УДК 002.6; 004.7; 004.722

# Проектирование и автоматизация работы облачной кластерной системы с учетом интеграции с внешними информационными системами

**В статье описывается задача проектирования и автоматизации работы облачной кластерной системы с учетом интеграции с внешними информационными системами. По результатам анализа определены основные проблемы работающей системы, сформулированы задачи по автоматизации работы облачной инфраструктуры, описаны и обоснованы способы решения поставленных задач.**

**А. Н. КУРБАЦКИЙ,**
заведующий кафедрой технологий программирования
Белорусского государственного университета,
д. т. н., профессор

**В. П. КОЧИН,**
начальник Центра информационных технологий
Белорусского государственного университета,
к. т. н., доцент

**О. В. СЛЕСАРЕНКО,**
магистрант института бизнеса БГУ

**Введение.** При проектировании сложных интегрированных систем (не только информационных) нельзя полагаться лишь на опыт архитектора или разработчика. В подавляющем большинстве случаев успешное создание сложной системы возможно при условии правильной организации процесса проектирования и правильного использования разнообразных инструментов – от аппаратно-программного обеспечения до методов математического анализа.

Проектирование сложных интегрированных систем имеет две ярко выраженные стадии. Первая относится к функционально-структурным вопросам и называется макропроектированием, а вторая – микропроектирование – связана с проектированием элементов системы как физических единиц оборудования.

Макропроектирование, рассматривающее выбор и организацию функций и структуры системы в целом, предполагает выяснение целей, для которых должна служить система, основных решаемых ею задач, исследование свойств внешней среды и определение характеристик ее воздействия на систему, а также обоснование технических требований к системе и взаимодействию ее с внешней средой в результате анализа этих данных. Оно связано с техническими решениями по основным элементам, их конструкции и параметрам, режимам эксплуатации и т. д.

Такое разделение проектирования системы на части является в значительной мере условным, и не всегда его границы четко вырисовываются. Особенно это относится к вопросам, связанным с организацией взаимодействия между элементами системы в процессе ее функционирования. Организация взаимодействия невозможна без достаточно глубокого проникновения в природу элементов системы, вместе с тем она является одной из основных общесистемных проблем.

Проектирование сложных технических систем, в частности информационных систем и систем управления, относят к специальной области системной инженерии – проектированию системы систем (System Engineering). Системная инженерия представляет собой совокупность точек зрения, методов и подходов, связанных с макропроектированием сложных систем [1].

Цели и задачи системы определяются исходя из потребностей практики, тенденций и особенностей развития современной техники, а также экономической целесообразности. В настоящее время нет возможности указать какие-либо формальные правила для решения этого вопроса, за исключением, быть может, рекомендации рассматривать конкретную систему как подсистему сложной системы более высокого уровня.

С инженерной точки зрения важно, что практически любая сложная интегрированная система нередко состоит даже не из десятков, а из миллионов элементов, динамически взаимодействующих друг с другом. В таких случаях надо понимать, какой максимальный уровень декомпозиции системы нужно считать необходимым и какую систему считать минимально неделимой и простой в рамках текущего проекта. Определившись с этим, можно ограничить глубину анализа, вследствие чего некоторые сложные подсистемы могут стать простыми.

Исходя из определения, проектирование и автоматизация работы облачной кластерной системы

относятся к классу задач, связанных с проектированием сложных интегрированных систем. В зависимости от условий подходы к решению данной задачи могут быть следующими:

1) проектирование облачной кластерной системы с нуля;

2) проектирование облачной кластерной системы на основе существующей системы.

**Автоматизация работы облачной инфраструктуры.** В Республике Беларусь в настоящее время многие учреждения и предприятия уже создали и развивают как собственную облачную инфраструктуру, так и отдельные сервисы, работающие по модели облачных технологий [2–4]. Однако по ряду причин существующая инфраструктура нуждается в модернизации и существенной доработке.

В рамках данной статьи рассмотрим второй подход.

По результатам анализа литературы можно выделить следующие основные проблемы функционирования облачных систем:

• Низкая скорость взаимодействия с внешними системами и обработки данных (система создавалась в изоляции и без оглядки на взаимодействие с другими системами).

