УДК 004.056.53

МЕТОДЫ СНИЖЕНИЯ РИСКОВ ПОТЕРЬ ДОСТУПНОСТИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ КРИТИЧЕСКИХ ИНФОРМАЦИОННЫХ СИСТЕМ

С.А. Арустамов, М.Г. Генин

Рассматриваются возможности и ограничения резервирования для снижения риска потери доступности критических информационных систем. Приводится описание метода декомпозиции обновлений для снижения риска потери доступности критических систем при проведении обновлений. Доказывается, что использование метода декомпозиции совместно с резервированием снижает риски потери доступности критических систем при обновлениях. Приводится пример, иллюстрирующий эффективность использования метода декомпозиции.

Ключевые слова: информационные системы, резервирование, доступность, риск потери доступности, обновление, метод декомпозиции, простое обновление.

Введение

В работе рассматриваются риски потери доступности программного обеспечения (ПО), не связанные с вмешательством злоумышленников, на примере критических информационных систем (КИС). Будем называть информационные системы критическими, если они имеют жесткие требования по ресурсной доступности. Под доступностью в этой работе будем понимать возможность системы выполнять заявленные функции их легальными пользователями. При этом не будем касаться таких тем, как угроза потери или искажения информации в результате взлома или других злонамеренных действий, хотя, безусловно, эти действия тоже влияют на возможность системы выполнять свои функции [1]. Говоря о доступности, будем рассматривать разного рода неполадки оборудования или ПО, которые не являются результатом чьего-то злого умысла, а происходят из-за сбоев оборудования, ошибок в ПО или являются следствием установленных обновлений. Риск потери доступности определим как пару, состоящую из вероятности наступления того или иного события, приведшего к потере доступности, и ущерба от наступления этого события:

$$R_i=\{P_i, I_i\},$$

где R_i – риск наступления i-го события; P_i – вероятность i-го события; I_i – ущерб от i-го события. Ущерб от i-го события I_i, в свою очередь, может быть представлен в виде составляющих:

$$I_i=\{i_{ij}\}, j=\{1, 2, \ldots, N\},$$

где i_{ij} – j-я составляющая ущерба. Такими составляющими могут быть как прямой финансовый ущерб, так и другие виды ущерба, наступившие в результате потери клиентов, уменьшения доли контролируемого рынка, штрафа со стороны надзорных органов, отзыва лицензии и т.д.

Объекты рисков потери доступности

Для дальнейшего рассмотрения рисков потери доступности и их влияния на работоспособность системы представим всю систему в виде составляющих, к которым могут применяться риски потери доступности. Будем рассматривать систему как состоящую из статической и динамической частей. К статистической части (СЧ) будем относить то, что воспринимается как «черный ящик», который в случае неполадок можно легко заменить в любой момент без существенного ущерба для работы (каналы связи, сетевое оборудование, серверы, дисковые массивы и т.д.). Например, если сервер, выполняющий определенную роль, может быть легко заменен на резервный, то его тоже можно отнести к СЧ.

К динамической части (ДЧ) будем относить только те ПО и данные, замена которых на аналогичные в случае неполадок или сбоев не приводит к полному восстановлению работы системы без потери функциональности. Такая ситуация может сложиться, если сбой носит логический характер, например, ошибка в коде программы или в структуре базы данных (БД). В этом случае использование резервной копии может оказаться бессмысленным, так как логическая ошибка также будет присутствовать и в ней.

Как правило [2], вопросы обеспечения доступности ПО связываются с применением методов резервирования оборудования, ПО и данных. Рассмотрим возможности и ограничения применения традиционного резервирования для снижения риска потери доступности КИС.

Резервирование как метод снижения рисков потери доступности и области его использования

Для СЧ систем существуют способы добиться такого уровня резервирования, когда система автоматически переключается на работу с резервными ресурсами полностью прозрачно для бизнес-пользователей. Это достигается как на уровне каналов связи [3], когда выполняется автоматическое переключение на резервный канал при потере доступности основного, так и на уровне серверного оборудования [4, 5], когда вместо основного сервера начинает использоваться резервный, будь то сервер приложений или сервер БД.

