
Недостатки	отсутствие удаленного доступа	необходимость доступа к центральному серверу	необходимость хранения всей истории
	необходимость хранения всей истории версий		нет блокировки файла или группы файлов
			нет возможности слежения за определённым файлом или их группой
			нет единой сквозной нумерации версий системы и/или файлов, в которой номер версии монотонно возрастает
			работа пользователя со всей историей версий, а не ее выборкой

Каждая из существующих централизованных и распределенных систем поддерживает один или несколько сетевых протоколов передачи данных, наиболее часто встречаемыми из которых являются: HTTP (в 60% СУВ из 30 наиболее распространённых), HTTPS (26%), Email (27%) и собственные протоколы, работающие через SSH (50%).

Так же выделяют класс программ, называемых хранилищем данных - программную подсистему, сочетающую в себе функции системы управления версиями, поисковой машины и системы управления базами данных (СУБД). В функционал хранилища данных входит:

- хранение структурированных и неструктурированных данных,
- полнотекстовый поиск,
- версионирование,
- поддержка транзакций,
- наблюдение за контентом.

Опираясь на эти и другие характеристики современных программных продуктов, было принято решение о необходимости формирования новых методов хранения исходных текстов и различных внутренних представлений, формируемых в ходе разработки, трансляции и выполнения функционально потоковых параллельных программ, написанных на языке Пифагор. Для достижения данной цели решаются следующие задачи:

- на основе анализа особенностей языка программирования формируется внутренняя структура хранилища (репозитория) функционально-потоковых параллельных программ;
- рассматриваются основные функции, обеспечивающие работы с репозиториями;
- анализируются возможности совместного использования нескольких репозиториев при создании одной программы.

Среди основных требований к репозиторию можно выделить:

- децентрализованное хранение данных, при котором каждый из разработчиков может создавать свой репозиторий;
- поддержка коллективной разработки библиотек и версий отдельных функций;
- автоматический учет зависимостей выбранной функции;
- удаленный доступ;
- разграничение прав доступа как к отдельным элементам репозитория, так и ко всему в целом;
- осуществления поиска внутри репозитория;
- возможность полного или частичного исполнения выбранной функции на стороне репозитория;
- графический интерфейс пользователя.

Приняв во внимание описанные выше требования к разрабатываемому репозиторию, а так же проанализировав взаимосвязь данного модуля с другими подсистемами разрабатываемого программного комплекса, пришли к заключению о необходимости создания клиент-серверного программного обеспечения класса хранилища данных. Поддерживающего распределенную модель хранения данных и сочетающего в себе подсистемы: управления версиями, управления базами данных, поиска и предоставления вычислительных ресурсов как сервиса (Рисунок 1).

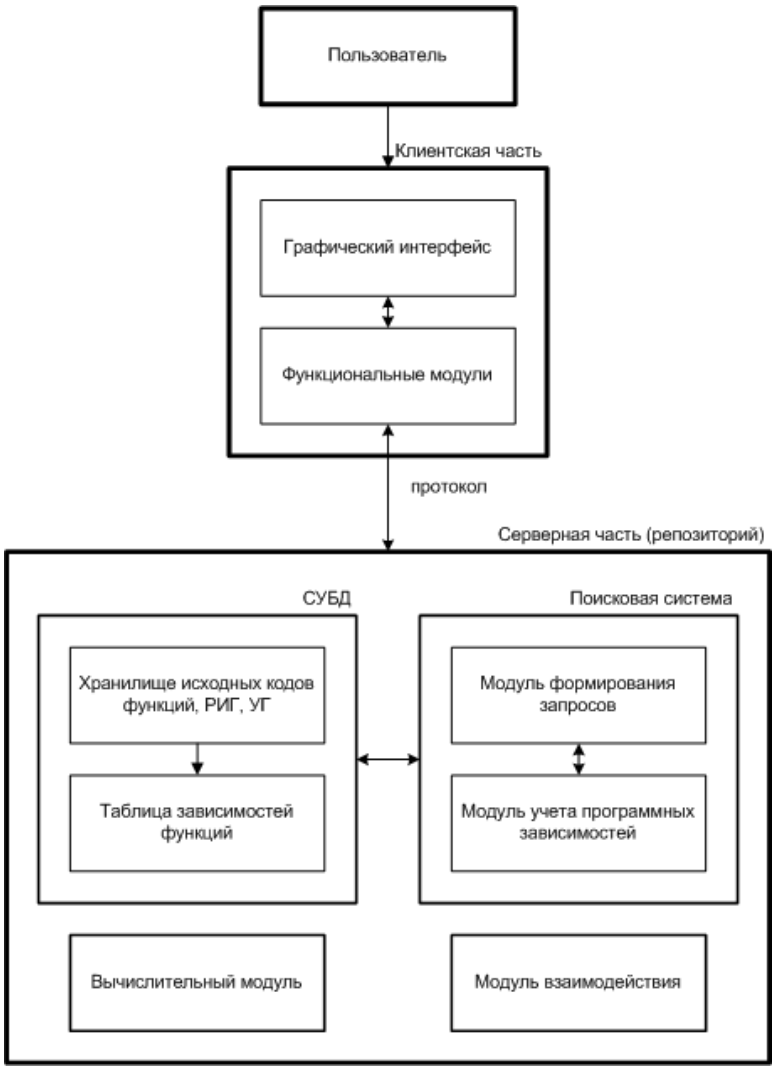

Рисунок 1 – Структура репозитория

Работа выполняется в рамках ФЦП НПП № 14.A18.21.0396 «Инструментальная поддержка архитектурно-независимой разработки параллельных программ на основе функционально-потоковой парадигмы параллельного программирования».

Список использованных источников:

1. Makino J. GRAPE-DR and Next-Generation GRAPE. – Japan: Center for Computational Astrophysics and Division Theoretical Astronomy National Astronomical Observatory of Japan, 2009. – 51 с.

2. Легалов А.И. Функциональный язык для создания архитектурно-независимых параллельных программ. / А.И. Легалов // Вычислительные технологии, № 1 (10) – 2005. - С. 71-89.