УДК 004.415.2.032.24

# ФОРМАЛИЗМ ДЛЯ ОПИСАНИЯ ПРОГРАММНЫХ СИСТЕМ И ВЫЧИСЛИТЕЛЬНЫХ ПРОЦЕССОВ ЦИКЛИЧЕСКОЙ ПАРАЛЛЕЛЬНОЙ ОБРАБОТКИ ДАННЫХ РЕАЛЬНОГО ВРЕМЕНИ[1]

***И. В. Стручков,***
*аспирант*
***В. М. Ицыксон,***
*канд. техн. наук*
*Санкт-Петербургский государственный политехнический университет*

*Рассматриваются вопросы формализованного описания программных систем и вычислительных процессов циклической параллельной обработки данных реального времени для специализированных информационно-вычислительных комплексов. Для указанного класса систем предлагается новый способ формализованного описания – граф генераторов и преобразователей данных, который может использоваться для выделения функциональных модулей-задач, имитационного моделирования, статического и динамического распределения задач по процессорам в многопроцессорной системе.*

*The article addresses a formal description of cyclic real-time parallel computational processes and corresponding software systems for specialized computing complexes. A new formal approach for describing the specified class of systems is introduced: the data generator-transformer graph. This graph can be utilized for the decomposition of task modules, simulation, static and dynamic task mapping in a multiprocessor system.*

Задача параллельной обработки большого объема потоковой информации является типичной для многоканальных систем анализа сигналов о состоянии пространственно-распределенного объекта. Такого рода системы используются в различных прикладных областях, таких как гидроакустика, радиолокация и подобных им, где возникают следующие задачи: обнаружение и идентификация объектов, сопровождение обнаруженных объектов, прогнозирование местоположения и скорости движения объектов, классификация объектов, спектральный и корреляционный анализ сигналов. Для систем подобного класса характерны следующие особенности:

– данные от аналогового источника поступают в реальном времени с высокой частотой дискретизации (до сотен мегагерц);

– количество параллельных каналов ввода может составлять от нескольких сотен до нескольких тысяч;

– для данных, полученных от различных каналов ввода, предусмотрена одинаковая или однотипная математическая обработка;

– вычислительные процедуры обработки данных представляют собой последовательный многоэтапный процесс, допускающий конвейерную организацию вычислений;

– вычислительные процедуры обработки повторяются многократно для новых порций данных и, таким образом, являются циклическими процессами.

При этом разработчики математического и программного обеспечения должны учитывать следующие условия:

– ограниченность вычислительных ресурсов системы, часто связанную со специальными требованиями, предъявляемыми к используемым в составе аппаратуры компонентам;

– предопределенность архитектуры аппаратного обеспечения системы обработки, вызванную высокой стоимостью и большой длительностью процесса разработки аппаратуры;

– жесткие требования к временным характеристикам.

[1] Статья выполнена в рамках работы по гранту для поддержки научно-исследовательской работы аспирантов вузов Федерального агентства по образованию 2004 года № А04-3. 16-482.

Совместный учет перечисленных особенностей и условий при разработке программного обеспечения требует специальной методики. При этом возможность применения стандартных технологий и средств разработки существенно ограничена.

Основным способом достижения требуемой производительности является параллельная организация вычислений, поэтому первостепенная задача – это эффективное распараллеливание вычислений. Она имеет два пути решения:

статическое распараллеливание – осуществляется на этапе проектирования либо сборки программной системы на основе анализа моделей вычислительных процессов и потоков данных;

динамическое распараллеливание – осуществляется в процессе функционирования программной системы с учетом фактических данных о загруженности процессоров и коммуникационных каналов.

При любом пути решения существенным оказывается учет особенностей системы и задачи обработки данных. Такой учет возможен при наличии формализованной модели вычислительного процесса, адекватно отражающей суть происходящих явлений.

Формализованное описание вычислительного процесса является важнейшей частью всего процесса разработки, и от качества выполнения данного этапа зависит качество системы в целом.

