СТРУКТУРНАЯ ОПТИМИЗАЦИЯ ПРОГРАММНО-АППАРАТНОГО КОМПЛЕКСА ВИРТУАЛЬНЫХ ЛАБОРАТОРИЙ С ПРИМЕНЕНИЕМ МЕТОДОВ ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ И НЕЧЕТКОЙ ЛОГИКИ

В.В. Девятков (Казань), Д.В. Жевнерчук, А.В. Николаев (Чайковский)

Введение

Задача планирования загрузки программно-аппаратных комплексов (ПАК) существует давно. Есть множество методов [1,2], которые, как правило, эффективны в случае ряда ограничений: количество вычислительных узлов, количество пользователей, вид вычислительного процесса и др. В работе [3] рассматривается ПАК, представляющий собой совокупность виртуальных лабораторий, функционирующих в сети Интернет, далее открытое виртуальное исследовательское пространство (ОВИП). ОВИП строится на основе множества программно-аппаратных платформ, расширяем по виртуальным лабораториям и обладает свойством масштабируемости. Кроме того, считаем, что ОВИП не ограничено по количеству вычислительных узлов и пользователей. Выделены два основных этапа планирования загрузки ОВИП: планирование структуры ПАК и формирование расписания обслуживания заявок. В докладе рассмотрена гибридная модель структурной оптимизации ОВИП, включающая блоки имитации загрузки и нечеткую подсистему оптимизации структуры ОВИП.

Выбор методов имитационного моделирования обусловлен высокой степенью сложности ОВИП, наличием стохастических процессов и способом представления ОВИП как системы массового обслуживания. Блок имитации позволит оценить характеристики текущей структуры ОВИП.

Нечеткие методы позволят уменьшить вычислительную сложность задачи планирования ОВИП.

Основные положения

Рассмотрим общую структуру ОВИП. Аппаратным и программным ресурсом будем называть соответственно совокупность всех компьютеров, зарегистрированных в ОВИП и инструментального программного обеспечения, используемого для решения научных задач в ОВИП. Введем классификацию групп вычислительных узлов ОВИП. Подсетью физического уровня (ПФУ) считаем совокупность компьютеров, подключенных к ОВИП через один компьютер или точку входа. Считаем, что в ОВИП один компьютер может принадлежать только одной ПФУ. Подсеть логического уровня (ПЛУ) считаем совокупность компьютеров, на которых установлен одинаковый программный ресурс. Два любых компьютера подсети ОВИП логического уровня могут принадлежать одной подсети физического уровня или двум разным подсетям физического уровня. На рис. 1 показана связь между подсетями физического и логического уровня. Заштрихованная область представляет собой совокупность компьютеров с одинаковым программным обеспечением.

Доступ пользователя к виртуальной лаборатории предоставляется по заявке, в которой указываются требования к запрашиваемому ресурсу. Различаем 3 типа заявок:

- заявка типа 1 поступает на обслуживание без предварительного планирования, без гарантированной производительности,
- заявка типа 2 поступает на обслуживание с предварительным планированием, с гарантированной производительностью,
- заявка типа 3 поступает на обслуживание с предварительным планированием, с полным использованием ресурса вычислительного узла.

С учетом типов заявок введем понятие подсети логического уровня ОВИП с определенным типом доступа к ресурсу (ПЛУД), представляющую собой совокупность вычислительных узлов, обслуживающих заявки одного типа.

Система обработки экспериментов включает в себя N вычислительных узлов, разделенных на m подмножеств: $K = \{K_1, K_2, \ldots, K_m\}$. На K_i – ом подмножестве узлов установлена определенная группа программного обеспечения. Физически узлы, попавшие в одинаковые подмножества, могут находиться в разных локальных сетях или отдельных машинах, удаленных друг от друга. Внутри каждого K_i подмножества выделяют 3 типа групп узлов: группы с негарантированной производительностью (Гр1), группы с гарантированной производительностью (Гр2), группы с полным процессорным временем (Гр3). Каждый узел можно классифицировать по принадлежности: к ПФУ, ПЛУ, ПЛУД.

