НАУЧНОЕ ПРИБОРОСТРОЕНИЕ, 1999 том 9, № 2 с.22-29

ОРИГИНАЛЬНЫЕ СТАТЬИ

УДК 681.3.06

ФОРМАЛЬНАЯ МОДЕЛЬ ДИНАМИЧЕСКИХ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ В ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМАХ ОБРАБОТКИ ЭКСПЕРИМЕНТАЛЬНЫХ ДАННЫХ

© Ю.Е. Шейнин

Санкт-Петербургский государственный университет аэрокосмического приборостроения, E-mail: ysh@mt.spb.su

Поступила в редакцию 15 января 1999 г.

Массивно-параллельные вычислительные системы и распределенные кластерные структуры на серийно-выпускаемых микрокомпьютерах находят широкое применение для обработки экспериментальных данных и управления комплексами научного оборудования. Решаемые ими задачи зачастую не поддаются эффективному статическому распараллеливанию. Предлагается новая формальная модель вычислений — модель Асинхронных Развивающихся Процессов (АРП-модель), дающая возможность представления динамических параллельных вычислений, соответствующих адаптивным, ветвящимся и недетерминированным алгоритмам обработки данных. Описываются основы построения и свойства Общей АРП-модели, задающей правила функционирования динамических неинтерпретированных схем параллельных программ, полностью распределенные механизмы управления параллельными вычислениями.

ВВЕДЕНИЕ

В системах обработки экспериментальных данных и больших научных приборных комплексах находят широкое применение параллельные вычислительные системы и распределенные комплексы [1]. Так, обработка данных физических экспериментов всегда требовала большой вычислительной мощности. В последние годы необходимые характеристики вычислительных средств получают, организуя распределенные высокопараллельные вычислительные комплексы из относительно недорогих серийно выпускаемых компьютеров, рабочих станций и ПЭВМ — так называемые "процессорные фермы для физиков" ("Physics Farms"). Такие вычислительные средства являются как важной составляющей больших приборных комплексов для физических экспериментов, так и инструментарием для их проектирования. Например, в ЦЕРН организовано несколько больших процессорных ферм, в состав которых входит порядка 250 компьютеров, включающих в сумме 500 процессоров [2]. Эксперимент ATLAS по изучению взаимодействий протон-протон, по оценкам специалистов, потребует вычислительного комплекса с 2500–5000 процессорами. Хотя эта категория ВС показывает явные преимущества по технико-экономическим параметрам перед большими малопроцессорными суперкомпьютерами, проблемы встают при переносе на них стандартных библиотек и приложений, особенно — включающих динамическую параллельную обработку. Такие приложения встречаются, например, в задачах проектирования нового поколения ускорителей с высокой плотностью потока частиц (расчет динамической апертуры невзаимодействующих частиц потока [3]), ••• ••••••••••• • ••••••••••••• •••••• •••••• •• ••••••••• •••••-•••••, ••• ••••••••• • ••••••••• ••••••• [4] • ••. •ерегулярные процессы обработки, ветвящиеся и адаптивные алгоритмы не поддаются эффективному статическому распараллеливанию и априорному фиксированному распределению процессов по компонентам параллельной вычислительной структуры. Это порождает сложности в управлении вычислениями, в программировании и применении таких систем.

В значительной степени эти практические проблемы порождены отсутствием адекватной формальной модели, соответствующей динамическим параллельным вычислениям — вычислениям, динамически меняющим состав и взаимосвязи между параллельными процессами в ходе решения задачи. Программные средства типа Meiko Atomic Transaction Library или пакета MPI, реализуя набор примитивов обмена сообщениями, фактически перекладывают задачу распределения и управления вычислениями на прикладного программиста. Широкий спектр известных моделей параллельных вычислений — от CSP-модели Хоора, положенной в основу транспьютерных технологий, до сетей потоков данных (data-flow) Дж. Денниса и многочисленных вариаций на основе Сетей Петри, строят управление вычислениями в рамках парадигмы сети процессов (операторов, акторов, переходов и др.) с неизменной структурой, что остав-

ляет вопросы организации и управления динамическими параллельными вычислениями за рамками решаемых в них проблем.

