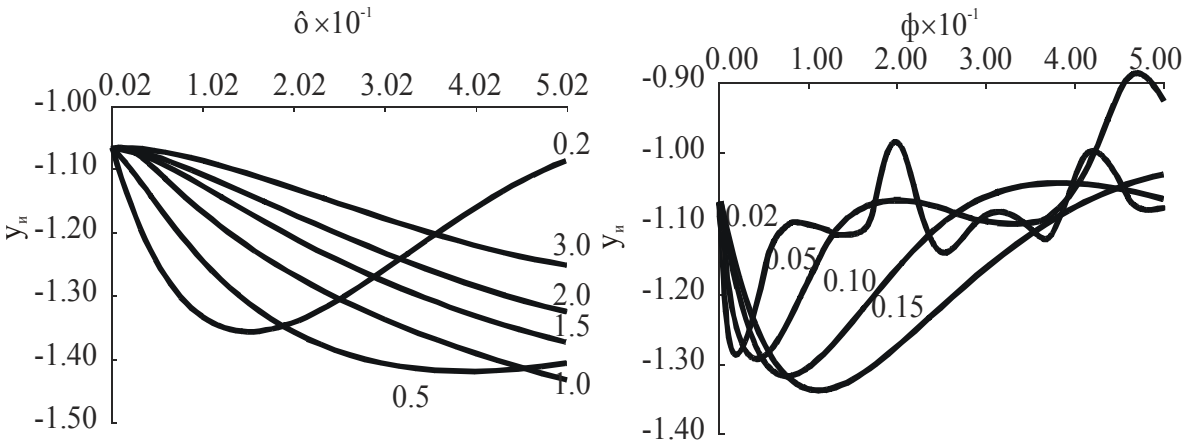


Рис. 3. Зависимость азимутальных напряжений σ_θ от τ при различных значениях γ

В данной работе проведено численное моделирование амплитуды акустического сигнала вблизи трека заряженной частицы. Получены графики описывающие динамику акустических сигналов и термонапряжений. Результаты могут быть использованы при моделировании процессов обработки материалов пучками ионов.

Литература

1. *White R. M. J.* Appl. Phys. 1963. № 7. P.2123.
2. *Perry F. C.* Appl. Phys. Lett. 1970. № 9. P.408.
3. *Калиниченко А. И, Лазурик-Эльцуфин В. Т.* Укр.физ. журн. 1974. № 3. С. 515.

ПРИМЕНЕНИЕ АСИНХРОННЫХ МОДЕЛЕЙ ВЫЧИСЛЕНИЙ В СИСТЕМАХ РАСПРЕДЕЛЕННОЙ ОБРАБОТКИ НА КЛАСТЕРНЫХ АРХИТЕКТУРАХ

А. В. Шостак

Введение

Системы распределенной обработки, основанные на кластерных архитектурах, являются на сегодняшний день наиболее перспективной технологией для решения задач, решение которых требует большого объема вычислений. Кластерные архитектуры позволяют достигать производительности в несколько раз выше, чем в централизованном компьютере. Из-за особенностей своей реализации производительность данных систем может быть легко увеличена.

Для организации вычислений в системах распределенной обработки используются различные модели организации вычислений и различные программные реализации данных моделей.

В данной работе предложена модель программной реализации секционной модели вычислений [1] на основе MPI(Message Passing Interface) [3]. Статья является продолжением работы начатой в [2]. В том случае секционная модель была реализована с использованием DCOM. Но из-за медленной работы данной технологии (в частности, большие задержки возникали при проверке прав при вызове функций через DCOM), от дальнейшей реализации данной программной оболочки пришлось отказаться. MPI лишен данной проблемы и его узкое место – это коммуникационная среда, но при данной технологической реализации сетевого оборудования, она не является превалирующей.

Математическое описание секционной модели параллельных вычислений

Параллельная программа в секционной модели представляется в виде координационных схем (КС). КС – это множество секций $S = \{s_i, i = 1,...,m\}$ с определенными условиями инициализации (УИ), где секция s_i – это двойка $s_i = (\alpha_i, \pi_i), i = \overline{1,m}$, α_i – условие инициализации (логическое выражение, истинность которого определяет готовность секции к выполнению); π_i – тело секции, которое представляет собой набор программ.

