Эвристики распределения задач для брокера ресурсов Grid

А.И. Аветисян, С.С. Гайсарян, Д.А. Грушин, Н.Н. Кузюрин, А.В. Шокуров

Аннотация. Последние работы в области Grid позволяют приложениям использовать вычислительные ресурсы, принадлежащие различным организациям, распределенным по различным странам и континентам. Одним из видов ресурсов Grid являются однородные многопроцессорные системы (кластеры), которые могут состоять из сотен или даже тысяч процессоров.

В работе представлена архитектура и основные принципы иерархического децентрализованного распределения параллельных задач в таких Grid окружениях. Рассматривается двухуровневая иерархия: на первом уровне несколько независимых брокеров распределяют задачи на кластеры в соответствии с некоторым критерием; на втором уровне каждый кластер распределяет задачи, поступившие от брокера, используя эвристики, основанные на алгоритмах упаковки в полосу. Эффективность предлагаемой схемы распределения задач проверяется на моделях.

1. Введение

Глобальные вычислительные сети Grid [1] были предложены в качестве новой парадигмы для решения крупномасштабных вычислительных задач в науке, технике и бизнесе [2]. Они дают возможность одновременного использования миллионов вычислительных ресурсов [3], принадлежащих различным организациям и расположенных в различных административных областях. Системы Grid объединяют разнородные вычислительные ресурсы (персональные компьютеры, рабочие станции, кластеры, суперкомпьютеры), используя разные стратегии доступа к ним, выполняя различные приложения (научные, инженерные и коммерческие), предъявляющие к системе различные требования. Ресурсы принадлежат различным организациям, имеющим свои правила управления ресурсами, их использования и определения их стоимости для различных пользователей в различное время. Доступность и загруженность ресурсов также может динамически изменяться во времени.

В окружении Grid владельцы и потребители ресурсов имеют различные цели, используют различные стратегии и экономические схемы регулирования спроса и предложения. Таким образом, актуальной проблемой является разработка систем управления ресурсами Grid, нацеленных на оптимизацию отношений между владельцами ресурсов и пользователями в соответствии с выбранными ими стратегиями.

Многие системы управления ресурсами Grid (такие как Legion [4], Condor [5], AppLeS PST [6], NetSolve [7], PUNCH [8], XtremWeb [9] и т.д.) используют *простые схемы* распределения, когда компонент, отвечающий за распределение, решает, какие задачи должны быть выполнены на каком ресурсе, используя функции стоимости, задаваемые системными параметрами. Целью таких систем распределения является увеличение пропускной способности системы, ее загруженности и уменьшения времени выполнения задач, а не увеличение рентабельности ресурсов и приложений. Они не рассматривают цену использования каждого ресурса, а это означает, что значимость выполнения всех приложений в любое время одинакова, что в реальности далеко не так – значимость должна возрастать с приближением срока выполнения прикладной задачи.

В [10] предложено использование экономического подхода к планированию и распределению ресурсов, когда решения о распределении ресурсов производятся динамически и зависят от текущих требований пользователей (такой подход реализован в системе GRACE [11]). Это – рыночная модель распределения ресурсов, когда цена каждого ресурса определяется потребностями в нем пользователей и его доступностью. Таким образом, в системе Grid пользователь конкурирует с другими пользователями и владелец ресурса с другими владельцами ресурсов. Экономический подход позволяет успешно управлять децентрализованными и гетерогенными ресурсами так, как это происходит в реальной экономике. Экономические системы управления ресурсами Grid динамически определяют наилучшие ресурсы, учитывая их цену и производительность, и распределяют задачи на этих ресурсах так, чтобы удовлетворить потребности пользователей.

Как и другие системы, определяемые целями, ресурсами и действиями, вычисления на Grid могут быть рассмотрены в экономических терминах. Быстрое развитие сетевых технологий изменило архитектуру высокопроизводительных вычислительных систем: вместо централизованных моделей управления ресурсами стали использоваться децентрализованные. Использование экономического подхода является естественным продолжением этого развития. Способность ценовых механизмов заменять локальные решения по управлению различными объектами определением глобально эффективных характеристик определяет их значение в организации вычислений в больших системах, таких как Grid или Internet. До последнего времени решения отдельных людей или организаций предоставить ресурсы для Grid в значительной степени мотивировались общественной пользой, призами, забавой, известностью, или совместной выгодой. Это в корне отличается от построения частных Grid систем (но с использованием добровольных ресурсов) или исследовательских проектов, таких как Distributed.net [11], SETI@Home [3], Condor pool [5], GUSTO [12], eGrid [13], World-Wide Grid [14].