В настоящей работе представлены основы построения модели динамических параллельных вычислений — Общей модели Асинхронных Развивающихся Процессов (АРП-модель). Построение модели идет в согласовании с качественными характеристиками и свойствами перспективного класса вычислительных средств - массивно-параллельных вычислительных систем и распределенных кластерных структур на серийно-выпускаемых микрокомпьютерах, находящих широкое применение для обработки экспериментальных данных и управления комплексами научного оборудования [1-3]. Эти классы вычислительных систем объединяет распределенная архитектура и организация взаимодействия вычислительных модулей обменом сообщениями. Параллельное децентрализованное управление вычислениями является естественным требованием в этом контексте.

ТРЕБОВАНИЯ К МОДЕЛИ

При построении любой формальной модели вычислений модель описывается в терминах некоторой абстрактной машины, для которой постулируются состав, свойства и правила функционирования ее компонентов. С другой стороны, известен подход установления соответствия между абстрактной машиной и языком — каждой абстрактной машине соответствует некоторый язык, а каждому языку соответствует некоторая абстрактная машина, реализующая универсальный алгоритм языка и определяющая семантику алгоритмического языка (рис. 1). Сопоставляя понятия алгоритмического языка и формальной модели вычислений можно сказать, что модель вычислений определяет правила функционирования схемы программы (неинтерпретированной или частично-интерпретированной), а полностью интерпретированная схема программы и есть программа на рассматриваемом языке [5].

Конструктивный подход к построению формальной модели параллельных вычислений требует как учета качественных характеристик той вычислительной среды, в которой будут функционировать рассматриваемые вычисления, так и учета свойств программ, свойств рассматриваемых классов параллельных вычислений.

Рассматриваем вычисления, характерные для класса универсальных вычислительных систем — вычисления, представляющиеся динамической системой неоднородных процессов [6]. Вычислительная среда для функционирования динамических параллельных процессов — параллельные ВС с распределенной архитектурой и распределенные вычислительные комплексы. Формальная модель для таких параллельных вычислений должна обеспечивать возможности для полностью распределенного управления вычислениями, не должна требовать синхронности выполнения действий в ВС над компонентами параллельной программы. Модель должна обеспечивать порождение большого числа параллельных процессов для эффективного заполнения ВС вычислениями, эффективной загрузки компонентов ВС (в мультипроцессном режиме работы модулей ВС), обеспечивать возможности многовариантного, функционально-эквивалентного выполнения параллельных процессов, образующих вычислительный процесс решения задачи, дающие операционной системе параллельной ВС возможности гибкого управления распределением ресурсов, распределением процессов по модулям ВС, возможность поддерживать высокую эффективную загрузку модулей ВС, а значит — высокую интегральную производительность ВС.

Таким образом, мы должны работать в

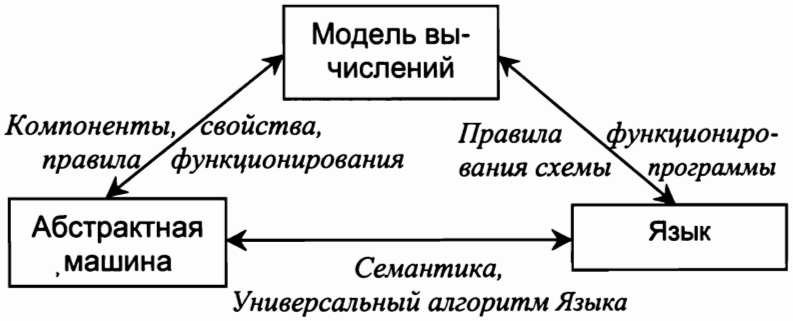

Рис. 1. Машина, модель вычислений, язык

классе асинхронно-параллельных моделей вычислений [7]. Требование порождения большого числа процессов в корреляции с гибкой адаптацией к постоянным изменениям ресурсной ситуации в ВС с возможностями динамической оптимизации хода вычислений, плюс качественные характеристики рассматриваемых классов вычислений определяют необходимость построения формальной модели параллельных вычислений как *динамических* сетей параллельных процессов, меняющих свой состав и взаимосвязи компонентов сети в ходе вычислительного процесса.

