© 2012 г. В.П. КУТЕПОВ, д-р техн. наук,
Н.А. ПАНКОВ, м.н.с
(Московский энергетический институт, кафедра Прикладной математики, Москва)

УПРАВЛЕНИЕ ПРОЦЕССАМИ В КОМПЬЮТЕРНЫХ СИСТЕМАХ

В статье обсуждаются архитектурные особенности и орианзационная схема управления процессами в компьютерных системах а также рассматриваются методы оптимизации управления их ресурсами.

PROCESS CONTROL IN COMPUTER SYSTEMS / V.P. Kutepov, N.A. Pankov (Moscow Power Engineering Institute, Krasnokazarmennaya 14, Moscow 111250, Russia, E-mail: Kutepov@appmat.ru). Architectural factors and scheme of process management system`s organization and optimization of resource management methods for computer systems are discussed in the paper.

1. Введение

Проблема управления процессами даже в однопроцессорных компьютерах, как известно, является нетривиальной задачей, и потребовался не один десяток лет, чтобы найти приемлемые с практической точки зрения решения, воплощенные в современных операционных системах (ОС). Сложности возникли уже с первой попытки найти эффективные методы планирования для случая пакетного режима, введенного в практику с целью сбалансированного использования различных ресурсов компьютера: процессоров, периферийных устройств и многоуровневой памяти. Главная проблема в формировании оптимальных пакетов программ состояла в отсутствии достоверной априорной информации о поведении программ. Случайный характер обращений к различным ресурсам при выполнении программы оказался настолько сложно прогнозируемым заранее, что пришлось полагаться на весьма далекие от реальности предсказания программиста о том, как распределены интенсивности обращений к различным ресурсам, чтобы на основе этого попытаться сформировать оптимальные пакеты.

Очевидно, радикальным решением проблемы является изначальное рассмотрение процессов выполнения программ как случайных процессов и разработка на этой основе методов стохастического управления ими. Это естественно требует изучения особенностей стохастических процессов, связанных с выполнением программ, методов и средств измерения в динамике их параметров и прогнозирования их изменения. Этот подход мы используем в наших работах [1-4], нацеленных на создание эффективных методов и программных средств для управления процессами в больших компьютерных системах. К ним мы относим многомашинные и многопроцессорные системы, часто называемые кластерами, распределенные системы, объединяющие сотни и тысячи компьютерных подсистем (серверов, кластеров и т.п.). Первые из них часто называются сильно связанными, а вторые – слабосвязанными системами.

Мы будем далее называть последние открытыми компьютерными системами или для сокращения компьютерными системами (КС), понимая под этим объединенные различными коммуникациями компьютерные ресурсы в том числе программные,

доступные для коллективного использования. Англоизычное название GRID, Clouds и т.п. мало что меняет в данном определении.

С формальной точки зрения функционирование КС представляет собой систему массового обслуживания с множеством различного рода взаимосвязанных ресурсов (компьютеров, памятью, периферией, коммуникациями и др.), выполняющих случайные потоки различных процессов (заданий) на их входах. В свое время мы исследовали средствами теории массового обслуживания проблему оптимального управления множеством программ, выполняемых на компьютерах с виртуальной страничной памятью [1].

Этот же подход мы пытаемся развить применительно к анализу и построению эффективных систем управления процессами в КС, в частности при выполнении ими параллельных программ [2,6]. При этом мы исходим из самых общих и реальных положений, состоящих в том, что заранее неизвестны параметры, описывающие случайные процессы, выполняемые различными ресурсами КС, однако предполагаем, что они близки к стационарным [4]. Это дает основания для применения адаптивных методов управления загруженностью компьютерных систем с целью оптимизации их работы. Кроме того, мы используем ряд эвристик, суть которых состоит в устранении «узких мест», возникающих в процессе функционирования КС, а также учете особенностей процессов, порождаемых при выполнении программ [6-8].

