Рисунок 1 иллюстрирует свойство состоятельности ОМП для модели (1). Видно также, что при $T > 30$ среднеквадратическая погрешность оценивания парметров для модели (1) меньше, чем для МНК-оценок аппроксимационной модели (7).

Литература

1. *Mariella L., Tarantino M.* Spatial temporal conditional Auto-Regressive Model: A New Autoregressive Matrix // Austrian journal of Statistic. 2010. Vol. 3. P. 223–244.
2. *Боровков А. А.* Математическая статистика. М., 1984.
3. *Альсевич В. В., Крахотко В. В.* Методы оптимизации: упражнения и задания: учебное пособие. Минск, 2005.

КОНВЕЙЕРНЫЕ ВЫЧИСЛЕНИЯ С ОБРАТНЫМИ СВЯЗЯМИ

И. В. Жураховский

Конвейер – один из способов обработки данных, заключающийся в том, что программа разбивается на последовательность независимых стадий, каждая из которых обрабатывается на своем узле конвейера. Мы рассмотрим модель конвейера, узлами которого служат процессы, поддерживающую сложную структуру программы.

Использование конвейера на процессах имеет свои преимущества. Если есть в наличии несколько ядер процессора, то можно распределить узлы-процессы между ними, повысив скорость обработки. Отдельные узлы могут содержать законченные подпрограммы, что позволяет проверять работоспособность каждого узла по отдельности, а также позволяет представить главную программу как конструкцию из готовых модулей.

В качестве узлов модели для простоты мы возьмем консольные приложения Windows, а качестве связей между ними – анонимные каналы. Сам конвейер предполагается работающим на единственном компьютере.

Анонимные каналы обеспечивают межпроцессное взаимодействие на локальном компьютере [1]. Каждый канал характеризуется дескрипторами входа и выхода. Данные, попадающие на вход канала из одного процесса, можно получить в порядке FIFO на выходе другого процесса.

Консольное приложение по умолчанию имеет три стандартных дескриптора – ввода, вывода и ошибок [2]. Для обеспечения связи мы будем заменять стандартные дескрипторы дескрипторами каналов.

Прежде всего, мы построим классический линейный конвейер. Запуск процессов-узлов и установление связей между ними мы организуем в главном процессе. Для переопределения стандартных дескрипторов можно поступить, например, следующим образом: сделать требуемые дескрипторы каналов наследуемыми, установить их вместо стандарт-

ных, а после создать дочерний процесс и закрыть их в главном. Это позволит в дочерних процессах работать с новыми дескрипторами так же, как и со стандартными [3].

РАЗВЕТВЛЕНИЕ КОНВЕЙЕРА

Если программа содержит смежные независимые участки, то можно выполнять их параллельно. Классический конвейер не позволяет ветвить программу из-за того, что у каждого узла ровно один выход. Однако, этой функциональности можно добиться, введя служебные участки конвейера – для *ветвления* и *слияния* ветвей.

Задача участка ветвления – продублировать данные для обработки каждой из ветвей. Можно выделить два основных подхода для его построения.

Первый подход заключается в том, что мы будем использовать множество простых узлов. Как упоминалось выше, каждый узел имеет два стандартных выхода – вывода и ошибок. Тогда для узла мы можем поставить следующую задачу: получить порцию данных и продублировать ее на оба выхода. Каждый такой узел увеличит число ветвей на одну; применив композицию, мы можем получить необходимое число узлов.

Второй подход состоит в том, что мы будем использовать только один узел ветвления. Однако он должен взять на себя ответственность за переключение входов каналов. Чтобы это было возможно, главный процесс должен передать узлу требуемые наследуемые дескрипторы. Например, это можно осуществить, используя аргументы командной строки.

Оба подхода имеют свои преимущества и недостатки. Первый подход будет работать при большом количестве ядер процессора, однако он требует дополнительные системные ресурсы на поддержку каждого из узлов и служебных каналов между ними. Второй подход требует всего лишь одного ядра для участка ветвления, но возникают дополнительные задержки за счет переключения каналов. Следует отметить, что в реальных задачах размерность ветвления обычно небольшая, потому оба подхода работают приблизительно равное время.

Задача участка слияния – упорядочить данные, полученные с различных ветвей, и отправить их в виде кортежа на дальнейшую обработку. Получить данные с нескольких ветвей в одном узле возможно, используя для ветвей общий выходной канал. Однако, этого недостаточно: из-за различного времени работы ветвей данные будут поступать асинхронно. Выходом из этой ситуации может послужить добавление к данным *тега* – служебной информации, позволяющей определить, с какой из ветвей пришли данные. Следует отметить, что введение тегов потре-

бует в участке слияния также наличия узлов-*врапперов* на окончании каждой из ветвей, которые и будут добавлять служебную информацию.