Распределенная архитектура Grid показана на рисунке 1 (который взят из [10]).Эта архитектура является достаточно полной и подходит для описания

различных моделей, используемых для распределения и управления ресурсами. Основными компонентами в системе Grid являются:

- Пользователь Grid и приложения (последовательные, параметрические, параллельные или совместные)
- Интерфейсное ПО уровня пользователя – высокоуровневые услуги и инструменты
- Программные окружения
- Брокеры ресурсов
- Интерфейсное ПО ядра Grid (распределение ресурсов и объединение распределенных ресурсов)
- Поставщики ресурсов и услуг Grid

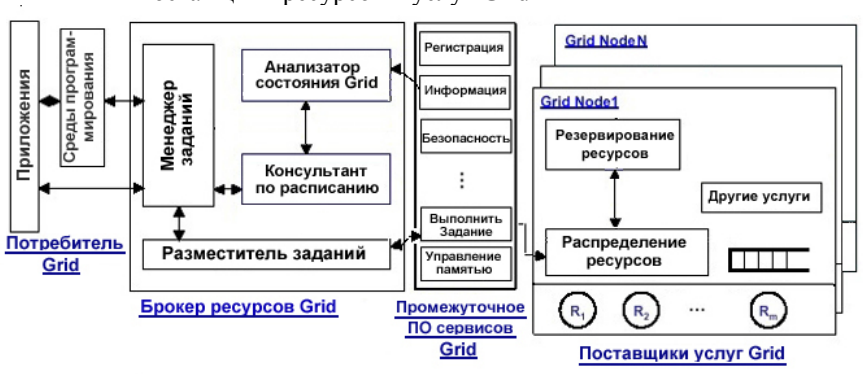

Рис. 1. Архитектура Grid

Двумя главными субъектами в системе Grid являются поставщики и потребители ресурсов. Оба субъекта имеют собственные стратегии. Потребители ресурсов применяют стратегии решения своих прикладных задач в зависимости от требуемого времени и наличного бюджета. Поставщики ресурсов используют стратегию получения наибольшей выгоды от вложенных средств. Владельцы ресурсов стараются максимизировать использование своих ресурсов.

Пользователи имеют возможность выбора поставщиков, которые в наибольшей степени удовлетворяют их требованиям. Если поставщики ресурсов имеют локальных пользователей, они будут стараться получить наилучшую отдачу от простаивающих ресурсов. Для этого системы Grid должны использовать утилиты и механизмы, которые позволяют поставщикам и потребителям задавать свои требования.

Потребители ресурсов Grid взаимодействуют с брокерами для задания своих требований, таких как сумма, которую они могут потратить на решение данной задачи и предел времени, в течение которого они хотят получить результат.

Они также должны иметь возможность выбирать между этими двумя группами требований. Поставщики ресурсов нуждаются в утилитах для задания ценовых политик и механизмов, которые помогут им максимизировать прибыль и утилизацию ресурсов.

Брокер ресурсов Grid выступает в роли соединительного звена между пользователями и поставщиками ресурсов Grid, используя интерфейсное ПО ядра для обеспечения услуг. Он отвечает за поиск ресурсов, их выбор, привязку программного обеспечения и данных к выбранным ресурсам, инициализацию вычислений, адаптацию к изменениям в системе. Брокер выступает для пользователей Grid как единый ресурс. Брокер состоит из следующих компонент:

Система управления задачами. Отвечает за прохождение задачи пользователя. Взаимодействует с планировщиком для генерации расписания, производит создание задачи, поддержку статуса задачи, взаимодействие с клиентами и пользователями, планировщиком и распределителем.