АСИНХРОННЫЕ РАЗВИВАЮЩИЕСЯ ПРОЦЕССЫ

Схема Программы в Общей АРП-модели

Предлагается новая модель динамических параллельных вычислений для ВС с распределенной архитектурой – модель Асинхронных Развивающихся Процессов, АРП-модель. Фактически, можно говорить о семействе АРП-моделей, формируемых в русле изложенного подхода к построению формальной модели вычислений в корреляции с абстрактной машиной и ее языком. Общая АРП-модель определяет правила функционирования неинтерпретированной схемы программы. На основе Общей АРП-модели определяются различные Частные АРП-модели. В Частных АРП-моделях вводится интерпретация, определяется семантика некоторых (типов) операторных вершин, также могут накладываться некоторые дополнительные ограничения на топологию представимых в этой модели схем параллельных программ. В настоящей статье рассматривается Общая АРП-модель.

Схема асинхронно-параллельной программы — ориентированный граф, задаваемый кортежем

$$S=\{Vp, Vd, F, W, M\},$$

где $v\in Vp$ — операторная вершина *(Оп)*,
$v\in Vd$ — вершина объекта-данного *(ОД)*,
F — отношение инцидентности графа, $F \subseteq Vp\times Vd$,
W — разметка дуг, определенных отношением F,
M — разметка состояний вершин графа S, представляющего схему программы.

Разметка W отражает типы дуг, связывающих вершины графа, представляющего схему программы. Алфавит $Д$ разметки W составляют метки типов дуг, связывающих вершины графа, $Д=\{\sigma_w, \sigma_r, \sigma_{re}, \sigma_{rew}\}$, где σ_w — дуга доступа к ОД по записи, σ_r — дуга доступа к ОД по чтению, σ_{re} — дуга доступа к ОД по чтению со стиранием, σ_{rew} — дуга доступа к ОД по чтению, стиранию, записи.

Разметка M отражает состояния вершин графа, компонентов схемы программы. Алфавит A разметки M графа S составляют метки состояний вершин схемы программы и он состоит из двух подалфавитов, $A = \{\zeta,\psi\}$:

- меток возможных состояний операторных вершин $v\in Vp$, $\zeta=(\xi n,\xi r,\xi e)$, где ξn — "операторная вершина не готова к запуску", ξr — "готовая к запуску операторная вершина", ξe — "операторная вершина находится в состоянии выполнения, срабатывания";
- меток возможных состояний вершин объектов-данных $v\in Vd$, $\psi=(\varphi e,\varphi f,\varphi b)$, где φe — "вершина объекта-данного находится в состоянии "пусто", φf — "вершина объекта-данного находится в состоянии "полно", φb — "вершина объекта-данного находится в состоянии "блокирована".

Множество всех элементов схемы программы обозначим $Y= Vp\cup Vd$.

Для Операторных вершин определим понятие входных и выходных вершин-ОД.

Для $x\in Vp$,

$*x = \{y|\ (xFy \wedge W(xFy) \in \{\sigma_r, \sigma_{re},\sigma_{rew}\}\}$ — множество входных элементов для x, $x^* = \{y|\ (xFy \wedge W(xFy)\in \{\sigma_w,\sigma_{rew}\}\}$— множество выходных элементов для x.

Неориентированный граф — основание графа S, обозначим $\bar{S}$, некоторую связную компоненту графа S — как Sc, а соответствующую связную компоненту графа $\bar{S}$ — как $\bar{Sc}$.

$Sc=(Vp_c, Vd_c, F_c, W_c, M_c)$, $Vp_c\subseteq Vp$, $Vd_c\subseteq Vd$, $F_c\subseteq F$, $W_c\subseteq W$, $M_c\subseteq M$.

Для удобства рассмотрения некоторых свойств схемы и отдельных аспектов ее функционирования Схеме S сопоставим также граф информационных связей вершин **G**. Множество вершин графа G совпадает с множеством вершин S, а множество дуг формируется на основании обращения направления дуг, помеченных метками σ_r, σ_{re} на противоположное — от вершины ОД к вершине типа Оп.

$$G=\{Vp, Vd, H, W_H, M\},$$

где $H\subseteq Vd\times Vp \cup Vp\times Vd$,

$H=\{xHy \mid (x\in Vd, y\in Vp, yFx \wedge W(yFx)\in \{\sigma_r,\sigma_{re},\sigma_{rew}\}) \vee (x\in Vp, y\in Vd, xFy \wedge W(xFy)= \{(\sigma_{rew},\sigma_w)\}\}$.

$$W_H(xHy)=\begin{cases} W(xFy), \text{если существует } xFy, \\ W(yFx), \text{если существует } yFx. \end{cases}$$

Используя H, вводим понятия и обозначения
°x = {y| yHx} — множество входных вершин для x и
x° = {y| xHy} — множество выходных вершин для x.