Сам узел слияния может действовать следующим образом. При поступлении очередного сообщения просматривается тег и определяется номер ветви. Полученные данные добавляются в очередь, соответствующую данной ветви. Если каждая из очередей не пуста, то берутся их головные элементы, из них формируется кортеж и отправляется следующему процессу.

ОБРАТНЫЕ СВЯЗИ В КОНВЕЙЕРЕ

Дальнейшим расширением структуры конвейера может служить организация *обратных связей* между узлами. В классической схеме конвейера, если узел *L* отправил свои данные узлу *R*, то узел *R* не может вернуть свои данные узлу *L*. Однако для построенной нами модели легко обеспечить поддержку конвейерных *петель*.

Перед узлом, в котором требуется получение обратных данных, поставим узел слияния. В узле, из которого данные будут поступать, задействуем в качестве входа для канала стандартный поток ошибок. Выход канала направим в узел слияния. Отметим, что, как и прежде, для определения, откуда пришла информация, требуется использовать теги и, соответственно, использовать узлы-врапперы.

Кроме прямой связи узлов возможны варианты организации и косвенной. Один из вариантов – использовать для передачи центральный узел, выполняющий роль маршрутизатора. При приеме сообщения он смотрит на тег и ищет адресата. Другой вариант – передавать сообщения лишь соседним узлам; они же реагируют в зависимости от тега и тоже могут передать сообщение дальше.

Для сложных конвейерных систем может быть эффективным использовать гибридную модель обратных связей.

Вообще говоря, поддержка в конвейере обратных связей позволяет обмениваться узлам не только данными, но и служебной информацией. Это можно использовать для оповещения узлов о состоянии конвейера или для его динамической настройки.

Например, если в одном из узлов произошла критическая ошибка, единственное, что можно сделать – это прекратить его работу. Однако все предыдущие узлы, не зная об ошибке, будут продолжать безрезультативную работу. Отправка сообщения об ошибке другим узлам может решить эту проблему, позволив им корректно завершить свою работу.

Один из основных факторов, снижающих эффективность конвейера – разное время обработки в узлах на линейных участках и на ветвях. В

общем случае задача нормализации времени работы узлов является трудно разрешимой. Но разницу во времени работы можно сгладить, задав различные приоритеты для узлов. Во время работы конвейера с помощью служебных сообщений замеряется время, затрачиваемое на обработку узлами и ветвями. Эта информация далее передается управляющему узлу; он анализирует ее и определяет оптимальные приоритеты для узлов. Результаты такой подстройки могут использоваться и в дальнейшем: зная оптимальные приоритеты, их можно задавать для узлов сразу же при создании. Таким образом, может быть получена оптимальная конфигурация для определенного конвейера.

Сделаем замечание о построенной модели конвейера. Прежде всего, упор в ней сделан на повторное использование узлов-модулей, частично принося в жертву общую эффективность. Если требуется снизить издержки на служебную передачу данных, имеет смысл объединять в одном узле не только обрабатывающие, но и служебные функции.

Построенная модель может использоваться для применения в численных задачах. Имея несколько готовых модулей, можно построить вычислительную схему, удовлетворяющую данной задаче.

Подведем итоги. Предложенная модель конвейера позволяет организовывать программы со сложными структурными элементами, такими как ветвления и петли. Обратные связи дают возможность реагировать на различные события. Наличие служебных узлов обеспечивает отделение кода обработки данных и кода связи между узлами.

Литература

1. Интернет-адрес: http://msdn.microsoft.com/en-us/library/windows/desktop/aa365782(v=vs.85).aspx.
2. Интернет-адрес: http://msdn.microsoft.com/en-us/library/3x292kth.aspx.
3. *Рихтер Дж*. Windows для профессионалов: создание эффективных Win32-приложений с учетом специфики 64-разрядной версии Windows / Дж. Рихтер. СПб.: Питер, 2004. 749 с.

ГАМИЛЬТОНОВОСТЬ ЛОКАЛЬНО СВЯЗНЫХ ГРАФОВ

П. А. Иржавский

1. ВВЕДЕНИЕ

Граф называется *гамильтоновым*, если в нем имеется *гамильтонов цикл*, т. е. простой цикл, содержащий все вершины этого графа. К проблеме гамильтоновости графов, а также к ее взвешенному аналогу – задаче о коммивояжере, проявляется устойчивый интерес в течение мно-