Научно-исследовательская работа по договору на тему «Разработка технологии реализации параллельных версий расчетных модулей».

Руководитель темы: к.ф.-м.н. Е.С. Кирик

Работа выполнялась в рамках развития программного комплекса «Сигма ПБ», предназначенного для решения задач по оценке пожарной безопасности зданий.

Целью НИР являлось расширение области применения программного комплекса «Сигма ПБ» на крупномасштабные расчеты, предполагающие десятки тысяч эвакуирующихся (например, на стадионах).

1 *Реализация поддержки многопроцессорных вычислений модулем расчета эвакуации*

Моделирование движения людей в многоэтажном здании естественным образом разбивается на движение большого количества людей по этажу и сравнительно небольшого (в силу малой пропускной способности) количества людей по межэтажным лестницам. Таким образом, разнесение расчета по этажам на разные процессы с передачей между ними людей, перемещающихся с этажа на этаж, как решает проблему возможной нехватки памяти за счет распределения полей расстояний между несколькими процессами, так и позволяет более полно загрузить ядра процессора и тем самым ускорить счет. Фактически это функциональная декомпозиция по числу этажей. Для реализации здесь оптимально подходит парадигма MPI; она позволяет легко оперировать богатым набором функций обмена сообщениями, и, при должной настройке окружения, запускать расчет различных этажей не только в разных процессах, но и на разных компьютерах. В нашеслучае использована свободно распространяемая реализация MPICH2 [**Ошибка! Источник ссылки не найден.**], существующая для платформы Windows. Схема MPI-запуска модуля расчета представлена на рисунке 1.

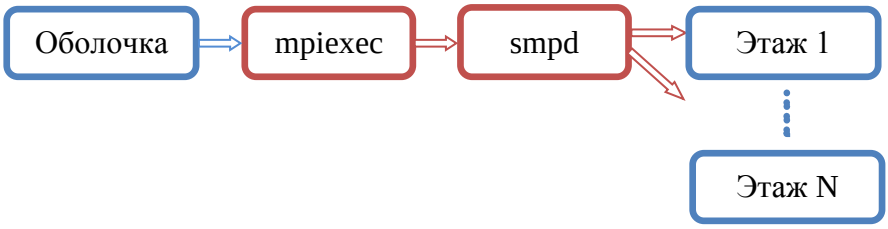

Рисунок 1 - Крупноблочное распараллеливание

Запуск происходит следующим образом:

1) пользователь с помощью оболочки (*wrapper.exe*) настраивает параметры расчета, выбирает входные файлы и формирует файл настроек, содержащий всю необходимую информацию;
2) по команде пользователя оболочка запускает программу *mpiexec.exe* с необходимыми параметрами. Например, команда *mpiexec.exe -localonly -n 6 "D:\Projects\Fire\evacalc.exe" "D:\Projects\Fire\obj\G6\eva\building.ini" MPI* обеспечивает запуск расчета эвакуации на локальном компьютере шестью MPI-процессами *evacalc.exe* (по одному на этаж здания, описанного в файле *building.ini*);
3) автоматически запускается программа *smpd.exe*, обеспечивающая межпроцессное взаимодействие, запускает собственно процессы *evacalc.exe* и начинается расчет;
4) в процессе расчета процессы используют коллективные взаимодействия и коммуникации «точка-точка» для обмена между собой людьми и различными данными. Передача людей производится один раз на каждом временном шаге. В конце результаты расчета («треки» движения людей, статистические данные) собираются на нулевом процессе, объединяются и сохраняются на диск для работы других модулей системы;
5) по окончании расчета программы *mpiexec.exe* и *smpd.exe* автоматически прекращают работу, оболочка получает статус завершения и информирует пользователя о результатах.

Еще одно важное достоинство этой реализации состоит в том, что при запуске всех процессов на том же компьютере, что и оболочка, появляется возможность визуального контроля хода расчета. Скриншот процесса расчета показан на рисунке 2.