Обозначим также c(x) множество вершин, смежных с вершиной x,

$$c(x) = {}^\circ x \cup x^\circ,\ x \in Y.$$

Отметим, что °x, x° и c(x) определены для вершин любых типов, как для операторных вершин, так и для вершин-ОД.

Линейно упорядоченная последовательность вершин графа G, $y_1, y_2, \ldots y_k$ называется путем из y_1 в y_k, если $\forall i,\ 1 \le i < k-1,\ y_i F y_{i+1}$. Обозначим ее $D(y_1, y_k)$.

Для схемы программы в АРП-модели выполняются (постулируются) следующие условия:

A0. $Vp \cap Vd = \varnothing$,

A1. $\forall x \in Vp,\ c(x) \subseteq Vd$.

Как видно из определения F, операторные вершины не могут быть смежными между собой в графе S.

A2. $(F \neq \varnothing) \wedge (\forall x \in Vd, \exists y \in Vp, yFx) \vee ((F=\varnothing) \wedge (Vd=\varnothing))$,
т.е. каждый элемент схемы программы – вершина объект-данное, смежно связан дугой хотя бы с одной операторной вершиной.

К операторным вершинам такого требования не предъявляется — могут быть изолированные операторные вершины. Орграф S не обязательно является связным графом, (в том числе — может состоять из одних несвязанных операторных вершин). Основание $\overline{S}$ орграфа S также не обязательно является связным Как известно [8, Теорема 2.2.1] каждый неориентированный граф распадается единственным образом в прямую сумму своих связных компонент, $\overline{S} = \cup_i \overline{S}_i$, что позволяет большинство вопросов о свойствах схемы программы сводить к их исследованию на связных компонентах графов S и $\overline{S}$.

A2a. В каждой компоненте связности орграфа должна быть хотя бы одна операторная вершина (*следствие* из A2)

$$\forall Sc \subseteq S,\ \exists z \in Yc,\ z \in Vp_c.$$

Функционирование неинтерпретированной Схемы программы

Готовность Операторных вершин (Оп) к срабатыванию

Вершина $v \in Vp$ в схеме программы S_i может сработать, если выполнены условия готовности вершины v к срабатыванию, задаваемые предикатом Φ(v). Вершина, готовая к срабатыванию, разметкой M помечается меткой ξr. Множество вершин $v \in Vp$, для которых Φ(v) = true, образует множество вершин, готовых к срабатыванию, R = {v| Φ(v) = true}.

Предикат Φ(v) определяется на множествах входных и выходных элементов для вершины v:

Для $v \in Vp$,

$$\Phi(v) = \begin{cases} \text{true}, & \text{если } (\forall y \in {}^*v,\ M(y) = \\ & = \varphi f) \wedge (\forall y \in v^*,\ M(y) = \varphi e), \\ \text{false}, & \text{в противном случае.} \end{cases}$$

Трансформации Схемы программы

Функционирование схемы состоит в срабатывании операторных вершин и в последующих изменениях Схемы программы: и в изменениях орграфа, и в изменениях разметки схемы. Назовем эти изменения трансформацией Γ схемы S в S' в результате срабатывания операторной вершины $v \in Vp$.

Трансформация Γ схемы S в результате срабатывания $v \in Vp$,

$\Gamma(S,v)$: $S \to S'$.

Трансформация $\Gamma(S,v)$ над схемой S состоит в изменениях орграфа — множеств операторных вершин, вершин объектов-данных, отношения инцидентности (проявляется как появление/уничтожение дуг, связывающих как вновь порождаемые вершины, так и ранее существовавшие) и в изменениях разметки схемы. Трансформацию Γ определяем как суперпозицию частных трансформаций, каждая из которых производит изменения в некоторой компоненте кортежа S={Vp, Vd, F, W, M}, представляющего схему программы,

$$\Gamma(S,v) = \Im 2(\Im 1(S,v)).$$

Набор частных трансформаций $\Im 1$ включает трансформации изменения множеств вершин Vp и Vd (удаления и порождения вершин) и дуг, определяемых F. В АРП-модели постулируется принцип однократного срабатывания операторных вершин — после срабатывания операторная вершина уничтожается, удаляется из схемы программы.