Что касается возможности и целесообразности использования резервирования для снижения риска потери доступности ДЧ, то здесь ситуация более сложная. Рассмотрим некоторые случаи возможных потерь доступности данных и ПО ДЧ.

Наиболее вероятная причина потери доступности данных – это их физическое повреждение, вызванное аппаратными сбоями. Как показано выше, такая проблема решается наличием так называемого «горячего» резерва оборудования. Более сложный случай – это логическое повреждение данных. В этом случае наличие «горячего» резерва может оказаться недостаточным, так как на «горячей» копии произошли те же самые изменения. Потребуется возврат к состоянию данных до сбоя путем восстановления данных из так называемого «холодного» резерва. При этом данные, введенные после создания последней «холодной» копии, будут утеряны.

В табл. 1 приведена информация об описанных выше типах сбоя данных с точки зрения возможности восстановления работоспособности системы с помощью резервирования.

Виды сбоя данных	Решение	Степень потери доступности	Возможный ущерб
Физическое повреждение данных	Наличие «горячей» резервной копии	С незначительными потерями	Минимальный
Логическое повреждение данных (пример: удаление небольшого количества записей из таблицы, удаление небольшой таблицы)	Возврат к состоянию до сбоя на текущей версии БД, частичное восстановление данных из резервной копии	С частичными потерями	Минимальный – средний
Серьезное логическое повреждение данных (пример: удаление большого количества записей из таблицы, удаление нескольких таблиц, некорректное изменение структуры при обновлении)	Полное восстановление из резервной копии	С существенной потерей	Средний – существенный

Таблица 1. Возможности снижения риска потери доступности путем резервирования (данные)

Перейдем теперь к рассмотрению случаев потери доступности ПО. Самый простой случай – это сбой ПО без влияния на данные. При данном типе сбоя наличие резервной копии ПО (или окружения с ПО) решает проблему доступности. Можно перейти на резервное окружение с ПО при неполадках основного, либо вернуться к предыдущей версии ПО, если проблемы возникли после обновления. Оба варианта реализуемы, так как структура данных не менялась.

Виды сбоя ПО	Решение	Степень потери доступности	Возможный ущерб
Сбой ПО без повреждения данных	Замена оборудования с ПО (текущая версия), замена оборудования с возвратом к предыдущей версии после обновления ПО (возможно, если не менялась структура БД)	С незначительными потерями	Минимальный
Сбой после обновления ПО при измененной во время обновления структуре БД	Возврат к предыдущей версии ПО с частичной потерей данных, введенных после изменения их структуры и обновления ПО (возможно при отсутствии внешних ограничений на структуру данных)	С частичными потерями	Минимальный – средний
Сбой после обновления ПО при измененной во время обновления структуре БД в случае невозможности использовать старое ПО и данные старой структуры (внешние ограничения)	Отсутствует	С полной потерей	Максимальный

Таблица 2. Возможности снижения риска потери доступности путем резервирования (ПО)

Более сложный случай – это сбой после обновления, при котором менялась не только версия ПО, но и структура данных. Простой возврат на предыдущую версию ПО в этом случае невозможен, так как структура данных уже изменилась. Для этого придется еще восстанавливать данные в старой структуре. Однако это может привести к частичной потере данных, введенных в систему после обновления, так как

они уже вводились в новую структуру. Приходится признать, что в случае сбоя такого типа резервирование лишь отчасти решает проблему восстановления доступности системы, так как некоторая потеря данных неизбежна. Самый сложный случай – это предыдущий, но с невозможностью возврата к данным старой структуры из-за внешних ограничений. Такими ограничениями могут быть вступившие в силу с определенной даты новые правила предоставления или обработки информации, нормативные документы, форматы обмена данными и т.д. Необходимо добиваться восстановления работоспособности нового ПО уже на новой структуре данных. Резервирование в данной ситуации не применимо.