Система составления расписаний. Отвечает за поиск ресурсов (используя систему исследования Grid), выбор ресурса и распределение задания (генерация расписания), чтобы убедиться в том, что требования пользователя удовлетворены.

Система исследования Grid. Отвечает за обнаружение ресурсов, взаимодействуя с информационным сервером Grid.

Система установки задачи. Отвечает за активацию выполнения задачи на выбранном ресурсе и периодическое обновление статуса выполняемой задачи.

В данной статье рассматривается проблема оптимизации распределения параллельных задач в таких системах. Рассматривается двухуровневая иерархия: на первом уровне несколько независимых брокеров распределяют вычислительные задачи на кластеры в соответствии с некоторым критерием. На втором уровне каждый кластер распределяет задачи, присвоенные ему локальным планировщиком. Такие локальные планировщики могут использовать эвристики, основанные на алгоритмах упаковки в полосу. Эффективность предлагаемого распределения проверяется с помощью моделирования и численных экспериментов.

В данной статье не рассматриваются детали реализации брокера и другие задачи, решаемые брокером, такие как поиск и выбор ресурсов. Главной задачей данной статьи является сравнение различных стратегий распределения ресурсов и определение их эффективности.

Статья организована следующим образом. В главе 2 рассматриваются компоненты системы. В главе 3 – различные стратегии распределения задач. Результаты численных экспериментов представлены в главе 4.

2. Кластеры и описание задач

Брокер ресурсов ответствен за выбор ресурсов и распределение задач таким образом, чтобы удовлетворить все требования пользователей и

оптимизировать общее время выполнения и стоимость используемых ресурсов. Цель состоит в определении эффективности такого распределения задач при использовании различных критериев.

В состав рассматриваемой системы входят следующие компоненты.

Кластеры. Процессоры одного кластера одинаковы, однако процессоры различных кластеров могут иметь различные объемы памяти и производительность. Таким образом, кластер C_i можно охарактеризовать количеством процессоров m_i, объемом памяти каждого процессора S_i и производительностью процессоров r_i по отношению к стандартному процессору. Стоимость K_i характеризует каждый кластер и равна стоимости единицы времени одного процессора этого кластера. Каждый кластер имеет очередь задач, которые должны быть выполнены.

Описание задач. Задачи T_j являются независимыми – во время их выполнения не происходит коммуникаций между задачами, что позволяет выполнять задачи в любом порядке. Мы предполагаем также, что каждая задача может быть решена с использованием только одного кластера (нет необходимости в совместных вычислениях на нескольких кластерах). Каждая задача может быть описана количеством требуемых процессоров p_j, максимальным временем выполнения t_j (на p_j стандартных процессорах), требованиями к памяти M_j и предельным временем. Когда время выполнения задачи T_j превышает t_j задача T_j должна быть прервана и отправлена обратно брокеру. Стоимость (или важность) задачи I_j определяется как функция от количества процессоров p_j и времени $t(T_j)$.

Брокеры. Каждый брокер получает задания от пользователей и распределяет задачи между кластерами учитывая информацию о задачах и текущем состоянии кластеров. Мы рассматриваем следующий сценарий. Брокер анализирует запрос на выполнение задачи и рассылает его всем кластерам, которые могут его выполнить. Каждый кластер возвращает время до которого задача может быть выполнена. Брокер принимает решение на основе этой информации и выбирает кластер, который удовлетворяет предельному времени выполнения задачи и имеет наименьшую стоимость (или использует другой критерий, например, минимальное время выполнения – *MCT*). Затем брокер заключает «контракт» с кластером, посылая кластеру описание задачи.

Таким образом, иерархическая система распределения задач имеет два уровня:

1) распределение задач по кластерам
2) распределение многопроцессорных задач на кластере

Первый уровень обслуживается брокерами, а второй – кластерами (локальными планировщиками кластеров). Рассматриваются различные алгоритмы для этих уровней и оценивается эффективность системы в целом.