A3. $\forall x \in Vp,\ \Gamma(S,x)$: $S \to S',\ x \notin S'$.

Соответственно, из F удаляются все дуги, инцидентные этой вершине. В свою очередь, удаление дуг, идущих от удаляемой оператор-

ной вершины к вершинам-данным $z \in {}^\circ v \cup v^\circ$ вызывает уничтожение вершин-данных, для которых это были последние входные дуги, инцидентные данной вершине.

Трансформация Схемы при срабатывании Оп может приводить не только к ликвидации, исключению из Схемы Оп, ОД и дуг, но и к порождению, включению в Схему новых компонентов, нового фрагмента схемы (далее по тексту — фрагмента) [9]. Включаемый фрагмент S^{Fr} является некоторой схемой, формируемой и описываемой в соответствии с общими правилами построения Схемы: $S^{Fr} = \{V^{Fr}p, V^{Fr}d, F^{Fr}, W^{Fr}, M^{Fr}\}$. Характер и конкретный состав фрагмента определяется семантикой оператора, сопоставляемого сработавшей Оп. Однако, на уровне неинтерпретированной схемы программы, функционирование которой описываем Общей АРП-моделью вычислений, по определению не представима семантика никакой из вершин схемы. В Общей АРП-модели мы будем налагать минимальные ограничения на S^{Fr}. В соответствии с постулируемыми свойствами схемы S (свойство A2), орграф S и его основание $\underline{S}$ могут быть несвязными графами. Если S^{Fr} включается в S отдельной компонентой (отдельными компонентами) связности, то на него не накладывается никаких ограничений. Ограничения налагаются на то, какие связи могут быть установлены между S^{Fr}, вновь порождаемым при срабатывании операторной вершины v, и имевшейся схемой S. Общий концептуальный подход состоит в том, что S^{Fr} может быть связан дугами с теми вершинами из S, к которым могла иметь доступ сама сработавшая вершина v, причем с учетом типа доступа — т.е. разметка W' новых дуг, связывающих вершины из S^{Fr} с вершинами S, должна соответствовать разметке типа доступа для дуг из v. Отношение F_v^{link} может задавать пустое или не пустое множество дуг, связывающих вершины S^{Fr} и S, описывает, как срабатывающая вершина v связывает порождаемый ею S^{Fr} с исходной схемой. Как и для фрагмента в целом, F_v^{link} зависит от семантики оператора, сопоставляемого вершине v. На уровне неинтерпретированной схемы программы мы говорим, что F_v^{link} — любое отношение, удовлетворяющее условию $F_v^{link} \subseteq \{x \times y\}$, $x \in Vp^{Fr}$, $y \in c(v)$.

Частные трансформации $\Im 1$ изменения Vp и Vd и дуг:

$$\Im 1(S,v) = \Im 1в(\Im 1б(\Im 1a(S,v))),$$

где

($\Im 1a$): $Vp' = Vp \setminus v \cup Vp^{Fr}$,

($\Im 1b$): $F' = F \setminus \{vFx \mid x \in Vd\} \cup F^{Fr} \cup F_v^{link}$,

($\Im 1c$): $Vd' = Vd \setminus Vers \cup Vd^{Fr}$.

Здесь $Vers = \{z \mid z \in Vd, \forall y \in V'p\ \{yF'z\} = \varnothing\}$.

Частные трансформации $\Im 2$ включают трансформации изменения разметок M и W. Изменение разметки дуг W сопутствует изменению состава дуг в ходе трансформации $\Im 1$:

$$\Im 2(S,v) = \Im 2в(\Im 2б(\Im 2a(S,v))),$$

где

($\Im 2a$): Для $x \in Vp'$, $y \in Vd'$,

$$W'(xF'y) = \begin{cases} W(xFy), \text{если } x \in Vp, y \in Vd, \\ W(xF^{Fr}y), \text{если } x \in Vp^{Fr}, y \in Vd^{Fr}, \\ \sigma' \in \{\underline{inh}(W(vFy))\}, \\ \quad \text{если } x \in Vp^{Fr}, y \in Vd, y \in c(v), \end{cases}$$