На различных этапах жизненного цикла программного обеспечения требуются различные типы формализованного описания:

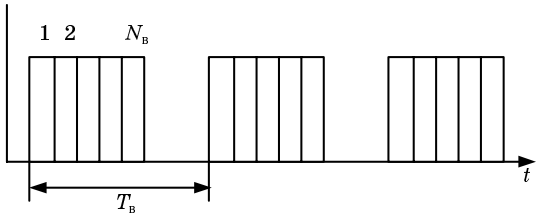
на этапе проектирования создаются структурные и поведенческие описания для выделения функциональных модулей-задач и определения информационных связей между ними;

на этапе разработки и компиляции формируются графы задач, потоков данных и соединений процессоров для статического распределения задач по имеющимся вычислительным ресурсам [1];

на этапе исполнения аналогичные графы используются для динамической балансировки нагрузки [2], автоматической диагностики и контроля работоспособности системы.

При исследовании многопроцессорного комплекса для обработки гидроакустических сигналов авторами была разработана теоретико-графовая модель параллельных вычислительных процессов – граф генераторов и преобразователей данных (ГГПД). Данная модель ориентирована на описание программ циклической параллельной обработки данных реального времени. Отличительной особенностью разработанной модели является возможность ее использования на всех трех перечисленных этапах жизненного цикла программного обеспечения.

При разработке модели ГГПД учитывалось, что важнейшей особенностью специализированных систем потоковой обработки информации является цикличность. Входные данные от внешних источников поступают, как правило, регулярными порциями фиксированного размера. Асинхронными и непериодическими событиями в системе являются изменения режима работы или параметров системы. Для функционирования системы в реальном времени необходимо, чтобы обработка очередной порции данных полностью завершалась в течение периода поступления входных данных.




■ ***Рис. 1.*** *Детерминированный всплесковой поток данных*

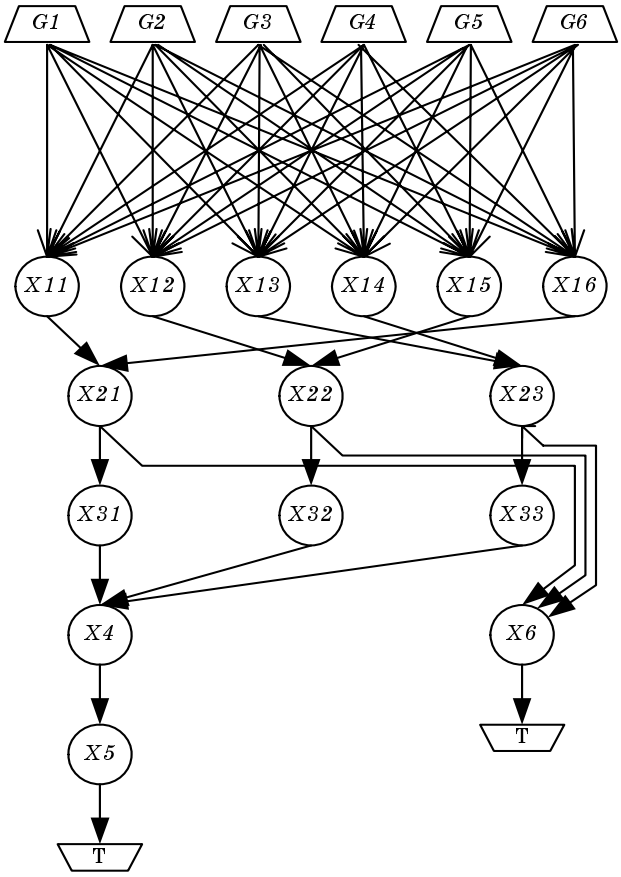
Анализ нескольких реальных систем обработки информации [3] показал, что потоки данных, создаваемые отдельными параллельными задачами, имеют всплесковый характер. Всплески состоят из одного или нескольких фрагментов данных, следующих непосредственно друг за другом (рис. 1). Периодичность всплесков задается детерминированной временной константой. Размер фрагмента данных фиксирован и задается коммуникационной подсистемой. Такие потоки далее будем называть детерминированными всплесковыми потоками.

Цикличность процессов в системе и всплесковый детерминированный характер потоков данных являются основой модели ГГПД. Граф ГПД содержит вершины трех типов: генераторы данных, преобразователи данных и терминаторы. Генератор автономно формирует всплески согласно заданной временной константе. Один генератор может создавать потоки одновременно для нескольких получателей. Генератор характеризуют следующие параметры: период всплесков ($T_в$); количество фрагментов во всплеске ($N_в$); получатели данных.