• Низкая надежность отдельных модулей системы, вследствие чего страдает отказоустойчивость системы в целом.

• Высокие затраты на обслуживание компонентов системы и выпуск новых обновлений.

• Устаревшие версии программного обеспечения, что усложняет дальнейшую разработку и поддержку системы.

В рамках решения поставленной задачи необходимо разработать концепт новой системы, выполняющий те же функции и учитывающий сильные стороны существующей системы, исправляя фундаментальные ошибки в проектировании.

Для этого необходимо решить следующие основные задачи:

1. Спроектировать виртуальную сетевую архитектуру, учитывая ошибки существующей системы.

2. Описать процесс работы и архитектуру компонентов системы в виде кода для автоматизации, безопасности и возможности использования того же концепта в других задачах.

3. Проанализировать и выбрать ПО для управления конфигурацией серверного оборудования.

4. Проанализировать и выбрать ПО для нагрузочного тестирования.

5. Предоставить концепт управления безопасностью в реализованной системе.

6. Описать взаимодействие с элементами других сложных систем.

Рассмотрим и проанализируем каждую задачу в отдельности.

*Спроектировать виртуальную сетевую архитектуру, учитывая ошибки существующей системы.* В данный момент существует и уже функционирует сложная система, имеющая ряд определенных проблем и уязвимостей. Один из наиболее болезненных недостатков – слабая интеграция системы с компонентами внешних систем. В процессе решения данной задачи будет проанализирована реализация нынешней системы и, учитывая достоинства и недостатки, спроектирована ее новая версия, упор в которой будет сделан на взаимодействие и интеграцию с внешними системами. Это одна из приоритетных задач для системного архитектора.

*Описать процесс работы и архитектуру компонентов системы в виде кода для автоматизации, безопасности и возможности использования того же концепта в других задачах.* В данной задаче необходимо заострить внимание на реализации наиболее универсальных компонентов системы. Любая грамотно спроектированная сложная система по определению является модульной. И чем универсальнее каждый модуль, тем проще его поддерживать, использовать в других проектах или же развивать при возможном масштабировании в будущем. Кроме того, упрощается задача ведения документации и передачи знаний новым коллегам. Несомненно, это приоритетная задача, которую должен учитывать любой грамотный системный архитектор.

*Проанализировать и выбрать ПО для управления конфигурацией серверного оборудования.* В данной задаче требуется провести сравнительный анализ наиболее популярных программных инструментов с целью выбора оптимального варианта для управления серверами в обновленной кластерной системе.

*Проанализировать и выбрать ПО для нагрузочного тестирования.* В данной задаче необходимо провести сравнительный анализ наиболее популярных программных инструментов с целью выбора оптимального варианта для нагрузочного тестирования обновленной кластерной системы. Важным моментом также является вопрос лицензирования. В зависимости от задач и размера финансирования может быть выбрано платное ПО или ПО с открытой лицензией. Это очень важный момент, который может повлиять в том числе и на некоторые инфраструктурные решения.

*Предоставить концепт управления безопасностью в реализованной системе.* Под реализацией данной задачи имеется в виду разработка концепта и выбор ПО для контроля и управления безопасностью в кластерной системе. В зависимости от типа работы и возможных функций это может быть платное ПО, ПО с

открытой лицензией или же услуга, предоставляемая облачным провайдером.

*Описать взаимодействие с элементами других сложных систем.* Реализация данной задачи – крайне важный момент при проектировании любой сложной системы. Согласно теории систем каждая система имеет свои прикладные интерфейсы взаимодействия с внешним миром. И это важно понимать, т. к. чем четче осознание того, что есть «импорт», а что есть «экспорт» для системы, тем логичнее и рациональнее будет построено ее взаимодействие с другими системами. Прикладные интерфейсы должны быть четко сформулированы и заложены на самом раннем этапе проектирования сложной интегрированной системы.

Существующая облачная система реализована на базе облачного провайдера AWS (Amazon Web Services). Система состоит из взаимосвязанных элементов, включающих в себя виртуальную сетевую архитектуру, виртуальные сервера, виртуальную базу данных, облачное хранилище данных, сервер с системой контроля версий, сервер для контроля и управления элементами инфраструктуры.