Так как существует несколько брокеров, работающих одновременно, они распределяют задачи по кластерам в режиме транзакций. Это означает, что существует общая очередь запросов брокеров, где все брокеры, желающие распределить задачи, оставляют свои запросы. Когда транзакция брокера

завершается, следующий брокер в очереди получает возможность распределить свои задачи. Время каждой транзакции ограничено, например, константой K, равной количеству задач, которые брокер может разместить (в общем случае K=1). Такое ограничение исключает ситуацию, когда один брокер может распределить много задач, тогда как другие должны ждать.

3. Стратегии распределения

В данной главе рассматриваются основные понятия теории распределения параллельных задач. Затем описываются некоторые специфические особенности распределения задач в окружении Grid, усложняющие задачу распределения. Наконец, описываются различные стратегии распределения, которые будут рассмотрены в следующей главе.

3.1. Распределение параллельных задач

Время выполнения параллельной задачи может зависеть от числа используемых процессоров и, в общем случае, может быть определено ассоциативной функцией $f_j : D \to R^+$, где D – множество возможных процессоров для задачи j. Существующие исследования по теории распределения для мультипроцессоров были сфокусированы на минимизации максимума времен выполнения C_{max}, суммы времен выполнения ΣC_j и пропускной способности системы (числа задач, завершенных до предельного времени выполнения). Как правило, распределение многопроцессорных задач является NP-полной задачей [10]. Тем не менее, было предложено несколько различных приближенных алгоритмов. В некоторых случаях распределение параллельных задач может рассматриваться как задача упаковки в полосу, где множество прямоугольников (соответствующих задачам) должно быть упаковано в полосу с ограниченной шириной и неограниченной высотой.

В [4] было показано, что асимптотическая эффективность эвристики "левый нижний угол" равна 3 (оценка достигается, когда прямоугольники отсортированы обратно пропорционально их ширине). В [6] было показано, что некоторые алгоритмы, где прямоугольники расположены по «полкам», используя эвристику одномерной бинарной упаковки, имеют асимптотическую эффективность 2.7 (оценка достигается, когда прямоугольники отсортированы по уменьшению их высоты). Асимптотическая эффективность лучших эвристик была в дальнейшем уменьшена до 2.5 [15], затем до 4/3 [12] и, наконец, до 5/4 [3]. Последние улучшения распределения параллельных задач представлены в [10, 12, 13].

Критерий пропускной способности был исследован в основном в классической теории распределения [3], при этом максимизируется количество ранних задач $(w_j)U_j$, где $U_j=1$ если задача T_j завершилась до предельного времени выполнения $d(T_j)$ и $T_j = 0$. В мультипроцессорных системах хорошие алгоритмы известны только для частных случаев (времена выполнения одинаковы и количество процессоров p_j для каждой задачи T_j фиксировано) [16].

Несмотря на большое количество теоретических и экспериментальных результатов относительно алгоритмов распределения, интересны способы распределения параллельных задач несколькими независимыми брокерами, что делает проблему более сложной и отличающейся от классических задач распределения.

3.2. Специфика распределения задач в окружении Grid

Просуммируем вкратце некоторые особенности распределения задач в окружении Grid.

1) различные представления задач

Различные пользователи могут использовать различные представления задач и различные ограничения. Например, одно приложение может потребовать завершения до предельного времени выполнения, другие могут потребовать ограниченную стоимость или выполнение на фиксированном кластере и т.д.

2) существует поток задач

Брокер должен распределять задачи, не зная всего множества задач. Это похоже на распределение в реальном времени, но брокер имеет возможность предварительно набрать некоторое количество задач для улучшения качества распределения.

3) нечеткое предполагаемое время выполнения

Брокеру известно только максимальное время выполнения задачи (а не реальное время ее выполнения). Таким образом, брокер принимает решение о размещении задачи, основываясь на неполной информации о точном числе требуемых процессоров и верхней границе времени выполнения.

4) динамическое поведение ресурсов

В окружении Grid вычислительные ресурсы могут изменять производительность, коммуникационные соединения могут изменять пропускную способность и т.д.

5) *множество различных критериев оптимизации (общая производительность, цена и т.д.)*

Различные критерии могут быть выбраны как целевая функция для определения эффективности политики распределения: общее время выполнения, цена, пропускная способность и т.д.