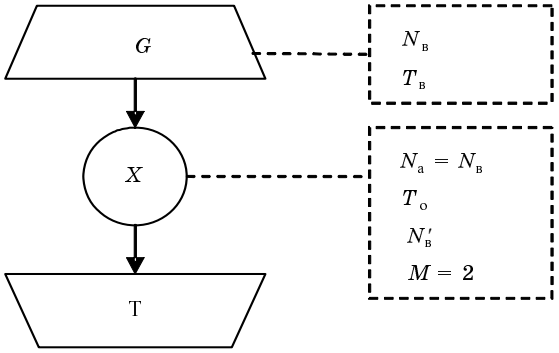
Преобразователь данных, в отличие от генератора, активизируется не автономно, а после получения заданного количества входных данных. Преобразователь характеризуют следующие параметры: количество входных фрагментов для активизации ($N_а$); время обработки ($T_о$); получатели данных; для каждого получателя – количество фрагментов во всплеске ($N_в$); для каждого получателя – кратность, т. е. количество активизаций, после которого генерируется всплеск ($M$).

Терминатор является просто заглушкой, не имеет параметров и на графе обозначает конец вычислений.

Таким образом, каждая задача в рассматриваемых системах выполняется циклически с обновленными входными данными. Период цикла зада-




■ ***Рис. 2.*** *ГГПД для программы параллельной обработки гидроакустических сигналов*




■ ***Рис. 3.*** *Пример простого ГГПД*

чи-генератора равен $T_в$. Период цикла задачи-преобразователя зависит от скорости поступления входных данных. Полному циклу соответствует $M$ поступлений по $N_а$ фрагментов данных (активизаций), причем при каждой активизации затрачивается время на обработку $T_о$.

Граф генераторов и преобразователей данных позволяет описывать параллельные программы из класса циклических программ параллельной обработки данных с детерминированными всплесковыми потоками данных. В частности, определенная комбинация генераторов и преобразователей на заданной топологии позволила адекватно моделировать реально исследованную задачу параллельной обработки гидроакустических сигналов [3]. Для исследованной параллельной программы ГГПД представлен на рис. 2 (задачи-генераторы обозначены $G_i$, задачи-преобразователи – $X_{jk}$, терминаторы – Т). Формально, с каждой вершиной графа связаны соответствующие параметры (рис. 3). На рис. 2 данные параметры не показаны в целях наглядности.

Поскольку ГГПД отражает логическую структуру программного обеспечения, данное описание может создаваться уже на этапе проектирования. На этапе разработки и компиляции необходимо решить задачу оптимального статического распараллеливания. Возможны два пути применения ГГПД на этом этапе:

– преобразование ГГПД в иную модель, для которой существует методика статического распараллеливания;

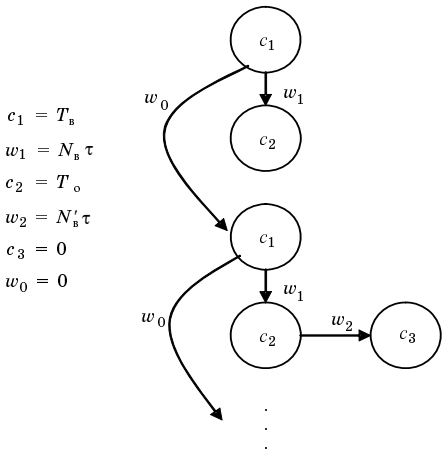
– создание методики статического распараллеливания, использующей ГГПД непосредственно в качестве исходной информации.

Покажем возможность эквивалентного перехода от ГГПД к другим моделям описания параллельных процессов. За основу возьмем простой граф (см. рис. 3).

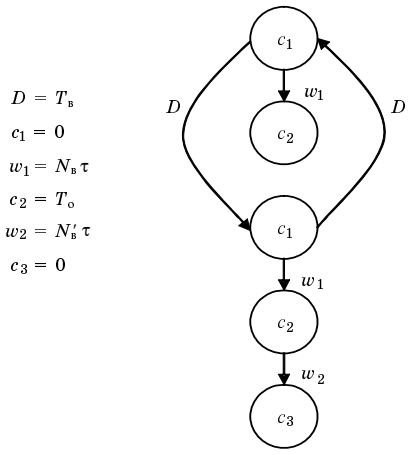
Наиболее распространенным способом формализованного описания параллельных процессов являются графы задач. В настоящее время предложено множество подходов к построению графов задач [4, 5], из которых в данной работе рассматриваются направленные ациклические графы (DAG), итеративные графы задач (ITG) и временные коммуникационные графы (TCG).

Рассмотрим переход от графа на рис. 3 к направленному ациклическому графу (DAG) на рис. 4.

