электронный научно-технический журнал
ИНЖЕНЕРНЫЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51036. ISSN 2307-0595

Применение языков программирования с возможностью замены кода для разработки распределённых систем

 # 03, март 2016

Богачёв А. Ю.

УДК 004.4

Россия, МГТУ им. Н.Э. Баумана
alexbog21@yandex.ru

Введение

В настоящее время значительно увеличилось разнообразие активно используемых программно-аппаратных платформ. Эти платформы зачастую имеют большие различия в вычислительных возможностях, и средствах взаимодействия с пользователем. Это привело к общему усложнению процесса разработки программных систем, их интеграции и сопровождения.

Различия в платформах препятствуют созданию единой кодовой базы, что в свою очередь увеличивают сроки и стоимость разработки, создают условия для появления дополнительных программных ошибок. Отсутствие своевременных исправлений программного обеспечения, а также сложность длительного сопровождения и выпуска исправлений программных продуктов отрицательно сказываются на безопасности и надёжности программного обеспечения.

Таким образом, требуется разработка программных средств, позволяющих комплексно решать проблемы взаимодействия, актуальности и переносимости компонентов программного обеспечения.

1. Постановка задачи

Одна из основных проблем, для решения которой необходимо создание специализированных средств разработки, связана с созданием программного обеспечения для микроконтроллеров распределённых систем сбора данных и управления оборудованием, для которых характерны следующие особенности:

- Малый объём памяти, который может значительно ограничивать функциональность отдельного узла системы, и вынуждая разработчиков вводить дополнительные узкоспециализированные узлы, что также может отрицательно сказаться на унификации компонентов.

- Необходимость внесения изменений в алгоритмы работы системы, без её остановки, в условиях, когда возможно «горячее» добавление и отключение устройств в системе, а также устройств, подключённых к отдельным контроллерам.
- Эксплуатация различных протоколов сетевого взаимодействия, например: ModBus, RS-485, CAN-шина, Ethernet, Bluetooth, в том числе и одновременно. Необходимость обеспечения прозрачной адресации и передачи данных, между программами системы.

Другая проблема, требующая создания специализированных средств разработки программного обеспечения связана с решением задач, программных систем таких как:

- Переносимость программного обеспечения между различными платформами.
- Создание общей кодовой базы, для разных вариантов клиент-серверного взаимодействия, например, для клиентских приложений и веб-приложений (рис. 1).
- Возможность реализации своевременного или незамедлительного выпуска исправлений программ и обновлений безопасности независимо от используемой программной платформы.
- Возможность выпуска обновлений и исправлений для ранее выпущенных версий программного обеспечения.

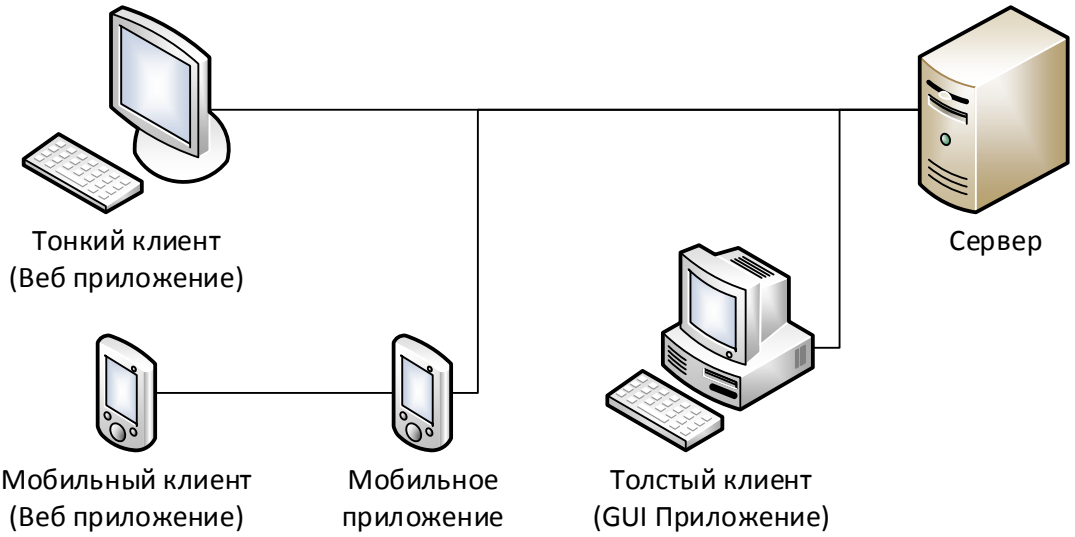

Рис.1. Актуальные варианты приложений клиентов

Таким образом, решение указанных задач связано с решением задачи обеспечения взаимодействия компонентов распределённой вычислительной системы, с возможностью распределения вычислительных задач между узлами вычислительной системы, изменения алгоритмов работы системы без её остановки, в соответствии с текущими потребностями системы.

