электронный журнал
МОЛОДЕЖНЫЙ НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51038.

УДК 004.75

Оценка эффективности механизмов связывания процессов при построении масштабируемых отказоустойчивых приложений с использованием языка Erlang

Петров Ю.К., *студент*
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана
кафедра «Компьютерные системы и сети»

Научный руководитель: Иванова Г.С., д.т.н., профессор
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана
v.suzev@bmstu.ru

Введение

Известно, что создание отказоустойчивых приложений требует не менее двух компьютеров [1]. Основной сложностью реализации такого приложения является обработка экстренных ситуаций, например, выход из строя одного узла и связанная с этим работа по перераспределению его задач на действующие узлы. Особенно остро эта проблема стоит в системах высокой доступности, таких, как серверы IM.

Одним из решений является использование соединений между процессами, позволяющих отслеживать программные и аппаратные сбои на узле, предпринимая при этом определённые действия.

Именно такой механизм положен в основу средств, предоставляемых языком программирования Erlang. Выбор данного языка для исследования обусловлен тем, что

1. Язык Erlang специально был создан для систем высокой доступности. Одним из изначальных требований при проектировании языка была надёжность по отношению к аппаратным и программным сбоям.

2. Только в языке Erlang максимально полно реализована перспективная с точки зрения построения отказоустойчивых приложений технология связывания процессов.

Целью статьи является анализ средств связывания процессов, предлагаемых языком программирования Erlang, и оценка их эффективность при построении масштабируемых систем с заданным классом надёжности.

1. Анализ использования механизмов связывания процессов при построении масштабируемых отказоустойчивых систем

При построении отказоустойчивых приложений с помощью средств языка Erlang применяется многослойная архитектура, позволяющая в случае возникновения исключительной ситуации обнаружить и изолировать её. При этом используются специальные процессы-наблюдатели.

На рисунке 1 показан пример построения системы со структурой, называемой деревом наблюдения. Листьями дерева наблюдения являются рабочие процессы (P), узлами – процессы-наблюдатели (H). В случае аварийного завершения рабочего процесса сигнал просто пересылается в дереве на уровень выше, изолируя исключительное поведение от остальных частей программы, что полностью соответствует одному из методов построения надёжных систем [1,2].

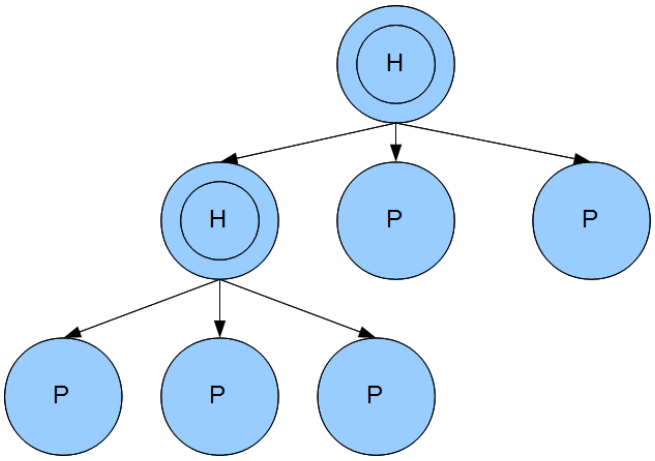

Рис. 1. Дерево наблюдения

Особенность процессов-наблюдателей состоит в том, что они не выполняют никакой работы, кроме наблюдения за связанными процессами. Суть технологии, использующей процессы-наблюдатели, заключается в том, что возможно установить соединение между процессами так, что один из процессов будет перехватывать сигналы выхода другого, предпринимая при этом заранее определённые действия: перезапуск завершившегося процесса, перераспределение работы по не вышедшим из строя процессам и т.д. Таким образом технология связанных процессов позволяет программисту тратить меньше времени на разработку различных сценариев появления ошибок и защитное программирование.

Соединения между процессами могут быть однонаправленные или двунаправленные.

Механизм двунаправленного связывания удобен, если в системе используются зависящие друг от друга процессы: в случае возникновения ошибки можно будет быстро завершить одновременно их все и тут же перезапустить.

Двунаправленные соединения также подходят для построения деревьев наблюдений. В случае если клиент вызывает процесс, на состояние которого не нужно влиять и которому не нужно получать сигналы выхода от других процессов, то лучше воспользоваться однонаправленным соединением.

Ещё одним полезным механизмом, реализованным в языке Erlang, является возможность создания процессов-перехватчиков. Процесс-перехватчик перехватывает сигнал выхода присоединённого к нему процесса, предпринимая при этом определённые действия. На рисунке 2 процесс-перехватчик B получает сигналы выхода процесса A.