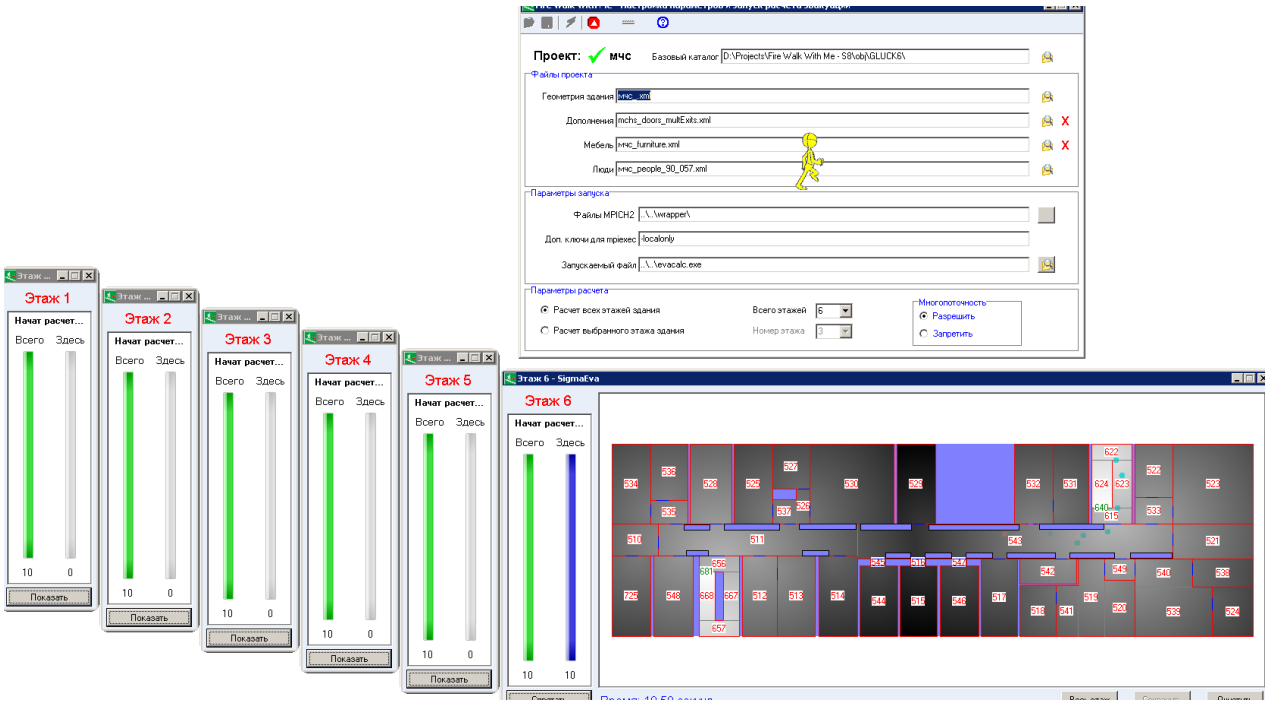

Рисунок 2 - Пример расчета

2 *Мелкозернистое распараллеливание*

Наряду с функциональной декомпозицией реализованная математическая модель допускает и распараллеливание при расчете эвакуации с каждого этажа. Рассмотрим в общих чертах основной вычислительный цикл модели.

1. Для каждого человека определяется предполагаемое направление движения и скорость на текущем временном шаге. При этом учитывается положение человека, окружающих его людей и препятствий. Новое положение фиксируется как предполагаемое.
2. Для каждого человека проверяется, не возникло ли пересечения его нового положения со старым положением какого-нибудь другого человека. Если это случается, человек остается на этом временном шаге на прежнем месте.
3. Все новые положения всех людей, прошедших проверку, проверяются на взаимное пересечение, производится разрешение коллизии путем оставления на прежнем месте кого-то одного из пары.
4. Выполняется фиксация новых положений переместившихся людей.
5. Выполняются прочие операции с людьми, обмен между этажами, разрешение коллизий на межэтажных переходах.

Очевидно, что пункты 1, 2 и 4 могут выполняться одновременно различными нитями для различных групп людей, так как они не содержат условий для взаимной блокировки на запись по данным; однако п.4 выполняется быстро и распараллеливать его нецелесообразно. Распараллеливание пунктов. 3, 5 неэффективно из-за наличия большого количества взаимных блокировок.