Создание эффективных средств управления процессами в КС предполагает решение трех взаимосвязанных задач, которые далее рассматриваются в работе.Первая из них – это разработка методов планирования процессов, учитывающих их особенности и важность. Вторая задача – это создание методов, обеспечивающих оптимизацию назначения ресурсов для выполняемых процессов. Наконец, третья задача – это определение архитектурных решений, связанных со структурной организацией самого управления, возможностью его реконфигурирования в связи с отказами и масштабированием КС и программной реализацией.

Конечно, напрашивается вопрос, а нельзя ли для создания систем управления процессами в КС воспользоваться тем, что уже создано для управления параллельными процессами на кластерах? К сожалению, сегодняшнее управление параллельными процессами в том виде, к которому мы стремимся, отсутствует.

К примеру, в MPI (MPI-1) программист сам должен заранее определить множество выполняемых параллельных процессов в программе, после чего их поровну распределить между узлами кластера для выполнения. Когда мы переходим к программам, которые могут порождать рекурсивно параллельные процессы и процессы с разными поведенческими особенностями, задача существенно усложняется, и пока нет удовлетворительного её решения.

Статьи [4-5] дают некоторое общее представление о наших подходах к созданию систем управления процессами в компьютерных системах.

В разделе 2 обсуждаются архитектурные особенности и организационная схема управления процессами в КС. Раздел 3 посвящен анализу функций управления в КС, а в разделе 4 рассматриваются методы оптимизации управления ресурсами КС.

2. Архитектура КС

Архитектуры сильно связанных КС (кластеров и многопроцессорных систем с общей памятью) и слабосвязанных КС, то есть распределенных территориально компьютерных ресурсов, заметно отличаются.

Коммуникации первых изначально занимают центральное место при их создании, особенно если принять во внимание их сегодняшний масштаб, достигающий нескольких сотен и миллионов компьютерных компонентов. Регулярная коммуникационная структура

связей в таких системах содержит три уровня иерархии. На самом нижнем уровне – уровне блоков, насчитывающих несколько десятков компонентов (компьютеров или процессоров) коммуникационые связи приближены к полносвязным структурам. По крайней мере, между любой парой компонентов в них существует несколько путей передачи данных (см., например, структуру связи Омега между компьютерами блока в системе SP2 [9], родоначальницы современных кластерных решений фирмы IBM). Объединенные в группы блоки физически помещаются в контейнеры, а последние, в свою очередь, группируются в виде аппаратурных стоек. При этом по-прежнему связи между блоками в контейнере более плотные, чем связи между стойками и обеспечивают более высокую пропускную способность при передаче данных.

Однако, сегодня коммуникационная среда в этих системах является общей как для передачи данных между вычислительными компонентами, так и для выполнения управляющих функций. Это сегодня кажется оправданным в силу примитивности самого управления. Но с увеличением масштаба КС и, как следствие, усложнением управления, важный аспект приобретает его организационная структура и собственная реактивность (время, затрачиваемое на выполнение управляющих функций).

Два крайних способа организации управления процессами в КС – централизованный и децентрализованный – страдают существенными недостатками. Первый становится «узким горлом» при увеличении масштаба системы и, начиная с некоторого количества узлов в КС, просто не может эффективно выполнять управляющие функции. Децентрализованный вариант управления в какой-то мере снимает ограничение на увеличение масштаба КС, но при том условии, что коммуникационная среда сможет успешно справляться с возрастающей интенсивностью «переговоров» между многочисленными управляющими компонентами. Как следствие, вариант иерархической организации управления КС, сотни лет используемый в управлении большими системами в самом широком смысле толкования этого понятия, имеет все основания для применения в КС. Это тем более оправдано, если учесть сказанное выше об иерархической организации самой КС, в которой, как, например, у кластеров, иерархическая «лестница» идет от блоков к группам блоков и т.д. Та же организационная логика наблюдается и в распределенных КС, добавляя в отличие от кластеров новые ступеньки, соотнесенные с локальными сетями, затем глобальными компьютерными сетями, то есть, территориально распределенными компьютерными ресурсами. Хороший пример такой организации – платформы облачных вычислений Azure [10], Amazon и др. Иерархическая организация управления позволяет наиболее затратные по времени функции измерения и прогнозирования загруженности компонентов КС выполнять на нижнем уровне, имеющем быстрые коммуникации, оставляя за следующими уровнями с меньшей средней пропускной способностью коммуникаций принятие решений о назначении ресурсов и их перераспределении с целью достижения оптимального использования, то есть, их средней загруженности.