Преимущества данного решения:

• Установка и удаление системы полностью автоматизированы.

• Полный процесс установки занимает менее двух часов.

• Каждая система является полностью изолированной от других на уровне виртуальной сети – это облегчает разработку, внедрение и тестирование новых функций.

• Система делится на подтипы разработки, тестирования, и нагрузки. Каждый из них отличается от других мощностью серверов, базы данных, а также подключением к внешним системам. Это облегчает процесс тестирования.

Тем не менее в процессе активной работы было выявлено большое количество проблем и уязвимостей как глобального, так и локального масштаба. Разделим их на две категории:

*Системные проблемы.*

1. Чрезмерная изоляция и, как следствие, слабая интеграция с внешними системами. Установка системы в изолированном приватном облаке со своей собственной сетевой инфраструктурой имеет свои преимущества и недостатки. Преимуществом данного подхода является простота тестирования и разработки, недостатком – практически полное отсутствие механизмов взаимодействия.

2. Инструментарий для управления платформой не поддерживает большинство стратегий выпуска новых версий продукта, таких как Blue-green deployment, Zero downtime и т. д. По этой причине некоторые версии продукта или самой системы невозможно выпустить без простоя платформы, что влияет и на внешние системы, работающие с ней в связке.

3. Даже малые компоненты системы ориентированы на использование виртуальных серверов без применения технологии контейнеризации, что несколько усложняет выпуск новых версий компонентов без простоя всей системы в целом.

*Инфраструктурные проблемы.*

1. Ошибки управления конфигурацией серверного оборудования. Для этого был выбран инструмент Puppet со стратегией настройки оборудования в режиме реального времени. Это приводит к тому, что введение нового сервера в работу может занимать до получаса, а также к меньшей надежности и большему простою системы в случае каких-либо неполадок.

2. Устаревание инструментов. Если не обновлять ПО регулярно (в частности, когда речь заходит об open source), это чревато существенной тратой ресурсов при создании нового функционала в системе, требующего обновления большого количества сторонних компонентов, либо же невозможностью обновления как такового.

3. Чрезмерная изоляция системы на уровне сетевой инфраструктуры приводит к тому, что при создании новой функции (например, система мониторинга ресурсов) нет возможности использовать существующее решение какой-либо другой системы. В результате один и тот же функционал приходится создавать несколько раз, что приводит к излишней трате времени и ресурсов.

4. Неэффективный менеджмент по контролю за ресурсами приводит к накоплению «мертвых» копий платформ, что влечет за собой дополнительные финансовые траты.

5. Отсутствие системы по контролю за безопасностью платформы приводит к невозможности оперативного устранения уязвимостей.

Как можно понять из вышеперечисленного – с течением времени накопилось большое количество проблем, которые трудно либо вообще нерешаемы в рамках текущей реализации системы. Следовательно, новая версия системы должна в той или иной мере исправлять данные недостатки. По результатам анализа были предложены следующие решения.

• *Высокий уровень изолированности системы.* Высокий уровень изолированности системы является одновременно и достоинством, и серьезным недостатком. Как уже было сказано, установка своей собственной изолированной копии системы очень удобна для разработки и тестирования новых функций. Однако это сильно мешает, когда новые функции требуют новых точек входа или выхода в системе, не заложенных в ней изначально.

В нынешней реализации, как видно из рисунка, каждая копия системы имеет свой собственный VPC (Virtual Private Cloud) – приватное облако, в котором компоненты системы полностью изолированы от внешнего мира, за исключением заранее заложенных в нем точек входа и выхода. Использование такого облака в каждой копии системы нерационально с точки зрения ресурсов, а также в случае интеграции с другими системами.

В качестве решения данной проблемы в концепте новой системы имеет смысл ввести понятие «платформа». Платформой будем считать приватное облако, обладающее определенным набором ресурсов (система мониторинга, безопасности и т. д.), в котором можно будет установить неограниченное количество копий систем.