Для работы с нитями использовался класс Delphi *TThread*, упрощающий порождение, запуск/остановку и уничтожение нитей. Очевидно, чтобы порождать и уничтожать нити на каждом временном шаге нерационально, поэтому нити порождаются однократно в должном количестве, определяемом в зависимости от числа ядер, доступных приложению, и количества процессов, работающих над зданием. Кроме того, при уменьшении количества людей в здании количество активных нитей уменьшается так, чтобы каждая обрабатывала не менее 30 человек (пока не останется только одна), с тем, чтобы повысить эффективность параллелизма.

Такой алгоритм распараллеливания диктует необходимость синхронного исполнения всех нитей, то есть после выполнения пунктов 1, 2 исполняющие нити должны синхронизироваться на барьере друг с другом и с основной нитью. Реализация барьера была взята из свободно распространяемой библиотеки pthreads-win32, для использования которой пришлось конвертировать необходимые константы и декларации из файла *pthreads.h* в формат Delphi.

На рисунке 3 приведена блок-схема реализации описанного распараллеливания.

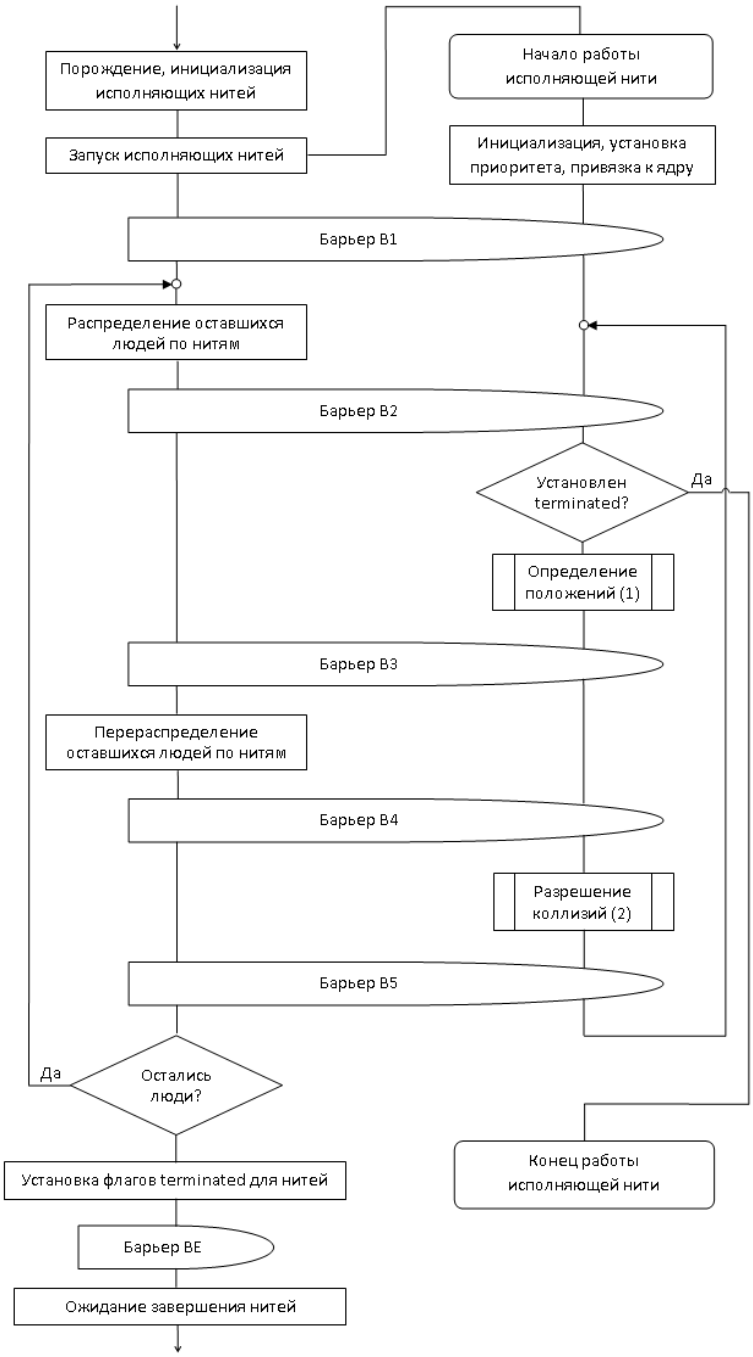

Рисунок 3 - Блок-схема многонитиевого исполнения основного цикла программы

Для простоты приведена только одна исполняющая нить и исключены действия программы, не относящиеся к параллелизму.