С позиции реализации управления, очевидно, надо опираться на уже достаточно опробированные методы и средства управления, существующие в ОС компьютеров, создавая на их основе операционные средства управления следующих уровней.

Последующие шаги в этом иерархическом централизованно-децентрализованном управлении должны строиться на основе делегирования стратегических функций управления от подсистем нижних уровней следующему уровню иерархии и т.д.

Если мы хотим иметь возможность масштабирования и реконфигурирования управления как целостной системы, придется строго определить функции управления, стандартизировать их с позиции программной реализации, обеспечив возможность их репликации и настройки соответственно конкретным задачам управления.

3. Функции управления КС

Функции управления КС можно разделить на три достаточно самостоятельные части:

- управление заданиями ($Y_з$),
- управление процессами ($Y_{пр}$),
- административное управление ($Y_{ад}$).

Общая организационная структура управления любого уровня приведена на рис. 1.

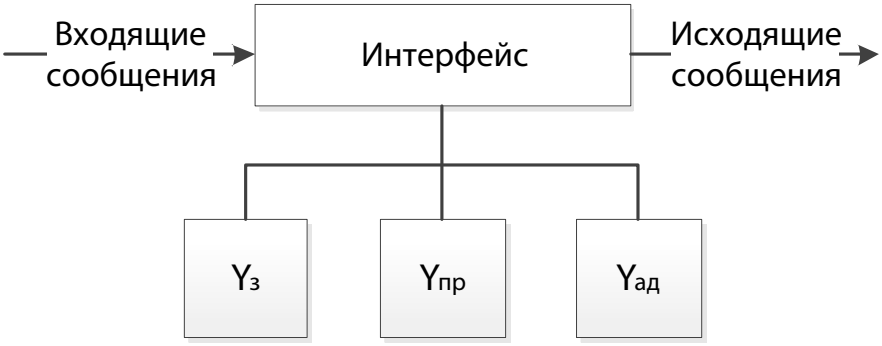

Рис. 1. Структурная схема управления КС

3.1. Функции управления заданиями

В общем случае пользователь или администратор может направлять свое задание интерфейсу управления любого уровня иерархического организованного управления КС. Блок управления заданиями, приняв конкретное задание, должен принять решение, где оно должно выполняться:

- в ресурсной среде подчиненной управлению, получившему задание, при условии наличия подходящих ресурсов,
- в ресурсной среде, контролируемой управлением следующего уровня, если подчиненные ресурсы не могут обеспечить адекватное (согласно требованиям пользователя) выполнение задания.

Постоянный диалог между управлениями различных уровней, осуществляемый через интерфейсный блок, является необходимым условием поиска ресурсной среды, могущей наилучшим образом обеспечить требуемые в описании задания условия его выполнения.

Блок управления заданиями выполняет следующие функции:

- прием и упорядочение заданий,
- анализ требований к ресурсам, указанных пользователем, директивных сроков или времени выполнения заданий, др.; определение на этой основе приоритета и времени инициализации выполнения задания,
- поиск в некотором смысле оптимальной ресурсной среды для выполнения задания на основе диалога с подчиненными управлениями или с управлением более высокого уровня,
- инициализацию и контроль выполнения задания, если это управление нижнего уровня,
- возвращение результата выполнения задания пользователю или посылка сообщений администратору о ситуациях, которые препятствуют выполнению задания (ошибки в программе, превышение директивных сроков её выполнения и др.).

