Ожидаемый конечный результат проекта. Универсальная программная среда, которая без дополнительного программирования (без написания кода):

а) настраивается на объект и требуемые модели,

б) автоматически управляет итерационными запусками моделей для их взаимного уточнения.

Область использования результатов проекта. Проектирование ГПС в машиностроении и приборостроении.

Основная идея. Предлагается применить принципы построения и функционирования экспертных систем к моделирующему программному обеспечению.

Будем ориентироваться на экспертные системы продукционного типа с представлением знаний в виде правил.

Структура правила: условия (антецеденты) => действия (консеквенты).

Знания о ГПС заключены в моделях. Каждую модель будем рассматривать как одно «большое» правило. Совокупность моделей даст экспертную систему, заполненную знаниями об ГПС.

Каждая модель предназначена для решения своих специфических задач. Вместе с тем, интерфейс всех моделей должен быть выполнен по единому шаблону. Кроме того, все модели должны отражать одинаковый набор состояний оборудования ГПС. В этом случае модели могут использоваться как сменные модули. Аналогом механизма логического вывода экспертной системы в нашем программном обеспечении будет выступать единый и неизменный алгоритм итерационного запуска моделей и обмена уточняющей информацией.

При таком подходе программное обеспечение будет единообразно и автоматически функционировать вне зависимости от вида и состава использующихся моделей.

УДК 681.3

Построение модели многопоточного параллельного приложения

Прихожий А.А., Карасик О.Н.
Белорусский национальный технический университет

Построение модели многопоточного приложения, ориентированной на оптимизацию исполнения потоков на ядрах процессора, требует учета ряда факторов. Важнейшими являются организация и структура исходного алгоритма и программного кода, объем заложенного в него потенциального параллелизма, архитектура многоядерной системы,

принципы построения операционной системы, управляющей выполнением многопоточного приложения. Анализ исходного алгоритма ставит целью выполнение экстракции параллелизма в полном объеме, разбиение алгоритма на параллельно выполняемые фрагменты, выявление взаимодействия и информационных зависимостей между фрагментами. Результаты анализа позволяют учесть особенности архитектуры многоядерной системы и найти оптимальное распределение фрагментов по потокам и оптимальное назначение потоков на ядра, что приводит к построению эффективной синхронизации потоков.

Модель оптимального планирования выполнения потоков сильно зависит от типа применяемой операционной системы: с кооперативной или вытесняющей многозадачностью. В первом случае задача планирования решается легче, во втором – тяжелее. Причина кроется в том, что система с вытесняющей многозадачностью сама планирует распределение общих ресурсов между потоками нескольких исполняемых приложений, при этом приложения не видят и не могут оказать влияния на распределение ресурсов. Построение полной картины совместного выполнения всех потоков, как исследуемых, так и фоновых, является сложной задачей.

Данные для анализа жизненных циклов потоков можно получить в операционной системе Windows с помощью подсистемы ETW (Event Tracing Windows). Эта подсистема высокоскоростного логирования пользовательской и системной информации используется нами для анализа жизненно важных событий в операционной системе. Для потока в ОС Windows такими событиями являются ContextSwitch и ReadyThread. Используя данные от этих событий, мы получаем точное представление о выполнении потока (когда он действительно выполнялся, когда его выполнение было прервано ОС, а управление передано другому потоку, когда поток вошел в синхронизационную блокировку и т.д.). Полное представление о жизненном цикле потока подсказывает, какую модель планирования, распределения и коммуникации необходимо использовать в распараллеливаемом приложении в зависимости от среды его исполнения.

УДК 681.3

Планирование решения задач в Grid-системе

Прихожий А.А., Фролов О.М., Шунько М.Г.
Белорусский национальный технический университет

Современные GRID-системы используют большое многообразие планировщиков задач. Планировщики могут быть централизованными, иерархическими, децентрализованными, адаптивными. В зависимости от