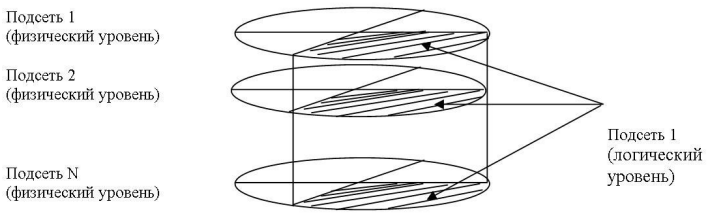

Рис. 1. Связь подсетей физического и логического уровня

Производительность системы будет характеризоваться квантом времени, выделяемым операционной системой вычислительного узла на отдельный процесс. На практике удобно оперировать величинами минимального количества одновременно запущенных процессов на узле группы i (min_{pi}) и максимального количества одновременно запущенных процессов на узле группы i (max_{pi}).

Параметры min_{pi} и max_{pi} подбираются экспериментальным путем. Вводится стоимость загрузки вычислительного узла каждой группы одним процессом s за период времени t_{min}, который является минимальным временем использования. В качестве t_{min} берется академический час (45 мин).

Гибридная модель оптимизации ОВИП

На рис. 2 представлена схема гибридной модели оптимизации структуры и расписания ОВИП.

Для получения характеристик текущей конфигурации ОВИП на вход блока имитации подаются следующие параметры:

- поток заявок – закон распределения для времени прихода заявок;
- $\vec{R}$ – вектор ресурсоемкости для указания класса сложности заявки;
- $\vec{P}$ – вектор производительности вычислительного узла;
- $\overrightarrow{conf_0}$ – вектор конфигурации программно аппаратного комплекса (распределение вычислительных узлов по группам производительности);
- $\vec{\Delta T}$ – вектор периодов, в течение которых должна работать заявка;
- $\overrightarrow{T_0}$ – вектор времен начала выполнения заявок.

После этого выполняются прогоны имитационной модели с целью получения статистики работы каждого элемента ПАК. Блок структурной оптимизации, основанный на использовании нечеткой логики, выполняет анализ структуры предложенной конфигурации ПАК ОВИП и формирует список необходимых перемещений для равно-

мерного распределения нагрузки между вычислительными узлами. После изменения состава вычислительных узлов в группах необходимо провести редактирование и оптимизацию расписания выполнения заявок в каждой группе. После составления нового расписания для заявок оно используется как рабочее на следующей итерации при прогоне модели.

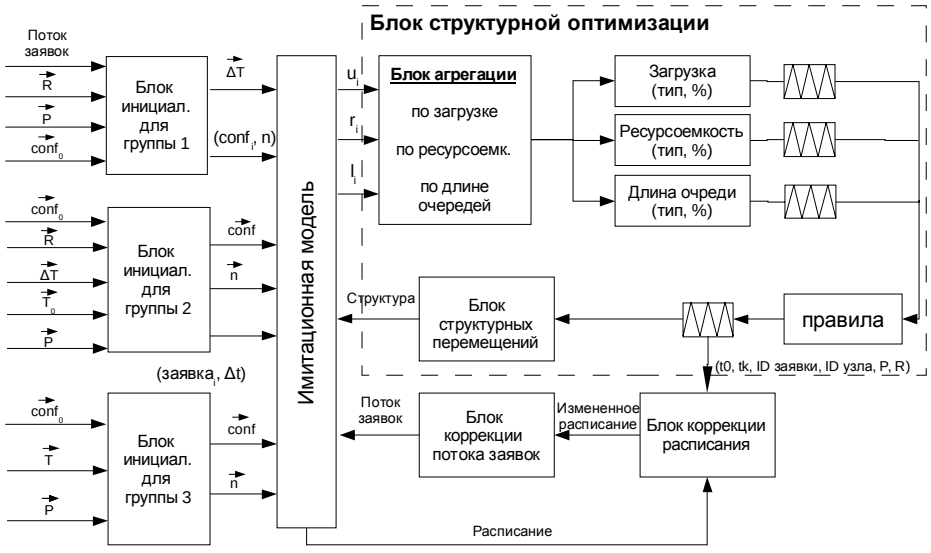

Рис. 2. Схема гибридной модели структурной оптимизации ОВИП

Планирование расписания необходимо проводить только для заявок, поступающих в группу 2 и 3. Для первой группы оптимизирующим фактором является алгоритм обслуживания, который нужно подобрать в зависимости от потока приходящих заявок.