3 *Ускорение и эффективность*

Сочетание описанных выше подходов позволяет полностью использовать ресурсы компьютера с многоядерным процессором (процессорами). Так, при расчете эвакуации из

многоэтажных зданий в основном работает крупноблочное распараллеливание, а количество нитей ограничено одной-двумя на процесс из-за нехватки ядер; при расчете одного этажа, напротив, количество нитей будет соответствовать количеству процессорных ядер.

Эффективность крупноблочного распараллеливания определяется следующим:

1. Перед началом моделирования процессами полностью независимо друг от друга производится считывание/распаковка или построение полей расстояний. Параллельная эффективность при выполнении этой существенной трудоемкой операции, особенно в случае построения полей, на этом этапе практически равна единице (если количество физических ядер там, где производится расчет, не меньше количества этажей).

2. Первая часть любого расчета эвакуации состоит в движении людей по этажам к лестницам и выходам. На этом этапе взаимодействие между процессами минимально, соответственно, эффективность близка к единице (с тем же уточнением, что и в предыдущем пункте).
3. Вторая часть состоит в преимущественном движении по лестницам (с заторами или без) и движении по этажу, на котором расположены выходы, от лестниц до выходов. Здесь процессы сравнительно активно обмениваются данными о переходящих с этажа на этаж людях, эффективность падает, но незначительно, так как пропускная способность лестниц, как правило, невелика.

С учетом также того, что без декомпозиции по пространству некоторые расчетные конфигурации просто не поместились бы в память одного 32-разрядного процесса, можно считать эффективность крупноблочного распараллеливания высокой, а использование такого подхода – полностью оправданным.

Эффективность многонитевого подхода проверялась путем прямых замеров.

Для экспериментов выбрана задача эвакуации полностью заполненного большого зала БКЗ (Большой концертный зал) г. Красноярска. В зале 8 выходов, перед началом эвакуации в помещении находится 1025 человек.

Первый эксперимент был поставлен на компьютере с 4-ядерным процессором Intel Core i5-760 под управлением ОС Windows 7 Pro x64. На рисунке 4 приведены графики,

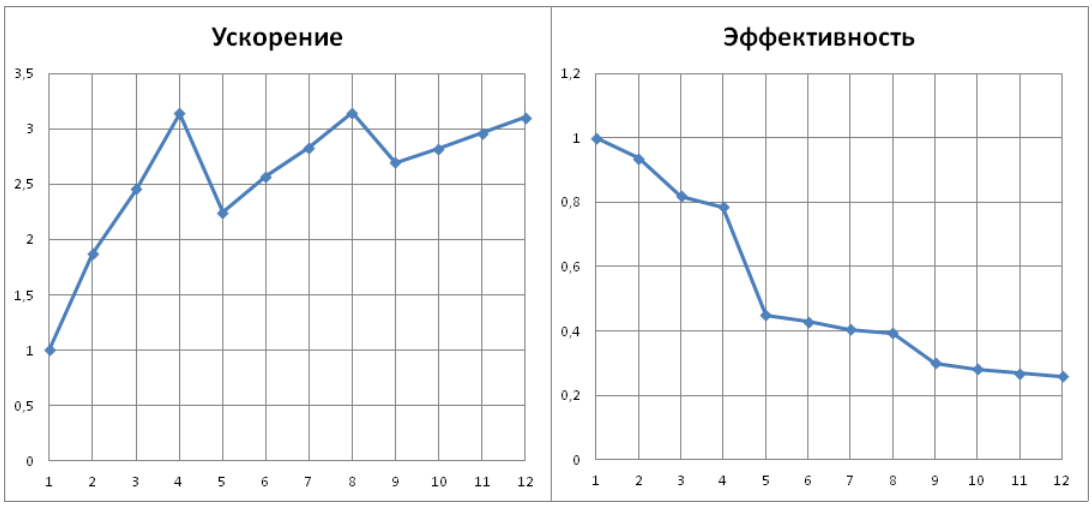

Рисунок 4 - Ускорение и эффективность на 4 ядрах

показывающие ускорение и эффективность для такой конфигурации в зависимости от количества исполняющих нитей. Каждый расчет выполнялся три раза, замеренные времена решения задачи усреднялись. Второй эксперимент был поставлен на сервере с 4-мя 4-ядерными процессорами Quad-Core AMD Opteron-8378 (всего 16 процессорных ядер) под управлением ОС Windows Server 2008 R2. Рисунок 5 отображает его результаты.