В табл. 2 приведена информация об описанных выше типах сбоя ПО с точки зрения возможности восстановления работоспособности системы с помощью резервирования.

Сведем результаты анализа в единую таблицу.

	Без потерь	С частичными потерями	С полной потерей
Аппаратная часть	Возможно		
БД	Возможно при физическом повреждении и при наличии «горячего» резерва. Сводится к резерву на уровне аппаратной части. Ущерб: минимальный	Возможно при логическом повреждении. Возврат к предыдущему состоянию на текущей версии БД, частичное/полное восстановление из резервной копии. Ущерб: минимальный – средний	Только при отсутствии резерва. Ущерб: максимальный
ПО	Возможно при сбое без повреждения данных и без изменения их структуры. Возврат к предыдущей версии ПО. Ущерб: минимальный	Возможно при измененной во время обновления ПО структуре БД и при отсутствии внешних ограничений. Возврат к предыдущей версии ПО и структуре БД. Ущерб: средний – существенный	При измененной во время обновления структуре БД в случае невозможности использовать БД старой структуры (внешние ограничения). Возврат к предыдущей версии ПО и БД невозможен. Ущерб: максимальный

Таблица 3. Возможности снижения риска потери доступности путем резервирования

Как видно из табл. 3 (нижний правый угол таблицы), наиболее рискованной с точки зрения возможностей резервирования является ситуация, когда при обновлении ПО одновременно меняется и структура БД, причем возврат к предыдущей версии ПО и БД невозможен из-за внешних ограничений (рис. 1). В этой ситуации резервирование не поможет в принципе, на каком бы уровне оно ни было организовано.

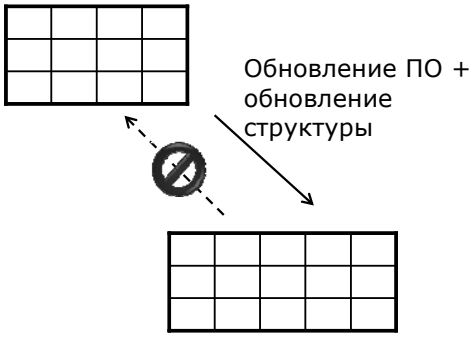

Рис. 1. Ситуация, при которой резервирование не работает

Соответственно, возможный ущерб от возникновения данной ситуации будет максимальным. Для уменьшения рисков доступности, не уменьшаемых с помощью резервирования, авторами предлагается иной подход к рассматриваемой проблематике.

Метод декомпозиции функциональных обновлений

Выше было показано, что наименее рискованными из всех рассмотренных ситуаций являются те, когда за один раз меняется только версия ПО или только структура БД, с возможностью возврата к предыдущему состоянию системы. Для таких ситуаций может применяться резервирование.

Вторая по уровню сложности ситуация – это одновременное обновление ПО и структуры БД, но с возможностью возврата к предыдущей конфигурации. Здесь резервирование применимо, но объем необходимого восстановления будет наибольшим.

Наиболее сложный случай – это одновременное выполнение обновления версии ПО и структуры БД при невозможности возврата к предыдущей версии ПО и БД из-за внешних ограничений. В этом случае резервирование не поможет, а ущерб при этом будет наибольшим. Для снижения риска потери доступности в данном, самом сложном случае, предлагается использовать метод декомпозиции. Суть его в том, чтобы свести наиболее рискованную ситуацию к нескольким, но менее рискованным, т.е. разбить планируемое сложное обновление на несколько с тем, чтобы при одном таком простом обновлении менять либо только структуру БД, либо только ПО (рис. 2).