3.3. Сравнение стратегий распределения

Рассмотрим n задач, m кластеров, общее предельное время выполнения T. Пусть t_j – время выполнения задачи j на p_j процессорах и m_i – число процессоров кластера i. В качестве целевой функции используется *плотность распределения*, которая определяется как сумма стоимостей всех задач которые завершились до предельного времени выполнения T деленное на общее число процессоров всех кластеров умноженное на T. Введем более формальное определение плотности распределения. Для данного отношения l: $\{1,\ldots,n\} \rightarrow \{1,\ldots,m\}$ задач к кластерам допустим, что E_l обозначает все задачи, завершенные до предельного времени выполнения. Пусть I_j определяет стоимость задачи j, где $I_j = p_j t_j$. Затем допустим

$$COST = \sum_{j \in E_l} I_j$$

и плотность распределения определяется как

$$I = \frac{COST}{T \cdot \sum_i m_i}$$

Таким образом, плотность распределения определяется как взвешенная целевая функция пропускной способности.

Как мы уже упоминали выше, каждому кластеру j соответствует некоторая стоимость K_j. Положим K_j равной отношению числа процессоров кластера j к числу процессоров наименьшего кластера.

Напомним, что брокер анализирует запрос на вычисление задачи и посылает его всем кластерам, которые могут его выполнить. Каждый кластер возвращает время, до которого задача может быть выполнена. Брокер принимает решение на основе этой информации и выбирает кластер, который удовлетворяет предельному времени выполнения задачи и имеет наименьшую стоимость (или использует другой критерий, например, минимальное время выполнения – *MCT*). Затем брокер заключает «контракт» с кластером, посылая кластеру описание задачи.

Исследуем эффективность эвристик брокера, основанных на:

1) распределении задачи на самый дешевый кластер, который удовлетворяет предельному времени выполнения
2) распределении задачи на кластер, который предоставляет наименьшее время выполнения задачи

Кроме того, рассмотрим влияние сортировки задач по ширине на целевую функцию

Во всех случаях кластер (локальный планировщик) размещает задачу, основываясь на эвристике "левый нижний угол" (в порядке поступления задач). В случаях CHEAP-SORT, MCT-SORT эвристики используют препроцессорную сортировку задач по уменьшению их ширины (ширина задачи j это число требуемых процессоров p_j).

Исследуем зависимость эффективности распределения от соотношения больших, средних и маленьких задач. Стоимость распределения определяется как

$$\sum_{j=1}^{m} K_j Q_j$$

где $Q_j = \sum_{i \in l^{-1}(j)} I_i$ и $l^{-1}(j)$ определяет число задач, распределенных на кластер j.

4. Численные эксперименты

Была спроектирована среда моделирования для проверки упомянутых стратегий распределения. В данной главе представлены результаты численных экспериментов. Предполагается, что время выполнения известно точно и что все процессоры кластеров имеют одинаковую производительность.

Рассмотрим две системы кластеров. Первая состоит из 6 кластеров (различного размера) и одного брокера. Два кластера являются большими (по 512 процессоров каждый), два имеют средний размер (по 128 процессоров) и два маленьких (по 32 процессора). Вторая система содержит 4 кластера (по 16 процессоров), 2 кластера (по 32 процессора), один кластер содержит 64 процессора, один – 128 процессоров и один – 512 процессоров.

Для первой системы кластеры делятся на три типа согласно количеству процессоров: большие, средние и маленькие. Задачи разделяются похожим способом в соответствии с требуемым количеством процессоров.

Предположим, что каждая задача (например, *j*) имеет случайное время выполнения между 10 и 300 секундами на p_j процессорах, причем *L%* задач соответствуют большим задачам ($256<p_j\leq512$), *M%* – средним задачам ($128<p_j\leq256$), а *S=100-M-L* это процент маленьких задач ($0<p_j\leq128$). Общее число задач во всех экспериментах равно 1000.

Результаты для первой системы приведены в таблицах 1-4. Во всех примерах *T* является общим предельным временем выполнения. При этом, MCT – общая плотность размещения для MCT-эвристики, а CHP – суммарная плотность размещения для CHP-эвристики.