Направленные ациклические графы являются наиболее простым и часто используемым в алгоритмах распределения нагрузки способом описания параллельных программ. В таком графе вершины представляют задачи, а ребра – передачу данных между задачами. При описании параллельных программ обычно используют взвешенные направленные ациклические графы. Веса, придаваемые вершинам, отражают вычислительную сложность данной задачи, а веса, придаваемые ребрам, – затраты на передачу данных между двумя данными задачами. Основным недостатком ациклических графов является невозможность адекватно описать итеративные вычисления. Вследствие этого с помощью ациклических графов обычно описывают только параллельные программы с крупным делением на задачи. В этом случае




■ ***Рис. 4.*** *Эквивалентный направленный ациклический граф (τ – время передачи одного фрагмента данных)*




■ ***Рис. 5.*** *Эквивалентный итеративный граф задач*

все итеративные вычисления считаются упрятанными внутри одной задачи (и, следовательно, являются последовательными). Другим решением является развертывание циклов в виде вложенных ациклических подграфов. Однако это допустимо только в том случае, когда число итераций известно на этапе компиляции, что случается достаточно редко.

На рис. 4 периоду генератора данных соответствует вершина графа $c_1$, времени обработки данных преобразователем соответствует вершина $c_2$. Отметим, что для описания кратности преобразователя данных $M$ в DAG требуется дублирование пары вершин $c_1$, $c_2$. В нашем случае $M = 2$, при больших значениях $M$ потребуется соответствующее число повторений вершин в DAG. Фиктивная передача данных с $w_0 = 0$ необходима для обеспечения связности графа. Ациклическая природа графа DAG не позволяет адекватно описать бесконечный процесс, поэтому граф неограничен.

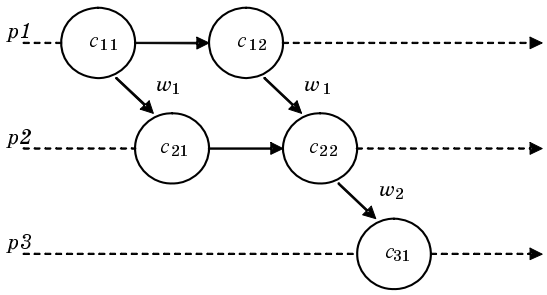
Преимуществом итеративного графа задач (ITG) является возможность компактного описания циклов, поэтому эквивалентный ITG является ограниченным. В итеративных графах задач ребру графа помимо затрат на передачу данных может присваиваться также дополнительное значение – задержка. При этом обязательно должно соблюдаться условие: в любом цикле должна находиться, по крайней мере, одна задержка. Ребра без задержек моделируют зависимости внутри одной итерации, связи с задержками – зависимости между итерациями. Тем не менее, описание кратности преобразователя данных также требует дублирования вершин. Период генератора в данном случае удобнее описать не весом вершины $c_1$, а задержкой $D$, $c_1$ при этом принимается равным нулю. В результате получаем граф, представленный на рис. 5.

Временной коммуникационный граф (TCG) для рассматриваемого случая является неограниченным, как и DAG. Некоторым преимуществом данного описания (рис. 6) является явное выделение параллельных процессов (задач): $p1$ соответствует задаче-генератору, $p2$ – задаче-преобразователю, $p3$ – задаче-заглушке.

Из рассмотренных примеров очевидно, что возможно построение формальных схем перехода от описания ГТПД к иным моделям графового представления параллельных вычислительных процессов. Большинство существующих алгоритмов статического и динамического распределения нагрузки в многопроцессорных системах [1, 2] базируются на рассмотренных стандартных графовых моделях описания вычислительных процессов в качестве исходной информации. Наличие формальных процедур преобразования описания ГТПД в одну из таких моделей позволит автоматически адаптировать имеющееся описание к стандартным средствам.

Имеющиеся подходы были разработаны для описания широкого класса параллельных программ и, как показано выше, не вполне эффективны для описания циклических многоэтапных процессов, подразумевающих однотипную математическую обработку. Предложенная модель ГТПД не только компактнее и нагляднее описывает процессы циклической параллельной обработки, но и может быть непосредственно использована в процессе разработки и оптимизации параллельных программ.