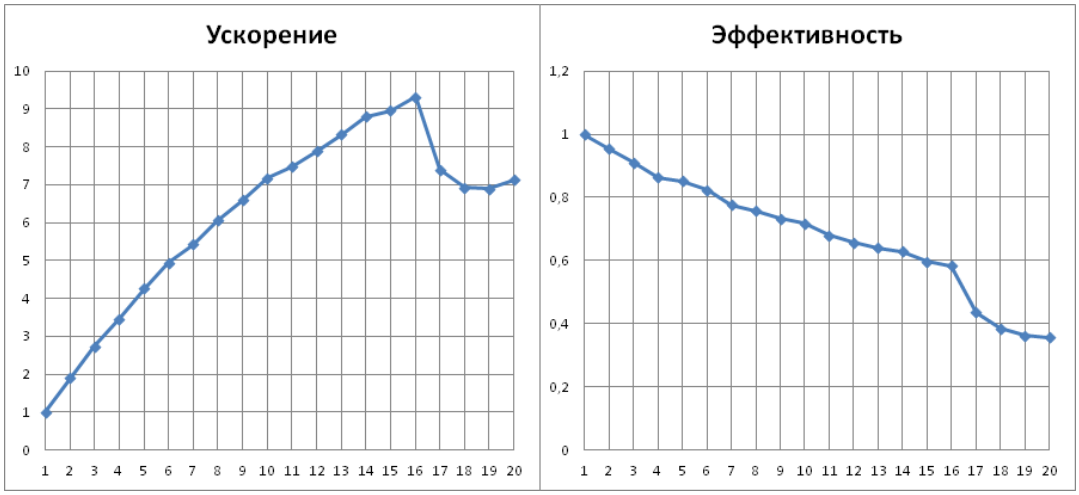

Рисунок 5 - Ускорение и эффективность на 16 ядрах

На графиках в диапазоне от единицы до количества ядер видна неплохая динамика эффективности, с ожидаемой деградацией, доходящей до 80% в первом случае и до 60% во втором. Ускорение достигает соответственно трех и десяти, что с точки зрения конечного пользователя можно считать существенным приростом. После того, как количество нитей становится больше количества ядер, происходит провал, но не столь сильный, как можно было бы ожидать. В первом эксперименте на 8 и 12 нитях мы даже

видим практически те же значения ускорения (и, соответственно, времени выполнения), то же самое наблюдается при расчете на 32 нитях во втором эксперименте (не показано на графике). Вместе с тем, эффективность подобного расчета мала, поэтому в использовании количества нитей, большего, чем количество доступных ядер, нет практического смысла.

ЗАКЛЮЧЕНИЕ

В рамках выполнения НИР были выполнены поставленные задачи. Ускорение расчетного времени реализовано за счет крупноблочного и мелкозернистого параллелизмов, которые допускаются математической моделью. Крупноблочное распараллеливание реализовано разнесением между процессами расчетов движения людей на отдельных этажах с передачей данных о перемещениях с этажа на этаж. Этот подход решает проблему возможной нехватки памяти за счет распределения полей расстояний между несколькими процессами и позволяет более полно загрузить ядра процессора. Мелкозернистый параллелизм реализован при расчете движения людей на каждом этаже. Нити порождаются однократно в количестве, определяемом в зависимости от числа ядер, доступных приложению, и количества процессов, работающих над зданием.

Публикации в научных изданиях по теме НИР

1. *Кирик Е.С., Дектерев А.А., Литвинцев К.Ю., Харламов Е.Б., Малышев А.В.* Математическое моделирование эвакуации при пожаре // Математическое моделирование. – Т. 26 (1), 2014. – С. 3-16. (http://elibrary.ru/item.asp?id=21276919)
2. *Kirik E., Malyshev A.* A discrete-continuous agent model for fire evacuation modeling from multistory buildings // Civil Engineering and Urban Planning III. – CRC Press, 2014. – P. 5-9. (http://dx.doi.org/10.1201/b17190-3)