2. Существующие пути решения задачи

Для решения поставленной задачи необходимо совместное использование механизма обмена данными через сообщения и возможности внесения изменений в программный код системы без её остановки.

Возможность обмена данными через механизм сообщений может быть реализована как в самом языке программирования, так и через специализированное промежуточное программное обеспечение, для обработки сообщений (Message-Oriented Middleware).

Как правило, использование такого программного обеспечения для обмена данными через сообщения позволяет:

1) Прозрачно адресовать приложения системы и передавать данные между ними, абстрагируясь от сетевых протоколов;
2) Представить систему в виде множества независимых процессов, обменивающихся сообщениями, что позволяет обеспечить слабую связность компонентов системы и упрощение внесения изменений в работающую систему;
3) Разрабатывать систему в рамках сервис-ориентированной архитектуры.

Возможность замены кода может быть реализована как применением интерпретируемых языков, так и применением компилируемых языков использующих виртуальную машину [1]. При помощи механизма изменения программ, работающих в системе, без их остановки, возможно:

1) Централизованно хранить программный код системы, и распределять его по узлам системы в соответствии с их возможностями и назначением;
2) Преодолеть ограничения, связанные с малым объёмом памяти на определённом узле системы, путём удаления ненужного для выполнения текущей задачи программного кода и загрузки необходимого кода с другого узла системы в процессе работы;
3) Уменьшить размер клиентских приложений, путём хранения программного кода редко используемых функций приложения, на сервере, и возможностью загрузки этих функций с сервера, или удалённого их выполнения;
4) Повысить унификацию кода бизнес-логики для приложений, работающих на различных платформах, и в разных вариантах сетевого взаимодействия;
5) Снижать нагрузку на сервер, путём переноса вычислений на сторону клиента в соответствии с его возможностями.

Существующие в настоящее время технологии и языки программирования, так или иначе, реализуют часть функциональности необходимой для решения данной задачи. Однако у всех них имеются недостатки, не позволяющие использовать их для её решения.

Язык программирования Erlang, предназначенный для использования в распределённых системах [2,3]. Имеет собственные механизмы выполнения параллельных процессов и передачи сообщений, основанные на математической модели акторов, а также возмож-

ность замены программного кода во время работы. Его недостатками является малая распространённость, высокая сложность освоения, связанная с особенностями синтаксиса языка и с использованием парадигм функционального программирования. А также отсутствие возможности встраивания виртуальной машины в приложения, и ограничения в механизме замены программного кода.

Язык программирования LUA, с возможностью встраивания интерпретатора в программы написанные на других языках [4]. Рассматривает функции как объекты первого класса, и способны работать с ними как с переменными, а также может интерпретировать произвольный текст в процессе работы, что позволяет ему изменять программный код функций без остановки системы. Недостатки языка связанны с отсутствием механизмов передачи сообщений.

Объектно-ориентированный язык программирования Smalltalk, использующий механизм объектов и сообщений, и обладающий широкими возможностями изменения алгоритмов работы программ и компонентов среды выполнения (например, сборщика мусора) без остановки системы [5]. Недостатком языка является отсутствие поддержки распределённых и параллельных вычислений.

3. Предлагаемое решение задачи

Изучив недостатки существующих языков программирования, было принято решение о создании собственного языка программирования, в котором предполагается использовать механизм передачи сообщений между изолированными процессами и специальную виртуальную машину, позволяющую изменять программный код в процессе его выполнения.

Требования к разрабатываемому языку программирования:

- имеет поддержку механизмов передачи сообщений;
- однозначно транслируется и отображается в Байт-коде;
- позволяет работать с функциями как с объектами первого класса;
- обладает небольшой, но полноценной стандартной библиотекой;
- обладает простым для освоения, синтаксисом и логикой поведения.

3.1. Процессы

Решение задачи прозрачного обмена сообщениями между процессами, работающими на различных узлах системы, должно быть основано на применении математической модели акторов [6].