3.2. Функции управления процессами

Под процессом мы понимаем любую самостоятельную последовательность действий в выполняемой в общем случае параллельной и распределенной программе, будь это вызов

специальных программных объектов или удаленных данных, а для параллельных программ это также независимо выполняемые программные компоненты.

Если программа идентифицируется как объект пользователя, то у процесса более сложное идентификационное описание, которое включает в общем случае его принадлежность конкретной программе, его место в истории выполнения программы, связь с его удаленными (вычисленными и хранимыми в памяти удаленных ресурсов) данными, снимок процессов, для которых выполняемый процесс порождает данные и др.

Еще одна важная особенность параллельных программ – это возможность порождения упреждающих процессов, обычно связанных с условными конструкциями [6-7]. В условном операторе *if P(x) then f1(x) else f2(x)* предикат *P(x)* и программы функций *f1* и *f2* можно выполнять одновременно, однако, если результат *P(x)* всегда необходим, то использование результата *f1* или *f2* станет известным только после вычисления *P(x)*. Использование упреждающих вычислений существенно расширяет возможность динамического распараллеливания программ, однако платой за это является более сложная логика обнаружения упреждающих процессов и их прерывание после того, как станет известно, что они не нужны.

Управление процессами в общем случае выполняет следующие функции:

- идентификацию и упорядочение порождаемых процессов;
- сбор и хранение статистических данных о загруженности и состоянии подчиненных подсистем КС;
- планирование процессов, учитывающее их особенности (упреждающие или неупреждающие, др.), предопределенные характером выполняемой программы;
- определение в некотором смысле оптимальной ресурсной среды перед инициализацией выполнения процесса;
- собственно выполнение процесса с реакцией на возникающие состояния порождения процессов, взаимодействие с другими процессами и др.; т.е. корректная реализация программы на ограниченных и распределенных ресурсах КС.

В разделе 4 мы более детально обсудим оптимизационные аспекты реализации данных функций.

3.3. Функции административного управления

По-видимому, административные функции должны выполнять администраторы, связываемые с управлением каждого уровня и имеющие ту же логику подчинения друг другу, что и логика подчинения между управлениями согласно их иерархической организации.

Это вовсе не означает, что один и тот же администратор не может выполнять функции администрирования, привязанного к нескольким управлениям.

Администратор должен быть способен:

- находить решения в любых аномальных ситуациях (отказы, стихийные бедствия и т.п.) в подчиненных ему подсистемах КС;
- реагировать на конфигурационные изменения в КС, инсталлируя при этом необходимые программные продукты, в том числе управляющие;
- реализовывать и контролировать ценовую политику оплаты за использование ресурсов пользователями;
- выполнять другие привычные функции администрирования.

4. Управление с оптимизационных позиций

С экономической точки зрения оптимальное управление должно примирить две конфликтующие цели: хозяина ресурсов и пользователя. Для первого это более высокая производительность и должнаяоплата за использование ресурсов, для второго – как можно меньшая цена, которая будет установлена за решение его задачи в соответствии с заявленными требованиями. При честной игре этих двух сторон естественно полагать, что управление будет стремиться удовлетворить в первую очередь пользователя, при этом не обижая хозяина ресурсов, и что всякого рода блефование в контроле состояния и выделения ресурсов для выполняемых программ пользователя исключено.

При поиске подходящей модели, которая позволила бы хотя бы ближе подойти к пониманию факторов и закономерностей, лежащих в основе решения данной проблемы, напрашивается обращение к теории систем массового обслуживания. С этой точки зрения КС представляет собой открытую систему массового обслуживания со случайными потоками заданий пользователей, обслуживаемых зафиксированными подсистемами КС. Ясно, что на нижнем уровне в качестве простейшего обслуживающего компонента выступает компьютер, сервер или параллельная система (кластер и т.п.). Мы всегда знаем предельную (максимальную) интенсивность обслуживания заявок для этих компьютерных ресурсов, которая определяется обычно умножением быстродействия процессора, измеряемого количеством операций, выполняемых в единицу времени, на количество процессоров в подсистеме.