Имея описание вычислительного процесса в виде ГТПД, разработчик системы получает возмож-




$c_{11} = c_{12} = T_в$
$w_1 = N_в\tau$
$c_{21} = c_{22} = T_о$
$w_2 = N'_в\tau$
$c_{31} = 0$

■ ***Рис. 6.*** *Эквивалентный временной коммуникационный граф*

ность осуществить имитационное моделирование ее функционирования до написания какого-либо программного кода. Такое моделирование на этапе проектирования позволит избежать множества ошибок при разработке. В ходе исследования эффективности коммуникационных подсистем для многопроцессорного комплекса [3] были разработаны имитационные модели аппаратной архитектуры, коммуникационных подсистем и прикладных задач. Для имитационных экспериментов использовалась среда моделирования OMNet++. Имитационные модели параллельных задач базируются на введенных абстракциях генераторов и преобразователей данных, что позволяет использовать описание ГГПД в качестве исходных данных без предварительного преобразования.

Описание ГГПД также может являться входной информацией алгоритма распределения задач по процессорам. Распределение задач по процессорам сводится к решению следующей многокритериальной и, в общем случае, нелинейной задачи оптимизации:

$$\min_{\forall M: S \xrightarrow{M} H} CF(M),$$

где $S$ – граф задач (в нашем случае, ГГПД); $H$ – направленный граф соединений процессоров (вершинам соответствуют процессоры, дугам – межпроцессорные соединения); $M$ принимает значение всех возможных отображений графа задач на граф соединений процессоров.

Для решения данной задачи применяются итеративные алгоритмы оптимизации. Общий вид целевой (минимизируемой) функции для итеративного алгоритма оптимизации можно представить в виде суммы трех компонент [1]:

$$CF(M, t) = F_{верш}(M, t) + F_{дуг}(M, t) + \\ + F_{марш}(M, t, RA),$$

где $M$ – отображение задач на процессоры; $t$ – шаг алгоритма оптимизации; $RA$ – функция маршрутизации сообщений и данных.

Покажем, как сформировать целевую функцию алгоритма оптимизации распределения задач на основе описания ГГПД.

Целевая функция вершин ($F_{верш}$) соответствует затратам на вычисления в процессорах. Для каждой задачи-преобразователя определяется вычислительный вес:

$$c_i = \frac{T_{oi}\sum_{j \in I}\lambda_j}{N_{ai}},\ i \in X; \\ \lambda_j = \frac{N_{вj}}{T_{cj}}; \qquad (*) \\ T_{cj} = \begin{cases} \dfrac{M_j T_{oTask(j)}}{c_{Task(j)}},\ Task(j) \in X; \\ T_{вj},\ Task(j) \in G, \end{cases}$$

где $T_{oi}$ – время обработки $i$-й задачи-преобразователя; $\lambda_j$ – интенсивность $j$-го входящего потока; $I$ – множество входящих потоков задачи; $N_{ai}$ – количество фрагментов для активизации $i$-й задачи-преобразователя; $X$ – множество задач-преобразователей; $N_{вj}$ – количество фрагментов во всплеске для $j$-го входящего потока; $T_{cj}$ – период всплесков $j$-го входящего потока (как от генератора, так и от преобразователя); $M_j$ – кратность для $j$-го входящего потока; $Task(j)$ – индекс задачи, генерирующей входящий поток $j$; $T_{вj}$ – период всплесков для $j$-го входящего потока (только от генератора); $G$ – множество задач-генераторов.

Значение целевой функции вершин рассчитывается, как суммарный дисбаланс нагрузок по всем процессорам:

$$F_{верш}(M, t) = \alpha(t)\sum_{i=1}^{N} | c_i - c_{ср} |,$$

где $\alpha(t)$ – нормировочный множитель, зависящий от шага алгоритма; $N$ – число процессоров; $c_i$ – суммарный вычислительный вес всех задач, распределенных на данный процессор; $c_{ср}$ – средний суммарный вычислительный вес по всем процессорам.

Целевая функция дуг ($F_{дуг}$) соответствует затратам на передачу данных:

$$F_{дуг} = \beta(t) \sum_{i,\ j \in T} \lambda_{ij} d_{ij} w,$$

где $\beta(t)$ – нормировочный множитель, зависящий от шага алгоритма; $T$ – множество всех задач; $\lambda_{ij}$ – интенсивность потока данных от задачи $i$ к задаче $j$ из формулы (*); $d_{ij}$ – расстояние от задачи $i$ до задачи $j$ в узлах, согласно таблице маршрутиза-

ции; $w$ – затраты на передачу единицы (фрагмента) данных между соседними узлами.