Для обеспечения простоты и удобства восприятия системы предполагается императивное описание процессов, в виде обработчика событий (Event Loop) состоящего из функций получения сообщений, их обработки, выполнения связанных с ними действий, выполнения собственных действий или ожидания сообщений.

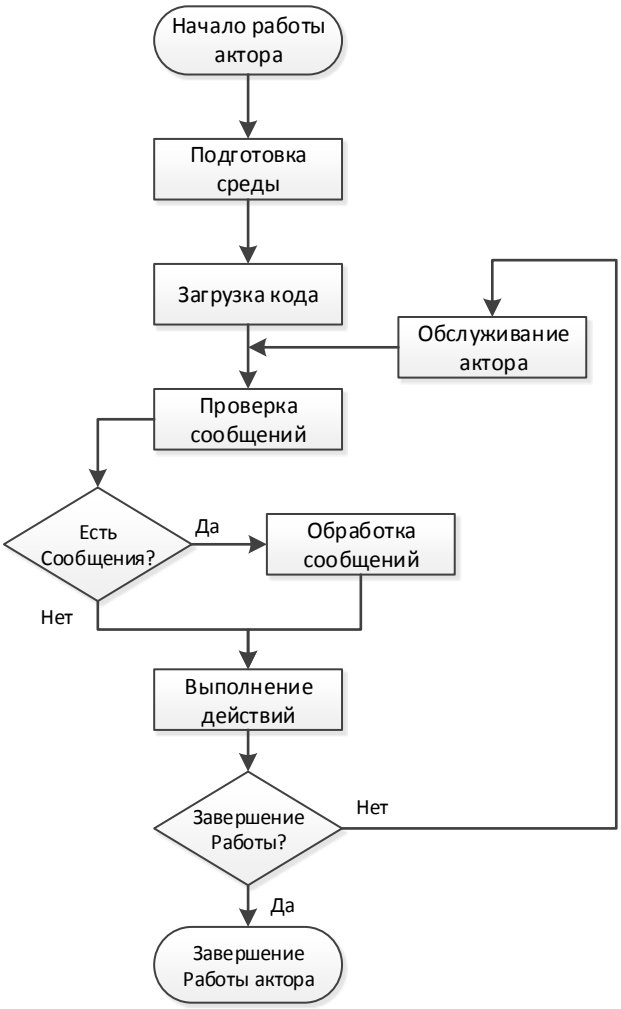

Рис. 2. Алгоритм работы процесса на основе Event Loop.

Применение императивного подхода к описанию позволяет просто и гибко программировать поведение каждого процесса, работающего в системе.

3.2. Хранение данных виртуальной машины

Обрабатываемые процессами данные должны хранятся в памяти вместе с метками (сигнатурами) определяющими тип хранимых данных, и флаги, определяющие применение коллекций.

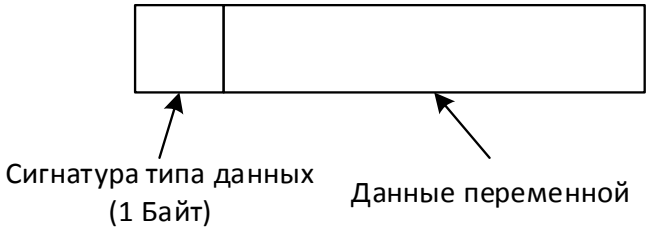

Рис. 3. Хранение переменных в памяти

В случае хранения структур данных, сигнатура должна состоять из ХЕШ функции от имени структуры и описания хранящихся в структуре переменных и вложенных структур.

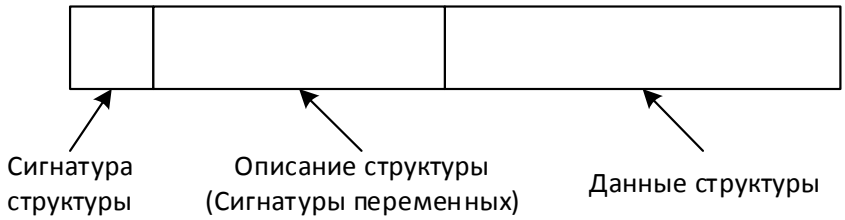

Рис. 4. Сигнатуры структур данных

Такой подход позволяет упростить процедуру передачи данных по сети, и необходим для работы механизма мультиметодов. При этом предполагаются незначительные накладные расходы (1 байт для переменной любого типа в том числе и включённой в структуру), и несколько байт для структуры (в зависимости от её сложности).