Состояние секций из множества S может быть представлено с помощью кортежа текущих состояний отдельных секций: $< (u_t(s_i), v_t(s_i)), i = 1, 2, ..., m >$, где $u(s_i)$ и $v(s_i)$ определяют соответственно общее число инициализаций и завершения секции s_i в момент времени t.

Аксиомы секционной модели были подробно описаны в [1].

Вводится абстрактная машина, называемая монитором координационной схемы (МКС), задача которого проверка УИ и запуск секции на выполнение, если УИ истинно.

Модель программной реализации секционной модели параллельных вычислений на основе MPI

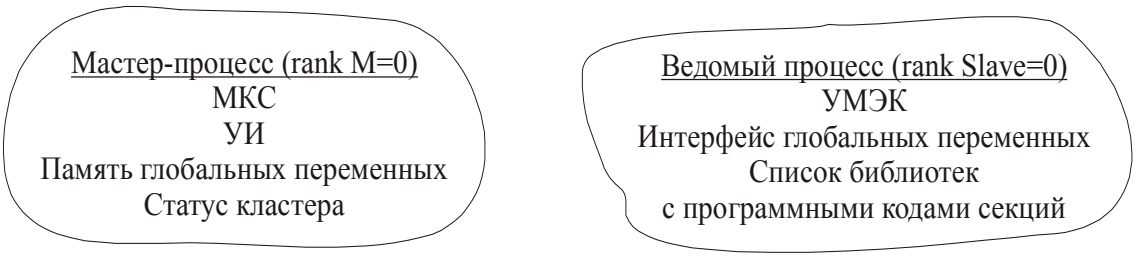
Представим кластер как единый домен процессов $P = \{P_1,...,P_k\}$ для MPI. Выделим среди процессов мастер-процесс M и множество ведомых процессов $\{Slave_i\}$, $i = 2,...,k$ ($P = M \cup \{Slave_i\}$). В терминах MPI положим $rank\ M = 0$, $rank\ Slave_i = i, i = 2,...,k$. Данные ранги будут присвоены процессам автоматически при создании домена процессов MPI.

Каждая секция в данной модели представлена в виде динамически подсоединяемой библиотеки с экспортируемой функцией входа. Эти

библиотеки распространяются между всеми узлами кластера при старте системы.

Кроме МКС введем также понятие управляющего монитора элемента кластера (УМЭК), который фактически и является тем ведомым процессом $Slave_i, i = \overline{2,k}$ и обеспечивает в паре с мастер-процессом M функционирование секционной модели.

Опишем далее функции мастер-процесса M и ведомых процессов $Slave_i, i = \overline{2,k}$. Схематично они представлены на рисунках 1 *а* и 1 *б*.



а б

Рис. 1. Схема процессов кластера
а) мастер-процесс; *б)* ведомый процесс

Статус кластера – это массив состояний узлов кластера (свободен-занят). Данный массив заполняется при старте системы, путем получения сообщений от УМЭК$_i, i = \overline{2,k}$, когда он создается и осуществляет предварительные операции. УИ каждой секции мастер-процесс хранит локально, для обеспечения высокой производительности и минимизации коммуникационных затрат. УИ – это фактически список функций, которые возвращают true/false.

МКС просматривает УИ каждой секции $s_j, j = \overline{1,n}$ и если $\alpha_j = true$, то посылается MPI-сообщение о запуске данной секции на свободный узел кластера, который находится через в массиве статуса кластера. Данное сообщение получает УМЭК$_i$ данного узла, который посылает сообщение о запуске s_j мастер-процессу. Процесс M, получая данное сообщение, делает пометку в статусе кластера, что данный узел занят, и полагает $u_t(s_j) = u_t(s_j) + 1$. После окончания работы s_j, УМЭК$_i$ посылает следующее сообщение о завершении работы. Далее процесс M, получая данное сообщение, обновляет статус кластера и полагает $v_t(s_j) = v_t(s_j) + 1$.

Описанный выше алгоритм обеспечивает работу по запуску и остановке секций в данной модели.

Для полноты модели вводим глобальные переменные. Под глобальными переменными мы будем понимать переменные, которые могут быть прочитаны/записаны любой секцией на любом узле кластера.

В данной программной реализации секционной модели глобальные переменные реализованы как классы стандартных типов. Данные переменные создаются мастер-процессом и доступны по своему имени в области глобальных переменных. Причем мастер-процесс имеет доступ к данным переменным как к локальным переменным.

