МОДЕЛЬ СРЕДЫ МУЛЬТИВЕРСИОННОГО ИСПОЛНЕНИЯ ПРОГРАММНЫХ МОДУЛЕЙ

Котенок А.В.
Сибирский государственный аэрокосмический университет имени академика М.Ф. Решетнева
Красноярск, Россия
st_andy@mail.ru

Мультиверсионное программирование, начиная с 1977 года, используется там, где требуется выдача точного результата в реальном времени. Причем, в отличие от других методик программной отказоустойчивости, требование к времени выдачи результата является более жестким. Очевидно, что при создании систем с использованием данной концепции будут возникать трудности, в первую очередь, связанные с построением среды, которая будет производить мультиверсионное выполнение программных модулей.

Основой построения среды мультиверсионного исполнения (СМВИ) является проработка ее структуры и связей между элементами, так как от этого зависит не только способность расширяемости и удобство в использовании среды, но и такие параметры как *надежность* и *производительность*.

Пользуясь общими принципами построения систем управления, можно выделить следующие модульные составляющие такой системы:

- программные модули, представляющие версии вычислительного процесса;
- модуль сравнения двух результатов;
- файл проекта;
- исполнительное устройство (ИУ);
- датчики;
- среда мультиверсионного исполнения.

Рассмотрев содержание структуры, перейдем к более сложному вопросу: определение типов связей между модулями системы. Попробуем определить, в каких модулях может произойти непредвиденный отказ, последствия которого нужно предотвратить. Файл проекта, загружается в начале работы, и сохраняется при завершении — отказу тут произойти негде. Отказ ИУ и датчиков в любом случае приведет к отказу всей системы. Модуль сравнения двух результатов содержит довольно простой алгоритм и просто хорошим тестированием этого модуля можно свести вероятность отказа в нем к нулю. Следовательно, связь этих модулей со средой исполнения можно реализовать через статические или динамические библиотеки. Таким образом, единственное «слабое звено» нашей системы – это версии, так как в них производятся сложные вычислительные операции. Здесь возможны несколько вариантов взаимодействия со средой (представлены, по порядку возрастания потребности в памяти).

1. **Статическая компоновка: из исходных кодов, либо с использованием статических библиотек**
2. **Компоновка с использованием динамических библиотек**
3. **Компоненты выполнены в отдельных исполняемых модулях**

Для запуска проекта необходимо запустить все эти модули. Делать это вручную и нецелесообразно, так как это ставит в зависимость производительность среды от оператора. Поэтому следует создать специальные программы-агенты, следящие за запуском всех модулей и перезапускающие их при необходимости. При использовании такой модели для исполнения всех моделей на одном компьютере, агентов можно встроить в саму среду исполнения. А при сетевом распределении модулей встроить агентов в среду очень сложно, так как возникнут трудности с отслеживанием состояния процесса на удаленной машине и его удаленного перезапуска.

Данную модель можно разделить на две, по виду взаимодействия между исполняемыми модулями:

A) через общие участки памяти (например, файлы)
B) через сетевой протокол

С учетом разнообразия моделей можно сделать вывод о нереальности создания действительно универсальной среды мультиверсионного исполнения программных модулей, в частности, из-за противоречивости требований платформенной независимости и максимальной надежности.

Подводя итог, хотелось бы резюмировать ряд требований к исходному коду для того, чтобы приблизить конкретную реализацию среды исполнения к универсальной. А именно:

- использовать открытый исходный код
- все обращения к функциям ОС вынести в отдельный модуль, и сделать его функционирование прозрачным для СМВИ
- использовать язык C или C++