ИНФОРМАТИКА

2008 **апрель-июнь** **№ 2**

УДК 681.3.012

А.И. Якимов

МОДЕРНИЗАЦИЯ ПРОГРАММНО-ТЕХНОЛОГИЧЕСКОГО КОМПЛЕКСА ИМИТАЦИИ СЛОЖНЫХ СИСТЕМ BELSIM ДЛЯ ОРГАНИЗАЦИИ РАСПРЕДЕЛЕННЫХ ВЫЧИСЛЕНИЙ

Предлагается модернизация программно-технологического комплекса имитации (ПТКИ) сложных систем на основе MPI-технологии для распределения вычислений. Комплексное использование технологии имитационного моделирования, распределенных вычислений и XML-технологии позволяет построить ПТКИ, интегрированный в автоматизированную информационную систему промышленного предприятия.

Введение

Современная концепция имитационного моделирования хорошо согласуется с моделью параллельных вычислений MPMD (Multiple Program – Multiple Data) [1, с. 9]. Во-первых, сложные системы состоят из параллельно функционирующих компонент на разных уровнях иерархической структуры, что приводит к использованию в имитационной модели (ИМ) множества модулей, одновременно реализующих поведение элементов и подсистем объекта; во-вторых, разнородность элементов и подсистем порождает разнородность применяемых математических схем и, соответственно, разнородность программной реализации модулей ИМ [2, с. 210].

ПТКИ BelSim предназначен для комплексного решения задач моделирования и ориентирован в первую очередь на предприятия с современными ERP-системами управления [3]. Помимо стандартных возможностей ПТКИ BelSim обладает средствами проведения структурного анализа системы и протекающих в ней процессов, эффективного представления сложных многоуровневых систем. Имеется возможность интеграции модели в информационную систему предприятия с целью получения исходных данных для моделирования и использования модели в составе системы управления.

ПТКИ BelSim обладает следующими преимуществами по сравнению с известными системами имитационного моделирования (СИМ):

– развитыми средствами декомпозиции сложных систем на основе концепции IDEF0;

– базовым языком C++ для построения ИМ, что позволяет строить их с использованием стандартных библиотек, не накладывает ограничений на состав и функции компонентов системы, а также обеспечивает более высокую скорость работы ИМ по сравнению с СИМ, использующими другой базовый язык, например Java;

– процессным способом имитации, который является наиболее универсальным и может применяться для построения ИМ сложных систем без ограничений, накладываемых другими способами имитации на модель объекта;

– открытостью интерфейсов компонентов ИМ и СИМ, что позволяет легко изменять состав компонентов ИМ (процессов и активностей) и алгоритмы их функционирования, а также использовать готовые компоненты для построения новых моделей;

– наличием программного обеспечения (ПО) Experimenter для исследования свойств ИМ, планирования (ПО Experiment Designer), проведения и передачи результатов имитационных экспериментов (макросы DesignOf Experiment и ExperimentData) в пакет прикладных программ STATISTICA для статистической обработки.

1. Организация распределенных вычислений в ПТКИ BelSim

Вычислительные эксперименты с моделями функционирования промышленного предприятия реализуются программным модулем Experimenter (рис. 1) и требуют значительных временных затрат, для сокращения которых в ПТКИ BelSim [4] реализована возможность про-

гона модели с использованием ресурсов локальной вычислительной сети. Для этого применяется библиотека MPI-функций обмена данными между процессами, реализованными для языка C++. Программным средством реализации MPI является MPICH, обеспечивающая выполнение всех функций MPI в исполнительной среде [1, с. 28]. Для реализации MPI-версии в программу Experimenter необходимо внести следующие изменения:

1. Добавить новые переменные для получения информации об активном процессе и о программе Experimenter в целом: идентификатор номера процесса – MyId, количество выполняемых процессов – NumProcs, отметка начала времени работы программы – Start.

2. Для слияния XML-файлов с результатами эксперимента добавить функцию mergeXML:

void mergeXML(*wstring fileNameDist, wstring fileNameSrc, int madeMerge*),

где fileNameDist – путь к файлу, в который будет добавлен файл-источник;

fileNameSrc – путь к файлу-источнику;

madeMerge – количество вызовов функции mergeXML.