Продемонстрируем схематично работу с глобальной переменной целого типа.

Положим для удобства размер данной переменной 1 байт, и пусть ее имя будет «*I* ». Для реализации операций чтение/запись введем класс *CInt*, отвечающий за работу с глобальной переменной данного типа. При создании данного класса ему передается в качестве параметра для конструктора класса имя переменной «*I* ». Также в классе реализованы методы *Read* и *Write*, для операций чтения и записи соответсвенно.

Предположим нам надо прочитать данную переменную из области глобальных переменных. Для этого мы вызываем *CInt.Read*. Данный метод путем MPI посылает сообщение мастер-процессу на выдачу данных по переменной «*I* ». Процесс *M* , получив данное сообщение, посылает в ответ информацию с именем переменной и ее данные. УМЭК$_i$ запросившего узла передает эти данные в запросивший ее класс. Для реализации этого механизма используется интерфейс глобальных переменных.

Аналогично выполняется операция записи. Вызывая *CInt.Write*, мы тем самым инициируем посылку сообщения мастер-процессу об обновлении данной переменной «*I* » в области глобальных переменных.

Заключение

Предложенная модель вычислений является основой для последующей технической реализации секционной модели вычислений на кластерных архитектурах. Данная модель вычислений и ее техническая реализация может позволить нам достичь приемлемых скоростей обработки данных, которые не были достигнуты нами в предыдущей работе [2].

Литература

1. *Буза М. К., Зимянин Л. Ф.* Секционная модель параллельных вычислений // Программирование. 1990. № 4. С. 54–62.
2. *Шостак А. В.* Программная поддержка моделей распределенной обработки // Научные труды молодых ученых, аспирантов, соискателей. Н34 В 3ч. Ч. 3. Мн.: БГУ, 2002. С. 148–152.

3. *Шпаковский Г. И., Серикова Н. В.* Программирование для многопроцессорных систем в стандарте MPI: Пособие Мн.: БГУ, 2002. 323 с.

ПРОБЛЕМА ИЗВЛЕЧЕНИЯ ЗНАНИЙ ДЛЯ СОЗДАНИЯ ТЕСТОВ В ОБУЧАЮЩИХ СИСТЕМАХ

Гази Шаках

Системы компьютерного обучения (СКО) применяются в органах государственного и местного управления, в системе высшего и среднего образования и на корпоративном уровне. Наибольшие темпы развития имеет корпоративное обучение, что вызвано резким увеличением зависимости эффективности работы предприятия от темпов и уровня обучения персонала. Важнейшей частью СКО являются модули тестирования, с помощью которых выполняется постоянный мониторинг персонала с целью выявления узких мест или определения реальных результатов переподготовки. До настоящего времени эти средства разрабатывают группы, состоящие из эксперта, инженера знаний и программиста с использованием дорогостоящих программных средств. Это долговременный, трудоемкий, дорогостоящий и психологически сложный процесс (Рис. 1), разработанный в 1990-х годах.

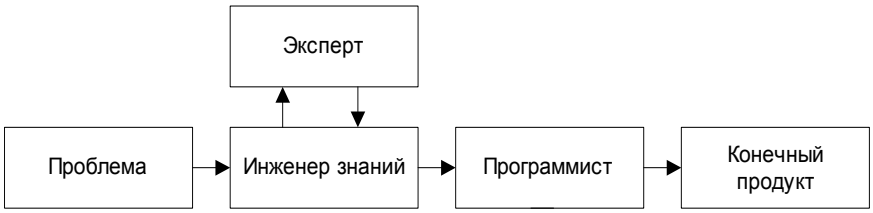


Рис.1. Традиционный процесс разработки тестовых систем

Характерные особенности этого процесса:

- сложность подбора психологически совместимой группы разработчиков;
- большие затраты времени и высокая стоимость разработки;
- необъективность тестов, так как эксперт, как правило, один;
- внести изменения в базу знаний или создать новую могут только разработчики.

В последние годы появилась потребность одновременного обучения и регулярного мониторинга уровня знаний посредством тестовых систем практически всего персонала корпораций часто в количестве десятков тысяч человек. Очевидно, что традиционный способ не соответствует новым условиям, так как за время создания теста тема может просто устареть. Требуется разработать новую технологию, которая обеспечит дос-