1. HealthKit Depelover guidance [Электронный ресурс]. - Электронные данные. Режим доступа: https://developer.apple.com/documentation/healthkit
2. Harvard Health Publishing [Электронный ресурс]. - Электронные данные. Режим доступа: https://www.health.harvard.edu
3. Human Interface Guidelines [Электронный ресурс]. - Электронные данные. Режим доступа: https://developer.apple.com/design/human-interface-guidelines/

БАЛАНСИРОВКА НАГРУЗКИ МЕЖДУ УЗЛАМИ ПРИ ПОСТРОЕНИИ РАСПРЕДЕЛЕННЫХ СИСТЕМ

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Вабищевич Е. Г..

Таборовец В.В. – к.т.н.,доцент

Работа посвящена вопросам исследования существующих способов построения распределенных систем с целью создания базовой архитектуры распределенной системы обработки запросов.

Распределенная система — это набор независимых компьютеров, представляющийся их пользователям единой объединенной системой [1]. Среди эмпирических свойств распределенной системы можно выделить следующие:

1) устойчивость системы при выходе из строя одного или нескольких элементов;
2) возможность изменения конфигурации системы в процессе работы;
3) отсутствие полных знаний о системе в одном конкретном элементе системы.

В процессе исследования распределенных систем были рассмотрены следующие архитектуры: архитектура клиент-сервер, архитектура peer to peer, кластерная архитектура.

Кластеры распределения нагрузки – способ организации программного обеспечения и системного оборудования, при котором происходит распределение запросов через один или несколько входных узлов. После этого из входных узлов запросы перенаправляются на обработку в вычислительные узлы. Данный подход обладает следующими преимуществами:

- возможность горизонтального масштабирования;
- равномерное распределение нагрузки между узлами;
- обеспечение высокой доступности системы. Данный пункт достигается благодаря возможностям первых двух пунктов.

В результате рассмотрения принципов построения распределенных систем была разработана следующая базовая архитектура.

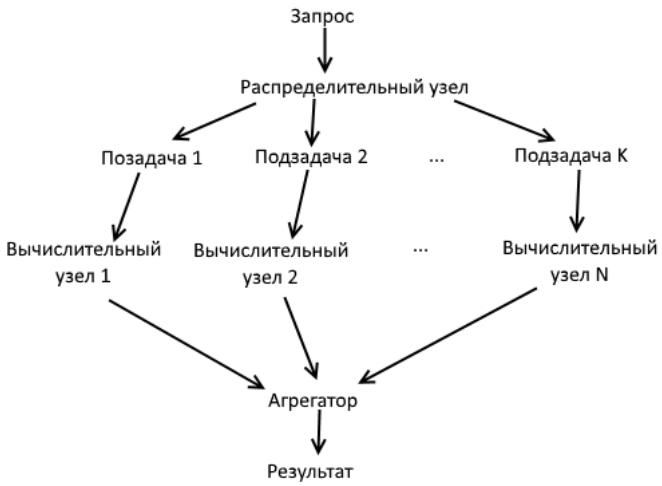

Рис. 1 - Базовая архитектура распределенной системы

Описание рисунка 1: в систему приходит запрос → запрос обрабатывается распределительным узлом систем, разбивается на подзадачи меньшего масштаба → каждая из подзадач отправляется на один из вычислительных узлов → вычислительный узел выполняет обработку подзадачи; результат выполнения отправляется агрегатору → агрегатор дожидается обработки всех подзадач запрос, собирает из них итоговый результат, отдает результат клиенту.

Список использованных источников:

1. Э. Таненбаум, М. Ван Стеен. Распределенные системы. Принципы и парадигмы // СПб.: Питер, 2003. 877 с.

ИСПОЛЬЗОВАНИЕ ЛЕКСИЧЕСКОЙ ОБФУСКАЦИИ VHDL-ОПИСАНИЙ ДЛЯ ВНЕДРЕНИЯ ВОДЯНЫХ ЗНАКОВ.

Видничук В.Н.

Белорусский государственный университет информатики и радиоэлектроники

г. Минск, Республика Беларусь

Иванюк А.А. – д.т.н., доцент

В статье описывается метод лексической обфускации VHDL- описаний, его основные положения а также метод внедрения водяных знаков в это описание. Демонстрируются множества символов используемых для обфускации описаний.

В наши дни быстрыми темпами растёт производство цифровой техники и в связи с этим растёт проблема пиратства. Для решения этой проблемы предлагается рассмотреть метод внедрения водяного знака в лексическую обфускацию VHDL-описаний.

Обфускация (от англ. obfuscate – делать неочевидным, запутанным, сбивать с толку) – широко известная методика защиты исходных кодов программ от обратного проектирования [1].

Основной целью лексической обфускации является затруднение понимания исходных кодов программы. Лексическая обфускация это один из многих способов затруднить понимания функционирования программы.

Частным случаем лексической обфускации является переименование идентификаторов VHDL-описания на похожие друг на друга. Название идентификатора можно представить в виде множества символов:

$$< Id >= \{[A..z] \cap [\ \ ..\ \ ] \cap [0..9]\}$$

где, идентификатор не должен начинаться с цифры и может состоять из латинских и кириллических символов. В связи с этим набор символов при выполнении метода переименования идентификаторов можно представить следующим множеством:

$$< Id' >= \{O, O', 0, Q, 1, l, I, j\},$$

в котором O' является кириллическим символом O и O латинская являются разными символами, однако в латинском языке могут использоваться диакрические знаки например: Í

$$< Di >= \{\acute{O}, \tilde{O}, \grave{I}\ \acute{I}, \quad , \quad , \quad , \quad , \hat{I}, \ddot{I}\},$$

при синтезе VHDL-описания данные символы приводятся к латинской O, однако имеют разные коды символов, поэтому при выполнении анализа лексически обфусцированного кода могут возникать ошибки и непонятности. Отсюда, следует возможность использования диакрических символов для формирования названий идентификаторов повышенной сложности.

Далее приведён пример использования диакрических символов для затруднения понимания кода: идентификатор OIOOIIOIIOIIOIÖ при синтезе будет равняться идентификатору OIOOIIOIIOIIOIO или OIOOIIOIIOIIOIOÓ, но при попытке найти идентификатор OIOOIIOIIOIIOIÖ с помощью поиска или сторонних программ найдётся только он сам. Следовательно, с помощью использования данных символов можно запутывать не только человека, но и программные средства анализа кода или, например, идентификатор OIO'OIIOIIOILÖ и идентификатор OIOOIIOIIOILÖ для человека будут выглядеть как идентичные, однако при синтезе они будут являться разными идентификаторами.

При использовании метода переименования идентификаторов следует знать, что все идентификатора должны быть похожи и отличаться друг от друга на 1 символ, в связи с этим существует возможность внедрения водяного знака в VHDL-описание путём выбора мест замены символов в соответствии с ключом водяного знака.

Данный метод можно применять несколькими способами, например, генерируется первичный идентификатор для переименования, состоящий из нулей и единиц, символ выбирается с помощью равномерного распределения. В итоге получается идентификатор вида: 100101001011 это делается для того, чтобы расстояние между двумя идентификаторами не превышало 1. После этого