где $\underline{inh}(\sigma)$ — правило наследования разметок:

$$\underline{inh}(\sigma) \in \begin{cases} \{\sigma_w\}, \text{если } \sigma = \sigma_w, \\ \{\sigma_r, \sigma_{re}\}, \text{если } \sigma = \sigma_{re}, \\ \{\sigma_r\}, \text{если } \sigma = \sigma_r, \\ \{\sigma_r, \sigma_{re}, \sigma_w, \sigma_{rew}\}, \text{если } \sigma = \sigma_{rew}. \end{cases}$$

($\Im 2б$):

$$M'(x) = \begin{cases} \varphi e, \text{если } (x \in {}^*v) \wedge W(vFx) \in (\sigma_{re}, \sigma_{rew}), \\ M(x), \text{если } (x \in {}^*v) \wedge W(vFx) = \sigma_r, \\ \varphi f, \text{если } x \in v^*, \\ M^{Fr}(x), \text{ если } x \in Vd^{Fr}. \end{cases}$$

($\Im 2в$): $M'(u) = \begin{cases} \xi n, \text{если } (u \in U \wedge \Phi(u) = \underline{false}), \\ \xi r, \text{если } (u \in U \wedge \Phi(u) = \underline{true}), \\ M(u), \text{если } u \notin U, \end{cases}$

где $U = c(c(v)) \cup Vp^{Fr}$.

В результате трансформации изменится состав разметка Оп z, смежных с ОД, для которых изменилась разметка; для них будет перевычислен предикат Ф и изменена разметка готовности M(z)'. Соответственно, изменится множество вершин, готовых к срабатыванию, R(S'). Схема программы функционирует в дискретном времени. Считаем, что трансформация схемы S в S' в результате срабатывания v происходит в один момент дискретного времени (акторная модель вычислений).

Запуск Операторных вершин

В каждый момент времени имеется некоторое множество **R** операторных вершин, готовых

к срабатыванию. Модель вычислений должна определять правила, какие из готовых к срабатыванию операторных вершин, запускаются на выполнение, срабатывают.

Естественно для асинхронных моделей вычислений с децентрализованным управлением принять правило, что может сработать произвольное подмножество операторных вершин, готовых к срабатыванию $R' \subseteq R$ (как, например, в моделях потоков данных). Однако, в отличие от моделей потоков данных, где данные находятся "на дугах" и операторные вершины не могут иметь общих входных или выходных вершин-данных (в моделях потока данных — "дуг") в нашей модели наличие у различных операторных вершин дуг, связывающих их с одними и теми же ОД-вершинами, не исключается. Операторные вершины могут иметь общие ОД по входу или по выходу.

Аналогичные проблемы встают и в других моделях вычислений, допускающих наличие в схеме параллельных вычислений операторов (акторов, переходов) с несовместными условиями готовности. Например, в Сети Петри срабатывание перехода может забрать фишки из входных мест другого перехода, что повлечет изменение условий его срабатывания, [10]. В Сетях Петри наложено условие, что в каждый момент срабатывает только один переход из числа готовых. Второе важное условие — это акторное, мгновенное срабатывание перехода и полное связанное с этим изменение разметки в его входных/выходных местах. За счет этих условий в Сети Петри не возникает неоднозначности результата срабатывания запущенного перехода. Плата за такое разрешение проблемы в Сетях Петри — это фактический перевод ее в класс последовательных моделей вычислений (в один момент срабатывает только один переход), централизованное и синхронное управление вычислениями. Кто-то должен наблюдать за всей сетью и управлять, чтобы в каждый момент сработал только один переход. Сеть Петри асинхронна и децентрализована по определению условий готовности переходов, но централизованна (и синхронна !) по запуску переходов на срабатывание.

В нашей модели также необходимо исключить неопределенность, связанную с одновременным срабатыванием операторных вершин с несовместными условиями готовности. И это надо сделать так, чтобы не привело к фактическому введению глобальной синхронизации и централизации управления, чтобы в правилах срабатывания и функционирования сохранилась та же асинхронность (отсутствие необходимости синхронности, одномоментности каких либо действий — проверок, трансформаций, изменения разметок, состава вершин и т.д., над схемой программы в целом) и децентрализация управления (отсутствие единого координатора действий над всеми компонентами схемы).