Задача сведена к классической постановке и допускает использование алгоритмов планирования процессов ОС. В такой постановке однозначно эффективный алгоритм сложно спрогнозировать, потому что система расширяема и неизвестно, какие задачи она будет решать в дальнейшем. Поэтому был реализован удобный механизм формирования алгоритмов планирования. Расширяемость обеспечена на уровне требований к описанию алгоритмов на языке GPSS, интерфейса для взаимодействия с клиентской частью модели и шаблонов алгоритмов.

Блок имитации

Технология функционирования имитационного приложения такова, что после ввода всех исходных данных пользователем, должна быть для конкретного эксперимента сгенерирована имитационная модель. Данную роль в имитационном приложении выполняет специальная программа «Генератор моделей».

На рис. 3 показано, как происходит процесс генерации программного кода модели. В целом код модели зависит от конкретных значений данных, введенных пользователем. Исходные данные, формируемые приложением, состоят из данных, вводимых пользователем в процессе диалога и массивов статистики из внешних систем, например, из систем мониторинга. Основой модели является ряд типовых элементарных блоков (шаблонов), которые заранее созданы разработчиком приложения.

На основе всех этих входных данных и в соответствии с реализованной внутренней логикой генератор собирает исходный код модели на языке GPSS World для конкретного эксперимента. Если пользователь изменит исходные данные, изменится и код модели.

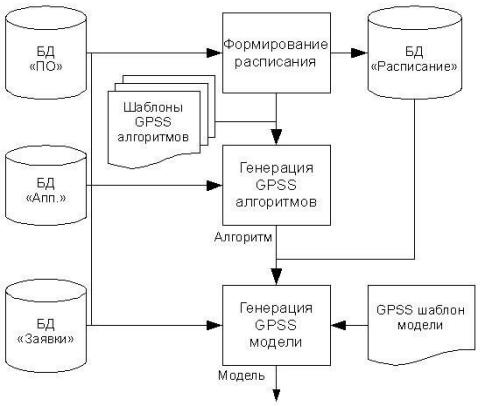

Рис. 3. Общая условная схема генератора моделей

Текст имитационной модели складывается из следующих внутренних элементов (частей): набора команд инициализации матриц с исходными данными по вычислительным узлам, программным ресурсам и виртуальным лабораториям, набора команд определения функций, множества блоков генерации и обработки заявок на ресурсы первой группы производительности, множества блоков генерации заявок на ресурсы, не относящиеся к первой группе производительности (формируется на основе расписания), множества блоков обработки заявок на ресурсы, не относящиеся к первой группе. С помощью сгенерированной модели имитируется функционирование ОВИП определенной конфигурации. Получаемая в итоге имитационного эксперимента статистика заносится в базу данных для анализа нечеткой подсистемой перемещения вычислительных узлов.

Структурная оптимизация

Для оптимизации структуры ПАК используется аппарат нечеткой логики, который на концептуальной схеме (см. рис. 2) представлен блоком структурной оптимизации. На вход блока подаются сведения об узлах, очередях и заявках, получаемые в результате имитационного эксперимента или из БД. После анализа структуры генерируется список узлов для перемещения. Список далее подается в блок структурных перемещений, после чего вновь проводится тестирование имитационной модели. Рассмотрим функционирование блока структурной оптимизации (рис. 4). На первом этапе производится оценка загруженности каждого из вычислительных узлов, класса сложности заявок и загруженности очередей (блоки SUtil, SReq, SQue соответственно). Для этого используется пакет расширения MatLab FuzzyLogic ToolBox. Блоки возвращают числовое значение, определяющее степень загруженности для узлов и очередей в процентах [0..100] и класс сложности заявок в баллах [0..10]. После этого, в блоках подсчета EC_U, EC_R, EC_Q происходит подсчет попадания полученных значений в интервалы по шкале «малый-средний-большой». В результате в БД для каждой группы записывается количество элементов, попавших в определенный интервал, например, 10 узлов малой загруженности, 85 заявок среднего класса сложности и т.д. Следующим этапом является определение общей загруженности группы. Для этого используется блок SGroup (нечеткая логика). Блок возвращает числовое значение загруженности группы в процентах [0..100].