В MPI-программе после строк определения переменных следуют три обязательные строки:

MPI_Init(&argcs,&argvs);

MPI_Comm_size(MPI_COMM_WORLD,&NumProcs);

MPI_Comm_rank(MPI_COMM_WORLD,&MyId).

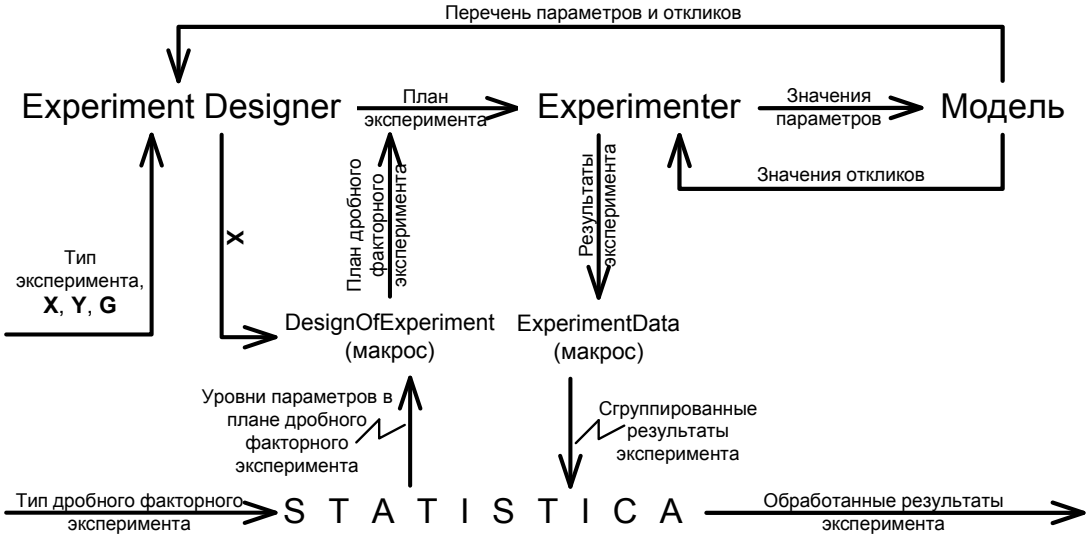

Рис. 1. Схема постановки имитационных экспериментов в ПТКИ BelSim

Обращение к MPI_Init должно быть первым, так как оно инициализирует MPI.

Коммуникатор MPI_COMM_WORLD описывает состав процессов и связи между ними. Вызов MPI_Comm_size возвращает в NumProcs число процессов (прогонов модели), задаваемых пользователем в программе. Вызывая MPI_Comm_rank, каждый процесс выясняет свой номер (rank) в группе, связанной с коммуникатором [1, с. 25].

Для синхронизации процессов используется функция MPI_Barrier, которая блокирует работу вызвавшего ее процесса до тех пор, пока все другие процессы группы также не вызовут эту функцию. Завершение работы функции MPI_Barrier возможно только всеми процессами одновременно [5, с. 111].