Затраты на маршрутизацию ($F_{\text{марш}}$) учитывают загруженность коммуникационных каналов (линков). Загруженность $l$-го линка определяется по формуле

$$AD_l = \sum_{i,\ j \in T} \lambda_{ij} w RA(i,\ j,\ l),$$

где $RA(i, j, t)$ – функция маршрутизации ($RA = 1$, если данные от задачи $i$ до задачи $j$ проходят через линк $l$).

Целевая функция маршрутизации $F_{\text{марш}}$ определяется, как суммарный дисбаланс загруженностей линков:

$$F_{\text{марш}}(M,\ t,\ RA) = \gamma(t) \sum_{l=1}^{L} | AD_l - AD_{\text{ср}} |,$$

где $\gamma(t)$ – нормировочный множитель, зависящий от шага алгоритма; $L$ – число линков; $AD_l$ – загруженность $l$-го линка; $AD_{\text{ср}}$ – среднее значение загруженности по всем линкам.

## Выводы

В данной статье были рассмотрены способы формализованного описания параллельных программ для вычислительных систем с распределенной памятью и представлена новая модель графового описания параллельных программ – граф генераторов и преобразователей данных. ГГПД является расширением графа потоков данных и применим для описания параллельных программных систем и вычислительных процессов циклической обработки данных реального времени. Примером систем данного класса является программное обеспечение вычислительного комплекса многоканальной обработки гидроакустических сигналов.

В вычислительных процессах указанного класса потоки данных имеют периодический всплесковый характер. Параметры вершин в ГГПД основаны на данном представлении потоков данных. Анализ применения ГГПД для описания реальной параллельной программы и сравнение с другими моделями описания позволяют сделать следующие выводы.

1. Описание ГГПД является естественным и наглядным визуальным формализмом для разработчиков системы, так как каждой вершине графа соответствует вполне реальная программная сущность – задача.

2. Описание ГГПД является наиболее компактным для указанного класса задач, по сравнению с эквивалентными описаниями DAG, ITG и TCG, так как скрывает цикличность и кратность обработки данных в семантике вершин.

3. Можно предложить формальную схему перехода от ГГПД к любой из рассмотренных моделей, что позволит разработать автоматизированный программный инструмент для подобных преобразований.

4. Описание ГГПД может быть непосредственно использовано как входная информация для имитационного моделирования системы или алгоритма распределения задач.

Таким образом, модель ГГПД является новой и более эффективной формой описания параллельных программ и вычислительных процессов в классе циклических задач обработки данных реального времени. Модель ГГПД программы, благодаря своей наглядности, может быть построена разработчиком (системным архитектором) на этапе проектирования. В процессе разработки описание ГГПД может быть переведено в эквивалентное представление другого типа, что может оказаться необходимым условием для автоматического (автоматизированного) распределения задач по процессорам вычислительной системы.

Направлением дальнейшего развития предложенного подхода является исследование возможности применения модели ГГПД для статического и динамического распределения задач. В результате исследования должна быть предложена эффективная методика такого распределения и разработан прототип системы управления параллельными задачами на основе данной методики.

## Литература


1. **Hluchэ L., Dobrovodskэ M., Dobruckэ M.** Static Mapping Methods for Processor Networks. http://citeseer.ist.psu.edu/472693.html
2. **Hu Y., Blake R.** An optimal dynamic load balancing algorithm. Technical Report DL-P95 -011. Daresbury Laboratory. Warrington, UK, 1995. http://citeseer.ist.psu.edu/hu95optimal.html
3. **Стручков И. В., Ицыксон В. М., Мелехин В. Ф.** Исследование эффективности коммуникационных систем для многопроцессорного комплекса с распределенной памятью // XXXII Неделя науки СПбГПУ: Материалы межвуз. науч. конф. / СПбГПУ. СПб., 2004. С. 87–89.
4. **Lo Virginia M.** Temporal Communication Graphs: Lamport's Process-Time Graphs Augmented for the Purpose of Mapping and Scheduling // Journal of Parallel and Distributed Computing. 1992. N 16(4). December. ftp://ftp.cs.uoregon.edu/pub/lo/tcg.ps.Z
5. **Sinnen O., Sousa L.** A Platform Independent Parallelising Tool Based on Graph Theoretic Models // Lecture Notes in Computer Science. 2001. Vol. 1981. http://www.ece.auckland.ac.nz/~sinnen/articles/Sinnen2001pip.ps