Вводим отношение несовместности по условиям готовности вершин $v,w \in Vp$, $v \neq w$,

$$v \neg w.$$

Операторные вершины v и w, $v,w \in Vp$, $v \neq w$, имеют несовместные условия готовности, если срабатывание одной из них, v, приводит к тому, что вызванная ею трансформация схемы определяет невыполнение условий готовности w, $\Phi(w) \neq \underline{true}$. При этом до срабатывания v, $\Phi(v) = \underline{true}$ (сработать может только вершина, для которой выполнено условие готовности, $\Phi(v) = \underline{true}$), а $\Phi(w) = \underline{true}$ или $\Phi(w) \neq \underline{true}$; после срабатывания v, $\Phi(w) = \underline{false}$.

Введенное отношение не является рефлексивным, $v \neg v$ не входит в график этого отношения. Отношение несовместности не является транзитивным, из $v \neg w$ и $w \neg u$ не следует $v \neg u$. Поэтому мы будем говорить, в основном, о попарной несовместности вершин по условиям готовности. Отношение несовместности не является и симметричным: из $v \neg w$ не обязательно следует $w \neg v$.

Вводим понятие кластера несовместности операторных вершин — множества вершин, имеющих попарно-несовместные условия готовности (одна Оп может входить в несколько кластеров, т.к. по разным ОД может иметь несовместные условия готовности с разными другими Оп), и $\underline{sample}(X)$ — операцию выборки одного элемента из X. Кластер несовместности $Кл \subseteq Vp$

$$Кл = \{v \mid \forall v,x,\ v \neq x\ M(v) = \xi r \underset{\Gamma(S,v)}{\longrightarrow} M'(x) = \xi n\}.$$

Тогда, Параллельная Трансформация $\wp(S)$ схемы S определяется как :

$$\wp(S) = \bigcup_{\underline{sample}(Кл^k)} \Gamma(S,v).$$

Трасса вычислений и самоизменяемость схемы программы

Динамическая схема программы определяется как схема программы, трансформируемая в ходе своего функционирования в результате срабатываний составляющих Схему операторных вершин. В отличии от подавляющего большинства моделей вычислений, как последовательных, так и параллельных, опре-

деляемых в терминах статической, не меняемой в ходе вычислительного процесса схемы программы, в АРП-модели описание функционирования программы не может быть дано в терминах изменения разметок какой-то одной схемы программы. Сама схема программы меняется, модифицируется на каждом шаге вычислений.

Функционирование схемы программы представляется последовательностью размеченных схем программ S, трассой функционирования динамической схемы программы. Определяется начальная схема, S_0, и трасса T представляется последовательностью наблюдаемых Схем программ $S_0 \rightarrow S_1 \rightarrow S_2 \ldots \rightarrow S_k \rightarrow \ldots$, $T = \{S_0, S_1, \ldots S_k \ldots\}$.

Самоизменяемость схемы программы используется в АРП-модели и как базовый механизм управления ходом вычислений в зависимости от значений обрабатываемых данных — необходимой черты любой модели вычислений для обеспечения ее алгоритмической полноты. В АРП-модели зависимость развития вычислений от значений обрабатываемых данных реализуется различием в трансформациях, генерацией при срабатывании некоторых операторных вершин разных фрагментов схемы программы в зависимости от значений данных. Здесь зависимость хода вычислений от значений обрабатываемых данных реализуется не в виде некоторых функциональных зависимостей от них значений предикатов готовности того или иного вида (охраняемых команд Дейкстры, спусковых функций Котова-Нариньяни, структурированных семафоров Корнеева и др.), не в виде переключения направления потока данных (влияние на предикат готовности поступлением/не поступлением меток-данных), то есть не в виде влияния значений данных на предикаты готовности множества операторных вершин схемы, а в форме изменения самого множества операторных вершин, проверяемых на готовность. Это снимает проблему, что делать с "ненужной" при этих значениях обрабатываемых данных частью схемы программы, как синхронизировать слияние потоков данных, прошедших через разные наборы альтернативных ветвей и т.д. В АРП-модели альтернативная невостребованная "ветвь" вычислений просто не порождается, не появляется в схеме программы.

