Метод надёжной временной изоляции для ARINC 653 ОСРВ

В.Ю.Чепцов cheptsov@ispras.ru

Институт системного программирования им. В.П. Иванникова РАН, Москва

В основе архитектуры ARINC 653[1] лежит надёжная изоляция (Robust Partitioning). Концепция надёжной изоляции изначально появилась в DO-248[2] и была дополнена в части многоядерности CAST-32A[3]. В ARINC 653 представлена реализация этой концепции, которая состоит из надёжной изоляции ресурсов (Robust Resource Partitioning) и надёжной временной изоляции (Robust Time Partitioning). Хотя стандарт ARINC 653 и налагает архитектурные ограничения на ОСРВ, например, двухуровневая система планирования задач, включающая статически заданные расписания выполнения разделов, для обеспечения переносимости непосредственные программно-аппаратные способы достижения надёжной временной изоляции в стандарте не сформулированы. Данная задача должна решаться разработчиком ОСРВ, исходя из следующих ограничений:

- Сбои в одном разделе не могут приводить к негативным последствиям в других разделах.
- Раздел не может выполняться на процессорном ядре более длительное время, чем ему было выделено, вне зависимости от активности или неактивности других разделов на других процессорных ядрах.

Основной вопрос, который необходимо решить при разработке ARINC 653 совместимой ОСРВ в таких ограничения, заключается в построении архитектурного решения в части обработки асинхронных событий. При нормальной работе операционной системы необходимо обеспечивать выполнение отложенных событий (таймеров) и реализацию обменов данными с внешним миром. Наивный способ добавления такой поддержки в ОСРВ – использовать аппаратные прерывания с некоторым периодом, которые будут выполнять задачи по перепланированию событий и обеспечивать передачу данных во внешний мир. Однако такой способ приводит к появлению большого количества накладных расходов:

- При выполнении приложения процесс может быть прерван в произвольный момент времени для обработки прерывания.
- При расчёте наихудшего времени выполнения кода необходимо учитывать максимальную длительность и минимальный период обработки прерывания.
- Для приближения оценки наихудшего времени выполнения кода к реальной необходимо вычислять время выполнения на достаточно длинных трассах, чтобы не считать время обработки прерывания несколько раз.
- Низкая частота обработки прерываний приводит к высокой латентности системных сервисов, когда же высокая частота приводит к повышению наихудшего времени выполнения кода.
- Для снижения времени обработки прерываний и повышения эффективности обменов данными с внешним миром драйверы передатчиков данных часто размещаются в пространстве ядра ОСРВ, что повышает вероятность отказа и расширяет поверхность атаки на привилегированный код.

В данной работе предлагается дополнить концепцию использования непериодических таймеров для планирования задач в ARINC 653 ОСРВ выравниванием на заданный квант времени. Основная идея непериодического таймера планировщика заключается в установке таймера на указанное время в момент выбора временного окна или установки таймера. При

этом, если два события требуют отложенной реакции, значение таймера выставляется на то из них, которое должно случиться раньше. При срабатывании таймера на первое событие, последующее извлекается из программной очереди и обновляет значение таймера.

Попытки применить непериодические таймеры в локальных задачах ОСРВ возникали в разное время. Например, FreeRTOS позволяет отключать периодический планировщик в момент ухода в режим пониженного энергопотребления, тем самым фактически одновременно реализуя периодический и непериодический планировщики. Основной причиной, по которой непериодические таймеры в ОСРВ до сих пор не используются в качестве несущих таймеров планировщиков, является предельная частота возникновения прерываний. Если у функционального ПО есть возможность взводить таймеры (в ARINC 653 такие интерфейсы есть в каждом наборе сервисов), то пользовательский код может сымитировать ситуацию, в которой асинхронные прерывания будут возникать слишком часто, что приведёт к деградации производительности и ухудшению времени наихудшего случая. При неудачном стечении обстоятельств от повышения латентности могут быть даже нарушены гарантии реального времени, что неприемлемо для ОС с жёстким реальным временем.

Для решения данной проблемы в работе предлагается *совместить подходы периодического и непериодического таймера планировщика в одном*. При этом гарантией фиксированной латентности периодического таймера является квант времени, выбранный в качестве периода между двумя последующими срабатываниями. Для задействования квантов времени в непериодическом таймере, минимальное время срабатывания таймера устанавливается не чаще, чем раз в квант. Для этого временные метки, которые устанавливаются в таймере, необходимо привести к абсолютной временной шкале с делениями в один квант посредством простого округления вверх к ближайшему значению.

Применение данного подхода при построении планировщика позволяет полностью исключить срабатывание «холостых» прерываний от таймера, снизить время кванта в сравнении с периодическим таймером, при этом сохранив латентность на предсказуемом уровне за счёт корректировки кванта планировщика под задачу.

Кроме алгоритмической задачи планирования, для демонстрации жизнеспособности разработанного метода в работе также решена техническая задача реализации данного метода на оборудовании на архитектурах ARM, MIPS и PowerPC. При этом сохранена высокая временная точность и обеспечена защита от остановки планировщика, которые присутствуют в планировщиках на основе периодических таймеров.

Список литературы:

[1] ARINC Specification 653P0-3. Avionics Application Software Standard Inter- face Part 0. Overview of ARINC 653. — AEEC : SAE-ITC, 2021. — 51 стр.

[2] DO-248C. Supporting Information for DO-178C and DO-278A.— RTCA : SC-205, 2011. — 166 стр.

[3] CAST-32A. Multi-core Processors. — The Federal Aviation Admin- istration : Certification Authorities Software Team, 2016. — 23 стр.