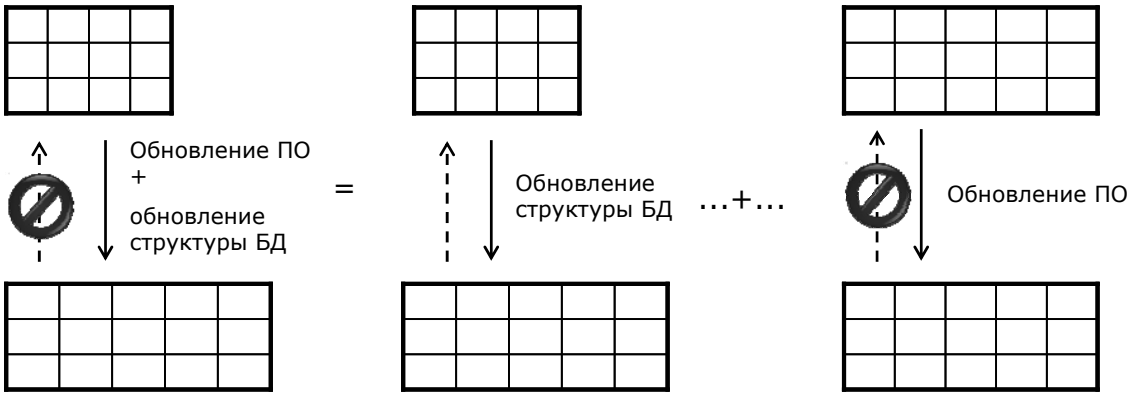

Рис. 2. Разбиение одного общего обновления ПО и БД на несколько отдельных (либо только ПО, либо только БД)

Самое последнее обновление – обновление ПО – в этой цепочке простых обновлений будет включать новую требуемую функциональность, без наличия которой, начиная с определенного момента, система дальше работать не может по тем или иным внешним причинам. Все предыдущие простые обновления (ПО или БД), которые готовят систему к установке последнего, ключевого обновления ПО, могут быть отменены вплоть до установки последнего. На этапе установки каждого из подготовительных простых обновлений возможен возврат к предыдущему состоянию системы, для выполнения которого может применяться резервирование. Возможный ущерб при этом, согласно табл. 3, будет либо минимальным, либо средним. Необходимым условием для такого способа разбиения является работоспособность системы после установки каждого из промежуточных подготовительных обновлений.

Покажем, что применение метода декомпозиции для такого рода обновлений снижает риски потери доступности. Для этого перейдем к формализации описанной методики. Обозначим U^{DB+SW} – все полное сложное обновление, включающее в себя обновление ПО и структуры БД; u_i^{DB} и u_i^{SW} – простые *i*-е обновления БД и ПО соответственно. Тогда можно записать:

$$U^{DB+SW} = u_1^{DB} + u_2^{SW} + u_3^{DB} + u_4^{SW} + ... + u_{N-3}^{DB} + u_{N-2}^{SW} + u_{N-1}^{DB} + u_N^{SW}, \quad (1)$$

где N – четное. При этом

$\forall u_i^{DB} : i\{1,3,5,...,N-1\}, \exists\, \overline{u}_i^{DB}$,

$\forall u_j^{SW} : j\{2,4,6,...,N-2\}, \exists\, \overline{u}_j^{SW}$,

для u_N^{SW} не существует $\overline{u}_N^{SW}$,

т.е. все обновления, кроме последнего, обратимы.

Можно перейти от этой нотации к более простой. Отметим, что в (1) все нечетные обновления являются обновлениями БД, а четные – обновлениями ПО. Обозначим

$U = U^{DB+SW}$,

$u_i = u_i^{DB}, \forall i\{1,3,5,...,N-1\}$,

$u_j = u_j^{SW}, \forall j\{2,4,6,...,N\}$,

где N – четное. Тогда формула (1) приобретает следующий упрощенный и обобщенный вид:

$$U = u_1 + u_2 + u_3 + u_4 + ... + u_{N-3} + u_{N-2} + u_{N-1} + u_N, \quad (2)$$

$\forall u_i : i\{1,...,N-1\}, \exists\, \overline{u}_i$,

для u_N не существует $\overline{u}_N$.