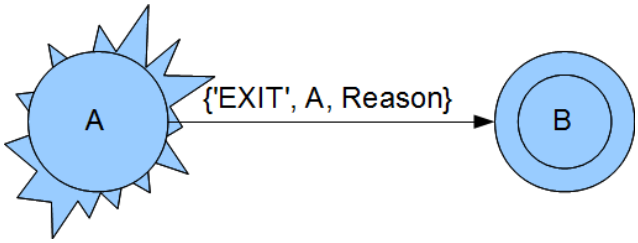

Рис. 2. Процесс-перехватчик

Для перехвата сигналов выхода необходимо установить соответствующий флаг. После этого получаемые процессом сигналы выхода преобразуются в обычные сообщения, которые будут поступать в почтовый ящик процесса-перехватчика.

Все механизмы связывания, реализуемые средствами языка Erlang, доступны как для локальных, так и для удалённых процессов. При этом код практически не требует модификации, единственным отличием служит только обязательное указание узла, на котором запущен связанный процесс.

Таким образом использование рассмотренных механизмов позволяет построить систему, относящуюся к следующим классам [2]:

1. *Необслуживаемые системы*. Построенная система не требует обязательного регулярного технического обслуживания, что подтверждается проведёнными компанией Ericsson исследованиями схожих систем: надёжность коммутатора ATM AXD301 оценивается равной 99,99999 % [1].

2. *Восстанавливаемые системы*. Построенная система по возможности автоматически восстанавливает экстренно завершившиеся процессы.

Что подтверждает получение систем высокой долговечности и с максимально возможными для данного аппаратного обеспечения коэффициентами готовности.

Единичные показатели надёжности [2] определяются для каждой конкретной реализации системы.

2. Экспериментальная оценка эффективности средств обеспечения безотказной работы языка Erlang

При экспериментальной оценки эффективности технологии связывания, реализуемой средствами языка Erlang, использовалось локальное приложение со связыванием.

При выборе модели приложения для проведения эксперимента были учтены следующие требования:

1. Случайные задержки должны быть минимальны.
2. Модель должна обеспечивать достаточную достоверность информации, запуская и контролируя как можно больше процессов.

Исходя из первого требования, подходит лишь программа, запущенная в одной виртуальной машине (задержки в сообщении между двумя виртуальными машинами напрямую зависят от трафика общей шины, задержки между узлами в локальной сети зависят от качества связи и нагрузке на сеть).

Для выполнения второго требования потребовалось, чтобы в модели должно быть как можно больше рабочих процессов. Таким образом, с одной стороны, имитируется высокая нагрузка на процесс-перехватчик, с другой стороны, среднее время задержки получается максимально приближенным к реальному.

Структура исследуемого приложения изображена на рисунке 3. Приложение выполняет следующие операции: главный узел запускает рабочие процессы с заданными параметрами на рабочих узлах, одновременно связывая их с процессом-наблюдателем. Наблюдатель перехватывает все сигналы от рабочих процессов, разделяя их на 2 вида: результат и ненормальное завершение. В случае ненормального завершения наблюдатель перезапускает процесс, связывая его с собой. В случае получения результата он возвращает его главному узлу.

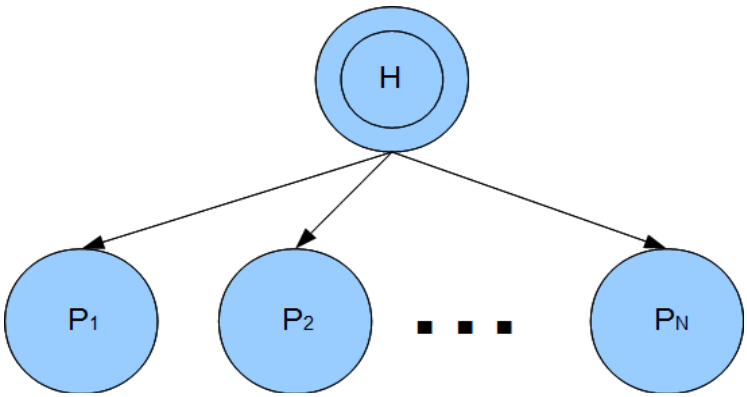

Рис. 3. Модель перехватчика с N связанными процессами

Всего было проведено 1000 экспериментов в двух сериях: по 100 экспериментов для *N*=1000, 1500, 5000, 7500, 10000. В первой серии экспериментов всем процессам передавался неправильный параметр, они аварийно завершались, наблюдатель перехватывал сигналы выхода и перезапускал процессы с правильным параметром, после чего они успешно завершались. Во второй серии экспериментов все процессы сразу получали правильный параметр. Полученные значения времен были усреднены.

Тест проводился на процессоре Intel Core 2 Quad Q8400 @2,66GHz, использовалось только одно процессорное ядро. Результаты представлены в таблице и на рисунке 4.