Ниже представлен фрагмент программы Experimenter с дополнениями для организации распределенных вычислений:

```
int wmain(int argc, wchar_t *argv[])
clock_t start = clock(); // отмечаем время запуска эксперимента
argvs[1] = "experimenter.exe";
int argcs = 1, MyId/*id текущего процесса*/, NumProcs/*количество процессов*/;
MPI_Init(&argcs,&argvs); // инициализация MPI
MPI_Comm_size(MPI_COMM_WORLD,&NumProcs); // получаем количество процессов
MPI_Comm_rank(MPI_COMM_WORLD,&MyId); // получаем Id текущего процесса
if(MyId == 0) // чтобы не выводить на экран несколько раз одну и ту же информацию
  cout << endl << g_szTitle << endl;
if(MyId == 0) // чтобы не выводить на экран несколько раз одну и ту же информацию
cout << endl;
        if(MyId == 0) {
        cout << endl << L"Тип эксперимента:" << endl;
        switch (spExperimentData->getExperimentType())
        {
        case CExperimentData::AccuracyEvaluation:
                cout << L"  оценка погрешности" << endl;
                break;
        case CExperimentData::TransientAndStabilityAnalysis:
                cout << L"  анализ переходного процесса и устойчивости" << endl;
                break;
        case CExperimentData::SensitivityAnalysis:
                cout << L"  анализ чувствительности" << endl;
                break;
        case CExperimentData::FullFactorial:
                cout << L"  полный факторный эксперимент" << endl;
                break;
        case CExperimentData::FractionalFactorial:
                cout << L"  дробный факторный эксперимент" << endl;
if(MyId == 0) // первый поток записывает свои результаты в целевой файл
spExperimentData->save(argv[3]);
else // остальные записывают результаты во временный файл
spExperimentData->save(argv[3] + uint2wstring(myid) + wstring(L".xml"));
cout << L" OK" << endl;
MPI_Barrier(MPI_COMM_WORLD); // дожидаемся, пока все потоки запишут свои результаты, чтобы можно было начать их собирать
if(MyId == 0) {// собирает все файлы только первый поток
for(int i = 1; i < numprocs; i++) {
CExperimenter::mergeXML(argv[3], argv[3] + uint2wstring(i) + wstring(L".xml"), i);
// записываем результаты потока с номером i
std::string s = wstring2string(argv[3] + uint2wstring(i) + wstring(L".xml"));
remove(s.c_str()); // удаляем временный файл
}
}
float duration = (float) (clock() - start) / CLK_TCK;
cout<<"Time - "<<duration<<"sec"<<endl;         // выводим время работы программы
void CExperimenter::mergeXML(wstring fileNameDist, wstring fileNameSrc, int madeMerge)
// копирует результаты из файла с именем fileNameSrc в файл fileNameDist
void CExperimenter::run(SP<CModel> model, SP<CModelData> modelData,
if(runNo % NumProcs == MyId) { // запускаем только те эксперименты, которые должен запустить наш процесс
```

Результаты прогонов модели записываются в отдельный XML-файл. Слияние результатов эксперимента из отдельных временных XML-файлов в одном целевом файле реали-

зуется функцией mergeXML. При этом одному из процессов присваивается нулевой идентификатор и он создает целевой файл. Другие процессы создают временные файлы. Функция mergeXML создает структуру для XML-документов docSrc и docDist. Результаты из docSrc добавляются в docDist, который сохраняется как целевой файл с результатами эксперимента.

2. Исследование эффективности распределения вычислений в ПТКИ BelSim

ПТКИ BelSim позволяет моделировать задачи не только для проблем промышленного предприятия, но и для смежных предметных областей. Типовой для оптимизационных задач является задача об оптимизации грузоперевозок на автотранспортном предприятии [6, с. 110–112], которая формулируется следующим образом.

Предприятие имеет N заказов на перевозку грузов. Каждый заказ (j=1, ..., N) характеризуется объемом (O_j), сроком выполнения (Sr_j) и координатой на карте местности ($||S_{fh}||$ – матрица расстояний между базой и всеми пунктами назначений; $||V_{fh}||$ – матрица средних скоростей между всеми пунктами назначения и базой; f, h=0, ..., N).

Автотранспортное предприятие предоставляет услуги по грузоперевозкам в территориальных пределах, задаваемых картой местности. Автопарк предприятия представлен некоторым количеством грузовых машин (M). Каждая машина (i=1, ..., M) имеет следующие характеристики: грузоподъемность (T_i), расходы на топливо (TO_i) и оплату времени работы водителя (Z_i). Для полного учета времени работы водителя существуют нормативы на скорость погрузки (Vp) и разгрузки (Vr). Заказ всегда выполняется полностью за одну поездку. При превышении сроков выполнения некоторого заказа с предприятия взимается пеня (Zpr). Требуется составить план грузоперевозок ($||A_{ig}||$ – матрица распределения заказов по машинам, g=1, ..., 2*N+1; i=1, ..., M), при котором выполняются все заказы и при этом общие затраты автотранспортного предприятия минимальны:

$$\sum_{i=1}^{M}\left(\sum_{j=1}^{2*N} S_{A_{ij}A_{ij+1}}\cdot TO_i + \left(\sum_{j=1}^{2*N} S_{A_{ij}A_{ij+1}}/V_{A_{ij}A_{ij+1}} + \sum_{j=1}^{2*M+1} O_{A_{ij}}\cdot(Vp+Vr)\right)\cdot Z_i\right) + Zpr\cdot Tpr \to \min. \quad (1)$$