Создание и управление общей платформой предоставит следующие преимущества:

- более рациональное использование ресурсов;

- изменение в платформе отразится на всех копиях системы, что ускорит время их обновления;

- остается возможность создать копию существующей платформы для тестирования (однако их количество будет ограничено);

- облегчает возможность использования стратегии Blue-green для выпуска новых версий системы с нулевым простоем.

- разные типы платформ можно проектировать с различными точками входов и выходов. Таким образом, можно будет создать полностью и частично изолированные платформы для разных целей;

- даст возможность размещения сильносвязных систем в одной платформе, подразумевая возможность эволюций отдельных систем в одну большую интегрированную систему.

Высокоуровневая схема платформы и размещения в ней ресурсов представлены на рисунке.

• *Инструментарий для управления платформой.* Инструментарий, используемый для управления платформой и ресурсами внутри нее, не менее важен, чем сама платформа. Как уже было сказано, инструментарий текущей системы недостаточно эффективен из-за того, что в нем не была заложена возможность релиза новых версий системы без ее простоя. Это проблема могла бы быть решена в рамках текущей системы, однако, учитывая, что ее новая версия будет использовать иной принцип работы, появляется возможность выбрать новый инструментарий, а также заложить в нем все необходимые функции на этапе внедрения.

• *Технология контейнеризации.* Технология расположения ПО в контейнерах уже давно и прочно стало стандартом индустрии информационных технологий. Поэтому при разработке новых сервисов целесообразно использовать данную технологию.

• *Управление конфигурацией серверного оборудования.* Как уже было сказано, в существующем решении был выбран инструмент по контролю и управлению конфигурацией Puppet. Прежде чем определить, почему он не является лучшим решением из существующих, определимся с понятием «система управления конфигурациями».

Система управления конфигурациями (Configuration Management) в контексте настройки серверного оборудования – это система, отвечающая за автоматизацию одновременной настройки большого количества виртуальных серверов согласно заданным параметрам. Цель таких инструментов – автоматизировать настройку и исключить варианты ручной правки конфигурации серверов и приложений, тем самым стандартизировать процесс изменений и минимизировать человеческий фактор. Эти инструменты могут работать по трем моделям:

1. Push-модель – вариант, когда существует один мастер-сервер, который знает о количестве хостов и настраивает конфигурацию на каждом из них.

2. Pull-модель – когда существует как мастер-сервер, так и агент-сервер, установленный на каждом хосте. «Агент» раз в определенный промежуток времени опрашивает «мастера» на предмет изменений.

3. Гибридная модель – возможность работать в обеих моделях по выбору администратора.

Наиболее популярными инструментами по управлению конфигурацией являются Puppet, Chef, Ansible и Salt. Puppet и Chef работают по pull-модели, Ansible – по push-модели, Salt – по гибридной.

Puppet и Chef – давние представители в своей области. Они разрабатывались в то время, когда большинство серверов были статичными и изменениям подвергались нечасто. Сейчас же, когда более рациональным вариантом считается пересоздание виртуальных серверов или создание их новой копии, Puppet оказывается неповоротливым инструментом. Конфигурация серверов в реальном времени замедляет процесс выпуска, а соединение «мастер – агент» не всегда надежно – если оно прервется, сервер останется ненастроенным. Более того, обновление такого инструмента – трудоемкий процесс. Существующая практика движется в сторону заранее подготовленных серверных образов, с которых запускаются новые виртуальные машины, тем самым уменьшая время выпуска новой версии системы и сокращая количество возможных ошибок. Но данная стратегия разработки несовместима с инструментом Puppet, так его архитектура и схема работы не подразумевают подобное использование. Более рациональным выбором являются Ansible и Salt, которые способны работать в «локальном» режиме – когда на сервере находится конфигурационный файл, по схеме которого настраивается

сам сервер и с которого в последующем создается серверный образ. Однако в последнее время Salt развивается медленнее, чем Ansible, вследствие чего у него менее широкая аудитория поддержки. Таким образом, инструмент Ansible – оптимальный выбор для управления конфигурацией новой системы.