Разложение обновления на составляющие в виде (2) полностью[1] соответствует определению полного обновления U как суммы простых обновлений u_i, которое было введено в [6]. Действительно:

1. в определенный момент времени устанавливается только одно обновление u_i;
2. после успешной установки каждого обновления u_i система работоспособна;
3. в случае возникновения проблем после установки любого обновления u_i возврат к предыдущему состоянию системы происходит за допустимый интервал времени и либо не приводит к потере данных вообще, либо приводит лишь к минимальной потере данных, связанных с новым функционалом.

В этом случае, в соответствии с [6], можно сделать вывод о том, что вероятность успешной установки полного обновления U в формуле (2), выполненного методом декомпозиции, будет выше, чем при выполнении всего обновления U сразу. Так как формула (1) является частным случаем формулы (2), то данный вывод справедлив и для исходного разложения U^{DB+SW} на составляющие u_i^{DB} и u_i^{SW}. Так как возможный ущерб от ошибок и их последующего исправления при выполнении простых обновлений u_i^{DB} или u_i^{SW} будет либо минимальным, либо средним, то можно сделать вывод о том, что полное обновление U, выполненное методом декомпозиции, менее рискованно, чем выполненное единовременно, когда ущерб может быть максимальным.

Таким образом, метод декомпозиции может быть использован совместно с резервированием для снижения риска потери доступности при проведении комплексных обновлений критических информационных систем. Действительно, все подготовительные обновления, кроме последнего, обратимы. Риск потери доступности для них может быть снижен путем резервирования. Последнее «простое» обновление является обновлением ПО и лишь «включает» требуемую функциональность. Оно тоже обратимо, однако при этом необходимая функциональность будет недоступна. Что же касается риска сбоя последнего «простого» обновления, то он существенно ниже риска сбоя всего полного обновления U. В случае ошибки ПО в последнем обновлении легче локализовать ошибку и организовать процесс ее исправления.

Описание экспериментов, тестирующих предложенную методику

Для проверки предложенной методики использовалась инфраструктура финансового-кредитного учреждения (ФКУ) с головным отделением в Санкт-Петербурге и филиалом в Москве. В каждом из офисов было созданы тестовые платформы, на которых была развернута система дистанционного банковского обслуживания (система «Банк–Клиент», далее БК).

Система БК предоставляет удаленным клиентам возможность обмена документами с ФКУ в режиме веб-доступа (онлайн-клиенты) и с помощью приложения с БД на стороне клиента с доступом в ФКУ по модему, либо через Интернет (оффлайн-клиенты). В ФКУ система состояла из веб-сервера для подключения онлайн-клиентов, транспортной станции для подключения оффлайн-клиентов, общего сервера БД и сервера обмена данными с приложениями ФКУ.

На каждой тестовой платформе в ФКУ была развернута одна банковская часть, обслуживающая по 5 онлайн- и оффлайн-клиентов. Обновление, которое предполагалось применить на обеих тестовых платформах, добавляло новый функционал в один из существующих типов документов. Изменялась структура документа, добавлялись новые поля, новые справочники, новые правила контроля. Формат файлов обмена с внешними подсистемами расширялся за счет добавления новых полей. Обновление затрагивало почти все части системы. Менялась структура БД, процедуры обмена данными, веб-сервер (для онлайн-клиентов), клиентское приложение с собственной БД для оффлайн-клиентов. Неизменной оставалась только транспортная станция.

Пакет обновления был подготовлен в двух вариантах – для единовременной установки и для поэтапной установки. При единовременной установке предполагалось одновременное обновление клиентской платформы и платформы ФКУ. Для обновления оффлайн-клиентов новая версия приложения была предварительно разослана клиентам.