Задача о грузоперевозках принята для оценки эффективности распределенных вычислений. В качестве экспериментальной базы использована локальная вычислительная сеть из трех компьютеров (табл. 1).

Таблица 1

Экспериментальная база при исследовании распределенных вычислений

Компьютер	Процессор	ОЗУ, МБ	Сетевая карта	Операционная система
PC1: Compaq Evo N610c	Intel Pentium 4 / 2,0 ГГц	512	Ethernet / 100 Мбит/с; TCP/IP	Windows XP Professional (версия 2002)
PC2: HP Compaq nc 6000	Intel Pentium M / 1,6 ГГц			
PC3: Compaq Evo N620c	Intel Pentium M / 1,4 ГГц			

Для оценки эффективности проведения эксперимента с распределенными вычислениями определена величина T_{ij} – длительность i-го эксперимента на j-м компьютере (i – порядковый номер эксперимента, i=1, …, 3; j – идентификатор компьютера, j=PC1 ,…, PC3) при заданном количестве прогонов модели в i-м эксперименте N_i. Результаты проведенного исследования представлены в табл. 2.

Таблица 2

Результаты экспериментов с распределенными вычислениями в ПТКИ BelSim

Количество прогонов модели, N_i	Продолжительность эксперимента T_{ij}, с					
	на одном PC	на двух PC		на трех PC		
	T1(PC1)	T2(PC1)	T2(PC2)	T3(PC1)	T3(PC2)	T3(PC3)
60	21,99	12,22	12,81	8,17	8,24	8,91
100	35,73	19,11	19,90	13,26	13,34	14,38
200	71,69	35,94	37,28	26,18	26,10	28,07
400	146,64	73,14	76,29	54,30	54,44	58,37
600	227,89	108,81	114,38	79,46	79,61	86,12

Эффективность распределения вычислений оценена по следующему выражению:

$$T_{i\max} = \max_{\forall PC_j}(T_i(PC_j)), \tag{2}$$

где $T_{i\max}$ – максимальное значение продолжительности эксперимента, определяемое по наибольшему значению $T_i(PC_j)$ для наименее производительного компьютера. Анализ эффективности распределения вычислений в соответствии с выражением (2) дан на рис. 2.