Время работы тестового прило-жения	Количество процессов, шт					Среднее время, мс
	1000	1500	5000	7500	10000	
$t_{сп}$, мс	24,23	40,28	88,37	125,87	150,55	
$t_{бп}$, мс	7,31	11,16	30,06	46,79	63,03	
$t_{сп}$-$t_{бп}$, мс	16,92	29,12	58,31	79,08	87,52	
На 1 процесс, мс	0,0169	0,01941	0,01166	0,01054	0,00875	0,0134582
На перезапуск, мс	0,0096	0,01197	0,00565	0,00430	0,00244	0,0067975

В таблице использованы следующие обозначения:

$t_{сп}$ – время работы тестового приложения с перезапуском процессов;

$t_{бп}$ – время работы тестового приложения без перезапуска процессов.

В строке «на 1 процесс» представлена вычисляемая величина: отношение разности времён работы тестовой программы с перезапуском и без перезапуска к количеству процессов. Данная величина иллюстрирует, на сколько увеличивает время работы каждый процесс в случае перезапуска. В строке «на перезапуск» показана вычисляемая величина, отображающая, сколько времени уходит на перезапуск одного процесса. Данная величина

вычисляется как разность времени работы с перезапуском и двух времён работы без перезапуска (так как каждый процесс выполняется дважды). В столбце «в среднем» приведены средние величины для строк «на 1 процесс» и «на перезапуск».

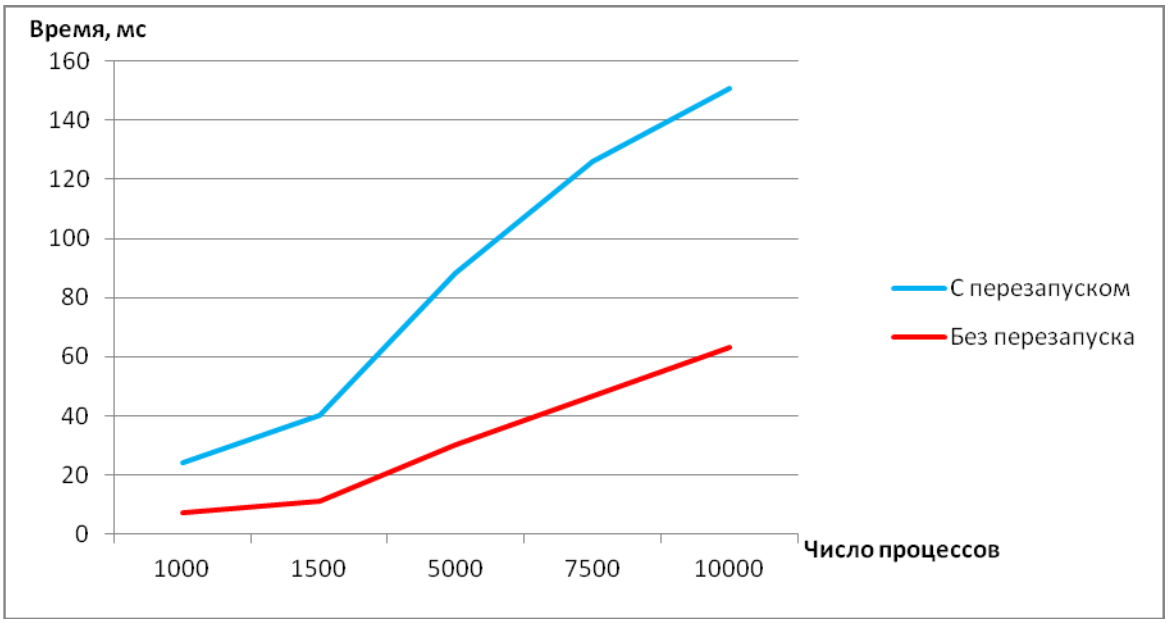

Рис. 4. Зависимость времени работы приложения от количества процессов

Необходимо отметить, что таймер в Erlang/OTP имеет погрешность около 15-16 мс, что сказалось на значениях N=1000 и N=1500 процессов. Этим фактором и объясняется изгиб графика.

Выводы

Реализация средств поддержки механизма связывания процессов в языке Erlang

1) позволяет создавать отказоустойчивые масштабируемые системы;

2) является высокоэффективной, так в результате выполненного эксперимента получено, что операции связывания и перезапуска процессов сопоставимы по скорости с вызовом функции (перезапуск занимает около 7 мкс).

Список литературы

1. Чеззарини Ф., Томпсон С. Программирование в Erlang. М.: Изд-во «ДМК», 2012. С.167-181, 273-286.
2. Бройдо В.Л., Ильина О.П. Вычислительные системы, сети и телекоммуникации. 4-е издание. СПб.: Изд-во «Питер», 2011. С.493-499.