К сожалению, реальная интенсивность выполнения процессов, как показывают эксперименты, не стационарна и общие характеристики её поведения как случайной величины неизвестны [4].

Большие сложности модельного представления функционирования КС возникают из-за естественной возможности динамического перемещения процессов между подсистемами с целью достижения оптимального использования их ресурсов. Если к этому добавить реальную ситуацию реконфигурирования КС, как из-за отказов определенных ресурсов, так и по другим причинам, то становится ясным, что огромная компьютерная система, даже при тех ограничениях на правила её функционирования, о которых мы говорили в начале этого раздела, вряд ли будет полностью «математизирована» с позиции оптимального управления ею. Аналогия с управлением большими производственными, социальными системами, регулированием рыночной экономики и с тем состоянием далеко не только не оптимального, но и неудовлетворительного управления процессами в этих образованиях оправдывает сказанное выше.

Поэтому решения придется искать подобно этим системам: устранять узкие места в работе КС, создавать систему эвристических правил для эффективного планирования процессов и распределения ресурсов, научиться достаточно точно измерять, существенные для управления параметры функционирования КС и прогнозировать их изменения.

Ниже мы рассмотрим, каким образом это можно делать на практике уже сегодня.

При этом по общеметодологическим причинам, да и по сути, в управлении заданиями и процессами мы выделяем в качестве главных две задачи: планирование заданий и процессов и назначение ресурсов для их выполнения.

4.1. Планирование заданий и назначение ресурсов для их выполнения

Рассмотрим сначала, какую информацию о своем задании пользователь может указать, которая может быть использована для реализации упения.

1) Характер задания: отладочные функции или выполнение программы, язык программирования, последовательная или параллельная программа требуемый распараллеливающий компилятор, др.
2) Особенности программы: предполагаемая сложность, наличие упреждающих и рекурсивно порождаемых процессов, нагрузка на каналы (большая, средняя или незначительная).
3) Требования к ресурсам: пользователь в общем случае может указать, используя справочную информацию об КС, какие конкретно ресурсы (подсистемы КС) должны быть использованы для выполнения его программы. При этом эти указания могут быть детализированы с точностью до задания количества процессоров или узлов (если это параллельная программа, которая должна выполняться на кластере), требуемых объемов ОП и др.
4) Требования к времени выполнения программы: директивные сроки – интервал времени, в течении которого должна быть выполнена программа, и собственно допустимое время выполнения самой программы (особо важный фактор для параллельных программ).

От того, насколько точно будут заданы пользователем эти данные, зависит, очевидно, и эффективность управления, и удовлетворение требований пользователя. По крайней мере, директивные сроки и время выполнения заданий, а мы имеем в виду сложные программы, требующие больших ресурсов, являются главными параметрами для планирования заданий.

Перейдем собственно к проблеме планирования заданий и назначения ресурсов. Пусть задания, поступающие в КС, разделены на N групп по наиболее критическим параметрам для управления: директивным срокам, затребованным ресурсам, сложности и др. и пусть $\lambda_i(t)$ и $\mu_i(t), i = 1,2, \dots, N$ – интенсивности потока заданий и их обслуживания в момент времени t в каждой группе. Отношение $c_i'(t) = \lambda_i(t)/\mu_i(t)$ определяет степень загруженности ресурсов КС, используемых для выполнения заданий группы i. Очевидно, если $c_i'(t) \geq 1$, мы можем говорить, что выделенные ресурсы для заданий группы iне справляются с их эффективным обслуживанием или, другими словами, среднее время обслуживания заданий начинает заметно увеличиваться.