Исследование свойств вычислений, порождаемых динамической схемой программы в АРП-модели, производится на различных статических формах представления динамических вычислений. Вводится понятие профилей вычисления — различных представлений состоявшегося (потенциального или реального) вычисления над некоторой начальной Схемой в виде единого интегративного объекта (в отличие от трассы), на котором исследуются свойства вычисления, порождаемого параллельной программой. Примером профиля является след вычисления — граф, включающий совокупность Оп, сработавших в ходе вычисления, и совокупность ОД, на которые указывали входные/выходные указатели этих операторов.

ЗАКЛЮЧЕНИЕ

АРП-модель относится к классу асинхронных моделей вычислений с децентрализованным управлением. Предложенная модель позволяет представлять и исследовать динамические параллельные вычисления, ветвящиеся, адаптивные и недетерминированные параллельные алгоритмы. Она хорошо согласуется с качественными свойствами параллельных ВС с распределенной архитектурой и методами организации распределенных параллельных вычислений, позволяет в рамках единой модели отразить основные подходы к построению моделей параллельных вычислений.

Варьируя определением ограничений на топологию сетей процессов и интерпретацию отдельных операторов в частично-интерпретированных схемах программы, мы можем как строить различные варианты частных АРП-моделей, для оптимизации тех или иных свойств и характеристик специфицировать разные известные модели параллельных вычислений в качестве ее частных случаев. Фактически, АРП-модель может выступать в качестве метамодели при исследовании и сопоставлении широкого класса разных моделей параллельных вычислений,

Она служит также теоретическим базисом для построения новых параллельных вычислительных систем, для организации управления процессами в новых операционных системах [11] и языках параллельного программирования [12].

ЛИТЕРАТУРА

1. *McIntosh, E., Panzer-Steindel B.* Parallel Processing in CERN. //Proceedings of the HEPiX96. October 1996. Caspur Rome. P.10-20.
2. *Robertson L.* Computing Trends & Strategy at CERN. CERN, IT Division. 1998. 4 p.
3. *Giovannozzi M., McIntosh E.* Development of parallel codes for the study of nonlinear beam dynamics //International Journal of Modern Physics. 1997. V. 8. P.155–170.
4. ATLAS DAQ, EF, LVL2 and DCS Technical Progress Report. CERN. 30 June 1998. 86 p.
5. *Котов В.Е., Сабельфельд В.К.* Теория схем

программ. 1991. М. Наука. 248 с.
6. *Игнатьев М.Б., Шейнин Ю.Е.*//Распределенные вычисления и системы на СБИС: Межвуз. сб. под ред. М.Б. Игнатьева, Ю.Е. Шейнина. 1988. Л. ЛИАП. С.5-13.
7. *Котов В.Е., Черкасова Л.А.* //Кибернетика и вычислительная техника. Под ред. В.А. Мельникова. 1986. Вып. 2. М. Наука. С.75-94.
8. *Оре О.* Теория графов. 1980. М. Наука. 215 с.
9. *Игнатьев М.Б., Мясников В.А., Шейнин Ю.Е.* //Кибернетика. 1984. N 3. С.48-53
10. *Котов В.Е.* Сети Петри. 1987. М. Наука. 160 с.
11. *Aladova T., Mikhailichenko D., Nedzelskaya N., Sheynin Yu.* //Real Time Magazine, 1997. N 3. P.23-28
12. *Tatkov D.E., Sheynin Yu.* //Int. Simp. On Problems of Modular Information Computer Systems and Networks. Abstracts. 1997. Moscow. P.32-33.

FORMAL MODEL OF DYNAMIC PARALLEL COMPUTATIONS IN PARALLEL COMPUTERS FOR EXPERIMENTAL DATA PROCESSING

Yu. E. Sheinin

St. Petersburg State University of Aerospace Instrumentation

Mass-parallel computers and distributed computer clusters are widely used in DAQ and off-line processing of experimental data as well as in complex instruments control and monitoring. Some of their tasks are dynamic in nature. A formal model of parallel computations is considered in correlation with a virtual machine features and its language, defining rules of operation for the uninterpreted (or partially interpreted) program schemes. A model of parallel computations - Asynchronous Growing Processes (AGP-model) is proposed for parallel and distributed computing. It generates dynamic computations as self-modified dynamic networks of processes and controls them in an asynchronous fully distributed manner.