3.3. Операторы

Язык программирования должен иметь поддержку необходимых математических операторов, операторов структурного программирования, операторов для работы с виртуальной машиной.

Кроме этого в нем должна быть поддержка функций, вызываемых из сторонних библиотек, в том числе и из внешних подключаемых разделяемых библиотек.

Также, возможность внесения изменений в программный код работающей системы требует, чтобы виртуальная машина и язык программирования поддерживали перегрузку операторов с помощью механизма мультиметодов [7].

Механизм мультиметодов должен сравнивать типы данных в описании аргументов перегруженной функции, с фактическими типами, указанными в сигнатурах аргументов, в момент вызова функций, и выбирать подходящую функцию. При необходимости выбора функции при вызове мультиметода, производится поиск подходящей функции в соответствии с типами аргументов. Если одним из аргументов вызываемой функции является структура, то осуществляется быстрый поиск функции с использованием Хеша имени структуры. В случае если имеется несколько вариантов с подходящими Хешами производится полный поиск по сигнатуре структуры, описывающий её содержимое.

Заключение

Решение проблем разработки программного обеспечения, зачастую требует специальных узкоспециализированных средств. Рассмотренные в данной статье языки программирования, лишь частично подходят для решения поставленных задач. В связи с этим было принято решение о создании собственного специализированного языка программирования, реализующего модель акторов с применением императивного подхода к программированию акторов и возможностью обмена программным кодом между ними.

Данный язык должен стать простым и понятным инструментом для разработчиков, и эффективно решать задачи обеспечения взаимодействия компонентов распределённой

вычислительной системы, с возможностью распределения вычислительных задач между узлами вычислительной системы и последующего внесения изменений в эти системы в процессе эксплуатации.

Список литературы

[1]. Линда Дейли Полсон. Разработчики переходят на динамические языки // Открытые системы. 2007. Режим доступа: http://www.osp.ru/os/2007/02/4108153/ (дата обращения 1.02.2016)

[2]. Joe Armstrong. Erlang // Communications of the ACM. 2010. Vol. 53, no. 9. P. 68-75. DOI: 10.1145/1810891.1810910. Режим доступа: http://cacm.acm.org/magazines/2010/9/98014-erlang/fulltext (дата обращения 1.02.2016)

[3]. Чезарини Ф., Томпсон С. Программирование в Erlang = Erlang Programming. М.: ДМК Пресс. 2012. 488 с. — ISBN: 978-5-94074-617-1

[4]. Роберто Иерузалимски. Программирование на языке Lua. 3-е изд. М.: ДМК Пресс. 2014. 381 с. — ISBN: 978-5-94074-767-3

[5]. Кирютенко Ю.А., Савельев В.А. Объектно-ориентированное программирование. Язык Smalltalk. М.: Вузовская книга. 2006. 328 с. — ISBN: 5-9502-0097-7.

[6]. Agha Gul Abdulnabi. ACTORS: A Model of Concurrent Computation in Distributed Systems // DSpace@MIT. 1985. Режим доступа: https://dspace.mit.edu/handle/1721.1/6952 (дата обращения 1.02.2016)

[7]. Peter Pirkelbauer, Yuriy Solodkyy, Bjarne Stroustrup. Open Multi-Methods for C++. // GPCE '07 Proceedings of the 6th international conference on Generative programming and component engineering. ACM. New York, NY, USA. 2007. P. 123-134. DOI:10.1145/1289971.1289993 Режим доступа: http://www.stroustrup.com/multimethods.pdf (дата обращения 1.02.2016)

[8]. Виноградова М.В. Методика создания мультиаспектной информационной системы с алгоритмоориентированной структурой данных: автореф. дис. ... канд. техн. наук. М.: МГТУ им. Н. Э. Баумана. 2005. 16 с.

[9]. Балдин А.В., Данчул А.Н. О реализации сервис-ориентированной архитектуры // Инженерный вестник. МГТУ им. Н.Э. Баумана. Электрон. журн. 2012. № 7. Режим доступа: http://engbul.bmstu.ru/doc/457870.html (дата обращения 28.02.2015).

[10]. Танненбаум Э., Ван Стеен М. Распределённые системы. Принципы и парадигмы. СПб.: Питер. 2003. 877 с. — ISBN: 5-272-00053-6