В настоящее время любой ресурс (компьютер, сервер, кластер, др.) характеризуют предельной производительностью, определенной обычно по быстродействию процессора или для многопроцессорных (многокомпьютерных) систем путем умножения количества процессоров на их быстродействие.

Пусть $\mu_i^{(\text{пр})}, i = 1,2, \dots, N$ – предельная производительность ресурсов, выполняющих задания i-ой группы.Отношение или коэффициент использования ресурсов $c_i''(t) = \mu_i(t)/\mu_i^{(\text{пр})}$ характеризует, пусть приближенно, насколько близка реальная производительность к предельной и насколько эффективно управление распоряжается ресурсами.

Очевидно, отношение $c_i''(t)$ всегда меньше или равно единице и его значение позволяет косвенно судить о степени влияния памяти, коммуникаций и реактивности самого управления на интенсивность выполнения заданий i-ой группы.

Напомним, что как только в системе (компьютере, кластере) заметно увеличивается интенсивность обменов страницами между оперативной и дисковой памятью (возникает так называемыйswapping), производительность резко падает [11].

Аналогичную картину можно наблюдать при выполнении сложной параллельной программы с частыми взаимодействиями между её фрагментами, когда среднее время передачи данных по каналу между узлами резко возрастает из-за образования к нему очереди.

Исходя из сказанного, управление будет «идеально» выполнять свои функции, если $c_i'(t) < 1$ для каждого i, среднее время выполнения наступающих в КС заданий минимально, а $c_i''(t)$ максимально. Последнее говорит, что негативное влияние, которое оказывает на время выполнения заданий память и каналы сведено к минимуму.

Во многих реальных сложных системах, например пассажирский и другой транспорт, производственные системы и др., проблема решается просто путем введения заранее определенных ограничений на количество обслуживаемых объектов (пассажиров, производимых изделий) и время обслуживания. В КС управлению придется полагаться на измерение случайных параметров $c_i'(t)$, прогнозирование их изменений, чтобы на основе этого принимать решения. В частности, при $c_i'(t) > 1$, т.е. когда КС оказывается неспособной удовлетворительно обслуживать потоки входящих заданий, естественным выходом из этого положения является откладывание части заданий на более поздние сроки их выполнения. А при честной игре пользователь – КС это потребует от управления способности вести убеждающий пользователей диалог, почему их задания не могут быть выполнены в указанные ими директивные сроки. Этот диалог неизбежен и при выполнении заданий, если обнаруживается, что заявленные пользователем требования на время выполнения его задания не могут быть выполнены.

Сформулируем ряд эвристических правил, вытекающих из опыта управления большими системами, которые, по всей вероятности, в той или иной форме будут использованы для планирования заданий в КС.

1) Не допускать «перегрузки» ресурсов КС, постоянно измеряя значение $c_i'(t)$ и не допуская путем откладывания выполнения части заданий, чтобы оно было больше единицы. Прогноз изменения $c_i'(t)$ помогает улучшить принимаемые решения.
2) Отдавать предпочтение в выполнении в первую очередь «коротким» заданиям с небольшими директивными сроками. Положительный результат – увеличение среднего числа выполняемых процессов в единицу времени, и , как следствие, уменьшение информационной базы, используемой для управления.
3) Стараться назначать для заданий ресурсы, которые при равных условиях требуют меньше коммуникационного времени. При необходимости перемещения порождаемых при выполнении задания процессов с целью выполнения их на соответствующих ресурсах придерживаться этого же принципа. Это так называемый принцип локализации работ в ближайшем ресурсном окружении.
4) Планирование заданий должно базироваться на директивных сроках их выполнения с учетом правил, перечисленных в пунктах 1-4. Любое обнаруживаемое отклонение от них должно быть предметом диалога управления с пользователем с целью возможного увеличения директивных сроков, изменения требований пользователя к ресурсам и др.