	Число задач	**Плотность распределения**		**Стоимость распределения**	
		CHP	**MCT**	**CHP**	**MCT**
Все задачи	1000	0,7625	0,7828	6,2194E7	6,4130E7
маленькие	400	0,1166	0,1328	0,3262E7	0,4604E7
Средние	300	0,2256	0,2324	1,2691E7	1,3583E7
Большие	300	0,4203	0,4175	4,6241E7	4,5942E7

Таблица 1 Неотсортированная очередь. Кластеры имеют размеры: 128, 128, 256, 256, 512, 512 процессоров.

	Число задач	**Плотность распределения**		**Стоимость распределения**	
		CHP	**MCT**	**CHP**	**MCT**
Все задачи	1000	0,8475	0,8435	7,0046E7	6,9760E7
маленькие	400	0,1205	0,1181	0,3528E7	0,3406E7
Средние	300	0,2533	0,2517	1,4393E7	1,4230E7
Большие	300	0,4737	0,4737	5,2125E7	5,2125E7

Таблица 2 Отсортированная очередь. Кластеры имеют размеры: 128, 128, 256, 256, 512, 512 процессоров.

	Число задач	**Плотность распределения**		**Стоимость распределения**	
		CHP	**MCT**	**CHP**	**MCT**
Все задачи	1000	0,8042	0,8434	5,1061E7	5,3517E7
маленькие	500	0,1863	0,1985	0,8394E7	0,7632E7
Средние	400	0,4223	0,4417	2,6191E7	2,8768E7
Большие	100	0,1955	0,2031	1,6475E7	1,7117E7

Таблица 3 Неотсортированная очередь. Кластеры имеют размеры: 128, 128, 256, 256, 512, 512 процессоров.

	Число задач	**Плотность распределения**		**Стоимость распределения**	
		CHP	**MCT**	**CHP**	**MCT**
Все задачи	1000	0,8441	0,8460	5,2175E7	5,2392E7
маленькие	500	0,1250	0,1317	0,2640E7	0,2903E7
Средние	400	0,4457	0,4408	2,6501E7	2,6457E7
Большие	100	0,2734	0,2734	2,3033E7	2,3033E7

Таблица 4 Отсортированная очередь. Кластеры имеют размеры: 128, 128, 256, 256, 512, 512 процессоров.

В таблицах 5-6 представлены результаты для второй системы, которая состоит из 4 кластеров по 16 процессоров, 2 кластеров по 32 процессора, одного кластера из 64 процессоров, одного кластера из 128 процессоров и одного кластера из 512 процессоров. Каждая задача имеет случайное время выполнения между 10 и 300 секундами. Все задачи маленькие ($0<p_j\leq128$).

	Число задач	**Плотность распределения**		**Стоимость распределения**	
		CHP	**MCT**	**CHP**	**MCT**
маленькие	1000	0,9004	0,9219	8,9449E7	9,0216E7

Таблица 5 Неотсортированная очередь. Кластеры имеют размеры: 16, 16, 16, 16, 32, 32, 64, 128, 512 процессоров.

	Число задач	Плотность распределения		Стоимость распределения	
		CHP	MCT	CHP	MCT
маленькие	1000	0,9447	0,9169	9,0501E7	8,9466E7

Таблица 6 Отсортированная очередь. Кластеры имеют размеры: 16, 16, 16, 16, 32, 32, 64, 128, 512 процессоров.

5. Заключительные замечания

Вычислительные эксперименты показывают, что предложенная иерархическая система независимых брокеров может производить эффективное распределение, даже если она использует сравнительно простые алгоритмы распределения. Достигнутая плотность распределения составляет как минимум 75%. Показано, что препроцессорный шаг, когда задачи сортируются по ширине, улучшает плотность распределения на 10%. Дополнительным фактором, влияющим на качество распределения, является соотношение между большими и дешевыми задачами: чем дешевле задачи, тем больше эффективность.

Используя лучшие алгоритмы упаковки (для кластеров) и позволяя задачам мигрировать с одного кластера на другой, брокер может увеличить эффективность распределения. Это является предметом дальнейших исследований. Таким образом, интересно рассмотреть следующие вопросы:

1) *как нечеткие значения времени выполнения влияют на эффективность?*

Задачи имеют только оценочное время выполнения [11]. Интересно предположить, что время выполнения каждой задачи – случайная величина, распределенная в некотором интервале. При таких предположениях ожидаемое время выполнения становится «нечетким».