• *Устаревание инструментов.* Как уже было сказано, необходимо непрерывное обновление текущего инструментария. Это достаточно трудоемкий процесс, но его отсутствие сильно затруднит поддержку системы в будущем. Решением может быть документация процесса слежения за последними обновлениями, а также внедрение системы по напоминанию обновления того или иного инструмента – однако это выходит за рамки проектирования самой системы.

• *Чрезмерная изоляция системы на уровне сетевой инфраструктуры.* Изоляция каждой системы в приватном облаке – неверное решение, в том числе потому, что если существует несколько сильносвязных систем, то в рамках текущей реализации кластерной системы не будет возможности использовать инструментарий друг друга для экономии ресурсов.

Решением данной проблемы, как было сказано ранее, может быть создание платформы с одним приватным облаком, в котором будет размещено несколько систем. В случае же, когда множество систем запущено в одном приватном облаке, такая проблема не возникнет.

• *Неэффективный менеджмент по контролю за ресурсами.* При частом использовании тестовых копий системы при разработке со временем накапливаются неиспользуемые ресурсы. Причины могут быть разными: ошибки при удалении, человеческая забывчивость и т. д. Все это сильно влияет на финансовые и технические затраты по их поддержке.

Для решения данной проблемы необходимо ввести следующие сервисы:

1. Сервис мониторинга и удаления тестовых систем в зависимости от определенных критериев.

2. Сервис pause – resume. Его функцией является остановка тестовых копий систем для экономии затрат и возобновление их работы по запросу.

• *Отсутствие системы по контролю за безопасностью.* Система по контролю безопасности крайне необходима не только для предотвращения хакерских атак или утечек данных, но и для предупреждения утечек секретной информации из-за человеческого фактора. Например, члены команды могут утерять важные данные, случайным образом добавить их в открытую часть системы, могут быть скомпрометированы ключи доступа вследствие уязвимостей в схемах шифрования и т. д.

В качестве решения данной проблемы необходимо ввести проверки безопасности следующего типа:

1. Использование встроенного сервиса облачного провайдера Amazon Inspector для анализа существующей инфраструктуры на предмет потенциальных уязвимостей.

2. Разработка инструмента для оперативных исправлений найденных уязвимостей.

3. Введение стандартных политик безопасности, таких как ротация паролей, приватных ключей раз в определенный промежуток времени.

4. Применение встроенных функций в используемых инструментах на поиск уязвимостей, открытых паролей, хешей ключей в конфигурационных файлах.

**Заключение.** Суммируя вышесказанное, можно заключить, что для каждой из восьми проблем, существующих в текущей реализации системы, были найдены как решения, так и альтернативы.

## ЛИТЕРАТУРА

1. Системная инженерия: принципы и практика / Александр Косяков, Уильям Н. Свит, Сэмюэль Дж. Сеймур, Стивен М. Бимер; ред. перевода В. К. Батоврин. – 2-е изд. – Москва: ДМК Пресс, 2017. – 623 с.
2. Управление программным обеспечением и обеспечение отказоустойчивости IaaS-облака / Ю. И. Воротницкий, В. П. Кочин, В. А. Волчок, А. И. Бражук // Электроника инфо. – 2013. – № 9. – С. 21–24.
3. **Кочин, В. П.** Управление программными проектами на основе облачного сервиса PaaS суперкомпьютера СКИФ / В. П. Кочин, А. В. Жерело // Электроника инфо. – 2013. – № 9. – С. 35–36.
4. Суперкомпьютерные технологии в образовательном процессе и научных исследованиях в БГУ / Ю. И. Воротницкий, А. В. Жерело, В. П. Кочин, Г. Г. Крылов, Л. З. Утко // Суперкомпьютерные системы и их применение (SSA'2010): материалы третьей международной научной конференции, Минск, 25-27 октября 2010 г. / ОИПИ НАН Беларуси; редкол.: С. В. Абламейко [и др.]. – Минск, 2010. – С. 229–234.

***The article describes the task of designing and automating the operation of a cloud cluster system, taking into account integration with external information systems. Based on the results of the analysis, the main problems of the operating system are identified, the tasks for automating the work of the cloud infrastructure are formulated, and the ways of solving the tasks are described and substantiated.***

*Получено 12.03.2021.*