Далее происходит определение количества изымаемых или дополняемых узлов (блок CUmove). В БД для каждой группы записывается количество узлов для добавления [+] или изъятия [-]. Последним этапом является определение того, какие именно узлы в какую группу требуется определить (блок AllocU).

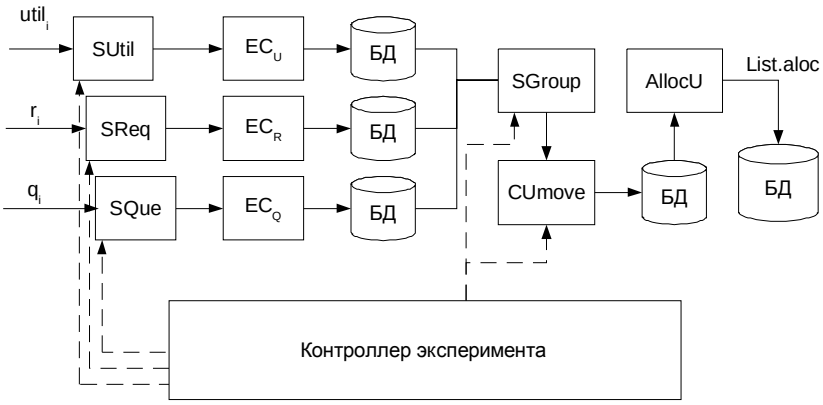

Рис. 4. Блок структурной оптимизации

Логические схемы блоков определения загруженности узлов и очередей и класса сложности заявок представлены на рис. 5.

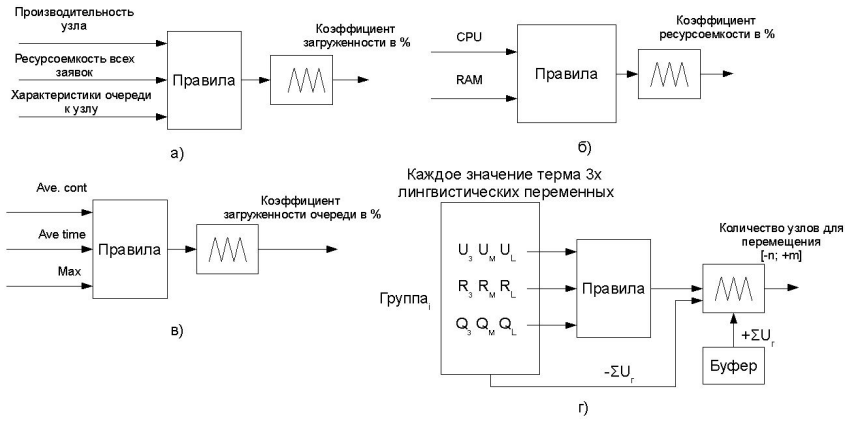

Рис. 5. Блок структурной оптимизации: а – SUtil, б – SReq, в – SQue, г – CUMove

Выводы

Построена система, с помощью которой можно исследовать показатели функционирования ОВИП определенной конфигурации. Система включает ряд алгоритмов планирования заявок на обслуживание и расширяема по ним, что позволяет исследовать эффективность произвольных алгоритмов планирования загрузки ПАК ОВИП. Систему можно использовать для моделирования загрузки ПАК виртуально-лабораторного ресурса любого размера.

Литература

1. **Коффман Э. Г.** Теория расписаний и вычислительные машины / Под ред. Коффмана Э. Г. Пер с англ. Амочкина А. В. М.: Наука, 1984. 336 с.
2. **Костенко В. А.** Оценки сложности и качества различных итерационных алгоритмов построения расписаний// Искусственный интеллект. 2004. С. 101–104.
3. **Николаев А. В.** Методика проведения имитационного эксперимента с моделью открытого виртуального исследовательского пространства// Информационно-математические технологии в экономике, технике и образовании. Сб. мат. 2-й междунар. науч. конф. Екатеринбург: УГТУ-УПИ. Вып. 4. Прикладные аспекты моделирования и разработки систем информационно-аналитической поддержки принятия решений. 2008. С. 253–260. ISBN 978-5-321-01317-5.