2) *насколько использование лучших алгоритмов упаковки может улучшить эффективность распределения?*

3) *Как миграция задач с одного кластера на другой влияет на эффективность распределения?*

В данном случае задача может быть возвращена брокеру (возможно с некоторым штрафом), если кластер получит больше выгодных задач от других брокеров. Брокер может перераспределить возвращенную задачу на другой кластер.

Интересно исследовать также случай “гибких” задач, когда число процессоров для задачи не задается заранее пользователем, а может выбираться брокером.

Литература

1. P. Brucker, Scheduling Algorithms, Springer Ferlag (1998), 217-218.
2. L. Hluchy, V.D. Tran, D. Froehlich, and W. Castaings, Methods and Experiences of Parallelizing Flood Models, The 10th EuroPVM/MPI conference. LNCS 2840. Sept. 2003, Venice. pp. 677-681.
3. B.S. Baker, D.J. Brown and H.P. Katseff, A 5/4 algorithm for two-dimensional packing, J. of Algorithms, 1981, v. 2, pp. 348-368.
4. B.S. Baker, E.J. Coffman and R.L. Rivest, Orthogonal packings in two dimensions, SIAM J. Computing, 1980, v. 9, pp. 846-855.
5. R. Buyya, D. Abramson, J. Giddy, An Economy Driven Resource Management Architecture For Global Computational Power Grids, International Conference on Parallel and Distributed Processing Techniques and Applications, 2000.
6. E.J. Coffman, M.R. Garey, D.S. Johnson and R.E. Tarjan, Performance bounds for level-oriented two-dimensional packing algorithms, SIAM J. Computing, 1980, v. 9, pp. 808-826.
7. The Global Grid Forum. http://www.gridforum.org
8. Foster, C. Kesselman, editors. The Grid: Blueprint for a future computing infrastructure, Morgan Kaufmann, San Fransisco, 1999.
9. K. Czajkowski, S. Fitzgerald, I. Foster, C. Kesselman, Grid Information Services for Distributed Resource Sharing, 10th IEEE International Symposium on High-Performance Distributed Computing, 2001.
10. M. Drozdowski, Scheduling multiprocessor tasks - an overview, European J. of Oper. Research, 1996, v. 94, pp. 215-230.
11. S. Orlando, P. Palmerini, R. Perego, F. Silvestri, Scheduling high performance data mining tasks on a data Grid environment, Euro-Par 2002, LNCS 2400, Springer-Ferlag Berlin Heidelberg 2002, pp. 375-384.
12. K. Jansen, Scheduling malleable parallel tasks: an asymptotic fully polynomial-time approximation scheme, Proc. European Symposium on Algorithms, ESA, 2002.
13. C. Kenyon and E. Remila, A near optimal solution to a two-dimensional cutting stock problem, Mathematics of Operations Research, 25 (2000), 645-656.
14. W. Ludwig and P. Tiwari, Scheduling malleable and nonmalleable parallel tasks, Proc. 5th ACM-SIAM Symposium on Discrete Algorithms, SODA (1994), 167-176.
15. D.D.Sleator, A 2.5-times optimal algorithm for bin packing in two dimensions, Inf. Processing Letters, 1980, v. 10, pp. 37-40.
16. A.V. Fishkin, G. Zhang, On maximizing the throughput of multiprocessor tasks, Proceedings 27th International Symposium on Mathematical Foundations of Computer Science (MFCS'02), Warszawa - Otwock, Poland, LNCS 2420, Springer Verlag: Berlin, 2002, 269-279.
17. R. Wolski, Dinamically forecasting network performance using the network weather service, Cluster Computing (1998).
18. Abraham, R. Buyya, B. Nath, Nature's heuristics for scheduling jobs on Computational Grids, International Conference on Advanced Computing and Communications (2000).
19. N. Sample, P. Keyani, G. Wiederhold, Scheduling under uncertainty: Planning for the Ubiquitois Grid, International Conference on Coordination models and languages (2002).