Пакет обновления для поэтапной установки был разбит на части, каждую из которых нужно было устанавливать отдельно в определенной последовательности. Для подтверждения корректности установки каждой из частей пакета система должна была отработать в продукции не менее одного дня перед установкой, следующей по порядку части. Обновление в ФКУ включало в себя расширение структуры БД, добавление новых справочников, обновление ПО сервера обмена данными с использованием режима совместимости со старым и новым форматом обмена, обновление ПО веб-сервера. У оффлайн-клиентов обновлялась версия ПО с применением режима совместимости со старой структурой документа. После того, как все оффлайн-клиенты подтверждали факт обновления ПО у себя, режим совместимости отключался на стороне ФКУ.

---

[1] Запрет на возврат к предыдущему состоянию системы для последнего простого обновления u_N в (2) обусловлен исключительно внешними ограничениями и не является свойством самого обновления u_N. Технически возврат системы от u_N к u_{N-1} возможен, однако при этом требуемая функциональность будет недоступна.

В табл. 4 приведены данные об ошибках, возникших во время установки обновлений на тестовые платформы в офисах Санкт-Петербурга и Москвы.

	Время установки всех обновлений	Всего ошибок	Из них привели		Максимальное время полной недоступности системы
			к частичной недоступности	к полной недоступности	
Тестовая платформа 1, единовременная установка (офис в Санкт-Петербурге)	1 день	3	2 Устранены за 15 мин	1 Устранена за 48 часов	48 часов
Тестовая платформа 2, поэтапная установка (офис в Москве)	12 дней	2	2 Устранены за 15 мин	–	–

Таблица 4. Данные о потере доступности системы при установке обновлений

Как видно из приведенных данных, поэтапная установка обновлений заняла существенно больше времени по сравнению с единовременной, но при этом оказалась гораздо менее рискованной с точки зрения доступности приложения.

Заключение

Были рассмотрены возможности и ограничения резервирования для снижения риска потери доступности информационных систем, в частности, при проведении обновлений. Предложен метод декомпозиции, который позволяет снизить риски при проведении комплексных обновлений критических информационных систем. Доказано, что риск потери доступности критических информационных систем при проведении комплексных обновлений снижается при использовании метода декомпозиции. Проведено тестирование предложенного метода, которое подтвердило корректность приведенных положений.

Литература

1. Арустамов С.А., Генин М.Г. Минимизация рисков потери доступности программного обеспечения после установки обновлений или изменения функциональности // Научно-технический вестник СПбГУ ИТМО. – 2011. – № 3 (73). – С. 111–116.
2. Резервное копирование как неотъемлемый элемент обеспечения ИБ. Информационная безопасность [Электронный ресурс]. – Режим доступа: http://www.itsec.ru/articles2/control/rezervnoe-kopirovanie-kak-neotemlemyi-element-obespecheniya-ib, свободный. Яз. рус. (дата обращения 26.03.2012).
3. Резервирование сетевых соединений в критически важных приложениях. Network Systems Group [Электронный ресурс]. – Режим доступа: http://www.nsg.ru/doc/whitepapers/wp_nsg_dualuplink.pdf, свободный. Яз. рус. (дата обращения 26.03.2012).
4. High availability. That works. Vision solutions [Электронный ресурс]. – Режим доступа: http://www.visionsolutions.com/Products/High-Availability-Overview.aspx, свободный. Яз. англ. (дата обращения 26.03.2012).
5. Check Point High Availability for IP Appliances. CheckPoint [Электронный ресурс]. – Режим доступа: http://www.checkpoint.com/products/ip-appliances/check-point-high-availability.html, свободный. Яз. англ. (дата обращения 26.03.2012).
6. Генин М.Г. Методика проведения обновлений, снижающая риски доступности информационных ресурсов. Информатика, моделирование, автоматизация проектирования // Сборник научных трудов / Под ред. Н.Н. Войта. – Ульяновск :УлГТУ, 2011. – С. 125–137.

Арустамов Сергей Аркадьевич – Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, доктор технических наук, профессор, sergey.arustamov@gmail.com

Генин Михаил Геннадьевич – Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, аспирант, gmg@rambler.ru