Для параллельных программ при отклонении реального времени их выполнения в сторону увеличения по сравнению с указанным пользователем нужен дополнительный диалог о необходимости «усовершенствования» программы, использования большей или меньшей степени её распараллеливания, изменения ресурсной среды (более производительные кластеры и т.п.).

Мы оставляем здесь открытым вопрос об установлении величины «честной» оплаты пользователем за оказанные КС услуги, хотя понятно, что должно быть оправданная обеими сторонами процедура взаимного учета использования ресурсов и удовлетворения требований пользователя.

4.2. Планирование процессов и распределение ресурсов для их выполнения

Мы уже отмечали двойственность понятий задание и процесс и их адекватность, когда речь идет об их выполнении. Однако, процесс мы рассматриваем как самостоятельную

последовательность действий, которая порождается при его выполнении задания и которая требует независимого от других процессов ресурсного обеспечения.

Проблема планирования процессов и распределения ресурсов детально рассмотрена в наших работах [2-4] применительно к реализации языка граф-схемного потокового параллельного программирования на кластерных системах. При создании системы управления для этого случая используются методы адаптивного управления, специально разработанные алгоритмы измерения и прогнозирования загруженности компонентов компьютерной системы, а также учитываются особенности порождаемых процессов, которые мы обсуждали ранее. В настоящее время эта система проходит экспериментальную проверку ее эффективности на кластерных системах.

5. Заключение

В статье мы не имели возможности подробно остановиться на аспектах оптимизации самой программы, учитывая ее выполнение на конкретной компьютерной системе. При этом важными задачами, влияющими на эффективность выполнения программы, являются выбор зернистости или степени распараллеливания программы, использование или неиспользование упреждающих процессов в ней и другое. Решение этих задач, а также создание средств оптимального управления процессами в КС могут обеспечить эффективное использование ресурсов больших компьютерных систем.

СПИСОК ЛИТЕРАТУРЫ

1. *Горицкий Ю.А., Кутепов В.П., Старобогатова Н.Н.* О выборе оптимального числа программ для ВС со страничной стратегией распределения памяти // Техническая кибернетика, 1975, №1.
2. *Кутепов В.П.* Интеллектуальное управление процессами и загруженностью в вычислительных системах // Изв. РАН, Теория и системы управления, 2007, №5.
3. *Kutepov V.P.* Scheduling parallel processes and load balancing in large scale computing systems // Proc. Inter. Symp. on Distributed Computing and Applications for business, engineering and science. China, 2007, N1.
4. *Бражникова Ю.С., Горицкий Ю.А., Кутепов В.П., Панков Н.А., Смоляков Д.С.* Исследование поведения процессов с целью создания эффективных методов управления компьютерными системами // Изв. РАН, Теория и системы управления (в печати).
5. *Кутепов В.П.* Об интеллектуальных компьютерах и больших компьютерных системах нового поколения // Изв. РАН, Теория и системы управления, 2000, №5.
6. *Кутепов В.П., Маланин В.Н., Панков Н.А.* Граф-схемное потоковое параллельное программирование: язык, процессная модель, реализация на компьютерных системах // Изв. РАН, Теория и системы управления, 2012, №1.
7. *Кутепов В.П., Фальк В.Н.* Формы, языки представления, критерии и параметры сложности параллелизма // Программные продукты и системы, 2010, №3.
8. *Кутепов В.П.* О параллелизме с разных сторон // 5-я Международная конференция «Параллельные вычисления и проблемы управления», РАН. М: Институт проблем управления, 2010.
9. *Шмидт В.* Система IBM SP2 // Журн. Открытые системы, 1995, №6 (14).
10. *Мартынов Д., Федоров А.*WindowsAzure–облачная платформа Microsoft // http://download.microsoft.com/documents/rus/msdn/Windows_Azure_Web.pdf
11. *Лян Лю* Исследование эффективности реализации параллельных программ на кластере МЭИ // Вестник МЭИ, 2007, №5.