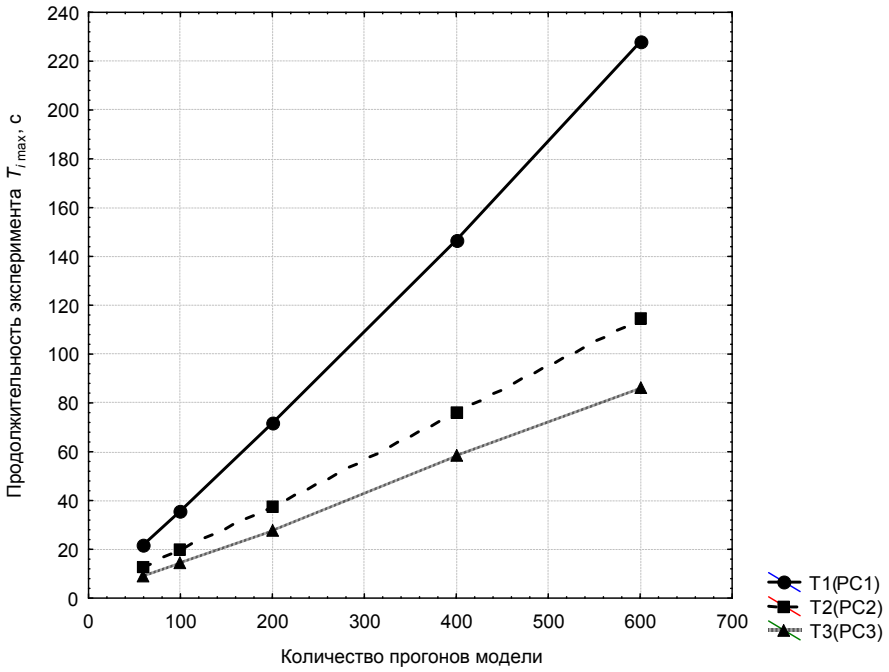

Рис. 2. Эффективность распределения вычислений в ПТКИ BelSim

Например, при сравнении экспериментов с N_i=400 на одном и двух компьютерах продолжительность прогонов модели сокращается в 1,9 раз, т. е. почти в два раза. На трех компьютерах продолжительность эксперимента по сравнению с одним компьютером сокращается в 2,5 раза.

В соответствии с сетевым законом Амдала [1, с. 12–13] при общей эффективности вычислений не наблюдается прямо пропорционального снижения продолжительности экспериментов с увеличением числа используемых компьютеров в сети. Это объясняется, в частности,

тем, что для синхронизации процессов при распределенных вычислениях применяется функция MPI_Barrier, которая блокирует работу вызвавшего ее процесса до тех пор, пока все другие процессы группы также не вызовут эту функцию. Завершение выполнения этой функции возможно только всеми процессами одновременно.

Заключение

Комплексное использование технологии имитационного моделирования, распределенных вычислений и XML-технологии позволяет построить программно-технологический комплекс имитации сложных систем, интегрированный в автоматизированную информационную систему промышленного предприятия. При этом появляется возможность прогнозирования динамики функционирования предприятия, определения стратегии и тактики его развития с учетом данных, накопленных в хранилищах информационной системы, и их эффективной обработки. Этот вывод подтвержден авторским опытом практической разработки динамической производственно-экономической модели завода органического синтеза.

Список литературы

1. Шпаковский, Г.И. Программирование многопроцессорных систем в стандарте MPI / Г.И. Шпаковский, Н.В. Серикова. – Минск: Изд-во БГУ, 2002. – 323 с.

2. Олзоева, С.И. Особенности автоматизированного распределения вычислительного процесса для имитационного моделирования систем / С.И. Олзоева // Высокопроизводительные параллельные вычисления на кластерных системах: материалы Четвертого Междунар. науч.-практ. семинара. – Самара: СГАУ, 2004. – С. 210–214.

3. Гладкова, И. Глобализация. Открытый мир. Коллаборативная модель управления предприятиями / И. Гладкова // Оборудование. – 2002. – № 10 (70). – [Электронный ресурс]. – Режим доступа: http://home.expert.ru/oborud/02/10-02/data. – Дата доступа: 24.12.2007.

4. Якимов, А.И. Имитационное моделирование в ERP-системах управления / А.И. Якимов, С.А. Альховик. – Минск: Бел. наука, 2005. – 198 с.

5. Букатов А.А. Программирование многопроцессорных вычислительных систем / А.А. Букатов, В.Н. Дацюк, А.И. Жегуло. – Ростов-на-Дону: ЦВВР, 2003. – 208 с.

6. Альховик, С.А. Генетический алгоритм в задаче оптимизации плана грузоперевозок / С.А. Альховик, А.В. Сазоненко, А.А. Ковалевич // Известия Гомельского государственного университета им. Ф. Скорины. – 2006. – 4 (37). – С. 110–112.

Поступила 17.01.08

Белорусско-Российский университет,
Могилев, пр. Мира, 43
e-mail: ykm@tut.by

A.I. Yakimau

TECHNOLOGY OF SOFTWARE PACKAGE BELSIM MODERNIZATION FOR ORGANIZATION OF DISTRIBUTED CALCULATIONS

A modernization of software package on the basis of MPI-technology for distribution of calculations is offered. Multiple use of simulation technologies, distributed calculations and XML-technologies allow to build a software package for imitations of complex systems integrated into automated information system of industrial enterprise.